PIE Mini-Project 1 Report (Bike Lights)
Anna Luiza Reis Braun & Kuhu Jayaswal
September 9th, 2025

I. Mini-Project 1 Introduction:

We designed and built a bike light system that cycles between multiple modes at the press of a button. The system integrates both digital and analog controls, with a push button to switch modes and a potentiometer to adjust LED brightness. Our objective was to create a compact and reliable system that enhances rider safety by offering clear signaling and adjustable visibility in different conditions.

This project combined electrical, mechanical, and software applications, giving us experience across multiple engineering domains. Electrically, we calculated resistor values, selected suitable components, and assembled a manufacturable circuit with clean connections. Mechanically, we designed a bike helmet CAD model in SolidWorks that can integrate the electrical components (e.g. arduino, breadboard, ...) on itself while maintaining visibility and rider comfort. On the software side, we programmed the Arduino in C++ to process both analog and digital inputs to generate six distinct LED output patterns.

Completing this project strengthened our skills in circuit design, component analysis, CAD modeling, and embedded programming. It also showed us how effective documentation supports problem solving and system development. More importantly, the final system demonstrates a real-world benefit: customizable lighting modes that can help bikers remain safe and visible under a variety of riding scenarios.

II. Project Breakdown:

1. <u>Electrical Aspect:</u>

We developed a functional circuit that powered five LEDs, a potentiometer, and a push button. The LEDs served as the primary output, while the potentiometer (analog input) adjusted brightness and the push button (digital input) cycled through the different modes. We designed the layout to be simple, manufacturable, and adaptable, while ensuring that the circuit operated safely and effectively.

2. <u>Software Aspect:</u>

We wrote C++ code in the Arduino IDE to control the LEDs. The program used the push button to switch between lighting modes and the potentiometer to adjust brightness levels. We structured the code for clarity and included comments to support debugging and be comprehensible by anyone viewing the code. Working on the program gave us practical experience with digital and analog inputs, as well as the logic needed to integrate them into a single system that works simultaneously with one another.

3. <u>Mechanical Aspect:</u>

We created a bike helmet in SolidWorks that integrates the arduino, breadboard, wires and LEDs in itself. The design balanced human anatomy, safety and practicality, protecting the components without obstructing the rider. This part of the project emphasized how mechanical design supports electrical and software systems, ensuring the light system is both effective and usable in real-world biking conditions but assuming that the arduino doesn't need to be connected to a computer and the rider could easily press a button in the helmet while biking.

III. Implementation & Functionality:

1. Electrical Aspect:

Hardware Used:

- Arduino UNO R4 WiFi
- 5 LEDs (red, orange, yellow, green, blue)
- Push Button
- Potentiometer
- Resistors

The LEDs were connected to PWM-capable microcontroller pins to allow brightness modulation through the potentiometer. Pin selection was deliberate, as only certain Arduino pins support PWM. Resistor values were calculated for each LED color using datasheet specifications, ensuring safe current levels and consistent brightness. We calculated resistor values for each LED using Ohm's Law:

$$R = \frac{V_{supply} - V_{forward}}{I_{forward}}$$

V_{supply}	=	Arduino output voltage (5V)
$V_{forward}$	=	forward voltage of the LED (from datasheet)
$I_{forward}$	=	forward current of the LED (from datasheet)

The following table shows the resistances we calculated for each LED using the equation above:

LED	**Calculated Resistance**
Red	150 ohms
Orange	290 ohms
Yellow	290 ohms
Green	1.55k ohms
Blue	78-90 ohms

