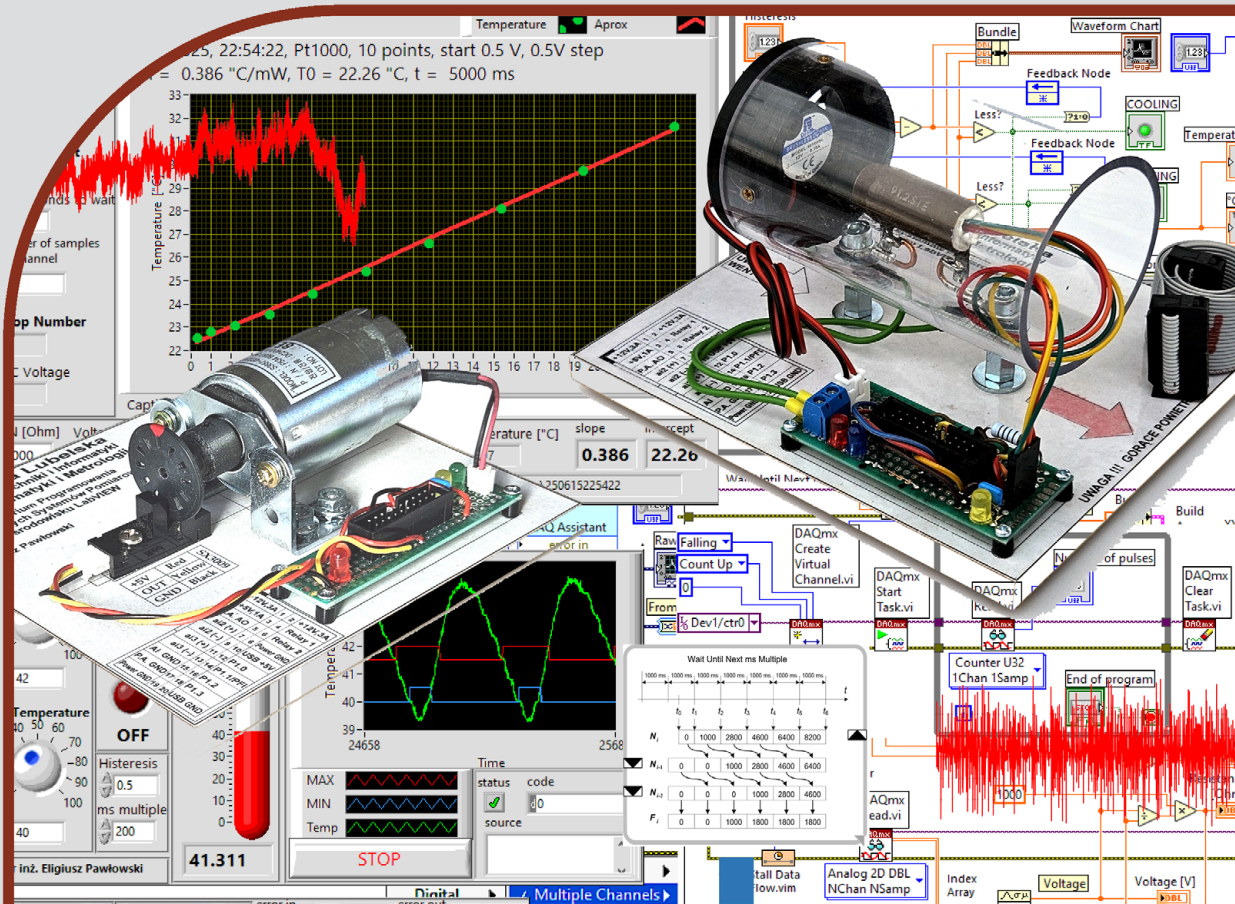
PODRĘCZNIKI

POLITECHNIKA LUBELSKA

# Programowanie systemów pomiarowych

## Laboratorium komputerowych systemów pomiarowych programowanych w środowisku LabVIEW

Eligiusz Pawłowski



**Dla moich wnuczek:**
**Marii, Antoniny i Zuzanny**

**Rada Naukowa Wydawnictwa Politechniki Lubelskiej**

**Przewodnicząca:**
Agnieszka Rzepka

**Dyrektor CINT:**
Katarzyna Weinper

**Przedstawiciele Dyscyplin Naukowych Politechniki Lubelskiej:**
Marzenna Dudzińska
Małgorzata Franus
Arkadiusz Gola
Beata Kowalska
Grzegorz Komarzyniec
Jarosław Latalski
Tomasz Lipecki
Zbigniew Łagodowski
Lucjan Pawłowski
Natalia Przesmycka
Halina Rarot
Magdalena Rzemieniak
Arkadiusz Syta

**Wydawnictwo Politechniki Lubelskiej:**
Magdalena Chołojczyk
Karolina Famulska-Ciesielska
Katarzyna Pełka-Smętek

PODRĘCZNIKI POLITECHNIKI LUBELSKIEJ

# Programowanie systemów pomiarowych

## Laboratorium komputerowych systemów pomiarowych programowanych w środowisku LabVIEW

Eligiusz Pawłowski

WYDAWNICTWO
POLITECHNIKI
LUBELSKIEJ

Lublin 2026

Recenzenci:

prof. dr hab. inż. **Robert Hanus**, Politechnika Rzeszowska im. Ignacego Łukasiewicza
dr hab. inż. **Romuald Maśnicki**, prof. UMG, Uniwersytet Morski w Gdyni

Autor:

dr inż. **Eligiusz Pawłowski**, ORCID: 0000-0001-7039-199X
Politechnika Lubelska – ROR: https://ror.org/024zjzd49

Redaktor prowadzący: **Aleksandra Drozd**
Korekta językowa: **Magdalena Osica, Aleksandra Drozd**
Skład i łamanie: **Katarzyna Pełka-Smętek**
Projekt graficzny serii: **Łukasz Maj**

Publikacja została sfinansowana w ramach programu projakościowego prorektora ds. studenckich: Konkurs na wydanie podręcznika akademickiego lub skryptu.

Ilustracja na okładce została wykonana przez autora. O ile nie wskazano inaczej, wszystkie ilustracje stanowią opracowanie własne autora.

Publikacja wydana za zgodą **Rektora Politechniki Lubelskiej**

**ISBN:** 978-83-7947-681-7 (wersja drukowana)
**ISBN:** 978-83-7947-682-4 (wersja elektroniczna)
**DOI:** 10.35784/9788379476824
**Wydawca:** Wydawnictwo Politechniki Lubelskiej
www.wpl.pollub.pl
ul. Nadbystrzycka 36C, 20-618 Lublin
tel. (81) 538-46-59

WYDAWNICTWO
POLITECHNIKI
LUBELSKIEJ

Książka udostępniona jest na licencji Creative Commons Uznanie autorstwa – na tych samych warunkach 4.0 Międzynarodowe (CC BY-SA 4.0)
**Nakład:** 50 egz.

# Spis treści


**Wstęp** 9

**1. Pomiary wartości chwilowych napięcia przemiennego** 13
- 1.1. Cel ćwiczenia i zadania do wykonania 13
- 1.2. Opis stanowiska laboratoryjnego 14
- 1.3. Procedura włączania i wyłączania stanowiska laboratoryjnego 23
- 1.4. Uruchamianie i testowanie stanowiska 25
- 1.5. Programowanie 38
- 1.6. Opracowanie raportu z realizacji ćwiczenia 62

**2. Pomiary parametrów napięcia w sieci energetycznej** 63
- 2.1. Cel ćwiczenia i zadania do wykonania 63
- 2.2. Opis stanowiska laboratoryjnego 64
- 2.3. Parametry napięcia przemiennego w sieci energetycznej 64
- 2.4. Uruchamianie i testowanie stanowiska laboratoryjnego 70
- 2.5. Programowanie 72
- 2.6. Pomiary częstotliwości i ocena dokładności 96
- 2.7. Opracowanie raportu z realizacji ćwiczenia 98

**3. Pomiary częstotliwości sygnałów cyfrowych** 99
- 3.1. Cel ćwiczenia i zadania do wykonania 99
- 3.2. Opis stanowiska laboratoryjnego 100
- 3.3. Dołączanie sygnału do wejścia licznika impulsów modułu NI USB-6001 105
- 3.4. Uruchamianie i testowanie stanowiska laboratoryjnego 107
- 3.5. Programowanie 110
- 3.6. Badania generatora wzorcowego 130
- 3.7. Opracowanie raportu z realizacji ćwiczenia 132

**4. Badanie silnika DC małej mocy** 133
- 4.1. Cel ćwiczenia i zadania do wykonania 133
- 4.2. Opis stanowiska laboratoryjnego 134
- 4.3. Dołączanie wyjścia przetwornika DAC i wejścia licznika modułu NI USB-6001 141
- 4.4. Uruchamianie i testowanie stanowiska laboratoryjnego 144
- 4.5. Programowanie 148
- 4.6. Badania silnika DC małej mocy 163
- 4.7. Opracowanie raportu z realizacji ćwiczenia 167

**5. Badanie wzmacniacza prądu stałego** 169
- 5.1. Cel ćwiczenia i zadania do wykonania 169
- 5.2. Opis stanowiska laboratoryjnego 170
- 5.3. Przygotowanie stanowiska laboratoryjnego do pracy 177
- 5.4. Uruchamianie i testowanie stanowiska laboratoryjnego 179

5.5. Programowanie 183
5.6. Badania wzmacniacza prądu stałego 199
5.7. Opracowanie raportu z realizacji ćwiczenia 202
**6. Wyznaczanie charakterystyk prądowo-napięciowych dwójników** 205
6.1. Cel ćwiczenia i zadania do wykonania 205
6.2. Opis stanowiska laboratoryjnego 206
6.3. Przygotowanie stanowiska laboratoryjnego do pracy 211
6.4. Uruchamianie i testowanie stanowiska laboratoryjnego 214
6.5. Programowanie 218
6.6. Badania 236
6.7. Opracowanie raportu z realizacji ćwiczenia 243
**7. Układ automatycznej regulacji temperatury** 245
7.1. Cel ćwiczenia i zadania do wykonania 245
7.2. Opis stanowiska laboratoryjnego 246
7.3. Przygotowanie stanowiska laboratoryjnego do pracy 255
7.4. Uruchamianie i testowanie stanowiska laboratoryjnego 257
7.5. Programowanie 261
7.6. Opracowanie raportu z realizacji ćwiczenia 291
**8. Pomiary rezystancji termicznej czujnika temperatury** 293
8.1. Cel ćwiczenia i zadania do wykonania 293
8.2. Opis stanowiska laboratoryjnego 294
8.3. Efekt samonagrzewania czujników termorezystancyjnych 300
8.4. Uruchamianie i testowanie stanowiska laboratoryjnego 306
8.5. Programowanie 311
8.6. Badania 328
8.7. Opracowanie raportu z realizacji ćwiczenia 331
**Bibliografia** 333

## Programowanie systemów pomiarowych. Laboratorium komputerowych systemów pomiarowych programowanych w środowisku LabVIEW

**Streszczenie**

Niniejsze opracowanie stanowi podręcznik do laboratorium komputerowych systemów pomiarowych w środowisku LabVIEW. Materiał obejmuje osiem zadań realizowanych w ramach 30-godzinnego kursu semestralnego, które mogą być prowadzone w trybie stacjonarnym lub zdalnym. Tematyka ćwiczeń koncentruje się na pomiarach parametrów napięcia w sieci energetycznej, częstotliwości, charakterystyk dwójników, a także na badaniu silnika DC, wzmacniacza prądu stałego, czujników oraz układów automatycznej regulacji temperatury. Metodyka opiera się na przyrostowym rozwoju oprogramowania – studenci na każdych kolejnych zajęciach modyfikują i rozbudowują jeden bazowy algorytm, co uczy inżynierskiego podejścia do testowania kodu. Wszystkie zadania są zorientowane na pracę z rzeczywistym sprzętem, wykorzystując moduł akwizycji danych NI USB-6001 wraz z autorskimi panelami rozszerzeń, co zapewnia bezpośredni, fizyczny kontakt z obiektami sterowanymi przez program. Skrypt łączy analizę gotowych przykładów z zadaniami o charakterze projektowym, a jego struktura pozwala studentom o zróżnicowanym stopniu zaawansowania na maksymalne wykorzystanie czasu laboratoryjnego. Publikacja jest adresowana do studentów kierunków technicznych oraz inżynierów praktyków.

**Słowa kluczowe:** LabVIEW, systemy pomiarowe, akwizycja danych, NI USB-6001, wirtualne przyrządy pomiarowe, moduły DAQ

## Programming of measurement systems. Laboratory on computer-based measurement systems programmed in the LabVIEW environment

**Summary**

This academic textbook is a laboratory guide for programming computer-based measurement systems in the LabVIEW environment. The material covers eight topics, delivered in a 30-hour semester course, and is adaptable to both on-campus and remote learning settings. The exercises focus on measuring voltage and power grid parameters, frequency, and two-terminal network characteristics, as well as testing DC motors, DC amplifiers, sensors, and automatic temperature control systems. The methodology is based on incremental software development, in which students modify and expand a single baseline algorithm across subsequent classes, thereby teaching an engineering approach to code testing. All tasks are oriented toward hands-on work with physical hardware, utilising the NI USB-6001 data acquisition module along with custom extension panels, which provide direct physical contact with software-controlled objects. The textbook combines the analysis of ready-made examples with project-based tasks, and its structure ensures maximum utilisation of laboratory time by students of varying skill levels. The publication is addressed to engineering students and practising professionals.

**Keywords:** LabVIEW, measurement systems, data acquisition, NI USB-6001, Virtual Instruments, DAQ modules

# Wstęp

Nauczanie programowania komputerowych systemów pomiarowych na Wydziale Elektrotechniki i Informatyki Politechniki Lubelskiej ma już swoją długą historię. Pierwsze takie zajęcia odbyły się na ówczesnym Wydziale Elektrycznym Politechniki Lubelskiej w semestrze zimowym roku akademickiego 1991/92. Studenci realizowali wtedy zadania w laboratorium na komputerach klasy IBM PC/XT, które pracowały pod systemem operacyjnym DOS. Programy sterujące systemami pomiarowymi tworzone były wówczas w Basicu i Pascalu. Pierwsza udana wersja systemu Windows 3.0 pojawiła się niewiele wcześniej, w 1990 roku. Pierwsza wersja LabVIEW dla systemu Windows została natomiast udostępniona w 1992 roku. Od tamtego czasu środowisko LabVIEW zyskiwało coraz więcej zwolenników. Powoli stawało się światowym standardem programowania komputerowych systemów pomiarowych. Od pierwszych zajęć nasze laboratorium ulegało ciągłym modyfikacjom, a udział środowiska LabVIEW w tematyce przedmiotu stawał się coraz większy. Gwałtowna zmiana nastąpiła w październiku 2020 roku, kiedy to uczelnie musiały niespodziewanie przejść na zdalny tryb nauczania w związku z pandemią COVID-19. Dla zajęć laboratoryjnych było to szczególnie trudne wyzwanie. Wtedy właśnie powstała koncepcja obecnej wersji laboratorium programowania komputerowych systemów pomiarowych. Opiera się ona na kilku założeniach. Po pierwsze, cały program laboratorium bazuje na

środowisku LabVIEW. Po drugie, wszyscy studenci pracują na takich samych stanowiskach i wykonują te same ćwiczenia w tym samym czasie. po trzecie zaś, każde kolejne zadanie dotyczy innego problemu pomiarowego, ale wykorzystuje programy opracowane na wcześniejszych zajęciach.

Taka koncepcja prowadzenia zajęć w laboratorium programowania systemów pomiarowych wynika z mojego osobistego doświadczenia jako programisty i metrologa zarazem. Zgodzą się ze mną chyba również inni praktykujący informatycy, że przystępując do pracy nad nowym programem, rzadko kiedy tworzy się go od początku. W pierwszej kolejności analizuje się wcześniej stworzone programy pod kątem możliwości zastosowania ich w nowym projekcie. Takie podejście ma swoje głębokie praktyczne uzasadnienie. Największym problemem przy tworzeniu nowego programu są nieuniknione błędy. Każdy, kto próbował programować, wie, że nowy program zazwyczaj zaraz po napisaniu nie działa wcale albo działa źle. Wtedy rozpoczyna się żmudny proces testowania programu i poszukiwania błędów. Dlatego zawsze lepiej i szybciej jest próbować zmodyfikować posiadany, przetestowany, dobrze działający program i przystosować go do realizacji nowych zadań, niż tworzyć nowy program od podstaw. W ten sposób jest również zorganizowane laboratorium opisane w tym skrypcie. Zaplanowane zadania wydają się skomplikowane i trudne do realizacji. Diagramy programów sprawiają niekiedy wrażenie zawiłych i niezrozumiałych. są to jednak pozory. Studenci zaczynają od bardzo prostego programu, który jest modyfikowany i rozwijany na kolejnych zajęciach. Tym sposobem koniec semestru przynosi studentom satysfakcję z tego, jak skomplikowane zadania udało się zrealizować.

Podstawowym wyposażeniem stanowiska laboratoryjnego potrzebnego do przeprowadzenia zajęć jest moduł pomiarowy NI USB-6001 produkowany przez National Instruments. Na jego bazie został zaprojektowany i zbudowany panel ćwiczeniowy z dodatkowymi modułami rozszerzającymi. Łącznie studenci dysponują sześcioma identycznymi stanowiskami laboratoryjnymi, dzięki czemu zajęcia są realizowane tzw. równym frontem. Optymalna liczebność zespołów pracujących na każdym stanowisku to dwie lub trzy osoby, grupa nie powinna przekraczać 15 osób.

Moduł NI USB-6001 należy do najprostszych układów tego rodzaju, ale mimo to umożliwia realizację bardzo wielu ciekawych zajęć w laboratorium. Często na koniec semestru słyszę od studentów, że na tych zajęciach coś się

dzieje, coś się rusza, kręci, świeci, dmucha, grzeje. Nie ma tu żadnych symulacji na ekranie komputera, jest za to bezpośredni fizyczny kontakt ze sprzętem laboratoryjnym oraz pisanie programów sterujących prawdziwymi obiektami, które można zobaczyć i dotknąć. Studenci potrafią to docenić i chwalą sobie takie zajęcia.

Przedstawione w materiałach zadania są opracowane w taki sposób, że umożliwiają prowadzenie zajęć zarówno laboratoryjnych, jak i projektowych. W części laboratoryjnej zajęć każdy temat jest najpierw omawiany na przykładzie gotowych programów zamieszczonych w materiałach. Przykładowe programy można w niezmienionej postaci przenieść do środowiska LabVIEW, uruchomić i przetestować ich działanie z dołączonymi dodatkowymi modułami rozszerzającymi. Nie są to jednak zadania zamknięte; każde z nich można rozwiązać na różne sposoby, a także rozwijać i wzbogacać o kolejne zagadnienia o charakterze projektowym.

Na zajęciach organizacyjnych przede wszystkim przeprowadzane jest obowiązkowe w każdym laboratorium szkolenie BHP. Następnie studenci są zapoznawani z programem i harmonogramem zajęć, ogólnymi zasadami realizacji kolejnych tematów oraz sposobem oceniania i warunkami otrzymania pozytywnej oceny z przedmiotu. Podręcznik opisuje ćwiczenia z zakresu ośmiu tematów, a każdy z nich zaplanowany został do realizacji w czasie trzech godzin. Na koniec semestru omawiane są raporty przygotowane przez studentów, wystawiane są oceny końcowe i dyskutowane problemy, które pojawiły się podczas realizacji poszczególnych tematów. Łącznie z zajęciami organizacyjnymi i zaliczeniowymi otrzymujemy zatem 30 godzin w semestrze.

Zadania zaplanowane do realizacji w ramach każdego z tematów mogą wydawać się bardzo obszerne, wymagające poświęcenia im więcej niż planowe trzy godziny. To w pewnym stopniu prawda, nie wszyscy studenci będą w stanie zrealizować w dostępnym czasie każdy z tematów. Odwieczny dylemat każdego nauczyciela polega na ustaleniu dla jakich uczniów przygotować zajęcia. Dla tych najlepszych, dla przeciętnych, czy dla najsłabszych? W szkole podstawowej i średniej jest to bardzo trudne do rozstrzygnięcia, ponieważ każde dziecko powinno mieć stworzone równe szanse w edukacji. Jednak na studia powinna dostawać się najlepsza młodzież, przygotowana i przede wszystkim zainteresowana wybranym kierunkiem. Zakładam, że właśnie takie osoby trafiają na politechnikę. Jednocześnie podczas zajęć wiele razy spotykałem się z sytacją,

gdy grupa dobrych studentów bardzo szybko realizowała cały program ćwiczenia i kończyła przed czasem. W ten sposób została przez nich bezpowrotnie stracona szansa na zdobycie dodatkowej porcji wiedzy. Dlatego w tym skrypcie program każdego z ćwiczeń zaplanowałem pod kątem najlepszych. Dzięki temu każdy student ma możliwość w pełni wykorzystać czas przeznaczony na zajęcia w laboratorium. Słabiej przygotowani studenci również będą mieli wypełniony czas, ale mogą nie zdążyć zrealizować całego programu ćwiczenia, co znajdzie swoje odzwierciedlenie w ocenie końcowej.

Jestem przekonany, że podręcznik może być ciekawy i przydatny również dla innych osób zainteresowanych tą tematyką, w tym także dla studentów Politechniki Lubelskiej, którzy nie mają w programie studiów tego rodzaju zajęć. Materiały te mogą być inspiracją i pomocą podczas realizacji projektów inżynierskich i prac dyplomowych. Dużo satysfakcji sprawiłaby mi również informacja, że podręcznik wykorzystywany jest w czasie zajęć na innych uczelniach.

Wszystkie zamieszczone w materiałach programy zostały przetestowane w laboratorium i sprawdzone pod względem poprawności działania. Starałem się również opisać proces programowania na tyle szczegółowo, aby nie sprawiał on problemów nawet studentom gorzej znającym środowisko LabVIEW. Być może jednak użytkownicy dostrzegą potrzebę dokładniejszego opisania wybranych zagadnień. Dlatego będę wdzięczny za wszelkie uwagi dotyczące tekstu oraz struktury programów. Z przyjemnością i zainteresowaniem zapoznam się również z propozycjami innych ciekawych zagadnień, które mogłyby uatrakcyjnić przyszłe wersje tej publikacji. Proszę przesyłać wszelkie uwagi na adres: e.pawlowski@pollub.pl lub elek.pawlowski@gmail.com.

*Eligiusz Pawłowski*

# 1. Pomiary wartości chwilowych napięcia przemiennego

## 1.1. Cel ćwiczenia i zadania do wykonania

Celem ćwiczenia 1 jest zapoznanie studentów z problematyką programowania układu pomiarowego w środowisku LabVIEW z wykorzystaniem przetwornika analogowo-cyfrowego ADC (ang. *Analog to Digital Converter*), który jest elementem składowym uniwersalnego modułu akwizycji danych pomiarowych NI USB-6001.

Program laboratorium obejmuje następujące zagadnienia:

- zapoznanie się z konstrukcją i wyposażeniem stanowiska laboratoryjnego,
- instalowanie fizycznego modułu NI USB-6001 w systemie komputerowym,
- tworzenie symulowanego modułu NI USB-6001 w programie NI-MAX,
- testowanie modułu NI USB-6001 w programie NI-MAX,
- testowanie przetwornika ADC w programie NI-MAX,
- konfigurowanie modułu NI USB-6001 do konwersji analogowo-cyfrowej,
- tworzenie wykresów z wartości chwilowych zmierzonych napięć,
- wykonywanie obliczeń na chwilowych wartościach próbkowanego napięcia,
- generowanie symulowanych przebiegów napięć przemiennych do testowania poprawności działania programu.

Zadania realizowane w trakcie ćwiczenia polegają na przygotowaniu odpowiednich programów w środowisku LabVIEW, uruchomieniu ich, sprawdzeniu poprawności ich działania, usunięciu błędów, wykonaniu przykładowych pomiarów i sporządzeniu raportu.

W ramach ćwiczenia 1 studenci opracowują trzy programy:

1. Pierwszy program do jednokrotnego pobierania z przetwornika analogowo-cyfrowego tablicy wartości chwilowych napięcia przemiennego z zadanymi parametrami próbkowania.
2. Drugi program do próbkowania ciągłego w pętli While i z prostymi obliczeniami na wartościach chwilowych próbkowanego napięcia.
3. Trzeci program z możliwością symulowania próbkowania napięcia przemiennego o zadanych parametrach.

Szczegółowe wymagania wstępne stawiane studentom w zakresie wiedzy i umiejętności zawarte są w sylabusie do przedmiotu. Przede wszystkim student powinien znać podstawy programowania w środowisku LabVIEW, omawiane i ćwiczone w ramach wcześniej realizowanych zgodnie z tokiem studiów przedmiotów. W razie potrzeby należy uzupełnić wiedzę w tym zakresie, korzystając z ogólnie dostępnych podręczników [1–4]. Kolejnym ważnym źródłem wiedzy, z którym powinni samodzielnie zapoznać się studenci, jest dokumentacja techniczna modułu pomiarowego NI USB-6001 [5, 6].

## 1.2. Opis stanowiska laboratoryjnego

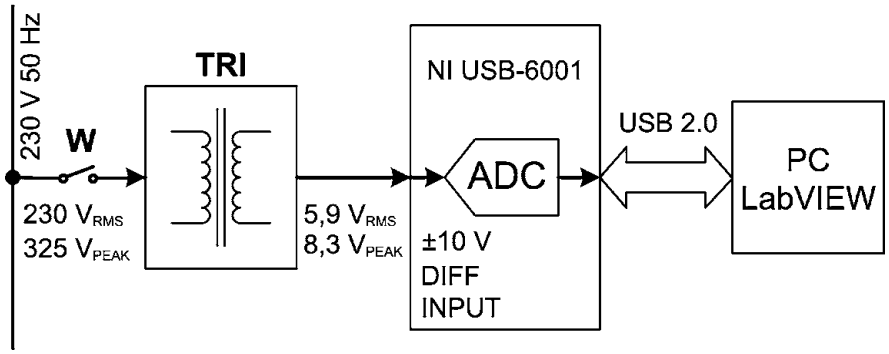
### 1.2.1. Konstrukcja panelu ćwiczeniowego

Stanowisko laboratoryjne wykorzystywane w ćwiczeniu 1 składa się z następujących komponentów:

- komputera PC z zainstalowanym środowiskiem National Instruments LabVIEW,
- modułu akwizycji danych pomiarowych NI USB-6001 firmy National Instruments,
- modułu wejściowego z transformatorem izolacyjnym.

Schemat blokowy stanowiska laboratoryjnego przedstawiono na rycinie 1.1. Moduł akwizycji danych pomiarowych NI USB-6001 jest dołączony do

komputera PC za pomocą interfejsu USB 2.0. Do pomiarów wykorzystano wejście analogowe modułu pracujące w trybie różnicowym DIFF na zakresie ±10 V. Do tego wejścia dołączony jest moduł TRI, zawierający transformator izolacyjny i rezystancyjny dzielnik napięcia, który obniża napięcie sieciowe ~230 V/50 Hz do wartości skutecznej ok. 6 V, bezpiecznej i mieszczącej się w zakresie pomiarowym ±10 V modułu NI USB-6001. Dodatkowo transformator zwiększa bezpieczeństwo pracy, zapewniając galwaniczną izolację modułu od sieci energetycznej.




**Ryc. 1.1.** Schemat blokowy stanowiska laboratoryjnego w ćwiczeniu 1

Moduł akwizycji danych pomiarowych NI USB-6001 oraz moduł z transformatorem izolacyjnym są na stałe zainstalowane na panelu ćwiczeniowym specjalnie zaprojektowanym do realizacji wszystkich ćwiczeń w ramach całego programu laboratorium. Panel ćwiczeniowy w postaci płyty HDF o wymiarach 30 cm × 40 cm jest wyposażony w obwody zasilania z sieci ~230 V/50 Hz, załączane łącznikiem kołyskowym typu Rocker z podświetleniem oraz zabezpieczone bezpiecznikiem topikowym 250 V/1 A. Niezbędnych napięć stałych dostarcza impulsowy zasilacz napięcia stałego 12 V/40 W o wydajności prądowej 3,3 A oraz liniowy stabilizator 5 V/1 A. Na panelu ćwiczeniowym zainstalowane są również: wzmacniacz mocy napięcia stałego, kwarcowy generator częstotliwości wzorcowych z wielostopniowym dzielnikiem binarnym, izolowany optycznie moduł z dwoma przekaźnikami elektromechanicznymi oraz 20-stykowe gniazdo IDC umożliwiające dołączanie zewnętrznych modułów ćwiczeniowych z dodatkowym wyposażeniem. Wszystkie elementy wyposażenia panelu ćwiczeniowego oraz moduły ćwiczeniowe będą sukcesywnie dokładnie omawiane w kolejnych ćwiczeniach.

### 1.2.2. Budowa i główne parametry modułu NI USB-6001

Zastosowany w ćwiczeniu moduł akwizycji danych pomiarowych NI USB-6001 należy do serii modułów National Instruments o ogólnym symbolu NI USB-600X, różniących się podstawowymi parametrami, takimi jak rozdzielczość, szybkość i dokładność. Podstawowe parametry modułów 6001, 6002 i 6003 przedstawiono w tabeli 1.1 [5].

Z porównania parametrów wynika, że moduł 6001 jest najprostszą wersją z tej serii modułów. Można również spotkać się z modułami NI USB-6008 oraz 6009, ale są to starsze, nieprodukowane już modele. Producent oferuje modele z serii 600X jako rozwiązania budżetowe, opracowane głównie do zastosowań edukacyjnych. Z powodzeniem mogą być jednak stosowane również w mniej wymagających badaniach naukowych oraz w przemyśle do nadzorowania nieskomplikowanych procesów produkcyjnych.

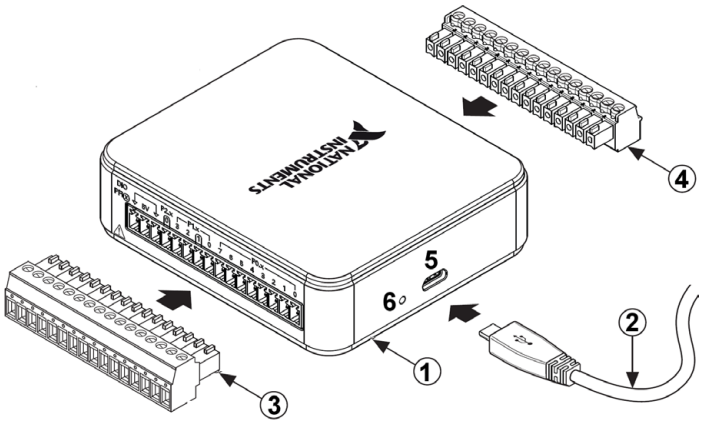
Wygląd zewnętrzny modułów z serii 600X pokazano na rycinie 1.2. Moduły z tej serii nie różnią się wyglądem zewnętrznym, poza symbolem modułu i numerem seryjnym umieszczonymi od spodu obudowy na tabliczce znamionowej. Symbol modułu i jego numer seryjny można również odczytać w programie NI-MAX, co będzie zrobione w czasie realizacji ćwiczenia.

**Tab. 1.1.** Najważniejsze parametry modułów serii NI USB-600X [6]

| Parametr | 6001 | 6002 | 6003 |
|---|---|---|---|
| Rozdzielczość przetwornika ADC | 14 bitów | 16 bitów | 16 bitów |
| Maksymalna szybkość próbkowania ADC (sumarycznie dla wszystkich kanałów) | 20 kS/s | 50 kS/s | 100 kS/s |
| Dokładność ADC w całym zakresie | 6 mV | 6 mV | 6 mV |
| Szumy toru analogowego | 0,7 $mV_{RMS}$ | 0,4 $mV_{RMS}$ | 0,4 $mV_{RMS}$ |
| Rozdzielczość DAC | 14 bitów | 16 bitów | 16 bitów |
| Maksymalna szybkość przetwarzania DAC (symultanicznie dla każdego kanału) | 5 kS/s | 5 kS/s | 5 kS/s |
| Dokładność DAC w całym zakresie | 9,1 mV | 8,6 mV | 8,6 mV |

Moduł zamknięty jest w plastikowej obudowie o wymiarach 86,2 mm × 17,4 mm × 23,6 mm. Na krótszej ściance modułu umieszczone jest złącze interfejsu USB 2.0 w wersji micro-B. Na dłuższych ściankach modułu znajdują się dwa rzędy styków. Moduł posiada łącznie 32 styki pogrupowane w dwóch złączach po 16 styków każde. Sygnały analogowe i cyfrowe są rozmieszczone

po przeciwnych stronach obudowy modułu. Wpływa to korzystnie na zmniejszenie ewentualnego wzajemnego zakłócania się części cyfrowej i analogowej.



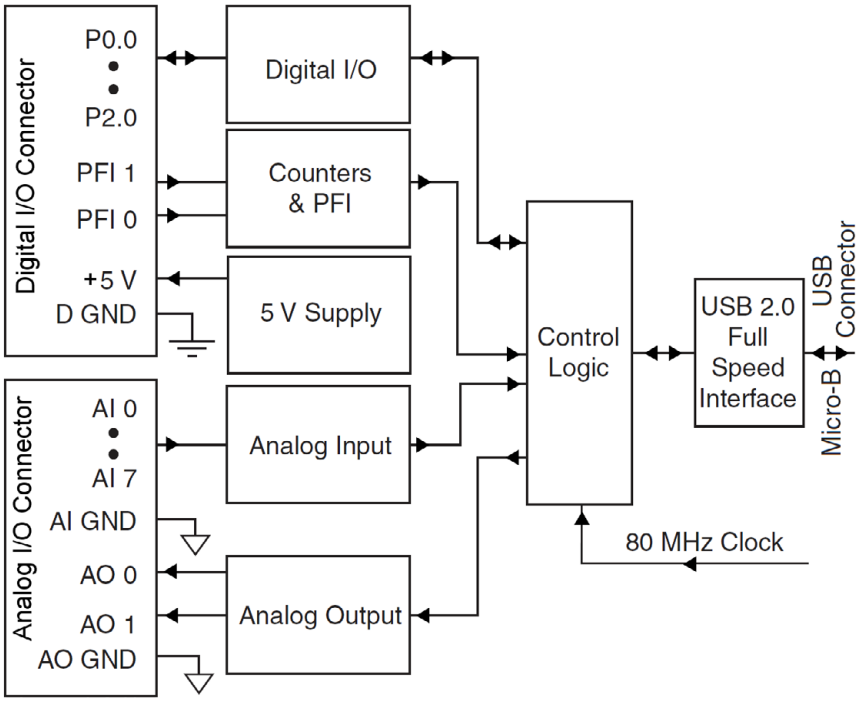

**Ryc. 1.2.** Wygląd zewnętrzny modułów pomiarowych z serii NI-USB 600X [5]: 1 – moduł pomiarowy; 2 – przewód interfejsu USB 2.0; 3 – złącze sygnałów cyfrowych; 4 – złącze sygnałów analogowych; 5 – złącze USB 2.0 Micro-B; 6 – kontrolka LED sygnalizująca status modułu

Obok gniazda USB znajduje się niebieska kontrolka LED sygnalizująca status modułu. Podczas normalnej pracy kontrolka powinna świecić w sposób ciągły. Wszystkie stany kontrolki LED przedstawia tabela 1.2.

**Tab. 1.2.** Status modułu z serii NI USB-600X sygnalizowany kontrolką LED [5]

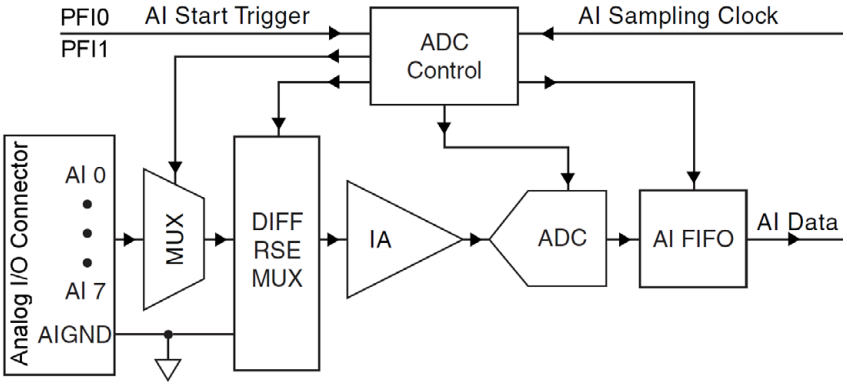
| Stan kontrolki LED | Status modułu NI USB-600X |
|---|---|
| Wyłączona | Urządzenie nie jest dołączone lub w trybie wstrzymania (*suspend mode*) |
| Świeci ciągle | Urządzenie jest dołączone i funkcjonuje normalnie |
| Błyska | Urządzenie napotkało błąd. Odczekaj 10 sekund na próbę naprawienia błędu przez urządzenie. Jeśli LED błyska w dalszym ciągu, odłącz i dołącz ponownie urządzenie. Jeśli błąd powtarza się uporczywie, skontaktuj się z serwisem |

Wewnętrzną budowę modułu NI USB-6001 przedstawiono na schemacie blokowym pokazanym na rycinie 1.3. Na złączu sygnałów analogowych moduł posiada 8 wejść analogowych AI 0–AI 7, dwa wyjścia analogowe AO 0, AO 1 oraz masę AI GND dla wejść analogowych i masę AO GND dla wyjść analogowych. Pomiary napięcia umożliwia jeden przetwornik analogowo-cyfrowy ADC z ośmiokanałowym multiplekserem na wejściu.




**Ryc. 1.3.** Schemat blokowy modułu NI USB-6001 [5]

Na rycinie 1.4 przedstawiono schemat blokowy toru analogowego modułu NI USB-6001, który będzie wykorzystywany w ćwiczeniu 1. Pomiary napięcia realizuje jeden przetwornik ADC z sukcesywną aproksymacją o rozdzielczości 14 bitów i maksymalnej szybkości próbkowania (ang. *sampling rate*) 20 kSPS (ang. *kilosamples per second*).



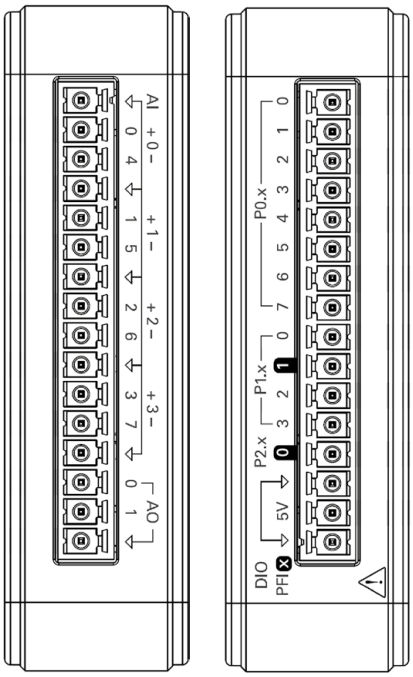

**Ryc. 1.4.** Schemat blokowy toru analogowego modułu NI USB-6001 [5]

Jest to maksymalna szybkość próbkowania podana sumarycznie dla wszystkich aktualnie używanych kanałów (ang. *aggregate*). Oznacza to, że przykładowo, robiąc pomiary jednocześnie w dwóch kanałach, maksymalna szybkość próbkowania wynosi 10 kSPS w każdym z tych dwóch kanałów. Na wejściu toru analogowego znajduje się ośmiokanałowy multiplekser MUX umożliwiający pomiar napięcia w 8 niezależnych kanałach w trybie jednokońcówkowym RSE (ang. *Referenced Single-Ended* – niesymetryczne wejścia jednoprzewodowe odniesione do wspólnej masy) lub w 4 kanałach różnicowych w trybie DIFF (ang. *Differential Mode*). Wybór konfiguracji jednokońcówkowej lub symetrycznej zapewnia dodatkowy multiplekser DIFF/RSE MUX. W trybie pomiaru RSE każdy kanał umożliwia niezależny pomiar napięcia z ośmiu różnych źródeł względem wspólnej analogowej masy wejściowej AI GND. W trybie pomiaru DIFF (dwuprzewodowe wejścia różnicowe) jedno wejście analogowe wykorzystuje dwa kanały wejściowe, co umożliwia pomiary napięcia z czterech różnych źródeł. Zarówno w trybie różnicowym DIFF, jak i w trybie niesymetrycznym RSE możliwy jest pomiar napięć tylko na jednym zakresie ±10 V. W każdym trybie pomiarów napięcie na każdym z zacisków wejściowych nie może przekraczać wartości ±10 V względem masy analogowej AI GND oraz ±10 V względem każdej innej końcówki. Wejścia analogowe są zabezpieczone przed przepięciami do ±30 V względem masy GND przy włączonym zasilaniu i tylko ±20 V przy wyłączonym module. Dlatego nie należy dołączać żadnych sygnałów do wejść analogowych, gdy moduł jest wyłączony. Wejścia są również wyposażone w układy ograniczające napięcie wejściowe (ang. *clipping*) do ±10,5 V. Każde z wejść analogowych ma rezystancję wejściową co najmniej 1 GΩ względem masy AI GND, niezależnie od wybranego trybu pracy (DIFF lub RSE). Wejścia nigdzie niedołączone (tzw. wiszące w powietrzu) mogą osiągnąć dowolną, nieokreśloną wartość z przedziału ±10,5 V. Pozostałe dane techniczne są dostępne w dokumentacji [5, 6]. Rozmieszczenie sygnałów na obu złączach zestawiono w tabeli 1.3, a ich szczegółowy opis przedstawiono w tabeli 1.4.

Napięcia wyjściowe generowane są przez dwa niezależne przetworniki cyfrowo-analogowe DAC. Na złączu sygnałów cyfrowych moduł posiada 13 uniwersalnych dwukierunkowych linii cyfrowych zgrupowanych w 3 porty – P0, P1 i P2. Dwie linie cyfrowe PFI 0, PFI 1 (ang. *Programmable Function Interface*) mają dodatkowe przeznaczenie i mogą być użyte jako wejście licznika impulsów lub źródło impulsów wyzwalania dla przetworników ADC

lub DAC. Konfigurowalne linie wejścia/wyjścia cyfrowego są podłączone do mikrokontrolera sterującego pracą modułu. Dodatkowo na złączu sygnałów cyfrowych wyprowadzone jest napięcie +5 V o wydajności prądowej 150 mA do zasilania zewnętrznych układów. Na tym złączu znajduje się również styk masy D GND dla sygnałów cyfrowych i obwodu zasilania. Wszystkie komponenty modułu są zasilane poprzez interfejs USB.

**Tab. 1.3.** Rozmieszczenie linii sygnałów analogowych (po lewej) i cyfrowych (po prawej) na złączach modułu NI USB-6001. Pogrubioną czcionką zaznaczono styki wykorzystane w ćwiczeniu 1 [5]

| Sygnał | PIN | Złącze analogowe i cyfrowe | PIN | Sygnał |
|---|---|---|---|---|
| AI GND | 1 |  | 17 | P0.0 |
| AI 0 (AI 0+) | 2 | | 18 | P0.1 |
| AI 4 (AI 0–) | 3 | | 19 | P0.2 |
| AI GND | 4 | | 20 | P0.3 |
| **AI 1(AI 1+)** | **5** | | 21 | P0.4 |
| **AI 5 (AI 1–)** | **6** | | 22 | P0.5 |
| **AI GND** | **7** | | 23 | P0.6 |
| AI 2(AI 2+) | 8 | | 24 | P0.7 |
| AI 6 (AI 2–) | 9 | | 25 | P1.0 |
| AI GND | 10 | | 26 | P1.1/PFI 1 |
| AI 3(AI 3+) | 11 | | 27 | P1.2 |
| AI 7 (AI 3–) | 12 | | 28 | P1.3 |
| AI GND | 13 | | 29 | P2.0/PFI 0 |
| AO 0 | 14 | | 30 | D GND |
| AO 1 | 15 | | 31 | +5V/150mA |
| AO GND | 16 | | 32 | D GND |



## 1.2.3. Moduł transformatora izolacyjnego

Wykorzystywany w ćwiczeniu moduł akwizycji danych NI USB-6001 umożliwia pomiar napięcia tylko na jednym zakresie ±10 V, przy czym w każdym trybie pracy wartość chwilowa napięcia na każdym z zacisków wejściowych nie może przekraczać wartości ±10 V względem masy analogowej AI GND oraz ±10 V względem każdej innej końcówki. Natomiast w sieci energetycznej niskiego napięcia znamionowa wartość skuteczna napięcia fazowego wynosi 230 $V_{RMS.}$

Zakładając sinusoidalny kształt tego napięcia (co w przybliżeniu jest spełnione), amplituda tego napięcia jest większa √2 razy od wartości skutecznej, czyli wynosi około 325 $V_{PEAK.}$ Jednocześnie jednak, zgodnie z przepisami [15, 17], napięcie to może zmieniać się o ±10 %, czyli wartość skuteczna może wzrosnąć do 253 VRMS, a amplituda do 358 $V_{PEAK.}$ Dodatkowo w sieci mogą pojawić się krótkotrwałe zakłócenia impulsowe (przepięcia) o znacznej wartości.

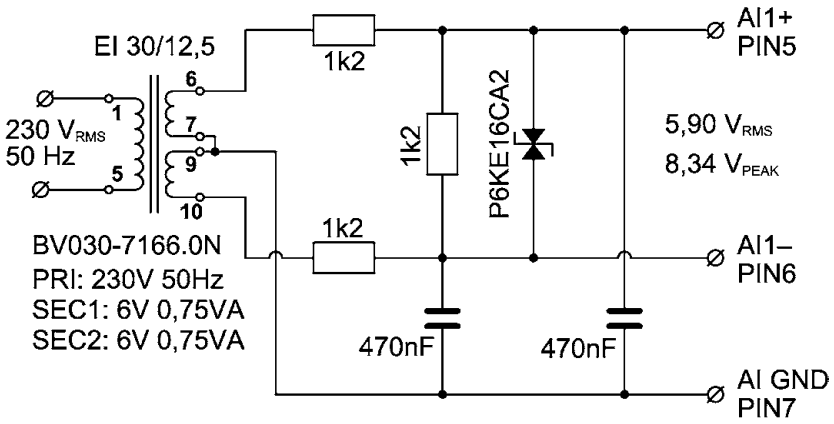
**Tab. 1.4.** Opis linii sygnałowych na złączach modułu NI USB-6001 [5]

| Nazwa sygnału | Odniesienie | Kierunek | Opis |
|---|---|---|---|
| AI GND | — | — | **Analog Input Ground** – masa odniesienia dla wejściowych sygnałów analogowych mierzonych w trybie RSE. |
| AI <0..7> | AI GND | Wejście | **Analog Input Channels 0 to 7** – 8 wejść analogowych w trybie niesymetrycznym jednokońcówkowym RSE lub 4 wejścia analogowe w trybie różnicowym DIFF. W trybie RSE każdy sygnał stanowi oddzielny analogowy kanał wejściowy względem masy AI GND. W trybie DIFF AI 0 i AI 4 stanowią dodatnie (AI 0+) i ujemne (AI 0–) wejścia różnicowego kanału wejściowego AI 0. Następujące pary sygnałów tworzą kolejne różnicowe kanały wejściowe: AI <1, 5>, AI <2, 6> oraz AI <3, 7>. |
| AO GND | – | – | **Analog Output Ground** – masa odniesienia dla wyjściowych sygnałów analogowych. |
| AO <0, 1> | AO GND | Wyjście | **Analog Output Channels 0 and 1** – analogowe sygnały wyjściowe, napięcie wyjściowe z przetworników cyfrowo-analogowych DAC. |
| P0.<0..7> | D GND | Wejście lub Wyjście | **Port 0 Digital I/O Channels 0 to 7** – ośmiobitowy port P0, każdy kanał można konfigurować indywidualnie jako wejście lub wyjście cyfrowe. |
| P1.<0..3> | D GND | Wejście lub Wyjście | **Port 1 Digital I/O Channels 0 to 3** – czterobitowy port P1, każdy kanał można konfigurować indywidualnie jako wejście lub wyjście cyfrowe. Linię P1.1 można skonfigurować jako PFI 1 |
| P2.0 | D GND | Wejście lub Wyjście | **Port 2 Digital I/O Channel 0** – jednobitowy port P2. Linię P2.0 można konfigurować jako wejście lub wyjście cyfrowe lub jako PFI 0. |
| PFI 0, 1 | D GND | Wejście | **Programmable Function Interface or Digital I/O Channels** – wejście licznika zliczającego impulsy lub wejście wyzwalające przetwarzanie w modułach ADC lub DAC. |
| D GND | — | — | **Digital Ground** – masa odniesienia dla sygnałów cyfrowych. |
| +5 V | D GND | Wyjście | **+5 V Power Source** – napięcie zasilania +5 V, prąd do 150 mA. |

Aby więc umożliwić bezpieczny pomiar napięć w sieci 230 $V_{RMS}$ za pomocą modułu NI USB-6001, należy zastosować odpowiedni układ obniżający maksymalną spodziewaną wartość napięcia chwilowego w sieci (±358 V) do

maksymalnej dopuszczalnej wartości na wejściu modułu (±10 V). Dodatkowo należy wejście modułu zabezpieczyć przed przepięciami w sieci oraz ryzykiem porażenia napięciem niebezpiecznym dla życia i zdrowia człowieka.

W tym celu zastosowano układ spełniający powyższe wymagania, którego schemat przedstawiono na rycinie 1.5. Głównym elementem jest zalany żywicą transformator izolacyjny, który obniża napięcie i galwanicznie izoluje moduł pomiarowy od niebezpiecznego napięcia sieciowego. Transformator ma dwa jednakowe uzwojenia wtórne, co pozwala na uzyskanie w pełni symetrycznego układu współpracującego z wejściem modułu w trybie różnicowym. Zapewnia to skuteczne tłumienie zakłóceń, które mogłyby przeniknąć z sieci energetycznej do układu pomiarowego. W tabeli 1.5 zestawiono dane techniczne zastosowanego transformatora [7].




**Ryc. 1.5.** Schemat modułu transformatora izolacyjnego TRI

**Tab. 1.5.** Dane techniczne transformatora izolacyjnego [7]

| Parametr | Wartość parametru |
|---|---|
| Typ | EI 30/12,5 |
| Model | BV 030-7166.0 N |
| Producent | ERA |
| Moc znamionowa | 1,5 VA |
| Napięcie pierwotne | 230 V 50 Hz |
| Znamionowe napięcie wtórne | 2 x 6 V |
| Znamionowy prąd wtórny | 2 x 125 mA |
| Jałowe napięcie wtórne | 2 x 9,1 V |

Transformator ten nie jest typowym przekładnikiem napięciowym i dokładność jego przekładni nie jest określona. Należy również zauważyć, że znamionowa przekładnia tego transformatora (230 V/6 V) została podana przez producenta dla prądu obciążenia 125 mA. W rzeczywistości podczas pomiarów transformator jest obciążany bardzo małym prądem (ok. 5 mA), wynikającym z dużej rezystancji dzielnika napięciowego (3,6 kΩ). Można więc praktycznie założyć, że transformator będzie pracował w stanie jałowym. Na dwóch połączonych szeregowo uzwojeniach wtórnych należy więc spodziewać się napięcia skutecznego ponad 18 $V_{RMS,}$ a przy wzroście o 10 % będzie to 20 $V_{RMS}$, czyli amplituda wyniesie ponad 28 $V_{PEAK.}$ Jest to prawie trzy razy za wysokie napięcie w stosunku do zakresu modułu pomiarowego ±10 V. Dlatego dodatkowo zastosowano symetryczny dzielnik napięciowy składający się z 3 jednakowych rezystorów 1,2 kΩ. Kondensatory 470 nF tworzą z rezystorami dzielnika napięciowego filtr dolnoprzepustowy o częstotliwości granicznej ok. 840 Hz, co tłumi zakłócenia wysokoczęstotliwościowe z sieci. Dwukierunkowy transil P6KE 15CA ogranicza ewentualne przepięcia do bezpiecznej wartości ±15 V [8]. Łączna stała przetwarzania całego układu, uwzględniająca przekładnię transformatora w stanie jałowym i dzielnik napięciowy, wynosi około 39 V/V. Na każdym stanowisku w laboratorium jest to nieznacznie inna wartość ze względu na rozrzuty produkcyjne parametrów transformatora. Dlatego parametr ten będzie wyznaczony eksperymentalnie podczas zajęć w laboratorium. Dla napięcia znamionowego 230 V amplituda napięcia na wyjściu opisanego układu powinna wynosić około 8,34 V, a przy napięciu podwyższonym o dopuszczalną wartość 10 % nie powinna przekroczyć 9,2 V. Układ zapewnia więc prawidłowe pomiary napięcia sieciowego na zakresie ±10 V modułu pomiarowego.

## 1.3. Procedura włączania i wyłączania stanowiska laboratoryjnego

Niezależnie od programu realizowanego ćwiczenia obowiązuje taka sama, ogólna procedura uruchamiania stanowiska laboratoryjnego oraz kończenia pracy na nim. Dodatkowe szczegóły opisane są w programie każdego ćwiczenia.

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan panelu ćwiczeniowego oraz komputera. **Zasilanie jednego i drugiego powinno być wyłączone, a moduł pomiarowy – odłączony od portu USB komputera**.

Stanowisko laboratoryjne uruchamiamy w następującej kolejności:

1. Sprawdzamy poprawność połączeń na panelu ćwiczeniowym, zgłaszamy prowadzącemu ewentualne nieprawidłowości.
2. Dołączamy do panelu ćwiczeniowego dodatkowe moduły ćwiczeniowe, jeśli przewiduje to program ćwiczenia.
3. Zgłaszamy prowadzącemu gotowość i po uzyskaniu zgody włączamy komputer.
4. Po uruchomieniu się systemu operacyjnego komputera przeprowadzamy czynności opisane w instrukcji, niewymagające dołączenia modułu pomiarowego.
5. Zgłaszamy prowadzącemu gotowość i po uzyskaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera.
6. Sprawdzamy stan niebieskiej kontrolki LED na obudowie modułu NI USB-6001 według opisu podanego w tabeli 1.2. Po upewnieniu się, że moduł działa poprawnie, przeprowadzamy czynności opisane w instrukcji niewymagające włączenia napięcia zasilającego na panelu ćwiczeniowym.
7. Zgłaszamy prowadzącemu gotowość i po uzyskaniu zgody włączamy napięcie zasilające na panelu ćwiczeniowym.
8. Realizujemy program ćwiczenia opisany w instrukcji.

Po zrealizowaniu programu ćwiczenia kończymy pracę na stanowisku laboratoryjnym według następującej procedury:

1. Kopiujemy na nośnik danych USB wszystkie pliki zapisane na dysku komputera podczas realizacji ćwiczenia.
2. Przedstawiamy prowadzącemu protokół do zatwierdzenia i podpisania.
3. Wyłączamy napięcie zasilające na panelu ćwiczeniowym.
4. Zamykamy system operacyjny komputera.
5. Po wyłączeniu się komputera odłączamy od niego przewód USB modułu pomiarowego.
6. Odłączamy od panelu ćwiczeniowego dodatkowe moduły ćwiczeniowe (jeśli były do niego dołączone w ramach zrealizowanego ćwiczenia).

## 1.4. Uruchamianie i testowanie stanowiska

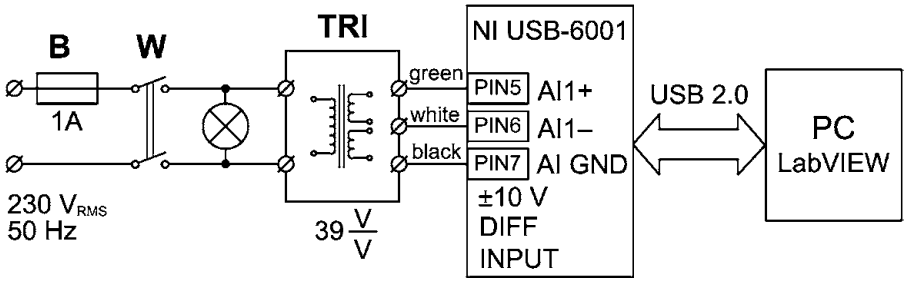
### 1.4.1. Dołączanie sygnału do wejścia toru analogowego modułu NI USB-6001

Przed przystąpieniem do realizacji ćwiczenia sprawdzamy stan stanowiska ćwiczeniowego oraz komputera. **Zasilanie komputera i stanowiska ćwiczeniowego powinno być wyłączone, a moduł pomiarowy – odłączony od portu USB komputera.**

Wszelkie zauważone odstępstwa od tego stanu należy zgłosić prowadzącemu zajęcia.

**Nie wolno samowolnie włączać zasilania komputera i stanowiska ćwiczeniowego bez wcześniejszego uzyskania wyraźnej zgody prowadzącego zajęcia.**

**UWAGA!** Należy przestrzegać ogólnej zasady, aby przed włączeniem zasilania przyrządów pomiarowych nigdy nie dołączać do wejść aparatury pomiarowej żadnych sygnałów. Mierzone sygnały pochodzą z sieci zasilającej panel ćwiczeniowy. Natomiast moduł pomiarowy NI USB-6001 jest zasilany z komputera poprzez interfejs USB. Dlatego zawsze najpierw należy włączyć zasilanie komputera, a dopiero potem dołączyć moduł pomiarowy do interfejsu USB komputera. Następnie należy obserwować niebieską kontrolkę LED na obudowie modułu, aby upewnić się, że pracuje on prawidłowo. Dopiero wtedy można załączyć zasilanie panelu ćwiczeniowego. Wyłączanie aparatury należy przeprowadzać w odwrotnej kolejności.




**Ryc. 1.6.** Schemat układu pomiarowego w ćwiczeniu 1

Pomiary napięcia za pomocą modułu NI USB-6001 będą realizowane w układzie pokazanym na rycinie 1.6. Układ jest w całości połączony na panelu ćwiczeniowym i nie wymaga żadnych dodatkowych czynności, poza uruchomieniem komputera i dołączeniem modułu pomiarowego do jego portu USB, dołączeniem panelu ćwiczeniowego do sieci zasilającej i włączeniem napięcia zasilającego na panelu. **Każdą z tych czynności można wykonać dopiero w momencie wskazanym w instrukcji oraz tylko po każdorazowym uzyskaniu zgody prowadzącego zajęcia.**

Układ pomiarowy jest w całości zamknięty w plastikowej obudowie elektroizolacyjnej, zabezpieczającej przed ryzykiem porażenia. Do załączania i wyłączania zasilania służy czerwony łącznik kołyskowy (typu Rocker) z podświetleniem. Obecność napięcia zasilającego stanowisko jest potwierdzana włączeniem podświetlenia łącznika. Obwód zasilania jest zabezpieczony rurkowym bezpiecznikiem topikowym 1 A. Budowę i zasadę działania modułu transformatora izolacyjnego opisano szczegółowo we wcześniejszym punkcie. Wyprowadzenie punktu wspólnego uzwojeń wtórnych transformatora (czarny przewód) dołączone jest do masy analogowej AI GND (PIN 4). Napięcia wyjściowe z symetrycznego dzielnika rezystancyjnego są dołączone do wejścia toru analogowego AI 1, skonfigurowanego w trybie różnicowym DIFF (ang. *Differential*). Wykorzystane są styki AI 1+ (PIN 5, zielony przewód) oraz AI 1– (PIN 6, biały przewód). Wykorzystywane zaciski na złączu sygnałów analogowych zostały wyróżnione pogrubioną czcionką w tabeli 1.3. Przed rozpoczęciem pomiarów należy sprawdzić zgodność połączeń ze schematem i zgłosić prowadzącemu ewentualne zauważone różnice.

Jeśli układ jest połączony prawidłowo, zgłaszamy prowadzącemu zajęcia gotowość do rozpoczęcia realizacji ćwiczenia.

**Po otrzymaniu od prowadzącego zajęcia wyraźnej zgody włączamy tylko zasilanie komputera. Nie włączamy jeszcze zasilania stanowiska ćwiczeniowego. Nie dołączamy modułu pomiarowego do portu USB komputera.**

Po uruchomieniu się systemu operacyjnego nie wykonujemy żadnych czynności i przechodzimy do realizacji kolejnego punktu.

### 1.3.2. Instalowanie rzeczywistego modułu fizycznego NI USB-6001 w systemie

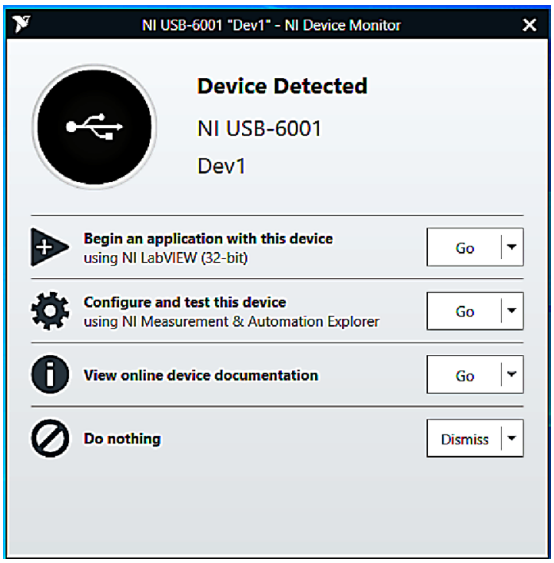
Po uruchomieniu komputera podczas pierwszych zajęć należy utworzyć na dysku katalog do przechowywania wszystkich plików. Nazwę katalogu i jego lokalizację w drzewie katalogów należy uzgodnić z prowadzącym zajęcia. Pełną ścieżkę dostępu należy zanotować w protokole. Wszystkie pliki utworzone w ciągu całego semestru należy zapisywać wyłącznie w tym katalogu. Dla wygody można w nim tworzyć podkatalogi dla kolejnych realizowanych ćwiczeń. Podczas zajęć wszystkie pliki należy zapisywać na dysku twardym komputera. Dopiero po zakończeniu każdego ćwiczenia należy skopiować wszystkie pliki na własny nośnik i wykorzystać je do opracowania raportu. Do końca semestru nie wolno kasować żadnych własnych plików z dysku komputera.

Moduł akwizycji danych pomiarowych USB-6001 wymaga zainstalowania na komputerze pakietu oprogramowania NI-DAQmx w wersji 7.5 lub nowszej. Pakiet NI-DAQmx to zestaw sterowników do modułów pomiarowych firmy National Instruments [9]. Bez zainstalowanego pakietu NI-DAQmx można używać środowiska LabVIEW, ale nie będzie możliwe korzystanie z modułów pomiarowych. Komputery w laboratorium mają zainstalowane odpowiednie oprogramowanie. Jednym z jego elementów jest aplikacja NI Device Monitor, która jest uruchamiana automatycznie podczas startu systemu operacyjnego. Aplikacja ta na bieżąco kontroluje stan komponentów pomiarowych obecnych w systemie.

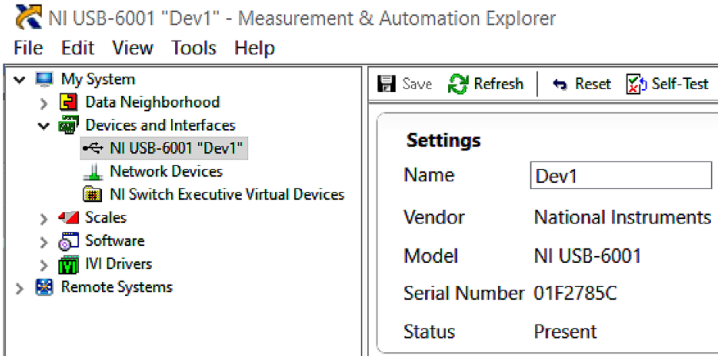
**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera.** Sprawdzamy stan niebieskiej kontrolki LED na obudowie modułu i porównujemy z opisem w tabeli 1.2. Notujemy wynik obserwacji w protokole.

Przy pierwszym dołączeniu modułu NI USB-6001 do portu USB komputera zostanie on wykryty przez program NI Device Monitor i automatycznie zainstalowany w systemie. Jeśli w systemie nie ma zainstalowanych wcześniej innych modułów, to przy pierwszej instalacji moduł będzie nosił nazwę NI USB-6001 **Dev1**. Instalacja modułu jest wymagana tylko raz, gdy moduł jest dołączany do komputera po raz pierwszy. Następnym razem, gdy ten sam moduł zostanie dołączony do tego samego komputera, powinien zostać prawidłowo rozpoznany przez system dzięki swojemu indywidualnemu numerowi seryjnemu.

Po poprawnym zainstalowaniu modułu program NI Device Monitor pokaże komunikat przedstawiony na rycinie 1.7. Notujemy w protokole nazwę urządzenia przypisaną do zainstalowanego właśnie modułu (przykładowo tutaj **Dev1**). Wybieramy opcję **Configure and test this device**. Spowoduje to automatyczne uruchomienie kolejnego programu narzędziowego NI Measurement & Automation Explorer, w skrócie NI MAX.




**Ryc. 1.7.** Potwierdzenie poprawnego zainstalowania modułu fizycznego w systemie przez program NI Device Monitor




**Ryc. 1.8.** Okno programu NI MAX z listą zainstalowanych w systemie urządzeń

Po chwili otworzy się okno programu NI MAX pokazane na rycinie 1.8, zawierające listę zainstalowanych w systemie urządzeń. Zainstalowany

przed chwilą moduł powinien być wyróżniony na liście. W protokole należy zanotować przypisaną do urządzenia nazwę (**Dev1** lub inną, jeśli w systemie były już zainstalowane wcześniej inne urządzenia), model (powinien być: NI USB-6001), numer seryjny i status urządzenia (powinien być: **Present**). Porównujemy odczytany z programu numer seryjny z numerem umieszczonym na stanowisku laboratoryjnym. Przechodzimy do kolejnego punktu, w którym zostanie przeprowadzona procedura instalowania w systemie urządzenia symulowanego.

Należy pamiętać, że komputery w laboratorium są używane do realizacji różnych przedmiotów i współpracują z różnymi układami pomiarowymi. Dlatego mogą być w nich już zainstalowane wcześniej inne moduły pomiarowe (ang. *Data Acquisition* – DAQ). Może się również zdarzyć, że moduł DAQ, używany i zainstalowany już w systemie na wcześniejszych zajęciach, został z nieznanych powodów (najczęściej przypadkowo) usunięty przez innych studentów. Należy więc zawsze na początku zajęć upewnić się, jaki numer urządzenia został przypisany w systemie do używanego aktualnie modułu pomiarowego. Trzeba również upewnić się, czy na każdych zajęciach pracujemy na tym samym stanowisku laboratoryjnym, gdyż każdy wyprodukowany moduł DAQ ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczonych w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**. Jeśli więc wykorzystywany moduł został zainstalowany pod inną nazwą, należy odpowiednio zmodyfikować tworzone programy.

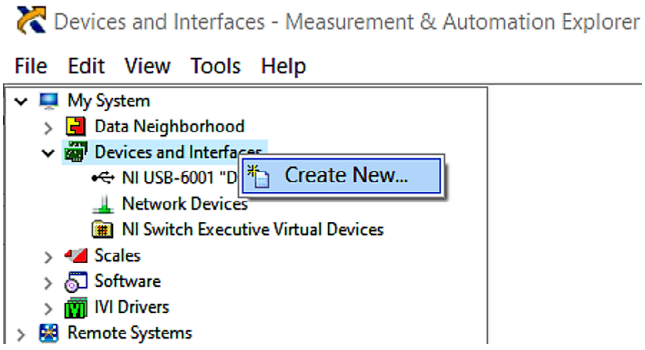
### 1.4.3. Tworzenie symulowanego modułu NI USB-6001 w systemie

Rzeczywiste fizyczne moduły pomiarowe są dostępne dla studentów tylko w laboratorium na uczelni. Aby umożliwić realizację zadań na komputerach domowych, studenci powinni stworzyć w systemie symulowane urządzenie pomiarowe NI-DAQmx zarówno w laboratorium, jak i na komputerze domowym. Celem instalowania w systemie symulowanych urządzeń jest umożliwienie rozpoczęcia programowania bez fizycznego sprzętu pomiarowego poza laboratorium. Dzięki temu możliwe jest sprawdzenie poprawności składni programu i zweryfikowanie, czy zastosowana konfiguracja (zakresy pomiarowe, numery wejść itp.) spełnia specyfikację zastosowanego modułu DAQ. Symulowane urządzenie NI-DAQmx działa prawie tak samo jak prawdziwe urządzenie fizyczne. Można więc użyć symulowanych urządzeń NI-DAQmx do

tworzenia opisanych w instrukcji algorytmów i ich testowania. Poniżej opisano procedurę, którą studenci powinni wykonać na komputerach w laboratorium podczas zajęć oraz na komputerach domowych, jeśli zamierzają zainstalować środowisko LabVIEW i rozszerzyć swoje umiejętności programowania.

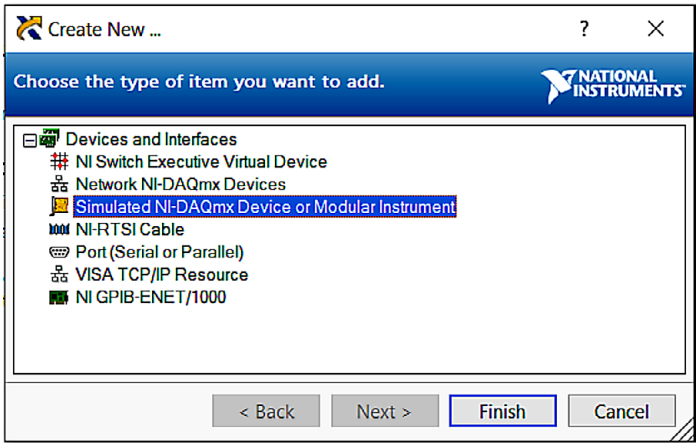
Aby utworzyć symulowane urządzenie NI-DAQmx, na komputerze musi być zainstalowany pakiet oprogramowania NI-DAQmx 7.4 lub nowszy oraz program Measurement & Automation Explorer (NI MAX). Pakiet ten jest zazwyczaj instalowany automatycznie podczas instalacji środowiska LabVIEW. Kolejność postępowania jest następująca [10]:

1. Kontynuuj pracę z poprzedniego punktu lub uruchom ponownie program NI MAX.
2. W lewej części okna programu NI MAX zaznacz opcję **My System > Devices and Interfaces** i kliknij na nią prawym przyciskiem myszy.
3. Pojawi się opcja **Create New...**, tak jak pokazano na rycinie 1.9. Kliknij na tej opcji.



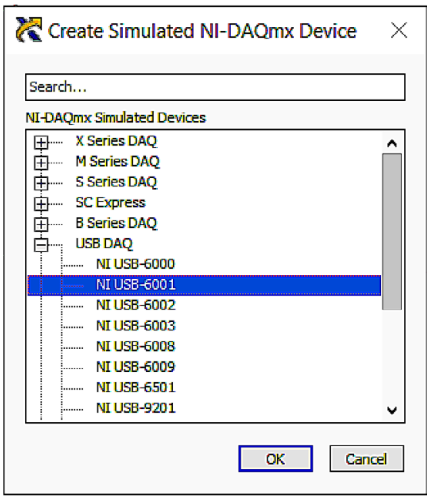

**Ryc. 1.9.** Tworzenie urządzenia symulowanego w programie NI MAX

4. W następnym oknie pokazanym na rycinie 1.10 wybierz opcję **Devices and Interfaces > Simulated NI-DAQmx Device or Modular Instrument**. Kliknij Finish.




**Ryc. 1.10.** Wybór rodzaju tworzonego urządzenia symulowanego

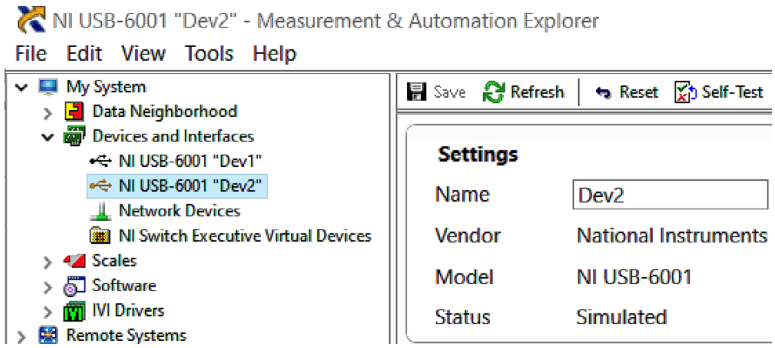
5. W oknie Create Simulated NI-DAQmx Device pokazanym na rycinie 1.11 wybierz zakładkę USB DAQ i wyszukaj urządzenie NI USB-6001. Wybierz urządzenie NI USB-6001 i kliknij OK.




**Ryc. 1.11.** Wybór typu tworzonego urządzenia symulowanego

6. Symulowane urządzenie NI-DAQmx pojawi się teraz w oknie programu NI MAX w zakładce **My system > Device and Interfaces**, tak jak pokazano na rycinie 1.12. Kolor ikony symulowanych urządzeń NI-DAQmx jest żółty, a dla urządzeń fizycznych – czarny. Zainstalowany przed chwilą

moduł powinien być wyróżniony na liście. W protokole należy zanotować przypisaną do urządzenia nazwę (**Dev2** lub inną, jeśli w systemie były już zainstalowane wcześniej inne urządzenia), model (powinien być: NI USB-6001) i status urządzenia (powinien być: **Simulated**). Należy zwrócić uwagę, że symulowane urządzenie nie ma numeru seryjnego. Nie podejmujemy żadnych innych czynności i przechodzimy do kolejnego punktu, w którym będzie przeprowadzony test zainstalowanych urządzeń.




**Ryc. 1.12.** Rezultat poprawnie utworzonego urządzenia symulowanego

**UWAGA!** Zgodnie z przedstawioną powyżej procedurą komputery w laboratorium na uczelni będą miały standardowo zainstalowane jedno fizyczne urządzenie pomiarowe o nazwie **Dev1** i drugie symulowane urządzenie o nazwie **Dev2**, jak pokazano na rycinie 1.12. Studenci na swoich komputerach domowych powinni mieć zainstalowane tylko jedno symulowane urządzenie o nazwie **Dev1**. W rezultacie programy napisane przez studentów w domu, skonfigurowane do pracy z urządzeniem symulowanym o nazwie **Dev1**, powinny od razu działać prawidłowo z urządzeniem fizycznym w laboratorium na uczelni, również mającym nazwę **Dev1**.

Celem stosowania symulowanych urządzeń jest umożliwienie rozpoczęcia programowania bez sprzętu, sprawdzenie poprawności algorytmu programu, zaprojektowanie panelu programu i zweryfikowanie, czy dane konfiguracyjne spełniają specyfikację modułu pomiarowego. Symulowane urządzenie

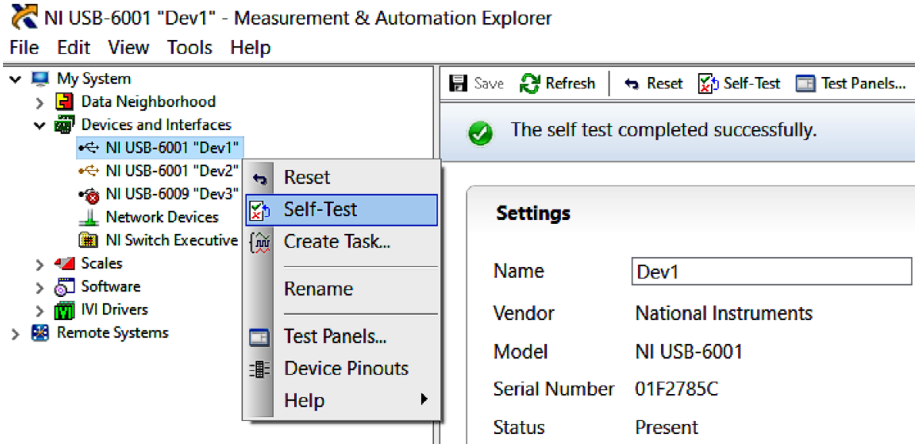
NI-DAQmx działa więc prawie tak samo jak prawdziwe urządzenie. Istnieją jednak pewne ograniczenia w korzystaniu z symulowanych urządzeń w porównaniu z fizycznymi urządzeniami pomiarowymi. Poniżej przedstawiono kilka z tych ograniczeń [11]:

- wszystkie symulowane urządzenia zwracają analogowe dane wejściowe w postaci fali sinusoidalnej wypełniającej cały zakres pomiarowy z dodanym szumem o wartości 3% zakresu pomiarowego,
- gdy w zadaniu jest wiele kanałów, dane dla każdego kanału są nieznacznie przesunięte w czasie,
- dane cyfrowe są zwracane tak, jakby każdy 8-bitowy port zliczał w górę,
- dane z licznika impulsów są zawsze zwracane jako 0,
- symulowane urządzenia nie symulują czasu reakcji urządzenia fizycznego – dane są zawsze natychmiast dostępne do odczytu,
- poza sprawdzeniem, czy wartości mieszczą się w dozwolonych granicach, dane zapisane na symulowanym urządzeniu NI-DAQmx nie mają żadnych ograniczeń,
- operacje, takie jak autotest, kalibracja i resetowanie, zawsze kończą się powodzeniem i nigdy nie zgłaszają błędu
- dane, które są normalnie przechowywane na urządzeniu fizycznym, takie jak numer seryjny, są zawsze zwracane jako 0,
- symulowanych urządzeń nie można używać w tym samym zadaniu z urządzeniami fizycznymi.

### 1.4.4. Self-test modułu NI USB-6001

Po podłączeniu modułu NI USB-6001 do komputera i poprawnym zainstalowaniu go w systemie należy wykonać test jego działania. Testowanie przeprowadza się w programie NI MAX. Kontynuujemy pracę z poprzedniego punktu lub uruchamiamy ponownie program NI MAX. W oknie programu NI MAX, w zakładce **My System**, należy odczytać i zanotować w protokole parametry systemu: nazwę hosta, model, numer seryjny, system operacyjny, wielkość pamięci fizycznej, pojemność dysku. Następnie należy otworzyć zakładkę **My System > Devices and Interfaces** i sprawdzić, jakie moduły pomiarowe są zainstalowane w systemie. Przykładową sytuację przedstawiono na rycinie 1.13. Zainstalowany i prawidłowo dołączony do komputera moduł fizyczny jest wyświetlany jako aktywny (czarna ikona). Prawidłowo zainstalowany

moduł symulowany zawsze jest wyświetlany jako aktywny (żółta ikona). Zainstalowany wcześniej w systemie moduł fizyczny, który obecnie nie jest dołączony do komputera, jest wyświetlany jako nieaktywny z dodatkowym znakiem „x" na czerwonym polu. Należy zanotować w protokole zainstalowane w systemie urządzenia i ich aktualny status (aktywny, symulowany, nieaktywny). Należy wykonać zrzut ekranu do pliku i zamieścić go w raporcie, wzorując się na przykładzie ryciny 1.13.




**Ryc. 1.13.** Wykonywanie autotestu w programie NI MAX

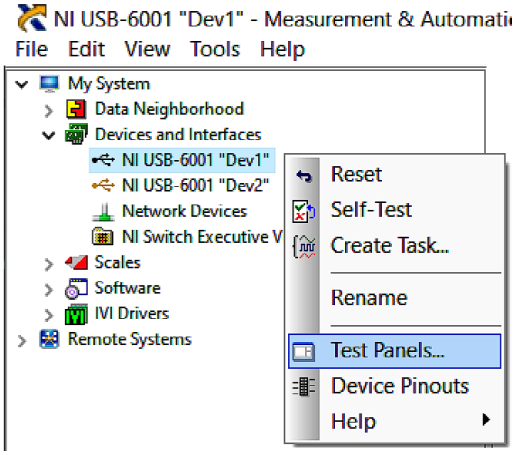
Jeśli używane podczas zajęć urządzenia NI USB-6001 są poprawnie zainstalowane, należy przeprowadzić ich autotest w sposób pokazany na rycinie 1.13. Najpierw testujemy urządzenie fizyczne z czarną ikoną. Należy wybrać testowane urządzenie z listy urządzeń i, klikając prawym przyciskiem myszy nazwę modułu, otworzyć listę zadań, a następnie wybrać opcję **Self-Test**. Po wykonaniu testu otrzymamy informacje o stanie urządzenia. W protokole należy zanotować otrzymany komunikat o wyniku testu, model urządzenia, numer seryjny i status.

Procedurę autotestu powtarzamy dla urządzenia symulowanego. W przypadku urządzenia symulowanego Self-Test zawsze kończy się powodzeniem. Zwróć uwagę na numer seryjny urządzenia symulowanego i jego status, a następnie zanotuj je w protokole.

### 1.4.5. Testowanie wejścia analogowego modułu NI USB-6001

Po przeprowadzeniu Self-Testu wykonujemy test pomiarowy. Self-Test przeprowadza ogólny test sprawności całego modułu pomiarowego, natomiast właściwy test sprawdza poprawność działania poszczególnych części modułu. Zgodnie ze schematem blokowym pokazanym na rycinie 1.3 moduł NI USB-6001 składa się z czterech głównych układów: wejść analogowych, wyjść analogowych, licznika impulsów i dwukierunkowych wejść/wyjść cyfrowych. Zazwyczaj nie ma potrzeby testowania wszystkich podzespołów modułu pomiarowego. Sensowne jest testowanie tylko tych wejść i wyjść modułu, do których są dołączone układy zewnętrzne dostarczające odpowiednie sygnały testowe na wejście modułu lub wskazujące na obecność obecność sygnałów wyjściowych z modułu. W ćwiczeniu 1 wykorzystywane będzie tylko jedno wejście analogowe w trybie różnicowym, dlatego tylko to wejście będzie testowane.

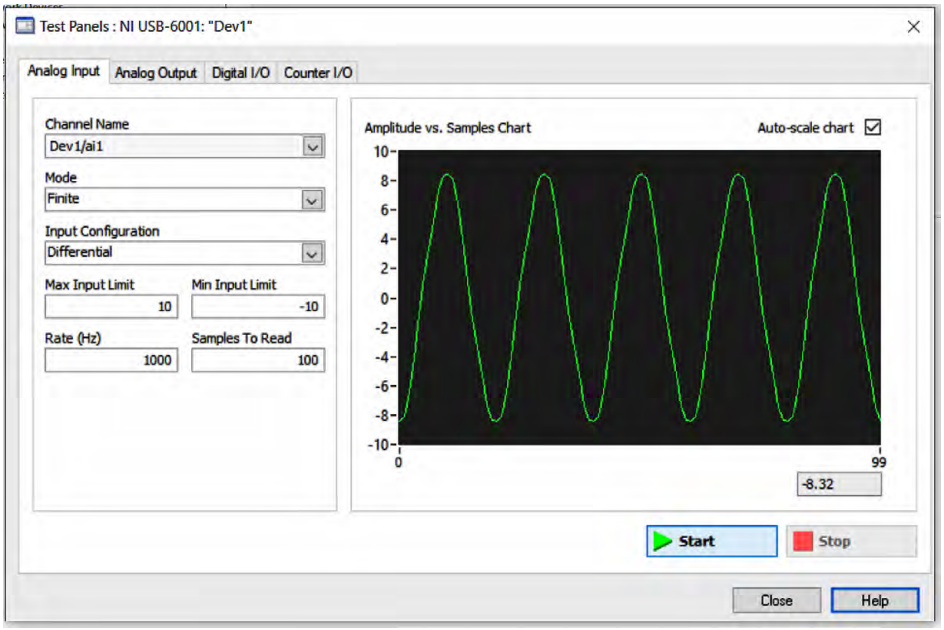
Przed rozpoczęciem testu upewniamy się, czy moduł pomiarowy jest podłączony do układu pomiarowego zgodnie ze schematem przedstawionym na rycinie 1.6. **Nie załączamy jeszcze zasilania stanowiska ćwiczeniowego.** Następnie w otwartym oknie programu NI MAX ponownie klikamy prawym przyciskiem myszy nazwę testowanego modułu i z rozwijanej listy zadań wybieramy opcję **Test Panels…**, tak jak pokazano na rycinie 1.14.




**Ryc. 1.14.** Uruchamianie panelu testowego w programie NI MAX

Otworzy się okno panelu testowego pokazane na rycinie 1.15. W nagłówku otwartego okna sprawdzamy nazwę testowanego urządzenia. Następnie na

karcie **Analog Input** wybieramy następujące opcje: **Channel Name: Dev1/ai1**, **Mode: Finite**, **Input Configuration: Differential**, **Max Input Limit: 10**, **Min Input Limit: –10**, **Rate (Hz): 1000**, **Samples To Read: 100**. Uruchamiamy test przyciskiem **Start**.



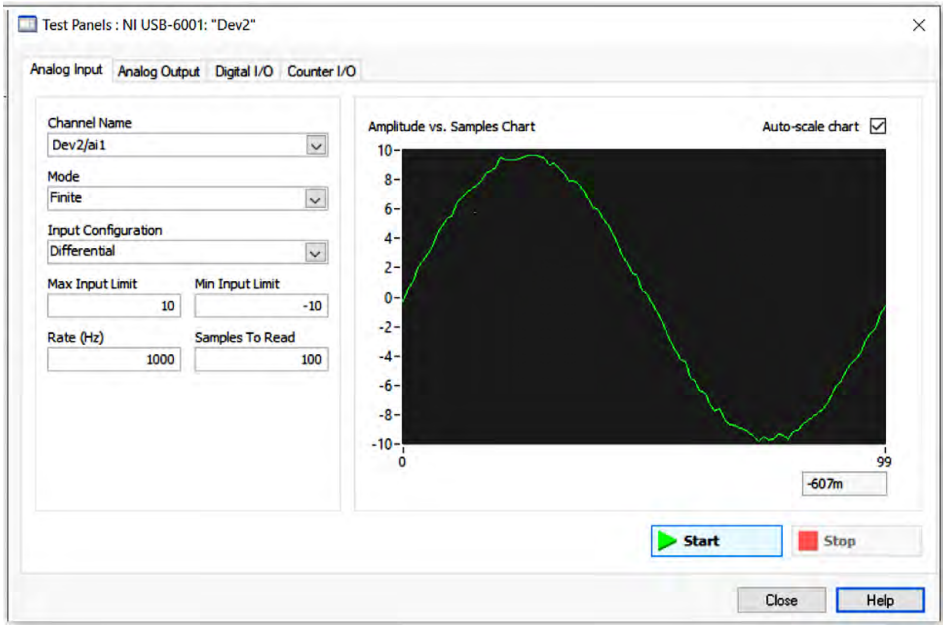

**Ryc. 1.15.** Testowanie wejścia analogowego urządzenia fizycznego

Ponieważ test został uruchomiony przy wyłączonym zasilaniu stanowiska ćwiczeniowego, powinny pokazać się wyniki pomiarów w postaci przypadkowych zakłóceń impulsowych o wartości kilku mV. Wartość średnia powinna być bliska zeru. Wniosek należy zapisać w protokole.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody włączamy zasilanie stanowiska ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku powinno potwierdzić obecność napięcia zasilającego.

Ponownie uruchamiamy test. Tym razem powinniśmy uzyskać przebieg podobny do widocznego na rycinie 1.15. Oceniamy uzyskany przebieg, porównując go z parametrami napięcia podłączonego do testowanego wejścia analogowego ai1. Test polega na pobraniu 100 próbek napięcia sieciowego z szybkością próbkowania 1000 SPS, czyli trwa 100 ms. Ponieważ znamionowa częstotliwość napięcia sieciowego wynosi 50 Hz, a jego okres 20 ms, test

powinien zarejestrować 5 okresów napięcia sinusoidalnego. Dla napięcia znamionowego 230 V amplituda napięcia na wyjściu opisanego wcześniej układu z transformatorem izolacyjnym powinna wynosić około 8,3 V. Oceniamy, czy otrzymany przebieg testu, jest zgodny z tymi wyliczeniami. Zapisujemy wnioski w protokole. Wykonujemy zrzut ekranu do pliku i wykorzystujemy go do sporządzenia raportu. Notujemy nazwę pliku w protokole.




**Ryc. 1.16.** Testowanie wejścia analogowego urządzenia symulowanego

Powtarzamy test dla urządzenia symulowanego. Testowanie urządzenia symulowanego nie wymaga żadnego dodatkowego sprzętu i może również odbywać się przy wyłączonym zasilaniu stanowiska ćwiczeniowego. Na rycinie 1.16 przedstawiono przykładowy wynik testu dla urządzenia symulowanego. Niezależnie od ustawionych parametrów testu zawsze powinniśmy otrzymać jeden okres przebiegu sinusoidalnego w pełnym zakresie pomiarowym toru analogowego wejścia z szumem o wartości 3 % zakresu. Zapisujemy wnioski z przeprowadzonego testu w protokole. Wykonujemy zrzut ekranu do pliku i wykorzystujemy go do sporządzenia raportu. Po uzyskaniu pozytywnych wyników testu zamykamy okno **Test Panels** i kończymy pracę programu **Measurement & Automation Explorer**.

**UWAGA!** Bardzo ważne jest zakończenie pracy programu NI MAX poprzez zamknięcie wszystkich jego okien. Pozostawienie aktywnego okna testowania modułu pomiarowego uniemożliwi uruchomienie programu tworzonego w środowisku LabVIEW w kolejnym punkcie ćwiczenia.

## 1.5. Programowanie

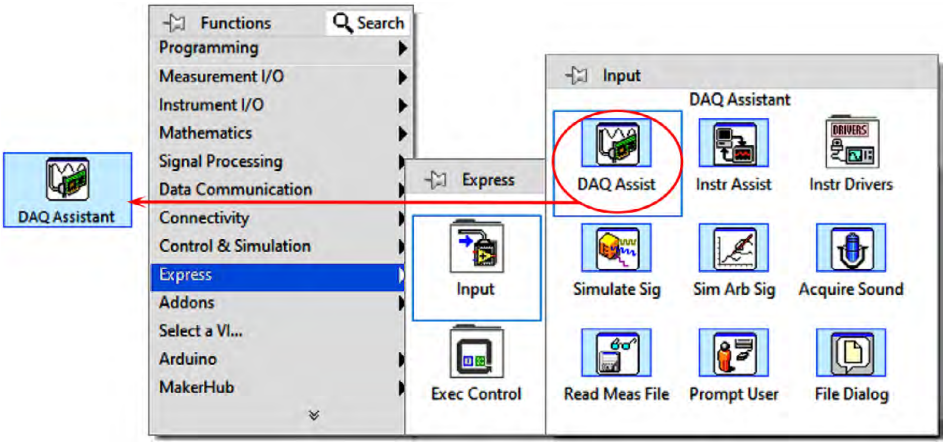
### 1.5.1. Pierwszy program do pomiaru napięcia modułem NI USB-6001

Aby wykonać pomiar napięcia w środowisku LabVIEW za pomocą modułu NI USB-6001, należy przygotować odpowiedni program, który nazywany jest przyrządem wirtualnym (ang. *Virtual Instrument* – VI). Dlatego właśnie pliki programów w środowisku LabVIEW mają rozszerzenie .vi. Program ten powinien najpierw skonfigurować moduł pomiarowy, przypisując do tworzonego zadania pomiarowego (ang. *task*) wybrany kanał fizyczny (ang. *physical channel*) modułu pomiarowego skojarzony z konkretnymi zaciskami na złączu. Zadanie pomiarowe będzie reprezentować ten fizyczny kanał pomiarowy na diagramie przyrządu wirtualnego. Następnie zadanie to należy uruchomić (ang. *start task*), odczytać wynik pomiaru (ang. *read*), zakończyć zadanie (ang. *stop task*), wyczyścić zasoby zajęte przez zadanie (ang. *clear task*) i na koniec obsłużyć ewentualne błędy (ang. *error handling*).

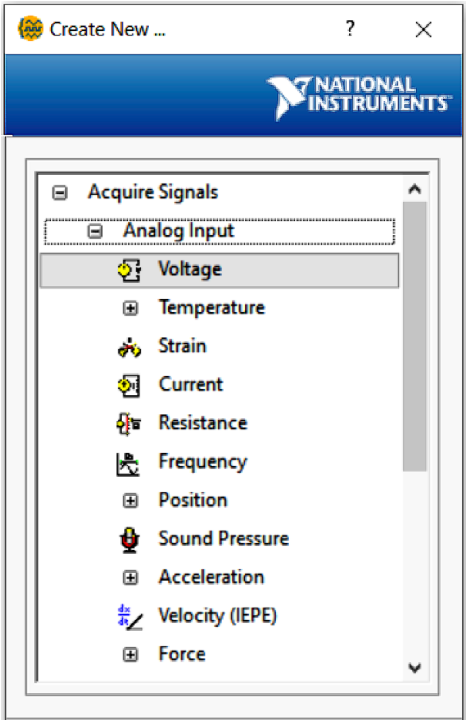
Do przygotowania diagramu realizującego te operacje wygodnie jest użyć wirtualnego kreatora (ang. *wizard*) o nazwie DAQ Assistant. To interaktywne oprogramowanie wspomagające tworzenie i konfigurowanie kanałów i zadań na diagramach programów tworzonych w środowisku LabVIEW. Pewnym mankamentem tego rozwiązania są problemy z uruchomieniem narzędzia DAQ Assistant na niektórych komputerach, zależnie od wersji systemu operacyjnego, rodzaju procesora i zainstalowanej wersji LabVIEW [12]. Problem ten został szczegółowo opisany na stronie internetowej firmy National Instruments [13]. W takim przypadku należy zastosować alternatywne sposoby tworzenia diagramu, opisane w dalszej części instrukcji w punkcie 1.5.4.

Po uruchomieniu LabVIEW otwieramy nowy przyrząd wirtualny poleceniem **File > New VI**. Przełączamy widok na okno diagramu (**Ctrl+E**). Klikając

prawym przyciskiem myszy, otwieramy zasobnik z funkcjami, a następnie znajdujemy funkcję **Functions > Express > Input > DAQ Assistant** i umieszczamy ją na diagramie programu w sposób pokazany na rycinie 1.17.

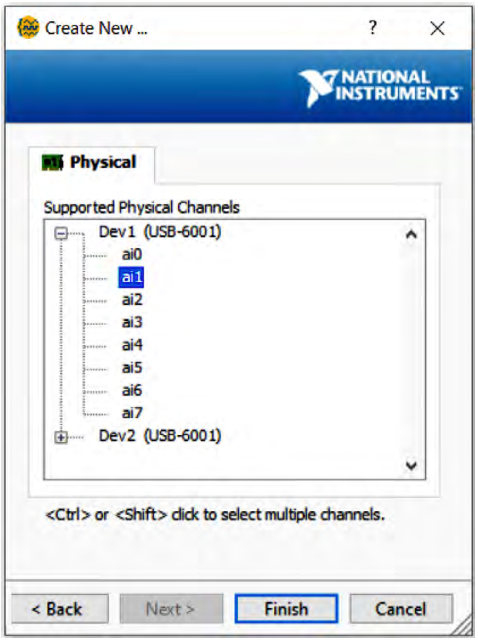



**Ryc. 1.17.** Umieszczanie kreatora DAQ Assistant na diagramie programu




**Ryc. 1.18.** Wybieranie rodzaju pomiaru dla tworzonego zadania

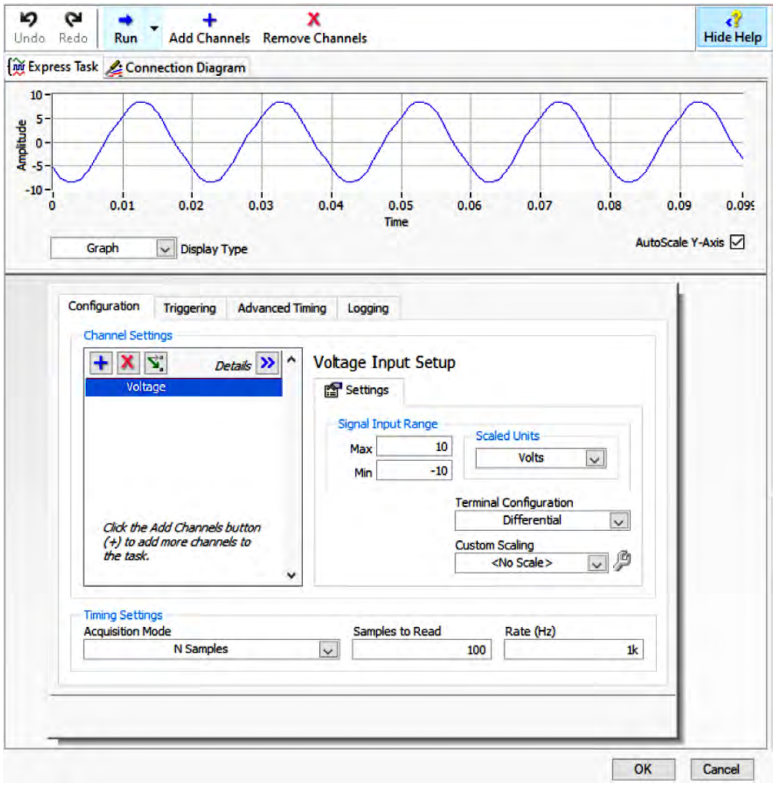
Automatycznie rozpocznie się inicjalizacja kreatora i po chwili otworzy się pierwsze okno kreatora **Create New** pokazane na rycinie 1.18, w którym rozpoczynamy tworzenie nowego zadania (ang. *task*). Zadanie to zestaw jednego lub kilku wirtualnych kanałów z przypisanymi zakresami pomiarowymi, parametrami próbkowania, trybami wyzwalania i innymi właściwościami. W pierwszej kolejności w otwartym oknie **Create New** wybieramy dla tworzonego zadania pomiar napięcia w sposób pokazany na rycinie 1.18: **Acquire Signals > Analog Input > Voltage**.




**Ryc. 1.19.** Wybieranie kanału fizycznego dla tworzonego zadania

Otworzy się kolejne okno kreatora pokazane na rycinie 1.19, w którym wybieramy kanał fizyczny dla tworzonego zadania. Kanał fizyczny (ang. *physical channel*) to zestaw konkretnych zacisków na złączu sygnałowym konkretnego modułu pomiarowego, które przypisujemy tworzonemu zadaniu pomiarowemu. Dlatego w otwartym właśnie oknie mamy dostępne listy wszystkich obsługiwanych kanałów fizycznych dla wszystkich urządzeń dostępnych w naszym systemie. Wybieramy kanał fizyczny **ai1** w urządzeniu **Dev1** (NI USB-6001), do którego dołączony jest układ pomiarowy na panelu ćwiczeniowym. Klikamy **Finish**. W ten sposób łączymy świat rzeczywisty (konkretne przewody

w układzie pomiarowym i zaciski na złączu modułu pomiarowego) ze światem wirtualnym (połączenia na diagramie programu w LabVIEW tworzące kanał wirtualny). Można to przyrównać do lustra opisanego w słynnej powieści Lewisa Carrolla, przez które Alicja przechodziła ze swojego świata do Krainy Czarów. My również podążamy w tym kierunku...

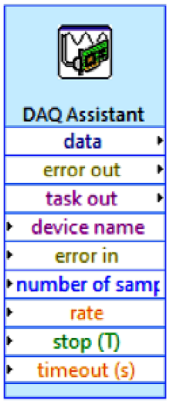



**Ryc. 1.20.** Konfigurowanie parametrów zadania, kanału wirtualnego i fizycznego

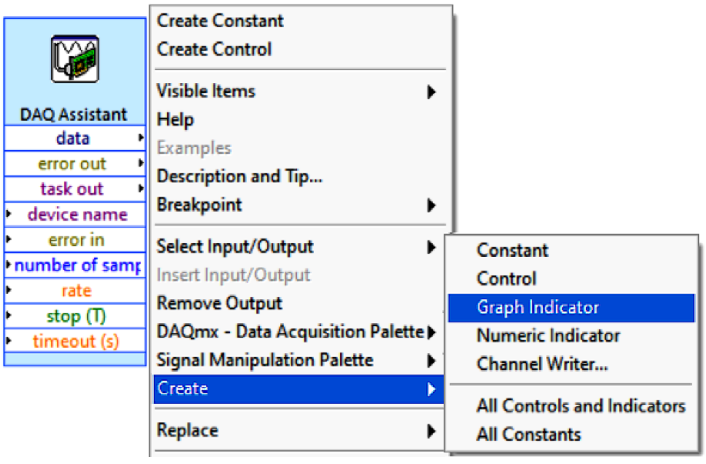
Po kliknięciu przycisku **Finish** otworzy się kolejne okno pokazane na rycinie 1.20, w którym możemy konfigurować wszystkie parametry tworzonego zadania oraz powiązanych z nim kanałów wirtualnych i fizycznych. Konfigurujemy zakres pomiarowy: **Voltage Input Setup > Signal Input Range > Max: 10, Min: –10, Scaled Units: Volts, Terminal Configuration: Differential, Custom Scaling: No Scale**. Następnie konfigurujemy parametry próbkowania: **Timing Settings > Acquisition Mode: N Samples, Samples to Read: 100, Rate (Hz): 1k**. Klikamy przycisk **Details >>** i jeszcze raz upewniamy się co do poprawności konfiguracji kanału fizycznego, którą możemy w tej

opcji zmodyfikować. Klikamy przycisk **Connection Diagram** i sprawdzamy poprawność fizycznych połączeń przewodów i zacisków w układzie pomiarowym. Klikamy przycisk **Run** i sprawdzamy poprawność przebiegu mierzonego napięcia. W górnej części okna powinniśmy zaobserwować przebieg analogiczny do uzyskanego podczas testowania wejścia analogowego modułu, co można zobaczyć na rycinie 1.15.

Konfigurację narzędzia **DAQ Assistant** kończymy, klikając przycisk **OK**. Pojawi się komunikat **Please wait... Building VI**. Po chwili na diagramie pojawi się ikona wirtualnego przyrządu **DAQ Assistant**, którą wygodnie jest rozciągnąć w dół do postaci pokazanej na rycinie 1.21, gdzie można zobaczyć wszystkie dostępne do użycia zaciski.

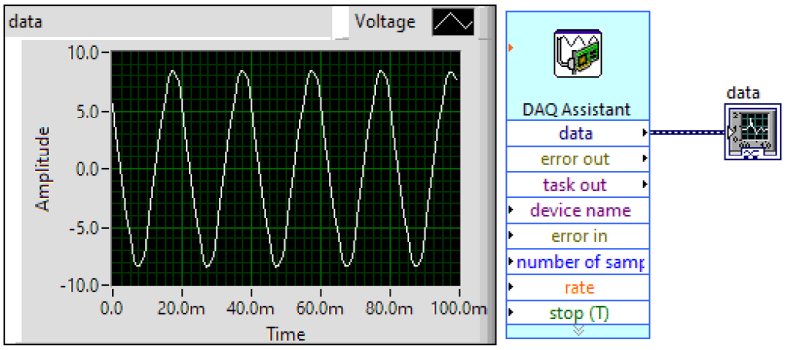



**Ryc. 1.21.** Postawiony na diagramie DAQ Assistant z widocznym wszystkimi zaciskami




**Ryc. 1.22.** Tworzenie wskaźnika graficznego na wyjściu „data" DAQ Assistant

Korzystając z narzędzia w kształcie szpulki, klikamy prawym przyciskiem myszy zacisk wyjściowy danych **data** i tworzymy wskaźnik graficzny (**Create > Graph Indicator**), jak pokazano na rycinie 1.22. Ikona **Waveform Graph** utworzona na diagramie zostanie przypisana do wskaźnika o tej samej nazwie, który pojawi się automatycznie na panelu, tak jak na rycinie 1.23.



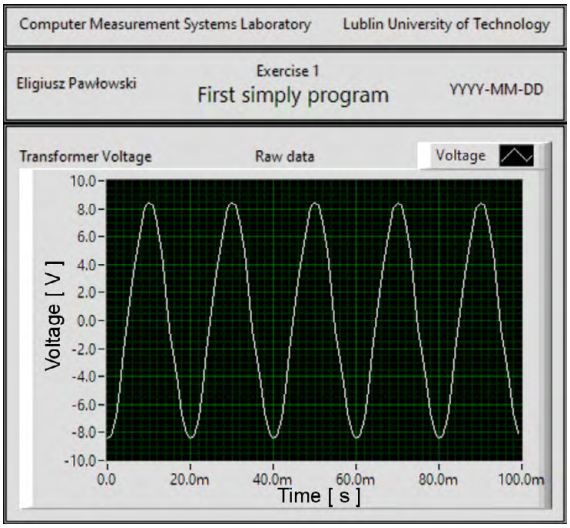

**Ryc. 1.23.** Panel i diagram pierwszej wersji najprostszego programu do pomiaru napięcia z użyciem DAQ Assistant

Otrzymaliśmy najprostszy program wykonujący pomiary napięcia za pomocą modułu NI USB-6001. Uruchamiamy go przyciskiem , co sygnalizuje zmiana jego wyglądu na . Po chwili na panelu pojawia się wynik serii 100 chwilowych wartości napięcia. Po aktywacji przycisku środowisko LabVIEW będzie automatycznie uruchamiać ponownie program w sposób ciągły, co sygnalizuje zmiana wyglądu przycisków na i . Zatrzymujemy program przyciskiem . Powinniśmy otrzymać wynik podobny do pokazanego na rycinie 1.23 oraz zbliżony do wyniku testu przedstawionego na rycinie 1.15. Łączna stała przetwarzania układu wejściowego TRI z transformatorem izolacyjnym wynosi około 39 V/V. Dla wartości skutecznej napięcia znamionowego w sieci energetycznej 230 $V_{RMS}$ amplituda wynosi 325 $V_{PEAK,}$ czyli amplituda napięcia na wejściu analogowym modułu powinna wynosić około 8,3 $V_{PEAK.}$ Porównujemy parametry zmierzonego sygnału z wartościami wyliczonymi teoretycznie i upewniamy się, że wyniki pomiaru są poprawne. Notujemy wnioski w protokole.

Analizując rycinę 1.23, należy zwrócić uwagę na opisy osi wykresu. W środowisku LabVIEW podczas tworzenia wykresów automatycznie dodawane są domyślne opisy osi: **Time** dla osi odciętych i **Amplitude** dla osi rzędnych.

Należy pamiętać, aby zawsze zmodyfikować domyślnie utworzone opisy osi odpowiednio do treści wykresu, podając dla każdej osi właściwą nazwę wielkości oraz jej jednostkę miary. Zostanie to wykonane w kolejnym kroku tworzenia programu.

Opracowujemy graficznie panel programu. Przykładowy docelowy wygląd panelu programu pokazano na rycinie 1.24. Korzystając z elementów graficznych dostępnych w zasobniku **Controls > Decorations**, porządkujemy i grupujemy obiekty na panelu. Uzupełniamy panel następującymi informacjami: nazwa przedmiotu, nazwa uczelni, dane osobowe zespołu laboratoryjnego, numer grupy dziekańskiej, data ćwiczenia, numer ćwiczenia, numer programu i jego tytuł. Informacje te należy podać w formie komentarza umieszczonego na panelu.




**Ryc. 1.24.** Panel końcowej wersji najprostszego programu do pomiaru napięcia z użyciem DAQ Assistant, dodane opisy i elementy graficzne

Aktualizujemy opisy wszystkich obiektów na panelu zgodnie z ich rolą oraz uaktualniamy opisy osi na wykresie, podając odpowiednie dla nich nazwy wielkości i jednostki miary. Do opisu osi odciętych **Time** dodajemy jednostkę miary [s]. Opis osi rzędnych **Amplitude** zamieniamy na **Voltage** [V].

Używamy narzędzi **Align Objects**, **Distribute Objects**, **Resize Objects**, **Reorder Objects** dostępnych na pasku narzędziowym, aby wyrównać położenie obiektów na panelu, ich odstępy i rozmiary oraz zmienić ich kolejność w warstwach. Na koniec zaznaczamy wszystkie obiekty na panelu

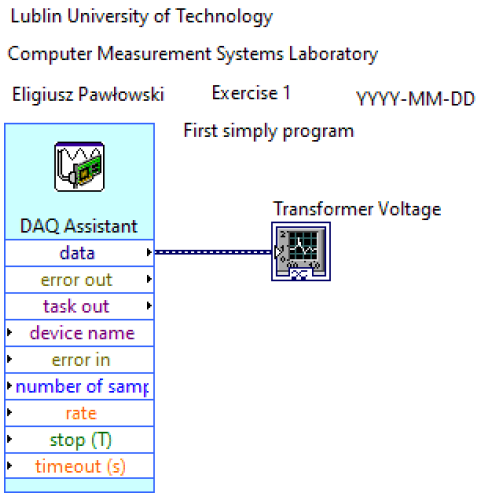
poleceniem **Edit > Select All** (**Ctrl+A**), grupujemy je (**Group**) i blokujemy położenie (**Lock**). Grupowanie i blokowanie obiektów zabezpiecza panel przed przypadkową zmianą jego wyglądu. Należy pamiętać, że zablokowanie panelu z wykresami uniemożliwia eksport wykresów do pliku (**Export Simplified Image**) w zatrzymanym programie, ale jest to nadal możliwe podczas działania programu.

Przełączamy się na diagram i uzupełniamy go o odpowiednie informacje w formie komentarza tekstowego, podobnie jak na panelu. Przykładowy docelowy wygląd diagramu programu pokazano na rycinie 1.25.

Zapisujemy program w katalogu wskazanym przez prowadzącego zajęcia. Nazwy plików z programami należy tworzyć według schematu:

YYYYMMDD_Nazwisko_KSP_X-N-L.vi,

gdzie: YYYY – cyfry roku, MM – cyfry miesiąca, DD – cyfry dnia, X – numer ćwiczenia, N – numer programu w realizowanym ćwiczeniu, L – litera (a, b, c itd.) dla kolejnych wersji tego samego programu realizowanego w danym punkcie ćwiczenia. W nazwie pliku nie wolno używać polskich liter, spacji i innych znaków specjalnych. Notujemy nazwę pliku i katalogu w protokole. W nazwie tego pliku powinien występować numer programu 1-1-a.




**Ryc. 1.25.** Diagram końcowej wersji najprostszego programu do pomiaru napięcia z użyciem DAQ Assistant, dodane opisy

Wybierając opcję **File > Print > VI Documentation**, zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Kolejność postępowania jest następująca:

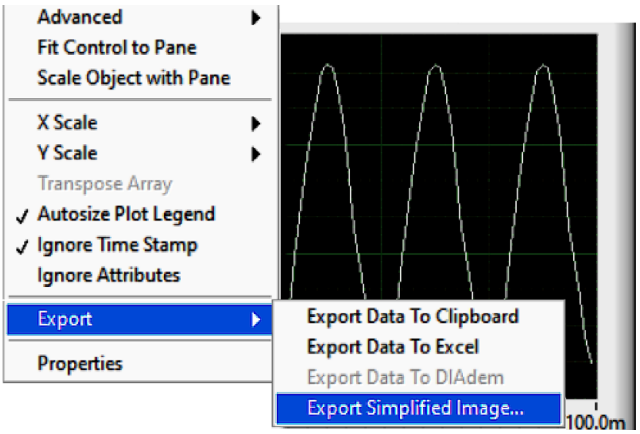
- wybieramy opcję **File > Print...** i w oknie **Select VI(s)** wybieramy nazwę programu, którego dokumentacja ma być wygenerowana, a następnie naciskamy **NEXT >**,
- w oknie **Print Contents** wybieramy opcję **VI documentation** i naciskamy **NEXT >**,
- w oknie **VI documentation** zaznaczamy opcje: **Front Panel**, **Block Diagram**, a następnie naciskamy **NEXT >**,
- w oknie **Destination** wybieramy opcję **HTML File** i naciskamy **NEXT >**,
- w oknie **HTML** wybieramy konfigurację: **Image format: JPG**, **Color depth: 256 colors**, **JPG quality: 100** i naciskamy **SAVE**,
- w oknie **SAVE** wybieramy indywidualny katalog dla danego zespołu realizującego ćwiczenie (jeśli go jeszcze nie ma, tworzymy go zgodnie z instrukcjami prowadzącego zajęcia). Nazwa pliku zostanie wygenerowana automatycznie na podstawie nazwy pliku z programem VI. Nazwa pliku nie ma podanego rozszerzenia, ponieważ zostaną zapisane trzy pliki: jeden z rozszerzeniem .html i dwa z rozszerzeniem .jpg. Możemy również wybrać inny format plików graficznych (PNG ma najmniejszą objętość), jeśli mamy pewność, że oprogramowanie naszego domowego komputera poprawnie je przetworzy. Następnie zapisujemy plik, klikając **OK**,
- znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość.

