HelmetIQ

A Smart Bike Helmet Solution

Christopher Bowerfind, Hector Agosta, Ethan Hymans, Ana-Victoria Elias

University of Central Florida School of Electrical Engineering and Computer Science, Orlando, FL

***Abstract* — An ECE project containing a custom PCB, mobile application, and embedded software. HelmetIQ – A Smart Bike Helmet Solution is a unique take on enhancing safety and visibility of cyclists. An advanced integration of sensors and drivers including a gyroscope, accelerometer, capacitive touch sensor, light sensor, haptic motors, led drivers, and a Bluetooth module enable the project to function. This project takes an existing bike helmet and seamlessly integrates these sensors within it. An MCU takes in these sensors readings and makes adjustments like dynamic lighting, blinker control, and most importantly collision detection. When the MCU algorithm detects a collision there is a flag sent to our mobile application that allows the rider to verify if they have been in a collision. If a collision has been deemed to occur the application sends an emergency SMS to the user's emergency contacts. This project adds an extra layer of protection and peace of mind to any cyclist using the helmet system.**

I. Introduction

Cycling has no doubt been an incredibly popular method of transportation over the years. However, when riding on roads and in other conditions, the cyclist has been at the mercy of drivers and other elements around them. The basic bike helmet serves one purpose and that is to help protect the riders head in case of a collision. We thought, what can we do to help decrease the odds of a collision entirely, and what can we do to immediately help if an unfortunate collision was to occur. This is where the idea of HelmetIQ spawned from.

The best way to promote the use of HelmetIQ and make it so that it was a seamless integration into something cyclists already had with them on rides, was to build the system directly into the bike helmet. We didn't want to create a system that required cyclists to have to carry any extra gear or any complicated devices. This is why we elected to implement the HelmetIQ system directly onto the bike helmet itself. We have designed the system so that there is minimal weight change and so that the system is intuitive to use with the helmet on your head. Additionally, we determined that most cyclists today carry their phone with them to either listen to music, give directions, etc. Because of this we decided that adding a mobile application component to HelmetIQ made sense. This application can have more advanced control over the system as well as create a place where everything the rider would need like maps, light control, and ride information would all be in one place.

II. System Components

In order to introduce the system functionality, we must start by defining the system components. These components were selected after a rigorous comparison with other technologies and other specific parts. Each component has served the correct purpose for HelmetIQ, and these components will be defined in this section.

A. Microcontroller Unit (MCU)

The center of the PCB and heart of the on-helmet aspect of HelmetIQ is the ESP-WROOM-32. This MCU has a built in Bluetooth module that works over Bluetooth 4.0 which was a critical aspect of communication to our mobile app. The MCU runs on an adjustable clock from 80MHz-240MHz and has 520 KiB of SRAM [1]. It takes a 3.3V input power and has 34 physical GPIO pins. Some of these pins are default SDA, SCL lines which are needed for sensor communication over I^2C.

B. Light Emitting Diodes (LEDs)

The HelmetIQ system uses multiple LEDs to enhance rider visibility and communicate essential signals to surrounding traffic. The lighting system includes a headlight, brake light, and turn signal lights, all powered and controlled efficiently to minimize power consumption and maximize brightness.

Headlight: Cob 3.3V White Light LED (400mA) The headlight uses a COB (Chip-on-Board) LED, which packs many LED chips together for high light output in a small space. This is crucial for headlights, as they need to be both bright and compact. It operates at 3.3V and draws 400mA, indicating a powerful light source.

Brake Light: Red Light 2.2V Cree LED (350mA) To enhance visibility during braking events, the rear lighting system employs two Cree LEDs configured in series. This design choice necessitates a 4.4V power source (2.2V per LED) while maintaining a consistent current of 350mA through both LEDs. The resultant increase in luminous intensity ensures a prominent red signal, effectively communicating the vehicle's deceleration to trailing traffic.

Turn Signal: White Light 2.9V Cree LED (350mA)
The turn signal system employs a single Cree LED operating at 2.9V and 350mA to effectively communicate directional intent. This configuration provides sufficient illumination to ensure clear visibility without excessive power consumption, striking a balance between signaling efficacy and energy efficiency.

