Huddy Buddy: The Smart Heads-Up Display for Increased Driver Awareness and Safety

Logan Glowth, Pedrhom Nafisi, Evan Hall, and Aaron Majdali

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The Huddy Buddy is designed to increase driver awareness and safety while driving a vehicle. This is accomplished by providing the driver with information about the vehicle's speed, fuel economy, and navigation data to a collimated display located in the driver's point of view. This enable the driver to see critical information while remaining focused on the surrounding objects. In the unlikely event of an accident, the Huddy Buddy has an onboard emergency response system to lower the response time of emergency services.**

***Index Terms* — Collimated Display, Crash Detection, Heads-Up Display, OBD II, OLED Display**

I. Introduction

In today's world, where mobile technology has become an essential part of our lives, it is often difficult to disconnect and put these mobile devices away for much longer than a few minutes. This becomes a major hazard when getting behind the wheel of a vehicle. There is a myriad of distractions to consider when driving a vehicle. According to the National Highway Traffic Safety Association, "distracted driving is dangerous, claiming 3166 lives in 2017 alone". Distracted driving not only puts the driver's life in danger, but also the lives of the other drivers and passengers on the road, creating dangers such as speeding. The overarching goal for this project is to have created a heads-up display (HUD) that can display content relevant to someone driving a vehicle in a manner that is safe, efficient, and user friendly. The Huddy Buddy is created to fit into any vehicle regardless of what design the interior has. This HUD will use the on-board diagnostic II (OBD-II) to supply the ATMega 2560 with data from the car using the parameter identifier (PID) which allows the HUD to use digital trouble codes (DTC) to access data to display on the screen. There is also a mobile Android application that is developed to work with the HUD. It is a functional navigation aid to ensure the driver has a consistent and beneficial experience while using the system. From the application, a destination can be specified to start a route. In the event of a crash an emergency contact will be stored in the application. That emergency contact is linked to the crash detection system in the HUD unit. The accelerometer that is in the HUD will detect if there is a large g-force detected and if over the threshold it will take the current location of the HUD and then transmit the coordinates to the emergency contact via SMS text message. This capability comes from the SIM808 module that is incorporated into this design in order to grab current GPS coordinates and to also be able to send out SMS text messages. The data to be displayed will be sent to an LCD screen which will go through two different lenses to focus the image at infinity. This means that there will be no need to focus on the HUD or on the road, the driver can focus on both without compromising attention to the road.

II. System Components

The Huddy Buddy heads-up display system can be best described by the individual components that make up the entire system itself. Each subsystem operates to combine their functionalities into one final product that can be packaged together to create the Huddy Buddy. Below is an introduction to each of the components that make up the Huddy Buddy.

A. On-Board Diagnostics II

In 1991, the California Air Resources Board (CARB) began to push vehicle manufacturers to make new vehicles that included Onboard Diagnostics for emission control. The Onboard Diagnostics would provide information about how much fuel the vehicle was using, and the types of emissions were being produced as a result of driving. The OBD-II was chosen not only for its convenience in sending and receiving data to and from the car, but also because it can power the entire system from a regulated 5-volt, 2.1-amp power supply that is already incorporated into the device. The DTC is an efficient way to take in the data that was chosen to be displayed, such as speed read out and fuel consumption rate. The vehicle's speed is read by accessing the PID of value 0x0D which returns a 1-byte value giving the exact speed in kilometers per hour which is converted to miles per hour. The fuel consumption rate is based on the following equation that is derived from the ideal ratio of air to gasoline; the mass air flow (MAF) sensor is used to help calculate this.

$$GPH_i = \frac{MAF\ gram/sec}{14.7\ gram/sec} \times \frac{1\ lb}{454\ gram} \times \frac{1\ gal}{6.701\ lb} \times 3600\ sec \quad (1)$$

$$MPG_i = \frac{Speed}{GPH_i} \quad (2)$$

B. LM2577-ADJ

The optimal voltage input to the ATMega2560 is between 7 –12 volts. The LM2577-ADJ is set to step up 5 volts supplied from the OBD-II to 9 volts. This is achieved with the equation below. Also, in Fig. 1, shows how this was implemented.

