WinAVR, Eclipse, AVR8 Burn-O-Mat – niezbędne narzędzia programistyczne dla mikrokontolerów z rodziny AVR.

Wojciech Tarnawski

8 grudnia 2013

Spis treści

1 Wstęp

Eclipse jest kompletnym środowiskiem programistycznym wspierająca wiele języków programowania (C,C++, JAVA, PHP itd.). Środowisko jest dostępne nieodpłatnie i często na jego podstawie firmy tworzą własne oprogramowanie, przykładem takiego komercyjnego produktu jest środowisko Atolic dla mikrokontrolerów z rodziny ARM–Cortex. Zaleta środowiska programistycznego jest integracja wielu narzędzi, które są dostępne w jednym miejscu.
Zalety programu Eclipse z dodatkiem dla AVR:

- Automatycznie tworzenie Makefile - bardzo wygodne przy większej ilości plików w projekcie.
- Kreator tworzenia projektu AVR - wybieranie podstawowych parametrów z pozycji dostępnych na listach rozwijanych.
- Narzędzie IntelliSense - podpowiedzi funkcji, zmiennych w czasie pisania kodu wraz z pomocą do nich.
- Sprawdzanie poprawności kodu w trybie onnline - szybkie wykrywanie literówek, braku średników itd.
- Kompilacja i wgrywanie programu do mikrokontrolera za pomocą jednego kliknięcia.
- Duża popularność oprogramowania - duża grupa użytkowników służących pomocą.

Wymienione powyżej funkcje są obecnie standardem w programowaniu z wykorzystaniem zaawansowanych środowisk programistycznych ponieważ bardzo przyśpieszają proces pisania bezbłędnego kodu programu. Pozwalają skupić się głownie na tworzeniu kodu aplikacji od strony funkcjonalności.

1.1 Instalacja

Pierwszym krokiem do rozpoczęcia swojej przygody z mikrokontrolerami AVR jest instalacja niezbędnych narzędzi. Należy rozpocząć od instalacji kompilatora, bibliotek, plików nagłówkowych dla poszczególnych układów.

1.2 Instalacja WinAvr

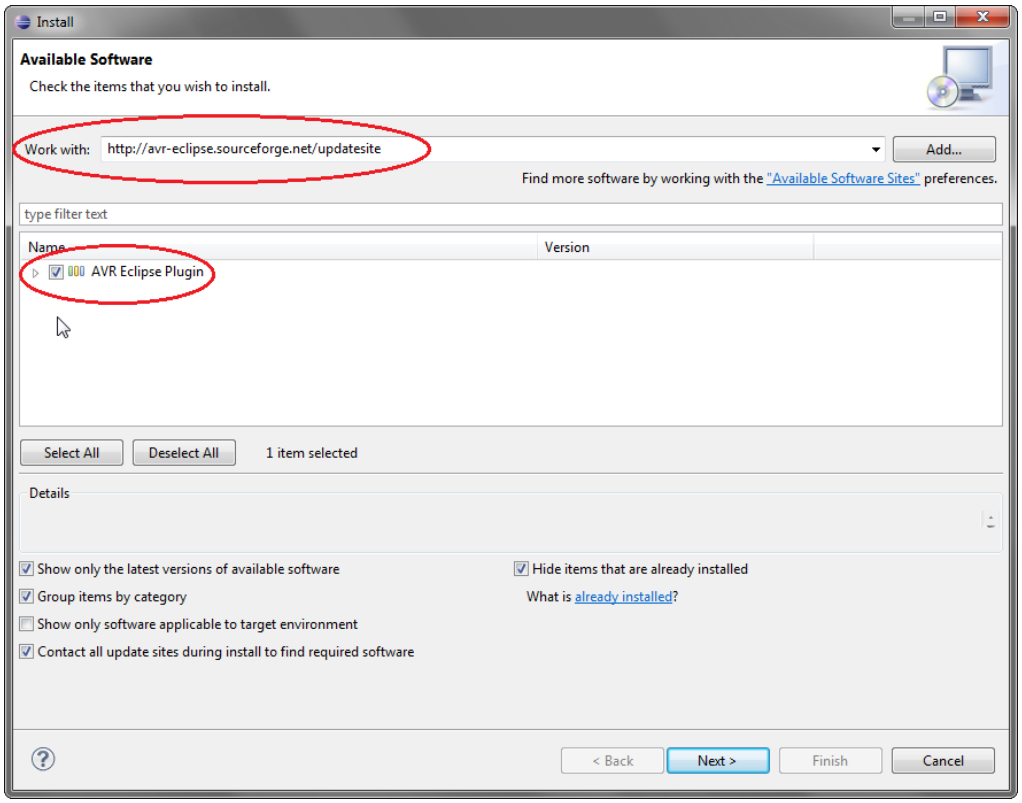
Ściągamy oprogramowanie ze strony: `http://sourceforge.net/projects/winavr/files/WinAVR/` i postępujemy zgodnie z kolejnymi krokami instalatora. Wszystkie ścieżki najlepiej zostawić bez zmian, pozwoli to uniknąć problemów konfiguracji kolejnych który aplikacji.

1.3 Instalacja Eclipse

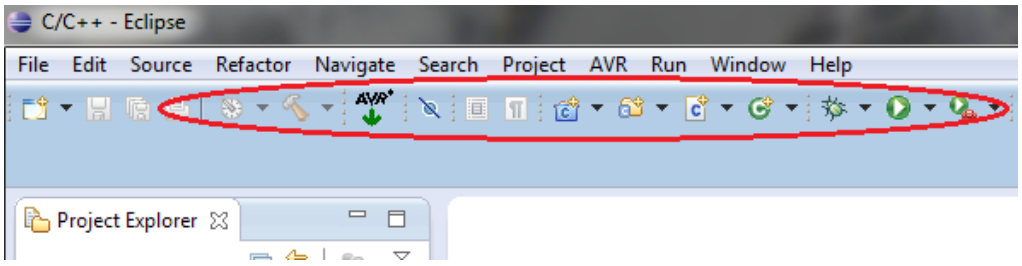
Oprogramowanie pobieramy ze strony: `http://www.eclipse.org/downloads/`, wybieramy wersję „Eclipse IDE for C/C++ Developers Eclipse IDE for C/C++ Developers". Instalujemy oprogramowanie zgodnie z kolejnymi krokami instalatora.

