Lab 1 - BYOBL: Build Your Own Bike Light

Noah Rivkin, Seungin Lyu

September 2017

1 Introduction

In this lab, we built a bike light circuit to control five red LEDs switching between five different operational modes. We built the circuit in a solderless breadboard by implementing the schematic shown in Figure 1. We used an Arduino Uno as a single-board microcontroller for the circuit. We used a push button so that the user can switch between different modes. Additionally, we used a potentiometer to provide analog input for the Arduino. Then, we used the value from the analog input to vary the interval between LED blinking.

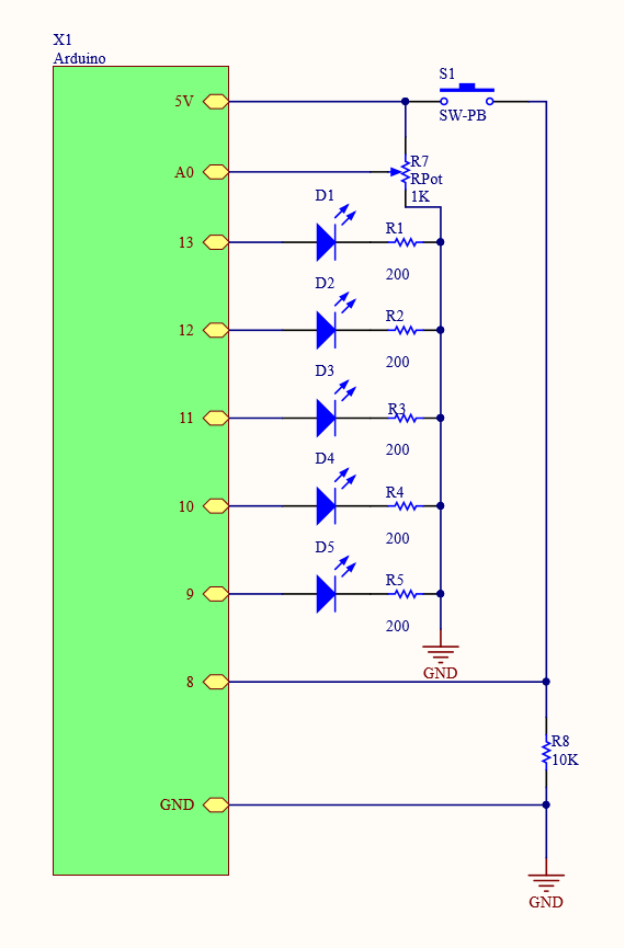

Figure 1: The schematic for the bike light circuit.

2 Procedures

We began by creating the LED part of the circuit in isolation. We used five red LEDs. A link to the LED datasheet is included in the appendix. According to the datasheet, the LEDs have a maximum operating current of 30 mA. We decided to use resistors in series with the LEDs to create a current of 20 mA. At 20 mA the voltage drop across the LED is approximately 2 V. The Arduino digital output is 5 V, resulting in a voltage drop of 3 V across the resistors. We used Ohm's law to find the ideal resistor value.

$$V = IR \tag{1}$$

$$3V = .02A * R \tag{2}$$

$$R = \frac{3V}{.02A} = 150\Omega \tag{3}$$

Using Ohm's law (V = IR), we determined that the ideal resistor value would be 150Ω. We chose to use the closest value possible with a single resistor, 200Ω.

$$I = \frac{V}{R} = \frac{3V}{200\Omega} = .015A \tag{4}$$

This resulted in a current of 15 mA, well within the LEDs operating range found in the datasheet.

Once we successfully turned on all 5 LEDs with the 5V input from the Arduino, we began implementing five different operational modes for the bike light. This step involved both writing the control code on the Arduino and the addition of a push button to the circuit. The button used a 10kΩ pull-down resistor. We implemented 5 modes, which could be alternated between using the button. We included a list of these operational modes in Figure 2.

