Lab 1: Build Your Own Bikelight

Prava Dhulipalla and Anupama Krishnan

September 2017

1 Introduction

In Lab 1, we learned how to build our own "bike light"–a circuit with at least three LEDs and a push button that cycles the LEDs through five different modes when pressed. In the second part of the lab, we added a potentiometer to our system of LEDs, and used it to control the brightness of the LEDs and the speed at which they flash. Our five different modes were:

- Red light on
- Yellow light on
- All lights blink simultaneously
- Lights alternate blinking (the speed of the blinking is controlled by the potentiometer)
- The lights can be dimmed and brightened using the potentiometer

2 Procedure

We built a circuit on a solderless breadboard using an Arduino (and our laptops to power the Arduino). To start building the circuit, we first referred to the schematic provided on the webpage for the lab report. However, it was missing resistor values for the LEDs, so we then referred to the spec sheet for the LED lights and calculated our resistor values.

According to the LED data sheet, the electrical/optical characteristic forward voltage for the LEDs is usually around 1.9V and the current at that voltage is 0.010A. Using these values, the 5V value input, and Ohm's law, we then calculated the approximate value for resistors that we should use:

$$V = IR$$

$$(5V - 1.9V) = (0.010A)R$$

$$R = 310\Omega$$

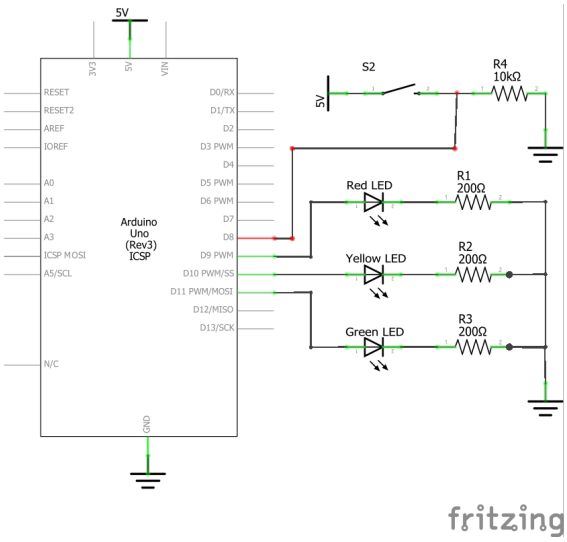

Figure 1: Original Circuit

The calculated resistor value was 310 Ω, but for simplicity we used 200 Ω resistors for each LED. The resistors are used to limit current to the LED. If the current is not limited to an acceptable threshold, excessive voltage will pass through the LED and cause electrical damage. For our lab, we chose three different color LEDs: red, yellow, and green. The NINJAs informed us that we could use the same data sheet for all three, so the current-limiting resistor values ended up being the same.

There is a pull down resistor connected to the pushbutton, which means that the button will read HIGH when pressed. This ended up affecting the way we coded the lab. We connected the Arduino ports, the resistors, and the LEDs to the breadboard according to the schematic seen in Figure 1.

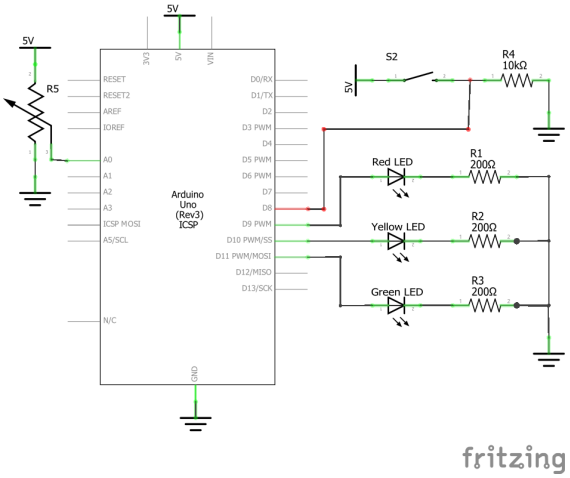

Figure 2: Circuit with Potentiometer Added

For the second part of the lab, we added a potentiometer (see Figure 2). A potentiometer acts like a variable resistor. In this case, we used the potentiometer in two of our modes: one to change the rate at which the LEDs blinked and the other to change the brightness of the LEDs.