$$V_{out} = 1.23V\left(1 + \frac{R_1}{R_2}\right) \tag{3}$$

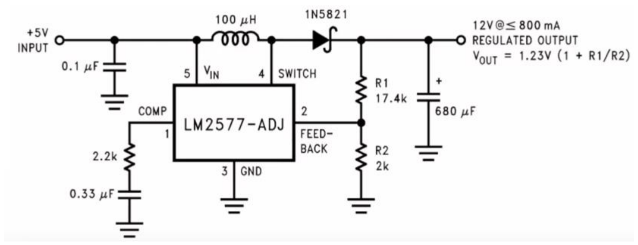


Fig. 1. Schematic of LM2577-ADJ showing how to use R_1 and R_2 to alter the output voltage.

C. Backup Battery System

The backup up battery system is comprised of two Samsung 25R 18650 2500mAh 20A batteries capable of suppling 3.6 volts each for a combined 7.2 volts. The set up for this is as follows, while the stepped up 9 volts is flowing a blocking diode keeps the 7.2 volts from powering anything keeping it charged until needed. In the event of a crash and the main power is lost, the diode will allow the power from the battery into the 5 volt regulator and continue function for a text message allowing the coordinates of the crash to be sent to the emergency contact. It works as shown in Fig. 2 below.

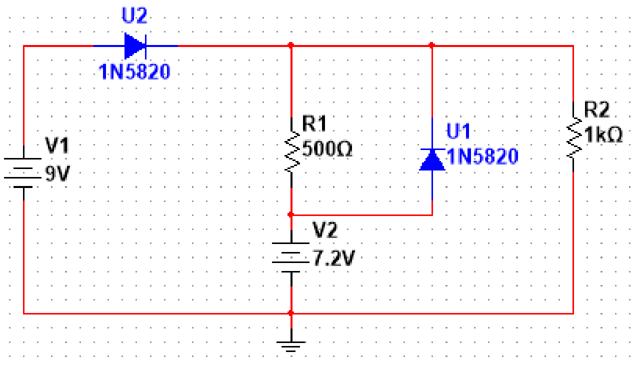


Fig. 2. Here it shows how to the backup battery is supposed to take over the main power supply if it fails.

D. ATMega 2560

The ATMega 2560 microcontroller was chosen for this project as it provides a lot of functionality while maintaining low cost and high efficiency. The hardware required to implement the ATMega 2560 is open source and readily available, giving the ability to create a custom implementation without infringing on copyrighted or patented designs. Additionally, this microprocessor is well known for its flexibility and portability when applied to embedded designs like our heads-up display. The ATMega 2560 supports a wide array of protocols including UART, I2C, and other communication tools that enable the microcontroller to be used with a myriad of additional components. To support this large number of available components, the ATMega 2560 is capable of powering 54 digital I/O pins, 15 of which provide PWM output, as well as 16 analog input pins. It operates at 7-12 volts, enabling a small boost converter to allow devices such as our OBD-II connection to power the device and its components. The specifications for the microcontroller are given in Table 1 below.

Processor Speed	16 MHz
Data Bus Bandwidth	8-bit
RAM	8 KB
Flash Memory	256 KB
UART Channels	4
I/O Pins	54
Operating Voltage	5 Volts
Special Features	Open Source

Table 1. Summary of Capabilities of ATMega 2560

E. SIMCOM SIM808 GSM/GPRS Module

The SIMCOM SIM808 is a Quad-Band GSM/GPRS module that enables communication with cellular and satellite networks. The SIM808 implements functionality of SMS test messaging, mobile calling, and GPS Navigation. We chose the SIM808 module for its robust functionality at a low overall cost to implement. The SIM808 module uses the AT Command set, which provides a standardized function model to send and receive data over the cellular and GPS networks. The objective for the SIM808 module in our project was to provide an emergency response system in the event of an accident. When the HUD is powered on, the microcontroller sends AT Commands to the SIM808 to initialize itself on the GPS network with an attached external active GPS antenna, pairing it to geosynchronous satellites in orbit above the device. In the event of an accident, the microcontroller sends more AT

Commands to the SIM808 to grab its current location from the GPS satellites, queue up an SMS text message to a designated emergency contact containing the previously obtained coordinates, and then sending the text message over the cellular network using the attached GSM antenna. This process takes just a few seconds to complete, keeping the overall time for response at a minimum and increasing the likelihood for those affected by the crash to get the care they need.

