Warsztaty AVR

Instalacja i konfiguracja środowiska Eclipse dla mikrokontrolerów AVR

Dariusz Wika

SATLAND
PROTOTYPE

1.Krótki wstęp:

Eclipse to rozbudowane środowisko programistyczne, które dzięki możliwości instalowania wtyczek pozwala nam pisać programy w wielu językach oraz programować wiele różnych platform.

2.Instalacja

Instalacje Eclipse jak każdą inną należy zacząć od pobrania plików z Internetu albo skopiowania ich od kolegi. W tym miejscu pragnę zaznaczyć że oprogramowanie które zostanie przez nas użyte jest w pełni legalne.

Eclipse'a pobieramy ze strony projektu, zwracając przy tym uwagę na zgodność wersji naszego systemu operacyjnego z pobieraną wersja środowiska.

https://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr2

Oprócz samego Eclipse'a będzie nam potrzebne WinAVR oraz AVR GCC Toolchain, które znajdziemy na stronie warsztatów AVR naszego koła naukowego.

http://www.ue.eti.pg.gda.pl/~bpa/kolo_chip/warsztaty_avr/

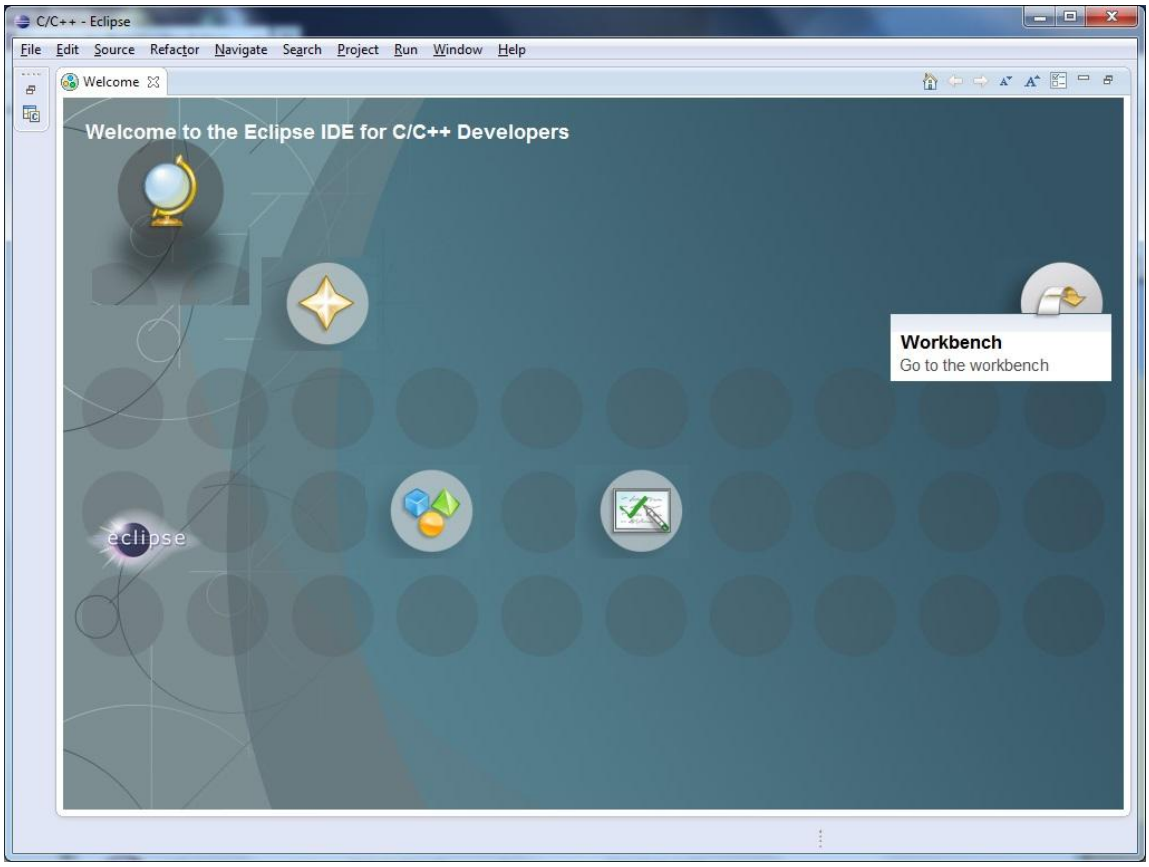
Po pobraniu środowiska rozpakowujemy go w wybranej przez nas lokalizacji. Tak nie przesłyszeliście się Eclipse nie wymaga instalacji, wystarczy go rozpakować. Po rozpakowaniu sprawdzamy czy uda nam się uruchomić środowisko. Okno przedstawiające poprawnie uruchamiające się środowisko Eclipse przedstawia rysunek 1. Jeśli Eclipse nie uruchomił się poprawnie na to 99% jest to wina niezgodności wersji środowiska z wersją systemu lub braku zainstalowanej Javy. W przypadku problemów z Javą problem rozwiązuje deinstalacja wcześniejszych wersji i instalacja wersji z godnej z Eclipse, jeśli korzystamy z Eclipsex64 powinniśmy zainstalować javę w wersji x64.

Okno **Workspace Launcher** służy do wyboru folderu w którym będą przechowywane nasze projekty, jeśli nie będziemy często przełączać folderów z projektami warto zaznaczyć opcję **Use is as the default and do not ask again.**

