HelmetIQ

Smart Bike Helmet with Intelligent Lighting, Ride Data, and Safety Features

Department of Electrical Engineering and Computer Science
University of Central Florida
Dr. Chung Yong Chan
SD2 Final Report

Group 8

Ana-Victoria Elias	Electrical Engineering
Hector Agosto	Electrical Engineering
Christopher Bowerfind	Computer Engineering
Ethan Hymans	Computer Engineering

Reviewers

Dr. Kalpathy Sundaram
Mrs. Jacqueline Sullivan
Dr. Sonali Das
Dr. Ronald DeMara

Table of Contents

1. Executive Summary

The rapid advancement of wearable technology over the past two decades has led to a growing focus on fitness and healthy living. As a result, cycling has become an increasingly popular mode of transportation for professionals, children, and adults in urban and suburban areas, whether commuting to work or school or simply riding for leisure. However, despite the numerous benefits of cycling, riders face significant risks, primarily from motorists rather than their own actions. The National Highway Traffic Safety Administration reports that approximately 45,000 cyclists are involved in accidents annually, with a helmet often being the only protective gear worn. While encouraging cyclists to wear or carry additional safety equipment can be challenging, integrating extra safety features into an already widely-used item, such as a helmet, could provide a practical solution to enhance cyclist safety without burdening riders with the need to manage multiple pieces of gear.

Our project HelmetIQ, aimed to prevent accidents by incorporating intelligent lighting features and ride data collection into a product that cyclists already use. The primary objective wass to provide enhanced safety features without encumbering the cyclist with additional equipment, as traditional safety gear often requires extra items that can be burdensome and detract from the enjoyment of cycling.

Designed to seamlessly integrate safety and tracking technology, HelmetIQ is indistinguishable from a traditional bike helmet when worn. It features dynamic lighting that adjusts according to the time of day, automatic brake lights, and a mobile application for data tracking. By combining the functionality of multiple cycling accessories, such as bike computers, light bars, and standard helmets, into a single, user-friendly device, HelmetIQ presents a comprehensive solution that improves both safety and convenience for cyclists.

Our project aimed to combine the often-competing demands of safety and convenience by offering a product that not only safeguards cyclists but also enhances their overall riding experience. By consolidating essential cycling tools into a single device, HelmetIQ distinguishes itself as an all-in-one solution tailored to the needs of contemporary cyclists.

HelmetIQ's versatility extends beyond cycling, making it suitable for other forms of personal transportation, such as skateboarding, rollerblading, and electric scooters, which are gaining popularity in urban areas as people seek efficient and eco-friendly commuting options. The helmet's dynamic lighting, automatic brake lights, and ride data tracking capabilities offer increased protection and awareness for users of these alternative transport methods, promoting safety across various activities.

Our group is confident in HelmetIQ's potential to revolutionize personal safety for cyclists and users of other transport methods. We believe that our innovative approach to integrating safety features, data tracking, and user-friendly design established HelmetIQ as an indispensable tool for those seeking efficient, safe, and enjoyable transportation options. By prioritizing safety without sacrificing convenience, HelmetIQ addresses a pressing need in today's urban and suburban environments. We take pride in the tangible benefits our project offers to individuals looking for practical and secure ways to navigate their surroundings. Through our dedication to continuous improvement and innovation, we are confident that HelmetIQ not only succeeded as a project but also provided an invaluable experience to those involved in its development and use.

2. Project Description

2.1. Motivation and background

The bicycle helmet's primary purpose is to reduce head injuries for cyclists in the event of a collision. While there are many brands and variations of helmets equipped with different injury prevention technologies, HelmetIQ is not intended as a protection device. Instead, it will utilize an already existing, safety-certified helmet as the base of construction. With the recent projects around the UCF campus tailored towards pedestrian and biker safety, HelmetIQ aims to extend this by providing further safety features to the user.

Our team, consisting of avid cyclists at different points in our lives, has observed a growing reluctance to continue cycling, largely stemming from increased hazards, distracted drivers, and the overall lack of safety precautions for bicyclists in the commuter infrastructure. Bicycles typically lack integrated safety features such as lights, which must be purchased separately and attached. These aftermarket lights often suffer from aesthetic and functional issues, and it is easy to forget to turn them off during the day, leading to a high risk of dead batteries when the lights are most needed. Additionally, the lack of synchronization between front and rear lights can lead to inconsistent visibility.

HelmetIQ addresses these challenges by incorporating adaptive lighting, automatic brake lights, and ride data tracking into a single, cohesive unit that enhances both safety and convenience for cyclists. Running wires along the bike frame for synchronization is cumbersome and impractical for the average cyclist, so our objective is to modernize the safety aspects of cycling by seamlessly integrating advanced safety features into the helmet.

To persuade cyclists to use HelmetIQ, we explored additional functionalities beyond safety features and lights. Given the modern affinity for data analytics, ease of use, and intuitive interfaces, we have decided to create an app that connects directly to the helmet. The primary objective of the application is to alert when the cyclist has been in an accident. Using the existing sensors within the helmet, we will be able to accurately calculate when the cyclist has been in a collision, and the mobile application will take action by alerting the user's emergency contacts.

As a group, we are driven by practicality and function, and HelmetIQ is a project that allows us to innovate and learn different skills while engineering a product that could be intensely beneficial in everyday life. Although none of us aspire to be professional cyclists, we share a passion for integrating technology into everyday life in practical and beneficial ways. If even one person in the real world finds this project useful, then that is a success for us. We are committed to improving lives by making common activities like cycling safer and more enjoyable.

2.2. Existing Products and Prior Related Works

The market for "smart" bike helmets is relatively unsaturated, with existing products offering basic features such as integrated lights, face shields, and speakers. However, these helmets often come with a hefty price tag, making them inaccessible to the average consumer. For instance, some helmets retail for over $200, despite only incorporating inexpensive components like $20 lights and $5 plastic shields. This pricing seems unjustified given the lack of significant smart features in these products.

We have created a better helmet with more advanced technology at a similar price point. While current products have the advantage of seamless integration between the helmet design and technology, resulting in a more aesthetically pleasing appearance, our project will focus on incorporating a greater amount of cutting-edge technology. Although our helmet has a less conventional design due to our inability to manufacture the helmet base itself, we are confident that the advanced features compensate for any minor aesthetic compromises.

Public reviews of existing "smart" bike helmets provide valuable insights into the current market and reveal areas where these products fall short. Some common complaints include inconsistent performance of features like turn signals and excessive weight causing neck strain. By examining these shortcomings, we can optimize our design and technology to address the issues faced by our predecessors while also improving the consistency and scale of the implemented technology. The availability of user experiences and opinions in public reviews gives us an advantage in our research process, as we leveraged this information to create a product that better meets the needs and expectations of potential customers.

2.3. Goals and Objectives

This project's goals are to design and implement an advanced smart helmet system with multiple integrated sensors for enhanced safety. The key components include collision detection sensors, touch-activated turn signals with haptic feedback, intelligent night-time lighting, and instant mobile app collision response.

Basic objectives included the implementation and integration of sensors to calculate and detect if an impact has occurred; touch-activated turn signals developed using capacitive touch sensors embedded into the helmet to allow users to activate turn signals with a touch and provide immediate haptic feedback to confirm activation; brake lights that communicate with an accelerometer to automatically turn on during deceleration/acceleration and turn off when motion stops.

Advanced objectives included the creation of a companion app to monitor and control helmet features via wireless communication (e.g., Bluetooth). The app provides real-time status updates, battery information, ability to input emergency contacts, and other ride metric data.

Stretch goals will included the integration of voice control technology to enable hands-free operation of turn signals. This will further enhance the safety and convenience for the user by allowing them to keep their hands on the handlebars while signaling.

2.4. Description of Features/Functionalities

Haptic Feedback Turn Signals

- The haptic feedback turn signals are designed to provide a reliable method of signaling an intention to turn. The system uses capacitive touch sensors embedded in the helmet to detect user touch inputs. To avoid false positives from accidental touches or unwanted external factors, the system incorporated a debouncing mechanism that allows only deliberate touches to be registered. Upon detecting a valid touch input, the system provides immediate tactile feedback through haptic actuators. This feedback confirms to the user that the turn signal has been activated, enhancing confidence and reliability.

Accelerometer

- A 3-axis microelectromechanical accelerometer is used to continuously monitor the cyclist's linear acceleration. The data gathered from this device is used to estimate the cyclist's motion and detect deceleration or acceleration by setting various thresholds. When the accelerometer detects that the cyclist is slowing down or speeding up, it activates the brake light, providing a clear indication to other road users. In addition to this function, the accelerometer is programmed to detect excessive deceleration, which may indicate a potential collision. In such cases, the device sends a signal to the companion mobile app, triggering a notification to be sent to the user's designated emergency contact. This feature ensures that in the event of an accident, help can be dispatched quickly, enhancing the overall safety of the cyclist.

Dynamic Lighting System

- The dynamic lighting system operates in a manner similar to the automatic on/off functionality found in modern car headlights. By utilizing a light sensor, the helmet continuously assess the surrounding environment and determine when to activate the front and rear lights. This feature proves particularly beneficial during nighttime rides, as it significantly enhances the cyclist's visibility to other road users. Moreover, by automating the lighting system, the helmet eliminates the need for the cyclist to manually control the lights, allowing them to focus their attention on the road and their surroundings. This hands-free operation contributes to a safer and more convenient riding experience, ensuring that the cyclist is always visible and well-illuminated in low-light conditions.

Collision Detection System

- The collision detection system actively processes data from the accelerometer and gyroscope sensors integrated into the helmet using the microcontroller unit (MCU). By continuously analyzing this data, the MCU can identify any sudden changes in acceleration or orientation that may indicate a collision. In the event that a collision is detected, the MCU immediately transmit the relevant data through Bluetooth, to the companion mobile app. Upon receiving this information, the app automatically sends a text message to the user's pre-designated emergency contact(s), alerting them to the potential incident and providing them with the necessary details. This swift communication system ensures that in the case of an accident, the cyclist's loved ones or emergency services can be promptly notified, facilitating a rapid response and increasing the chances of timely assistance.

2.5. Design Requirements

To accomplish the listed goals and objectives of this project, our design must adhere to the following requirements:

- Capacitive Touch Sensors Sensitivity: The sensors must detect touch on the helmet material with high accuracy and minimal false positives.
- Response Time: The system must provide immediate feedback, with a maximum delay of 0.5 seconds from touch detection to actuation.
- Durability: Sensors must withstand minimal outdoor conditions, including temperature variations, low level moisture, and indirect physical impacts.
- Haptic Feedback Actuators: This must provide noticeable feedback without causing discomfort.
- LED Brake Light: These LEDs will light up red to indicate that a cyclist is slowing down and will serve as a signal to cyclists following behind. These will be located on the back of the helmet.
- LED Turning Signal: These LEDs are to designate whether the cyclist is turning left or right. This is done by flashing the turn light on the side corresponding to their turn.
- Front Headlight: This will be a light designed to clearly make the road in front visible during nighttime or low visibility situations.
- Brightness: These LEDs should have high visibility to allow other cyclists to detect them at all hours, day and night.
- Collision Detection: This sends a notification to the application. It is very important to be both accurate and fast.
- Comfort: The helmet should fit snugly and be comfortable for multiple hours of bike riding.
- Battery Life: With the average cyclist having an average of four hours of cycling time, battery life should be at an absolute minimum of 4 hours long.
- Safety: The helmet will also meet both federal and industry standards to ensure the maximum safety of customers.

2.6. Engineering Specifications

Table 2.1 *- Engineering Specifications*

Component(s)	Parameter	Specification

Haptic Feedback System	Feedback Time	< 0.5 seconds of activating the blinkers
Collision Detection System	Detection Delay	Respond within 2 seconds on the mobile app with collision popup
Environmental Lighting Adjustment	Activation Delay	Respond to changes in environmental lighting within 2 seconds
Capacitive Touch Sensors	Sensitivity	Detects 3 to 6 newtons of force
MCU and Actuators	Response Time	< 0.5 seconds
LED Indicators	Brightness	Visible from 40 meters
Battery	Battery Life	3 hours continuous use
Helmet Weight	Weight	Less than 3 kg additional weight
Helmet Enclosure	Durability	Withstands 10 pounds of force
Communication Module (optional)	Range	Maintain connection within 15 meters
Light Sensor	Light Detection	Detects Lux value within 1 seconds
Headlight	Visibility	Visible from 100 meters

The rows highlighted in blue are the specifications that were demonstrated.

2.7. House of Quality

The House of Quality is an analysis of our project by aligning the engineering requirements with the marketing requirements and identifying the relationships between the different aspects. Understanding these tradeoffs enhances our ability to predict the impact of modifying individual components of the project allowing for more informed decision making when these tradeoffs must be decided.

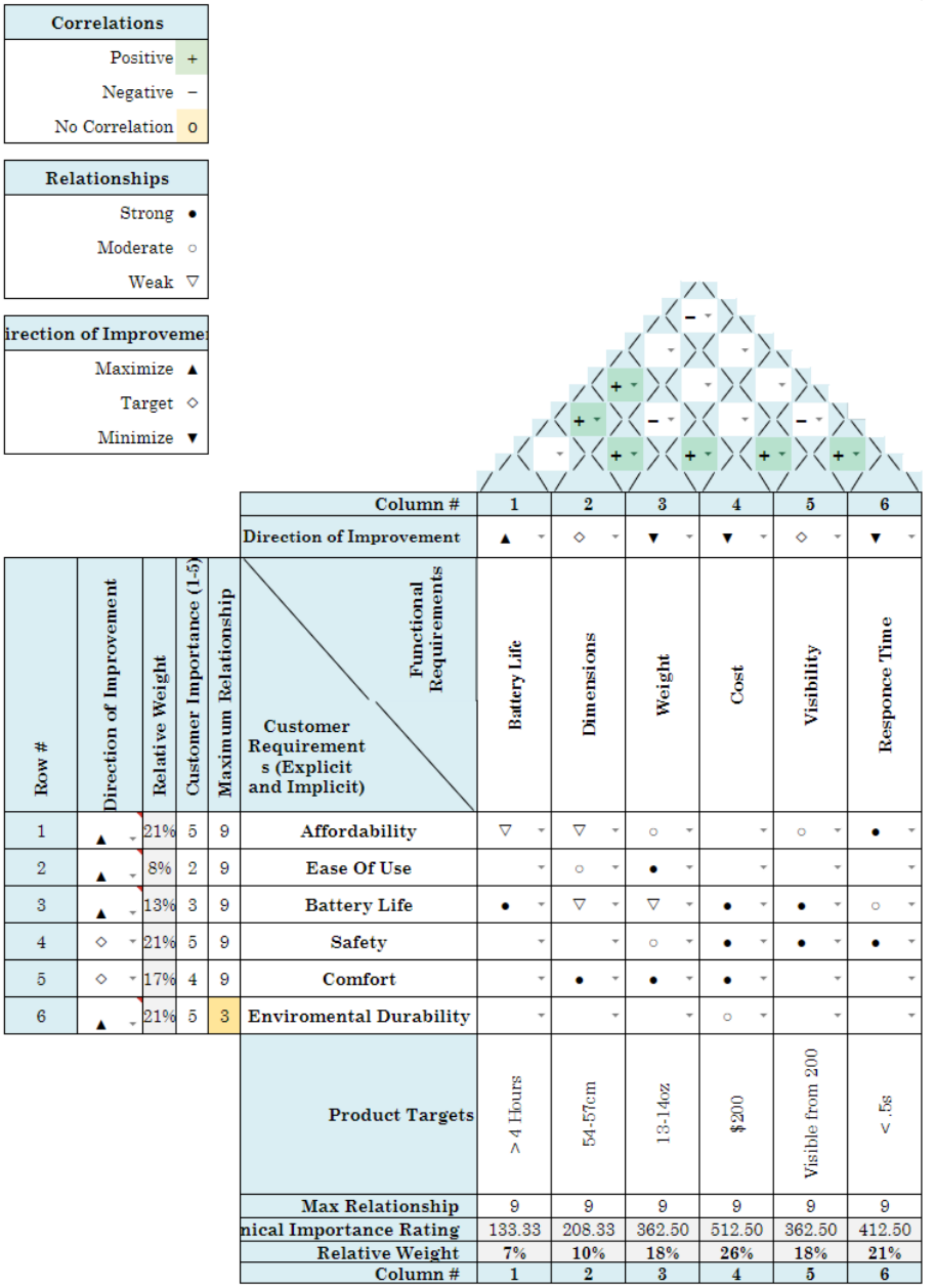

Figure 2.1 *- House of Quality Diagram*

2.8. Hardware Block Diagram

This hardware model diagram outlines the various components integrated into a microcontroller unit (MCU) for HelmetIQ. The components include headlights for improved visibility, turn signal LEDs for directional changes triggered by haptic feedback, and brake lights for enhanced safety during deceleration. The system also features a gyroscope and accelerometer to monitor and respond to impacts and changes in speed. A light sensor adjusts the default lighting setting based on ambient conditions. Bluetooth connectivity ensures communication with the mobile app and a battery powers the entire system. The battery, MCU, Bluetooth module, accelerometer, and gyroscope are all be part of the PCB. The rest of the sensors and lights are required to be exposed to the environment so selecting components that can withstand certain environmental possibilities are essential to the reliability of the technology. Also, the sensors must be able to measure within the value ranges we need to measure.

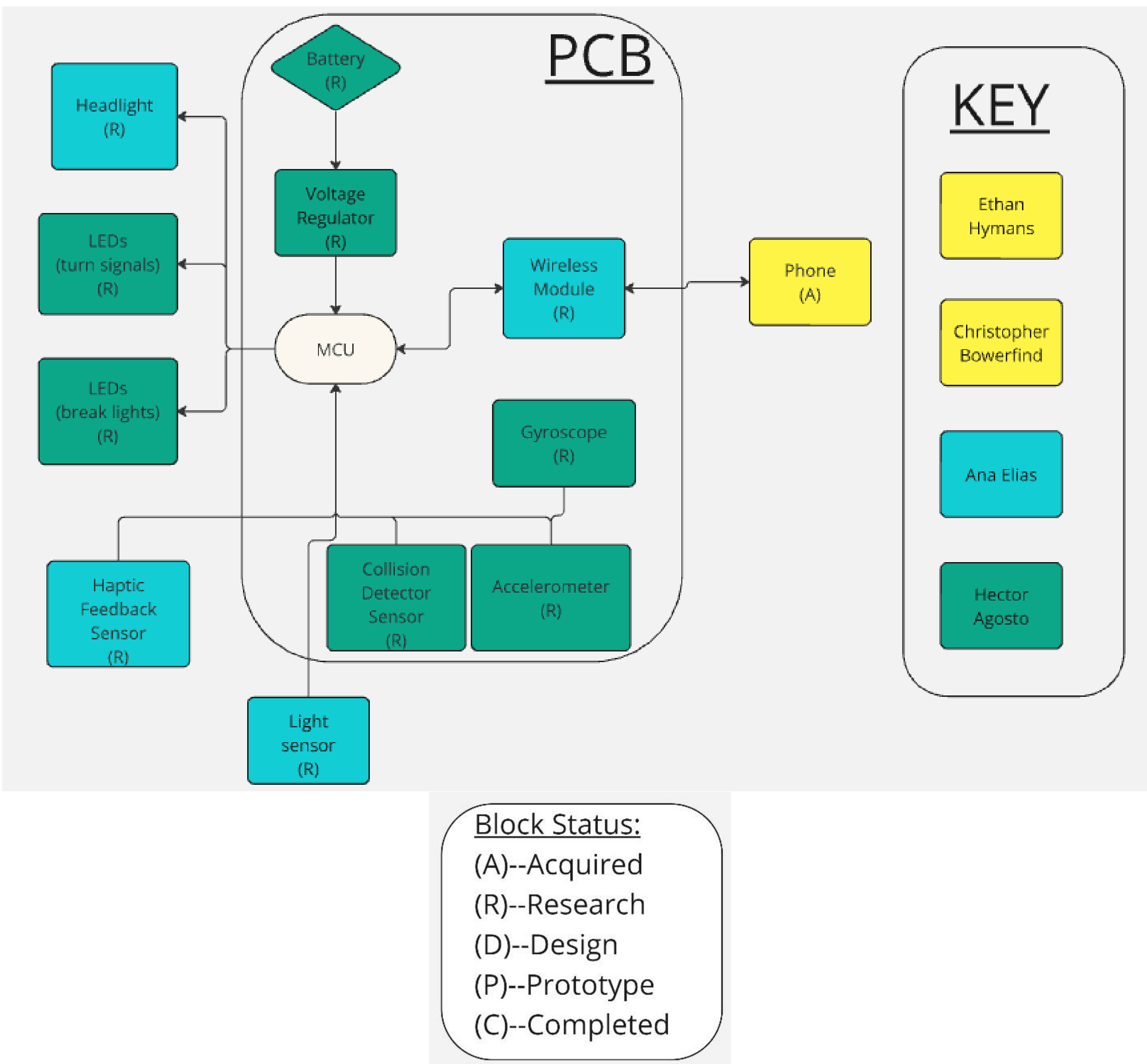

Figure 2.2 - *Hardware Block Diagram*

2.9. System Response Block Diagram

This MCU software flowchart illustrates the parallel processes that monitor HelmetIQ's sensor data and responding accordingly. The flowchart shows comparative statements based on inputs from various sensors. The light sensor determines whether to turn on the LEDs based on ambient light levels. The accelerometer monitors deceleration to trigger brake lights. Haptic touch inputs are used to toggle left or right turn signals, and the combined data from the gyroscope and accelerometer detects potential collisions. If a collision threshold is surpassed, a collision detection response is activated, sending a wireless signal to a paired phone. This flowchart ensures that the helmet's systems respond accurately and correctly to environmental and user inputs, ensuring reliability and expected functionality.

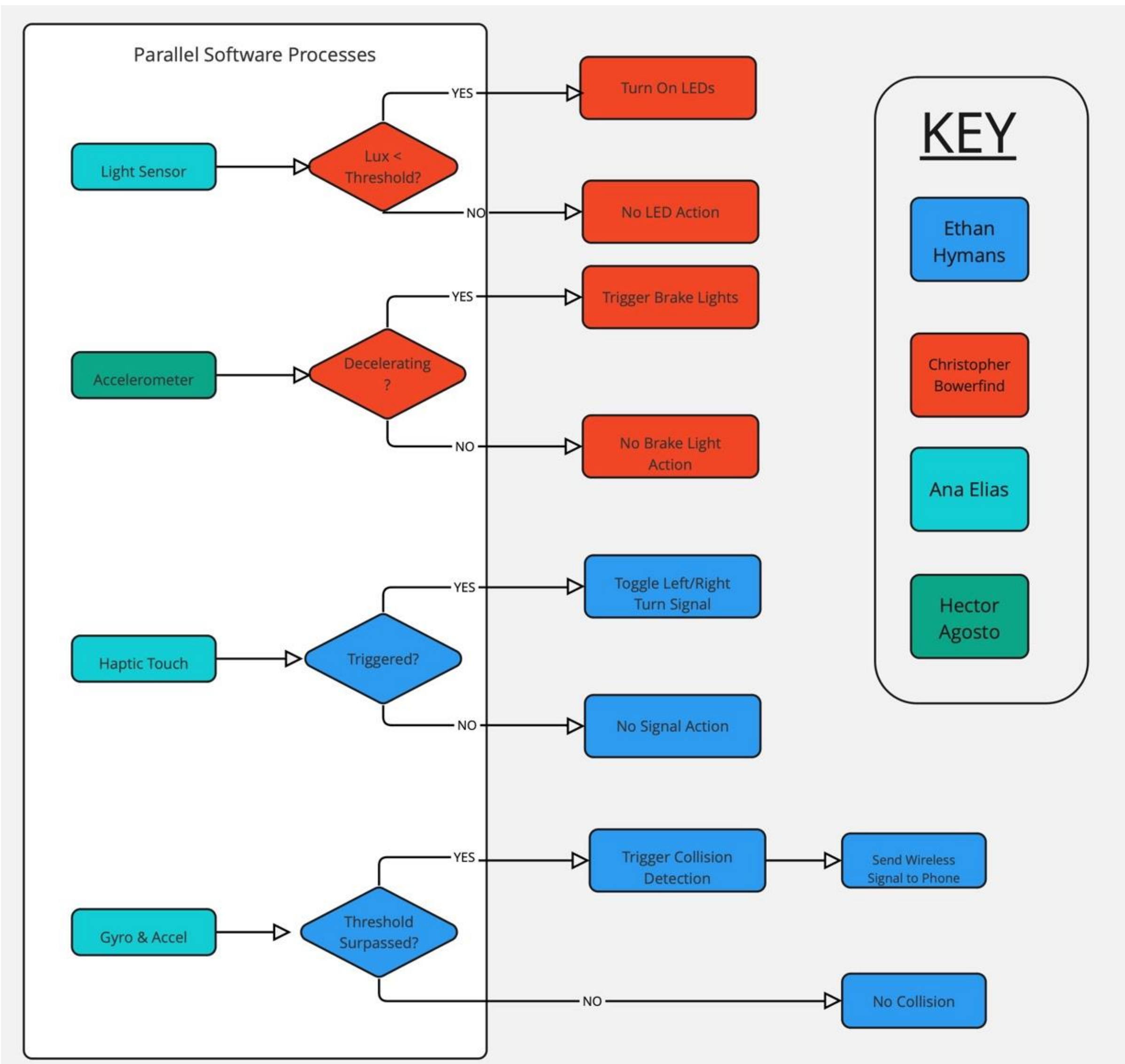

Figure 2.3 - *System Response Block Diagram*

This mobile software flowchart depicts the communication and response for collision detection. When a collision is detected by the MCU, it sends a Bluetooth signal to a paired mobile device. Upon receiving this signal, the mobile app opens to readily make a call to the rider's emergency contact. If no collision is detected, the signal is ignored to prevent false alarms. This flowchart creates a closed loop within the mobile device and doesn't need to rely on other factors like cellular or an internet connection to function properly.

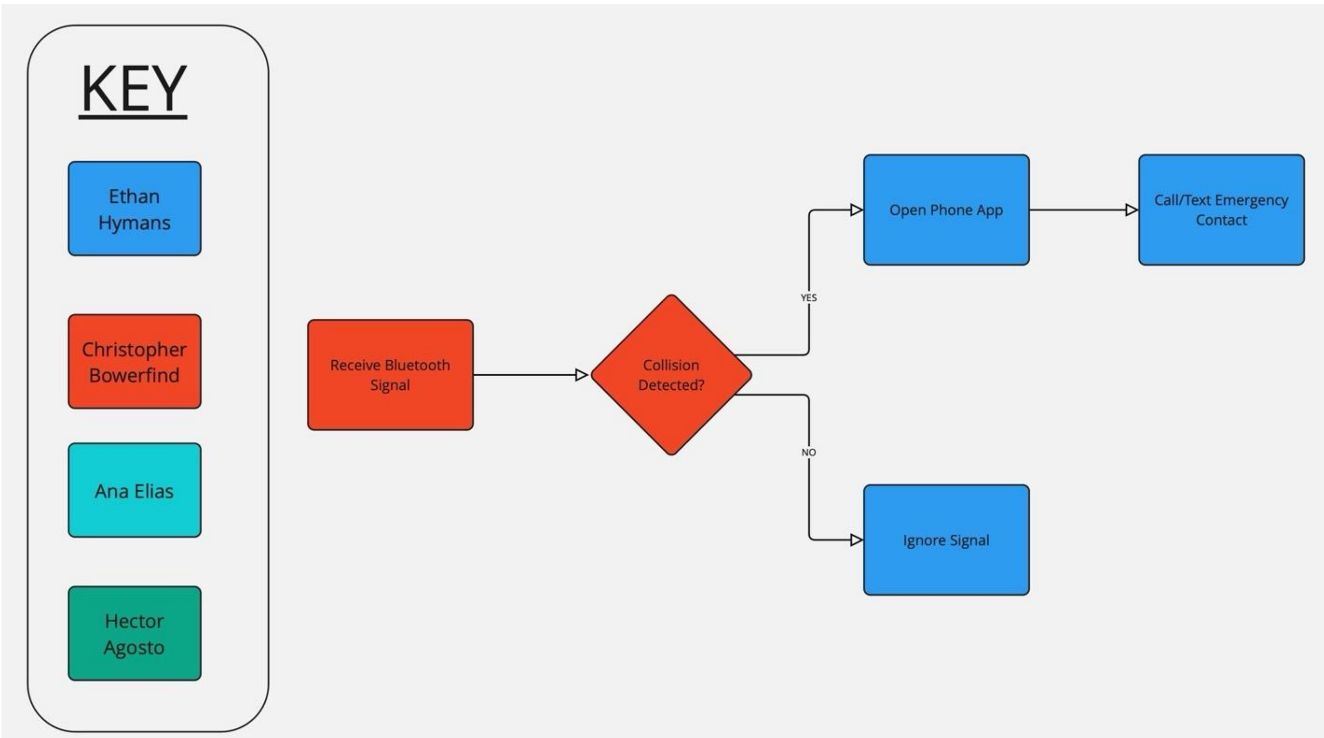

Figure 2.4 *- Mobile Software Block Diagram*

3. Research and Investigation

3.1. Technology Comparison and Selection

For our project the research and selection of specific technologies is important to ensure that the entire system is compatible. If any technologies are incompatible, we will have wasted time and money ordering parts that will not fit together. This could be potentially disastrous for the project timeline. These chosen technologies include all sensors, MCU, wireless technologies and I/O technologies. For this project, there are specific specifications that must be met and in order to do so the correct existing technologies must be selected to guarantee that we are able to meet all of the specifications that we have set.

3.1.1. Haptic Technology

Haptic technology, also known as tactile feedback technology, simulates the sense of touch by applying forces, vibrations, or motions to the user. It enhances user interaction by providing physical feedback in response to user actions, making it particularly useful in applications such as mobile devices, virtual reality, and wearable technology. This project integrates haptic feedback to alert the user that an action, in this case the activation of turn signals, has been initiated.

Haptic feedback is categorized into three types: vibrotactile feedback, force feedback, and electrostatic feedback. Vibrotactile feedback uses vibrations to convey information and is the most relevant to our project. It is commonly used in smartphones and gaming controllers to provide users with a sense of touch through small, precise vibrations. This feedback offers minimal vibrations, strong enough to alert the user that it has been activated and sensitive enough to not irritate and distract.

Force feedback actuators provide physical resistance or force to the user and is mostly utilized in virtual reality systems for the purpose of enhancing realism and immersion. The scope of this project requires small sensors and sensitive feedback.

Electrostatic actuators provide haptic feedback by modulating the friction between the user's skin and the device surface using electrostatic forces, creating a sensation of texture or movement. By applying an alternating voltage to a conductive surface, the electrostatic forces between the surface and the user's skin are modulated. This changes the friction, creating a tactile sensation. In general, these sensors require sophisticated control systems to modulate the electrostatic forces effectively and may be out-of-scope for this project.

Vibrotactile and electrostatic feedback offer the most relevant actuators for this project and will be discussed in detail in the following sections.

3.1.1.1. Vibrotactile Actuators

Eccentric Rotating Mass (ERM) Motors

ERM motors are a common type of actuator used for generating haptic feedback. These motors create vibrations by spinning a mass that is offset from the center of the motor's shaft, causing an imbalanced centrifugal force. ERMs are used in devices where simple, low-cost haptic feedback is required. They consist of a small DC motor with an off-center weight attached to the shaft. When the motor is powered, the shaft spins, and the eccentric mass creates a vibration due to the unbalanced centrifugal force. The frequency and amplitude of the vibration are determined by the speed of the motor and the size of the offset mass.

The core components of an ERM consist of a DC motor which drives the rotation, an eccentric mass which is the off-center weight attached to the motor shaft, and a housing. These actuators produce motion across two axes.

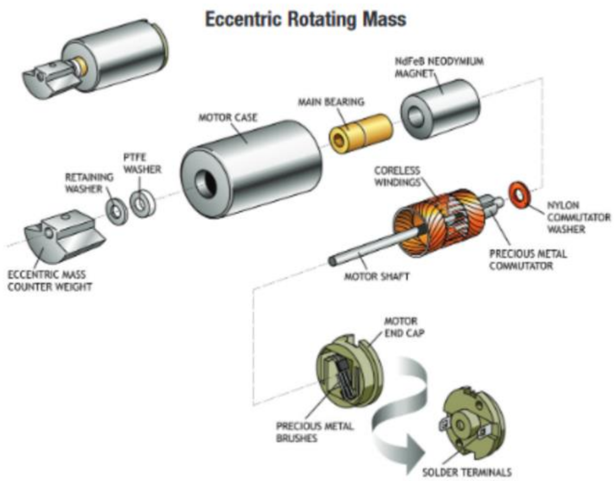

Figure 3.1 *- An exploded view of a modern coin-style ERM by Precision Microdrives (Image courtesy of Precision Microdrives) [37]*

ERMs are usually controlled via Pulse Width Modulation (PWM) signals to vary the motor speed and vibration intensity. The amplitude of the vibration is correlated to the motor speed and operates within a frequency of 50 to 100 Hertz. However, their response to these signals is less precise compared to Linear Resonant Actuators (LRAs) and piezoelectric actuators.

It is important to note that ERMs are cost effective and readily available yet require a higher power input compared to other haptic actuators. They can draw significant current, especially during startup and have a slow response time, typically between 50 to 100 milliseconds.

Piezoelectric Actuators

These actuators use piezoelectric materials that change shape when an electric field is applied, producing vibrations. They are efficient and can produce a wide range of frequencies. Piezo Bender Actuators are used for tactile feedback and are made of piezoelectric material, typically ceramic or polymer. The deformation is proportional to the applied voltage. These actuators offer fast response time, low power consumption, and high precision.

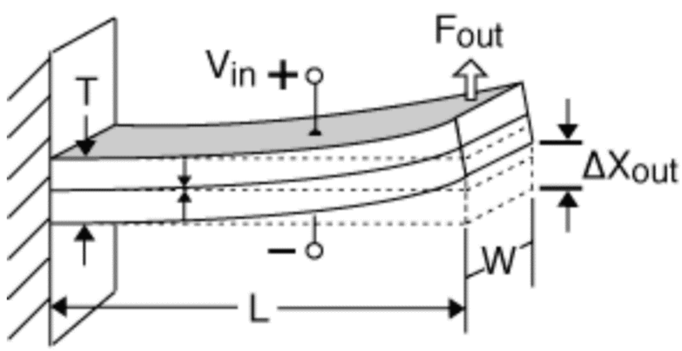

Figure 3.2 *- A simple mechanical diagram of a 2-layer bending mode piezo actuator (Motola-Barnes, 2019) [34]*

Stack actuators produce a larger overall displacement between piezoelectric material when a voltage is applied, causing a large force output. They are generally used in applications where substantial mechanical power is needed.

Shear piezoelectric actuators work based on the shear deformation effect. When an electric field is applied perpendicularly to the polarization direction of the piezoelectric material, the material deforms in a shear mode, causing lateral displacement. These actuators are capable of precise displacement with high accuracy, and their size allows for them to be integrated into smaller projects.

While shear piezoelectric actuators offer precise lateral displacement and fast response times, they are not well-suited for this smart helmet project. The design and integration of shear actuators would require specialized high-voltage drive electronics, which would increase the complexity, power consumption, and cost of the system. Furthermore, shear actuators are more prone to material fatigue over time, which could compromise the reliability of the system. Given these limitations, alternative haptic technologies, such as vibrotactile motors, which provide more noticeable and reliable feedback with simpler integration and lower power requirements, are preferred for the haptic feedback system in the smart helmet.

Note: Piezo bender actuators require careful handling due to the fragility of piezoelectric materials. The material is prone to breaking under mechanical stress.

Linear Resonant Actuators (LRAs)

LRAs operate based on the resonance phenomenon. The core components include a magnetic mass, a spring, and a housing. When an alternating current at the resonant frequency is applied, the magnetic mass oscillates along a single axis, producing vibrations. This resonant frequency is where the system's natural frequency aligns with the driving frequency, resulting in vibration generation. LRAs provide more precise and efficient feedback than ERMs, and are simpler to integrate, making them suitable for the smart helmet project. Furthermore, their small form factor makes them easy to integrate into various devices, including wearable technology.

LRA’s are driven by an alternating current at the resonant frequency, often controlled by Pulse Width Modulation (PWM) signals from microcontrollers. The driving voltage is low, typically at 2V. Due to the resonance peak, LRAs operate within 170 to 180 Hz.

Electrostatic Actuators

Electrostatic actuators provide haptic feedback via the friction between the user's skin and a conductive surface using electrostatic forces. This method creates tactile sensations such as texture or movement, enhancing user interaction in devices with touchscreens or other touch-sensitive surfaces. Electrostatic actuators operate on electro vibration; when an alternating voltage is applied, it creates an electrostatic force between the surface and the user’s finger which therefore generates a tactile sensation.

Electrostatic actuators typically hold these components: conductive electrodes which are layers that apply the electric field necessary to create electrostatic forces, insulation layers which are materials that separate and protect the electrodes and the user's finger and control circuits which generate and modulate the alternating voltage applied to the electrodes.

While electrostatic actuators offer significant advantages in terms of high-resolution feedback and low power consumption, their limited force output and dependency on surface properties may make them less suitable for the smart helmet project. The requirement for immediate feedback when activating turn signals makes the Linear Resonant Actuators (LRAs) a more desirable choice. LRAs can provide stronger and more consistent tactile feedback, which is important for ensuring that users receive clear and reliable signals while wearing the helmet.

3.1.1.2. Comparison of Haptic Technology

While both technologies can deliver tactile feedback, the ERM offers a simpler integration process and broader compatibility with driver ICs like the DRV2605L. Additionally, ERMs are cost-effective and provide adequate feedback intensity for our application. LRAs, while more precise and responsive, require a more complex drive circuit and are generally more expensive, which didn't align with the constraints of our design and budget. The ERM's ability to generate strong, noticeable vibrations made it a better fit for our project's requirements.

During our testing phase, we initially used the SparkFun Qwiic Haptic Driver - DA7280, a breakout board designed to simplify the integration of haptic feedback into projects. This setup included an LRA motor due to its compact size, low power consumption, and operating voltage range, with the board operating on I2C communication. However, after evaluating the performance and considering project requirements, we transitioned to using an ERM motor instead. The ERM provided adequate feedback and was able to integrate with our existing driver IC, the DRV2605L. While the LRA offered precision, its higher cost and additional driver complexity made it less practical for our application. Ultimately, the ERM proved to be the better fit for our final design.

Table 3.1 *– Linear Resonant Actuator Comparison*

Feature/Criteria	DA7280 LRA	Precision Microdrives 307-100	AAC 1027
Operating Voltage	2.7V to 5.5V	2.5V to 3.5V	3.0V to 3.6V
Resonant Frequency	175 Hz	175 Hz	235 Hz
Maximum Acceleration	1.0 G	0.8 G	1.2 G
Dimensions	10mm x 3.2mm x 2.0mm	10mm x 3.4mm x 2.0mm	8mm x 2.7mm x 2.4mm
Current Consumption	40mA	30mA	50mA
Response Time	<10 ms	<15 ms	<10 ms

Lifetime	1,000,000 cycles	500,000 cycles	1,500,000 cycles
Cost	Moderate	Low	High

Table 3.2 *– DA7280 Specifications*

Haptic Driver DA7280	
Communication	I2C
Frequency Tracking Support	300Hz
Supply Voltage	2.7V
Supply Current	360nA
Dimensions(board)	10mm x 3.2mm x 2.0mm

The DRV2605L was chosen for a variety of reasons such as having a wide operating voltage range which supports a voltage range from 2.0V to 5.2V, making it compatible with our 3.3V system, haptic waveform library which includes a pre-loaded library of haptic effects, and its I2C interface, simplifying integration with the ESP32 microcontroller. This interface allows for data transfer between the driver and the MCU.

The DRV2065L also has low power consumption, which is ideal for battery-operated systems, contributing to the overall power management strategy of the smart helmet.

Table 3.3 - *Haptic Driver Comparison*

Feature/Criteria	**DRV2605L**	**TI DRV2625**	**ADXL343**
Operating Voltage	2.0V - 5.2V	2.7V - 5.5V	2.0V - 3.6V
Haptic Actuator Support	LRA, ERM	LRA, ERM	LRA, ERM
Interface	I2C	I2C	I2C
Waveform Library	Pre-loaded library with over 100 effects	Pre-loaded library with over 123 effects	None
Low Power Consumption	Yes	Yes	Yes
Feedback Modes	Closed-loop and open-loop	Closed-loop and open-loop	Open-loop only

Special Features	Audio-to-haptics mode, real-time playback	Enhanced drive performance, LRA auto-resonance	Basic haptic control
Cost	Moderate	Higher	Lower
Dimensions	Compact	Compact	Compact
Ease of Integration	High (I2C, built-in library, low power)	High (I2C, enhanced features)	Moderate (limited to basic applications)

3.1.2. Touch Sensors

3.1.2.1. Capacitive Touch Sensors

Capacitive touch sensors are widely used for their ability to detect touch inputs with high sensitivity and reliability. Unlike mechanical switches, capacitive touch sensors can detect touch through non-conductive materials, making them ideal for seamless designs. They operate by measuring changes in capacitance caused by the presence of a conductive object, such as a human finger, near the sensor surface.

The sensor consists of conductive electrodes arranged in a pattern. These electrodes form a grid of capacitors with the surrounding environment. When a voltage is applied to these electrodes, an electric field is created around them. The electrodes have a certain baseline capacitance with respect to the ground which is the reference point for detecting changes.

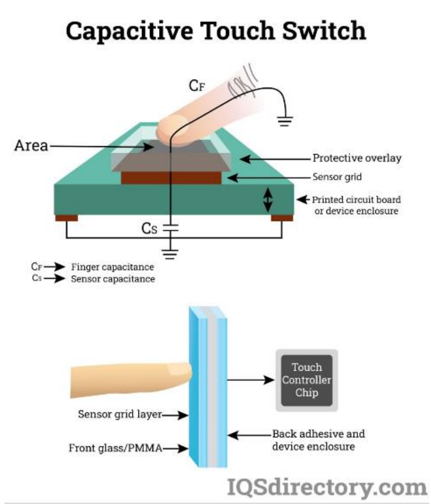

Figure 3.3 *- Industrial Quick Search Capacitive Touch Switch [11]*

When a finger touches the sensor surface, it introduces a conductive object into the electric field. The human body, being conductive, disturbs the electric field and increases the capacitance at the point of contact. This change in capacitance is detected by the sensor's controller, which processes the data to determine the touch location. When the capacitance passes a predefined threshold, the controller registers a touch event.