1.4 Instalacja dodatku AVR dla Eclipse

Sam program Eclipse nie jest przygotowany do programowania mikrokontrolerów AVR dlatego niezbędne jest doinstalowanie dodatku umożliwiającego programowanie (wymagane jest połączenie z internetem). Uruchamiamy program Eclipse i wybieramy (Help > Install New Software...) (Rys. 1) w polu „Work with" wpisujemy adres:
`http://avr-eclipse.sourceforge.net/updatesite`,
przyciskamy enter i po załadowaniu zaznaczamy pole opcji „AVR Eclipse Plugin".
Postępujemy zgodnie z kolejnymi krokami instalatora, w programie powinien pojawić się nowy pasek narzędzi (Rys. 2). Od tej pory mamy przygotowane całe środowisko programistyczne i możemy utworzyć swój pierwszy projekt.



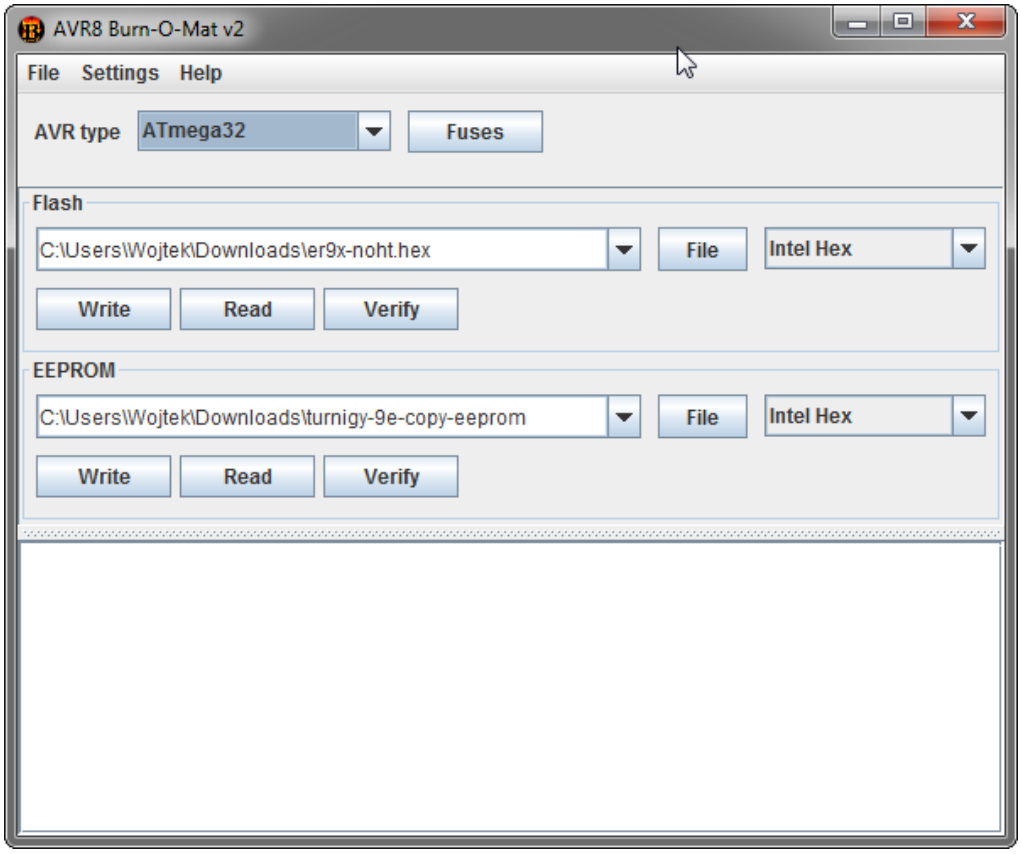
Rysunek 1: Instalacja dodatku AVR.



Rysunek 2: Pasek narzędzi do programowania AVR.

