LabVIEW

PLATFORMA EDUKACYJNA

Lekcja 5

LabVIEW i Arduino – konfiguracja środowiska i pierwszy program

Przygotował: Jakub Wawrzeńczak

1. Wprowadzenie

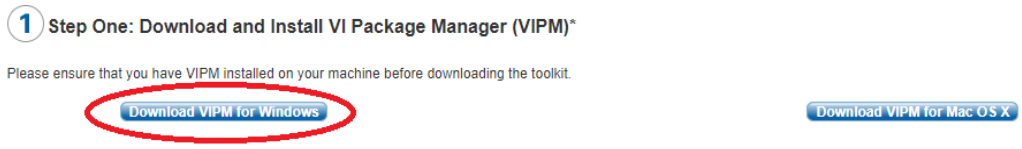
Lekcja przedstawia wykorzystanie środowiska LabVIEW 2016 w wersji 32 bitowej do programowania modułu Arduino UNO R3. Warto dodać, że możliwe jest również programowanie modułu Arduino Mega. Ważne jest, aby programowane płytki Arduino były oficjalnymi modułami od producenta, a nie ich klonami, ze względu brak komunikacji tych urządzeń ze środowiskiem LabVIEW. W tym kursie będzie przedstawiona instalacja wymaganych bibliotek i rozszerzeń do przygotowania środowiska, a następnie zostaną zaprezentowane przykładowe programy wykorzystujące LabVIEW oraz Arduino.

2. Przygotowanie środowiska

Aby było możliwe wykorzystanie środowiska LabVIEW do programowania Arduino, konieczne jest pobranie odpowiedniego dodatku. Należy wejść na stronę internetową producenta oprogramowania:

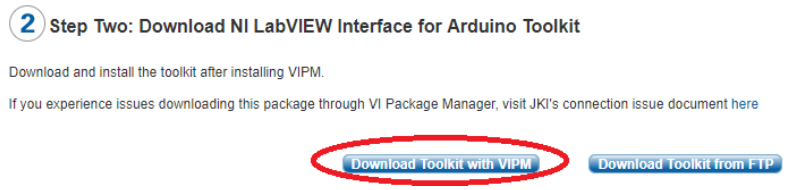
http://www.ni.com/gate/gb/GB_EVALTLKTLVARDIO/US

a następnie postępować według kroków zaprezentowanych na stronie, tj.:
Pobrać VI Package Manager (jeżeli nie pobrano wcześniej) klikając na przycisk pokazany na rys. 1:



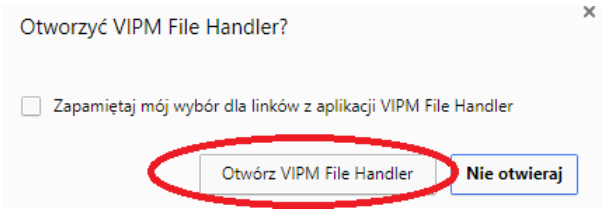
Rys. 1 – Pobieranie VI Package Manager

Zainstalować VIPM postępując zgodnie z kolejnymi krokami instalatora.
Pobrać NI LabVIEW Interface for Arduino Toolkit korzystając z VI Package Manager (rys. 2):