Zapisujemy nazwę utworzonego katalogu i nazwy plików z opisem zawartości w protokole. Używamy zapisanych plików z panelem i diagramem programu do opracowania raportu.

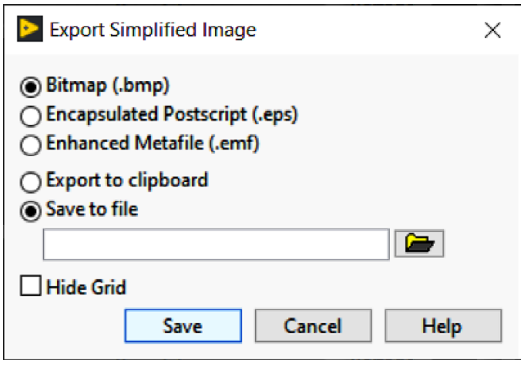
Zapisujemy uzyskany na panelu wykres zmierzonego sygnału do pliku na dysku. Aby zapisać przebieg sygnału widoczny na panelu programu, wykonujemy następujące czynności:

- umieszczamy kursor myszy na oknie przebiegu sygnału i klikamy prawym przyciskiem myszy – otworzy się menu przedstawione na rycinie 1.26,
- z otwartego menu wybieramy opcję **Export > Export Simplified Image**,
- w kolejnym oknie pokazanym na rycinie 1.27 zaznaczamy opcje **Bitmap (.bmp)** oraz **Save to file**,

- rozwijamy drzewo katalogów i wybieramy katalog utworzony przez grupę laboratoryjną na początku zajęć, wprowadzamy nazwę pliku zgodnie z zawartością wykresu, potwierdzamy przyciskiem **OK** i zapisujemy przyciskiem **Save**,
- sprawdzamy zawartość pliku, notujemy nazwę pliku z zarejestrowanym przebiegiem w protokole oraz opisujemy jego zawartość. Przykładowa postać zapisanego wykresu pokazana jest na rycinie 1.28. Używamy zapisanego wykresu do przygotowania raportu.



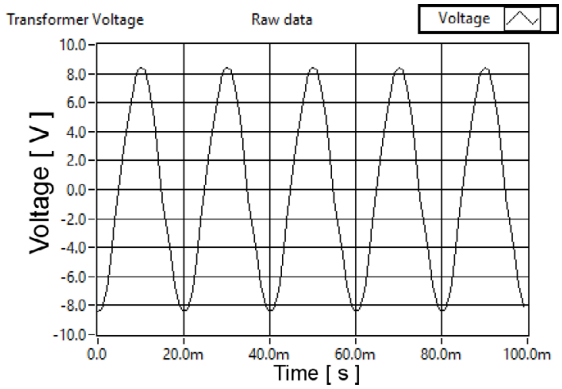

**Ryc. 1.26.** Menu umożliwiające zapis wykresu do pliku dyskowego




**Ryc. 1.27.** Konfigurowanie zapisu wykresu do pliku dyskowego

Zwracamy uwagę na kolorystykę zapisanego wykresu – są to czarne linie na białym tle. Jest ona odpowiednia do raportu, który może zostać wydrukowany. Wykres wygenerowany na panelu programu ma natomiast kolorystykę dopasowaną do oglądania na ekranie komputera (białe linie na czarnym tle),

która bardzo źle i niewyraźnie wygląda po wydrukowaniu, szczególnie na drukarce atramentowej.




**Ryc. 1.28.** Przykładowy wykres zapisany do pliku dyskowego

Upewniamy się, że zapisaliśmy ostateczną wersję programu z tego punktu ćwiczenia za pomocą opcji **File > Save**. Tworzymy teraz kopię tego programu na potrzeby następnego punktu, zmieniając numer programu na 1-2-a. Nie modyfikujemy programu, który stanowi ostateczną wersję poprzednio wykonanego zadania. Zawsze pamiętamy, aby po zapisaniu ostatecznej wersji programu realizującego w pełni dany punkt ćwiczenia utworzyć jej kopię, nadając jej nową nazwę. To bardzo ważne, ponieważ podczas modyfikowania dobrze działającego programu często popełniamy poważne błędy, które trudno poprawić. W takiej sytuacji często lepiej jest ponownie otworzyć wcześniejszą wersję pliku.

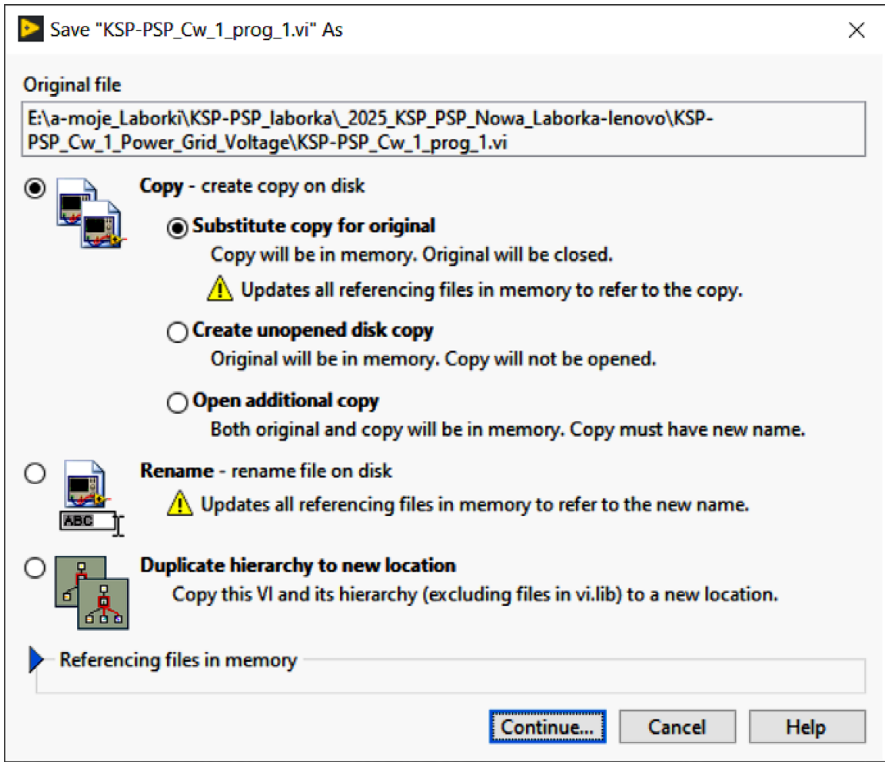
Kopię programu wykonujemy za pomocą opcji **File > Save As**, która otworzy okno pokazane na rycinie 1.29. Zaznaczamy najpierw pierwszą opcję **Copy – create copy on disk**, a poniżej kolejną opcję **Substitute copy for original**. Zgodnie z objaśnieniem *Copy will be in memory. Original will be closed*, kopia programu pozostanie w pamięci komputera, a oryginał zostanie zamknięty bez zapisywania na dysku. Na dysku zapisana zostanie tylko kopia. **Dlatego ważne jest, aby wcześniej zapisać oryginał pod dotychczasową nazwą.**

Po kliknięciu przycisku **Continue...** wybieramy katalog, modyfikujemy nazwę pliku do zapisu i klikamy **OK**.

### 1.5.2. Program z pętlą While i obliczaniem wartości napięcia w sieci energetycznej

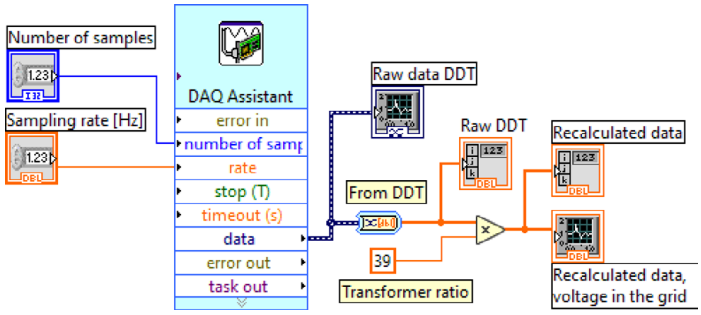
Uruchamiamy zapisaną kopię programu 1-2-a, naciskając przycisk . Ponieważ program nie zawiera żadnych pętli, przycisk pojawi się tylko na chwilę, a po jednym wykonaniu program powinien zatrzymać się automatycznie.

Uruchamiamy program, naciskając przycisk , i sprawdzamy, czy są wyświetlane ikony i . Przechodzimy do okna diagramu (**Ctrl+E**) i używamy ikony żarówki na pasku narzędziowym, aby uruchomić podgląd przepływu danych na schemacie. Zwracamy uwagę na zmianę prędkości programu przy włączonym podglądzie przepływu danych i zapisujemy wniosek w protokole. Zatrzymujemy program, naciskając przycisk . Sprawdzamy, czy ponownie pojawią się ikony i . Wyłączamy podgląd przepływu danych na schemacie.




**Ryc. 1.29.** Zapisywanie kopii programu z nową nazwą pliku

Uzupełniamy program 1-2-a o obliczenie wartości napięcia po stronie pierwotnej transformatora izolacyjnego. Przykładowy diagram pokazano na rycinie 1.30.

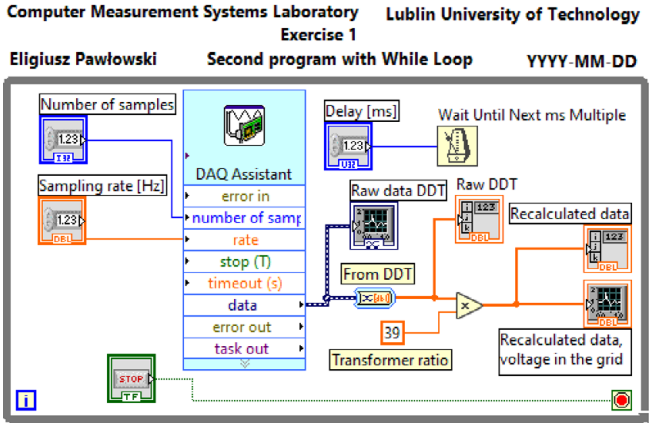



**Ryc. 1.30.** Diagram programu uzupełniony o obliczanie wartości napięcia po pierwotnej stronie transformatora izolacyjnego

Kolejność postępowania jest następująca: najpierw musimy uzyskać dane numeryczne w formacie pozwalającym na wykonywanie obliczeń. Na wyjściu bloku **DAQ Assistant** dostępne są dane w formacie DDT (ang. *Dynamic Data Type*). Dane w formacie DDT zawierają tablicę z wynikami pomiarów oraz dodatkowe informacje o parametrach próbkowania. Dzięki temu wskaźnik graficzny **Waveform Graph**, zastosowany w tym programie, potrafi poprawnie wygenerować oś czasu na wykresie widocznym na rycinie 1.23.

Możemy sprawdzić, że po zmianie parametrów próbkowania w programie oś czasu zostanie automatycznie zmodyfikowana. Niestety danych w formacie DDT nie można użyć bezpośrednio w obliczeniach. W tym celu dane numeryczne muszą zostać oddzielone od informacji o parametrach próbkowania. Służy do tego funkcja **Functions > Express > Signal Manipulation > From DDT**, którą umieszczamy obok wyjścia **data** z prawej strony bloku DAQ Assistant. Otworzy się okno konfiguracji tej funkcji, w którym wybieramy opcję **1D array of scalars**. Spowoduje to utworzenie jednowymiarowej tablicy zawierającej wyniki pomiarów wydzielonych z danych DDT, co umożliwi wykonywanie dowolnych obliczeń arytmetycznych. Wyjście **data** bloku DAQ Assistant łączymy z wejściem funkcji **From DDT**, a na jej wyjściu, klikając prawym przyciskiem myszy, tworzymy wskaźnik (**Create > Indicator**), który umożliwia przeglądanie jednowymiarowej tablicy 1D zawierającej wyniki pomiarów napięcia na wejściu analogowym modułu NI USB-6001. Uzyskaną

tablicę mnożymy przez stałą przetwarzania układu wejściowego z transformatorem izolacyjnym (wstępnie przyjmujemy wartość 39 V/V). Na wyjściu funkcji mnożenia, klikając prawym przyciskiem myszy, tworzymy kolejny wskaźnik (**Create > Indicator**), który umożliwia przeglądanie tablicy 1D zawierającej wyniki pomiarów napięcia przeliczone na stronę pierwotną transformatora izolacyjnego, czyli napięcia w sieci energetycznej zasilającej laboratorium. Przechodzimy na panel (**Ctrl+E**) i rozciągamy wskaźniki tablicowe w dół, aby widocznych było kilka wartości napięcia. Klikając prawym przyciskiem myszy na panelu, otwieramy zasobnik z kontrolkami **Controls > Modern > Graph**. Umieszczamy na panelu obiekt **Waveform Graph**, a na diagramie łączymy jego wejście z tablicą zawierającą przeliczone wyniki pomiarów. Klikając prawym przyciskiem myszy, tworzymy zadajniki (**Create > Control**) na wejściach **number of samples** i **rate** bloku **DAQ Assistant**. Powinniśmy otrzymać diagram podobny do pokazanego na rycinie 1.30. Uruchamiamy program przyciskiem  i sprawdzamy, czy działa prawidłowo. Zatrzymujemy program, naciskając przycisk  . Tworzymy kopię programu, zmieniając jego numer na 1-2-b.

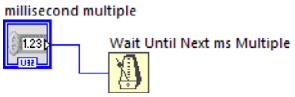



**Ryc. 1.31.** Przykładowy diagram drugiego programu z pętlą While

Uzupełniamy diagram o pętlę **While** w sposób pokazany na rycinie 1.31. Na wejściu terminala **Conditional Terminal** pętli **While**, klikając prawym przyciskiem myszy, tworzymy kontrolkę (**Create > Control**) do zatrzymywania programu. Na panelu powinien pokazać się przycisk **Stop**. Porządkujemy ułożenie kontrolek i wykresów na panelu oraz nadajemy im opisy zgodnie

z pełnioną funkcją. Uruchamiamy program przyciskiem i sprawdzamy, czy działa prawidłowo. Ponieważ program zawiera pętlę **While**, przycisk powinien być widoczny cały czas, aż do zatrzymania programu. Obserwujemy przebiegi widoczne na wykresach. Czy można łatwo ocenić ich kształt? Czy można odczytać jakieś wartości z tych wykresów? Zapisujemy wnioski w protokole. Sprawdzamy działanie przycisku **Stop** na panelu programu. Tym razem nie zatrzymujemy programu przyciskiem. Ponieważ program zawiera pętlę **While**, nie uruchamiamy go również przyciskiem.

Program w obecnej postaci wykonuje pomiary w przypadkowych chwilach w stosunku do fazy napięcia w sieci. Z tego powodu przebiegi widoczne na wykresach są niestabilne, zmieniają się bardzo szybko i trudno cokolwiek na nich zaobserwować. Dlatego uzupełniamy program o możliwość synchronizowania pomiarów z napięciem w sieci. Na diagramie wewnątrz pętli **While** wstawiamy funkcję **Functions > Programming > Timing > Wait Until Next ms Multiple**. Na jej wejściu, klikając prawym przyciskiem myszy, tworzymy zadajnik (**Create > Control**) w sposób pokazany na rycinie 1.32.



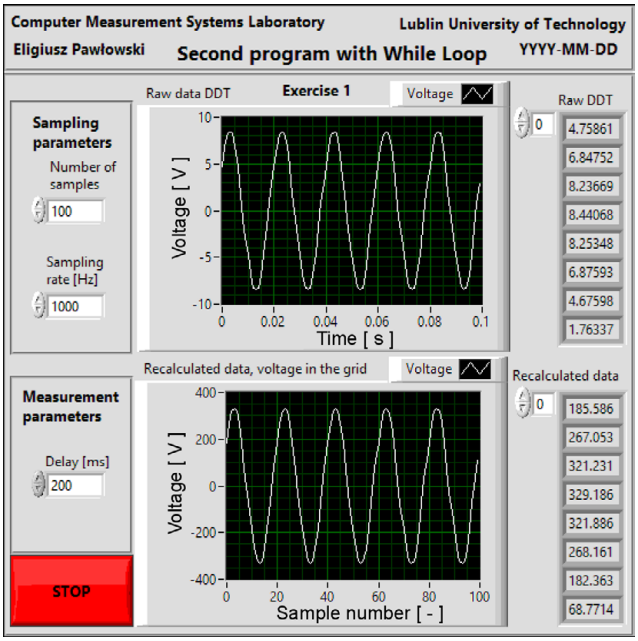

**Ryc. 1.32.** Zastosowanie funkcji kontrolującej upływ czasu w programie

Wpisujemy do zadajnika początkową wartość 150 ms. Funkcja **Wait Until Next ms Multiple** wykorzystuje czas systemowy komputera do kontrolowania upływu czasu w programie. System operacyjny komputera zlicza na bieżąco milisekundowe impulsy z zegara systemowego, które następnie wykorzystuje funkcja **Wait Until Next ms Multiple** i wstrzymuje działanie programu, aż upłynie kolejna zadeklarowana wielokrotność milisekund. W ten sposób pętla **While** zawsze rozpoczyna się w jednakowych odstępach czasu, niezależnie od szybkości działania samego programu. Stwarza to możliwość wykonywania pomiarów zawsze w tej samej fazie napięcia w sieci, o ile tylko jej częstotliwość będzie stabilna.

Naciskamy przycisk, aby uruchomić program. Zmieniając wartość opóźnienia czasowego, obserwujemy zachowanie wykresów z mierzonymi przebiegami napięcia. Eksperymentalnie ustalamy wartość opóźnienia czasowego, które pozwoli uzyskać stabilny przebieg na wykresach. Bierzemy przy

tym pod uwagę znamionową częstotliwość napięcia w sieci i przeliczamy ją na okres w milisekundach. Jaka wartość opóźnienia zapewnia stabilny obraz? Dlaczego? Wnioski zapisujemy w protokole i uwzględniamy w raporcie.

Na panelu używamy elementów graficznych dostępnych w zasobniku **Controls > Decorations**, aby uporządkować jego zawartość. Aktualizujemy opisy wszystkich obiektów zgodnie z ich rolą. Używamy narzędzi **Align Objects**, **Distribute Objects** i **Reorder** dostępnych na pasku narzędziowym, aby wyrównać położenie obiektów na panelu, zgrupować je (**Group**) i zablokować ich położenie (**Lock**). Przykład uzupełnionego panelu programu pokazano na rycinie 1.33.




**Ryc. 1.33.** Przykładowy panel drugiego programu z pętlą While

Jeśli program działa prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Dzięki temu każde ponowne wczytanie tego pliku automatycznie odtworzy domyślne nastawy w programie. Zapisujemy ostateczną wersję programu 1-2 na dysku i notujemy nazwę pliku w protokole. Drukujemy dokumentację programu do pliku i używamy jej do sporządzenia raportu. Tworzymy

kopię programu do realizacji następnego punktu, zmieniając jego numer na 1-3-a.

### 1.5.3. Program z symulacją przebiegu mierzonego napięcia

Uruchamiamy zapisaną kopię programu 1-3-a, naciskając przycisk  . Upewniamy się, czy program działa prawidłowo. Teraz sprawdzimy, jak program współpracuje z urządzeniem symulowanym. Zmieniamy konfigurację wejścia analogowego na urządzenie symulowane. W tym celu przełączamy się na diagram i klikamy dwukrotnie obiekt **DAQ Assistant**. Po chwili otworzy się okno konfiguracyjne pokazane na rycinie 1.20. Klikamy przycisk **Details >>**. Klikając prawym przyciskiem myszy nazwę zdefiniowanego kanału fizycznego, otwieramy menu. Wybieramy opcję **Change Physical Channel**. Otworzy się okno pokazane na rycinie 1.19. Tym razem wybieramy kanał fizyczny **ai1** w urządzeniu **Dev2** (NI USB-6001), które jest urządzeniem symulowanym. Klikamy **OK** i powracamy do okna pokazanego na rycinie 1.20. Kończymy konfigurację narzędzia **DAQ Assistant**, klikając przycisk **OK**. Pojawi się znany już komunikat **Please wait... Building VI**. Po chwili ponownie pojawi się diagram programu bez żadnych widocznych na zewnątrz zmian. Jednak zmiany wewnątrz obiektu **DAQ Assistant** zostały wykonane.

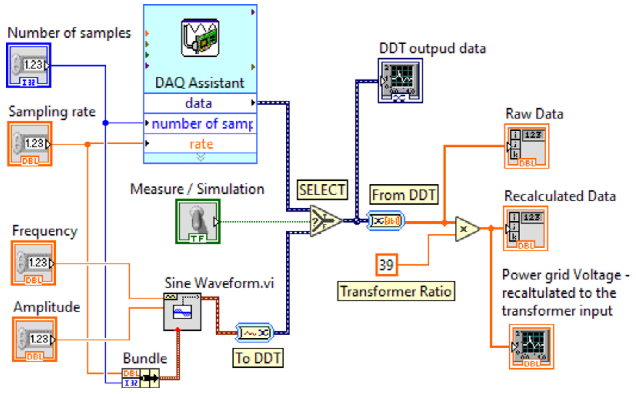
Uruchamiamy program, naciskając przycisk  . Zmieniamy parametry próbkowania (**Liczba próbek** i **Częstotliwość próbkowania**) i obserwujemy przebiegi sygnałów na panelu. Zapisujemy przykładowy przebieg sygnału z urządzenia symulowanego do pliku, notujemy nazwę pliku w protokole i wykorzystujemy go do sporządzenia raportu. Jeśli pracujemy z symulowanym urządzeniem, przebiegi sygnałów na wykresach nie zmieniają się. Dzieje się tak, ponieważ wszystkie symulowane urządzenia zawsze zwracają z wejścia analogowego wyniki w postaci fali sinusoidalnej dla pełnego zakresu z 3-procentowym szumem. Taki przebieg jest mało przydatny do testowania prawidłowości działania programu. Dlatego uzupełniamy program o symulację sygnału pomiarowego i odpowiedni przełącznik. Odpowiednią modyfikację programu przedstawiono na rycinie 1.34. Dla większej przejrzystości schematu pominięto na nim pętlę **While** i kontrolę czasu. Rycina uwzględnia tylko wewnętrzną zawartość pętli.

Najpierw rozciągamy pętlę **While** w dół, tworząc odpowiednie miejsce na dodatkową część diagramu. Następnie wstawiamy w to miejsce następujące funkcje:

- **Signal Processing > Waveform Generation > Sine Waveform,**

- **Express > Signal Manipulation > To DDT**,
- **Programming > Cluster, Class & Variant > Bundle**,
- **Programming > Comparison > Select**.

Rozmieszczamy odpowiednio te funkcje i łączymy je zgodnie z diagramem pokazanym na rycinie 1.34. Dla ułatwienia pracy włączamy okno pomocy kontekstowej za pomocą przycisku ze znakiem zapytania w prawym górnym rogu okna programu. Klikając prawym przyciskiem myszy, tworzymy zadajniki (**Create > Control**) na wejściach **frequency** i **amplitude** funkcji **Sine Waveform**. Nadajemy im początkowe wartości 50 Hz i 8,34 V. Na wejściu sterującym funkcji **Select** tworzymy zadajnik logiczny w postaci przełącznika.



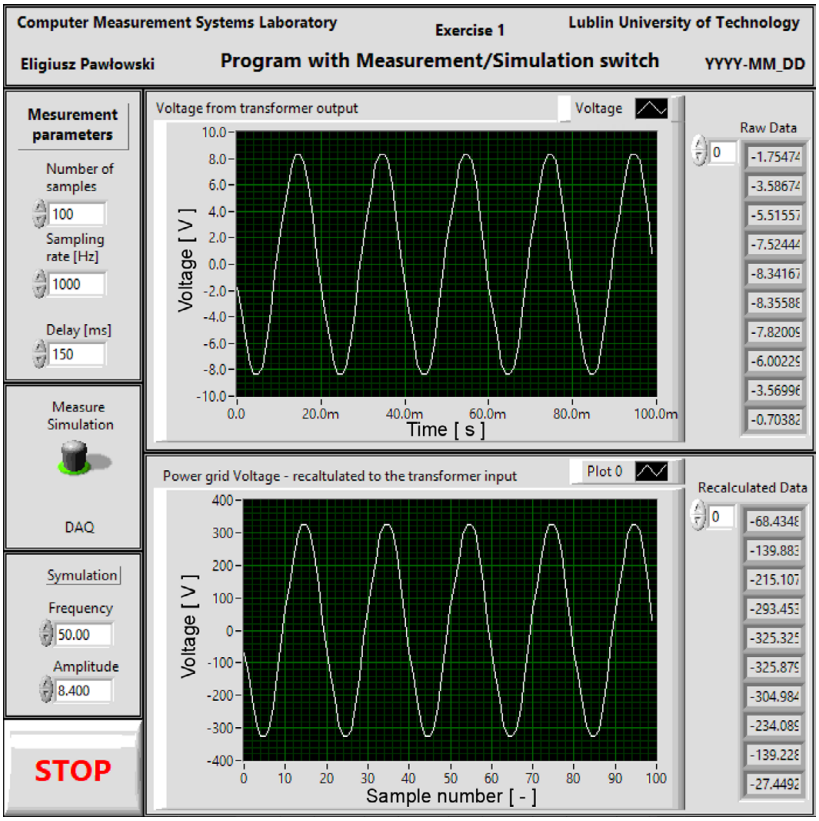

**Ryc. 1.34.** Przykładowy diagram uzupełniony o symulację napięcia sinusoidalnego

Porządkujemy elementy na panelu i uzupełniamy go niezbędnymi informacjami według przykładu pokazanego na rycinie 1.35. Podobnie uzupełniamy opisy na diagramie programu według przykładu pokazanego na rycinie 1.36.

Uruchamiamy program, naciskając przycisk . Przełączamy program w tryb symulacji napięcia. Sprawdzamy, czy program działa prawidłowo. Zatrzymujemy program, naciskając przycisk **Stop**. Jaka jest wartość amplitudy napięcia przeliczonego na stronę pierwotną transformatora? Czy jest ona prawidłowa i dlaczego? Zapisujemy wnioski i uwzględniamy je w raporcie.

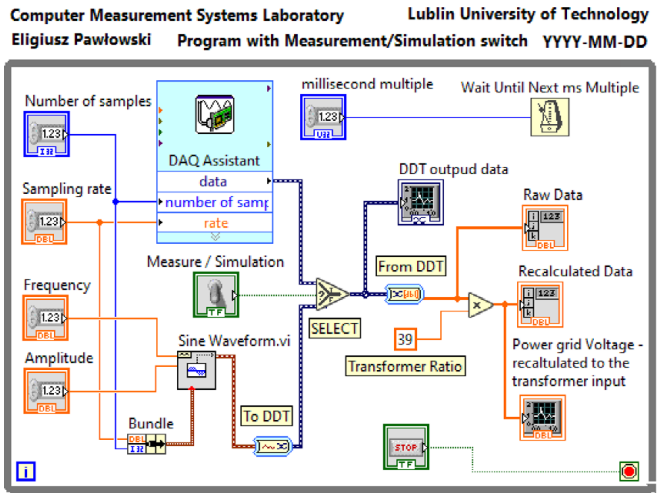
Zmieniamy ponownie konfigurację bloku **DAQ Assistant** na kanał fizyczny **ai1** urządzenia rzeczywistego **Dev1**. Uruchamiamy program i przełączamy go na pomiary napięcia w sieci. Ustawiamy szybkość próbkowania na 5000 Hz. Dzięki temu będzie można dokładnie odczytać amplitudę napięcia. Mierzymy wartość skuteczną napięcia w sieci woltomierzem cyfrowym i jednocześnie

zatrzymujemy program. Odszukujemy w tablicy z wynikami amplitudę napięcia w sieci i amplitudę napięcia odczytaną z wyjścia przetwornika ADC. Zapisujemy te wartości w protokole, uwzględniając 5 cyfr znaczących. Wykonujemy odpowiednie obliczenia, zapisując je w protokole: przeliczamy napięcie skuteczne zmierzone w sieci woltomierzem na amplitudę, zakładając sinusoidalny kształt napięcia. W tym celu mnożymy wartość skuteczną napięcia z woltomierza przez współczynnik szczytu, który dla sinusoidy wynosi $\sqrt{2}$. We wszystkich obliczeniach uwzględniamy 5 cyfr znaczących. Porównujemy otrzymany wynik z amplitudą napięcia w sieci odczytaną z programu. Czy te wartości są sobie równe? Zapisujemy wniosek w protokole. Dzielimy wartość amplitudy obliczonej ze wskazań woltomierza przez amplitudę napięcia odczytaną z wyjścia przetwornika ADC. Otrzymamy rzeczywistą wartość stałej przetwarzania układu wejściowego. Notujemy ją w protokole. Czy różni się ona od wstępnie przyjętej wartości 39 V/V?

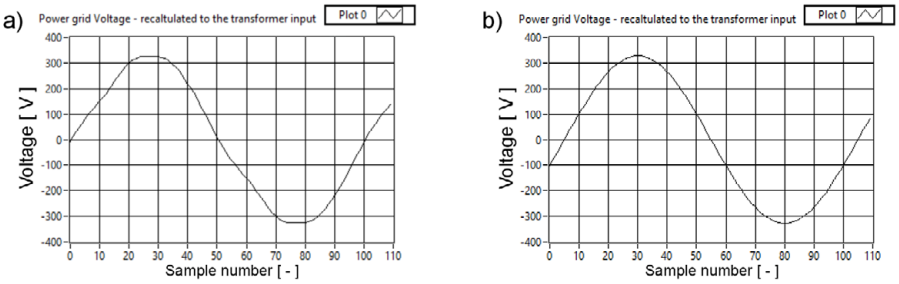



**Ryc. 1.35.** Przykładowy panel programu trzeciego

Odszukujemy na diagramie pokazanym na rycinie 1.36 stałą numeryczną, która ustala w programie wartość stałej przetwarzania układu wejściowego. Początkowo powinna mieć wartość domyślną równą 39. Zamieniamy ją teraz na obliczoną przed chwilą rzeczywistą wartość stałej przetwarzania układu wejściowego.




**Ryc. 1.36.** Przykładowy diagram programu trzeciego




**Ryc. 1.37.** Przykładowe przebiegi napięcia: a) zmierzone w sieci, b) symulowane

Uruchamiamy program i powtarzamy pomiary oraz obliczenia. Czy teraz wyniki pomiarów wykonanych zmodyfikowanym programem są dokładniejsze? Odczytujemy amplitudę napięcia na wejściu toru analogowego modułu pomiarowego. Wpisujemy tę wartość jako amplitudę napięcia symulowanego. Przełączamy program w tryb symulacji i odczytujemy amplitudę symulowanego napięcia w sieci. Czy jest ona równa amplitudzie rzeczywistego napięcia

w sieci, obliczonego ze wskazań woltomierza? Zapisujemy do plików przebiegi uzyskane z pomiarów i symulacji. Przykładowe przebiegi pokazano na rycinie 1.37.

Porównujemy kształt przebiegów napięcia rzeczywistego w sieci i napięcia symulowanego. Powinny one mieć te same amplitudy i ten sam sinusoidalny kształt. Czy to jest prawda? Zwracamy szczególną uwagę na wierzchołki obu przebiegów. Zapisujemy wnioski w protokole. Jeśli program działa prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Zapisujemy ostateczną wersję programu 1-3 na dysku i notujemy nazwę pliku w protokole. Drukujemy dokumentację programu do pliku i używamy jej do sporządzenia raportu.

Tworzymy kopię programu do realizacji następnego ćwiczenia, zmieniając jego numer na 2-1-a. Drugie ćwiczenie będzie realizowane na kolejnych zajęciach jako kontynuacja ćwiczenia pierwszego.

### 1.5.4. Alternatywna wersja programów bez zastosowania DAQ Assistant

Podczas realizacji ćwiczenia na niektórych stanowiskach może się okazać, że występują problemy z uruchomieniem kreatora **DAQ Assistant**. Zależy to od wersji systemu operacyjnego komputera, rodzaju procesora i zainstalowanej wersji środowiska LabVIEW [12]. Problem ten został szczegółowo opisany na stronie firmy National Instruments [13]. Poniżej opisano alternatywny sposób tworzenia programu do pierwszego zadania. Utworzony w ten sposób program zastępuje funkcję **DAQ Assistant** i stosujemy go we wszystkich zadaniach realizowanych w ćwiczeniu pierwszym i drugim.

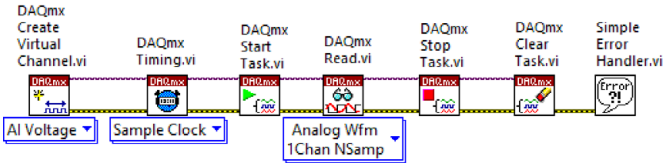
Tworzymy nowy, pusty dokument (**File > New > Blank VI**). Zapisujemy go w pierwszej kolejności na dysku (**File > Save**) i notujemy nazwę pliku w protokole. Przechodzimy do okna diagramu (**Ctrl+E**) i klikając prawym przyciskiem myszy, otwieramy paletę funkcji **Functions**. Wybieramy kategorię **Measurement I/O > NI DAQmx > DAQmx – Data Acqusition**. Wybieramy i wstawiamy do diagramu kolejno, od lewej do prawej, następujące funkcje:

- **DAQmx Create Virtual Channel**,
- **DAQmx Timing**,
- **DAQmx Start Task**,
- **DAQmx Read**,

- **DAQmx Stop Task**,
- **DAQmx Clear Task**.

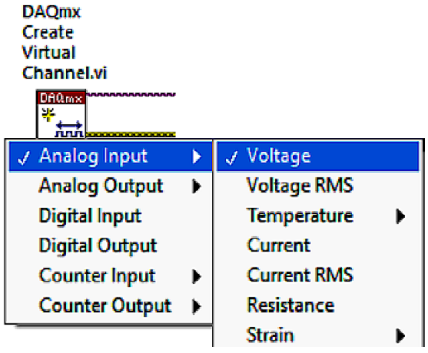
Z palety funkcji **Programming > Dialog & User Interface** wstawiamy do diagramu z prawej strony funkcję **Simple Error Handler**.

Wyrównujemy ułożenie wszystkich funkcji względem siebie i łączymy ze sobą wszystkie wyjścia **task out** z kolejnymi wejściami **task in**. Podobnie łączymy wyjścia **error out** z wejściami **error in**. Dla wygody, klikając prawym przyciskiem myszy na każdej funkcji, włączamy jej etykietę: **Visible Items > Label**. Uzyskujemy diagram jak przykładowo pokazano na rycinie 1.38.



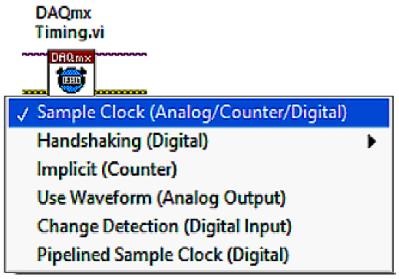

**Ryc. 1.38.** Zestaw funkcji zastępujących DAQ Assistant do pomiaru napięcia

Trzy zastosowane funkcje mają w dolnej części selektor polimorficzny (**Polymorphic VI Selector**) służący do wyboru odpowiedniej konfiguracji. Selektory polimorficzne umożliwiają przeprowadzenie konfiguracji analogicznie do konfigurowania kreatora **DAQ Assistant**, co pokazano na rycinach od 1.18 do 1.20. W pierwszej kolejności konfigurujemy funkcję **DAQmx Create Virtual Channel** do pomiaru napięcia na wejściu analogowym zgodnie z ryciną 1.39. Rozwijamy menu **Polymorphic VI Selector** i wybieramy opcję **Analog Input > Voltage**. Odpowiada to tworzeniu nowego zadania do pomiaru napięcia w kreatorze **DAQ Assistant**, pokazanego na rycinie 1.18.



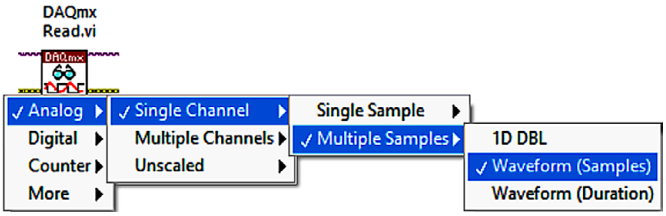

**Ryc. 1.39.** Konfigurowanie funkcji DAQmx Create Virtual Channel do pomiaru napięcia

Następnie konfigurujemy funkcję **DAQmx Timing** do zadawania parametrów próbkowania napięcia na wejściu analogowym zgodnie z ryciną 1.40. Rozwijamy menu **Polymorphic VI Selector** i wybieramy opcję **Sample Clock**.




**Ryc. 1.40.** Konfigurowanie funkcji DAQmx Timing do zadawania parametrów próbkowania

Na koniec konfigurujemy funkcję **DAQmx Read** do odczytu w jednym kanale serii próbek z wejścia analogowego zgodnie z ryciną 1.41. Rozwijamy menu **Polymorphic VI Selector** i wybieramy opcję **Analog > Single Channel > Multiple Samples > Waveform (Samples)**.




**Ryc. 1.41.** Konfigurowanie funkcji DAQmx Read do odczytu serii próbek z wejścia analogowego

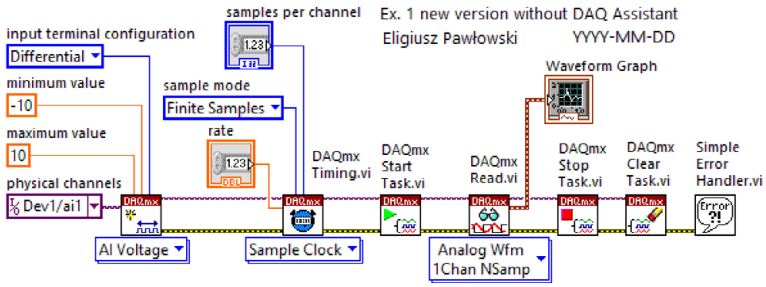
Dołączamy odpowiednie dane do wejść konfigurujących poszczególne funkcje zgodnie z ryciną 1.42. Dla funkcji **DAQmx Create Virtual Channel** tworzymy zadajniki wartości stałych (**Create > Constant**) na wejściach: **input terminal configuration** (wybieramy opcję **Differential**), **minimum value** (wprowadzamy wartość –10), **maximum value** (wprowadzamy wartość 10) oraz **physical channels** (wybieramy z listy wejście ai1 w urządzeniu **Dev1**).

Dla funkcji **DAQmx Timing**, klikając prawym przyciskiem myszy, tworzymy zadajniki numeryczne (**Create > Control**) na wejściach: **samples per channel** (wprowadzamy wartość 100) oraz **rate** (wprowadzamy wartość

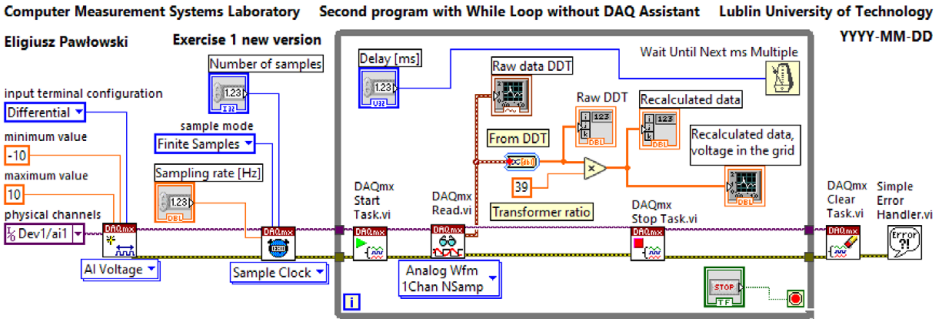
1000). Na wejściu **sample mode** tworzymy zadajnik wartości stałej (**Create > Constant**) i wybieramy opcję **Finite Samples**.

Następnie przechodzimy do okna panelu (**Ctrl+E**) i klikając prawym przyciskiem myszy, otwieramy zasobnik z kontrolkami, po czym z palety **Graph** wybieramy i wstawiamy na panel obiekt **Wavweform Graph**. Wracamy do okna diagramu (**Ctrl+E**) i łączymy wyjście data funkcji **DAQmx Read** z wejściem obiektu **Waveform Graph**.

Diagram pokazany na rycinie 1.42 zastępuje pierwszy z programów wykorzystujący funkcję **DAQ Assistant**, pokazany na rycinie 1.25. Panel opracowujemy w taki sam sposób jak w pierwotnej wersji programu pokazanej na rycinie 1.24.




**Ryc. 1.42.** Diagram programu realizującego jednokrotny odczyt serii próbek z wejścia analogowego bez DAQ Assistant




**Ryc. 1.43.** Diagram programu realizującego wielokrotny odczyt serii próbek z wejścia analogowego w pętli While bez DAQ Assistant

W drugim programie stosujemy pętlę **While**. Pierwotną wersję programu przedstawiono na rycinierysunku 1.31, na którym pętla **While** obejmuje w całości wszystkie zastosowane na diagramie funkcje. W obecnej wersji

programu pętla **While** powinna objąć tylko część diagramu w sposób pokazany na rysunku 1.43. Po lewej stronie pętli należy pozostawić funkcje **Create** oraz **Timing**, a po prawej **Clear** oraz **Simple Error Handler**. Te funkcje wykonają się tylko jeden raz. Wewnątrz pętli znajdują się funkcje **Start**, **Read** oraz **Stop**. Te funkcję będą wykonywane w każdym obiegu pętli **While**. Obliczenia zrealizowane są tak samo jak w opisie do rysunku 1.31. Opóźnienie czasowe wprowadzone zostało analogicznie do sposobu pokazanego na 1.32. Panel programu opracowujemy według ryciny 1.33.

W trzecim programie dodajemy symulację napięcia sinusoidalnego. To zadanie realizujemy tak samo jak w opisie do ryciny 1.34. Panel trzeciego programu realizujemy zgodnie z ryciną 1.35. Rzeczywistą wartość stałej przetwarzania układu wejściowego wyznaczamy eksperymentalnie w sposób opisany w punkcie 1.4.3.

## 1.6. Opracowanie raportu z realizacji ćwiczenia

Raport z przeprowadzonego ćwiczenia powinien zawierać:

- stronę tytułową z danymi osobowymi zespołu, datą ćwiczenia i jego tytułem,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań zgodnie z programem ćwiczenia,
- opis każdego z wykonanych zadań lub podanie przyczyny niewykonania,
- dokumentację zrealizowanych i uruchomionych trzech programów,
- wykresy wygenerowane przez program w środowisku LabVIEW,
- liczbowe wyniki pomiarów uzyskane z opracowanych programów,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.

W raporcie ocenie podlegają:

- poprawność działania programów,
- czytelność i przejrzystość diagramu,
- estetyka panelu, dobór kolorów i elementów graficznych,
- funkcjonalność panelu pod kątem wygody obsługi,
- zakres zrealizowanych zadań,
- trafność wniosków końcowych.

# 2. Pomiary parametrów napięcia w sieci energetycznej

## 2.1. Cel ćwiczenia i zadania do wykonania

Ćwiczenie 2 jest kontynuacją ćwiczenia 1. Ostatni program opracowany w ćwiczeniu 1 wykorzystamy jako wstępny program do rozpoczęcia ćwiczenia 2.

Celem ćwiczenia 2 jest zapoznanie się z problematyką obliczania podstawowych parametrów napięcia w sieci elektroenergetycznej na podstawie próbek wartości chwilowych.

Program laboratorium obejmuje następujące zagadnienia:

- obliczanie wartości skutecznej napięcia,
- obliczanie wartości średniej napięcia,
- obliczanie amplitudy napięcia,
- obliczanie współczynnika szczytu i współczynnika kształtu,
- obliczanie okresu i częstotliwości sygnału,
- tworzenie wykresów i histogramów,
- zapisywanie wyników pomiarów do pliku.

W ramach ćwiczenia 2 opracowujemy cztery programy:

1. Program obliczający wartość skuteczną, średnią, amplitudę, współczynnik szczytu oraz współczynnik kształtu na podstawie próbek napięcia,
2. Program obliczający okres i częstotliwość na podstawie próbek napięcia,

3. Program tworzący histogram i wykres zmienności napięcia w czasie oraz zapisujący wyniki pomiarów do pliku.
4. Czwarty program wykonujący pomiary częstotliwości o podwyższonej dokładności.

Jeśli na poprzednich zajęciach nie zrealizowaliśmy w całości programu ćwiczenia 1, najpierw musimy go dokończyć. Dopiero wtedy możemy rozpocząć ćwiczenie 2.

## 2.2. Opis stanowiska laboratoryjnego

Stanowisko laboratoryjne wykorzystywane w ćwiczeniu 2 ma dokładnie taką samą konfigurację jak w ćwiczeniu 1. Przed rozpoczęciem pomiarów ponownie zapoznajemy się z opisem stanowiska. Szczególną uwagę zwracamy na kolejność postępowania przy uruchamianiu stanowiska i kończeniu na nim pracy.

## 2.3. Parametry napięcia przemiennego w sieci energetycznej

### 2.3.1. Podstawowe definicje

W idealnych warunkach wartość chwilowa napięcia w sieci energetycznej jest opisana sinusoidalną funkcją czasu:

$$u(t) = A \sin(2\pi f\ t + \varphi), \tag{2.1}$$

gdzie: $u(t)$ jest wartością chwilową napięcia, $A$ jest amplitudą, $f$ jest częstotliwością, a $\phi$ jest fazą. Z postaci zależności (2.1) wynika, że napięcie w sieci energetycznej jest okresowe i przemienne. Okres w sekundach jest równy odwrotności częstotliwości wyrażonej w hercach:

$$f = \frac{1}{T}. \tag{2.2}$$

Do podstawowych parametrów okresowego napięcia przemiennego należą: wartość maksymalna, wartość średnia i wartość skuteczna. Dla sygnałów sinusoidalnych wartość maksymalna (szczytowa) równa jest amplitudzie $U_{max} = A$. Wartość średnia $U_{sr}$ jest równa całce z wartości chwilowych $u(t)$ za okres $T$, podzielonej przez ten okres:

$$U_{sr} = \frac{1}{T}\int_0^T u(t)\,dt\,. \tag{2.3}$$

Dla napięć przemiennych z definicji wartość średnia jest równa zero. Dla napięć występujących w sieci energetycznej wartość średnia w praktyce jest bardzo bliska zeru. Dlatego interesująca jest wartość średnia wyprostowana $U_{sr\,abs,}$ obliczana jako wartość średnia z wartości bezwzględnej napięcia chwilowego:

$$U_{sr\,abs} = \frac{1}{T}\int_0^T \left|u(t)\right| dt\,, \tag{2.4}$$

która liczbowo jest równa wartości średniej liczonej za połowę okresu, dlatego nazywana jest również wartością średnią półokresową:

$$U_{sr\,abs} = \frac{2}{T}\int_0^{\frac{T}{2}} u(t)\,dt\,. \tag{2.5}$$

W energetyce najbardziej przydatna jest wartość skuteczna napięcia $U_{sk,}$ gdyż od tej wartości uzależniona jest moc czynna i energia czynna pobierana z sieci przez odbiorniki. Utrzymanie odpowiedniej wartości skutecznej napięcia w sieci gwarantuje, że pobór mocy i energii przez odbiorniki będzie zgodny z ich znamionowymi parametrami. Jest to więc ważny parametr opisujący jakość energii elektrycznej dostarczanej do odbiorcy. Z tego powodu dokładny pomiar wartości skutecznej napięcia jest istotnym zagadnieniem. Wartość skuteczna $U_{sk}$ jest równa pierwiastkowi z uśrednionego kwadratu wartości chwilowych napięcia $u(t)$:

$$U_{sk} = \sqrt{\frac{1}{T}\int_0^T u^2(t)\,dt}\,. \tag{2.6}$$

W literaturze angielskojęzycznej wartość skuteczna bywa zazwyczaj określana skrótem RMS (**Root Mean Square** – dosłownie: pierwiastek średniego kwadratu, w skrócie: średnia kwadratowa).

W rzeczywistych warunkach napięcie w sieci energetycznej często odbiega kształtem od sinusoidy. Do oceny wielkości tego odkształcenia stosujemy między innymi współczynnik szczytu i współczynnik kształtu.

Współczynnik szczytu (**Crest Factor** – CF) to stosunek wartości maksymalnej (szczytowej) $U_{max}$ do wartości skutecznej sygnału $U_{sk}$:

$$CF = \frac{U_{\max}}{U_{sk}}. \tag{2.7}$$

Współczynnik kształtu (**Waveform Factor** – WF) to stosunek wartości skutecznej $U_{sk}$ do wartości średniej półokresowej (wyprostowanej) $U_{sr\,abs}$:

$$WF = \frac{U_{sk}}{U_{sr\,abs}}. \tag{2.8}$$

Dla sinusoidy współczynnik szczytu $CF_{\sin}$ wynosi:

$$CF_{\sin} = \sqrt{2} \cong 1,4142, \tag{2.9}$$

a współczynnik kształtu dla sinusoidy $WF_{\sin}$ jest równy:

$$WF_{\sin} = \frac{\pi}{2\sqrt{2}} \cong 1,111. \tag{2.10}$$

Dla przebiegów o kształcie innym niż sinusoidalny wartości tych współczynników są inne, co pozwala nam ocenić poziom odkształcenia napięcia. Jest to bardzo ważne zagadnienie, ponieważ w odkształconym napięciu występują składowe harmoniczne o wyższych częstotliwościach, które bardzo negatywnie wpływają na pracę sieci i dołączonych do niej odbiorników. Dla cyfrowych systemów pomiarowych szczególnie niebezpieczne są sygnały o dużym współczynniku szczytu *CF*, które przy stosunkowo niedużej wartości skutecznej $U_{sk}$ mogą mieć bardzo duże wartości szczytowe $U_{max.}$ Może to spowodować przekroczenie zakresów pomiarowych wejść analogowych, ich przesterowanie, nasycenie wzmacniaczy w torze pomiarowym, a nawet uszkodzenie aparatury.

Wartości napięć zgodnie z podanymi definicjami mierzą mierniki elektromechaniczne, których fizyczna zasada działania jest zgodna z przedstawionymi wzorami całkowymi. Wartość średnią (2.3) mierzą mierniki magnetoelektryczne, wartość średnią wyprostowaną (2.4) – mierniki magnetoelektryczne prostownikowe, a wartość skuteczną (2.6) – mierniki elektromagnetyczne i elektrodynamiczne. Niestety wadą klasycznych mierników elektromechanicznych jest ich stosunkowo niska dokładność. Wyższą dokładność uzyskujemy, stosując cyfrową technikę pomiarową. Układy cyfrowe nie operują jednak na przebiegach ciągłych, nie mogą więc wykonywać pomiarów zgodnie z przedstawionymi wzorami całkowymi.

Cyfrowe układy pomiarowe zamiast na przebiegach ciągłych operują na przebiegach dyskretnych, skwantowanych w czasie i amplitudzie. Dlatego wzory całkowe zastępujemy odpowiednimi operacjami sumowania i uśredniania próbek. Napięcie sinusoidalne opisane wzorem (2.1) po spróbkowaniu będzie reprezentowane w systemie cyfrowym przez ciąg próbek:

$$u\left[i\right] = A\sin\left(2\pi f\ i\ T_S + \varphi\right), \tag{2.11}$$

gdzie: $i$ jest numerem próbki, a $T_s$ jest okresem próbkowania. Jeśli pobraliśmy $K$ próbek napięcia, kolejne próbki będą ponumerowane od $i = 0$ do $K - 1$. Zwracamy uwagę, że zwyczajowo w systemach cyfrowych, w tym również w środowisku LabVIEW, próbki są numerowane od zera, a ostania próbka ma numer $K$-1. Okres próbkowania $T_s$ jest odwrotnością częstotliwości próbkowania $f_s$, przy czym zazwyczaj zamiast pojęcia częstotliwość próbkowania w hercach używamy szybkości próbkowania (SR – **Sampling Rate**), podawanej w liczbie próbek na sekundę (SPS – **Samples Per Second**). Musimy znać te pojęcia, ponieważ stosujemy je podczas tworzenia programów w środowisku LabVIEW.

Po spróbkowaniu sygnał ciągły w dziedzinie czasu $u(t)$ będzie reprezentowany w systemie cyfrowym przez ciąg $K$ próbek o wartościach $u[0]$, $u[1]$, $u[2]$, ... $u[i]$, ..., $u[K\text{-}1]$. Próbki są pobierane w chwilach $t[0]$, $t[1]$, $t[2]$, ..., $t[i]$, ... $t[K\text{-}1]$. Odstęp czasu pomiędzy dwiema kolejnymi próbkami jest równy okresowi próbkowania $t[i] - t[i–1] = T_s$. Na podstawie tak pobranych próbek sygnału ciągłego wyznaczamy wartość średnią:

$$U_{sr} = \frac{1}{K}\sum_{i=0}^{K-1} u[i], \tag{2.12}$$

wartość średnią wyprostowaną (półokresową):

$$U_{sr\,abs} = \frac{1}{K}\sum_{i=0}^{K-1} \left|u[i]\right|, \tag{2.13}$$

oraz wartość skuteczną:

$$U_{sk} = \sqrt{\frac{1}{K}\sum_{i=0}^{K-1} u^2[i]}\,. \tag{2.14}$$

Wzory (2.12), (2.13) i (2.14) zastosujemy w tworzonych programach do obliczenia wartości średnich i skutecznych na podstawie próbek napięcia zmierzonych modułem NI USB-6001. Następnie obliczymy współczynniki szczytu *CF* i kształtu *WF* według wzorów (2.7) i (2.8). Aby obliczenia wykonane na podstawie próbek sygnału $u[i]$ były równoważne odpowiednim całkom z przebiegu ciągłego $u(t)$, musi zostać spełniony dodatkowy warunek: łączny czas pobierania $K$ próbek (czyli tzw. okno pomiarowe) powinien być równy okresowi sygnału $T$ lub jego całkowitej wielokrotności:

$$K \cdot T_S = N \cdot T\,, \tag{2.15}$$

gdzie $N$ jest liczbą naturalną (1, 2, ...). Jeśli warunek (2.15) nie zostanie spełniony, pojawi się niekorzystne zjawisko przecieku widma (*spectral leakage*), co spowoduje zwiększenie błędów pomiaru. Jednocześnie obowiązuje warunek Nyquista, który nakazuje próbkować sygnał z szybkością co najmniej dwukrotnie większą od największej częstotliwości występującej w tym sygnale. Ponieważ w napięciu sieciowym występują liczne składowe harmoniczne, w praktyce pobieramy kilkadziesiąt próbek w jednym okresie (który wynosi znamionowo 20 ms), a okno pomiarowe przyjmujemy jako równe kilku okresom. Takie warunki przyjmiemy również w realizowanym ćwiczeniu.

### 2.3.2. Wymagania przepisów

W Polsce, zgodnie z normą [14], napięcie znamionowe sieci niskiego napięcia wynosi 230/400 V. Parametry jakościowe napięcia zasilającego, które powinny być utrzymane w publicznych sieciach elektroenergetycznych, są określone w normie [15]. Metody pomiaru jakości energii elektrycznej opisano w normie [16]. Szczegółowe warunki funkcjonowania systemu elektroenergetycznego precyzuje rozporządzenie [17].

Wymagane parametry napięcia zasilającego są zdefiniowane w zależności od grupy przyłączeniowej, do której zaliczany jest dany odbiorca. Gospodarstwa domowe, szkoły, warsztaty rzemieślnicze i inne podmioty, których urządzenia, instalacje lub sieci są przyłączane bezpośrednio do sieci o napięciu znamionowym równym 1 kV lub niższym oraz mocy przyłączeniowej nie większej niż 40 kW, zaliczają się do grupy przyłączeniowej V.

Zgodnie z przepisami [17] dla podmiotów zaliczanych do grup przyłączeniowych III–V ustala się następujące parametry jakościowe energii elektrycznej w przypadku sieci funkcjonującej bez zakłóceń:

1. wartość średnia częstotliwości mierzonej przez 10 s zawiera się w przedziale:
   - 50 Hz ± 1 % (od 49,5 Hz do 50,5 Hz) przez 99,5 % tygodnia,
   - 50 Hz + 4 % / – 6 % (od 47 Hz do 52 Hz) przez 100 % tygodnia;
2. w każdym tygodniu 95 % ze zbioru 10-minutowych średnich wartości skutecznych napięcia zasilającego zawiera się w przedziale odchyleń ±10 % napięcia znamionowego.

W ramach ćwiczenia 2 opracujemy programy, które umożliwią pomiar wartości skutecznej napięcia sieci energetycznej oraz częstotliwości w celu porównania ich z wymaganiami przepisów [17]. Szczególnie ważnym zagadnieniem jest pomiar częstotliwości napięcia w sieci energetycznej, a zwłaszcza ciągłych, niewielkich fluktuacji częstotliwości wokół wartości znamionowej 50 Hz. Zagadnienie to znalazło również interesujące zastosowanie w nowoczesnych technikach kryminalistycznych [18, 19]. Kryterium częstotliwości sieci elektrycznej (**Electric Network Frequency** – ENF) jest stosunkowo nową techniką kryminalistyczną, opracowaną w celu ustalania autentyczności nagrań audio i wideo. Polega ona na wydzieleniu sygnału ENF z nagrań dowodowych i dopasowaniu go do bazy danych ENF. Taką bazę danych tworzy np. Metropolitan Police Forensic Audio Laboratory w Londynie [18].

W większości sytuacji, aby wykazać brak autentyczności nagrania, wystarczy wizualne porównanie spektrogramów sygnału ENF nagrania z bazą danych ENF. Bardziej szczegółowe badanie przeprowadzamy metodą analizy krótkich okien czasowych. Stabilność sygnału ENF możemy ustalić poprzez porównanie zsynchronizowanych nagrań wykonanych w różnych lokalizacjach geograficznych tej samej sieci. Stosując kryterium ENF w analizie kryminalistycznej dźwięku, możemy określić, czy i gdzie nagranie zostało edytowane, ustalić, czy zostało wykonane w podanym czasie, a także zidentyfikować czas i datę operacji rejestrowania [19].

## 2.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 2.4.1. Uruchamianie stanowiska laboratoryjnego

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Zasilanie komputera i panelu ćwiczeniowego powinno być wyłączone. Moduł pomiarowy powinien być odłączony od portu USB komputera.**

Wszelkie zauważone odstępstwa od tego stanu zgłaszamy prowadzącemu zajęcia.

Sprawdzamy układ zgodnie z opisem w ćwiczeniu 1. Jeśli układ jest połączony prawidłowo, zgłaszamy prowadzącemu zajęcia gotowość do rozpoczęcia realizacji ćwiczenia.

**Dopiero po otrzymaniu od prowadzącego zajęcia wyraźnej zgody włączamy tylko zasilanie komputera. Nie włączamy jeszcze zasilania panelu ćwiczeniowego. Nie dołączamy modułu pomiarowego do portu USB komputera.**

Po uruchomieniu się systemu operacyjnego nie wykonujemy żadnych czynności i przechodzimy do realizacji kolejnego punktu.

### 2.4.2. Self-test modułu NI USB-6001

Pierwszą czynnością po uruchomieniu komputera jest sprawdzenie, czy w systemie są zainstalowane dwa urządzenia NI USB-6001: jedno fizyczne jako **Dev1** i drugie symulowane jako **Dev2**. W tym celu uruchamiamy

program **NI MAX**, rozwijamy listę **Devices and Interfaces** i sprawdzamy zainstalowane w systemie urządzenia (ryc. 1.12). Notujemy w protokole nazwy, typy, numery seryjne i statusy zainstalowanych w systemie urządzeń z serii **NI USB**. Moduł używany na panelu ćwiczeniowym powinien być jeszcze odłączony, a więc powinien być na liście oznaczony czerwonym znakiem „x" i posiadać status **Not Present**. Jeśli na liście nie ma wymaganych urządzeń, musimy je ponownie zainstalować zgodnie z opisem w instrukcji do ćwiczenia 1.

Pamiętamy, że moduł używany i zainstalowany już w systemie na wcześniejszych zajęciach mógł zostać z nieznanych powodów (często przez pomyłkę, przypadkowo) usunięty przez innych studentów. Dlatego zawsze na początku zajęć upewniamy się, jaki numer urządzenia został przypisany w systemie do aktualnie używanego modułu pomiarowego. Sprawdzamy również, czy na każdych zajęciach pracujemy na tym samym stanowisku laboratoryjnym, gdyż każdy wyprodukowany moduł **DAQ** ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczone w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**, jeśli więc wykorzystywany moduł został zainstalowany pod inną nazwą, odpowiednio modyfikujemy tworzone programy.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera. Nie włączamy jeszcze zasilania na panelu ćwiczeniowym.** Sprawdzamy zachowanie niebieskiej diody LED na obudowie modułu. Notujemy wynik obserwacji w protokole.

W programie NI MAX przeprowadzamy *self-test* modułu fizycznego i symulowanego zgodnie z opisem zamieszczonym w instrukcji do pierwszego ćwiczenia (ryc. 1.13). Notujemy rezultaty *self-testu* w protokole.

### 2.4.3. Testowanie przetwornika analogowo-cyfrowego

Przy wyłączonym zasilaniu stanowiska laboratoryjnego uruchamiamy panel testowy (ryc. 1.14) i przeprowadzamy test wejścia analogowego dla następującej konfiguracji: **Channel Name: Dev1/ai1, Mode: Finite, Input Configuration: Differential, Max Input Limit: 10, Min Input Limit: −10, Rate (Hz): 1000, Samples To Read: 100** (ryc. 1.15).

Ponieważ test przeprowadziliśmy przy wyłączonym zasilaniu stanowiska ćwiczeniowego, powinny pojawić się wyniki pomiarów w postaci przypadkowych zakłóceń o wartości kilku mV, a wartość średnia powinna być zbliżona do zera.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody włączamy zasilanie stanowiska ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku powinno potwierdzić obecność napięcia zasilającego.

Ponownie uruchamiamy test. Tym razem powinniśmy uzyskać pięć okresów przebiegu sinusoidalnego napięcia o amplitudzie ok. 8,3 V. Powtarzamy test dla urządzenia symulowanego. Zapisujemy wnioski z przeprowadzonych testów w protokole. Po uzyskaniu pozytywnych wyników testu zamykamy okno **Test Panel** i kończymy pracę programu **Measurement & Automation Explorer**.

## 2.5. Programowanie

### 2.5.1. Pierwszy program obliczający wybrane parametry napięcia sieciowego

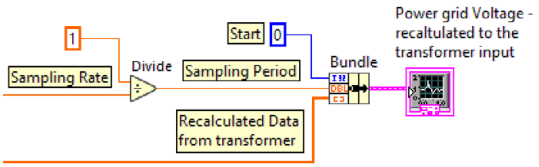
**UWAGA!** Jeśli na poprzednich zajęciach nie zrealizowaliśmy w pełni programu ćwiczenia pierwszego, musimy go dokończyć. Do realizacji ćwiczenia 2 potrzebujemy programu z ostatniego punktu ćwiczenia 1. Na zakończenie ćwiczenia 1 powinniśmy byli wykonać kopię ostatniego zrealizowanego programu o numerze 1-3 i zapisać ją pod nową nazwą, zawierającą numer 2-1-a.

Odszukujemy ten program i uruchamiamy go przyciskiem  . Upewniamy się, czy program działa prawidłowo w obu pozycjach przełącznika **Pomiar/Symulacja**. Zatrzymujemy program przyciskiem **STOP**. Ustawiamy wartość częstotliwości symulowanego sygnału na 50 Hz oraz wartość amplitudy na 8,34 V. Sprawdzamy w algorytmie wartość stałej przetwarzania układu wejściowego. Powinna być zbliżona do wartości 39 V/V, ale obliczona dokładniej na podstawie wskazań woltomierza cyfrowego z poprzednich zajęć. Uruchamiamy ponownie program i sprawdzamy, czy przebiegi w trybie pomiaru i symulacji wyglądają tak samo i czy mają zbliżoną amplitudę. Porównujemy wartości pokazywane przez program ze wskazaniami woltomierza cyfrowego. Zapisujemy wniosek w protokole. Zatrzymujemy program przyciskiem **STOP**. Ustaw przełącznik **Pomiar/Symulacja** w pozycji **Pomiar**. Deklarujemy aktualne nastawy

w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Zapisujemy program z tymi ustawieniami.

Przystępujemy do modyfikacji programu. W tym ćwiczeniu wszystkie obliczenia będziemy realizować na wartościach chwilowych napięcia przeliczonych na pierwotną stronę transformatora izolacyjnego. Dlatego kasujemy z diagramu zbędny już wskaźnik graficzny **Waveform Graph,** pokazujący wyniki bezpośrednio odczytane z **DAQ Assistant**. Kasujemy również z diagramu wskaźnik tablicy z wartościami napięcia z wyjścia **DAQ Assistant**. Elementy skasowane z diagramu powinny również automatycznie zniknąć z panelu. Zostawiamy natomiast wskaźnik graficzny i tablicowy pokazujące wyniki przeliczone na stronę pierwotną transformatora izolacyjnego.

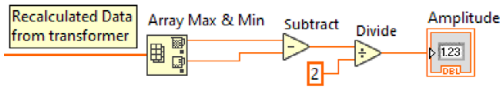
Zwracamy uwagę, że wskaźnik graficzny pokazuje poprawne wartości napięcia w sieci energetycznej, ale nie ma poprawnej skali czasu. Oś pozioma pokazuje numer próbki, a powinna być wyskalowana w jednostkach czasu, czyli w sekundach. W tym celu dane liczbowe z wartościami napięcia uzupełniamy informacjami o skali czasu. Kasujemy tymczasowo połączenie wskaźnika **Waveform Graph** z resztą diagramu. Skalę czasu możemy poprawnie wygenerować na podstawie okresu próbkowania $T_s$ i chwili początkowej $t[0]$. Wszystkie te informacje łączymy w jeden klaster reprezentujący dane typu **Waveform Data Type**. W tym celu umieszczamy na diagramie funkcję **Programming > Cluster & Variant > Bundle** i rozciągamy ją tak, aby uzyskać trzy terminale wejściowe. Na pierwszym od góry wejściu tworzymy prawym kliknięciem wartość stałą równą zero – będzie to początek skali czasu. Obliczamy odwrotność szybkości próbkowania **Sampling Rate** i podłączamy do środkowego wejścia funkcji Bundle – jest to odstęp czasu pomiędzy kolejnymi próbkami. Do dolnego wejścia dołączamy tablicę z przeliczonymi wartościami napięcia. Do wyjścia **Bundle** ponownie dołączamy wskaźnik **Waveform Graph**. Przykład odpowiedniego fragmentu zmodyfikowanego diagramu pokazano na rycinie 2.1.




**Ryc. 2.1.** Fragment diagramu tworzącego skalę czasu dla wykresu napięcia

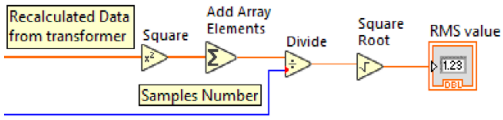
Uruchamiamy program i sprawdzamy, czy wykres ma poprawną skalę czasu i czy zmienia się ona odpowiednio dla innych parametrów próbkowania. Przykład poprawnie wygenerowanej skali czasu pokazano na rycinie 2.6. Zmieniamy parametry próbkowania tak, aby zarejestrować jeden okres napięcia w sieci – powinien on na wykresie trwać 20 ms. Sprawdzamy, czy to się zgadza. Jeśli program działa prawidłowo, zapisujemy go na dysku i robimy jego kopię, zapisując ją jako plik z numerem 2-1-b.

Uzupełniamy program o obliczanie amplitudy napięcia. W tym celu w tablicy z wynikami pomiarów wyszukujemy wartość maksymalną i minimalną, odejmujemy je od siebie i dzielimy przez 2. Wykorzystujemy do tego funkcję **Programming > Array > Max & Min** oraz funkcje odejmowania **Programming > Numeric > Subtract** i dzielenia **Divide**. Na wyjściu funkcji dzielenia tworzymy prawym kliknięciem wskaźnik numeryczny (**Create Indicator**) i nadajemy mu adekwatną nazwę. Przykład odpowiedniego fragmentu diagramu pokazano na rycinie 2.2.




**Ryc. 2.2.** Fragment diagramu obliczający amplitudę napięcia

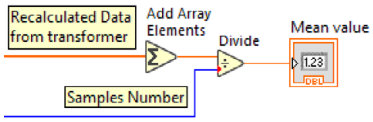
Uzupełniamy program o obliczanie wartości skutecznej napięcia według wzoru (2.14). Przykładowy fragment algorytmu pokazano na rycinie 2.3. Z palety funkcji numerycznych **Programming > Numeric** wykorzystujemy funkcje: **Square**, **Add Array Elements**, **Divide** i **Square Root**. Do dzielenia używamy liczby próbek, która jest zadawana do **DAQ Assistant** jako parametr próbkowania. Na wyjściu funkcji dzielenia tworzymy wskaźnik numeryczny i nadajemy mu adekwatną nazwę.




**Ryc. 2.3.** Fragment diagramu obliczający wartość skuteczną napięcia

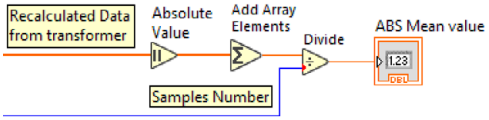
Uzupełniamy program o obliczanie wartości średniej napięcia według wzoru (2.12). Przykładowy fragment algorytmu pokazano na rycinie 2.4.

Z palety funkcji numerycznych **Programming > Numeric** wykorzystujemy funkcje **Add Array Elements** oraz **Divide**. Na wyjściu funkcji dzielenia tworzymy wskaźnik numeryczny i nadajemy mu adekwatną nazwę.




**Ryc. 2.4.** Fragment diagramu obliczający wartość średnią napięcia

Podobnie uzupełniamy program o obliczanie wartości średniej wyprostowanej według wzoru (2.13). Przykładowy fragment algorytmu pokazano na rycinie 2.5. Dodatkowo wykorzystujemy funkcję **Programming > Numeric > Absolute Value**. Na wyjściu funkcji dzielenia tworzymy wskaźnik numeryczny i nadajemy mu adekwatną nazwę.




**Ryc. 2.5.** Fragment diagramu obliczający wartość średnią wyprostowaną napięcia

Uzupełniamy program o obliczanie współczynnika szczytu i współczynnika kształtu, odpowiednio według wzorów (2.7) i (2.8). Wykorzystujemy tylko funkcję dzielenia **Programming > Numeric > Divide** i na jej wyjściu tworzymy odpowiednie wskaźniki numeryczne. Powinniśmy uzyskać diagram pokazany na rycinie 2.7. Uruchamiamy program i sprawdzamy, czy działa poprawnie.

Uaktualniamy na panelu i diagramie niezbędne informacje: dane osobowe zespołu laboratoryjnego, numer grupy dziekańskiej, datę ćwiczenia, numer ćwiczenia, numer programu i jego tytuł. Aktualizujemy opisy wszystkich obiektów zgodnie z ich rolą. Używamy odpowiednich narzędzi, aby wyrównać obiekty, rozmieścić je równomiernie i zmienić kolejność tak, aby nie zasłaniały się wzajemnie. Grupujemy również wszystkie obiekty i blokujemy ich pozycję. Przykład uzupełnionego panelu programu pokazano na rycinie 2.6. Zapisujemy program na dysku, a jego nazwę notujemy w protokole.

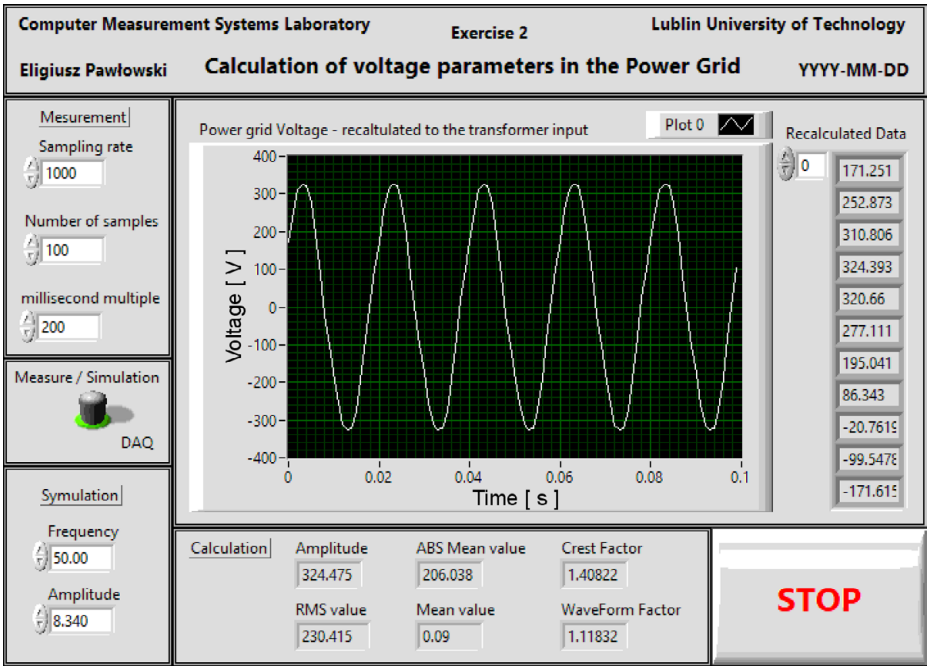
Uruchamiamy program i upewniamy się, czy działa on prawidłowo w obu pozycjach przełącznika. W pozycji symulacji odczytujemy wartość skuteczną, wartość średnią, wartość średnią wyprostowaną oraz współczynniki szczytu i kształtu. Pamiętamy, że dla sygnału sinusoidalnego współczynnik szczytu wynosi 1,4142, a współczynnik kształtu 1,111. Sprawdzamy, czy program prawidłowo oblicza te wartości dla symulowanego sygnału, który możemy traktować jako sygnał wzorcowy. Zapisujemy wnioski w protokole.

Przełączamy program w tryb pomiarów. Mierzymy woltomierzem cyfrowym napięcie skuteczne w sieci zasilającej laboratorium. Porównujemy wynik tego pomiaru ze wskazaniem programu. Sprawdzamy, czy program mierzy poprawnie mierzy wartość skuteczną.

Program oblicza wartość skuteczną na podstawie stałej przetwarzania układu wejściowego, na który składa się transformator izolacyjny i symetryczny, rezystancyjny dzielnik napięcia. Ten fragment układu pomiarowego jest szczegółowo opisany w ćwiczeniu 1 i przedstawiony na rycinie 1.5. Wstępnie w ćwiczeniu 1 przyjęliśmy przybliżoną wartość stałej przetwarzania układu wejściowego równą 39 V/V. Wartość tej stałej jest na każdym stanowisku inna ze względu na rozrzuty produkcyjne parametrów transformatora i rezystorów. Dlatego w ćwiczeniu 1 wyznaczyliśmy dokładniej tę wartość na podstawie pomiarów woltomierzem cyfrowym. Przy założeniu sinusoidalnego kształtu napięcia w sieci przeliczyliśmy w ćwiczeniu 1 wartość skuteczną zmierzoną woltomierzem cyfrowym na amplitudę i podzieliliśmy przez amplitudę napięcia odczytaną z wyjścia przetwornika ADC. Dzieląc przez siebie te dwie wartości, obliczyliśmy rzeczywistą wartość stałej przetwarzania układu wejściowego dla każdego stanowiska. Nie jest to jednak dokładna metoda, ponieważ napięcie w sieci nie jest dokładnie sinusoidalne, a współczynnik szczytu różni się od $\sqrt{2}$. Dlatego nasz program nie mierzy teraz dokładnie wartości skutecznej. Musimy teraz ponownie wyznaczyć wartość stałej przetwarzania układu wejściowego, ale porównując wartości skuteczne napięcia zamiast wartości amplitudy.

Wpisujemy tymczasowo na diagramie wartość stałej przetwarzania układu wejściowego równą 1. Dzięki temu program będzie pokazywał wartości napięcia odczytane bezpośrednio z przetwornika ADC. Uruchamiamy program i przełączamy go w tryb pomiaru napięcia w sieci. Ustawiamy szybkość próbkowania na 10 000 Hz oraz liczbę próbek na 1000, co zapewni większą dokładność

obliczeń dzięki większej liczebności zbioru danych. Mierzymy woltomierzem cyfrowym wartość skuteczną napięcia w sieci i jednocześnie zatrzymujemy program. Notujemy w protokole wartość skuteczną odczytaną z woltomierza i wartość skuteczną odczytaną z programu, uwzględniając co najmniej pięć cyfr znaczących. Wykonujemy odpowiednie obliczenia i notujemy je w protokole: dzielimy wartość skuteczną napięcia zmierzoną w sieci woltomierzem przez wartość skuteczną wskazaną przez program. W ten sposób otrzymujemy dokładną wartość stałej przetwarzania układu wejściowego. Sprawdzamy, czy różni się ona od wstępnie przyjętej wartości 39 V/V oraz od wartości wyznaczonej w ćwiczeniu 1. Wyznaczoną wartość stałej wpisujemy do programu w odpowiednim miejscu diagramu, zachowując co najmniej pięć cyfr znaczących.

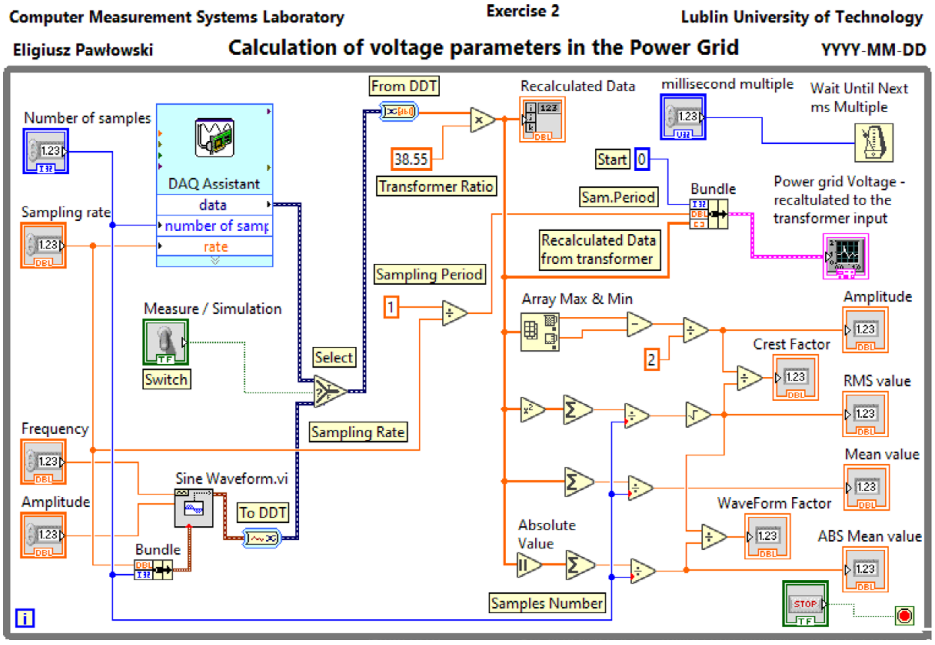



**Ryc. 2.6.** Przykładowy panel programu realizującego pierwsze zadanie

Uruchamiamy program i porównujemy wskazania wartości skutecznej odczytane z woltomierza ze wskazaniami programu. Zapisujemy te dwie wartości w protokole i obliczamy względny błąd procentowy pomiaru dla programu. Wskazanie woltomierza traktujemy jako wzorcowe.

Oceniamy dokładność pomiarów opracowanym programem i sprawdzamy, czy jest ona wyższa niż dokładność przyrządów analogowych.

Analizujemy, ile cyfr wyniku pokazywanego przez program możemy uznać za dokładne i na którym miejscu znaczącym występuje błąd. Następnie ustalamy, ile cyfr znaczących powinien pokazywać program. Zgodnie z ogólną, powszechnie stosowaną zasadą **rozdzielczość przyrządu pomiarowego powinna być lepsza od jego dokładności**. Oznacza to, że powinna być wyświetlana jedna cyfra wyniku więcej za pozycją, na której występuje błąd.




**Ryc. 2.7.** Przykładowy diagram programu realizującego pierwsze zadanie

Program jest równoważny woltomierzowi cyfrowemu o zakresie pomiarowym 250 V. Ustalamy, ile cyfr po przecinku powinno być wyświetlanych zgodnie z przeprowadzonymi pomiarami i obliczeniami. Zapisujemy wnioski w protokole.

Ponownie ustawiamy szybkość próbkowania na 1000 Hz oraz liczbę próbek na 100. Deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Zapisujemy ostateczną wersję programu 2-1 z tymi ustawieniami.

Drukujemy dokumentację programu do pliku i używamy jej do przygotowania raportu. Odszukujemy zapisane pliki na dysku i sprawdzamy ich

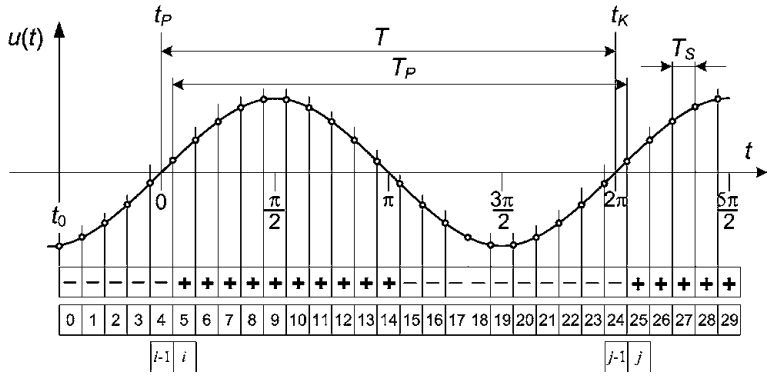
zawartość. Tworzymy kopię programu do realizacji następnego punktu, zmieniając numer programu na 2-2-a.

### 2.5.2. Drugi program do pomiaru częstotliwości napięcia sieciowego

Zgodnie z zależnością (2.2), częstotliwość *f* napięcia w sieci energetycznej możemy wyznaczyć jako odwrotność okresu *T*. Okres, jak sama nazwa wskazuje, jest parametrem sygnałów okresowych (ogólnie funkcji okresowych). Sygnał okresowy to taki rodzaj sygnału, którego wartości powtarzają się co pewien przedział czasu. Najkrótszy czas, po którym wartości sygnału się powtarzają, nazywamy okresem. Możemy to zapisać w postaci równania:

$$u(t) = u(t+N+T), \tag{2.16}$$

gdzie *T* jest okresem napięcia, a *N* jest dowolną liczbą naturalną. Teoretycznie możemy wyznaczyć okres pomiędzy dwiema dowolnie wybranymi wartościami sygnału, np. pomiędzy wartościami maksymalnymi (wierzchołkami sygnału). W praktyce jednak, aby uzyskać wysoką dokładność pomiaru, szczególnie dla sygnału odkształconego oraz w obecności zakłóceń, wybieramy najbardziej stromą część sygnału. Dla przebiegów sinusoidalnych jest to punkt przejścia sygnału przez poziom zerowy, gdyż tam pochodna ma największą wartość. Znanych jest wiele metod pomiaru częstotliwości wykorzystujących punkty przejścia przez zero (*Zero Crossing Technique*) [20–24]. Metodę zastosowaną w ćwiczeniu wyjaśniono na rycinie 2.8 [20].

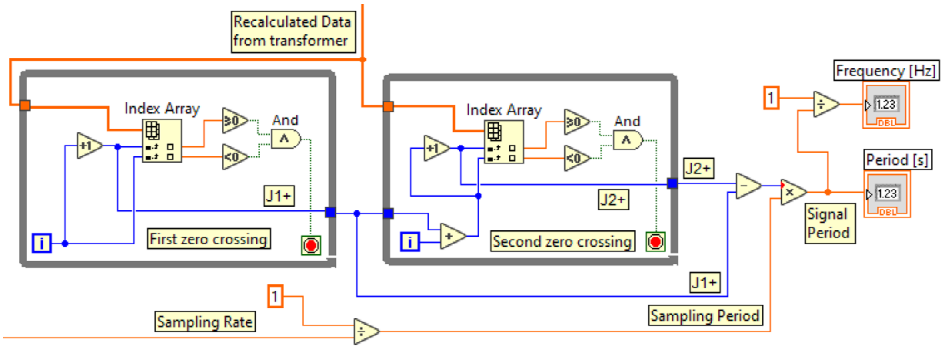



**Ryc. 2.8.** Zasada pomiaru okresu zastosowana w programie

Pokazano na niej 30 pierwszych próbek napięcia obejmujących 1,5 okresu przebiegu napięcia $u(t)$. Okres sygnału $T$ znajduje się pomiędzy chwilą początkową $t_p$ i końcową $t_k$, w których narastające zbocze sygnału przechodzi przez poziom zerowy. Aby ustalić położenie tych punktów, porównujemy znaki dwóch kolejnych próbek sygnału. Pierwsza zmiana znaku z ujemnego na dodatni oznacza, że pomiędzy tymi dwiema próbkami znajduje się miejsce zerowe. Na rycinie 2.8 ma to miejsce pomiędzy próbkami 4 i 5. Przyjmujemy (z pewnym nieuniknionym błędem), że początek zmierzonego okresu *Tp* wyznacza położenie pierwszej dodatniej próbki, czyli piątej próbki (ogólnie próbki o indeksie *i*). Analogicznie koniec okresu *tk* wyznacza położenie 25. próbki (ogólnie próbki o indeksie *j*). Uwzględniając, że sygnał jest próbkowany z szybkością $f_s$, czyli że okres próbkowania wynosi $T_s$, wynik pomiaru okresu możemy obliczyć z zależności:

$$T \approx T_P = (j-i)\,T_S = (j-i)\frac{1}{f_S}. \tag{2.17}$$

Przykładowo na rycinie 2.8 szybkość próbkowania wynosi 1000 SPS, czyli okres próbkowania wynosi 1 ms. Okres sygnału jest równy czasowi pobrania 20 próbek, od próbki $i = 5$ do próbki $j = 25$. Zatem zgodnie ze wzorem (2.17) okres sygnału wynosi $T = 20$ ms, a jego częstotliwość $f = 50$ Hz.




**Ryc. 2.9.** Przykładowy diagram obliczający okres i częstotliwość napięcia

Uzupełniamy teraz program z poprzedniego punktu o pomiar częstotliwości przedstawioną metodą. Uruchamiamy zapisaną kopię programu 2-2a i upewniamy się, że działa prawidłowo. Zatrzymujemy program i uzupełniamy

go fragmentem algorytmu pokazanym na rycinie 2.9. Kompletny diagram programu po niezbędnych uzupełnieniach przedstawiono na rycinie 2.10.

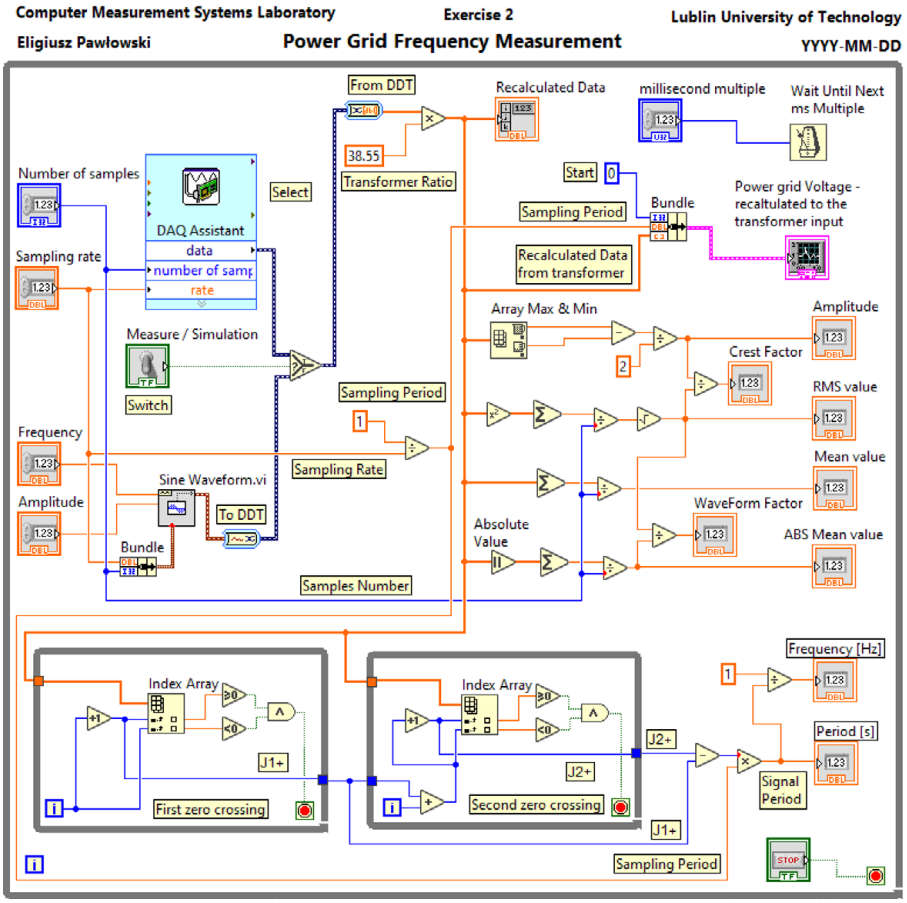
Najpierw w diagramie pokazanym na rycinie 2.7 rozciągamy w dół główną pętlę **While** tak, żeby utworzyć miejsce na dodatkowy fragment algorytmu pokazanego na rycinie 2.9. Główna część tego dodatkowego algorytmu składa się z dwóch podobnych do siebie pętli **While**. Pierwsza pętla przegląda tablicę z próbkami wartości chwilowych napięcia i wyszukuje w niej początek pierwszego okresu sygnału, a druga pętla wyszukuje koniec tego okresu.

Najpierw tworzymy pierwszą (lewą) pętlę, która znajduje w tablicy z próbkami wartości chwilowych napięcia indeks J1+ pierwszej próbki sygnału po jego przejściu przez poziom zerowy w kierunku dodatnim, tzn. na narastającym zboczu sygnału. Przykładowo na rycinie 2.8 jest to próbka o indeksie $i = 5$. Wykorzystujemy do tego następujące funkcje z zasobnika **Programming**:

- **Array > Index Array**,
- **Comparison > Less Than 0?**,
- **Comparison > Greater Or Equal To 0?**,
- **Boolean > And,**
- **Numeric > Increment**.

Z tych funkcji tworzymy fragment algorytmu widoczny w lewej części ryciny 2.9, który następnie obejmujemy pętlą **While**. Tablicę z próbkami napięcia przeszukujemy za pomocą indeksu pętli **i**. Pętla zatrzymuje się po znalezieniu dwóch sąsiednich próbek różniących się znakiem, z których pierwsza jest ujemna, a następna dodatnia. Kopiujemy pętlę **While** wraz z zawartością i umieszczamy tę kopię po prawej stronie. Modyfikujemy prawą pętlę tak, aby znajdowała koniec okresu. Dlatego musi ona przeszukiwać tablicę nie od początku, ale od miejsca, w którym zakończyła pracę pierwsza pętla. W tym celu stosujemy w drugiej pętli dodatkowo funkcję **Numeric > Add**. Tunele w obu pętlach powinny pracować w trybie **Last Value**, co sprawdzamy prawym kliknięciem na krawędzi pętli **While** w miejscu tunelu. Na podstawie indeksów próbek znalezionych przez obie pętle obliczamy okres według wzoru (2.17) i częstotliwość według wzoru (2.2). Do tego celu należy użyć funkcji z palety **Programming > Numeric**: **Subtract**, **Multiply**, **Divide**. Ostatecznie prawym kliknięciem tworzymy dwa wskaźniki numeryczne (**Create Indicator**) do prezentacji zmierzonego okresu i częstotliwości. Odszukujemy te wskaźniki na panelu i umieszczamy je w widocznym miejscu. Uaktualniamy na panelu

i diagramie niezbędne informacje. Przykład diagramu programu i jego panelu pokazano na rycinach 2.10 i 2.11. Zapisujemy ostateczną wersję programu 2-2 na dysku, a jego nazwę notujemy w protokole.



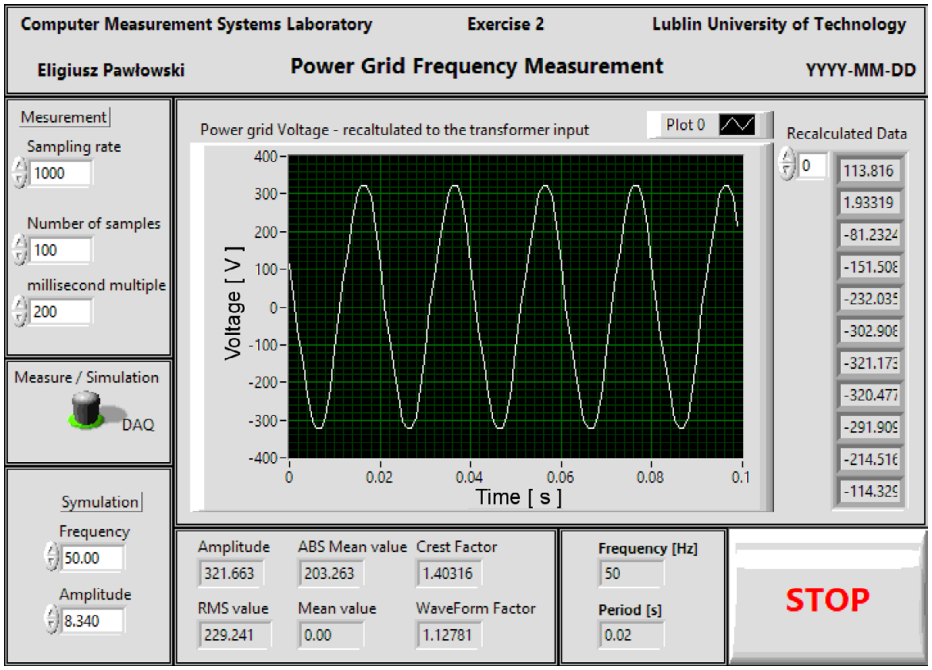

**Ryc. 2.10.** Przykładowy diagram programu realizujący drugie zadanie

Uruchamianie otrzymanego programu wymaga ostrożności. Jeśli pętle wyszukujące przejścia sygnału przez zero zawierają błędy w algorytmie, program może się zawiesić, ponieważ nigdy nie zostanie spełniony warunek zakończenia pętli. W takim przypadku program zatrzymujemy przyciskiem Stop na pasku narzędziowym **LabVIEW**.

Ustawiamy przełącznik w pozycji **Symulacja**. Uruchamiamy program i upewniamy się, czy działa prawidłowo. Zmieniamy symulowaną częstotliwość w górę i w dół o kilka–kilkadziesiąt herców i sprawdzamy, czy program

oblicza prawidłowe wartości okresu i częstotliwości. Oceniamy dokładność pomiarów, a wnioski zapisujemy w protokole.

Zwracamy uwagę, aby nie ustawiać przełącznika w pozycji **Pomiar**, jeśli używamy symulowanego urządzenia **DAQ** na domowym komputerze, i analizujemy, z czego wynika to ograniczenie. Jeśli pracujemy z urządzeniem fizycznym w laboratorium, przełączamy program w tryb pomiarów. Dobieramy opóźnienie w programie, aby otrzymać stabilny obraz przebiegów. Zapisujemy wyniki pomiarów parametrów napięcia sieciowego w protokole. Sprawdzamy, czy są one zgodne z wartościami znamionowymi dla sieci niskiego napięcia.




**Ryc. 2.11.** Przykładowy panel programu realizujący drugie zadanie

Podczas działania programu wyłączamy zasilanie na panelu ćwiczeniowym. Mierzone napięcie powinno spaść do zera. Notujemy wyniki pomiarów w protokole. Zapisujemy uzyskany wykres do pliku i wykorzystujemy go do przygotowania raportu. W raporcie wyjaśniamy zaobserwowany efekt.

Ponownie włączamy napięcie na panelu ćwiczeniowym i sprawdzamy, czy program znowu zaczął ponownie poprawnie mierzyć. Analizujemy przyczyny zaobserwowanego efektu, a wnioski zapisujemy w protokole.

Przełączamy program w tryb symulacji i zatrzymujemy go. Klikamy dwukrotnie moduł **DAQ Assistant** i zmieniamy konfigurację wejścia analogowego do pracy z urządzeniem symulowanym **Dev2**. Uruchamiamy program w trybie symulacji. Zmieniamy symulowaną częstotliwość i upewniamy się, że program działa poprawnie. Ustawiamy przełącznik trybu pracy w pozycji **Pomiar**. Obserwujemy zachowanie programu i opisujemy ten efekt w protokole. Ponownie ustawiamy przełącznik w tryb symulacji i zmieniamy częstotliwość symulowanego przebiegu. Analizujemy i wyjaśniamy nietypowe zachowanie programu, a następnie zapisujemy wnioski w protokole.

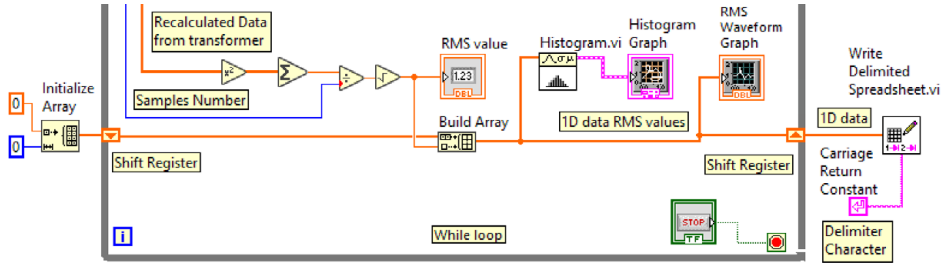
Zatrzymujemy program i ponownie zmieniamy konfigurację wejścia analogowego do pracy z urządzeniem fizycznym **Dev1**. Przełączamy program w tryb rzeczywistych pomiarów. Deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Zapisujemy ostateczną wersję programu 2-2 z tymi ustawieniami.

Drukujemy dokumentację programu do pliku i używamy jej do przygotowania raportu. Odszukujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Tworzymy kopię programu do realizacji następnego punktu, zmieniając odpowiednio numer programu na 2-3-a.

### 2.5.3. Trzeci program z histogramem i rejestracją wyników

Uruchamiamy zapisaną kopię programu 2-3-a i upewniamy się, czy program działa prawidłowo. Pamiętamy, aby nie ustawiać przełącznika w pozycji **Pomiar**, jeśli program jest skonfigurowany do pracy z symulowanym urządzeniem **DAQ**! Ustawiamy przełącznik na pracę symulowaną i, zmieniając symulowaną częstotliwość, sprawdzamy, czy program mierzy ją poprawnie. Przełączamy program na pomiary i sprawdzamy, czy wyniki są zbliżone do wartości znamionowych w sieci energetycznej. Jeśli sprawdzenie wypadło pomyślnie, zatrzymujemy program, naciskając przycisk **STOP** na panelu programu. Jeśli program działa prawidłowo, **nie używamy** przycisku Stop  na pasku narzędziowym! Teraz uzupełniamy program o tworzenie wykresu wartości skutecznej RMS zmierzonego napięcia, tworzenie histogramu z wartości napięcia RMS oraz zapis wyników pomiarów napięcia RMS do pliku. Niezbędne uzupełnienie algorytmu przedstawiono na rycinie 2.12. Kompletny panel nowego programu pokazano na rycinie 2.13, a uzupełniony diagram – na rycinie 2.14. W pierwszej kolejności uzupełniamy diagram tak, aby

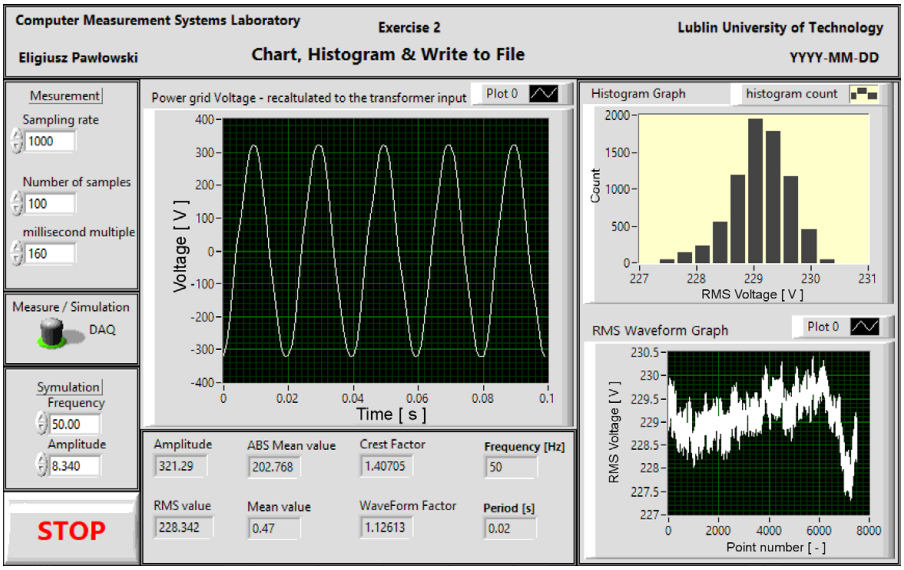
generować histogram na podstawie kolejnych zmierzonych wartości skutecznych RMS. Histogram wykorzystuje dane w formie tablicy (**Array**). Aby utworzyć tablicę z kolejnych wyników pomiaru, tworzymy rejestr przesuwny, klikając prawym przyciskiem myszy (**Add Shift Register**) na krawędzi pętli **While**. Pomiędzy utworzonymi rejestrami przesuwnymi wstawiamy funkcję **Programming > Array > Build Array** i rozciągamy ją w dół, aby utworzyć dwa terminale wejściowe. Łączymy lewy rejestr przesuwny z górnym wejściem, a prawy rejestr z wyjściem funkcji **Build Array**. Do dolnego zacisku tej funkcji dołączamy wartość **RMS Value**. W ten sposób każdy następny wynik pomiaru wartości skutecznej napięcia zostanie dodany do tablicy jako kolejny element. Pustą tablicę inicjujemy za pomocą funkcji **Programming > Array > Initialize Array** na wejściu rejestru przesuwnego znajdującym się na lewej krawędzi pętli **While**. Dodajemy do diagramu funkcję **Mathematics > Prob & Stat > Histogram** i tworzymy prawym kliknięciem (**Create Indicator**) na jej wyjściu wykres **Histogram Graph** przedstawiający histogram ze zgromadzonych wyników pomiarów wartości skutecznych. Przechodzimy na panel i przesuwamy wykres z histogramem w odpowiednie miejsce, wzorując się na rycinie 2.13. Modyfikujemy odpowiednio opisy na osiach histogramu. Uruchamiamy program na kilkadziesiąt sekund i sprawdzamy, czy histogram jest poprawnie generowany. Powinien on mieć kształt zbliżony do krzywej Gaussa.



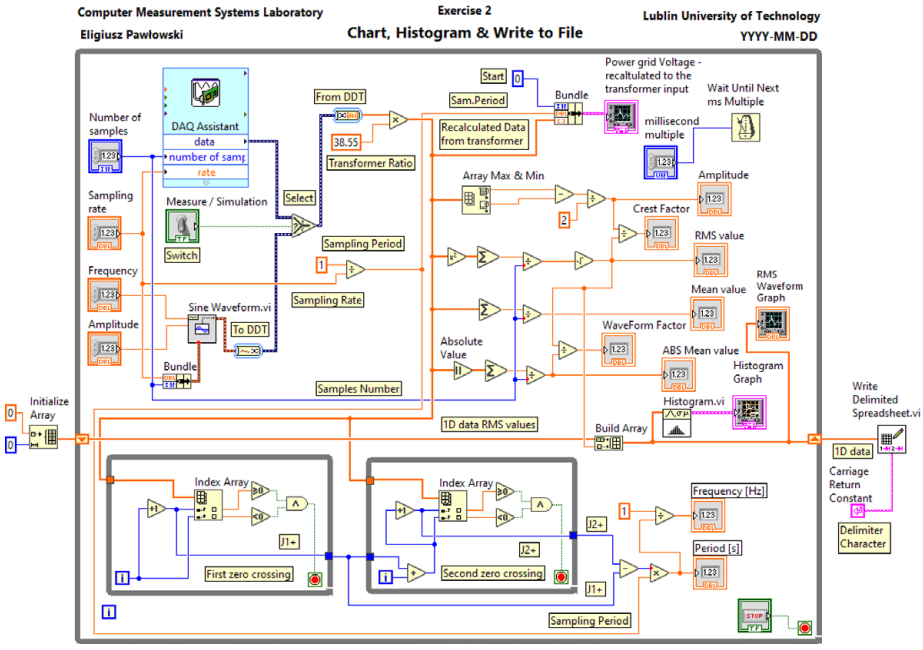

**Ryc. 2.12.** Przykładowy diagram tworzący histogram i zapisujący wyniki do pliku

Uzupełniamy panel programu o wykres pokazujący kolejne wyniki pomiarów zgromadzone w tablicy. W tym celu umieszczamy na panelu obiekt graficzny **Controls > Graph > Waveform Graph**, a na diagramie dołączamy do jego wejścia tablicę z wynikami pomiarów napięcia RMS. Modyfikujemy odpowiednio opisy na osiach wykresu **RMS Waveform Graph** zgodnie

z ryciną 2.13. Uruchamiamy program i sprawdzamy, czy wykres z wartościami napięcia RMS jest poprawnie generowany. Zapisujemy program i tworzymy kopię 2-3-b do dalszej pracy.




**Ryc. 2.13.** Przykładowy panel programu realizującego trzecie zadanie




**Ryc. 2.14.** Przykładowy diagram programu realizującego trzecie zadanie

Uzupełniamy program o zapisywanie wyników pomiaru napięcia RMS w postaci numerycznej do pliku tekstowego na dysku. Aby to zrobić, wykorzystujemy funkcję **Programming > File I/O > Write Delimited Spreadsheet**, umieszczamy ją po prawej stronie (poza główną pętlą **While**), a następnie łączymy wyjście rejestru przesuwnego z jej wejściem **1D Data** i dołączamy stałą tekstową **Programming > String > Carriage Return Constant** do jej wejścia **Delimiter**.

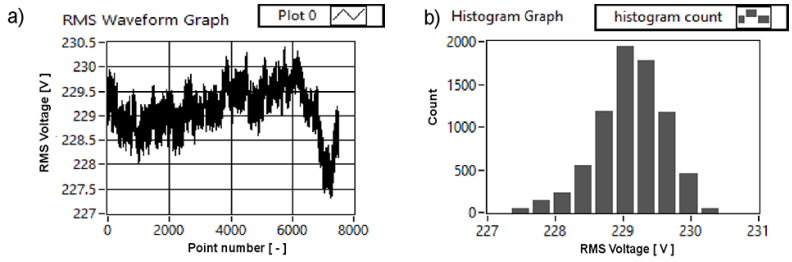
Uruchamiamy program na kilkadziesiąt sekund i zatrzymujemy go przyciskiem STOP na panelu programu. Pamiętamy, aby nie używać przycisku Stop  na pasku narzędziowym **LabVIEW**, gdyż wtedy pętla **While** nie zakończy się poprawnie, a program nie zapisze wyników pomiarów do pliku! Po zatrzymaniu programu powinno otworzyć się okienko dialogowe zapisu pliku z wynikami, w którym wpisujemy nazwę pliku. W nazwie pliku dodajemy po kropce rozszerzenie .txt lub .csv, co ułatwi otworzenie pliku w Notatniku albo w Excelu. Sprawdzamy, czy plik został zapisany oraz czy zawiera liczbowe wyniki pomiarów w postaci jednej kolumny danych.

Jeśli program działa poprawnie, wykonujemy przykładową serię pomiarów do sporządzenia raportu. Jeśli pracujemy w laboratorium i używamy rzeczywistego, fizycznego urządzenia **DAQ**, ustawiamy przełącznik w pozycji **Pomiar**. Uruchamiamy program i sprawdzamy, czy działa prawidłowo. Pamiętamy, aby nie ustawiać przełącznika w pozycji **Pomiar**, jeśli używamy symulowanego urządzenia **DAQ –** w takim przypadku ustawiamy przełącznik w pozycji **Symulacja**.

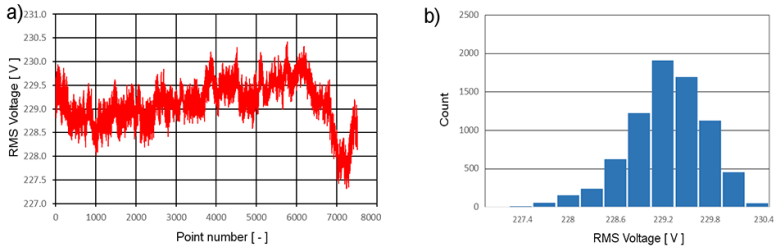
Rejestrujemy napięcie RMS w sieci energetycznej przez kilka minut. Jeśli używamy symulowanego **DAQ**, podczas pomiaru zmieniamy losowo co chwilę symulowaną amplitudę o kilka woltów. W ten sposób zasymulujemy naturalne wahania napięcia występujące w rzeczywistej sieci energetycznej. Po zatrzymaniu programu przyciskiem **STOP** na panelu wprowadzamy nazwę pliku z rozszerzeniem .txt lub .csv i zapisujemy wyniki do pliku. Za pomocą opcji **Export Simplified Image** zapisujemy do plików graficznych BMP otrzymany histogram oraz wykres napięcia RMS. Wykorzystujemy je następnie do przygotowania raportu.

W raporcie zamieszczamy również histogram i wykres opracowane za pomocą programu Excel na podstawie wyników liczbowych zapisanych do pliku tekstowego. W raporcie przedstawiamy obok siebie wykresy

wygenerowane przez program w **LabVIEW** oraz opracowane samodzielnie w programie Excel. Przykładowe rezultaty działania programu w **LabVIEW** przedstawiono na rycinie 2.15a, a wykresy opracowane w Excelu – na rycinie 2.16a. Zwracamy uwagę na podobieństwo tych wykresów.