Capacitive touch sensors can detect even light touches, providing a highly responsive user experience. Yet it is important to note that the sensors can be affected by moisture, dirt, and other conductive materials, leading to false touches or interference.

3.1.2.2. Resistive Touch Sensors

Resistive touch sensors consist of two conductive layers separated by a thin gap. When pressure is applied to the surface, the two layers come into contact, changing the resistance at the point of touch. This change is measured to determine the touch coordinates. Under normal conditions, the two conductive layers are separated by the air gap, and there is no contact between them, resulting in an open circuit. When pressure is applied to the surface, the flexible top layer is pushed down and makes contact with the bottom layer, creating a voltage divider circuit, where the touch point's position can be determined by measuring the voltage. The sensor controller measures the resistance at the touch point and the exact coordinates of the touch can be determined.

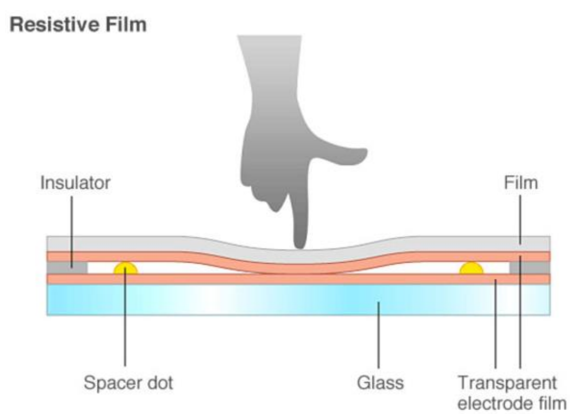

Figure 3.4 *- MIKROE Time-Saving Embedded Tools Resistive Film Touch Sensor [41]*

Resistive touch sensors are generally less expensive to produce than capacitive sensors and can be used with gloves, or other objects, which is particularly useful in environments where users may need to wear protective gear or use specialized tools. The design of resistive sensors makes them less susceptible to interference from water, dust, or other contaminants. However, it's important to note that resistive touch sensors are generally less effective than capacitive touch sensors. Light touches may not register effectively.

3.1.2.3. Comparison of Touch Sensors

After thorough consideration of the available touch sensor technologies, we have decided to use the TTP223 capacitive touch sensor for our smart helmet project. The capacitive touch sensor offers better sensitivity and responsiveness, which are crucial for the scope of the helmet. The capacitive sensor can be integrated into non-metallic surfaces such as plastic and glass and is also suitable for various environmental conditions, ensuring durability and longevity. Additionally, the low power consumption and adjustable sensitivity make it an efficient choice for our application.

***Table 3.3** - Resistive vs. Capacitive Touch Sensors*

	Resistive Touch Sensor (DF9-16)	Capacitive Touch Sensor (TTP223)
Operating Voltage	1V	2.5 ~5.5V DC
Thickness	0.4mm	1.35128mm
Trigger Force	20g, triggered when resistance is less than 200kΩ	0-50pF
Response Time	<10ms	60ms at fast mode, 220ms at low power mode @VDD=3V
Activation Time	<0.01s	Initial stable-time of 0.5s after power-on
Weight	1g/0.04oz	1.0 Ounce
Control Method	Pressure	Touch

The TTP223 is a single-channel capacitive touch sensor that offers self-calibration and low power consumption, making it ideal for portable and battery-powered applications such as the smart helmet. It supports both momentary and toggle touch modes and can be easily integrated. Compared to the other sensors, the TTP223 has the lowest current consumption.

***Table 3.4** – Capacitive Touch Sensor Comparison*

Feature/Criteria	**TTP223 Capacitive Touch Sensor**	**MPR121 Capacitive Touch Sensor**	**AT42QT1010 Capacitive Touch Sensor**
Channels	1	12	1
Operating Voltage	2.5V to 5.5V	1.71V to 3.6V	1.8V to 5.5V
Current Consumption	1.5µA (low power mode)	29µA	8µA (low power mode)
Interface	I2C	I2C	Digital Output
Touch Modes	Momentary, Toggle	Configurable via I2C	Momentary
Self-Calibration	Yes	Yes	Yes

Additional Features	Low power consumption, simple interface	Touch threshold configurable via I2C	Proximity sensing capability
Price	Low	Moderate	Low

3.1.3. Light Sensor

For HelmetIQ, we needed to incorporate an ambient light detection sensor for the main headlight of the helmet; the light sensor provides the information needed to facilitate intelligent illumination control. This system decides whether to activate the headlight, optimizing its use in varying light conditions.

3.1.3.1. Light Sensor Technology

Photoresistors

Photoresistors are composed of semiconductors and are light dependent. This kind of technology, which is known as Light-Dependent Resistors (LDR), are typically made from cadmium sulfide or lead sulfide. They rely on the radiation of light to trigger the movement of electrons within it, changing the current flow within. The higher the increase in the amount of light, the easier the flow of electrons; the higher the flow of electrons, the less resistance. This change in resistance is called photoconductivity and is the process by which electrons are excited and move from the valence band to the conduction band. These are simple and inexpensive but not very sensitive. They also can have resistance in total darkness; this is called dark resistance. They also have a very slow response time, as well as a small range of spectral responses, limiting their use.

Photodiodes

Photodiodes work by converting light into an electric current. These are usually made from p-n junctions, two different regions, p-type and n-type semiconductor regions. Both with different dopant concentrations that together create a built-in voltage. When light hits the photodiode, it causes an electron-hole pair to be formed. This difference creates a region with no mobile charge, called the depletion region. The pair cannot remain within this region, so it then moves the electron to the positive side and the hole to the negative side. In this process, the light is converted to an electrical current. This type of diode has better sensitivity than the photoresistor and faster response times.

Phototransistors

Phototransistors convert light energy into an electrical current, very similar to how photodiodes do. Although they are usually made from bi-polar junction transistors with two different options, n-p-n or p-n-p. Bi-polar junctions are composed of three layers. These layers are the base, collector, and emitter. Phototransistors can use the weak current generated by light, which acts as the base current, to amplify the current that corresponds to the transistor's current gain. This gives it the ability to see different intensities of light including lower light intensities than a photodiode. The ability to control the output current based on the input voltages makes phototransistors very versatile. Using a phototransistor would have a couple of advantages including higher sensitivity, simplicity of use, and cost-effectiveness. Yet, it has slower response times and is sensitive to temperature changes due to the changes in the current gain of the transistor.

Photomultipliers

Photomultipliers are generally phototubes that are extremely sensitive to light. Mostly used for scientific purposes and can amplify the weakest light signals. Photomultipliers are very fast and have a very wide spectral range. Photomultipliers work with a process called the photoelectric effect. This process is when light interacts with matter causing the release of electrons from the matter, or electron ejection. This occurs when the energy from the photo surpasses the energy binding the electrons to the material. The electrons being emitted by the matter are called photoelectrons. This process has two key factors that affect it including the wavelength of the light since not all wavelengths can cause electron ejection. The second key factor is the material. Certain materials have too high of a binding energy to allow the process to occur. This kind of sensor would be suboptimal, exceeding the necessary needs in information, complexity, and utility.

Table 3.5 *– Comparison of Light Sensor Technology*

Feature/Criteria	**Photoresistor**	**Photodiode**	**Phototransistor**	**Photomultipliers**
Speed	Slow	Fast	Faster than diodes	Very Fast
Detection Range	Visible Light	Visible, Near Infrared	Visible, Near Infrared	UV, Visible, Near Infrared
Response Speed	Slow	Fast	Moderate	Very High
Amplification	No	No	Yes	No
Sensitivity	Low	Moderate	Moderate	Very High
Voltage Production	No	Yes	No	Yes
Cost	Low	Moderate	Moderate	High

When considering the different types of technology for the light sensor, there needs to be a moderately fast sensor, usable in visible light, and moderately sensitive. Looking at both the phototransistor and the photomultiplier, these would exceed the requirements and cause unnecessary spending. Leaving only photoresistors and photodiodes as a viable option. The photoresistor for the most part will not provide the speed and sensitivity needed, making a photodiode the best choice.

3.1.3.2. VEML7700 High Accuracy Ambient Light Sensor

The first light sensor to go over is the VEML 7700. This is a high accuracy sensor that detects ambient lighting and provides a numerical lux value. Using this value, an average algorithm can be used to collect a range of values over a certain time period then take the average of those values. Using that average, a minimum threshold is used to determine when the headlight should become active. This works very well as most cyclists ride through parks and forests where the lighting is constantly fluctuating. By using the average value, this prevents the flashlight from turning itself on and off constantly. Extreme temperatures or overuse may cause the need for periodic calibrations to maintain accuracy and precision. This device is typically used for Arduino and will need the Adafruit VEML 7700 library to use. This creates a seamless integration with an Arduino or an ESP32 microcontroller.

Table 3.6 *- VEML 7700 Specifications*

VEML 7700 High Accuracy Ambient Light Sensor	
Communication	I2C
Resolution	0.0036lx/ct
Supply Voltage	3.3V or 5V
Supply Current	1.5mA
Dimensions (board)	17mm x 17mm x 4mm
Dimensions (sensor)	6.5mm x 2.35mm x 3.0mm
Dynamic Range	0 -120k lux
Visible Spectrum	400 - 700nm

3.1.3.3. GL55XX LDR Resistor Light-Dependent Photoresistor

Another option for the light sensor is to use a GL55XX Photoresistor. This sensor works when the photoresistor receives light, the current would flow by reducing the resistance and keep the headlight deactivated. This would be a very small and cost-effective device. The sensitivity can be chosen by selecting a specific resistance, it can provide high sensitivity, it's reliable, and responds very quickly. With those benefits, it does have some drawbacks. The main issue here is that with constant changes in the ambient light, the cyclists could experience inaccurate readings, causing a constantly flashing light. Using a photoresistor would also require the design of a circuit that would detect the resistance changes in the photoresistor to determine if the headlight would be active.

Table 3.*7 - GL55XX Specification*

GL55XX Specification (at 10 lux)	
Size	4.3mm x 32mm x 2.1 mm
Dark Resistance	1.0MΩ
Power Dissipation	100mW
Light resistance	8-20KΩ
Gamma value	0.7
Max Voltage	150V

3.1.3.4. SparkFun AS7262

A third option is to use the AS7262; this device is a 6-channel visible spectrum ID. That means this device is able to measure the light intensity of different colors and output values. It is integrated with an LED driver with programmable current. This device uses nano-optic deposited interference filters to help measure a different range of colors. This is done directly on standard CMOS silicon. Lastly it uses LGA packaging for precise light management; this helps to improve the signal by reducing noise interference. Unfortunately, it does not provide any actual lux values. Fortunately, using different calculations and by calibrating the data, we can output some lux values. This is a good option but is relatively expensive compared to the others. The AS7262 also has extra functions which are unnecessary to the helmet. These include the ability to break down light into its different colors as well as measure color intensity for each color, quantifying how much of each color there is and which of those is the dominant color.

Table 3.8- *SparkFun AS7262*

Spark AS7262	
Communication	I2C
Supply Voltage	2.7V to 3.6V
Analog-to-Digital Converter	16-bit
6-channel Spectrometry	450nm, 500nm, 550nm, 570nm, 600nm, 650nm
Dimensions	90mm x 50mm x 5mm

3.1.3.5. Adafruit TSL2591

This is an ultra-high-range luminosity sensor containing both infrared and full spectrum diodes on a CMOS integrated circuit. This device uses a wide range of digital sensors ranging from 188 microlux to 88000 Lux. It is double buffed to ensure the integrity of the data. It can go into different low-power sleep modes and is also very fast, with I2C up to 400kbits/s. The device can also set thresholds on the light sensor both high and low. It is more precise than older similar light sensors. It contains configurable gain, giving the user the option to optimize the sensitivity of the sensor. The sensor can also reject UV light which helps to make other readings such as in visible light more accurate. Includes programmable interrupts and integration time which give control of the measurement period of light intensity.

***Table 3.9** – Adafruit TSL2591 specifications*

Adafruit TSL2591	
Communication	I2C
Supply Voltage	3.3V to 5V
Dynamic Range	188µLux to 88,000Lux
Digital Conversion	16-bit
Dimensions	19mm x 16mm x 1mm

3.1.4. LED

When choosing the LEDs needed for the headlight of the helmet, there are a few different specifications that must be addressed. To get the right LED, we must have an LED bright enough to light the way at night in trails. To be able to do this, we need at least 800 to 1500 lumens. With this brightness, we must also consider the heat as well as the power consumption of the different options. Another factor when choosing the LED is the beam pattern of the LED. This component is simply what the light sensor determines whether it should be on or off depending on the values being received. The next LED is the brake light. On the HelmetIQ there are two brake lights on the back. These are red LEDs that must have 50- 100 lumens nowhere near as much as the headlight as they only need to be used as a signal and not to light a trail. The final LED needed is the turn signal. This LED like the brake light also serves as a signal needing only 50 to 100 lumens

3.1.4.1. LED Technology

LED DIP (Dual in-line Package)

This type of LED is cylindrical with a hemisphere on top. The inside structure is made up of a wire bond, a reflective cavity, a semiconductor die, an anvil, and a post. The DIP only emits a single color but comes in a range of colors. Typically, these are durable and relatively inexpensive. They are also bigger and designed with two parallel pins which make it easy for through-hole soldering which in turn allows them to be soldered by hand.

LED SMD (Surface Mounted Device)

Surface mounted Device LEDs are square or rectangular and are comprised of an LED chip, a tiny semiconductor chip, and bonding wires that connect everything, these are typically made up of gold. This is all encapsulated in a resin keeping everything safe from its environment and maintaining a very low profile. Lastly, a metal pad for everything to sit on and serves as connection points to a PCB. These types of LEDs are much smaller than DIPs and have much higher density. They are very reliable and can work with pick-and-place technology which makes them very fast to assemble.

LED COB (Chip on Board)

The Chip on Board LEDs have multiple LEDs mounted on a single substrate which helps to create a large concentration of LEDs. These LEDs use bonding wires, usually gold, for high conductivity. The COB is covered in a phosphorus layer; this layer is excited by the blue light that the chips emit and creates a white light. Finally, these are encapsulated in epoxy to protect it from its environment. COBs Provide high lumen output, and uniform light distributions all while still being compact. These come in a range of types including Standard, High-power, Mid-power, Integrated circuits, color mixing, and array. The downside to COB is they are relatively more expensive as well as harder to control the color.

***Table 3.10** – LED Technology Comparison*

Feature/Criteria	LED COB	LED SMD	LED DIP
Size	Varies	Smaller	Larger
Cost	High	Low	Low
Power Consumption	Typically Higher	Low	Low
Luminosity	Variable	10-900lm	10-100lm
Lumen Efficiency	80-150 lm/w	50-200 lm/w	20-80 lm/w

In the table above, there are a few different key factors to consider when deciding which kind of LED technology is the right pick for the HelmetIQ. Needing a headlight and signal lights leads to the selection of different types of LED. For the headlight due to the need for higher lumens, the best option is to use a COB LED, since it provides the higher lumen output required. For the brake and turn signals the best option is to use the SMD LED, as a lower lumen value is needed as well as to retain a lower profile.

3.1.4.2. Cree XP-G3 Warm White LED

These LEDs are optimized for situations in which a high lumen is needed. These LEDs work with the principles of semiconductors conducting electricity only under certain conditions. They can range in brightness from 100 - 800 lumens. This level of brightness would work great with the headlight as it requires a higher intensity of brightness. They have an excellent lifetime and color stability and have a very small footprint. Some other factors to also consider with this option are first, the heat management, as this type can produce lots of heat and needs another heat sink. It is also very important to keep the driving current at its proper level to keep the correct brightness and lifespan this can be done using an LED driver. One final aspect is that this LED has a wide beam angle which would be ideal for the HelmetIQ.

Table 3.11 *–Cree XP-G3 Specifications*

Cree XP-G3 Warm White LED	
Color	3000k
Max Current	200mA
Supply Voltage	2.83V
Dimensions	20mm x 20mm
Brightness	100 lumens

3.1.4.3. Wurth Elektronik LED 1206 SMD

This surface-mounted LED with a dome-like structure provides a low level of lumens perfect for signal and brake light. This LED is highly reliable with an MTBF rating meaning it has tens of thousands of hours that it can run. There are a few different ways these can come in. Tape and reel, which is an unmodified roll of continuous tape with leaders and trailers. Cut Tape is very similar to tape and reel but without a leader or trailer. Digi-reel comes with an 18-inch leader and trailer. In the structure of the tape and reel, there is an embossment, several chip cavities for the LEDS, and a sprocket hole. This helps a lot when in the assembly process as some are compatible with pick and place.

Table 3.12 *– Wurth Elektronik LED 1206 SMD Specifications*

Wurth Elektronik LED 1206 SMD	
Color	6000k
Max Current	20mA
Supply Voltage	2.00V
Dimensions	3.20mm x 1.6mm
Brightness	80 lumens

3.1.4.4. FD-5WSRGB-A Diffused RGB

One simple option for the headlight of the helmet is the use of RGB common anode or cathode diode. We can use multiple diodes that have been put together into one LED. There is a red, green, and blue diode altogether in one. By controlling the amount of current going in each of the different colors, it can be easily programmed to create a wide range of colors. This would be very useful for the brake, turn signals, and headlight. It would consume very little power and be very cheap. This option has multiple disadvantages, the main one being the brightness; One diode only has about 20 lumens which varies with color and does not meet the favorable beam pattern needed. To achieve the brightness needed for the different parts of the helmet, a large amount would be needed causing both the size to increase and raising the cost significantly.

Table 3.13 *– FD-5WSRGB-A*

FD-5WSRGB-A Diffused RGB	
Supply Voltage	1.8V to 3.6V
Forward Current	20mA
Reverse Voltage	5V
Reverse Current	10 µA
Power Dissipation	150mW
Dimensions	8.7mm x 1.0mm x 1.0mm
Brightness	20 lumens

3.1.4.5. PATIKIL 3W COB LED Strip

This LED is a COB LED or chip-on-board LED. This kind is comprised of multiple LED chips that are mounted to and make direct contact with the substrate that produces the LED arrays in the chip. To get the warm white color, the chips are covered in a layer of phosphorus usually cerium-doped YAG. Using this kind of chip has multiple advantages. One, they provide a very high-uniform, high-intensity light helping with shadows and hotspots. Using this method, it is also possible to group more together producing a higher lumen density as well as a higher output than regular LEDs. Another very important factor is these chips are usually made with materials such as ceramic or metal to help disperse the heat away from the chip, providing an Improved thermal performance. This also needs an LED driver circuit to maintain the correct current to extend the battery life and help with battery consumption. Overall, these LEDs help provide the brightness needed for a head light as well as the compactness and light weight needed for the helmet as well as still consuming less power.

Table 3.14 *– PATIKIL COB LED Strip Specifications*

PATIKIL COB LED Strip	
Supply Voltage	3.0V - 3.7V
Supply Current	1000mA
Color Temperature	6500K
Dimension	3.62mm x 2.4mm x 0.63mm
Brightness	300 lumens
Wattage	3W

3.1.4.6. Luminus SST-20-WxH

The SST- 20 is a high-lumen density SMD LED able to produce a significant amount of light some models exceeding 1000 lumens. Due to its monolithic design, the LED has very efficient thermal management. It has a very uniform light out that helps to achieve a consistent beam pattern. Many of these features work together to increase the lifespan and the total performance of the LED making it a very reliable option. The LED itself is mounted on a ceramic substrate, has a phosphorus layer to produce the right color, and is completely enclosed in a resin. It also has a wide range of colors that can be very beneficial to signal the light of the helmet.

Table 3.15 *– Luminus SST-20-WxH*

Luminus SST-20-WxH	
Supply Voltage	2.5V to 3.1V
Supply Current	.2A
Color Temperature	2700K, 3000K, 3500K, 4000K
Dimension	3.5mm x 3.5mm x 1.9mm
Brightness	120 Lm to 550Lm
Wattage	1.16W

3.1.4.7. Nichia NVSW219F-V1

This is an SMD LED with hard glass and silicon resin, with phosphor as the sheet material. This LED also contains ceramic packaging, a silicon resin lens, gold-plated electrodes, and a die heat sink. It can achieve very high brightness on the headlight of the helmet. It is very compact allowing smooth integrations. This LED has a long life span and low power consumption, allowing durable and reliable LEDs on the helmet. The direction of the light emitted helps to create a focused beam pattern. This LED also comes with a few different packaging options such as tape and cut which can help with assembly.

Table 3.16 *–Nichia NVSW219F-V1*

Nichia NVSW219F-V1	
Supply Voltage	2.7 - 3.3V
Supply Current	1800mA
Color Temperature	5700K
Dimension	2.3mm x 3.2mm x 3.2mm
Brightness	600 lumens
Wattage	6W

3.1.5. Accelerometer/Gyroscope

For the detection of the speed, acceleration, and orientation of the HelmetIQ, the components needed will need to be at minimum a 3-axis gyroscope, and an accelerometer MEMS device. Although these different sensors could be purchased separately to both minimize cost and decrease the amount of space taken up by them, there are the more commonly used options of chips that come with both. The purpose of this in the HelmetIQ serves to provide data on acceleration and orientation. This data is crucial as it provides speed, acceleration, and orientation. If there is an extreme disturbance in this sensor it also determine if a collision has occurred.

3.1.5.1. Microelectromechanical Systems (MEMS)

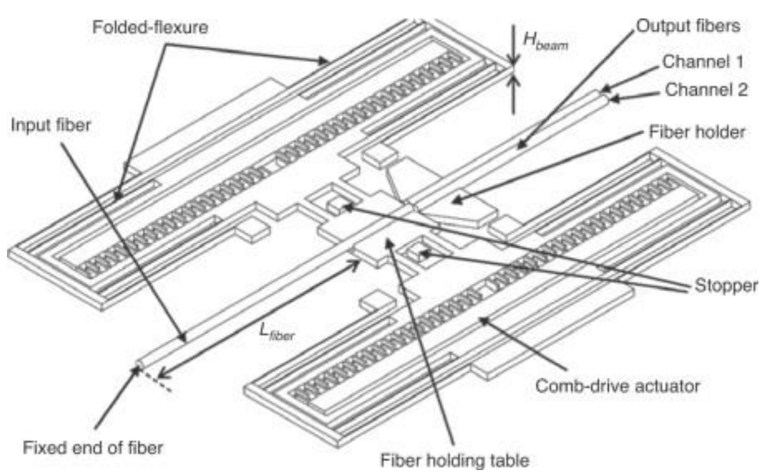

Figure 3.5 *- Image of the Comb-driver [39]*

Capacitive

This component is a MEMS device or microelectromechanical system. These devices work in several different ways, one being with a microscopic comb drive that is virtually suspended within the chip. The comb-like structure inside can move around, and change according to the inputs of its movements. These comb-drive movements are caused by external movements. These small changes within the chip change the capacitance within each comb-like structure. As these different changes occur, they are also categorized in the direction of the axis. These changes are measured and help collect the necessary data to calculate the acceleration, which can lead to finding the speed.

Piezoelectric

This kind of accelerometer relies on the use of the piezoelectric effect. The piezoelectric effect is the ability that piezoelectric materials have where an electrical charge is formed based on mechanical stress such as pushing or squeezing. In accelerometers, the accelerations that are occurring outside the device cause a force that with the piezoelectric effect is converted to voltage, this voltage is then measured. Since the force and the acceleration are proportional to each other, the acceleration can be calculated from the change in the voltage. In this process, there are three very important elements, the piezoelectric material, seismic mass, and the cantilever. The cantilever, relative to MEMS devices, is a long thin structure that is suspended. This acts as a spring and bends proportionally to the force that is applied. The seismic mass is a small weight made to resist acceleration and is attached to the end of the cantilever. This mass helps convert the physical force to an electrical signal. Overall, these kinds of accelerometers typically have very high bandwidth as well as high performance with no power consumption as it is a passive device.

Piezoresistive

This device works on the changes that occur to the piezoresistors also called the Piezoresistance Effect and is the basis of this sensor. This device is comprised of three major components the Seismic mass, cantilever, and piezo resistor. The seismic mass serves as a small yet heavy weight that resists motion and then exerts that force on the other components. The Cantilever is a long beam connected to the seismic mass which deforms when force is passed to it. The piezoresistor is a silicon-based resistor that deforms according to the different changes within the device. These changes from stress and strain cause the piezoresistor to deform, this reaction is called the Piezoresistance Effect. The changes to the resistance of the system also cause changes in the voltage; these changes are read and measured providing the acceleration. These devices are very dynamic, durable, and relatively low-cost. Unfortunately, they have low sensitivity and are very temperature dependent.

***Table 3.17** – Accelerometer Technology Comparision*

Feature/Criteria	Piezoelectric	Capacitive	Piezoresistive

Cost	Moderate	Moderate	Low
Sensitivity	1mV/g to 1000 mV/g	5mV/g to 50mV/g	.01mV/g to 10 mV/g
Temperature Dependency	High	Moderate	Very High
Power Consumption	1mW to 10mW	0.1mW to 5mW	Passive
Complexity	Moderate	High	Low
Working principle	Piezoelectric effect	Capacitance changes	Stain gauge

3.1.5.2. Gyroscope

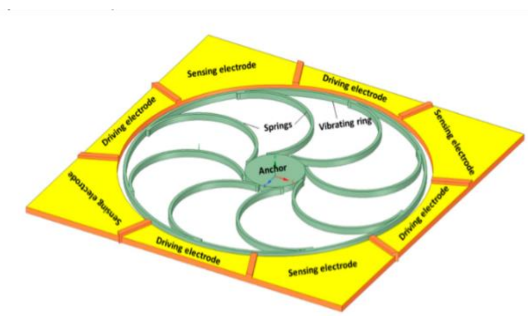

Figure 3.6*- Vibrating Ring MEMS gyroscopes [26]*

Vibrating Ring

Although there are a few common forms of a MEMs gyroscope, we are going to go over only a couple that pertain to our project. The first one is a Vibrating Ring Gyroscope (VRG). When the VRG experiences rotation, with the micromachined structures it can measure the rotational velocity. This is done using the Coriolis effect which creates a secondary vibration that is perpendicular to the original. Using the data received, the device can calculate the orientation. This device is shaped like a ring. Surrounding the outside of the ring are alternating drive electrodes as well as sensing electrodes. The inner side of the ring contains spring-like structures that connect to the ring and all meet at the center where an anchor is located. Electrical signals cause the ring to consistently vibrate; when these vibrations are interrupted by other vibrations different electrodes are altered. These changes in vibrations are measured and are what determines both the rate and direction of rotation.

Gimbaled

The Gimbaled Gyroscope is comprised of two major components, the MEMS Sensing element and the gimbal structure it is suspended in. The gimbal structure, configured as either a dual or single axle, serves to keep the sensing element isolated and away from external influences. With this structure, it helps to increase the gyroscope's accuracy and stability. The MEMS sensing element is driven into resonance with an electric field, at this point the Coriolis Force acts on the vibrating device causing displacement. This displacement is measured by capacitive or piezoelectric sensors and then used to determine the angular velocity. Gimbaled gyroscopes typically have reduced noise, increased durability, a wide range of uses, and customizable configurations.

Tuning-fork

A tuning fork gyroscope is a single-axis gyroscope, able to only measure rotations along a single axis. The device uses a structure that is similar to a tuning fork but with an anchor point. It is used similarly to a tuning fork as different vibrations move through the device. The tuning forks are driven into a constant vibration at their resonance frequency, accomplished by an electric field. At this state, once the device rotates in any direction the Coriolis force causes the tuning fork to move in different directions. These different movements cause changes to the voltage in the device through the piezoelectric effect. Since these voltages are proportional to angular velocity they can be used to determine and measure precise rotational rates.

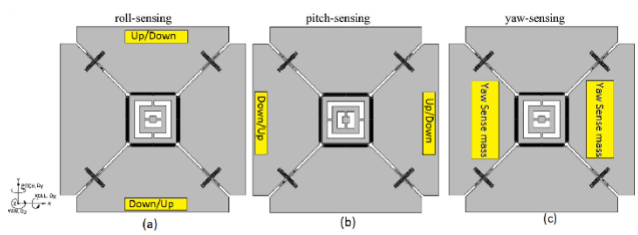

Figure 3.7*- Image of MEMS Multi-Axis [17]*

Multi-Axis

Multi-axis gyroscopes work by measuring the different angular rate changes in the different axes, typically X, Y, and Z. Three axes are the most common, but more than that can be used. With 3 axes, there are 6 different ways the parts within can move. These are typically multiple sets of capacitive plates that all vibrate at resonance. With electrodes on the end of the plates, once the Coriolis force begins to affect it, the capacitance begins to change. These different changes in the capacitance, which are proportional to the angular rate, are measured and then used to calculate the orientation of the device.

Table 3.18 *– Gyroscope Technology Comparision*

	Vibrating ring	**Multi-Axis**	**Tuning-fork**	**Gimbaled**
Power Consumption	.01mW to 1mW	.02mW to 5mW	.01mW to2mW	1mW to 10mW

Cost	Moderate	Very High	Moderate	High
Bandwidth	Moderate	Variable	Moderate	High
Sensitivity	50 deg/s to 500deg/s	50 deg/s to 2000deg/s	10 deg/s to 1000deg/s	.01 deg/s to 100deg/s
Complexity	Moderate	Very High	Moderate	High

Looking at the different technologies used in gyroscopes many of the different parameters can vary significantly based on the manufacturer of the device. First, we discuss using a gimbaled gyroscope this choice would be excessive and more expensive as a simpler device can meet the needs. A tuning fork is a good choice and very popular in the market with multiple options. The best option in this case is to go with a multi-axis gyroscope since it has the sensitivity needed to properly detect a collision from multiple angles. Selecting this also allows the option to choose between a variable number of axes in the device.

3.1.5.3. ICM-42670- TDK Accelerometer Gyroscope Module

The first gyroscope accelerometer component to review is the ICM-42670 3-axis. This device comprises of 6 axes, 3 for the gyroscope part, and 3 for the accelerometer and digital motion processor (DMP). It also has 3, 16-bit analog-to-digital converters, digitizing the outputs from both the gyroscope and the accelerometer. This low-power consumption device is very important due to its limited power supply. The device is also small and very light which helps with the limited space and the comfort of the HelmetIQ. Lastly, it is also relatively cheap, contributing to the total affordability. Some of the disadvantages of this device are the limited data and the accuracy. The chips have decent accuracy and should meet the requirement, but higher accuracy could remove doubts about performing correctly.

***Table 3.19**- Specifications for ICM-42670*

ICM-42670- TDK Accelerometer Gyroscope Module Accelerometer	
Communication	I2C,SPI
Supply Voltage	2.375V to 3.46V
Gyroscope range	±250, ± 500, ±1000, ±2000 dps
Accelerometer Range	±2g, ±4g, ±8g, ±16g
Supply Current	3.6mA

Dimensions	21.2mm x 16.4mm x 3.3mm

3.1.5.4. Adafruit BNO055

This device contains 9 axes, 3 for the accelerometer, 3 for a gyroscope, and 3 for a magnetometer. It has a 32-bit high-speed ARM cortex M0-based processor, that provides valuable and meaningful data. This unit has a triaxial, 16-bit gyroscope, which ranges up to 2000 degrees per second. BNO055 focuses mainly on obtaining the orientations. It uses the gyroscope, accelerometer, and magnetometer all together to have the data be as accurate as possible. It does so by using an onboard sensor fusion algorithm. One major downfall of this device is that although it can work with ESP32, some implementations violate the I2C protocol, which causes situations in which it might not be as reliable as it should be.

Table 3.20 *- Specifications for Adafruit BNO055*

Adafruit BNO055	
Communication	I2C,SPI
Supply Voltage	2.4V to 3.6V
Supply Current	6.5mA to 13mA
Gyroscope Range	±150, ±2000 dps
Accelerometer Range	±2g, ±4g, ±8g, ±16g
Dimensions	20mm x 27mm x 4mm

3.1.5.5 LSM6DS33 3D accelerometer and 3D gyroscope

The LSM6DS33 comes with a high-performance 3-axis digital gyroscope and a 3-axis digital accelerometer. This device is one of the best to be able to detect motion. It is very high performing while still maintaining low power consumption. It has ultra-low noise performance to provide a higher level of accuracy. It has an "always-on" mode that also has very low power consumption, as well as a temperature sensor. This device uses two main technologies including MEMS and CMOS.

Table 3.21*- Specifications for LSM6D33*

LSM6D33	
Communication	I2C,SPI
Supply Voltage	1.71V to 3.6V

Gyroscope range	±125, ±245, ± 500, ±1000, ±2000 dps
Accelerometer Range	±2g, ±4g, ±8g, ±16g
Dimensions	3mm x 3mm x .86mm

3.1.6. Battery and Power Management

Power management is a critical component of the smart helmet project, ensuring that all electronic components, including sensors, LEDs, gyroscopes, accelerometers, GPS, and Bluetooth modules, function reliably and efficiently. The choice of battery and power management strategy directly affects the performance, usability, and safety of the smart helmet.

Table 3.22*- Differences between Li-Ion, LiPo, and NiMH Rechargeable Batteries (ElectronicsHub, U.S.A)*

Feature/Criteria	Lithium-Ion (Li-Ion)	Lithium Polymer (LiPo)	Nickel-Metal Hydride (NiMH)
Energy Density	High (150-200 Wh/kg)	High (150-200 Wh/kg)	Moderate (60-120 Wh/kg)
Self-Discharge Rate	0.5-1% per month	5% per month	13.9%-70% per month
Cycle Life	500-1000 cycles	300-500 cycles	300-500 cycles
Voltage per Cell	3.6-3.7V	3.6-3.7V	1.2V
Charge and Discharge current	0.5-1C	1-5C	0.1-1C
Safety	Protection circuit needed	Prone to swelling, overheating	Minimal Risk

3.1.6.1. Lithium-Ion (Li-Ion) Batteries

Lithium-Ion batteries are commonly used in portable electronics due to their high energy density and long cycle life. They consist of a cathode, anode, and an electrolyte that allows for the movement of lithium ions between the electrodes during charging and discharging cycles. Li-ion batteries have a low self-discharge rate per month, which means they retain their charge for longer when not in use.

Li-Ion batteries can store a large amount of energy relative to their size and weight and have a longer cycle life compared to other types. Though, to prevent overcharging, over-discharging, and overheating, Li-Ion batteries require a protection circuit which is usually built-in. For this reason, Li-Ion batteries are implemented into the project.

3.1.6.2. Lithium Polymer (LiPo) Batteries

LiPo batteries are a type of Li-Ion battery with a flexible polymer electrolyte, allowing for varied shapes and sizes. This flexibility makes them suitable for applications where space constraints and form factor are critical. They provide high energy storage and are lightweight, making it ideal for portable applications.

Though, LiPo batteries are prone to overheating, swelling, and are more sensitive to overcharging and discharging, causing it to have potential safety hazards such as fires or explosions.

3.1.6.3. Nickel-Metal Hydride (NiMH) Batteries

The NiMH battery uses nickel oxyhydroxide for the positive electrode, and a hydrogen-absorbing alloy for the negative electrode. These work together making it a strong, rechargeable battery. Compared to other batteries, it falls around the middle in terms of weight.

NiMH batteries offer high battery capacity when compared to other alkaline batteries of the same size, leading to longer battery life. It has a low self-discharge which contributes to the lifespan and total battery charging capabilities, can handle a very wide range of temperature changes, including both hot and cold temperatures. Though, the batteries store less energy per unit weight and has a slower charging time.

3.1.7. Battery Management System

A Battery Management System (BMS) is crucial for the smart helmet project, ensuring the battery's safe, efficient, and reliable operation and all connected components.

Lithium-based batteries, such as Li-Ion and LiPo, pose significant risks if over-charged. Overcharging can lead to overheating, battery swelling, or thermal runaway, which can cause fires or explosions. Excessive discharge of a battery causes irreversible damage, reducing its capacity and lifespan significantly. Proper management prevents the battery from discharging below a safe voltage threshold.

Short circuits can result in excessive current flow, leading to overheating and potential damage to the battery and connected electronics. This necessitates protection mechanisms to detect and interrupt short circuits. Drawing excessive current from the battery can cause overheating and reduce the battery's operational life. Overcurrent protection ensures that current levels remain within safe limits.

The BMS uses these protection mechanisms to maintain battery health and performance, such that the smart helmet operates safely and efficiently. Implementation of a BMS is essential for eliminating risks associated with lithium-based batteries, enhancing the reliability of the project.

3.1.7.1. Basic Battery Management System

A basic BMS provides essential protection features necessary to ensure the safe operation of battery packs. These features typically include: overcharge protection which prevents the battery from exceeding its maximum voltage, potentially leading to overheating other hazards, over-discharge protection which ensures the battery does not drop below a safe voltage threshold, and short circuit protection which detects and stops excessive current flow to protect the battery and connected electronics from damage.

3.1.7.2. Advanced Battery Management System

Advanced BMS units offer enhanced functionalities beyond basic protection, including better monitoring and control features. These systems are designed for high-performance applications where battery health and efficiency are critical to the system.

Advanced battery management systems include everything from a basic battery management system and also integrates: cell balancing which ensures that all cells in a battery pack maintain equal charge levels, improving battery life, temperature monitoring which uses temperature sensors to prevent overheating and ensure safe operation across different environmental conditions, State of Charge (SOC) estimation which provides accurate information on the remaining battery capacity, helping manage power usage effectively, includes real-time monitoring and control through protocols like I2C, SPI, or CAN bus, enabling integration with microcontrollers and other smart systems, and data logging which records data on battery performance and usage, useful for diagnostics and optimization.

Table 3.23 *- Comparison of Basic and Advanced BMS*

Feature/Criteria	Basic BMS	Advanced BMS
Overcharge Protection	Yes	Yes
Over-Discharge Protection	Yes	Yes
Short Circuit Protection	Yes	Yes
Overcurrent Protection	Yes	Yes
Cell Balancing	No	Yes
Temperature Monitoring	No	Yes
State of Charge (SOC) Estimation	No	Yes
Communication Interface	No	Limited
Data Logging	No	No

Remote Monitoring	No	No
Cost	Low	Moderate
Ease of Integration	High	Moderate
Applications	Low-power, basic devices	High-capacity and complex systems

For the smart helmet project, a basic Battery Management System (BMS) can incorporate protection features to ensure the safe operation of the lithium-ion battery. Basic BMS units are designed to handle fundamental protection tasks without the complexity and cost associated with advanced BMS systems.

3.1.7.3. Basic BMS Comparison

For the smart helmet project, the Daly BMS 3S 12V is recommended due to its comprehensive protection features, ease of integration, and suitability for low to moderate power applications. The Daly BMS unit ensure the safe operation of the lithium-ion battery by preventing overcharge, over-discharge, short circuits, and overcurrent conditions, while also being relatively small and easy to integrate, making it ideal choices for the project.

Table 3.24 *- Comparison of BMS*

Feature/Criteria	JBD-SP04S001 4S 12V BMS	TP4056 BMS Module	Daly BMS 3S 12V
Overcharge Protection	Yes	Yes	Yes
Over-Discharge Protection	Yes	Yes	Yes
Short Circuit Protection	Yes	Yes	Yes
Overcurrent Protection	Yes	No	Yes
Size	65mm x 45mm x 10mm	25mm x 19mm x 10mm	55mm x 40mm x 8mm
Max Current	60A	1A	60A
Voltage Range	12V	5V	12V
Cost	Low	Very Low	Moderate
Ease of Integration	High	High	High

Applications	Small battery packs, portable devices, e-bikes	DIY projects, small portable devices	Portable devices, small solar systems

3.1.8. Voltage Regulators

Different components in the smart helmet, such as touch sensors, LEDs, and microcontrollers, require different operating voltages. Since the selected battery is above the 3.3V requirement for the sensors and the microcontroller, a buck converter is used to step-down the voltage from the battery so that the sensors can operate at the required voltage.

Voltage regulators are critical for maintaining stable voltage, ensuring components receive a constant voltage despite fluctuations in the input voltage, and preventing damage from voltage spikes or drops.

3.1.8.1. Boost Converters

A boost converter is a type of DC-DC converter that steps up a lower input voltage to a higher output voltage. It is widely used in applications requiring an increase in voltage, such as battery-powered devices, LED drivers, and power supplies for portable electronics. It operates by storing energy in an inductor during one phase of operation and releasing it to the output in another phase, resulting in a higher output voltage.

The key components of a boost converter includes: an inductor, which stores energy when current flows through it and releases energy when the current flow is interrupted, a switch, typically a transistor (MOSFET) which controls the flow of current through the inductor, a diode which ensures current flows in the correct direction, preventing backflow from the output, and a capacitor which smooths out the voltage at the output to provide a stable DC voltage.

When the switch of the boost converter is closed the current flows through the inductor, storing energy in its magnetic field. During this phase, the diode is reverse-biased, and no current flows to the output. When the switch opens, the circuit breaks, and the inductor releases its stored energy. This causes the inductor's voltage to reverse, forward-biasing the diode and allowing current to flow to the output capacitor and load. The voltage across the inductor and the input voltage combine to create a higher output voltage.

Boost Converter (Step-Up):

1. Inductor (L): Placed in series with the input voltage.
2. Switch (S): Connected between the inductor and ground.
3. Diode (D): Connected between the inductor-output junction and the output capacitor.
4. Capacitor (C): Connected across the output to smooth the voltage.

Due to the nature of the smart helmet project, most sensors and the microcontroller operate at voltages lower than the 3.7V provided by the chosen Li-Ion battery. Therefore, a boost converter, which steps up the voltage, is unnecessary. Instead, a buck converter is more appropriate as it steps down the battery voltage to match the requirements of the sensors and microcontroller.

3.1.8.2. Buck Converters

A buck converter is a type of DC-DC converter that steps down a higher input voltage to a lower output voltage. It is widely used in applications requiring a reduction in voltage, such as powering microcontrollers and sensors from higher voltage sources like batteries.

The buck converter operates by storing energy in an inductor when the switch is on and releasing it to the output when the switch is off, resulting in a lower output voltage. The main components includes: an inductor, a capacitor, a diode, and a switch, for similar reasons as a boost converter.

When the switch is closed, the current flows through the inductor, storing energy in its magnetic field. During this phase, the diode is reverse-biased, and no current flows through it. When the switch is opened, the inductor releasing its stored energy. This forward-biases the diode, allowing current to flow to the output capacitor and load. The voltage across the inductor and the input voltage combine to create a lower output voltage.

