# Section 2, Mini Project 1, Kenta, Mihir and Zayn

## Introduction and Reflection

The goal of this project was to use the Arduino microcontroller and IDE, 3-4 LEDs, and a potentiometer to build our own bike light. We wrote five different functions corresponding to five different lighting modes.

In terms of things that went well, we were able to construct the circuit in twenty minutes on the first day. We used our knowledge from a previous course to calculate the resistor values, using current values from the LED datasheets. We could've improved our initial code design, we avoided using Arduino's `millis()` function which resulted in us writing more complex and buggy code using `delay(500)` to switch between states.

## Source Code and Arduino Function Descriptions

Demo Video

**blink() Function:**
This function toggles all LEDs on and off simultaneously at a regular interval defined by the period variable. It checks if enough time has passed since the last state change by comparing the current time (in milliseconds) with pastTime. If the time interval specified by period has elapsed, it toggles the state (ON to OFF or OFF to ON) of all four LEDs. It updates the pastTime variable to the current time to mark the time of the last state change.
**Result:** This function creates a blinking effect where all LEDs blink on and off together.

**swap() Function:**
This function alternates between two pairs of LEDs, creating a swapping effect between the pairs. It uses the firstCounter variable to determine which pair of LEDs to light up.
If firstCounter is even, it turns on the first pair of LEDs (LEDs 0 and 1) and turns off the second pair (LEDs 2 and 3). If firstCounter is odd, it does the opposite, turning off the first pair and turning on the second pair. It checks if enough time has passed since the last state change based on the period variable and then updates pastTime to the current time after making a state change. Finally, the firstCounter variable is updated to alternate between even and odd values, causing the LEDs to swap each time this function is called.
**Result:** This function creates a swapping effect between two pairs of LEDs.

**on() Function:**
This function turns on all four LEDs by setting their pins to HIGH. The light stays constant until the button is pressed, triggering another state.
**Result:** This function lights up all LEDs simultaneously.

**randomled() Function:**
This function randomly selects one LED and toggles its state at a regular interval defined by the period variable. It generates a random number between 0 and numLeds - 1 to select one of the

four LEDs randomly and toggles the state (ON to OFF or OFF to ON) of the randomly selected LED. It then checks if enough time has passed since the last state change based on the period variable and updates pastTime to the current time after making a state change.
**Result:** This function creates a random blinking effect, where one LED blinks randomly while the others stay constant.

**snake() Function:**
This function creates a "snake" effect by toggling LEDs one by one with a certain time delay (period) between each LED. It checks if enough time has passed since the last state change for each LED and if enough time has passed, it toggles the state of the LEDs sequentially, starting from LED 0 and moving to LED 1. Finally, it updates pastTime to the current time after each state change. This function effectively creates a wave-like animation where LEDs turn on and off in sequence.
**Result:** This function produces a "snake" animation where LEDs light up one after another in a continuous loop.

