Bicycle LIDAR Warning System

Design Document

Team number: sdmay18-36
Client: Dr. Tuttle
Advisers: Dr. Zambreno

Team Members/Roles:
George Alphonse - Hardware Lead
Nathan Gilbert - Git Guru
Joe Gleason - Meeting Facilitator
Rob Powell - Software Lead
Nate Sorlien - Test Lead
Robert Wernsman - Report Manager

Email: sdmay18-36@iastate.edu
Website: sdmay18-36.sd.ece.iastate.edu

Revised: 2017-12-10 Version 2

Table of Contents

1. Introduction

1.1 Acknowledgement

This project would not be possible without support from the ISU ECpE Department. This includes both advisory and financial support. Without class funding, we wouldn't have the freedom to experiment with cutting-edge technology. A special thank you goes to Dr. Tuttle for guiding our team, overseeing our progress, and helping us make design decisions along the way. His enthusiasm in our project could not be more genuine, fueled by his interest in new technologies and his passion for cycling.

We would also like to acknowledge our class advisor Dr. Zambreno and his Teaching Assistants that provide us with helpful, timely feedback. With their help, each new report or document version that we create is of equal or greater quality than the last. We would also like to acknowledge ETG for providing equipment and helping us purchase components throughout the design process.

1.2 Problem and Project Statement

In today's connected world, using your smartphone and car infotainment system is becoming part of the daily driving routine. While they put more information and convenience at the driver's fingertips, they also have the potential to be very distracting. Distracted drivers are a threat to everyone on the road, but bicyclists have the most to lose when encountering a distracted driver. Getting hit from behind is the leading cause of cycling fatalities. From 2005 to 2010, the number of cyclists killed from distracted driving went up by 30%. Additionally, by 2021, several major car manufacturers will have autonomous cars available for highway use. According to a UC Berkeley research engineer, "Bicycles are probably the most difficult detection problem that autonomous vehicle systems face." Cyclists need a way detect vehicles approaching from the rear while cycling on highways because they are at a higher risk of being hit by distracted drivers.

To satisfy this need, our project aims to provide a warning system to improve the safety of bicyclists. This system will warn cyclists of vehicles approaching from the rear. It will utilize LIDAR (Light Detection and Ranging) technology, which uses light in the form of a pulse laser to detect objects. The LIDAR warning system will be battery powered, mounted under the seat, and interface with a smartphone to provide alerts to the cyclist. This safety system equipped on a bicycle could give the cyclist the forewarning they need to avoid a potentially life-threatening injury.

Our main goal is to create a system that consistently alerts the bicyclist of approaching vehicles. Without alert consistency, the cyclist cannot trust the system and does not get any safety benefit.

1.3 Operational Environment

The expected operational environment for our system is similar to the intended operating environment of a road bicycle. The end product will be exposed to both extremes of outside temperatures as well as potentially higher temperatures from direct sunlight. Our product will be mounted under the seat of a bicycle facing the road behind the cyclist. Due to the exposed nature of this mounting location, our product will be exposed to dust and particles from the road. It could

also be exposed to water if the cyclist uses the product in the rain. Our design aims to address these environmental conditions to provide a rugged system to the user.

1.4 Intended Users and uses

We categorized our users into three main groups: bicycling enthusiasts, safety-conscious users, and gadget enthusiasts. Bicycling enthusiasts consist of people who are above average bikers or race professionally. The end product for bicycling enthusiasts would offer accurate status reports on their speed, distance and time. The end product for safety-conscious users would offer highly accurate, timely notices of vehicles approaching from behind. The user can receive alerts both through the Android app and through a secondary light and buzzer system. Gadget enthusiasts consist of users who are interested in cutting-edge technology. For these users, price has no limit. Gadget enthusiasts desire many bells and whistles from their devices. The end product for the gadget enthusiasts should offer a rear-view camera, GPS integration, and low power consumption.

After considering our three types of users, there is a range of features that we intend to implement in our design. With our design, we want to target all three types of users. The end product will include features such as but not limited to: an accurate LIDAR sensor, low power consumption, a rear view camera, GPS and statistics about the user's ride.

