RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Mechanical and Industrial Engineering
Program: Mechanical Engineering/Industrial Engineering

Course Number	MEC 825
Course Title	Mechanical Design
Semester and Year	Winter 2019
Instructor	Dr. V. Chan

REPORT:	Final Report: LIDAR Navigated Robot Car

Section No.	02
Group Name	Kappastone Design Engineering Consulting
Submission Date	04/05/19
Due Date	04/05/19

Name	Student ID	Signature*
Kevin Lin	33260	
Tianhao William Jiang	38719	
Ryan Yuen	17076	
Sharfaraz Ahmed	38806	

(Note: remove the first 4 digits from your student ID)

**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/policies/pol60.pdf*

RYERSON UNIVERSITY
FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF MECHANICAL AND INDUSTRIAL ENGINEERING

LIDAR NAVIGATED ROBOT CAR

Kevin Lin - 500633260
Tianhao William Jiang - 500638719
Ryan Yuen - 500617076
Sharfaraz Ahmed - 500638806

MEC825 - Design Project Report

Submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Engineering (BEng)

Faculty Advisor: Dr. Siyuan He
Date: April 05, 2019

ABSTRACT

This report presents the design and construction of a LIDAR navigated robot car based on the Mechbot used in the MEC733 course. By implementing LIDAR, this project will aim to improve the navigation capabilities of the Mechbot, which currently uses IR for sensing and navigation.

Utilizing a combination of off the shelf parts such as the Garmin LIDAR-Lite v3HP, a stepper motor, an Arduino microprocessor and a Sparkfun Big Easy Driver, the single point LIDAR device was transformed into a 2-D device by sweeping 180° back and forth on the horizontal axis. However, due to limitations with the speed of the stepper motor, the sweep frequency was limited, which was also lower than the theoretical maximum that the LIDAR-Lite v3HP could support.

To display the scanning mechanism, the programming IDE Processing was used to create a live map of the Mechbot's surroundings. By utilizing the data obtained from the LIDAR unit on the LIDAR Arduino (master), the data can then be sent to the Mechbot Arduino (slave) for navigation and control purposes. Currently, the Mechbot is capable of operating autonomously in a straight line. Future development would allow the Mechbot to turn and navigate around corners and objects.

For future recommendations, a different motor could be implemented to allow for a higher sweep frequency, increasing resolution and response time. A 360° sweep angle could also be utilized, allowing for SLAM to be implemented, greatly improving the navigation capabilities of the Mechbot.

Overall, this project can be considered a success, due to the fact that the existing Mechbot was able to be modified to support LIDAR in a cost-effective manner. This will allow students in upcoming MEC733 courses to experiment with LIDAR during their course project.

TABLE OF CONTENTS

NOMENCLATURE

Acronyms

IR- Infrared
LIDAR - Light Detection and Ranging
LED - Light Emitting Diode
PWM - Pulse Width Modulation
SLAM - Simultaneous Localization and Mapping
DC - Direct Current
AC - Alternating Current
CPU - Central Processing Unit
RAM - Random-Access Memory
ROM - Read-Only Memory
LCD - Liquid Crystal Display
I2C - Inter-Integrated Circuit
SDA - Serial Data
SCL - Serial Clock
RX - Receive
TX - Transmit
RTAB - Real Time Appearance Based (Mapping)

Units

V - Voltage (V)
I - Current (A)
R - Resistance (Ω)
P - Power (W)
J - Moment of Inertia (kg m^2)
T - Torque (N m)
θ - Angular Position (rad)
α - Angular Acceleration (rad s^{-2})

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The objective of this capstone project is to create a LIDAR navigated robot car, based off the Mechbot used in MEC733 under the supervision of Dr. Siyuan He. Currently, the Mechbot utilizes IR for navigation, which is both inaccurate and limited. By adding LIDAR to the Mechbot, it will be able to navigate with higher precision, providing a better learning experience to students in MEC733.

To tackle this project, it was split into multiple stages as shown below:

1. Choose LIDAR system and suitable microprocessor
2. Develop mounting system and purchase/3D print necessary components
3. Create code to translate LIDAR scans into meaningful information used for navigation
4. Initial testing and further refinement

The team chose the LIDAR-Lite v3HP by Garmin as the LIDAR unit. A 180° scan width (half revolution) was chosen with a frequency of 1 Hz due to the low travel speed of the Mechbot. To mount the LIDAR, it was decided that a 3D printed base will be attached to the top of the Mechbot.

In terms of hardware, a stepper motor with a step angle of 0.198° was selected as it eliminated the need for an encoder. The motor dimensions were small enough to fit into the Mechbot and the torque was adequate to drive the LIDAR load as well as the mounting hardware. One LED in reverse and one LED emitting were used as the calibration mechanism for the motor in the event of a power failure or initial startup. Lastly, the Mechbot Arduino was used as a slave to drive the Mechbot motors, controlling the speed and direction.

Throughout the project, the team will optimize the scan time, the control algorithm and the response from the Mechbot to satisfy the project's requirements. The team will strive to develop a working prototype by the end and demonstrate the results to the supervising professor. If successful ,the Mechbots in the MEC733 course will be re-equipped to use the new system developed by the team.

2.0 LITERATURE REVIEW & BACKGROUND RESEARCH

Current solutions for single LIDAR cameras involve an IR camera on a oscillating gimbal to read data. Depending on its application, the sweeping motion of the system either scans in a 2D (single plane scan) [2] or a 3D (multiple angled single plane scan) [1] space in which data is collected. Currently, the most sought after implementation of LIDAR camera systems is in (semi) autonomous vehicles. Since LIDAR cameras have high scanning rates (usually > 1kHz), it makes them ideal for scanning the vicinity of a vehicle. In addition, since the camera module itself shoots a beam of light and senses the return, the system becomes ideal for aiding drivers in conditions that may otherwise be impossible to drive in, such as pitch darkness [3].

This is actually one of the main advantages of a LIDAR camera, over the currently more commonly used motion based cameras. Which are positioned in a 360° array with (usually) an with an extra IR camera placed in the front to increase the precision of readings [7]. Having image based cameras allows the vehicle to process colours from signs, stoplights, and emergency vehicles, which can be complemented by LIDAR cameras.

With LIDAR, the system which the camera feeds its data must be considered. The majority of camera systems utilize a real-time appearance based mapping (RTAB-Map) algorithm which implements simultaneous location and mapping (SLAM) [8]. In simple terms, the system scans the space (2D or 3D) of interest and creates a map of it. Generally, the data is stored in the form of software arrays with the distance reading and angle of scan. This is then used for the algorithms that the robot will run.

Considering the above systems, the finished prototype will integrate mechanisms that are currently available in the market and apply them to the scale of the project assigned. The LIDAR will be on a oscillating in a sinusoidal motion with PWM control. Having a separate microcontroller for reading and controlling the motor would be the ideal solution to reduce excess delay on the master arduino and to also improve response time for faster navigation.

2.1 History, Theory of Operation, and Contemporary Applications of LIDAR

LIDAR stands for Light Detection and Ranging. [1] In essence, LIDAR sends out a light pulse from an emitter and then receives the reflected pulse from the environment using a receiver in the same package. With the combination of some processing and computing, the distance of the object of which the light pulse reflected back to the LIDAR can be determined. By obtaining a large number of data points, a map of the surface can then be generated. A simple LIDAR unit typically consists of a laser and a receiver. Higher end units may have built in mirrors and motors, which allow the unit to rotate and tilt in multiple directions. These units may also have multiple lasers, which allow for a higher resolution and better data collection. [1]

Originally adopted by the US government for weather purposes, LIDAR was first used to map clouds. It was then brought to popularity during its in the Apollo 15 mission, where LIDAR was used to map the surface of the moon. Today, LIDAR is used in various applications and industries, from agriculture to transportation. Funded by military and defense research, LIDAR costs have dropped over the years. Today, the cost of a simple LIDAR unit is within reach of the general consumer.

In the autonomous vehicle industry, multiple LIDAR units are typically placed on top of the vehicle. They spin 360° and provide a very detailed map of the vehicle's surroundings. High performance LIDAR units are now capable of identifying a 15 cm sized object at a distance of 50 m while the vehicle is travelling 65 km/h [1]. Industry leaders such as Velodyne have been working with various partners such as Uber and Google to develop fully autonomous cars. Currently, Ontario allows autonomous cars to be tested on public roads.

2.2 Stepper Motor Control

Stepper motors allow for very precise angular displacements. This property is very useful in situations where motor angle needs to be held at any certain position since any ordinary DC motor cannot produce the stopping force necessary to hold a certain angle. Use of brakes to complement DC motors are usually not ideal and is wasteful.

Since stepper motors allow precise control of position, it is desirable to know the position of the motor relative to a specific starting point. Since the stepping of this kind of motor is done through exciting different coils in the motor, the position can be estimated by noting when the coils are actuated. This would necessitate that all the coils are individually addressable by the motor controller as well as the initial position information of the motor. The solution of this manner relies entirely on software manipulation for position and movement of the motor. [14]

A hardware solution to the above mechanism is also available, where an external source will provide clock to some excitation hardware built into the motor. This will then charge up the necessary phases to achieve the desired rotation. [5] The advantage of this approach is that it relieves the external source of the duty of supplying and controlling the motor. This could translate into CPU cycles being freed up in a processor, or better utilization of available RAM in the case of a microcontroller. However, this method would still require the initial position of the motor to be known.

Another method of obtaining position information would be to have a sensor built into the stepper motor. This type of system would be a combination of a stepper motor with an encoder. [14] With this approach, the initial position of the motor is no longer important since the absolute position information is captured and remembered by the encoder.

One unique way of controlling the position of a stepper motor is to only use line voltages and apply an algorithm on the obtained values to approximate the position of the motor. More specifically, by applying the Extended Kalman filter on the line voltages and currents, the system can be monitored and controlled. [4]

2.3 LIDAR Mapping

In order for the data from the LIDAR to be used, a map must be created. This map allows the vehicle to navigate through the a given course autonomously. Conventional LIDAR devices usually have very slow sample rates for high-speed applications [1]. For the application of a robot car, a slow sample rate is adequate.

The LIDAR device used for this project is the Garmin LIDAR-Lite v3HP. This is a single point LIDAR device which only emits and receives singular laser beams at a time. In contrast to LIDAR devices that use spinning mirrors or other dynamic data capture methods, the LIDAR-Lite must be manually rotated/oscillated if an area of data points is required [1]. Thus, a stepper motor is implemented to rotate the LIDAR unit.

The LIDAR device for this project can be considered a single point or a "0-D" LIDAR device. This means that it can only detect a single point at a time. For this project, a minimum of 1-D is required for the robot to navigate. Although 2-D LIDAR will give better accuracy, 1-D will be sufficient for planar navigation. This is a similar situation to the experiment that Reizer, et al. conducted regarding the 3-D visualization of maize plants. Their application involved a 2-D LIDAR device for "obtaining georeferenced 3-D point clouds of maize plants". [9]

There are opportunities where noise can interfere with the LIDAR signal. This can be easily accounted for by using a filter (either digital filter or physical filter). [9]. The 2-D LIDAR used for the 3D point cloud mapping of the maize plants had to be physically be altered to exhibit the specific details of the plants. This was done by having the LIDAR unit horizontally mounted while tilting it down around 35° [9]. This allows for a top down view of the plants. Thus the data gathered from the LIDAR unit is not just range information, but height information as well. For the application of the 1-D LIDAR in this project, range will be the only consideration. However, the orientation and tilt which the LIDAR will be mounted may be important in determining accurate data points. For example, if the LIDAR unit is tilted too low then it may pick up the ground as a potential barrier, in which is a false positive. Or if the unit is tilted too high, then it will not read anything at all, rendering the robot car blind. Therefore, the orientation of the LIDAR is very important when gathering distance data.

