



MID-AMERICA TRANSPORTATION CENTER

Report # MATC-KU: 152-2

Final Report
WBS: 25-121-0005-152-2

UNIVERSITY OF
Nebraska
Lincoln

THE UNIVERSITY
OF IOWA

THE UNIVERSITY OF
KU KANSAS

MISSOURI
S&T

LINCOLN
UNIVERSITY
MISSOURI



UNIVERSITY OF
Nebraska
Omaha

University of Nebraska
Medical Center

KU MEDICAL
CENTER
The University of Kansas

LIDAR, Electric Bikes, and Transportation Safety - Phase II

Christopher Depcik, PhD

Professor

Department of Mechanical Engineering

University of Kansas

KU THE UNIVERSITY OF
KANSAS

2020

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

MATC

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.

**LIDAR, Electric Bikes, and Transportation Safety – Phase II
Final Report**

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

Deven Mittman
Graduate Student
Department of Mechanical Engineering
University of Kansas

A Report on Research Sponsored by

Mid-America Transportation Center
University of Nebraska–Lincoln

January 2020

Technical Report Documentation Page

1. Report No. 25-1121-0005-152-2	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle LIDAR, Electric Bikes, and Transportation Safety – Phase II		5. Report Date January 2020	
		6. Performing Organization Code	
7. Author(s) Christopher Depcik, Ph.D., https://orcid.org/0000-0002-0045-9554		8. Performing Organization Report No. 25-1121-0005-152-2	
9. Performing Organization Name and Address University of Kansas Department of Mechanical Engineering 1530 W 15 th St 3138 Learned Hall Lawrence, KS 66045		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. 69A3551747107	
12. Sponsoring Agency Name and Address Office of the Assistant Secretary for Research and Technology 1200 New Jersey Ave., SE Washington, D.C. 20590 Mid-America Transportation Center Prem S. Paul Research Center at Whittier School 2200 Vine St. Lincoln, NE 68583-0851		13. Type of Report and Period Covered Final Report, January 2019-December 2019	
		14. Sponsoring Agency Code MATC TRB RiP No. 91994-27	
15. Supplementary Notes Conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.			
16. Abstract Mobile light detection and ranging (lidar) technology offers a significant opportunity to increase transportation safety and efficiency. However, most commercial systems are prohibitively expensive for usage with consumer products like bicycles and in widespread implementation throughout our transportation infrastructure. Therefore, cost-effective lidar systems are needed and this effort describes the development of two options targeted for different safety outcomes. The first option involved the generation of a lidar system that can create three-dimensional point clouds with upwards of 700,000 data points as a cost of less than \$300. Initial results highlight its potential in monitoring pavement quality as an example of its capability in providing data for transportation-related reports. The second option built on prior electric bike (e-bike) lidar testing efforts and created a similarly cost effective two-dimensional lidar system that was able to capture the interaction of an e-bike with surrounding motor vehicles. Overall, both options require further refinement before extensive deployment can take place. Nonetheless, this work demonstrates that low-cost lidar systems are a prospective route for enhancing safety within the transportation environment.			
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 71	22. Price

Table of Contents

Disclaimer	viii
Abstract	ix
Chapter 1 Expansion of Mobile LIDAR Data Collection System	1
1.1 Abstract	1
1.2 Background	2
1.3 Hardware and Software	4
1.3.1 Configuration I: Servo Motors	5
1.3.2 Configuration II: Stepper Motors	8
1.3.3 Point Cloud Software	10
1.4 Results and Discussion	11
1.5 Conclusions	20
Chapter 2 Upgrades to 3-D Mobile lidar Data Collection System	22
2.1 Hardware Upgrades	22
2.2 Software Upgrades	27
2.3 Initial System Performance Tests	28
2.4 Pavement Quality Tests	31
2.5 Future Work	35
2.6 Potential System Expansion	36
Chapter 3 Data Collection from a 2-D lidar System Designed for Bicycles	37
3.1 Third Generation 2-D lidar System Hardware	37
3.2 Third Generation 2-D lidar System Software	45
3.3 Stationary Data Collection	48
3.4 Mobile Data Collection	55
3.5 System Diagnosis	60
References	63
Appendix A	68

List of Figures

Figure 1.1 Configuration I illustrating the LIDAR-Lite v3 rangefinder and the two Towerpro servos on a bent aluminum base	6
Figure 1.2 Wiring diagram for Configuration I with the Mega supplying energy to both the lidar rangefinder and the servos	7
Figure 1.3 (Left) Isometric front view of Solidworks Computer Aided Drafting model of the second configuration housing and (Right) Isometric front view of printed second configuration assembly with stepper motors, motor controller, and rangefinder attached.....	9
Figure 1.4 Wiring diagram for Configuration II illustrating an advanced complexity over Configuration I (figure 1.1) due to the addition of two stepper motors.....	9
Figure 1.5 (Left) First point cloud generated using Configuration I and (Right) the corresponding picture location.....	12
Figure 1.6 (Top) Photo of control room for engine test cell on campus and (Bottom) corresponding top view of the point cloud generated of this room in MATLAB	13
Figure 1.7 Front view of point cloud using Configuration II with the server box and window (see figure 1.6 top) now distinguishable.....	14
Figure 1.8 (Top) Front view reference photo for an auditorium classroom on campus with one chair placed on top of the desk and (Bottom) the subsequent point cloud generated using Configuration II	15
Figure 1.9 (Top) Point cloud with axes and (Bottom) without axes of the auditorium classroom in figure 1.8 after implementing code upgrades to Configuration II	17
Figure 1.10 (Top) Reference image for the multi-cylinder engine test cell on campus and (Bottom) the corresponding point cloud.....	18
Figure 1.11 (Top) Reference image of the Formula SAE car and (Bottom) the related point cloud	19
Figure 2.1 Circuit diagram of the 3-D lidar system.....	23
Figure 2.2 Adafruit data logger shield (Earl 2013).....	24
Figure 2.3 Arduino Mega Proto Shield Rev3, the second stackable shield used in the 3-D lidar system (Arduino 2019b)	25
Figure 2.4 The second version of the 3-D lidar system as assembled	27
Figure 2.5 Corner of room scanned for initial 3-D lidar testing	29
Figure 2.6 Point cloud model of room corner in figure 2.5	29
Figure 2.7 Models of empty beaker, beaker filled with clean water, and beaker filled with dirty water (from left to right)	30
Figure 2.8 Scanned pothole with a tape measure providing one foot as a reference.....	32
Figure 2.9 Modeled pothole from figure 2.8.....	33
Figure 2.10 The second scanned pothole.....	34
Figure 2.11 The lidar data from the second scanned pothole of figure 2.10	34
Figure 3.1 Terabee 60 m Evo lidar distance sensor (TeraBee 2017).....	38
Figure 3.2 Arduino Mega 2560 microcontroller (Arduino 2019a).....	38
Figure 3.3 Third-generation 2-D lidar system circuit diagram.....	39
Figure 3.4 Bipolar stepper motor used in the 2-D lidar system (Trinamic Motion Control GmbH & Co. KG 2019).....	40
Figure 3.5 Adafruit micro SD card breakout board (Adafruit 2019).....	41

Figure 3.6 Main housing box model for 2-D lidar system (left) and removable front wall (right)	42
Figure 3.7 3-D CAD models of block stand for motor (left) and lidar rangefinder mount (right)	43
Figure 3.8 Assembled 2-D lidar system closed (top left) and wall removed (top right)	44
Figure 3.9 Empty sections of Arduino C++ code	46
Figure 3.10 Blind spot LEDs mounted on e-bike handlebars to identify obstacles in left, center, and right lanes.	47
Figure 3.11 Image of initial 2-D lidar system test area	49
Figure 3.12 Two sweeps of the 2-D lidar system during initial stationary testing	50
Figure 3.13 Only lidar data registered during stationary test is a false positive of the road divider: visual (left) and data model (right)	51
Figure 3.14 Successful stationary testing with moving vehicles in 30 mph speed limit (left) and lit right lane LED with resulting model of snapshot (right)	52
Figure 3.15 Visual of car (left) registered at a maximum speed of 15 mph (right)	55
Figure 3.16 Completed 2-D lidar system mounted on the back of the electric bicycle	56
Figure 3.17 Typical visual behind e-bike (left) and sweep data (right) from first mobile test with downward lidar angle	57
Figure 3.18 Example of successful data collection and blind spot monitoring of stationary vehicles during a mobile test: visual camera (left) and lidar modeling (right)	57
Figure 3.19 Example of successful data collection and blind spot monitoring of moving vehicle during a mobile test: visual camera (left) and lidar modeling (right)	58
Figure 3.20 Skewed data caused by rapid turning of e-bike during data collection: visual camera (left) and lidar modeling (right)	59
Figure 3.21 Leaning while turning the e-bike causes a false-positive result: visual camera (left) and lidar modeling (right)	60

List of Tables

Table 3.1 Lidar system timing conditions with a 100° sweep angle and optimized software running time of 0.021 s	53
--	----

List of Abbreviations

American Standard Code for Information Interchange (ASCII)
Computer-Aided Design (CAD)
Double-Pole, Single-Throw (DPST)
Electric Bike (e-bike)
Ground (GND)
Highway Performance Monitoring System (HPMS)
In-Circuit Serial Programming (ICSP)
Input/Output (I/O)
Inter-Integrated Circuit (I²C)
Infrared (IR)
Light Emitting Diode (LED)
Light Detecting and Ranging (LiDAR)
Light Imaging Detection and Ranging (lidar)
Light Imaging Detection and Ranging (LIDAR)
Matrix Laboratory (MATLAB)
Megabytes (MB)
Mid-America Transportation Center (MATC)
Miles per Hour (mph)
Nebraska Transportation Center (NTC)
Printed Circuit Board (PCB)
Revolutions per Minute (rpm)
Secure Digital (SD)
Serial Clock Line (SCL)
Serial Data Line (SDA)
Single-pole, Single-Throw (SPST)
Three-Dimensional (3-D)
Two-Dimensional (2-D)
Universal Asynchronous Receiver/Transmitter (UARL)
Universal Serial Bus (USB)
Volts Direct Current (VDC)

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

Mobile light detection and ranging (lidar) technology offers a significant opportunity to increase transportation safety and efficiency. However, most commercial systems are prohibitively expensive for usage with consumer products like bicycles and in widespread implementation throughout our transportation infrastructure. Therefore, cost-effective lidar systems are needed and this effort describes the development of two options targeted for different safety outcomes. The first option involved the generation of a lidar system that can create three-dimensional point clouds with upwards of 700,000 data points as a cost of less than \$300. Initial results highlight its potential in monitoring pavement quality as an example of its capability in providing data for transportation-related reports. The second option built on prior electric bike (e-bike) lidar testing efforts and created a similarly cost effective two-dimensional lidar system that was able to capture the interaction of an e-bike with surrounding motor vehicles. Overall, both options require further refinement before extensive deployment can take place. Nonetheless, this work demonstrates that low-cost lidar systems are a prospective route for enhancing safety within the transportation environment.

Chapter 1 Expansion of Mobile LIDAR Data Collection System

Note: This chapter is published as Wiklund, T., Heim, M., Halberstadt, J., Duncan, M., Mittman, D., DeAgostino, T., and Depcik, C., “Design and Development of a Cost-Effective LIDAR System for Transportation,” Proceedings of the ASME 2019 International Mechanical Engineering Congress and Exposition, November 11-14, Salt Lake City, UT, USA, 2019, doi: 10.1115/IMECE2019-11279.

1.1 Abstract

Light Imaging Detection and Ranging (LIDAR) cameras and Light Detecting and Ranging (LiDAR) rangefinders were initially implemented in the 1960s as a higher-resolution and increased capability alternative to radar. Since then, LIDAR and LiDAR (hereto called lidar) have expanded into applications in aerial geographical surveying and collision-detection systems for autonomous vehicles. Current commercial systems are relatively expensive and potentially oversized for non-commercial applications. Consequently, this deters their use on consumer products like bicycles, where lidar systems can enable safety advancements that are necessary to counter the rising numbers of hazards affecting riders. In addition, widespread usage of inexpensive lidar systems can facilitate a more complete picture of our transportation infrastructure by delivering information (e.g., pavement quality) suited for U.S. Department of Transportation Highway Performance Monitoring System (HPMS) reports. This will aid in the creation of a safer infrastructure by highlighting critical areas in need of improvement and repair.

As a result, this effort outlines the development of a compact and cost-effective lidar system. The constructed system includes the ability to generate a static image by collecting several hundred thousand distance signals measured by a lidar rangefinder. Since the rangefinder has no self-contained rotation or translation systems, an Arduino Mega 2560 v3 microcontroller

operates a pair of stepper motors that adjusts its azimuthal angle and pitch. Coalescing these signals into an ASCII text file for viewing in MATLAB results in a reasonably accurate picture of the surroundings. While the current system takes 1-2 hours to complete a full sweep, it has the potential to provide sufficient accuracy for HPMS reports at a moderate expenditure: the entire system costs less than \$300. Finally, upgrading to a more powerful microprocessor, implementing slip rings for enhanced electrical connectivity, and refining the code by including interpolation between points will enable faster point cloud generation while still maintaining an inexpensive device.

1.2 Background

Methods of transportation can vary for individuals depending on weather, destination, purpose, or other factors. Some might use public transit, personal vehicles, bicycles, or walking as their preferred mode of transportation. Unfortunately, this wide variance of options with disparate speeds results in a complex environment with vehicular collisions accounting for a quarter (24.9%) of all accidental deaths in the United States in 2016 (Xu et al. 2018). Understandably, safety is a major concern for most commuters and while the number of accidents has decreased in the past, there has been a recent rise since 2014 (National Center for Statistics and Analysis 2018).

A primary safety concern is the existence of blind spots. Typically, rear and side-view mirrors help drivers monitor the area behind them. While additional mirrors are suggested to completely eliminate blind spots, watching multiple mirrors will slow drivers' reaction time (Mole and Wilkie 2017). Therefore, it is preferable to monitor the area surrounding the vehicle or bicycle via another system. A detection system to alert drivers, visually and/or audibly, would help improve reaction time while potentially providing more consistent benefit than mirrors

alone. A secondary safety issue includes the condition of the road. Inadequate road infrastructure is listed as a frequent cause of single-vehicular mishaps, especially rollover accidents (Goniewicz et al. 2016, Anarkooli, Hosseinpour, and Kardar 2017). With the United States infrastructure currently in poor condition (American Society of Civil Engineers 2019), having a detection system monitor road conditions in addition to blind spots would result in a significant opportunity to improve safety.

Lidar is one remote sensing method that can facilitate an effective monitoring of both safety concerns. Briefly, lidar works similar to radar systems by using near visible light waves instead of radio waves and can map the surrounding environment in three-dimensions (3-D) (Puente et al. 2013). Current applications for drone-mounted aerial lidar systems include forest mapping to track growth, modeling forest fire behavior, classifying land and environmental types, and charting various other environments for a variety of purposes (Kelly and Di Tommaso 2015, Garcia-Gutierrez, Goncalves-Seco, and Riquelme-Santos 2011, Yang et al. 2013, Chiang et al. 2017). Additionally, ground-based mobile lidar systems can recognize various road types and identify defects in their respective surfaces while monitoring the environment surrounding roads for potential dangers (Kromer et al. 2015). In areas where valleys and other steep slopes are adjacent to roads, rail lines, and canals, landslides are detrimental to transportation and infrastructure. Here, lidar systems can be used to inspect surface material and identify changes and patterns that might lead to landslides (Neupane and Gharaibeh 2019).

While these applications illustrate lidar's propensity to provide accurate and detailed representations, it is often costly to collect these data while respectively difficult to analyze the point cloud files that result from the collocation of this information (Kelly and Di Tommaso 2015). Commercial lidar systems are highly capable; however, their individual cost (\$6k to

\$100k (Lienert and Nellis 2019)) might be excessive for numerous vehicle-mounted systems. For instance, a vehicular system does not have to scan wide areas of land at a time, only the immediate vicinity if there is a targeted goal in mind (e.g., road conditions versus automated driving). Hence, designing an inexpensive and small lidar system to identify vehicle, pedestrian, and bicycle proximity along with road defects could significantly benefit transportation safety while providing for widespread implementation.

As a result, this effort describes the design of a more accessible and inexpensive lidar system while briefly discussing the potential impact it can have for transportation safety. First, two configurations of the hardware employed are presented highlighting a change from servomotors to stepper motors to enhance accuracy. Next, a straightforward methodology in data collection is indicated to generate the point cloud information via text files. Finally, the implementation of both configurations is presented stressing the lessons learned while culminating in the development of an instrument that should cost less than \$300 and be capable of producing relatively accurate 3-D point clouds.

1.3 Hardware and Software

Since lidar uses infrared light, its wavelength (e.g., 905 nm) is reduced significantly in comparison to comparable radar systems (e.g., 50 cm). This provides it the capability to generate a high-resolution image (aka high point density point cloud). Lidar rangefinders determine the distance of objects by emitting short pulses of light and recording the time it takes for this light to return to the detector. Object distance is determined by multiplying the speed of light by half the time it took the laser pulse to return. Subsequently, combining this distance with known horizontal and vertical angles of the rangefinder determines the x , y , and z positions of individual points. A point cloud is generated from this 3-D map using an appropriate software program.

The fabrication of a complete lidar system includes integrating a lidar rangefinder with some mechanism of sweeping this component in three-dimensions. Furthermore, a microprocessor is required to process and store these data. Previous experience in creating this lidar system for the back of an electric bicycle demonstrated limited success and generated only two-dimensional information (Blankenau et al. 2018). Building on this prior knowledge, this effort expanded the system's capabilities into 3-D via two successive hardware configurations. In both configurations, the Garmin LIDAR-Lite v3 module is employed since it has a greater range and accuracy ($40 \text{ m} \pm 10 \text{ cm}$) than other inexpensive alternatives: e.g., Taidacent TOF 10120 ($1.8 \text{ m} \pm 5\%$) and the Benewake TF Mini ($12 \text{ m} \pm 6 \text{ cm}$). The LIDAR-Lite v3 also provides several different configuration settings that can be explored to enhance resolution accuracy.

