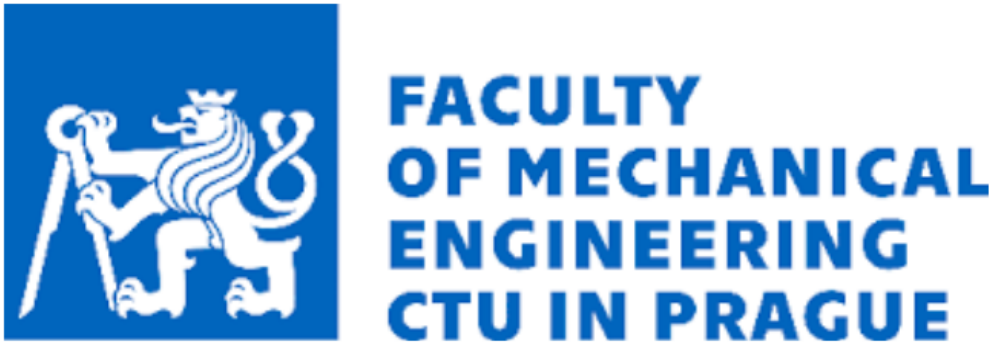
Czech Technical University

Faculty of Mechanical Engineering

Department of Instrumentation and Control Engineering



Autonomous Control of a Car Model

Diploma Thesis

Prem Wongsagoon

Supervisor: doc. Ing. Martin Novák, Ph.D.

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Wongsagoon Prem** Personal ID number: **496761**

Faculty / Institute: **Faculty of Mechanical Engineering**

Department / Institute: **Department of Instrumentation and Control Engineering**

Study program: **Automation and Instrumentation Engineering**

Specialisation: **Automation and Industrial Informatics**

II. Master's thesis details

Master's thesis title in English:

Autonomous control of a car model

Master's thesis title in Czech:

Autonomní řízení modelu vozidla

Guidelines:

1. Create a simulation model of a small car, that will automatically avoid simulated obstacles
2. Mount the Lidar on the car model and program the autonomous driving on an dSpace
3. Experimentally verify

Bibliography / sources:

[1] Bimbraw, Keshav. (2015). Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology. ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings. 1. 191-198. 10.5220/0005540501910198.
[2] Margarita Martínez-Díaz, Francesc Soriguera, Autonomous vehicles: theoretical and practical challenges, Transportation Research Procedia, Volume 33, 2018, Pages 275-282, ISSN 2352-1465

Name and workplace of master's thesis supervisor:

doc. Ing. Martin Novák, Ph.D. Division of electrotechnics FME

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **27.10.2022** Deadline for master's thesis submission: **26.01.2023**

Assignment valid until: _____

doc. Ing. Martin Novák, Ph.D. Supervisor's signature	Head of department's signature	doc. Ing. Miroslav Španiel, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Someone once said you could not have it all. I am here to tell you that you can.

Declaration of Authorship

I hereby certify that this thesis has been composed by me and is based on my own work, using only the literature and sources listed in the reference.

In Prague:..... .....

Signature

Czech Technical University

Abstract

Faculty of Mechanical Engineering
Department of Instrumentation and Control Engineering

By Prem Wongsagoon

The main target of this thesis is to create a car which can drive autonomously without any human inputs. In order to do so, this thesis will have to focus on the key component which makes the car drive autonomously, which is the sensor. The Lidar is chosen for this thesis due to the fact that it can sense 360 degrees of the surrounding environment. The sensor is simply mounted with the first front half of the car body. The dSpace MicroAutoBox III is used as a controller which is connected with other different electrical components. The next step is creating the MATLAB/Simulink model to read inputs from the sensors and send signals to trigger the actuators depending on the condition and algorithm defined. As a result, the autonomous system needs to sense the environment, determine the exact position on the road, and decide how it should behave in a given situation. Thus, combining sensor physics and the mechanical actuation of the vehicle by a software. The last part focuses on testing the built model in the clear surrounding environment and documenting the results in terms of charts showing the detection done by the sensors and their effect on the movement of the car. In the end a conclusion is made based on the work done in the project and the results obtained, in addition to an overview of future improvements and possibilities of producing a fully autonomous vehicle for consumers.

Keyword: dSpace, MicroAutoBox III, ConfigurationDesk, ControlDesk, MATLAB, Simulink, Serialine Communication, Autonomous Car, Sensor, Lidar, Raspberry pi

Acknowledgements

I would like to thank my supervisor doc. Ing. Martin Novák, Ph.D. with the help, knowledge and guidance provided for me to be able to develop this thesis. He helps me express my own ideas throughout this thesis.

Also, I would like to thank the professors of the university that provided me with the required knowledge to complete my thesis. I would like to thank Czech Technical University for giving me the opportunity to complete my master's degrees in "Mechanical Engineering".

At the same time, I would like to thank Porsche Engineering Service in Prague for providing me with the required tools and equipment for completing this project. Especially thanks to my colleagues Adam Barak, Ph.D., Marin Toupal, Ing., and Pierrick Solze, Ing. who support and believe in me.

And lastly, I would like to thank my partner, Mr. Kevin Brand, who always encourages me to be myself and supports me even in the darkest of times.

Contents

Nomenclature

Roman Symbol

Symbol	Meaning	Unit
a	Acceleration	$[m^2/s]$
A	Ampere	[A]
c	Speed of Light	$[m/s]$
d	Distance to the Target	[m]
F	Force	[N]
l	Length	[m]
m	Mass	[kg]
N	Newton	[N]
r	Radius	[m]
rpm	Revolution per Second	[rpm]
t	Time	[s]
v	Velocity	$[m/s]$
W	Watt	[W]
x	Direction in X axis	[m]
Y	Direction in Y axis	[m]

Greek Symbol

Symbol	Meaning	Unit
ϕ	steering angle	Degrees
θ	heading angle	Degrees
μ	Micro	10e-06

Acronyms

Acronym	Meaning
GND	Ground
MCTL	Motor Controller
RX	Receiver
P Controller	Proportional Controller
PWM	Pulse Width Modulation
TX	Transmitter
UART	Universal Asynchronous Receiver-Transmitter
VCC	Common Collector Voltage

Chapter 1 Introduction

Autonomous car or self-driving car is a ground vehicle that is capable of sensing its environment and moving safely with little or no human input. It combines a variety of sensors to perceive its surroundings, such as thermographic cameras, Radar, Lidar, Sonar. The development of autonomous cars has been conducted on automated driving systems (ADS) since at least the 1920s. The first autonomous car was developed in 1925 in the United States. It was the radio-controlled "American Wonder". The vehicle was in fact a Chandler Motor car which was equipped with a transmitting antenna and was operated by a second car that followed it and sent out radio impulses which were caught by the receiving antenna. The antennae would then forward the signals to circuit-breakers which operated small electric motors that directed every movement of the car. [1]

However, the real breakthrough dates back to Futurama, an exhibit at the 1939 New York World's Fair, when General Motors demonstrated its vision of a futuristic world which included automated highway systems that would guide self-driving cars. Nowadays, cars do depend on several autonomous features, but the dream of building full-fledged autonomous vehicles is still alive to this day. [2]

The important component which makes the car to be able to interact according to the surrounding environment and drive autonomously is sensors. There are different kinds of sensors that are used in an autonomous car, as mentioned above. However, this project will focus on only one type of sensor which is a Lidar. Due to the fact that this sensor is able to scan 360 degrees of the surrounding environment in a wide range, which will be further discussed in the next topic. Apart from the functions of the Lidar, this project will also focus on the data communication between the dSpace product, and autonomous car components.

1.1) Sensor

A sensor is a device that produces an electrical output signal from physical environment input. The input can be in the form of light, heat, motion, moisture, pressure, or any other environmental phenomena. There are certain features which have to be considered when choosing a sensor such as accuracy, environmental condition - usually has limits for temperature/humidity, range - measurement limit of sensor, calibration - essential for most of the measuring devices as the readings changes with time, resolution - smallest increment detected by the sensor, cost and repeatability – the reading that varies is repeatedly measured under the same environment. According to personal research, Lidar can scan 360 degrees surrounding environments, also providing the distance and how large the objects are. Due to these benefits therefore this type of sensor has been chosen.

1.1.1) Lidar

Since originating in the 1970s, Lidar technology has been widely implemented in several sectors. Its first application came in meteorology, where the National Center for Atmospheric Research used it to measure clouds. However, the accuracy and usefulness of Lidar systems became publicly realized in 1971 during the Apollo 15 mission, when astronauts used a laser altimeter to map the surface of the moon [3]

Lidar, which stands for Light Detection and Ranging, measures the distance by firing rapid pulses of laser light, usually up to 150,000 pulses per second of either visible ultraviolet or near infrared light, at a target. When the light hits the target, it gets reflected back to the sensor which then measures the time taken for the pulse to bounce back from the target. The distance is then deduced by using the speed of light to calculate the distance traveled accurately using the relation,

$$d = \frac{c\,t}{2} \tag{1}$$

Where "d" is the distance to the target, "c" is the speed light and "t" is the time of flight of the laser light.

The result is precise three-dimensional information about the target object and its surface characteristics. In the case of self-driving cars, Lidar is used to generate huge 3D maps that the car can then navigate through. High end Lidars can even identify the details of a few centimeters at a distance of more than 100 m. For example, not only can it detect pedestrians but it can also tell which direction they're facing. Of course, such autonomous technology isn't without its downfalls – take for example the fatal accident by an Uber self- driving car in Arizona where the technology failed to pick up a pedestrian crossing the road. [4]

However, as Lidar becomes more sophisticated, it will be increasingly capable of detecting and tracking objects. Improvements will mean higher resolution imagery will be possible and it will be able to operate at longer ranges so that the technology is capable of differentiating between someone walking, and someone on a bike, their speed, and direction.
Choosing a suitable Lidar sensor for the project requires deep research and analysis of the suitability of different models of Lidar sensors, in addition to availability in the Czech Republic. One of the main specifications is the "working Range" of the sensor, as in the minimum and maximum distance the laser beam can travel to the target and be reflected back while keeping in mind that the longer a beam travels, the more it will alternate as it gets reflected by the target. Another very important sensor property is of course the "Distance Resolution" which can tell us the smallest change in distance that the sensor can detect. The "Scanning Frequency" of a Lidar sensor tells us the period of time required for one full scan of a complete line when the target is

being scanned sequentially using a laser beam. On the other hand, in signal processing, sampling is a very crucial mathematical operation which performs the reduction of continuous-time signals to discrete-time signals. "Sampling Rate" is the average number of samples, a set of values at a point in time or space, which are processed in one second. According to the Lidar choices, the decision came down to RPLidar A2M8 and HLS-LFCD2 360 Degree Laser Scanner

Model	**Working Range [m]**	**Distance Resolution [mm]**	**Scanning Frequency [Hz]**	**Sampling Rate [times/s]**	**Price [CZK]**
HLS-LFCD2 360 Degree Laser Scanner[5]	0.12~3.5	<0.5	6~12	1800	1200
RPLIDAR A2M8 360 Degree Laser Scanner[6]	0.15 ~ 12	<0.5	5~15	8000	8300

Table 1 Lidar Sensor Model

1.2) dSpace MicroAutoBox III

The hardware which operates as an electronic control unit (ECU) from the dSpace developer who integrated hardware and software tools. It provides Real-Time Interface (RTI) testing. There are three softwares used in order to operate the dSpace MicroAutoBox III [7].

1.2.1) Simulink

It is a software MATLAB-based graphical programming environment for modeling, simulating and analyzing dynamical systems[8]. For this thesis, the used to simulate avoiding obstacle logic models.

1.2.2) dSpace ConfigurationDesk

This software is used to implement a configuration of input and output ports of dSpace MicroAutoBox III. In further serial communications, PMW, digital/analog input/output ports are set in dSpace ConfigurationDesk software. Also there is a special library in Simulink where the

assigned ports are connected directly to the Simulink model. After all implementations, the model is also compiled in dSpace ConfigurationDesk[9].

1.2.3)dSpace ControlDesk

This software provides Real-Time Interface (RTI) testing[10].

1.3) Summary

The main focus in this diploma thesis was to use a Lidar sensor which can be implemented in an autonomous vehicle and detect the distance between the vehicle and obstacles. And also the hardwares is decided upon the availability from the Faculty of Mechanical Engineering and Porsche Engineering Services. There are two choices, which are HLS-LFCD2 360 Degree Laser Scanner and RPLIDAR A2M8 360 Degree Laser Scanner. At the beginning of this thesis phase, the model was decided for a HLS-LFCD2 360 Degree Laser Scanner. However due to communication problems, the HLS-LFCD2 360 Degree Laser Scanner was not able to perform the scanning accurately therefore I decided to change to RPLIDAR A2M8 360 Degree Laser Scanner.

Chapter 2 Outline

This chapter will be focusing more on the technical side of the project where several components are required for the car model will be discussed thoroughly. Which are Lidar sensor, dSpace MicroAutoBox III, electric motor, and servo motor will be essential to the functionality of the car model

2.1) Lidar Sensor

The RPLidar A2 is a 360 degrees 2D scanner. It can take up to 4000 samples of the ranging per second with high rotation speed. The system can perform scanning within a 8 m range. The typical scanning frequency of the RPLidar A2 is 10hz (600rpm)[6]. The communication interface uses separate 5V DC power for powering the range scanner core and the motor system.

Figure 1 RPLidar A2 Cabling [6]

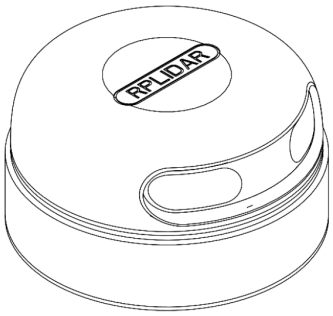


Figure 2 RPLidar A2 [6]

Color	Signal Name	Type	Description	Typical
Red	VCC	Power	Total Power	5V
Yellow	TX	Output	Serial port output of the scanner core	3.3V
Green	RX	Input	Serial port input of the scanner core	3.3V
Black	GND	Power	GND	0V
Blue	MCTL	Input	Scan motor /PWM Control Signal (active high, internal pull down)	5V (25,000Hz with 50% duty cycle)

Table 2 Configuration of RPLidar A2 [6]

2.2) dSpace MicroAutoBox III

The powerful system can be added to or replace an electronic control unit (ECU), and lets you experience and test control functionalities in a real environment. MicroAutoBox III is ideal for many different rapid control prototyping (RCP) applications from autonomous driving to zero emissions either as a single demonstrator or for equipping entire test fleets [19]. In figure 3 shown the different types of the dSpace MicroAutoBox III which are single deck and double deck. The double deck will be used for this thesis due to availability in the laboratory.

Figure 3 dSpace MicroAutoBox III single and double deck variant [19]

2.3) Electric Motor

The car model will be equipped with the SGM25F-370 motor. This type of motor is a DC geared motor which is a combination of a motor and a gearbox. The SGM25F-370 motor has a compatible dimension with 25 mm in diameter, 51 mm in body length, and the nominal torque is enough to drive 7kg mass [11]. It is suitable for a mini autonomous robotic car.

Operating Voltage	9-14 (12V)
Nominal Current	300 mA
Nominal Speed	37 rpm
Nominal Torque	258 Nmm

Table 3 SGM25F-370 DC Geared Motor specifications [11]

Figure 4 SGM25F-370 DC Geared Motor [11]

2.5) H-bridge motor driver

For controlling the motor, an L298N H-bridge motor driver is required. The role of the H- bridge is to reverse the polarity/direction of the motor, but can also be used to brake the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted [12].

Logical Voltage	5V
Drive Voltage	5V - 35V
Logical Current	0-36 mA
Drive Current	2A
Max Power	25W

Table 4 The L298N H-Bridge specifications [12]

Figure 5 The L298N H-Bridge[12]

2.4) Servo Motor

A servo motor is a type of motor which allows the precise control of angular position. In this project, the Hitec HS-311 Servo Motor model was chosen for the precise control of the steering of the car's front wheels. The servo motor, when connected to the microcontroller along with the sensors, will be programmed to determine the direction in which the car would need to move.

Voltage Range	4.8V - 6.0 V
No-Load Speed	0.19 s / 60 degrees
Max PWM Signal Range	575-2460 μs
Max Travel	202.5 degrees

Table 5 Hitech HS-311 Servo Motor specifications [13]

Figure 6 Hitech HS-311 Servo Motor[13]

2.5) MAX3232 - RS232 to TTL Serial Port Converter Module

A dSpace MicroAutoBox III is using RS232 while RPLidar A2 is using UART communication protocol[14]. Based on the oscilloscope, the main difference between these two is logic high, logic low, and logic levels. The UART communication from RPLidar A2 is between 0V and 5V while RS232 from the dSpace MicroAutoBox III is between -10V and 10V.With the help of the TTL to RS 232, it makes the communication of these two devices compatible. The figure below shows the configuration of the electrical component.

