Bike Navigation Assistant

By
Akshat Khajuria
Arvind Arunasalam
Saumil Agrawal

Final Report for ECE 445, Senior Design, Spring 2018
TA: Nicholas Ratajczyk

May 2, 2018
Project No. 48

Abstract

Currently, people find it difficult to use navigation applications on their phone and ride a bike at the same time. In this report, the development of a bike navigation assistant is discussed. To use this system, a cyclist only needs to input their destination on a mobile application. The mobile application then communicates with LEDs and a beeper on the handlebar of the bike that tell the cyclist where to go. There are two sets of LEDs and if the left LEDs light up, a left turn needs to be made and similarly if the right LEDs light up, a right turn needs to be made. The LEDs start lighting up when the upcoming turn is less than 50 meters away. Additionally, for safety, there are blinkers and the LEDs and the buzzer are used warn the cyclist if there is an obstruction less than five meters away.

Introduction

1.1 Purpose

In today's world, it is quite difficult for people to use bikes in places that they are not very familiar with specific locations and directions. It can be a big hassle to use Google Maps or a similar application on your phone as it results in stopping frequently to check for directions. Moreover, there are numerous safety concerns regarding riding bikes, especially on crowded streets. In 2015 in the United States, there were a over 1000 bike-related deaths and 467,000 injuries [1]. Our project aims to counter the problem of bike navigation as well as improve bike safety by attaching a few modules to the bike.

We wanted to do this project to improve people's experience when riding bikes. In addition to people adding these modules to personal bikes for convenience and safety, this could be hugely beneficial if they are added to rental bikes in big cities. When people travel to new cities, they mostly use taxis to get from one place to another but if rental bikes had these features, they would attract more people. Overall, the bike industry has been pretty static for the past few years [2] and we hope that this will attract more people to the environment friendly practice of riding bikes and result in less bike-related accidents.

1.2 Functionality

The main aim of our project is to make it easier for cyclists to navigate. To do that, we have on bike peripherals, LEDs and a beeper on the handlebar that communicate with a mobile application. The mobile application uses the Google Maps API to receive navigation data. It then sends that to a microcontroller using a Bluetooth module. The LEDs and the beeper indicate to the rider when they are approaching a turn. If the upcoming turn is a left, the left LEDs light up and similarly if a right turn is coming, the right LEDs light up. Furthermore, as the rider gets closer to the turn, more leds light up and the frequency of the buzzer tone gets higher.

Secondly, in order to improve bike safety, we have an ultrasonic sensor at the front of the bike for a warning system. The ultrasonic sensor is used to light up all the LEDs and gives a continuous beeper tone if there is an obstruction less that five meters away from the bike. We also added blinkers at the back of the bike that the rider can control from switches on the handlebar. This eliminates the need for the biker to use their hands to indicate when they are turning and will allow them to keep both their hands on the handle at all times.

Finally, we created a speedometer using a reed switch and a magnet. It serves as a backup when the GPS signals are not strong enough for the navigation component of our project.

1.3 Subsystem Overview

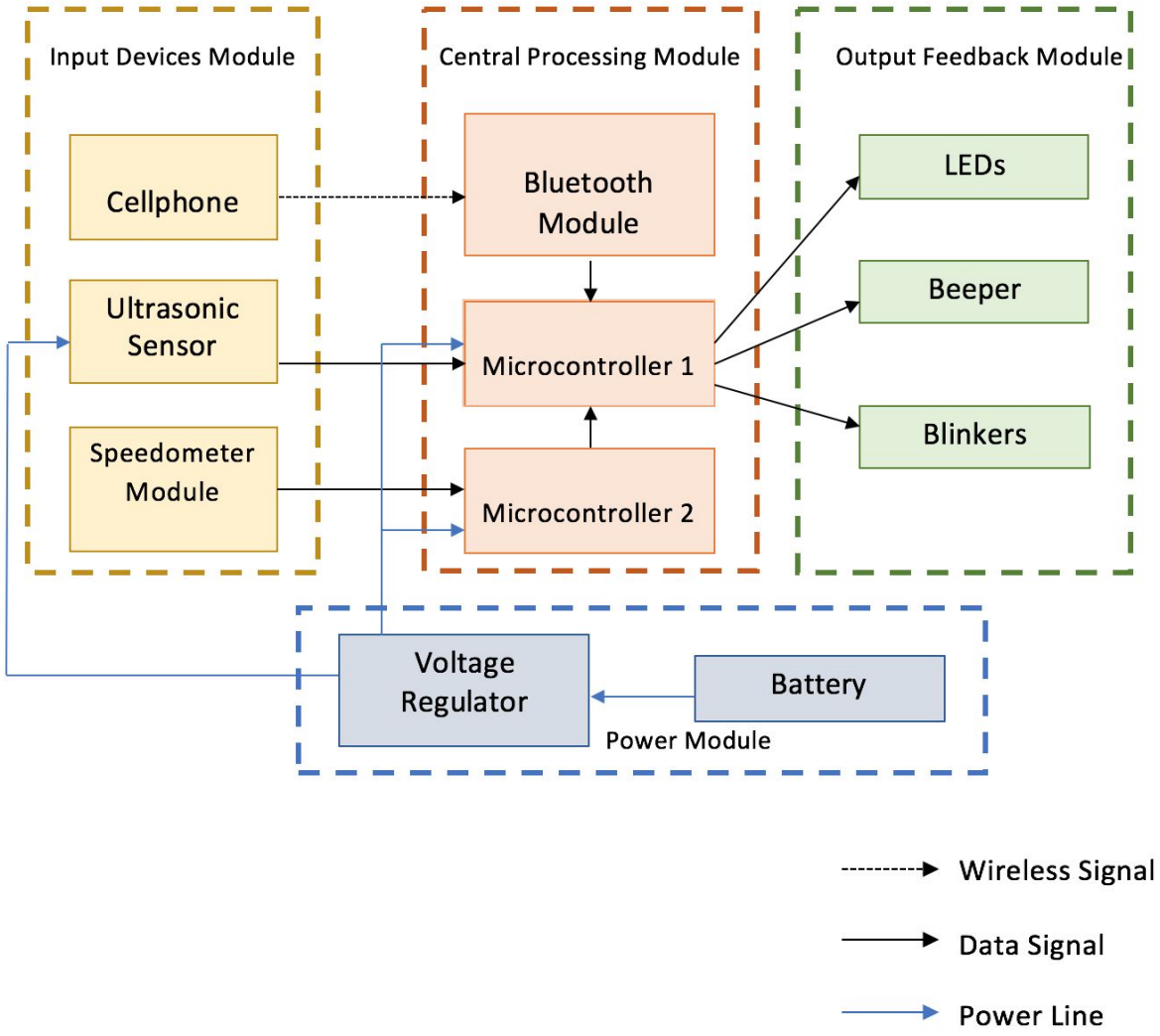

Figure 1: Block Diagram

Figure 1 represents our block diagram and we have four modules. The first one is the input devices module where we have the cellphone that hosts the mobile application, the ultrasonic sensor, and speedometer module that consists of the reed switch and the magnet. Next, we have the central processing module that has the Bluetooth module that send information navigation data from the phone to the first microcontroller. The first microcontroller also takes in information from the ultrasonic sensor and the second microcontroller. The second microcontroller calculates the speed of the bike and sends the distance travelled by the bike to the first microcontroller.The main microcontroller then controls the output feedback module, that includes that LEDs, buzzer and the blinkers. Finally we have a power module that has a battery

and voltage regulator that outputs 5V and powers that microcontrollers and the ultrasonic sensor.

1.4 Physical Design

Figure 2: Physical Design

Figure 2 shows the physical design of our project. At locations marked ‘1’ on the handle of the bike, the microcontrollers, beeper, bluetooth module and the power supply are placed. At location ‘2’, the LEDs are placed. At location marked locations marked ‘3’, the ultrasonic sensor is placed. At location ‘4’, the blinkers are placed. Lastly, at locations marked ‘5’, the speedometer module is placed.