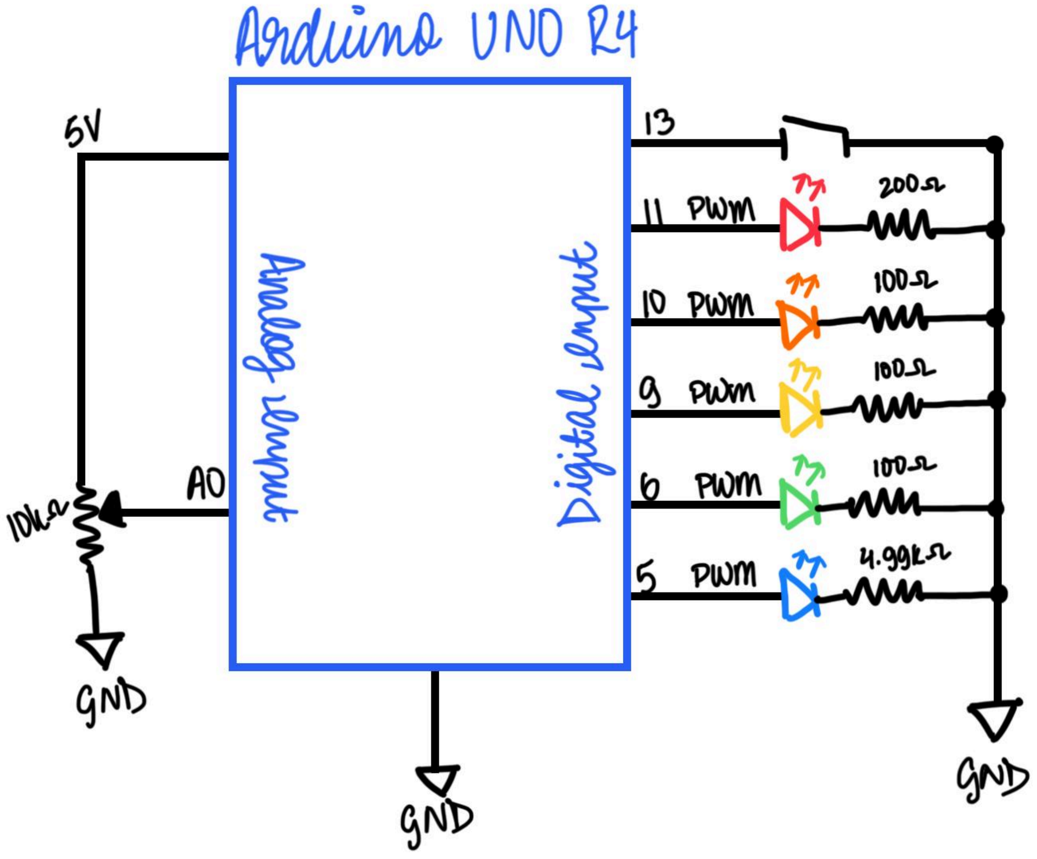

Image 1: Circuit Diagram with LEDs connected to PWM-capable microcontroller pins for brightness modulation. Resistor values were calculated using each LED's datasheet. Pin selection ensures PWM compatibility.

2. Software Aspect:

Software Used: Arduino IDE (C++)

We programmed the push button as a digital input with an internal pull-up resistor. Each time the button was pressed, the system advanced to the next mode by incrementing a counter variable (pattern). Once the counter reached six, it reset back to one, creating a continuous loop of modes. This gave the rider simple, single-button control over all light behaviors.

The potentiometer acted as an analog input connected to pin A0. We used the analogRead() function to measure its value (0–1023) and then mapped it to a PWM output range (0–255) with the map() function. This allowed us to adjust LED brightness in real time using analogWrite(). For example, in braking mode all LEDs turned on at the brightness level set by the potentiometer, while in blinking or sequential modes, the same brightness adjustment carried through each pattern.

Code:

```
// LED pins (must be PWM-capable pins to control brightness via potentiometer)
int led1 = 5; // red
int led2 = 6; // orange
int led3 = 9; // yellow
int led4 = 10; // green
int led5 = 11; // blue

// button variables
int button = 13; // pushbutton pin
int pattern = 1; // current LED pattern
int prev_button = HIGH; // previous button state
int curr_button = HIGH; // current button state

void setup() {
  // set LEDs as outputs
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);

  // set button as input with pull-up resistor
  pinMode(button, INPUT_PULLUP);

  // start serial monitor for debugging
  Serial.begin(9600);
}

void loop() {
  // read potentiometer (analog input)
  int pot = analogRead(A0);
  // map potentiometer value to brightness
  int brightness = map(pot, 0, 1023, 0, 255);
  // Serial.println(brightness); // print brightness for testing

  // read button state
  curr_button = digitalRead(button);
```

```
// detect button press and change pattern
if (prev_button == HIGH && curr_button == LOW) {
  pattern = pattern + 1;     // go to next pattern