F. ADXL 335 Accelerometer

The Analog Devices ADXL 335 Accelerometer is a triple-axis accelerometer that comes with pinouts for x, y and z axis measurements. It has an onboard voltage regulator that steps the voltage down from 5 volts to 3.3 volts, allowing a large range of components to power the device. The ADXL 335 can measure forces with up to ±3g of sensitivity, enabling the detection of large changes in forces such as in a car accident. Our emergency response system is a vital part of ensuring the safety the driver and passengers that are using the Huddy Buddy. As such, it is imperative that the design of the Huddy Buddy be durable enough to withstand the sudden forces accompanied in a car accident. The ADXL 335 is a great solution for measuring sudden forces as it can withstand forces up to 10,000 g's, increasing the likelihood for it to survive and produce accurate outputs during a car accident.

G. HC-05 Bluetooth Module

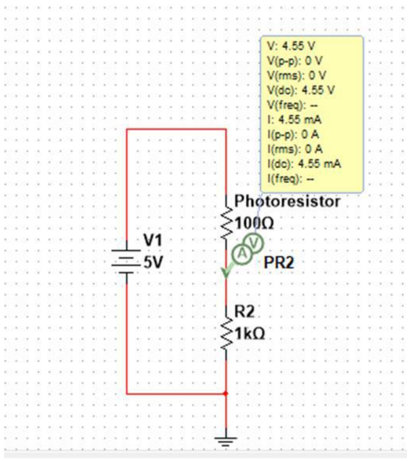
The HC-05 Bluetooth Module is a serial module that is setup in slave mode in order to transfer data packets between the user's phone with our application on it and our HUD. It uses wireless protocol Bluetooth V2.0+EDR which is a balance between Bluetooth Low Energy and new Bluetooth protocols in terms of performance and hardware requirements. The module is developed and tested in an Arduino environment and is set to receive latitude and longitude coordinates, as well as a distance value. This information is provided by the application which implements Mapbox APIs in order to do routing and resultant output is sent to be received by this module.

H. Grove OLED Display

The Grove OLED is an external OLED display that outputs a monochrome image at a 128x64 pixel resolution. We chose this display as it has adequate space to display the information we wish to provide to the driver, while maintaining readability at a high contrast and brightness levels. Additionally, the Grove OLED display implements data transfer over the Inter-Integrated Circuit (I2C) protocol allowing our ATMega 2560 microcontroller to drive the display.

I. Light Sensor

The idea of having a light sensor is to have a way to dim and brighten the LCD screen. A little bit of code makes the Arduino take in analog data from the photodiode, which is the Adafruit 161 photodiode, and convert the voltage to a lux value. This lux value then determines how bright the display will be. This is shown in Fig. 3 and in Table 2



below.

Fig. 3. Demonstration of how a voltage divider with a photodiode is used to create analog voltage values for input into the ATMega 2560.

Light is Like	Lux Value	Approximate Photoresistor Value	Photoresistor + Pulldown Resistor	Voltage (Volts)
Moon-light	1	70 kΩ	71 kΩ	0.07
Dark Room	10	10 kΩ	11 kΩ	0.45
Bright Room	100	1.5 kΩ	2.5 kΩ	2
Over-cast Day	1000	300 Ω	1.3 kΩ	3.85
Day-light	10000	100 Ω	1.1 kΩ	4.55

Table 2. Determining Lux Values with Photoresistor

J. Collimating Lenses and Mirror Unit

The lenses that we use are a 74.5mm diameter positive lens with a focal length of 100mm in conjunction with a 49.5mm diameter negative lens with a focal length of 100mm. Separating the two is a 76.5mm square mirror positioned at a 45-degree angle to direct the image from the negative lens up to the positive lens. The display is positioned 49mm away from the negative lens, which produces a virtual image that is 100mm away from the positive lens. The distances listed were determined by calculating the intersection of two rays. With the object of a certain height and a lens with a certain focal length, we were able to calculate where the ray from the height of the object to the center of the lens would intersect with a ray that would go from the lens at the height of the object to the focal point on the optical axis. We were then able to adjust the object distance to best match the size constraints of our system as well as produce an acceptable image. By utilizing the negative and positive lenses, the projected image has much less distortion than if only the positive lens were used.