2 Design

2.1 Input Devices Module

2.1.1 Cellphone

The core of this project is a mobile application that we used to interface with the Google Maps Directions API and the bike. We developed an application that will automatically get the users current location, take the user input for destination and then call on the Google Maps API every specified number of seconds to get the most updated trip route. There are four things that either the application does automatically or the user has to do in order for the navigation to start:

1. Bluetooth Connection - The user has to scan and search for a bluetooth device and specifically connect to the HC-05 bluetooth module that is mounted on the microcontroller
2. Current Location - The application gets this automatically every two seconds to ensure constant route updations
3. Destination - The user enters their intended destination
4. Navigate Button - This button is used to start the navigation. Once this button is pressed, every two seconds (until the “Stop” button is pressed):
    a. The most updated current location and user inputted destination are sent to the Google Maps Directions API which returns a JSON file. This JSON file is parsed to generate a string that just contains the distances and another that just contains the directions
    b. The distance and direction strings are sent to the HC-05 bluetooth module

    This constant communication ensures accurate and timely communication about upcoming turns with the biker. This is portrayed in Figure 3**.**

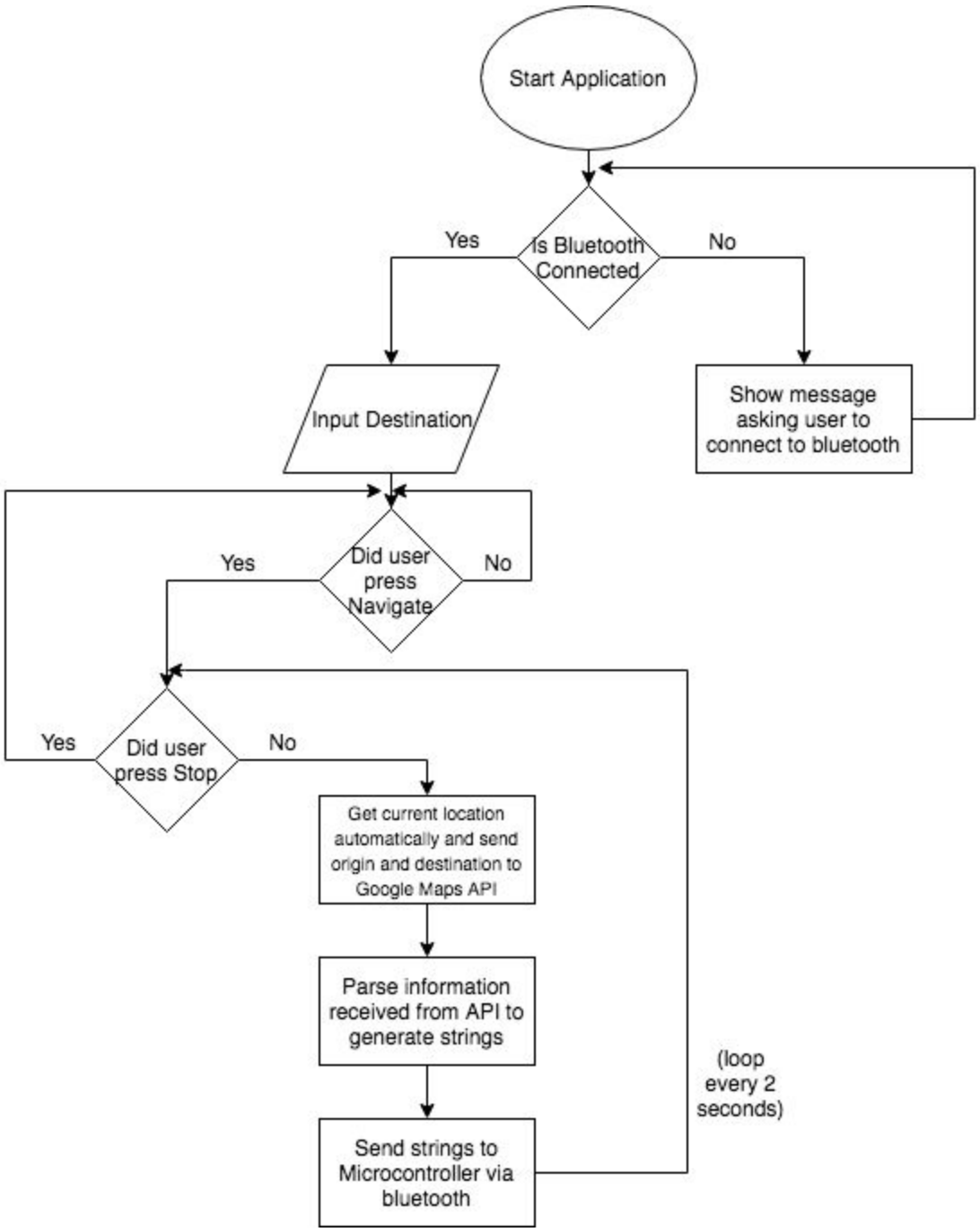

Figure 3. Mobile Application Logic

2.1.2 Ultrasonic Sensor

We used the LV-MaxSonar-EZ2 Ultrasonic Sensor [3] to detect obstacles that posed a risk to the biker. The sensor has an operating range from 15 cm to 645 cm and has three communication interfaces which are the Serial interface, the Analog interface, and the Pulse Width Interface. We used the Pulse Width interface as it was the first among the three to give out a reading and only required a single digital input pin. It works by first pulling the Trig (RX) pin high for at least 20 microseconds. This forces the the sensor to generate sound waves. The PULSE WIDTH pin then sends a HIGH to the microcontroller till it receives the bounced back pulse. We obtain the distance through calculations using the time for which the PULSE WIDTH pin was HIGH. The equation used to calculate the distance from the time taken is shown in Equation 1. If the sensor detects an object within 500 cm, it warns the biker by forcing all the LEDs to go HIGH and the beeper to beep continuously. Snow and rain can affect the obtained values, hence, we chose a sensor designed specifically from outdoor use.

$$Distance\ (cm) = \frac{Time\ (microseconds)}{2 \times 29}$$

Equation 1: Ultrasonic Calculation

2.1.3 Speedometer

The speedometer consists of a magnetic reed switch and a magnet. We placed the reed switch on the bike fork and the magnet on a spoke of the wheel. Whenever the magnet comes close to the read switch, the switch closes the microcontroller reads a HIGH and we know that the wheel completed one revolution. The speed is calculated by measuring the diameter and calculating the circumference of the bike wheel and dividing it by the time taken between two HIGHs that the microcontroller reads as shown in Equation 3. This provides the instantaneous speed of the bike. Keeping a count of the number of revolution also provides us with the total distance the bike has covered as shown in Equation 2. Using this, we could estimate the distance to the next turn and hence this acted as a backup for the navigation component when the GPS signal wasn't strong enough to provide the current location.

$$Distance\ (cm) = Number\ of\ Revolutions \times 2 \times \pi \times Radius\ (cm)$$

Equation 2: Distance Travelled

$$Speed\ (cm/s) = \frac{Number\ of\ Revolutions \times 2 \times \pi \times Radius\ (cm)}{Time\ Taken\ (s)}$$

