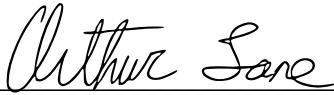


GONZAGA UNIVERSITY
School of Engineering and Applied Science
Center for Engineering Design and Entrepreneurship

Final Project Report
April 24, 2019

Smart Helmet
(Bike Safety Smart Sensor)


Prepared by:



Arthur Lane



Damon George



Robert Brajcich

Reviewed by:

Tim Ecklund
Faculty Project Advisor

TABLE OF CONTENTS

1 Executive Summary	2
2 Project Summary	3
3 Project Details	3
3.1 Project Goals and Objectives	3
3.2 Project Features	4
4 Research	7
4.1 Lithium Ion Batteries	7
4.2 DSRC Protocol Specification	7
4.3 FreeRTOS Usage	8
4.4 PCB Design	8
5 Engineering Work	8
5.1 Completed Tasks	8
5.2 Unfinished Tasks	10
6 Project Management	11
6.1 Schedule Review	11
6.2 Final Budget	11
6.3 Meeting Review	11
7 Analysis of Results	12
7.1 Analysis of Completed Work	12
7.3 Analysis of Incomplete Work	14
7.3 Final Product Improvements and Steps	15
8 Appendices	16
8.1 Abbreviations	16
8.2 Detailed Project Requirements	16
8.3 Total Budget Spend	18
8.4 PCB Bill of Materials	20
8.5 Core Software Specifications	21

1 Executive Summary

As emerging smart vehicle technologies enter our roadways, Vehicle-to-Everything (V2X) communication will improve the safety of road users. However, as cars and trucks undergo technological changes involving ample power and space for advanced electronics, cyclists may fall behind. In this study, we attempt to incorporate cyclists in the growing V2X ecosystem by developing a prototype, the 'Smart Helmet'. This product broadcasts positional and inertial data about the cyclist to nearby cars using the predefined Dedicated Short-Range Communications (DSRC) standard, and receives similar data from nearby cars to provide the cyclist with haptic and audible collision warnings from the helmet. We construct a custom printed circuit board for the system to easily incorporate into a bike helmet as a consumer product. Furthermore, we develop a traffic simulator program to easily test traffic scenarios with the product. Simulation results show that even with the limited processing power available, this product can adequately detect harmful collisions. Limited usability testing shows that the haptic and audible warnings could function well in a consumer product. However, the V2X communication and advanced GPS are still immature, and creating effective yet unobtrusive audio warnings requires much testing. Overall, this study proves that this V2X system could greatly increase cyclist safety and commonly feature on roadways in the next decade as smart vehicle technologies continue to mature.

2 Project Summary

Technology that increases the visibility of cyclists to vehicles and vehicles to cyclists benefits both the cyclist and vehicle communities. Blind corners, crowded intersections, driveways, and street parking present roadway threats to cyclists and vehicles alike. New technology focuses on the vehicle side of this dilemma, leaving the cyclist to rely on driver decision making. Introducing cyclists into the new technologies for vehicle communication increases road safety for both groups.

The bike safety smart sensor uses a communication system between vehicles and cyclists in order to provide an alert system for cyclists. Our prototype uses an unlicensed RF band for testing; however, the system design allows for configurability to the protocols of autonomous or semi-autonomous vehicles. Global Positioning System (GPS) data from both parties determines speed, direction, and location data which the units transmit over radio channels to others on the road. This method of communication does not rely on line of sight, making it useful in dense urban areas or other low visibility scenarios. Further, the cyclist's device includes audio equipment to notify them of nearby vehicle threats. This feature mitigates the risk of any vehicles not equipped with transmitters. The system will mount on a cyclist's helmet to make it easy for anyone to incorporate it along with their current gear.

3 Project Details

3.1 Project Goals and Objectives

The main objective of this project is to create a functional prototype of our design that mounts on a bicycle helmet. At minimum the prototype consists of a microcontroller-driven system determining its GPS location data, transmitting it to nearby vehicles, receiving positional data from these vehicles, and issuing audible warnings to the bicyclist based on any detected hazards. This final deliverable also encompasses the necessary software to implement these features, which runs on the onboard microcontroller (MCU). We will also have a PCB that incorporates all the system hardware on a single board as well as an accompanying Bill of Materials (BOM) for assembling the board.

This system needs a supporting software deliverable to simulate potentially hazardous traffic scenarios for testing purposes. The supporting software runs as a Java desktop application and includes a graphical interface featuring a top-down view of the simulated world. The computer running the software connects through Universal Serial Bus (USB) to a microcontroller which sends the spoofed vehicle data via a transceiver to the unit on the bicycle helmet.

We also included three PCB designs for testing the power supply system as well as integrating the different components into a smaller form factor. Though this involves a long lead time, due to manufacturing and assembly at the board houses, it serves as a good indication of

what a final product would look like. By going through the PCB design process, we hoped to get a sense of the cost and size of what the final product would look like.

The secondary objective involves our goal as students to learn more in the field of embedded design, PCB design, and software design, and to experience the typical steps involved in designing a prototype with standard engineering methodologies.

3.2 Project Features

The essential features of the final product include the GPS connection, Vehicle to Everything (V2X) communication, risk assessment, rechargeable batteries/power system, audio warnings, and traffic simulation software, as detailed in Table 8.2.1 of the Appendix. V2X communication and GPS connection form the basis for the idea of the project as the simulated V2X communication between cyclists and vehicles requires a GPS connection to determine speed, location and directional data. Radio transceivers at 915 MHz take the place of the actual V2X chips due to the high cost and low availability of the new V2X chips. These transceivers simulate V2X communication by following the standards of the protocol.

The core risk assessment software determines any dangers to the biker using the wirelessly received GPS data from nearby vehicles. The system then notifies the cyclist using audio warnings through speakers on the helmet. Short audio messages do not confuse or distract the biker, and the speaker's placement does not inhibit the cyclist's awareness of the nearby environment.

The product also features rechargeable batteries. Our design goal is to provide power to the device for at least 2-3 hours. We implemented the rechargeable battery feature through a custom power PCB (printed circuit board). To power all the electronic components in the design, the board must provide both 3.3V and 5V from two series Lithium Ion batteries. The battery recharging through a USB port became too difficult to test and verify, so charging requires removal of the batteries from the system. An off the shelf product serves as a backup plan in the event that the PCB's do not function as anticipated.

The high level design of the system is shown next in Figure 3.2.

