



The University of
Nottingham

UNITED KINGDOM • CHINA • MALAYSIA

**DEPARTMENT OF ELECTRICAL AND
ELECTRONIC ENGINEERING**

**The Design, Implementation and Evaluation of an
Autonomous Indoor Vehicle with 2D-Mapping**

AUTHOR	Mr Zixiang Lu
SUPERVISOR	Dr Steve Bull
MODERATOR	Dr Christian Klumpner
DATE	18 May 2018

Table of Contents

Abstract	3
Chapter 1: Introduction and Background	4
1.1 Introduction to Autonomous Vehicles.....	4
1.2 Project Aim and Objectives.....	7
1.3 Report Structure.....	9
Chapter 2: Hardware Infrastructure	10
2.1 Mechanical Structures	10
2.2 Electrical Infrastructure.....	10
2.3 Microcontrollers.....	16
2.4 Power Supplies	16
2.5 Vehicle Appearance	17
Chapter 3: Sensing Methodology	19
3.1 Sensor Selections	19
3.2 Sensor and Controller Communication	20
3.3 Practical Performance of the Selected Sensors.....	20
Chapter 4: Control System and Algorithm	26
4.1 Vehicle Control with Integrated Sensors	26
4.2 Hazard Prevention	30
Chapter 5: Localization and Mapping Methodology	31
5.1 Global Coordinate System	31
5.2 Mapping Obstacles	32
5.3 Vehicle Position Logging.....	33
5.4 Converting Local Coordinates to Global Coordinates	35
5.5 Route Planning.....	39
5.6 Overall Mapping Algorithm	42
Chapter 6: Results and Evaluation	44
6.1 Static Mapping.....	44
6.2 Dynamic Mapping.....	45
6.3 Mapping with Route Planning.....	47
Chapter 7: Improvements and Future Work	48
7.1 Real time map display.....	48
7.2 Video Streaming.....	49
7.3 Future Work.....	49
Chapter 8: Project Review and Conclusion	52
8.1 Achieved Objectives and Deliverables	52
8.2 Attempted Ideas	52
8.3 Time and Risk Management.....	53
8.4 Project Conclusions.....	54
References	55
Appendix	56

Abstract

Autonomous vehicles contributing to a more intelligent world aiming to release the burden of human have been developed rapidly conducted by numerous companies in recent years. Various technologies are required to integrate on this highly advanced product, while the fundamental technological issue to resolve is self-navigation which request an efficient mapping and localizing methodology. This project aims to develop an indoor experimental autonomous vehicle with 2D mapping and other advanced features. This vehicle is based on an off-the-shelf mobile robot platform with Arduino and Raspberry Pi microcontrollers as the controlling core. The software algorithms and their implementations in terms of mapping and localizing are emphasized in this project and the hardware infrastructures enabling this vehicle are also demonstrated in detail. This thesis reports the entire developing process in all aspects including hardware and software implementations, results and evaluation, as well as improvements and conclusions.

Key Words: Autonomous Vehicle, Indoor Mapping and Localizing, Arduino, Raspberry Pi

Chapter 1: Introduction and Background

This chapter presents a basic introduction to this project including a brief background of autonomous vehicles, a demonstration of the project aim and objectives as well as the structure of this thesis.

1.1 Introduction to Autonomous Vehicles

In the era of 21st century, the trend of technological development tends to be concentrated on autonomous control technologies to satisfy the demand in products that would bring more convenience into human's daily life. Autonomous vehicle is one of the popular research topics in this emerging field, particularly under high investment from a large number of IT and automotive companies. From the latest development progress, the major prospects of the autonomous vehicle are these three following applications: self-driving car, home robotics, and planet exploration rover. However, each of these applications has a common fundamental requirement for mapping and localizing to enable other more advanced technologies (route planning and self-driving).

1.1.1 Self-driving Car

Automobiles are the most popular private transportation and also the most convenient and comfortable point-to-point commuting method to some extent. However, conventional automotive companies has been impacted by the emerging technology of autonomous driving and job of professional drivers is being replaced by intelligent programs. Due to the reasons of traffic safety, economy, efficiency and environmental-friendly [1], the bright future of self-driving vehicles have been predicted by numerous companies and they are aiming at producing a revolution of car industry as well as road traffics.

Up to the year 2018, Google is accelerating the autonomous vehicle industry with their advanced self-driving car project Waymo. This project was formerly named "Google Self-driving Car" and it is now individually operated in order to minimize the time for bringing self-driving vehicles into commercially used [2]. Figure 1.1 shows the two latest version of their self-driving products: Firefly (left) and Chrysler Pacifica Hybrid (right). Apart from Google, there are also many other enterprises including Tesla, Uber, Baidu and Toyota that target at this high-tech industry and make efforts to get autonomous vehicles popularized.



Figure 1.1, Pictures of Google Waymo Self-driving Cars [2]

Driving is not a very high level intelligence required task if taking a deeper look in this activity. Without too many creativities, driving is a routine work containing actions like accelerating, braking, reversing and making turns. Hence autonomous driving is very

feasible to implement by machines to some extent, however, what makes this so difficult to achieve is the complexity and uncertainty of the environment to deal with. Which means that the detail of the environment have to be collected correctly to enable proper reactions be made especially in emergencies. Accordingly, to drive safely and flexibly, autonomous vehicles require an accurate mapping of the surroundings and reliable algorithms to make reasonable decisions. Figure 1.2 shows plenty of sensors on a typical autonomous vehicle [3].

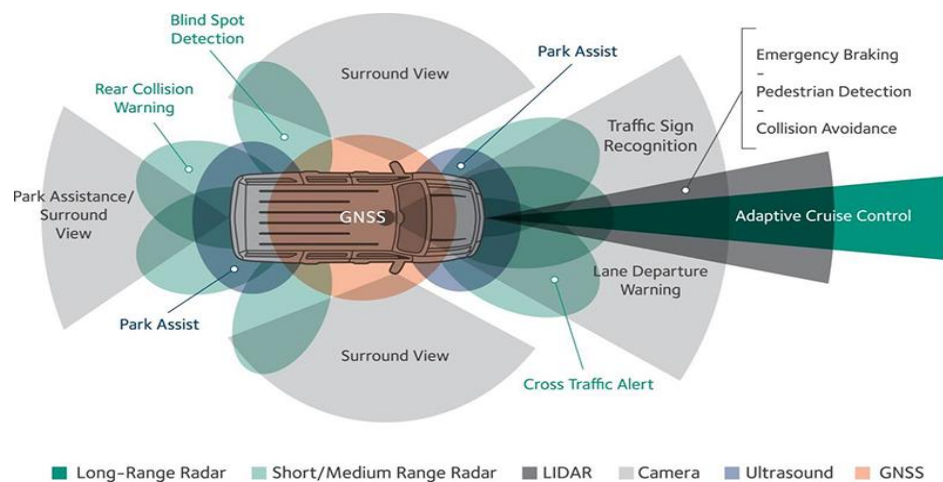


Figure 1.2, Plenty of sensors on an autonomous vehicle [3]

Different companies have different designs on their autonomous products based on different autonomy levels. In general, the levels of autonomy are divided into 5 stages as shown in Figure 1.3 [4]:

- Level 1: The vehicle can either control the steering wheel OR the pedals.
- Level 2: The vehicle is able to control both the wheel AND the pedals but only under partial conditions.
- Level 3: The vehicle can be fully controlled by the system under most of the conditions but requires human as fallback.
- Level 4: The system completely control the vehicle without any human intervention under most of the conditions.
- Level 5: The system is capable of fully governing the vehicle without fallbacks under any road conditions.

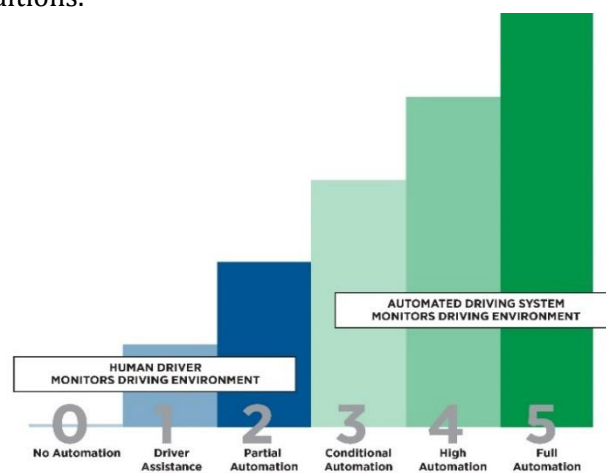


Figure 1.3, Autonomy Levels proposed by SAE [4]

1.1.2 Home robotics

With the increasing demand in smart home and the rapid growth of Internet of Things (IoT), intelligent robots that can assist householders to enjoy a more convenient life is another popular application of autonomous vehicles. Being no longer just amusements for children, home robotics have more functional tasks to deal with including floor cleaning, health care and security protection. Integrating with IR sensors, LiDAR, sonar and cameras, these mobile robots navigate itself in the rooms and make decisions to complete specific jobs based on its program [5].

Floor cleaning robots are currently the most typical autonomous home robotics. Up to the year 2018, the competition between robotic vacuum has become extremely fierce. Numerous technological companies have already released their products and Xiaomi Mi Robot has been gaining a popularity on its cost-effective. Pricing at 200 pound, this latest version of robotic vacuum has all the functions of congeneric products which cost 300 pound more [5]. Figure 1.4 shows the appearance of a Mi robot. Xiaomi applies the advanced SLAM algorithm enabling its robot to navigate itself in an indoor environment and produce a map as a reference for the cleaning plan.

Other household robots like “Kuri” (shown in Figure 1.5) from Mayfield Robotics automatically filming home-made videos is another potential market in home robotics. In addition, security robots “K3” from Knightscope which has the ability to patrol around the house with 24/7 video monitoring will probably be the next generation of security force.



Figure 1.4, Image of a Xiaomi Mi robot [6]



Figure1.5, Image of a Kuri Home Robot [7]

1.1.3 Planet Exploration Rover

Not only because of curiosity, exploring planets outside the earth has more benefits in scientific research. However, sending human directly to the planet is a high risk task and requires extremely high investment. Instead, employing an autonomous vehicle to the planet is a relatively safe and manageable operation. Up till now, human has successfully sent 6 exploration rover to Mars and two of them “Opportunity” and “Curiosity” are still in action [8].

Autonomous navigation and driving is essential tasks for the rover to reduce the burden of human operator, since real-time controlling the vehicle is impossible due to the 13-minute signal transmission delay on Mars. The auto-navigation system on “Opportunity” rover is based on two stereo cameras mounted on the body to produce a 3-D map of the surroundings. Evaluating the nearby terrains, “Opportunity” (shown in Figure 1.6)

simulates multiple possible routes to the specified end point and selects the optimized one in terms of safety and efficiency. Then it moves one step (0.5~2 meters) closer to the goal along the selected path until it finally reach the end point. [9]

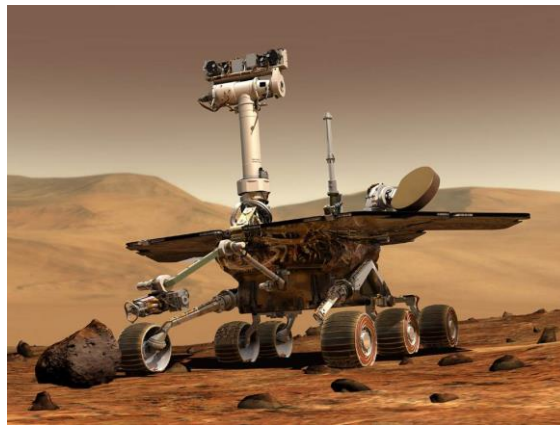


Figure 1.6, Image of the Opportunity Mars Exploration Rover travelling on Mars[9]

1.2 Project Aim and Objectives

1.2.1 Project aim

The aim of this project is to construct and develop a mini experimental vehicle that is capable of autonomously mapping the layout of simple spaces. Through this project, the quintessence of autonomous systems would be acquired and the methodologies of solving electronic engineering problems would be familiarized.

1.2.2 Project objectives

From low to high levels, five general objectives in order to achieve the project aim are outlined below. Each of these objectives is split into more detailed tasks to step by step fulfill the corresponding objective. Several tasks that marked with [Advanced] are for additional research which would not affect the accomplishment of objectives.

1. Review the state-of-the-art on autonomous systems and relevant emerging applications.
 - Study the technological and commercial prospects of the development of autonomous systems.
 - Research the underlying theories and relevant technologies applied on a typical autonomous system.
 - Investigate the development and implementation of the most recent applications of autonomous systems for civilian use (e.g. home robotics, self-driving vehicles...)
2. Construct a robotic vehicle platform that can perform basic movements as directed by the user implementing key commands on an Arduino microprocessor.
 - Assemble the provided off-the-shelf kits to build up a mini vehicle platform.
 - Investigate the rationale for H-bridge and servo motors, then integrate them on

- the constructed platform.
- Familiarize the structure, variables and special functions in Arduino programming language, and produce simple codes that select and control the desired motor to perform forward and reverse rotation.
 - Test the vehicle under the control of Arduino to achieve forward, backward, left and right movements, then combine these actions to follow predefined routes.
 - [Advanced] Research deeper on the Arduino board (e.g. I/O ports, system frequencies, timers, PWM modules...) as well as the implementation of PID control algorithm on microcontrollers in order to accurately regulate the velocity, turning angle and displacement of the vehicle.
3. Incorporate sensors onto the vehicle platform and interface them with the Arduino in order that the vehicle can 'sense' its environment and log its position as it moves.
 - Research the available Arduino sensors including sharp IR distance sensors, ultrasonic range measurement module, crash sensors and gyroscopes to familiarize their behavior, applications, usability and cost.
 - Determine appropriate sensors that would be integrated on the vehicle and develop relevant codes that allow Arduino to manage them.
 - Design, in both hardware and software, a user interface containing an LCD screen and buttons (or joystick if necessary) that provides man-machine interaction to command the robot or read internal data in real time.
 - Develop algorithms that enable the vehicle equipped with sensors to automatically avoid obstacles, hitting the wall or dropping from height, then test them in an open space with its position data recorded.
 4. Implement appropriate control algorithms so that the vehicle can autonomously map out the extent and major features of simple spaces.
 - Customize a coordinate system to record the track of the vehicle.
 - Develop algorithms that initially detect the surroundings to predict the optimum starting point of mapping then find and follow the edges of the wall and finally travel back to the starting point.
 - Record the entire traveling track of the vehicle (in the form of discrete coordinates), and sketch it on the coordinate system to obtain the map of this room.
 - [Advanced] Explore various techniques of SLAM and compare between their feasibility and efficiency.
 5. Implement a vision system that allows the vehicle to capture images as it autonomously maps out a space.
 - Research on the Raspberry PI microcontroller and its camera module to apply a visual system on the vehicle.
 - Develop codes to enable the Raspberry PI camera and capture a 360° image to map out the room.
 - [Advanced] Apply image recognition on Raspberry PI to autonomously recognise the feature of the space.

1.3 Report Structure

This report has been separated into 8 Chapters according to the development details, result evaluations and conclusions. The basic structure of this report is listed below:

- Chapter 1: A brief background introduction of autonomous vehicles and the overview of the entire project
- Chapter 2: The detail design of the hardware infrastructure that enables the basic movement of the vehicle
- Chapter 3: The demonstration of the issues related to the sensors integrated on the vehicle and the discussion of the practical performance of the sensors.
- Chapter 4: The demonstration of the control algorithms applied on the vehicle to prepare for mapping and prevent from hazard.
- Chapter 5: The detailed illustration of the mapping and localization methodology in terms of theoretical derivation and practical implementation
- Chapter 6: Discussion of the mapping results in terms of different scenario
- Chapter 7: Demonstration of the improvements made on the vehicle as well as the future work
- Chapter 8: Review on the entire project and the conclusion of this project.

Chapter 2: Hardware Infrastructure

2.1 Mechanical Structures

The premier step is to construct a physical platform for the autonomous vehicle. The mechanical structure plays a vital role so that selecting a proper vehicle platform is very essential. Since the emphasis of this project is concentrated on the software part, self-designing a hardware platform is not the case of this project. Therefore, a pre-designed off-the-shelf robot platform kit is preferred for this project.

Considering with the complexity of control, a two-wheel mobile platform (Figure 2.1) is selected as the framework of the autonomous mapping vehicle which is able to perform flexible movements. Basically, this two-layer platform with plenty of holes is consisted of two controllable wheels and an omnidirectional caster. Driven by two conventional DC motors, the vehicle is able to perform zero radius turning by using motor speed differential control.

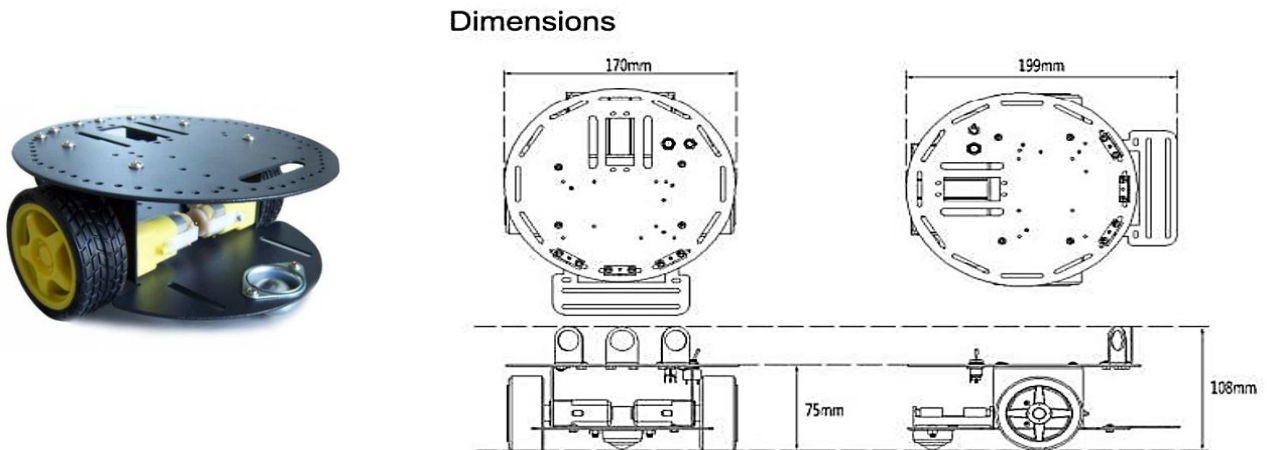


Figure 2.1, Two-wheel vehicle platform image (Left) and dimensions (right) [10]

2.2 Electrical Infrastructure

2.2.1 DC Motor

The drives provided by the kits are the most common low-voltage DC motors. This type of motor has an advantage of easy to control. The speed of the motor is directly proportional to the current flowing through this motor, and the current is also directly proportional to the applied voltage for a fix load and a fix electrical resistance. This implies that controlling the speed of the motor can be achieved by controlling the applied voltage which is a relatively easy task for microcontrollers.

From the datasheet, the rated voltage of these motors is between 3V~6V, the no-load speed is between 100RPM~200RPM and the no-load current is between 60mA~71mA.

However, due to the manufacturing error of these two motors, it is observed that there is a resistance difference between the two motors which produces a different current (and hence motor speed) under the same applied voltage.