Buck Converter (Step-Down)

1. Inductor (L): Placed in series with the input voltage.
2. Switch (S): Connected between the input voltage and the inductor.
3. Diode (D): Connected between the inductor and ground (reverse-biased when the switch is closed).
4. Capacitor (C): Connected across the load to smooth the output voltage.

For the smart helmet project, a buck converter is used to step down the 3.7V output from the chosen Li-Ion battery to the 3.3V required by most sensors and the microcontroller. This selection is based on the need to reduce the input voltage to match the operating voltage of the components.

3.1.8.3. Linear Voltage Regulator

Linear regulators work by adjusting the resistance between the input and output to maintain a stable output voltage. They use a feedback control mechanism to compare the output voltage with a reference voltage and adjust the pass element's resistance accordingly. The key components that make up a linear voltage regulator include: a transistor, typically a Bipolar Junction Transistor (BJT) or a Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) that acts as a variable resistor, a reference voltage, which is a stable voltage source typically provided by the bandgap reference, which remains constant, an error amplifier which compares the output voltage with the reference voltage and controls the pass element to maintain the desired output voltage, and a feedback network, which is a resistive divider that feeds back a portion of the output voltage to the error amplifier.

The error amplifier continuously adjusts the pass element to maintain the output voltage equal to the reference voltage. If the output voltage rises above the desired value, the error amplifier reduces the conduction of the pass element, lowering the output voltage. If the output voltage drops, the error amplifier increases the conduction of the pass element, raising the output voltage.

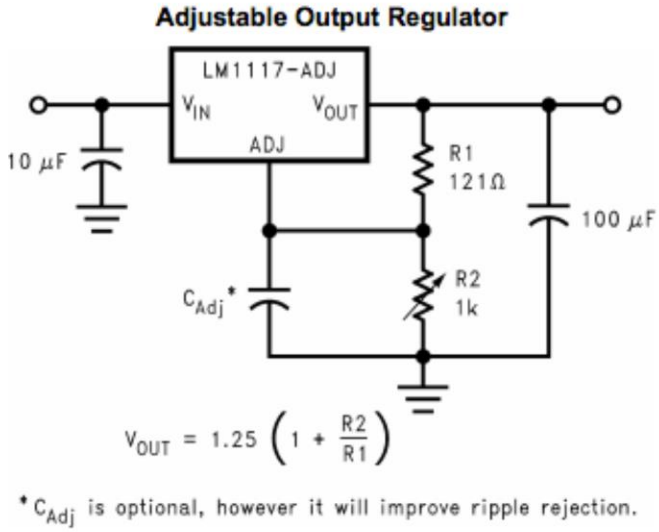

Figure 3.8 - *Linear Voltage Regulator Reference Circuit [52]*

3.1.8.4. Switching Voltage Regulator

Switching regulators control the energy transfer by quickly switching a transistor on and off. The duty cycle (ratio of on-time to the total cycle time) determines the output voltage. Energy is stored in inductors or capacitors during the on phase and released to the load during the off phase. The key components in a switching regulator include: an inductor, which stores energy when current flows through it and releases it when the current flow is interrupted, a switch (usually a MOSFET) which rapidly switches on and off, a diode which ensures current flow is in the correct direction, a capacitor which smooths out the output voltage and provides a stable DC voltage, and a controller IC which manages the switching of the transistor and helps regulate the output voltage.

When the switch is closed, the current flows through the inductor, storing energy in its magnetic field. When the switch is open, the stored energy in the inductor goes to the output through the diode, smoothing capacitor, and load. Switching regulators are highly efficient, typically between 80% and 95%, as they dissipate less power as heat compared to linear regulators and do not need a heatsink to regulate temperature, as linear voltage regulator needs.

For the smart helmet project, a buck converter is used to efficiently step down the 3.7V from the Li-Ion battery to the required 3.3V for the sensors and microcontroller. As a type of switching regulator, the buck converter operates by rapidly switching on and off, using an inductor, capacitor, and switch (typically a transistor) to convert the higher input voltage to a stable, lower output voltage.

Table 3.25 - *Voltage Regulator Comparison*

Feature/Criteria	Linear Voltage Regulators	Switching Voltage Regulators
Operation Principle	Adjusts resistance to maintain output voltage by dissipating excess power as heat.	Uses high-frequency switching to transfer energy efficiently.
Efficiency	Low (typically 30-60%) due to continuous power dissipation.	High (typically 80-95%) due to minimal power dissipation.
Heat Generation	Significant, requiring heat sinks for thermal management.	Minimal, reducing the need for extensive cooling.
Response Time	Fast, with minimal delay in voltage regulation.	Fast, but may have slight delays due to switching.
Size	Larger due to heat dissipation components.	Generally smaller, but depends on external components like inductors.
Cost	Typically lower cost.	Higher cost due to complexity and component count.
Applications	Low-power, noise-sensitive applications (e.g., analog circuits, RF devices).	High-power, efficiency-critical applications (e.g., battery-powered devices, computing equipment).

For the smart bike helmet project, the TPS62162 buck regulator was selected over the LM117DT-3V linear voltage regulator due to efficiency concerns. The buck regulator achieves much higher efficiency by stepping down voltage with minimal power loss, making it well-suited for our application where energy efficiency and thermal management are critical. The linear regulator dissipates excess voltage as heat, which leads to significant inefficiencies when stepping down from 7.4V to 3.3V. This inefficiency would not only drain the battery faster but also introduce unnecessary thermal challenges, which are undesirable in the helmet.

3.1.9. MCU

In selecting the optimal microcontroller unit (MCU) for the HelmetIQ project, we evaluated three prominent options: the Texas Instruments CC2652R LaunchPad, the Arduino Nano 33 BLE Sense, and the ESP32-WROOM. Each of these development boards offers unique features that could potentially serve our smart bike helmet's requirements.

3.1.9.1. Texas Instruments: CC2652R LaunchPad

The Texas Instruments CC2652R Launchpad is a highly capable development board that offers a range of features suitable for the HelmetIQ project. With its powerful Arm Cortex-M4F processor running at 48 MHz and 352 KB of flash memory, this board provides ample processing power and storage for the helmet's various functionalities. The CC2652R Launchpad also includes a wide array of peripherals, such as I2C, SPI, UART, and GPIO, which simplifies the integration of sensors, lights, and other components. Additionally, the board supports Bluetooth 5.1 and Zigbee 3.0 wireless connectivity, enabling seamless communication between the helmet and the mobile app. The CC2652R Launchpad's low power consumption and energy-efficient design make it an attractive option for battery-powered applications like the HelmetIQ. However, it is worth noting that the board's relatively higher cost compared to other options and the need for familiarity with Texas Instruments' development ecosystem may be factors to consider when making a final decision.

3.1.9.2. Arduino Nano 33 BLE Sense

The Arduino Nano 33 BLE Sense is a compact and feature-rich development board that could be a strong contender for the HelmetIQ project. Powered by a 64 MHz Arm Cortex-M4 processor and equipped with 256 KB of flash memory, this board offers sufficient processing power and storage for the helmet's various features. One of the standout aspects of the Arduino Nano 33 BLE Sense is its integrated sensors, which include an accelerometer and gyroscope. These built-in sensors can greatly simplify the implementation of collision detection and other safety features. The board also supports Bluetooth Low Energy (BLE) connectivity, allowing for easy integration with the mobile app. Additionally, the Arduino ecosystem is known for its extensive library support and user-friendly development environment, which can expedite the development process. However, the Arduino Nano 33 BLE Sense may face limitations in terms of expandability, as it has a limited number of I/O pins compared to other boards. This could potentially restrict the number of external components that can be connected to the helmet.

3.1.9.3. ESP32

The ESP32-WROOM is a versatile and cost-effective development board that presents a compelling option for the HelmetIQ project. This board has a dual-core Xtensa LX6 microprocessor running at 240 MHz, providing plenty of processing power to handle the helmet's various features. With 4 MB of flash memory, the ESP32-WROOM offers good storage capacity for firmware and data. One of the key advantages of this board is its integrated Wi-Fi and Bluetooth connectivity, which enables communication between the helmet and the mobile app without the need for additional modules. The ESP32-WROOM also features a wide range of I/O options, including SPI, I2C, UART, and numerous GPIO pins, making it easy to connect and control the helmet's sensors, lights, and other components. Moreover, the ESP32 development ecosystem has grown significantly in recent years, with extensive documentation, libraries, and community support available. However, it is important to consider that the ESP32-WROOM may consume more power compared to other low-power alternatives, which could impact battery life in a portable application like the HelmetIQ.

Table 3.26 *- MCU Comparison*

	TI CC2652R Launchpad	Arduino Nano 22 BLE Sense	ESP32-WROOM
Processor	ARM Cortex-M4F 48MHz	ARM Cortex-M4 64 MHz	Dual-core Xtensa LX6 240 MHz
Flash Memory	352 KB	256 KB	4MB
Integrated Sensors	No	Accelerometer, Gyroscope	No

Wireless Connectivity	Bluetooth 5.1, Zigbee 3.0	Bluetooth Low Energy (BLE)	Wifi, Bluetooth
I/O Options	I2C, SPI, UART, GPIO	Limited I/O pins	I2C, SPI, UART, GPIO
Development Ecosystem	Moderate	Extensive	Extensive
Library Support	Moderate	Extensive	Extensive
Community Support	Moderate	Large	Large
Cost	Higher	Moderate	Low
Ease of Use	Moderate	High	High
Expandability	High	Limited	High
Energy Efficiency	High	Moderate	Moderate

3.1.10. MCU Development Languages

To program an MCU for this project we researched a variety of high-level languages that allow us to implement sensors to a variety of different MCU chips. The language selection is important because things like overhead, memory usage and speed are critical aspects of our real-time system that can be determined through the selected MCU language. The above researched MCU boards support a variety of languages but most popular are C, C++, MicroPython, and Lua. These languages are the ones we decided to focus our research on.

3.1.10.1. C/C++

C and C++ are some of the most popular programming languages for MCUs. The main difference between these two languages is that C++ is an object-oriented language while C is not. C++ is generally preferable over C because it allows for low-level hardware control while still using high-level programming structures for logic. Since most programmers are familiar with high-level programming, C++ allows more flexibility and availability of libraries to access, read, and modify low-level hardware. C++ has a massive community of embedded system developers, and this is beneficial to us because there are many resources and documentation to help with any bugs or problems we may encounter.

The object-oriented nature of C++ allows us to manage our codebase in an efficient way using header files and library imports. In some cases, hardware manufactures build libraries in C++ that make implementing their hardware more efficient. Other open-source libraries like FreeRTOS are built by a community that allow for real-time operations. Since our system is technically a real-time system access to the FreeRTOS ecosystem is an extremely asset to using C++.

C is beneficial in a system where you need individual register access. It allows for much controlled and specific setup of serial communication methods like I2C and SPI. This setup requires knowing the specific registers of the MCU that the data is being fed to and then using these registers in any data reading, writing, or manipulation. This can make things more complex and the code less readable. C also makes it very difficult to allow parallel processes to be executing at once. Since our system is real-time, we have multiple tasks running at once so having parallel processes is important to project success. C allows for very specific memory management in systems where memory is scarce and every bit matters. Depending on the MCU selected this capability could be a potential benefit/necessity.

3.1.10.2. MicroPython

Python is infamously known for its simplicity and its broad assortment of libraries. MicroPython is an offshoot of Python that is specifically optimized for programming MCUs. It maintains the high-level nature of a language like Python, and this makes its readability and simplicity unmatched. MicroPython unlocks capabilities not found in Python by including modules that allow the programmer to access low-level hardware. These can all be handled through high-level libraries but the having the flexibility to access the low-level hardware when necessary, gives it a great advantage over using standard Python.

MicroPython generally performs worse that C/C++ because of its increased overhead. This makes MicroPython slightly less suitable for a real-time system. Also, the included/necessary libraries may take up too much memory on a hardware restricted system so depending on the MCU selected MicroPython may not be a possibility. MicroPython is not the best language if we want to derive maximum performance from our system, but it is the best if we want a simplified development process.

3.1.10.3. Lua

Lua is a more niche high-level language that is compatible with almost all MCUs. The reason Lua is popular to some programmers is because of its efficient scripting techniques. Lua is not limited to MCU development, and it is a fully capable language for developing web and mobile based applications. For our project Lua brings the main benefit of speed. The language is developed for performance and is the fastest language of all compiled scripting languages. This speed is an obvious benefit for our real-time system. Additionally, it is compatible with all C compilers which is why it is also compatible with nearly all MCUs.

Lua has a unique advantage for embedded development which is that because of its great performance, it consequently is good for hardware constrained systems. If our project decides to choose an MCU that is hardware scarce then Lua may be the best language for us to take advantage of the hardware. Lua is also compatible with libraries that are written in C/C++ so it holds the same benefits of outside documentation. The main issue for our group with this language is that the syntax is unfamiliar as well as the implementation process. Additionally, Lua is not an extremely popular language on its own so finding certain debugging documentation would be extremely difficult. Lua's interpreted nature may limit its performance in scenarios demanding high-speed processing and real-time responses. However, Lua's efficiency and ease of use make it a viable choice for projects that don't require the highest level of performance and can benefit from the flexibility and rapid development offered.

Table 3.27 *- MCU Language Comparison*

Features	C	C++	MicroPython	Lua
Object-Oriented	No	Yes	No	No
Low-Level Hardware Control	Yes	Yes	Limited	Limited
High-Level Programming Structures	No	Yes	Yes	Yes
Library Availability	Moderate	Extensive	Moderate	Moderate
Community Support	Moderate	Extensive	Moderate	Limited
Ease of Use	Complex	Moderate	Easy	Easy
Real-Time Operation	Limited	Yes	Limited	Limited
Memory Management	Detailed	Moderate	Abstracted	Abstracted
Performance	High	High	Moderate	High
Development Speed	Slow	Moderate	Fast	Fast
Parallel Processing	Difficult	Yes	Limited	Limited
Compatibility with MCUs	High	High	Moderate	High
FreeRTOS	Yes	Yes	No	No
Flexibility	Low	High	High	High
Register Access	Yes	Yes	Limited	Limited
Readability	Low	Moderate	High	High
Real-time System Suitability	Low	High	Moderate	Moderate

3.1.11. Mobile Development Languages

3.1.11.1. Kotlin and Java (Android)

Kotlin and Java are the primary languages for Android app development. Kotlin, a modern and expressive language, has become the preferred choice for many developers due to its improvements over Java, such as null safety, data classes, and coroutines for asynchronous programming. These features make Kotlin an excellent choice for developing robust and maintainable Android applications.

Kotlin's concise and expressive syntax reduces boilerplate code and improves code readability. Kotlin prioritizes developer productivity, user interface design, and integration with extensive software libraries and services. It provides features like garbage collection and runtime environments that abstract away low-level details, allowing developers to focus on building user-facing features and ensuring a smooth user experience.

3.1.11.2. Swift and Objective-C (iOS)

Swift and Objective-C are the main languages for iOS development. Swift is more concise and safer than Objective-C, which is an older language that still sees use due to its deep integration with the iOS ecosystem.

Swift's modern syntax and powerful features make it the preferred choice for new iOS development projects. Its type safety and error handling mechanisms help prevent common programming errors, resulting in more reliable and secure applications. While Objective-C remains relevant for maintaining legacy codebases, it is generally considered less efficient and more verbose than Swift.

3.1.11.3. JavaScript (React Native)

JavaScript, through frameworks like React Native, enables developers to write mobile apps using JavaScript and React. This approach allows for the development of cross-platform apps with a single codebase, targeting both Android and iOS. React Native utilizes native components, providing near-native performance and look and feel.

Using web development skills for mobile app development is a significant advantage of React Native, reducing development time and effort. However, it may introduce performance trade-offs compared to native development. It may be required to optimize certain parts of the app using native code to achieve the required performance.

Table 3.28 - *Mobile Language Comparison*

	Java	**Kotlin**	**Swift**	**Objective-C**	**Javascript**
Modern Syntax	No	Yes	Yes	No	Yes
Null Safety	No	Yes	Yes	No	Yes

Data Classes	No	Yes	Yes	No	No
Coroutines/Asynchronus	No	Yes	Yes	No	Yes
Code Conciseness	Moderate	High	High	Low	Moderate
Readability	Moderate	High	High	Low	Moderate
Developer Productivity	Moderate	High	High	Low	Moderate
UI Design Integration	High	High	High	Moderate	High
Library Integration	High	High	High	Moderate	Moderate
Garbage Collection	Yes	Yes	Yes	Yes	Yes
Runtime Environment	Yes	Yes	Yes	Yes	Yes
Type Safety	No	Yes	Yes	No	No
Error Handling	Moderate	High	High	Low	Moderate
Performance	High	High	High	High	Moderate
Cross-Platform	No	No	No	No	Yes
Legacy Code Maintenance	High	High	Moderate	High	Low
Community Support	High	High	High	High	High

3.1.12. IDE Environments

When developing a complex project like HelmetIQ, selecting the appropriate Integrated Development Environment (IDE) is crucial as it significantly impacts the development workflow, ease of debugging, code management, and overall productivity. We needed an IDE for MCU development as well as mobile application development.

3.1.12.1. Arduino IDE

The Arduino IDE is widely known for its simplicity and ease of use which is great for rapid prototyping. It supports a range of Arduino boards and compatible MCUs. The Arduino IDE's interface requires minimal setup and configuration, allowing us to quickly begin writing and uploading code. Its extensive library support and large community provide abundant resources, tutorials, and forums for troubleshooting and learning. However, the Arduino IDE lacks some advanced features found in more sophisticated environments, such as integrated debugging tools and code analysis capabilities, and can become cumbersome for large projects requiring complex code management.

3.1.12.2. PlatformIO

PlatformIO is a popular IDE known for its features and versatility, supporting multiple frameworks and MCUs. PlatformIO provides integrated debugging tools, advanced code editing features such as IntelliSense and real-time code analysis. The IDE is highly customizable and can be modified to fit specific development needs. However, PlatformIO has a steeper learning curve compared to the Arduino IDE and requires more system resources, which can be a drawback on lower-end hardware.

3.1.12.3. VS Code

Visual Studio Code is an IDE that allows for cross-platform mobile app development as well as some MCU development. VS Code is highly customizable with a vast selection of extensions, enabling programmers to tailor the environment. However, setting up VS Code for mobile development involves multiple configurations and can be time-consuming. Some advanced mobile features may be impossible to implement in VS Code because in order to implement them you must use the native IDEs for both Android and iOS.

3.1.12.4. Xcode

Xcode is Apple's official IDE for iOS development, and it supports Swift and Objective-C. Xcode integrates with Apple's ecosystem, providing tools for designing, coding, testing, and debugging iOS applications. The visual interface builder would allow us to design user interfaces using a drag-and-drop interface. Xcode has real-time syntax checking, code completion, and a powerful debugger. These are advanced tools that would be useful in writing the code for our project. However, Xcode is only available on macOS, limiting its accessibility to members in the group.

3.1.12.5. Android Studio

Android Studio is Google's official IDE for Android development, and it supports Java and Kotlin. Android Studio is in direct contrast to Xcode in that it is compatible with the entire Android ecosystem. This is a much larger ecosystem than the Apple counterpart because of the availability, accessibility, and open nature of Android in general. This means that many more applications have been developed using Android Studio and by consequence there are many more libraries to aid in development. In our project the use of Bluetooth in the hardware is key and the access to hardware specific modules on Android phones is much simpler than iOS. The Android Studio IDE makes it so that all hardware specific modules are accessible and modifiable and then can be simulated on a virtual machine within the IDE. This is great for testing and prototyping. More importantly, it is accessible on macOS and Windows so all members of the group can have access.

Table 3.29 *- MCU IDE Comparison*

	Arduino IDE	PlatformIO	VS Code
Difficulty	Low	Moderate	Moderate
Setup and Configuration	Minimal	Moderate	Moderate
Library Support	Extensive	Extensive	Extensive

Community Support	Large	Large	Large
Debugging Tools	Limited	Integrated	Integrated
Code Analysis	No	Yes	Yes
Rapid Prototyping	Yes	Yes	Yes
Advanced Features	Limited	Yes	Yes
Resource Requirements	Low	High	High
Customization	Low	Yes	Yes

Table 3.30 *- Mobile IDE Comparison*

	Xcode	**Android Studio**	**VS Code**
Difficulty	Low	Low	Moderate
Setup and Configuration	Moderate	Minimal	High
Library Support	Extensive	Extensive	Moderate
Community Support	Large	Large	Large
Debugging Tools	Integrated	Integrated	Integrated
Code Analysis	Yes	Yes	Yes
UI Design Tools	Yes	Yes	No
Compatibility	macOS	Cross-platform	Cross-platform
Ecosystem Support	iOS	Android	Cross-platform
Hardware Module Access	Moderate	High	Moderate
Emulation	Yes	Yes	Yes
Resource Requirements	High	Moderate	Moderate

3.1.13. Auto-SMS Technology

In the HelmetIQ project, the auto SMS feature is crucial for ensuring immediate communication in the event of a collision. Selecting the right auto SMS technology is vital to ensure reliability, speed, and ease of integration. Below are comparisons of different auto SMS companies and technologies, in addition to using standard SMS off the user's phone.

3.1.13.1. Twilio

Twilio is a widely used cloud communications platform known for its robust API and reliable service. Twilio's SMS API allows for easy integration into various systems and offers extensive documentation and support. The platform supports global messaging, making it suitable for international use. Twilio's scalability ensures it can handle a high volume of messages, which is essential in emergency scenarios. However, Twilio's pricing can be higher compared to some other services, especially for high-volume messaging.

3.1.13.2. Nexmo (Vonage API)

Nexmo is another popular choice for SMS messaging services. Nexmo's API is user-friendly and supports a wide range of programming languages, making it easy to integrate with different systems. It offers competitive pricing and reliable service, with a strong global reach. Nexmo also provides features such as SMS delivery receipts and message tracking, which can be useful for ensuring messages are sent and received. However, while Nexmo offers good documentation, it may not be as extensive as Twilio's, potentially making the integration process slightly more challenging.

3.1.13.3. Plivo

Plivo is known for its cost-effective SMS and voice solutions. Plivo's API is designed for easy integration and supports high-volume messaging at a lower cost compared to Twilio and Nexmo. The platform also offers good reliability and global coverage. Plivo's customer support is responsive, which is beneficial during the integration phase and for troubleshooting. However, Plivo's documentation, while sufficient, is not as comprehensive as Twilio's, which might require additional effort for integration.

3.1.13.4. MessageBird

MessageBird is a global cloud communications platform that offers reliable and scalable SMS services. MessageBird's API is easy to integrate and provides a range of features such as delivery reports, message batching, and scheduling. The platform is designed for high-volume messaging and offers competitive pricing. MessageBird also supports omnichannel messaging, which can be advantageous if the HelmetIQ project expands to include other communication methods. However, like Plivo, MessageBird's documentation is not as extensive as Twilio's, which might pose a challenge for us.

3.1.13.5. ClickSend

ClickSend is a versatile communication platform that offers SMS, email, voice, and other services. ClickSend's SMS API is straightforward to integrate and supports global messaging. The platform provides delivery reports and message tracking, ensuring messages are reliably sent and received. ClickSend is known for its competitive pricing and ease of use. However, it may lack some advanced features found in other platforms like Twilio, making it more suitable for straightforward SMS applications rather than complex integrations.

3.1.13.6. Standard SMS via Phone

Compared to the other technologies listed above, another option for HelmetIQ is to use the standard text (SMS) functionality on the user's phone. Instead of outsourcing the emergency SMS functionality to another company, all the intended functionalities can be accomplished with the user granting SMS permissions via the mobile app. Compared to the other technologies, this option will allow the SMS to be transmitted using the user's phone number – which is much more familiar to their defined emergency contacts than a new number from API usage. However, this option may not be comfortable to all users as they may not want to give the HelmetIQ mobile application SMS permissions.

3.1.13.7. Auto-SMS Selection

***Table 3.31** – Auto-SMS Comparison Table*

	Twilio	Nexmo	Plivo	MessageBird	ClickSend	User's Phone
Ease of Integration	Very Easy	Easy	Easy	Easy	Very Easy	Very Easy
API Documentation	Extensive	Good	Sufficient	Good	Sufficient	N/A
Global Messaging Support	Yes	Yes	Yes	Yes	Yes	Yes
Scalability	High	High	High	High	Moderate	Extremely High
Pricing	High	Competitive	Low	Competitive	Competitive	N/A
Features	Robust, Delivery Tracking	Delivery receipts, message tracking	Basic but cost effective	Delivery reports, message batching	Basic	N/A
Customer Support	Excellent	Good	Responsive	Good	Good	N/A
Reliability	Very High	High	High	High	Moderate	N/A

When comparing the different technologies that could be used to transmit the emergency SMS, a major factor that needs to be considered is the legal guidelines. In the United States, you must have a registered business to send auto-SMS messages to unverified phone numbers. This requirement includes business verification documentation, a registered business entity, a clear use case that complies with messaging policies, and registration with The Campaign Registry for A2P (Application-to-Person) messaging. These restrictions exist primarily to prevent spam and ensure compliance with telecommunications regulations. Since HelmetIQ is for Senior Design, we do not have a registered business to activate auto-SMS functionality for unverified phone numbers. Therefore, to successfully implement the features of HelmetIQ's collision detection and response system, we choose to use the user's phone given that they allow SMS permissions through the mobile app. This was the best choice as we could demonstrate the full functionality of HelmetIQ as intended – without the use of a virtual phone or workaround that would need to be used if we choose another SMS technology or API.

3.2. Helmet Base Comparison

3.2.1 Mountain Bike Helmet Base

Mountain bike helmets are designed for off-road cycling and provide extensive coverage, especially at the back and sides of the head. They often include features such as visors to protect against sun and debris. These helmets are robust and durable, making them well-suited for incorporating advanced safety features like dynamic lighting and collision detection. The ample space and rugged design can accommodate various sensors and electronics. However, for urban cyclists who prioritize comfort and daily usability, a mountain bike helmet might feel overly bulky and heavy compared to commuter helmets.

Figure 3.9 *– Full Coverage Mountain Bike Helmet [53]*

3.2.2. BMX Bike Helmet Base

BMX helmets offer substantial protection with their full coverage design, making them ideal for high-impact activities. They are constructed from durable materials to withstand frequent falls and collisions, common in BMX riding. While the BMX helmet ensures maximum safety, its weight and design might not align with the needs of more casual cyclists who prefer a more streamlined, lightweight helmet for daily commuting. The BMX helmet's ruggedness, though advantageous for certain applications, might be excessive for regular city riding. The hard outer shell also makes the BMX helmet much more difficult to modify and implement our sensors into. The durability of the BMX helmet is advantageous for safety but for our modification needs it does not work as well.

Figure 3.10 *– BMX Bike Helmet Base [54]*

3.2.3. Full Face Helmet Base

Full-face helmets provide comprehensive protection, covering the entire head and face, including a chin guard. This extensive coverage is beneficial for high-risk activities, offering the best safety. The large surface area is ideal for integrating multiple sensors and lighting systems. However, the full-face helmet's weight and reduced ventilation can lead to discomfort during everyday use. For urban cyclists, the full-face helmet's design may be excessive, as it prioritizes protection over comfort and convenience. Similar to the BMX helmet, the full-face helmet is a hard outer shell which is harder to implement external sensors. However, there is a lot more space inside the helmet in which to run ribbon cable and hide PCBs.

***Figure 3.11** – Full Face Helmet Base [55]*

3.2.4. Commuter Bike Helmet Base

Commuter bike helmets are designed specifically for everyday urban cycling, balancing comfort, protection, and style. They provide ventilation and sufficient coverage, making them suitable for city riding. The practical and lightweight design of commuter helmets ensures that they are comfortable for daily use, making them ideal for the HelmetIQ project. They offer the perfect platform for integrating adaptive lighting, collision detection, and ride data collection without compromising on style or comfort. This is because they do not have a thick hard-shell outer layer that is hard to penetrate. Additionally, the many ventilation cutouts allow for many places for us to run ribbon cables to connect our sensors to the main PCB. For most cyclists, commuter helmets offer the best blend of functionality and user-friendliness.

***Figure 3.12** – Commuter Bike Helmet Base [56]*

3.2.5. Helmet Base Comparison Table

Table 3.32 - *Helmet Base Comparison*

	Mountain Bike Helmet	BMX Helmet	Full-Face Helmet	Commuter Helmet
Protection	High	High	Very High	Adequate
Weight	Moderate	High	Very High	Low
Ventilation	Good	Limited	Limited	Good
Daily use Comfort	Moderate	Low	Low	High
Ease of Integration	Suitable	Difficult	Difficult	Suitable
Practicality	Moderate	Low	Low	High
Aesthetics	Sporty	Rugged	Bulky	Sleek
Adaptability for Technology	High	Medium	Medium	Very High

3.3. Part Comparison and Selection

3.3.1. MCU Selection

The ESP32 was selected for the HelmetIQ project based on several key factors that align with the project's requirements:

1. The built-in Wi-Fi and Bluetooth capabilities are crucial for the HelmetIQ's advanced features - such as transmitting a collision detection to the user's cell phone.
2. The dual-core processor and high clock speed ensure that the system can handle all calculations required for this project.
3. The ESP32's rich set of peripherals matches the needs for implementing features like haptic feedback turn signals, brake lights, and collision detection systems.
4. The low power consumption modes help in maintaining a longer battery life.
5. The ESP32 provides a powerful yet affordable solution, which is critical in keeping the overall project within budget constraints while delivering high functionality.

While TI microcontrollers offer industrial-grade reliability and Arduino boards provide ease of use and strong community support, the ESP32 stands out as the most suitable choice for the HelmetIQ project due to its combination of high performance, integrated wireless communication, rich peripheral support, power efficiency, and cost-effectiveness.

3.3.2. Light Sensors

Table 3.33 *– Light Sensor Comparison*

	VEML7700	***AS7262***	***TSL2591***	***GL55xx***

Type	Photodiode	Photodiode	Photodiode	Photoresistor
Supply Voltage	3.3V or 5V	2.7V to 3.6V	3.3V to 5V	N/A
Size	6.5mm x 2.35mm x 3.0mm	90mm x 50mm x 5mm	19mm x 16mm x 1mm	4.3mm x 32mm x 2.1mm
Cost	$7.35	$27.95	$8.48	$0.29
Provide Lux?	Yes	Can be Calculated	Can be Calculated	No

Using the table above there are several things to take into consideration when deciding which Light Sensor would be the best fit for the HelmetIQ. At first glance the best option seems to be the photoresistor; this would be a fast, responsive method to determine the low ambient light. It is also extremely cheap and very small. The main issue here is it fails to be dynamic and would work as an on-or-off switch. Due to the constant fluctuations of light in an everyday bike ride, this would not be the most reliable.

The first option is to use the TSL2591. This is a very good option as it provides lux values, has the ambient light sensor, and is on the cheaper side. The price for this sensor is mid-range and has a few features that would be unnecessary for the headlight. Next, we can look at the AS7262. This component provides lots of information ranging from light intensity, and specific color, to even lux values. Negatively, although it has what we need, it also goes beyond the features needed to determine whether the headlight should be on. It should rely on the lux value and not the color of the light; this is unnecessary and causes a major impact on the price, this being the most expensive out of the three. Therefore the chosen sensor is the VEML 7700. It is a cheap, quick, and accurate sensor, perfect for the headlight system.

3.3.3. Mobile Development Language Selection

In our project, we chose Android Studio and Kotlin for our mobile app development for several reasons:

1. Kotlin's modern syntax and features enhance developer productivity and code safety, making it an excellent choice for developing robust applications.
2. One of our team members has significant experience with Kotlin and Android Studio, allowing us to leverage existing expertise to streamline development and ensure a high-quality application. This experience speeds up the development process and reduces the learning curve associated with adopting a new technology stack.

By leveraging Kotlin, we ensure the mobile app is user-friendly, feature-rich, and seamlessly integrates with the MCU, achieving a cohesive and effective overall system. The active Kotlin community provides ample resources and frameworks to assist in development, troubleshooting, and maintaining best practices.

3.3.4. Gyroscope/Accelerometer

3.

Table 3.34- *Gyroscope/Accelerometer Comparison*

	ICM-42670	ADA BNO055	LSM6DS33
Supply Voltage	2.375V or 3.46V	2.4V to 3.6V	1.71V to 3.6V
Size	21.2mm x 16.4mm x 3.3mm	20mm x 27mm x 4mm	3mm x 3mm x .86mm
Cost	$3.33	$24.95	$29.95
Free Fall Interrupts	No	Yes	Yes
Accuracy	Low	Low	Reasonable
Calibration required	Yes	Yes	No

When considering the different accelerometers/gyroscopes, different factors play important roles. One of these is the hardware interrupts. It is important for the collision detection system that the device has a free-fall interrupt. Another factor to consider is accuracy; in this case, the level of accuracy is less important. This is because the main goal is to detect significant and sudden changes to get the correct collision detection. Small changes in accuracy do not affect this. A major deciding factor is the cost. The best choice for the HelmetIQ would be the LSM6DS33 but due to financial constraints, the selected device is the ICM-42670.

3.3.5. LED Comparison

Table 3.35 *– LED Comparison*

Feature/Criteria	Diffused RGB	Cree XP-G3	COB LED Strip	Nichia NVSW219F-V1	Wurth Elektronik LED 1206 SMD	Luminus SST-20-WxH
Supply Voltage	1.8V to 3.6V	2.83V to 3.3V	3.0V to 3.7V	2.7V to 3.3V	2.00 V to 2.7V	2.5V to 3.1V
Size	8.7mm x 1.0mm x 1.0mm	20mm x 20mm	3mm x 3mm x .86mm	2.3mm x 3.2mm x 3.2mm	3.20mm x 1.6mm	3.5mm x 3.5mm x 1.9mm

Cost	$0.95	$2.65	$3.31	$1.54	$0.30	$4.50
Brightness	20 Lm	262 Lm	300 Lm	600 Lm	80 Lm	550 Lm

When comparing the different LEDs there are a few key factors to consider, in the case of the headlight the most important factor is the lumens the lumens should reach the specified amount to properly light the way for cyclists. This can be difficult as different levels of current passing through the LED produces different light levels. To reach the higher lumen output there is a higher cost and may require multiple of the same LED. The best option would be to use the COB LED strip although it is more expensive it is at a reasonable range. The Cree XP-G3 is a strong consideration for the brake light and turn signal if the current draw is set to a lower level.

3.3.6. Battery

For the smart helmet project, the SciGold rechargeable 18650 batteries were chosen due to their high energy density, reliable performance, and suitability for the project's power requirements. While there are other viable options like the Panasonic NCR18650B or Samsung INR18650-30Q, the SciGold 18650 batteries offer a cost-effective and robust solution, ensuring consistent performance for the helmet's operations.

To achieve the necessary voltage for the project, two 3.7V SciGold 18650 Li-Ion batteries are connected in series using a 2-slot battery holder. This configuration provides a total voltage of 7.4V, which aligns well with the system's power needs. The 7.4V output from the battery pack is efficiently regulated down to 3.3V using the TPS62162 buck converter. This setup ensures minimal power loss, reduced heat generation, and a stable power supply for the ESP32 microcontroller and other components, enhancing the overall reliability and energy efficiency of the smart helmet.

Table 3.36 *– Li-Ion Battery Comparison*

Feature/Criteria	SciGold 18650	Samsung INR18650-30Q	LG MJ1
Voltage	3.7V	3.6V	3.6V
Capacity	12600mAh	3000mAh	3500mAh
Max Discharge Current	6.8A	15A	10A
Cycle Life	>1000 cycles	>500 cycles	>500 cycles
Energy Density	High	High	High
Dimensions	18mm x 67mm	18.6mm x 65.2mm	18.6mm x 65.2mm
Weight	-	48g	49g
Self-Discharge Rate	<2% per month	<2% per month	<2% per month

3.3.7. Voltage Regulation

The primary power source for the smart helmet is a 7.4V battery (Two 3.7V batteries in series), and most microcontrollers and sensors used in this project, including the ESP32 microcontroller, operate at 3.3V. To ensure stable and efficient power conversion in the smart helmet project, we have chosen the TPS62162 buck converter over other options. This decision was based on its superior efficiency, compact design, and ability to handle the project's power demands with minimal heat generation. The TPS62162 provides a steady 3.3V output necessary for the proper functioning of the ESP32 and other components, achieving efficiencies of up to 95%. Its 1A output current is more than sufficient for our system's requirements.

The TPS62162 incorporates components to stabilize its output effectively:

- Input Capacitor: Stabilizes the input voltage by filtering out high-frequency noise and transient voltage spikes, ensuring a smooth power supply to the converter.
- Output Capacitor: Filters out noise and reduces ripple in the output voltage, maintaining stability for connected devices.
- Inductor: Essential for the buck converter's operation, it stores and releases energy to regulate voltage effectively while maintaining high efficiency.

Table 3.37 *– Voltage Regulators Comparison*

Feature/Criteria	TPS62162	Texas Instruments LM2596 Buck Converter	LM1117DT-3.3V Voltage Regulator
Input Voltage Range	2V to 15V	3V to 40V	4.5V to 15V
Output Voltage	3.3V	Adjustable (1.23V to 37V)	3.3V
Output Current	Up to 1A	Up to 3A	Up to 800mA
Efficiency	Up to 95%	Up to 92%	Not specified
Dimensions	13mm x 10mm x 3mm	45mm x 20mm x 14mm	8mm x 7mm x 4mm
Safety Features	Thermal shutdown, reverse voltage protection, overcurrent protection	Thermal shutdown, overcurrent protection	Overcurrent protection, Thermal Shutdown
Cost	$1.98	$7.99	$0.95

3.3.8. MCU Language Selection

In our smart helmet project, we ultimately chose C++ over MicroPython and Lua due to its superior performance, fine-grained control over hardware resources, and extensive library support. C++ allows us to write efficient code that maximizes the capabilities of the ESP32, ensuring that the firmware is robust, responsive, and capable of handling real-time data processing and hardware interaction. Additionally, C++'s large community and support resources enable us to leverage existing solutions and libraries, accelerating the development and troubleshooting processes.

3.3.9. IDE Selection

After evaluating the various options, we chose the Arduino IDE for MCU development and Android Studio for mobile app development. The Arduino IDE's simplicity, user-friendly interface, strong community support, and compatibility with many possible MCUs make it an excellent choice for rapid prototyping and initial development. For mobile app development, Android Studio's robust toolset and features like the visual layout editor and performance profiling tools align with our project's requirements for a high-quality mobile app. Additionally, the significant experience one of our team members has with Android Studio and this allows us to leverage existing expertise and streamline the development process. The combination of the Arduino IDE and Android Studio provides a balanced and effective development environment for the HelmetIQ project, enabling us to achieve our goals efficiently and effectively.

3.3.10. Helmet Base Selection

The commuter bike helmet base has been selected as the optimal base for the HelmetIQ project due to its blend of comfort, practicality, and adaptability. Unlike mountain bike helmets which can be bulky and cumbersome for daily use, the commuter helmet is specifically designed for urban cycling. Its lightweight and well-ventilated design ensures comfort during extended wear, making it suitable for everyday use. Additionally, the helmet's ample ventilation cutouts provide convenient access for integrating sensors and cables necessary for the HelmetIQ project, without compromising the helmet's style or functionality. This balance of user comfort and practical modification capabilities makes the commuter helmet the best choice for implementing advanced safety features, sensors, and in result achieving the goals of HelmetIQ.

3.3.11. 3D Print Filament

When creating a clear cover for a custom PCB with light sensors, selecting the right type of clear filament is crucial. The filament needs to provide high transparency to ensure that the light sensors function correctly without interference. Below are the comparisons of different clear filaments.

3.3.11.1. PLA (Polylactic Acid)

PLA is one of the most commonly used filaments in 3D printing due to its ease of use and availability. Clear PLA offers good transparency and is easy to print with minimal warping. It provides high transparency and a glossy finish, is biodegradable and eco-friendly, and has a low printing temperature (180-220°C). Additionally, PLA experiences minimal warping, making it suitable for large prints. However, it is brittle compared to other filaments, which may not be as suitable for covers that need to endure physical stress, and it has lower heat resistance, which might be an issue if the PCB generates significant heat.

3.3.11.2. PTEG (Polyethylene Terephthalate Glycol)

PETG is a durable and versatile filament that combines the ease of printing of PLA with the strength and durability of ABS. Clear PETG offers excellent transparency and is known for its chemical resistance. It is strong and durable, with good impact resistance, and is chemically resistant, which can be beneficial in certain environments. PETG is also easy to print with, having low warping (print temperature: 220-250°C). However, it is prone to stringing and requires fine-tuning of printer settings and is slightly more challenging to print than PLA.

3.3.11.3. PC (Polycarbonate)

Polycarbonate is known for its great strength and high impact resistance. Clear PC filament provides excellent transparency and is often used in applications where durability is most important. Its transparency is nearly as clear as glass, and is extremely strong and durable. PC also has high heat resistance, making it suitable for high-temperature environments (print temperature: 250-300°C). However, it is difficult to print with, requiring high temperatures and a heated bed, and it is prone to warping, especially on large prints. An enclosure is usually necessary to maintain print temperature and avoid warping.

3.3.11.4. Nylon

Nylon filament is known for its strength, flexibility, and durability. Clear nylon can offer good transparency, though it might not be as clear as PLA or PETG. Nylon is strong and flexible, with good impact resistance and chemical resistance. It is suitable for functional parts that require durability (print temperature: 240-260°C). However, it has lower transparency compared to PLA, PETG, and PC, is prone to absorbing moisture, which can affect print quality, and requires fine-tuning and specific printer settings.

3.3.11.5. PMMA (Acrylic)

PMMA or acrylic, is known for its glass-like transparency and high gloss finish. It is a good choice for parts that require excellent clarity. PMMA offers the highest transparency among common filaments and has a glossy finish. It is also good for rigidity and UV resistance (print temperature: 240-260°C). However, it is brittle and not as impact-resistant as PETG or PC. It can be challenging to print with and is prone to warping and cracking. It requires specific print settings and possibly an enclosure.

3.3.11.6. Filament Selection

Table 3.38 *– Filament Comparison*

	PLA	**PTEG**	**PC**	**Nylon**	**PMMA**
Transparency	High	High	Very High	Moderate	Very High

Strength	Moderate	High	Very High	High	Low
Durability	Low	High	Very High	Very High	Low
Ease of Printing	Very Easy	Moderate	Difficult	Difficult	Difficult
Heat Resistance	Low	Moderate	Very High	High	Moderate
Flexibility	Low	Low	Low	High	Low
Chemical Resistance	Low	High	Moderate	High	Moderate
Moisture Sensitivity	Low	Moderate	Low	High	Low

For HelmetIQ PLA emerges as the best choice. PLA's ease of printing, minimal warping, and low cost make it ideal for creating the clear cover needed for the custom PCB with light sensors. Additionally, PLA's moderate strength and sufficient durability align well with the requirements for protecting the PCB while ensuring it remains lightweight and user-friendly. Despite its lower heat and UV resistance compared to ABS and PETG, PLA's biodegradability and eco-friendliness provide an added advantage, making it a suitable and responsible material choice for this project.