Assume the LIDAR unit is horizontally mounted, oscillated 180° at 1 Hz and sampled 1 kHz frequency. The acquired resolution should be around 0.36°. This would result in around a resolution of 0.0314 m at a range of 5 m, see Appendix F for detailed calculation. This resolution is much higher than necessary for the project.

Knowing the resolution, a map can now be created. Each data point gathered should be around 0.031 m apart. However this depends on the speed of the rotating LIDAR device as the velocity may not be constant. For 3-D LIDAR applications, 2-D Gaussian mixture maps can be used for localization [10]. A Gaussian mixture map contains both the reflectivity and height [10]. A similar application can be used for this project, but with a 1-D map. Utilizing a stepper motor, the output pulses to the motor should be constant. Each pulse can then be correlated to the current orientation of the LIDAR device. This data can be linked to the data that the LIDAR creates. Creating a 2-D array with these values, the distances of obstacles around the robot car can be determined. This map would be constantly refreshed as the robot car is moving. The LIDAR input will be processed to update motor outputs to effectively avoid obstacles.

3.0 PROTOTYPE DESIGN

The prototype design consists of the following general components:

- Garmin LIDAR-Lite v3HP
- Stepper motor
- Motor controller
- Reflective sensor (2 LEDs)
- 3D printed mounting plates
- Miscellaneous fasteners
- NiMH square battery pack

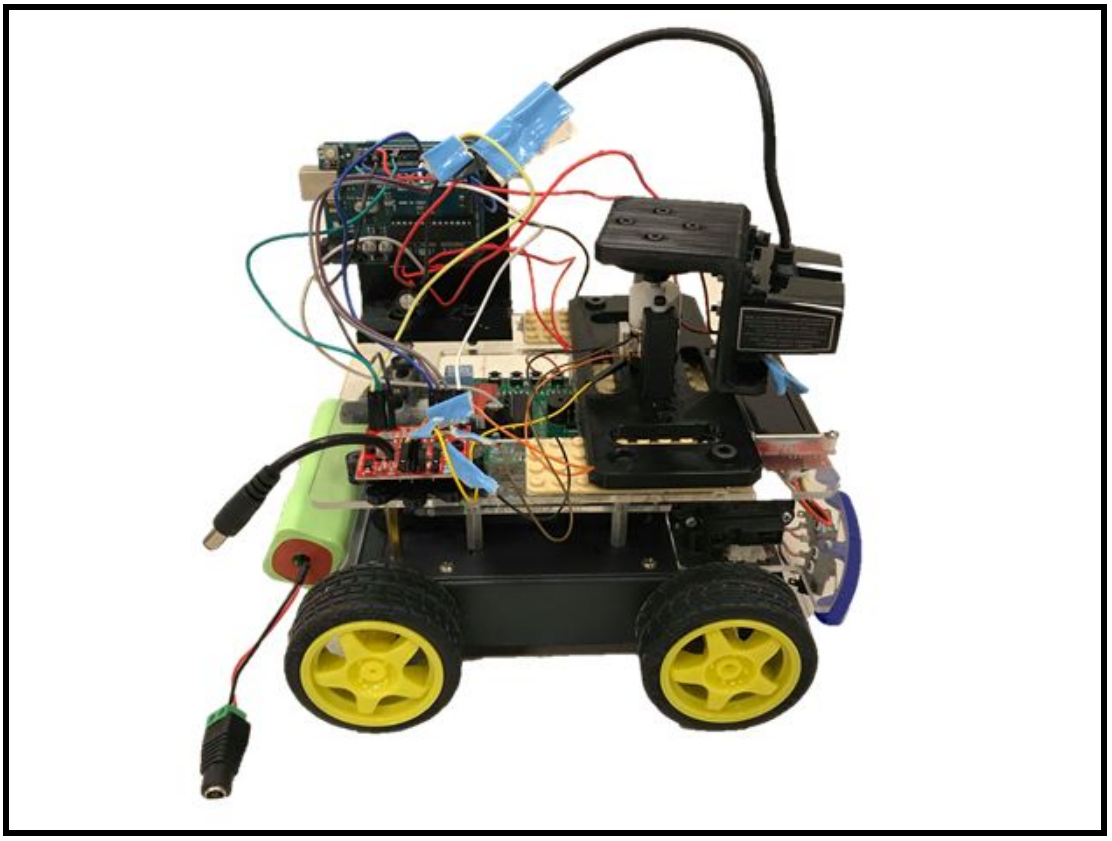

Figure 3.0.1: Completed prototype

Figure 3.0.1 shows the completed prototype. The following sections will highlight component selection in detail as well as the function of each parts within the prototype. Further details can be found in Appendix B.

3.1 Mechanical

Even though the LIDAR is rated for scanning speeds of greater than 1kHz, the most reliable data was obtained when the LIDAR was operated at a speed of 50 points per sweep. With a range of 5 to 40 cm and accuracy of ±2.5cm, the chosen LIDAR would fulfil the requirements for the project.

With the LIDAR in place, it was spun around by the stepper motor of choice. The size of the motor had to be inside an area of 370 mm by 65 mm. The height of the motor had to be as low as possible in order to keep the LIDAR scanning as close to the ground as possible.

Another part of the selection process was ensuring that the motor had adequate torque to rotate the LIDAR mechanism. This involved evaluating the weight of the entire oscillation assembly and the desired acceleration, which depended on the frequency of the LIDAR sweeps. These values were then compared to the torque given in the motor's data sheet. The detailed calculations are shown in Appendix F.

The NMB stepper motor chosen has a step angle of 0.198° per step. This would allow the motor to be stepped 9 times before the LIDAR is ready to take a reading. This would decrease our resolution from the theoretical max of 0.36° per reading to 1.782° per reading. However, this is still adequate for the application and is more ideal than operating the LIDAR setup at the theoretical limit since it tends to produce unreliable data when the LIDAR is operated at those speeds.

To mount the various components of the prototype to the mechbot, 3D printed parts were utilized. These parts were modelled in SolidWorks by obtaining relevant dimensions from data sheets and 3D printing the parts in the lab. The following section will highlight the physical implementation and component layout on the Mechbot.

3.1.1 Physical Implementation and Component Layout

For this project, five components were 3D printed in order to mount various parts to the Mechbot. This included brackets for the Arduino and motor controller, a base mount attaching the LIDAR mechanism to the Mechbot, a motor shaft adapter, as well as bracket for the LIDAR unit itself. These parts and associated layout are shown in the figures below.

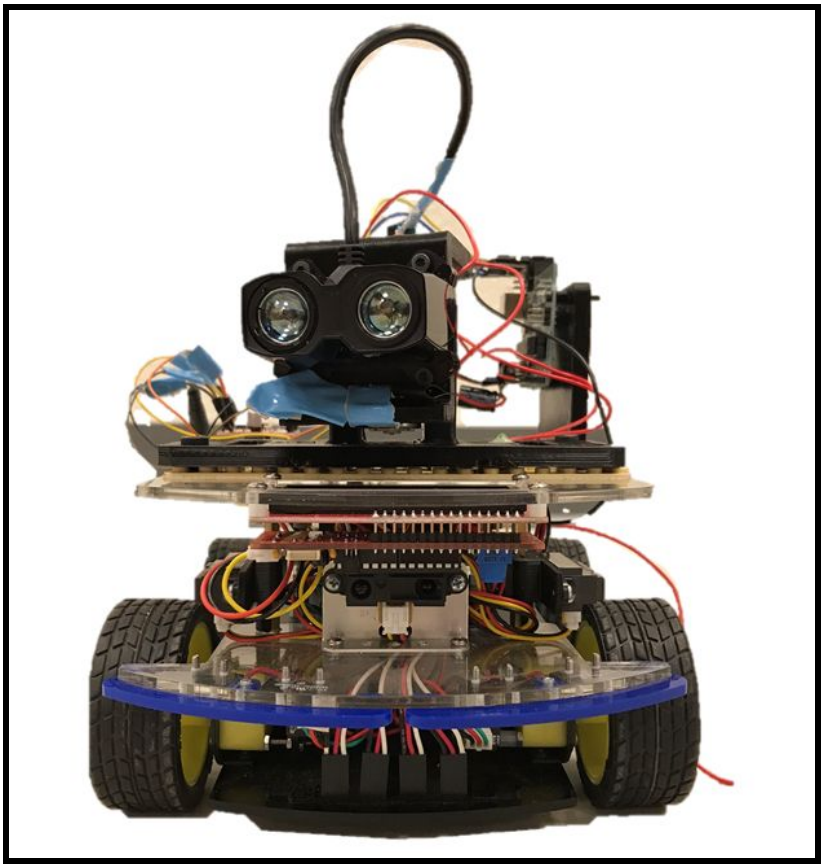

Figure 3.1.1: Mechbot with all required LIDAR components mounted on it

From Figure 3.1.1, it can be seen that the LIDAR unit was mounted as far forward and low as possible on the Mechbot, this was limited by the serial LCD monitor at the front of the plexiglass base. This lowered the sensing plane for the LIDAR unit, allowing it to detect objects in plane with the Mechbot. Doing so decreases the chances of not detecting a short obstacle.

Figure 3.1.2: NMB motor shaft adapter

Figure 3.1.2 shows the NMB motor shaft adapter, which was used to attach the LIDAR unit to the stepper motor. This part was designed to fit over the flat end of the shaft of the stepper motor, preventing slip at high speed operation.

Lastly, the Arduino and motor controller were placed near the rear of the Mechbot, to allow for convenient wiring. Additionally, the battery for the motor was affixed with velcro to the rear of the Mechbot for weight distribution and space limitations.

3.2 Electrical

Electrical components played a large role within this project. These components include the motor controller, Arduino microcontrollers, and reflective sensors. The wiring diagram in Appendix C shows the electrical components as well as the wiring to the mechanical components.

For this project, the Sparkfun Big Easy Driver motor controller was chosen. This component was selected due to the fact that it met the current and voltage requirements of the NMB stepper motor. Excellent documentation alongside low purchase price were also factors that were taken into consideration during the selection process.

LEDs served as a way for the LIDAR unit to calibrate itself at the beginning of the program. An LED was placed at the bottom of the LIDAR unit, and a reversed LED in the baseplate acting as a light sensor. This allowed the LIDAR unit to to calibrate itself by detecting changes in intensity when it has passed over the light sensor, which indicated the starting position for the scanning code. The LEDs selected were off-the-shelf units.

Arduino microcontrollers were used in this project due to the fact that the Mechbot already uses an Arduino for motor control. Moreover, the documentation for the LIDAR-Lite v3HP and Sparkfun Big Easy Driver were written and recommended the use of Arduino microcontrollers.

3.3 Software

The main program used for coding and control of the LIDAR system was the Arduino IDE. This is largely because the controller used was the Arduino R3. Processing was for the display of the readings because of its ease of use to display serial readings to a graph. A copy of the Arduino and processing code can be found in Appendix E.

The code for the controller for the LIDAR system had the following design points in mind; it had to be easily implemented regardless of the motor specifications, it had to be usable without encoders, it had to be capable of varying the speed of the motor which should not affect the accuracy of the camera, and it had to comply within the memory of the Arduino.

To satisfy the design points, constants were used so that only they would have to be changed if any adjustments were made. The constants chosen were `STEPANGLE`, `MAXSTEPS`, `DELAYAMOUNT`, `STEPSPERREADING`, and `INITANGLE`.

`MAXSTEPS` was used in the code to control the amount of data points collected in a sweep. `STEPANGLE` is the angle per step of the motor. `DELAYAMOUNT` is used to control the speed of the motor step. It should be noted that the step of the motor is controlled by bit-banging the output pin. `INITANGLE` is the initial angle that is desired for the sweep. With INITANGLE and `STEPANGLE` fixed, the following equation must be satisfied to determine `MAXSTEPS` and `STEPSPERREADING`:

$$MAXSTEPS * STEPSPERREADING * STEPANGLE = ABS(2 * INITANGLE)$$

`DELAYAMOUNT` was determined through trial and error such that the speed of the motor was maximized. The values of `MAXSTEPS` and `STEPSPERREADING` should be as close to integer numbers as possible.