1.3.1 Configuration I: Servo Motors

Learning from the preceding effort, an Arduino Mega 2560 Rev3 (16 MHz: henceforth Mega) was used as the microprocessor instead of an Adafruit Feather System or a Raspberry Pi 3B+. The open-source Arduino Integrated Development Environment and modified C++ programming language is well documented and respectively easy to learn for undergraduate students (the primary authors of this paper). Furthermore, Garmin officially supports the LIDAR-Lite v3 rangefinder on the Arduino platform and a library is supplied on Github (Garmin Ltd. 2018). In contrast, while the Adafruit system worked previously, it did not provide sufficient processing speed and tended to be unreliable. Moreover, while the greater processing speed of the Raspberry Pi 3B+ (1.4 GHz, 64-bit quad-core) is advantageous for mobile systems, the primary issue of the aforementioned efforts suggested the focus be on point cloud accuracy over computational speed. This goal, when combined with a greater difficulty in learning the native Raspbian operating system and Python programming language (along with the LIDAR-Lite v3

not being officially supported on the Raspberry Pi platform), further solidified the choice of the Mega.



Figure 1.1 Configuration I illustrating the LIDAR-Lite v3 rangefinder and the two Towerpro servos on a bent aluminum base

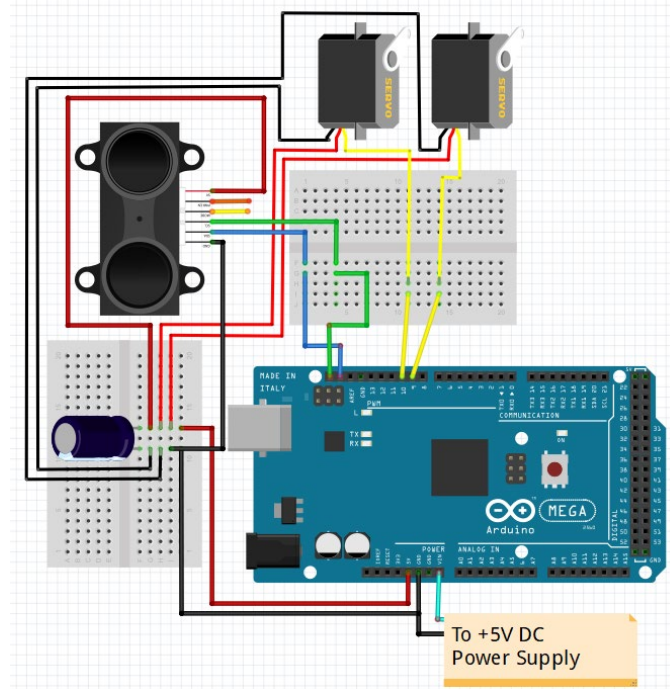


Figure 1.2 Wiring diagram for Configuration I with the Mega supplying energy to both the lidar rangefinder and the servos

In the first configuration shown in figure 1.1, two TowerPro MG996R digital metal gear servomotors were employed to rotate in the x - y and y - z directions, respectively. A laptop computer supplied power for the entire system and a capacitor was used to protect the rangefinder from voltage spikes or current bursts. In figure 1.2, the rangefinder and Mega communicated over the Inter-Integrated Circuit protocol using Serial Data Line and Serial Clock Line Mega pins, colored blue and green in the figure, respectively. Furthermore, signals to the servos were sent using the Mega's Pulse Width Modulation pins.

Programming of the servomotors included adding a servo library to the Arduino code (Arduino 2019e). Using the angles of the motors retrieved from this library during operation, trigonometry was employed to calculate the x , y , and z -coordinates of each distance measured from the rangefinder. Unfortunately, the servos chosen could only rotate 180° in 1° increments.

As a result, this configuration was unable to create a spherical point-cloud and had a relatively large degree per step value.

1.3.2 Configuration II: Stepper Motors

Similar to Configuration I, the Mega was used as the microprocessor for Configuration II. Now, two Kiatronics 28BYJ-48 5 VDC stepper motors, controlled by a Kiatronics ULN2003 motor controller, were employed to move the rangefinder. Each stepper motor had a gear reduction of 1/64 allowing for a rotation of 0.08° per step, facilitating a significant improvement in point cloud resolution (shown later in Section 1.4).

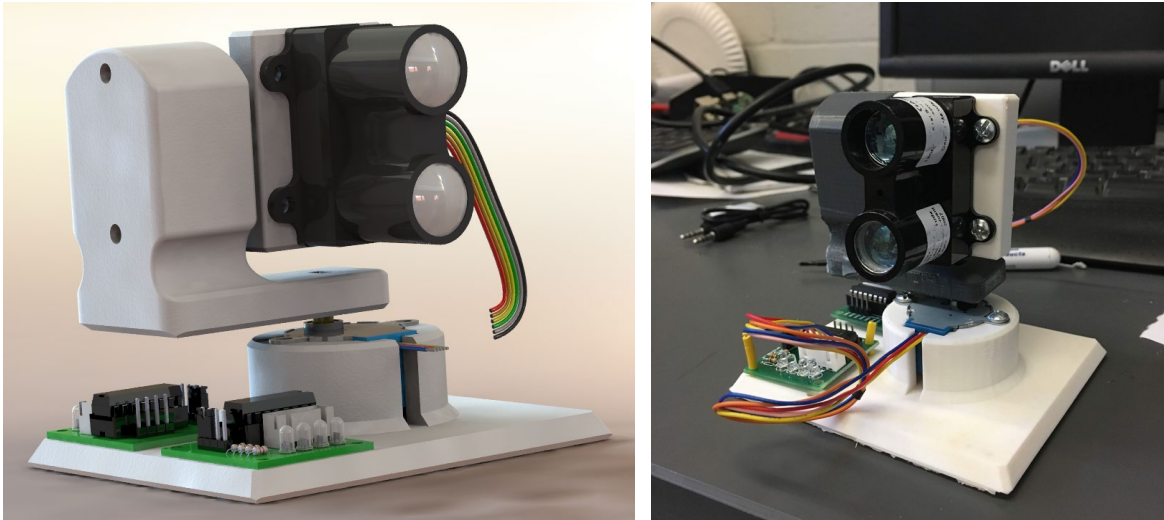


Figure 1.3 (Left) Isometric front view of Solidworks Computer Aided Drafting model of the second configuration housing and (Right) Isometric front view of printed second configuration assembly with stepper motors, motor controller, and rangefinder attached

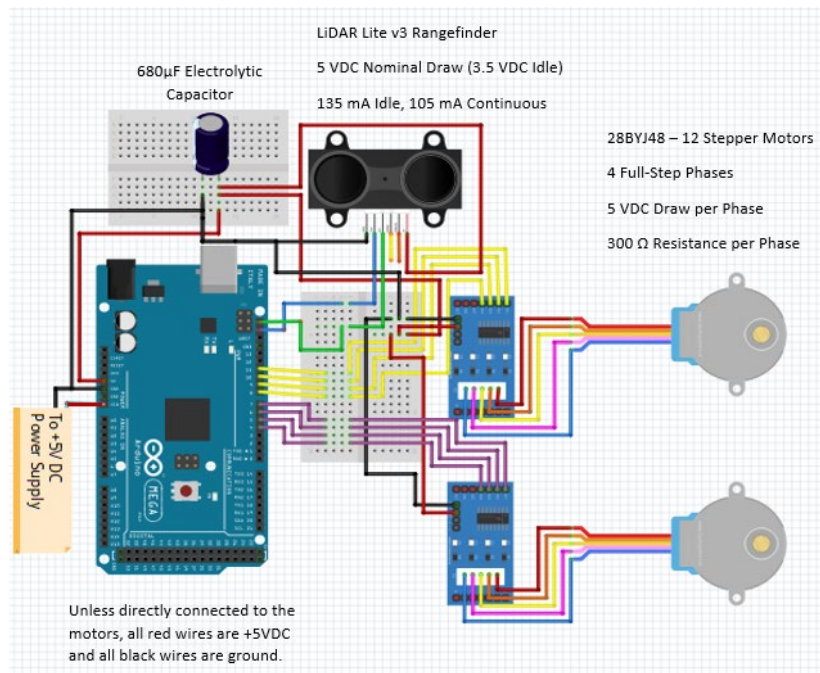


Figure 1.4 Wiring diagram for Configuration II illustrating an advanced complexity over Configuration I (figure 1.1) due to the addition of two stepper motors

During testing of Configuration I, the bent aluminum structure (figure 1.1) flexed during operation resulting in the rangefinder not rotating around a fixed point in space. Moreover, the motor shafts did not line up to the fixture point of the rangefinder resulting in data that did not have a common origin. Instead, the second configuration included a 3-D printed housing, as illustrated in figure 1.3, which provided a solid base, minimized vibration during usage, and created a common origin. This housing was printed from acrylonitrile butadiene styrene using two Stratasys Mojo fused deposition modeling printers and took 9.1 hours to complete while utilizing 5.7 in³ of material.

Like the first configuration in figure 1.2, wiring of the second version in figure 1.4 involved power being supplied by a laptop computer and a capacitor was implemented to protect the rangefinder. Now, the Mega communicated with two motor controllers connected to the stepper motors that operate using four electromagnets. These motors can be rotated at half steps between the magnets enabling an advanced resolution. Unfortunately, the available Arduino stepper motor library did not properly communicate with these motor controllers (Arduino 2019f). Therefore, code was written to directly change the voltages of the electromagnets inside these motors, one magnet at a time.

1.3.3 Point Cloud Software

The data coming from a rangefinder includes the raw distance; hence, the most straightforward format for generating point clouds is through the American Standard Code for Information Interchange (ASCII) .xyz file type that features three columns of x , y , and z -coordinates for the thousands of points in a point cloud. Most commercial software packages that generate point clouds are setup to read the industrial standard .las and .laz lidar data. While initially the Trimble Realworks Viewer 11.0 was used because it can plot both .xyz and .las

formats enabling a transition between the generated raw distance data into the industry format, it was decided to employ MATLAB as an alternative point cloud processing tool.

The LIDAR-Lite v3 rangefinder utilized is not capable of detecting color and is not officially supported to provide signal strength data. Whereas, .las and .laz file types allow for incorporation of color and signal strength. Furthermore, the students involved in this effort are familiar with MATLAB programming through their undergraduate curriculum. As a result, MATLAB code was generated that can parse data arrays from the Arduino system and concatenate this information into x , y , and z -coordinates. When reviewing these data in the following section, it was found that some datasets had points that were not near the subject of interest; i.e., random outliers. Code was added to filter this outlying data to ensure presentation of only the area of interest. These data are then plotted using the 3-D scatter plot option in MATLAB with color used as the legend to determine the distance away from the rangefinder. Except for one instance, ASCII .xyz text files were used to generate the point cloud images in the next section.

1.4 Results and Discussion

The first point cloud generated using Configuration I and plotted using the Trimble Realworks Viewer is illustrated in figure 1.5. Overall, these data took four minutes to capture and the servomotors were programmed to rotate 30° horizontally and 45° vertically. While the edge of the monitor on the right is somewhat visible in the point cloud at a slightly different angle, the overall point cloud resolution is poor. It is possible that the monitor screen material interfered with the rangefinder's laser pulses by absorbing or reflecting them away from the rangefinder; hence, it shows up as an empty screen area. In addition, while the monitor on the left is partially visible in the point cloud, the window behind the monitors prevented any further

details from appearing as the laser pulses went through into the next room and did not return to the rangefinder.

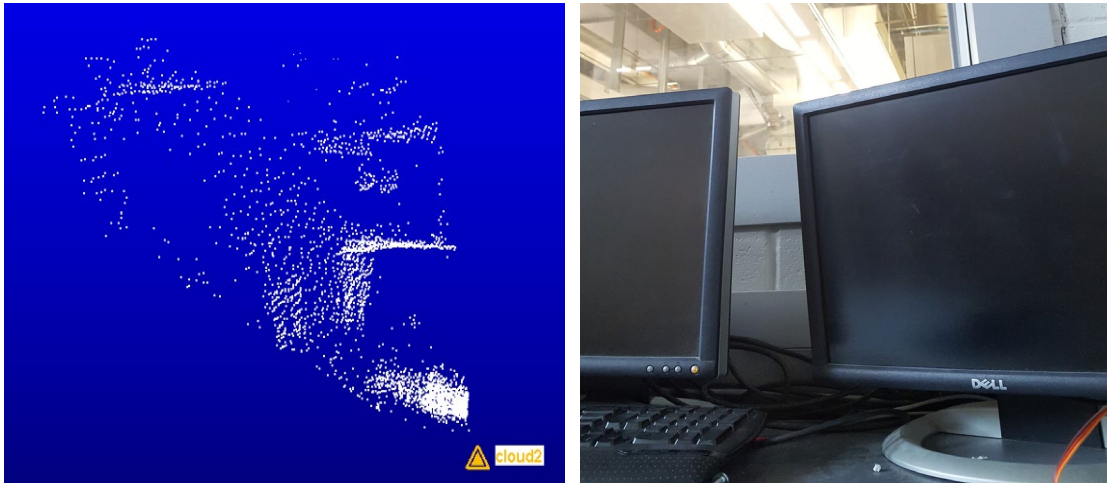


Figure 1.5 (Left) First point cloud generated using Configuration I and (Right) the corresponding picture location

At this point, a second set of data were taken using Configuration I to see if any improvements could be made to the setup or the underlying Arduino code. This time MATLAB was used to generate the point cloud with the corresponding picture and point cloud shown in figure 1.6. It took eight minutes to generate these data and during this process the aluminum mount was seen to wiggle after each horizontal sweep was completed, resulting in the double image seen in this point cloud. The point cloud still has a respectively poor resolution and the system loses accuracy as the distance from the rangefinder increases; i.e., the points get further apart the farther they are away from the rangefinder.

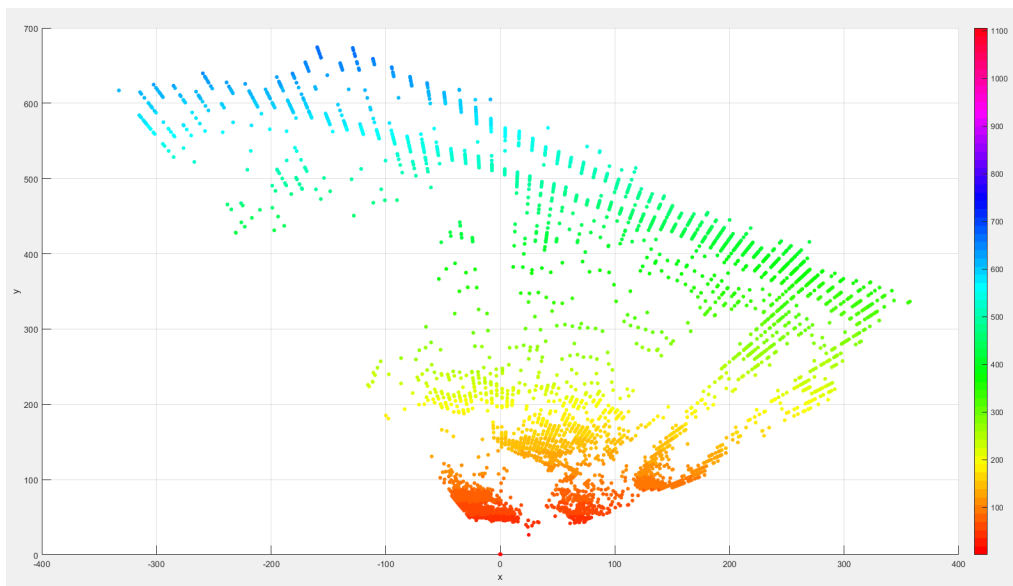
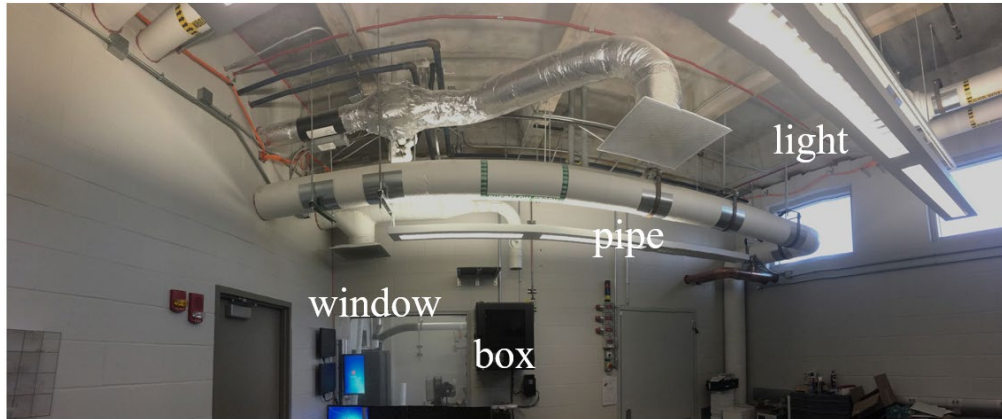


Figure 1.6 (Top) Photo of control room for engine test cell on campus and (Bottom) corresponding top view of the point cloud generated of this room in MATLAB

It was at this point that Configuration II was constructed to increase the number of data points taken from about 3,000 data points in Configuration I to 15,000-20,000. This upgraded system took ten minutes to collect the same picture location as figure 1.6. Figure 1.7 illustrates that the service box on the wall to the right of the window is now more clearly seen jutting out of the wall along with the window itself becoming distinguishable. The walls are now discernable and a large cylindrical pipe near the ceiling is present. The respectively bright rectangular light

can (somewhat) be seen lower in the image and closer to the rangefinder. Of importance, the filtering routine implemented in MATLAB removed data behind the window because it skewed the overall point cloud picture. In the point cloud figures moving forward, the legend color indicates the distance in [cm] from the rangefinder in all three-directions.