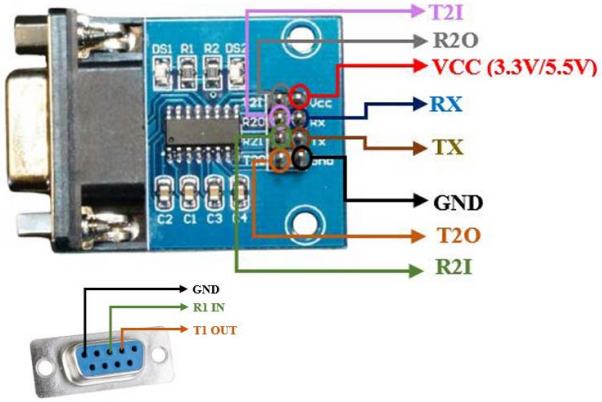


Figure 7 MAX3232 - RS232 to TTL Serial Port Converter Module [14]

This converter does not require the power supply. Therefore it can be connected directly to other components. There are 3 pins used, which be shown in the table below:

Pin No.	Comment
2	Ground(GND)
3	Receive(Rx)
5	Transmit(Tx)

Table 6 Configuration of TTL to RS 232 Converter [14]

Chapter 3 Connection and Assembly

After discussing the main components needed in section.2, now we will dive into the electrical assembly of the components and Simulink model.

3.1) Set up the configuration pins for the dSpace MicroAutoBox III in dSpace the ConfigurationDesk

The dSpace MicroAutoBox III can provide digital-analog input/output according to users' configuration in a software called ConfigurationDesk. By drag-and-drop the desired input pin and output pin according to figure 8 to the working space, "Functions", in figure 10. The corresponding configuration pins are referenced in the dSpace user document[16]. Figure 8 shows the inputs and outputs of "DS1403 Processor Board " and "DS1513 Multi-I/O Board" of the dSpace MicroAutoBox III. For this project, the pins from "DS1513 Multi-I/O Board" board will be used, which contains digital outputs. For the connectivity between the dSpace MicroAutoBox III and others electrical components will be discussed in the following topic

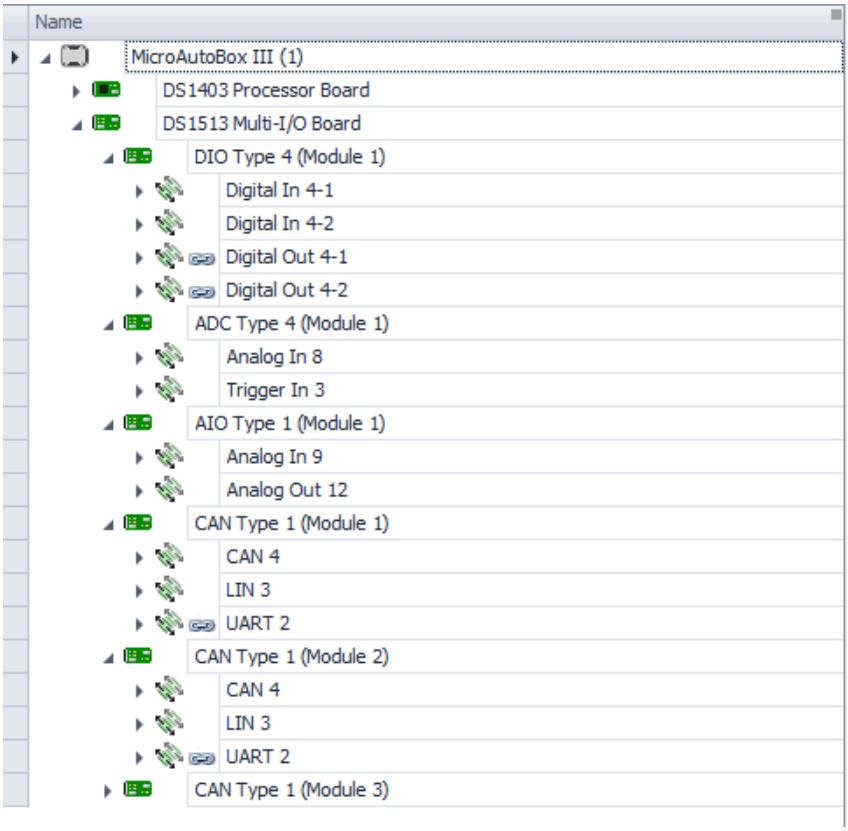


Figure 8 Inputs and Outputs Board of "DS1403 Processor Board " and "DS1513 Multi-I/O Board" of the DSpace MicroAutoBox III [9],[15]

After finishing drag-and-dropping the desired pins into “Functions” space, the next step is to propagate these set-up pins to the “Model Port Blocks” space. The blocks in this area will be presented in the Simulink model. Which will be further used for implementing the algorithm of the car and control the electrical components. Figure 10 shows the configuration of the dSpace MicroAutoBox III in ConfigurationDesk software. There are 4 blocks in total, which are “DS151x UART RS232 Receiver” which is the UART receiver that will connect to the Lidar for receiving the data, “Send the Command” which is the UART transmitter that will also be connected with the Lidar for sending the command from the dSpace MicroAutoBox III, “Steering” and “PWM/PFM Out(1)” which is the digital out PWM which will send the signal to the servo motor and Lidar respectively. All of the blocks in “Functions” space are propagated to “Model Port Blocks” space that will show in the Simulink model, which will be discussed in chapter 4. After finishing all model set-up, the “Start Build” button in figure 9 is used to compile the code. The products are the files which will be used in Real-Time Interface (RTI) in ControlDesk software, however the process to set-up the Real-Time Interface (RTI) will not be discussed in this report.

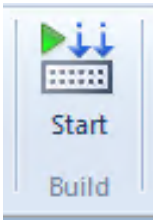


Figure 9 Start Build Button in ConfigurationDesk Software [9]

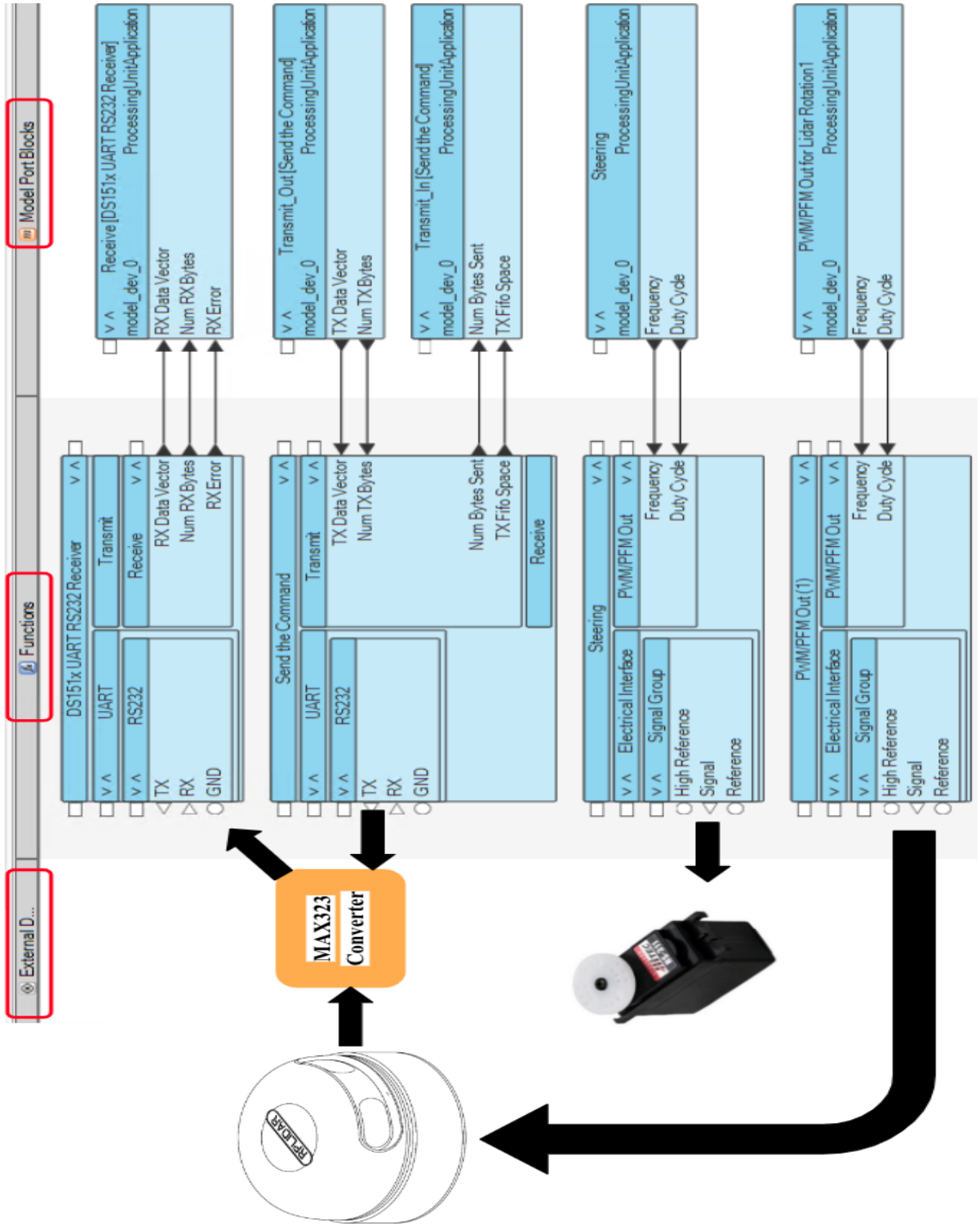


Figure 10 Overview Configuration Between Lidar, MAX323, Servo Motor, and ConfigurationDesk [9],[15]

3.2) Electrical Assembly of dSpace MicroAutoBox , Servo Motor, MAX3232 Converter, and Electrical Motor

After input pins and output pins are set up, it is now the part of assembly

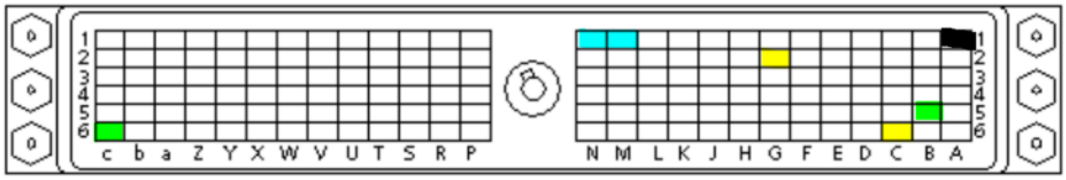


Figure 11 dSpace Input and Input Pins Configuration [15]

Using jump wires, the components will be connected to the dSpace MicroAutoBox III via extension of a breadboard. Since a digital signal is used in order to produce PWM, N1 and M1 have to be connected together to produce the reference voltage which is 5V. Since all grounds(GND) of the dSpace MicroAutoBox III are hardware inside the hardware, therefore only one pin of GND can be used. In this thesis pin A1 is ground. The connections will be detailed in the following table.

Component	Location of Pins in dSpace MicroAutobox
RPLidar A2 (MCTL)[6]	G2
TTL to RS 232 Convertor (2,3,5)[14]	c6,B5,A1
Hitec HS0311 Servo Motor (Signal Control)[13]	C6
L298N H-bridge (In1, In2)[12]	M1,A1

Table 7 Configuration of dSpace MicroAutoBox III [6],[9],[13],[14],[15]

The spinning speed of the RPLidar A2 will be dependent on PWM in the same way as the direction of rotation of Servo Motor. It is connected to pin G2. The servo motor has three total connections only, such as the yellow wire which has to be connected to a digital pin C6 and the orange wire and black wire to both the voltage and ground pins respectively. While L298N H-bridge has three separate connections other than voltage and ground. The Enable A pin is used for enabling and controlling the speed of the motor which is to be connected to the terminals A of the H-bridge, while the purpose of both Input 1 and Input 2 logic pins is to determine the direction of rotation of the motor. All the VCC(voltage common collector), Servo Motor, Enable L298N H-bridge, and Electrical Motor are connected with an external stationary power supply which is 5V.

3.3) Electrical Assembly of Lidar Sensor

Component	Location of Pins in TTL and Rs232 Converter
RPLidar A2 (Tx, Rx)[6]	Tx, Rx

Table 8 Configuration of RPLidar A2 [6]

According to figure 12, the RPLidar A2 Rx and Tx are connected directly to a TTL and Rs232 converter. It will send and receive data through the pins Tx and Rx accordingly. A VCC (voltage common collector) of RPLidar A2 is connected to an external stationary power supply. The reasons for using separate power supplies to individual components are preventing the dropping of the power to be able to provide a constant power. To sum up all the configuration that explained above, the figure 12 is visualized the hardware connectivity of this thesis.

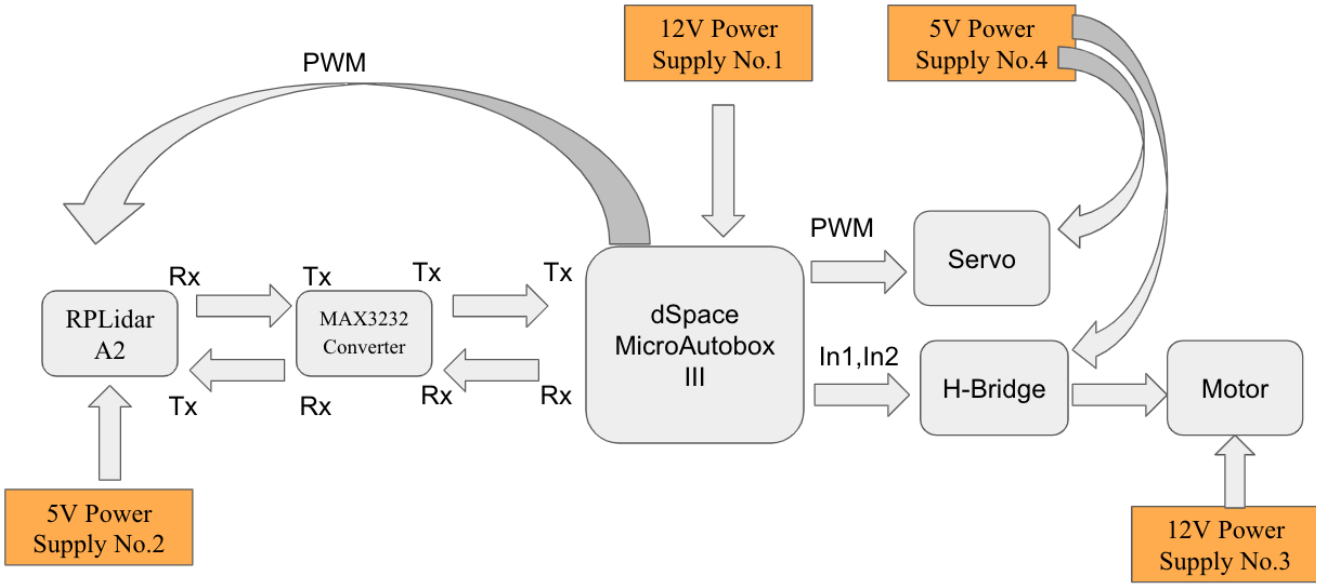


Figure 12 Overview Hardware Connectivity

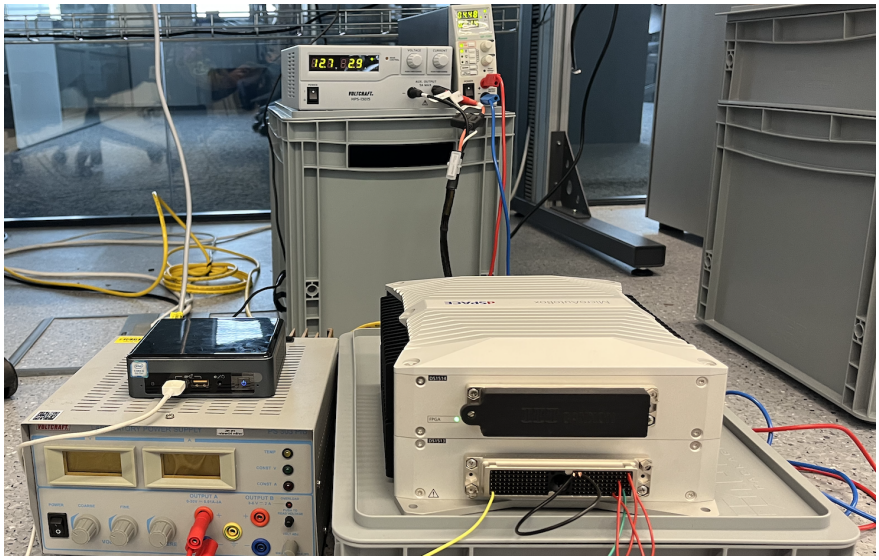


Figure 13 3 dSpace MicroAutoBox III and Stationary of Power Supplies for a DC Motor, dSpace MicroAutoBox III, and RPLidar A2

