

Laboratorium Komputerowych Systemów Pomiarowych

ĆWICZENIE NR 3

ROZPROSZONY SYSTEM POMIAROWY Z INTERFEJSEM RS-485

(opracował Eligiusz Pawłowski)

Cel i zakres ćwiczenia

Celem ćwiczenia jest praktyczne zapoznanie się studentów z problematyką programowania w środowisku LabVIEW rozproszonego systemu pomiarowego złożonego z przetworników ADAM-4000 wyposażonych w interfejs szeregowy RS-485.

Program laboratorium obejmuje następujące zagadnienia:

- zapoznanie się ze środowiskiem LabVIEW i podstawami programowania,
- umieszczanie kontrolki na panelu programu i tworzenie interfejsu użytkownika,
- tworzenie diagramu sterującego przebiegiem programu,
- zapoznanie się z systemem przetworników ADAM-4000 i interfejsem RS-485,
- programowanie przetworników ADAM-4000 poprzez interfejs RS-485,
- sterowanie grzałką i wentylatorem za pomocą modułu ADAM-4060 oraz pomiar temperatury czujnikiem termorezystancyjnym Pt100 za pomocą modułu ADAM-4013 w modelu procesu technologicznego,
- programowanie algorytmów pomiarowych, sterowania i regulacji automatycznej w środowisku LabView.

1. Opis stanowiska laboratoryjnego

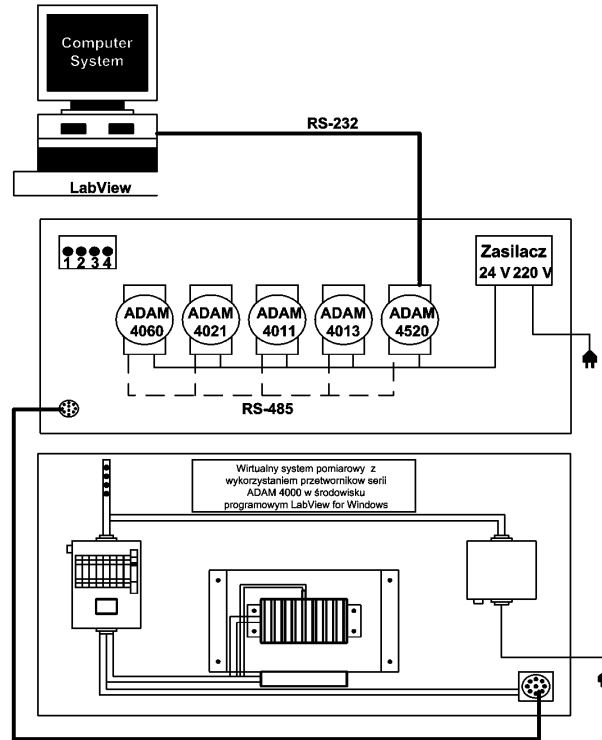
Wykorzystywane w ćwiczeniu stanowisko laboratoryjne składa się z następujących elementów składowych:

- modelu procesu technologicznego w którym zachodzą procesy grzania i stygnięcia,
- sieci przetworników serii ADAM-4000 umożliwiających pomiary i sterowanie,
- konwertera interfejsów RS-485 / RS-232 umożliwiającego sterowanie przetwornikami ADAM-4000 z komputera PC wyposażonego w interfejs szeregowy RS-232,
- komputera PC z zainstalowanym oprogramowaniem LabVIEW.

Wszystkie elementy składowe są ze sobą połączone i gotowe do realizacji ćwiczenia. Studenci nie wykonują samodzielnie żadnych połączeń w układach pomiarowych i sterujących. Zadania realizowane podczas ćwiczenia polegają na samodzielnym przygotowaniu odpowiednich programów w środowisku LabVIEW, uruchomieniu ich, przetestowaniu poprawności działania, usunięciu błędów i wykonaniu przykładowych pomiarów. Wygląd stanowiska laboratoryjnego przedstawia rysunek 1.

Stanowisko zrealizowane jest w postaci dwóch tablic zawieszonych na ścianie, połączonych ze sobą wielożyłowym kablem sterującym oraz dołączonych do komputera klasy PC za pomocą interfejsu RS-232C. Tablica z modelem procesu technologicznego zawiera układ grzania i chłodzenia obiektu pomiaru składającego się z dwóch radiatorów umieszczonych w przezroczystej osłonie, układ zasilania z odpowiednimi zabezpieczeniami, czujnik temperatury Pt100, diody sygnalizacyjne i złącze wielostykowe do połączenia z tablicą przetworników ADAM-4000. Na tablicy z przetwornikami umieszczono cztery przetworniki z serii ADAM (4060, 4021, 4011, 4013) oraz konwerter interfejsów RS-485 / RS-232C typu ADAM-4520. Wszystkie przetworniki ADAM połączone są ze sobą w sieć interfejsem RS-485 za pomocą dwuprzewodowej skrętki, natomiast konwerter 4520 umożliwia sterowanie przetwornikami za pomocą komputera klasy PC wyposażonego w

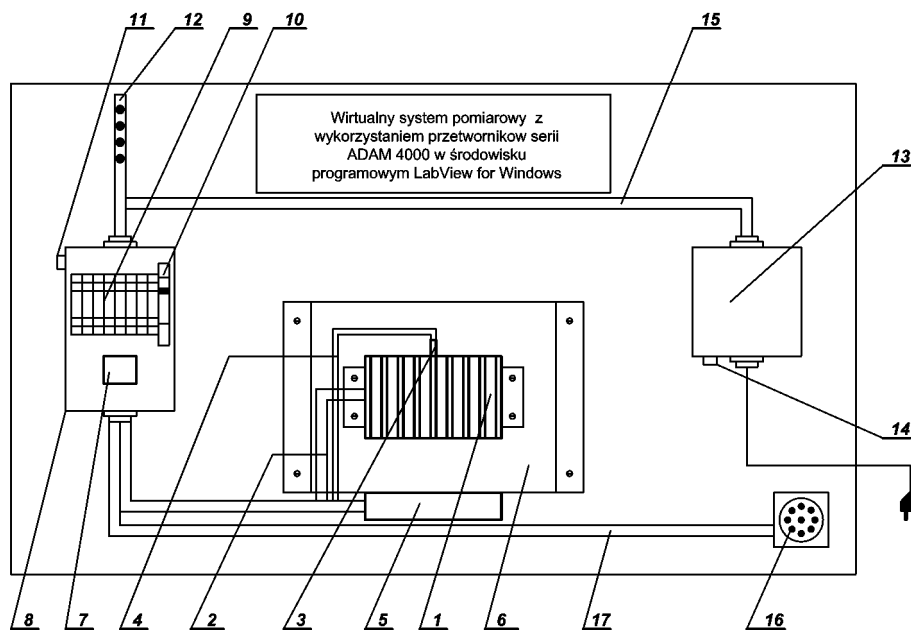
interfejs RS-232C. W ćwiczeniu wykorzystane będą tylko przetworniki: ADAM-4060 i ADAM-4013. Pozostałe dwa dostępne przetworniki: ADAM-4021 i ADAM-4011 studenci mogą wykorzystać realizując dodatkowe zadania lub na potrzeby pracy dyplomowej. Każda tablica posiada niezależny zasilacz sieciowy. Model procesu technologicznego zasilany jest napięciem 12V, a sieć RS-485 z przetwornikami ADAM-4000 zasilana jest napięciem 24V.



Rys.1. Rysunek poglądowy kompletnego stanowiska laboratoryjnego

1.1. Model procesu technologicznego

Model procesu technologicznego umieszczony jest na tablicy powieszonyj na ścianie. Model symuluje proces technologiczny, w którym zachodzą naprzemiennie etapy grzania i chłodzenia. Wygląd zewnętrzny tablicy przedstawiono na rysunku 2.



Rys.2. Rysunek poglądowy tablicy z modelem procesu technologicznego

Tablica z obiektem na którym realizowane są pomiary i sterowanie składa się z: układu grzania, układu chłodzenia (wentylacji) (5), czujnika do pomiaru temperatury Pt100 (3), układu sygnalizacji (12) oraz układu zasilania (13) i rozdziału energii (8). Układ grzania stanowią połączone ze sobą dziesięć rezystorów o mocy 8 W i rezystancji 10Ω każdy. Rezystory połączone są szeregowo–równolegle i umieszczone pomiędzy płaskimi powierzchniami dwóch radiatorów typu A 5723 L7 (1). Wypadkowa rezystancja układu grzania wynosi 4Ω , sumaryczna moc wynosi 80W, maksymalny prąd zasilający wynosi 4,47A. Radiatory zapewniają odprowadzanie ciepła wytworzonego przez rezystory na skutek przepływu prądu. Napięcie zasilające rezystory ($\sim 12V$) jest doprowadzane przewodami miedzianymi (2) za pomocą przełącznika R15–12V (7). Prąd zasilający rezystory $I = 3A$ jest mniejszy od prądu wynikającego z dopuszczalnej mocy rezystorów.

Ciepło wydzielające się w układzie grzania zwiększa temperaturę radiatorów. Temperatura radiatorów jest mierzona za pomocą czujnika temperatury (3) Pt100, umieszczonego w górnej części układu grzania. Czujnik posiada długi przewód, co umożliwia pomiar temperatury w dowolnym miejscu układu grzania. Przewody zasilające czujnik (4) są połączone bezpośrednio z przetwornikiem ADAM 4013.

Układ wentylacji stanowi wentylator typu PL 80S12H (5) z silnikiem 12V=. Zasilany jest bezpośrednio z przetwornika ADAM 4060 i podłączony jest pod styk RL2. Wentylator jest przymocowany do obudowy wykonanej z tworzywa sztucznego pleksiglas (6). Obudowa pełni rolę dekoracyjną oraz zapewnia bezpieczeństwo pracy. Dla ułatwienia serwisu, obudowa zamocowana jest tak, że jej demontaż umożliwia pracę układów z jednoczesną kontrolą i wizualizacją.

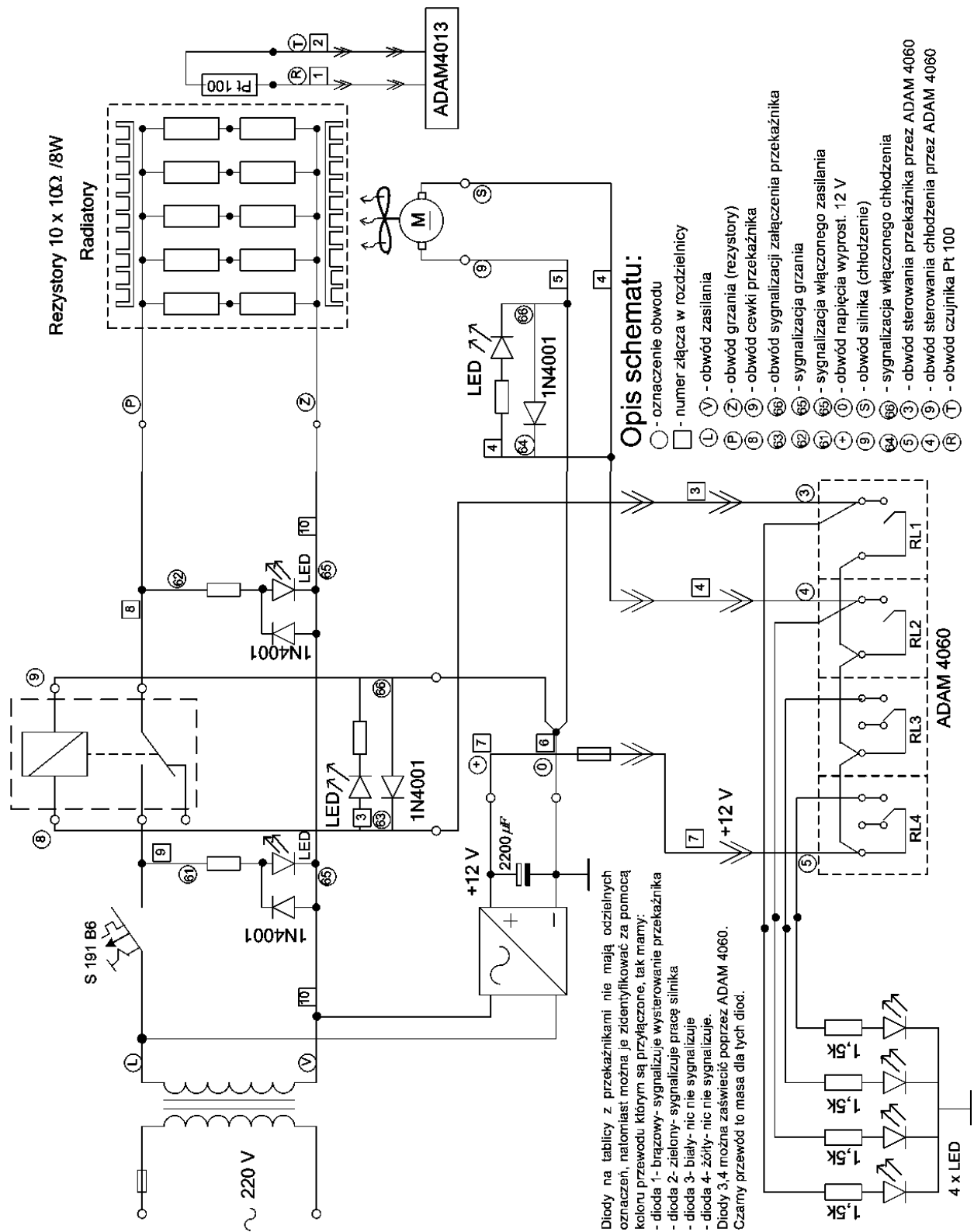
Układ rozdziału energii stanowi rozdzielnica typu N/T z szyną EP-LUX (8). W rozdzielnicy znajdują się, również umieszczone na szynie DIN, złączki typu ZUG 4MM/301 (9), do których są doprowadzone przewody, a także przełącznik R15–12V (7). Na szynie DIN umieszczony jest dodatkowo bezpiecznik typu S191 / B6 $\sim 230V$ (10). Zabezpiecza on instalację elektryczną po stronie wtórnej transformatora zasilającego. Podobną rolę pełni bezpiecznik topikowy (11), zabezpieczając obwód sygnalizacji.

Układ sygnalizacji (12) stanowią cztery diody LED umieszczone w oprawkach LED 5mm. Poszczególne diody sygnalizują: zasilanie układu (Power), zasilanie grzałki, sterowanie grzałki, zasilanie wentylatora.

Układ zasilania służy do zamiany napięcia przemiennego 230V~ na napięcie stałe 12V=. Stanowi go transformator toroidalny typu TTH 220/12 V o mocy 105W (13). Transformator jest umieszczony w puszcze typu P6 4X10. Po stronie wysokiego napięcia transformator zabezpiecza bezpiecznik topikowy (14). Okablowanie znajduje się w korytkach instalacyjnych (15). Pełne sterowanie obiektem jest możliwe dzięki umieszczoneму na tablicy gnieździe (16), za pomocą, którego jest realizowane fizyczne połączenie przetworników i komputera z obiektem. W rurze instalacyjnej typu RL18 (17) znajdują się przewody łączące poszczególne elementy obiektu z przetwornikami. Opisy gniazd transmisyjnych znajdują się na obu tablicach. Do opisu zostały użyte oznaczenia zamieszczone w tabeli 1. Numery przewodów dotyczą przewodów znajdujących się w rurze instalacyjnej (17).

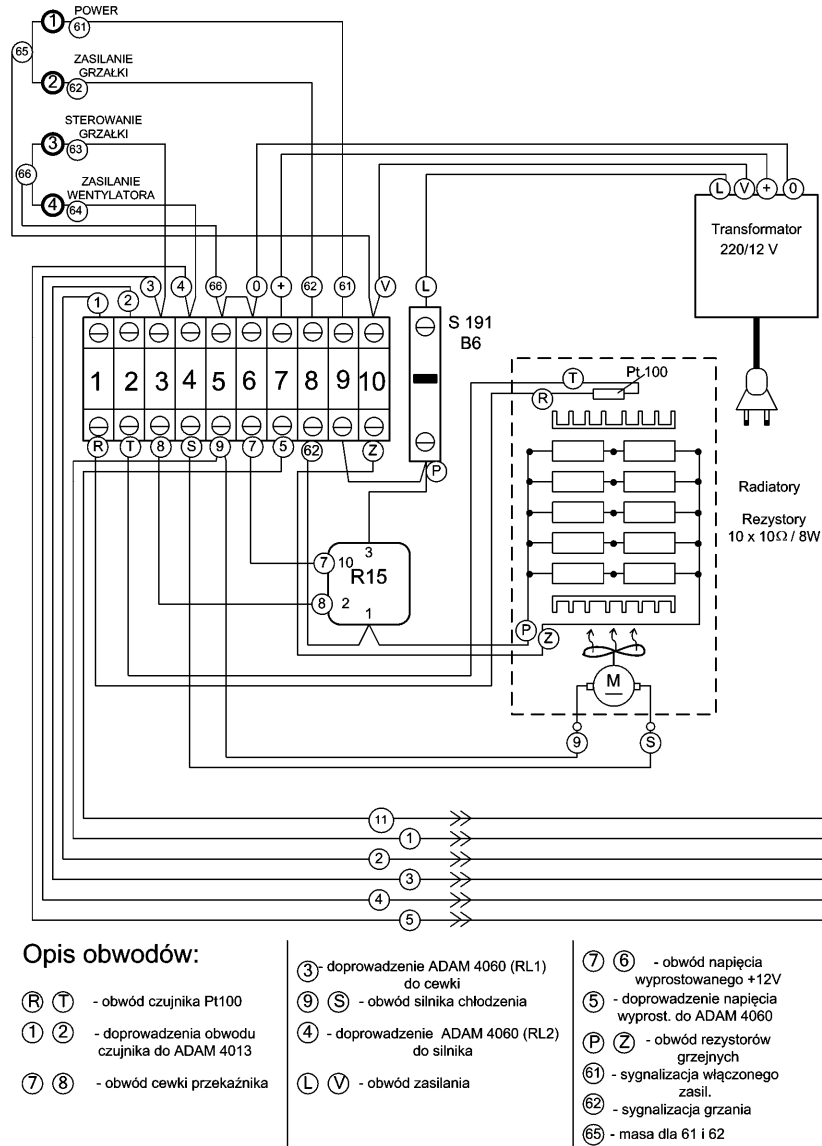
Na rysunku 3 przedstawiono schemat ideowy połączeń modelu procesu technologicznego z uwzględnieniem numerów styków gniazd połączeniowych (6 sztuk styków, tab. 1) oraz przetworników ADAM-4060 i ADAM-4013 umieszczonych na drugiej tablicy. Należy zwrócić uwagę, że cztery diody LED umieszczone na tablicy z przetwornikami ADAM sygnalizują bezpośrednio stan styków przełączników przetwornika ADAM-4060, a cztery diody LED umieszczone na tablicy z modelem procesu technologicznego sygnalizują: napięcie zasilające, napięcie na grzałce, napięcie sterujące na cewce przełącznika oraz napięcie na silniku wentylatora. Dzięki temu jest możliwa kontrola poprawności działania

całego układu oraz identyfikowanie ewentualnych uszkodzeń, gdy np. sygnalizowane jest zamknięcie styku przełącznika RL1 w module ADAM-4060, a brak jest sygnalizacji napięcia na grzałce. Analogicznie można analizować sterowanie silnikiem wentylatora.



