MID-AMERICA TRANSPORTATION CENTER

Report # MATC-KU: 152-1

Final
WBS: 25-1121-0005-152-1

Impact of Electric Bikes on Rider Safety on Campus - Phase I

Christopher Depcik, Ph.D.
Associate Professor
Department of Mechanical Engineering
University of Kansas

2018

A Cooperative Research Project sponsored by
U.S. Department of Transportation- Office of the Assistant Secretary for Research and Technology

Impact of Electric Bikes on Rider Safety on Campus - Phase I

Christopher Depcik, Ph.D.
Associate Professor
Department of Mechanical Engineering
University of Kansas

A Report on Research Sponsored by

Mid-America Transportation Center

University of Nebraska–Lincoln

December 2018

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. 25-1121-0005-152-1	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle Impact of Electric Bikes on Rider Safety on Campus - Phase I		**5. Report Date** December 2018 **6. Performing Organization Code**
7. Author(s) Christopher Depcik, Ph.D., https://orcid.org/0000-0002-0045-9554		**8. Performing Organization Report No.** 25-1121-0005-152-1
9. Performing Organization Name and Address Mid-America Transportation Center 2200 Vine St. PO Box 830851 Lincoln, NE 68583-0851		**10. Work Unit No.** **11. Contract or Grant No.** 69A3551747107
12. Sponsoring Agency Name and Address University of Kansas, Department of Mechanical Engineering 3144C Learned Hall 1530 W. 15th Street Lawrence, Kansas, USA, 66045-4709		**13. Type of Report and Period Covered** Final Report (May 2017-December 2018) **14. Sponsoring Agency Code** MATC TRB RiP No. 91994-18

15. Supplementary Notes
Conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

16. Abstract
Electric bikes, or e-bikes, provide a potentially significant avenue to facilitate large reductions in greenhouse gases and hazardous emissions while promoting the usage of public transportation. However, little research exists on how these faster, heavier, and quieter vehicles impact rider safety. The goal of this effort was to drive a biomechanically designed e-bike throughout a campus environment in order to obtain quantitative and qualitative data on its operation enabling better models of e-bikes for driving simulator projects, emissions studies, and other transportation related efforts. Key to this endeavor was the development of a low-cost and mobile Light Detection and Ranging (LIDAR) system that could accurately measure the distance between objects (i.e., pedestrians and cars) and the e-bike. While some success was obtained, the limited processing rate of the components chosen precluded completion. As a result, the development of a second-generation LIDAR system is currently progressing along with two other synergistic activities that will expand the original efforts in order to provide information pre-crash to reduce risks and after accidents as part of post disaster inspection systems.

17. Key Words Safety, Risk, Laser radar, Bicycle travel, Bicycles, Testing	**18. Distribution Statement** No restrictions.		
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 45	**22. Price**

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Table of Contents

List of Figures

List of Tables

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

Abstract

Electric bikes, or e-bikes, provide a potentially significant avenue to facilitate large reductions in greenhouse gases and hazardous emissions while promoting the usage of public transportation. However, little research exists on how these faster, heavier, and quieter vehicles impact rider safety. The goal of this effort was to drive a biomechanically designed e-bike throughout a campus environment in order to obtain quantitative and qualitative data on its operation enabling better models of e-bikes for driving simulator projects, emissions studies, and other transportation related efforts. Key to this endeavor was the development of a low-cost and mobile Light Detection and Ranging (LIDAR) system that could accurately measure the distance between objects (i.e., pedestrians and cars) and the e-bike. While some success was obtained, the limited processing rate of the components chosen precluded completion. As a result, the development of a second-generation LIDAR system is currently progressing along with two other synergistic activities that will expand the original efforts in order to provide information pre-crash to reduce risks and after accidents as part of post disaster inspection systems.

Chapter 1 Initial Development of Low-Cost LIDAR System

Note: This chapter is published as Blankenau, I., Zolotor, D., Choate, M., Jorns, A., Homann, Q., and Depcik, C., "Development of a Low-Cost LIDAR System for Bicycles," SAE Technical Paper 2018-01-1051, 2018, doi: 10.4271/2018-01-1051.

1.1 Background

Environmental and health issues within cities resulting from traffic emissions have led to some municipalities banning or restricting internal combustion engines (National Safe Routes to School Task Force 2008, Weinert et al. 2007, Rose 2012). In response, commuters often adapt by using bicycles and electric-assisted bicycles (e-bikes), subsequently making cycling more popular in urban areas (Rose 2012, National Highway Traffic Safety Administration 2016). In addition to environmental benefits, many are urged to bicycle to improve health through exercise (National Safe Routes to School Task Force 2008, Rose 2012). While the large-scale adoption of bicycling as a primary source of transportation has tremendous potential to increase the quality of people's lives, it can only do so after mitigating the hazards that cyclists face (National Safe Routes to School Task Force 2008).

Generally, several factors make people reluctant to use e-bikes and conventional bicycles as transportation. Weather and the impact cycling has on one's appearance can deter some (Weinert et al. 2007, Du et al. 2013). However, the concern for safety is the most substantial barrier to adopting cycling as primary means of transportation (Götschi, Garrard, and Giles-Corti 2016). This potential for harm is attributed primarily to infrastructure, motorists, and the absence of protection for the cyclist (Weinert et al. 2007). In particular, cyclists in the United States (US) reported that motorists are their principal concern (Schroeder and Wlibur 2012). This is understandable considering that in the US, there were 818 cyclist fatalities and 45,000 cyclist injuries from motor vehicle-related accidents in 2015 (National Highway Traffic Safety

Administration 2016). Overall, the number of cyclist deaths per year has been increasing, with cyclist fatalities steadily becoming a more significant percentage of the total transportation fatalities (National Highway Traffic Safety Administration 2016, Bureau of Transportation Statistics 2016).

In this area, the enforcement of strict adherence to road rules for both cyclists and motorists will improve cyclist safety (Räsänen, Koivisto, and Summala 1999, Summala et al. 1996, Yang et al. 2015). Additionally, competency and awareness can reduce the likelihood of collisions (Räsänen, Koivisto, and Summala 1999, Summala et al. 1996, Walker 2005). However, motorists are by no means the only reason for accidents as cyclists also have lapses in judgment. Specifically, many cyclists stop adhering to traffic laws when they are not held to the same standards as motorists (Du et al. 2013, Wu, Yao, and Zhang 2012). For instance, cyclists will continue riding even though there is a stop sign or stop light with those on e-bikes more likely to do so due to improved acceleration capabilities (Du et al. 2013, Yang et al. 2015, Wu, Yao, and Zhang 2012). In addition, the existing infrastructure contributes to safety issues. Many bicyclists view integrated road conditions as four times more onerous than the environment in dedicated bike lanes; thus, there have been efforts to separate cyclists from motorists (Hunt and Abraham 2007). Hence, decreasing the interaction between cyclists and motorists improves cyclist safety, potentially through integrated bike lanes (DiGioia et al. 2017). Of note, this modification does not prevent collisions in intersections and considerations must be made for the costs incurred. While a long-term infrastructural design shift will foster safer conditions for bicyclists, such changes are unlikely until cyclists represent a more significant portion of transportation (Macmillan et al. 2014, Flusche 2009).

Therefore, while the infrastructure slowly evolves and adapts, an immediate solution is required to improve cyclists' safety. This answer depends on the conditions cyclists face and the shortcomings of current safety measures. Because of highly variable speeds, road surfaces, and live traffic conditions, it can be difficult to maintain rear facing awareness (National Highway Traffic Safety Administration 2016). A standard resolution to this problem is to install mirrors, on either the handlebars or helmet, to reduce the time taken by rearward observations. However, the field of view in mirrors is often limited and tends to offer poor depth perception. Moreover, mirrors provide intermittent performance by giving feedback only while being observed. Additionally, there is the added risk that the rider's attention is distracted from their front, which is a significant risk since 84% of cyclist fatalities occur from head-on collisions (National Highway Traffic Safety Administration 2016). Instead, a rear-mounted system capable of continuously tracking motor vehicles along with their distances and speeds could provide an early warning system for cyclists, subsequently reducing the occurrence of accidents.

However, any system designed specifically for use on a bicycle faces unique constraints. It must be affordable and not negatively influence the ride experience. Taking lessons from the helmet, bicyclists tend to be reluctant to accept these costs in exchange for safety (Finnoff et al. 2001). Therefore, reception hinges on providing a reasonable sense of security and reliability, all while reducing cost, weight, and maintenance. In consideration of these expectations, a Light Detection and Ranging (LIDAR)-based system is feasible, given its capabilities of both high speed and accurate monitoring of traffic situations with relatively low computational requirements for data processing (Williams et al. 2013, Puente et al. 2013, Glennie 2009, Jeon and Rajamani 2016). In this area, there have been several previous attempts to equip bicycles to monitor road conditions and improve safety.

As early as 2011, a team at Rutgers University began developing a computer vision system to detect cars (Smaldone et al. 2011). In 2014, a team at Northeastern University created a distance based sensor system that would provide feedback to riders based on the distance of an object (Castellanos 2014). The system used a small array of stationary ultrasonic distance sensors situated on both the front and rear of the bike and feedback was presented through light and noise notifications. In the same year, Wallich built a system using a prior version of the LIDAR sensor that is used in this report. Employing an Arduino-based platform, he used LIDAR as the rangefinder to detect any oncoming traffic from the rear (Wallich 2015). A few years later, a team from the University of Minnesota developed a multi-sensor bicycle safety system that included the same LIDAR element that is used in this project and mounted it to a stepper motor to add a second dimension of measurement (Woongsun and Rajamani 2016). Because of the low acquisition rate of the sensor, the team built an algorithm to track objects instead of measuring through a continuous sweep. Currently, Garmin has a commercial product available (Varia™) that uses radar to detect the presence and relative velocity of approaching traffic (Garmin 2017).

1.1.1 Focus

While all of these efforts had varying levels of success, there remains a fundamental need by the cycling community for a low-cost system that can effectively monitor traffic conditions and improve rider safety. Moreover, seeing as how most cyclist fatalities caused by vehicles occur at the front of the vehicle (NHTSA's National Center for Statistics and Analysis 2017), cyclists are especially concerned with incoming hazards. Hence, monitoring vehicles from the rear of the bike will allow cyclists to better focus on navigation and oncoming potential issues while still being alert to rearward threats.

As a result, this chapter describes the integration of inexpensive commercial microcontrollers with LIDAR based distance measurements for use on the rear of an e-bike. The following sections first describe the hardware and software of the system illustrating an iterative process at creating the least expensive solution while incorporating an open-platform software package for vehicle recognition. Subsequent testing on the rear of an e-bike finds successful automobile identification; however, processing limitations preclude on-road efforts. Therefore, this paper ends with a discussion of future upgrades required to handle traffic conditions.