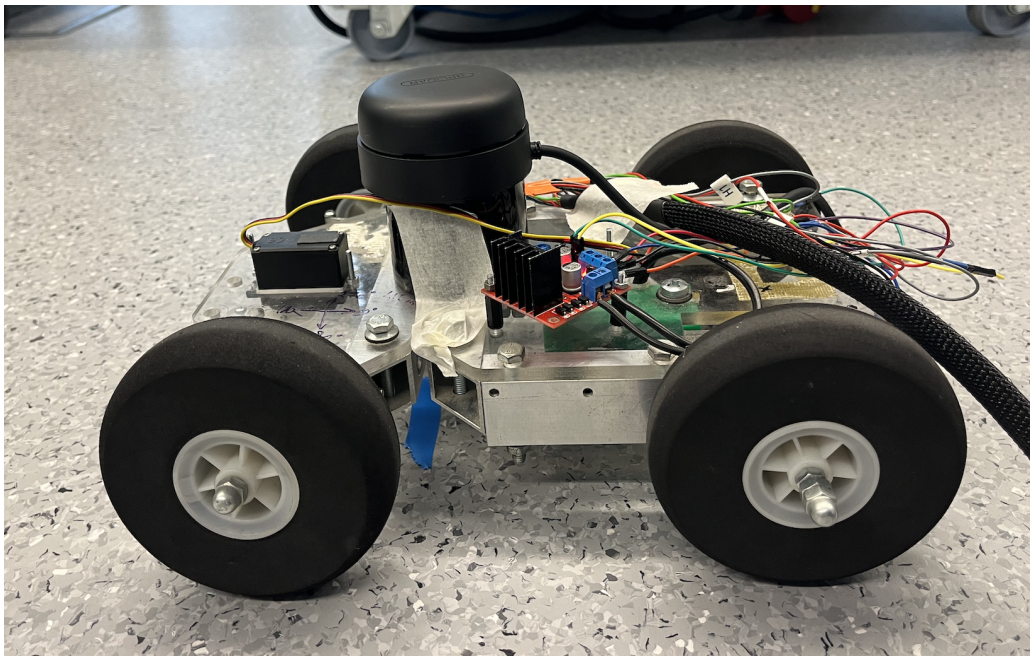


Figure 14 the Autonomous Car Mount with a RRLidar A2

Chapter 4 Offline and Online Model and Experiment

This chapter will describe models, algorithms, and how it works. In this thesis, there are 2 Simulink Models. The first is an offline model which runs in Simulink. While the second model runs in Real-Time Interface(RTI) in the dSpace MicroAutobox. There are 3 main reasons why the model has to separate into 2. First reason is to see the behavior of the car by feeding the known constant data. This allows us to observe whether the simulation results will be as expected or not. Also it is easy to re-correct the error and to tune the model. Another reason is that dSpace has its own Simulink library for RTI. To put it in another words, they have the same algorithm, only Simulink libraries are different. Lastly, the offline model is a simulation which is derived from the equation, while the online model does not require ones.

4.1) Offline Model

4.1.1) Model

The offline model is attached in appendix 1. It is running only in a MATLAB/Simulink model. By implementing the differential equation of motion of the mass position (x,y) below[16].

$$x' = v\,cos\theta\,sin\phi \tag{2}$$

$$y' = v\,sin\theta\,cos\phi \tag{3}$$

$$\theta' = v\,\frac{sin\phi}{l} \tag{4}$$

$$F = \Sigma ma \tag{5}$$

$$\tau = Fr \tag{6}$$

Where (θ) is car heading angle, (v) is velocity of a car, (ϕ) is the steering angle of the wheels, (l) is the length f between front and back rear, (x') is velocity in x-axis, and (y') is velocity in y-axis [16]. In equation 5 is Newton's second law of motion, it is the summation of the production between mass and acceleration[17]. By combining equation 5 and 6 together, acceleration and velocity can be calculated. Lastly, by integrating the velocity in x-axis (x') and velocity in y-axis (y') with respect to time, giving the position of a car in x direction and y direction respectively. The Simulink model is attached in appendix 1.

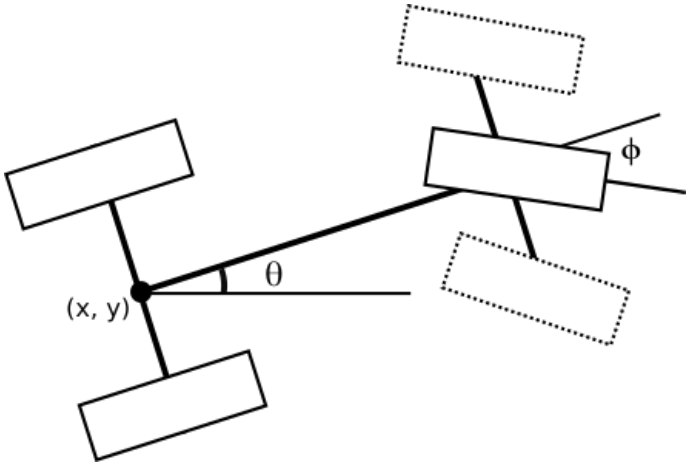


Figure 15 The Trajectory of a Car [16]

As mentioned above, the input of the model is a known constant data. As shown in figure 15, The known input data in the "Constant" block called "angle/distance", which is distances and angles that are detected from obstacles in the known surrounding environment. Based on these 2 data, the polar plot can be obtained. The figure 16 shows the plot of the target environment in MATLAB.

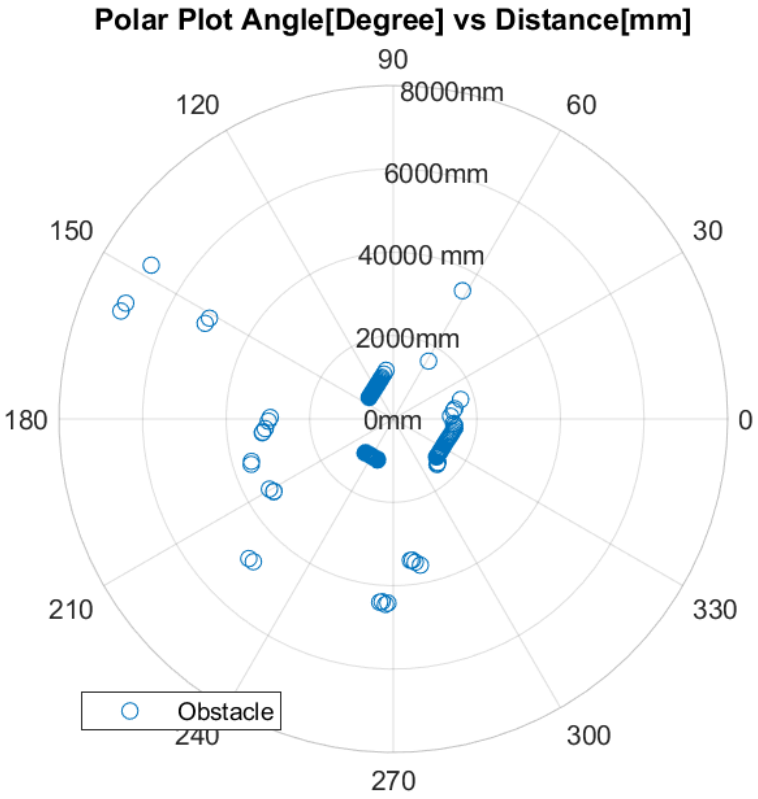


Figure 16 a Polar Plot of the Surrounded Environment Before Filter Out Undesired Objects

However, the obstacles that matter are the one that is located in front of the car. Therefore, the undesired objects are filtered out. According to the equation (2) to (6), the result of the simulated model will be in cartesian coordinates. Therefore the data has to be converted in order to combine the result with the car trajectory. In figure 17 and 18 show target object is shown in polar plot and cartesian plot.

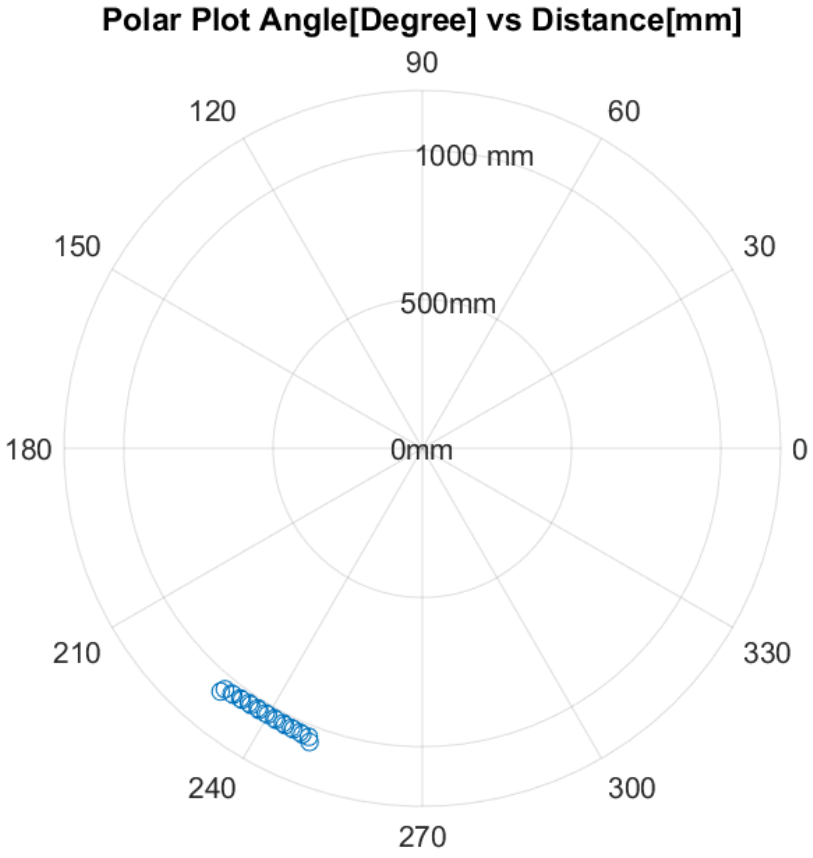


Figure 17 a Polar Plot of the Surrounded Environment After Filter Out Undesired Objects

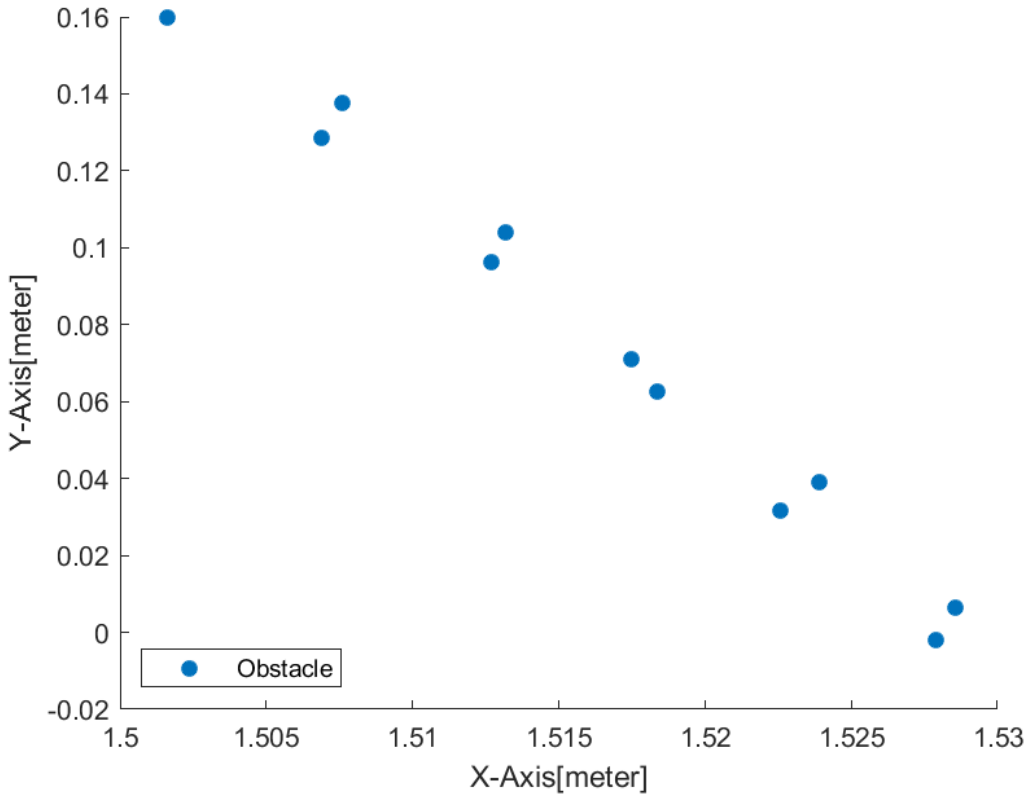


Figure 18 a CartesianPlot of the Surrounded Environment After Filter Out Undesired Objects

are fed to the "S-Function" block called "Builder_". In this function block, it creates a array of distances in 360 indexes, each index represent its own angle as shown below:

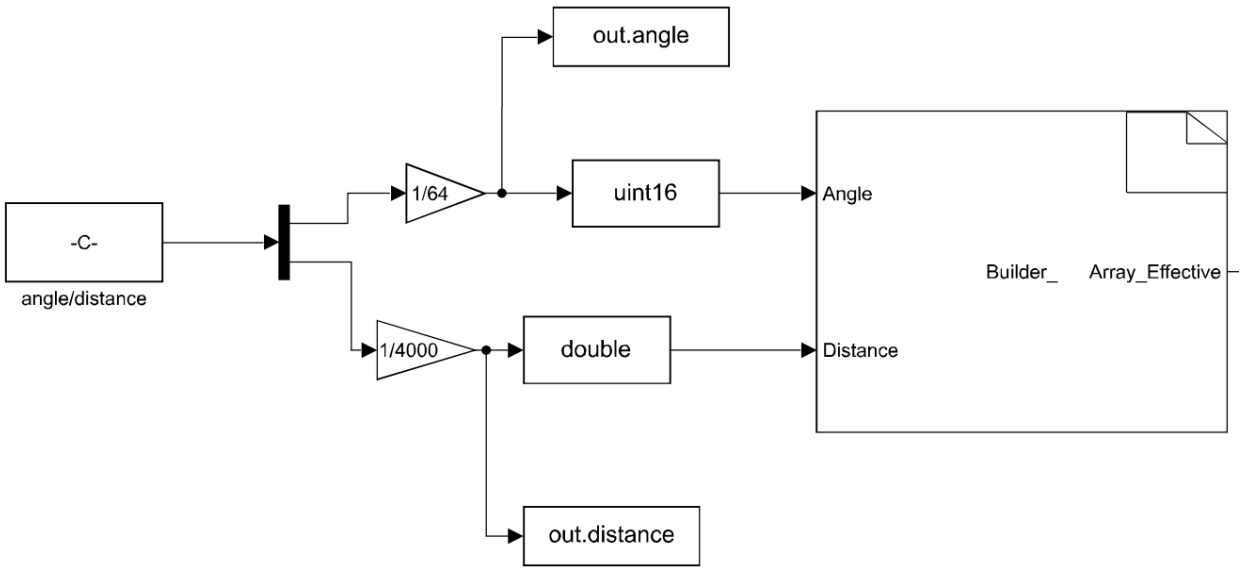


Figure 19 S-Function block in Simulink Model

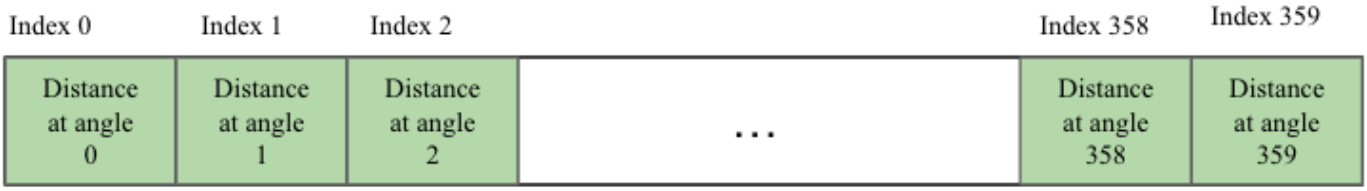


Figure 20 a Distance Array with 360 index which Represent Their Angles

4.1.2) Algorithm

As mentioned above, the offline and online model have the same algorithm. After introducing the differential equation of motion for the offline model and the distance array as an input for both models, the conditions for the driving of the car can be written. Which will be written in another "S-Function Builder" called "Check_Steering". An "if loop" is used here to determine the conditions under which the car shall be controlled. From programming we know that an "if loop" performs an action if a particular condition is satisfied. The first step begins by allowing the car to start moving forward with no steering angle if the car doesn't detect any obstacle at a distance less than 50 cm. However, should an obstacle suddenly appear at 50 cm or less than the front side of the car, then the car should be turning left. Once the sensor does not detect any object less than 50cm, the car will turn back to the same position. In figure 21, there are 3 inputs which are "object_distance" which are connected with the "From" block that linked to "Goto" from "Builder_" in figure 19, "car_distance_x" and "car_distance_y" from equation model in figure 15.