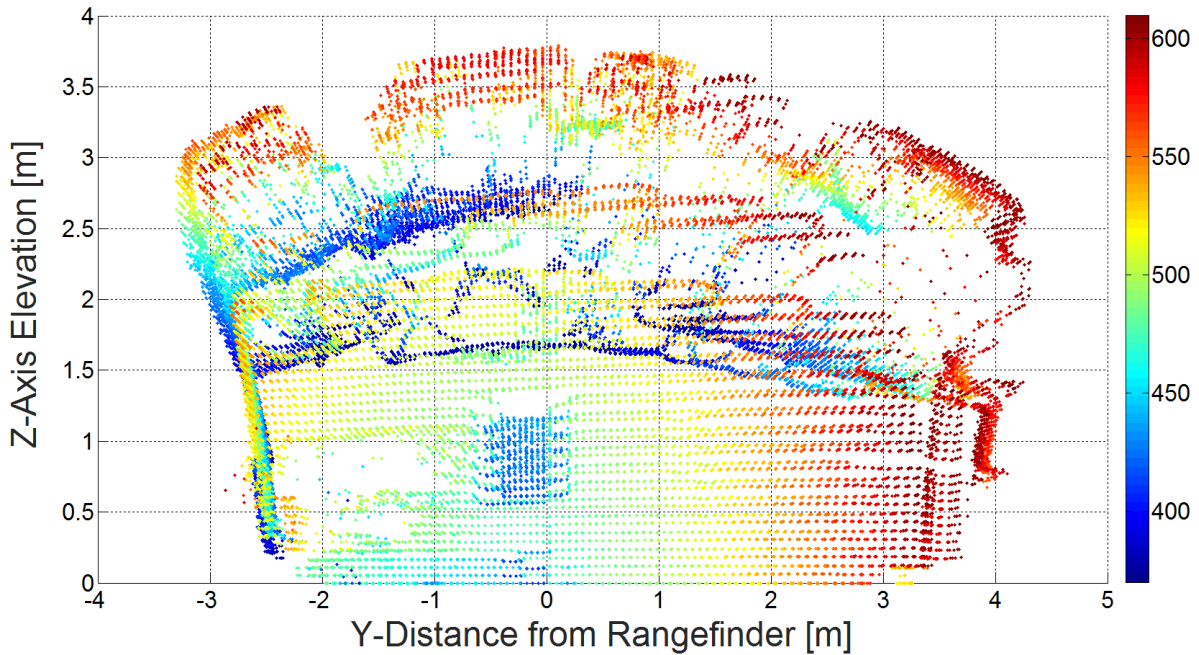


Figure 1.7 Front view of point cloud using Configuration II with the server box and window (see figure 1.6 top) now distinguishable

However, when attempting to capture a classroom on campus with numerous objects (figure 1.8), respectively few distinguishing characteristics are seen. Except for the overall shape of the auditorium and the ceiling, there are not many recognizable features. Upon reviewing the Arduino code, it was found that there was a mistake in the electromagnet voltage specifications that limited the horizontal resolution of the point clouds. Many unique processing steps were counted as the same step that caused the resulting point clouds to have multiple points in one location. Furthermore, this version of the code incremented the vertical motors as a full rotation

around the magnets. This caused a relatively large jump in the angle upwards when it could have been respectively smaller. Both figure 1.7 and figure 1.8 illustrate these issues with numerous points in the horizontal direction missing along with a reduced accuracy (i.e., jumps) in the vertical direction.

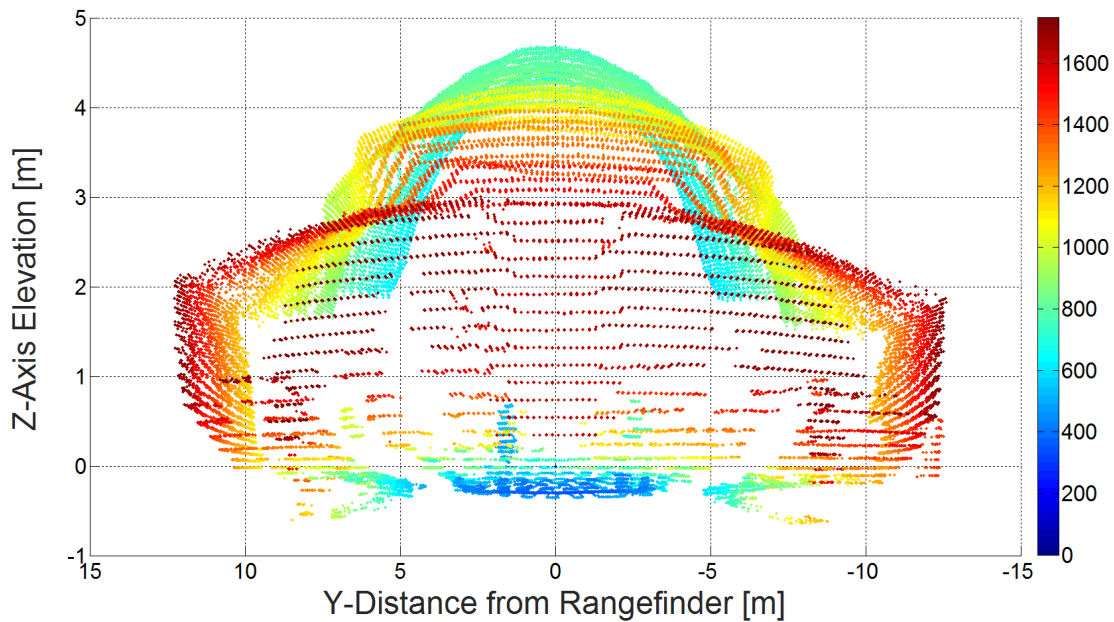


Figure 1.8 (Top) Front view reference photo for an auditorium classroom on campus with one chair placed on top of the desk and (Bottom) the subsequent point cloud generated using Configuration II

A subsequent upgrade to the code fixed the horizontal bug that augmented the resolution in this direction by seven times. Moreover, additional code was written to loop half steps between each electromagnet of the vertical motor. Rather than a full rotation around all four magnets, the motor rotated once between magnets one and two. After another horizontal sweep, the vertical motor then moved to magnet two. After another horizontal sweep, the motor moved between magnets two and three, and so on. This amplified the vertical resolution by seven times; hence, bringing the total resolution growth to forty-nine times the previous code. Unfortunately, this increased resolution created a data collection issue. After 300,000 data points are collected, the serial monitor within the Mega began deleting distance measurements collected from the beginning of a test. Currently, a third-party serial monitoring program (CoolTerm (Meier 2019)) is installed in the laptop that uses the same communication port connected to the Arduino and writes these data directly to a text file. Ideally, direct communication between Arduino and MATLAB would allow MATLAB to read these serial monitor data and plot the point cloud in real time while fixing the data deletion issue.

Figure 1.9 presents the updated Configuration II point cloud for the same location as figure 1.8. This data set took 130 minutes to create, contained over 700,000 points, and generated a text file with a size of 8 MB. Nearly all seats are clearly visible, especially those close to the front. Moreover, the chair placed on top of the desk in the middle of the classroom is seen clearly. On the left side of the auditorium and to the right of the left walkway, one outlet on every desk starting from the front and ending towards the back was lifted. While difficult to see in this figure, after expanding the image to a larger size, these outlets are shown as small bumps in the point cloud.

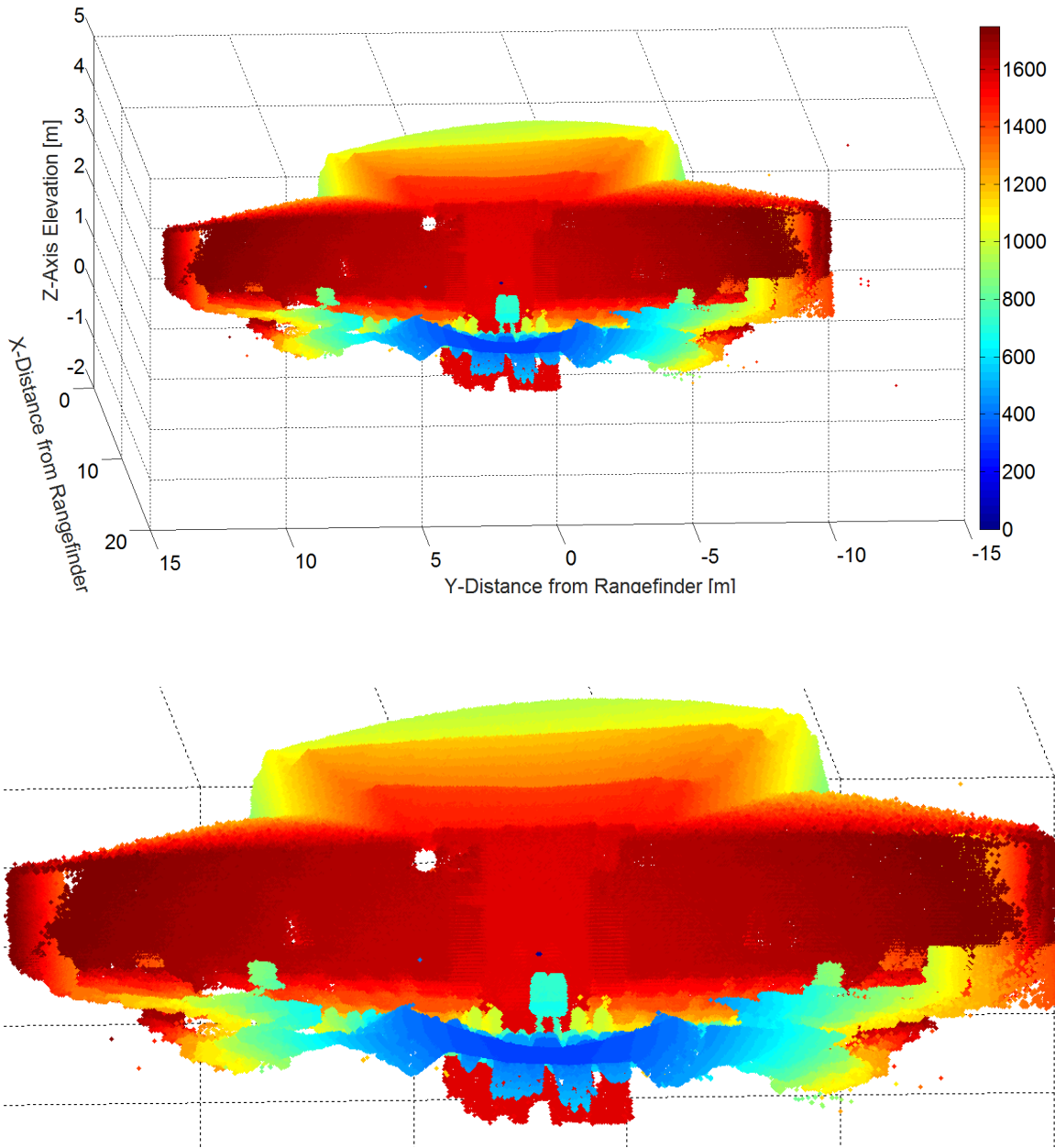


Figure 1.9 (Top) Point cloud with axes and (Bottom) without axes of the auditorium classroom in figure 1.8 after implementing code upgrades to Configuration II

This success led to another point cloud being taken of a multi-cylinder engine test cell on campus in figure 1.10. This dataset took 85 minutes to create and data beyond a certain range were removed to better utilize color grading within MATLAB for the subjects of interest. After

deletion, there remains about 200,000 points with the dynamometer on the right hand side of the engine clearly seen. In addition, the curved pipe starting at the floor and ending at the engine is noticeable. It is important to note that only the default settings on the rangefinder were used; hence, configuring it to its short-range option might increase the detail in scenarios, such as figure 1.10, where the objects are closer to the rangefinder.

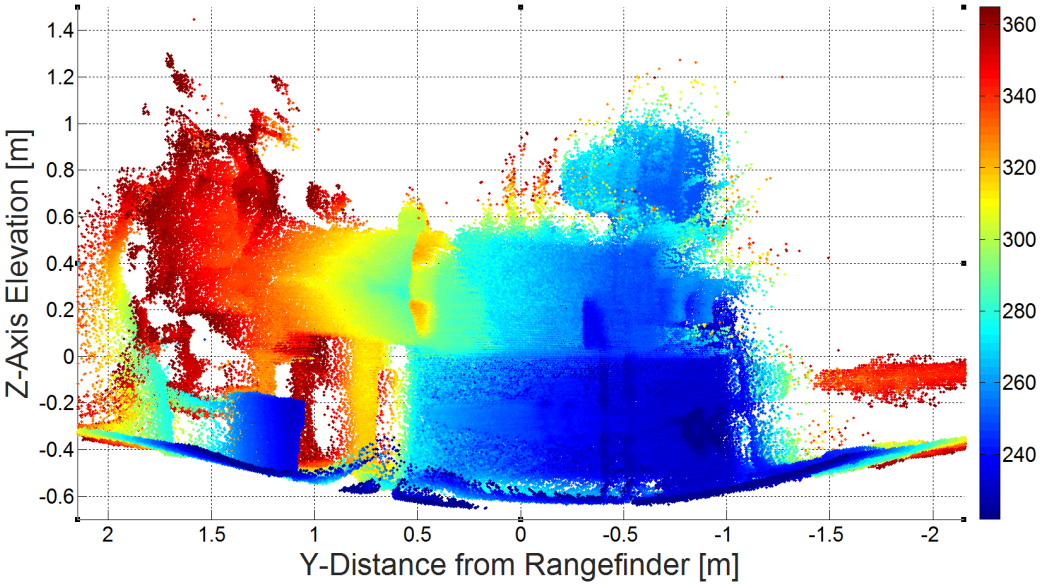
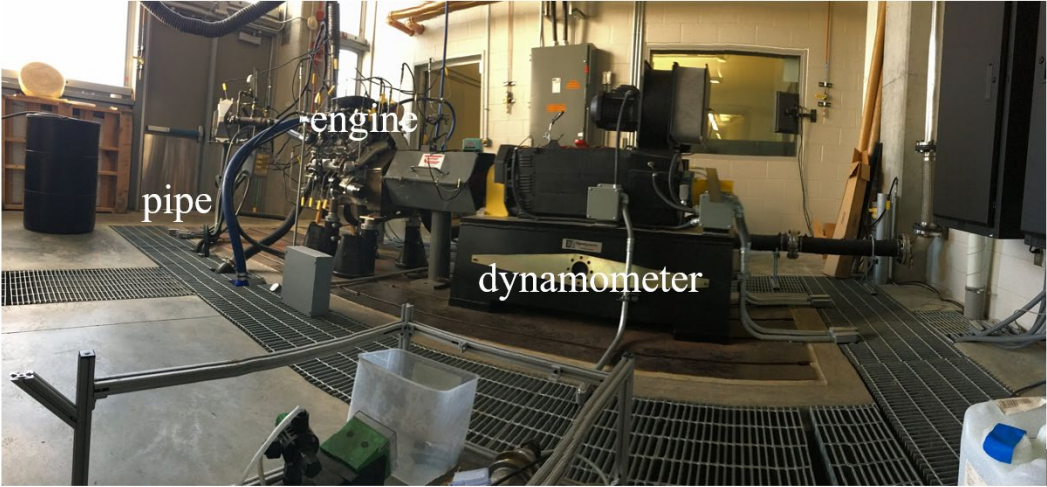


Figure 1.10 (Top) Reference image for the multi-cylinder engine test cell on campus and (Bottom) the corresponding point cloud

To highlight how data post-processing can improve point cloud detection of the subject of interest, figure 1.11 presents a picture and point cloud of a Formula SAE car. By strategically removing data points beyond a certain distance, the picture of the vehicle becomes rather recognizable. This demonstrates that successful lidar usage requires the fabrication of a capable hardware system coupled to efficient software routines.

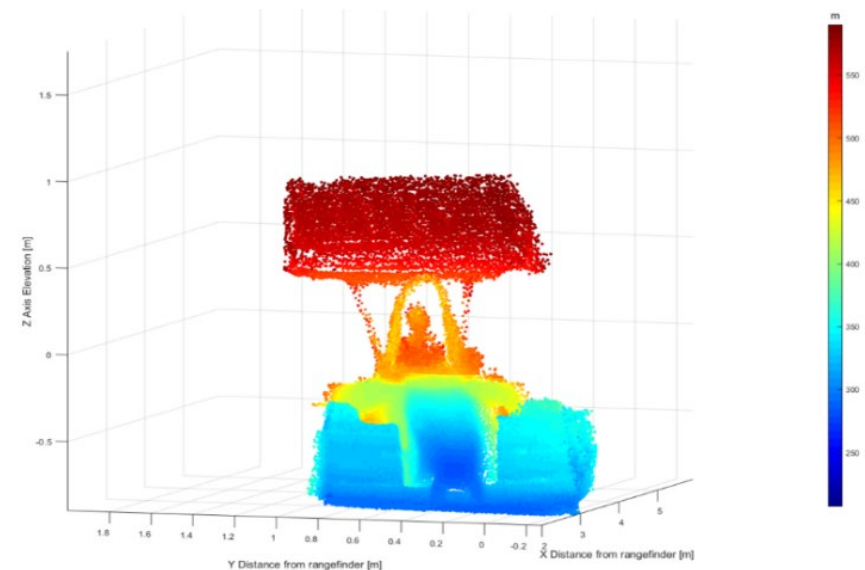


Figure 1.11 (Top) Reference image of the Formula SAE car and (Bottom) the related point cloud

Overall, this effort illustrates that higher resolution point clouds take significantly longer to create. Placing a system with this level of detail onto mobile platforms (e.g., electric bikes) where immediate knowledge of threats is needed appears unfeasible. Instead, like the previous effort, use of a rangefinder in conjunction with a camera can sweep an area significantly faster; hence, detecting vehicles more quickly along with the distance of that vehicle to alert riders of potential danger. Other possibilities include integrating this rangefinder with more extensive software algorithms that can track objects of interest (Jeon and Rajamani 2019). However, this system appears suitable for delivering information for 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). Since the total system cost is less than \$300 (not considering the 3-D printed mount estimated at less than \$30), it is possible to facilitate widespread implementation of lidar across the entire transportation infrastructure to enhance the information gathered. Finally, moving to a Raspberry Pi 3B+ microprocessor and implementing slip rings in the setup can help create a stand-alone system that is robust, fast, and, in combination with code upgrades that include interpolation between points, can generate high quality point clouds at a minimum expenditure.

