Learning How to Code in Arduino

Worksheet 1: The Basics

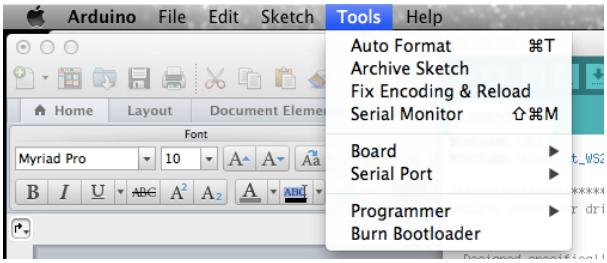
Jason Krugman – Physical Computing - Fall 2012

***This guide is your basic step-by-step Arduino process that you will go through every time you use the software interface
It doesn't tell you all the setup and debugging info – for that, go to http://arduino.cc/en/Guide/HomePage

I. Connecting your Arduino board to your computer – Arduino file menu

Every time you plug your Arduino into your computer (using one of the USB inputs), you will need to set 2 parameters in the Arduino's menu, both under the "Tools" heading –

1. Board - The type of Arduino board you are using, for example "Uno"
2. Serial Port - The port that you are connected to. Usually set automatically but you should check



II. The basic outline of each program

There 3 sections of every Arduino program you write:

1. The top part where you will declare your global variables (see part V - Variables for definition of "global variables")

2. The "Setup" - where you layout the initial conditions for the program so that the Arduino knows what kind of work its going to be doing

3. The "Loop" - runs over and over and over doing your bidding

For example:

```
//variables (not included yet)

void setup(){

}

void loop(){


}
```

That was a quick example. The above shows the basic format for creating these 3 sections.

The line of code below is a comment, meaning that it doesn't do anything. It just reminds you to put some variables there. Its invisible to the Arduino because of the slashes at the beginning

```
//variables (not included yet)
```

This one is your setup function:

```
void setup(){

}
```

And this one is your loop function:

```
void loop(){

}
```

What is a function you may ask? A function is basically like a little machine created out of code… you build it and when you call its name, it does something. Setup and Loop are functions that are built into Arduino. Arduino always expects to see them because they are essential parts of its structure. Why the "void"? You don't have to know that but if interested look here: http://arduino.cc/en/Reference/Void

What are the parentheses and curly brackets for? The parentheses don't do anything for these two functions but they are essential. We will use them later on when we start to write more functions. The curlies → {} delineate the information inside the function. Think algebra and order of operations. Remember PEMDAS? You are basically telling the Arduino what information pertains to the Setup or Loop operation…. where is starts and where it ends.

Now I am going to write some code that explains how comments work.

```
//i am writing a comment here by using the "//". this means that I am
//effectively hiding this from the Arduino. It won't cause it to do
//anything. there are two ways to write comments
//the way where you begin each line with the back slashes, or you can
//also do use a slash and an asterisk

/*and now you can write a whole bunch of stuff in here and you don't
have to use the slashes for each line, just the beginning and end with
the asterisk. to end the comment, i will do this */

/*notice the order of the slash and asterisk to start is different than
to end */

/*writing comments with the slash and asterisk is very useful when you
want to deactivate a large block of code but not delete it. it will
still be there but the arduino will not see it
*/
```

Okay now for a program with no comments that actually does something. Notice my declaring the variables, and then including void setup( ) { } and void loop( ) { }. Afterwards, I will include the same program except with comments so you can follow along better.

```
int ledPinJason = 5;
int ledPinJasonVal = 0;


void setup(){

pinMode(ledPinJason, OUTPUT);

}

void loop(){

digitalWrite(ledPinJason, HIGH);
delay(100);
digitalWrite(ledPinJason, LOW);
delay(100);

}
```

That wasn't so bad. Now lets review. I declared (coding speak for "created") two variables. The variables' names are "ledPinJason" and "ledPinJasonVal". Notice that some of the letters are capitals and other not. This way of writing variable names is called "camelCase" …. because it has humps like a camel. The Arduino language is case sensitive, meaning that "ledPinJason" is a totally different thing than "ledpinjason". So why would I not make it simpler and just use all lowercase? Because it's easier to read the other way, and it's also an easy formula to follow when naming your variables.