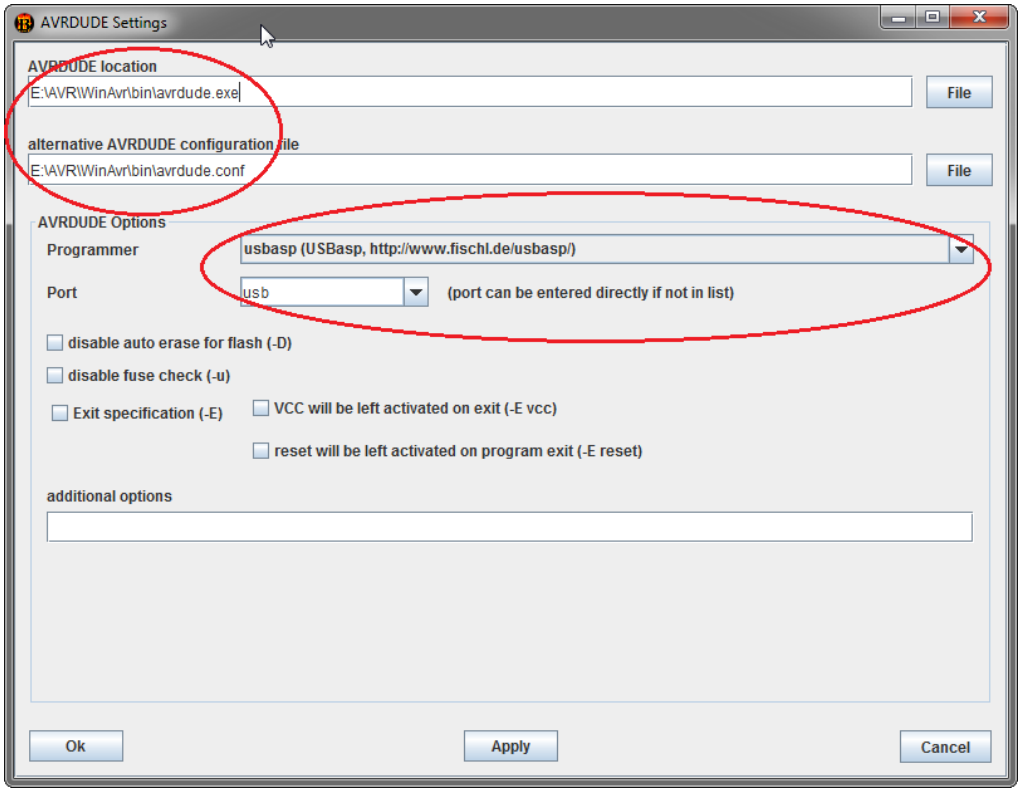
1.5 Instalacja AVR8 Burn-O-Mat

Program służy do łatwej i w miarę bezpiecznej zmiany ustawień dotyczących fusebit-ów. Dodatkowo posiada opcje wgrywania/odczytywania do mikrokontrolera pliku w formacie „.HEX". Jest to graficzna nakładka na AVRDUDE, które jest domyślnie instalowane z pakietem WinAVR. Do działania wymagana jest zainstalowana Java.
Oprogramowanie pobieramy ze strony:
`http://avr8-burn-o-mat.aaabbb.de/avr8_burn_o_mat_avrdude_gui_en.html`
Po pierwszym uruchomieniu (Rys.3) należy skonfigurować program.



Rysunek 3: Główne okno programu AVR8 Burn-O-Mat.

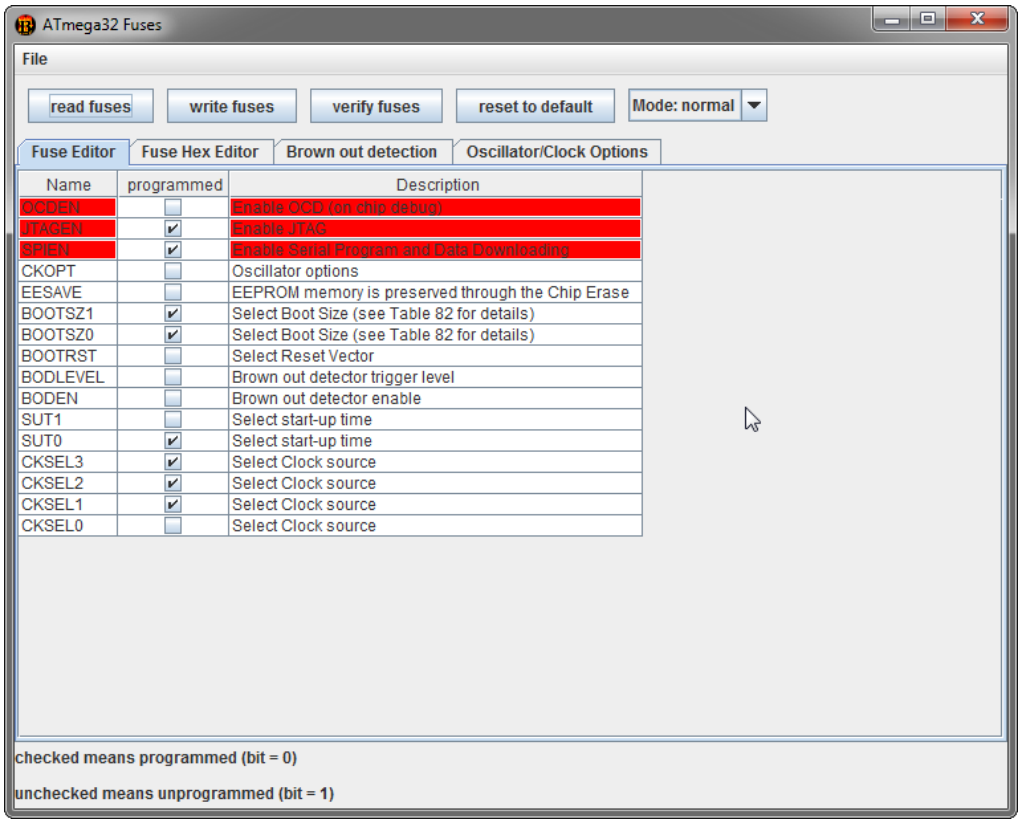
Z menu opcji wybieramy „Settings". W oknie (Rys.5).



Rysunek 4: Konfigurowanie programu AVR8 Burn-O-Mat.

W nowym oknie podajemy w polu „AVRDUDE location" i „AVR configuration file" prawidłową ścieżkę do „avrdude.exe" i „avrdude.conf", które znajdują się w katalogu bin w miejscu zainstalowanego WinAVR. Po prawidłowych ustawieniach resetujemy program i ponownie wchodzimy do „settings". Teraz powinniśmy móc wybrać programator w opcji „Programmer" - jeśli lista nie jest widoczna to oznacza, że mamy źle ustawioną ścieżkę do pliku „avrdude.conf". Wybieramy posiadamy programator i na jakim porcie pracuje, zamykamy okno. Od teraz program AVR8 Burn-O-Mat jest gotowy do pracy.

Konfiguracja fusebit-ów (Rys.) odbywa się w edytorze w którym za-



Rysunek 5: Konfigurowanie fusebit.

