Mini-Project 1 Lab Report
Section 4, Simrun and Melody

Narrative Description

We started off this project by using the schematic provided and building up the circuit. We chose values of resistors based on the LED data sheet. The second step was to verify that the circuit worked. We did this by turning all of the LEDs on and off using the Blink example code provided by Arduino IDE. Next, we figured out how to change the state of the LEDs using a button. We struggled with changing the state based on when the button is pushed. It was tricky to tell when the button was initially pushed. We had to try out a few different ideas before figuring out how to do this smoothly.

The next step was to decide on all the different bike light states that we were going to use. We decided on doing on, off, blink, bounce and quick blink. We wrote separate functions for each of these modes and then called them in our main loop using a switch case. We encountered some difficulties in timing. Initially we were using the delay function to create a pause between when the LED was off and on. However, we realised that using delay stops the entire loop and so we switched to using millis() in order to control the timing for the LEDs better.

The last step was to integrate a potentiometer into our circuit. Initially we tried to connect it to the LEDs itself but this was causing the LEDs to become very dim. We then decided to just let the potentiometer have its own circuit and connect it to the Arduino separately. We wrote a simple function that changes the speed at which the LEDs blink based on the resistance value from the pot and used this to change the blink speed of the LEDs.

Circuit Description

For this circuit schematic, we chose values of resistors based on the red LED data sheet. We calculated that the red LED should have a resistance of (5V-2V)/10mA = 300 ohms.

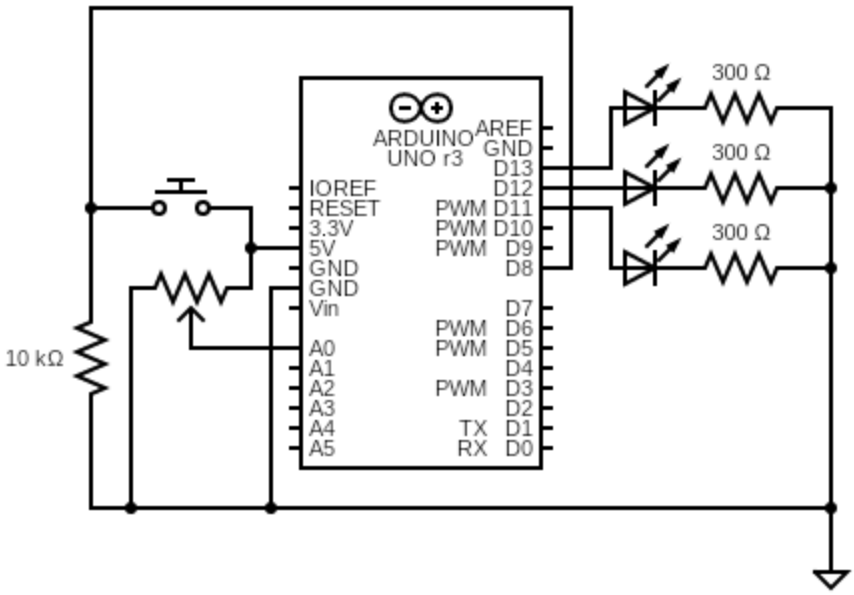

Fig 1. Circuit schematic

Source Code

We have the logic for detecting when the button is pressed in our void loop. As shown in Fig. 2, we have a variable buttonWasLow that is set to true if the button is not currently being pressed. We detect that the button is ‘pressed’ if a high signal is received from the button pin (i.e. the button is currently being pressed) and also if it was not previously pressed (i.e. buttonWasLow is true). If, by this logic, the button is pressed, the system of LEDs switches to the next state which we keep track of using the ledState variable.

```
117  void loop() {
118    // read the state of the pushbutton value:
119    buttonState = digitalRead(buttonPin);
120    // if the button is currently not pushed, say that button was low is true
121    if (buttonState == LOW)
122    buttonWasLow = true;
123    // check if the pushbutton is pressed and that it was previously not pushed
124    if (buttonState == HIGH && buttonWasLow) {
125      // change the state of the bike light
126      ledState = ledState + 1;
127      // if it goes through all the states, then go back to the first one
128      if (ledState > 6){
129        ledState = 1;
130      }
131      // set the button was low value to false
132      buttonWasLow = false;
133    }
134    // run the correct bike light mode based on the LED state
135    switch (ledState){
136      case 1:
137      offMode();
138      break;
139      case 2:
140      onMode();
141      break;
142      case 3:
143      blinkMode();
144      break;
145      case 4:
146      bounceMode();
147      break;
148      case 5:
149      quickBlinkMode();
150      break;
151      case 6:
152      pot();
153      break;
154    }
155  }
```

Fig 2. Void loop

There are a few different bike light states including off, on, blink, bounce and quick blink that are controlled with switch cases. As we wrote each of our LED state functions, we added to the list of global variables at the top of our program, shown in Fig. 3. This is also where we wrote our setup function that sets up the LEDs and button pins.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 8;     // the number of the pushbutton pin
const int ledPin1 =  11;     // the number of the LED pin
const int ledPin2 =  12;     // the number of the LED pin
const int ledPin3 =  13;     // the number of the LED pin
const int potPin = A0;
const uint16_t blink_interval = 100;
const uint16_t bounce_interval = 500;
const uint16_t quick_blink_interval = 50;

// variables will change:
int buttonState = 1;         // variable for reading the pushbutton status
int ledState = 1;            // variable for changing the bike light mode
int potValue = 0;            // potentiometer sensor reading
bool buttonPress = false;    // if the button is currently pressed or not
bool buttonWasLow = true;    // if the button was previously pressed
uint16_t blink_time;         // current time
uint16_t bounce_time;        // current time
int bounce_state = 1;        // integer representing which LED is flashing
uint16_t pot_interval;       // amount of time between blinks when the pot is an input

void setup() {
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // initialize blink and bounce time as time
  blink_time = millis();
  bounce_time = millis();
}
```

Fig 3. Code block in which variables are declared and pins are set-up.

Firstly, we have a state called offMode where all the LEDs are off and a state called onMode where all the LEDs are on. These functions simply turn each LED off or on by sending a low or high voltage to that pin, respectively (Fig. 4).