**Ryc. 2.15.** Przykładowe wykresy wygenerowane na panelu w programie LabVIEW: a) wartości skuteczne napięcia w sieci, b) histogram wartości skutecznych napięcia




**Ryc. 2.16.** Wykresy opracowane w programie Excel z danych przedstawionych na rysunku 2.15, zapisanych automatycznie przez program LabVIEW w postaci numerycznej do pliku tekstowego: a) wartości skuteczne napięcia w sieci, b) histogram wartości skutecznych napięcia

Zapisujemy ostateczną wersję programu 2-3 na dysku, a jego nazwę notujemy w protokole. Drukujemy dokumentację programu do pliku i używamy jej do przygotowania protokołu. Odszukujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Tworzymy kopię programu do realizacji następnego punktu, zmieniając numer programu na 2-4-a.

### 2.5.4. Czwarty program do dokładnych pomiarów częstotliwości

Program w obecnej postaci mierzy okres na podstawie liczby próbek odległych od siebie o 1 ms. Ponieważ znamionowo okres napięcia w sieci wynosi 20 ms, błąd pomiaru może sięgać maksymalnie ±5 %, co oznacza, że częstotliwość

o wartości znamionowej 50 Hz jest mierzona z błędem rzędu ±2,5 Hz. Jest to bardzo duży błąd, ponieważ w praktyce częstotliwość napięcia podczas normalnej pracy sieci zmienia się w granicach ±0,02 Hz. Kontynentalny Europejski System Energetyczny stanowi jedną całość – od Portugalii aż po Ukrainę i Turcję. Wszystkie urządzenia dołączone do sieci energetycznej na tym obszarze pracują synchronicznie. Przekroczenie granicy 50 Hz±2,5 Hz spowodowałoby automatyczne rozłączenie wszystkich generatorów pracujących w tej sieci. Największy zanotowany w ostatnim czasie spadek częstotliwości sieci miał miejsce 8 stycznia 2021 r. Tego dnia o godz. 14:05:07 zanotowano spadek częstotliwości poniżej 49,75 Hz, co w konsekwencji spowodowało rozłączenie europejskiej sieci na dwie części wzdłuż linii przechodzącej przez Chorwację, Serbię i Rumunię. Przyczyną była awaria na podstacji Ernestinovo w Chorwacji. Zdarzają się również mniejsze, ale długotrwałe obniżenia częstotliwości w sieci. W idealnych warunkach pracy sieci utrzymywana jest stała liczba 4 mln 320 tys. okresów w ciągu każdej doby ($24 \times 60 \times 60 \times 50$), zatem średnia wartość częstotliwości zmierzona w ciągu całej doby powinna wynosić dokładnie 50 Hz. Dzięki temu częstotliwość sieci może być używana w wielu zastosowaniach jako częstotliwość wzorcowa. Powszechnie korzystają z niej zegary w sprzęcie AGD stale podłączonym do sieci (piekarniki, płyty grzewcze, mikrofalówki itp.), ale korzysta z niej również wiele urządzeń przemysłowych. W styczniu 2018 r. wartość średnia częstotliwości obniżyła się na długi czas do 49,996 Hz, co skutkowało spóźnianiem się zegarów dołączonych do sieci energetycznej o 7 s/dobę. Zanim rozwiązano problem, łączne spóźnienie zegarów wyniosło ponad 6 minut. W mediach pojawiły się alarmujące doniesienia o „złodziejach czasu". Powodem były problemy z produkcją energii elektrycznej w Kosowie, ale faktyczne przyczyny miały podłoże polityczne.

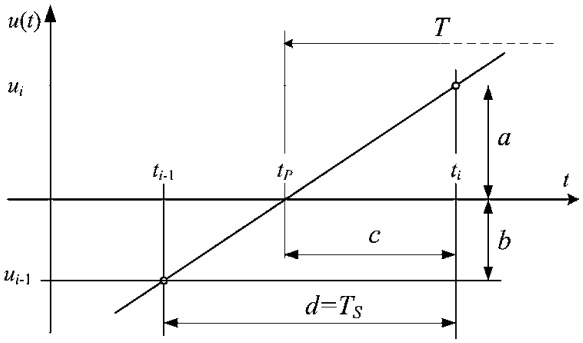
Aktualne informacje na temat średniej częstotliwości i odchylenia czasowego względem UTC możemy znaleźć na stronie internetowej szwajcarskiego operatora systemu przesyłowego Swissgrid, który odpowiada za monitorowanie częstotliwości dla Europy kontynentalnej. Aktualną wartość częstotliwości możemy sprawdzić na stronach: www.swissgrid.ch, gridradar.net lub www.mainsfrequency.com. Jeśli czas sieci różni się od czasu UTC o więcej niż +20 s pod koniec doby, punkt nastawy częstotliwości jest zmniejszany o 10 mHz (do 49,990 Hz). W przypadku ujemnego odchylenia czasu sieci o 20 s w porównaniu do czasu UTC punkt nastawy częstotliwości jest zwiększany o 10 mHz

(do 50,010 Hz). Przy zmianach częstotliwości mniejszych niż ±0,01 Hz, żadne mechanizmy regulacji nie są uruchamiane.

Modyfikujemy teraz program tak, aby umożliwiał pomiar częstotliwości napięcia w sieci z dokładnością pozwalającą porównać wyniki z wartościami podawanymi przez szwajcarskiego operatora Swissgrid. W tym celu musimy dokładniej wyznaczyć chwilę przejścia napięcia przez poziom zero [22–24]. Zasadę pomiaru wyjaśniono na rycinie 2.17. Korzystamy ze znanej właściwości funkcji sinus:

$$\lim_{x\to 0}\frac{\sin x}{x}=1, \tag{2.18}$$

z której wynika, że funkcja sinus w pobliżu zera zbliża się do linii prostej. Dla przykładu, jak wiadomo, sin 30° = 0,5, przy czym kąt 30° jest równy w mierze łukowej π/6, czyli około 0,524 radiana. Różnica pomiędzy funkcją sinus i linią prostą wynosi więc niecałe 5 % w punkcie stosunkowo odległym od punktu zerowego. Przy próbkowaniu z szybkością 1000 SPS próbki są odległe od siebie o 1 ms, co odpowiada 1/20 okresu napięcia przy częstotliwości 50 Hz. Jest to równoważne kątowi 18°, czyli π/10 = 0,314 radiana. Natomiast sin 18° = 0,309, zatem błąd względem linii prostej wynosi już tylko 1,5 %. To uzasadnia wyznaczenie punktu przejścia sinusoidy przez zero jako miejsca zerowego linii prostej przechodzącej przez dwa punkty.




**Ryc. 2.17.** Zasada wyznaczania chwili przejścia napięcia przez poziom zerowy

Na rycinie 2.17 przedstawiono sposób wyznaczania chwili początkowej $t_p$ mierzonego okresu $T$ napięcia $u(t)$. Próbki napięcia *ui*-1 w chwili *ti*-1 oraz *ui* w chwili *ti* są przeciwnego znaku, a więc chwila *tp* na pewno leży pomiędzy

*ti*-1 oraz *ti*. Korzystamy z twierdzenia Talesa. Na podstawie ryciny 2.17 można ułożyć proporcję:

$$\frac{a}{c} = \frac{a+b}{d}, \tag{2.19}$$

gdzie: *a* jest wartością próbki *ui*, *b* jest wartością próbki *ui*-1, *c* wyznacza położenie chwili *tp* względem chwili próbkowania *ti*, *d* jest okresem próbkowania *TS*. Po podstawieniu tych wartości do zależności (2.19) i wykonaniu nieskomplikowanych przekształceń, otrzymujemy wzór pozwalający obliczyć chwilę początkową *tp*:

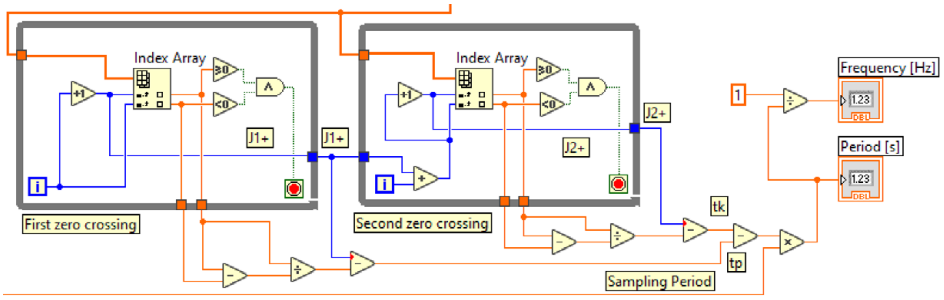
$$t_P = \left( i - \frac{u_i}{u_i - u_{i-1}} \right) T_S . \tag{2.20}$$

Na rycinie 2.18 przedstawiono odpowiednio uzupełniony algorytm pomiaru częstotliwości wykorzystujący sposób wyznaczania chwili początkowej $t_{p\,\mathrm{w}}$edług zależności (2.20). Chwilę końcową *tk* wyznaczamy analogicznie z tej samej zależności (2.20). Najpierw uzupełniamy lewą pętlę o obliczenie chwili początkowej *tp* zgodnie ze wzorem (2.20), a następnie w identyczny sposób uzupełniamy pętlę prawą o obliczenia chwili końcowej *tk*. Następnie obliczamy okres *T* sygnału odejmując od siebie te dwie obliczone wartości i mnożąc przez okres próbkowania:

$$T = \left( t_k - t_p \right) T_S . \tag{2.21}$$

Uruchamiamy program i sprawdzamy jego działanie. Jeśli wyniki pomiarów w sieci są zgodne z przewidywaniami, przełączamy program na tryb symulowanych pomiarów. Ustawiamy symulowaną częstotliwość na 50 Hz. Sprawdzamy, czy program zmierzył ją poprawnie. Ustawiamy częstotliwość 50,1 Hz. Program w poprzedniej wersji nie mierzył poprawnie takiej częstotliwości. Weryfikujemy, czy program w obecnej postaci mierzy ją poprawnie. Obserwujemy wyniki i zauważamy, że zmieniają się one w powtarzalny sposób wokół wartości dokładnej. Zapisujemy największą i najmniejszą wartość zmierzonej częstotliwości. Wynik pomiaru częstotliwości powinien być pokazywany z czterema miejscami po przecinku; w razie potrzeby zmieniamy ustawienia dla tego wskaźnika

w zakładce **Properties**. Ustawiamy symulowaną częstotliwość na 50,01 Hz, a następnie na 50,001 Hz. Sprawdzamy, czy program potrafi zmierzyć tak niewielkie różnice częstotliwości względem wartości znamionowej 50,000 Hz. Wyniki obserwacji zapisujemy w protokole.



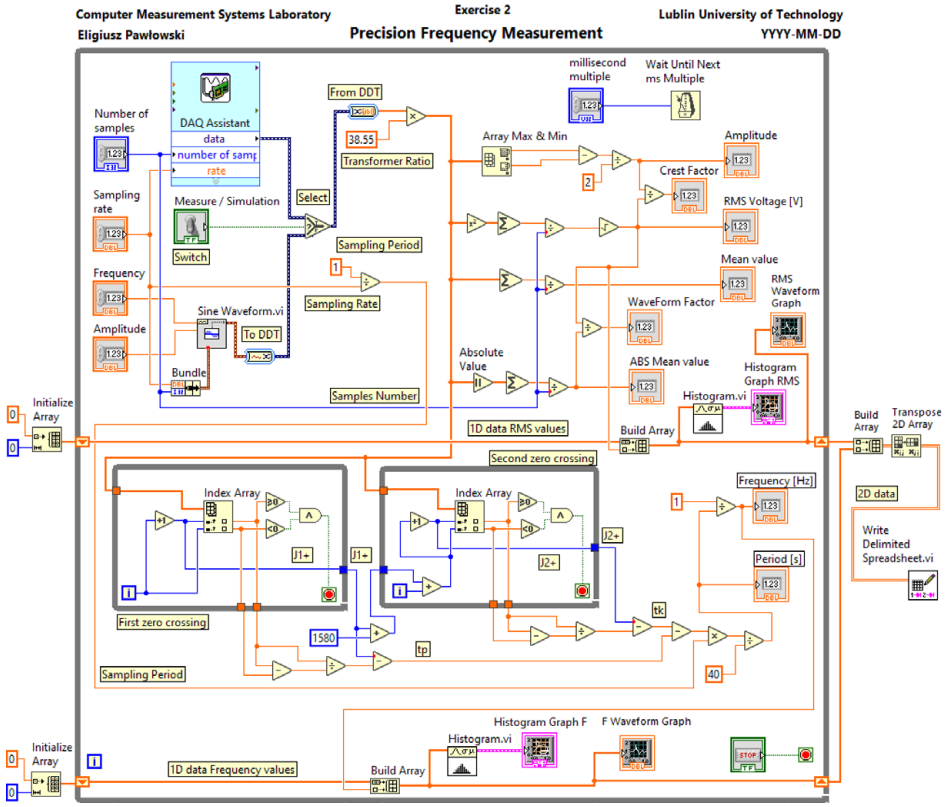

**Ryc. 2.18.** Diagram algorytmu dokładnego wyznaczania przejść przez zero dla jednego okresu

Program w obecnej postaci mierzy częstotliwość napięcia w sieci znacznie dokładniej, ale modyfikujemy go dodatkowo tak, aby móc porównywać jago wskazania z wynikami prezentowanymi na stronie szwajcarskiego operatora Swissgrid. W tym celu wprowadzamy dwie zmiany: zwiększymy szybkość próbkowania do 2000 SPS oraz wydłużamy czas pomiaru z jednego do 40 okresów. W związku z tym najpierw zwiększymy liczbę próbek ze 100 do 1700. Pomiary wykonujemy co sekundę, tak jak u operatora Swissgrid. Dodatkowo wyniki pomiarów częstotliwości zapisujemy do pliku.

Zapisujemy program w obecnej postaci i tworzymy kopię 2-4-b do dalszej pracy. Ustawiamy na panelu nowe wartości parametrów próbkowania: szybkość próbkowania 2000 SPS, liczbę próbek 1700 oraz opóźnienie czasowe 1000 ms. Deklarujemy te nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i od razu zapisujemy program z tymi wartościami na dysku, zanim zaczniemy go modyfikować. Aby program mierzył 40 okresów, druga pętla **While** powinna zacząć przeszukiwać tablicę próbek z przesunięciem 39,5 okresu względem próbki znalezionej przez pierwszą pętlę. Przy szybkości próbkowania 2000 SPS w jednym okresie trwającym 20 ms mieści się 40 próbek, dlatego pomiędzy pierwszą a drugą pętlą musimy dodać 1580 próbek. Uzupełniamy diagram o operację dodawania liczby 1580 pomiędzy wyjściem pierwszej pętli a wejściem pętli drugiej, zgodnie z ryciną 2.19.

Dzięki temu druga pętla rozpocznie poszukiwanie przejścia przez zero w połowie czterdziestego okresu. W tej uproszczonej metodzie zakres pomiaru

okresu wyniesie więc ±0,5 okresu przy 40 mierzonych okresach, co oznacza, że dopuszczalne są względne zmiany rzędu ±1,25 %. Stąd wynika zakres poprawnie mierzonych częstotliwości od 49,375 do 50,625 Hz. Jest to zakres całkowicie wystarczający, ponieważ zgodnie z przepisami zmiany częstotliwości w poprawnie działającej sieci nie przekraczają ±1 % wartości znamionowej częstotliwości 50 Hz, czyli ±0,5 Hz. Przy dużych zmianach częstotliwości możemy zmierzyć 39 lub 41 okresów zamiast 40, co spowodowałoby błąd pomiaru. Zbadamy ten problem, dysponując gotowym programem.



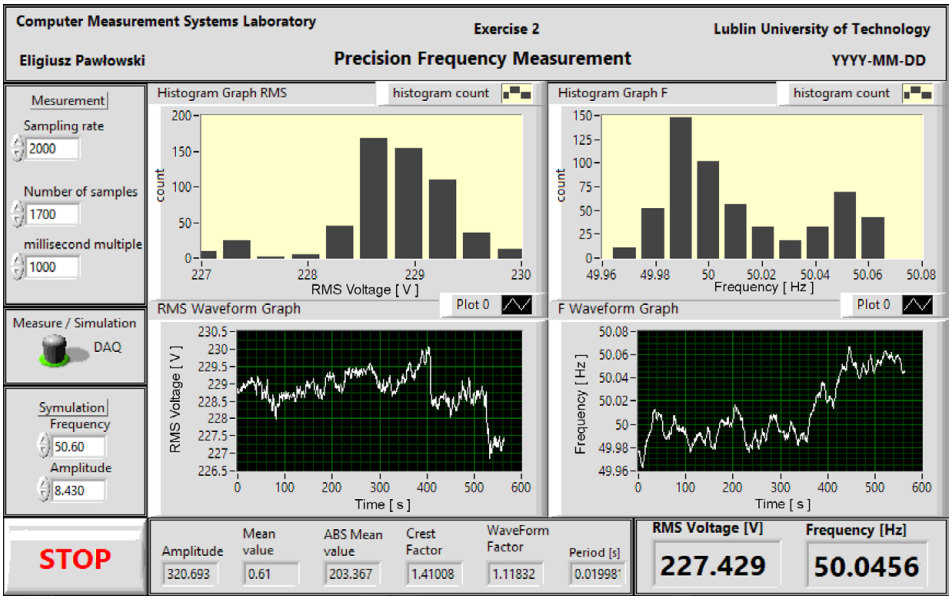

**Ryc. 2.19.** Przykładowy program do dokładnego pomiaru częstotliwości

Kończąc modyfikację algorytmu, w celu obliczenia średniej wartości okresu $T$, wyznaczony przez obie pętle przedział czasu $t_k - t_p$ dzielimy przez 40. Przykładową postać algorytmu po tych modyfikacjach pokazano na rycinie 2.19. Zapisujemy

program na dysku i sprawdzamy jego działanie. W tym celu najpierw uruchamiamy program w trybie symulacji.

Ustawiamy kolejno symulowane częstotliwości na 50,001 Hz, 50,01 Hz i 50,1 Hz. Sprawdzamy, czy program w obecnej postaci mierzy poprawnie tak niewielkie różnice częstotliwości względem wartości znamionowej 50,000 Hz. Podobnie jak poprzednio, zauważamy, że wyniki zmieniają się w powtarzalny sposób wokół wartości dokładnej. Zapisujemy największą i najmniejszą wartość zmierzonej częstotliwości dla każdej ustawionej wartości. Oceniamy, czy program po wprowadzonych modyfikacjach mierzy częstotliwość dokładniej niż poprzednia wersja. Wyniki obserwacji zapisujemy w protokole. Zapisujemy program w obecnej postaci i tworzymy kopię 2-4-c do dalszej pracy.

W kolejnym kroku pracy nad programem uzupełniamy go o tworzenie wykresu i histogramu na podstawie wyników pomiaru częstotliwości oraz zapis wyników na dysku. Tworzenie wykresu i histogramu wymaga zbudowania tablicy z wynikami pomiarów częstotliwości. Ten fragment algorytmu realizujemy identycznie jak w poprzednim punkcie, w którym zbudowaliśmy tablicę z wynikami pomiaru napięcia skutecznego RMS. Dodatkowy fragment diagramu wygodnie jest umieścić w dolnej części głównej pętli **While** programu, jak pokazano na rycinie 2.19.




**Ryc. 2.20.** Przykładowy panel programu do dokładnego pomiaru częstotliwości

Skupiamy się teraz na wartościach skutecznych napięcia i na częstotliwości. Na panelu usuwamy wykres i tablicę wartości chwilowych napięcia jako już nieprzydatne w tej wersji programu. Na diagramie pojawią się wówczas błędne połączenia, które musimy usunąć. Rozmieszczamy na panelu dwa wykresy i dwa histogramy oraz uaktualniamy opisy na osiach, podając odpowiednie nazwy wielkości i jednostki miary. Porządkujemy pozostałe obiekty. Przykładową postać panelu po tych modyfikacjach pokazano na rycinie 2.20.

Zapisujemy program na dysku i sprawdzamy jego działanie. W tym celu uruchamiamy program w trybie pomiarów rzeczywistych i obserwujemy tworzenie wykresów i histogramów dla pomiarów napięcia i częstotliwości.

Jeśli program pracuje zgodnie z oczekiwaniami, zapisujemy go w obecnej postaci na dysku i tworzymy kopię 2-4-d do dalszej pracy. W ostatnim kroku pracy nad programem uzupełniamy go o zapis wyników do pliku.

W obecnej postaci program zapisuje do pliku jednowymiarową tablicę 1D z wynikami pomiarów wartości skutecznej napięcia RMS, z których możemy wykonać wykres i histogram w programie Excel, pokazane przykładowo na rycinie 2.16. Teraz tworzymy dwuwymiarową tablicę 2D z dwoma kolumnami zawierającymi wyniki pomiarów napięcia i częstotliwości. Odpowiednie uzupełnienia w diagramie pokazano na rycinie 2.19, po prawej stronie pętli While. W tym celu wykorzystujemy funkcje **Programming > Array > Build Array** oraz **Transpose 2D Array**. Usuwamy natomiast stałą tekstową **Carriage Return Constant** z wejścia **Delimiter** funkcji **Write Delimited Spreadsheet**.

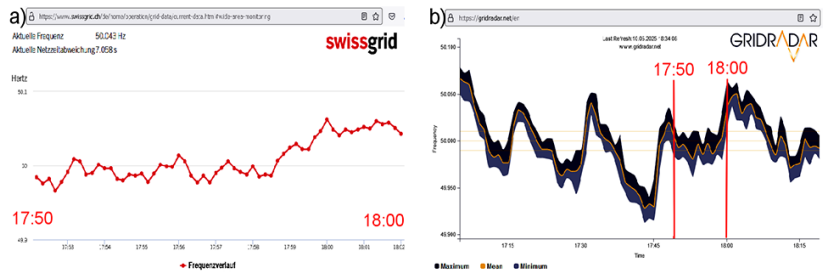
Zapisujemy program na dysku i sprawdzamy jego działanie. W tym celu uruchamiamy program w trybie pomiarów rzeczywistych i obserwujemy tworzenie wykresów i histogramów dla pomiarów napięcia i częstotliwości. Po kilkudziesięciu sekundach zatrzymujemy program przyciskiem **STOP** na panelu programu. Po zatrzymaniu programu powinno otworzyć się okienko dialogowe zapisu pliku. W nazwie pliku dodajemy po kropce rozszerzenie .txt albo .csv. Sprawdzamy, czy plik został zapisany oraz czy zawiera wyniki pomiarów w postaci dwóch kolumn danych. W lewej kolumnie powinny znajdować się wartości napięcia, a w prawej – częstotliwości. Próbujemy otworzyć plik w programie Excel i wygenerować wykres, który powinien być zgodny z wykresem na panelu programu.

Jeśli program pracuje zgodnie z oczekiwaniami, zapisujemy jego ostateczną postać na dysku. Drukujemy dokumentację programu do pliku i używamy jej do przygotowania raportu. Odszukujemy zapisane pliki na dysku i sprawdzamy ich zawartość.

## 2.6. Pomiary częstotliwości i ocena dokładności

W Internecie dostępne są strony na bieżąco pokazujące aktualną częstotliwość w europejskiej sieci energetycznej. Najbardziej wiarygodne dane udostępnia szwajcarski operator *Swissgrid* na swojej stronie: www.swissgrid.ch. Dostępny tam wykres pokazuje wyniki z ostatnich 10 minut, a pomiary są uśredniane w kolejnych przedziałach czasu równych 10 sekund. Na stronie *Mains frequency* (www.mainsfrequency.com) dostępny jest wykres z ostatnich 5 minut oparty na pomiarach wykonywanych co sekundę, ale witryna ta często nie działa poprawnie. Na stronie *Gridradar* (gridradar.net) dostępny jest wykres z pomiarami z ostatniej godziny, uśrednianymi co minutę. W Internecie możemy znaleźć również inne witryny udostępniające wyniki pomiarów częstotliwości, w tym brytyjskie czy amerykańskie. Pamiętajmy jednak, że sieci te nie są zsysnchronizowane z europejską siecią energetyczną.

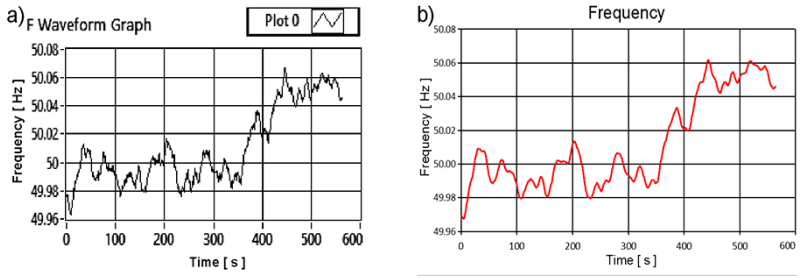
Na rycinie 2.21 pokazano przykładowe wykresy pobrane ze stron *Swissgrid* oraz *Gridradar*, prezentujące rzeczywiste wyniki pomiarów częstotliwości w europejskiej sieci energetycznej w tym samym przedziale czasu. Zauważamy, że wyniki mieszczą się w przedziale częstotliwości od 49,9 Hz do 50,1 Hz.




**Ryc. 2.21.** Przykładowe zmiany częstotliwość w europejskiej sieci energetycznej: a) wykres dostępny na stronie Swissgrid, wykres dostępny na stronie Gridradar

Otwieramy na komputerze lub telefonie stronę *Swissgrid* i odnajdujemy zakładkę z pomiarami częstotliwości. Uruchamiamy program **LabVIEW** w trybie pomiarów rzeczywistych. Porównujemy na bieżąco wyniki na stronie *Swissgrid* z wynikami prezentowanymi przez program. Wyniki nie są identyczne, ponieważ pomiary nie są wykonywane dokładnie w tych samych chwilach, a częstotliwość stale ulega wahaniom. Jednak wyniki powinny być zbliżone i wykazywać taką samą tendencję zmian. Po 10 minutach, gdy wykres na stronie *Swissgrid* zapełni się wynikami, zatrzymujemy program **LabVIEW** i wykonujemy zrzut ekranu z wykresem na stronie *Swissgrid*. Zapisujemy wyniki pomiarów do pliku tekstowego i notujemy nazwę pliku w protokole. Pamiętamy, aby dodać w nazwie pliku rozszerzenie .txt lub .csv. Zapisujemy wykresy z panelu programu do plików graficznych BMP za pomocą opcji **Export Simplified Image**. Sprawdzamy zawartość zapisanych plików.

W raporcie należy wykonać wykresy w programie Excel na podstawie wyników zapisanych do pliku tekstowego. Przykładowe wykresy pomiarów wykonanych w laboratorium w Lublinie pokazano na rycinie 2.22. Wykres otrzymany z programu pokazuje wyniki rejestrowane co jedną sekundę. Wykres w Excelu został wykonany po wcześniejszym uśrednieniu wyników z 10 sekund, co pozwala na lepsze porównanie z wykresem na stronie *Swissgrid*.




**Ryc. 2.22.** Przykładowe zmiany częstotliwości w sieci energetycznej w tym samym przedziale czasu co na rysunku 2.21, zarejestrowane opracowanym programem: a) wykres wygenerowany na panelu programu w LabVIEW, b) ten sam wykres opracowany w Excelu na podstawie danych numerycznych zapisanych przez program

Wszystkie przykładowe wykresy na rycinach 2.21 i 2.22 obejmują ten sam przedział czasu. Widać na nich wyraźną zgodność wyników pomiarów wykonanych w Szwajcarii i dostępnych w Internecie z pomiarami wykonanymi

przez program w laboratorium Politechniki Lubelskiej. Wykonując pomiary podczas zajęć, powinniśmy uzyskać podobną zgodność zarówno w wartościach częstotliwości, jak i w tendencjach zmian. W raporcie należy ocenić uzyskane wyniki pod tym względem.

## 2.7. Opracowanie raportu z realizacji ćwiczenia

Raport z przeprowadzonego ćwiczenia powinienzawierać:

- stronę tytułową z danymi osobowymi zespołu, datą ćwiczenia i jego tytułem,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań zgodnie z programem ćwiczenia,
- opis każdego z wykonanych zadań lub podanie przyczyny jego niewykonania,
- dokumentację czterech zrealizowanych i uruchomionych programów,
- wykresy i histogramy wygenerowane przez program w środowisku **LabVIEW**,
- wykresy i histogramy opracowane w arkuszu kalkulacyjnym Excel,
- wykresy pobrane z Internetu,
- liczbowe wyniki pomiarów uzyskane przy użyciu przygotowanych programów,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.

W raporcieocenie podlegają:

- poprawność działania programów,
- czytelność i przejrzystość diagramu,
- estetyka panelu, dobór kolorów i elementów graficznych,
- funkcjonalność panelu pod kątem wygody obsługi,
- zakres zrealizowanych zadań,
- trafność wniosków końcowych.

# 3. Pomiary częstotliwości sygnałów cyfrowych

## 3.1. Cel ćwiczenia i zadania do wykonania

Celem ćwiczenia 3 jest zapoznanie się z problematyką programowania komputerowego systemu pomiarowego w środowisku **LabVIEW** z wykorzystaniem funkcjonalności uniwersalnego licznika impulsów, będącego częścią modułu akwizycji danych pomiarowych NI USB-6001.

Program ćwiczenia 3 obejmuje następujące zagadnienia:

- testowanie licznika impulsów w programie NI-MAX,
- konfigurowanie modułu NI USB-6001 do zliczania impulsów w środowisku **LabVIEW**,
- symulacja zliczanych impulsów w trybie symulacji modułu **DAQmx**,
- pomiary częstotliwości poprzez zliczanie impulsów licznikiem z programowym odmierzaniem czasu pomiaru,
- dodatkowa analiza sygnału: tworzenie histogramu, rejestracja i zapis wyników pomiarów do pliku oraz tworzenie wykresów.

Podczas ćwiczenia przygotowujemy odpowiednie programy w środowisku **LabVIEW**, uruchamiamy je, sprawdzamy poprawność ich działania, usuwamy błędy, wykonujemy pomiary i sporządzamy protokół.

W ramach ćwiczenia 3 opracowujemy cztery programy:

1. Pierwszy program – odczytujący zawartość licznika z możliwością symulacji liczby impulsów dla symulowanego modułu **DAQ**.
2. Drugi program – uzupełniony o zadawanie częstotliwości symulowanych impulsów.
3. Trzeci program – mierzący częstotliwość impulsów rzeczywistych i symulowanych.
4. Czwarty program – uzupełniony o dodatkową analizę wyników, obliczanie błędów, szacowanie niepewności, tworzenie wykresu z wynikami pomiarów, budowanie histogramu.

## 3.2. Opis stanowiska laboratoryjnego

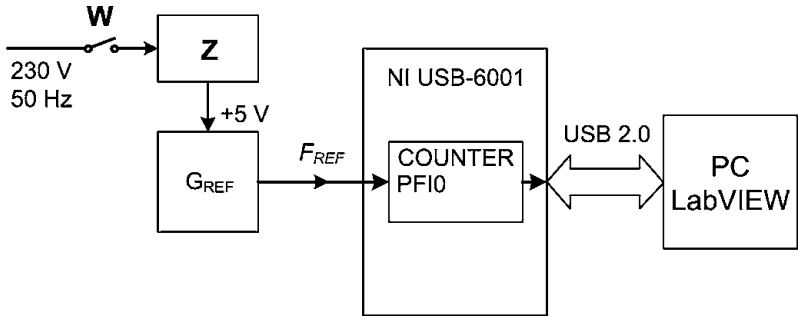
### 3.2.1. Schemat blokowy stanowiska laboratoryjnego

Stanowisko laboratoryjne wykorzystane w ćwiczeniu 3 składa się z następujących komponentów:

- komputera PC z zainstalowanym środowiskiem LabVIEW,
- modułu akwizycji danych pomiarowych NI USB-6001,
- kwarcowego generatora impulsów TTL o wzorcowych częstotliwościach.

Schemat blokowy stanowiska laboratoryjnego wykorzystywanego w ćwiczeniu 3 przedstawiono na rycinie 3.1. Moduł akwizycji danych pomiarowych NI USB-6001 jest dołączony do komputera PC za pomocą interfejsu USB. Wyjście $F_{Ref}$ generatora częstotliwości wzorcowych (referencyjnych) $G_{\mathrm{Ref}}$ jest podłączone do wejścia PFI 0 (*Programmable Function Interface*) modułu NI USB-6001, które jest jednocześnie wejściem licznika zliczającego impulsy. Generator $G_{\mathrm{Ref}}$ i moduł pomiarowy NI USB-6001 są na stałe zamontowane na panelu ćwiczeniowym. Generator $G_{\mathrm{Ref}}$ jest zasilany napięciem +5 V z zasilacza sieciowego Z, zamontowanego również na panelu ćwiczeniowym. Łącznik kołyskowy W umożliwia załączanie i wyłączanie napięcia 230 V zasilającego panel ćwiczeniowy. Natomiast moduł pomiarowy zasilany jest bezpośrednio z portu USB komputera. Dzięki temu układ pomiarowy wykorzystywany w ćwiczeniu 3 nie wymaga wykonywania żadnych połączeń poza dołączeniem pa-

nelu ćwiczeniowego do sieci zasilającej 230 V oraz dołączeniem modułu NI USB-6001 do portu USB komputera.




**Ryc. 3.1.** Schemat blokowy stanowiska laboratoryjnego w ćwiczeniu 3

### 3.2.2. Budowa i parametry licznika wbudowanego w module NI USB-6001

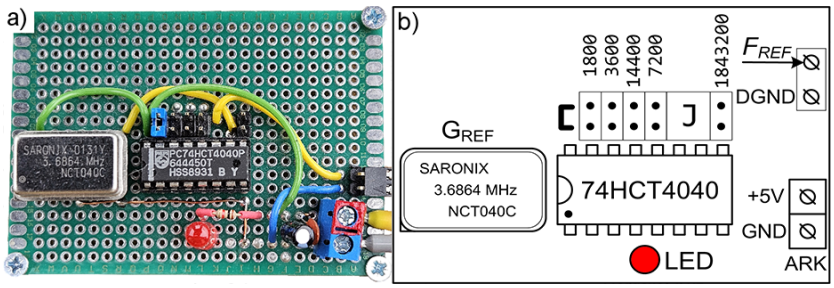
W porównaniu z innymi, bardziej zaawansowanymi modułami **DAQ**, licznik zastosowany w module NI USB-6001 ma uproszczoną konstrukcję [5]. Licznik nie ma układu bramkującego impulsy, co utrudnia realizację pomiarów częstotliwości poprzez zliczanie impulsów w zadanym czasie. W tym liczniku impulsy są zliczane po podaniu sygnału cyfrowego na odpowiednie wejście licznika. Licznik może zliczać impulsy sygnałów cyfrowych zgodnych ze standardem TTL tylko w górę, z maksymalną częstotliwością 5 MHz. Możemy natomiast wybrać jedno z dwóch wejść licznika (PFI 0 lub PFI 1) oraz ustalić zbocze, na które będzie reagował licznik: narastające (**rising edge**) lub opadające (**falling edge**). Domyślnie licznik jest skonfigurowany do zliczania impulsów na wejściu PFI 0 i w taki sposób wykorzystujemy go w ćwiczeniu 3. W innych ćwiczeniach wykorzystamy wejście PFI 1, co będzie wymagało dodatkowego skonfigurowania licznika. Licznik jest 32-bitowy i zlicza w kodzie binarnym, co oznacza, że może maksymalnie zliczyć $2^{32} = 4\,294\,967\,296$ impulsów. Oznacza to, że przy maksymalnej częstotliwości impulsów 5 MHz licznik może zliczać te impulsy przez prawie 15 minut bez przepełnienia. Dla mniejszych częstotliwości licznik może oczywiście zliczać impulsy odpowiednio dłużej. Po przepełnieniu licznika zliczanie jest kontynuowane ponownie od zera. W tabeli 3.1 przedstawiono najważniejsze parametry licznika impulsów wbudowanego w moduł NI USB-6001.

**Tab. 3.1.** Główne parametry licznika impulsów modułu NI USB-6001 [6]

| Parametry | Wartości |
|---|---|
| Rozdzielczość | 32 bity |
| Standard impulsów | TTL |
| Możliwe pomiary | zliczanie zboczy narastających lub opadających |
| Kierunek zliczania | tylko w górę |
| Źródło zliczanych impulsów | PFI 0 lub PFI 1, domyślne wejście PFI 0 |
| Maksymalna częstotliwość impulsów | 5 MHz |
| Minimalny czas trwania stanu wysokiego | 100 ns |
| Minimalny czas trwania stanu niskiego | 100 ns |

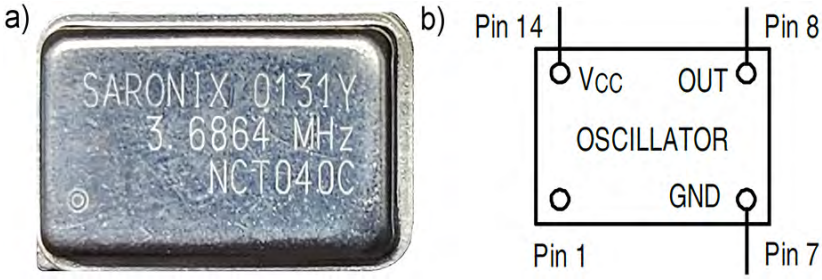
### 3.2.3. Budowa i parametry kwarcowego generatora częstotliwości wzorcowych

W ćwiczeniu 3 realizujemy pomiary częstotliwości przebiegu prostokątnego z wykorzystaniem licznika zliczającego impulsy. Jest to więc całkowicie inna metoda pomiaru częstotliwości niż ta stosowana w ćwiczeniu 2. W module NI USB-6001 zastosowanym na stanowisku laboratoryjnym wbudowany jest 32-bitowy licznik, który może zliczać impulsy spełniające standard TTL, tzn. przyjmujące wartości napięcia z zakresu od 0 do 5 V. Na panelu ćwiczeniowym zainstalowano generator impulsów odpowiedni do eksperymentów z tym licznikiem. Ma on postać niewielkiej płytki PCB przedstawionej na rycinie 3.2. Generator zasilany jest z zasilacza +5 V/1 A zainstalowanego na panelu ćwiczeniowym.




**Ryc. 3.2.** Płytka PCB generatora impulsów w standardzie TTL o wzorcowej częstotliwości $F_{REF}$: a) rzeczywisty wygląd płytki PCB, b) rozmieszczenie elementów: GREF – generator kwarcowy 3,686400 MHz, 74HCT4040 – dzielnik, J – jumper do wyboru częstotliwości, LED – czerwona dioda świecąca sygnalizująca napięcie zasilania, *FREF* – wyjście dzielnika częstotliwości, DGND – cyfrowa masa sygnałowa, +5 V – napięcie zasilające, GND – masa zasilania

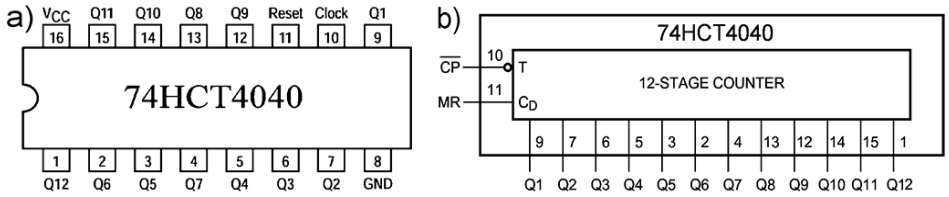
Źródłem częstotliwości jest wzorcowy generator kwarcowy firmy SaRonix o częstotliwości znamionowej $225 \cdot 2^{14}$ Hz = 3 686 400 Hz, zamknięty w niewielkiej metalowej obudowie o wymiarach zgodnych ze standardem obudowy DIP14 [25]. Wygląd generatora i opis wyprowadzeń przedstawiono na rycinie 3.3. Jak widać na rycinie 3.3, generator jest wykonany w odmianie C, tzn. jego tolerancja stabilności wynosi ±100 ppm, czyli 0,01 %, uwzględniając wszystkie warunki: tolerancję kalibracji, temperaturę roboczą, zmiany napięcia wejściowego, zmiany obciążenia, starzenie, wstrząsy i wibracje. Generator zasilany jest napięciem +5 V względem masy GND. Obecność napięcia zasilającego generator sygnalizuje czerwona dioda LED na płytce PCB.




**Ryc. 3.3.** Obudowa generatora SaRonix oraz rozmieszczenie wyprowadzeń (widok od góry obudowy) [25]

Sygnał wyjściowy z generatora o częstotliwości 3 686 400 Hz jest podany na wejście zegarowe CP (**Clock Pulse**) 12-bitowego licznika binarnego typu 74HCT4040 [26]. Na rycinie 3.4 pokazano obudowę układu 74HCT4040 oraz jego schemat blokowy. Licznik ten pełni rolę 12-stopniowego dzielnika częstotliwości, gdzie każdy stopień dzieli częstotliwość przez 2. Powyżej układu 74HCT4040 (ryc. 3.2) umieszczony jest zestaw pięciu styków konfiguracyjnych, tzw. jumperów **J**, umożliwiających wybór częstotliwości podawanej dalej na wejście PFI0 licznika w module NI USB-6001. Wszystkie częstotliwości dostępne na wyjściach Q1 ... Q12 dzielnika częstotliwości zestawiono w tabeli 3.2. Zauważamy, że są to częstotliwości: 900 Hz, 1800 HZ, 3600 Hz, 7200 Hz, 14 400 Hz itd. Wartości te są często stosowane do taktowania szybkości transmisji w różnego rodzaju interfejsach szeregowych. Na przykład częstotliwość 115 200 Hz jest jedną ze standardowych częstotliwości taktujących w systemie transmisji szeregowej RS-232. Dlatego zastosowany na stanowisku generator ma taką na pozór „dziwną" wartość częstotliwości, która dzieli się

bez reszty przez 2 aż do dzielnika równego $2^{14} = 16\,384$, dla którego otrzymujemy częstotliwość 225 Hz. Zauważmy, że liczba 225 jest iloczynem kwadratów dwóch najmniejszych nieparzystych liczb pierwszych: $3^2 \cdot 5^2 = 225$. Pozwala to uzyskać inne standardowe szybkości transmisji szeregowej RS-232.




**Ryc. 3.4.** Rozmieszczenie wyprowadzeń na obudowie licznika binarnego – dzielnika częstotliwości 74HCT4040 oraz jego schemat funkcjonalny

Na stanowisku wykorzystujemy tylko pięć wartości częstotliwości dostępnych na wyjściach Q1, Q8 ... Q11, zaznaczonych w tabeli 3.2 pogrubioną czcionką. Wybór pożądanej wartości częstotliwości uzyskujemy, zakładając zworkę (**jumper**) na odpowiedniej pozycji zgodnie z oznaczeniami pokazanymi na rycinie 3.2.

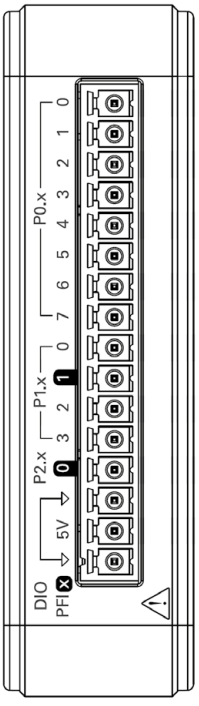
**Tab. 3.2.** Częstotliwości wyjściowe z dzielnika 74HCT4040, pogrubioną czcionką zaznaczono wyjścia i częstotliwości wykorzystane na stanowisku laboratoryjnym

| Numer pinu | Sygnał | Dzielnik | $F_{out}$ [Hz] |
|---|---|---|---|
| 10 | Input | 1 | 3 686 400 |
| 9 | Q1 | 2 | 1 843 200 |
| 7 | Q2 | 4 | 921 600 |
| 6 | Q3 | 8 | 460 800 |
| 5 | Q4 | 16 | 230 400 |
| 3 | Q5 | 32 | 115 200 |
| 2 | Q6 | 64 | 57 600 |
| 4 | Q7 | 128 | 28 800 |
| 13 | Q8 | 256 | 14 400 |
| 12 | Q9 | 512 | 7 200 |
| 14 | Q10 | 1024 | 3 600 |
| 15 | Q11 | 2046 | 1 800 |
| 1 | Q12 | 4096 | 900 |

## 3.3. Dołączanie sygnału do wejścia licznika impulsów modułu NI USB-6001

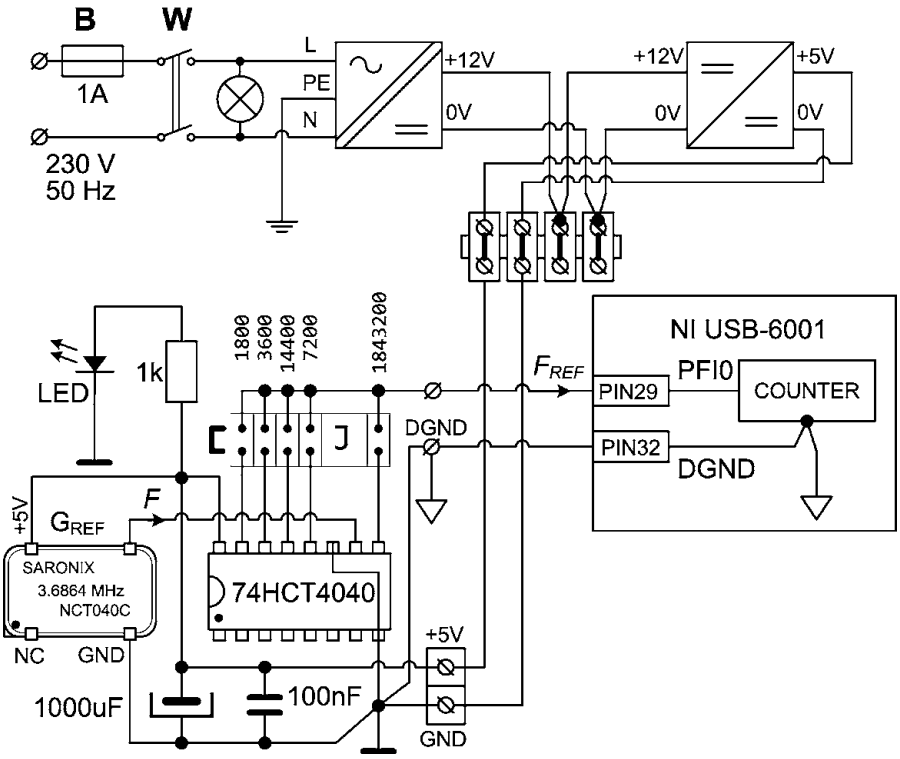
Licznik w module NI USB-6001 ma dwa wejścia PFI 0 oraz PFI 1, przy czym w danej chwili możemy korzystać tylko z jednego. Oba wejścia licznika znajdują się na złączu sygnałów cyfrowych modułu NI USB-6001. Rozkład sygnałów cyfrowych na tym złączu pokazano w tabeli 3.3. Łącznie dostępnych jest 13 linii cyfrowych, które możemy dowolnie i niezależnie konfigurować jako wejścia lub wyjścia. Linie cyfrowe są zgrupowane w trzech portach: P0, P1 i P2. W porcie P0 znajduje się 8 linii oznaczonych od P0.0 do P0.7, w porcie P1 znajdują się 4 linie oznaczone od P1.0 do P1.3, a w porcie P2 dostępna jest tylko jedna linia P2.0, która jednocześnie pełni rolę wejścia PFI 0 dla 32-bitowego licznika impulsów. Podobnie podwójną rolę pełni linia P1.1, która jednocześnie jest wejściem PFI 1 dla licznika impulsów. Wszystkie linie cyfrowe mają wspólną linię masy cyfrowej D GND.

**Tab. 3.3.** Sygnały cyfrowe na złączu modułu NI USB-6001

| Złącze cyfrowe | PIN | Sygnał |
|---|---|---|
|  | 17 | P0.0 |
| | 18 | P0.1 |
| | 19 | P0.2 |
| | 20 | P0.3 |
| | 21 | P0.4 |
| | 22 | P0.5 |
| | 23 | P0.6 |
| | 24 | P0.7 |
| | 25 | P1.0 |
| | 26 | P1.1/PFI 1 |
| | 27 | P1.2 |
| | 28 | P1.3 |
| | **29** | **P2.0/PFI 0** |
| | 30 | D GND |
| | 31 | +5 V/150 mA |
| | **32** | **D GND** |

W ćwiczeniu 3 wykorzystujemy wejście licznika dołączone do linii P2.0/PFI 0. Linia ta może pracować jako standardowa dwukierunkowa linia cyfrowa portu P2.0 lub jako programowalna linia interfejsu PFI 0 (**Programmable Function Interface**), która działa wtedy tylko jako wejście cyfrowe dla licznika lub jako wejście **Start Trigger** wyzwalające pomiary ADC lub przetwarzanie DAC.

Najważniejsze parametry licznika przedstawiono w tabeli 3.1. Podobnie jak wszystkie inne linie cyfrowe, wejście licznika ma również wewnętrzny rezystor **pull-down** 47,5 kΩ dołączony do masy D GND, który wymusza stan niski na wejściu i umożliwia bezpośrednią współpracę z pasywnymi stykami elektromechanicznymi dołączonymi do napięcia +5 V.




**Ryc. 3.5.** Schemat układu połączeń w ćwiczeniu 3

Na stanowisku laboratoryjnym wykorzystujemy linię P2.0/PFI 0 wyłącznie jako wejście PFI 0 licznika. Licznik jest fabrycznie domyślnie skonfigurowany do zliczania impulsów na wejściu PFI 0, dzięki czemu nie musimy go dodatkowo konfigurować w programie. Rycina 3.5 przedstawia schemat

układu wykorzystanego w ćwiczeniu 3. Układ jest w całości zmontowany na panelu ćwiczeniowym i nie wymaga od nas wykonywania żadnych dodatkowych połączeń. Wyjście generatora referencyjnego $G_{Ref}$ jest dołączone do wejścia dzielnika częstotliwości 74HCT4040. Z pięciu wyjść dzielnika opisanych w tabeli 3.2 możemy wybrać jedną wartość częstotliwości wzorcowej $F_{Ref}$, zakładając na odpowiedniej pozycji zworkę konfiguracyjną J. Wyjście dzielnika częstotliwości dołączone jest do wejścia PFI0 (PIN29) modułu pomiarowego NI USB-6001. Generator i dzielnik częstotliwości są zasilane napięciem +5 V z zasilacza napięcia stałego. Obecność napięcia +5 V na płytce PCB sygnalizuje czerwona dioda LED.

## 3.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 3.4.1. Uruchamianie stanowiska laboratoryjnego

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Zasilanie komputera i panelu ćwiczeniowego powinno być wyłączone. Moduł pomiarowy powinien być odłączony od portu USB komputera.**

Wszelkie zauważone odstępstwa od tego stanu zgłaszamy prowadzącemu zajęcia.

Sprawdzamy układ zgodnie z opisem w poprzednim punkcie. Jeśli układ jest połączony prawidłowo, zgłaszamy prowadzącemu zajęcia gotowość do rozpoczęcia realizacji ćwiczenia.

**Dopiero po otrzymaniu od prowadzącego zajęcia wyraźnej zgody włączamy wyłącznie zasilanie komputera. Nie włączamy jeszcze zasilania panelu ćwiczeniowego. Nie dołączamy modułu pomiarowego do portu USB komputera**.

Po uruchomieniu systemu operacyjnego nie wykonujemy żadnych czynności i przechodzimy do realizacji kolejnego punktu.

### 3.4.2. *Self-test* modułu NI USB-6001

Jako pierwszą czynność po uruchomieniu komputera sprawdzamy, czy w systemie są zainstalowane dwa urządzenia NI USB-6001: jedno fizyczne jako **Dev1** i drugie symulowane jako **Dev2**. W tym celu uruchamiamy program NI MAX, rozwijamy listę **Devices and Interfaces** i sprawdzamy zainstalowane w systemie urządzenia (ryc. 1.12). Notujemy w protokole nazwy, typy, numery seryjne i statusy zainstalowanych w systemie urządzeń z serii NI USB. Moduł używany na panelu ćwiczeniowym powinien być jeszcze odłączony, a więc na liście powinien być oznaczony czerwonym znakiem „x" i mieć status **Not Present**. Jeśli na liście nie ma wymaganych urządzeń, ponownie instalujemy je zgodnie z opisem w instrukcji do ćwiczenia 1.

Pamiętamy, że moduł używany i zainstalowany już w systemie na wcześniejszych zajęciach mógł zostać z nieznanych powodów (najczęściej przypadkowo) usunięty przez innych studentów. Dlatego zawsze na początku zajęć upewniamy się, jaki numer urządzenia został przypisany w systemie do aktualnie używanego modułu pomiarowego. Upewniamy się również, czy na każdych zajęciach pracujemy na tym samym stanowisku laboratoryjnym, gdyż każdy wyprodukowany moduł DAQ ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczonych w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**, jeśli więc wykorzystywany moduł został zainstalowany pod inną nazwą, musimy odpowiednio zmodyfikować tworzone programy.

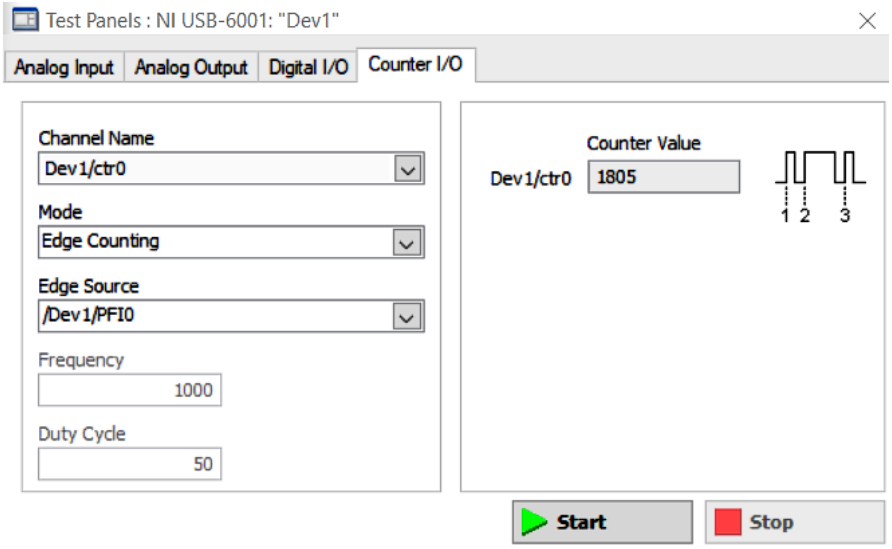
**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera. Nie załączamy jeszcze zasilania panelu ćwiczeniowego.** Sprawdzamy zachowanie niebieskiej kontrolki LED na obudowie modułu. Notujemy wynik obserwacji w protokole.

Jeśli używane podczas zajęć urządzenie NI USB-6001 zostało poprawnie zainstalowane, w programie NI MAX przeprowadzamy *self-test* modułu fizycznego i symulowanego zgodnie z opisem zamieszczonym w instrukcji do pierwszego ćwiczenia (ryc. 1.13). Notujemy rezultaty *self-testu* w protokole.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody włączamy zasilanie panelu ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku powinno potwierdzić obecność napięcia zasilającego.

### 3.4.3.Testowanie zliczania impulsów z generatora wzorcowego

Test zliczania impulsów wykonujemy przy dołączonym do wejścia PFI 0 generatorze częstotliwości wzorcowej ustawionej na 1800 Hz, zgodnie z ryciną 3.2. Po sprawdzeniu układu połączeń załączamy zasilanie stanowiska laboratoryjnego. W otwartym oknie **Measurement & Automation Explorer** wybieramy zakładkę **Test Panels** (ryc. 1.14). Otworzy się okno **Test Panels**, w którym wybieramy zakładkę **Counter I/O**. Konfigurujemy test licznika zgodnie z ryciną 3.6, zwracając uwagę na wybór wejścia PFI 0 w okienku **Edge Source**. Uruchamiamy test, naciskając przycisk **START**. W oknie **Counter Value** sprawdzamy, czy liczba impulsów się zwiększa. Zatrzymujemy test przyciskiem **STOP**. Ponownie uruchamiamy test i po jednej sekundzie go zatrzymujemy. Sprawdzamy liczbę zliczonych impulsów – powinna być zbliżona do 1800. Notujemy w protokole zaobserwowane rezultaty testu.




**Ryc. 3.6.** Testowanie licznika impulsów na wejściu PFI 0 w programie NI MAX

Powtarzamy test licznika dla urządzenia symulowanego i sprawdzamy, ile impulsów zliczył licznik w symulowanym module. Notujemy wniosek w protokole. Każdorazowo po testowaniu licznika wykonujemy zrzuty ekranu do plików i zamieszczamy je w raporcie, wzorując się na przykładzie ryciny 3.6.

Pamiętamy, że istnieją pewne ograniczenia w korzystaniu z symulowanych urządzeń NI-DAQmx w porównaniu z rzeczywistymi urządzeniami

fizycznymi. Podczas realizacji ćwiczenia 3 istotne ograniczenie polega na tym, że dane odczytywane z licznika są zawsze zwracane jako wartość równa zero. Stanowi to poważny problem, gdyż wynik pomiaru częstotliwości z użyciem urządzenia symulowanego również zawsze będzie równy zero. Dlatego musimy dodatkowo symulować zliczanie impulsów odpowiednim fragmentem programu.

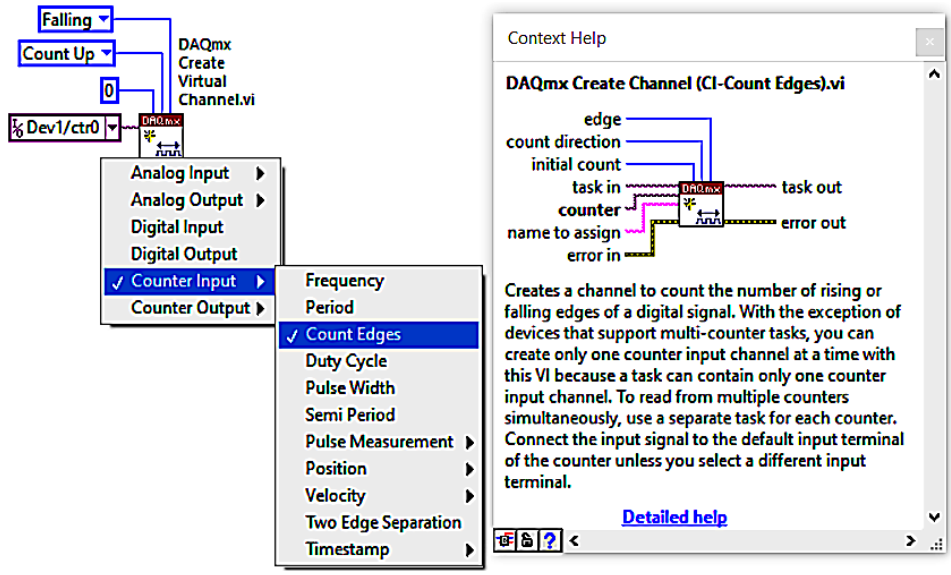
## 3.5. Programowanie

### 3.5.1. Pierwszy prosty program do odczytu licznika impulsów modułu NI USB-6001

Aby odczytać zawartość licznika impulsów modułu NI USB-6001, przygotowujemy odpowiedni program. Przykładowy diagram takiego programu przedstawia rycina 3.9. Będziemy go tworzyć krok po kroku. Najpierw tworzymy kanał wirtualny (**Create Virtual Channel**), konfigurujemy go, przypisując mu odpowiedni kanał fizyczny modułu pomiarowego powiązany z określonymi zaciskami na złączu, i definiujemy parametry: typ zbocza, kierunek zliczania, wartość początkową itp. Następnie uruchamiamy zadanie (**Start Task**), które będzie reprezentowało utworzony kanał wirtualny na diagramie połączeń. Gdy zadanie jest uruchomione, możemy odczytać zawartość licznika (**Read**). Kończąc działanie przyrządu wirtualnego, kończymy zadanie (**Clear Task**) i obsługujemy ewentualne błędy, które mogą pojawić się podczas działania programu.

Tworzenie kanału wirtualnego i konfigurowanie licznika pokazano na rycinie 3.7. Kolejność działań jest następująca. Uruchamiamy środowisko **LabVIEW**, tworzymy nowy, pusty przyrząd wirtualny (**Create Project > Blank VI**) i przełączamy się na diagram (**Ctrl+E**). Otwieramy okienko pomocy **Context Help**, klikając znak zapytania  w prawym górnym rogu okna programu. Prawym kliknięciem w oknie diagramu otwieramy paletę funkcji (**Functions**) i wybieramy zasobnik z funkcjami pomiarowymi dla modułów **DAQmx: Measurement I/O > NI DAQmx**. W tym zasobniku znajdują się wszystkie niezbędne funkcje do obsługi modułów pomiarowych **NI-DAQmx**. Najpierw wybieramy funkcję tworzącą wirtualny kanał **Create Channel**,

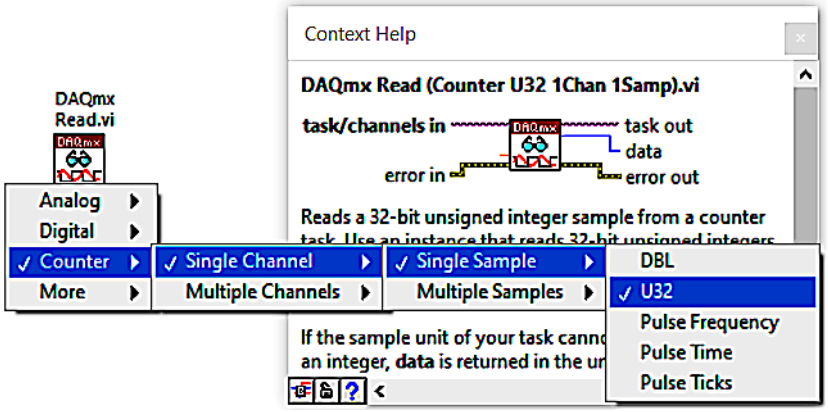
umieszczamy ją na pustym diagramie i, rozwijając selektor poniżej ikony funkcji, wybieramy konfigurację licznika: **Counter Input > Count Edges**. Korzystając z okna pomocy Context Help, prawym przyciskiem myszy tworzymy stałe (**Create > Constant**) na wejściach: **edge (Falling), count direction (Count Up), initial count (0)**. Podczas tworzenia stałej na wejściu **counter** wybieramy opcję przeglądania i z rozwiniętej listy dostępnych urządzeń wybieramy aktualnie używany moduł **Dev1/ctr0**. Dla zwiększenia czytelności tworzonego diagramu dodajemy etykietę do każdej funkcji: prawym kliknięciem na ikonie funkcji otwieramy menu konfiguracyjne i wybieramy opcję **Visible Items > Label**. Porządkujemy wszystkie umieszczone na diagramie obiekty tak, aby zajmowały jak najmniej miejsca, nie zakrywały się wzajemnie oraz żeby wszystkie połączenia były dobrze widoczne.




**Ryc. 3.7.** Konfiguracja licznika do zliczania impulsów - DAQmx Create Channel

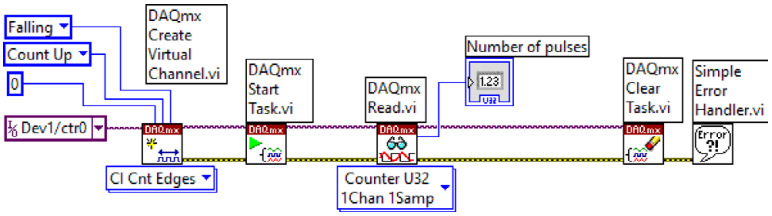
Następnie umieszczamy na diagramie następujące funkcje: **Start Task**, **Read**, **Clear Task** oraz **Simple Error Handler** z menu **Functions > Programming > Dialog & User Interface**. Włączamy widoczność etykiety dla każdej funkcji (**Visible Items > Label**). Funkcje **Start Task**, **Clear Task** oraz **Simple Error Handler** nie wymagają konfiguracji. Konfigurujemy jeszcze tylko funkcję **Read** w sposób pokazany na rycinie 3.8. Rozwijamy selektor poniżej ikony funkcji i wybieramy konfigurację **Counter > Single Channel >**

**Single Sample > U32**, dzięki czemu każde wywołanie tej funkcji odczyta jednorazowo 32-bitową zawartość licznika.




**Ryc. 3.8.** Konfiguracja funkcji Read do odczytu licznika – DAQmx Read

Na wyjściu **Data** funkcji **Read** prawym kliknięciem myszy tworzymy wskaźnik (**Create Indicator**) i nadajemy mu nazwę **Number of Pulses**. Sprawdzamy, czy na panelu pojawił się odpowiedni wskaźnik. Następnie łączymy wszystkie wyjścia **task out** z wejściami **task in** kolejnych modułów. Analogicznie łączymy wyjścia **error out** z wejściami **error in**. Powinniśmy otrzymać diagram przedstawiony na rycinie 3.9.




**Ryc. 3.9.** Pierwsza wersja programu odczytującego zawartość licznika

Jeśli na diagramie nie ma żadnego błędu, kilkakrotnie uruchamiamy otrzymany program przyciskiem . Program jednorazowo odczytuje zawartość licznika. Ponieważ jednak funkcja **Create Virtual Channel** każdorazowo po uruchomieniu programu wpisuje do licznika wartość początkową równą zero, program odczytuje z licznika zawsze podobną, niewielką liczbę impulsów, zależnie od tego, ile impulsów z generatora pojawi się na wejściu licznika pomiędzy wyzerowaniem licznika a jego odczytem.

Taki program nie ma więc jeszcze żadnego praktycznego zastosowania. Ponieważ jednak program ten będziemy dalej modyfikować aż do osiągnięcia końcowej wersji, zapisujemy go już teraz na dysku. Program zapisujemy w katalogu wskazanym przez prowadzącego zajęcia. Nazwy plików z programami tworzymy według schematu:

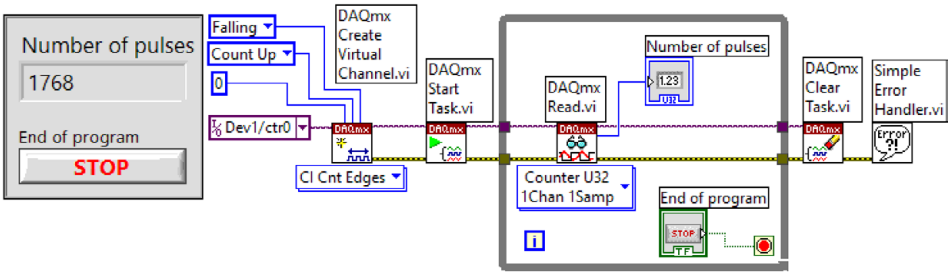
YYYYMMDD_Nazwisko_KSP_X-N-L.vi

gdzie: YYYY – cyfry roku, MM – cyfry miesiąca, DD – cyfry dnia, X – numer ćwiczenia, N – numer programu w realizowanym ćwiczeniu, L – litera (a, b, c itd.) dla kolejnych wersji tego samego programu realizowanego w danym punkcie ćwiczenia. W nazwie pliku nie wolno używać polskich liter, spacji ani innych znaków specjalnych. Notujemy nazwę pliku i katalogu w protokole. W nazwie tego pliku powinien występować numer programu 3-1-a.

Zanim jednak przystąpimy do dalszej modyfikacji programu, zawsze najpierw wykonujemy jego kopię, zapisując go pod nową nazwą i odpowiednio modyfikując końcowe cyfry lub literę w nazwie pliku (czyli w tym przypadku: 3-1-b). Jest to bardzo ważne, gdyż modyfikując dobrze działający program, często popełniamy poważne błędy, które trudno naprawić. Wtedy często po prostu lepiej jest ponownie otworzyć wcześniejszą wersję pliku.

Zmodyfikujemy teraz algorytm w taki sposób, aby po początkowym wyzerowaniu licznika program działał w pętli i wielokrotnie odczytywał zawartość licznika. W tym celu z zasobnika **Functions > Programming > Structures** pobieramy pętlę **While Loop** i umieszczamy ją na diagramie w sposób pokazany na rycinie 3.10. Na terminalu warunku zakończenia pętli dodajemy prawym kliknięciem (**Create Control**) przycisk STOP. Jeśli w miejscach przecięcia linii **Data** i **Error** z krawędzią pętli **While** zostały domyślnie wstawione rejestry przesuwne (**Shift Register**), zamieniamy je na tunele (**Loop Tunnel**) w trybie przekazywania ostatniej wartości (**Last Value**). W tym celu klikamy prawym przyciskiem myszy na symbolu wyjściowego rejestru przesuwnego i wybieramy opcję **Replace with Tunnels**. Następnie ponownie klikamy na otrzymanym symbolu tunelu wyjściowego z pętli i wybieramy opcję **Tunnel Mode > Last Value**. Powtarzamy te czynności dla pozostałych rejestrów wyjściowych.

Porządkujemy elementy na diagramie i panelu. Powinniśmy ostatecznie otrzymać program wyglądający podobnie jak na rycinie 3.10.




**Ryc. 3.10.** Panel i diagram programu odczytującego licznik w pętli

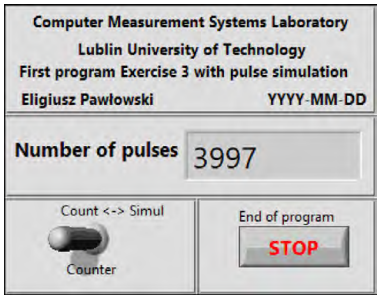
Jeśli środowisko **LabVIEW** nie zgłasza żadnych błędów, uruchamiamy program przyciskiem , co sygnalizuje zmiana wyglądu przycisku na . Ponieważ program ma w swojej strukturze pętlę **While**, będzie działał nieprzerwanie do momentu naciśnięcia przycisku STOP na panelu. **Nie używamy** przycisku Stop na pasku narzędziowym środowiska **LabVIEW**! Kolejne odczyty licznika powinny zwiększać się z szybkością proporcjonalną do częstotliwości odniesienia $F_{Ref}$ podanej na wejście PFI 0 licznika. Po każdym zatrzymaniu i ponownym uruchomieniu programu zliczanie impulsów powinno rozpoczynać się od zera.

Ustawiamy zworkę konfiguracyjną generatora częstotliwości wzorcowej na 1800 Hz i staramy się uruchomić program dokładnie na 1 sekundę. Program powinien zliczyć około 1800 impulsów. Po sprawdzeniu działania programu zapisujemy go na dysku oraz od razu wykonujemy kopię pod nową nazwą (z końcówką 3-1-c), gdyż program ten będziemy dalej modyfikować.

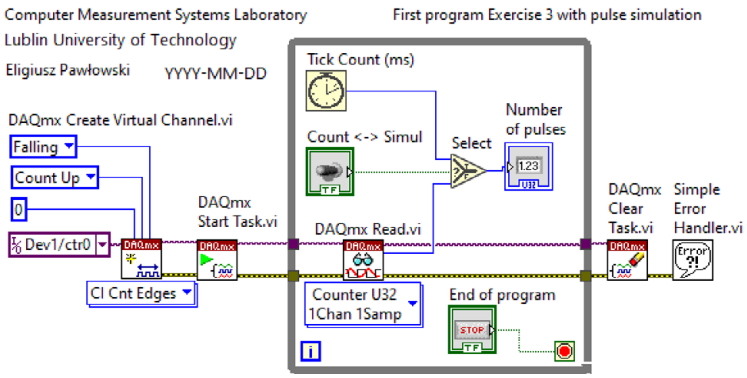
Zmieniamy wartość zadajnika na wejściu **counter** funkcji **Create Channel** na urządzenie symulowane **Dev2/ctr0**. Uruchamiamy program i sprawdzamy jego działanie. Niestety, jeśli używamy symulowanego urządzenia, wartość odczytana z licznika zawsze wynosi zero. Dzieje się tak, ponieważ symulowane urządzenie nie może realizować wszystkich funkcji urządzenia fizycznego. Dlatego w tym przypadku konieczne jest zasymulowanie liczby impulsów odczytywanych z licznika. W tym celu można wykorzystać funkcję **Tick Count (ms)**, która odczytuje z zegara systemu operacyjnego komputera liczbę milisekund, jakie upłynęły od chwili włączenia komputera. Aby

przełączać program między rzeczywistym odczytem licznika a odczytem symulowanym licznika milisekund, użyjemy funkcji **Select**.

Najpierw rozciągamy pętlę **While** w górę, aby zrobić miejsce na dodatkowe elementy diagramu. Z zasobnika **Functions > Programming > Timing** pobieramy funkcję **Tick Count (ms)** i wstawiamy ją do pętli. Z zasobnika **Function > Programming > Comparision** pobieramy funkcję **Select**, wstawiamy ją do pętli i na wejściu **s** selektora tworzymy prawym kliknięciem kontrolkę (**Create Control**). Przełączamy się na panel i prawym kliknięciem na przełączniku selektora wybierz **Replace > Boolean > Horizontal Toggle Switch**. Ponownie prawym kliknięciem na przełączniku wybieramy **Properties > Appearance** i konfigurujemy przycisk (**Label**, **Colors**, **Boolean text**) tak, aby jego działanie było jednoznaczne dla użytkownika. Przykładowy wygląd panelu po niezbędnych zmianach pokazano na rycinie 3.11. Połącz diagram uzupełniony o nowe funkcje zgodnie z ryciną 3.12.




**Ryc. 3.11.** Panel końcowej wersji pierwszego programu odczytującego licznik uzupełniony o symulację zliczania impulsów




**Ryc. 3.12.** Diagram końcowej wersji pierwszego programu odczytującego licznik uzupełniony o symulację zliczania impulsów

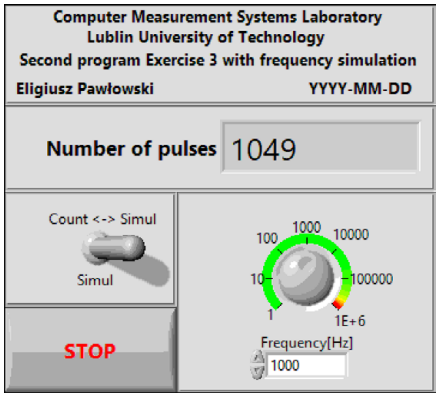
Korzystając z elementów graficznych dostępnych w zasobniku **Controls > Decorations**, porządkujemy i grupujemy obiekty na panelu. Uzupełnij panel i diagram następującymi informacjami: nazwa przedmiotu, uczelnia, dane osobowe zespołu laboratoryjnego, numer grupy dziekańskiej, data ćwiczenia, numer ćwiczenia, numer programu i jego tytuł. Informacje te podajemy w formie komentarza umieszczonego na panelu i na diagramie. Aktualizujemy opisy wszystkich obiektów zgodnie z ich rolą. Używamy narzędzi **Align Objects**  , **Distribute Objects**  , **Reorder**  dostępnych na pasku narzędziowym, aby wyrównać położenie obiektów na panelu, zgrupować je (**Group**) i zablokować ich położenie (**Lock**).

Uruchamiamy program przyciskiem  . Obserwujemy zachowanie wskaźnika liczby impulsów dla obu pozycji przełącznika wyboru trybu pracy oraz porównujemy działanie programu dla urządzenia fizycznego i symulowanego, sprawdzając, jaką różnicę można zauważyć. Notujemy wnioski w protokole. Sprawdzamy działanie przycisku STOP, a jeśli program nie działa prawidłowo, znajdujemy i poprawiamy błędy. Zapisujemy program do pliku i przygotowujemy do użycia stoper, np. w telefonie komórkowym. Uruchamiamy program w trybie zliczania impulsów licznikiem i staramy się go wyłączyć dokładnie po upływie 10 sekund, po czym sprawdzamy, ile impulsów zliczył program, i analizujemy, dlaczego uzyskano taki wynik. Powtarzamy eksperyment w trybie symulacji i sprawdzamy, ile impulsów zliczył program tym razem. Notujemy wyniki obserwacji i wnioski w protokole.

Wybierając opcje **File > Print > VI Documentation**, zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Odnajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Notujemy nazwę utworzonego katalogu i nazwy plików wraz z opisem zawartości w protokole. Zapisane pliki wykorzystujemy do sporządzenia raportu. Wykonujemy kopię programu do realizacji następnego zadania, zmieniając jego numer na kolejny (3-2-a). Nie modyfikujemy programu będącego ostateczną wersją poprzednio wykonanego zadania. Zawsze pamiętamy, aby po zapisaniu ostatecznej wersji programu w pełni realizującego dany punkt ćwiczenia wykonać jej kopię, nadając jej nową nazwę poprzez zmianę ostatniej cyfry lub litery na kolejną.

### 3.5.2. Drugi program z symulacją zliczania impulsów o zadanej częstotliwości

Pierwszy program wykonany w poprzednim punkcie może symulować tylko impulsy o jednej częstotliwości równej 1000 Hz, ponieważ funkcja **Tick Count (ms)** odczytuje z systemu operacyjnego komputera liczbę milisekund. Jest to istotne ograniczenie tego programu. Dlatego następnym zadaniem będzie program symulujący impulsy o dowolnej częstotliwości, którą będziemy zadawać wirtualnym pokrętłem. Przykładowy panel programu realizującego taką symulację pokazano na rycinie 3.13.




**Ryc. 3.13.** Panel drugiego programu odczytującego licznik z symulacją częstotliwości zliczanych impulsów

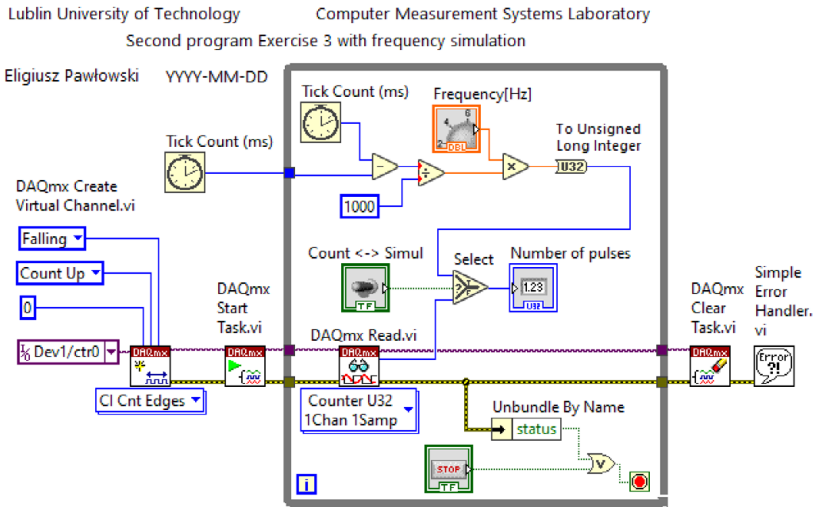
Otwieramy kopię programu zapisaną na koniec poprzedniego punktu, uruchamiamy go i upewniamy się, czy działa prawidłowo. Jeśli tak, uzupełniamy diagram tak, jak pokazano na rycinie 3.14. Zadanie to wykonujemy w kolejnych krokach opisanych poniżej.

W programie tym wykorzystujemy funkcję **Tick Count (ms)** dwa razy: raz przed pętlą **While** i drugi raz wewnątrz pętli **While**. Następnie obliczamy różnicę, która jest równa liczbie milisekund od momentu uruchomienia programu. Następnie przeliczamy milisekundy na sekundy i mnożymy przez zadaną częstotliwość. Uzyskujemy w ten sposób liczbę symulowanych impulsów, którą zliczyłby rzeczywisty licznik od chwili uruchomienia programu. Możemy w ten sposób zasymulować zliczanie impulsów o dowolnej częstotliwości.

Najpierw umieszczamy na panelu kontrolkę **Controls > Numeric > Knob** i nadajemy jej nazwę **Frequency**. Konfigurujemy tę kontrolkę prawym

kliknięciem, wybierając opcję **Properties**. Dodajemy wyświetlacz cyfrowy oraz skalę logarytmiczną i ustawiamy zakres od 1 do 1 000 000. Odpowiednio skonfigurowana kontrolka powinna wyglądać tak, jak na rycinie 3.13.

Następnie na diagramie obliczamy liczbę milisekund od momentu uruchomienia programu i dzielimy przez 1000, co przelicza milisekundy na sekundy. Mnożymy liczbę sekund przez częstotliwość ustawioną za pomocą pokrętła **Frequency**. W ten sposób program zasymuluje liczbę impulsów o dowolnej częstotliwości od chwili uruchomienia programu. Ponieważ jednak, przeliczając milisekundy na sekundy, możemy otrzymać wartość ułamkową, również obliczona liczba impulsów może okazać się ułamkiem, a powinna być zawsze liczbą całkowitą. Nie możemy przecież zliczyć w liczniku tylko części impulsu – zawsze zliczane są całe impulsy. Dlatego obliczoną liczbę impulsów konwertujemy na zmienną typu całkowitego, 32-bitową, bez znaku (U32). Taki sam typ ma wartość odczytywana z rzeczywistego licznika w module NI USB-6001. Sygnalizuje to połączenie w niebieskim kolorze.




**Ryc. 3.14.** Diagram drugiego programu odczytującego licznik z symulacją częstotliwości zliczanych impulsów

W tym celu z zasobnika **Functions > Programming > Numeric > Conversion** pobieramy funkcję **To Unsigned Long Integer** i wstawiamy ją do diagramu. Uruchamiamy program i przełączamy go w tryb symulacji. Zmieniamy ustawienie symulowanej częstotliwości, obserwujemy liczbę impulsów i sprawdzamy, czy program działa prawidłowo. Ustawiamy symulowaną częstotliwość

na 1000 Hz i staramy się uruchomić program dokładnie na 10 sekund. Program powinien zliczyć około 10 000 impulsów. Jeśli program nie działa prawidłowo, znajdujemy i poprawiamy błędy.

Uzupełniamy program o kontrolę błędów. Używamy funkcji **Programming > Cluster, Class, & Variant > Unbundle by Name** i funkcji **Programming > Boolean > OR**, aby zatrzymać pętlę **While**, jeśli wystąpi błąd podczas odczytu licznika.

Aby zweryfikować działanie sprawdzania błędów, zmieniamy konfigurację funkcji **Create Virtual Channel** na nieprawidłową. Na przykład zmieniamy kierunek zliczania impulsów na **Count Down**. Licznik modułu NI USB-6001 zlicza tylko w górę i nie obsługuje funkcji zliczania w dół. Uruchamiamy program i obserwujemy jego działanie. Jeśli pojawi się komunikat o błędzie, notujemy go w protokole i opisujemy podczas opracowywania raportu (możemy również użyć w tym celu funkcji **Print Screen**).

Ustawiamy ponownie prawidłową konfigurację funkcji **Create Virtual Channel** na **Count Up**.

Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami, wzorując się na przykładzie pokazanym na rycinie 3.12.

Zapisujemy drugi program na dysku i notujemy nazwę pliku w protokole.

Drukujemy dokumentację programu do pliku, po czym odnajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwy plików wraz z opisem zawartości notujemy w protokole i wykorzystujemy je później do przygotowania raportu. Wykonujemy kopię programu na potrzeby następnego punktu ćwiczenia, dodając do nazwy pliku odpowiednią końcówkę (3-3-a).

### 3.5.3. Trzeci program do pomiaru częstotliwości w zadanym czasie

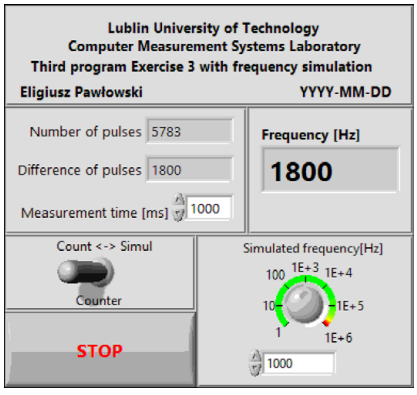
Zadaniem trzeciego programu jest pomiar częstotliwości impulsów za pomocą licznika modułu NI USB-6001. Znanych jest wiele różnych metod pomiaru częstotliwości z wykorzystaniem licznika i komputera [27, 28]. W tym ćwiczeniu zastosujemy stosunkowo prostą metodę. Polega ona na zliczaniu impulsów w zadanym czasie, a następnie podzieleniu liczby tych impulsów przez czas zliczania wyrażony w sekundach. W ten sposób otrzymujemy wynik pomiaru częstotliwości w hercach, zgodnie z zależnością:

$$F_X = \frac{N_X}{T_P}, \tag{3.1}$$

gdzie: $F_X$ jest mierzoną częstotliwością w hercach, $T_P$ jest czasem pomiaru, czyli czasem zliczania impulsów w sekundach, $N_X$ jest liczbą impulsów zliczonych w liczniku. Problemem w bezpośrednim zastosowaniu tej metody jest to, że w module NI USB-6001 nie ma układu bramkującego impulsy podawane na wejście licznika. Po uruchomieniu programu licznik zlicza impulsy cały czas, bez przerwy. Dlatego zastosujemy tzw. metodę odczytu licznika w locie (*on the fly*). W tej metodzie odczytujemy licznik wielokrotnie, przy czym odstęp czasu pomiędzy kolejnymi odczytami jest dokładnie znany. Następnie obliczamy różnicę dwóch kolejnych odczytanych z licznika liczb impulsów, którą dzielimy przez czas pomiędzy tymi odczytami, zgodnie z zależnością:

$$F_i = \frac{N_i - N_{i-1}}{t_i - t_{i-1}} = \frac{N_i - N_{i-1}}{T_P}, \tag{3.2}$$

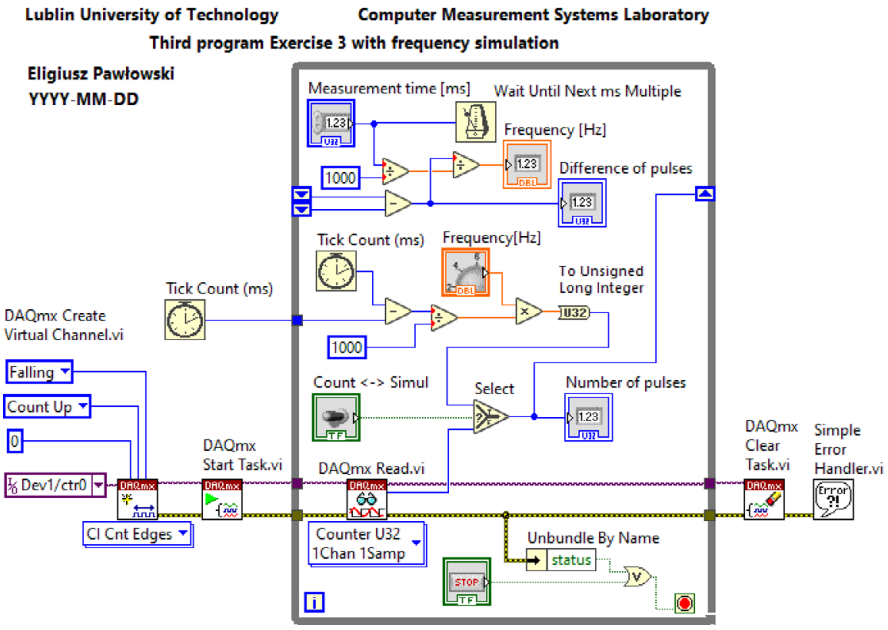
gdzie: $F_i$ jest $i$-tym wynikiem pomiaru częstotliwości, $N_i$ jest aktualną liczbą impulsów odczytaną z licznika w chwili $t_i$, $N_{i-1}$ jest poprzednią liczbą impulsów odczytaną z licznika w chwili $t_{i-1}$, oraz $T_P$ jest czasem pomiaru równym różnicy pomiędzy chwilami odczytu, czyli $T_P = t_i - t_{i-1}$. Program zrealizowany w poprzednim punkcie odczytywał licznik wielokrotnie tak często, jak tylko pozwalał na to system operacyjny komputera. Każda odczytana zawartość licznika była na bieżąco wyświetlana bez żadnego dodatkowego przetwarzania danych. W tym programie wprowadzamy do pętli **While** opóźnienie czasowe i z każdych dwóch kolejnych odczytów licznika obliczamy częstotliwość impulsów. Przykładowy panel odpowiedniego programu pokazano na rycinie 3.15.




**Ryc. 3.15.** Przykładowy panel programu do pomiaru częstotliwości

Widać na nim dodane nowe okienko pokazujące wynik pomiaru częstotliwości oraz nowy zadajnik, którym możemy ustawić czas trwania pomiaru. Dodatkowo, dla kontroli działania programu, pokazujemy aktualną liczbę impulsów odczytaną z licznika oraz różnicę impulsów z dwóch kolejnych odczytów licznika. Odpowiednie modyfikacje programu opracowanego w poprzednim punkcie wykonujemy etapami opisanymi poniżej.

Otwieramy kopię programu zapisaną w poprzednim punkcie i uzupełniamy diagram programu w sposób pokazany na rycinie 3.16.




**Ryc. 3.16.** Przykładowy diagram programu do pomiaru częstotliwości

Kolejne kroki wykonujemy w następujący sposób: na diagramie rozciągamy pętlę **While** do góry, robiąc miejsce na dodatkowe elementy, a następnie uzupełniamy pętlę o rejestr przesuwny. W tym celu klikamy prawym przyciskiem myszy na prawej krawędzi pętli **While** i z menu wybieramy opcję **Add Shift Register**. Na lewej krawędzi pętli **While** pojawi się rejestr przesuwny wejściowy, a na prawej –wyjściowy. Prawym kliknięciem na rejestrze wejściowym otwieramy dodatkowe menu i dodajemy do rejestru jeden element, wybierając opcję **Add Element**. Do rejestru wyjściowego dołączamy wartość odczytaną z licznika. Obliczamy różnicę z dwóch wartości pobranych z zacisków rejestru wejściowego i tworzymy prawym kliknięciem wskaźnik numeryczny

**Difference of pulses**. Wskaźnik ten będzie więc pokazywał różnicę impulsów odczytanych z licznika w dwóch kolejnych obiegach pętli **While**. Możemy teraz testowo uruchomić program. Ponieważ pętla **While** jest wykonywana bardzo szybko, pokazywana liczba impulsów będzie niewielka i zależna od ustawionej zworką częstotliwości wzorcowej.

Umieszczamy opóźnienie czasowe wewnątrz pętli **While**, wstawiając funkcję **Programming > Timing > Wait Until Next ms Multiple**. Na jej wejściu tworzymy prawym kliknięciem zadajnik (**Create Control**). Ustawiamy domyślną wartość opóźnienia na 1000 ms. Ten czas opóźnienia pętli będzie czasem pomiaru $T_P$ zgodnie z zależnością (3.2), dlatego na panelu zmieniamy etykietę tego zadajnika na **Measurement time [ms]**. Czas pomiaru przeliczamy z milisekund na sekundy, aby otrzymać wynik pomiaru częstotliwości w hercach. Różnicę liczby impulsów **Difference of pulses** dzielimy przez czas pomiaru przeliczony na sekundy. Na wyjściu operatora dzielenia tworzymy prawym kliknięciem wskaźnik numeryczny i nadajemy mu nazwę **Frequency [Hz]**.

Uruchamiamy program przyciskiem  . Jeśli program nie działa prawidłowo, znajdujemy i poprawiamy błędy. Aktualizujemy odpowiednio panel i diagram niezbędnymi informacjami, analogicznie jak w poprzednim punkcie.

Przełączamy program z trybu symulacji na tryb pomiaru rzeczywistym licznikiem. Ustawiamy zworką generator na częstotliwość wzorcową 1800Hz. Sprawdzamy, czy program poprawnie zmierzył częstotliwość impulsów. Odczytujemy i notujemy w protokole kilka kolejnych wyników pomiaru częstotliwości, sprawdzając, czy są one jednakowe. Analizujemy, jakiego rzędu są rozrzuty wartości kolejnych wyników, i notujemy wnioski w protokole. Sprawdzamy działanie programu po ustawieniu czasu pomiaru na zero, po czym staramy się wyjaśnić zaobserwowany efekt i notujemy wnioski. Zwiększamy czas pomiaru od zera co jedną milisekundę i sprawdzamy, dla jakiego najkrótszego czasu pomiaru program zacznie poprawnie mierzyć częstotliwość, a odpowiedni wniosek notujemy w protokole.

Przełączamy program w tryb symulacji i ustawiamy symulowaną częstotliwość na 1000 Hz. Powtarzamy eksperymenty w taki sam sposób, jak w przypadku licznika rzeczywistego, a wnioski notujemy w protokole.

Aktualizujemy odpowiednio opisy na panelu i diagramie programu zgodnie z rycinami 3.15 i 3.16. Zapisujemy trzeci program na dysku i notujemy nazwę pliku w protokole.

Drukujemy dokumentację programu do pliku, po czym odnajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwy plików wraz z opisem zawartości notujemy w protokole, a same pliki (panel i diagram) wykorzystujemy do przygotowania raportu. Wykonujemy kopię programu na potrzeby następnego kroku, nadając jej nową nazwę z końcówką 3-4-a.

### 3.5.4. Czwarty program z tworzeniem histogramu, obliczaniem błędów i niepewności

Czwarty program w ćwiczeniu 3 uzupełniamy o dodatkową analizę wyników pomiaru: tworzenie histogramu, uśrednianie wyników, obliczanie błędów i niepewności, obliczanie poziomu zapełnienia licznika oraz kasowanie zbędnych wyników pomiarów. Taki program umożliwia sprawdzanie generatorów częstotliwości wzorcowej [29] oraz częstościomierzy cyfrowych [30]. Lepszą metodą sprawdzania generatorów wzorcowych jest bezpośrednie porównywanie ich częstotliwości z częstotliwością wzorcową o wyższej dokładności [31]. Źródłem takiej częstotliwości wzorcowej o wysokiej dokładności może być nawet stosunkowo niedrogi odbiornik nawigacji satelitarnej GPS [32] lub odbiornik radiowych sygnałów czasu systemu DCF77.

Aby zmniejszyć ryzyko popełnienia błędów podczas programowania, niezbędne zmiany i uzupełnienia w programie będziemy realizować etapami. Końcową wersję programu po niezbędnych modyfikacjach przedstawiono na rycinach 3.22 (panel) i 3.23 (diagram). Odpowiednie zmiany w programie opracowanym w poprzednim punkcie wprowadzamy w opisanych poniżej kilku krokach.

Otwieramy zapisaną w poprzednim punkcie kopię programu 3-4-a. Uruchamiamy go, przełączamy w tryb pomiarów rzeczywistych z wykorzystaniem licznika i upewniamy się, czy pomiary realizowane są prawidłowo. Jeśli program działa poprawnie, zaczynamy uzupełniać jego diagram o funkcję budowania tablicy z wynikami pomiarów oraz tworzenia histogramu, zgodnie z przykładem pokazanym na rycinie 3.17. Najpierw rozciągamy ramkę pętli **While** w dół, aby utworzyć dodatkowy wolny obszar na uzupełnianie diagramu.

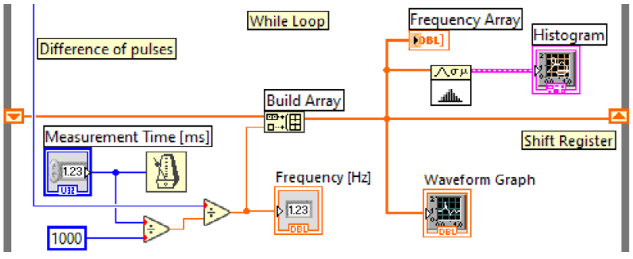
W pierwszej kolejności uzupełniamy diagram tak, aby tworzyć histogram na podstawie kolejnych zmierzonych wartości częstotliwości. Histogram wykorzystuje dane w formie tablicy (**Array**). Aby utworzyć tablicę z kolejnych

wyników pomiaru, tworzymy rejestr przesuwny, klikając prawym przyciskiem myszy na krawędzi pętli **While** i wybierając opcję **Add Shift Register**. Pomiędzy utworzonymi rejestrami przesuwnymi wstawiamy funkcję **Programming > Array > Build Array** i rozciągamy ją w dół, aby utworzyć dwa terminale wejściowe. Łączymy rejestry przesuwne z wejściem i wyjściem funkcji **Build Array**. Dla wygody tworzenia diagramu przenosimy fragment obliczający częstotliwość z górnej do dolnej części pętli **While**, co można łatwo zaobserwować, porównując ryciny 3.16 i 3.23.

Następnie dołączamy wynik obliczania częstotliwości do dolnego zacisku funkcji **Build Array**, zgodnie z ryciną 3.19. W ten sposób każdy kolejny wynik pomiaru częstotliwości zostanie dodany do tablicy jako następny w kolejności element. Na wyjściu tablicy tworzymy wskaźnik (**Create Indicator**) i nadajemy mu nazwę **Frequency Array**. Na panelu rozciągamy ten wskaźnik w dół, aby pokazywał kilka kolejnych elementów tablicy (przykład pokazano na rycinie 3.22).

Dodajemy do diagramu funkcję **Mathematics > Prob & Stat > Histogram** i tworzymy prawym kliknięciem (**Create Indicator**) na jej wyjściu wykres typu **Histogram Graph**, przedstawiający rozkład ze zgromadzonych w tablicy wyników pomiarów częstotliwości. Na panelu powinien automatycznie pojawić się wykres z histogramem, który przesuwamy w widoczne miejsce.

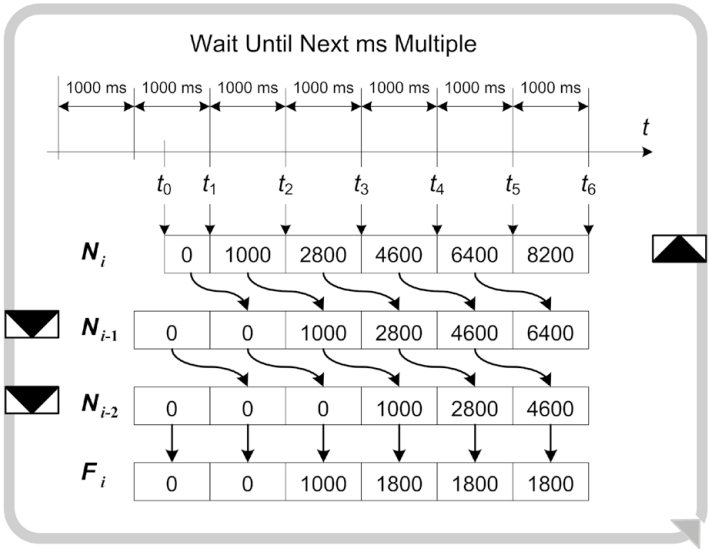
Uzupełniamy panel programu o wykres pokazujący kolejne wyniki pomiarów zgromadzone w tablicy. W tym celu umieszczamy na panelu obiekt graficzny **Controls > Graph > Waveform Graph**, a na diagramie dołączamy do jego wejścia tablicę z wynikami pomiarów. Przykładowa postać tej części diagramu po modyfikacjach jest pokazana na rycinie 3.17. Na panelu zmieniamy opisy osi histogramu i wykresu **Waveform Graph**, wzorując się na rycinie 3.22.




**Ryc. 3.17.** Przykładowy fragment diagramu tworzący tablicę wyników i histogram

Uruchamiamy program i sprawdzamy, czy tablica z wynikami pomiarów jest budowana zgodnie z oczekiwaniami, oraz oceniamy poprawność tworzonego histogramu i wykresu z wynikami pomiarów. Zwracamy uwagę na kilka pierwszych wyników zgromadzonych w tablicy i na wykresie. Analizujemy, czy wszystkie wyniki są zgodne z ustawioną częstotliwością wzorcową, a wnioski notujemy w protokole.

Program w obecnej postaci nie mierzy poprawnie częstotliwości od razu po uruchomieniu. Pierwsze wyniki pomiarów nie są poprawne, co wyjaśniono na rycinie 3.18.




**Ryc. 3.18.** Wyjaśnienie przyczyny błędnych wyników pierwszych trzech pomiarów: częstotliwość mierzona 1800 Hz, opóźnienie w pętli 1000 ms, pierwszy odczyt licznika w chwili $t_0$ nie jest zsynchronizowany, początkowe wartości w rejestrach są równe zero

Przyjęto dla niej wartość mierzonej częstotliwości 1800 Hz oraz czas pomiaru 1000 ms. Problem wynika z faktu, że program bezpośrednio po uruchomieniu potrzebuje jednego obiegu pętli **While** na zsynchronizowanie się z zegarem systemowym, z którego korzysta funkcja **Wait Until Next ms Multiple** podczas wyznaczania kolejnych chwil odczytu licznika. Poza tym wejściowe rejestry przesuwne początkowo mają wartości równe zero i dopiero w trzecim obiegu pętli są tam dostępne wyniki odczytane z licznika. Ale pierwszy odczyt licznika nie jest poprawny, ponieważ pierwszy obieg pętli jest krótszy od ustawionego w pętli **While** czasu opóźnienia. W efekcie

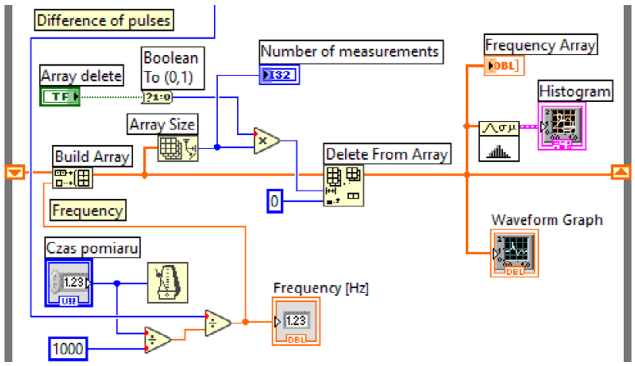
tego dopiero czwarta wartość częstotliwości jest obliczana poprawnie. Z tego powodu również histogram jest początkowo niepoprawny, ponieważ zawiera kilka pierwszych wyników znacząco odstających od wszystkich następnych. Po zatrzymaniu i ponownym uruchomieniu programu sytuacja jest podobna, ale rejestry przesuwne pamiętają poprzednie wartości. Podobny problem występuje, gdy podczas pracy programu zmieniamy czas pomiaru. Jeśli chcemy dokładniej zgłębić to zagadnienie, możemy wstawić do diagramu i na panel dodatkowe wskaźniki pokazujące zawartość rejestrów wejściowych. Umożliwi nam to dokładne zaobserwowanie opisywanego zjawiska.

Uzupełniamy więc program o możliwość kasowania zawartości tablicy z błędnymi wynikami pomiarów, dzięki czemu uzyskamy poprawny histogram. Odpowiednią, kolejną modyfikację diagramu pokazano na rycinie 3.19.

Na diagramie kasujemy połączenie na wyjściu funkcji **Build Array** i wstawiamy w to miejsce funkcję **Programming > Array > Delete From Array** kasującą elementy w tablicy. Na wejściu **Index 0** tworzymy prawym kliknięciem stałą wartość równą 0, dzięki czemu kasowane będą elementy od pierwszego (w środowisku LabVIEW pierwszy element tablicy ma indeks 0). Używamy funkcji **Programming > Array > Array Size** do wyznaczenia liczby elementów w tablicy, które mają być usunięte (usuwane będą wszystkie elementy). Wartość tę mnożymy przez 0 lub 1, w zależności od tego, czy zawartość tablicy ma być skasowana, czy nie.

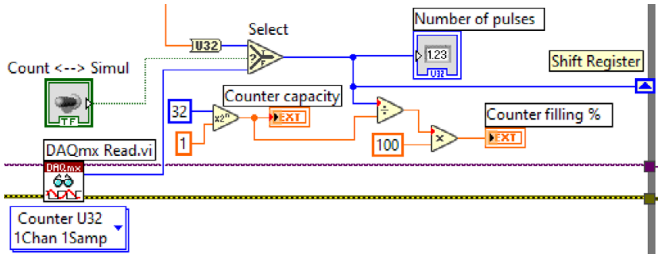
Zwróćmy uwagę, że symbole 0 i 1 mają w środowisku LabVIEW podwójne znaczenie: mogą reprezentować wartości logiczne typu **Boolean** lub wartości numeryczne typu **integer**. Operacja mnożenia liczb jest możliwa tylko na zmiennych numerycznych, natomiast obiekty na panelu typu **Boolean** (przyciski, przełączniki) dostarczają zmiennych logicznych tego samego typu, które należy przekształcić na zmienne numeryczne. W tym celu umieszczamy na diagramie funkcję **Programming > Boolean > Bool to (0, 1)** i na jej wejściu tworzymy prawym kliknięciem zadajnik logiczny **Create Control**. Prawym kliknięciem wchodzimy we właściwości (**Properties**) tego zadajnika i wybieramy opcję **Operation > Latch When Relased** oraz nadajemy mu nazwę **Array delete**. Wstawiamy na diagram funkcję mnożenia i uzupełniamy połączenia zgodnie z ryciną 3.19. Na wyjściu funkcji **Array Size** tworzymy wskaźnik (**Create Indicator**), który będzie pokazywał liczbę elementów w tablicy (**Number of measurements**).

Uruchamiamy program i sprawdzamy działanie przycisku do kasowania tablicy, zwracając uwagę na to, czy liczba elementów w tablicy jest poprawnie wyświetlana oraz czy kasowanie całej jej zawartości przebiega poprawnie. Weryfikujemy również, czy po skasowaniu zawartości tablicy wyniki są poprawne już od pierwszego pomiaru i czy zawartość histogramu i wykresu także ulega wyczyszczeniu. Wszystkie wnioski z tych obserwacji notujemy w protokole.




**Ryc. 3.19.** Przykładowy fragment diagramu kasujący zawartość tablicy z wynikami pomiarów

W kolejnym kroku modyfikacji programu uzupełniamy diagram o kontrolę i wizualizację poziomu zapełnienia licznika w procentach. Przykład odpowiedniej części uzupełnionego diagramu pokazano na rycinie 3.20.

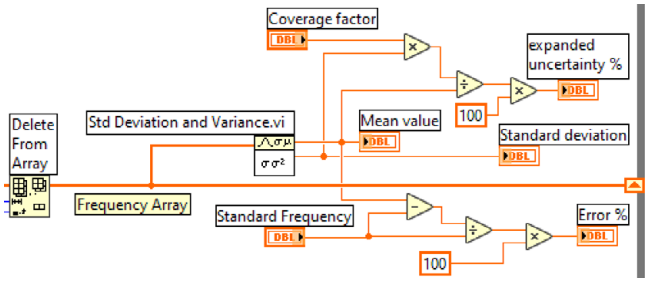



**Ryc. 3.20.** Przykładowy fragment diagramu obliczający poziom zapełnienia licznika

Najpierw obliczamy pojemność 32-bitowego licznika (używając funkcji **Programming > Numeric > Scale By Power Of 2**), dzielimy przez nią bieżącą zawartość licznika i przeliczamy wynik na procenty. Umieszczamy na panelu wskaźnik liniowy **Controls > Numeric > Horizontal Fill Slide** z włączonym wyświetlaczem cyfrowym (**Visible Items > Digital Display**) i zakresem ustawionym od 0 do 100 %.

Uzupełniamy program tak, aby obliczał błędy i niepewności pomiarowe. Odpowiedni fragment uzupełnionego diagramu pokazano na rycinie 3.21.

Najpierw dla całej tablicy wyników obliczamy wartość średnią i odchylenie standardowe, używając funkcji **Mathematics > Prob & Stat > Std Deviation** and **Variance**. Na panelu umieść zadajnik numeryczny Standard Frequency i nadaj mu początkową wartość aktualnie ustawionej na generatorze częstotliwości wzorcowej $F_{\text{Ref.}}$ Następnie oblicz względny błąd procentowy wartości średniej zmierzonej częstotliwości w odniesieniu do wartości częstotliwości wzorcowej $F_{\text{Ref}}$ i wyświetl wartość błędu na panelu.



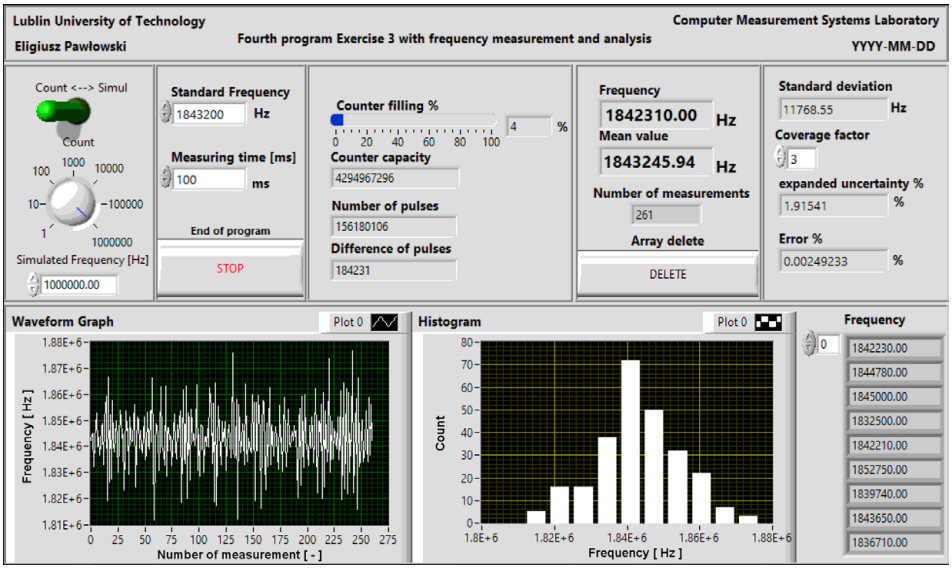

**Rys. 3.21.** Przykładowy fragment diagramu obliczający błędy i niepewności pomiarowe

Na panelu umieść zadajnik numeryczny Coverage Factor i nadaj mu typową przy pomiarach wartość współczynnika rozszerzenia równą 3, przy kalibracji zalecaną wartością jest 2. Z odchylenia standardowego oblicz w procentach względną niepewność rozszerzoną odniesioną do wartości średniej i wyświetl jej wartość na panelu. Uruchom program i sprawdź jego działanie. Zatrzymaj program i wykonaj kalkulatorem kontrolne obliczenia błędu i niepewności. Porównaj je z wartościami pokazywanymi przez program. Czy błędy i niepewności są obliczane poprawnie? Zapisz wnioski do protokołu.

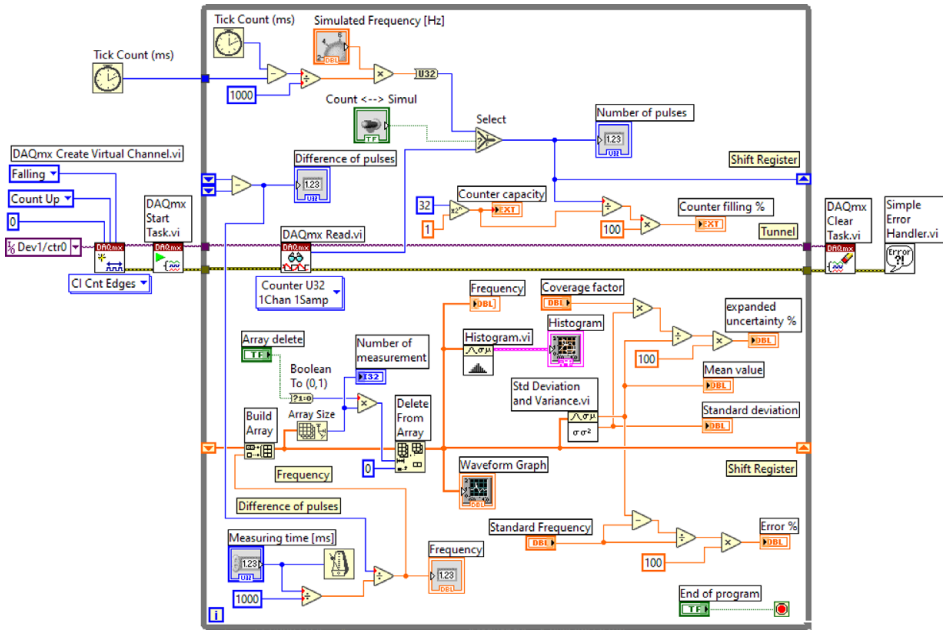
Uzupełnij panel i diagram niezbędnymi informacjami analogicznie jak w poprzednich punktach. Uporządkuj zawartość panelu i diagramu. Przykładowa postać kompletnego programu jest pokazana na rysunkach 3.22 i 3.23.