  if (pattern > 6) {         // loop back after 6
    pattern = 1;
  }
  delay(200); // general delay
}
prev_button = curr_button;

// choose LED behavior based on pattern number

// pattern 1: biker is braking
if (pattern == 1) {
  // all LEDs on with brightness
  analogWrite(led1, brightness);
  analogWrite(led2, brightness);
  analogWrite(led3, brightness);
  analogWrite(led4, brightness);
  analogWrite(led5, brightness);
}

// pattern 2: biker is at rest (not "driving")
else if (pattern == 2) {
  // all LEDs off
  analogWrite(led1, 0);
  analogWrite(led2, 0);
  analogWrite(led3, 0);
  analogWrite(led4, 0);
  analogWrite(led5, 0);
}

// pattern 3: biker is "driving"
else if (pattern == 3) {
  // all LEDs blink together
  analogWrite(led1, brightness);
  analogWrite(led2, brightness);
  analogWrite(led3, brightness);
```

```
    analogWrite(led4, brightness);
    analogWrite(led5, brightness);
    delay(300);
    analogWrite(led1, 0);
    analogWrite(led2, 0);
    analogWrite(led3, 0);
    analogWrite(led4, 0);
    analogWrite(led5, 0);
    delay(300);
  }

  // pattern 4: biker is turning left
  else if (pattern == 4) {
    // LEDs light up one by one (red to blue)
    analogWrite(led1, brightness); delay(150); analogWrite(led1, 0);
    analogWrite(led2, brightness); delay(150); analogWrite(led2, 0);
    analogWrite(led3, brightness); delay(150); analogWrite(led3, 0);
    analogWrite(led4, brightness); delay(150); analogWrite(led4, 0);
    analogWrite(led5, brightness); delay(150); analogWrite(led5, 0);
  }

  // pattern 5: biker is turning right
  else if (pattern == 5) {
    // LEDs light up one by one (blue to red)
    analogWrite(led5, brightness); delay(150); analogWrite(led5, 0);
    analogWrite(led4, brightness); delay(150); analogWrite(led4, 0);
    analogWrite(led3, brightness); delay(150); analogWrite(led3, 0);
    analogWrite(led2, brightness); delay(150); analogWrite(led2, 0);
    analogWrite(led1, brightness); delay(150); analogWrite(led1, 0);
  }

  // pattern 6: biker has emergency lights on
  else if (pattern == 6) {
    // only first and last LED blink alternately
    analogWrite(led5, brightness); delay(150); analogWrite(led5, 0);
    analogWrite(led1, brightness); delay(150); analogWrite(led1, 0);
  }
}
```