Equation 3: Bike Speed

2.2 Central Processing Module

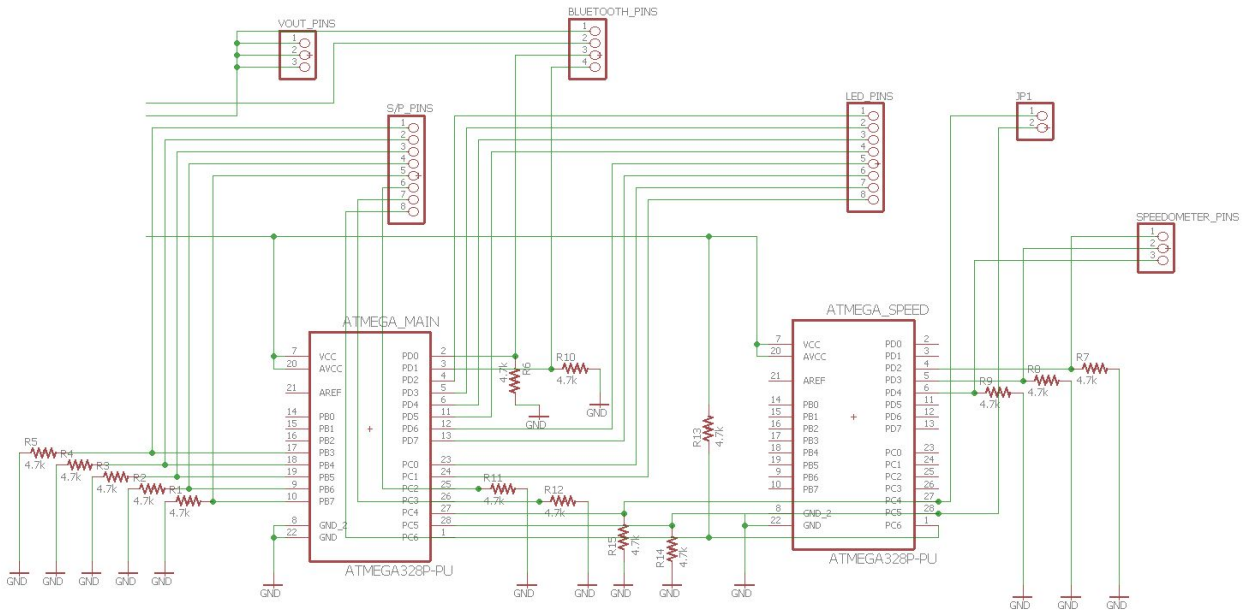

Figure 4. Microcontrollers Schematic

2.2.1 Bluetooth Module

We used the HC-05 bluetooth module [4] to receive the data transmitted from the the mobile phone and send it to the primary microcontroller. The HC-05 was connected to the primary microcontroller using the serial interface. The mobile phone first connects to the bluetooth device and then periodically sends data to it. The bluetooth module receives the data and converts it to serial packets that can be sent to the microcontroller through the TX and RX pins.

2.2.2 Primary Microcontroller

The microcontroller will serve as the core processing unit for our project will be the interface between the mobile application and the devices on the bike. For the purposes of our project, we will be using the ATmega328P - PU [5] since it provides 14 I/O pins and has 32KB of flash storage which will be enough to serve the purpose of our project. Additionally, the microcontroller can be easily interfaced with our bluetooth module and programmed to perform the required work. We mounted the bluetooth module to this microcontroller and this microcontroller is also connected to the secondary microcontroller via the I2C protocol. It performs multiple tasks:

1. Receives data from the mobile application via the bluetooth module and parses this information to generate two arrays for distances and directions.
2. Receives revolution count data from the speedometer and uses that to calculate the distance traveled in a scenario where GPS is not available.

3. Based on data received from the phone and calculations done on data received from speedometer, the microcontroller logic determines if the LEDs and buzzer should turn on. It also contains logic for determining which specific LEDs light up and with what frequency should the buzzer turn on.
4. Receives data from the ultrasonic sensor and in the scenario where the distance to the closest object is less than seven meters, sends a signal to light up all the LEDs and to turn on the buzzer

This logic is demonstrated in Figure 5**.** Figure 14 which is in the appendix is a supplement to Figure 5 and consists of the LED/Buzzer Logic which specifies exactly which LEDs should light up and with what frequency the buzzer should turn on based on the distance to and direction of the next turn.

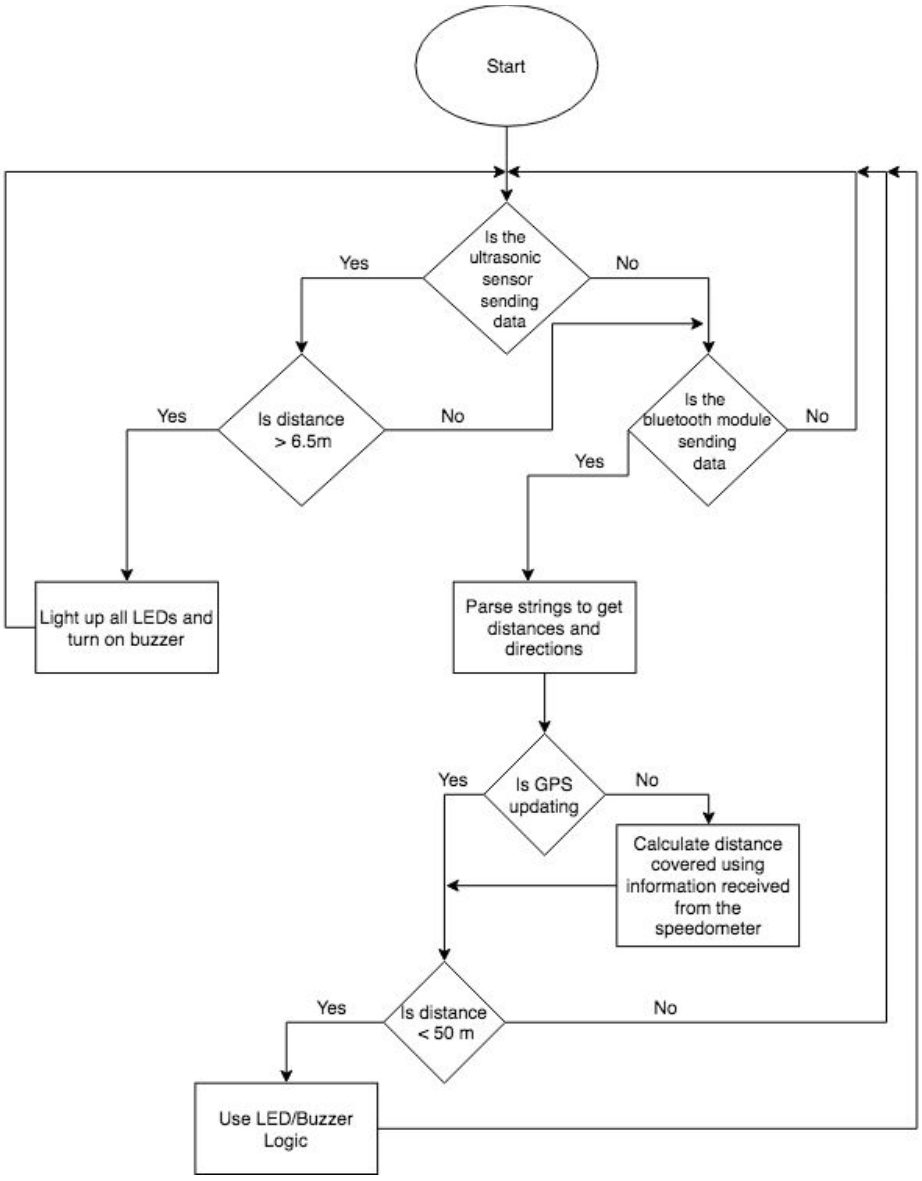

Figure 5. Primary Microcontroller Logic

2.2.3 Secondary Microcontroller

The secondary microcontroller's primary purpose is to interact with the reed switch and calculate the speed of the bike. The microcontroller reads a HIGH every time the reed switch is closed and calculates the speed using the diameter of the bike wheel and time between two wheel revolutions. It also sends the total distance travelled by the bike to the main microcontroller. The logic of the microcontroller can be seen in Figure 6. We decided to use a second microcontroller because we ran out I/O pins on the main microcontroller and we didn't want the speedometer logic to interfere with the main logic as it used a lot of delays. That would have made the calculations of speed inaccurate.