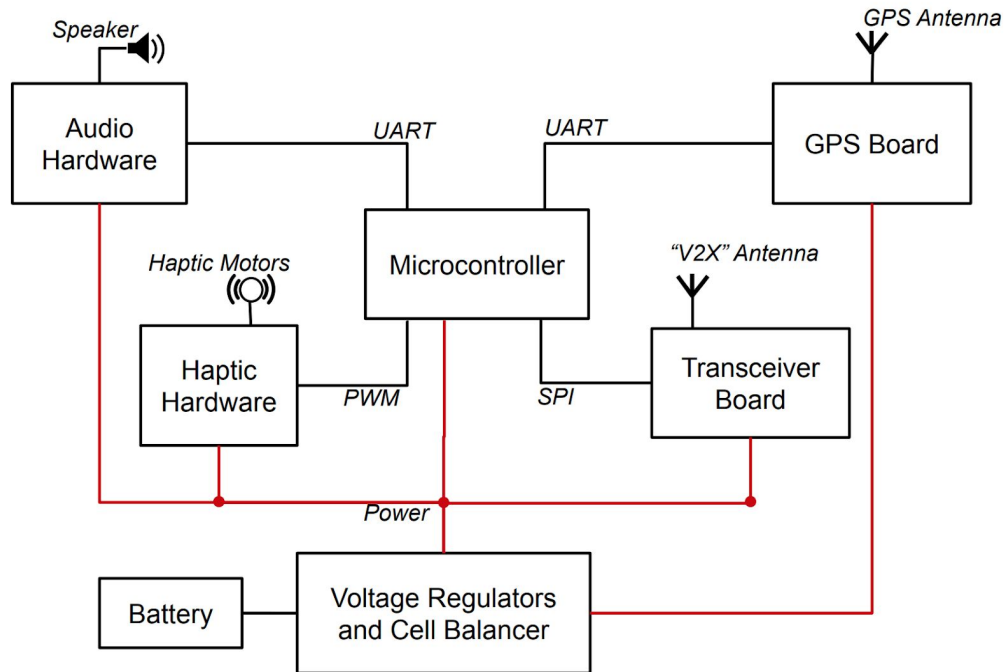


Figure 3.2.1: Full Design Block Diagram

The traffic simulation software provides evidence that the bike successfully receives data from nearby vehicles. This allows for testing the functionality of the device using real-world traffic scenarios. Furthermore, this software shows the full functionality of our finished prototype.

Part of this project includes additional features outlined in Table 8.2.2 of the Appendix. Haptic feedback was the highest priority additional feature because it provides an additional warning system for the biker using vibration motors inside the helmet to signal the direction to nearby vehicles. A full custom PCB is another stretch feature. We were able to complete the design, however, the lead time on getting the board fabricated and assembled was longer than anticipated. Due to this uncertainty, we also made a second iteration of the board which we believed would have a greater probability of complete success on the first trial.

3.3 Project Constraints

The most pressing constraint on the development of our project centers around hardware development. Due to cost issues for manufacturing the battery and power board, and delivery time for the microcontrollers, we had to perform additional work not originally planned. Further, many of the components and systems in our design required difficult manufacturing processes which required extended lead times and assembly by a skilled manufacturer.

Our original intention involved having the power PCB fabricated and tested before winter break. However, we fell short of this goal. We decided at the start of the second semester that

implementing the battery charging capability required time and money that we could not spare due to the design limitations involving the necessary components. To come up with a “midway” board that would provide the same power supply requirements, we completed a quick design spin that removed the battery charging circuit on the first board and added the haptic motor drivers. The goal of this board would be to use with our demonstration on Design Exposition Day to make our prototype function similarly to the final design.

We also had all the components soldered on at an assembly house with a pick and place machine to eliminate any errors from soldering the small components. We were under the impression that the house would be able to return the assembled boards back in a week, however, it ended up taking over three weeks which hurt our project timeline.

We ordered the final board with the full system integration at the end of March, and have learned that the manufacturing process takes a long time. This combined with the time it takes to get the board assembled with all the components pushed the timeline for the final product back. Such features as laser drill vias, 4 mil trace width and spacing, ENIG finish, BGA and QFN components, and component placement on both sides of the board, all contributed to the long lead times for manufacturing.

Overall, these steps helped work towards the final design as they provide insight into what the cost of producing this product at scale would be and would be the first step towards achieving a finished product.

Another constraint discovered during initial module testing is the bandwidth capability of the transceiver modules. An inverse relationship exists between the data rate through the air and the range of the radios. The transceivers achieve enough bandwidth to support between 5 and 10 vehicles to exchange their messages at short range (in the same room, for example). This configuration allows for traffic simulator test runs that include multiple vehicles, with the helmet close to the simulator’s transceiver. Data rates that only handle a couple of vehicles allow for longer ranges of multiple blocks in open areas. This constraint influences how testing occurs using the Traffic Simulator and real world scenarios.

4 Research

4.1 Lithium Ion Batteries

H. Weiss and S. Volgger, "Insular power supply battery manager with balancing and protection unit for Lithium-Ion batteries," *2014 12th International Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, Novosibirsk, 2014, pp. 789-792.

Using Lithium Ion batteries in electronics design requires several essential design considerations for a safe and effective final product. First, the popularity of Lithium Ion batteries comes from their high energy density and low discharge rate which allow for longer lasting power and battery life. These good qualities come with the trade-off of higher price and potential safety concerns with the batteries themselves if the batteries reach an overcharged or uncharged state. To mitigate these risks, incorporating Lithium Ion batteries into electronics design also involves the need for supporting circuitry. This includes a charge control circuit which monitors under and over voltage conditions as well as matches the source and load voltages. If the battery pack consists of two or more series batteries, then the design also requires a balancing circuit to maintain the same voltage across each series battery. These design features enable the safe use of Lithium Ion batteries and increase the life of the batteries. Our project involves the use of a cell balancing IC to regulate the two series batteries for our project. Also, a charger IC charges the two Lithium Ion batteries from a 5V USB input.

4.2 DSRC Protocol Specification

After receiving approval for the budget, we purchased access to the current Dedicated Short-Range Communication (DSRC) Protocol Specifications. The DSRC Protocol governs the format and rules of the wireless V2X messages used in this project. This project focuses on implementing the 'Personal Safety Message' from this protocol's message set, which is a periodic informational message designed for biker and pedestrian transmissions. This message includes all the data from the onboard sensors of the smart helmet. The design also implements the 'Basic Safety Message' which specifies motorized vehicle message transmissions. This allows the product to receive information from nearby vehicles to determine any threats to the biker, which is the main purpose of this product.

The DSRC Protocol also specifies how to securely encode, decode, and randomize parts of the messages sent from V2X devices. However, implementing this part of the protocol falls out of the scope of this project. This smart helmet prototype only uses standard wireless packet radios, and not the state-of-the-art V2X transceivers that would go in a final product. Using V2X transceivers allows for securely encoding and decoding the safety messages themselves, which is why message coding and decoding does not feature in our project.

4.3 FreeRTOS Usage

Part of this semester's research involved studying how to implement a real-time system to perform multiple tasks through various interfaces simultaneously without significant latency. We chose to solve this dilemma by using a Real Time Operating System (RTOS) to handle task scheduling. Using an RTOS does not introduce the significant overhead of a full operating system distribution such as Linux. While various RTOSes exist on the market, we selected FreeRTOS for our project due to its wide scale use, well-documented operation, and free availability. To prepare ourselves for implementing a real-time system using this framework, we thoroughly read through the manual and tutorials and developed the various software modules, or *tasks* necessary for implementing our project.

4.4 PCB Design

The third and final PCB design for this project involved incorporating all the subsystems of the final, working prototype on to a single board to best replicate what the final product would look like. Such a design involved many different complex features including buck converter, RF system, microcontroller, audio amplifier, motor driver, and general best PCB layout. To try and achieve the highest level of success with the first board, we obtained schematics and layouts for the breakout boards and studied them to determine ideal component placement, trace routing, via use, layer selection and manufacturing processes. We conducted research on best PCB design practices to make up for subjects not taught in our coursework. Intel provided great documentation on high speed design layout guidelines which were used for general layout guidelines while Texas Instruments featured a number of articles on routing for specific types of systems such as RF traces, antennas, buck converter, audio amplifiers and digital drivers.

5 Engineering Work

5.1 Completed Tasks

This semester we tested our first PCB iteration and completed the design for our second and third. Assembly for the second board finished with a week to spare; however, the final board took longer than anticipated for manufacturing. This involved developing a schematic and layout for two boards, determining and ordering components and maintaining a library of all the parts. To accomplish successful routing for the two boards, we studied the schematics of the breakout boards we used in prototyping and researched layout guidelines from trusted manufacturers.