Jeśli program działa poprawnie konfigurację programu jako domyślną poleceniem **Edit > Make Current Values Default**. Zapisujemy końcową wersję programu na dysku i notujemy nazwę pliku i katalogu w protokole. Drukujemy dokumentację programu do pliku, po czym odnajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu oraz

nazwy plików z opisem zawartości notujemy w protokole, a same pliki wykorzystujemy do przygotowania raportu.



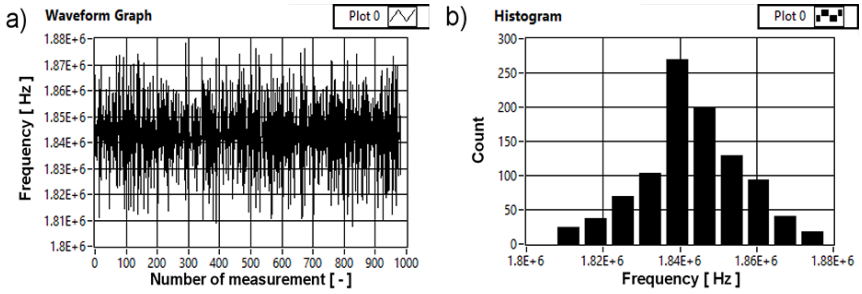

**Ryc. 3.22.** Przykładowy panel końcowej wersji programu



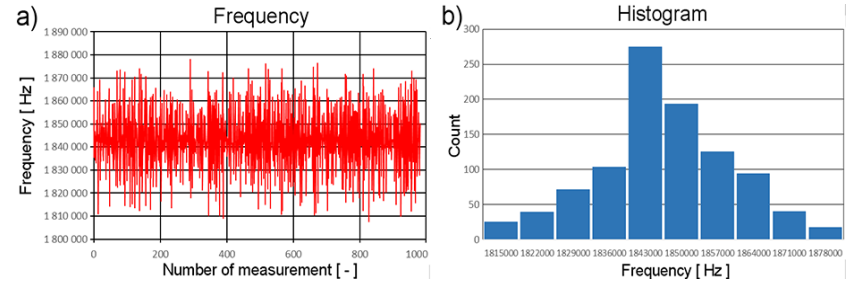

**Ryc. 3.23.** Przykładowy diagram końcowej wersji programu

## 3.6. Badania generatora wzorcowego

Ustawiamy zworkę na płytce generatora (ryc. 3.2) w pozycji największej dostępnej wartości częstotliwości wzorcowej i wprowadzamy ją również do programu w polu **Standard Frequency**. Uruchamiamy program, kasujemy zawartość tablicy i wykonujemy serię co najmniej 100 pomiarów. Zapisujemy uzyskany wykres i histogram do plików graficznych BMP (klikając prawym przyciskiem myszy i wybierając polecenie **Export > Export Simplified Image**) i zamieszczamy je w raporcie. Przykładowy wykres i histogram pokazano na rycinie 3.24. Dodatkowo za pomocą polecenia **Export > Export Data To Excel** zapisujemy dane z wykresu **Waveform Graph** do pliku i wykorzystujemy je do wykonania wykresu i histogramu w programie Excel (lub innym o podobnych możliwościach). Przykładowy wykres i histogram opracowany w arkuszu kalkulacyjnym pokazano na rycinie 3.25.




**Ryc. 3.24.** Wyniki pomiarów uzyskane w programie LabVIEW: a) wykres, b) histogram




**Ryc. 3.25.** Wyniki pomiarów opracowane w programie Excel: a) wykres, b) histogram

Na wykresie i histogramie zauważamy, że wyniki pomiarów częstotliwości ulegają pewnym fluktuacjom. Jednym z powodów są nieuchronne zakłócenia przenikające do układu pomiarowego. Dokładność pomiaru częstotliwości

możemy zwiększyć, stosując uśrednianie wagowe [33], co jest bardzo skuteczną metodą tłumienia zakłóceń okresowych (np. pochodzących z sieci energetycznej 50 Hz).

Przełączamy program do pracy symulacyjnej i ustawiamy wartość symulowanej częstotliwości na tę samą, która była wprowadzona na generatorze częstotliwości wzorcowej. Powtarzamy w trybie symulacji eksperymenty, które wykonaliśmy wcześniej w układzie rzeczywistym z wykorzystaniem licznika. Porównujemy przede wszystkim histogramy otrzymane w trybie symulacji i z pomiarów rzeczywistych. Analizujemy, czy są do siebie podobne, czy też różnią się w istotny sposób, a zaobserwowane zjawisko staramy się wyjaśnić w raporcie.

Korzystając z opracowanego programu, badamy właściwości generatora częstotliwości wzorcowej zastosowanego na panelu ćwiczeniowym. Pomiary przeprowadzamy dla kilku nastaw częstotliwości wzorcowej i kilku czasów pomiaru, według tabeli 3.4. Liczba pomiarów $N$ powinna wynosić od kilkudziesięciu do kilkuset.

W raporcie analizujemy otrzymane wyniki. Zastanawiamy się, które z trzech badanych parametrów (odchylenie standardowe w hercach, niepewność względna w procentach i błąd względny w procentach) zależą od wartości mierzonej częstotliwości, a które od czasu trwania pomiaru, oraz sprawdzamy, czy obie te wartości wpływają na każdy z parametrów w jednakowy sposób.

**Tab. 3.4.** Wyniki badań generatora częstotliwości wzorcowych

| $F_{Ref}$ | Time | N | Std. Dev. | Uncertainty | Error |
|---|---|---|---|---|---|
| Hz | ms | - | Hz | % | % |
| 1843200 | 10 | ... | ... | ... | ... |
| 1843200 | 100 | ... | ... | ... | ... |
| 1843200 | 1000 | ... | ... | ... | ... |
| 14400 | 1000 | ... | ... | ... | ... |
| 1800 | 1000 | ... | ... | ... | ... |

## 3.7. Opracowanie raportu z realizacji ćwiczenia

Raport z przeprowadzonego ćwiczenia powinien zawierać:

- stronę tytułową z danymi osobowymi zespołu, datą ćwiczenia i jego tytułem,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań zgodnie z programem ćwiczenia,
- opis każdego z wykonanych zadań lub podanie przyczyny jego niewykonania,
- dokumentację czterech zrealizowanych i uruchomionych,
- wykresy wygenerowane przez program w środowisku LabVIEW,
- wykresy opracowane w arkuszu kalkulacyjnym Excel,
- liczbowe wyniki pomiarów uzyskane za pomocą opracowanych programów,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.

W raporcie ocenie podlegają:

- poprawność działania programów,
- czytelność i przejrzystość diagramów,
- estetyka paneli, dobór kolorów i elementów graficznych,
- funkcjonalność panelu pod kątem wygody obsługi,
- zakres zrealizowanych zadań,
- trafność wniosków końcowych.

# 4. Badanie silnika DC małej mocy

## 4.1. Cel ćwiczenia i zadania do wykonania

Ćwiczenie 4 jest kontynuacją ćwiczenia 3. Trzeci program do pomiaru częstotliwości impulsów, zrealizowany w ćwiczeniu 3, stanowi podstawę do realizacji ćwiczenia 4.

Celem ćwiczenia 4 jest zapoznanie studentów z problematyką programowania układu pomiarowego w środowisku **LabVIEW** z wykorzystaniem przetwornika cyfrowo-analogowego DAC (**Digital to Analog Converter**) i uniwersalnego licznika impulsów, które są elementami składowymi modułu akwizycji danych pomiarowych NI USB-6001.

Program ćwiczenia 4 obejmuje następujące zagadnienia:

- testowanie przetwornika DAC i licznika impulsów w programie NI-MAX,
- konfigurowanie modułu NI USB-6001 do konwersji cyfrowo-analogowej,
- konfigurowanie modułu NI USB-6001 do zliczania impulsów ze zmianą domyślnego wejścia PFI 0, stosowanego w ćwiczeniu 3, na wejście PFI 1,
- pomiary prędkości obrotowej silnika prądu stałego małej mocy poprzez zliczanie impulsów z przetwornika obrotowo-impulsowego dołączonego do wejścia PFI 1,
- automatyczne testowanie silnika w zadanym przedziale napięcia zasilającego.

Zadania realizowane w trakcie ćwiczenia polegają na przygotowaniu odpowiednich programów w środowisku LabVIEW, uruchomieniu ich, sprawdzeniu poprawności ich działania, usunięciu błędów, wykonaniu pomiarów i sporządzeniu raportu.

W ramach ćwiczenia 4 studenci opracowują trzy programy:

1. Pierwszy program do prostego sterowania ręcznego przetwornikiem cyfrowo-analogowym,
2. Drugi program do automatycznego sterowania przetwornikiem cyfrowo-analogowym w pętli FOR dla zadanego przedziału napięć,
3. Trzeci program do automatycznego testowania silnika elektrycznego prądu stałego.

## 4.2. Opis stanowiska laboratoryjnego

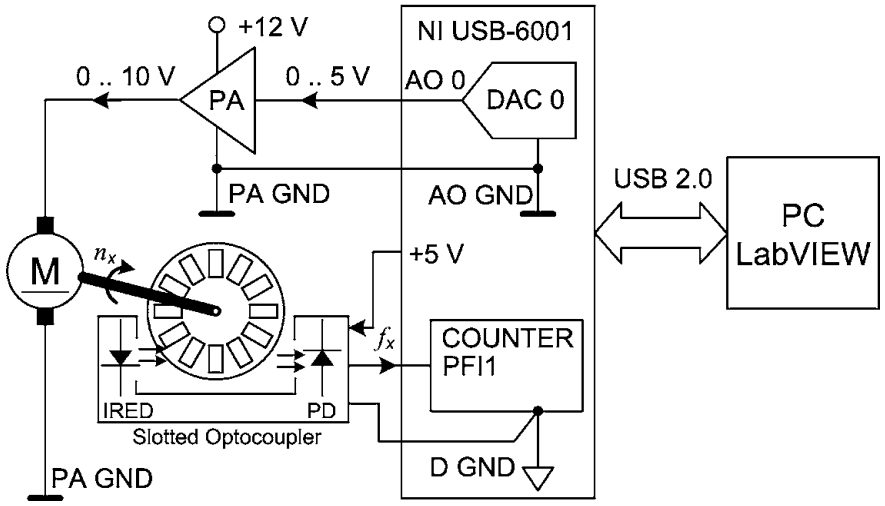
### 4.2.1. Schemat blokowy stanowiska laboratoryjnego

Stanowisko laboratoryjne wykorzystywane w ćwiczeniu 4 składa się z następujących komponentów:

- komputera PC z zainstalowanym środowiskiem LabVIEW,
- modułu akwizycji danych pomiarowych NI USB-6001,
- wzmacniacza mocy prądu stałego do zasilania silnika prądu stałego małej mocy,
- modułu ćwiczeniowego z silnikiem elektrycznym prądu stałego i przetwornikiem obrotowo-impulsowym.

Schemat blokowy stanowiska laboratoryjnego wykorzystywanego w ćwiczeniu 4 przedstawiono na rycinie 4.1. Moduł akwizycji danych pomiarowych USB-6001 jest dołączony do komputera PC za pomocą interfejsu USB. Wyjście AO0 (**Analog Output 0**) przetwornika cyfrowo-analogowego DAC jest dołączone do wejścia wzmacniacza mocy prądu stałego PA (**Power Amplifier**). Wzmacniacz mocy PA jest zasilany z zasilacza 12 V/3 A i wzmacnia napięcie dwukrotnie. Napięcie wyjściowe wzmacniacza PA zasila badany silnik. Wał silnika jest połączony z przetwornikiem obrotowo-impulsowym, który składa się z dwóch części: zamocowanej na wale silnika tarczy z 12 otworami i transoptora szczelinowego (**Slotted Optocoupler**). Tarcza z otworami umieszczona

jest w szczelinie transoptora. Transoptor po jednej stronie szczeliny posiada diodę IRED emitującą w podczerwieni (**InfraRed Emitting Diode**), a po drugiej stronie fotodiodę PD (**Photo Diode**). Tak więc każdy obrót wału silnika generuje 12 impulsów na wyjściu transoptora. Wyjście transoptora jest podłączone do wejścia PFI 1 (**Programmable Function Interface 1**) licznika w module pomiarowym NI USB-6001. Należy zauważyć, że w ćwiczeniu 3 było wykorzystywane wejście PFI 0 licznika impulsów. Z tego względu w ćwiczeniu 4 konieczne jest odpowiednie przeprogramowanie konfiguracji licznika.




**Ryc. 4.1.** Schemat blokowy stanowiska laboratoryjnego w ćwiczeniu 4

### 4.2.2. Budowa i parametry licznika wbudowanego w module NI USB-6001

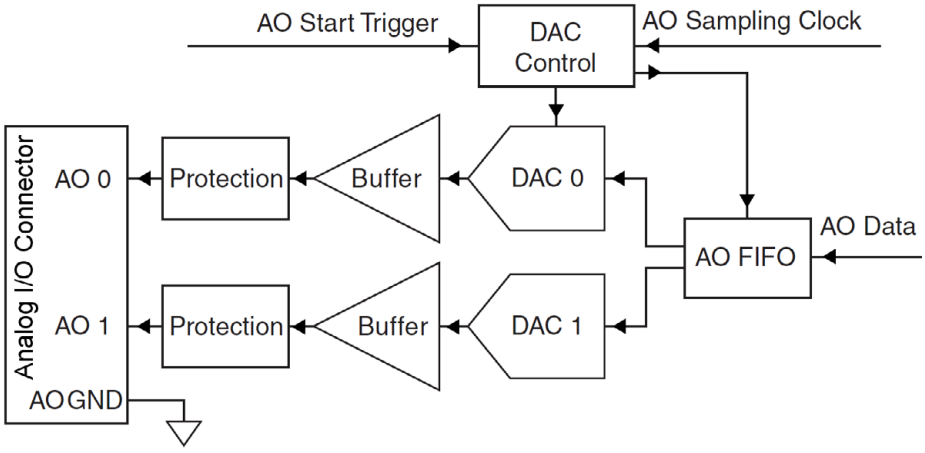
Tabela 4.1 przedstawia najważniejsze parametry licznika wbudowanego w moduł NI USB-6001. Parametry licznika są szczegółowo omówione w ćwiczeniu 3. W ćwiczeniu 4 należy jednak zwrócić uwagę, że licznik ma rozdzielczość 32 bitów, co oznacza, że może zliczyć $2^{32} = 4\ 294\ 967\ 296$ impulsów. Natomiast testowany w ćwiczeniu silnik elektryczny ma maksymalną prędkość obrotową 5000 obr./min przy napięciu znamionowym 24 V, więc maksymalna częstotliwość impulsów z wyjścia przetwornika obrotowo-impulsowego z 12 otworami w tarczy wynosi 1000 Hz. Licznik może więc taką częstotliwość zliczać bez przepełnienia przez prawie 50 dni bez przerwy. Pojemność licznika jest więc do takich badań wystarczająca z bardzo dużym zapasem.

**Tab. 4.1.** Główne parametry licznika impulsów modułu NI USB-6001

| Parametry | Wartości |
|---|---|
| Rozdzielczość | 32 bity |
| Standard impulsów | TTL |
| Możliwe pomiary | zliczanie zboczy narastających lub opadających |
| Kierunek zliczania | tylko w górę |
| Źródło zliczanych impulsów | PFI 0 lub PFI 1, domyślne wejście PFI 0 |
| Maksymalna częstotliwość impulsów | 5 MHz |
| Minimalny czas trwania stanu wysokiego | 100 ns |
| Minimalny czas trwania stanu niskiego | 100 ns |

### 4.2.3. Budowa i parametry przetwornika DAC wbudowanego w module NI USB-6001

Na rycinie 4.2 przedstawiono schemat blokowy analogowych układów wyjściowych w module NI USB-6001. Dostępne są dwa identyczne, niezależne przetworniki DAC. Wyjście każdego z przetworników jest buforowane i zabezpieczone przed przeciążeniem. Oba przetworniki posiadają wspólną masę AO GND. Każdy z przetworników ma swój własny rejestr FIFO o pojemności 2047 próbek.




**Ryc. 4.2.** Schemat blokowy analogowych układów wyjściowych w module NI USB-6001

Tabela 4.2 przedstawia najważniejsze parametry tych przetworników. Każdy przetwornik ma rozdzielczość 14 bitów i zakres napięć wyjściowych od –10 V do +10 V. Oznacza to, że krok kwantowania przetwornika wynosi 1,22 mV. Typowa wartość błędu wynosi 9,1 mV na końcu zakresu, ale maksymalna wartość to 34 mV w dopuszczalnym zakresie temperatur. Uwzględniając krok kwantowania przetwornika i jego błędy, uzasadnione jest zadawanie napięcia wyjściowego z rozdzielczością 1 mV. Maksymalny prąd wyjściowy wynosi tylko 5 mA, a więc niezbędne jest zastosowanie dodatkowego wzmacniacza mocy, aby sterować większym obciążeniem, np. silnikiem elektrycznym badanym w ćwiczeniu 4.

**Tab. 4.2.** Główne parametry przetwornika cyfrowo-analogowego modułu NI USB-6001

| Parametry | Wartości |
|---|---|
| Liczba wyjść analogowych | 2 |
| Rozdzielczość | 14 bitów |
| Zakres napięć wyjściowych | –10 V do +10 V |
| Maksymalna szybkość przetwarzania | 5000 próbek na sekundę w każdym kanale jednocześnie, symultanicznie |
| Maksymalny prąd wyjściowy | ±5 mA |
| Ograniczenie zwarciowe | ±11 mA |
| Błąd typowy na końcu zakresu | 9,1 mV |
| Błąd maksymalny w zakresie temperatur | 34 mV |

### 4.2.4. Budowa i parametry wzmacniacza mocy PA

Badany silnik elektryczny może pobierać maksymalnie prąd o wartości 0,9 A, natomiast maksymalny prąd wyjściowy przetwornika DAC wynosi tylko 5 mA. Dlatego na panelu ćwiczeniowym umieszczono odpowiedni wzmacniacz napięcia stałego dużej mocy. Zastosowano do tego celu wzmacniacz zbudowany w oparciu o układ scalony TDA2050 produkcji STMicroelectronics [34]. Wykorzystano odpowiednio zmodyfikowany gotowy moduł wzmacniacza akustycznego [35, 36]. Wzmacniacz ten będzie szczegółowo omówiony i badany w ćwiczeniu 5, dlatego na potrzeby ćwiczenia 4 podane są w tabeli 4.3 tylko jego główne parametry. Należy zwrócić uwagę, że zastosowany wzmacniacz wzmacnia napięcia stałe DC, co odróżnia go od wzmacniaczy akustycznychpracujących typowo w paśmie od 20 Hz do 20 kHz. Wzmocnienie napięciowe

wynosi 2 V/V, ale wzmacniacz pracuje poprawnie dla napięć wyjściowych powyżej około 0,7 V i nasyca się przy napięciu około 10,5 V. Poszczególne egzemplarze wzmacniacza mogą różnić się nieznacznie pod tym względem. Moduł wzmacniacza jest na stałe zamocowany na panelu ćwiczeniowym i połączony z pozostałą częścią układu, nie wymaga więc wykonywania żadnych połączeń podczas realizacji ćwiczenia. Napięcie wyjściowe wzmacniacza PA AO (**Power Amplifier Analog Output**) dostępne jest na styku 5, a jego masa PA GND na styku 17 złącza IDC20, do którego będzie dołączony moduł z badanym silnikiem elektrycznym.

**Tab. 4.3.** Główne parametry wzmacniacza mocy prądu stałego [34, 35]

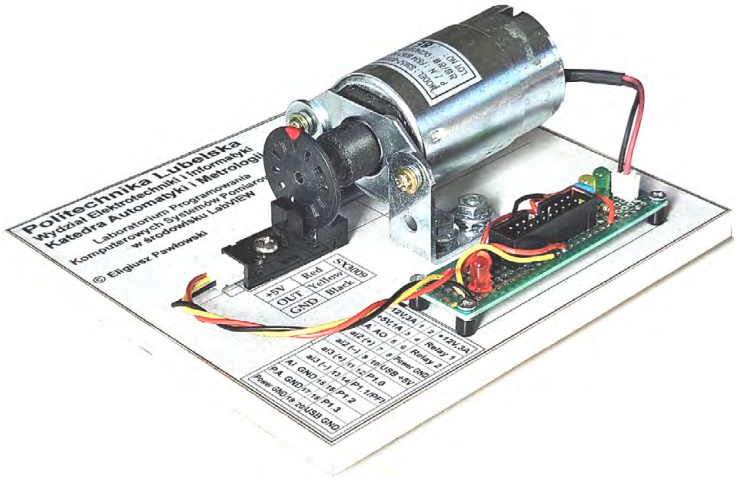
| Parametry | Wartości |
|---|---|
| Napięcie zasilania | +12 V niesymetryczne względem masy |
| Maksymalna moc | 50 W przy zasilaniu ±22,5 V |
| Ograniczenie prądowe | 2 $A_{RMS}$ |
| Pasmo przenoszenia | DC ... 20 kHz |
| Wzmocnienie napięciowe | 2 V/V |
| Zakres napięć wyjściowych | 0,7 V ... 10,5 V |

### 4.2.5. Budowa i parametry modułu z badanym silnikiem DC małej mocy

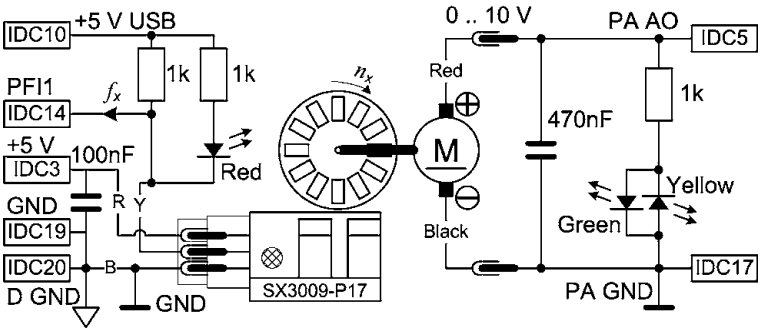
Badany silnik umieszczony jest na dodatkowym module ćwiczeniowym, przedstawionym na rycinie 4.3. Podstawę modułu stanowi płyta HDF o wymiarach 13 cm × 15 cm, na której zamontowano badany silnik, transoptor szczelinowy i płytkę PCB ze złączem IDC20. Na wale silnika zamocowana jest tarcza impulsatora mająca 12 otworów. Czerwony znacznik na tarczy ułatwia policzenie otworów.

Schemat ideowy połączeń na module z badanym silnikiem przedstawiono na rycinie 4.4. Główne parametry przepisane z tabliczki znamionowej silnika przedstawiono w tabeli 4.4. Silnik zasilany jest napięciem z wyjścia PA AO wzmacniacza mocy, które jest pobierane ze styku 5 złącza IDC względem masy dostępnej na styku 17. Ponieważ badany silnik jest klasycznym silnikiem szczotkowym, dołączono do niego równolegle kondensator 470 nF tłumiący zakłócenia. Dwie diody LED w kolorach zielonym i żółtym sygnalizują biegunowość tego napięcia. Dioda zielona świeci dla napięć dodatnich. Koło impulsatora,

zamocowane na wale badanego silnika, umieszczone jest w szczelinie transoptora SX2009-P17 firmy OMRON [37]. Transoptor zasilany jest napięciem +5 V z zasilacza sieciowego, dostępnym na styku 3 złącza IDC.




**Ryc. 4.3.** Moduł do badania silnika DC małej mocy stosowany w ćwiczeniu 4




**Ryc. 4.4.** Schemat modułu do badania silnika DC małej mocy

**Tab. 4.4.** Główne parametry badanego silnika prądu stałego z jego tabliczki znamionowej

| Parametry | Wartości |
|---|---|
| Producent | NISCA |
| Model | S3657-022200-3GHVFCPL |
| Napięcie zasilania znamionowe | DC 24 V max |
| Prąd | 0,9 A max |
| Prędkość obrotowa znamionowa | 5000 obr./min |

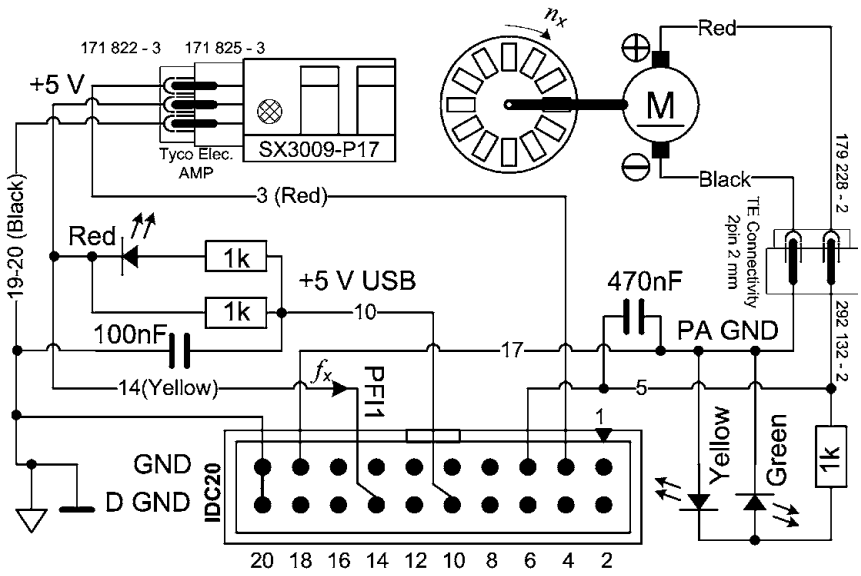
Budowę transoptora szczelinowego i jego schemat wewnętrzny pokazano na rycinie 4.5. Wewnątrz jego obudowy umieszczone są naprzeciw siebie dioda IRED emitująca podczerwień (w obudowie z dwiema nóżkami) oraz fotodioda PD ze wzmacniaczem i przerzutnikiem Schmitta gwarantującym stabilne i szybkie przełączanie stanu logicznego na wyjściu (w obudowie z trzema nóżkami). Transoptor ma wyjście cyfrowe typu otwarty kolektor, dlatego do jego wyjścia dołączono rezystor 1 kΩ typu **pull up**, podciągający wyjście transoptora do napięcia +5 V (styk 10 na złączu IDC, ryciny 4.4 oraz 4.6), pobranego z modułu NI USB-6001 (PIN 31 na cyfrowym złączu modułu, tabela 4.5). Gwarantuje to zachowanie odpowiednich parametrów sygnału cyfrowego w standardzie TTL. Bez rezystora podciągającego wyjście transoptora nigdy nie osiągnęłoby stanu logicznego wysokiego. Do wyjścia transoptora dodatkowo dołączona jest czerwona dioda LED. Umieszczenie przeszkody w szczelinie transoptora wymusza stan niski na jego wyjściu, co jest sygnalizowane zaświeceniem czerwonej diody LED.




**Ryc. 4.5.** Transoptor szczelinowy OMRON SC3009-P17: a) widok zewnętrzny, b) wnętrze po zdjęciu obudowy, c) schemat blokowy układu elektronicznego [37]

Moduł z badanym silnikiem dołączany jest do panelu ćwiczeniowego za pomocą płaskiego przewodu zakończonego wtyczkami IDC20. Natomiast na panelu ćwiczeniowym oraz na dodatkowych modułach zainstalowane są gniazda IDC20. Na rycinie 4.6 przedstawiono styki gniazda IDC20 wykorzystywane w module z badanym silnikiem. W ćwiczeniu 4 wykorzystywanych jest tylko 7 styków tego złącza, a pozostałe styki mają zastosowanie w innych ćwiczeniach. Styk numer 1 jest oznaczony na obudowie gniazda trójkątnym symbolem. Dodatkowo zaznaczono ten styk na obudowie gniazda czerwoną kropką. Podobne oznaczenia znajdują się na przewodzie połączeniowym, co ma ułatwić prawidłowe dołączanie wtyczki do gniazdka. Zwróćmy uwagę, że na płytce PCB połączone są ze sobą dwie masy: masa GND zasilacza sieciowego 5 V (styk 19) oraz masa cyfrowa D GND (styk 20) modułu pomiarowego

NI USB-6001. Zwróćmy również uwagę, że w obudowie gniazda IDC20 znajduje się wcięcie przy stykach 9 i 11, natomiast na wtyczkach IDC20 znajdują się w tym miejscu odpowiednie wypustki. Jest to tzw. klucz sprzętowy, zapewniający odpowiedni sposób dołączania wtyczek do gniazd. **Uwaga: próba nieodpowiedniego dołączenia wtyczki spowoduje uszkodzenie gniazda !**



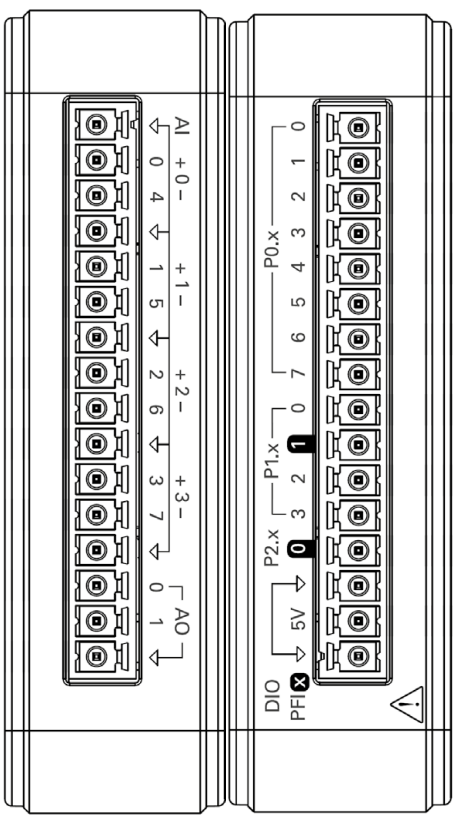

**Ryc. 4.6.** Złącze IDC i zaznaczone styki wykorzystywane w ćwiczeniu 4

## 4.3. Dołączanie wyjścia przetwornika DAC i wejścia licznika modułu NI USB-6001

Rozmieszczenie sygnałów analogowych i cyfrowych na dwóch złączach modułu NI USB-6001 pokazane jest w tabeli 4.5. Wyjście przetwornika DAC znajduje się na lewym złączu sygnałów analogowych. Dostępne są dwa przetworniki cyfrowo-analogowe. W ćwiczeniu 4 wykorzystamy wyjście analogowe AO 0 na styku PIN 14 oraz masę analogową wyjściową AO GND na styku PIN 16. Wejścia licznika znajdują się na prawym złączu sygnałów cyfrowych. Dostępny jest jeden licznik, który ma dwa wejścia PFI 0 oraz PFI 1. W ćwiczeniu 4 wykorzystamy wejście PFI 1 licznika na styku PIN 26 oraz masę cyfrową

D GND na styku PIN 32. Dodatkowo wykorzystamy również napięcie zasilające +5 V dostępne na styku PIN 31. Wszystkie linie sygnałowe i masy oraz zasilanie wykorzystane w tym ćwiczeniu wyróżniono pogrubioną czcionką w tabeli 4.5.

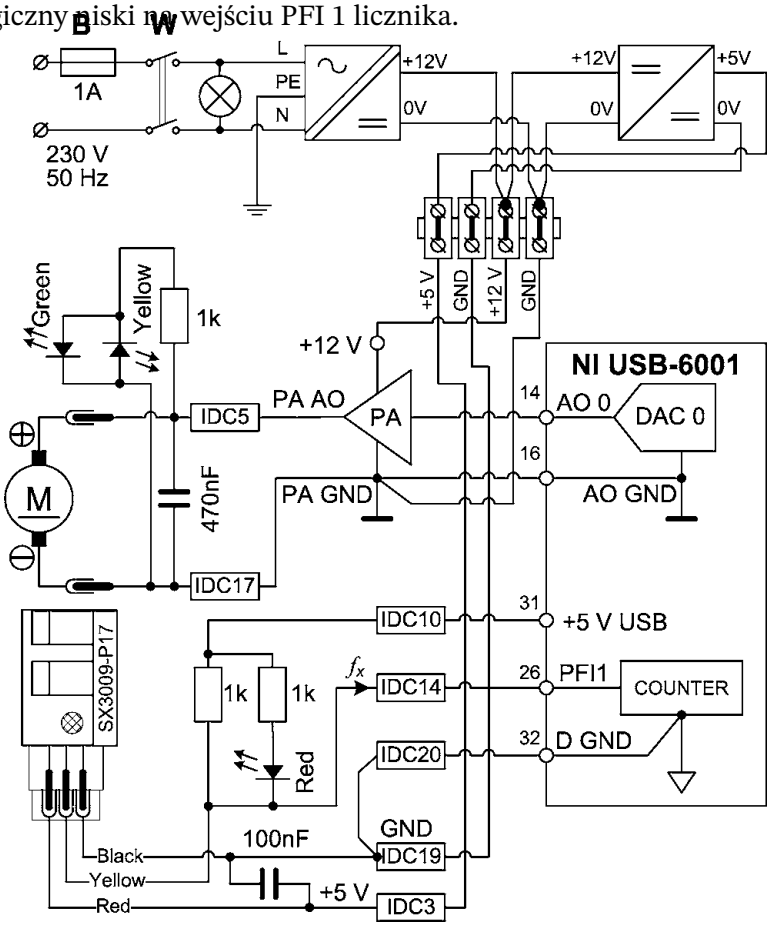
**Tab. 4.5.** Rozmieszczenie sygnałów analogowych (po lewej) i cyfrowych (po prawej) na złączach modułu NI USB-6001, pogrubioną czcionką zaznaczono styki wykorzystane w ćwiczeniu 4 [5]

| Sygnał | PIN | Złącze analogowe i cyfrowe | PIN | Sygnał |
|---|---|---|---|---|
| AI GND | 1 |  | 17 | P0.0 |
| AI 0 (AI 0+) | 2 | | 18 | P0.1 |
| AI 4 (AI 0–) | 3 | | 19 | P0.2 |
| AI GND | 4 | | 20 | P0.3 |
| AI 1(AI 1+) | 5 | | 21 | P0.4 |
| AI 5 (AI 1–) | 6 | | 22 | P0.5 |
| AI GND | 7 | | 23 | P0.6 |
| AI 2(AI 2+) | 8 | | 24 | P0.7 |
| AI 6 (AI 2–) | 9 | | 25 | P1.0 |
| AI GND | 10 | | **26** | **P1.1/PFI 1** |
| AI 3(AI 3+) | 11 | | 27 | P1.2 |
| AI 7 (AI 3–) | 12 | | 28 | P1.3 |
| AI GND | 13 | | 29 | P2.0/PFI 0 |
| **AO 0** | **14** | | 30 | D GND |
| AO 1 | 15 | | **31** | **+5V/150mA** |
| **AO GND** | **16** | | **32** | **D GND** |



Rycina 4.7 przedstawia schemat połączeń układu pomiarowego zastosowanego w ćwiczeniu 4. Dwa zasilacze sieciowe dostarczają napięcia +12 V do zasilania wzmacniacza mocy PA oraz napięcia +5 V do zasilania transoptora szczelinowego. Wejście wzmacniacza mocy PA dołączone jest na stałe do wyjścia AO 0 przetwornika cyfrowo-analogowego na styku PIN 14. Wyjście wzmacniacza mocy PA dołączone jest do styku 5 gniazda IDC, skąd dalej poprzez płaską taśmę 20-przewodową zasila dodatni zacisk badanego silnika M.

Masa analogowa wzmacniacza mocy PA dołączona jest do ujemnego zacisku silnika M poprzez styk IDC 17. Transoptor szczelinowy zasilany jest napięciem +5 V z zasilacza 5 V/1 A poprzez styk IDC 3 i masę zasilaczy na styku IDC 19. Wyjście transoptora jest podciągnięte rezystorem **pull-up** do napięcia +5 V dostępnego na styku IDC 10, dołączonego do styku PIN 31 modułu NI USB-6001. Zielona i żółta dioda LED sygnalizują polaryzację napięcia na styku IDC 5, zasilającego badany silnik, odpowiednio dodatnią i ujemną względem masy analogowej PA GND na styku IDC 17. Czerwona dioda LED sygnalizuje stan logiczny niski na wejściu PFI 1 licznika.




**Ryc. 4.7.** Schemat układu połączeń stanowiska do ćwiczenia 4

## 4.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 4.4.1. Uruchamianie stanowiska laboratoryjnego

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Zasilanie komputera i panelu ćwiczeniowego powinno być wyłączone. Moduł pomiarowy powinien być odłączony od portu USB komputera**.

Wszelkie zauważone odstępstwa od tego stanu zgłaszamy prowadzącemu zajęcia.

Sprawdzamy układ zgodnie z opisem w poprzednim punkcie. Jeśli układ jest połączony prawidłowo, zgłaszamy prowadzącemu zajęcia gotowość do rozpoczęcia realizacji ćwiczenia.

**Dopiero po otrzymaniu od prowadzącego zajęcia wyraźnej zgody włączamy tylko zasilanie komputera. Nie włączamy jeszcze zasilania panelu ćwiczeniowego. Nie dołączamy modułu pomiarowego do portu USB komputera.**

Po uruchomieniu się systemu operacyjnego nie wykonujemy żadnych czynności i przechodzimy do realizacji kolejnego punktu.

### 4.4.2. Self-test modułu NI USB-6001

Po uruchomieniu komputera najpierw sprawdzamy, czy w systemie są zainstalowane dwa urządzenia NI USB-6001: jedno fizyczne jako **Dev1** i drugie symulowane jako **Dev2**. W tym celu uruchamiamy program NI MAX, rozwijamy listę **Devices and Interfaces** i sprawdzamy zainstalowane w systemie urządzenia (ryc. 1.12). Notujemy w protokole nazwy, typy, numery seryjne i statusy zainstalowanych w systemie urządzeń z serii NI USB. Moduł używany na panelu ćwiczeniowym powinien być jeszcze odłączony, a więc powinien być oznaczony na liście czerwonym znakiem „x" i mieć status **Not Present**. Jeśli na liście nie ma wymaganych urządzeń, instalujemy je ponownie zgodnie z opisem w instrukcji do ćwiczenia 1.

Pamiętajmy, że moduł używany i zainstalowany już w systemie na wcześniejszych zajęciach, mógł zostać z nieznanych powodów (najczęściej przypadkowo) usunięty przez innych studentów. Dlatego na początku zajęć zawsze upewniamy się, jaki numer urządzenia został przypisany w systemie do

używanego aktualnie modułu pomiarowego. Upewniamy się również, czy na każdych zajęciach pracujemy na tym samym stanowisku laboratoryjnym, gdyż każdy wyprodukowany moduł DAQ ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczonych w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**, jeśli więc wykorzystywany moduł został zainstalowany pod inną nazwą, musimy odpowiednio zmodyfikować tworzone programy.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera. Nie załączamy jeszcze zasilania na panelu ćwiczeniowym.** Sprawdzamy zachowanie niebieskiej kontrolki LED na obudowie modułu. Notujemy wynik obserwacji w protokole.

Jeśli używane podczas zajęć urządzenie NI USB-6001 zostało poprawnie zainstalowane, w programie NI MAX przeprowadzamy **self-test** modułu fizycznego i symulowanego zgodnie z opisem zamieszczonym w instrukcji do pierwszego ćwiczenia, w sposób pokazany na rycinie 1.13. Notujemy rezultaty **self-testu** w protokole i zapisujemy odczytane numery seryjne modułów.

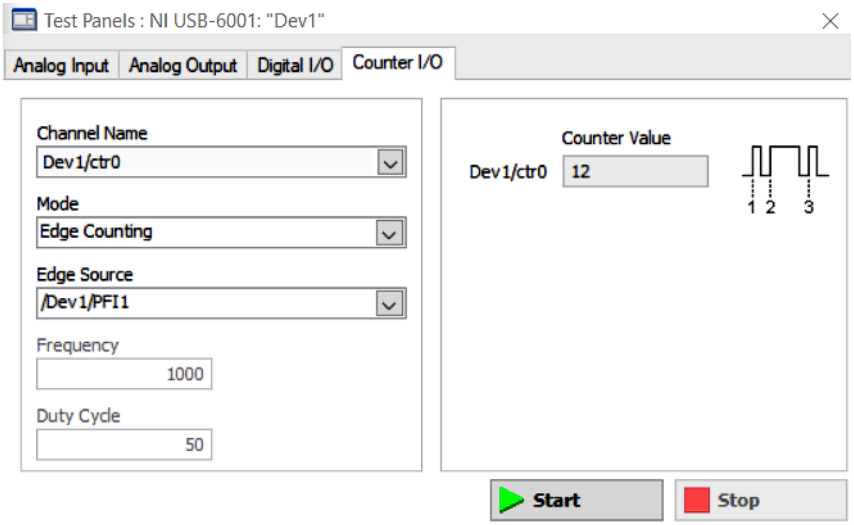
**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody włączamy zasilanie panelu ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku powinno potwierdzić obecność napięcia zasilającego.

### 4.4.3. Testowanie zliczania impulsów z przetwornika obrotowo-impulsowego

Test zliczania impulsów wykonujemy przy dołączonym do wejścia PFI 1 przetworniku obrotowo-impulsowym, połączonym z badanym silnikiem. Po sprawdzeniu układu połączeń załączamy zasilanie stanowiska laboratoryjnego. W otwartym oknie **Measurement & Automation Explorer** wybieramy zakładkę **Test Panels** (ryc. 1.14). Otworzy się okno **Test Panels**, w którym wybieramy zakładkę **Counter I/O**. Konfigurujemy test licznika zgodnie z rycinią 4.8, zwracając uwagę na wybór wejścia **Dev1/PFI 1** w polu **Edge Source**.

Uruchamiamy test, naciskając przycisk **START**. W oknie Counter Value liczba impulsów powinna wynosić zero i nie zmieniać się. Teraz ręcznie powoli obracamy wałem silnika i sprawdzamy, czy liczba impulsów wzrasta. Zgodnie z symbolem impulsów obok pola **Counter Value**, licznik powinien zliczać

narastające zbocza. Obserwując czerwoną diodę LED dołączoną do wyjścia transoptora szczelinowego, sprawdzamy, kiedy licznik zlicza kolejny impuls – przy zapaleniu czy przy zgaszeniu czerwonej diody LED. Zatrzymujemy test przyciskiem **STOP**. Uruchamiamy ponownie test i ręcznie wykonujemy dokładnie jeden obrót wałem silnika. Sprawdzamy, ile impulsów zostało zliczonych. Liczba zliczonych impulsów powinna być zgodna z liczbą otworów w tarczy. Czy przeprowadzony test to potwierdza? Notujemy w protokole zaobserwowane rezultaty wszystkich przeprowadzonych testów. Dodatkowo wykonujemy przykładowy zrzut ekranu z przeprowadzonego testu i zamieszczamy go w raporcie, wzorując się na rycinie 4.8.



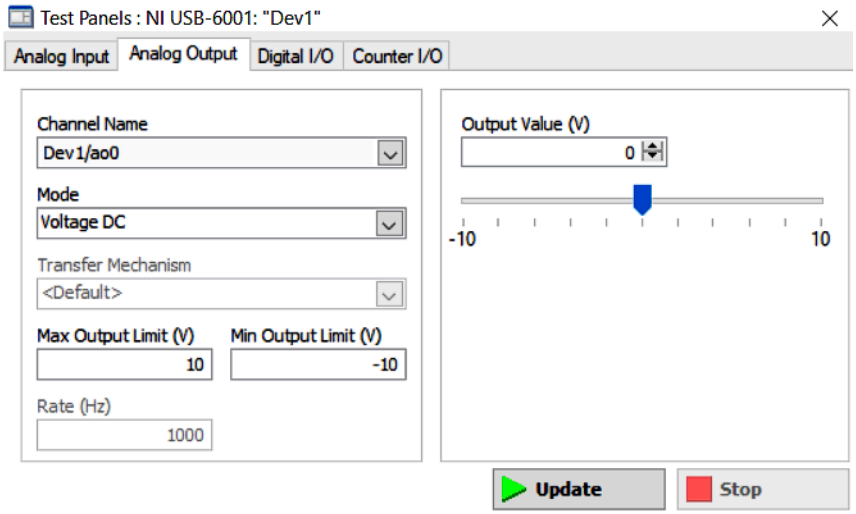

**Ryc. 4.8.** Testowanie licznika impulsów na wejściu PFI 1 w programie NI MAX

Powtarzamy test licznika dla urządzenia symulowanego. Sprawdzamy, jakie wartości zwraca test licznika w urządzeniu symulowanym i czy ulegają one zmianie.

**Pozostawiamy otwarte okno Counter** I/O skonfigurowane do testowania licznika fizycznego z wejściem PFI 1, **uruchamiamy test licznika** i przechodzimy do testowania przetwornika cyfrowo-analogowego.

### 4.4.4. Testowanie przetwornika cyfrowo-analogowego

Test przetwornika cyfrowo-analogowego wykonujemy przy dołączonym do wyjścia analogowego AO 0 wzmacniaczu mocy PA i module z badanym silnikiem. Pozostawiamy otwarte okno **Test Panels** z uruchomionym testem licznika w aktywnej zakładce **Counter I/O**. Ponownie w otwartym oknie **Measurement & Automation Explorer** wybieramy zakładkę **Test Panels**. Otworzy się kolejne okno **Test Panels**, w którym wybieramy zakładkę **Analog Output**. Ustawiamy na ekranie obok siebie oba okienka **Test Panels**. Konfigurujemy test wyjścia analogowego zgodnie z ryciną 4.9, zwracając uwagę na wybór wyjścia **Dev1/ao0** w polu **Channel Name**. Zwróćmy uwagę, że fizycznie dostępne na złączu modułu wyjście analogowe AO 0 pisane jest wielkimi literami (tabela 4.5), natomiast w programie odpowiadająca mu nazwa kanału ao0 pisana jest małymi literami (ryc. 4.9).




**Ryc. 4.9.** Testowanie wyjścia analogowego w programie NI MAX

W polu **Output Value** wpisujemy wartość 0 V i uruchamiamy test, naciskając przycisk **Update**. Zielona i żółta dioda LED na module z silnikiem powinny być zgaszone, co oznacza brak napięcia na wyjściu wzmacniacza mocy PA. Wał silnika powinien być nieruchomy, a czerwona dioda LED może być zapalona lub zgaszona zależnie od położenia wału silnika, co było już testowane w poprzednim punkcie. W oknie testowania licznika **Counter I/O** wartość **Counter Value** powinna być stała. Przy ręcznym obracaniu wałem

silnika czerwona dioda powinna migać. Sprawdzamy, czy układ zachował się poprawnie.

W polu **Output Value** wpisujemy wartość +5 V i ponownie uruchamiamy test, naciskając przycisk **Update**. Powinna zapalić się zielona dioda LED sygnalizująca dodatnie napięcie na silniku. Wał silnika powinien się obracać, a czerwona dioda LED powinna migać. W oknie testowania licznika **Counter I/O** wartość **Counter Value** powinna się zwiększać.

Wpisując różne wartości napięcia w polu **Output Value** i zatwierdzając je przyciskiem **Update**, sprawdzamy, przy jakim napięciu zaczyna świecić zielona dioda LED. Postępując analogicznie, ustalamy, przy jakim napięciu zaczyna pracować testowany silnik, i zapisujemy te wartości.

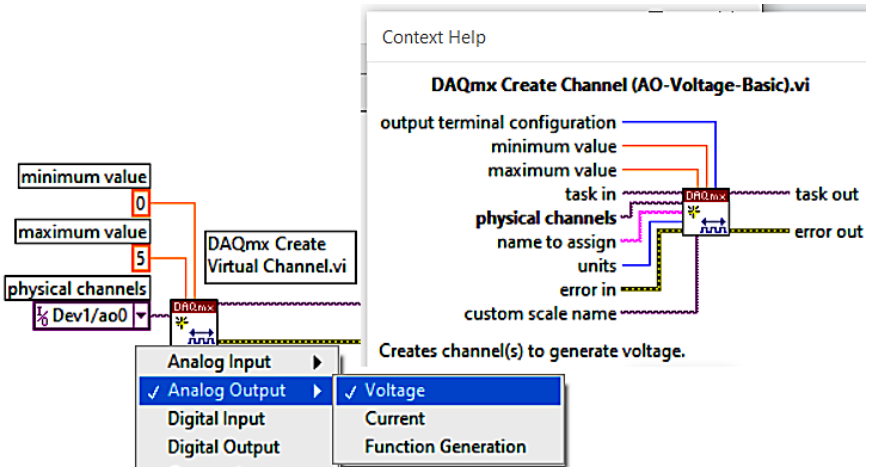
Notujemy w protokole zaobserwowane rezultaty wszystkich przeprowadzonych testów. Dodatkowo wykonujemy przykładowy zrzut ekranu z przeprowadzonego testu i zamieszczamy go w raporcie, wzorując się na rycinie 4.9.

## 4.5. Programowanie

### 4.5.1. Pierwszy program do ręcznego sterowania przetwornikiem DAC w pętli WHILE

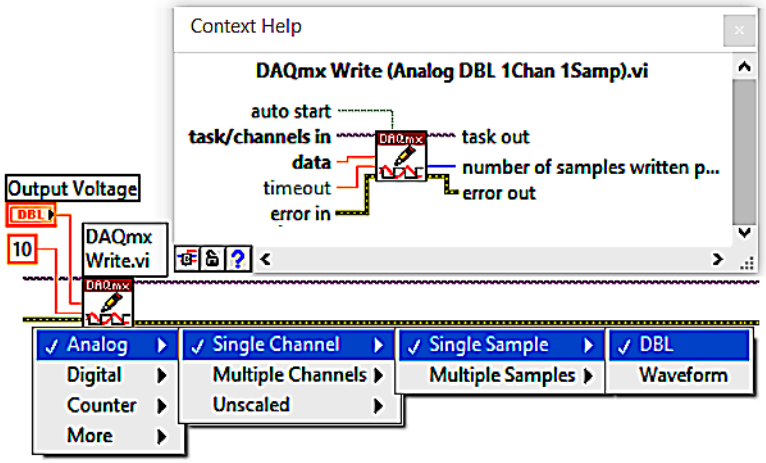
Aby ręcznie sterować wartością napięcia na wyjściu przetwornika cyfrowo-analogowego, przygotowujemy odpowiedni program. Najpierw tworzymy kanał wirtualny, konfigurujemy go, przypisując mu odpowiedni kanał fizyczny wybranego modułu pomiarowego powiązany z określonymi zaciskami na jego złączu, i definiujemy parametry kanału: wartość minimalną i wartość maksymalną. Następnie uruchamiamy zadanie (**Start Task**), które będzie reprezentowało utworzony kanał wirtualny na diagramie połączeń. Gdy zadanie jest uruchomione, możemy zapisać do niego wartość napięcia (**Write**). Kończąc działanie przyrządu wirtualnego, kończymy zadanie (**Clear Task**) i obsługujemy ewentualne błędy, które mogą pojawić się podczas działania programu. Docelową postać panelu programu pokazano na rycinie 4.12. Odpowiedni diagram pokazano na rycinie 4.13, ale przygotowujemy go etapowo w przedstawionej poniżej kolejności.

Najpierw tworzymy kanał wirtualny i konfigurujemy wyjście analogowe w sposób pokazany na rycinie 4.10. Kolejność działań jest następująca: uruchamiamy środowisko LabVIEW, tworzymy nowy pusty przyrząd wirtualny (**Create Project > Blank VI**) i przełączamy się na diagram (**Ctrl+E**). Otwieramy pomocnicze okno pomocy **Context Help**, klikając na znak zapytania w prawym górnym rogu okna programu. Prawym kliknięciem w oknie diagramu otwieramy zasobnik z funkcjami (**Functions**) i wybieramy zasobnik z funkcjami pomiarowymi dla modułów DAQmx: **Measurement I/O>NI DAQmx**. W tym zasobniku znajdują się wszystkie niezbędne funkcje do obsługi modułów pomiarowych NI DAQmx. Najpierw wybieramy funkcję tworzącą wirtualny kanał **Create Channel**, umieszczamy ją na pustym diagramie i, rozwijając selektor poniżej ikony funkcji, wybieramy konfigurację wyjścia analogowego: **Analog Output > Voltage**. Korzystając z okna pomocy **Context Help**, prawym przyciskiem myszy tworzymy stałe (**Create > Constant**) na wejściach: **minimum value = 0**, **maximum value = 5**. Podczas tworzenia stałej na wejściu **physical channels** wybieramy opcję przeglądania i z rozwiniętej listy dostępnych urządzeń wybieramy aktualnie używany moduł **Dev1/ao0**. Dla zwiększenia czytelności tworzonego diagramu dodajemy etykietę do każdej funkcji: prawym kliknięciem na ikonie funkcji otwieramy menu konfiguracyjne i wybieramy opcję **Visible Items > Label**. Porządkujemy wszystkie postawione na diagramie obiekty tak, aby zajmowały jak najmniej miejsca, nie zakrywały się wzajemnie oraz żeby wszystkie połączenia były dobrze widoczne.




**Ryc. 4.10.** Konfiguracja wyjścia analogowego DAQmx Create Channel

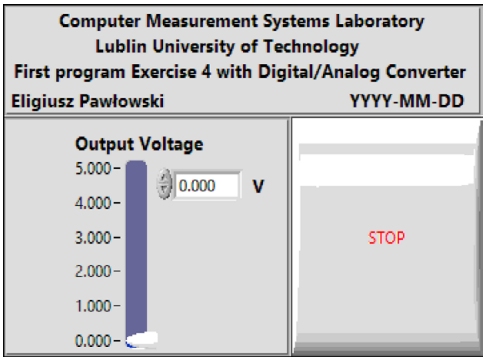
Następnie umieszczamy na diagramie następujące funkcje: **Start Task**, **Write** (dwukrotnie) oraz **Clear Task**. Z menu **Functions > Programming > Dialog & User Interface** wstawiamy funkcję **Simple Error Handler**. Włączamy widoczność etykiety dla każdej funkcji (**Visible Items > Label**). Przykładowe rozmieszczenie tych funkcji na diagramie pokazano na rycinie 4.13. Funkcje: **Start Task**, **Clear Task** oraz **Simple Error Handler** nie wymagają konfiguracji. Konfigurujemy pierwszą od lewej strony funkcję **Write** w sposób pokazany na rycinie 4.11: **Analog > Single Channel > Single Sample > DBL**, dzięki czemu każde wywołanie tej funkcji zapisze jednorazowo wartość zmiennej o podwójnej precyzji DBL do zdefiniowanego wyjścia analogowego. Na wejściu **timeout** funkcji **Write** tworzymy stałą (**Create > Constant**), która definiuje czas oczekiwania (10 sekund), po którym zostanie zgłoszony błąd zapisu. Na wejściu **data** funkcji **Write** prawym przyciskiem myszy tworzymy zadajnik numeryczny (**Create > Control**) i nadajemy mu nazwę **Output Voltage**. Sprawdzamy, czy na panelu pojawiła się odpowiednia kontrolka, i prawym kliknięciem otwieramy menu, wybierając **Replace > Numeric > Vertical Pointer Slide**. Ponownie prawym kliknięciem otwieramy menu **Properties** i konfigurujemy zadajnik: **Appearance > Show digital display**, potem **Data Entry > Minimum = 0, Coerce, Maximum = 5, Coerce, Increment = 0.001, Coerce to nearest**, oraz na koniec **Scale > Scale Range Minimum = 0, Maximum = 5**. Panel powinien wyglądać podobnie jak na rycinie 4.12.




**Ryc. 4.11.** Konfiguracja funkcji Write do zapisu wyjścia analogowego

Drugą funkcję **Write** po prawej stronie diagramu konfigurujemy podobnie, z tym że na wejściu **data** zamiast zadajnika tworzymy wartość stałą (**Create > Constant = 0**), dzięki czemu po zatrzymaniu programu na wyjściu analogowym napięcie zostanie zmniejszone do zera.

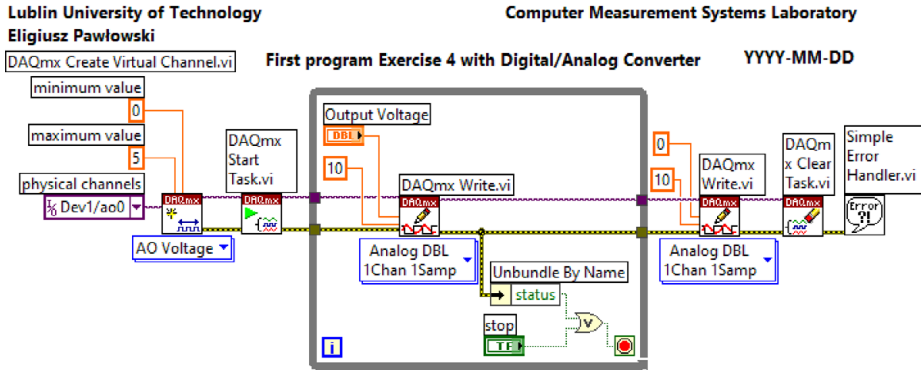
Z zasobnika **Functions > Programming > Structures** pobieramy pętlę **While Loop** i umieszczamy ją na diagramie w sposób pokazany na rycinie 4.13. Na terminalu warunku zakończenia pętli dodajemy prawym kliknięciem (**Create > Control**) przycisk STOP. Odszukujemy na panelu kontrolkę tego przycisku, powiększamy ją i umieszczamy w wygodnym do obsługi miejscu, przykładowo tak jak na rycinie 4.12.




**Ryc. 4.12.** Panel pierwszego programu do ręcznego sterowania przetwornikiem cyfrowo-analogowym DAC w pętli While

Następnie na diagramie łączymy wszystkie wyjścia **task out** z wejściami **task in** kolejnych modułów. Analogicznie łączymy wyjścia **error out** z wejściami **error in**. W miejscach przecięcia linii **data** i **error** z krawędzią pętli **While** powinny zostać automatycznie wstawione tunele (**Loop Tunnel**) w trybie przekazywania ostatniej wartości (**Last Value**). Sprawdzamy to, klikając prawym przyciskiem myszy na symbolu tunelu wyjściowego, i w razie potrzeby wybieramy opcję **Tunnel Mode > Last Value**. Powtarzamy te czynności dla pozostałych tuneli. Uzupełniamy program o kontrolę błędu zapisu funkcji **Write**. W tym celu dodajemy do diagramu funkcję **Programming > Cluster, Class, & Variant > Unbundle by Name** oraz funkcję **Programming > Boolean > OR**. Łączymy diagram tak, aby zatrzymać pętlę **While**, jeśli wystąpi błąd podczas zapisu wartości napięcia do wyjścia analogowego lub (funkcja OR) po wciśnięciu przycisku STOP. Porządkujemy elementy na diagramie

i panelu. Powinniśmy ostatecznie otrzymać diagram programu wyglądający podobnie jak na rycinie 4.13.




**Ryc. 4.13.** Diagram pierwszego programu do ręcznego sterowania przetwornikiem cyfrowo-analogowym DAC w pętli While

Jeśli środowisko LabVIEW nie zgłasza żadnych błędów, uruchamiamy program przyciskiem , co sygnalizowane jest zmianą formy przycisku na . Ponieważ program ma w swojej strukturze pętlę **While**, będzie on działał nieprzerwanie do momentu naciśnięcia przycisku STOP lub wystąpienia błędu. Pamiętajmy, że po włączeniu komputera wszystkie linie analogowe są automatycznie ustawiane na zero. Po uruchomieniu programu linia **AO 0** będzie miała napięcie ustawiane za pomocą zadajnika numerycznego na panelu programu, co możemy sprawdzić, obserwując pracę silnika i diody LED umieszczone na panelu silnika. Sprawdzamy, przy jakiej wartości napięcia zaczyna świecić zielona dioda LED oraz przy jakiej wartości napięcia zaczyna obracać się wał badanego silnika. Weryfikujemy również, czy po zatrzymaniu programu przyciskiem STOP silnik także się zatrzymuje, oraz czy zatrzymuje się on po przerwaniu działania programu przyciskiem **Abort Execution** (czerwony znak STOP na pasku narzędziowym). Zapisujemy wyniki tych obserwacji w protokole. Jeśli program nie działa prawidłowo, znajdujemy i poprawiamy błędy.

Korzystając z elementów graficznych dostępnych w zasobniku **Controls > Decorations**, porządkujemy i grupujemy obiekty na panelu. Uzupełnij panel i diagram o następujące informacje: nazwę przedmiotu, uczelnię, dane osobowe zespołu laboratoryjnego, numer grupy dziekańskiej, datę ćwiczenia, numer ćwiczenia, numer programu i jego tytuł. Informacje te podajemy w formie komentarza umieszczonego na panelu i na diagramie. Aktualizujemy opisy

wszystkich obiektów zgodnie z ich rolą. Używamy narzędzi **Align Objects**, **Distribute Objects**, **Reorder** dostępnych na pasku narzędziowym, aby wyrównać położenie obiektów na panelu, zgrupować je (**Group**) i zablokować położenie (**Lock**).

Zapisujemy program w katalogu wskazanym przez prowadzącego zajęcia. Nazwy plików z programami tworzymy według schematu:

YYYYMMDD_Nazwisko_KSP_X-N-L.vi

gdzie: YYYY – cyfry roku, MM – cyfry miesiąca, DD – cyfry dnia, X – numer ćwiczenia, N – numer programu w realizowanym ćwiczeniu, L – litera (a, b, c itd.) dla kolejnych wersji tego samego programu realizowanego w danym punkcie ćwiczenia. W nazwach plików nie wolno używać polskich znaków, spacji ani innych znaków specjalnych. Notujemy nazwę pliku i katalogu w protokole. W nazwie tego pliku powinien występować numer programu 4-1.

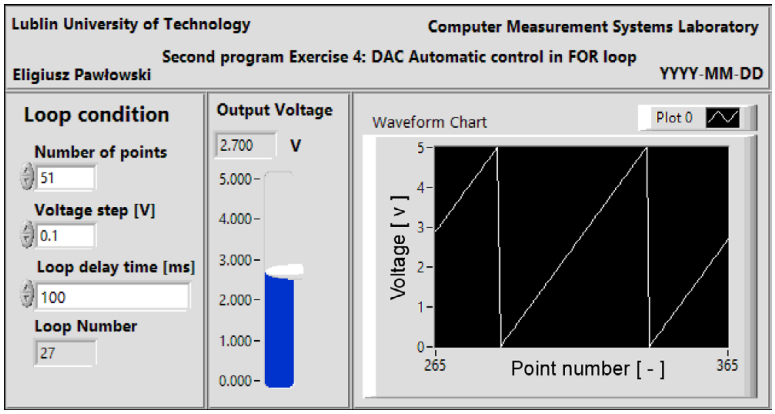
Wybierając opcję **File > Print > VI Documentation**, zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Zapisujemy nazwę utworzonego katalogu i nazwy plików z opisem zawartości w protokole. Używamy zapisanych plików do sporządzenia raportu.

Robimy od razu kopię programu dla realizacji następnego zadania, zmieniając numer programu na kolejny, tzn. 4-2-a. Nie modyfikujemy programu, który jest ostateczną wersją poprzednio wykonanego zadania. Zawsze pamiętamy, aby po zapisaniu ostatecznej wersji programu realizującego w pełni punkt w ćwiczeniu zrobić jej kopię, nadając jej nową nazwę i zmieniając ostatnią cyfrę na następną wartość.

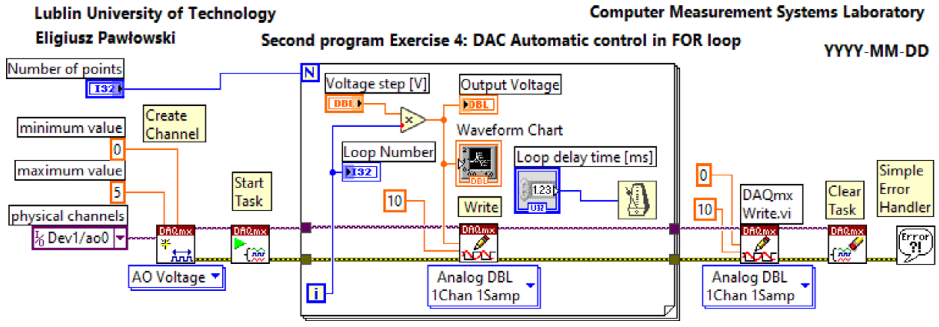
### 4.5.2. Drugi program do automatycznego sterowania przetwornikiem DAC w pętli FOR

Otwieramy kopię programu zapisaną na zakończenie poprzedniego punktu. Uruchamiamy go przyciskiem  i sprawdzamy, czy działa prawidłowo. Jeśli tak, uzupełniamy diagram i panel, jak pokazano na rycinach 4.14 i 4.15. Program powinien automatycznie zapisywać rosnące wartości napięcia do przetwornika cyfrowo-analogowego. Do programu możemy zadać liczbę kroków, przyrost napięcia w jednym kroku i czas trwania jednego kroku.

Kolejność postępowania jest następująca: najpierw zamieniamy pętlę **While** na pętlę **For**: prawym kliknięciem na krawędzi pętli While otwieramy menu i wybieramy opcję **Replace with For Loop**. Kasujemy z panelu przycisk STOP, który w poprzednim programie zatrzymywał pętlę While. Na wejściu zacisku N pętli **For** tworzymy zadajnik, nadajemy mu nazwę **Number of points** i ustawiamy jego początkową wartość na 51. Na wyjściu zacisku i pętli For tworzymy wskaźnik (**Create > Indicator**) i nadajemy mu nazwę **Loop Number**. Wewnątrz pętli For wstawiamy funkcję **Timing > Wait Until Next ms Multiple** i na jej wejściu tworzymy prawym kliknięciem zadajnik (**Create > Control**) do ustawiania czasu opóźnienia w milisekundach. Ustawiamy początkową wartość tego opóźnienia na 100 ms. Ta wartość będzie czasem trwania jednego kroku napięcia. Przy zadanych 51 punktach pętla wykona się w czasie 5 sekund.




**Ryc. 4.14.** Panel drugiego programu do automatycznego sterowania przetwornikiem cyfrowo-analogowym DAC w pętli For




**Ryc. 4.15.** Diagram drugiego programu do automatycznego sterowania przetwornikiem cyfrowo-analogowym DAC w pętli For

Ponieważ w tym programie napięcie będzie zadawane automatycznie, zbędny jest zadajnik napięcia **Output Voltage** dołączony do wejścia **data** funkcji **Write**. Zamieniamy ten zadajnik na wskaźnik, klikając na nim prawym przyciskiem myszki i wybierając z menu opcję **Change to Indicator**. Na diagramie pojawi się błąd połączenia, ponieważ wejście **data** funkcji **Write** nie ma określonej wartości. Wartość tę obliczamy, mnożąc wskaźnik **i** obiegu pętli For przez wartość kroku napięcia. W tym celu na panelu umieszczamy kontrolkę numeryczną (**Controls > Numeric > Numeric Control**) do zadawania wartości kroku napięcia i nadajemy jej początkową wartość 0,1 V. Uzupełniamy połączenia na diagramie zgodnie z ryciną 4.15. Uzupełniamy program wykresem pokazującym kolejne wartości napięcia: umieszczamy na panelu kontrolkę (**Controls > Graph > Waveform Chart**) i na diagramie dołączamy do jej wejścia zmienną **Output Voltage**. Modyfikujemy opisy osi wykresu, podając odpowiednie nazwy wielkości i jednostki miar.

Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne związki ze sobą. We wszystkich kontrolkach i wskaźnikach wpisujemy w pole **Properties > Appearance > Label** odpowiedni opis danego obiektu, a w pole **Properties > Appearance > Caption** wpisujemy symbol jednostki danej wielkości.

Uruchamiamy program przyciskiem  . Obserwujemy wskaźnik napięcia wyjściowego oraz wykres. Program powinien w ciągu 5 sekund zadać na wyjściu przetwornika DAC 51 wartości napięcia od 0 do +5 V z krokiem 0,1 V, a następnie ponownie ustawić wartość 0 V. W tym czasie silnik powinien obracać się z coraz większą prędkością i na koniec się zatrzymać. Zielona dioda LED powinna świecić z coraz większą jasnością i na koniec zgasnąć. Sprawdzamy, czy program działa zgodnie z oczekiwaniami, i zapisujemy wyniki obserwacji i wnioski w protokole.

Jeśli program działa prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Dzięki temu każdorazowe ponowne wczytanie tego pliku automatycznie odtworzy domyślne nastawy w programie.

Sprawdzamy działanie programu dla innych nastaw parametrów. Zwiększamy liczbę kroków i wartość przyrostu napięcia, aby przekroczyć maksymalne napięcie przetwornika DAC. Sprawdzamy, jak zachował się program

oraz silnik, a także czy został zgłoszony błąd. Jeśli pojawi się okno komunikatu z błędem, zapisujemy je do pliku i umieszczamy w raporcie z wyjaśnieniem treści otrzymanego komunikatu. Zapisujemy obserwacje i wnioski w protokole. Zapisujemy uzyskany przykładowy wykres napięcia do pliku graficznego BMP prawym kliknięciem za pomocą polecenia **Export > Export Simplified Image** i zamieszczamy go w raporcie.

Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdź ich zawartość. Zapisujemy nazwę utworzonego katalogu i nazwy plików z opisem zawartości w protokole. Używamy zapisanych plików do sporządzenia raportu.

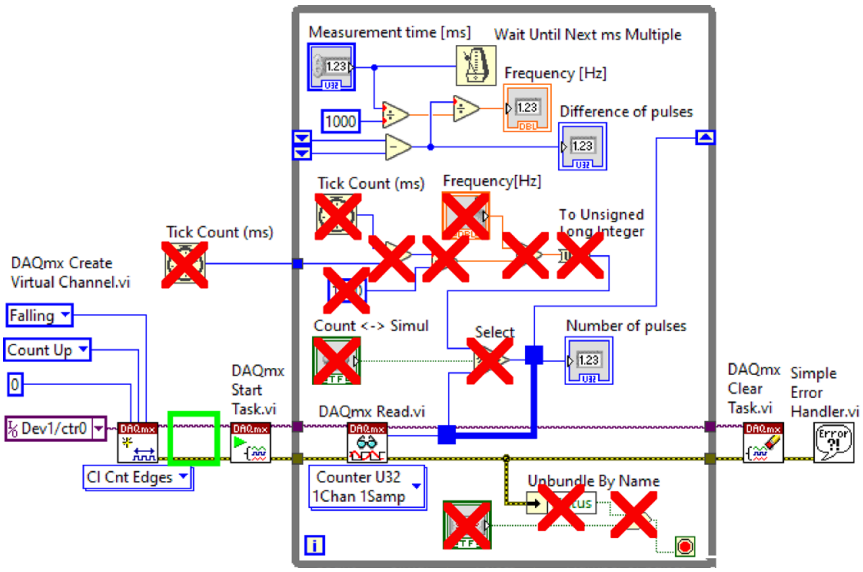
Robimy kopię programu dla realizacji następnego zadania, zmieniając numer programu na kolejny, tzn. 4-3. Nie modyfikujemy programu, który jest ostateczną wersją poprzednio wykonanego zadania. Zawsze pamiętamy, aby po zapisaniu ostatecznej wersji programu realizującego w pełni punkt w ćwiczeniu, zrobić jej kopię, nadając jej nową nazwę, zmieniając ostatnią cyfrę na następną wartość.

### 4.5.3. Trzeci program do automatycznego badania silnika DC małej mocy

Badanie silnika polega na automatycznym wyznaczeniu zależności prędkości obrotowej od napięcia zasilania. W tym celu program zadaje narastająco kolejne wartości napięcia do przetwornika DAC. Napięcie z przetwornika DAC jest dwukrotnie wzmacniane we wzmacniaczu mocy i zasila badany silnik. Dla każdej nowej wartości napięcia program mierzy częstotliwość impulsów na wyjściu transoptora szczelinowego i przelicza ją na prędkość obrotową [38]. Na podstawie wyników pomiarów program wykreśla charakterystykę silnika: prędkość obrotową w obrotach na minutę w funkcji napięcia zasilającego w woltach. Zadanie to realizujemy poprzez połączenie w jedną całość dwóch wcześniej już opracowanych programów: programu do pomiaru częstotliwości impulsów z ćwiczenia trzeciego i programu do automatycznego zadawania narastającego napięcia z wcześniejszego punktu ćwiczenia 4.

Ten punkt jest kontynuacją ćwiczenia 3. Trzeci program w ćwiczeniu 3 stanowi podstawę do uruchomienia programu do automatycznego badania silników elektrycznych prądu stałego małej mocy. Znajdujemy trzeci program

opracowany w trzecim ćwiczeniu (powinien mieć w nazwie pliku numer 3-3) i uruchamiamy go przyciskiem . Upewniamy się, że program działa prawidłowo. Jeśli tak, robimy kopię tego programu pod nową nazwą na potrzeby ćwiczenia 4, zmieniając odpowiednio w nazwie pliku numer programu na 4-3-a. Pamiętajmy, że nie modyfikujemy programu, który jest ostateczną wersją poprzednio wykonanego zadania.



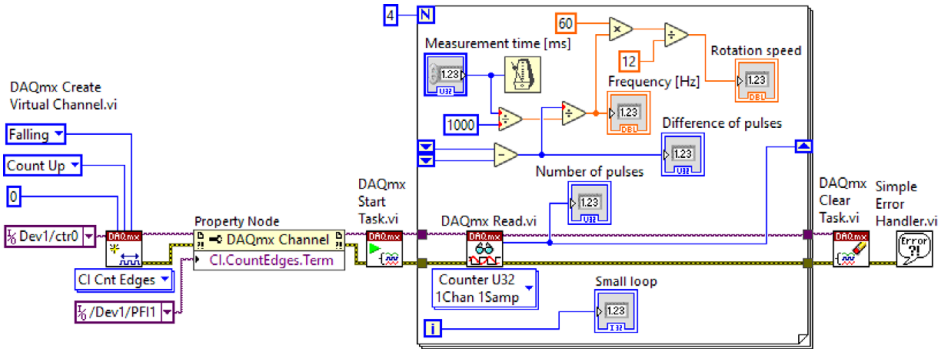

**Ryc. 4.16.** Diagram trzeciego programu z ćwiczenia 3 modyfikowany w ćwiczeniu 4

Przykładowy diagram programu z ćwiczenia 3, modyfikowanego na potrzeby ćwiczenia 4, przedstawiono na rycinie 4.16. W pierwszej kolejności kasujemy z diagramu przekreślone na czerwono, zbędne elementy oraz odpowiadające im kontrolki na panelu. Na diagramie zostaną zaznaczone liczne błędne połączenia, ale na razie nie kasujemy ich. Łączymy niebieską linię **data** z wyjścia funkcji **Read** do wyjściowego rejestru przesuwnego. Zamieniamy pętlę **While** na pętlę **For**: prawym kliknięciem na krawędzi pętli **While** otwieramy menu i wybieramy opcję **Replace with For Loop**. Na wejściu zacisku N pętli **For** tworzymy stałą (**Create > Constant**) i nadajemy jej wartość 4. Teraz usuwamy błędne połączenia na diagramie poleceniem **Edit > Remove Broken Wires** lub kombinacją klawiszy **Ctrl+B**. Uruchamiamy program i sprawdzamy wynik pomiaru częstotliwości. Otrzymany program wykonuje jeden pomiar częstotliwości,

pomijając kilka początkowych odczytów z licznika, które są błędne. Zagadnienie to jest wyjaśnione w ćwiczeniu 3 na rycinie 3.18.

W ćwiczeniu 3 program ten mierzył częstotliwość impulsów z generatora wzorcowego dołączonego do wejścia PFI 0 licznika. Wejście PFI 0 jest wejściem domyślnie używanym przez licznik w module NI USB-6001, dlatego w ćwiczeniu 3 nie było konieczności jawnego konfigurowania wejścia licznika. Jeśli teraz program również zmierzył poprawnie częstotliwość ustawioną na generatorze wzorcowym zgodnie z ryciną 3.2, zapisujemy go na dysku oraz robimy jego kopię z numerem 4-3-b.

W ćwiczeniu 4 wyjście z transoptora szczelinowego dołączone jest do wejścia PFI 1 licznika, dzięki czemu nie musimy robić w układzie pomiarowym robić żadnych zmian w połączeniach elektrycznych, musimy jednak odpowiednio programowo skonfigurować wejście licznika. Do tego celu w LabVIEW stosujemy odpowiedni węzeł właściwości **Property Node**, który wstawiamy pomiędzy funkcje **Create Virtual Channel** i **Start Task**, w miejscu zaznaczonym zielonym prostokątem na rycinie 4.16. Diagram po odpowiednich uzupełnieniach i modyfikacjach pokazano na rycinie 4.17.




**Ryc. 4.17.** Zmodyfikowany diagram trzeciego programu z ćwiczenia 3 do pomiaru prędkości obrotowej w ćwiczeniu 4

Kolejność postępowania jest następująca: kasujemy połączenia **task** i **error** pomiędzy funkcjami **Create Virtual Channel** i **Start Task** oraz odsuwamy je od siebie, tworząc miejsce na węzeł właściwości **Property Node**. Z zasobnika funkcji **Measurement I/O > NI DAQmx** pobieramy obiekt **Channel Node** i stawiamy go na diagramie. Prawym kliknięciem na obiekcie **Property Node** otwieramy jego menu i konfigurujemy ustawienia filtra

**Select Filter... > Show All Attributes**. Następnie sprawdzamy i ewentualnie korygujemy klasę węzła: **Select Class > DAQmx > DAQmx Channel**. Ponownie otwieramy menu prawym kliknięciem na **Property** w dolnej części węzła i wybieramy odpowiednią właściwość węzła: **Select Property > Counter Input > Count Edges > Input > Terminal**. Ponownie otwieramy menu prawym kliknięciem na utworzonym terminalu **CI.CountEdges.Term** w dolnej części węzła i wybieramy kierunek terminala do zapisu: **ChangeTo > Write**. Prawym kliknięciem na lewym zacisku tego terminala tworzymy stałą (**Create > Constant**) i z rozwijanego menu $^{I}/_{O}$ wybieramy wejście **Dev1/PFI1**. Uruchamiamy program i sprawdzamy jego działanie. Program wykonuje 4 razy pętlę For i dopiero ostatni wynik po zakończeniu programu jest poprawny. Dla zatrzymanego silnika wynik powinien wynosić zero. Uruchamiamy pierwszy program z ćwiczenia 4 i zadajemy napięcie 5 V na wyjściu analogowym przetwornika DAC, co powinno spowodować uruchomienie silnika. Uruchamiamy tworzony teraz program, który powinien zmierzyć częstotliwość około 400 Hz. Jeśli program zmierzył oczekiwaną wartość częstotliwości, zapisujemy go na dysku oraz robimy jego kopię z numerem 4-3-c.

W pierwszym programie z ćwiczenia 4 zadajemy napięcie 0 V i kończymy jego pracę, co powinno spowodować zatrzymanie silnika.

W kolejnym kroku uzupełniamy diagram o przeliczanie częstotliwości na prędkość obrotową zgodnie ze wzorem:

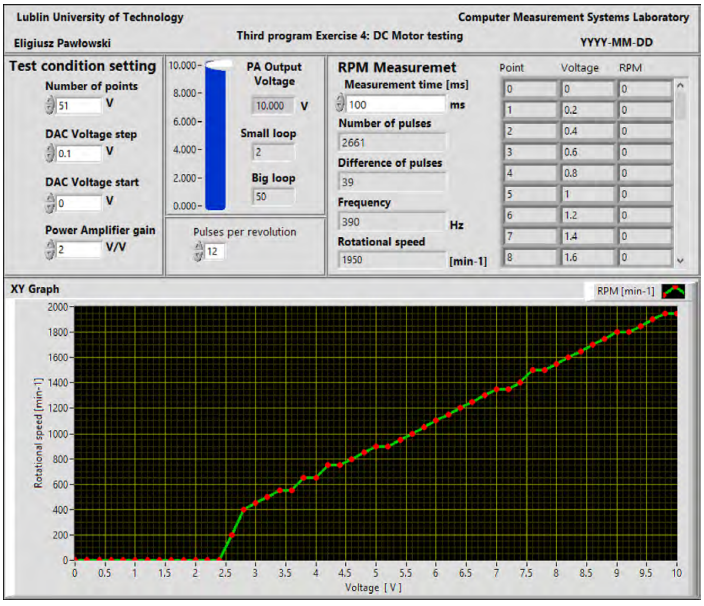
$$n_x = \frac{60 f_x}{z}, \tag{4.1}$$

gdzie: $n_x$ – prędkość obrotowa w obrotach na minutę, $f_x$ – częstotliwość impulsów w hercach na wyjściu transoptora szczelinowego, $z$ – liczba otworów w tarczy, 60 – przelicznik sekund na minuty [39, 40]. Pobieramy operatory mnożenia i dzielenia z zasobnika **Functions > Programming > Numeric**. Na wejściu operatora mnożenia prawym przyciskiem myszy tworzymy stałą (**Create > Constant**) o wartości 60. Podobnie na wejściu operatora dzielenia tworzymy stałą równą liczbie otworów w tarczy transoptora szczelinowego. Na wyjściu operatora dzielenia prawym przyciskiem myszy tworzymy wskaźnik numeryczny (**Create > Indicator**), który będzie pokazywał wynik pomiaru prędkości obrotowej **Rotational Speed**. Na wyjściu zacisku i pętli

**For** tworzymy wskaźnik (**Create > Indicator**) i nadajemy mu nazwę **Small Loop**, ponieważ tę pętlę **For** będziemy teraz nazywać małą pętlą.

Uruchamiamy program i sprawdzamy jego działanie. Dla zatrzymanego silnika wynik powinien wynosić zero. Uruchamiamy pierwszy program z ćwiczenia 4 i zadajemy napięcie 5 V na wyjściu analogowym przetwornika DAC. Program powinien zmierzyć prędkość obrotową około 2000 rpm (**revolutions per minute**). Jeśli program zmierzył oczekiwaną wartość prędkości obrotowej, zapisujemy go na dysku oraz robimy jego kopię z numerem 4-3-d. Notujemy wyniki testowania programu i wnioski w protokole.

Teraz przystępujemy do tworzenia końcowej wersji programu do badania silnika. Program ten powinien wysyłać w pętli **For** narastające wartości napięcia do przetwornika cyfrowo-analogowego DAC i jednocześnie mierzyć częstotliwość impulsów z przetwornika obrotowo-impulsowego oraz przeliczać ją na prędkość obrotową. W tym celu łączymy w jedną całość dwa programy: program o numerze 4-3 (jest to kopia ostatniej wersji programu zrealizowanego w punkcie 4.4.2) oraz zapisany przed chwilą program 4-3-d. Przykładowa postać panelu końcowej wersji programu pokazana jest na rycinie 4.18, a jego diagram na rycinie 4.19.




**Ryc. 4.18.** Przykładowy panel ostatecznej wersji trzeciego programu do automatycznego badania silnika DC małej mocy

Kolejność postępowania jest następująca: otwieramy zapisaną kopię programu 4-3 i powiększamy do góry pętlę **For** tak, aby utworzyć miejsce na pętlę z programu 4-3-d. Usuwamy z pętli opóźnienie czasowe **Wait Until Next ms Multiple** i powiązany z nim zadajnik numeryczny **Loop delay time [ms]**. Kasujemy wykres **Waveform Chart**. Dodajemy na panelu zadajnik numeryczny **DAC Voltage Start** i modyfikujemy odpowiednio obliczanie napięcia zadawanego do przetwornika DAC. Obliczamy napięcie na wyjściu wzmacniacza mocy **PA Output Voltage**, uwzględniając wzmocnienie ustawiane na panelu zadajnikiem **Power Amplifier Gain**. Nazwę wskaźnika **Loop Number** zamieniamy na **Big Loop**, ponieważ tę pętlę **For** będziemy teraz nazywać dużą pętlą.

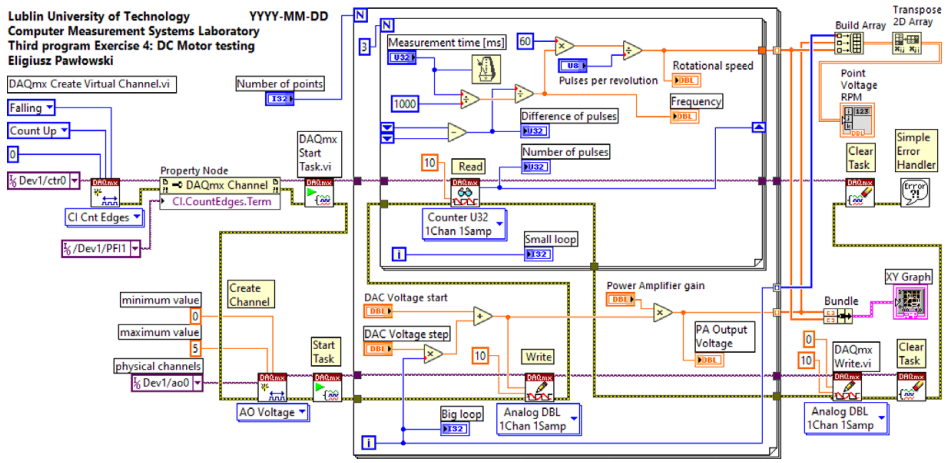
W programie 4-3-d kasujemy połączenia **task** i **error** na wejściu i wyjściu pętli **For**. Tak oddzieloną od reszty diagramu małą pętlę **For** z programu 4-3-d kopiujemy do wnętrza dużej pętli **For** w programie 4-3. Na wejściu zacisku  małej pętli **For** zamieniamy liczbę 4 na 3. Kopiujemy z programu 4-3-d pozostałe funkcje **Create Virtual Channel, Start Task, Clear Task** oraz **Simple Error Handler** do programu 4-3, odpowiednio po lewej i prawej stronie dużej pętli **For**. Łączymy ponownie linię **task**, tak jak była pierwotnie połączona w programie 4-3-d. Modyfikujemy przebieg linii **error**, która ustala kolejność realizowanych w programie działań. Zwróćmy uwagę, że linie **task** dwóch zadań przechodzą równolegle od lewej do prawej strony diagramu, natomiast jedna linia **error** meandruje po całym diagramie, przechodząc kolejno przez wszystkie funkcje obu zadań. Kasujemy zbędny dolny obiekt **Simple Error Handler**. Wszystkie przejścia linii **task** i **error** przez krawędzie obu pętli **For** powinny być oznaczone jako tunele **Tunnel** w trybie **Last Value**. Sprawdzamy to prawym kliknięciem i ewentualnie odpowiednio modyfikujemy w opcji **Tunnel Mode > Last Value**.

Kolejnym etapem będzie utworzenie wykresu **XY Graph** oraz dwuwymiarowej tablicy **Array** z wynikami pomiarów.

Wykres tworzymy za pomocą funkcji **Programming > Cluster, Class, & Variant > Bundle**, którą umieszczamy na diagramie na zewnątrz dużej pętli **For**, po jej prawej stronie. Na panelu umieszczamy obiekt graficzny do rysowania wykresu **Graph > XY Graph**, odszukujemy jego ikonę na diagramie i umieszczamy po prawej stronie funkcji **Bundle**. Do górnego zacisku **Bundle** dołączamy wartość napięcia na wyjściu wzmacniacza mocy **PA**

**Output Voltage**, która będzie stanowiła oś X na wykresie. Do dolnego zacisku **Bundle** dołączamy wynik pomiaru prędkości obrotowej **Rotational Speed**, który będzie stanowił oś Y na wykresie. W tunelu wyjściowym z małej pętli **For** wyłączamy indeksowanie (**Tunnel Mode > Last Value**), tak aby do dalszych obliczeń został przekazany tylko ostatni wynik pomiaru prędkości obrotowej. W odpowiednich tunelach wyjściowych dużej pętli **For** włączamy indeksowanie (**Tunnel Mode > Indexing**), co spowoduje utworzenie tablic z wartościami numeracji dużej pętli **For**, napięcia na wyjściu wzmacniacza mocy i zmierzonej prędkości obrotowej. Tablicę z wynikami pomiarów tworzymy za pomocą funkcji **Programming >Array > Build Array,** którą stawiamy na diagramie po prawej stronie dużej pętli **For** i rozciągamy w dół, aby otrzymać trzy zaciski wejściowe. Do górnego zacisku łączymy numerację dużej pętli **Big Loop**, a do dwóch kolejnych zacisków dołączamy te same sygnały co w funkcji **Bundle**. Otrzymaną na wyjściu funkcji **Build Array** tablicę transponujemy za pomocą funkcji **Programming > Array > Transpose 2D Array.**

Na jej wyjściu prawym przyciskiem myszy tworzymy wskaźnik tablicowy **Create > Indicator**. Prawym przyciskiem myszy otwieramy menu tego wskaźnika i konfigurujemy **Properties > Appearance > Show vertical scroll bar** oraz **Size > 2 Dimension(s)**.




**Ryc. 4.19.** Przykładowy diagram ostatecznej wersji trzeciego programu do automatycznego badania silnika DC małej mocy

Zamieniamy stałą numeryczną z liczbą otworów w tarczy na zadajnik (**Change to Control**) i przypisujemy mu reprezentację bajtu bez znaku (**Representation > Unsigned Byte U8**). Porządkujemy panel i diagram oraz uzupełniamy je niezbędnymi informacjami, jak w poprzednich punktach. Aby uzyskać bardziej kompaktową postać diagramu, odznaczamy prawym przyciskiem myszy na wszystkich kontrolkach i wskaźnikach opcję **View As Icon**. Nadajemy początkowe wartości parametrów w programie według ryciny 4.18.

Uruchamiamy program przyciskiem  i sprawdzamy, czy działa poprawnie. Program powinien wykreślić charakterystykę badanego silnika w 51 punktach dla napięć od zera do 10 V. Prędkość obrotowa powinna zmieniać się od zera do około 2000 obrotów na minutę, zależnie od egzemplarza badanego silnika. Wszystkie punkty pomiarowe z wykresu powinny być dostępne do odczytania w przewijanej dwuwymiarowej tablicy. Notujemy wyniki testowania programu i wnioski w protokole.

Jeśli program działa prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Zapisujemy trzeci program na dysku. Notujemy nazwę pliku i katalogu, a następnie używamy jej w raporcie. Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Zapisz nazwę katalogu i nazwy plików z opisem zawartości do protokołu. Używamy zapisanych plików do sporządzenia raportu.

## 4.6. Badania silnika DC małej mocy

Korzystając z opracowanego programu, przeprowadzamy badania silnika DC małej mocy dla różnych konfiguracji. Celem tych eksperymentów jest ustalenie optymalnych warunków badania silnika. Pomiary przeprowadzamy dla kilku sposobów sterowania przetwornikiem DAC oraz dla różnych czasów pomiarów w układzie licznika impulsów. Zazwyczaj zależy nam na wykonaniu badań szybko i dokładnie. Zazwyczaj jednak są to przeciwstawne oczekiwania. Pomiary możemy wykonać albo dokładnie w długim czasie, albo szybko, ale mało dokładnie. W przypadku badania silnika możemy zadawać dużo kroków

napięcia o małej wartości, co zwiększa dokładność badań, ale wydłuża czas ich trwania. Możemy również zwiększyć dokładność, wydłużając czas pomiaru częstotliwości, co jednak także wydłuża całkowity czas badań. Chcąc wykonać badania możliwie dokładnie przy ograniczonym czasie, ustalamy, czy lepiej przeprowadzić dużo pomiarów z małym krokiem napięcia i krótkim czasem pomiaru częstotliwości, czy może wydłużyć czas pomiaru częstotliwości i wykonać mniej pomiarów z większym krokiem napięcia [41]? W tym celu wykonujemy pomiary według tabeli 4.6. Wszystkie pomiary wykonujemy w zakresie napięć od 0 do +5 V zadawanych do przetwornika DAC, co oznacza, że po dwukrotnym wzmocnieniu we wzmacniaczu mocy silnik zasilany jest napięciem od 0 do +10 V. Liczbę punktów pomiarowych zmieniamy w zakresie od 11 do 501, krok napięcia od 500 mV do 10 mV, a czas pomiaru od 10 ms do 1000 ms. Łączny czas pojedynczego badania zmienia się od ułamka sekundy do kilkunastu minut.

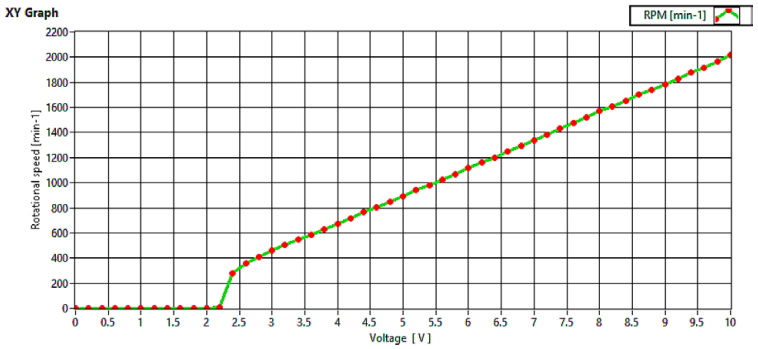
**Tab. 4.6.** Wyniki badań silnika DC małej mocy

| Liczba punktów | Krok napięcia | Czas pomiaru | $U_{STARTt}$ | $RPM_{MAX}$ | Liczba impulsów | Czas badania | Nazwa pliku z wykresem |
|---|---|---|---|---|---|---|---|
| – | **mV** | **ms** | **V** | **$min^{-1}$** | – | **s** | – |
| 11 | 500 | 10 | – | – | – | – | – |
| 11 | 500 | 100 | – | – | – | – | – |
| 11 | 500 | 1000 | – | – | – | – | – |
| 51 | 100 | 10 | – | – | – | – | – |
| 51 | 100 | 100 | – | – | – | – | – |
| 51 | 100 | 1000 | – | – | – | – | – |
| 101 | 50 | 10 | – | – | – | – | – |
| 101 | 50 | 100 | – | – | – | – | – |
| 101 | 50 | 1000 | – | – | – | – | – |
| 501 | 10 | 10 | – | – | – | – | – |
| 501 | 10 | 100 | – | – | – | – | – |

Wykonujemy tyle pomiarów, na ile pozwoli czas trwania zajęć w laboratorium. Pomiary możemy wykonać w innej kolejności niż podana w tabeli 4.6, szczególnie jeśli jest zbyt mało czasu na wykonanie wszystkich badań. Każdy uzyskany wykres zapisujemy do pliku graficznego BMP opcją **Export Simplified Image**, nazwę pliku notujemy w tabeli 4.6. Przeglądając na panelu

programu tabelę z wynikami pomiarów, notujemy również w tabeli 4.6 wartość napięcia $U_{START,}$ przy którym silnik zaczął się obracać, maksymalną prędkość obrotową *RPMMAX* oraz całkowitą liczbę impulsów zliczoną w liczniku, odczytaną na panelu programu. Mierzymy również stoperem lub szacujemy czas trwania każdego badania.

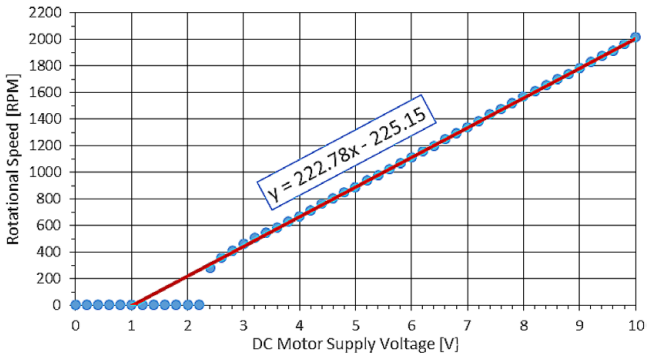
Przeglądając wyniki zapisane w tabeli 4.6 oraz porównując uzyskane wykresy, wybieramy optymalne warunki przeprowadzenia testu zapewniające uzyskanie w możliwie krótkim czasie wystarczająco dokładnej charakterystyki silnika, tzn. nie gorszej od charakterystyk uzyskanych w dłuższym badaniu. Notujemy te parametry w protokole, powtarzamy badanie z tymi parametrami i zapisujemy uzyskany wykres do pliku graficznego BMP. Przykładowy wykres otrzymany opracowanym programem pokazano na rycinie 4.20 (niekoniecznie są to optymalne warunki badania). Dodatkowo eksportujemy wyniki liczbowe z wykresu do arkusza kalkulacyjnego opcją **Export Data To Excel**. Jeśli w komputerze nie jest zainstalowany program Excel, to opcja ta nie będzie aktywna. W takiej sytuacji dane eksportujemy do schowka, następnie wklejamy do pustego pliku w Notatniku i zapisujemy do pliku tekstowego z rozszerzeniem .txt lub .csv.




**Ryc. 4.20.** Przykładowa charakterystyka silnika DC małej mocy w przedziale napięć 0–10 V, wygenerowana programem opracowanym w aplikacji LabVIEW

W raporcie dodatkowo wykonujemy wykres w arkuszu kalkulacyjnym, korzystając z zapisanych w pliku danych liczbowych. Aproksymujemy otrzymany wykres linią prostą w jego prawej, prostoliniowej części. Odczytujemy z wykresu nachylenie charakterystyki w obrotach na minutę na wolt oraz przecięcie z osią X w woltach. Przykładowy wykres odpowiednio opracowany

w programie Excel pokazano na rycinie 4.21. Zwracamy uwagę, że aproksymowana charakterystyka silnika nie przechodzi przez początek układu współrzędnych. Silnik ma również pewną minimalną prędkość obrotową, poniżej której nie jest w stanie się obracać. Staramy się w raporcie samodzielnie wyjaśnić kształt otrzymanej charakterystyki oraz jej związek z budową i zasadą działania komutatorowego silnika prądu stałego ze wzbudzeniem magnesami trwałymi.




**Ryc. 4.21.** Charakterystyka silnika DC małej mocy z rysunklu 4.20 opracowana w programie Excel, prostoliniowa część charakterystyki aproksymowana metodą najmniejszych kwadratów

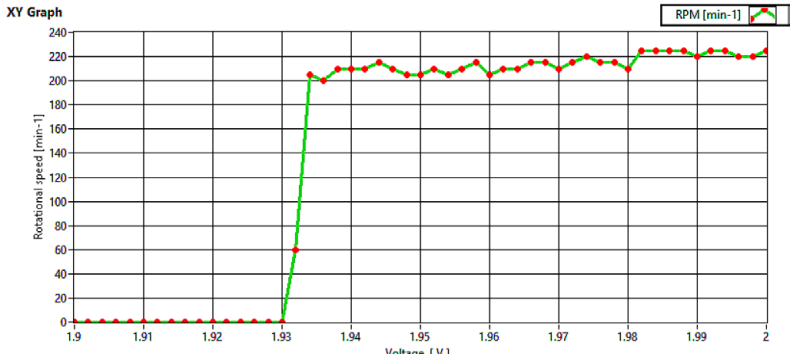
Badamy dokładniej rozruch silnika: zadajemy na panelu napięcie początkowe **DAC Voltage Start** nieznacznie mniejsze od zaobserwowanego w tabeli 4.6 napięcia $U_{START}$. Ustawiamy odpowiednio mały krok napięcia i odpowiednią liczbę kroków. Podobnie badamy zatrzymanie silnika: zadajemy odpowiednio większe napięcie początkowe i ujemną wartość kroku napięcia. Zapisujemy otrzymane charakterystyki do pliku graficznego BMP. Przykładowy wykres dla rozruchu silnika pokazano na rycinie 4.22, a dla zatrzymania silnika na rycinie 4.23.




**Ryc. 4.22.** Przykładowa charakterystyka rozruchu badanego silnika DC małej mocy wygenerowana w aplikacji LabVIEW, zwróć uwagę na różnicę w stosunku do rysunku 4.23

Zwracamy uwagę, że napięcie rozruchu i zatrzymania różnią się o kilkaset miliwoltów. Staramy się wyjaśnić w raporcie ten efekt. Sprawdzamy w protokole, jakie napięcie rozruchu silnika zanotowaliśmy podczas testowania wyjścia analogowego w programie NI MAX na początku ćwiczenia. Analizujemy również, jakie napięcie rozruchu silnika zanotowaliśmy podczas testowania pierwszego programu do ręcznego sterowania przetwornikiem DAC, i sprawdzamy, czy są to wartości zbliżone do siebie. Zestawiamy je w raporcie w samodzielnie zaprojektowanej tabeli.

Powtarzamy kilka razy eksperyment z rozruchem i zatrzymaniem silnika. Sprawdzamy, czy zawsze otrzymujemy te same wartości napięcia rozruchu i zatrzymania. Staramy się wyjaśnić zaobserwowany efekt w sprawozdaniu. Pamiętamy, że badany jest silnik komutatorowy z magnesami trwałymi.




**Ryc. 4.23.** Przykładowa charakterystyka zatrzymania badanego silnika DC małej mocy wygenerowana w aplikacji LabVIEW, zwróć uwagę na różnicę w stosunku do rysunku 4.22

## 4.7. Opracowanie raportu z realizacji ćwiczenia

Raport z przeprowadzonego ćwiczenia powinien zawierać:

- stronę tytułową z danymi osobowymi zespołu, datą ćwiczenia i jego tytułem,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań zgodnie z programem ćwiczenia,
- opis każdego z wykonanych zadań lub podanie przyczyny niewykonania,
- dokumentację zrealizowanych i uruchomionych trzech programów,
- wykresy wygenerowane przez program w środowisku LabVIEW,

- wykres opracowany w arkuszu kalkulacyjnym Excel,
- liczbowe wyniki pomiarów uzyskane z opracowanych programów,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.

W raporcie ocenie podlegają:

- poprawność działania programów,
- czytelność i przejrzystość diagramu,
- estetyka panelu, dobór kolorów i elementów graficznych,
- funkcjonalność panelu pod kątem wygody obsługi,
- zakres zrealizowanych zadań,
- trafność wniosków końcowych.

# 5. Badanie wzmacniacza prądu stałego

## 5.1. Cel ćwiczenia i zadania do wykonania

Ćwiczenie 5 jest kontynuacją ćwiczenia 4. Drugi program zrealizowany w ćwiczeniu 4, służący do sterowania przetwornikiem DAC w pętli **While**, stanowi podstawę do rozpoczęcia ćwiczenia 5. Jednocześnie ćwiczenie 5 jest wstępem do ćwiczenia 6 i 8. Ćwiczenia 5, 6 i 8 wykorzystują ten sam układ pomiarowy, ale realizujemy w nich inne zadania. Ostatni program zrealizowany w ćwiczeniu 5 jest punktem wyjścia do realizacji ćwiczenia 6.

Celem ćwiczenia 5 jest zapoznanie się z problematyką programowania systemu pomiarowego w środowisku LabVIEW z jednoczesnym wykorzystaniem przetwornika cyfrowo-analogowego i analogowo-cyfrowego do wyznaczenia charakterystyki przetwarzania wzmacniacza prądu stałego.

Program laboratorium obejmuje następujące zagadnienia:

- konfigurowanie modułu NI USB-6001 do jednoczesnego sterowania przetwornikiem DAC i pomiaru napięcia przetwornikiem ADC,
- wyznaczanie charakterystyki przetwarzania wzmacniacza prądu stałego,
- aproksymacja charakterystyki linią prostą i wyznaczanie błędów przetwarzania,
- aproksymacja wykresu błędów wielomianem i wyznaczanie błędów liniowości.

W ramach ćwiczenia 5 opracowujemy trzy programy:

1. Pierwszy program do wyznaczania charakterystyki wzmacniacza mocy DC.
2. Drugi program do aproksymacji charakterystyki przetwarzania linią prostą.
3. Trzeci program do aproksymacji błędów liniowości parabolą.

## 5.2. Opis stanowiska laboratoryjnego

### 5.2.1. Schemat blokowy stanowiska laboratoryjnego

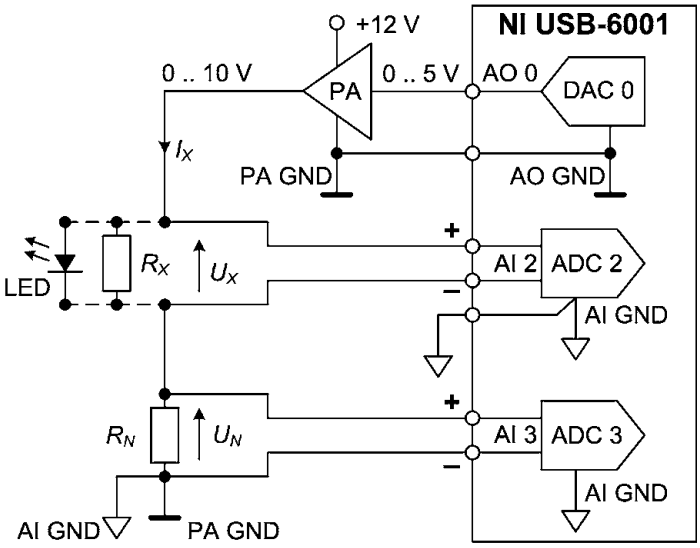
Stanowisko laboratoryjne wykorzystywane w ćwiczeniach 5, 6 i 8 składa się z następujących komponentów:

- komputera PC z zainstalowanym środowiskiem LabVIEW,
- modułu akwizycji danych pomiarowych NI USB-6001,
- wzmacniacza mocy prądu stałego do zasilania układu pomiarowego,
- modułu z zestawem zacisków laboratoryjnych do łączenia badanych dwójników,
- zestawu pasywnych dwójników liniowych i nieliniowych do badań,
- zestawu rezystorów wzorcowych.

Schemat blokowy stanowiska laboratoryjnego wykorzystywanego w ćwiczeniach 5, 6 oraz 8 przedstawiono na rycinie 5.1 [42]. Moduł akwizycji danych pomiarowych NI USB-6001 dołączamy do komputera PC za pomocą interfejsu USB. Napięcie wyjściowe z przetwornika cyfrowo-analogowego AO 0 podajemy na wejście wzmacniacza mocy PA (**Power Amplifier**) o wzmocnieniu równym 2 V/V. Wzmacniacz ma wbudowane w układ scalony zabezpieczenie przeciwzwarciowe o wartości około 2 A i zasilamy go napięciem +12 V z zasilacza impulsowego o wydajności 3,3 A. Napięcie wyjściowe ze wzmacniacza zasila układ pomiarowy składający się z szeregowego połączenia badanego dwójnika $R_X$ i rezystora wzorcowego $R_N$.

Do pomiarów wykorzystujemy dwa wejścia analogowe: AI 2 oraz AI 3, pracujące w trybie różnicowym. Napięcie $U_X$ na dwójniku $R_X$, który będziemy badać w ćwiczeniu 6, mierzymy w kanale AI 2, a napięcie *UN* na rezystorze wzorcowym mierzymy w kanale AI 3. Zadając na przetworniku DAC napięcia w zakresie od 0 do 5 V, na wyjściu wzmacniacza otrzymujemy napięcia od około 0 do 10 V. Wzmacniacz nie jest typu **rail-to-rail** [34] i pracuje poprawnie powyżej napięcia

około 400 mV na wejściu. Zbadamy to w ramach programu ćwiczenia 5. Dobierając odpowiednio rezystor wzorcowy $R_N$ z przedziału od 1 Ω do 1 kΩ, możemy na stanowisku badać dwójniki w zakresie napięć do około 8 V i prądu do 2 A, tak aby spadek napięcia na rezystorze wzorcowym wynosił przynajmniej 2 V na końcu zakresu pomiarowego. Badania takie będziemy realizować w ćwiczeniu 6. W ćwiczeniu 5 natomiast w miejsce badanego dwójnika *RX* będziemy włączać czerwoną diodę świecącą LED, ale tylko podczas wstępnego testowania stanowiska. Podczas wyznaczania charakterystyki przetwarzania wzmacniacza PA dwójnik $R_X$ (dioda LED) będzie odłączony, czyli prąd $I_X$ będzie równy zero. W takiej konfiguracji wejściem AI 2 będziemy mierzyć napięcie wyjściowe na nieobciążonym wzmacniaczu PA. Spadek napięcia na rezystorze wzorcowym $R_N$ wyniesie zero. Dlatego w ćwiczeniu 5 w ogóle nie będziemy mierzyć napięcia na wejściu AI 3.




**Ryc. 5.1.** Schemat blokowy stanowiska laboratoryjnego w ćwiczeniu 5 [42]

### 5.2.2. Parametry przetworników DAC i ADC w module NI USB-6001

Budowę torów analogowego wejścia i wyjścia modułu NI USB-6001 opisano już we wcześniejszych ćwiczeniach. Jednak dopiero w tym ćwiczeniu będą one pracować jednocześnie w jednej aplikacji. Dlatego zwracamy uwagę na parametry obu torów jednocześnie i porównujemy je ze sobą. W tabeli 5.1 zestawiono wybrane parametry przetworników ADC i DAC zastosowanych w module NI USB-6001 [6].

**Tab. 5.1.** Porównanie parametrów przetworników DAC i ADC modułu NI USB-6001 [6]

| Przetwornik | DAC | ADC |
|---|---|---|
| Liczba kanałów | 2 niezależne | 4 lub 8 multipleksowane |
| Rozdzielczość | 14 bitów | 14 bitów |
| Znamionowy zakres napięć | −10 V do +10 V | −10 V do +10 V |
| Rzeczywisty zakres napięci | −10 V do +10 V | −10,5 V do +10,5 V |
| Krok kwantowania | 1,22 mV | 1,28 mV |
| Maksymalna szybkość przetwarzania | 5000 SPS symultanicznie | 20 kSPS agregowane |
| Błąd typowy na końcu zakresu | 9,1mV | 6 mV |
| Błąd maksymalny w zakresie temperatur | 34mV | 26 mV |
| Rezystancja wyjściowa/wejściowa | 0,2 Ω | > 1 GΩ |
| Maksymalny prąd wyjściowy/ wejściowy | ±5 mA | ±200 pA |

Zauważamy, że w obu przetwornikach błędy są kilka razy większe od kroku kwantowania, co wynika z niedoskonałości ich części analogowej. Oba przetworniki mają taką samą rozdzielczość 14 bitów oraz takie same zakresy pomiarowe ±10 V, ale krok kwantowania przetwornika ADC jest większy i wynosi 1,28 mV. Wynika to z faktu, że rzeczywisty zakres przetwarzanych napięć w przetworniku ADC jest nieznacznie większy od zakresu znamionowego i wynosi ±10,5 V [5]. Jest to typowe podejście producentów cyfrowej aparatury pomiarowej. Podobnie większość multimetrów cyfrowych może mierzyć wartości większe od zakresu, nawet o 20 %. Przetwornik DAC nie może natomiast wygenerować na wyjściu napięcia większego od 10 V, dlatego jego krok kwantowania jest mniejszy i wynosi 1,22 mV. Zauważymy to podczas wyznaczania charakterystyki przetwarzania wzmacniacza, dla którego błędy nieliniowości okazują się porównywalne z krokiem kwantowania przetworników ADC i DAC.