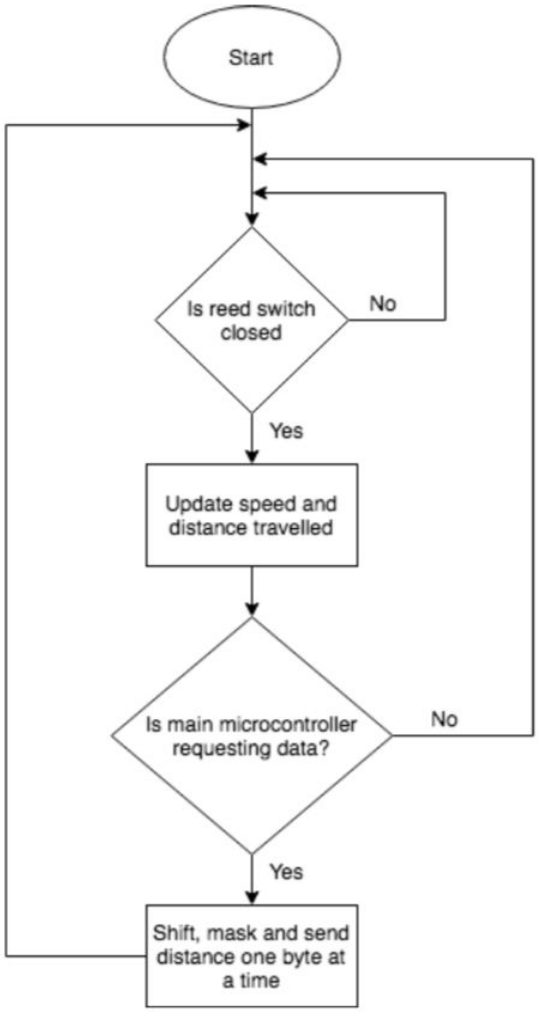

Figure 6. Secondary Microcontroller Logic

2.3 Output Feedback Module

2.3.1 LEDs

There are two sets of 4 LEDs on each side of the handlebar. The LEDs in the direction the turn needs to be made, light up when the turn is 50 meters away. One LED lights up when the turn is in the range of 40-50 meters, two LEDS light up when the turn is 30-40 meters away, three when the turn is 20-30 meters away and all four LEDs light up when the turn is less than 20 meters away. Moreover, all 8 LEDs light up when the ultrasonic sensor detects an object less than 5 meters away.

2.3.2 Beeper

In addition to the LEDs a beeper is attached on the bike. We used the beeper *CEM-1203(42)* [6]. This was used to augment the LEDs in case the rider doesn't notice the LEDs. The beeper has better chance of catching the rider's attention. Like the LEDs, it buzzes when approaching a turn or when there is an obstacle less than 5 meters away. The frequency of the beeper increases as the rider gets closer to a turn and it plays a constant tone if there is an obstruction in front.

2.3.3 Blinkers

To improve bike safety, we attached blinkers at the back of the bike. These are controlled by switches attached on the handlebar. They blink at a constant frequency as long as the rider presses the switches.

2.4 Power Module

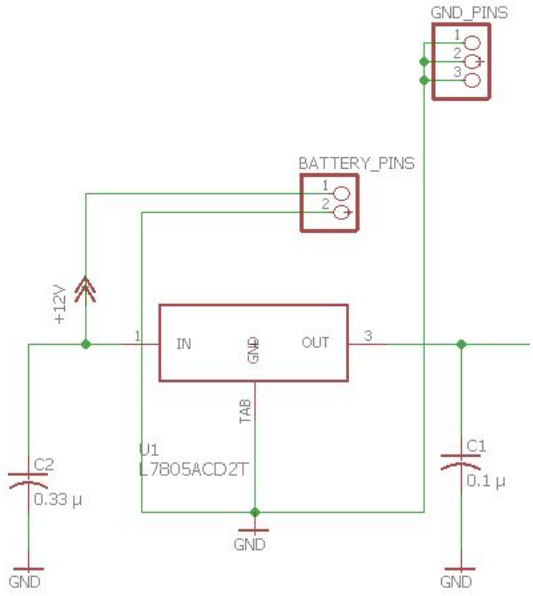

Figure 7. Power Module Schematic

2.4.1 Battery Pack

To power all our electronic components, we used a 9 V/10000mAh battery pack. At full charge, this battery pack would have sufficient power to ensure that all components can function continuously for at least 40 hours as shown in Equation 4. The battery pack has an USB port that is connected to the system through a USB connector which has one end soldered onto the board. When the battery has been drained, the biker can easily remove the pack, and recharge it.

Components	Max Power Consumption	Quantity	Subtotal Power Consumption
Microcontroller	0.2mA	2	0.4mA
Ultrasonic sensor	2mA	1	2mA
Bluetooth Module	50mA	1	50mA
LEDS	15mA	8	120mA

Blinkers LEDs	15mA	2	30mA
Buzzer	35mA	1	35mA
Total Power Consumption			237.4mA

Table 1: Power Requirement of Components

$$Duration\ (h) \ = \ \frac{10000\ mAh}{237.4\ mA} \ = \ 42.12\ h$$

Equation 4: Uptime of System with Fully Charged Battery

2.4.2 Voltage Regulator

All our components require 5 Volts to work but our battery pack supplies 9 Volts. To step down the supply voltage, we used a L7805cv 5V voltage regulator [7]. The potential maximum current requirement of our circuit is around 500 mA which is well within the voltage regulator's limit of 1500 mA. A 0.1μF and a 0.33μF capacitor connected to the output and the input of the voltage regulator to keep the output variance to a minimum. The output is connected directly to the two microcontrollers, the bluetooth module and the ultrasonic sensor. All the other components are powered through the microcontrollers.

3 Design Verification

3.1. Cellphone

In order to verify the application's correct functioning, we checked if the application can call the Google Maps Directions API and show the correct trip route on a test label in the application. We additionally wanted to make sure the latency was not high for which we checked the latency results on the Google Maps Developers Console. For the first, we sent in different origin and destinations to the API via the application and compared the returned distances and directions with the distances and directions obtained when inputting the same origin and destination on http://maps.google.com. The correct result was obtained in 100% of the cases. As shown in Figure ?, we were able to achieve a low latency that improved the applications performance. The median latency we obtained was 31 ms.

3.2 Ultrasonic Sensor

It order to verify that the ultrasonic sensor was functioning properly, we performed 2 different verification methods. The first verification method was done using the Pulse Width interface. We wired the sensor to a microcontroller and connected the microcontroller to a computer. We then placed the sensor at a specific distance from a wall and started to take readings. Using Equation 1, the microcontroller calculated the distance and sent it the computer so that we could see the calculated distances on the serial monitor. We then compared the measured distance with the actual distance and checked if the difference between the 2 values was less than 5cm as shown in Figure 8.

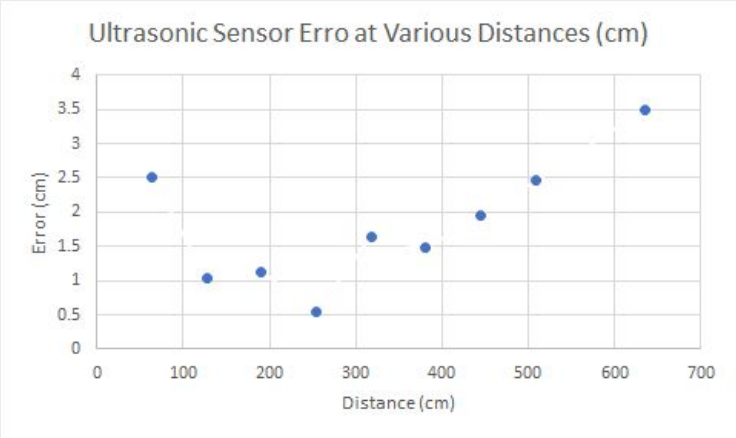

Figure 8: Ultrasonic Sensor Distance Error (cm)

The verification method involved the Analog Interface. We connected TRIG pin of the sensor to a microcontroller and the ANALOG pin to an oscilloscope. We placed the sensor a set distance

away from a wall and started to take readings. We observed the voltage on the oscilloscope and plotted the voltage values against the distances to ensure that the ultrasonic sensors was working within the intended range as shown in Figure 9.

