PIE Mini-Project 1: Bike Light

September 13, 2022 | Diana Garcia & Gia-Uyen Tran

Introduction

This objective of this mini-project was to build a bike light with the following main components:

- At least three 3 LEDs
- Current-limiting resistors for LEDs
- 1 Push button
- 1 Pull-down resistor for button (10kΩ)
- 1 Potentiometer

Part 1 called for a button to switch between a minimum of 5 modes in a system with at least 3 LEDs. Part 2 required the addition of some analog input that would alter the behavior of the circuit.

Procedure

For Part 1, we selected 4 LEDs with different colors: green, blue, red, and yellow. We chose 5 modes: all off, all on, all blinking at once, alternating blinking (i.e., bouncing), and blinking in binary up to 15.

We used Ohm's laws and respective LED datasheets to calculate the most optimal resistor values. Ohm's Law states that

$$V = IR$$

for some voltage V, current I, and resistance R. The Arduino supplies 5 V. Suitable currents were taken from respective LED datasheets. With the given voltage and current, we solved for the ideal resistor value. From there, we picked the resistors with the closest resistances. Table 1 shows the current, calculated ideal resistance, and selected resistor values for each LED.

LED color	Current (mA)	Calculated Ideal Resistance (Ω)	Selected Resistor (Ω)
Red	30	167	160
Yellow	20	250	200
Green	25	200	200
Blue	30	167	160

Table 1: Resistor Calculations and Selections.

After calculating for the resistor values, we assembled our circuit with only LEDs and resistors to verify our calculations. The red LED was connected to pin 6, the yellow to pin 5, the green to pin 11, and the blue to pin 10. Notice that these are all PWM pins. While this was not necessary for Part 1, they were selected in anticipation of Part 2. With a simple program to turn on all LEDs as shown in Figure 1, we verified that the selected resistor values resulted in appropriately high LED brightnesses.

```
// Connect components to pins
const int RED_LED = 6;
const int YELLOW_LED = 5;
const int GREEN_LED = 11;
const int BLUE_LED = 10;

void setup() {
  pinMode(BLUE_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
}

void loop() {
  // Turn all LEDs on
  digitalWrite(RED_LED, HIGH);
  digitalWrite(GREEN_LED, HIGH);
  digitalWrite(BLUE_LED, HIGH);
  digitalWrite(YELLOW_LED, HIGH);
}
```

Figure 1: Arduino program to check LED brightness.

Once resistors were selected, we wrote each of the five modes as separate functions and tested them individually. We then added the button to the circuit. At this point, we isolated the button to ensure that our program could detect when the button was pressed. To change the mode every time the button is pressed, we declared a variable named “mode”. The mode variable ranges from 0 to 4 for 5 total modes and is incremented by 1 every time the button is pressed (if mode is 4, it will loop back to 0). The value of mode was then printed to the Serial monitor for troubleshooting purposes. The test code is shown in Figure 2.

```
// Connect components to pin
const int BUTTON = 8;

// intialize counting variables
int mode = 0;
int button_state = 0;
int previous_button = 0;
int button_count = 0;

void setup() {
  Serial.begin(9600);
  pinMode(BUTTON, INPUT);
```

```
}

void loop() {

  // print current mode to serial monitor
  Serial.print("mode:");
  Serial.println(mode);

  // Change mode when button is pressed
  button_state = digitalRead(BUTTON);

  if (button_state != previous_button) {
    if (button_state == HIGH) {
      button_count++;
      mode = button_count % 5;
    }
  }
  previous_button = button_state;
}
```

Figure 2: Arduino program to check button pushes.

The final step in Part 1 was to integrate the push button with the LEDs. We accomplished this with a simple set of if-else statements. For each possible value of the mode variable, one of five mode functions was called.

Changing between modes with the push button was the most difficult stage in our process, largely stemming from issues in using the delay() function to control timing. While this approach worked when testing individual functions, it was less effective when integrating the entire system. If a button was pushed during a delay, it was unable to be detected. To fix the issue, we had to rewrite all functions that relied on delay() to instead use time relative to a universal timing system (at least in the context of this system) with the millis() function.

We then proceeded to Part 2, which requires an analog input that alters the behavior of the bike light. We chose to change the brightness of the LEDs based on analog input from a potentiometer. We connected the potentiometer to the A1 pin and added a few lines of code to map the range of possible potentiometer readings (0 to 1023) to the LED brightness (0 to 255). Notice that the LED brightness range requires the PWM pins to function. We tested this mapping with the program shown in Figure 3.

```
// Connect components to pins
const int RED_LED = 6;
const int YELLOW_LED = 5;
const int GREEN_LED = 11;
const int BLUE_LED = 10;
const int POT = A1;

void setup() {
  pinMode(BLUE_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
```

```
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(POT, INPUT);
}

void loop() {

  // Map potentiometer reading to LED brightness
  int POTValue = analogRead(POT);
  int brightness = map(POTValue, 0, 1023, 0, 255);

  // Turn all LEDs on
  analogWrite(RED_LED, brightness);
  analogWrite(GREEN_LED, brightness);
  analogWrite(BLUE_LED, brightness);
  analogWrite(YELLOW_LED, brightness);

}
```

Figure 3: Arduino program to check mapping potentiometer reading to LED brightness.

