Remote Control Car with Depth Mapping and Exploration

BY

IAN KOONG FUN TSOON

A REPORT

SUMITTED TO

UNIVERSITI TUNKU ABDUL RAHMAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

BACHELOR OF INFORMATION TECHNOLOGY (HONS) COMPUTER ENGINEERING

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

(PERAK CAMPUS)

Jan 2016

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Remote Control car With Depth Mapping and Exploration**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENT

Throughout the whole project, I would like to first express my gratitude to my FYP supervisor, Mr. Wong Chee Siang who gives me a lot patient, support and guidance. As this project I am doing require skills in embedded system, I would like to take this opportunity to thanks Mr. Lee Wai Kong also as I have attended a few of his classes and the knowledge I am able to get and play it in the project. As the project proceeds, many inspiration and ideas I was able to gather from some friends of mine which is very useful to improve the project I am doing.

Last but not least, I would also like to thank FICT for giving me this opportunity to do this project as my FYP as I am really interested in topic like this. Throughout this project, other than embedded system, I also have the opportunity to learn mobile application programming which is an extra asset to me.

ABSTRACT

Autonomous car comes in many different forms and variety of instrument installed to help the car move depending on the motive of the autonomous car is created. The most challenging part of creating an autonomous car is to keep the cost as low as possible as in certain condition, it doesn't need that much function.

Autonomous car is also used in daily life product such as vacuum which allows it to clean the house without having user to walk around. The latest model autonomous vacuum is a success as it manages to clean the house entirely and it has the ability of knowing traveled path and untraveled path while cleaning

Some researchers also use autonomous car to use for exploring places human are not able to reach. For an instance, earthquake causes building to collapse and certain entrance which are sealed. For rescuers to understand how inside looks like while waiting for the entrance to be open, they can use an autonomous car to scan and understand the internal situation before making decision on how the rescue operation should be.

In this project, we propose on developing a smartphone controlled autonomous car prototype to assist in exploring in dangerous situation with ultrasonic mapping. The car is a 4 wheel car with Bluetooth connection for the smartphone controller, arm with infrared sensors for autonomous purpose and 2 ultrasonic sensors for depth scanning. The reason Bluetooth controller is requires in an autonomous car is to help assist the car movement is certain conditions where the autonomous function is not able to handle it.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY		I
ACKNOWLEDGEMENTS		II
ABSTRACT		III
TABLE OF CONTENTS		IV
LIST OF FIGURE		VI-X
LIST OF TABLE		XI
LIST OF ABBREVIATIONS		XII
CHAPTER 1	INTRODUCTION	1
1.1	PROBLEM STATEMENT	1
1.2	PROJECT OBJECTIVE	2
1.3	BACKGROUND	3
CHAPTER 2	LITERATURE REVIEW	4-7
CHAPTER 3	HARDWARE DEVELOPEMENT	8
3.1	MICROCONTROLLER AND DEVELOPMENT BOARD	8-10
3.2	L298N MOTOR DRIVER	11-16
3.3	HC-SR04 ULTRASONIC	17-19
3.4	HC-06 BLUETOOTH MODULE	20
3.5	INFRARED SENSOR	21-22
3.6	SMARTPHONE (SAMSUNG GALAXY S3)	23
CHAPTER 4	SOFTWARE DEVELOPEMENT	24
4.1	ARDUINO	24-25

4.2	ECLIPSE JUNO	26-29
CHAPTER 5	SYSTEM INTEGRATION	30
5.1	INFRARED SENSOR + AUTONOMOUS MODE	31-35
5.2	DIRECTION PAD	36-37
5.3	ULTRASONIC + SERVO + MAPPING	38-42
CHAPTER 6	FINAL PRODUCT	43
6.1	REMOTE CONTROL CAR	43-44
6.2	ANDROID APPLICATION	45-46
CHAPTER 7	TESTING AND RESULT	47
7.1	DPAD + IR	47
7.1.1	DPAD + IR + TESTING	47
7.1.2	DPAD + IR + RESULT	48-51
7.2	AUTONOMOUS + IR + ULTRASONIC	52
7.2.1	AUTONOMOUS + IR + ULTRASONIC + TESTING	52
7.2.2	AUTONOMOUS + IR + ULTRASONIC + RESULT	53
7.3	MAPPING	54
7.3.1	MAPPING TESTING	54-56
7.3.2	MAPPING RESULT	57-59
CHAPTER 8	CONCLUSION AND DISCUSSION	60
CHAPTER 9	REFERENCE	61-62