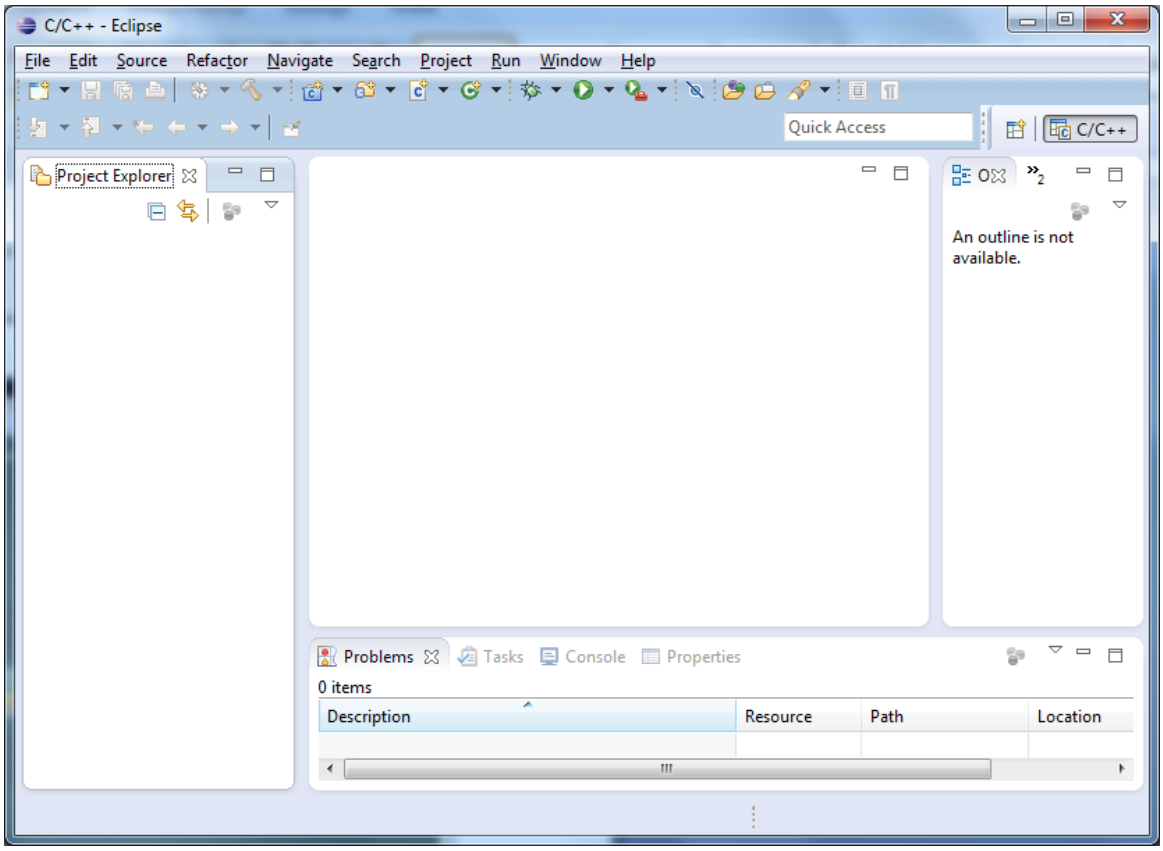
Po wskazaniu wybranego przez nas folderu naszym oczom ukaże się okno przedstawione na rysunku 2. Aby przejść do pisania programu wybieramy opcje **Go to the workbench**. Gotowe do pracy środowisko Eclipse przedstawia rysnek 3.

Kolejnym krokiem będzie instalacja WinAVR, integracja Eclipse z avrdude oraz zadbanie aby kompilator miał dostęp do wymaganych bibliotek.