3. <u>Mechanical Aspect:</u>

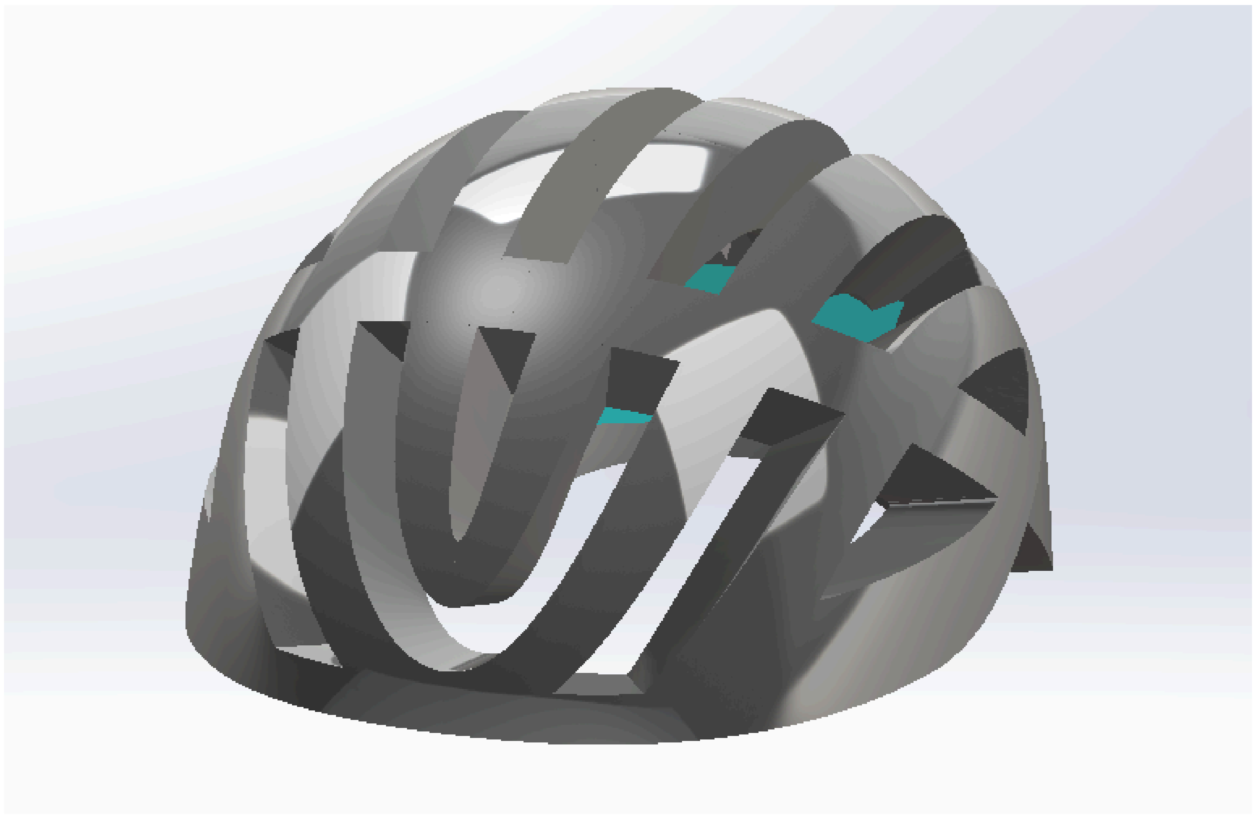

Image 2: Isometric View of Helmet CAD model

The mechanical design of the helmet was executed using SolidWorks, with a focus on ergonomic fit, component integration, and manufacturability. The project aimed to create a wearable system that securely houses the electrical components,Arduino, breadboard, LEDs, and wiring, while maintaining comfort, structural integrity, and ease of assembly. The design process was guided by anthropometric data, precision measurements, and iterative modeling, with all physical dimensions verified using digital calipers to ensure accuracy.