1.2 Hardware and Software

The base component of the system incorporates the LIDAR-Lite v3 module produced by Garmin. It is capable of communicating over either Serial Peripheral Interface (SPI) or Inter-integrated Circuit (I^2C) connections and can provide distance measurements with an accuracy of +/- 2.5 cm at a frequency of up to 500 Hz (Garmin International Inc. 2017). Because the LIDAR module can only perform one-dimensional measurements, it is mounted directly on a 400-count stepper motor (StepperOnline®, 14HR05-0504S) that traverses in a horizontal direction providing a second dimension to track targeted objects on the road. LIDAR calculations and its control are based upon the communication between two subsystems: a microcontroller and a small computer. Over the course of the research, two different subsystems were built. The first was designed primarily as a testbed and, therefore, each part was chosen for its versatility with lower priority placed on size and cost. Construction of the second system focused primarily on size and cost.

1.2.1 Testbed Subsystem

The microcontroller implemented in the first iteration was an Arduino Mega 2560 R3. This is an open source product built around the Atmega2560 8-bit Atmel Microcontroller and

operates at 16 MHz. It is capable of powering sensors at either 3.3 or 5 VDC while requiring 7-12 VDC to run. It has four built-in hardware serial ports for expedient use with sensors; hence, it does not need to emulate serial ports with General-purpose input/output (GPIO) pins, which is considerably less efficient. Notably, a serial port emulator must process the data bit-by-bit, and since it does not contain a bus to save the incoming bits, it cannot process in clusters. This prohibits processing while data are being read.

The Raspberry Pi 3 Model B was chosen as the computer subsystem for the first design iteration. It is a single board computer optimized for running Raspbian, a Debian-based Linux distribution Operating System (OS) and is capable of running the Open Source Computer Vision Library (OpenCV) C++ software package (OpenCV team 2017). This computer has a Quad-core 1.2 GHz Broadcom BCM2837 64-bit CPU with 1 GB of RAM. It comes with both Wi-Fi and Bluetooth modules built in, four Universal Serial Bus (USB) ports, two Camera Serial Interface (CSI) ports, an Ethernet port, an auxiliary (AUX) port, a High-Definition Multimedia Interface (HDMI) port, and 40 GPIO pins. It was integrated with Raspberry Pi's Camera Module v2 (Pi Cam) through one CSI connection to track cars through a live video stream (Raspberry Pi 2017). The Pi Cam was chosen based on its compatibility and its low video quality configuration (480p), which is ideal for image processing with limited resources.

Communication between the Model B and Mega was conducted initially over a serial connection. However, the intermittent nature of the data produced by the Raspberry Pi resulted in slow and unreliable data transfer. The communication protocol was switched to I^2C, which provides simple short distance intra-board communication within a single system and only requires two signal wires from each board to exchange information (NXP Semiconductors 2014).

Switching to I^2C solved the serial connection issues while making data transfer quicker and more reliable.

Regarding this application, the primary downside to both the Mega and Model B are their cost and size. Despite being relatively inexpensive (both under $50), the goal of a bicycle mounted system provides unique constraints where cost and size are heavily weighted. After testing the initial system and providing its validity, a smaller and lower cost system was constructed with the intention of better suiting the design goals.

1.2.2 Final Subsystem

The first change was to implement the Adafruit Feather System as a replacement for the Arduino Mega. Feather is a complete line of development boards that are stackable, expandable, and Arduino programmable (Adafruit 2017d). It is the platform for motor control, data logging, Global Positioning System (GPS) tracking, and application of the LIDAR sensor. Feather allows for the integration of most elements that are required for the project in predesigned chips. The master board is the Adafruit Feather 32u4 Adalogger that is an 'all-in-one' data logger with built-in USB and battery charging (Adafruit 2017a). One of two wings for the master board is the Adafruit Direct Current (DC) Motor + Stepper FeatherWing (Adafruit 2017c). It allows for the use of two bipolar stepper motors or four brushed DC motors (or one stepper and two DC motors) and is used here to control the 400-count stepper motor. The second wing is the Adafruit Ultimate GPS FeatherWing (Adafruit 2017b). It provides a precise, sensitive, and low power GPS module for location identification anywhere in the world. It can also keep track of time after synchronizing with satellites using an inbuilt Real-Time-Clock (RTC).

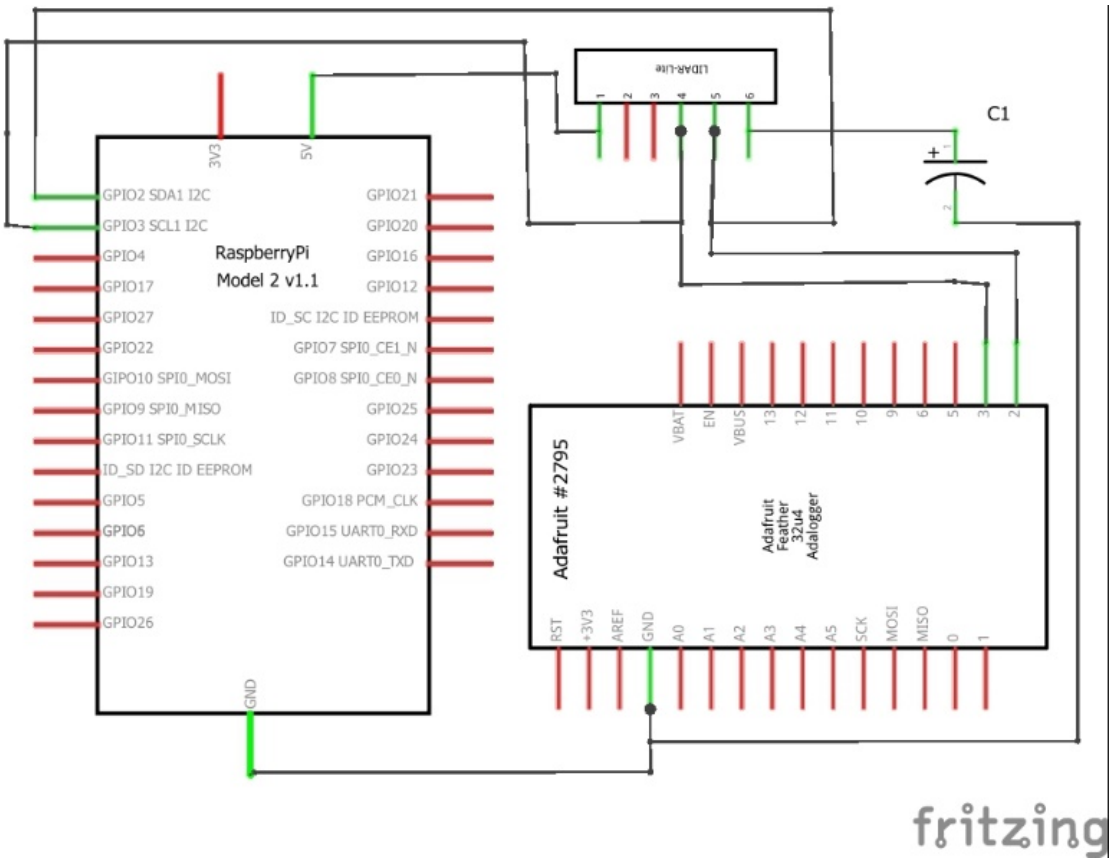

Figure 1.1 Wiring schematic for the final revision of the LIDAR system.

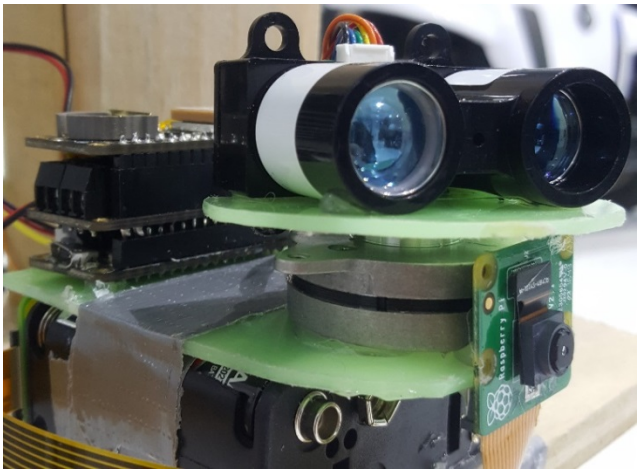

Figure 1.2 Garmin LIDAR-Lite v3 (right top) mounted to stepper motor (middle). The Raspberry Pi Cam (right middle) is connected via a ribbon cable to the Adafruit Feather stack (left middle).

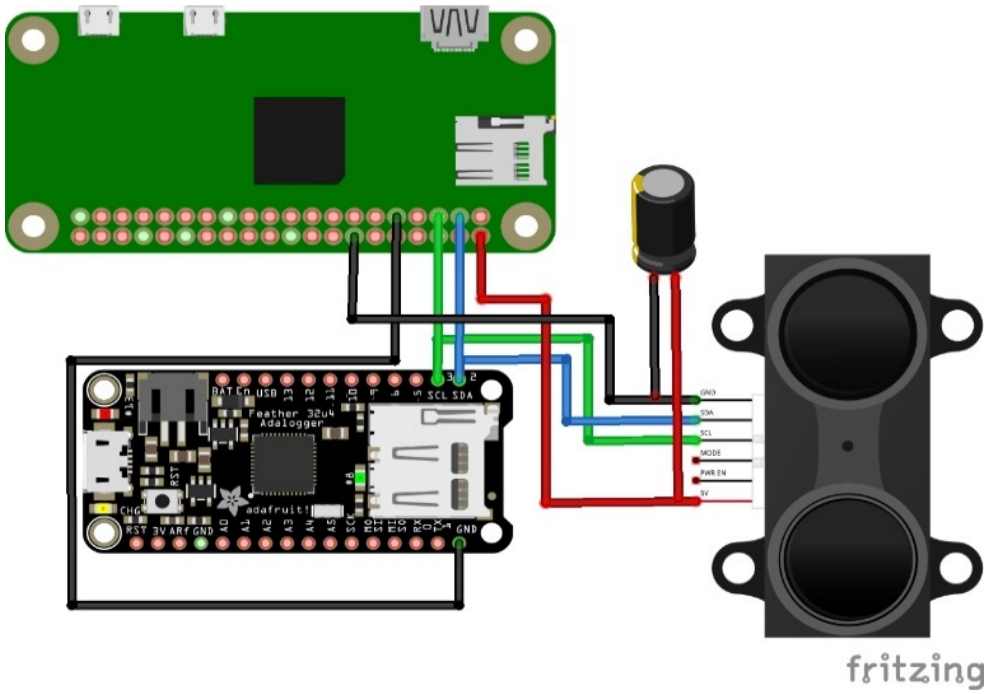

Figure 1.3 Simplified connection diagram joining the Raspberry Pi Zero (top) with the Feather stack (bottom left), a 680µF capacitor specified by Garmin to regulate LIDAR power requirements (middle right), and Garmin LIDAR-Lite v3 (bottom right).