Rysunek 1

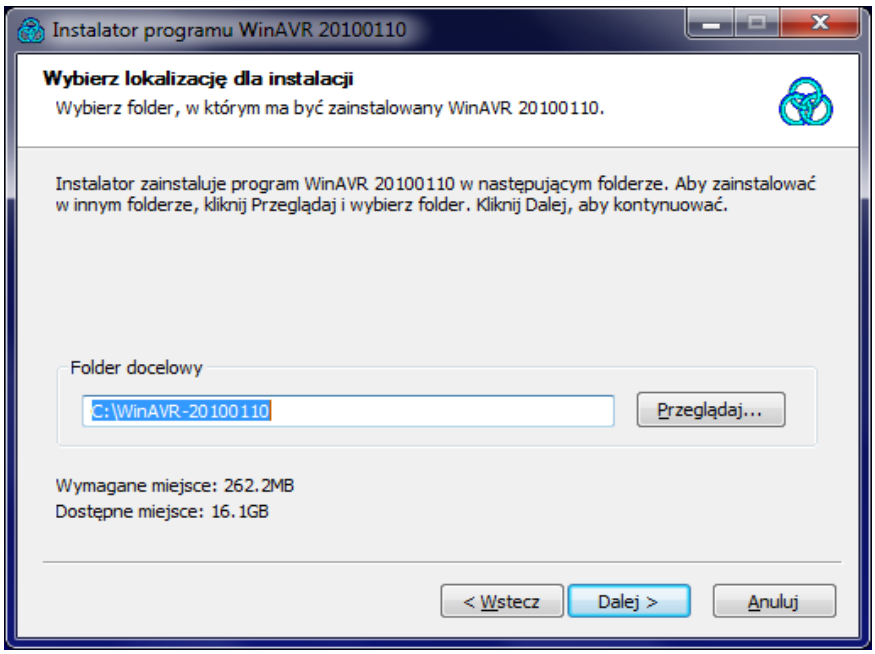


Rysunek 2



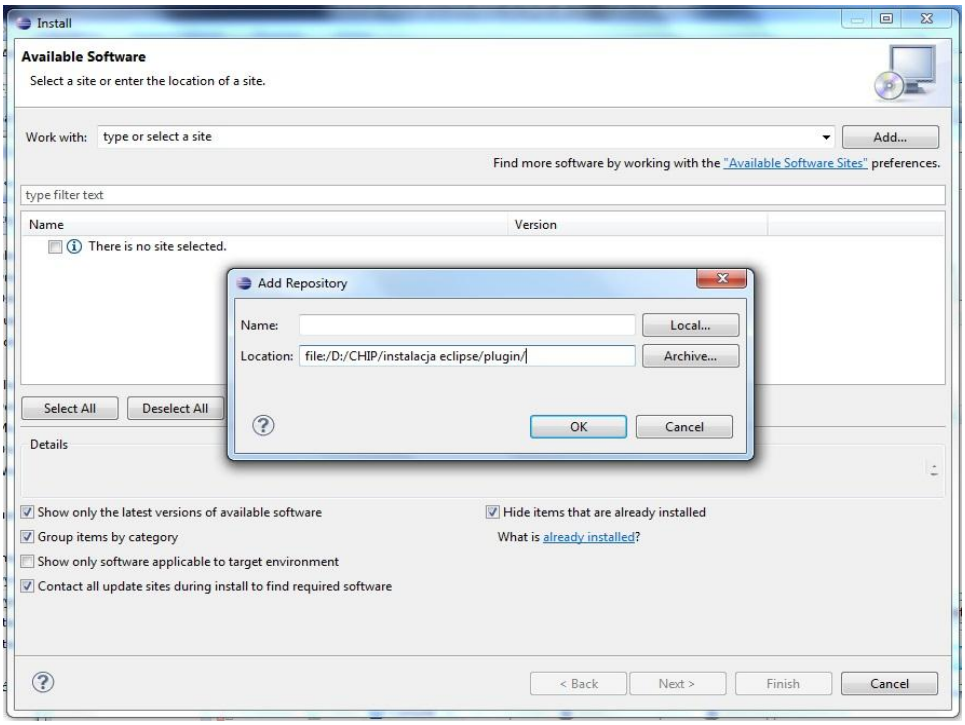
Rysunek 3

Aby uniknąć dodatkowych problemów WinAVR najlepiej zainstalować w domyślnej lokalizacji. Okno instalatora wraz z domyślnym folderem docelowym przedstawia rysunek 4. Pozostałem opcję możemy pozostawić bez zmian i postępować zgodnie z poleceniami instalatora.



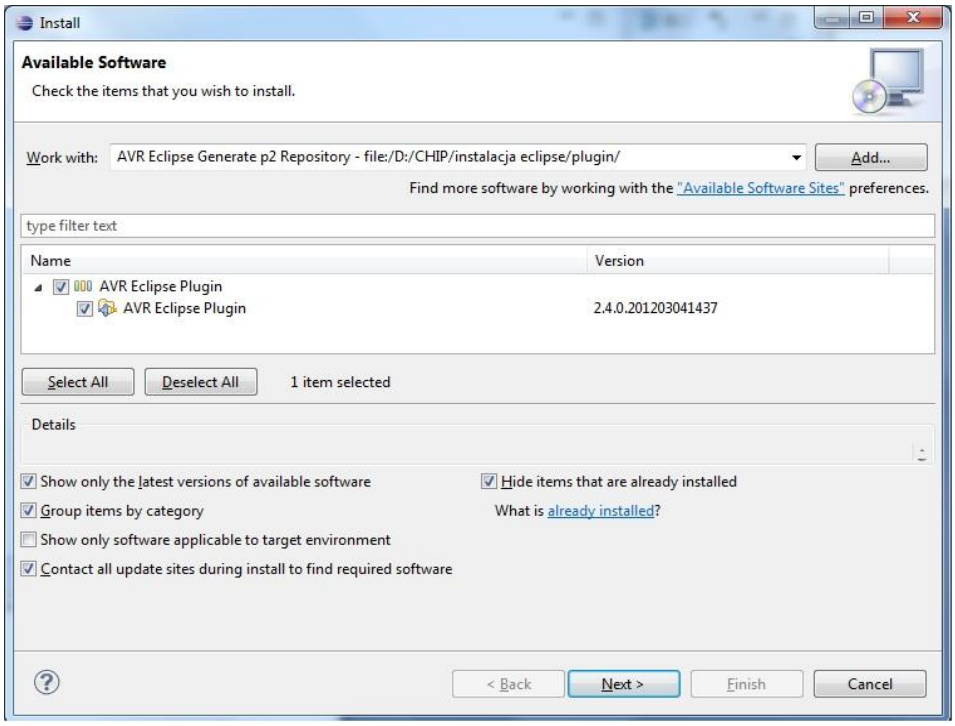
Rysunek 4

Następnie przejdziemy do instalacji PLUGIN'u do środowiska Eclipse. W temu celu w środowisku Eclipse z menu **Help** wybieramy opcje **Instal New Software….** Po pojawieniu się okna **Install** klikamy w przycisk **Add** i w oknie **Add Repository** klikając w przycisk **Local** wskazujemy ścieżkę do folderu z pobranym przez nas plugin'em w sposób pokazany na rysunku 5.



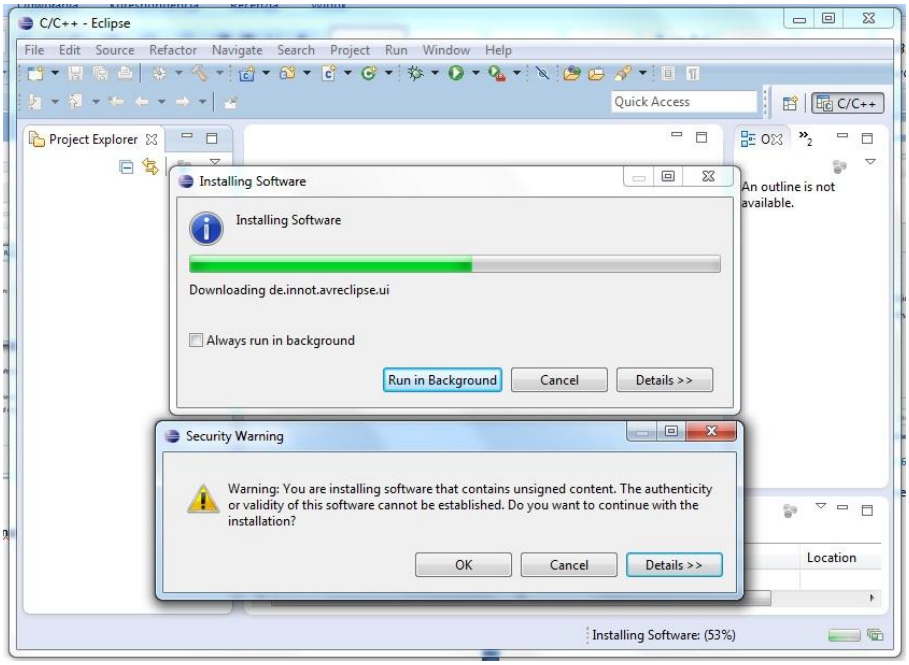
Rysunek 5

Po wciśnięciu przycisku **OK** okno **Install** powinno wyglądać w sposób pokazany na rysunku 6.