Rys. 3. Schemat ideowy modelu procesu technologicznego

Na rysunku 4 przedstawiono schemat montażowy tablicy z modelem procesu technologicznego, na którym poszczególne elementy rozmieszczono zgodnie z ich położeniem w rzeczywistym układzie. Należy zwrócić uwagę na kolejność umieszczenia diod sygnalizacyjnych LED oraz na wyłącznik nadprądowy S191 B6, który może automatycznie wyłączyć zasilanie w przypadku wystąpienia przeciążenia lub zwarcia.



Rys. 4. Schemat montażowy modelu procesu technologicznego

Tab. 1. Opis gniazda transmisyjnego

Symbol	Znaczenie
1/SENSE+ / 4013	Przewód nr 1, wyjście modułu SENSE+, moduł 4013
2/SENSE- / 4013	Przewód nr 2, wyjście modułu SENSE-, moduł 4013
3/RL1 NO / 4060	Przewód nr 3, wyjście modułu RL1 NO, moduł 4060
4/RL2 NO / 4060	Przewód nr 4, wyjście modułu RL2 NO, moduł 4060
5/RL4 COM / 4060	Przewód nr 5, wyjście modułu RL4 COM, moduł 4060
11/MASA / LED(2,3,4)	Przewód nr 11, masa diod LED na tablicy z rys. 3.10

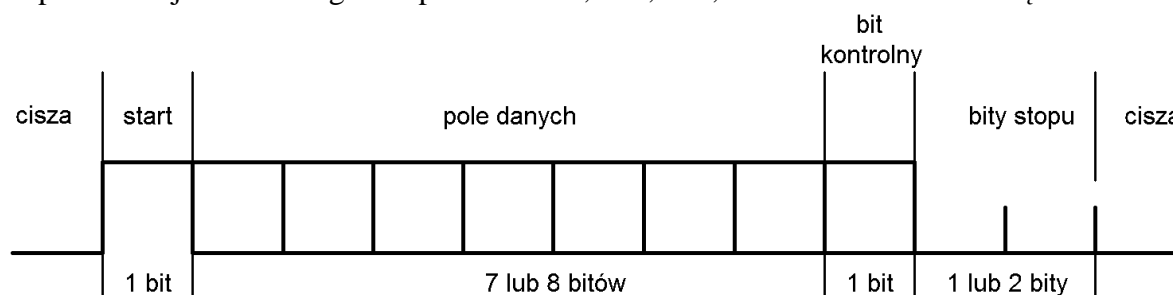
1.2. System interfejsu RS-232C

Standard RS-232 (*Recommended Standard*) został wprowadzony w 1962 roku przez *Electronic Industries Association* (EIA) w celu normalizacji interfejsu pomiędzy urządzeniem końcowym dla danych (DTE – Data Terminal Equipment) a urządzeniem komunikacyjnym dla danych DCE (*Data Communication Equipment*). Pierwotnie standard ten przewidziany był jako system interfejsu pomiędzy terminalem (DTE) a modemem (DCE), ale standard ten jest również bardzo często stosowany przy szeregowej transmisji danych pomiędzy różnymi typami urządzeń DTE. W sierpniu 1969 roku EIA wprowadziło zmodyfikowaną normę oznaczoną RS-232C, która opisuje powszechnie akceptowany sposób transmisji danych na nieduże odległości, z niewielką szybkością, przez niesymetryczne łącze.

W standardzie RS-232C transmisja danych odbywa się szeregowo bit po bicie, przy czym definiuje się dwa rodzaje transmisji: asynchroniczną transmisję znakową oraz transmisję synchroniczną.

Asynchroniczna transmisja znakowa polega na przesyłaniu pojedynczych znaków, które posiadają ściśle określony format. Początek znaku stanowi jeden bit startu, jałowy z punktu widzenia przesyłanej informacji i służący jedynie celom synchronizacyjnym. Dalej następuje pole danych, na które składają się kolejne bity stanowiące treść znaku (począwszy od bitu najmniej znaczącego LSB). Liczba bitów w polu danych może być różna, najczęściej przesyłanych jest 7 lub 8 bitów. Bezpośrednio za polem danych znajduje się bit kontrolny, służący zabezpieczeniu przed przekłamaniami informacji znajdującej się na polu danych. Jest to bit, który może ale nie musi wystąpić, przy czym stosuje się kontrolę parzystości lub nieparzystości. Decyzja o stosowaniu zabezpieczenia ma charakter globalny i dotyczy wszystkich znaków. Transmitowany znak kończy jeden lub dwa bity stopu. Zdefiniowany powyżej zespół bitów tworzy tzw. jednostkę informacyjną, której format przedstawiony jest na rys. 5. W ramach jednostki informacyjnej bity przesyłane są synchronicznie, to znaczy zgodnie z taktowaniem nadajnika. Natomiast poszczególne jednostki są przesyłane asynchronicznie – ich wyprowadzenie nie jest synchronizowane żadnym sygnałem, a więc odstęp pomiędzy nimi jest dowolny.

Czas trwania bitu w jednostce informacyjnej nazywa się odstępem jednostkowym i jest oznaczany przez t_b . Jego odwrotność ($f=1/t_b$) określa szybkość transmisji w bodach, czyli w bitach na sekundę (bps - bits per second). Szybkość jednego boda oznacza przesyłanie jednego bitu w ciągu jednej sekundy (1bd = 1bit/s = 1bps). Typowe wartości szybkości transmisji przy asynchronicznej transmisji znakowej wynoszą: 1200, 2400, 4800, 9600 bd, co przy założeniu 10-bitowej długości jednostki informacyjnej i przesyłaniu znaków bezpośrednio jeden za drugim odpowiada 120, 240, 480, 960 znakom na sekundę.



Rys. 5. Format jednostki informacyjnej w standardzie RS-232C

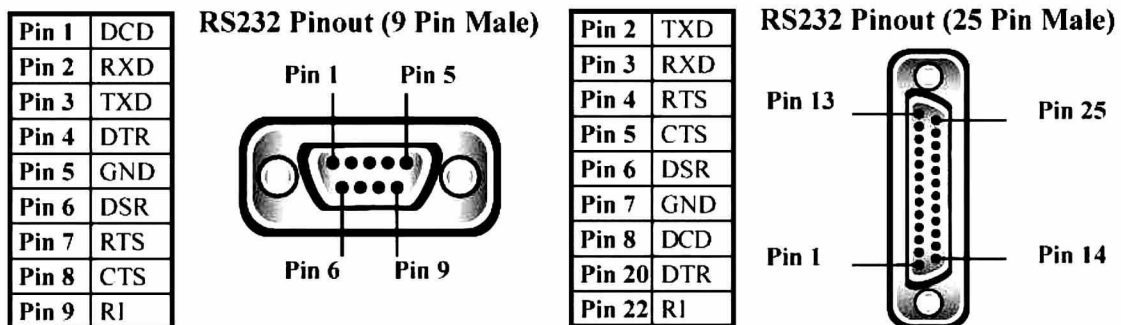
Przy asynchronicznej transmisji znakowej przyjmuje się, że zarówno odbiornik jak i nadajnik pracują z tą samą częstotliwością (szybkością transmisji w bodach), aczkolwiek takty nadawania i odbioru nie są zsynchronizowane. Ze względu na małą długość jednostki

informacyjnej, niewielka różnica częstotliwości generatorów taktu w nadajniku i odbiorniku nie powoduje błędnego odbioru znaków.

Bit kontrolny jest najczęściej tzw. bitem parzystości, którego stan określa się według jednej z dwóch zasad: kontrola parzystości lub kontrola nieparzystości. Kontrola parzystości polega na sprawdzeniu liczby jedynek na polu danych i ustawieniu bitu kontrolnego na „1”, w przypadku nieparzystej liczby jedynek, lub na „0” w przypadku parzystej liczby jedynek. Oznacza to, że przy kontroli parzystości, łączna liczba jedynek w polu danych i w bicie kontrolnym, jest zawsze parzysta. Podczas kontroli nieparzystości stosuje się regułę przeciwną: bit kontrolny ustawia się na jeden, w przypadku parzystej liczby jedynek na polu danych, lub na zero, jeżeli liczba jedynek jest nieparzysta. Bit kontroli parzystości pozwala wykryć fakt przekłamania znaku na polu danych, pod warunkiem, że liczba przekłamań jest nieparzysta.

Transmisja synchroniczna polega na przesyłaniu dużych bloków danych przy rezygnacji z bitów określających początek i koniec znaku. Poszczególne bity wyprowadzane są zgodnie z taktem nadawania. Po ostatnim bicie znaku poprzedniego wysyłany jest natychmiast pierwszy bit znaku następnego. Grupowanie bitów w znaki (bajty) po stronie odbiorczej umożliwia specjalny znak synchronizacyjny umieszczony na początku bloku. Blok kończy inny wyróżniony znak. Transmisja synchroniczna jest szybsza od asynchronicznej ze względu na brak „jałowych” bitów start i stop. Jednak jej realizacja jest trudniejsza i wymaga złożonych układów odbiorczych do poprawnego rozpoznawania bitów i grupowania bitów w znaki.

W interfejsie RS-232C stosowane są dwa rodzaje złączy: 25 – stykowe złącze szufladowe typu Cannon DB25 lub 9 – stykowe złącze szufladowe typu DB9. W złączu DB9 występują tylko najważniejsze sygnały przeznaczone dla asynchronicznej transmisji znakowej. W takie właśnie złącze najczęściej wyposażone są komputery klasy PC. Rozmieszczenie sygnałów na poszczególnych stykach złączy DB9 i DB25 przedstawiono na rys. 6.

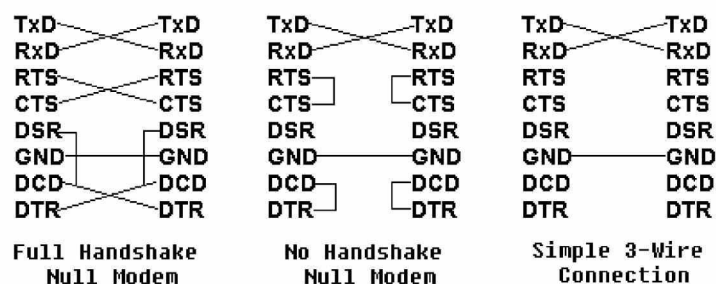


Rys. 6. Rozmieszczenie sygnałów w złączach interfejsu RS-232C

W każdym systemie transmisji RS-232C wykorzystywany jest co najmniej jeden (lub obydwa) z sygnałów: danych odbieranych (RXD) oraz danych nadawanych (TXD). Pozostałe sygnały są opcjonalne i służą do tzw. sprzętowej synchronizacji transmisji (hardware handshaking).

Aby w systemie interfejsu RS-232C mogła być zrealizowana skuteczna i bezbłędna transmisja należy zastosować odpowiedni kabel połączeniowy, tzn. wyjście danych nadawanych TXD urządzenia nadawczego musi być dołączone do wejścia danych odbieranych RXD urządzenia odbiorczego (i odwrotnie) oraz również analogicznie należy „skrzyżować” odpowiednie sygnały synchronizacji sprzętowej - jeśli są wykorzystywane. Jednym z popularnych połączeń tego rodzaju jest tzw. null-modem. W praktyce wykorzystywanych jest wiele różnych połączeń tego rodzaju, a producenci aparatury pomiarowej zazwyczaj szczegółowo opisują w dokumentacji wymagany sposób połączenia i

oferują odpowiedni kabel połączeniowy. Na rys. 7 przedstawiono przykładowe, jedne z częściej stosowanych, połączenia typu null-modem.



Rys. 7. Przykładowe połączenia typu null-modem interfejsu RS-232C

Poza połączeniem za pomocą odpowiedniego okablowania, niezbędne jest również zastosowanie w urządzeniach połączonych interfejsem RS-232C tych samych parametrów transmisji: szybkości transmisji w bodach, liczby bitów danych i liczby bitów stopu, kontrolę parzystości/nieparzystości lub jej brak oraz kontrolę sprzętową (hardware handshaking). W przeciwnym razie poprawna transmisja nie będzie możliwa.

1.3. System interfejsu RS-485

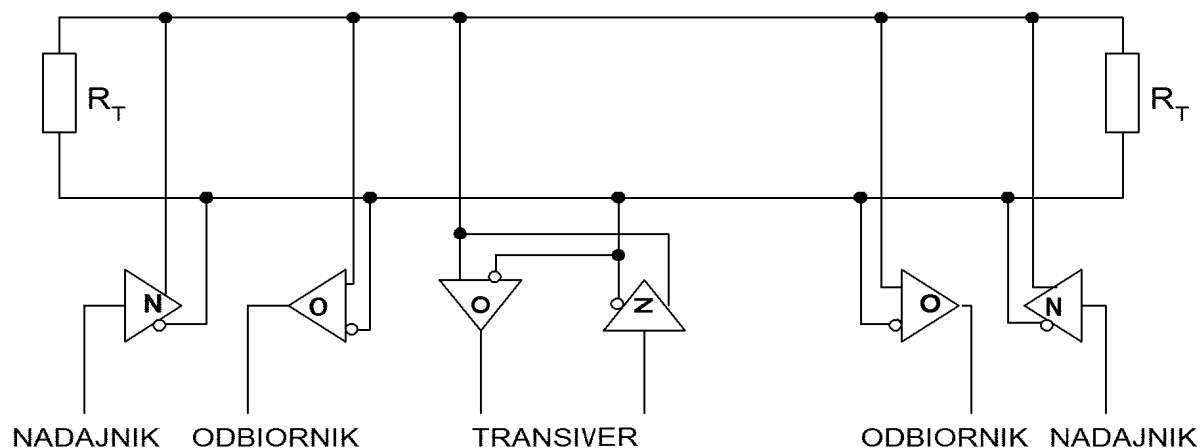
W interfejsie RS-232C stosuje się niesymetryczne linie sygnałowe. Niesymetryczna linia sygnałowa oznacza brak symetrii względem ziemi, tzn. dane przesyłane są przewodami, z których jeden ma stałe potencjał ziemi (styk GND), a drugi zmienia swój potencjał względem ziemi (sygnały RXD, TXD). Taki sposób transmisji jest stosunkowo prosty w realizacji, ale ogranicza szybkość i zasięg transmisji oraz zwiększa podatność na zakłócenia. Dlatego łącze RS-232C może być wykorzystywane jedynie na odległość do 15m (w oryginalnej specyfikacji: 50 stóp) oraz tylko w środowiskach o niskim poziomie zewnętrznych pól magnetycznych i elektrycznych. Z tych względów łącze RS-232C w praktyce nie jest przydatne do stosowania w warunkach przemysłowych.

W celu zwiększenia szybkości i zasięgu transmisji oraz dla ochrony przed zakłóceniami opracowano kolejne standardy transmisji szeregowej: RS-423A, RS-422A oraz RS-485, w których wprowadzono dwuprzewodowe, symetryczne obwody transmisji. Dodatkowo, dla ograniczenia wpływu zakłóceń, linie sygnałowe wykonuje się postaci tzw. skrętki, czyli dwóch przewodów ściśle ze sobą skręconych. Standard RS-423A określa elektryczną charakterystykę napięciowego obwodu transmisyjnego złożonego z niesymetrycznego nadajnika oraz symetrycznego odbiornika. Standardy RS-422A oraz RS-485 określają symetryczny, zrównoważony system transmisji danych złożony z różnicowego nadajnika, dwuprzewodowego zrównoważonego toru przesyłowego oraz odbiornika o różnicowym obwodzie wejściowym. Dzięki tym modyfikacjom zasięg transmisji zwiększył się do 1200 m oraz znacznie wzrosła odporność na zakłócenia. Porównanie parametrów standardów transmisji szeregowej przedstawiono w tabeli 1.

Standard RS-485 wprowadzono w 1983 roku jako rozwinięcie RS-422A. Łącze RS-485 jest symetryczne i zrównoważone, przy czym dopuszcza się w nim wiele odbiorników i wiele nadajników podłączonych jednocześnie do jednej linii. Nadajniki muszą być trójstanowe, ponieważ w danym przedziale czasu może nadawać tylko jeden z nich, a pozostałe muszą być wyłączone (powinny znajdować się w stanie wysokiej impedancji). Na rys. 8 przedstawiony jest wielopunktowy, zrównoważony interfejs cyfrowy zgodny ze standardem RS-485.

Dopasowanie toru transmisyjnego stanowią rezystory R_T , umieszczone na początku i na końcu linii.

Należy pamiętać, że wszystkie odmiany standardów transmisji szeregowej stosują ten sam format jednostki informacyjnej, przedstawionej na rysunku 5 i różnią się wyłącznie parametrami elektrycznymi linii transmisyjnej i systemem okablowania.