Porównujemy również szybkość przetwarzania obu przetworników. Przetwornik ADC jest znacznie szybszy od przetwornika DAC. W pewnych sytuacjach przetwornik ADC może wykonać pomiar, zanim jeszcze przetwornik DAC ustawi nową wartość napięcia na wyjściu. Dlatego w odpowiednim miejscu diagramu wstawiamy niewielkie opóźnienie czasowe na ustabilizowanie się nowych wartości napięć w układzie pomiarowym.

Z drugiej jednak strony zauważamy, że dla przetwornika ADC podano agregowaną szybkość próbkowania, czyli dla jednoczesnej pracy wszystkich

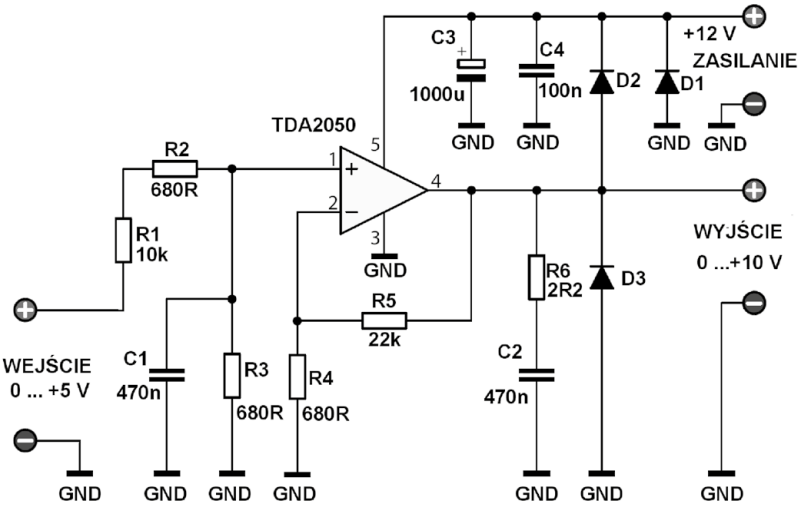
lub kilku kanałów. Agregowana szybkość próbkowania przetwornika ADC dzieli się więc pomiędzy wszystkie pracujące jednocześnie wejściowe kanały analogowe. Szybkość przetwarzania DAC jest natomiast taka sama (symultaniczna) dla każdego wyjściowego kanału analogowego, niezależnie od tego, czy pracuje tylko jeden z nich, czy dwa jednocześnie.

Maksymalny prąd wyjściowy ±5 mA przetwornika DAC jest bardzo mały, co ogranicza jego możliwe zastosowania. Dlatego stosujemy dodatkowy wzmacniacz mocy. Natomiast prąd polaryzujący wejścia przetwornika ADC wynoszący ±200 pA uznajemy za bardzo mały, dzięki czemu możemy bezpośrednio mierzyć napięcia w układzie pomiarowym bez konieczności stosowania dodatkowych wzmacniaczy buforujących na wejściu.

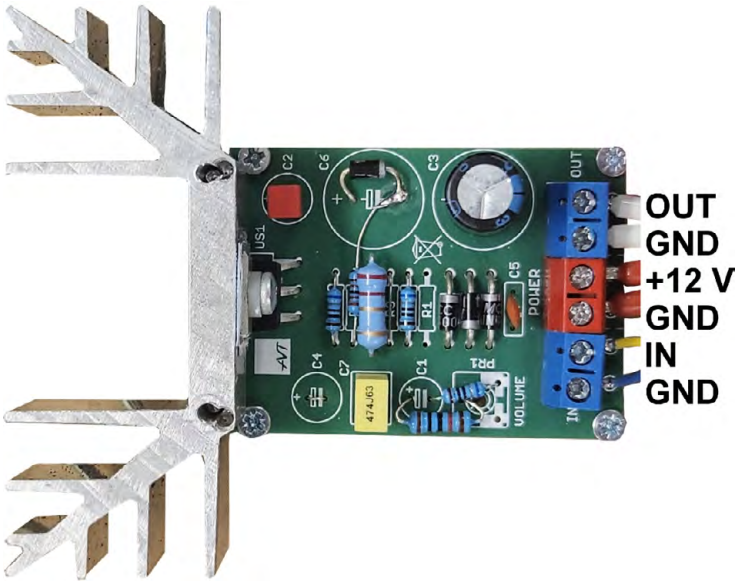
### 5.2.3. Budowa i parametry modułu wzmacniacza mocy

Ponieważ wydajność prądowa wyjścia analogowego modułu NI USB-6001 wynosi tylko ±5 mA, stosujemy dodatkowy wzmacniacz dużej mocy, który będzie w stanie dostarczyć do układu pomiarowego większy prąd. Na panelu ćwiczeniowym zastosowano wzmacniacz zbudowany na bazie układu TDA2050 produkcji STMicroelectronics [34]. Jest to wzmacniacz akustyczny o mocy chwilowej do 50 W przy zasilaniu symetrycznym 22,5 V na obciążeniu 4 Ω. Może również pracować z mniejszą mocą przy zasilaniu niesymetrycznym 12 V [35]. Wzmacniacz ma zabezpieczenie przeciwzwarciowe oraz termiczne. Może on dostarczyć do obciążenia prąd o wartości maksymalnej nawet 2 $A_{RMS.}$ Na panelu ćwiczeniowym zastosowano gotowy moduł wzmacniacza dostępny w postaci zestawu do samodzielnego montażu (tzw. kitu) [36] znanego polskiego producenta AVT. Na potrzeby ćwiczenia odpowiednio go jednak zmodyfikowano, tak aby wzmacniał napięcia stałe DC. W oryginalnych aplikacjach akustycznych wzmacniacz pracuje w paśmie od 20 Hz do 20 kHz, a więc nie wzmacnia napięć stałych, ponieważ zgodnie ze schematem aplikacyjnym [34] na wejściu ma włączony kondensator blokujący sygnały DC. W aplikacjach akustycznych zalecana jest praca tego wzmacniacza ze wzmocnieniem 30,5 dB, czyli 33,5 V/V. Natomiast zalecane minimalne wzmocnienie wynosi 24 dB, czyli 15,8 V/V [34]. Do pomiarów w ramach ćwiczenia zaplanowano wzmocnienie 2 V/V, aby zapewnić możliwość współpracy ze starszymi modelami modułów NI USB-6008/6009, które mają zakres wyjścia analogowego tylko do +5 V. Ponieważ jednak wzmacniacz TDA2050 nie może pracować przy tak

niskim wzmocnieniu ze względu na ryzyko wzbudzenia się, zastosowano na jego wejściu dzielnik napięcia.




**Ryc. 5.2.** Schemat ideowy wzmacniacza mocy zastosowanego na panelu ćwiczeniowym, zmodyfikowany kit AVT 1597 [36]




**Ryc. 5.3.** Moduł wzmacniacza mocy zastosowany na panelu ćwiczeniowym

Zmodyfikowany układ wzmacniacza pokazano na rycinie 5.2. Rezystory sprzężenia zwrotnego R4, R5 ustalają wzmocnienie układu TDA2050 na zalecaną wartość 33,5 V/V (30,5 dB). Rezystory dzielnika wejściowego R1, R2, R3 zapewniają tłumienie 16,7 razy, a więc łączne wzmocnienie całego układu wynosi 2 V/V. Z oryginalnego schematu usunięto kondensatory kształtujące charakterystykę częstotliwościową odpowiednią do zastosowań akustycznych. Pozostałe elementy są zgodne z zalecaną specyfikacją producenta [34].

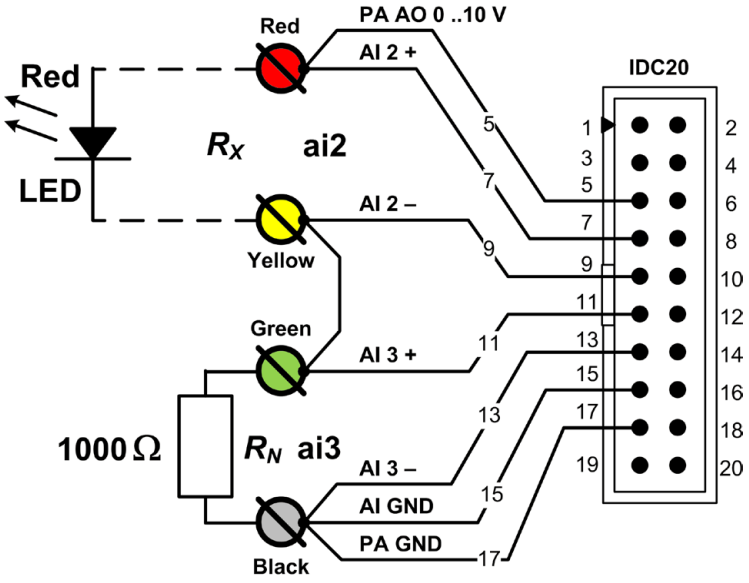
Na rycinie 5.3 pokazano rzeczywisty wygląd modułu wzmacniacza z opisanym złączem sygnałowym i zasilającym. Układ TDA2050 przykręcono do aluminiowego radiatora zapewniającego prawidłowe chłodzenie. Zasilanie oraz sygnały wejściowe i wyjściowe są dołączoneza pomocą złącza śrubowego typu ARK. Moduł wzmacniacza jest na stałe zamocowany na panelu ćwiczeniowym i połączony z pozostałą częścią układu, dlatego podczas realizacji ćwiczenia nie musimy wykonywać żadnych dodatkowych połączeń.

### 5.2.4. Budowa modułu do badania dwójników

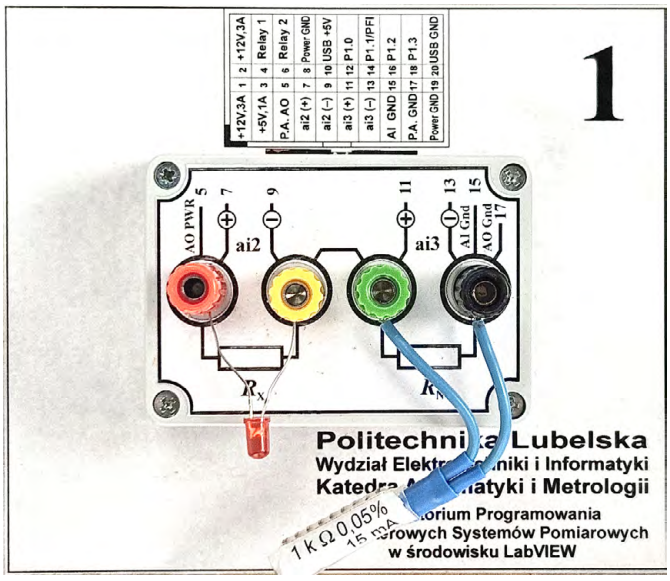
Schemat modułu do badania dwójników przedstawiono na rycinie 5.4. Umożliwia on dołączanie do układu pomiarowego badanego dwójnika $R_X$ oraz rezystora wzorcowego $R_N$. W tym celu na górnej powierzchni modułu zamontowane są cztery zaciski laboratoryjne w kolorach: czerwonym, żółtym, zielonym i czarnym. Zacisk czerwony połączony jest z wyjściem wzmacniacza mocy i zawsze ma on potencjał dodatni względem zacisku czarnego, który jest połączony z masą. Jest to ważne przy badaniu dwójników wymagających odpowiedniej polaryzacji, np. diod LED. Aby zminimalizować wpływ rezystancji przewodów na mierzone spadki napięć $U_X$ oraz $U_N$, zastosowano czteroprzewodowe połączenie Kelvina. Przewody doprowadzające prąd ze wzmacniacza mocy łączą się z przewodami napięciowymi dopiero na zaciskach laboratoryjnych. Rozdzielone są również obwody masy analogowej wejściowej AI GND od masy wzmacniacza mocy PA GND, którą powraca prąd zasilający obwód pomiarowy.

Na bocznej ściance modułu zamontowane jest dwurzędowe złącze szufladowe IDC20, które umożliwia połączenie modułu z panelem ćwiczeniowym. Do tego celu służy płaski przewód 20-żyłowy o długości 30 cm, zakończony wtyczkami IDC20. W ćwiczeniu 5 nie wykorzystamy w pełni tego modułu, a jedynie podczas wstępnego testowania stanowiska dołączymy do niego diodę

LED w miejsce dwójnika $R_X$. Podczas badania wzmacniacza dołączymy do modułu tylko rezystor $R_N$. Rzeczywisty wygląd modułu z dołączoną czerwoną diodą LED i rezystorem wzorcowym $R_N$ pokazano na rycinie 5.5.



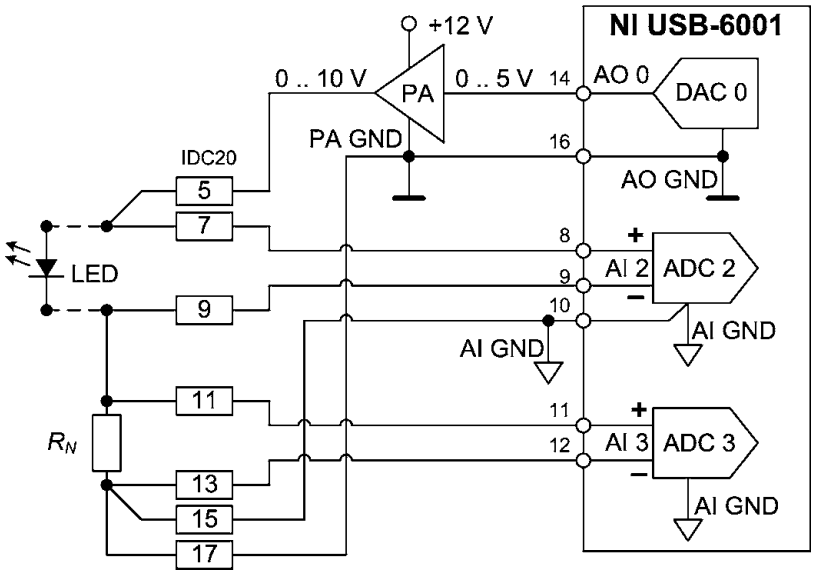

**Ryc. 5.4.** Schemat modułu do badania wzmacniacza




**Ryc. 5.5.** Moduł do badania wzmacniacza z dołączoną diodą LED

## 5.3. Przygotowanie stanowiska laboratoryjnego do pracy

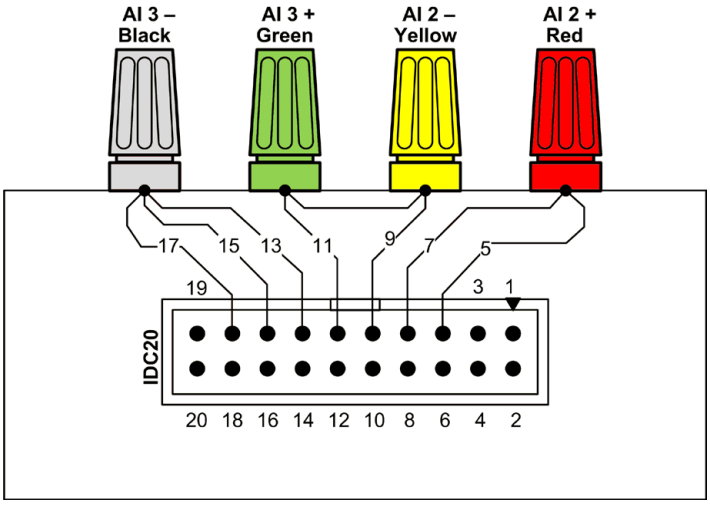
Schemat połączeń stanowiska laboratoryjnego do automatycznego wyznaczania charakterystyki przetwarzania wzmacniacza prądu stałego pokazano na rycinie 5.6. Moduł pomiarowy NI USB-6001 dołączamy do komputera PC za pomocą interfejsu USB. W tabeli 5.2 pokazano zestawienie styków na złączu analogowym modułu pomiarowego wykorzystywanych w ćwiczeniu 5.




**Ryc. 5.6.** Schemat układu połączeń stanowiska do ćwiczenia 5

Napięcie wyjściowe z przetwornika cyfrowo-analogowego AO 0 (PIN 14) jest podawane na wejście wzmacniacza mocy. Napięcie wyjściowe ze wzmacniacza dołączone jest do styku 5 na złączu IDC20, a stamtąd do czerwonego zacisku laboratoryjnego na module ćwiczeniowym. Rozmieszczenie wykorzystywanych sygnałów na złączu IDC20 pokazano na rycinie 5.7. Układ pomiarowy, składający się z szeregowego połączenia diody LED i rezystora wzorcowego $R_{N,}$ połączony jest pomiędzy czerwonym i czarnym zaciskiem laboratoryjnym. Czarny zacisk, poprzez styk 17 na złączu IDC20, połączony jest z masą AO GND wzmacniacza mocy, dzięki czemu zamknięty jest obwód dla prądu pomiarowego. Czarny zacisk jest również, poprzez styk 15 na złączu IDC20, połączony oddzielnym przewodem z analogową masą wejściową AI GND modułu pomiarowego NI USB-6001 (PIN 10). Do pomiarów w ćwiczeniu 5 wykorzystamy tylko wejście

analogowe AI 2 pracujące w trybie różnicowym. Czerwony zacisk poprzez styk 7 na złączu IDC20 jest dołączony do wejścia AI 2+ (PIN 8). Analogicznie żółty zacisk poprzez styk 9 na złączu IDC20 jest dołączony do wejścia AI 2- (PIN 9). Dioda LED będzie dołączona tylko na czas testowania stanowiska. Podczas badania wzmacniacza dioda LED będzie odłączona, a wtedy napięcie $U_N$ na rezystorze wzorcowym wyniesie zero. Zaciski zielony i czarny, poprzez styki 11 i 13 na złączu IDC20, dołączone są do wejść AI 3 + (PIN 11) i AI 3- (PIN 12). Podczas badania wzmacniacza nie będziemy wykorzystywać wejścia AI 3. Będzie ono natomiast niezbędne do realizacji ćwiczeń 6 i 8.

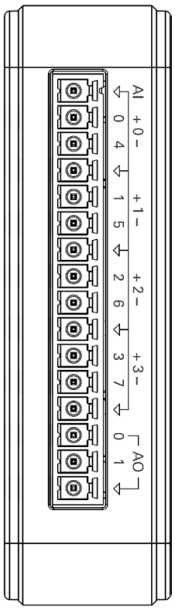



**Ryc. 5.7.** Styki na złączu IDC20 wykorzystywane w ćwiczeniu 5, 6 oraz 8

Stanowisko do pracy przygotowujemy, dołączając moduł z zaciskami laboratoryjnymi do panelu ćwiczeniowego za pomocą płaskiego przewodu zakończonego wtyczkami IDC20. Zwracamy uwagę na położenie styku 1 w gniazdach i wtyczkach IDC20. Prawidłowe połączenie ułatwia wycięcie w obudowie gniazda IDC20 w pobliżu styków 9 i 11, co możemy zauważyć na rycinie 5.7. Następnie do czerwonego i żółtego zacisku laboratoryjnego dołączamy czerwoną diodę LED w miejsce dwójnika $R_X$. Anodę diody dołączamy do zacisku czerwonego, a katodę do zacisku żółtego. Anoda w diodach LED jest zawsze dłuższa od katody. Natomiast przy katodzie na obudowie diody LED znajduje się niewielkie spłaszczenie. Dioda LED, jak każda inna dioda, przewodzi prąd tylko w jednym kierunku: od anody do katody. Przy takim dołączeniu

dioda LED emituje światło w określonym kolorze. Problemem jest natomiast jej stosunkowo niska wytrzymałość napięciowa przy polaryzacji w kierunku zaporowym – wynosi ona tylko kilka woltów. Dlatego dioda LEDmoże łatwo ulec uszkodzeniu przy podłączeniu w odwrotnym kierunku! Do zielonego i czarnego zacisku laboratoryjnego dołączamy rezystor wzorcowy $R_N = 1000\ \Omega$.

**Tab. 5.2.** Rozmieszczenie sygnałów analogowych na złączu modułu NI USB-6001, pogrubioną czcionką zaznaczono styki wykorzystane w ćwiczeniu 5 [6]

| Sygnał | PIN | Złącze analogowe |
|---|---|---|
| AI GND | 1 |  |
| AI 0 (AI 0+) | 2 | |
| AI 4 (AI 0–) | 3 | |
| AI GND | 4 | |
| AI 1(AI 1+) | 5 | |
| AI 5 (AI 1–) | 6 | |
| AI GND | 7 | |
| **AI 2(AI 2+)** | **8** | |
| **AI 6 (AI 2–)** | **9** | |
| **AI GND** | **10** | |
| **AI 3(AI 3+)** | **11** | |
| **AI 7 (AI 3–)** | **12** | |
| AI GND | 13 | |
| **AO 0** | **14** | |
| AO 1 | 15 | |
| **AO GND** | **16** | |



## 5.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 5.4.1. Uruchamianie stanowiska laboratoryjnego

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Upewniamy się, że zasilanie komputera i panelu ćwiczeniowego jest wyłączone, a moduł pomiarowy – odłączony od portu USB komputera.** Sprawdzamy układ zgodnie z opisem

w poprzednim punkcie. Jeśli układ jest połączony prawidłowo, zgłaszamy prowadzącemu zajęcia gotowość do rozpoczęcia realizacji ćwiczenia. Wszelkie zauważone odstępstwa od tego stanu zgłaszamy prowadzącemu zajęcia.

**Dopiero po otrzymaniu od prowadzącego zajęcia wyraźnej zgody włączamy tylko zasilanie komputera. Nie włączamy jeszcze zasilania panelu ćwiczeniowego. Nie dołączamy modułu pomiarowego do portu USB komputera.**

Po uruchomieniu się systemu operacyjnego nie wykonujemy żadnych dodatkowych czynności i przechodzimy do kolejnego punktu.

### 5.4.2. *Self-test* modułu NI USB-6001

Po uruchomieniu komputera w pierwszej kolejności sprawdzamy, czy w systemie są zainstalowane dwa urządzenia NI USB-6001: jedno fizyczne jako **Dev1** i drugie symulowane jako **Dev2**. W tym celu uruchamiamy program NI MAX, rozwijamy listę **My System > Devices and Interfaces** i sprawdzamy, jakie moduły pomiarowe są zainstalowane w systemie. Zainstalowany i prawidłowo podłączony do komputera moduł fizyczny jest widoczny jako aktywny (czarna ikona). Prawidłowo zainstalowany moduł symulowany zawsze widnieje jako aktywny (żółta ikona). Z kolei zainstalowany wcześniej w systemie moduł fizyczny, który obecnie nie jest dołączony do komputera, wyświetla się jako nieaktywny z dodatkowym znakiem „x" na czerwonym tle. Na komputerach w laboratorium powinniśmy mieć zainstalowane jedno aktywne urządzenie fizyczne o nazwie **Dev1** oraz drugie symulowane urządzenie o nazwie **Dev2.** Mogą być na nich zainstalowane również inne urządzenia fizyczne, ale powinny one pozostać nieaktywne. Na komputerach domowych powinniśmy mieć zainstalowane jedno symulowane urządzenie o nazwie **Dev1**. Dzięki temu programy, które napiszemy w domu, powinny od razu działać poprawnie w laboratorium.

Notujemy w protokole nazwy, typy, numery seryjne i statusy zainstalowanych w systemie urządzeń z serii NI USB. Moduł używany na panelu ćwiczeniowym mamy jeszcze odłączony, więc na liście powinien on widnieć z czerwonym znakiem „x" i mieć status **Not Present**. Jeśli na liście na ma wymaganych urządzeń, instalujemy je ponownie zgodnie z opisem w instrukcji do ćwiczenia 1.

Pamiętamy, że moduł używany i zainstalowany w systemie na wcześniejszych zajęciach mógł z nieznanych powodów (najczęściej przypadkowo)

zostać usunięty. Dlatego zawsze na początku zajęć upewniamy się, jaki numer urządzenia system przypisał do aktualnie używanego modułu pomiarowego. Sprawdzamy również, czy na każdych zajęciach pracujemy na tym samym stanowisku laboratoryjnym, gdyż każdy moduł DAQ ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczone w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**. Jeśli więc nasz moduł figuruje w systemie pod inną nazwą, odpowiednio modyfikujemy tworzone programy.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera. Nie załączamy jeszcze zasilania na panelu ćwiczeniowym.** Obserwujemy zachowanie niebieskiej kontrolki LED na obudowie modułu, a wynik notujemy w protokole.

Jeśli poprawnie zainstalowaliśmy używane podczas zajęć urządzenie NI USB-6001, w programie NI MAX przeprowadzamy *self-test* modułu fizycznego i symulowanego zgodnie z opisem w instrukcji do pierwszego ćwiczenia. Rezultaty *self-testu* notujemy w protokole.

**Zgłaszamy gotowość prowadzącemu zajęcia i po otrzymaniu zgody załączamy zasilanie panelu ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku potwierdzi obecność napięcia zasilającego.

### 5.4.3. Testowanie przetwornika cyfrowo-analogowego

Test przetwornika cyfrowo-analogowego wykonujemy podobnie jak w opisie ćwiczenia 4, w sposób pokazany na rycinie 4.9, przy dołączonym do wyjścia analogowego AO 0 wzmacniaczu mocy PA. Tym razem jednak do panelu ćwiczeniowego dołączamy moduł do badania dwójników. W miejsce dwójnika $R_X$ dołączamy czerwoną diodę LED. Zwracamy uwagę na biegunowość diody LED. Anoda ma dłuższe wyprowadzenie i polaryzujemy ją dodatnio względem katody (krótsze wyprowadzenie). **Anodę dołączamy do czerwonego zacisku, a katodę do żółtego. Odwrotne dołączenie może uszkodzić diodę!** Jako rezystor wzorcowy dołączamy $R_N = 1000\ \Omega$. W oknie programu NI MAX wybieramy na liście urządzeń **Dev1** i otwieramy zakładkę **Test Panels**, co było już omówione w ćwiczeniu 1 na rycinie 1.14. W otwartym oknie **Test Panels** wybieramy zakładkę **Analog Output**. Konfigurujemy test wyjścia analogowego analogicznie jak w ćwiczeniu 4, w sposób pokazany na rycinie 4.9.

Zwracamy uwagę na wybór w okienku **Channel Name** wyjścia **Dev1/ao0**. W okienku **Output Value** wpisujemy wartość 0 V i uruchamiamy test, naciskając przycisk **Update**. Dioda nie powinna świecić, co oznacza brak napięcia na wyjściu wzmacniacza mocy PA. Czy układ zachował się poprawnie?

W okienku **Output Value** wpisujemy wartość +5 V i ponownie uruchamiamy test, naciskając przycisk **Update**. Dioda powinna się wówczas zaświecić, sygnalizując obecność napięcia na wyjściu wzmacniacza.

Wpisując różne wartości napięcia w okienku **Output Value** i zatwierdzając je każdorazowo przyciskiem **Update**, sprawdzamy, czy jasność świecenia diody się zmienia.

W ten sposób sprawdzamy poprawność działania układu – od przetwornika DAC aż po zaciski na module do badania dwójników. Wszystkie zaobserwowane rezultaty przeprowadzonych testów notujemy w protokole.

Pozostawiamy otwarte okno **Test Panels** z zakładką **Analog Output** skonfigurowaną do testowania wyjścia analogowego **Dev1/ao0** i przechodzimy do testowania przetwornika analogowo-cyfrowego.

### 5.4.4. Testowanie przetwornika analogowo-cyfrowego

Test przetwornika analogowo-cyfrowego wykonujemy podobnie jak w opisie ćwiczenia 1, ale tym razem będziemy mierzyć dwa napięcia: na badanym dwójniku $R_X$ i na rezystorze wzorcowym $R_N$. Pozostawiamy otwarte okno **Test Panels** z uruchomionym testem przetwornika cyfrowo-analogowego w aktywnej zakładce **Analog Output**. Ponownie w otwartym oknie NI MAX wybieramy zakładkę **Test Panels**. Otworzy się kolejne okno **Test Panels**, w którym wybieramy zakładkę **Analog Input**. Ustawiamy na ekranie obok siebie oba okna **Test Panels**. W polu **Output Value** wpisujemy wartość 0 V i zatwierdzamy przyciskiem **Update**. Dioda LED nie powinna świecić.

Następnie na karcie **Analog Input** wybieramy następujące opcje: **Channel Name: Dev1/ai2, Mode: Finite, Input Configuration: Differential, Max Input Limit: 10, Min Input Limit: −10, Rate (Hz): 1000, Samples To Read: 100.** Uruchamiamy test przyciskiem **Start**. Wynik powinien być zbliżony do kilkuset miliwoltów, zależnie od egzemplarza badanego wzmacniacza. Powtarzamy test na wejściu **Dev1/ai3**. Tym razem wynik powinien być zbliżony do zera z widocznymi zakłóceniami na poziomie pojedynczych miliwoltów. Zaobserwowane wyniki notujemy w protokole.

W polu **Output Value** wpisujemy wartość 5 V i zatwierdzamy przyciskiem **Update**. Dioda powinna zaświecić. Ponownie na karcie **Analog Input** uruchamiamy test na wejściu **Dev1/ai2**, a następnie na wejściu **Dev1/ai3**. Wyniki notujemy w protokole. Na świecącej czerwonej diodzie LED powinien wystąpić spadek napięcia $U_X$ około 2 V, a na rezystorze $R_N$ powinno pojawić się napięcie $U_N$ około 8 V (razem około 10 V), ponieważ wzmacniacz powinien dwukrotnie wzmocnić napięcie z przetwornika DAC równe 5 V. Ze wzoru (5.1) możemy również łatwo obliczyć, że przez diodę płynie prąd *IX* około 8 mA.

$$I_X = \frac{U_N}{R_N} \approx \frac{8\ \text{V}}{1\ \text{k}\Omega} = 8\ \text{mA}\,. \tag{5.1}$$

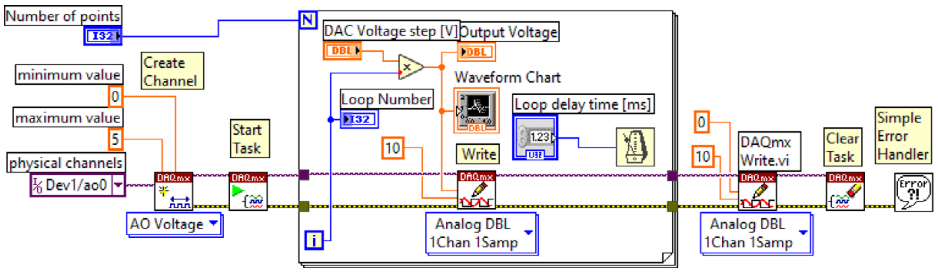
Czy test potwierdził sprawność całego stanowiska? Wnioski z przeprowadzonych testów zapisujemy w protokole. Dodatkowo na potrzeby raportu wykonujemy i zapisujemy do pliku przykładowy zrzut ekranu z przeprowadzonego testu w obu otwartych oknach DAC i ADC. Po uzyskaniu pozytywnych wyników testu zamykamy oba okna **Test Panel** i kończymy pracę programu NI MAX. Uruchamiamy środowisko LabVIEW i przechodzimy do kolejnego punktu.

## 5.5. Programowanie

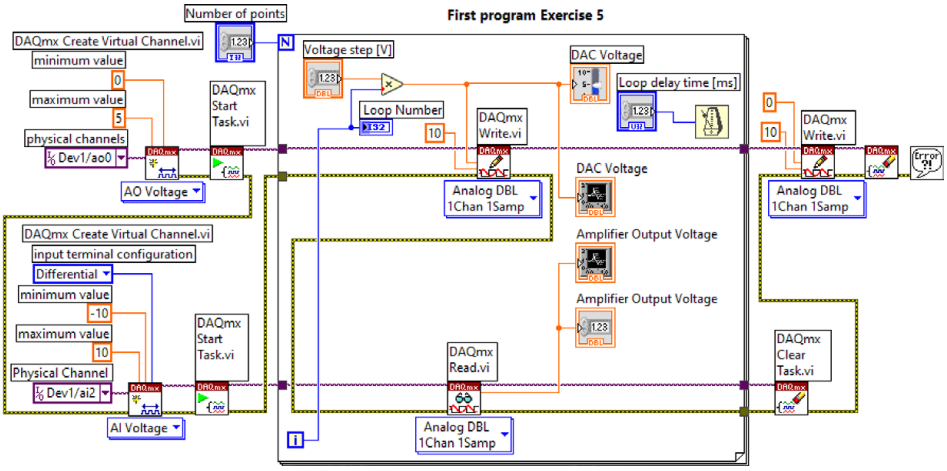
### 5.5.1. Pierwszy program do wyznaczania charakterystyki wzmacniacza prądu stałego

Do modułu z zaciskami laboratoryjnymi powinna być dołączona odpowiednio czerwona dioda LED i rezystor wzorcowy 1 kΩ, tak jak podczas testowania przetwornika DAC. Ćwiczenie 5 jest kontynuacją ćwiczenia 4. Drugi program, opracowany w ćwiczeniu 4, stanowi podstawę do rozpoczęcia ćwiczenia 5. Przykładowy diagram programu 4-2, opracowanego w ćwiczeniu 4, przedstawiono na rycinie 5.8. Znajdujemy ten program na dysku komputera w katalogu ćwiczenia 4. Ustawiamy 51 punktów charakterystyki z krokiem napięcia 0,1 V i opóźnieniem czasowym 100 ms. Uruchamiamy program i upewniamy się, że działa poprawnie. Program powinien zadać do przetwornika DAC narastające wartości napięcia od 0 do 5 V. Na wyjściu wzmacniacza powinno pojawić

się napięcie w zakresie od 0 do 10 V. Dioda powinna świecić coraz silniej przez 5 sekund i zgasnąć po zakończeniu programu. Jeśli program działa poprawnie, robimy jego kopię na potrzeby pierwszego zadania w ćwiczeniu 5, zmieniając numer programu na 5-1-a. Pamiętamy, aby nie modyfikować programu, który jest ostateczną wersją poprzednio wykonanego zadania.




**Ryc. 5.8.** Drugi program z ćwiczenia 4 potrzebny do rozpoczęcia ćwiczenia 5




**Ryc. 5.9.** Diagram programu uzupełnionego o pomiar napięcia

Uzupełniamy program o pomiar napięcia w różnicowym kanale wejściowym analogowym ai2 urządzenia **Dev1**. Przykładowy diagram odpowiednio uzupełnionego programu 5-1-a przedstawiono na rycinie 5.9. W tym celu rozciągamy pętlę **While** w dół, żeby zrobić miejsce na dodatkowe funkcje. Wybieramy zasobnik z funkcjami **Measurement I/O > NI DAQmx > DAQmx – Data Acquisition**. Wybieramy i wstawiamy do diagramu, po lewej

stronie pętli **While**, funkcje **DAQmx Create Virtual Channel** oraz **DAQmx Start Task**. Wewnątrz pętli **While** wstawiamy funkcję **DAQmx Read.** Na zewnątrz pętli While, po jej prawej stronie, umieszczamy funkcję **DAQmx Clear Task**. Wyrównujemy ułożenie wszystkich funkcji względem siebie i łączymy ze sobą wszystkie wyjścia **task out** z kolejnym wejściem **task in** według ryciny 5.9. Podobnie łączymy wyjścia **error out** z wejściami **error in**. Dla wygody, klikając każdą funkcję prawym przyciskiem myszy, włączamy dla niej opis: **Visible Items > Label**.

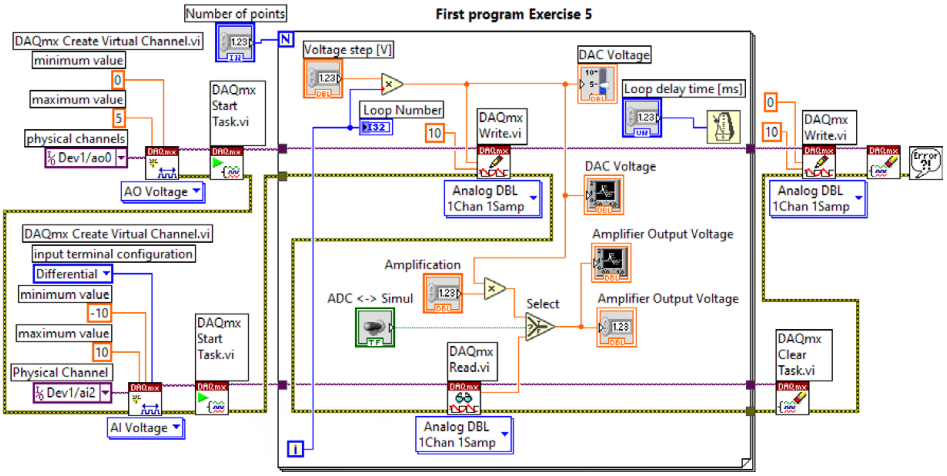
Na panelu umieszczamy wskaźnik mierzonego napięcia **Controls > Modern > Numeric > Numeric Indicator** oraz wykres przebiegu **Controls > Modern > Graph > Waveform Chart.** Na diagramie dołączamy nowo utworzone wskaźniki do wyjścia funkcji **Read**. Zwracamy uwagę, aby wszystkie tunele były ustawione w trybie **Last Value**. Teraz konfigurujemy funkcję **DAQmx Create Virtual Channel** do pomiaru napięcia na wejściu analogowym. Rozwijamy menu **Polymorphic VI Selector** i wybieramy opcję **Analog Input > Voltage**. Następnie konfigurujemy funkcję **DAQmx Read** do odczytu w jednym kanale jednej próbki z wejścia analogowego. Rozwijamy menu **Polymorphic VI Selector** i wybieramy opcję **Analog > Single Channel > Single Sample > DBL.**

Dołączamy odpowiednie dane do wejść konfigurujących funkcji **DAQmx Create Virtual Channel**. Tworzymy zadajniki wartości stałych (**Create Constant**) na wejściach: input terminal configuration (wybieramy opcję Differential), **minimum value** (wprowadzamy wartość -10), **maximum value** (wprowadzamy wartość 10) oraz **physical channel** (wybieramy z listy wejście ai2 w urządzeniu **Dev1**).

Uruchamiamy program i sprawdzamy, czy działa poprawnie. Program powinien mierzyć napięcie na diodzie, które powinno narastać do około 2 V. Jeśli tak jest, zapisujemy program i tworzymy jego kopię 5-1-b do dalszych modyfikacji.

Uzupełniamy diagram o symulację mierzonego napięcia w sposób przedstawiony na rycinie 5.10. Dzięki temu sprawdzimy, czy tworzone algorytmy przetwarzania danych z przetwornika ADC działają poprawnie. Na panelu umieszczamy zadajnik numeryczny **Controls > Modern > Numeric > Numeric Control**, nadajemy mu nazwę **Amplification**, wprowadzamy wartość 2 i łączymy go z funkcją mnożenia, linią napięcia DAC i funkcją przełącznika

**Programming > Comparison > Select**. Na wejściu sterującym funkcji **Select** tworzymy prawym kliknięciem zadajnik logiczny w postaci przełącznika.



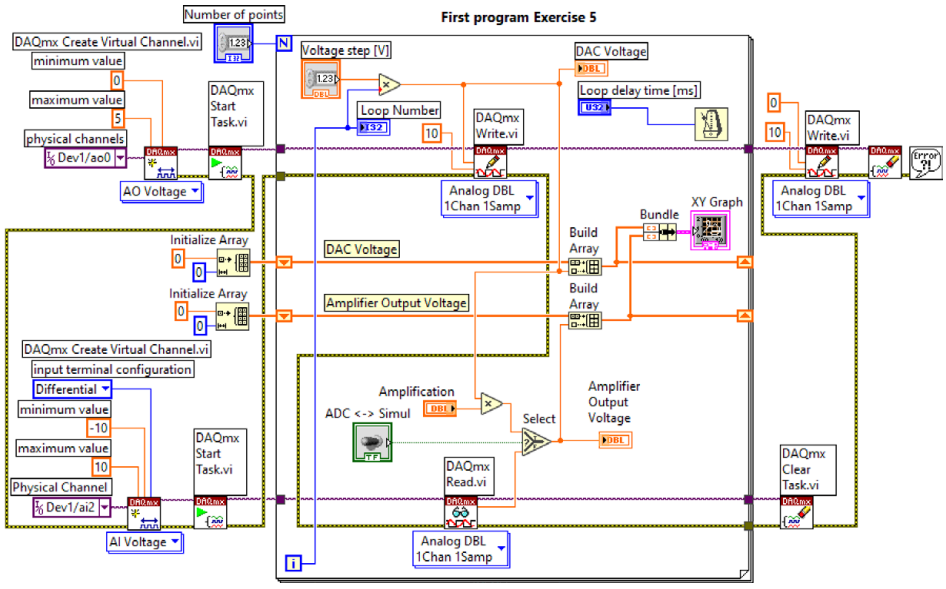

**Ryc. 5.10.** Diagram programu uzupełnionego o symulację pomiaru napięcia

Uruchamiamy program i sprawdzamy, czy działa poprawnie. W trybie pomiarów rzeczywistych powinien dałać on tak jak poprzednio. W trybie symulacji powinien pokazywać liniowo narastające wartości symulowanego napięcia od 0 do 10 V. Takie napięcie mierzy w idealnej sytuacji przetwornik ADC na wyjściu wzmacniacza o wzmocnieniu równym 2 V/V. Jeśli tak jest, zapisujemy program i robimy jego kopię 5-1-c do dalszych modyfikacji.

Uzupełniamy program o budowanie dwóch tablic z wartościami napięcia: zadawanego do przetwornika DAC oraz mierzonego przez przetwornik ADC na wyjściu wzmacniacza. Następnie z danych zgromadzonych w tych dwóch tablicach tworzymy wykres XY. Zbędne staną się natomiast dwa wykresy typu Waveform Chart, na których dotychczas pokazywane było napięcie zadawane do przetwornika DAC oraz napięcie zmierzone przetwornikiem ADC. Wykresy te usuniemy później, w odpowiednim momencie. Przykładowy diagram tak zmodyfikowanego programu przedstawiono na rycinie 5.11. Zadanie to realizujemy w opisanych poniżej etapach.

Najpierw budujemy tablice z danymi numerycznymi do wygenerowania wykresu XY. Budowanie tablic byłoszczegółowo wyjaśnione w ćwiczeniu 3, podczas badania generatora częstotliwości wzorcowych. Najpierw umieszczamy rejestry

przesuwne na krawędzi pętli **While**. Rejestry te zawsze powstają jako para: wejściowy i wyjściowy. Następnie wewnątrz pętli **While** umieszczamy funkcję **Programming > Array > Build Array**. Na zewnątrz pętli, przy rejestrach wejściowych, inicjalizujemy dwie puste tablice funkcją **Programming > Array > Initialize Array**. Łączymy umieszczone funkcje i rejestry przesuwne zgodnie z ryciną 5.11.



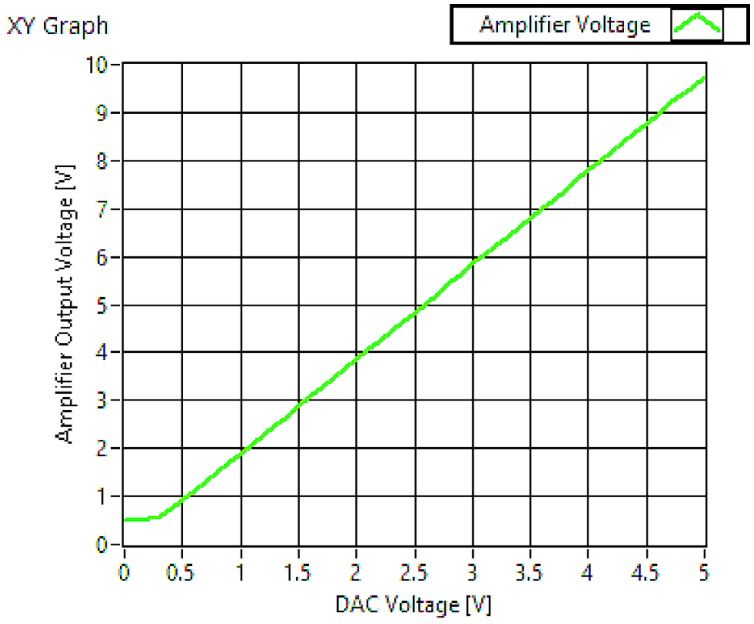

**Ryc. 5.11.** Diagram programu uzupełniony o budowę dwóch tablic i wykres XY

Teraz możemy utworzyć wykres XY na podstawie danych zgromadzonych w dwóch tablicach. Tworzenie wykresu XY było wyjaśnione w ćwiczeniu 4, podczas badania silnika DC małej mocy. Najpierw używamy funkcji **Programming > Cluster, Class, & Variant > Bundle** do połączenia dwóch tablic w jeden klaster danych. Do górnego wejścia funkcji **Bundle** dołączamy dane dla osi X tworzonego wykresu, a do dolnego wejścia – dane dla osi Y.

Przełączamy się na panel programu i usuwamy dwa wykresy typu **Waveform Chart**. W ich miejsce umieszczamy na panelu wykres typu XY: **Controls > Modern > Graph > XY Graph**. Wracamy na diagram, usuwamy połączenia pozostałe po usuniętych z panelu wykresach i łączymy wyjście funkcji **Bundle** z wejściem **XY Graph**. Klikając wykres prawym przyciskiem myszy, wybieramy opcję **Properties** i konfigurujemy go. Nadajemy osiom X i Y odpowiednie nazwy, definiujemy kolory, styl i grubości linii oraz styl i kolor punktów itp. Wykazujemy

się przy tym minimalnym poczuciem estetyki. Wyłączamy automatyczne skalowanie osi X i Y. Zakres osi X ustawiamy od 0 do 5 V, a zakres osi Y od 0 do 10 V. Pamiętamy o podaniu jednostek dla każdej z osi. Przykładową postać panelu po tych modyfikacjach pokazano na rycinie 5.13.

Odłączamy diodę od układu, dzięki czemu w analogowym kanale wejściowym ai2 będziemy mierzyć napięcie na wyjściu badanego wzmacniacza. Zostawiamy dołączony rezystor wzorcowy. Ponieważ w ten sposób praktycznie przerywamy obwód dla przepływu prądu pomiarowego z wyjścia wzmacniacza, spadek napięcia na rezystorze wzorcowym wynosi zero. Przyjmujemy, że ujemny zacisk kanału wejściowego AI 2– jest teraz połączony z masą analogową AI GND. Uruchamiamy uzyskany program i sprawdzamy, czy działa zgodnie z oczekiwaniami. W trybie symulacji powinniśmy otrzymać idealną linię prostą o nachyleniu 2 V/V. Jest to charakterystyka przetwarzania idealnego wzmacniacza o dwukrotnym wzmocnieniu. W przypadku pomiarów rzeczywistych wykres powinien wyglądać podobnie, ale na początku charakterystyki zauważymy wyraźną nieliniowość. Przykładową charakterystykę, którą otrzymano na stanowisku ćwiczeniowym za pomocą omawianego programu, pokazano na rycinie 5.12.



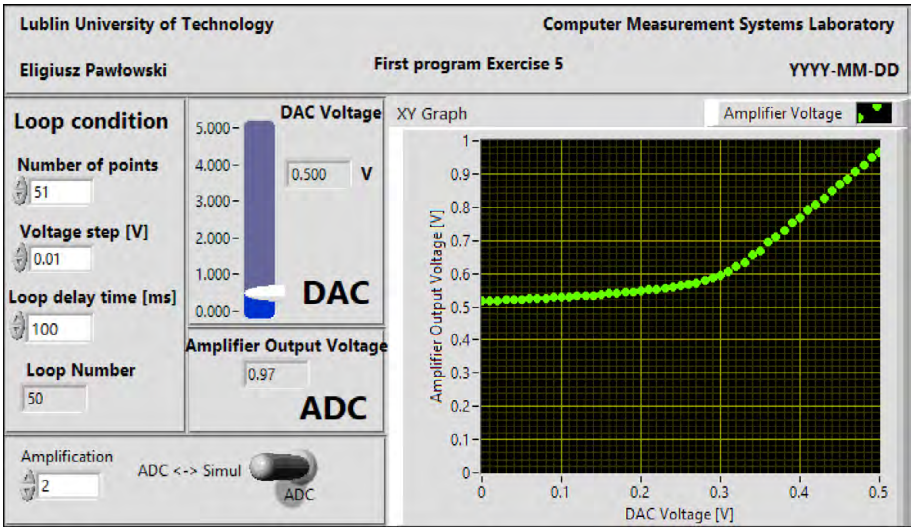

**Ryc. 5.12.** Przykładowa charakterystyka wzmacniacza w pełnym zakresie napięć

Na otrzymanym wykresie widzimy wyraźnie, że wzmacniacz nie pracuje poprawnie dla małych wartości napięć. Nie jest to więc na pewno wzmacniacz typu *Rail-to-Rail* [34]. Dla napięć wejściowych bliskich zeru na wyjściu wzmacniacza występuje już napięcie rzędu kilkuset miliwoltów. Dopiero przy napięciach wejściowych zbliżających się do 0,5 V napięcie na wyjściu wzmacniacza zaczyna narastać. Dla większych napięć wzmacniacz pracuje już poprawnie, a napięcie na wyjściu jest dwukrotnie większe od napięcia wejściowego. W przypadku różnych egzemplarzy wzmacniaczy zastosowanych na stanowiskach laboratoryjnych ten początkowy fragment charakterystyki wygląda inaczej, mimo że wszystkie są tego samego typu. Wynika to z nieuniknionych rozrzutów produkcyjnych. Zauważamy również, że tego typu wzmacniacze akustyczne nie są przeznaczone do pracy z napięciami wyjściowymi bliskimi napięciom zasilania. Dodatkowo na stanowisku laboratoryjnym wzmacniacze te pracują w nietypowej dla siebie konfiguracji. Dlatego charakterystykę, pokazaną na rycinie 5.12, uznajemy za zgodną z oczekiwaniami.

Ustawiamy krok napięcia na mniejszą wartość, np. 0,01 V, i badamy dokładniej początek charakterystyki wzmacniacza. Dobieramy odpowiednio liczbę kroków w programie. Dopasowujemy zakresy osi do otrzymanych wyników, widocznych na rycinie 5.12. Przykładowo ustawiamy zakres osi X od 0 do 0,5 V, a zakres osi Y od 0 do 1 V. Jeśli dla badanego wzmacniacza początek wykresu na rycinie 5.12 wygląda inaczej, dobieramy inne zakresy osi, odpowiednio większe lub mniejsze. Badamy, dla jakich napięć wzmacniacz zaczyna pracować poprawnie, tzn. dla jakich wartości wykres zaczyna w przybliżeniu spełniać zależność $Y \approx 2X$. Zapisujemy uzyskany wykres do pliku za pomocą opcji **Export Simplified Image** i zamieszczamy go w raporcie. Nazwę pliku z wykresem zapisujemy w protokole. Przykładową postać wykresu pokazano na rycinie 5.13.

Ustawiamy czas opóźnienia na 0 ms i przyciskiem  uruchamiamy program. Środowisko LabVIEW automatycznie ponawia uruchamianie programu. Zmieniamy czas opóźnienia co 1 ms i sprawdzamy, czy wpływa to na kształt charakterystyki. Wnioski zapisujemy w protokole. Zatrzymujemy program przyciskiem Stop  na pasku narzędziowym. Ustawiamy poprzednie zakresy osi dla pełnego zakresu napięć i nastawy kroku napięcia oraz opóźnienia. Jeśli program działał prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**.

Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne wzajemne związki. Używamy narzędzi **Align**, **Distribute**, **Resize**, **Reorder**, aby wyrównać położenie obiektów na panelu, pogrupować je (**Group**) i zablokować ich położenie (**Lock**). Przykładowy panel finalnej wersji pierwszego programu do badania wzmacniacza pokazano na rycinie 5.13.




**Ryc. 5.13.** Przykładowy panel pierwszego programu do badania wzmacniacza

Program zapisujemy w katalogu wskazanym przez prowadzącego zajęcia. Nazwy plików tworzymy według schematu:

YYYYMMDD_Nazwisko_KSP_X-N-L.vi

gdzie poszczególne człony oznaczają: YYYY – rok, MM – miesiąc, DD – dzień, X – numer ćwiczenia, N – numer programu, L – literę (a, b, c itd.) dla kolejnych wersji tego samego zadania. Pamiętamy, aby w nazwie nie używać polskich znaków, spacji ani innych znaków specjalnych. Nazwę pliku oraz katalogu notujemy w protokole. W nazwie tego pliku uwzględniamy numer programu 5-1.

Za pomocą polecenia **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Odnajdujemy

zapisane pliki na dysku i sprawdzamy ich zawartość. W protokole zapisujemy nazwę utworzonego katalogu i nazwy plików wraz z opisem ich zawartości. Zapisane pliki wykorzystujemy do sporządzeniaraportu.

Tworzymy kopię programu na potrzeby kolejnego zadania, zmieniając jego numer na 5-2-a. Pamiętamy, aby nigdy nie modyfikować programu będącego ostateczną wersją poprzednio wykonanego zadania. Zawsze po zapisaniu ostatecznej wersji programu, który w pełni realizuje dany punkt ćwiczenia, tworzymy jego kopię i nadajemy jej nową nazwę.

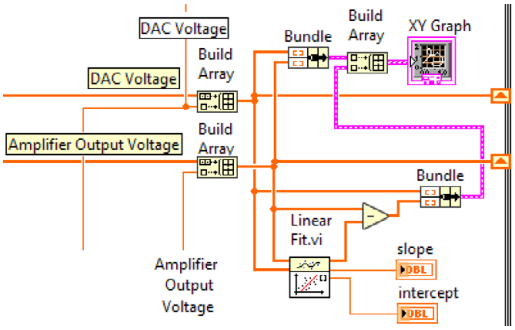
### 5.5.2. Drugi program do liniowej aproksymacji charakterystyki wzmacniacza

Teraz uzupełniamy program o liniową aproksymację charakterystyki, obliczanie błędów nieliniowości wzmacniacza i wyświetlanie ich na wykresie. Odpowiedni fragment diagramu pokazano na rycinie 5.14.

Najpierw umieszczamy na diagramie funkcję **Mathematic > Fitting > Linear Fit** i, klikając prawym przyciskiem myszy jej wyjścia **slope** i **intercept**, tworzymy wskaźniki numeryczne (**Create Indicator**) pokazujące nachylenie i przesunięcie linii aproksymującej. Przechodzimy na panel, odszukujemy te wskaźniki i umieszczamy je w widocznym miejscu. Następnie odejmujemy tablicę dostępną na wyjściu **Best Linear Fit** od tablicy napięć na wyjściu wzmacniacza zmierzonych przez przetwornik ADC. Otrzymana w ten sposób tablica zawiera błędy nieliniowości wzmacniacza. Obliczone błędy nieliniowości przedstawiamy na wykresie **XY Graph**, który w obecnej postaci pokazuje już charakterystykę wzmacniacza. Dodajemy do niego kolejny wykres. Funkcją **Bundle** tworzymy drugi klaster z tablicami dla osi X i Y. Dane dla osi X pozostają te same, tzn. **DAC Voltage**. Danymi dla osi Y drugiego wykresu jest tablica z obliczonymi błędami nieliniowości. Usuwamy połączenie na wejściu **XY Graph** i w to miejsce wstawiamy funkcję **Build Array** z dwoma wejściami. Łączymy sygnały zgodnie z ryciną 5.14.

Błędy nieliniowości są bardzo małe, dlatego dodajemy do wykresu drugą oś Y z odpowiednio dobranym, mniejszym zakresem. Przechodzimy na panel. Klikając prawym przyciskiem myszy oś Y wykresu, dodajemy nową oś za pomocą polecenia **Duplicate Scale**. Klikając wykres prawym przyciskiem myszy, wybieramy opcję **Properties** i konfigurujemy go. W zakładce **Scales** nadajemy nazwę nowej osi, a w zakładce **Plots** nadajemy nazwę nowej linii na wykresie

oraz przypisujemy jej odpowiednie źródło danych dla osi **Y-scale** i **X-scale**. Definiujemy kolory, grubości linii itp. Wyłączamy automatyczne skalowanie wszystkich osi. Dla nowej osi błędów Y ustawiamy wstępnie zakres od –0,1 V do +0,1 V.

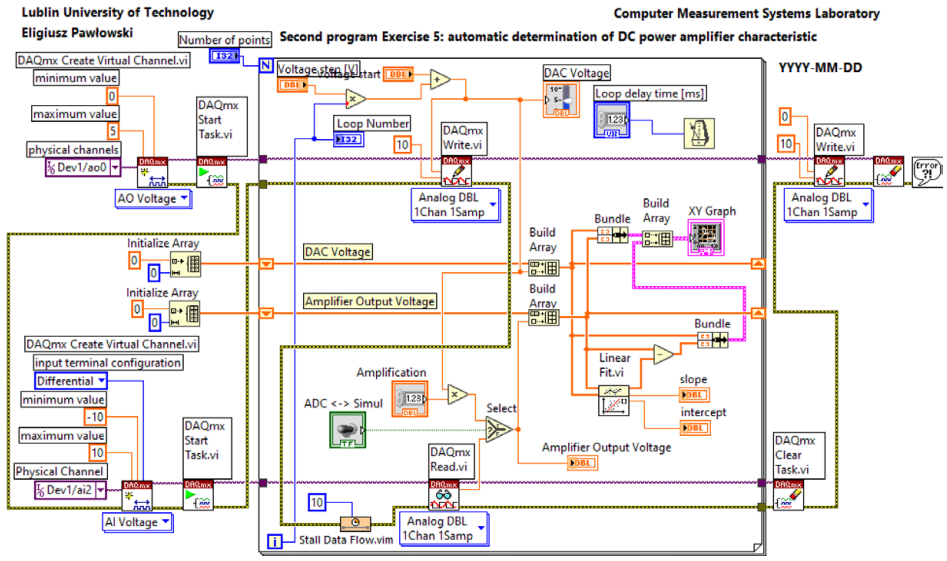



**Ryc. 5.14.** Fragment diagramu realizujący aproksymację liniową i wykres błędów nieliniowości

Uruchamiamy program i sprawdzamy, czy działa poprawnie. W trybie symulacji program powinien pokazać liniowo narastające wartości symulowanego napięcia od 0 do 10 V. Błędy nieliniowości w całym zakresie powinny wynosić zero. Symulowany wzmacniacz jest oczywiście wzmacniaczem idealnym. W trybie pomiarów rzeczywistych błędy nieliniowości są duże na początku charakterystyki. Potwierdzamy w ten sposób zaobserwowany już wcześniej wniosek, że wzmacniacz nie pracuje poprawnie dla małych napięć rzędu kilkuset mV. Jednak w znacznej części charakterystyki wzmacniacz zachowuje liniowość. Modyfikujemy program tak, aby zbadać go tylko w zakresie pracy liniowej.

Na diagramie odszukujemy funkcję mnożenia, która oblicza wartość napięcia zadawanego do przetwornika DAC na podstawie kroku napięcia i numeru pętli. Usuwamy połączenie na wyjściu tej funkcji mnożenia i umieszczamy w tym miejscu funkcję sumowania. Na jej jednym wejściu, klikając prawym przyciskiem myszy, tworzymy kontrolkę do zadawania początkowej wartości napięcia i wstępnie przypisujemy jej wartość 0. Uzupełniamy połączenia zgodnie z ryciną 5.15. Uruchamiamy program i sprawdzamy jego działanie po tej modyfikacji. Zadajemy początkową wartość napięcia 1 V i zmniejszamy liczbę kroków do 41. Pamiętamy, aby zawsze dobierać liczbę kroków odpowiednio do napięcia początkowego i kroku napięcia. Nie zadajemy do przetwornika DAC napięcia większego od 5 V, ponieważ z pewnością przesteruje to wzmacniacz. Czy wykres błędów jest

teraz zgodny z oczekiwaniami? Zmieniamy początkowe napięcie i obserwujemy wykres błędów. Wnioski zapisujemy w protokole.




**Ryc. 5.15.** Diagram programu uzupełniony o liniową aproksymację

Zauważamy, że pierwszy pomiar zawsze wyraźnie odstaje od pozostałych, a jego błąd jest znacznie większy. Problemem jest różnica w szybkości działania przetworników DAC i ADC, co pokazano w tabeli 5.1. Przetwornik ADC jest znacznie szybszy od przetwornika DAC. Przetwornik ADC wykonuje pomiar wcześniej, zanim przetwornik DAC wygeneruje pierwszą wartość napięcia. Dzieje się tak, pomimo że pomiar ADC na diagramie realizujemy później niż przetwarzanie DAC. Aby zaradzić temu problemowi, wprowadzamy niewielkie opóźnienie czasowe przed wyzwoleniem pomiaru ADC. Używamy do tego celu funkcji **Programming > Timing > Stall Data Flow**. Umieszczamy tę funkcję na diagramie przed funkcją **Read**. Na jej wejściu **milliseconds to wait** tworzymy stałą numeryczną o wartości 10. Opóźnia to wykonanie pomiaru o 10 ms; możemy wypróbować również inne, mniejsze opóźnienia. Usuwamy połączenie na wejściu **error in** funkcji **Read** i łączymy linię **error** poprzez wstawioną funkcję opóźnienia czasowego. Przykładowy sposób wprowadzenia opisanych zmian pokazano w dolnej części ryciny 5.15.

Uruchamiamy program i sprawdzamy, czy działa poprawnie. Zadajemy początkową wartość napięcia 1 V i zmniejszamy liczbę kroków do 41. W trybie

symulacji nie obserwujemy żadnych zmian w działaniu programu. W trybie pomiarów rzeczywistych błędy nieliniowości na początku charakterystyki powinny być zbliżone do wszystkich pozostałych. Zmniejszamy napięcie początkowe i znajdujemy wartość, dla której błąd nieliniowości pierwszego pomiaru gwałtownie wzrasta. Wnioski zapisujemy w protokole i porównujemy je z wcześniejszymi notatkami na ten temat. Czy otrzymaliśmy podobną wartość?

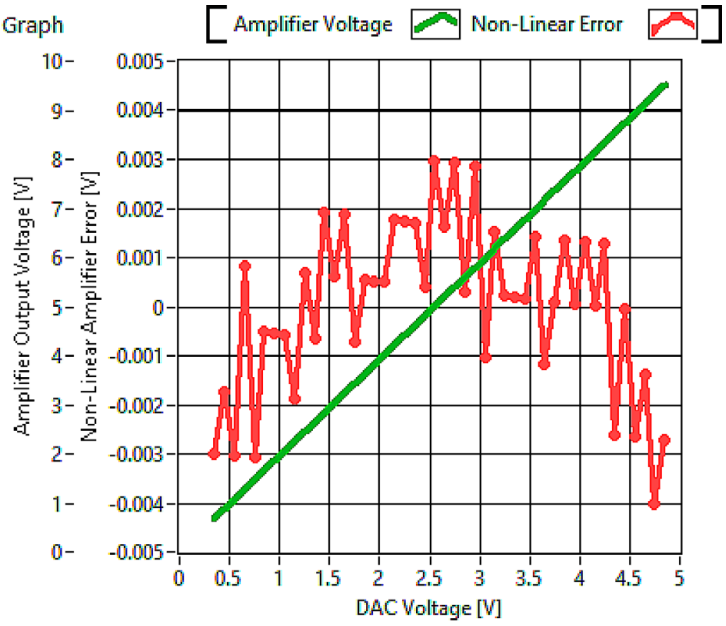
Dobieramy krok napięcia i liczbę kroków, aby otrzymać charakterystykę wzmacniacza od ustalonej minimalnej wartości do 5 V. Sprawdzamy błędy nieliniowości na końcu zakresu. Dla dużych napięć wzmacniacz może ulec nasyceniu i przekroczyć zakres pracy liniowej. W razie potrzeby zmniejszamy liczbę kroków. Zwracamy również uwagę na napięcie odczytane z przetwornika ADC, który pozwala zmierzyć napięcie maksymalnie do około 10,5 V [6]. Sprawdzamy ten efekt, zwiększając liczbę kroków i obserwując ostatnią wartość napięcia odczytanego z przetwornika ADC. Dla jakiej liczby kroków wartość ta już się nie zmienia? Jakie maksymalne napięcie zmierzył przetwornik ADC? Czy jest to wartość większa od 10 V? Wnioski zapisujemy w protokole.

Biorąc pod uwagę wszystkie zaobserwowane efekty, dobieramy optymalne parametry w programie i mierzymy charakterystykę wzmacniacza w całym jego zakresie pracy liniowej. Dobieramy optymalny zakres osi Y dla wykresu błędów. Zapisujemy wykres do pliku graficznego BMP (korzystając z opcji **Export Simplified Image**) i zamieszczamy go w sprawozdaniu. Przykładowy wykres pokazano na rycinie 5.16. Ponieważ każdy egzemplarz wzmacniacza ma inne błędy, wykresy otrzymane na różnych stanowiskach nie są takie same.

Porównujemy otrzymany wykres z wykresami z innych stanowisk. Zwracamy uwagę na pewną ogólną, średnią tendencję w zmianie wartości błędów oraz na pewien składnik przypadkowy. Oceniamy przypadkowe, skokowe zmiany błędu nieliniowości i porównujemy je z błędami kwantowania przetworników ADC i DAC, podanymi w tabeli 5.1. Wnioski zapisujemy w protokole.

Wykres na rycinie 5.16 pokazuje, że błędy nieliniowości wzmacniacza są obarczone błędami kwantowania przetworników ADC i DAC. Wynika to z faktu, że błędy nieliniowości wzmacniacza są małe i porównywalne z błędami kwantowania przetworników. Jednak błędy kwantowania są błędami przypadkowymi, a błędy nieliniowości mają charakter deterministyczny. Możemy

spróbować je rozdzielić za pomocą odpowiedniej aproksymacji. Realizujemy to w następnym punkcie ćwiczenia.




**Ryc. 5.16.** Przykładowa charakterystyka przetwarzania wzmacniacza i jego błędy nieliniowości

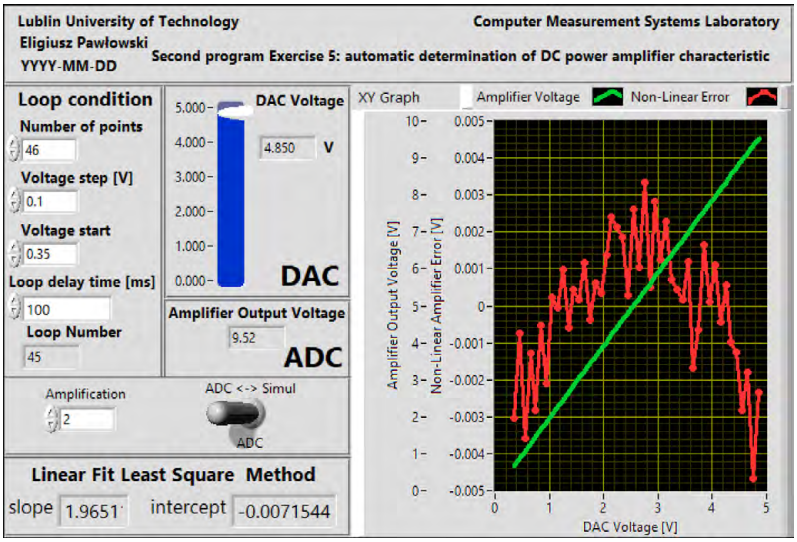
Jeśli program działał prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne, wzajemne związki. Przykładową postać panelu dla drugiego programu pokazano na rycinie 5.17.

Zapisujemy program do pliku. Nazwę pliku i katalogu notujemy w protokole. W nazwie pliku uwzględniamy numer programu 5-2.

Za pomocą polecenia **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Odnajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwy plików wraz z opisem ich zawartości zapisujemy w protokole. Zapisane pliki wykorzystujemy do sporządzenia raportu.

Tworzymy kopię programu na potrzeby kolejnego zadania, zmieniając jego numer na 5-3-a. Pamiętamy, aby nigdy nie modyfikować programu będącego ostateczną wersją poprzednio wykonanego zadania. Zawsze po zapisaniu

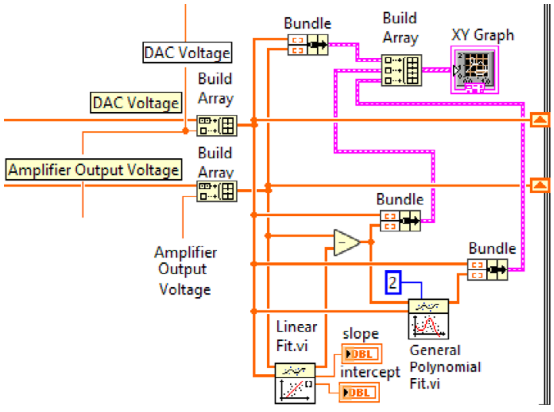
ostatecznej wersji programu, który w pełni realizuje dany punkt ćwiczenia, tworzymy jego kopię i nadajemy jej nową nazwę.




**Ryc. 5.17.** Przykładowy panel drugiego programu do badania wzmacniacza

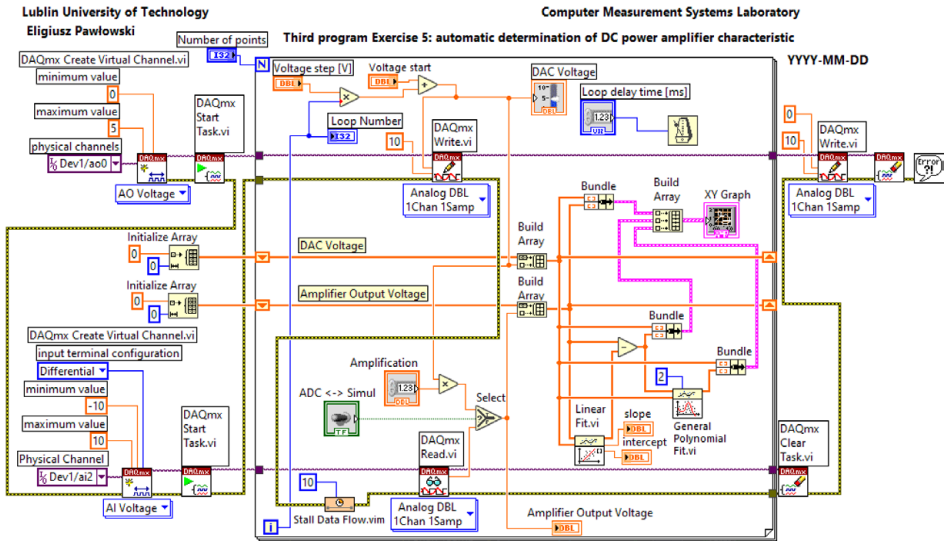
### 5.5.3. Trzeci program do aproksymacji błędów liniowości wielomianem

Teraz uzupełniamy program o aproksymację błędów. Obserwując wykres błędów na rycinie 5.16, zauważamy, że po usunięciu składnika przypadkowego przypomina on parabolę. Stosujemy więc aproksymację wielomianem drugiego stopnia. Odpowiedni fragment diagramu pokazano na rycinie 5.18.




**Ryc. 5.18.** Fragment diagramu realizujący aproksymację wielomianową błędów nieliniowości

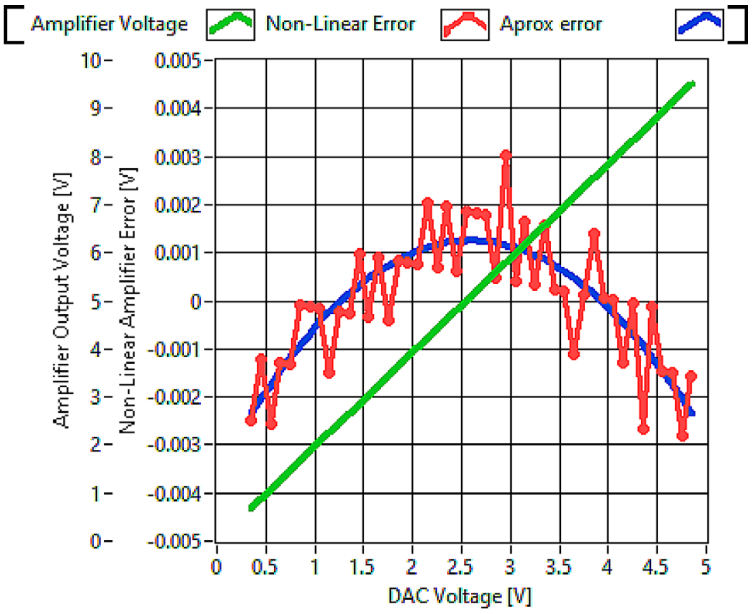
Najpierw umieszczamy na diagramie funkcję **Mathematic > Fitting > General Polynomial Fit** i, klikając prawym przyciskiem myszy jej wejście **polynomial order**, tworzymy stałą numeryczną o wartości 2. Do wejścia X dołączamy tablicę **DAC Voltage**, a do wejścia Y dołączamy tablicę z błędem aproksymacji. Do wyjścia **Best Polynomial Fit** dołączamy funkcję **Bundle** (do jej dolnego wejścia) – będzie to oś Y tworzonego wykresu. Do górnego wejścia funkcji **Bundle** dołączamy tablicę **DAC Voltage** – będzie to oś X dla tego wykresu. Funkcję **Build Array** rozciągamy do dołu i tworzymy trzecie wejście, do którego dołączamy wyjście funkcji **Bundle** z nowym wykresem. Nie tworzymy nowej skali dla nowego wykresu, ponieważ będzie on miał wspólną skalę z błędami nieliniowości. Klikając wykres prawym przyciskiem myszy, wybieramy opcję **Properties** i konfigurujemy go. W zakładce **Plots** nadajemy nazwę nowej linii na wykresie oraz przypisujemy jej odpowiednie dane dla osi **Y-scale** i **X-scale**. Definiujemy kolory, grubość linii itp. Docelową postać diagramu pokazano na rycinie 5.19.




**Ryc. 5.19.** Diagram trzeciego programu do badania wzmacniacza

Uruchamiamy program i sprawdzamy, czy działa poprawnie. Przykładowy wykres z aproksymacją błędów nieliniowości pokazano na rycinie 5.20. Widzimy na nim, że wielomian drugiego stopnia dobrze przybliża błąd nieliniowości badanego wzmacniacza. Teraz możemy oszacować błędy nieliniowości wzmacniacza na około ±2 mV, a więc mają one wartości porównywalne

z błędami kwantowania przetworników ADC i DAC [6]. Zauważamy również, że błędy widoczne na rycinach 5.16, 5.17 oraz 5.20 wyglądają podobnie, ale mają inne wartości. Potwierdza to wniosek, że błędy kwantowania mają charakter losowy. Jeśli natomiast uruchomimy program wielokrotnie, otrzymamy bardzo podobną aproksymację błędów nieliniowości. Potwierdza to wniosek, że błędy nieliniowości mają charakter deterministyczny. Pamiętamy również, że przedstawione wykresy są przykładowe. Ze względu na rozrzuty produkcyjne parametrów wzmacniaczy, wykresy otrzymane na poszczególnych stanowiskach różnią się między sobą.

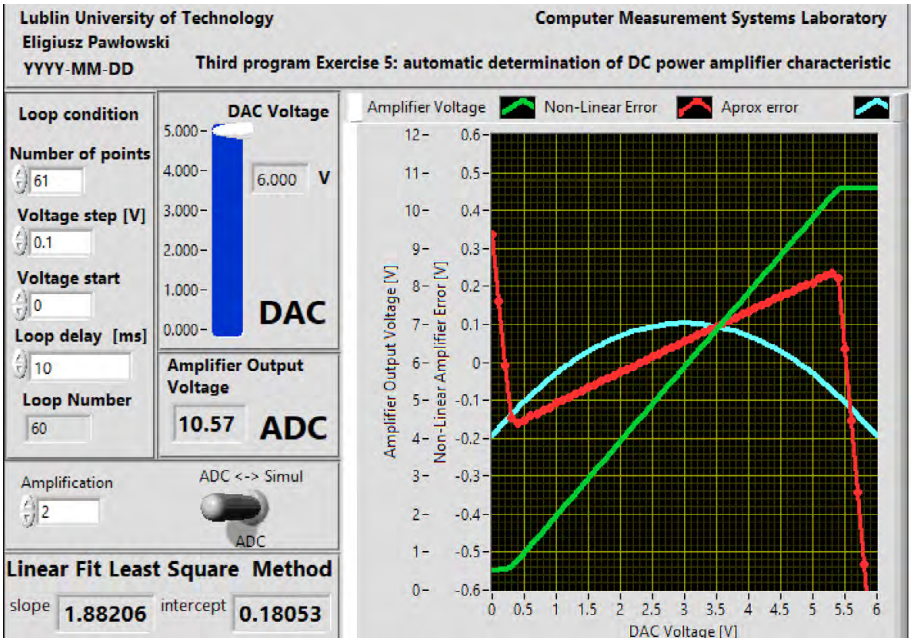



**Ryc. 5.20.** Przykładowa charakterystyka wzmacniacza z aproksymacją błędów nieliniowości wielomianem drugiego stopnia

Uaktualniamy odpowiednio panel oraz uzupełniamy go niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne, wzajemne związki. Przykładowy panel odpowiednio uzupełnionego programu przedstawiono na rycinie 5.21.

Jeśli program działa prawidłowo, deklarujemy aktualne nastawy w programie jako domyślne za pomocą polecenia **Edit > Make Current Values Default**. Zapisujemy program w pliku. W nazwie tego pliku uwzględniamy

numer programu 5-3. Program ten jest niezbędny do realizacji ćwiczenia 6, dlatego dokładnie notujemy nazwę pliku i katalogu w protokole.




**Ryc. 5.21.** Panel trzeciego programu do badania wzmacniacza

Za pomocą polecenia **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Odnajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. W protokole zapisujemy nazwę utworzonego katalogu i nazwy plików wraz z opisem ich zawartości. Zapisane pliki wykorzystujemy do sporządzenia raportu.

## 5.6. Badania wzmacniacza prądu stałego

### 5.6.1. Wstępna ocena zakresu pracy wzmacniacza

Korzystając z ostatniego, trzeciego opracowanego programu, przeprowadzamy badania wzmacniacza prądu stałego zastosowanego na panelu ćwiczeniowym. Celem pierwszego eksperymentu jest wstępne ustalenie zakresu pracy liniowej wzmacniacza. Zadajemy w programie parametry odpowiednie do wykonania

pomiarów w zakresie od 0 do 6 V na wyjściu przetwornika DAC. Jest to z pewnością zakres przekraczający obszar pracy liniowej badanego wzmacniacza. Zauważamy nieprawidłowości w charakterystyce przetwarzania dla skrajnych wartości. Przykładowe nastawy w programie to: napięcie początkowe 0 V, krok napięcia 0,1 V, liczba kroków 61. Możemy użyć innych nastaw z innym krokiem i liczbą kroków, ale w tym samym przedziale napięć. Włączamy automatyczne skalowanie na wszystkich osiach wykresu. Uruchamiamy program. Otrzymujemy wykres podobny do widocznego na rycinie 5.21. Charakterystyka przetwarzania (zielona linia) w środkowej części jest zbliżona do linii prostej. Wartości błędów nieliniowości są bardzo duże, około stukrotnie większe niż na rycinie 5.20. Jest to skutek zniekształcenia charakterystyki na początku i na końcu zakresu przeprowadzonych pomiarów.

Odczytujemy z wykresu i zapisujemy w protokole wartości napięć na wejściu i wyjściu wzmacniacza, które określają współrzędne początkowego i końcowego punktu liniowej charakterystyki przetwarzania wzmacniacza. Jest to wstępna ocena parametrów wzmacniacza, więc odczytujemy je z wykresu na tyle dokładnie, na ile jest to możliwe. Wyniki zapisujemy w tabeli 5.3. Zakładając liniową charakterystykę wzmacniacza, obliczamy odpowiednie przyrosty napięć i wyznaczamy wzmocnienie wzmacniacza oraz przesunięcie charakterystyki w zerze. Stosujemy znane wzory na wyznaczanie współczynników $a$ i $b$ linii prostej o równaniu $y = ax + b$ przechodzącej przez dwa punkty o znanych współrzędnych $(x_1, y_1)$ oraz $(x_2, y_2)$. Czy otrzymana wartość wzmocnienia jest bliska 2 V/V? W protokole zapisujemy wniosek, czy pomiary potwierdzają parametry znamionowe wzmacniacza.

**Tab. 5.3.** Wstępna ocena zakresu liniowej pracy wzmacniacza

| Napięcie | Początek | Koniec | Przyrost | Wzmocnienie | Przesunięcie |
|---|---|---|---|---|---|
| | V | V | V | V/V | V |
| Wejściowe | ... | ... | ... | ... | ... |
| Wyjściowe | ... | ... | ... | | |

## 5.6.2. Badania w liniowym zakresie pracy wzmacniacza

W tym punkcie dokładniej określamy parametry wzmacniacza w zakresie pracy liniowej. Planujemy nastawy w programie tak, aby pomiary obejmowały zakres napięć zawężony o 0,5 V na początku i na końcu względem zakresu ustalonego

zgrubnie w poprzednim punkcie. Przykładowo dla wykresu pokazanego na rycinie 5.21 planujemy pomiary w zakresie od 0,8 V do 4,8 V. Na każdym stanowisku zakres ten może być inny. Pomiary wykonujemy 10-krotnie, po czym obliczamy wartość średnią, odchylenie standardowe oraz przedział 3σ. Wyniki i obliczenia notujemy w tabeli 5.4. Wzmocnienie i przesunięcie charakterystyki odczytujemy z parametrów aproksymacji liniowej, podanych przez program.

### 5.6.3. Wyznaczenie maksymalnego zakresu liniowej pracy wzmacniacza

Maksymalny zakres pracy liniowej wzmacniacza wyznaczamy jako maksymalny zakres napięć wejściowych, w którym zmierzone wzmocnienie nie różni się o więcej niż 3σ od wartości średniej, wyznaczonej w poprzednim punkcie. Na podstawie tabeli 5.4 obliczamy i wpisujemy w tabeli 5.5 minimalną i maksymalną wartość wzmocnienia, wynikającą z kryterium 3σ.

**Tab. 5.4.** Badania wzmacniacza prądu stałego w liniowym zakresie pracy

| Parametry badania | |
|---|---|
| Napięcie początkowe | ... |
| Krok napięcia | ... |
| Liczba kroków | ... |
| Napięcie końcowe | ... |

| Pomiary | | |
|---|---|---|
| lp | Wzmocnienie | Przesunięcie |
| 1 | ... | ... |
| 2 | ... | ... |
| 3 | ... | ... |
| 4 | ... | ... |
| 5 | ... | ... |
| 6 | ... | ... |
| 7 | ... | ... |
| 8 | ... | ... |
| 9 | ... | ... |
| 10 | ... | ... |
| Średnia | ... | ... |
| Odch. Std. | ... | ... |
| 3σ | ... | ... |

Uruchamiamy program kilkakrotnie, początkowo z nastawami jak w poprzednim punkcie, zmniejszając za każdym razem napięcie początkowe. Jednocześnie zmniejsza się również napięcie końcowe, ale nie zwracamy na to uwagi. Porównujemy wyznaczone przez program wzmocnienie z wartościami granicznymi zapisanymi w tabeli 5.5. Zapisujemy najmniejszą wartość napięcia początkowego, dla której spełniony jest warunek 3σ. W ten sposób znajdujemy dolną granicę pracy liniowej wzmacniacza.

**Tab. 5.5.** Wyznaczenie maksymalnego zakresu liniowej pracy wzmacniacza

| Wzmocnienie | |
|---|---|
| Maksymalne | ... |
| Średnie | ... |
| Minimalne | ... |

| Nastawy | Początek badania | Koniec badania |
|---|---|---|
| Napięcie początkowe | ... | ... |
| Krok napięcia | ... | ... |
| Liczba kroków | ... | ... |
| Napięcie końcowe | ... | ... |

Kontynuujemy eksperyment, tym razem jednak przy każdym uruchomieniu programu zwiększamy liczbę kroków. Napięcie początkowe pozostaje więc bez zmian, zwiększa się natomiast napięcie końcowe. Podobnie jak poprzednio, porównujemy wyznaczone przez program wzmocnienie z wartościami granicznymi. W ten sposób znajdujemy napięcie końcowe określające górną granicę pracy liniowej wzmacniacza.

## 5.7. Opracowanie raportu z realizacji ćwiczenia

W raporcie z przeprowadzonego ćwiczenia zamieszczamy:

- stronę tytułową z danymi członków zespołu, datą ćwiczenia i jego tytułem,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań zgodnie z programem ćwiczenia,
- opis każdego z wykonanych zadań lub podanie przyczyny jego niewykonania,
- dokumentację zrealizowanych i uruchomionych 3 programów,

- wykresy wygenerowane przez program w środowisku LabVIEW,
- liczbowe wyniki pomiarów uzyskane z opracowanych programów,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.

Pamiętamy również, że ocenie podlegają następujące elementy:

- poprawność działania programów,
- czytelność i przejrzystość diagramu,
- estetyka paneli, dobór kolorów i elementów graficznych,
- funkcjonalność paneli pod kątem wygody obsługi,
- zakres zrealizowanych zadań.

# 6. Wyznaczanie charakterystyk prądowo-napięciowych dwójników

## 6.1. Cel ćwiczenia i zadania do wykonania

Ćwiczenie 6 jest kontynuacją ćwiczenia 5. Trzeci program opracowany w ćwiczeniu 5, służący do wyznaczania charakterystyki przetwarzania wzmacniacza prądu stałego, stanowi podstawę do rozpoczęcia bieżącego zadania. Wykorzystujemy w nim ten sam układ pomiarowy, który stosowaliśmy poprzednio. Kontynuacją ćwiczenia 6 są z kolei ćwiczenia 7 i 8.

Celem ćwiczenia 6 jest zapoznanie się z problematyką programowania systemu pomiarowego w środowisku LabVIEW z jednoczesnym wykorzystaniem przetworników cyfrowo-analogowego i analogowo-cyfrowego w dwóch kanałach.

Program laboratorium obejmuje następujące zagadnienia:

- konfigurowanie modułu NI USB-6001 do jednoczesnego sterowania przetwornikiem DAC i pomiaru napięcia przetwornikiem ADC w dwóch kanałach,
- przygotowanie programu w środowisku LabVIEW do automatycznego wyznaczania charakterystyk prądowo-napięciowych liniowych i nieliniowych dwójników pasywnych.

W ramach ćwiczenia 6 opracowujemy trzy programy:

1. Pierwszy program – do wyznaczania charakterystyk prądowo-napięciowych dwójników.

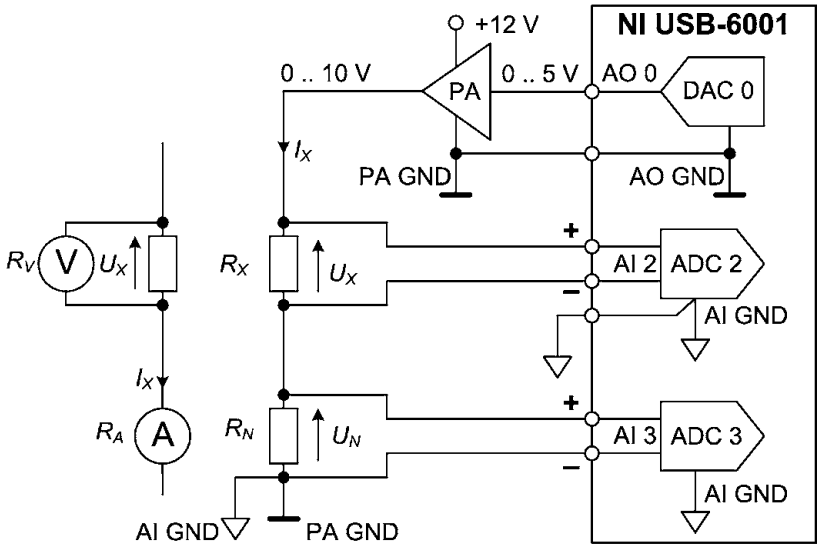
2. Drugi program – uzupełniony o obliczanie rezystancji i mocy dwójnika.
3. Trzeci program – z automatycznym zapisem wyników w pliku.

## 6.2. Opis stanowiska laboratoryjnego

### 6.2.1. Schemat blokowy stanowiska laboratoryjnego

Do realizacji ćwiczenia 6 wykorzystujemy stanowisko laboratoryjne składające się z następujących elementów:

- komputera PC z zainstalowanym środowiskiem LabVIEW,
- modułu akwizycji danych pomiarowych NI USB-6001,
- wzmacniacza mocy prądu stałego do zasilania układu pomiarowego,
- modułu z zestawem zacisków laboratoryjnych do łączenia badanych dwójników,
- zestawu pasywnych dwójników liniowych i nieliniowych do badań
- zestawu rezystorów wzorcowych.




**Ryc. 6.1.** Schemat blokowy stanowiska laboratoryjnego w ćwiczeniu 6 [1]

Schemat blokowy stanowiska laboratoryjnego wykorzystywanego w ćwiczeniu 6 przedstawiono na rycinie 6.1 [42]. Moduł akwizycji danych pomiarowych NI USB-6001 łączymy z komputerem PC za pomocą interfejsu USB.

Napięcie wyjściowe z przetwornika cyfrowo-analogowego AO 0 podajemy na wejście wzmacniacza mocy PA o wzmocnieniu równym 2 V/V. Parametry wzmacniacza zostały szczegółowo opisane w ćwiczeniu 5. Napięcie wyjściowe ze wzmacniacza zasila układ pomiarowy składający się z szeregowego połączenia badanego dwójnika $R_X$ i rezystora wzorcowego $R_N$. Do pomiarów wykorzystujemy dwa wejścia analogowe: AI 2 oraz AI 3, pracujące w trybie różnicowym. Napięcie $U_X$ na badanym dwójniku jest mierzone w kanale AI 2, a napięcie $U_N$ na rezystorze wzorcowym w kanale AI 3. Zadając na przetworniku DAC napięcia w zakresie od 0 do 5 V, na wyjściu wzmacniacza otrzymujemy napięcia od około 0 do 10 V. Wzmacniacz nie jest typu **Rail-to-Rail** i pracuje poprawnie dla napięć wejściowych powyżej około 400 mV. Zagadnienie to dokładnie zbadaliśmy w ćwiczeniu 5. Dobierając odpowiednio rezystor wzorcowy $R_N$ z przedziału od 1 Ω do 1 kΩ, możemy badać dwójniki w zakresie napięć do około 8 V i prądów do 2 A, tak aby spadek napięcia na rezystorze wzorcowym wynosił przynajmniej 2 V na końcu zakresu pomiarowego. Możemy również zwiększać wartość rezystora wzorcowego nawet do 1 MΩ i badać dwójniki wysokorezystancyjne, jednak nie jest to przewidziane w programie ćwiczenia.

W ćwiczeniu zastosowano techniczną metodę pomiaru rezystancji, zwaną również metodą amperomierza i woltomierza. Metoda ta występuje w dwóch odmianach: z poprawnie mierzonym prądem i poprawnie mierzonym napięciem. Rolę woltomierza pełni wejście analogowe AI 2, natomiast rolę amperomierza pełni rezystor wzorcowy $R_N$ z dołączonym wejściem analogowym AI 3. Analizując schemat na rycinie 6.1, zauważamy, że zastosowano wersję układu z poprawnie mierzonym napięciem. Prąd *IX* płynący w układzie obliczamy na podstawie zależności:

$$I_X = \frac{U_N}{R_N}. \tag{6.1}$$

Znając wartość prądu *IX* płynącego przez badany dwójnik oraz napięcie $U_X$ zmierzone na nim w kanale AI 2, obliczamy rezystancję dwójnika $R_X$:

$$R_X = \frac{U_X}{I_X} = \frac{U_X}{U_N} R_N, \tag{6.2}$$

oraz moc $P_X$ traconą w dwójniku:

$$P_X = U_X I_X = U_X \frac{U_N}{R_N}. \tag{6.3}$$

W technicznej metodzie pomiaru rezystancji wykorzystujemy również pojęcie rezystancji granicznej $R_{gr,}$ która pozwala nam dobrać odpowiednią wersję metody, minimalizując błędy wynikające ze stosowania niedoskonałej aparatury pomiarowej. W teorii idealny woltomierz ma nieskończenie wielką rezystancję, a amperomierz – rezystancję równą zero. Ponieważ nie istnieją idealne przyrządy pomiarowe, w rzeczywistości pojawiają się błędy pomiarowe. Metodę z poprawnie mierzonym napięciem stosujemy do pomiaru tzw. małych rezystancji, czyli wartości mniejszych od rezystancji granicznej. Rezystancja graniczna RGR jest równa pierwiastkowi z iloczynu rezystancji amperomierza $R_A$ i rezystancji woltomierza $R_{V:}$

$$R_{gr} = \sqrt{R_A R_V}. \tag{6.4}$$

Według dokumentacji modułu NI USB-6001 [6] rezystancja wejściowa toru analogowego wynosi co najmniej 1 GΩ, przyjmujemy więc, że jest to również rezystancja „woltomierza" $R_V = 1$ GΩ. Natomiast rezystancja „amperomierza" $R_A$ jest równa rezystancji wzorcowej $R_N$. Podczas pomiarów stosujemy dwa rezystory wzorcowe: 1 Ω oraz 1 kΩ. Dla $R_A = R_N = 1$ Ω otrzymujemy ze wzoru (6.4) $R_{gr} = 1$ kΩ, natomiast dla $R_A = R_N = 1$ kΩ otrzymujemy ze wzoru (6.4) $R_{gr} = 31{,}6$ kΩ. Obliczone w ten sposób wartości rezystancji granicznej wyznaczają zakresy pomiarowe zastosowanej metody.

### 6.2.2. Parametry przetworników DAC i ADC w module NI USB-6001

Budowa torów analogowego wejścia i wyjścia modułu NI USB-6001 zostały już szczegółowo opisane w ćwiczeniu 5, a w tabeli 5.1 porównano najważniejsze parametry przetworników ADC i DAC zastosowanych w tym module [6]. Na potrzeby realizacji ćwiczenia 6 przypominamy sobie tę część instrukcji do ćwiczenia 5.

Z punktu widzenia realizacji programu ćwiczenia 6 zwracamy uwagę, że maksymalny prąd wyjściowy przetwornika DAC jest bardzo mały (wynosi zaledwie ±5 mA). Ogranicza to znacząco możliwe zastosowania całego modułu NI USB-6001. Dlatego stosujemy dodatkowy wzmacniacz mocy. Natomiast prąd polaryzujący wejścia przetwornika ADC, wynoszący ±200 pA, uznajemy

za bardzo mały, dzięki czemu możemy bezpośrednio mierzyć napięcia w układzie pomiarowym bez konieczności stosowania dodatkowych wzmacniaczy buforujących.

### 6.2.3. Budowa i parametry modułu wzmacniacza mocy

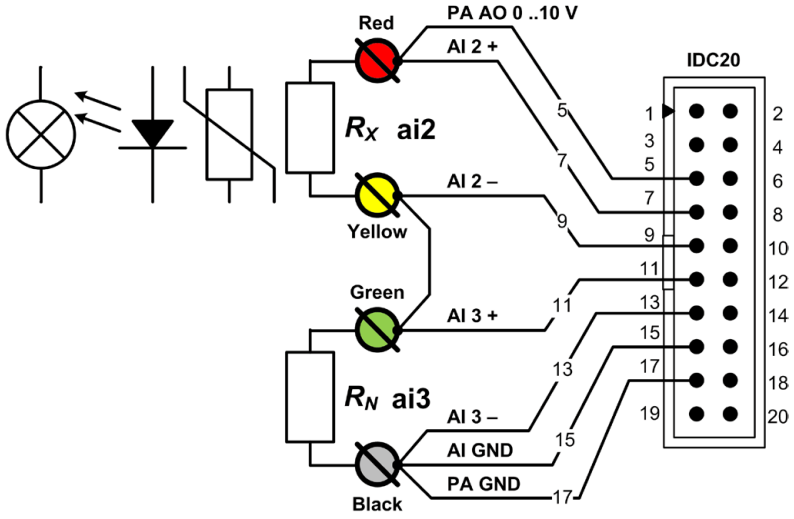
We wstępie do ćwiczenia 5 szczegółowo opisano budowę wzmacniacza mocy zastosowanego na panelu ćwiczeniowym. Podano również jego najważniejsze dane techniczne. Dodatkowo, w ramach programu ćwiczenia 5, zbadano wybrane parametry tego wzmacniacza. Jednak w ćwiczeniu 5 wzmacniacz był badany bez obciążenia, natomiast w ćwiczeniu 6 zasila on układ pomiarowy stosunkowo dużą wartością prądu. Dlatego konieczne jest dodatkowe podanie kilku istotnych parametrów wzmacniacza z punktu widzenia programu ćwiczenia 6.

Wzmacniacz zbudowany jest w oparciu o układ TDA2050 produkcji STMicroelectronics [34]. Jest to wzmacniacz akustyczny o mocy chwilowej do 50 W przy zasilaniu symetrycznym 22,5 V na obciążeniu 4 Ω. Może on również pracować z mniejszą mocą przy zasilaniu niesymetrycznym 12 V [35]. Wzmacniacz ma zabezpieczenie przeciwzwarciowe oraz termiczne. Może dostarczyć do obciążenia prąd o maksymalnej wartości nawet 2 $A_{RMS}$. Na panelu ćwiczeniowym zastosowano gotowy moduł wzmacniacza, dostępny w postaci kitu AVT 1597 do samodzielnego montażu [36]. Na potrzeby ćwiczenia odpowiednio go jednak zmodyfikowano, tak aby wzmacniał napięcia stałe DC. Do pomiarów w ramach ćwiczenia zaplanowano wzmocnienie 2 V/V dla zapewnienia możliwości współpracy ze starszymi modelami modułów NI USB-6008/6009, które mają zakres wyjścia analogowego tylko do +5 V. Dokładny schemat ideowy wzmacniacza zamieszczono i szczegółowo opisano w ćwiczeniu 5. Moduł wzmacniacza jest na stałe zamocowany na panelu ćwiczeniowym i połączony z pozostałą częścią układu, dlatego nie wymaga on wykonywania żadnych połączeń podczas realizacji ćwiczenia.

### 6.2.4. Budowa modułu do badania dwójników pasywnych

Schemat modułu do badania dwójników pasywnych przedstawiono na rycinie 6.2. Zastosowaliśmy go już w ćwiczeniu 5, jednak faktycznie przydatny był tylko podczas testowania modułu NI USB-6001. Natomiast w ćwiczeniu 6 odgrywa on istotną rolę, ponieważ umożliwia nam dołączenie do układu

pomiarowego badanego dwójnika $R_X$ oraz rezystora wzorcowego $R_N$. W tym celu na górnej powierzchni modułu zamontowane są 4 zaciski laboratoryjne w kolorach: czerwonym, żółtym, zielonym i czarnym. Zacisk czerwony połączony jest z wyjściem wzmacniacza mocy, dzięki czemu zawsze ma on potencjał dodatni względem zacisku czarnego, który z kolei połączony jest z masą. Pamiętamy o tym przy badaniu dwójników wymagających odpowiedniej polaryzacji, np. diod LED.




**Ryc. 6.2.** Schemat modułu do badania dwójników pasywnych

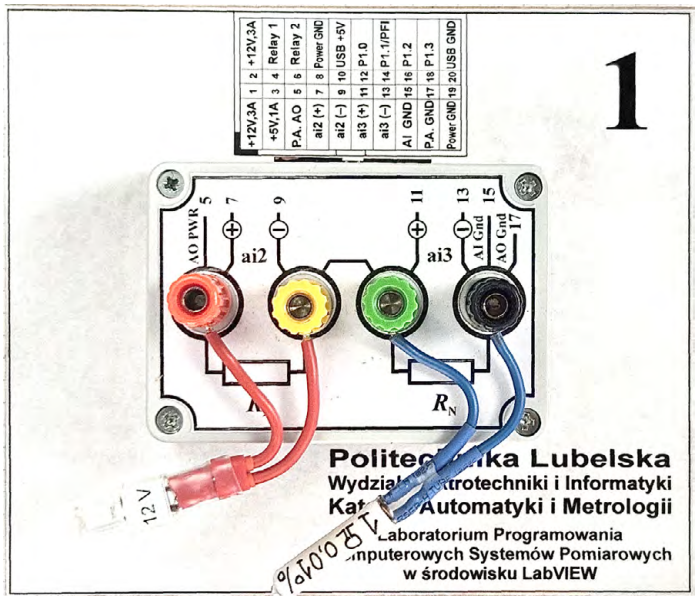
W celu zminimalizowania wpływu rezystancji przewodów na mierzone spadki napięć $U_X$ oraz $U_N$ zastosowano czteroprzewodowe połączenie Kelvina. Przewody doprowadzające prąd ze wzmacniacza mocy łączą się z przewodami obwodów napięciowych dopiero na zaciskach laboratoryjnych. Rozdzielono również obwody masy analogowej wejściowej AI GND od masy wzmacniacza mocy PA GND, którą powraca prąd zasilający obwód pomiarowy.

Na bocznej ściance modułu zamontowane jest dwurzędowe złącze szufladowe IDC20, które umożliwia połączenie modułu z panelem ćwiczeniowym. Do tego celu służy płaski przewód 20-żyłowy o długości 30 cm, zakończony wtyczkami IDC20. Rzeczywisty wygląd modułu pokazano na rycinie 6.3. Jako badany dwójnik $R_X$ dołączona jest żarówka halogenowa typu G4 (12 V, 5 W). Widoczną na rycinie 6.3 żarówkę badamy jako przykład dwójnika nieliniowego. Obok dołączony jest manganinowy rezystor wzorcowy $R_N = 1\ \Omega$, 0,01 %.

W ramach ćwiczenia 6 badamy następujące dwójniki:

1. Rezystor warstwowy metalizowany AT 1/4W, 453 Ω, 0,5 %,
2. Żarówka halogenowa G4, 12 V, 5 W,
3. Diody LED: czerwona, żółta, zielona, niebieska i biała,
4. Termistor NTC 1 kΩ przy 25°C.

Opcjonalnie możemy również zbadać własne dwójniki o parametrach prądowych i napięciowych zgodnych ze stanowiskiem pomiarowym.

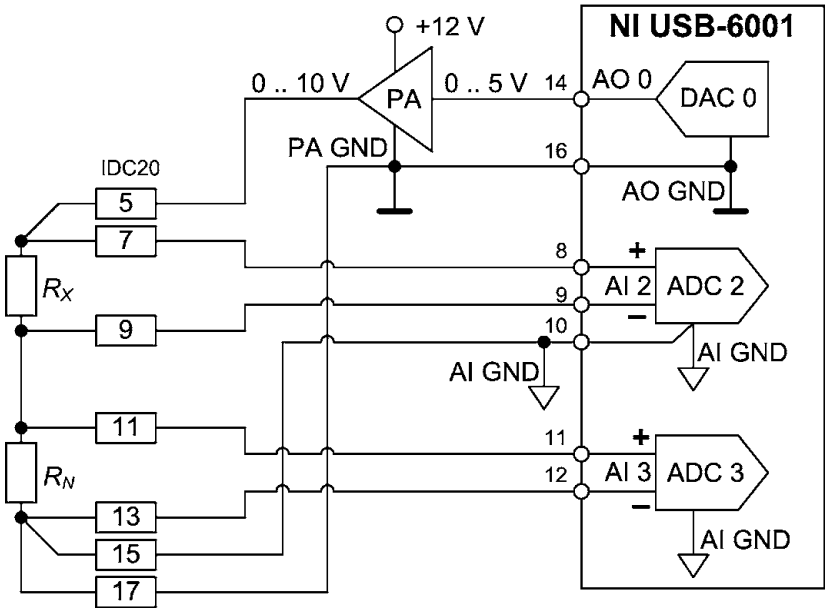



**Ryc. 6.3.** Moduł do badania dwójników pasywnych z dołączoną żarówką halogenową

## 6.3. Przygotowanie stanowiska laboratoryjnego do pracy

Schemat połączeń stanowiska laboratoryjnego do automatycznego wyznaczania charakterystyk prądowo-napięciowych dwójników pasywnych pokazano na rycinie 6.4. Jest to praktycznie ten sam układ pomiarowy omówiony już w ćwiczeniu 5. Jednak w ćwiczeniu 6 będzie on wykorzystywany inaczej. Napięcie wyjściowe z przetwornika cyfrowo-analogowego AO 0 (PIN 14) jest podawane na wejście wzmacniacza mocy. Z kolei napięcie wyjściowe ze wzmacniacza podawane jest do styku 5 na złączu IDC20, skąd trafia ono do czerwonego zacisku laboratoryjnego na module ćwiczeniowym.

Rozmieszczenie wykorzystywanych sygnałów na złączu IDC20 i ich układ połączeń z kolorowymi zaciskami laboratoryjnymi pokazano na rycinie 6.5. Układ pomiarowy, składający się z szeregowego połączenia badanego dwójnika $R_X$ i rezystora wzorcowego $R_N$, włączony jest pomiędzy czerwonym a czarnym zaciskiem laboratoryjnym. Czarny zacisk poprzez styk 17 na złączu IDC20 połączony jest z masą wzmacniacza mocy, dzięki czemu zamknięty jest obwód dla prądu pomiarowego. Ten sam czarny zacisk jest również połączony oddzielnym przewodem, poprzez styk 15 na złączu IDC20, z analogową masą wejściową modułu pomiarowego NI USB-6001 (PIN 10). Do pomiarów wykorzystujemy dwa wejścia analogowe pracujące w trybie różnicowym. Napięcie $U_X$ na badanym dwójniku mierzymy w kanale AI 2. Czerwony zacisk poprzez styk 7 na złączu IDC20 jest dołączony do wejścia AI 2+ (PIN 8). Analogicznie, żółty zacisk poprzez styk 9 na złączu IDC20 jest dołączony do wejścia AI 2- (PIN 9). Podobnie napięcie $U_N$ na rezystorze wzorcowym mierzymy w kanale AI 3. Zaciski zielony i czarny, poprzez styki 11, 13 na złączu IDC20, dołączone są odpowiednio do wejść AI 3 + (PIN 11) oraz AI 3- (PIN 12).

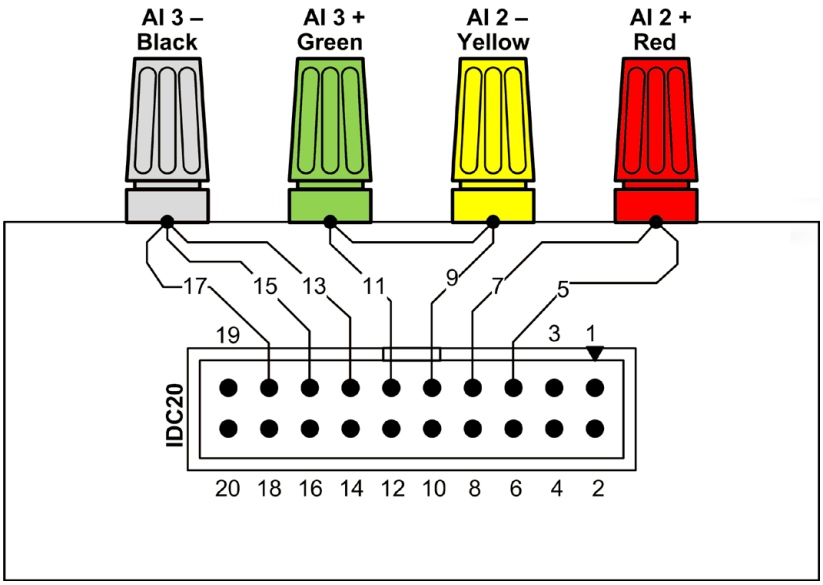



**Ryc. 6.4.** Schemat układu połączeń stanowiska do ćwiczenia 6

Stanowisko do pracy przygotowujemy, dołączając moduł z zaciskami laboratoryjnymi do panelu ćwiczeniowego za pomocą płaskiego przewodu

zakończonego wtyczkami IDC20. Zwracamy uwagę na położenie styku 1 w gniazdkach i wtyczkach IDC20. Prawidłowe połączenie ułatwia wycięcie w obudowie gniazda IDC20 w pobliżu styków 9 i 11, co można zauważyć na rycinie 6.5. Następnie do zacisków laboratoryjnych dołączamy badany dwójnik $R_X$ oraz rezystor wzorcowy $R_N$. Wartość rezystora wzorcowego dobieramy tak, aby przy napięciu 10 V nie pobierać z wyjścia wzmacniacza prądu większego niż 2 A, uwzględniając przy tym spadek napięcia na badanym dwójniku. Ze względu na błędy pomiarowe spadek napięcia na rezystorze wzorcowym $R_N$ nie powinien być mniejszy niż 1 V na końcu zakresu pomiarów. Nie możemy przy tym również przekroczyć prądu dopuszczalnego dla badanego dwójnika $R_X$. Stosując nieodpowiedni rezystor wzorcowy $R_N$, możemy już przy pierwszym pomiarze „przepalić" badaną diodę LED. Mając do dyspozycji odpowiedni zapas, możemy przeprowadzić na stanowisku również badania niszczące diody LED.

Znając przedstawione parametry stanowiska oraz wymagania dla badanego dwójnika $R_X$ i rezystora wzorcowego $R_N$, po uzgodnieniu z prowadzącym zajęcia możemy w ramach ćwiczenia 6 przebadać również inne, własne dwójniki.




**Ryc. 6.5.** Złącze IDC z zaznaczonymi stykami wykorzystywanymi w ćwiczeniu 6

## 6.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 6.4.1. Uruchamianie stanowiska

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Upewniamy się, że zasilanie komputera i panelu ćwiczeniowego jest wyłączone, a moduł pomiarowy – odłączony od portu USB komputera**.

Wszelkie zauważone odstępstwa od tego stanu zgłaszamy prowadzącemu zajęcia.

Sprawdzamy układ zgodnie z opisem w poprzednim punkcie. Jeśli połączyliśmy go prawidłowo, zgłaszamy prowadzącemu gotowość do rozpoczęcia ćwiczenia.

**Dopiero po uzyskaniu wyraźnej zgody od prowadzącego włączamy zasilanie samego komputera. Nie włączamy jeszcze zasilania panelu ćwiczeniowego ani nie dołączamy modułu pomiarowego do portu USB komputera.**

Po uruchomieniu systemu operacyjnego nie wykonujemy na razie żadnych innych czynności i przechodzimy do kolejnego punktu.

### 6.4.2. Self-test modułu NI USB-6001

Po uruchomieniu komputera w pierwszej kolejności sprawdzamy, czy w systemie mamy zainstalowane dwa urządzenia NI USB-6001: jedno fizyczne jako **Dev1** i drugie symulowane jako **Dev2**. W tym celu uruchamiamy program NI MAX, rozwijamy listę **My System > Devices and Interfaces** i sprawdzamy, jakie moduły pomiarowe są zainstalowane w systemie. Zainstalowany i prawidłowo dołączony do komputera moduł fizyczny program wyświetla jako aktywny (czarna ikona). Z kolei prawidłowo zainstalowany moduł symulowany program zawsze wyświetla jako aktywny (żółta ikona). Zainstalowany wcześniej moduł fizyczny, którego obecnie nie dołączyliśmydo komputera, program wyświetla jako nieaktywny z dodatkowym znakiem „x” na czerwonym polu. Na komputerach w laboratorium powinny mieć zainstalowane jedno aktywne urządzenie fizyczne o nazwie **Dev1** oraz drugie symulowane urządzenie o nazwie **Dev2**. W systemie mogą znajdować się również zainstalowane

inne urządzenia fizyczne, jednak powinny być nieaktywne. Pracując na domowych komputerach, instalujemy jedno symulowane urządzenie o nazwie **Dev1**. W rezultacie programy, które przygotowujemy w domu, od razu działają poprawnie w laboratorium.