To further illustrate this issue, the electrical behaviours of the DC motors are tested and recorded below:

Electrical Behaviours of the DC motors

Table 2.1, Actual resistance of the steering motors

	Left Steering Motor	Right Steering Motor
Resistance	2.7 Ω	3.3 Ω

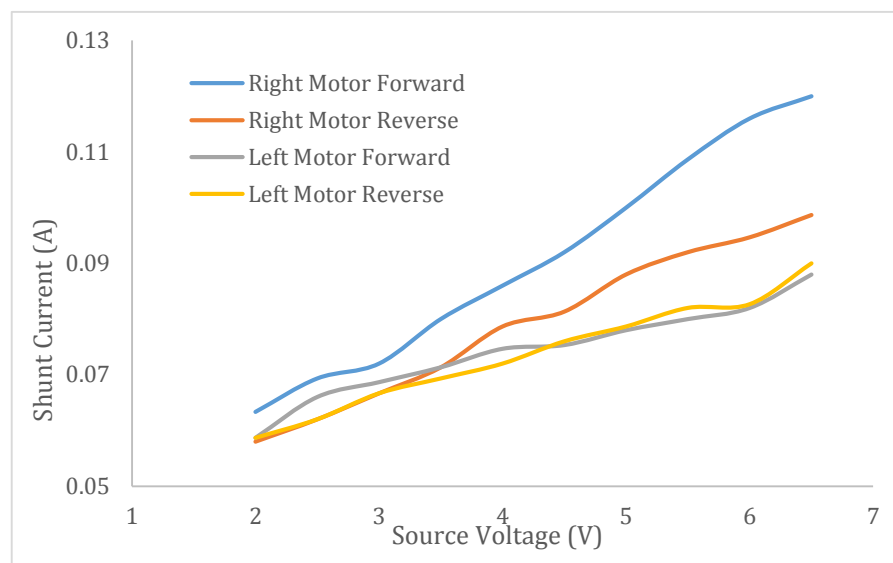


Figure 2.2, Motor Current vs. Supply Voltage relationship

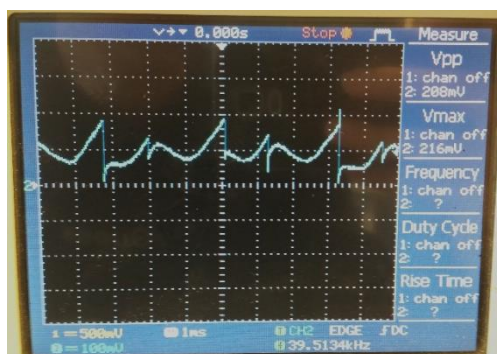


Figure 2.3, DC Motor Current Waveform
(Source Voltage = 3V)

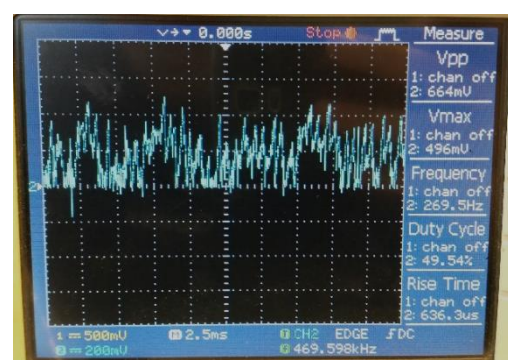


Figure 2.4, DC Motor Current Waveform
(Source Voltage = 6V)

Although this type of DC motors are extremely easy to control, the apparent drawback of the motor is lacking of precision so that it increases the variance of control output. One of the solution to this problem is to calibrate the motors' resistances by adding a

potentiometer in serial to balance their resistance. However, in practical applications the mechanical load of the motor is not constant due to various reasons (e.g. different frictions) and consequently the adjusted resistance could be unbalance again. Another more accurate solution is to add feedback sensors and real time examine the state of the motor to perform a close-loop control. The detailed sensing methods and control algorithms are demonstrated in Chapter 3 and Chapter 4 respectively.

The relationship between the motor rotatory direction and the resulting movement are summarized at Table 2.2 below:

Table 2.2, Vehicle Movement and Motor Rotary Direction Relationship

Left Motor Direction	Right Motor Direction	Vehicle Movement		
		$v_L = v_R$	$v_L < v_R$	$v_L > v_R$
S	S	Stop (SS)	n/a	n/a
S	F	n/a	Left Turn with Forward (LT-F)	n/a
S	R	n/a	Left Turn with Backward (LT-B)	n/a
F	S	n/a	n/a	Right Turn with Forward (RT-F)
R	S	n/a	n/a	Right Turn with Backward (RT-B)
F	F	Straight Forward (FF)	Forward with Left Turn (F-LT)	Forward with Right Turn (F-RT)
R	R	Straight Backward (BB)	Backward with Left Turn (B-LT)	Backward with Right Turn (B-RT)
R	F	Left Yaw (LY)	Left Yaw with Forward (LY-F)	Left Yaw with Backward (LY-B)
F	R	Right Yaw (RY)	Right Yaw with Forward (RY-F)	Right Yaw with Backward (RY-B)

To better demonstrate the traveling directions of the vehicle, Figure 2.5 is included below to illustrate the basic vehicle movements:

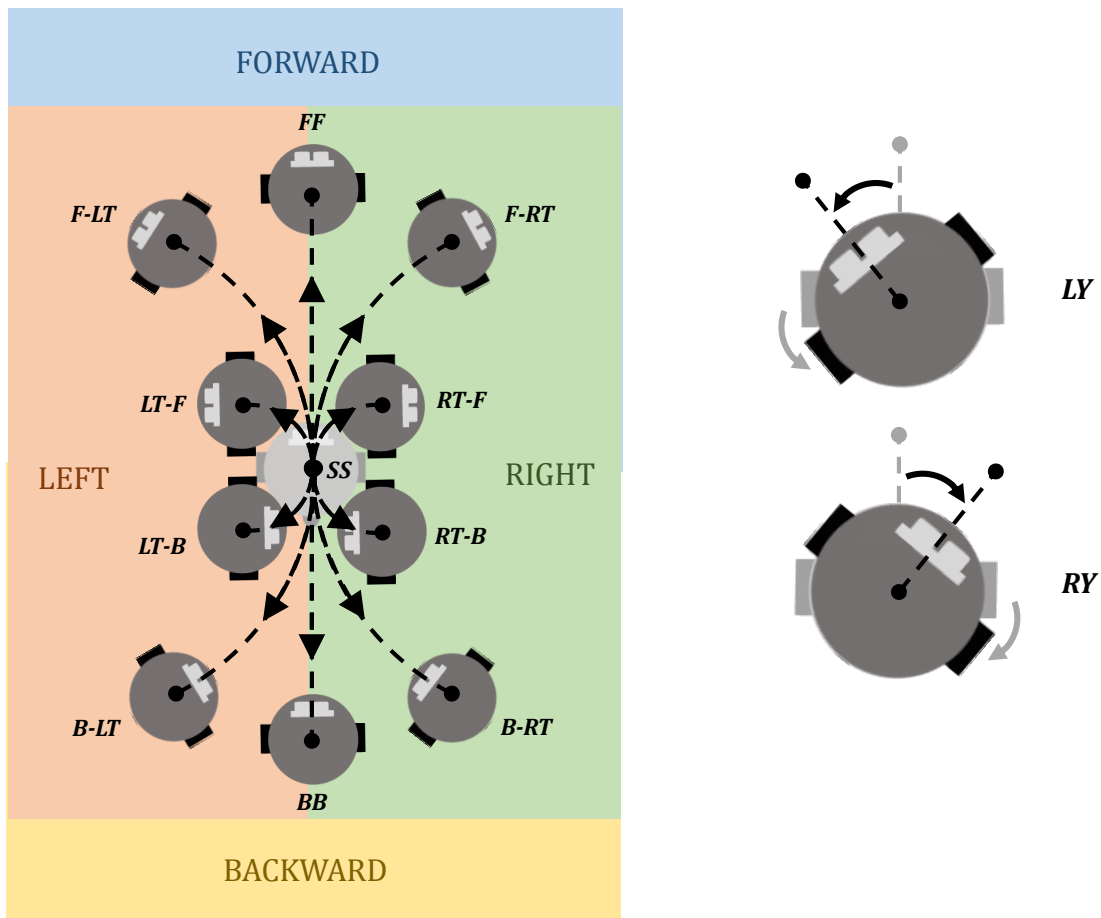


Figure 2.5, Basic movements of the vehicle

2.2.2 Driver H-Bridge

To enable this two-wheel vehicle with omnidirectional movement (i.e. forward, backward, left turn, right turn, left yaw and right yaw), each of the DC motor have to be controlled individually with forward and reverse rotation. In order to perform a separate control for each of the motors, two DC-motor H-bridge drivers are constructed to accomplish this task. Basically, this driver is consisted of two identical DC H-bridges with PWM input control. For a faster switching with low impedance, these H-bridges are designed based on MOSFETs. The source voltage is selected as 6V with the purpose to enable the motors to gain high power. The schematic of Driver H-bridge is shown in Figure 2.6 below.

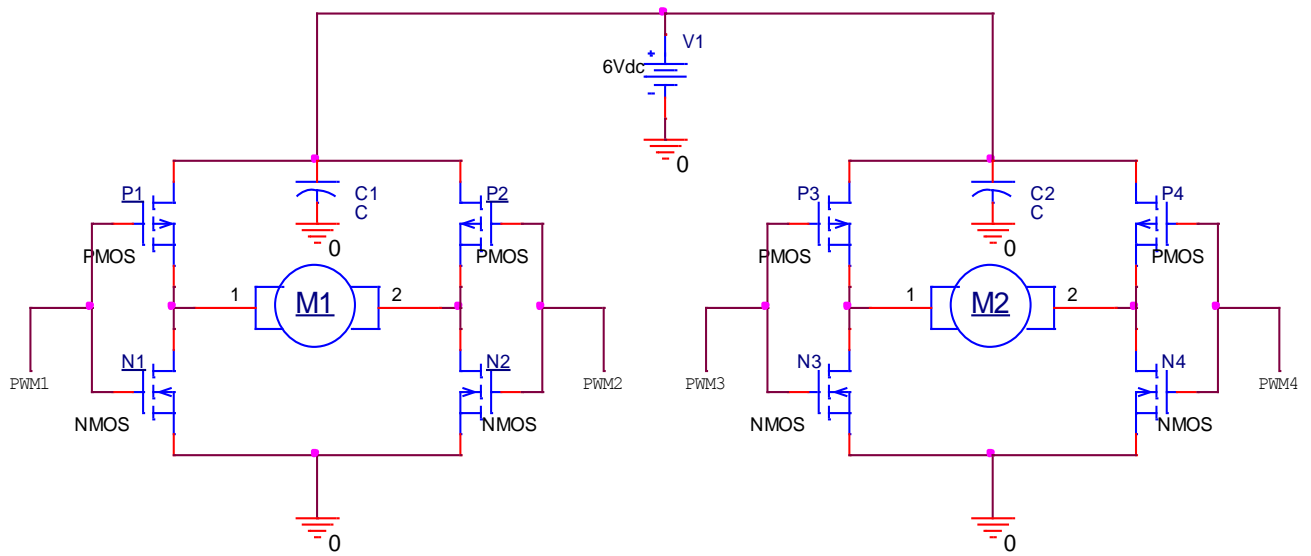


Figure 2.6, Schematic Diagram of H-Bridge

The direction of the motor is controlled by the combinational state of PWM1 and PWM2 (or PWM3 and PWM4).

Table 2.3, Truth table of M1 H-bridge

PWM1	PWM2	P1	N1	P2	N2	Current Direction	M1 Direction
0	0	ON	OFF	ON	OFF	P1 to P2 (No ground)	S
0	1	ON	OFF	OFF	ON	P1 to N2 (Positive current)	F
1	0	OFF	ON	ON	OFF	P2 to N1 (Negative Current)	R
1	1	OFF	ON	OFF	ON	N1 to N2 (No source)	S

[Note: the positive current is defined as the current flowing from pin 1 to pin 2 of the motor]

(Since the H-bridge for M1 is identical to that of M2, the truth table of M2 H-bridge is similar to above)

Combining the individual control of the steering motors, this two wheel robot is able to perform an omnidirectional movement without difficulty. Notably, the vehicle can achieve zero turning radius when taking a left or right yaw.

Table 2.3, Truth Table of Vehicle Movement

Left Motor Control Pulse		Right Motor Control Pulse		Left Motor Direction	Right Motor Direction	Vehicle Movement
PWM1	PWM2	PWM3	PWM4			
0	0	0	0	S	S	Stop
0	0	0	1		F	Left Turn
0	0	1	0		R	Left Turn
0	0	1	1		S	Stop
0	1	0	0	F	S	Right Turn
0	1	0	1		F	Forward
0	1	1	0		R	Right Yaw
0	1	1	1		S	Right Turn
1	0	0	0	R	S	Right Turn
1	0	0	1		F	Left Yaw
1	0	1	0		R	Backward
1	0	1	1		S	Right Turn
1	1	0	0	S	S	Stop
1	1	0	1		F	Left Turn
1	1	1	0		R	Left Turn
1	1	1	1		S	Stop

From the above tables, the desired movements of the vehicle can be conveniently control by the combination of input PWM pulses. This can be easily achieved by microcontrollers with multiple PWM modules (e.g. Arduino UNO)

2.2.3 Stepper Motor Behaviour

Stepper Motor that is able to produce an accurate step angle is selected to carry the ultrasonic sensor for mapping the environment. The performance of the stepper motor are tested in terms of the parameters below:

Table 1, Stepper Motor Behaviour

Time of one step (us)	Measured Period	Measured RPM
700 (Minimum)	3.10s	19.35
800	3.63s	16.53
900	3.97s	15.11
1000	4.34s	13.82
1100	4.78s	12.55
1200	5.13s	11.69

2.3 Microcontrollers

Microcontrollers are the core of autonomous vehicles. With dedicated program running on these processors, the vehicle is able to be self-controlled without any human intervention. In this project, Arduino UNO (Figure 2.7) and Raspberry Pi 3 Model B+ (Figure 2.8) are selected as the brain of the robot. The Arduino is a development-friendly platform with extensive open source codes. Equipped with plenty of digital and analog I/O and multiple individual PWM modules, the Arduino is ideal for controlling the hardware components like actuators and sensors. The Raspberry Pi is a compact but powerful single-chip CPU board that is suitable for commanding the vehicle as well as processing the collected data.

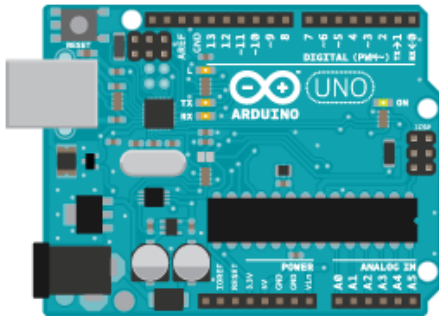


Figure 2.7, Arduino Board Layout [11]

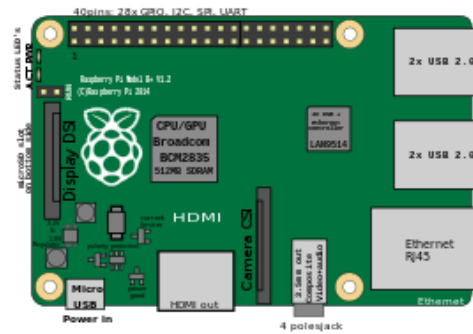


Figure 2.8, Raspberry Pi 3B+ Board Layout [12]

2.4 Power Supplies

The required supply voltage and power density of each of the components are listed below:

Table 2, Power Requirements of Vehicle Components

Components	Supply Voltage	Power Density
Arduino	5V	Low
Raspberry Pi	5V	Low
H-bridge Driver (Steering Motors)	6V	High
Stepper Motors	6V	High

In order to maximize the mobility of the autonomous vehicle, the power supplies for this project are based on batteries. Taking into account the required supply voltage, the required power density, the price and the weigh, a 6V NiMH AA battery pack and a 5V Li-ion power bank are selected as the power supplies.

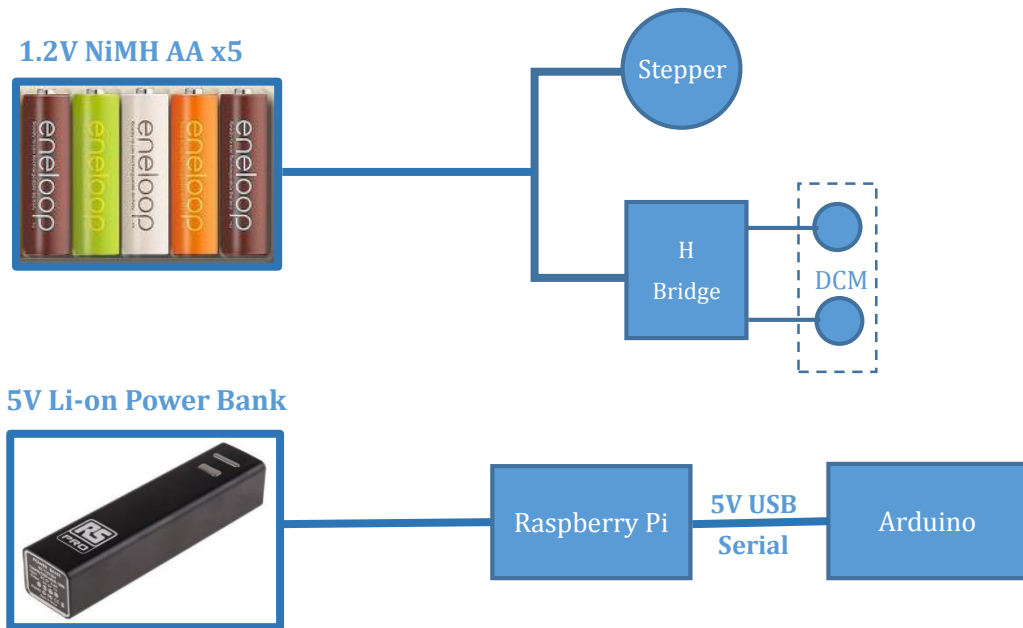


Figure 2.9, Power Supply Connections

The parameters of the batteries are given below:

Table 3, Battery Parameters

	Voltage	Capacity	Output Current
NiMH AA pack	$1.2V \times 5 = 6V$	1900mAh	>1.5A
Power Bank	5V	2200mAh	1A

2.5 Vehicle Appearance

The front, rear and top view of the vehicle are included below:

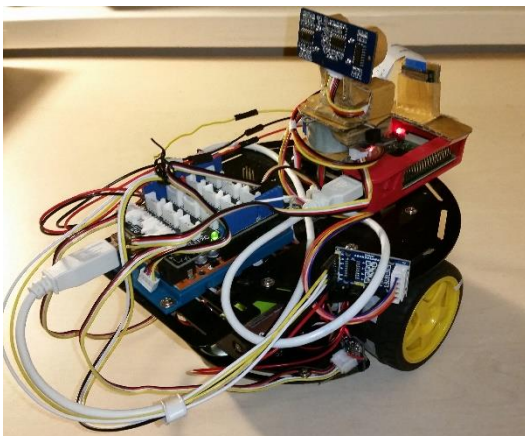


Figure 2.10, Vehicle Overview

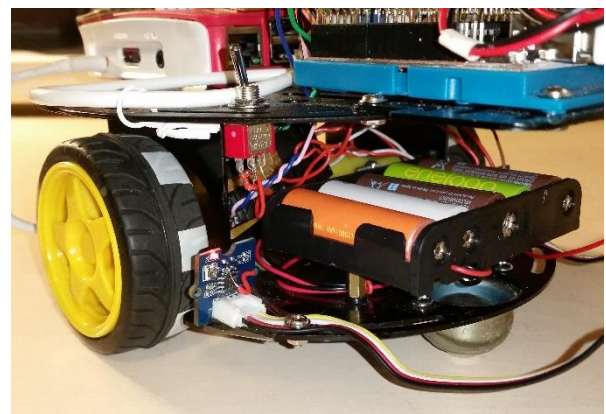


Figure 2.11, Vehicle Rear View

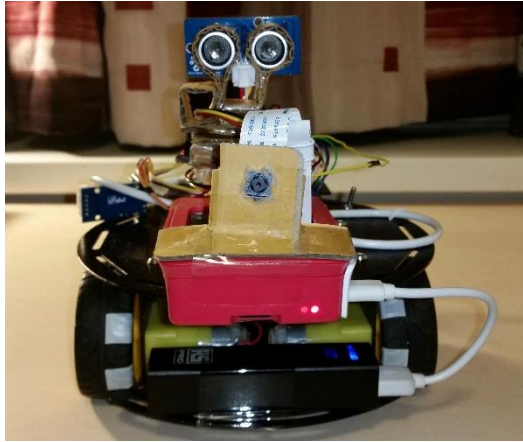


Figure 2.12, Vehicle Front View

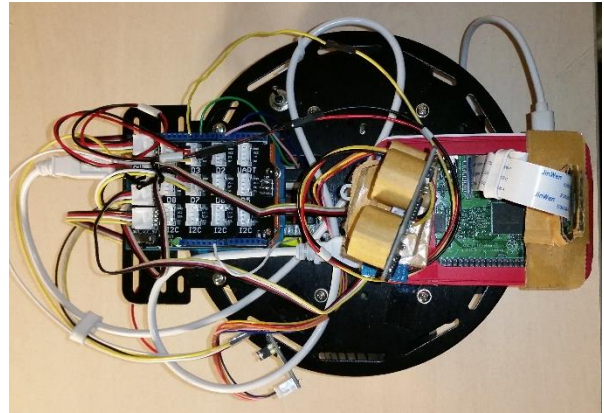


Figure 2.13, Vehicle Top View






Chapter 3: Sensing Methodology

3.1 Sensor Selections

To collect information from the environment as well as the vehicle itself, sensors are essential components to be integrated onto the platform. With the aids of sensors, the vehicle is able to measure the traveling distance, the turning angle as well as detect obstacles and measure the distance to them.