C. Haptic Drivers

The HelmetIQ system incorporates two DRV2605L haptic drivers, each positioned near a touch sensor on either side of the helmet. These drivers are connected to a TCA9548APWR I^2C multiplexer, which provides separate communication channels for each haptic driver via the I^2C bus. This setup allows the system to control the haptic feedback on each side independently, delivering targeted tactile alerts when the respective blinker is activated.

When a touch sensor triggers a blinker, the corresponding haptic driver delivers a vibration to provide immediate feedback to the rider, confirming the blinker is active without requiring visual confirmation. This tactile response allows cyclists to remain focused on their surroundings while staying aware of system status. The DRV2605L drivers can generate a range of vibration patterns, ensuring clear and intuitive feedback from the ERM motors.

The ERM motors were selected for their effectiveness in delivering distinct vibration patterns. The DRV2605L driver provides programmability, allowing us to fine-tune vibration intensities and change patterns based on user needs.

D. Light Sensor

The light sensor we elected to use was a VEML 7700 [2]. This is a light sensor with 16-bit resolution and includes a photodiode, low noise amplifier, and 16-bit ADC. This light sensor communicates over the I^2C protocol which works well with our chosen ESP-WROOM-32. This sensor allows us to measure Lux value, Ambient Light Value (ALV), and white light. For our purposes we are only using the Lux value because it is a more standard measurement. However, ALV would have also been a possibility for our system. The light sensor detects environmental lighting conditions and determines when to trigger the headlights/taillights for visibility in the dark.

E. Accelerometer/Gyroscope

The ICM-42670P is a 6-axis motion sensor that combines a 3-axis accelerometer and a 3-axis gyroscope [3], communicating with the ESP-WROOM-32 via I^2C. In the HelmetIQ system, this sensor plays a dual role. It detects deceleration or acceleration, triggering the brake light to signal movement and enhance visibility to nearby drivers. Additionally, the ICM-42670P serves as the primary sensor for collision detection. Upon sensing a high-force impact, the ESP-WROOM-32 activates the hazard lights to alert others of a potential emergency and sends a Bluetooth notification to the mobile app. This capability is crucial for improving cyclist safety by proactively preventing accidents through increased visibility and providing immediate alerts in case of a collision.

F. Capacitive Touch Sensors

HelmetIQ features two TTP223 Capacitive Touch Sensors, one placed on each side of the helmet, to control distinct turn signal blinkers. These sensors detect touch input from the cyclist, allowing for quick and easy activation of the blinkers with a simple tap on the respective side of the helmet. Each sensor is connected to the ESP-WROOM-32 MCU via a dedicated GPIO pin, ensuring independent control of the left and right blinkers. The TTP223 operates with low power consumption, making it ideal for battery-powered applications on the helmet. The sensors provide an intuitive, hands-free way to signal turns, significantly improving safety and ease of use during rides without requiring any additional external devices.

G. I^2C Multiplexer

To manage multiple I^2C devices within the HelmetIQ system, we implemented the TCA9548APWR I^2C multiplexer. This component allows for the seamless integration of various I^2C devices, such as the DRV2605L haptic drivers and VEML7700 light sensor, by providing up to eight separate communication channels. Each I^2C device is assigned its own channel, ensuring that the signals remain isolated and preventing address conflicts on the bus.

In HelmetIQ, the TCA9548APWR is used to manage the communication between the ESP-WROOM-32 MCU and the haptic drivers on either side of the helmet. By routing the I^2C signals through the multiplexer, the system can independently control the haptic feedback for each blinker, ensuring clear and responsive feedback when the blinkers are activated. The TCA9548APWR also optimizes the overall system performance by allowing multiple I^2C devices to communicate simultaneously without interference.

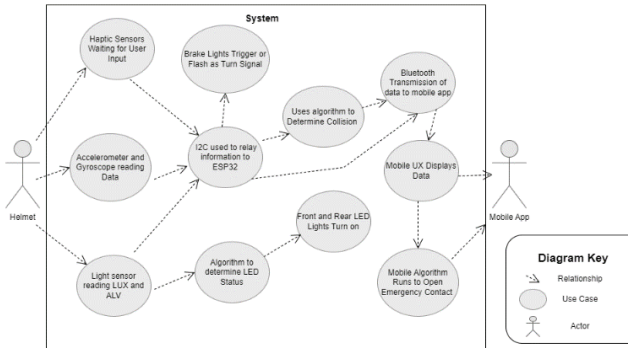
The HelmetIQ system integrates all components directly onto a custom PCB housed within the helmet, ensuring a compact and seamless design. The ESP-WROOM-32 MCU serves as the central controller, interfacing with the ICM-427670P accelerometer, VEML7700 light sensor, and TTP223 touch sensors, all of which communicate via I^2C, facilitated by the TCA9548APWR multiplexer. Haptic feedback is provided by two DRV2605L haptic drivers, positioned near the touch sensors. All components work together to provide a streamlined, responsive system. The code that controls the system's functionality is detailed in subsequent sections.