W protokole notujemy nazwy, typy, numery seryjne i statusy urządzeń z serii NI USB zainstalowanych w systemie. Ponieważ moduł używany na panelu ćwiczeniowym pozostaje jeszcze odłączony, na liście widnieje przy nim czerwony znak „x" oraz status **Not Present**. Jeśli na liście nie ma wymaganych urządzeń, instalujemy je ponownie zgodnie z opisem w instrukcji do ćwiczenia 1.

Pamiętamy, że inni użytkownicy mogli z nieznanych powodów usunąć moduł zainstalowany w systemie na wcześniejszych zajęciach. Dlatego zawsze na początku zajęć upewniamy się, jaki numer urządzenia system przypisał do aktualnie używanego modułu pomiarowego. Upewniamy się również, czy na każdych zajęciach pracujemy na tym samym stanowisku laboratoryjnym, ponieważ każdy wyprodukowany moduł DAQ ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczonych w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**, jeśli więc wykorzystywany przez nas moduł otrzymał w systemie inną nazwę, odpowiednio modyfikujemy tworzone programy.

**Zgłaszamy gotowość prowadzącemu i po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera. Nie załączamy jeszcze zasilania na panelu ćwiczeniowym.** Sprawdzamy zachowanie niebieskiej kontrolki LED na obudowie modułu. Wynik obserwacji notujemy w protokole.

Jeśli używane podczas zajęć urządzenie NI USB-6001 zainstalowaliśmy poprawnie, w programie NI MAX przeprowadzamy **self-test** modułu fizycznego i symulowanego zgodnie z opisem zamieszczonym w instrukcji do pierwszego ćwiczenia. Rezultaty **self-testu** notujemy w protokole.

**Zgłaszamy gotowość prowadzącemu i po otrzymaniu zgody włączamy zasilanie panelu ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku potwierdza obecność napięcia zasilającego.

### 6.4.3. Testowanie przetwornika cyfrowo-analogowego

Test przetwornika cyfrowo-analogowego wykonujemy identycznie jak w opisie ćwiczenia 5, z dołączonym do panelu ćwiczeniowego modułem do badania dwójników. W miejsce badanego dwójnika $R_X$ dołączamy czerwoną diodę LED.

Zwracamy przy tym uwagę na biegunowość diody LED. Anoda ma dłuższe wyprowadzenie i polaryzujemy ją dodatnio względem katody (krótsze wyprowadzenie). **Anodę podłączamy do czerwonego zacisku, a katodę do żółtego. Odwrotne dołączenie może uszkodzić diodę LED!** Jako rezystor wzorcowy dołączamy $R_N = 1000\ \Omega$. **Użycie innego rezystora może przeciążyć diodę i trwale ją uszkodzić**.

**UWAGA!** Zaciski laboratoryjne dokręcamy ostrożnie i z wyczuciem, ponieważ łatwo możemy uszkodzić delikatny gwint w nakrętkach. W gwintowanej części zacisków znajduje się otwór, przez który przewlekamy dołączany przewód. Dzięki temu uzyskujemy dobry kontakt elektryczny.

W oknie programu NI MAX wybieramy na liście urządzeń **Dev1** i otwieramy zakładkę **Test Panels**. W otwartym oknie **Test Panels** wybieramy zakładkę **Analog Output**. Konfigurujemy test wyjścia analogowego, zwracając uwagę na wybór wyjścia **Dev1/ao0** w okienku **Channel Name**. W okienku **Output Value** wpisujemy wartość 0 V i uruchamiamy test, naciskając przycisk **Update**. Dioda LED nie powinna świecić, co potwierdza brak napięcia na wyjściu wzmacniacza mocy PA. Sprawdzamy, czy układ zachował się poprawnie. Następnie w okienku **Output Value** wpisujemy wartość +5 V i ponownie uruchamiamy test, naciskając przycisk **Update**. W tym momencie dioda LED powinna zaświecić, sygnalizując pojawienie się napięcia na wyjściu wzmacniacza.

Wpisując różne wartości napięcia w okienku **Output Value** i zatwierdzając je przyciskiem **Update**, sprawdzamy, czy jasność świecenia diody LED się zmienia.

W ten sposób sprawdziliśmy poprawność działania układu od przetwornika DAC aż do zacisków badanego dwójnika. Zaobserwowane rezultaty wszystkich przeprowadzonych testów notujemy w protokole.

Pozostawiamy otwarte okno **Analog Output** skonfigurowane do testowania wyjścia analogowego **Dev1/ao0** i przechodzimy do testowania przetwornika analogowo-cyfrowego.

### 6.4.4. Testowanie przetwornika analogowo-cyfrowego

Test przetwornika analogowo-cyfrowego wykonujemy identycznie jak w opisie ćwiczenia 5. Mierzymy dwa napięcia: na badanym dwójniku $R_X$ i na rezystorze wzorcowym $R_N$. Pozostawiamy otwarte okno **Test Panels**

z uruchomionym testem przetwornika cyfrowo-analogowego w aktywnej zakładce **Analog Output**. Ponownie w otwartym oknie programu NI MAX wybieramy zakładkę **Test Panels**. Otworzy się kolejne okno **Test Panels**, w którym wybieramy zakładkę **Analog Input**. Ustawiamy na ekranie obok siebie oba okna **Test Panels**. W oknie **Output Value** wpisujemy wartość 0 V i zatwierdzamy przyciskiem **Update**. Upewniamy się, że dioda LED nie świeci.

Następnie na karcie **Analog Input** wybieramy następujące opcje: **Channel Name: Dev1/ai2, Mode: Finite, Input Configuration: Differential, Max Input Limit: 10, Min Input Limit: −10, Rate (Hz): 1000, Samples To Read: 100.** Uruchamiamy test przyciskiem **Start**. Upewniamy się, że uzyskany wynik jest zbliżony do kilkuset miliwoltów. Powtarzamy test na wejściu **Dev1/ai3**. Tym razem wynik jest zbliżony do zera, z widocznymi zakłóceniami na poziomie miliwoltów. Zaobserwowane wyniki notujemy w protokole.

W okienku **Output Value** wpisujemy wartość 5 V i zatwierdzamy przyciskiem **Update**. W rezultacie dioda LED powinna zaświecić się. Ponownie w oknie na karcie **Analog Input** uruchamiamy test na wejściu **Dev1/ai2**, a następnie na wejściu **Dev1/ai3**. Otrzymane wyniki notujemy w protokole. Na świecącej czerwonej diodzie LED występuje spadek napięcia rzędu 2 V, a na rezystorze napięcie wynosi około 8 V. Razem daje to około 10 V, ponieważ wzmacniacz dwukrotnie wzmacnia napięcie z przetwornika DAC równe 5 V. Ze wzoru (6.1) łatwo obliczamy, że przez układ płynie prąd o natężeniu około 8 mA, przy którym czerwona dioda LED świeci dość intensywnie.

Na podstawie przeprowadzonych testów oceniamy sprawność całego stanowiska, a wnioski zapisujemy w protokole. Dodatkowo na potrzeby raportu wykonujemy przykładowy zrzut ekranu do pliku z przeprowadzonego testu, z widocznymi dwoma otwartymi oknami DAC i ADC. Po uzyskaniu pozytywnych wyników testu zamykamy oba okna **Test Panel** i kończymy pracę programu **NI MAX**. Uruchamiamy środowisko LabVIEW i przechodzimy do kolejnego punktu.

## 6.5. Programowanie

### 6.5.1. Pierwszy program do wyznaczania charakterystyk dwójników pasywnych

Ćwiczenie 6 jest kontynuacją ćwiczenia 5. Podstawę do jego rozpoczęcia stanowi trzeci program, który opracowaliśmy na poprzednich zajęciach. Odszukujemy ten program na dysku komputera w katalogu ćwiczenia 5 i wczytujemy go do środowiska LabVIEW. Upewniamy się, że do modułu z zaciskami laboratoryjnymi, w miejsce dwójnika $R_X$, odpowiednio dołączyliśmy czerwoną diodę LED, tak samo jak podczas testowania przetwornika DAC. Pamiętamy, że dioda LED wymaga zachowania określonej biegunowości napięcia zasilającego. Anoda ma dłuższe wyprowadzenie i polaryzujemy ją dodatnio względem katody (krótsze wyprowadzenie). **Anodę podłączamy do czerwonego zacisku, a katodę do żółtego. Odwrotne podłączenie może uszkodzić diodę LED!** Jako rezystor wzorcowy dołączamy $R_N = 1000\ \Omega$.

W programie ustawiamy następujące parametry: 51 punktów charakterystyki z krokiem napięcia 0,1 V, napięcie początkowe 0 V oraz opóźnienie czasowe 100 ms. Uruchamiamy program i upewniamy się, że działa poprawnie. Prawidłowo działający program zadaje do przetwornika DAC narastające wartości napięcia od 0 do 5 V. Na wyjściu wzmacniacza pojawia się wówczas napięcie w zakresie od 0 do 10 V. Dioda LED świeci coraz intensywniej przez 5 sekund i gaśnie po zakończeniu programu.

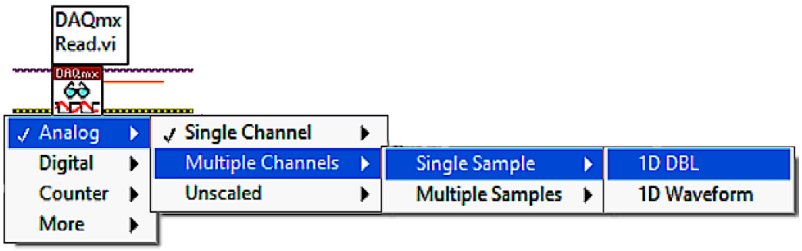
Włączamy automatyczne skalowanie na wszystkich skalach wykresu i ponownie uruchamiamy program. Sprawdzamy, czy program tworzy 3 różne wykresy. Nie mają one głębszego sensu fizycznego, ale w ten sposób upewniamy się, że program działa poprawnie, a układ generuje i mierzy odpowiednie napięcia. Weryfikujemy jedynie fakt, że napięcie mierzone na wyjściu wzmacniacza stanowi teraz spadek napięcia na czerwonej diodzie LED i osiąga maksymalną wartość około 2 V.

Jeśli program działa poprawnie, robimy jego kopię na potrzeby pierwszego zadania w ćwiczeniu 6, zmieniając numer programu na 6-1-a. Pamiętamy, że nie możemy modyfikować programu, który stanowi ostateczną wersję poprzednio wykonanego zadania.

W obecnej postaci program zadaje na przetwornik DAC napięcia z zakresu od 0 do 5 V, które po – po dwukrotnym wzmocnieniu przez wzmacniacz mocy – zasilają połączone szeregowo diodę LED i rezystor wzorcowy 1 kΩ. Jednocześnie za pomocą przetwornika ADC w kanale ai2 mierzymy napięcie na diodzie LED. Na podstawie tych danych program tworzy 3 różne wykresy, zgodnie z tematem wcześniej zrealizowanego ćwiczenia 5.

Program do wyznaczania charakterystyk prądowo-napięciowych dwójników pasywnych działa bardzo podobnie. Również kreśli on 3 różne wykresy, ale z innego zestawu danych, które będą inaczej przetwarzane. Niemniej jednak w realizowanym obecnie ćwiczeniu 6 z powodzeniem wykorzystujemy dużą część programu z ćwiczenia 5. Dlatego najpierw usuwamy z programu zbędne elementy, a następnie uzupełniamy go o nowe funkcje. Podczas modyfikacji często uruchamiamy program i upewniamy się, czy nie usunęliśmy z niego zbyt dużej części diagramu. Docelową postać programu realizowanego w tym punkcie pokazano na rycinie 6.8. Odpowiednie zmiany wprowadzamy do programu stopniowo, w opisanej poniżej kolejności.

Przełączamy się na diagram i usuwamy z niego funkcje **Linear Fit**, **General Polynomial Fit** oraz dołączone do nich zadajniki i wskaźniki. Usuwamy również funkcję odejmowania połączoną z **Linear Fit**. Nie usuwamy jednak żadnej z funkcji **Bundle**, **Build Array** ani **XY Graph**. Po skasowaniu wymienionych obiektów tylko jedna, górna funkcja **Bundle** ma podłączone dwa wejścia. Wejścia pozostałych funkcji **Bundle** łączymy w taki sam sposób, do tych samych linii danych. Uruchamiamy program. W rezultacie kreśli on trzy nałożone na siebie wykresy. Sprawdzamy to, zmieniając skalę na jednej z pionowych osi i ustawiając różne rodzaje linii i punktów w zakładce **Properties**.




**Ryc. 6.6.** Konfigurowanie funkcji DAQmx Read do odczytu dwukanałowego

W kolejnym kroku zmieniamy konfigurację przetwornika ADC tak, aby wykonywał on pomiary w dwóch kanałach wejściowych. Prawidłowy sposób konfiguracji funkcji **DAQmx Read** pokazano na rycinie 6.6. Rozwijamy menu **Polymorphic VI Selector** i wybieramy opcję **Analog > Multiple Channels > Single Sample > 1D DBL**. Na wyjściu **data** zauważymy wówczas błędne połączenie, wynikające z niezgodności typów danych. Tymczasowo kasujemy to połączenie. Przywrócimy je w dalszej części instrukcji.

Rozciągamy pętlę **While** w dół, aby przygotować miejsce na dodatkowe funkcje. Docelową postać algorytmu pokazano na rycinie 6.8. W wolnym miejscu wstawiamy do pętli **While** funkcję **Programming > Array > Index Array** i rozciągamy ją w dół, aby uzyskać dwa wejścia indeksujące. Funkcji tej używaliśmy już w ćwiczeniu 2 podczas wyszukiwania miejsca zerowego w tablicy próbek napięcia chwilowego sieci energetycznej. Na górnym wejściu indeksującym tworzymy stałą numeryczną o wartości 1, a na dolnym o wartości 0. Wejście tablicy łączymy z wyjściem **data** funkcji **Read**. Górne wyjście funkcji **Index Array** łączymy z dolnym wejściem przełącznika **Select**, z którego wcześniej usunęliśmy błędne połączenie. To połączenie przekazuje wyniki pomiaru napięcia na badanym dwójniku $R_X$. Na dolnym wyjściu funkcji **Index Array** otrzymujemy wyniki pomiaru napięcia na rezystorze wzorcowym $R_N$, które przeliczamy na wartość prądu. Na diagramie umieszczamy funkcję dzielenia **Divide** i na wejściu mianownika tworzymy zadajnik numeryczny (**Create Control**) o nazwie „RN". Na panelu wpisujemy początkową wartość 1000, odpowiadającą rezystancji wzorcowej w omach. Na wyjściu funkcji dzielenia tworzymy wskaźnik numeryczny (**Create Indicator**) o nazwie **Current [A]**. Wejście licznika funkcji **Divide** łączymy z dolnym wyjściem funkcji **Index Array**. Kasujemy połączenia na dolnych wejściach funkcji **Build Array**, które dotychczas budowały tablice danych dla osi X i Y wykresów. Zgodnie z konfiguracją górna tablica zawiera dane dla osi X, a dolna dla osi Y. Obecnie oś X tworzą wyniki pomiaru napięcia na dwójniku, natomiast oś Y – wartości prądu. Zmieniamy opisy przy połączeniach tych tablic na **Voltage** (górna linia) i **Current** (dolna linia). Do wolnych, dolnych wejść funkcji **Build Array** dołączamy sygnał napięcia z wyjścia selektora i sygnał prądu z wyjścia funkcji **Divide**. Choć diagram pozostaje niekompletny, w tej postaci uruchamiamy program testowo, aby sprawdzić poprawność dotychczasowych połączeń.

Uruchamiamy program i sprawdzamy, czy działa. Niestety, w trybie pomiaru wykresy nie mają żadnego logicznego sensu. Wynika to z faktu, że wszystkie wyniki pomiarów pochodzą z tego samego kanału fizycznego AI 2, w którym mierzymy napięcie na dwójniku *RX*. Natomiast w trybie symulacji otrzymujemy wykresy podobne do wcześniejszych, ponieważ oś X otrzymuje symulowane wartości napięcia. Z kolei wartość prądu program oblicza na podstawie napięcia na dwójniku *RX*, a więc na razie zupełnie niepoprawnie. Dlatego teraz konfigurujemy kolejny kanał fizyczny AI 3, w którym mierzymy napięcie na rezystorze wzorcowym *RN*.

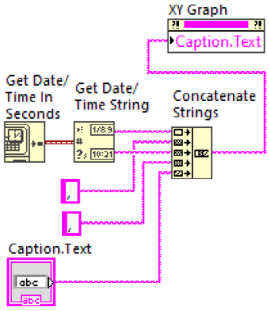
Kasujemy linię błędu na wejściu dolnej funkcji **Create Virtual Channel**, tworzącej kanał wejścia analogowego ai2. Zaznaczamy tę funkcję wraz ze wszystkimi zadajnikami i kopiujemy ją poniżej na diagramie. W utworzonej kopii funkcji **Create** zmieniamy konfigurację wejścia **physical channel** na **Dev1/ai3**. Ponownie łączymy linię błędu **error –** najpierw do dolnej funkcji **Create**, a następnie do górnej. Analogicznie łączymy wyjście **task out** z wejściem **task in**.

Uruchamiamy program w trybie pomiaru. Teraz otrzymujemy już poprawną charakterystykę prądowo-napięciową dla czerwonej diody LED, która zaczyna przewodzić prąd przy napięciu na złączu około 1,8 V. Przy spadku napięcia rzędu 2 V na diodzie LED, na rezystorze wzorcowym występuje spadek napięcia około 8 V, a zatem maksymalny prąd osiąga wartość około 8 mA. Sprawdzamy, czy otrzymaliśmy wykres zgodny z oczekiwaniami.

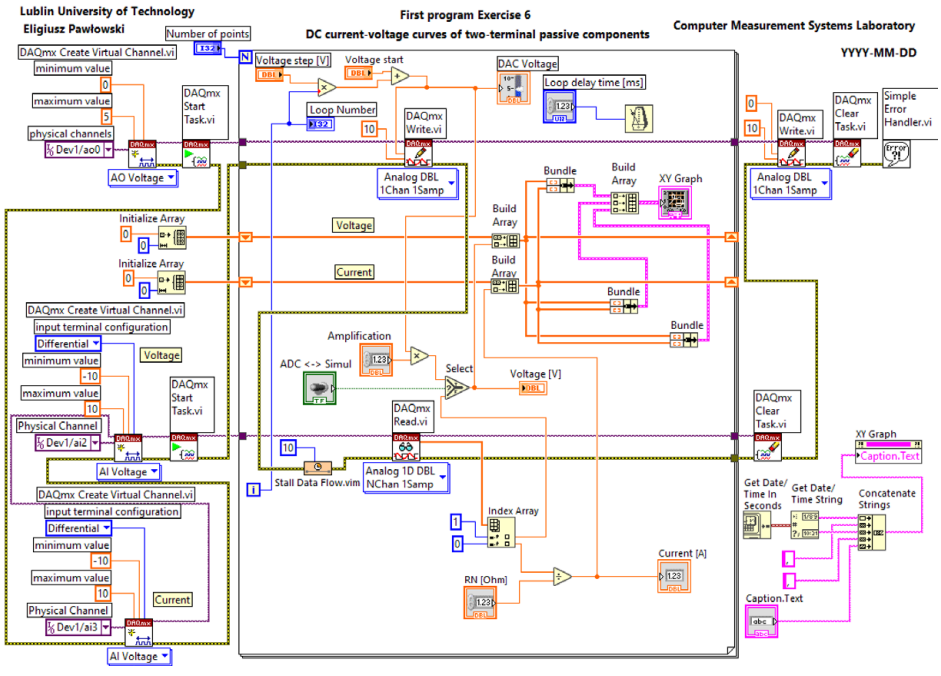
W rzeczywistości otrzymaliśmy trzy identyczne wykresy nałożone na siebie. Wchodzimy w opcję **Properties** wykresu. W zakładce **Scales** wyłączamy tymczasowo widoczność skali Y dla błędów nieliniowości. Dla tej skali tymczasowo wyłączamy również siatkę **Grid Style and Colors**. Pozostałym skalom X i Y nadajemy nazwy **Voltage [V]** i **Current [A]**. Z kolei w zakładce **Plots** tymczasowo wyłączamy widoczność wykresów błędów i ich aproksymacji.

Uruchamiamy program i oceniamy, czy wykres spełnia nasze oczekiwania. Dodajemy jeszcze do niego opis badanego obiektu oraz datę i czas pomiarów pobierane z systemu komputera. Opis ten stanowi element wykresu, dzięki czemu możemy go wyeksportować za pomocą opcji **Export Simplified Image**. W tym celu wykorzystujemy obiekt **Caption** wykresu. Na panelu prawym kliknięciem na wykresie rozwijamy menu i zaznaczamy opcję **Visible Items > Caption**. Widoczny teraz opis wykresu **Caption** przesuwamy w dobrze

widoczne miejsce w obszarze wykresu. Chcemy automatycznie wpisywać do tego obiektu datę, czas i opis badanego dwójnika. W tym celu wykorzystujemy odpowiedni obiekt, nazywany węzłem właściwości **Property Node**. Węzeł właściwości **Property Node** stosowaliśmy już w ćwiczeniu 4 do zmiany wejścia licznika impulsów. Teraz pełni on inną funkcję. Na diagramie prawym kliknięciem na ikonie wykresu **XY Graph** rozwijamy menu, wybieramy opcję **Create > Property Node > Caption > Text** i umieszczamy obiekt w przestrzeni roboczej. W ten sposób tworzymy fragment algorytmu przedstawiony na rycinie 6.7.




**Ryc. 6.7.** Dodawanie daty, czasu i opisu do wykresu

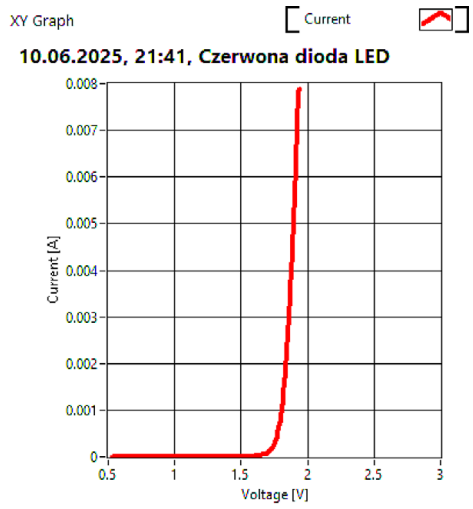



**Ryc. 6.8.** Diagram programu do wyznaczania charakterystyki prądowo-napięciowej

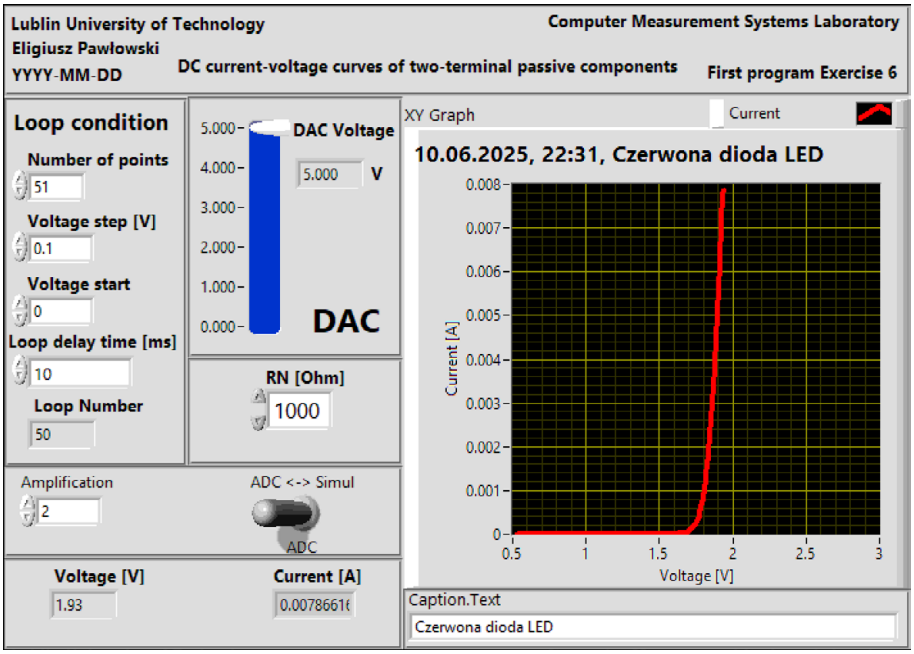
Co ciekawe, ta część diagramu nie jest połączona z pozostałą i możemy ją wstawić w dowolnym miejscu. Umieszczamy ją jednak poza pętlą **While**, tak jak na rycinie 6.8. Prawym kliknięciem na dolnej części węzła **Property Node** wybieramy polecenie **Change to Write**. Do tego wejścia podłączamy tekst dla opisu **Caption**. Na diagramie umieszczamy funkcje przetwarzające datę i czas: **Programming > Timing > Get Date/Time In Seconds** oraz **Get Date/Time String**. Opis wykresu zestawiamy w jedną całość za pomocą funkcji łączenia łańcuchów znaków **Programming > String > Concatenate Strings**, którą rozciągamy w dół, aby uzyskać 5 wejść. Wyjście tej funkcji łączymy z wejściem **Caption.Text** węzła **Property Node**. Do pierwszego i trzeciego wejścia funkcji **Concatenate Strings** podłączamy datę i czas. Z kolei na drugim i czwartym wejściu tworzymy stałe tekstowe zawierające przecinek i spację. Na piątym wejściu tworzymy zadajnik tekstowy, po czym znajdujemy go na panelu i ustawiamy w dobrze widocznym miejscu. Nadajemy mu nazwę **Caption Text** i wpisujemy do niego opis badanego dwójnika, np. „Czerwona dioda LED". Przykładowy wygląd gotowego diagramu pokazano na rycinie 6.8.

W programie ustawiamy parametry do zmierzenia charakterystyki czerwonej diody LED: liczba kroków 51, krok napięcia 0,1 V, napięcie początkowe 0 V oraz opóźnienie 100 ms. Upewniamy się, że dołączyliśmy rezystor wzorcowy 1000 Ω i taką samą wartość wprowadziliśmy do programu. Włączamy automatyczne skalowanie osi X i Y, a następnie uruchamiamy program. Sprawdzamy, czy podczas pomiaru badana dioda zaświeciła się, po czym oceniamy otrzymany wykres. Spadek napięcia na czerwonej diodzie LED osiąga wartość około 2 V przy prądzie rzędu 8 mA. Z kolei na wykresie w polu **Caption** pojawia się opis zawierający datę, godzinę, minutę i nazwę badanego dwójnika. Jeśli wykres jest zgodny z oczekiwaniami, wyłączamy automatyczne skalowanie osi i ręcznie dobieramy zakres obu skal. Skalę osi Y konfigurujemy tak, aby zaczynała się od wartości 0 i obejmowała ostatnią, największą wartość prądu. Natomiast skalę osi X dobieramy w taki sposób, aby dokładnie ocenić napięcie, przy którym dioda zaczyna przewodzić prąd. Otrzymaną charakterystykę zapisujemy do pliku graficznego BMP poleceniem **Export Simplified Image** i zamieszczamy w sprawozdaniu. Przykładową charakterystykę czerwonej diody LED, wyznaczoną opisywanym programem, pokazano na rycinie 6.9. Zauważamy, że pionowa oś prądu jest wyskalowana w amperach, podczas gdy prąd płynący przez diodę LED osiąga wartość zaledwie kilku miliamperów.

Nie zmieniamy jednak skali na tej osi, ponieważ inne badane dwójniki pobierają znacznie większe prądy, a tworzony program z założenia ma uniwersalny charakter.




**Ryc. 6.9.** Charakterystyka czerwonej diody LED z opisem, datą i godziną




**Ryc. 6.10.** Panel programu do wyznaczania charakterystyki prądowo-napięciowej dwójników

Jeśli program działa prawidłowo i wygenerował poprawną charakterystykę prądowo-napięciową czerwonej diody LED, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich wzajemne logiczne związki. Przykładowy panel tak uzupełnionego programu przedstawiono na rycinie 6.10.

Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Odszukujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików wraz z opisem zawartości wpisujemy do protokołu. Zapisane pliki wykorzystamy następnie do sporządzenia sprawozdania.

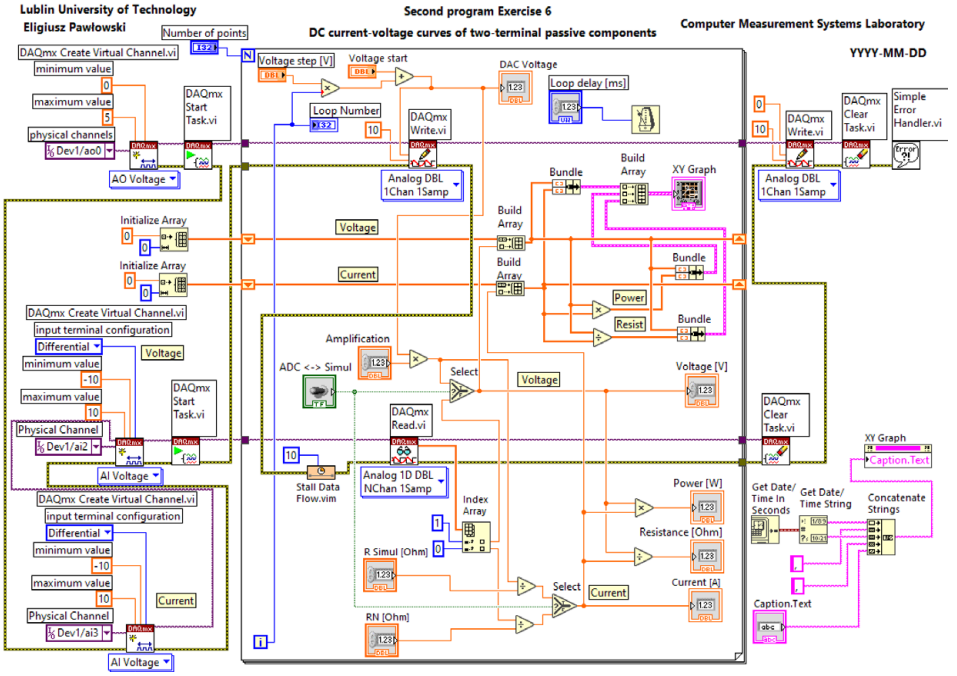
Na potrzeby realizacji następnego zadania tworzymy kopię programu, zmieniając jego numer na kolejny, tzn. 6-2-a. Zgodnie z zasadami nie modyfikujemy programu, który stanowi ostateczną wersję poprzednio wykonanego zadania. Pamiętamy, aby po zapisaniu ostatecznej wersji programu, w pełni realizującego dany punkt ćwiczenia, zawsze utworzyć jego kopię, nadając mu nową nazwę poprzez zmianę ostatniego znaku na kolejną wartość.

### 6.5.2. Drugi program obliczający moc i rezystancję dwójników pasywnych

W drugim programie, na podstawie zmierzonych wartości prądu i napięcia, dodatkowo obliczamy moc traconą w badanym dwójniku oraz jego rezystancję. Wykresy mocy i rezystancji umieszczamy razem z wykresem prądu na tym samym obiekcie graficznym XY Graph. Dodatkowo uzupełniamy program o symulację prądu, dzięki czemu możemy sprawdzić poprawność obliczania mocy i rezystancji. Docelową postać diagramu zmodyfikowanej wersji programu przedstawia rycina 6.11.

Uruchamiamy zapisaną w poprzednim punkcie kopię programu 6-2-a i upewniamy się, czy działa ona prawidłowo. W pierwszej kolejności uzupełniamy diagram o symulację prądu, wzorując się na fragmencie generującym symulowane wartości napięcia. Symulowaną wartość prądu obliczamy poprzez podzielenie symulowanego napięcia przez symulowaną rezystancję. Na diagramie umieszczamy funkcję **Divide** (zaraz powyżej funkcji obliczającej

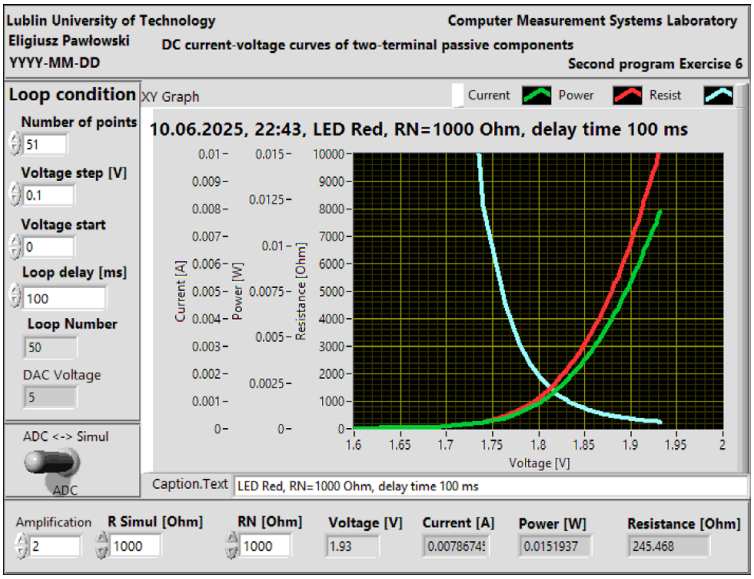
zmierzony prąd), a na jej dolnym wejściu tworzymy zadajnik (**Create Control**) o nazwie **R Simul.** [Ohm] i początkowej wartości 1000. Górne wejście funkcji łączymy z napięciem symulowanym, dostępnym na górnym wejściu przełącznika **Select**. W ten sposób otrzymujemy symulowaną wartość prądu dla przyjętej rezystancji. Na diagramie odszukujemy funkcję **Divide** obliczającą rzeczywisty prąd i kasujemy połączenie na jej wyjściu. Wstawiamy w tym miejscu drugi przełącznik **Select** i łączymy ze sobą wejścia sterujące obu selektorów, tak aby przełączały się one jednocześnie. Uzupełniamy połączenia przełącznika **Select** zgodnie z ryciną 6.11.




**Ryc. 6.11.** Diagram programu uzupełniony o obliczanie mocy i rezystancji

Po przełączeniu na panel odszukujemy zadajnik symulowanej rezystancji i umieszczamy go w widocznym miejscu. Przykład uzupełnionego panelu pokazano na rycinie 6.12. Wyłączamy automatyczne skalowanie na wszystkich osiach, przełączamy program w tryb symulacji i go uruchamiamy. Program generuje wówczas liniowy wykres o nachyleniu 1 mA/V. Następnie przełączamy go w tryb pomiarów i ponownie uruchamiamy. W tym przypadku program wykreśla charakterystykę czerwonej diody LED, która pozostaje dołą-

czona do układu. Wyniki testowania programu i wnioski notujemy w protokole.




**Ryc. 6.12.** Panel programu uzupełnionego o wykres mocy i rezystancji

Jeśli program działa prawidłowo, zapisujemy go na dysku i tworzymy jego kopię o nazwie 6-2-b na potrzeby dalszych modyfikacji. Teraz uzupełniamy program o obliczanie mocy i rezystancji. Na diagramie umieszczamy funkcję mnożenia i prawym kliknięciem tworzymy na jej wyjściu wskaźnik (**Create Inicator**), któremu nadajemy nazwę **Power [W]**. W pobliżu umieszczamy funkcję dzielenia i podobnie tworzymy na jej wyjściu wskaźnik (**Create Inicator**) o nazwie **Resistance [Ohm]**. Wejścia obu funkcji łączymy z wyjściami przełączników **Select** w taki sposób, aby zgodnie ze znanymi podstawami elektrotechniki obliczyć rezystancję i moc według wzorów (6.2) i (6.3). Uruchamiamy program w trybie symulacji i sprawdzamy poprawność obliczeń. Jeśli ostatnią symulowaną wartością napięcia jest 10 V, wartość prądu wynosi 10 mA, moc 100 mW, a rezystancja 1000 Ω. Ustawiamy opóźnienie na 1000 ms, aby łatwiej śledzić działanie programu. Ponownie uruchamiamy program. Pierwsza wyznaczona wartość rezystancji to **NaN** (z ang. **Not-a-Number**), co w środowisku LabVIEW sygnalizuje w tym przypadku próbę dzielenia zera przez zero. Wszystkie z kolejnych 50 wartości wyznaczonej rezystancji wynoszą dokładnie 1000 Ω.

Jeśli program działa prawidłowo, zapisujemy go na dysku i tworzymy jego kopię o nazwie 6-2-c na potrzeby dalszych modyfikacji. Teraz uzupełniamy program o tworzenie wykresów mocy i rezystancji. Ten fragment programu jest już prawie gotowy, ponieważ w obecnej postaci generuje on 3 wykresy, ale z tego samego zestawu danych. W tym celu generujemy nowy zestaw danych, podłączamy je do odpowiednich osi i konfigurujemy parametry wszystkich trzech wykresów. Najpierw uzupełniamy diagram o obliczanie mocy i rezystancji podobnie jak w poprzednim kroku, jednak tym razem obliczenia wykonujemy na tablicach. Na diagramie znajdują się 3 funkcje **Bundle**. Pierwsza od góry tworzy wykres prądu – tej części diagramu nie modyfikujemy. Górne wejścia wszystkich funkcji **Bundle** generują oś X, jednakową dla poszczególnych wykresów. Tych połączeń również nie modyfikujemy. Kasujemy jedynie połączenia na dolnych wejściach dwóch pozostałych funkcji **Bundle**. Środkowa funkcja **Bundle** generuje wykres mocy, a dolna – wykres rezystancji. W kolejnym kroku przystępujemy do obliczania mocy i rezystancji na danych tablicowych. Przy środkowej funkcji **Bundle** umieszczamy na diagramie funkcję mnożenia i łączymy jej wyjście z dolnym wejściem środkowej funkcji **Bundle**. Z kolei przy dolnej funkcji **Bundle** umieszczamy funkcję dzielenia i analogicznie łączymy jej wyjście z dolnym wejściem dolnej funkcji **Bundle**. Wejścia funkcji mnożenia i dzielenia łączymy z odpowiednimi tablicami napięcia i prądu w taki sposób, aby zgodnie ze wzorami (6.2) i (6.3) wyznaczyć rezystancję i moc.

Uruchamiamy program w trybie symulacji i sprawdzamy poprawność tworzenia wykresów. Niestety, na ekranie widzimy tylko wykres prądu. Program generuje również wykresy mocy i rezystancji, jednak ze względu na bieżącą konfigurację pozostają one ukryte. Prawym kliknięciem na obszarze wykresu wchodzimy w opcję **Properties**. Najpierw w zakładce **Scales** włączamy widoczność opisów (**Show scale label**) i widoczność samej skali (**Show scale**) dla wszystkich czterech osi. Początkowo włączamy również automatyczne skalowanie. Modyfikujemy odpowiednio nazwy wszystkich skal zgodnie z ich przeznaczeniem. Następnie w zakładce **Plots** włączamy widoczność wszystkich trzech wykresów. Zgodnie z przeznaczeniem modyfikujemy nazwy wszystkich wykresów (**Current**, **Power**, **Resistance**) zgodnie z ich przeznaczeniem. Ostatecznie dla każdego wykresu wybieramy odpowiednią skalę X i Y. Wykresy widnieją na rozwijanej liście od góry do dołu w tej samej kolejności, w jakiej podłączamy dane do wejść funkcji **Build Array**, łączącej oddzielne przebiegi w jeden obiekt

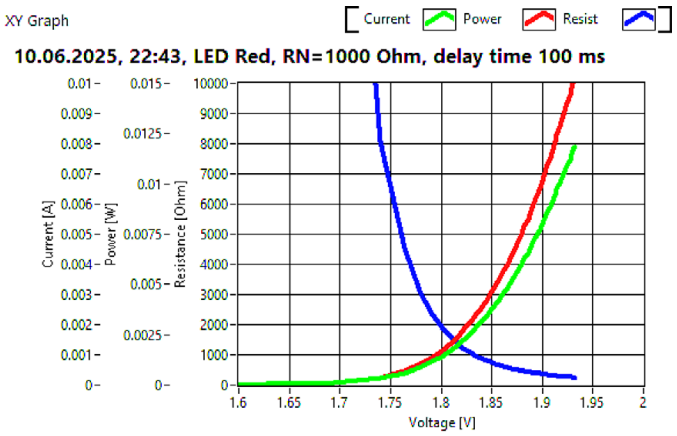
graficzny **XY Graph**. Dlatego sama zmiana nazwy wykresu nie ma wpływu na sposób jego kreślenia. Istotna jest tylko kolejność połączeń na diagramie.

Na koniec ustalamy kolory, style i grubości linii oraz punktów. Zachowujemy przy tym umiar i dbamy o estetykę, ponieważ nie wszystkie kombinacje kolorów są dobrze widoczne na monitorze, a szczególnie w raporcie wygenerowanym do formatu PDF. Przykładowy panel po niezbędnych modyfikacjach pokazano na rycinie 6.12. Uruchamiamy program w trybie symulacji i sprawdzamy poprawność tworzenia wykresów. Prawidłowy wykres prądu jest linią prostą przechodzącą przez początek układu współrzędnych i ma nachylenie 1 mA/V. Z kolei wykres mocy przybiera kształt paraboli, tzn. jeśli dla napięcia 5 V moc wynosi 25 mW, to dla napięcia dwukrotnie większego (10 V) jest ona czterokrotnie większa (100 mW). Oś Y dla rezystancji ustawiamy na zakres od 999 do 1001. Wykres rezystancji stanowi wówczas linię prostą równoległą do osi X przyjmującą stałą wartość 1000.

Jeśli program działa prawidłowo i wygenerował poprawnie trzy wykresy, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit>Make Current Values Default** i zapisujemy program do pliku. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Kasujemyz panelu liniowy wskaźnik napięcia z przetwornika DAC, pozostawiamy tylko wskaźnik numeryczny. Następnie przesuwamy pozostałe wskaźniki i zadajniki, aby wygospodarować więcej miejsca na wykresy. Ponieważ wykres stanowi najważniejszy element panelu, nadajemy mu możliwie największy rozmiar. Dbamy jednak o to, aby wszystkie obiekty na panelu były widoczne na ekranie, bez konieczności przewijania obrazu. Na koniec porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich wzajemne, logiczne związki.

W polu **Caption.Text** wpisujemy krótki opis badanej diody i konfiguracji programu. Uruchamiamy program w trybie pomiarów i sprawdzamy, czy generuje on wykresy zgodnie z oczekiwaniami. Upewniamy się przy tym, że na wykresie widnieje odpowiedni opis oraz prawidłowa data i godzina. Przykładową postać wykresów prądu, mocy i rezystancji dla czerwonej diody LED pokazano na rycinie 6.13. Wyłączamy automatyczne skalowanie osi i dobieramy ich zakresy ręcznie, aby uzyskać przejrzysty wykres. Otrzymany wykres zapisujemy do pliku i umieszczamy w raporcie. Sprawdzamy, czy na zapisanym do pliku wykresie widać również opis z datą i godziną. Zauważamy, że pionowe osie

prądu, mocy i rezystancji są wyskalowane odpowiednio w amperach, watach i omach, podczas gdy prąd płynący przez diodę LED osiąga wartość zaledwie kilku miliamperów, moc jest rzędu kilkudziesięciu miliwatów, a rezystancja sięga wielu kiloomów. Nie zmieniamy jednak skali na tych osiach, ponieważ inne badane dwójniki pobierają większe prądy i moce przy mniejszych wartościach rezystancji. Dzięki temu nasz program zachowuje uniwersalny charakter.




**Ryc. 6.13.** Przykładowy wykres prądu, mocy i rezystancji dla czerwonej diody LED z opisem, datą i godziną, rezystor wzorcowy 1000 Ω, czas 100 ms, prąd diody ograniczony do 8 mA

Wyłączamy automatyczne skalowanie osi i dobieramy ich zakresy ręcznie, aby uzyskać przejrzysty wykres. Otrzymany wykres zapisujemy do pliku i umieszczamy w raporcie. Sprawdzamy, czy na zapisanym do pliku wykresie widać również opis z datą i godziną. Zauważamy, że pionowe osie prądu, mocy i rezystancji są wyskalowane odpowiednio w amperach, watach i omach, podczas gdy prąd płynący przez diodę LED osiąga wartość zaledwie kilku miliamperów, moc jest rzędu kilkudziesięciu miliwatów, a rezystancja sięga wielu kiloomów. Nie zmieniamy jednak skali na tych osiach, ponieważ inne badane dwójniki pobierają większe prądy i moce przy mniejszych wartościach rezystancji. Dzięki temu nasz program zachowuje uniwersalny charakter.

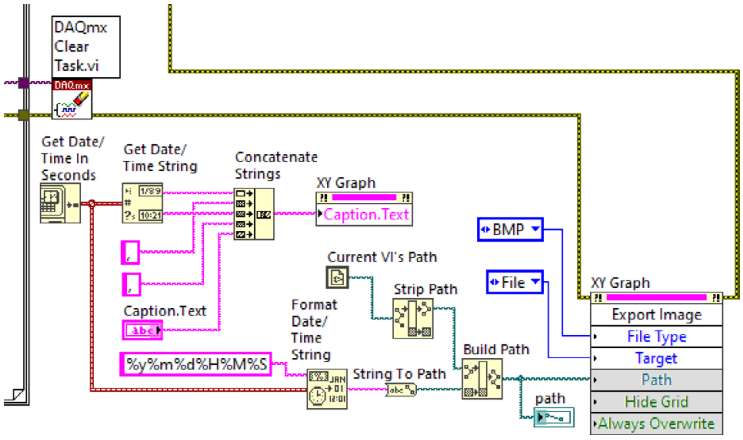
Zapisujemy program na dysku i notujemy jego nazwę w protokole. Dokumentację programu drukujemy do pliku. Odszukujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwy plików z opisem ich zawartości wpisujemy do raportu. Pliki z zapisanym widokiem panelu i diagramu

wykorzystujemy podczas opracowywania raportu. Na potrzeby realizacji następnego zadania tworzymy kopię programu, zmieniając jego numer na kolejny, czyli 6-3-a.

### 6.5.3. Trzeci program z automatycznym zapisywaniem wyników do pliku

Zapisywanie wykresów do pliku graficznego BMP za pomocą opcji **Export Simplified Image** jest uciążliwe i kłopotliwe. Za każdym razem musimy wybrać format pliku, wskazać katalog do zapisu i wprowadzić nazwę pliku. Teraz automatyzujemy te czynności. Uzupełnimy algorytm w taki sposób, aby po wykonaniu pomiarów i wygenerowaniu wykresu program automatycznie zapisywał go do pliku.

Aby program samoczynnie zapisywał wykres do pliku, w algorytmie programujemy kolejno następujące czynności: 1) generujemy ścieżkę dostępu do pliku, 2) generujemy unikalną nazwę pliku, 3) definiujemy format pliku, 4) wskazujemy źródło danych do zapisu, 5) wywołujemy funkcję zapisującą dane do pliku. Wszystkie te zadania realizuje fragment diagramu przedstawiony na rycinie 6.14.




**Ryc. 6.14.** Fragment diagramu zapisujący wykres do pliku graficznego BMP

Ścieżka dostępu jest identyczna ze ścieżką pliku uruchomionego programu. Wyniki pomiarów zapisujemy więc w tym samym katalogu, w którym znajduje się nasz program. Możemy to jednak łatwo zmodyfikować, wskazując inną ścieżkę – względną lub bezwzględną. Nazwę pliku docelowego

tworzymy na podstawie aktualnej daty i godziny wykonania pomiarów. Sam wykres zapisujemy w formacie bitmapy BMP za pomocą węzła wywołania metody **Invoke Node**.

Kolejność postępowania jest następująca: na panelu, prawym kliknięciem na ikonie wykresu XY Graph rozwijamy menu, wybieramy opcję **Create > Invoke Node > Export Image** i umieszczamy wybrany węzeł **Invoke Node** na diagramie po prawej stronie pętli **For**. Kasujemy linię błędu na wyjściu **error out** funkcji **DAQmx Clear Task** i dołączamy w to miejsce wejście **error in** umieszczonego przed chwilą węzła **Invoke Node**. Na jego wejściu **File Type** prawym kliknięciem tworzymy stałą (**Create Constant**) i z rozwijanej listy wybieramy pozycję **BMP**. Analogicznie, na wejściu **Target** tworzymy stałą (**Create Constant**) i wybieramy pozycję **File**. Wejścia **Hide Grid** oraz **Always Overwrite** pozostawiamy niepodłączone. Do wejścia **Path** dołączamy ścieżkę dostępu do zapisywanego pliku. W tym celu z palety **Functions > Programming > File I/O** pobieramy i umieszczamy na diagramie obok siebie następujące funkcje: **File Constants > Current VI's Path**, **Strip Path** oraz **Build Path**. Łączymy je ze sobą zgodnie z ryciną 6.14. Na diagramie umieszczamy również funkcję formatującą tekst z datą i czasem: **Functions > Programming > Timing > Format Date/Time String**. Na wejściu **time format string** tworzymy stałą tekstową (**Create Constant**) i nadajemy jej wartość **%y%m%d%H%M%S**, co powoduje wygenerowanie łańcucha znaków zawierającego kolejne cyfry bieżącej daty i czasu co do sekundy. Posłuży on jako unikalna nazwa pliku do zapisu wykresu. Otrzymany tekst zamieniamy na ścieżkę dostępu za pomocą funkcji **Functions > Programming > String > Path/Array/String Conversion > String To Path** i dołączamy do wejścia funkcji **Build Path**. Na jej wyjściu tworzymy wskaźnik (**Create Indicator**), po czym odnajdujemy go na panelu i umieszczamy pod wykresem (przykład pokazano na rycinie 6.16). Na koniec uzupełniamy na diagramie brakujące połączenia zgodnie z ryciną 6.14.

Zapisujemy program na dysku i go uruchamiamy. Następnie sprawdzamy, czy na panelu programu widnieje aktualna data i godzina. Weryfikujemy również, czy wyświetla się poprawna ścieżka dostępu do pliku z wykresem. Kontrolujemy zawartość katalogu z naszym programem i sprawdzamy, czy znajduje się w nim plik graficzny w formacie BMP, którego nazwa składa się z daty i czasu wykonania pomiarów. Upewniamy się, że wygenerowany plik

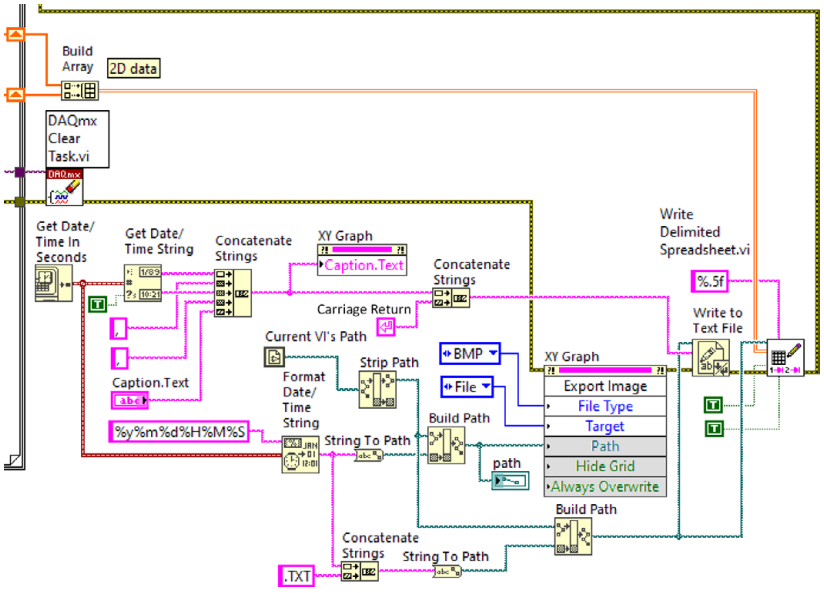
zawiera wykres zgodny z tym widocznym na panelu programu, a w szczególności – że uwzględnia on opis pomiarów wraz z datą i godziną. Na koniec notujemy w protokole nazwy zapisanych plików wraz z ich opisem, a także wyniki obserwacji i płynące z nich wnioski.

Na potrzeby kolejnego etapu tworzymy kopię programu, zmieniając jego numer na 6-3-b. Samoczynne zapisywanie wykresu z wynikami pomiarów bardzo ułatwia automatyzację badań. Niestety, wykres zapisany w postaci pliku graficznego BMP nie umożliwia ich dalszego przetwarzania. Dlatego w ostatnim etapie uzupełniamy program o zapis wyników w postaci liczbowej do pliku tekstowego. Umożliwia to ich dalsze przetwarzanie, np. tworzenie w programie Excel zbiorczych wykresów dla poszczególnych badanych dwójników.

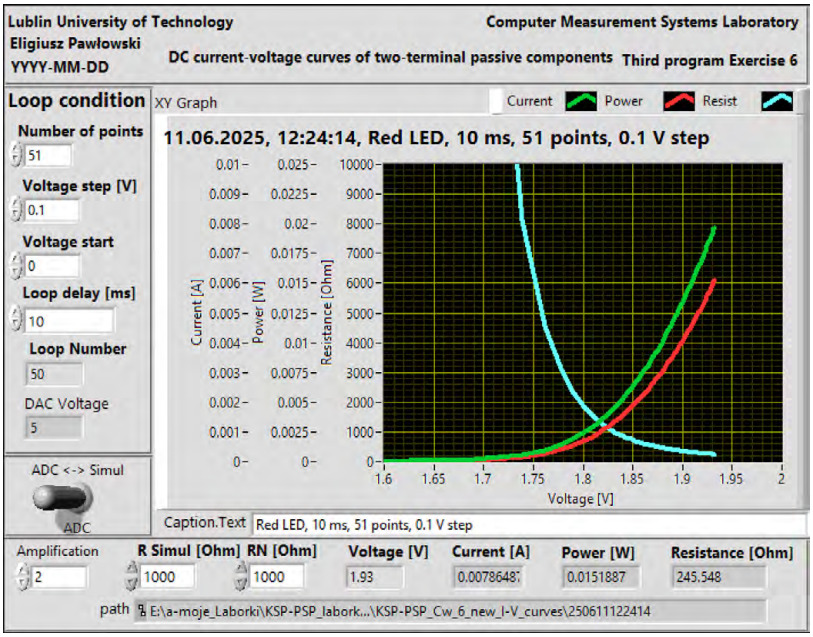
Odpowiednio uzupełniony fragment diagramu przedstawia rycina 6.15. Do pliku tekstowego zapisujemy jedną linię zawierającą opis wykresu, a w kolejnych liniach – dwukolumnową tablicę z wynikami pomiarów napięcia i prądu. Dwuwymiarową tablicę tworzymy z danych dostępnych na wyjściach dwóch rejestrów przesuwnych za pomocą funkcji **Build Array**. Zapis do pliku tekstowego realizujemy w dwóch krokach. Najpierw za pomocą funkcji **Functions > Programming > File I/O > Write to Text File** zapisujemy jedną linijkę tekstu tworzącego opis rysunku (**Caption Text**). Za pomocą funkcji **Functions > Programming > String > Concatenate Strings** dodajemy jedynie na jego końcu znak powrotu karetki (**Functions > Programming > String > Carriage Return Constant**). W podobny sposób do tekstu zawierającego datę i godzinę dodajemy rozszerzenie pliku tekstowego **.txt** (pamiętając o kropce!) i tworzymy ścieżkę dostępu odpowiednią do zapisu pliku tekstowego. Dopisanie tablicy z wynikami pomiarów do utworzonego już pliku tekstowego realizuje funkcja **Functions > Programming > File I/O > Write Delimited Spreadsheet**. Na jej wejściu **format** tworzymy stałą tekstową **%.5f**, która formatuje wyniki z pięcioma miejscami po przecinku. Z kolei na wejściach **append to file?** oraz **transpose?** tworzymy stałe logiczne o wartości **True**. Na koniec uzupełniamy brakujące połączenia zgodnie z rycina 6.15.

Wprowadzamy jeszcze jedną przydatną modyfikację. Przy automatycznym zapisywaniu pomiarów możemy uruchomić program kilka razy w ciągu jednej minuty. Dlatego do opisu wykresu dodajemy również liczbę sekund. Na diagramie odszukujemy funkcję **Get Date/Time String** i na jej wejściu **want seconds?** tworzymy stałą logiczną **True**.

Panel programu nie wymaga żadnych innych zmian ani uzupełnień, ponieważ modyfikacje wprowadzone na diagramie automatycznie skutkują na panelu. Przykładową postać panelu ostatecznej wersji programu pokazano na rycinie 6.16.

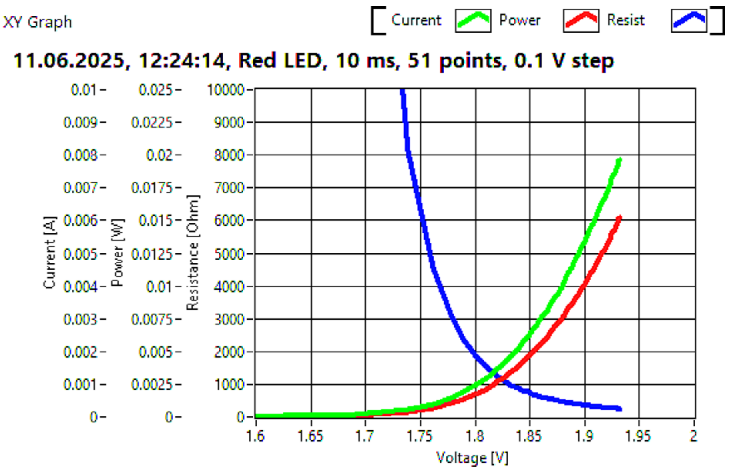



**Ryc. 6.15.** Fragment diagramu uzupełniony o zapis danych liczbowych do pliku tekstowego TXT




**Ryc. 6.16.** Przykładowy panel trzeciego programu z automatycznym zapisem do pliku

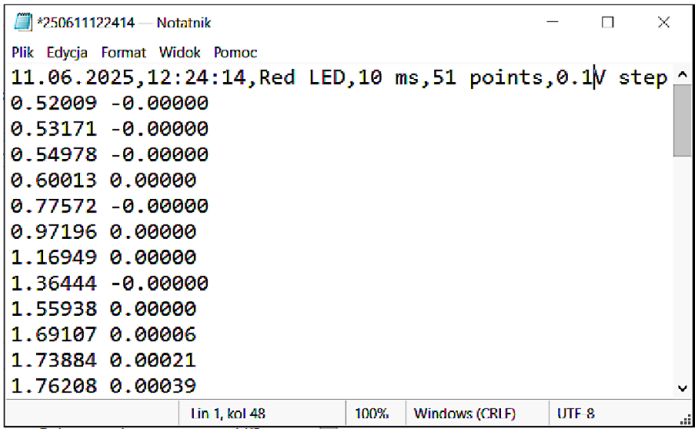
Z opisanych już wcześniej powodów pionowe osie prądu, mocy i rezystancji pozostają wyskalowane w amperach, watach i omach. Zostawiamy ten sposób skalowania, mimo że prąd płynący przez diodę LED osiąga wartość zaledwie kilku miliamperów, moc jest rzędu kilkudziesięciu miliwatów, a rezystancja sięga wielu kiloomów. W końcowej części ćwiczenia badamy inne dwójniki, które pobierają większe prądy i moce przy mniejszych wartościach rezystancji. Zmiana skali na osiach wykresu jest możliwa, ale wymagałaby odpowiedniej modyfikacji na diagramie. Taki program straciłby jednak na uniwersalności przy badaniu innych dwójników. Zapisujemy program na dysku i notujemy jego nazwę w protokole. Uruchamiamy program i sprawdzamy jego działanie. Zwracamy uwagę, czy w opisie wykresu **Caption Text** widnieje informacja o czasie z uwzględnieniem sekund. Ścieżka dostępu do pliku widoczna pod wykresem powinna kończyć się tymi samymi cyframi tworzącymi datę i czas. Jeśli program poprawnie wykonał pomiary i wygenerował wykres, sprawdzamy zawartość pliku graficznego i tekstowego. Przykładowy wykres oraz zawartość pliku tekstowego z wynikami pomiarów pokazano na rycinach 6.17 i 6.18.




**Ryc. 6.17.** Przykładowy wykres zapisany automatycznie do pliku graficznego BMP

Plik graficzny powinien być zapisany poprawnie, tak jak poprzednio. Odszukujemy plik tekstowy i sprawdzamy, czy po kropce posiada on poprawne rozszerzenie .txt. Następnie otwieramy ten plik w Notatniku. Sprawdzamy zawartość pierwszej linii, która powinna być zgodna z opisem wykresu. Pierwsze wyniki pomiarów powinny znajdować się w drugiej linii, a nie na końcu

pierwszej, tuż za opisem wykresu. Decyduje o tym stała tekstowa **Carriage Return**, umieszczona na diagramie i dodana na końcu łańcucha **Caption Text**.




**Ryc. 6.18.** Przykładowa zawartość pliku tekstowego z wynikami pomiarów

Jeśli program działa prawidłowo, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Odszukujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu oraz nazwy plików wraz z opisem zawartości wpisujemy w protokole. Zapisane pliki wykorzystujemy następnie do sporządzenia raportu.

## 6.6. Badania

### 6.6.1. Badania przykładowych dwójników pasywnych

Korzystając z opracowanego programu, przeprowadzamy badania kilku różnych dwójników liniowych i nieliniowych. W tym celu wykonujemy pomiary zgodnie z tabelą 6.1. Wszystkie pomiary realizujemy w zakresie napięć od 0 do +5 V, podawanych na przetwornik DAC. Po dwukrotnym wzmocnieniu we wzmacniaczu mocy zasilamy badane dwójniki (połączone szeregowo z rezystorem wzorcowym) napięciem od 0 do +10 V. Liczbę punktów pomiarowych ustalamy na 51, krok napięcia na 0,1 V, napięcie początkowe na 0 V, a opóźnienie na 100 ms. Wartość tego opóźnienia zmieniamy jedynie podczas badania żarówki halogenowej.

**Tab. 6.1.** Wyniki badań dwójników liniowych i nieliniowych

| Badany dwójnik | Rezystor wzorcowy | Prąd | Napięcie | | Rezystancja | | Nazwa pliku z wykresem |
|---|---|---|---|---|---|---|---|
| | | MAX | MIN | MAX | MIN | MAX | |
| | Ω | mA | V | V | Ω | Ω | |
| LED czerwony | 1000 | ... | ... | ... | ... | ... | ... |
| LED żółty | 1000 | ... | ... | ... | ... | ... | ... |
| LED zielony | 1000 | ... | ... | ... | ... | ... | ... |
| LED niebieski | 1000 | ... | ... | ... | ... | ... | ... |
| LED biały | 1000 | ... | ... | ... | ... | ... | ... |
| Termistor NTC1kΩ | 1000 | ... | ... | ... | ... | ... | ... |
| Rezystor 453Ω | 1000 | ... | ... | ... | ... | ... | ... |
| Symulowany 1000Ω | 1000 | ... | ... | ... | ... | ... | ... |
| Żarówka, 10 ms | 1 | ... | ... | ... | ... | ... | ... |
| Żarówka, 100 ms | 1 | ... | ... | ... | ... | ... | ... |
| Żarówka, 1000 ms | 1 | ... | ... | ... | ... | ... | ... |

Większość pomiarów zestawionych w tabeli 6.1 wykonujemy z rezystorem wzorcowym $R_N = 1000\ \Omega$, co gwarantuje bezpieczną pracę badanych dwójników przy prądach poniżej 10 mA. Jedynie w przypadku żarówki halogenowej wymieniamy rezystor wzorcowy na mniejszy, o wartości $R_N = 1\ \Omega$, ponieważ podczas badania pobiera ona znacznie większy prąd, sięgający nawet 1 A. Zjawisko to zależy m.in. od szybkości narastania napięcia, co ma związek z dużą bezwładnością cieplną włókna żarówki. Efekt ten poddajemy dalszemu badaniu.

Diody LED dołączamy do układu, zwracając szczególną uwagę na ich biegunowość. Anodę zawsze dołączamy do zacisku czerwonego, a katodę – do zacisku żółtego. Zacisk czerwony na module ma potencjał dodatni względem zacisku żółtego. Wyprowadzenie anody w diodach LED jest dłuższe od katody. Z kolei przy katodzie na obudowie znajduje się niewielkie spłaszczenie. Dioda LED, jak każda inna dioda, przewodzi prąd wyłącznie w jednym kierunku: od anody do katody. Przy takim dołączeniu emituje ona światło w określonym kolorze. Problemem pozostaje natomiast niska wytrzymałość napięciowa diody w kierunku zaporowym, która wynosi zaledwie kilka woltów. Z tego względu element ten może łatwo ulec uszkodzeniu przy odwrotnym dołączeniu do napięcia rzędu 10 V.

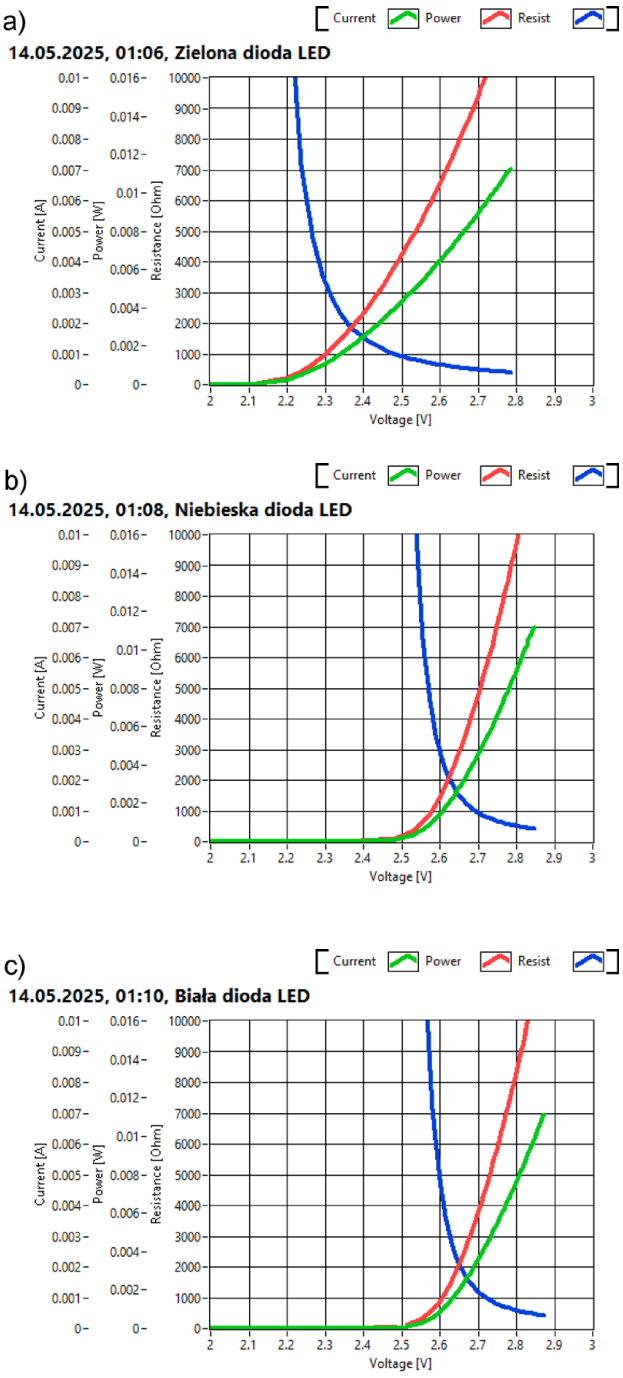
Wykonujemy tyle pomiarów, na ile pozwala czas trwania zajęć w laboratorium. Przed każdym pomiarem uaktualniamy opis w polu **Caption Text**.

Ponieważ program automatycznie zapisuje do plików wykresy i dane liczbowe, badania przeprowadzamy bardzo szybko. Pomiary możemy wykonać w innej kolejności niż ta podana w tabeli 6.1, szczególnie jeśli mamy zbyt mało czasu na realizację wszystkich badań. Po każdym pomiarze sprawdzamy zawartość plików z wykresem i danymi liczbowymi, a nazwę pliku notujemy w tabeli 6.1. Ponieważ dla każdego badanego dwójnika uzyskujemy wyniki pomiarów o innych wartościach, w razie potrzeby odpowiednio modyfikujemy zakresy osi. Niestety, program zapisuje pliki natychmiast po wykonaniu pomiarów, dlatego zmiana zakresów na osiach przynosi skutek dopiero przy jego kolejnym uruchomieniu. Z tego powodu nierzadko wykonujemy kilka pomiarów dla tego samego dwójnika. W tabeli 6.1 wpisujemy wówczas jedynie nazwę pliku z ostateczną wersją wykresu.

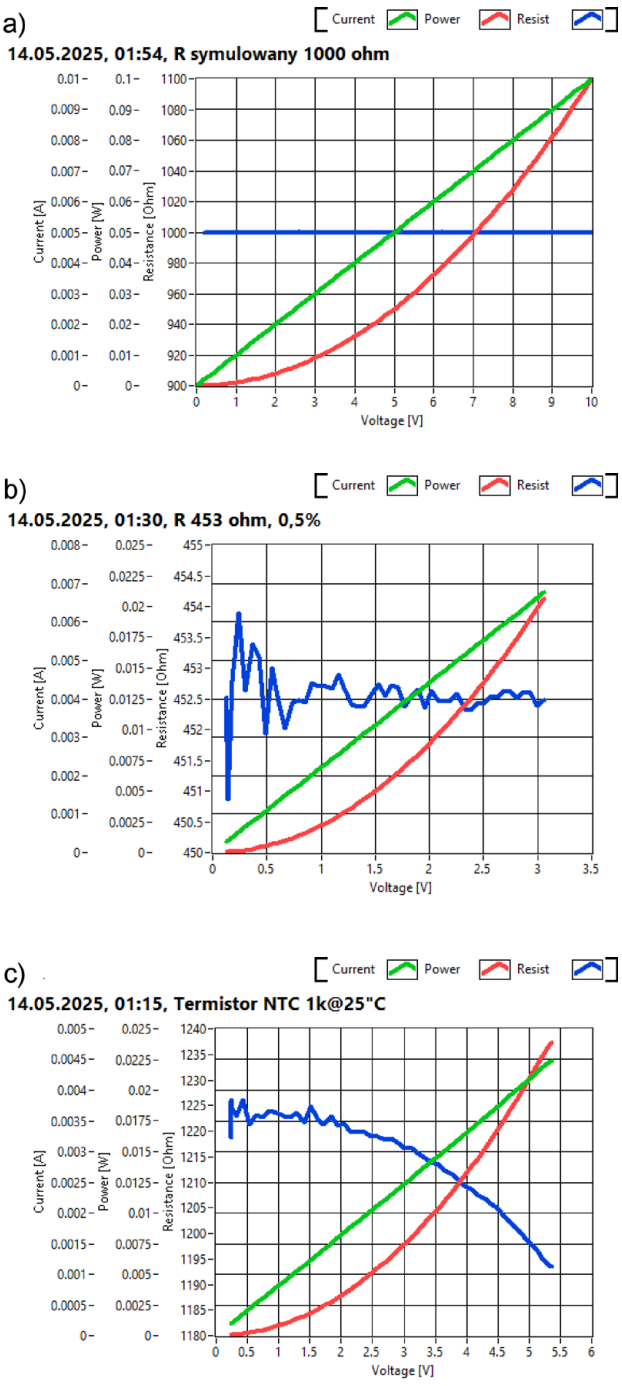
Przykładowe wykresy dla diod LED zielonej, niebieskiej i białej pokazano na rycinie 6.19. Z kolei wykresy dla diody czerownej widać na rycinie 6.17. Wyraźnie na nich obserwujemy, że napięcie przewodzenia diody LED zależy od koloru emitowanego światła. Najniższym napięciem przewodzenia (ok. 1,7 V) charakteryzuje się dioda czerwona, a najwyższym (ok. 2,5 V) – dioda biała, przy czym bardzo zbliżone parametry wykazuje dioda niebieska. Charakterystyki prądowo-napięciowe różnią się również stromością – w przypadku diody zielonej prąd narasta znacznie wolniej niż dla pozostałych.

Przykładowe wykresy dla dwójników rezystancyjnych przedstawiono na rycinie 6.20. Wykres dla symulowanego rezystora o wartości 1000 Ω (ryc. 6.20a) pozostaje zgodny z teoretycznymi zależnościami i w pełni potwierdza poprawność działania naszego programu. Obserwujemy na nim, że prąd narasta liniowo i proporcjonalnie do napięcia, moc zwiększa się zgodnie z funkcją kwadratową, a rezystancja zachowuje stałą wartość.

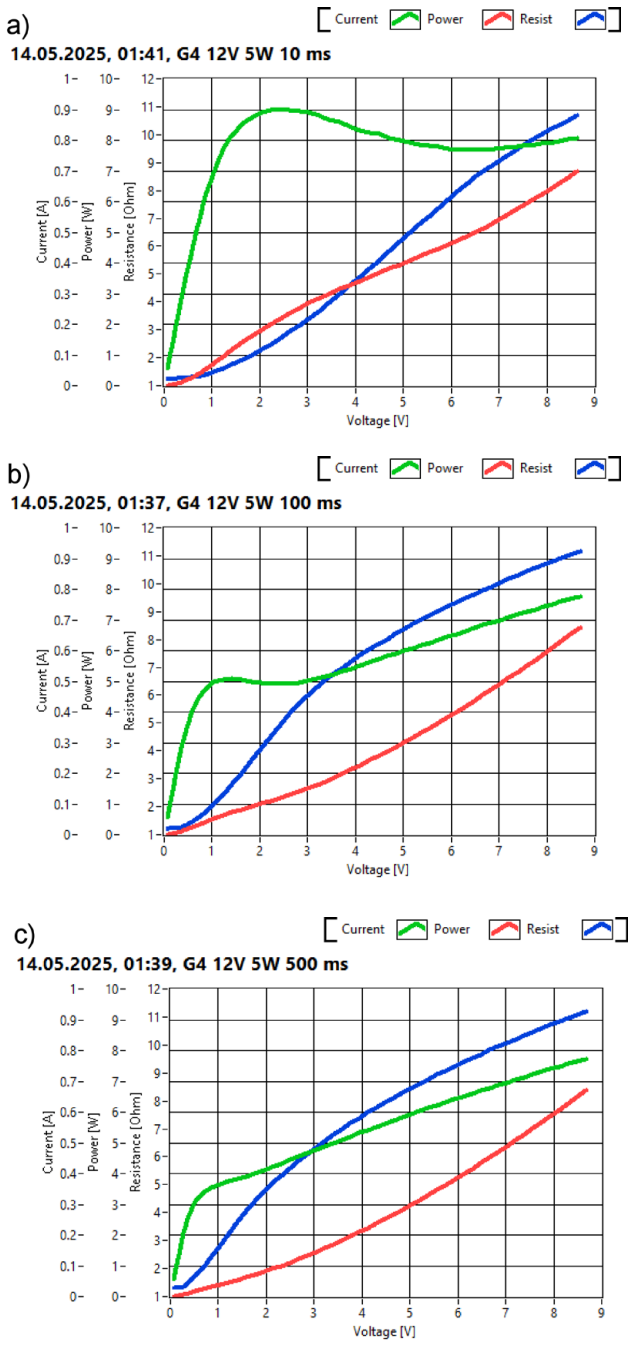
Wykres dla rezystora warstwowego metalizowanego AT 1/4W, 453 Ω, 0,5 %, potwierdza jego wysoką dokładność (ryc. 6.20b). Błąd wynosi zaledwie 0,5 Ω, czyli 0,12 %, a więc element ten okazuje się znacznie dokładniejszy, niż deklaruje producent. Zauważalne na początku charakterystyki podwyższone niepewności pomiarowe wynikają z błędów przetwornika ADC w dolnej części zakresu pomiarowego. Efekt ten dostrzegamy wyłącznie dzięki temu, że skala rezystancji obejmuje bardzo wąski zakres od 450 Ω do 455 Ω. Gdyby skala zaczynała się od 0 Ω, zjawisko to pozostałoby praktycznie niewidoczne.




**Ryc. 6.19.** Przykładowe charakterystyki zbadanych diod LED: a) zielonej, b) niebieskiej, c) białej, pomiary z rezystorem wzorcowym $R_N$ = 1000 Ω, opóźnienie 100 ms, 51 punktów pomiarowych, krok napięcia 100 mV na ADC, prądy w zakresie do 7 mA




**Ryc. 6.20.** Przykładowe charakterystyki zbadanych elementów rezystancyjnych: a) rezystor symulowany 1000 Ω, b) rezystor rzeczywisty 453 Ω, 0,5 %, c) termistor NTC 1 kΩ @ 25°C, rezystor wzorcowy $R_N$ = 1000 Ω, opóźnienie 100 ms, 51 punktów, krok napięcia 100 mV na ADC




**Ryc. 6.21.** Przykładowe charakterystyki żarówki halogenowej G4 12V 5 W: a) opóźnienie 10 ms, b) opóźnienie 100 ms, c) opóźnienie 5000 ms, pomiary z rezystorem wzorcowym $R_N = 1\ \Omega$, 51 punktów pomiarowych, krok napięcia 100 mV na ADC, prądy w zakresie do 0,9 A

Wykres uzyskany dla badanego termistora (ryc. 6.20c) potwierdza jego ujemny współczynnik temperaturowy NTC. Termistor nagrzewa się pod wpływem przepływającego prądu, a jego rezystancja maleje. Badany element charakteryzuje się rezystancją 1 kΩ w temperaturze 25°C. Ponieważ w laboratorium panuje temperatura niższa niż 25°C, początkowa rezystacja termistora jest odpowiednio wyższa i maleje wraz ze wzrostem traconej w nim mocy.

Na rycinie 6.21 pokazano przykładowe wyniki badania żarówki halogenowej dla różnych szybkości narastania napięcia. Według danych technicznych podanych przez producenta jest to żarówka halogenowa o mocy znamionowej 5 W przy napięciu 12V. Wyniki pomiarów nie potwierdzają jednak tych parametrów – żarówka osiąga moc 5 W już przy napięciu ok. 7 V. Uzyskane charakterystyki są silnie zależne od szybkości narastania napięcia, co wynika z wysokiej temperatury włókna żarówki i dużych zmian jego rezystancji. Rezystancja wolframowego włókna zmienia się w trakcie pomiaru ponad dziesięciokrotnie. Z tego względu przy szybkich zmianach napięcia (ryc. 6.21a) włókno nie nadąża się nagrzać, a prąd osiąga duże wartości (sięgające 0,9 A dla niskiego napięcia 2 V). Rezystancja żarówki wynosi w tym punkcie zaledwie 2,2 Ω. Następnie prąd maleje, mimo że napięcie rośnie, ponieważ włókno z czasem się rozgrzewa. Przy wolniejszych zmianach napięcia (ryc. 6.21c) prąd nie osiąga tak dużych wartości, ponieważ włókno ma więcej czasu na nagrzanie się do wyższej temperatury. Obserwowany efekt wynika zatem z bezwładności cieplnej włókna żarówki. Tradycyjne żarówki z włóknem wolframowym straciły na znaczeniu w wyniku powszechnego obecnie stosowania źródeł światła typu LED. Jednak unikalne właściwości nieliniowej charakterystyki włókna wolframowego wciąż znajdują wiele zastosowań w aparaturze pomiarowej i badawczej.

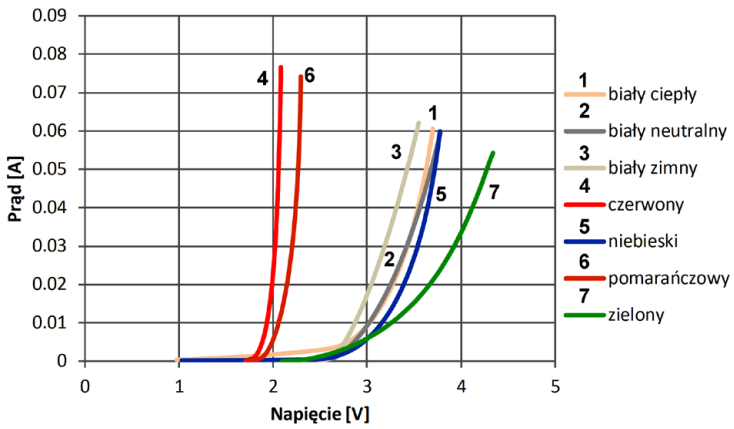
W sprawozdaniu, korzystając z programu Excel, umieszczamy na jednym wykresie charakterystyki prądowo-napięciowe wszystkich zbadanych przez nas diod LED. W tym celu wczytujemy wszystkie pliki tekstowe do programu, tworzymy jedną tabelkę ze wszystkimi zebranymi danymi i przedstawiamy je na wspólnym wykresie. Przykładowe zestawienie pokazano na rycinie 6.22. Wszystkie wykresy korzystają z tych samych osi XY. Pamiętamy przy tym, że każdy z wykresów posiada własne dane liczbowe dla współrzędnych X i Y.

Wykresy przedstawione na rycinie 6.21 uzyskano, wykonując pomiary z rezystorem wzorcowym $R_N = 100\ \Omega$, dlatego zmierzono około dziesięciokrotnie większe wartości prądów w porównaniu z ryciną 6.19. Ponieważ tego

rodzaju pomiary grożą zniszczeniem diody LED przy zbyt długim czasie ich trwania, wykonano je z opóźnieniem wynoszącym 10 ms.

**UWAGA!** Podczas zajęć w laboratorium nie używamy rezystora wzorcowego 100 Ω do badania diod LED, ponieważ niektóre z egzemplarzy mogłyby ulec zniszczeniu, a nawet rozerwaniu, stwarzając niebezpieczeństwo dla osób znajdujących się w pobliżu.

Po uzgodnieniu z prowadzącym zajęcia możemy zbadać inne, własne dwójniki. Pamiętamy przy tym, że wydajność prądowa wzmacniacza wynosi około 2 A. Rezystor wzorcowy dobieramy tak, aby ograniczyć maksymalną wartość prądu i zapewnić wystarczający spadek napięcia do pomiaru prądu (przynajmniej 1 V). Maksymalne napięcie wynosi 10 V, ale zmieniając liczbę kroków i ich wartość, możemy odpowiednio zmniejszyć ten zakres.




**Ryc. 6.22.** Charakterystyki zbadanych diod LED zestawione na wspólnym wykresie w programie Excel, pomiary z rezystorem wzorcowym 100 Ω, prądy 50–80 mA

## 6.7. Opracowanie raportu z realizacji ćwiczenia

W raporcie z przeprowadzonego ćwiczenia zamieszczamy:

- stronę tytułową z danymi zespołu, datą ćwiczenia i jego tytułem,
- opis środowiska pomiarowego,

- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań zgodnie z programem ćwiczenia,
- opis każdego wykonanego zadania (lub uzasadnienie przyczyny jego niewykonania),
- dokumentację trzech zrealizowanych i uruchomionych programów,
- wykresy wygenerowane przez program w środowisku LabVIEW,
- wykres opracowany w arkuszu kalkulacyjnym Excel dla diod LED,
- liczbowe wyniki pomiarów uzyskane z opracowanych programów,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.

Przygotowując raport, pamiętamy, że ocenie podlegają:

- poprawność działania programów,
- czytelność i przejrzystość diagramu,
- estetyka panelu (w tym dobór kolorów i elementów graficznych),
- funkcjonalność panelu pod kątem wygody obsługi,
- zakres zrealizowanych zadań,
- trafność wniosków końcowych.

# 7. Układ automatycznej regulacji temperatury

## 7.1. Cel ćwiczenia i zadania do wykonania

Ćwiczenie 7 jest kontynuacją ćwiczenia 6. Podstawą do jego rozpoczęcia jest drugi program, który opracowaliśmy w poprzednim ćwiczeniu, służący do wyznaczania charakterystyki prądowo-napięciowej dwójnika.

Celem ćwiczenia 7 jest zapoznanie się z problematyką programowania systemu pomiarowego w środowisku LabVIEW z jednoczesnym wykorzystaniem dwóch kanałów przetwornika analogowo-cyfrowego oraz dwóch wyjść cyfrowych. Przetwornik analogowo-cyfrowy wykorzystujemy do pomiaru temperatury, natomiast dwa wyjścia cyfrowe służą nam do sterowania grzałką i wentylatorem chłodzącym. Nowym zagadnieniem w tym ćwiczeniu jest również tworzenie podprogramów subVI w środowisku LabVIEW oraz ich wykorzystanie w programie nadrzędnym.

Program laboratorium obejmuje następujące zagadnienia:

- konfigurację modułu NI USB-6001 do jednoczesnego pomiaru napięcia za pomocą dwóch kanałów przetwornika analogowo-cyfrowego oraz obliczanie rezystancji platynowego czujnika Pt1000 w celu wyznaczenia temperatury,
- konfigurację tego samego modułu do jednoczesnego ustawiania stanów logicznych na dwóch wyjściach cyfrowych w celu sterowania grzałką

i wentylatorem chłodzącym za pośrednictwem dwukanałowego modułu z przekaźnikami,
- tworzenie podprogramów w środowisku LabVIEW, edycję ikon oraz konfigurację panelu złączy dla tworzonych modułów subVI, tak abyśmy mogli wykorzystywać je jako kompletne funkcje w programie nadrzędnym,
- przygotowanie programu LabVIEW do automatycznego sterowania temperaturą grzałki z wykorzystaniem programowania modułowego i opracowanych wcześniej funkcji subVI.

W trakcie ćwiczenia przygotowujemy odpowiednie programy w środowisku LabVIEW, uruchamiamy je, sprawdzamy poprawność ich działania, usuwamy ewentualne błędy, a następnie wykonujemy pomiary i sporządzamy raport.

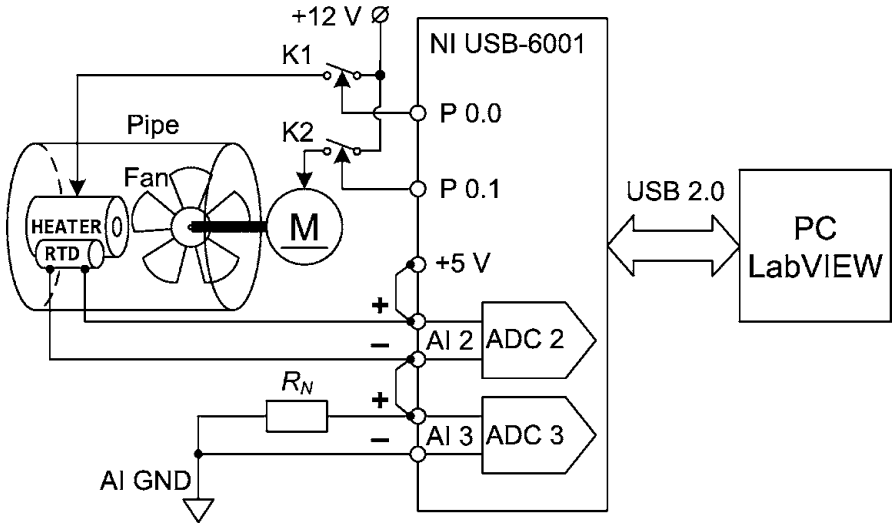
W ramach tego ćwiczenia opracowujemy cztery programy:
1. Pierwszy – służący do pomiaru temperatury za pomocą platynowego czujnika termorezystancyjnego Pt1000.
2. Drugi – przeznaczony do sterowania grzałką i wentylatorem za pośrednictwem modułu z dwoma przekaźnikami.
3. Trzeci – realizujący automatyczną regulację temperatury.
4. Czwarty – rozbudowany o automatyczną regulację temperatury z uwzględnieniem histerezy.

## 7.2. Opis stanowiska laboratoryjnego

### 7.2.1. Schemat blokowy stanowiska laboratoryjnego

Stanowisko laboratoryjne, z którego korzystamy w ćwiczeniu 7, składa się z następujących komponentów:
- komputera PC z zainstalowanym środowiskiem LabVIEW,
- modułu akwizycji danych pomiarowych NI USB-6001,
- zasilaczy napięcia stałego 12 V oraz 5 V,
- modułu z dwoma przekaźnikami elektromagnetycznymi,
- modułu z grzałką, wentylatorem i platynowym czujnikiem temperatury Pt1000.

Schemat blokowy stanowiska laboratoryjnego, wykorzystywanego w tym ćwiczeniu do automatycznej regulacji temperatury, przedstawiono na rycinie 7.1. Stanowisko to stanowi uproszczony model dydaktyczny przykładowego procesu technologicznego ze stabilizacją temperatury w określonych granicach. Model ten składa się z grzałki, wentylatora chłodzącego oraz modułu pomiarowego i sterującego dwoma przekaźnikami. Naszym zadaniem jest opracowanie programu, który na podstawie bieżących wyników pomiaru steruje zasilaniem grzałki i wentylatora tak, aby utrzymać zadaną temperaturę obiektu.




**Ryc. 7.1.** Schemat blokowy stanowiska laboratoryjnego w ćwiczeniu 7

Moduł akwizycji danych pomiarowych NI USB-6001 dołączamy do komputera PC za pomocą interfejsu USB. Wykorzystujemy w nim dwa wyjścia cyfrowe do sterowania grzałką (P 0.0) i wentylatorem (P 0.1), dwa wejścia analogowe w trybie różnicowym (AI 2, AI 3) do pomiaru temperatury grzałki oraz wyjście napięcia +5 V do zasilania czujnika temperatury [6]. Ponieważ wyjścia cyfrowe modułu NI USB-6001 charakteryzują się wydajnością prądową wynoszącą zaledwie ±4 mA [5], do sterowania większymi obciążeniami stosujemy dodatkowe przekaźniki K1 i K2. Na naszym stanowisku wykorzystujemy w tym celu moduł z dwoma przekaźnikami elektromagnetycznymi oraz układami izolacji optycznej. Wyjście cyfrowe P 0.0 steruje pierwszym przekaźnikiem, którego styk zwierny K1 włącza zasilanie grzałki (zasilanej napięciem stałym 12 V). Z kolei wyjście cyfrowe P 0.1 steruje drugim przekaźnikiem,

którego styk zwierny K2 uruchamia silnik wentylatora (również zasilany napięciem 12 V). Aby zapobiec ryzyku poparzenia, układ z grzałką i czujnikiem temperatury umieszczony jest w przezroczystej rurze z poliwęglanu, w której znajduje się również wentylator chłodzący.

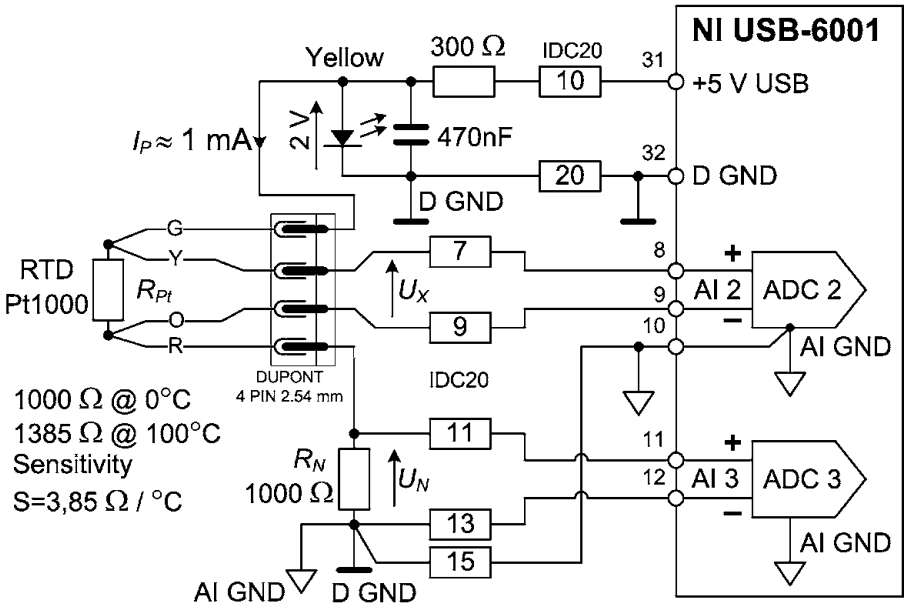
Do pomiaru temperatury grzałki wykorzystujemy rezystancyjny czujnik temperatury RTD (**Resistance Temperature Detector**) typu Pt1000 [43]. Umieszczony jest on wewnątrz grzałki, w metalowej osłonie wypełnionej dodatkowo silikonową pastą termoprzewodzącą. Do samego pomiaru temperatury stosujemy metodę ratiometryczną [44]. W tym celu wykorzystujemy dwa wejścia analogowe (AI 2 i AI 3) przetwornika analogowo-cyfrowego, pracujące w trybie różnicowym. Nasz układ ma konfigurację zbliżoną do tej z ćwiczenia 6. W szereg z czujnikiem Pt1000 włączony jest rezystor wzorcowy $R_N$ o rezystancji 1000 Ω. Całość zasilamy napięciem +5 V, dostępnym na złączu modułu NI USB-6001 [6]. Spadek napięcia na czujniku RTD mierzymy w kanale AI 2, natomiast spadek na rezystorze wzorcowym $R_N$ – w kanale AI 3. Na podstawie tych dwóch wartości oraz znanej rezystancji $R_N$ obliczamy rezystancję czujnika RTD, a następnie wyznaczamy jego temperaturę [44].

### 7.2.2. Układ do pomiaru temperatury

Schemat ideowy układu do pomiaru temperatury za pomocą czujnika termorezystancyjnego RTD typu Pt1000 przedstawiono na rycinie 7.2. Czujnik ten posiada dokładnie znaną i znormalizowaną charakterystykę przetwarzania temperatury na rezystancję [43]. W temperaturze 0°C jego rezystancja wynosi 1000 Ω, natomiast w temperaturze 100°C wzrasta do 1385 Ω. W przedziale od 0°C do 100°C czułość czujnika Pt1000 wynosi zatem 3,85 Ω/°C. W celu uzyskania dokładniejszych wyników pomiaru temperatury należy zastosować charakterystykę opisaną wielomianem drugiego stopnia [43].

Wykorzystywana przez nas metoda ratiometryczna (stosunkowa, od łac. *ratio* – stosunek) polega na wyznaczeniu rezystancji czujnika RTD na podstawie stosunku dwóch napięć. Czujnik temperatury Pt1000 i rezystor wzorcowy $R_N$ łączymy ze sobą szeregowo. Ze względu na efekt samonagrzewania (*self-heating effect*) występujący w czujnikach termorezystancyjnych [45, 46], musimy odpowiednio ograniczyć prąd przepływający przez element pomiarowy. Dokładny dobór wartości tego prądu wymaga wcześniejszego wyznaczenia rezystancji termicznej czujnika [47], co stanowi przedmiot ćwiczenia 8. Na naszym

stanowisku przyjmujemy prąd pomiarowy o wartości około 1 mA. Wymaga to zasilenia połączonych szeregowo elementów (czujnika RTD i rezystora wzorcowego $R_N$) napięciem około 2 V. W tym celu w roli stabilizatora napięcia wykorzystujemy żółtą diodę LED, zasilaną z napięcia +5 V poprzez rezystor 300 Ω. Płynący przez diodę prąd o wartości około 10 mA wytwarza na niej wymagany spadek napięcia 2 V, a jej świecenie dodatkowo sygnalizuje obecność napięcia zasilającego +5 V z modułu NI USB-6001. Aby wyeliminować wpływ rezystancji przewodów, stosujemy czteroprzewodowe połączenie czujnika z układem pomiarowym (tzw. układ Kelvina). Spadek napięcia $U_X$ na czujniku Pt1000 mierzymy różnicowo w kanale AI 2. Z kolei kanał AI 3 służy nam do pomiaru napięcia $U_N$ na znanym oporze odniesienia $R_N = 1000\ \Omega$.




**Ryc. 7.2.** Schemat ideowy układu do pomiaru temperatury metodą ratiometryczną z dołączeniem czujnika w układzie Kelvina [44]

Rezystancję czujnika temperatury obliczamy zgodnie z prawem Ohma. W pierwszej kolejności wyznaczamy dokładną wartość prądu pomiarowego $I_P$, płynącego przez czujnik Pt1000 oraz połączony z nim szeregowo rezystor odniesienia $R_N$:

$$I_P = \frac{U_N}{R_N}. \tag{7.1}$$

Znając wartość prądu pomiarowego $I_P$ płynącego przez czujnik Pt1000 oraz zmierzone na nim napięcie $U_X$ w kanale AI 2, obliczamy jego rezystancję $R_{Pt}$:

$$R_{Pt} = \frac{U_X}{I_P} = \frac{U_X}{U_N} R_N . \tag{7.2}$$

Zakładając liniową charakterystykę przetwarzania czujnika Pt1000, opisujemy ją zależnością:

$$R_{Pt} = R_0 \left(1 + \alpha T\right), \tag{7.3}$$

gdzie: $R_{Pt}$ jest rezystancją czujnika w temperaturze $T$, $R_0 = 1000\ \Omega$ – rezystancją czujnika w temperaturze początkowej 0°C, natomiast $\alpha$ – współczynnikiem temperaturowym rezystancji równym 0,00385°C$^{-1}$. Uwzględniając podane wartości parametrów, ostatecznie obliczamy temperaturę czujnika $T$ (w stopniach Celsjusza) na podstawie jego rezystancji $R_{Pt}$:

$$T = \frac{R_{Pt} - R_0}{\alpha R_0} = \frac{R_{Pt} - 1000}{3,85}. \tag{7.4}$$

Aby uzyskać dokładniejszy pomiar temperatury, przyjmujemy charakterystykę czujnika w postaci wielomianu drugiego stopnia [43]:

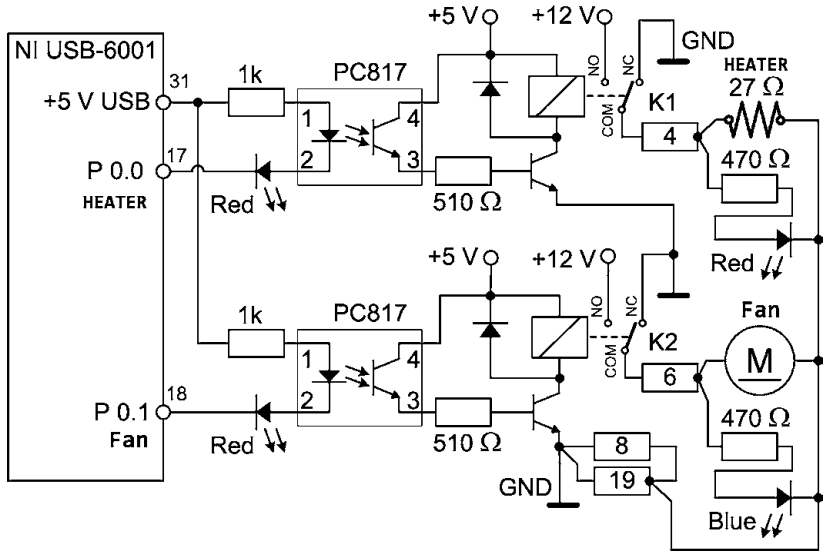
$$R_{Pt} = R_0 \left(1 + AT + BT^2\right), \tag{7.5}$$

gdzie współczynniki wielomianu wynoszą: $A = 3,9083 \times 10^{-3}$ °C$^{-1}$, $B = -5,775 \times 10^{-7}$ °C$^{-2}$ [43]. Powyższą charakterystykę stosujemy dla temperatur powyżej 0°C, natomiast w przypadku temperatur ujemnych wykorzystujemy wielomian trzeciego stopnia [43]. Po przekształceniu równania (7.5) temperaturę czujnika $T$ obliczamy ze wzoru [44]:

$$T = \frac{-A + \sqrt{A^2 - 4B(1 - R_{Pt}/R_0)}}{2B}. \tag{7.6}$$

### 7.2.3. Układ do sterowania grzałką i wentylatorem

Schemat układu do sterowania grzałką i wentylatorem chłodzącym za pomocą dwukanałowego modułu przekaźnikowego pokazano na rycinie 7.3. Po włączeniu zasilania modułu NI USB-6001 wszystkie jego linie cyfrowe działają domyślnie jako wejścia. Każda z linii cyfrowych wyposażona jest w wewnętrzny rezystor *pull down* o wartości 47,5 kΩ, który wymusza na wejściu stan niski. Takie wejście zachowuje się jednak jednocześnie jak wyjście cyfrowe w tzw. słabym stanie niskim (*weak Low*). Jeśli do takiej linii dołączymy układ zasilany napięciem +5 V, w obwodzie popłynie prąd o wartości do 0,1 mA. Może to spowodować nieplanowane załączenie czułego odbiornika zaraz po uruchomieniu zasilania modułu NI USB-6001, jeszcze przed startem naszego programu sterującego. Musimy to uwzględnić, projektując układ sterowania, aby uniknąć ryzyka awarii.



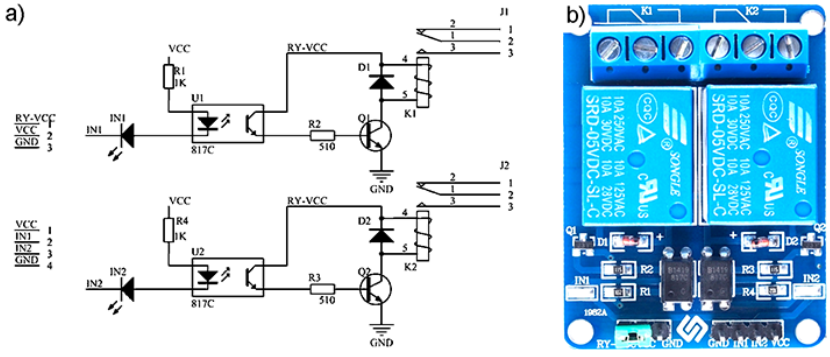

**Ryc. 7.3.** Schemat ideowy układu z dwoma przekaźnikami elektromechanicznymi do sterowania grzałką i wentylatorem

Po skonfigurowaniu linii cyfrowych modułu NI USB-6001 jako wyjścia, działają one domyślnie w fabrycznej konfiguracji *active drive* [5]. W tym trybie pracy charakteryzują się wydajnością prądową wynoszącą zaledwie ±4 mA [6]. Z tego względu do sterowania większymi obciążeniami stosujemy dodatkowe przekaźniki. Wyjścia cyfrowe modułu sterują cewkami przekaźników poprzez

transoptory PC817 oraz tranzystory, dzięki czemu prąd pobierany z wyjść nie przekracza 2 mA. Dodatkowo umożliwia to galwaniczne oddzielenie obwodu zasilania cewek (z zewnętrznego zasilacza sieciowego +5 V) od napięcia +5 V z portu USB, z którego korzystamy w module NI USB-6001 do zasilania układu pomiarowego z czujnikiem Pt1000. Rozdzielamy również obwody masy obu tych układów, co znacząco zwiększa bezpieczeństwo użytkowania modułu pomiarowego. Prąd sterujący transoptorem przepływa przez czerwoną diodę LED, wlutowaną obok przekaźnika, co dodatkowo sygnalizuje nam stan pracy układu.

Przekaźnik załączamy, podając stan niski (**Low**) na odpowiednie wyjście cyfrowe. Powoduje to przełączenie styku COM (**common** – wspólny) z pozycji NC (**normally closed** – normalnie zamknięty) na pozycję NO (**normally open** – normalnie otwarty). Obciążenia (grzałka i wentylator) są dołączone do styków COM przekaźników. Z kolei styki NO są połączone z linią zasilania +12 V, natomiast styki NC – z masą (GND). Tym samym ustawienie stanu niskiego na wyjściu P 0.0 uruchamia grzałkę, a na wyjściu P 0.1 – wentylator. Z kolei stan wysoki (**High**) na danym kanale wyłącza przypisane do niego obciążenie. Równolegle do grzałki dołączona jest czerwona dioda LED, a do wentylatora – niebieska. Sygnalizują one obecność napięcia +12 V na tych obciążeniach. Grzałka i wentylator są umieszczone na dodatkowym module ćwiczeniowym, natomiast przekaźniki z zasilaczem +12 V – na głównym panelu.

Na panelu ćwiczeniowym wykorzystujemy gotowy moduł zawierający dwa przekaźniki elektromechaniczne z układami optoizolacji [48]. Schemat oraz wygląd zewnętrzny tego modułu pokazano na rycinie 7.4.



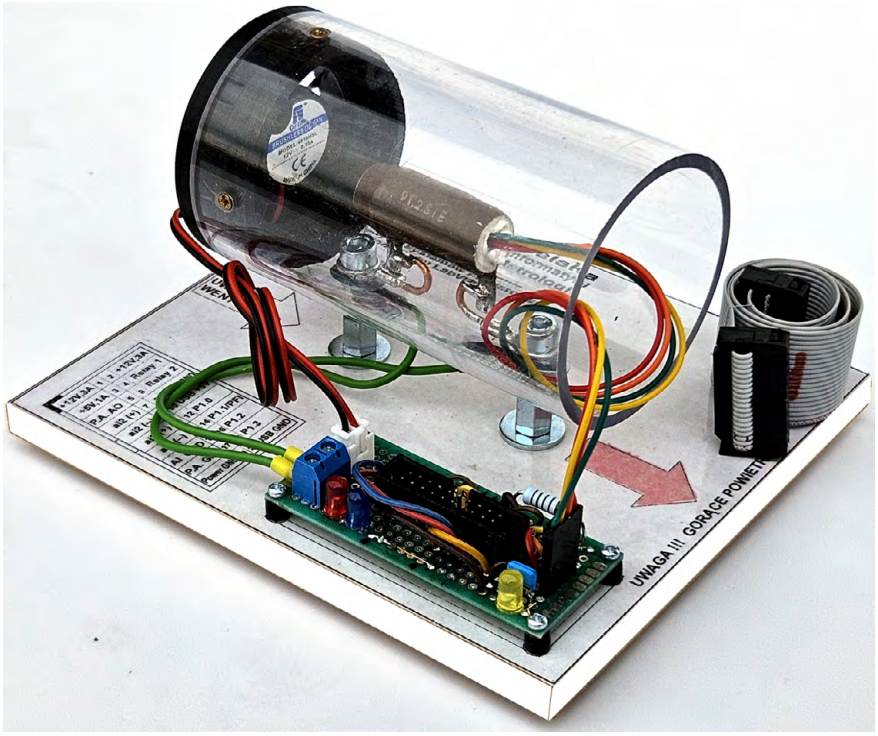

**Ryc. 7.4.** Dwukanałowy moduł z przekaźnikami: a) schemat ideowy, b) rzeczywisty wygląd

Wykorzystywany przez nas układ to uniwersalny, dwukanałowy moduł przekaźnikowy z optoizolacją. Cewki przekaźników wymagają zasilania napięciem 5 V, które podłączamy do styku RY-VCC. Obwody wejściowe transoptorów również zasilamy napięciem +5 V, jednak doprowadzamy je do oddzielnego styku VCC. Transoptor oraz połączony z nim przekaźnik załączamy, zwierając odpowiednie wejście (IN1 lub IN2) do masy (GND). Każdy z kanałów wymaga do załączenia prądu sterującego o wartości rzędu 1,5–2 mA. Styki przekaźników charakteryzują się obciążalnością prądową 10 A przy napięciu AC 250 V lub 10 A dla napięcia DC 30 V. Punkty VCC i RY-VCC możemy dołączyć do dwóch różnych napięć zasilających lub do wspólnego napięcia, pamiętając jednak, że w tym drugim przypadku rezygnujemy z separacji galwanicznej obu obwodów.

### 7.2.4. Budowa modułu z grzałką i wentylatorem

Dydaktyczny model procesu technologicznego ze stabilizacją temperatury stanowi oddzielny moduł ćwiczeniowy, przedstawiony na rycinie 7.5. Jego podstawę tworzy płyta HDF (*High-Density Fiberboard*) o wymiarach 13 cm × 15 cm, na której zamontowano przeźroczystą rurę z poliwęglanu o średnicy 60 mm i długości 100 mm. Na jednym z końców rury zainstalowany jest wentylator chłodzący z silnikiem zasilanym napięciem 12 V DC, którego mocowanie wydrukowano na drukarce 3D. Wewnątrz rury, na miedzianych wspornikach, zamocowano grzałkę z czujnikiem temperatury Pt1000. Rolę grzałki pełni rezystor drutowy 27 Ω w ceramicznej obudowie o mocy znamionowej 7,5 W. Podczas pracy rezystor ten zasilany jest napięciem stałym 12 V, co skutkuje wydzielaniem się na nim mocy około 5,3 W. Dzięki temu temperatura obiektu narasta od wartości pokojowej (początkowo z szybkością ok. 0,3°C/s) i przy wyłączonym wentylatorze stabilizuje się na poziomie ok. 130°C w czasie niespełna 10 minut. Z kolei po włączeniu wentylatora temperatura zasilanej grzałki utrzymuje się na poziomie około 80°C. Po jej wyłączeniu wentylator obniża temperaturę układu (ok. 0,3°C/s), pozwalając na powrót do wartości pokojowej (ok. 24°C) w czasie do 10 minut. Poliwęglan, z którego wykonano rurę, to samogasnące tworzywo sztuczne o temperaturze zapłonu wynoszącej ok. 340°C. Jego temperatura mięknienia to 150°C, topnienia – 230°C, natomiast dopuszczalna długotrwała temperatura pracy sięga 120°C. Ponieważ grzałka nie ma bezpośredniego kontaktu z rurą, układ pozostaje całkowicie

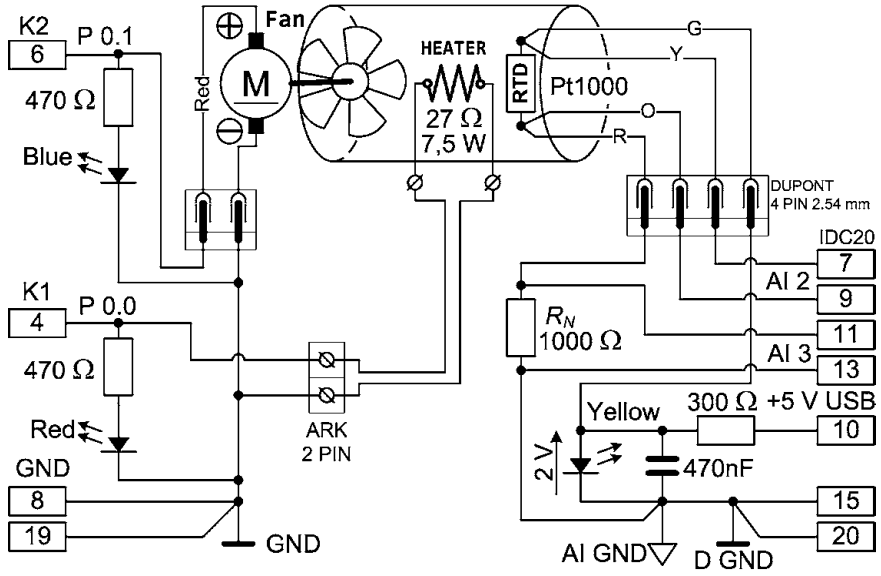
bezpieczny nawet po jej włączeniu na bardzo długi czas. Rezystor, z którego wykonano ten element, ma postać ceramicznej rurki z wewnętrznym otworem. Dzięki temu czujnik temperatury został umieszczony w samym jej środku, w metalowej osłonie wypełnionej dodatkowo silikonową pastą termoprzewodzącą.