In addition to the hardware work done on the schematic and layout of the power PCB, we also completed the core software, according to the Core Software Specifications located in the Appendix. These specifications detail the software running on the MCU to control all peripherals

and sensors, identify risks, and respond accordingly. Our design consists of 6 distinct tasks ordered by decreasing priority:

1. **Manager Task:** Manages all other tasks, providing the ability to sleep unused modules, during lost GPS signal or a stationary user.
2. **Audio Task:** Outputs audible collision warning messages to the cyclist through the onboard speaker.
3. **Transceiver Task:** Periodically broadcasts Personal Safety Messages from the cyclist, and also reads all incoming messages from nearby vehicles.
4. **GPS Task:** Reads the stream of positional data from the external GPS module.
5. **Haptic Task:** Periodically updates the haptic vibration motors on the cyclist's helmet to warn of nearby vehicles and any potential collisions.
6. **Core Task:** Receives data from the GPS and transceivers to calculate any potential risks to the cyclist.

These tasks and their interactions are visible in Figure 5.1.1 below.

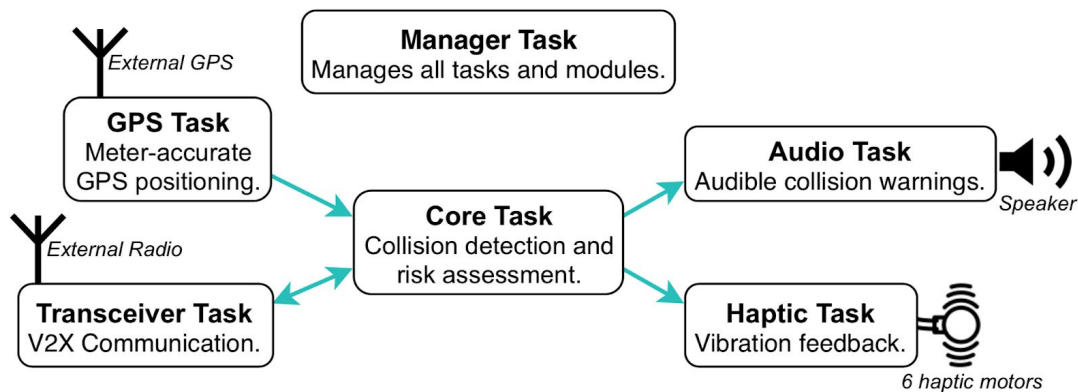


Figure 5.1.1: Free RTOS Tasks

We used multiple forms of testing to confirm the functionality of the core software. First, we incorporated the Ceedling test framework to compile unit and regression tests on PC. This, along with python scripting for converting spreadsheets to test scripts, allows us to test all the collision detection calculations using detailed spreadsheets of confirmed collision scenarios. Secondly, we conducted extensive testing with our Java desktop Traffic Simulator to confirm the full functionality of the prototype.

In addition to the core software, we also completed the Traffic Simulator. The previous version of the Java desktop simulator was upgraded to use the more advanced Lightweight Java Game Library to improve the performance and functionality of the simulator. The final version of the Traffic Simulator allows the user to drive the bike and a selected car simultaneously using the keyboard. The user generates cars in collision scenarios using the detailed control panel and

also uses the number keys on the keyboard to run simulation scripts. This allows for pre-programming 9 different collision scenarios, creating a consistent test environment for the full prototype. Figure 5.1.2 shows the Traffic Simulator in action.

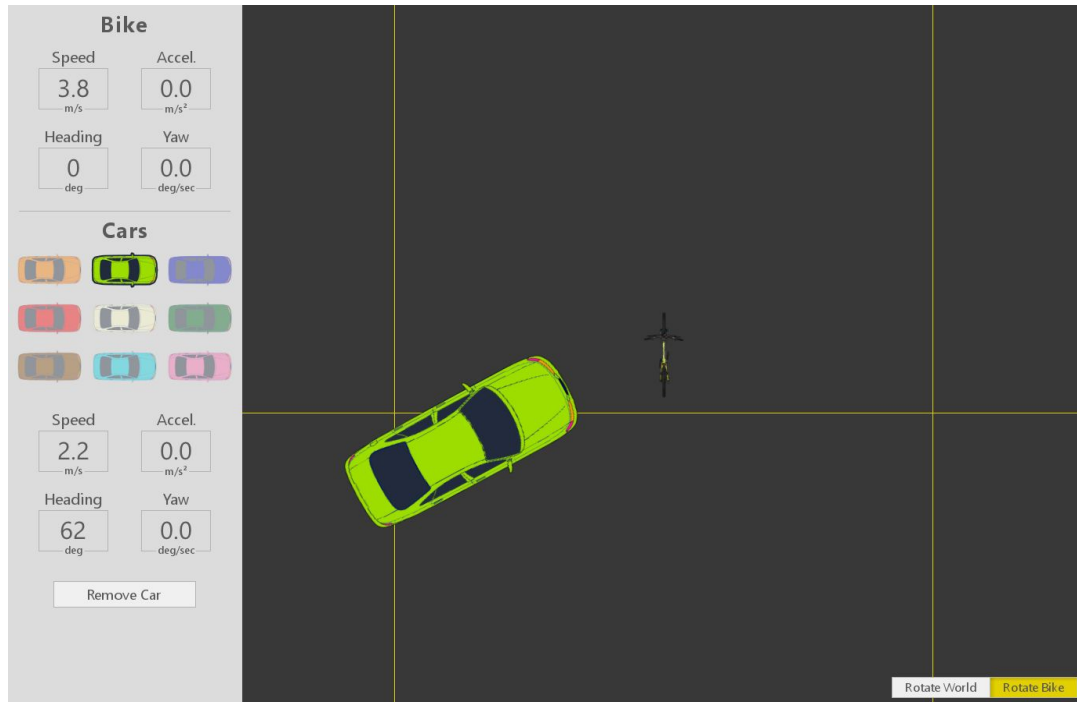


Figure 5.1.2: Traffic Simulator Interface

5.2 Unfinished Tasks

The second iteration of the board must undergo full testing to explore incorporating it into our working prototype demonstration. Also, the third iteration of the board needs assembly. We have all the components and require the arrival of the boards before sending them to our assembler. After completion of the third board, we need to perform tests and measurements on each of the individual systems. Having this working fully would provide the optimal example of our working prototype, however, the lead times for board manufacturing and assembly delayed accomplishing this. Finally, best practice dictates putting together a library of the component symbols and footprints used in the design for future usage.

The completion of the core software and traffic simulator met the goals of our schedule. Ideally both programs would undergo more detailed unit and system tests, but the time constraints of the project prevented us from completing such tests.

6 Project Management

6.1 Schedule Review

Our accomplishments meet the team goals and Gantt Chart we developed at the beginning of the semester. We completed each step of the project within our capabilities. We had hoped to figure out the battery charging circuit but could not find the source of the error on the first board iteration. It provided a good learning opportunity for the next boards we developed though. We fell behind testing the two PCBs during April due to unanticipated long lead times for the board assemblies. It took the assembler 3 weeks rather than the anticipated 1 week to assemble the second board. Further, the design for manufacturability (DFM) checking by the board house (Seeed Studio) added an additional week added to the ordering of the third board. We completed the main project goals, but would have liked to spend more time on some of the additional features. Our final project includes all the software tasks with additional test scripts for running checks on the core software after implementing any changes to the code.