Rysunek 6

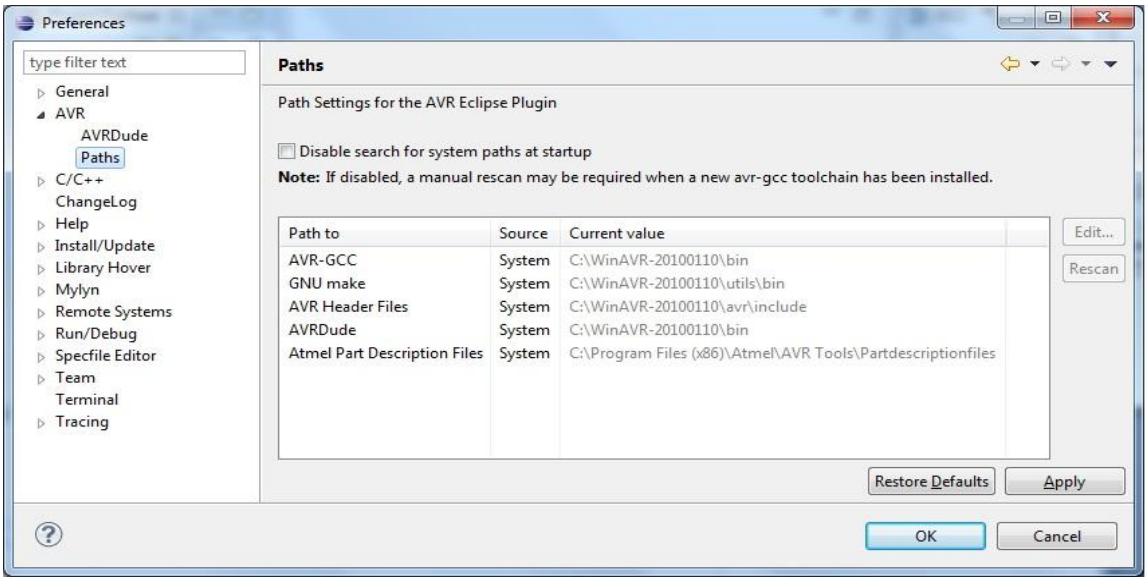
Po dwukrotnym wciśnięciu przycisku **Next >** zaakceptowaniu warunków licencji klikamy przycisk **Finish**, który rozpocznie przycisk instalacji. W trakcie instalacji powinien pojawić się komunikat pokazany na rysunku 7 aby kontynuować instalację należy wcisnąć przycisk **OK**.



Rysunek 7

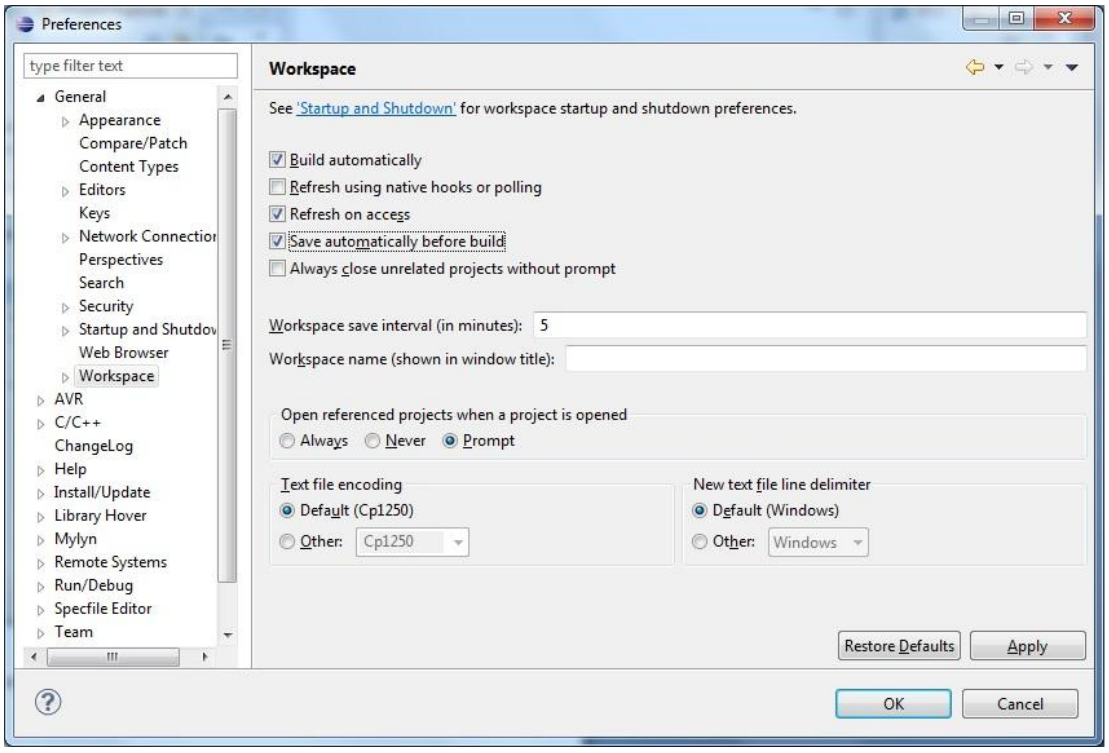
Po zakończeniu instalacji pojawi się okno **Software Updates** gdzie należy wcisnąć przycisk **Yes** aby ponownie uruchomić środowisko Eclipse.

Po ponownym uruchomieniu programu sprawdzimy czy ścieżki do avrdude i bibliotek są poprawne. W tym celu z menu **Window** wybieramy opcję **Preferences**. W oknie **Preferences** wybieramy kolejno opcję **AVR** i **Paths**, ścieżki powinny wyglądać tak jak na rysunku 8.