The logic for the motor speed is determined in the LIDAR code. The reading is calculated as the max distance directly in front of Mechbot and based on a comparison to the constants `STOPDISTANCE` and `SLOWDISTANCE`, a character is written to Serial which will be read by the Mechbot. The Mechbot has a simple if-else statement and controls the motor speeds from the Serial. The flowchart for the logic of the program can be found in Appendix D.

4.0 IMPLEMENTATION AND INTERFACING

The LIDAR-Lite V3HP is a device that is easily read through a microcontroller. The LIDAR unit emits a laser and reads the corresponding reflected laser to obtain a result. This result is quantifiable as the distance that the laser has travelled. There are two main ways that the LIDAR unit can output this data: I2C and PWM. Being more familiar with I2C and having most of the digital pins occupied by the motor, it was the chosen way to interface the LIDAR to the microcontroller.

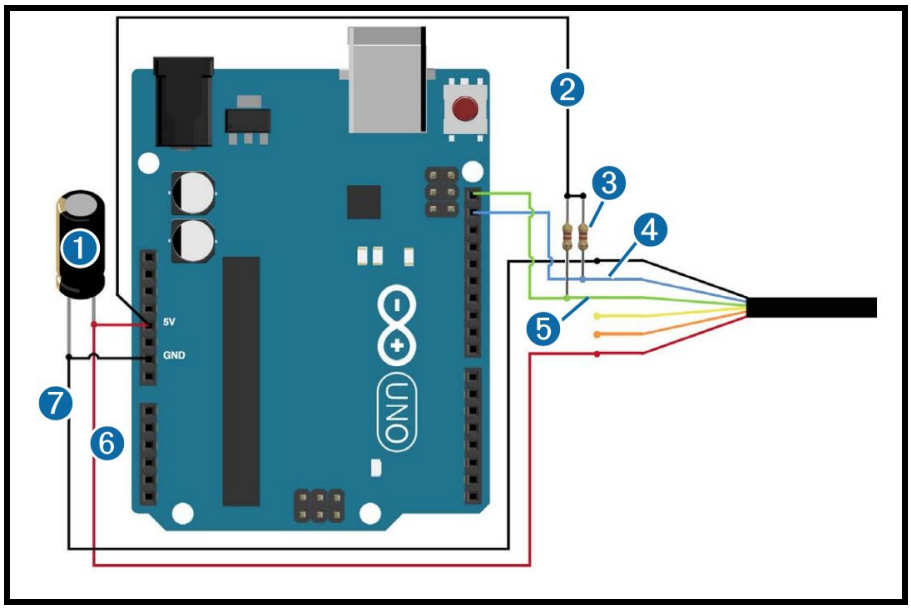

Figure 4.0.1: Standard I2C wiring for the LIDAR unit to the Arduino Uno [11]

As shown in Figure 4.0.1, the SDA and SCL on the LIDAR unit is connected to the SDA and SCL on the Arduino respectively. The SDA line transfers the data gathered by the LIDAR unit. The SCL is the clock line which is used to synchronize the transferring of data. The data is then read using the functions provided by Garmin in their "LIDARLite" and LIDARLite_v3HP" libraries.

As for creating the "map" used for controlling the Mechbot, a single array is used. This array is filled with the step angle (which increments whenever the motor increments) and its corresponding reading from the LIDAR unit. As the LIDAR sweeps back, the existing data for that particular motor angle is replaced with the new LIDAR reading. Depending on the required quantity of data, multiple points can be read by the LIDAR between the motor steps for higher resolution.

The data in the array can then be used for both creating an image of the map and/or controlling the robot car. For this project, all the gathered data is outputted from the Arduino through serial. This allows Processing to read the serial port and use the data to create a visual map. As for controlling the robot car, the LIDAR Arduino (master) communicates with the Mechbot Arduino (slave) via the RX and TX serial communication ports. See the following section for samples of the maps created using the LIDAR data.

5.0 TESTING AND RESULTS

In order to qualify the performance of the LIDAR unit, testing procedures were devised and improved upon throughout the length of the project. The initial testing procedure involved placing obstacles within the scanning area of the LIDAR unit and comparing the data obtained by plotting it into a graph in Excel. As seen in Figures 5.0.1 and 5.0.2, the graph generated by Excel correlates to the obstacles placed in front of the LIDAR unit. Although the map was quite rough, this visualization of the data acted as a proof of concept. However, this method of monitoring the LIDAR output did not facilitate live mapping.

Figure 5.0.1: Setup for initial testing

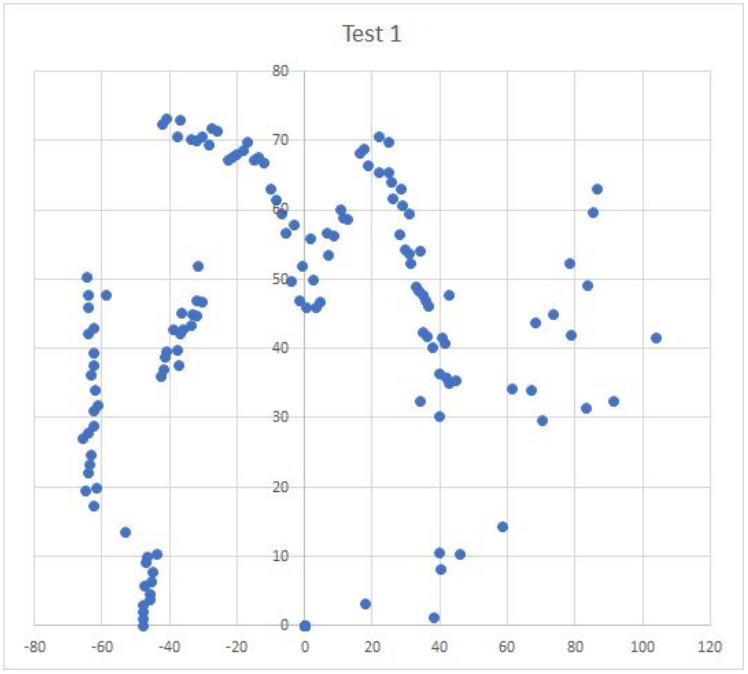

Figure 5.0.2: Map of setup above generated by Excel

Building on the issues of the first test setup, the second test setup aimed at improving resolution while also providing live mapping. Processing was utilized to provide live mapping of the LIDAR. The test setup and the resulting output from Processing can be seen below:

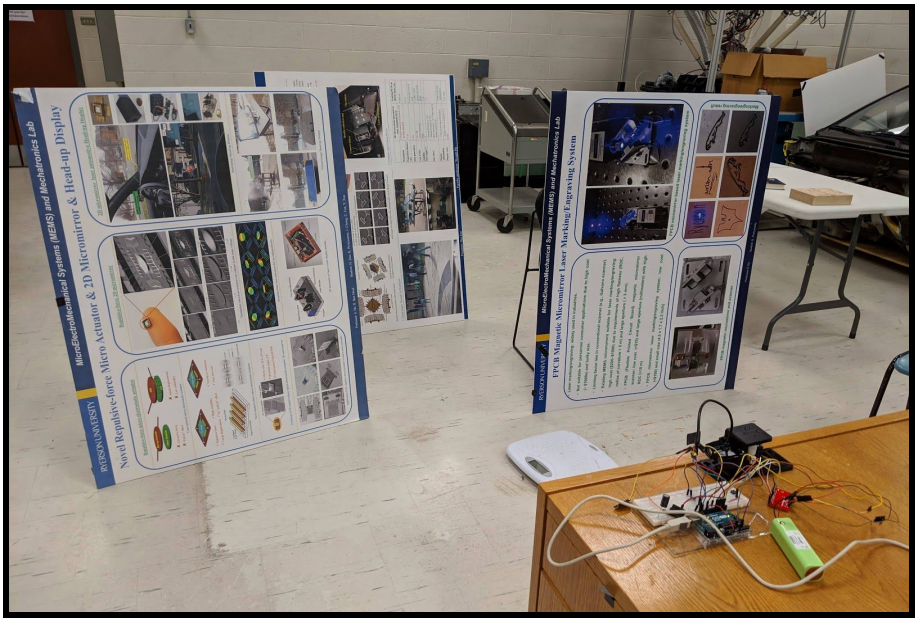

Figure 5.0.3: Setup for initial static testing

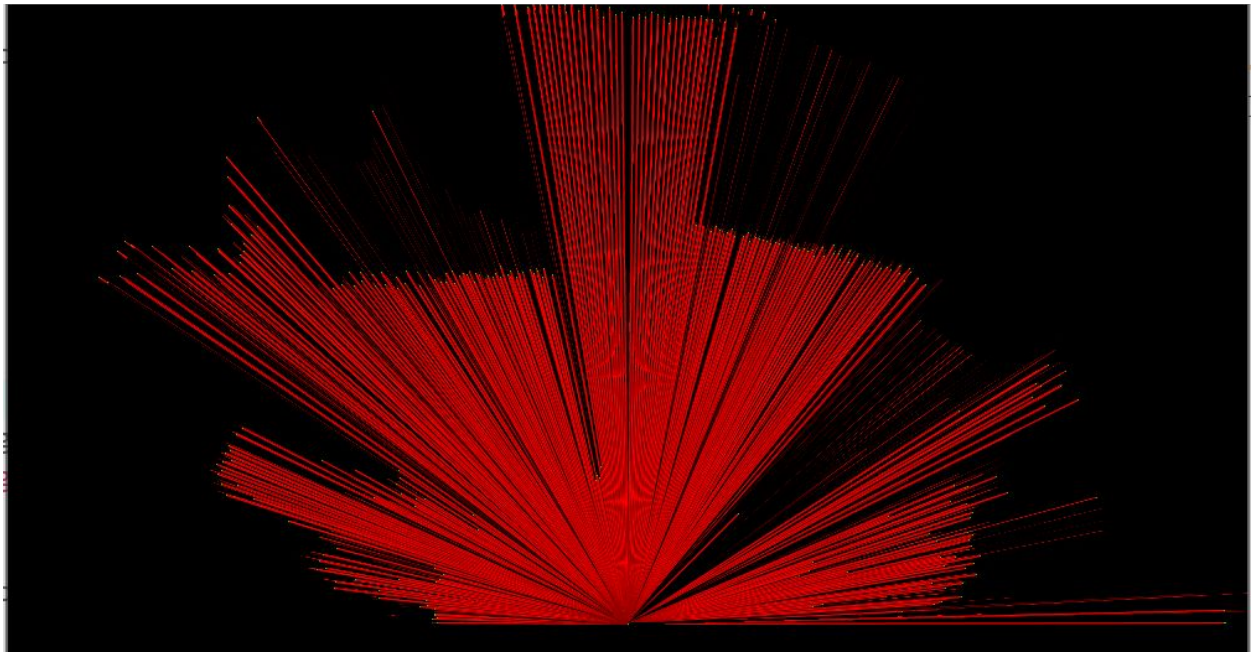

Figure 5.0.4: Real-time map display of the initial setup

Figures 5.0.3 and 5.0.4 show the room layout and the output in Processing. Compared to Figure 5.0.2, it can be seen that the map generated by Processing is much finer than the map generated by Excel. Further improvements were made to speed up the data collection and improve the accuracy of the LIDAR setup. The results can be seen in the figures below, which depicts a much more accurate graph than the one displayed in Figure 5.0.4.