The initial step involved defining the geometry of the helmet based on human head dimensions for a medium-sized adult. Reference values for head width and length were 177mm and 228mm respectively.

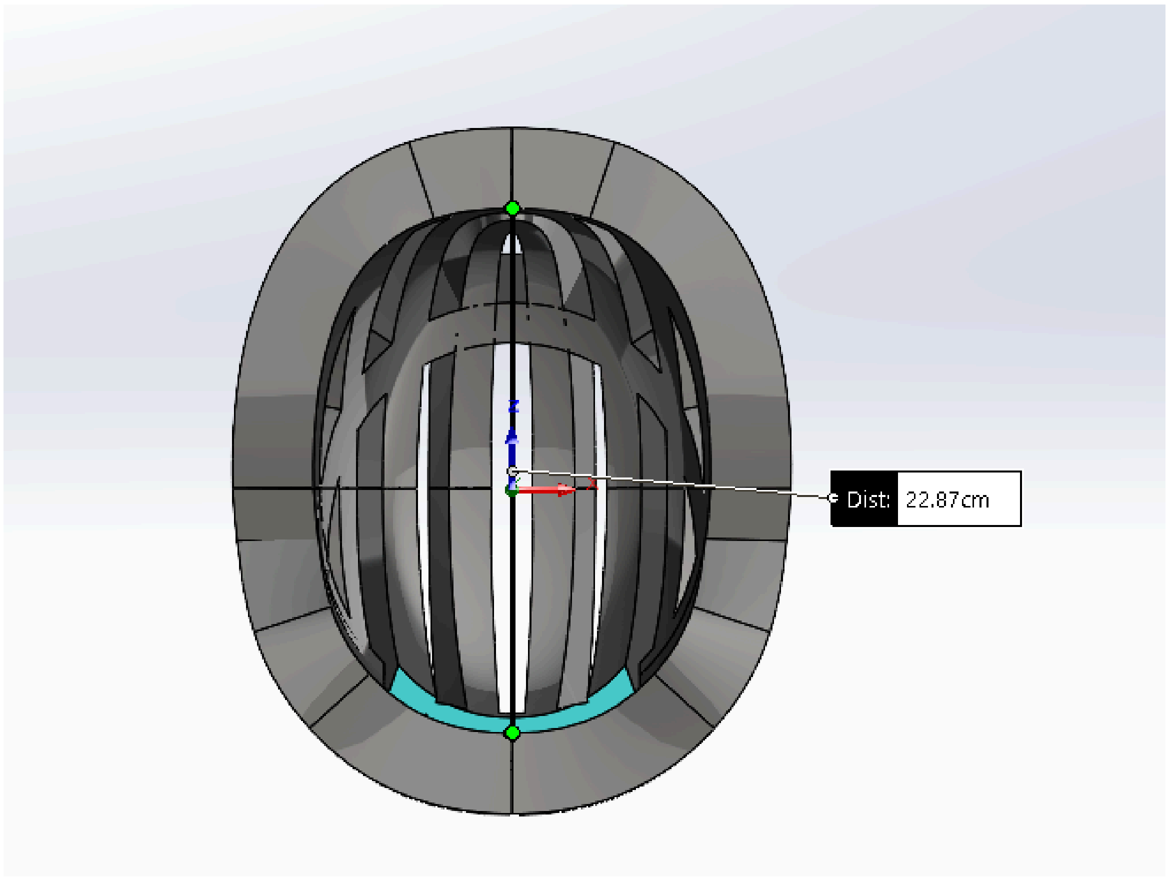

Image 3: Length of helmet

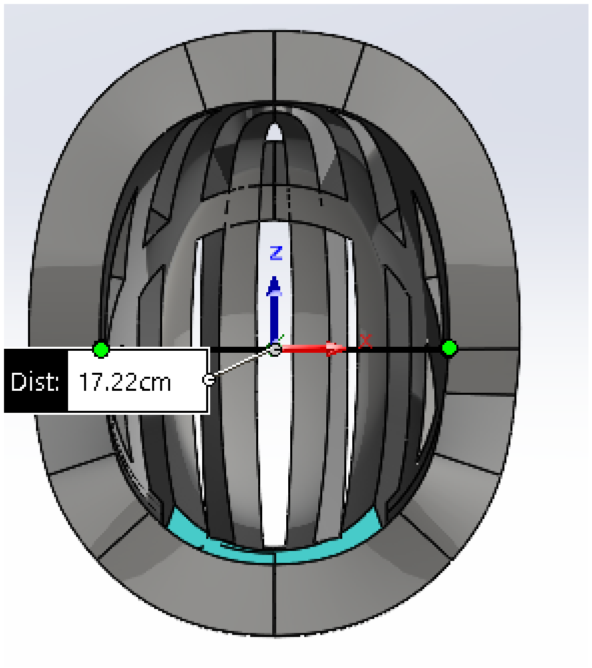

Image 4: Width of Helmet

To optimize internal space and maintain a low profile, the Arduino and breadboard were stacked vertically, with the Arduino placed beneath the breadboard, as seen in *Image 4.* This configuration minimized the footprint and allowed the LEDs to face outward for optimal visibility. The combined thickness of the two components was measured at 21.15 mm. A recessed pocket was designed into the rear section of the helmet to house this stack.

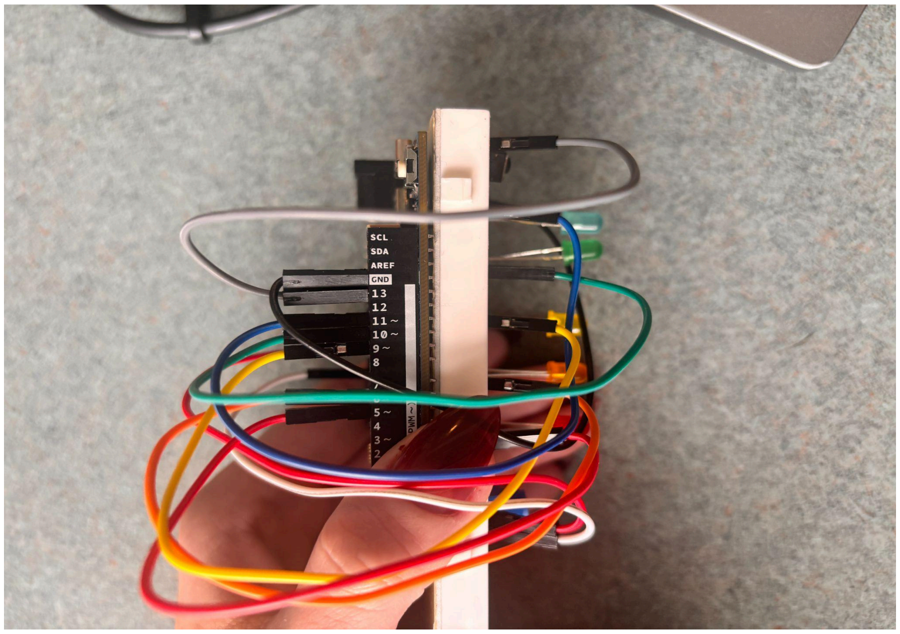

Image 5: Picture of arduino and breadboard stacked considered for CAD

The pocket depth was modeled at 21.77 mm, providing a tolerance of 0.62 mm to account for material expansion, assembly variability, and potential covering.