1.5 Conclusions

The extensive application of lidar systems throughout the transportation infrastructure can facilitate a safer environment for travelers. These systems can enable the public to be aware of imminent threats while helping highlight critical areas in need of improvement and repair. However, current commercial lidar systems are relatively expensive, subsequently reducing their potential widespread feasibility. This effort endeavored to minimize expenditures when

attempting to generate a lidar system of similar accuracy to commercial options. This was accomplished by utilizing a Garmin LIDAR-Lite v3 as the rangefinder and an Arduino Mega 2560 v3 microcontroller in combination with two stepper motors. Overall, it was possible to generate relatively accurate point clouds in MATLAB from ASCII text files with upwards of 700,000 data points. With a cost less than \$300 (not including a 3-D printed mounting), this increases the possibility of wide-ranging implementation. Currently, this system is not suitable for mobile applications as data collection time took around 1-2 hours. Nevertheless, the system appears suitable for delivering information for public transport reports. Finally, potential upgrades to the system (e.g., microprocessor and slip rings) can further improve speed, robustness, and accuracy while not significantly growing its cost.

Chapter 2 Upgrades to 3-D Mobile lidar Data Collection System

The previous chapter describes the first generation of a 3-D mobile lidar collection system targeted for transportation-related activities. This chapter details upgrades to that system accomplished since the publication of the work.

2.1 Hardware Upgrades

A second version of the 3-D lidar system improves on some of the previous design flaws, decreases its size and weight, and enhances mobility. This second version keeps the system circuitry and electrical components as similar as possible while allowing these improvements. The Garmin lidar rangefinder, Arduino microcontroller, motors, and motor housings are the same as the first version to lessen the potential for electrical problems. However, the motor controller circuit boards, capacitor for the lidar rangefinder, and connections made on the breadboard are soldered directly onto stackable protoboard shields. Additionally, the second 3-D lidar system includes a self-contained battery power supply, data storage, and a power switch as illustrated in figure 2.1.

The major issues with the first system are that it required connections to both a large power supply attached to a wall outlet and a computer to collect data in real-time. Hence, the goal was to make the second 3-D lidar system completely portable along with being easy to transport and use. Therefore, the power supply must be self-contained in the system. As a result, power is now divided between two battery packs at the bottom of the system. The first battery pack has a voltage of 9 VDC and supplies power to the Arduino microcontroller through the power switch connected to the Vin and GND pins (figure 2.1). However, this battery pack does not provide enough power for every component of the lidar system through the microcontroller. In specific, the two stepper motors require more current than the Arduino microcontroller can

supply, which causes a rapid voltage drop and results in the system turning itself off for protection. Therefore, a second battery pack of 6 VDC supplies power directly to the stepper motors through the same power switch as before, without running current through the microcontroller. Here, it is important to note that the stepper motors are still controlled by the Arduino microcontroller; i.e., the motors simply get their power from a separate source.

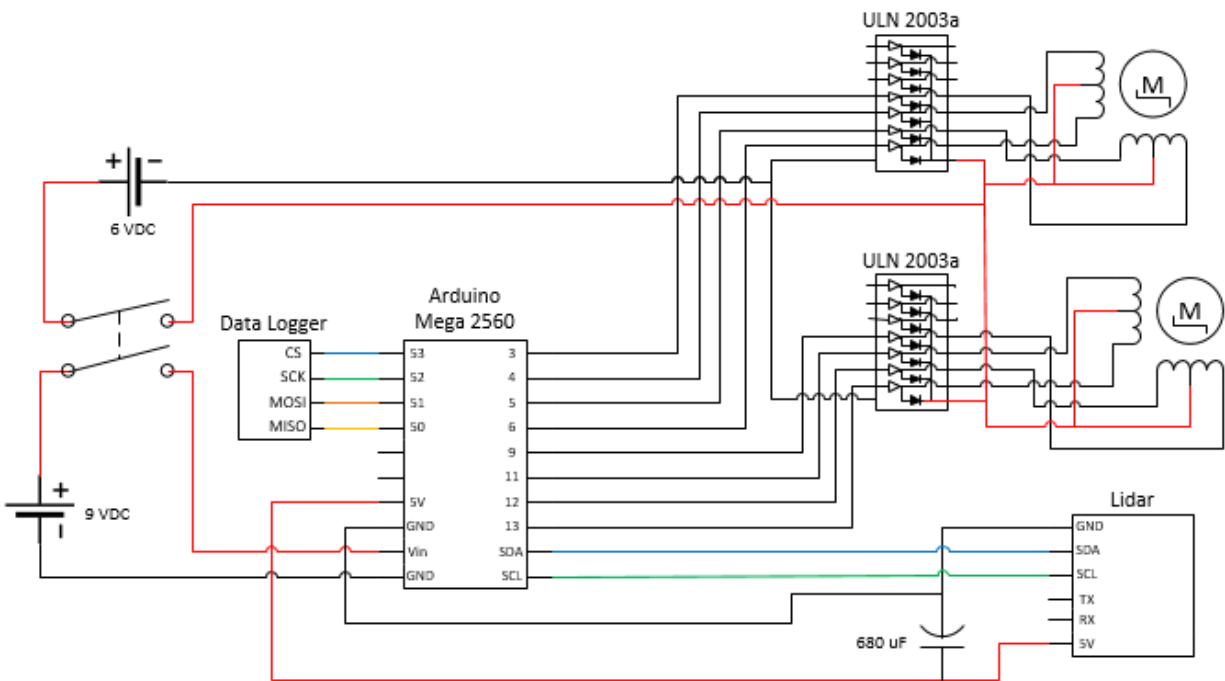


Figure 2.1 Circuit diagram of the 3-D lidar system

A second issue with the initial 3-D lidar system was its inability to save data on board the device. Instead, it collected data through the serial readout on a connected laptop computer and manually saved these data to a .txt file once data collection was finished. The second 3-D lidar system addresses this issue by including a Secure Digital (SD) card. This SD card connection is pre-mounted on a data logging shield that has an area for direct soldering of the circuitry to the

board (Earl 2013). The Adafruit data logging shield (figure 2.2) is the same size as the popular Arduino Uno microcontroller and is smaller than the Mega 2560 microcontroller used in this system. However, the two boards are still stackable. Here, the Mega 2560 simply extends past the end of the data logging shield. This top shield holds the double-pole, single-throw (DPST) power switch, and lidar capacitor. In addition, it connects the data storage lines through the In-Circuit Serial Programming (ICSP) connections to the necessary pins on the Arduino microcontroller. Furthermore, because this shield has no obstructions to tangle wires as they move, the Garmin lidar rangefinder's Inter-Integrated Circuit (I²C) and power lines are connected to the microcontroller through the corresponding pins on this shield.

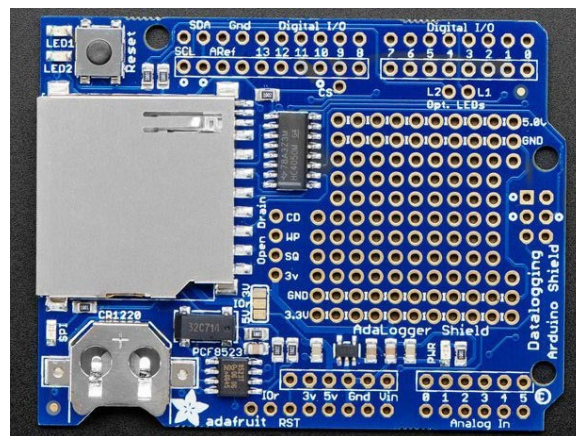


Figure 2.2 Adafruit data logger shield (Earl 2013)

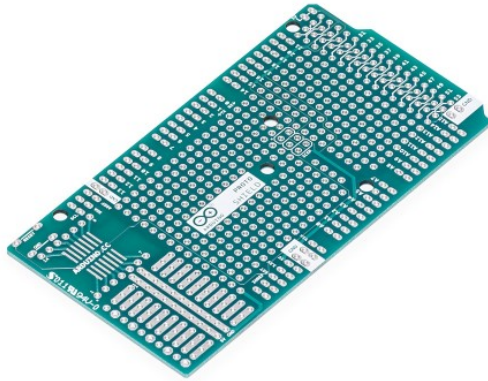


Figure 2.3 Arduino Mega Proto Shield Rev3, the second stackable shield used in the 3-D lidar system (Arduino 2019b)

The stackable shields allow the Arduino Mega 2560 microcontroller to expand the number of possible connections and helps to permanently connect circuits directly to the board; i.e., eliminating the less permanent breadboard connections employed in the prior system. The second shield employed is the Arduino Mega Proto Shield Rev3 (figure 2.3). It is the same size as the Mega 2560 and is a printed circuit board (PCB) (Arduino 2019b). The circuitry needed to power and control the stepper motors and the SD card are soldered onto this shield. To save space, the motor controller boards in the first design were recreated with the same transistors and the stepper motor pin connectors were directly soldered on the proto shield (Geeetech Wiki 2012). Additionally, the ICSP lines from the top data logging shield are connected to the microcontroller through this protoboard shield.

The stepper motors and their housing are unchanged from the first 3-D lidar system (figure 1.3). The horizontal motor turns the vertical motor and its housing, and the vertical motor turns the lidar rangefinder. However, the motor controller boards were eliminated and replaced with only their primary components onto the second protoboard shield in the new 3-D lidar system. Each motor controller board consisted of pin connections to the motor and the Arduino

microcontroller, a ULN 2003a Darlington transistor, power supply connections, and indicator LEDs to show the state of the motor at any given time. As the LEDs are not necessary for proper motor function, they were left off the new system design. One of the components from the motor controller board that was necessary to include is the Darlington transistor, which sends the control signal from the microcontroller to the motor and supplies the motor's power. A second needed component from this board is the female pin connector to fit the male connector of the motor wires (Geeetech Wiki 2012). While the Darlington transistors are short and easily fit beneath the data logger shield installed above the protoboard shield, the female motor pin connectors are too tall and were installed in the area not covered by the smaller data logger shield.

As previously mentioned, the Garmin lidar rangefinder is the same version used in the previous 3-D lidar system. Therefore, the correct connections between the rangefinder, 680 μF capacitor, and the microcontroller pins are already known. The Garmin Lidar Lite v3 rangefinder is connected to through the top data logger shield to allow freedom of motion as it rotates during data collection. This rangefinder has several different modes of operation suitable for numerous purposes ranging between long and short-distance measurements, high speed and high accuracy, and a general balance data collection.

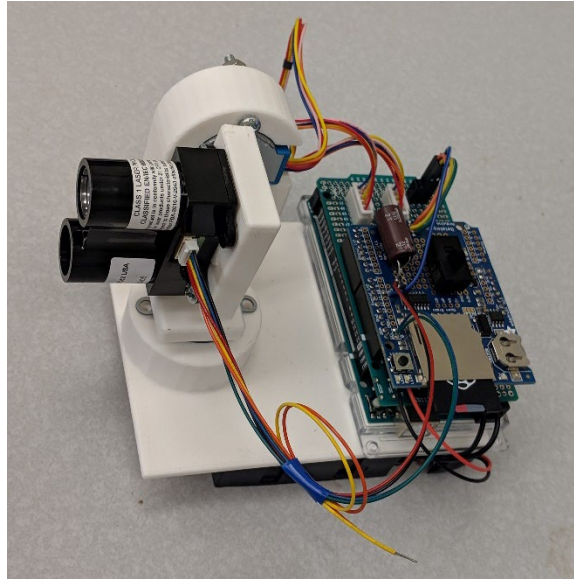


Figure 2.4 The second version of the 3-D lidar system as assembled

Assembled, the new 3-D lidar system is a fraction of the size and weight of the previous system (figure 2.4). Additionally, the new system is entirely self-contained and portable. It does not need to be connected to a power outlet or computer to receive power or save data.

2.2 Software Upgrades

The program code has undergone several alterations from the first version of the 3-D lidar system. First, the new code includes a Stepper.h library to simplify control of the motors. Without this library, the eight possible input combinations of the motors' pins must be described at the start of the code explicitly. In the new version, this library reduces the initialization down to one line of code that performs the same function and turns the motors as needed. In particular, the horizontal motor turns one step after every data point until it reaches the end of its sweep angle at which point it turns the opposite way. When the horizontal motor changes direction, the vertical motor turns one step upwards. As a result, the lidar system thoroughly covers a 3-D space and never collects the same data point twice.

The second major change from the first program is the inclusion of a SD.h library. Since the first 3-D lidar system did not utilize an SD or other memory card, there was no need to use such a library. Here, the SD.h library allows the system to communicate with the microcontroller, access information, create, edit, and save data files onto an SD card (Arduino 2019d). The data saved to the SD card is a .txt file consisting of the horizontal motor's angle (azimuth), the vertical motor's angle (elevation), and the lidar distance measurement. The system is programmed to save each data point to a data file and will end the program once the entire set volume has been mapped. Then, the data file is uploaded to a computer for modeling using a similar MATLAB code as prior accomplished. This MATLAB code converts the azimuth, elevation, and distance values to corresponding Cartesian points, filters out extraneous or flawed data points, and creates a scatter plot of the data.

2.3 Initial System Performance Tests

A performance test was run on the upgraded 3-D lidar system by modeling a portion of a room. This serves to identify potential issues with the system and provides a functioning comparison with the previous 3-D system. As stated prior, the lidar data and motor positions are saved to a data file and modeled as a point cloud in MATLAB. The initial positions of both motors and the system's orientation must be noted for each test as these factors will affect the accuracy of the computer model.



Figure 2.5 Corner of room scanned for initial 3-D lidar testing

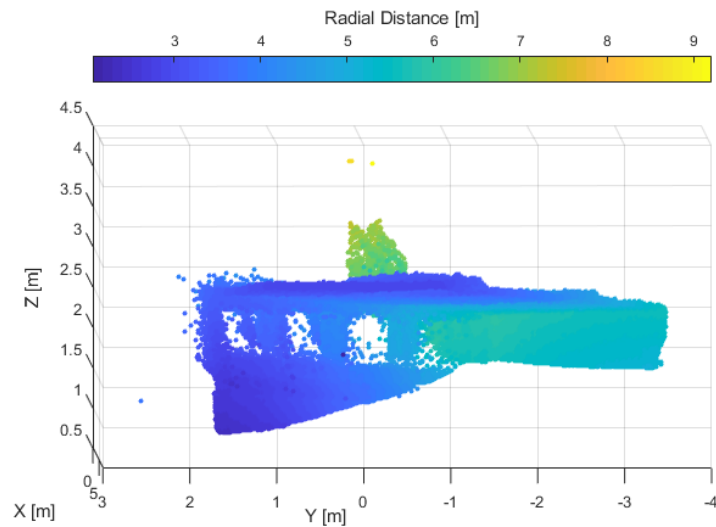


Figure 2.6 Point cloud model of room corner in figure 2.5

The portion of the room in the initial test covers an upper corner, windows, and an exterior wall visible through the windows (figure 2.5). The 3-D lidar system scanned 90° horizontally and 45° vertically in about an hour, resulting in over 123,000 data points. Overall, the resulting model appears accurate (figure 2.6). However, the point cloud model is less accurate around the windows and the recessed lighting in the ceiling. In specific, the model shows large rectangular recesses in the ceiling where lighting is located. These greater recesses

could possibly be due to the corrugated texture of the lighting panels reflecting the lidar. Moreover, it could be a function of using lidar to directly scan a source of near infrared light as it can interfere with its distance calculations. The impact of secondary light sources is discussed further in the following chapter when testing a mobile 2-D lidar system. Furthermore, the 3-D lidar system can detect the exterior wall visible through the windows. This is not surprising as inferred light has a wavelength near visible light and will behave similarly. As the wall can be seen through the glass by the human eye, the 3-D lidar system can also detect it. However, the data collected in figure 2.6 is not perfect as glass can diffract the signal and slightly skew the data.

In order to further investigate the capabilities of the 3-D lidar system when it comes to different materials, a secondary test was accomplished. Since this system is meant to be employed in a transportation environment, the most common material it may encounter outside is water. As such, the system was set to measure an empty beaker, a beaker filled with clean water, and a beaker filled with dirty water. The lidar system was set on a table facing the beakers and the results are provided in figure 2.7

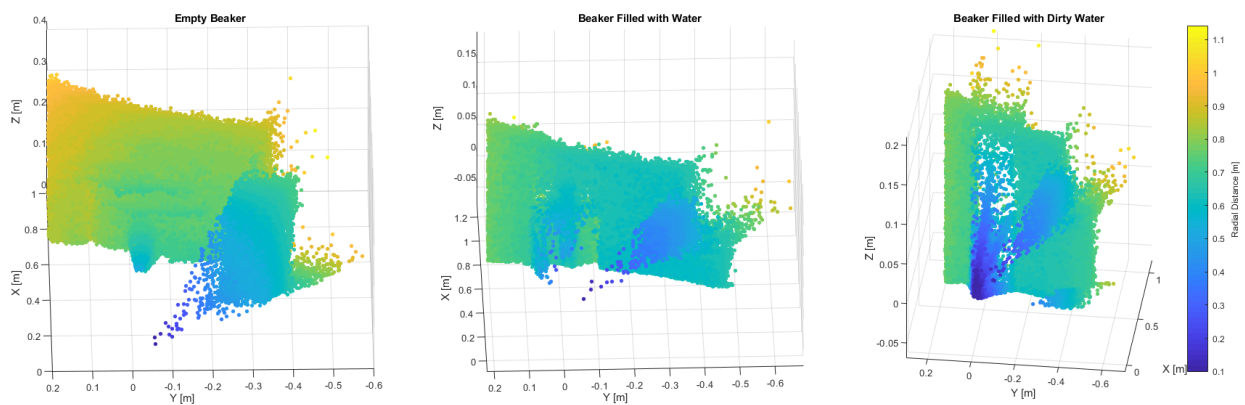


Figure 2.7 Models of empty beaker, beaker filled with clean water, and beaker filled with dirty water (from left to right)

As expected, the empty beaker did not interact with the lidar system as an opaque object would. Instead, both the empty and clean water beakers reflected the lidar signal at the curved edges of the beaker, where the lidar signal would pass through the most amount of solid glass. At a more perpendicular angle, the glass and clean water allow the signal to pass straight through and the system only detects the wall behind the beaker. In comparison, a beaker filled with dirty water provided a greater interaction with the lidar signal; however, it still did not result in a model highlighting a beaker shape. This is an important finding as it might somewhat limit the applications of lidar around bridges and bodies of water.

