PoE Lab 1: Bike Light

Anusha Datar

September 2018

1 Description

For this lab, I created a bike light with three LEDs, five main modes, and a distance sensor. The first two modes are all LEDs on and all LEDs off, both of which toggle the digital pins associated with the LEDs to either high or low. The third and fourth modes involve the distance sensor: in the third mode, as an object gets closer to the sensor, the lights progress from green to red and get brighter. The fourth mode has the opposite distance sensor response, so the lights progress from red to green and get brighter depending on the distance that an object is away from the sensors. I used empirically determined thresholds to calibrate the distance sensor and I used the PWM feature of the digital pins I set for the LEDs to adjust the brightness of the lights. The fifth mode sequentially turns on and off the LEDs in a blinking pattern.

To switch between modes, I use a counter and an interrupt to listen for a button press. To ensure that the mechanical limitations of the button would not compromise the accuracy of button presses, I check that at least 200 milliseconds have passed prior to the earlier switch within the interrupt handler. Prior to adding this debouncing function, I was facing very unpredictable behavior from the switch, but when I added it, my bike light consistently maintained the desired behavior.

2 Deliverables

2.1 Schematic

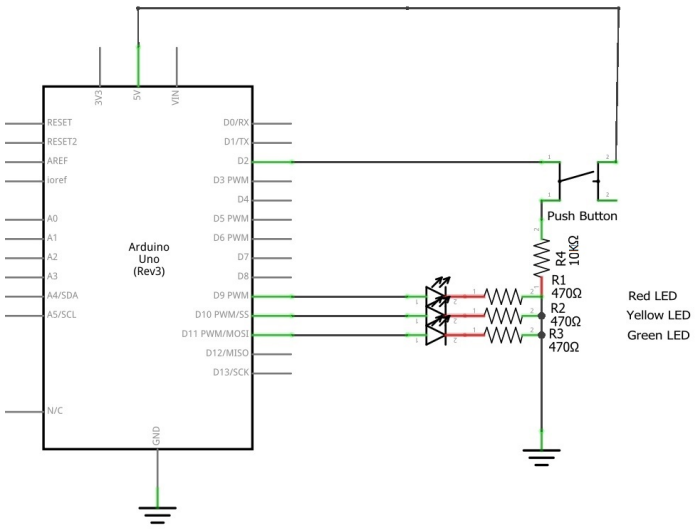

Figure 1: Schematic associated with bike light

2.2 Source Code

This code was compiled and run through the Arduino IDE.

```
1
2  // Number of possible modes. While the cases and
3  // incremements are 0 indexed, this should just be
4  // the counting number of cases.
5  int numberOfPossibleCases = 5;
6
7  // Pin Definitons.
8  // LED Pins.
9  int redPin = 9;
10 int yellowPin = 10;
11 int greenPin = 11;
12 // Input values from distance sensor.
13 int distPin = A0;
14 // Counter for modes based on input from switch.
15 int switchCount = 0;
16
17
18 void setup() {
19   Serial.begin(9600);
20   // Set up pins
21   pinMode(switchPin, INPUT);
22   pinMode(redPin, OUTPUT);
```

```
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(switchPin, INPUT);
  pinMode(distPin, INPUT);
  // Kick off interrupts to make it easier to switch modes.
  interrupts();
  // Interrupt on switch on pin two.
  attachInterrupt(digitalPinToInterrupt(2), loopSwitch, HIGH);

}

void loop() {
  /*
   * Allows for mode selection based on a simple counter and
   * then kicks off relevant functions.
   */
  switch (switchCount) {
    case 0:
      allHigh();
      break;
    case 1:
      allLow();
      break;
    case 2:
      distanceSensorResponse();
      break;
    case 3:
      distanceSensorOppositeResponse();
      break;
    case 4:
      blinking();
      break;
  }
 }

void allHigh() {
  /*
   * Mode 0 :Sets all of the LEDs to high.
   */
  Serial.print("Mode␣0:␣All␣LEDs␣High.␣\n");
  digitalWrite(redPin, HIGH);
  digitalWrite(yellowPin, HIGH);
  digitalWrite(greenPin, HIGH);
}

void allLow() {
  /*
   * Mode 1 : Sets all of the LEDs to low.
   */
  Serial.print("Mode␣1:␣All␣LEDs␣Low.␣\n");
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, LOW);
}

void distanceSensorResponse() {
  /*
```

```
    * Mode 2 : Adjusts brightness and color (from green for farther
    * and red for closer) of LEDs based on distance sensor reading.
    */
  Serial.print("Mode␣2:␣Lights␣incremement␣and␣become␣brighter␣");
  Serial.print("based␣on␣distance␣sensor␣results.␣\n");
  allLow();
  int distanceRead = analogRead(distPin);
  // Tolerances are based on empirical testing.
  if (distanceRead > 150) {
    if (distanceRead > 255) {
      distanceRead = 255;
    }
    analogWrite(greenPin, distanceRead);
  }
  else if (distanceRead > 75) {
    analogWrite(yellowPin, distanceRead);
  }
  else {
    analogWrite(redPin, distanceRead);
  }
}

void distanceSensorOppositeResponse() {
  /*
   * Mode 3 : Adjusts brightness and color (from green for farther
   * and red for closer) of LEDs based on distance sensor reading.
  */
  Serial.print("Mode␣3␣:␣Lights␣decremement␣and␣become␣brighter␣");
  Serial.print("based␣on␣distance␣sensor␣results.␣\n");
  allLow();
  int distanceRead = analogRead(distPin);
  // Tolerances are based on empirical testing.
  if (distanceRead > 150) {
    if (distanceRead > 255) {
      distanceRead = 255;
    }
    analogWrite(redPin, distanceRead);
  }
  else if (distanceRead > 75) {
    analogWrite(yellowPin, distanceRead);
  }
  else {
    analogWrite(greenPin, distanceRead);
  }
}

void blinking() {
   /*
    * Mode 4 : Blinks LEDs from red to green and then resets.
   */
  Serial.print("Mode␣4␣:␣Blinking␣\n");
  digitalWrite(redPin, HIGH);
  delay(500);
  digitalWrite(yellowPin, HIGH);
  delay(500);
  digitalWrite(greenPin, HIGH);
  delay(500);
```

```
137   digitalWrite(redPin, LOW);
138   delay(500);
139   digitalWrite(yellowPin, LOW);
140   delay(500);
141   digitalWrite(greenPin, LOW);
142 }
143
144 void loopSwitch() {
145   /*
146    * Interrupt handler for when the button is pushed.
147    *
148    * Uses timer for debounce protection (mainly due to
149    * mechanical limitations of the button) and then
150    * incremements the switch counter. If the switch
151    * counter has exceeded the number of cases, the value
152    * resets to 0.
153   */
154   static unsigned long prior_interrupt_time = 0;
155   unsigned long current_interrupt_time = millis();
156   //
157   if (current_interrupt_time - prior_interrupt_time > 200) {
158     if (switchCount < (numberOfPossibleCases - 1)) {
159       switchCount++;
160     }
161     else {
162       switchCount = 0;
163    }
164   }
165   prior_interrupt_time = current_interrupt_time;
166 }
```

3 Reflection

This lab exercise proved to be a straightforward and engaging way to interface with the arduino board and its programming environment, especially since the basic idea of the project was simple but there were a variety of possible augmentations such as using input from the distance sensor and adjusting the brightness of the lights without an analog pin. While there were some interesting questions to answer such as using PWM and debouncing the signal from the button in the interrupt handler, there were not major roadblocks, which allowed me to explore other challenges.