Techshop EEE-201: Basic programming

Chapter 1

Welcome to chapter 1! Let's get started learning the basics of how to use an Arduino. In this chapter, we will cover the following topics:

- `setup()` function
- `loop()` function
- The serial monitor and the `println()` function
- Using comments

Table of contents

Introduction

In this class, we will be working with the Educato board - an Arduino compatible microcontroller board. For all the material that follows, I'll simply be referring to it as Arduino since that's the platform & software we'll be using. So what is a microcontroller and what is Arduino?

You can think of a microcontroller as a simple, mini-computer; let's see what wikipedia has to say about it:

> A microcontroller (or MCU, short for microcontroller unit) is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

So, basically, it's an all in one computer package packed into a single chip. Understand that I'm using the word computer pretty loosely here - it by no means compares to standard desktop or laptop computers. Microcontrollers are designed to do pretty basic tasks, have very little memory (comparatively) and can be slow. But for what we will be accomplishing in this class, it's a perfect platform!

So what about Arduino? The following quote is taken from the arduino.cc site:

> Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

You can think of Arduino as a hardware/software package that we can use for all our DIY projects. We will be writing code, or a set of instructions, that will tell the microcontroller what to do. This set of instructions is also known as a sketch and is named so because the Arduino platform was designed for artists. The sketches are written on a regular desktop or laptop and then "uploaded" to the Arduino through their software IDE.

Part 1 - bare minimum

Let's get started! Open the Arduino IDE and copy/paste the BareMinimum.ino sketch. You should see two sections of code, the first looks like:

```
void setup() {
  // put your setup code here, to run once:

}
```

The `setup()` function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the board.

Next we see the `loop()` section of the sketch:

```
void loop() {
  // put your main code here, to run repeatedly:

}
```

The `loop()` function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond as it runs. Code in the `loop()` section of your sketch is used to actively control the board.

Any line that starts with two slashes (`//`) will not be read by the Arduino, so you can write anything you want after it. The two slashes may be put after functional code to keep comments on the same line. Commenting your code like this can be particularly helpful in explaining, both to yourself and others, how your program functions step by step.

Part 2 - serial monitor

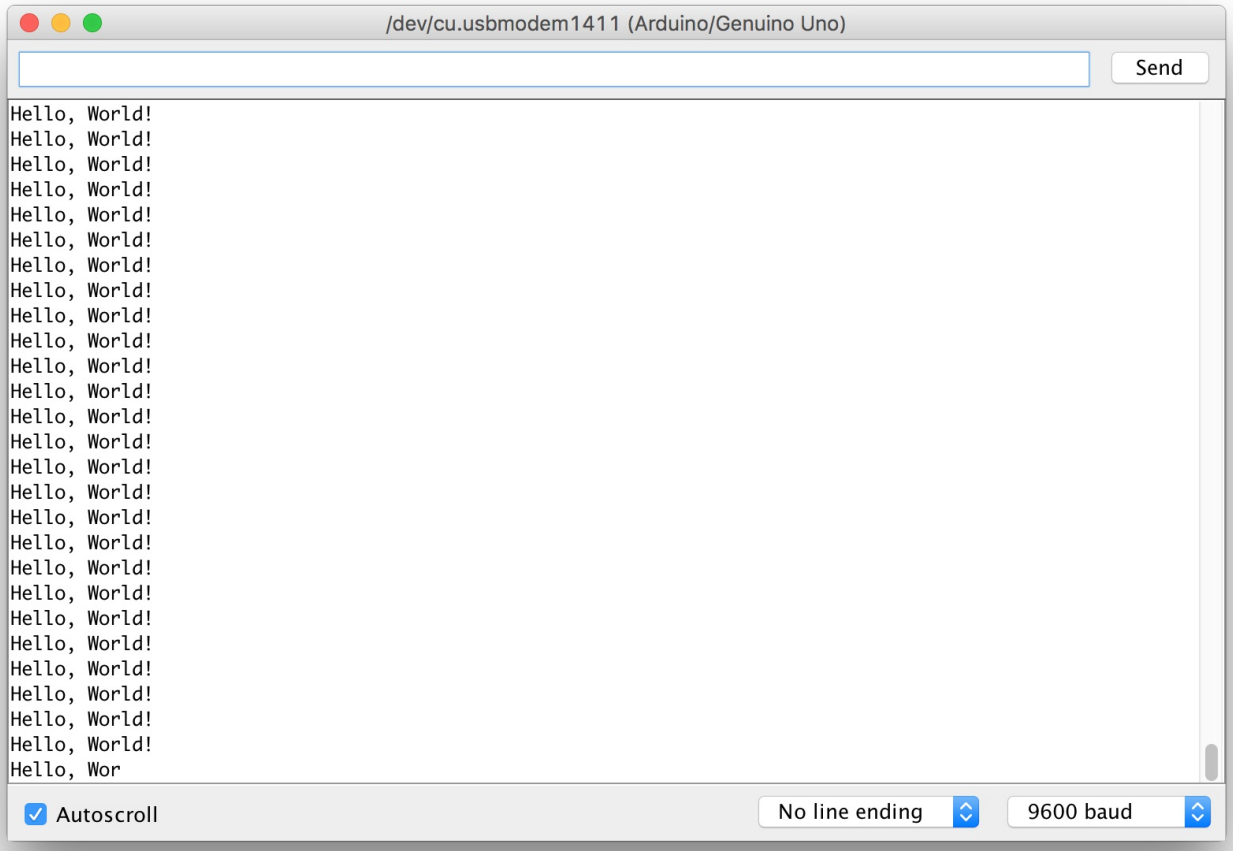
So far we haven't actually done anything with the Arduino, let's fix that! We begin with the quintessential *Hello, World!* example; copy/paste HelloWorld.ino into your IDE. Next we will want to upload our sketch to the Arduino so that it knows what we want it to do; here's how:

1. Open the Arduino software (IDE)
2. Copy and paste your sketch into the window
3. Save a copy somewhere
4. Click the *Upload* button which looks like a right-arrow or choose from the menu *Sketch -> Upload*

If you have any problems uploading your sketch, check these two things:

- You have the proper board selected under *Tools -> Board* (it should be `Arduino/Genuino Uno`)
- You have the proper port selected

After uploading has finished, open the serial monitor (*Tools -> Serial Monitor*) and you should see something like this:



Let's walk through how this works; the first part of our sketch looks like this:

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // open the serial port at 9600 bps
}
```

All we've done is add the line `Serial.begin(9600)` - what this means is that we are telling the Arduino we want to use the serial monitor and that we are going to use it at a speed (also known as baud of 9600 bits per symbol (bps). The serial monitor is our way of communicating between our desktop/laptop and the Arduino; we're going to use it to let the Arduino "talk" to us. The idea of baud can get pretty confusing, so for now just think of it as how fast we are sending data.

Now that we've got the serial monitor setup, let's start using it!

```
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello, World!"); // print our message
}
```

Remember, the `loop()` just repeats *Ad nauseam*; in this case, we use the `println()` function to print text to it so that we can see it on our screen. Since that's the only line of code we've added, and since the execution of the code will simply repeat itself, the Arduino keeps interpreting that print statement over and over.

Explore

- what happens if you change the line of code to `Serial.print("Hello, World!")`?