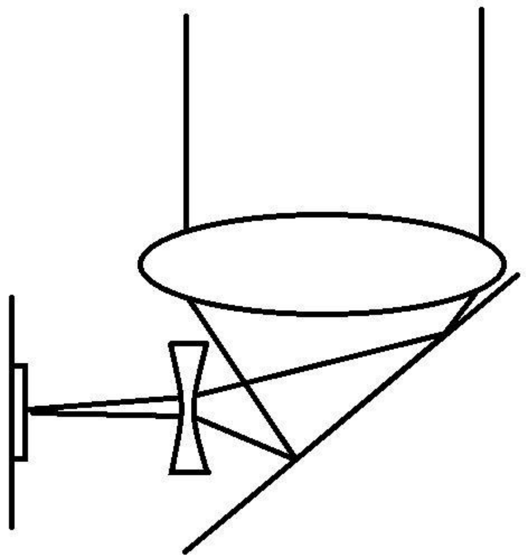


Fig. 4. The path of rays emanating from a point on the OLED display as they pass through the two lenses and reflect from the mirror. Note how the rays are parallel after leaving the larger positive lens, thus creating a collimated image.

In Figure 4 above, we see that the rays from a point on the OLED display will diverge once they are refracted by the negative lens. The diverging rays will be reflected off the mirror and travel up to the positive lens. The positive lens will then refract the rays to be parallel to one another. The purpose of including the smaller negative lens is to create a virtual image of the OLED that will be positioned at the focal point of the larger positive lens. By including the second lens, we are able to reduce distortion and create an overall clearer image for the viewer.

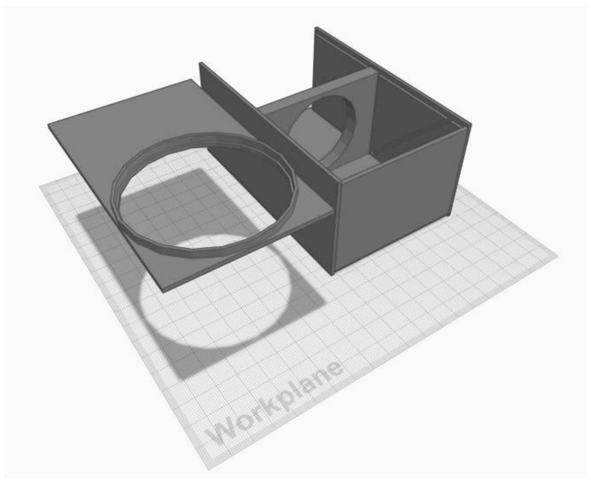


Fig. 5. The enclosure for the mirror, lenses, and OLED display. The backing that would hold the display has been left out to better see the piece holding the negative lens. The lid is shown removed to better see the inside of the enclosure. Note the angled cutout for the mirror.

Figure 5 above shows a 3D model of the planned enclosure for the OLED display, the two lenses, and the mirror. This design will consist of six separate pieces that will be held together with an adhesive. The spacing of the components have been determined by physical constraints as well as ray tracing to determine the optimal positioning of the lenses and the display. Ideally, this enclosure would be made with sheets of a material like Delrin. Due to current events, the pieces will be 3D printed as 3D printing is most easily accessible at this time.

K. Android Application

In order to help support our HUD with routing and contact information, an Android application was developed in Java to allow the user to input critical information that allows the HUD to reach its potential. Features include a Mapbox based mapping system that allows the user to input destinations with auto complete as well as the saving of prior and newly inputted addresses. Once the user has chosen a destination, the application will calculate a route based on longitude and latitude coordinates where points of interest is sent to the HUD as well as whether this POI needs a turn, a merge, or a straight. The android application feeds the HUD waypoint information one POI at a time in order to minimize the need of constant data transfer. The app also lets the user check the connection status between the HUD and phone as well as to set their emergency contact for the HUD's real time crash services

III. TESTING RESULTS

After completing our initial research for how we could accomplish the goals set out in Senior Design I, we began to acquire the components above in Section II to implement each subsystem. Each component is modular, enabling us to test one component at a time. Initially, we obtained our Arduino Mega 2560 development environment, SIM808 Cellular GPS Module, OBD-II UART adapter, and OLED Display.