The candidates of sensor selections and their potential usage are summarized below:

Table 4, Sensor Selection Listing [13]

Sensor Type	Model	Image	Features	Purpose
Ultrasonic Sensor	Grove - Ultrasonic Ranger v2.0 SKU 101020010		<ul style="list-style-type: none"> ▪ Detecting Range: 3cm – 350cm ▪ Detecting angle limit: 30 ° 	<ul style="list-style-type: none"> ▪ Obstacle Detection ▪ Range Measurement
IR Distance Sensor	Grove - IR Distance Interrupter v1.2 SKU 101020175		<ul style="list-style-type: none"> ▪ Adjustable Detecting Range: 9cm – 30cm 	<ul style="list-style-type: none"> ▪ Obstacle Detection ▪ Table Edge Detection
IR Reflective Sensor	Grove - Infrared Reflective Sensor v1.2 SKU 101020174		<ul style="list-style-type: none"> ▪ Adjustable Detecting Range: 0.4cm – 1.5cm 	<ul style="list-style-type: none"> ▪ Wheel Rotary Detection (Colour Contrast Detection)
Digital Gyroscope	Grove - 3-Axis Digital Gyro (ITG 3200) SKU 101020050		<ul style="list-style-type: none"> ▪ I2C Communication ▪ Sensitivity: 14.375 LSBs per °/s 	<ul style="list-style-type: none"> ▪ Turning Angle Measurement
Digital Accelerometer	Grove - 3-Axis Digital Accelerometer (±16g) SKU 101020054		<ul style="list-style-type: none"> ▪ I2C Communication ▪ Sensitivity: 3.9mg/LSB 	<ul style="list-style-type: none"> ▪ Gyro Calibration

3.2 Sensor and Controller Communication

There are multiple communication methods between the microcontroller and the sensors including Analog I/O, Digital I/O and I2C.

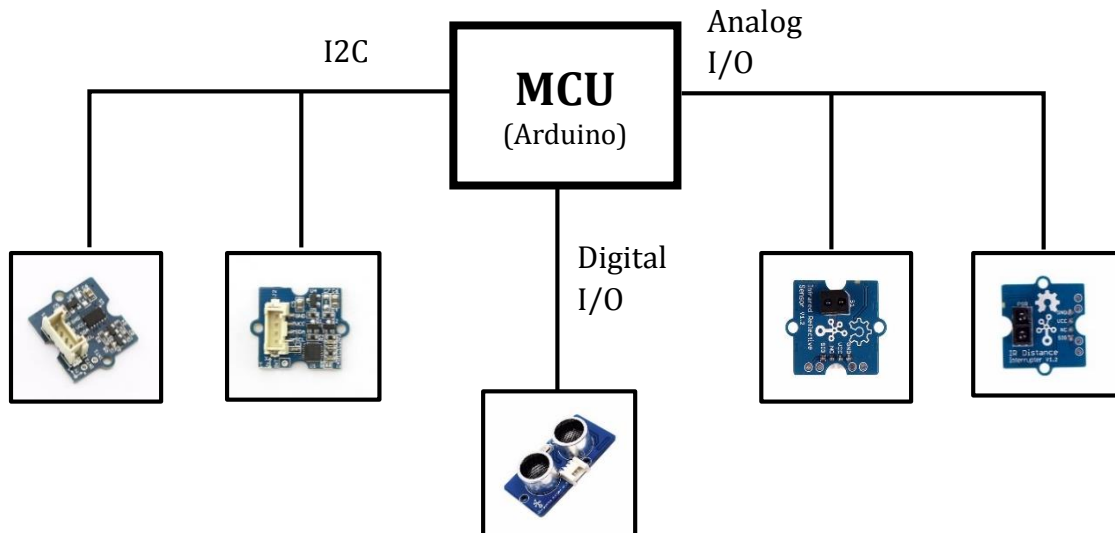


Figure 3.1, Sensor and Controller Communication Diagram

3.3 Practical Performance of the Selected Sensors

3.3.1 Ultrasonic Ranger

a) Distance limit

The ultrasonic range finder is tested to have a distance limit of 293cm. If the obstacle is close to or beyond this range, the reading from the ultrasonic sensor is unstable and incorrect. This is mainly due to the travel time of the ultrasonic wave exceeding the maximum counting time of the sensor's timer. However, this limit hardly affects this project, since this autonomous mapping vehicle is designed for indoor scenarios that do not require a long distance.

b) Angle limit

Angle limit is a significant factor that affects the performance of the ultrasonic sensor and hence the mapping results. If the ultrasonic wave from the sensor does not inject to the surface of the object at a normal incidence, there is a possibility for the ultrasonic wave to be reflected.

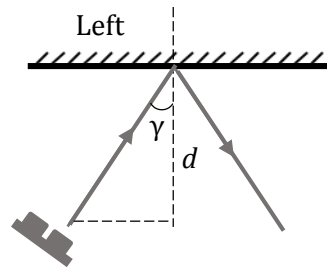


Figure 3.2, Ultrasonic Signal Reflection (Left)

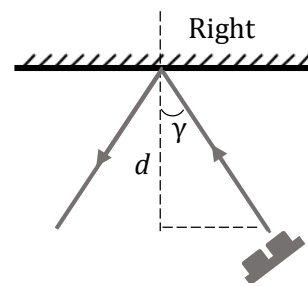


Figure 3.3, Ultrasonic Signal Reflection (Right)

The angle limit is related to the distance (d) between the surface and the sensor. Placing on the left and then on the right of the surface, the angle limits are examined separately according to different distances.

Table 5, Ultrasonic Angle Limit and Distance Relationship

Distance (d)	Angle Limit (γ)	
	Left	Right
10cm	40°	40°
20cm	30°	40°
40cm	25°	35°
60cm	25°	30°
100cm	25°	30°

From the above results, the angle limit is generally decreased as the distance increases, because a larger distance would produce a larger attenuation of the ultrasonic signal and reduce the possibility to receive the signal at a large angle γ . In addition, it can be observed that, at the same distance, the angle limit is smaller when the sensor is placed on the left of the surface comparing to that on the right.

c) Different Materials:

Material is another potential factor affecting the performance of the sensor. Six different materials are tested in terms of 3 distance levels.

Table 6, Different Material Testing

Actual Range (cm)	Test Range (cm)					
	Paper	Plastic	Concrete	Fabric	Metal	Wood
20	20	20	20	20	20	20
50	51	50	51	54	51	52
100	105	102	106	105	103	107

Generally from the above data, the measured range has a slight variation under different test materials and plastic is the material that produce the least effects. Additionally, the experimental results are accurate when the range is short, but as the range increases the accuracy of the measurement is decreased.

d) Corner reflection

Due to the angle limit of the ultrasonic sensor when detecting a surface, there is a large error if statically mapping against a wall. Moreover, an additional signal reflection exists if mapping against a corner (two perpendicular walls) and hence the error will be enlarged. So the following experiments are performed to examine this issue.

The vehicle is placed at 9 different locations against a corner which are classified as short range ($\approx 20\text{cm}$), middle range ($\approx 60\text{cm}$) and long range ($\approx 110\text{cm}$) from the corner. Each range levels have 3 different relative locations to the corner $\theta = 45^\circ$, $\theta > 45^\circ$, $\theta < 45^\circ$ (where, θ is the angle between the vehicle and Wall 1).

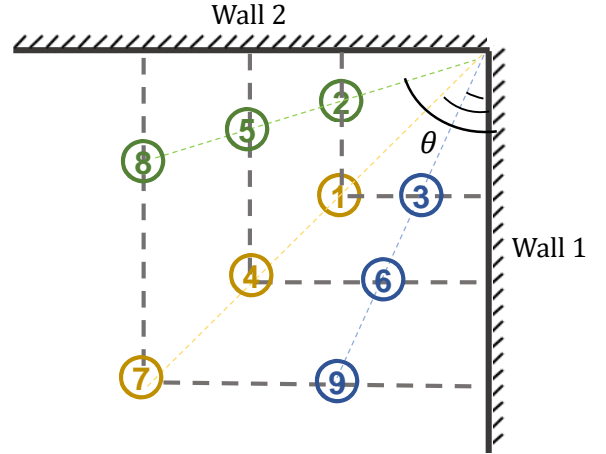


Figure 3.4, Schematic of Vehicle Positions against the Corner

[red dots: vehicle location; blue dots: collected point; grey line: real location of the walls]

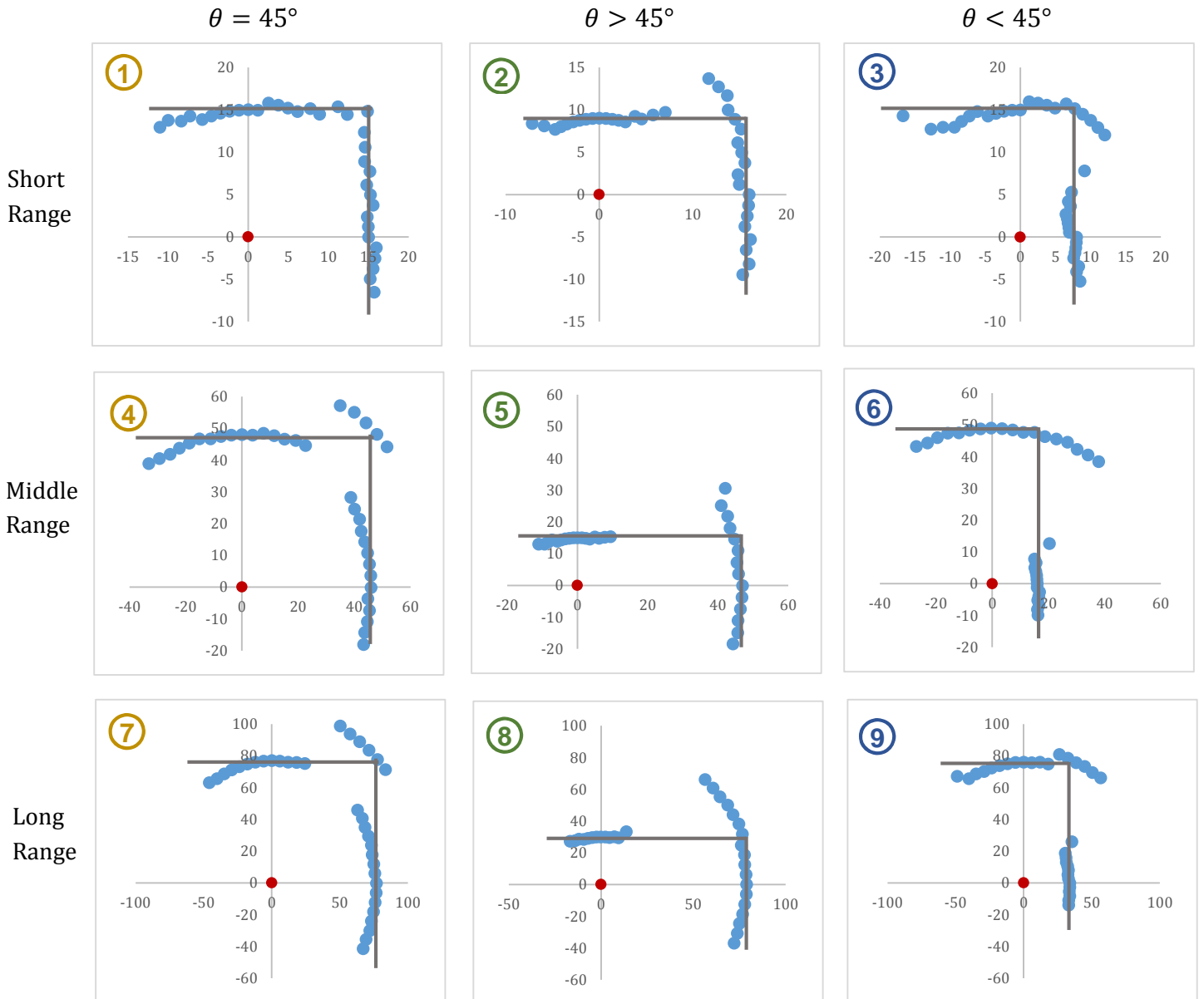


Figure 3.5, Corner reflection results

Conclusions:

- i) The shorter range to the corner, the larger accuracy the mapping can provide. Because the ultrasonic sensor has larger angle limits at shorter ranges.
- ii) When $\theta = 45^\circ$, the total accuracy of the mapping is the largest. When $\theta > 45^\circ$, the accuracy of Wall 1 is the largest. When $\theta < 45^\circ$, the accuracy of Wall 2 is the largest.
- iii) The measured wall is a curve rather than a straight line especially when the range is large. Because the precision of the ultrasonic sensor is 1cm, and every difference smaller than 1cm cannot be recognized. So when the difference of the range is very close the measured ranges are the same and forms a curve when plotting on the rectangular coordinate system (in Chapter 5).

3.3.2 IR reflective sensor

The IR reflective sensor is used to differentiate black and white marks (colour contrast detection) for vehicle's wheel odometry which is discussed in Chapter 4. The original IR sensor board is designed for digital I/O (HIGH for black and LOW for white), however, this could be failed when the sensor is extremely close to the wheel and the results are also very unstable when the source voltage is fluctuate. Therefore, a better solution to improve the results is to directly measure and compare the voltage of the IR sensor output with analog I/O (5 available pins on Arduino UNO). This is a software amendable method that can adapt to any application scenarios.

However, the challenge of using analog I/O for communication is to deal with the noise of the sensor. Because of the inaccurate of the analog ADC on the Arduino, the resulting output waveform is full of spiky noise which may affect the detection. Which means that downward spike at the black mark could be detected as a white mark (and vice-versa). Thus filtering (or debouncing) the output waveforms are essential in this application and more details are included in Chapter 4.

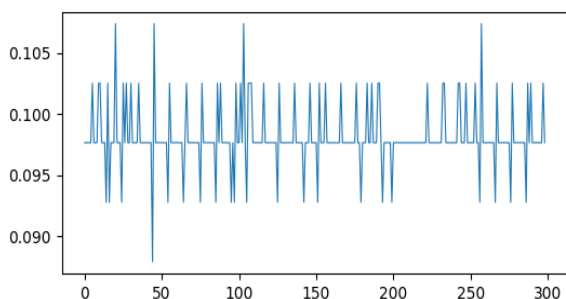


Figure 3.6, IR output waveform when Black detected

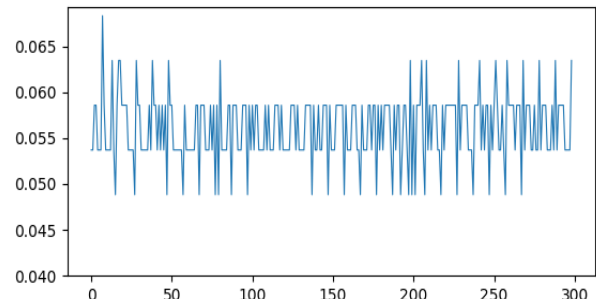


Figure 3.7, IR output waveform when white detected

3.3.3 Gyroscope and Accelerometer

a) Gyro Drift

Since the gyroscope is measuring the angle velocity of rotation (ω), the rotary angle (θ) have to be calculated by an integration of the angle velocity (ω) over the sample time (Δt).

$$\theta = \int_0^T \omega \cdot \Delta t$$

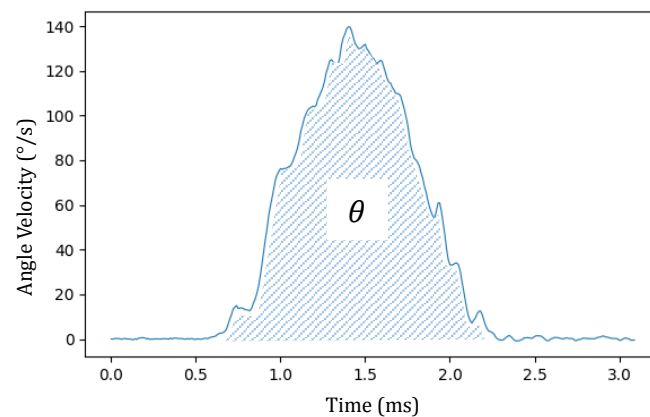


Figure 3.8, Gyroscope output waveform

However, a small error of the measurement could be accumulated and amplified due to the integration under a long period of time. It is a common issue in most of the gyros and this phenomena is called the 'Drift' of gyroscope. The problem can be minimized by a complementary filter with accelerometer (in Chapter 4).

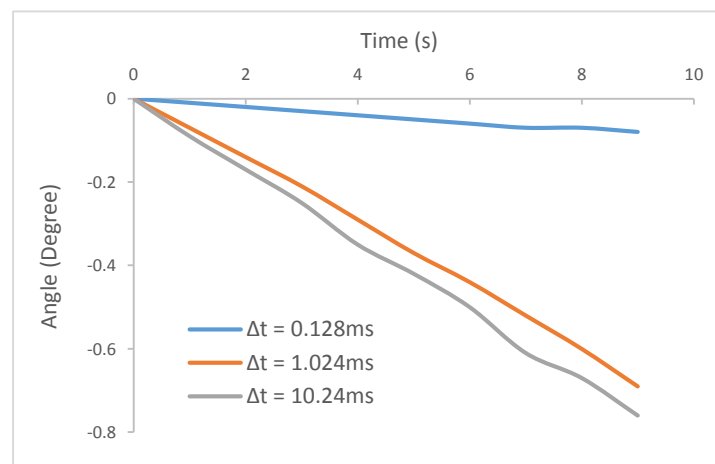


Figure 3.9, Gyroscope Drift Experiment

As shown from the experimental results, a smaller sample causes a larger drift of the gyroscope because errors are accumulated more frequently.

b) Accelerometer Noise

The high sensitivity of the accelerometer produces a large amount of noise even there is no movement applying on the sensor. This is a significant effect that disables the accelerometer to directly measure the rotary angle. A complementary filter with the measurement of gyroscope is possible to solve this problem (in Chapter 4).

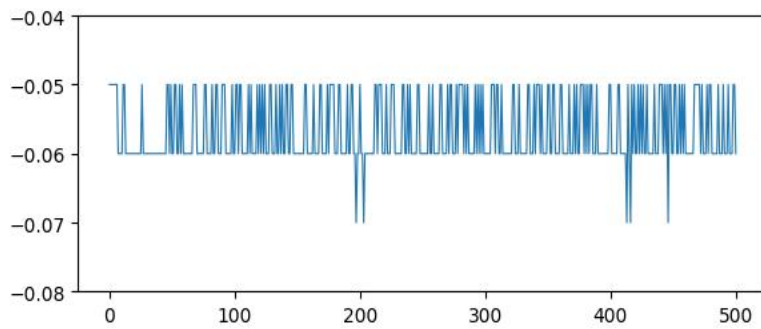


Figure 3.10, Accelerometer Noisy Waveform

Chapter 4: Control System and Algorithm

4.1 Vehicle Control with Integrated Sensors

The internal measurements of the vehicle that is essential for mapping is the travelling distance and turning angle which assist localizing and provide references for the map.