1. All LEDs OFF
2. ALL LEDs ON
3. ALL LEDs Blinking at an interval
4. LEDs bouncing back and forth at an interval
5. Blinking a randomly selected LED at an interval

Figure 2: A list of operational modes

To begin writing the Arduino sketch, we first took a look at the Arduino's official documentation found on its website. We included the link to this documentation in the appendix for future reference. We began by declaring the digital input and output(I/O) pins as global variables as shown in Figure 3. We allocated digital I/O pins from 9 to 13 as output pins that provide current through each of the LEDs and pin 8 as the input pin for the push button. We chose these pin numbers arbitrarily, so choosing any other digital pins should produce the same results.

Then we began implementing the loop() function. We began by declaring the global variables and functions that are used in the loop, as shown in Figure 4. We chose not use an array to store all the pins together because we believed writing code for each pin individually makes it easy to modify and extend the code later. Using arrays in this code may help simplify the code but does not make any noticeable difference in execution.

We implemented the push button logic as shown in Figure 5. Notice that we chose not to use the delay() function but used millis() instead. Check the Arduino reference for detailed comparison. We made this design choice to avoid getting stuck in forced delays caused by the delay() function. We also declared a

```
// OUTPUT Pin Number Declaration
int ledPin1 = 13;
int ledPin2 = 12;
int ledPin3 = 11;
int ledPin4 = 10;
int ledPin5 = 9;

// INPUT Pin Number Declaration
int inPin = 8; // button input (digial)

// VALUES read from each INPUT Pins
int buttonIn = 0;

void setup() {
    // put your setup code here, to run once:
    pinMode(ledPin1, OUTPUT);
    pinMode(ledPin2, OUTPUT);
    pinMode(ledPin3, OUTPUT);
    pinMode(ledPin4, OUTPUT);
    pinMode(ledPin5, OUTPUT);
    pinMode(inPin, INPUT);
    Serial.begin(9600); // Sets the data rate for serial data transmission.
}
```

Figure 3: Digital I/O Pins Setup

```
// Variables used in loop()
int mode = 0;
unsigned long previousClickTime;
unsigned long previousBlinkTime;
boolean ledAllOn = true; // led flag for all leds

// Mode 3 (Bouncing Mode)
int bouncePin = 9;
int previousBouncePin;
boolean bounceBack = false;

// Mode 4 (Random Mode)
int randPin;
int previousRandPin;

// Turn off all the LEDs, set ledAllOn to false
void turnAllOff(){
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
    digitalWrite(ledPin4, LOW);
    digitalWrite(ledPin5, LOW);
    ledAllOn = false;
}

// Turn on all the LEDs, set ledAllOn to true
void turnAllOn(){
    digitalWrite(ledPin1, HIGH);
    digitalWrite(ledPin2, HIGH);
    digitalWrite(ledPin3, HIGH);
    digitalWrite(ledPin4, HIGH);
    digitalWrite(ledPin5, HIGH);
    ledAllOn = true;
}
```

Figure 4: Global variables and Functions used in loop()

clickInterval(300ms) variable to set a forced interval between each clicks to avoid accidental double clicks. In addition, we declared the constant variable blinkInterval to set an interval for blinking. We turn off all the LEDs of before switching to the next mode to reinitialize the state of the LEDs.

```
void loop(){
   unsigned long currentTime = millis();
    const long clickInterval = 300; // sets 300ms interval to prevent double-clicks
    const long blinkInterval = 300;
    buttonIn = digitalRead(inPin);

    // Checks if the button is clicked and if the click is after the clickInterval
    if((buttonIn == HIGH) && (currentTime - previousClickTime >= clickInterval)){
        previousClickTime = currentTime;
        mode = (mode + 1) % 5; // increments modes between 0 ~ 4
        turnAllOff(); // Turn off all the LEDs before switching to next mode
    }


}
```

Figure 5: Declaration of Variables within loop()

We used switch-case statements to implement the logic for different operational modes. Figure 6 includes the codes for mode 1 and mode 2.

```
 switch(mode){
        case 0: // All off
        turnAllOff();
        break;
        case 1: // All on
        turnAllOn();
        break;
    }
```

Figure 6: Switch-Case Implementation of Modes

We used variables previousBlinkTime and blinkInterval to set an interval between each blink for mode 3,4 and 5 (as shown in Figure 7). We used the boolean ledAllOn flag to check the status of the LEDs.

```
 case 2: // All flashing Mode
        if (currentTime - previousBlinkTime >= blinkInterval) {
            previousBlinkTime = currentTime;
            // if the LED is off turn it on and vice-versa:
            if (!ledAllOn) {
                turnAllOn();
                } else {
                turnAllOff();
            }
        }
        break;
```