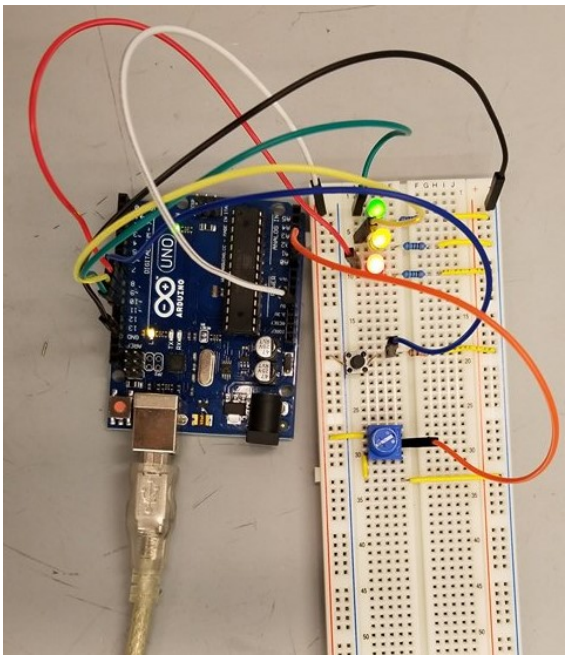

Figure 3: Physical Breadboard

3 Code

3.1 Debouncing

In order to debounce, two important variables need to be defined: the last time the output pin has been toggled, and a threshold for how long since the output pin has been toggled last. The button state will change if the button has been pushed down once, and if the state fluctuates within the threshold, it is not reflected in the button state. In this section, we also use a counter to keep track of how many times the button has been pressed. Each time one button press is counted after debouncing, the counter value increases by one.

```
// debouncing
unsigned long lastDebounceTime = 0;
unsigned long debounceThreshold = 50;

// button press counter
int count = 0;
```

```
 // makes sure it isn't reading one press as multiple presses
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  // makes sure only one button press is being counted, sets a time threshold
  if ((millis() - lastDebounceTime) > debounceThreshold) {

    // checks whether the button state has been changed
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == HIGH) {
        // only counts if the button has been completely pressed
        count ++;
      }
    }
  }
```

3.2 Delays (or Lack Thereof)

Initially, we were using delays for our blinking function. We soon learned that delays are bad for the Arduino, as they shut down the entire Arduino for a certain period of time. Instead, we used the millis() function in place of our delay, which returns the number of milliseconds since the Arduino began running the program. We would record the milliseconds from a 'starting point', collect it from a 'current point', and check to see if they were greater than an certain interval (essentially a delay). We then used if statements to toggle between lights all on and lights all off.

```
// for blinking
unsigned long previousMillis = 0;
unsigned long interval = 100;
unsigned long currentMillis = 0;
```

```
  // Mode 3
  else if (count%3 == 0) {
    // blinks the lights
    currentMillis = millis();
    if (currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      if (redLEDState == LOW & yellowLEDState == LOW & greenLEDState == LOW){
        greenLEDState = HIGH;
        yellowLEDState = HIGH;
        redLEDState = HIGH;
      }
      else {
        greenLEDState = LOW;
        yellowLEDState = LOW;
        redLEDState = LOW;
      }
    }
  }
```

3.3 Analog Versus Digital + Our Modes

Digital signals are discrete. In terms of LEDs, this means that they can either be HIGH or LOW. Analog signals are continuous, which means that they can fluctuate between HIGH and LOW values, changing the brightness of the LED. We used digitalRead() to read the button value because it is either compressed or uncompressed. However, to read the potentiometer, we used analogRead() because the value of the potentiometer fluctuates depending on what you choose to set it at.

For our fading/brightening mode, we used analogWrite() to send the state values to the LEDs. We did not have to do that for our potentiometer controlled blinking mode because the potentiometer value was not controlling the brightness of the LED, but the speed with which it was turning on and off. All of our modes other than the brightening/dimming one (Mode 1), used digitalWrite().

We used the modulo function in order to toggle through the states. This was done because we could easily add in more modes afterwards by moving the if statements down and changing the main if statement with modulo, instead having to reset the counter at a different part and set a different limit even if we just checked for equality.