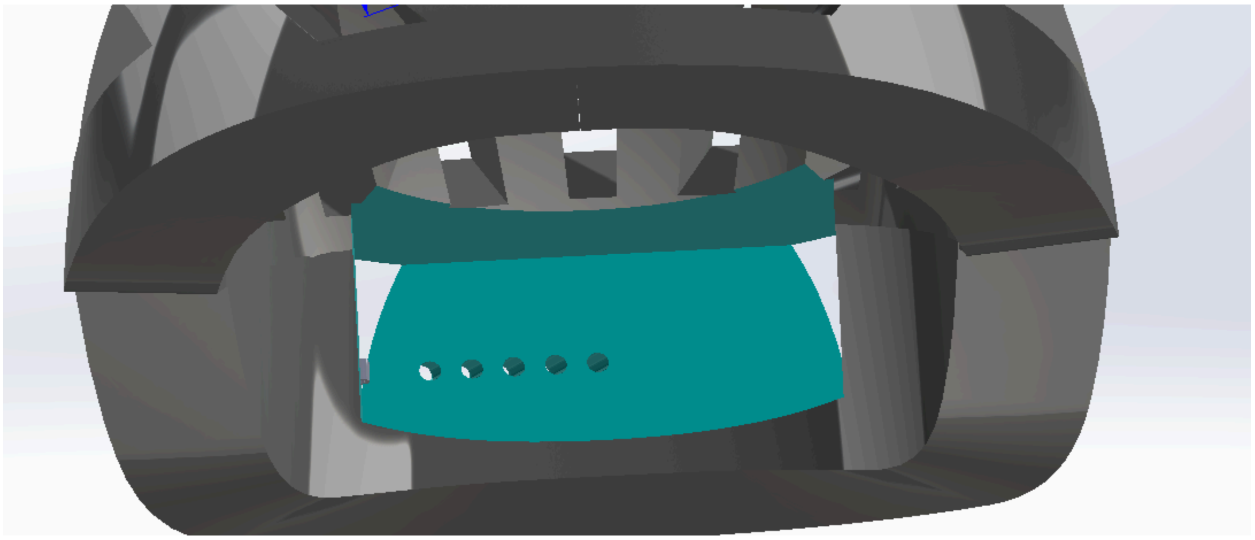

Image 6: Inside view of helmet CAD showing the pocket for arduino and breadboard

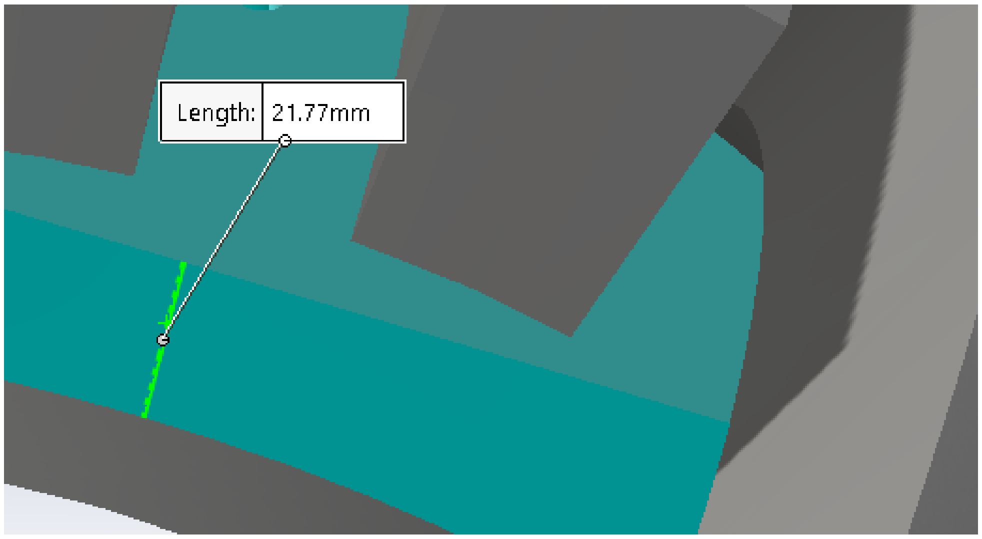

Image 7: Picture showing the width of pocket used

The breadboard measured 53.8 mm in width and approximately 111 mm in length. These dimensions were used to define the bounding box of the internal cavity and guide the placement of the LED array. Each LED had a diameter of 5 mm, and to ensure a secure press-fit during future fabrication, a tolerance of +0.05 mm was applied.

The LEDs were spaced 9.1 mm apart and positioned on the left side of the breadboard, aligned along the central width axis. The distance from the left edge of the breadboard to the first LED was measured at 15.13 mm, as seen in *Image 7.*