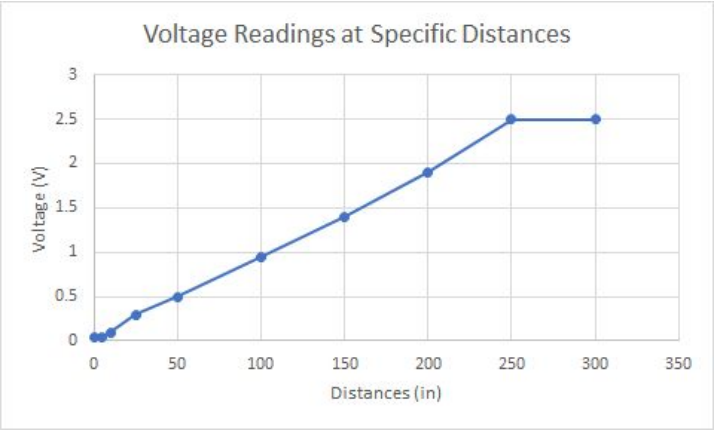

Figure 9: Output Voltage against Distance

3.3 Speedometer

In order to test our setup for the speedometer, we had to verify if the microcontroller detects every time the magnet comes in close proximity to the reed switch. To do this, we spun the wheel of the bike at two different speeds and measured the number of times the microcontroller detected a HIGH.

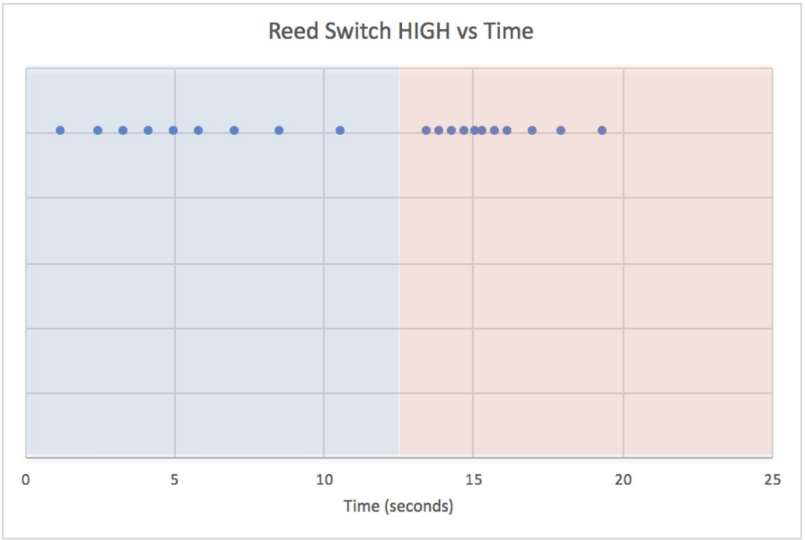

Figure 10: Reed Switch HIGH vs Time

In Figure 10, the first section of the graph shows the data when we spun the wheel at a lesser speed. As can be seen, the HIGHs that are read by the microcontroller are spaced out. In the second section, we spun the wheel faster, and the HIGHs read by the microcontroller are much more concentrated. This clearly shows that the microcontroller was able to detect every single time the magnet crossed the reed switch because otherwise if the microcontroller wasn't detecting some, the HIGHs would not be so concentrated when the wheel was spun at a faster rate.

3.4 Bluetooth Module

In order to verify the correct functioning of the HC-05 module, we connected it to an Arduino to show the strings the module was receiving on a serial monitor. We additionally developed a test application to send test strings of varying length to the bluetooth module. We set a delay of two seconds and measured the latency of sending strings of varying length to the bluetooth module. In Figure 11, we show the latency for a string containing 30 words and in Figure 12 we show the latency for a string consisting of 240 words. We find that there is no major difference between the latency plots and that the overall latency is 2.00 ± 0.26 seconds which is in line with our tolerance analysis from the Design Document.

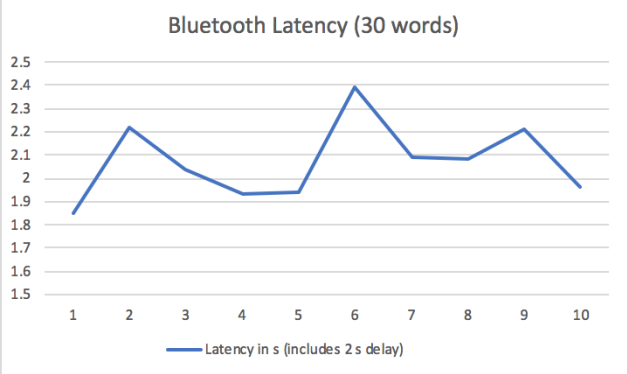

Figure 11: Bluetooth Latency with 30 words

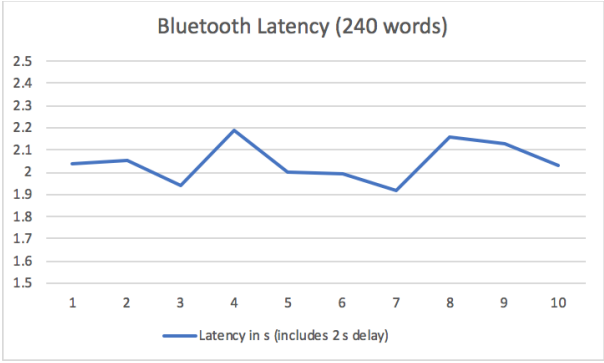

Figure 12: Bluetooth Latency with 240 words

3.5 Microcontroller

To ensure the correct functioning of the microcontroller, we first measured the power and current supply. We obtained an input voltage of 5V and input current of 0.2 mA which met our specified verification. For the second verification, we developed a test mobile application through which we could send test strings to the microcontroller via the bluetooth module to ensure that the correct LEDs light up. We sent different test strings from the mobile application and the correct LEDs lit up 100% of the time with very minimal latency.

3.6 LEDs, Beeper and Blinkers

To ensure that the LEDs and the beeper were functioning as required, we sent test data from the cellphone. This included sending different distances and directions to the microcontroller and checking if the correct LEDs lit up and the correct beeper tone played. We were able to verify this by sending distances in the range of 40-50 meters, 30-40 meters, 20-30 meters and less than 20 meters and checking if one, two, three or four LEDs lit up respectively for both left and right turn. Similarly we checked if the beeper played the correct frequency corresponding to the distance. All the LEDs and the beeper worked as expected. To check the blinkers we attached the switches on the handlebar and made sure the blinkers turned on whenever the switches were pressed.

3.7 Power Supply

To ensure that we were getting a consistent 5V power supply, we used a voltage regulator. We tested it's functioning by providing various input voltages from 0V to 25V. We find that the output voltage from the voltage regulator is 5 ± 0.03372V which falls within our 10% tolerance limit specified in the initial requirements.

4 Costs and Schedule

4.1 Costs

4.1.1 Parts

Part	**Model**	**Unit Cost ($)**	**Quantity**	**Total ($)**
Ultrasonic Sensor	LV-MaxSonar-EZ2	27.95	1	6.00
Microcontroller	ATmega328P-PU-ND	4.30	2	8.60
Bluetooth Module	HC-05	10.95	1	10.95
Beeper	CEM-1203(42)	1.95	1	1.95
Battery	KMASHI 10000mAh Portable Power Bank	13.99	1	13.99
Reed Switch	N/a	1.95	3	5.85
Magnet	N/a	3.95	1	3.95
Switches	N/a	2.95	2	5.90
Various Capacitors, Resistors, LEDs	N/a	N/a	N/a	5.76

Table 2: Parts Cost

The total cost for our parts is $62.95.

