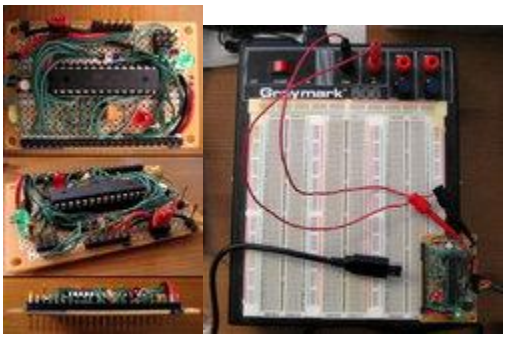
ARDUINO



If you have been following this blog at all you probably noticed that I have done a fair number of microcontroller projects. In my experience working with the PIC and AVR microcontrollers I ran into a number of issues:

1. The PICBasic programming environment , while easy to learn, only works on Windows
2. The C programming environment for the AVR requires more effort than I wish to put into a casual hobby enterprise and I have been unable to get it working in Linux

As I looked for more project ideas I noticed a lot of people using the Arduino development board. The Arduino is an open source hardware and software environment similar in concept to the BASIC Stamp (except it's not expensive).

Basically all the Arduino does is provide a standardized microcontroller board using the AVR ATMega168 processor and various power, I/O, and programming connections. They can be purchased as a completed board for around $35 (several versions of unassembled kits are also available). The biggest advantage from my perspective is that the Arduino software is truly cross-platform since it runs in Java and therefore can be used in Windows, Mac OSX and most importantly for me Linux.

One version called the Bare Bones Arduino removes the standard USB-serial adapter from the board itself and instead substitutes a FTDI USB-serial cable to connect the board to your PC. This is done to minimize cost since the adapter cable is a one-time $20 dollar purchase that can be used with an infinite number of compatible boards instead of paying for the adapter chip on the standard Arduino every time you get a new board. Another version of the Arduino called the Boarduino modifies the form factor of the circuit board into one more convenient for use on a solderless breadboard. otherwise it is essentially the same as the Bare Bones Arduino in that it also uses the FTDI adapter cable. Both of these boards are completely interchangeable with the Arduino.

While these are good products, I decided I wanted to build my own version from scratch to better fit my electronics setup.

I used the schematic from the Boarduino website to base my design on, but I used the same form factor as the Bare Bones Arduino. Since I use a Graymark 808 Protoboard which has a built in power supplies I removed the power circuitry from my design. In its place I simply put two headers, one for +5V and one for Gnd that connect to the protoboard's power supply (as shown in the pictures above). I also reduced the number of headers (which stab into the solderless breadboard) used to fit the Radio Shack PC board I used for my layout. I retained the power select jumper to choose whether the board is powered by the USB programming cable or the protoboard's power supply. I also left the two indication LEDs, reset button, and the 6 pin ISP and USB programming headers as they are on the Boarduino.

Total cost for my homebrew Arduino was $9 (not including the USB adapter cable).

- FTDI USB to TTL Serial Adapter Cable, 5V version
- Atmel ATMega168 microcontroller
- 16MHz ceramic resonator
- Tactile Pushbutton
- 40 Pin Strip Header
- Strip Header Shunt

- Radio Shack Multi Purpose PC Board, 417 holes

Having never dealt with the Arduino's software package before, I wasn't sure what to expect. It is a simplistic Java application which provides a very user friendly environment to write your programs (or *sketches* as they are called) in. The programming syntax used is similar to C, but the environment has many useful functions already built in so it's very easy to do simple tasks such as set a pin as an output or toggle an output on & off. As someone who has used other languages, PICBasic for example, I can attest to how intuitive these functions are when compared to manually setting register ports in BASIC. Like in C you can also create your own functions and call them, making this a very powerful language despite its simplicity.

The installation of the Arduino software is fairly straightforward, even on Linux, and I encountered no issues. In Ubuntu it entails downloading the application from the Arduino Software page and following their well written instructions. These mainly involve installing Java and removing a package which inadvertently thinks the Arduino is a braille reading device and grabs your computer's USB port.

Left out of the instructions is an issue which caused me some problems; the Arduino software should be run with root privileges in order to gain access to the USB port and consequently the Arduino board. This is done by opening the terminal and executing the following commands:

cd /home/username/Arduino-0010 (navigates to the Arduino software's folder)
sudo ./arduino (runs the Arduino software script with root permissions)

The application will now launch. Once running I selected my board under **Tools - Board - Arduino Diecimila**(currently the newest board design and bootloader) and picked my USB port under **Tools - Serial Port - /dev/ttyUSB0**. Since I built my Arduino from scratch, my ATMega168 did not come pre-burned with the Arduino Diecimila bootloader. The bootloader functions as a sort of operating system for the microcontroller, allowing you to transfer files over a serial port instead of having to re-burn the entire firmware every time you change your program, thus simplifying the entire process. In order to burn the bootloader I plugged my AVR programmer into the 6 pin ISP header on the board and selected **Tools - Burn Bootloader - w/ USBtinyISP**. The software displays its progress on the bottom of the screen and lets you know when it has finished burning the file to the chip. To check if the bootloader is running properly follow this guide (since the various bootloaders behave differently).

Next I wrote a simple LED flasher sketch and after plugging in the FTDI cable I successfully uploaded the sketch to the board and it ran perfectly.

Source: http://www.highonsolder.com/blog/2008/3/8/arduino.html