Podstawy Systemów Mikroprocesorowych

Instalacja środowiska programistycznego dla mikrokontrolerów AVR v.0.3

Dariusz Tefelski

2020-10-11

Spis treści

Podstawy Systemów Mikroprocesorowych **3**
Instalacja środowiska programistycznego 3
System operacyjny Microsoft Windows 3
System operacyjny GNU/Linux 11
System operacyjny macOS 14

Podstawy Systemów Mikroprocesorowych

Instalacja środowiska programistycznego

System operacyjny Microsoft Windows

1. Instalacja toolchain-a: kompilatora, linker-a, narzędzi do budowania (avr-gcc) oraz oprogramowania do programowania mikrokontrolerów (avrdude).

Istnieje kilka sposobó na instalację oprogramowania do pracy z mikrokontrolerami AVR. Możliwe jest wykorzystanie oprogramowania udostępnianego przez producenta mikrokontrolerów (firma Microchip), oprogramowania dystrybuowanego w pakiecie WinAVR - dostępny pakiet binarny jest bardzo stary - z 2010r., zbudowanie własnego toolchain'a, wykorzystanie już zbudowanego toolchain'a z najnowszym kompilatorem avr-gcc. Wykorzystamy ostatnią opcję.

- Pobranie archiwum *avr-gcc* ze strony: https://blog.zakkemble.net/avr-gcc-builds/

Na chwilę obecną najnowszy plik archiwum to: *avr-gcc-10.1.0-x64-windows.zip* Bezpośredni link do pobrania: avr-gcc-10.1.0-x64-windows.zip

Po rozpakowaniu archiwum zajmuje ok. 466 MB, dlatego upewnij się, że masz więcej wolnej przestrzeni dyskowej.

- Rozpakowanie archiwum **avr-gcc**. Pobrane archiwum zip należy rozpakować. Można kliknąć prawym klawiszem myszy na archiwum i z podręcznego menu wybrać "Wyodrębnij wszystkie". W polu pod napisem: Pliki zostaną wyodrębnione do folderu, ustawić: **C:**. A następnie kliknąć przycisk **Wyodrębnij**.
- Zostanie automatycznie stworzony katalog: C:\avr-gcc-10.1.0-x64-windows
- Następnie wejdź do: Panel Sterowania -> System i zabezpieczenia -> System i kliknij **Zaawansowane ustawienia systemu**. A następnie w zakładce **Zaawansowane** kliknij przycisk **Zmienne środowiskowe**.
- W polu na dole **Zmienne środowiskowe** wybierz zmienną **Path** i kliknij przycisk **Edytuj**
- W nowym okienku, przejdź na początek wartości zmiennej i dopisz na początku następujący tekst, pozostawiając resztę nie zmienioną: **C:\avr-gcc-10.1.0-x64-windows\bin;** Następnie kliknij **Ok**

W Windows 10 zmienne środowiskowe ustawia się następująco: **Ustawienia**–>**System**–>**Informacje**, a następnie z prawej strony wybieramy **Informacje o systemie**. Następnie po lewej wybieramy **Zaawansowane ustawienia systemu** i w okienku **Właściwości systemu** klikamy na przycisk **Zmienne środowiskowe**, który jest po prawej na dole (Ewentualnie wyszukujemy **Panel sterowania**–>**System i zabezpieczenia**–>**System** i wybieramy po lewej **Zaawansowane ustawienia systemu** itd.). W otworzonym okienku **Zmienne środowiskowe**, w dolnym polu *Zmienne systemowe* wyszukujemy **Path** i klikamy przycisk **Edytuj** znajdujący się pod polem wyboru. W okienku **Edycja zmiennej środowiskowej** klikamy przycisk po prawej **Nowy** i wpisujemy **C:\avr-gcc-10.1.0-x64-windows\bin** i klikamy przycisk **Ok**.

