LabVIEW

PLATFORMA EDUKACYJNA

Lekcja 6

LabVIEW i Arduino – programy wykorzystujące wyświetlacz LCD, czujnik temperatury, PWM i diodę LED

Przygotował: Jakub Wawrzeńczak

1. Wprowadzenie

Lekcja przedstawia wykorzystanie środowiska LabVIEW 2016 w wersji 32 bitowej do programowania modułu Arduino UNO R3. W tym kursie będą przedstawione przykładowe programy wykorzystujące wyświetlacz ciekłokrystaliczny i analogowy czujnik temperatury oraz diodę LED sterowaną przez PWM przy wykorzystaniu środowiska LabVIEW oraz płytki rozwojowej Arduino.

2. Program do wyświetlania temperatury

Przedstawiony w tym punkcie program będzie odczytywał temperaturę z analogowego czujnika LM 35 i wyświetlał ją na wyświetlaczu LCD.

Pierwszym krokiem jest zmontowanie układu. Pierwszym elementem jest wyświetlacz LCD. W poniższej tabeli są zestawione piny wyświetlacza oraz gdzie je podłączyć:

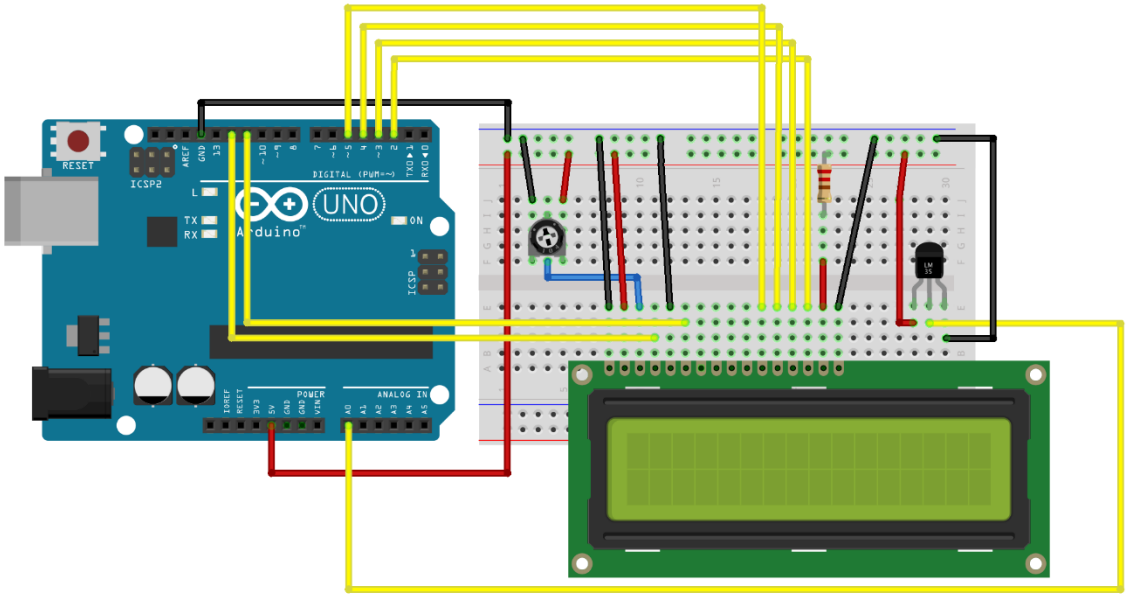
LCD		
1	GND	GND Arduino
2	VCC	5V Arduino
3	VEE	Potencjometr
4	RS	Pin 12 Arduino
5	R/W	GND Arduino
6	EN	Pin 11 Arduino
7	DB0	-
8	DB1	-
9	DB2	-
10	DB3	-
11	DB4	Pin 5 Arduino
12	DB5	Pin 4 Arduino
13	DB6	Pin 3 Arduino
14	DB7	Pin 2 Arduino
15	BackLight +	5V Arduino
16	BackLight -	GND Arduino

Kolejnym elementem jest analogowy czujnik temperatury LM 35. W poniższej tabeli są zestawione piny czujnika oraz gdzie je podłączyć:

LM 35		
1	VCC	5V Arduino
2	Output	A0 Arduino
3	GND	GND Arduino

Oprócz powyższych elementów potrzebny jest potencjometr 10 kΩ oraz jeden rezystor 220 Ω, który jest włączony pomiędzy pinem numer 15 wyświetlacza a pinem 5V Arduino.

