Instalacja i konfiguracja toolchaina:

Jeżeli chcecie użyć innego środowiska niż WinAVR – wasza sprawa. Naszym zdaniem najłatwiej jest skonfigurować na nasze potrzeby właśnie to środowisko.

1. Pobierz i zainstaluj pakiet WinAVR ze strony: http://sourceforge.net/projects/winavr/files/
Uwaga! Wskazane jest, żeby program WinAVR zainstalować w folderze, którego ścieżka nie zawiera spacji. W przeciwnym wypadku podczas kompilacji będziecie otrzymywali błąd:

make: Interrupt/Exception caught (code = 0xc00000fd, addr = 0x4217b3)

http://www.avrfreaks.net/forum/make-interruptexception-caught-code-0xc00000fd-addr
A przynajmniej ja tak miałem I reinstalacja w folderze domyślnym proponowanym przez instalator (zamiast w ProgramFiles (x86)) pomogła.

2. Otwórz program 'MFile':
   2.1. Ustaw **Makefile > Main file name** na wartość '**main**'
   2.2. Ustaw **Makefile > MCU Type > ATmega > atmega128**
   2.3. Ustaw **Makefile > Port > lpt1**
   2.4. Ustaw **Makefile > Enable Editing of Makefile**
   2.5. W pliku zmień wartość zmiennej '**AVRDUDE_PROGRAMMER**'

```
AVRDUDE_PROGRAMMER = usbasp
```

   2.6. **File > Save As**. Zapisz plik Makefile w folderze głównym Twojego projektu.

3. Kompilacja programu:
W programie Programmers Notepad:
Program kompilujemy wybierając: **Tools > Make All**
Funkcja **Tools > Make Clean**: Kompilator jest na tyle sprytny, że wykrywa zmiany w plikach projektu, który kompiluje i będzie kompilował tylko te pliki, które się zmieniły. Tak więc, jeżeli z jakiegoś powodu chcesz skompilować cały projekt (bo np. zmieniłeś optymalizację w pliku Makefile) bądź zmieniłeś któryś z plików nagłówkowych **.h* , konieczne jest, aby użyć komendy **Tools > Make Clean**.

4. Zaprogramowanie uC (wgranie pliku **.hex* do pamięci flash)
Tu bez niespodzianki: **Tools > Program**
Poprawiony sterownik dla programatora USBasp dla Windows
http://www.ulrichradig.de/forum/viewtopic.php?p=3792#p3792

Wgranie ustawień ATmegi

Wpisz w konsoli:
avrdude -p atmega128 -c usbasp -P usb -B 8 -V -u -U lfuse:w:0xFF:m -U hfuse:w:0xD9:m -U efuse:w:0xFF:m

Aby nasza ATmega działała prawidłowo (odpowiednie taktowanie, możliwość programowania SPI, itp.), należy jej ustawić odpowiednie 'fuse bity'. Jest to linijka kodu wygenerowana przez program MkAVRCalculator. Można ją uruchomić przez terminal.

Układ plików projektu

Plik *main.c* , którym będzie znajdowała się funkcja main, od której zaczyna się wykonywanie programu.
Plik *hardware.h* , w którym będziemy mieli dla każdego pinu napisaną jego definicję portu (czyli rejestrów IO, do których jest on przypisany).
Wszystkie pozostałe pliki będą tworzyły pary plików: pliku nagłówkowego *nazwa_pliku.h* oraz pliku źródłowego *nazwa_pliku.c* . W plikach nagłówkowych będziemy mieli napisane wszelkie deklaracje zmiennych lub funkcji, a dopiero w plikach źródłowych ich definicje.

Dobrą praktyką jest, aby w plikach nagłówkowych deklarować wszystkie funkcje danej biblioteki, natomiast tylko te zmienne, których potem będziemy używali poza naszą parą plików (po co pisać, że mają być dostępne też w innych plikach, skoro i tak nie będziemy z nich tam korzystać?)

Reszta plików będzie bibliotekami. Każda biblioteka to para plików - pliku nagłówkowego oraz źródłowego. Na jakiej zasadzie tworzymy biblioteki? Tu nie ma żadnych narzuconych reguł - czy to ze strony kompilatora czy producenta mikrokontrolera. Tworzymy je według własnych zasadach, w praktyce każda biblioteka zajmuje się innym podzespołem naszego projektu - stąd biblioteki sterujące komunikacją UART, pomiarem ADC z fototranzystorów, sterowaniem silnikami, czy peryferiami (guziki , diody).

Projekt I (na 16.12.2015)

Przygotować bibliotekę *outskirts* (nazwijcie ją sobie dowolnie), ktora będzie zawierała dwie funkcje: jedną sterującą ledami, a drugą sprawdzającą, czy podany w argumencie funkcji przycisk jest wciśnięty. Oczywiście nie zapomnijcie też o funkcji *Outskirst_init()* , w której znajdą się odpowiednie ustawienia rejestrów I/O, których ta biblioteka będzie używała.
Aby sprawdzić poprawność biblioteki, napisać program, który za pomocą dwóch przycisków będzie sterował czasem włączania / wyłączania się diod. Po naciśnięciu pierwszego przycisku, diody przełączają się wolniej, po naciśnięciu drugiego przycisku - szybciej.

Dodatkowo, uruchomić moduł HC-05. Używając dostarczonej przez nas biblioteki, wyświetlić w programie BlueTerm (na platformie Android) tekst: "*Moje uszanowanko. Witam kierownika. Oto Rat ${name}*".

Disclaimer: Sekwencja migania diodami, działanie przycisków oraz tekst są przykładowe. Możesz wymyślić swoje własne, jeżeli przykłady Tobie nie odpowiadają.