4. Standards and Design Constraints

4.1. Technical Standards

4.1.1. I2C Serial Communication Protocol Overview

The Inter-Integrated Circuit (I2C) is a widely adopted, multi-master, multi-slave, packet-switched, single-ended, serial communication bus. It is designed for efficient and robust communication between integrated circuits over short distances within a single device or on a single PCB. The I2C protocol requires only two bidirectional open-drain lines: the Serial Data Line (SDA) and the Serial Clock Line (SCL). These lines are pulled up with resistors, allowing multiple devices to be connected to the bus without interference, as the open-drain configuration enables devices to either pull the line to ground or leave it floating.

One of the core features of I2C is its simplicity and efficiency in data transfer. The protocol allows for multiple masters and multiple slaves, allowing complex device interactions. Communication is initiated by a master device, which generates a start condition by pulling the SDA line low while the SCL line remains high. Following the start condition, the master sends an address byte that includes a unique 7-bit or 10-bit identifier for the target slave device, along with a read/write bit to indicate the intended operation. The slave device that recognizes its address responds with an acknowledgment (ACK) by pulling the SDA line low during the acknowledgment clock pulse. This mechanism ensures that the master knows the slave is ready for communication.

Data transfer on the I2C bus is serial and occurs in 8-bit packets, with each byte followed by an acknowledgment bit from the receiving device. This acknowledgment process is critical for reliable communication, as it allows the master to verify that the slave has successfully received the data. If the slave does not pull the SDA line low during the acknowledgment bit, it signals a negative acknowledgment (NACK), indicating that the data was not received correctly or that the slave is not ready for further communication. The master can then decide to resend the data or terminate the communication with a stop condition, which is generated by pulling the SDA line high while the SCL line is high.

I2C supports multiple data transfer speeds to accommodate various application needs. The standard mode operates at up to 100 kbit/s, suitable for slower devices and simple communication tasks. The fast mode increases the data rate to 400 kbit/s, enabling more efficient data transfer for moderately demanding applications. For even higher speed requirements, the fast mode plus supports up to 1 Mbit/s, while the high-speed mode can reach up to 3.4 Mbit/s, though these higher speeds often require special considerations, such as reduced bus capacitance and more stringent signal integrity requirements.

4.1.1.1. Implementation of I2C

In our HelmetIQ project, we utilize the I2C protocol for efficient communication between the ESP32 microcontroller and various sensors integrated into the system. The ESP32, a powerful microcontroller with built-in I2C interfaces, is well-suited for managing multiple peripheral devices on the same bus. Implementing I2C simplifies the wiring, reduces the number of GPIO pins required, and allows for the efficient integration of multiple sensors, each with its unique address.

The first step in implementing I2C in our project is configuring the I2C bus on the ESP32. This involves designating the appropriate GPIO pins for the SDA and SCL lines and initializing the I2C interface in the microcontroller's firmware. The ESP32's SDK (Software Development Kit) provides convenient functions for setting up the I2C bus, specifying parameters such as the bus frequency and the pins used. For our application, we choose a bus frequency that balances speed and reliability, ensuring stable communication without signal degradation.

Each sensor connected to the I2C bus must be assigned a unique address. These addresses are typically predefined by the sensor manufacturers and are critical for ensuring that the master device, the ESP32, can communicate with the correct slave device. Addressing is crucial in a multi-device environment, as it allows the master to selectively communicate with individual sensors without interference. The 7-bit addressing scheme allows for up to 128 unique addresses, while the 10-bit scheme extends this capability to 1024 addresses, although 7-bit addressing is more commonly used.

One of the primary sensors in our HelmetIQ system is the capacitive touch sensor, which is used for haptic feedback turn signals. The I2C interface on the capacitive touch sensor enables seamless data transmission to the ESP32. When a touch is detected, the sensor sends the corresponding data over the I2C bus to the microcontroller. The ESP32 processes this data and activates the haptic feedback actuators, providing tactile confirmation to the user. The reliable and fast communication facilitated by I2C ensures that the feedback is instantaneous, enhancing the user experience.

Another crucial component is the accelerometer, responsible for detecting deceleration and triggering the brake lights. The accelerometer communicates with the ESP32 via the I2C bus, transmitting data that the microcontroller analyzes to determine if the cyclist is slowing down. If a significant deceleration is detected, the ESP32 activates the brake lights, enhancing rider safety. The continuous and real-time data transfer enabled by I2C is vital for ensuring timely and accurate responses from the brake light system.

Additionally, our project incorporates a gyroscope for collision detection and a light sensor for dynamic lighting adjustments. The gyroscope provides data on the helmet's orientation and detects impacts, while the light sensor adjusts the helmet's lighting based on ambient light conditions. Both sensors communicate with the ESP32 over the I2C bus, allowing centralized data processing. By integrating all sensor data through the I2C bus, we streamline the system architecture and ensure coordinated responses to various environmental and operational conditions.

The use of I2C in our HelmetIQ project offers several advantages. It simplifies the hardware design by reducing the number of required connections, as multiple devices share the same SDA and SCL lines. This shared bus architecture is particularly beneficial in space-constrained environments like wearable technology, where minimizing wiring complexity is crucial. Furthermore, the scalability of I2C allows for easy addition of new sensors or devices to the bus, facilitating future expansions and upgrades to the HelmetIQ system.

Error detection and correction are inherent features of the I2C protocol, enhancing communication reliability. The acknowledgment bit mechanism ensures that data is received correctly. These features are critical in our project, where reliable sensor data is essential for ensuring the safety and functionality of the HelmetIQ system.

The I2C protocol is an important part of our HelmetIQ project, providing a robust and efficient means of communication between the ESP32 microcontroller and various sensors. Its simplicity, flexibility, and scalability make it the ideal choice for integrating multiple peripheral devices into a cohesive and responsive safety system for cyclists. By using I2C, we ensure accurate sensor data processing, timely feedback, and enhanced safety features for users, thereby delivering a reliable and effective smart helmet solution.

4.1.2. Bluetooth Standard Overview

Bluetooth technology, governed by the Bluetooth Special Interest Group (SIG), follow rigorous standards that ensure interoperability, security, and efficiency in wireless communication. The Bluetooth SIG standards cover various facets of Bluetooth technology, including the physical layer protocols, core system architecture, and application profiles. These standards are needed for developing and implementing Bluetooth-enabled devices that can communicate seamlessly and securely.

The foundation of Bluetooth standards lies in the Bluetooth Core Specification. This comprehensive document defines the core technologies of Bluetooth, encompassing the physical radio layer, baseband, link manager, and host controller interface. The physical radio layer specifies the frequencies, modulation techniques, and transmission power levels used for communication. Bluetooth operates in the 2.4 GHz ISM band, utilizing frequency hopping spread spectrum (FHSS) to minimize interference and ensure reliable communication. The baseband layer handles the creation of Bluetooth packets and manages the link's access control and addressing. Error correction mechanisms within the baseband layer enhance data integrity during transmission.

The link manager protocol (LMP) is responsible for establishing, controlling, and securing the Bluetooth link. It manages tasks such as device discovery, connection setup, and link encryption. The host controller interface (HCI) provides a standardized interface that allows the host system, such as a microcontroller or computer, to communicate with the Bluetooth module. This standardization ensures compatibility between different Bluetooth hardware and software implementations.

Bluetooth technology includes various profiles, each defining specific applications and use cases for Bluetooth communication. These profiles ensure that devices designed for similar functions can interoperate seamlessly. Common Bluetooth profiles include the Generic Access Profile (GAP), which manages device discovery and connection procedures, and the Serial Port Profile (SPP), which emulates a serial cable connection between devices. The Human Interface Device (HID) profile is used for devices such as keyboards and mice, while the Advanced Audio Distribution Profile (A2DP) handles high-quality audio streaming.

Security is an important part of Bluetooth technology, and the Bluetooth SIG standards incorporate robust mechanisms to protect data integrity and privacy. Bluetooth employs a combination of encryption, authentication, and frequency hopping to secure communication. Encryption ensures that transmitted data cannot be easily intercepted and read by unauthorized parties. Authentication verifies the identity of devices attempting to connect, preventing unauthorized access. Frequency hopping further enhances security by rapidly changing the communication frequency, making it difficult for potential hackers to track and intercept data.

4.1.2.1. Implementation of Bluetooth

In the HelmetIQ project, Bluetooth technology plays a crucial role in enabling wireless communication between the ESP32 microcontroller and a paired mobile device. The Bluetooth module integrated into the ESP32 adheres to the Bluetooth SIG standards, ensuring reliable and secure communication. Notably, Bluetooth is used exclusively to transmit data to the mobile app, with no Bluetooth communication occurring between the sensors.

The implementation of Bluetooth begins by using Bluetooth module on the ESP32 microcontroller. The ESP32 has built-in Bluetooth capabilities and serves as the central hub for processing data from various sensors embedded in the helmet. The Bluetooth module on the ESP32 supports both Bluetooth Classic and Bluetooth Low Energy (BLE) modes, providing flexibility for different types of communication.

Our project utilizes Bluetooth Low Energy (BLE) due to its lower power consumption, which is crucial for maintaining long battery life in the helmet. BLE operates by establishing a connection between the helmet and a paired mobile device, allowing efficient data transmission and reception. The ESP32 uses the BLE Generic Attribute Profile (GATT) to define how data is organized and exchanged between the devices.

When the helmet detects an event, such as a collision or a change in light conditions, the ESP32 processes the sensor data and update the corresponding BLE characteristics. The paired mobile device, running a custom app, receives this data and when an event is triggered, activate the collision functionality of the mobile app. This allows real-time monitoring and response to events, enhancing the safety and functionality of the helmet.

Security measures are implemented to protect the data transmitted over Bluetooth. The ESP32 uses Bluetooth Secure Simple Pairing (SSP) to establish a secure connection with the mobile device. SSP employs a combination of numerical comparison, passkey entry, and out-of-band pairing methods to ensure that only authorized devices can connect to the helmet. Additionally, data encryption is enabled to protect the integrity and confidentiality of the transmitted data.

The Bluetooth SIG standards provide a comprehensive framework for implementing reliable, secure, and efficient wireless communication in our project. By adhering to these standards, we ensure that the helmet's Bluetooth functionality can deliver the enhanced safety features required for our smart helmet system. The integration of Bluetooth Low Energy on the ESP32 enables seamless communication between the ESP and the paired mobile device, providing real-time monitoring and response to critical events.

4.2. Design Constraints

4.2.1. Helmet Safety Constraints

With the goal of this project being to provide a safer helmet to bike riders, it is essential that the general properties and pre-established safety standards of a bike helmet are preserved. These safety standards are defined in the Code of Federal Regulations (CFR) – 16 CFR Part 1203 – Safety Standards for Bicycle Helmets.

The safety standards listed in the CFR are extensive. For this reason, the path of least resistance for the construction of the smart bike helmet is to work around these already established safety standards. In other words, we would like to add all features of this smart bike helmet without making any physical alterations that would change the helmet's structural integrity. There are several benefits to this approach, being:

1. Helmets do not need to be re-certified after the implementation of HelmetIQ.
2. Reduced production costs – No need for a custom helmet shell.
3. Repairability
4. Easier adoption by consumers who already own compatible helmets.

In addition to the safety standards outlined in the CFR, it is important to consider the impact of the added electronic components on the helmet's overall safety. While the goal is to avoid physical alterations to the helmet's structure, the integration of the PCB, battery, and wiring should not compromise the helmet's ability to absorb and dissipate impact energy.

To ensure that the HelmetIQ meets safety requirements, rigorous testing should be conducted to evaluate the helmet's performance with the added components. This may include impact tests, penetration tests, and retention system tests as specified in the CFR. Additionally, the placement of the electronic components should be carefully considered to minimize any potential interference with the helmet's protective foam liner.

However, in order to circumvent physical alterations to the bike helmet, we introduce further design constraints that must be dealt with and solved.

4.2.1.1. Space Limitation

Reiterating the goal of not modifying the original helmet structure / integrity, we introduce several hardware space constraints. With the intention of HelmetIQ being as sleek and compact as possible, the only potential spot to place this hardware comes at the expense of helmet ventilation.

Figure 4.1 *– Bicycle Helmet Ventilation Cutouts [19]*

As seen within the image, there are several cut outs that are within the bike helmet shell. These cutouts serve the primary purpose of providing ventilation to the rider. If we were to place main hardware components such as the PCB or battery inside of these cutouts, the final product maintains its structural integrity and professional look.

The limited space within the helmet's ventilation cutouts presents challenges in terms of component selection and layout. The PCB and battery must be carefully chosen to fit within these confined spaces while still providing the necessary functionality and power capacity.

To optimize the use of available space, custom-designed PCBs with miniaturized components can be employed. Flexible PCBs or multiple interconnected smaller PCBs can also be considered to conform to the helmet's curved interior surfaces. Additionally, the use of 3D modeling and computer-aided design (CAD) tools can help in creating an efficient layout that maximizes the utilization of the available space.

4.2.1.2. Weight Distribution

With the space limitations being solved using pre-existing helmet cutouts, this solution brings its own constraints in the form of weight distribution. As there is a limited number of cutouts on the bike helmet, we must place the hardware in optimal positions on the helmet to prevent the final product from having an uneven weight distribution. If one side of the helmet is heavier than the other, the helmet would not only be uncomfortable to the rider, but potentially unsafe. This design constraint can be solved with proper planning and small counterweights if needed.

Ensuring proper weight distribution is crucial for both comfort and safety. Uneven weight distribution can cause neck strain, fatigue, and potentially compromise the helmet's stability on the user's head.

To achieve optimal weight distribution, the placement of heavier components, such as the battery and PCB, should be carefully planned. Placing these components closer to the center of gravity of the helmet can help minimize any imbalance. Additionally, using lightweight materials for non-electronic components, such as mounting brackets or cable housings, can further contribute to a balanced weight distribution.

In cases where perfect weight distribution is challenging to achieve, small counterweights can be strategically placed to offset any imbalances. These counterweights should be lightweight and securely attached to the helmet to prevent any shifting or detachment during use.

4.2.1.3. Wiring Limitation

HelmetIQ has components on all sides of the bike helmet. Given the structural constraints previously discussed, and the goal of maintaining a sleek and professional look, the connectivity of components needed to be designed accordingly.

With some existing helmet ventilation being removed due to the space constraints discussed in section 4.2.1.1, it is essential to preserve unused ventilation cut-outs by placing the wiring around said cut-outs. Although the task seems simple, we must design the wiring to conform to all other design constraints and requirements of HelmetIQ.

These design constraints and wiring limitations can be mitigated by using *ribbon cables*. Ribbon cables offer several benefits for a compact design such as HelmetIQ. Their flat, thin profile allows them to fit into tight spaces, making them ideal for our purposes. Ribbon cables can also be easily folded and routed around components and ventilation cut-outs.

4.2.2. Durability and Weatherproofing

HelmetIQ was exposed to various environmental factors during its use, including sweat, rain, sun, temperature fluctuations, and vibration. These elements can potentially damage the integrated lighting, sensors, and electronic components, compromising the helmet's functionality and safety. To ensure the longevity and reliability of HelmetIQ, it is essential to address the constraints related to durability and weatherproofing.

One of the primary concerns is water resistance. The helmet must be designed to withstand exposure to rain and sweat without allowing moisture to penetrate the electronic components. To achieve this, HelmetIQ should incorporate waterproof or water-resistant materials for the outer shell and ensure that all openings and joints are properly sealed. Gaskets, O-rings, and waterproof sealants can be used to prevent water ingress around buttons, charging ports, and other potential entry points. The PCB and other sensitive components should be coated with conformal coatings to provide an additional layer of protection against moisture.

In addition to water resistance, HelmetIQ must also be designed to resist corrosion. Exposure to sweat, rain, and humid environments can lead to corrosion of metallic components, such as connectors and contacts. To mitigate this issue, corrosion-resistant materials, such as stainless steel or gold-plated contacts, should be used for critical connections. Additionally, applying anti-corrosion coatings or using sealed connectors can further enhance the helmet's resistance to corrosion.

Prolonged exposure to sunlight and UV radiation can degrade plastics and other materials used in the helmet's construction. To prevent UV degradation, HelmetIQ should incorporate UV-resistant materials for the outer shell and any exposed components. UV stabilizers can be added to plastics to improve their resistance to sunlight, and UV-resistant coatings can be applied to protect sensitive components.

The choice of connectors and cables is crucial for ensuring the durability and reliability of HelmetIQ. The connectors should be rated for the expected environmental conditions, such as temperature range and moisture exposure. Sealed or waterproof connectors can provide an additional layer of protection against water ingress. The cables should be flexible, durable, and resistant to abrasion and repeated bending. Using strain relief features at cable entry points can help prevent damage caused by excessive bending or pulling.

Potting and sealing techniques can be employed to further enhance the durability and weatherproofing of HelmetIQ's electronic components. Potting involves encapsulating the PCB and other sensitive components in a protective resin or compound, which provides insulation, shock absorption, and moisture resistance. Sealing techniques, such as applying silicone sealants or using gaskets, can be used to create watertight seals around openings and joints.

To validate the durability and weatherproofing of HelmetIQ, rigorous testing should be conducted. This may include water resistance tests, vibration tests, and temperature cycling tests. These tests help identify potential weaknesses and allow for design improvements to be made before the final revision of HelmetIQ is produced.

By addressing the constraints related to durability and weatherproofing, HelmetIQ can be designed to withstand the demanding conditions encountered during regular use. Careful material selection, sealing techniques, and thorough testing ensured that the helmet's electronic components remain functional and reliable, providing users with a safe and dependable smart cycling experience.

4.2.3. Battery Constraints

The limited space for the battery within the helmet's cutouts necessitates careful consideration of battery technology and power management. Additionally, the selected battery must also have a form factor that fits within the designated cutout and allows for easy replacement when necessary.

To maximize battery life and ensure reliable performance, efficient power management techniques should be implemented. This can include the use of low-power components, power-saving modes for various functions, and intelligent algorithms that optimize power consumption based on usage patterns. Additionally, the mobile app can provide battery status information and alerts to the user, reminding them to charge the helmet when the battery level is low.

Considering the battery constraints of the user's smartphone is also important. The mobile app should be designed to minimize its impact on the phone's battery life. Techniques such as background processing optimization, efficient data transfer, and user-configurable power-saving settings can help mitigate the app's battery consumption.

4.2.4. Weight Constraints

In addition to the weight distribution constraints discussed in section 4.2.1.2, the overall weight of the HelmetIQ must be carefully considered. Adding electronic components and a battery to the helmet inevitably increases its weight compared to a standard bike helmet. However, it is crucial to minimize this weight increase to ensure user comfort and reduce neck strain during extended use.

To address this constraint, lightweight materials should be used for the PCB, wiring, and any additional structural components. The battery should also be carefully selected to balance capacity and weight. Lithium-polymer (LiPo) batteries are a good choice due to their high energy density and lightweight nature.

Furthermore, the weight of the HelmetIQ should be evenly distributed to prevent any imbalance or discomfort for the user. This can be achieved through careful placement of components and the use of counterweights, if necessary, as mentioned in section 4.2.1.2.

Selecting a lightweight base helmet to build upon can help alleviate some of these weight constraints. The market is dense with lightweight helmet options, and we obviously did not have the opportunity to test all of them so we lived with the selection we make based on the research that we have done. Looking at customer reviews and video reviews of different helmets can allow us to evaluate more potential base helmets without necessarily getting our hands on them.

4.2.5. Communication Constraints

In order to limit the production cost and development time of HelmetIQ, the physical product only use Bluetooth as its main form of communicating with the mobile app. Because the mobile app is used on a smartphone, we can leverage the communication capabilities of the smartphone to implement all internet-related functionality of HelmetIQ. This includes text alerts in the event of a collision.

Although this implementation solves the limitations of only having a Bluetooth module on HelmetIQ, if the user does not have a smartphone the functionality of HelmetIQ is limited. Though some functions are automatic, such as the automatic headlights and break lights, the text messages sent in the event of a possible collision would not be available, in addition to the other functionality offered by the mobile app.

4.2.6. Usability Constraints

While the HelmetIQ offers numerous advanced features, it is essential to ensure that the helmet remains user-friendly and intuitive to operate. The integration of electronics and the mobile app should not compromise the ease of use or require extensive training for the user.

To address usability constraints, the HelmetIQ should have clearly labeled buttons or touch-sensitive areas for activating features like turn signals and emergency notifications. The mobile app should also have a clean and intuitive user interface that allows for easy pairing, customization, and control of the helmet's functions.

Moreover, the HelmetIQ should be designed to accommodate different head sizes and shapes comfortably. Adjustable straps and padding can help ensure a secure and comfortable fit for a wide range of users.

4.2.7. Cost Constraints

The development and production of the HelmetIQ involve additional costs compared to traditional bike helmets due to the integration of electronic components, sensors, and the development of the mobile app. However, to make the HelmetIQ accessible to a wide range of consumers, it is important to manage these costs effectively.

One approach to managing cost constraints is to leverage existing technologies and components whenever possible. Using off-the-shelf sensors, Bluetooth modules, and batteries that are readily available can help reduce development costs and time-to-market.

Additionally, optimizing the design for manufacturing and assembly can help minimize production costs. This involves designing the helmet with fewer parts, using standardized components, and considering the ease of assembly during the design process. Collaborating with experienced manufacturing partners and leveraging economies of scale can also help control costs.

4.2.8. Manufacturing Constraints

As HelmetIQ originates from a Senior Design project, it is not practical for us to go about the means of finding a way to design and manufacture a bike helmet base to work upon. Because of this, we have the constraint of having to make another company’s design for a complete product work for us. If it were possible we would develop a CAD model of a helmet that has cutouts for all of our ribbon cable and slots for the PCB and other sensors to be inserted. Unfortunately, since we do not have this luxury, we are constrained to the designs and helmet architecture that exists on the market currently.

Aa major factor discussed in the earlier comparison sections was about how easy a helmet base would have be to modify. This is something we looked at closely because certain helmet styles like a BMX or full-face helmet are made with a thick outer layer of plastic that help prevent against more extreme collisions. This outer shell is not meant to be penetrated and with our current plans to have the PCB slotted into a cutout the top of the helmet, these were not feasible options.

Our current plan is to use the commuter bike helmet base since it has the thinnest layer of hard plastic on the top and cut out a slice of the top foam below that plastic as the place to store our PCB.

Within the commuter bike helmet space there are hundreds of different options to choose from and beside the cost constraint we have had to think about what happens if we mess up the cutting of a helmet and have to get another. This has led us to be more conservative in the price of the helmet that we used to build because of the inevitability that an error somewhere would occur.

All these problems stem from the fact that we cannot manufacture the base of a helmet to our specifications. This is going to be the largest hurdle we have to overcome in the building process because the electronics and sensors worked but implementing them onto a bike helmet profile was a challenge.

5. Large Language Models

In the sector of education, particularly in a capstone course like Senior Design, leveraging AI language models can provide significant benefits. These platforms can assist in various aspects of a project, from ideation and planning to documentation and even some problem-solving. However, they also come with several limitations and potential drawbacks. This section compares the limitations, pros, and cons of several AI language models and provide examples to illustrate their impact on the learning experience in Senior Design.

5.1. Limitations, Pros, and Cons of Large Language Models (LLMs)

5.1.1. Pros

1. Idea Generation: LLMs can serve as valuable brainstorming tools, helping students generate ideas and explore different perspectives on a given topic. By inputting prompts or questions, students can receive diverse and creative responses, stimulating their thought process and inspiring novel approaches to their projects.

2. Research Assistance: LLMs can aid in research tasks by quickly summarizing large volumes of text, extracting key information, and providing insights on specific topics. This can save students significant time and effort in sifting through extensive literature, allowing them to focus on critical analysis and synthesis of information.

3. Writing Support: LLMs can assist students in various aspects of report writing, such as generating outlines, suggesting relevant topics to cover, and providing examples of well-structured sentences and paragraphs. They can also help with proofreading and editing, identifying grammatical errors and suggesting improvements to enhance clarity and coherence.

5.1.2. Cons

1. Lack of Deep Understanding: While LLMs can generate coherent and contextually relevant responses, they lack true understanding of the subject matter. They operate based on patterns and associations learned from vast amounts of data, but they don't possess the deep knowledge and critical thinking skills that humans possess. This limitation can lead to superficial or misleading outputs if not carefully evaluated.

2. Potential for Bias and Misinformation: LLMs are trained on large datasets that may contain biases and inaccuracies. If students rely solely on LLM-generated content without fact-checking or critical evaluation, they risk incorporating biased or incorrect information into their work. It's crucial for students to verify the accuracy and credibility of LLM outputs using reliable sources.

3. Overreliance and Intellectual Laziness: The ease and convenience of using LLMs may tempt students to overly rely on them, potentially hindering the development of their own critical thinking, problem-solving, and research skills. It's important for students to strike a balance between leveraging LLMs as tools and actively engaging in the learning process themselves.

5.1.3. Limitations

1. Domain-Specific Knowledge: LLMs are trained on broad datasets and may not have in-depth knowledge of specific domains or niche topics. For highly specialized areas, such as cutting-edge scientific research or industry-specific jargon, LLMs may struggle to provide accurate and nuanced responses.

2. Lack of Common-Sense Reasoning: LLMs can sometimes generate responses that lack common sense or contextual understanding. They may produce outputs that are grammatically correct but semantically inconsistent or illogical. Students need to critically evaluate LLM outputs and apply their own reasoning to ensure the validity and coherence of the information.

3. Inability to Provide Original Insights: While LLMs can generate novel combinations of ideas, they are ultimately limited by the data they are trained on. They cannot provide truly original insights or groundbreaking ideas that go beyond the scope of their training data. Students should use LLMs as a complement to their own creativity and critical thinking rather than relying on them for original research contributions.

In conclusion, LLMs offer valuable assistance to students in Senior Design courses by facilitating idea generation, research, and writing support. However, it's crucial for students to be aware of the limitations and potential drawbacks, such as the lack of deep understanding, potential for bias, and the risk of overreliance. By using LLMs judiciously, fact-checking their outputs, and actively engaging in the learning process, students can harness the benefits of these tools while mitigating their limitations.

5.2. ChatGPT and Claude

ChatGPT and Claude are two LLMs developed by OpenAI and Anthropic, respectively. ChatGPT was introduced in 2022 as part of the GPT (Generative Pre-trained Transformer) series. It is built on the GPT-4 architecture, which is the latest model offered by OpenAI. Claude was introduced in 2023, with its latest model Claude 3 Opus. Both models are designed to generate human-like text based on the input they receive from users and can perform relatively well on a variety of tasks, including answering complex questions, composing texts, summarizing information, writing code, and more. This makes them applicable to all fields in one way or another.

In a course such as Senior Design, the use of LLMs such as ChatGPT and Claude can facilitate enhanced writing quality, process comprehension, research and development, and more. However, just like physical tools such as a hammer or screwdriver when it comes to completing a job, different LLMs have different use-cases, and are better at different things. The following section compares the differences between ChatGPT and Claude in three categories. The first category is based on creativity – asking the LLMs about potential ideas for a Senior Design project. The second category is testing the LLMs understanding of legal regulations and guidelines for bike helmets. The third category is testing the LLMs ability to put together and break down the entire process of a smart bike helmet project and the components it involves.

5.2.1. Idea Generation

To give ChatGPT and Claude the best chance of outputting results of the desired quality, it must have useful context to the request in question. In this scenario, we want to generate ideas for our Senior Design project. Although the request is simple, the requirements for a project such as that in Senior Design are not. The prompt that ChatGPT and Claude were given are located in *Appendix B.* The output received from both ChatGPT and Claude is located in *Appendix C, Section 1a* and *Section 1b* respectively.

5.2.1.1. Results Comparison

The project ideas generated by ChatGPT and Claude for a Senior Design project share many similarities, focusing on common themes such as IoT, robotics, drones, smart systems, and wearables. While the concepts are relevant and technically feasible, they lack originality and struggle to demonstrate out-of-the-box thinking.

It is important to recognize that language models like ChatGPT and Claude have limitations when it comes to creativity. Those limitations were outlined in section 5.1.3 and were confirmed to be present in this example. Both are trained on vast amounts of existing data, which allows them to identify patterns and make associations but hinders their ability to generate truly innovative ideas.

For a Senior Design project, the ideas can certainly be used as a starting point for further ideation and discussion. By combining the models' suggestions with personal creative thinking and domain expertise, the team can develop a truly innovative and impactful Senior Design project that goes beyond the somewhat generic and repetitive ideas provided by the language models.

5.2.2. Bike Helmet Regulations and Guidelines

When building a bike helmet such as ours, it is important for the modified helmet to satisfy all federal safety regulations for a bike helmet. In this example, ChatGPT and Claude were both asked what regulations were required for bike helmets. The prompt given to ChatGPT and Claude was the following:

"What Federal / State safety regulations are required for a bike helmet in the state of Florida?"

The output for both ChatGPT and Claude is located in *Appendix C, Section 2a* and *Section 2b* respectively.

5.2.2.1. Results Comparison

The results from both ChatGPT and Claude are both useful, yet quite different in terms of content. ChatGPT took a broader approach to the question and gave me rules such as all cyclists under the age of 16 must wear a helmet, the helmet must have a sticker from the Consumer Product Safety Commission (CPSC), and some legal implications for not wearing a helmet. Claude took the question much more directly and provided the key requirements and specifications a bike helmet must meet – impact protection, 105 degrees of peripheral vision on both sides, retention system that prevents the helmet from coming off in a crash, and labeling.

While both ChatGPT and Claude provided the correct federal regulation (16 C.F.R. part 1203), Claude went into the actual CPSC standards and provided the necessary specifications that our modified bike helmet must be able to meet. From an engineering standpoint, Claude's result could easily be considered more applicable versus the response from ChatGPT, however given both LLMs had no context as to what kind of response would be desirable, ChatGPT and Claude both did a good job at answering the question and would provide a great starting point for someone who doesn't know anything about bicycle helmet safety regulations.

5.2.3. Technical Aspects of a Smart Bike Helmet

The third experiment asks about the technical aspects of implementing a smart bike helmet. This experiment provides valuable insight into how different LLMs can assist with the engineering and design process. The prompt given to both ChatGPT and Claude is the following:

"What are the key components and technologies needed to create a smart bike helmet with intelligent lighting, safety features, and mobile app connectivity? Please provide a high-level system architecture and explain how these components would work together."

The output for both ChatGPT and Claude is located in *Appendix C*, *Section 3a* and *Section 3b* respectively.

5.2.3.1. Results Comparison

The results received from both LLMs provides a very detailed summary of a smart bike helmet project, and the components and features involved. The response from ChatGPT provides a comprehensive and well-structured answer. It breaks down the key components and technologies into clear categories. The response from Claude also provides a good overview but with a slightly different structure.

Overall, ChatGPT provides a better, more detailed response to the query. It offers a more comprehensive breakdown of the components, technologies, and their interactions. The explanation of the system architecture is more thorough, giving a clearer picture of how the smart helmet would function as a complete system.

While Claude provides some unique points (like mentioning the helmet shell and cloud backend), it doesn't go into much detail about the functioning of each component and their interactions. ChatGPT's response would likely be more valuable to someone looking to understand or implement such a system.

6. Hardware Design

6.1. Power System

Figure 6.1 *– Power System*

The system is powered by a DC input, which can accept various voltage levels up to 24V. The input voltage is connected through a DC jack (J1), providing an easy interface for connecting an external power source. The power is then routed through a switch (S1), which allows the user to turn the power on and off manually. The switch ensures that the system can be easily powered down when not in use, conserving battery life and protecting the components. There is also an LED that lights when the system is receiving power.

A diode (D1, 1N4007) is placed immediately after the switch to protect the system from reverse polarity. This diode ensures that if the power supply is accidentally connected backward, the circuit is not damaged. The 1N4007 is chosen for its high current handling capability and low forward voltage drop.

The voltage regulator (U1, TPS62152RGTR) steps down the input voltage to a stable 3.3V, which is required by all the components in our system. TheTPS62152RGTR is a low dropout voltage regulator, capable of providing a consistent 3.3V output with minimal input-to-output voltage difference. A 10µF capacitor (C18), 22uF capacitor (C16) and 390pF (C15) are placed at the input of the regulator to filter out any noise or fluctuations from the power supply. Similarly, a 2.2uH inductor (L5) is placed at the output to stabilize the 3.3V supply and filter any noise generated by the regulator itself.

6.2. Haptic Driver System

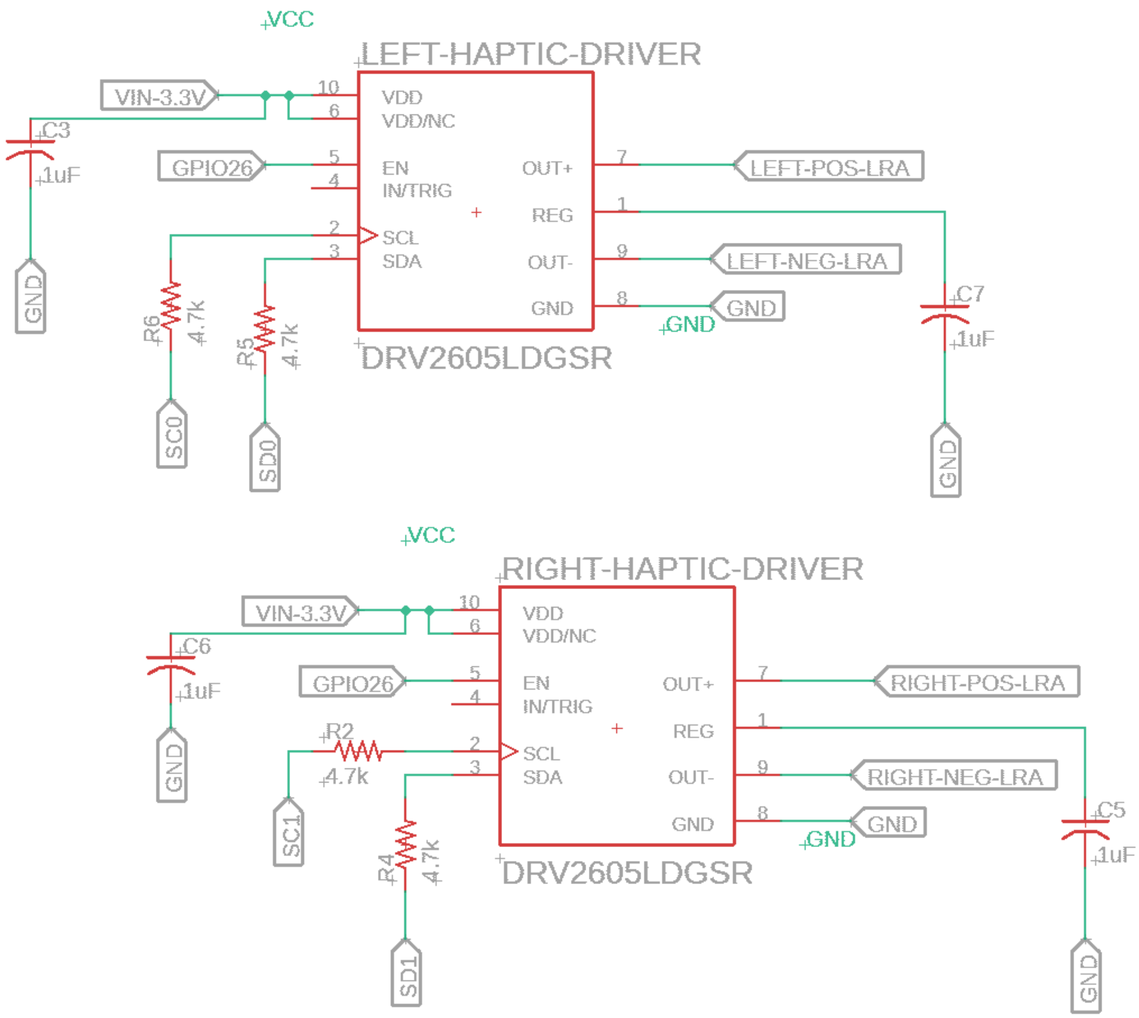

***Figure 6.2** – Haptic Drive System*

The haptic feedback system in our project uses DRV2605L haptic drivers to control ERM motors, providing tactile feedback to the user. The system is designed to independently control two haptic drivers for the left and right sides of the helmet, connected to a multiplexer. The OUT+ and Out- pins are connected independently to the ERM motor, near the capacitive touch sensor on the left and right side of the helmet.

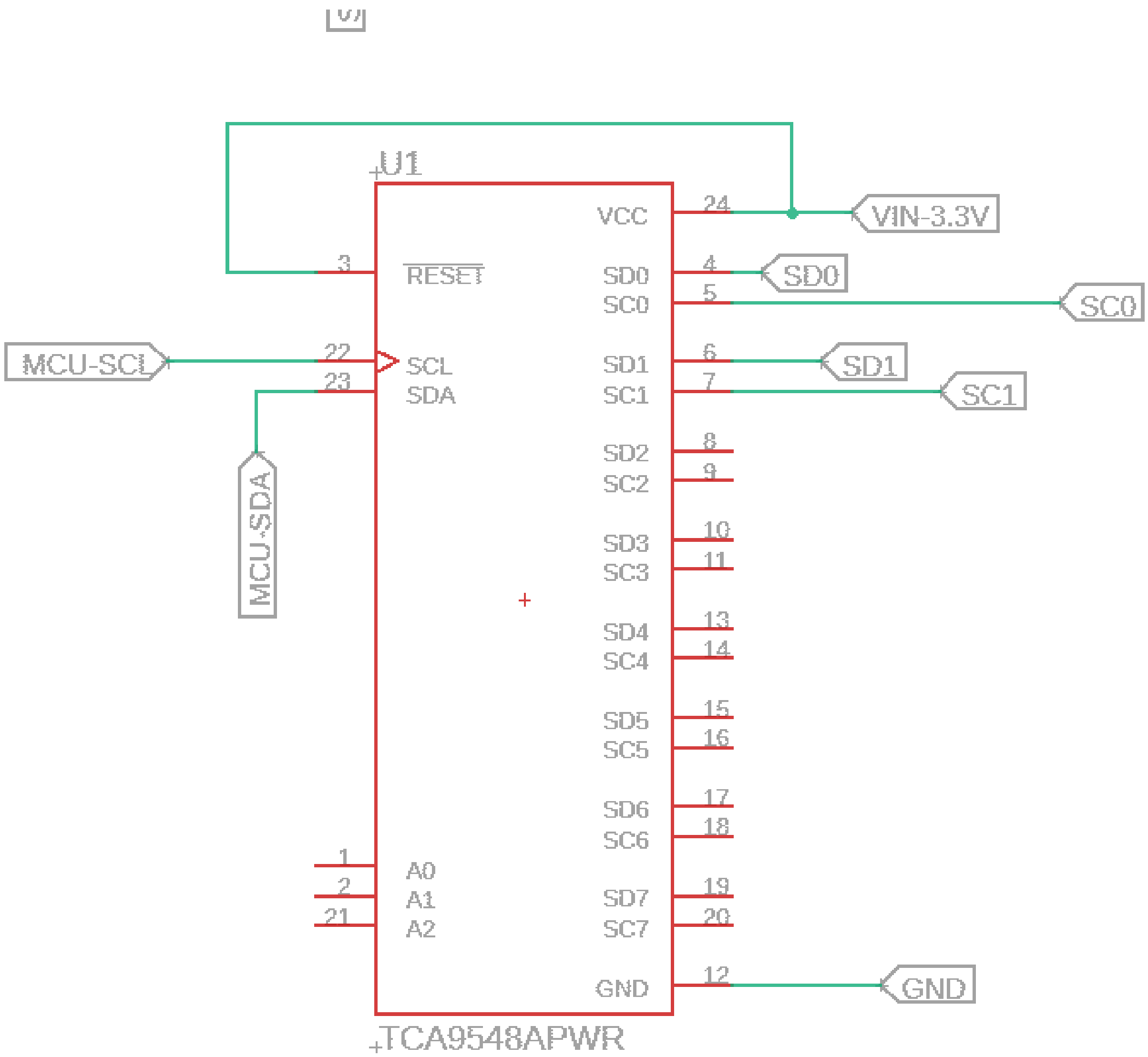

Figure 6.3 *– 8-channel I2C Multiplexer/switch*

The TCA9548A I2C multiplexer is used in our system to manage multiple I2C devices with the same address, such as the DRV2605L haptic drivers. This allows us to control the haptic feedback motors independently.

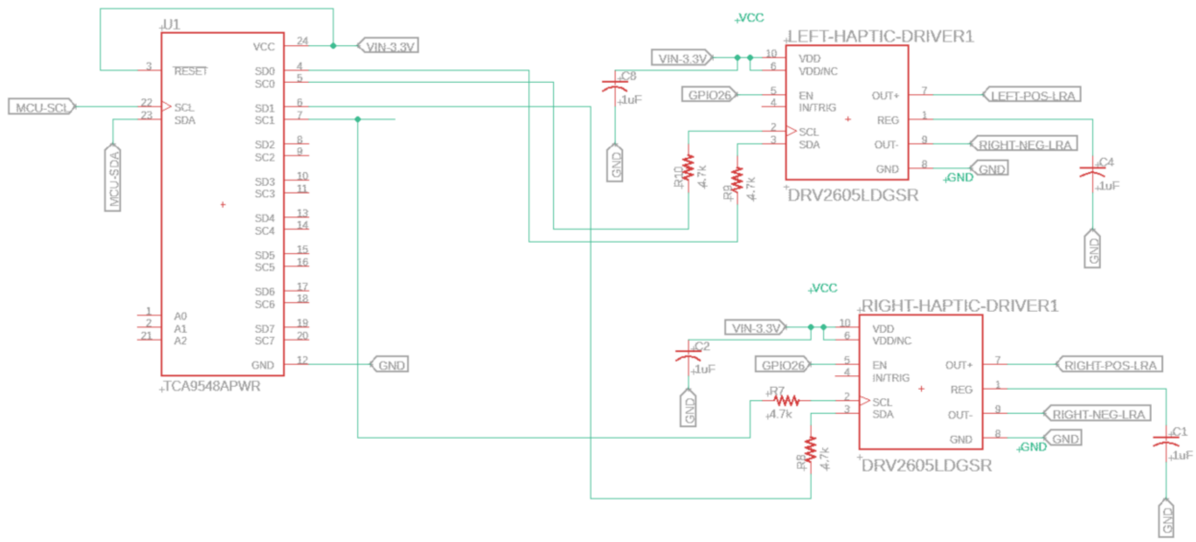

***Figure 6.4** – Overview I2C Connection for Haptic Drivers*

Two DRV2605L haptic drivers are used to control the ERM motors for providing haptic feedback. Each haptic driver is connected to one of the multiplexer channels (SD0 and SD1) and communicates with the ESP32 through the I2C bus. Decoupling capacitors (0.1µF) are placed near the power pins of the haptic drivers to filter out noise and stabilize the power supply. Pull-up resistors (4.7kΩ) are connected to the SDA and SCL lines of each haptic driver to ensure proper logic levels.

