

Instrukcja obsługi do kontrolera 4 osiowego dla drukarek 3D typu RepRap



Spis treści

1 Wstęp.....	3
2 Schemat podłączeniowy	3
3 Opis zworek odpowiadających za zasilanie.....	5
4 Opis zworek odpowiedzialnych za wybór kroków.....	6
5 Sterowniki i oprogramowanie dla systemu Windows.....	7
5.1 Instalacja pakietu Arduino.....	7
5.2 Instalacja pakietu Teensyduino.....	10
5.3 Instalacja pakietu FLIP.....	14
6 Sterowniki i oprogramowanie dla systemów Linux Ubuntu.....	19
7 Kompilacja i wgranie programu Marlin do sterownika KOS.....	23
7.1 Edycja Marlina.....	23
7.2 Kompilacja Marlina.....	25
7.3 Wgranie Marlin.cpp.hex do pamięci sterownika KOS na systemie Windows.....	28
7.4 Wgranie Marlin.cpp.hex do pamięci sterownika KOS na systemie Linux.....	34
8 Źródła oraz dodatkowe konfiguracje dla sterownika KOS.....	36
8.1 Schemat podłączenia LCD,Enkoder,SD-card.....	36
9 Bibliografia.....	37

1 Wstęp

KOS 1.2 jest elektroniką stworzoną z uwzględnieniem doświadczenia zebranego na budowaniu drukarek 3D Prusa. Wyposażono ją w złącza śrubowe, szeroko stosowane w elektronice przemysłowej, które eliminują błędy montażowe takie jak źle zaciśnięte wtyczki, czy też zbyt mała powierzchnia styku. Sterownik KOS 1.2 został dodatkowo wyposażony w czytelne złącza pod ekran LCD, czytnik karty SD oraz enkoder z guzikiem 47. Dzięki temu możemy bez problemu umieścić panel sterujący drukarką tam, gdzie nam się podoba. Na płycie umieszczono dodatkowo potencjometr do regulacji kontrastu wyświetlacza oraz rezystor regulujący intensywność podświetlenia, co zmniejsza poziom skomplikowania przy montażu dodatkowych peryferiów.

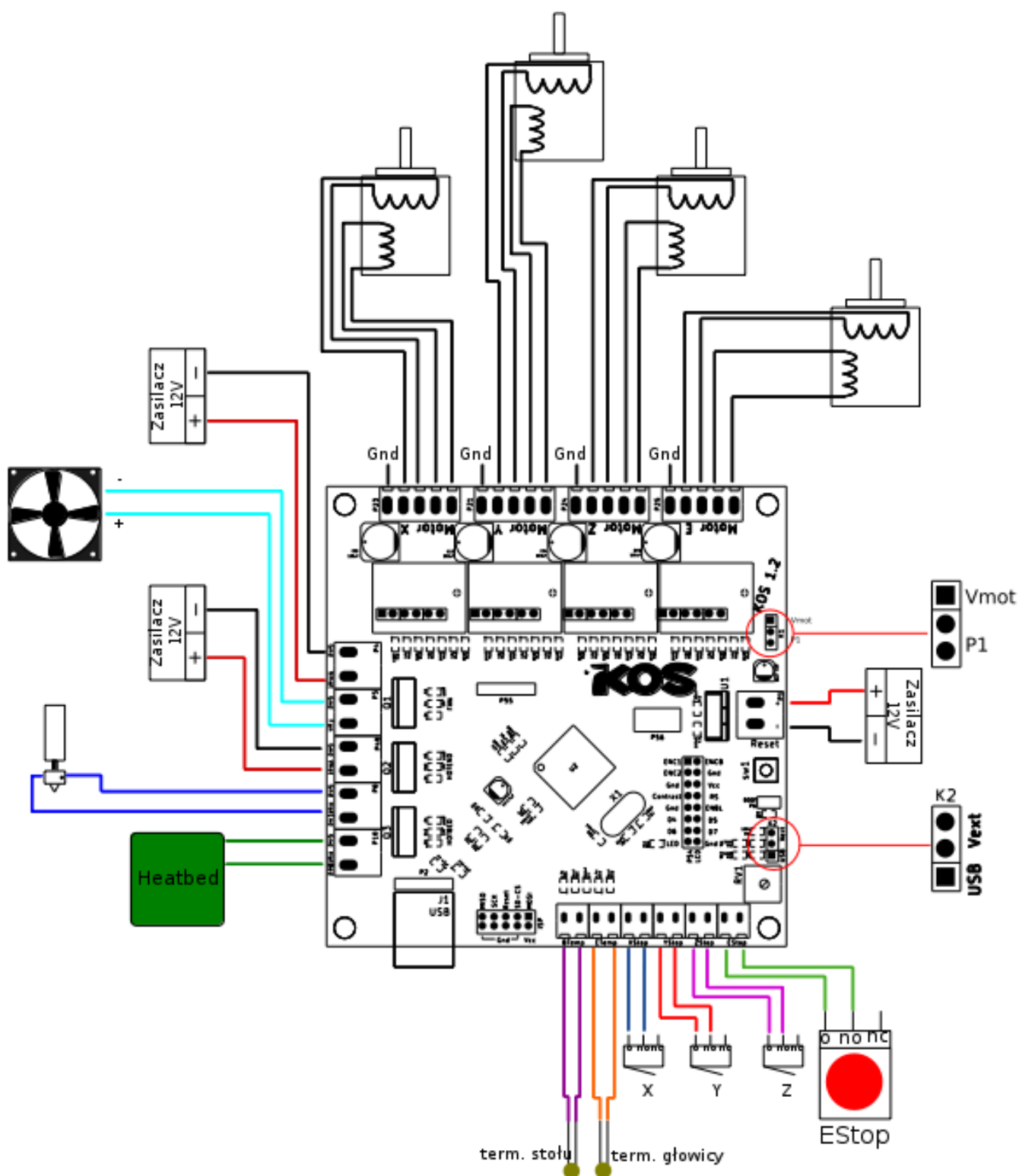
Jedną niezaprzeczalną zaletą elektroniki KOS jest możliwość stosowania naraz 3 różnych napięć. Oznacza to, że możemy silniki zasilić napięciem V_{mot} od 12-30V, grzałek napięciem V_{hot} od 12-30V i logiki napięciem $P1 = 12V$ bądź zasilić logikę z portu USB 5. Taka ilość możliwości nie każdemu jest potrzebna, więc wprowadzono dodatkowo zworki pozwalające na przełączanie zasilania dla uproszczenia instalacji.

Elektronika KOS 1.2 jest wyposażona w procesor AT90USB1286/7, który jest szeroko stosowanym kontrolerem 8 bitowym w drukarkach 3D typu RepRap. Jego zaletą jest natywna obsługa portu USB, pozwalająca na szybką wymianę danych i bezproblemową komunikację pomiędzy komputerem, a drukarką. Procesor ten jest stosowany w elektronikach takich jak Teensylu, Printrboard, SAV MKI, Sunbeam, Uniqueone i wiele innych. Jest to najpopularniejszy procesor z rodziny AVR w świecie druku 3D.

2 Schemat podłączeniowy

Poniżej przedstawiono schemat podłączenia podstawowych peryferiów drukarki 3D takich jak: grzałka, podgrzewany stół, silniki itd. Trzeba pamiętać o dobraniu odpowiednich kabli o odpowiednim przekroju.

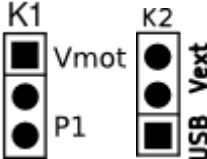
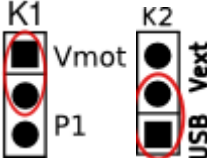
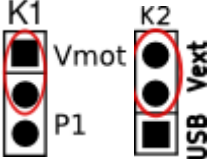
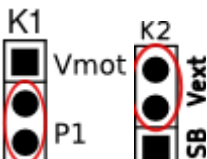
- Silniki: 0.35mm^2
- Grzałki: 1.5mm^2
- Krańcówki, termistory, wyłącznik bezpieczeństwa E-Stop: 0.35mm^2
- Zasilacze: 1.5mm^2



Ilustracja 1: Schemat podłączenia peryferiów do elektroniki KOS






2.1 Opis zworek odpowiadających za zasilanie

W elektronice KOS występuje możliwość podpięcia różnych rodzajów zasilania w zależności od aplikacji. W związku z tym, umieszczono 2 listwy kołkowe, które odpowiadają za przełączenie zasilania elektroniki z zasilania z portu USB na zasilanie ze złącza P1 – zasilania do 12V wyłącznie dla celów procesora i logiki, bądź z łącza Vmot – gdy zasilamy silniki napięciem 12V, możemy je wykorzystać również do zasilania procesora i logiki przełączając odpowiednią zworkę. W poniższej tabeli zaprezentowano możliwe konfiguracje.

lp.	Zwórka	Opis
1		Brak wybranego zasilania dla elektroniki
2		Zasilanie elektroniki z portu USB bez względu na wybór źródła napięcia na zworze K1
3		Zasilanie ze źródła zewnętrznego Vmot wybranego na zworze K1
4		Zasilanie ze źródła zewnętrznego P1 wybranego na zworze K1

2.2 Opis zworek odpowiedzialnych za wybór kroków

Do każdego silnika krokowego są przypisane zworki odpowiadające za podział kroków na sterownikach silników krokowych opartych o układ scalony firmy Allegro A4988. Poniżej opisano możliwe konfiguracje.

lp.	Zworki	Opis
1		Tryb pełno krokowy (1)
2		Tryb półkrokowy ($\frac{1}{2}$)
3		Tryb $\frac{1}{4}$ kroku ($\frac{1}{4}$)
4		Tryb $\frac{1}{8}$ kroku ($\frac{1}{8}$)
5		Tryb $\frac{1}{16}$ kroku ($\frac{1}{16}$)

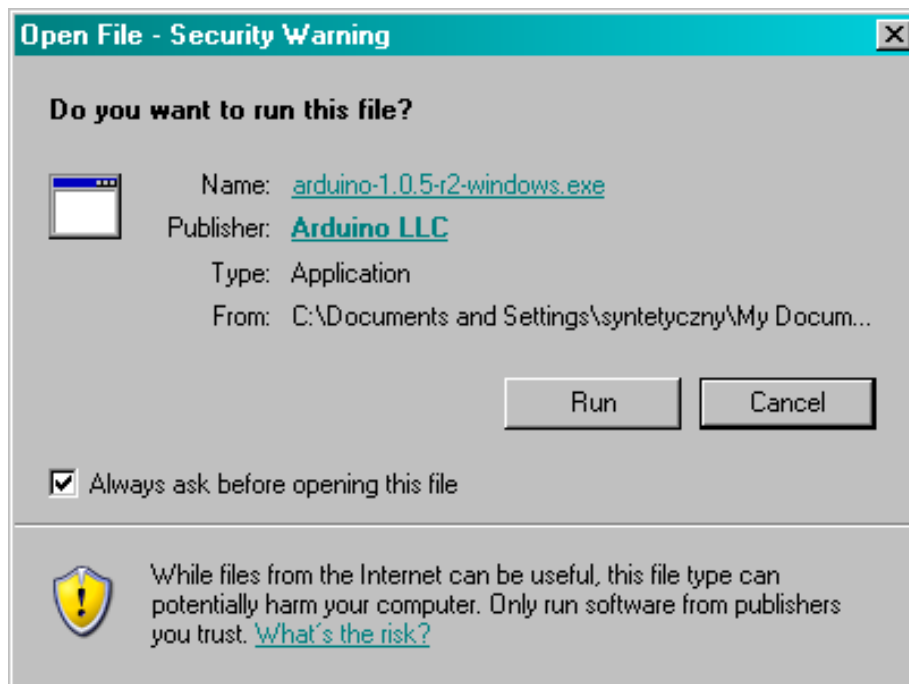
3 Sterowniki i oprogramowanie dla systemu Windows

Elektronika KOS jest zaprojektowana tak aby była kompatybilna wstecznie z elektroniką teensylu, więc aby poprawnie działała z komputerem, wymagana jest instalacja sterowników. Jeżeli elektronika jest dostarczona z już wgrany program Marlin, wystarczy zainstalować sterowniki ze strony projektu teensy 1.

W przeciwnym przypadku musimy stworzyć nową kompilację Marlin'a postępując zgodnie z instrukcją w dziale „Kompilacja Marlina” 5. Jednakże, zanim przejdziemy do tego, nasza elektronika wymaga bibliotek od oprogramowania Arduino, Teensyduino, oraz FLIP, które wcześniej musimy zainstalować:

- arduino ze strony autorów 2
- pakietu teensyduino 3
- programu FLIP for Windows 4

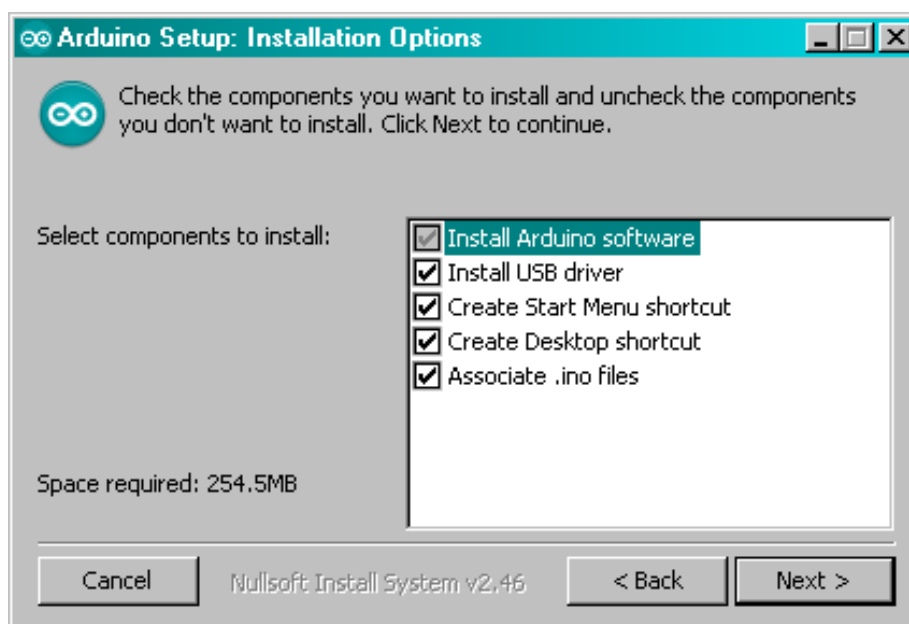
3.1 Instalacja pakietu Arduino



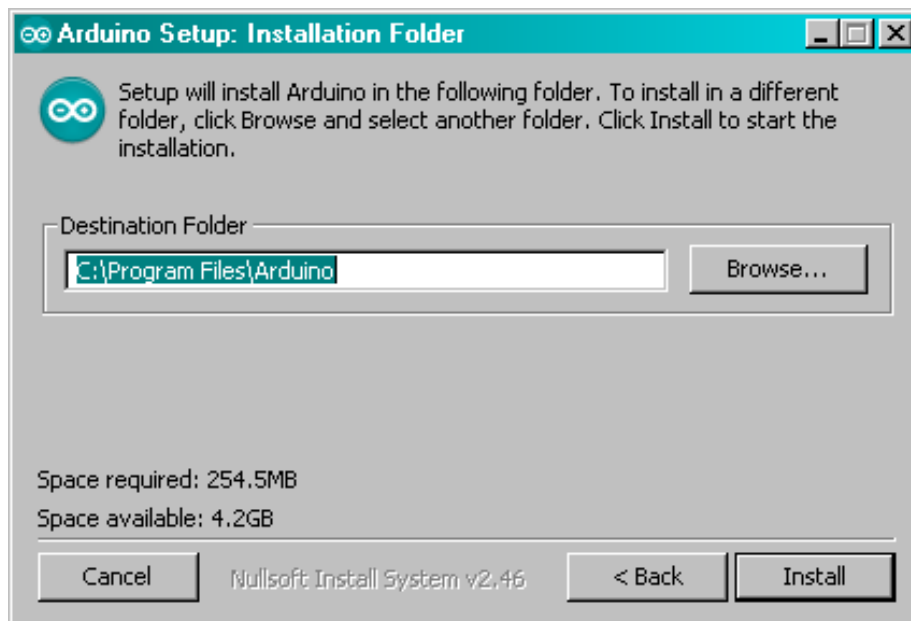
Ilustracja 2: Zezwolenie na uruchomienie pliku z niewiadomego źródła