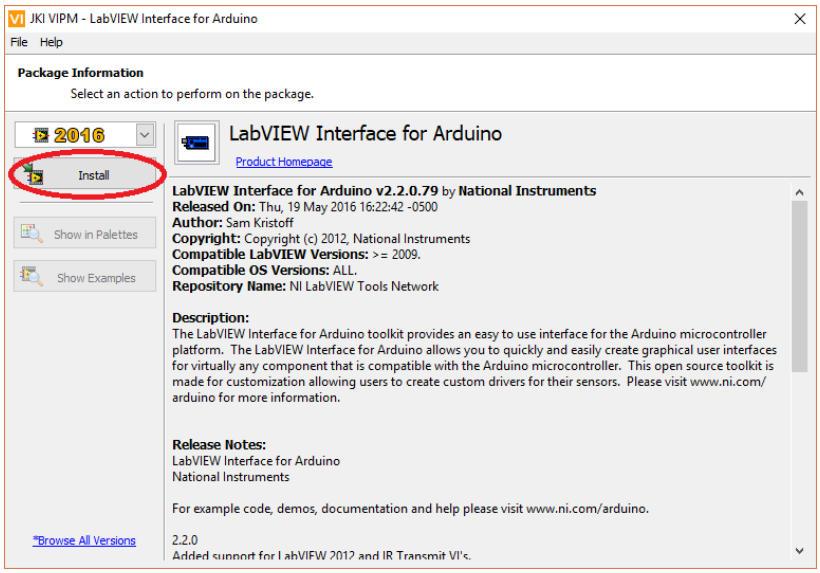
Rys. 2 – Pobieranie NI LabVIEW Interface for Arduino Toolkit

Po kliknięciu *Download Toolkit with VIPM* należy zezwolić na uruchomienie programu klikając przycisk pokazany na rys. 3:



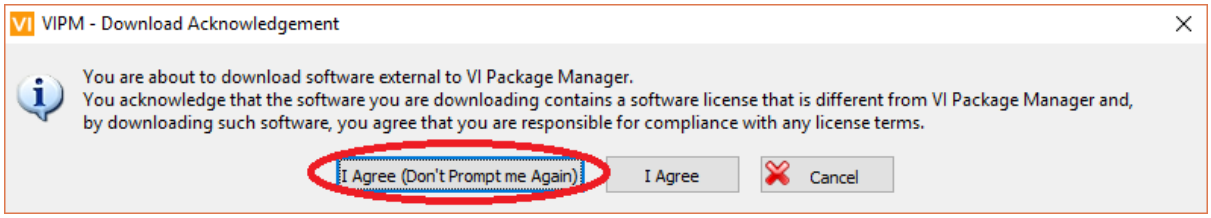
Rys. 3 – Zezwolenie na uruchomienie VIPM

Na tym etapie powinien uruchomić się VI Package Manager.
Następnie aplikacja wyszuka odpowiednie pliki i otworzy dodatkowe okno programu(rys. 4), w którym należy rozpocząć instalację dodatku do LabVIEW:



Rys. 4 – Instalacja LabVIEW Interface for Arduino

Po kliknięciu na przycisk *Install* pojawi się okno (rys. 5), w którym należy potwierdzić i zaakceptować warunki pobrania dodatkowego oprogramowania:



Rys. 5 – Instalacja LabVIEW Interface for Arduino

Jeżeli instalacja przebiegła pomyślnie to należy potwierdzić tą przyciskiem *Finish* oraz powinno uruchomić się okno startowe środowiska LabVIEW. Po zakończeniu instalacji można wyłączyć VI Package Manager oraz LabVIEW.

Należy również zainstalować niezbędny sterownik NI VISA. Można go pobrać ze strony:

http://www.ni.com/download/ni-visa-17.0/6646/en/

Po pobraniu wymaganych plików instalacyjnych należy przeprowadzić instalację sterownika postępując zgodnie z kolejnymi krokami instalatora.