Rys. 8. Symetryczna linia transmisyjna w standardzie RS-485

Tab. 2. Porównanie standardów transmisji szeregowej

Parametr	RS-232C	RS-423A	RS-422A	RS-485
Rodzaj transmisji	Niesymetryczna	Niesymetryczna	Różnicowa	Różnicowa
Dozwolona ilość nadajników i odbiorników	1 nadajnik 1 odbiornik	1 nadajnik 10 odbiorników	1 nadajnik 10 odbiorników	32 nadajniki 32 odbiorniki
Maksymalna długość kabla [m]	15	1200	1200	1200
Maksymalna szybkość transmisji (bity/s)	20 k	100 k	10 M	10 M
Maksymalne napięcie wspólne	± 25 V	± 6 V	+ 6 V - 0,25 V	+12 V -7 V
Wyjście nadajnika	± 5 V min ± 15 V max	$\pm 3,6$ V $\pm 6,0$ V	± 2 V min	$\pm 1,5$ V min
Obciążenie nadajnika	3 k Ω do 7 k Ω	450 Ω min	100 Ω min	60 Ω min
Rezystancja wyjściowa nadajnika-wysoka impedancja	Zasilanie załączone			120 k Ω
	Zasilanie wyłączone	300 Ω	60 k Ω	60 k Ω
Rezystancja wejściowa odbiornika	3 k Ω do 7 k Ω	4 k Ω	4 k Ω	12 k Ω
Czułość odbiornika	± 3 V	± 200 mV	± 200 mV	± 200 mV

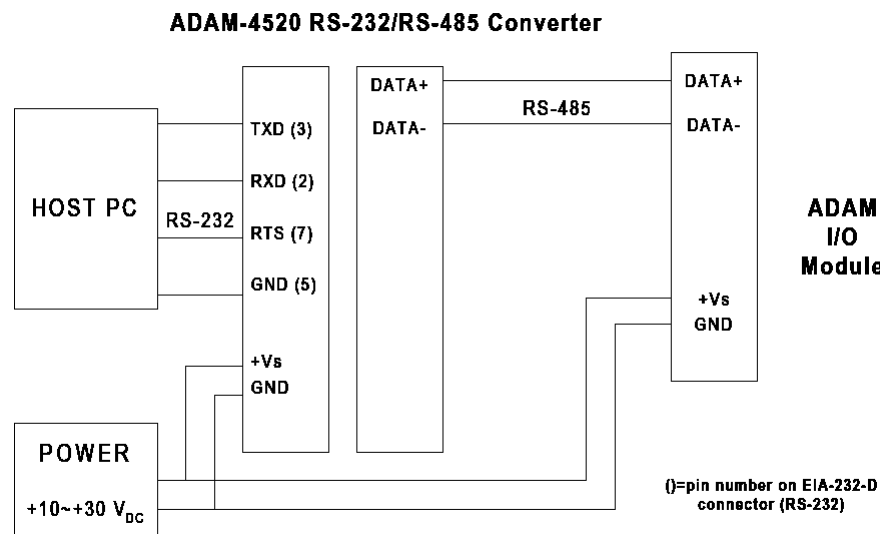
1.4. Przetworniki serii ADAM-4000

Seria przetworników ADAM-4000 obejmuje łącznie kilkadziesiąt typów różnych modułów pomiarowych z wejściami i wyjściami cyfrowymi oraz analogowymi, a także moduły zawierające liczniki impulsów. Dodatkowo dostępne są moduły komunikacyjne: konwerter interfejsów RS-232/RS-485 umożliwiający sterowanie przetwornikami na odległość do 1200m za pomocą komputera klasy PC wyposażonego w interfejs RS-232, konwerter interfejsów RS-485/RS-232 umożliwiający dołączenie do systemu przetworników dowolnego przyrządu pomiarowego wyposażonego w interfejs RS-232 oraz repeater interfejsu RS-485 umożliwiający zwiększenie zasięgu transmisji o kolejne 1200m. Łącznie w jednej sieci może pracować jednocześnie 256 przetworników serii ADAM-4000 (32 przetworniki w jednym segmencie sieci z konwerterem lub repeaterem), z których każdy posiada indywidualny adres, dzięki któremu jest rozróżniany.

W laboratorium studenci wykorzystują moduł komunikacyjny ADAM-4520 oraz moduł wyjść cyfrowych ADAM-4060 i moduł wejść analogowych ADAM-4013. Pełny wykaz modułów serii ADAM-4000 wraz z opisem sposobu programowania zawiera dokumentacja firmowa ADVANTECH [8].

1.4.1. Konwerter RS-232/RS485 ADAM-4520

Schemat blokowy konwertera RS-232/RS485 ADAM-4520 przedstawiono na rys. 9. Konwerter łączy się z komputerem sterującym za pomocą 4-przewodowego kabla ze złączami DB9, realizującego połączenie sygnałów TXD, RXD, RTS oraz masy GND. Z modułami serii ADAM-4000 konwerter komunikuje się za pomocą dwuprzewodowej, symetrycznej linii zawierającej sygnały DATA+ i DATA-. Konwerter zapewnia całkowitą izolację galwaniczną sieci modułów ADAM-4000 od komputera sterującego PC.



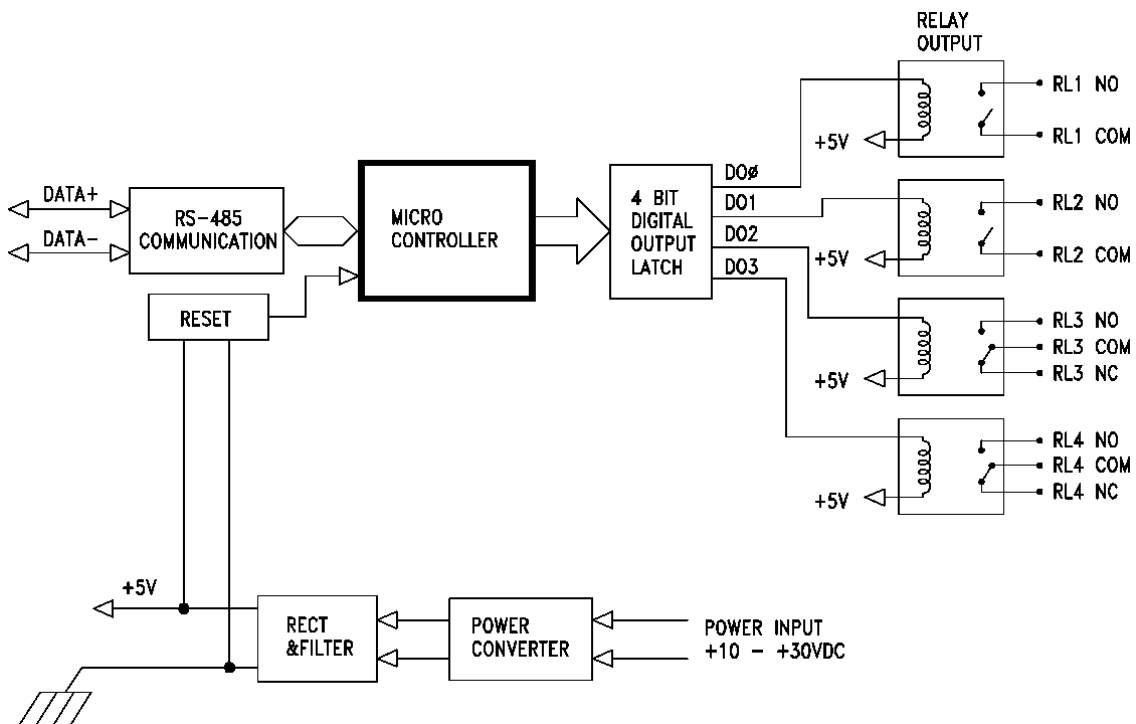
Rys. 9. Schemat blokowy konwertera RS-232/RS485 ADAM-4520 [8]

Konwerter posiada przełączniki ustalające parametry transmisji. Fabrycznie ustawiona jest transmisja o następujących parametrach: 1 bit startu, 8 bitów danych, 1 bit stopu, bez kontroli parzystości, szybkość transmisji 9600 bodów (bps). Nastaw tych **nie należy zmieniać**, gdyż nie będzie wtedy możliwa komunikacja z dołączonymi do konwertera modułami ADAM-4000. Nastawy te należy natomiast zastosować podczas pisania programu realizującego transmisję poprzez interfejs RS-232C w środowisku LabVIEW. Z punktu widzenia użytkownika konwerter ADAM-4520 jest całkowicie „przezroczysty”, tzn. nie wymaga on w

programie sterującym żadnej obsługi i po włączeniu zasilania od razu jest gotowy do pracy. Do jednego konwertera ADAM_4520 można dołączyć maksymalnie 32 moduły przy łącznej długości linii transmisyjnej 1200m. Dołączenie większej liczby modułów lub dłuższej linii transmisyjnej wymaga zastosowania repeatera ADAM-4510, każdy repeater umożliwia dołączenie kolejnych 32 modułów i wydłużenie linii transmisyjnej o kolejne 1200m. Linia transmisyjna powinna być zakończona terminatorem - rezystorem o wartości 120Ω.

1.4.2. Moduł wyjść cyfrowych ADAM-4060

Schemat blokowy modułu ADAM-4060 przedstawiono na rys. 10. Mikrokontroler odbiera komendy sterujące z symetrycznej linii danych (DATA+, DATA-) interfejsu RS-485, rozkodowuje je i steruje cewkami czterech przekaźników elektromechanicznych. Przekaźniki dołączone do linii danych DO0 i DO1 są typu „form A”, tzn. posiadają po jednym styku zwiernym NO (Normally Open), przekaźniki dołączone do linii danych DO2 i DO3 są typu „form C”, tzn. posiadają styki przełączane NO-NC (Normally Open - Normally Closed). Styki wszystkich czterech przekaźników są od siebie odizolowane, dzięki czemu mogą być użyte w niezależnych obwodach elektrycznych. Moduł może być zasilany napięciem stałym od +10V do +30V. Szczegółowe dane techniczne modułu ADAM-4060 przedstawiono na rys.11. Więcej szczegółów można znaleźć w dokumentacji producenta [8].



Rys. 10. Schemat blokowy modułu wyjść cyfrowych ADAM-4060 [8]

Sterowanie modułem ADAM-4060 polega na wysłaniu do niego poprzez interfejs RS-485 odpowiedniej komendy. Wszystkie komendy w systemie przetworników ADAM-4000 mają postać tekstową, tzn. są odpowiednim łańcuchem znaków przesyłanych kodem ASCII. Na rys.12 przedstawiono wszystkie komendy, które można wysłać do modułu ADAM-4060. Są wśród nich komendy wejściowe, wyjściowe i konfiguracyjne. W ćwiczeniu studenci będą wykorzystywać tylko jedną komendę cyfrowego wyjścia: *Digital Data Out*.

Format komendy sterującej cyfrowego wyjścia *Digital Data Out* jest szczegółowo wyjaśniony na rys. 12. Komenda to ma stałą długość: zawsze zawiera 8 znaków kodu ASCII, przy czym 7 pierwszych znaków należy do grupy tzw. znaków drukowalnych ASCII

(printable characters), czyli znaków które posiadają swoją drukowalną postać na urządzeniach wyjściowych (drukarka, monitor), a ostatni znak jest z grupy tzw. niedrukowalnych znaków ASCII (nonprintable characters), Znaki niedrukowalne są to tzw. znaki sterujące (control characters), które sterują pracą urządzeń wyjściowych i nie mają swojej drukowalnej postaci. W systemie ADAM-4000 wszystkie komendy kończą się znakiem sterującym CR (Carriage return - powrót karetki) o kodzie ASCII w zapisie szesnastkowym 0Dhex.

ADAM-4060 Specifications

Digital output	4-channel relay, 2 form A, 2 form C
Input speed (bps)	RS-485 (2-wire) 1200, 2400, 4800, 9600, 19.2K, 38.4K
maximum distance	4000 ft. (1200 m)
Contact rating	AC: 0.6 A/125 V; 0.3 A/250 V DC: 2 A/30 V; 0.6 A/ 110 V
Breakdown voltage	500 V _{AC} (50/60 Hz)
Relay on time (typical)	3 msec
Relay off time (typical)	1 msec
Total switching time	10 msec
Insulation resistance	1000 MΩ minimum at 500 V _{DC}
Watchdog timer	Yes
Power supply	+10 to +30 V _{DC} (non-regulated)
Power consumption	0.8 W

Rys. 11. Specyfikacja parametrów modułu wyjść cyfrowych ADAM-4060 [8]

Komenda *Digital Data Out* ma następującą składnię:

#AABB(data)(CR) , gdzie:

- # (hash) jest znakiem rozdzielającym (delimiter character) o kodzie ASCII 23hex.
- AA jest polem adresowym zawierającym dwuznakowy adres modułu w kodzie szesnastkowym w zakresie 00-FF. Wykorzystywany w ćwiczeniu moduł ADAM-4060 posiada adres w zapisie szesnastkowym 64hex.
- BB jest polem konfiguracyjnym sposób sterownia przekaźnikami. Jeśli BB=00 to możliwe jest sterowanie jednocześnie wszystkimi przekaźnikami, jeśli chcemy sterować pojedynczym przekaźnikiem, to pierwszy znak jest zawsze równy 1, a drugi jest numerem przekaźnika do sterowania (od 0 do 7).
- (data) jest dwuznakową liczbą szesnastkową reprezentującą cyfrowe dane wyjściowe:
 - jeśli sterujemy pojedynczym przekaźnikiem, to pierwszy znak jest zawsze równy 0, a drugi jest równy 0 (wyłącza przekaźnik) lub 1 (załącza przekaźnik),
 - jeśli sterujemy wszystkimi przekaźnikami jednocześnie, to dwuznakowa liczba szesnastkowa jest interpretowana w zapisie binarnym w ten sposób, że każdy bit steruje niezależnie jednym przekaźnikiem, szczegóły są widoczne na rys. 13.
- (CR) (Carriage return - powrót karetki) jest znakiem sterującym ASCII 0Dhex oznaczającym koniec komendy.

ADAM-4060/4068 Command Table

Command Syntax	Command Name	Command Description
%AANNTTCCFF	Configuration	Sets address, baud rate, and/or checksum status, to a digital I/O module
\$AA6	Digital Data In	Returns the values of the digital I/O channels of the addressed module
#AABB(data)	Digital Data Out	Writes specified values to either a single channel or all channels simultaneously
##	Synchronized Sampling	Orders all digital I/O modules to sample their input values and store them in a special register
\$AA4	Read Synchronized Data	Return the value of a specified digital I/O module that was stored after an ## command was issued
\$AA2	Configuration Status	Returns the configuration parameters of a specified digital I/O module
\$AA5	Reset Status	Indicates whether a specified digital I/O module was reset after the last time the \$AA5 command was issued
\$AAF	Read Firmware Version	Return the firmware version code from the specified digital I/O module
\$AAM	Read Module Name	Return the module name from the specified digital I/O module
\$AAX0TTTTDDDD	Write Safty Value	Force the DO channels to safety status when communication is time-out and over pre-defined period.
\$AAX1	Read Safty Value	Read the time-out setting and pre-defined safety status of DO channels.
\$AAX2	Read Safty Flag	Requests the Safty Flag of the addressed digital I/O module to see whether the safety value has been executed since Write Safety Value command was set.

Rys. 12. Komendy do obsługi modułu wyjść cyfrowych ADAM-4060 [8]

Przykładowo, komenda *Digital Data Out* o następującej postaci:

#640000(CR)

spowoduje wyłączenie wszystkich przekaźników w module o adresie 64hex, a komenda:

#64000A(CR)

w module o adresie 64hex załączy przekaźniki o numerach 1 i 3 oraz wyłączy przekaźniki o numerach 0 i 2, ponieważ liczba szesnastkowa 0Ahex ma zapisie binarnym postać: 00001010 bin (dla przypomnienia: $0A_{16} = 10_{10} = 00001010_2$).

Command Set**4050, 4060, 4055, 4068****#AABB**

Name	Digital Data Out
Description	The command either sets a single digital output channel or sets all digital output channels simultaneously.
Syntax	#AABB(data)(cr) # is a delimiter character. AA (range 00-FF) represents the 2-character hexadecimal address of the digital I/O module you want to set its output value. BB is used to indicate whether all channels will be set or a single channel will be set. In the last case BB also indicates which channel. Writing to all channels (write a byte): both characters should be equal to zero (BB=00). Writing to a single channel (write a bit): First character is 1, second character indicates channel number which can range from 0 to 7. (data) is the hexadecimal representation of the digital output value(s). When writing to a single channel (bit) the first character is always 0. The value of the second character is either 0 or 1. When writing to all channels (byte) , both characters are significant (range 00h-FFh). The digital equivalent of these two hexadecimal characters represent the channels values. The amount of channels on the ADAM-4050, ADAM-4055, ADAM-4060 and ADAM-4068 differs. The value 7A would mean the following for the 8 channels on the ADAM-4050, ADAM-4055 and ADAM-4068:

digital value:	0	1	1	1	1	0	1	0
ADAM-4050/4055/4068 channel no.	7	6	5	4	3	2	1	0

Since the ADAM-4060 has only four output channels all the meaning full values lie between 00h and 0Fh. The value 0Ah would mean the following for the ADAM-4060:

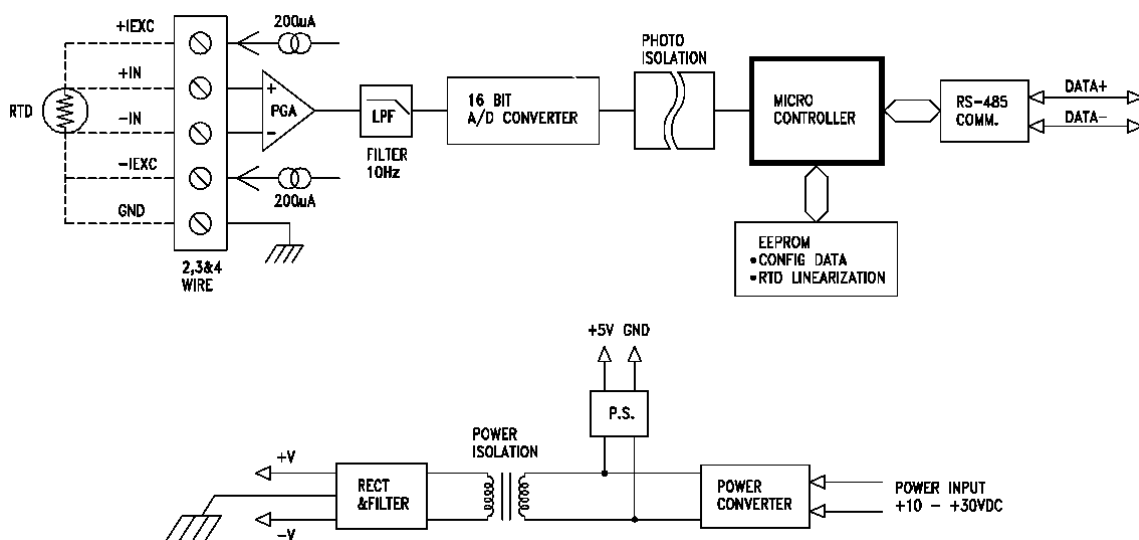
digital value:	0	0	0	0	1	0	1	0
ADAM-4060 channel no.	-	-	-	-	3	2	1	0

Rys. 13. Format komendy sterującej wyjściami cyfrowymi modułu ADAM-4060 [8]

1.4.3. Moduł wejść analogowych ADAM-4013

Schemat blokowy modułu ADAM-4013 przedstawiono na rys. 14. Moduł przeznaczony jest do pomiaru temperatury za pomocą czujników termorezystancyjnych RTD (*Resistance Temperature Detector*) platynowych typu Pt100 lub niklowych typu Ni100. Moduł zawiera dwa źródła prądowe o wydajności $200\mu\text{A}$ do zasilania czujnika, filtr dolnoprzepustowy 10Hz do tłumienia zakłóceń, 16-bitowy przetwornik analogowo-cyfrowy, optyczny układ izolacji galwanicznej, mikrokontroler sterujący, bufor magistrali interfejsu RS-485 oraz układy zasilania. Zależnie od zastosowanego czujnika oraz wybranego zakresu pomiarowego możliwy jest pomiar temperatury od -100°C do $+600^{\circ}\text{C}$. Konfigurację modułu przeprowadza się wysyłając do niego odpowiednią komendę (%AA - rys. 16). Mikrokontroler odbiera komendy sterujące z symetrycznej linii danych (DATA+, DATA-) interfejsu RS-485, rozkodowuje je i steruje procesem pomiaru temperatury. Po wykonaniu pomiaru mikrokontroler wysyła wynik pomiaru na magistralę interfejsu RS-485.