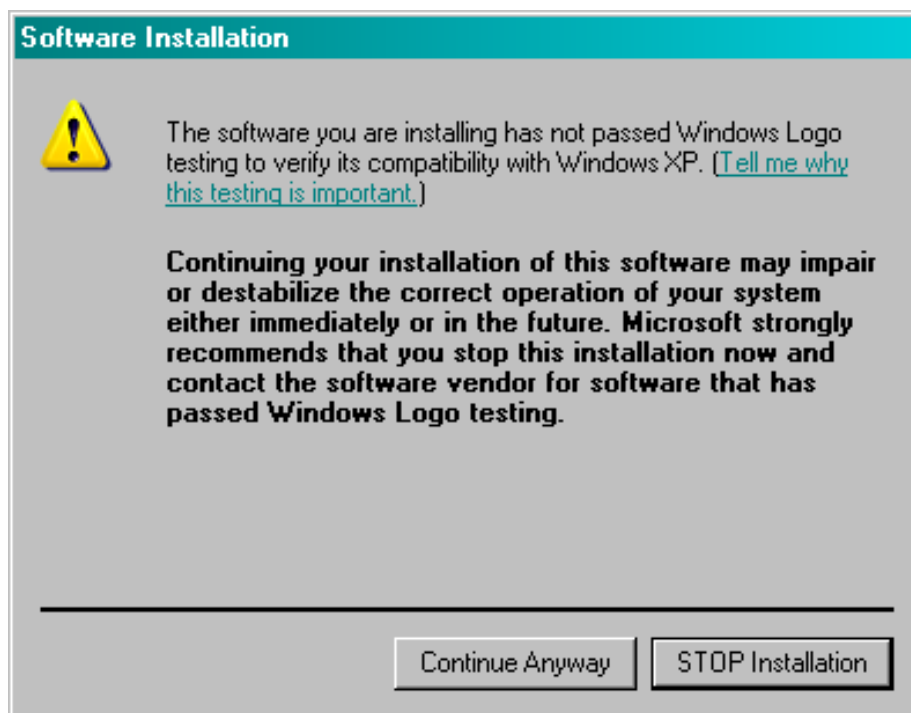
Ilustracja 3: Akceptacja licencji GNU



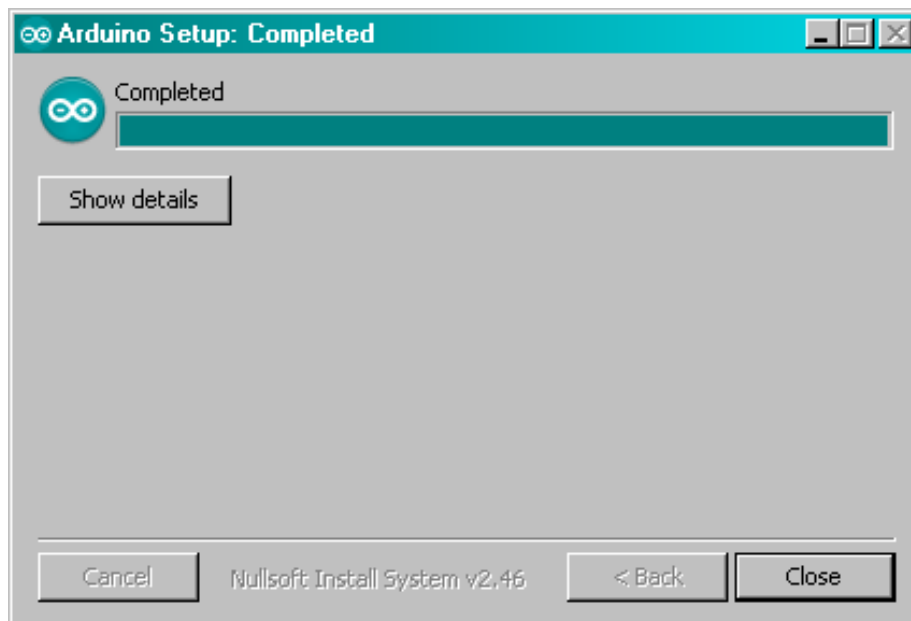
Ilustracja 4: Wybór pakietów, które zostaną zainstalowane



Ilustracja 5: Wybór lokalizacji, gdzie zostanie zainstalowane Arduino

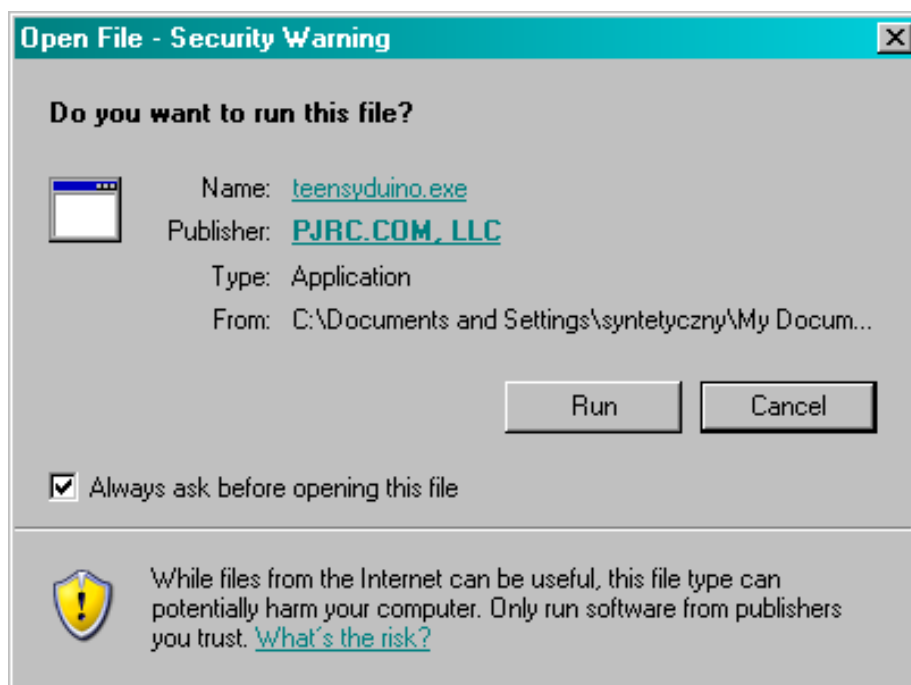


Ilustracja 6: Ostrzeżenie systemu Windows przed instalowaniem niezwyfikowanego oprogramowania

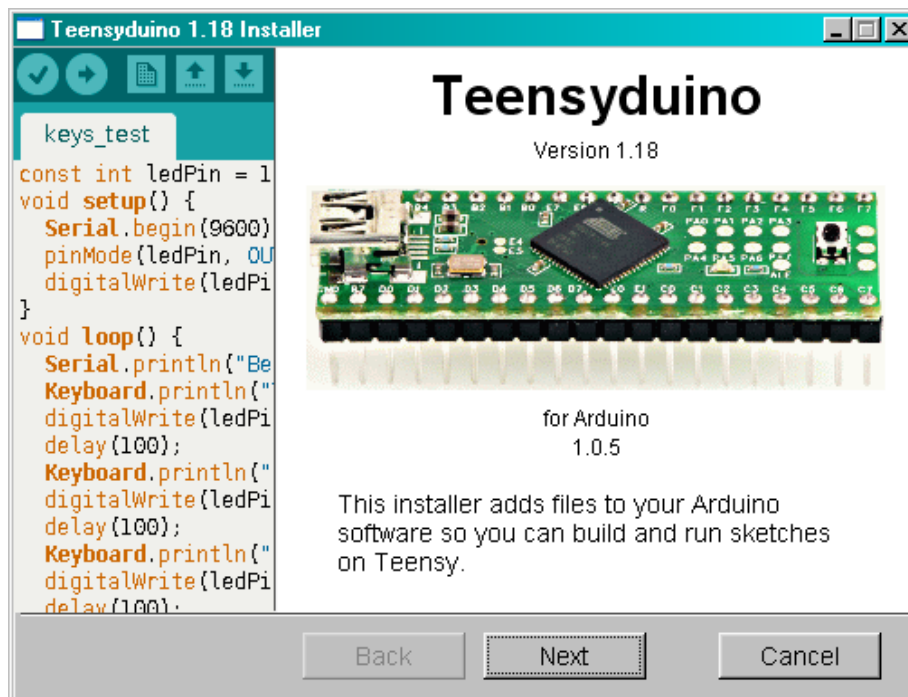


Ilustracja 7: Instalacja Arduino zwięńczona sukcesem!

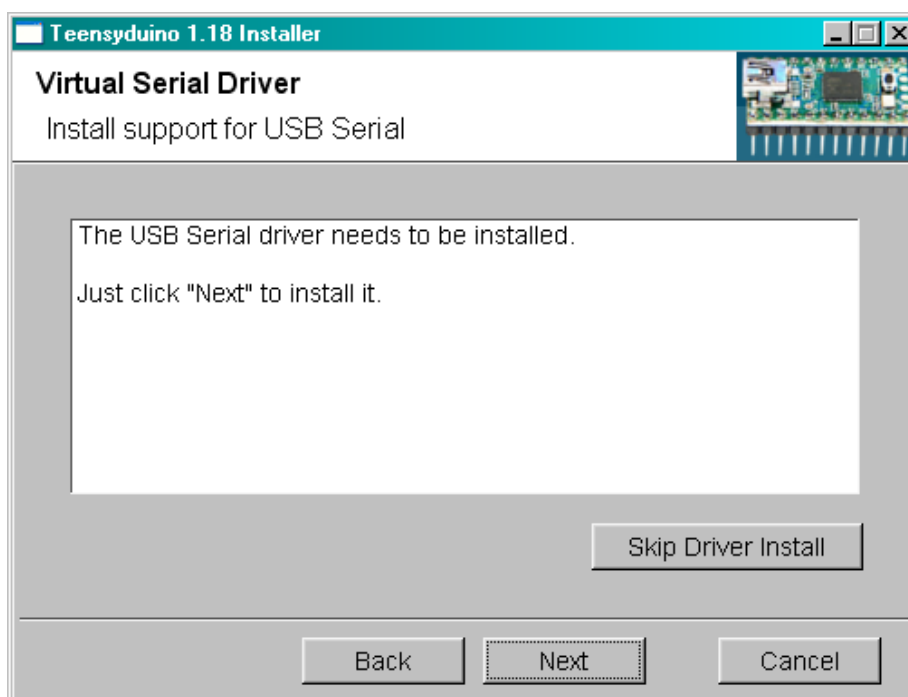
3.2 Instalacja pakietu Teensyduino



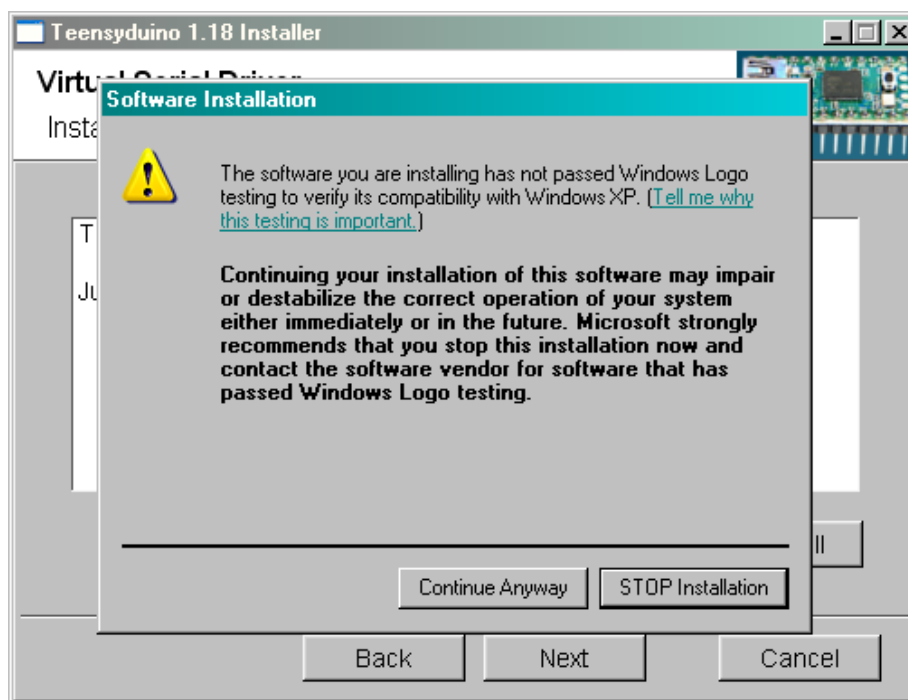
Ilustracja 8: Podobnie jak w przypadku Arduino - zezwolenie na uruchomienie pliku z niewiadomego źródła



Ilustracja 9: Wybieramy NEXT!



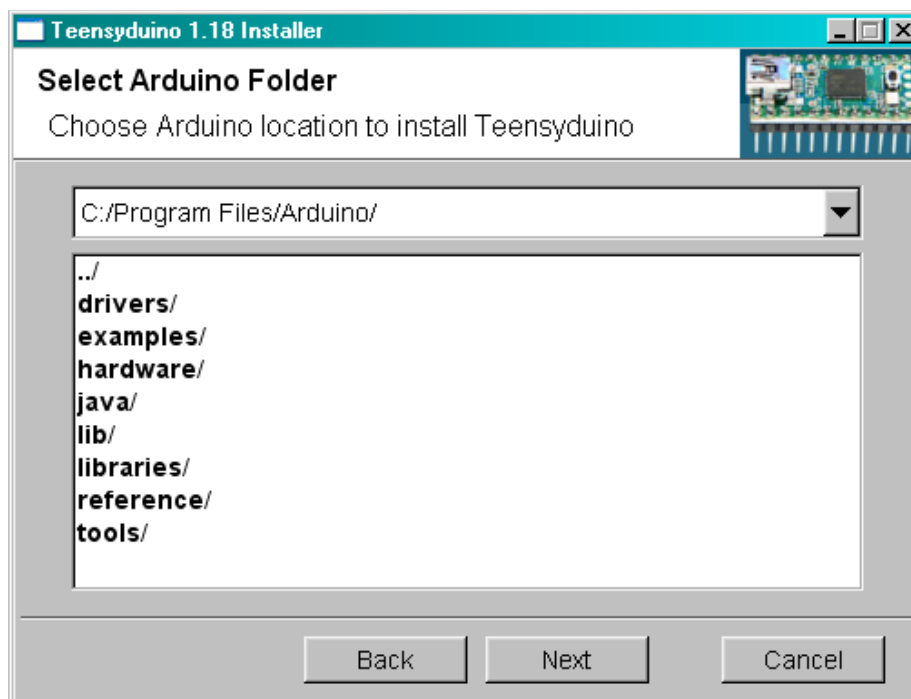
Ilustracja 10: Ponownie, wybieramy NEXT! Jeżeli nie instalowaliśmy wcześniej serial_installer.exe, instalator zainstaluje sterowniki.