**Ryc. 7.5.** Moduł z grzałką i wentylatorem

Schemat ideowy połączeń na module przedstawiono na rycinie 7.6. Silnik wentylatora zasilany jest napięciem +12 V poprzez styk 5 złącza IDC, podłączony do przekaźnika K2 na panelu ćwiczeniowym. Przekaźnik ten załączany jest stanem niskim na wyjściu cyfrowym P 0.1, a obecność napięcia na wentylatorze sygnalizuje niebieska dioda LED. Grzałka zasilana jest również napięciem +12 V, jednak poprzez styk 4 na złączu IDC, który jest połączony z przekaźnikiem K1. Przekaźnik ten uruchamiany jest stanem niskim na wyjściu cyfrowym P 0.0, co z kolei sygnalizuje czerwona dioda LED. Z kolei żółta dioda wskazuje na obecność napięcia +5 V z portu USB, wyprowadzonego ze złącza modułu NI USB-6001 (PIN 31). Spadek napięcia na tej diodzie wynosi

2 V i jest wykorzystany do zasilania połączonych szeregowo: czujnika Pt1000 i rezystora odniesienia $R_N$.




**Ryc. 7.6.** Schemat ideowy modułu z grzałką i wentylatorem

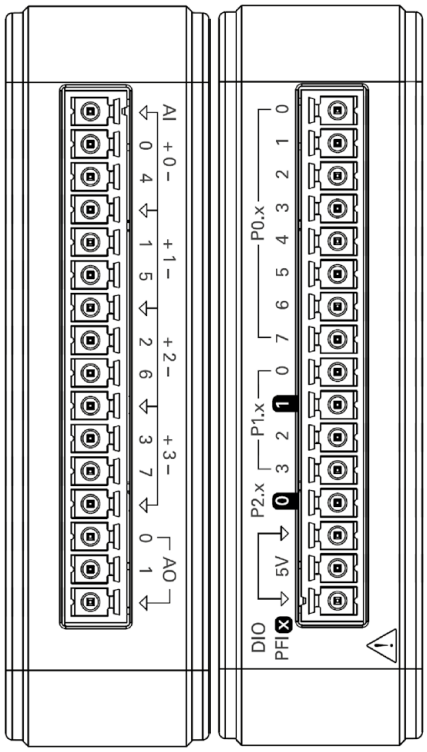
## 7.3. Przygotowanie stanowiska laboratoryjnego do pracy

Moduł z grzałką, wentylatorem i układem do pomiaru temperatury dołączamy do panelu ćwiczeniowego za pomocą płaskiego przewodu zakończonego wtyczkami IDC20. Pozostałe elementy układu pomiarowego (moduł pomiarowy NI USB-6001, moduł z przekaźnikami oraz zasilacze +5 V i +12 V) zamontowane są na stałe na panelu ćwiczeniowym i połączone zgodnie z przedstawionymi wcześniej schematami.

W tabeli 7.1 pokazano rozmieszczenie sygnałów analogowych i cyfrowych na dwóch złączach modułu NI USB-6001. Wykorzystywane przez nas styki zaznaczono pogrubioną czcionką. Na lewym złączu sygnałów analogowych wykorzystujemy dwa wejścia analogowe pracujące w trybie różnicowym: AI 2 (PIN8, PIN9) oraz AI 3 (PIN 11, PIN 12). Dodatkowo do pomiarów analogowych służy nam masa analogowa AI GND (PIN10).

Na prawym złączu sygnałów cyfrowych wykorzystujemy dwie linie cyfrowe pracujące w trybie wyjściowym: P 0.0 (PIN17) oaz P 0.1 (PIN18). Służą one do sterowania pracą grzałki i wentylatora. Z kolei do zasilania układu pomiarowego z czujnikiem temperatury służy nam napięcie +5 V, dostępne na styku PIN31, oraz masa D GND.

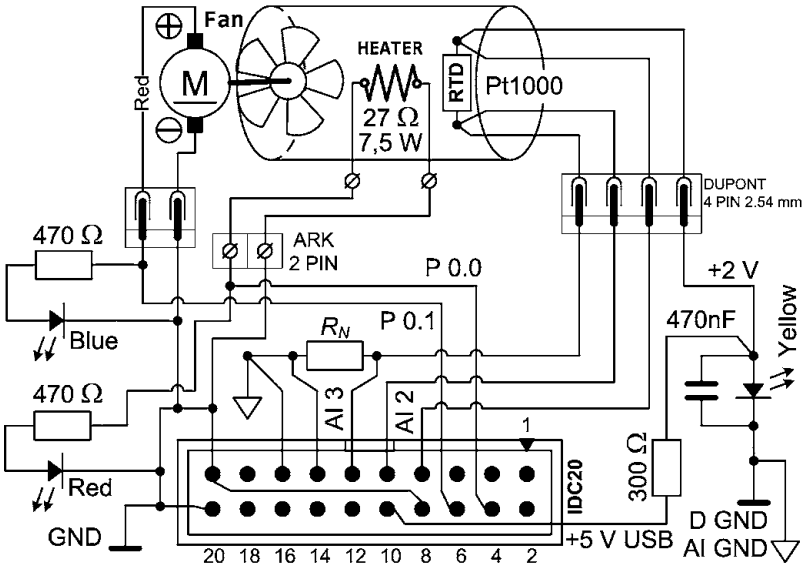
**Tab. 7.1.** Rozmieszczenie sygnałów analogowych (po lewej) i cyfrowych (po prawej) na złączach modułu NI USB-6001, pogrubioną czcionką zaznaczono styki wykorzystane w ćwiczeniu 7 [5]

| Sygnał | PIN | Złącze analogowe i cyfrowe | PIN | Sygnał |
|---|---|---|---|---|
| AI GND | 1 |  | 17 | P0.0 |
| AI 0 (AI 0+) | 2 | | 18 | P0.1 |
| AI 4 (AI 0–) | 3 | | 19 | P0.2 |
| AI GND | 4 | | 20 | P0.3 |
| AI 1(AI 1+) | 5 | | 21 | P0.4 |
| AI 5 (AI 1–) | 6 | | 22 | P0.5 |
| AI GND | 7 | | 23 | P0.6 |
| **AI 2(AI 2+)** | **8** | | 24 | P0.7 |
| **AI 6 (AI 2–)** | **9** | | 25 | P1.0 |
| **AI GND** | **10** | | 26 | P1.1/PFI 1 |
| **AI 3(AI 3+)** | **11** | | 27 | P1.2 |
| **AI 7 (AI 3–)** | **12** | | 28 | P1.3 |
| AI GND | 13 | | 29 | P2.0/PFI 0 |
| AO 0 | 14 | | 30 | D GND |
| AO 1 | 15 | | **31** | **+5V/150mA** |
| AO GND | 16 | | **32** | **D GND** |



Na rycinie 7.7 przedstawiono styki gniazda IDC20, wykorzystywane w module z grzałką i wentylatorem. W trakcie tego ćwiczenia wykorzystujemy łącznie 11 z nich. Prawidłowe dołączenie modułu do panelu zapewnia sama konstrukcja gniazda IDC20. Musimy pamiętać, że w systemie tym wykorzystujemy tzw. klucz sprzętowy, wymuszający właściwy sposób dołączania wtyczek do gniazd. Dodatkowo styk numer 1 (zarówno na gnieździe, jak i we wtyczce) zaznaczono kolorem czerwonym.

**Próba nieprawidłowego dołączenia wtyczki spowoduje uszkodzenie gniazda!**




**Ryc. 7.7.** Dołączenia sygnałów do złącza IDC w module z grzałką i wentylatorem

## 7.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 7.4.1. Uruchamianie stanowiska

Rozpoczynając realizację ćwiczenia, w pierwszej kolejności sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Upewniamy się, że zasilanie komputera i panelu ćwiczeniowego jest wyłączone, a moduł pomiarowy – odłączony od portu USB**.

Wszelkie zauważone odstępstwa od tego stanu zgłaszamy prowadzącemu zajęcia. Następnie sprawdzamy układ zgodnie z opisem z poprzedniego punktu. Jeśli wszystko jest połączone prawidłowo, zgłaszamy gotowość do rozpoczęcia ćwiczenia.

**Dopiero po uzyskaniu wyraźnej zgody prowadzącego włączamy zasilanie samego komputera. Na tym etapie nie załączamy jeszcze zasilania panelu ćwiczeniowego ani nie dołączamy modułu pomiarowego do portu USB.**

Gdy system operacyjny się uruchomi, powstrzymujemy się od jakichkolwiek działań i przechodzimy do kolejnego kroku instrukcji.

### 7.4.2. Self-test modułu NI USB-6001

Po uruchomieniu komputera w pierwszej kolejności sprawdzamy, czy w systemie są zainstalowane dwa urządzenia NI USB-6001: jedno fizyczne (jako **Dev1**) i drugie symulowane (jako **Dev2**). W tym celu uruchamiamy program NI MAX, rozwijamy listę **Devices and Interfaces** i sprawdzamy zainstalowane urządzenia. W protokole notujemy nazwy, typy, numery seryjne i statusy zainstalowanych urządzeń z serii NI USB. Ponieważ moduł na panelu ćwiczeniowym pozostaje na tym etapie odłączony, na liście figuruje przy nim czerwony znak „x" oraz widnieje status **Not Present**. Jeśli zestawienie nie zawiera wymaganych urządzeń, instalujemy je ponownie, postępując zgodnie z opisem w instrukcji do ćwiczenia 1.

Musimy pamiętać, że moduł zainstalowany w systemie na wcześniejszych zajęciach mógł zostać z nieznanych powodów (najczęściej przypadkowo) usunięty przez innych studentów. Z tego względu na początku zajęć zawsze upewniamy się, jaki numer przypisano w systemie do naszego modułu pomiarowego. Sprawdzamy również, czy pracujemy na tym samym stanowisku laboratoryjnym, gdyż każdy moduł DAQ ma swój indywidualny numer seryjny. Wszystkie przykłady programów zamieszczone w instrukcji wykorzystują moduł fizyczny o nazwie Dev1. Jeśli więc nasz moduł figuruje pod inną nazwą, musimy odpowiednio zmodyfikować tworzone programy.

Zgłaszamy gotowość prowadzącemu i po uzyskaniu jego zgody dołączamy moduł NI USB-6001 do portu USB. Nie załączamy jeszcze zasilania na panelu ćwiczeniowym. Obserwujemy stan niebieskiej kontrolki LED na obudowie modułu pomiarowego. Na dołączonym module ćwiczeniowym świeci wówczas żółta dioda, natomiast czerwona i niebieska pozostają zgaszone. Wyniki tych obserwacji notujemy w protokole.

Jeśli nasze urządzenie NI USB-6001 jest poprawnie zainstalowane, w programie NI MAX przeprowadzamy self-test modułu fizycznego i symulowanego (zgodnie z instrukcją do ćwiczenia 1). Wyniki testu notujemy w protokole.

Zgłaszamy gotowość prowadzącemu i po uzyskaniu jego zgody włączamy zasilanie panelu ćwiczeniowego czerwonym przyciskiem kołyskowym. Podświetlenie tego przycisku potwierdza nam obecność napięcia zasilającego.

### 7.4.3. Testowanie przetwornika analogowo-cyfrowego

Test przetwornika analogowo-cyfrowego przeprowadzamy podobnie jak w ćwiczeniu 1, jednak tym razem mierzymy dwa napięcia: na czujniku temperatury Pt1000 i na rezystorze wzorcowym $R_N$. Żółta dioda LED sygnalizuje obecność napięcia zasilającego +5 V oraz jednocześnie pełni rolę stabilizatora. Połączone szeregowo czujnik temperatury i rezystor wzorcowy charakteryzują się zbliżoną rezystancją (około 1000 Ω). Zasilamy je napięciem ok. 2 V, odkładającym się na żółtej diodzie. Jeśli stanowisko działa poprawnie, napięcie na każdym z tych elementów wynosi w przybliżeniu 1 V.

Najpierw sprawdzamy napięcie na czujniku Pt1000. W oknie programu NI MAX wybieramy z listy urządzeń moduł **Dev1** i otwieramy zakładkę **Test Panels**. W wyświetlonym panelu przechodzimy do zakładki **Analog Input** i ustawiamy następujące parametry: **Channel Name: Dev1/ai2, Mode: Finite, Input Configuration: Differential, Max Input Limit: 10, Min Input Limit: −10, Rate (Hz): 1000, Samples To Read: 100.** Uruchamiamy test przyciskiem **Start**. Otrzymany wynik powinien wynosić około 1 V (z widocznymi zakłóceniami rzędu 1 mV). Następnie powtarzamy procedurę dla rezystora wzorcowego $R_N$, wykorzystując wejście **Dev1/ai3**. Zmierzone napięcie ponownie powinno być zbliżone do 1 V, jednak powinno być niższe o kilkadziesiąt miliwoltów od napięcia na czujniku Pt1000. Wynika to z faktu, że w temperaturze pokojowej czujnik Pt1000 charakteryzuje się rezystancją o kilkadziesiąt omów wyższą niż rezystor wzorcowy $R_N$. Zaobserwowane wyniki oraz sformułowane wnioski notujemy w protokole. Nie zamykamy okna **Test Panels**, lecz przechodzimy bezpośrednio do sprawdzania wyjść cyfrowych.

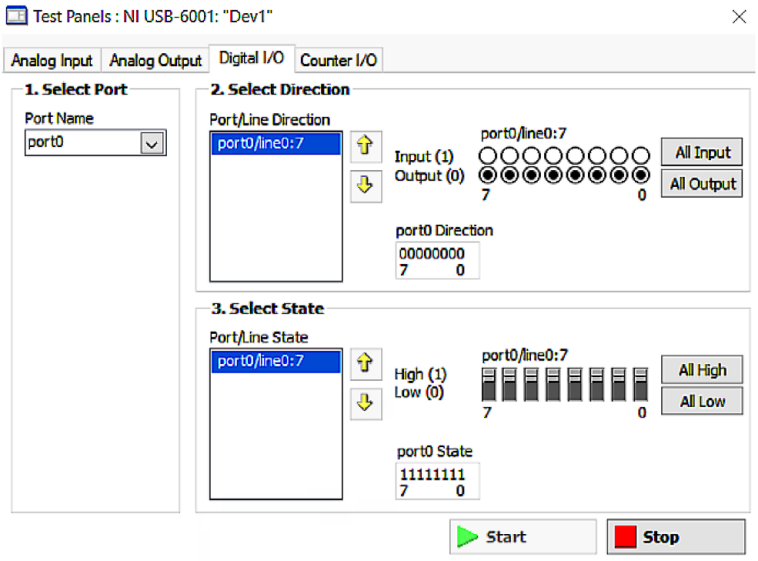
### 7.4.4. Testowanie wyjść cyfrowych

Pozostawiamy otwarte okno **Test Panels** z trwającym testem przetwornika analogowo-cyfrowego w zakładce **Analog Input**. Następnie w głównym oknie programu NI MAX ponownie wybieramy zakładkę **Test Panels**. W nowo otwartym panelu przechodzimy do zakładki **Digital I/O**. Rozmieszczamy oba okna obok siebie na ekranie. Parametry wyjść cyfrowych konfigurujemy zgodnie z ryciną 7.8. Ustawiamy następujące opcje: **1. Select Port > Port Name**: **port0**, **2. Select Direction > All Output**, **3. Select State > All High**. Upewniamy się, że wszystkie linie **port0/line0..7** pracują jako wyjścia **Output (0)**, w stanie wysokim **High (1)**. Uruchamiamy test przyciskiem **Start**. Stan wysoki

wyłącza przekaźniki sterujące grzałką i wentylatorem. Czerwona oraz niebieska dioda na module ćwiczeniowym powinny być zgaszone. Zgaszone powinny być również czerwone diody LED, wlutowane na module z przekaźnikami. Weryfikujemy stan układu, a wyniki notujemy w protokole.

Sprawdzanie układu zaczynamy od pierwszego przekaźnika i grzałki. W sekcji **3. Select State** klikamy przełącznik dla linii **port0/line0**, co ustawia ją w stanie niskim **Low (0)** w oknie **port0 State**. Powoduje to załączenie grzałki, co potwierdza czerwona dioda na module ćwiczeniowym oraz zapalenie się drugiej czerwonej diody LED (oznaczonej jako IN1) na module z przekaźnikami (ryc. 7.4). Następnie przywracamy testowanej linii stan wysoki **High (1)**, co natychmiast wyłącza grzałkę. Wyniki testu oraz nasze obserwacje notujemy w protokole.

Przechodzimy do sprawdzania drugiego przekaźnika i wentylatora. W sekcji **3. Select State** klikamy przełącznik dla linii **port0/line1**, co ustawia ją w stanie niskim **Low (0)** w oknie **port0 State**. Powoduje to uruchomienie wentylatora, co możemy łatwo usłyszeć i zaobserwować. Jednocześnie potwierdza to niebieska dioda na module ćwiczeniowym oraz zapalenie się kolejnej czerwonej diody LED (oznaczonej jako IN2) na module z przekaźnikami. Następnie przywracamy testowanej linii stan wysoki **High (1)**, co natychmiast wyłącza wentylator. Wyniki testu oraz nasze obserwacje notujemy w protokole.




**Ryc. 7.8.** Testowanie wyjść cyfrowych w programie NI MAX

W sekcji **2. Select Direction** wybieramy opcję **All Input**. Wszystkie linie **port0/line0..7** przechodzą wówczas w tryb wejściowy **Input (1)**. Zwracamy uwagę na stany odczytywane na liniach 0 i 1 w porównaniu do pozostałych. Następnie dokładnie obserwujemy czerwone diody na module z przekaźnikami – sprawdzamy, czy się świecą oraz czy grzałka i wentylator się uruchomiły. Próbujemy wyjaśnić zaobserwowane zachowanie układu (zjawisko to wyjaśniono w punkcie 7.1.3). Wyniki tych obserwacji, wnioski oraz odpowiednie uzasadnienie notujemy w protokole.

Zwracamy uwagę, że obwody wejściowe transoptorów zasilamy napięciem +5 V z modułu NI USB-6001, natomiast ich obwody wyjściowe oraz cewki przekaźników – napięciem +5 V z zasilacza sieciowego. Po podłączeniu modułu do portu USB wszystkie jego linie cyfrowe domyślnie przechodzą w tryb wejściowy (**Power-On State**). W przypadku niewłaściwie zaprojektowanych układów sterowania może spowodować to niekontrolowane załączenie obciążenia (szczegóły tego zjawiska podano w punkcie 7.1.3).

Po zakończeniu testów potwierdzających sprawność całego stanowiska zapisujemy wyciągnięte wnioski w protokole. Dodatkowo do raportuwykonujemy zrzut ekranu przedstawiający przebieg testu w obu otwartych oknach: **Analog Input** oraz **Digital I/O**. **Przed przejściem dalej upewniamy się, że grzałka i wentylator są wyłączone.** Zamykamy oba okna **Test Panels** i kończymy pracę w programie NI MAX, po czym uruchamiamy środowisko LabVIEW i przechodzimy do kolejnego punktu.

## 7.5. Programowanie

### 7.5.1. Pierwszy program do pomiaru temperatury czujnikiem termorezystancyjnym

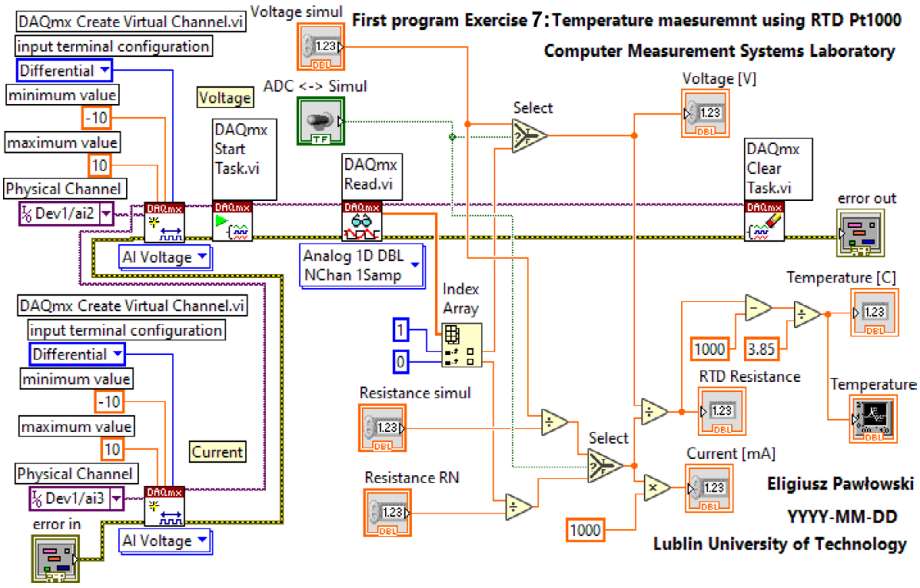
Ćwiczenie 7 stanowi kontynuację ćwiczenia 6, dlatego podstawę do jego rozpoczęcia stanowi drugi opracowany tam program. Wyznacza on zależność prądu od napięcia dla badanego dwójnika oraz oblicza jego rezystancję z prawa Ohma. Odszukujemy ten program, otwieramy okno z jego diagramem i analizujemy ponownie zasadę jego działania. Aplikacja w pętli **For** zadaje napięcie na wyjściu analogowym modułu pomiarowego, po czym mierzy prąd płynący przez dwójnik

oraz spadek napięcia na tym elemencie. Na podstawie tych pomiarów układ generuje dwie tablice z danymi, prezentowane następnie na wykresie. W obecnym ćwiczeniu pomiary realizujemy w bardzo podobny sposób. Badanym dwójnikiem jest tu rezystancyjny czujnik temperatury, połączony szeregowo z rezystorem wzorcowym. Tym razem nie wykorzystujemy jednak przetwornika cyfrowo-analogowego (DAC), ponieważ napięcie zasilające badany dwójnik ma stałą wartość około 2 V. Z tego względu wszystkie wyniki pomiarów zrealizowanych w pętli **For** będą do siebie zbliżone, a zamiast ciągłego wykresu na ekranie pojawią się tylko trzy punkty. Dla napięcia rzędu 1 V prąd wynosi około 1 mA, rezystancja – około 1100 Ω, a moc – około 1 mW. Z kolei wartość rezystancji wzorcowej $R_N$ wynosi 1000 Ω. Tak przygotowany program uruchamiamy w trybie pomiarów rzeczywistych.

Sprawdzamy, czy program działa prawidłowo. Jeśli zgodnie z oczekiwaniami wygenerował on na wykresie tylko trzy punkty, tworzymy jego kopię na potrzeby następnego kroku, nadając jej oznaczenie 7-1-a. Pamiętamy przy tym, by nie modyfikować pliku stanowiącego ostateczną wersję poprzednio wykonanego zadania. Zmiany w panelu i diagramie wprowadzamy wyłącznie w nowo utworzonej kopii 7-1-a. Na potrzeby bieżącego ćwiczenia usuwamy z niego wszystkie funkcje związane z obsługą wyjścia analogowego (DAC), tworzeniem tablic z wynikami pomiarów i kreśleniem wykresu. Przykładowy diagram tak zmodyfikowanego programu pokazano na rycinie 7.9. Cały proces przekształceń realizujemy zgodnie z poniższymi etapami. Najpierw usuwamy z diagramu następujące funkcje w kanale wyjść analogowych przetwornika DAC: **DAQmx_Create_Virtual_Channel**, **DAQmx_Start_Task**, **DAQmx_Write** (dwa razy), **DAQmx_Clear_Task** i **Simple Error Handler**. Następnie usuwamy pętlę **For** oraz wszystkie funkcje użyte do zbudowania wykresu **XYGraph**: **Initialize Array** (dwa razy), **Build Array** (trzy razy), **Bundle** (trzy razy). Usuwamy także fragment tworzący opis na wykresie za pomocą węzła właściwości **Property Node** oraz opóźnienie czasowe przed wyzwoleniem pomiaru ADC (**Stall Data Flow**). Na koniec uzupełniamy skasowane połączenia linii **task** oraz linii błędu **error**.

Klikając prawym przyciskiem myszy (**Create > Control**) na wejściowym terminalu linii błędu **error in** w funkcji **DAQmx_Create_Virtual_Channel**, tworzymy zadajnik **error in**. Podobnie tworzymy wskaźnik błędu **error out**, klikając prawym przyciskiem myszy (**Create > Indicator**) na wyjściowym terminalu linii błędu **error out** w funkcji **DAQmx_Clear_Task**.

Na koniec uzupełniamy diagram o obliczanie temperatury na podstawie rezystancji czujnika Pt1000, wykorzystując w tym celu wzór (7.4). Na panelu programu dodajemy wskaźnik numeryczny (**Numeric Indicator**) zmierzonej temperatury oraz wskaźnik wykresu typu **Waveform Chart**.

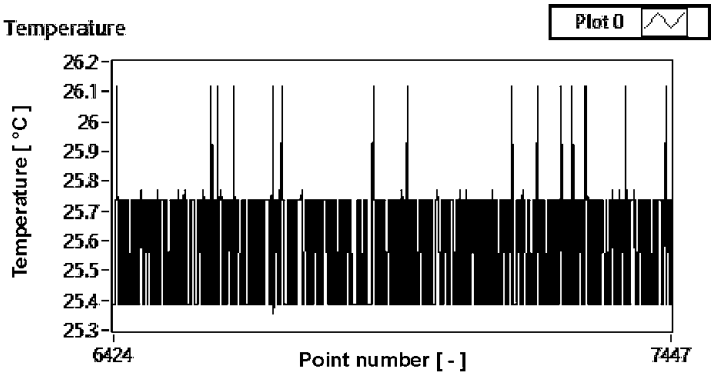



**Ryc. 7.9.** Pierwszy krok modyfikacji diagramu do pomiaru temperatury

Uruchamiamy program do pracy ciągłej odpowiednim przyciskiem  . Sprawdzamy, czy na ekranie wyświetlają się właściwe ikony  i  , i upewniamy się, że całość działa prawidłowo. Zmierzony prąd wynosi w przybliżeniu 1 mA, z kolei w temperaturze pokojowej (25°C) rezystancja czujnika Pt1000 osiąga wartość około 1096 Ω. Przy użyciu kalkulatora wykonujemy obliczenia kontrolne, aby potwierdzić poprawne działanie programu. Otrzymane wyniki oraz same obliczenia notujemy w protokole. Następnie analizujemy wykres zmierzonej temperatury. Zapisujemy go do pliku BMP (korzystając z opcji **Export Simplified Image**), aby móc go później dołączyć do raportu.

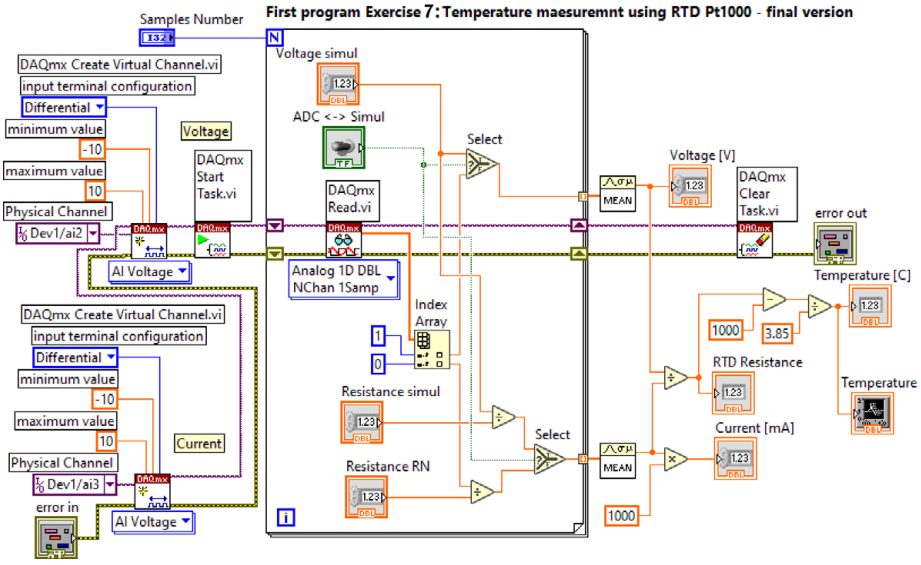
Przykład wykresu zmierzonej temperatury pokazano na rycinie 7.10. Jak widzimy na wykresie, rejestrowana temperatura charakteryzuje się stosunkowo dużym rozrzutem wartości, osiągającym prawie 1°C. Znanym z podstaw metrologii sposobem na polepszenie dokładności jest uśrednianie wyników z całej serii pomiarów. Z właściwości rozkładu normalnego wynika, że niepewność

maleje proporcjonalnie do pierwiastka kwadratowego z liczby uśrednianych próbek. Niekorzystnym efektem tego zabiegu jest jednak wydłużanie całkowitego czasu pomiaru.




**Ryc. 7.10.** Wykres zmierzonej temperatury pierwszą wersją programu

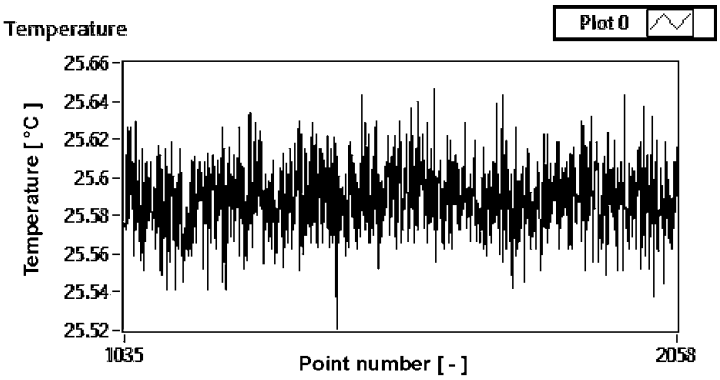
Zmodyfikujemy teraz program, dodając pętlę **For** do wykonania serii pomiarów oraz funkcję **Mean** do uśredniania wyników. Najpierw tworzymy kopię programu, nadając jej oznaczenie 7-1-b. Przykładowy diagram tak zmodyfikowanego programu przedstawia rycina 7.11. Cały proces realizujemy zgodnie z poniższymi etapami.




**Ryc. 7.11.** Drugi krok modyfikacji diagramu do pomiaru temperatury z uśrednianiem

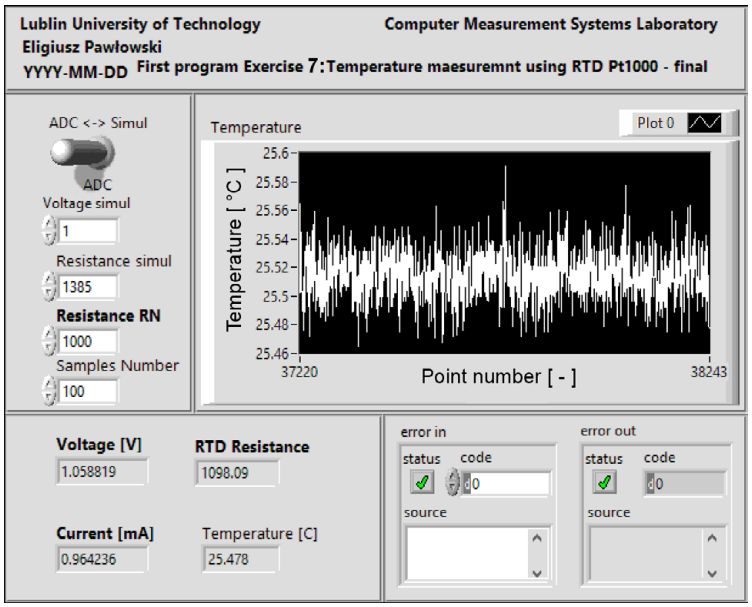
Najpierw uzupełniamy program o pętlę **For**. Zwracamy przy tym uwagę na prawidłowe utworzenie rejestrów przesuwnych i tuneli. Tunele powinny pracować w trybie **Indexing**. Na zewnątrz pętli **For** dodajemy w obu kanałach funkcję obliczania wartości średniej **Mean**. Na wejściu pętli **For** tworzymy zadajnik (**Create > Control**) i ustawiamy jego początkową wartość (czyli liczbę uśrednianych pomiarów) na 10.

Uruchamiamy program do pracy ciągłej przyciskiem  . Sprawdzamy, czy na ekranie wyświetlają się właściwe ikony  i  , i upewniamy się, że całość działa prawidłowo. Zwracamy uwagę na wykres zmierzonej temperatury. Podczas działania programu zmieniamy liczbę uśrednianych pomiarów i obserwujemy zachodzące zmiany. Większa liczba uśrednianych pomiarów powinna skutkować mniejszymi rozrzutami wartości zmierzonej temperatury. Niestety, jednocześnie wydłuża się czas pomiaru, przez co program zaczyna działać coraz wolniej. Przeprowadzamy eksperyment, zmieniając liczbę uśrednianych pomiarów na: 2, 5, 10, 20, 50, 100, 200, 500 itd., i oceniamy, jaka maksymalna ich liczba jest możliwa do realizacji. Zaobserwowane rozrzuty wyników dla różnych długości serii notujemy w protokole. Ustalamy optymalną liczbę uśrednianych pomiarów, znajdując kompromis pomiędzy dokładnością a szybkością działania programu. Otrzymany wykres zapisujemy do pliku, aby wykorzystać go w raporcie. Przykład takiego wykresu (dla uśredniania 100 pomiarów) pokazano na rycinie 7.12. Jak widzimy, temperatura charakteryzuje się na nim znacznie mniejszym rozrzutem wartości (nieprzekraczającym 0,14°C). Porównujemy ten wynik z wykresem z ryciny 7.10.




**Ryc. 7.12.** Wykres zmierzonej temperatury z uśrednianiem 100 pomiarów

Uaktualniamy odpowiednio panel i diagram i uzupełniamy je niezbędnymi informacjami: nazwą przedmiotu, uczelnią, danymi osobowymi zespołu, numerem grupy dziekańskiej, datą i numerem ćwiczenia, a także numerem programu i jego tytułem. Informacje te podajemy w formie komentarza umieszczonego na panelu i diagramie. Porządkujemy rozmieszczenie kontrolek i wskaźników na panelu, uwzględniając ich logiczne powiązania. Używamy narzędzi **Align**, **Distribute**, **Resize** oraz **Reorder**, aby wyrównać położenie obiektów na panelu, pogrupować je (**Group**) i zablokować ich pozycję (**Lock**). Przykładowy panel tak uzupełnionego programu przedstawiono na rycinie 7.13.




**Ryc. 7.13.** Panel programu do pomiaru temperatury z uśrednianiem

Program zapisujemy w katalogu wskazanym przez prowadzącego. Nazwy plików z programami tworzymy według następującego schematu:

YYYYMMDD_Nazwisko_KSP_X-N-L.vi

gdzie: YYYY –rok, MM –miesiąc, DD –dzień, X – numer ćwiczenia, N – numer programu w realizowanym ćwiczeniu, L – litera (a, b, c itd.) dla kolejnych wersji tego samego programu. W nazwach plików nie wolno używać

polskich znaków, spacji ani innych znaków specjalnych. Zapisaną nazwę oraz wybraną ścieżkę katalogu notujemy w protokole.

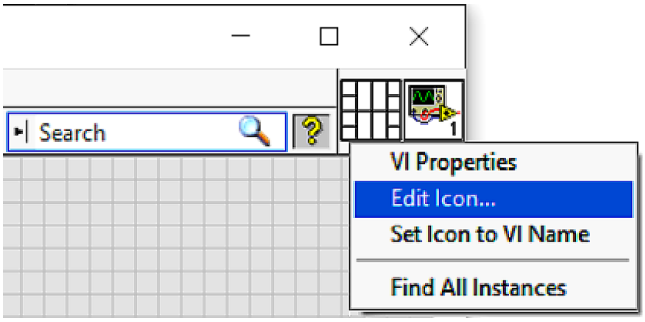
Wybierając opcję **File > Print > VI Documentation**, zapisujemy do pliku dokumentację programu (obejmującą panel i diagram). Następnie odszukujemy te pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików wraz z opisem ich zawartości notujemy w protokole. Zgromadzone materiały wykorzystujemy później do sporządzenia końcowego raportu.

### 7.5.2. Tworzenie podprogramu w środowisku LabVIEW

LabVIEW to środowisko umożliwiające programowanie hierarchiczne. Oznacza to, że każdy utworzony w nim program możemy przekształcić w podprogram, a następnie wykorzystać jako moduł na diagramie połączeń kolejnej budowanej aplikacji. Program w tym środowisku nazywamy przyrządem wirtualnym (**Virtual Instrument**) i oznaczamy skrótem VI, natomiast podprogramy określamy krótko jako subVI. W praktyce każdy VI składa się z wielu subVI, przy czym te z kolei również mogą zawierać kolejne warstwy zagnieżdżonych subVI. Siła i elastyczność LabVIEW tkwią właśnie w hierarchicznej, modułowej naturze programów VI. Dzięki temu jako użytkownicy możemy dowolnie tworzyć kolejne subVI, rozszerzając w ten sposób zestaw standardowych funkcji dostępnych w środowisku.Każda funkcja pobrana z zasobnika funkcji i użyta na diagramie wyróżnia się na tle pozostałych trzema elementami: indywidualną nazwą, spersonalizowaną ikoną oraz zestawem zacisków (terminali) wejściowych i wyjściowych. Tworząc własny subVI, nadajemy mu zatem indywidualną nazwę, projektujemy komunikatywną ikonę oraz wybieramy odpowiedni zestaw terminali wejściowych i wyjściowych. Proces ten realizujemy krok po kroku.

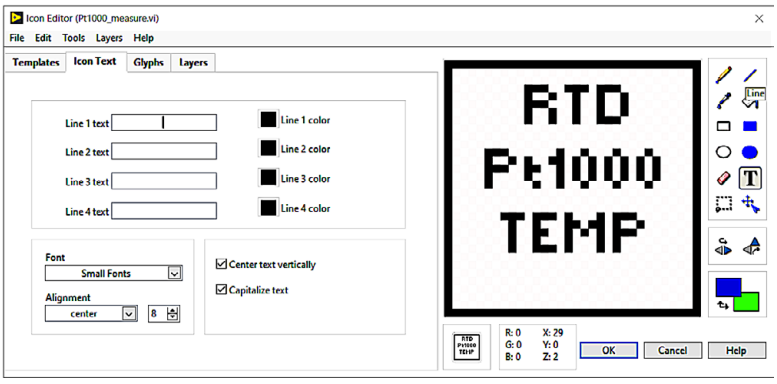
Tworzony samodzielnie subVI otrzymuje automatycznie nazwę identyczną z nazwą pliku, w którym zapisujemy dany program. Ze względu na wygodę użytkowania podprogramu nazwa ta powinna być krótka i komunikatywna. Dlatego w pierwszej kolejności zapisujemy opracowany wcześniej program pod nową, krótką i sensowną nazwą (np. **Pt1000_measure.vi** lub inną, według własnego uznania). **Uwaga:** w nazwie pliku nie wolno stosować spacji, polskich znaków ani innych znaków specjalnych, aby środowisko LabVIEW poprawnie rozpoznawało nazwę naszego subVI. Wybraną nazwę pliku notujemy w protokole; będzie ona niezbędna podczas wykorzystywania utworzonego podprogramu w kolejnych zadaniach.

Każdy nowo tworzony program VI otrzymuje również automatycznie domyślną ikonę oraz standardowy zestaw terminali. Są one widoczne w prawym górnym rogu okna LabVIEW, co pokazano na rycinie 7.14.




**Ryc. 7.14.** Domyślna ikona i panel złącza nowo utworzonego VI

Ikona ta wygląda identycznie jak ta, która pojawia się po umieszczeniu subVI na diagramie połączeń w innym programie. Stanowi ona zatem graficzną reprezentację podprogramu i może zawierać tekst, rysunek lub kombinację obu tych elementów. Aby łatwo odróżnić tworzony subVI od pozostałych funkcji na diagramie, zmieniamy domyślną ikonę na spersonalizowaną, nawiązującą swoim wyglądem do realizowanego zadania. Do tego celu służy wbudowany w LabVIEW dedykowany edytor graficzny. Uruchamiamy go (co przedstawionona rycinie 7.14) klikając prawym przyciskiem myszy edytowaną ikonę i wybierając z listy opcję **Edit Icon...**. Można również od razu kliknąć dwa razy na ikonie. Otwiera się wówczas okno edytora, widoczne na rycinie 7.15.




**Ryc. 7.15.** Edytowanie ikony dla tworzonego podprogramu subVI do pomiaru temperatury czujnikiem RTD typu Pt1000

Ikona jest rysunkiem rastrowym o stosunkowo niewielkiej rozdzielczości 32 × 32 pikseli. Dlatego starannie planujemy jej wygląd, aby jednoznacznie identyfikowała nasz podprogram. Wykorzystując standardowe narzędzia (znane z popularnych programów graficznych), tworzymy prostą grafikę lub wprowadzamy krótki tekst.

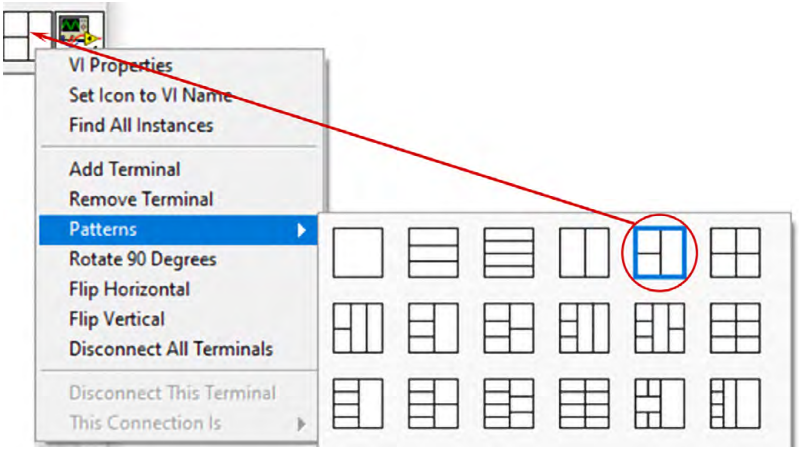
Używamy narzędzi dostępnych po prawej stronie okna edytora, aby w powiększonym obszarze utworzyć własny projekt. Obraz ikony w docelowym rozmiarze podglądamy w polu poniżej, co ilustruje rycina 7.15. Tworzymy ikonę możliwie prostą, ale zarazem jednoznacznie reprezentującą funkcję realizowaną przez nasz podprogram. Na wspomnianej rycinie 7.15 przedstawiono przykładowy projekt dla subVI mierzącego temperaturę za pomocą czujnika RTD typu Pt1000. Wykazujemy się w tym miejscu własną inicjatywą i prezentujemy nasze zdolności artystyczne (nawet jeśli studiujemy na politechnice).

Aby użyć programu VI jako podprogramu (subVI) w innym programie nadrzędnym, definiujemy również odpowiedni panel złącza (**Connector Pane**). Panel ten znajduje się w prawym górnym rogu okna panelu programu (**Front Panel**), tuż po lewej stronie ikony, co widać na rycinie 7.14. Stanowi on zestaw terminali odpowiadających elementom sterującym i wskaźnikom tworzonego subVI – działa to analogicznie jak lista parametrów wywołania funkcji w tekstowych językach programowania. To właśnie w nim określamy zaciski wejściowe i wyjściowe (**terminals**), umożliwiające następnie wykonanie odpowiednich połączeń na diagramie programu nadrzędnego.

Zdefiniujemy teraz wymagane połączenia, przypisując odpowiedni element sterujący lub wskaźnik panelu przedniego do każdego z terminali panelu złącza. Najpierw wybieramy dla tworzonego subVI odpowiedni wzór terminali panelu złącza (**Connector Pane Terminal Pattern**), co pokazano na rycinie 7.16.

Klikając panel złącza prawym przyciskiem myszy, otwieramy rozwijane menu i wybieramy opcję **Patterns**. Z dostępnej listy wybieramy odpowiedni wzór terminali (wzór aktualnie skojarzony z ikoną naszego subVI podświetla się w pełnej niebieskiej obwódce). Wybierając wzór dla naszego podprogramu, planujemy terminale wejściowe po lewej stronie, a wyjściowe po prawej. Jak pokazano wcześniej na rycinie 7.16, wybrany wzorzec możemy obracać, odbijać w pionie lub poziomie, a także dodawać lub usuwać w nim pojedyncze terminale. Liczba użytych terminali odpowiada liczbie wejść i wyjść

tworzonego podprogramu. Dla tworzonego podprogramu planujemy wejście i wyjście dla linii błędu oraz jedno wyjście dla wyniku pomiaru temperatury. Optymalny okazuje się wzór z dwoma wejściami po stronie lewej i jednym wyjściem po prawej (który następnie odbijamy lustrzanie). Oczywiście możemy wybrać inny wzór i zdefiniować dodatkowe wejścia/wyjścia (np. dla liczby uśrednianych pomiarów, rezystora wzorcowego czy wyniku pomiaru prądu). Wprawdzie każdy zadajnik na panelu możemy przypisać do terminala wejściowego, a wskaźnik do wyjściowego, to jednak rozsądnie jest ograniczyć ich liczbę do niezbędnego minimum, co znacznie ułatwia późniejsze korzystanie z subVI.

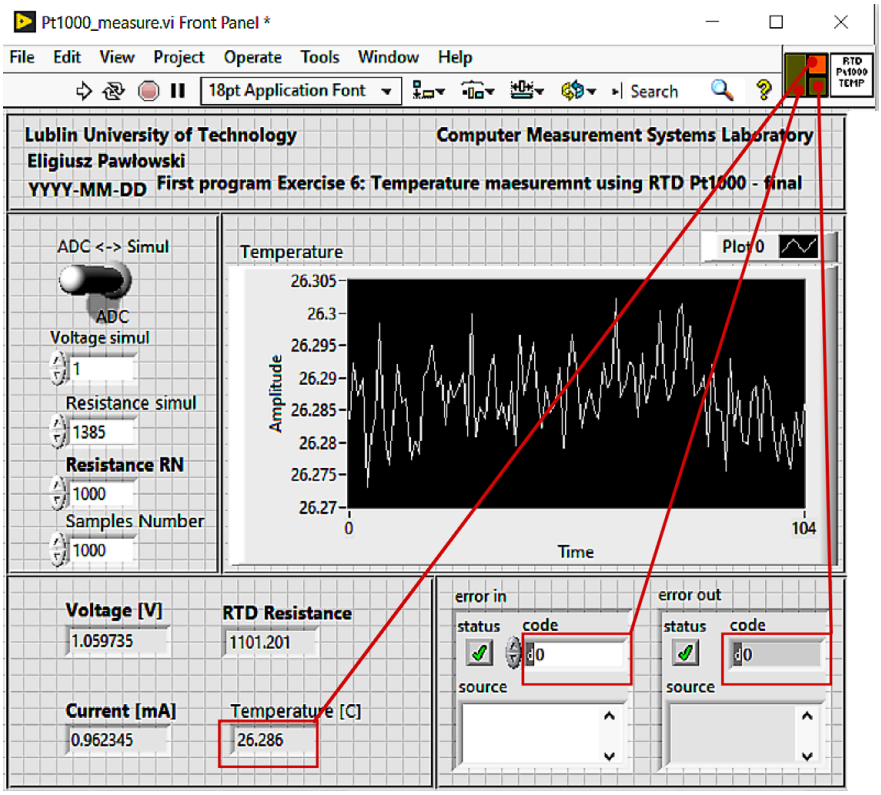



**Ryc. 7.16.** Wybieranie odpowiedniego wzoru terminali dla panelu złącza

Po wybraniu pożądanego wzoru definiujemy połączenia, przypisując wybrane kontrolki i wskaźniki z panelu programu VI do właściwych terminali na panelu złącza. Sposób postępowania w tym zakresie przedstawiono na rycinie 7.17.

Każdy prostokąt na panelu złącza reprezentuje jeden terminal. Aby przypisać go do kontrolki lub wskaźnika na panelu programu, wykonujemy odpowiednie połączenie narzędziem w kształcie szpulki z drutem. W tym celu klikamy wybrany terminal na panelu złącza, a następnie klikamy docelową kontrolkę lub wskaźnik na panelu programu. Wybór zatwierdzamy, klikając w dowolne puste miejsce na panelu programu. Terminal zmienia wówczas swój początkowy, biały kolor na barwę odpowiadającą typowi podłączonych danych, co z kolei potwierdza poprawne zdefiniowanie połączenia. Możemy

również odwrócić tę kolejność: najpierw wskazać kontrolkę lub wskaźnik, a potem terminal. Pamiętamy przy tym, że choć korzystamy z narzędzia przypominającego szpulkę z drutem, program nie rysuje w tym miejscu żadnych fizycznych przewodów. Zgodnie z dobrą praktyką, podczas przypisywania terminali wejściowych grupujemy je po lewej stronie panelu złącza, natomiast wyjściowych po prawej. Połączenia linii błędu tradycyjnie umieszczamy na terminalach dolnych. Pozwala to uniknąć późniejszego chaosu i plątaniny przewodów na docelowym diagramie programu nadrzędnego.




**Ryc. 7.17.** Definiowanie połączeń terminali wejściowych i wyjściowych

Podsumowując, kolejność postępowania jest następująca: używając narzędzia ze szpulką, klikamy lewy terminal na panelu złącza. Następnie klikamy kontrolkę **error in** na panelu programu. Klikamy puste miejsce na panelu programu, a wybrany terminal zmienia kolor na ciemnozielony. Klikamy prawy dolny terminal na panelu złącza. Następnie klikamy wskaźnik **error out** na panelu programu. Klikamy puste miejsce na panelu programu, a wybrany

terminal zmienia kolor na ciemnozielony. Klikamy prawy górny terminal na panelu złącza, a następnie klikamy wskaźnik **Temperature [C]** na panelu programu. Klikamy puste miejsce na panelu programu, po czym wybrany terminal powinien zmienić kolor na pomarańczowy.

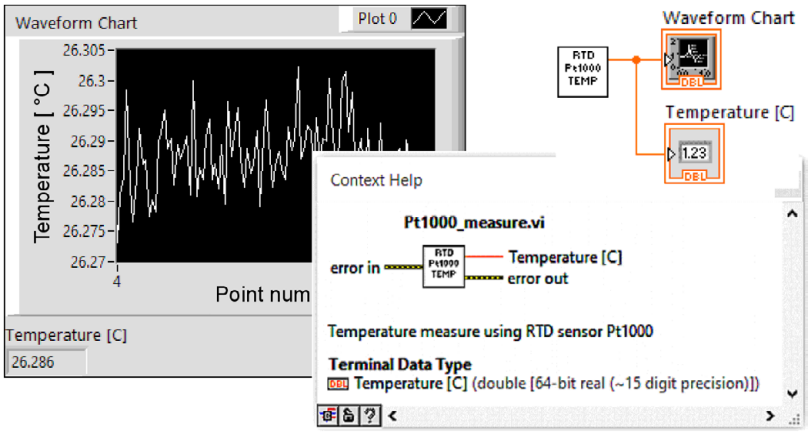
W ten sposób wykonaliśmy niezbędne połączenia. W razie utworzenia jednego lub kilku błędnych połączeń możemy je usunąć pojedynczo lub wszystkie razem. Odpowiednie opcje znajdujemy w rozwijanym menu pokazanym na rycinie 7.16.

Na koniec dodajemy opis utworzonego podprogramu subVI, który będzie się pojawiał w oknie **Context Help**. Klikając ikonę podprogramu prawym przyciskiem myszy, otwieramy dodatkowe menu (przedstawione na rycinie 7.14) i wybieramy opcję **VI Properties**. Z rozwijanej listy wybieramy zakładkę **Documentation**. W polu **VI description** wprowadzamy krótki opis działania naszego podprogramu (np. „Temperature measure using RTD sensor Pt1000" lub inny, adekwatny do realizowanych przez niego funkcji).

Zapisujemy subVI do pliku. Jest on gotowy do użycia w innym programie, co wykorzystamy w dalszej części ćwiczenia. Wcześniej jednak upewniamy się, że stworzony podprogram subVI faktycznie działa prawidłowo. W tym celu realizujemy jak najprostszy program z użyciem naszego subVI i testujemy jego działanie.

W menu głównym LabVIEW wybieramy opcję **File > NewVI**. Otwiera się nowe, puste okno do tworzenia programu. Przełączamy się na diagram (**Ctrl+E**). Klikając pusty diagram prawym przyciskiem myszy, wybieramy opcję **Functions > Select a VI...**. Wyszukujemy plik z utworzonym subVI po nazwie i umieszczamy go na diagramie. Otwieramy okno **Context Help** i sprawdzamy, czy wyświetla się w nim odpowiednia informacja. Widzimy tam nazwę subVI, opisy terminali wejściowych i wyjściowych oraz opis działania podprogramu. Opisy terminali są zgodne z nazwami kontrolek na panelu subVI. Upewniamy się, że wejścia i wyjścia znajdują się we właściwych miejscach na ikonie. Przykład ten pokazany jest na rycinie 7.18. Klikając prawym przyciskiem myszy wyjście pomiaru temperatury, tworzymy wskaźnik (**Create Indicator**). Uruchamiamy program i sprawdzamy, czy temperatura jest mierzona poprawnie. Umieszczamy na panelu programu wskaźnik graficzny (**Cotrols > Graph > Waveform Chart**) i łączymy go z wyjściem pomiaru temperatury. Uruchamiamy program do pracy ciągłej i sprawdzamy,

czy generuje się poprawny wykres temperatury. Przykładowy efekt testowania pracy podprogramu pokazano na rycinie 7.18. Wykonany zrzut ekranu zapisujemy do pliku, aby wykorzystać go w raporcie.




**Ryc. 7.18.** Podprogram subVI zastosowany w programie nadrzędnym i jego opis w okienku pomocy Context Help

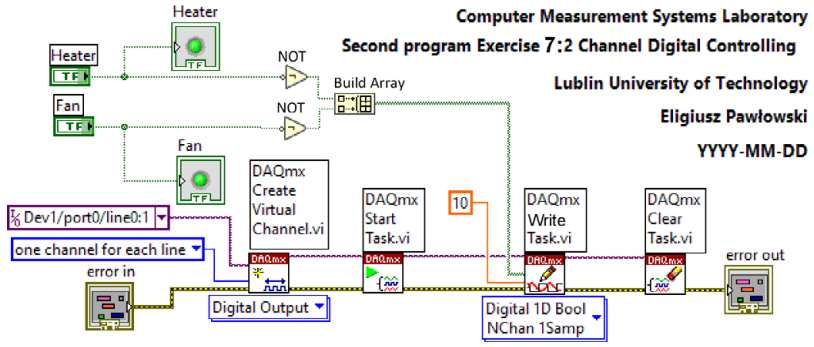
Przy działającym programie klikamy dwukrotnie ikonę subVI. Otwiera się wówczas okno z panelem podprogramu, widoczne na rycinie 7.13. Zmieniamy w nim liczbę uśrednianych pomiarów. Sprawdzamy efekt na panelu programu nadrzędnego. Sformułowany wniosek notujemy w protokole. Zamykamy wszystkie programy i przechodzimy do realizacji kolejnego punktu.

### 7.5.3. Drugi program do sterowania grzałką i wentylatorem

Aby automatycznie sterować grzałką i wentylatorem chłodzącym w środowisku LabVIEW przy użyciu modułu NI USB-6001, najpierw przygotowujemy program umożliwiający ręczne sterowanie za pomącą kontrolek umieszczonych na panelu programu. Następnie przekształcamy ten program w podprogram, który wykorzystamy w docelowej aplikacji do automatycznej regulacji temperatury grzałki.

Otwórz nowy, pusty VI poleceniem **File > NewVI**. Pusty program od razu zapisujemy do pliku, nadając mu nazwę z numerem 7-2-a. Następnie budujemy diagram programu przedstawiony na rycinie 7.19. Zadanie to realizujemy etapami, w opisanej poniżej kolejności.

Najpierw konfigurujemy cyfrowy kanał wyjściowy do sterowania dwiema liniami w porcie P0 modułu NI USB-6001, zgodnie ze schematem blokowym stanowiska przedstawionym na rycinie 7.3. W tym celu wybrany kanał fizyczny (**physical channel**), powiązany z określonymi zaciskami na złączu modułu pomiarowego (zgodnie z tabelą 7.1), przypisujemy do kanału wirtualnego (**virtual channel**), który reprezentuje go następnie na diagramie w uruchomionym zadaniu (**task**).




**Ryc. 7.19.** Diagram programu do sterowania dwoma liniami cyfrowymi

Sposób postępowania jest następujący. Najpierw pobieramy z zasobnika **Functions > Measurement I/O > NI DAQmx** i umieszczamy na diagramie następujące cztery funkcje: **DAQmx Create Virtual Channel**, **DAQmx Start Task**, **DAQmx Write** oraz **DAQmx Clear Task**. Zaznaczamy wszystkie funkcje i wyrównujemy je poleceniem **Align > Top Edges**  . Klikając dodane funkcje prawym przyciskiem myszy, włączamy w nich widoczność etykiety (Visible Items > Label).

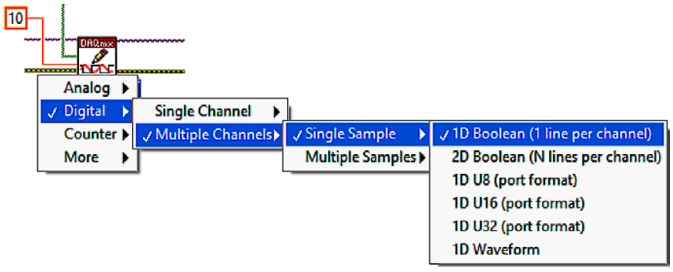
Teraz ustalamy, że będzie tworzony program do obsługi cyfrowych linii wyjściowych. W tym celu rozwijamy listę opcji poniżej funkcji **DAQmx Create Virtual Channel** i wybieramy pozycję **Digital Output**. Następnie określamy nazwy linii cyfrowych, które posłużą nam do utworzenia kanału wirtualnego. W tym celu klikamy prawym przyciskiem myszy terminal **lines** funkcji **DAQmx Create Virtual Channel**. Z menu wybieramy opcję **Create > Constant**. Z rozwijanej listy utworzonego zadajnika wybieramy pozycję **Dev1/port0/line0**. Niestety, nie możemy od razu wybrać opcji sterowania dwiema liniami – wymaga to od nas ręcznej edycji wartości tego zadajnika. Klikamy dwukrotnie okno z nazwą wybranej linii, co umożliwia jej ręczną

edycję. Dokonujemy odpowiedniej edycji i zamieniamy końcówkę tekstu z pojedynczą linią **line0** na dwie linie **line0:1**.

Teraz konfigurujemy oddzielny kanał wirtualny dla każdej linii cyfrowej. W tym celu na lewej krawędzi funkcji **DAQmx Create Virtual Channel** klikamy prawym przyciskiem myszy terminal **line grouping** i wybieramy opcję **Create > Constant**. Z rozwijanej listy utworzonego zadajnika wybieramy pozycję **one channel for each line**.

Klikając prawym przyciskiem myszy wejściowy terminal linii błędu **error in** w funkcji **DAQmx Create Virtual Channel**, tworzymy zadajnik **error in** (opcja **Create > Control**). Podobnie tworzymy wskaźnik błędu **error out**, klikając prawym przyciskiem myszy wyjściowy terminal linii błędu **error out** w funkcji **DAQmx Clear Task** (opcja **Create > Indicator**). Wykonujemy brakujące połączenia dla linii danych (**task out – task in**) oraz linii błędu (**error out – error in**), zgodnie z ryciną 7.19.

Konfigurujemy funkcję **DAQmx Write** do zapisu danych w dwóch kanałach cyfrowych. Sposób jej prawidłowej konfiguracji przedstawiono na rycinie 7.20. W tym celu rozwijamy listę opcji poniżej tej funkcji i wybieramy kolejno pozycje: **Digital > Multiple Channels > Single Sample/1D Boolean (1 line per channel)**. Następnie, klikając prawym przyciskiem myszy terminal **timeout**, tworzymy stałą (**Create > Constant**) i nadajemy jej wartość 10 (sekund).




**Ryc. 7.20.** Konfiguracja funkcji Write do zapisu dwóch linii cyfrowych

Przechodzimy do tworzenia panelu programu. Panel zawiera na razie tylko kontrolkę i wskaźnik do obsługi linii błędów. Umieszczamy na panelu dwie kontrolki logiczne (**Controls > Boolean > Vertical Toggle Switch** lub inne podobne). Nadajemy im nazwy **Heater** i **Fan** (lub inne o podobnym znaczeniu). W elektrotechnice tradycyjnie stosujemy łączniki dźwigienkowe, które

w górnej pozycji załączają obciążenie, a w dolnej je wyłączają. Konwencję tę zachowujemy również w naszym programie. Przy każdym łączniku umieszczamy wskaźnik w postaci diody LED (**Controls > Boolean > Round LED** lub innej podobnej). Nadajemy im te same nazwy co łącznikom. Na diagramie ustawiamy łączniki oraz wskaźniki obok siebie i łączymy je ze sobą. Łącznik w górnym położeniu ustawia na swoim wyjściu stan wysoki, który zaświeca połączoną z nim diodę LED. Takie działanie jest poprawne. Na rycinie 7.3 pokazano jednak, że załączenie obciążenia wymaga ustawienia stanu niskiego na właściwym wyjściu. Dlatego przed wysłaniem sygnału z łącznika na wyjście cyfrowe stosujemy negację logiczną **NOT**, aby poprawnie sterować przekaźnikami połączonymi z wentylatorem i grzałką. Następnie używamy funkcji **Build Array**, aby połączyć oba sygnały logiczne w jedną tablicę. Wyjście funkcji **Build Array** dołączamy do wejścia **data** funkcji **DAQmx Write**. Jeśli nie popełniliśmy żadnego błędu, program jest gotowy do uruchomienia.

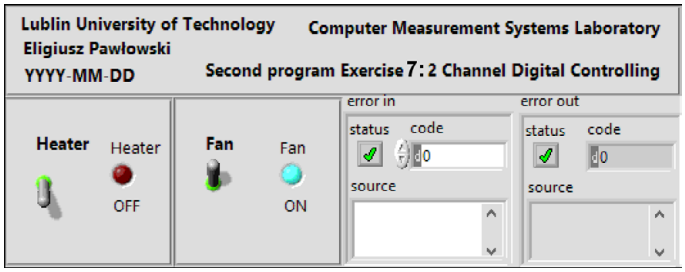
Uruchamiamy program do pracy ciągłej przyciskiem  . Upewniamy się, że wyświetlają się ikony  i  . Sprawdzamy, czy program działa prawidłowo. W dolnych pozycjach przełączników odpowiednie diody LED na panelu programu są zgaszone, a grzałka i wentylator pozostają wyłączone. Przełączenie łącznika w górną pozycję zaświeca odpowiednią diodę LED na panelu i załącza odpowiadające mu obciążenie (zgodnie z opisem łącznika). Upewniamy się, że łącznik opisany jako **Heater** steruje grzałką, a **Fan** steruje wentylatorem. Sformułowane wnioski notujemy w protokole.

Załączamy jednocześnie wentylator i grzałkę, a następnie zatrzymujemy program przyciskiem  . Sprawdzamy, czy wentylator i grzałka uległy wyłączeniu. Zastanawiamy się, jakie byłyby skutki, gdyby błąd w programie spowodował jego zatrzymanie w sytuacji, gdy wcześniej grzałka była włączona. Analizujemy przy tym, czy taka sytuacja byłaby bezpieczna. Sformułowane wnioski notujemy w protokole. Uruchamiamy na chwilę program ponownie i wyłączamy grzałkę oraz wentylator.

Odpowiednio konfigurujemy wskaźniki na panelu programu. Klikając wybraną diodę LED prawym przyciskiem myszy, wybieramy opcję **Properties** i w zakładce **Appearance** ustawiamy kolory zgodne z tymi na panelu ćwiczeniowym (stan wysoki sygnalizujemy jasnym kolorem, a stan niski ciemnym). Włączamy widoczność napisów **ON/OFF**, a następnie analogicznie

konfigurujemy drugi wskaźnik. W podobny sposób możemy również skonfigurować łączniki.

Uaktualniamy panel i diagram, uzupełniając je o niezbędne informacje: nazwę przedmiotu, uczelnię, skład zespołu, numer grupy, datę, a także numer i tytuł ćwiczenia oraz programu. Informacje te podajemy w formie komentarzy umieszczonych na panelu i na diagramie. Porządkujemy rozmieszczenie elementów na panelu, uwzględniając ich wzajemne powiązania logiczne. Wykorzystujemy narzędzia **Align**, **Distribute**, **Resize**, **Reorder**, aby wyrównać obiekty, pogrupować je (**Group**) i zablokować ich położenie (**Lock**). Przykładowy, poprawnie uzupełniony panel przedstawiono na rycinie 7.21.

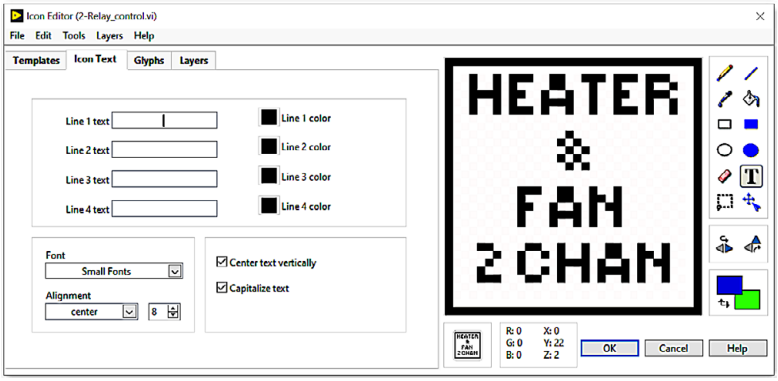



**Ryc. 7.21.** Panel programu do ręcznego sterowania grzałką i wentylatorem

Zapisujemy program na dysku we wskazanym przez nauczyciela katalogu, a jego nazwę notujemy w protokole. Wybierając opcję **File > Print > VI Documentation**, zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików z opisem zawartości notujemy w protokole. Zapisane pliki wykorzystujemy następnie do sporządzenia raportu.

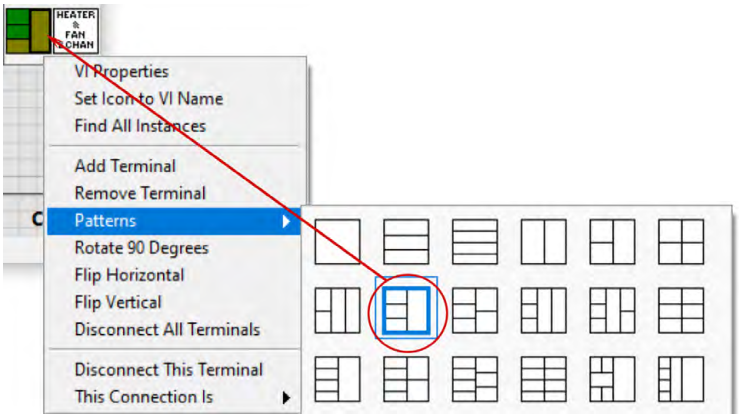
Następnym krokiem jest przekształcenie opracowanego programu w podprogram subVI. Zagadnienie to zostało szczegółowo opisane w poprzednim punkcie. Korzystamy z tego opisu i powtarzamy całą procedurę dla drugiego programu. W pierwszej kolejności zapisujemy program pod nową, krótką i komunikatywną nazwą. W nazwie pliku nie mogą występować spacje, polskie znaki ani inne znaki specjalne. Przykładowa nazwa programu sterującego przekaźnikami to **2-Relay_control.vi**, ale możemy zaproponować własną. Nazwę pliku z podprogramem notujemy w protokole. Kolejnym krokiem jest edycja

ikony podprogramu. Przykładową ikonę podprogramu sterującego grzałką i wentylatorem za pomocą dwóch przekaźników pokazano na rycinie 7.22.




**Ryc. 7.22.** Edytowanie ikony podprogramu do ręcznego sterowania grzałką i wentylatorem

Następnie wybieramy odpowiedni wzór terminali na panelu złącza podprogramu. Podprogram ten ma wejście i wyjście dla linii błędu oraz dwa wejścia sygnałów logicznych do sterowania grzałką i wentylatorem. Wybieramy zatem wzór z trzema terminalami po lewej stronie i jednym po prawej, tak jak pokazano na rycinie 7.23.

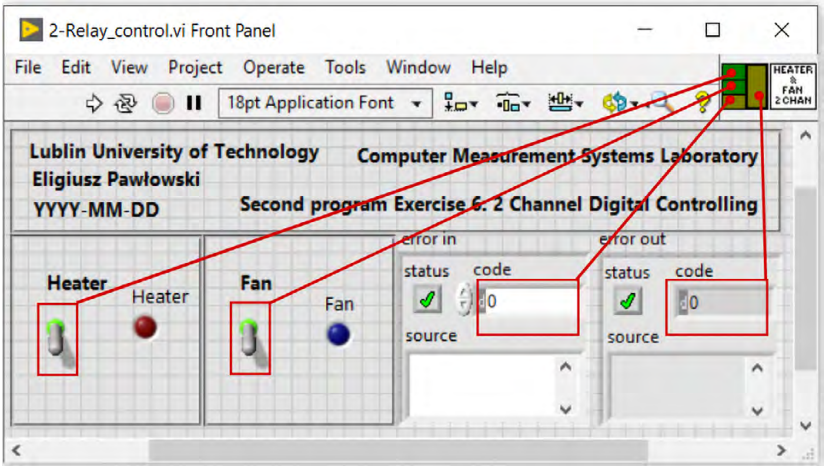



**Ryc. 7.23.** Wybór wzoru terminali dla panelu złącza

Mając wybrany wzór terminali dla panelu złącza, wykonujemy połączenia z panelem programu. Podczas łączenia kontrolek i wskaźników na panelu programu z terminalami na panelu złącza umieszczamy wejścia po lewej stronie, a wyjścia po prawej. Linie błędu łączymy na dolnych terminalach, a linie

danych na górnych. Poprawne skojarzenie terminali z kontrolkami i wskaźnikami na panelu programu pokazano na rycinie 7.24.

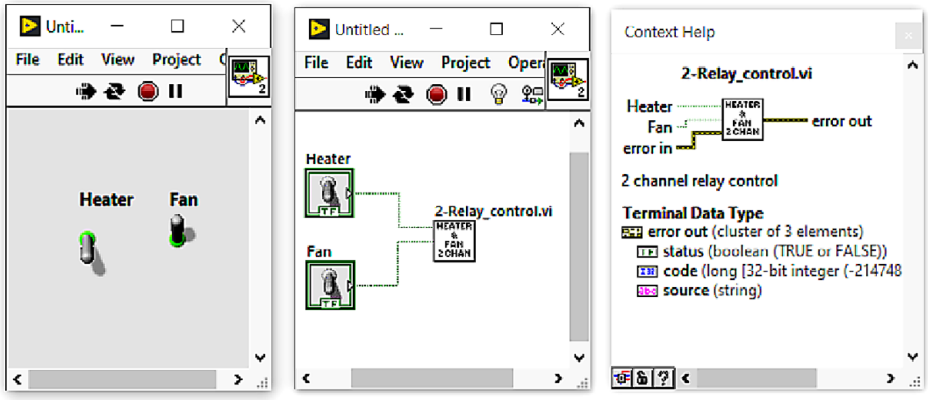
Ostatnim etapem jest dodanie opisu utworzonego podprogramu subVI, który będzie wyświetlał się w oknie **Context Help**. Klikając ikonę podprogramu prawym przyciskiem myszy, otwieramy dodatkowe menu, wybieramy opcję **VI Properties** i z rozwijanej listy przechodzimy do zakładki **Documentation**. W polu **VI description** wprowadzamy krótki opis działania naszego podprogramu (np. „2 channel relay control" lub inny, adekwatny do realizowanych przez niego funkcji). Zapisujemy gotowy podprogram na dysku. Jest on tym samym gotowy do wykorzystania w innej, nadrzędnej aplikacji.




**Ryc. 7.24.** Definiowanie połączeń terminali wejściowych i wyjściowych

Upewniamy się jeszcze tylko, czy stworzony podprogram subVI faktycznie działa prawidłowo. W tym celu realizujemy jak najprostszy program z użyciem stworzonego przed chwilą subVI i testujemy jego działanie. W menu głównym LabVIEW wybieramy opcję **File > NewVI**. Otwiera się nowe, puste okno do tworzenia programu. Przełączamy się na diagram (**Ctrl+E**). Klikając pusty diagram prawym przyciskiem myszy, wybieramy opcję **Functions > Select a VI...**. Wyszukujemy plik z utworzonym podprogramem po nazwie i umieszczamy go na diagramie. Klikając go prawym przyciskiem myszy, włączamy widoczność etykiety (**Visible Items > Label**). Otwieramy okienko **Context Help** i sprawdzamy, czy wyświetla się w nim odpowiednia informacja. Widzimy tam nazwę subVI, opisy terminali wejściowych i wyjściowych oraz opis jego

działania. Opisy terminali są zgodne z nazwami kontrolek na panelu podprogramu. Upewniamy się, że wejścia i wyjścia znajdują się we właściwych miejscach na ikonie. Klikając prawym przyciskiem myszy górne wejścia sygnałów sterujących, tworzymy zadajniki (**Create > Control**). Przechodzimy na panel programu, na którym znajdują się już dwa przełączniki dźwigienkowe. Program testowy jest gotowy! Uruchamiamy program do pracy ciągłej i sprawdzamy, czy grzałka i wentylator są sterowane zgodnie z opisami przełączników. W górnych pozycjach łączniki załączają obciążenie, a w dolnych je wyłączają. Rozmieszczamy na ekranie komputera obok siebie okno panelu programu, okno diagramu oraz okienko **Context Help**. Przykład ten przedstawiono na rycinie 7.25. Wykonany zrzut ekranu zapisujemy do pliku, aby wykorzystać go w raporcie. Wyłączamy grzałkę i wentylator. Zatrzymujemy program testowy przyciskiem Stop.




**Ryc. 7.25.** Podprogram subVI zastosowany w programie nadrzędnym

### 7.5.4. Trzeci program do automatycznej regulacji temperatury grzałki

Po utworzeniu dwóch podprogramów w poprzednich częściach ćwiczenia możemy ich teraz dowolnie używać przy tworzeniu nowych programów. Musimy jedynie znać ich nazwy i lokalizację na dysku. Aby zastosować podprogram w programie nadrzędnym, klikamy diagram prawym przyciskiem myszy, wybieramy opcję **Functions > Select a VI...**, po czym przechodzimy do odpowiedniego katalogu na dysku komputera i wybieramy właściwy plik.

Po umieszczeniu ikony subVI na diagramie stosujemy go tak samo jak inne funkcje standardowo wbudowane w środowisko LabVIEW.

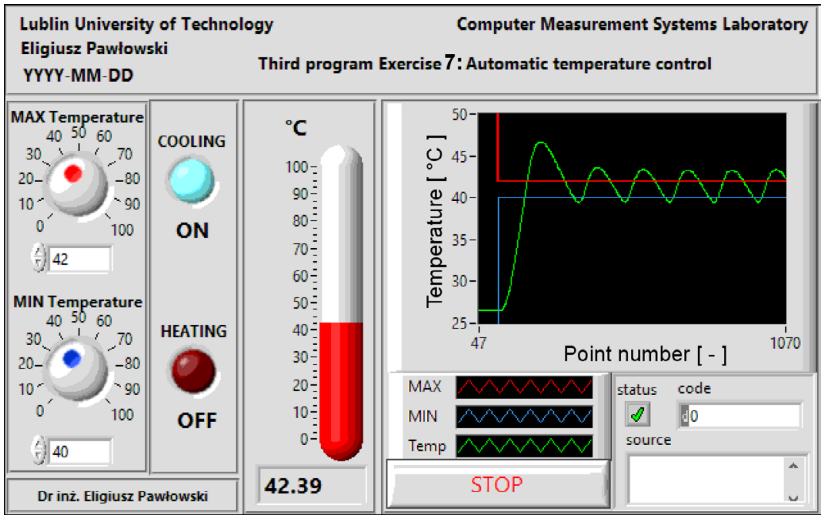
W każdej chwili możemy zmodyfikować podprogram z poziomu programu nadrzędnego, klikając dwukrotnie jego ikonę. Możemy to zrobić również w trakcie działania programu nadrzędnego. Pozwala nam to w wygodny sposób dobierać parametry pracy podprogramu (np. modyfikując liczbę uśrednianych pomiarów temperatury). Wszelkie zmiany wprowadzone w podprogramie dotyczą tylko jego bieżącej instancji, dopóki nie zapiszemy go po modyfikacjach do pliku. Po zapisaniu pliku wprowadzone zmiany wpływają na wszystkie wywołania tego podprogramu w używanym środowisku LabVIEW, a nie tylko na bieżącą instancję.

Po tym wprowadzeniu można przystąpić do tworzenia programu, który będzie automatycznie sterować grzałką i wentylatorem chłodzącym, wykorzystując opracowane wcześniej dwa podprogramy. Dzięki temu szybko stworzymy nowy program o przejrzystej strukturze. Poleceniem **File > NewVI** otwieramy nowy program i od razu zapisujemy go na dysku pod nazwą zawierającą numer 7-3-a. Nazwę programu notujemy w protokole.

Tworzymy program, którego przykładowy panel pokazano na rycinie 7.26, a postać diagramu – na rycinie 7.27. Najpierw umieszczamy na panelu programu dwa zadajniki numeryczne (**Controls > Numeric > Knob**) do zadawania minimalnej i maksymalnej temperatury. Wchodzimy w ich ustawienia (**Properties**) i ustalamy dla nich zakres skali (od 0 do 100), włączamy widoczność wskaźnika numerycznego (**Appearance > Show digital display(s)**) oraz ustawiamy kolor czerwony i niebieski odpowiednio dla maksymalnej i minimalnej temperatury. Następnie dodajemy na panelu dwa wskaźniki logiczne typu LED (**Controls > Boolean > Round LED**) i konfigurujemy je podobnie jak w podprogramie do sterowania przekaźnikami. Umieszczamy na panelu wskaźnik numeryczny typu termometr (**Controls > Numeric > Thermometer**) i konfigurujemy go podobnie jak zadajniki numeryczne. Ostatnim elementem, jaki umieszczamy na panelu, jest wykres (**Controls > Graph > Waveform Chart**). Przykładowe rozmieszczenie tych obiektów na panelu pokazane jest na rycinie 7.26. Przycisk STOP i status błędu dodamy później.

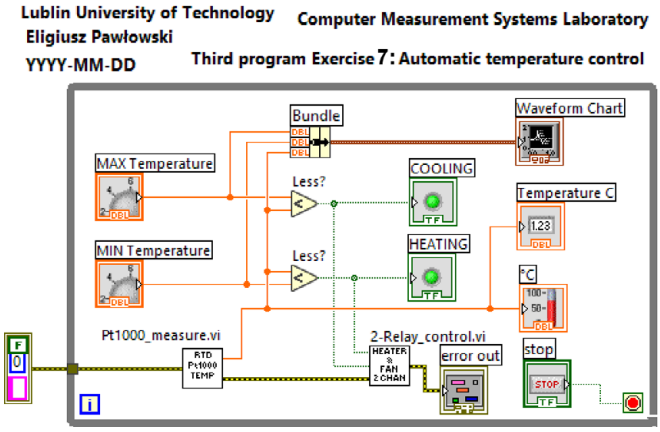
Przełączamy się na diagram i umieszczamy na nim dwa utworzone wcześniej subVI. Po lewej stronie ustawiamy subVI do pomiaru temperatury, a po

prawej – do sterowania przekaźnikami, ponieważ taki jest naturalny kierunek przepływu sygnałów na diagramie (od lewej do prawej). Najważniejszym zadaniem podczas pracy nad diagramem jest opracowanie algorytmu sterowania grzałką i wentylatorem na podstawie zmierzonej temperatury i nastawionych wartości minimalnej i maksymalnej. Zgodnie z założeniami grzałka włącza się, gdy jest za zimno, czyli gdy zmierzona temperatura jest niższa od ustawionej wartości minimalnej. Z kolei wentylator włącza się, gdy jest za gorąco, czyli gdy zmierzona temperatura przekracza ustawioną wartość maksymalną. Jest to więc bardzo prosty algorytm, a jednak, jak pokazuje praktyka, wiele osób popełnia błędy przy jego tworzeniu.




**Ryc. 7.26.** Przykładowy panel programu do automatycznej regulacji temperatury

Przedstawiony algorytm możemy zrealizować na wiele sposobów. Na rycinie 7.27 pokazano jedną z możliwości. Zastosowaliśmy tu dwie takie same funkcje porównujące (**Functions > Programming > Comparision > Less?**) z dwoma wejściami i jednym wyjściem. Funkcja **Less?** jest odpowiednikiem komparatora w układach elektronicznych. Dwie funkcje (**Less?**) tworzą odpowiednik tzw. komparatora okienkowego. Gdy temperatura znajduje się wewnątrz „okienka", oba przekaźniki pozostają wyłączone. Poniżej „okienka" włącza się grzałka, a powyżej niego – wentylator. Łączymy fragment algorytmu sterowania przekaźnikami i diodami LED według diagramu przedstawionego na rycinie 7.27.




**Ryc. 7.27.** Przykładowy diagram programu do automatycznej regulacji temperatury

Budujemy klaster danych do tworzenia wykresu. Używamy w tym celu funkcji **Functions > Programming > Cluster, Class & Variant > Bundle**. Klaster ten budujemy z nastaw temperatury minimalnej i maksymalnej oraz temperatury zmierzonej. Wchodzimy w opcje **Properties** wykresu i ustawiamy odpowiednie kolory dla trzech linii.

Klikając prawym przyciskiem myszy wejście **error in** podprogramu do pomiaru temperatury, tworzymy początkową wartość stałą (opcja **Create > Constant**). Na wyjściu **error out** subVI sterującego przekaźnikami tworzymy wskaźnik (**Create > Indicator**).

Wstawiamy na diagramie pętlę **While**. Tworzymy kontrolkę przycisku STOP, klikając prawym przyciskiem myszy terminal warunku zakończenia pętli (**loop condition**) i wybierając (**Create > Control**). Uzupełniamy brakujące połączenia na diagramie zgodnie z ryciną 7.27.

Na panelu ustawiamy temperaturę minimalną 0°C i maksymalną na 100°C. Ustalamy te wartości jako domyślne poleceniem **File > Edit > Make Current Values Default**. Gotowy program zapisujemy na dysku.

Uruchamiamy program przyciskiem  . Program mierzy temperaturę i rysuje wykres, jednak grzałka i wentylator pozostają wyłączone. Zwracamy również uwagę na wskaźniki wentylatora chłodzącego (niebieska dioda LED) i grzałki (czerwona dioda LED). Jeśli grzałka lub wentylator włączyły się, oznacza to błąd w algorytmie, który musimy odnaleźć i poprawić. W protokole notujemy wartość zmierzonej temperatury (jest to temperatura otoczenia). Odczytujemy z termometru rzeczywistą temperaturę w laboratorium i również

notujemy ją w protokole. Weryfikujemy w ten sposób, czy program poprawnie zmierzył temperaturę.

Przy pracującym programie ustawiamy temperaturę maksymalną o jeden stopień niższą od aktualnie zmierzonej temperatury otoczenia. Załącza się wówczas wentylator, jednak ponieważ dmucha on powietrzem w temperaturze otoczenia, nie przynosi to żadnego skutku. Ustawiamy ponownie temperaturę maksymalną na 100°C. Wentylator w rezultacie się wyłącza.

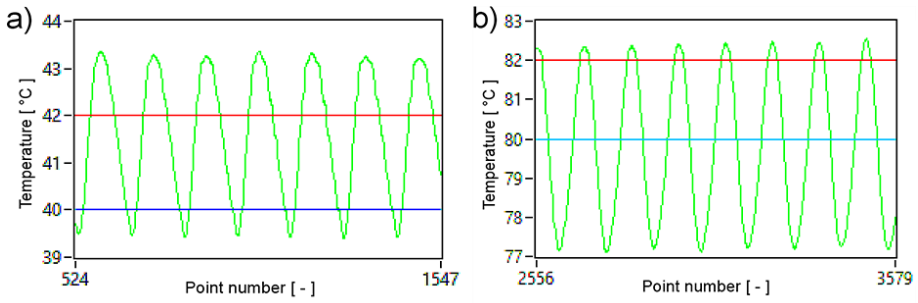
Ustawiamy temperaturę minimalną o jeden stopień wyższą od aktualnie zmierzonej temperatury otoczenia. Załącza się wówczas grzałka. Temperatura zaczyna narastać. Po osiągnięciu ustawionej wartości temperatury minimalnej grzałka się wyłącza. Temperatura nadal jednak wzrasta, ponieważ rezystancyjny element grzejny osiągnął wyższą temperaturę niż ta zmierzona przez czujnik Pt1000. Wynika to z występowania pewnej pojemności cieplnej grzałki i rezystancji termicznej pomiędzy grzałką a czujnikiem. Po wyłączeniu zasilania grzałka nadal oddaje ciepło do czujnika. Po pewnym czasie następuje stan równowagi cieplnej i temperatura zaczyna się obniżać. Ponieważ jednak różnica temperatur pomiędzy grzałką a otoczeniem jest niewielka, proces stygnięcia zachodzi bardzo powoli. Ustawiamy temperaturę maksymalną na 0°C. Włącza się wówczas wentylator, temperatura szybko spada, a po chwili ponownie załącza się grzałka. Jeśli program działa poprawnie, ponownie ustawiamy temperaturę minimalną na 0°C i maksymalną na 100°C. Grzałka i wentylator wyłączają się, po czym zatrzymujemy program przyciskiem **STOP** na panelu programu. Pamiętamy przy tym, aby nie używać przycisku Stop  na pasku narzędziowym środowiska LabVIEW! Wnioski z testowania programu notujemy w protokole.

Uzupełniamy panel i diagram oraz diagram o niezbędne informacje: skład zespołu, numer grupy, datę, numer i tytuł ćwiczenia oraz numer programu. Informacje te podajemy w formie komentarzy umieszczonych na panelu sterowania oraz na diagramie. Na panelu wykorzystujemy obiekty graficzne (dekoracji), aby uporządkować jego zawartość i nadać jej przejrzystą strukturę. Aktualizujemy opisy wszystkich obiektów zgodnie z ich funkcją w programie. Wykorzystujemy narzędzia **Align**  , **Distribute**  , **Resize**  , **Reorder**  , aby precyzyjnie wyrównać położenie obiekty na panelu, pogrupować je (**Group**) i zablokować ich położenie (**Lock**). Przykładowy wygląd

panelu programu przedstawiamy na rycinie 7.27. Gotowy program zapisujemy na dysku.

Teraz sprawdzamy skuteczność działania układu regulacji temperatury. Na panelu ustawiamy temperaturę minimalną na 0°C i maksymalną na 100°C. Uruchamiamy program odpowiednim przyciskiem  . Program mierzy temperaturę i rysuje wykres, jednak grzałka i wentylator pozostają wyłączone. Przy działającym programie ustawiamy temperaturę minimalną na 40°C i maksymalną na 42°C. Grzałka załącza się aż do osiągnięcia temperatury 40°C, a następnie się wyłącza. Ze względu na bezwładność cieplną grzałki temperatura nadal rośnie i przy 42°C włącza się wentylator. Dopiero po pewnym czasie temperatura zaczyna maleć, a gdy obniży się ponownie do 42°C, wentylator się wyłącza. Następnie cykl pracy się powtarza, co możemy zaobserwować na rycinie 7.26. Wykres temperatury zapisujemy do pliku i wykorzystujemy do przygotowania raportu. Przykładowy wykres dla nastaw 40–42°C pokazano na rycinie 7.28a. Zwracamy uwagę, że czasy pracy grzałki i wentylatora różnią się od siebie. Również przeregulowanie w górę i w dół jest inne. Analizujemy przy tym, który proces jest skuteczniejszy: grzanie czy chłodzenie. Sformułowany wniosek notujemy w protokole.

Zmieniamy nastawy temperatury minimalnej na 80°C i maksymalnej na 82°C. Czekamy, aż działanie układu regulacji ustabilizuje się dla nowych nastaw. Uzyskany wykres temperatury zapisujemy do pliku. Przykładowy wykres dla nastaw 80–82°C pokazano na rycinie 7.28b. Analizujemy, czy skuteczność regulacji temperatury jest taka sama jak dla poprzednich nastaw, starając się wyjaśnić przyczynę odmiennego działania układu. Podobne eksperymenty możemy przeprowadzić również dla innych nastaw. Sformułowane wnioski notujemy w protokole.




**Ryc. 7.28.** Wykres regulacji temperatury dla nastaw: a) 40–42°C, b) 80–82°C

Podczas działania programu przełączamy się na diagram. Klikamy dwukrotnie ikonę podprogramu do pomiaru temperatury. Otwiera się wówczas panel czołowy tego podprogramu, na którym widoczne są wyniki pomiaru temperatury. Następnie klikamy dwukrotnie ikonę podprogramu do sterowania przekaźnikami. Porządkujemy obszar roboczy tak, aby jednocześnie widoczne były trzy panele: programu nadrzędnego i dwóch podprogramów. Zgodnie z wcześniejszymi eksperymentami podprogram do pomiaru temperatury pracuje z dużą liczbą uśrednianych próbek (pomiędzy 100 a 1000). Podczas działania programu zmieniamy liczbę próbek na 5. Wyniki pomiarów temperatury stają się w efekcie bardzo zaszumione. Zwracamy przy tym uwagę na proces przełączania przekaźników. Staje się on bardzo niestabilny. Eksperymentalnie ustalamy liczbę próbek zapewniającą stabilne przełączanie obu przekaźników. Sformułowany wniosek notujemy w protokole. Zauważamy, że zwiększanie liczby uśrednianych pomiarów wprawdzie korzystnie zmniejsza poziom zakłóceń, ale jednocześnie niekorzystnie zwiększa czas pomiaru i spowalnia układ regulacji. Niezbędne jest zatem znalezienie rozwiązania optymalnego. Zagadnienie to jest powszechnie znane w automatyce. Prostym i skutecznym rozwiązaniem jest tu wprowadzenie histerezy. Kroku tego dokonamy w następnym punkcie.

Jeśli sprawdzenie działania programu wypadło pomyślnie, zapisujemy program na dysku i notujemy jego nazwę w protokole. Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików z opisem zawartości notujemy w protokole. Zapisane pliki wykorzystujemy do sporządzenia sprawozdania.

Tworzymy kopię programu do realizacji następnego zadania, zmieniając jego numer na kolejny (tzn. 7-4-a). Nie modyfikujemy przy tym programu, który stanowi ostateczną wersję poprzednio wykonanego zadania. Zawsze pamiętamy, aby po zapisaniu ostatecznej wersji programu (realizującej w pełni dany punkt ćwiczenia) utworzyć jego kopię pod nową nazwę, zmieniając ostatnią cyfrę na kolejną wartość.

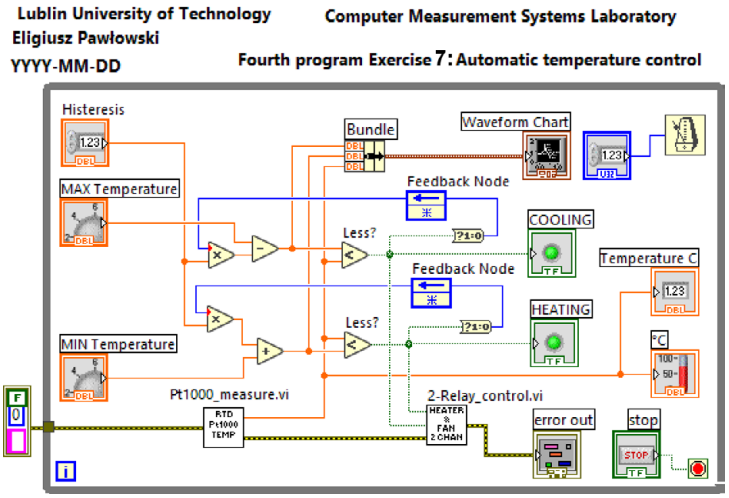
### 7.5.5. Czwarty program z histerezą w układzie regulacji

Wprowadzenie histerezy do układu regulacji to skuteczny sposób na zapewnienie stabilnego przełączania przekaźników przy dużym poziomie zakłóceń. Zjawisko histerezy powszechnie wykorzystujemy również w układach elektronicznych opartych na komparatorach. Wprowadzając w komparatorze dodatnie sprzężenie zwrotne, uzyskujemy odporność na wysoki poziom szumów w sygnale. Histereza polega na zmianie progu przełączania komparatora w zależności od kierunku zmian sygnału. Inny (wyższy) jest próg przełączania, gdy sygnał narasta, a inny (niższy), gdy sygnał maleje. Taki sam efekt możemy uzyskać metodą programową. Rolę komparatora w programie pełni funkcja **Less?** Aby uzyskać histerezę metodą programową, przesuwamy próg przełączania na wejściu funkcji **Less?** w zależności od stanu jej wyjścia. W odróżnieniu jednak od elektronicznego układu komparatora, na diagramie w LabVIEW nie możemy połączyć wyjścia funkcji bezpośrednio z jej wejściem. Ograniczenie to wynika z podstawowej zasady przepływu danych: każda funkcja w **środowisku** LabVIEW oblicza wartość na wyjściu tylko wtedy, gdy ma dostępne dane na wszystkich wejściach. Nie jesteśmy zatem w stanie tak „zapętlić" algorytmu, aby te same dane znajdowały się jednocześnie na wejściu i wyjściu funkcji. Poznamy jednak sposób ominięcia tego ograniczenia.

Otwieramy program zapisany w poprzednim punkcie pod numerem 7-4-a. Modyfikujemy go według przykładu przedstawionego na rycinie 7.29. Pracę zaczynamy od modyfikacji w części sterującej wentylatorem. Do wyjścia górnej funkcji **Less?** dołączamy funkcję konwersji typu logicznego na numeryczny (**Functions > Programming > Boolean > Bool to (0, 1)**). Co prawda zmienne logiczne przyjmują wartości 0 i 1, jednak domyślnie możemy na nich wykonywać jedynie operacje logiczne, natomiast w naszym algorytmie musimy przeprowadzić operacje arytmetyczne: mnożenie, dodawanie oraz odejmowanie. Dlatego też taka zamiana typu z logicznego na numeryczny jest w tym przypadku niezbędna.

Kasujemy połączenie z wyjścia zadajnika temperatury maksymalnej. W to miejsce wstawiamy funkcję mnożenia (**Multiply**) i odejmowania (**Subtract**). Zadajnik temperatury maksymalnej dołączamy do górnego wejścia funkcji **Subtract**. Jej dolne wejście łączymy z wyjściem funkcji **Multiply**. Z kolei wyjście funkcji **Bool to (0, 1)** łączymy z górnym wejściem funkcji **Multiply**. Środowisko LabVIEW automatycznie wstawi w to połączenie węzeł **Feedback**

**Node**. Węzeł ten jest niezbędny, ponieważ w programie nie możemy tworzyć zapętleń sygnałów, tzn. nie możemy sygnału z wyjścia funkcji podłączyć do wejścia tej samej funkcji – ani bezpośrednio, ani pośrednio (poprzez inne funkcje). Jak już wyjaśniono wcześniej, taki algorytm nie mógłby działać, ponieważ wartości na wejściach funkcji muszą być znane w chwili obliczania wartości na jej wyjściu. Jeśli jednak program działa cyklicznie w pętli, możemy na wejście funkcji podać wartość z jej wyjścia, pochodzącą z poprzedniej iteracji. Takie zadanie realizuje właśnie węzeł **Feedback Node**. Alternatywnym rozwiązaniem tego problemu jest zastosowanie rejestrów przesuwnych.




**Ryc. 7.29.** Diagram programu do automatycznej regulacji temperatury z wprowadzoną histerezą przełączania przekaźników

Przełączamy się na panel, umieszczamy tam zadajnik numeryczny (**Numeric Control**), nadajmy mu nazwę **Histereza** i wpisujemy wartość początkową 0,5. Wchodzimy we właściwości (**Properties**) i w zakładce **Data Entry** ustawiamy wartości: **Minimum** na 0, **Maximum** na 10 oraz **Increment** na 0,1. W okienkach obok wybieramy opcje **Coerce**. Na diagramie dołączamy wyjście kontrolki **Histereza** do dolnego wejścia funkcji **Multiply**. Podobnie modyfikujemy program w części sterującej grzałką. Zamiast funkcji odejmowania (**Subtract**) stosujemy tu jednak funkcję dodawania (**Add**). Wynika to z faktu, że wentylator załącza się powyżej temperatury maksymalnej, a grzałka – poniżej temperatury minimalnej. Dlatego też w tych dwóch układach histereza musi działać w przeciwnych kierunkach.

Aby móc wpływać na szybkość działania programu, umieszczamy jeszcze na diagramie funkcję **Timing > Wait Until Next ms Multiple** i na jej wejściu tworzymy zadajnik do ustawiania czasu opóźnienia w milisekundach (klikając prawym przyciskiem myszy i wybierając opcję **Create > Control**). Ustawiamy początkową wartość tego opóźnienia na 100 ms. Na panelu ustawiamy temperaturę minimalną na 0°C i maksymalną na 100°C. Ustalamy wszystkie wprowadzone wartości jako domyślne poleceniem **File > Edit > Make Current Values Default**. Zapisujemy program na dysku. Program jest tym samym gotowy do testowania.

Uruchamiamy program przyciskiem  . Program mierzy temperaturę i rysuje wykres, jednak grzałka i wentylator pozostają wyłączone. Zmieniamy opóźnienie czasowe na 0 ms, dzięki czemu program pracuje z maksymalną szybkością. Przełączamy się na diagram i klikamy dwukrotnie ikonę podprogramu do pomiaru temperatury. Zmieniamy liczbę uśrednianych próbek na 5. Wyniki pomiarów temperatury stają się w efekcie bardzo zaszumione.

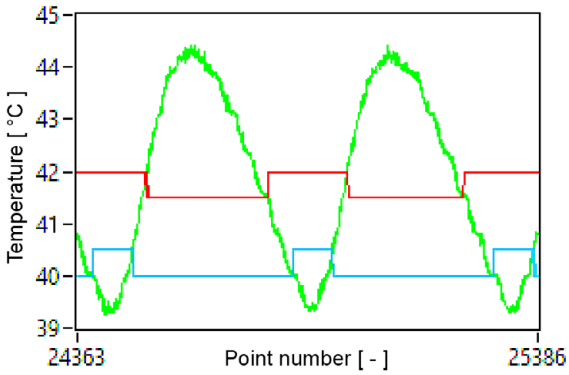
Na panelu programu ustawiamy temperaturę minimalną na 40°C i maksymalną na 42°C. Grzałka załącza się aż do osiągnięcia temperatury 40°C, a następnie się wyłącza. Czekamy odpowiedni czas, aż praca programu się ustabilizuje, a grzałka i wentylator zaczną się cyklicznie włączać i wyłączać. Jeśli nie popełniliśmy błędu w algorytmie, wykres wygląda podobnie jak na rycinie 7.30. W chwili załączania grzałki i wentylatora odpowiednia temperatura przełączania skokowo zmienia się o 0,5°C. Jeśli poziom szumów nie przekracza tej wartości, praca przekaźników jest stabilna.

**Zwracamy szczególną uwagę, aby podczas tych eksperymentów nie dopuszczać do długotrwałej pracy układu przy niestabilnym przełączaniu przekaźników. Zjawisko to wpływa bowiem bardzo niekorzystnie na trwałość styków przekaźników.**

Zmniejszamy stopniowo histerezę, aż praca przekaźników stanie się niestabilna. Wartość tę notujemy w protokole. Zwiększamy następnie opóźnienie czasowe na 200 ms. Obserwujemy, czy praca przekaźników uległa poprawie. Ustawiamy histerezę na zero. Sprawdzamy, jak pracuje w tej sytuacji układ. Analizujemy przy tym, jakie opóźnienie zapewnia stabilne przełączanie przekaźników. Sformułowane wnioski notujemy w protokole. Dobieramy nastawy histerezy i opóźnienia czasowego, aby uzyskać wykres podobny do tego na

rycinie 7.30. Możemy przy tym użyć innych nastaw temperatury. Otrzymany wykres zapisujemy do pliku i wykorzystujemy w sprawozdaniu.

Na pracę programu wpływają ostatecznie cztery parametry: poziom szumów w układzie pomiaru temperatury, liczba uśrednianych pomiarów, histereza i opóźnienie w pętli programu. Obiektywny poziom szumów w układzie jest od nas niezależny. Dlatego też to właśnie do występującego poziomu szumów dobieramy pozostałe parametry pracy programu. Możemy to zrobić na kilka różnych sposobów. Zwiększanie liczby uśrednianych próbek zmniejsza wprawdzie poziom zakłóceń, jednak jednocześnie spowalnia działanie programu. Z kolei histereza zapewnia stabilną pracę układu regulacji, lecz z drugiej strony zwiększa błędy regulacji temperatury. Wprowadzenie opóźnienia również umożliwia stabilną pracę układu, ale – podobnie jak uśrednianie – spowalnia program, co tym samym zwiększa ostateczny błąd regulacji. Podczas eksperymentów weryfikujemy, czy udało nam się zaobserwować wszystkie te zjawiska. Analizujemy również, jakie wartości tych parametrów okazują się w praktyce optymalne. Ustawiamy wybrane wartości parametrów, a otrzymany wykres zmian temperatury zapisujemy do pliku graficznego. Sformułowane wnioski notujemy w protokole.




**Ryc. 7.30.** Wykres regulacji temperatury dla nastaw 40–42°C z histerezą 0,5°C

Jeśli sprawdzenie działania programu wypadło pomyślnie, ustawiamy temperaturę minimalną na 0°C i maksymalną na 100°C. Upewniamy się przy tym, czy grzałka i wentylator zostały wyłączone. Następnie zapisujemy program na dysku i notujemy jego nazwę w protokole. Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu,

zawierającą panel oraz diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików wraz z opisem ich zawartości notujemy w protokole. Zapisane pliki wykorzystujemy ostatecznie do sporządzenia sprawozdania.

## 7.6. Opracowanie raportu z realizacji ćwiczenia

Kompletny raport z przeprowadzonego ćwiczenia zawiera następujące elementy:

- stronę tytułową z danymi osobowymi zespołu, datą oraz tytułem ćwiczenia,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy zrealizowanych zadań przewidzianych w programie ćwiczenia,
- opis każdego z wykonanych zadań (bądź podanie uzasadnionej przyczyny jego niewykonania),
- pełną dokumentację zrealizowanych i uruchomionych programów,
- wykresy wygenerowane przez nasze programy w środowisku LabVIEW,
- opracowane wyniki obserwacji i wnioski zanotowane wcześniej w protokole,
- ewentualne uwagi dotyczące błędów zauważonych w instrukcji lub przykładowych programach.
- Ocenie podlegają ponadto następujące aspekty:
- poprawność działania programów,
- czytelność i przejrzystość diagramów,
- estetyka panelu (odpowiedni dobór kolorów i elementów graficznych),
- funkcjonalność panelu pod kątem wygody obsługi,
- całkowity zakres zrealizowanych zadań,
- trafność sformułowanych wniosków końcowych.

# 8. Pomiary rezystancji termicznej czujnika temperatury

## 8.1. Cel ćwiczenia i zadania do wykonania

Ćwiczenie 8 stanowi kontynuację ćwiczenia 6, wykorzystując przy tym wybrane elementy z ćwiczeń 5 oraz 7. Podstawę do rozpoczęcia ćwiczenia 8 stanowi trzeci program opracowany w ramach ćwiczenia 6, służący do wyznaczania charakterystyki prądowo-napięciowej dwójników wraz z zapisem wyników do pliku. Program ten uzupełniamy o algorytm obliczania temperatury na podstawie wyników pomiarów rezystancji czujnika Pt1000 (zadanie to realizowaliśmy w ćwiczeniu 7). Wykorzystujemy w nim również aproksymację liniową charakterystyki metodą najmniejszych kwadratów, którą stosowaliśmy wcześniej w ćwiczeniu 5.

Celem ćwiczenia 8 jest zapoznanie się z problematyką programowania systemu pomiarowego w środowisku LabVIEW z jednoczesnym wykorzystaniem przetworników cyfrowo-analogowego i analogowo-cyfrowego w dwóch kanałach na potrzeby zbadania efektu samonagrzewania w czujniku termorezystancyjnym typu Pt1000.

Program laboratorium obejmuje następujące zagadnienia:

- konfigurowanie modułu NI USB-6001 do jednoczesnego sterowania przetwornikiem DAC i pomiaru napięcia przetwornikiem ADC w dwóch kanałach,
- zbadanie efektu samonagrzewania w czujniku termorezystancyjnym,

- wyznaczenie rezystancji termicznej dla czujnika termorezystancyjnego,
- przygotowanie programu LabVIEW do automatycznego zbadania wewnętrznego i zewnętrznego efektu transportu ciepła w czujniku termorezystancyjnym.

W ramach ćwiczenia 8 opracujemy trzy programy:

1. Pierwszy program do badania efektu samonagrzewania w czujniku temperatury,
2. Drugi program do wyznaczania rezystancji termicznej czujnika temperatury,
3. Trzeci program do automatycznego badania efektu samonagrzewania.

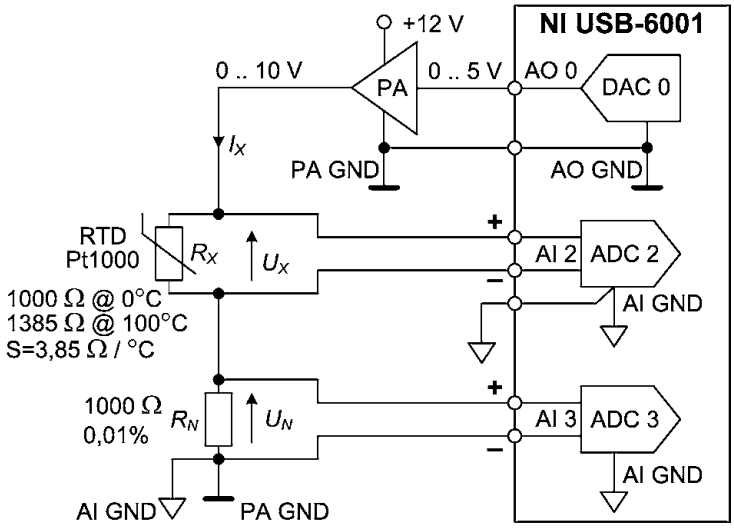
## 8.2. Opis stanowiska laboratoryjnego

### 8.2.1. Schemat blokowy stanowiska laboratoryjnego

W ćwiczeniu 8 wykorzystujemy to samo stanowisko laboratoryjne, które stosowaliśmy w ćwiczeniu 6, dlatego nie będziemy go ponownie szczegółowo opisywać. Zostaną przypomniane jedynie najważniejsze zagadnienia i omówione kluczowe różnice. Schemat blokowy stanowiska laboratoryjnego wykorzystywanego w ćwiczeniu 8 przedstawiono na rycinie 8.1 [42]. Moduł akwizycji danych pomiarowych NI USB-6001 jest dołączony do komputera PC za pomocą interfejsu USB. Napięcie wyjściowe z przetwornika cyfrowo-analogowego AO 0 podajemy na wejście wzmacniacza mocy PA o wzmocnieniu równym 2 V/V. Stosowaliśmy go już w ćwiczeniu 4, a jego parametry szczegółowo opisaliśmy i zbadaliśmy w ćwiczeniu 5. Napięcie wyjściowe ze wzmacniacza zasila układ pomiarowy, składający się z szeregowego połączenia badanego czujnika temperatury Pt1000 i rezystora wzorcowego $R_N = 1000\ \Omega$. Badamy czujnik platynowy tego samego typu co zastosowanyw ćwiczeniu 7.

Analizując układ pomiarowy, zwracamy uwagę, że ćwiczenie 8 stanowi połączenie dwóch wcześniejszych: ćwiczenia 6 (dotyczącego badania dwójników) i ćwiczenia 7 (poświęconego pomiarom temperatury). Dlatego też w ćwiczeniu 8 badany czujnik Pt1000 występuje jednocześnie w dwóch rolach. Z jednej strony rozpatrujemy go jako dwójnik o nieliniowej charakterystyce prądowo-napięciowej (podobnie jak w ćwiczeniu 6), a z drugiej strony – jako czujnik temperatury o liniowej charakterystyce rezystancji w funkcji temperatury (jak w ćwiczeniu 7). Stanowi on zatem jednocześnie element liniowy

oraz nieliniowy, w zależności od tego, z jakiego punktu widzenia rozpatrujemy jego właściwości.




**Ryc. 8.1.** Schemat stanowiska laboratoryjnego w ćwiczeniu 8 [42]

Zgodnie z normą [43] rezystancja badanego czujnika Pt1000 w temperaturze 0°C wynosi $R_{Pt} = 1000\ \Omega$, a w temperaturze 100°C wzrasta do $R_{Pt} = 1385\ \Omega$. W przedziale temperatur 0–100°C czułość czujnika platynowego Pt1000 wynosi zatem 3,85 Ω/°C. Ze względu na nieliniowość charakterystyki przetwarzania czujnika platynowego, dla dokładniejszych pomiarów w szerokim zakresie temperatur stosujemy charakterystykę podaną w normie [43], która dla temperatur mniejszych od 0°C przyjmuje postać wielomianu trzeciego stopnia, a dla temperatur większych od 0°C – wielomianu drugiego stopnia. Ponieważ jednak w ćwiczeniu 8 zmiany temperatury czujnika są stosunkowo niewielkie, traktujemy go jako element liniowy z punktu widzenia charakterystyki przetwarzania temperatury na rezystancję. Jednocześnie jednak prąd płynący przez czujnik powoduje wydzielanie się w nim ciepła, co podnosi jego temperaturę, a w konsekwencji – zwiększa również jego rezystancję proporcjonalnie do temperatury, którą osiągnął on w efekcie samonagrzewania. Dlatego charakterystyka prądowo-napięciowa badanego czujnika jest nieliniowa. Należy również dodatkowo wyjaśnić, że w ćwiczeniu 8 czujnik nie mierzy temperatury otoczenia (jak miało to miejsce w ćwiczeniu 7), lecz własną temperaturę, która ulega zmianie w wyniku efektu samonagrzewania. Z kolei temperatura

otoczenia (którym jest nieruchome powietrze w pomieszczeniu laboratorium) nie ulega zmianie.