Furthermore, the Raspberry Pi Model B was replaced in the second iteration with the Raspberry Pi Zero, currently the cheapest computer available at only $5. The Zero only has a single-core operating at 1 GHz with 512 MB of RAM while also not containing either Bluetooth or Wi-Fi modules. Because of the similarities in computer architecture between the Model B and the Zero, it was possible to transfer directly the Secure Digital (SD) card containing the memory and OS from the Model B. Because of this, the OpenCV software running on the Zero is similar to the OpenCV software developed for the Model B, with only a few minor changes to account for the low processing capabilities of the Zero. Communication between the Feather system and the Zero is achieved via the I^2C serial bus in the same way as the first testbed. Figure 1.1 provides the final connection schematic between the Zero and Feather data logger.

Since the LIDAR, Zero, Feather, and motor control all operate over I^2C, it is essential to keep separate message addresses in order to maintain stable and reliable communication. At this time, the GPS sensor is used solely for an accurate timestamp and communicates over a serial connection; hence, it does not interfere with the I^2C interface. This timestamp along with the most recent vehicle distance and angle measurements (discussed in the next section) are saved to an SD card for post-processing.

For the final system illustrated in figure 1.2 and figure 1.3, the stepper motor is powered by a battery pack consisting of eight AA batteries. This provides enough voltage for the motor controller to run reliably, while also lasting long enough for extended testing. This battery pack is connected to the motor controller via a 9 VDC socket connector so power can be quickly cut off when testing is not taking place. The stacked Feather system runs on a 3.7 VDC lithium polymer battery (DataPower (DataPower Technology Limited 2015)) using a SubMiniature version A (SMA) connection integrated onto the Adalogger board. A 5 VDC USB powers the Zero from the 4000 mAh, 5 VDC external battery (Prime Line PL-1365), initially intended for charging cell phones and tablets.

1.2.3 Vehicle Recognition

Vehicle recognition is achieved via the OpenCV software package, as previously mentioned, with the authors' code provided in the following reference (Zolotor 2017). When the Zero starts, a Python script begins and runs in the background. Then, the vehicle detection program begins, and the video stream from the Pi Cam opens. Because the camera is moving with the bike, a background subtraction algorithm cannot be used. Instead, a cascade (via an Extensible Markup Language (XML) file) is loaded into the program. This cascade is an image classifier that was trained by feeding over a thousand positive and negative samples of cars

(Fergus, Perona, and Zisserman 2001). Subsequently, each frame of the video stream is passed to the classifier, and if a car is found, the car's attributes are added to a list. Specifically, the location of each car from the left-hand side of the screen is converted to an angle in degrees and added to a list of angles. The list is sorted, and the smallest angle is saved to a text file. This angle has +/- 1° of uncertainty due to the non-uniform curvature of the camera's lens. Then, the Python script looks for a change in the angle stored in the file and sends it to the Feather (as an integer value proxy) over I^2C if one is found. These values determine if a car is expected; hence, a Boolean value of one (otherwise zero) is saved alongside the measurements from the Feather stack at the input angle. Since the stepper motor has no feedback indicating its current position, the program calibrates itself periodically by continuously rotating in one direction on a low torque setting, butting up against a stopper at a known angle.

In an effort to decrease the number of false positives, lane detection can be implemented, providing a region of interest to search for cars. If a lane is detected, its vanishing point will be calculated, and a region of interest will be defined: the area bounded by the lines tangent to the outermost lane markings and the road's vanishing point. All area outside of this region will become black in the frame, and then it will search for cars in this new frame with a higher tolerance. If a lane is unable to be detected, the program will run as described in the prior paragraph. For the testbed subsystem, lane detection was tried with limited success. Hence, it was removed for the final subsystem due to Zero's processing limitations.

1.3 Results and Discussion

Initial testing and troubleshooting of the testbed system were conducted by substituting a video stream of traffic in lieu of live video collected by the Pi Cam. This method expedited adjustments to both the OpenCV program sensitivity and the communication between the

subsystems. Subsequently, to demonstrate that the final system is capable of both detecting and determining the position of cars, a stationary test was conducted utilizing a single vehicle (2000 Infinity G20). Specifically, the front of the vehicle was directed toward the system in an otherwise empty parking lot, simplifying the process of taking manual measurements.

At a height of 3 feet, LIDAR measurements were found to be unreliable. This is likely due to the angle and reflectivity of the hood and windshield, which reflects light away from the light detection sensor. This is to be expected given the acknowledgment within the Garmin LIDAR manual that unless the sensor is normal to a specular surface, it will be incapable of taking accurate measurements. (Garmin International Inc. 2017). Furthermore, smooth reflective surfaces may not disperse light back towards the receiver (Beraldin, Blais, and Lohr 2010, Lichti and Skaloud 2010). However, by lowering the system height to 1.5 feet, the results became significantly more dependable. At this height, the front of the car is at an angle more conducive to the LIDAR sensor, subsequently allowing for accurate measurements.

Table 1.1 Measurement data from the stationary test that describes the vehicle position as found by both direct measurement with a measuring tape (columns 2 and 3) and as determined by the LIDAR system (columns 4 and 5).

Test	Measured Distance [in]	Measured Angle [deg]	LIDAR Distance [in]	Raspberry Pi Angle [deg]
1	292.3	40.95	298	41
2	322.0	57.67	334	58
3	227.8	18.16	226	18
4	152.8	11.60	155	15

Test	Measured Distance [in]	Measured Angle [deg]	LIDAR Distance [in]	Raspberry Pi Angle [deg]
5	497.0	31.33	509	31
6	385.0	31.70	395	30
7	259.0	31.10	265	34
8	673.1	32.04	660	31
9	623.9	37.54	588	39
10	119.9	47.57	130	48

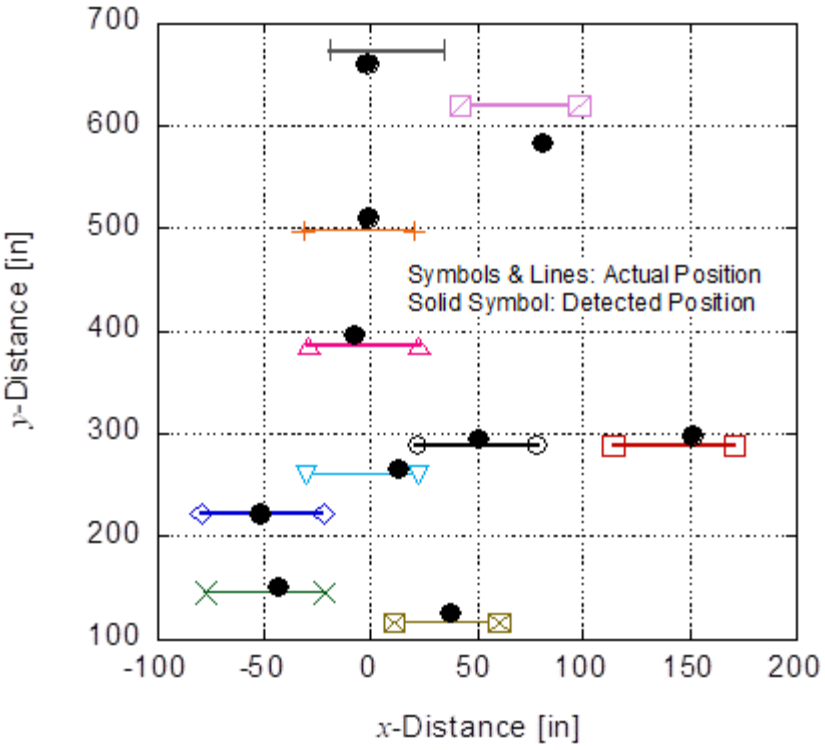

Figure 1.4 Position plot of the stationary test. Points indicate the coordinates of the car as determined by the LIDAR system. The lines show the frontal span of the vehicle as measured directly.

Data were then recorded for ten different car positions as a function of distance from the LIDAR system to the center of the vehicle as shown in table 1.1. The direct distances and angle measurements were found by finding the x and y distances to the vehicle using a tape measure and exercising Pythagorean's theorem and trigonometry, respectively. The LIDAR distance in this table is the distance to the vehicle as determined by the corresponding sensor. Furthermore, the OpenCV program determined the Raspberry Pi Angle measurements in this table. Figure 1.4 plots these positions on an x-y plane and it is important to note that the frontal area of the car cannot be accurately described as a single point. Hence, in figure 1.4 a line is drawn between the measurement taken at the leftmost and rightmost positions of the front of the vehicle. Moreover, solid circles in this figure provide the positions of the vehicle as determined by the LIDAR system. Overall, the LIDAR system appears to find the vehicle successfully. Moreover, accuracy can be determined by the proximity of each point in comparison to the front center of the vehicle. On average, the LIDAR system found the center of the vehicle with 82.3% and 96.7% accuracy in the x- and y-directions, respectively. Here, the deviations between the indirect and direct measurements may be a function of the curvature of the front of the car along with the relative reflectivity of the lights and grill.

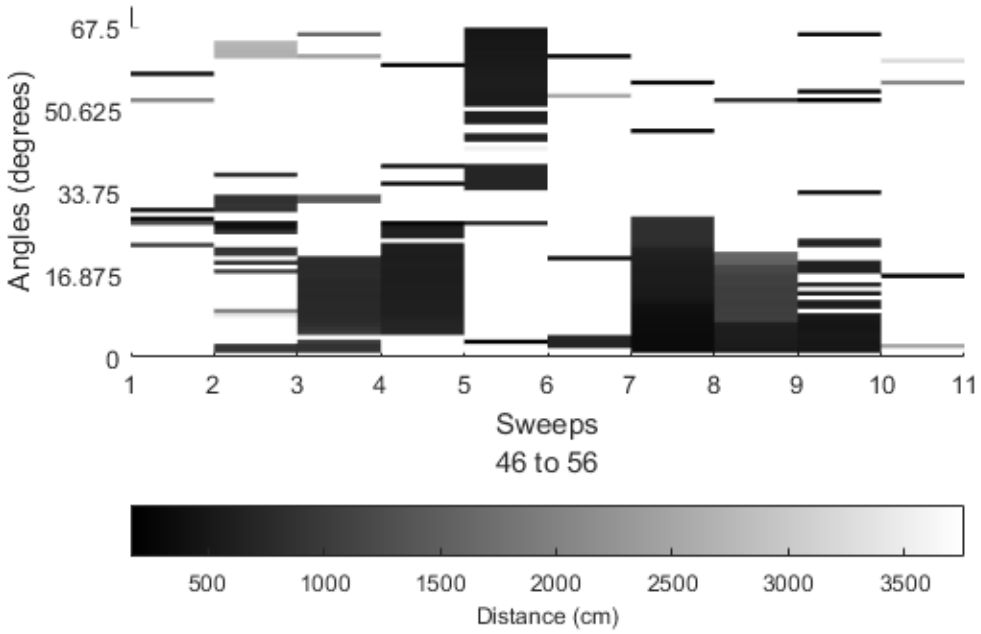

Figure 1.5 LIDAR system data while being followed by a car during the period of 106 to 131 seconds.