The first two modes simply send a HIGH value to the desired LED. The third mode is blinking lights (using millis instead of delays). The forth mode used the potentiometer to change speed of the blinking lights. The fifth mode mapped the potentiometer values to dim the LEDs.

```
void loop() {
  // read button value
  int reading = digitalRead(button);

  // read potentiometer value
  potValue = analogRead(potentiometer);
```

```
  // Mode 5
```

```
  if (count%5 == 0) {
    // maps the potentiometer values between 0-255 to fade and brighten the LEDs
    dimmerVal = map(potValue,0,1023,0,255);
    greenLEDState = dimmerVal;
    yellowLEDState = 255 - dimmerVal; // causes yellow light to brighten/dim in the opposite
        way
    redLEDState = dimmerVal;
    }
```

```
  // Mode 4
  else if (count%4 == 0) {
    // uses the potentiometer values to change the speed with which the lights flash
    if(millis()>timenow + potValue){
      if (redLEDState == LOW & greenLEDState == LOW){
        greenLEDState = HIGH;
        yellowLEDState = LOW;
        redLEDState = HIGH;
      }
      else {
        greenLEDState = LOW;
        yellowLEDState = HIGH;
        redLEDState = LOW;
      }
      timenow = millis();
      }
    }
```

```
  // Mode 3
  else if (count%3 == 0) {
    // blinks the lights
    currentMillis = millis();
    if (currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      if (redLEDState == LOW & yellowLEDState == LOW & greenLEDState == LOW){
        greenLEDState = HIGH;
        yellowLEDState = HIGH;
        redLEDState = HIGH;
      }
      else {
        greenLEDState = LOW;
        yellowLEDState = LOW;
        redLEDState = LOW;
      }
    }
  }
```

```
  // Mode 2
  else if (count%2 == 0) {
    // turns on the yellow light
    redLEDState = LOW;
    yellowLEDState = HIGH;
    greenLEDState = LOW;
    }
```

```
 // Mode 1
  else if (count%1 == 0) {
    // turns on the red light
    redLEDState = HIGH;
    yellowLEDState = LOW;
    greenLEDState = LOW;
  }
```

```
  if (count%5 == 0) {
    // uses analogWrite() to dim and brighten the lights
    analogWrite(redLED, redLEDState);
```

```
    analogWrite(yellowLED, yellowLEDState);
    analogWrite(greenLED, greenLEDState);
  }
  else {
    // uses digitalWrite() turn the lights on and off
    digitalWrite(redLED, redLEDState);
    digitalWrite(yellowLED, yellowLEDState);
    digitalWrite(greenLED, greenLEDState);
  }
```

4 Problems and Reflection

There were many sources of frustration, mainly surrounding 'stupid mistakes' that we made throughout the course of the lab. For instance, in the beginning our circuit didn't work, but this was because the ground wire was actually connected to the Vref port on the Arduino. In addition, there were several typos in the code that caused a slew of errors (luckily we caught them). The bulk of our initial mistakes were encompassed by small errors like this.

The largest difficulty we had was with the concept of button presses, debounce, and delays. Push buttons have mechanical problems that result in accidentally tracking multiple button presses with one press. This is contrary to the main function of a button–where one press is equal to one press. In order to address this problem we used debounce. This was another problem, because neither of us were familiar with the concept of debouncing. Button debouncing checks the buttonpress twice in a specified interval of time in order to make sure the button was pressed - and then it is only tracked once. Finally, we didn't realize that 1) delays are bad for the Arduino and that 2) we weren't supposed to use them for the lab. We used them initially for our blinking mode. However, because they act like a blocking function - turning off the Arduino completely - it stops all other processes from happening.
We also realize we could have done some things better in the lab. For coding, instead of modulo, we could have used cases - essentially a way to make the code simpler. We could have also used more functions in the code. It's not required in this case because it wouldn't have made the lab coding any less hard, but if we wished to expand the scope of this lab it would have been necessary.

After doing this lab, we both learned a lot about Arduino coding and good practices. For instance, debouncing is often a recommended technique when using buttons as more than just a switch–it prevents a button press from being 'read' more than once. Similarly, we learned not to use the delay function in our code, as it's harmful to the Arduino. The delay function is what is known as a 'blocking function' and it prevents all other processes from occurring during that time period. It is especially preventative when used in conjunction with button presses. When delay() is called, button presses are not tracked either. The alternative suggested method is using millis(), which tracks the time (in milliseconds). We can then use the time to perform other functions, like delays, without shutting down the entire Arduino.