Ilustracja 11: Wybieramy kontynuację instalacji(Continue Anyway), jeżeli chcemy aby instalacja przebiegła poprawnie

UWAGA! Ważny krok!

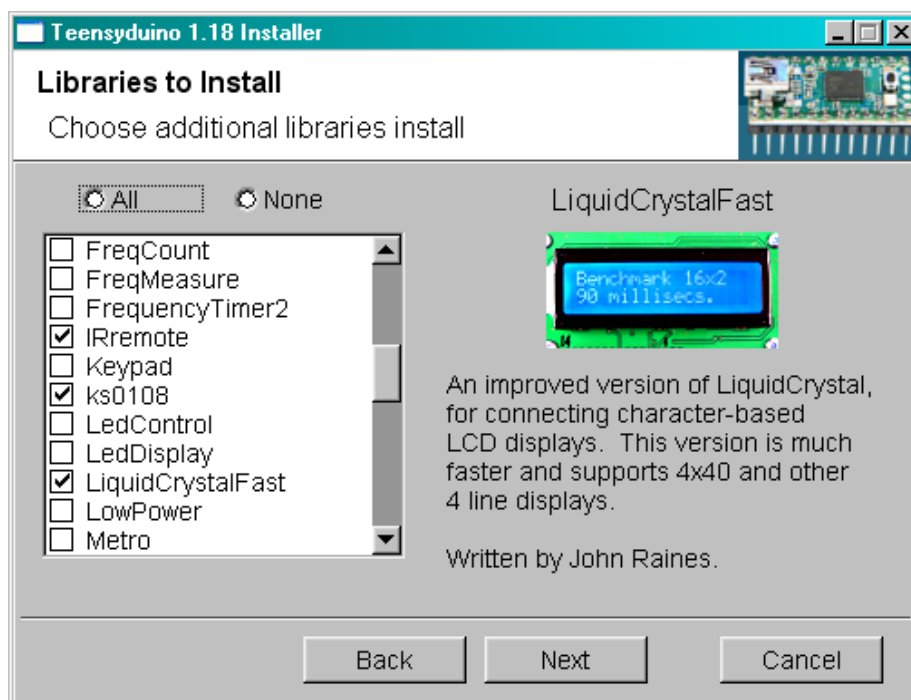
Aby instalacja zakończyła się sukcesem, musimy pamiętać o podaniu poprawnej lokalizacji pakietu Arduino, który będziemy używać.



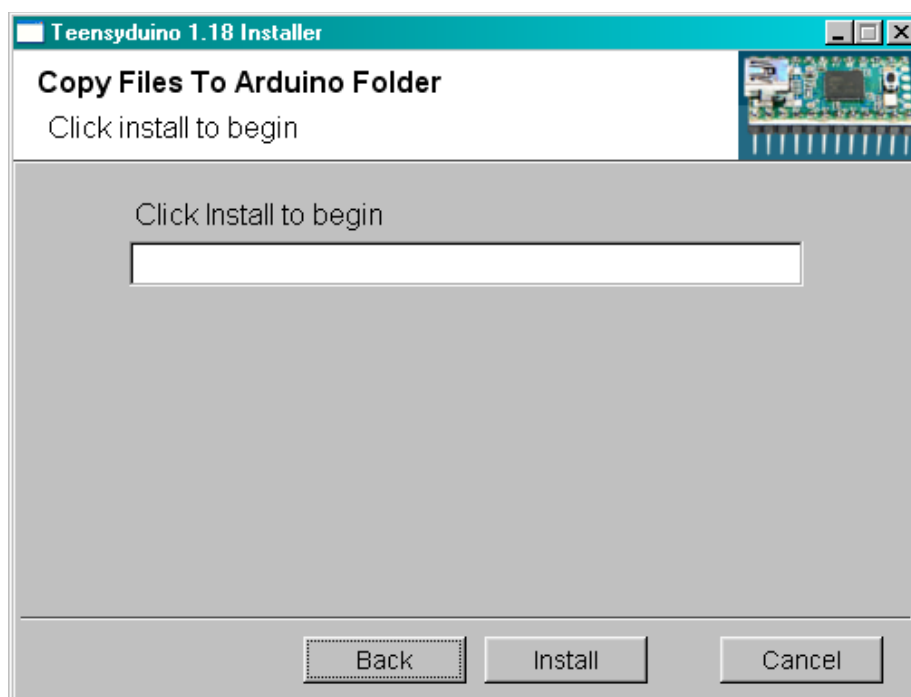
Ilustracja 12: Wybieramy lokalizację, w której wcześniej zainstalowaliśmy Arduino

PAMIĘTAJ!

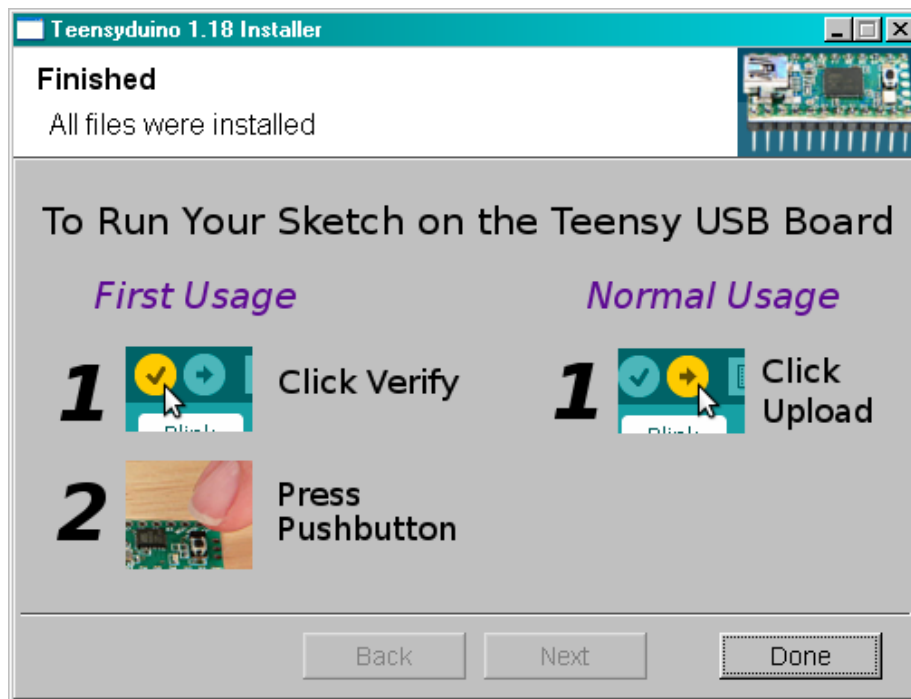
Wybieramy tylko 3 biblioteki IRremote, ks0108 oraz LiquidCrystalFast.



Ilustracja 13: Wybieramy 3 biblioteki, IRremote, ks0108 oraz LiquidCrystalFast

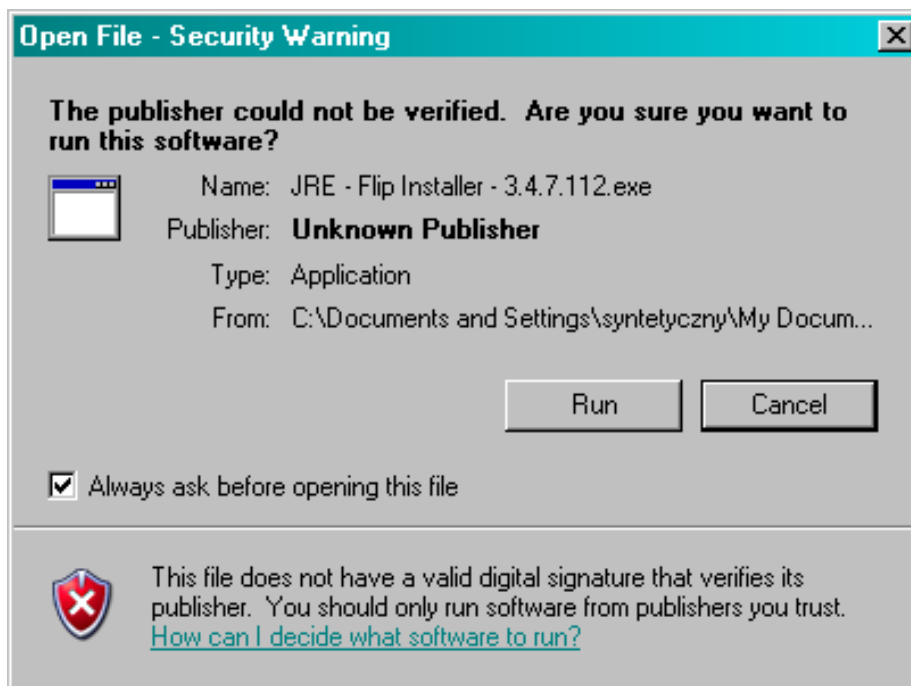


Ilustracja 14: Instalujemy!

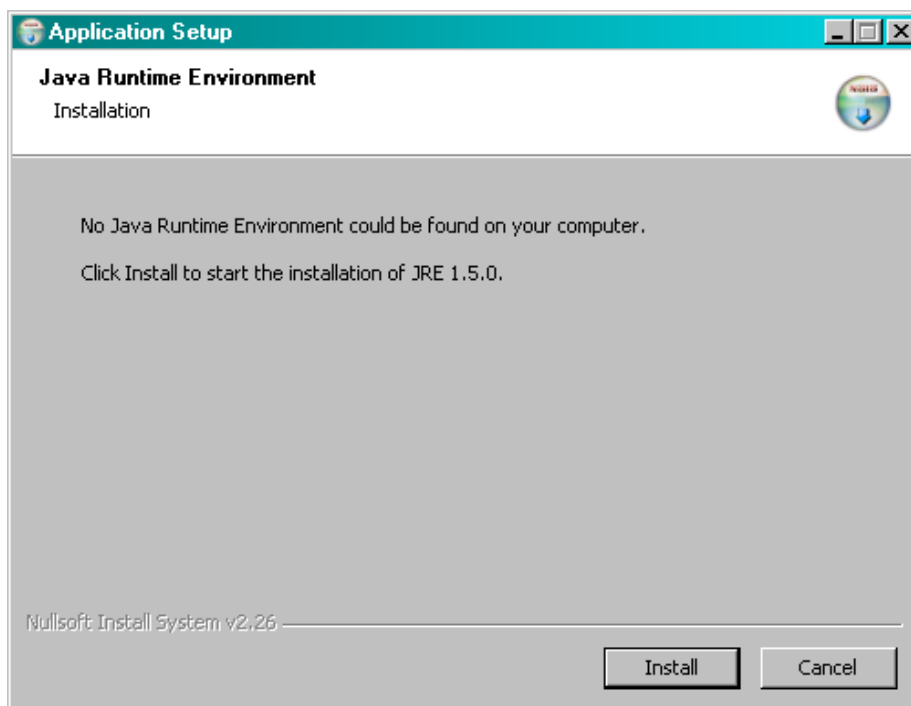


Ilustracja 15: Instalacja Teensyduino zakończona sukcesem! Można przejść do następnego kroku.

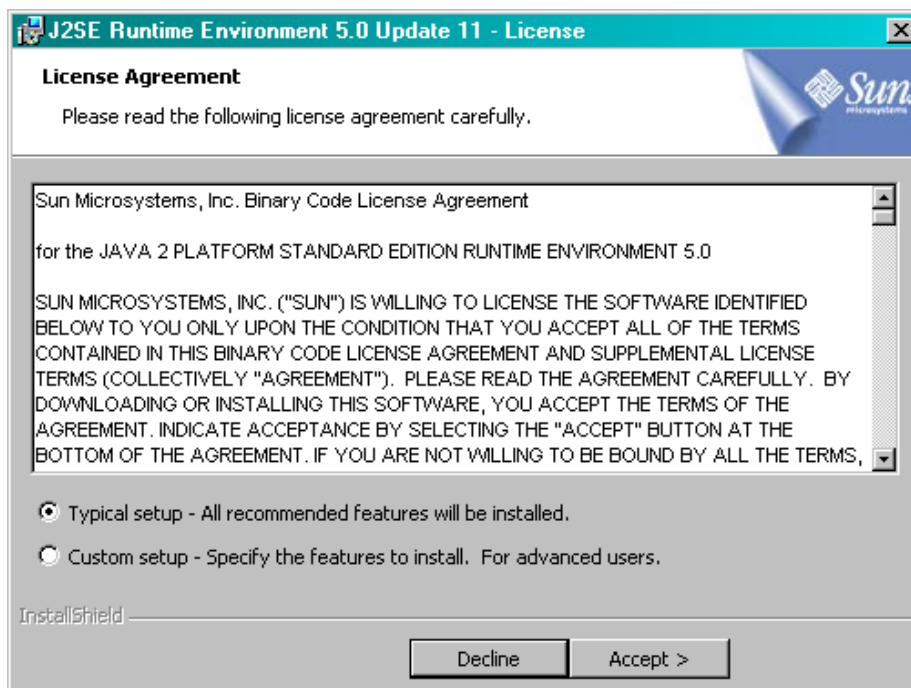
3.3 Instalacja pakietu FLIP



Ilustracja 16: Ponownie, zezwolenie na uruchomienie pliku z niewiadomego źródła



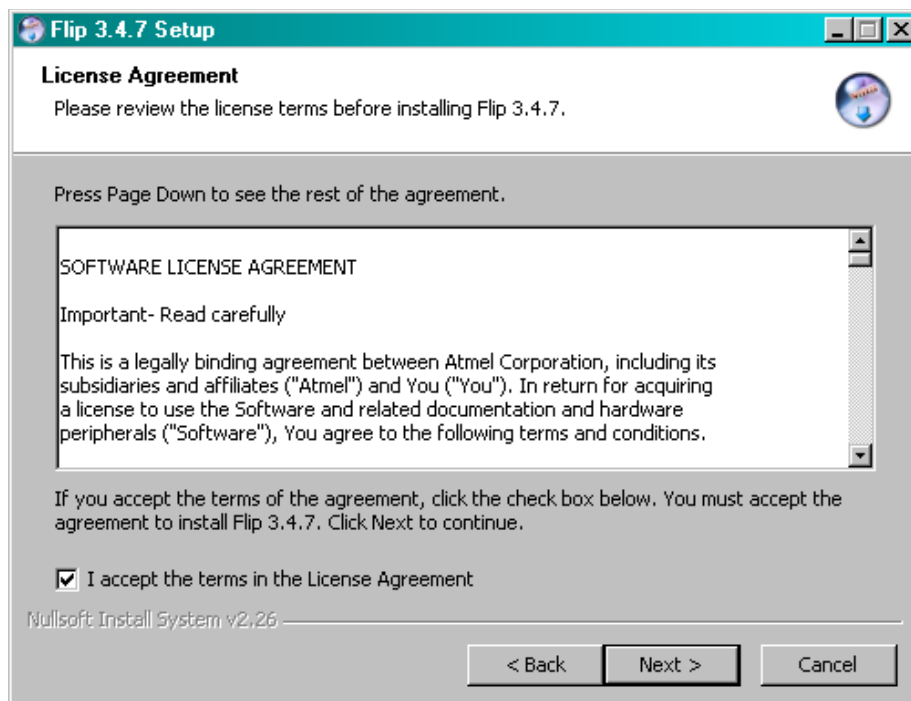
Ilustracja 17: W przypadku braku zainstalowanego pakietu Javy musimy ją doinstalować