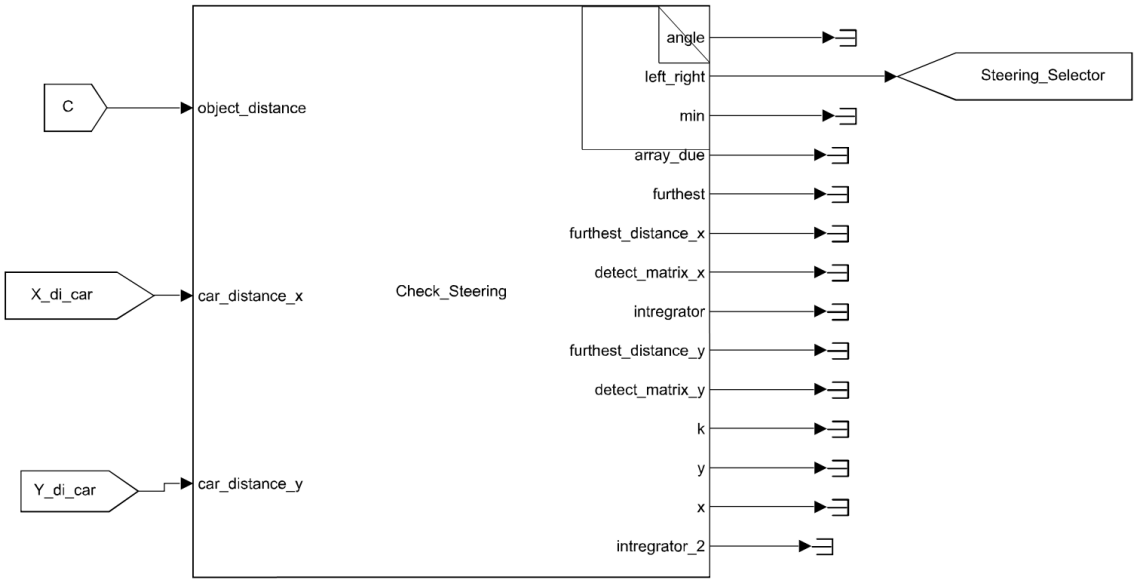


Figure 21 S-Function block in Offline Simulink Model

As shown in figure 22, the selected angle from the “S-Function” in figure 21 is feeded to the “Multiport Switch” selecting between the angle 0 radian and 0.35 radian which represent moving straight and turning left respectively. The P controller block is used to control the position of the car to return to the initial position in y direction. The controlled variable which is steering angle will be feeded to equation 2,3, and 4. Figure 22 and figure 23 visualize the overall algorithm.

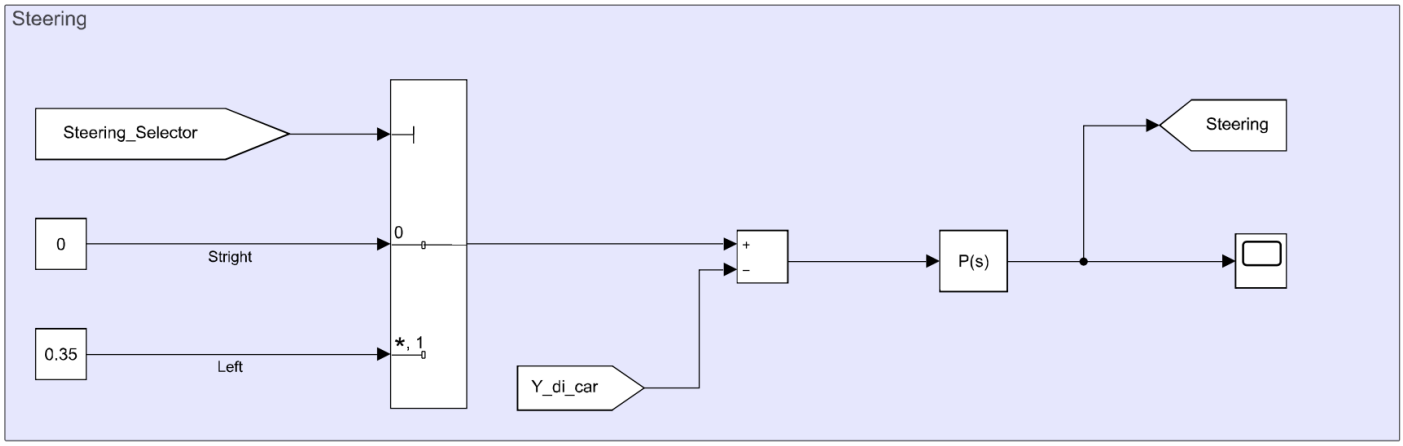


Figure 22 Control of the Steering in Offline Simulink Model

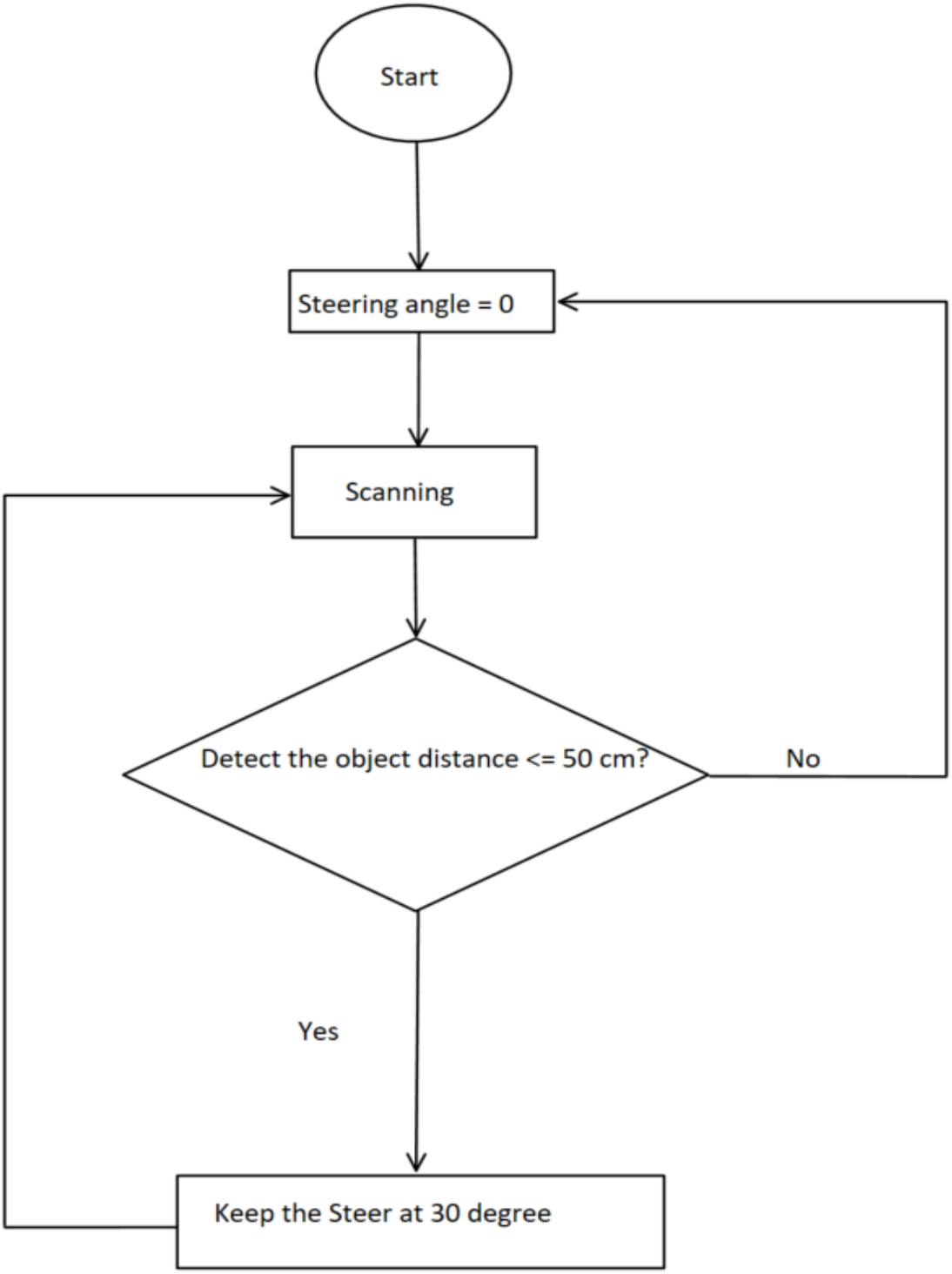


Figure 23 Flow Chart of the Implemented Algorithm

4.1.3) Simulation Result of the Offline Model

The purpose of this offline testing is to test how the car interacts with the given surrounding environment. The data is logged separately. By placing the RPLidar A2 in the desired environment and running a Python script. The Python script is sending the command "Start spinning" the motor, "Start scanning" for 10 s, "Stop scanning", and "Stop spinning" respectively. The result of this Python script is .bin file. Next step is to filter out undesired data points in MATLAB and then feed this data into the Simulink model. The result of this model is the car position in x-axis and y-axis, steering angle, and heading of a car.

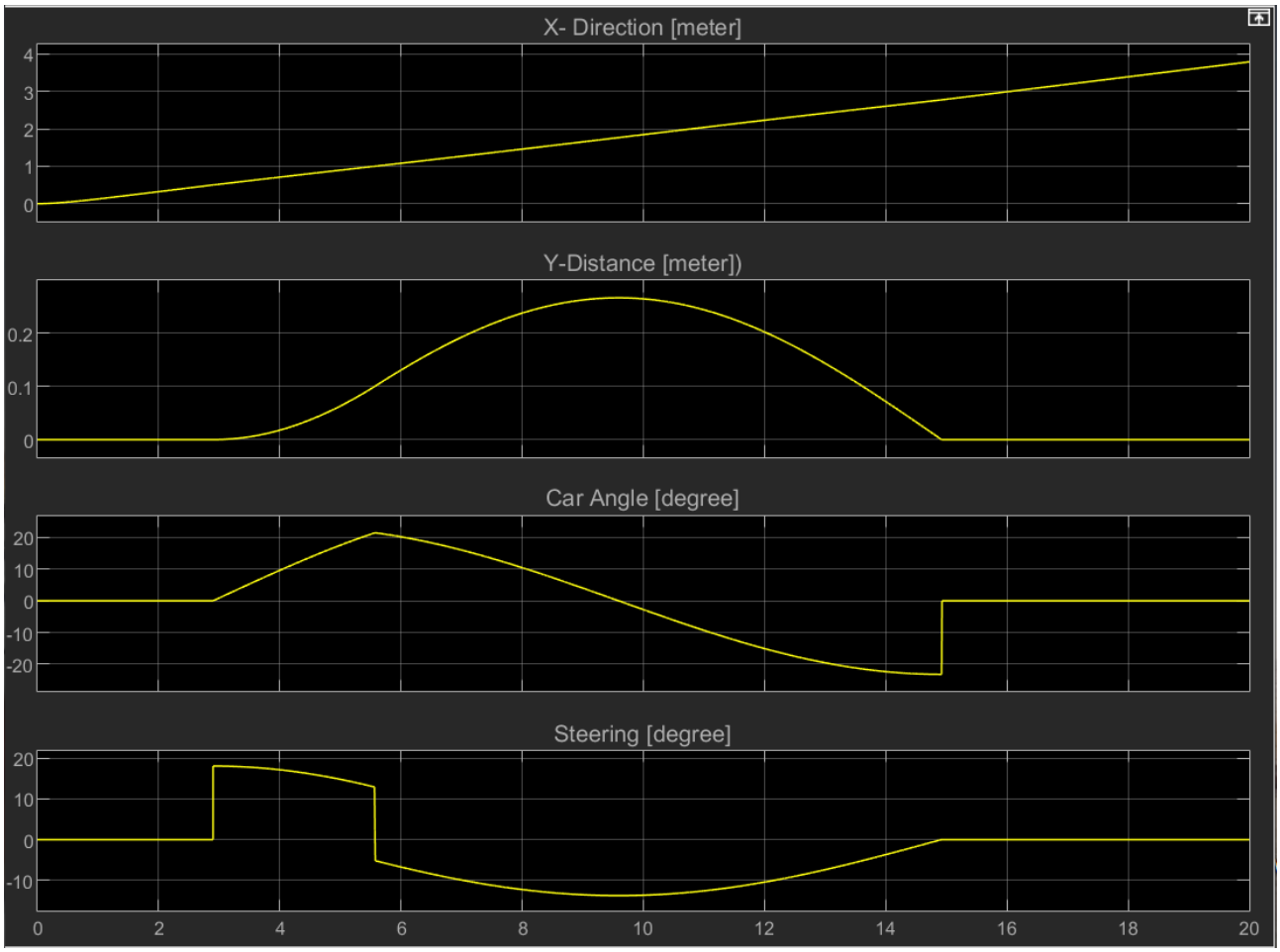


Figure 24 Result from Simulink Model which Represent X-Direction, Y-Direction, Car Angle, and Steering Angle

According to figure 24, the simulation time is 20 s while the car is moving forward in the x-direction until it reaches 4 m. However in the y-direction, the value starts to increase at time t = 2.8 s until it reaches the maximum point of 0.25 m at t = 10 s, then it drops to 0 m at time t = 15 s. For the car angle, it starts from 0 degrees. It is starting to increase at time t = 2.8 s. And it continuously increases until it reaches the maximum point of 20 degrees at time t = 5.8 s. After

that it dramatically decreases to the minimum point which is equal to -22 degrees at time t = 15 s and then jumps up to 0 degrees. Lastly is steering angle, it also starts at the value 0. And then it jumps up to 20 degrees at time = 2.8 s and slightly drops to 12 degrees once the time reaches 5.8 s. Then it dramatically decreased to -12 degrees at t = 8.7 s. Then it regulates back to 0 degrees at time = 15 s. The plot of the car position in x-axis and y-axis, red color, is shown below along with the map of the surrounding obstacles. The sharp turn at x = 2.75 m and y = 0 occurred due to an external reset integration block in the Simulink model to prevent the integration holding the value.