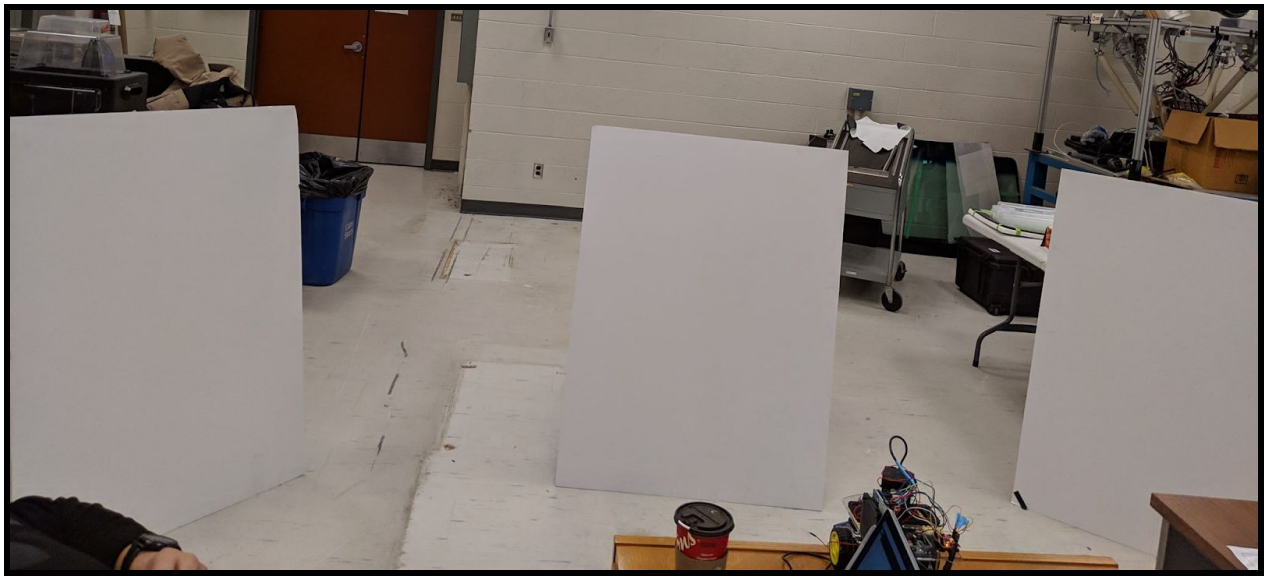

Figure 5.0.5: Revamped test setup

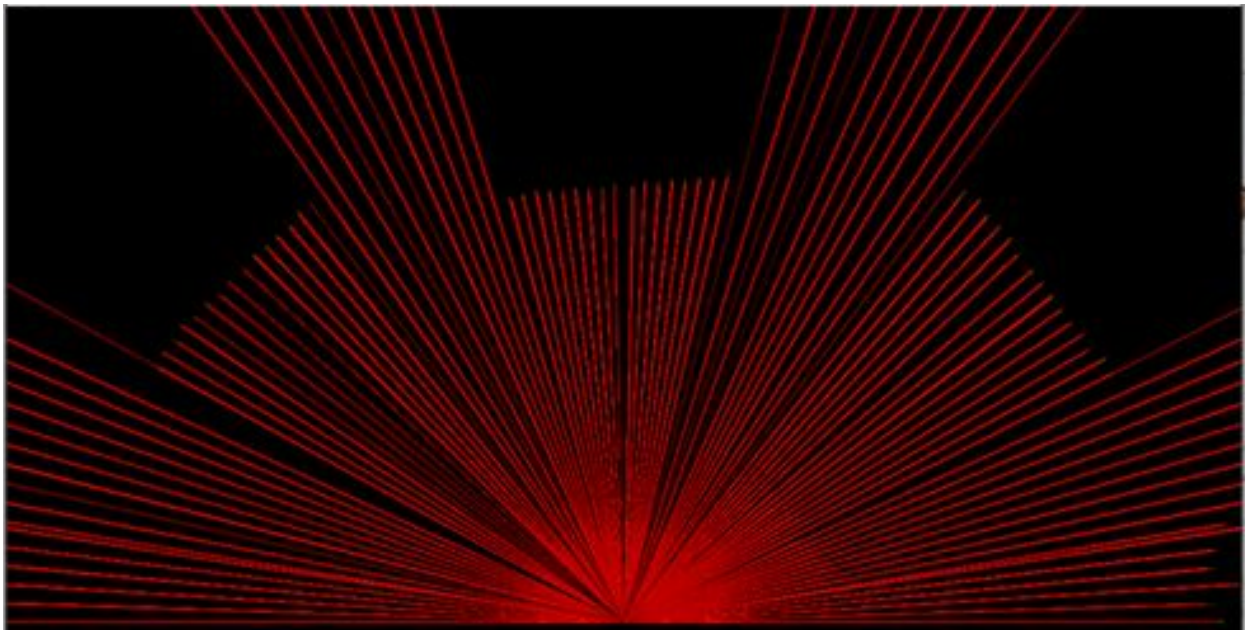

Figure 5.0.6: Real-time map of revamped test setup

After obtaining positive test results as shown in the figures above, the next step was to test the Mechbot with real life obstacles. Programmed to slow down and stop when the LIDAR detects an obstacle within range of the Mechbot, the following figures exhibit the Mechbot reacting to an obstacle while it is in motion.

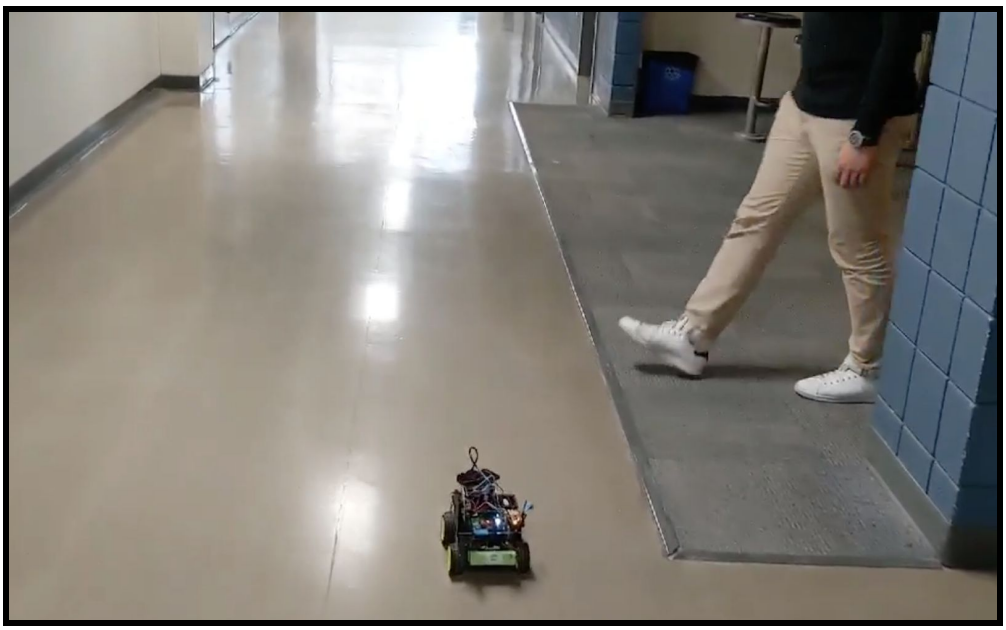

Figure 5.0.7: Mechbot detecting a pedestrian crossing the hallway

Figure 5.0.8: Mechbot stopping for a pedestrian crossing the hallway

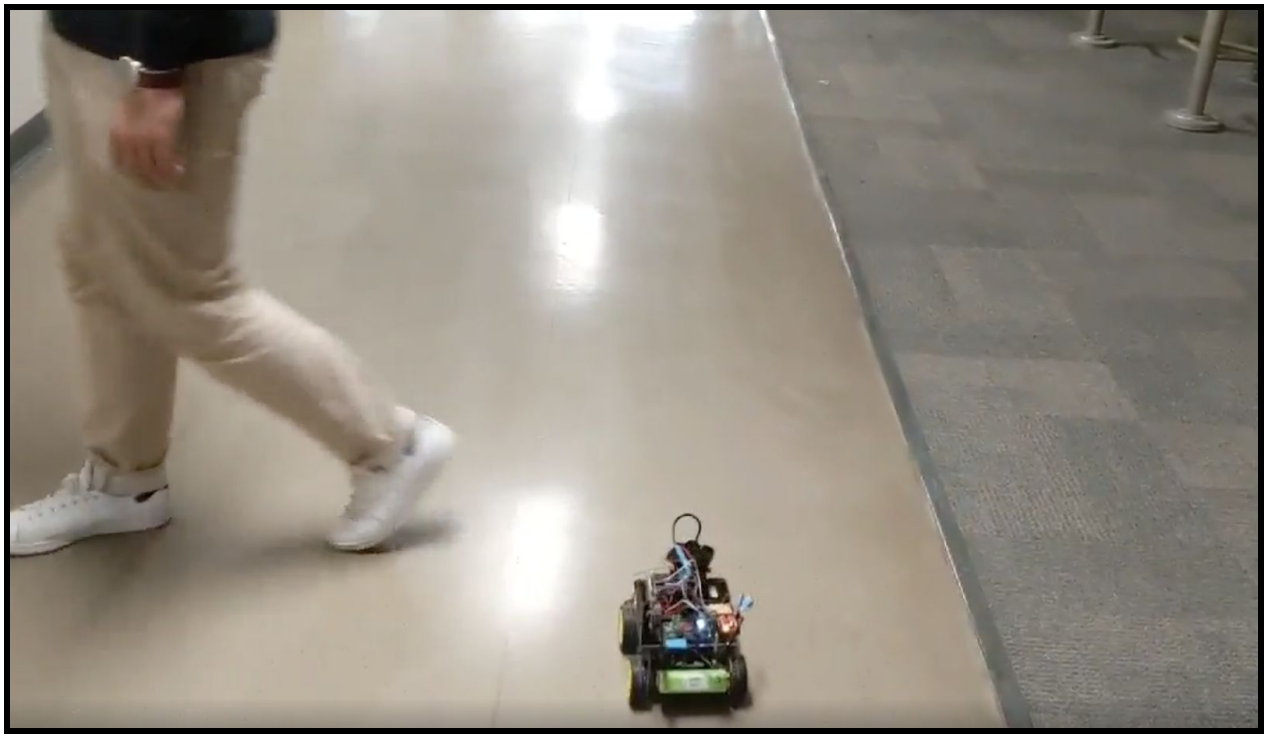

Figure 5.0.9: Mechbot resuming motion after obstacle is out of range

Further testing with more obstacles of varying sizes at different distances from the Mechbot was be performed in order to ensure that the Mechbot functions well in different scenarios. Code parameters were also optimized for the best performance.

6.0 EVALUATION

Through trial and error, the amount of data points gathered per sweep was decreased to 50 data points. This was because it was not necessary to have any more data points for this application as the distances required to be read are at most 5 metres away. Another reason why the number of LIDAR readings were decreased was because the motor was not able to run fast enough. This was due to the limitations of the stepper motor utilized. Updating the LIDARLite library to the most recent release helped reduce the overhead time required by the LIDAR readings.

Referring to the previous section, it can be determined by the 2-D maps where exactly each obstacle stands. There are two Arduino codes that run the entire system. One for the LIDAR and motor and another for the Mechbot. The first Arduino processes the data gathered and within each sweep and calculates the lowest distance. This is then compared to a set range of distance values. Depending on the outcome of the comparison, the robot will go forward at full speed, slow down, or stop. Images of these tests can be seen in the previous section. They were successful as long as the obstacle or moving object was not moving towards the robot at a fast rate.

The LIDAR was originally planned to scan a 90° arc in front of the Mechbot. This was later decreased to around 45°. Due to this, objects on the side, and/or behind the Mechbot have no influence on the Mechbot. A 360° rotating LIDAR would further improve this system in terms of practicality, flexibility, and utility.