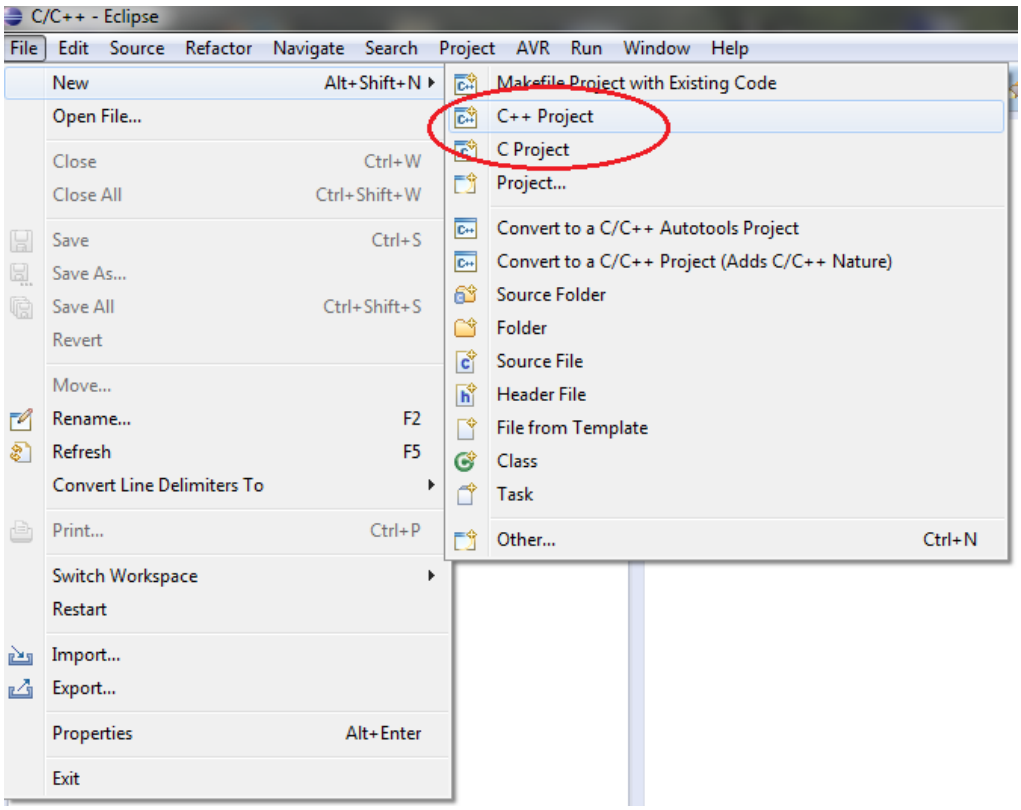
znaczamy odpowiednie ustawienia, dodatkowo w zakładkach można wybrać gotowe ustawienia (np. zmiana sygnału zegarowego z wewnętrznego an zewnętrzny odbywa się w zakładce „Oscillator/Clock option").

Uwaga!!!
Ręczne niewłaściwe ustawienie fusebit-ów może prowadzić do zablokowania mikrokontrolera.
Należy konfigurować fusebit-y bardzo ostrożnie.

2 Pierwsze kroki

2.1 Tworzenie pierwszego projektu - konfiguracja

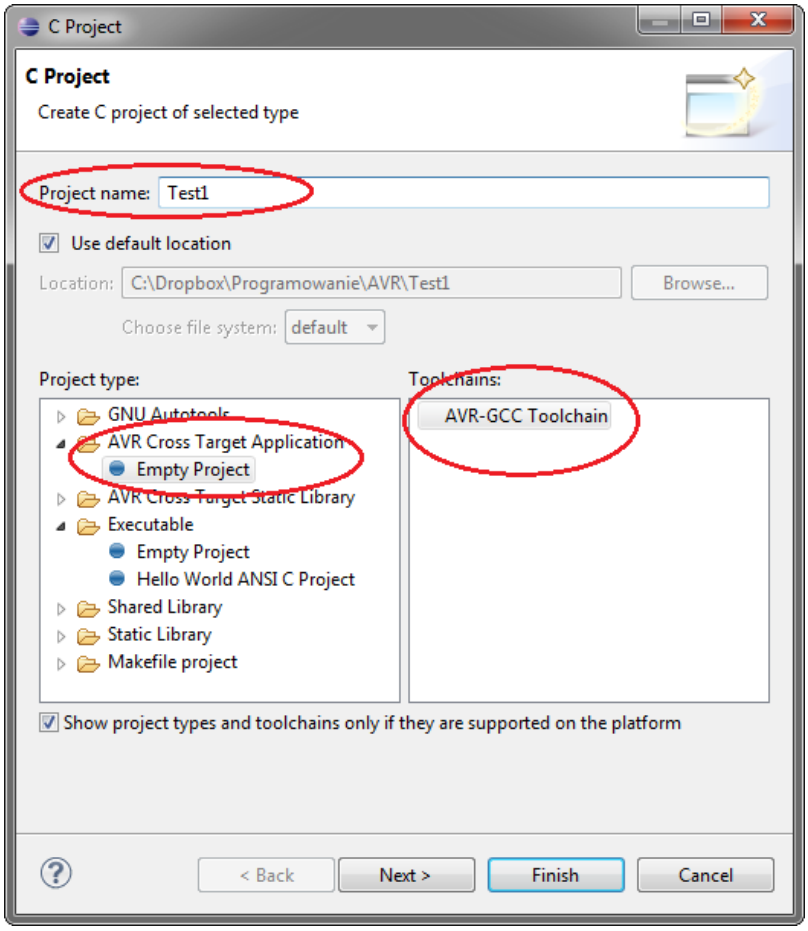
Po uruchomieniu programu Eclipse wybieramy (File > New > C Project lub C++ Project) (Rys.6).