Subsequently, dynamic tests helped to evaluate the performance of the system in active situations. The first test consisted of mounting the LIDAR system on the back of an e-bike at a height of 1.5' and riding through a parking lot past parked cars while also being followed by a car (2000 Oldsmobile Bravada). The path chosen consisted of riding up one lane of the parking lot and then down the adjacent lane. During the ride, the LIDAR sensor rotates through 75 unique angle measurements from 0 to 67.5°, where 33.75° is designated as the angle directly behind the e-bike. Henceforth, a sweep refers to the LIDAR rotating from 0 to 67.5°, then returning to 0°.

Figure 1.5 provides insight into the operation of the LIDAR system. Specifically, as the LIDAR-Lite performed a sweep (*x*-axis), it captured the distance of surrounding vehicles (color bar) along with their relative angle to the e-bike (*y*-axis). Furthermore, darker bars represent

objects nearer to the LIDAR system. Of note, data below 1 m was removed because the manual states there is non-linearity in measurements below this level and visual inspection via OpenCV photos did not find anything. Overall, determination of the relative movement of objects occurs through the change in angle and distance of these objects between consecutive sweeps. However, unlike its potential fixed location on an automobile, the relative orientation of the LIDAR camera changes dynamically during a ride as the e-bike jostles, rolls, and sways. Hence, reviewing the data from figure 1.5 illustrates that it becomes difficult to determine definitive locations of vehicles without visual cues (e.g., OpenCV photos). In other words, the three-dimensional space continuously varies as the horizon seemingly tilts. Therefore, differentiating between objects becomes onerous and the implementation of an accelerometer or gyroscope synced to the LIDAR data should help with fidelity.

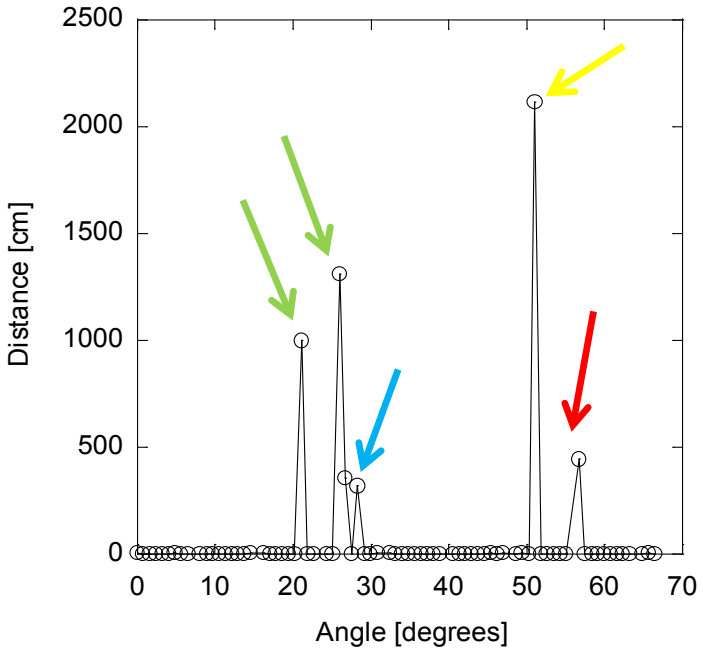

Figure 1.6 LIDAR system data from the 46th sweep, or at approximately 106 seconds.

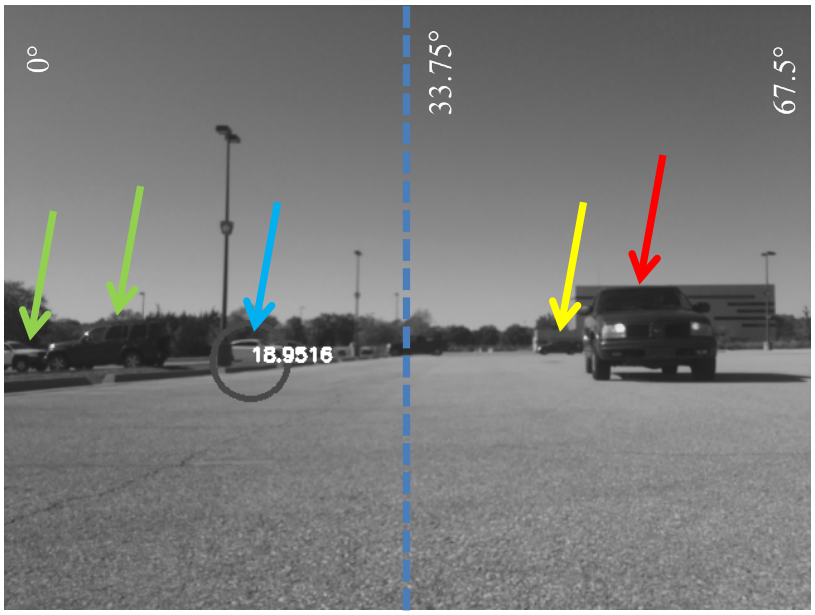

Figure 1.7 Photo taken by the OpenCV software at 106 seconds into testing (about 4.05º into the 46th sweep). In this frame, the central vehicle is missed by the software but is picked up by the LIDAR.

In order to improve clarity behind how the LIDAR system operates, figure 1.6 provides an array of data during the small window of time corresponding to the first sweep in figure 1.5. Each sweep takes approximately 2.45 seconds, which may lead to discrepancies while measuring the same object. Moreover, this makes comparisons to photo taken by the Pi Cam difficult. Specifically, this camera takes photos nearly instantaneously as compared to the duration of a sweep; however, the OpenCV software is not quick enough to analyze photos at the same frequency as the LIDAR sensor. For example, the image corresponding to approximately the same time as sweep 46 is shown in figure 1.7 with approximate LIDAR angles indicated on the picture. Here, sweep 46 begins before taking the photo and the LIDAR sensor returns zero values when it does not find an object within its range of 40 m.

Figure 1.8 Photo taken by the OpenCV software at 239 seconds into testing. Limitations in sensitivity lead to inaccurate vehicle identification. Example of bicycle sway and tilt shown.

Figure 1.9 Photo taken by the OpenCV software at 201 seconds into testing. Not all objects identified OpenCV are vehicles.

The first two peaks from the left in figure 1.6 appear to correspond to the cars (green arrows) on the left edge of figure 1.7 and the tallest peak should link to the distant car (yellow arrow) just left of the Oldsmobile Bravada. As anticipated, the LIDAR sensor captures other objects beyond vehicles, as it appears the light pole next to the vehicle highlighted with a blue arrow appears in these data. In addition, the distance measurements compared with the photo for the blue and green highlighted vehicles do not seem correct. This may be because the LIDAR sensor is sweeping through the range and the photo only illustrates a snapshot of this sweep. For instance, the LIDAR sensor sweeps from left to right; hence, the green highlighted vehicles would be closer initially as the e-bike travels away from them. Moreover, while the LIDAR sensor appears to provide fidelity in picking up all vehicles, the OpenCV software misses all but one since its sensitivity is too low and insufficient processing speed was provided. Hence, future work should ensure that LIDAR sensor recognition and OpenCV data collection occur at the same frequency.

Interestingly, because of fixed system integration on the e-bike, both the LIDAR sensor and OpenCV recognition software will experience the same roll as the bicycle. However, this should not affect the photos as much as the LIDAR measurements because photos cover a much larger area. Instead, the roll and pitch of the e-bike will make the LIDAR sensor less likely to detect an object at the bounds of its rotation (see fig. 1.8). Furthermore, the OpenCV system will be less capable of detecting vehicles when not level, because of limitations in learning the appearance of a vehicle. Similarly, as mentioned prior, the sensitivity set on the OpenCV system has a substantial impact on its vehicle detection accuracy. Limitations on computing power led to a reduction in sensitivity of the OpenCV system. Hence, it misses detecting some vehicles (e.g., fig. 1.7 and fig. 1.8), and occasionally other objects are labeled as vehicles, as seen in figure 1.9.

These issues should not be as prevalent for automotive LIDAR usage; however, systems designed for bicycles and e-bikes will need to account for these facets because of their unique constraints (e.g., low cost and small size). Because of the found limitations, it was decided to forgo experimentation in real collision scenarios and instead focus on enhancement of the system for future efforts.

1.3.1 System Diagnosis

A significant drawback of the camera vision system running on the reduced processing power of the Raspberry Pi Zero is the relatively low framerate that the system is capable of handling. Specifically, the video stream was processed at a rate of one frame every 2.1 seconds during testing. A processing rate this low can lead to the issue of insufficient reaction time for the rider to vehicular threats. Furthermore, with a LIDAR range of 40 m, any vehicle traveling over 19 m/s (42.5 mph) relative to the e-bike could potentially move through the entire detection range without being sensed.

Ideally, when the subsystems are effectively communicating, the video feed sends angle measurements directly to the Arduino-based subsystem for subsequent saving. However, communication between the Feather and the Zero proved to be unreliable and would often cause the Feather system to crash. Hence, testing did not employ the interface between these subsystems, consequently requiring a combining of data in post-processing as mentioned prior.

Moreover, it takes an average of 2.45 seconds for the LIDAR system to make all 75 unique angle measurements; i.e., distances at each angle are found only once every 2.45 seconds. This low repeat frequency is undesirable for the same reasons the low framerate is detrimental: a vehicle can go undetected for too long. Furthermore, the relatively low memory and processing speeds of the Feather contributes significantly to the process times. Specifically, data are written

to the SD card once every 15 measurements. Therefore, it takes on average 0.07 s to save the data to the SD card, which means that 0.4 s of every rotation can be eliminated by using a microcontroller with more embedded memory.

Finally, the limiting factor in the system is the maximum frequency of the LIDAR-lite v3 at 500 Hz. Hence, the minimum time possible for the system to complete full cycle would be 0.15 s. Since it takes approximately 0.5 s for the stepper motor to traverse 75 angles, the system is currently set up to complete a sweep in less than one second. However, processing the data leads to the greater time mentioned prior. Therefore, to equate the system speed to that of the LIDAR sensor, a different stepper motor is required. To increase speed beyond this level, multiple LIDAR sensors would need to be employed, or the number of measurements per sweep would have to be lowered.

1.4 Conclusions

Safety concerns deter many people from cycling, but currently available solutions have difficulty gaining traction because of unique constraints. Specifically, riders wish to employ affordable and unobtrusive solutions that do not negatively influence their experience. In this area, LIDAR-based systems provide an opportunity to improve safety dramatically from the rear because of their high speed and precision monitoring capabilities at relatively low computation requirements. Furthermore, recent advances in sensor technology are beginning to bring costs into a range that allows implementation on all bicycles and e-bikes.

As a result, this chapter described the lowest cost system possible using the Garmin LIDAR-Lite v3 module interacting with Adafruit Feather boards and a Raspberry Pi Zero as the microcontroller and computer, respectively. This system can measure the distance of a stationary vehicle accurately after training the OpenCV software package running on the Zero. However,

subsequent dynamic tests found limited success as the LIDAR sensor was able to find surrounding vehicles; whereas, the limited processing capability of the Zero dramatically reduced OpenCV's ability. Moreover, data provided by the LIDAR sensor are difficult to analyze to find moving vehicles given its recognition capacity; i.e., it notices many items beyond just vehicles. Furthermore, at a proposed cost of $280, the system (while cheaper than many commercial LIDAR sensors) is still too expensive for use by the cycling community.