7.0 CONCLUSIONS AND RECOMMENDATIONS

Overall, the goal of designing and creating a LIDAR navigated robot car was successful, with quantifiable results shown in the results section above. Future students in the MEC733 course will be able to use this setup and utilize LIDAR in their projects.

Throughout this project, the group was constantly refining the design in order to increase sweep speed, which would increase the accuracy and reliability of the LIDAR system. This was one of the main challenges that the group faced. There are many ways to increase the sweep speed, but for this project, the group focused mainly on software changes due to time constraints from ordering and upgrading new hardware. For future recommendations, a motor with a larger step size can be used to increase the sweep speed. Using a DC motor could also be explored, but this would require an encoder to be added. At the moment, the stepper motor is rated to run at a speed up to 1 kHz. However in practice, only about 800 Hz could be achieved. If the decision was to continue using stepper motors, the optimal solution would be to use a stepper motor with a higher frequency. That would allow more headroom in case the motor performs slightly worse than the theoretical limits proposed by the manufacturers.

In terms of navigation, the Mechbot was successful in accelerating and stopping in a straight line. Further refinement would allow the Mechbot to navigate around corners and obstacles, such as the maze in the MEC733 course. To increase the Mechbot's mapping and navigation accuracy, a 360° sweep angle could be implemented, through the use of a slip ring and a faster motor. This would allow simultaneous location and mapping (SLAM) to be utilized. In a SLAM setup, the robot knows its live position within the map, which allows it to proactively navigate accordingly to the obstacles it knows are ahead, as opposed to operating in a reactive manner when something appears in front of the robot, such as the current setup.

8.0 REFERENCES

[1] B. Schwarz, "Lidar: Mapping the world in 3D," *Nature Photonics,* vol. 4, *(7),* pp. 429-430, 2010. Available: https://search-proquest-com.ezproxy.lib.ryerson.ca/docview/873168000

[2] M. Pavelka and Vaclav Jirovsky, "Lidar based object detection near vehicle," 2017 *Smart Cities Sympoisum Prague (SCSP),* 2017. Available: https://ieeexplore-ieee-org.ezproxy.lib.ryerson.ca/document/7973852

[3] H.J. Kamps, "Ford tests Lidar-equipped car in pitch darkness," *TechCrunch,* 2016. Available: https://search-proquest-com.ezproxy.lib.ryerson.ca/docview/1780008785

[4] M. Bendjedia *et al*, "Position Control of a Sensorless Stepper Motor," *IEEE Transactions on Power Electronics,* vol. 27, *(2)*, pp. 578-587. 2012. Available: https://journals-scholarsportal-info.ezproxy.lib.ryerson.ca/pdf/08858993/v27i0002/578_pcoassm.xml

[5] P. Acarnley, "Stepping Motors: a guide to theory and practice," *Stepping Motors: a guide to theory and practice*, 2002. Available: https://app.knovel.com/web/toc.v/cid:kpSMAGTPEF/viewerType:toc//root_slug:stepping-motors-guide/url_slug:stepping-motors?=undefined&issue_id=kpSMAGTPEF

[6] G. Carbone and F. Gomez-Bravo, "Motion and Operation Planning of Robotic Systems," *Motion and Operation Planning of Robotic Systems Mechanisms and Machine Science*, 2015.
H. Ergezer and K. Leblebicioglu, "Visual Detection and Tracking of Moving Objects," *2007 IEEE 15th Signal Processing and Communications Applications*, 2007. Available: https://www.researchgate.net/publication/4273567_Visual_Detection_and_Tracking_of_Moving_Objects

[7] C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys, "3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection," *Image and Vision Computing*, vol. 68, pp. 14–27, 2017. Available: https://journals-scholarsportal-info.ezproxy.lib.ryerson.ca/details/02628856/v68icomplete/14_3vpfsccmlaod.xml

[8] M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2018. Available: https://onlinelibrary-wiley-com.ezproxy.lib.ryerson.ca/doi/full/10.1002/rob.21831

[9] D. Reiser, M. Vázquez-Arellano, D. S. Paraforos, M. Garrido-Izard, and H. W. Griepentrog, “Iterative individual plant clustering in maize with assembled 2D LIDAR data,” *Computers in Industry*, vol. 99, pp. 42–52, 2018. Available: https://www-sciencedirect-com.ezproxy.lib.ryerson.ca/science/article/pii/S0166361517304748

[10] R. W. Wolcott and R. M. Eustice, “Fast LIDAR localization using multiresolution Gaussian mixture maps,” *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015. Available: https://journals-sagepub-com.ezproxy.lib.ryerson.ca/doi/full/10.1177/0278364917696568?utm_source=summon&utm_medium=discovery-provider

[11] "LIDAR-Lite v3HP", *Garmin*, 2019. [Online]. Available: https://buy.garmin.com/en-CA/CA/p/578152#specs. [Accessed: 12- Feb- 2019].

[12] "PG20L-D20-HHC0 NMB Technologies Corporation", *Digikey.ca*, 2019. [Online]. Available: https://www.digikey.ca/product-detail/en/nmb-technologies-corporation/PG20L-D20-HHC0/P14334-ND/2417058. [Accessed: 12- Feb- 2019].

[13]"Big Easy Driver - ROB-12859 - SparkFun Electronics", *Sparkfun*, 2019. [Online]. Available: https://www.sparkfun.com/products/12859. [Accessed: 13- Feb- 2019].

[14] I. Virgala et al, "Stepper Motor Control by ATMEL AVR Microcontroller," Applied Mechanics and Materials, vol. 816, pp. 321-326, 2015. Available: http://ezproxy.lib.ryerson.ca/login?url=https://search-proquest-com.ezproxy.lib.ryerson.ca/docview/1903481789?accountid=13631. DOI: http://dx.doi.org.ezproxy.lib.ryerson.ca/10.4028/www.scientific.net/AMM.816.321.

[15] “1600 mAH 9.6 V NiMH Square Battery Pack ELCO”, *RobotShop,* 2019. [Online]. Available: https://www.robotshop.com/ca/en/hitec-54114--battery.html?fbclid=IwAR1T-xbcEN_nBFsVQ0vVEI1sYNPTHvKFl8uVSpedvZZ5CjyJsIYsz_FiuK0. [Accessed: 30- Mar- 2019].

[16] “Arduino Power, Current, and Voltage Limitations”, *Robotics*, 2019. [Online]. Available: http://robotics.lib-ieronimoub.gr/?p=715. [Accessed: 30- Mar- 2019].

[17] Vishay, “TCRT5000, TCRT5000L product information,” *Vishay*. [Online]. Available: https://www.vishay.com/optical-sensors/list/product-83760/. [Accessed: 03-Apr-2019].

9.0 APPENDICES

9.1 Appendix A: Bill of Materials and Assembly Drawing

The prototype design consists of the LIDAR oscillation mechanism sitting on top of the Mechbot. This is done by various fasteners and 3-D printed parts. The LIDAR unit is fixed onto an L-bracket which connects to a motor adapter part. This allows the stepper motor to directly control the oscillation motion of the LIDAR. Additional 3-D printed parts were used to secure the Arduino Uno and motor controller onto the Mechbot. See Figure 9.1.1 below for the assembly drawing and bill of materials.

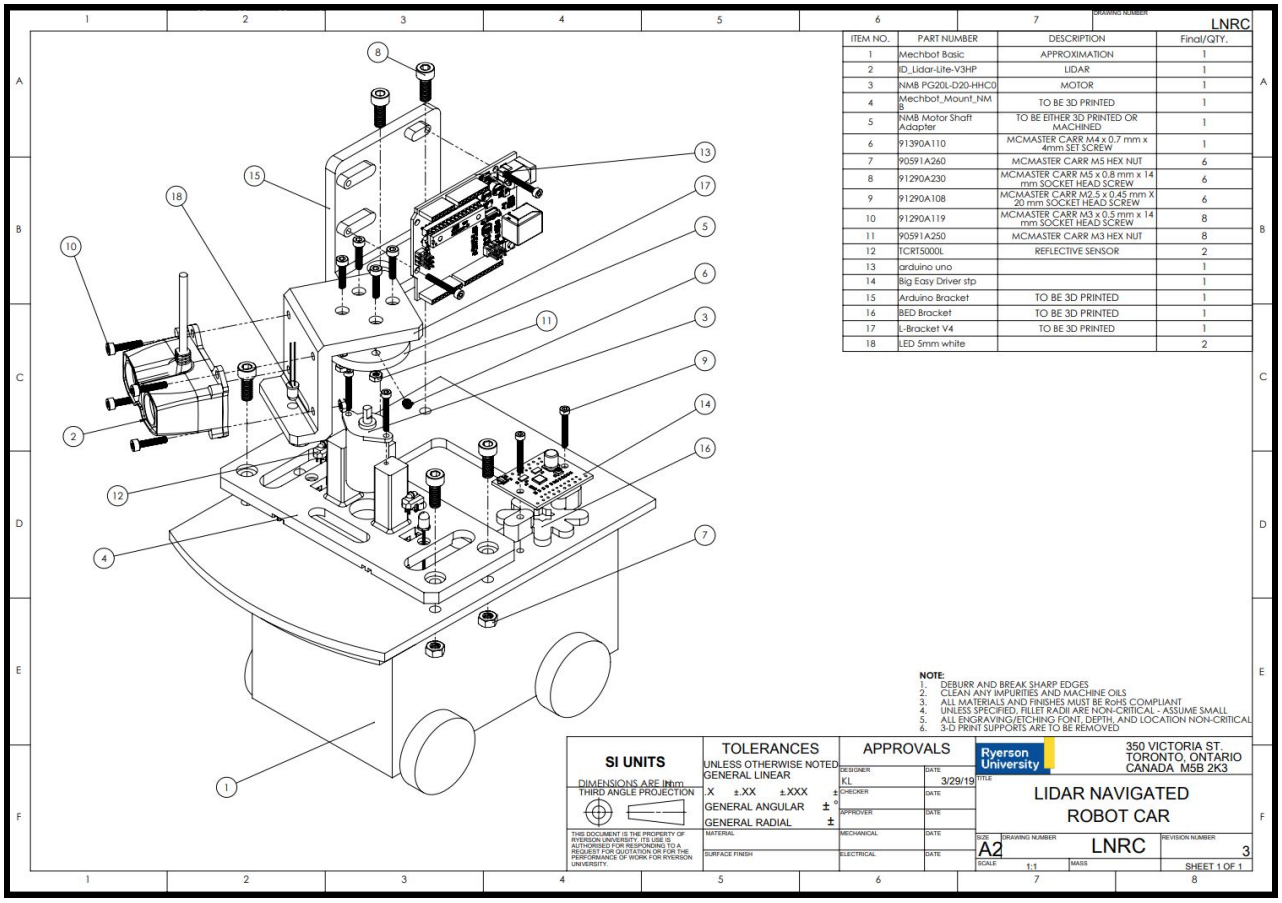

Figure 9.1.1: Assembly Drawing of LIDAR Oscillation Mechanism

9.2 Appendix B: Component Specifications

Many of the components that were used are of-the-shelf items. Some of the critical components are shown and listed below.

