Principles of Integrated Engineering (PIE)

Mini-Project 1 - Section 2: Group 9

Sparsh Gupta, Mark Belanger, Noah Rand

Franklin W. Olin College of Engineering

October 15, 2023

Introduction

The purpose of this project is to develop a deeper understanding of Arduino and basic circuitry. We used an Arduino UNO R4/R3 and a series of LEDs, a switch, and a potentiometer to create a bike light with five different operational modes.

Implementation and Functionality

LED Resistor Selection
The LEDs have a max current load that they can endure so we have to make sure that the current through the LED does not exceed this maximum. The data sheets indicate that the LED's load cannot exceed 25 mA and the I/O pins we use are 5V so in order for the current to be at about 25 mA we need a 200Ω resistor. This was derived because $V = IR$ and $I = V/R$ so $5V/200\Omega = 25mA$.

The Potentiometer and Analog Input
The potentiometer output is connected to the analog input pin A0 of the Arduino. The 'pot_value' variable records the voltage reading from the potentiometer output which ranges from 0 to 1023 (assuming a 10-bit analog-to-digital converter on the microcontroller). The map() function is used to convert the analog input value from the potentiometer to the delay value which ranges from 50ms to 500ms. The delay value can be adjusted by rotating the potentiometer wiper which modifies the speed of the modes for the LEDs.

Push Button Debounce
When a button is pressed, it often bounces a few times before settling in the down or on position. This is problematic because microcontrollers like Arduino are often fast enough to process these bounces causing one button press to act like two or three. To account for this problem, we implemented a debounce time of 100ms into the code. Debounce works by effectively increasing the time between taking readings of a button press, eliminating false toggles caused by bouncing.

LED Modes
The LEDs cycle between five different modes that is achieved using switch-case statements in the code to write digital output signals to the Arduino:

1. All LEDs off: This is achieved by configuring all the LEDs to have a LOW output state (0V).
2. All LEDs flashing: If the current time modulo 1000ms is less than 500ms, the LEDs are set to HIGH, otherwise, they are set to LOW. This creates a blinking effect with a 50% duty cycle.
3. All LEDs on: This is achieved by configuring all the LEDs to have a HIGH output state (5V).
4. Bouncing lights: In this mode, the LEDs appear like they are bouncing because only one LED is set to HIGH state whereas the rest of the LEDs have a LOW output state. The 'bounce_led' variable keeps track of which LED is set to HIGH voltage.

5. Random lights: This mode randomly selects an LED to light up. The 'rand_led' variable chooses a random value between 0 to 4 and sets that LED to HIGH output state. This mode gives the effect that random LEDs are lighting up.

Switch Button
The switch button (SW1) is connected to the circuit to cycle through the modes when it is pressed and the state of this switch is tracked in the program.