4.1.2 Labor

Name	Hours Invested	Hourly Rate	Total Cost = Rate*Hours*2.5
Akshat Khajuria	150	$30	$11250
Arvind Arunasalam	150	$30	$11250
Saumil Agrawal	150	$30	$11250
Total	450	NA	$33750

Table 3: Labor Cost

4.1.3 Total Cost

The total cost for parts and labor is $62.95 + $33750 = $33812.95

4.2 Schedule

Week	Task	Person
2/19/18	Design Doc: Circuit Schematics, Calculations Design Doc: Functional Overview, Requirements, Flow Chart, Conclusion Design Doc: Introduction, Block Diagram, Functional Overview, Requirements	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
2/26/18	Design review, Designed ultrasonic sensor circuit Design review, Researched how to integrate Google Maps API with mobile application Design review, determined how to connect bluetooth module to the application	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
3/5/18	Started PCB layout Determined how to get current location on the app every 2 seconds Researched how to make the speedometer work	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
3/12/18	Developed PCB circuit Started building the mobile application Developed PCB circuit	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
3/19/18	SPRING BREAK	Team
3/26/18	Developed ultrasonic sensor logic for microcontroller Continued working on mobile application Developed speedometer logic for the microcontroller	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
4/2/18	Developed parsing logic to get distances and directions Debugged the application Developed I2C code to send distance to main microcontroller	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
4/9/18	Integrated all the code and developed logic to use distance instead of GPS	Arvind Arunasalam

	Worked on storing all distances and directions on the microcontroller Worked on code to choose GPS or distance for navigation	Akshat Khajuria Saumil Agrawal
4/16/18	Wired the bike and tested the perf board Wired the bike and tested the application and bluetooth module Wired the bike and tested sensors and output peripherals	Arvind Arunasalam Akshat Khajuria Saumil Agrawal
4/23/18	Worked on final paper and presentation Worked on final paper and presentation Worked on final paper and presentation	Arvind Arunasalam Akshat Khajuria Saumil Agrawal

Table 4: Schedule

5 Conclusion

5.1 Accomplishments

We were able to successfully design and build a system which can correctly indicate to the biker the distance to and the direction of the next turn. To notify the biker, we were able to integrate this logic with LEDs and a buzzer that turn on based on how far the biker is from a turn and what the direction of the next turn is. Our system periodically updates to ensure accuracy and can provide a new route if the biker misses a required turn. In addition, we added blinkers that were successfully integrated as well and the biker could turn them on by pressing button placed on the left and right handles. We mounted this setup on a bike and road tested it. We tested primarily in the evenings and the application communicated with the microcontroller consistently and accurately and the LEDs and buzzer turned on at the right times and caught our attention even when we were driving on a busy road. Overall, we were able to accomplish our primary goal of being able to help a user navigate on a bike without having to pull out their phone and that for us was an accomplishment.

5.2 Uncertainties

One of our biggest uncertainties was the speedometer. The speedometer worked fine and produced accurate readings but the reed switch was extremely brittle and easily broke. During road tests, our final reed switch broke and the speedometer stopped working. This was not a major issue since we were testing in an area with strong GPS strength, but not having a working speedometer can lead to latency issues if riding the bike in an area with low GPS strength. Another major concern was the accuracy of the ultrasonic sensor. When we were verifying the correct functionality of the ultrasonic sensor, we did it in a controlled environment. Our obstacle was a fixed big object (a wall), the are around the sensor was clear, dust was minimal and there was no snow or rain. When we mounted it on the bike and tested the system on the road, the sensor did not perform as expected. The readings obtained were inaccurate and the false positive rate was extremely high. We decided to disable the ultrasonic sensor as false warnings are a hazard to bikers.

5.3 Future Work

We plan on developing a more compact circuit in the future with neater wiring to prevent the possibility of wires getting in the way of the bike’s regular operation. A more compact circuit will lead to a more user friendly design and that will be our first step towards commercializing the bike. We hope to potentially partner with bike sharing companies in big cities, partner with bike manufacturing companies and also sell our product as a standalone product.

5.4 Ethical Considerations

The greatest ethical risk with our project is that people who are not familiar with our bike's navigation system can pose a potential risk to public safety since the LEDs, and other features incorporated can confuse them, thus resulting in accidents. To ensure that the IEEE Code of Ethics #1 "to hold paramount the safety…public or the environment" [8] is abided by, we plan on including a tutorial on the mobile application (that is communicating with the microcontroller) to explain how to use the navigation assistant, and also make users aware of some of the potential risks inherent with using the navigation assistant without prior experience.

Another ethical risk with our project is if someone was to hack into the users mobile application and change the final destination, it could lead to kidnappings or other malicious acts. To ensure that IEEE Code of Ethics #9 "to avoid injuring others….malicious action" [8] is abided by, we need to incorporate certain security measures that would help mitigate this risk.

Our project is an effort to contribute to intelligent systems and thus is an implementation of the IEEE Code of Ethics #5 "to improve the understanding….including intelligent systems" [8]. The other codes are not of concern to us due to the following reasons: we plan on working together as a group and bring up any concerns as and when they arise, we have an organized plan and timeline for the project and will work towards the project with integrity, we are not requesting sponsorship other than the $40 funding from the Electrical and Computer Engineering department, and lastly we hope to work closely with the course staff to deliver an impactful project.

6 References

[1] Centers for Disease Control and Prevention (CDC). Web-based Injury Statistics Query and Reporting System (WISQARS). Atlanta, GA: Centers for Disease Control and Prevention, National Center for Injury Prevention and Control. [Online] Available at https://www.cdc.gov/motorvehiclesafety/bicycle/index.html.

[2] NBDA.com, “Industry Overview 2015”, 2018 [Online]. Available at https://nbda.com/articles/industry-overview-2015-pg34.htm.

[3] maxbotix.com “LV-MaxSonar -EZ Series Datasheet”, 2018 [Online]. Available at https://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf.

[4] GM Electronic, “HC-05 Bluetooth Module User Manual”, 2018. [Online]. Available at https://www.gme.cz/data/attachments/dsh.772-148.1.pdf.

[5] microchip.com, “ATMega328P - PU Microcontroller Data Sheet”, 2018. [Online]. Available at http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf.

[6] Sparkfun.com “CEM 1203(42) Magnetic Buzzer Datasheet”, 2018. [Online]. Available at https://cdn.sparkfun.com/datasheets/Components/General/cem-1203-42-.pdf.

[7] mouser.com “L7805CV Voltage Regulator Datasheet”, 2018. [Online]. Available at https://www.mouser.com/datasheet/2/389/l78-974043.pdf

[8] IEEE.org, "IEEE Code of Ethics", 2018. [Online]. Available at http://www.ieee.org/about/corporate/governance/p7-8.html.

7 Appendix

Requirements	Verification
Cellphone: 1) The mobile app on the cellphone should get the correct data from Google Maps which should be converted into data which can be read by the microcontroller **6 points**	1) Send test data from the Google Maps and check on a terminal if it can process the data and make it readable for the microcontroller
Ultrasonic Sensor: 1) should be be able to determine the presence of a significant sized object up to 6.5 meters away 95% of the time **5 points**	1) Place an object up to 6.5 meters away from the ultrasonic sensor and check if the the sensors give out a value on the oscilloscope. 2) Connect the sensor to a microcontroller which is hooked up to a computer. Have the microcontroller calculate the distance between the sensor and the object and display the value on the serial monitor
Speedometer: 1. The Speedometer must periodically calculate the the speed at which the bike is moving and pass the information to the microcontroller. Speedometer should give accurate speed 95% of the time **5 points**	1) Spin the bike's wheel and check if the microcontroller detects a change when the magnet on the wheel spokes comes into close contact with the reed sensor on the bike fork
The LEDs: 1) The correct LEDs should light up when a signal is received from the microcontroller 2) The LEDs should catch the biker's attention within 0.5 seconds at least 90% of the time **5 points**	1) When the biker is approaching a right turn the right LEDs on the handlebar should start blinking and similarly for a left turn, the left LEDs should light up. When the biker is too close to the vehicle in front, the corresponding LED should light up. 2) Perform rider tests where the biker uses a stopwatch to measure the reaction time. 3) Create a test application that allows a user to send any random current

