SMART Bike Helmet

Team 10

Spring 2023

Michelle Liang, Paul Chen, Joshuah Chen

1. Introduction

1.1. Problem To Be Solved

Biking is an enjoyable activity to do for leisure or athletics, especially in a flat city like Los Angeles. However, California is considered one of the most dangerous states for cycling. Based on West Coast Trial Lawyers, cyclist deaths make up 4 percent of all car accidents, which is double the national average. This not only means that there are significantly more people who bike in California but also that bikers are more likely to get into accidents. Outside of California, according to a study performed by the Consumer Product Safety Commission, one-third of all non-fatal bicycle injuries in the United States are head-related, and a majority of the 80,000 cycling-related head injuries treated in emergency rooms are brain injuries. Even if they don't cause death, brain injuries can significantly reduce a person's quality of life, whether physically or mentally. Even aware of the potential risks, more than half of cyclists do not wear helmets. Though helmets are a cyclist's best line of defense – reducing head injury by more than 50% – most people don't wear a helmet either because they think it's a hassle or because they just don't find the benefit of using one if they don't get into accidents often. Although these statistics only cover bikers, there are other modes of traveling, such as scootering or skating, in which people do not often use helmets. In recent years, more and more people have started to use electric scooters and skateboards; these can achieve higher speeds and cause more severe accidents. Thus, it is important for people to take measures to stay safe on the road.

1.2. Objective and Scope of the Project

The goal of this project is to design and build a smart bicycle helmet that will allow users to have a safer experience traveling on any light vehicle and incorporate additional functionalities that can aid the user while they are on the road. Most light vehicle users need to follow certain rules, such as using an arm to indicate in which direction they will turn. But this rule requires them to remove a hand from the handle, which can be dangerous. It is also dangerous when bikers need to turn their heads to check if something fast is approaching them from either side or directly behind. To help reduce these potential safety hazards, we want to add features to bike helmets that can aid bikers with signaling on the road while also providing them with information about their surroundings. In addition, if bikers get into an accident and the accident is serious enough, they may not be able to get the help they need, especially if they are alone in a rural area. Thus, we also want to include safety features that will allow emergency teams to be notified when an accident happens and provide the biker's location.

2. System Architecture and Components

For the design of this system, we wanted to have all the sensors and communications incorporated into the helmet. This way, users will not have to worry too much about installing various components onto their bikes by themselves, as this will introduce complications and may cause the product to malfunction if the installation is not done properly. However, our final design will still contain one external circuit to be installed on the bike, which will be a switch that will send a turn signal to the helmet. We decided on this because it is much safer to control something on the handle than to raise your hand while biking to perform the signal by, for example, pressing a button on the bike helmet.

Therefore, our design of a smart helmet consists of two circuit boards, with the master board on the helmet and the slave board on the handle. Each one of them will use a microcontroller to control the modules on the board. The main sensors and components will be located on the helmet board, while only the turning switch and a radio module will be on the handle board. These two boards will communicate using radio signals. A high-level overview of the whole system can be seen in Figure 1.