Overall, this novel integration and application of existing microcontrollers in tracking vehicles has applicability beyond cyclist safety and into areas, such as unmanned aerial vehicles and robotics, where object detection can be crucial. In addition, this study demonstrates that the employed methodology has limited success when price constraints (currently) cause reduced computational capabilities. Hence, future efforts should enhance processing capabilities to provide a better linking of the LIDAR sensor and OpenCV recognition software in order to remove false vehicle positives while endeavoring to do so at a minimum of expense.

Chapter 2 Expanding the Use of Inexpensive LIDAR Systems

As indicated in the previous chapter, limited success was achieved in building a low-cost LIDAR system for e-bike usage in order to track rearward threats. Moreover, upon discussion with MATC personnel, it was learned that inexpensive LIDAR systems could provide numerous other opportunities to improve safety in the transportation arena. In specific, mobile applications of a LIDAR system can provide a wealth of data required for Highway Performance Monitoring System (HPMS) reports including, but not limited to: traffic information to mitigate roadway delays, accident/crash investigation, soil and rock slope stability, flood risk mapping, pavement quality monitoring, and clearance data for highway overpasses and power lines (Williams et al. 2013). This information can lead to safety improvements for transportation workers, the traveling public, and the general public in Region 7. Furthermore, employing an effective and mobile LIDAR system can help highlight the major stressors that affect safety performance. Hence, it can positively enable the Research Topics of the MATC by working to provide information pre-crash to reduce risks and after accidents as part of post disaster inspection systems.

As a result, three synergistic activities are currently underway that expand on the original efforts in order to enhance the benefit for the MATC:

1. Development of 2nd Generation Mobile LIDAR System
2. Commercial LIDAR Systems Research
3. Undergraduate Capstone Design LIDAR Project

In brief, the lessons learned in Chapter 1 are being employed in the development of a second-generation mobile LIDAR system in order to accomplish the original research proposed. Furthermore, because of the difficulties involved in fabricating the system in Chapter 1, it was decided to re-review commercial LIDAR systems available in order to understand their

capabilities and determine if an “out-of-the-box” system can deliver similar performance at a comparable cost. Finally, in order to train the next generation of students, the PI is overseeing a capstone design project in the Department of Mechanical Engineering where students are building a stationary LIDAR system to investigate the data coming from the Garmin LIDAR-Lite v3 sensor along with how to process it appropriately. In the following sections, each activity is explained in detail.

2.1 Development of Second Generation Mobile LIDAR System

In Chapter 1, the System Diagnosis section highlights the issues found with the original mobile LIDAR system along with how to improve its capabilities. As a result, the students who contributed to its development subsequently began improving the system in order to accomplish the original research objectives. Unfortunately, after nearly completing its fabrication, these students determined they were unable to continue on this project given their other responsibilities and impending graduation. As a result, the PI hired a new graduate student from an underrepresented minority in engineering to take over, learn the system, and finish its construction. Hence, the majority of the most recent work involves this student learning the hardware and coding language, developing wiring diagrams, researching comparable commercial LIDAR systems, and working to collect data with the second-generation LIDAR system.

2.1.1 Updated Hardware

The second-generation LIDAR vehicle detection system is comprised of a Garmin LIDAR-Lite v3, an Arduino Mega2560, and a Raspberry Pi 3 Model B. A Pi Camera v2 supplies video feed to the Raspberry Pi 3 over the Pi Cam Ribbon. The Garmin LIDAR-Lite v3 is mounted on a SparkFun Stepper Motor that is controlled by the Arduino Mega2560 using the

SparkFun EasyDriver. Power is supplied to the Arduino and Pi by a 9 VDC lithium ion battery pack (fig. 2.1).

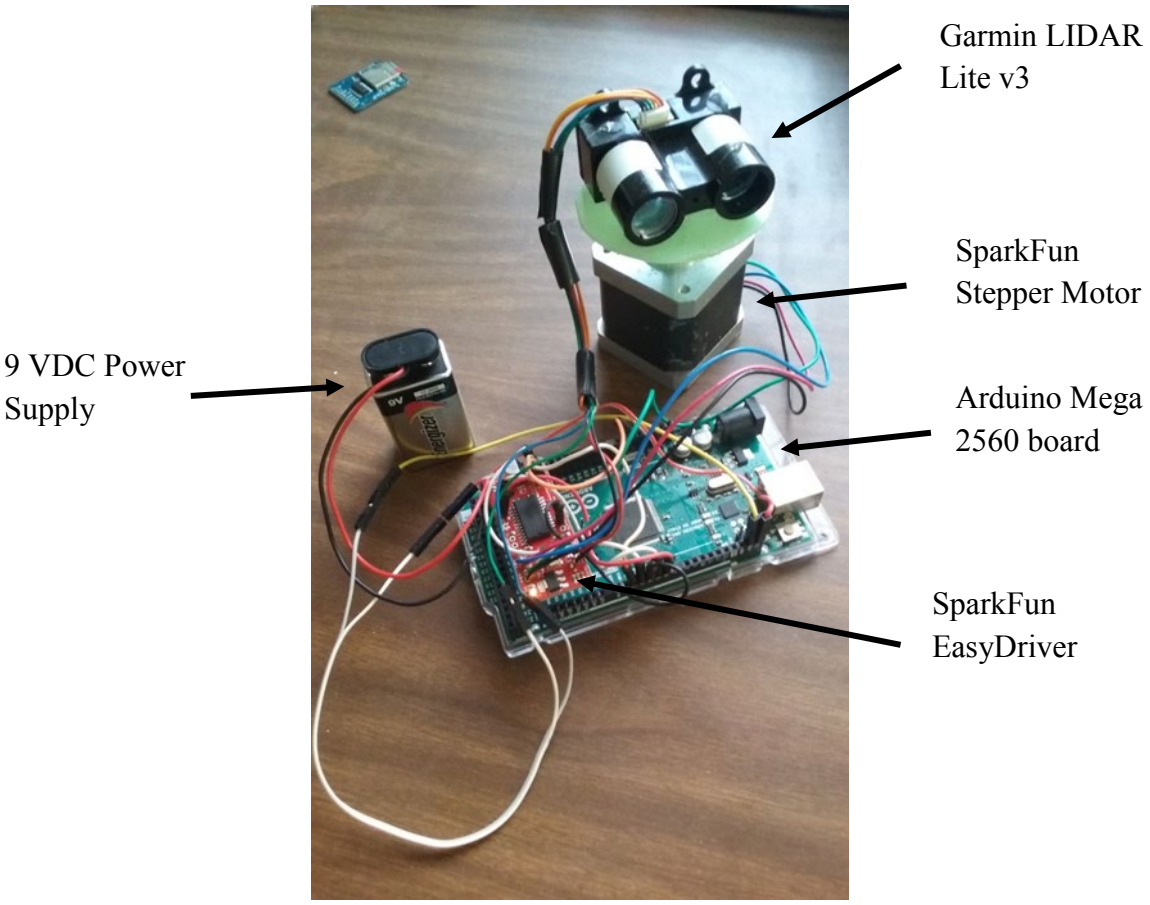

Figure 2.1 Second-generation LIDAR Lite v3 System.

The choice of the Arduino Mega2560 board enhances the processing of the signals from the LIDAR camera because of its greater computational capabilities and larger Random Access Memory (RAM) capacity when compared to other Arduino boards. The LIDAR-Lite v3 is wired to the Arduino board following the standard Arduino Inter-Integrated circuit (I^2C) wiring connection diagram found in the LIDAR-Lite v3 Operation Manual and Technical

Specifications. However, there is a slight change in this endeavor because of the inclusion of the SparkFun Stepper Motor and Raspberry Pi board. Instead, the power supply runs through the Raspberry Pi instead of the Arduino Mega2560 (fig. 2.2).

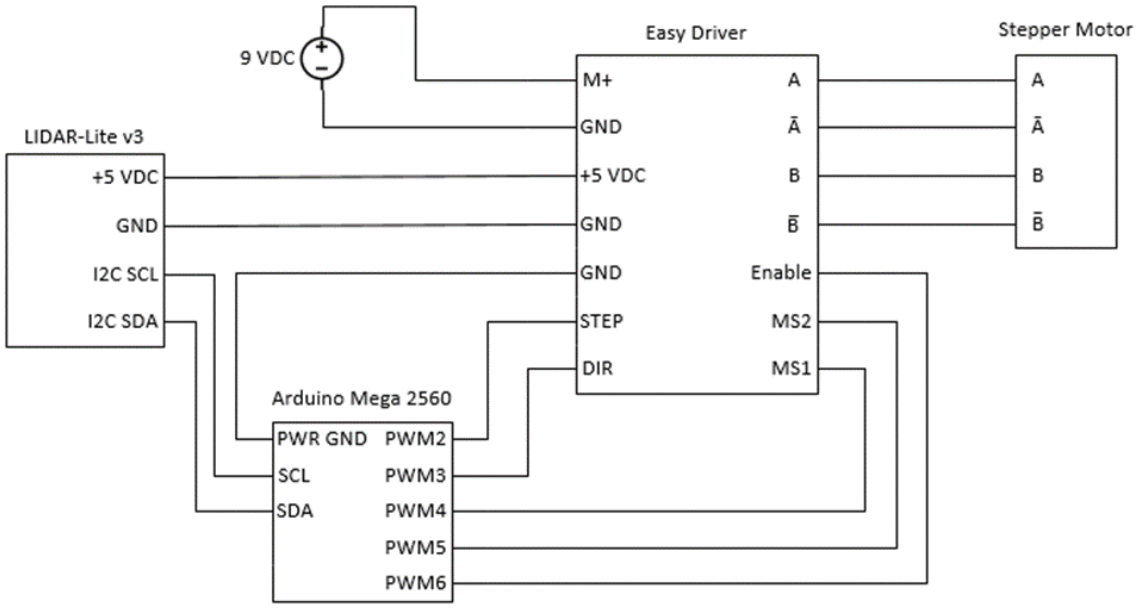

Figure 2.2 Pinout diagram of second-generation LIDAR System.

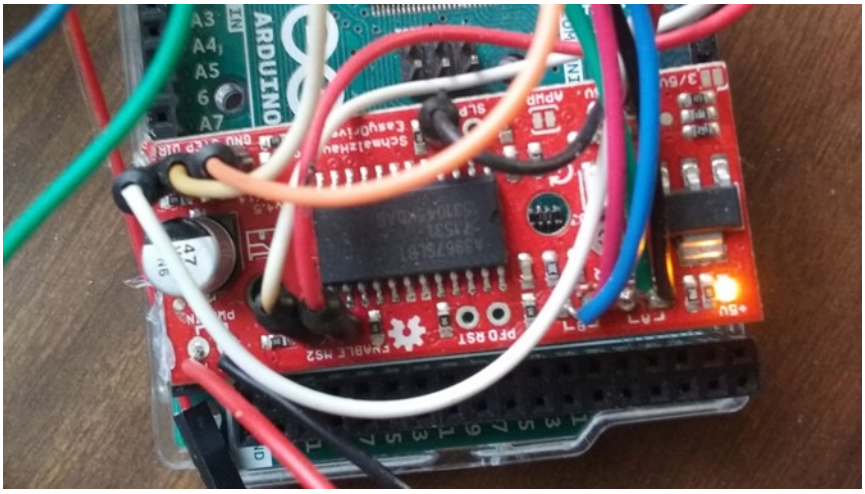

Figure 2.3 Close-up of the connected EasyDriver board connected to power.