4.4.1 Vehicle Cruise Distance Measurement

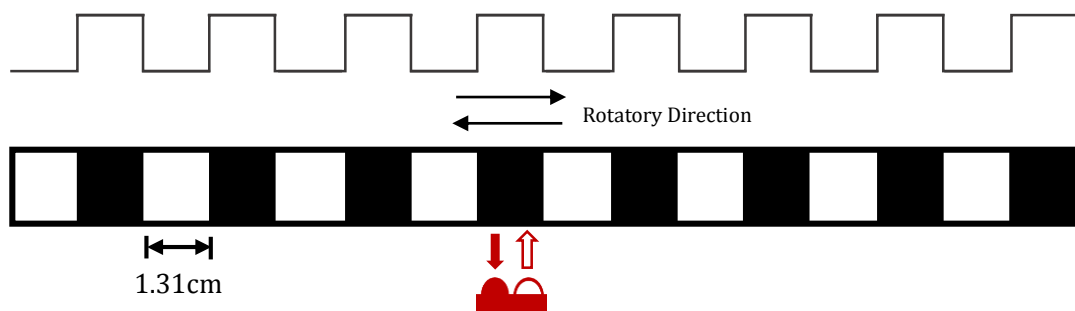
a) Wheel Odometry

As the steering motors on this autonomous vehicle are conventional DC motors without position feedbacks, a physical method is designed to obtain the wheels' rotary information. Taking advantage of the colour contrast recognition of the IR reflective sensors, each of the wheels are labeled with black and white marks with identical intervals. When switching between black and white marks, the output voltage of the IR reflective sensor experiences a sharp change indicating the transition between two marks. If the wheel is rotating, a square wave can be generated by the IR sensor to feed into the microcontroller for processing. Generally the smaller interval between two marks the greater accuracy this wheel odometer can provide. However, considering the size of the IR detector and the processing speed of the Arduino, the interval between two marks could not be too small, otherwise the IR detector may miss some marks during high speed rotation.

The circumference of the wheel is measured as 21cm and the wheel is divided by 8 pairs of black and white marks so that each mark has a width of $21/(2*8)=1.31\text{cm}$. Therefore, this wheel odometer can provide a precision up to 1.31cm which is sufficient in this project.



Figure 4.1, Black and White Marks on the Wheels



The output voltage from the IR sensor is measured by the ADC module on the Arduino UNO and the output waveform when the wheel is rotating is recorded below:

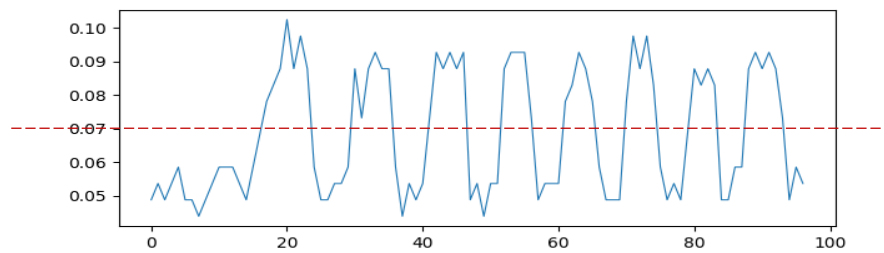


Figure 4.2, Raw waveform from IR reflective sensor

It can be observed that there are sharp transitions between black and white marks although the presence of noise distorts the waveform. However, setting a voltage reference (red dashed) to compare with the output signal can produce a perfect square wave (i.e. all the voltage larger than the reference are recognized as HIGH and all the voltage lower than the reference are recognized as LOW).

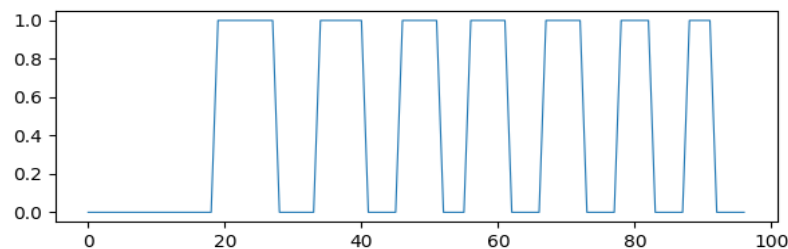


Figure 4.3, Output Square Wave after comparing with voltage reference

b) Debounce

From the above demonstration, the presence of noise may cause glitches on the output waveforms. If directly use this output for counting the black and white marks, the result is larger than the actual value because the glitches are also recognized as sharp transitions. This is the problem that similar to the button bouncing so an extra algorithm is applied to debounce the output from IR sensor. Counters are added to test the stability of a state. Only if the counter counts 5 samples remaining in the same state in a row, this state is confirmed as a stable state. Otherwise, this state is interfered by glitches and it will be neglected.

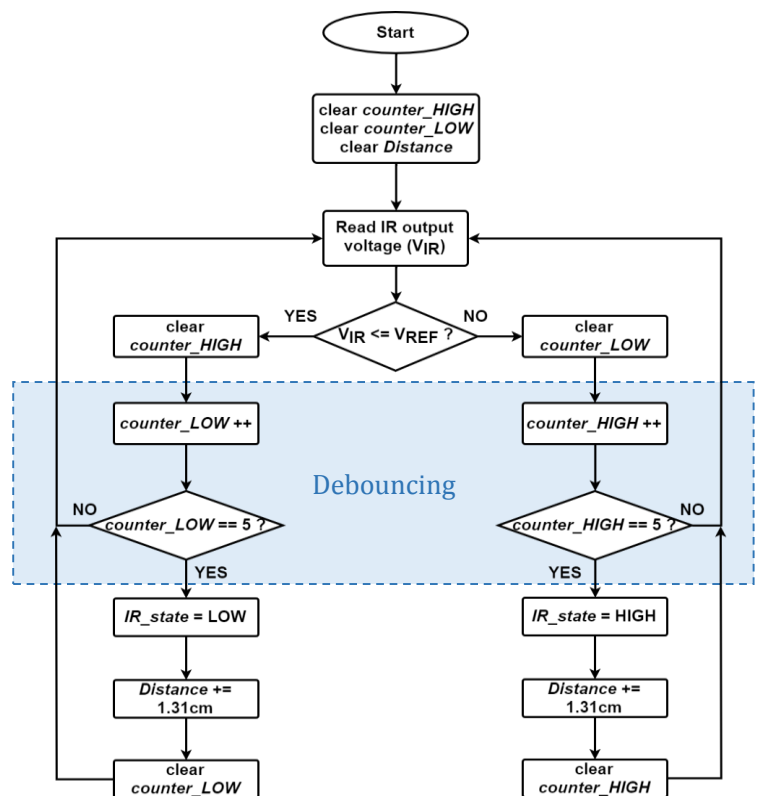


Figure 4.4, Flowchat of Signal Debouncing

4.1.2 Vehicle Yaw Angle Measurement

a) Method 1: Gyroscope and Accelerometer with Complementary Filter

As discussed in Chapter 3, there will be significant problem for individually use of gyroscope or accelerometer. So a complementary filter is suggested to improve the measurement of the vehicle's turning angle. Basically, gyroscope can only be applied in short-term (due to 'drift') and accelerometer can only be applied in long-term (due to noises). So combining short-term gyroscope readings with long-term accelerometer measurements, the equation is given below:

$$\text{angle} = 0.98 * (\text{angle} + \text{gyro}_{\text{reading}}) + 0.02 * \text{acc}_{\text{reading}}$$

The resulting complementary waveform in comparison with the raw gyro and accelerometer waveforms are sketched below:

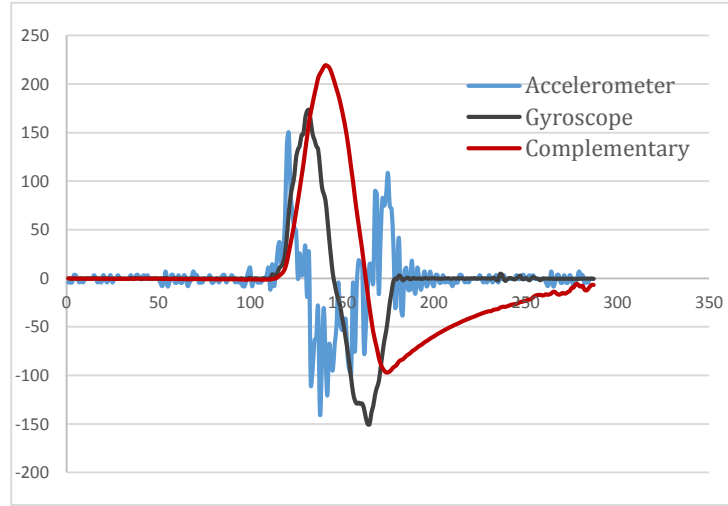


Figure 4.5, Complementary Filtered Waveform vs. Sensor Raw Waveforms

After applying the filter, the complementary waveform is clean and the effect of gyro drift is eliminated. However, it can be noticed from the graph that the complementary waveform experiences significant delays and it takes a particularly long time dragging back to zero.

b) Method 2: Geometry Calculation

The traveling distance of each wheel l_1 and l_2 can be measured by the wheel odometer discussed in 4.4.1. Taking advantage of these information, the deflection angle of the vehicle (ϕ) can be deduced by geometry calculation. Generally there are two turning methods for a two wheel:

a) Turning (identical rotary direction of both wheels)

$$\begin{aligned}
 l_1 &= \alpha \cdot r_1 \\
 l_2 &= \alpha \cdot (r_1 + L) = \alpha \cdot \left(\frac{l_1}{\alpha} + L\right) = l_1 + \alpha \cdot L \\
 \varphi_1 = \alpha &= \frac{l_2 - l_1}{L}
 \end{aligned}$$

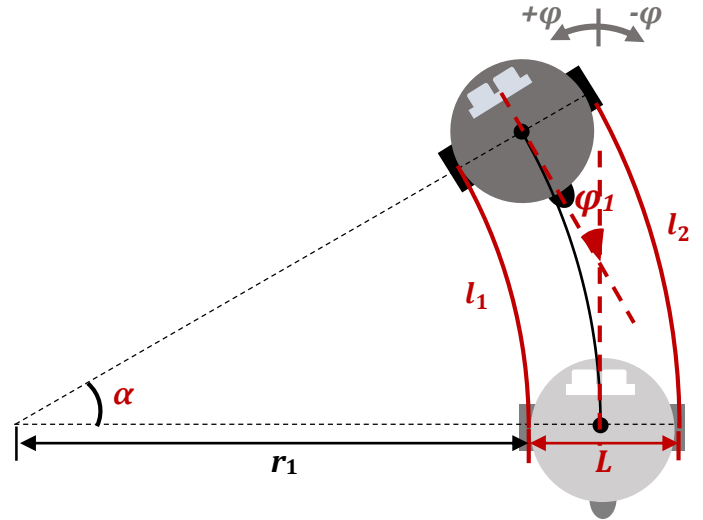


Figure 4.6, Yaw angle calculation in the case of turning

b) Yawing (different rotary direction between two wheels)

$$\begin{aligned}
 l_1 &= \alpha \cdot r_1 \\
 L &= \frac{l_1}{\alpha} + \frac{l_2}{\alpha} \\
 \varphi_2 = \alpha &= \frac{l_2 + l_1}{L}
 \end{aligned}$$

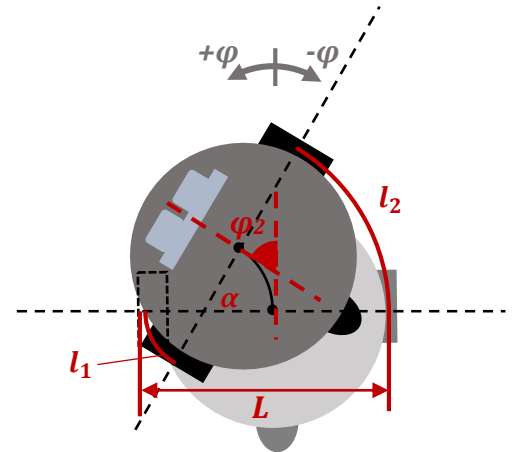


Figure 4.7, Yaw angle calculation in the case of yawing

[L is the diameter of the vehicle which is a given parameter in Chapter 2]

Furthermore, a more generic equation can be obtained if l_1 and l_2 are unsigned parameters:

$$\varphi = \alpha = \frac{l_2 - l_1}{L} \quad \left(\begin{array}{l} l_1, l_2 > 0 \text{ if forward rotation} \\ l_1, l_2 < 0 \text{ if reverse rotation} \end{array} \right)$$

4.2 Hazard Prevention

4.2.1 Obstacles Detection

The ultrasonic sensor is able to detect obstacles right in front of the vehicle with the algorithm below. By setting a distance reference D_{REF} and compare it with the measurement of the ultrasonic sensor, the vehicle can notice whether the object is close enough to block itself. Then it can take actions if it detects obstacles, for example, avoiding the obstacle, changing direction, or stop.

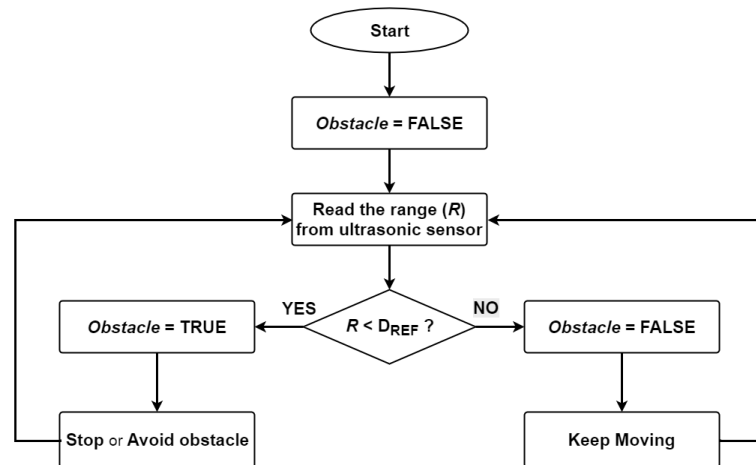


Figure 4.7, Flowchart of obstacle detection

4.2.2 Anti-dropping

An IR distance sensor is installed at the bottom front of the vehicle to detect cliffs and prevent dropping down. If there are signals received by the IR sensor (output HIGH), it indicates that the vehicle is on the ground and it is safe to move. But if the IR sensor cannot receive signals (output LOW), it implies that the ground is far away and the vehicle is beside a cliff. In this case, vehicle have to stop or reverse avoiding dropping from height.

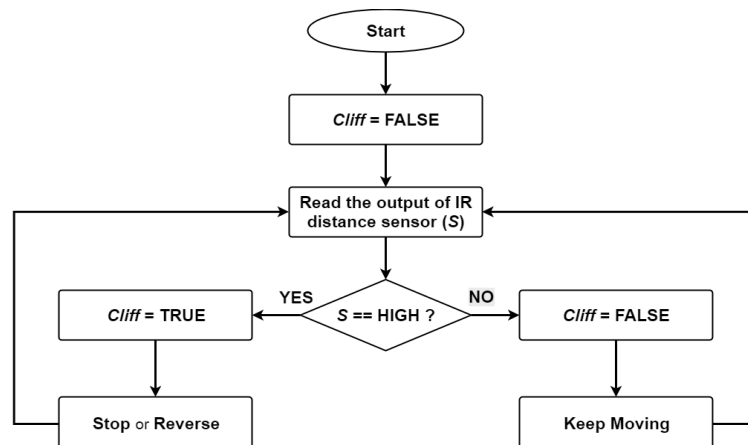


Figure 4.8, Flowchart of anti dropping

o

Chapter 5: Localization and Mapping Methodology

5.1 Global Coordinate System

The format of the map produced by the vehicle is selected to base on a Cartesian coordinate system which is visualized for readers. To produce a reference for the map, a Global coordinate system is created based on the starting location of the vehicle. All the detected obstacles as well as the updated vehicle locations are converted to point coordinates to plot on this Global coordinate system. The ultimate map is generated according to this Global coordinate system. This coordinate system is created every time when the vehicle is started or reset so the origin always represents the initial location of the vehicle. For the convenience of Trigonometric calculation in the next step, the initial orientation of the vehicle (vehicle's forward direction) is regarded as the $+x$ axes of this Global coordinate system.

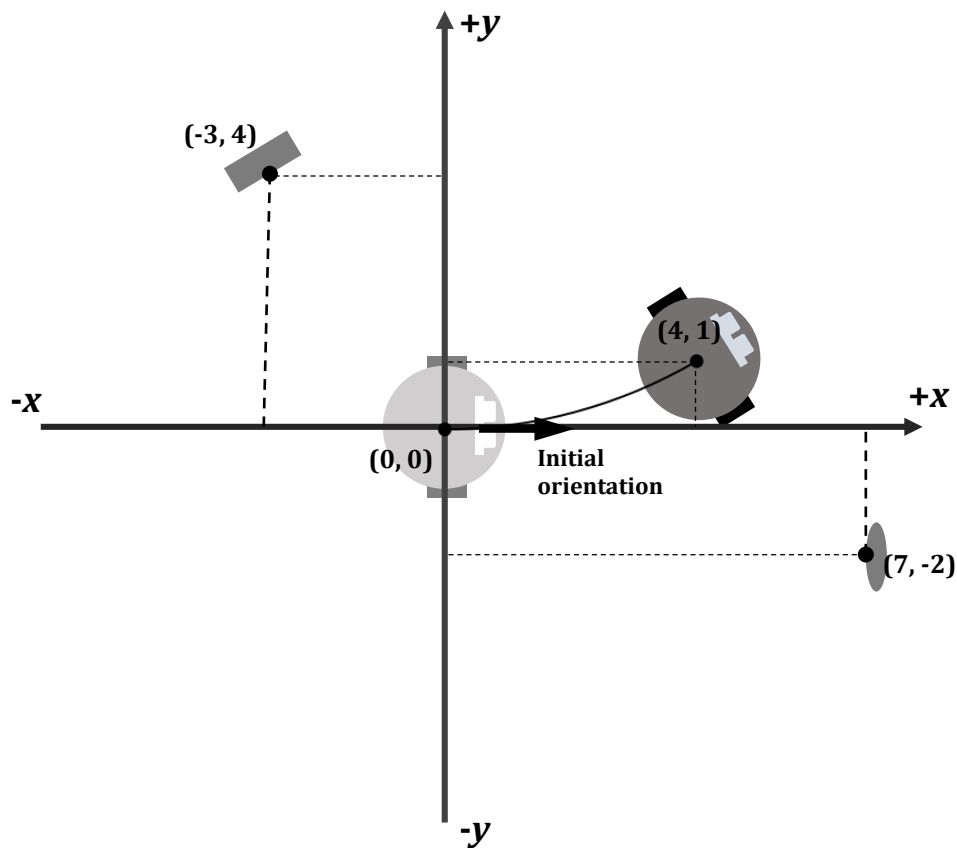


Figure 5.1, Example of a global coordinate system

Figure 5.1 shows an example of the Global coordinate system. It describes the vehicle starting at the initial location (0, 0) moves to another location (4, 1) while obstacles at the locations of (-3, 4) and (7, -2) are detected.

5.2 Mapping Obstacles

The methods of mapping the obstacles' locations with respect to the vehicle's position is based on simple trigonometric formula.

From the measurement of ultrasonic sensor, the distance between the obstacle and the vehicle (**R**) can be obtained. By counting the steps of the stepper motor, the turning angle of the ultrasonic sensor (**θ**) can also be obtained. Therefore, the point coordinate of the detected obstacle (*x, y*) can be calculated with the Trigonometric formula:

$$x = R \cdot \cos\theta, \quad y = R \cdot \sin\theta$$

Rewritten as the matrix form:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} R \cdot \cos\theta \\ R \cdot \sin\theta \end{pmatrix}$$

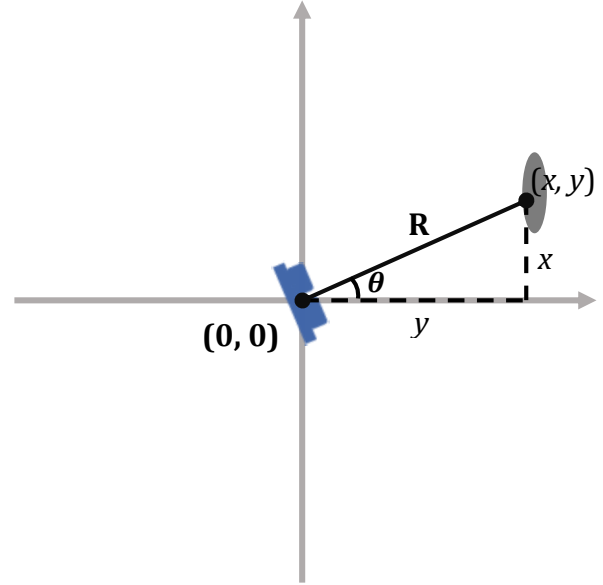


Figure 5.2, Example of trigonometric calculation

Rotating 360° by the stepper motor, the ultrasonic sensor scans one cycle and measures the distance to the obstacles [**R₁, R₂, R₃, R₄ R_n**] at each step of the stepper motor (Where, **n** is the total number of points being collected).