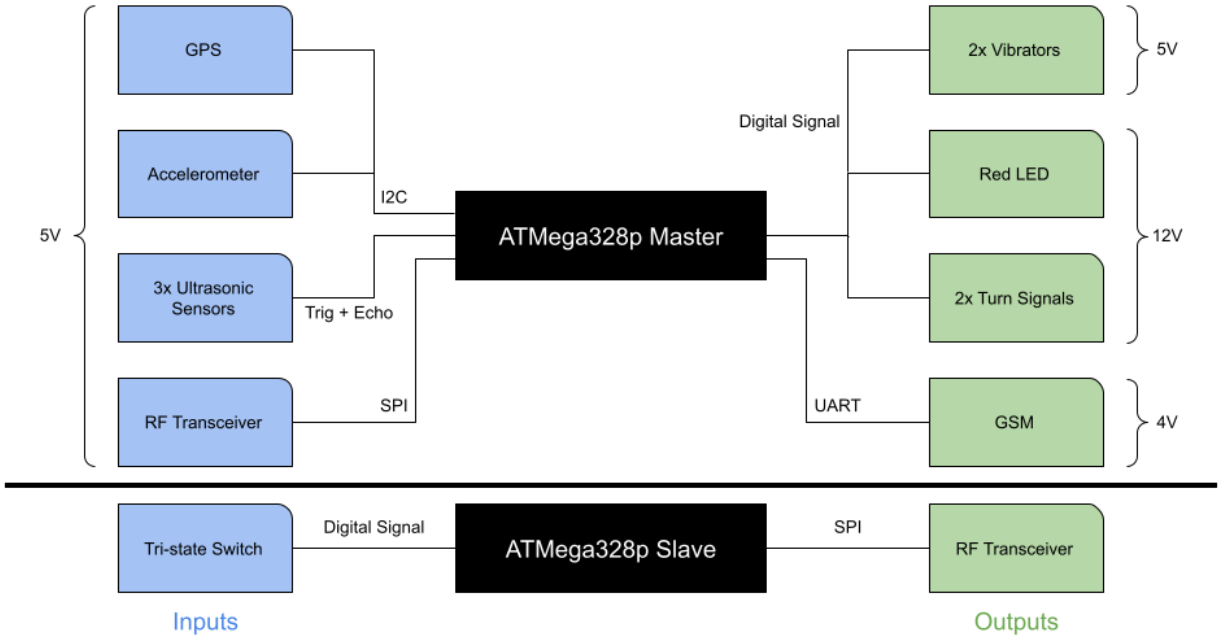

Figure 1. Overview of the Design

2.1. Microcontroller

The ATMega328p microcontroller serves as the brain of our system, providing processing power and controlling the various sensors and modules. It is programmed using the AVR toolchain, an efficient and reliable method for embedded systems programming.

2.2. Sensors

2.2.1. Ultrasonic sensors

The HC-SR04 ultrasonic sensors emit ultrasonic waves and detect the time taken for the waves to bounce back after hitting an object. By calculating the time difference, the system can determine the distance to nearby objects, which provides the microcontroller with information about the surroundings of the biker. This information will then be used to enhance the biker's situational awareness. The ultrasonic sensor uses its own communication standard, which uses two pins to send and receive data from our microcontroller.

2.2.2. Accelerometer

The accelerometer, LIS3DH, measures acceleration in a 3-dimensional way, which provides information about the change in movements, and their intensity, on the biker. Based on these values, the system is able to identify the directional movement of the biker, whether they are actually braking or not, and if they even got into an accident. This uses I2C communication to send information to our microcontroller.

2.2.3. GPS Module

The GPS module, PA1010D, provides accurate location data, enabling features like route tracking and navigation. The GPS also uses I2C communication to send information to our microcontroller, offering a simple and efficient way to exchange data.

2.3. Communication Modules

2.3.1. GSM

The GSM module, SIM800L (HiLetGo), operates at 4V and communicates with the microcontroller via UART on serial ports. This enables features like sending text messages or making calls in emergencies, as long as a SIM card with 2G service is provided.

2.3.2. Radio

The radio module, RFM69HCW, uses SPIs for communication, providing wireless connectivity for data transmission between the master and slave boards.

2.4. Peripheral Signals

2.4.1. Turn Signals

The turn signals comprise a 33 SMD LED Arrow (Leadtops) that operates at 12V. It is controlled using digital pin control, allowing the microcontroller to manage its on and off states.

2.4.2. Brake Signal

The brake signal utilizes an LED strip (HYADA) operating at 12V to provide a clear and visible indication of the biker's intention to slow down or stop. This feature enhances road safety as the turn signals by effectively communicating the biker's intentions to other road users.

2.4.3. Vibrator Motors

The system incorporates two vibration motors (Miskall) to alert the user about potential hazards or objects on their left, right, or back. These motors are controlled through digital pin control, enabling the ATmega microcontroller to manage their activation based on the sensor data. This haptic feedback enhances the biker's situational awareness and promotes safer cycling.