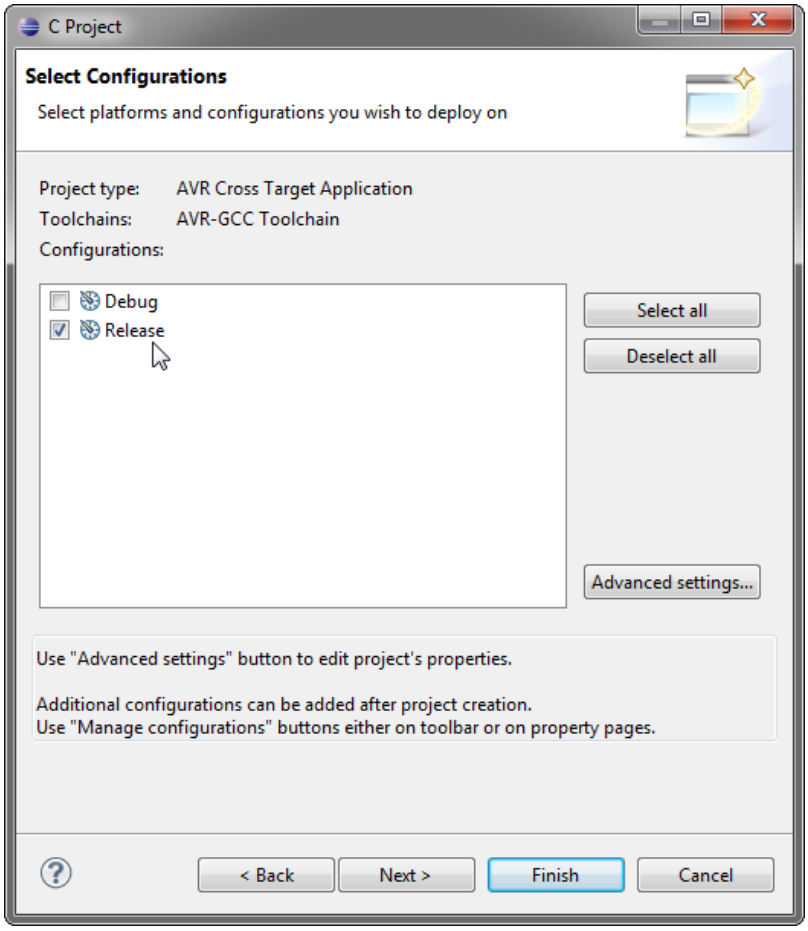
Rysunek 6: Eclipse - nowy projekt.

W polu „Project name" podajemy nazwę dla naszego nowego projektu i katalog zapisu. Następnie wybieramy (AVR Cross Target Application > Empty Project), upewniamy się, że w prawym oknie mamy wybrane „AVR-GCC Toolchain" (Rys.7) i klikamy „Next >".
W kolejnym oknie zaznaczam tylko opcje „Relase" (Rys. 8) i przechodzimy dalej.

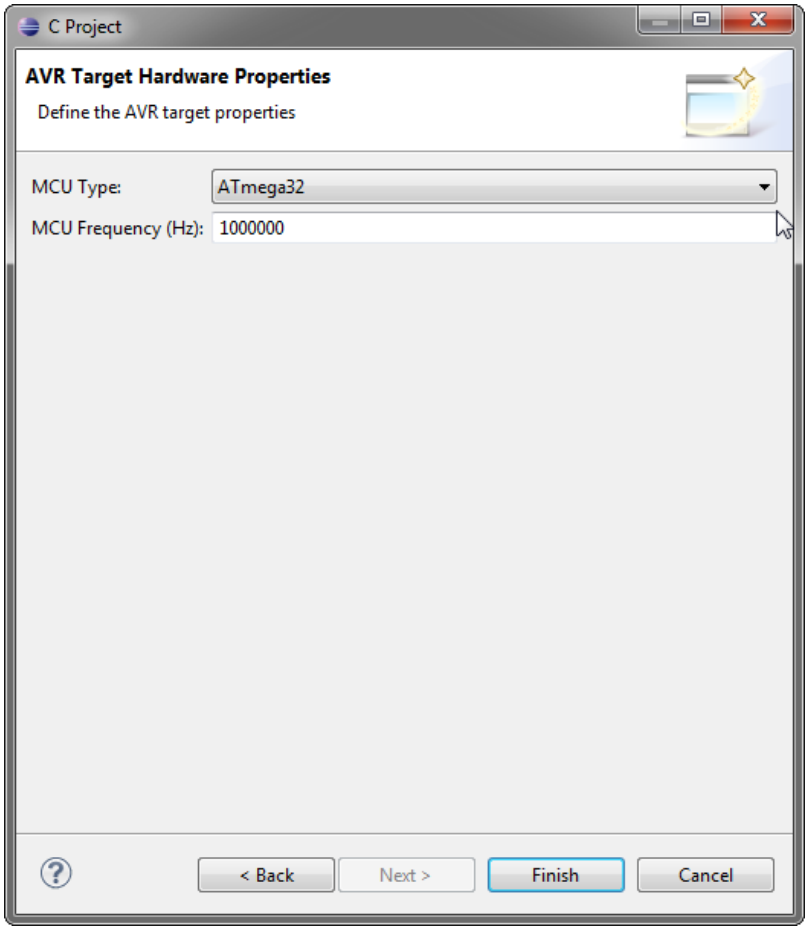
Teraz wybieramy z listy na jakim układzie będziemy pracować i podajemy jego częstotliwość taktowania (Rys.9). Klikamy „Finish".



Rysunek 7: Eclipse - wybór nazwy i rodzaju projektu.