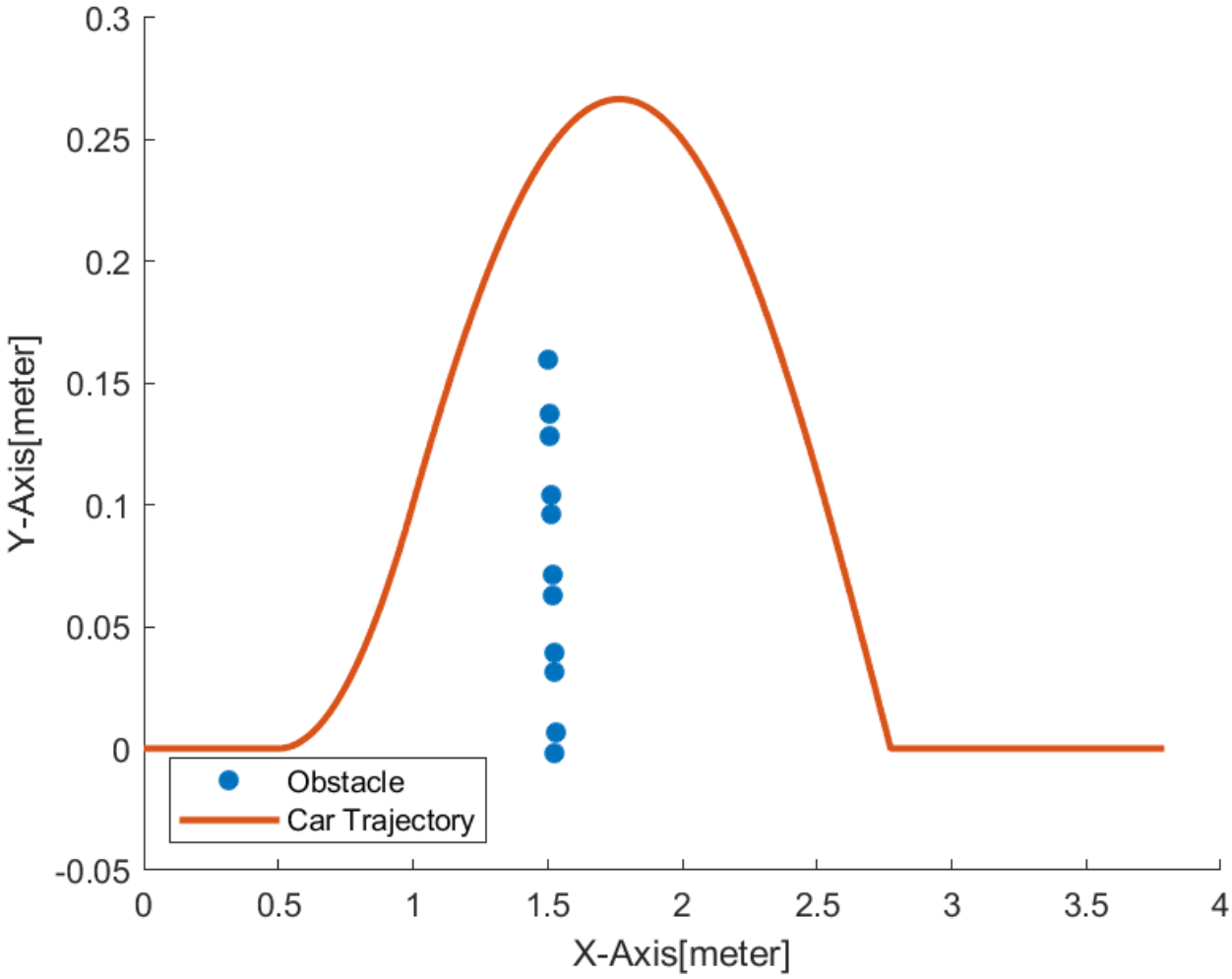


Figure 25 Plot of a Car Avoiding the Obstacle. The Red Line Represent a Trajectory of a Car while the Blue Dots Reference the Obstacle

For the offline experiment, once the car detected the obstacle, it took a left turn in order to avoid the detected obstacle. Once the car reached the x = 0.25 m and y = 2 m, it started steering back to the initial position of y which is 0 m from origin, where x = 2.8 m

4.2) Online Model

In parallel the online model, which is attached in appendix 2, does not require the above equations. The motion itself will come from the drive from the electrical motor from topic 2.3, which is powered by the external stationary power supply. Apart from that, the model contains 2 PWM controller library blocks, which will be connected to the Lidar for the rotation and another one is for the servo motor for the car steering. This block consists of 2 inputs which are *Frequency* and *Duty cycle* as shown in figure 26. The specification of the PWM for Lidar as mentioned in topic 2.1.

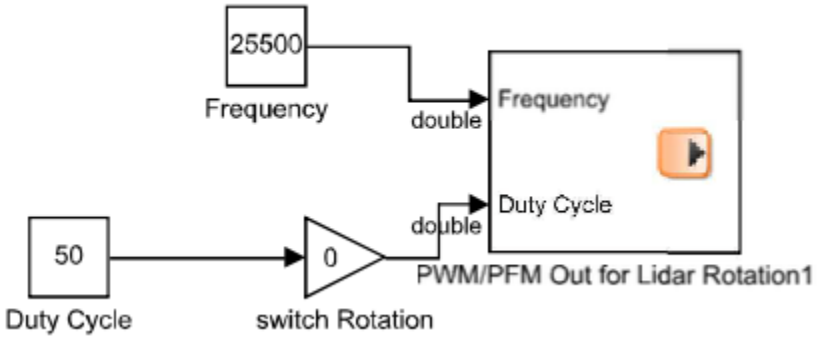


Figure 26 the Model of Lidar Rotation in Online Simulink

In figure 27, the "Constant" block "Start Scanning" contains an array which consists of 4 bytes, which are 2 bytes of "Start Scanning" and 2 bytes of "Stopr Scanning". The command will be discussed in section 4.2.3. With the help of the "Index Selection" block, the commands "Start Scanning" and "Stopr Scanning" can be selected for the real time testing. The "Switch" is to activate the data sending to the "Transmit_Out[Send the Command]" that will be switched between 0 and 1 which means activate the send and deactivate the send respectively.

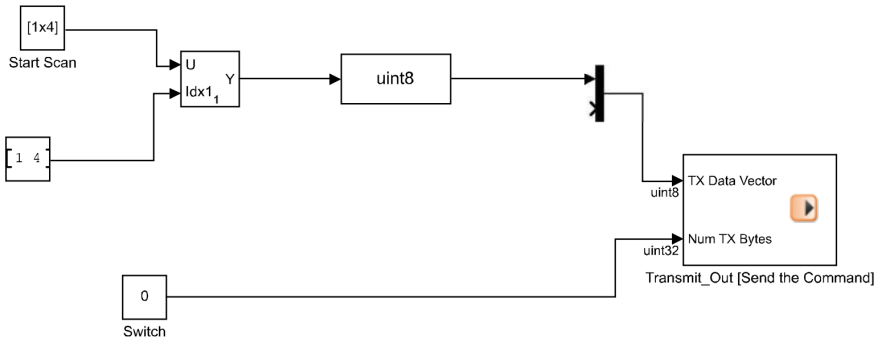


Figure 27 the Model of Sending Command in Online Simulink

4.2.1) Model

The whole model will be run in the dSpace Microautobox III where the input data will be feeded real-time by the block called “Receive [DS151x UART RS213 Receiver]1” as shown in figure 28.

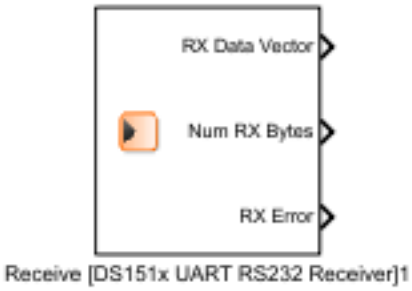


Figure 28 Receive Block from dSpace Library in Online Simulink

It will be decrypted into distance and angle in the "S-function Builder” blocks call "Distance_model_dev_0” and "Angle_model_dev_0”. The descriptive codes of these 2 block are derived according to the Interface Protocol [18] which will be shown below:

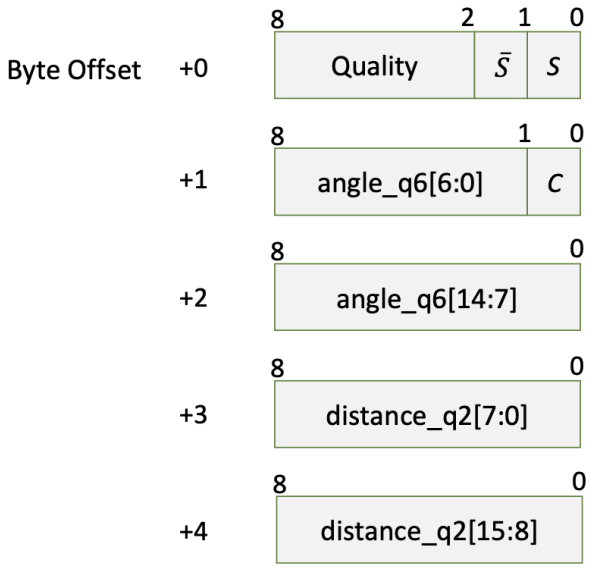


Figure 29 Format of a RPLIDAR Measurement Result Data Response Packet [18]

Variable Name	Description	Example/Note
S	Start flag bit of a new scan	When S is set to 1, the current and S Start flag bit of a new scan incoming packets belong to a new 360 degrees scan
$\overline{S}$	Inverted start flag bit, always has $\overline{S}$!= S	Can be used as a data check bit
C	Check bit, constantly set to 1	Can be used as a data check bit
quality	Quality of the current measurement Related the reflected laser pulse sample	Related the reflected laser pulse sample strength.
angle_q6	The measurement heading angle Refer to the below figure related to RPLIDAR's heading. In details. degrees unit, [0-360) Actual heading = Stored using fix point number.	Refer to the below figure related to RPLIDAR's heading. The degrees unit, [0-360) Actual heading = Stored using fix point number. angle_q6/64.0 Degree
distance_q2	Measured object distance related to RPLIDAR's rotation center. In millimeter (mm) unit. Represents using fix point. Set to 0 when the measurement is invalid.	Refer to the below figure for details. Actual Distance =distance_q2/4.0 mm

Table 9 Field Definition of a RPLIDAR Measurement Result Data Response Packet [18]

The result of processing data are distances and angles. The data will be feeded into the "S-Function Builder" block so called "Builder_model_dev_1 " which is equivalent to "Builder_" in offline model. This function block creates an array of distances in 60 indexes which represents the front view of the car so called "Array_Front". Where the effective angle of the car is set from 329 degrees to 359 degrees and 0 degrees to 29 degrees. The reason for this is because the car is only driving forward. And in order to steer the car back, the Lidar has to check the availability space on the right side of the car. Therefore the second array is used to represent the distance from angle 60 degrees to 120 degrees, so called "Array_Right". Therefore the in front obstacle counted. The figures 30, figure 31 and figure 32 are visualized these method:

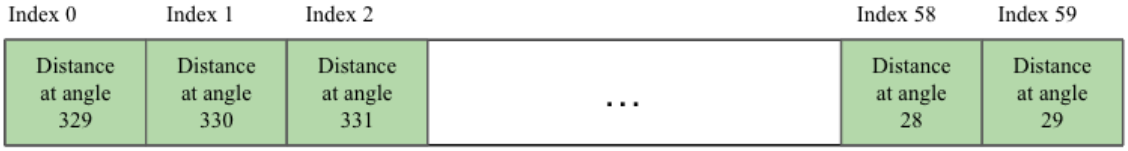


Figure 30 a Frontal Distance Array with 60 index which Represent Their Angles

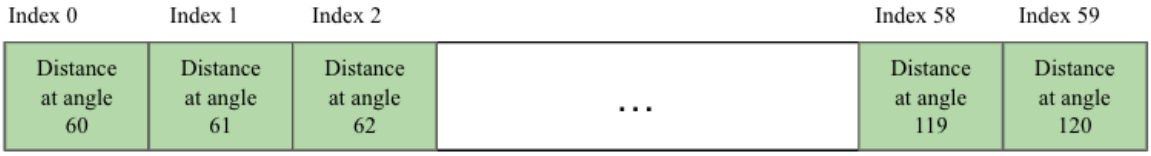


Figure 31 a Right Side Distance Array with 360 index which Represent Their Angles

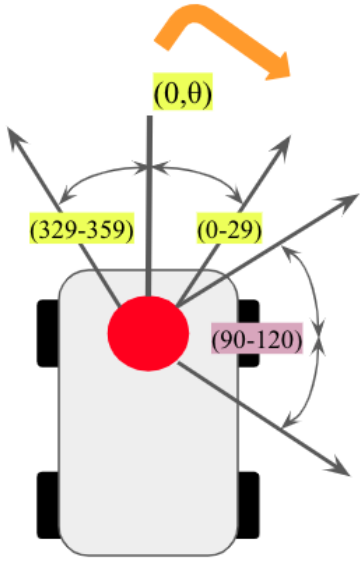


Figure 32 the Frontal Angle and the Right Side Angle in a Car, the Red Circle Represents Lidar

4.2.2) Algorithm

The offline model and online mode have exactly the same algorithm except there is no reference of the car from the starting point. To put it in another words, the model does not know how far the car has traveled in both x direction and y direction. Therefore the car has no reference to steer back into the same position. As shown in figure 33, by solving the problem, the Integration block is used. Once the obstacle is detected, the "Steering_Selection_model_dev_0" block will send the signal called "Intregrator_Trigger" to the Integration block which will use a reference of a car in y direction. It will keep integrating a positive value until the car succeeds to avoid the obstacle. Once the car is turning back it will integrate a negative value. Therefore the summation of this integration is equal to zero. Apart from that the S-Function block will send the signal called "Trigger_Steering" which will act as a switch in order to switch between driving straight or turning left. The "GoTo" block calls "Steering_Selector" will be fed in the "Multiport Switch" in figure 34.

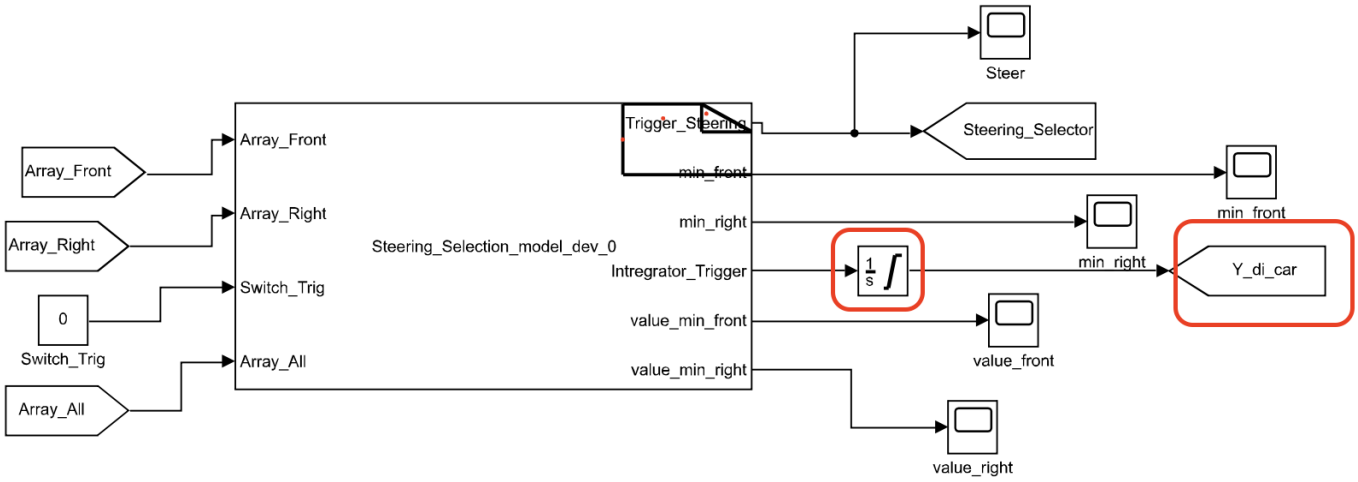


Figure 33 the Model of Steering Selection with "S-Function" block in Online Simulink

For the servo motor, the input frequency = 40 Hz while the duty cycle is switched between 4.5% and 6.5%, according to table 10. The P controller block is used to control the position of the car to return to the initial position in y direction.

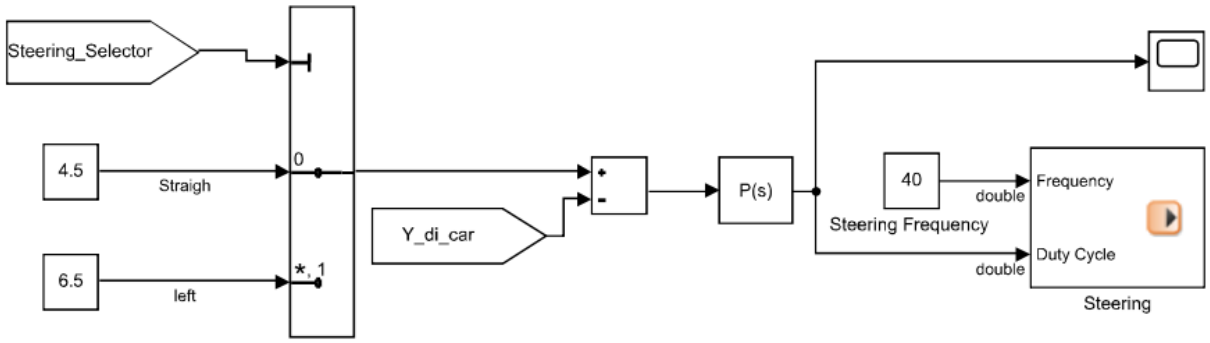


Figure 34 Control of the Steering in Offline Simulink Model

Duty Cycle [%]	Steering Angle [degrees]
2.5	+45
4.5	0
6.5	-45

Table 10 Duty Cycle [%] vs Steering Angle [degrees] [13]

4.2.3) Start Lidar Scanning

After assembling all of the components and mounting them on top of the car model, the next important step will be creating the model in MATLAB/Simulink in order to spinning the Lidar and send the command scanning, receive the data, and be able to decode and run it in a Real-Time application. The Simulink model consists of input/output blocks from dSpace's library. After power up the Lidar with 5V power supply. The motor control of the Lidar is controlled by a PWM signal according to table 10, which is 25,500 Hz frequency and 50% duty cycle. As shown below, the "gain" block is switched between "1" and "0" which represents "turn on the rotation" and "turn off the rotation" respectively. This switch will be controlled in the software called dSpace ControlDesk. The next step is to send the command to the Lidar. There are 2 commands that will be used in this thesis. The "Start Scanning" and "Stop Scanning" commands. The "Start Scanning" is used to start the Lidar scanning. After the Lidar receives this command, it will send 400 samples/s each sample containing 5 bytes which will be descript according to figure 26 and table 9. And "Stop Scanning" is to stop the process of the Lidar. Once the Lidar receives this command, I will stop sending the data. The command is shown as below :

Command	Comment
0xa5 0x21	Start Scanning
0xa5 0x25	Stop Scanning

Table 11 Command Start and Stop Scanning for RPLidar A2 [17]

The following figure is to visualize the whole process of the Lidar spinning and scanning. Starting from spinning the Lidar, following by sending the command to the scanning the data and sending the data, stop scanning, and stop spinning.