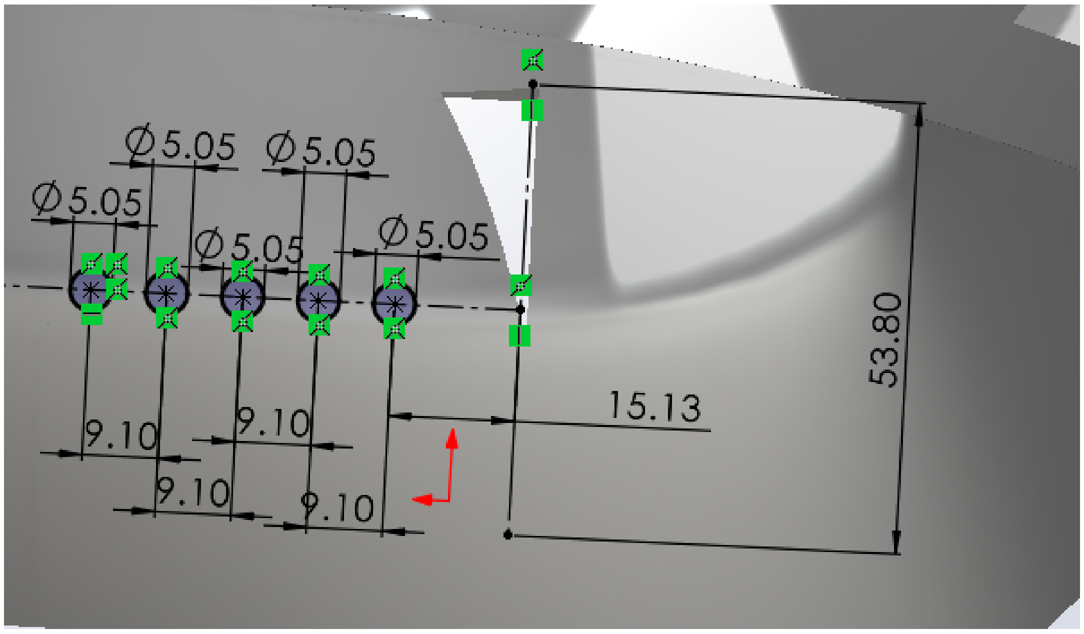

Image 8: Dimensions for the LED holes in the back of the helmet

To address potential issues with wire disconnection, a “hook” feature was incorporated into the helmet design. This hook serves as a secure placement for the wires to reduce possibilities of disconnections while biking.

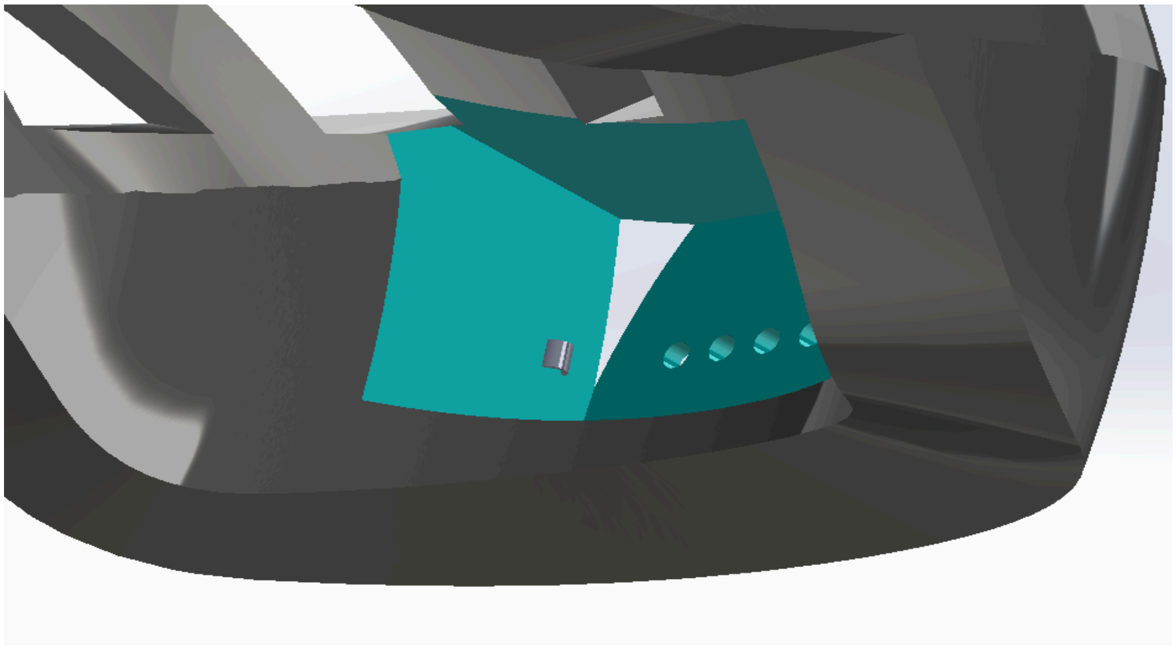

Image 9: Position of the wire hook in the pocket

4. <u>System Integration:</u>

We implemented six modes, controlled through the push button as shown in the table below. These modes cover common biking scenarios and provide the rider with clear, customizable signals. The potentiometer further enhances usability by adjusting brightness, which is especially useful in low-light, foggy, or night conditions. We designed the sequence to transition smoothly, allowing riders to switch between modes with minimal effort; just a press of a button.

Pattern	**Mode**	**Description**
1	Braking Mode	All LEDs turn on to indicate braking.
2	Rest Mode	All LEDs remain off when the bike is not moving.
3	Driving Mode	All LEDs blink to signal motion.
4	Left Turn Signal	LEDs light up sequentially from right to left.
5	Right Turn Signal	LEDs light up sequentially from left to right.
6	Emergency Mode	LEDs flash alternately in red and blue.