LIST OF FIGURES

Figure 2-1: the design of the autonomous car	5
Figure 2-2: SFR08 Beam Pattern	5
Figure 2-3: the design of the autonomous car	6
Figure 2-4: mapping concept art	7
Figure 3-1-1: Pin layout of Atmel Atmega2560	8
Figure 3-1-2: Pin layout of Arduino MEGA 2560	10
Figure 3-2-1: L298N board	11
Figure 3-2-2: Arduino MEGA, DC motor and L298N schematic	12
Figure 3-2-3: Turn Right Code	13
Figure 3-2-4: Wheels Movement: Forward	13
Figure 3-2-5: Wheels Movement: Backward	13
Figure 3-2-6: Wheels Movement: Turn Left	14
Figure 3-2-7: Wheels Movement: Turn Right	14
Figure 3-2-8: H-bridge working principle	14

Figure 3-2-9: H-bridge (Clockwise)	15
Figure 3-2-10: H-bridge (anti-clockwise)	15
Figure 3-2-11: H-bridge (Short circuit)	16
Figure 3-3-1: HC-SR04	17
Figure 3-3-2: Ultrasonic Mapping placing	17
Figure 3-3-3: Schematic of Ultrasonic Mapping	18
Figure 3-3-4: Ultrasonic Mapping Servo Rotation	18
Figure 3-3-5: HC-SR04 basic principle of work	19
Figure 3-3-6: Servo's angle of turning	19
Figure 3-4-1: HC-06	20
Figure 3-4-2: Schematic of connecting Bluetooth module to Arduino	20
Figure 3-5-1: HC-06	21
Figure 3-5-2: Schematic of IR Connect to Arduino	21
Figure 3-5-3: IR placement	22
Figure 3-6-1: Samsung Galaxy s3	23
Figure 4-1-1: Arduino 1.6.4	24
Figure 4-2-1: Eclipse Juno	26
Figure 4-2-2: Eclipse Juno graphical layout tool	27
Figure 4-2-3: bCom Main menu layout	28

Figure 4-2-4: bCom DPAD layout	28
Figure 4-2-5: bCom mapping layout	29
Figure 4-2-6: bCom about layout	29
Figure 5-0-0: Block Diagram of the final product	30
Figure 5-1-1: IR placement	31
Figure 5-1-2: Motor move straight code	31
Figure 5-1-3: Left hand side IR detected obstacle	32
Figure 5-1-4: code for turning left with certain ms	32
Figure 5-1-5: code for reverse with certain ms	32
Figure 5-1-6: Right hand side IR detected obstacle	32
Figure 5-1-7: code for turning right with certain ms	33
Figure 5-1-8: Rare IR detected obstacle	33
Figure 5-1-9: code for go straight with certain ms	33
Figure 5-1-10: Front IR detected obstacle	34
Figure 5-1-11: Compare distance code	34
Figure 5-1-12: Code to run upon bottom IR lost reading	34
Figure 5-2-1: Android application On touch listener code	37
Figure 5-2-2: Arduino command receive code for going straight	37
Figure 5-2-3: Arduino command receive code for stop instantly	37
Figure 5-2-4: Arduino sample code for turning right without setting time	37

Figure 5-3-1: Arduino servo rotation and distance reading code	38
Figure 5-3-2: Sending data to android with for loop	39
Figure 5-3-3: Android: String received	39
Figure 5-3-4: Android: receive data code + append	39
Figure 5-3-5: Android: Break Down string	40
Figure 5-3-6: trigonometry diagram for sensor[1,7]	40
Figure 5-3-7: trigonometry diagram for sensor[3,5]	41
Figure 5-3-8: trigonometry diagram for sensor[2,6]	41
Figure 6-1-1: Front view of remote control car	43
Figure 6-1-2: top view of remote control car	43
Figure 6-1-3: rare view of remote control car	43
Figure 6-1-4: side view of remote control car	44
Figure 6-2-1: main menu	45
Figure 6-2-2: DPAD + autonomous	45
Figure 6-2-3: Mapping	46
Figure 6-2-4: About	46
Figure 7-1-1-1: track for testing	47
Figure 7-1-1-2: obstacle testing	47
Figure 7-1-1-3: stairs testing	47
Figure 7-2-1-1: track for testing	52

Figure 7-2-1-2: test on stairs	52
Figure 7-2-1-3: face obstacle testing	52
Figure 7-3-1-1: rectangular angle scan	54
Figure 7-3-1-2: rectangular angle expected result	54
Figure 7-3-1-3: triangular angle scan	55
Figure 7-3-1-4: triangular expected result	55
Figure 7-3-1-5: rectangular angle with obstacle scan	56
Figure 7-3-1-6: rectangular angle with obstacle scan expected result	56
Figure 7-3-2-1: rectangular angle scan result	57
Figure 7-3-2-2: rectangular angle with obstacle scan result	58
Figure 7-3-2-3: triangular angle scan result	59