- Wciśnij (WinKey + R) - Uruchom, albo w Menu wpisz w polu: **cmd**
- Wpisz avr-gcc i naciśnij **Enter**
- Powininno pojawić się: avr-gcc: fatal-error: no input files compilation terminated
- Świadczy to o prawidłowym ustawieniu ścieżki dostępu - po wpisaniu gdziekolwiek komendy avr-gcc, uruchamia się właściwy program ze ścieżki

3. Instalacja edytora **Geany**

- Ze strony: https://www.geany.org/download/releases/ należy pobrać plik **geany-1.36_setup.exe**

Na chwilę obecną, najnowsza wersja geany to 1.36. Bezpośredni link do pobrania: Geany 1.36

- Uruchomić program instalacyjny geany i zainstalować w standardowy sposób według domyślnych ustawień.
- Na pulpicie powinna pojawić się charakterystyczna ikonka: „żółtej lampy' '. Po uruchomieniu edytora, warto zapisać plik z rozszerzeniem **.c** - nastąpi przełączenie do kontekstu plików źródłowych języka C. Można wtedy skonfigurować sobie skróty w menu: **Zbuduj**, wybierając ostatnią pozycję: **Zdefiniuj polecenia budowania**. Proponuję kliknąć na pierwszy od góry, po lewej przycisk i wpisać: Make, a obok, gdzie jest polecenie wpisać **make**. W ten sposób wciskając F8, będziemy wykonywać budowanie projektu. Drugi przycisk proponuję nazwać: **Program**, a polecenie wpisać: **make program**, w ten sposób wciskając F9, zaprogramujemy przygotowany program w mikrokontrolerze. Trzeci przycisk od góry proponuję nazwać: **Clean** i wpisać polecenie: **make clean**. Nie ma on skrótu klawiszowego, ale zwykle nie trzeba z niego często korzystać, więc można go wywoływać z menu.

4. Instalacja sterownika programatora

- Do programowania mikrokontrolerów AVR wykorzystamy programator *USBasp*, ze strony: https://www.fischl.de/usbasp/ Programator będzie obsługiwany przez oprogramowanie *avrdude*, które zainstalowało się razem z narzędziami z punktu 1. Nie potrzeba zatem pobierać nic ze strony twórcy programatora.
- Do obsługi programatora w systemach MS Windows konieczna jest instalacja sterownika (skojarzenie VID/PID z biblioteką *libusb*. W tym celu najwygodniej skorzystać z oprogramowania *Zadig*, które należy pobrać ze strony: https://zadig.akeo.ie/

Na chwilę obecną, najnowsza wersja to 2.5. Bezpośredni link do pobrania: https://github.com/pbatard/libwdi/releases/download/b730/zadig-2.5.exe

- Podłączyć programator do komputera poprzez złącze USB.
- Po uruchomieniu programu zadig, pojawi się okienko, w którym należy wybrać sterownik. W zależności od systemu niektóre sterowniki działają, a z niektórymi są problemy. Za pomocą programu **zadig** można w łatwy sposób zmieniać te sterowniki.
- Upewnij się, że w polu wyboru pośrodku widnieje: **USBasp**, Vendor ID ma wartość 16c0, a Product ID ma wartość 05dc (wartości obok etykiety USB ID). Jeśli nie to spróbuj kliknąć na to pole i wybrać **USBasp**. Jeśli mimo to nie widać programatora, to odpiąć programator od gniazda USB w komputerze i jeszcze raz podłączyć.
- Przy strzałeczce, nad dużym przyciskiem, Wybierz sterownik **libusbK** i naciśnij duży przycisk pod spodem **Install Driver)**. Jeśli będzie zmiana sterownika to etykieta przycisku zmieni się na **Replace Driver**. Operacja może chwilkę zająć - po chwili powinno wyskoczyć okienko z informacją o prawidłowo zainstalowanym sterowniku.

5. Test działania programatora

- Proponuję utworzyć katalog z Projektami, i każdy kolejny tworzony projekt nazywac wg numeru zajec, np z0_test, z1_gpio, itd. Ponadto różne warianty tworzonych programów proponuję także umieszczać w osobnych katalogach i dodawać w nazwie kolejny numer.
- W katalogu z0_test umieścić plik **Makefile** przeznaczony dla systemu Windows, do pobrania ze strony laboratorium: AVR Makefile dla Windows

Uwaga! Proszę zwrócić uwagę, czy plik **Makefile** nie zapisał się np. z rozszerzeniem .txt. Ważne jest aby nie było dodanego rozszerzenia. Plik musi nazywać się po prostu **Makefile** i musi znajdować się w tym samym katalogu, co plik **main.c**.