III. System Concept

To illustrate the operation of HelmetIQ as a complete system, an activity diagram and hardware block diagram are provided. The activity diagram outlines the sequence of

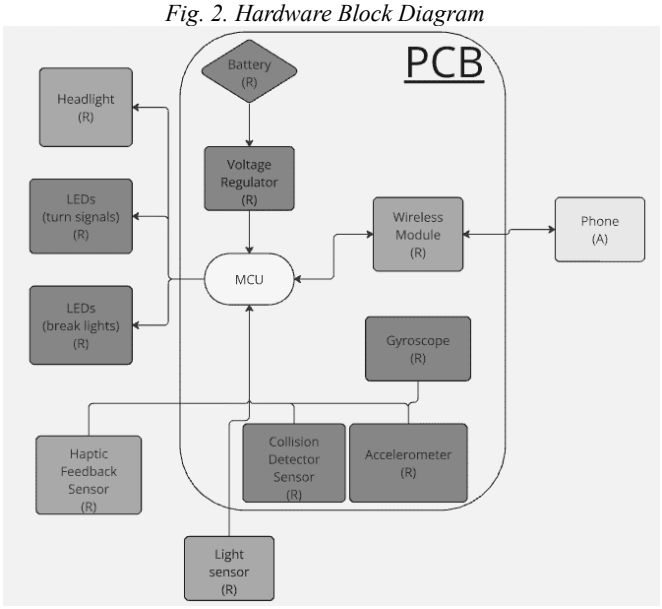
actions, from sensor input to system responses such as light control and haptic feedback. The hardware block diagram illustrates the systems relationship between components. These diagrams present a clear and concise view of HelmetIQ's functional structure and how each component works together.

Fig. 1. System Activity Diagram



The diagram shows how various components interact within the system, including the helmet's sensors and the mobile app. Key activities such as haptic sensors awaiting user input, the accelerometer and gyroscope reading data, and the light sensor detecting environmental conditions are all represented.

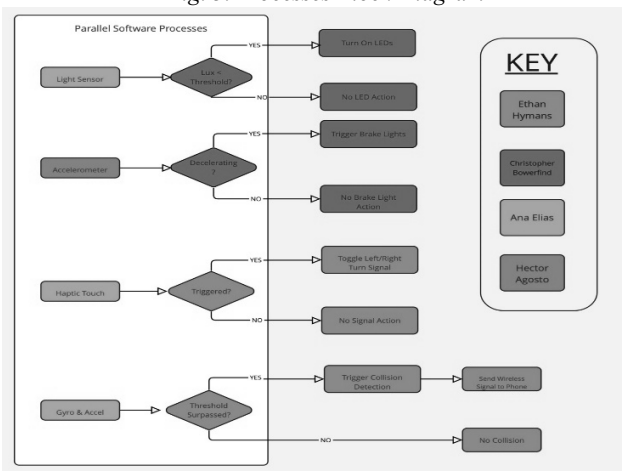
Fig. 2. Hardware Block Diagram



A. System Hardware Concept

With the system flow previously described, we can now detail the hardware interface that brings the HelmetIQ system to life. The major components of our hardware setup are organized to optimize data processing and real-time responsiveness. Figure 3 presents a block diagram that highlights the relationships and I/O flow between the components of our system.

Fig. 3. Processes Block Diagram



The major data buses and communication interfaces utilized in HelmetIQ are as follows:

I2C Bus: The primary communication protocol used for several system sensors and drivers. This includes the ICM42760P accelerometer/gyroscope, VEML7700 ambient light sensor, and DRV2605L haptic drivers. The I2C bus enables efficient data transfer between these components and the ESP-WROOM-32 microcontroller

GPIO Connections: The ESP-WROOM-32 microcontroller manages input and output operations using dedicated GPIO pins. These pins are used for direct connections to the capacitive touch sensors, LED drivers, and other digital interfaces, providing reliable signal handling and control.