2.5. Other Hardware Components

2.5.1. Demultiplexers and Inverters

Demultiplexers help us save output ports, as ultrasonic sensors require many output ports. However, since our ultrasonic sensors are triggered through “high” signals, and the demultiplexers are low-output, we needed to connect all the demultiplexer outputs through inverters to achieve this.

2.5.2. Voltage regulators

Voltage regulators ensure a stable power supply to the 4V GSM module and the 12V LEDs. This setup guarantees smooth operation and extends the lifespan of the electronic components.

2.5.3. MOSFETs

To efficiently control the LEDs from the ATmega microcontroller, we can utilize MOSFETs as electronic switches. This approach allows us to handle higher current loads and isolate the microcontroller from potential voltage spikes, ensuring reliable operation and safeguarding the system from damage.

3. System Design and Implementation

3.1. Circuit Design and Schematics

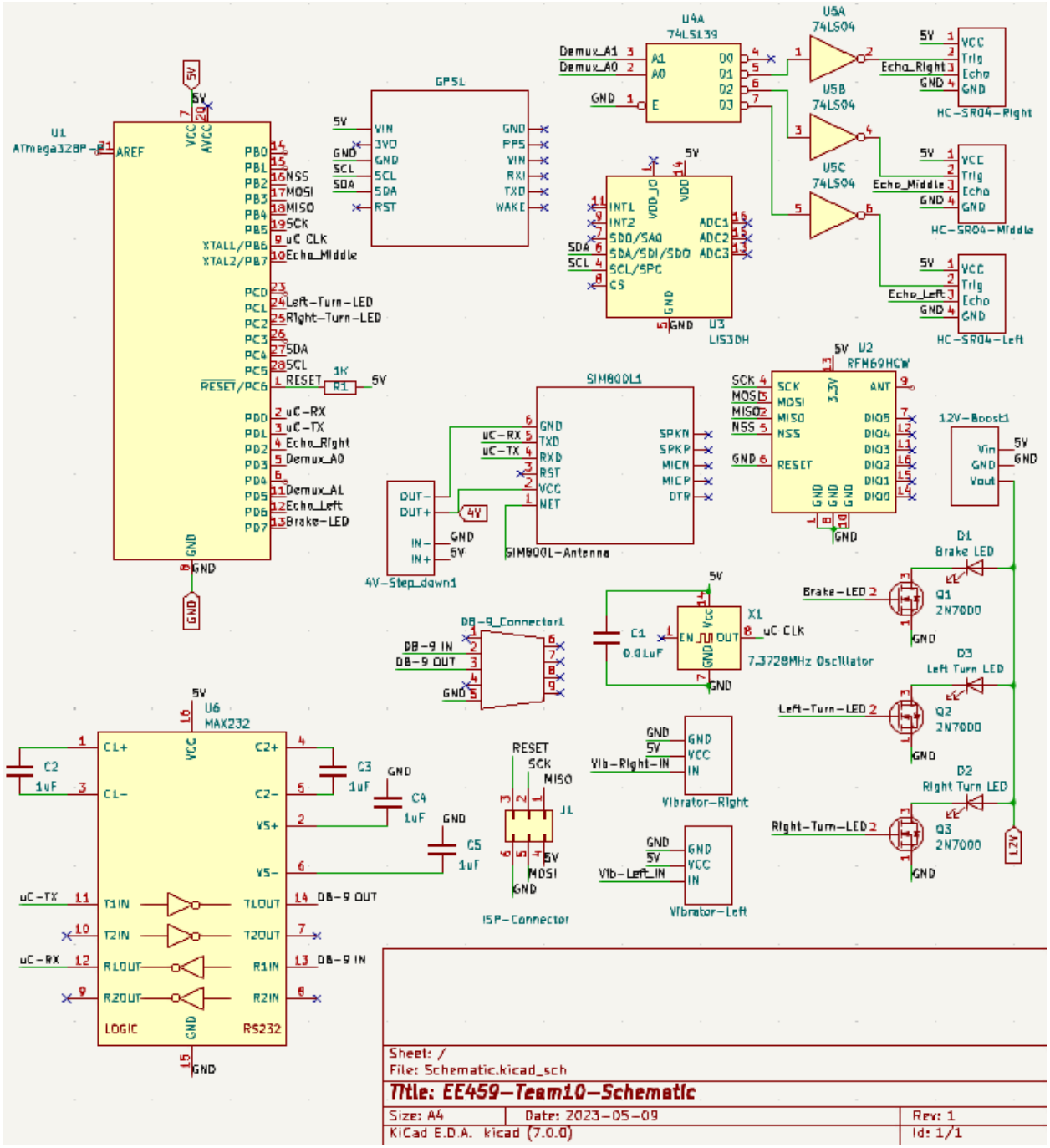

Figure 2. Schematic for Helmet Board

Note: No schematic for handle board because the radio connections are identical, and switch pins are mentioned in Section 3.2.5.

3.2. Hardware Assembly and Integration

3.2.1. Ultrasonic Sensor:

Three ultrasonic sensors are integrated into the system, each with its pairs of trigger and echo pins. Due to a limited number of output pins, the trigger pins are connected to a 4-to-2 demultiplexer and an inverter, as the demultiplexer produces inverted outputs. The trigger pin is used to signal the ultrasonic sensor to start emitting pulses. In response, the ultrasonic sensor sets the echo pin high and then back to low once it receives the reflected pulse, and the width of the pulse signal determines the distance the sensor detected. Because the echo pin needed to be properly connected, it was necessary to use their pins, so we had to dedicate 3 different echo pins for each of our ultrasonic sensors.

3.2.2. GPS and Accelerometer (I2C):

The I2C interface is implemented in hardware by connecting the SDA and SCL pins on the microcontroller to two wires and using two pull-up resistors (typically 4.7k or 10k Ohms) on each line to ensure proper voltage levels. The GPS and accelerometer modules have their SDA and SCL pins wired to the corresponding SDA and SCL lines coming out of the ATmega microcontroller. This allows for efficient communication and data exchange between the microcontroller and the sensors because the I2C protocol only requires us to dedicate two pins and to communicate with two sensors.

