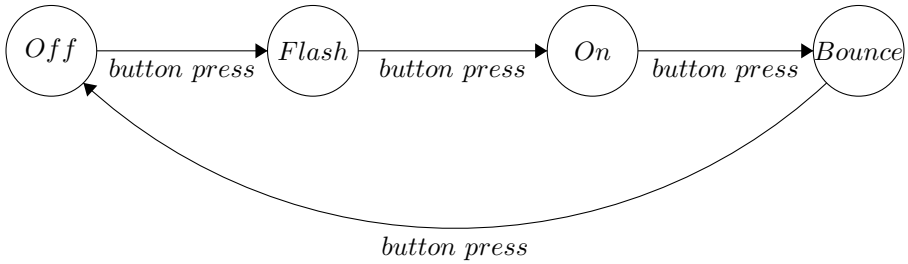
POE Lab Report: Building a Bike Light

Dennis Chen & Jennifer Wei

September 2015

1 Bike Light Description

We built a bike light with four modes of operation: off, flashing, on, and bouncing. All LEDs in the bike light are on or off in the on and off modes (respectively). In the flashing mode, the LEDs blink together. In the bouncing mode, the LEDs turn on and off, blinking in turn. Pushing a button switches between modes. A potentiometer controls the brightness of all LEDs in the flashing, on, and bouncing modes. Below is a finite state machine diagram for our bike light.



2 Circuit Diagram

The schematic for our bike light is shown in figure 1. We used digital pins 6, 5, and 3 on the Arduino to power the LEDs because of their PWM (Pulse Width Modulation) capability, which allows us to easily dim the LEDs. To keep the system simple while still demonstrating the four modes of operation, we used three LEDs.

To calculate the resistance of the resistors connected to the LEDs, we considered what voltage across the LEDs would be safe. A datasheet told us that we should put a maximum voltage of 2.5V across the LEDs, and a current of 30 mA would flow through them. Our arduino supplied 5V, so we calculated that the nessecary resistance was 83.3 Ohms from Ohm's Law: $83.3\Omega = \frac{(5V-2.5V)}{30mA}$

We connected the button pulldown resistor to a digital pin on the Arduino so that we could read the voltage across it and detect button presses. Then we chose to connect the center of a potentiometer to A0 on the Arduino so we could read it's analog voltage and then adjust the brightness of the LEDs accordingly.

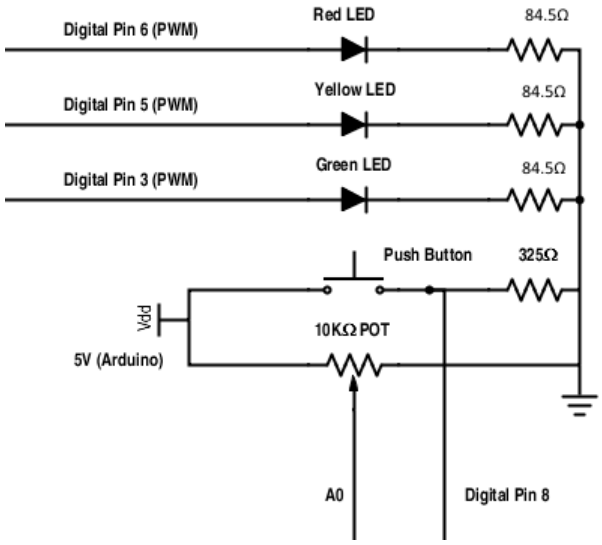


Figure 1: Circuit diagram for our bike light

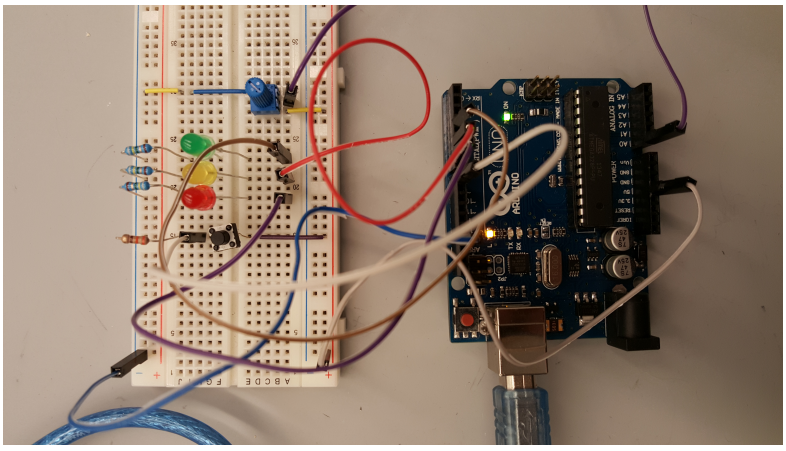


Figure 2: Breadboard for bike light circuit

3 Code

We first wrote a script to change the light mode when the button is pressed. To do this, we defined the modes as "states", where 0 is off, 1 is flashing, 2 is bouncing, and 3 is on (Lines 15-19). Then, we made a loop function. When the button is pressed, we check the current state and change to the next state (using the *switchState* function, Line 78). From there, we check the state and call the appropriate function (flash, on, bounce, or off).