- Stworzyć w katalogu z0_test plik **main.c**, w którym należy umieścić kod testowy, mrugający diodą LED podłączoną do pinu PC0:

Uwaga! Poniższy kod źródłowy najlepiej jest ręcznie wpisać w edytorze, gdyż po skopiowaniu zamiast znaków z puli kodów **ASCII**, pojawią się znaki rozszerzone z puli **UTF-8**, które nie będą właściwie interpretowane przez kompilator, np. znak ^. W wyniku czego podczas kompilacji wyświetli się komunikat o błędzie, który trzeba będzie poprawić.

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

int main(void){

    DDRC=0xff;
    PORTC=0xff;

    while(1){
        PORTC^=(1<<PC0);
        _delay_ms(500);
    }

    return 0;
}
```

- Wykonaj polecenie **make** (klawisz F8 lub właściwa opcja z menu **Zbuduj**). Prawidłowe budowanie nie powinno zgłosić błędów. Informacje na końcu powinny zawierać wielkość utworzonego pliku elf.
- Podłącz przewodem Ż-Ż (żeńsko-żeńskim), pin PC0 z pierwszym pinem ze złącza *LEDS*
- Podłącz programator USBasp do płytki EvB 5.1 do złącza ISP kablem - tasiemką (złącza typu KANDA), zwróć uwagę na klucz w złączu uniemożliwiający odwrotne podłączenie. Na tasiemce czerwonym kolorem zaznaczona jest pierwsza linia.
- Podłącz programator USBasp do komputera PC przez złącze USB. W efekcie na płytkę EvB 5.1 powinno zostać podane zasilanie (+5 V), które spowoduje zaświecenie się ekranu LCD i ewentualnie diody LED. Może także działać demonstracyjny program, który będzie mrugał diodami LED.
- Wykonaj polecenie **make program** (klawisz F9). Wynikiem powinno być bezbłędne zaprogramowanie mikrokontrolera, jego reset i uruchomienie naszego testowego programu. Powinniśmy widzieć zapalanie się diody LED z okresem 1 sekundy.

6. Instalacja środowiska Eclipse

- Pobrać wersę Eclipse z zainstalowanym CDT (przeznaczoną do tworzenia aplikacji C/C++)

Różne pakiety Eclipse dostępne są do pobrania na stronie: https://www.eclipse.org/downloads/packages/
Obecnie najnowsza wersja to 2020.09, którą można pobrać bezpośrednio z: Eclipse CDT 2020.09

- Rozpakować archiwum zip Eclipse. Kliknąć prawym klawiszem na pliku zip, a następnie wybrać **Wyodrębnij wszystkie**. Ponieważ archiwum zawiera podkatalog eclipse, wystarczy w polu miejsca docelowego wybrać dysk, np.: **C:**
- Dla wygody można stworzyć odnośnik na pulpicie, poprzez kliknięcie prawym klawiszem myszy i wybranie **Nowy** -> **Skrót** a następnie wskazanie pliku **C:\eclipse\eclipse.exe**

W Windows 10 nie ma dostępnego środowiska JDK Java, które jest niezbędne do uruchomienia Eclipse. Należy zatem pobrać środowisko ze strony https://jdk.java.net. Najnowsze dostępne środowisko ma numer 15. Link do pobrania bezpośrednio: JDK 15. Należy kliknąć prawym klawiszem myszy na pobrane arichiwum i z menu wybrać **Wyodrębnij wszystkie**. Następnie w polu docelowym wpisać **C:**. Archiwum zostanie rozpakowane do **C:\jdk-15**. Ponownie należy dołączyć ścieżkę do zmiennych środowiskowych (patrz opis do instalacji avr-gcc). Nowa ścieżka, którą trzeba dodać to: **C:\jdk-15\bin**.

- Uruchomić program Eclipse, a następnie w menu **Help** wybrać **Eclipse Marketplace**
- W polu Find wpisać **avr** i wyszukać
- Zainstalować dodatek: **AVR Eclipse Plugin 2.3.4** - w tym celu należy zgodzić się na warunki licencyjne i nacisnąć przycisk **Finish**. Potem zezwolić na instalację oprogramowania „niepodpisanego’ ’ **Install anyway**. Eclipse po instalacji zaproponuje swoje ponowne uruchomienie: **Restart Now**, na co należy się zgodzić.

7. Przetestowanie środowiska Eclipse CDT wraz z zainstalowanym wcześniej oprogramowaniem avr-gcc i avrdude

AVR Eclipse Plugin ma drobne wady, które należy poznać, tworząc projekt nowego programu na mikrokontroler. Jedną z nich jest brak domyślnie wybranej opcji optymalizacji kompilacji dotyczącej minimalizacji wielkości utworzonego programu. Jest to opcja, która na małe mikrokontrolery jest stosowana domyślnie.

Zacznijmy od dostosowania Eclipse.

- Usunięcie Launch Bar. W przypadku programowania mikrokontrolerów AVR, jest on nieprzydatny i tylko zajmuje miejsce. Wybierz z menu u góry, **Window**–>**Preferences**–>**Run/Debug**–>**Launching**–>**Launch Bar** i odznacz checkbox **Enable Launch Bar**, a następnie kliknij przycisk **Apply and Close**.

- Ustawienie autora programu i czasu zapisu. Wybierz z menu u góry, **Window**–>**Preferences**–>**C/C++**–>**Code Style**–>**Code Templates**–> i w okienku obok wybrać **Comments** –> **Files**, a następnie kliknąć przycisk **Edit**. Zapisz wzorzec wg następującego schematu, oczywiście wpisując swoje dane (imię, nazwisko, e-mail) w pole Author:.