All power to the system is supplied from the 9 VDC battery through the SparkFun EasyDriver at the "power in" connections, subsequently drawing 2 A. The EasyDriver then lights up a light-emitting diode (LED) labeled +5V to indicate it is supplying 5 VDC to power the LIDAR camera at 0.135 A while in continuous operation (fig. 2.3). Both the STEP and DIR pins on the EasyDriver are logic inputs. This logic indicates a change from low (0 VDC) to high (3.3 or 5 VDC) of the STEP signal will cause the motor to turn one step. The DIR pin controls the direction the stepper motor will turn depending on whether the input state is low or high. The stepper motor is connected to the EasyDriver through the pins A and B along with their respective halves of the bi-polar motor coils. The input pin ENABLE is also a logic pin, and determines if the motor will be controlled by the EasyDriver. Put simply, it is the on/off switch for the motor. The MS1 and MS2 pins are additional logic input pins. They work together and their particular combination of low and high signals determines the micro-step resolution of an eight, squatter, half, or full step (SparkFun 2018). The I^2C connections on the LIDAR-Lite v3 camera operates at 3.3 VDC up to a maximum of 5 VDC. The SCL line connects to the Arduino board and acts as a clock, keeping track of when data signals are taken. In addition, the SDA connected to the Arduino is the corresponding 7-bit data line responsible reading and writing information to the board (Arduino 2018). Finally, the Arduino Mega2560 board controls the signals sent to the EasyDriver and LIDAR camera with the programming discussed further in Section 2.1.2.

2.1.2 LIDAR Software

The software currently in use employs the Arduino Integrated Development Environment (IDE) to program the Arduino board using "sketches" (aka code) while controlling the stepper motor and the LIDAR camera. Here, specific libraries (i.e., Wire, LIDARLite,

ADAfruit_MotorShield, SPI, and SD) are used to effectively communicate with the entire LIDAR system. In addition, there are several separate Arduino sketches that each contain a unique function needed to run the main sketch.

LidarProjectMain §

```
#include <Wire.h> //Allows communication using SDA and SCL pins
#include <SPI.h> //Communicate with external interfaces
#include <SparkFunLSM9DS1.h> //Integrate  SPI or I2C device
#include <LIDARLite.h>

LSM9DS1 imu;
LIDARLite myLidarLite;

#define LSM9DS1_M 0x1E // Would be 0x1C if SDO_M is LOW
#define LSM9DS1_AG  0x6B // Would be 0x6A if SDO_AG is LOW

#define stp 2
#define dir 3
#define MS1 4
#define MS2 5
#define EN  6

#define PRINT_CALCULATED

#define DECLINATION -2.12


int stepperMotorAngle = 0;
int counter = 0;
int lDistance = 0;

char user_input;
int x;
int y;
int state;
```

Figure 2.4 LidarProjectMain.ino sketch.

The first sketch is called the LidarProjectMain (fig 2.4) and assigns the STEP, DIR, MS1, MS2, and ENABLE signals to the correct pins on the Arduino board while establishing several constants to be used throughout the other sketches.

Setup §

```
void setup()
{Serial.begin(115200);
  Serial1.begin(115200);
  Serial2.begin(115200);

  Serial.println("1");
   pinMode(stp, OUTPUT);
  pinMode(dir, OUTPUT);
  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(EN, OUTPUT);
  resetEDPins();

  myLidarLite.begin(0, true); // Set configuration to default and I2C to 400 kHz
  myLidarLite.configure(3);

  //imu.settings.device.commInterface = IMU_MODE_I2C;
   //imu.settings.device.mAddress = LSM9DS1_M;
  //imu.settings.device.agAddress = LSM9DS1_AG;

   Serial.println("2");
Serial.println("3");
}
```

Figure 2.5 Setup.ino sketch.

The next sketch, Setup (fig 2.5), defines the baud rate of the controller and all communications. Moreover, it defines the STEP, DIR, MS1, MS2, and ENABLE signals as outputs from the Arduino Mega2560 board.

Stepper §

```
void stepperControl()
{
  user_input = '4'; //Read user input and trigger appropriate function
      digitalWrite(EN, LOW); //Pull enable pin low to allow motor control
  for(x= 1; x<5; x++)  //Loop the forward stepping enough times for motion to be visible
  {
    //Read direction pin state and change it
    state=digitalRead(dir);
    if((counter%360) >180)
    {
      digitalWrite(dir, LOW);
    }
    else
    {
      digitalWrite(dir,HIGH);
    }
  }
      digitalWrite(stp,HIGH); //Trigger one step
      delay(1);
      digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
      delay(1);


  resetEDPins();
  return 0;
}

 void resetEDPins()
{
  digitalWrite(stp, LOW);
  digitalWrite(dir, LOW);
  digitalWrite(MS1, LOW);
  digitalWrite(MS2, LOW);
  digitalWrite(EN, HIGH);
}
```

Figure 2.6 Stepper.ino sketch.

Subsequently, the stepper motor is controlled with the Stepper sketch (fig. 2.6). After enabling control over the motor, a loop moves the motor one-step at a time forward or backward depending on the current position in the sweep arc. Of importance, each motor step requires the STEP signal to reset. Once the loop is finished, all the pins are set to low except for ENABLE to prevent any further control over the motor.

LIDAR §

```
int lidarDistance()
{
  if (counter%100 == 0)
  {
  return myLidarLite.distance();
  }
  else
  {
   return myLidarLite.distance(false);
  }
}
```

Figure 2.7 LIDAR.ino sketch.

The next sketch interacts with the LIDAR-Lite v3 device (fig. 2.7), which simply gathers distance data for each sample taken.

Loop §

```
void loop()
{
 Serial.print("Angle: ");
  stepperControl();

 Serial.print("Distance: ");
 lDistance = lidarDistance();
 if (lDistance < 100)
 {
  Serial.println("LOOOK OUTT!!!");
 }
 Serial.println(lDistance);

  Serial.println("Acceleration: " +accelerationData());

  Serial.println();

  counter++;
}
```

Figure 2.8 Loop.ino sketch.

Subsequently, the next step involves the Loop sketch (fig. 2.8). This is a single loop to alert the user if the LIDAR camera identifies an object closer than the set safe distance and at what angle from the camera it is located. Moreover, it displays the system's acceleration data and increases the counter.

LidarProjectNoGPS §

```
#include <Wire.h>
#include <LIDARLite.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"
#include <SPI.h>
#include <SD.h>

// Create the motor shield object with (0x61 i2c adress)
 Adafruit_MotorShield AFMS = Adafruit_MotorShield(0x61);

// Connect a stepper motor with 400 steps per revolution (.9 degree)
// to motor port #1 (M1 and M2)
Adafruit_StepperMotor *myMotor = AFMS.getStepper(400, 2);

LIDARLite myLidarLite;

const int arcSweep = 75;
const int chipSelect = 4;
int collectedAngle = 0;
int currentStepPosition = 0;
int newStepPosition = 0;
int measuredDistance = 0;
int loopCounter = 0;
int newAnglePosition = 0;

String dataString = "Angle,Distance,Time";

void setup()
{


  // set up Serial library at 115200 bps
  Serial.begin(115200);
 delay(2000);
  Serial.print("Initializing SD card...");
  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println("card initialized.");
```

Figure 2.9 LidarProjectNoGPS.ino sketch.

```
  delay(2000);
//  LIDAR lite setup
  myLidarLite.begin(1, true); // Set configuration to default and I2C to 400 kHz
  myLidarLite.configure(3);  //Maximum range. Uses 0xff maximum acquisition count.

  //motor control setup
  AFMS.begin(1600);  // default maximum frequency of 1.6KHz PWM
  myMotor->setSpeed(255);  // Maximum motor speed 255

}

void loop()
{
    if (newStepPosition < arcSweep && newStepPosition >= 0)
    {
          myMotor->onestep(BACKWARD, DOUBLE);
          measuredDistance = myLidarLite.distance();
          newAnglePosition = newStepPosition / .9;
          dataString = dataString + String(newAnglePosition) + ","  + String(measuredDistance) + "," + String(millis()) + " " + "\n";
          Serial.println(dataString);
          newStepPosition++;
          if(newStepPosition%15 == 0 && newStepPosition != 0)
          {
             saveData(dataString);
          }
    }
    else
    {

     saveData(dataString);
      myMotor->step(arcSweep+20, FORWARD, DOUBLE);
      newStepPosition = 0;


}
loopCounter++;
Serial.println(loopCounter);
}

boolean saveData(String data)
{
   File dataFile = SD.open("test16.txt", FILE_WRITE);
     // if the file is available, write to it:
     if (dataFile)
     {
       dataFile.println(data);
       dataFile.close();
       // print to the serial port too:
       Serial.println(data);
       dataString = "";
       return true;
     }
     // if the file isn't open, pop up an error:
     else
     {
       Serial.println("error opening datalog.txt");
       return false;
     }
}
```

Figure 2.10 LidarProjectNoGPS.ino sketch (cont.).

The last bit of programming involves the LidarProjectNoGPS sketch that combines most of the other sketches while collecting data (fig. 2.9 and fig. 2.10). In specific, this code connects the motor shield, sets up the stepper motor conditions, and moves the motor one step at a time

while collecting and saving data samples each time. Finally, the data are organized into a text file and then saved on a connected microSD card.

2.1.3 Current Issues and Next Steps

Currently, some of the connections between the components of the system are delicate and occasionally come apart. Once the system has demonstrated the ability to work as intended, the next step will be to solidify these connections and make the system more secure and compact. This will involve soldering wires to the correct types of connectors. Then, a case will be made to contain the Arduino and EasyDriver boards next to the stepper motor.

Presently, the LIDAR camera is not communicating as it should with the microSD card; hence, data cannot be taken. Moreover, the sketch that runs the system is not currently configuring the LIDAR camera or the stepper motor. Due to the student's unfamiliarity with Arduino hardware and sketches, it is unknown if the issue is with the hardware connections between the Adafruit MicroSD card breakout board and the rest of the system, or a detail in the code is being overlooked. As a result, the student is exploring numerous options to aid in her knowledge of the system and provide a greater familiarity with the Arduino platform.