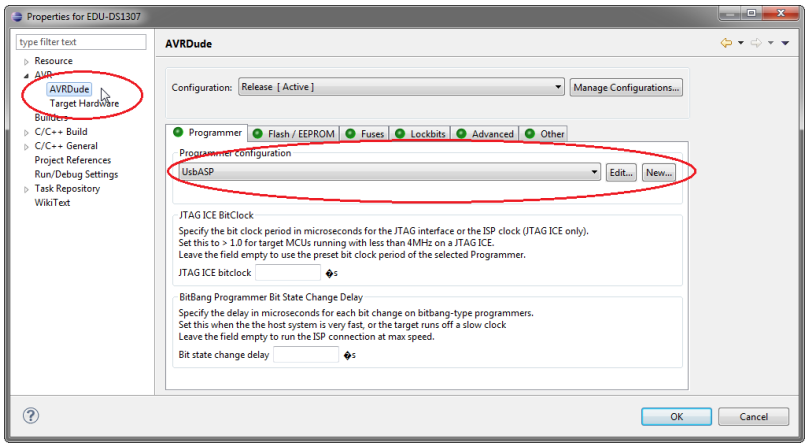


Rysunek 8: Eclipse - wybór trybu projektu.



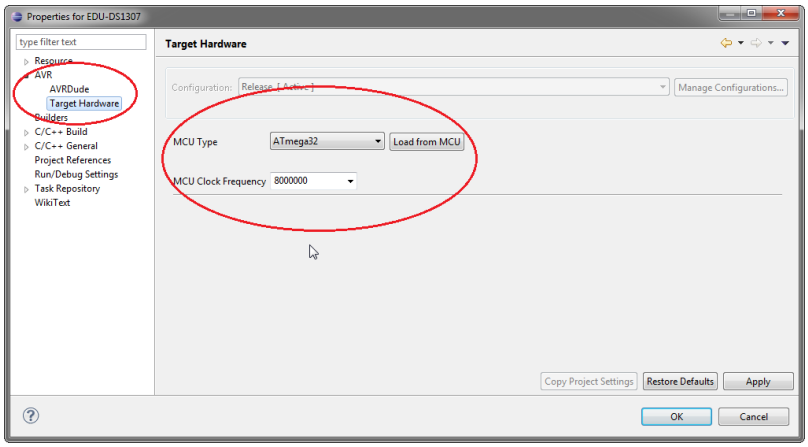
Rysunek 9: Eclipse - wybór mikrokontrolera i częstotliwości taktowania.

W lewym oknie utworzył się nowy projekt. Teraz należy ustawić jeszcze programator, w tym celu klikamy prawym przyciskiem myszy nazwę nowego projektu i wybieramy pozycję „Properties". Nastepnie wybieramy (AVR > AVRDude). W nowym oknie (Rys.10) z listy rozwijanej wybieramy używany programator. Jeśli nie mamy na liście naszego programatora to klikamy przycisk „Edit" i tworzymy nową konfiguracje.



Rysunek 10: Eclipse - właściwości, ustawienie programatora.

W zakładce „Target Hardware" (Rys.11) mamy ponownie możliwość wyboru/zmiany mikroprocesora jak i wyboru częstotliwości.



Rysunek 11: Eclipse - właściwości, wybór mikrokontrolera i częstotliwości taktowania.

2.2 Tworzenie programu – pisanie kodu

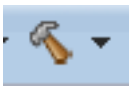
Mamy już gotowy projekt do pracy. Teraz musimy utworzyć pierwsze pliki w których będzie nasz kod programu. Pisząc program należy starać się dzielić kod na poszczególne moduły umieszczone w osobnych plikach (na wzór programowania obiektowego – klasy). W lewym oknie klikamy na nazwę naszego projektu i wybieramy „New" a następnie typ pliku jaki chcemy utworzyć. Najlepiej zawsze tworzyć pary plików „Header File" i „Source File". W pliku nagłówkowym (Header) dokonujemy definicji nagłówków funkcji zmiennych globalnych, importujemy dodatkowe biblioteki. Natomiast w pliki „source" implementujemy kod funkcji, kod programu. Pierwszymi plikami jakie należy utworzyć są „main.h" i „main.c". W pliku „main.c" tworzymy funkcję startową "int main() ...".

Pisanie kodu w środowisku Eclipse jest bardzo przyjemne. Środowisko na bieżąco sprawdza poprawność naszego kodu i w razie literówek podkreśla na czerwono błędy. Dodatkowa zaleta jest funkcja podpowiadania. Wystarczy wpisać kilka początkowych liter zmiennej, funkcji a rozwinie się lista z podpowiedziami.

Porada
Jeśli lista nie rozwija się automatycznie to można wymuśić jej rozwinięcie przez przyciśnięcie kombinacji klawiaszy „CTRL + SPACE".

2.3 Kompilacja i wgrywanie do mikrokontrolera

Jak mamy już gotowy program to należy go skompilować ikoną przedstawiającą młotek (Rys.12).



Rysunek 12: Eclipse - ikona kompilacji projektu.

Jeśli projekt nie zawiera błędów to w dolnym oknie - konsola pojawi się komunikat o zakończonej kompilacji z sukcesem (Rys.13). Jeśli projekt na błędy należy je poprawić, przed próbą wgrania projektu do układu.
Do wgrywania projektu służy ikonka z zieloną strzałką i napisem „AVR*" (Rys.14). Postęp wgrywania pokazywany jest w konsoli i jeśli nie wystąpiły błędy to po chwili mamy wgrany nowy program do naszego układu.

```
Invoking: Print Size
avr-size --format=avr --mcu=atmega32 EDU-DS1307.elf
AVR Memory Usage
----------------
Device: atmega32

Program:    2972 bytes (9.1% Full)
(.text + .data + .bootloader)

Data:         40 bytes (2.0% Full)
(.data + .bss + .noinit)

Finished building: sizedummy


15:01:53 Build Finished (took 2s.328ms)
```

Rysunek 13: Eclipse - kompilacja zakończona sukcesem.