Moduł umożliwia dwu- trzy- lub czteroprzewodowe dołączenie czujnika, dzięki czemu możliwa jest eliminacja wpływu rezystancji przewodów na pomiar temperatury. 16-bitowy przetwornik analogowo-cyfrowy zapewnia pomiar temperatury z rozdzielczością 0.01°C i błędami nie większymi od 0,05%. Szczegółowe dane techniczne modułu ADAM-4013 przedstawiono na rys.15. Więcej szczegółów można znaleźć w dokumentacji producenta [8].



Rys. 14. Schemat blokowy modułu wejść analogowych ADAM-4013 [8]

Wykorzystanie modułu ADAM-4013 polega na wysłaniu do niego poprzez interfejs RS-485 odpowiedniej komendy do pomiaru temperatury i odczytaniu wyniku. Na rys.16 przedstawiono wszystkie komendy, które można wysłać do modułu ADAM-4013. Są wśród nich komendy wejściowe, wyjściowe i konfiguracyjne.

Na potrzeby ćwiczenia moduł 4013 został skonfigurowany następująco: adres modułu ustawiono na BBhex, zakres pomiarowy od 0°C do $+200^{\circ}\text{C}$ dla czujnika Pt100 $\alpha=0,00385$ (Range Code = 22hex), format danych wyjściowych: Engineering Units od $+000.00^{\circ}\text{C}$ do $+200.00^{\circ}\text{C}$, rozdzielczość 0.01°C (8-bit parameter =00hex). Szczegóły konfiguracji modułu ADAM-4013 z wykorzystaniem komendy %AA zawiera dokumentacja firmowa [8].

W ćwiczeniu studenci będą wykorzystywać tylko jedną komendę analogowego wejścia: *Analog Data In*. Format komendy analogowego wejścia *Analog Data In* jest szczegółowo wyjaśniony na rys. 17. Komenda to ma stałą długość: zawsze zawiera 4 znaki kodu ASCII, przy czym 3 pierwsze znaki należą do grupy znaków drukowalnych ASCII, a ostatni czwarty

znak jest niedrukowalnym znakiem sterującym CR. Po odebraniu komendy *Analog Data In* moduł ADAM-4013 wykonuje pomiar temperatury i wysyła wynik pomiaru przez interfejs RS-485. Format wysyłanych danych przedstawiony jest na rys. 17, jest on zależny od sposobu skonfigurowania modułu za pomocą komendy %AA. Dla przedstawionego sposobu konfiguracji moduł wysyła wynik pomiaru w postaci ciągu 9 znaków ASCII.

Komenda Analog Data In ma następującą składnię:

#AA(CR) , gdzie:

- # (hash) jest znakiem rozdzielającym (delimiter character) o kodzie ASCII 23hex.
- AA jest polem adresowym zawierającym dwuznakowy adres modułu w kodzie szesnastkowym w zakresie 00-FF. Wykorzystywany w ćwiczeniu moduł ADAM-4013 posiada adres w zapisie szesnastkowym BBhex.
- (CR) (Carriage return - powrót karetki) jest znakiem sterującym ASCII 0Dhex oznaczającym koniec komendy.

Po wykonaniu pomiaru moduł ADAM-4013 wysyła wynik pomiaru w postaci:

>(data)(CR) , gdzie:

- > (znak większości) jest znakiem rozdzielającym o kodzie ASCII 3Ehex.
- (data) jest polem danych zawierającym wynik pomiaru o formacie zależnym od sposobu skonfigurowania modułu. Moduł zastosowany w ćwiczeniu skonfigurowano tak, że w polu danych odsyła zawsze 7 znaków: znak + (plus), 3 cyfry wyniku, kropkę dziesiętną i dwie cyfry ułamkowej części wyniku.
- (CR) (Carriage return - powrót karetki) jest znakiem sterującym ASCII 0Dhex oznaczającym koniec komendy.

Podsumowując, moduł ADAM-4013 skonfigurowany na potrzeby ćwiczenia po wykonaniu pomiaru wysyła wynik pomiaru w postaci ciągu 9 następujących znaków:

>+XXX.XX(CR) , gdzie:

- + znak plus, zawsze występuje.
- XXX 3 cyfry całkowitej części wyniku w °C.
- kropka rozdzielająca część całkowitą od ułamkowej w zapisie dziesiętnym. Należy zwrócić uwagę, że w komputerze z zainstalowaną polską wersją systemu Windows oczekiwany jest znak rozdzielający w postaci przecinka, co utrudnia konwersję wyniku pomiaru z postaci tekstowej na liczbową.
- XX 2 cyfry ułamkowej części wyniku w °C.

Przykładowo, komenda *Analog Data In* o następującej postaci:

#BB(CR)

spowoduje wykonanie pomiaru temperatury przez moduł ADAM-4013 o adresie BBhex, a następnie wysłanie wyniku pomiaru o przykładowej postaci:

>+064.15(CR)

co oznacza, że zmierzono temperaturę dodatnią o wartości 64,15°C. Nieznaczące zera są również zawsze wysyłane, a więc wynik pomiaru zawsze składa się z ciągu zawierającego 9 znaków ASCII, przy czym ostatni znak (CR) jest znakiem niedrukowalnym, a więc nie będzie wyświetlany na ekranie monitora.

ADAM-4013 Specifications

Input range	Pt and Ni RTD
Output	RS-485 (2-Wire)
speed (in bps)	1200, 2400, 4800, 9600, 19.2K, 38.4K
maximum distance	4000 ft. (1200 m.)
Accuracy	±0.05% or better
Zero drift	±0.01 °C/ °C
Span drift	±0.01 °C/ °C
Input connections	2, 3, or 4 wires
Isolation-rated voltage	3000 V _{DC}
CMR @ 50/60 Hz	150 dB
NMR @ 50/60 Hz	100 dB
Bandwidth	4 Hz
Conversion rate	10 samples/sec.
Input impedance	2 MΩ
Watchdog timer	Yes
Power supply	+10 to +30 V _{DC} (non-regulated)
Power consumption	0.7 W

Rys. 15. Specyfikacja parametrów modułu wejść analogowych ADAM-4013 [8]

Do sprawdzenia aktualnej konfiguracji modułu ADAM-4013 można wykorzystać komendę odczytu *Configuration Status* (rys. 16).

Komenda *Configuration Status* ma następującą składnię:

\$AA2(CR) , gdzie:

- \$ (dolar) jest znakiem rozdzielającym (delimiter character) o kodzie ASCII 24hex.
- AA jest polem adresowym zawierającym dwuznakowy adres modułu w kodzie szesnastkowym w zakresie 00-FF. Wykorzystywany w ćwiczeniu moduł ADAM-4013 posiada adres w zapisie szesnastkowym BBhex.
- 2 cyfra 2 jest stałym elementem komendy Configuration Status.
- (CR) (Carriage return - powrót karetki) jest znakiem sterującym ASCII 0Dhex oznaczającym koniec komendy.

Po odebraniu komendy Configuration Status moduł ADAM-4013 wysyła odpowiedź zawierającą informację o aktualnej konfiguracji modułu w postaci:

!AATTCCFF(CR) , gdzie:

- ! (wykrzyknik) jest znakiem rozdzielającym o kodzie ASCII 21hex.
- AA jest dwuznakowym adresem modułu w kodzie szesnastkowym
- TT (Range Code) zawiera informację o rodzaju czujnika i zakresie pomiarowym.
- CC (Baud Rate Code) zawiera informację o ustawionej szybkości transmisji.
- FF (8-bit parameter) zawiera informację o formacie wysyłanych danych.
- (CR) (Carriage return - powrót karetki) jest znakiem sterującym ASCII 0Dhex oznaczającym koniec komendy.

Odpowiedź na wysłaną komendę *Configuration Status* ma zawsze długość 10 znaków ASCII. Szczegółowe znaczenie poszczególnych pól statusu konfiguracji zawiera dokumentacja firmowa [8].

Przykładowo, komenda *Configuration Status* o następującej postaci:

%BB2(CR)

spowoduje wysłanie przez moduł ADAM-4013 o adresie BBhex następującej odpowiedzi:

!BB220600(CR)

co oznacza, że moduł jest skonfigurowany następująco:

- BB adres modułu w kodzie szesnastkowym ustawiono na BBhex.
- 22 (Range Code) zakres pomiarowy od 0°C do +200°C dla czujnika Pt100, $\alpha=0,00385$.
- 06 (Baud Rate Code) szybkości transmisji ustawiona na 9600bps.
- 00 (8-bit parameter) format wysyłanych danych ustawiony na Engineering Format.

ADAM-4013 Command Table

Command Syntax	Command Name	Command Description
%AANNTTCCFF	Configuration	Sets the address, baud rate, data format, checksum status, and/or integration time for a specified analog input module
#AA	Analog Data In	Returns the input value from a specified analog input module in the currently configured data format
\$AA0	Span Calibration	Calibrates an analog input module to correct for gain errors
\$AA1	Offset Calibration	Calibrates an analog input module to correct for offset errors
#**	Synchronized Sampling	Orders all analog input modules to sample their input values and store them in special registers
\$AA4	Read Synchronized Data	Returns the value that was stored in the specified module's register after the #** command
\$AA2	Configuration Status	Returns the configuration parameters for the specified analog input module
\$AAF	Read Firware Version	Returns the firmware version code from the specified analog input module
\$AAM	Read Module Name	Returns the module name from the specified analog input module

Rys. 16. Komendy do obsługi modułu wejść analogowych ADAM-4013 [8]

**4011, 4011D, 4012, 4013, 4015, 4016,
4017, 4017+, 4018, 4018+, 4019**

#AA

Name Analog Data In

Description The command will return the input value from a specified (AA) module in the currently configured data format.

Syntax #AA(cr)

is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.

(cr) is the terminating character, carriage return (0Dh).

Response >(data)(cr)

There is no response if the module detects a syntax error or communication error or if the specified address does not exist.

> is a delimiter character.

(data) is the input value in the configured data format of the interrogated module. (For data formats, see Appendix B).

(cr) is the terminating character, carriage return (0Dh).

Example

command: #33(cr)

response: >+5.8222(cr)

The command interrogates the analog input module at address 33h for its input value.

The analog input module responds with +5.8222 volts. (The configured data format of the analog input module in this case is engineering units.)

Example

command: #21(cr)

response: +7.2111+7.2567+7.3125+7.1000
+7.4712+7.2555+7.1234+7.5678 (cr)

The command interrogates the analog input module at address 21h for its input values of all channels.

The analog input module responds with channels from 0 to 7 with +7.2111 volts, +7.2567 volts, +7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts and +7.5678 volts.

Rys. 17. Format komendy odczytującej dane z modułu wejść analogowych ADAM-4013


2. Programowanie modułów ADAM-4000 w środowisku LabVIEW








2.1. Podstawy programowania w środowisku LabVIEW

Studenci samodzielnie powinni zaopatrzyć się w literaturę wspomagającą naukę programowania w LabView. Przykładowe pozycje, w tym dostępne w Internecie darmowe podręczniki, zamieszczono w wykazie literatury.

LabView jest graficznym środowiskiem programistycznym przeznaczonym do tworzenia programów zorientowanych na obsługę systemów pomiarowych. Aby napisać program w środowisku LabView należy:

- a- uruchomić środowisko LabView,
- b- utworzyć nowy plik programu: New\Blank_VI,
- c- rozmieścić potrzebne kontrolki (*controls*) w oknie panelu programu,
- d- rozmieścić potrzebne elementy funkcyjne (*functions*) w oknie diagramu programu,
- e- wykonać odpowiednie połączenia na diagramie realizujące algorytm programu,
- f- uruchomić program, ocenić poprawność działania i wyszukać błędy,
- g- usunąć błędy w programie modyfikując zawartość panelu i diagramu programu,
- h- powtarzać punkty f oraz g aż do osiągnięcia oczekiwanego rezultatu.

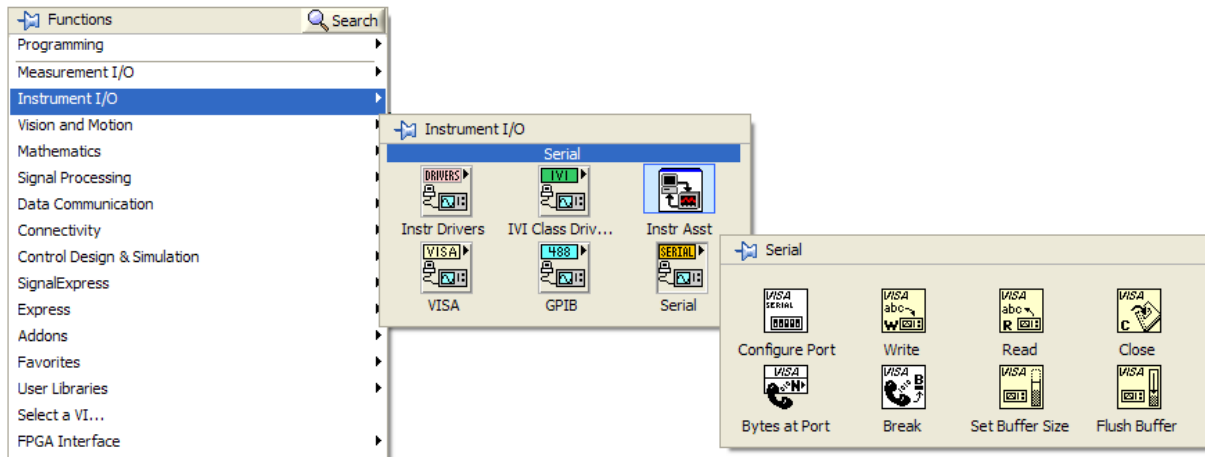
Każda aplikacja przygotowana w środowisku LabView składa się z dwóch części: Panelu i Diagramu. Panel stanowi graficzny interfejs użytkownika aplikacji, natomiast Diagram jest graficznym zapisem algorytmu realizowanego przez tę aplikację. Po otwarciu aplikacji w środowisku LabView widoczny jest jej Panel sterujący. Przelączenie pomiędzy widokiem Panelu i Diagramu jest możliwe za pomocą kombinacji klawiszy **CTRL+E**. Kombinacją klawiszy **CTRL+T** możemy wybrać jednoczesny widok okna Panelu i Diagramu. Analizę Diagramu programu można sobie znacznie ułatwić włączając przyciskiem  okno pomocy kontekstowej **Context Help**.

Do uruchomienia programu służą przyciski  i . Proste programy nie posiadające w swej strukturze pętli programowych należy zazwyczaj uruchamiać przyciskiem , dzięki czemu pracują one w sposób ciągły i można je zatrzymać przyciskiem . Programy bardziej złożone posiadające w swej strukturze pętle programowe należy uruchamiać zazwyczaj przyciskiem , a do ich zatrzymywania służy odpowiedni przycisk sterujący w tej aplikacji. Uruchomienie programu sygnalizowane jest zmianą postaci przycisków na  i .

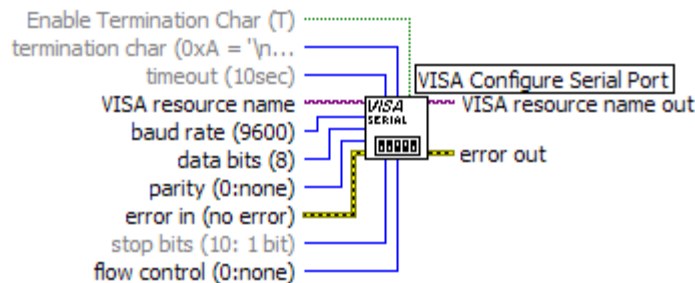
2.2. Obsługa interfejsu RS-232C w środowisku LabVIEW

Do obsługi interfejsu RS-232C w środowisku można wykorzystać funkcje z biblioteki VISA (*Virtual Instruments Software Architecture*). Funkcje te są dostępne w widoku Diagramu na palecie *Function* w opcji *→Instrument I/O →Serial*, tak jak pokazano to na rys. 18. W ćwiczeniu wykorzystywane będą następujące funkcje: *VISA Configure Serial Port*, *VISA Write*, *VISA Read* oraz *VISA Close*.

Funkcję *VISA Configure Serial Port* przedstawiono na rys. 19. Służy ona do konfigurowania portu szeregowego RS-232C, co wymaga podania na odpowiednie wejścia następujących danych: numeru portu (*VISA resource name*), szybkości transmisji (*baud rate*), liczby bitów danych (*data bits*), sposobu kontroli parzystości (*parity*), liczby bitów stopu (*stop bits*), sposobu kontroli transmisji (*flow control*), czasu oczekiwania (*timeout*) oraz znaku końca transmisji (*termination char*). Zawsze niezbędne jest podanie numeru portu, pozostałe parametry, jeśli nie będą podane jawnie, funkcja ustawi je na wartości domyślne, podana w nawiasach na rys. 19. Wejścia i wyjścia: *VISA resource name*, *VISA resource name out*, *error in*, *error out* mogą być wykorzystane do kaskadowego łączenia funkcji, co gwarantuje ich odpowiednią kolejność wykonania przez program.