Układ po zmontowaniu powinien wyglądać jak na rys. 1:



Rys. 1 – Schemat połączeń (LCD, LM 35, Arduino)

Kolejnym krokiem jest przystąpienie do napisania programu w środowisku LabVIEW. Po utworzeniu nowego projektu należy przejść do okna *Block Diagram* i dodać element *Init* z palety *Functions* i zakładki *Adruino*, który pozwoli na inicjalizację programu z platformą Arduino. Po najechaniu na blok *Init* przy odpowiednich wejściach powinno wyświetlić się:
- przy drugim wejściu: *Baud Rate (115200)*
- przy trzecim wejściu: *Board Type (Uno)*
- przy czwartym wejściu: *Bytes per packet (15)*
- przy piątym wejściu: *Connection Type (USB/Serial)*

Należy teraz dodać kolejne bloki. Z palety *Functions*, zakładki *Arduino* i kategorii *Sensors* wybrać podkategorię *LCD*, w której znajduje się *LCD Configure 4-bit.* Blok ten jest odpowiedzialny za konfigurację odpowiednich pinów wykorzystanych w Arduino. Blok *Init* należy połączyć z nowo dodanym blokiem. Wyjście *Arduino Resource* bloku inicjalizacji należy połączyć z wejściem *Arduino Resource* bloku konfiguracji oraz wyjście *Error out* bloku inicjalizacji trzeba połączyć z wejściem *Error in* bloku konfiguracji. Przy tym połączeniu należy zwrócić uwagę na to, że każdy kolejny blok dotyczący Arduino trzeba łączyć w ten sam sposób, tj. wyjście *Arduino Resource* bloku poprzedzającego z wejściem *Arduino Resource* bloku następnego oraz wyjście *Error out* bloku poprzedzającego z wejściem *Error in* bloku następnego. Dodatkowo przy bloku konfiguracji trzeba zdefiniować numery wykorzystywanych pinów. Należy utworzyć nowy element *Control* poprzez kliknięcie prawym przyciskiem myszy na

wejściu *LCD Pin Config 4 Bit* i wybraniu opcji *Create*. Po wykonaniu tego kroku powinien utworzyć się *Numeric Control* w oknie aplikacji VI. Następnym blokiem jest blok *LCD Init*, który znajduje się w palecie *Functions*, zakładce *Arduino,* kategorii *Sensors* i podkategorii *LCD.* Na wejściu *Columns* utworzyć zmienną *Constant* i przypisać jej wartość 16 (ilość kolumn, z których będzie korzystał wyświetlacz) oraz na wejściu *Rows* utworzyć zmienną *Constant* i przypisać jej wartość 2 (ilość wierszy, z których będzie korzystał wyświetlacz).

W kolejnym etapie należy utworzyć pętlę *While.* Do elementu *Loop Condition* należy utworzyć element *Control*. Po dodaniu, na aplikacji VI powinien pojawić się przycisk *Stop* zatrzymujący główną pętlę programu. W pętli *While* z palety *Functions*, zakładki *Arduino*, kategorii *Sensors* i podkategorii *LCD* dodać blok *LCD Set Cursor Position*. Na wejściu *Column* utworzyć zmienną *Constant* i przypisać jej wartość 0 (jest to pierwsza kolumna wyświetlacza) oraz na wejściu *Row* utworzyć zmienną *Constant* i przypisać jej wartość 0 (jest to pierwszy wiersz wyświetlacza). Kolejny blok, *Analog Read Pin* znajduje się w palecie *Functions*, zakładce *Arduino* i kategorii *Low Level*. Na wejściu *Analog Input Pin* utworzyć zmienną *Constant* i przypisać jej wartość odpowiadającą numerowi wejścia analogowego w Arduino, czyli w tym przypadku będzie to wartość 0.

Kolejnym etapem jest konwersja napięcia z czujnika analogowego na temperaturę według poniższego wzoru:

$$temperaura = (5 * odczyt\_z\_czujnika * 100)/1024$$

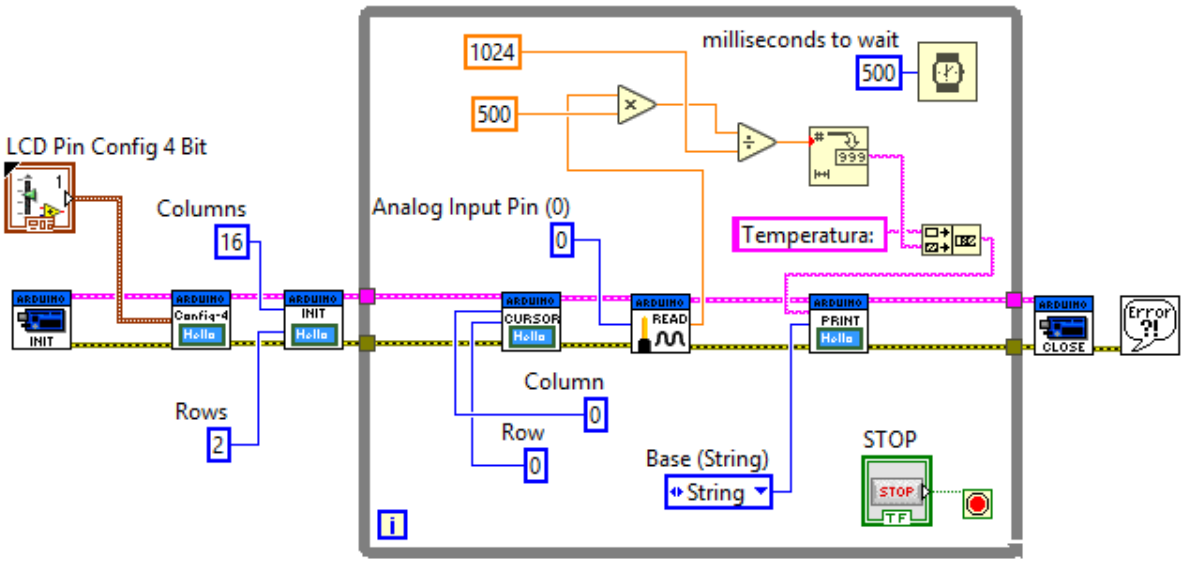
Aby wykorzystać to równanie w programie należy dodać bloki odpowiadające za mnożenie i dzielenie, które znajdują się w palecie *Functions*, zakładce *Numeric*. Na wejścia bloku mnożącego należy podać odczyt z czujnika oraz zmienną typu *Constant* z wartością 500. Na wejścia bloku odpowiadającego za dzielenie należy podać wyjście bloku mnożącego oraz zmienną typu *Constant* z wartością 1024. Otrzymaną wartość należy podać na wejście bloku, który przekonwertuje wartość liczbową na ciąg znaków. Blok *Number to Decimal String* znajduje się w palecie *Functions*, kategorii *String* i podkategorii *Number/String Conversion*. Wyjście bloku dzielącego trzeba podać na wejście bloku konwertującego zmienną. Ostatnim krokiem tego etapu jest utworzenie finalnego tekstu wyświetlanego na ekranie. Aby to wykonać należy połączyć ze sobą dwie zmienne typu *String* (jedna odpowiada wartości temperatury, natomiast drugą trzeba utworzyć i przypisać jej wartość „Temperatura: ”). Blokiem, który to umożliwia jest blok *Concatenate String*, który znajduje się w palecie *Functions* w kategorii *String*. Na jego wejścia należy podać wyjście z bloku konwertującego wartości liczbowe na zmienne typu *String*, które odpowiada wartości temperatury oraz zmienną o wartości „Temperatura: ”.