To debounce the button input, we added a button press counter (*buttonPressedCount*). The button press counter is used to debounce the input so that the state only changes once for each button press (used in the *detectButtonPress* function, Line 64). To achieve this, the state only changes when *buttonPressedCount* is over a threshold.

We initially tried to time the bouncing and flashing modes by using the *delay()* function, but found that it was a blocking operation and caused the loop to slow down and for button presses to be missed. To get around this, we used the *millis()* function to determine the elapsed time, and then determined whether LEDs in the flashing/bouncing mode should be on based on the elapsed time.

For part two of the lab, we use the *analogRead* function (Line 55) to read the potentiometer, which has a maximum value of 1023, and scale that down to the maximum brightness of the LEDs, which is 255. We also shifted the LEDS down to the PWM pins to use the analogWrite function to pass in the brightness value of the potentiometer. This made the brightness of the LEDs change when the potentiometer was turned.

4 Reflection

In this lab we learned how to continuously perform both sensing and control with the same Arduino. A large challenge when coding the Arduino was that we wanted to continuously perform two operations: listen for button presses, and flash the lights according to the selected mode. Since everything happens in the main loop() function of the Arduino program, however, we had to first listen for button presses, and then flash the lights. This meant that we had to make sure that each loop completed quickly so that we would not miss a button press while the lights flashed. To do this, we had to deal with several timing challenges. This lab helped us learn about the Arduino language and several basic Arduino functions, like digitalRead, analogRead, analogWrite, and digitalWrite. It also helped to refresh our memory on basic aspects of building circuits as well.

5 Appendix

5.1 Source Code

```
/*
Lab 1: Bike Light
Dennis Chen & Jennifer Wei
 */

//Pin outs
const int GREEN = 3;
const int YELLOW = 5;
const int RED = 6;

//Pin ins
const int BUTTON = 8;
const int POT = 0;

//Modes
const int OFF = 0;
const int FLASHING = 1;
const int BOUNCING = 2;
const int ON = 3;

//Flash and bounce intervals in ms
const int FLASH_INT = 1000;
const int BOUNCE_INT = 1000;
const int BUTTON_DEBOUNCE_INT = 300;

const int MAXBRIGHTNESS = 255;
const int MAXANALOGREADVAL = 1023;

int state; // state stays off until button press
int loopCount;
int buttonPressedCount; // button state tracking to account for debouncing
int brightness; // where 255 is max LED brightness
boolean flashingOn; // used to track status of LEDs in flashing mode

// the setup function runs once when you press reset or power the board
void setup() {
  state = 0;
  loopCount = 0;
  buttonPressedCount = 0;
  brightness = 0;
  flashingOn = false;
  pinMode(BUTTON, INPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
```

```
  Serial.begin(9600);
}

// the loop function runs over and over again forever
void loop() {
  detectButtonPress();
  brightness = analogRead(POT) * 1.0 / MAXANALOGREADVAL * MAXBRIGHTNESS;
  if (state == FLASHING) {
    flash(loopCount);
  } else if (state == ON) {
    on();
  } else if (state == BOUNCING) {
    bounce(loopCount);
  } else if (state == OFF) {
    off();
  }
}

void detectButtonPress(){
  int buttonPressed = digitalRead(BUTTON);
  if (buttonPressed){
    buttonPressedCount ++;
  }
  else{
    buttonPressedCount = 0;
  }
  if (buttonPressedCount >= BUTTON_DEBOUNCE_INT){
    switchState();
    buttonPressedCount = 0;
  }
}

void switchState() { // goes off, flashing, on, bouncing, off
  if (state == FLASHING) {
    state = ON;
  } else if (state == ON) {
    state = BOUNCING;
  } else if (state == BOUNCING) {
    state = OFF;
  } else if (state == OFF) {
    state = FLASHING;
  }
}

void setAll(int val) {
  analogWrite(GREEN, val);
  analogWrite(YELLOW, val);
  analogWrite(RED, val);
}

```

```
  void off() {
97  setAll(LOW);
  }
99
  void on() {
101  setAll(brightness);
  }
103
  void bounce(int count) {
105  int timeModInterval = millis() % BOUNCE_INT*3;
    if (timeModInterval < BOUNCE_INT){
107    setAll(LOW);
      analogWrite(GREEN, brightness);
109  }
    else if (timeModInterval >= BOUNCE_INT && timeModInterval < BOUNCE_INT*2){
111    setAll(LOW);
      analogWrite(YELLOW, brightness);
113  }
    else if (timeModInterval >= BOUNCE_INT*2){
115    setAll(LOW);
      analogWrite(RED, brightness);
117  }
  }
119
  void flash(int count) {
121  int timeModInterval = millis() % BOUNCE_INT*2;
    if (timeModInterval < BOUNCE_INT){
123    setAll(brightness);
    } else {
125    setAll(LOW);
    }
127 }
  }
```