```
// function to turn all the LEDs off
void offMode() {
  digitalWrite(ledPin1, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(ledPin2, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(ledPin3, LOW);    // turn the LED off by making the voltage LOW
}

// function to turn all the LED's on
void onMode() {
  digitalWrite(ledPin1, HIGH);    // turn the LED off by making the voltage HIGH
  digitalWrite(ledPin2, HIGH);    // turn the LED off by making the voltage HIGH
  digitalWrite(ledPin3, HIGH);    // turn the LED off by making the voltage HIGH
}
```

Fig 4. Code block showing two mode functions: All LEDs are off and on.

Next we have the blinkMode state where the LEDs blink constantly. We also have the function quickBlinkMode which again has all the LEDs blink together, but in faster intervals. We used the built-in Arduino function millis() to have a variable *t* that is the number of milliseconds that have passed since the Arduino board began running the program. In our function we have a conditional statement that checks if t is greater than the sum of blink_time and blink_interval. Blink_interval is the amount of time we want the LEDs to pause before switching and blink_time is the time at which the LEDs last switched. If this condition is met, this means that enough time has passed since the LEDs last switched so that they should now switch again. The code in the if statement says that each LED should turn on if it was previously off or turn off if it was previously on. Blink_time is also set to t, the current time, as the most recent switch time.

```
49  // function to blink the LEDs
50  void blinkMode() {
51    uint16_t t;
52    t = millis();
53    if (t > blink_time + blink_interval){
54      digitalWrite(ledPin1, !digitalRead(ledPin1));
55      digitalWrite(ledPin2, !digitalRead(ledPin2));
56      digitalWrite(ledPin3, !digitalRead(ledPin3));
57      blink_time = t;
58    }
59  }
60
61  // function to quickly blink all the LEDs
62  void quickBlinkMode() {
63    uint16_t t;
64    t = millis();
65    if (t > blink_time + quick_blink_interval)
66    {
67      digitalWrite(ledPin1, !digitalRead(ledPin1));
68      digitalWrite(ledPin2, !digitalRead(ledPin2));
69      digitalWrite(ledPin3, !digitalRead(ledPin3));
70      blink_time = t;
71    }
72  }
```

Fig 5. Blinking state functions

Next, we have the bounceMode function which controls the LEDs to light up one-by-one. We achieve this by having three bounce states: each state being a different LED turned on while others are off. The logic for this function is similar to the previous blinking functions, but instead of switching each LED, we switch the bounce state. So, if we are currently on bounce state 1 where only the first LED is turned on, at the next

interval, we switch to bounce state 2 where only the second LED is turned on. If the bounce state is 3, the next switch should go back to bounce state 1.

```
74  // function making the LEDs light up one at a time
75  void bounceMode() {
76    uint16_t t;
77    t = millis();
78    if (t > bounce_time + bounce_interval)
79    {
80      bounce_state = bounce_state + 1;
81      if (bounce_state > 3)
82        bounce_state = 1;
83      bounce_time = t;
84    }
85    if (bounce_state == 1){
86      digitalWrite(ledPin1, HIGH);
87      digitalWrite(ledPin2, LOW);
88      digitalWrite(ledPin3, LOW);
89    }
90    if (bounce_state == 2){
91      digitalWrite(ledPin1, LOW);
92      digitalWrite(ledPin2, HIGH);
93      digitalWrite(ledPin3, LOW);
94    }
95    if (bounce_state == 3){
96      digitalWrite(ledPin1, LOW);
97      digitalWrite(ledPin2, LOW);
98      digitalWrite(ledPin3, HIGH);
99    }
100 }
```

Fig 6. Bounce state function

Lastly, we have a function that controls the LEDs to blink at varying times based on the resistance of a potentiometer. This function is similar to the previous blinking functions, but instead of a constant blink-interval variable determining how long the LEDs pause before switching, this interval time is based on the sensor value read from the potentiometer.

```
102  // function to make the LEDs blink at a speed based on the pot
103  void pot() { // min - max -> 0 - 623
104    float sensorValue = analogRead(potPin);
105    pot_interval = sensorValue + 200;
106    uint16_t t;
107    t = millis();
108    if (t > blink_time + pot_interval)
109    {
110      digitalWrite(ledPin1, !digitalRead(ledPin1));
111      digitalWrite(ledPin2, !digitalRead(ledPin2));
112      digitalWrite(ledPin3, !digitalRead(ledPin3));
113      blink_time = t;
114    }
115  }
```

Fig 7. Variable speed blinking function

Reflection

Simrun

One of my main learnings from this lab was understanding the way arduino code differed from writing code in another language. Because arduino code continuously loops, the format for everything was a little different and it was a challenge to understand how timing works. I think building the circuit itself was fairly straightforward and I did not have too many struggles with that.

Melody

Similar to Simrun, the biggest thing I gained from this mini-lab was the practical knowledge of how to use an Arduino. We had the chance to get acquainted with Arduino functions like millis() and Series.print() which will definitely be useful in future projects. I also found it very fun to work with a breadboard, with how easily we could move things around and add a potentiometer to the circuit.