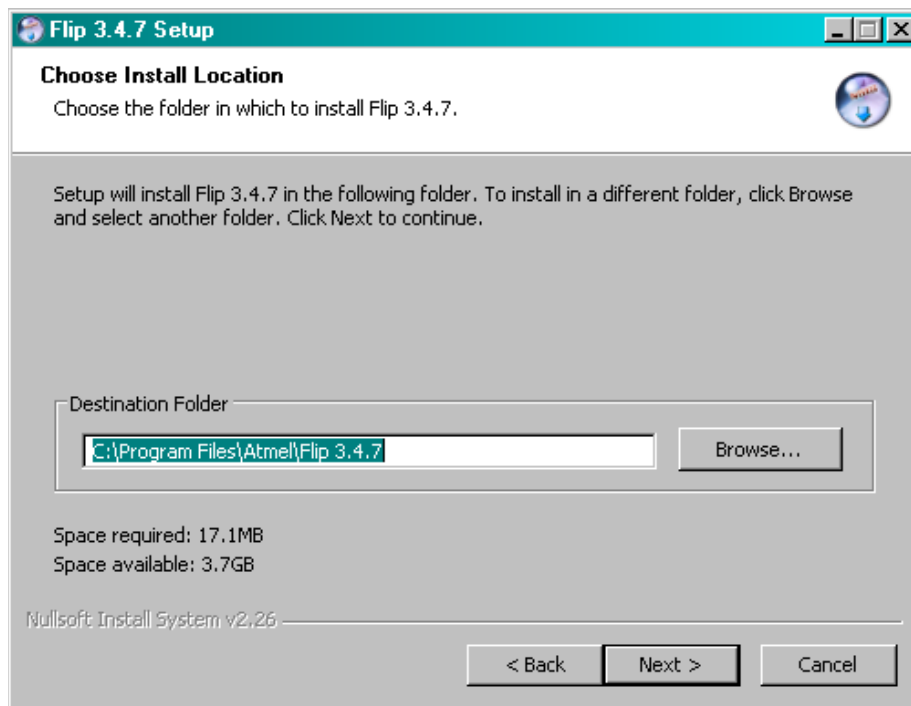
Ilustracja 18: Wybieramy standardową instalację i akceptujemy. Możemy przejść do następnego kroku.



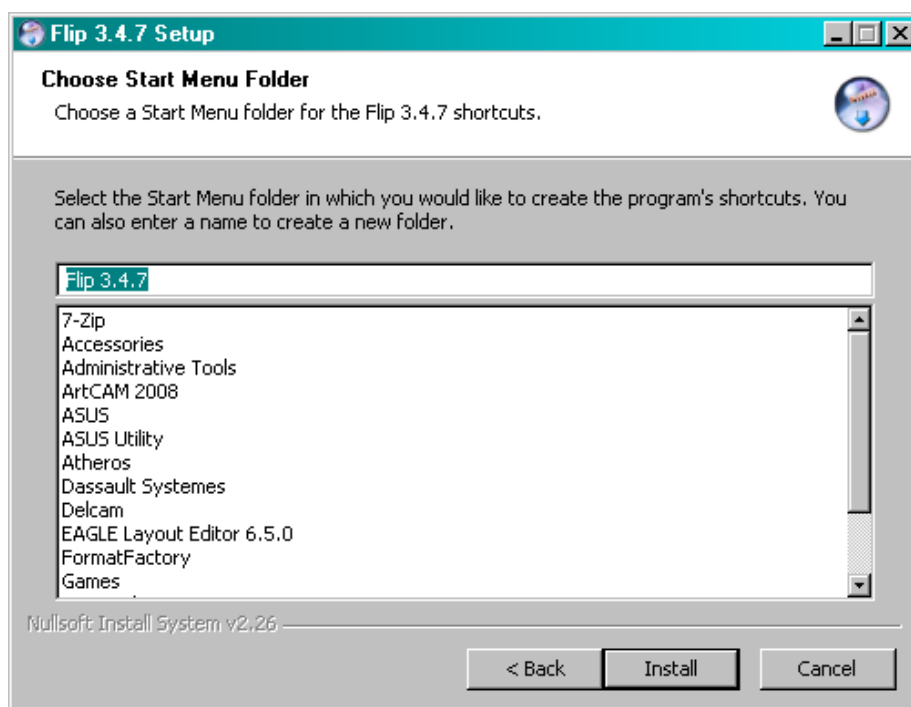
Ilustracja 19: Wciskamy NEXT!



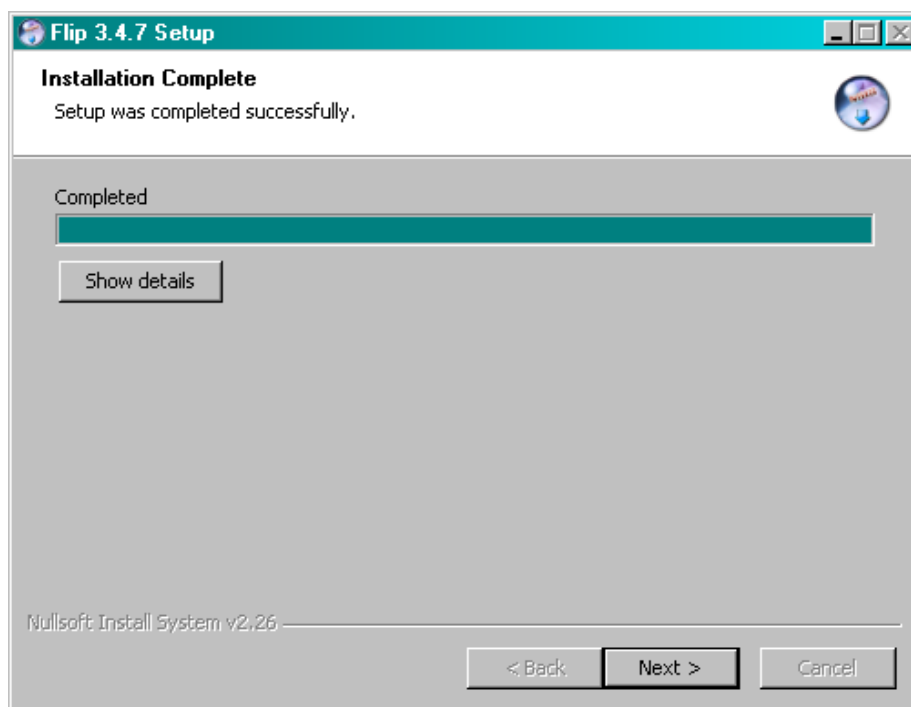
Ilustracja 20: Akceptujemy warunki licencji



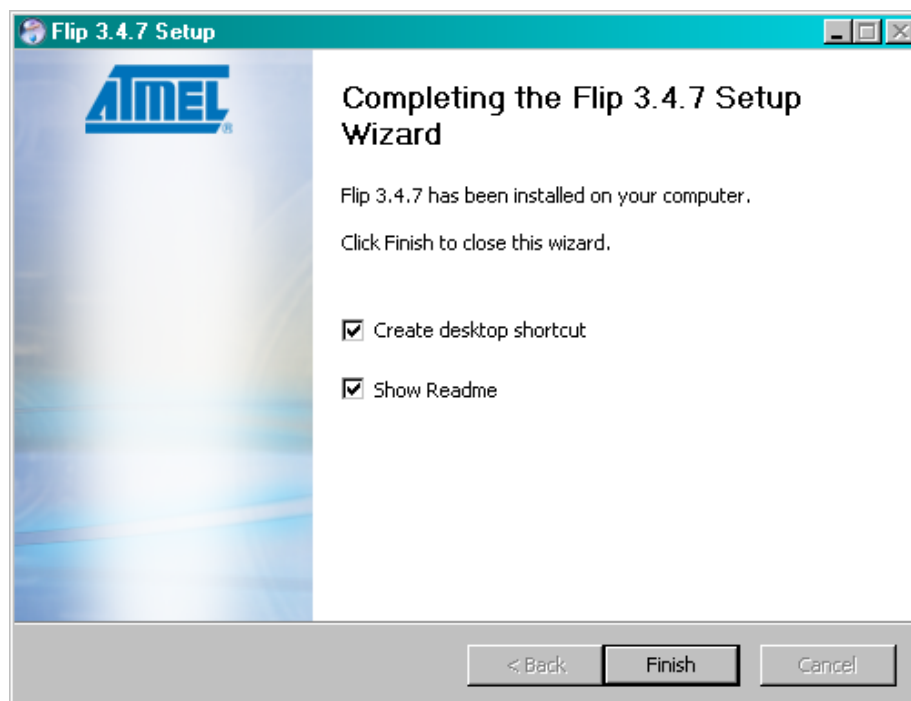
Ilustracja 21: Wybór ścieżki instalacyjnej do oprogramowania FLIP



Ilustracja 22: Wybór folderu paska Startowego, gdzie zostanie umieszczony skrót programu



Ilustracja 23: Instalacja Zakończona Sukcesem!



Ilustracja 24: Wybór ikon, które zostaną zapisane na pulpicie

4 Sterowniki i oprogramowanie dla systemów Linux Ubuntu

Tak samo jak w przypadku sterowników dla systemu Windows, zaczynamy od ściągnięcia pakietu Arduino ze strony producenta [2](#) . I rozpakowaniu w dowolnej lokalizacji. Następnie przy pomocy terminala, doinstalowujemy potrzebne pakiety:

```
sudo apt-get install gcc-avr avr-libc
```

```
sudo apt-get install openjdk-6-jre
```

Następnie, dodajemy prawa wykonywania dla programu Arduino. W tym celu musimy przejść do folderu z rozpakowanym Arduino i przy użyciu prawego klawisza myszki wejść we *Właściwości* i w zakładce *Uprawnienia* zaznaczyć *Zezwolenie na wykonanie pliku jako programu* .

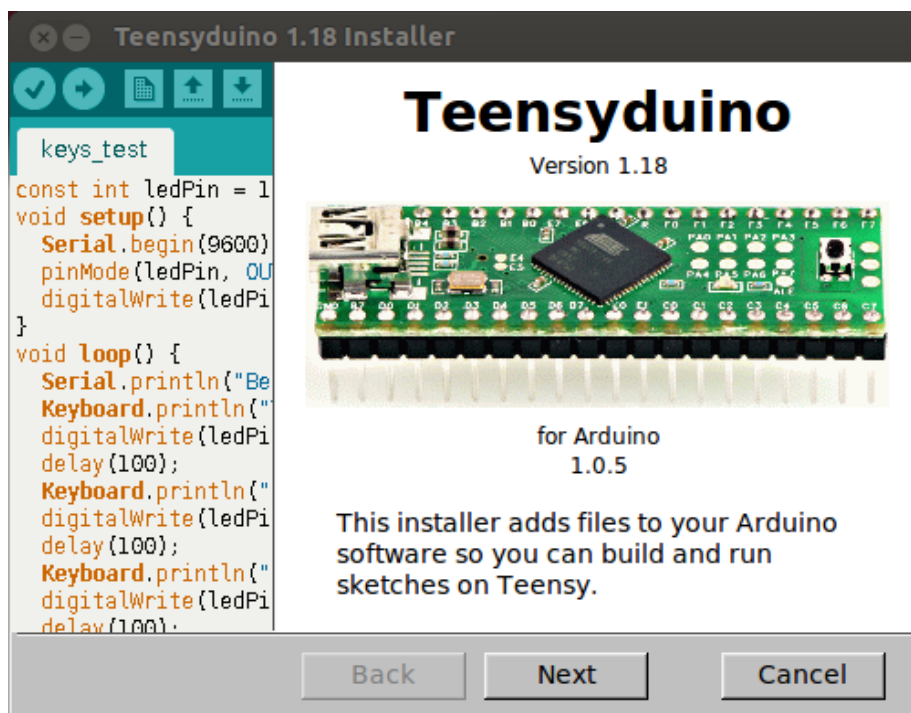
Następnie, pobieramy teensyduino ze strony autorów w wersji zgodnej z naszym systemem(32 / 64 bity) i zapisujemy w dogodnej dla nas lokalizacji. Aby móc uruchomić program, otwieramy terminal (ctrl + alt + t) i przechodzimy do miejsca, gdzie znajduje się plik:

```
cd /home/nazwa_użytkownika/Lokalizacja/Pliku
```

Następnie za pomocą komendy *chmod* nadajemy prawa wykonywania pliku *teensyduino.32bit*

```
chmod 755 teensyduino.32bit
```

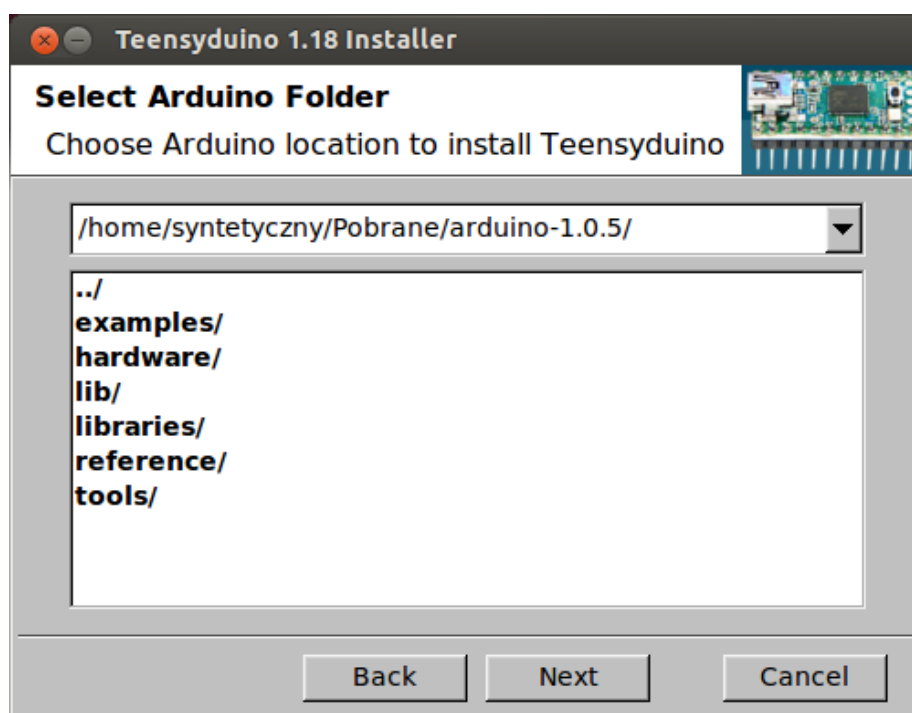
A następnie uruchamiamy instalator przy pomocy komendy *./teensyduino.32bit* i postępujemy zgodnie z poleceniami instalatora:



Ilustracja 25: Pierwsze okno instalatora, wciskamy Next

UWAGA!