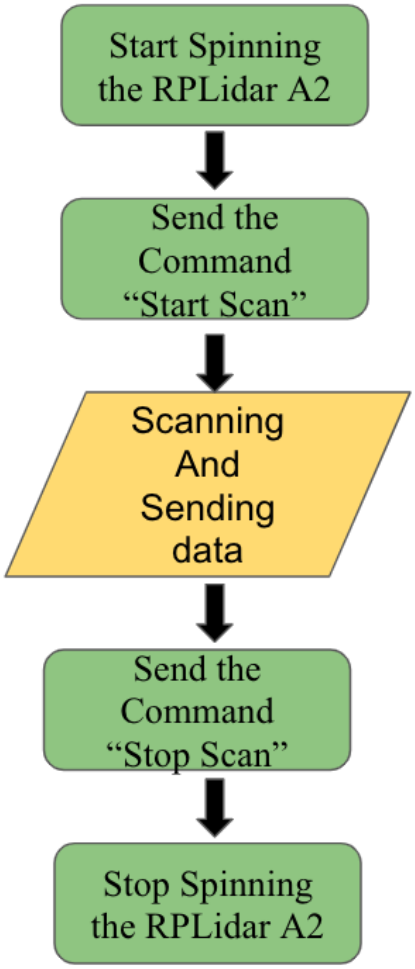


Figure 35 the FlowChart Shows the Sequence of Scanning and Spinning

4.2.3) Online Testing

As we arrive at the final chapter, this experiment will be concluded by a series of three test drives to evaluate the functionality and the limitations of the car model. The car will be driven in an almost clear surrounding environment with 5 experiments of an obstacle.

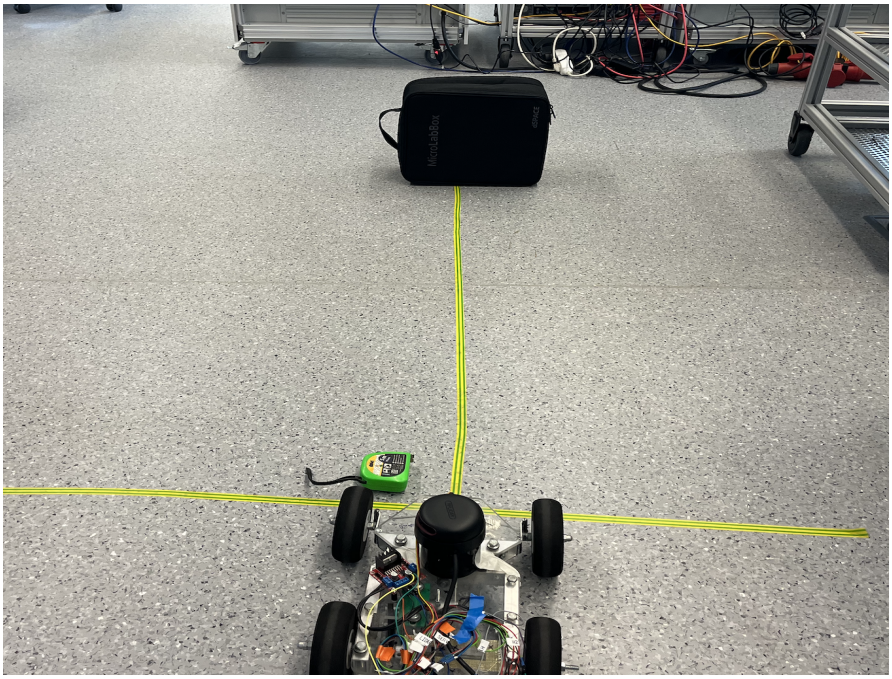


Figure 36 the Car with the Obstacle In Front

The collected data which plots the result surrounding the environment is logged by the Python script. As shown in figure 36, the color tape is used as a reference to origin (x=0,y=0). The position of a car is measured by measuring tape and timer, it is recorded manually. There are 2 separate charts. The first chart represents the position of the car in x-axis and y-axis versus time. In the second chart is the movement of a car in a plot of the surrounding environment. The car is initially moving only forward with steering angle 0 degrees. The steering angle of this car is set to be -45 degrees as minimum and +45 degrees as maximum. The aim of repeating the same test 5 times is to be able to see whether there will be large changes in the results.

4.2.3.1) The First Test

Time [s]	Direction in x-axis [m]	Direction in y-axis [m]
0	0	0
5	0.4	-0.1
10	0.6	-0.3
20	1	-0.65
30	1.3	-0.8
35	1.8	-1

Table 12 Result from Experiment no.1 Showing Time[s] vs Direction in X-Axis [m] vs Direction in Y-Axis [m]

At the time t = 0 s, the car starts at the position x = 0 m and y = 0 m. The position of a car in both x and y direction started to increase at times greater than 0 s. As the time goes by, the position of the car increases continuously until it reaches x = 1.8 m and y = 1m in absolute value. The plot of the car position in x-axis and y-axis, red color, is shown below along with the map of the surrounding obstacles. The green rectangle represents the target object that the car will avoid.

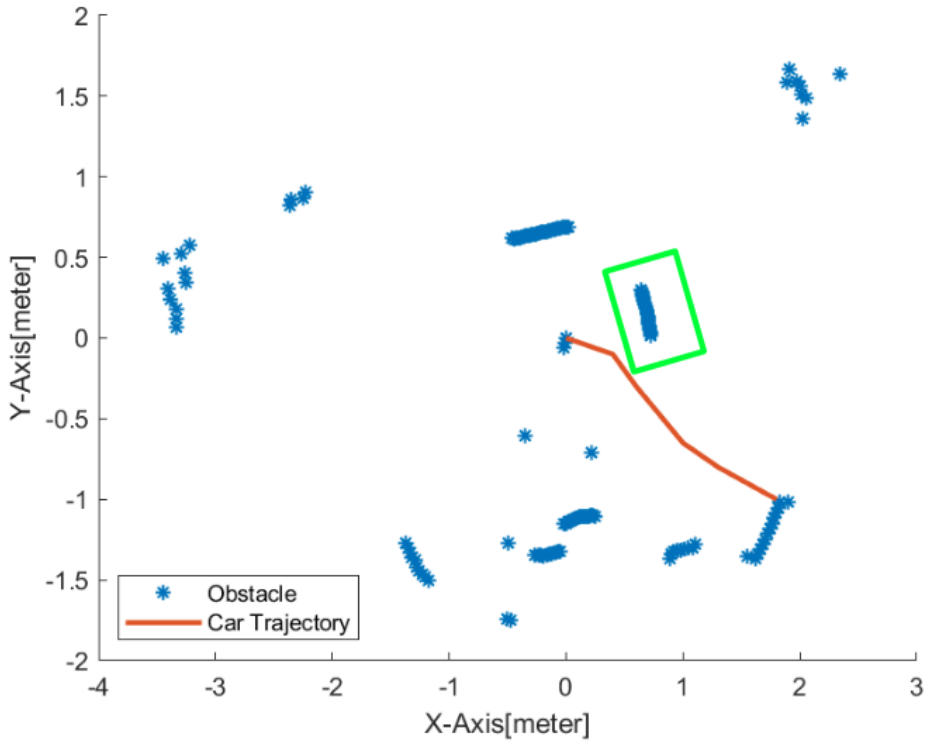


Figure 37 Plot of Surrounding Environment with a Car Trajectory from Experiment no.1

For the first experiment, it took up to 35 s to finish the experiment. Once the car detected the obstacle, it took a left turn with 45 degrees to avoid. And it kept the steering angle and it kept moving forward, and suddenly crashed with the surrounding obstacle as shown in figure 34. The car failed to make itself back to the same position. According to personal hypothesis, there is one big object that is located behind the main obstacle, which is not shown in the figure 18 due to unreachable initial Lidar scanning. And once the car was about to steer back to the same potion, the Lidar detected that the huge second object was less than 50 cm away. Therefore the car kept the right steering until it hit the object that was located near-by.

4.2.3.2) The Second Test

Time[s]	Direction in x-axis [m]	Direction in y-axis [m]
0	0	0
5	0.3	-0.1
10	0.6	-0.3
15	1	-0.32
20	1.3	-0.25
25	1.6	-0.21
30	2	-0.15
35	2.3	-0.10

Table 13 Result from Experiment no.2 Showing Time[s] vs Direction in X-Axis [m] vs Direction in Y-Axis [m]

At the time t = 0 s, the car starts at the position x = 0 m and y = 0 m. The position of a car in both x direction and y direction increases continuously until the time t = 15 s where the absolute value of the x and y direction are 1 m and 32 m respectively. Which is represented as the car is trying to make the curvature or turning. And then the position in y direction decreases almost to the same initial position which is 0.10 m in absolute value. While the position in x direction increases to 2.3 m. Which is represented as the car is attempted to move forward while getting back to the started position. The plot of the car position in x-axis and y-axis, red color, is shown below along with the map of the surrounding obstacles. The green rectangle represents the target object that the car will avoid.

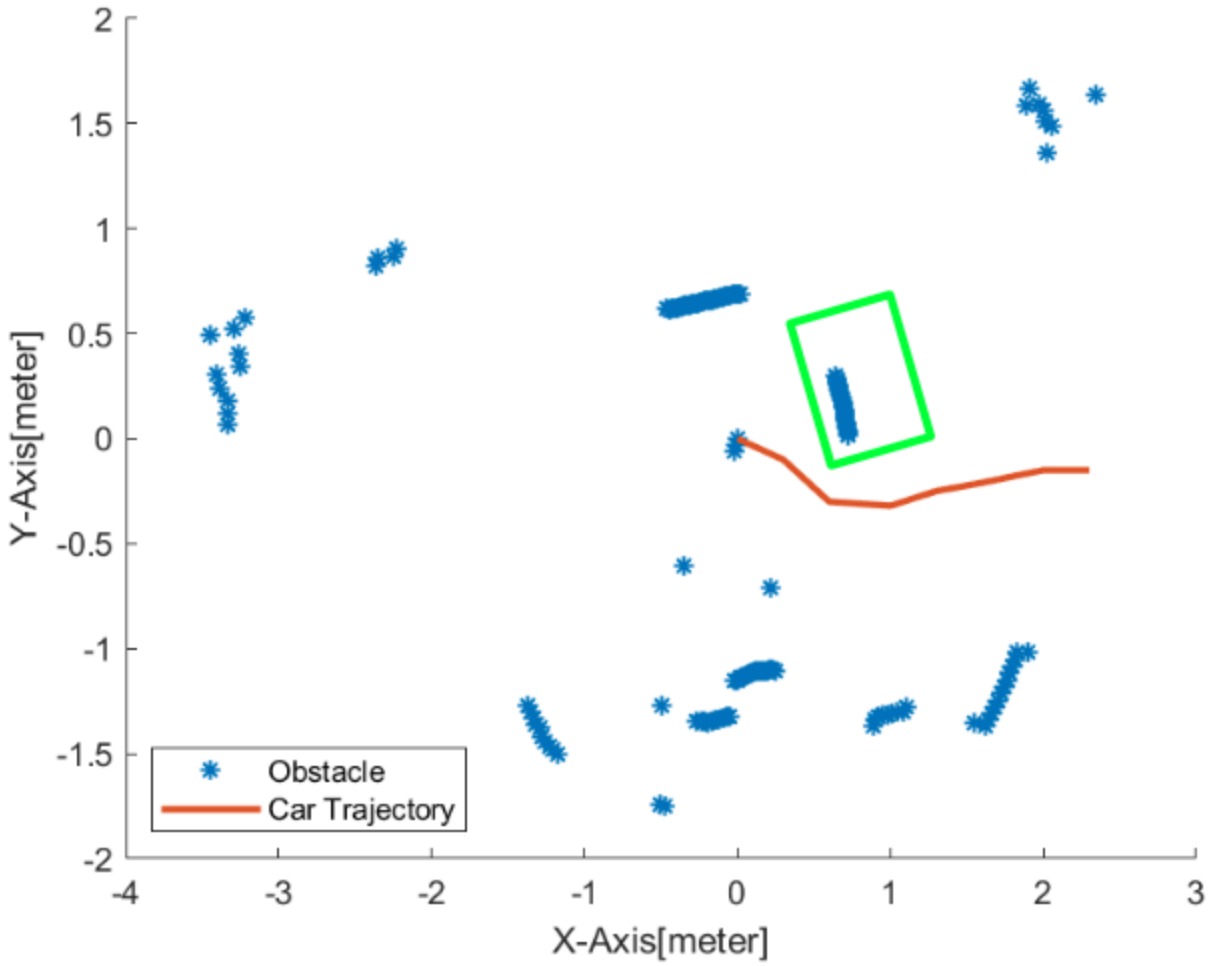


Figure 38 Plot of Surrounding Environment with a Car Trajectory from Experiment no.2

For the second experiment, the huge obstacle behind the main obstacle was moved further away from the view of the experiment. The second test took up to 35 s. Once the car detected the obstacle at a time greater than 5 s, it took a left turn of 45 degrees to avoid it. Also continued driving straight at a time 10 s to 15 s. And once the car safely avoided the obstacle and there were no more in front obstacles. The car was starting to steer back at a time greater than 20 s. Lastly the car was back to the almost same y position with 15 cm error.

4.2.3.3) The Third Test

Time[s]	Direction in x-axis[m]	Direction in y-axis[m]
0	0	0
5	0.2	-0.1
10	0.6	-0.3
15	1	-0.35
20	1.3	-0.3
25	1.6	-0.3
30	2	-0.2
35	2.3	-0.15
40	2.5	-0.12

Table 14 Result from Experiment no.3 Showing Time[s] vs Direction in X-Axis [m] vs Direction in Y-Axis [m]

At the time t = 0 s, the car starts at the position x = 0 m and y = 0 m. The position of a car in both x direction and y direction increases continuously until the time t = 20 s. Which is also represented as the car is trying to make the curvature or turning. After that the value in the y direction is constant at the time t = 20 s till t = 25 s which is 0.3m in absolute value. While the position of the x-axis continues increasing to 1.6 m. Which can be represented as the car is driving straight during this period of time. At the time greater than 30 s, the position in y direction decreases almost to the same initial value which is 0.12 m in absolute value. While the position in x direction continues to increase to 2.5 m. Which is represented as the car is attempted to move forward while getting back to the started position. The plot of the car position in x-axis and y-axis, red color, is shown below along with the map of the surrounding obstacles. The green rectangle represents the target object that the car will avoid.