Since the step angle (**θ**) is a constant and **θ = 360°/n**, the conversion to point coordinates can be calculated as:

$$\begin{aligned} x_1 &= R_1 \cdot \cos\theta, & y_1 &= R_1 \cdot \sin\theta; \\ x_2 &= R_2 \cdot \cos 2\theta, & y_2 &= R_2 \cdot \sin 2\theta; \\ x_3 &= R_3 \cdot \cos 3\theta, & y_3 &= R_3 \cdot \sin 3\theta; \\ x_4 &= R_4 \cdot \cos 4\theta, & y_4 &= R_4 \cdot \sin 4\theta; \\ &\dots\dots\dots \\ x_n &= R_n \cdot \cos(n\theta), & y_n &= R_n \cdot \sin(n\theta); \end{aligned}$$

Converting to matrix form given that:

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} R_n \cdot \cos(n\theta) \\ R_n \cdot \sin(n\theta) \end{pmatrix} \text{ at } O(0,0)$$

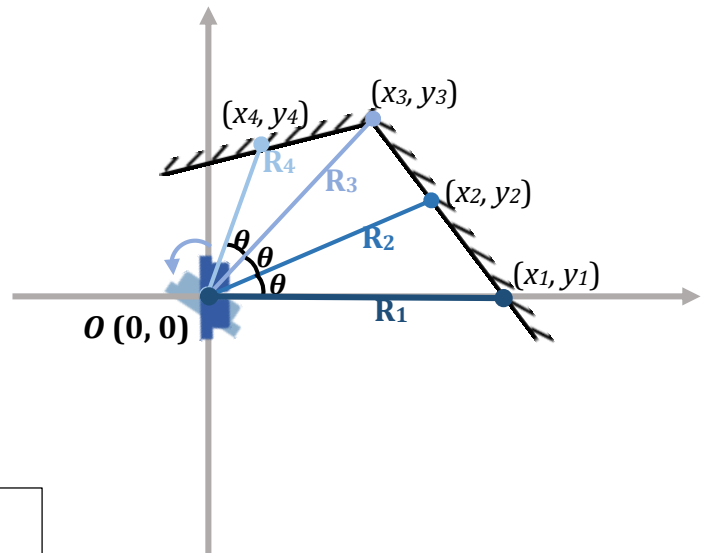


Figure 5.3, Example of obstacle mapping

Plotting these point coordinates (*x_n, y_n*) to the coordinate system, a map of the surrounding obstacles' locations relative to the vehicle's initial position (0, 0) can be generated. The smaller the step angle is (i.e. more number of points **n**), the larger accuracy this map will provide.

5.3 Vehicle Position Logging

Instead of statically scanning the environment at only one position, the vehicle should move to other positions to collect more information especially when the vehicle is placed at a large area beyond the range limit of the ultrasonic sensor. So tracking the vehicle's position after every movement and converting its new positions onto the coordinate system is very essential.

By reading the wheel odometer, the traveling distance of each wheel l_1 and l_2 can be obtained. Again, deducing from geometry calculation, the relationship between the displaced vehicle position $O_1(Ux, Uy)$ and the traveling distance l_1, l_2 can be obtained in terms of two situations when the vehicle is turning or yawing:

a) Turning (identical rotary direction of both wheels)

$$Ux = (r_1 + \frac{L}{2}) \cdot \sin \alpha = (\frac{l_1}{\alpha} + \frac{L}{2}) \cdot \sin \alpha$$

$$Uy = (r_1 + \frac{L}{2}) - (r_1 + \frac{L}{2}) \cdot \cos \alpha = (\frac{l_1}{\alpha} + \frac{L}{2}) \cdot (1 - \cos \alpha)$$

[where, $\alpha = \varphi = \frac{l_2 - l_1}{L}$]

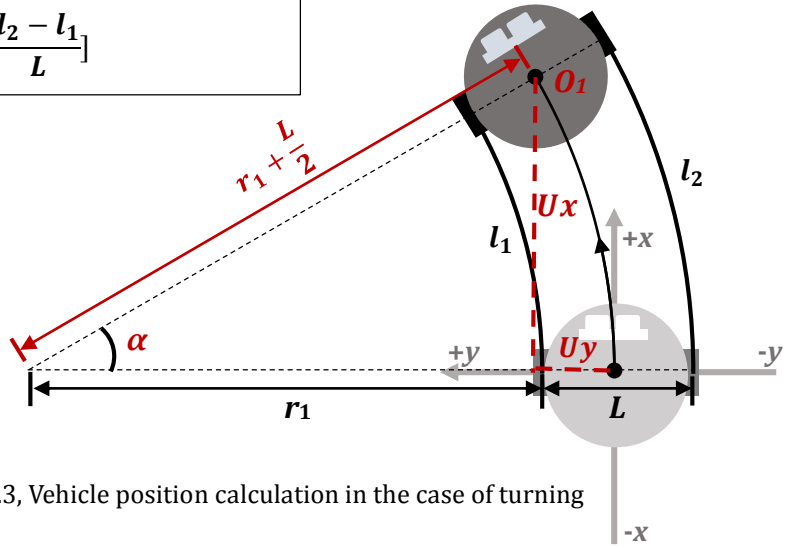


Figure 5.3, Vehicle position calculation in the case of turning

If taking into account the motors' rotary direction ($\pm l$) and the yaw angle direction ($\pm \varphi$), the above formula could be extended to two equations in terms of left turning O_1, O_2 and right turning O_3, O_4 of the vehicle motion.

Left Turning: $O_1(X_1, Y_1) = O_2(X_2, Y_2) = \begin{pmatrix} Ux \\ Uy \end{pmatrix} = \begin{pmatrix} (\frac{l_1}{\varphi} + \frac{L}{2}) \cdot \sin \varphi \\ (\frac{l_1}{\varphi} + \frac{L}{2}) \cdot (1 - \cos \varphi) \end{pmatrix} \quad [l_1 < l_2]$

Right Turning: $O_3(X_3, Y_3) = O_4(X_4, Y_4) = \begin{pmatrix} Ux' \\ Uy' \end{pmatrix} = \begin{pmatrix} -(\frac{l_2}{\varphi} + \frac{L}{2}) \cdot \sin \varphi \\ -(\frac{l_2}{\varphi} + \frac{L}{2}) \cdot (1 - \cos \varphi) \end{pmatrix} \quad [l_2 < l_1]$

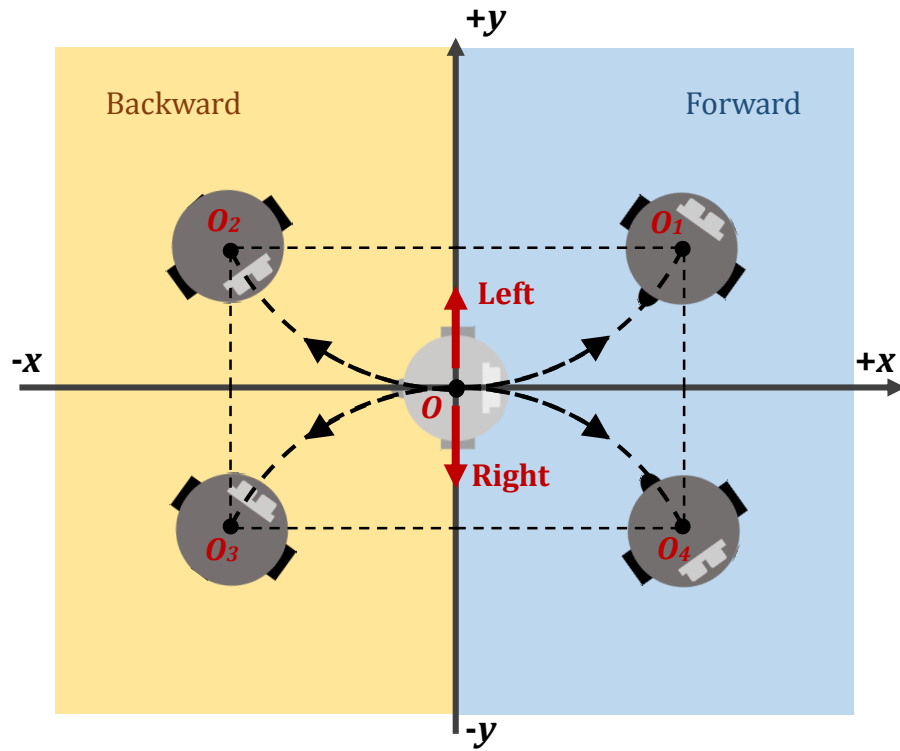


Figure 5.4, Demonstration of the vehicle position after displacement

b) Yawing (different rotary direction between two wheels)

$$\begin{aligned}
 Ux &= \left(r_2 - \frac{L}{2}\right) \cdot \sin\alpha = \left(\frac{l_2}{\alpha} - \frac{L}{2}\right) \cdot \sin\alpha \\
 Uy &= \left(r_2 - \frac{L}{2}\right) - \left(r_2 - \frac{L}{2}\right) \cdot \cos\alpha = \left(\frac{l_2}{\alpha} - \frac{L}{2}\right) \cdot (1 - \cos\alpha) \\
 [where, \alpha &= \varphi = \frac{l_2 + l_1}{L}]
 \end{aligned}$$

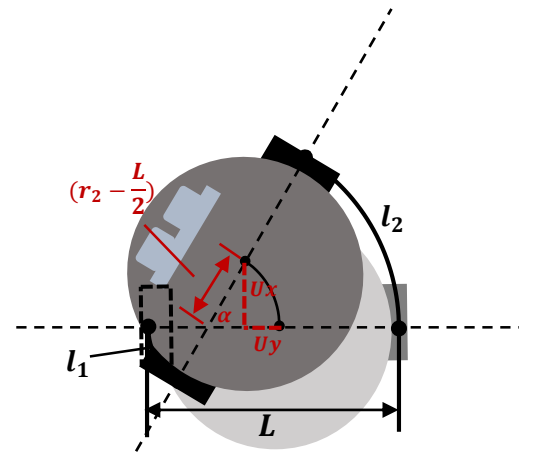


Figure 5.5, Vehicle position calculation in the case of yawing

[As shown from the graph as well as from the calculation,
 Ux and Uy in this case is between 1~2 cm which is eligible]

5.4 Converting Local Coordinates to Global Coordinates

In this project, the ultrasonic radar constantly starts rotating from the vehicle's forward orientation at every mapping position. So the $+x$ direction of a coordinate system is set to the vehicle's forward direction because the Trigonometric calculation is based on the $+x$ axes. However, after the vehicle moves forward, backward, or turning left and right, the original coordinate system has been translated or rotated. The new coordinate system generated at the latest vehicle position is termed the Local coordinate system. Therefore, investigating a method to convert this Local coordinate system to the Global coordinates is essential. The theory of coordinate transformations are demonstrated below:

Coordinate Translation

The translation equation given that every point on a coordinate system can be transformed to the original coordinate by moving an equivalent distance:

$$x' = x + Ux$$

$$y' = y + Uy$$

Transferred to matrix format given that:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} Ux \\ Uy \end{pmatrix}$$

and its reverse form:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} Ux \\ Uy \end{pmatrix}$$

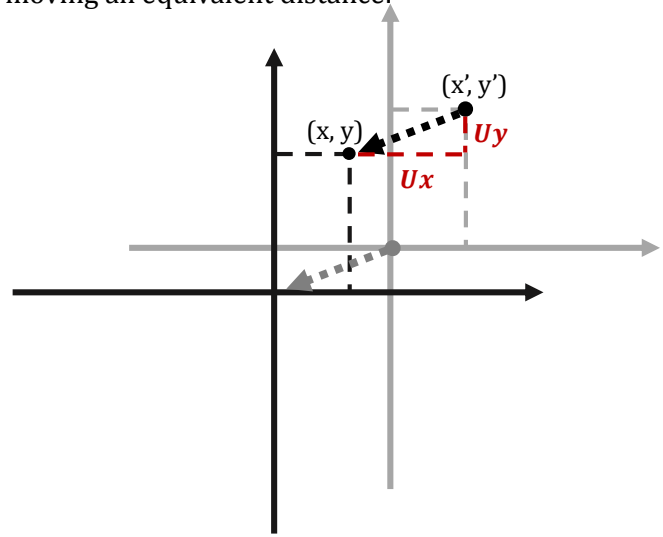


Figure 5.6, coordinate translation

Coordinate Rotation

The rotation of a coordinate system can be deduced by trigonometric derivation where θ is the angle of rotation:

$$x' = r \cdot \cos(\theta + \beta) = r \cdot \cos\theta \cos\beta - r \cdot \sin\theta \sin\beta$$

$$y' = r \cdot \sin(\theta + \beta) = r \cdot \sin\theta \cos\beta + r \cdot \cos\theta \sin\beta$$

$$[\text{where, } x = r \cdot \cos\beta, \quad y = r \cdot \sin\beta]$$

$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$

$$y' = x \cdot \sin\theta + y \cdot \cos\theta$$

Transferred to matrix format:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

and the reverse form:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix}$$

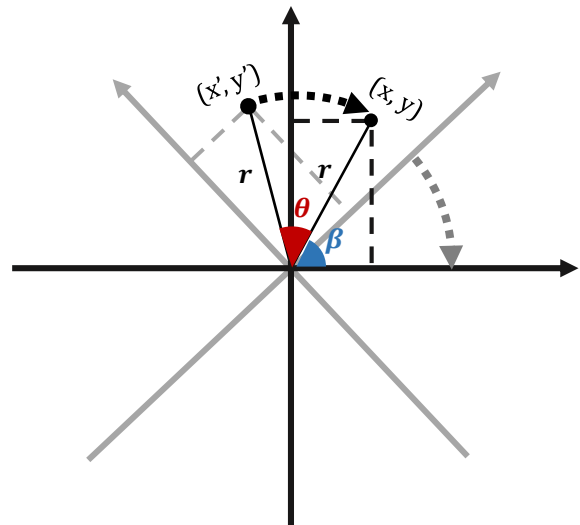


Figure 5.7, coordinate rotation

Since the Global coordinate system is constructed based on the initial position of the vehicle $O(0,0)$, the Local coordinates (which is constructed based on the latest position of the vehicle $O_N(X_N, Y_N)$) have to be transformed to the Global coordinates for integrating the latest obtained mapping points. An example is made to illustrate the derivation of coordinate system transformation according to this project.

Supposing the vehicle is initially placed at origin O before travelling to the first position O_1 then to the second position O_2 , and an obstacle is found at location Z during the this process.

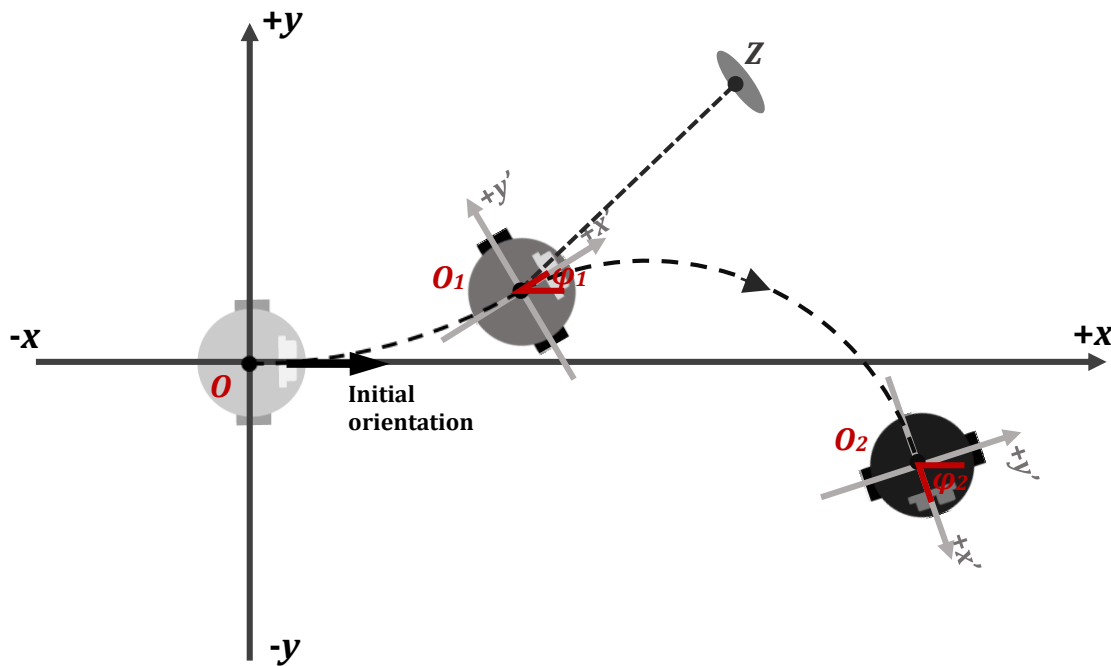


Figure 5.8, Example of a typical mapping case

As the vehicle has turned its direction, the yaw angle φ is no longer zero and the value at position O_1 and O_2 is φ_1 and φ_2 respectively. These values can be individually calculated by the formula $\varphi = \alpha = (l_2 - l_1)/L$ which is deduced in Chapter 4 (where l_1, l_2 can be real-time measured by the wheel odometers). Continuously calculating the latest Local yaw angle α_N and accumulating itself with the previous Global yaw angle φ_{N-1} , the current Global yaw angle value of the vehicle φ_N can be obtained. [Where, Local yaw angle represents the current yaw angle with respect to the previous vehicle position and the Global yaw angle represents the current yaw angle corresponding to the initial vehicle position]

Derivation of
real-time yaw angle:

$$\varphi_0 = \alpha_0 = 0$$

$$\varphi_1 = \alpha_1 + \varphi_0$$

$$\varphi_2 = \alpha_2 + \varphi_1$$

.....

$$\varphi_N = \alpha_N + \varphi_{N-1}$$

As the calculation of the displaced vehicle position (derived in section 5.3) is regardless of the coordinate rotation, the corresponding calculated coordinate (Ux_N, Uy_N) is using the same scale of the Global coordinate system by default. So a transformation have to be applied to convert (Ux_N, Uy_N) to the actual coordinate (X_N, Y_N) in terms of the rotated axes if the vehicle has a yaw angle. As shown in the graph below, the calculated point coordinate O_2' using the formula in section 5.3 is based on the axes before rotation, and the actual point coordinate O_2 (real vehicle position at the next step) is with respect to the axes after rotation. The value of the rotation angle is the vehicle's yaw angle vehicle φ_1 at position O_1 .