Voltage Regulation Path: The TPS62152 buck converter efficiently steps down the voltage from the 7.4V battery pack to 3.3V, supplying consistent power to the entire system. This regulated power path ensures that all components operate within their specified voltage ranges, reducing power loss and heat generation.

Bluetooth Communication: The ESP-WROOM-32 microcontroller features an integrated Bluetooth module, which handles communication with the mobile application. This allows the HelmetIQ system to send and receive data wirelessly, allowing for features like collision alerts and light control from the app. The communication to the app happens over Bluetooth 4.0 as a serial connection. This allows for serial terminal monitoring during debugging.

IV. Hardware Detail

Each of the major system components outlined in Section II, System Components will now be explained in technical detail

A. Microcontroller and Communication Module

The ESP-WROOM-32 allowed for multiple sets of GPIO pins to be dedicated to I^2C communication. This was a useful feature but one that we ended up not using due to the presence of the Mux. The main issue was that even while using the Mux not all I^2C devices were connected into its channels. Due to this there was some conflict when it came to determining which I^2C device had control of the line at a certain time. This led to some firmware debugging and issues with the ESP-WROOM-32 randomly restarting whenever there was a conflict over this I^2C bus. There were no issues with the Bluetooth module, and this was a smooth process during testing and implementation into the full system. The integrated module into the chip removed the need for us to have a dedicated Bluetooth module on the PCB.

B. Voltage Regulation System

The TPS62152 [4] is a step-down DC-DC converter specifically designed for high efficiency and small footprint applications. It features an adjustable output voltage and can maintain high efficiency even at light loads, making it suitable for battery-powered devices like our helmet. The regulator's high efficiency minimizes heat generation, an important consideration given the space and thermal constraints of our design.

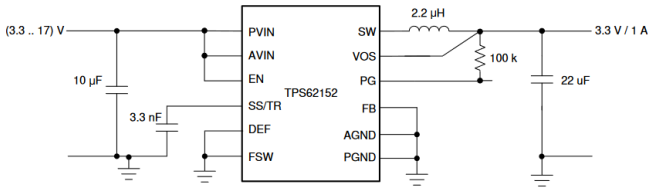
Key specifications of the TPSS62152 include:

(1) Input Voltage Range: 3V to 17V
(2) Output Voltage: Fixed at 3.3V for our design
(3) Maximum Output Current: 1A
(4) Efficiency: Up to 90%, which reduces power loss and heat

The voltage regulator is responsible for stepping down the voltage from the two 3.7V lithium-ion batteries (configured in series to provide 7.4V) to 3.3V, which is the operating voltage of all the components and sensors. To minimize power loss and improve system efficiency, the regulator is placed on a separate PCB, reducing the risk of noise affecting the sensitive components on the main PCB.

The schematic, shown in Figure 4, serves as the basis for our implementation, providing a stable 3.3V output at up to 1A.

Fig. 4. 3.3V, 1A- Power Supply from Texas Instrument



The key components and connections are described as follows:

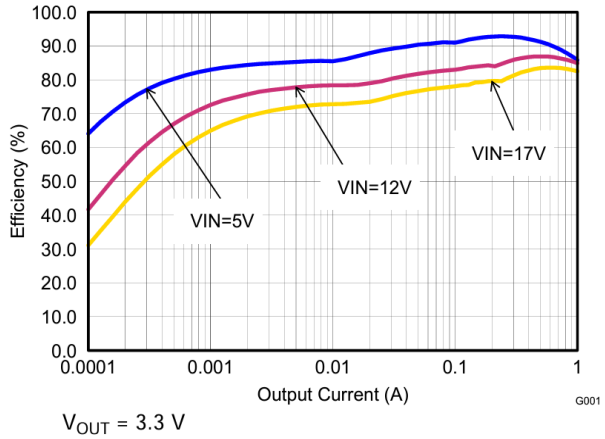
- Input Capacitors (10 µF and 3.3 nF): These capacitors are placed at the input to filter high-frequency noise and ensure stable operation of the voltage regulator. The 10 µF capacitor helps smooth out voltage fluctuations from the 7.4V battery input, while the 3.3 nF capacitor is used for additional noise suppression.
- Inductor (2.2 µH): The 2.2 µH inductor is a crucial component in the buck converter topology, storing energy and smoothing out the current supplied to the load. It helps to maintain the desired voltage level and reduce output ripple.
- Output Capacitor (22 µF): This capacitor is connected at the output to further filter and stabilize the 3.3V output. It ensures low output voltage ripple and provides a steady power supply to the connected components.
- Feedback Resistor (100 kΩ): The 100 kΩ feedback resistor sets the output voltage of the regulator. By adjusting the feedback voltage, the regulator maintains the desired 3.3V output.
- Enable (EN) and Soft Start/Tracking (SS/TR) Pins: The EN pin is used to enable or disable the regulator. The SS/TR pin allows for soft start, which gradually ramps up the output voltage to prevent inrush currents that could potentially damage sensitive components.