LIDAR Device:

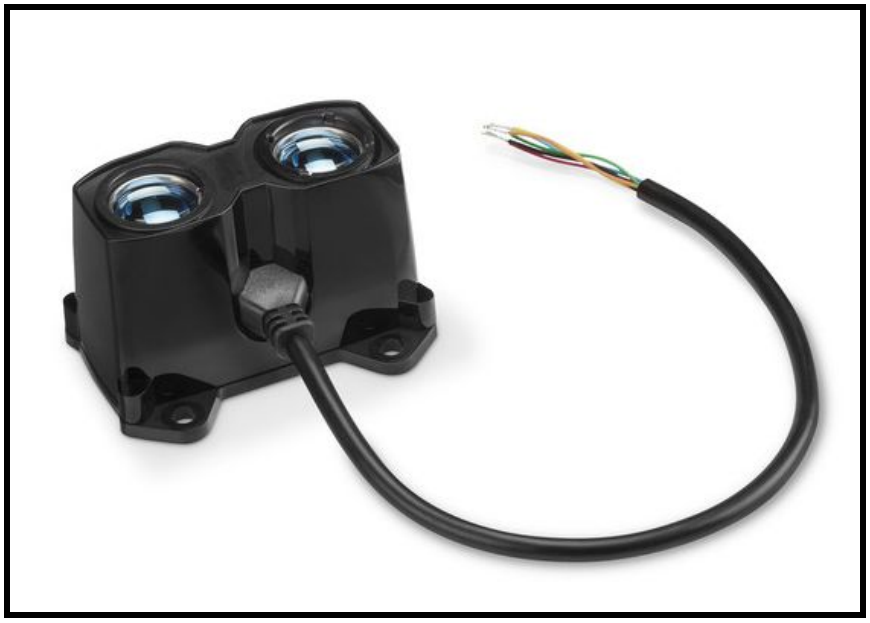

Figure 9.2.1: Garmin LIDAR-Lite v3HP [11]

Table 9.2.1: LIDAR-Lite v3HP Specifications [11]

Name	LIDAR-Lite v3HP
Range (m)	0.05 - 40
Resolution (cm)	1
Scanning Frequency (Hz)	>1000
Max. Voltage Input (VDC)	4.5 - 5.5
Min. Current Required (mA)	65
Beam Divergence (mRad)	8
Interface	I2C or PWM

The LIDAR-Lite V3HP was chosen because of the requirement of using a single point LIDAR in this project. This allows the concept of using a single point LIDAR device to map an environment and navigating the robot car through obstacles. The reason for the high power over the normal LIDAR-Lite V3 was due to the increasing scanning capabilities for the high-powered version.

Motor:

Figure 9.2.2: NMB Motor PG20L-D20-HHC0 [12]

Table 9.2.2: NMB Motor Specifications [12]

Name	NMB Motor PG20L-D20-HHC0
Motor Type	Stepper (permanent magnet)
Polarity	Bipolar constant current
Phase	2 Phase
Max. Voltage Input (V)	10
Min. Current Required per phase(mA)	350
Step Angle (°)	0.198
Velocity Range (pps)	350 - 1100

Torque Range (Nm)	0.4 - 1.5

This motor was chosen due to its compact size and low voltage requirements. Typically, stepper motors are powered by 24 V. That would be impractical for this project because the power supplied must be a portable supply and thus cannot be connected to a wall outlet or be bulky. Additionally, the flat shaft provides a mounting point for objects to be spun by the motor.

Motor Controller Board:

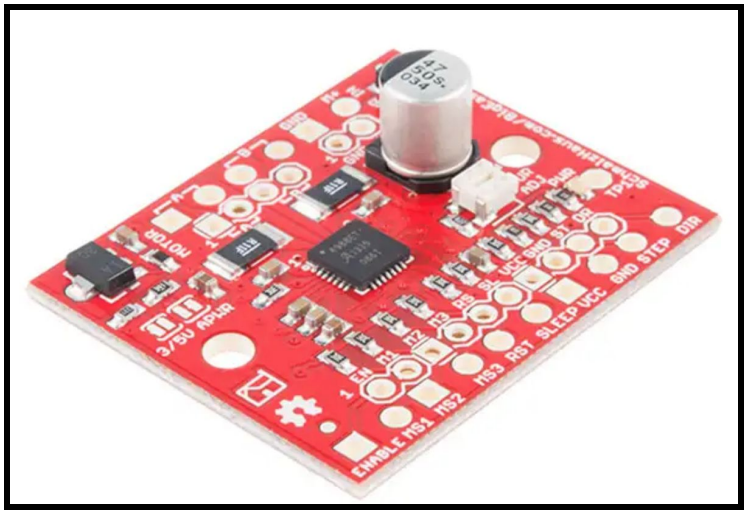

Figure 9.2.3: Sparkfun Big Easy Driver [13]

Table 9.2.3: Sparkfun Big Easy Driver Specifications [13]

Name	Big Easy Driver
Motor Driver Type	Stepper
Motor Supply Voltage Range (VDC)	8 - 30
Max. Current Output per phase(mA)	2000
Polarity Compatibility	Bipolar
Excitation Mode	1 - 2 Phase
Microstep Resolution	1/16 of a step

The motor controller was selected by matching the specifications of the stepper motor shown in Figure 9.2.2. From the specification Table 9.2.3, it will be able to run the chosen stepper motor.

Power Source:

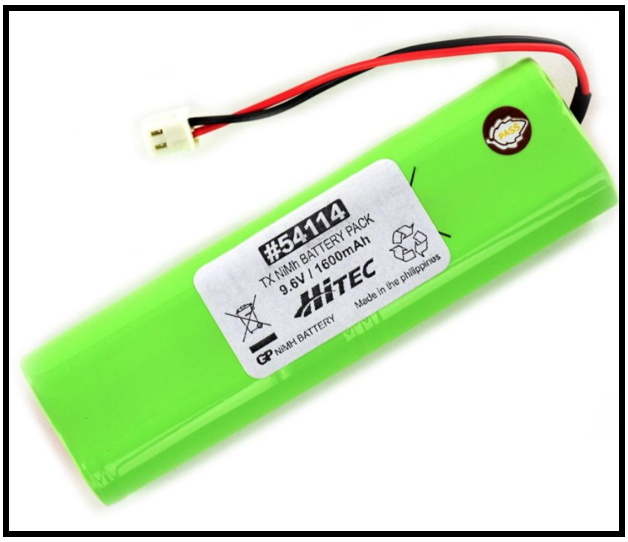

Figure 9.2.4: 9.6 V NiMH Battery Pack [15]

This project requires a mobile power source for the Arduino Uno and stepper motor. The power source chosen was the battery pack shown in Figure 9.2.4. It supplies a DC voltage of 9.6 V, current of 1.6 A and a capacity of 1600 mAH. The stepper motor requires a minimum of 700 mA (# of phases x current per phase) and the Arduino Uno requires a maximum of 700 mA as well (500 mA for the 5V pin and 200 mA max for the digital I/O pins). [16] Thus, a total of 1.4A required can be assumed. Using this total current value and the battery capacity, the battery should last for around $\frac{1600\ mAH}{1400\ mA} \approx 1.14\ hours$, which is more than sufficient for this project.

The other components (LIDAR Lite V3HP, LEDs, etc) all run off the Arduino Uno. From the specifications stated in the tables above, the Arduino Uno will easily be able to run those components.

Photodetector:

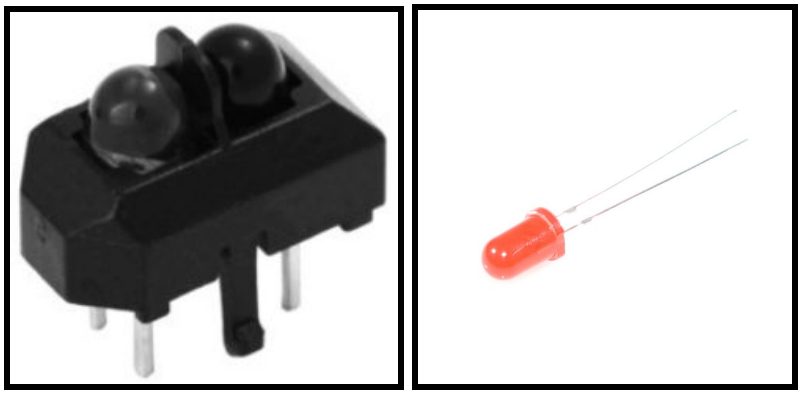

Figure 9.2.5: Vishay TCRT5000L [17] and the LED replacement

A photodetector is required to provide feedback as the control of the motor is an open loop setup. A photodetector would be extremely useful to help determine the limits of the motor motion and assist in calibration during initial power on phase as well as reboot situation.

The photodetector by Vishay, the TCRT5000L can be used to detect changes in colour. If this photodetector is not adequate, then an LED wired up in reverse can be used to detect light.

From testing, the TCRT5000L turned out to be not sensitive enough and would require some sort of amplification to utilize it. Thus, a diode (LED wired in reverse) and LED combination was used. A green 5mm LED would be attached to the LIDAR holding bracket to emit a green light through a hole of around 3.5 mm. This would be read by the 3.5 mm diode attached to the base of the Mechbot, allowing the motor to be calibrated every instance of usage.