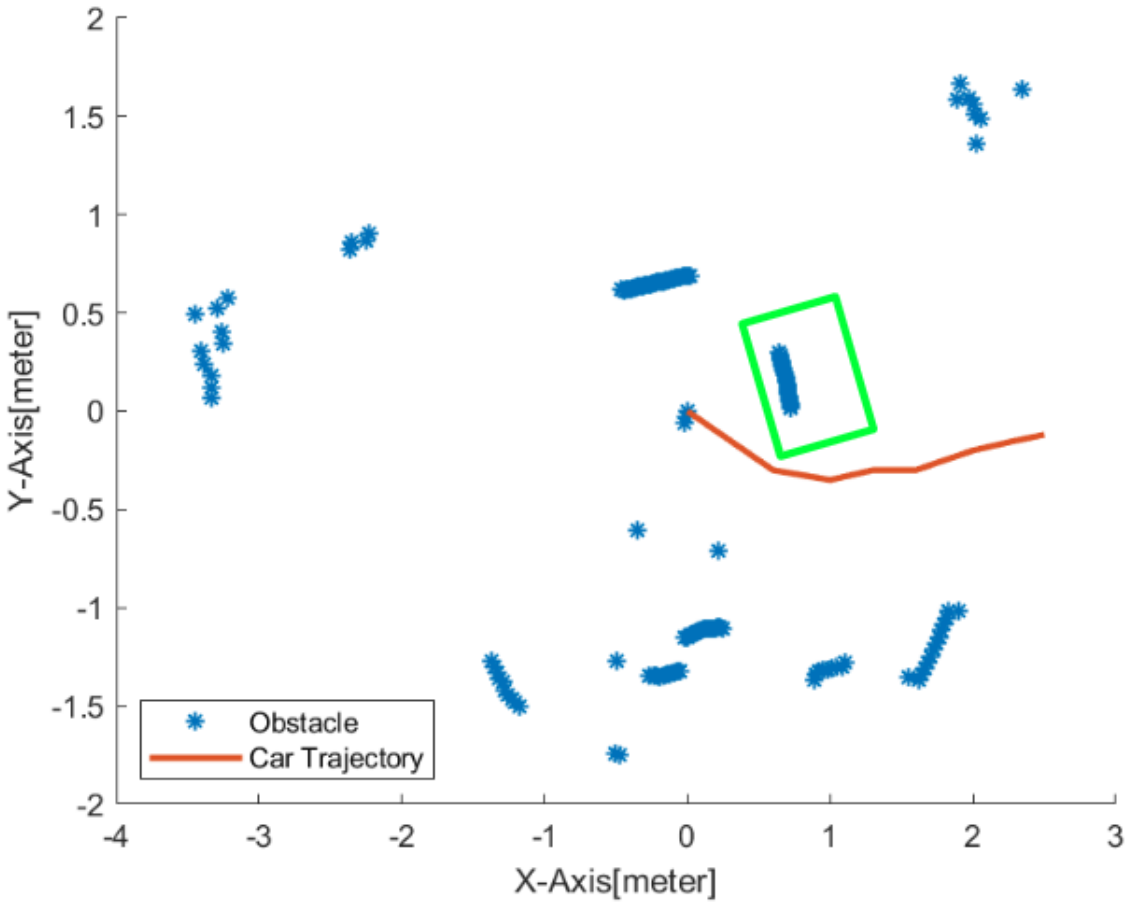


Figure 39 Plot of Surrounding Environment with a Car Trajectory from Experiment no.3

For this experiment, it took up to 40 s. Once the car detected the obstacle at a time= 5 s, it took a left turn to avoid it. Also continued driving straight at a time 10 s to 25 s. And once the car safely avoided the obstacle and there were no more in front obstacles, the car was starting to steer back at a time greater than 30 s. Lastly the car was back to the almost same y position with 12 cm error. Compared to the second experiment, this experiment took 10 s longer to steer the car back to the same position.

4.2.3.4) The Fourth Test

Time[s]	Direction in x-axis[m]	Direction in y-axis[m]
0	0	0
5	0.15	0
10	0.4	-0.15
15	0.62	-0.37
20	0.77	-0.40
25	0.95	-0.42
30	1.3	-0.34
35	1.48	-0.25
40	1.55	-0.12
45	1.65	-0.05
50	1.74	-0.05
55	1.88	-0.04

Table 15 Result from Experiment no.4 Showing Time[s] vs Direction in X-Axis [m] vs Direction in Y-Axis [m]

At the time t = 0 s, the car starts at the position x = 0 m and y = 0 m. The position of a car in x direction increases to 0.25 m at t = 5 s while in y direction maintains at 0 m. At the time t = 20 s, the position of the y direction starts to change to 0.15 m. Which is also represented as the car is trying to make the curvature or turning. Both distances in x and y direction continue to increase until t = 25 s. The car reaches the maximum distance in y direction which is 0.42 m while x = 0.95 m. At the time t greater than 30 s, the distance in y direction starts to decrease continuously almost to the same initial value while the position in x direction continues to increase. Which is represented as the car is attempted to move forward while getting back to the started position. The plot of the car position in x-axis and y-axis, red color, is shown below along with the map of the surrounding obstacles. The green rectangle represents the target object that the car will avoid.

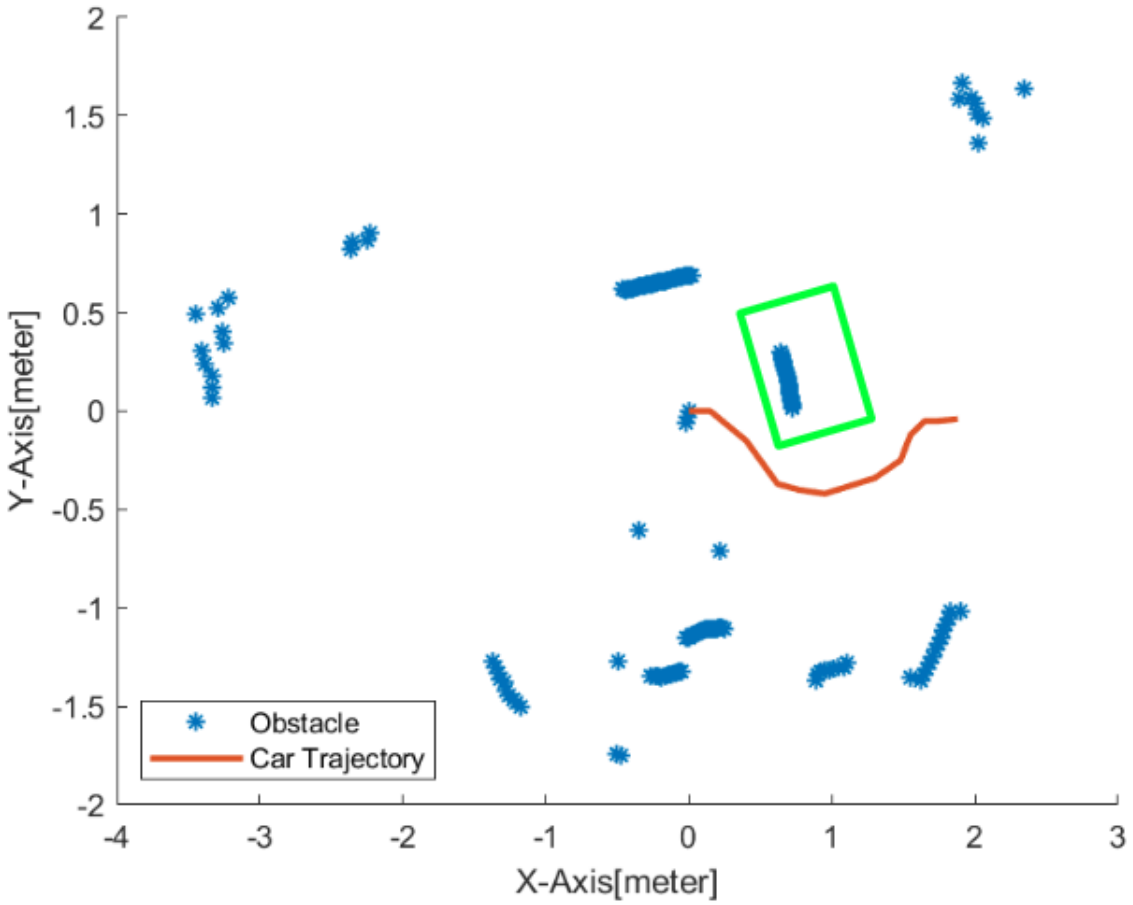


Figure 40 Plot of Surrounding Environment with a Car Trajectory from Experiment no.4

For the fourth experiment, it took up to 55 s to complete the experiment. The car drove straight for 5s. And once the car detected the obstacle at a time greater than 5 s, it took a left turn to avoid it. As shown in figure 20, the car took a big steering angle. It took time to steer in order the obstacle from t = 10 s to t = 25 s. And once the car safely avoided the obstacle and there were no more in front obstacles, the car was starting to steer back at a time greater than 35s. Lastly the car was back to the almost same y position with 4 cm error. Compared to all experiments, this experiment took the longest time.

4.2.3.5) The Fifth Test

Time[s]	Direction in x-axis[m]	Direction in y-axis[m]
0	0	0
5	0.42	-0.15
10	0.65	-0.37
15	0.74	-0.40
20	0.90	-0.42
25	1.2	-0.35
30	1.45	-0.29
35	1.62	-0.15
40	1.8	-0.10
45	2.2	-0.09

Table 16 Result from Experiment no.5 Showing Time[s] vs Direction in X-Axis [m] vs Direction in Y-Axis [m]

The car starts at the position x = 0 m and y = 0 m at time t = 0 s. The position of a car in both x direction and y direction increases continuously until the time t = 20 s where absolute value of the car in x and y direction are equal to 0.90 m and 0.42 m, respectively. Which is also represented as the car is trying to make the curvature or turning. At the time t greater than 25 s, the distance in y direction starts to decrease continuously to 0.09 m in absolute value. At the same initial value while the position in x direction continues to increase to 2.2 m. Which is represented as the car is attempted to move forward while getting back to the started position. The plot of the car position in x-axis and y-axis, red color, is shown below along with the map of the surrounding obstacles. The green rectangle represents the target object that the car will avoid.

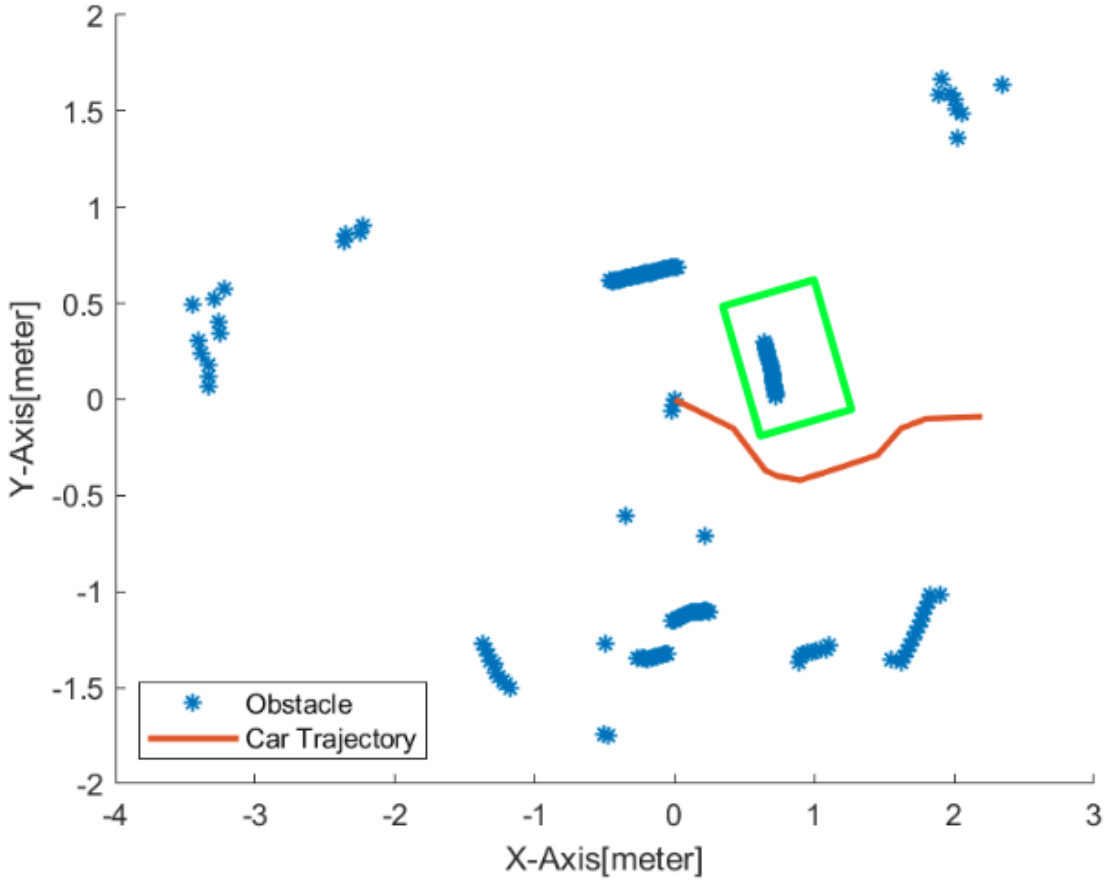


Figure 41 Plot of Surrounding Environment with a Car Trajectory from Experiment no.5

For the last experiment, it took up to 45 s to complete the experiment. Once the car detected the obstacle at a time greater than 5 s, it took a left turn to avoid it. As shown in figure 20, the car also took a big steering angle. And once the car safely avoided the obstacle and there were no more in front obstacles, the car was starting to steer back at a time greater than 35s. Lastly the car was back to the almost same y position with 9 cm error. Compared to all experiments, this experiment took the longest time.

Chapter 5 Discussion and Conclusion

5.1) Discussion

First, the offline model. The model is created in the software so called MATLAB/Simulink which is a graphical programming environment for modeling, simulating and analyzing dynamical systems[8]. For this approach, the used to simulate avoiding obstacle logic models. Once the model is created, the next step is to feed information of the distance and angle of the obstacle under the known environment into the developed model in Matlab/Simulink and then run in a loop containing conditions for the behavior of the components. Depending on the values of the inputs, or in other words the distances measured by the sensors, the code checks which conditions are satisfied and then proceeds to send signals to the components. The behavior of the components then reflects the position of the car with respect to the surrounding obstacles and defines its movement. According to figure 21 and 22, once the car is getting closer (in the direction of x) to the obstacle, the steering increases such that the car increases its y direction in order to avoid the obstacle. And once the car passes the obstacle and there are no more objects in front of the write side, the car steers back to the initial position of y.

Second part is the online model. By replacing the simulated position and steering, and constant known feeding with DC motor and servo motor, and Lidar respectively. The approach to develop this project was made with the help of dSpace MicroAutoBox III where all heart components for controlling the car's movement, as the Lidar is connected via the converter which is communicated directly to dSpace. The servo motor was used to control the steering of the front wheels of the car while the DC motor was responsible for the forward movement of the rear wheels. The L298n H-bridge played a role in controlling the DC motor. However the main part is to descript the received data from the Lidar and produce inputs to force a reaction to the components. These so-called reactions are what generate the action of autonomous driving. Basically, the sensors read the distance between the car and the obstacle, and send the data into the dSpace MicroAutoBox III to descript into the products of distances and angles. Then these 2 pieces of information is feeded into the developed model in MATLAB/Simulink and then run in a loop containing conditions for the behavior of the components. Depending on the values of the inputs, or in other words the distances measured by the sensors, the code checks which conditions are satisfied and then proceeds to send signals to the components. The behavior of the components then reflects the position of the car with respect to the surrounding obstacles and defines its movement.

The concept of a car avoiding the obstacle is to simulate the real road environment. Once the Lidar detects an obstacle, the car should be able to avoid the left lane of the road and be able to steer back to the same position once there is no longer an obstacle. However, due to the limitations of the clear surrounding environment, the ride was not as smooth as expected. In most of the test results, we are able to observe moments where the readings of the sensors

managed to confuse the car and send it off its expected track such as in the first experiment. A main reason which can be attributed to the confusions is the default from the algorithm. Once the Lidar no longer detects the object in some certain angle, it does not overwrite the array of that certain position. In other words, the algorithm is not able to update the empty array.

5.2) Conclusion

This thesis has examined the possibility of developing an autonomous model and the challenges associated with such a project. The biggest challenges are processing fast feeding data in real time, and mobility difficulty for the experimentation due to many power supplies and heavy hardwares. However, by taking the time to solve the problem step by step, models in both offline and online were successfully tested.

5.3) Future work and Proposition

Despite the overall success of this project, each part of the system could be improved. As it stands, the system represents the bare minimum functionality expected of an autonomous vehicle. One improvement can be done to improve the algorithm to be more effective, and make the object more precise and accurate. And in addition, the testing experiment should be more mobility and in a sufficient surrounding environment in order to validate the experiment. Also the DC motor is not powerful enough to carry the power supplies and a dSpace MicroAutoBox III, the longer cable would help with the scenario.

References

[1] “Self-driving car.” *Wikipedia*, https://en.wikipedia.org/wiki/Self-driving_car. Accessed 03 December 2022.

[2] “Bonnie. History of The Autonomous Car. TitleMax.” *GRINGER*, 26 September 2019, https://www.bosch-mobility-solutions.com/en/products-%20and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-%20systems/construction. Accessed 01 January 2023.