3.2.3. Radiofrequency Transmitter:

The radio modules are implemented on both the helmet board and the handlebar board. The SPI interface between the radio module and the microcontroller is implemented by connecting the MOSI, MISO, SCL, and CS pins of the radio to pins 16 to 19 of the microcontroller, which are designated pins for the ATMega328p (its internal SPI hardware is connected to these pins). The CS pin is controlled by the microcontroller and lets the radio know when an SPI exchange is about to take place. The RST pin on the radio is tied to GND, allowing the radio to be turned on whenever it is supplied with power. The EN pin on the radio is tied to Vin and supplies power to the radio whenever Vin is supplied with 5V. The G0 pin on the radio is wired to an interrupt-enabled pin on the microcontroller. The interrupt-enabled pin is initialized so that it is triggered whenever an internal radio interrupt is detected, and necessary actions are taken in the software. A 6.5-inch wire antenna is soldered to the radio since the radio works at 433 MHz.

3.2.4. GSM Module:

The GSM module is connected to the microcontroller's serial TX and RX pins, enabling communication between the microcontroller and the GSM module. This setup allows the

ATmega microcontroller to send and receive data using UART through the GSM network. In addition, an antenna is required to be either soldered to or attached to its dedicated pin, depending on the type of antenna you will be using, as the module comes with two different ones. However, to actually communicate with a device through SMS or calls, it is required to have an active SIM card inserted into the dedicated SIM card slot on the module. Apart from this, the SIM card needs to have an active service, and the provider uses a 2G network, which is very rare now, but some providers (such as T-Mobile) still have it.

3.2.4. Output Signals:

The vibration motors have built-in MOSFETs, which only require an output pin from the ATmega microcontroller for control. On the other hand, the turn signals and brake signal LEDs require separate MOSFETs for each, as they operate at 12V and require a higher current than the ATmega microcontroller can directly supply. By using MOSFETs for the LEDs, we can efficiently control their operation and safely handle the higher current loads, ensuring reliable performance and protecting the microcontroller from potential voltage spikes or damage.