```
/* Project: ${project_name}
 * File: ${file_name}
 *
 * Created on: ${date} ${time}
 * Author: Dariusz Tefelski <dariusz.tefelski@cern.ch>
 */
```

- Ustawienie automatycznego zapisu plików przed budowaniem. Wybierz z menu u góry **Window**–>**Preferences**–>**General**–>**Workspace**–>**Build** i zaznacz checkbox obok **Save automatically before manual build**. Naciśnij **Apply and Close**.
- Umożliwienie wpisywanie polskiego znaku „ć’ ’ (problem dotyczy Windows 7, pod Linux-em AltGr+c daje „ć‘‘). Należy wejść do **Window**–>**Preferences**–>**General**–>**Keys** i odnaleźć komendę **Insert ChangeLog entry**. Obok w kolumnie *Bind* będzie widoczna kombinacja klawiszy **Ctrl+Alt+C**. Bedą widoczne 2 takie komendy. W obu przypadkach należy, po zaznaczeniu komendy, kliknąć na przycisk **Unbind Command**. Na koniec nacisnąć **Apply and Close**. W ten sposób pozbędziemy się tego skrótu klawiszowego, do nieużywanej funkcji, a w zamian będzie można korzystać z polskiego znaku „ć”.
- Usunięcie kontroli pisowni. Sprawdzanie pisowni - ortografii w większości przypadków przy pisaniu kodu niepotrzebnie rozprasza - zwłaszcza, gdy przygotowujemy fragmenty tekstu, albo komentarze w innych językach niż angielski. Należy wejść do **Winow**–>**Preferences**–>**General**–>**Editors**–>**Text Editors**–>**Spelling** i odznaczyć checkbox **Enable spell checking**. Na koniec nacisnąć **Apply and Close**
- Utwórzmy projekt programu dla mikrokontrolera AVR ATmega32. Należy wybrać w menu u górze: **File**–>**New**–>**Project...**, albo kliknąć prawym klawiszem myszy w polu **Project Explorer** i wybrać **New**–>**Project...**. Następnie w zakładce **C/C++** wybrać **C Project**. Wpisać nazwę projektu np. “Startowy1”, a w polu **Project Type** otworzyć **AVR Cross Target Application** i wybrać **Empty Project**. W polu Toolchains powinien być zaznaczony **AVR-GCC Toolchain**. Nacisnąć przycisk **Next**. W kolejnym oknie w polu **Configurations** odznaczyć checkbox **Debug** (środkowe pole), gdyż nie będziemy z niego korzystać. Pozostać ma natomiast **Release**. Możemy w tym momencie kliknąć **Finish**i potem “wygodniej” ustawić typ mikrokontrolera i częstotliwość zegara, albo klinkąć **Next** i w następnym oknie wybrać **MCU Type** jako **ATmega32** i w polu **MCU Frequency (Hz)** ręcznie wpisac **16000000** i zakończyć przyciskiem **Finish**.

Ważne ustawienia w utworzonym projekcie

- Ustawienie optymalizacji rozmiaru kodu wykonywalnego. To ustawienie dotyczy otwartego projektu. Należy w **Project Explorer** kliknąć prawym klawiszem myszy na utworzonym projekcie np. “Startowy1” a następnie wybranie z menu **Properties**, i dalej w otwartym oknie, po lewej wybranie **C/C++ Build**–>**Settings**, a następnie w zakładce **Tool Settings** otworzyć **AVR Compiler**–>**Optimization** i po prawej w polu **Optimization Level** zamiast **No Optimizations (-O0)** wybrać **Size Optimizations (-Os)**. Na koniec nacisnąć **Apply and Close**.