6.2 Final Budget

Our final project spend came out to roughly \$2,600 because a number of line items did not have tax and shipping included. Table 8.3.1 of the Appendix details the entire budget spend. Our board assembler still needs payment for assembling the second and possibly third iteration of the board. The assembly step uses a solder machine to easily solder numerous components at once to a board, allowing us to use smaller components in our design and eliminate errors from difficult soldering.

6.3 Meeting Review

Throughout the course of the project our team utilized WhatsApp as our main method for communicating with one another. This worked out well as it allowed us to have quicker communication than email and allows different device families (Android vs iOS) to work well together. Our group also met weekly to discuss past, current, and future work. This helped keep everyone up to speed with the direction and progress of the project. This also gave time for our advisor to provide technical guidance for design work and industry best practices. We also shared all our documents on Google Drive which enabled us to work simultaneously on the same project and review each other's work.

7 Analysis of Results

7.1 Analysis of Completed Work

The second semester, our focus involved using the research, preparation, and testing from the first semester to develop a working prototype of the entire system. This involved designing and testing three PCBs, finishing the simulation program, writing the core software tasks, testing the drivers for systems connections, and connecting all the protoboards together on the helmet to develop a working model.

We successfully tested one of our stretch features, the haptic warning system. We developed a simple motor driver circuit and selected components for the system. Our first step involved testing the circuits on proto boards. This step helped us because we learned we needed to include pull-down resistors since when the microcontroller powers on, its outputs are in an undefined state which caused random motor spasms. We then implemented the circuits on our second and third PCBs for use in our final product.

After testing our first PCB, we discovered a few errors on the schematic and fixed those. We also learned the importance of isolating the individual systems to optimize our ability to test different parts of the board without disturbing the other parts. The battery charging circuit on this board added a complexity that fell out of the scope of the project. Although we would surely revisit the design when developing a true consumer product, it introduced too much variability and uncertainty associated with testing it for our prototype.

The second PCB we developed included the cell balancer, two buck converters, and haptic motor drivers. Rather than soldering all the components on ourselves, we chose to send them to a board assembler with the hopes of expediting the soldering process, eliminating any soldering errors, and using smaller component package sizes. Further, we manufactured these boards cheaply since we did not need laser vias, narrow trace margins, or more than 2 layers.

The third iteration of the PCB involves the entire system incorporated on to a single board. Developing this board contributes to the development of a final product as it gives an estimate for the size, weight, and cost of the product. We developed a full bill of materials (BOM) for the final product, located in Section 8.4 of the Appendix, which allowed us to estimate production costs for the product in quantities of 1, 10, and 1,000. See Table 7.1.1 below for the estimated production costs for the product. As a part of the design of this board we isolated each of the systems to enable individual testing and current draw measurements. This feature allows us to determine the proper sizing of the battery/power supply system for our application. This board includes many advanced layout techniques including buck converters, fast edges, antennas, large current spikes and RF traces that could prove difficult to test and get working properly. The assembler will also put together this board due to the high quantity of small components in a confined space.

Table 7.1.1: Estimated Production Costs

Quantity	Cost
1	\$292.00
100	\$165.00
1000	\$122.00

The core software running on our MCU on the helmet includes all the components we had hoped to implement within our project scope. Using the FreeRTOS kernel, we developed numerous tasks to simultaneously interact with all the peripheral chips while tying all the associated data together with the core collision and threat detection logic.

We began by programming the software-level interface to each of the connected hardware components one at a time including the GPS, transceiver, audio, and haptic drivers. We carefully engineered the code so that each component could undergo individual testing to verify its functionality before integrating the various pieces into one system. After the testing and verification of the proper functionality of each system, we implemented the core software structures for holding and transferring the information.

Finally, we added a task to manipulate all this data and route it accordingly. This process began by working through equations to detect collisions and near-collisions between vehicles given their positional and movement data. We ensured that our model would detect both linear collisions and vehicle turning collisions. After a significant amount of performance testing, we settled on an algorithm that detects these scenarios, and offers projected data about the vehicles at the time of the incident. Once successfully implementing this equation in the code, we used the information from this algorithm to assess the severity of threats from all the nearby vehicles on the road. After determining the threat levels of the collisions, the haptic and audio tasks simply had to understand the most pressing dangers so that they could convey this information to the cyclist. This collision detection and alert system serves as a great example of how our prototype could be expanded and refined to become a reliable safety product.

The traffic simulator desktop program originally began as a way for us to safely test dangerous collision scenarios without having to actually take to the roads with untested hardware. We implemented a basic interface in which a bike could travel with a specified speed and direction, and cars could generate such that they would collide with or closely encounter the bike. We then coded a serial interface between the simulator and our data relay so that the virtual world within the simulator could transfer wirelessly to the actual smart helmet just as if the authentic data came from nearby cars.

After some basic testing with this simulator, and considering the extent of our collision detection algorithms, we determined improvements for the traffic simulator. We began working

on a second iteration of the program that included a much smoother interface to control the simulation and visualize the world. This version also implemented keyboard controls that enable a user to drive bikes or cars around to achieve whatever test scenarios they might want. Finally, this version added the ability to run predefined scripts so that demonstrations could quickly and reliably show off a few prepared scenarios. This program improved our software testing and debugging phase and serves as a valuable tool when demonstrating the functionality of our prototype.

7.3 Analysis of Incomplete Work

Although we successfully put together a helmet with the development boards, we still have one more step to the final hardware product. The power supply system from the first iteration of the board never received 100% verification due to a few mistakes in the design that made it difficult to properly test. Therefore, the second iteration of the board with the power supply system and haptic drivers must undergo full testing to verify the proper design of the buck converters. This testing process involves using a power supply to power each of the rails individually and verify their outputs. After testing with a power supply, Lithium Ion batteries for the final design will power the system. Having completed the battery test, we could then move on to including the cell balancer circuit in the design and testing its functionality by unevenly charging the two series cells and measuring the output current of the cell balancer.

We already verified the haptic motor drivers, thus the completion of the power supply testing would allow us to replace the off-the-shelf regulator and batteries on our current helmet with the custom board. This would better replicate the final design and reduce the amount of hardware on the helmet.

We also need to run through a full system test of the full system PCB (third iteration). We use jumpers for isolation between each of the systems so that we can test each system's functionality separate from the others. After verifying the individual functionality we could move on to testing the entire system by running the core software on the hardware. We would first do this with a power supply before using the batteries because we do not have official measurements for the power consumption of the individual systems as used by our product. The jumper configuration for each system allows us to break the circuit and make current measurements for each of the systems. This feature aids the development of the prototype because it allows us to evaluate the battery selection for this application and determine how we can properly size the components of the power supply system for optimal functionality.

The remaining work on the core software involves running more strenuous tests to ensure proper operation for an extended period of time. More in-depth trials for different scenarios and adjustments to the threat-detection algorithm could also benefit the overall performance of the unit. We also looked into implementing bluetooth communications for the device, but did not move forward with this optional feature due to time restraints and the complexity of the required code for our use case.

The traffic simulator meets the standards we envisioned for it. The finalized code for the simulator just needs to undergo further testing. A few final bugs need resolution in the next couple of weeks and some endurance testing performed to ensure no bugs present themselves when the program runs for extended periods of time. Finally, the script files for predefined scenarios require development and refinement for an effective and impressive system demonstration.

7.3 Final Product Improvements and Steps

During this year we achieved our main goal of developing a fully functioning prototype of our design which involved testing and verifying the functionality of each essential system. We also completed the simulation program for testing and displaying the functioning product, as well as 3 PCB designs which serve as important test points for a final product. Improving our design would involve upgrading the hardware as well as working on software additions.