We used the pretty printer software to convert the following code from our Arduino sketch.

```
// Constants for controlling LED behavior
const uint16_t period = 400; // Time interval between toggling LED in
milliseconds
const int numLeds = 4;
int ledPins[numLeds] = {4, 5, 6, 7};
uint32_t pastTime; // Global variable to store the time that LED last changed
state
int firstCounter = 0;
int potPin = A0; //Initializes Potentiometer pin
const int buttonPin = 3;  // the number of the pushbutton pin
int buttonState = 0;
int modeCounter = 0;

// Function to check if it's time to change LED state
bool it_is_time(uint32_t t, uint32_t t0, uint16_t dt) {
    return ((t >= t0) && (t - t0 >= dt)) ||         // The first disjunct
handles the normal case
    ((t < t0) && (t + (~t0) + 1 >= dt));
}


void setup() {
    // Set LED pins as OUTPUT
    Serial.begin(9600);
    for (int i = 0; i < numLeds; i++) {
        pinMode(ledPins[i], OUTPUT);
    }
    pinMode(buttonPin, INPUT);
    pastTime = millis();
}
```

```
void loop() {
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH) {
        modeCounter = modeCounter + 1;
        if (modeCounter >= 5){
            modeCounter = 0;
        }
        Serial.println(modeCounter);
    }
    else {
        Serial.print(modeCounter);
        if (modeCounter == 0){
            blink();
        }
        if (modeCounter == 1){
            swap();
        }
        if (modeCounter == 2){
            on();
        }
        if (modeCounter == 3){
            randomled();
        }
        if (modeCounter == 4){
            bounce();
        }

    }

}


// Function to create a blinking effect for all LEDs
void blink() {
    int potValue = analogRead(potPin);
    // Map the potentiometer value (0-1023) to the desired range of period
(e.g., 100-2000)
    uint16_t mappedPeriod = map(potValue, 0, 1023, 100, 2000);
    // Update the period based on the potentiometer value
    int period = mappedPeriod;
    // Local variable to store the current value of the millis timer
    uint32_t t;
    // Get the current value of the millis timer
    t = millis();
    // If BLINK_INTERVAL milliseconds have elapsed since blink_time, Toggle the
state of all LEDs
    if (it_is_time(t, pastTime, period)) {
        digitalWrite(ledPins[0], !digitalRead(ledPins[0]));
        digitalWrite(ledPins[1], !digitalRead(ledPins[1]));
```

```
        digitalWrite(ledPins[2], !digitalRead(ledPins[2]));
        digitalWrite(ledPins[3], !digitalRead(ledPins[3]));
        // Update the time of the last state change
        pastTime = t;
    }
}

// Function to create a swapping effect between LED pairs
void swap() {
    int potValue = analogRead(potPin);
    // Map the potentiometer value (0-1023) to the desired range of period
(e.g., 100-2000)
    uint16_t mappedPeriod = map(potValue, 0, 1023, 100, 2000);
    // Update the period based on the potentiometer value
    int period = mappedPeriod;
    uint32_t t; // Local variable to store the current value of the millis
timer
    t = millis();
    if (it_is_time(t, pastTime, period)) { // If BLINK_INTERVAL milliseconds
have elapsed since blink_time,
        if (firstCounter % 2 == 0) { // Turn on the first pair of LEDs and turn
off the second pair
            digitalWrite(ledPins[0], HIGH);
            digitalWrite(ledPins[1], HIGH);
            digitalWrite(ledPins[2], LOW);
            digitalWrite(ledPins[3], LOW);
        }
        // Turn off the first pair of LEDs and turn on the second pair
        else {
            digitalWrite(ledPins[0], LOW);
            digitalWrite(ledPins[1], LOW);
            digitalWrite(ledPins[2], HIGH);
            digitalWrite(ledPins[3], HIGH);
        }
        // Increment the counter and update the time of the last state change
        firstCounter = firstCounter + 1;
        pastTime = t;
    }
}

// Function to turn on all LEDs
void on() {
    digitalWrite(ledPins[0], HIGH);
    digitalWrite(ledPins[1], HIGH);
    digitalWrite(ledPins[2], HIGH);
    digitalWrite(ledPins[3], HIGH);
}
```

```
// Function to create a random blinking effect for one LED
void randomled() {
    int potValue = analogRead(potPin);
    // Map the potentiometer value (0-1023) to the desired range of period
(e.g., 100-2000)
    uint16_t mappedPeriod = map(potValue, 0, 1023, 100, 2000);
    // Update the period based on the potentiometer value
    int period = mappedPeriod;
    uint32_t t;
    t = millis();
    int randomLED = random(numLeds);
    if (it_is_time(t, pastTime, period)) {
        // Toggle the state of a randomly selected LED
        digitalWrite(ledPins[randomLED], !digitalRead(ledPins[randomLED]));
        // Update the time of the last state change
        pastTime = t;
    }
}

// Function to create a "snake" effect
void bounce() {
    int potValue = analogRead(potPin);
    // Map the potentiometer value (0-1023) to the desired range of period
(e.g., 100-2000)
    uint16_t mappedPeriod = map(potValue, 0, 1023, 100, 2000);
    // Update the period based on the potentiometer value
    int period = mappedPeriod;
    for (int i = 0; i < numLeds; i++) {
        digitalWrite(ledPins[i], HIGH);
        // while(millis() < time_now + period){
        // }
        delay(period);
        digitalWrite(ledPins[i], LOW);
    }
    for (int i = numLeds - 1; i >= 0; i--) {
        digitalWrite(ledPins[i], HIGH);
        // while(millis() < time_now + period){
        // }
        delay(period);
        digitalWrite(ledPins[i], LOW);
    }
}
```