6.2.1. LED Drivers

The headlight driver circuit uses the MCP1643 LED driver to control the brightness and power of the helmet's headlights. The schematic diagram illustrates the connections. By using dedicated LED drivers for the headlights, turn signals, and brake lights, our system ensures that each lighting component operates efficiently and reliably.

The headlight driver circuit utilizes the MCP1643 LED driver to manage the headlights' power and brightness. The schematic illustrates the connections required for this setup. Key components include decoupling capacitors (C1 and C4), which are placed close to the driver to filter out noise and stabilize the voltage supply. The LED driver is connected to the ESP32 microcontroller, which sends control signals to adjust the headlights' brightness based on ambient light conditions detected by the VEML7700-TR light sensor.

The brake light system uses the AP3019AKTR LED driver to manage the brake LEDs. This driver is designed to provide a stable and responsive light output when the ICM-42670 sensor detects deceleration. The schematic includes connections for decoupling capacitors (C3 and C4) and a resistor (R3) to ensure reliable operation. The driver is connected to the ESP32, which controls the brake lights based on input from the motion sensor.

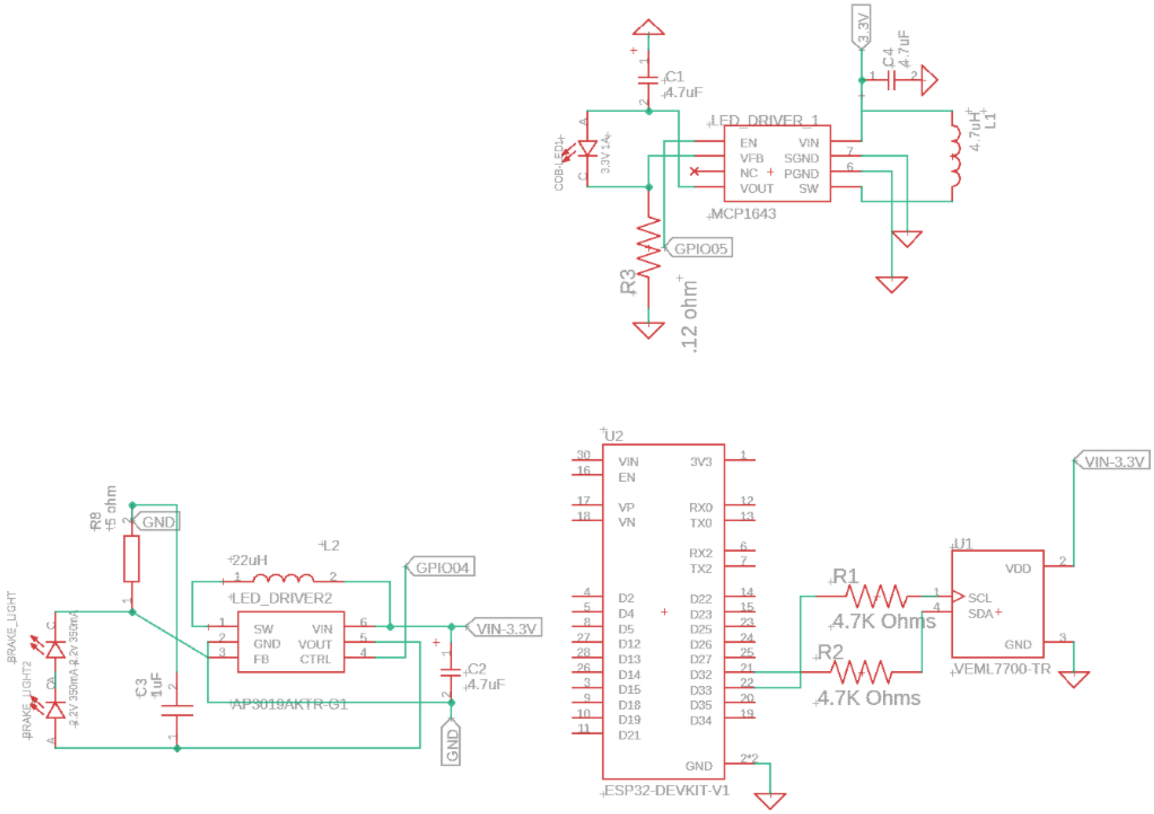

Figure 6.5 *– LED Driver for Headlight and brake light, Light Sensor*

For the turn signals, the LTC3490ES8PBF LED drivers are used. These drivers are responsible for providing stable power to the LEDs, ensuring they operate correctly when activated by the capacitive touch sensors. The schematic shows the connection of the turn signal drivers, including the placement of decoupling capacitors (C1, C2, and C4) to ensure smooth operation and reduce electrical noise. Each driver is connected to a GPIO pin on the ESP32, allowing the microcontroller to control the left and right turn signals independently.

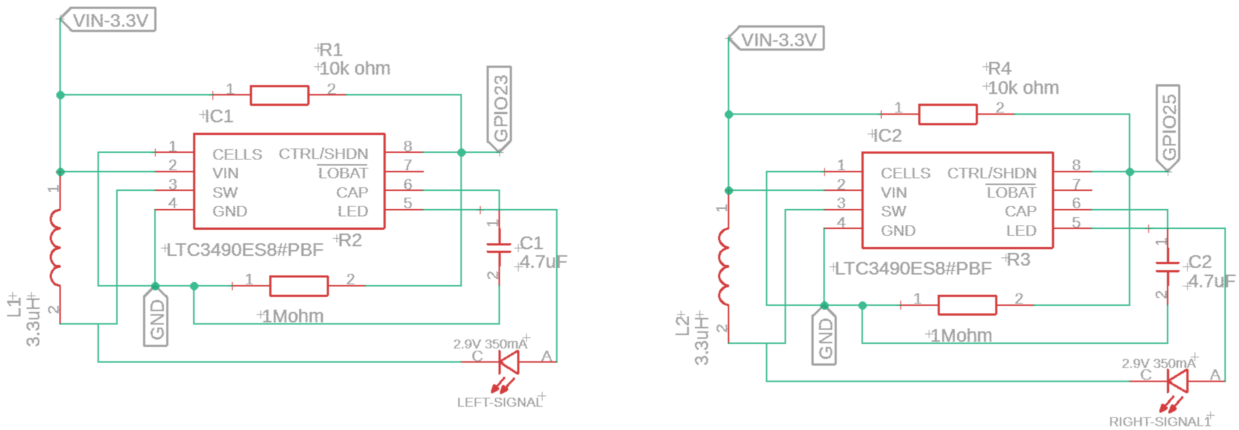

***Figure 6.6** – LED Drivers for Turn Signals*

6.2.2. Accelerometer and Gyroscope System

The figure below illustrates the ICM-42670, an integrated 6-axis Accelerometer and Gyroscope inertial measurement unit (IMU). This sensor combines a 3-axis gyroscope and a 3-axis accelerometer, providing comprehensive motion tracking capabilities. The ICM-42670 communicates using the I2C protocol, utilizing the SCL (Serial Clock Line) and SDA (Serial Data Line) for data transmission. In our schematic, the SCL and SDA lines are connected to the appropriate pins on our MCU (Microcontroller Unit) along with 4.7kΩ pull-up resistors to ensure a stable signal and prevent data corruption. The pull-up resistors are crucial for maintaining proper I2C communication, especially in environments with potential electrical noise.

To facilitate basic functionality, only essential pins are connected, and the unused pins are appropriately managed. The AUXCL and AUXDA pins, which are auxiliary I2C lines for interfacing with additional sensors, are left unconnected since they are not required for our application. Similarly, the RESV (reserved) pins are left open as they have no defined function in this configuration.

The AD0 pin is set to ground, which configures the sensor to use its default I2C address of 0x68. This address setting simplifies the integration process and ensures compatibility with standard I2C communication protocols used in our MCU. The INT (interrupt) pin is connected to a GPIO (General Purpose Input/Output) pin on the MCU. This connection allows the sensor to trigger an interrupt signal when specific events occur, such as detecting a collision or significant movement, enhancing the responsiveness of our collision detection system.

Additionally, decoupling capacitors (0.1µF) are placed near the power supply pins of the ICM-42670 to filter out any noise from the power lines and ensure stable operation. These capacitors are essential for minimizing voltage fluctuations and maintaining accurate sensor readings.

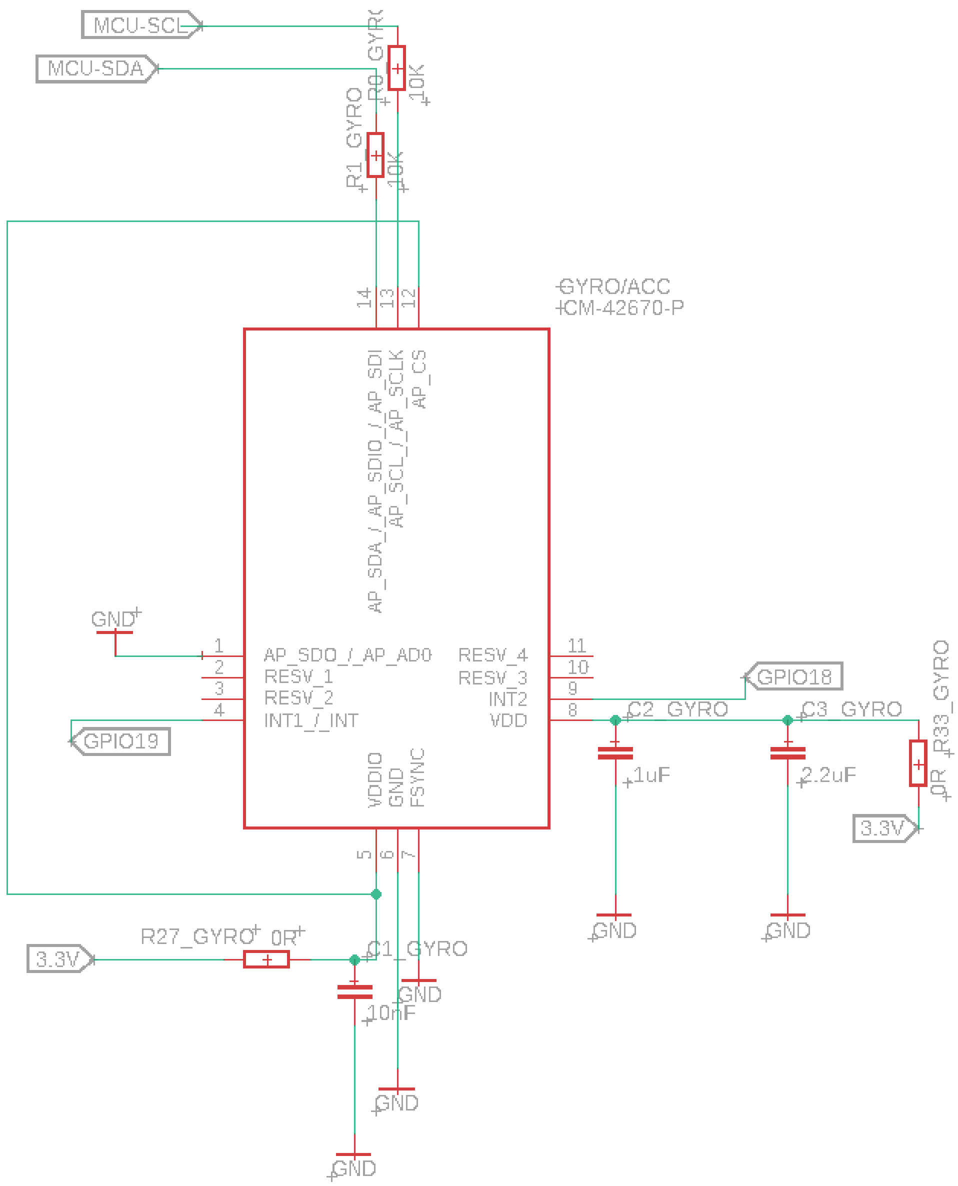

Figure 6.7 – *Gyroscope/Accelerometer*

6.3. MCU

The ESP32 microcontroller serves as the central processing unit for the smart helmet system, interfacing with various sensors, drivers, and other peripherals. The schematic of the ESP32 microcontroller includes the following connections:

Power Supply:

- VIN: Connected to a 3.3V voltage regulator output, ensuring stable power supply to the ESP32.

- 3V3: The internal 3.3V output of the ESP32 used for powering other peripherals if needed.
- GND: Ground connections ensuring a common ground for the entire system.

Peripheral Connections:

- GPIO Pins: The ESP32 has multiple Input/Output (GPIO) pins, used to interface with sensors, haptic drivers, LEDs, and other components. These pins are programmable and can serve various functions such as digital input, digital output, PWM output, and more.

I2C Communication:

- SCL (GPIO22): I2C clock line used to synchronize data transfer between the ESP32 and I2C devices like the ICM-42670 gyroscope/accelerometer, light sensor, and the TCA9548A I2C multiplexer.
- SDA (GPIO21): I2C data line for bi-directional data transfer between the ESP32 and connected I2C devices.

UART:

- TX (GPIO1), RX (GPIO3): Used for serial communication with external devices for debugging or additional connectivity.

Enable Pins:

- EN (GPIO16): Enable pin used to reset or wake up the ESP32 from a low-power state.

Peripheral Configuration:

- INT (GPIO13): Interrupt pin connected to sensors like the ICM-42670 to signal the ESP32 for immediate processing.
- GPIO Pins: Configured for various inputs and outputs such as control signals for LED drivers, touch sensors, and haptic feedback drivers.

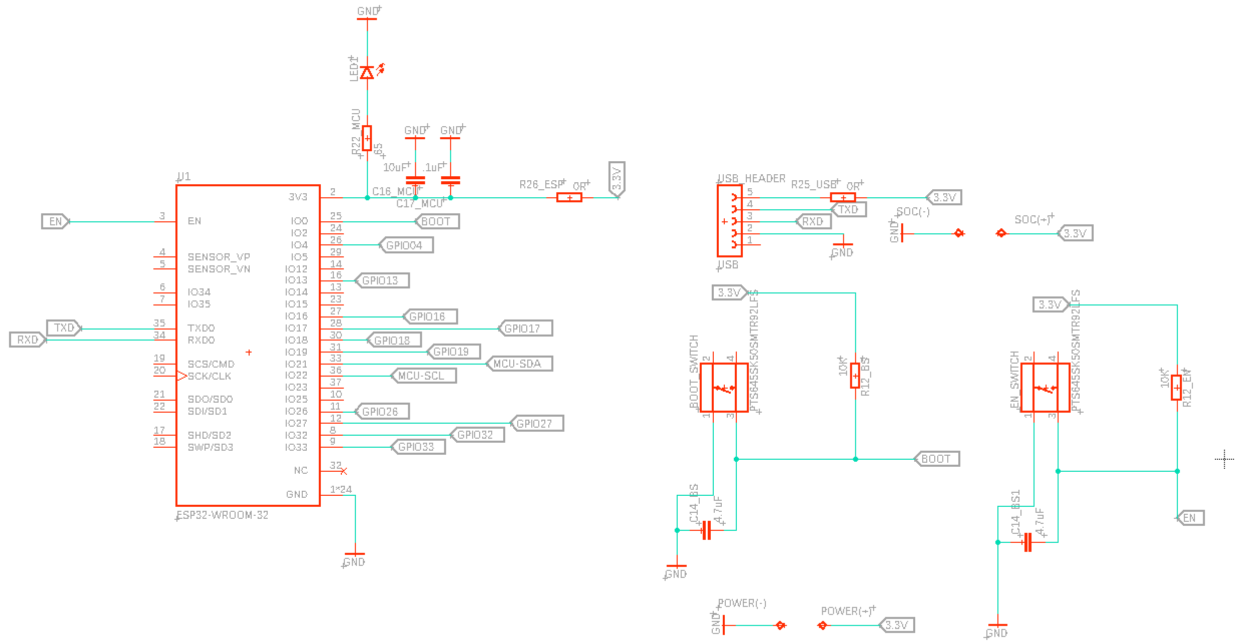

Figure 6.8 *– MCU Connections*

6.4. Lower-Level Subsystem

The architecture of the smart helmet system is designed to integrate all the sensors, actuators, and the MCU in a compact and efficient manner. The ESP32 serves as the brain of the system, coordinating all operations. The structure is designed to ensure reliable data communication, power distribution, and user feedback.

- Voltage Regulators: Positioned on it own PCB the power input to convert the input voltage to a stable 3.3V required by most of the components.
- MCU (ESP32): Placed to minimize the length of connections to various peripherals, ensuring low latency and reduced signal interference. The MCU is placed near the end of the board for easy connection to the USB, when needed.
- Decoupling Capacitors: Placed close to the power pins of critical components like the ESP32 and sensors to filter out noise and stabilize the voltage supply.
- Haptic Drivers and LED Drivers: Positioned near the edge of the PCB for easy connection to the external actuator, touch sensors, and LEDs.
- Sensors: Placed to minimize interference and ensure accurate data collection. For example, the gyroscope/accelerometer is placed centrally to accurately capture helmet movements.

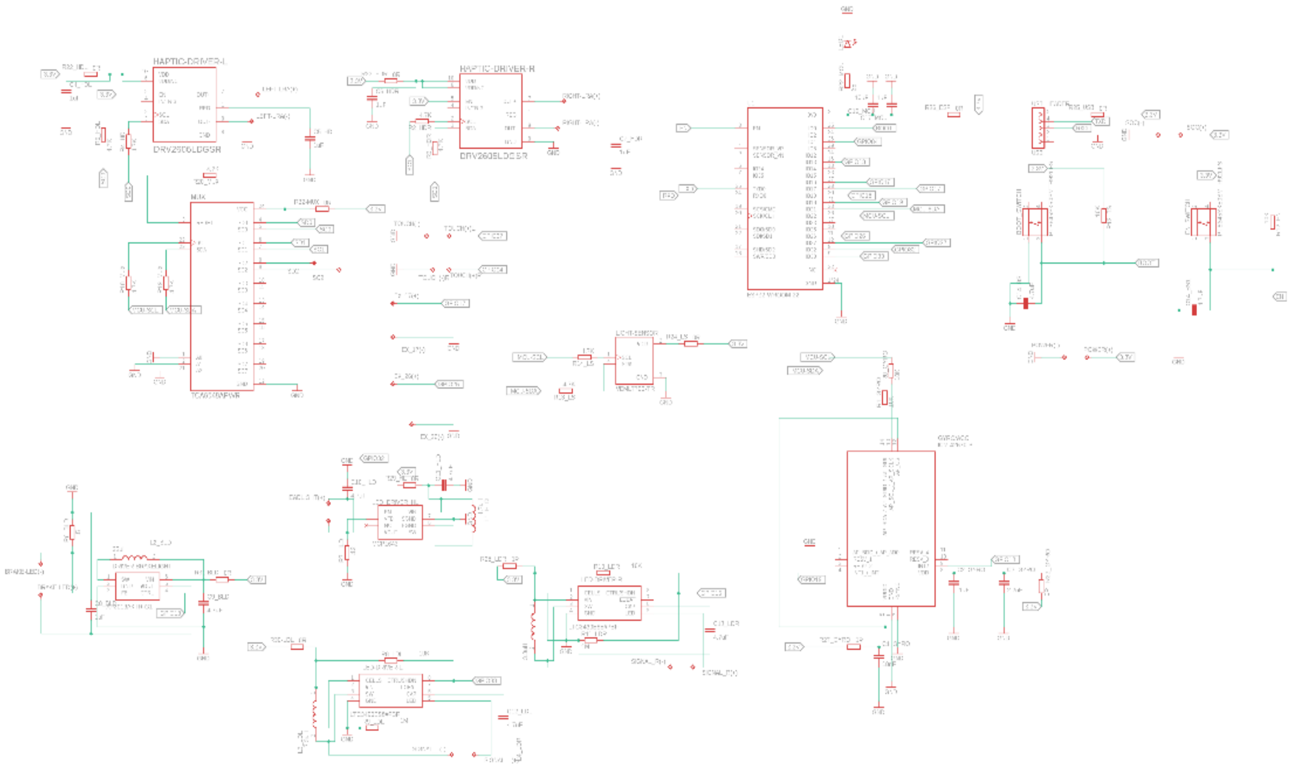

Figure 6.9 *– Schematic Overview*

6.5. External Sensor Connections

In the smart helmet project, several external sensors and actuators, including LEDs for turn signals and brake lights, haptic feedback motors, and capacitive touch sensors, need to be connected to the main PCB. Connector pads are placed near the edge of the PCB, serving as solder points for wires that connect to the external components. The placement and labeling of these pads are designed to ensure ease of assembly and reduce the risk of miswiring.

Connector Pads Layout:

- Connector pads are located at the periphery of the PCB to allow for straightforward soldering of wires.
- Each pad is clearly labeled to indicate the specific component it connects to (e.g., "LEFT-SIGNAL", "BRAKE-LIGHT", "ERM-MOTOR").
- Connector pads are grouped based on their function to minimize routing complexity.

Capacitive Touch Sensors:

- Touch Sensor Output (Q): Connected to a GPIO pin on the ESP32.
- VDD: Connected to the 3.3V output from the voltage regulator.
- GND: Connected to the ground plane.

LEDs for Turn Signals and Brake Lights:

- Anode (+) of LED: Connected to the output of the LED driver on the PCB.
- Cathode (-) of LED: Connected to ground.
- Series Resistor: Placed between the LED driver output and the LED anode to limit current.

Haptic Feedback Motors (ERM):

- OUT+ and OUT- of DRV2605L: Connected to the ERM motor terminals.
- VDD: Connected to the 3.3V output from the voltage regulator.
- SCL and SDA: Connected to the I2C bus on the ESP32.

7. Software Design

7.1. Software Use Case Diagram

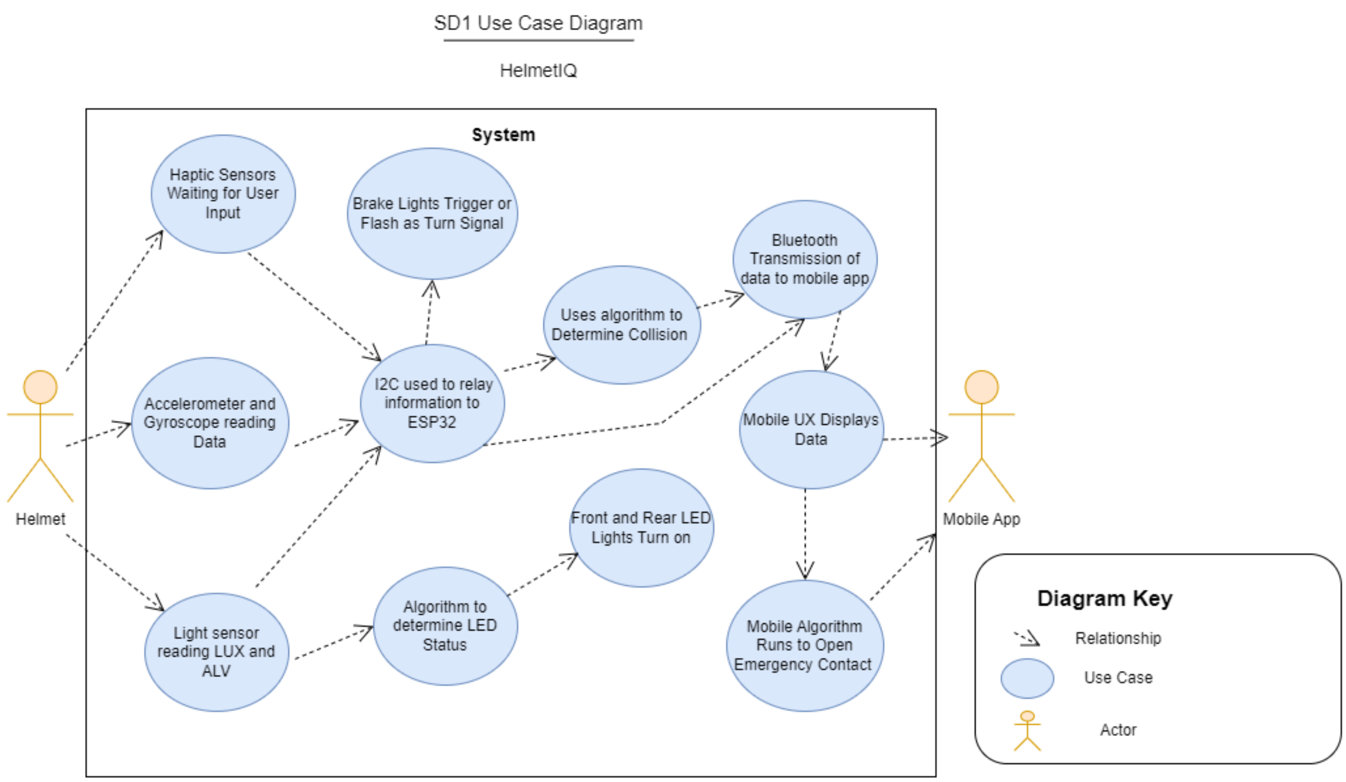

***Figure 7.1** – Software Use Case Diagram by Authors*

The provided use case diagram for the HelmetIQ system outlines the interactions and functionalities involving three main actors: the helmet, the mobile app, and the system itself. The helmet, equipped with an accelerometer, gyroscope, and light sensors, collects data on motion and light levels. This data is transmitted to the ESP32 via I2C communication. The system then processes this information using algorithms and interrupts to determine the status of LED lights, detect collisions, and trigger brake lights or turn signals accordingly. Additionally, haptic sensors on the helmet wait for user input to further interact with the system. The ESP32 also communicates with a mobile app via Bluetooth, allowing the app to display relevant data and, in case of a collision, run an emergency contact algorithm. This comprehensive setup ensures that the helmet not only enhances user safety with its real-time responses and alerts but also integrates seamlessly with mobile technology for enhanced user experience and emergency handling.

7.2. Software Activity Diagram

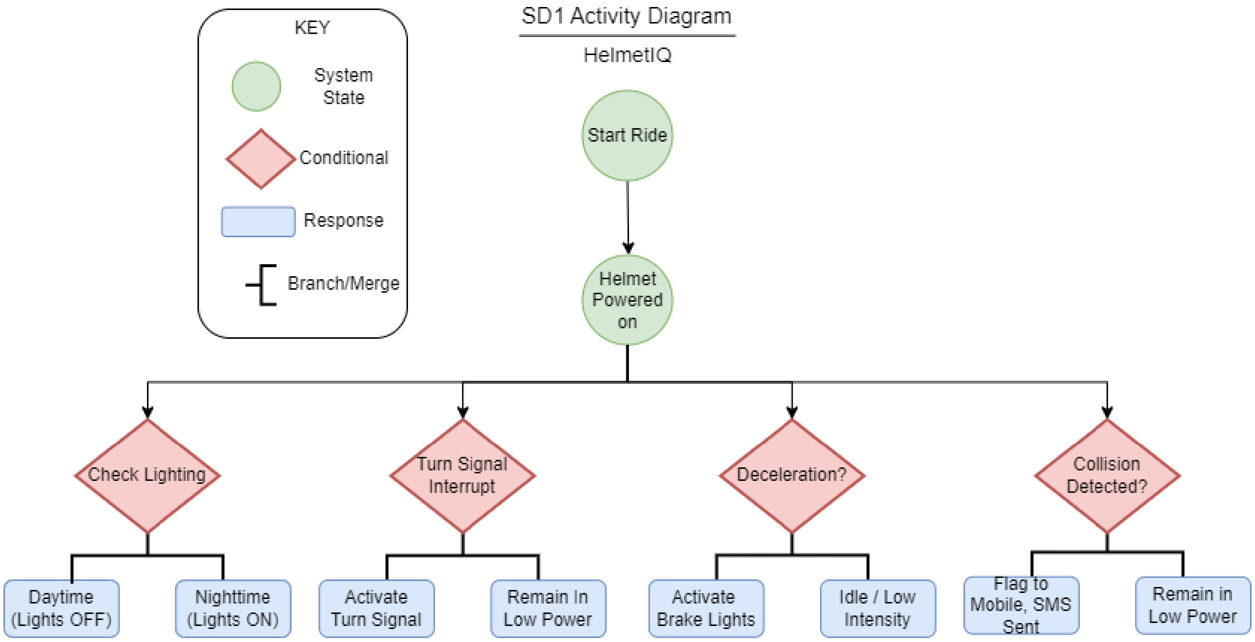

***Figure 7.2** – Software Activity Diagram*

The activity diagram for HelmetIQ represents the various processes and decision points that occur during a ride. The ride begins when the user powers on the helmet. The system continuously monitors the lighting conditions to determine whether it is daytime or nighttime. If it is daytime, the lights remain off; otherwise, the lights are kept on for nighttime visibility.

Throughout the ride, the ESP32 microcontroller remains in a waiting state, ready to respond to interrupts from various sensors. If the user activates the haptic feedback touch sensor, an interrupt is triggered, prompting the system to activate the turn signals accordingly. This allows for a more responsive and efficient handling of turn signal requests.

The system also employs an interrupt-driven approach to detect sudden changes in motion or freefall events, which may indicate a potential collision. When the accelerometer/gyroscope detects such an event, it triggers an interrupt, alerting the ESP32. In response, the ESP32 sends a signal to the mobile app, initiating a collision notification process.

Upon receiving the signal, the mobile app prompts the user to confirm whether they were involved in a collision or not. If the user does not respond within a predefined time frame, the mobile app takes a proactive step to ensure the user's safety. It automatically sends an SMS message to the user's designated emergency contacts, informing them of a potential collision and the need to check on the user's well-being.

The ride concludes when the user decides to end it and powers off the helmet. Throughout the ride, the HelmetIQ system intelligently manages the various safety and communication features, adapting to the user's actions and environmental conditions. By leveraging interrupts and real-time monitoring, the system ensures prompt responses to critical events, enhancing the overall safety and user experience.

The activity diagram effectively captures the complex interactions and decision-making processes within the HelmetIQ system. It showcases the system's ability to handle multiple tasks concurrently, respond to user input, and take appropriate actions based on sensor data. By incorporating interrupt-driven mechanisms and a robust mobile app integration, HelmetIQ provides a comprehensive and reliable solution for helmet safety and communication.

7.3. Lighting Response Diagram

The lighting response diagram in Figure 7.3 provides a comprehensive overview of how the HelmetIQ system adapts its lighting behavior based on different riding scenarios and environmental conditions. The diagram showcases the intelligent lighting control capabilities of the helmet, ensuring optimal visibility and safety for the rider.

During daytime operation, the system prioritizes energy efficiency while maintaining essential safety features. The rear brake lights remain off unless the user is decelerating, conserving battery power. When deceleration is detected, the rear brake lights illuminate at 100% brightness, employing a simple pattern blinking to draw attention to the rider's slowing down. This dynamic lighting response enhances the rider's visibility to other road users, promoting safer riding conditions.

In nighttime operation, the HelmetIQ system recognizes the need for heightened visibility. The rear brake lights remain steadily illuminated at a minimum of 75% brightness, even when the rider is not decelerating. This constant illumination ensures that the rider remains visible to other vehicles and pedestrians in low-light conditions. The rear brake lights also employ a steady blinking pattern, further increasing the rider's conspicuity. When the rider begins to decelerate during nighttime riding, the system intensifies the rear brake light to 100% brightness and switches to a more pronounced pattern blinking. This change in lighting behavior emphasizes the rider's deceleration, alerting nearby road users to the change in speed and reducing the risk of rear-end collisions.

In the event of a potential collision, the HelmetIQ system activates an emergency lighting response. The rear brake lights illuminate at maximum brightness (100%) and engage in an erratic blinking pattern. This intense and irregular lighting pattern is designed to capture the attention of surrounding road users, signaling the urgency of the situation. Additionally, the turn signals begin blinking simultaneously, functioning as supplementary hazard lights. The combination of the erratically blinking rear brake lights and the flashing turn signals creates a highly visible warning system, increasing the chances of other road users noticing the rider's situation and taking appropriate precautions.

The lighting response diagram effectively conveys the adaptability and intelligence of the HelmetIQ system in managing the helmet's lighting under various conditions. By dynamically adjusting the brightness, blinking patterns, and activation of different light components, the system optimizes visibility, energy efficiency, and safety. The diagram showcases how the helmet responds to daytime and nighttime riding, as well as potential collision scenarios, demonstrating its capability to enhance the rider's conspicuity and mitigate risks on the road.

Moreover, the lighting response diagram highlights the seamless integration of various sensors and control mechanisms within the HelmetIQ system. The system relies on accurate detection of deceleration events, ambient light conditions, and potential collisions to trigger the appropriate lighting responses. This integration ensures that the lighting behavior is automated and responsive to the rider's actions and the surrounding environment, minimizing the need for manual intervention and allowing the rider to focus on the road.

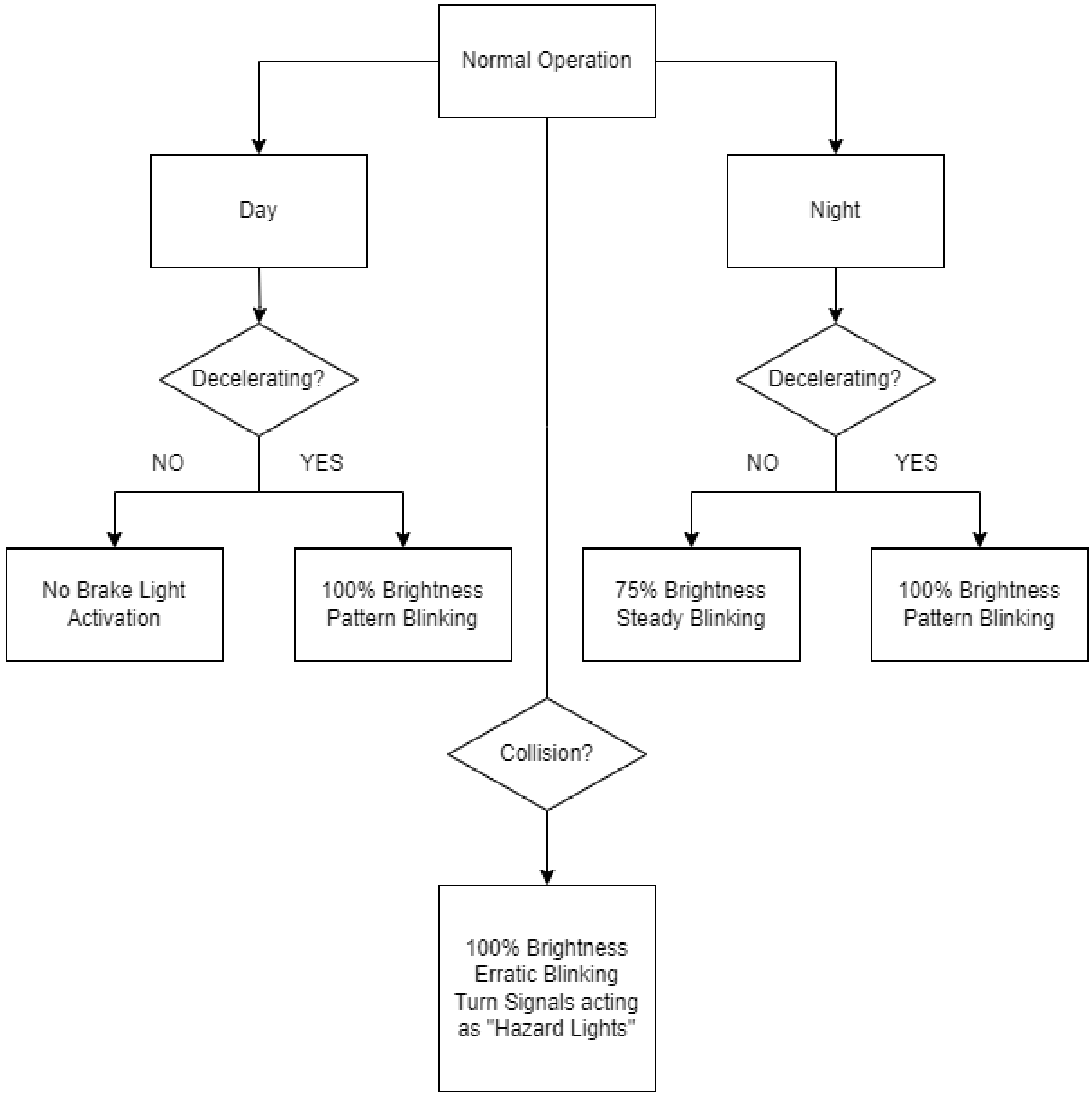

Figure 7.3 *– Rear Light Response Diagram*

7.4. Haptic Sensor Response Diagram

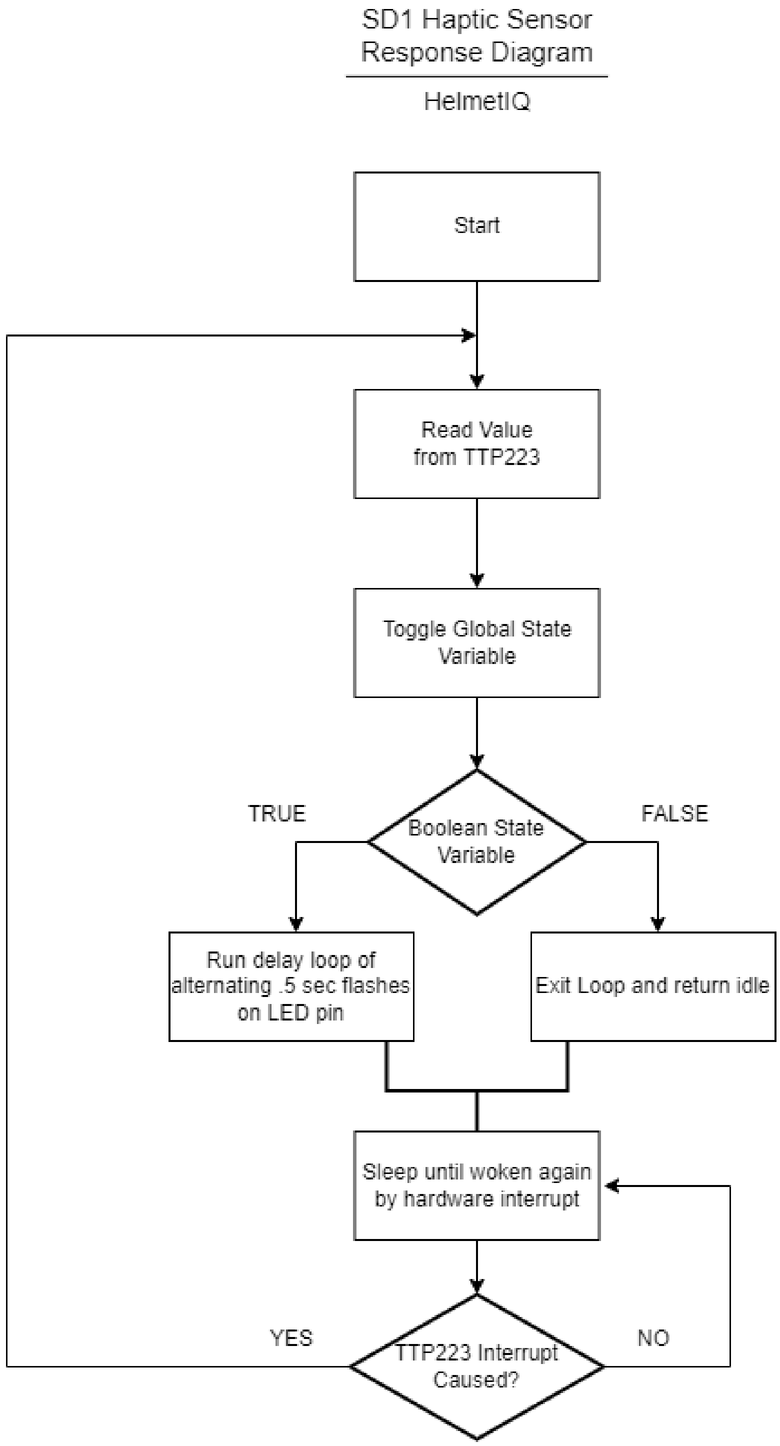

Figure 7.4 *– Haptic Sensor Response Diagram*

The haptic sensor response diagram for HelmetIQ illustrates the process flow for handling inputs from the TTP223 touch sensor. The process starts by reading the sensor value, which toggles a global state variable. Depending on the state of this variable, the system either runs a delay loop that causes the LED pin to flash alternately every 0.5 seconds or exits the loop and returns to an idle state. After executing the appropriate action, the system enters a sleep mode until it is woken by a hardware interrupt caused by the TTP223 sensor. If an interrupt is detected, the process starts again. This diagram captures the sensor's role in toggling LED signals and managing the system's power states, ensuring responsive and efficient operation.