3.2.5. Tristate Switch:

A tristate switch is used in the handle board to determine which turn signal do they want to activate or if none at all. The input wire is connected to the 5V power supply, while the two output wires are connected to pins PC2 and PD3 in the handle microcontroller. Both wires are pulled down with a 10kΩ resistor to protect the microcontroller from a high current when a wire is switched on.

3.3. Software Development and Programming

3.3.1. Ultrasonic Sensor:

Our program uses three ultrasonic sensors, which we alternate between using a flag to prevent overwriting the counter while waiting for echoes to return from the other sensors. When we signal an ultrasonic sensor to emit pulses, it sets the echo pin high and then back to low once it receives the reflected pulse.

If an interrupt is triggered while waiting for echoes from other sensors, we set a flag to indicate that the counter should not be overwritten until the corresponding echo has been received. By alternating between the three sensors in this way, we can accurately detect the presence or distance of objects in our environment while having the same polling rates.

For more detailed information, please visit our GitHub page (Appendix 8.2).

3.3.2. GPS and Accelerometer (I2C):

The I2C protocol is widely used in embedded systems to communicate with sensors and other devices. It allows two-way communication between a master device (such as a microcontroller) and one or more slave devices (such as sensors). The master device can send commands and data to the slave devices, and the slave devices can send data back to the master device. The message typically contains the address of the slave device being addressed, along with data or commands to be sent or received.

In our project, we use the I2C functions provided by the EE459 website to obtain sensor values. To retrieve sensor values using I2C, we typically need to write specific values to certain registers in the sensor. The sensor will then return values corresponding to the requested data. The length of the requested data varies, and one element is usually 1 Byte.

For example, when communicating with an accelerometer using I2C, we can expect to receive three values containing the acceleration in X, Y, and Z. We calculate the overall acceleration and direction to ensure the deceleration is not coming from gravity. On the other hand, when we communicate with a GPS sensor, we receive NMEA data, which we need to extract using parsing techniques. To keep our data up to date, we are constantly polling these sensors.

3.3.3. GSM Module:

The GSM module in our project uses UART communication protocol to communicate with the microcontroller. UART is a popular asynchronous serial communication protocol used for transmitting and receiving data between two devices. In our project, we have implemented UART communication using code from one of our labs.

To control the GSM module, we simply send a specific command over the serial connection from the microcontroller to the GSM module. The command is followed by a "\r\n" character sequence, which indicates the end of the command. The GSM module then responds to the command by sending data back over the serial connection.

The UART protocol is reliable and widely used for serial communication, particularly in embedded systems and microcontrollers. It is simple to implement and supports data transfer rates of up to several megabits per second. By using UART communication, we are able to control the GSM module and receive data from it in a straightforward and efficient manner.

3.3.4. Radiofrequency Transceiver:

For software development of the radio, the cristi85/RFM69 Github repository was referenced. Many functions are needed for radio initialization, transmission, and reception.

To allow SPI communication between the microcontroller and the radio, some SPI functions are implemented:

- SPI_init: CS, MOSI, MISO, and SCK are initialized, the microcontroller is set as master, and clock rate is set
- SPI_transfer8: eight bits of data are transferred over SPI
- SPI_receive8: eight bits of data are received over SPI
- readReg: SPI transfer: send address of radio register to receive from, then receive register value
- writeReg: SPI transfer: address of radio register to write to, then transfer value to write to register

The radio has to be initialized by the master through SPI by configuring different registers in the radio:

- initialize:
    - Initialize radio register values for different radio functions: including operation mode, set bit rate, set frequency, configure synchronization, configure radio for packet mode, set maximum payload length, set FIFO threshold

Functions are also needed for the radio to send and receive data. Below are some important functions for sending and receiving.

- send:
    - If the radio cannot send yet, then finish receiving any packets
    - Otherwise, send frame of data to the other radio
- sendFrame:
    - Write to FIFO: drive CS pin low and start SPI transfer:
        - Send FIFO address with write bit, and then transfer data size, address of radio node being sent to, address of own node, control byte
        - Send the data in the buffer
        - Pull CS pin high
- interruptHandler:
    - If radio mode is RX and a payload is ready:
        - SPI transfer: send FIFO register address with read bit, and then start receiving from FIFO: receive payload length, targetID, data length, sender ID, control byte, and then the data bytes

Aside from the radio functions, separate code files are needed for the helmet radio and the handlebar radio to communicate with each other.