2.4 Pavement Quality Tests

A potential usage for this 3-D lidar system is to accurately map road conditions and determine pavement quality, such as potholes in the surface of the road. To demonstrate this outcome, the upgraded 3-D lidar system was tested to see if it could accurately model potholes while stationary. Due to the housing design, the lidar rangefinder is unable to point at a steep downward angle. Therefore, to map a pothole, the lidar system must either be placed further away from the pothole, or the system can be turned on its side such that the housing no longer inhibits the motion of the lidar rangefinder. As a result, to increase data point density and limit the possibility of external interference, the lidar system was situated near a couple potholes and oriented sideways.



Figure 2.8 Scanned pothole with a tape measure providing one foot as a reference

The system was then positioned, turned on, and left alone until it completed scanning a couple potholes and their surrounding road surface. One such pothole is illustrated in figure 2.8 with a tape measure indicating a reference foot of measurement. Then, as before, the lidar distance and motor position data, along with the orientation of the system are modeled in MATLAB to produce a rendering of the pothole in figure 2.9.

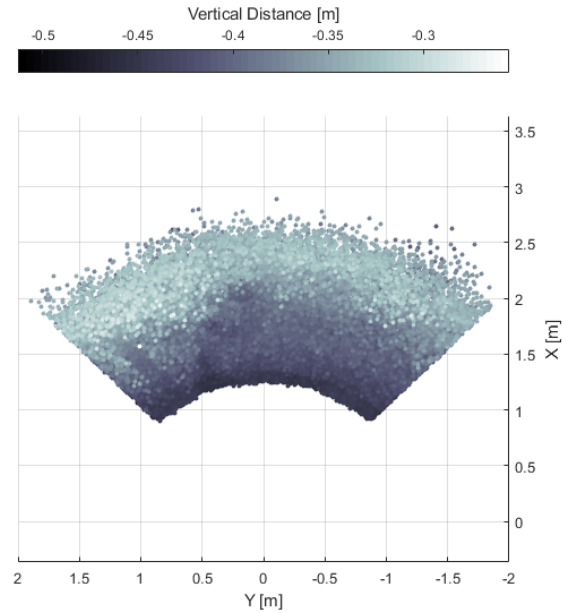


Figure 2.9 Modeled pothole from figure 2.8

The model of the first pothole shows the lower parts in the darker areas. The single straight edge on the left side of the pothole in figure 2.8 can also be seen on the left side of the model in figure 2.9. However, the pothole is relatively shallow and lacks hard edges. Therefore, the rest of the model is difficult to match to the pothole. Moreover, the rotational nature of the data captured does not provide a one-to-one direct comparison between the model and the picture.

The second pothole modeled is a smooth bowl shape in the pavement (figure 2.10). While this pothole is deeper than the first, it still lacks definite edges. As a result, the model of the second pothole shows the dramatic change in pavement surface closer to the lidar system. However, farther away, the changes in the surface become less apparent as the lidar signal interacts with the pavement at a shallower angle (figure 2.11). Again, the rotational nature of the model prevents a straightforward comparison between the model and the picture.



Figure 2.10 The second scanned pothole

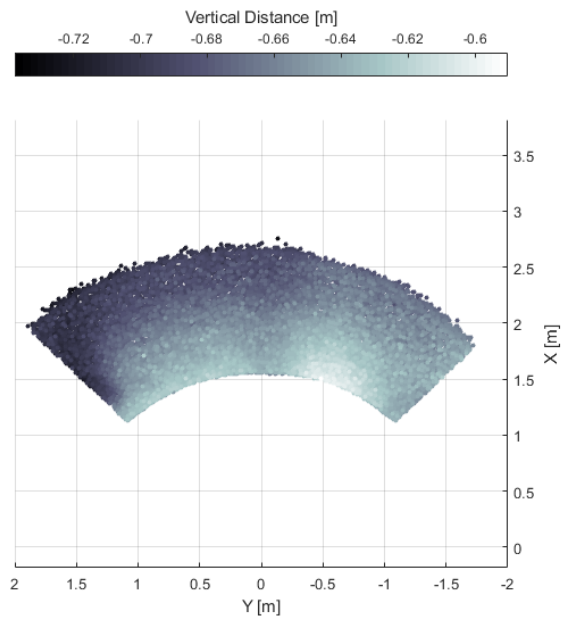


Figure 2.11 The lidar data from the second scanned pothole of figure 2.10

2.5 Future Work

While the 3-D lidar system can reasonably identify dramatic changes in the pavement surface, data collection is relatively slow. Each pothole requires the system to remain motionless for 20 minutes. To increase data collection speed, the step angle of the motors between each data point can be increased. This will decrease the sweep time, but it will reduce the density of the data. Another option to increase data collection speed is to implement a faster data processor in the microcontroller. The Arduino Mega 2560 microcontroller has an operating speed of 16 MHz (Arduino 2019a). A faster microprocessor, such as the Raspberry Pi 4B, which has a 1.5 GHz processor, would be able to map potholes in a fraction of the time (Raspberry Pi Foundation 2019). However, this will come at a greater level of code complexity as the open-source nature of the Arduino software will be replaced with custom-designed algorithms in C++.

Now, if the 3-D lidar system needs to map potholes while moving, an accelerometer should be added to the system. Currently, the lidar system has no method of modelling system motion. Therefore, if the system moves during data collection, then the entire data set becomes skewed and is most likely unusable. As a result, adding an accelerometer will aid the model in determining where each data point is in relation to the other data points.

Finally, with respect to image processing, point cloud resolution is not consistent due to rotation of the sensors that incurs faster rotational velocities at the edge of scanning. This leads to a skewing of the image and a direct comparison with pictures is not possible. Hence, when measuring an obstacle that is not in the line of sight of the sensor, point cloud accuracy is high due to the current slow motion of the motors but the depth accuracy is low, which is sensitive to the orientation of the sensor. Therefore, it would be advantageous to develop a synchronization mechanism that aligns the sampling rate of the sensor with its orientation.

2.6 Potential System Expansion

While the focus has been on developing a cost effective 3-D system that can generate accurate point clouds, it is relatively straightforward to modify the system to create an effective two-dimensional (2-D) system that can help with vehicle classification (e.g., (Asborno, Burris, and Hernandez 2019)) and other transportation-related research activities. Specifically, the code can be changed to keep the horizontal motor stationary and only turn the vertical motor so the lidar rangefinder moves in one direction. Moreover, by increasing the angle between each data point and decreasing the point density, the rangefinder will move faster to better map moving vehicles. This will need to be done in combination with saving the data on the SD card every ten or so data points instead of after every single point. Another option on data collection would be to connect the system directly to a laptop and save the data via the Serial Monitor feature of the Arduino hardware (i.e., similar to what was accomplished in Chapter 1). Finally, by upgrading the code using timestamps on the data, this would facilitate a time history of the data; hence, the system could be set up at any intersection, highway point, or railroad crossing.

Chapter 3 Data Collection from a 2-D lidar System Designed for Bicycles

In the previous year of this effort, a two-dimensional (2-D) lidar system was developed that encountered limited success in recognizing moving vehicles (Blankenau et al. 2018). Based on these findings, a re-imagining of this system was undertaken to better detect vehicles while situated on an electric bike (e-bike). Moreover, a methodology to inform the rider of e-bike via Light Emitting Diodes (LEDs) was incorporated to enhance the rider's safety. This chapter explains the hardware, software, and results obtained for this new system.

3.1 Third Generation 2-D lidar System Hardware

The third generation 2-D lidar system uses a different lidar rangefinder from the previous versions described in Blankenau, et al. (Blankenau et al. 2018). In specific, this new system uses a Terabee Evo 60 m single point lidar rangefinder (figure 3.1). Here, the Terabee's most significant improvement over the Garmin Lite v3 lidar rangefinder is its detection range. While the Garmin has a maximum range of 40 m, the Terabee Evo 60 m has a maximum range of 60 m (Garmin Ltd. 2016, TeraBee 2018a, 2017). For the purpose of moving vehicle detection, a farther detection range increases the likelihood of sensing incoming vehicles and gives the rider a longer reaction time. In addition, the Terabee was designed for drone applications; hence, it is smaller, costs marginally less, and weighs half as much as the Garmin lidar version. Furthermore, the Terabee does not require external circuitry to be controlled by an Arduino microcontroller (Garmin Ltd. 2016, TeraBee 2018a). The Terabee Evo 60 m sensor uses the Inter-Integrated Circuit (I²C)/Universal Asynchronous Receiver/Transmitter (UART) backboard so it can connect to the microcontroller using only two communication wires without extra circuitry (TeraBee 2018a, 2017).

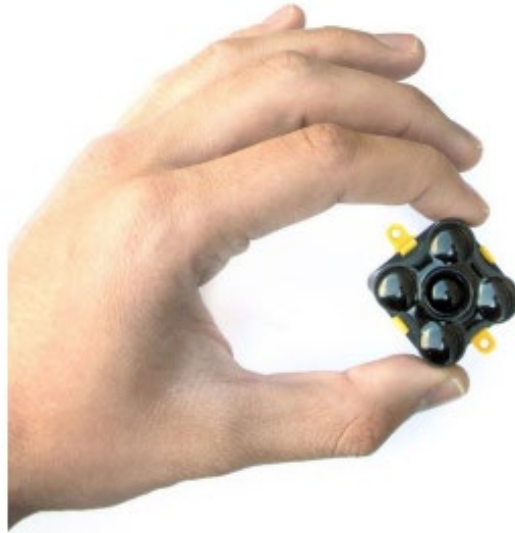


Figure 3.1 Terabeer 60 m Evo lidar distance sensor (TeraBee 2017)



Figure 3.2 Arduino Mega 2560 microcontroller (Arduino 2019a)

The Terabeer lidar sensor, along with the rest of the system, is connected and controlled by an Arduino Mega 2560 microcontroller (figure 3.2). The Arduino range of microcontrollers was preferable to other brands due to its extensive online documentation, open-source software, and ease of learnability. While the Mega 2560 is larger than other Arduino microcontrollers, it has 54 digital input/output (I/O) pins that facilitate communication with the lidar range finder,

stepper motor, micro Secure Digital (SD) card breakout board, and LEDs (Arduino 2019a). Additionally, the Mega 2560 is powered by a 9 VDC battery while operating and supplying a nominal 5 VDC necessary to power the various aspects of the entire system (figure 3.3) (Guadalupi 2019).

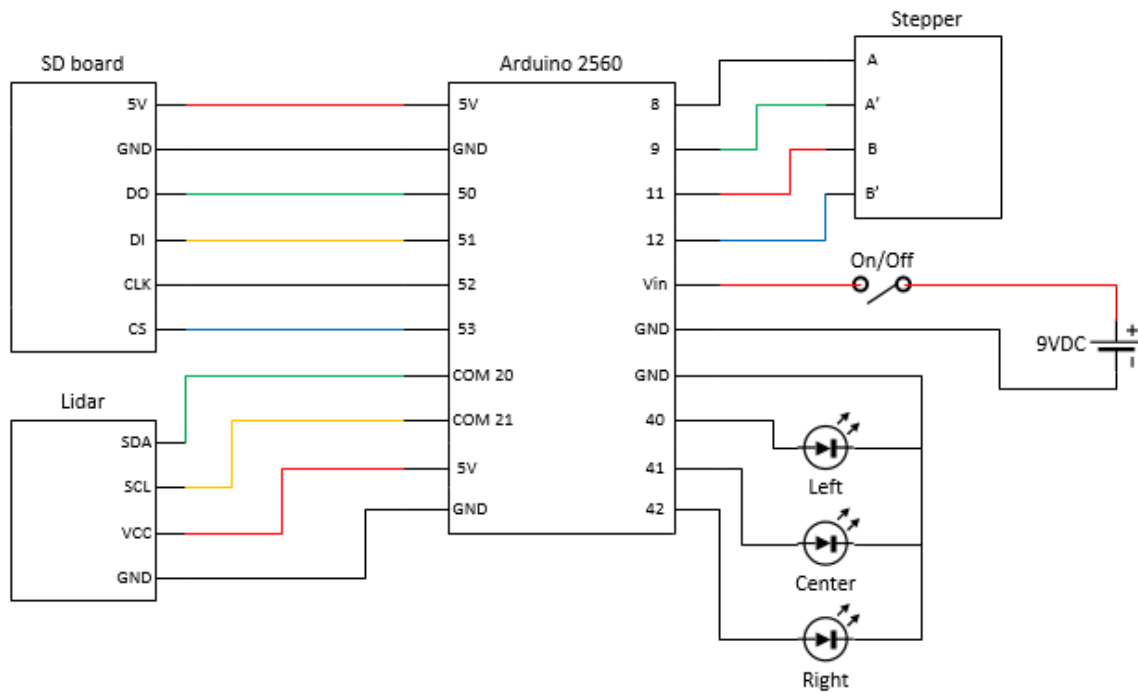


Figure 3.3 Third-generation 2-D lidar system circuit diagram

The microcontroller is connected directly to a QSH2818 stepper motor (figure 3.4) through four digital output pins without need for an external motor driver board to complicate the circuit or programming. The bipolar stepper motor is rated for 3.8 VDC to 6.2 VDC and has a NEMA 11 construction. Bipolar stepper motors have only four lead wires compared to unipolar stepper motors that have either five or six; hence, this results in a simpler circuit (Condit and Jones 2004). This motor's small size and weight are ideal for the goals of the mobile lidar

system. While the maximum torque output of the motor is 0.07 N·m, the lidar rangefinder only weighs 12 grams; therefore, a larger torque capability is not a priority (TeraBee 2017, Trinamic Motion Control GmbH & Co. KG 2019).



Figure 3.4 Bipolar stepper motor used in the 2-D lidar system (Trinamic Motion Control GmbH & Co. KG 2019)

As a stepper motor turns, it does so in discrete increments that allows the lidar rangefinder to remain at a fixed position during each data sample. The motor chosen has 200 distinct steps per revolution; therefore, each step angle is 1.8° in rotation (Trinamic Motion Control GmbH & Co. KG 2019). While stepper motors with smaller step angles exist, for the purposes of vehicle detection, an average vehicle would have to be just over 60 m away from the lidar system for the change in motor angle to miss the vehicle entirely. Given this information and, as previously mentioned, the lidar rangefinder will only have a maximum distancing range of 60 m under ideal conditions, a smaller step angle was determined to be unnecessary (TeraBee 2018b). Finally, the motor's flat-sided shaft ensures the lidar camera's mounting will turn with the motor without slipping (Trinamic Motion Control GmbH & Co. KG 2019).

The final component powered and controlled by the microcontroller is the Adafruit micro SD breakout board (figure 3.5). While the Mega 2560 microcontroller can store variables and code script, it cannot store large amounts of data by itself. Typically, it is connected to a computer through the universal serial bus (USB) port and the computer stores the data. However, to keep the system mobile, it must be able to store data collected by itself. The micro SD breakout board can readily store large data files and is connected to the Arduino board through one of the in-circuit serial programming (ICSP) pins. While the micro SD board runs on a nominal voltage of 3.3 VDC, it also has a 5 VDC pin connected to an onboard fixed-output voltage regulator to lower the voltage and increase the current throughout the board (Texas Instruments 2016). The micro SD board runs at a relatively high current of 100 mA, twice as much as the Mega 2560 is capable of supplying through the 3.3 VDC power output pin (Arduino 2019a). To make certain the micro SD board will always have the required current, it must be powered by a 5 VDC power output pin by the Arduino microcontroller. In addition, this removable data card allows access to the saved data file without disturbing the rest of the lidar system.

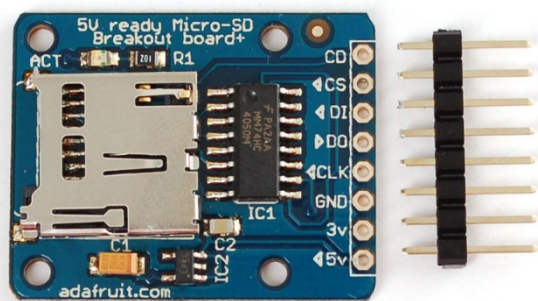


Figure 3.5 Adafruit micro SD card breakout board (Adafruit 2019)

As previously mentioned, the entire 2-D lidar system is powered by a single 9 VDC rechargeable battery. Here, the Arduino microcontroller requires an input voltage between 7 VDC and 12 VDC to adequately supply either 5 VDC at 20 mA or 3.3 VDC at 50 mA to external components (Arduino 2019a). There are three pins available to output 5 VDC to the micro SD card board and the lidar rangefinder to prevent current from being divided between the components. Moreover, there are digital output pins for power, to control the stepper motor, and to turn on blind spot LEDs added to alert the rider. Arduino microcontrollers are designed to run their program if they are properly powered. Therefore, a single-pole, single-throw (SPST) on/off switch was installed between the 9 VDC battery and the power supply pin on the Mega 2560 microcontroller. When off, the battery's positive line is not connected to the Arduino microcontroller. Once the switch is turned on, the battery is connected, the microcontroller receives adequate power and runs the pre-loaded program from the beginning continuously until the switch is turned off.