7.5. Collision Detection System Response Diagram

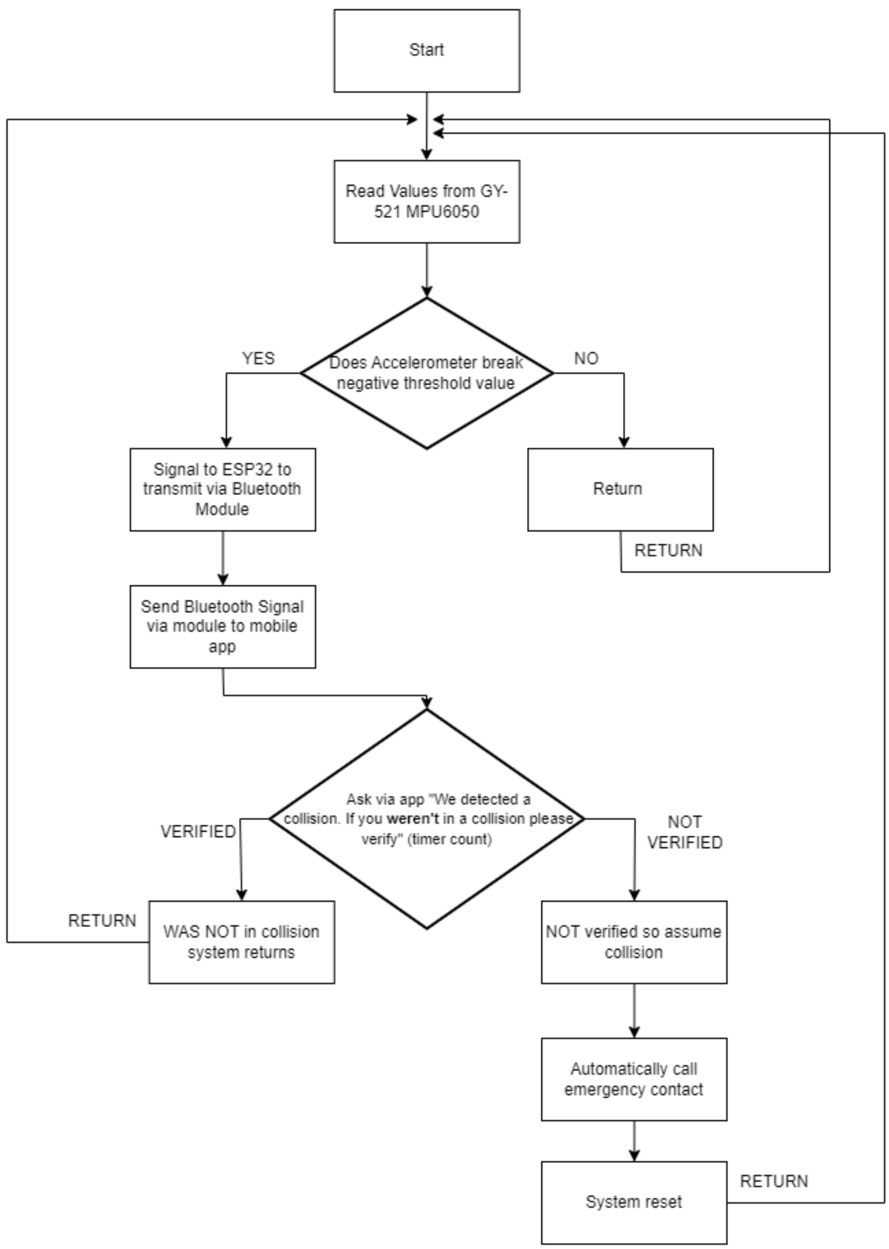

***Figure 7.5** – Collision Detection System Response Diagram*

The system starts by reading values from a ICM-42670 sensor. If the accelerometer detects a value exceeding a negative threshold, indicating a potential collision, it signals the ESP32 to transmit a collision flag via Bluetooth to a mobile app. The app then prompts the user to verify if a collision occurred. If verified as not a collision, the system returns to its initial state. If not verified within a set time, the system assumes a collision has occurred and automatically notifies an emergency contact before resetting. If no collision is detected initially, the system simply returns to the beginning of the loop. This automated process aims to quickly detect collisions and provide emergency assistance when needed, while also allowing for user verification to prevent false alarms.

7.6. User Interface Design

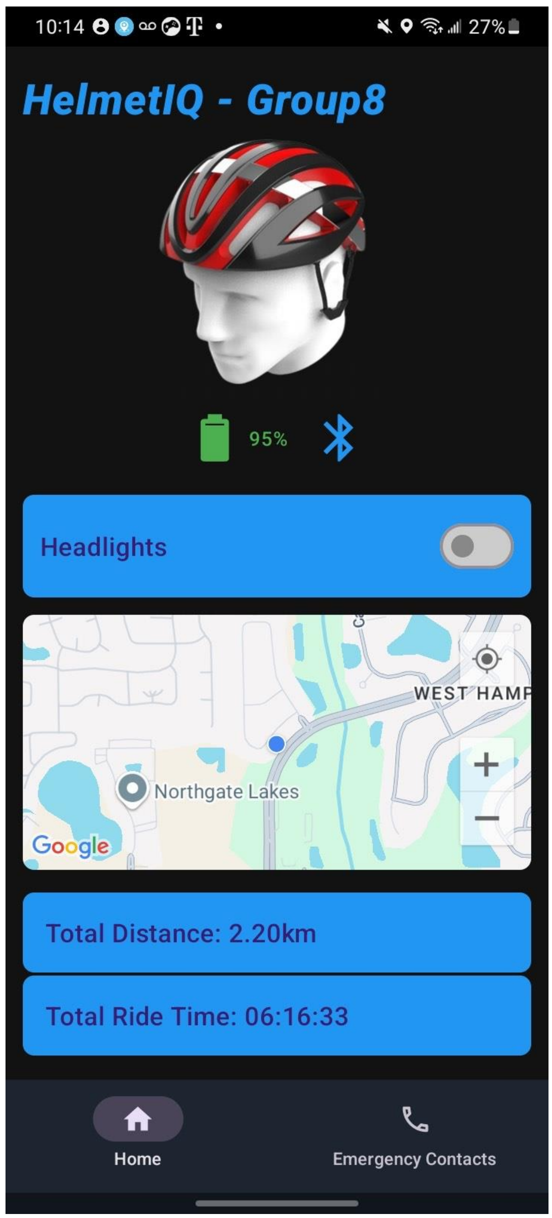

***Figure 7.6** – Mobile Home Page UI Design*

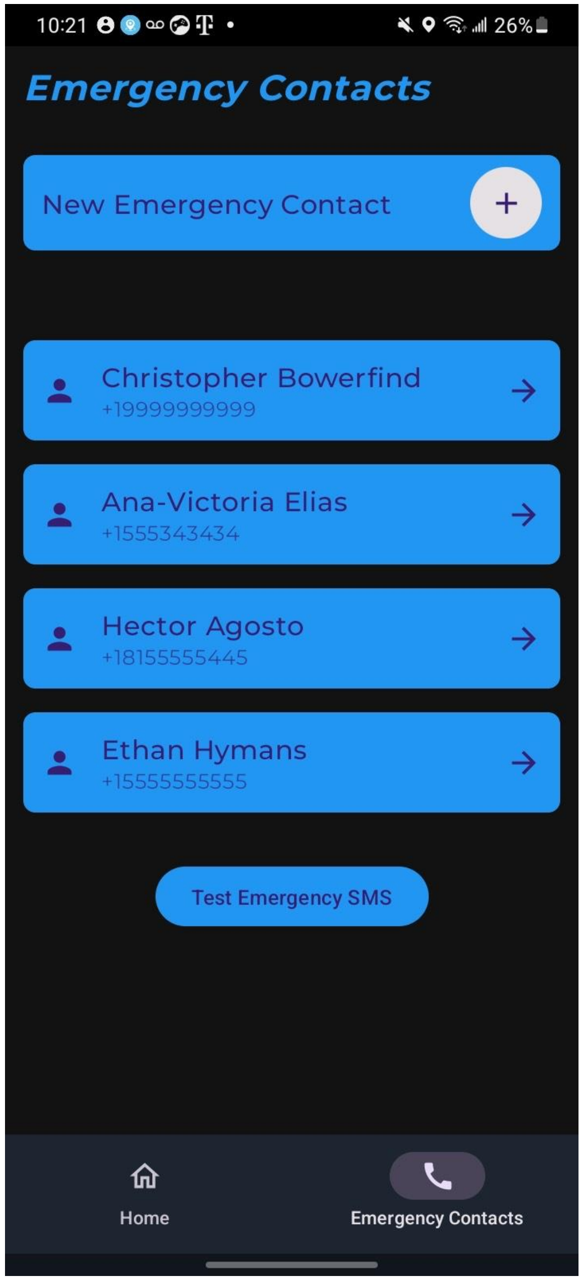

***Figure** 7.7 – Emergency Contact Page UI Design*

For the user interface design, the layout is relatively simple and contains the necessary features for full functionality of HelmetIQ. On the Home page, the user is able to see battery and Bluetooth status, along with ride metrics such as total distance traveled and total ride time. The user is also able to activate/deactivate the front headlights of the bike helmet if they are not needed, or if the user is just taking a break and wants to conserve battery life.

On the SOS Contacts page, the user has the option to add a New Emergency Contact and activate/deactivate the Potential Collision SMS message sent when the helmet detects that the user was involved in a potential accident. Below these options are the user's listed emergency contacts. If the user selects an individual contact, they can edit the name and phone number, or delete the contact entirely.

7.7. Data Transfer

For data transfer between the ESP32 and the mobile app, Bluetooth is used to send flags from the ESP32 to the mobile app. The flags that are sent include battery status, accelerometer/gyroscope interrupt flag, and LED headlight on/off flag. The battery status provides real-time updates to the current charge level, allowing the user to monitor and receive alerts when recharging is necessary. This data is essential for maintaining the helmet's operational integrity and ensuring HelmetIQ is ready for use.

Additionally, the accelerometer/gyroscope flag is transmitted to indicate significant movements or impacts, enabling the app to assess the user's safety dynamically. This triggers emergency notifications/SMS if abnormal activity is detected. The LED headlight on/off flag ensures that the app can remotely control and monitor the helmet's lighting status, if the user wanted to briefly turn off the helmet headlight to talk to someone face-to-face for example.

To send this data to the mobile app via Bluetooth, the data must be structured in a way that the receiving application can easily parse and interpret. This could be done in two different ways. The first being JSON Formatting, the second being delimiter-separated formatting.

```
{
    "battery": 85,
    "accel_gyro_interrupt": true,
    "led_status": "ON"
}
```

Figure 7.8 *– Example of data sent with JSON formatting*

JSON Formatting is structured, and relatively easy to parse on the receiving end. Alternatively, we could sent data through delimiter-separated strings, where each piece of data sent is separated by a specific character such as a comma or semicolon.

```
battery=85;accel_gyro_interrupt=true;led_status=ON
```

Figure 7.9 – *Example of data sent with delimiter-separated formatting.*

Since almost all of the processing is done on HelmetIQ itself, the choice to go with delimiter-separated formatting is a no-brainer. As we are only sending just a few status lines via Bluetooth, keeping things simple and using delimiter-separated formatting does the job perfectly fine.

8. System Fabrication/Prototype Construction

8.1. Prototype Description

The smart helmet prototype PCB was designed using EAGLE CAD software, including an ESP32 microcontroller, voltage regulators, haptic drivers, LED drivers, and multiple sensors. The PCB design ensures efficient power distribution, signal integrity, and ease of assembly. Components such as LEDs, ERMs, and touch sensors are connected via wires to through-hole connectors on the PCB.

8.2. Light Sensor Response

The light sensor (VEML7700-TR) is responsible for measuring ambient light levels to adjust the brightness of the helmet's LEDs. The sensor is connected to the ESP32 via the I2C bus, with appropriate pull-up resistors to ensure reliable communication. The sensor's placement on the PCB minimizes interference from other components. During testing, the light sensor demonstrated accurate response to varying light conditions, effectively adjusting the LED brightness in real time. Decoupling capacitors placed near the sensor's power pins ensured stable operation.

8.3. Haptic Sensor Response

The haptic feedback system uses DRV2605L drivers to control ERM motors, providing tactile feedback to the user. The haptic drivers are connected to the ESP32 through a TCA9548A I2C multiplexer, allowing independent control of each motor. The drivers are placed near the edge of the PCB for easy connection to the external ERMs. Decoupling capacitors and resistors ensure noise-free operation. The haptic feedback system was tested by simulating turn signal activations, with the ERMs providing consistent and responsive feedback.

8.4. Collision Detection Response

Collision detection is managed by the ICM-42670 gyroscope and accelerometer. This sensor is connected to the ESP32 via I2C, with decoupling capacitors placed close to the power pins to filter out noise. The sensor is mounted centrally on the PCB to accurately capture motion data from all directions. During testing, the collision detection system accurately detected abrupt movements and impacts, triggering appropriate responses such as activating brake lights or alerting the user through haptic feedback.

8.5. Automatic Brake Light Response

The automatic brake light system utilizes the ICM-42670 sensor to detect deceleration. The brake lights are controlled by the AP3019AKTR LED driver, which ensures bright and responsive illumination. The LED driver is placed near the edge of the PCB for easy connection to the external brake LEDs. Decoupling capacitors are used to stabilize the power supply to the driver. Testing showed that the brake lights responded quickly and reliably to deceleration events, enhancing the safety features of the helmet.

8.6. System Fabrication

8.6.1. PCB Layout

The PCB layout was meticulously designed to optimize signal integrity, power distribution, and component placement. The ESP32 microcontroller was placed centrally to facilitate easy access to all peripheral connections. A separate PCB for the Voltage regulators (e.g., TSP62120) was made to ensure efficient power distribution. Decoupling capacitors were placed close to the power pins of the microcontroller and other ICs to filter out noise and stabilize the voltage. Ground planes and polygon stitching were implemented for efficient layer connection and noise reduction.

Component placement was done with consideration of signal path lengths, heat dissipation, and ease of soldering. High-frequency components were placed to minimize noise interference. Power and ground planes were used to ensure a stable power supply. Special routing considerations were made for high-frequency signals to prevent interference. The PCB design included dedicated connector pads for external components, ensuring easy assembly and reliable connections.

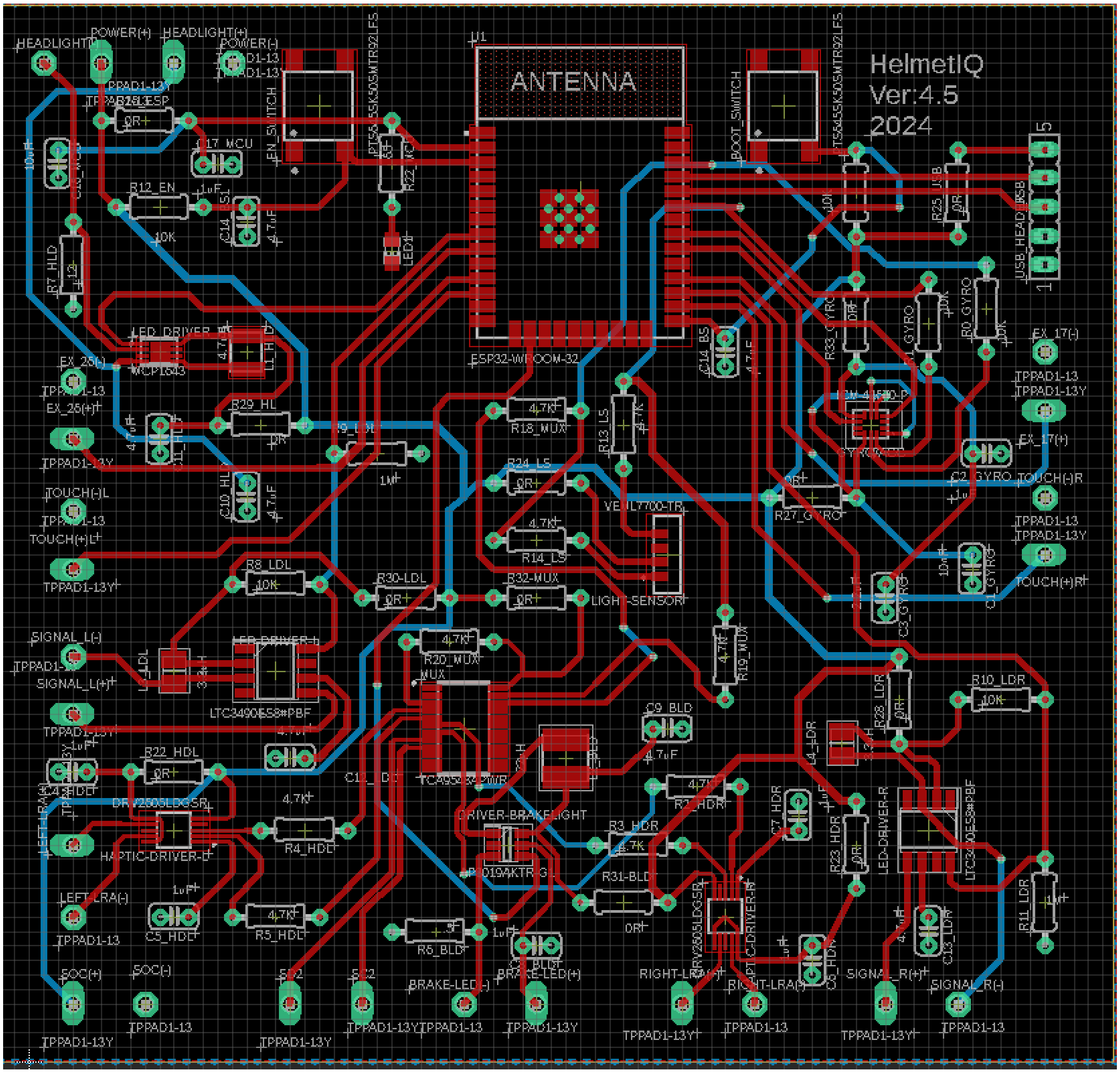

***Figure 8.1** – Initial PCB Layout*

8.6.2. Component Selection

- Microcontroller (ESP32): Central placement for optimal access to peripherals.
- Voltage Regulators (TSP62120): On separate PCB near the power input for efficient power distribution.
- Haptic Drivers (DRV2605L): Located near the edge of the PCB for easy connection to external ERM motors.
- LED Drivers: Positioned for easy connection to the helmet’s LED indicators.
- Sensors (VEML7700-TR and ICM-42670): Placed to minimize interference and ensure accurate readings.

Decoupling capacitors were placed close to the power pins of all major ICs to ensure stable voltage levels and minimize noise. This is critical for maintaining the integrity of signals and preventing malfunction due to power supply fluctuations.

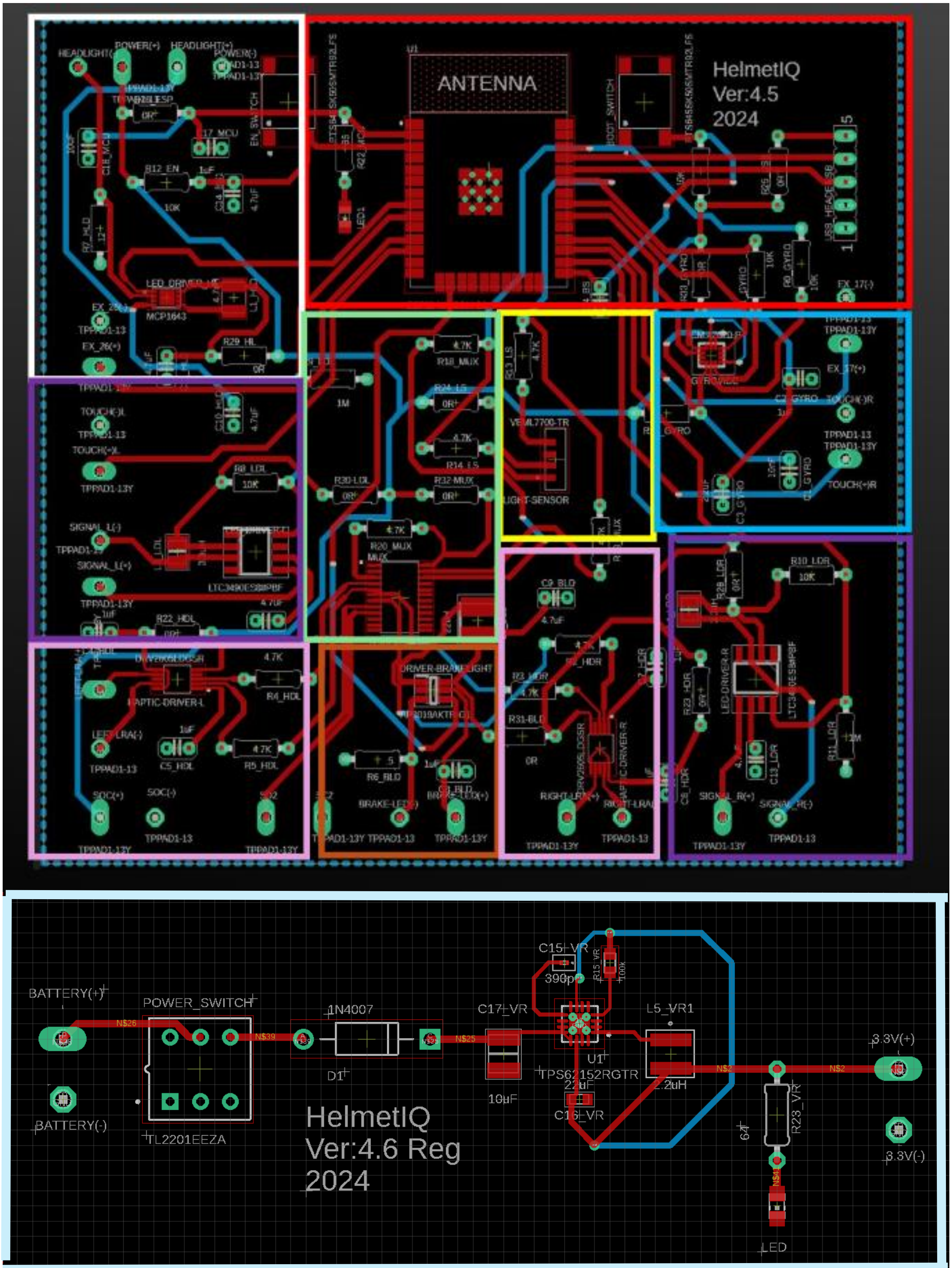

Figure 8.2 *-PCB Sub-systems layout*

The image above highlights several key subsystems on the PCB.

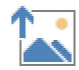

Figure 8.2.1 *- MCU Esp32*

- **Red Box -** The images show a close-up of the ESP32-Wroom-32, serving as the system's central processing unit (CPU) and primary microcontroller (MCU), providing the overall functionality and communication within the Helmet IQ.

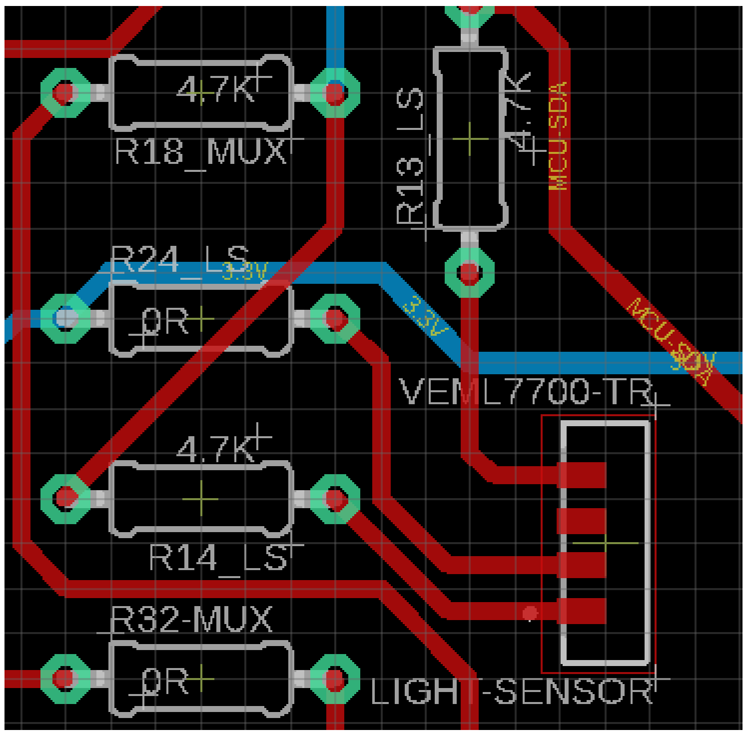

Figure 8.2.2 *-Light Sensor*

- **Yellow Box** -This area houses the VEML7700 light sensor and one of the LED drivers, responsible for controlling and for managing illumination.

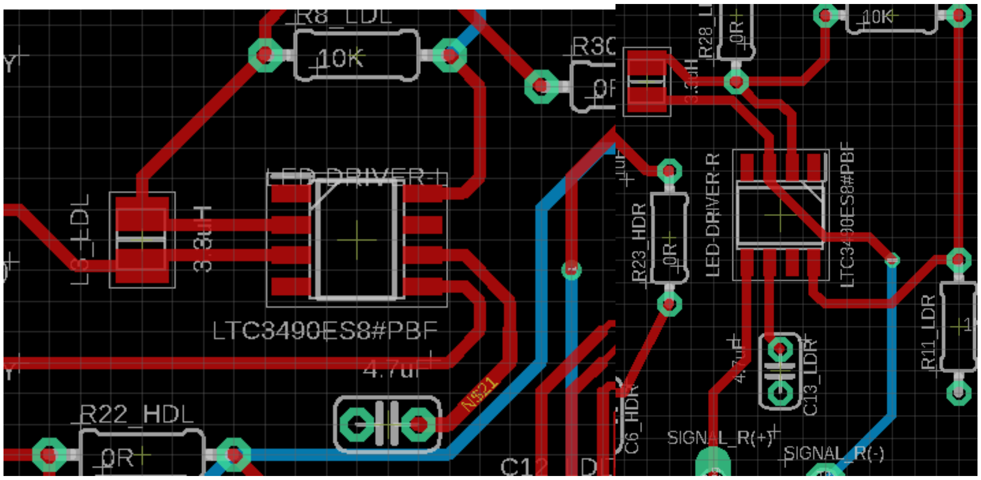

Figure 8.2.3 *- Left and Right LED Driver*

- **Magenta Box -** Left and Right LED drivers proving a stable current, voltage and the ability to use two different GPIO pins.

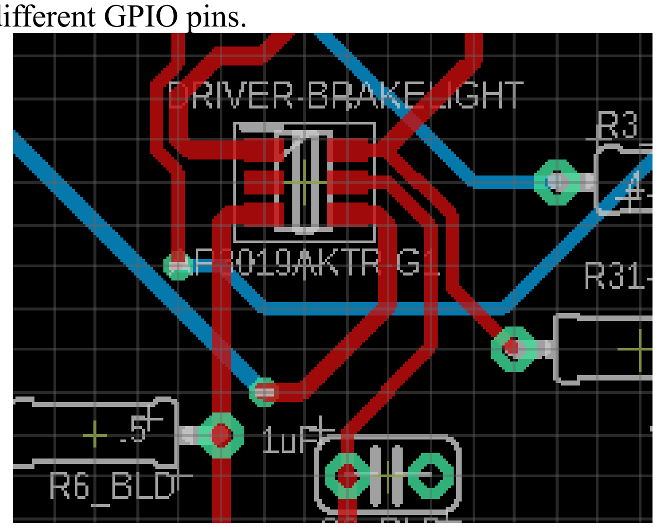

Figure 8.2.4*- Brake LED Driver*

- **Orange Box** – Brake light LED driver ensuring a stable connection providing the total voltage for two brake lights in series.

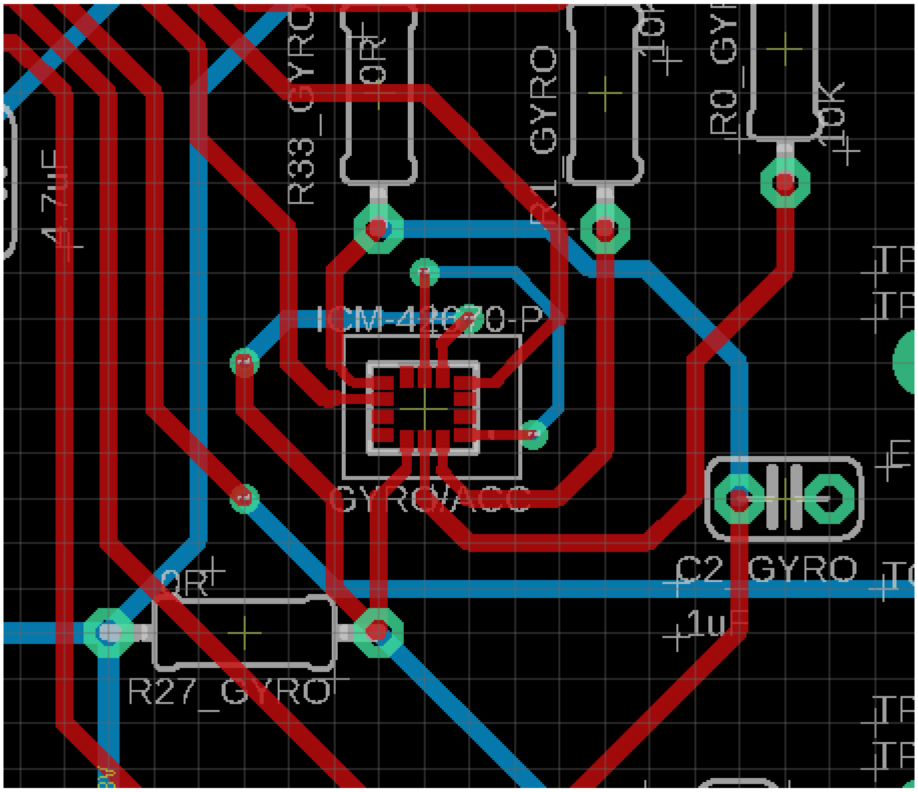

Figure 8.2.5 - Gyroscope/Accelerometer

- **Blue Box** -The ICM-42670, a 6-axis inertial measurement unit (IMU) consisting of a gyroscope and accelerometer, is located here.

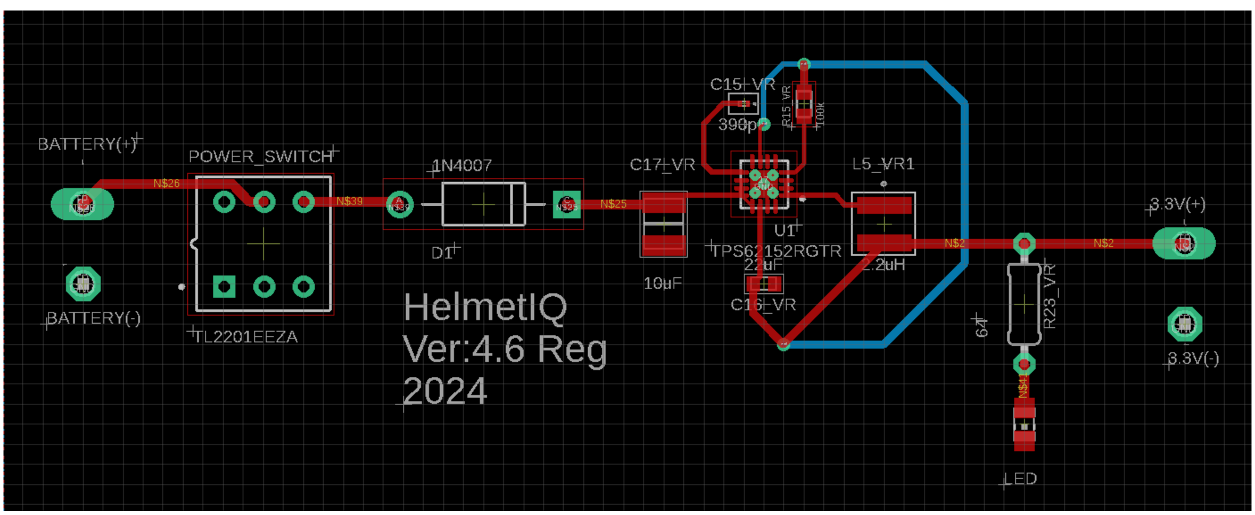

***Figure 8.2.6** -Power System*

- **Light Blue Box** -This is the power system that ensures a stable and reliable power supply to every component on the PCB, optimizing performance and efficiency.

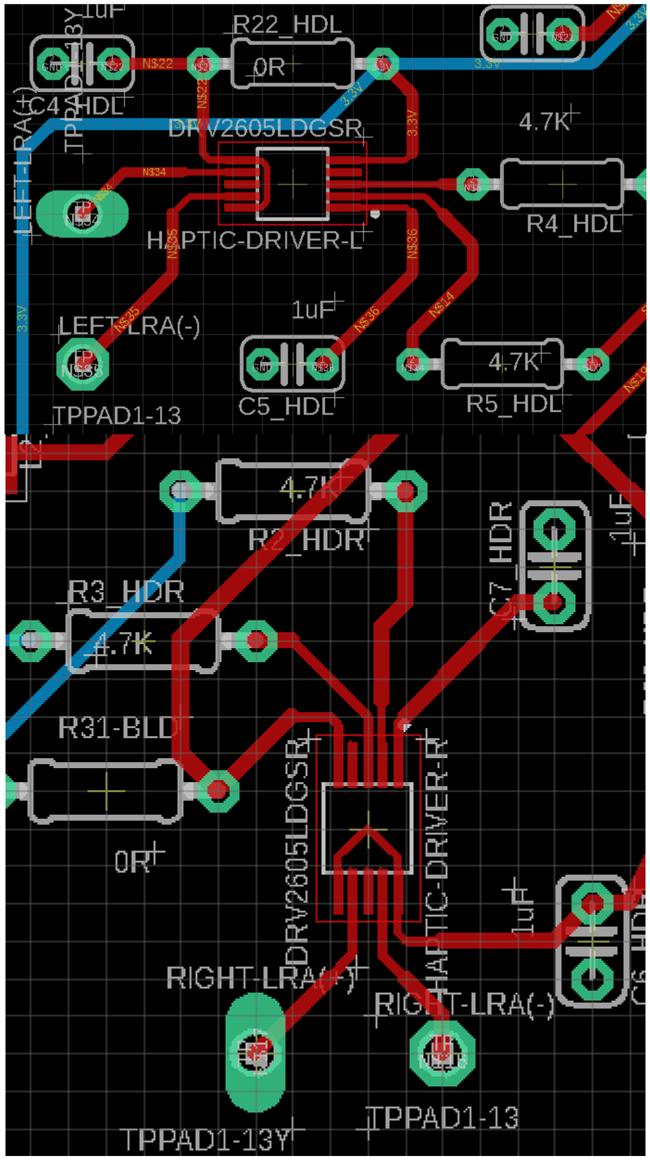

Figure 8.2.7 *- Haptic feedback Driver*

- **Pink Box**- Most of the haptic feedback components, including the MUX (multiplexer) and the two haptic feedback drivers, are clustered in this region.

The PCBs (printed circuit boards) for Helmet IQ were designed using Autodesk Eagle. Due to specific placement requirements, certain components were prioritized during layout. These included the battery port, the USB port for the processor, and other connectors that needed to be accessible along the PCB edges for external connections.

For LEDs, haptic feedback motors, touch sensors, copper pads were strategically placed for soldering. Once these location-sensitive components were positioned, subsystems were clustered together to minimize trace lengths and optimize signal integrity.

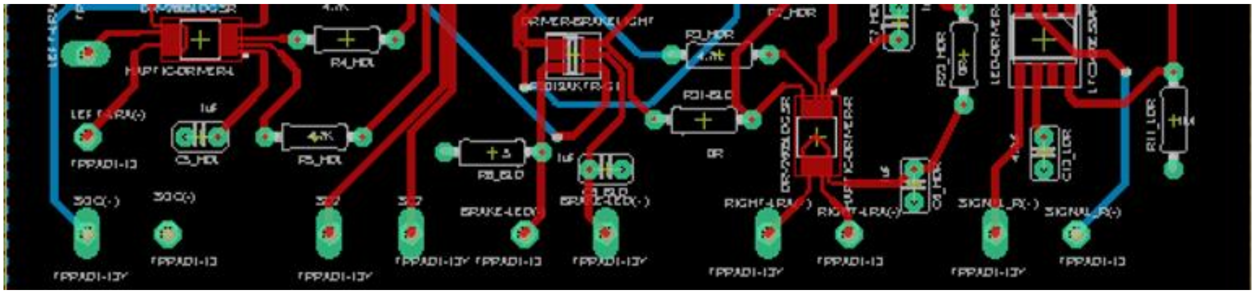

Figure 8.2.8 *- Position Critical Components*

To ensure robust grounding and noise reduction, two ground layers were incorporated—one on the top and one on the bottom—creating a unified ground signal. This approach is particularly beneficial in a helmet environment where electromagnetic interference is a concern.

At this stage in the design, components were strategically placed, prioritizing subsystem clustering while adhering to any predetermined placement constraints. This iterative process involved numerous adjustments, continuous fine-tuning, and even occasional complete restarts to achieve an optimal PCB layout for Helmet IQ. In some cases, it was necessary to revisit the schematic to correct errors or refine the design for improved board performance.

The final phase of PCB design involved adding drill holes, ensuring design quality, and refining the silkscreen layer. This included adjusting component labels and adding any missing labels for clarity and ease of assembly.

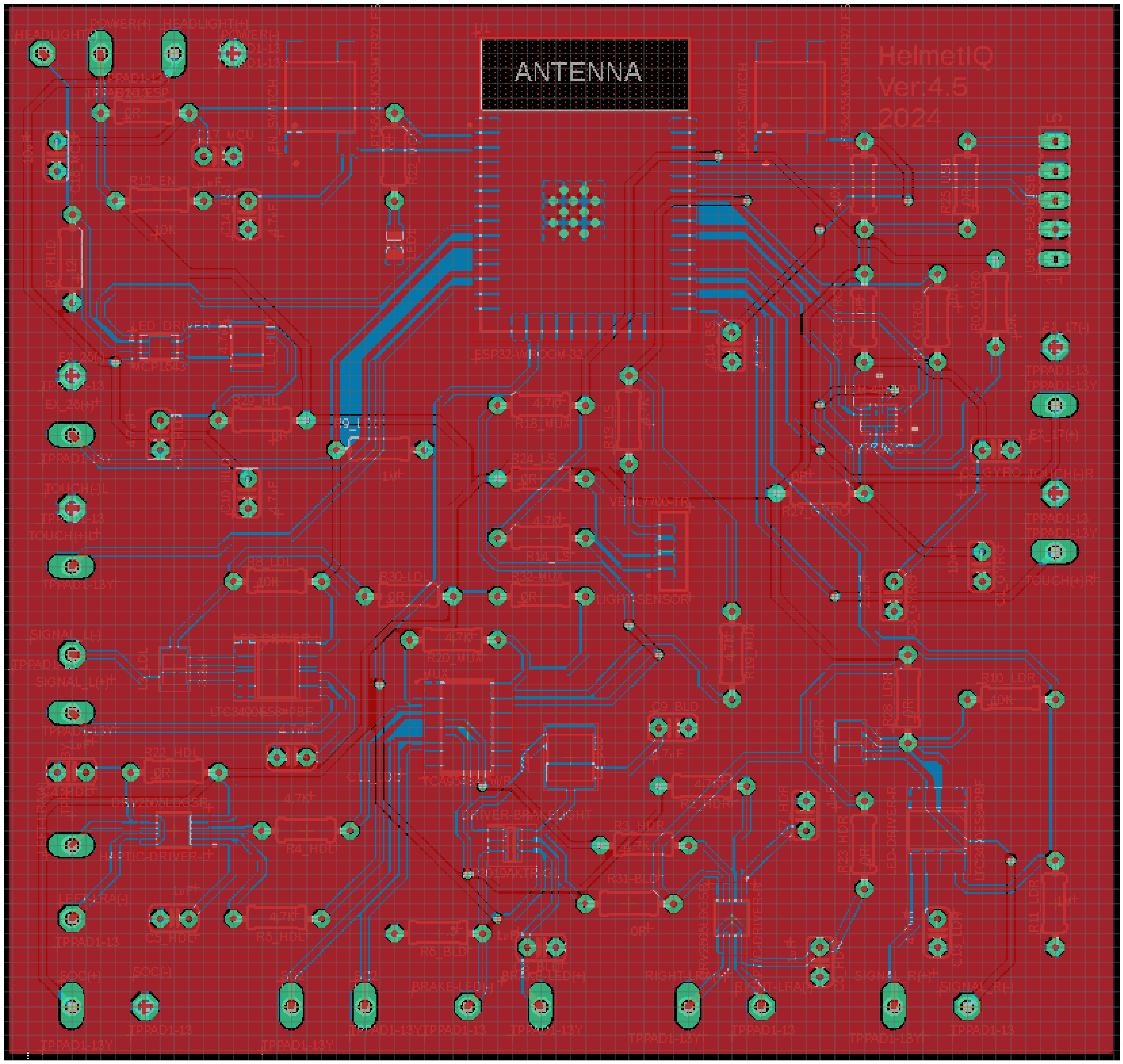

Figure 8.3.1 - *Final HelmetIQ PCB Ver.4.5*

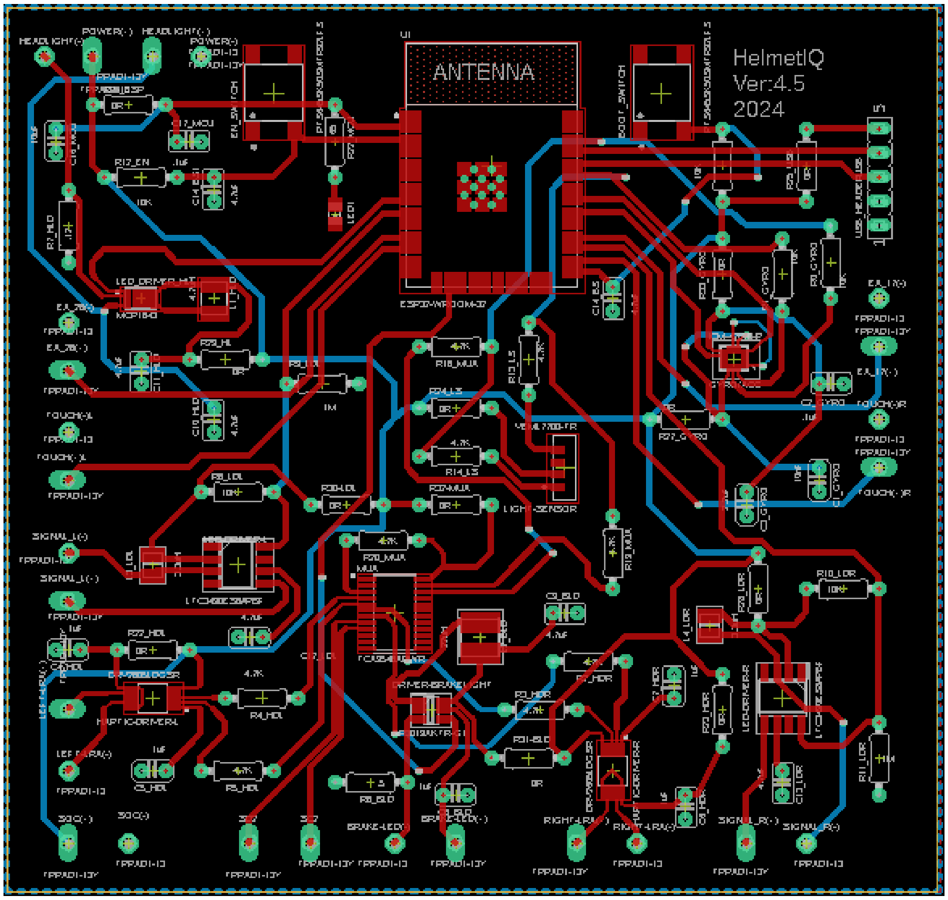

***Figure 8.3.2* -** Final HelmetIQ PCB Ver.4.5

8.6.3. Structural Illustration

The smart helmet's design is centered around strategic placement of all its components to optimize performance, user comfort, and safety. For the concept illustration, we utilized a CAD model of a bike helmet from the GrabCAD library to visualize the placement and integration of the smart helmet components. The model, available at GrabCAD, provided an accurate representation of our concept design process

The illustrations below offer a preliminary overview of the component placements:

Front section:

- Headlight: Positioned to provide maximum visibility for the user in low-light conditions.