Helmet radio code:

- Initialize SPI
- Initialize radio with 433 MHz frequency, node ID, and network ID
- In infinite while loop: put radio in listen mode and wait for packet to be received; once packet is received, turn on left or right turn signal or neither based on the data received

Handlebar radio code:

- Initialize SPI
- Initialize radio with 433 MHz frequency, a different node ID than helmet radio, and the same network ID as helmet radio
- In infinite while loop: set radio mode to RX, check if radio can send a message, and send a message
    - The message is determined by the state of the tristate switch wired to the same microcontroller the radio is wired to

4. Testing and Evaluation (Functionality)

4.1. Sensor Performance and Accuracy

4.1.1. Ultrasonic Sensors

The ultrasonic sensors were very accurate within the range of 5 ~ 250cm. Anywhere outside of this range, the sensor will likely misread the value but is still likely to provide a reasonable measurement. Unfortunately, our right sensor did not function properly, even after it was replaced. The original sensor had a problem where its echo line had a low voltage of 1.2V, instead of 0V. This was above the threshold voltage for our microcontroller to consider it a “low” digital signal, which was required to trigger our interrupt. Because of this, our microcontroller was stuck waiting for a low signal, which stopped it from polling the other ultrasonic sensors and stopping our whole sensing system from working. Thus it needed to be replaced. The ultrasonic sensor we used to replace it unfortunately didn’t work either, as for some reason, it never returned any echo signal, even after a trigger signal was sent to it. Because of time constraints, we were unable to obtain a third ultrasonic sensor to replace it.

4.1.2. Accelerometer

The accelerometer was accurate and provided three values for each axis. Although there was no documentation of the conversion between the value sensed to the force applied, we realized that gravity gave a value of 64. Thus we did our calculations based on this value.

4.1.3 GPS Module

The GPS module is able to obtain information from different satellites. However, the only information we really need is from a satellite with an abbreviated name of GNGGA. Although this information is not received constantly, it is received at least once every 2-3 seconds, consistent enough to provide this information in an emergency. The main problem with this module is that its location doesn't work properly inside buildings, thus if the user is indoors or between tall buildings, their GPS may not have any data.

4.2. Communication Modules Functionality

4.2.1. Radio Modules

Both radios had a working MOSI line and could receive messages from their respective microcontrollers over SPI. Transferred data was seen on the oscilloscope when the probe was attached to the MOSI line. The oscilloscope also showed that both the SCL and CS lines were working correctly. When the radio was selected by the master, 8 clock cycles appeared on the oscilloscope for 8 bits of data transfer. The radio on the helmet board also had a working MISO line and could transfer data back to the master over SPI. However, the radio on the handlebar board had a broken MISO functionality for SPI communication, so the microcontroller could not receive the requested values from the radio over MISO. This hardware problem was not discovered until much debugging had been done, so the radio functionality could not be fully implemented.

4.2.2 GSM Module

Our GSM Module worked well as long as a SIM Card with a 2G network was placed in the slot. Although we originally obtained an AT&T SIM card, AT&T doesn't provide any 2G service, so our GSM module could not connect to the cellular network. However, one of our teammates uses a T-Mobile SIM Card, and T-Mobile does provide a 2G cellular network. With this SIM card used, the GSM module was able to send SMS messages to the dedicated phone number with the information we needed to send. If this module were used next year, it would be possible that it may not work anymore as T-Mobile is shutting down its 2G cellular service on April 2nd, 2024.

5. Future Improvements

5.1 - Sensor Quality

The current ultrasonic sensors can be inefficient due to their shared interrupts, which require alternating between the three sensors. This setup limits their reaction speed and performance. To improve distance measurement and efficiency, a LiDAR sensor can be considered as an alternative.