Rysunek 8

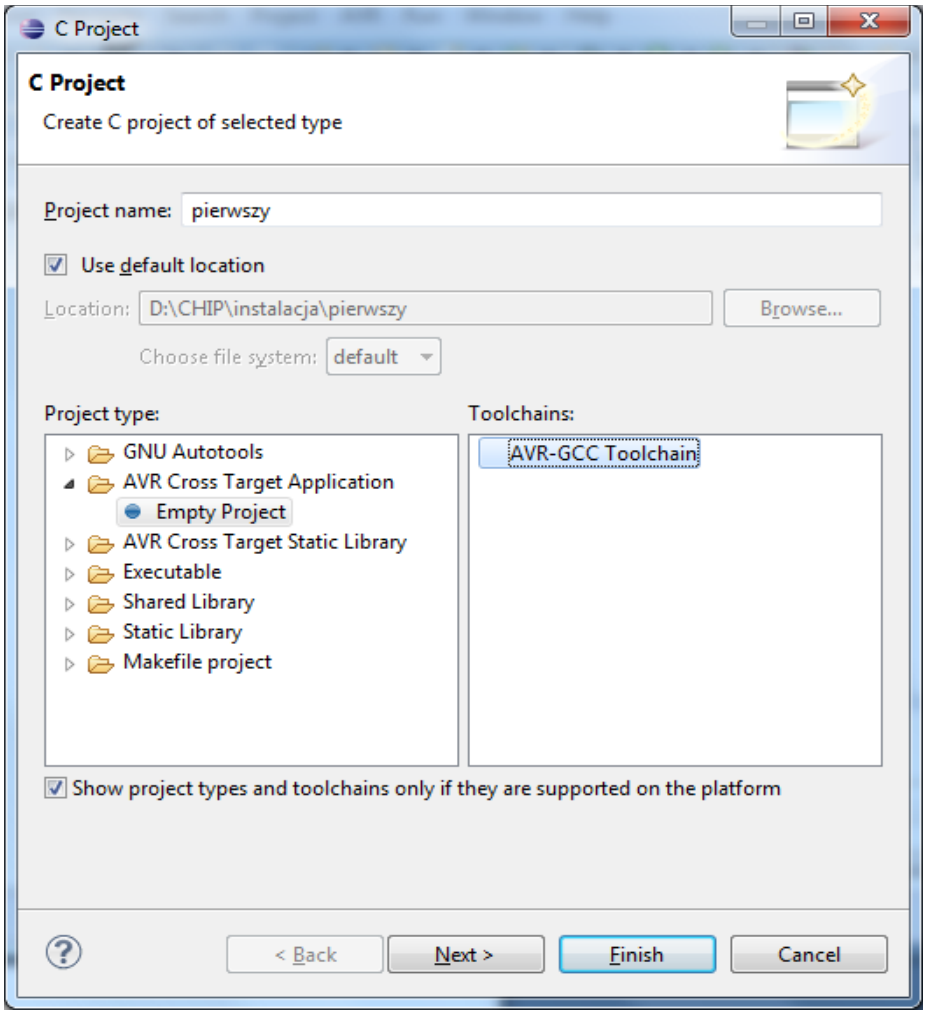
Kolejną przydatną opcja jest włączenie automatycznego zapisywaniu pliku przed kompilacją, w tym celu z w tym samym oknie w wybieramy kolejno opcję **General** i **Workspace** i zaznaczamy opcję **Save automatically before build** w sposób pokazany na rysunku 9. Zmiany potwierdzamy klikając **Apply** i zamykając okno przyciskiem **OK.**



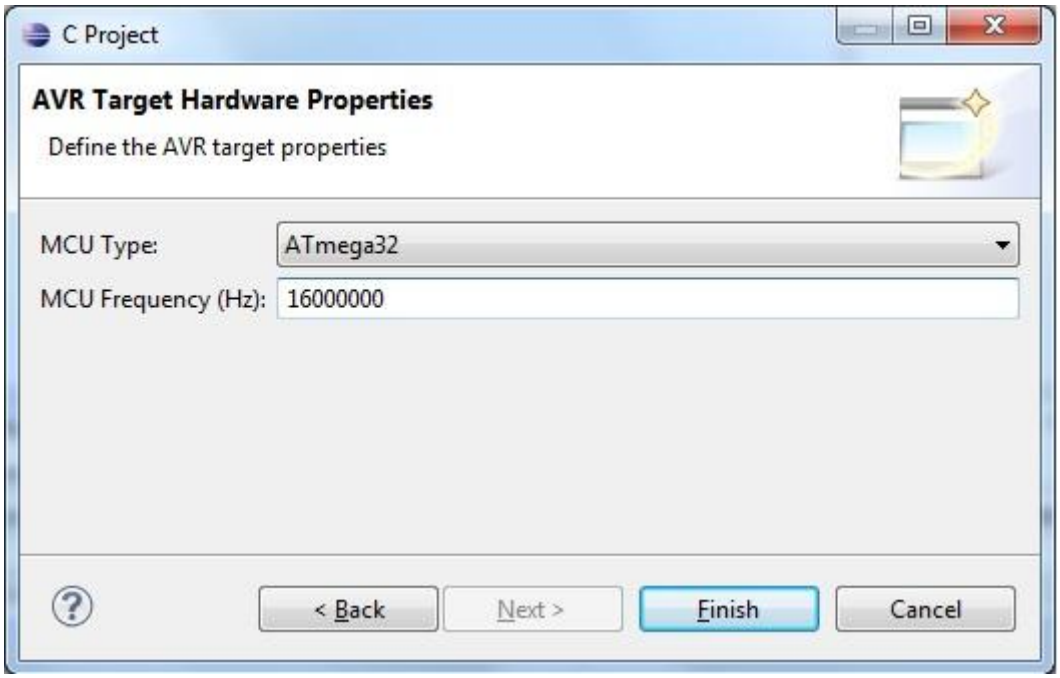
Rysunek 9

Tak przygotowane środowisko powinno być już gotowe do pracy teraz zajmiemy się przygotowanie pierwszego projektu.