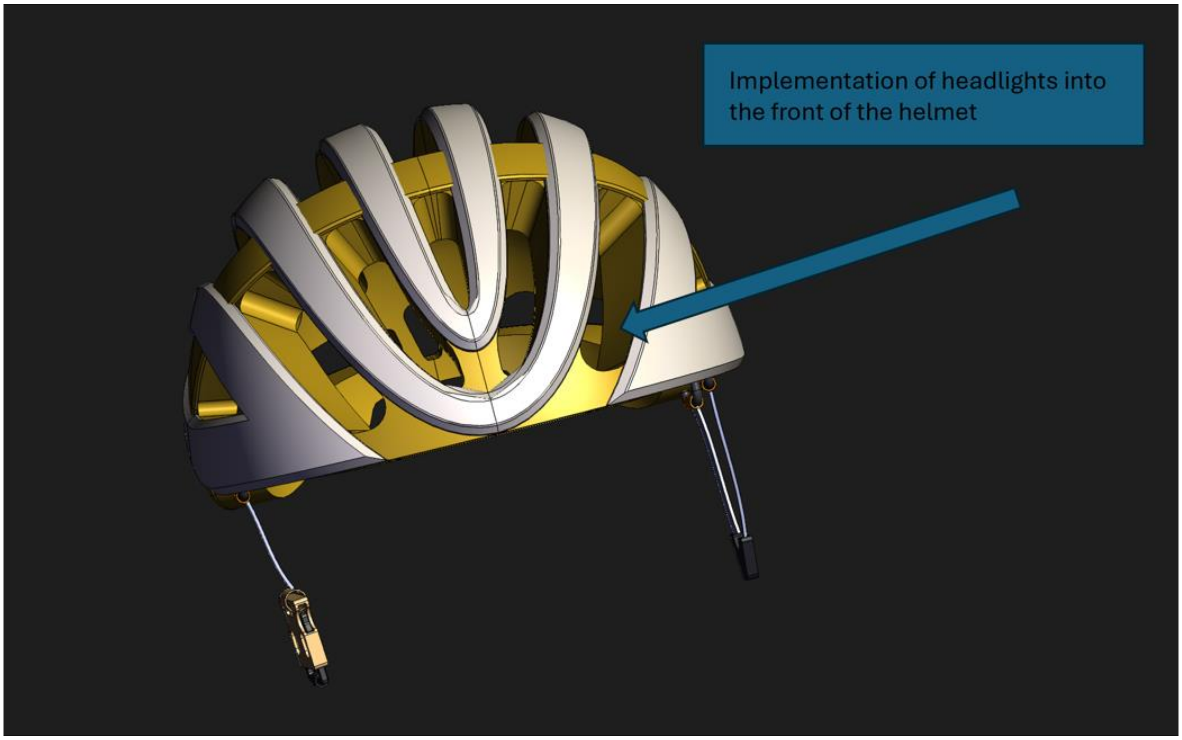

Figure 8.4 *-1 Concept Illustration of Implementation of Headlights*

Back section:

- Brake Lights and Turn Signals: Strategically placed for maximum visibility to following vehicles, enhancing safety during deceleration and signaling turns.

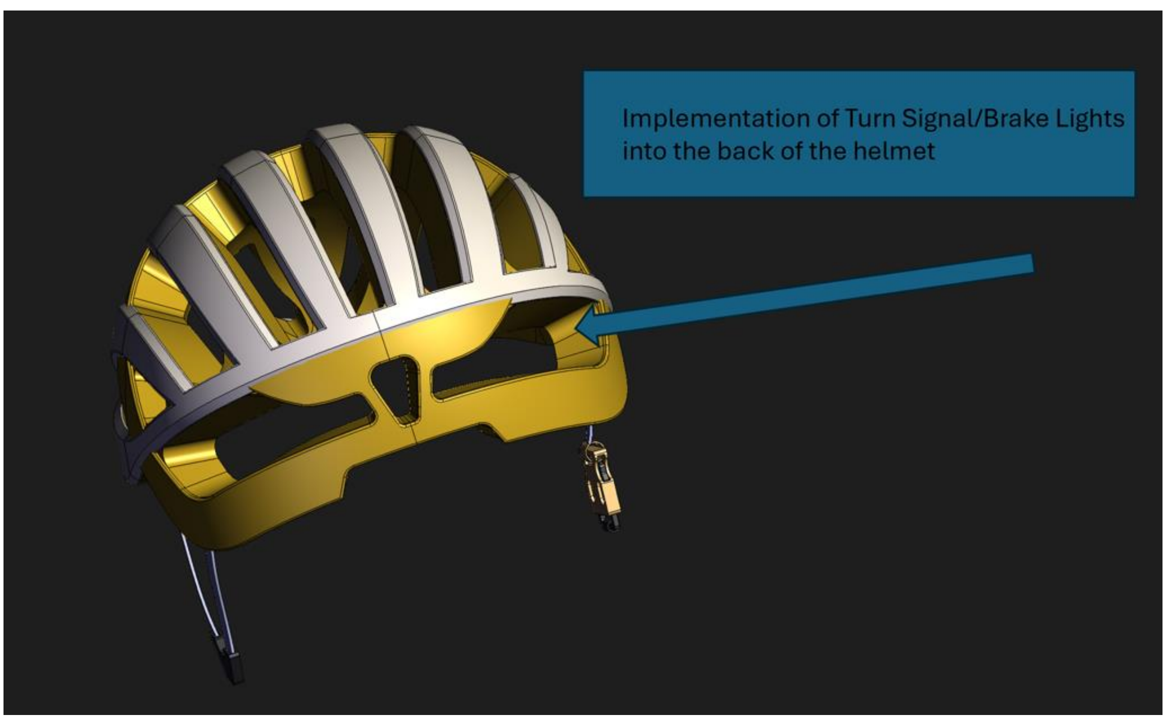

***Figure 8.5**2 -Concept Illustration of Implementation of Brake Lights and Turn Signals*

Side Section:

- ERM Motors (Eccentric Rotating Mass): Positioned to provide effective haptic feedback to the user.
- Capacitive Touch Sensor: The sensors are positioned such that they can be easily reached by the user's fingers without causing discomfort or distraction.

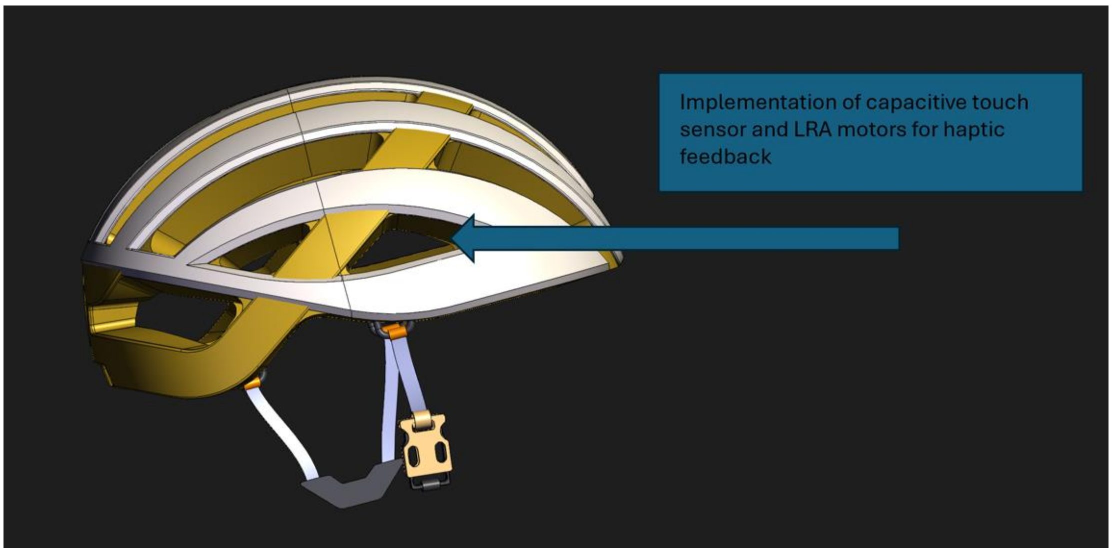

***Figure** 38.6 - Concept Illustration of Implementation of Haptic Feedback and Capacitive Touch Sensors*

Top section:

- Primary PCB: Centrally located to minimize wiring complexity and ensure balanced weight distribution.
- ICM-42670 Sensor: Placed for best detection of helmet movement and collision impacts.

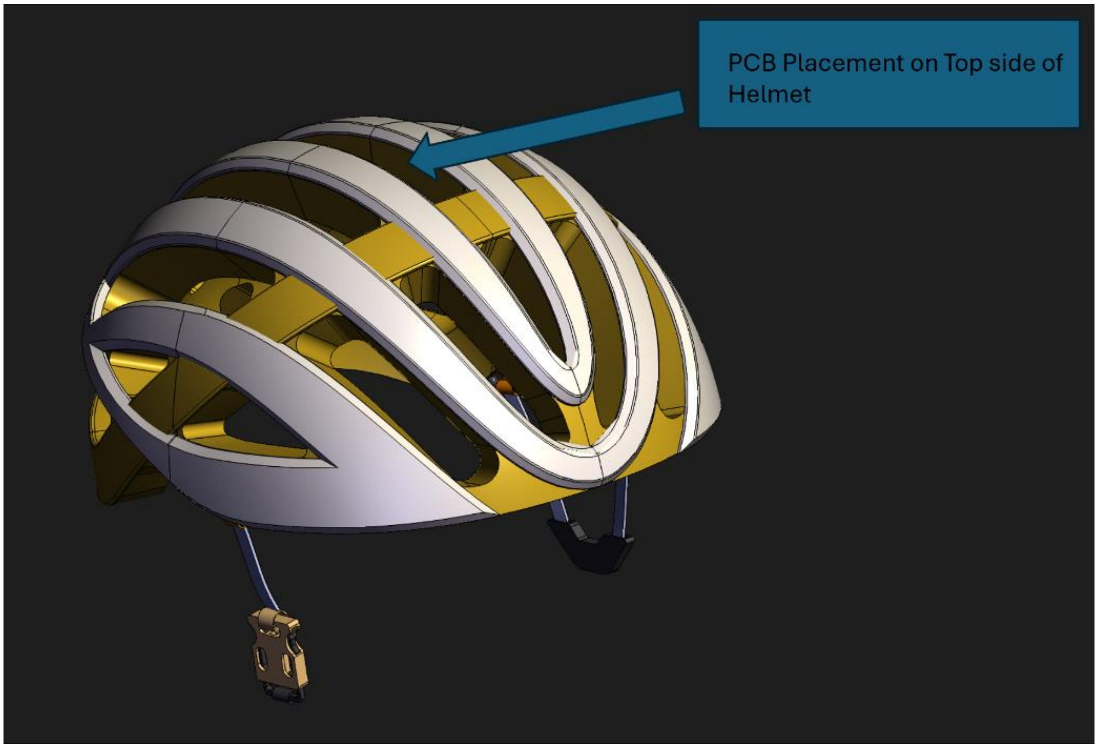

Figure 8.*74 -Concept Illustration of PCB Placement*

Concept Illustration of Implementation of Headlights:

Note: The CAD model used in the illustrations is sourced from GrabCAD and serves only for conceptual visualization. The original model is available at GrabCAD.

8.7. Prototype Construction

The first step in constructing HelmetIQ is the ordering of the Printed Circuit Board (PCB), which serves as the backbone of our entire system. This PCB, designed according to our determined specifications, is a sophisticated platform that integrates an array of sensors and essential circuitry, all of which are designed to be mounted in a stationary configuration within the helmet.

Upon receiving the manufactured PCB, we enter a phase of component-level testing. This stage is fundamental and cannot be rushed, as the reliability of the entire system hinges on the proper functioning of each individual element. Our testing protocol is comprehensive and systematic, involving the use of specialized equipment such as oscilloscopes, multimeters, and testing scripts. We examined every sensor, microcontroller, power management unit, and auxiliary component mounted on the PCB.

For analog sensors, such as accelerometers or gyroscopes, we verified their sensitivity, accuracy, and signal-to-noise ratio across their full operating range. Digital components, including microcontrollers and communication interfaces, underwent rigorous functional testing to ensure they meet timing specifications and can handle the expected data throughput. Power management circuits were stress-tested to confirm they can provide stable voltages under various load conditions, which is crucial for the reliable operation of sensitive components.

This individual component testing phase is not just about verifying functionality. It is also an opportunity to calibrate sensors, adjust parameters, and optimize performance. We created test results for each component, documenting performance metrics, any deviations from expected behavior, and steps taken to address any issues encountered. This documentation was important for future iterations and potential troubleshooting.

Only after we have conclusively verified the proper operation of each discrete component can we progress to the next critical stage which is the integration and assembly of all sensors and peripherals into a cohesive major system. This integration process is where the true complexity of our design becomes apparent, as we must ensure that all components not only function individually but also work together as part of a larger ecosystem.

During this integration phase, we were particularly detailed about potential issues that were identified during the PCB design stage. One significant concern is the possibility of electromagnetic interference or 'noise' affecting some of the traces due to their close proximity on the board. This is a common challenge in dense PCB layouts, especially those involving both analog and digital circuits.

The integration process also involved writing and testing the software that coordinates the functions of various components. This software development phase is crucial, as it's the way that everything in the system is able to communicate and complete their designed task. We used standard best practices in embedded systems programming, including the use of a real-time operating system library (RTOS), to ensure deterministic behavior and efficient resource management.

As we assemble the system, we conducted a series of integration tests. These tests started with basic functionality checks and progressively move to more complex scenarios that simulate real-world usage. We used data logging and analysis tools to capture and interpret the system's behavior under various conditions. This might involve creating simulation tests that can simulate motion, impacts, and other relevant environmental factors.

Once we were satisfied with the performance of the integrated system in the lab environment, we moved on to the next phase of prepping the helmet base for modification. This process begins with the acquisition of a suitable commuter-style bike helmet. To manage costs effectively during this prototyping phase, we'll explore various sourcing options. While online retailers offer a wide selection, we're particularly interested in sourcing from used bike shops or surplus stores. This approach helps control costs.

It's crucial to emphasize that this initial helmet served as a prototype and not the final product. We knew that this first attempt at modifying the helmet base would present a list of unforeseen challenges. These might include issues related to weight distribution, heat dissipation, weatherproofing, and ergonomics. By acknowledging upfront that we likely need multiple iterations. We are setting realistic expectations and creating a mindset geared towards learning and improvement.

The process of modifying the helmet base involved several steps. First, we conducted a thorough assessment of the helmet's structure, identifying potential mounting points for our PCB and sensors. We'll need to consider how to integrate our components without compromising the helmet's primary safety function. This involved carving out recesses in the helmet's foam liner and designing protective enclosures for our electronics.

We also needed to address power management considerations. This involved integrating a rechargeable battery pack and designing a convenient charging interface. We needed to carefully consider the placement of this power source to ensure it doesn't negatively impact the helmet's balance or comfort.

In addition to integrating all of the components in the system, we also 3D printed a top cover to add some weatherproofing and a customized design to create the look of a complete product. This was made from a black PLA filament and allows for light to move through the cover (via a cutout) and still allow functionality for our dynamic lighting system. This 3D print took multiple iterations because we had to adapt the print to match the cutouts in the helmet base.

Throughout these iterations, we documented our processes, challenges faced, and solutions implemented. This documentation was important for refining our approach in subsequent iterations and formed the core of our manufacturing process.

As we progressed through multiple iterations of the helmet design, we gained insights into which modification strategies work best. We learned about the most effective methods for securely mounting components, the optimal placement of sensors for accurate data collection, and the best approaches for managing heat dissipation and power consumption. Each iteration brought us closer to a design that seamlessly integrates our technology with the helmet's structure without compromising its primary safety function.

After we perfected this integration process through multiple prototypes, we began building the final version of HelmetIQ. For this final iteration, we selected a higher-quality, more realistic helmet base that meets or exceeds relevant safety standards. We then applied all the knowledge and techniques gained during our prototyping phase to integrate the system with the helmet as seamlessly as possible.

The final assembly process was a culmination of all our previous efforts. We used specialized tools and techniques to embed our PCB and sensors into the helmet structure. We payed particular attention to ensuring that all connections are secure and protected against vibration and impact.

Once the system is implemented in this final form factor, we conducted a comprehensive series of functionality tests. These tests were more rigorous and extensive than our previous evaluations, as they'll need to account for real-world usage scenarios. We tested the system's performance under various environmental conditions, including different temperatures, humidity levels, and light exposures. We also conducted durability tests to ensure that our modifications haven't compromised the helmet's safety measures.

No issues were found during these tests, we then focused on refining the aesthetic aspects of the system integration. This involves ensuring that all components are seamlessly blended into the helmet's design, creating a product that looks professional and appealing to potential users. We then 3D printed a custom cover that is black but allows the light sensor on the PCB to be exposed to the environments lighting conditions while protecting the PCB.

After the system met our visual and functional standards, we conducted a final round of independent testing and monitoring of each aspect of the system. This involves rigorously evaluating every feature and component to confirm they are operating correctly and meeting our performance criteria.

Finally, we'll conduct extensive testing of the collision detection system, which represents the culmination of multiple integrated aspects within the overall system. This critical safety feature requires thorough validation to ensure it functions properly and reliably under various conditions. We simulated different types of impacts and motions to verify that the system can accurately detect and respond to potential collision scenarios.

9. System Testing and Evaluation

9.1. Overview of Testing

The testing phase is a critical part in the development process, ensuring that all hardware and software components function correctly and reliably. This section outlines the methods and procedures used to test various components independently before they are integrated into the system. Each component was subjected to a series of tests designed to validate its electrical connections, operational functionality, and overall performance. The tests were conducted systematically, starting with basic circuit verification and progressing to more complex functional tests, including software integration and data communication. The results of these tests provide confidence in each parts ability to function correctly and reliably when they are integrated into our larger system.

9.2. Hardware Testing

This section details the specific hardware components tested, the procedures used, and the results. The components include the ESP32 development board, VEML7700 light sensor, PATIKIL 3W LED, ICM-42670 3-Axis Accelerometer Gyroscope, and TTP223 haptic sensors. Each subsection provides a description of the setup, connection details, and testing. The results show that each part is able to function independently and should be able to be integrated into our larger system. These tests are crucial for identifying any potential issues early in the development process

9.2.1. PCB Power Supply Module

To ensure the reliability and performance of the power supply module before physical prototyping, we conducted extensive simulations using Multisim. The circuit components, including the TPS62162 buck converter, were accurately modeled in Multisim. Additional components, such as capacitors, resistors, and the DC input jack, were also included in the simulation to closely replicate the physical circuit. The output voltage was monitored to ensure it remained stable at 3.3V, meeting the requirements of the ESP32 microcontroller and other connected components. The input voltage was varied to simulate different battery conditions, confirming the buck converter's efficiency and stability across a range of operating scenarios.

9.2.2. ESP32

To test the ESP32 development board we connected it to power through its micro-USB port. The onboard power LED illuminated, confirming that the device was receiving power. Subsequently, the voltages on the GPIO pins were checked using a digital multimeter to ensure proper functionality. The reset button was tested by pressing it and observing the board's response, which verified that the reset mechanism was operational. Additionally, the ESP32 was connected to a computer via USB, and the serial monitor was opened using the Arduino IDE. A simple sketch was uploaded to blink the onboard LED, confirming that the board could be programmed successfully.

9.2.3. VEML 7700 Light Sensor

The VEML7700 light sensor was tested by connecting it to the ESP32 using a breadboard. The 3.3V output from the ESP32 was connected to the Vin of the sensor, ground was connected to ground, and the SDA and SCL lines were connected to GPIO pins 21 and 22. Once powered, the sensor's power LED lit up, showing that it was receiving power. Software testing was conducted to ensure proper communication over the I2C bus, confirming the successful transfer of data through the SDA and SCL lines. The sensor was then placed under different lighting conditions to verify its responsiveness. Data readings were taken showing variations in light intensity, which confirmed that the sensor was accurately measuring ambient light levels. The sensor was able to display data relating to the total Lux, Ambient Light Value, and White Light Value.

9.2.4. PATIKIL 3W LED

To test the PATIKIL 3W LED we connected it to an LED driver that could handle its voltage and current requirements. The LED driver was powered on via breadboard and ESP32, and the LED was observed to light up, confirming it was receiving power. A multimeter was used to verify the voltage and current supplied to the LED matched its specifications. Additionally, the LED's brightness was observed to make sure that it would match the project specifications. The LED was subjected to a series of tests, including varying the input voltage to observe its response and measuring the temperature of the LED to ensure it operated within safe thermal limits. The LED was also tested for prolonged periods to check for stability and reliability over time.

9.2.5. ICM-42670 3-Axis Accelerometer Gyroscope

The ICM-42670 sensor was tested by connecting it to the ESP32. The VCC pin of the sensor was connected to the 3.3V output of the ESP32, the ground to ground, and the SDA and SCL lines to GPIO pins 21 and 22. Once powered on, the sensor's onboard LED indicated that it was functioning. Software was then used to read accelerometer and gyroscope data, verifying that the sensor was accurately capturing motion and orientation information. The sensor was subjected to various movements and orientations to test its accuracy and sensitivity. Data was collected and analyzed to ensure the readings matched expected values. Calibration procedures were also performed to minimize any offset or drift in the sensor's output.

9.2.6. TTP223 Haptic Sensors

To test the haptic sensors, we connected them to the ESP32. The power pins were connected to the 3.3V and ground outputs of the ESP32, while the signal pins were connected to appropriate GPIO pins. Once powered on, the sensors were tested by triggering haptic feedback through software commands. The strength and responsiveness of the haptic feedback were evaluated to ensure the sensors were operating correctly. Various types of haptic feedback, such as vibrations of different intensities and durations, were tested to confirm the sensors' versatility. The sensors were also tested in different environments and under various conditions to assess their durability and reliability.

9.2.7. TCA9548A Multiplexer

To test the TCA9548A I2C multiplexer we connected it to the ESP32 development board. The VCC pin of the multiplexer was connected to the 3.3V output of the ESP32, and the ground pin was connected to ground. The SDA and SCL lines of the TCA9548A were connected to GPIO 19 and GPIO 18 on the ESP32. The address selection pins (A0, A1, and A2) were configured by connecting them to either ground or VCC to set the desired I2C address. The EN (enable) pin was connected to VCC to activate the multiplexer.

Once powered on the TCA9548A's power LED illuminated, showing that it was receiving power. A sketch was uploaded to the ESP32 to scan for I2C devices on each of the multiplexer's channels. This involved enabling each channel sequentially and scanning for connected I2C devices.

Data communication was tested by reading sensor values from the connected I2C devices through the multiplexer. This ensured that the multiplexer correctly routed the I2C signals from the ESP32 to the selected channel. The responsiveness and reliability of the TCA9548A were confirmed by rapidly switching channels and verifying consistent data acquisition without loss or corruption.

The multiplexer's performance was further validated by running extended tests to monitor its stability and response under continuous operation. These tests confirmed that the TCA9548A maintained proper functionality and reliable communication across all channels over prolonged periods, ensuring it could handle multiple I2C devices seamlessly in the final application.

9.3. Software Testing

9.3.1. ESP32

To test the functionality of the ESP32 microcontroller, a series of software tests can be conducted using the Arduino IDE and its built-in Serial Monitor. The first step is to verify the successful upload of the firmware to the ESP32 by checking for any compilation or upload errors in the

Arduino IDE. Once the firmware is uploaded, the ESP32's basic functionality can be tested by writing a simple sketch that performs basic tasks such as digital I/O operations, analog readings, and serial communication. The Serial Monitor can be used to display the output of these tests, confirming that the ESP32 is functioning as expected. Additionally, more advanced tests can be performed to verify the ESP32's Bluetooth capabilities by establishing connections and exchanging data with other devices.

Starting with a simple software test of the ESP32, the string "Helmet IQ Coming Soon!" was output to the serial monitor of the Arduino IDE. More importantly however is the confirmation that we are able to successfully able to upload our code to the ESP32 without any built errors or unexpected results. This gives the green light to move into more advanced tests, leading into individual component testing.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1448
load:0x40078000,len:14844
ho 0 tail 12 room 4
load:0x40080400,len:4
load:0x40080404,len:3356
entry 0x4008059c
Helmet IQ Coming Soon!
```

Figure 9.1 *– Simple Serial Monitor Test of ESP32*

To test the Bluetooth functionality of the ESP32, a Bluetooth connection was first established with another device. Next, Tera Term was utilized to establish a new serial connection – Standard Serial over Bluetooth Link. Matching the baud rate set within the Arduino IDE – 115200, it is now possible to send testing messages through Tera Term to the ESP32 over Bluetooth.

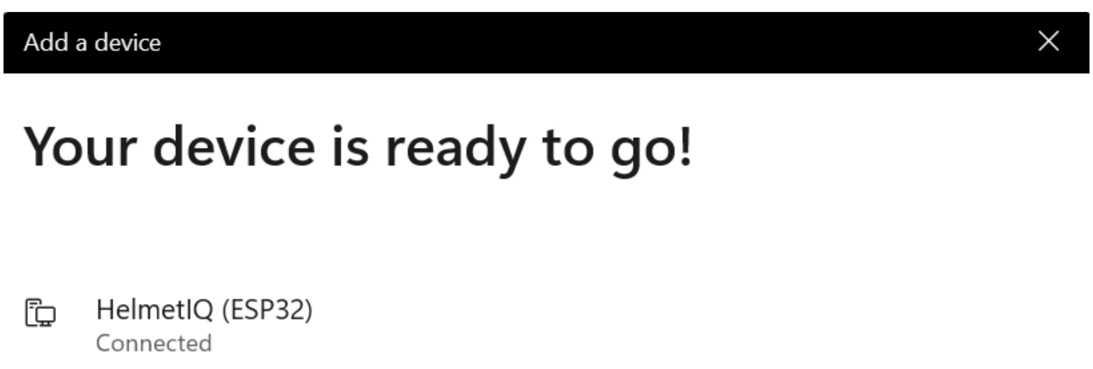

Figure 9.2 *– Establishing Bluetooth Connectivity with the ESP32 in Windows 11*

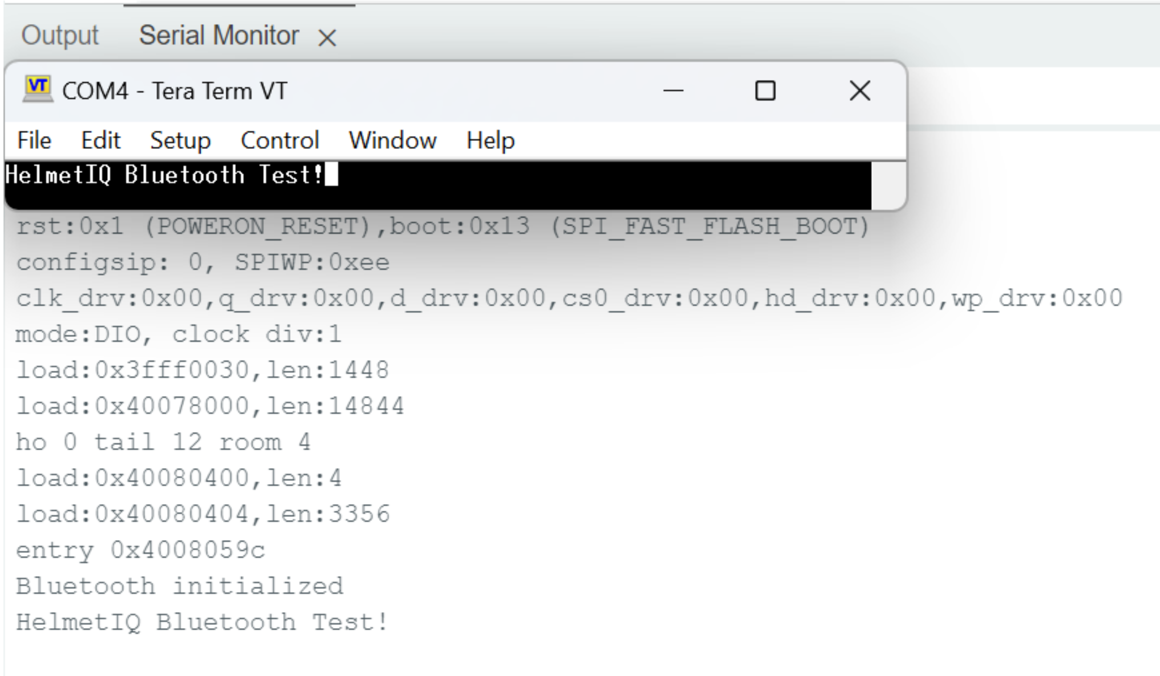

Figure 9.3 *– Sending a Test Message from Tera Term to The ESP32*

To complete the initial software testing of essential ESP32 functions, the last test is to establish a Bluetooth connection and send data from the ESP32 to the external Bluetooth enabled device. The purpose of this test is to simulate sending data from the ESP32 to the mobile app, as the app receives a stream of data from the different sensors. Following the previous steps to establish Bluetooth connectivity and set up Tera Term, the ESP32 was configured to transmit the accelerometer data via Bluetooth every 5 seconds.

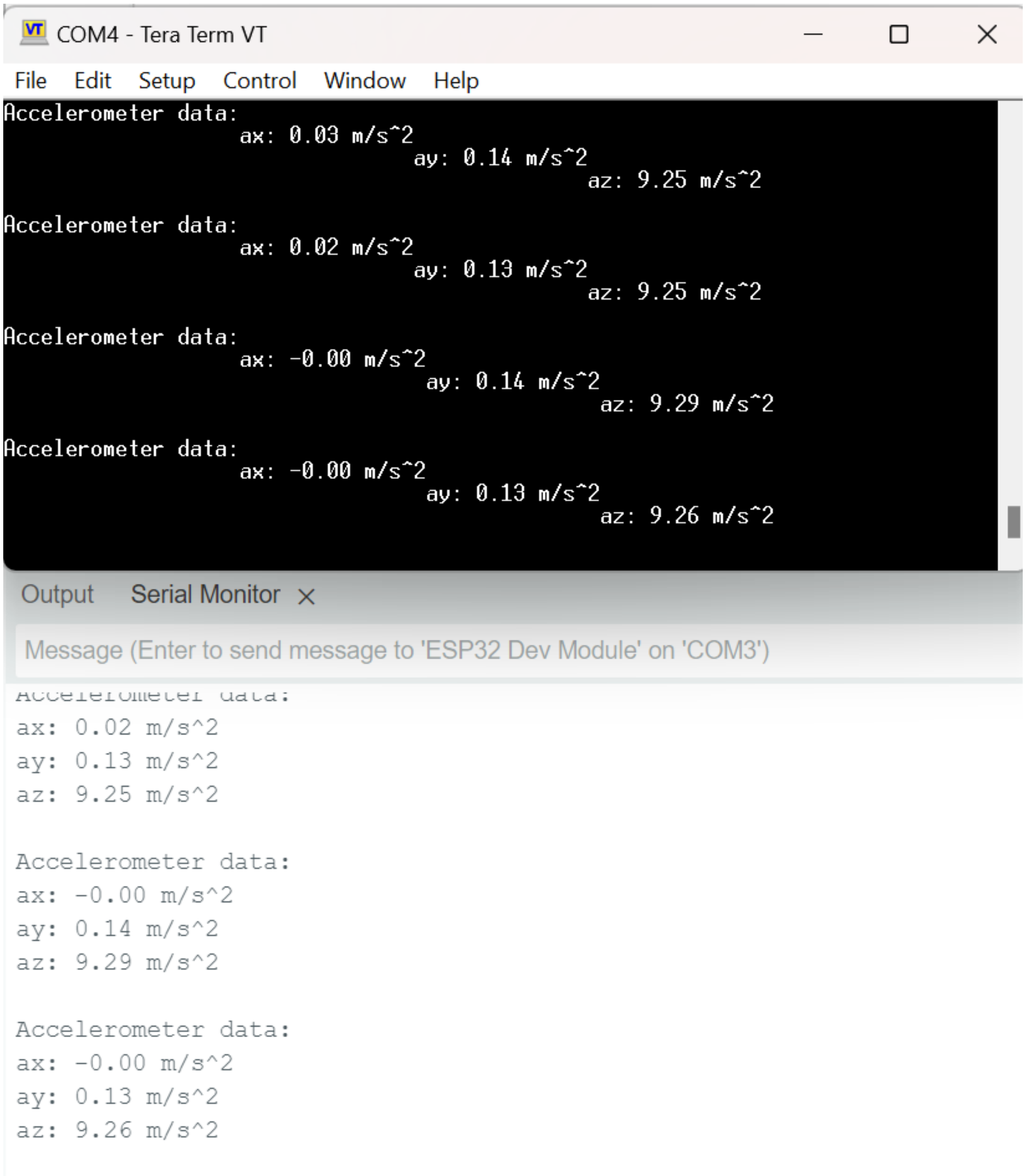

Figure 9.4*– Bluetooth Transmission of ICM-42670 Data to Tera Term*

9.3.2. VEML 7700 Light Sensor

Software testing for the VEML 7700 light sensor involves verifying its integration with the ESP32 and ensuring accurate light intensity measurements. To begin, the appropriate library for the VEML 7700 should be installed in the Arduino IDE. A sketch can then be written to initialize the sensor and read light intensity values at regular intervals. The Serial Monitor can be used to display these values, allowing for a visual inspection of the sensor's output. To validate the accuracy of the measurements, the sensor can be exposed to different light conditions, ranging from complete darkness to bright sunlight. The readings from the sensor should correspond to the expected light intensity levels. Additionally, the responsiveness of the sensor can be tested by rapidly changing the light conditions and observing how quickly the readings adapt to the new environment.

```
15:22:13.077 -> Lux: 73.86
15:22:13.701 -> White: 20736
15:22:13.701 -> Raw Ambient Light Sensing: 10258
15:22:13.701 ->
15:22:14.715 -> Lux: 80.07
15:22:15.322 -> White: 21650
15:22:15.322 -> Raw Ambient Light Sensing: 11121
15:22:15.322 ->
15:22:16.319 -> Lux: 78.24
15:22:16.914 -> White: 21202
15:22:16.914 -> Raw Ambient Light Sensing: 10866
15:22:16.914 ->
15:22:17.896 -> Lux: 0.94
15:22:18.504 -> White: 661
15:22:18.504 -> Raw Ambient Light Sensing: 131
15:22:18.504 ->
15:22:19.477 -> Lux: 0.93
15:22:20.109 -> White: 661
15:22:20.109 -> Raw Ambient Light Sensing: 129
```

***Figure 9.5** – Software Testing Output of the VEML 7700 Light Sensor*

9.3.3. PATIKIL 3W LED

Testing the software for the PATIKIL 3W LED involves ensuring proper control and functionality when integrated with the ESP32. The first step is to connect the LED to the appropriate GPIO pins on the ESP32 and verify the connections. A sketch can then be written to control the LED's behavior, such as turning it on and off, adjusting its brightness using PWM (Pulse Width Modulation), and creating various lighting patterns. The Serial Monitor can be used to send commands to the ESP32, triggering different LED functions and observing the LED's response. It is important to test the LED's performance at different brightness levels and ensure that it operates as expected without any flickering or inconsistencies. Additionally, the LED's power consumption can be monitored to ensure it stays within the specified limits and does not cause any power-related issues to the ESP32 or other components.

9.3.4. ICM-42670 3-Axis Accelerometer Gyroscope

Software testing for the ICM-42670 Accelerometer/Gyroscope involves verifying its integration with the ESP32 and ensuring accurate motion tracking. The first step is to install the necessary ICM-42670 library in the Arduino IDE. A sketch can then be written to initialize the sensor and read accelerometer and gyroscope data at regular intervals. The Serial Monitor can be used to display the raw sensor readings, allowing for a visual inspection of the data. To validate the accuracy of the measurements, the sensor can be subjected to various motion scenarios, such as tilting, rotating, and shaking. The readings from the sensor should accurately reflect the performed motions. Additionally, the responsiveness of the sensor can be tested by rapidly changing its orientation and observing how quickly the readings update. Calibration procedures can also be implemented and tested to ensure the accuracy of the sensor over time.