After testing the hardware and determining any changes needed for proper functionality, we would work on optimizing the layout of the board to reduce its size. We would also replace all the system jumper connections with 0402, 0 Ohm resistors. Our current and power measurements on the third board would also allow us to determine the proper sizing of the batteries, traces, and other power related components. After completing this step we would work on a 3D printed enclosure that could attach to the helmet and protect the electronics. Further, a true consumer product would require emissions testing for any RF radiation. We would also revisit the battery charging circuit so that users would not have to open the case and remove the batteries to charge them.

Another upgrade to the system will come after the release of the V2X chips from NXP and Qualcomm, requiring us to tweak some of the designs to work on the different frequency band and with the new chips.

The next software step involves developing an app for interfacing with the helmet over Bluetooth. The third PCB iteration includes a Bluetooth antenna, so we could test this functionality with the current boards. The app interface would allow for battery monitoring, and haptic and audio sensitivity adjustments. We could also use it for field testing rather than using the simulator that currently runs on laptops and desktops only.

7.4 Societal Impact

The driving force behind the idea for this project focused on the concern of incorporating cyclists into the autonomous vehicle and smart city ecosystem. Since most of the technology that enables this product to function at its full capability has not gone to market yet, it will not have an immediate societal impact. As autonomous and semi-autonomous vehicles continue to roll out though, the need for the general public to safely interact with them will increase. Cyclists will need the Smart Helmet because it allows them to ride on the road safely with cars. It will also improve vehicle safety, as the extra data from the cyclist's helmet allows for vehicles and drivers

to make smarter decisions when interacting with cyclists on the road. Further, smart city systems equipped with the proper receiver and transmitter technology can communicate with cyclists too. This product paves the way for enhancing cyclist safety and improving road conditions for all road users.

8 Appendices

8.1 Abbreviations

BGA - Ball Grid Array
 BOM - Bill of Materials
 DRC - Design Rule Check
 DSRC - Dedicated Short Range Communication
 ENIG - Electroless Nickel Immersion Gold
 GPS - Global Positioning System
 GUI - Graphical User Interface
 IC - Integrated Circuit
 IMU - Inertial Measurement Unit
 LED - Light Emitting Diode
 MCU - Microcontroller
 PCB - Printed Circuit Board
 RTOS - Real Time Operating System
 V2X - Vehicle to Everything

8.2 Detailed Project Requirements

Table 8.2.1: Major Features

Feature	Description
<i>GPS Connection</i>	A GPS chip and antenna receive satellite data to calculate speed, direction, and location of the biker. The system must receive this GPS data at least 5 times a second. The position should be accurate to within 3 meters, the speed to within 0.5 m/s, and the heading to within 5 degrees.
<i>V2X Communication</i>	The system must transmit Basic Safety Messages at 10 Hz at all times. These messages must contain essential information about the bicycle's position and actions. The system must also receive all V2X messages transmitted from other units within 200 meters line of sight

	of biker and 90% of all messages within 100 meters not line of sight. The message delay must be below 50ms.
<i>Risk Assessment</i>	The product determines different levels of potential risks from the received V2X data.
<i>Rechargeable Battery</i>	The product must run consistently for at least 2 hours off of a full charge making it usable for multiple or long bike rides.
<i>Traffic Simulation</i>	The Traffic Simulation program simulates cars to meter-accurate GPS positions around the biker in order to be able to properly test the product. It sends the simulated V2X data to the bike at 10 Hz for each simulated vehicle.
<i>Audio Warnings</i>	Audio messages must warn the biker of nearby dangers early with time to help the biker - more specifically, with at least 2 seconds before actual danger. False alarms are kept to a minimum and are not dangerous.

Table 8.2.2: Project “Stretch” Features

Feature	Description
<i>Haptic Feedback</i>	Haptic motors located within the helmet notify cyclist of nearby vehicles. This involves experiments with different patterns for conveying direction and speed of vehicles.
<i>Smartphone Display</i>	A bluetooth connected smartphone mounted on the bikers’ handlebars presents visual information about nearby dangers. The app could also allows users to configure settings for the prototype wirelessly.
<i>Custom PCB</i>	Allows for a more compact unit to easily mount on a helmet and demonstrate the helmet as a feasible prototype.
<i>Visibility Lights</i>	LEDs on the module provide visibility for other vehicles, perhaps based on time of day or ambient light level, and a brake light triggered by deceleration.
<i>Module Enclosure</i>	A well designed enclosure protects electronics from outside noise and provides protection.

8.3 Total Budget Spend

Table 8.3.1: Budget Spend from Project Start

Vendor/Order	Description	Cost (no tax/shipping)
Digikey Order 1	Power PCB passive components, IC's, and batteries	\$145.32
Mouser Order 1	Power PCB passive component	\$11.25
Amazon Order 1	Soldering tools and test equipment	\$284.37
Texas Instruments Order 1	Power PCB IC's	\$26.58
Adafruit Order 1	Transceivers, audio boards, IMU's, and haptic motors	\$160.20
mRobotics Order 1	GPS boards	\$150.98
SAE Order 1	DSRC Protocol	\$78.00
Sparkfun Order 1	Antennae and test MCU's	\$99.45
Adafruit Order 2	Power Supply/misc	\$171.45
Amazon Order 3	Helmet Supplies	\$39.91
Amazon Order 3	Speakers	\$28.29
Sparkfun Order 2	IMU	\$34.95
Sparkfun Order 3	Audio/Radios	\$113.96
Seed Studio Order 2	Power Supply/Haptic Driver PCB	\$51.04
Digikey Order 2	Helmet Test	\$7.62
ChipStop Order 1	Apollo2 Blue EVB	\$212.00
Digikey Order 3	Board 2/3 Components	\$167.93
Digikey Order 4	Board 3 Components	\$97.67
VLSI Solutions	Audio Processing SoC	\$116.23
Mouser Order 2	Board 3 Components	\$25.97

mRobotics Order 2	GPS Connectors	\$21.60
Seeed Studio Order 3	Full System PCB	\$453.09
Amazon Order 3	Helmet and Glue Gun	\$33.15
SparkFun Order 4	Audio Amplifiers	\$31.10

Total Budget Spend: \$2562.11

8.4 PCB Bill of Materials

Attached next.