## Circuit Diagram [Figure 1]

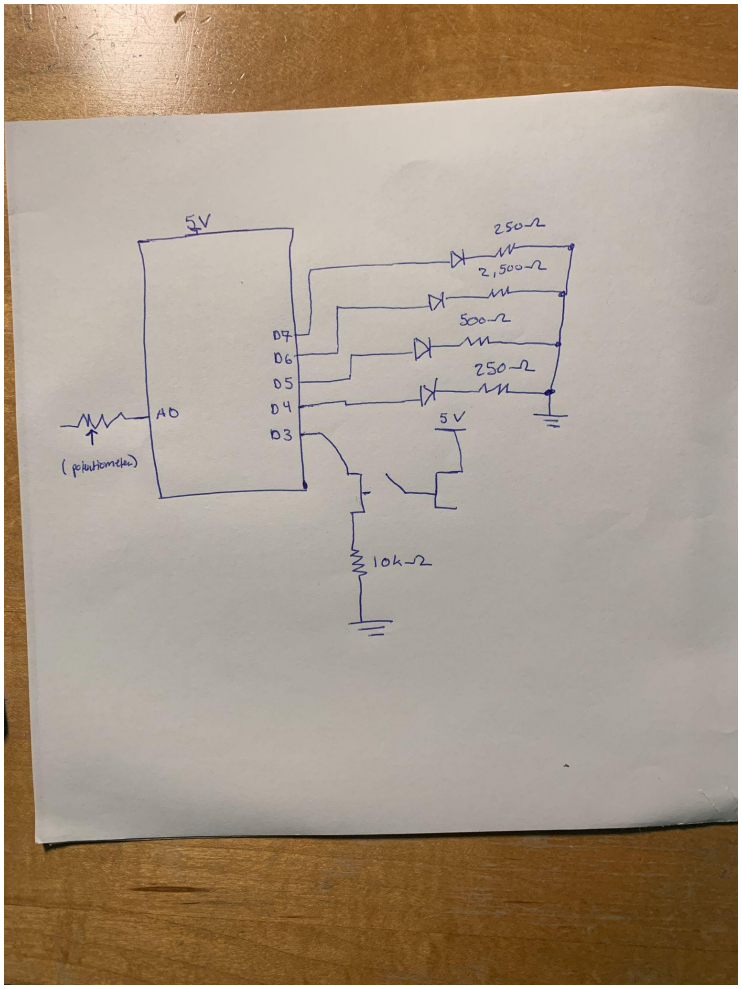


Figure 1 shares the circuit diagram we used for mini project #1. We calculated the resistor values by using LED specific datasheets from digikey (example). We looked at the “Current - Test” parameter on these datasheets as our I and 5V as our V. This resulted in the following math:

**Blue and Red resistor:**

5 V / 0.02 A = 250 Ω

**Yellow resistor:**

5 V / 0.010 A = 500 Ω

**Green resistor:**

5 V / 0.002 = 2,500 Ω

As for the other elements of the circuit, we chose a potentiometer since we had experience with it and didn’t have a specific reason for choosing any of the digital pins.