Kolejnym etapem jest dodanie bloku wyświetlającego dany napis na wyświetlaczu. Blok *LCD Print* znajduje się w palecie *Functions*, zakładce *Arduino*, kategorii *Sensors* i podkategorii *LCD*. Na wejście *Data* bloku *LCD Print* należy podać wyjście z bloku

łączącego zmienne typu *String*. Na wejściu *Base* należy utworzyć zmienną *Constant* i ustawić wartość jako *String*.

Ostatnim etapem jest dodanie opóźnienia w pętli *While* poprzez dodanie bloku *Wait (ms)* i podaniu na jego wejście wartość odpowiadającej 500 ms opóźnienia. Należy również zamknąć połączenie z modułem Arduino. Trzeba dodać blok *Close*, który znajduje się w zakładce *Arduino* w palecie *Functions.* Blok umieścić za pętlą *While* i połączyć z ostatnim blokiem dotyczącym Arduino, tj. *LCD Print*. Na wyjście *Error out* bloku *Close* należy dodać ostatni element, który można znaleźć wyszukując w palecie *Functions* po nazwie: *Simple Error Handler.vi*.

Ostateczna wersja programu w oknie *Block Diagram* powinna wyglądać jak na rys. 2:



Rys. 2 – Kod programu w oknie Block Diagram

Wygląd aplikacji VI powinien wyglądać podobnie jak na rys. 3:

Rys. 3 – Aplikacja VI

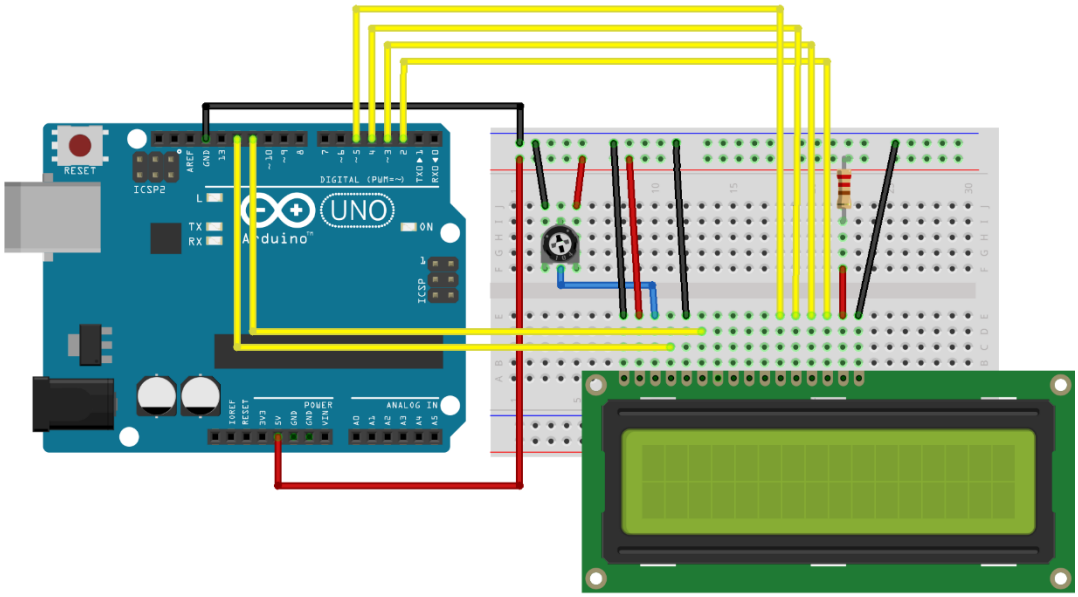
3. Program do przesuwania napisu na wyświetlaczu

Przedstawiony w tym punkcie program będzie przesuwał napis na wyświetlaczu LCD. Na potrzeby ćwiczenia zmienna typu *String* będzie napisem: „LabVIEW is great!”.