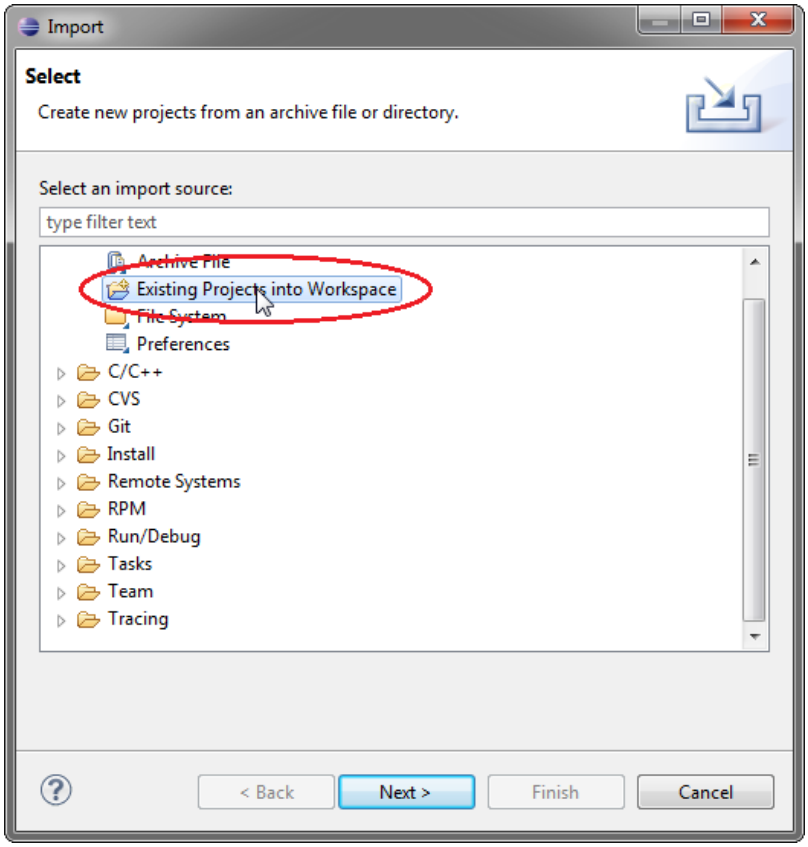
Rysunek 14: Eclipse - Ikona wgrywania projektu do mikrokontrolera.

Uwaga!!!
Jeśli pojawiają się błędy przy wgrywaniu programu należy sprawdzić poprawność podłączenia programatora, zasilania. Jeśli wszystkie połączenia są prawidłowe to należy odłączyć programator od portu USB i ponownie podłączyć.

2.4 Import projektu

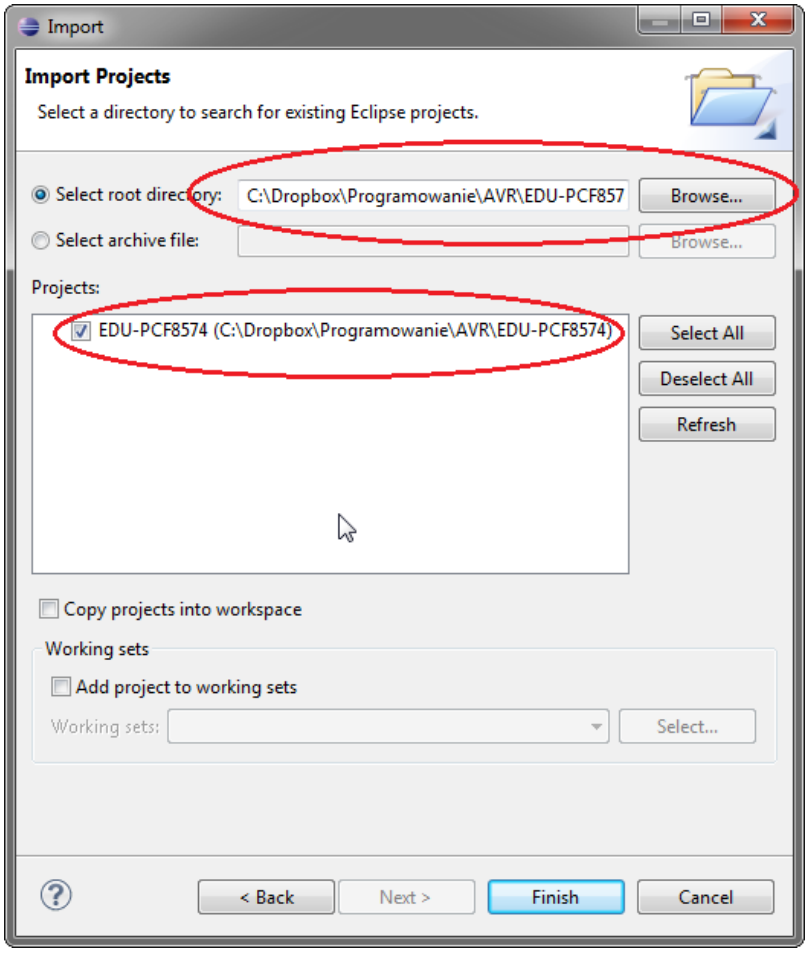
Ważną kwestią jest otwieranie projektu, który został wcześniej zrobiony. W Eclipsie zaleca się dokonywać tej operacji przez importowanie projektu, dzięki czemu mamy gwarancję, że program załaduje wszystkie ustawienia projektu a nie tylko same pliki. Importowanie projektu polega na wskazaniu katalogu na dysku zawierającego nasz projekt.

Wybieramy „File->Import" w otwartym oknie (Rys.15) wybieramy opcje „Existing project in Workspace" i klikany „Next >".