Rys. 18. Funkcje VISA Instrument I/O do obsługi interfejsu RS-232C



Rys. 19. Funkcja VISA Configure Serial Port

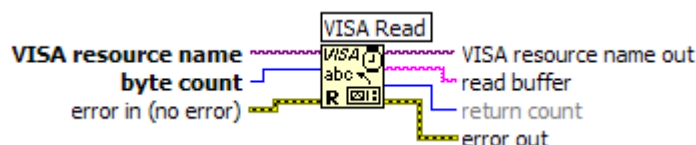
Funkcję *VISA Write* przedstawiono na rys. 20. Służy ona do wysyłania danych przez port szeregowy RS-232C. Wymaga ona podania numeru portu (*VISA resource name*) i danych do wysłania (*write buffer*). Dane do wysłania są zmienną typu tekstowego (różowy kolor linii). Podobnie jak dla poprzedniej funkcji, wejścia i wyjścia: *VISA resource name*, *VISA resource name out*, *error in*, *error out* mogą być wykorzystane do kaskadowego łączenia funkcji. Na wyjściu (*return count*) dostępna jest informacja o liczbie wysłanych znaków. Należy zauważyć, że dane wysyłane za pomocą funkcji *VISA Write* są typu tekstowego, a więc podczas wysyłania danych liczbowych należy je uprzednio przekonwertować na zmienną tekstową.



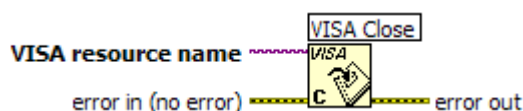
Rys. 20. Funkcja VISA Write

Funkcję *VISA Read* przedstawiono na rys. 21. Służy ona do odbierania danych z portu szeregowego RS-232C. Wymaga ona podania numeru portu (*VISA resource name*) i liczby bajtów danych do odczytania (*byte count*). Przy błędnym podaniu liczby bajtów wystąpią błędy: albo nie zostaną odczytane wszystkie dane (gdy podano za małą wartość *byte count*) lub funkcja będzie bezskutecznie oczekiwała na kolejne dane (gdy podano za dużą wartość *byte count*), aż do przekroczenia czasu *timeout*. Gdy nie jest znana liczba bajtów do odczytania, można skorzystać z funkcji *Bytes at Port* (rys. 18). Podobnie jak dla poprzedniej funkcji, wejścia i wyjścia: *VISA resource name*, *VISA resource name out*, *error in*, *error out*

mogą być wykorzystane do kaskadowego łączenia funkcji. Na wyjściu *return count* dostępna jest informacja o liczbie odebranych znaków. Odebrane znaki dostępne są na wyjściu *read buffer*. Należy zauważyć, że dane odebrane za pomocą funkcji *VISA Read* są typu tekstowego, a więc podczas transmisji przez port RS-232C danych liczbowych należy odebrane znaki przekonwertować ze zmiennej tekstowej na zmienną numeryczną odpowiedniego typu. Szczególną uwagę należy zwrócić na problem stosowania różnych znaków oddzielających część ułamkową w zapisie dziesiętnym (stosowana jest kropka lub przecinek).

Rys. 21. Funkcja *VISA Read*

Funkcję *VISA Close* przedstawiono na rys. 22. Służy ona do zamykania portu szeregowego RS-232C po zakończeniu transmisji. Wymaga ona podania tylko numeru portu do zamknięcia (*VISA resource name*). Podobnie jak dla poprzedniej funkcji, wejścia i wyjścia: *VISA resource name*, *error in*, *error out* mogą być wykorzystane do kaskadowego łączenia funkcji.

Rys. 22. Funkcja *VISA Close*

Kompletna obsługa transmisji przez port RS-232C wymaga kolejno zastosowania funkcji: *VISA Configure Serial Port*, *VISA Write*, *VISA Read* oraz *VISA Close*. Na rys. 23 przedstawiono przykładowy Diagram programu obsługującego interfejs RS-232 do odczytu konfiguracji modułu ADAM-4013 (patrz p. 1.4.3) Wykorzystano w tym celu komendę *Configuration Status* (rys. 16), moduł ADAM-4013 posiada adres BBhex. W pierwszej kolejności za pomocą funkcji *VISA Configure Serial Port* skonfigurowano port szeregowy w następujący sposób: numer portu COM1, szybkość transmisji 9600bps, 8 bitów danych, brak kontroli parzystości, jeden bit stopu, bez sprzętowej kontroli transmisji. Do szybkiego skonfigurowania funkcji *VISA Configure Serial Port* warto skorzystać z opcji *Create* → *Constant*, dostępnej po prawym kliknięciu myszką na skonfigurowanym wejściu.

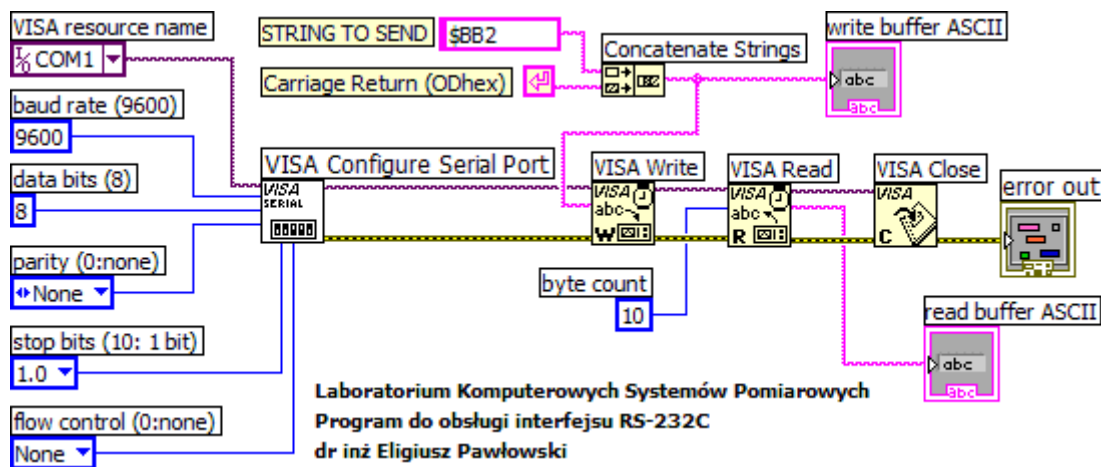
Za pomocą funkcji *Concatenate String* (paleta *Funktion* → *Programming* → *String*) utworzono komendę sterującą w postaci łańcucha znaków: \$BB2(CR). Niedrukowalny znak sterujący ASCII (CR) jest dostępny jako *Carriage Return Constant* (paleta *Funktion* → *Programming* → *String*). Utworzoną daną tekstową dołączono do wejścia *write buffer* funkcji *VISA Write* oraz do umieszczonego na panelu wskaźnika (*indicator*).

Na wejściu *byte count* funkcji *VISA Read* utworzono stałą numeryczną o wartości 10, ponieważ moduł ADAM-4013 wysyła dane konfiguracyjne w postaci łańcucha o długości dziesięciu znaków. Na wyjściu *read buffer* utworzono wskaźnik (*Create* → *Indicator*), który na Panelu będzie pokazywał dane odczytane z portu szeregowego.

Jako ostatnią użyto funkcję *VISA Close* do zamknięcia portu szeregowego. Na jej wyjściu *error out* utworzono wskaźnik (*Create* → *Indicator*) pokazujący na Panelu ewentualne błędy w transmisji.

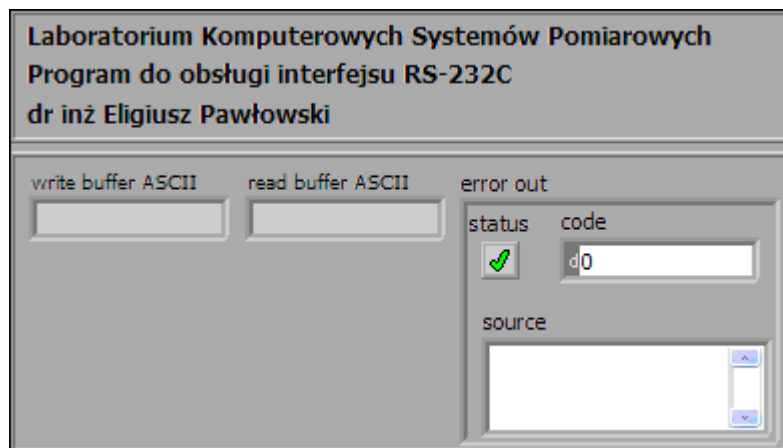
Aby wszystkie użyte funkcje były wykonane w odpowiedniej kolejności (konfiguracja portu → wysłanie danych → odbiór danych → zamknięcie portu) połączono ze sobą

odpowiednio wszystkie wejścia *VISA resource name*, *error in* i wyjścia: *VISA resource name out*, *error out* kolejnych funkcji.



Rys. 23. Przykładowy Diagram programu obsługującego interfejs RS-232 do odczytu komendą *Configuration Status* konfiguracji modułu ADAM-4013 pod adresem BBhex

Na rys. 24 przedstawiono przykładowy Panel omawianego programu. Prezentowane są na nim dane wysyłane przez port szeregowy (*write buffer ASCII*) i dane odbierane (*read buffer ASCII*) oraz ewentualne błędy transmisji (*error out*).



Rys. 24. Przykładowy Panel programu z rys. 23

Przedstawiony program jest bardzo uproszczony i jego możliwości są niewielkie, można jednak z powodzeniem wykorzystać go do testowania poprawności transmisji, sprawdzenia działania pojedynczych modułów ADAM-4000, konfigurowania ich, pomiarów itp. Należy w tym celu ustalić postać komendy do wysłania oraz liczbę znaków do odebrania i wpisać je w odpowiednie miejsca na Diagramie.

2.3. Sterowanie modułem ADAM-4060 w środowisku LabVIEW

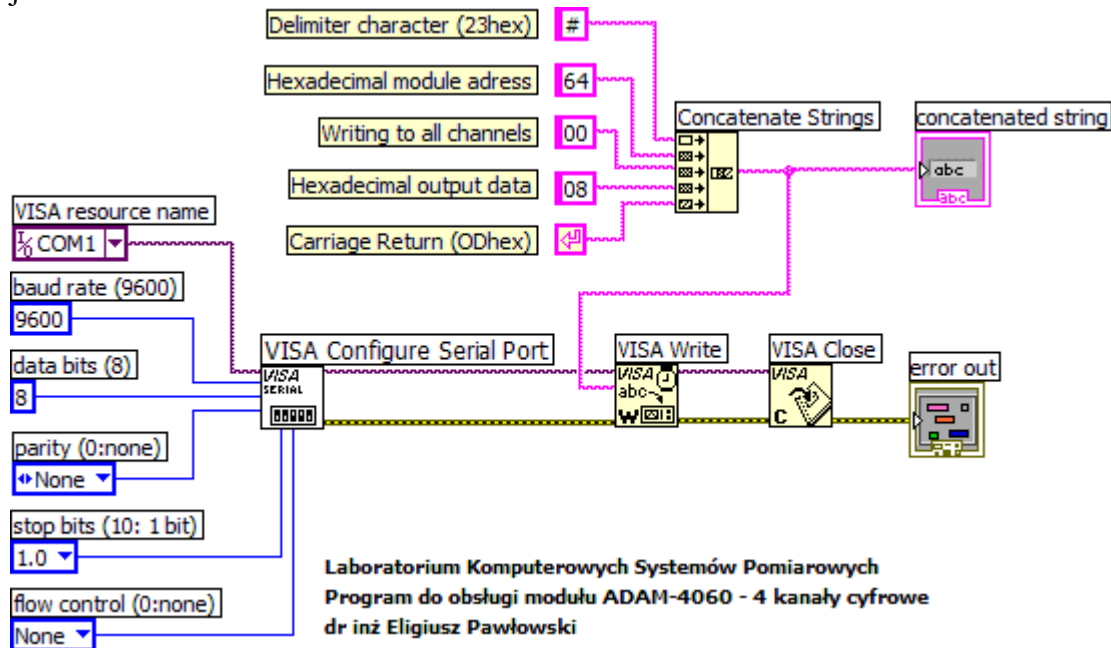
Sterownie modułem wyjść cyfrowych ADAM-4060 wymaga wysłania do niego komendy sterującej, która spowoduje zamknięcie lub otwarcie styków przekaźników znajdujących się w tym module. Strukturę odpowiedniej do tego celu komendy sterującej *Digital Data Out* szczegółowo omówiono w p. 1.4.2. Na rys. 25 przedstawiono przykładowy Diagram programu sterującego modułem ADAM-4060. Do obsługi portu szeregowego zastosowano

funkcje: *VISA Configure Serial Port*, *VISA Write*, oraz *VISA Close*. Funkcja *VISA Read* do tego celu nie jest potrzebna. Do sterowania modulem ADAM-4060 służy komenda *Digital Data Out* (rys. 13), modulem ADAM-4060 posiada adres 64hex. W pierwszej kolejności za pomocą funkcji *VISA Configure Serial Port* skonfigurowano port szeregowy analogicznie jak opisano to w p. 2.2.


Komendę sterującą *Digital Data Out* utworzono za pomocą funkcji *Concatenate String* (paleta *Funktion* → *Programming* → *String*) zgodnie ze składnią przedstawioną na rys. 13. Utworzony łańcuch znaków w postaci: #640008(CR) ma za zadanie załączyć przełącznik dołączony do linii DO3 (rys. 10) i zapalić diodę LED4 (rys. 1 i rys. 3), decyduje o tym wartość *Hexadecimal output data* = 08 ($08_{16} = 8_{10} = 00001000_2$). Niedrukowalny znak sterujący ASCII (CR) wstawiono jako *Carriage Return Constant* (paleta *Funktion* → *Programming* → *String*). Utworzoną daną tekstową dołączono do wejścia *write buffer* funkcji *VISA Write* oraz do umieszczonego na panelu wskaźnika *concatenated string*.

Jako ostatnią użyto funkcję *VISA Close* do zamknięcia portu szeregowego. Na jej wyjściu *error out* utworzono wskaźnik pokazujący na Panelu ewentualne błędy w transmisji.

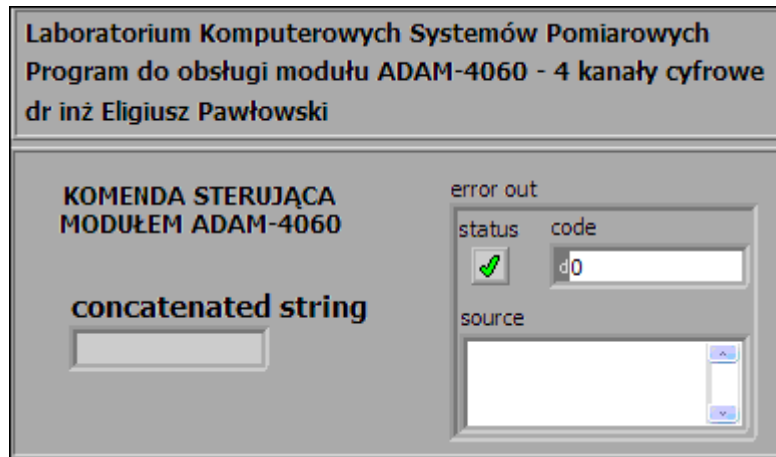
Aby wszystkie użyte funkcje były wykonane w odpowiedniej kolejności (konfiguracja portu → wysłanie danych → zamknięcie portu) połączono ze sobą odpowiednio wszystkie wejścia *VISA resource name*, *error in* i wyjścia: *VISA resource name out*, *error out* kolejnych funkcji.



Rys. 25. Diagram programu sterującego modulem wyjść cyfrowych ADAM-4060

Na rys. 26 przedstawiono przykładowy Panel omawianego programu. Prezentowane są na nim dane wysyłane przez port szeregowy (*concatenated string*) oraz ewentualne błędy transmisji (*error out*). Po uruchomieniu programu przyciskiem  powinna zapalić się dioda LED4 zainstalowana na panelu z przetwornikami ADAM. Aby sterować pozostałymi przełącznikami modułu ADAM-4060 należy zmodyfikować wartość *Hexadecimal output* zgodnie ze składnią komendy *Digital Data Out* przedstawionej na rys.13.

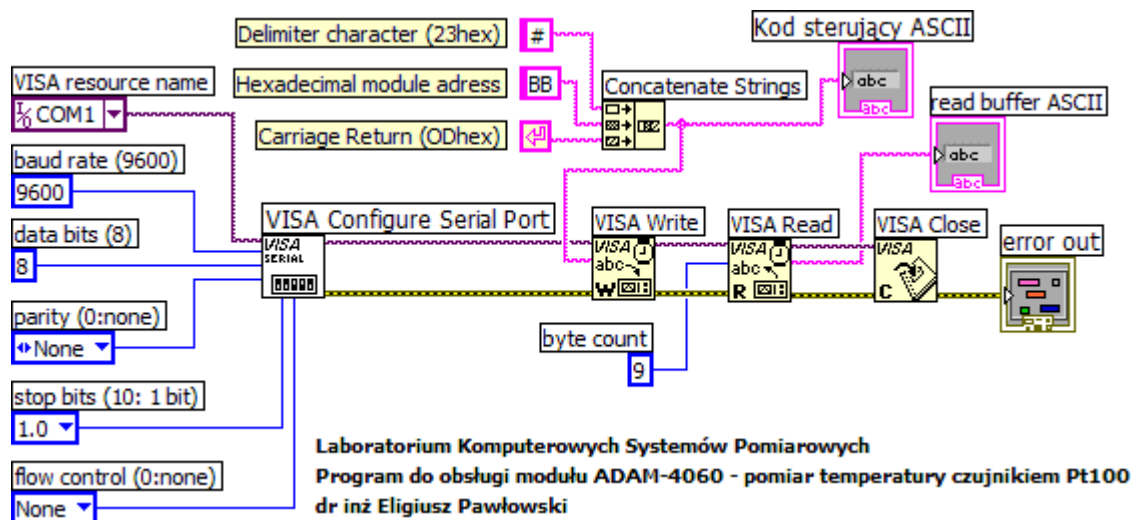
Przedstawiony program jest bardzo uproszczony i jego możliwości są niewielkie. Aby uzyskać większą funkcjonalność programu należy umożliwić niezależne sterowanie poszczególnymi przełącznikami za pomocą zmiennych logicznych (*True/False*), co wymaga odpowiedniej konwersji typów zmiennych z logicznego na numeryczny i na tekstowy. Zagadnienie to będzie omówione w dalszej części instrukcji.