A. Arduino Mega 2560 Development Environment

To implement the microcontroller that we wished to use for the Huddy Buddy, we began testing on the Arduino Mega 2560. Arduino provides an open source schematic for the development environment that the ATMega 2560 needs to work. This was ideal as it gave us a baseline design for building our own custom PCB with similar functionality. Additionally, there are many tutorials available online for implementing embedded projects similar to ours using the Arduino Mega 2560 environment.

B. SIMCOM SIM808 GSM/GPRS Module

We began testing the SIM808 by connecting it to the Arduino Mega 2560 using the TX/RX pins on the breakout module. This allows communication between the Mega 2560 and the breakout over the UART protocol. The AT command set needed to send and receive data to the SIM808 module was done by initiating a serial connection with the UART and sending each AT command in succession, then reading the data that is returned by the SIM808. We were able to successfully send an SMS text message to a desired phone number over a cellular network. After initial testing of the GPS functionality, we encountered an anomaly where the SIM808 module would shut down when attempting to make a connection with either the cellular or GPS network. Upon further research, we discovered that the SIM808 can draw up to 3 amps of current when operating. If this current target is not met, the module will shut down. To mitigate this issue, we created a separate power system for the SIM808 module that is comprised of two 18650 lithium polymer batteries in series, producing 7.2 volts and has a continuous discharge rating up to 20 amps. Once the external power system was connected to the SIM808 module, we were able to reliably communicate with the SMS and GPS networks successfully.

C. *Analog Devices ADXL 335 Accelerometer*

To ensure that the ADXL 335 was providing accurate data to the Mega 2560, we attached the accelerometer to the microcontroller environment with the analog inputs and created a serial program to read in the data provided. We then began to measure the baseline data to establish normal g-forces when no force is applied. We then measured how this data changed when quick changes in force were applied to the accelerometer. We determined that an adequate force to demonstrate the abilities of the accelerometer would be 1.5 g's of force in either the x, y, or z axes. This can be reprogrammed to a higher threshold to eliminate false positives if the driver is on a bumpy road.

D. Grove OLED Display

In order to output data from the Mega 2560 to the OLED display, the data needs to be formatted correctly to turn on the pixels of the display to produce the desired image. Upon further research of display libraries, we discovered an open-source library that is compatible with the SSD1306 chipset that the Grove OLED is driven by. The u8g2 graphics display library created by olikraus on GitHub contains prebuilt functions to print and display graphics and text onto the OLED display. These functions enabled us to provide the driver with information in a method that is both user friendly and efficient to display and update. To create a display frame, the programmer sets a desired coordinate to print accompanied by the information that is to be printed. Once the frame has been completely built, it is sent to the display over the I2C bus on the microcontroller. Another advantage of the u8g2 library is the ability to mirror the display, allowing us to maintain an image that is correct for the orientation of reflecting the display off the vehicle's windshield. Other advanced functions of the display include many different fonts, scrolling text, shapes, and images.

E. On-Board Diagnostics II

Our research into OBD-II adapters for Arduino modules led us to obtaining the Freematics OBD-II UART adapter. This adapter can be connected the Mega 2560's Serial1 port to communicate with the vehicle. Freematics provides documentation and libraries for initializing connection between the vehicle, the adapter, and the Mega 2560. Once we were able to obtain the connection, we could analyze data like speed and battery voltage straight form the vehicle. This data could be easily stored as a variable, which is then sent directly to the Grove OLED Display. Since each frame is built individually, we can tune the refresh rate for both obtaining the data from the OBD-II and displaying the data on the OLED Display. We found that a 200ms refresh rate was adequate for providing the user with accurate information while maintaining smooth operation.