Manufacturer Part Number	Manufacturer	Digi-Key Part Number	Reference Designator	Quantity	Unit Price	Extended Price
1048	Keystone Electronics	36-1048-ND	BH1	1	\$ 4.43	\$ 4.43
10118193-0001LF	Amphenol ICC (FCI)	609-4616-1-ND	J1	1	\$ 0.45	\$ 0.45
ABS06-107-32.768KHZ-T	Abracon LLC	535-12373-1-ND	X1	1	\$ 0.84	\$ 0.84
AC0402FR-078K2L	Yageo	YAG3514CT-ND	R31, R32	2	\$ 0.10	\$ 0.20
AMCA31-2R450G-S1F-T3	Abracon LLC	535-14095-1-ND	A1	1	\$ 0.48	\$ 0.48
BLM15HG601SN1D	Murata Electronics North America	490-3998-1-ND	L3	1	\$ 0.19	\$ 0.19
BM06B-GHS-TBT(LF)(SN)(N)	JST Sales America Inc.	455-1582-1-ND	U\$3	1	\$ 0.56	\$ 0.56
BQ29209DRBR	Texas Instruments	296-27677-1-ND	U1	1	\$ 0.90	\$ 0.90
CC0402JRNPO9BN101	Yageo	311-1024-1-ND	C15, C18, C22, C23	4	\$ 0.10	\$ 0.40
CC0402KRX7R9BB102	Yageo	311-1036-1-ND	C43	1	\$ 0.10	\$ 0.10
CGA2B2X7R1E223K050BA	TDK Corporation	445-5609-1-ND	C7, C8	2	\$ 0.10	\$ 0.20
CGA2B3X7R1H473K050BB	TDK Corporation	445-6897-1-ND	C5, C6	2	\$ 0.13	\$ 0.26
CL05A105KP5NNNC	Samsung Electro-Mechanics	1276-1076-1-ND	C9, C11, C14, C17, C33, C47, C48, C56, C59, C76, C77, C81	12	\$ 0.04	\$ 0.49
CL05A224KP5NNNC	Samsung Electro-Mechanics	1276-1049-1-ND	C10, C12	2	\$ 0.10	\$ 0.20
CL05A334KP5NNNC	Samsung Electro-Mechanics	1276-6665-1-ND	C4	1	\$ 0.13	\$ 0.13
CL05A474KP5NNNC	Samsung Electro-Mechanics	1276-1173-1-ND	C49	1	\$ 0.10	\$ 0.10
CL05B103KO5NNNC	Samsung Electro-Mechanics	1276-1051-1-ND	C34	1	\$ 0.10	\$ 0.10
CL05B104KO5NNNC	Samsung Electro-Mechanics	1276-1001-1-ND	C1, C2, C3, C29, C30, C32, C37, C40, C44, C45, C72, C74, C79	13	\$ 0.02	\$ 0.29
CL05C390JB5NNNC	Samsung Electro-Mechanics	1276-1016-1-ND	C19, C20, C21, C28, C31, C36, C39, C63, C66, C67, C71, C75, C78	14	\$ 0.03	\$ 0.35
CL31A476MPHNNNE	Samsung Electro-Mechanics	1276-3063-1-ND	C24, C25, C26, C27	4	\$ 0.75	\$ 3.00
CL31B226MPHNNNE	Samsung Electro-Mechanics	1276-3148-1-ND	C13, C16	2	\$ 0.59	\$ 1.18
COM-13909	SparkFun Electronics	1568-1394-ND	U5	1	\$ 5.95	\$ 5.95
CRGCQ0402F270R	TE Connectivity Passive Product	A129620CT-ND	R8	1	\$ 0.10	\$ 0.10
CRGH0805F1M0	TE Connectivity Passive Product	A126423CT-ND	R37	1	\$ 0.10	\$ 0.10
CSTNE12M0GH5C000RO	Murata Electronics North America	490-17946-1-ND	Y1	1	\$ 0.40	\$ 0.40
EEE-1EA101XP	Panasonic Electronic Components	PCE3898CT-ND	C64	1	\$ 0.47	\$ 0.47
EN2342QI	Intel	544-2948-1-ND	U2, U3	2	\$ 10.44	\$ 20.88
ERA-2AED101X	Panasonic Electronic Components	P100DECT-ND	R9, R12, R49, R50	4	\$ 0.22	\$ 0.88
ERA-2AED102X	Panasonic Electronic Components	P1.0KDECT-ND	R46	1	\$ 0.22	\$ 0.22
ERA-2AED103X	Panasonic Electronic Components	P10KDECT-ND	R21, R22, R47, R48	4	\$ 0.22	\$ 0.88
ERA-2AED104X	Panasonic Electronic Components	P100KDECT-ND	R3, R4, R38, R39, R41, R43, R44, R45, R52, R53	10	\$ 0.19	\$ 1.88
ERA-2AED753X	Panasonic Electronic Components	P75KDECT-ND	R27	1	\$ 0.22	\$ 0.22
ERJ-2GEOR00X	Panasonic Electronic Components	P0.0JCT-ND	F1, F2, R36, R60	4	\$ 0.10	\$ 0.40
ERJ-2RKF1072X	Panasonic Electronic Components	P10.7KLCT-ND	R28	1	\$ 0.10	\$ 0.10

ERJ-2RKF1690X	Panasonic Electronic Components	P169LCT-ND	R7	1	\$	0.10	\$	0.10
ERJ-2RKF2201X	Panasonic Electronic Components	P2.20KLCT-ND	R40, R42	2	\$	0.10	\$	0.20
ERJ-2RKF2202X	Panasonic Electronic Components	P22.0KLCT-ND	R25	1	\$	0.10	\$	0.10
ERJ-2RKF2872X	Panasonic Electronic Components	P28.7KLCT-ND	R30	1	\$	0.10	\$	0.10
ERJ-2RKF3742X	Panasonic Electronic Components	P37.4KLCT-ND	R23	1	\$	0.10	\$	0.10
ERJ-2RKF4021X	Panasonic Electronic Components	P4.02KLCT-ND	R19, R20	2	\$	0.10	\$	0.20
ERJ-2RKF4700X	Panasonic Electronic Components	P470LCT-ND	R56, R59	2	\$	0.10	\$	0.20
ERJ-2RKF4751X	Panasonic Electronic Components	P4.75KLCT-ND	R33, R34, R35	3	\$	0.10	\$	0.30
ERJ-2RKF6192X	Panasonic Electronic Components	P61.9KLCT-ND	R26	1	\$	0.10	\$	0.10
ERJ-2RKF7322X	Panasonic Electronic Components	P73.2KLCT-ND	R24	1	\$	0.10	\$	0.10
ERJ-2RKF9762X	Panasonic Electronic Components	P97.6KLCT-ND	R29	1	\$	0.10	\$	0.10
ESR10EZPF30R0	Rohm Semiconductor	RHM30.0AECT-ND	R5, R6, R13, R14, R17, R18	6	\$	0.17	\$	1.02
FTSH-105-01-F-DV-K-P-TR	Samtec Inc.	SAM13159CT-ND	U2	1	\$	3.88	\$	3.88
GCM155R71H472KA37D	Murata Electronics North America	490-4915-1-ND	C65, C69	2	\$	0.10	\$	0.20
GJM1555C1H1R2BB01D	Murata Electronics North America	490-8085-1-ND	C53, C54, C55	3	\$	0.19	\$	0.57
GJM1555C1H3R0BB01D	Murata Electronics North America	490-8095-1-ND	C41, C42	2	\$	0.19	\$	0.38
GRJ155R60J106ME11D	Murata Electronics North America	490-13211-1-ND	C35, C38, C57, C58, C60, C62, C68, C70, C73, C78	10	\$	0.17	\$	1.66
GRM155R61A225KE95D	Murata Electronics North America	490-10451-1-ND	C50, C52	2	\$	0.13	\$	0.26
GRT188R61E106ME13D	Murata Electronics North America	490-12323-1-ND	C46, C51	2	\$	0.51	\$	1.02
L-07C3N3SV6T	Johanson Technology Inc.	712-1416-1-ND	L5, L6	2	\$	0.10	\$	0.20
LM4853MM/NOPB	Texas Instruments	LM4853MM/NOPBCT-ND	U10	1	\$	1.79	\$	1.79
LQM18PN4R7MFRL	Murata Electronics North America	490-12065-1-ND	L4	1	\$	0.42	\$	0.42
LT1962EMS8-5#TRPBF	Linear Technology/Analog Devices	LT1962EMS8-5#TRPBFCT-ND	U4	1	\$	4.25	\$	4.25
M20-9990246	Harwin Inc.	952-2262-ND	JP1, JP2, JP3, JP4, JP5, JP6, JP7, JP8, JP9, JP10, JP11, JP12, JP13	14	\$	0.10	\$	1.38
MBR120ESFT3G	ON Semiconductor	MBR120ESFT3GOSCT-ND	D1, D2, D3, D4, D5, D6, D8, D9	8	\$	0.40	\$	3.20
MGSF1N02LT1G	ON Semiconductor	MGSF1N02LT1GOSCT-ND	Q1, Q2, Q3, Q4, Q5, Q6	6	\$	0.56	\$	3.36
MM8430-2610RA1	Murata Electronics North America	490-4980-1-ND	U7	1	\$	0.54	\$	0.54
MX25L1606EM1I-12G	Macronix	1092-1122-ND	U8	1	\$	0.59	\$	0.59
PRT-12895	SparkFun Electronics	1568-1488-ND	Batteries	2	\$	5.95	\$	11.90
PTS645SM43SMTR92 LFS	C&K	CKN9112DKR-ND	S1	1	\$	7.17	\$	7.17
RT0402BRD0720KL	Yageo	YAG1388CT-ND	R1, R2, R10, R11, R15, R16	6	\$	0.38	\$	2.28
SRN2010TA-2R2M	Bourns Inc.	SRN2010TA-2R2MCT-ND	L1, L2	2	\$	0.39	\$	0.78
NA	Amazon	NA	Helmet	1	\$	16.99	\$	16.99
a15071300ux0251	Uxcell	NA	JP12	1	\$	3.62	\$	3.62
PCB	Seeed Fusion	NA	PCB	1	\$	58.47	\$	58.47