In addition, and as mentioned above, the GPS module's performance may suffer indoors or in densely-built urban environments. To address this, we could use a more advanced GPS module or leverage the internet to access Google Maps' Geolocation API. This would enable the system to obtain approximate location data even in challenging conditions.

5.2 - Power Supply

We can combine a rechargeable battery with a solar panel to ensure a reliable and environmentally-friendly power supply. The solar panel can charge the battery during daylight hours, while the battery provides a consistent power source for the system day and night. This configuration reduces the need for frequent battery replacement and minimizes our environmental footprint.

5.3 - SOS Phone Calls

Our current implementation only supports SOS text messages. We can integrate a microphone and speaker with the GSM module to enhance the system's emergency communication capabilities. This will allow the user to make SOS phone calls, providing a more direct and immediate line of communication during emergencies.

6. Operation of Device

Our goal for this device is for it to be as user-friendly as possible, allowing the user to use it off the shelf without the need to perform too many setups. The only installation required will be to secure the handle board on either side of the bike handle, depending on the user’s preference. Of course, batteries will have to be added on both the helmet and handle boards, a SIM card will have to be provided either by the manufacturer or the customer, and a new method will have to be implemented for a unique emergency contact to be set up by the user. In our current design, we assume that the emergency contact information has already been pre-set by the manufacturer, such as 9-1-1 or any other contact provided by the customer, and thus the user doesn’t have to input it themselves.

When the product is properly installed, the user will only have to turn on the helmet board, and they can start using it immediately. Once on the road, if any objects approach the user from the left, right or behind, motors on the respective sides will vibrate twice to indicate where the object is coming from. If it's the right side, only the right motor will vibrate, and only the left motor will vibrate if the object comes from the left. If it's from the back, both motors will vibrate. They will not vibrate again until the object has left the "close vicinity" of the user. Unfortunately, the right sensor of our current prototype is not functioning, so the vibrators will only react when objects approach the user from the left or back.

Any deceleration will also be detected, and the red LED located at the back of the helmet will turn on. It is independent of the brakes of the bike, so if the user is still pressing on the brakes, the red LED will not turn on because it is not detecting any deceleration. This could potentially be a drawback because the helmet does not have the functionality of letting people behind the user know that the user is starting to accelerate after being still for a while (e.g. when a user has their brakes on while stopped at a red light and then stops braking once the light turns green). However, our implementation still serves the more important purpose of indicating to people behind the user that the user is decelerating. As for the switch on the handlebar, if it is toggled to the left or right, it will start blinking the respective turn signal LEDs on the helmet until the switch is toggled back to neutral. Our current design could not get the radio module working properly, so we could not have both boards communicate with each other. But to show that the LED turn signals work as intended, if the user were to place their hand close to the left ultrasonic sensor, the left turn signal would turn on. Meanwhile, if they place their hand close to the back ultrasonic sensor (as the right does not work), then the right turn signal will turn on. Of course, this is not how we intend to have users do their signaling, but it was a solution that was implemented for demonstration purposes.

If the user ever gets in an accident and they experience a force greater than a pre-set value, the helmet will enter its "emergency state" mode. In this mode, both vibrators will be constantly vibrating, and the brake lights (red LED) will be constantly flashing. It will not stop until the user turns the helmet board off through the on-off switch. If the user doesn't turn the helmet off after 10 seconds of the "emergency state" mode starting, the board will start to send an SMS message to the emergency contact. The SMS message will contain the GPS location of the helmet. A message will be sent every 10 seconds and will be sent continuously until the helmet is turned off. This is to help the user in the case that they are, for example, knocked off of the road and rolling down a long cliff and cannot call for help themselves.

7. Conclusion

The goal of this project was to create a smart helmet that could enhance the safety and enjoyability of biking or travel on light vehicles, especially because Los Angeles has an abundance of bikers and consequently bike-related injuries.