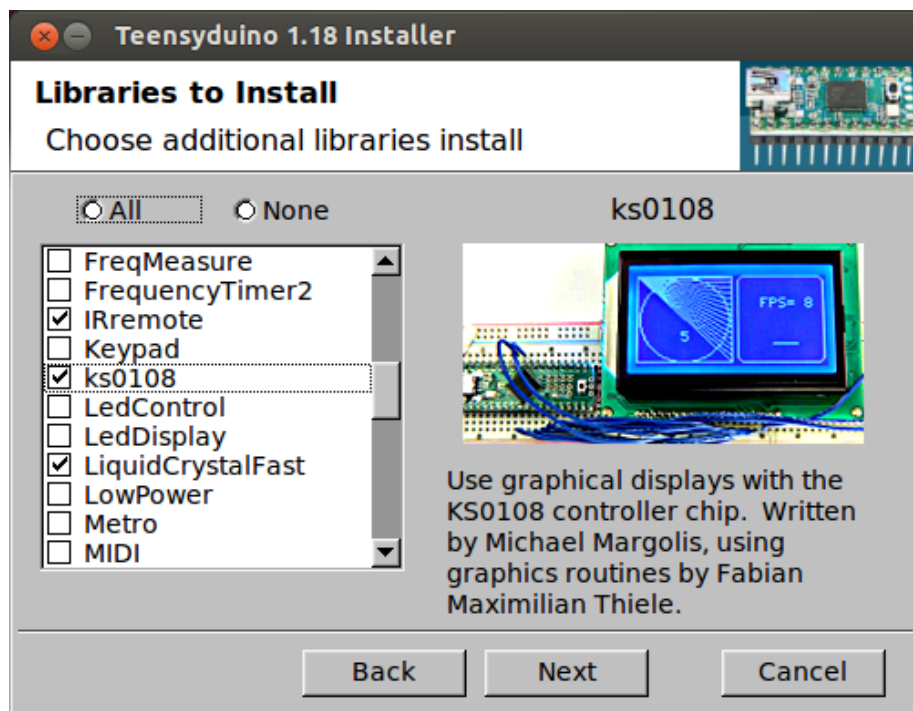
Wybieramy lokalizację instalacji (czyli miejsce rozpakowania) aplikacji Arduino.



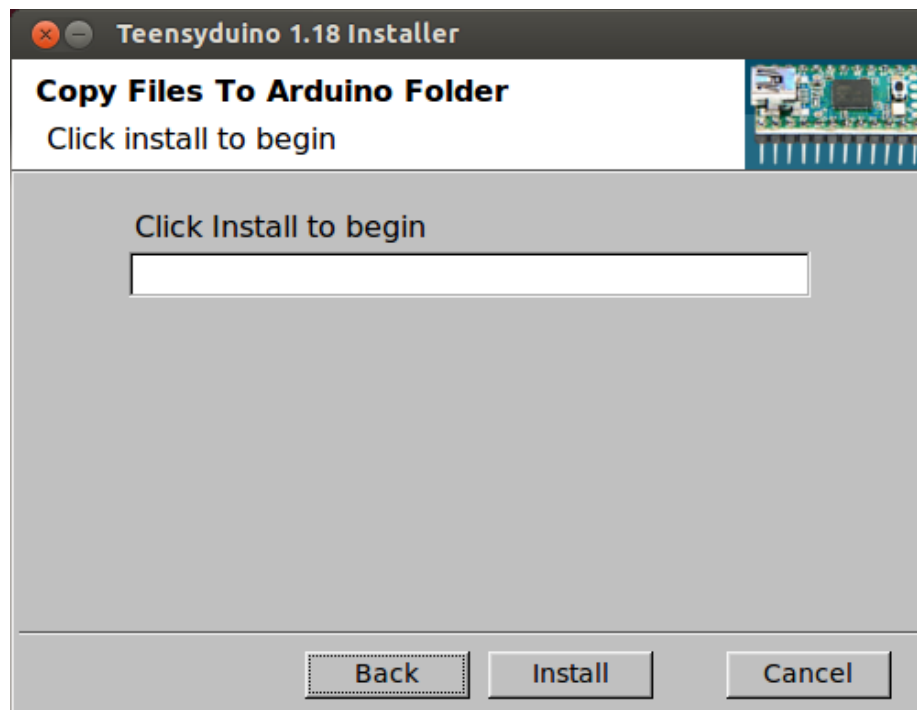
Ilustracja 26: Wybranie lokalizacji programu Arduino

Ważny krok instalacji!

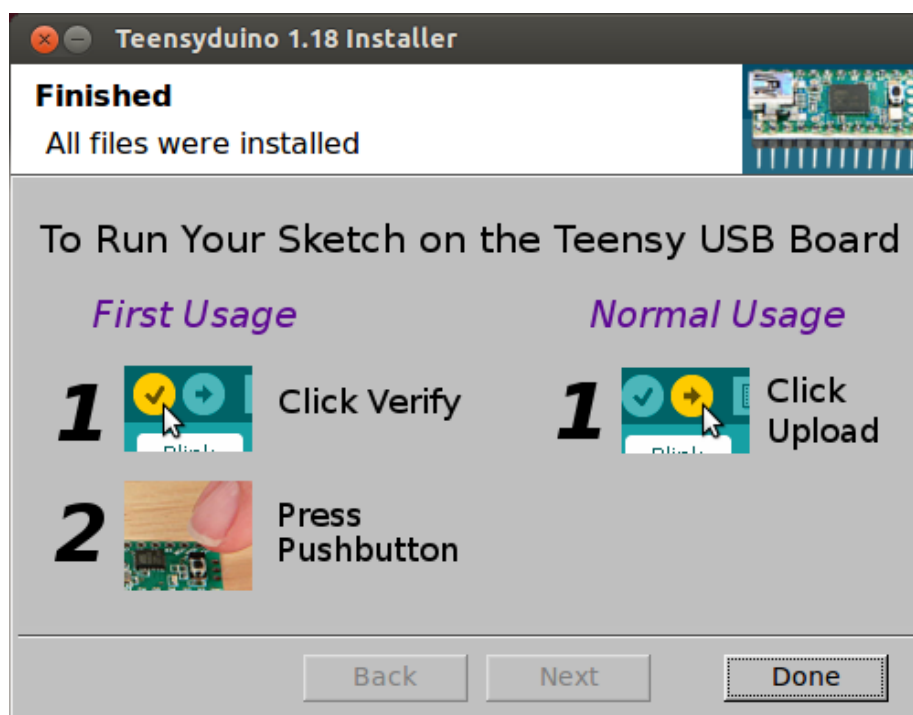
Teraz wybieramy biblioteki, potrzebne do poprawnego działania Arduino. Pamiętaj, aby wybrać IRremote, js0108, LiquidCrystalFast.



Ilustracja 27: Wybieramy potrzebne biblioteki i wciskamy Next



Ilustracja 28: Wystarczy wcisnąć Install.



Ilustracja 29: Wciskamy Done kończąc tym samym instalację.

Kolejnym krokiem przy instalacji programów jest zainstalowanie odpowiednika programu Flip do programowania procesorów serii *AT90USB*. Wciskając na klawiaturze *Ctrl + Alt + T* wywołujemy terminal, w który wklejamy za pomocą prawego klawisza myszki komendę:

```
sudo apt-get install dfu-programmer
```

Instalator sam pobierze niezbędną aplikację z internetu i zainstaluje program na dysku naszego komputera. I w ten oto sposób, zakończyliśmy instalację niezbędnych sterowników/bibliotek służących do kompilacji programu Marlin dla sterownika KOS.

5 Kompilacja i wgranie programu Marlin do sterownika KOS

W tym dziale zostanie ogólnie opisana kompilacja programu Marlin. Jest to spowodowane tym, że ustawienia są różne dla różnych konfiguracji sprzętowych. Dodatkowo dział ten zostanie podzielony na 2 sekcje Windows i Ubuntu, ze względu na pewne różnice podczas wgrywania pliku binarnego hex do pamięci mikroprocesora.

5.1 Edycja Marlina

Pobrane wcześniej pliki ze strony autorów Marlina 5 rozpakowujemy do katalogu, w którym będziemy edytować program. Następnie przy pomocy dowolnego narzędzia edytorskiego (np. Notepad++), otwieramy plik *Configuration.h* i przychodzimy do linijki *#define BAUDRATE 250000* gdzie zamiast 250000 wpisujemy 115000.

Następnie przechodzimy do linijki *#define MOTHERBOARD 7* i zmieniamy 7 na 8. Tym samym zmieniając domyślną konfigurację dla płyty sterownika drukarki ULTIMAKERa, na płytkę teensylu, z którą nasz sterownik jest kompatybilny (ze względu na użycie tego samego procesora).

Kolejnym krokiem będzie wybór termistorów jakie używamy w naszej konfiguracji. Standardowo, w większości głowic stosuje się termistory wysokotemperaturowe firmy EPCOS. Przejdźmy zatem do linijki, która zaczyna się tak:

```
#define TEMP_SENSOR_0 -1  
  
#define TEMP_SENSOR_1 -1  
  
#define TEMP_SENSOR_2 0  
  
#define TEMP_SENSOR_BED 0
```

w każdym kolejnym wierszu mamy definicję użytego termistora dla 3 głowic, oraz dla grzanego stołu. Wartości jakie nas interesują to 1 dla TEMP_SENSOR_0 oraz jeżeli używamy podgrzewanego stołu TEMP_SENSOR_BED.

```
#define TEMP_SENSOR_0 1  
  
#define TEMP_SENSOR_1 0  
  
#define TEMP_SENSOR_2 0  
  
#define TEMP_SENSOR_BED 1
```

W ten sposób, zdefiniowaliśmy termistory, używane w naszych drukarkach. Kolejnymi parametrami, którymi użytkownik powinien się zainteresować, to konfiguracja silników, a dokładniej kroki na mm oraz kierunek obrotu silników. Są to parametry, które mają znaczenie już po uruchomieniu drukarki, ale warto o nich wspomnieć już teraz.

Kierunek ruchu silników jest zdefiniowany w linijce:

```
#define INVERT_X_DIR true  
  
#define INVERT_Y_DIR false  
  
#define INVERT_Z_DIR true  
  
#define INVERT_E0_DIR false  
  
#define INVERT_E1_DIR false  
  
#define INVERT_E2_DIR false
```

Każdy wiersz opisuje kolejno ruch w osiach *X,Y,Z* oraz kierunek obracania się ekstrudera *E0,E1,E2*. W większości przypadków mamy do czynienia z 1 ekstruderem *E0*.

Wartości *true* oraz *false*, odpowiadają za kierunek obrotu silnika. Jeżeli w obecnej konfiguracji, silniki kręcą się w złym kierunku, zmieniamy *true* na *false* bądź odwrotnie w zależności, co było w danej osi zdefiniowane. Aby sprawdzić, czy ekstruder kręci się w poprawnym kierunku, wystarczy wpisać w okienku komend *M302* i wciskając *extrude* sprawdzić czy w poprawnym kierunku kręci się silnik wylączarki.

Następnie przejdziemy do wiersza odpowiadającego za przypisanie ilości kroków na mm.

```
#define DEFAULT_AXIS_STEPS_PER_UNIT {78.7402,78.7402,200.0*8/3,760*1.1}
```

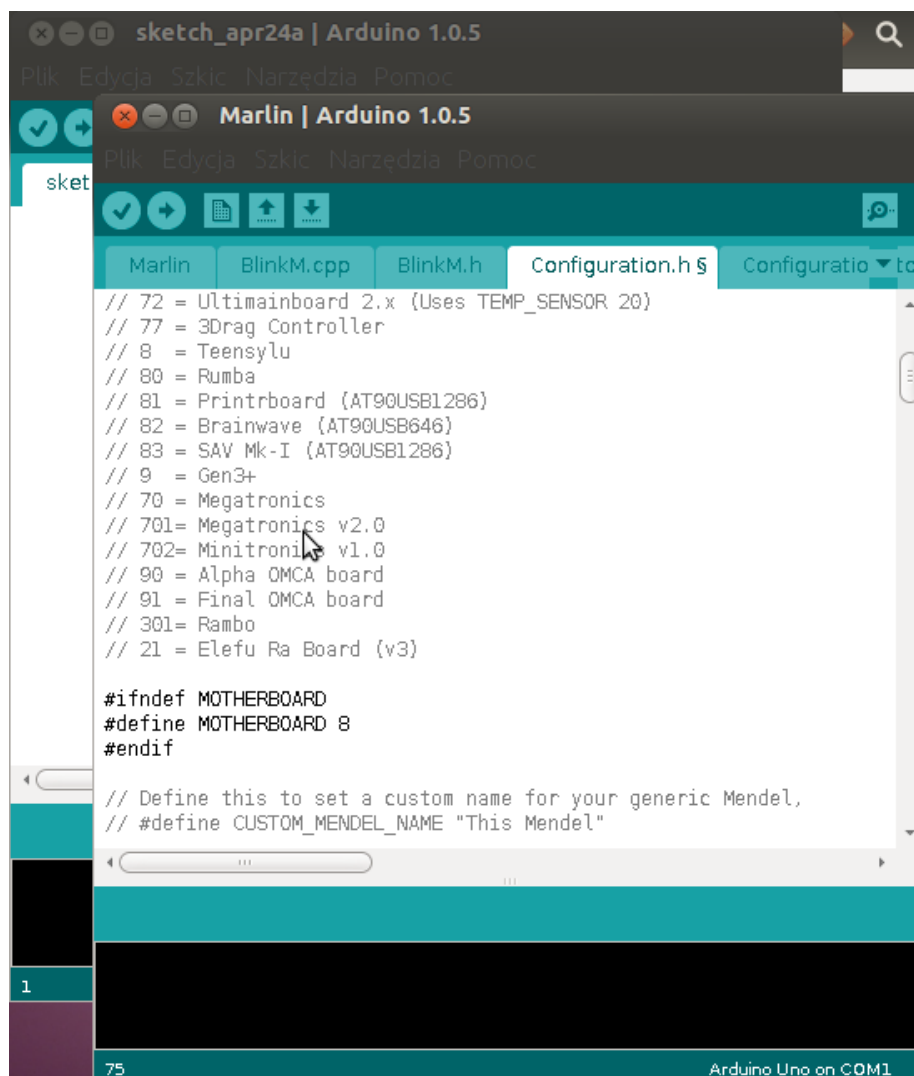

Powyższa linijka odpowiada za przypisanie ilości kroków na 1 mm w osiach {X , Y, Z, E}, których wartość można w łatwy sposób wyliczyć, korzystając z kalkulatora *Josefa Prusy* 6 .

Mając tak przygotowany plik, możemy go zapisać i przejść do programu *Arduino*.