Figure 7: blinkInterval Logic with millis()

We included the codes for mode 4 (the bouncing mode) in Figure 8. Notice that we have a variable named previousBouncePin that stores the previous pin. We turn off the previously blinked LED before turning on the current LED connected to bouncePin. We use the boolean flag bounceBack to check the direction of the LEDs.

```
case 3: // Bouncing Mode
        if (currentTime - previousBlinkTime >= blinkInterval) {
            previousBlinkTime = currentTime;
            digitalWrite(previousBouncePin, LOW);
            digitalWrite(bouncePin, HIGH);
            previousBouncePin = bouncePin;
            if(!bounceBack){
                bouncePin++;
                if(bouncePin > 13){
                    bounceBack = !bounceBack;
                    bouncePin = 13;
                }
            }
            else{
                bouncePin--;
                if(bouncePin < 9){
                    bounceBack = !bounceBack;
                    bouncePin = 9;
                }
            }
        }
```

Figure 8: Bounce LEDs Implementation

For mode 5 (Random Mode), we used the Arduino random function to generate a random pin number between 9 and 13 inclusively (as shown in Figure 9).

```
        case 4: // Random led blinking Mode
        randPin = random(9, 14);
        if (currentTime - previousBlinkTime >= blinkInterval) {
            // save the last time you blinked the LED
            previousBlinkTime = currentTime;
            digitalWrite(randPin, HIGH);
            digitalWrite(previousRandPin, LOW);
            previousRandPin = randPin;
        }
        break;
        default:
        Serial.print(\\\"error\\\");
        break;
```

Figure 9: Code for loop()

Once we confirmed that all the operational modes were working, we moved onto adding an analog device to the circuit. We chose to use a 10kΩ potentiometer to provide an analog input to the Arduino. We allocated analog pin A0 as the analog input pin, and the anagloIn variable stored the value from A0 pin. To vary the blinking interval according to the state of the potentiometer, we set the blinkInterval variable with a value mapped from the original analog input value that ranges from 0 to 1023 to a range from 100(ms) to 1000(ms) as shown in Figure 10. This range is an arbitrary choice, so you are free to choose any range you prefer.

3 Reflection

- Seungin Lyu : This lab was a great "hello world" project into designing a simple circuit using the Arduino micro controller. I learned that it is very important to refer to the official datasheet and the documentation to resolve any uncertainties before moving too far ahead with the project. For example, I had written down the entire code with "delay()" function to create intervals between clicks

```
1
2 int analogInPin = 0; // potentiometer input A0
3 int analogIn = 0;
4
5 void loop(){
6 // Reads the potentiometer value from analog input pin, then maps the value to blinkInterval
7     analogIn = analogRead(analogInPin);
8     int blinkInterval = map(analogIn, 0, 1023, 100, 1000); // maps the original analog
          output to 100ms to 1000ms
9 }
```

Figure 10: Using analog input to set blinkInterval

and each blinking. However, a Ninja pointed out that the circuit would get stuck in the delay and won't allow the push button input to interrupt and force change the mode. Then I read the official Arduino reference and found the millis() function that resolved the issue. I could've saved some time if I had carefully done the mini-research before I jumped right into coding. But overall, I really enjoyed this first POE lab!

- Noah Rivkin : This lab helped me to remember the importance of proper documentation, both for hardware and for software. I was not familiar with the millis() function prior to this lab, and I think it will be very useful going forward. It also helped emphasize the importance of making sure that the hardware and the software are designed to with each other in mind, rather than fitting one to the other.