- Ustawienie dołączania tylko wymaganych funkcji z bibliotek, a nie dołączanie całego kodu biblioteki w trakcie łączenia (linkowania) plików wynikowych. Należy w **Project Explorer** kliknąć prawym klawiszem myszy na utworzonym projekcie np. “Startowy1” a następnie wybranie z menu **Properties**, i dalej w otwartym oknie, po lewej wybranie **C/C++ Build**–>**Settings**, a następnie w zakładce **Tool Settings** otworzyć **AVR C Linker**–>**General** a następnie w polu **Other Arguments** po prawej dodać: **-Wl,--gc-sections**. Na koniec nacisnąć **Apply and Close**.
- Ustawienie właściwego typu mikrokontrolera (jeśli nie zrobiliśmy tego wcześniej) i częstotliwości zegara. Należy w **Project Explorer** kliknąć prawym klawiszem myszy na utworzonym projekcie np. “Startowy1” a następnie wybranie z menu **Properties**, i dalej w otwartym oknie, po lewej wybranie **AVR**–>**Target Hardware** i w polu **MCU Type** wybranie **ATmega32** a w polu **MCU Clock Frequency** wybranie **16000000**. Na koniec nacisnąć **Apply and Close**.
- Ustawienie programatora dla programu AVRDude. Należy w **Project Explorer** kliknąć prawym klawiszem myszy na utworzonym projekcie np. “Startowy1” a następnie wybranie z menu **Properties**, i dalej w otwartym oknie, po lewej wybranie **AVR**–>**AVRDude** i w zakładce **Programmer configuration** należy kliknąć przycisk **New**. W otwartym oknie u góry w polu **Configuration name** nazwać programator np. **USBasp**, a w polu **Programmer Hardware (-c)** wybrać **USBasp, http://www.fischl.de/usbasp/**, a następnie kliknąć przycisk **OK** w prawym dolnym rogu okna. Na koniec nacisnąć **Apply and Close**.
- Usunięcie zbędnej weryfikacji zaprogramowanego kodu w mikrokontrolerze. Należy w **Project Explorer** kliknąć prawym klawiszem myszy na utworzonym projekcie np. “Startowy1” a następnie wybranie z menu **Properties**, i dalej w otwartym oknie, po lewej wybranie **AVR**–>**AVRDude** i w zakładce **Advanced** w polu **Verify Check (-V)** należy zaznaczyć checkbox **Disable automatic verify check**. Na koniec nacisnąć **Apply and Close**.
- Przygotowanie nowego pliku w projekcie. Należy kliknąć prawym klawiszem myszy w **Project Explorer** na projekcie “Startowy1” a następnie wybrać w menu **New**–>**Source File**. W otwartym oknie, w polu **Source file:** wpisać **main.c**. W polu **Template** powinien być **Default C source template**. Nacisnąć **Finish**. W projekcie pojawi się nowy plik main.c i zostanie on także otworzony w edytorze w centralnej części Eclipse (jeśli jesteśmy w perspektywie C/C++).

Wpiszmy ponownie prosty kod testowy, mrugający diodą LED, tym razem zmieniając częstotliwość mrugania przez zmianę opóźnienia na 100 ms. Przypominam, że w Eclipse można łatwo skorzystać z pomocy (automatyczne uzupełnianie, sugerowanie rozwiązań) poprzez skrót klawiszowy **Alt + Space**.:

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

int main(void){

    DDRC=0xff;
    PORTC=0xff;

    while(1){
        PORTC^=(1<<PC0);
        _delay_ms(100);
    }

    return 0;
}
```