1.5 Assumptions and Limitations

Assumptions:

- Cyclists will be riding on highways, and cars will not be going faster than 65 mph.
- Cyclist will spend the majority of their time on the right side of the road.
- Cyclists are already using safe cycling practices to further reduce their chances of getting hit by approaching vehicles.
- The optimal sensing conditions for the sensor is clear skies, with no dust or debris.
- The cyclist will be going at least 20 MPH on the highway (for adequate notification time).

Limitations

- The system will be battery powered, and will have a battery life of roughly 6 hours.
- The system can sense vehicles up to 45m away, which gives cyclist anywhere between 3-4 seconds to respond depending on the speed of the vehicle.
- This system should be able to detect multiple approaching vehicles.
- The system will only detect approaching vehicles (may not detect approaching cyclists/motorcycles).
- There may be some inconsistencies in car detection when cyclists are going around a corner.
- The system will only be compatible with Android smartphones.
- The system will be small enough to be mounted under the bicycle seat.
- This project will take two semester to complete.
- The budget for this project is $600.

1.6 Expected End Product and Deliverables

The final product will consist of three main components. The main module will be mounted to the back of the bike and will contain the microcontroller and LIDAR sensor. A secondary module will contain a visual and/or audio output to alert the cyclist of an approaching vehicle. The final component will be an Android application to interface with the system as an alternative to the secondary module. In addition to the product itself, there will also be a basic user manual describing the setup and use of the product. These components will be completed by the end of 492.

The main module will contain the majority of the components in the system. In addition to the microcontroller, LIDAR sensor, and camera, it will also contain the batteries needed to power the system. The size of this component should be reasonably small so that it can be mounted conveniently to the back of the majority of bicycles.

The secondary module will contain one or more displays, such as a combination of LED lights and a seven-segment display, and possibly a speaker to output an audible sound. This component's purpose is to effectively alert the cyclist of an approaching vehicle, and if possible, provide additional information, such as the distance or time between the cyclist and the approaching vehicle. This module will most likely be connected to the main module via a small wire running along the bike frame and mounted near the center of the handlebars.

The Android application will be available as a wireless alternative to the secondary module and fulfil the same purpose. The device will be mounted to the handlebars or center frame using a commercially available bike mount for cell phones. In addition, the app may provide more information to the user, such as velocity and distance traveled on the bike ride.

2. Specifications and Analysis

2.1 Proposed Design

We have decided to use the LeddarTech Leddar Evaluation Kit LIDAR sensor, a cheaper variant of the Leddar M16 with a 45 degree beam width, 16 detection segments, and up to 50 meter range. This sensor supports the USB, RS-485, and CAN interface. We will choose the interface based on bandwidth and throughput requirements, easy of use, and MCU compatibility. We will use the TI MSP430 as our early-prototype development platform. This platform is relatively easy to work with and will be used until our functional prototype is complete. At that point, we will develop our own PCB with an MCU featuring a more dedicated feature set. We have chosen to use WiFi Direct to facilitate communication between the mobile application and the MCU. We are using Android Studio to develop the mobile application, so it will be exclusive to the Android platform. We have made a template for the UI and are currently investigating methods to display a live video stream. We are still investigating different options for the camera; this will be a detail addressed in the second semester of Senior Design.

Our design should adhere to the following functional and nonfunctional requirements:

Functional Requirements:

- Working LiDAR sensor with adequate range and resolution

- Algorithm to detect an approaching vehicle from sensor data
- Microcontroller (MCU) that can simultaneously communicate with the Android app, camera, and sensor, as well as process sensor data
- Wireless communication between MCU and Android app
- Android application displays statistics, vehicle detection status, and a live video feed

Non-Functional Requirements:

- Give a cyclist 10 seconds to react to a vehicle approaching from behind at highway speeds
- This leads to a maximum range of about 270 meters
- Algorithm able to finish execution more than 60 times per second
- Battery life to sustain a typical day of cycling (4 hours)
- Small enough to mount on the rear of a bike
- Android app is responsive and phone battery lasts at least 4 hours

2.2 Design Analysis

The team's work so far has consisted of both researching hardware/software solutions and implementing solutions to create a prototype for each of the project's components.