Circuit Diagram

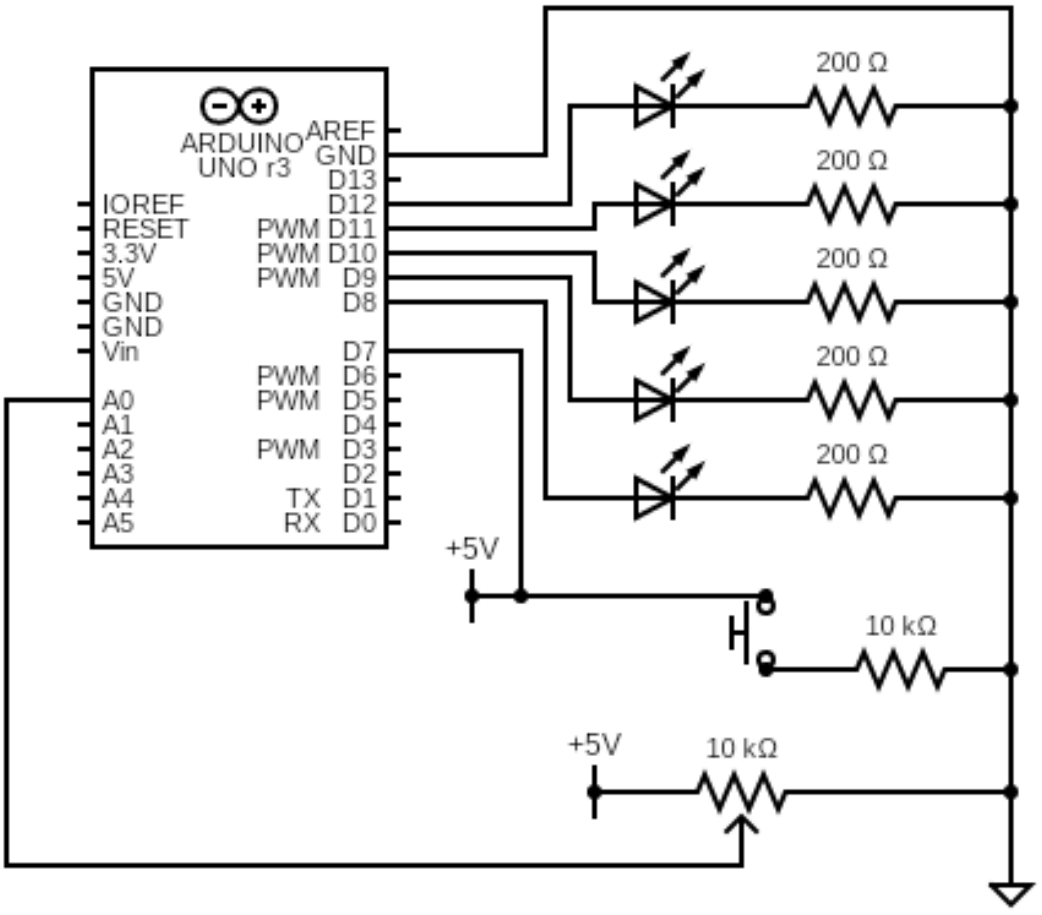

Figure 1: Bike Light Circuit Diagram

Code

```
const int LED[] = {8, 9, 10, 11, 12};      // LEDs connected to pins
const uint8_t SW1 = 7;                     // SW1 connected to D7
const uint8_t POT = 0;                     // POT wiper connected to
   A0

const uint16_t DEBOUNCE_INTERVAL = 100;    // Set the debounce
   interval

uint8_t switch1_state;            // Global var to store the state of
    SW1
uint8_t mode = 0;                 // Global var to store the mode
int8_t bounce_led = 0;            // Global var to store the current
   active LED for bouncing lights
int8_t rand_led = 0;              // Global var to store the current
   active LED for random lights
int pot_value = 0;                // Global var to store the
   potentiometer value
uint32_t last_debounce_time = 0; // Global var to store the last
   debounce time

void setup() {
  for (int i = 0; i < 5; i++){
    pinMode(LED[i], OUTPUT);              // Setting up LEDs as
       outputs
  }
  pinMode(SW1, INPUT);                    // Setting up SW1 as input

  switch1_state = digitalRead(SW1);
}

void loop() {
  uint32_t current_time = millis();       // Record the time
  uint8_t SW1_state = digitalRead(SW1);   // Record the state of SW1

  if (SW1_state != switch1_state && (current_time -
     last_debounce_time) > DEBOUNCE_INTERVAL) {
    if (SW1_state == LOW) {
      mode = (mode + 1) % 5;            // Cycle through the 5 modes
    }
    switch1_state = SW1_state;
    last_debounce_time = current_time;
  }

```

```
35   // Read potentiometer value
36   pot_value = analogRead(POT);
37
38   // Five different modes
39   switch (mode) {
40     case 0:  // All LEDs off
41       for (int i = 0; i < 5; i++) {
42         digitalWrite(LED[i], LOW);
43       }
44       break;
45
46     case 1:  // All LEDs flashing
47       for (int i = 0; i < 5; i++) {
48         digitalWrite(LED[i], current_time % 1000 < 500 ? HIGH : LOW)
             ;
49       }
50       break;
51
52     case 2:  // All LEDs on
53       for (int i = 0; i < 5; i++) {
54         digitalWrite(LED[i], HIGH);
55       }
56       break;
57
58     case 3:  // Bouncing lights
59       for (int i = 0; i < 5; i++) {
60         digitalWrite(LED[i], i == bounce_led ? HIGH : LOW);
61       }
62       bounce_led = (bounce_led + 1) % 5;
63       break;
64
65     case 4:  // Random lights
66       for (int i = 0; i < 5; i++) {
67         digitalWrite(LED[i], i == rand_led ? HIGH : LOW);
68       }
69       rand_led = random(5);
70       break;
71   }
72
73   // Adjust the speed of the modes using potentiometer
74   delay(map(pot_value, 0, 1023, 50, 500));  // Set up delay values
       from 50ms to 500ms
75 }
```

Reflection

Teaming

Overall, we did a good job of managing workload, debugging code, teaching and learning from each other, and having fun while we worked. For example, one us took the initiative of doing a significant number of Arduino tutorials and practice problems before the rest of the group began meeting. That allowed him to significantly help the group as we were troubleshooting problems. However, I think we could have done a better job of communication. At the beginning of the project, we planed on not working on it before the first class however, some people worked ahead and completed the project before others knew it started. To overcome this challenge, we decided to each build and program the circuit individually. This worked well as we were able to help each other troubleshoot and had more individual practice and learning.