MRO-GPS004-MR	mRobotics	NA	GPS	1	\$ 69.90	\$	69.90
652-PTR902-1015FA203	Bourns	652-PTR902-1015FA203	POT1	1	\$ 3.07	\$	3.07
	Ambiq Micro	NA	U6	1	\$ 7.47	\$	7.47
VS1000D-L	VLSI Solutions	VS1000D-L	U9	1	\$ 18.00	\$	18.00
1865	Adafruit		1865 X1	1	\$ 2.50	\$	2.50
1201	Adafruit	NA	H1, H2, H3, H4, H5, H6	1	\$ 11.70	\$	11.70
CX2016DB48000COWPLA2	Kyocera	581-CX2016DB48CWPLA2	X3	1	\$ 0.90	\$	0.90
MRC0206	mRobotics		GPS Connection	1	\$ 3.35	\$	3.35

8.5 Core Software Specifications

Attached next.

GONZAGA UNIVERSITY
School of Engineering and Applied Science
Center for Engineering Design and Entrepreneurship

Core Software Specifications
ENSC 02 - Smart Helmet

Damon George & Robert Brajcich

Revision 2
Last Update: 4/15/2018

TABLE OF CONTENTS

Introduction	3
RTOS Tasks	3
Manager Task	3
Audio Task	3
Transceiver Task	3
GPS Task	4
Haptic Task	5
Core Task	5
Interrupts	6
UART0 ISR	6
Audio FX Pin ISR	6
SPI0 ISR	6
Transceiver Pin ISR	6
Data Structures	7
Global GPS Data	7
Global IMU Data	7
Vehicle	7
Vehicle List	7
Bike	7
Collision Info	7
Personal Safety Message	8
Audio Message Queue	8
Haptic Threat	8
Top Haptic Threats	8
Synchronization Variables	9
Mutexes	9
Global GPS Mutex	9
Global IMU Mutex	9
Global Haptic Threats Mutex	9
Semaphores	9
SPI0 Binary Semaphore	9
Task Notifications	9
GPS Notification	9
Transceiver Notification	9
Audio Notification	10

Timers**10**

Transceiver Software Timer

10

Introduction

This document details the layout, design, and functionality of the core software for the ENSC 02 Senior Design Project - Bike Safety Smart Sensor. This is the software that will run on the Apollo2 Blue MCU on the Smart Helmet. This document will be updated as software changes are made.

RTOS Tasks

In order of decreasing priority.

Manager Task

As the highest priority task, this is responsible for managing all other RTOS tasks. Namely, this task handles starting and stopping different tasks. This task receives messages in the Manager Queue, and then starts or sleeps task using the Task Management Event Group. This allows the manager task to sleep certain modules when not in use. For example, when the GPS detects a lack of movement for a certain period of time, all other tasks are slept until the helmet starts moving against.

Audio Task

As the highest priority of the managed tasks, this is responsible for determining and sending commands to the Audio FX board to play different audio files. This task will RTOS block on the Audio Message Queue that is filled by the core task. When a collision message is added to the queue, this task will calculate the appropriate audio message, turn on the audio amplifier, and send the short play command over UART1 to the Audio FX board. Then it will block until the rising edge pin interrupt from the Audio FX board, at which point it will turn off the amplifier and return to blocking on the message queue again.

Transceiver Task

This task will handle all the communications via the transceiver. It will be accomplished by one task rather than 2 due to the inability of the transceiver to buffer send and receive messages simultaneously. It will consist of a loop that begins with a statement that performs an RTOS block waiting for either a task notification from the transceiver isr (see below) or a periodic timer (which notifies every ~10Hz To send out a new Basic Safety Message, possibly randomized slightly). It then performs relevant functions to interact with the transceiver on a low level to read or write messages. The main pseudocode of the module is shown below (2 possible versions shown):

Block on periodic timer AND transceiver interrupt notification

```

if periodic timer notification:
    if !sending:
        set transceiver mode to idle
        sending = true;
        load message into transceiver FIFO
        set transceiver into TX mode

if ISR notification:
    read interrupt flags register
    if TX done flag and sending:
        clear interrupt flags register
        set transceiver into RX mode
        sending = false
    else if RX done flag and !sending:
        read message from FIFO
        clear interrupt flags register
    else (in case RX interrupt came after 'sending' set to true)
        clear interrupt flags register (only RX bit?)

```

The ISR discussed above is simply tied to the pin G0 on the transceiver module. When an interrupt occurs and calls the ISR, it will send a task notification to the transceiver task, which can then handle the interrupt accordingly in due time.

The SPI communication layer for this module will have a custom implementation in order to be able to initiate a transaction, then perform an RTOS block until the dedicated hardware finishes the transaction. The SPI transfer function will follow a flow as shown below:

```

spi_transfer(...)
    initiate hardware spi
    take binary semaphore (waiting for ISR to release it)

```

And the onboard SPI transaction complete interrupt will be tied to an ISR that releases the spi_transfer function by giving the semaphore.

GPS Task

The GPS task is focused on receiving and parsing the serial data stream from the NEO-M8N GPS Module. The main loop will wait on a notification from the UART0 ISR. When it receives a notification, it will parse the newly received sentence using a third party library. If the new data is valid, this task will copy the new data to the Global GPS Data Buffer, which requires taking the mutex for that buffer. Both the Notification and the Mutex will be RTOS Blocking. If either of these timeouts, errors will occur.

Furthermore, every couple minutes this task will use the timestamp from the GPS data to update the MCU's real time clock (RTC).

Haptic Task

This task periodically (~10 Hz) reads through the Nearby Vehicle Vectors to update the PWM outputs for the haptic motors according to the top threats from the list. Accessing the List requires holding the associated mutex.