Our approach to creating a smart safety helmet was to include a brake light and turn signal lights on the helmet, methods to detect and warn a user of approaching objects, and an emergency unit that would send an SOS message with the GPS coordinates of the user if the user got in an accident. Once we had our general design, we started looking for parts that would help us implement our design. The main parts we ended up choosing were three ultrasonic sensors (for detecting objects coming from the left, right, or behind), an accelerometer (for determining if the user is decelerating or if a user has crashed/ been subject to a great force), a GSM (for sending text messages in case of emergency), a GPS module (to get the GPS coordinates of a user in case of emergency), two radio transceivers (for communication between the handlebar switch and the helmet for turn signal functionality), the ATMega328p microcontroller, and brake lights and turn signal lights. Of course, many other parts, such as boost converters, voltage regulators, and MOSFETs, were needed for all the different electronic components to work together and be powered correctly.

The different components were soldered to a board and necessary connections between parts were made by wire-wrapping or soldering. We looked at datasheets to make the correct connections between input and output pins as well as add any necessary passive or active components for correct operation.

On the software side, a USBtinyISP was used to program the ATMega328p, and many debugging steps were taken to make sure that data could be read correctly from the sensors over the required communication protocols, and that the motors and light signals worked as desired. Much software debugging was done through printing values using the serial communications interface on the ATMega328p and monitoring those values on a computer through the use of a TTL to RS-232 converter and a DB-9 connector.

Debugging of the device was complex due to the interconnection of hardware and software, but tools such as an oscilloscope and a multimeter were helpful in allowing us to see what was happening in the hardware and why the hardware sometimes did not do what the software was telling it to do.

The working prototype we built has everything implemented and working from our initial design except for the radio transceiver, which was supposed to allow the switch on the handlebar board to control the turn signals on the helmet board. A temporary solution was implemented where if a user places their hand close to the back ultrasonic sensor (as the right one is faulty), the right turn signal will turn on.

For future improvements, alternatives for sensors can be considered for better accuracy. For example, a LiDAR sensor might be used for more precise distance sensing, and a more advanced GPS could be used that can obtain information on user location even in more challenging environments. For power supply, a solar panel could be used for a more environmentally friendly implementation of our device.

8. References

For code referenced and tutorials used
GSM https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/
Ultrasonic
https://github.com/SmithIsMyName/ATMega328p_HC-SR04_DMD/blob/master/main.c
GPS https://github.com/adafruit/Adafruit_GPS
Accelerometer https://github.com/microbuilder/LPC11U_LPC13U_CodeBase
RF transmitter https://github.com/cristi85/RFM69

9. Appendix

9.1 - Parts and Prices Breakdown

Part Name	Manufacturer	Part Number	Cost	Quantity	Total Cost
Ultrasonic Sensor	Adafruit	HC-HR04	$0.30	3	$0.90
Accelerometer	Adafruit	LIS3DH	$4.95	1	$4.95
Radio Module	Adafruit	RFM69HCW 433MHz	$9.95	2	$19.90
SOS Chip Module	HiLetgo	SIM800L	$8.99	1	$8.99
12V Boost Converter	KOOBOOK	DC to DC Boost Converter	$7.21	1	$7.21
5V Regulator	Sparkfun	L7805	$1.05	1	$1.05
Vibrator	Miskall	Vibration Motor Module x3	$9.89	1	$9.89
GPS	Adafruit	PA1010D	$29.95	1	$29.95
Turn Signal	Leadtops	33 SMD LED Arrow	$13.99	1	$13.99
Brake Signal	HYADA		$9.95	1	$9.95
MOSFETs	Onsemi	2N7000-D75Z	$0.40	3	$1.20
AA Battery Holder	Keystone	2477	$1.87	2	$3.74
3 pin toggle switch			$0.50	1	$0.50
SIM Card(T mobile)			$61.47	1	$61.47
Buck Connector		LM2596S	$2.40	1	$2.40

9.2 Code

All of our code can be found at https://github.com/pacchen/SmartBikeHelmet

Signature Sheet

	Michelle Liang	Paul Chen	Joshuah Chen
System design	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$
Component selection	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$
Hardware design	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$
Software design	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$
Documentation	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$
Project report (oral)	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$
Project report (written)	$33.\overline{3}\%$	$33.\overline{3}\%$	$33.\overline{3}\%$

We split everything pretty much evenly.

Michelle Liang

Paul Chen

Joshuah Chen