5.2 Kompilacja Marlina

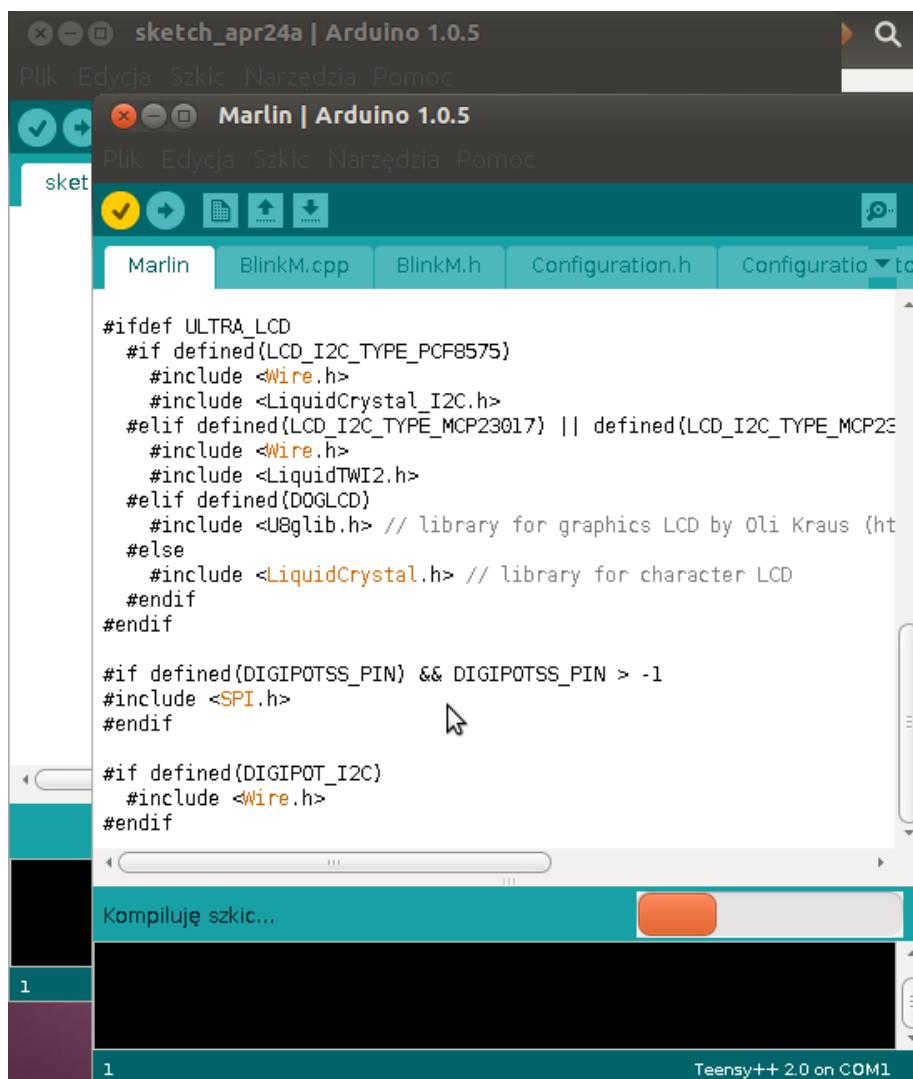
Otwarcie projektu Marlin wymaga od nas wybrania *Plik* → *Otwórz...* i wybranie pliku *Marlin.ino*.

Po czym otworzy nam się projekt *Marlin* jak na ilustracji poniżej 30.



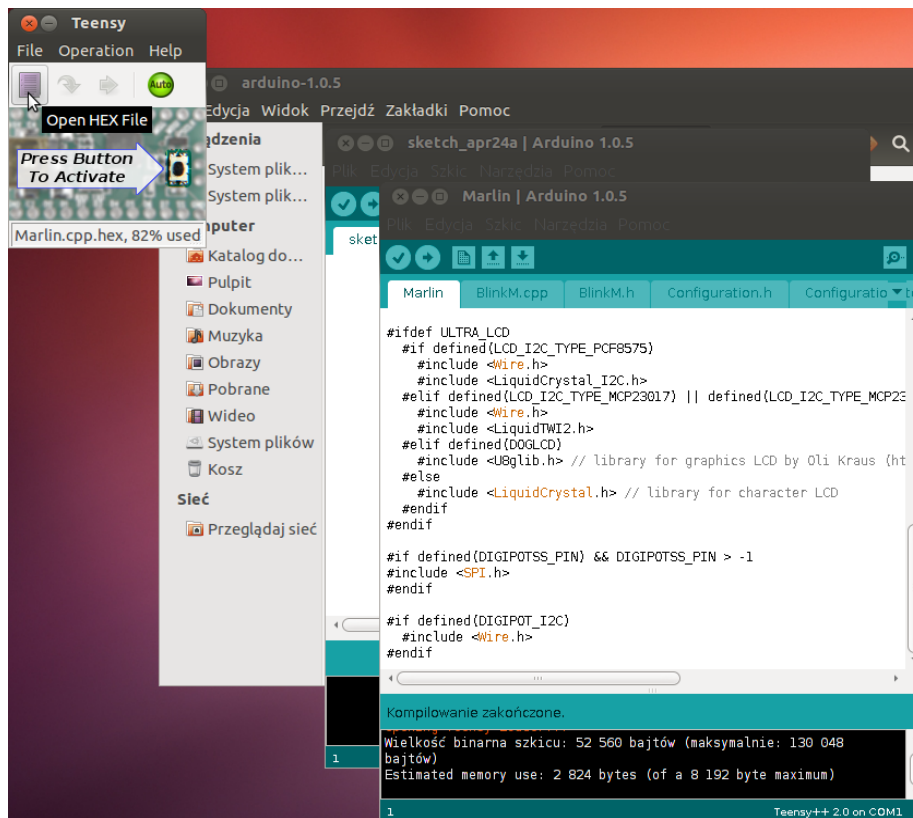
Ilustracja 30: Okno Arduino

Wszystkie ważne informacje w projekcie na które mamy wpływ, zostały zawarte w pliku *Configuration.h* i dotyczą podstawowej konfiguracji maszyny. W zakładce *Narzędzia* → *Board* należy wybrać *Teensy2.0++* i przejść do kompilacji projektu poprzez kliknięcie lewym klawiszem myszy w ptaszka znajdującego się w lewym górnym rogu aplikacji jak na ilustracji poniżej 31.



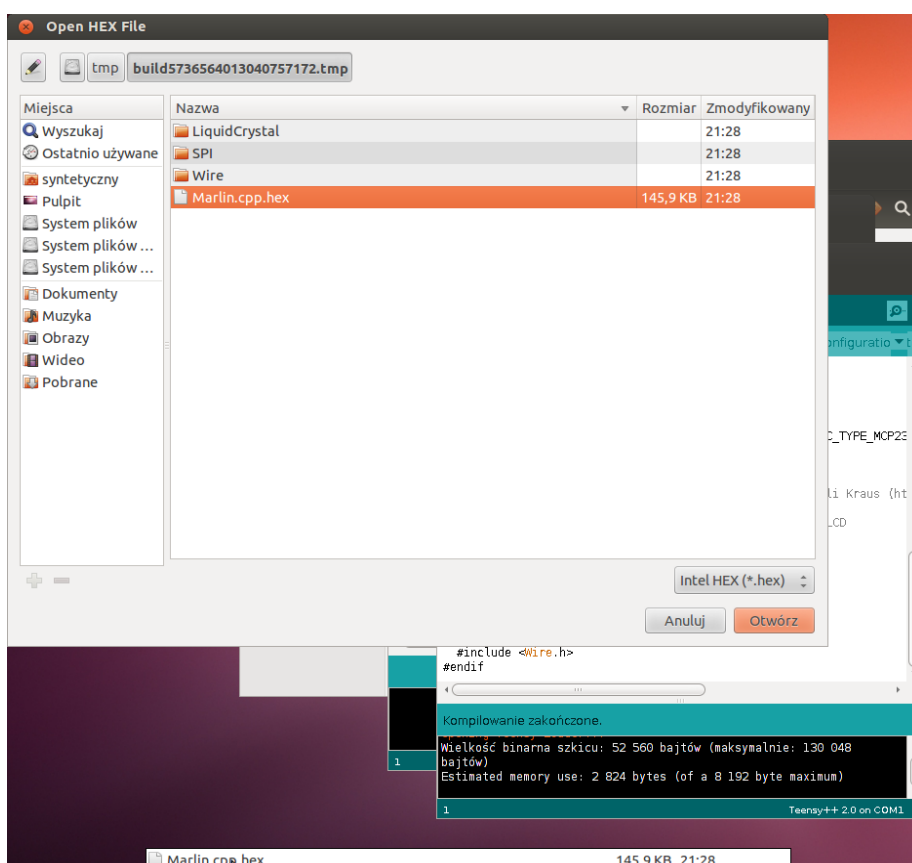
Ilustracja 31: Rozpoczęcie kompilacji Marlina

Następnie po ukończeniu kompilacji powinno pojawić się okno z wyborem programu pod nazwą *Marlin.cpp.hex* 32.



Ilustracja 32: Wybór pliku binarnego

Wybierając ikonę poniżej zakładki *File* , albo wybierając *File* → *Open Hex File* wybieramy plik *Marlin.cpp.hex* i przeciągamy na pulpit lub w inne dogodne dla nas miejsce i opuszczamy 33.

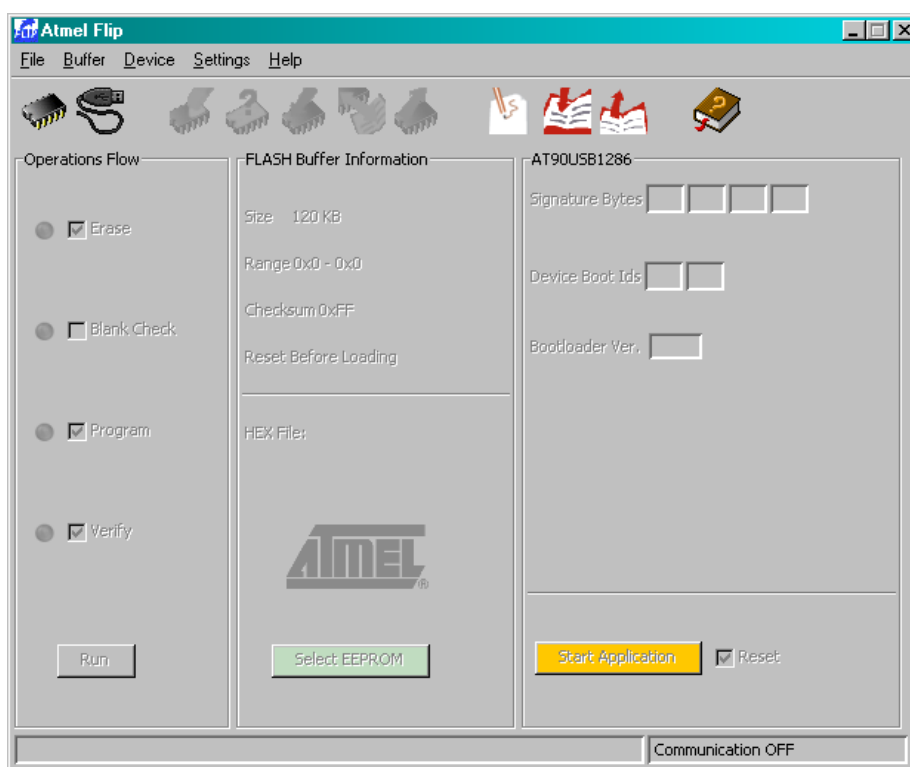


Ilustracja 33: Przeciąganie pliku *Marlin.cpp.hex*

5.3 Wgranie *Marlin.cpp.hex* do pamięci sterownika KOS na systemie Windows

Wgranie programu do sterownika odbywa się za pośrednictwem oprogramowania *Flip*. Zanim jednak przejdziemy do programowania podłączamy sterownik KOS do komputera, wyciągamy zworkę *BOOT* i wciskamy guzik reset przez 5sec w celu wprowadzenia sterownika w stan programowania. Jeżeli robimy to pierwszy raz, system poinformuje nas o nowym sprzęcie. Instalacja sterowników przebiega tak samo jak w przypadku innych urządzeń typu *USB*, z tym wyjątkiem, że podczas instalacji wskazujemy konkretny folder ze sterownikiem w folderze *Atmel/Flip 3.4.7/usb*.

Po instalacji zakończonej sukcesem przechodzimy do programu Flip.

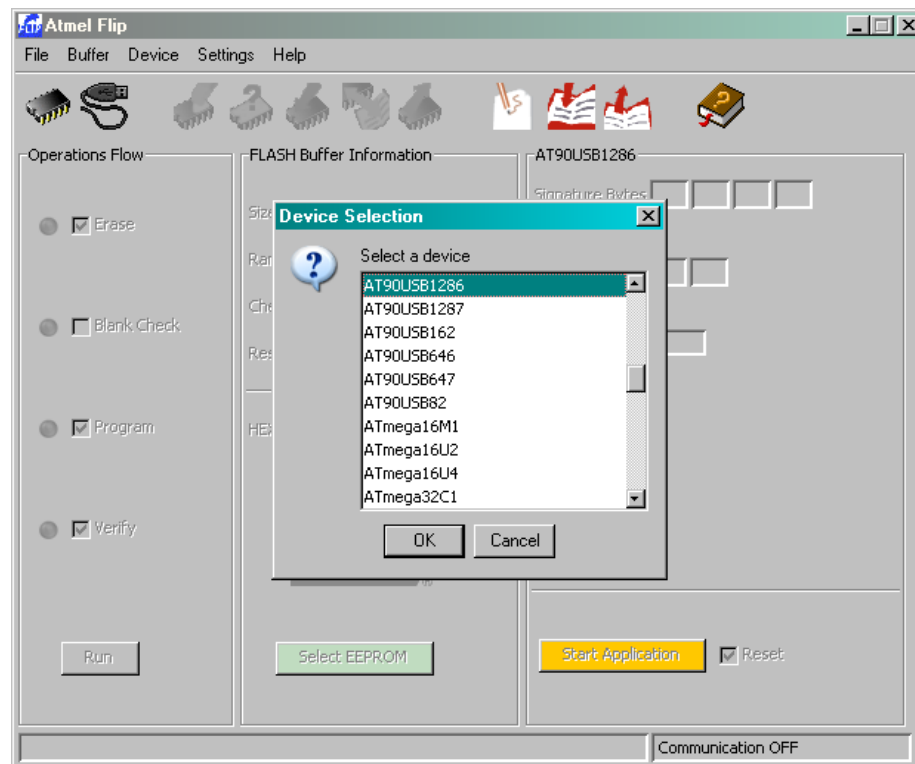


Ilustracja 34: Główne okno programu Flip

Program wymaga wskazania procesora, który będziemy programować, odbywa się to za pomocą pierwszej ikony z lewej.