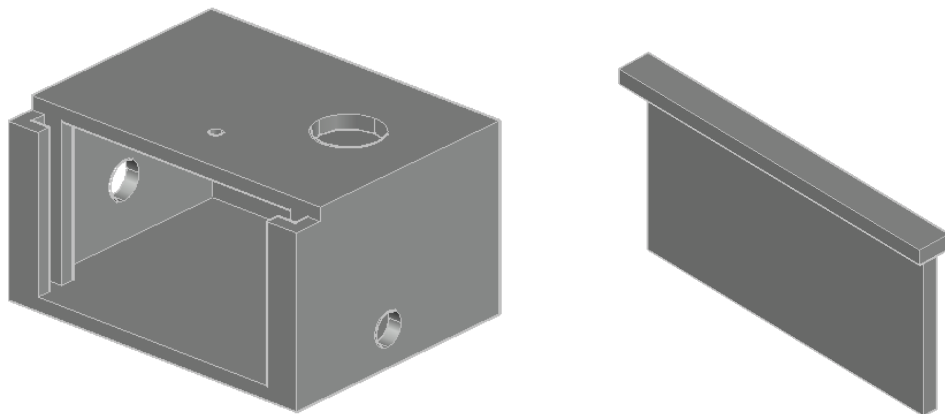


Figure 3.6 Main housing box model for 2-D lidar system (left) and removable front wall (right)

Housing and mounting components were 3-D printed using computer-aided design (CAD) software as illustrated in figure 3.6. There were several requirements for the housing component: it must allow easy access to the circuitry and have holes for the power switch, motor shaft, lidar rangefinder, and blind spot LED wires. Moreover, it must keep out any dirt and water the system could encounter while on the back of an e-bike. The largest component, the Arduino Mega 2560 microcontroller, determined the size and shape of the housing box. The microcontroller fits along the back side of the housing wall, facing the removable front wall to readily monitor pin connections. The power switch is mounted to the left side, the motor shaft goes through the top, and LED wires are fed through the housing's right side. Additionally, a slip ring is installed on the top of the housing, connecting the lidar rangefinder to the microcontroller, allowing the camera to turn freely without twisting wires. The front wall is recessed from the rest of the housing to allow a joining slot and can be removed when lifted. This provides access to the microcontroller for reprogramming throughout testing along with access to the battery for charging, as well as the micro SD card for retrieving data.

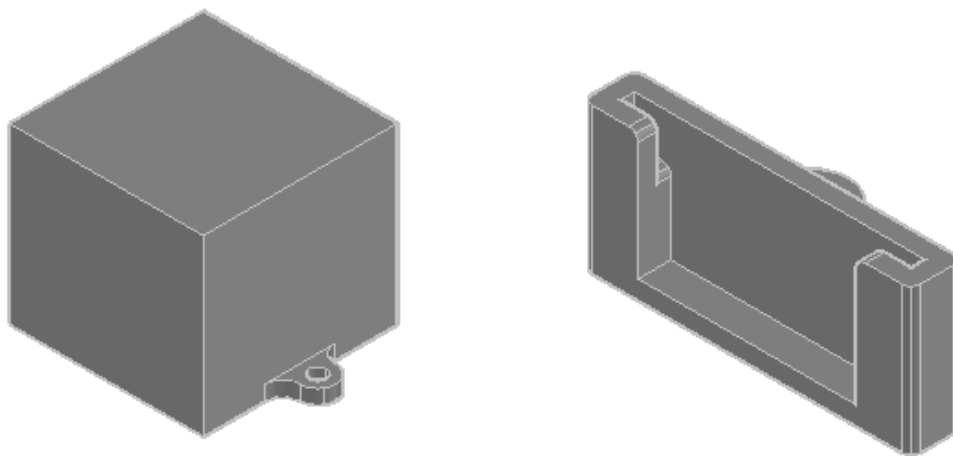


Figure 3.7 3-D CAD models of block stand for motor (left) and lidar rangefinder mount (right)

Due to the housing's height, the stepper motor requires a small block stand so the shaft can reach through the top. The final 3-D component firmly connects the lidar rangefinder to the motor shaft such that it will turn with the motor and not lift off or jostle when the e-bike hits a bump in the road (figure 3.7). The housings require 16.5 cubic inches of 3-D printing plastic. The entire system is just over 4 inches tall, 5 in long, and 3.85 in wide. When assembled, the lidar system weighs roughly 1 pound. Additionally, the system costs roughly \$320 excluding 3-D printing costs (figure 3.8).

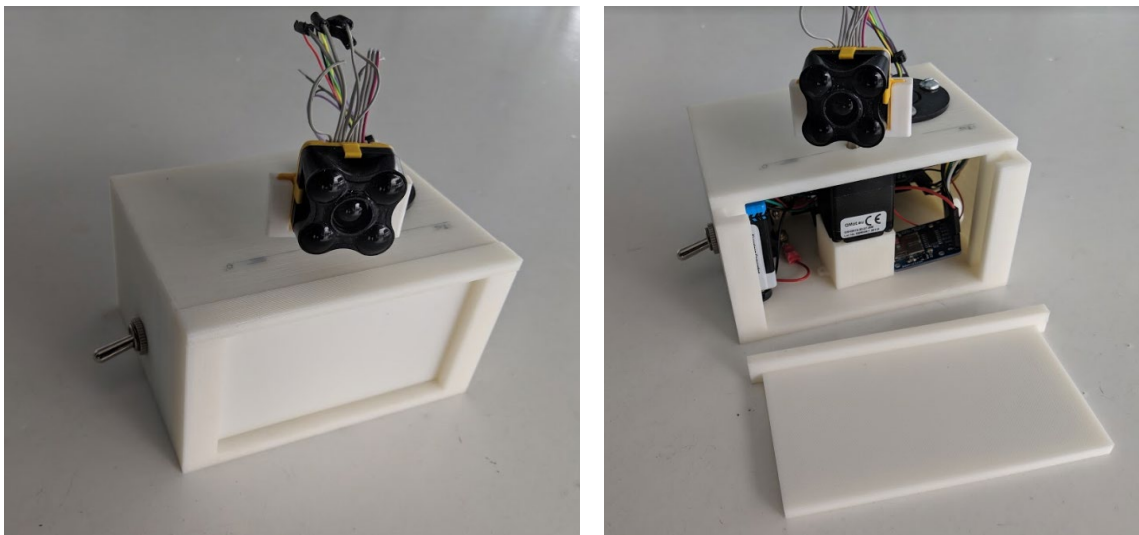


Figure 3.8 Assembled 2-D lidar system closed (top left) and wall removed (top right)

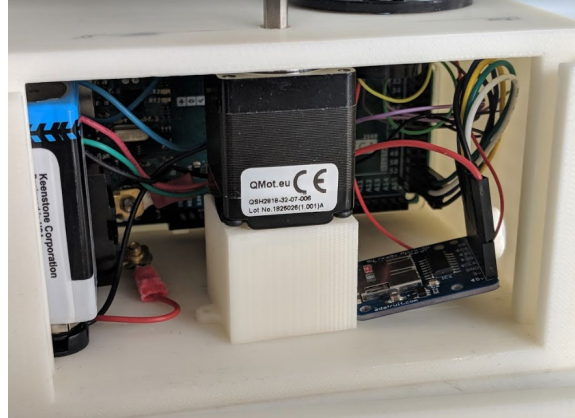
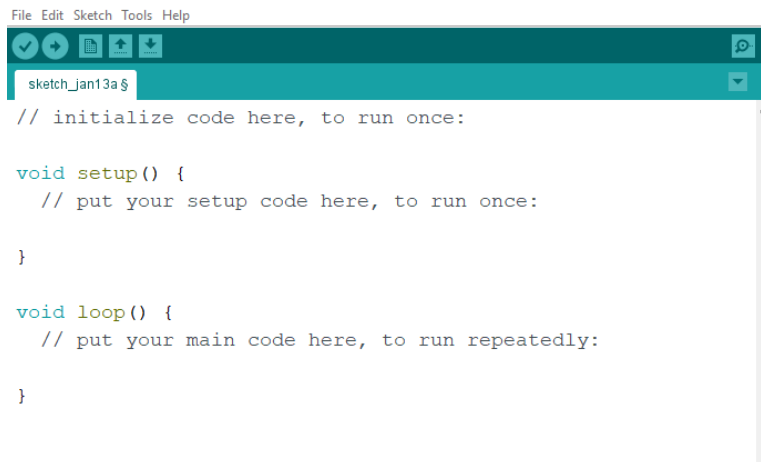


Figure 3.8 cont. Assembled 2-D lidar system interior (bottom)

3.2 Third Generation 2-D lidar System Software

The Arduino microcontroller uses C++ programming via combined .ino files and there are extensive open-source libraries and coding examples available online, as well as wiring connections between the Mega 2560 and each component. In combination, there are collections of coding functions via libraries (.h files) that serve complementary purposes. For example, the SD library has several functions that work to communicate with an SD card connected to the microcontroller. These functions write data, read data, open data files, and erase data files from SD cards. In general, libraries allow a program to replace dozens of lines of code with one function to accomplish the same task. For the 2-D lidar system, only three libraries were required. The Wire.h library allows for I²C communication along Serial Data Line (SDA) and Serial Clock Line (SCL) options used on the Terabeelidar range finder (Arduino 2019d). The SD.h library will, among other things, write and save data to the micro SD card (Adafruit 2019, Arduino 2019d). Finally, the Stepper.h library dramatically simplifies commands for the stepper motor (Trinamic Motion Control GmbH & Co. KG 2019, Arduino 2019d).