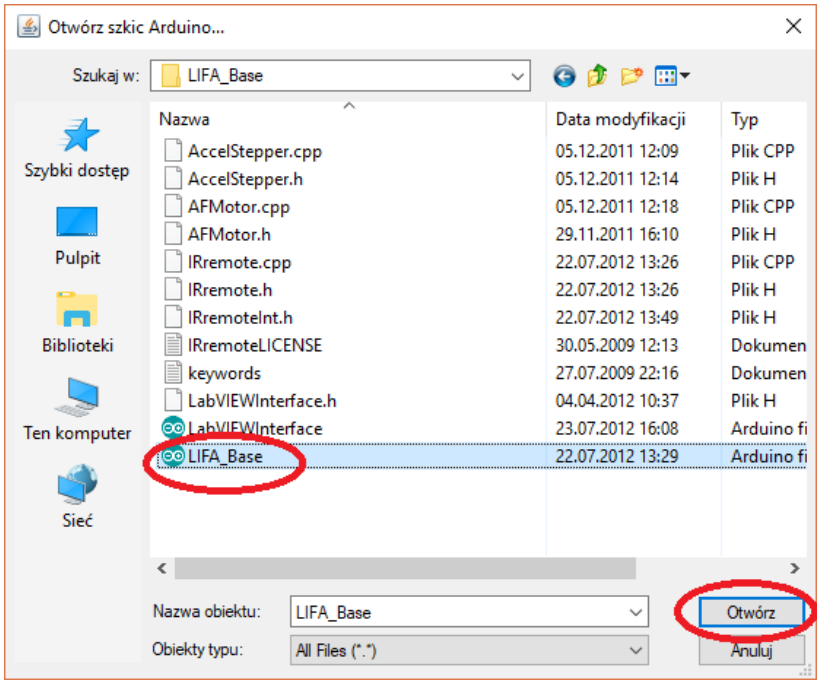
Drugim etapem jest instalacja środowiska Arduino IDE. Najnowszą wersję można pobrać ze strony producenta płytki:

https://www.arduino.cc/en/Main/Software

Po pobraniu instalatora, zainstalować Arduino IDE postępując zgodnie z kolejnymi krokami. W kolejnym kroku należy uruchomić aplikację, kliknąć *Plik* na pasku narzędzi i następnie *Otwórz.* 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* zaznaczony na rys. 6 i następnie otworzyć go:



Rys. 6 – Przygotowanie środowiska Arduino

W oknie środowiska powinno otworzyć się kilka plików podobnie jak na rys. 7:

Rys. 7 – Pliki źródłowe z LabVIEW Interface for Arduino

Plik LIFA_Base jest plikiem, który należy wgrywać na płytkę rozwojową Arduino.

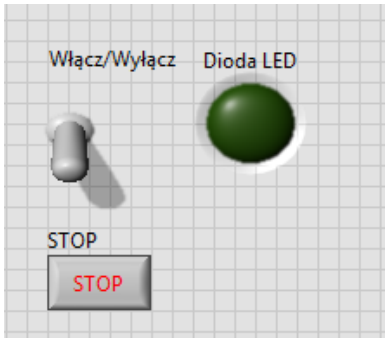
Na tym etapie konfiguracja środowisk została zakończona. Można przejść do zaprogramowania Arduino poprzez LabVIEW.

3. Przykładowy program

Programowanie z wykorzystaniem platformy Arduino odbywa się w środowisku LabVIEW.

Przykładowy program pozwoli na sterowanie diodą podłączoną do Arduino z poziomu LabVIEW. W pierwszej kolejności należy utworzyć plik *.vi*, dzięki któremu możliwe będzie sterowanie diodą. Z palety *Controls* i zakładki *Boolean* należy wybrać: przełącznik *Vertical Toggle Switch*, który pozwoli na włączanie oraz wyłączanie diody, wirtualną diodę *Round LED*, która będzie zapalała się razem z rzeczywistą diodą podłączoną do modułu Arduino. Po utworzeniu tych elementów należy przejść do okna *Block Diagram* utworzyć nową pętlę *While* i dodać przycisk STOP. Aby to wykonać trzeba kliknąć prawym przyciskiem myszy na *Loop Condition* i wybrać opcję *Create Control*.