Technical Implementation

As we each spent time working on the circuit individually, we originally had multiple different methods for implementing our light's state. Noah originally tried to write functions representing each state of the bike light, with the goal of toggling between them in the loop based on the most recent button press duration. However, due to his unfamiliarity with Arduino/C++, he ran into significant hurdles and ended up switching to a method more inline with Brad's videos. Mark's implementation was similar to Sparsh's and was inspired by Brad's video on using states however was not as efficient as Sparsh's. Sparsh's method focused on cycling through different modes using switch-case statements. Sparsh's method tracked which mode to choose by incrementing the value of the 'mode' variable after every button press. In the end, we settled on Sparsh's method above as it was the most efficient and elegant.

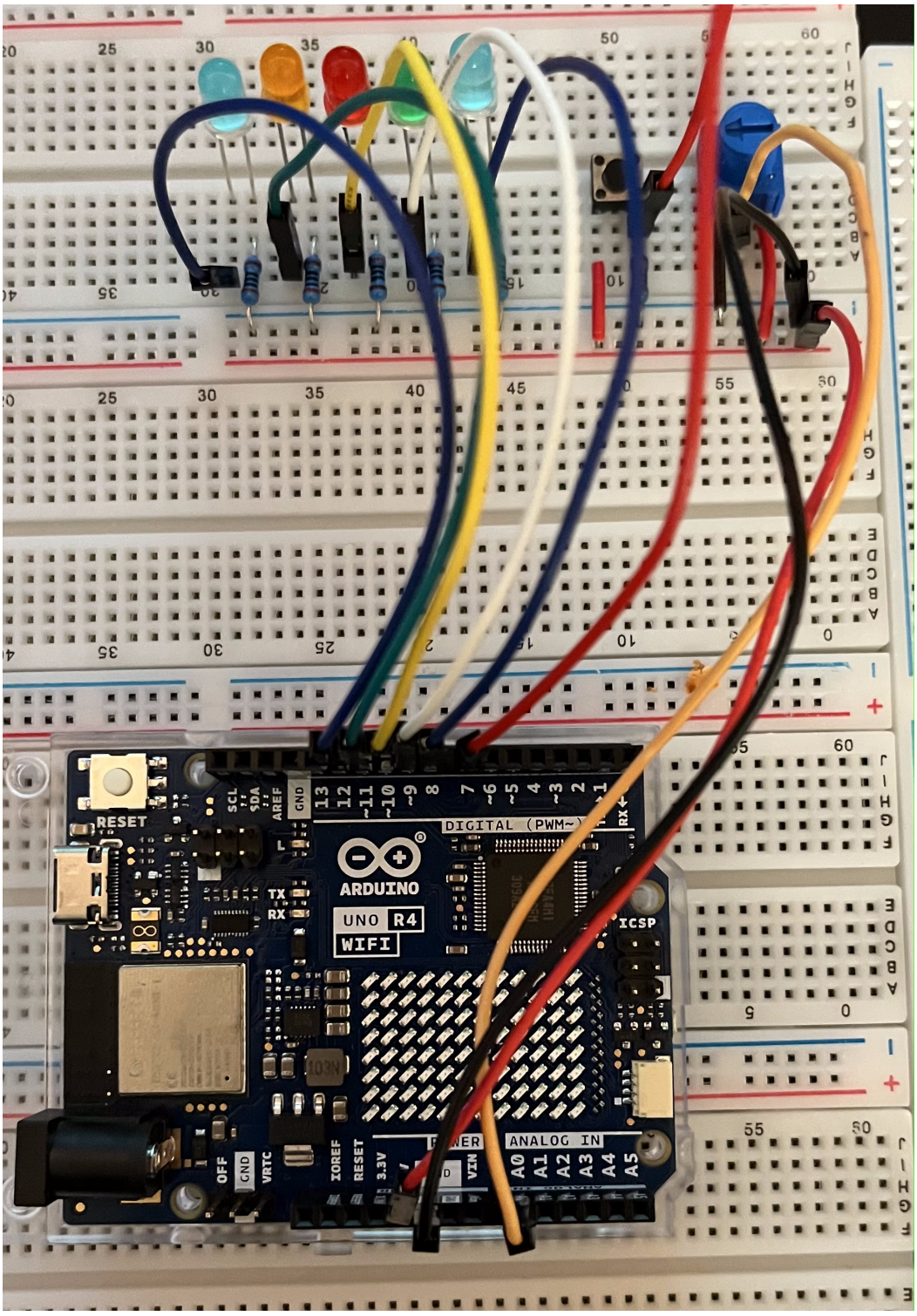

Figure 2: Circuit Implementation