Uwaga, na kopiowanie kodu z instrukcji - niektóre znaki źle się kopiują (powstają niewłaściwe znaki), więc lepiej przepisać kod ręcznie.

- Zbudować projekt możemy używając przycisku z ikoną młotka w menu u górze (albo kombinacją klawiszy **Ctrl + B** - z tym, że budowanie realizuje wszystkie projekty, które są otwarte - i czasem może okazać się wygodniej ustawić skrót np. klawisz F7 do budowania tylko aktywnego (zaznaczonego) projektu. Opcja w **Window**–>**Preferences**–>**General**–>**Keys**. Wyszukać komendę **Build Project**, zaznaczyć i w polu Binding nacisnąć klawisz np. F7. Na koniec nacisnąć **Apply and Close**.
- Zaprogramować mikrokontroler możemy używając przycisku z napisem AVR i zieloną strzałką w dół, albo możemy użyć skrótu klawiszowego: **Ctrl + Alt + U**.
- Po wgraniu programu do mikrokontrolera powinnniśmy zaobserwować mruganie diody LED o większej częstotliwości. Można spróbować zmienić opóźnienie i sprawdzić jak dioda będzie mrugać.

Jaką częstotliwość mrugania diody LED w Hz uzyskamy przy wprowadzonym opóźnieniu w ms?

8. Instalacja wirtualnego portu szeregowego. Na płytce prototypowej port szeregowy jest łączony z

konwerterem USB/RS232 - jest to układ Future Technology FTDI FT232RL, który powinien być rozpoznawany przez Windows i sterowniki powinny zostać zainstalowane automatycznie. Jeśli tak nie jest należy pobrać sterownik ze strony: https://www.ftdichip.com/Drivers/VCP.htm i wskazać wyodrębnione pliki przy instalacji sterownika, albo posłużyć się wersją z instalatorem: Setup.

- Do obsługi portu szeregowego z poziomu Windows przyda się instalacja odpowiedniego oprogramowania, proponowane to:
  - bray terminal
  - realterm
  - hterm
  - yat
  - termie
  - termite
  - putty

System operacyjny GNU/Linux

Systemy operacyjne GNU/Linux są lepiej dostosowane do wykorzystania jako środowisko programistyczne. W większości przypadków wystarczy zainstalować potrzebne pakiety z dystrybucji systemu. Najczęściej będą to pakiety o nazwach **avr-gcc**, **avrdude**, **avr-libc**, **avr-binutils**. W różnych dystrybucjach mogą mieć trochę inną nazwę, ale zwykle pomaga wyszukanie *avr*.

- W dystrybucji Mabox Linux (ewentualnie Manjaro albo ArchLinux) odpowiednio wyszukuje się przez:

```
pacman -Ss avr
```

a instaluje przez np.

```
pacman -Sy avr-gcc avrdude avr-libc avr-binutils geany make
```

Podobnie jest w innych dystrybucjach, np w Ubuntu/Debian, wyszukanie poprzez:

```
apt search avr
```

a instalacja np. przez

```
apt install geany
```

W dystrybucjach opartych na Red Hat Linux

```
yum search avr
```

albo (Fedora)

```
dnf search avr
```

i podobnie instalacja jak wyżej.