Na tym etapie można zakończyć tworzenie aplikacji VI. Powinna ona wyglądać podobnie jak na rys. 8:



Rys. 8 – Aplikacja VI

W drugim etapie należy przystąpić do programowania w oknie *Block Diagram.*

W palecie *Functions* z listy trzeba wybrać zakładkę *Arduino* i następnie element *Init*, który pozwoli na inicjalizację programu z platformą Arduino. Jeżeli wszystkie dodatki do środowiska zostały prawidłowo zainstalowane, 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)*

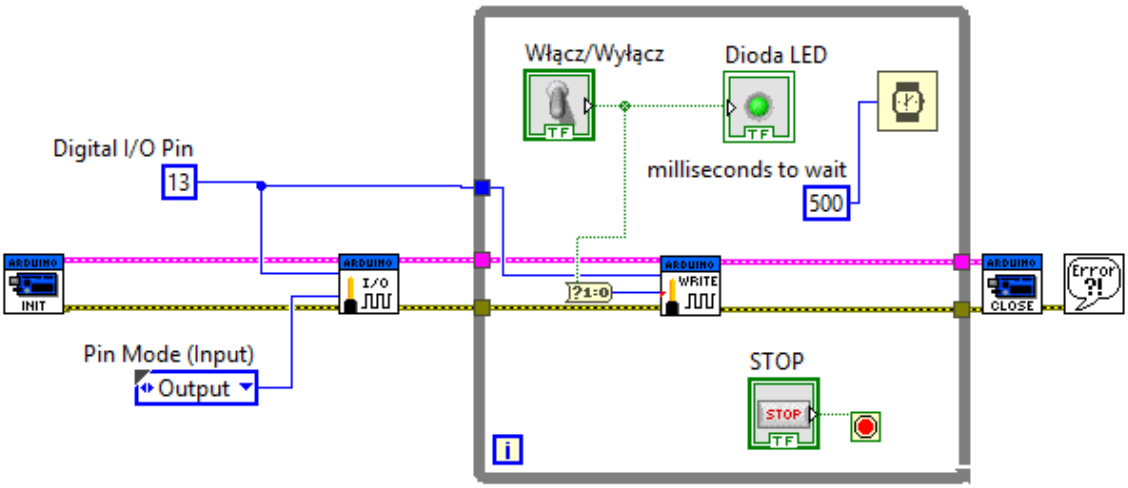
Należy teraz dodać kolejne bloki. Z palety *Functions*, zakładki *Arduino* i kategorii *Low Level* wybrać *Set Digital Pin Mode.* Blok ten jest odpowiedzialny za konfigurację odpowiedniego pinu wykorzystanego w Arduino. Na potrzeby tego ćwiczenia niech będzie to pin numer 13 (nie wymaga to podłączania dodatkowej diody, ze względu na to iż moduł Arduino posiada diodę podłączoną pod pin 13). 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 pinu trzeba zdefiniować numer wykorzystywanego pinu oraz tryb pinu (wejście lub wyjście). Należy utworzyć nową zmienną *Constant* poprzez kliknięcie prawym przyciskiem myszy na wejściu *Digital I/O Pin* i wybraniu opcji *Create*. Zmiennej przypisać numer pinu, czyli 13. Podobnie należy zrobić przy wejściu *Pin Mode* i ustawić wartość zmiennej jako *Output*.

Następnym etapem jest wysterowanie zdefiniowanego wcześniej pinu 13. Wewnątrz utworzonej wcześniej pętli *While* należy dodać blok *Digital Write Pin*, który znajduje się w palecie *Functions*, zakładce *Arduino* i kategorii *Low Level.* Wewnątrz pętli muszą znaleźć się również bloki odpowiedzialne za przełącznik i diodę (elementy te były dodane na etapie tworzenia interfejsu aplikacji VI). Blok przełącznika należy połączyć z blokiem diody. Z palety *Functions* i kategorii *Boolean* wybrać element *Bool to (0,1)* i jego wejście połączyć z wyjściem przełącznika. Element ten jest potrzebny, gdyż na wejście *Value* bloku *Digital Write Pin* podawana jest wartość 0/1. Wyjście bloku *Bool to (0,1)* połączyć z wejściem *Value* bloku *Digital Write Pin* oraz wejście *Digital I/O Pin* tego bloku połączyć ze zmienną *Constant* odpowiadającą numerowi pinu Arduino (13). Końcowym elementem w pętli jest utworzenie opóźnienia. Z palety *Functions* i kategorii *Timing* wybrać blok *Wait (ms)* i na jego wejściu utworzyć zmienną *Constant* i przypisać jej wartość 500.