Pierwszym krokiem jest zmontowanie układu. Warto zwrócić uwagę, iż ten program również korzysta z wyświetlacza. Zaleca się więc, aby nie rozmontowywać poprzedniego układu w całości, lecz odłączyć tylko czujnik analogowy LM 35. Jeżeli układ został rozmontowany, to poniżej znajduje się zestawienie pinów oraz schemat:

LCD		
1	GND	GND Arduino
2	VCC	5V Arduino
3	VEE	Potencjometr
4	RS	Pin 12 Arduino
5	R/W	GND Arduino
6	EN	Pin 11 Arduino
7	DB0	-
8	DB1	-
9	DB2	-
10	DB3	-
11	DB4	Pin 5 Arduino
12	DB5	Pin 4 Arduino
13	DB6	Pin 3 Arduino
14	DB7	Pin 2 Arduino
15	BackLight +	5V Arduino
16	BackLight -	GND Arduino

Układ po zmontowaniu powinien wyglądać jak na rys. 4:



Rys. 4 – Schemat połączeń (LCD, Arduino)

Oprócz wyświetlacza potrzebny jest potencjometr 10 kΩ oraz jeden rezystor 220 Ω, który jest włączony pomiędzy pinem numer 15 wyświetlacza a pinem 5V Arduino.

Kolejnym krokiem jest przystąpienie do napisania programu w środowisku LabVIEW. Po utworzeniu nowego projektu należy przejść do okna *Block Diagram* i dodać element *Init* z palety *Functions* i zakładki *Adruino*, który pozwoli na inicjalizację programu z platformą Arduino. Po najechaniu na blok *Init* przy odpowiednich wejściach powinno wyświetlić się:
- przy drugim wejściu: *Baud Rate (115200)*
- przy trzecim wejściu: *Board Type (Uno)*
- przy czwartym wejściu: *Bytes per packet (15)*
- przy piątym wejściu: *Connection Type (USB/Serial)*

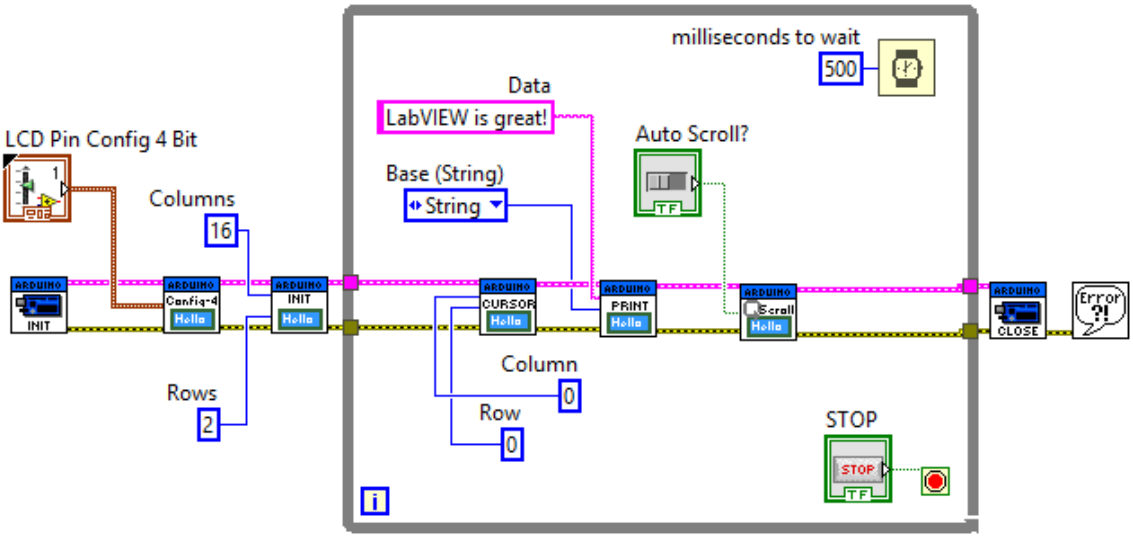
Należy teraz dodać kolejne bloki. Z palety *Functions*, zakładki *Arduino* i kategorii *Sensors* wybrać podkategorię *LCD*, w której znajduje się *LCD Configure 4-bit.* Blok ten jest odpowiedzialny za konfigurację odpowiednich pinów wykorzystanych w Arduino. Blok *Init* należy połączyć z nowo dodanym blokiem. Wyjście *Arduino Resource* bloku inicjalizacji należy połączyć z wejściem *Arduino Resource* bloku konfiguracji oraz wyjście *Error out* bloku inicjalizacji trzeba połączyć z wejściem *Error in* bloku konfiguracji. Przy tym połączeniu należy zwrócić uwagę na to, że każdy kolejny blok dotyczący Arduino trzeba łączyć w ten sam sposób, tj. wyjście *Arduino Resource* bloku poprzedzającego z wejściem *Arduino Resource* bloku następnego oraz wyjście *Error out* bloku poprzedzającego z wejściem *Error in* bloku następnego. Dodatkowo przy bloku konfiguracji trzeba zdefiniować numery wykorzystywanych pinów. Należy utworzyć nowy element *Control* poprzez kliknięcie prawym przyciskiem myszy na wejściu *LCD Pin Config 4 Bit* i wybraniu opcji *Create*. Po wykonaniu tego kroku powinien utworzyć się *Numeric Control* w oknie aplikacji VI. Następnym blokiem jest blok *LCD Init*, który znajduje się w palecie *Functions*, zakładce *Arduino,* kategorii *Sensors* i podkategorii *LCD.* Na wejściu *Columns* utworzyć zmienną *Constant* i przypisać jej wartość 16 (ilość kolumn, z których będzie korzystał wyświetlacz) oraz na wejściu *Rows* utworzyć zmienną *Constant* i przypisać jej wartość 2 (ilość wierszy, z których będzie korzystał wyświetlacz).