The efficiency of the TPS62152 buck converter was a critical factor in our design, as it determines how effectively we can step down the voltage from the two 3.7V lithium-ion batteries (in series, providing 7.4V) to the necessary 3.3V output. Minimizing power loss was essential to extend battery life and reduce heat generation, given the space and thermal constraints of the helmet.

Figure 5 demonstrates the efficiency performance of the TPS62152 at various input voltages. Our system operates with a 7.4V input, placing us between the 5V and 12V efficiency curves shown. Importantly, our

design draws up to 1A of current, where the efficiency approaches 90%. This high efficiency ensures that energy loss is minimized, maximizing the available power for critical system components such as the MCU, sensors, and lighting.

Fig. 5. Voltage Regulator Efficiency, Texas Instrument



V_{OUT} = 3.3 V

C. Turn Signal Implementation

Our smart helmet incorporates a turn signal system designed to improve rider safety and convenience. The system integrates capacitive touch sensors for user input and a haptic feedback mechanism to provide immediate confirmation of signal activation.

The capacitive touch sensors serve as the primary interface for activating the helmet's turn signals. These sensors are embedded on the helmet's surface, allowing the rider to engage the turn signals effortlessly without searching for physical buttons. The sensors detect the presence of a finger by measuring changes in capacitance. When a user touches a designated area, the microcontroller registers the change and triggers the corresponding turn signals. Since all the sensors share the same I^2C address, we implemented a multiplexer, which enables the microcontroller to selectively communicate with each sensor. This approach ensures smooth operation and reliable detection.

The haptic feedback system works simultaneously with the capacitive touch sensors by providing physical confirmation of signal activation. When a touch sensor is activated, the haptic system delivers a short vibration. This feedback ensures the rider is aware that the turn signal has been engaged without needing to look at the indicator lights.

We used the DRV2605L haptic driver to deliver tactile feedback, providing the rider with physical confirmation of turn signal activation. The driver supports both ERM (Eccentric Rotating Mass) and LRA (Linear Resonance Actuator) motors, offering flexible and efficient haptic responses. The DRV2605L features a proprietary smart-loop architecture that enhances actuator performance. This architecture includes automatic overdrive, braking, resonance tracking (for LRA), and actuator diagnostics. By leveraging these features, the haptic feedback system maintains consistent vibration intensity and minimizes response times. These drivers work with the ERM (Eccentric Rotating Mass) motors. The ERM motors create vibrations by rotating an off-center mass, producing a centrifugal force that translates into tactile feedback. The speed of rotation, and the intensity of the vibration, is controlled by adjusting the input voltage or using pulse-width modulation (PWM) signals.

The DRV2605L driver efficiently drives the ERM motors using the smart-loop architecture, which optimizes the motor's performance. Features such as automatic overdrive and braking are particularly beneficial, reducing the startup and braking times for the ERM motors.

Table 1. Pin Configuration for Haptic Feedback System

Component	Pin	Description
Microcontroller	**GPIO Pins**	Connected to capacitive touch sensors for turn signal detection and activation.
	I2C Pins (SCL, SDA)	Used to communicate with the DRV2605L haptic driver for configuring haptic feedback.
DRV2605L Haptic Driver	**IN/TRIG Pin**	Trigger input for real-time playback, activating specific haptic effects.
	I2C Pins (SCL, SDA)	Communication with the microcontroller
	VDD and GND Pins	Operating Voltage
	OUT+ and OUT- Pins	Differential outputs driving the ERM motors
ERM Motors	**OUT+ and OUT- Pins**	Connected to the DRV2605L, receiving vibration signals for turn signal feedback.