With our mobile application, we initially wanted to target both IOS and Android devices. However, after researching available frameworks, we decided instead to develop only for Android. The available frameworks provide tools for simultaneous development on both Android and IOS, but each platform's build and development tools are still required for testing and deployment. This was a problem for our team since no team members own an OSX device, which is required for iOS development. As a result, we are only focusing on Android development for this project.

For communication, we chose to use Wi-Fi Direct over Bluetooth since the MCU needs to stream video data to the user's Android device. Bluetooth does not have the bandwidth to stream video other than at very low resolutions. There are some tradeoffs associated with using Wifi Direct. While it can support a higher bandwidth, it does this at a cost of higher power consumption for both the sender and receiver. We have started development on the Android application UI as well as a service to manage Wi-Fi Direct connections and communication over those connections.

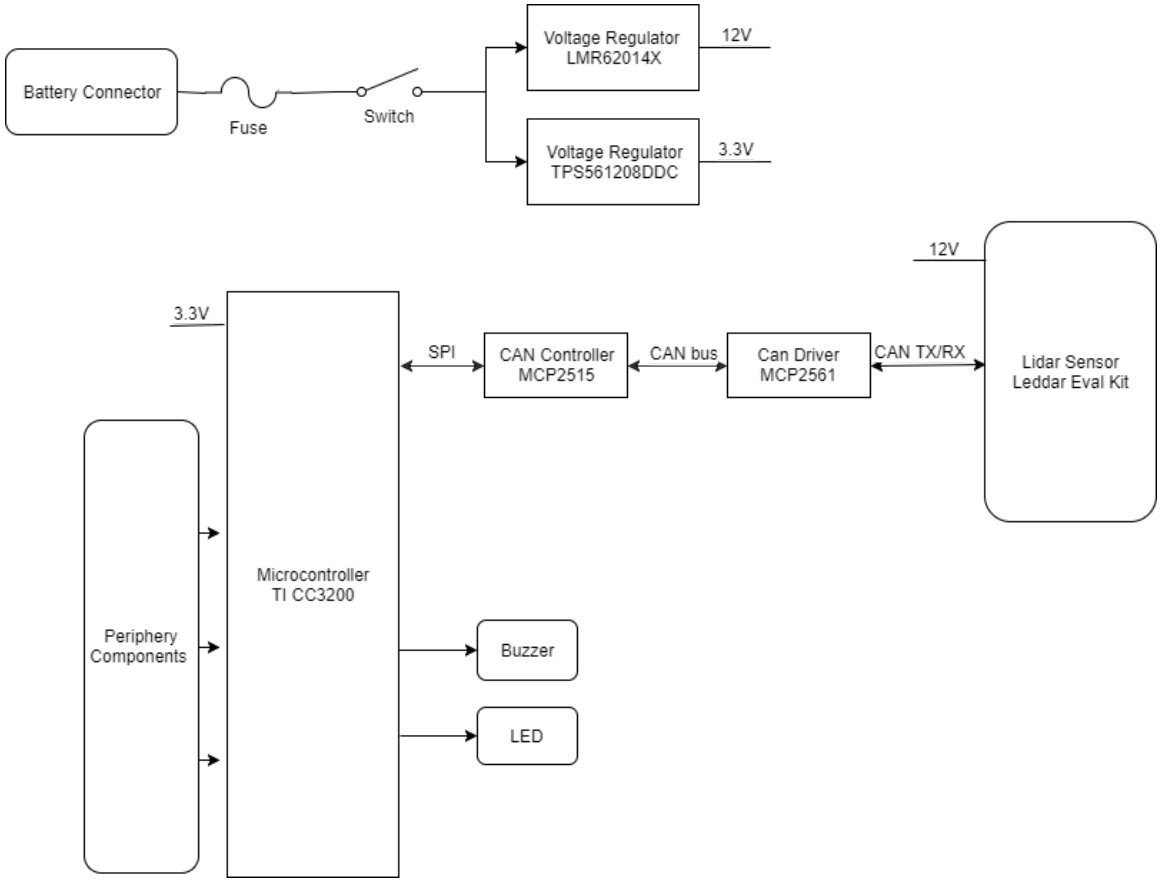
The purpose for our MCU is to communicate with the Android application, work with camera data, and work with LIDAR sensor data. For the first semester, we are not focusing on any development regarding the camera, although we plan to use SPI for camera communication. Our main MCU focus for the first semester was establishing communication with the LIDAR sensor. The LIDAR sensor we purchased (Leddar Evaluation Kit) can be accessed using USB (with Windows drivers only), RS-485, and CAN Bus. Since very few MCUs support these protocols out of the box, we decided to choose CAN. We chose CAN because the sensor supported a higher data rate; it also seemed like the more modern bus communication approach.

