Lab 1: Arduinos

September 19, 2016

Contents

1 Objectives 1

2 Pre-Lab: Connecting your Arduino 1

3 Blinking an LED 2

4 Switch 3

5 Fading an LED with analog output 5

6 Rainbow fade 6

1 Objectives

By the end of this lab, you will:

• Connect your arduino to your computer
• Connect and control an LED
• Create analog outputs

We will be providing the parts necessary for this lab (other than the Arduino).

2 Pre-Lab: Connecting your Arduino

1. Download the software specific to your OS at http://arduino.cc/en/Main/Software

2. Install the software:

• OSX: The .dmg should mount automatically, if not, double-click it. Simply drag the Arduino app to your Applications folder
• Windows (using the .exe installer): Follow the prompted installation, making sure to leave ”Install USB driver” checked. For more detailed instructions, use https://www.arduino.cc/en/Guide/Windows
• Linux:
  – Extract the compressed TAR into a suitable folder.
  – Open a terminal and CD into the directory you just extracted into and run:

        root@dev: ~$ ./install.sh

3. Launch the Arduino application

4. Connect your Arduino with the USB cable

5. Select your board using the software by going to Tools > Board.

6. Check the connection port by going to Tools > Serial Port
   
   • OSX and Linux: This should be '/dev/tty.usbmodem' or something similar
   
   • Windows: This will likely be COM3 or higher

7. Test the board connection by loading up the blink test sketch: File > Examples > 1.Basics > Blink.

8. You should see the code of the sketch. Don’t worry about what the code does yet, we just want to test that it will upload

9. Click the upload button. After a few seconds, the Arduino app should say ”Done uploading”. If it has an error, call over a TA.

3 Blinking an LED

Now we are going to make the physical connections to make an LED blink using. You will need:

1. A resistor between 200Ω and 1.5KΩ

2. Either a single color or RGB LED.

Hookup your circuit to match this circuit diagram:

![Basic LED circuit diagram]

Figure 1: Basic LED circuit

With your arduino and patch board, it should look something like this:
Note: an RGB LED is a little more complex. Use this schematic to hookup your RGB LED to use just one color (as if it were a single color LED):

Your LED should blink using the example Blink sketch. Make you understand why the sketch is making the LED blink and call over a TA for any clarifying questions.

**Task:** Create your own sketch to blink the LED in any non-symmetric on/off timing. For example, having the led off for a second and then on for two seconds.

## 4 Switch

You will need:

1. 1 LED.
2. 1 pushbutton switch
Create the following switch circuit:

![Switch Circuit Diagram](image)

Figure 4: switch circuit

![Arduino Board and Breadboard Diagram](image)

Figure 5: source: [arduino.cc](http://arduino.cc)

Note that direction of this switch is not obvious based on its 4 pins. Here, the two right pins of the pushbutton are internally connected as well as the left two. Pushing the button (closing the circuit) bridges the right and left sides.

When the button is unpressed (open), the input pin is connected to ground through the resistor. When the button is pressed (closed), the input pin is connected to 5V.
and reads HIGH. Note the concept of the pull-down resistor. If we did not include it, we would get inconsistent and unpredictable results since the input pin is essentially dangling and the slightest ambient noise could cause it to read HIGH or LOW. By grounding the input pin with the 10KΩ resistor, we ensure it reads LOW when open.

In order to read the button, add this to your sketch:

1. In the sketch setup, setup the input pin using `pinMode()` (https://www.arduino.cc/en/Reference/pinMode)
2. In the main loop, read the state of the button using `digitalRead()` (https://www.arduino.cc/en/Reference/DigitalRead)

Task: Make a state change for your switch (like pressing it) trigger some task, like turning on an LED.

5 Fading an LED with analog output

This section will use the same circuit as the previous section.

The arduino can only output digital values: high or low. If we want to create values inbetween, we need to use Pulse Width Modulation (PWM) (https://en.wikipedia.org/wiki/Pulse-width_modulation). PWM essentially creates an output that is rapidly changing between high and low. When connected to an LED, this rate is rapid enough that our eyes cannot detect it. Instead, the period of on and off blend together to make the LED appear dimmer versus being constantly on. We are going to use this method to create a fade effect with the LED. Consider the following PWM diagram:

![PWM Diagram]

By using these various PWMs, we can create three distinct brightness levels. Note that the term ”Duty Cycle” which is the proportion of high to low.
The Arduino has built-in PWM functionality on several of its pins which vary by board type. If your current circuit is using a PWM capable pin, you do not need to change it. Otherwise, rewire your circuit to use a PWM pin.

**Task:** using the `analogWrite()` function (https://www.arduino.cc/en/Reference/AnalogWrite), create a sketch to fade an LED between off and on.

## 6 Rainbow fade

You will need:

1. 3 resistors between 200Ω and 1.5KΩ
2. 1 RGB LED.

Modify your circuit to connect the R, G, and B LED pins by performing the same wiring steps previously in the lab for the two remaining LED pins. You will now have three output pins from your Arduino going into your LED.

To create a RGB rainbow effect, you will use 3 sine waves in different phases:

![Figure 7: source: markandey.com](markandey.com)

The peak of these waves correspond to 255, the maximum brightness output.

**Task:** Using the sine curve, create a sketch to fade the LED through the rainbow.

Note: The Arduino sin function can be slow. We can speed this up by trading some precision by creating a lookup table. While this is not necessary for this lab, keep it in mind for future projects. Additionally, a linear approximation of the sine wave is acceptable. Consider parametrizing the rate as a constant.