Rysunek 15: Eclipse - Wybieranie typu importu.

Teraz wybieramy „Browse" i odszukujemy katalog z naszym projektem (Rys.16). Po wybraniu poprawnej lokalizacji w oknie poniżej załaduje się nam wybrany projekt. Klikamy „Finish" i możemy przystąpić do pisania programu.

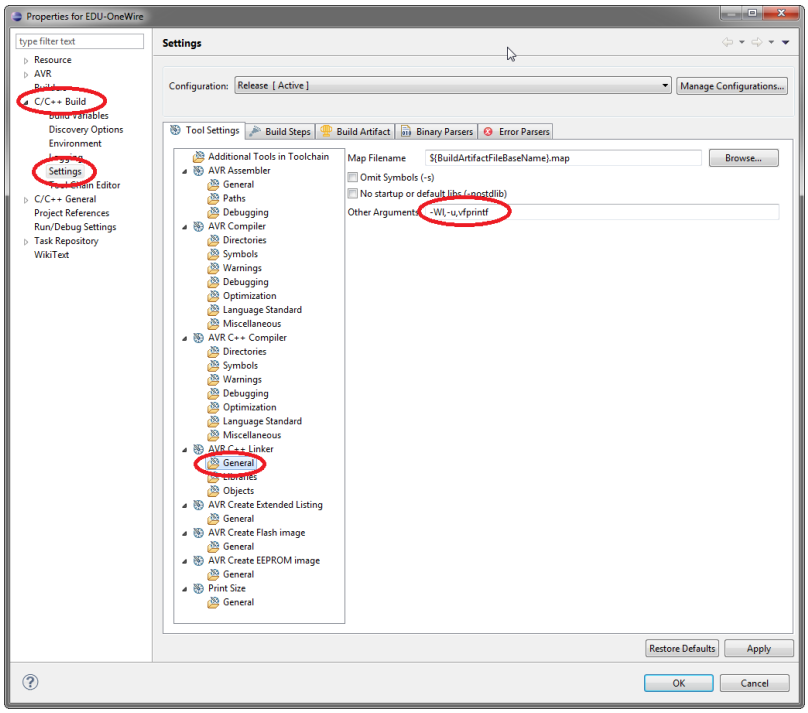


Rysunek 16: Eclipse - Wybieranie lokalizacji projektu do importu.

3 Dodatki

3.1 Wykorzystanie typu float w funkcji „sprintf”

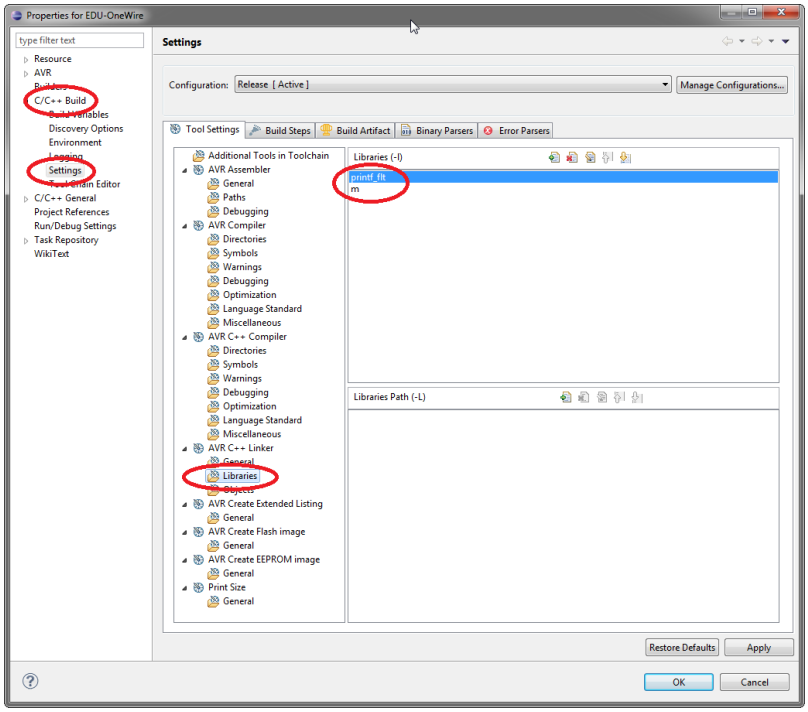
Gdy chcemy wykorzystać zmiennie typu float, double w funkcji „sprintf” należy poinformować kompilator o takiej potrzebie. Jeśli kompilator nie dołączy dodatkowych bibliotek to zamiast liczby pojawi się znak „?”. Do poprawnego wyświetlania zmiennych zawierających liczby z przecinkiem kompilator musi dodać dodatkowe biblioteki do naszego projektu - zwiększa to kod wynikowy aplikacji.



Rysunek 17: Eclipse - Konfiguracja funkcji „sprintf” do obsługi liczb typu float.

Wchodzimy do właściwości projektu i wybieramy „C/C++ Builder” -> „Settings” -> „AVR C++ Linker” -> „General” i w polu „Other Arguments” wpisujemy: „-Wl,-u,vfprintf” (Rys. 17). Należy dodać jeszcze dwie biblioteki: „libm.a” i „libprintf_flt.a”. Wybieramy „C/C++ Builder” -> „Settings” -> „AVR C++ Linker” -> „Libraries” i dodajemy nowe biblioteki przy użyciu przycisku z plusem. W nowym oknie wpisujemy „m”, powtarzamy czynność

i tym razem wpisujemy „printf_flt” (rys. 18). Eclipse został poinformowany, że ma dodawać dodać do projektu podane wcześniej biblioteki. Po tym zabiegu należy dodatkowo wybrać z listy projektu opcję „Clean Project” a następnie „Build Project”.



Rysunek 18: Eclipse - Konfiguracja funkcji „sprintf” do obsługi liczb typu float.