After the issues with the system and sketches have been alleviated, data collection using the second-generation LIDAR system will occur. Transformation of the raw data saved to the microSD card into a .las or .laz file will be the next step. This will generate a three-dimensional (3-D) model of the data using point cloud software. It has been established that LIDAR point cloud data can accurately recreate 3-D space with most efforts in this area employing this option to map large areas of land from above (Basgall, Kruse, and Olsen 2014). However, this project's focus is to model moving objects on a smaller scale that will most likely result in a reduced level of detail. Therefore, it will be important to be aware of the trade-off existing between sampling

rate and the range of data acquisition during future testing. In specific, when the LIDAR-Lite v3 camera is set at a higher sampling rate, the maximum signal range decreases via an exponential relationship. In addition, at fast sampling rates, the sensitivity is reduced by an unidentified amount (Garmin International Inc. 2017); hence, understanding the range as a function of sampling rate will need to be considered.

2.2 Commercial LIDAR System Research

In order to assess the success of the second-generation LIDAR build, it will be compared to the performance of a commercial LIDAR system. Since this endeavor will be utilizing LIDAR on an e-bike during transit, some aspects of performance are more important than others. In specific, these criteria include being lightweight and relatively easy to use while additionally being inexpensive and having the ability to integrate with point cloud software (table 2.1). Moreover, the types of commercial LIDAR systems under consideration must be self-contained and not require additional components to function as needed. Reviewing the industry, most commercial LIDAR systems fall into one of two categories: small and cheap, or expensive and industrial. The small systems are lightweight and inexpensive enough to be considered viable alternatives; however, they have a shorter signal range that could be a safety concern for bicyclists. Instead, the other type of commercial LIDAR systems have a long range finding capability and are of high quality. Unfortunately, they are significantly heavier than suitable for biking and quite expensive. One thing that almost all commercial systems have in common is that they employ an accompanying point cloud software tool while providing documentation about data processing. In contrast, since the LIDAR-Lite v3 device was designed to be connected to microcontrollers (i.e., Arduino Mega2650), it has substantially more documentation about wiring and connectivity but significantly less about data processing.

Table 2.1 Comparison of assembled second-generation and commercial LIDAR systems.

LIDAR System	Maximum Range (m)	Maximum Sampling Rate (Hz)	Weight (gm)	Est. Cost ($)
Second-generation system	40	500	400	175
YDLIDAR X4 (YDLIDAR 2018)	10	5000	180	100
RPLIDAR A2M8 (Slamtec 2018)	8	8000	190	300
YellowScan Surveyor (YellowScan 2018)	100	300,000	1600	N/A
Velodyne (Velodyne LIDAR 2018)	200	1.2 Million	1000	5000

Investigating table 2.1, the smaller commercial LIDAR systems weigh less than the second-generation system developed in this report. However, they might be unsuited for use on an e-bike because of their limited scanning range. As a result, these small commercial systems will not identify objects until they are significantly closer to the e-bike with a 4-5× reduced range. Hence, this would not positively influence the safety of the rider. On the other hand, the industrial systems are definitely not suitable because of their larger size and cost. Therefore, it would be impractical to try implementing one of these options on the e-bike (recall the discussion in Section 1.1 revolving around consumer acceptance). In addition, if the second-generation system were to fail at some point, it would still be easier and cheaper to repair or replace a component rather than purchasing another commercial system.

2.3 Undergraduate Capstone Design LIDAR Project

Reviewing the early efforts for this project resulted in the PI wanting more students to become familiar with this topic area and LIDAR cameras in general. As a result, the PI is

overseeing a senior capstone design project in the Department of Mechanical Engineering involving four undergraduate students. Their goal is to build a stationary LIDAR system that can take data on campus and format it appropriately for point cloud software. More specifically, since a mobile application is underway in Section 2.1, it was decided by the PI to simplify the project for undergraduates in order to build a system that could be set-up at a specific location to capture transportation-related data. To accomplish this objective, the students were tasked with two outcomes:

1. Take Stationary LIDAR Data
2. Investigate & Utilize Point Cloud Software

First, the students must become familiar with the LIDAR camera and learn what output data are provided. Second, these data must be formatted properly in order for its implementation in a point cloud software program that can be used to provide the distance to objects. In the following sections, the current progress of the students with respect to these two outcomes are described.

2.3.1 Stationary LIDAR Data

The students' first task was to understand what data the LIDAR camera measures and the options of the device including, but not limited to range and output data file types. LIDAR cameras work by emitting short pulses of low frequency light from an emitter lens, and measuring the time it takes the light to return to the detector lens. Then, this time is used in a relatively simple calculation involving the constant speed of light to determine the straight-line distance to the detected surface. Subsequently, knowing this value in conjunction with identified horizontal and vertical angular positions determines the x, y, and z distance in relationship to the camera. Then, after a collection of points is found, they are combined together into a "point-cloud" with .las and .laz files the industry-standard binary format files that store LIDAR data.

Hence, by understanding the data output of the LIDAR-Lite v3 camera and converting it into a point cloud, one can generate a detailed 3-D image of the surroundings.

Similar to Section 2.1, the undergraduate students have decided to explore the use of both Arduino Mega2560 and Raspberry Pi 3B+ microprocessors to run the LIDAR system. Currently, the work is being accomplished using the Arduino Mega2560 because of its ease of connecting to the LIDAR camera (fig. 2.7); however, the students plan to use the Raspberry Pi 3B+ in the future because of its greater processing speed and power. Furthermore, in order to capture 3-D data, the students' have incorporated two servos; one moves in the *x*-*y* plane with the other moving in the *z* plane.

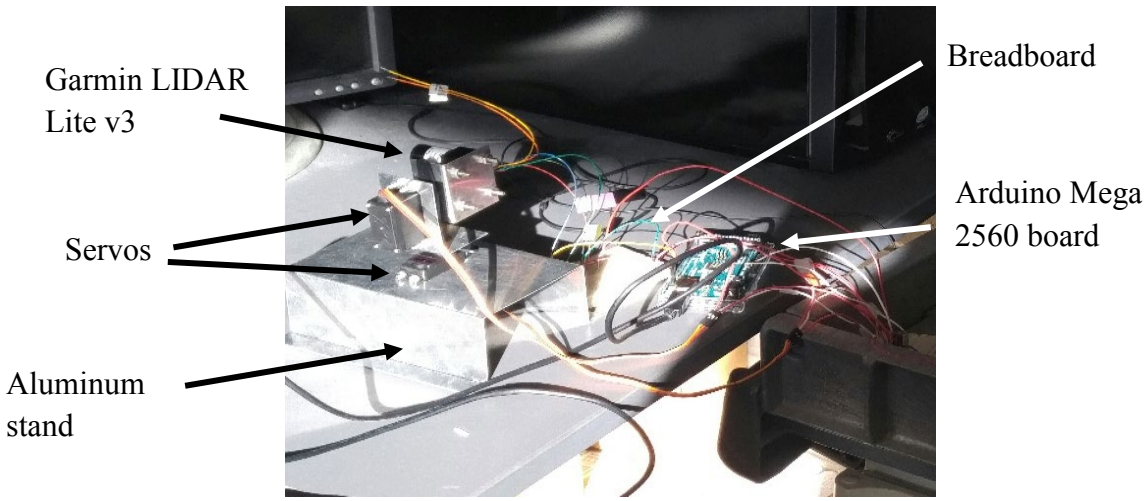

Figure 2.11 Current system built by the capstone design team in order to capture 3-D data

In figure 2.11, the current system as constructed is illustrated. The initial servomotors chosen are only able to rotate 180º in 1º increments and the cables connecting the *z*-axis servo to the LIDAR camera are quite short. As a result, this can cause the cables to disconnect unexpectedly if the breadboard and Arduino are not properly positioned with respect to the

aluminum stand. Future alterations to this system will fix this issue by securing a breadboard to the stand, subsequently reducing the strain in the cables.

2.3.2 Point Cloud Software

The goal of integrating the camera data with point cloud software is to generate a detailed 3-D view of the surroundings. Figure 2.12 illustrates the example output of a point cloud software program illustrating its ability to render an environment from which it is possible to pick transportation-related features (e.g., stop sign).

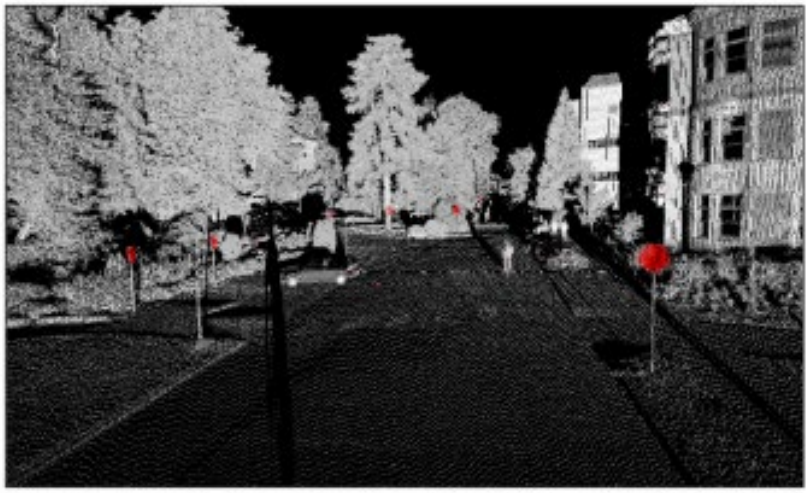

Figure 2.12 Example point cloud software outcome highlighting the stop sign in red that was extracted from the data (Williams et al. 2013)

For the point cloud software program, the students' chose Trimble RealWorks because of its ability to operate as a free indefinite trial version. Moreover, RealWorks is able to read the ASCII .xyz file format, as well as the industry standard .las format and its compressed cousin .laz. The choice of the .xyz format allows for a more direct analysis using the *x*, *y*, and *z*

coordinates of each point to be displayed. In addition, RealWorks has incorporated image processing, as well as filtering tools to remove points of inconsistency.

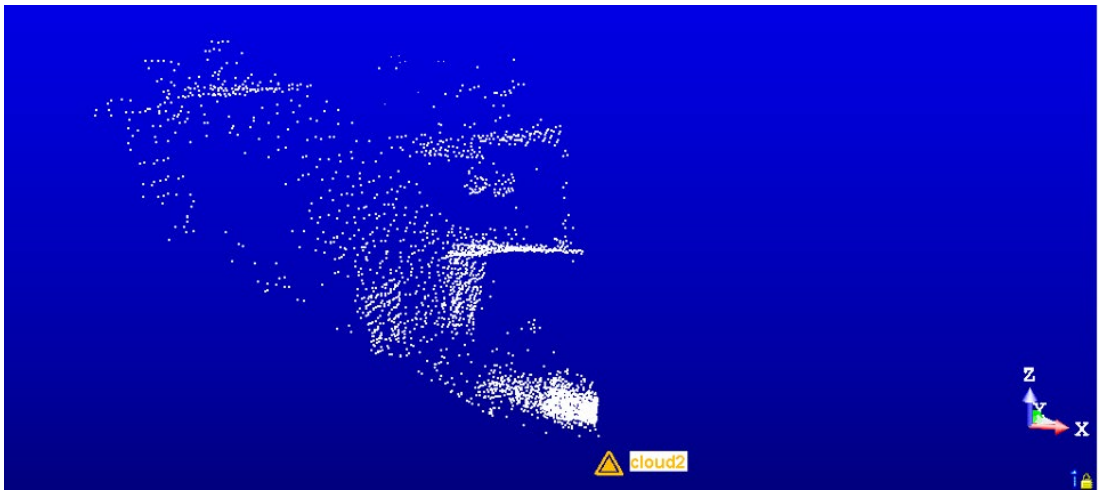

Figure 2.13 Latest point cloud generated of a room in RealWorks using the system developed by the capstone design students.