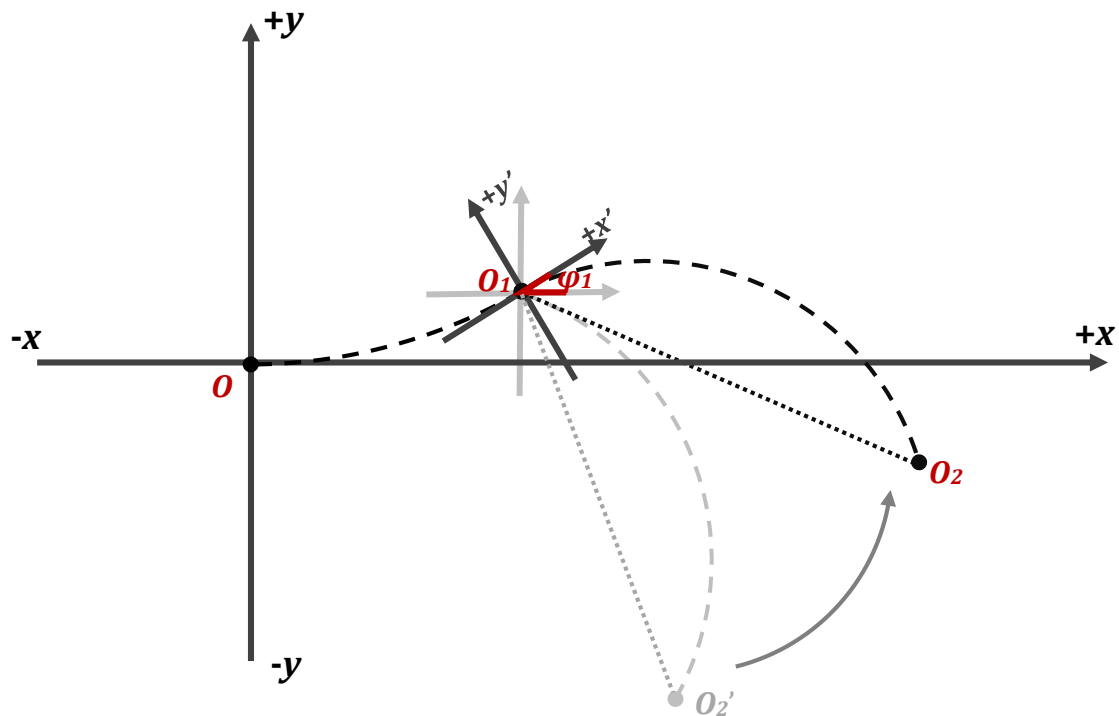


Figure 5.9, conversion of the vehicle position

The latest vehicle position $O_N (X_N, Y_N)$ in terms of the Global coordinate system is obtained by rotating the calculated coordinate (Ux_N, Uy_N) with the former yaw angle φ_{N-1} and then translating it to the Global coordinates by adding the former vehicle position $O_{N-1} (X_{N-1}, Y_{N-1})$.

Derivation of real-time vehicle position:

$$O: \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$O_1: \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = \begin{pmatrix} \cos\varphi_0 & \sin\varphi_0 \\ -\sin\varphi_0 & \cos\varphi_0 \end{pmatrix} \cdot \begin{pmatrix} Ux_1 \\ Uy_1 \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix}$$

$$O_2: \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} \cos\varphi_1 & \sin\varphi_1 \\ -\sin\varphi_1 & \cos\varphi_1 \end{pmatrix} \cdot \begin{pmatrix} Ux_2 \\ Uy_2 \end{pmatrix} + \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}$$

.....

$$O_N: \begin{pmatrix} X_N \\ Y_N \end{pmatrix} = \begin{pmatrix} \cos\varphi_{N-1} & \sin\varphi_{N-1} \\ -\sin\varphi_{N-1} & \cos\varphi_{N-1} \end{pmatrix} \cdot \begin{pmatrix} Ux_N \\ Uy_N \end{pmatrix} + \begin{pmatrix} X_{N-1} \\ Y_{N-1} \end{pmatrix}$$

The mapping of the surrounding obstacles in section 5.2 is based on the Local coordinate system at the latest vehicle position, so a conversion to the Global coordinates have to apply on the collected point of obstacle $Z_n(x'_n, y'_n)$. As shown in the graph below, an obstacle at location Z is detected when the vehicle is at position O_I and then it is plotted on the Local coordinates as (x, y) . After converting to the Global coordinates, the point coordinate of the obstacle Z is then (x', y')

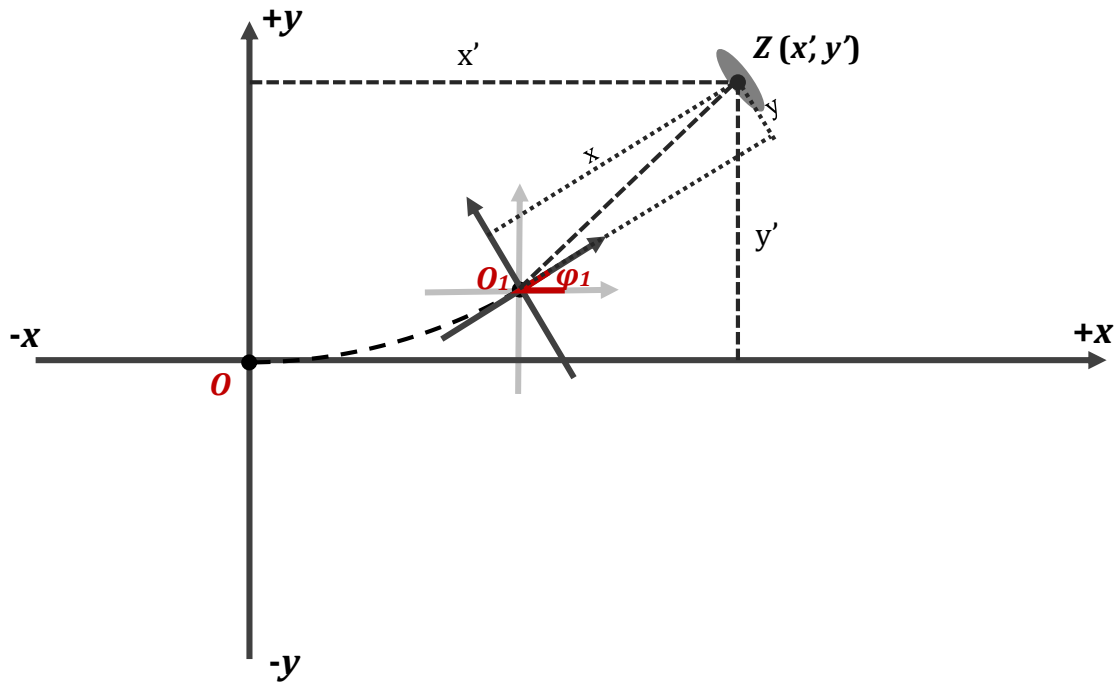


Figure 5.10, conversion of the obstacle location

The obstacle location (x'_n, y'_n) with respect to the Global coordinate system is obtained by rotating the point (x_n, y_n) on Local coordinates with the axes rotation angle φ_N (latest vehicle yaw angle) and then translating it to the Global coordinates by adding the coordinate of the latest vehicle position $O_N(X_N, Y_N)$.

$$\begin{pmatrix} x'_n \\ y'_n \end{pmatrix} = \begin{pmatrix} \cos\varphi_N & \sin\varphi_N \\ -\sin\varphi_N & \cos\varphi_N \end{pmatrix} \cdot \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} X_N \\ Y_N \end{pmatrix} \quad \text{at } O_N(X_N, Y_N)$$

$$[\text{where, } \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} R_n \cdot \cos(n\theta) \\ R_n \cdot \sin(n\theta) \end{pmatrix}]$$

The method of converting Local coordinate systems to the Global coordinates is shown as the flowchart below. The algorithm is integrated into the mapping procedure (i.e. the process of conversion is operated in real time) to minimize the execution time of the program. The vehicle yaw angle and the vehicle location is updated before the conversion begins so the newly collected points can be real time transformed to the Global coordinates using the above conversion formula with these updated data.

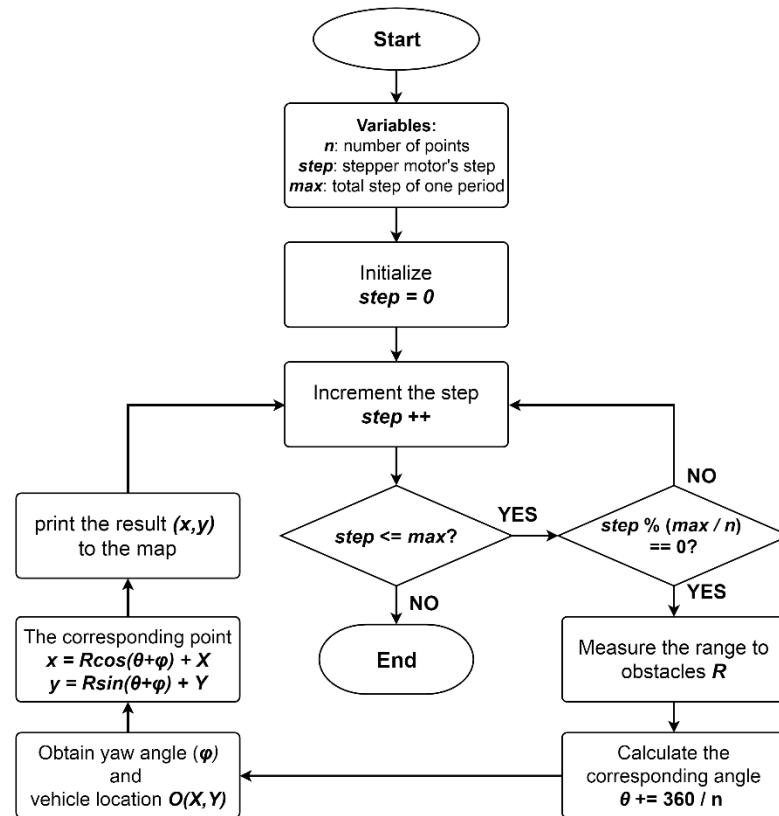


Figure 5.11, Flowchart of mapping and localizing

5.5 Route Planning

The previous sections are only preparations for achieving autonomous driving. This section which is mainly about the routing planning strategy is the key to implement autonomous driving. An autonomous vehicle have to decide an optimal route by itself in terms of the collected information from the environment. The quality of the route planning directly determines the intelligent level of the autonomous vehicle. There are numerous types of route planning strategies from Computer Science, however, a complex route planning strategy requires complicated algorithms and high hardware performance so it is difficult to implement with Arduino. Thus a relatively simple and straightforward logic of route planning is firstly investigated in this project to quicker reach the objectives.

Basically this route planning method is based on selecting vehicle directions and it relies on the data collected from the ultrasonic radar. As the ultrasonic sensor can provide the distance information of the surroundings, theoretically, the vehicle is able to detect which direction has the largest distance. A larger distance to the obstacle implies a larger space for the vehicle to travel and hence a higher possibility of an optimal route (except for some special scenarios like a maze). According to this idea, the vehicle is programmed to obtain the traveling direction by searching for the largest area (i.e. the largest distance to obstacles).

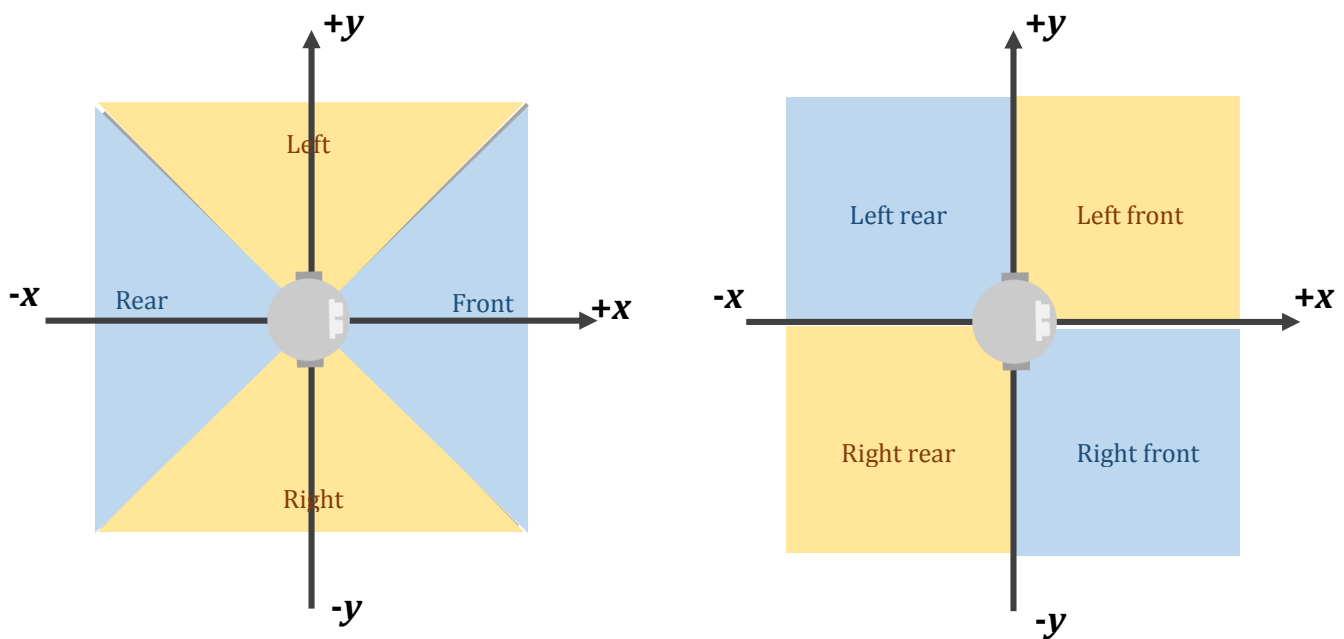


Figure 5.12, Demonstration of the classification of the surroundings

Generally, there are 8 alternative directions available to select at every vehicle position and this is consistent with the 8 motion directions of the vehicle. So the space around the vehicle is classified as these 8 directions: Left front, Left, Left rear, Rear, Right rear, Right, Right front, and Front. For the convenience of software implementation, these 8 directions are marked with number 1 – 8 respectively as shown in the figures below.

To find out which direction has the largest space for the vehicle to travel, the total distance of each direction is computed and compared with each other. The largest total distance can be obtained and the corresponding direction is then the target direction for the next movement of the vehicle. Continuously performing a search at each vehicle position and moving one step towards the resulting direction, an optimal route is able to be found in most of the cases.

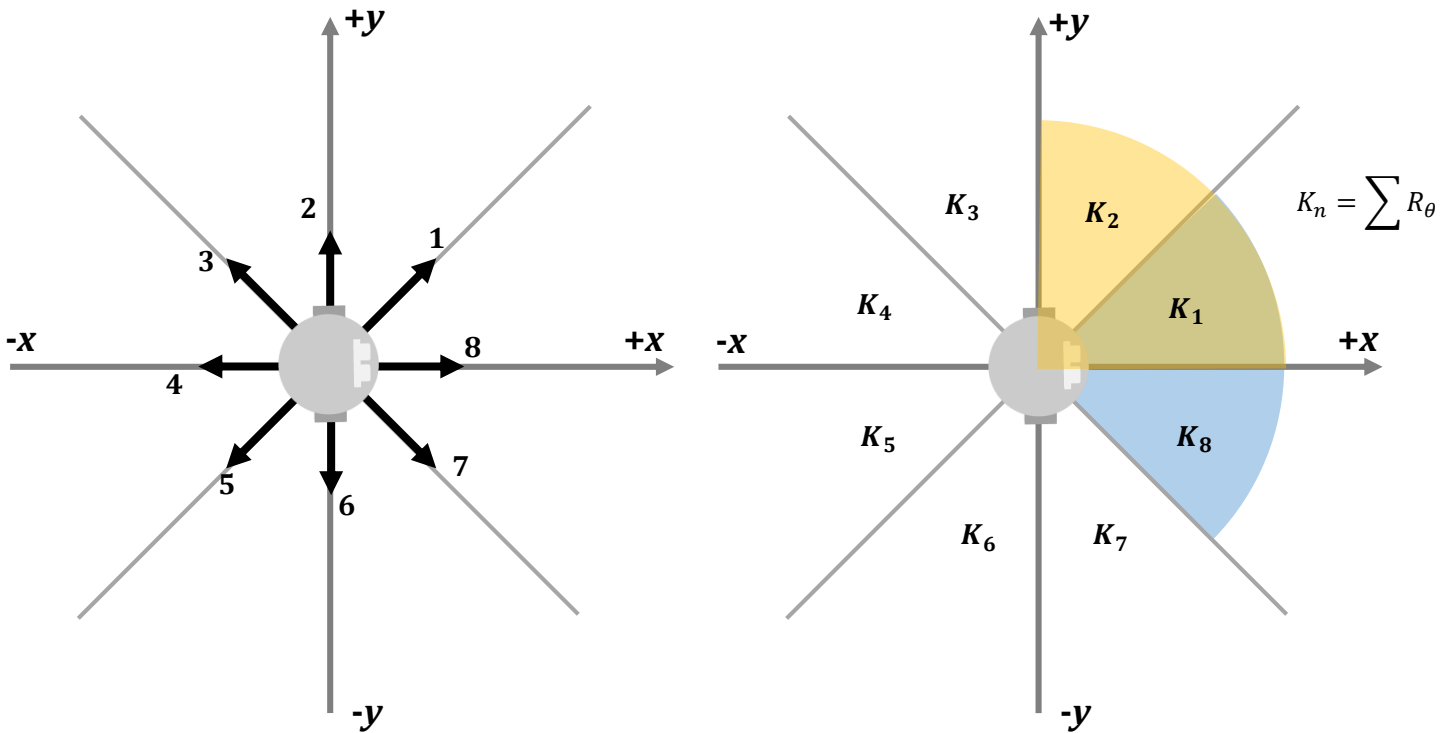


Figure 5.13, Demonstration of the route planning strategy

<p>1 = Left front</p> <p>2 = Left</p> <p>3 = Left rear</p> <p>4 = Rear</p> <p>5 = Right rear</p> <p>6 = Right</p> <p>7 = Right front</p> <p>8 = Front</p>	<p>Left front: $K_1 + K_2$</p> <p>Left: $K_2 + K_3$</p> <p>Left rear: $K_3 + K_4$</p> <p>Rear: $K_4 + K_5$</p> <p>Right rear: $K_5 + K_6$</p> <p>Right: $K_6 + K_7$</p> <p>Right front: $K_7 + K_8$</p> <p>Front: $K_8 + K_1$</p>
---	---

The detailed software implementation of this route planning strategy is illustrated by the flowchart below. In general, the filtration of the largest total distance (i.e. largest space) is consisted of a data logging stage and a data comparison stage. While the ultrasonic radar is scanning the surroundings, the Arduino simultaneously calculating the total distance at each direction and recording them into two arrays $a[Ctrl]$ (represents K_n) and $b[Ctrl]$ (represents K_{n-1}), where the index of the arrays $[Ctrl]$ represents the direction number. Then these two arrays are added $a[Ctrl] + b[Ctrl]$ (represents $K_n + K_{n-1}$) for the convenience of comparison. After finishing one cycle of scanning, the Arduino then compares each element in $a[Ctrl] + b[Ctrl]$ to search out the largest value and its corresponding direction is the resulting direction.