D. Headlight and Brake Light Implementation

(1) Ambient Light Detection

To manage the automatic activation of the headlights, we utilized the VEML7700 ambient light sensor. The VEML7700 features a 16-bit resolution and communicates with the ESP-WROOM-32 microcontroller via the I2C protocol. It integrates a photodiode, a low noise amplifier, and a 16-bit ADC, making it highly efficient in low-light detection. For our system, the VEML7700 measures the Lux value of the surrounding environment and triggers the headlights when ambient light falls below a pre-defined threshold. This automated adjustment ensures that the rider remains visible in low-light conditions, such as at night or when passing through tunnels.

(2) Headlight Implementation

The MCP1643 boost converter was selected to power the headlight. The MCP1643 efficiently steps up the voltage from the 3.3V battery output to drive high-power LEDs for the headlight, ensuring consistent illumination. This converter's compact and lightweight design is ideal for our helmet application, which has space and weight constraints. The integration of the headlight system enhances rider safety by providing forward illumination and making the cyclist more visible to others in low-visibility scenarios.

(3) Brake Light Activation

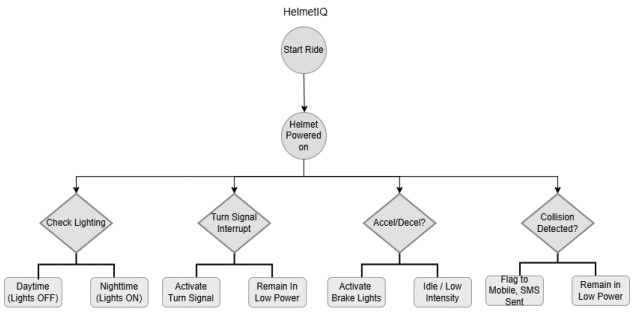
For the brake light, we utilized the AP3019AKTR-G1 LED driver, which incorporates a boost regulator to deliver constant current to the LEDs. This ensures stable and consistent brightness even when driving two red LEDs in series, which require a higher voltage than the system provides. The driver's integrated boost functionality eliminates the need for a separate regulator, simplifying the circuit design.

The brake light is triggered by an ICM-42670P accelerometer/gyroscope sensor. When this sensor detects significant deceleration, it signals the ESP-WROOM-32 microcontroller. The microcontroller then activates the AP3019AKTR-G1 driver, illuminating the LEDs and signaling to drivers and pedestrians that the rider is slowing down. This real-time response is crucial for preventing collisions and improving overall road safety.

VI. MCU FIRMWARE

The software on the ESP-WROOM32 is built upon the Free RTOS library. This library allows for us to run concurrent tasks on the ESP and allows our system to respond in real-time without delays. Below is the Activity Diagram that shows the parallel processes occurring on the ESP32 during HelmetIQ operation.

Fig. 6. ESP32 Firmware Activity Diagram



A breakdown of the processes is as follows. First, the VEML 7700 is reading a Lux value, from this Lux value a threshold has been set. When the Lux value falls below this defined threshold, the headlight and taillight automatically come on for a "night" mode. Secondly, the capacitive touch sensors trigger a hardware interrupt which then activates a haptic motor and LED blinker. Each touch sensor, haptic motor, and blinker LED are specific for each side of the helmet. Third, The ICM42670P accelerometer is constantly reading accelerometer values in all directions. Then the delta in all directions is taken to compute the change in between the values between readings. If the change is greater than a threshold in any direction the collision flag will raise. Lastly, once the collision flag has been raised, an int flag is sent to our mobile app to signal a collision has occurred. This is a very high-level overview of the software, lower-level details like the mux initialization, haptic drivers' selection, and sensor initialization are critical to the system but not detailed here.

A. System Firmware

The Firmware for our ESP32 is written entirely in C++. This is mostly due to the nature of the Free RTOS library that our processes were built upon. The library was in C++, so it clearly dictated our path regarding firmware language. Additionally, we used the Arduino IDE which provided an easy Serial Monitor for debugging and verifying testbenches for each component. Since we used Free RTOS we did not have anything running in our main loop instead everything was defined as a task. An example of such task declaration is below.