Core Task

The core task will primarily be focused on the ongoing gathering of data from the various other modules, processing this data to assess risks and generate notifications, and performing any other ongoing background tasks. The primary loop of the core task will consist of the following logic:

```

if new_gps_data flag set:
    lock gps_data_mutex
    create copy of GPS data for core task to use
    clear new_gps_data flag
    unlock gps_data_mutex

if new_imu_data flag set:
    lock imu_data_mutex
    create copy of IMU data for core task to use
    clear new_imu_data flag
    unlock imu_data_mutex

for each new packet in bsm_queue:
    if new packet isn't too old:
        if new packet is simulated bike data:
            update bike data and timestamp
        else:
            update vehicle data and timestamp using linear search through the
            vehicle_list

get next vehicle in vehicle_list:
    if next vehicle is null:
        //we've reached the end of the linked list, start over
        current vehicle = head of vehicle_list
    else:
        if the vehicle's timestamp is old:
            remove vehicle
        else:
            calculate risk for the vehicle
            update top audio and haptic threats if necessary

```

The top audio threat is tracked by the core task, and top 3 haptic threats are tracked as well. If the top audio threat is above a certain threshold, the collision data will be sent to the Audio Message Queue. The top three haptic threats (above a certain threshold) will be constantly read and output by the haptic task.

Interrupts

In order of decreasing priority.

UART0 ISR

This interrupt is triggered by the hardware UART0 interface used by the GPS task. This is the highest priority interrupt because otherwise the UART0 hardware FIFO could easily fill up and begin losing data, which would prevent the device from getting accurate GPS data.

This ISR first checks the UART0 interrupt status register to determine the cause of the interrupt. If the cause is a FIFO threshold, then this will copy the data from the hardware FIFO to the UART0 memory buffer that was created in the initialization of the GPS task. As this ISR copies each character to memory, it will look for end of line characters. If one is found, this ISR will notify the GPS Task so that the GPS Task can move to the Ready state and parse the new data. This ISR will finish by calling `portYieldFromISR()`.

Audio FX Pin ISR

This ISR is triggered by the ACT pin on the Sound FX board, which goes low when audio is being played. On the rising edge, this interrupt will give a Binary Semaphore to the Audio Task to acknowledge that the audio file successfully finished playing.

SPI0 ISR

This interrupt is triggered by the hardware SPI0 interface which is used by the Transceiver Task. This ISR will first check the cause of the interrupt. If this is a FIFO threshold interrupt, then this ISR will handle filling/emptying the FIFO from/into the associated memory buffers created in the initialization. If this is a transaction complete ISR, then it will give the Binary Semaphore to the Transceiver Task to signal that the SPI transaction is finished.

Transceiver Pin ISR

This interrupt is triggered by the interrupt pin on the LoRa chip. This ISR will simply notify the Transceiver Task that this interrupt has occurred.

Data Structures

Global GPS Data

The GPS Data will be stored in global struct requiring a mutex to access. This data includes the following:

- Longitude
- Latitude
- Speed
- Course
- Timestamp
- Fix Status
- Horizontal Dilution of Precision.

Global IMU Data

The IMU Fusion data will also be stored in a global struct requiring a mutex to access. This data includes:

- 3 Axis of Acceleration
- Euler Vectors

Vehicle

A struct used by the core task to hold data about a single vehicle, including id, location, speed, etc, as well as a timestamp holding the time of the most recent update to the data.

Vehicle List

The list of current vehicles around the biker, local to the Core Task. This is a Linked List which is ordered by Vehicle ID. This will be traversed using a linear search on vehicle ID.

Bike

The current info about the biker - local to the Core Task. This holds the biker's GPS and IMU data, copied from the corresponding global data so as to provide easier access for the Core Task.

Collision Info

This is a struct used across the core software to hold the following information regarding a calculated collision scenario for the biker:

- The current location of the car around the biker
- The location of the car right before the collision
- The distance between the car and biker at the collision

- The time until the collision
- The relative speed of the car to the biker at the collision
- The relative angle of the car to the biker at the collision

Personal Safety Message

A byte array holding all the data contained in a DSRC personal safety message. Data is accessible through functions like "PSM_ID PSM_GetId(PersonalSafetyMessage *bsm)" or "void PSM_SetId(PersonalSafetyMessage *bsm, PSM_ID id)". The data includes:

- ID - 4 byte random identifier.
- timestamp - 2 byte integer from 0 to 60999 referring to milliseconds within current minute.
- type - 1 byte representing type of user (value of 2 for cyclist).
- message count - 1 byte from 0 - 127.
- Latitude - signed 4 bytes representing 0.1 microdegrees from -90 to 90 degrees. 900000001 represents unavailable.
- Longitude - signed 4 bytes representing 0.1 microdegrees from -179.9 to 180 degrees. 1800000001 represents unavailable.
- Elevation - integer from -4096 to 61439 for decimeters.
- Positional Accuracy - 1 byte accuracy in 5 cm increments, for each 2 axis.
- Speed - 2 bytes from 0 to 8190 in units of 0.2m/s with 8191 representing N/A.
- Heading - 2 bytes from 0 to 28799 in units of 0.0125 degrees with 28800 being N/A
- Acceleration - 2 axis of acceleration. Each is 2 signed bytes from -2000 to 2000 in units of 0.01m/s² with 2001 representing N/A..
- Vertical Acceleration - 1 byte from -127 to 127 in units of 0.02 G.
- Yaw - 2 signed bytes in units of 0.01 degrees/sec.

Audio Message Queue

An RTOS Queue holding collision info to be turned into audio messages and played to the user by the Audio Task. The Audio Task blocks on this Queue. This Queue can hold up to 5 collisions to warn the user of.

Haptic Threat

A simple struct holding the id of a vehicle, its distance and angle from the biker, and its haptic threat level.

Top Haptic Threats

A list of the top 3 haptic threats as determined by the core task. This list is synchronized by the Global Haptic Threats Mutex and is read by the Haptic Task to determine the output of the vibration motors on the Smart Helmet.

Synchronization Variables

This section details the objects used to synchronize the RTOS tasks. Mutexes are used to protect data reads or writes to memory from being interrupted by another task looking to read or write the same data. Binary Semaphores are used by tasks to wait for another task or interrupt to complete some operation. Notifications are a simple way for tasks or interrupts to notify another task that an event has occurred.

Mutexes

Global GPS Mutex

The Mutex required to read or write the Global GPS Data.

Global IMU Mutex

The Mutex required to read or write the Global IMU Data.

Global Haptic Threats Mutex

The Mutex required to read or write the Top Haptic Threats list.

Semaphores

SPI0 Binary Semaphore

The semaphore used by the SPI0 ISR and the Transceiver Task. The Transceiver Task blocks on this semaphore after starting an asynchronous SPI0 transaction. When the transaction finishes, the SPI0 ISR will return the semaphore, unblocking the Transceiver Task.

Task Notifications

GPS Notification

The UART0 ISR notifies the GPS Task when it encounters a new sentence from the serial data stream from the GPS Module. This will simply increment the GPS Task's Notification Variable.

Transceiver Notification

Both the Transceiver Pin ISR and the Transceiver Timer can notify the Transceiver Task. The Transceiver Pin ISR notifies the Transceiver Task every time it runs by setting bit 0 of the Transceiver Task's Notification Variable. The Transceiver Timer notifies the Transceiver Task at 10 Hz by setting bit 1 of the Transceiver Task's Notification Variable.

Audio Notification

The Audio FX Pin ISR notifies the Audio Task every time it runs by incrementing the Audio Task's Notification Variable.

Timers

Transceiver Software Timer

The 10 Hz RTOS timer used to notify the transceiver when a Safety Message needs to be transmitted. Random milliseconds are added to this timer to prevent the transmission of safety messages from constantly preventing the reception of another safety message being sent at the same time and frequency.