```
Acceleration X: 0.01, Y: 0.17, Z: 9.29 m/s^2
Rotation X: -0.08, Y: 0.04, Z: -0.00 rad/s
Temperature: 25.54 degC

Acceleration X: -0.01, Y: 0.20, Z: 9.28 m/s^2
Rotation X: -0.08, Y: 0.04, Z: -0.01 rad/s
Temperature: 25.52 degC

Acceleration X: 0.04, Y: 0.17, Z: 9.29 m/s^2
Rotation X: -0.08, Y: 0.04, Z: -0.01 rad/s
Temperature: 25.53 degC

Acceleration X: -0.03, Y: 0.19, Z: 9.28 m/s^2
Rotation X: -0.08, Y: 0.04, Z: -0.01 rad/s
Temperature: 25.57 degC
```

***Figure 9.6** – Software Testing Output of ICM-42670*

9.3.5. TTP223 Haptic Sensors

Software testing for the TTP223 Haptic Sensors involves verifying their integration with the ESP32 and ensuring reliable touch detection. The first step is to connect the haptic sensors to the appropriate GPIO pins on the ESP32 and verify the connections. A sketch can then be written to initialize the sensors and continuously monitor their state. The Serial Monitor can be used to display the touch events detected by each sensor. To test the responsiveness and accuracy of the sensors, various touch scenarios can be simulated, such as single taps, double taps, and long presses. The sketch should accurately detect and differentiate between these different touch events. Additionally, the sensors' sensitivity can be adjusted programmatically, and the impact of these adjustments on touch detection accuracy should be verified. It is also important to test the sensors' performance in different environmental conditions, such as varying temperatures and humidity levels, to ensure consistent and reliable touch detection.

9.3.6. Interrupt Testing

9.3.6.1. ICM-42670 Accelerometer/Gyroscope

To test the interrupt functionality of the ICM-42670, the sensor is connected to the ESP32, with the interrupt pin connected to a designated GPIO pin. A sketch is developed to configure the ICM-42670 to generate interrupts based on specific conditions, such as motion detection or data availability.

The sketch initializes theICM-42670 and sets up the interrupt configuration, specifying the desired interrupt sources and thresholds. The ESP32 is programmed to attach an interrupt service routine (ISR) to the designated GPIO pin, which is triggered whenever an interrupt occurs.

During the testing process, various motion scenarios are simulated to verify that the ICM-42670 generates interrupts as expected. The ISR is designed to capture and process the interrupt events, such as reading the sensor data or updating the system state. The Serial Monitor is used to display the interrupt events and the corresponding sensor data, allowing for visual verification of the interrupt functionality.

Additional tests are performed to ensure the reliability and responsiveness of the interrupts, including subjecting the sensor to rapid motion changes and verifying that the interrupts are triggered consistently and without false positives.

9.3.6.2. TTP223 Haptic Sensors

Testing the interrupt functionality of the TTP223 haptic sensors follows a similar approach. Each sensor's output pin is connected to a separate GPIO pin on the ESP32, configured as an interrupt input. A sketch is written to attach ISRs to each interrupt pin, enabling the ESP32 to detect and respond to touch events.

The sketch initializes the haptic sensors and sets up the interrupt configuration, specifying the desired interrupt trigger mode (e.g., rising edge or falling edge) and any necessary debounce settings to avoid false triggers.

During the testing process, various touch scenarios are simulated, such as single taps, double taps, and long presses, to verify that the sensors generate interrupts accurately. The ISRs are designed to capture and process the interrupt events, updating the system state or triggering specific actions based on the type of touch event detected.

The Serial Monitor is utilized to display the interrupt events and the corresponding sensor states, allowing for visual confirmation of the interrupt functionality. Additional tests are conducted to ensure the reliability and responsiveness of the interrupts, including subjecting the sensors to rapid and repeated touch events and verifying that the interrupts are triggered consistently and without false positives.

By thoroughly testing the interrupt functionality of the ICM-42670 Accelerometer Gyroscope and TTP223 haptic sensors individually, confidence is established in their ability to provide reliable and responsive interrupt-driven communication when integrated into HelmetIQ. This testing process helps identify and address any potential issues or limitations before proceeding with the full component integration.

9.4. Component Integration

After successfully testing each component individually, the next crucial step is to integrate all the components into a single, cohesive system. The integration process involves connecting the ESP32 microcontroller with the VEML7700 light sensor, PATIKIL 3W LED, ICM-42670 3-Axis Accelerometer Gyroscope, and TTP223 haptic sensors.

The ESP32 serves as the central controller, and all other components are connected to it. The VEML7700 light sensor is connected to the ESP32 using the I2C communication protocol, with the SDA and SCL lines connected to GPIO pins 21 and 22, respectively. The PATIKIL 3W LED is connected to an appropriate GPIO pin on the ESP32, which is configured as an output to control the LED's brightness using PWM.

The ICM-42670 Accelerometer Gyroscope is connected to the ESP32 using interrupt-driven communication. The interrupt pin of the ICM-42670 is connected to a designated GPIO pin on the ESP32, allowing the microcontroller to be notified immediately when new data is available from the sensor. This approach ensures fast response times and efficient data processing, which is crucial for detecting potential collisions.

Similarly, the TTP223 haptic sensors are connected to the ESP32 using interrupts. Each sensor's output pin is connected to a separate GPIO pin on the ESP32, configured as an interrupt input. When a touch event occurs, the corresponding interrupt is triggered, and the ESP32 can quickly respond to the user's input.

To enable wireless communication between the smart bike helmet and a mobile app, the ESP32's built-in Bluetooth functionality is utilized. The ESP32 is configured as a Bluetooth Low Energy (BLE) peripheral, allowing it to transmit sensor data to the mobile app. The mobile app, acting as a BLE central device, receives the data and processes it further.

The software integration process involves developing a comprehensive sketch that combines the functionality of all components, including interrupt handling for the ICM-42670 and haptic sensors, Bluetooth communication, and collision detection algorithms. The main sketch initializes the sensors, sets up the necessary interrupts, and establishes a BLE connection with the mobile app.

In the event of a potential collision, detected by the ICM-42670 accelerometer and gyroscope data, the ESP32 sends an alert to the mobile app via Bluetooth. The mobile app, upon receiving the alert, can then send out a text message to the user's listed emergency contacts, informing them of the situation and providing relevant information such as the user's location.

Throughout the integration process, rigorous testing is conducted to ensure the proper functioning of the interrupt-driven communication, Bluetooth data transmission, and collision detection algorithms. This testing includes simulating various scenarios, such as sudden impacts and changes in orientation, to verify the system's responsiveness and reliability.

By successfully integrating all the components using the ESP32 microcontroller, interrupt-driven communication, and Bluetooth connectivity, the smart bike helmet provides a comprehensive solution for enhancing cyclist safety. The integration of the mobile app further extends the helmet's capabilities, enabling quick notification of emergency contacts in the event of a potential collision.

9.5. Plan for SD2

The Senior Design 2 phase of the HelmetIQ project was a crucial period dedicated to transforming the conceptual design into a fully functional product. This phase encompasses three main areas: construction, testing, and final demonstration.

The construction phase began with the integration of the various components onto the finalized PCB design. This process involved carefully assembling the hardware components, including the microcontroller unit (MCU), sensors (accelerometer, gyroscope, light sensor), LEDs for lighting and signaling, and the Bluetooth module. The team ensured that all components are properly connected and communicating with each other as designed.

A significant part of the construction phase was the implementation of the haptic feedback system for turn signals. This required precise integration of capacitive touch sensors and haptic actuators into the helmet structure, ensuring they function reliably through the helmet material.

The dynamic lighting system, including the front headlight, rear brake light, and turn signal LEDs, were carefully installed to provide optimal visibility and functionality. The team payed special attention to the weatherproofing of these components to ensure durability in various cycling conditions.

Along with the hardware assembly, the team finalized the development of the companion mobile app. This involved implementing the user interface, ensuring seamless Bluetooth communication with the helmet, and refining the collision detection response system.

Testing:

Once the prototype is assembled, the team entered an intensive testing phase. This involved comprehensive evaluation of all HelmetIQ features, including:

1. Haptic feedback turn signals: Testing response time, reliability, and user experience.
2. Accelerometer-based brake light activation: Ensuring accurate detection of deceleration and timely light activation.
3. Dynamic lighting system: Verifying proper function in various light conditions.
4. Collision detection system: Rigorous testing of impact detection and emergency response activation.
5. Battery life and charging capabilities: Ensuring the system meets the 4-hour continuous use requirement.
6. Mobile app functionality: Testing all features, including emergency contact upon collision detection.

The team conducted both laboratory and real-world tests to ensure the helmet performs as expected in various scenarios. This may include simulated collision tests, extended wear tests for comfort and battery life, and field tests in different weather and lighting conditions.

Any issues or shortcomings identified during the testing phase were addressed through careful adjustments and refinements to both hardware and software components. The team iterated on the design as necessary to meet all specified engineering requirements and ensure a high-quality user experience.

Final Demonstration:

The culmination of the Senior Design 2 phase was the preparation and delivery of the final presentation and demonstration. This involved creating a comprehensive showcase of HelmetIQ's features and capabilities.

The demonstration included a live display of the helmet's dynamic lighting system, showcasing how it adapts to changing light conditions.

A key part of the demonstration was showing the collision detection system and its integration with the mobile app. This could involve a controlled impact test to trigger the emergency response feature, highlighting the potential life-saving capabilities of HelmetIQ.

Throughout this process, the team maintained a focus on meeting the project's core objectives: enhancing cyclist safety through intelligent lighting, extra safety sensors, and offering a seamless user experience. The final HelmetIQ iteration aims to be a comprehensive solution that integrates multiple features into a single, user-friendly device, setting a new standard for smart bike helmets in terms of functionality, safety, and convenience.

10. Administrative Content

10.1. Budget

The budget for the smart helmet project includes expenses for components, prototyping, testing, and miscellaneous items. Funding was sourced from personal contributions from team members. The goal of this project is to keep all costs within $270. This was quickly passed, and the project was far over budget.

Table 10.1 *– Estimated Budget Allocation*

Expense Category	Estimated Cost ($)	Description
Components	220	Includes sensors, microcontrollers, PCB materials, haptic motors, and other electronic parts.
Prototyping	10	3D printing materials, helmet base, and initial prototype fabrication.
Testing		Equipment for testing sensors and sensor resilience
Miscellaneous	30	Unforeseen expenses, shipping costs, and administrative fees.
Total	**260**	

10.2. Bill of Materials

Table 10.2 *- Bill of Materials*

Material	Unit Cost	Quantity	Total Cost
Capacitive Touch Sensors	$5.99	2	$11.58
Haptic Drivers	$11.23	1	$11.23
Haptic Motor Driver	$11.23	2	$22.26
ESP32 MCU	$27.00	3	$27.00
LED Strip Lights	$14.99	5	$74.95
Helmet (2 identical ones)	$30.00	1	$30.00
Battery Rechargable AA	$20.00	2	$40.00
Battery(Li-Ion)	$2.50	3	$7.50
Multiplexer	$6.08	1	$6.08
Accelerometer/Gyroscope	$10.00	1	$10.00
LED Driver	$5.99	3	$17.97
1st PCB	$155.95	1	$155.95
2nd PCB	$88.77	1	$88.77
Digi-key (Parts)	$128.98	1	$128.98
HL-LED/Driver	$16.23	1	$16.23
Gyro-breakout	$9.99	1	$9.99
Light-breakout	$12.99	1	$12.99
Turn-LED	$7.49	1	$7.49
USB-CP2102	$7.49	1	$7.49
BrakeLED	$20.56	1	$20.56
Phone SMS Plan	$45	1	$45
TOTAL			**$752.02**

10.3. Work Distribution

The worktable for describing and assigning tasks to certain group members is below. It is important to note that the member listed is responsible for the component, but other members may be actively involved in that aspect as well. This is because many systems within our project are sequentially dependent so one must be completed for the development of another to begin. We succeeded as a team and similar to the workplace the explicit designation of tasks was not nearly as important as the project getting done and us meeting our objectives. This table shows what each member is responsible for.

Table 10.3 *- Distribution of Work*

Electrical Engineering	Responsibilities
Ana-Victoria Elias	Integration of turn signals
	Integration of Haptic Touch Feedback
	PCB Design
	Determining battery pack configuration and protection limits

Electrical Engineering	Responsibilities
Hector Agosto	Integration of Headlight
	Integrations of Light Sensor
	Integration of Gyroscope/Accelerometer
	PCB Design

Computer Engineering	Responsibilities
Christopher Bowerfind	Mobile UX Design
	Light sensor MCU implementation
	LED MCU implementation
	Website Management

Computer Engineering	Responsibilities
Ethan Hymans	Component Integration
	Software Design and Implementation
	Wireless Communications
	Helmet Design

10.4. Project Milestones

Table 10.4 *- Senior Design I Documentation Milestones*

Milestone	Start Date	Estimate End Date	Duration
Project Idea, Planning, and Research	5/21/2024	5/24/2024	3 days
Finalized Project Direction	5/21/2024	5/30/2024	1.5 weeks
10-Page Milestone (D&C)	5/24/2024	5/31/2024	1 week
30-Page Milestone	5/31/2024	6/13/2024	2 weeks
60-Page Milestone	6/14/2024	6/28/2024	2 weeks
90-Page Milestone	6/28/2024	7/12/2024	2 weeks
120-Page Milestone	7/9/2024	7/23/2024	2 weeks

Table 10.5 *- Senior Design I Project Design Milestones*

Milestone	Start Date	Estimate End Date	Duration
Initial Code Design with Arduino	6/14/2024	6/28/2024	2 weeks
Individual Component Testing	6/14/2024	6/28/2024	2 weeks
Initial Mobile App Development	6/28/2024	7/05/2024	1 week
Initial PCB Prototype Design	6/21/2024	7/19/2024	4 weeks
Finalized PCB Layout	7/12/2024	7/23/2024	1.5 weeks

Table 10.6 *- Senior Design II Project Design Milestones*

Milestone	Start Date	Estimate End Date	Duration
Final PCB Ordered/Arrives	7/23/2024	8/19/2024	4 weeks
Features Integrated into Prototype	8/19/2024	9/16/2024	4 weeks
Final App Development	9/09/2024	9/23/2024	2 weeks
Thorough Testing and Adjustments	9/23/2024	10/14/2024	3 weeks
Finalize Documentation	10/14/2024	10/28/2024	2 weeks
Practice Final Presentation / Demo	10/28/2024	11/04/2024	1 week

11. Conclusion

The HelmetIQ smart bike helmet project represents a significant step forward in cyclist safety and technology integration. By combining intelligent lighting, advanced sensors, and mobile app connectivity, HelmetIQ offers a comprehensive solution that enhances visibility, situational awareness, and emergency response capabilities for riders. This project not only showcases the power of interdisciplinary collaboration but also highlights the potential for technology to transform the way we approach personal safety in various domains.

Throughout the development process, our team faced and overcame numerous challenges, from hardware selection and integration to software design and implementation. The research and investigation phase allowed us to evaluate various technologies and components, ultimately selecting those that best aligned with our project goals and constraints. Key decisions, such as the choice of the ESP32 microcontroller, VEML7700 light sensor, ICM-42670 accelerometer and gyroscope, and TTP223 capacitive touch sensors, laid the foundation for a robust and efficient system. These selections were based on careful consideration of factors such as power consumption, accuracy, and compatibility, ensuring that each component contributed to the overall performance and reliability of the HelmetIQ system.

The hardware design process involved planning and execution, with a focus on power management, sensor integration, and PCB layout. By optimizing component placement, utilizing interrupt-driven communication, and implementing effective power regulation, we ensured reliable operation and minimized power consumption. The modular design approach allowed for flexibility and scalability, enabling future enhancements and adaptations. This modularity not only simplifies maintenance and upgrades but also opens up possibilities for customization and integration with other smart devices and platforms.

On the software front, our team developed an architecture that seamlessly integrated the various subsystems. From lighting control and sensor data processing to collision detection and mobile app communication, each software component was designed to work in harmony, providing a responsive and intuitive user experience. The use of interrupt-driven programming and efficient data structures optimized system performance and resource utilization. By leveraging best practices in embedded systems development and adhering to coding standards, we ensured that the software was not only functional but also maintainable and extensible.

The integration of a companion mobile app adds a new dimension to the HelmetIQ experience. By leveraging Bluetooth connectivity and intuitive UI design, the app empowers users to customize settings, monitor ride data, and access valuable safety features. The inclusion of the auto-SMS for emergency contact notification exemplifies our commitment to leveraging existing technologies to enhance rider safety. The app not only provides a seamless interface for interacting with the helmet but also opens up possibilities for future enhancements, such as real-time route planning, community-driven safety insights, and integration with other fitness and health platforms.

Looking ahead, the HelmetIQ project has the potential to revolutionize not only the cycling accessory market but also the broader landscape of personal safety technology. Its modular design and scalable architecture lay the groundwork for future iterations and expansions. Possible enhancements could include the integration of additional sensors for environmental monitoring, such as air quality and temperature sensors, enabling riders to make informed decisions about their routes and riding conditions. The development of predictive maintenance features, leveraging machine learning algorithms to analyze sensor data and detect potential issues before they occur, could further enhance the longevity and reliability of the helmet. Moreover, the exploration of advanced data analytics and personalization techniques could unlock new insights into rider behavior and preferences, enabling the development of tailored safety features and recommendations.

The impact of HelmetIQ extends far beyond the realm of cycling. The core technologies and design principles employed in this project have applications in various other domains, such as construction, mining, and extreme sports. By adapting the HelmetIQ framework to the specific needs and challenges of these industries, we can contribute to the development of safer, smarter, and more connected personal protective equipment. For example, in the construction industry, a smart helmet equipped with fall detection, proximity sensors, and real-time communication capabilities could significantly reduce the risk of accidents and improve emergency response times. Similarly, in the mining sector, a helmet with integrated gas sensors, headlamps, and location tracking could enhance worker safety and efficiency in hazardous environments.

Moreover, the success of the HelmetIQ project highlights the importance of interdisciplinary collaboration and user-centric design in addressing complex challenges. By bringing together expertise from electrical engineering and computer engineering, we were able to create a solution that not only met technical requirements but also prioritized the needs and preferences of end-users. This approach underscores the value of diverse perspectives and skill sets in driving innovation and solving real-world problems.

As we reflect on the journey of developing HelmetIQ, we are proud of the progress we have made and the potential impact our project can have on the lives of cyclists and beyond. However, we also recognize that this is just the beginning. As technology continues to advance and new challenges emerge, we must remain committed to pushing the boundaries of what is possible in the realm of smart safety gear. This requires ongoing research, collaboration, and a willingness to adapt and evolve our solutions in response to changing needs and contexts.

In conclusion, the HelmetIQ smart bike helmet project exemplifies the power of interdisciplinary collaboration, innovative thinking, and a user-centric approach to problem-solving. By successfully integrating cutting-edge technologies and thoughtful design, we have created a product that has the potential to save lives and revolutionize the way cyclists interact with their equipment. As we move forward, we remain committed to refining and expanding the capabilities of HelmetIQ, serving as a catalyst for the development of safer, smarter, and more connected personal protective equipment across industries. With its strong foundation and limitless potential, HelmetIQ is poised to make a lasting impact on the cycling community and beyond, setting a new standard for intelligent, proactive, and user-centric safety solutions.

Appendix

Appendix A - References

[1] Adafruit Industries, "Adafruit 9-DOF Absolute Orientation IMU - BNO055," Adafruit. [Online]. Available: https://www.adafruit.com/product/159. [Accessed: Jul. 2, 2024].

[2] Adafruit Industries, "Adafruit BNO055 Absolute Orientation Sensor," Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bno055-absolute-orientation-sensor.pdf. [Accessed: Jul. 8, 2024].

[3] Adafruit Industries LLC, "Adafruit SCD-40 - True CO2, Temperature and Humidity Sensor," Digi-Key Electronics. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/5378/16056942. [Accessed: Jul. 8, 2024].

[4] Adafruit, "Adafruit VEML7700 Lux Sensor Breakout," Adafruit. [Online]. Available: https://learn.adafruit.com/adafruit-veml7700/pinouts. [Accessed: Jul. 08, 2024].

[5] Amazon, "UNIT 1 FARO Smart Bike Helmet with MIPS, LED Turn Signals, Brake Lights, Crash Detection and Automatic SOS," Amazon. [Online]. Available: https://www.amazon.com/UNIT-Certified-Detection-Automatic-Blackbird/dp/B08BCLTJPX/. [Accessed: May 30, 2024].

[6] A. Johnson, "Comfortable and High-Tech," Amazon Customer Reviews. [Online]. Available: https://www.amazon.com/gp/customer-reviews/R6N71K1G3S8BN/. [Accessed: May 30, 2024].

[7] A. Molnar, “Types of Light Sensors,” Gamma Science, Mar. 27, 2020. [Online]. Available: https://gamma-sci.com/2020/03/27/types-of-light-sensors/ [Accessed: Jun. 24, 2024].

[8] Android Developers, "Write your app," Android Developers. [Online]. Available: https://developer.android.com/studio/write. [Accessed: Jul. 2, 2024].

[9] Apple Developer Documentation. "Transitioning to Swift from Objective-C." 2023. [Online]. Available: https://developer.apple.com/documentation/swift/transitioning_to_swift_from_objective_c. [Accessed: Jun. 24, 2024].

[10] Codemagic Blog, "Android Studio vs Visual Studio Code," Codemagic. [Online]. Available: https://blog.codemagic.io/android-studio-vs-visual-studio-code/. [Accessed: Jul. 1, 2024].

[11] DFRobot, "DFRobot Capacitive Touch Sensor SKU DFR0030," DFRobot Wiki. [Online]. Available: https://wiki.dfrobot.com/DFRobot_Capacitive_Touch_Sensor_SKU_DFR0030. [Accessed: Jul. 2, 2024].

[12] Digi-Key Electronics, "Triple Axis Accelerometer Breakout - ADXL335," Available: https://www.digikey.com/en/htmldatasheets/production/1732757/0/0/1/sen-11028. [Accessed: Jul. 8, 2024].

[13] Embedded Systems Design Magazine. "Choosing the Right Development Framework." 2022. [Online]. Available: https://www.esdmagazine.com/articles/framework-comparison. [Accessed: Jun. 24, 2024].

[14] Explore Labs, "LSM6DS33 3D Accelerometer + 3D Gyroscope Board," Tindie, Available: https://www.tindie.com/products/explorelabs/lsm6ds33-3d-accelerometer-3d-gyroscope-board/. [Accessed: Jul. 7, 2024].

[15] freeCodeCamp, "What's Boilerplate and Why Do We Use It?," freeCodeCamp. [Online]. Available: https://www.freecodecamp.org/news/whats-boilerplate-and-why-do-we-use-it-let-s-check-out-the-coding-style-guide-ac2b6c814ee7/. [Accessed: Jul. 2, 2024].

[16] GrabCAD Community, "Bike Helmet," GrabCAD. [Online]. Available: https://grabcad.com/library/bike-helmet-15. [Accessed: May 28, 2024].

[17] IEEE. "Improving Developer Productivity with PlatformIO: An Empirical Study." 2021. [Online]. Available: https://www.ieee.org/publications/platformio-study. [Accessed: Jun. 24, 2024].

[18] IEEE, "A survey on the role of wireless sensor networks in cyber-physical systems," IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7758229&tag=1. [Accessed: Jul. 1, 2024].

[19] Incycle, "Cannondale Quick CE EN Adult Helmet," Incycle. [Online]. Available: https://www.incycle.com/products/cannondale-quick-ce-en-adult-helmet. [Accessed: Jul. 03, 2024].

[20] IRCOBI, "Development and validation of a bicyclist anthropometric test device," IRCOBI. [Online]. Available: https://www.ircobi.org/wordpress/downloads/irc17/pdf-files/30.pdf. [Accessed: Jul. 3, 2024].

[21] Journal of Mobile Computing. "Efficiency and Performance in Cross-Platform Mobile Development." 2022. [Online]. Available:

https://www.journalofmobilecomputing.com/articles/react-native-vs-native. [Accessed: Jun. 24, 2024].

[22] J. Smith, "Amazing Helmet with Great Features," Amazon Customer Reviews. [Online]. Available: https://www.amazon.com/gp/customer-reviews/R247RXN1WXBOIZ/. [Accessed: May 30, 2024].

[23] KTH Royal Institute of Technology, "GL5537 Photoresistor Datasheet," Available: https://www.kth.se/social/files/54ef17dbf27654753f437c56/GL5537.pdf. [Accessed: Jul. 6, 2024].

[24] Lua, "About Lua," Lua. [Online]. Available: https://www.lua.org/about.html. [Accessed: Jul. 1, 2024].

[25] Lumos Helmet. "Lumos Ultra." [Online]. Available: https://ridelumos.com/products/lumos-ultra. [Accessed: May 30, 2024].

[26] MDPI, "Sensors: Real-Time Monitoring in Smart Cities," MDPI. [Online]. Available: https://www.mdpi.com/1424-8220/22/19/7405. [Accessed: Jul. 1, 2024].

[27] Miro. "Miro Online Whiteboard." [Online]. Available: https://miro.com/app/dashboard/. [Accessed: May 30, 2024].

[28] Mouser Electronics, "Cree XLamp XP-G3 LEDs," Mouser. [Online]. Available: https://eu.mouser.com/new/cree/cree-xlamp-xp-g3-horizon-leds/. [Accessed: Jul. 2, 2024].

[29] National Highway Traffic Safety Administration. "Bicycle Safety." Countermeasures That Work. [Online]. Available: https://www.nhtsa.gov/book/countermeasures-that-work/bicycle-safety#:~:text=Bicyclist%20injuries%20remain%20consistently%2C%20disproportionately,crashes%20averaged%20about%2045%2C400%20yearly. [Accessed: May 30, 2024].

[30] N. S. E. Display, "DIP LED Vs. SMD LED: Major Differences and Applications," [Online] NSELED, https://nseledcloud.com/dip-led-vs-smd-led/. [Accessed: Jul. 08, 2024].

[31] OneSignal Blog, "The Benefits and Challenges of iOS vs Android App Development," OneSignal. [Online]. Available: https://onesignal.com/blog/the-benefits-and-challenges-of-ios-vs-android-app-development/. [Accessed: Jul. 3, 2024].

[32] P. K. "NiMH Vs LiPo Vs Li ion – Which Is Better?" ElectronicsHub, 22-Jan-2024. [Online]. Available: https://www.electronicshub.org/nimh-vs-lipo-vs-li-ion/. [Accessed: Jul. 2, 2024].

[33] PATIKIL 3W DC 3-3.7V LED Chip Bulb, Aluminum COB Lamps Bulb White Light 6500K 300LM 1000mA 60x8mm [Product on Amazon.com], [Accessed: July 8, 2024].

[34] PiezoDrive, "Piezo Bender Actuators," PiezoDrive. [Online]. Available: https://www.piezodrive.com/actuators/piezo-bender-actuators/. [Accessed: Jul. 2, 2024].

[35] Piezo, "Haptic Actuators: Comparing Piezo, ERM, and LRA," Blog, Piezo. [Online]. Available: https://blog.piezo.com/haptic-actuators-comparing-piezo-erm-lra. [Accessed: Jul. 2, 2024].

[36] PlatformIO, "What is PlatformIO?," PlatformIO. [Online]. Available: https://docs.platformio.org/en/latest/what-is-platformio.html. [Accessed: Jul. 2, 2024].

[37] Precision Microdrives, "Eccentric Rotating Mass Vibration Motors (ERMs)," Precision Microdrives. [Online]. Available: https://www.precisionmicrodrives.com/eccentric-rotating-mass-vibration-motors-erms. [Accessed: Jul. 2, 2024].

[38] Precision Microdrives, "Linear Resonant Actuators (LRAs)," Precision Microdrives. [Online]. Available: https://www.precisionmicrodrives.com/linear-resonant-actuators-lras. [Accessed: Jul. 2, 2024].

[39] ScienceDirect, "Comb Drive Actuator," ScienceDirect. [Online]. Available: https://www.sciencedirect.com/topics/engineering/comb-drive-actuator. [Accessed: Jul. 1, 2024].

[40] Silicon Lightworks, "What are COB LEDs?" Available: https://siliconlightworks.com/resoures/what-are-cob-leds#:~:text=As%20a%20result%2C%20a%20relatively,%2C%20ultra%2Defficient%20light%20systems. [Accessed: Jul. 8, 2024].

[41] SMBOM, "Latest Trends in PCB Technology," SMBOM. [Online]. Available: https://www.smbom.com/news/13602. [Accessed: Jul. 08, 2024].

[42] SparkFun Electronics, "AS7262 6-Channel Visible Spectral_ID Device," Available: https://cdn.sparkfun.com/assets/f/b/c/c/f/AS7262.pdf. [Accessed: Jul. 6, 2024].

[43] SparkFun, "SparkFun Environmental Sensor Breakout - BME68x (Qwiic) Hookup Guide," SparkFun. [Online]. Available: https://learn.sparkfun.com/tutorials/sparkfun-environmental-sensor-breakout---bme68x-qwiic-hookup-guide/introduction. [Accessed: Jul. 2, 2024].

[44] “SparkFun Qwiic Haptic Driver - DA7280 - ROB-17590 - SparkFun Electronics,” www.sparkfun.com. https://www.sparkfun.com/products/17590 (accessed Jul. 08, 2024).

[45] Stack Overflow Developer Survey. "Most Popular Developer Tools and Environments." 2023. [Online]. Available: https://insights.stackoverflow.com/survey/2023. [Accessed: Jun. 24, 2024].

[46] Texas Instruments, "Fundamentals of Linear Regulators," SSZB151, Oct. 2018. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ti.com/lit/ml/sszb151/sszb151.pdf. [Accessed: Jul. 2, 2024].

[47] University of Southampton. "A Comparative Study of Microcontroller Development Environments." 2022. [Online]. Available: https://www.southampton.ac.uk/compstudy/mcu-environments. [Accessed: Jun. 24, 2024].

[48] Vishay Intertechnology, Inc., "VEML7700: Ambient Light Sensor with I2C Interface," Available: https://www.vishay.com/docs/84286/veml7700.pdf. [Accessed: Jul. 8, 2024].

[49] Vishay, "VEML7700: Ambient Light Sensor with I2C Interface," Vishay. [Online]. Available: https://www.vishay.com/en/product/84286/. [Accessed: Jul. 08, 2024].

[50] TDK InvenSense. "ICM-42670-P: High-Performance 6-Axis MotionTracking™ IMU." 2023. [Online]. Available: https://invensense.tdk.com/products/motion-tracking/6-axis/icm-42670-p/. [Accessed: Nov. 24, 2024].

[51] Second Chance, "Livall BH51M Smart Urban Cycle Helmet," Second Chance. [Online]. Available: https://www.secondchance.co.uk/livall-smart-helmets-exclusive-uk-distributors/livall-bh51m-smart-urban-cycle-helmet/bh51m_exploded/. [Accessed: Jul. 08, 2024].

[52] "Instructables, 'LM1117 Linear Voltage Regulator,' [Online]. Available: https://www.instructables.com/LM1117-Linear-Voltage-Regulator/. [Accessed: 21-Jul-2024]."

[53] Bike Tires Direct, "Giro Switchblade MIPS MTB Helmet," Bike Tires Direct. [Online]. Available: https://www.biketiresdirect.com/product/giro-switchblade-mips-mtb-helmet?v=mbgm00m&adl=1&gad_source=1&gclid=CjwKCAjw4_K0BhBsEiwAfVVZ_0gbTGtbPJXBSVDth6J7IccK4vKkDSpEpkqM-iKsnCdHVCQ6WqwsaRoChQAQAvD_BwE. [Accessed: Jul. 08, 2024].

[54] Protec, "Low Pro Certified Helmet - Matte Black," Protec. [Online]. Available: https://protecbrand.com/collections/bike-bmx/products/low-pro-certified-helmet-matte-black. [Accessed: Jul. 08, 2024].

[55] Bell Helmets, "Qualifier Motorcycle Helmet," Bell Helmets. [Online]. Available: https://www.bellhelmets.com/motorcycle/p/qualifier-motorcycle-helmet/BL-7049225.html?googleMerchant=1. [Accessed: Jul. 08, 2024].

[56] Cannondale, "Quick Adult Helmet," Cannondale. [Online]. Available: https://www.cannondale.com/en-us/gear/helmets/adult/quick-adult-helmets/quick-adult-helmet?sku=CH4701U40SM&

[57] "Bike Helmet," GrabCAD, Accessed: Jul. 23, 2024. [Online]. Available: https://grabcad.com/library/bike-helmet-15

Appendix B – LLM Prompt

"I would like some help brainstorming ideas for my Senior Design project. Here are the guidelines for the project: Senior Design is a two-semester sequence (3+3 credit hours). SD 1:

investigation, design, data, schematic, technical writing. SD 2: implementation, testing, redesign, improvement, final testing, demonstration, presentation
- Open-ended design and implementation (ABET)
- Experience how to apply knowledge you have learned and skills you have developed; bridge course work to real world application
- Opportunity to grow technical depth and breadth, self learning
- Project and time management
- Making progress on a consistent manner
- Completing tasks continuously and regularly
- Self motivation
- Work in a team (3-4 students) and be responsible for successes and mistakes Peer review, different grade in one group.
- Prototype must be fully functional as specified. Not performing is not an option.
- Must design your own printed circuit board (PCB) to meet significant PCB layout design requirement. No breadboard is allowed in the final prototype. Resources: tutorial, CAD software, equipment, tools, etc.

Our group is composed of 2 electrical engineering students and 2 computer engineering students. Give me a list of ideas that I could potentially design and accomplish with my group for Senior Design."

Appendix C – LLM Results

Section 1a: ChatGPT Idea Generation Results

"Given the mix of electrical and computer engineering skills in your group and the comprehensive nature of your Senior Design project requirements, it's important to choose a project that leverages the strengths and interests of all group members while meeting the technical and educational goals. Here are some project ideas that could be interesting and challenging:

1. <u>Smart Home Energy Management System:</u>
    - Design a system that intelligently manages power distribution and usage within a home. Use sensors and real-time data analytics to optimize energy consumption, incorporate renewable energy sources, and reduce costs.
2. <u>Autonomous Mobile Robot for Indoor Navigation:</u>
    - Develop a robot that can navigate indoor environments autonomously using sensors and cameras. This could be useful for security surveillance, delivery within large buildings like hospitals or universities, or as a guide for the visually impaired.
3. <u>IoT-Based Health Monitoring System:</u>
    - Create a network of wearable devices that monitor various health metrics such as heart rate, temperature, and movement. Data can be sent to a central system to monitor patients remotely, useful in eldercare or chronic disease management.
4. <u>Advanced Disaster Response Drone System:</u>

- Develop a drone equipped with sensors and cameras that can be deployed in disaster areas for search and rescue operations. Features might include thermal imaging for finding people, hazardous material detection, and automated pathfinding in difficult terrains.

5. Automated Aquaponics System:
    - Design a system that combines aquaculture (raising fish) and hydroponics (growing plants without soil) in a symbiotic environment. Use sensors to monitor water quality, fish health, and plant growth, automating feeding and watering schedules based on real-time data.
6. Smart Traffic Management System:
    - Develop a network of sensors and cameras to monitor traffic conditions and adjust traffic light timings in real-time to optimize flow, reduce congestion, and enhance safety.
7. Intelligent Video Surveillance System:
    - Create a system that uses machine learning to analyze video feeds in real-time for security purposes, capable of recognizing suspicious behaviors, tracking individuals, and sending alerts to authorities.
8. Voice-Controlled Home Automation System:
    - Design a comprehensive home automation system controlled by voice commands. It could integrate lighting, security, media, and climate systems into a cohesive voice-activated network.
9. E-Waste Material Sorter:
    - Develop a system that can sort electronic waste based on material composition using sensors, helping to automate the recycling process.
10. Smart Parking Solution:
    - Create a system that uses sensors and data analytics to manage and optimize parking space availability in real-time, reducing time spent searching for parking spots in busy areas.

These ideas can be scaled in complexity and scope depending on your team's specific interests, expertise, and the resources available. The key is to select a project that not only fulfills the academic requirements but also ignites enthusiasm and commitment among all team members." (OpenAI, 2024)

Section 1b: Claude Idea Generation Results

"Here are some potential ideas for your Senior Design project, considering your group's composition and the project guidelines:

1. Autonomous Drone for Search and Rescue Missions
- Design a drone with advanced sensors and image processing capabilities
- Implement algorithms for object detection, tracking, and navigation
- Integrate a communication system for remote control and data transmission

2. Smart Home Energy Management System
- Develop a system to monitor and control energy consumption in a household

- Integrate various sensors (e.g., temperature, occupancy, light) and smart appliances
- Create a user-friendly interface for monitoring and controlling the system remotely

<u>3. Wearable Health Monitoring Device</u>
- Design a compact, wearable device to monitor vital signs (e.g., heart rate, blood pressure, oxygen levels)
- Implement algorithms for data analysis and anomaly detection
- Develop a mobile app for data visualization and alerts

<u>4. Intelligent Traffic Management System</u>
- Create a system to optimize traffic flow using real-time data from sensors and cameras
- Implement machine learning algorithms for traffic pattern analysis and prediction
- Design a user interface for traffic controllers to monitor and manage the system

<u>5. Augmented Reality (AR) based Educational Tool</u>
- Develop an AR application to enhance learning experiences in various subjects (e.g., science, history, art)
- Create interactive 3D models and animations to visualize complex concepts
- Integrate a learning management system for tracking progress and providing feedback

<u>6. Autonomous Underwater Vehicle (AUV) for Marine Research</u>
- Design an AUV with various sensors (e.g., camera, sonar, temperature, pH) for data collection
- Implement navigation and control algorithms for autonomous operation
- Develop a data analysis platform for processing and visualizing collected data

<u>7. Smart Irrigation System</u>
- Create a system to optimize water usage in agriculture using sensor data (e.g., soil moisture, weather)
- Implement machine learning algorithms for predicting water requirements and controlling irrigation
- Develop a mobile app for remote monitoring and control of the system

<u>8. Gesture-controlled Robotic Arm</u>
- Design a robotic arm with multiple degrees of freedom
- Implement computer vision algorithms for gesture recognition
- Create a user interface for intuitive control of the robotic arm using hand gestures

<u>9. Intelligent Waste Management System</u>
- Develop a system to optimize waste collection and segregation using sensor data (e.g., weight, volume, type)
- Implement machine learning algorithms for waste classification and route optimization
- Create a web-based dashboard for monitoring and managing the waste management process

<u>10. Smart Parking System</u>
- Design a system to monitor and manage parking spaces in a city or campus
- Integrate various sensors (e.g., ultrasonic, camera) to detect parking space occupancy

- Develop a mobile app for users to find available parking spaces and navigate to them

These ideas cover a range of applications and technologies, providing ample opportunities for your group to apply their electrical and computer engineering skills. Remember to consider the feasibility, timeline, and resources available when selecting your project idea." (Anthropic, 2024)

Section 2a: ChatGPT Bike Helmet Regulations Results

The model used for the ChatGPT test was ChatGPT 4.0. Given that this is the premium model, it has the ability to search the web, and can provide the sources from information gathered. As a result, the results are not limited by a "knowledge cutoff" which is the date at which an LLM stops receiving training data. ChatGPT's results to this query are shown:

"In Florida, bike helmet regulations are primarily focused on riders under the age of 16. Here's a summary of the key points:

1. Mandatory Helmet Use for Young Riders: Florida law requires all cyclists and passengers under the age of 16 to wear helmets. These helmets must be properly fitted and securely fastened, and they need to meet federal safety standards as defined under the Code of Federal Regulations (16 C.F.R. Part 1203) (Flanagan & Bodenheimer Law).
2. Federal Safety Standards**:** Helmets should have a sticker from the Consumer Product Safety Commission (CPSC), ensuring they meet federal safety standards. This is an important criterion when purchasing a helmet (Big Momma's Bicycles).
3. No Statewide Adult Helmet Law**:** Florida does not mandate helmet use for individuals over the age of 16. However, it's highly recommended for all riders to wear helmets for safety (Big Momma's Bicycles).
4. Local Ordinances: Some local areas might have stricter laws. For example, Miami-Dade County requires helmets for riders under 18, and some places like the City of Miami Beach mandate helmets for all riders on certain busy streets (CycleBlissful).
5. Legal and Safety Implications: Not wearing a helmet can affect legal proceedings and liability determinations in the event of an accident, especially if the rider is under 16. Courts may consider helmet use when assessing liability and damages (No-Fee Personal Injury Lawyer in Tampa).

Understanding and complying with these helmet laws is crucial for safety and can also impact legal outcomes in case of accidents. For specific local regulations, it's a good idea to check with local authorities or legal experts in your area." (OpenAI, 2024)

Section 2b: Claude Bike Helmet Regulations Results

The model used for the Claude test was Claude 3 Opus. Claude does not have the ability to access the internet, so the information given in the output is limited by Claude's knowledge cutoff of August 2023. Shown below are the query results for Claude:

"In the United States, bicycle helmets are subject to the U.S. Consumer Product Safety Commission (CPSC) federal safety standard (16 C.F.R. part 1203). This standard establishes minimum performance requirements for bicycle helmets, including impact attenuation, retention system strength, and labeling.

Key requirements of the CPSC standard include:

1. Impact protection: Helmets must reduce the force of impact and prevent head penetration.
2. Peripheral vision: Helmets must allow a field of vision of at least 105° to each side.
3. Retention system: Chin straps must be strong enough to prevent the helmet from coming off in a crash.
4. Labeling: Helmets must be labeled with a CPSC sticker and instructions for proper fit and use.

As for the state of Florida, there is no state-wide law requiring the use of bicycle helmets. However, some counties and municipalities in Florida have local ordinances that mandate helmet use for certain age groups, typically for children under 16.

While Florida does not have additional state-specific regulations for bicycle helmet safety, helmets sold in the state must comply with the federal CPSC standard. It is always recommended to wear a properly fitted, CPSC-certified helmet while riding a bicycle to reduce the risk of head injuries in the event of a crash." (Anthropic, 2024).

Section 3a: ChatGPT Technical Overview Results

"Creating a smart bike helmet with intelligent lighting, safety features, and mobile app connectivity involves integrating various technologies into a coherent system. Below is a high-level overview of the key components and technologies needed, followed by a description of how they would work together in a system architecture:

Key Components and Technologies

<u>Microcontroller/Processor</u>: Acts as the central processing unit of the helmet, managing inputs from sensors and controlling outputs like lighting and data communication.

<u>Sensors:</u>

- *Accelerometer and Gyroscope:* Detect changes in position and motion to provide crash detection and stability control.
- *Ambient Light Sensor:* Adjusts the brightness of the helmet's lights based on surrounding lighting conditions.
- *Proximity Sensor:* Detects nearby objects or vehicles, potentially alerting the rider to dangers.

<u>Lighting System:</u>

- *LEDs:* Integrated into the helmet for visibility. Includes front, rear, and possibly turn signal LEDs that can be controlled for brightness and activation.
- *Control Circuitry:* Manages power and operation of LEDs based on sensor inputs and user settings.

Connectivity Modules:

- *Bluetooth/Wi-Fi:* Enables data exchange between the helmet and a smartphone app.
- *GPS Module:* Allows real-time location tracking and route navigation.

Power Supply:

- *Rechargeable Battery:* Powers all electronic components.
- *Power Management System*: Ensures efficient energy usage and battery health.

Mobile App (software):

- *User Interface:* Allows users to customize settings (e.g., lighting preferences, alert types).
- *Data Analytics:* Monitors and records trip data, system alerts, and potentially the rider's health metrics.
- *Navigation and Communication Tools:* Provides route guidance and emergency communication features.

Safety Features:

- *Impact Detection System:* Utilizes accelerometer and gyroscope data to detect falls or collisions and can trigger emergency alerts.
- *Emergency Response System:* Automatically sends location and alert messages to emergency contacts or services when a severe impact is detected.

System Architecture and Interaction

Sensor Input Processing:

- Sensors constantly gather data about the rider's environment and motion.
- The microcontroller processes this data to make real-time decisions, such as activating lighting or sending alerts.

Lighting Control:

- Ambient light and proximity sensors input data to the microcontroller, which adjusts the LED lighting according to environmental conditions and traffic proximity.

Communication and Data Handling:

- The microcontroller connects to the mobile app via Bluetooth/Wi-Fi. It sends data to the app, such as travel logs, sensor data, and alerts.
- The app can send commands back to the helmet, allowing the user to customize settings or request GPS navigation.

Safety and Emergency Protocols:

- On detecting a potential crash via the impact detection system, the helmet's microcontroller activates the emergency response system.
- GPS coordinates and a distress signal are sent through the mobile app to predefined emergency contacts or services.

Power Management:

- The power management system monitors battery status, managing charging and discharging to maximize battery life.
- It ensures that critical components like lighting and communication modules remain powered during rides.
- This architecture relies on seamless integration and real-time processing capabilities to ensure both the functionality and safety of the helmet. The microcontroller is the core, coordinating between different hardware components and the software application to deliver a responsive and intuitive user experience. " (OpenAI, 2024)

Section 3b: Claude Technical Overview Results

"Here is a high-level system architecture for a smart bike helmet with intelligent lighting, safety features, and mobile app connectivity:

Key Components:

1. Helmet Shell: A durable, impact-resistant outer shell to protect the rider's head.

2. Embedded Sensors:

- Accelerometer and gyroscope to detect crashes, impacts, and sudden movements
- Proximity sensors to detect approaching vehicles or objects
- Ambient light sensor to adjust lighting based on environment

3. Intelligent Lighting System:

- Bright, energy-efficient LED lights (headlights, taillights, turn signals)
- Automatic light adjustment based on ambient light levels
- Brake lights that illuminate when deceleration is detected
- Turn signals activated by buttons on the helmet or via the mobile app

4. Microcontroller: A small, low-power microcontroller to process sensor data, control lighting, and communicate with the mobile app (e.g., Arduino, Raspberry Pi).

5. Battery: A rechargeable lithium-ion battery to power the helmet's electronics.

6. Wireless Connectivity: Bluetooth Low Energy (BLE) or Wi-Fi module for connecting to the mobile app and sharing data.

7. Mobile App: A companion smartphone app for helmet configuration, ride tracking, emergency contact alerts, and social features.

8. Cloud Backend: A cloud-based server for storing ride data, processing analytics, and supporting app features.

System Architecture:

- The embedded sensors continuously monitor the rider's movement and environment, sending data to the microcontroller.
- The microcontroller processes the sensor data and controls the intelligent lighting system based on predefined algorithms and user settings.
- In the event of a crash or emergency, the microcontroller can send an alert to the mobile app via the wireless connection.
- The mobile app allows the user to customize helmet settings, view ride statistics, share data with friends, and set up emergency contacts.
- Ride data is synced from the helmet to the mobile app, and then to the cloud backend for storage and analysis.
- The cloud backend provides additional features such as ride history, leaderboards, and community-driven safety insights.

Data Flow:

Sensors -> Microcontroller -> Wireless Module -> Mobile App -> Cloud Backend

This architecture enables a smart helmet that enhances rider safety through intelligent lighting, crash detection, and connected features. The combination of embedded sensors, responsive lighting, and mobile app integration creates a comprehensive system for improving visibility, awareness, and emergency response for cyclists." (Anthropic, 2024)

Appendix D – AI Disclosure Statement

This paper includes content reviewed and edited with the assistance of AI tools, including ChatGPT, Claude, and Copilot. These AI technologies were utilized to enhance the clarity, coherence, and overall quality of the writing. While AI provided valuable input, all final decisions and reviews were conducted by the authors to ensure accuracy and alignment with the intended project objectives and conclusions.