Rys. 26. Panel programu sterującego modułem wyjść cyfrowych ADAM-4060

2.4. Pomiary temperatury modułem ADAM-4013 w środowisku LabVIEW

Realizacja pomiarów temperatury modułem wejść analogowych ADAM-4013 wymaga wysłania do niego komendy wymuszającej wykonanie pomiaru i następnie umożliwiającej odczytanie wartości zmierzonej temperatury. Strukturę odpowiedniej do tego celu komendy *Analog Data In* szczegółowo omówiono w p. 1.4.3. Na rys. 27 przedstawiono przykładowy Diagram programu realizującego pomiar temperatury modułem wejść analogowych ADAM-4013. Do obsługi portu szeregowego zastosowano funkcje: *VISA Configure Serial Port*, *VISA Write*, *VISA Read* oraz *VISA Close*. Do pomiarów temperatury modułem ADAM-4013 służy komenda *Analog Data In* (rys. 17), moduł ADAM-4013 posiada adres BHex. W pierwszej kolejności za pomocą funkcji *VISA Configure Serial Port* skonfigurowano port szeregowy analogicznie jak opisano to w p. 2.2.



Rys. 27. Diagram programu do pomiarów temperatury modułem ADAM-4013


Komendę *Analog Data In* utworzono za pomocą funkcji *Concatenate String* (paleta *Funkction* → *Programming* → *String*) zgodnie ze składnią przedstawioną na rys. 17. Utworzony łańcuch znaków w postaci: #BB(CR) ma spowodować wykonanie pomiaru temperatury przez moduł ADAM-4013 posiadający adres BHex i wysłanie przez port

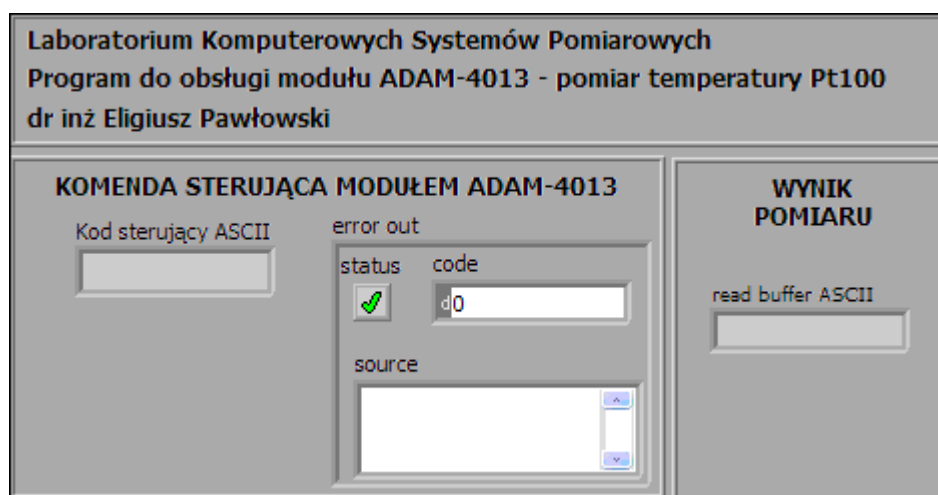
szeregowy wyniku tego pomiaru. Niedrukowalny znak sterujący ASCII (CR) wstawiono jako *Carriage Return Constant* (paleta *Funktion* → *Programming* → *String*). Utworzoną daną tekstową dołączono do wejścia *write buffer* funkcji *VISA Write* oraz do umieszczonego na panelu wskaźnika *Kod sterujący ASCII*.

W celu odczytania wyniku pomiaru zastosowano funkcję *VISA Read*. Na wejściu *byte count* funkcji *VISA Read* utworzono stałą numeryczną o wartości 9, ponieważ zawsze moduł ADAM-4013 wysyła wynik pomiaru temperatury w postaci łańcucha o długości dziewięciu znaków (szczegóły opisano w p. 1.4.3). Na wyjściu *read buffer* utworzono wskaźnik *read buffer ASCII* (*Create* → *Indicator*), który na Panelu będzie pokazywał wynik pomiaru temperatury odczytany z portu szeregowego.

Jako ostatnią użyto funkcję *VISA Close* do zamknięcia portu szeregowego. Na jej wyjściu *error out* utworzono wskaźnik pokazujący na Panelu ewentualne błędy w transmisji.

Aby wszystkie użyte funkcje były wykonane w odpowiedniej kolejności (konfiguracja portu → wysłanie danych → zamknięcie portu) połączono ze sobą odpowiednio wszystkie wejścia *VISA resource name*, *error in* i wyjścia: *VISA resource name out*, *error out* kolejnych funkcji.

Na rys. 27 przedstawiono przykładowy Panel omawianego programu. Prezentowane są na nim dane wysyłane przez port szeregowy (*Kod sterujący ASCII*), ewentualne błędy transmisji (*error out*) oraz wynik pomiaru temperatury (*read buffer ASCII*). Po uruchomieniu programu przyciskiem  moduł ADAM-4013 powinien wykonać pomiar temperatury, a wynik tego pomiaru powinien pokazać się na Panelu programu.



Rys. 28. Panel programu do pomiarów temperatury modułem ADAM-4013

Przedstawiony program jest bardzo uproszczony i jego możliwości są niewielkie. Przede wszystkim odczytany i przedstawiony na Panelu wynik pomiaru temperatury jest zmienną tekstową, a więc nie można na niej zrealizować żadnych obliczeń, nie można wykonać wykresu, porównywać wartości z innymi zmiennymi itp. Aby uzyskać większą funkcjonalność programu należy przede wszystkim zastosować odpowiednią konwersję typów zmiennych z tekstowego na numeryczny. Należy przy tym pamiętać, że wynik pomiaru otrzymany z modułu ADAM-4013 zawiera kropkę rozdzielającą część całkowitą od ułamkowej w zapisie dziesiętnym, podczas gdy w komputerze z zainstalowaną polską wersją systemu Windows oczekiwany jest znak rozdzielający w postaci przecinka. Wymaga to szczególnej uwagi podczas konwersji wyniku pomiaru z postaci tekstowej na liczbową. Zagadnienie to będzie omówione w dalszej części instrukcji.

3. Ogólne zasady realizacji ćwiczenia

Ćwiczenie opiera się na wykorzystaniu środowiska LabVIEW, sieci modułów serii ADAM-4000 z interfejsem RS-485 oraz modelu procesu technologicznego. Dokumentacja techniczna modułów ADAM-4000 stanowi załącznik do instrukcji [8]. Studenci samodzielnie powinni zaopatrzyć się w literaturę wspomagającą naukę programowania w LabVIEW. Przykładowe pozycje, w tym dostępne w Internecie darmowe podręczniki, zamieszczono w wykazie literatury.

3.1. Organizacja systemu plików na komputerze

Przystępując do realizacji ćwiczenia należy najpierw uruchomić komputer, a dopiero w drugiej kolejności włączyć zasilanie pozostałej aparatury. Wyłączamy aparaturę w kolejności odwrotnej.

Wszystkie pliki z programami napisanymi podczas realizacji ćwiczenia należy zapisywać w katalogu C:\LABORKA_KSP\STUDENT\RRRRMMDD_G_Z, gdzie RRRR, MM, DD oznaczają kolejno: rok, miesiąc i dzień rozpoczęcia wykonywania ćwiczenia, a G i Z oznaczają grupę i zespół laboratoryjny. Zapisywane pliki powinny posiadać nazwę utworzoną z początkowych liter nazwiska jednego z członków zespołu oraz kolejnych cyfr 1, 2, 3 itd., np.: **KOWALS_1.Vi**, **KOWALS_2.Vi** itd. Pliki utworzonych w czasie zajęć przyrządów wirtualnych do obsługi pojedynczych modułów powinny również zawierać w nazwie pliku programu symbol obsługiwanego modułu (np. 4013, 4060, RS232).

Podczas realizacji kolejnego zadania często należy wykorzystać program opracowany w ramach wcześniejszego punktu. Nie wolno jednak w tym celu modyfikować programu będącego finalną wersją wcześniej zrealizowanego zadania. Po zapisaniu finalnej wersji programu realizującego w pełni określony punkt ćwiczenia, należy wykonać jego kopię nadając mu nową nazwę, zmieniając końcową cyfrę na kolejną wartość. Kopię programu należy wykonywać opcją *File\Save As*, wybierając następnie opcję *Copy - create copy on disk\Substitute copy for original (Copy will be in memory. Original will be closed)*. Pracując na tak utworzonej kopii programu należy ją uzupełniać o nowe elementy, aż do zrealizowania kolejnego punktu ćwiczenia.

Wszystkie realizowane podczas zajęć programy powinny zawierać dane osobowe zespołu laboratoryjnego, numer grupy dziekańskiej oraz datę wykonywania ćwiczenia. Informacje te należy podać w postaci **komentarza umieszczonego na panelu sterującym i na diagramie**.

W każdym realizowanym punkcie oczekiwana jest inwencja własna studentów. Wydruki programów oraz przykładowe pliki udostępniane ćwiczącym należy traktować wyłącznie jako przykłady, a nie jako obowiązujące wzorce. Przykłady te zazwyczaj realizują zadania stawiane w ćwiczeniach tylko w ograniczonym zakresie, **mniejszym od wymaganego**.

3.3. Wydruk dokumentacji programu

Środowisko LabVIEW pozwala w prosty sposób utworzyć na dysku twardym komputera pliki z dokumentacją zrealizowanego przyrządu wirtualnego. Pliki będą zawierać obraz Panelu oraz Diagramu. Kolejność postępowania jest następująca:

- wybrać opcję **File/Print..** i w oknie **Select VI(s)** zaznaczyć nazwę aplikacji do wydruku, wcisnąć **NEXT**,
- w oknie **Print Contents** zaznaczyć opcję **VI documentation**, wcisnąć **NEXT**,
- w oknie **VI Documentation** zaznaczyć opcje: **Front Panel, Controls (connected Controls), Descriptions, Data type information, Label, Block Diagram**, wcisnąć **NEXT**,
- w oknie **Destination** wybrać opcję **HTML File**, wcisnąć **NEXT**,

- w oknie **HTML** wybieramy **Image format: GIF (uncompressed), color depth: 256 colors**, wcisnąć **SAVE**,
 - w oknie **SAVE** wybrać katalog (jeśli go jeszcze nie ma, to należy go utworzyć zgodnie z podaną wcześniej regułą). Zapisać plik.
 - Odszukać zapisane pliki na dysku i sprawdzić ich zawartość.
- Zanotować w protokole nazwę utworzonego katalogu i nazwy zapisanych w nim plików z opisem zawartości.**

3.4. Zapisywanie wykresów do plików graficznych

Aby zapisać do pliku dyskowego uzyskany w programie na wykresie przebieg sygnału, należy wykonać kolejno następujące czynności:

- ustawić kursor myszki na oknie przebiegu sygnału i kliknąć prawym przyciskiem myszki,
- z otworzonego menu wybrać opcję **Export Simplified Image**, zaznaczyć opcję **Bitmap (BMP)** i **Save to file**,
- wybrać katalog utworzony przez grupę laboratoryjną na początku zajęć i wpisać nazwę pliku odpowiednio do zawartości, zatwierdzić **OK** i zapisać **Save**,
- sprawdzić zawartość pliku i zanotować w protokole nazwę pliku z zapisanym przebiegiem i opisać jego zawartość.

3.5. Przekształcanie przyrządu wirtualnego w podprogram

Środowisko LabVIEW umożliwia przekształcenie napisanego samodzielnie programu w podprogram, który może być potem wykorzystywany w innych aplikacjach w ten sam sposób, jak wszystkie inne elementy dostępne w palecie *Functions*. W tym celu należy utworzyć ikonę, której obraz będzie identyfikował na diagramie dany podprogram, oraz zdefiniować konektory, które będą służyć jako wejścia i wyjścia danych w diagramie programu. Kolejność postępowania jest następująca:

- otworzyć w LabVIEW program, który chcemy przekształcić w podprogram,
- przełączyć się na widok Panelu,
- w prawym górnym rogu okna Panel znajduje się ikona utworzonego programu, należy umieścić kursor na tej ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać „*Edit Icon*”,
- dokonać edycji ikony w sposób umożliwiający łatwą identyfikację podprogramu po umieszczeniu go na diagramie,
- ponownie umieścić kursor na ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać opcję „*Show Connector*”. W prawym górnym rogu zostanie wyświetlona siatka wejść i wyjść programu,
- ponownie umieścić kursor na ikonie i prawym kliknięciem rozwinąć listę opcji, wybrać opcję „*Patterns*” i z dostępnych wzorów zacisków wybrać konfigurację odpowiadającą potrzebom tworzonego podprogramu. W miarę potrzeby dla zwiększenia czytelności połączeń można siatkę konektorów obrócić o 90 stopni opcją *Rotate 90 Degrees*,
- w celu przypisania wejść i wyjść danych do konektorów należy kliknąć na wybrany konektor (zmieni on kolor) i następnie kliknąć w oknie Panelu na kontrolkę reprezentującą wybrane wejście lub wyjście danych w naszym podprogramie. Konektor zmieni kolor odpowiednio do typu danych z nim powiązanych,
- przypisać kolejno wszystkie konektory do wejść i wyjść podprogramu,
- ponownie umieścić kursor na ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać opcję „*Show Icon*”.
- ponownie umieścić kursor na ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać opcję „*VI properties*”. W celu utworzenia opisu podprogramu w widoku Panelu wybrać opcję *Documentation* i wpisać w polu "*VI description*" tekst opisujący przeznaczenie programu, który będzie pojawiał się w okienku helpu kontekstowego,

- zapisać program do pliku dyskowego i zamknąć,

W celu wywołania utworzonego podprogramu należy w kolejnym tworzonym nowym programie przejść do okna Diagramu i z palety „*Functions*” wybrać „*Select a VI ..*”, odnaleźć na dysku komputera wcześniej utworzony podprogram i wstawić do diagramu tworzonego nowego programu.

4. Wykonanie ćwiczenia

4.1. Uruchomienie stanowiska ćwiczeniowego



- Zgodnie z zaleceniami podanymi w rozdziale 3 uruchomić stanowisko: włączyć komputer i poczekać na uruchomienie systemu operacyjnego. Uruchomić środowisko LabView. Zanotować do protokołu konfigurację komputera: wersję systemu operacyjnego, typ procesora, częstotliwość taktowania, ilość pamięci RAM, wielkość dysku twardego, wersję LabVIEW. Odczytać z obudowy typy modułów zainstalowanych na stanowisku i zapisać do protokołu.




- Zapoznać się z konstrukcją stanowiska i porównać ze schematami przedstawionymi na rysunkach 1, 2, 3, 4. Włączyć zasilanie tablicy z modułami ADAM-4000 i tablicy z modelem procesu przemysłowego.




- Utworzyć na dysku komputera katalog do zapisywania programów i pozostałych danych zgodnie z zasadą przedstawioną w punkcie 3.1. Zanotować nazwę utworzonego katalogu do protokołu.

4.2. Programowanie obsługi portu RS-232C

- Przygotować program obsługujący transmisję danych portem szeregowym RS-232C według punktu 2.2. Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1.

- Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie powinien się zatrzymać.



- Uruchomić program przyciskiem , zwrócić uwagę, czy pojawiły się ikony  . Ocenić poprawność działania programu i usunąć ewentualne błędy. Zanotować do protokołu odpowiedź wysłaną przez moduł ADAM-4013. Odszukać w dokumentacji [8] informacje umożliwiające rozkodowanie poszczególnych pól informacyjnych i zapisać do protokołu odczytaną konfigurację modułu ADAM-4013.

- Przełączyć się na okno Diagramu i za pomocą ikony żarówki uruchomić podgląd przepływu danych w diagramie, zwrócić uwagę na zmianę szybkości działania programu z włączonym podglądem przepływu danych i zapisać wniosek do protokołu. Zatrzymać działanie programu przyciskiem , zwrócić uwagę, czy ponownie pojawiły się ikony  . Wyłączyć podgląd przepływu danych w diagramie.







- Jeśli program działa poprawnie, wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

4.3. Programowanie sterowania modułem ADAM-4060

- Przygotować program sterujący modułem ADAM-4060 według punktu 2.3. Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1.

- Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie

powinien się zatrzymać. Sprawdzić, czy zadziałał odpowiedni przełącznik i zapaliła się odpowiednia dioda LED. Ocenic poprawność działania programu i usunąć ewentualne błędy. Zmienić wartość danych programujących moduł i sprawdzić działanie innych przełączników. Zanotować do protokołu postać wysyłanych komend i odpowiadające im zachowanie się przełączników oraz diod LED.

- Uruchomić program przyciskiem , zwrócić uwagę, czy pojawiły się ikony  . Przełączyć się na okno Diagramu i za pomocą ikony żarówki uruchomić podgląd przepływu danych w diagramie, zwrócić uwagę na zmianę szybkości działania programu z włączonym podglądem przepływu danych. Zatrzymać działanie programu przyciskiem , zwrócić uwagę, czy ponownie pojawiły się ikony  . Wyłączyć podgląd przepływu danych w diagramie.