Another consideration for the MCU was wireless capabilities. Since we chose Wifi Direct, we wanted to select a MCU that would support this protocol natively. We identified the TI CC3200 as our ideal choice; this MCU includes a development environment similar to the simpler TI Launchpads while also providing networking capabilities. This allowed us to use a readily-available TI MSP430 Launchpad for development before we finalized our choice for the CC3200.

After we decided that we would use CAN to communicate with our sensor, we began searching for a controller that would be compatible with the CC3200. We found that the MCP2515 CAN controller was a very popular option that uses SPI to communicate with an MCU. We purchased a board containing this controller, a transceiver, and other required components to begin working on the communication using a TI MSP430 launchpad. We found third party libraries designed to communicate with the MCP2515 using the MSP430 and began to modify them to meet our requirements. We are now able to request a set of detections from the sensor and then store those detections in memory on the MCU. The next step is to modify our code to support the SPI implementation on the CC3200 and then implement the first version of

our algorithm on the MCU itself. We also need to test how well our algorithm runs on the MCU, so it remains to be seen how effective our MCU code and communication is.

We have started PCB development this semester, but the majority of the work will be completed in 492. We have begun researching what periphery components are needed for the TI CC3200 microcontroller. Additionally, we have also chosen voltage regulators to supply 12V to the LIDAR sensor, and 3.3V to the microcontroller. We chose voltage regulators with high efficiency, and the lowest footprint. In the schematic we have also outlined how the LIDAR sensor will communicate with the microcontroller using SPI and CAN.



The algorithm designed to process data from the sensor was implemented in C. The basic idea involves simply looking at each of the 16 detection segments and determining whether or not there has been an object getting closer to the sensor over the past three detections. If so, there is a car approaching. There is a slight issue in the sense that an approaching vehicle will not always stay within one segment. It can instead move from one segment to an adjacent segment. To deal with this, "phantom readings" were added as follows: after storing all readings from the sensor, for each empty segment, add a "phantom reading" at the same distance as a real reading in an adjacent segment. This "phantom reading" is to be treated as a regular reading. This will handle nearly all cases of an object jumping segments.

