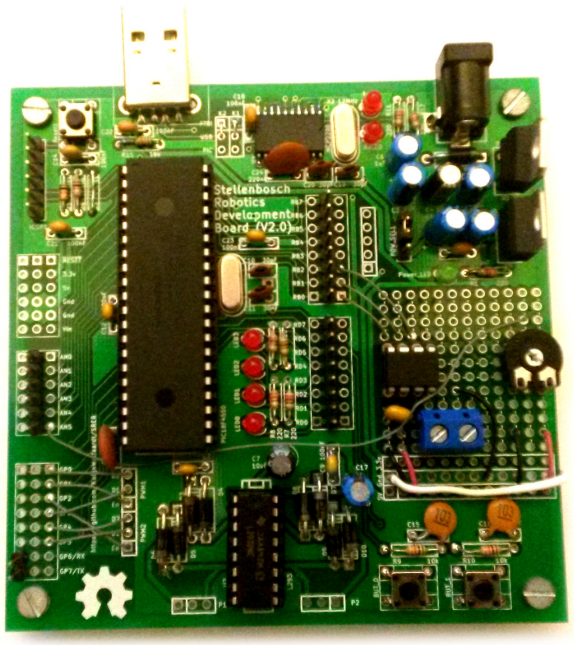
SREB Tutorial

August 18, 2014

Hello. This tutorial show you how to use the Stellenbosch Robotics dEvelopment Board.



1 Hardware

1.1 Power Supply

The main power supply of the board is used to convert battery power to the regulated 5v needed by the PIC. The power input is protected by a diode, to prevent reverse connection. The battery voltage needs to be above 7v to use the

board. The protection diode causes a 0.7v drop, and the 7805 regulator requires at least a 6.2v input. Drawing large ammounts of current from the board will probably cause an additional voltage drop on the battery.

The 5v source for the board is selectable between the battery and the usb. See section 1.6 for details. The power led will turn on when the 5v is supplied to the board. The board also contains a 3.3v regulator, should you need 3.3v for any devices.

1.2 PIC and USB

The PIC we are using is a PIC18F4550. It is an 8-bit PIC, with a built-in USB module.

1.3 GPIO Pins

The GPIO (General purpose input output) pins are laid out in a simmilar fashion as an arduino, but the board is not compatible with arduino shields.

GPIO pins The two banks to the right of the PIC, RB and RD contain general purpose pins which can be used for anything. RD5 to RD0 are used for the LEDs and buttons. You can connect other devices to these lines if you want to, as long as you are aware of the possible interaction. There is another GPIO bank, to the lowwer left of the PIC, called the GP bank. The bottom 2 pins, GP6 and GP7 are used for the UART transmit and receive lines, and should not be used for any other purpose. The rest of the bank should probably be used for the motor driver, since the pwm signals are also connected there.

Analog and Power The two banks to the left of the pic are power and analog input, respectively. Below the prototyping area are 3 power rails, one for 5v, one for 3.3v and one for ground. There is also another unlabeled power rail, close to the power selection jumper. This rail provides V_in, the input voltage after the protection diode.

1.4 Motor Driver

The motor driver is the L298B, and is used to control two motors. It has 6 control inputs, 3 for each motor. The motor driver receives power from the battery input, and will not work over usb power.

1.5 Buttons and LEDS

The buttons and LEDs are provided as a simple method of input an output. The LEDs are connected to ground via a current-limiting resistor. The buttons are connected so that they pull a line down to ground. The line is held high using a resistor. A small capacitor helps to prevent the switch bouncing.

1.6 Jumpers And configuration

Power selection There are two important configuration parts on the board. The first is the power select jumper. This allows you to select the 5v power source for the rest of the board. The input can be either the usb connector or the 5v regulated from the battery.

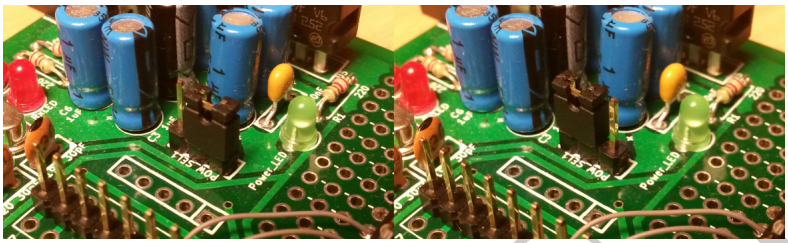


Figure 1: Left: Power from USB. Right: Power from battery

If the usb power is selected you should not try to run high-power devices from the board, such as motors. The motor driver on the board is not powered from usb. Instead it gets power from the battery input, so it will only work when a battery is connected.