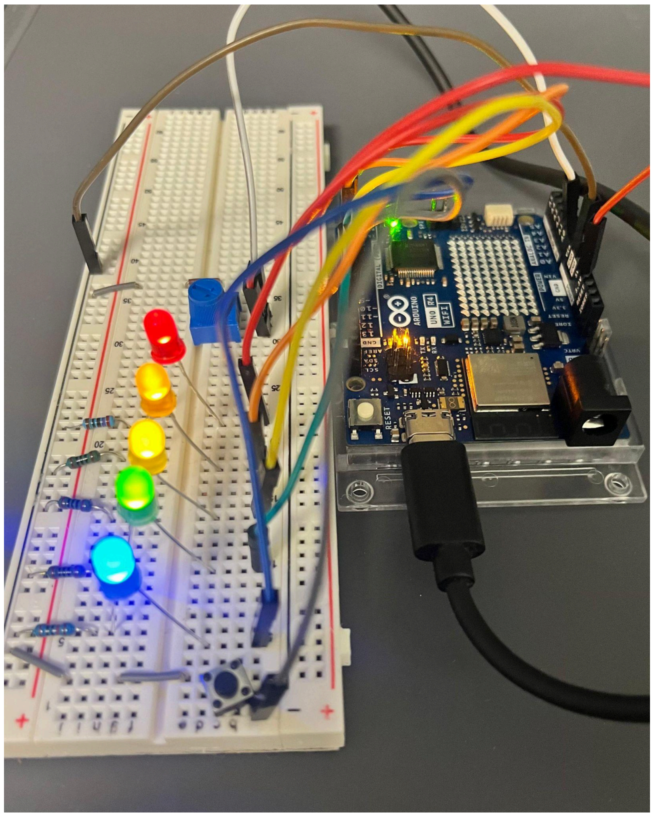

Image10: Arduino Circuit Picture

IV. Results:

Video Demo: PIE Mini-Project 1 Demo.mp4

V. Reflection:

1. Strengths:

We approached the project methodically, relying on datasheets and documentation for accurate design choices. We built a simple, manufacturable circuit and wrote clean, functional code. We also embraced new challenges, experimenting with both a potentiometer and an infrared distance sensor to extend functionality. We also learned how to integrate electrical components in mechanical designs and advanced our SolidWorks abilities using surface modeling for the helmet. Our teamwork and willingness to test different ideas helped us troubleshoot efficiently and adapt when issues arose.

2. Areas for Improvement:

While our system worked as intended and we are extremely proud of what we produced, upon reflection we identified several areas that could be improved for future iterations.

Electrically, we noticed discrepancies between our calculated resistor values and real-world performance. When we used the calculated values, the LEDs displayed uneven brightness, which resulted in a messy output with sputtering lights. For example, the blue and red LEDs were excessively bright, while the green LED was noticeably dim. To correct this, we experimented with different resistor values, as shown in the table, until the LEDs produced consistent brightness. These variations may have been caused by manufacturing tolerances between different companies, slight inaccuracies in datasheet specifications, or differences in the forward voltage of individual LEDs.

The resistances we used are noted in the table below:

LED	**Calculated Resistance**	**Experimetal Resistance**
Red	150 ohms	200 ohms
Orange	290 ohms	100 ohms
Yellow	290 ohms	100 ohms
Green	1.55k ohms	100 ohms
Blue	90 ohms	4.99k ohms

Mechanically, the improvements would be on enhancing the integration of the electrical components for better stability and ease of maintenance. The current helmet design uses a stacked configuration for the arduino and breadboard which minimizes footprint but may still lead to potential issues with component movement or disconnections. A more modular approach, where the components are housed individually, in easy and accessible compartments would allow for more flexibility.

Another area for improvement lies in the handling of the wiring and connections. The current design includes a hook feature to secure the wires in place, which helps prevent disconnections. However, there could be further refinement in wire management to minimize the risk of wear and tear during prolonged use. For example, using flexible, durable wiring or protective sheaths would help avoid damage from constant motion.

On the software side, while our code worked well, it relied heavily on if–else statements. Using cleaner structures such as loops or arrays or switch cases would make the program more efficient and scalable for future designs.This is a skill we hope to explore and build on as we get onto more projects.

VI. References:

1. Data Sheets:
    - Red LED: https://www.digikey.com/en/products/detail/kingbright/WP7113ID/1747663
    - Yellow LED: https://www.digikey.com/en/products/detail/lite-on-inc/LTL-4273/3198546
    - Green LED: https://www.digikey.com/en/products/detail/kingbright/WP7113LGD/1747664
    - Blue LED: https://www.digikey.com/en/products/detail/w-rth-elektronik/151051BS04000/4490009

2. Arduino:
    - Button: https://docs.arduino.cc/built-in-examples/digital/Button/
    - Potentiometer: https://docs.arduino.cc/learn/electronics/potentiometer-basics/
    - IR Sensor: https://www.makerguides.com/sharp-gp2y0a21yk0f-ir-distance-sensor-arduino-tutorial/

3. CAD:
    - How to build surfaces: https://youtu.be/Z7mrcn6Jssw
    - Surface modeling: https://youtu.be/_2X4MbfUZTo
    - Human head sizes: How Big Is a Human Head? Average Size and Variations - Biology Insights