F. Collimating Lenses and Mirror Unit

In order to develop a proper heads up display we needed the produced image to be collimated, or focused at infinity. With a monochromatic point source, collimating is as simple as placing the object at the focal length of a positive lens. Initial testing with a positive lens gave a noticeable amount of distortion, such that the image was only somewhat visible if viewed head-on. With ray-tracing equations we determined that a negative lens placed in between the display and positive lens would give us a smaller virtual image that would act as the "display" for the positive lens. This allowed us to generate a projected image with much less distortion, at the cost of increased weight, expense, and size. By arranging the elements together, in conjunction with a sheet of plexiglass acting as the "windshield", we found that the ideal image is produced when the negative lens is roughly halfway between the positive lens and the display. Our original test with the negative lens a mere 30mm away from the display resulted in a distorted projected image that was not much better than the projected image from just the positive lens. However, with the correct spacing we were able to produce a clear image with little distortion. While this was done with the elements on the same optical axis, the results should be closely matched by a system incorporating the elements as well as a mirror between the two lenses.

G. Custom Printed Circuit Board (PCB)

A huge thanks is needed to be given to Quality Manufacturing Services, as they helped put the parts onto the PCB. A lot of practice went into learning how to solder, but they helped put the smaller, and harder pieces on the PCB. Once assembled testing began on the board and several issues arose. The biggest one of all was the 16 MHz crystal that the ATMega 2560 uses. In order to place code onto the ATMega 2560 via the In Circuit Serial Programming (ICSP), two 22 pF capacitors were needed to be placed on the ends of the crystal. To test if programming was going to be an issue an initial test a simple one second incremental counter was used, and then displayed onto the OLED screen. While running the code each individual second took approximately 20 seconds between increments showing that the crystal that is used as a clock signal was giving an inaccurate signal. It was soon found that there is an internal clock in the ATMega 2560 that can be used as a clock signal which solved the timing issue. Another problem was within the PCB itself. With the first prototype there were to footprints that were wrong such as the pins for the ICSP, and the footprint for the MIC29302WU had wrong pin sizes so wires were needed to properly connect the part on. Another lesson learned was to not always trust the schematic because new parts are needed to replace old ones and they have different attributes. An example is the status LEDs that were used would not turn on simply because of a resistor value didn't allow enough current to flow through to the LEDs so a new value was calculated. Another revision of the PCB fixed those problems.

H. HC-05 Bluetooth Module

Initial testing of the HC-05 involved hardware implementation and the usage of an Arduino Nano in order to send sketches to the HC-05. Upon further examination and testing, it was discovered a 1kΩ and 2kΩ resistor voltage divider was needed for the RX pin for proper voltage to be delivered to it in order to function properly. Once the development circuit was implemented properly, the LED built into the HC-05 would flash multiple times a second in order to show that it was ready for connections as a slave or a master. Tera Term is a free open source SSH terminal software that was used for initial testing of the module when connected to a laptop. To confirm the validity of data sent and received, a string data packet with the words "Communication Test" was pushed to the module via Bluetooth and then read back to the laptop via serial hookup of the Arduino Nano. Once the string was confirmed to be both sent and received properly, code was written so that the module knows the kinds of data that it will receive and how to handle it, from coordinates to street names to a phone number.

I. Android Application

Research for what navigation service should be used was initially narrowed down to Google Maps and Mapbox. Upon examination of the terms and conditions of Google Maps, a vehicular heads up display was against their terms of use and so the group decided to go with Mapbox. Mapbox is free and open source with a large amount of documentation on how to use their APIs provided straight from their website. The language of choice was narrowed between Java and Kotlin, and ultimately Java was used due to the group's prior knowledge of Java, although Kotlin is just as viable of a language. Initial testing included simply getting the Mapbox map interface to show up and have access to a phone's location services in order to find an address of choice. The second round of testing included route generation and how the code generated coordinates to be stored in an array to be sent off to the HUD. Different data structures were tested in order to find what was optimal in terms of bandwidth and spatial locality for organizing, storing, and sending all the data needed. Once coordinate generation was tested, we moved on to testing the app's ability to send coordinate data to the HUD via Bluetooth from the phone to the HC-05 module. Upon first time running of the app, an emergency contact phone number must also be sent via Bluetooth to be interpreted and prepared by the ATMega and finally sent off to the SIM808 to send the text message to that number when the accelerometer detects a crash. Final testing included moving the app environment from the computer's Android Studio IDE to an actual phone in order to make sure the interface looked good and that all the features work as intended.