[3] “Davies, Alex. “What Is Lidar, Why Do Self-Driving Cars Need It, and Can It See Nerf Bullets?” *WIRED*, 6 February 2018, https://www.wired.com/story/lidar-self-driving-cars-luminar-video/. Accessed 06 January 2023.

[4] “Uber crash shows 'catastrophic failure' of self-driving technology. The Guardian.” *LEVIN, Sam*, 26 September 2019, https://www.theguardian.com/technology/2018/mar/22/self-

[5] “슬라이드 0.” *ROBOTIS e-Manual*, 17 February 2014, https://emanual.robotis.com/assets/docs/LDS_Basic_Specification.pdf. Accessed 15 March 2021.

[6] “RPLidar A2 Low Cost 360 Degree Laser Range Scanner. Introduction and Datasheet.” *Shanghai Slamtec.Co.,Ltd*, https://bucket-download.slamtec.com/20b2e974dd7c381e46c78db374772e31ea74370d/LD208_SLAMTEC_rplidar_datasheet_A2M8_v2.6_en.pdf. Accessed 05 December 2022.

[8] “Science: Radio Auto - TIME.” *Videos Index on TIME.com*, http://content.time.com/time/magazine/article/0,9171,720720,00.html. Accessed 03 December 2022.

[7] “Simulink - Simulation and Model-Based Design - MATLAB & Simulink.” *MathWorks*, https://au.mathworks.com/products/simulink.html. Accessed 10 October 2022.

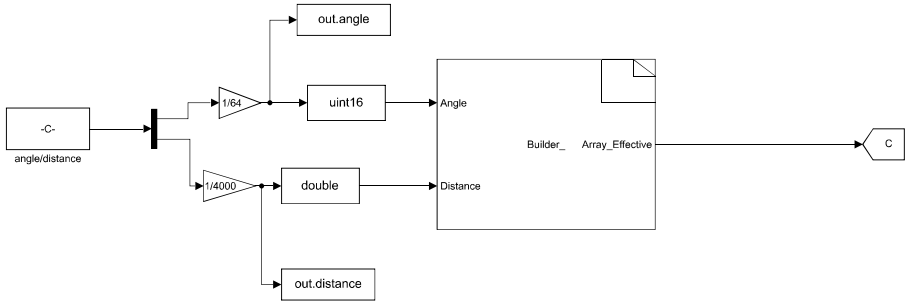
[9] “ConfigurationDesk Configuration and implementation software for dSPACE real-time hardware.” *dSpace*, 26 September 2019, https://www.dspace.com/en/inc/home/products/%20sw/impsw/%20configurationdesk.cfm#175_25012. Accessed 05 January 2023.

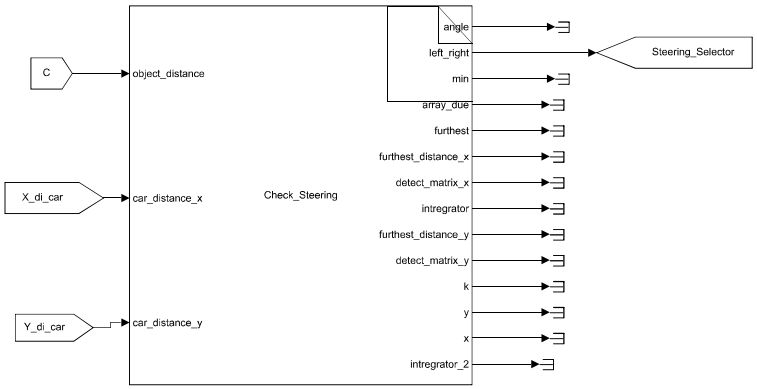
[10] “ControlDesk.” *dSPACE*, https://www.dspace.com/en/inc/home/products/sw/experimentandvisualization/controldesk.cfm. Accessed 04 January 2023.

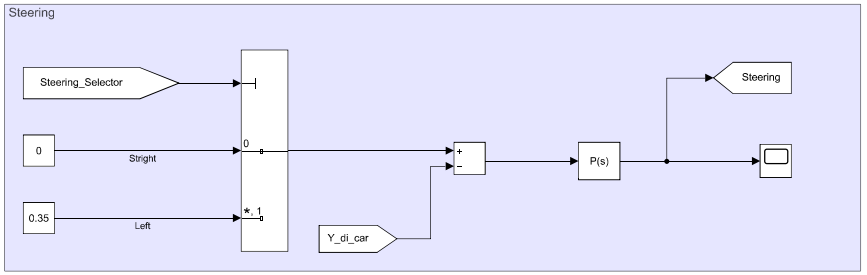
[11] “DC Gear Motor 12V 250RPM - SGM25-370.” *Makerlab Electronics*, https://www.makerlab-electronics.com/product/dc-gear-motor-12v-250rpm-sgm25-370/. Accessed 10 January 2023.

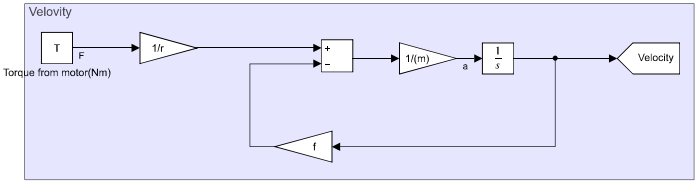
[12] “L298N Dual H Bridge Motor Driver. Spark Fun.” *L298N Dual H Bridge Motor Driver*, http://www.robotpark.com/L298N-Dual-H-Bridge-Motor-Driver. Accessed 02 January 2023.

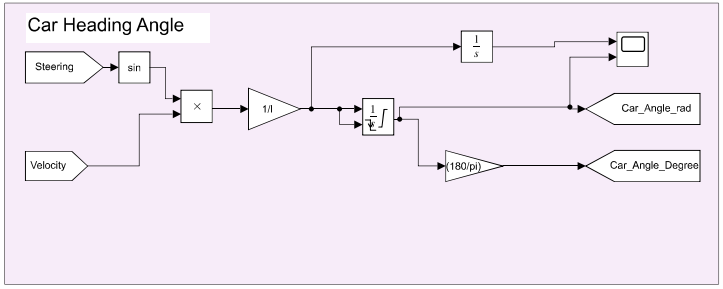
[13] “HS-311 Servo-Stock Rotation.” *ServoCity*, https://www.servocity.com/hs-311-servo. Accessed 02 January 2023.
[14] “MAX3232 - RS232 to TTL Serial Port Converter Module.” *Components101*, 24 June 2021, https://components101.com/modules/max3232-rs232-to-ttl-serial-port-converter-module. Accessed 03 January 2023.
[15] “MicroAutoBox Hardware - MicroAutoBox III.” *dSPACE*, https://www.dspace.com/en/inc/home/products/hw/micautob/microautobox3.cfm#175_50060. Accessed 05 October 2022.P
[16] “Ackerman Steering • Computer Science and Machine Learning.” *xarg.org*, https://www.xarg.org/book/kinematics/ackerman-steering/. Accessed 15 March 2021.
[17] “Newton's Laws - Lesson 3 - Newton's Second Law of Motion.” *The Physics Classroom*, https://www.physicsclassroom.com/class/newtlaws/Lesson-3/Newton-s-Second-Law. Accessed 10 March 2021.
[18] “RPLidar A2 Low Cost 360 Degree Laser Range Scanner.Interface Protocol and Application.” *RPLIDAR*, https://bucket-download.slamtec.com/6494fd238cf5e0d881f56d914c6d1f355c0f582a/LR001_SLAMTEC_rplidar_protocol_v2.4_en.pdf. Accessed 05 December 2022.
driving-car-uberdeath-woman-failure-fatal-crash-arizon. Accessed 15 December 2022.
[19] “dSpace AutoBox III.” *dSpace*, 26 September 2019, http://www.dspace.com/en/pub/home/support/documenta%20tion.cfm?hlevel2=%20MicroAutoBoxIII&hlevel1=DSHardware. Accessed 02 January 2023.

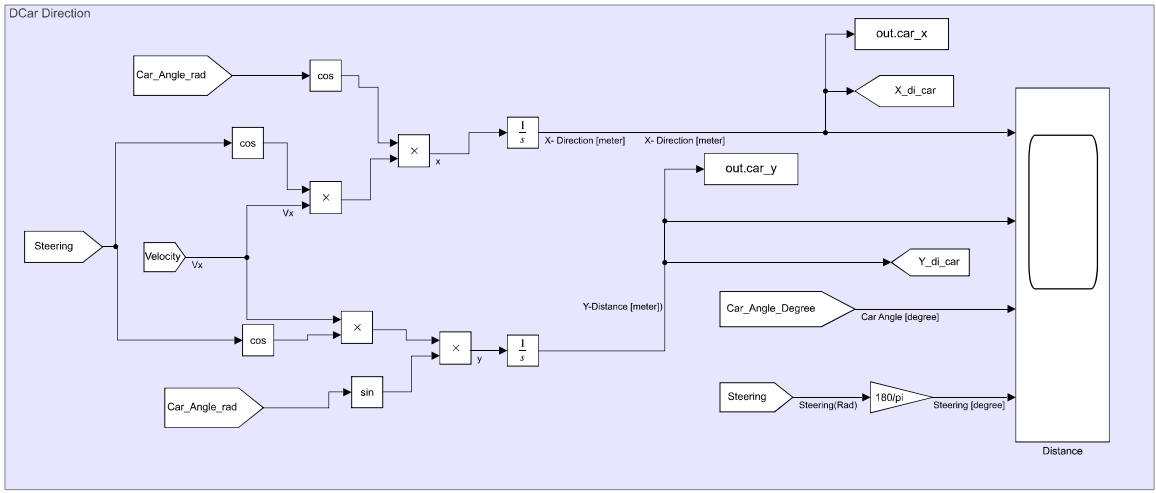

out.angle
-C-
angle/distance
1/64
uint16
Angle
Builder_
Array_Effective
C
1/4000
double
Distance
out.distance

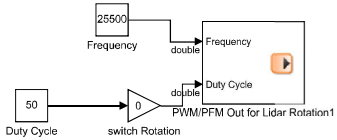

C
object_distance
X_di_car
car_distance_x
Check_Steering
Y_di_car
car_distance_y
angle
left_right
Steering_Selector
min
array_due
furthest
furthest_distance_x
detect_matrix_x
intregrator
furthest_distance_y
detect_matrix_y
k
y
x
intregrator_2

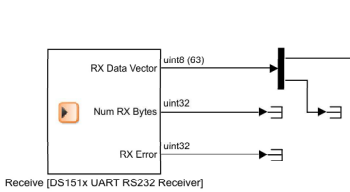

Steering
Steering_Selector
0
Stright
0.35
Left
Y_di_car
P(s)
Steering

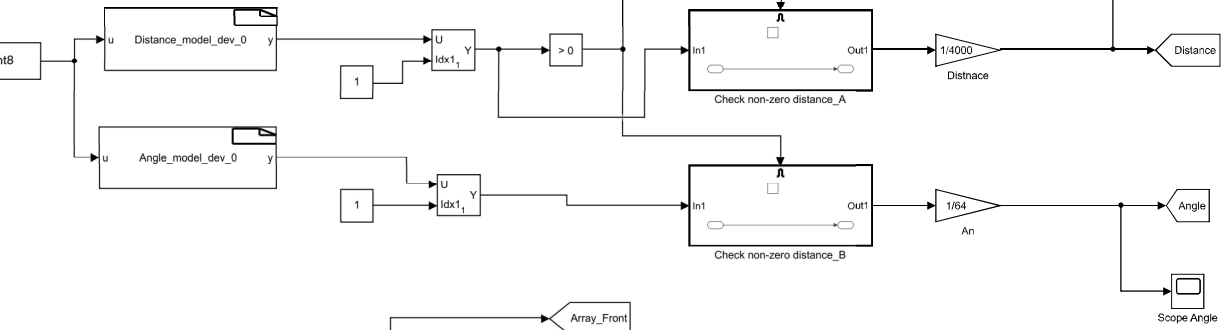

Velovity
T
F
Torque from motor(Nm)
1/r
1/(m)
a
Velocity
f

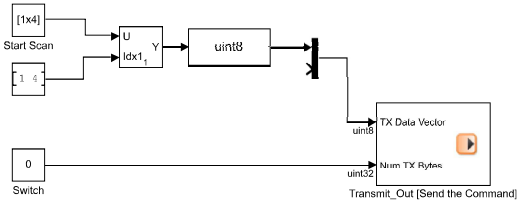

Car Heading Angle
Steering
sin
1/I
Velocity
Car_Angle_rad
(180/pi)
Car_Angle_Degree

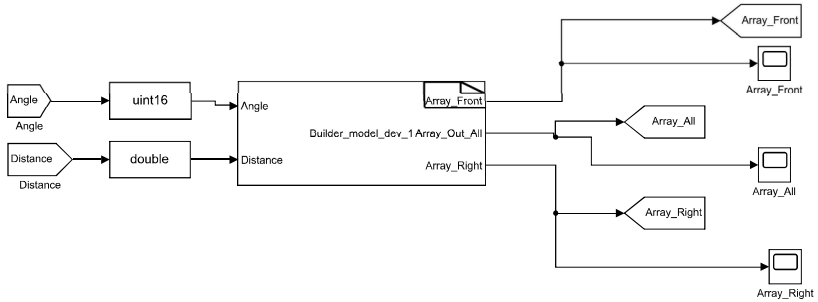

DCar Direction
Car_Angle_rad
cos
X- Direction [meter]
out.car_x
X_di_car
out.car_y
Steering
Velocity
Vx
Y_di_car
Y-Distance [meter]
Car_Angle_Degree
Car Angle [degree]
sin
Steering(Rad)
180/pi
Steering [degree]
Distance

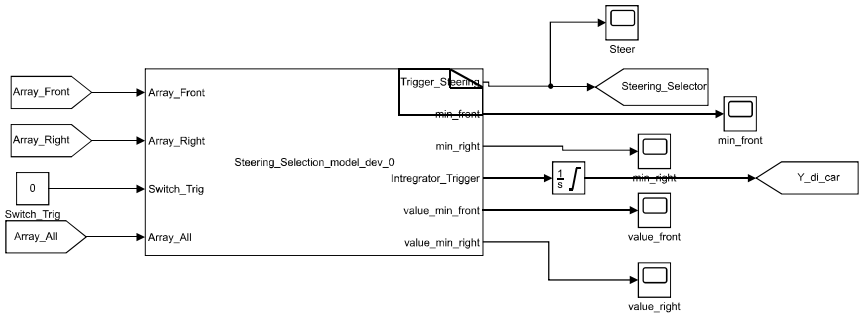

25500
Frequency
double
Frequency
Duty Cycle
50
Duty Cycle
0
switch Rotation
PWM/PFM Out for Lidar Rotation1

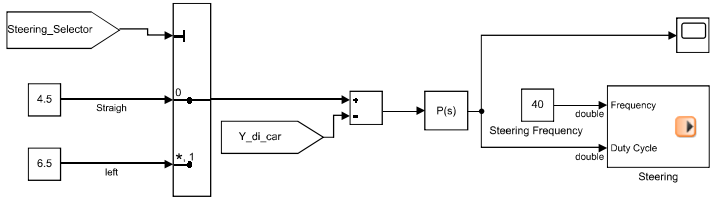

RX Data Vector
Num RX Bytes
RX Error
uint8 (63)
uint32
uint32
Receive [DS151x UART RS232 Receiver]


uint8
u
Distance_model_dev_0
y
u
Angle_model_dev_0
y
1
1
U
Idx1
Y
> 0
In1
Out1
Check non-zero distance_A
Check non-zero distance_B
1/4000
Distnace
1/64
An
Distance
Angle
Scope Distance
Scope Angle


[1x4]
Start Scan
U
Idx1
Y
uint8
0
Switch
TX Data Vector
uint8
Num TX Bytes
uint32
Transmit_Out [Send the Command]


Angle
Angle
uint16
Distance
Distance
double
Angle
Distance
Array_Front
Builder_model_dev_1
Array_Out_All
Array_Right
Array_Front
Array_Front
Array_All
Array_All
Array_Right
Array_Right


Array_Front
Array_Right
0
Switch_Trig
Array_All
Array_Front
Array_Right
Switch_Trig
Array_All
Steering_Selection_model_dev_0
Trigger_Steering
min_front
min_right
Integrator_Trigger
value_min_front
value_min_right
Steer
Steering_Selector
min_front
min_right
value_front
value_right
Y_di_car


Steering_Selector
4.5
Straigh
6.5
left
0
*_1
Y_di_car
P(s)
40
Steering Frequency
Frequency
Duty Cycle
double
double
Steering