Ostatnim etapem jest zamknięcie połączenia z modułem Arduino. Należy dodać blok *Close*, który znajduje się w zakładce *Arduino* w palecie *Functions.* Blok umieścić za pętlą *While* i połączyć z blokiem *Digital 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 jak na rys. 9:

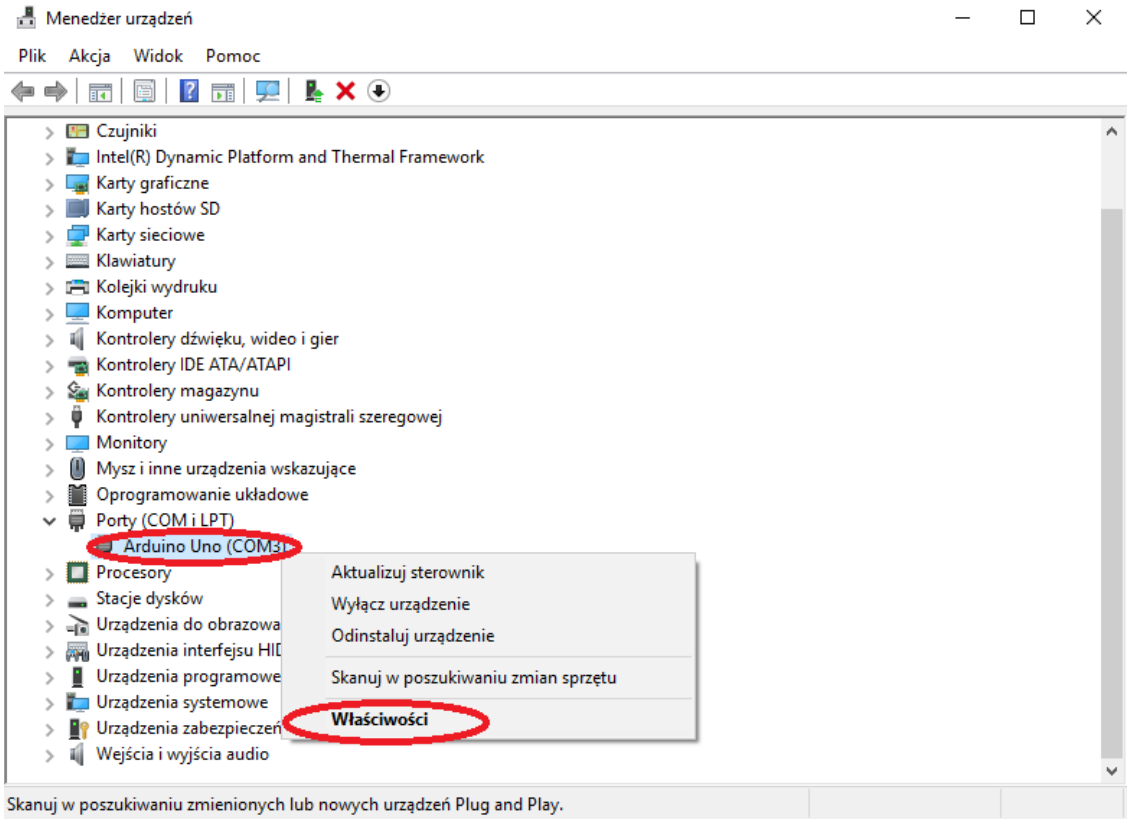


Rys. 9 – Kod programu w oknie Block Diagram

Taka wersja programu pozwoli na włączanie i wyłączanie diody połączonej z pinem 13 na płytce Arduino poprzez aplikację VI z poziomu środowiska LabVIEW.