Fig. 7. ESP32 Firmware Task Declaration

```
xTaskCreatePinnedToCore(handleBlinkers,
"BlinkerTask", 4096, NULL, 1,
&blinkerTaskHandle, 0);
```

Aside from the tasks, the other major component was the Bluetooth declaration and integration into the ESP32. The ESP-WROOM-32 has a built in Bluetooth module on the chip running Bluetooth version 4.0. This is not a Bluetooth LE (low energy) version, and this was considered when determining total power. A serial Bluetooth connection is

used to communicate with our mobile app, and this was chosen because of its simplicity. All of the computation to detect a collision is done on the ESP and so the only communication over Bluetooth needed to be a heartbeat (to maintain connection) and a flag (int) to signal that a collision had occurred. Additionally, our mobile development advanced to where we could control our lighting with a toggle from the mobile app and so this requires the ESP to receive a flag and assign that flag to modify a global variable. This is a more complicated two-way Bluetooth data stream than we had planned for, but we believe it significantly increases the functionality of HelmetIQ. Below is an example of the Bluetooth data send and receive over serial.

Fig. 8. ESP32 Bluetooth Data Send

```
// Check if any delta exceeds the collision
threshold
    if (deltaX > collisionThreshold ||
deltaY > collisionThreshold || deltaZ >
collisionThreshold) {
      Serial.println("Collision
detected!");
      SerialBT.println("10");
      inCollision = true;
```

Fig. 9. ESP32 Bluetooth Data Receive

```
while (SerialBT.available()) {
        char c = SerialBT.read();
        if (c == '\n' || c == '\r') break; //
Terminate on newline or carriage return
        incomingData += c;
      }

      // Process the received data
      if (incomingData.length() > 0) {
        Serial.print("Received Data: ");
        Serial.println(incomingData);
```

A final note about the MCU firmware is the use of mutex locks. This is a synchronization mechanism to ensure that all of these parallel processes do not cause crashes. Initially, the MCU firmware did not employ these locks and throughout most of the development these locks were not found to be needed. However, through deeper testing, we found that the volatility of the two-way Bluetooth data stream could cause the Bluetooth receive function to override everything else whenever data had been sent. This would cause the ESP to crash and the Bluetooth to disconnect. Using these locks, we were able to separate these tasks in a way they could not overlap increasing system stability.

VII. MOBILE SOFTWARE

The HelmetIQ mobile application was developed using Kotlin and Jetpack Compose for the Android platform, providing both a control interface for the helmet's features and an emergency response system. The application architecture follows a layered pattern, with distinct components managing specific system functionalities such as navigation, Bluetooth communication, and emergency response protocols.

The application's navigation system is built upon Navigation Compose, featuring a bottom navigation bar that provides access to two main screens: Home and Emergency Contacts. This navigation framework ensures smooth transitions between different sections while maintaining state consistency. The home screen serves as the primary interface, displaying critical information including the helmet's Bluetooth connection status, headlight controls, a mini map showing current location, and basic ride statistics.

Bluetooth communication is managed through a dedicated BluetoothManager class that handles device discovery, connection establishment, and data communication with the helmet's ESP32 module. The BluetoothViewModel maintains connection state and processes incoming signals using Kotlin Flow, ensuring reliable real-time updates. When the system receives a collision detection signal from the ESP32, it triggers the emergency response protocol regardless of which screen the user is currently viewing.

The emergency response system operates across all application screens through a state management system build with Kotlin Coroutines and Flow. Upon detecting a potential collision, the application displays a dialog with a 10-second countdown, during which the user can either confirm they are safe or allow the timer to expire to trigger emergency protocols. If the timer expires or the emergency is confirmed, the system automatically sends SMS alerts to pre-configured emergency contacts with the user's location and status information.

Location services are implemented using the Google Maps SDK for Android, providing real-time location tracking and visualization. This functionality is critical for the emergency response system, as it enables accurate location sharing in emergency SMS messages. The application maintains a persistent store of emergency contacts and user preferences through the PrefsHelper utility, ensuring critical information is readily available when needed.

State management throughout the application is handled using Kotlin's StateFlow, providing reactive updates and consistent behavior across configuration changes and process termination. This approach ensures reliable performance for critical safety features while maintaining a responsive user interface. The state management system tracks various aspects including Bluetooth connection status, emergency contact information, user preferences, collision detection state, and location updates.