W kolejnym etapie należy utworzyć pętlę *While.* Do elementu *Loop Condition* należy utworzyć element *Control*. Po dodaniu, na aplikacji VI powinien pojawić się przycisk *Stop* zatrzymujący główną pętlę programu. W pętli *While* z palety *Functions*, zakładki *Arduino*, kategorii *Sensors* i podkategorii *LCD* dodać blok *LCD Set Cursor Position*. Na wejściu *Column* utworzyć zmienną *Constant* i przypisać jej wartość 0 (jest to pierwsza kolumna wyświetlacza) oraz na wejściu *Row* utworzyć zmienną *Constant* i przypisać jej wartość 0 (jest to pierwszy wiersz wyświetlacza). Kolejny blok odpowiada za wyświetlenie napisu na LCD. Blok *LCD Print* znajduje się w palecie *Functions*, zakładce *Arduino*, kategorii *Sensors* i podkategorii *LCD*. Na jego wejściu należy utworzyć zmienną *Constant* typu *String* i zapisać do niej napis: „LabVIEW is great!”. Następnym blokiem jest blok *LCD Auto Scroll*, który umożliwia przesuwanie (tzw. scrollowanie)

napisu na wyświetlaczu. Na jego wejściu trzeba utworzyć nowy element *Control*. Można zauważyć, że utworzył się przełącznik w oknie aplikacji VI. Ten przełącznik umożliwi włączenie i wyłączenie scrollowania.

Ostatnim etapem jest dodanie opóźnienia w pętli *While* poprzez dodanie bloku *Wait (ms)* i podaniu na jego wejście wartość odpowiadającej 500 ms opóźnienia. Należy również zamknąć połączenie z modułem Arduino. Trzeba dodać blok *Close*, który znajduje się w zakładce *Arduino* w palecie *Functions.* Blok umieścić za pętlą *While* i połączyć z ostatnim blokiem dotyczącym Arduino, tj. *LCD Print*. Na wyjście *Error out* bloku *Close* należy dodać ostatni element, który można znaleźć wyszukując w palecie *Functions* po nazwie: *Simple Error Handler.vi*.

Ostateczna wersja programu w oknie *Block Diagram* powinna wyglądać jak na rys. 5:



Rys. 5 – Kod programu w oknie Block Diagram

Wygląd aplikacji VI powinien wyglądać podobnie jak na rys. 6:

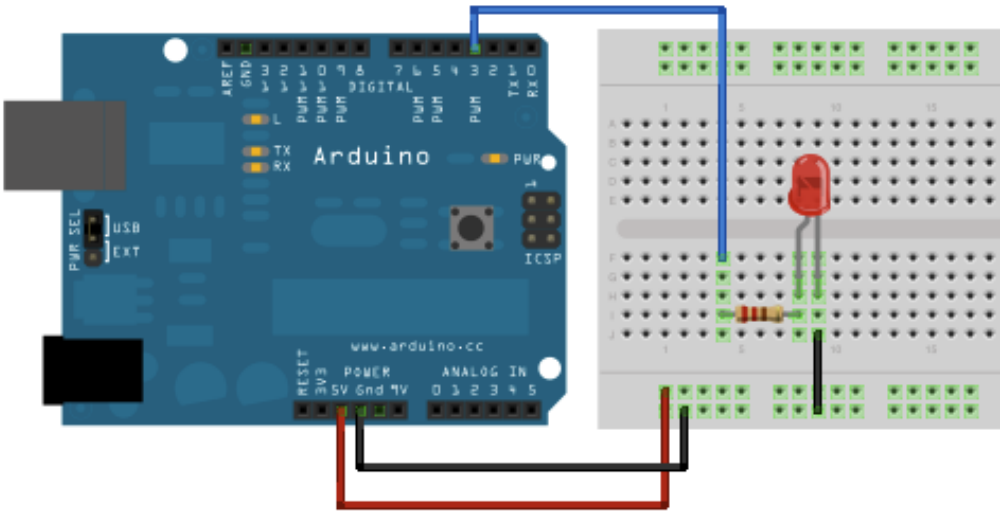
Rys. 6 – Aplikacja VI

4. Program wykorzystujący PWM do sterowania jasnością diody

Przedstawiony w tym punkcie program będzie wykorzystywał dostępne w Arduino wyjście PWM do sterowania jasnością diody LED.