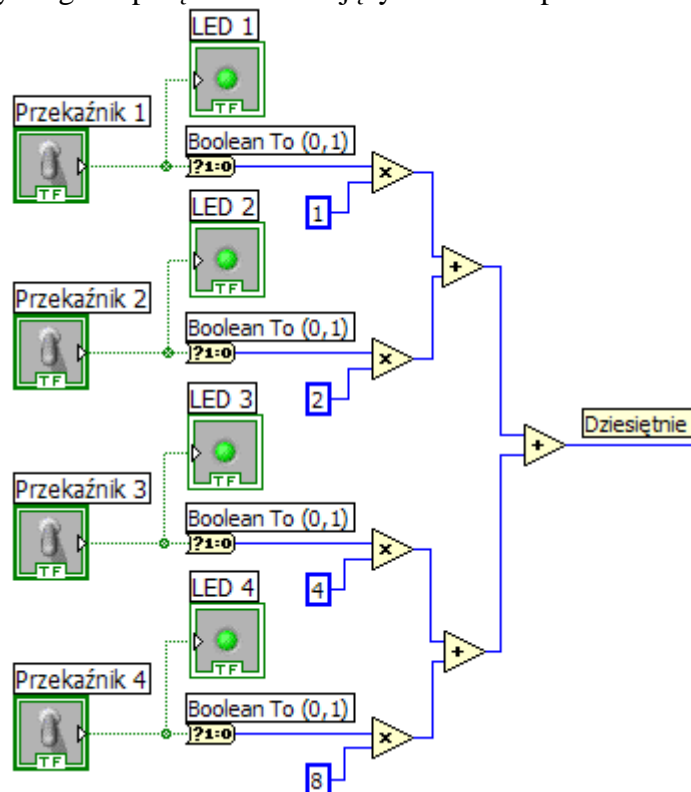
- Jeśli po zakończeniu eksperymentów z programem pozostała załączona grzałka, to należy wyłączyć na chwilę zasilanie tablicy z modułami ADAM-4000, aby skasować stan wszystkich przełączników.

- Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.

- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

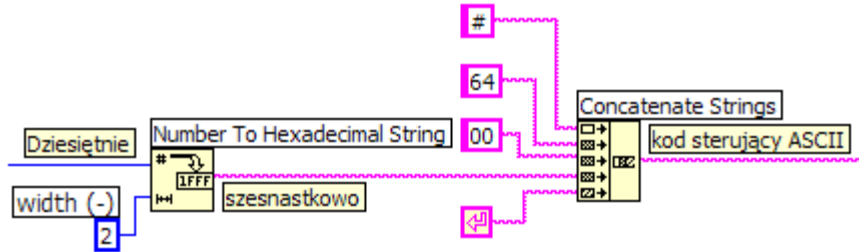
- Wykonać kopię programu na potrzeby realizacji kolejnego punktu (wg. p. 3.1).

- Uzupełnić kopię utworzonego w poprzednim punkcie programu o możliwość sterowania przełącznikami modułu ADAM-4060 za pomocą wirtualnych przełączników umieszczonych na Panelu programu. W tym celu należy na Panelu umieścić cztery przełączniki bistabilne i cztery lampki sygnalizujące stan przełączników. Dane logiczne z wyjść przełączników należy przekonwertować na dane liczbowe (0, 1) za pomocą funkcji *Boolean To (0,1)*, przemnożyć przez kolejne potęgi liczby 2 i zsumować ze sobą. W ten sposób otrzymamy daną liczbową zawierającą stan wszystkich przełączników zakodowany na kolejnych bitach jednego bajta danych. Przykładowy diagram połączeń realizujący to zadanie przedstawiono na rys. 29.



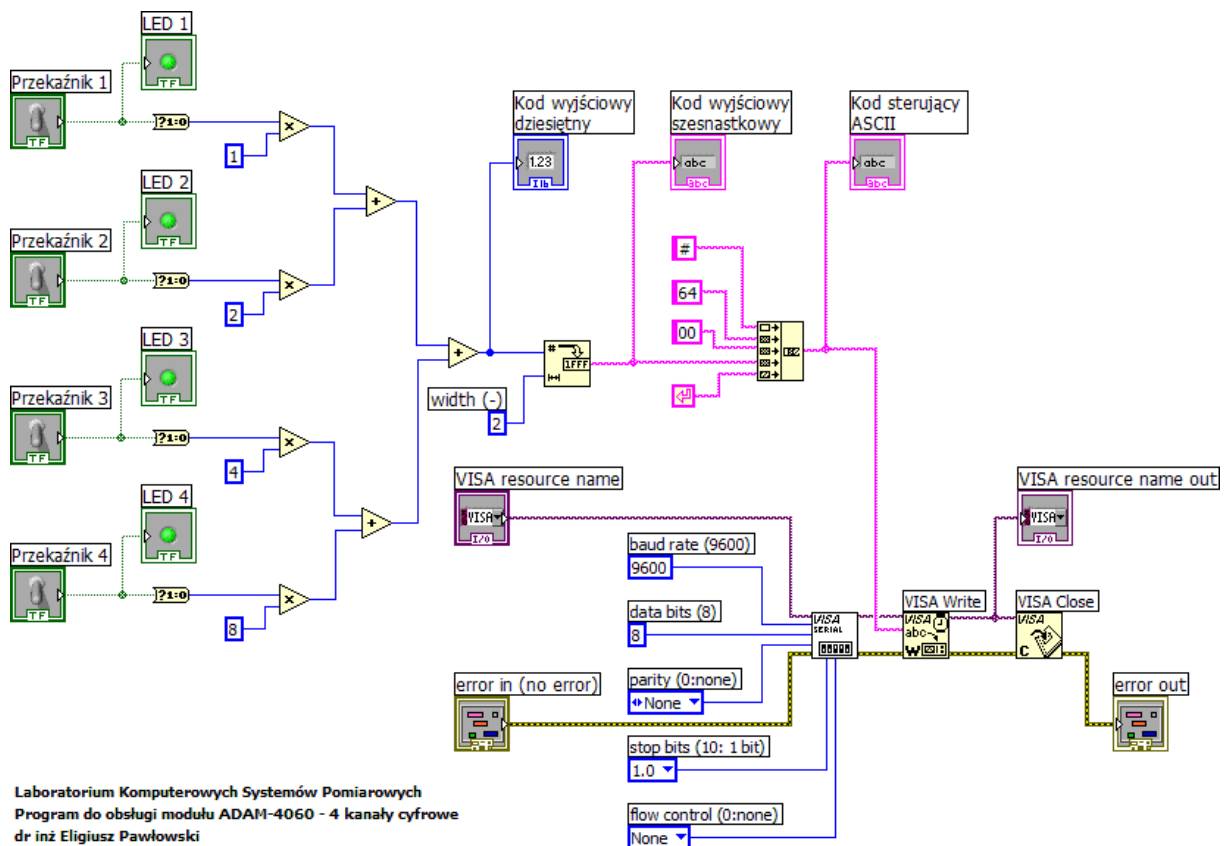
Rys. 29. Przetwarzanie stanów przełączników na daną numeryczną

Następnie należy przekonwertować otrzymaną daną liczbową na dwuznakowy łańcuch tekstowy reprezentujący liczbę szesnastkową za pomocą funkcji *Number To Hexadecimal String*. Następnie należy utworzyć ośmioznakowy łańcuch stanowiący kompletną komendę sterującą *Digital Data Out*, która jest szczegółowo omówiona w p. 1.4.2. Składnia komendy podana jest na rys. 13. Przykładowy diagram połączeń realizujący to zadanie przedstawiono na rys. 30.



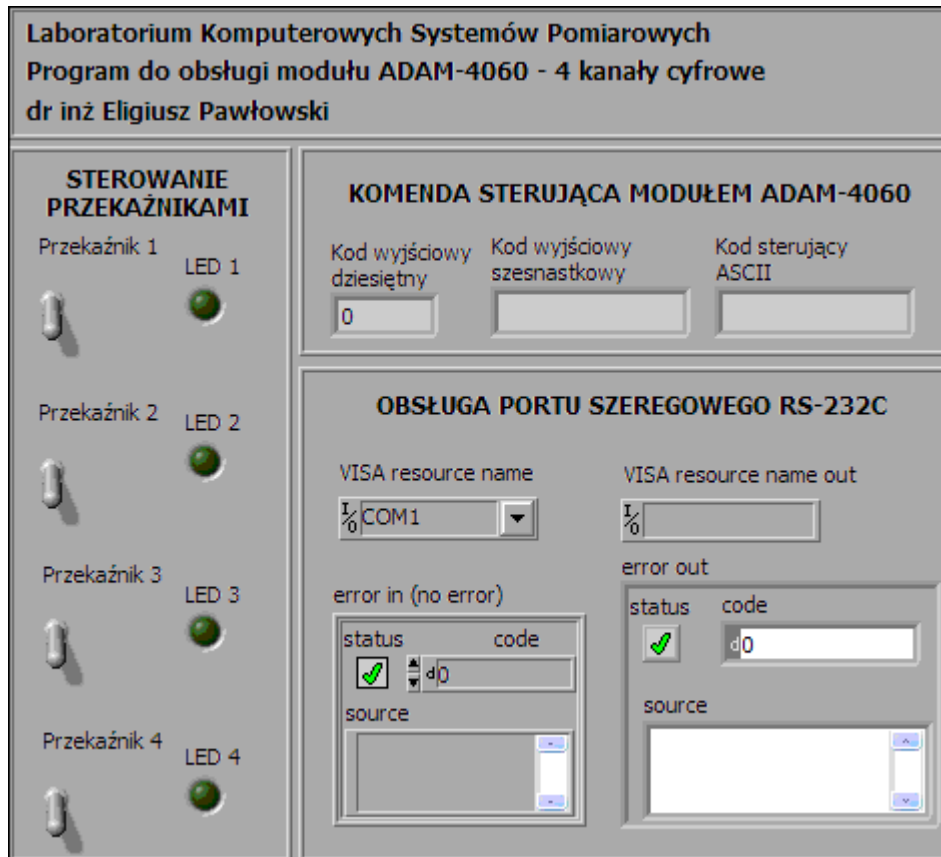
Rys. 30. Przetwarzanie liczby dziesiętnej na szesnastkową i na kod sterujący ASCII

Ostatecznie otrzymany kod sterujący ASCII należy dołączyć do wejścia *write buffer* funkcji *VISA Write*, zamiast łańcucha *Hexadecimal output data*. Uzupełnić program o wskaźniki pokazujące na Panelu kolejne etapy tworzenia komendy sterującej: daną numeryczną utworzoną na podstawie stanów przełączników, liczbę w zapisie szesnastkowym i kod sterujący ASCII. Zamienić stałą *VISA resource name* na wejściu funkcji *VISA Configure Serial Port* na zadajnik umieszczony na Panelu (prawym kliknięciem wybrać opcję *Change to Control*). Utworzyć zadajnik na wejściu *error in* funkcji *VISA Configure Serial Port* oraz wskaźnik *VISA resource name out* na wyjściu funkcji *VISA Write*. Przykładowy diagram programu realizujący wszystkie te zadania przedstawiono na rys. 31, a jego panel rys. 32.



Laboratorium Komputerowych Systemów Pomiarowych
 Program do obsługi modułu ADAM-4060 - 4 kanały cyfrowe
 dr inż Eligiusz Pawłowski

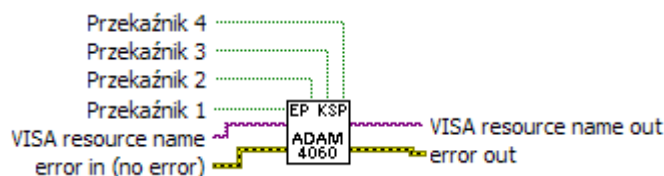
Rys. 31. Diagram programu sterującego modułem ADAM-4060



Rys. 32. Panel programu sterującego modułem ADAM-4060



- Uruchomić program i sprawdzić poprawność jego działania. Usunąć ewentualne błędy. Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.

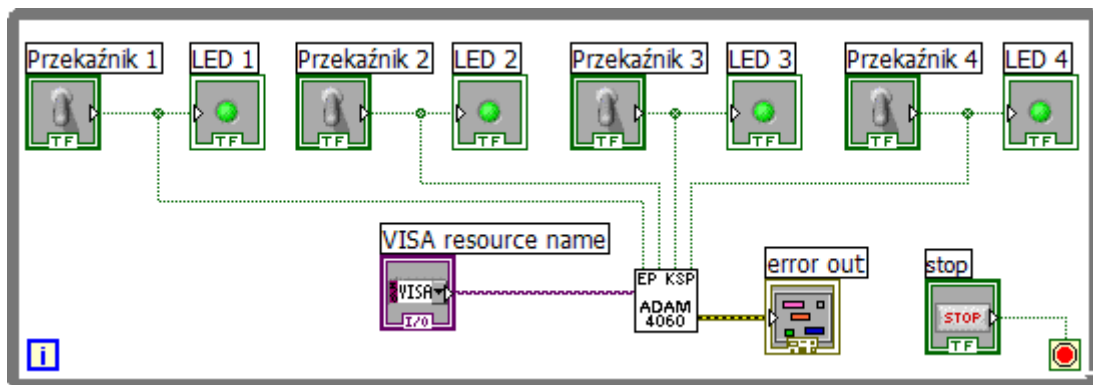
- Przekształcić otrzymany program w przyrząd wirtualny – podprogram sterujący modułem ADAM-4060. Sposób postępowania opisano w p. 3.5. W tym celu należy przeprowadzić edycję ikony podprogramu oraz przypisać zaciski wejścia – wyjścia siatki konektorów do odpowiednich elementów na Panelu. Potrzebnych będzie 8 konektorów: 6 wejściowych i 2 wyjściowe. Do konektorów wejściowych należy dołączyć cztery przełączniki oraz zadajniki *VISA resource name* i *error in*. Do konektorów wyjściowych należy dołączyć wskaźniki *VISA resource name out* i *error out*. Przykładowy wygląd podprogramu przedstawiono na rys. 31.



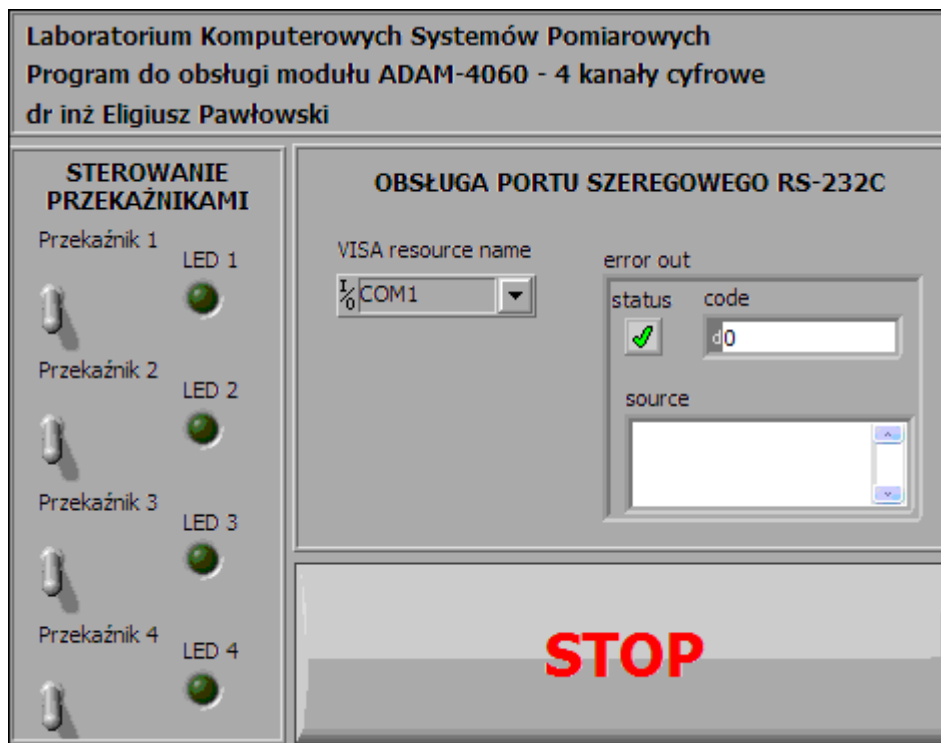
Rys. 31. Podprogram sterujący modułem ADAM-4060

- Po wprowadzonych zmianach zapisać ponownie podprogram na dysku (pod tą samą nazwą), od tej chwili podprogram jest gotowy do użycia w innych programach.
- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.
- Wykonać kopię programu na potrzeby realizacji kolejnego punktu (wg. p. 3.1).

- Zmodyfikować kopię utworzonego w poprzednim punkcie programu w ten sposób, aby wykorzystać utworzony podprogram i sprawdzić jego działanie. W tym celu należy na Panelu pozostawić tylko przyciski sterujące, lampki sygnalizacyjne i elementy związane z obsługą portu szeregowego. Pozostałe elementy na Panelu i na Diagramie należy usunąć. W ich miejsce należy wstawić utworzony w poprzednim punkcie podprogram. Dołączyć do niego obiekty sterujące pozostawione na Panelu. Całość diagramu objąć pętlą *While Loop* z palety *Function Structures*. Na wejściu *Loop Condition* (znak STOPu w rogu pętli) wygenerować prawym kliknięciem zadajnik (*Create Control*). Uporządkować Diagram i Panel. Uruchomić program przyciskiem . Sprawdzić poprawność działania programu. Zatrzymać program przyciskiem STOP na panelu (nie używać przycisku ). Przykładowa postać Diagramu i Panelu przedstawiana jest na rys. 32 i 33.





