Introduction to Microcontrollers and Arduino

In this section of the course you are going to learn about microcontrollers and their applications. You should already have an Arduino board... if not, please see one of the instructors.

What is a Microcontroller?

A microcontroller is a small, self-contained computer usually in a single IC. They are used for many purposes, ranging from cell phones to microwave ovens to televison sets, plus many applications in science and instrumentation. Anything with a screen and buttons smaller than a laptop is likely to contain one or more microcontrollers. They're also used in science and engineering -- a microcontroller can acquire data in an experimental setup and perform initial processing and feedback to control parameters in the experiment, interact with the user and provide an interface to a conventional computer.

How do you program a Microcontroller?

Usually in C (programming language). Occasionally in Assembly Language or other languages. We will be using C to program our microcontrollers in this class. The code is entered and compiled on a standard computer and downloaded into the microcontroller using a USB interface. The program is stored in flash memory (think USB stick) on the microcontroller chip itself and is *non-volatile,* meaning that it stays programmed until you over-write it with new code.

Features of most microcontrollers:

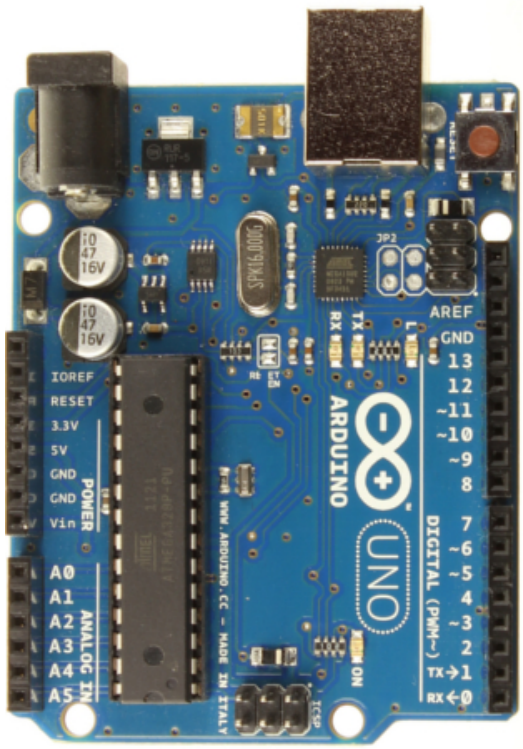
- CPU (processor, similar to CPU chip in a conventional computer)
- Program storage (flash memory) — Program code
- Volatile data storage (RAM) — Variables
- Non-volatile data storage (EEPROM) — Parameters
- Input/Output (analog, digital, serial USB etc)

What is an Arduino?

An Arduino is a small board with a microcontroller on it, packaged conveniently for experimenting. The one we will use in this class is an Arduino Uno R3, which has an ATMega328P microcontroller, along with a USB interface for programming and data transfer. The Arduino provides an easy-to-use programming environment and set of libraries, which work well on any common OS (Windows, Linux and OS-X are all supported).

Arduino Hardware

See http://arduino.cc/en/Main/arduinoBoardUno for details



This is an Arduino. Note the labeled sockets which can easily be connected to your breadboard with wires.

GND - most important!
Connect to ground (0v) on your breadboard

Digital - pins 0..13 can be used as digital inputs or outputs. Each can have two levels:
0 = 0V (output) or less than ~ 2V (input)
1 = 5V (output) or greater than ~3V (input)
Use **digitalWrite()** or **digitalRead()** to control

PWM - pins 3, 5, 6, 9, 10, 11 can be used as analog outputs using *pulse-width modulation* using the **analogWrite()** function

Analog In - pins A0-A5 measure voltages from 0-5V using the **analogRead()** function

3.3V and **5V** provide (low-current) power outputs. They *do not* need to be connected.

A few more Arduino notes:

Use the USB cable to connect your Arduino to your computer. This will provide it with power, so you do not need to supply voltage to the 3.3V or 5V pins (*and you should not!*)

Pin 13 has an LED connected on the board. When the pin is high (1) the LED is on.

The **PWM** outputs use pulse-width modulation (read http://arduino.cc/en/Tutorial/PWM).
An RC filter is required if you want DC (10k and 1uF is a good start).

The **Analog In** measure voltages from 0-5V with a 10 bit ADC. Each call to **analogRead** requires about 100uS so the maximum sampling rate you can achieve is 10 kHz[1]

[1] Faster rates up to 48kHz are possible, see i.e.
http://www.dayofthenewdan.com/projects/arduino-dso/index.html

Programming

Perhaps by now you have already figured this out, but you need to download the Arduino software and get it running on your computer. Read http://arduino.cc/en/Guide/HomePage. Read this, and figure out how to run the “blink” example program (**File->Examples->01.Basics->blink**) in the Arduino software.