The image shows a screenshot of the Arduino IDE interface. At the top, there is a menu bar with 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for file operations and a search icon. The main window displays the code editor for a sketch named 'sketch_jan13a'. The code is as follows:

```
File Edit Sketch Tools Help
sketch_jan13a$
// initialize code here, to run once:

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Figure 3.9 Empty sections of Arduino C++ code

All Arduino program codes have three main sections (figure 3.9). The first section loads and initializes the libraries and sets up the constants and variables that will be used in the code, as well as their data types. These data types include floating-point numbers (decimal values), integers (whole values), unsigned (value magnitudes), byte storage (any sized object in bytes), and characters (readable letters and words) (Arduino 2019c). The next section of code is the setup and includes items the program only needs to run once upon startup. This section begins the communication between the Arduino and external components in the system, creates data files, and defines pins as output or input signals. The final section of code is the loop that repeats until the code reaches some stall condition or the power supply is disconnected. This section is typically where most of the programming takes place and can involve smaller conditional loops, variable calculations, library functions, and responses to various input signals (Arduino 2019c, d).



Figure 3.10 Blind spot LEDs mounted on e-bike handlebars to identify obstacles in left, center, and right lanes.

Here, the setup section opens communication over I²C to the lidar rangefinder. Next, it identifies the blind spot LED pins (figure 3.10) and declares them as output signals. Then, it creates, sets up, and saves the data file onto the SD card. Finally, the setup section moves the stepper motor into its starting position. The loop section begins by collecting the line-of-sight distance reading from the lidar rangefinder and converts that information, along with the motor position, to Cartesian distances. Next, the time, sweep count, motor angle, line-of-sight distance, and blind spot LEDs statuses are saved to a text file on the micro SD card. Then, the program compares the x and y distance set conditions to determine if there is a vehicle approaching the e-bike in that direction. These conditions are as follows: (1) was an object detected by the lidar rangefinder, (2) is it closer than 30 meters to the e-bike, and (3) is it close enough to the previous data point to be an incoming vehicle. If these conditions are all met, the program will turn on the LED that corresponds to the lane the data indicates, either the right, center, or left lane. The LED will remain on until the rangefinder returns to that point in space and no data fitting a vehicle's criteria occurs. After the LEDs are turned on or off accordingly, the motor turns one step either

clockwise or counter-clockwise. Overall, the motor sweeps an area of roughly 100° starting at 40° from perpendicular to the direction of the e-bike. At this point, the program loops back to take another data point from the lidar rangefinder and repeats the process until the power switch is turned off (see Appendix for code).

3.3 Stationary Data Collection

While the lidar system was designed for mobile use, it is simpler to fix bugs and make functional changes before installing the system on the e-bike. Here, several safety conditions were identified as requirements and measurements of success at the start of testing. A typical reaction time of 2 seconds was determined to be the minimum time needed for a bicycle rider to react to an upcoming vehicle (Jurecki, Stańczyk, and Jaśkiewicz 2017). Assuming, when in motion, upcoming vehicles are moving 20 miles per hour (mph) faster than the electric bicycle. This is a reasonable assumption for urban and suburban areas as it is unlikely an e-bike would legally ride along faster roads, such as highways and freeways. Given the reaction time and speed difference, the critical distance from the e-bike is 58.7 ft (17.9 m). This is the minimum distance behind the e-bike the lidar system must identify a vehicle to signal the rider with enough time to react safely.

A second system requirement is to sweep three lanes behind the bicycle fast enough so a vehicle moving 20 mph faster than the e-bike does not have time to pass the bicycle before the system can identify it. To do so, the lidar rangefinder must sweep from the starting angle through the sweep area of 100° and back within the amount of time it would take a vehicle to drive through the critical distance of 58.7 ft and pass the bike. This results in a minimum motor speed of 16.667 revolutions per minute (rpm) or 1.745 radians per second (rad/s).

The final criterion for success is to distinguish approaching vehicles from stationary or non-vehicle obstacles. Due to the wide variety of vehicle sizes and potential varying speeds between them and the lidar system, this is a more difficult criterion to quantify. Using the minimum step angle possible and the minimum calculated motor speed, the time between each data sample is 0.018 seconds. Assuming an average vehicle speed 20 mph faster than the e-bike and a step angle of 1.8° , the lidar system will theoretically collect between three and six data points per vehicle depending on vehicle size (Federal Highway Administration 2004, Car and Driver 2019). To account for a vehicle closing the distance to the lidar system at a maximum of 30 mph faster than the lidar system if travelling, a point would be at most 2 ft closer than the previously collected data point 0.018 seconds before. If the lidar rangefinder is pointed at the side of the vehicle and turning opposite to the direction of the vehicle's motion, the second data point would be at most roughly 15 ft farther away from the lidar system than the previous data point.



Figure 3.11 Image of initial 2-D lidar system test area

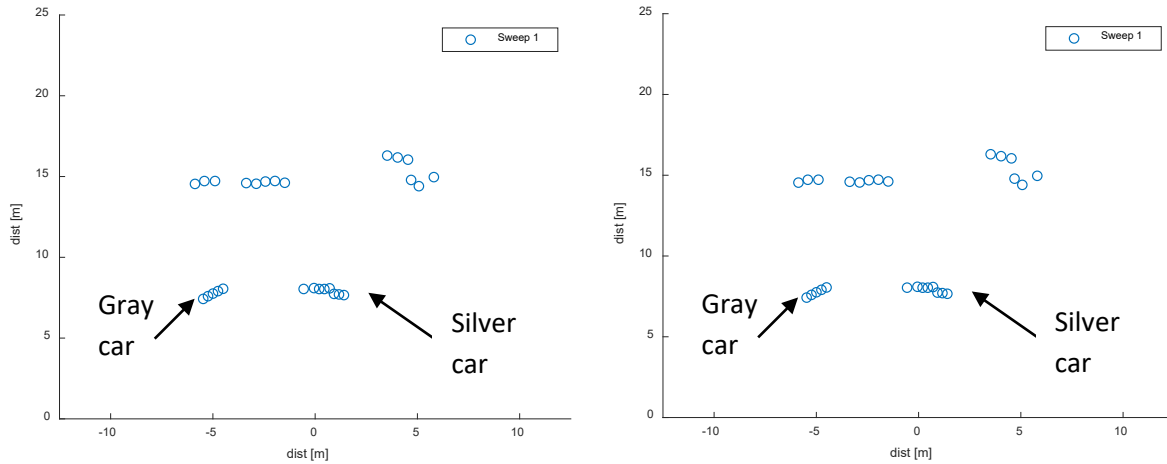


Figure 3.12 Two sweeps of the 2-D lidar system during initial stationary testing

After determining the appropriate criteria, the initial stationary tests had the sole purpose of verifying the functionality and accuracy of the lidar system. While held still at roughly 2 feet above the ground, the lidar system was aimed at static cars in a parking lot (figure 3.11). Overall, the system was successfully able to map the area accurately and showed two cars were in front of the wall of the building (figure 3.12). However, some inaccuracies can be noted in the model recreation of the parking lot. Primarily the data points vary slightly between each sweep of the lidar system. Additionally, the model has a curve to the data given the rotational nature of the system and has difficulty showing the difference between the side and rear of the car on the left.

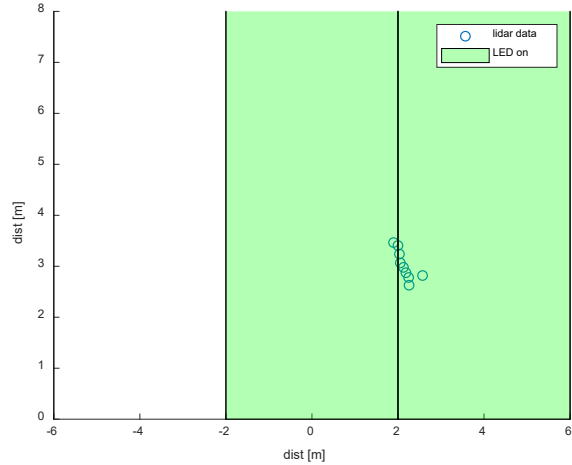


Figure 3.13 Only lidar data registered during stationary test is a false positive of the road divider: visual (left) and data model (right)

The second stationary testing effort was largely unsuccessful. While on the sidewalk, the lidar system was pointed toward oncoming traffic with a speed limit of 30 mph (figure 3.13). It is noteworthy that most vehicles slowed as they neared the system, possibly out of curiosity or safety concerns. Despite the potentially lower vehicle speed, the lidar system almost never collected data points of these vehicles. The time stamp for each data sample showed the lidar system took 1.9 seconds to turn 100° when it should take a maximum of 1 second. Additionally, the blind spot LEDs could not accurately distinguish between moving vehicles and empty space.

The microcontroller's coding was adjusted to address the blind spot LED flaw. It was discovered that the center LED would always remain on due to the lidar rangefinder's signal. If the lidar rangefinder detects no object, the signal received is a measurement of 1.0. This is read by the microcontroller as an object one meter away from the system, which results in a permanent object in the center lane. By filtering these data points, the center LED is able to turn on and off as actual objects enter view. Furthermore, the stepper motor speed was specified to be 16.667 rpm in order to sweep 100° in one second.

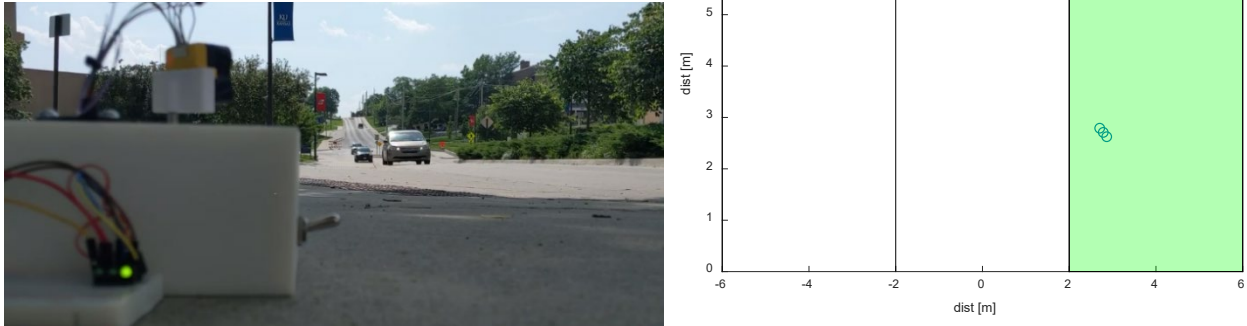


Figure 3.14 Successful stationary testing with moving vehicles in 30 mph speed limit (left) and lit right lane LED with resulting model of snapshot (right)

With these coding changes, the next stationary test had limited success. Here, the lidar system was turned to face oncoming traffic directly (figure 3.14). Out of six passing cars, only one turned into the right lane. Unfortunately, the LED remained on for the next three passing vehicles despite the lidar rangefinder collecting no data from them. Furthermore, there were instances when the lidar rangefinder detected an object that should have triggered an LED to turn on; however, one did not.

Two potential issues are mostly likely to blame for the performance in this stationary test. The first is the elevation of the lidar system. During the tests, the system is resting on the sidewalk and may only interact with the wheels and not the bumpers of most vehicles. The second issue is that the sweep time of the lidar system was still too slow and several vehicles were able to pass through the field of view before the lidar rangefinder is turned in their direction.

The slow rotation speed is affected by the time the system takes to save data to the SD card between each data sample. On average, one line of code running the lidar system takes 0.3 milliseconds (ms); however, a singular line of code that saves the data onto the micro SD card

takes 1.2 ms. Despite best efforts, the code cannot run any faster without risking data corruption. Since the lidar system can be turned off at any moment, the system must save each data point as it is collected or it risks corrupting the entire data set. Therefore, the microcontroller must save each data point as it is collected before turning the stepper motor and collecting the next data point. However, the time required for a single line of code to save to the data file is four times the amount of time as the other lines of code. In addition, the amount of time required to run a loop of the lidar system's code limits the speed of stepper motor rotation. Overall, the optimized code will always take 0.021 seconds to run between each data point collected; hence, resulting in the lidar system operating slower than the programmed stepper motor speed.

Table 3.1 Lidar system timing conditions with a 100° sweep angle and optimized software running time of 0.021 s

Step Angle and Motor Speed	Motor Turning Time [s]	Data Collection Time [s]	Total Sweep Time [s]
1.8° and 16.667 rpm	0.018	0.039	2.17
1.8° and 50 rpm	0.006	0.021	1.50
3.6° and 50 rpm	0.012	0.033	0.92

Therefore, to decrease the time between each data point collected, the motor speed was increased from 16.667 rpm to the maximum usable speed of 50 rpm. However, due to the small turning increments, this speed increase is unable to meet the 1 s sweep time requirement (table 3.1). This increased motor speed decreases the time between each data point from 2.17 s to 1.50 s. As a result, the most effective way to decrease the time taken for the lidar system to sweep 100° is to increase the step angle between each data sample. By doubling the steps between each data point from 1.8° to 3.6°, the amount of data collected is halved and the lidar system does not

spend as much time saving data. Thereby, meeting the previous overall threshold desired. As a result, in theory this system will always detect a vehicle traveling under 20 mph.

Subsequently, the final stationary testing effort was more focused. This involved driving one car towards at 10, 15, 18, and 20 mph towards the lidar system positioned 1.5 ft above the ground on a sunny day. Here, one must take a step back and review the operating principles of lidar. In general, lidar operates via the same fundamentals as radar. A signal is emitted, bounces off a target object, and then is received by the system. The distance to the target is determined from the time delay between emitting and receiving the signal. However, due to the nature of lidar technology, these detection ranges are readily affected by external conditions like infrared (IR) lighting and reflectivity of target's surface material (TeraBee 2018b). Furthermore, lidar rangefinders operate using IR light to bounce off a target object and, as a result, any ambient IR light, typically from sunlight, can interfere with these data. Consequently, using lidar systems on sunny days can dramatically reduce the accuracy and range of the sensor (TeraBee 2018b).

As a result, due to sunny weather conditions during the final stationary tests, the lidar detection distance was reduced and more prone to vehicle detection error. Specifically, the maximum vehicle approach speed registered was 15 mph and it was only represented by four data points in the lidar system (figure 3.15). At higher vehicle speeds, the car passed through the shortened range of detection faster than the lidar system was able to rotate. Therefore, to capture vehicles moving at higher speeds, the lidar rangefinder would have to sweep the area faster or operate under more favorable weather conditions to extend the detection range and increase the time a vehicle would be noticeable. Unfortunately, the only way to augment the lidar rangefinder speed without altering hardware or electronics is to increase the step angle again and lose data density. This would not be beneficial since having data points wider apart would increase the

likelihood of missing a passing vehicle and/or it would have too few data points for the system to recognize a vehicle. Furthermore, the weather conditions are outside the possibility of control and as such, the lidar system must be able to identify vehicles in most every situation. This is particularly true for weather conditions favorable for bicycle riding; i.e., bright and sunny days.

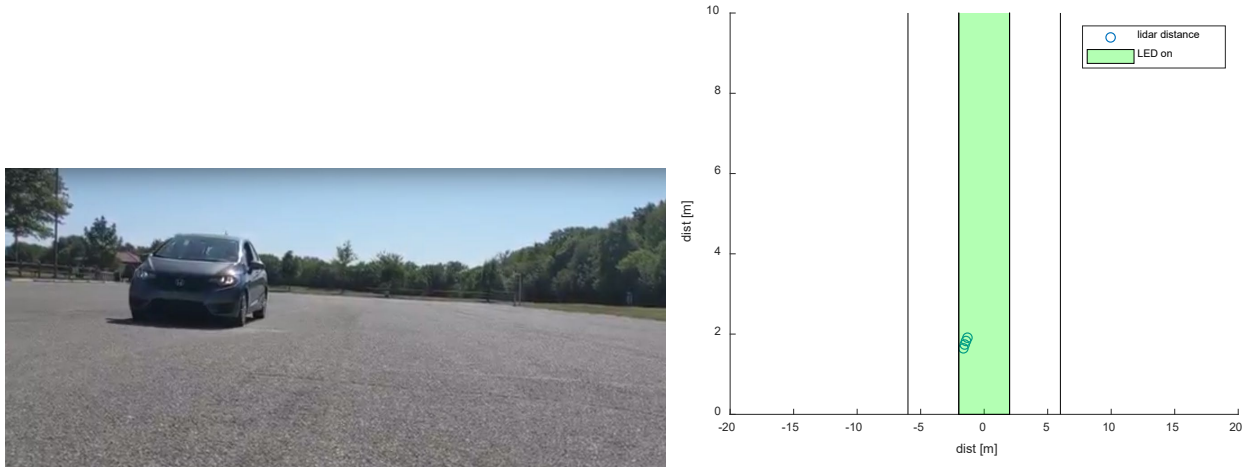


Figure 3.15 Visual of car (left) registered at a maximum speed of 15 mph (right)

To balance the speed of rotation of the lidar rangefinder and the density of data points collected, the motor speed was kept at 50 rpm and the step angle was set at 3.6° . Due to the low torque capabilities of the motor, it is unable to turn faster. Additionally, turning one 1.8° step between data points will not meet the previously calculated system rotation speed due to code processing. Finally, any step greater than 3.6° risks too much space between data points for a vehicle to be missed or not register enough data points to be recognized as a potential vehicle.

3.4 Mobile Data Collection

With the lidar system operating at the best of its capabilities, it was mounted onto the back of an electric bicycle (figure 3.16) designed and built by previous students at the University of Kansas (Moore et al. 2015). This e-bike was also used to test the prior vehicle detection lidar

system (Blankenau et al. 2018). A large bracket was installed onto the e-bike to hold the lidar system at a suitable height above the ground to better reflect the signal off the front bumper of the car. Specifically, the front bumper has a perpendicular angle of incidence to the lidar signal and is a more reliable part of the vehicle to detect. While the windshield offers a larger target, they are slanted and made of glass that offers poor reflection capabilities.

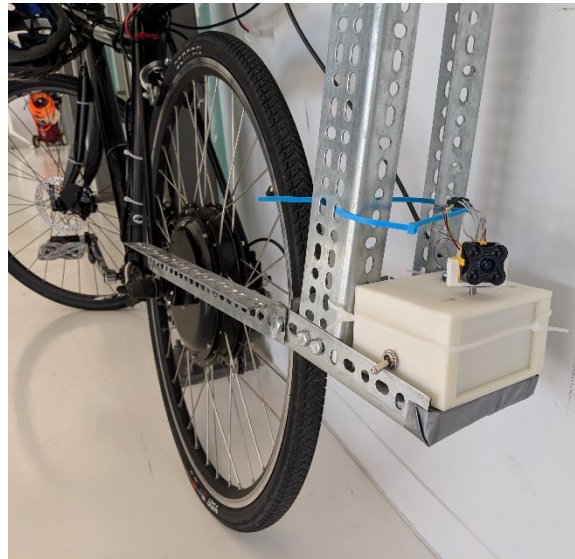


Figure 3.16 Completed 2-D lidar system mounted on the back of the electric bicycle

The lidar system was installed on the bracket of the e-bike and had blind spot monitoring LEDs connected to the Arduino microcontroller through a hole in the side of the 3-D printed housing, subsequently attached to the front of the e-bike at the handlebars. The LED states are recorded in the data .txt file along with the lidar distance measurements and motor sweep count. When a blind spot LED turns on, it is modelled with the lidar data by showing that lane in green. In an attempt to protect the lidar system from the motions and jostling of the e-bike (found in the prior effort to impact the accuracy of the system), a block of insulating foam was attached to the

shelf of the bracket under the system. However, after the first mobile test, the lidar system had a slight downward angle which affected the results. As a result, the system would often get signals reflected from the ground roughly 7 m (20 ft) behind the e-bike (figure 3.17). To correct the angle of the lidar system, the insulating foam was carved at an angle so lidar system became level with the road surface.



Figure 3.17 Typical visual behind e-bike (left) and sweep data (right) from first mobile test with downward lidar angle

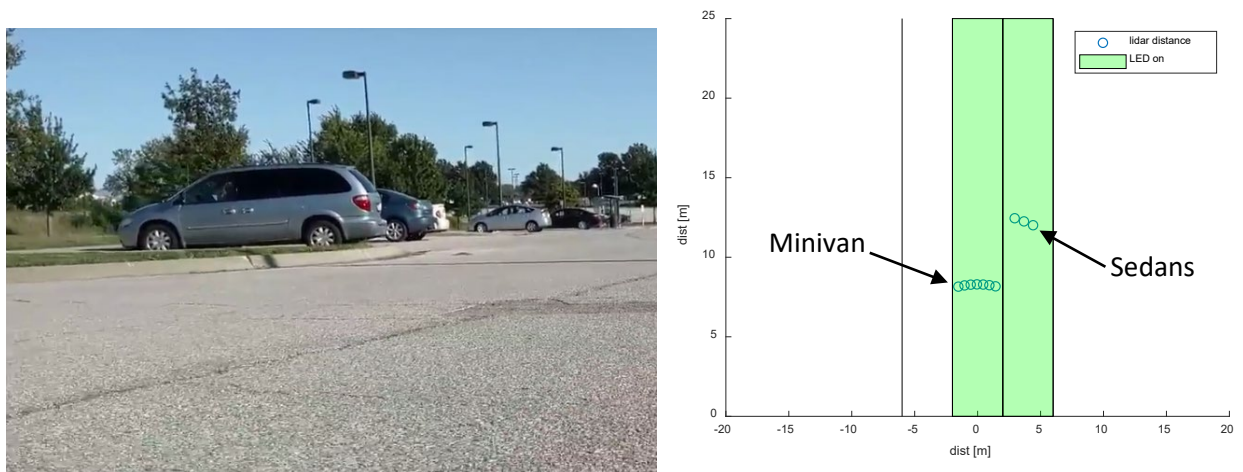


Figure 3.18 Example of successful data collection and blind spot monitoring of stationary vehicles during a mobile test: visual camera (left) and lidar modeling (right)

After fixing the angle of the lidar system, a mobile test occurred that involved riding the e-bike around a parking lot next to parked cars, slow-moving cars, and bushes at the edge of the pavement. Throughout this test, the lidar system was powered on and collected lidar data, motor angles, and the states of the blind spot LEDs. In addition, a video camera was set up to record the area behind the e-bike to match the lidar data to specific objects.

This mobile testing demonstrated several promising results. The lidar system was able to detect stationary (figure 3.18) and slow-moving vehicles (figure 3.19) in a parking lot at accurate distances and positions relative to the e-bike. However, moving vehicles were more likely to be missed as they typically do not register as many data points using the lidar sensor.

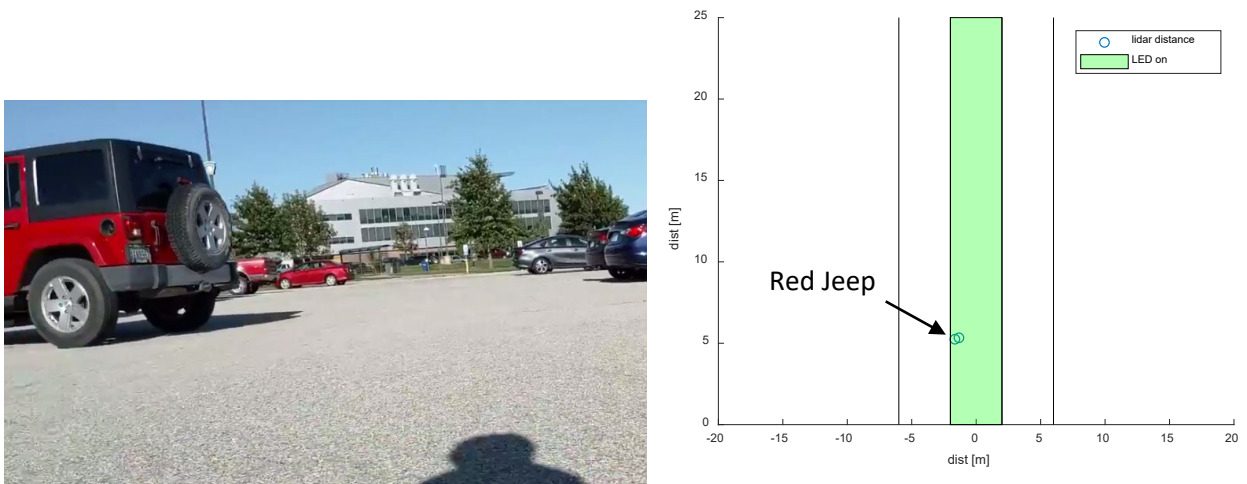


Figure 3.19 Example of successful data collection and blind spot monitoring of moving vehicle during a mobile test: visual camera (left) and lidar modeling (right)

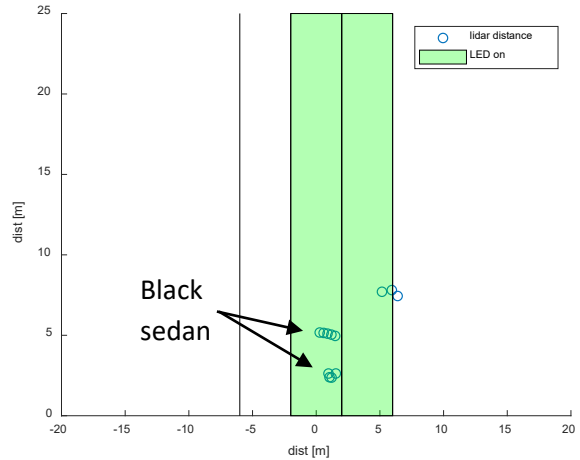
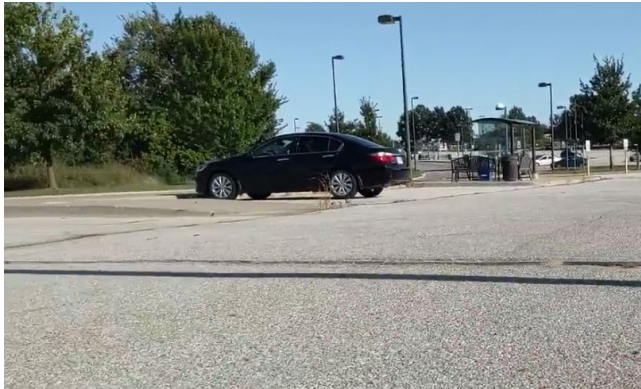


Figure 3.20 Skewed data caused by rapid turning of e-bike during data collection: visual camera (left) and lidar modeling (right)

Unfortunately, the lidar system is not able to account for every condition. During mobile testing, the e-bike turns slightly while in motion. Therefore, it is not always oriented in the same direction throughout a single sweep. This can lead to data points appearing behind other data points along with other skewed data results (figure 3.20). Additionally, when the e-bike tilts on its side while turning, the lidar system is momentarily pointed at the ground on one side and cannot distinguish these data points from actual obstacles or vehicles (figure 3.21). However, these false positives occur nearly every time the e-bike turns; thus, making them predictable. Here, adding another criterion to the microcontroller code to ignore lidar data too close to the lidar system would eliminate these false positives but might result in an increased risk to the rider due to close vehicles.

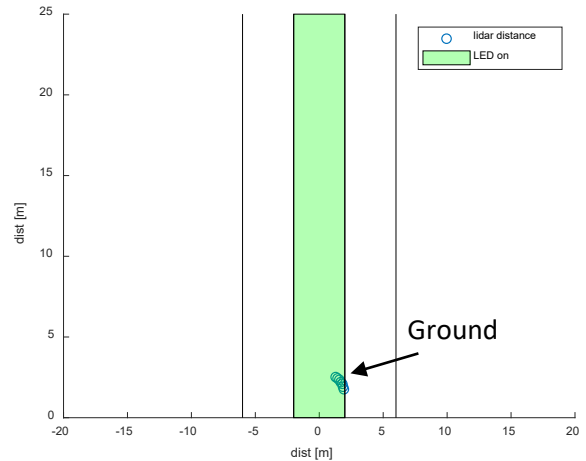


Figure 3.21 Leaning while turning the e-bike causes a false-positive result: visual camera (left) and lidar modeling (right)

Finally, the lidar system is sensitive to jostling. Sharp vertical motion, caused by a pothole or large crack in the pavement, can cause a momentary loss of power to the system. As a result, the program restarts after the bump and the motor turns as if it is at the start of a sweep. This can cause the lidar system to only scan on one side of the e-bike. Therefore, all mobile testing must be done cautiously and at a slower speed to minimize the impact to the system. The best potential solution here is to create a more secure connection between the microcontroller and the rest of the lidar system (similar to what was accomplished in Chapter 2 for the upgraded 3-D lidar system) and implement a better shock absorbing system than foam.

3.5 System Diagnosis

This third generation 2-D lidar system included several changes from the previous version discussed in the prior report. This new system more closely resembles the first generation system, as it is simpler in construction. Arduino microcontrollers are more multi-purpose, easier to learn, and adaptable with circuitry design changes in comparison to the Raspberry Pi and Adafruit Feather stackable system used in the second generation lidar system.

However, this ease of use and design comes with slower processing speeds. The Raspberry Pi Model B microcontroller operates at 1 GHz as compared to the Arduino Mega 2560 at 16 MHz. Therefore, improving performance of the lidar system requires returning to the faster Raspberry Pi version to decrease the time required between each data point collected. This does come with a significantly increased level of programming difficulty.

A Terabee lidar rangefinder is used with the new lidar system because it has a greater distancing range of 60 m than the Garmin LIDAR-Lite v3's 40 m range at a comparable size and weight. This lidar rangefinder also does not require an external capacitor; hence, simplifying the system. Furthermore, the new lidar system does not include a visual camera as the previous lidar system. This camera, while important for a visual record of testing also uses the OpenCV vehicle recognition software to visually distinguish between cars and other objects. This software requires a large database of known vehicles and non-vehicle images to compare to the data. This might not be necessary for a lidar system and it adds more circuitry while slowing down data processing.

The majority of the previous lidar system's weight is a result of the battery pack made of eight AA batteries. The new lidar system requires only one 9 VDC battery. Additionally, the new system is more user-friendly. It includes a single on/off switch, and markings on the housing to show the lidar rangefinder's starting position. Furthermore, all circuitry is contained inside a 3-D printed housing to prevent dirt or water affecting or damaging the system. Moreover, the addition of a slip ring allows the lidar rangefinder to turn freely without twisting or pulling on the wires connected to the microcontroller. Finally, this new system includes LEDs and coding to attempt vehicle recognition solely from lidar data.

Any future lidar systems based on this or previous systems should include faster microcontrollers. The Raspberry Pi used prior is a better choice than the Arduino Mega 2560 and the Terabee 60 m Evo lidar rangefinder has a better range and simpler operation than the Garmin LIDAR-Lite v3. Additionally, the compact structure and direct soldering connections on the prior lidar system should reduce potential wiring issues while the system is jostled. Furthermore, a 3-D accelerometer could add another source of information for the system to help remove erroneous data points. This information could tell when the e-bike is turning or leaning, and potentially adjust for skewed data and ignore false-positive results from the ground.

Overall, lidar systems will always suffer from temperamental operation. Weather conditions will continue to affect ranging distances, and surface reflectivity, opacity, and angle of incidence can alter lidar readings. Additionally, true vehicle identification may never be possible using only 2-D lidar systems as a grouping of data points simply indicates roughly an object's width. The different variables affecting how a vehicle or other object may approach a lidar system means it might be impossible to account for every situation with a single model. Finally, more work is needed to provide repeatable data suited for various traffic and weather conditions.

References

- Adafruit. 2019. "Micro SD Card Breakout Board Tutorial." <https://learn.adafruit.com/adafruit-micro-sd-breakout-board-card-tutorial?view=all>.
- American Society of Civil Engineers. 2019. "2017 Infrastructure Report Card." <https://www.infrastructurereportcard.org>.
- Anarkooli, A. J., M. Hosseinpour, and A. Kardar. 2017. "Investigation of factors affecting the injury severity of single-vehicle rollover crashes: A random-effects generalized ordered probit model." *Accident Analysis and Prevention* 106:399-410. doi: 10.1016/j.aap.2017.07.008.
- Arduino. 2019a. "Arduino Mega 2560 Rev3." <https://store.arduino.cc/usa/mega-2560-r3>.
- Arduino. 2019b. "Arduino mega Proto Shield Rev3 (PCB)." <https://store.arduino.cc/usa/arduino-mega-proto-shield-rev3-pcb>.
- Arduino. 2019c. "Language Reference." <https://www.arduino.cc/reference/en/>.
- Arduino. 2019d. "Libraries." <https://www.arduino.cc/en/Reference/Libraries>.
- Arduino. 2019e. "Servo Library." <https://www.arduino.cc/en/reference/servo>.
- Arduino. 2019f. "Stepper Library." <https://www.arduino.cc/en/reference/stepper>.
- Asborn, M. I., C. G. Burris, and S. Hernandez. 2019. "Truck Body-Type Classification using Single-Beam Lidar Sensors." *Transportation Research Record* 2673 (1):26-40. doi: 10.1177/0361198118821847.
- Blankenau, I., D. Zolotor, M. Choate, A. Jorns, Q. Homann, and C. Depcik. 2018. "Development of a Low-Cost LIDAR System for Bicycles." SAE Technical Paper 2018-01-1051. doi: 10.4271/2018-01-1051.

- Car and Driver. 2019. "Smart Fortwo Features and Specs." <https://www.caranddriver.com/smart/fortwo/specs>.
- Chiang, K. W., G. J. Tsai, Y. H. Li, and N. El-Sheimy. 2017. "Development of LiDAR-Based UAV System for Environment Reconstruction." *IEEE Geoscience and Remote Sensing Letters* 14 (10):1790-1794. doi: 10.1109/Lgrs.2017.2736013.
- Condit, R., and D. Jones. 2004. "Stepping Motors Fundamentals." <http://www.t-es-t.hu/download/microchip/an907a.pdf>.
- Earl, B. 2013. "Adafruit Data Logger Shield." <https://learn.adafruit.com/adafruit-data-logger-shield?view=all>.
- Federal Highway Administration. 2004. "Federal Size Regulations for Commercial Motor Vehicles." U.S. Department of Transportation. https://ops.fhwa.dot.gov/freight/publications/size_regs_final_rpt/size_regs_final_rpt.pdf.
- Garcia-Gutierrez, J., L. Goncalves-Seco, and J. C. Riquelme-Santos. 2011. "Automatic environmental quality assessment for mixed-land zones using lidar and intelligent techniques." *Expert Systems with Applications* 38 (6):6805-6813. doi: 10.1016/j.eswa.2010.12.065.
- Garmin Ltd. 2016. "Lidar Lite v3 Operation Manual and Technical Specifications." https://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf.
- Geetech Wiki. 2012. "Stepper Motor 5V 4-Phase 5-Wire & ULN2003 Driver Board for Arduino." <http://eeshop.unl.edu/pdf/Stepper+Driver.pdf>.

- Goniewicz, K., M. Goniewicz, W. Pawlowski, and P. Fiedor. 2016. "Road accident rates: strategies and programmes for improving road traffic safety." *European Journal of Trauma and Emergency Surgery* 42 (4):433-438. doi: 10.1007/s00068-015-0544-6.
- Guadalupi, Arturo. 2019. MEGA2560_Rev3e Schematic.
https://www.arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf.
- Jeon, W., and R. Rajamani. 2019. "Active Sensing on a Bicycle for Simultaneous Search and Tracking of Multiple Rear Vehicles." *IEEE Transactions on Vehicular Technology* 68 (6):5295-5308. doi: 10.1109/TVT.2019.2911572.
- Jurecki, R. S., T. L. Stańczyk, and M. J. Jaśkiewicz. 2017. "Driver's reaction time in a simulated, complex road incident." *Transport* 32 (1):44-54. doi: 10.3846/16484142.2014.913535.
- Kelly, M., and S. Di Tommaso. 2015. "Mapping forests with Lidar provides flexible, accurate data with many uses." *California Agriculture* 69 (1):14-20. doi: 10.3733/ca.v069n01p14.
- Kromer, R. A., D. J. Hutchinson, M. J. Lato, D. Gauthier, and T. Edwards. 2015. "Identifying rock slope failure precursors using LiDAR for transportation corridor hazard management." *Engineering Geology* 195:93-103. doi: 10.1016/j.enggeo.2015.05.012.
- Lienert, P., and S. Nellis. 2019. "Cheaper Sensors Could Speed More Self-Driving Cars to Market by 2022." <https://www.reuters.com/article/us-autos-autonomous-lidar/cheaper-sensors-could-speed-more-self-driving-cars-to-market-by-2022-idUSKCN1TD2MY>.
- Meier, R. 2019. "Roger Meier's Freeware: CoolTerm." <https://freeware.the-meiers.org/>.
- Mole, C. D., and R. M. Wilkie. 2017. "Looking forward to safer HGVs: The impact of mirrors on driver reaction times." *Accident Analysis and Prevention* 107:173-185. doi: 10.1016/j.aap.2017.07.027.

- Moore, P., C. Vande Velde, R. Wagner, and C. Depcik. 2015. "Design and Analysis of Electric Bikes for Local Commutes." ASME 2015 International Mechanical Engineering Congress and Exposition, Houston, Texas. doi: 10.1115/IMECE2015-52135.
- National Center for Statistics and Analysis. 2018. Summary of Motor Vehicle Crashes: 2016 Data. *Traffic Safety Facts*. Washington, DC: National Highway Traffic Safety Administration. <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812580>.
- Neupane, S. R., and N. G. Gharaibeh. 2019. "A heuristics-based method for obtaining road surface type information from mobile lidar for use in network-level infrastructure management." *Measurement* 131:664-670. doi: 10.1016/j.measurement.2018.09.015.
- Puente, I., H. Gonzalez-Jorge, J. Martinez-Sanchez, and P. Arias. 2013. "Review of mobile mapping and surveying technologies." *Measurement* 46 (7):2127-2145. doi: 10.1016/j.measurement.2013.03.006.
- Raspberry Pi Foundation. 2019. "Raspberry Pi 4 Tech Specs." <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/>.
- TeraBee. 2017. "TeraRanger Evo 60m." <https://www.terabee.com/wp-content/uploads/2019/03/TeraRanger-Evo-60m-Specification-sheet.pdf>.
- TeraBee. 2018a. "User Manual for TeraRanger Evo single point distance sensors and backboards." <https://www.terabee.com/wp-content/uploads/2019/03/User-Manual-for-TeraRanger-Evo-single-point-distance-sensors-and-backboards-1-3.pdf>.
- TeraBee. 2018b. "TeraRanger Evo 60m sensor potential maximum range in varying outdoor conditions." <https://www.terabee.com/wp-content/uploads/2019/04/TeraRanger-Evo-60m-Test-Results-Report-Outdoor.pdf>.

- Texas Instruments. 2016. LP2981-N Micropower 100-mA Ultralow Dropout Regulator in SOT-23 Package." <https://www.ti.com/lit/ds/symlink/lp2981-n.pdf>.
- Trinamic Motion Control GmbH & Co. KG. 2019. "QSH2818 Manual." https://www.trinamic.com/_scripts/download.php?file=_articles%2Fproducts%2Fmotors%2Fqmot-qsh2818%2F_datasheet%2FQSH2818_manual.pdf.
- Williams, K., M. Olsen, G. Roe, and C. Glennie. 2013. "Synthesis of Transportation Applications of Mobile LIDAR." *Remote Sensing* 5 (9):4652-4692. doi: 10.3390/rs5094652.
- Xu, J., S. L. Murphy, K. D. Kochanek, B. Bastian, and E. Arias. 2018. Deaths: Final Data for 2016. *National Vital Statistics Reports*. Hyattsville, MD: National Center for Health Statistics. https://www.cdc.gov/nchs/data/nvsr/nvsr67/nvsr67_05.pdf.
- Yang, B. S., Z. Wei, Q. Q. Li, and J. Li. 2013. "Semiautomated Building Facade Footprint Extraction From Mobile LiDAR Point Clouds." *IEEE Geoscience and Remote Sensing Letters* 10 (4):766-770. doi: 10.1109/Lgrs.2012.2222342.

Appendix A

Code 1 2-D lidar system's Arduino microcontroller code