Rys. 32. Diagram programu wykorzystujący utworzony podprogram ADAM-4060

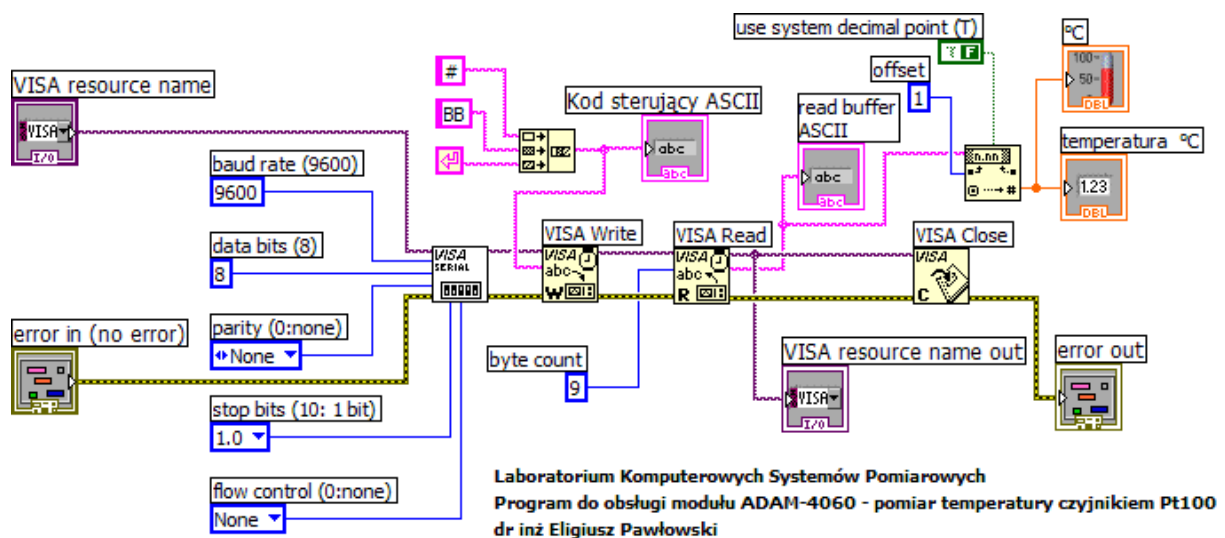


Rys. 33. Panel programu wykorzystującego podprogram ADAM-4060

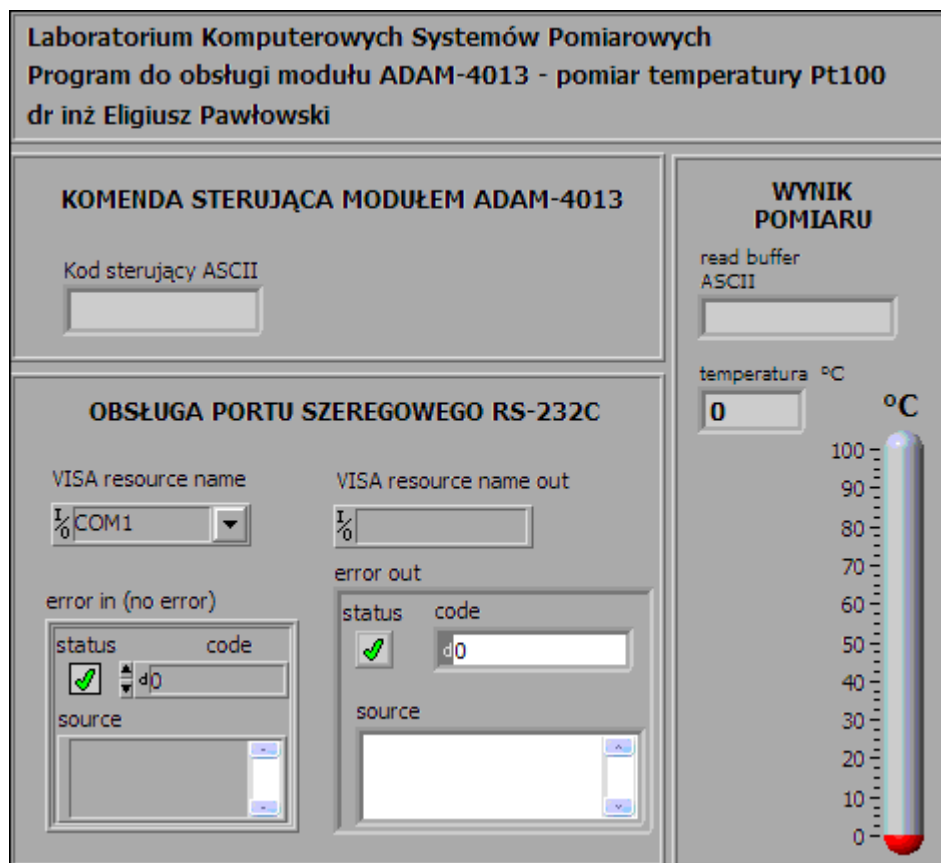
- Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.
- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

4.4. Programowanie pomiarów temperatury modułem ADAM-4013

- Przygotować program realizujący pomiar temperatury za pomocą modułu ADAM-4013 według punktu 2.4. Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1.
- Uruchomić program przyciskiem . Ocenić poprawność działania programu i usunąć ewentualne błędy. Zatrzymać działanie programu przyciskiem .
- Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.
- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.
- Wykonać kopię programu na potrzeby realizacji kolejnego punktu (wg. p. 3.1).
- Uzupełnić kopię utworzonego w poprzednim punkcie programu o konwersję odebranego z modułu ADAM-4013 wyniku pomiaru z postaci tekstowej na zmienną numeryczną. W tym celu należy wykorzystać funkcję *Fract/Exp String To Number* z palety *Function* → *Programming* → *String* → *String/Number Conversion*. Na wejście *offset* należy podać stałą numeryczną 1, na wejście *use system decimal point* podać wartość logiczną *False*. Na wyjściu funkcji należy utworzyć wskaźnik numeryczny oraz wskaźnik typu termometr. Zamienić stałą *VISA resource name* na wejściu funkcji *VISA Configure Serial Port* na zadajnik umieszczony na Panelu (prawym kliknięciem wybrać opcję *Change to Control*). Utworzyć zadajnik na wejściu *error in* funkcji *VISA Configure Serial Port* oraz wskaźnik *VISA resource name out* na wyjściu funkcji *VISA Read*. Przykładowy diagram programu realizujący wszystkie te zadania przedstawiono na rys. 34, a jego panel rys. 35.



Rys. 34. Diagram programu do pomiaru temperatury modułem ADAM-4013



Rys. 35. Panel programu do pomiaru temperatury modułem ADAM-4013



- Uruchomić program i sprawdzić poprawność jego działania. Usunąć ewentualne błędy. Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.

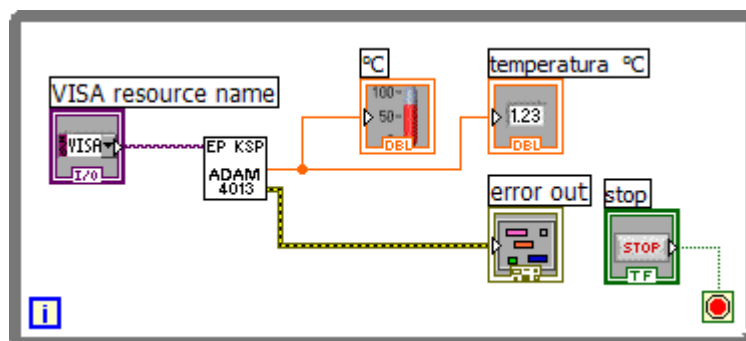
- Przekształcić otrzymany program w przyrząd wirtualny – podprogram do pomiaru temperatury modułem ADAM-4013. Sposób postępowania opisano w p. 3.5. W tym celu należy przeprowadzić edycję ikony podprogramu oraz przypisać zaciski wejścia – wyjścia siatki konektorów do odpowiednich elementów na Panelu. Potrzebnych będzie 6 konektorów: 2 wejściowe i 4 wyjściowe. Do konektorów wejściowych należy dołączyć zadajniki *VISA resource name* i *error in*. Do konektorów wyjściowych należy dołączyć wskaźnik pokazujący wynik pomiaru temperatury, wskaźnik pokazujący odebrany kod ASCII oraz wskaźniki *VISA resource name out* i *error out*. Przykładowy wygląd podprogramu przedstawiono na rys. 36.



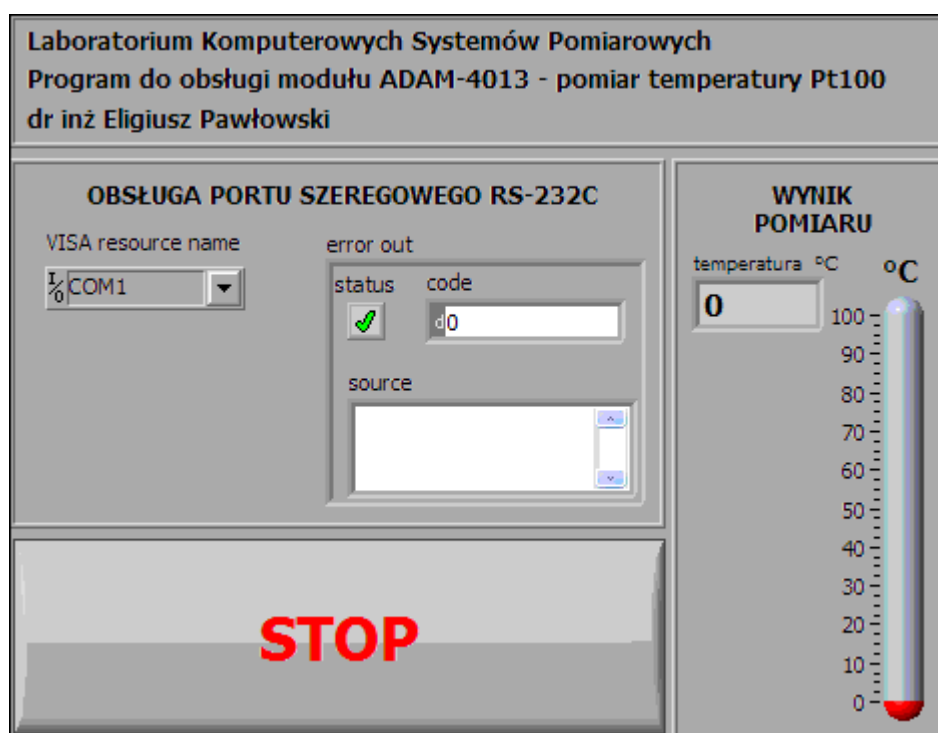
Rys. 36. Podprogram do pomiaru temperatury modułem ADAM-4013

- Po wprowadzonych zmianach zapisać ponownie podprogram na dysku (pod tą samą nazwą), od tej chwili podprogram jest gotowy do użycia w innych programach.
- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.
- Wykonać kopię programu na potrzeby realizacji kolejnego punktu (wg. p. 3.1).
- Zmodyfikować kopię utworzonego w poprzednim punkcie programu w ten sposób, aby wykorzystać utworzony podprogram i sprawdzić jego działanie. W tym celu należy na Panelu

pozostawić tylko wskaźniki pokazujące wynik pomiaru temperatury i elementy związane z obsługą portu szeregowego. Pozostałe elementy na Panelu i na Diagramie należy usunąć. W ich miejsce należy wstawić utworzony w poprzednim punkcie podprogram. Dołączyć do niego obiekty sterujące pozostawione na Panelu. Całość diagramu objąć pętlą *While Loop* z palety *Function Structures*. Na wejściu *Loop Condition* (znak STOPu w rogu pętli) wygenerować prawym kliknięciem zadajnik (*Create Control*). Uporządkować Diagram i Panel. Uruchomić program przyciskiem . Sprawdzić poprawność działania programu. Zatrzymać program przyciskiem STOP na panelu (nie używać przycisku ). Przykładowa postać Diagramu i Panelu przedstawiana jest na rys. 37 i 38.



Rys. 37. Diagram programu wykorzystujący utworzony podprogram ADAM-4013



Rys. 38. Panel programu wykorzystującego podprogram ADAM-4013

- Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.
- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

4.5. Programowanie układu automatycznej regulacji temperatury

- Przygotować program realizujący pomiar temperatury za pomocą modułu ADAM-4013 i automatyczne sterowanie grzałką oraz wentylatorem za pomocą modułu ADAM-4060 w celu utrzymania w obiekcie zadanej wartości temperatury. Należy zastosować utworzone w poprzednich punktach podprogramy do sterowania modułami 4013 i 4060. Na Panelu należy umieścić zadajniki do zadawania temperatury minimalnej i maksymalnej, wskaźnik pokazujący aktualną temperaturę, lampki sygnalizujące aktualny stan przełączników sterujących grzałką i wentylatorem oraz wykres *Waveform Chart* pokazujący historię zmierzonej temperatury i zdanych wartości. Program powinien włączać grzałkę gdy zmierzona temperatura jest niższa od zadanej wartości minimalnej oraz włączać wentylator gdy zmierzona temperatura jest wyższa od zadanej wartości maksymalnej. Odpowiednią kolejność wykonywania podprogramów należy ustalić łącząc ze sobą odpowiednio wejścia i wyjścia *VISA resource name in*, *VISA resource name out*, *error in*, *error out*. Całość algorytmu należy umieścić w pętli *While Loop* i dodać przycisk STOP wyłączający działanie programu. Przykładowa postać Diagramu i Panelu przedstawiana jest na rys. 38 i 39.

- Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1. Uruchomić i sprawdzić działanie programu. Ocenić poprawność działania programu i usunąć ewentualne błędy.

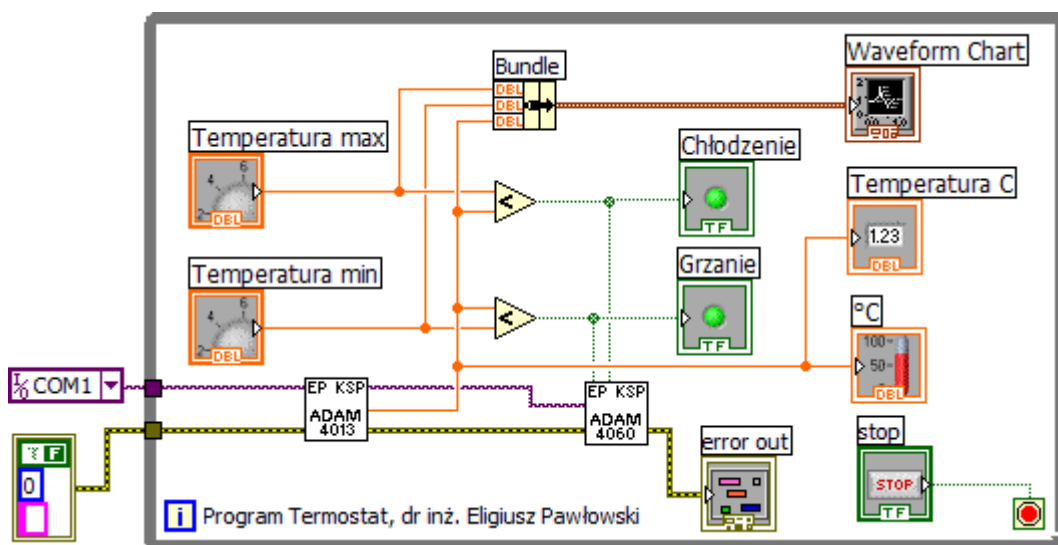
- Jeśli program działa poprawnie, zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.

- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

- Uruchomić program i zadać samodzielnie wybrane wartości temperatury minimalnej i maksymalnej. Zwrócić uwagę na wartości temperatury zadanej i uzyskiwanej na obiekcie, czasy przekroczenia zadanych wartości, szczególnie podczas stanu przejściowego po zmianie temperatury zadanej.

- Zapisać do pliku dyskowego otrzymany wykres zmian zmierzonej temperatury postępując zgodnie z opisem w p. 3.4.

- Zaproponować i zrealizować algorytm sterowania temperaturą pozwalający uzyskać lepsze parametry. Uzupełnić program o dodatkowe możliwości według własnego pomysłu, np. histogram wartości temperatury, zapis wyników do pliku dyskowego, reakcja na błędnie zadane wartości (minimum > maksimum), automatyczne wyłączenie grzałki przy zamykaniu programu, możliwość ręcznego sterowania grzałką i wentylatorem itp.



Rys. 39. Diagram programu do automatycznej regulacji temperatury



Rys. 40. Panel programu do automatycznej regulacji temperatury

5. Sprawozdanie ze zrealizowanego ćwiczenia

Każda grupa przygotowuje z ćwiczenia jedno całościowe sprawozdanie zawierające:

- stronę tytułową z danymi osobowymi zespołu, datą wykonania ćwiczenia i jego tytułem,
- opis wykorzystywanego środowiska ćwiczeniowego z podaniem wykazu wykorzystywanej aparatury oraz zestawieniem jej parametrów,
- schemat wykorzystywanego układu pomiarowego,
- tematy kolejnych zadań przewidzianych programem ćwiczenia,
- opis słowny każdego ze zrealizowanych zadań lub podanie przyczyny niezrealizowania,
- dokumentację samodzielnie opracowanych i skutecznie uruchomionych programów (niezbędne są **opisy na panelu** oraz **komentarze na diagramie** programu !!!),
- schematy blokowe algorytmów dla zrealizowanych **samodzielnie programów**,
- słowny opis zrealizowanych **samodzielnie** algorytmów,
- wyniki pomiarów uzyskane za pomocą przygotowanych programów,
- wnioski ze zrealizowanych ćwiczeń, a w szczególności: napotkane trudności i przyjęte sposoby ich rozwiązania, propozycje zmian i rozszerzeń programu realizowanych zadań, propozycje dalszych modyfikacji i ulepszeń realizowanego zadania, propozycje nowych tematów zadań, propozycje modyfikacji i rozszerzeń układu pomiarowego.

W sprawozdaniu ocenie podlegają:

- Poprawność działania programu oraz jego odporność na nietypowe sygnały i złą obsługę.
- Czytelność i przejrzystość diagramu,
- Estetyka utworzonego Panelu, dobór kolorystyki, elementów graficznych, wyrównanie położenia obiektów, funkcjonalność panelu pod kątem obsługi,
- Zakres zrealizowanych zadań,
- Trafność wniosków końcowych.

Uwaga! Nie należy stosować systemowej funkcji Print Screen (przycisk PrtScr na klawiaturze) do zapisywania Panelu i Diagramu na dysku jako dokumentacji programu. Sprawozdania zawierające zamiast dokumentacji programu wydruk okna Windows z Panelem i Diagramem uzyskane klawiszem PrtScr będą oceniane słabiej. Można i należy natomiast

wykonywać za pomocą funkcji PrtScr kopie Panelu w celu udokumentowania sposobu działania zrealizowanego programu

6. Literatura

1. Nawrocki W.: Komputerowe systemy pomiarowe, WKiŁ, Warszawa 2002.
2. Winiecki W.: Organizacja komputerowych systemów pomiarowych, Oficyna Wydawnicza PW, Warszawa 1997.
3. Świsulski D.: Komputerowa technika pomiarowa Oprogramowanie wirtualnych przyrządów pomiarowych w LabView, Wyd. PAK, Warszawa 2005.
4. Tłaczała W.: Środowisko LabVIEW w eksperymencie wspomaganym komputerowo, WNT Warszawa 2002.
5. Chruściel M.: LabVIEW w praktyce, Wyd. BTC, Warszawa 2008.
6. Witold Kaczurba: [Kurs dla początkujących w Labview](http://www.kaczurba.pl), <http://www.kaczurba.pl>.
7. National Instruments: LabVIEW Tutorial Manual.
8. ADAM-4000 series. Data acquisition Modules. User's Manual, Advantech Co., Ltd.