Z menu **File** wybieramy opcję **New** i klikamy **C Project**. Nowe okno **C Projekt** konfigurujemy w sposób pokazany na rysunku 10. Następnie wciskamy przycisk **Next >** w kolejnym oknie odznaczamy opcję **Debug** i ponownie klikamy **Next >**, ostatnim etapem tworzenia projektu jest wybór programowanego układu i jego częstotliwości taktowania pokazany na rysunku 11. Po dokonaniu wyboru klikamy **Finish** aby zakończyć tworzenie projektu.

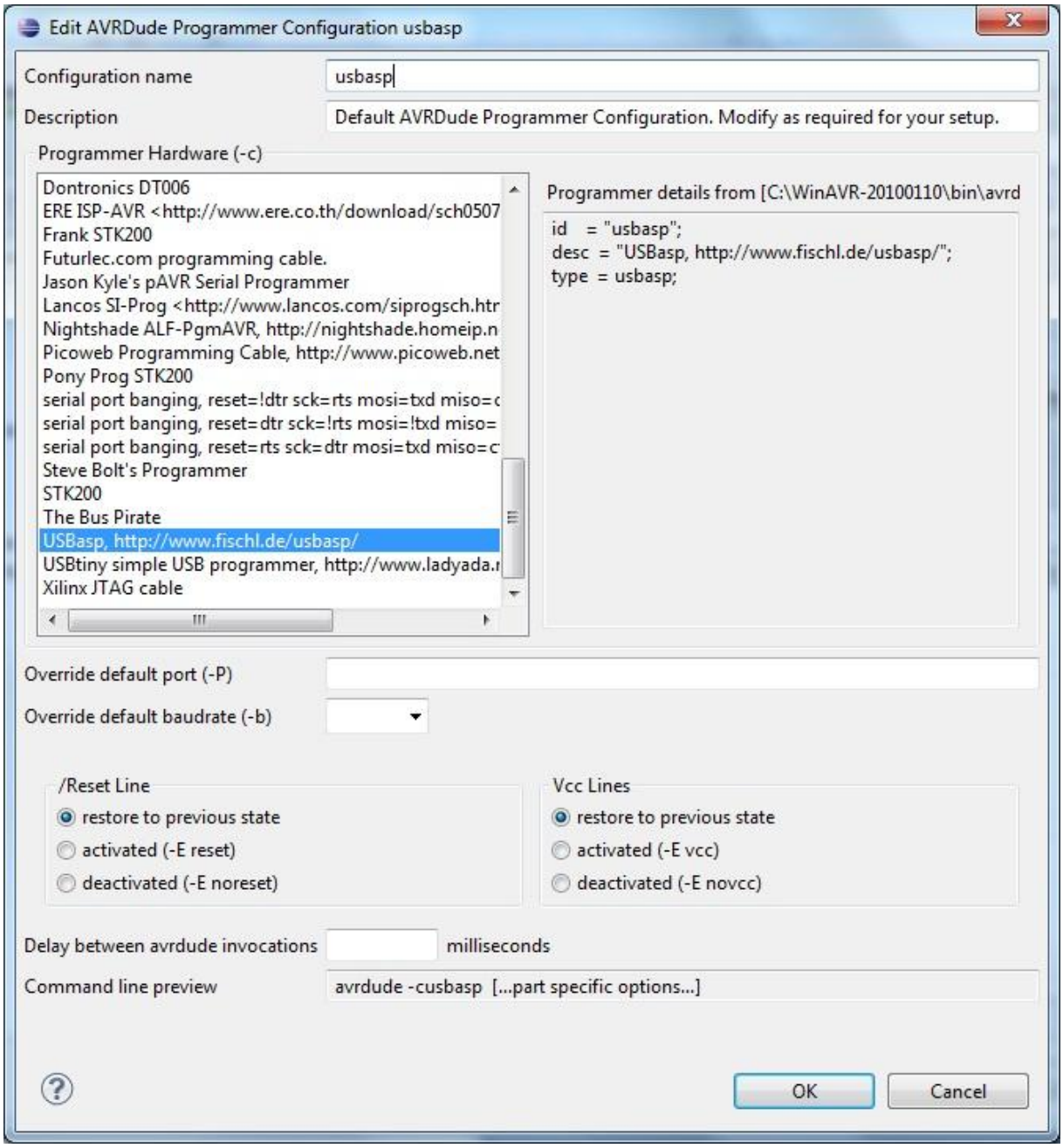


Rysunek 10



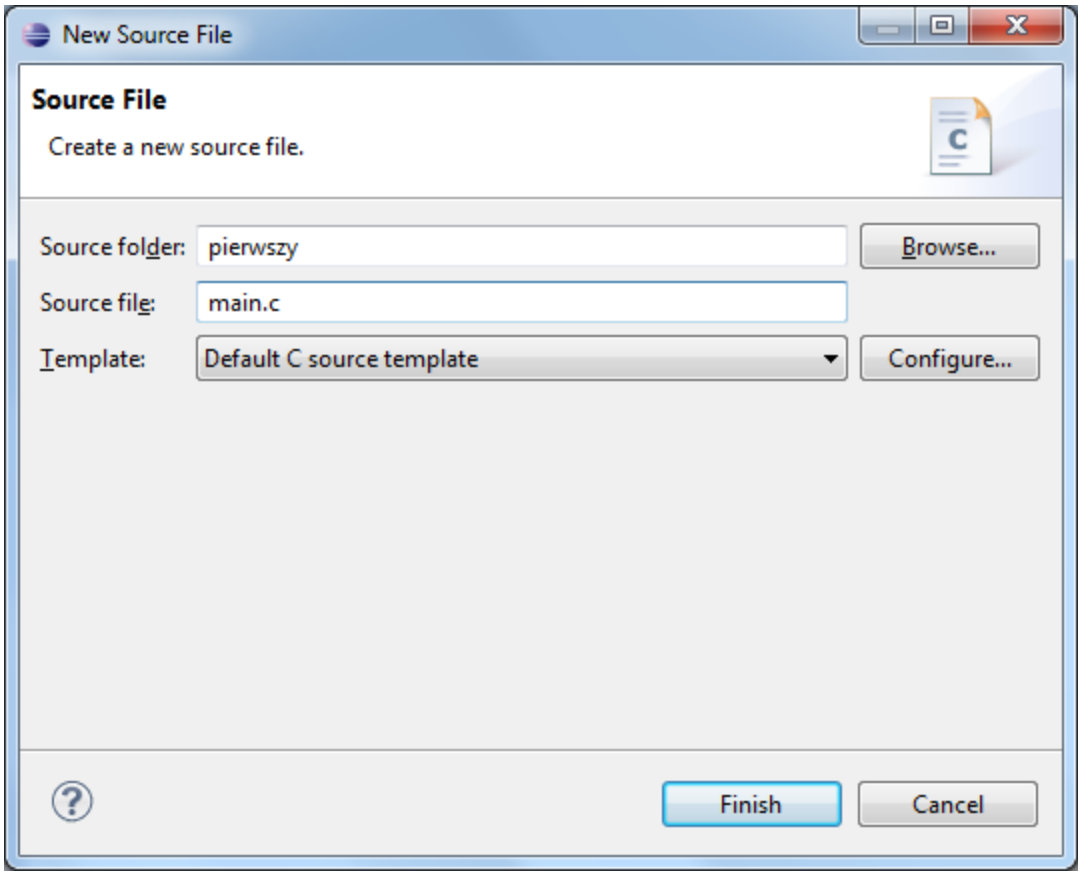
Rysunek 11

W oknie **Project Explorer** powinien pojawić się folder o nazwie pierwszy co świadczy o poprawnym utworzeniu projektu. Pozostaję jeszcze wybór programatora, w tym celu klikamy prawym przyciskanie na ikonę naszego projektu i wybieramy **Properties**, następnie w oknie **Properties for pierwszy**, klikamy kolejno **AVR** i **AVRDude,** w polu **Programmer Configuration** klikamy przycisk **New….