```
#include <Wire.h> //for I2C/TWI communication using SDA and SCL lines
#include <SD.h> //for reading and writing sd cards
#include <Stepper.h> //for controlling stepper motors

const float pi = 3.14159265; //[]
const int phi = 40; //[deg]
const int rpm = 50; //[rev per min]
const int sweepAngle = 100; //[deg]
const float stepAngle = 1.8; //[deg]
const int stepsPerRevolution = 200; //[]
const int chipSelect = 53; //pin
const int leftLED = 40; //pin
const int centerLED = 41; //pin
const int rightLED = 42; //pin
const int carGain = 200; //[mm] =.2[m] (car is 30mph faster than bike)
const int carLength = -4500; //[mm]
int currentDirection = -1; //1 is CW, -1 is CCW
const int criticalDistance = 27000; //27000[mm] = 27[m] ~ 88[ft]
const int leftLaneStart = -2000; //[mm] assuming bike is in center of lane
const int rightLaneStart = 2000; //[mm]
const int leftLaneEnd = -6000; //[mm]
const int rightLaneEnd = 6000; //[mm]
uint8_t evo[3]; //byte storage, no +/- signs, just number
int sweepCount = 0; //[]
float currentAngle = phi; //[deg]
uint16_t currentDistance; //[mm], no +/- signs, just number
float xDistance; //[mm]
float yDistance; //[mm]
float previouslyDistance; //[mm]
float storedLeftAngle; //[deg]
int storedLeftSweep; //[]
float storedCenterAngle; //[deg]
int storedCenterSweep; //[]
float storedRightAngle; //[deg]
int storedRightSweep; //[]

File dataFile; //create data file
Stepper myStepper(stepsPerRevolution, 8, 9, 11, 12);

void setup()
{
  //Setup Lidar Communication
```

```

#define LidarEvo 0x31 //declare address
Wire.begin(); //open communication over I2C

//Setup LED pins
pinMode(leftLED, OUTPUT);
pinMode(centerLED, OUTPUT);
pinMode(rightLED, OUTPUT);
digitalWrite(leftLED, LOW);
digitalWrite(centerLED, LOW);
digitalWrite(rightLED, LOW);

//SD Card Setup
SD.begin(chipSelect);
SD.remove("test.txt"); //delete existing data file
dataFile = SD.open("test.txt", FILE_WRITE); //create blank data file
dataFile.println("Time, Sweep, Angle, Distance, Left, Center, Right");
dataFile.close(); //Move motor to initial position

myStepper.setSpeed(20);
myStepper.step(currentDirection*phi/stepAngle);
delay(100);
}

void loop()
{
//collect lidar distance data
Wire.beginTransmission(LidarEvo);
Wire.write(0x00);
Wire.endTransmission();
delayMicroseconds(500);
Wire.requestFrom(LidarEvo, 3);
evo[0] = Wire.read(); //First byte
evo[1] = Wire.read(); //Second byte
evo[2] = Wire.read(); //Byte of checksum
currentDistance = (evo[0]<<8) + evo[1]; //[mm]
xDistance = currentDistance*cos(currentAngle*pi/180); //[mm]
yDistance = currentDistance*sin(currentAngle*pi/180); //[mm]

//prepare sd card and data file
String dataString = String(millis()) + "," + String(sweepCount) + "," + String(currentAngle) +
"," + String(currentDistance) + "," + String(digitalRead(leftLED)) + "," +
String(digitalRead(centerLED)) + "," + String(digitalRead(rightLED)) + "\n"; //gather current
data measurements
dataFile = SD.open("test.txt", FILE_WRITE); //open data file
dataFile.println(dataString); //add current data measurements
dataFile.close(); //save and close file

```

```

//identify cars and turn on LEDs
if ((yDistance <= criticalDistance) && (previousyDistance-yDistance <= carGain) &&
(previousyDistance-yDistance >= carLength) && (currentDistance != 1) && (currentDistance !=
0)) //criteria that recognizes a closing in car and no null data
{
  if ((xDistance <= leftLaneStart) && (xDistance >= leftLaneEnd)) //only in the left lane
  {
    digitalWrite(leftLED, HIGH); //turn on LED indicator
    storedLeftSweep = sweepCount; //remember the position the car was at
    storedLeftAngle = currentAngle;
  }
  if ((xDistance > leftLaneStart) && (xDistance < rightLaneStart)) //only in the center lane
  {
    digitalWrite(centerLED, HIGH);
    storedCenterSweep = sweepCount;
    storedCenterAngle = currentAngle;
  }
  if ((xDistance >= rightLaneStart) && (xDistance <= rightLaneEnd)) //only in the right lane
  {
    digitalWrite(rightLED, HIGH);
    storedRightSweep = sweepCount;
    storedRightAngle = currentAngle;
  }
}

//turn off LEDs at same angle on next sweep
if ((digitalRead(leftLED) == HIGH) && (sweepCount == storedLeftSweep+1) &&
(storedLeftAngle == currentAngle))
{
  digitalWrite(leftLED, LOW);
}
if ((digitalRead(centerLED) == HIGH) && (sweepCount == storedCenterSweep+1) &&
(storedCenterAngle == currentAngle))
{
  digitalWrite(centerLED, LOW);
}
if ((digitalRead(rightLED) == HIGH) && (sweepCount == storedRightSweep+1) &&
(storedRightAngle == currentAngle))
{
  digitalWrite(rightLED, LOW);
}

//Change direction as needed
if (currentAngle <= phi)
{

```



```
    currentDirection = -1; //CCW
    sweepCount++;
}
else if (currentAngle >= sweepAngle+phi)
{
    currentDirection = 1; //CW
    sweepCount++;
}

//turn motor one step
myStepper.setSpeed(rpm);
myStepper.step(2*currentDirection);

//Update current angle and distances
currentAngle = currentAngle-(currentDirection*2*stepAngle);
previousyDistance = yDistance;
}
```