After incrementally integrating our components, our systems finally looked like Figure 4. The final program as shown in the appendix only differed from the Part 1 code in that it added the potentiometer and mapped its input to a range of LED brightnesses.

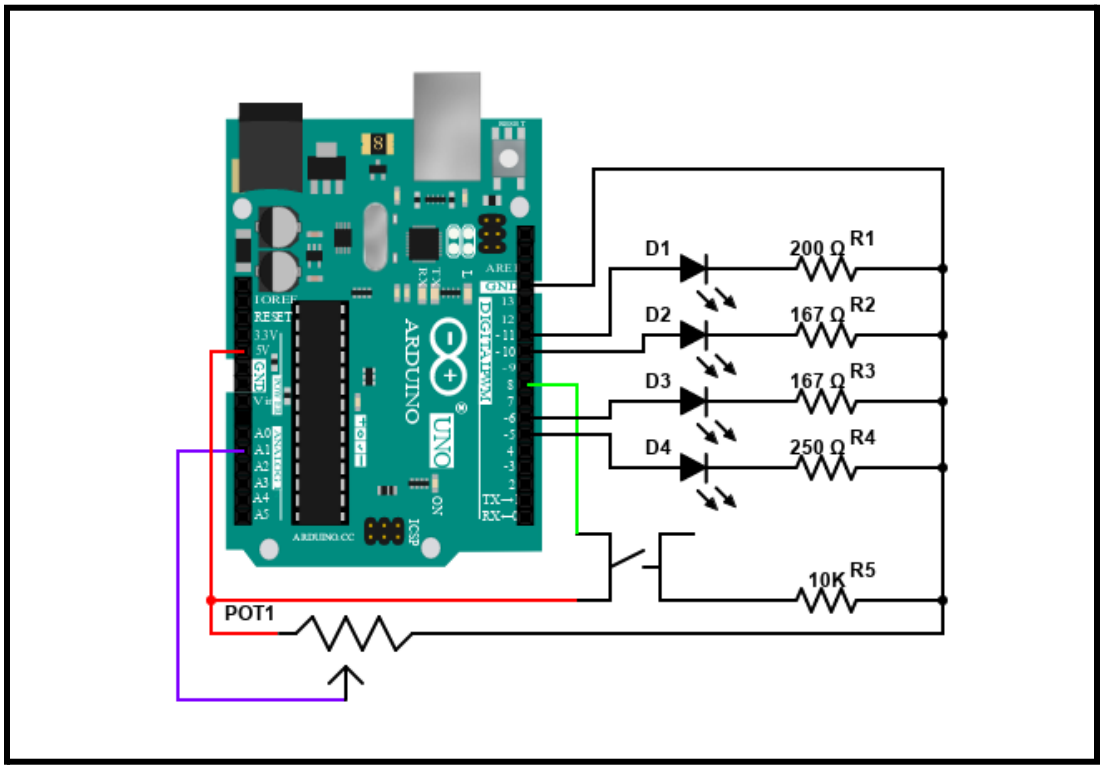

Figure 4: Electric schematic of our circuit.

Reflection

Individual

Diana: I really enjoyed this lab. I enjoyed being able to apply my knowledge to calculate the values of the resistors, which I had learned in the last lab. I am glad I was able to write code for the potentiometer. I have not had much experience programming with Arduino, but with this project I was able to learn about analog mapping and writing.

Uyen: I had very little prior experience with Arduino – certainly not enough that I knew how to intuitively approach the problem. I now have a much better understanding of how the Arduino works in general. It was also worthwhile to review the syntax and purpose of each part of a standard Arduino program. I worked mostly on the code, but I wish that I spend a little more time with the electrical aspects of this mini-project.

Teaming

While we both know the rationale behind each decision and could independently produce the same results, we had a distinct division of tasks. Diana primarily worked on the electrical side by designing the circuit and producing the schematic. Uyen primarily worked on the programming side by writing the key code. Both provided support to the other for minor tasks such as circuit troubleshooting and code debugging. While this compartmentalization was useful for efficiently finishing the mini-project, it wasn't that helpful for learning. Both team members naturally gravitated towards the tasks they were more comfortable with and missed out on learning about the things they had less experience with.