9.3 Appendix C: Wiring Diagrams

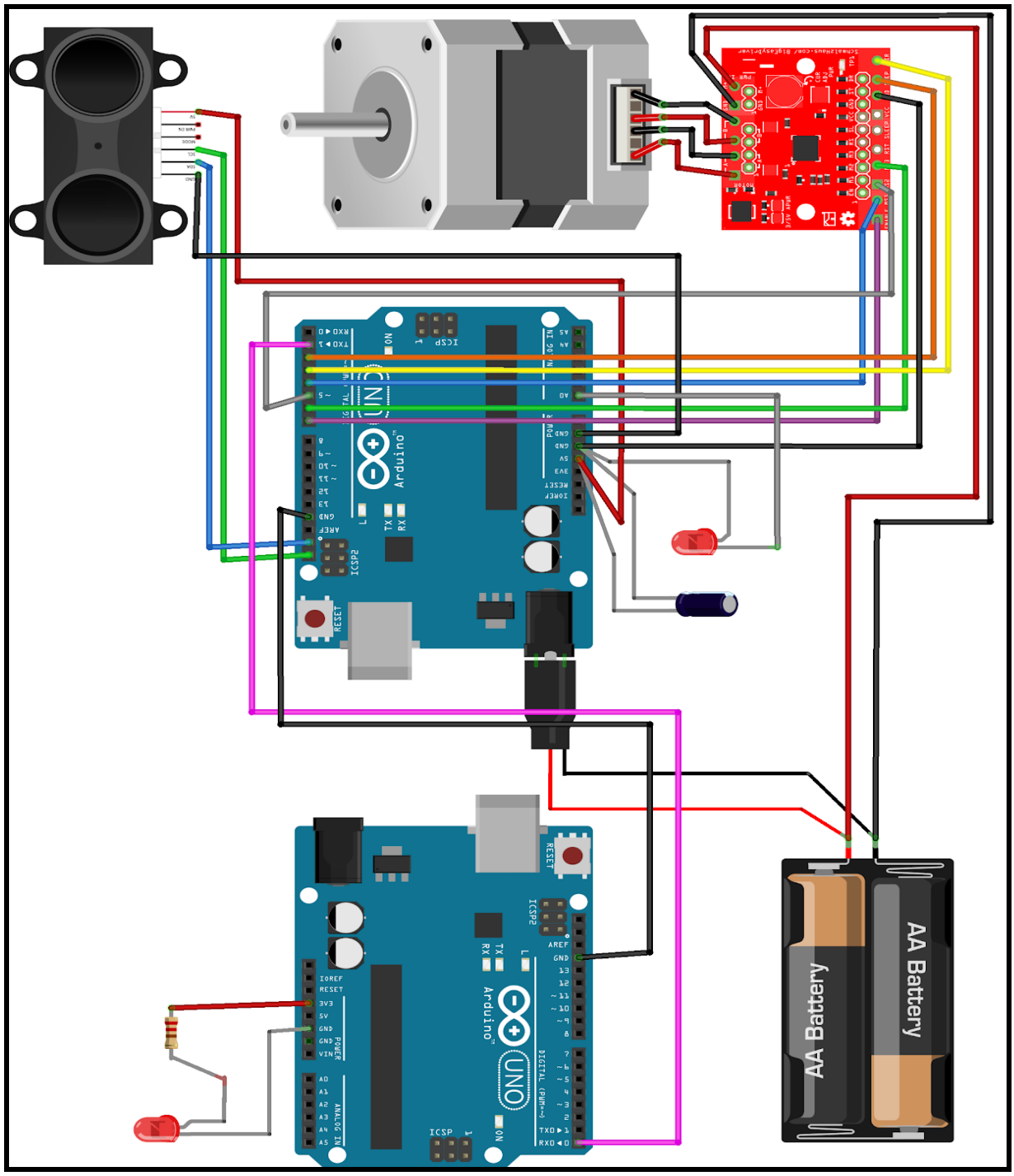

Figure 9.3.1: Prototype Wiring Diagram

9.4 Appendix D: Program Flowchart and Control System

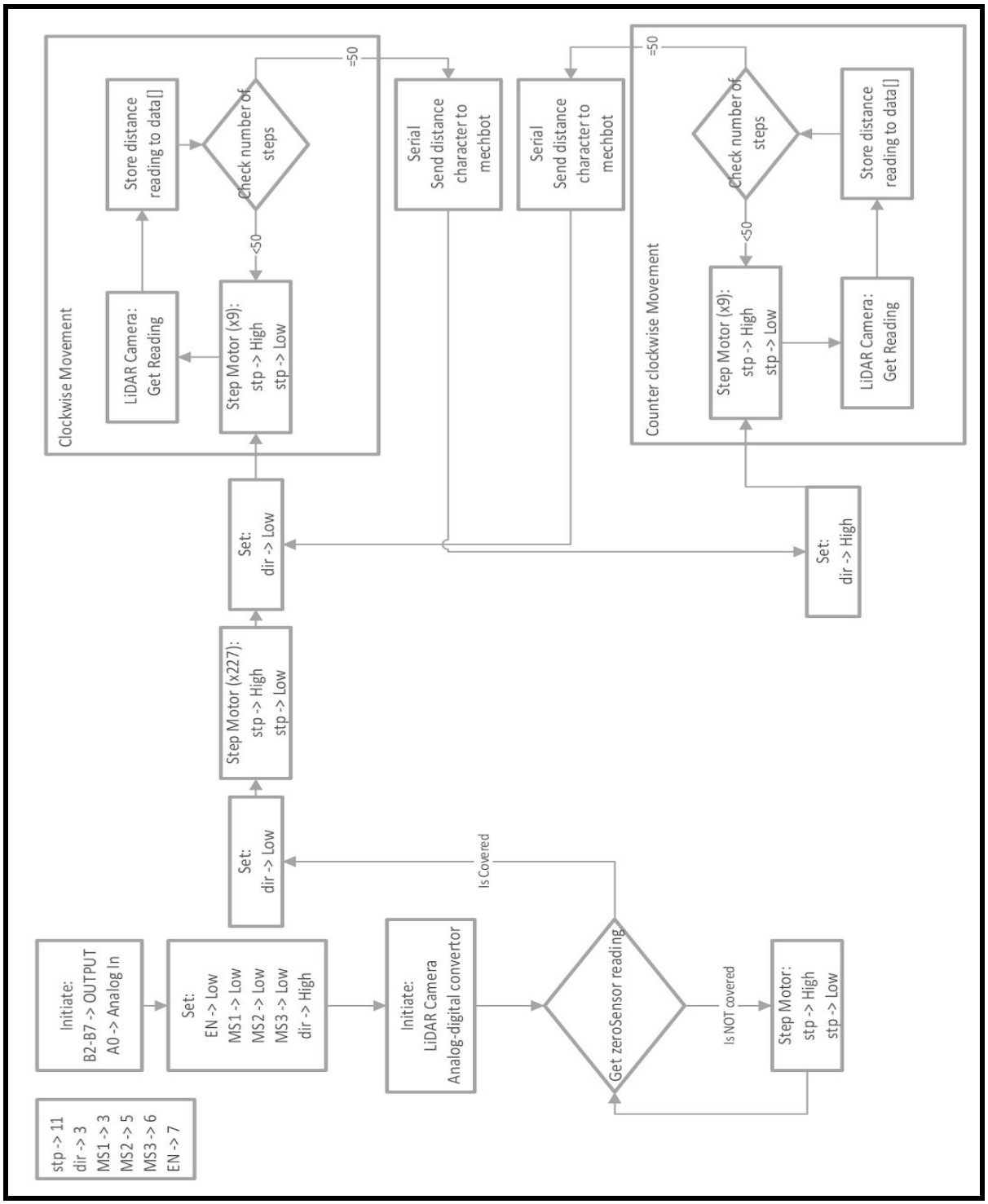

Figure 9.4.1: Program Flowchart Logic

9.5 Appendix E: Program Listing

LiDAR Arduino Code

```
/*This code will be the set up for the slave Arduino running the
  LiDAR and motor controller*/
#include <mechbotShield.h>
#include <Wire.h>
#include <LIDARLite.h>
#include <LIDARLite_v3HP.h>

//General Constants
//NOTE: All constants need to be adjusted in the final design
#define ENCODERMIN 390       //This value is when the light sensor is covered
#define STEPANGLE 0.198
#define MAXSTEPS 50                 //MAXSTEPS * STEPSPERREADING * STEPANGLE =
ABS(2*INITANGLE)
#define DELAYAMOUNT 800
#define STEPSPERREADING 9
#define INITANGLE -45
#define STOPDISTANCE 40
#define SLOWDISTANCE 70


//Port definitions for motor control
#define stp 11
#define dir 3
#define MS1 4
#define MS2 5
#define MS3 6
#define EN 7
#define BTN 13

LIDARLite_v3HP lidarLite;
int data[MAXSTEPS];
int8_t motorDirection = 1; //-1: dir = LOW, 1: dir = HIGH
float minDistance;

void setup() {
  int zeroSensor;

  //Motor
  pinMode(stp, OUTPUT);   //HIGH will make the motor step once
  pinMode(dir, OUTPUT);   //HIGH -> "reverse" or CCW, LOW -> "forward" or CW
  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(MS3, OUTPUT);
  pinMode(EN, OUTPUT);

  digitalWrite(EN, LOW);  //Pull low to set FETs active to allow for motor control. Otherwise
HIGH to lock
  digitalWrite(MS1, LOW);
  digitalWrite(MS2, LOW);
```

```
  digitalWrite(MS3, LOW);

  initADC();

  //LiDAR Camera
  Wire.begin();
  if (true) {
#if ARDUINO >= 157
    Wire.setClock(400000UL);
#else
    TWBR = ((F_CPU / 400000UL) - 16) / 2;
#endif
  }
  lidarLite.configure(0);

  Serial.begin(19200);
  //Setting up Photoresistors for reading
  digitalWrite(dir, HIGH);

  do {
    //Run motor towards zero
    digitalWrite(stp, LOW);
    delay(5);
    digitalWrite(stp, HIGH);
    delay(5);
    zeroSensor = analog(0);
  } while (zeroSensor > ENCODERMIN);

  //Code to set motor to start at 45 deg.
  digitalWrite(dir, LOW);
  motorDirection = -1;
  int offset = 0;
  do {
    offset++;
    //Run motor towards a 45 degree start
    digitalWrite(stp, LOW);
    delay(5);
    digitalWrite(stp, HIGH);
    delay(5);
  } while (offset < 227);
}

void loop() {
  int lidarReading;
  float currentDist;

  minDistance = 1000;

  for (int i = 0; i < MAXSTEPS; i++) {
    //Step Motor
    for (int j = 0; j < STEPSPERREADING; j++) {
      digitalWrite(stp, HIGH);
      delayMicroseconds(DELAYAMOUNT);
      digitalWrite(stp, LOW);
```

```
      delayMicroseconds(DELAYAMOUNT);
    }

    //LiDAR Reading
    lidarLite.takeRange();
    lidarReading = lidarLite.readDistance();

    //Storing into array
    data[i] = lidarReading;

    //Calculating max distance in the x direction (forward on mechbot)
    currentDist = abs(data[i] * cos((INITANGLE * motorDirection + STEPSPERREADING *
motorDirection * i * STEPANGLE) * 3.14159 / (180.0)));

    //Compare to see if dist is closer than previous
    if ((currentDist < minDistance) && (data[i] > 15)) {  //Check for the closest distance and
if it is above the optimal reading number
      if ((data[i] - data[i - 1]) < 50) {          //Check to see if a random reading is
present
        minDistance = currentDist;
      }
    }
  }
  //Printing to Serial port
  if (minDistance < STOPDISTANCE) {
    Serial.write('S');
  }
  else if (minDistance < SLOWDISTANCE) {
    Serial.write('F');
  }
  else {
    Serial.write('G');
  }

  //Switching motor direction
  motorDirection = motorDirection * -1;
  switch (motorDirection) {
    case 1:
      digitalWrite(dir, HIGH);
      delay(1);
      break;
    case-1:
      digitalWrite(dir, LOW);
      delay(1);
      break;
  }
}
```

Mechbot Control Code

```
//Included Libraries
#include <mechbotShield.h>
#include <avr/io.h>

//Constants
#define FASTMVMT 550
#define SLOWMVMT 350
#define NOMVMT 0

//Global Variables
char distMsg;           //If using char msg, S=STOP, F=FAR

void setup() {
  Serial.begin(19200);                              //Higher baud rate recommended for faster
response
  disableMotor();
  initMotor();
  initADC();
}

void loop() {
  if (Serial.available() > 0) {             //Reads from serial port IF there is something to
read
    distMsg = Serial.read();
  }
  //Serial.println(distMsg);
  if (distMsg == 'S') {
    motor(NOMVMT, NOMVMT);
  }
  else if (distMsg == 'F') {
    motor(SLOWMVMT, SLOWMVMT);
  }
  else {
    motor(FASTMVMT, FASTMVMT);
  }
  delay(1);
}
```

9.6 Appendix F: Calculations

Lidar scan frequency:

It is assumed that the LIDAR unit oscillates at around 1 Hz (two 180° rotations per second, back and forth). From the LIDAR specifications, it can be assumed that the scanning frequency is around 1 kHz. This would mean that in every sweep (one 180° rotation), there will be:

$$\frac{0.5\ s}{rotation} \cdot \frac{1000\ points}{s} = 500\ \frac{points}{rotation}$$

Thus, per 180° rotation, the LIDAR will be picking up around 500 points This would correlate to a angular resolution of:

$$500\frac{points}{rotation} \cdot \frac{1\ rotation}{180°} = 2.777\frac{points}{°} \Rightarrow \frac{1}{2.7777\frac{points}{°}} = 0.36\frac{°}{point}$$

This resulted in around a 0.36° resolution. If the LIDAR was scanning an object approximately 5 m away, then then distance resolution between the points would be calculated as follows:

$$0.36° \frac{\pi}{180°} \cdot 5\ m = 0.0314\ m$$

For the application of this project, it was determined that a 0.0314 m resolution at a 5 m range was more than sufficient

Motor speed:

It is required to achieve a 1 Hz rotational speed of the shaft of the motor since the LIDAR unit is attached directly to it. The motor specifications state a 0.198° step angle for the stepper motor. This means that a single pulse would cause the shaft of the motor to rotate 0.198°. Thus the number of pulses required to rotate the shaft 180° would be:

$$\frac{180°}{0.198\frac{°}{pulse}} = 909\ pulses$$

From the motor torque vs. frequency chart, the maximum pulse speed is denoted to be around 1000 pps (pulse per second). Using this information, the achievable shaft speed would be:

$$1000 \frac{pulse}{s} \cdot \frac{1\ rotation}{909\ pulse} = 1.1\ \frac{rotation}{s} = 1.1\ Hz$$

Since 1.1 Hz > 1 Hz, the motor is just barely able to rotate at the required 1 Hz required frequency