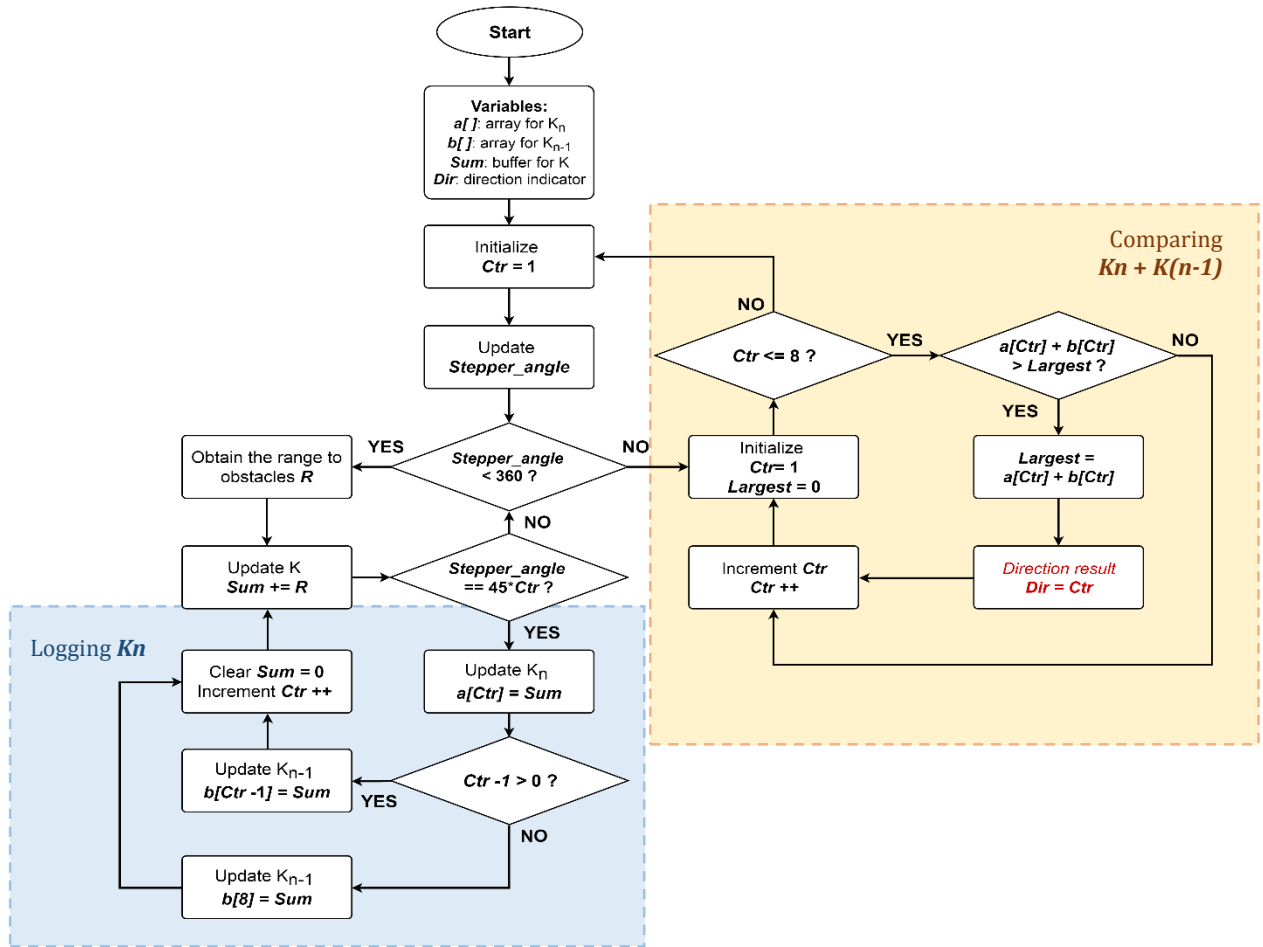


Figure 5.14, Flowchart of route planning

5.6 Overall Mapping Algorithm

After combining all the mapping and localizing algorithms together, an overall mapping algorithm is derived as shown in the flowchart below. The general process is moving one step (a duration of 1s) towards a direction, then update the latest yaw angle and vehicle position, before mapping at this new vehicle position and convert the Local coordinates to Global coordinates. Afterwards, the mapping results are exported to Raspberry Pi for data processing and real time display. Finally the vehicle direction is updated in terms of

the current mapping results and this new direction is used as a reference for the next vehicle movement (the 0 direction represents stop). Notably, the reading of the wheel odometers is put into an interrupt service routine to keep it executing at background without consuming the software resource. To achieve this, timer2 is set to overflow at every 256 μ s to regularly check the state of the IR sensors behind the steering wheels.

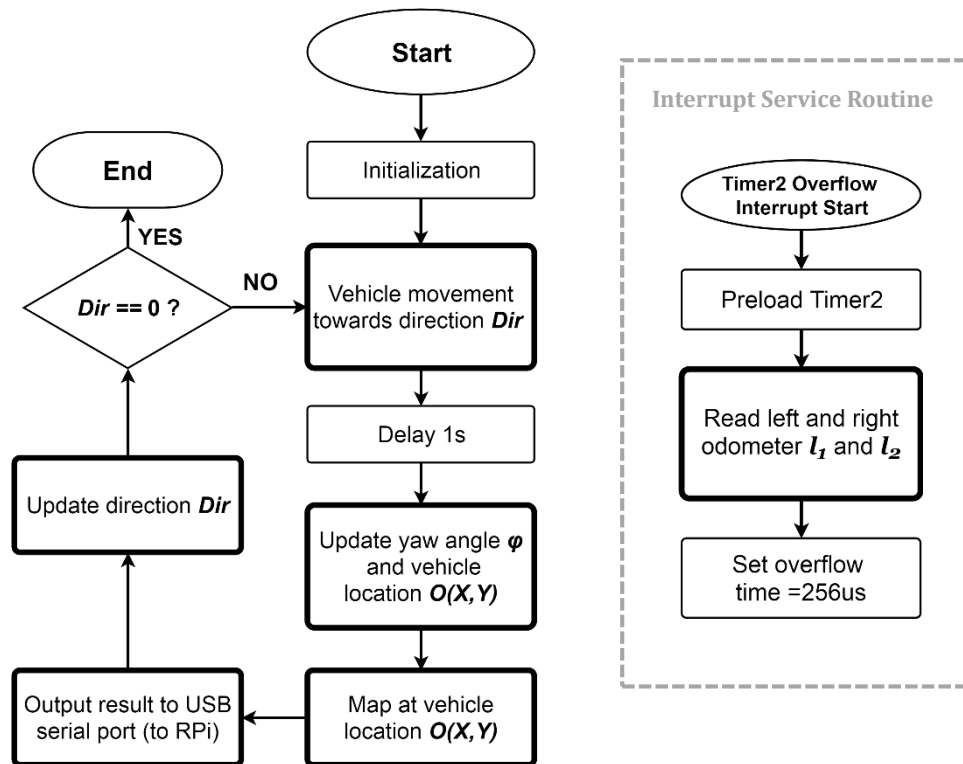


Figure 5.15, Flowchart of overall algorithm

Chapter 6: Results and Evaluation

6.1 Static Mapping

Primarily, the performance of this autonomous vehicle to map obstacles at a static position was examined to evaluate the behavior of the ultrasonic radar. A relatively small space was selected to eliminate the effects of the range and angle limit of the ultrasonic sensor.

Placing the vehicle in the middle of a 50x50cm square space with walls surrounded, the vehicle stands at the original location and maps around the environment by turning the ultrasonic sensor 360 degree.

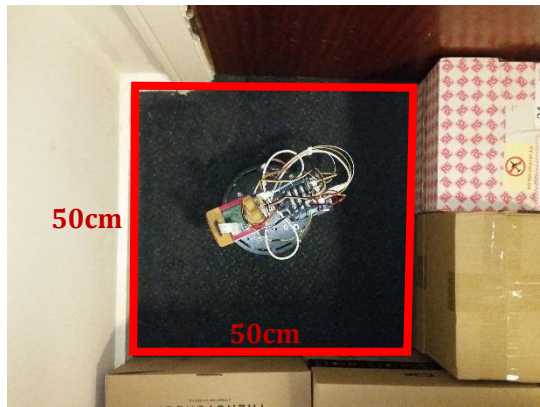


Figure 6.1, Test Environment: 50 cm Square

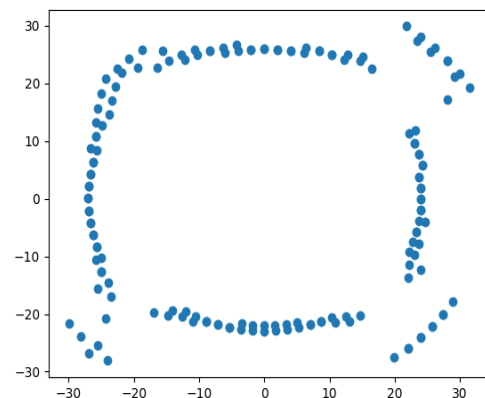
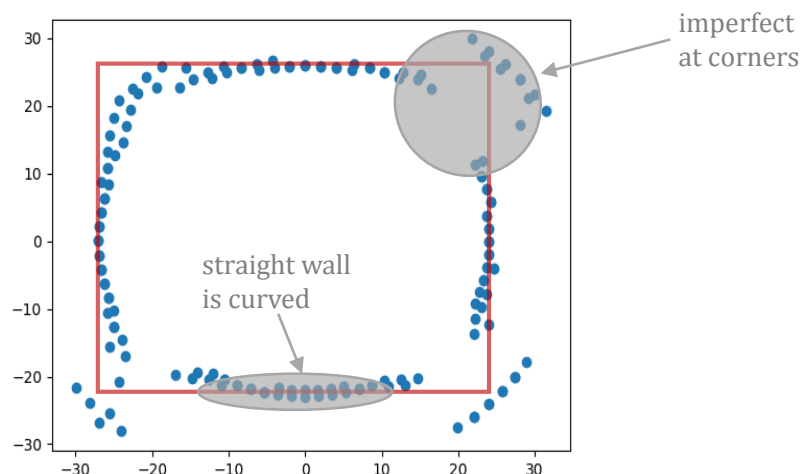


Figure 6.2, Static Mapping Results

After finishing one cycle of scanning, a general layout of the environment can be obtained but as mentioned in Chapter 3, the accuracy of the ultrasonic sensor when scanning against corners are very low. Also, the straight walls that plotted on the map is curved because of the low range precision of the ultrasonic sensor.



6.2 Dynamic Mapping

To test the performance of coordinate transformations, the vehicle should perform a dynamic mapping (with vehicle motions integrated) in a relatively large area. Since statically mapping a larger space could be very inaccurate on account of the distance and angle limit of the ultrasonic sensor. The vehicle is manually commanded to move to multi-positions to collect more data from the same environment intending to increase the mapping accuracy.

In this experiment, a 106x52cm rectangular space with walls surrounded is used for the test. Taking into account the size of the vehicle, 3 mapping locations are selected. The vehicle is programmed to travel to position P2 then position P3 from the initial position P1 and map at these 3 positions respectively then finally convert all the collected points on the Global coordinate system.

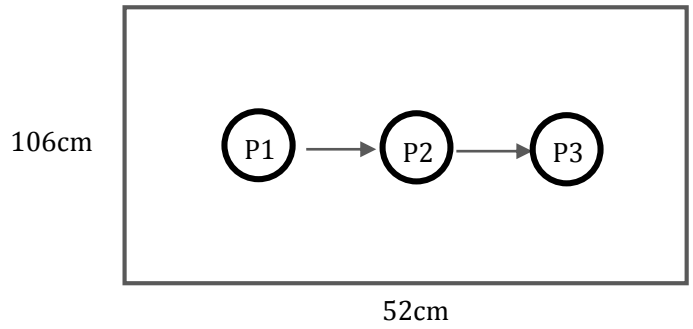


Figure 6.3, Test environment

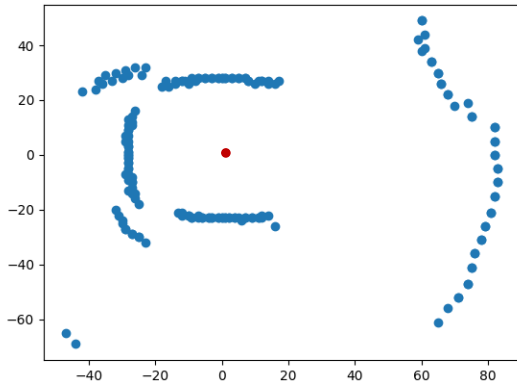


Figure 6.4, Position 1

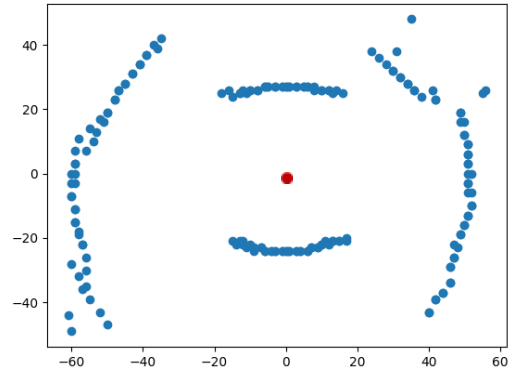


Figure 6.5 Position 2

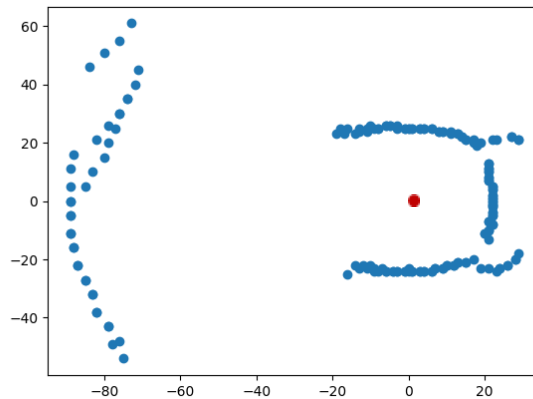


Figure 6.6 Position 3

The mapping results corresponding to position P1, P2 and P3 are shown on the above graphs respectively. It can be observed that the individual mapping results at each of the mapping positions is very inaccurate that all of the above maps fails to indicate the actual rectangular space. However, when looking deeper into each of the maps, the nearby obstacles was mapped accurately to some extent. So combining the three maps together using the coordinate transformations was attempted to generate a more accurate map. The map integrated with all the collected points is sketched below:

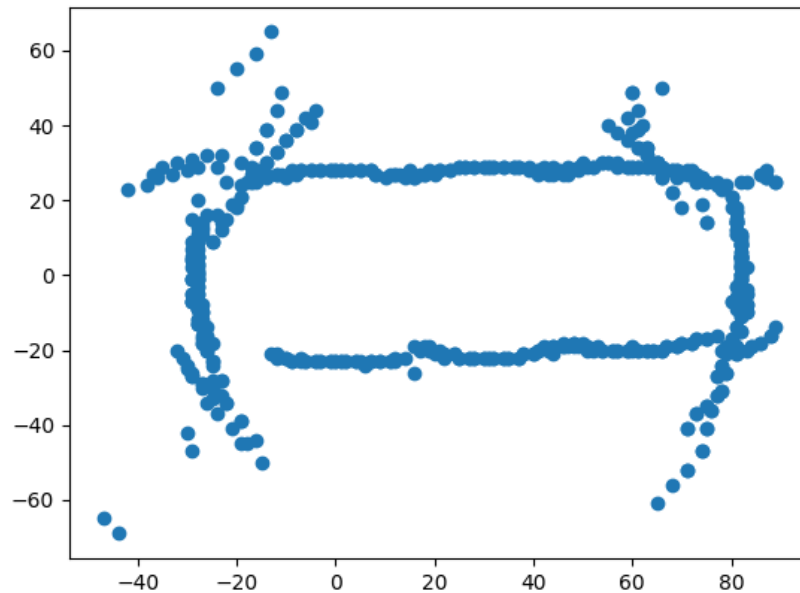
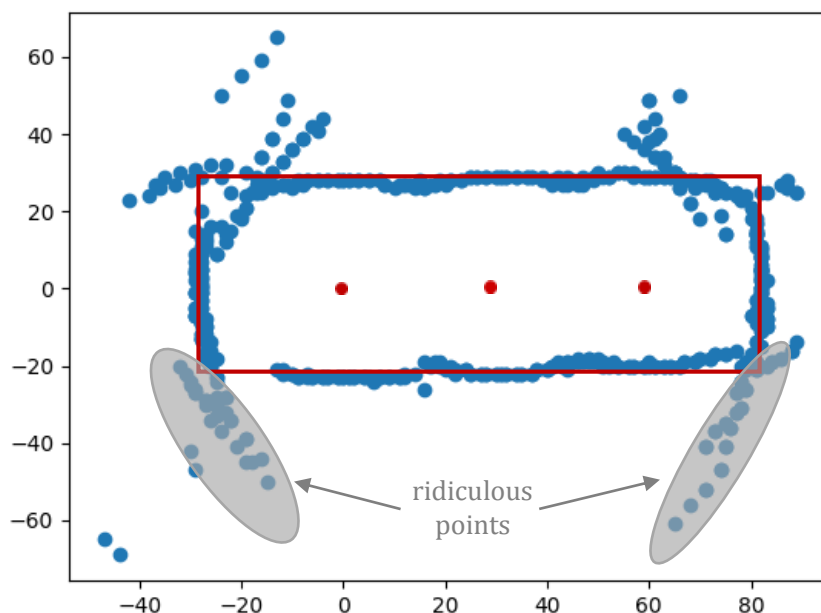


Figure 6.7 Mapping results after transformation

As shown from the above Global map, the accuracy has a significant increase compared to the individual maps. The rough shape of a rectangular space emerges and the dimensions shown from the map is identical to the real test space. Although mapping at corners and the straight walls still exist, the problems are not so obvious as in Section 6.1. However, it can also be observed that there are some irrelevant points been plotted on the map causing confusions. These noisy points could be further filtered to produce a perfect map.



6.3 Mapping with Route Planning

The performance of the vehicle mapping with route planning algorithm enabled was test in a relatively complex space. The vehicle was challenged to map at an “L” shape rectangular turn to test if the vehicle is able to correctly select the optimal route without hitting obstacles. This area was not enclosed to simulate the open area as the exit of this simple maze. The vehicle was placed at the starting point and all the rest of the positions are automatically selected by the route planner integrated on the vehicle.

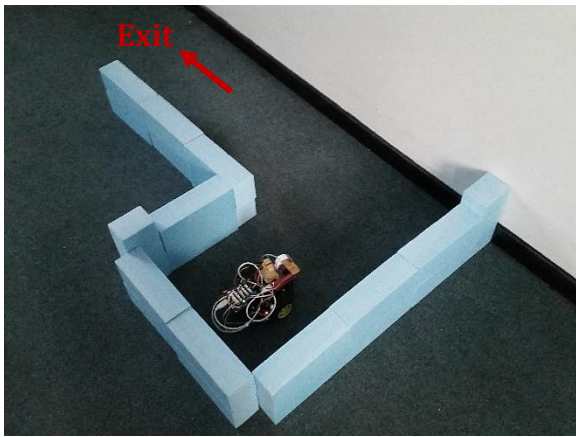


Figure 6.8 Test environment

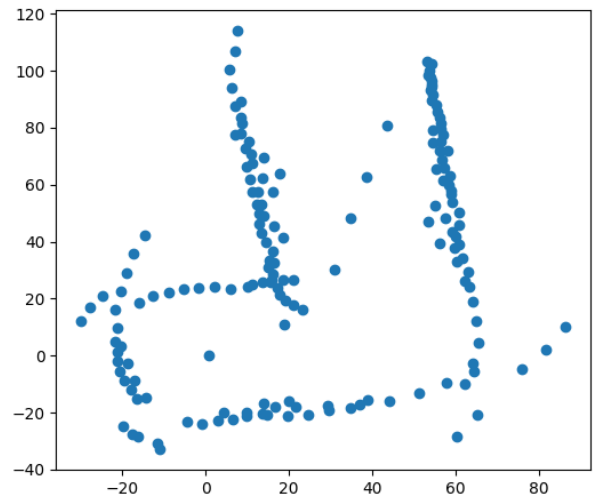
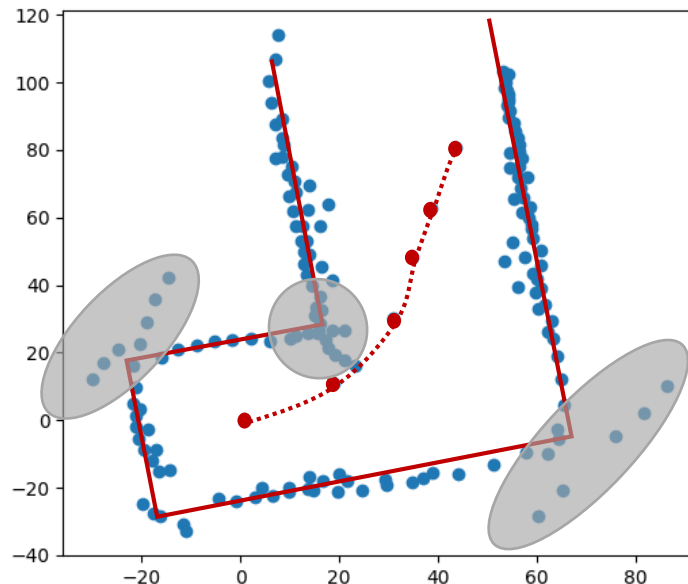


Figure 6.9 Mapping result

As shown from the generated map, the vehicle successfully avoided the obstacles and found the correct route. This route was generally an efficient route because it immediately turned before getting too close to the wall on the right, although the last position was not so reasonable due to the angle limit of the ultrasonic sensor. Evaluating the accuracy of the map, 85% of the points lies near the actual walls forming a clear layout of the test space and the problem of curved straight walls are not so obvious as before. But the imperfection at corners were still exist.