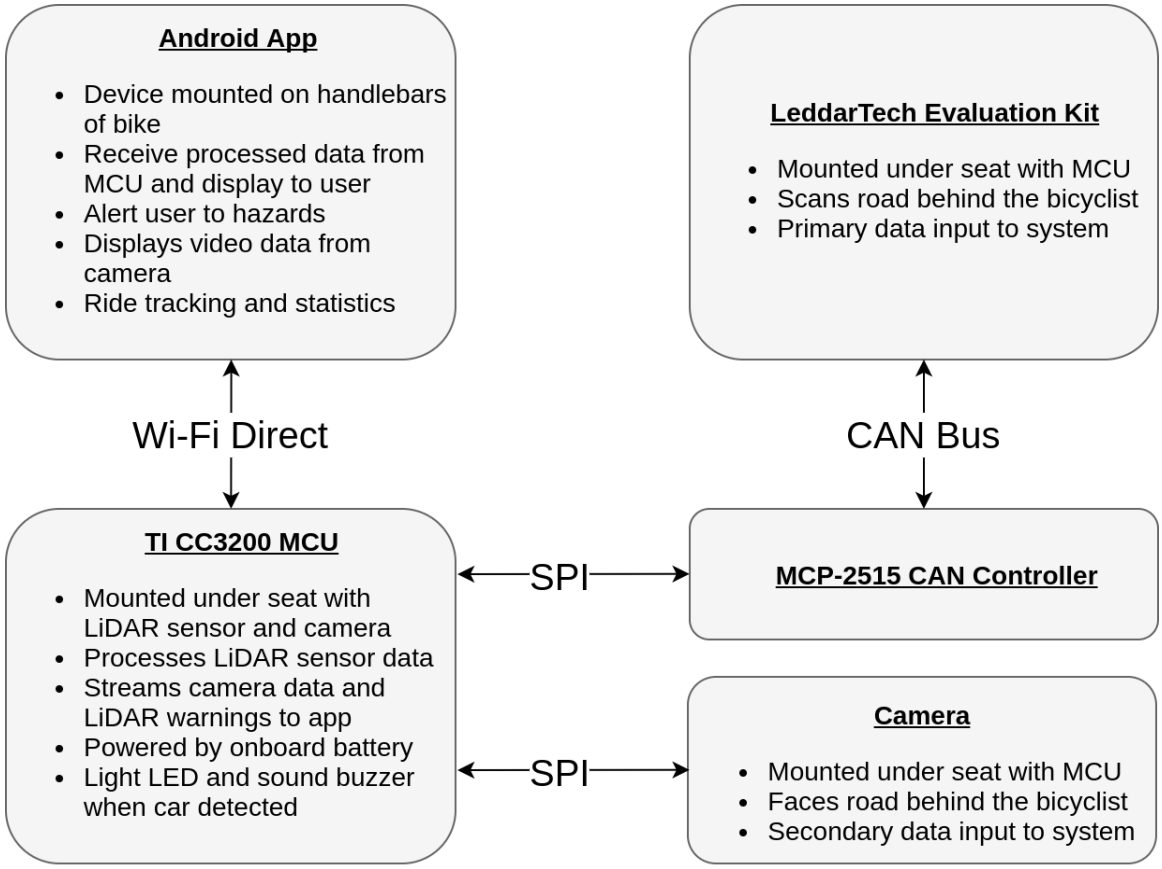
There are a few potential issues with this algorithm. Firstly, it relies on the assumption that an object will not skip an entire segment from one frame to another. It is also susceptible to noise when processing data from a stationary sensor. While this susceptibility to noise is reduced once the sensor starts moving, it is not gone entirely. Looking into the future, some resistance to noise will most likely be added. In addition to that, other algorithms which accomplish the same thing as this one will be researched. If the researched

algorithms perform better than the currently implemented one, changes may be made to resemble the researched one more.

3. Testing and Implementation

3.1 Interface Specifications

The CC3200 MCU will communicate with the user's Android device, a camera and the LeddarTech M16 LIDAR sensor. Communication with the android application will be handled with Wi-Fi direct using the MCU's dedicated networking interface. The MCU's SPI interface is used to send and receive data from the camera and a CAN controller used as an intermediary for communication with the LIDAR sensor.



3.2 Hardware and software

The Leddar sensor and the associated SDK provide a means of determining the distance an object is from the sensor. The specified performance of the sensor need to be verified before further progress

can be made. The configuration GUI provided with the sensor has an option for logging and recording data. This works perfectly for the feasibility testing procedure described below.

Testing the circuitry of the system requires running simulations using a circuit design software, as well as doing some bench testing in the lab. We plan on using Multisim for the design of our circuit and running simulations. After simulating the circuit design, we intend to build and test the circuit on a perfboard. The bench testing equipment that we will be using include power supplies, digital multimeters, oscilloscopes, and function generators. All of the bench testing equipment needed can be found in the Coover Hall lab rooms. Additionally, Multisim is also installed on every computer in Coover Hall. The final stage of hardware testing includes testing our PCB design using Ultiboard and the bench equipment.

Initial testing of Bluetooth communication was done using the BlueZ library on a Raspberry Pi 3 Model B and Bluetooth Sockets on Android using the RFCOMM interface. For WiFi Direct, the wpa_supplicant utility was used to make a connection between the Raspberry Pi and the Android device, which used its internal WiFi Direct connections menu. From there, the devices could communicate using normal POSIX sockets.