Motor Torque:

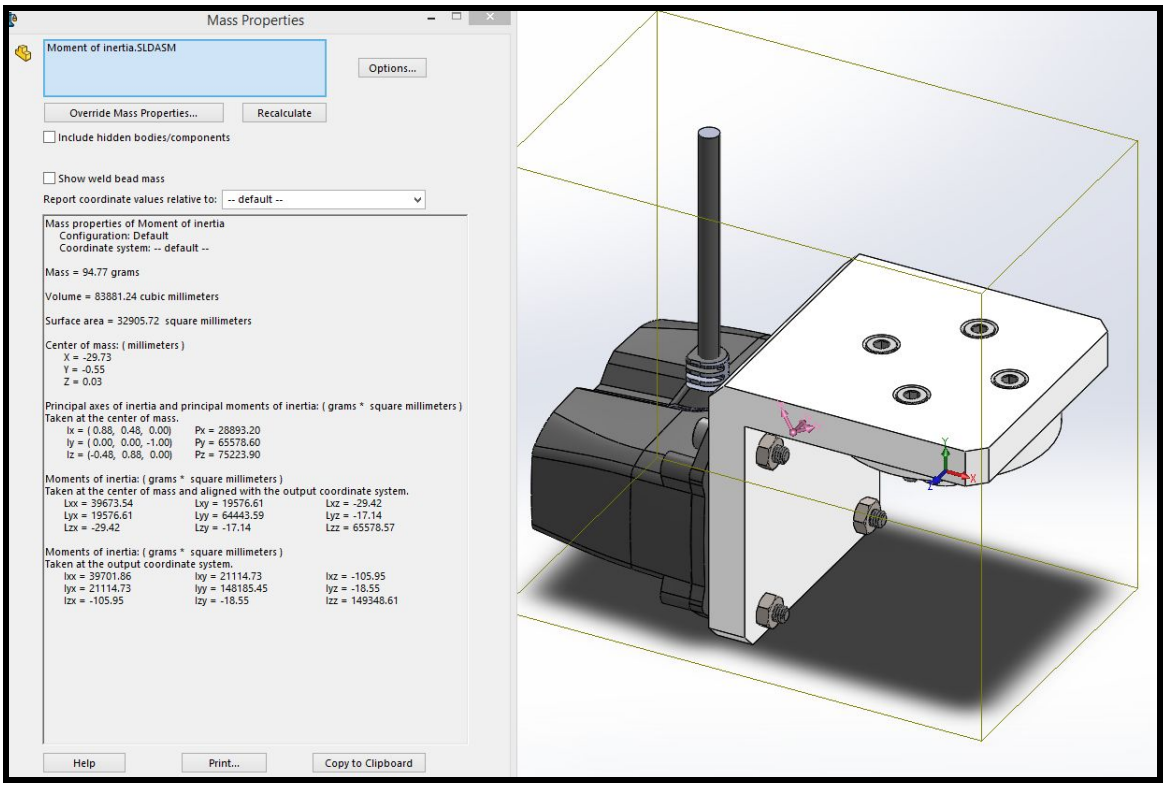

Figure 9.6.1: Use of CAD software to estimate the moment of inertia of the LIDAR oscillation assembly

Using SolidWorks and estimating the mass of the materials, the moment of inertia about motor axis was found to be around: J = **1.4819×10^{-4} kg m^2**

The approximate required peak angular velocity = ~1 Hz

The LIDAR assembly follows simple harmonic motion which can be modeled by the equation:

$$\theta = A\sin(\omega t) + \phi$$

Where,

$$A = \frac{\pi}{2}$$
$$\omega = 2\pi f = 2\pi(1\ H) = 2\pi$$

Thus,
$$\theta = \frac{\pi}{2} sin(\frac{\pi}{2} t) + \varphi$$

Taking the derivative twice:

$$\frac{d^2\theta}{dt^2} = -\ 2\pi^3 sin(2\pi t)$$

Therefore,
$$\alpha_{max} = \left| -\ 2\pi^3 \right| = 62.013\ \frac{rad}{s^2}$$

Using the equation:
$$T_{req} = J\alpha = (1.4819\ x\ 10^{-4}\ kg\ m^2)(62.013\ s^{-2}) = 9.190\ x\ 10^{-3}\ Nm$$

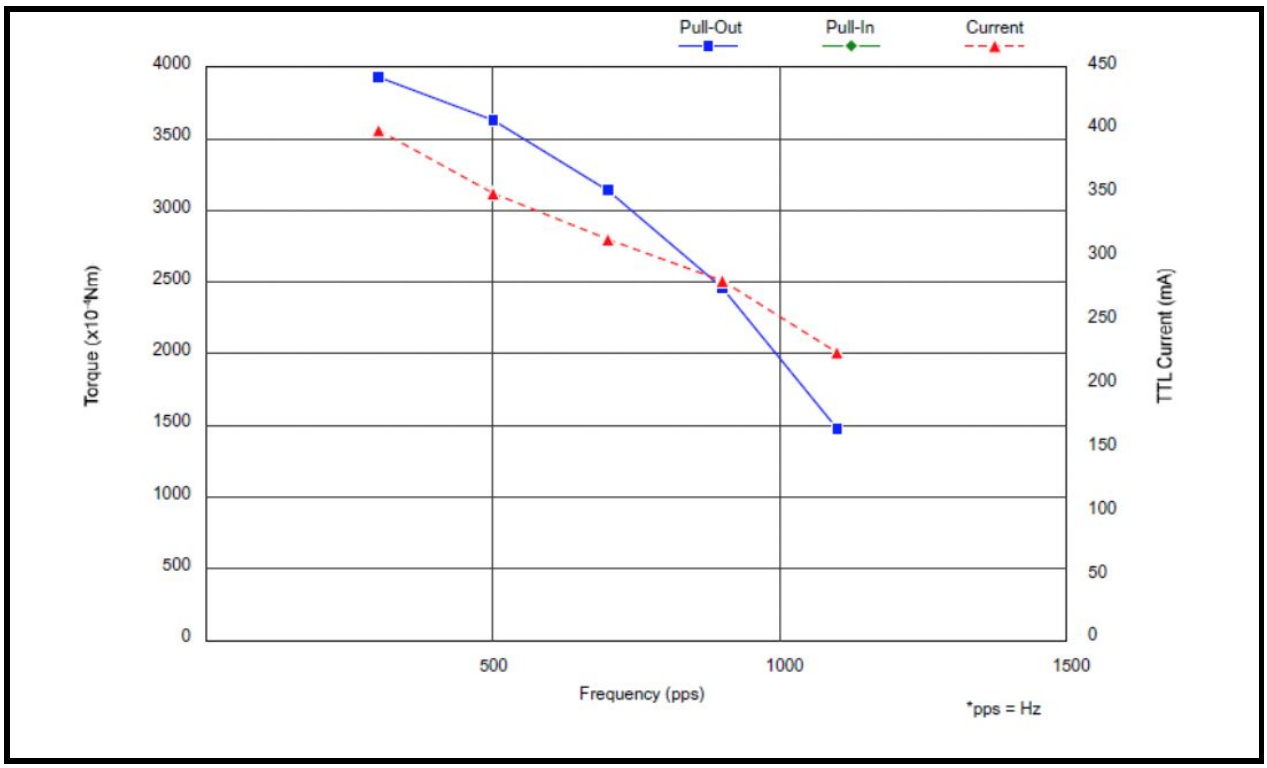

Figure 9.6.2: NMB PG20L-D20-HHC0 motor torque vs. frequency specifications [12]

For 0.198° step angle, 180° rotation => 909 pulses required for 180° rotation
$$pulse\ per\ second\ = (909\ pulses)(1\ Hz) = 909\ Hz$$

From graph above, T (@909pps) =~0.245 Nm

Since: $9.190\ x\ 10^{-3}\ Nm << 0.245\ Nm$ **the torque of the motor is more than adequate**

9.7 Appendix G: Gantt Charts

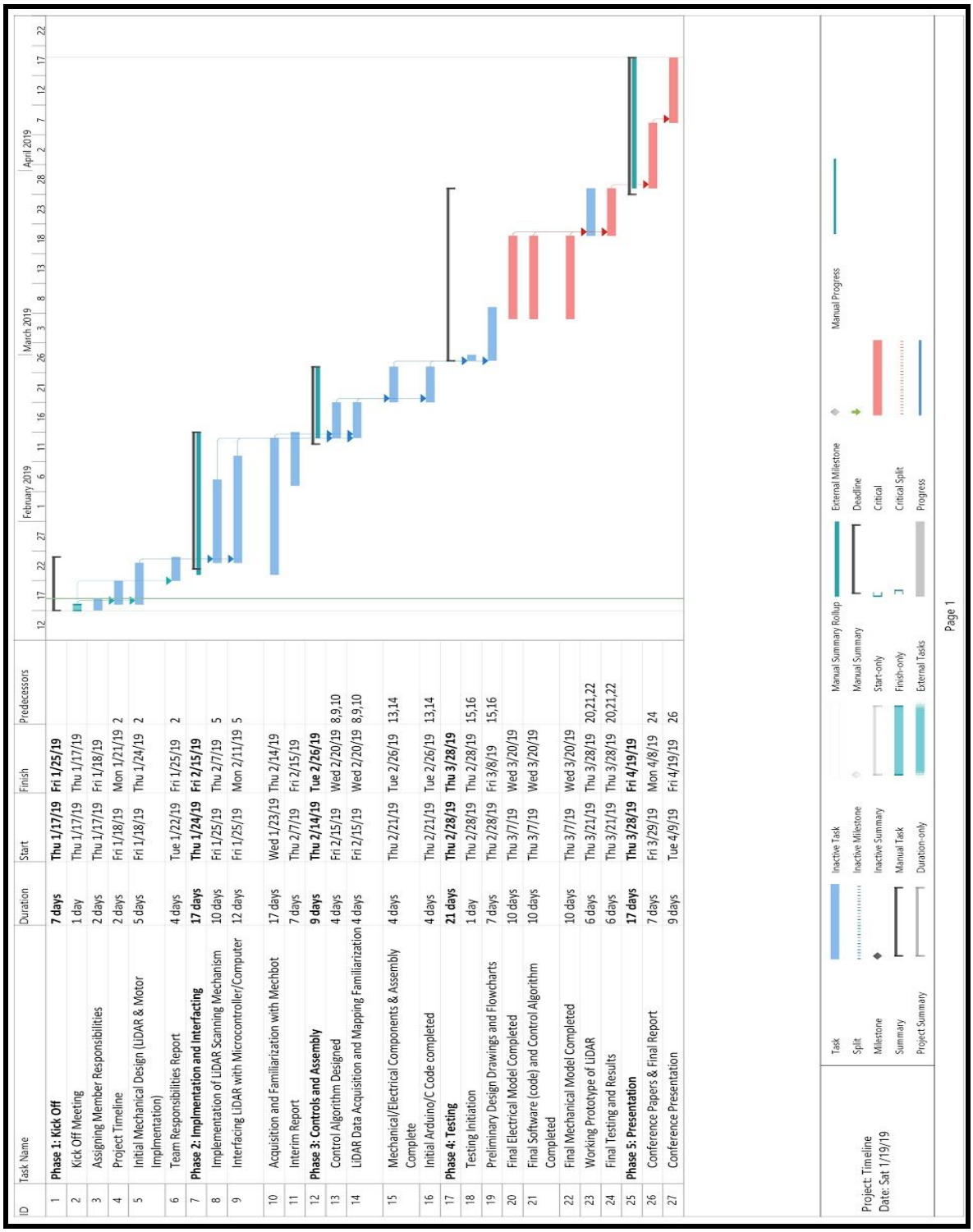

Figure 9.7.1: Initial Project Timeline Gantt Chart

Figure 9.7.2: Final Project Timeline Gantt Chart