Pierwszym krokiem jest zmontowanie układu. Na potrzeby tego programu wymagana jest jedynie dioda LED, rezystor 220 Ω oraz kilka przewodów aby podłączyć elementy z Arduino.

Układ po zmontowaniu powinien wyglądać jak na rys. 7:



Rys. 7 – Schemat połączeń (Dioda LED, Arduino)

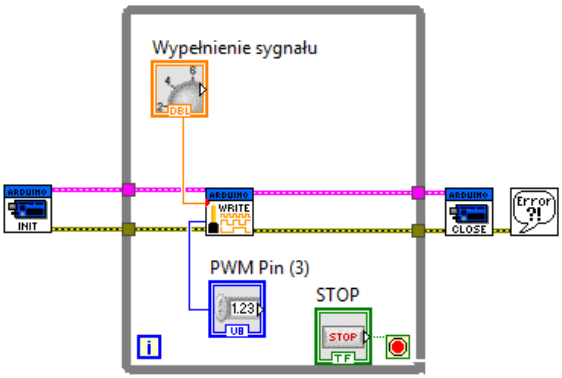
Kolejnym krokiem jest przystąpienie do napisania programu w środowisku LabVIEW. Po utworzeniu nowego projektu należy przejść do okna *Block Diagram* i dodać element *Init* z palety *Functions* i zakładki *Adruino*, który pozwoli na inicjalizację programu z platformą Arduino. Po najechaniu na blok *Init* przy odpowiednich wejściach powinno wyświetlić się:
- przy drugim wejściu: *Baud Rate (115200)*
- przy trzecim wejściu: *Board Type (Uno)*
- przy czwartym wejściu: *Bytes per packet (15)*
- przy piątym wejściu: *Connection Type (USB/Serial)*

Należy teraz dodać kolejne bloki. Należy utworzyć pętlę *While.* Do elementu *Loop Condition* należy utworzyć element *Control*. Po dodaniu, na aplikacji VI powinien pojawić się przycisk *Stop* zatrzymujący główną pętlę programu. Z palety *Functions*, zakładki *Arduino* i kategorii *Low Level* wybrać blok *PWM Write Pin.* Blok ten jest odpowiedzialny za sterowanie PWM dla odpowiednich pinów w Arduino. Blok *Init* należy połączyć z nowo dodanym blokiem. Wyjście *Arduino Resource* bloku inicjalizacji należy połączyć z wejściem *Arduino Resource* bloku konfiguracji oraz wyjście *Error out* bloku inicjalizacji trzeba połączyć z wejściem *Error in* bloku

konfiguracji. Przy tym połączeniu należy zwrócić uwagę na to, że każdy kolejny blok dotyczący Arduino trzeba łączyć w ten sam sposób, tj. wyjście *Arduino Resource* bloku poprzedzającego z wejściem *Arduino Resource* bloku następnego oraz wyjście *Error out* bloku poprzedzającego z wejściem *Error in* bloku następnego. Na wejście *PWM Pin* bloku *PWM Write Pin* należy utworzyć nowy element *Control*, dzięki któremu będzie możliwe zdefiniowanie numeru pinu wykorzystanego w Arduino (można to zrobić z poziomu aplikacji Vi; wartość powinna być ustawiona na 3). W oknie aplikacji VI należy utworzyć element *Dial*, który znajduje się w palecie *Controls* i zakładce *Numeric*. W oknie *Block Diagram* powinien utworzyć się nowy blok pod nazwą *Dial*, którego wyjście trzeba połączyć z wejściem *Duty Cycle* bloku *PWM Write Pin*.

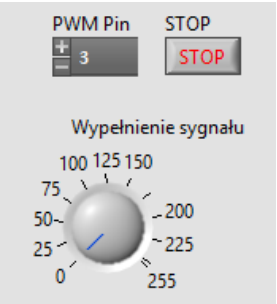
Ostatnim etapem jest dodanie opóźnienia w pętli *While* poprzez dodanie bloku *Wait (ms)* i podaniu na jego wejście wartość odpowiadającej 500 ms opóźnienia. Należy również zamknąć połączenie z modułem Arduino. Trzeba dodać blok *Close*, który znajduje się w zakładce *Arduino* w palecie *Functions.* Blok umieścić za pętlą *While* i połączyć z ostatnim blokiem dotyczącym Arduino, tj. *LCD Write Pin*. Na wyjście *Error out* bloku *Close* należy dodać ostatni element, który można znaleźć wyszukując w palecie *Functions* po nazwie: *Simple Error Handler.vi*.