Do wyznaczenia charakterystyki prądowo-napięciowej czujnika wykorzystujemy dwa wejścia analogowe: AI 2 oraz AI 3, pracujące w trybie różnicowym. Napięcie $U_X$ na badanym czujniku Pt1000 mierzymy w kanale AI 2, a napięcie $U_N$ na rezystorze wzorcowym – w kanale AI 3. Zadając na przetworniku DAC napięcia w zakresie od 0 do 5 V, na wyjściu wzmacniacza otrzymujemy napięcia rzędu od 0 do 10 V. Stosujemy rezystor wzorcowy o wartości $R_N = 1000\ \Omega$. Łączna rezystancja w układzie pomiarowym ($R_{Pt} + R_N$) wynosi zatem około 2000 Ω i ulega zmianom ze względu na samonagrzewanie czujnika platynowego. Ponieważ maksymalne napięcie zasilające układ pomiarowy wynosi około 10 V, napięcie odkładające się na czujniku osiąga maksymalnie około 5 V. Pomiary przeprowadzamy w zakresie prądów do około 5 mA. Powoduje to wydzielenie się w czujniku mocy nieprzekraczającej 25 mW, co jednak go podgrzewa (nawet o kilkanaście stopni Celsjusza). Zjawisko to nazywamy efektem samonagrzewania czujnika (**self-heating effect**); skutkuje ono błędami pomiaru temperatury w przypadku zastosowania zbyt dużej wartości prądu pomiarowego. Zagadnienie to będzie dokładniej opisane w dalszej części instrukcji.

Platynowy czujnik termorezystancyjny Pt1000 należy do grupy tzw. czujników parametrycznych, co oznacza, że wielkość mierzona (temperatura) zmienia pewien parametr czujnika – w tym przypadku rezystancję. Do pomiaru rezystancji czujnika stosujemy metodę techniczną, zwaną również metodą amperomierza i woltomierza. Metoda ta występuje w dwóch odmianach: z poprawnie mierzonym prądem oraz z poprawnie mierzonym napięciem. Rolę woltomierza pełni wejście analogowe AI 2, natomiast funkcję amperomierza – rezystor wzorcowy $R_N$ z podłączonym wejściem analogowym AI 3. Zagadnienie to było szczegółowo objaśnione w ćwiczeniu 6 (na rycinie 6.1). Analizując schemat na rycinie 8.1, zauważamy, że zastosowano tu wersję układu z poprawnie mierzonym napięciem. Prąd *IX* płynący w układzie możemy obliczyć na podstawie następującej zależności:

$$I_X = \frac{U_N}{R_N}. \tag{8.1}$$

Znając wartość prądu $I_X$ płynącego przez badany czujnik oraz napięcie $U_X$ zmierzone na nim w kanale AI 2, możemy obliczyć rezystancję czujnika $R_{Pt}$ na podstawie zależności:

$$R_{Pt} = \frac{U_X}{I_X} = \frac{U_X}{U_N} R_N, \tag{8.2}$$

oraz moc $P_X$ traconą w czujniku ze wzoru:

$$P_X = U_X I_X = U_X \frac{U_N}{R_N}. \tag{8.3}$$

Moc $P_X$ tracona w czujniku odgrywa kluczową rolę w tym ćwiczeniu, ponieważ to właśnie ona wywołuje efekt samonagrzewania. Aby zbadać ten efekt, wyznaczamy zależność temperatury czujnika od nagrzewającej go mocy elektrycznej (określonej wzorem 8.3), która zamienia się w ciepło. Z powodów wyjaśnionych na wstępie w ćwiczeniu 8 zakładamy liniową charakterystykę przetwarzania czujnika Pt1000, opisaną następującą zależnością:

$$R_{Pt} = R_0 \left(1 + \alpha T\right), \tag{8.4}$$

gdzie: $R_{Pt}$ to rezystancja czujnika w temperaturze $T$, $R_0 = 1000\ \Omega$ to rezystancja czujnika w temperaturze początkowej 0°C, a $\alpha$ jest współczynnikiem temperaturowym rezystancji, równym 0,00385°C$^{-1}$. Uwzględniając podane wartości parametrów liniowej charakterystyki czujnika Pt1000, ostatecznie możemy obliczyć jego temperaturę $T$ (w stopniach Celsjusza) według wzoru:

$$T = \frac{R_{Pt} - R_0}{\alpha R_0} = \frac{R_{Pt} - 1000}{3{,}85}. \tag{8.5}$$

Tak jak już wyjaśniono wcześniej, dokładniejszy pomiar temperatury uzyskujemy, przyjmując charakterystykę przetwarzania czujnika w postaci wielomianu drugiego stopnia [43]:

$$R_{Pt} = R_0 \left(1 + AT + BT^2\right), \tag{8.6}$$

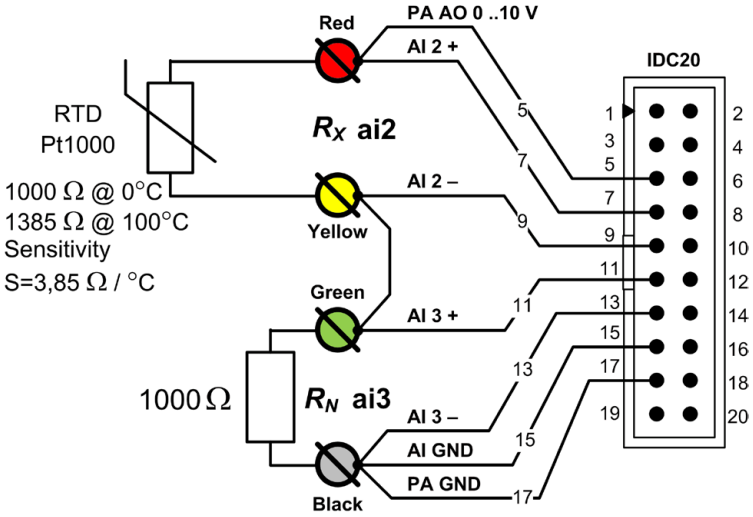
gdzie współczynniki wielomianu wynoszą odpowiednio: $A = 3{,}9083 \times 10^{-3}$°C$^{-1}$ oraz $B = -5{,}775 \times 10^{-7}$°C$^{-2}$ [43]. Charakterystykę opisaną zależnością (8.6) stosujemy dla temperatur powyżej 0°C. Z kolei dla temperatur ujemnych

wykorzystujemy wielomian trzeciego stopnia [43]. Po przekształceniu równania (8.6) [44] temperaturę $T$ czujnika możemy obliczyć z następującego wzoru:

$$T = \frac{-A + \sqrt{A^2 - 4B(1 - R_{Pt}/R_0)}}{2B}. \tag{8.7}$$

### 8.1.2. Dołączanie badanego czujnika do układu pomiarowego

Schemat modułu z zaciskami laboratoryjnymi, wykorzystywanego do badania efektu samonagrzewania czujnika termorezystancyjnego, przedstawiono na rycinie 8.2.

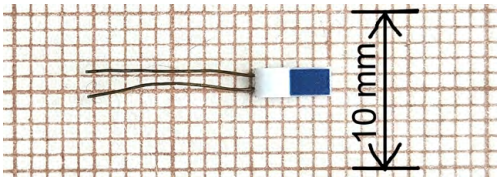



**Ryc. 8.2.** Schemat modułu do badania czujnika termorezystancyjnego

Moduł ten był już zastosowany w ćwiczeniu 6, jednak dołączano do niego wówczas różnego rodzaju dwójniki liniowe i nieliniowe oraz rezystory wzorcowe o rożnych wartościach. Z kolei w ćwiczeniu 8 będzie dołączony do niego cienkowarstwowy czujnik platynowy Pt1000, przedstawiony na rycinie 8.3. Zdjęcie to wykonano na tle papieru milimetrowego, co pozwala łatwo ocenić rzeczywiste wymiary czujnika.

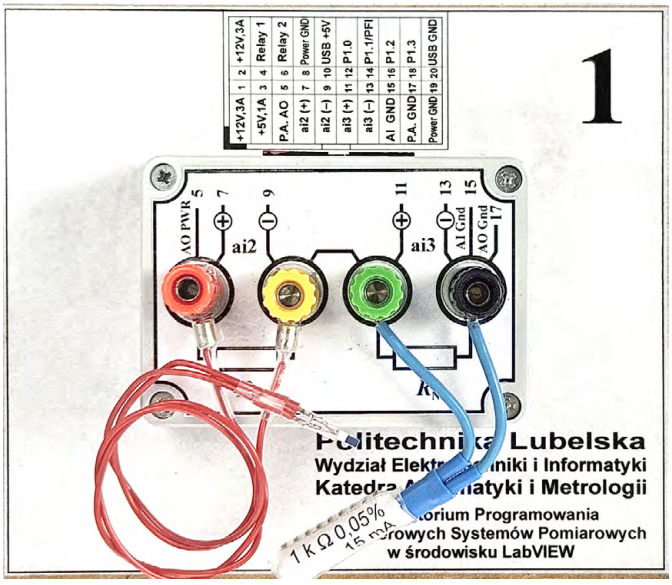
Czujnik ma postać prostokątnej, ceramicznej płytki o wymiarach 2mm × 5 mm. Rezystancyjny element czynny czujnika stanowi cienka warstwa platyny (o grubości poniżej 1 μm), napylona na ceramicznym podłożu

i ukształtowana w formie meandra. Od wierzchu warstwę platyny zabezpiecza szklana powłoka. Czujnik spełnia przy tym wymagania norm [43, 45].




**Ryc. 8.3.** Cienkowarstwowy czujnik platynowy Pt1000 badany w ćwiczeniu 8

Na potrzeby ćwiczenia do wyprowadzeń czujnika zostały przylutowane dłuższe przewody, a miejsca połączeń zabezpieczono rurkami termokurczliwymi. Badany czujnik podłączamy do zacisków czerwonego i żółtego. Czujnik umieszczamy na stanowisku w taki sposób, aby znajdował się w wolnej przestrzeni (w nieruchomym powietrzu). Zwracamy przy tym uwagę, aby czujnik nie dotykał żadnych innych przedmiotów. Biegunowość podłączenia czujnika nie ma w tym przypadku znaczenia. Z kolei do zacisków zielonego i czarnego podłączamy rezystor wzorcowy o rezystancji $R_N = 1000\ \Omega$. Moduł z zaciskami laboratoryjnymi oraz podłączonym czujnikiem i rezystorem wzorcowym przedstawiono na rycinie 8.4.




**Ryc. 8.4** Moduł z dołączonym badanym czujnikiem Pt1000 i rezystorem wzorcowym 1000 Ω

## 8.3. Efekt samonagrzewania czujników termorezystancyjnych

### 8.3.1. Problem efektu samonagrzewania

Każdy podzespół, na którym występuje spadek napięcia podczas przepływu prądu, zwiększa swoją temperaturę. Zjawisko to wiąże się z wydzielaniem ciepła, co wynika z mocy elektrycznej traconej w tym elemencie. Część energii elektrycznej dostarczanej ze źródła zasilania zamienia się w inną postać energii – np. w mechaniczną w silniku elektrycznym lub w energię promieniowania w żarówce czy diodzie LED. Ponieważ jednak sprawność tych elementów nigdy nie osiąga 100 %, część energii elektrycznej zawsze zamienia się w ciepło. W niektórych urządzeniach (np. w grzejnikach elektrycznych) jest to efekt pożądany. W innych jednak wpływa on negatywnie na działanie sprzętu, przyśpiesza jego starzenie oraz degradację, co w konsekwencji prowadzi do awarii. Negatywne skutki nie zależą jednak bezpośrednio od samej ilości wydzielającego się w elemencie ciepła, lecz od osiąganej przez element temperatury. To właśnie wysoka temperatura, jaką osiąga dany podzespół w procesie samonagrzewania, stanowi główne źródło problemów. Ostateczna wartość tej temperatury zależy od ilości wydzielonego ciepła oraz od warunków chłodzenia elementu. Skuteczność chłodzenia, czyli zdolność rozpraszania ciepła do otoczenia, opisujemy za pomocą parametru zwanego rezystancją termiczną.

Mała wartość rezystancji termicznej oznacza, że ciepło skutecznie przenika z podzespołu do otoczenia, a co za tym idzie – sam element nagrzewa się jedynie w niewielkim stopniu. Pomiar rezystancji termicznej nie należy do prostych zagadnień, dlatego w praktyce nierzadko spotykamy podzespoły zaprojektowane pod tym kątem niezbyt optymalnie, co skutkuje ich silnym nagrzewaniem, dużą awaryjnością i krótką żywotnością. Dobry przykład stanowią tu źródła światła oparte na diodach LED, którym producenci gwarantują żywotność rzędu 50 tys. godzin i więcej (co oznaczałoby 25 lat użytkowania), a w rzeczywistości często przestają one świecić często już po kilku miesiącach. Stanowi to bezpośredni skutek zbyt dużej rezystancji termicznej oraz wysokiej temperatury złącza półprzewodnikowego, które ulega w takich warunkach szybkiej degradacji i uszkodzeniu.

W elementach czysto rezystancyjnych cała energia elektryczna dostarczana ze źródła zasilania zamienia się w ciepło. Przykładem takiego elementu

jest rezystancyjny czujnik temperatury RTD (**Resistance Temperature Detector**), np. czujnik platynowy Pt1000, który stosowaliśmy w ćwiczeniu 7. Zasadniczo czujniki termorezystancyjne konstruuje się z myślą o pracy w wysokich temperaturach, dlatego samo zjawisko samonagrzewania nie powinno powodować ich fizycznego uszkodzenia. Przykładowo, zgodnie z normą [43], czujnik Pt1000 może pracować w szerokim przedziale temperatur: od -200 do +850°C. Niemniej jednak efekt samonagrzewania czujnika skutkuje dodatkowymi błędami pomiaru temperatury. Dlatego też zjawisko to stanowi kluczowe zagadnienie podczas wykonywania pomiarów temperatury za pomocą tego typu czujników.

### 8.3.2. Efekt samonagrzewania w czujnikach termorezystancyjnych

Termorezystancyjne, platynowe czujniki temperatury znajdują obecnie dwa główne zastosowania: jako czujniki termometrów wzorcowych (**Standard Platinum Resistance Thermometers** – SPRTs) oraz czujniki przemysłowe (**Industrial Platinum Resistance Thermometers** – IPRTs) [43]. Czujniki wzorcowe SPRTs wykorzystujemy przede wszystkim do realizacji skali temperatury ITS-90, opisanej w serii przewodników Międzynarodowego Biura Miar i Wag (BIPM). Wzorcowe termometry platynowe SPRTs kalibrujemy w ściśle określonych punktach stałych. Wykorzystując odpowiednie procedury interpolacyjne, stosujemy je do odtwarzania skali temperatury ITS-90 w zakresie pomiędzy punktem potrójnym równowagi wodoru (13,8033 K) a temperaturą krzepnięcia srebra (1234,93 K).

W ćwiczeniu badamy czujnik przemysłowy IPRT. Wymagania stawiane czujnikom przemysłowym IPRTs określają liczne normy, z których najczęściej stosujemy normę europejską [43] oraz amerykańską [45]. Normy te w dużej mierze pokrywają się ze sobą, jednak różnią się istotnie w zakresie definicji klas tolerancji. Typowa wartość rezystancji czujników IPRTs w temperaturze 0°C wynosi 100 Ω, jednak norma dopuszcza również wartości 10 Ω, 500 Ω oraz 1000 Ω.

Czujniki rezystancyjne należą do grupy czujników parametrycznych (zwanych również pasywnymi), które nie generują na wyjściu własnego sygnału, a jedynie zmieniają określony parametr – w tym przypadku rezystancję. Dlatego też tego typu elementy wymagają odpowiedniego zasilania, aby mogły wygenerować użyteczny sygnał pomiarowy. W celu pomiaru rezystancji

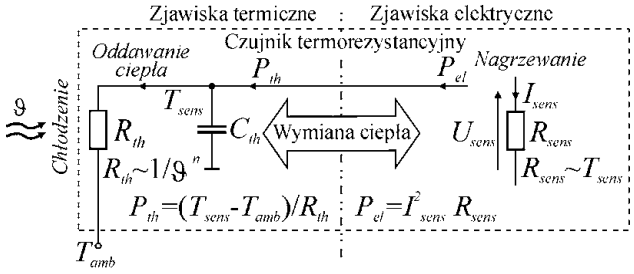
wymuszamy przepływ przez czujnik odpowiednio dobranego prądu i mierzymy powstały na nim spadek napięcia. Prąd przepływający przez czujnik powoduje wydzielanie się w nim ciepła i wzrost jego temperatury, co nazywamy efektem samonagrzewania (**self-heating effect**). W rezultacie czujnik nagrzewa się powyżej temperatury otoczenia, co bezpośrednio skutkuje powstaniem dodatniego błędu pomiaru.

Zgodnie z normą europejską [43] producent wyznacza efekt samonagrzewania dla czujników IPRTs w temperaturze pomiędzy 0°C a 30°C, w powietrzu przepływającym z prędkością (3±0,3) m/s oraz/lub w wodzie przepływającej z prędkością powyżej 0,2 m/s. Samonagrzewanie czujnika w tych warunkach nie przekracza 25 % wartości tolerancji (wynikającej z deklarowanej klasy tolerancji czujnika) przy maksymalnym deklarowanym prądzie pomiarowym. Zazwyczaj prąd ten nie przekracza 1 mA dla czujnika o rezystancji 100 Ω.

Tolerancja czujników przemysłowych IPRTs, niezależnie od ich rezystancji znamionowej, w temperaturze 0°C wynosi od ±0,1°C do ±0,6°C, a na końcu zakresu pomiarowego – od ±0,36°C do ±6,6°C (zależnie od klasy tolerancji) [43, 45]. Według normy europejskiej [43] dobieramy tak małą wartość prądu pomiarowego, aby samonagrzewanie czujnika nie przekraczało 25 % wartości wynikającej z jego deklarowanej klasy tolerancji. Z drugiej jednak strony większy prąd pomiarowy zapewnia wyższą dokładność pomiaru rezystancji elementu. Ustalenie optymalnej wartości prądu pomiarowego wymaga znajomości rezystancji termicznej, która silnie zależy od sposobu montażu samego czujnika i parametrów otaczającego go medium. Z tego powodu podany w karcie katalogowej współczynnik samonagrzewania czujnika (wyznaczony przez producenta zgodnie z normą [43]) jest dla użytkownika całkowicie nieprzydatny w większości praktycznych zastosowań. Dlatego też w celu odpowiedniego doboru prądu pomiarowego często musimy wyznaczyć rezystancję termiczną czujnika *in situ*.

Zjawiska cieplne zachodzące w czujniku termorezystanycjnym wyjaśniono na rycinie 8.5 [46]. Czujnik termorezystancyjny pracuje na styku dwóch obszarów: zjawisk elektrycznych i termicznych. Prąd $I_{sens}$ płynący przez rezystancję elektryczną czujnika $R_{sens}$ powoduje wydzielanie energii, a miarą szybkości tego zjawiska jest moc elektryczna, wyrażona wzorem $P_{el} = I^2_{sens} R_{sens.}$ Energia elektryczna dostarczona do czujnika zamienia się w ciepło, przy czym liczbowo moc elektryczna odpowiada mocy cieplnej $P_{el} = P_{th.}$ Wydzielające się

ciepło kumuluje się w czujniku, co podnosi jego temperaturę $T_{sens}$ względem temperatury otoczenia $T_{amb.}$ Powstała w ten sposób różnica temperatur skutkuje przekazywaniem ciepła z elementu do otoczenia. Szybkość narastania temperatury zależy od wygenerowanej mocy cieplej $P_{th}$ oraz pojemności cieplnej $C_{th.}$ Z kolei szybkość przekazywania ciepła do otoczenia zależy od różnicy pomiędzy temperaturą czujnika $T_{sens}$ a temperaturą otoczenia $T_{amb,}$ jak również od rezystancji termicznej $R_{th}$ tego elementu. Czujnik osiąga termiczny stan ustalony, gdy jego temperatura wzrasta do takiej wartości, przy której moc cieplna $P_{th}$ oddawana do otoczenia zrównuje się z mocą elektryczną $P_{el}$ wydzielaną wewnątrz elementu. Kluczową rolę w całym tym procesie wymiany ciepła odgrywa właśnie rezystancja termiczna czujnika $R_{th.}$




**Ryc. 8.5.** Zjawiska elektryczne i termiczne w czujniku termorezystancyjnym

### 8.3.3. Wyznaczanie rezystancji termicznej czujnika

Rezystancję termiczną czujnika $R_{th}$ definiujemy jako stosunek różnicy temperatury czujnika $T_{sens}$ i temperatury otoczenia Tamb (ośrodka, w którym znajduje się ten element) do mocy cieplnej $P_{th} = P_{el}$, wywołującej zjawisko samonagrzewania:

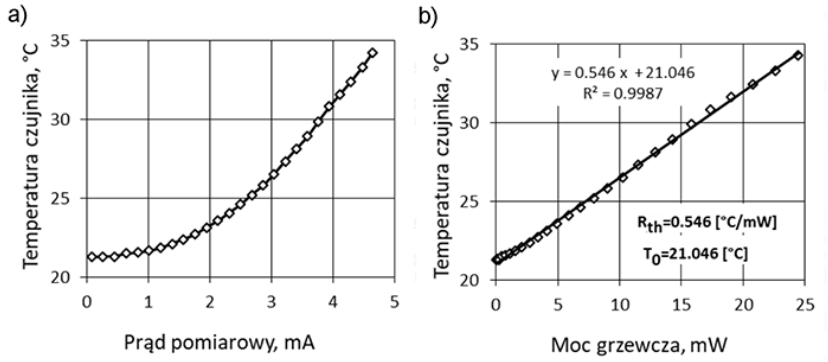
$$R_{th} = \frac{T_{sens} - T_{amb}}{P_{el}} = \frac{T_{sens} - T_{amb}}{I_{sens}^2 R_{sens}} \tag{8.8}$$

Temperatura czujnika $T_{sens}$ jest zawsze wyższa od mierzonej temperatury otoczenia $T_{amb,}$ co bezpośrednio prowadzi do błędów pomiarowych wynikających z efektu samonagrzewania. W ćwiczeniu 8 ośrodkiem, w którym umieszczamy badany czujnik, jest nieruchome powietrze w pomieszczeniu laboratorium.

Z przekształcenia wzoru (8.8) wynika, że temperatura czujnika $T_{sens}$ zależy liniowo od mocy grzania $P_{el}$ – równej iloczynowi kwadratu prądu $I_{sens}$ płynącego przez element i jego rezystancji $R_{sens}$ (która ulega niewielkim zmianom wskutek zjawiska samonagrzewania):

$$T_{sens} = R_{th} I_{sens}^2 R_{sens} + T_{amb} = R_{th} P_{el} + T_{amb} = ax + b\,. \qquad (8.9)$$

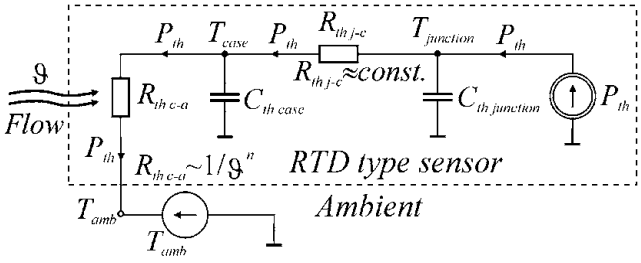
Przekształcony wzór (8.9) pozwala wyznaczyć rezystancję termiczną $R_{th}$ czujnika *in situ*, co objaśniono na rycinie 8.6. W pierwszej kolejności wykonujemy badanym czujnikiem serię pomiarów temperatury przy narastających wartościach prądu $I_{sens}$. Przykładowy wykres dla tego etapu pokazano na rycinie 8.6a. Następnie dla każdej wartości prądu $I_{sens}$ obliczamy moc grzewczą *Pel* uwzględniając przy tym zmieniającą się rezystancję czujnika *R*sens (zgodnie ze wzorem 8.3). Na tej podstawie sporządzamy wykres temperatury $T_{sens}$ w funkcji mocy grzewczej $P_{el}$, który ostatecznie aproksymujemy linią prostą ($y = ax+b$) metodą najmniejszych kwadratów, co widać na rycinie 8.6b. Poszukiwana rezystancja termiczna elementu odpowiada współczynnikowi kierunkowemu wyznaczonej prostej ($R_{th} = a$), natomiast jej wyraz wolny określa rzeczywistą temperaturę otoczenia, pozbawioną wpływu efektu samonagrzewania ($T_{amb} = b$).




**Ryc. 8.6.** Zasada wyznaczania rezystancji termicznej: a) efekt samonagrzewania czujnika, b) wyznaczanie rezystancji termicznej czujnika

W ramach ćwiczenia 8 opracowujemy program do automatycznego wyznaczania rezystancji termicznej czujnika Pt1000. Pomiary realizujemy w pętli dla narastających wartości napięcia zadawanego z przetwornika DAC, podobnie jak w ćwiczeniu 6. Zakres zadawanych wartości napięcia oraz wartość

zastosowanego rezystora wzorcowego $R_N$ ustalają zakres wartości prądu pomiarowego $I_{sens}$ badanego czujnika. Po ustawieniu każdej kolejnej wartości napięcia odmierzamy zadeklarowany czas opóźnienia w pętli (niezbędny na ustalenie się stanu cieplnego czujnika), po czym wykonujemy pomiary napięcia $U_X$ na rezystancji czujnika $R_{sens}$ oraz napięcia $U_N$ na rezystorze wzorcowym $R_N$. Każdy pomiar napięcia ($U_X$ oraz $U_N$) powtarzamy 100 razy, a do dalszych obliczeń bierzemy wartość średnią. Następnie, korzystając z prawa Ohma, obliczamy prąd pomiarowy czujnika $I_{sens}$ (wzór 8.1), rezystancję czujnika $R_{sens}$ (wzór 8.2) oraz moc grzewczą $P_{el}$ (wzór 8.3). Temperaturę czujnika $T$sens obliczamy z funkcji odwrotnej do liniowej charakterystyki elementu (wzór 8.5). Na podstawie tych obliczeń tworzymy pierwsze wykresy, przedstawione na rycinie 8.6a. Z kolei na drugim wykresie (8.6b) prezentowana jest zależność temperatury czujnika $T_{sens}$ od mocy grzewczej *Pel*, którą następnie aproksymujemy linią prostą ($y = ax+b$) metodą najmniejszych kwadratów. Ze współczynników tak wyznaczonej prostej odczytujemy rezystancję termiczną czujnika ($R_{th} = a$) oraz temperaturę otoczenia ($T_{amb} = b$) na podstawie wzoru (8.9). Uzyskane wykresy program zapisuje do plików w postaci graficznej, a wyniki liczbowe – do pliku tekstowego.




**Ryc. 8.7.** Schemat zastępczy rezystancyjnego czujnika temperatury uwzględniający stałą rezystancję termiczną wewnętrzną $R_{th\ j\text{-}c}$ i zmienną rezystancję termiczną zewnętrzną $R_{th\ c\text{-}a}$

Dokładniejsza analiza zjawiska transportu ciepła w czujniku termorezystancyjnym wymaga rozróżnienia efektu wewnętrznego i zewnętrznego. Efekt wewnętrzny określa przepływ ciepła pomiędzy platynowym elementem czynnym czujnika (*junction*) a jego obudową (*case*); wynika on przy tym wyłącznie z konstrukcji samego elementu. Z kolei efekt zewnętrzny natomiast wynika z przepływu ciepła pomiędzy obudową czujnika a jego otoczeniem (*ambient*) i zależy od sposobu zamocowania układu oraz parametrów fizycznych otaczającego go medium. Odpowiedni schemat zastępczy czujnika przedstawiono

na rycinie 8.7. Efekt wewnętrzny reprezentuje wewnętrzna rezystancja termiczna $R_{\text{th j-c}}$ pomiędzy złączem a obudową (*junction–case*), która stanowi wartość stałą, niezależną od warunków pracy czujnika. Odpowiednio, efekt zewnętrzny reprezentuje zewnętrzna rezystancja termiczna $R_{\text{th c-a}}$ pomiędzy obudową a otoczeniem (*case–ambient*), której wartość zależy od warunków pracy czujnika.

Ponieważ pojemność cieplna złącza (cienkiej warstwy platyny) $C_{\text{th}}$junction jest znacznie mniejsza od pojemności cieplnej obudowy czujnika (ceramicznej płytki) $C_{\text{th}}$ case, możemy niezależnie zbadać efekt wewnętrzny i zewnętrzny, wykonując pomiary z różną szybkością narastania prądu płynącego przez element. Stanowi to jedno z zadań, które realizujemy w ćwiczeniu 8. Pozwala to na niezależne wyznaczenie wewnętrznej rezystancji termicznej $R_{\text{th j-c}}$ oraz zewnętrznej rezystancji termicznej $R_{\text{th c-a.}}$ Warto również zauważyć, że znajomość rezystancji termicznej $R_{\text{th}}$ ma znaczenie nie tylko podczas badania efektu samonagrzewania w stanie statycznym, ale wpływa także na właściwości dynamiczne samego układu. Rezystancja termiczna *R*th w połączeniu z pojemnością cieplną $C_{\text{th}}$ czujnika (ryc. 8.5) tworzy układ inercyjny pierwszego rzędu, co ma kluczowe znaczenie podczas planowania szybkości wykonywania pomiarów temperatury. Dodatkowo, ponieważ rezystancja termiczna $R_{\text{th}}$ czujnika zależy od prędkości przepływu $\vartheta$ otaczającego go medium (np. powietrza), stwarza to możliwość budowy nowych układów przetworników przepływu typu **Hot-Wire**. Rozwiązania te bazują na estymacji termicznej stałej czasowej czujnika [49] lub na koncepcji przetwornika przepływu na częstotliwość impulsów z termicznym sprzężeniem zwrotnym [50].

## 8.4. Uruchamianie i testowanie stanowiska laboratoryjnego

### 8.4.1. Uruchamianie stanowiska

Przystępując do realizacji ćwiczenia, najpierw sprawdzamy stan stanowiska laboratoryjnego oraz komputera. **Upewniamy się, że zasilanie komputera i panelu ćwiczeniowego jest wyłączone. Moduł pomiarowy musi w tym momencie pozostać odłączony od portu USB komputera**.

Wszelkie zauważone odstępstwa od tego stanu natychmiast zgłaszamy prowadzącemu zajęcia.

Następnie sprawdzamy sam układ pomiarowy zgodnie z opisem w poprzednim punkcie. Jeśli układ jest połączony prawidłowo, zgłaszamy prowadzącemu gotowość do rozpoczęcia ćwiczenia.

**Dopiero po otrzymaniu wyraźnej zgody od prowadzącego włączamy zasilanie samego komputera. Nie włączamy przy tym jeszcze zasilania panelu ćwiczeniowego. Nadal nie dołączamy również modułu pomiarowego do portu USB.**

Po uruchomieniu systemu operacyjnego nie wykonujemy żadnych dodatkowych czynności i przechodzimy bezpośrednio do kolejnego punktu instrukcji.

### 8.4.2. *Self-test* modułu NI USB-6001

Pierwszym krokiem po uruchomieniu komputera jest sprawdzenie, czy w systemie widnieją dwa urządzenia NI USB-6001: jedno fizyczne (jako **Dev1**) i drugie symulowane (jako **Dev2**). W tym celu uruchamiamy program NI MAX, rozwijamy listę **My System > Devices and Interfaces** i sprawdzamy, jakie moduły pomiarowe wykrywa obecnie system. Program wyświetla zainstalowany i prawidłowo podłączony moduł fizyczny jako aktywny (czarna ikona). Z kolei prawidłowo skonfigurowany moduł symulowany zawsze widnieje jako aktywny (żółta ikona). Jeśli wcześniej zainstalowaliśmy moduł fizyczny, ale obecnie nie jest on dołączony do komputera, system wyświetli go jako nieaktywny, oznaczając go dodatkowym znakiem „x" na czerwonym tle. Upewniamy się zatem, że komputery laboratoryjne mają zainstalowane jedno aktywne urządzenie fizyczne o nazwie **Dev1** oraz drugie symulowane o nazwie **Dev2**. W systemie mogą znajdować się również zainstalowane inne urządzenia fizyczne, jednak muszą one pozostać nieaktywne.

Notujemy w protokole nazwy, typu, numery seryjne i statusy zainstalowanych w systemie urządzeń z serii NI USB. Ponieważ moduł używany na panelu ćwiczeniowym pozostaje jeszcze odłączony, na liście widnieje przy nim czerwony znak „x" oraz status **Not Present**. Jeśli na liście nie ma wymaganych urządzeń, instalujemy je ponownie zgodnie z opisem w instrukcji do ćwiczenia 1.

Pamiętajmy, że moduł zainstalowany w systemie podczas wcześniejszych zajęć mógł zostać z nieznanych powodów (najczęściej przypadkowo) usunięty przez innych studentów. Dlatego też na początku zajęć zawsze upewniamy się, jaki numer urządzenia system przypisał do aktualnie używanego modułu pomiarowego. Wszystkie przykłady programów zamieszczone w instrukcji wykorzystują moduł fizyczny o nazwie **Dev1**; jeśli zatem nasz moduł figuruje w systemie pod inną nazwą, musimy odpowiednio zmodyfikować tworzone programy.

**Zgłaszamy prowadzącemu gotowość, a po otrzymaniu zgody dołączamy moduł pomiarowy NI USB-6001 do portu USB komputera. Nie załączamy jeszcze zasilania panelu ćwiczeniowego.** Obserwujemy zachowanie niebieskiej diody LED na obudowie modułu. Wyniki tych obserwacji notujemy w protokole.

Jeśli poprawnie zainstalowaliśmy urządzenie NI USB-6001, w programie NI MAX przeprowadzamy *self-test* modułu fizycznego i symulowanego zgodnie z opisem zamieszczonym w instrukcji do ćwiczenia 1. Wyniki tego testu notujemy w protokole.

**Zgłaszamy prowadzącemu gotowość, a po otrzymaniu zgody włączamy zasilanie panelu ćwiczeniowego czerwonym przyciskiem kołyskowym.** Podświetlenie przycisku potwierdza obecność napięcia zasilającego.

### 8.4.3. Testowanie przetwornika cyfrowo-analogowego

Test przetwornika cyfrowo-analogowego wykonujemy podobnie jak w opisie ćwiczenia 5, przy dołączonym do panelu ćwiczeniowego module z badanym czujnikiem Pt1000 i rezystorem wzorcowym (zgodnie ze schematem z ryciny 8.2).

**UWAGA!** Zaciski laboratoryjne dokręcamy ostrożnie i z wyczuciem, ponieważ bardzo łatwo uszkodzić delikatny gwint w ich nakrętkach. W gwintowanej części zacisków znajduje się otwór, przez który przewlekamy dołączany przewód. Zapewnia to pewny kontakt elektryczny. Niestety badany czujnik nie sygnalizuje w widoczny sposób pojawienia się napięcia na wyjściu wzmacniacza, tak jak miało to miejsce w przypadku dołączonej do układu diody LED. Dlatego też musimy jednocześnie badać przetworniki DAC i ADC.

W oknie programu NI MAX wybieramy z listy urządzeń moduł **Dev1** i otwieramy zakładkę **Test Panels**. Następnie w otwartym oknie **Test Panels** przechodzimy do zakładki **Analog Output**. Konfigurujemy test wyjścia analogowego, zwracając szczególną uwagę, aby w polu **Channel Name** wybrać wyjście **Dev1/ao0**. W polu **Output Value** wpisujemy wartość 0 V i uruchamiamy test, naciskając przycisk **Update**. Niestety nie mamy bezpośredniej możliwości zaobserwowania, czy działanie to przyniosło oczekiwany skutek. Wykorzystamy w tym celu przetwornik ADC. Pozostawiamy otwarte okno **Analog Output** (skonfigurowane do testowania wyjścia analogowego **Dev1/ao0**) i przechodzimy do testów przetwornika analogowo-cyfrowego.

### 8.4.4. Testowanie przetwornika analogowo-cyfrowego

Test przetwornika analogowo-cyfrowego wykonujemy identycznie jak w opisie ćwiczenia 5. Mierzymy dwa napięcia: na badanym czujniku Pt1000 i na rezystorze wzorcowym $R_N$. Ponieważ oba te elementy mają zbliżoną rezystancję, napięcia na nich również przybierają zbliżone wartości. Czujnik Pt1000 w temperaturach dodatnich ma jednak rezystancję większą od 1000 Ω, dlatego też napięcie na nim jest nieco wyższe. Pozostawiamy otwarte okno **Test Panels** z uruchomionym testem przetwornika cyfrowo-analogowego w aktywnej zakładce **Analog Output**. Ponownie w głównym oknie programu NI MAX otwieramy zakładkę **Test Panels**. Otworzy się wówczas kolejne okno **Test Panels**, w którym wybieramy zakładkę **Analog Input**. Ustawiamy na ekranie obok siebie oba okna **Test Panels: Analog Output** i **Analog Input**. W polu **Output Value** pozostawiamy wpisaną wartość 0 V, zatwierdzoną wcześniej przyciskiem **Update**.

Na karcie **Analog Input** wybieramy następujące opcje: **Channel Name**: **Dev1/ai2, Mode: On Demand, Input Configuration: Differential, Max Input Limit: 10, Min Input Limit: −10**. Uruchamiamy test przyciskiem **Start**. Uzyskany wynik również jest zbliżony do 0 V. Zatrzymujemy pomiar przyciskiem **Stop** i powtarzamy całą procedurę dla wejścia **Dev1/ai3**. Wynik powinien również być zbliżony do 0 V. Zaobserwowane wyniki notujemy w protokole.

Ponownie na karcie **Analog Input** uruchamiamy pomiar dla wejścia **Dev1/ai2**. Obserwując cały czas napięcie mierzone w kanale **Dev1/ai2**, jednocześnie w polu **Output Value** wpisujemy wartość 5 V i zatwierdzamy ją

przyciskiem **Update**. Przez dłuższą chwilę obserwujemy wyniki pomiaru napięcia na czujniku **Pt1000** w kanale **Dev1/ai2**. Początkowo przyjmują one wartość zbliżoną do 5 V, a następnie powoli narastają. Po kilkudziesięciu sekundach wskazania stabilizują się. Notujemy początkową i końcową wartość napięcia na czujniku **Pt1000**. W ten sposób przeprowadziliśmy pierwszy test efektu samonagrzewania czujnika termorezystancyjnego. Następnie przełączamy pomiar na wejście **Dev1/ai3**. Odczytane w kanale **Dev1/ai3** napięcie również notujemy w protokole.

W obu kanałach odczytujemy napięcia bliskie 5 V. Na czujniku **Pt1000** występuje jednak spadek napięcia większy o kilkaset mV, ponieważ jego rezystancja w temperaturze pokojowej jest o około 100 Ω wyższa od rezystancji wzorcowej $R_N = 1000\ \Omega$. Z kolei na rezystorze $R_N$ napięcie przyjmuje wartość poniżej 5 V. Suma napięć odczytanych w obu kanałach wynosi około 10 V, ponieważ układ dwukrotnie wzmacnia napięcie z przetwornika DAC (równe 5 V). Korzystając ze wzoru (8.1), łatwo obliczamy, że przez czujnik Pt1000 płynie prąd o natężeniu około 5 mA, co z kolei powoduje wydzielanie się w nim mocy około 25 mW. Moc ta zamienia się w ciepło, które podnosi temperaturę elementu, a co za tym idzie – zwiększa jego rezystancję. To z tego powodu obserwowany wcześniej spadek napięcia na czujniku powoli narastał. W ten sposób zaobserwowaliśmy zjawisko samonagrzewania czujnika termorezystancyjnego w praktyce.

Czy test potwierdził sprawność całego stanowiska? Czy zaobserwowaliśmy efekt samonagrzewanie czujnika Pt1000? Wnioski z przeprowadzonych testów zapisujemy w protokole. Dodatkowo na potrzeby raportuwykonujemy przykładowy zrzut ekranu z przeprowadzonego testu (obejmujący dwa otwarte okna: DAC oraz ADC) i zapisujemy go do pliku.

Po uzyskaniu pozytywnych wyników testu wpisujemy w polu **Output Value** wartość 0 V i zatwierdzamy, naciskając przycisk **Update**. Ponownie na karcie **Analog Input** uruchamiamy pomiar dla wejścia **Dev1/ai2** i upewniamy się, że napięcie na czujniku spadło do zera. Jest to bardzo ważne, ponieważ **zamknięcie programu NI MAX nie ustawia automatycznie na wyjściu analogowym napięcia równego zero**. Zamykamy oba okna **Test Panels** i kończymy pracę programu **NI MAX**. Następnie uruchamiamy środowisko LabVIEW i przechodzimy do kolejnego punktu instrukcji.

## 8.5. Programowanie

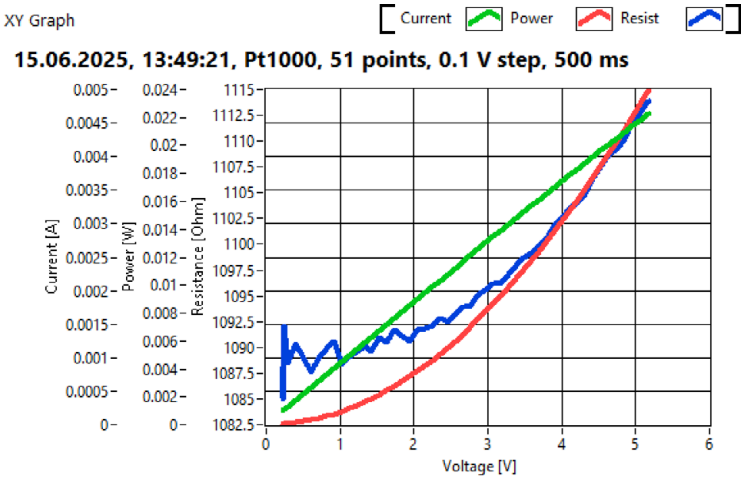
### 8.5.1. Pierwszy program do badania efektu samonagrzewania

Ćwiczenie 8 jest kontynuacją ćwiczenia 6. Trzeci program opracowany w ćwiczeniu 6 stanowi podstawę do rozpoczęcia ćwiczenia 8. Znajdujemy ten program na dysku komputera w katalogu ćwiczenia 6 i wczytujemy go do środowiska LabVIEW. W pierwszej kolejności robimy jego kopię na potrzeby pierwszego zadania w ćwiczeniu 8, zmieniając numer programu na 8-1-a. Pamiętajmy, że nie wolno modyfikować programu, który stanowi ostateczną wersję poprzednio wykonanego zadania.

Stanowisko laboratoryjne łączymy tak jak podczas wstępnych testów. Do modułu z zaciskami laboratoryjnymi, w miejsce dwójnika $R_{X,}$ dołączamy badany czujnik Pt1000. Jako rezystor wzorcowy dołączamy element $R_N = 1000\ \Omega$.

W programie ustawiamy następujące parametry: 51 punktów charakterystyki z krokiem napięcia 0,1 V, napięcie początkowe 0 V, opóźnienie czasowe 500 ms, $R_N = 1000\ \Omega$. Włączamy automatyczne skalowanie na wszystkich osiach. Wprowadzamy do opisu wykresu informacje o konfiguracji programu, np.: Pt1000, 51 points, 0,1 V step, 500ms. Uruchom program i upewniamy się, że działa poprawnie. Test wykonuje się w czasie 25 sekund. Program zadaje do przetwornika DAC narastające wartości napięcia od 0 do 5 V. Na wyjściu wzmacniacza powinno pojawić się napięcie w zakresie od 0 do 10 V. Napięcie na czujniku osiąga wartość około 5 V, a prąd około 5 mA. Końcowa wartość mocy jest zbliżona do 25 mW. Rezystancja czujnika zwiększa się o kilkadziesiąt omów w wyniku samonagrzewania elementu. Po wykonaniu pomiarów program automatycznie zapisuje do pliku wykres oraz plik tekstowy z wynikami pomiarów napięcia i prądu. Odszukujemy zapisane pliki i sprawdzamy ich zawartość. Przykładowy wykres przedstawiono na rycinie 8.8. Oceniamy, czy otrzymany wykres jest zgodny z oczekiwaniami. Przy tych pomiarach kluczowy jest czas, podobnie jak podczas badania żarówki halogenowej w ćwiczeniu 6. Szybkie zmiany napięcia nie pozwalają na nagrzanie się czujnika, przez co zmiany rezystancji są mniejsze. Przy dłuższym czasie pomiaru czujnik nagrzewa się do wyższej temperatury. Uruchamiamy program dla opóźnienia równego 10 ms i 1000 ms, wpisując do opisu wykresu ustawioną wartość. Porównujemy ze sobą wykresy, które program zapisał dla różnych nastaw. Jakie tym razem

otrzymaliśmy wyniki? Jakie różnice są widoczne na wykresach? Nasze wnioski zapisujemy w protokole.




**Ryc. 8.8.** Przykładowe wykresy dla czujnika Pt1000 otrzymane trzecim programem zrealizowanym w ćwiczeniu 6

Analizując wykresy na rycinie 8.8, zauważamy, że rezystancja czujnika wzrosła od wartości początkowej ok. 1090 Ω do wartości 1114 Ω. Uwzględniając zależność (8.5), obliczamy, że temperatura otoczenia wynosiła ok. 23,4°C, a czujnik nagrzał się do temperatury ok. 29,6°C przy mocy 24 mW. Na podstawie wzoru (8.8) obliczamy, że rezystancja termiczna czujnika wynosi 0,26°C/mW. W protokole wykonujemy analogiczne obliczenia na podstawie własnego wykresu. Czy otrzymaliśmy podobną wartość rezystancji termicznej? Wnioski zapisujemy w protokole.

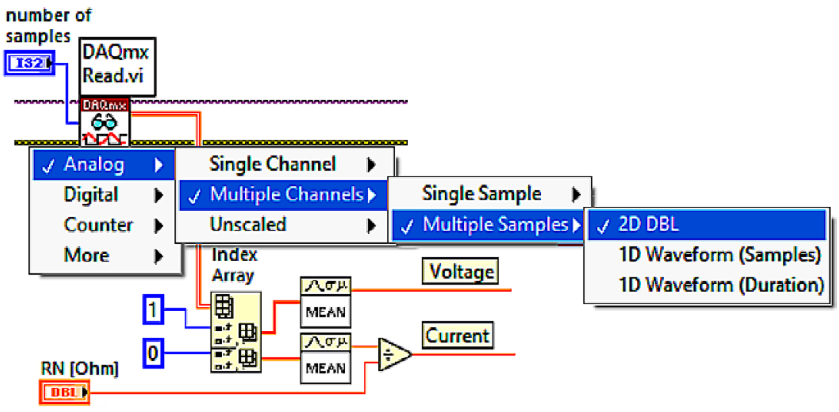
Celem ćwiczenia 8 jest opracowanie programu, który automatycznie wyznacza rezystancję termiczną czujnika, uwzględniając przy tym jej zależność od szybkości narastania prądu płynącego przez element. Zadanie to realizujemy etapami.

Najpierw usuwamy z programu zbędne elementy, a następnie uzupełniamy go o nowe funkcje. Podczas modyfikacji często zapisujemy program pod kolejnymi nazwami, uruchamiamy go i upewniamy się, czy przypadkiem nie usunęliśmy zbyt dużej części diagramu.

Przełączamy się na diagram i usuwamy z niego elementy związane z pracą w trybie symulacji. W pierwszej kolejności kasujemy dwie funkcje **Select**.

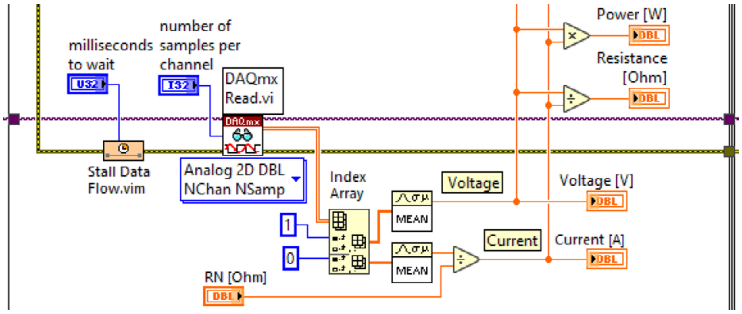
Na diagramie pojawia się wówczas wiele połączeń oznaczonych jako błędne, ale nie kasujemy ich jeszcze! Łączymy górne wyjście funkcji **Index Array** z wejściem wskaźnika napięcia **Voltage** [V]. Dolne wyjście funkcji **Index Array** prowadzi do funkcji dzielenia **Divide**, która oblicza wartość prądu. Łączymy wyjście tej funkcji **Divide** z wejściem wskaźnika prądu **Current** [A]. W ten sposób odtworzyliśmy połączenia niezbędne do dalszej pracy programu. Kasujemy przełącznik trybu pracy **Pomiar – Symulacja**. Kasujemy zadajnik wzmocnienia symulowanego wzmacniacza **Amplification** oraz połączoną z nim funkcję mnożenia **Multiply**. Kasujemy zadajnik symulowanej rezystancji **R Simul [Ohm]** oraz połączoną z nim funkcję dzielenia **Divide**. Dopiero teraz usuwamy wszystkie połączenia zaznaczone jako błędne poleceniem **Remove Broken Wires** (**Ctrl+B**). Przełączamy się na panel, ustawiamy krótki czas opóźnienia (10 ms) i uruchamiamy program. Jeśli program działa prawidłowo i wygenerował poprawne wykresy, robimy kopię programu do dalszych modyfikacji w tym punkcie ćwiczenia, zmieniając jego numer na 8-1-b.

Na rycinie 8.8 można zauważyć, że wykres rezystancji wykazuje duże błędy przypadkowe, szczególnie dla małych wartości napięcia. Podobny efekt obserwowaliśmy już w ćwiczeniu 6 (na rycinie 6–20b). Zagadnienie to analizowaliśmy również w ćwiczeniu 7 (na rycinach 7–10 i 7–12) przy okazji tworzenia programu do pomiaru temperatury. Rozwiązaniem tego problemu jest uśrednianie wyników z serii pomiarów. W tym celu modyfikujemy konfigurację funkcji **Read** w taki sposób, aby program wykonywał serię pomiarów. Sposób postępowania przedstawiamy na rycinie 8.9.




**Ryc. 8.9.** Konfigurowanie funkcji Read do wykonywania serii pomiarów

Rozwijamy selektor poniżej funkcji **Read** i wybieramy kolejno opcje: **Analog > Multiple Channels > Multiple Samples > 2D DBL**. Na wejściu **number of samples per channel** funkcji **Read** tworzymy zadajnik (Create Control) i po przejściu na panel nadajemy mu wartość 100. Linia wyjściowa z funkcji **Read** zmienia się na podwójną, a na diagramie pojawiają się błędne połączenia. Musimy zatem uśrednić wyniki z serii pomiarów. Kasujemy połączenie na górnym wyjściu funkcji **Index Array** i wstawiamy w to miejsce funkcję obliczającą wartość średnią (**Mathematics > Probality & Statistic > Mean**), co pokazano na rycinie 8.10. Uzupełniamy brakujące połączenia. Analogicznie wstawiamy funkcję **Mean** na dolnym wyjściu funkcji **Index Array**. Wszystkie typy zmiennych powinny być teraz uzgodnione. Wszystkie połączenia na diagramie powinny być zaznaczone jako poprawne. Zapisujemy program, przechodzimy na panel i uruchamiamy go z czasem opóźnienia 10 ms. Porównujemy wykres rezystancji dla uśrednianych 100 pomiarów oraz dla 2 pomiarów. Porównujemy rozrzuty wartości rezystancji na początku wykresu. Zapisujemy je w protokole. Czy widoczny jest pozytywny wpływ uśredniania na dokładność pomiarów? Wniosek ten zapisujemy w protokole.



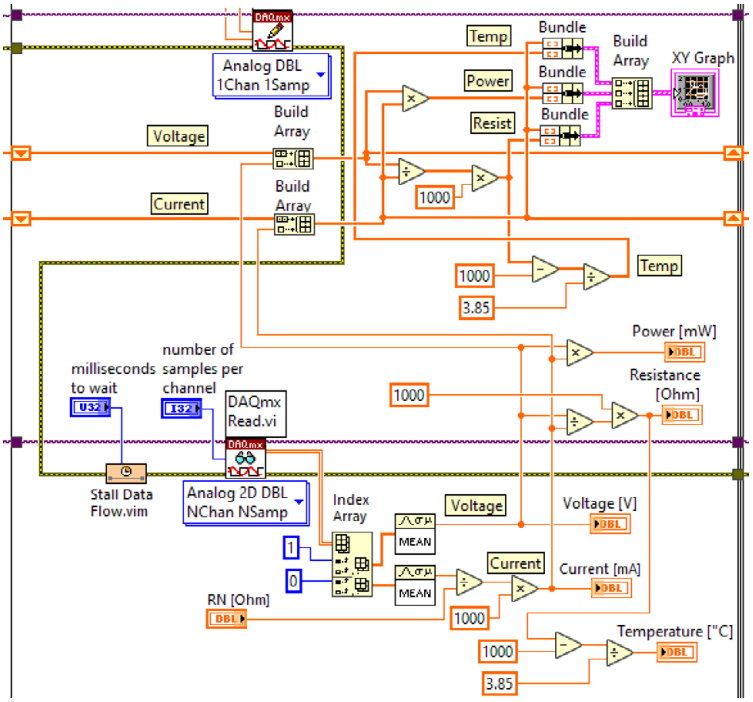

**Ryc. 8.10.** Fragment algorytmu z modyfikacją funkcji Read

W programie stosujemy dwie funkcje wprowadzające opóźnienie do wykonywanych pomiarów. Są to funkcje **Wait Until Next ms Multiple** oraz **Stall Data Flow**. Podczas badania efektu samonagrzewania kluczowy jest czas pomiędzy zadaniem kolejnej wartości prądu płynącego przez czujnik a wykonaniem pomiaru jego rezystancji. Czas ten możemy zmieniać w programie za pomocą opóźnienia **Stall Data Flow**. Odszukujemy tę funkcję na diagramie. Do jej wejścia **miliseconds to wait** dołączona jest wartość stała równa 10 ms. Klikając prawym przyciskiem myszy na tę wartość stałą, zamieniamy ją na

zadajnik (**Change to Control**), a po przejściu na panel nadajemy jej wartość początkową 100. Ponieważ funkcja **Wait Until Next ms Multiple** jest już w tym momencie zbędna, kasujemy ją z diagramu razem z zadajnikiem na jej wejściu. Uruchamiamy program i sprawdzamy jego działanie. Zmieniamy opóźnienie **miliseconds to wait** na 10 ms, 100 ms oraz 1000 ms i upewniamy się, czy szybkość wykonywania pomiarów ulega zmianie.

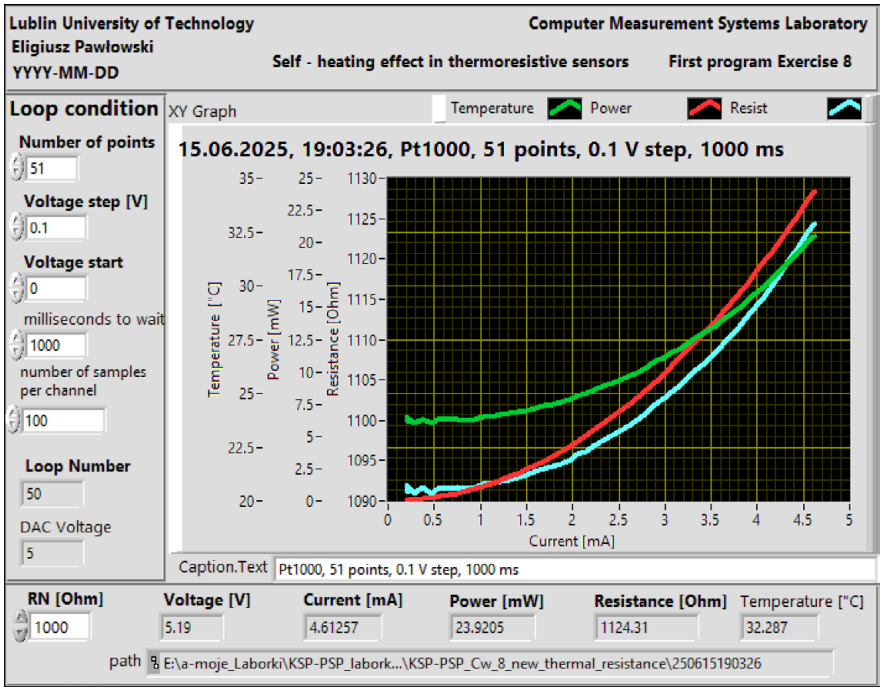
Jeśli program działa prawidłowo i wygenerował poprawne wykresy, robimy kopię programu do dalszych modyfikacji w tym punkcie ćwiczenia, zmieniając jego numer programu na 8-1-c.

W badaniach efektu samonagrzewania czujników termorezystancyjnych wygodniej jest wyrażać wartości prądu w mA, a mocy w mW. Dlatego zmierzoną wartość prądu w amperach mnożymy przez 1000. Dzięki temu wartości mocy od razu otrzymujemy w miliwatach. Niestety rezystancję otrzymujemy wówczas w kiloomach, co nie jest zbyt wygodne. Dlatego obliczoną wartość rezystancji również mnożymy przez 1000. Czynność tę wykonujemy w dwóch miejscach algorytmu: przy obliczaniu pojedynczej wartości i przy obliczaniu tablicy. Sposób wprowadzania tych modyfikacji przedstawiono na rycinie 8.11.




**Ryc. 8.11.** Fragment algorytmu z modyfikacją tworzenia wykresu

Samonagrzewanie czujnika jest skutkiem przepływu prądu przez jego rezystancję. Dlatego zmieniamy oś X tworzonych wykresów z napięcia na prąd. Dane dla osi X tworzonych wykresów są dołączone do górnych wejść funkcji **Bundle**. Aby uzyskać odpowiednią zmianę, na górnych wejściach trzech funkcji **Bundle** kasujemy połączenia z tablicą wartości napięcia i łączymy je z tablicą wartości prądu. Na panelu zmieniamy opis osi X na **Current [mA]**. Uruchamiamy program i oceniamy otrzymany wykres. Na osi X prąd zmienia się do wartości około 5 mA. Wartości mocy są pokazywane w mW, dlatego odpowiednio zmieniamy opis tej osi. Pozostałe osie są tworzone tak samo jak przed modyfikacjami. Jeden z wykresów pokazuje teraz prąd względem prądu, co oczywiście jest zupełnie nieprzydatne. Zamiast tego potrzebujemy wykresu temperatury czujnika względem prądu. Przeliczamy zatem rezystancję czujnika Pt1000 na temperaturę, co robiliśmy już w ćwiczeniu 7. Korzystamy w tym celu z zależności (8.5). Obliczenia wykonujemy w dwóch miejscach algorytmu: przy wyznaczaniu pojedynczej wartości temperatury (pokazywanej na bieżąco na panelu) oraz przy generowaniu tablicy dla wykresu temperatury. Sposób wprowadzania tych modyfikacji przedstawiamy na rycinie 8.11.




**Ryc. 8.12.** Panel programu do badania efektu samonagrzewania czujnika

Po wprowadzeniu zmian na diagramie przechodzimy na panel i modyfikujemy wykres. Przykład takiej modyfikacji przedstawionona rycinie 8.12. Zmieniamy opis osi Y dla prądu na **Temperature [°C]**. W opcjach wykresu (**Properties**) zmieniamy nazwę z **Current** na **Temperature**. Pamiętajmy, że nazwy osi (**Scales**) i nazwy wykresów (**Plots**) definiujemy niezależnie. Zazwyczaj nazwa wykresu i nazwa osi są takie same, jednak nie jest to regułą, a środowisko LabVIEW nie zmienia ich automatycznie. Możemy przykładowo wygenerować kilka wykresów w tych samych osiach (taki przypadek przedstawiliśmy w ćwiczeniu 5 na rycinie 5–20).

Zapisujemy program do pliku. Włączamy autoskalowanie dla wszystkich osi. Ustawiamy opóźnienie na 1000 ms i uruchamiamy program. Otrzymujemy w ten sposób trzy wykresy o bardzo zbliżonym kształcie. Następnie wyłączamy autoskalowanie dla wszystkich osi. Ustawiamy ręcznie zakresy wszystkich osi, dobierając na nich możliwie „równe" wartości.

Uruchamiamy program i sprawdzamy, czy wygenerowane wykresy są zgodne z oczekiwaniami. Zmieniamy opóźnienie na 100 ms, a opis wykresu również na 100 ms. Upewniamy się, że osie mają wyłączone autoskalowanie. Wykonujemy pomiar. Następnie zmieniamy opóźnienie na 10 ms i aktualizujemy opis wykresu. Ponownie wykonujemy pomiar. Odszukujemy zapisane przez program wykresy i porównujemy je ze sobą. Nazwy plików z wykresami notujemy w protokole i używamy ich do przygotowania sprawozdania. Wykresy mocy wyglądają podobnie. Z kolei wykresy rezystancji i temperatury wyraźnie się różnią, co łatwo zauważamy, jeśli wyłączyliśmy autoskalowanie, a zakresy osi na wykresach są takie same. Przy szybkich zmianach prądu przyrost temperatury jest niewielki. Przy powolnym narastaniu prądu otrzymujemy znaczne zwiększenie temperatury czujnika. W ten sposób uzyskaliśmy w pełni funkcjonalny program do badania efektu samonagrzewania w czujnikach termorezystancyjnych.

Jeśli program działa prawidłowo i wygenerował poprawne wykresy, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne powiązania ze sobą. Przykładowy panel tak uzupełnionego programu przedstawiamy na rycinie 8.12.

Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików z opisem zawartości zapisujemy w protokole. Wykorzystujemy zapisane pliki do przygotowania raportu.

Robimy kopię programu do realizacji następnego zadania, zmieniając jego numer programu na kolejny, tzn. 8-2-a. Zawsze pamiętajmy, aby po zapisaniu ostatecznej wersji programu w pełni realizującej dany punkt ćwiczenia wykonać jego kopię i nadać jej nową nazwę, zmieniając ostatni znak na kolejną wartość.

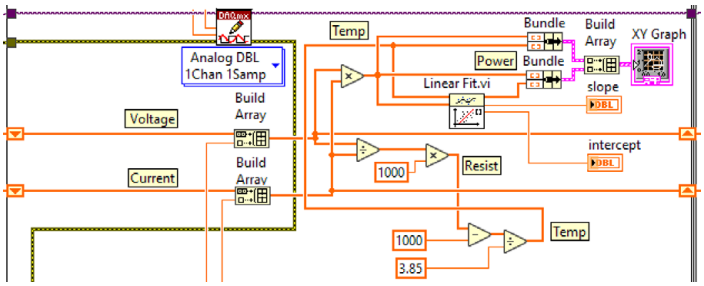
### 8.5.2. Drugi program obliczający rezystancję termiczną czujnika

W drugim programie wyznaczamy rezystancję termiczną czujnika w sposób opisany w punkcie 8.2.3. Zasadę postępowania przedstawiono na rycinie 8.6. Na podstawie pomiarów generujemy wykres zależności temperatury czujnika od mocy wydzielanej w nim przez przepływający prąd. Teoretycznie zależność ta jest liniowa (8.9). Dlatego otrzymany wykres aproksymujemy linią prostą, korzystając z metody najmniejszych kwadratów. Poszukiwana rezystancja termiczna czujnika $R_{th}$ jest równa współczynnikowi kierunkowemu linii prostej aproksymującej wykres.

Program opracowany w poprzednim punkcie wymaga tylko niewielkich modyfikacji. Wszystkie niezbędne dane wyliczyliśmy już w programie. Modyfikujemy zatem sposób tworzenia wykresu i aproksymujemy go linią prostą. W pierwszym kroku modyfikujemy wykres. W obecnej postaci program tworzy trzy wykresy: temperatury, mocy i rezystancji w funkcji prądu. Zamiast tego potrzebujemy wykresu temperatury w funkcji mocy oraz wykresu linii prostej aproksymującej. Dlatego jeden wykres usuwamy, a dwa pozostałe odpowiednio modyfikujemy. Docelową postać zmodyfikowanego fragmentu diagramu przedstawia rycina 8.13.

Na diagramie kasujemy trzecią, najniżej położoną funkcję **Bundle**, która tworzy dane dla wykresu rezystancji. Funkcję **Build Array** zmniejszamy od dołu do góry, pozostawiając w niej tylko dwa wejścia. Przechodzimy na panel i uruchamiamy program. Wykres rezystancji znika, jednak skala rezystancji pozostaje widoczna. Klikając prawym przyciskiem myszy na skalę rezystancji, usuwamy ją z wykresu za pomocą opcji **Delete Scale**. W ten sposób

zrealizowaliśmy pierwszy krok modyfikacji. Robimy kopię programu do dalszych modyfikacji w tym punkcie ćwiczenia, zmieniając jego numer na 8-2-b.

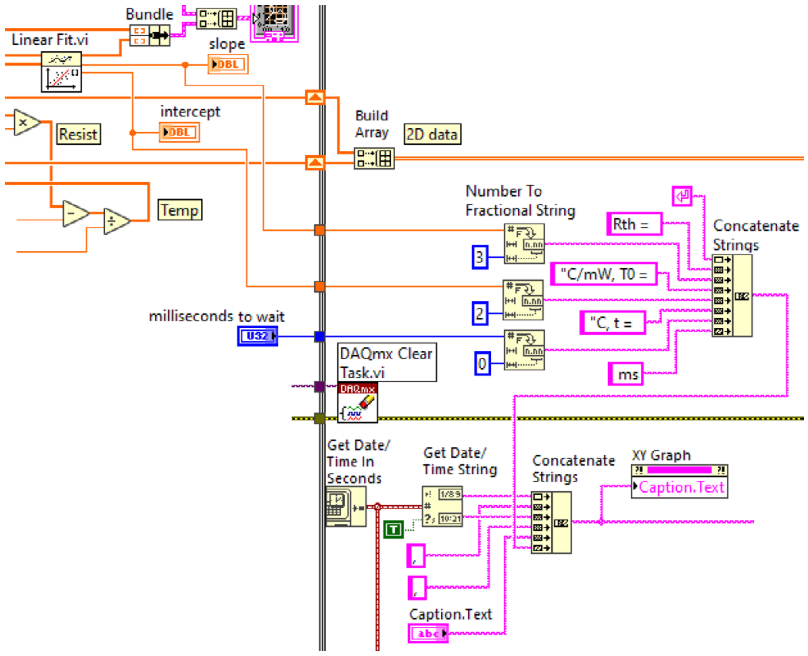



**Ryc. 8.13.** Modyfikacja diagramu do wyznaczania rezystancji termicznej czujnika

Na diagramie kasujemy połączenia na górnych wejściach obu funkcji **Bundle**. Są to dane do tworzenia osi X na wykresie. Dołączamy do tych wejść dane zawierające wartości mocy. Są one dostępne na dolnym wejściu dolnej funkcji **Bundle**. Przechodzimy na panel. Włączamy autoskalowanie na wszystkich osiach. Następnie uruchamiamy program. Otrzymujemy w ten sposób dwa wykresy: wykres temperatury w funkcji mocy oraz wykres mocy w funkcji mocy. Ten pierwszy jest potrzebny do wyznaczenia rezystancji termicznej. Ten drugi jest w tej postaci bezsensowny i zbędny, jednak w dalszych krokach dokonujemy na nim odpowiednich modyfikacji.

Teraz uzupełniamy program o liniową aproksymację wykresu temperatury. Zagadnienie liniowej aproksymacji było już szczegółowo omówione w ćwiczeniu 5. Najpierw wstawiamy na diagramie funkcję **Mathematics > Fitting > Linear Fit**. Do wejścia Y dołączamy tablicę z wartościami temperatury. Do wejścia X dołączamy tablicę z wartościami mocy. Wyjście **Best Linear Fit** dołączamy do dolnego wejścia dolnej funkcji **Bundle**, kasując wcześniej istniejące tam połączenie. Na wyjściach **slope** oraz **intercept** tworzymy wskaźniki numeryczne (klikając prawym przyciskiem myszy i wybierając opcję **Create Indicator**), które pokazują nachylenie i przesunięcie linii aproksymującej. Odszukujemy te wskaźniki na panelu i umieszczamy je w widocznym miejscu. Nazwę osi X zmieniamy na **Power** [mW]. Oś Y dla temperatury pozostawiamy bez zmian. W opcjach wykresu (**Properties**) zmieniamy nazwę z **Power** na **Aprox** oraz przypisujemy do niego oś Y o nazwie **Temperature**. Na wykresie kasujemy zbędną już oś Y dla wykresu mocy **Power**. Pozostaje zatem tylko jedna oś X opisana jako **Power** [mW] oraz jedna oś Y opisana jako **Temperature** [°C].

Uruchamiamy program i sprawdzamy, czy wykres temperatury w funkcji mocy jest poprawnie aproksymowany linią prostą. Wykresy leżą bardzo blisko siebie, co utrudnia ich ocenę. Dlatego w opcjach **Properties** modyfikujemy sposób wyświetlania danych na wykresie. Dla wykresu temperatury wyłączamy rysowanie linii i włączamy rysowanie wyłącznie punktów. Dla prostej aproksymującej postępujemy odwrotnie: wyłączamy rysowanie punktów i włączamy rysowanie linii. Ponieważ dla małych wartości prądów występują duże błędy przypadkowe, ustawiamy napięcie początkowe na 0,5 V. Zwiększamy krok na 0,5 V oraz zmniejszamy liczbę kroków na 10. Ustawiamy opóźnienie na 5000 ms. Uruchamiamy program i oceniamy otrzymany wykres. Powinien on wyglądać podobnie jak na rycinie 8.16. Rezystancja termiczna powinna wynosić około 0,3–0,5°C/mW, na każdym stanowisku będzie miała inną wartość. Zmieniamy opóźnienie na 1000 ms i sprawdzamy, czy przyrost temperatury i rezystancja termiczna uległy zmianie. Uzyskane wartości rezystancji termicznej zapisujemy w protokole.

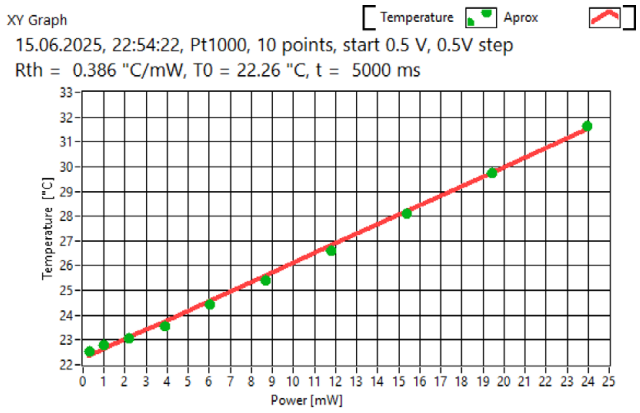



**Ryc. 8.14.** Modyfikacja diagramu do pełnego opisu wykresu

Jeśli program działa poprawnie, robimy kopię programu do dalszych modyfikacji w tym punkcie ćwiczenia, zmieniając jego numer na 8-2-c. Wykres

generowany przez program oraz wyznaczona z niego rezystancja termiczna zależą od szybkości narastania prądu w czujniku, czyli od ustawionego opóźnienia. Program automatycznie zapisuje wykres do pliku, jednak brakuje na nim wyznaczonej wartości rezystancji termicznej oraz ustawionego opóźnienia. Uzupełniamy zatem diagram o automatyczne dodawanie tych informacji do opisu wykresu. Tak uzupełniony diagram przedstawiono na rycinie 8.14.

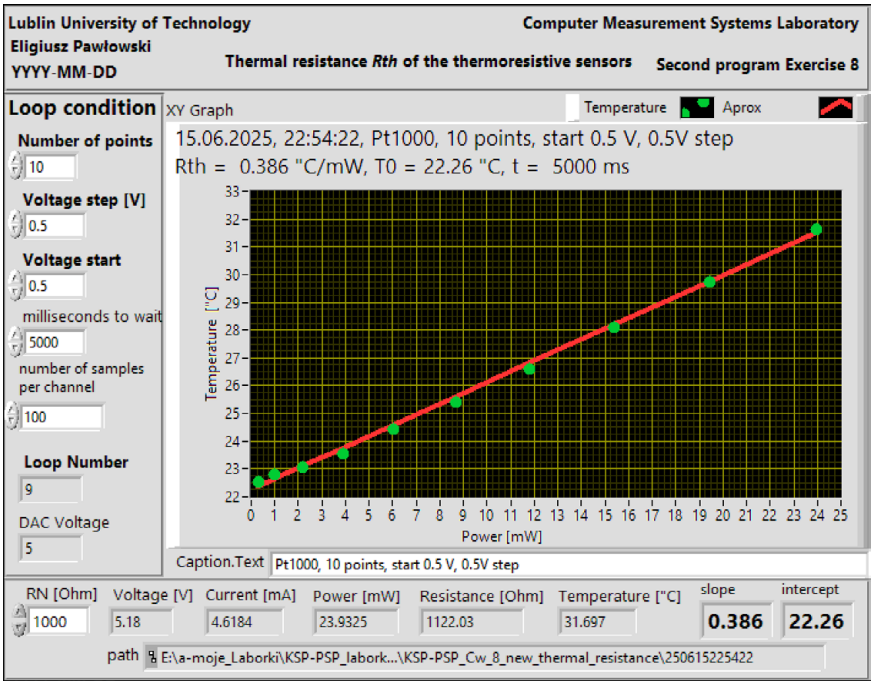
Uzupełniamy opis wykresu o automatyczne dopisywanie współczynników linii prostej aproksymującej wykres temperatury oraz ustawionej wartości opóźnienia w milisekundach. Są to dane liczbowe, które musimy zamienić na tekst. W tym celu wstawiamy na diagramie 3 razy funkcję konwersji liczby na tekst **Number to Fractional String**. Do wejść dołączamy z wnętrza pętli **For** wartości: **slope**, **intercept**, **miliseconds to wait**. Na krawędzi pętli **While** automatycznie pojawiają się tunele, które konfigurujemy w trybie **last value**. Dla każdej funkcji konfigurujemy liczbę miejsc po przecinku zgodnie z ryciną 8.14. Obok, po prawej stronie, wstawiamy funkcję łączenia łańcuchów **Concatenate Strings**. Rozciągamy ją na 8 wejść. Do pierwszego wejścia podłączamy stałą tekstową **Carriage Return**. Na wejściach 2, 4, 6 oraz 8 tworzymy stałe tekstowe zawierające odpowiednie opisy dla kolejnych danych. Do wejść 3, 5 oraz 7 dołączamy przekształcone na tekst dane: **slope**, **intercept**, **miliseconds to wait**. Istniejący już na diagramie obiekt **Concatenate Strings** (tworzący opis wykresu **Caption.Text**) rozciągamy do dołu, dodając jedno wejście. Do tego wejścia dołączamy utworzony łańcuch znaków zawierający wyniki aproksymacji.




**Ryc. 8.15.** Przykładowy wykres z wyznaczoną rezystancją termiczną czujnika umieszczonego w nieruchomym powietrzu

Skalujemy ręcznie oś X na zakres 0–25 mW. Włączamy automatyczne skalowanie osi Y. Wpisujemy w pole opisu wykresu **Caption.Text** odpowiednie informacje: liczbę punktów, napięcie początkowe i krok napięcia. Uruchamiamy program dla opóźnienia 5000 ms. Gdy program wygeneruje wykres, wyłączamy autoskalowanie dla osi Y i ustalamy ręcznie jej zakres, aby otrzymać czytelny widok. Uruchamiamy ponownie program i oceniamy otrzymany wykres. Przykładową postać wykresu przedstawiono na rycinie 8.15.

Czy na wykresie poprawnie wyświetlają się wartości rezystancji termicznej, temperatury początkowej oraz opóźnienia? Sprawdzamy, czy wykres zapisany automatycznie do pliku również zawiera wszystkie te informacje. Zmieniamy opóźnienie na 1000 ms i sprawdzamy, czy przyrost temperatury i rezystancja termiczna uległy zmianie. Wnioski z obserwacji zapisujemy w protokole.




**Ryc. 8.16.** Panel programu do wyznaczania rezystancji termicznej czujnika

Jeśli program działa prawidłowo i wygenerował poprawne wykresy, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami.

Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne powiązania ze sobą. Przykładowy, tak uzupełniony panel programu przedstawiono na rycinie 8.16. Na koniec zapisujemy program do pliku.

Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel oraz diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików z opisem zawartości zapisujemy w protokole. Wykorzystujemy zapisane pliki do przygotowania raportu.

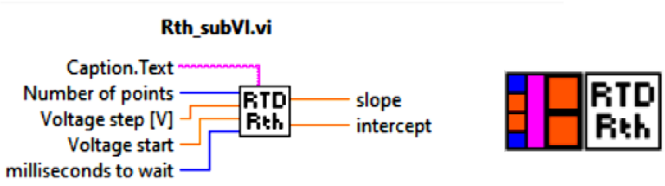
Robimy kopię programu do realizacji następnego zadania, zmieniając jego numer na kolejny, czyli 8-3-a.

### 8.5.3. Trzeci program do automatycznego badania efektu wewnętrznego i zewnętrznego

Opracowany w poprzednim punkcie program automatycznie wykonuje wszystkie pomiary i obliczenia niezbędne do wyznaczenia rezystancji termicznej czujnika. Jednak rezystancja termiczna czujnika wynika z występowania efektu wewnętrznego i zewnętrznego. Przy szybkich zmianach prądu dominuje efekt wewnętrzny, natomiast przy powolnych istotny staje się efekt zewnętrzny. Nie wiemy jednak, jakie zmiany możemy uznać za szybkie, a jakie za wolne. Aby zatem dokładnie poznać parametry czujnika, wykonujemy pomiary rezystancji termicznej, zmieniając czas opóźnienia w szerokich granicach. Jest to jednak zadanie dość żmudne i uciążliwe. W tym punkcie przygotujemy program, który w pełni automatyzuje takie badania.

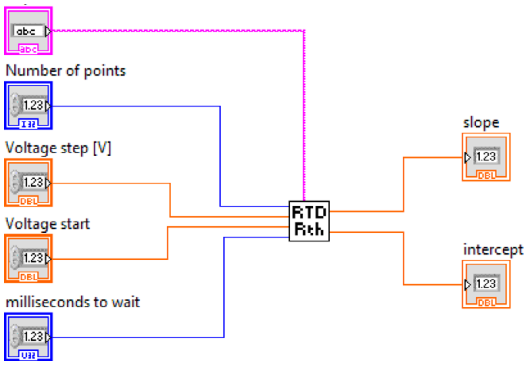
W pierwszej kolejności przekształcamy program opracowany w poprzednim punkcie w podprogram SubVI. W tym celu nadajemy programowi krótką i sugestywną nazwę, edytujemy ikonę, wybieramy odpowiedni wzór terminali na panelu złącza podprogramu oraz dokonujemy odpowiednich połączeń terminali wejściowych i wyjściowych z zadajnikami i wskaźnikami na panelu. Szczegółowy sposób postępowania został opisany w instrukcji do ćwiczenia 7. Przykładowy rezultat tworzenia podprogramu do pomiaru rezystancji termicznej pokazano na rycinie 8.17. Otwieramy kopię programu zapisaną w poprzednim punkcie, zawierającą w nazwie pliku numer 8-3-a. Upewniamy się, że program działa poprawnie. Zapisujemy program pod nową, krótką i komunikatywną nazwą, np. Rth_SubVI. Edytujemy ikonę programu, nadając jej indywidualny wygląd. Wybieramy wzór zacisków wejściowych i wyjściowych.

Potrzebujemy wzoru z pięcioma wejściami i dwoma wyjściami. Wybieramy wzór z czterema wejściami po lewej stronie, jednym wejściem na górze i dwoma wyjściami po prawej stronie. Zaciski wejściowe łączymy kolejno na panelu z zadajnikami: **Caption.Text**, **Number of points**, **Voltage step**, **Voltage start** oraz **milliseconds to wait**. Zaciski wyjściowe łączymy ze wskaźnikami **slope** i **intercept**. Przykładowe rozmieszenie zacisków oraz ikonę podprogramu pokazano na rycinie 8.17. Po wykonaniu wszystkich opisanych czynności zapisujemy podprogram do pliku.




**Ryc. 8.17.** Rozmieszenie zacisków oraz ikona podprogramu Rth_SubVI

Teraz utworzymy prosty program nadrzędny z wykorzystaniem opracowanego podprogramu. W menu głównym LabVIEW wybieramy opcję **File > NewVI**. Pojawia się nowe, puste okno do tworzenia programu. Przełączamy się (**Ctrl+E**) na diagram. Klikając prawym przyciskiem myszy na pustym diagramie, wybieramy opcję **Functions > Select a VI...**. Wyszukujemy po nazwie plik z utworzonym subVI i wstawiamy go na diagram. Otwieramy okno **Context Help** i sprawdzamy, czy pojawia się w nim odpowiednia informacja. Widoczna jest tam nazwa subVI, opisy terminali wejściowych i wyjściowych oraz opis działania subVI, co przedstawiono na rycinie 8.17.




**Ryc. 8.18.** Diagram programu nadrzędnego z wykorzystaniem podprogramu do wyznaczania rezystancji termicznej

Opisy terminali są zgodne z nazwami odpowiednich kontrolek na panelu subVI. Upewniamy się, że wejścia oraz wyjścia znajdują się w odpowiednich miejscach na rysunku ikony. Klikając prawym przyciskiem myszy kolejno na wszystkich terminalach wejściowych, tworzymy zadajniki (**Create Control**). Otrzymują one automatycznie odpowiednie nazwy. Analogicznie, klikając prawym przyciskiem myszy na dwóch terminalach wyjściowych, tworzymy wskaźniki (**Create Indicator**). Program jest gotowy. Przykładowy wygląd diagramu pokazano na rycinie 8.18. Zapisujemy program na dysku, umieszczając w nazwie pliku kolejny numer programu, czyli 8-3-b.

Na panelu programu ustawiamy następujące parametry: 10 punktów charakterystyki, krok napięcia 0,5 V, napięcie początkowe 0,5 V oraz opóźnienie czasowe 1000 ms. Wprowadzamy do opisu wykresu informacje o konfiguracji programu, np. Pt1000, 10 points, 0,5 V start, 0,5 V step. Klikamy dwukrotnie na ikonę podprogramu. Otwiera się wówczas okienko z panelem podprogramu. Ustawiamy na ekranie obok siebie okienko programu nadrzędnego i podprogramu.

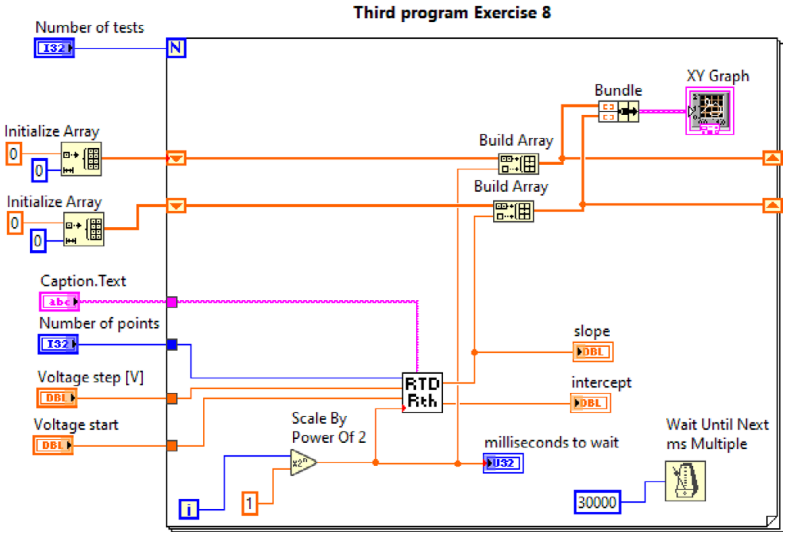
Uruchamiamy program nadrzędny i obserwujemy okienko podprogramu. Test wykonuje się w czasie 10 sekund. Sprawdzamy, czy w podprogramie pojawiły się wartości ustawione w programie nadrzędnym. Czy podprogram zapisał plik z wykresem? Czy zapisany do pliku wykres posiada opis zgodny z wprowadzonym do programu nadrzędnego?

Jeśli program działa prawidłowo i wygenerował poprawny wykres, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne powiązania ze sobą. Na koniec zapisujemy program do pliku.

Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu i nazwy plików z opisem zawartości zapisujemy w protokole. Wykorzystujemy zapisane pliki do przygotowania sprawozdania.

Robimy kopię programu do realizacji następnego, ostatniego już zadania, zmieniając jego numer na kolejny, czyli 8-3-c.

Teraz przygotowujemy program, który stanowi podsumowanie umiejętności zdobytych podczas realizacji całego laboratorium. Przykładowy diagram i panel tego programu pokazano na rycinach 8.19 i 8.20. Jest to na pozór bardzo prosty program, jednak w rzeczywistości jego działanie jest dość skomplikowane dzięki wywoływaniu podprogramu w pętli **For** ze zmianą parametru wywołania w każdym jej obiegu.



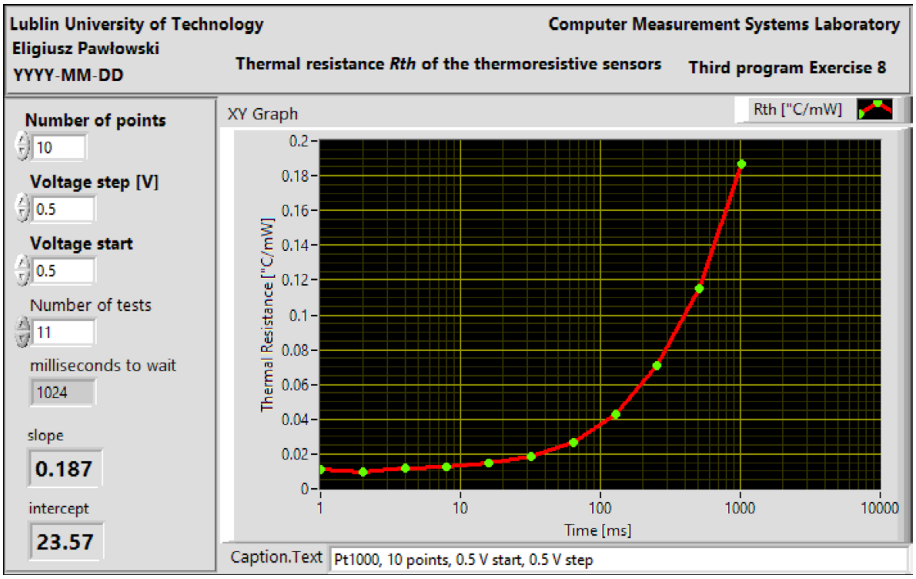

**Ryc. 8.19.** Diagram końcowej wersji ostatniego programu

Modyfikujemy zapisaną kopię programu 8-3-c. Wstawiamy do diagramu pętlę **For**. Na zacisku **N** pętli tworzymy zadajnik, nadajemy mu nazwę **Number of tests** i ustawiamy wartość początkową na 11. Zadajnik **milliseconds to wait** umieszczamy wewnątrz pętli i zamieniamy na wskaźnik (**Change to Indicator**). Pozostałe zadajniki pozostawiamy na zewnątrz pętli **For**. Opóźnienie czasowe wyliczamy na podstawie licznika pętli **For** za pomocą funkcji **Scale By Power Of 2**, co przedstawiono na rycinie 8.19. W ten sposób w każdym obiegu pętli **For** opóźnienie czasowe odpowiada w milisekundach kolejnym potęgom liczby 2, czyli 1 ms, 2 ms, 4 ms, 8 ms itd. To opóźnienie wprowadzamy w podprogramie pomiędzy kolejnymi pomiarami przy wyznaczaniu charakterystyki prądowo-napięciowej badanego czujnika. Po każdym wywołaniu podprogramu czujnik nagrzewa się do temperatury wyższej od otoczenia. Przed rozpoczęciem kolejnej serii pomiarów czujnik musi wystygnąć. Dlatego wstawiamy do pętli **For** drugie opóźnienie za pomocą funkcji **Wait**

**Until Next ms Multiple**. Na jej wejściu tworzymy wartość stałą i wstępnie – na potrzeby testów programu – nadajemy jej wartość 1000 ms.

Wyniki kolejnych pomiarów rezystancji termicznej w funkcji czasu opóźnienia prezentujemy na wykresie typu **XY Graph**. W sposób, który poznaliśmy we wcześniejszych ćwiczeniach, budujemy dwie tablice (**Build Array**) z danymi dla osi X oraz Y, korzystając z rejestrów przesuwnych. Na rejestrach wejściowych tworzymy puste tablice za pomocą funkcji **Initialize Array**. We właściwościach wykresu ustawiamy skalę X jako logarytmiczną.

Przechodzimy na panel i sprawdzamy wszystkie nastawy. Dla 11 testów opóźnienie w podprogramie zmienia się od 1 ms do 1024 ms. Każdy test składa się z 10 pomiarów, czyli trwa od 10 ms do 10,24 s. Pomiędzy poszczególnymi testami wstawiamy sekundę przerwy na ostygnięcie czujnika. Sumaryczny czas trwania tak skonfigurowanych testów wynosi około 30 sekund. W tym czasie podprogram zapisuje 11 wykresów oraz plików tekstowych z wynikami pomiarów.




**Ryc. 8.20.** Panel końcowej wersji programu, wykres dla czasów od 1 ms do 1024 ms, czujnik umieszczony w nieruchomym powietrzu

Ustawiamy na ekranie obok siebie okienko programu nadrzędnego i podprogramu. Uruchamiamy program nadrzędny i obserwujemy okienko podprogramu. Program nadrzędny wywołuje podprogram 11 razy. Czas opóźnienia

zmienia się od 1 ms do 1024 ms. Sprawdzamy, czy w podprogramie pojawiły się takie wartości. Czy podprogram zapisał 11 plików z wykresami? Czy zapisane do pliku wykresy posiadają opisy zgodne z informacjami wprowadzonymi do programu nadrzędnego? Zwracamy uwagę na wykres z wynikami dla opóźnienia 1024 ms. Widać na nim, że na początku czujnik, zamiast się nagrzewać, wciąż stygnie po poprzednim pomiarze. Dlatego docelowe badania wykonujemy, zadając dłuższą przerwę pomiędzy kolejnymi testami. W tym celu zmieniamy wartość opóźnienia na wejściu funkcji **Wait Until Next ms Multiple** na 30000 ms, czyli 30 sekund.

Jeśli program działa prawidłowo i wygenerował poprawny wykres, deklarujemy aktualne nastawy jako domyślne za pomocą polecenia **Edit > Make Current Values Default** i zapisujemy program do pliku. Uaktualniamy odpowiednio panel i diagram oraz uzupełniamy je niezbędnymi informacjami. Porządkujemy na panelu rozmieszczenie kontrolek i wskaźników, uwzględniając ich logiczne powiązania ze sobą. Przykład tak uzupełnionego panelu pokazano na rycinie 8.20. Na koniec zapisujemy program do pliku.

Poleceniem **File > Print > VI Documentation** zapisujemy do pliku dokumentację programu zawierającą panel i diagram. Znajdujemy zapisane pliki na dysku i sprawdzamy ich zawartość. Nazwę utworzonego katalogu oraz nazwy plików z opisem zawartości zapisujemy w protokole. Wykorzystujemy zapisane pliki do przygotowania raportu.

## 8.6. Badania

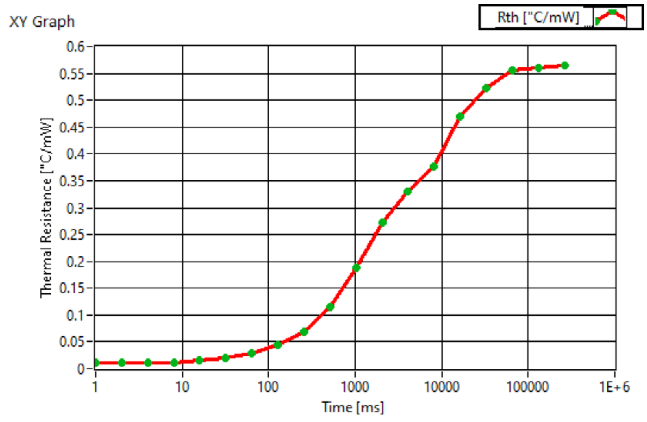
Zjawiska cieplne zachodzą stosunkowo wolno. Dlatego badania tego rodzaju zjawisk są z natury czasochłonne i pracochłonne. Jednak dzięki opracowanemu programowi redukujemy tę pracochłonność do minimum. Niestety nie możemy przyspieszyć samych pomiarów. Dlatego przed uruchomieniem programu rozsądnie planujemy parametry badania, biorąc pod uwagę czas do końca zajęć.

Na panelu programu ustawiamy następujące parametry: 10 punktów charakterystyki, krok napięcia 0,5 V oraz napięcie początkowe 0,5 V. Do opisu wykresu wprowadzamy informacje o konfiguracji programu, np. Pt1000, 10 points, 0,5 V start, 0,5 V step. Na diagramie sprawdzamy wartość opóźnienia na wejściu funkcji **Wait Until Next ms Multiple** – wynosi ona 30000 ms (30 sekund). Na

ekranie ustawiamy obok siebie okienko programu nadrzędnego i podprogramu. Kluczowym zagadnieniem jest dobranie parametru **Number of tests**. Po ostatnim testowym uruchomieniu programu liczbę testów ustawiamy na 11. Biorąc pod uwagę wszystkie opóźnienia czasowe, taki test będzie teraz trwał około 6 minut. Przykładowy wykres dla takich nastaw w programie przedstawiono na rycinie 8.20. Zależnie od dostępnego czasu do końca zajęć w laboratorium planujemy odpowiednią większą lub mniejszą liczbę testów. Pamiętamy, że czas testów zwiększa się nieliniowo, a skala na osi czasu jest logarytmiczna.

Uruchamiamy program i czekamy cierpliwie na jego zakończenie. Na panelu podprogramu kontrolujemy przebieg badania. Program nadrzędny nie zapisuje automatycznie wykresu. Po zakończeniu pomiarów zapisujemy otrzymany wykres do pliku BMP za pomocą opcji **Export Simplified Image**. Nazwę pliku zapisujemy w protokole i wykorzystujemy do przygotowania raportu. Sprawdzamy pliki zapisane przez podprogram. Znajdują się tam pliki graficzne z wykresami i tekstowe z wynikami pomiarów. Porównujemy je z plikami zapisanymi wcześniej, gdy opóźnienie pomiędzy testami wynosiło 1000 ms. Czy widać różnicę w kształcie wykresów i w wartościach wyników pomiarów? Zrób takie porównanie w raporcie.

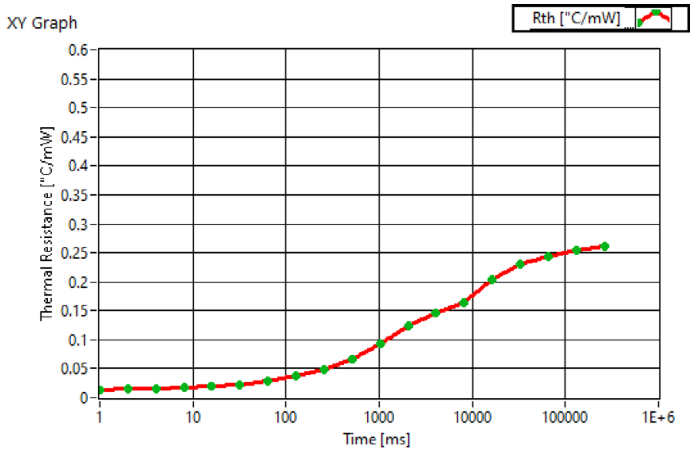
Na rycinie 8.21 pokazano przykładowy wykres otrzymany dla 19 testów. Opóźnienie zmieniało się od 1 ms do 262 s. Łączny czas trwania testów wynosił 96 minut, czyli ponad półtorej godziny. Zauważamy, że dla czasów poniżej 1024 ms wykres jest zgodny z ryciną 8.20.




**Ryc. 8.21.** Wykres rezystancji termicznej w funkcji czasu opóźnienia w zakresie od 1 ms do 262,144 sekundy, czujnik w nieruchomym powietrzu

Jak pokazano na rycinie 8.7, termiczne właściwości czujnika opisuje jego rezystancja termiczna wewnętrzna i zewnętrzna. Analizując otrzymany wykres, dla czasów krótszych niż 10 ms wyznaczamy rezystancję termiczną wewnętrzną $R_{\text{th j-c}} = 0{,}015°\text{C/mW}$, natomiast dla czasów dłuższych od 100 s – rezystancję termiczną zewnętrzną $R_{\text{th c-a}} = 0{,}56°\text{C/mW}$. W części teoretycznej było wyjaśnione, że rezystancja termiczna wewnętrzna wynika z konstrukcji czujnika i nie zależy od warunków otoczenia. Z kolei rezystancja termiczna zewnętrzna zależy od sposobu montażu czujnika i parametrów środowiska, w którym się on znajduje.

Aby potwierdzić te teoretyczne zależności, możemy powtórzyć pomiary dla czujnika zainstalowanego na module używanym w ćwiczeniu 7. Znajduje się tam cienkowarstwowy czujnik platynowy Pt1000 tego samego typu, jednak sposób jego montażu jest całkowicie inny. W ćwiczeniu 8 zbadaliśmy czujnik umieszczony w nieruchomym powietrzu. Natomiast w ćwiczeniu 7 taki sam czujnik znajduje się wewnątrz ceramicznej grzałki, w metalowej rurce ze stali nierdzewnej wypełnionej silikonową pastą termoprzewodzącą. Zatem przekazywanie ciepła od czujnika do obudowy jest znacznie lepsze, co oznacza mniejszą rezystancję termiczną zewnętrzną. Z kolei rezystancja termiczna wewnętrzna pozostaje taka sama.




**Ryc. 8.22.** Wykres rezystancji termicznej w funkcji czasu opóźnienia, pomiary dla czujnika zainstalowanego w module do ćwiczenia 7

Na rycinie 8.22 przedstawiono wyniki testu dla czujnika używanego w ćwiczeniu 7. Parametry ustawione w programie były takie same jak dla badania przedstawionego na rycinie 8.21. Dla ułatwienia porównania na obu wykresach zastosowano takie same zakresy skal. Widzimy wyraźnie, że wyniki testów znacznie się różnią. Przy krótkich czasach oba wykresy mają zbliżone wartości, co potwierdza, że w obu przypadkach rezystancja termiczna wewnętrzna $R_{\text{th j-c}}$ ma zbliżoną wartość. Z kolei przy długich czasach różnica staje się istotna. Rezystancja termiczna zewnętrzna $R_{\text{th c-a}}$ dla czujnika umieszczonego w powietrzu jest około dwa razy większa niż dla czujnika umieszczonego w metalowej osłonie wypełnionej silikonową pastą termoprzewodzącą. Wyniki pomiarów potwierdzają zatem teorię.

## 8.7. Opracowanie raportu z realizacji ćwiczenia

Raport z przeprowadzonego ćwiczenia zawiera:

- stronę tytułową z danymi zespołu, datą oraz tytułem ćwiczenia,
- opis środowiska pomiarowego,
- schemat wykorzystanego układu pomiarowego,
- tematy kolejnych zrealizowanych zadań, zgodnie z programem ćwiczenia,
- opis każdego z wykonanych zadań (lub podanie przyczyny jego niewykonania),
- dokumentację trzech zrealizowanych i uruchomionych programów,
- wykresy wygenerowane przez program w środowisku LabVIEW,
- opracowane wyniki obserwacji i wnioski zanotowane w protokole,
- zauważone błędy w instrukcji i przykładowych programach,
- propozycje zmian i rozszerzeń programu ćwiczenia.
- W raporcie ocenie podlega:
- poprawność działania programów,
- czytelność i przejrzystość diagramu,
- estetyka panelu, dobór kolorów oraz elementów graficznych,
- funkcjonalność panelu pod kątem wygody obsługi,
- zakres zrealizowanych zadań,
- trafność wniosków końcowych.

# Bibliografia


[1] Świsulski D.: *Komputerowa technika pomiarowa. Oprogramowanie wirtualnych przyrządów pomiarowych w LabVIEW*. Wyd. PAK, Warszawa 2005.

[2] Tłaczała W.: *Środowisko LabVIEW w eksperymencie wspomaganym komputerowo*. WNT, Warszawa 2014.

[3] Chruściel M.: *LabVIEW w praktyce*. Wyd. BTC, Warszawa 2008.

[4] Kiczma B., Smuda M., Wacławek M., Ziembik Z.: *LabVIEW dla studentów*. Wyd. Uniwersytetu Opolskiego, Opole 2007.

[5] *User Guide NI USB-6001/6002/6003 Low-Cost DAQ USB Device*. National Instruments, 374259A-01, 2014. URL: https://docs-be.ni.com/bundle/usb-6000-6001-6002-6003-features/raw/resource/enus/374259a.pdf

[6] *Specification NI USB-6001 Low-Cost DAQ USB Device*. National Instruments, 374369A-01, May 2014. URL: https://docs-be.ni.com/bundle/usb-6001-specs/raw/resource/enus/374369a.pdf

[7] *Laminated transformers*. Pulse Engineering Inc., LT2007 (12/09). URL: www.pulseeng.com, LT2007_22706-2902848.pdf

[8] *P6KExxA, P6KExxCA Datasheet, 600 W TVS in DO-15*. DS0681 - Rev 9, STMicroelectronics, Dec 2024. URL: https://www.st.com/resource/en/datasheet/p6ke15ca.pdf

[9] *NI-DAQmx User Manual*. Updated: 2024-10-22. URL: https://www.ni.com/docs/en-US/bundle/ni-daqmx/page/user-manual-welcome.html

[10] *Create Simulated NI-DAQmx Devices in NI MAX*. Updated: Dec 28, 2023. URL: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000x0PxCAI

[11] *Considerations With NI-DAQmx Simulated Devices*. Updated: May 4, 2023. URL: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019Nw0SAE

[12] *LabVIEW 2023 Q1 Known Issues*. Updated: Mar 4, 2024. URL: https://www.ni.com/en/support/documentation/bugs/23/-LabVIEW-2023-q1-known-issues.html

[13] *DAQ Assistant Missing, Not Showing Up, or Not Initializing Correctly in LabVIEW*. Updated: Dec 28, 2023. URL: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019Ld1SAE

[14] PN-EN 60038:2012, *Napięcia znormalizowane CENELEC*.

[15] PN-EN 50160:2023-10, *Parametry napięcia zasilającego w publicznych sieciach elektroenergetycznych*.

[16] PN-EN 61000-4-30:2015-05, *Kompatybilność elektromagnetyczna (EMC) - Część 4-30: Metody badań i pomiarów - Metody pomiaru jakości energii*.

[17] *Rozporządzenie Ministra Klimatu i Środowiska z dnia 22 marca 2023 r. w sprawie szczegółowych warunków funkcjonowania systemu elektroenergetycznego*, Dz.U. 2023 poz. 819.

[18] Wiszniewski A.: *Algorytmy pomiarów cyfrowych w automatyce elektroenergetycznej*. WNT, Warszawa 1990.

[19] Szafran J., Wiszniewski A.: *Algorytmy pomiarowe i decyzyjne cyfrowej automatyki elektroenergetycznej*. WNT, Warszawa 2001.

[20] Pawłowski E.: Pomiary małych częstotliwości w obecności zakłóceń. *Konferencja Naukowo-Techniczna „Podstawowe Problemy Metrologii PPM'01"*, Ustroń 7-9 maja 2001, Prace Komisji Metrologii Oddziału PAN w Katowicach, seria: Konferencje, nr 3/2001, s. 281–290.

[21] Pawłowski E.: Power Grid Frequency Estimation Based on Zero Crossing Technique Using Least Squares Method to Approximate Sampled Voltage Signal Around Zero Level. W: Hanus R., Mazur D., Kreischer C. (eds), *Methods and Techniques of Signal Processing in Physical Measurements. MSM 2018*, Lecture Notes in Electrical Engineering, vol. 548. Springer, Cham 2019. DOI: 10.1007/978-3-030-11187-8_21.

[22] Pawłowski E.: Application of the Least Squares Method to the approximation of equally spaced samples in frequency measurement approach. *Przegląd Elektrotechniczny*, 2020, vol. 96, nr 8, s. 143–146. DOI: 10.15199/48.2020.08.29.

[23] Cooper A.J.: An automated approach to the Electric Network Frequency (ENF) criterion - Theory and practice. *The International Journal of Speech, Language and the Law*, vol. 16, nr 2, 2009. DOI: 10.1558/ijsll.v16i2.193.

[24] Huijbregtse M., Geradts Z.: Using the ENF Criterion for Determining the Time of Recording of Short Digital Audio Recordings. W: Geradts Z.J.M.H., Franke K.Y., Veenman C.J. (eds), *Computational Forensics. IWCF 2009*, Lecture Notes in Computer Science, vol. 5718. Springer, Berlin, Heidelberg 2009. DOI: 10.1007/978-3-642-03521-0_11.

[25] *SaRonix Datasheet, Crystal Clock Oscillator, NCT Series, 5V, True TTL*. Technical Data, DS-129, Rev. E.

[26] *Philips Semiconductors Product data sheet: 74HC4040; 74HCT4040, 12-stage binary ripple counter*. Rev. 03, 14 September 2005.

[27] Jasik J.R., Pawłowski E., Toborek K.: Komputerowo wspomagane pomiary częstotliwości sygnałów z wykorzystaniem programowalnego układu licznikowego. *XXVI Międzyuczelniana Konferencja Metrologów MKM ,94*, Opole-Jarnołtówek 1994, Zeszyty Naukowe WSInż. w Opolu nr 203, Elektryka tom 2, Opole 1994, s. 193–200.

[28] Pawłowski E.: Zastosowanie programowalnych układów licznikowych do wspomaganych komputerem pomiarów częstotliwości. *V cykl seminariów „Zastosowanie komputerów w dydaktyce ,95"*, Polskie Towarzystwo Elektrotechniki Teoretycznej i Stosowanej oddział w Gdańsku, Zeszyty Naukowe Politechniki Gdańskiej, Gdańsk 1995, s. 121–128.

[29] Pawłowski E.: Wspomagane komputerowo sprawdzanie generatorów częstotliwości wzorcowych. *V Szkoła-Konferencja „Metrologia Wspomagana Komputerowo MWK'2001"*, Rynia k. Warszawy 21–24 maja 2001, Wojskowa Akademia Techniczna, Instytut Podstaw Elektroniki, Wyd. SOWA Sp. z o.o., s. 67–72.

[30] Pawłowski E.: Komputerowy system do sprawdzania częstościomierzy cyfrowych. *VI Konferencja „Metrologia Wspomagana Komputerowo MWK'2003"*, Waplewo 26–29 maja 2003, Wyd. Instytut Podstaw Elektroniki WAT, Warszawa 2003, tom II, s. 79–84. ISBN 83-89399-00-8.

[31] Pawłowski E.: Porównywanie częstotliwości wzorcowych w środowisku LabVIEW. *Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej*, Gdańsk 2013, nr 34, s. 53–62.

[32] Pawłowski E., Warda P.: Method and system for disciplining a local reference oscillator by GPS 1PPS signal. *Przegląd Elektrotechniczny*, 2018, vol. 94, nr 7, s. 38–41.

[33] Dorozhovets M., Pawłowski E., Świsulski D.: Frequency measurement research with weight averaging of pulse output signal of voltage-to-frequency converter. *Measurement*, 2023, vol. 216, 112912. DOI: 10.1016/j.measurement.2023.112912.

[34] *TDA2050 32 W hi-fi audio power amplifier, Data Sheet*. Doc ID 1461 Rev. 3, STMicroelectronics 2011. URL: https://www.st.com/resource/en/datasheet/tda2050.pdf

[35] Ciszewski M.: Wzmacniacz audio z układem TDA2030, TDA2040 lub TDA2050. *Elektronika Praktyczna*, 2010, nr 11, s. 57.

[36] *Wzmacniacz audio z TDA20xx 14W, 22W lub 35W, Kit AVT 1597, dokumentacja techniczna*. AVT Sp. z o.o. URL: https://serwis.avt.pl/manuals/AVT1597.pdf

[37] *Photomicrosensor (Transmissive) EE-SX3009-P1/-SX4009-P1*. Cat. No. JB301-E3-01, 3/05, OMRON ELECTRONIC COMPONENTS LLC. URL: http://www.components.omron.com

[38] Pawłowski E.: Cyfrowe przetwarzanie sygnału przetwornika obrotowo-impulsowego. *VII Konferencja Naukowa „Czujniki Optoelektroniczne i Elektroniczne COE 2002"*, Rzeszów 5–8 czerwca 2002, Oficyna Wydawnicza Politechniki Rzeszowskiej, tom 1, s. 313–318. ISBN 83-7199-214-9.

[39] Świsulski D., Pawłowski E., Porzeziński P.: Metoda zliczania zależnego w pomiarach i rejestracji prędkości obrotowej. Część 1 – Zasada pomiaru. *Elektronizacja: Podzespoły i Zastosowania Elektroniki*, nr 11/1999, s. 14–17.

[40] Świsulski D., Pawłowski E., Porzeziński P.: Metoda zliczania zależnego w pomiarach i rejestracji prędkości obrotowej. Część 2 – Oprogramowanie układu obrotomierza. *Elektronizacja: Podzespoły i Zastosowania Elektroniki*, nr 12/1999, s. 5–7.

[41] Pawłowski E.: Reconstruction of input signal of sensor with frequency output. *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, Międzyzdroje 2015, s. 909–914. DOI: 10.1109/MMAR.2015.7283998.

[42] Otomański P., Pawlowski E., Szlachta A.: Application of LabVIEW to Determine Characteristics of Two-Terminal Passive Components. *2023 14th International Conference on Measurement*, Smolenice, Slovakia 2023, s. 305–308. DOI: 10.23919/MEASUREMENT59122.2023.10164491.

[43] International Standard IEC 60751:2022. *Industrial Platinum Resistance Thermometers and Platinum Temperature Sensors*. Edition 3.0. International Electrotechnical Commission (IEC), Geneva, Switzerland 2022.

[44] King G., Fukushima T.: RTD Interfacing and Linearization Using an ADuC8xx MicroConverter. *AN-709 Analog Devices Application Note*, 2004, s. 1–12.

[45] ASTM Standard E1137/E1137M-08. *Standard Specification for Industrial Platinum Resistance Thermometers*. ASTM International Committee E20, West Conshohocken, PA, USA 2020.

[46] Pawłowski E.: Badania in situ rezystancji termicznej rezystancyjnych czujników temperatury. *Przegląd Elektrotechniczny*, 2024, vol. 100, nr 12, s. 260–263.

[47] Otomański P., Pawłowski E., Szlachta A.: In Situ Evaluation of the Self-Heating Effect in Resistance Temperature Sensors. *Sensors*, 2025, vol. 25, 3374.

[48] Pawłowski E., Szlachta A., Otomański P.: The Influence of Noise Level on the Value of Uncertainty in a Measurement System Containing an Analog-to-Digital Converter. *Energies*, 2023, vol. 16, 1060.

[49] Pawłowski E.: Single Sensor Hot-Wire Anemometer Based on Thermal Time Constant Estimation. *2018 XV International Scientific Conference on Optoelectronic and Electronic Sensors (COE)*, Warsaw, Poland 2018, s. 1–4. DOI: 10.1109/COE.2018.8435154.

[50] Pawłowski E.: Thermal heat-balance mode Flow-to-Frequency Converter. *Proc. SPIE 10161, 14th International Conference on Optical and Electronic Sensors*, 101610W, 2016. DOI: 10.1117/12.2243598.