So what is the "int"? Int is a type of variable, short for "integer". When you create a variable, you have to tell Arduino what *kind* of variable it is so that it knows how much memory to allocate to it. Some variables hold numbers, others characters, and others can hold long lists of numbers or characters. For a list of all the variable types, look at the Data Types listed here:
http://arduino.cc/en/Reference/HomePage
For now, int is the only one you will need to know… it will take you a long way as we get started.

Why is one variable set equal to five and the other zero? Hold on, we will get there soon.

In Arduino, most lines end with a semicolon. Be patient, you'll see the pattern.

Okay, so now we're at setup. void setup() {] is the function that sets up the Arduino so it knows whats what. In setup, we tell the Arduino the initial paremeters of what we are going to be doing in the program (which follows after in the loop and sometimes after the loop too).

what is "pinMode"?
When we write, `pinMode(ledPinJason, OUTPUT);` we are telling Arduino that the variable ledPinJason is an output. Since we set it equal to 5 when we "initiated" it (another coding word for "created"), the Arduino now knows that digital pin number 5 is an output. This refers to the little socket on the Arduino board with the number 5 next to it in the digital section.

```
void setup() {

  pinMode(ledPinJason, OUTPUT);

}
```

See how Arduino colors the different words? This is because it recognizes some of them from its internal structure. If you were to write, "pinmode" with no capital "m" or "output", it would not recognize them and you would have problems. This is because Arduino is case sensitive.

Okay, moving on to the loop. digitalWrite is a built in function (we know this because it will change color when we write it in Arduino) that does what it says… it digitally writes something. By "write" we mean sends a voltage to an output pin. Kind of like writing a message.

Notice the word digital, which means "on and off". There is no grey in digital. High or low, true or false, black or white, yes or no. The Arduino's digital outputs can "write" out either 0 or 5 volts, and nothing in between. They can also receive either 0 or 5 volts, but we will get to that later.

Okay, so lets move on to the loop. Notice how we ended the setup function with a }

This lets the Arduino know that we are moving on. Here's where we are in case you forgot:

```
int ledPinJason = 5;
int ledPinJasonVal = 0;


void setup(){

pinMode(ledPinJason, OUTPUT);

}

void loop(){

digitalWrite(ledPinJason, HIGH);
delay(100);
digitalWrite(ledPinJason, LOW);
delay(100);

}
```

So after we digitalWrite ledPinJason, HIGH, we delay for 100, then write ledPinJason, LOW, then delay again. And that's that! Then the program will go to the top of the loop and do the same thing, write ledPinJason, high, delay 100, write low, delay, repeat…. etc. It will keep doing this forever…. or as long as nothing fails or runs out of power.

The delay function is built in to Arduino as well. It takes an argument in milliseconds, and stops the whole program for that long. "Argument" is the word we use for whatever variable we pass in to a function. 100 is 100 milliseconds (or thousanths of a second). 100 ms is .1 seconds.

delay(1000) would be one second.

A common mistake is to not wait (delay) twice when doing a program like this. If we were to write:

```
void loop(){
digitalWrite(ledPinJason, HIGH);
delay(100);
digitalWrite(ledPinJason, LOW);
}
```

the time between the Arduino writing ledPinJason, LOW, and writing it HIGH would be so fast that we wouldn't even see it…. so the LED we attach to digital pin 5 would just appear to stay on all the time.

At the end of the loop, the program just goes back to the top of the loop again and does the same thing… over and over and over…. etc.

Each cycle through the loop happens really really fast. The Arduino can run at 16mHz, or 16 million cycles per second. The actual speed it takes for it to run through a simple loop like this is uncertain (it depends on a few different things), but for our purposes, its really really fast.

So what does this program actually do? Well, it does nothing unless you hook up a circuit to it…. but it can turn something on and off. Could be an LED, or it could also be something totally different, like a transistor that is amplifying the on/off signal to turn on a big relay, that could be controlling…… an electric leaf blower (I tried to think of the most random device possible).

Resources:

Getting Started – help setting up your computer with software and getting things up and running
http://arduino.cc/en/Guide/HomePage

Basic Digital Input / Output – ITP Tutorials for basic setup
http://itp.nyu.edu/physcomp/Labs/DigitalInOut