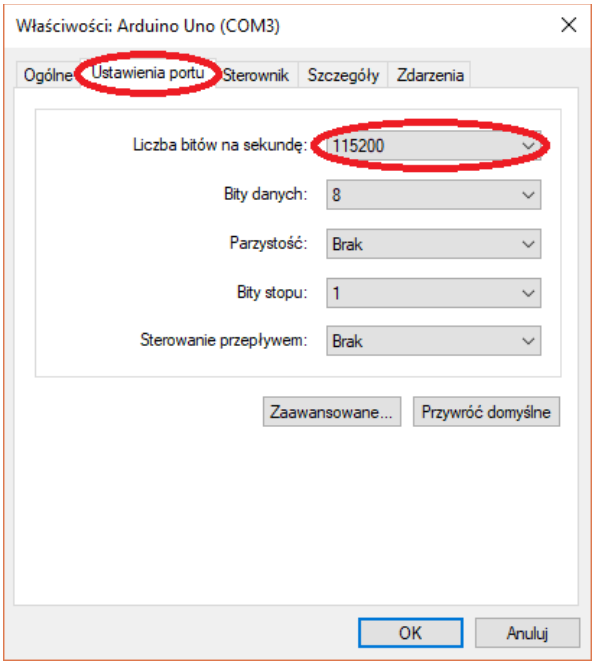
4. Komunikacja z Arduino

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):



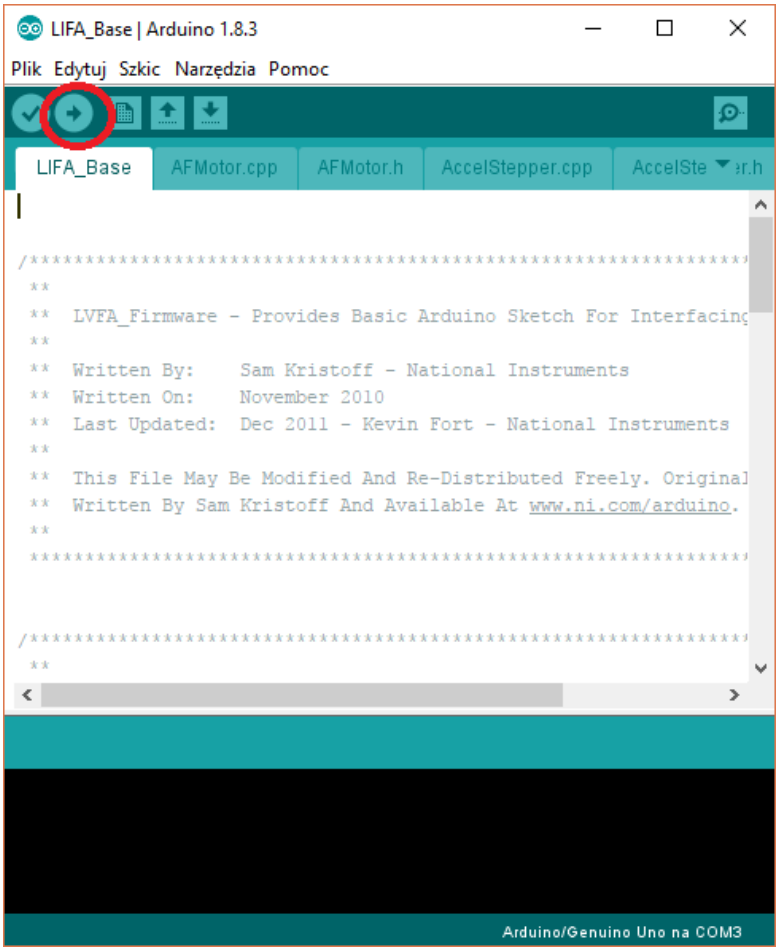
Rys. 11 – Widok właściwości portu COM

5. Wgranie i uruchomienie programu

Aby uruchomić program i sterować diodą podłączoną do Arduino 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

Kolejnym etapem jest uruchomienie stworzonego wcześniej programu. Jeżeli wszystko zostało wykonane prawidłowo to można zauważyć, że dioda podłączona do pinu 13 zapala się i gaśnie w zależności od położenia przełącznika wirtualnego w aplikacji VI.