** W nowym oknie wybieramy nasz programator czyli **usbasp** oraz w polu **Configuration Name** podajemy nazwę konfiguracji taką abyśmy potem mogli łatwo ją rozpoznać, sposób wyboru programatora przedstawia rysunek 12. Po wyborze programatora wciskamy **OK**i ponownie **OK.**



Rysunek 12

Przejdziemy teraz do pisania kodu naszego programu i programowania układu. Najpierw należy dodać plik **main.c** do projektu, w tym celu klikamy prawym przyciskiem mysz na ikonę naszego projektu i wybieramy **New** i **Source File**. W oknie **New Source File** podajemy nazwę pliku w sposób pokazany na rysunku 13.

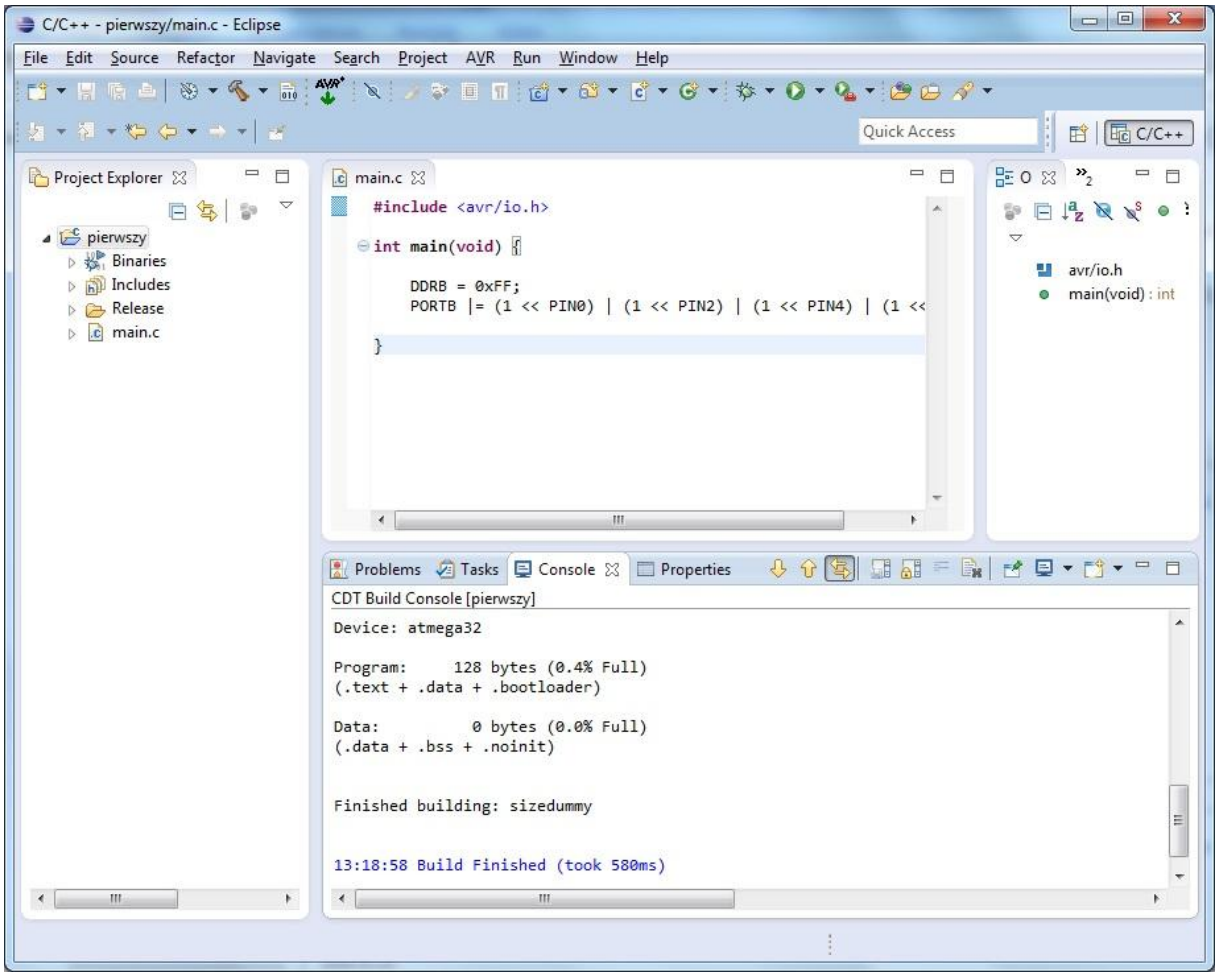


Rysunek 13

Po wciśnięciu przycisku **Finish** , plik main.c powinien automatycznie otworzyć się w środkowej części okna programu. Uzupełnijmy go prostym programem:

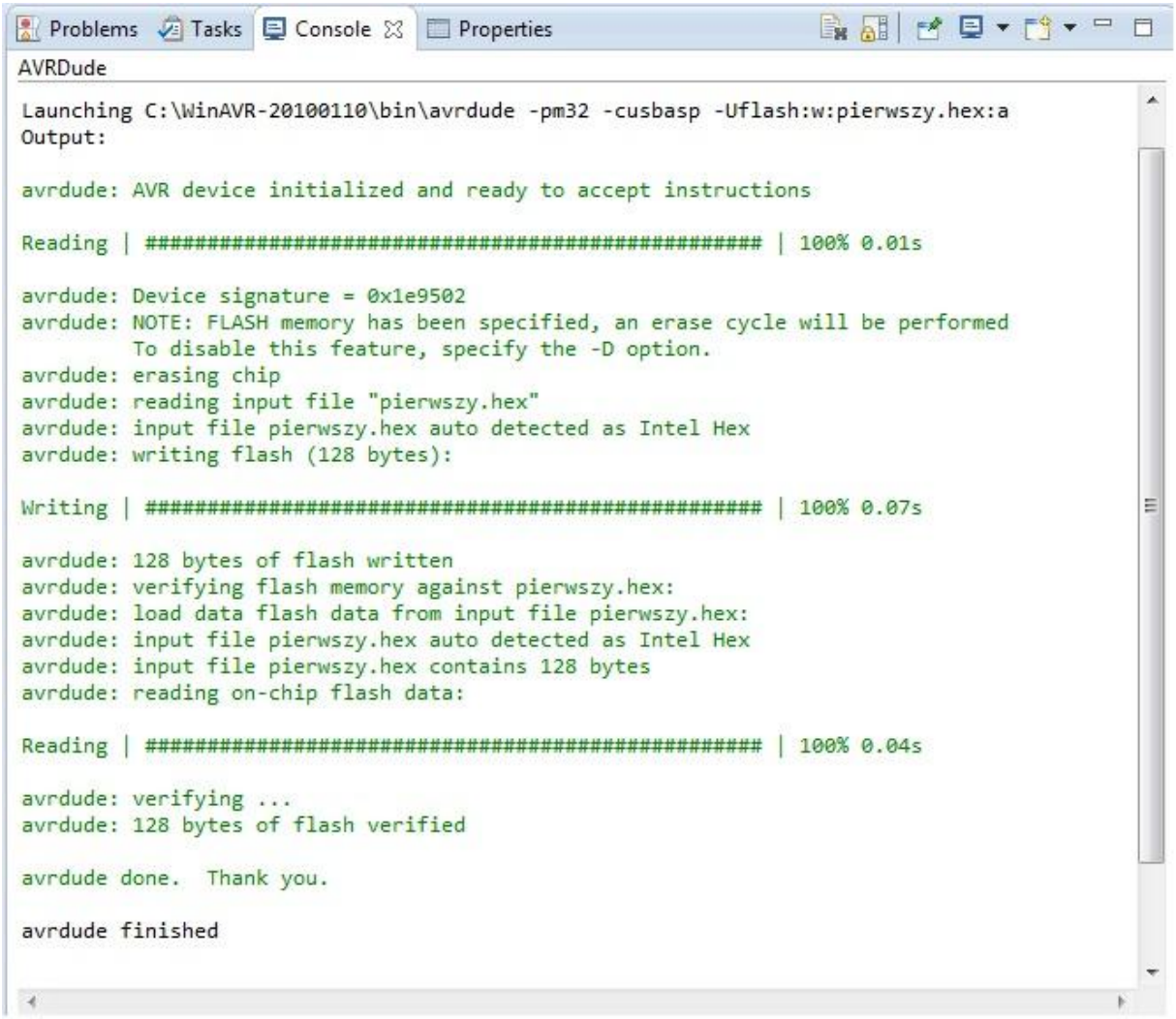
```
#include <avr/io.h>

int main(void) {

	DDRB = 0xFF;
	PORTB |= (1 << PIN0) | (1 << PIN2) | (1 << PIN4) | (1 << PIN6);

}
```

Następie przyciskiem  kompilujemy nasz program, po kompilacji okno programu powinno wyglądać jak na rysunku 14. W konsoli u dołu okna widać potwierdzenie poprawnej kompilacji, rozmiar kodu wynikowego oraz miejsce jakie zajmie on w pamięci układu.



Rysunek 14

Ostatnim etapem zabawy jest programowanie układu docelowego, w tym celu należy po prostu kliknąć ikonkę  . Jeśli wszystko poprawnie skonfigurowaliśmy, podłączyliśmy układ do komputera oraz mamy poprawnie zainstalowane sterowniki programatora powinniśmy otrzymać komunikat w konsoli przedstawiony na rysunku 15 a nasze diody powinny świecić.


Problems
Tasks
Console
Properties
AVRDude
Launching C:\WinAVR-20100110\bin\avrdude -pm32 -cusbasp -Uflash:w:pierwszy.hex:a
Output:
avrdude: AVR device initialized and ready to accept instructions
Reading | ## | 100% 0.01s
avrdude: Device signature = 0x1e9502
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "pierwszy.hex"
avrdude: input file pierwszy.hex auto detected as Intel Hex
avrdude: writing flash (128 bytes):
Writing | ## | 100% 0.07s
avrdude: 128 bytes of flash written
avrdude: verifying flash memory against pierwszy.hex:
avrdude: load data flash data from input file pierwszy.hex:
avrdude: input file pierwszy.hex auto detected as Intel Hex
avrdude: input file pierwszy.hex contains 128 bytes
avrdude: reading on-chip flash data:
Reading | ## | 100% 0.04s
avrdude: verifying ...
avrdude: 128 bytes of flash verified
avrdude done. Thank you.
avrdude finished

Rysunek 15