Initial testing of communication between the MCU and the LIDAR sensor was done using a TI MSP430 Launchpad. The MSP430 was connected to a MCP2515 CAN controller over SPI and communicated with the LIDAR sensor over the CAN bus. We used an oscilloscope at various stages to view the data being transmitted on the SPI and CAN busses.

The Android application will be tested with local unit tests and Android instrumented tests. Unit tests will be used to test methods that only use java libraries and can be run in the JVM. Instrumentation tests will be used for integration testing and to test application methods that require Android resource classes and/or hardware. The Android instrumentation API will be used to provide the required Android resources for these tests.

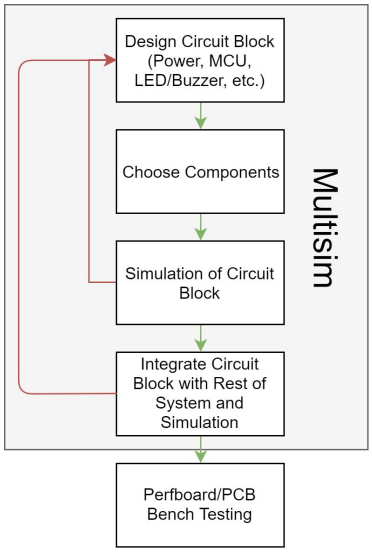
One major implementation challenge that we will face is transitioning from a development board to a PCB-mounted microcontroller. This will most likely be due to ordering incorrect parts, or configuring the MCU incorrectly on the PCB. Another implementation challenge that we will have to address is the movement of the bicycle affecting the accuracy of the LIDAR sensor. To address this issue we will have to create a robust detection algorithm, as well as identify a sensor with wide viewing angles.

Another issue we will face with this design is related to the LIDAR sensor we selected. Theoretically, the sensor will be able to provide measurements at a sufficient rate, but it will not provide adequate range. Our testing has revealed a maximum range for an oncoming compact car to be around 50 meters. However, our requirement of a 10 second warning of an approaching car at highway speeds requires a maximum range of 270 meters. As LIDAR technology matures, costs will come down and performance will increase, so we would likely be able to find a sensor within our cost and performance requirements in the future.

3.3 Process

We will first begin the design and testing of the circuit in Multisim. Multisim allows us to select specific components and use them in our circuit for simulations. When designing the circuit, we will design one circuit block at a time to make sure that the circuit block works correctly

individually. When testing a circuit block we will be looking for the correct output/behavior. Multisim simulations will not provide us a lot of information on noise, or parasitics associated with the PCB. After verifying the operation of the circuit block individually, we will then integrate it with the rest of the circuit, and make sure that system works. Once we have simulated the whole circuit in Multisim, we will then move on to building the circuit on a perfboard or PCB.



We will be using the same testing process as above to test our physical circuit on the perfboard or PCB. However, when testing the actual physical circuit, we will be able to better evaluate the performance of the circuit. Using the bench testing equipment, we will be able to accurately measure the power consumption, signal integrity, continuity, and speed of the circuit. Additionally, at this stage of testing, we will also be able to test that the MCU code works correctly, and that it integrates well with our circuit.

There will be three different procedures for the Leddar feasibility test. The first will test the field of view, range, and accuracy of the sensor, the second will test how the sensor responds to different materials, and the third will test how the sensor responds to moving vehicles specifically.

1. Field of View, Range, and Accuracy
    a. In an open space with ideal lighting, position the Leddar sensor at approximately the height of a bicycle seat.
    b. Place an ideal (reflective) object at a known location.
    c. Collect data from the sensor for 10 seconds.
    d. Repeat steps b and c for several known locations.
        i. The locations should sufficiently test the field-of-view and range of the Leddar sensor.
2. Various Materials
    a. In an open space with ideal lighting, position the Leddar sensor at approximately the height of a bicycle seat.
    b. Place a non-ideal object at a known location.
    c. Collect data from the sensor for 10 seconds.
    d. Repeat steps b and c for several known locations.
        i. The locations should sufficiently test the field-of-view and range of the Leddar sensor.
    e. Repeat steps b-d for several different materials.
        i. The types of materials should range in color and reflectivity.
3. Moving Vehicles
    a. Locate a mildly busy stretch of road.
    b. Position the Leddar sensor near the curb (off the road) at approximately the height of a bicycle seat, pointing down the road toward oncoming cars.
    c. Collect data from the sensor for approximately a minute while recording video of what the sensor sees.
        i. Let several cars pass by.
    d. Repeat steps a-c for a few different speed limit areas.
    e. Repeat steps a-d at midday, evening, and night.
        i. The purpose of this is to test different light levels.