	location and final destination to the bike. Observe if the correct LEDs light up when the values input by the user are close to a turn.
The Blinkers 1) The correct blinker should turn on based on which switch is pressed 95% of the time **5 points**	1) Press left/right buttons to ensure that the correct blinker is lighting up
Buzzer: 1) The Buzzer should produce the correct sound of at least 70 dB when a signal is received from the microcontroller 2) The buzzer should catch the biker's attention within 0.5 seconds at least 90% of the time **5 points**	1) When the biker is approaching a turn, the corresponding tone should be played and when the biker is too close to another vehicle, the corresponding tone should be played. 2) Perform rider tests where the biker uses a stopwatch to measure the reaction time. 3) Create a test application that allows a user to send any random current location and final destination to the bike. Observe if the buzzer beeps with the correct frequency when the values input by the user are close to a turn.
Microcontroller 1) Microcontroller should not use more than 200mA current when powered by a 5V source 2) The microcontroller should be able to process information received from the mobile application and ultrasonic sensor and send the required output to LEDs and the beeper 95% of the time. **12 points**	1) Measure correct power and current supply using a voltmeter and multimeter 2) Develop test script and run 100 times to ensure microcontroller is receiving data and then sending it to the appropriate LEDs and beepers
The bluetooth module 1) The bluetooth module should communicate information from the mobile application to the bike in 2 seconds with a tolerance of +/- 0.885s 2) Bluetooth module should be receiving the correct data and sending it to the microcontroller 95% of the time.	1) Develop multiple test scripts with varying amounts of data that has to be transmitted via the bluetooth module to the microcontroller to ensure data transmission is done in time 2) Ensure data being sent to microcontroller is accurate by looking at the serial monitor

5 points	
Power supply 1) It should give a voltage of 5V with enough current to power all devices with a variability of 10% **2 point**	1) Measure the power supply with a multimeter and ensure that both voltage and current stay with intended levels