J. How the System Works Together

Shown in Fig. 6, the system is linear and straightforward in operation. While most of the functions can be initiated once per trip, functions like navigation and speed readouts are constantly running from when they are activated to the journey's end.

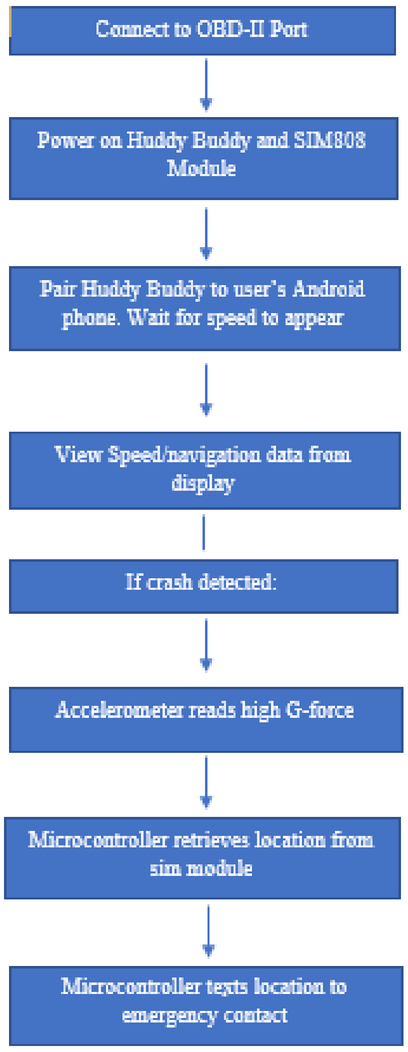

Connect to OBD-II Port
Power on Huddy Buddy and SIM808 Module
Pair Huddy Buddy to user's Android phone. Wait for speed to appear
View Speed/navigation data from display
If crash detected:
Accelerometer reads high G-force
Microcontroller retrieves location from sim module
Microcontroller texts location to emergency contact

Fig. 6. System flowchart shows the general order and flow of the major functions of the heads-up display.

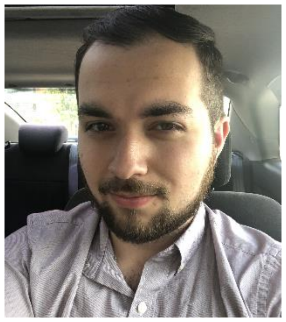
IV. CONCLUSION

We believe that the Huddy Buddy is a product that enables drivers to increase their awareness while reducing distractions when behind the wheel of a vehicle. Our Heads-Up display unit provides the driver with pertinent vehicle statistics and navigational information in real time. Our application is user friendly and allows for a wide range of navigation possibilities. The emergency response system reduces the response time for first responders in the event of an accident. The sections above demonstrate how we were able to realize such a system.

V. BIOGRAPHIES

Logan Glowth is a Senior at the University of Central Florida and will graduate with a Bachelor of Science in Computer Engineering in May 2020. While attending UCF, Logan worked as a Technical Intern at Ellucian beginning in 2017. Logan has accepted an offer from L3Harris to become a Software Engineer in June 2020.

Aaron Majdali is a Senior at the University of Central Florida and will graduate with a Bachelor of Science in Optics and Photonics Engineering in May 2020. He is pursuing a career in the field of optics with the intent to continue studies in the future.



Pedrhom Nafisi is a Senior at University of Central Florida and will graduate with a Bachelor of Science in Computer Engineering in May 2020. In the summer of 2019, Pedrhom attended an internship at Texas Instruments in Dallas, Texas and accepted the job offer given to be a Product Engineer beginning in June 2020.

Evan Hall is a Senior at the University of Central Florida and will graduate with a Bachelor of Science in Electrical Engineering in May 2020. He is pursuing a career in the field of power systems with the intent to continue studies in the future.

VI. REFERENCES

"U Drive. U Text. U Pay." NHTSA, 8 May 2019, https://www.nhtsa.gov/risky-driving/distracted-driving.

"olikraus/u8g2." https://github.com/olikraus/u8g2

"stanelyhuangyc/ArduinoOBD"
https://github.com/stanleyhuangyc/ArduinoOBD