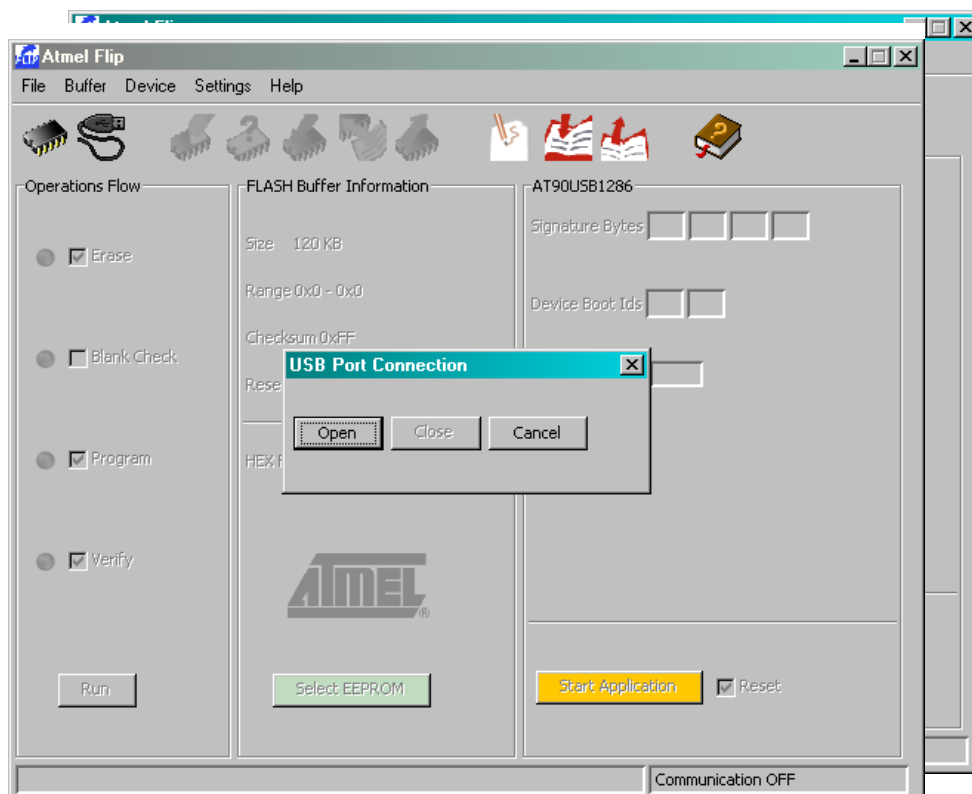
UWAGA!

Aby poprawnie wgrać program, upewnij się jaki procesor masz wlutowany na płytce. W zależności od dostępności może być użyty procesor *AT90USB1286* albo *AT90USB1287*.



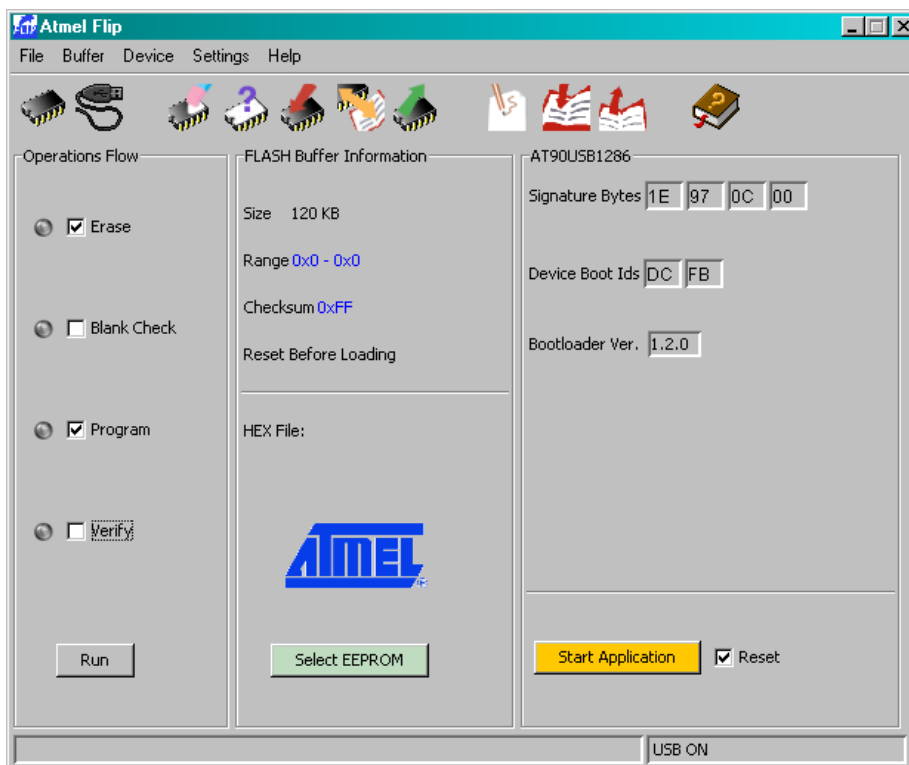
Ilustracja 35: Okno wyboru procesora

Następnie nawiązujemy połączenie USB za pomocą drugiej ikony od lewej.



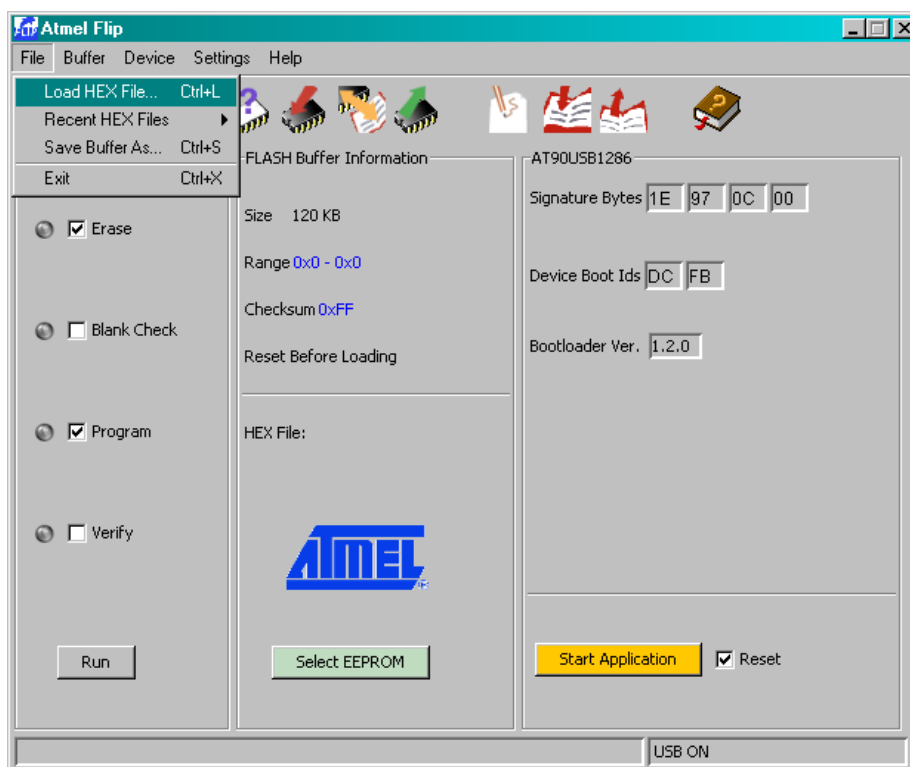
Ilustracja 37: Potwierdzenie otwarcia połączenia

Jeżeli urządzenie jest poprawnie zainstalowane program powinien nawiązać połączenie. Następnie powinniśmy wskazać kolejność operacji jakie chcemy wykonać. W tym przypadku wybieramy *Erase* i *Program* w oknie *Operations Flow*.



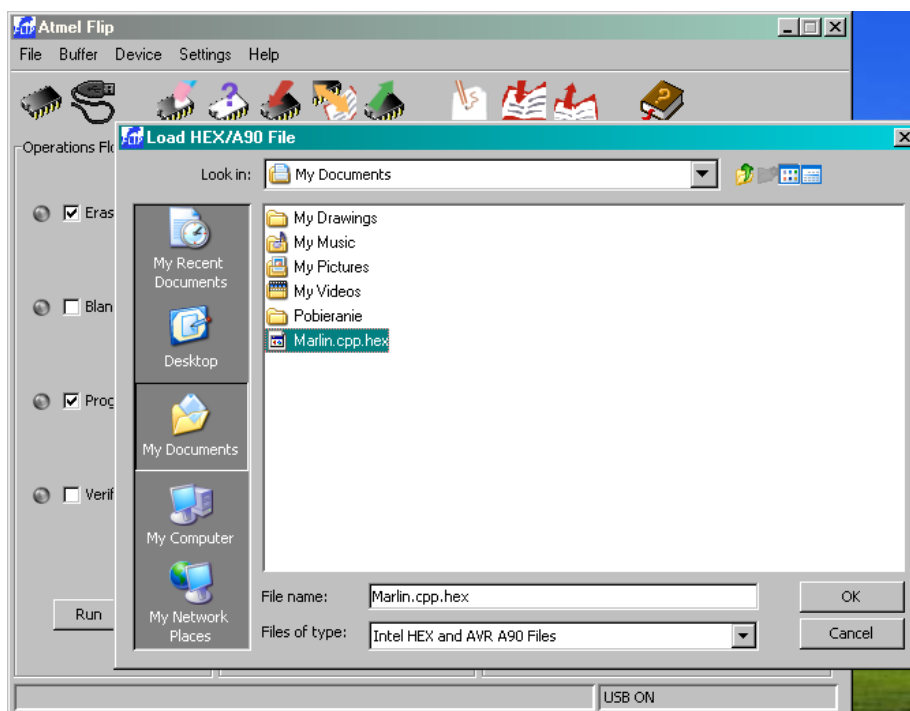
Ilustracja 38: Ustawienie kolejności operacji

Następnie wybieramy program jaki chcemy załadować do procesora.



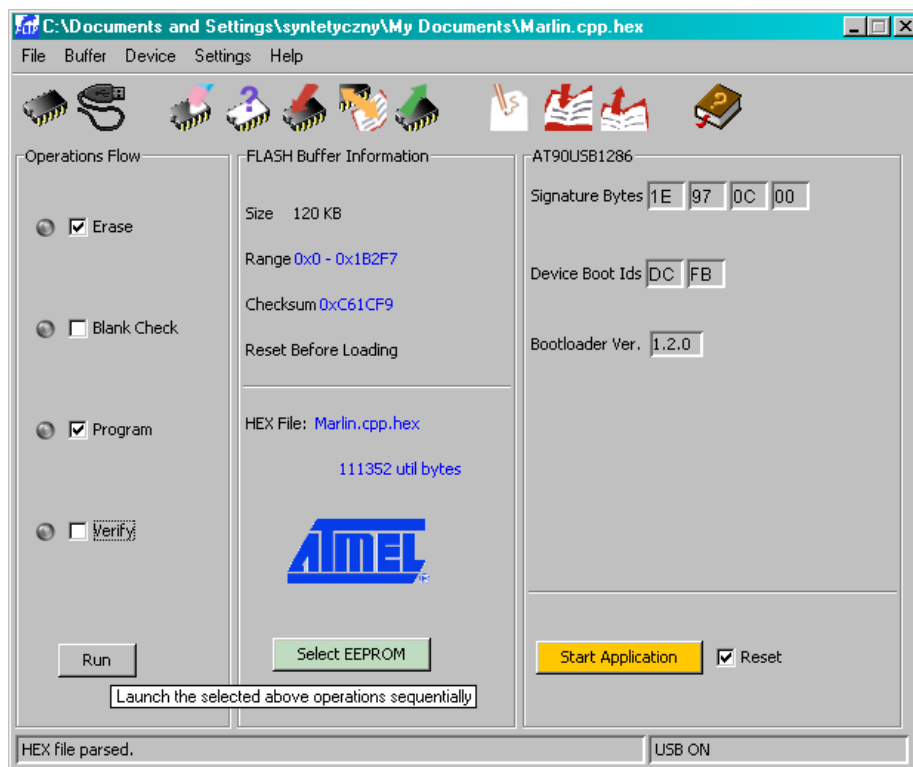
Ilustracja 39: Wybór programu do załadowania

W tym przypadku program został przeniesiony do Moich dokumentów.



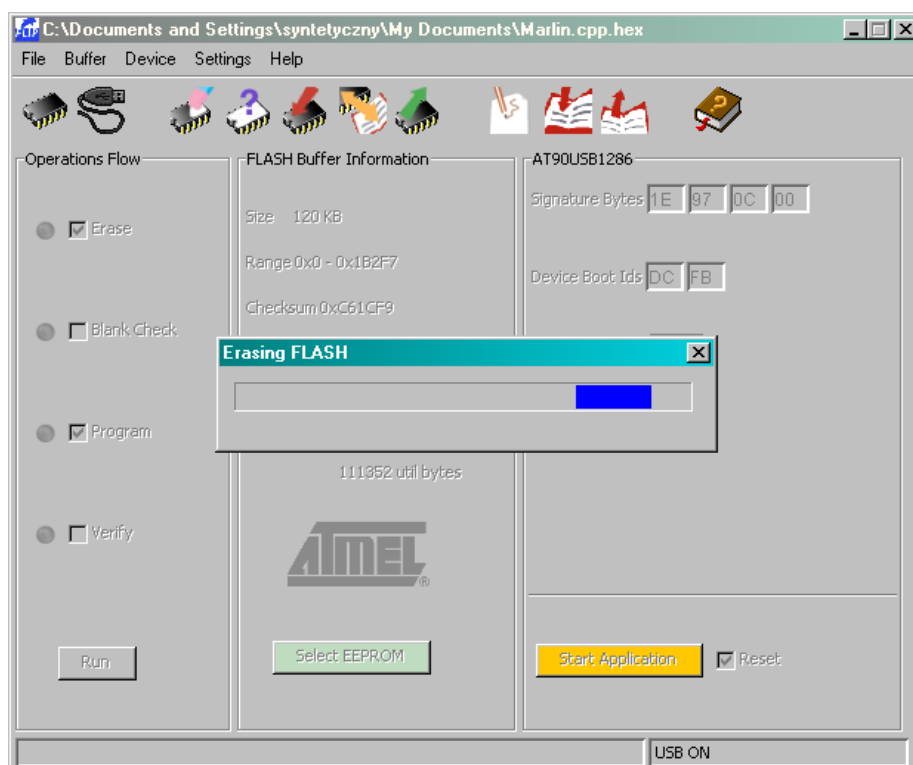
Ilustracja 40: Wybór programu do załadowania

Po wyborze poprawnego pliku *Marlin.cpp.hex* okno programu Flip będzie wyglądało w następujący sposób

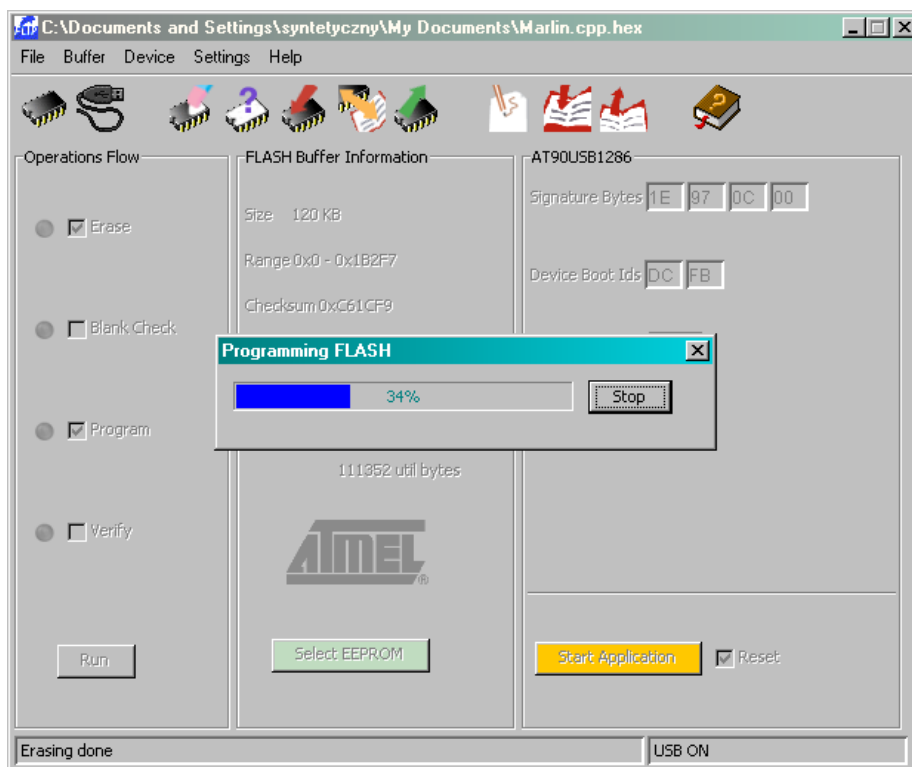


Ilustracja 41: Gotowy program Flip do wgrania oprogramowania Marlin do procesora

Klikając przycisk *Run*, program wpierw przeprowadzi czyszczenie procesora, a następnie przeprowadzi jego programowanie.