The mobile application's architecture emphasizes modularity and reliability, particularly in its emergency response capabilities. By utilizing modern Android development practices and libraries, the application maintains a lightweight footprint while providing robust safety features. This design approach allows for straightforward maintenance and future feature additions while ensuring stable performance for critical emergency response functionality.

VIII. Board Design

The printed circuit board (PCB) design for our smart helmet project was critical in ensuring the overall performance, reliability, and safety of the embedded system. Several design strategies were implemented to optimize the layout, minimize electromagnetic interference (EMI), and simplify troubleshooting during testing and assembly.

A. Layer Configuration

The PCB was designed as a two-layer board, with consideration given to signal integrity and power distribution. Both the top and bottom layers of the PCB feature a ground pour, which was implemented to reduce ground interference and mitigate EMI. The ground pour on both layers helps to create a more uniform return path for the signals, thereby minimizing noise and crosstalk between critical traces.

The top layer is dedicated to routing the signal and power traces for the microcontroller, capacitive touch sensors, haptic feedback driver, LED Drivers, and associated components. By keeping the signal routes as short and direct as possible, we reduced the risk of signal degradation. Sensitive signal lines were also routed away from power traces to prevent noise coupling, and decoupling capacitors were placed close to their designated component to ensure stable operation of the microcontroller and sensors.

The bottom layer was used for additional power and signal traces. This dual-layer approach helped improved EMI performance but also simplified the layout by reducing the number of vias and providing a stable grounding scheme throughout the board.

B. Component Placement and Separation

One significant design decision was to separate the voltage regulation circuitry into a dedicated PCB, distinct from the main PCB housing the microcontroller, sensors, and drivers. This separation was made to minimize heat generation near sensitive components and to reduce the potential for power supply noise interfering with signal integrity. The voltage regulator PCB, which houses the TPS62152 buck converter, is positioned for efficient power distribution to the main PCB. This setup ensures that the 3.3V output from the regulator is clean and stable, reducing the risk of voltage fluctuations.

Another crucial aspect of the PCB layout was the strategic placement of the pads. Since the board is centrally located on the helmet, pad placement was carefully considered to correspond with the connection points of the various components. For instance, the headlight pads were placed towards the front, the brake light pads towards the back, and the turn signal pads on their respective sides. This deliberate pad placement minimized wire clutter on the helmet and eliminated the need to cross wires over or under the PCB, resulting in a tidier and more efficient design

C. Routing Strategy

One of the key considerations on the PCB design was managing trace widths. The small pitch of many ICs meant initial trace sizes were quite small. To avoid potential signal integrity issues and excessive resistance, trace widening was employed, strategically increasing the width of traces as they moved away from the ICs. Furthermore, power lines were given larger traces to adequately handle the current demands of the components and minimize voltage drops that could affect performance

ACKNOWLEDGEMENT

The authors would like to thank the reviews Dr. Kalpathy Sundram, Dr. Sonali Das, Dr. Ronald DeMara, and Mrs. Jacqualine Sullivan for their time in reviewing and supporting this project.

REFERENCES

[1] Techonics LTD. "ESP-WROOM-32 Development Board | 2.4GHz Dual-Mode WiFi + Bluetooth Dual Cores Microcontroller Processor With built-in Antenna." [Online]. Available: https://www.techonicsltd.com/product/esp-wroom-32-development-board-2-4ghz-dual-mode-wifi-bluetooth-dual-cores-microcontroller-processor-with-built-in-antenna/. [Accessed: Nov. 8, 2024].

[2] Vishay Semiconductors. (2016). *VEML7700: High Accuracy Ambient Light Sensor with I²C Interface.* [Datasheet]. Retrieved from https://www.vishay.com/docs/84286/veml7700.pdf

[3] TDK InvenSense. "ICM-42670-P." [Online]. Available: https://invensense.tdk.com/products/motion-tracking/6-axis/icm-42670-p/. [Accessed: Nov. 8, 2024].

[4] Texas Instruments, "TPS6215x 1-A, 17-V, Synchronous Step-Down Converter in 3x3 QFN Package," TPS62152 Datasheet, Nov. 2014 [Revised Dec. 2017]. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps62152.pdf