Ostateczna wersja programu w oknie *Block Diagram* powinna wyglądać jak na rys. 8:



Rys. 8 – Kod programu w oknie Block Diagram

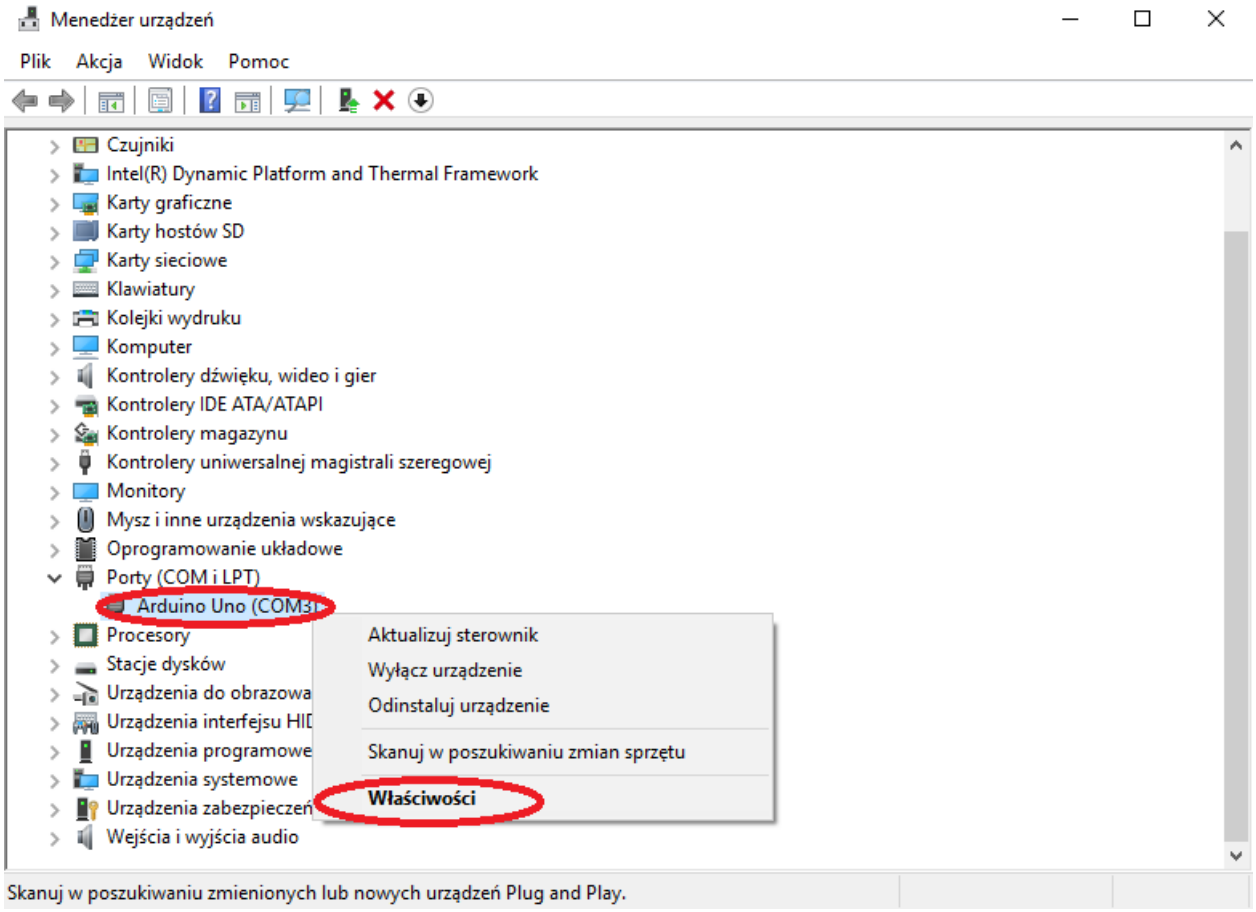
Wygląd aplikacji VI powinien wyglądać podobnie jak na rys. 9:



Rys. 9 – Aplikacja VI

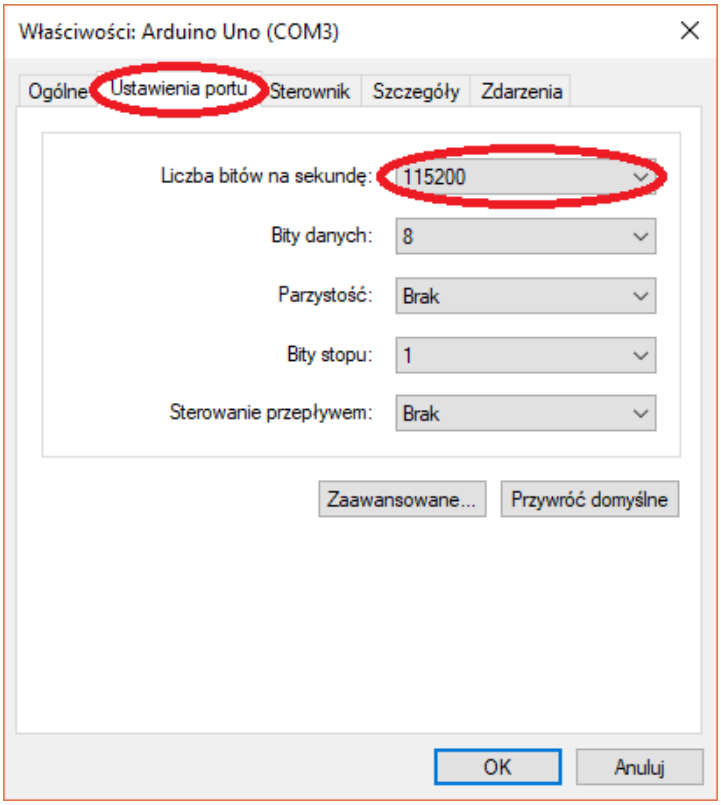
5. Komunikacja z Arduino

Aby program prawidłowo działał należy sprawdzić czy połączenie komputera z płytką Arduino jest prawidłowe. Arduino połączyć z komputerem poprzez port USB. Należy się upewnić, czy ustawiona wartość liczby bitów na sekundę jest zgodna z tą ustawioną w programie LabVIEW (domyślnie jest to 115200, ale można tą wartość zmienić dodając zmienną *Constant* o wartości 9600 na wejście *Baud Rate* bloku *Init*). Aby to zrobić należy przejść do *Menedżera Urządzeń* i rozwinąć zakładkę *Porty (COM i LPT).* Powinny wyświetlić się dołączone urządzenia. Należy zlokalizować Arduino Uno i prawym przyciskiem myszy rozwinąć menu kontekstowe, z którego trzeba wybrać opcję *Właściwości* jak na rys. 10:



Rys. 10 – Widok Menedżera Urządzeń

Należy przejść na zakładkę *Ustawienia portu* i sprawdzić liczbę bitów na sekundę (wartość ma być ustawiona na 115200):



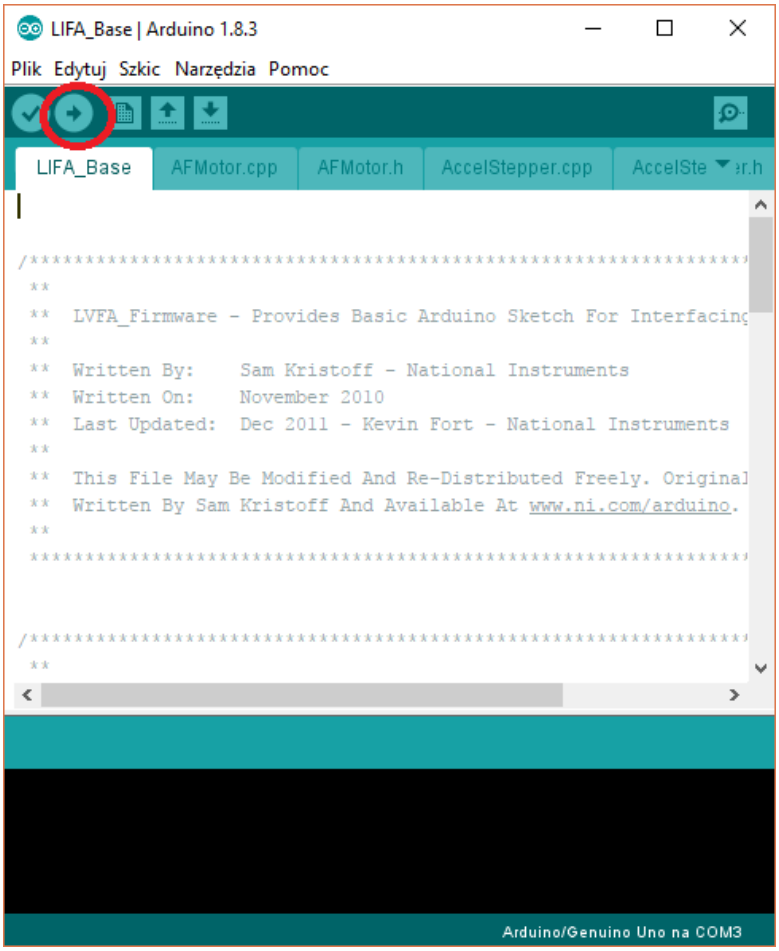
Rys. 11 – Widok właściwości portu COM

6. Wgranie i uruchomienie programu

Aby uruchomić program i sterować nim z poziomu LabVIEW należy najpierw wgrać odpowiedni program na płytkę rozwojową Arduino. Należy zlokalizować folder, w którym znajdują się odpowiednie pliki do otworzenia przez Arduino IDE. Przykładowa ścieżka do plików:

C:\Program Files (x86)\National Instruments\LabVIEW 2016\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA_Base

Należy wybrać plik *LIFA_Base* i wgrać go na płytkę rozwojową Arduino klikając przycisk jak na rys. 12:



Rys. 12 – Wgrywanie programu do Arduino

Ostatnim etapem jest uruchomienie stworzonych wcześniej programów.

Dla pierwszego programu na wyświetlaczu powinien pojawić się napis: „Temperatura: **”, gdzie w miejscu ** wyświetla się aktualna temperatura.

Dla drugiego programu na wyświetlaczu powinien pojawić się napis: „LabVIEW is great!”. Po wciśnięciu przycisku na aplikacji VI napis powinien zacząć się przesuwać.

Dla trzeciego programu po użyciu pokrętła *Dial* można zauważyć zmianę jasność diody.