4 Appendix

1. Red LED Datasheet : `http://www.kingbrightusa.com/images/catalog/SPEC/WP7113ID.pdf`
2. Arduino Official Reference : `https://www.arduino.cc/en/Reference/HomePage`
3. Source Code :

```
1  // OUTPUT Pin Number Declaration
2  int ledPin1 = 13;
3  int ledPin2 = 12;
4  int ledPin3 = 11;
5  int ledPin4 = 10;
6  int ledPin5 = 9;
7
8  // INPUT Pin Number Declaration
9  int inPin = 8; // button input (digial)
10 int analogInPin = 0; // potentiometer input A0
11
12 // VALUES read from each INPUT Pins
13 int buttonIn = 0;
14 int analogIn = 0;
15
16 // Variables used in loop()
17 int mode = 0;
18 unsigned long previousClickTime;
19 unsigned long previousBlinkTime;
20 boolean ledAllOn = true; // led flag for all leds
21
22 // Mode 3 (Bouncing Mode)
23 int bouncePin = 9;
24 int previousBouncePin;
25 boolean bounceBack = false;
26
27 // Mode 4 (Random Mode)
28 int randPin;
```

```
int previousRandPin;

// Turn off all the LEDs, set ledAllOn to false
void turnAllOff(){
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
    digitalWrite(ledPin4, LOW);
    digitalWrite(ledPin5, LOW);
    ledAllOn = false;
}

// Turn on all the LEDs, set ledAllOn to true
void turnAllOn(){
    digitalWrite(ledPin1, HIGH);
    digitalWrite(ledPin2, HIGH);
    digitalWrite(ledPin3, HIGH);
    digitalWrite(ledPin4, HIGH);
    digitalWrite(ledPin5, HIGH);
    ledAllOn = true;
}

void setup() {
    // put your setup code here, to run once:
    pinMode(ledPin1, OUTPUT);
    pinMode(ledPin2, OUTPUT);
    pinMode(ledPin3, OUTPUT);
    pinMode(ledPin4, OUTPUT);
    pinMode(ledPin5, OUTPUT);
    pinMode(inPin, INPUT);
    Serial.begin(9600);
}
void loop() {
    unsigned long currentTime = millis();
    const long clickInterval = 300; // sets 300ms interval to prevent double-clicks
    buttonIn = digitalRead(inPin);

    // Checks if the button is clicked and if the click is after the clickInterval
    if((buttonIn == HIGH) && (currentTime - previousClickTime >= clickInterval)){
        previousClickTime = currentTime;
        mode = (mode + 1) % 5; // increments modes between 0 ~ 4
        turnAllOff(); // Turn off all the LEDs before switching to next mode
    }
    // Reads the potentiometer value from analog input pin, then maps the value to
        blinkInterval
    analogIn = analogRead(analogInPin);
    int blinkInterval = map(analogIn, 0, 1023, 100, 1000); // maps the original analog
        output to 100ms to 1000ms
    switch(mode){
        case 0: // All off
        turnAllOff();
        break;
        case 1: // All on
        turnAllOn();
        break;
        case 2: // All flashing Mode
        if (currentTime - previousBlinkTime >= blinkInterval) {
            previousBlinkTime = currentTime;
            // if the LED is off turn it on and vice-versa:
            if (!ledAllOn) {
                turnAllOn();
                } else {
                turnAllOff();
            }
        }
        break;
        case 3: // Bouncing Mode
        if (currentTime - previousBlinkTime >= blinkInterval) {
```

```
95              previousBlinkTime = currentTime;
96              digitalWrite(previousBouncePin, LOW);
97              digitalWrite(bouncePin, HIGH);
98              previousBouncePin = bouncePin;
99              if(!bounceBack){
100                 bouncePin++;
101                 if(bouncePin > 13){
102                     bounceBack = !bounceBack;
103                     bouncePin = 13;
104                 }
105             }
106             else{
107                 bouncePin--;
108                 if(bouncePin < 9){
109                     bounceBack = !bounceBack;
110                     bouncePin = 9;
111                 }
112             }
113         }
114         break;
115         case 4: // Random led blinking Mode
116         randPin = random(9, 14);
117         if (currentTime - previousBlinkTime >= blinkInterval) {
118             // save the last time you blinked the LED
119             previousBlinkTime = currentTime;
120             digitalWrite(randPin, HIGH);
121             digitalWrite(previousRandPin, LOW);
122             previousRandPin = randPin;
123         }
124         break;
125         default:
126         Serial.print(\\\"error\\\");
127         break;
128     }
129 }
```