To date, the students have been able to capture 3-D data using their system as illustrated in figure 2.13. Obviously, while significant improvement is required in order to provide a more detailed understanding of the surroundings, the capstone design students have made significant progress in order to achieve their goals.

2.4 Conclusions

In general, the construction of an inexpensive LIDAR-based range finding system has benefits outside of just bicyclist safety. It has the ability to provide data useful for HPMS reports that can improve the overall transportation safety in Region 7. This chapter describes the on-going efforts in the construction of a mobile LIDAR system that can capture such data. To date, the hardware for this system has been determined but there are current issues involving

communication within the software. Subsequent fixing of these issues and comparison of the data with that of a commercial LIDAR system will highlight its enhanced range finding capabilities. Moreover, combining the efforts with that of an undergraduate capstone design team will enable the generation of detailed three-dimensional views of the surroundings. Overall, this effort is allowing students to learn about LIDAR and its applications with respect to transportation safety and development. This promotes a positive step towards improving all types of transportation safety and collision avoidance.

References

Adafruit. 2017a. "Adafruit Feather 32u4 Adalogger ID: 2795." https://www.adafruit.com/product/2795.

Adafruit. 2017b. "Adafruit Ultimate GPS FeatherWing ID: 3133." https://www.adafruit.com/product/3133.

Adafruit. 2017c. "DC Motor + Stepper FeatherWing Add-on for All Feather Boards ID: 2927." https://www.adafruit.com/product/2927.

Adafruit. 2017d. "Feather - A Complete Line of Development Boards from Adafruit that are Both Standalone and Stackable." https://www.adafruit.com/feather.

Arduino. 2018. "Wire Library." https://www.arduino.cc/en/Reference/Wire.

Basgall, Paul L., Fred A. Kruse, and Richard C. Olsen. 2014. "Comparison of lidar and stereo photogrammetric point clouds for change detection." SPIE Defense + Security.

Beraldin, J.-Angelo, François Blais, and Uwe Lohr. 2010. Three Dimensional Laser Scanning Technology. In *Airborne and Terrestrial Laser Scanning*, edited by G. Vosselman and H.-G. Mass. Scotland, UK: Whittles Publishers.

Bureau of Transportation Statistics. 2016. Chapter 6: Transportation Safety. In *Transportation Statistics Annual Report*. Washington, D.C. : U. S. Department of Transportation.

Castellanos, Sara. 2014. "Northeastern Students Develop 'Smart Bike' Tech to Curb Cycling Deaths." https://www.bizjournals.com/boston/blog/startups/2014/01/northeastern-students-develop-smart.html.

DataPower Technology Limited. 2015. "Product Specifications: Polymer Li-ion Rechargeable Battery." https://cdn.sparkfun.com/datasheets/Prototyping/spe-00-502535-400mah-en-1.0ver.pdf.

DiGioia, Jonathan, Kari Edison Watkins, Yanzhi Xu, Michael Rodgers, and Randall Guensler. 2017. "Safety impacts of bicycle infrastructure: A critical review." *Journal of Safety Research* 61:105-119. doi: http://dx.doi.org/10.1016/j.jsr.2017.02.015.

Du, Wei, Jie Yang, Brent Powis, Xiaoying Zheng, Joan Ozanne-Smith, Lynne Bilston, and Ming Wu. 2013. "Understanding on-road practices of electric bike riders: An observational study in a developed city of China." *Accident Analysis & Prevention* 59:319-326. doi: http://dx.doi.org/10.1016/j.aap.2013.06.011.

Fergus, R., P. Perona, and A. Zisserman. 2001. "Object Class Recognition by Unsupervised Scale-Invariant Learning." 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA.

Finnoff, Jonathan T., Edward R. Laskowski, Kathryn L. Altman, and Nancy N. Diehl. 2001. "Barriers to Bicycle Helmet Use." *Pediatrics* 108 (1):e4-e4. doi: 10.1542/peds.108.1.e4.

Flusche, Darren. 2009. The Economic Benefits of Bicycle Infrastructure Investments. Washington, DC: League of American Bicyclists.

Garmin. 2017. "Varia™ Rearview Radar." https://buy.garmin.com/en-US/US/p/518151.

Garmin International Inc. 2017. Lidar Lite v3 Operation Manual and Technical Specifications. Internet.

Glennie, Craig. 2009. "Kinematic Terrestrial Light-Detection and Ranging System for Scanning." *Transportation Research Record: Journal of the Transportation Research Board* 2105:135-141. doi: 10.3141/2105-17.

Götschi, Thomas, Jan Garrard, and Billie Giles-Corti. 2016. "Cycling as a Part of Daily Life: A Review of Health Perspectives." *Transport Reviews* 36 (1):45-71. doi: 10.1080/01441647.2015.1057877.

Hunt, J. D., and J. E. Abraham. 2007. "Influences on bicycle use." *Transportation* 34 (4):453-470. doi: 10.1007/s11116-006-9109-1.

Jeon, Woongsun, and Rajesh Rajamani. 2016. "Active Sensing on a Bicycle for Accurate Tracking of Rear Vehicle Maneuvers." (50701):V002T31A004. doi: 10.1115/dscc2016-9772.

Lichti, Derek, and Jan Skaloud. 2010. Registration and Calibration. In *Airborne and Terrestrial Laser Scanning*, edited by G. Vosselman and H.-G. Mass. Scotland, UK: Whittles Publishers.

Macmillan, A, J Connor, K Witten, R Kearns, D Rees, and A Woodward. 2014. "The Societal Costs and Benefits of Commuter Bicycling: Simulating the Effects of Specific Policies Using System Dynamics Modeling." *Environ Health Perspect* 122:335-344.

National Highway Traffic Safety Administration. 2016. Quick Facts 2015.

National Safe Routes to School Task Force. 2008. Safe Routes to School: A Transportation Legacy. Washington, D.C.: U.S. Department of Transportation.

NHTSA's National Center for Statistics and Analysis. 2017. Traffic Safety Facts: Bicyclists and Other Cyclists. National Highway Traffic Safety Administration.

NXP Semiconductors. 2014. I^2C-bus Specification and User Manual.

OpenCV team. 2017. "OpenCV Library." http://opencv.org/.

Puente, I., H. González-Jorge, J. Martínez-Sánchez, and P. Arias. 2013. "Review of mobile mapping and surveying technologies." *Measurement* 46 (7):2127-2145. doi: https://doi.org/10.1016/j.measurement.2013.03.006.

Räsänen, Mikko, Ilkka Koivisto, and Heikki Summala. 1999. "Car Driver and Bicyclist Behavior at Bicycle Crossings Under Different Priority Regulations." *Journal of Safety Research* 30 (1):67-77. doi: http://dx.doi.org/10.1016/S0022-4375(98)00062-0.

Raspberry Pi. 2017. "Raspberry Pi Camera Module." https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/RPiCamMod2.pdf.

Rose, Geoffrey. 2012. "E-bikes and urban transportation: emerging issues and unresolved questions." *Transportation* 39 (1):81-96. doi: 10.1007/s11116-011-9328-y.

Schroeder, P., and M. Wlibur. 2012. National Survey of Bicyclist and Pedestrian Attitudes and Behavior, Volume 2: Findings Report. Washington, DC: National Highway Traffic Safety Administration.

Slamtec. 2018. "RPLIDAR A2: Low Cost 360 Degree Laser Range Scanner." https://cdn.sparkfun.com/assets/e/a/f/9/8/LD208_SLAMTEC_rplidar_datasheet_A2M8_v1.0_en.pdf.

Smaldone, Stephen, Chetan Tonde, Vancheswaran K. Ananthanarayanan, Ahmed Elgammal, and Liviu Iftode. 2011. "The cyber-physical bike: a step towards safer green transportation." Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, Phoenix, Arizona.

SparkFun. 2018. "EasyDriver Hook-up Guide." https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide?_ga=2.88730270.488394569.1544122507-599108678.1543950021.

Summala, Heikki, Eero Pasanen, Mikko Räsänen, and Jukka Sievänen. 1996. "Bicycle accidents and drivers' visual search at left and right turns." *Accident Analysis & Prevention* 28 (2):147-153. doi: http://dx.doi.org/10.1016/0001-4575(95)00041-0.

Velodyne LIDAR. 2018. "Ultra Puck™." https://velodynelidar.com/vlp-32c.html.

Walker, Ian. 2005. "Signals are informative but slow down responses when drivers meet bicyclists at road junctions." *Accident Analysis & Prevention* 37 (6):1074-1085. doi: http://dx.doi.org/10.1016/j.aap.2005.06.005.

Wallich, Paul. 2015. An Early-Warning System for Your Bike: Low-cost LIDAR Can Detect Approaching Cars. *Spectrum.IEEE.org*.

Weinert, Jonathan, Chaktan Ma, Xinmiao Yang, and Christopher Cherry. 2007. "Electric Two-Wheelers in China: Effect on Travel Behavior, Mode Shift, and User Safety Perceptions in a Medium-Sized City." *Transportation Research Record: Journal of the Transportation Research Board* 2038:62-68. doi: doi:10.3141/2038-08.

Williams, Keith, Michael Olsen, Gene Roe, and Craig Glennie. 2013. "Synthesis of Transportation Applications of Mobile LIDAR." *Remote Sensing* 5 (9):4652.

Woongsun, Jeon, and R. Rajamani. 2016. "A novel collision avoidance system for bicycles." 2016 American Control Conference (ACC), 6-8 July 2016.

Wu, Changxu, Lin Yao, and Kan Zhang. 2012. "The red-light running behavior of electric bike riders and cyclists at urban intersections in China: An observational study." *Accident Analysis & Prevention* 49:186-192. doi: http://dx.doi.org/10.1016/j.aap.2011.06.001.

Yang, Xiaobao, Mei Huan, Mohamed Abdel-Aty, Yichuan Peng, and Ziyou Gao. 2015. "A hazard-based duration model for analyzing crossing behavior of cyclists and electric bike riders at signalized intersections." *Accident Analysis & Prevention* 74:33-41. doi: http://dx.doi.org/10.1016/j.aap.2014.10.014.

YDLIDAR. 2018. "YDLIDAR X4 Datasheet." https://www.superdroidrobots.com/product_info/TS-082-0X4.pdf.

YellowScan. 2018. "YellowScan Surveyor." https://www.yellowscan-lidar.com/products/yellowscan-surveyor.

Zolotor, Daniel. 2017. "Bicycle-mounted Vehicle Tracking System." https://github.com/dzolotor/LidarCV.