Table 5: Requirements and Verification

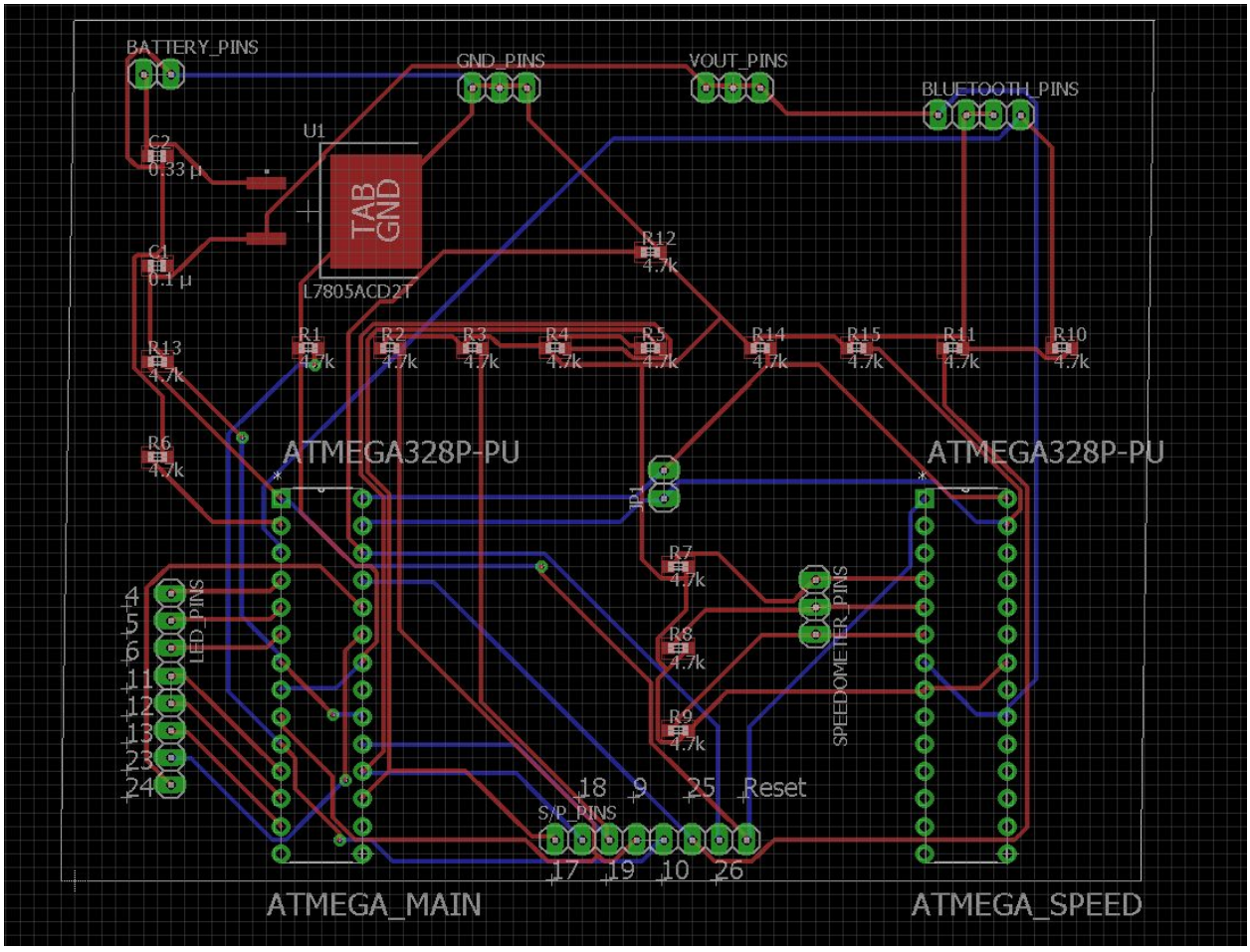

Figure 13: PCB Board

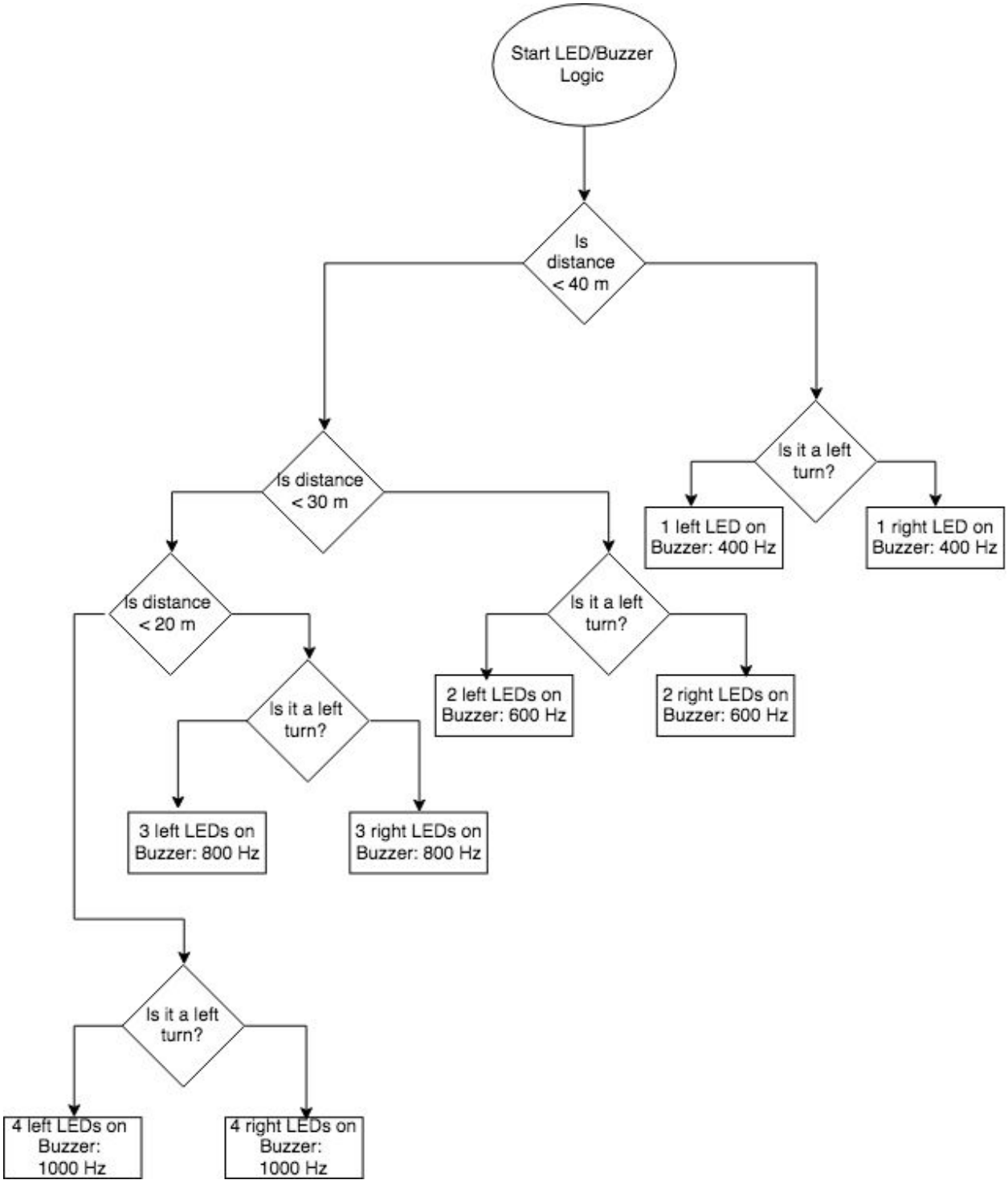

Figure 14. LED/Buzzer Logic

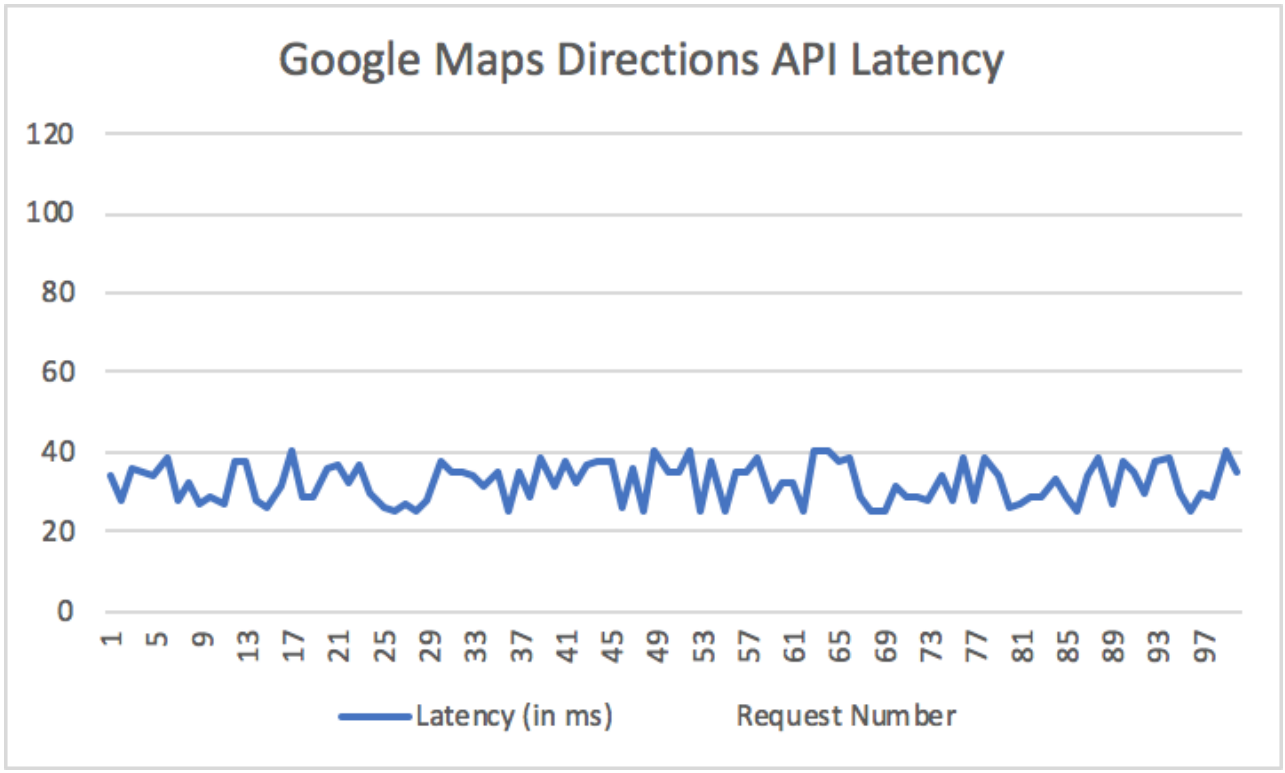

Figure 15. Google Maps API Latency

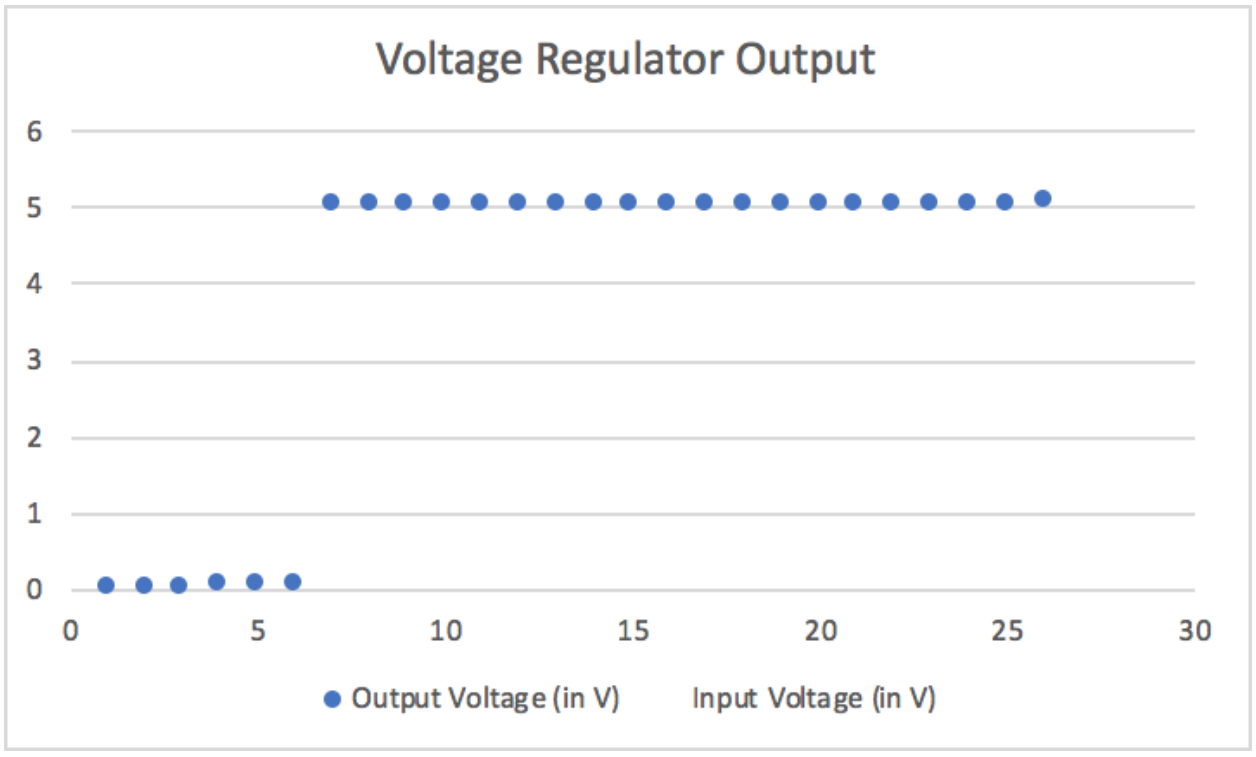

Figure 16. Voltage Regulator Output Voltage