Appendix

```
/*
*
* PIE Mini-Project 1: Bike Light
* Diana Garcia & Gia-Uyen Tran
* Submitted September 13, 2022
*
* As required by the assignment guidelines, this program switches
* between a group of 4 LEDs between 5 modes (see functions below)
* when a push button is pressed. The brightness of the LEDs can
* be adjusted with a potentiometer.
*
* Functions:
* all_off() - turns all LEDs off
* all_on() - turns all LEDs on
* all_blinking() - blinks all LEDs on and off in unison
* bouncing() - alternates LEDs in consecutive order (by location),
*    then reverse direction
* binary() - blinks binary numbers up to 15 as if each LED were a
*    bit in a binary number
*
* Hardware Connections:
* Yellow LED - Pin 5
* Red LED - Pin 6
* Button - Pin 8
* Blue LED - Pin 10
* Green LED - Pin 11
* Potentiometer - Pin A1
*
*/


// Connect components to pins
const int RED_LED = 6;
const int YELLOW_LED = 5;
const int GREEN_LED = 11;
const int BLUE_LED = 10;
const int POT = A1;
const int BUTTON = 8;

// intialize counting variables
int mode = 0;
int counter = 0;
int button_state = 0;
int previous_button = 0;
int button_count = 0;
int ledstate = 0;

// Global variable to store the time that LED last changed state
uint32_t blink_time;
```

```
void setup() {

  pinMode(BLUE_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(BUTTON, INPUT);
  pinMode(POT, INPUT);

  // Turn LEDs off initially
  digitalWrite(BLUE_LED, LOW);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(RED_LED, LOW);

  // Remember the current value of the millis timer
  blink_time = millis();
}


void loop() {

  // Map potentiometer reading to LED brightness
  int POTValue = analogRead(POT);
  int brightness = map(POTValue, 0, 1023, 0, 255);

  // Change mode when button is pressed
  button_state = digitalRead(BUTTON);

  if (button_state != previous_button) {
    if (button_state == HIGH) {
      button_count++;
      mode = button_count % 5;
    }
  }
  previous_button = button_state;

  if (mode == 0) {
      all_off();
    } else if (mode == 1) {
      all_on(brightness);
    } else if (mode == 2) {
      all_blinking(brightness);
    } else if (mode == 3) {
      bouncing(brightness);
    } else if (mode == 4) {
      binary(brightness);
    }

 delay(50);

}
```

```
// ------------- FUNCTIONS ------------- //


void all_on(int brightness) {

  // Turn all LEDs on
  analogWrite(RED_LED, brightness);
  analogWrite(GREEN_LED, brightness);
  analogWrite(BLUE_LED, brightness);
  analogWrite(YELLOW_LED, brightness);

}


void all_off() {

  // Turn all LEDs off
  digitalWrite(RED_LED,LOW);
  digitalWrite(YELLOW_LED,LOW);
  digitalWrite(BLUE_LED,LOW);
  digitalWrite(GREEN_LED,LOW);

}


void all_blinking(int brightness) {

  uint32_t t;
  t = millis();

  int speed = 250;

  if (t >= blink_time + speed) {
    // Switch states (1: on, 0: off)
    if (ledstate == 1) {
      ledstate = 0;
    } else {
      ledstate = 1;
    }

    if (ledstate == 1) {

      // Turn on all LEDs
      analogWrite(RED_LED, brightness);
      analogWrite(YELLOW_LED, brightness);
      analogWrite(GREEN_LED, brightness);
      analogWrite(BLUE_LED, brightness);

    } else {

      // Turn off all LEDs
```

```
      analogWrite(RED_LED, 0);
      analogWrite(YELLOW_LED, 0);
      analogWrite(GREEN_LED, 0);
      analogWrite(BLUE_LED, 0);

    }
    blink_time = t;
  }

}


void bouncing(int brightness) {

  int32_t t;
  t = millis();

  int speed = 250;

  if (t >= blink_time + speed) {

    counter++;
    counter = counter % 6;

    if (counter == 0) {

      // Turn on only red light
      analogWrite(RED_LED, brightness);
      analogWrite(YELLOW_LED, 0);
      analogWrite(GREEN_LED, 0);
      analogWrite(BLUE_LED, 0);

    } else if (counter == 1 || counter == 5) {

      // Turn on only yellow light
      analogWrite(RED_LED, 0);
      analogWrite(YELLOW_LED, brightness);
      analogWrite(GREEN_LED, 0);
      analogWrite(BLUE_LED, 0);

    } else if (counter == 2 || counter == 4) {

      // Turn on only green light
      analogWrite(RED_LED, 0);
      analogWrite(YELLOW_LED, 0);
      analogWrite(GREEN_LED, brightness);
      analogWrite(BLUE_LED, 0);

    } else if (counter == 3) {

      // Turn on only blue light
      analogWrite(RED_LED, 0);
      analogWrite(YELLOW_LED, 0);
```

```
      analogWrite(GREEN_LED, 0);
      analogWrite(BLUE_LED, brightness);

    }

    blink_time = t;

   }
}


void binary(int brightness) {

  int32_t t;
  t = millis();

  int speed = 250;

  all_off();

  if (t >= blink_time + speed) {

    if (counter <= 15){

      counter++;

      if((counter % 2) > 0) {analogWrite(RED_LED, brightness);
        } else { digitalWrite(RED_LED, LOW); }
      if((counter % 4) > 1) { analogWrite(YELLOW_LED, brightness);
        } else { digitalWrite(YELLOW_LED, LOW); }
      if((counter % 8) > 3) { analogWrite(GREEN_LED, brightness);
        } else { digitalWrite(GREEN_LED, LOW); }
      if((counter % 16) > 7) { analogWrite(BLUE_LED, brightness);
        } else { digitalWrite(BLUE_LED, LOW); }

    } else {
      counter = 0;
    }

    blink_time = t;
  }

}
```