Chapter 7: Improvements and Future Work

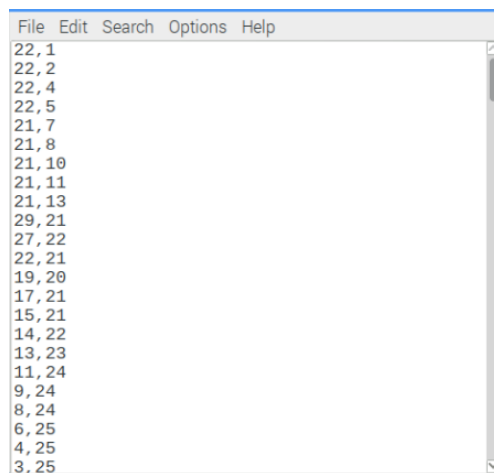
7.1 Real time map display

Displaying the generated map in real time could assist debugging and also contributes to a better user experience. The Raspberry Pi is a powerful tool to implement this. Since Arduino has a simple-to-use serial output, the mapping data could be real time forward to the USB port of the Raspberry Pi using serial communication. Then the Raspberry Pi can plot these data on the graph to form a map in real time.

The mapping data from Arduino are processed and output to the serial in the form of point coordinates corresponding to the location of the obstacles as well as the position of vehicle. The point coordinates are separated by a new line, and the value of X s and Y s are split by a comma in each line. The format of the data are simply produced by the following Arduino code:

```
Serial.print(X);  
Serial.print(",");  
Serial.println(Y);
```

Then using the Raspberry Pi serial reading function shown in the Python code, the data can be imported and save to a ".txt" file:



```
File Edit Search Options Help  
22,1  
22,2  
22,4  
22,5  
21,7  
21,8  
21,10  
21,11  
21,13  
29,21  
27,22  
22,21  
19,20  
17,21  
15,21  
14,22  
13,23  
11,24  
9,24  
8,24  
6,25  
4,25  
3,25
```

```
import serial  
  
file = open("test.txt", "w")  
ser = serial.Serial('/dev/ttyACM0', 9600)  
  
while True:  
    read_serial=ser.readline()  
    if read_serial == "*****\r\n":  
        break  
    file.write(read_serial)  
    print read_serial  
    file.close()  
    file = open("test.txt", "a")  
  
file.close()
```

Reading from this file, the data are loaded to plot on a graph using the *matplotlib* library in Python. The program stores the content of each line into an array and detects the symbol “,” at each line to extract the values of X and Y . These values are then loaded to two arrays correspondingly and plotted on a rectangular coordinate system. An animation method is used to refresh the collected points at a time interval of 500ms. This interval is relatively long compared to the vehicle’s scanning speed but it is sufficient in this case, otherwise the workload of the CPU and the power consumption will be too large if the update is too frequent.


```

import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

def animate(i):
    graph_data = open('test.txt','r').read()
    lines = graph_data.split('\n')
    xs = []
    ys = []
    for line in lines:
        if len(line) > 1:
            x, y = line.split(',')
            xs.append(float(x))
            ys.append(float(y))

    ax.clear()
    ax.scatter(xs,ys)

ani = animation.FuncAnimation(fig, animate, interval=500)
plt.show()

```

7.2 Video Streaming

Watching first-person view of the vehicle is not only entertaining but also assist developing an image based mapping system for future work. Making use of the WIFI module on the Raspberry Pi, the image captured by the Raspberry Pi camera could be real time streaming to the user devices (e.g. laptop or smartphone) via WIFI connection. The Raspberry Pi is able to communicate with, say, a laptop by connecting to the same router (i.e. in the same WLAN network).

In this project, Raspberry Pi is connected to a hotspot created by a laptop so the laptop itself is a router gateway that has the record of the IP address of Raspberry Pi. The Raspberry Pi camera continuously capture the image and simultaneously upload the image to the network via a built-in socket server with a high frame rate to form a video. The laptop then visit the IP address of Raspberry Pi and obtain the video stream from the socket server. The relevant code is borrow from the official document of Raspberry Pi camera. [14]

7.3 Future Work

7.3.1 Hardware Enhancement

Steering Motors and Wheel Odometers

The conventional DC steering motors used in this project significantly affect the

performance of the vehicle. These motors fail to produce precise rotary speed and angles which increases the difficulty for a precise movement control and measurement for mapping. Additionally, the resistance difference between the two DC motors challenge the vehicle to move in a straight line easily. While self-designing a servo system is time consuming and it is not the main objective of this project. So replacing the DC motors to an off-the-shelf servo motor with controller board supported is a better solution of this problem.

If the new servo system has a positioning feedback, the wheel odometer constructed in this project could be eliminated. If not, an improvement of the odometer could be applied by adding more scales on the wheel to increase the accuracy of the measurement.

Ultrasonic Sensor

Ultrasonic sensor is the most important sensor in this project that directly determines the mapping quality of the vehicle. As investigated in Chapter 3, the angle limit and the corner reflection of the ultrasonic sensor constantly exist so that there are large errors occurred and the performance of mapping is restricted. There is solution proposed to use multiple sensors or a sensor array, but this not only increase the difficulty in control and calculation but also increase the cost without resolving the underlying problem. A better solution is to replace the sensor with a laser ranger or a Lidar that is able to output very precise ranging information in a fast speed irrelative to the angle limit and corner reflection. However, this also considerably increase the price and have a potential issue on light sensitive (black or white) materials.

7.3.2 Software Improvement

Real time dynamic mapping

The current mapping methodology is relatively time consuming because the vehicle should wait for the ultrasonic radar to complete an entire cycle of scanning before traveling to the next position. This is due to the update of vehicle's latest yaw angle and position should wait after the vehicle stops to ensure the scanning process is stable (the travelling time of sound is too long to neglect when vehicle is moving).

A faster approach is to update the vehicle's yaw angle and position frequently at the background using interrupts. Then the vehicle can collect whatever points of obstacles and using the frequently updated odometer to convert these points on the Global map in real time. This method would require a faster processor speed for calculating the trigonometric equations, as it has been observed that these calculation time are too long to operate in the Arduino's interrupt service routine and cause the program stuck. In addition, it also requires the radar refresh in a much faster speed for a faster response, so Lidar is obviously the best option.

Occupancy Grid Mapping

As shown from Chapter 6, the mapping results more or less contain ridiculous noisy points due to the defection of ultrasonic sensor. As in most of the applications, a sensor

(even using Lidar) is impossible to produce an absolutely exact measurement. So a more advanced approach to minimize the effect of sensor error to improve the map is to generate an occupancy grid map using probabilistic models.

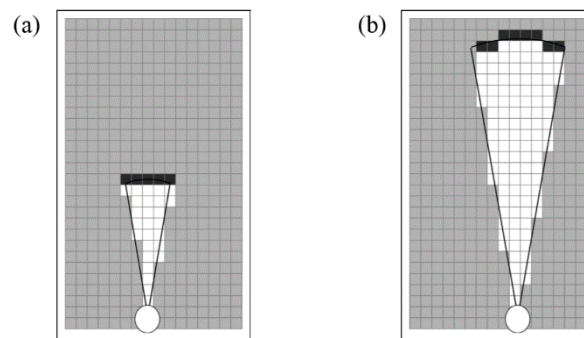


Figure 7.1 Example of occupancy grid maps [15]

Basically, the occupancy grid map is focus on whether an area is occupied by obstacles. The maps are generated to grid with identical cells and the probability of each cell been occupied is calculated. “1” represents the cell is occupied and “0” represents unoccupied. Then the occupancy of a cell is noted on the map with colors indicated. The darker the cell the higher probability this cell is occupied so generally white represents the unoccupied area and black represents the occupied area while grey represents unknown spaces. The corresponding probability is calculated based on the sensor collected data. The example conversion of the sensor data to the occupancy grid map is given below:

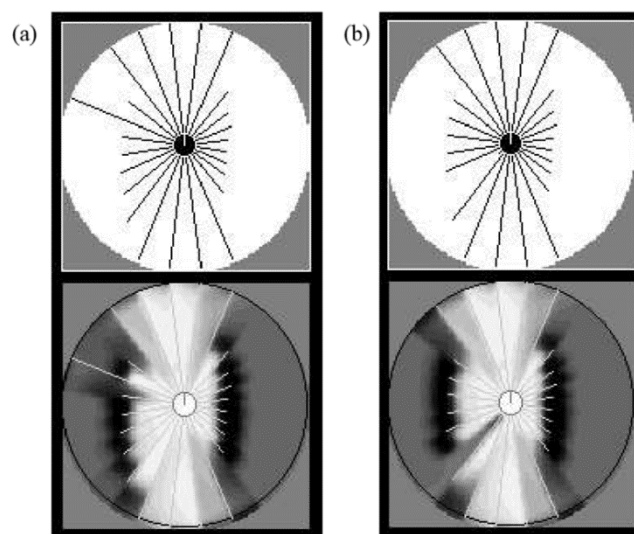


Figure 7.2 Performance of occupancy grid maps [15]

Since this approach takes the error into account with the probabilistic model, the map has a higher ability to deal with the sensor’s noise to produce a more reasonable map. The above two graphs illustrate the performance of occupancy grid map when the presence of noise. The map fuzz up the corresponding cell to indicate an uncertainty when the data experience sharp changes, so it does not affect the overview of the map.

Chapter 8: Project Review and Conclusion

8.1 Achieved Objectives and Deliverables

Basic Movements	Sensor Integration	Mapping and Localizing	Advanced
<ul style="list-style-type: none">▪ Omnidirectional movement	<ul style="list-style-type: none">▪ Wheel Odometer▪ Ultrasonic Radar▪ Gyroscope and Accelerometer	<ul style="list-style-type: none">▪ Map Generating▪ Vehicle Localizing▪ Static Mapping▪ Dynamic Mapping▪ Route Planning	<ul style="list-style-type: none">▪ PID controlling vehicle yaw angle▪ Real time mapping results display▪ Video Streaming

Table 8.1 Achieved Objectives and Deliverables

8.2 Attempted Ideas

8.2.1 Initial mapping method

The initial idea of mapping was to immobilize the ultrasonic sensor at the front of the vehicle without a stepper motor rotating it around. Simply yawing the entire vehicle platform 360 degree at the origin with a fixed yawing angle, the robot is able to map the environment at a static point. Then the vehicle yaws an accurate angle to point at the desired direction and moves straight forward to the desired location for dynamic mapping. The logic of this mapping method is extremely simple and there are less hardware requirements for this method, so this method was selected as the initial implementation of this project. However, a significant drawback of this method is that it depends on the vehicle to yaw very accurately which simultaneously requires an exact yaw angle measurement as well as a precise steering motor speed control, otherwise the inaccuracy of the mapping results will be accumulated to a very large value after a long period. But the gyroscope and accelerometer applied on this vehicle had constant issues of drift and noise (examined in Chapter 3) which increase the difficulty of producing accurate angular measurements. In addition, precisely controlling the imperfect manufactured DC motors used in this project took up much more time, efforts and resources. After attempted this idea, it had been observed that the mapping results were unstable depending on the behavior of the gyroscope and steering motors.

8.2.2 Accurate yaw angle turning

To achieve the initial idea of mapping, a PID control based method of yawing the entire vehicle an accurate angle was attempted. However, due to the significant and uncertain

delay of reading the gyroscope, the PID control sample time was difficult to set and the minimum controlled angle was larger than 15 degree which reduced the number of mapping points and hence reduced the accuracy of the map. In addition, the time for the vehicle to yaw a 360 degree was very long especially a small yawing angle was applied, therefore, this method consumed a relatively long time of mapping.

8.2.3 Moving straight

To simplify the calculation of vehicle positioning, the initial idea was to fix the vehicle's trajectory to straight lines. However, to control the vehicle moving straight is not a simple task since the DC steering motors have a resistance difference that increases the difficulty of producing identical rotary speed. The first solution to this problem was to calibrate the resistance of the motors, but the practical experiment showed that there was still speed difference during runtime even after balancing the motors' resistances due to the mechanical factors (i.e. different frictions applied on each of the motor). Another solution was to real time adjust one of the motor's speed to keep consistent with another motor's. This was achieved by linearly controlling the left motor's speed with the feedback measurement of angular velocity (i.e. increase left motor's speed if angular velocity is positive or decrease its speed if negative). This solution requires the gyroscope and Arduino to continually measure and control the vehicle's yawing angle so that the software resources was much consumed.

8.3 Time and Risk Management

The project initial time plan together with the actual time plan are included in the Appendix 1 and Appendix 2.

In summary, the time management of this project is successful as the actual progress was ahead of the initial schedule most of the time. Comparing the actual time plan with the initial one, all of the assigned objectives were achieved before the initial proposed deadline. As shown from the actual time plan, each of the tasks consumed less time than the estimation so the final task was finished 6 weeks prior to the proposed date. Therefore, these 6 weeks were taken advantage to make improvements on this project with more advanced features.

However, the project was not entirely smooth in the whole development period and there were several risks appeared. Although the first half of the project was smooth and the initial idea mentioned above was carry on without any issue, two risks was experienced in the second half of the project period. The first dominant risk was that the gyroscope

which is the core component in the original design was out of order. The gyroscope could cause the Arduino crash at runtime so that the corresponding vehicle's yaw angle measurement and control could not be performed. The second risk was caused by the first risk that the vehicle could not perform accurate trajectory (i.e. moving straight as well as precise angle turning) without the gyroscope. Hence the initial mapping method highly rely on the accurate yaw angle measurement and control could not be implemented as demanded, on account of these two risks. Because ordering a new module could takes up weeks of time, it was decided to explore a new mapping method independent on the gyroscope and alter most of the original design. Since these two risks had been taken into consideration in the project proposal, the mental preparation was awarded earlier before these incident occurs. With an idea that the actual motion pose of the vehicle could be calculated by geometry equations, an entire new method based on obtaining the exact vehicle pose was implemented with the wheel odometers. It had been found out that the vehicle's yaw angle and displacement could totally be obtained using this method so the corresponding algorithm was then derived. Within a week's time, the issue was solved and the vehicle was able to map without gyroscope. Surprisingly, the mapping results are better than the former ones and the new mapping method is quicker and more reliable, because this method is based on the exact measurements rather than relying on the precision of control. Eventually, the new design was selected as the final design of this project.

8.4 Project Conclusions

In summary, an autonomous vehicle with 2D mapping in the indoor environment was designed (in hardware and software), implemented (with basic, improved and advanced features) and evaluated (in terms of multiple scenarios) in this project. All the assigned tasks and objectives were achieved ahead of the proposed schedule and a number of additional improvements had also been made after the main tasks were accomplished. A few number of significant risks were experienced during the project period but they had been handled successfully.

This project have taken a meaningful role for the student through challenging and improving the student's ability in all aspects including creativity, planning, self-researching, execution, evaluation, conclusion, and time and risk management. Numerous important theoretical knowledge and practical skills have been acquired throughout the project. Hence the student is confident to challenge larger and more advanced projects in the future.

References

- [1] D. HENDRICKS, "5 Reasons You Should Embrace Self-Driving Cars," Startup Grind, 2016. [Online]. Available: <https://www.startupgrind.com/blog/5-reasons-you-should-embrace-self-driving-cars/>. [Accessed 11.04.2018].
- [2] M. MOON, "Waymo bids its self-driving bubble cars farewell" Engadget UK, 2017 [Online]. Available: <https://www.engadget.com/2017/06/13/waymo-retires-self-driving-bubble-cars/>. [Accessed 18.05.2018]
- [3] HEXAGON, "High-Precision GPS for Autonomous Vehicles" HEXAGON, 2017 [Online]. Available: <https://www.novatel.com/industries/autonomous-vehicles/#technology> [Accessed 18.05.2018]
- [4] Standard S A E. J3016, [J] "Sae international taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," levels of driving automation, 2014.
- [5] CORDLESS VACUUM GUIDE, "Bang-For-The-Buck Robots That Will Clean Your Home For You" CORDLESS VACUUM GUIDE, 2016 [Online]. Available: <https://www.bestcordlessvacuumguide.com/best-budget-robotic-vacuum/> [Accessed 18.05.2018]
- [6] I. Chandrakant, "Xiaomi unveils a smart vacuum cleaner bot in China" Mysmartprice, 2016 [Online]. Available: <https://www.mysmartprice.com/gear/2016/08/31/xiaomi-unveils-smart-vacuum-cleaner/> [Accessed 18.05.2018]
- [7] Kuri "Explore Kuri" Kuri, 2018 [Online]. Available: <https://www.heykuri.com/explore-your-home-robot> [Accessed 18.05.2018]
- [8] NASA, "Opportunity: Traverse Map Archive" NASA Jet Propulsion Laboratory, 2018 [Online]. Available: <https://mars.nasa.gov/mer/mission/tm-opportunity-all.html>
- [9] NASA, "Autonomous navigation" NASA Jet Propulsion Laboratory, 2013 [Online Document]. Available: <https://mars.nasa.gov/mer/home/posters/OpportunityPosterBack.pdf>
- [10] Long Fan Hui Zhou Robotics. "2WD Mobile Platform Instruction Manual" Long Fan Hui Zhou Robotics, 2013 [Online Document]. Available: <http://www.seeedstudio.com/document/3PA%20InstructionManual%20V1.1.pdf>
- [11] Arduino Official Website [Online]. Available: <https://www.arduino.cc/>
- [12] Raspberry Pi 3B+ Layout [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi
- [13] Seeedstudio Official Website [Online]. Available: <https://www.seeedstudio.com/>
- [14] Raspberry Pi official document <http://picamera.readthedocs.io/en/latest/recipes2.html>
- [15] Thrun S, Burgard W, Fox D. Probabilistic robotics[M]. MIT press, 2005.

Appendix

Appendix 1: original time plan

Tasks	Month /	Oct			Nov			Dec			Jan	Feb			Mar			Apr			May			Jun								
		3	4	5	6	7	8	9	10	11	12	13 ~ 18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	
Objective 1: Literature Review Prospects of autonomous systems Theory of autonomous systems Applications of autonomous systems																																
Objective 2: Vehicle Platform Construction																																
Assemble the provided kits																																
Enable the motors with H-bridge																																
Control the motors using Arduino																																
Perform basic movements of the vehicle																																
[Advanced] Apply PID control on the vehicle																																
Objective 3: Adding Sensors																																
Research the Arduino sensors																																
Determine and apply the sensors																																
Create a user-interface																																
Develop algorithms to prevent hazards																																
Objective 4: Mapping Algorithm																																
Create a coordinate system																																
Develop algorithms to find and follow walls																																
Record the traveling track of the vehicle																																
[Advanced] Explore other mapping methods																																
Objective 5: Image Capturing																																
Research Raspberry PI camera																																
Develop codes to enable the camera																																
[Advanced] Apply image recognition																																
Assessment Components																																
Project proposal																																
First interview with moderator																																
Second interview with moderator																																
Draft project thesis																																
Final project thesis																																
Third interview with moderator																																
Oral presentation																																
Individual task																																
General Objective																																
D1~D7: deliverables																																
Advanced tasks																																
M1~M8: milestones																																
Vacation and exam period																																
R1~R3: risks																																

Appendix 2: actual time plan

Tasks		Month / Week											Oct		Nov		Dec		Jan	Feb		Mar		Apr		May		Jun			
		3	4	5	6	7	8	9	10	11	12	13 ~ 18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Objective 1: Literature Review																															
Prospects of autonomous systems																															
Theory of autonomous systems																															
Applications of autonomous systems																															
Objective 2: Vehicle Platform Construction																															
Assemble the provided kits																															
Enable the motors with H-bridge																															
Control the motors using Arduino																															
Perform basic movements of the vehicle																															
[Advanced] Apply PID control on the vehicle																															
Objective 3: Adding Sensors																															
Research the Arduino sensors																															
Determine and apply the sensors																															
Create a user-interface																															
Develop algorithms to prevent hazards																															
Objective 4: Mapping Algorithm																															
Create a coordinate system																															
Develop algorithms to find and follow walls																															
Record the traveling track of the vehicle																															
[Advanced] Explore other mapping methods																															
Objective 5: Image Capturing																															
Research Raspberry Pi camera																															
Develop codes to enable the camera																															
[Advanced] Apply image recognition																															
Improvement Works																															
Adding stepper motor (ultrasonic radar)																															
Real time map display with RPi																															
Adding wheel odometer																															
Video streaming with RPi camera																															
Route planning																															
Assessment Components																															
Project proposal																															
First interview with moderator																															
Second interview with moderator																															
Draft project thesis																															
Final project thesis																															
Third interview with moderator																															
Oral presentation																															
Individual task																															
General Objective																															
Advanced tasks																															
Vacation and exam period																															