To test the communication between the MCU and the sensor, we first verified that the MCU and CAN controller were able to communicate. An oscilloscope was used to observe the SPI communications to verify that the data on the bus was what we expected it to be. Next, we used the debugging features of Code Composer Studio to check values being read from the controller's registers. Once we verified that we were successfully communicating with the controller, we began the process of configuring the controller registers to communicate with the sensor over the CAN bus. We again used an oscilloscope to view the CAN-High and CAN-Low channels to view the current state of the bus and to verify it was working as expected or to view error frames. Finally, we loaded the data from the sensor into memory and viewed it with the debugger to confirm that it was valid.

Non-functional testing is performed to meet the requirements of giving a cyclist 10 seconds to react to a vehicle approaching from behind at highway speeds. This feature for our system is significant for evaluating the usability of the system. Testing for this requirement is done with real traffic as well as a controlled test with one vehicle approaching. Additional non-functional testing is done to test the video and algorithm performance to ensure that video frames are sent at a minimum of 15 frames per second and that the algorithm executes at 15 iterations per second. More non-functional testing is performed to ensure that battery life of the bicycle-mounted system and the Android device meets or exceeds 3.6 hours. This is accomplished with real stress tests on both platforms.

3.4 Results

This project is a two-semester project and is still under way. While conclusive results for the system are not yet proven, we have many preliminary results, calculations, estimations, and details learned from our time spent developing this project. The results we have obtained and the conclusions we have drawn so far can be found in the Design Analysis and Testing & Implementation sections of this document.

4. Closing Material

4.1 Conclusion

So far, our team has researched and began development on the major areas of our project: LIDAR sensor, microcontroller, mobile app, and hardware design. We have received all components we have identified so far, and have been able to construct prototypes for the areas of development. This involved a range of different domains: Android software, MCU software, custom car detection software, bread-board debugging, and hardware design.

Our main goal is to improve the safety of a cyclist by providing a warning system that detects vehicles approaching from the rear. This system will give the cyclist a few extra seconds to react to the coming vehicle and increase their awareness about their surroundings. Our plan of action is to utilize the Leddar LIDAR sensor data to determine if a vehicle is approaching the cyclist at an unsafe speed or position. The sensor data will be processed with a MCU and alert the cyclist using companion smartphone app. There will also be a rear-view camera connected to the MCU that streams a live video to the app, giving the cyclist even greater spatial awareness. We are using both

a LIDAR sensor and a camera stream to not only alert the user, but also provide the user with visuals to help them cycle safer.

4.2 References

1.) "WiFi Direct Configuration Scripts." *WiFi Direct Configuration Scripts - Texas Instruments Wiki*, Wikipedia, processors.wiki.ti.com/index.php/WiFi_Direct_Configuration_Scripts.
2.) Huang, Albert. *An Introduction to Bluetooth Programming*, Cambridge University Press, Nov. 2002, people.csail.mit.edu/albert/bluez-intro/.
3.) *BluetoothSocket*. Android Developers, 20 July 2017, developer.android.com/reference/android/bluetooth/BluetoothSocket.html.
4.) *Leddar Evaluation Kit*. Leddartech, Aug. 2017, leddartech.com/leddar-evaluation-kit/.
5.) "List of Wi-Fi microcontrollers." *Wikipedia*, Wikimedia Foundation, 12 Sept. 2017, en.wikipedia.org/wiki/List_of_Wi-Fi_microcontrollers.
6.) "CC2640 (ACTIVE)." *CC2640 SimpleLink ultra-Low power wireless MCU for Bluetooth low energy | TI.Com*, Texas Instruments, 2017, www.ti.com/product/CC2640/compare.