- Konfiguracja środowiska Eclipse jest praktycznie identyczna jak przedstawiona w instalacji dla systemu MS Windows.
- Istotną kwestią pracy w systemie GNU/Linux jest nadanie uprawnień dla użytkownika do wykorzystania urządzeń podłączanych z zewnątrz. We współczesnych dystrybucjach jest to realizowane przez rozpoznanie podłączanego urządzenia i nadanie uprawnień wg plików konfiugracyjnych umieszczonych w katalogu **/etc/udev/rules.d**. W tym celu należy utworzyć jako **root** w tym katalogu plik o nazwie **50-usbasp.rules** zawierający następującą konfigurację.

```
# AVR USBASP
SUBSYSTEM!="usb", ACTION!="add", GOTO="usbasp_end"

SUBSYSTEMS=="usb", ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="05dc",
MODE="660", GROUP="uucp"

LABEL="asbasp_end"
```

Uwaga! W plikach udev każda linia jest traktowana jako reguła, w powyższym pliku *MODE=“660”, GROUP=“uucp”*, **musi** znaleźć się w jednej linii z *SUBSYSTEMS==“usb”*.

W plikach udev, w regułach “==” oznacza, sprawdzenie, czy coś jest spełnione, podobnie “!=” oznacza sprawdzenie “czy jest różne”, natomist pojedyncze “=” oznacza działanie - przypisanie, np. MODE=“660” oznacza nadanie praw do odczytu i zapisu dla właściciela (root-a) oraz dla grupy (w naszym przypadku uucp).

Jest ona przygotowana dla systemu opartego o Archlinux. Dotyczy to nazwy grupy, do której będzie przydzielone urządzenie. W Archlinux(Mabox Linux, Manjaro) grupa która może korzystać z urządzeń szeregowych (portów), to **uucp**. W innych dystrybucjach jest inna nazwa, np. w Debianie / Ubuntu i pochodnych jest to grupa **dialout**. Wystarczy w poniższym pliku zmienić nazwę *Group="uucp"* na *Group="dialout"*.

Aby użytkownik systemu mógł skorzystać z programator **USBasp** musi należeć też do stosownej grupy.

Polecenie *groups* w terminalu, pokaże do jakich grup należy użytkownik.

Użytkownika można dodać do grupy przez:

```
sudo usermod -a -G uucp darek
```

Gdzie oczywiście na końcu jest nazwa naszego użytkownika. Dla Debian/Ubuntu będzie to:

```
sudo usermod -a -G dialout darek
```

- Po ustawieniu grupy najlepiej się wylogować i zalogować ponownie. Jeśli pracujemy tylko w jednym terminalu - tymczasowo możemy wskazać, że mamy właściwą grupę, poleceniem:

```
newgrp uucp
```

Ale to zadziała tylko w tym jednym terminalu, w każdym innym trzeba będzie ponownie ją wpisywać, do czasu wylogowania i zalogowania ponownie, gdzie będzie to już utrwalone na stałe.

- Plik Makefile dla systemu GNU/Linux: AVR Makefile dla Linux
- Aplikacje do połączenia poprzez port szeregowy:
  - moserial
  - cutecom
  - hterm
  - gtkterm
  - picocom
  - minicom
  - screen
  - putty

Po podłączeniu płytki prototypowej kablem USB A-B do komputera, możliwe będzie wykorzystanie wirtualnego portu szeregowego do transmisji danych z i do mikrokontrolera. W tym momencie port dostępny będzie jako urządzenie **/dev/ttyUSB0**.

System operacyjny macOS

Ze względu na to, że system macOS na komputerach Apple oparty jest o jądro Unix-owe: NetBSD, kwestia wykorzystania tego systemu do programowania mikrokontrolerów AVR powinna być na podobnym poziomie trudności co dla systemu GNU/Linux.

Jednakże nie mam doświadczenia związanego z tą platformą. W internecie dostępne są różne materiały dotyczące programowania AVR pod macOS, np. po polsku http://marcingibas.pl/poradniki/6-jak-rozpoczac-programowanie-avr-na-macos.