USB data lines The usb data lines (D+ and D-) can be redirected between the MCP2200 IC, and the PIC. These should normally be connected to the MCP2200, as translate UART commands from the PIC to usb. The PIC has a built in usb module which you can use if you are programming the pic using your own programmer. This can be used to emulate various usb devices, such as keyboards or sound cards. However, this is not easy. The center connection is the usb input, the top connects to the MCP2200, and the bottom to the PIC.

2 Bootloader

The bootloader we are using is Tinybootloader `http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm` It was written for PICs, with the express goal of occupying very little memory in the PIC.

Basic conecpt The bootloader code is written to the PIC, and executes before the main program is run. A certain byte is sent to the PIC from the PC. If the PIC receives this before the timeout, it will start communicating with the PC and start programming. Else it will continue on to the main program.

Usage

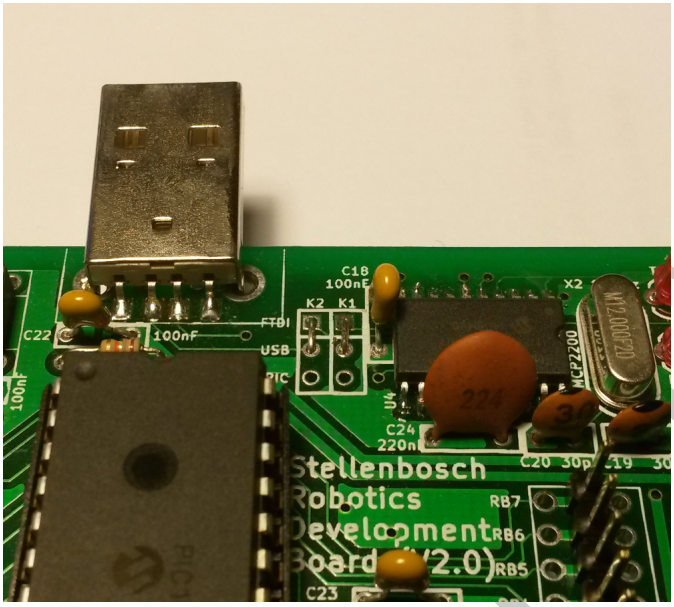


Figure 2: USB connection to usb-UART IC

1. Select the hex file you want to program to the device. If you compiled using the MPLABX environment, the file might be in one of the build output folders.
2. Select the correct connection settings for the UART. The bootloader uses a 115200 baud rate.
3. Click and hold the reset button on the board.
4. Click the **Write Flash** button, and then release the button on the board.
5. The bootloader should now program the device

It is not neccesary to use the **CheckPIC** button before writing to the PIC. It is only used to check if the bootloader detects the PIC.

The program also features a terminal interface, allowing you to send and receive messages from the PIC. An additional feature is the graphing function, which plots the received value as a byte. This is *very* helpfull when debugging sensor outputs. You can transmit the sensor value to the PC, and then whatch the output as you move the sensor.

On linux If you want to use TinyBootloader on linux you will need to run it under "http://en.wikipedia.org/wiki/Wine", a Linux compatibility layer for Windows. The details of installing and configuring wine will not be discussed here, and some familiarity with linux is assumed.

1. To use serial ports, you need to be a member of the dialout group. This group allows you acces to serial ports and modems. To show the groups you are currently added to:

```
groups
```

   If your are not in the dialout group, add yourself using

```
sudo adduser username dialout
```

   To apply the changes you need to log out and log back on.

2. Next you need to find out what device the board registers as. Plug in the board, and run the following command:

```
ls /dev
```

   The device should register as /dev/ttyUSB0 or /dev/ttyACM0

3. You need to tell wine about the device, so add a symbolic link called com1 to the device in the wine configuration folder. The wine configuration might be at a different location depending on your installation. You should also change the destination, the /dev/ttyACM0 part to the device the board registered as.

```
ln -s /dev/ttyACM0 ~/.wine/dosdevices/com1
```

4. Now run TinyBootloader in wine. It will not detect the com ports automatically, you will need to enter the com port by hand. After that it should work. Note that if you unplug and replug the board, you will need to restart TinyBootloader.

3 Writing Code

3.1 MPLABX IDE

The IDE we are using is MPLABX, which is supplied by Microchip. There is an accompaning compiler for the PIC, the XC8 compiler. These can be downloaded from the microchip site, and both work for windows and linux.

3.2 Skeleton code

Analog

UART

Timers

PWM

GPIO

4 Appendix: Git

5 Appendix: Full Schematic