And, overall, it gives us good insight to the thought process one might go through in order to make a bike light...

5 Appendix

Below is the entirety of our code.

```
// assigning the Arduino pin values to our components
int redLED = 9;
int yellowLED = 10;
int greenLED = 11;
int button = 8;
int potentiometer = A0;

// declaring initial states of other components
```

```
int redLEDState = HIGH;
int yellowLEDState = HIGH;
int greenLEDState = HIGH;
int buttonState = 0;
int lastButtonState = LOW;
int potValue = 0;
int timenow = 0;
int dimmerVal = 0;

// debouncing
unsigned long lastDebounceTime = 0;
unsigned long debounceThreshold = 50;

// button press counter
int count = 0;

// for blinkings
unsigned long previousMillis = 0;
unsigned long interval = 100;
unsigned long currentMillis = 0;

void setup() {
  // Declaring pin modes
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(button, INPUT);
}

void loop() {
  // read button value
  int reading = digitalRead(button);

  // read potentiometer value
  potValue = analogRead(potentiometer);

  // makes sure it isn't reading one press as multiple presses
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  // makes sure only one button press is being counted, sets a time threshold
  if ((millis() - lastDebounceTime) > debounceThreshold) {

    // checks whether the button state has been changed
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == HIGH) {
        // only counts if the button has been completely pressed
        count ++;
      }
    }
  }

  // Mode 1
  if (count%5 == 0) {
    // maps the potentiometer values between 0-255 to fade and brighten the LEDs
    dimmerVal = map(potValue,0,1023,0,255);
    greenLEDState = dimmerVal;
    yellowLEDState = 255 - dimmerVal;
    redLEDState = dimmerVal;
    }

  // Mode 2
  else if (count%4 == 0) {
    // uses the potentiometer values to change the speed with which the lights flash
    if(millis()>timenow + potValue){
      if (redLEDState == LOW & greenLEDState == LOW){
```

```
77            greenLEDState = HIGH;
78            yellowLEDState = LOW;
79            redLEDState = HIGH;
80          }
81          else {
82            greenLEDState = LOW;
83            yellowLEDState = HIGH;
84            redLEDState = LOW;
85          }
86          timenow = millis();
87          }
88        }
89
90      // Mode 3
91      else if (count%3 == 0) {
92        // blinks the lights
93        currentMillis = millis();
94        if (currentMillis - previousMillis > interval) {
95          previousMillis = currentMillis;
96          if (redLEDState == LOW & yellowLEDState == LOW & greenLEDState == LOW){
97            greenLEDState = HIGH;
98            yellowLEDState = HIGH;
99            redLEDState = HIGH;
100         }
101         else {
102           greenLEDState = LOW;
103           yellowLEDState = LOW;
104           redLEDState = LOW;
105         }
106       }
107     }
108
109     // Mode 4
110     else if (count%2 == 0) {
111       // turns on the yellow light
112       redLEDState = LOW;
113       yellowLEDState = HIGH;
114       greenLEDState = LOW;
115       }
116
117     // Mode 5
118     else if (count%1 == 0) {
119       // turns on the red light
120       redLEDState = HIGH;
121       yellowLEDState = LOW;
122       greenLEDState = LOW;
123     }
124
125     if (count%5 == 0) {
126       // uses analogWrite() to dim and brighten the lights
127       analogWrite(redLED, redLEDState);
128       analogWrite(yellowLED, yellowLEDState);
129       analogWrite(greenLED, greenLEDState);
130     }
131     else {
132       // uses digitalWrite() turn the lights on and off
133       digitalWrite(redLED, redLEDState);
134       digitalWrite(yellowLED, yellowLEDState);
135       digitalWrite(greenLED, greenLEDState);
136     }
137
138     lastButtonState = reading;
139 }
```

6 References

We referred to a couple of sources throughout our lab. These mainly surrounded new Arduino functions/ideas we wanted to learn more about and implement in our lab set-up:
Debounce: *https://www.arduino.cc/en/Tutorial/Debounce*
Millis: *https://www.arduino.cc/en/Reference/Millis*
Potentiometer: *https://www.allaboutcircuits.com/projects/using-the-arduinos-analog-io/*