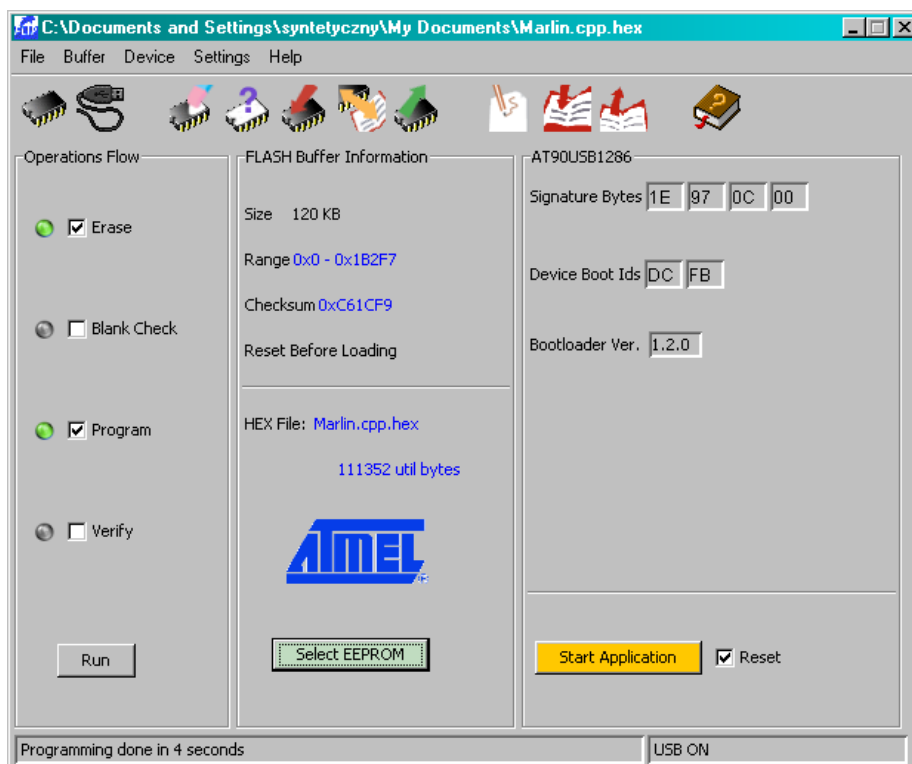


Ilustracja 42: Czyszczenie procesora



Ilustracja 43: Programowanie procesora AT90USB1286/7

Programowanie zakończone sukcesem!



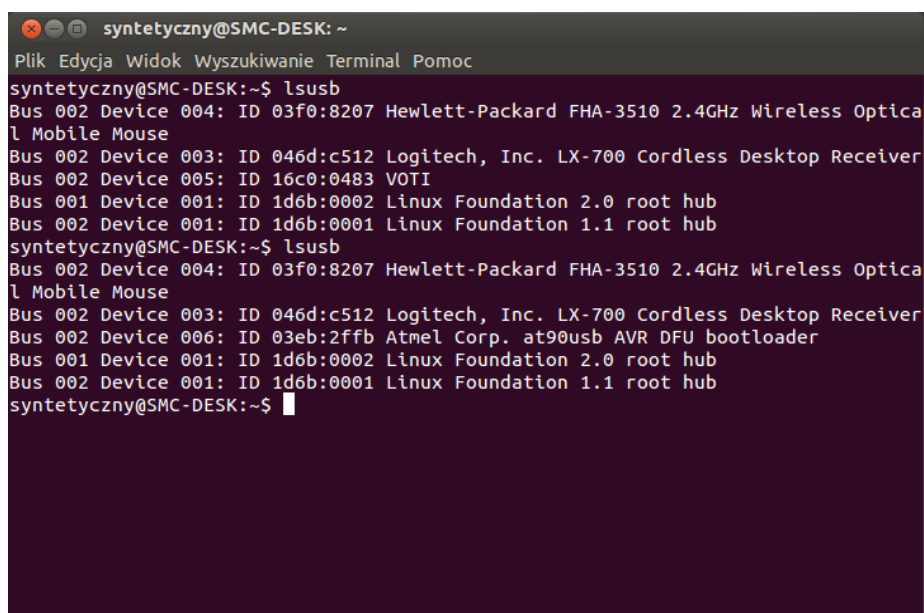
Ilustracja 44: Programowanie zakończone sukcesem!

Po ponownym założeniu zworki *BOOT* i jednokrotnym wciśnięciu guzika *Reset* sterownik powinien pojawić się jako port wirtualny. Tak zaprogramowany sterownik nadaje się już do używania z drukarką.

5.4 Wgranie *Marlin.cpp.hex* do pamięci sterownika KOS na systemie *Linux*

W celu zaprogramowania sterownika KOS przenosimy wpierw przygotowany plik *Marlin.cpp.hex* 33 do folderu głównego w systemie */home/nazwa_użytkownika/*. Następnie przywołując terminal za pomocą skrótu *Ctrl + Alt + T* i sprawdzamy czy plik jest w folderze głównym za pomocą komendy *dir -l Marlin.cpp.hex*.

Mając pewność, że jesteśmy we właściwym katalogu, podłączamy sterownik KOS do komputera, wyciągamy zworkę *BOOT* i wciskamy guzik reset przez 5 sekund w celu wprowadzenia sterownika w stan programowania. Korzystając z komendy *lsusb* sprawdzamy czy sterownik wszedł w stan programowania.



```
syntetyczny@SMC-DESK: ~
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
syntetyczny@SMC-DESK:~$ lsusb
Bus 002 Device 004: ID 03f0:8207 Hewlett-Packard FHA-3510 2.4GHz Wireless Optical Mobile Mouse
Bus 002 Device 003: ID 046d:c512 Logitech, Inc. LX-700 Cordless Desktop Receiver
Bus 002 Device 005: ID 16c0:0483 VOTI
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
syntetyczny@SMC-DESK:~$ lsusb
Bus 002 Device 004: ID 03f0:8207 Hewlett-Packard FHA-3510 2.4GHz Wireless Optical Mobile Mouse
Bus 002 Device 003: ID 046d:c512 Logitech, Inc. LX-700 Cordless Desktop Receiver
Bus 002 Device 006: ID 03eb:2ffb Atmel Corp. at90usb AVR DFU bootloader
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
syntetyczny@SMC-DESK:~$
```

Ilustracja 45: Sprawdzenie czy procesor jest w stanie programowania

Następnie, używając wcześniej zainstalowanego *dfu-programmer* czyścimy pamięć, a następnie wgrywamy program do pamięci procesora.

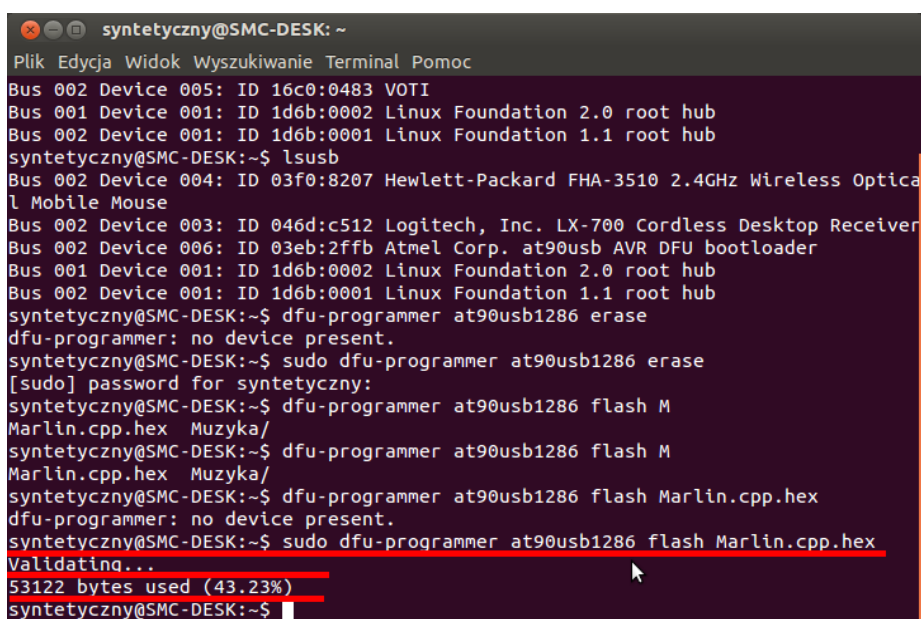
UWAGA!

Aby poprawnie wgrać program, upewnij się jaki procesor masz wlutowany na płytce. W zależności od dostępności może być użyty procesor *AT90USB1286* albo *AT90USB1287*.

```
sudo dfu-programmer at90usb1286 erase
```

```
sudo dfu-programmer at90usb1286 flash Marlin.cpp.hex
```

Poprawnie przeprowadzone programowanie powinno się skończyć jak na niżej przedstawionej ilustracji



```
syntetyczny@SMC-DESK: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
Bus 002 Device 005: ID 16c0:0483 VOTI  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
syntetyczny@SMC-DESK:~$ lsusb  
Bus 002 Device 004: ID 03f0:8207 Hewlett-Packard FHA-3510 2.4GHz Wireless Optical Mobile Mouse  
Bus 002 Device 003: ID 046d:c512 Logitech, Inc. LX-700 Cordless Desktop Receiver  
Bus 002 Device 006: ID 03eb:2ffb Atmel Corp. at90usb AVR DFU bootloader  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
syntetyczny@SMC-DESK:~$ dfu-programmer at90usb1286 erase  
dfu-programmer: no device present.  
syntetyczny@SMC-DESK:~$ sudo dfu-programmer at90usb1286 erase  
[sudo] password for syntetyczny:  
syntetyczny@SMC-DESK:~$ dfu-programmer at90usb1286 flash M  
Marlin.cpp.hex Muzyka/  
syntetyczny@SMC-DESK:~$ dfu-programmer at90usb1286 flash M  
Marlin.cpp.hex Muzyka/  
syntetyczny@SMC-DESK:~$ dfu-programmer at90usb1286 flash Marlin.cpp.hex  
dfu-programmer: no device present.  
syntetyczny@SMC-DESK:~$ sudo dfu-programmer at90usb1286 flash Marlin.cpp.hex  
Validating...  
53122 bytes used (43.23%)  
syntetyczny@SMC-DESK:~$
```

Ilustracja 46: Programowanie zakończone sukcesem

Po ponownym założeniu zworki *BOOT* i jednokrotnym wciśnięciu guzika *Reset* sterownik powinien pojawić się w jako *VOTI* po wpisaniu komendy *lsusb*. Tak zaprogramowany sterownik nadaje się już do używania z drukarką.

6 Źródła oraz dodatkowe konfiguracje dla sterownika KOS

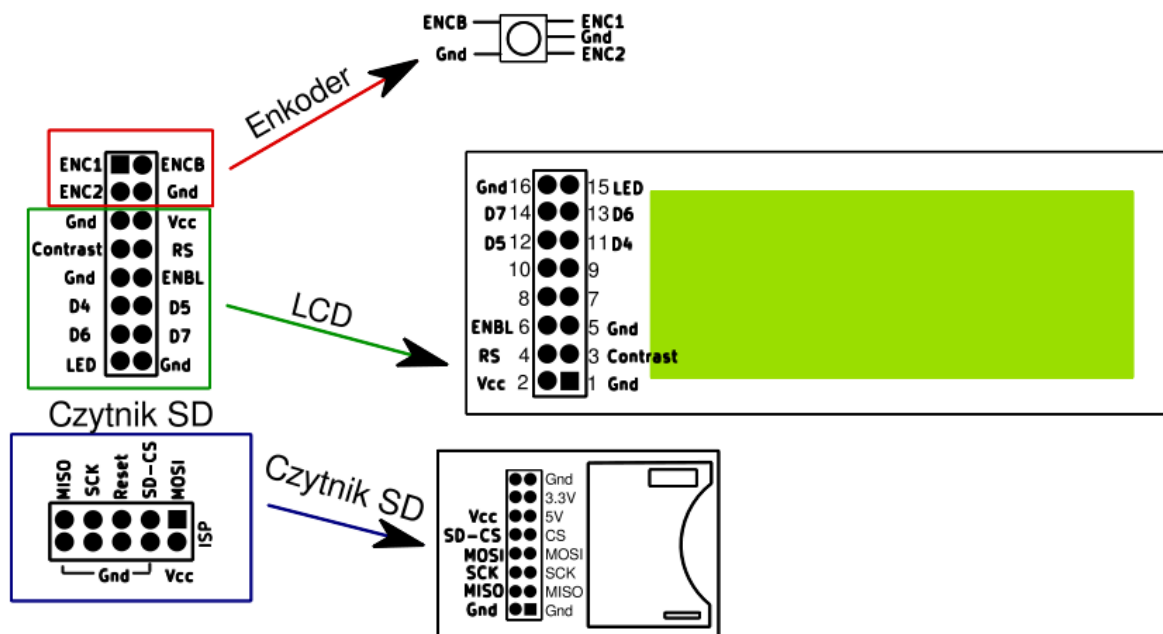
Oprócz Marlina można również używać oprogramowania Repetier. W tym celu zostały opracowane repozytoria github z forkami tego oprogramowania, które wymagają jedynie kompilacji z interesującą nas mechaniczną konfiguracją. Aby je pobrać wystarczy udać się pod adresy:

https://github.com/syntetyczny/Marlin/tree/Marlin_KOS

<https://github.com/syntetyczny/Repetier-Firmware>

Są tam już wstępnie skonfigurowane programy dla elektroniki KOS. Konfiguracje te przewidują podłączenie karty SD, enkodera oraz Ekranu LCD zgodnie ze schematem przedstawionym poniżej.

6.1 Schemat podłączenia LCD, Enkoder, SD-card



Ilustracja 47: Schemat podłączenia LCD, Enkodera, Czytnika SD

7 Bibliografia

1. serial_install.exe z dnia 10.02.2014, www.pjrc.com/teensy/serial_install.exe
2. Arduino installer z dnia 23.04.2014, <http://arduino.cc/en/Main/Software>
3. Teensyduino z dnia 23.04.2014, http://www.pjrc.com/teensy/td_download.html
4. Flip for Windows z dnia 23.04.2014, <http://www.atmel.com/tools/flip.aspx>
5. Marlin z dnia 30.04.2014, <https://github.com/ErikZalm/Marlin>
6. Kalkulator Josefa Prusa z dnia 30.04.2014, <http://calculator.josefprusa.cz/>