LIST OF TABLES

Table 3-1-1: Specification of Atmel ATmega2560	8
Table 3-1-2: Specification of Arduino MEGA 2560	9
Table 3-2-1: L298N Specification	11
Table 3-2-2: L298N int pin	12
Table 3-3-1: Specification of HC-SR04	17
Table 3-4-1: Specification of HC-06	20
Table 3-5-1: Specification of Infrared Sensor	21
Table 3-6-1: Specification of Samsung Galaxy s3	23
Table 5-1-1: Command for autonomous	31
Table 5-2-1: Command table	36
Table 5-3-1: Servo turning and ultrasonic reading	38
Table 7-1-2: test case for dpad	48-51
Table 7-2-2: test case for autonomous	53

LIST OF ABBREVIATIONS

FYP	Final Year Project
PWM	Pulse Width Modulation
UARTS	Micro Architecture
ICSP	In-Circuit Serial Programming
EEPROM	Electrically Erasable Programmable Read-Only Memory
CMOS	Complementary metal–oxide–semiconductor
IR	Infrared
USB	Universal Serial Bus
RC	Radio Control
GUI	Graphical User Interface
DPAD	Direction Pad
RF	Radio Frequency
DC	Direct Current
LHS	Left Hand Side
RHS	Right Hand Side

Chapter 1: Introduction

1.1 Problem statement

Most autonomous cars for exploration purposes are equipped with camera. However, in certain conditions, camera cannot produce the expected result. For instance, if the autonomous car is being deployed in a cave or a place where light is insufficient, the outcome will be not as what we desired as normal camera works best with light unless the car is installed with a light bright enough to shine the place up.

Another option is to use night vision camera which can work in the dark. However, this camera is expensive and does not meet the low cost objective.

Besides, places like caves may hamper Wi-Fi/Bluetooth signal strength which in turn will affect its data transfer performance. Thus, transferring video data is slower than transferring string of depth data, which is used by ultrasonic sensors

1.2 Project objective

This project aims to produce an android remote control car with autonomous and depth mapping functions for hazardous location exploration. The sub-objectives are as below:

1. Design the GUI for the Android application that can have both autonomous and manual control mode. It can also activate and display the output of depth mapping by ultrasonic sensors via Bluetooth.
2. Enable the car to go autonomous and activating the ultrasonic mapping.
3. Construct the car with Bluetooth module to enable communication between Arduino and Android smartphone.
4. Program the car's movement by controlling the rotation of the motor.
5. To place sensors such as infrared and ultrasonic sensors at correct places on the car to achieve maximum efficiency for autonomous car.
6. Translate depth mapping data into graphical representation on Android smartphone.

1.3 Background

Autonomous car has been developed for years since the early 1920s. As years passed by, more and more different types of autonomous car for different purposes were created. It begun with the concept of creating a car that does not require human input once the destination is set and others will be done by the car itself with the help of microcontroller and other sensors.

As time passed by, researchers have taken this concept and apply it on other fields such as replacing human to complete house chores. The autonomous car for vacuum cleaner has the ability to know which area is cleaned and which area is not. Some use it on high speed car race, while some implement it for escaping maze in the shortest time.

Autonomous car can also be used to collect data in various ways. The commonly used method is by using camera which can capture a more accurate data. For an instance, mapping of the floor plan. This method saves time as the car will move on its own and record the camera vision at the same time.

Chapter2: Literature Review

Chomtip Pornpanomchai Member, IACSIT, and Phichate Sukklay did a research with the title "Operating Radio-Controlled Cars by a Computer" 'Chomtip Pornpanomchai, 2011'. In this research, the RC car is controlled with computer and has voice and facial recognition. The strength of this research is that it can do complex programming such as facial recognition and voice processing as command. However, the car needs a computer to control which can be an inconvenience. With the advancement of Smartphone, program can be loaded into the phone and runs anywhere and the size of the smartphone is compact. Another problem is the voice recognition. The voice may not be accurate all the time as different people have different tone and different way of speaking and there for is not very efficient using voice to control the car.

Ritika Pahuja and Narender Kumar also did a research with the title of "Android Mobile Phone Controlled Bluetooth Robot Using 8051 Microcontroller" 'Ritika Pahuja, N.D.'. In many ways, this project is very similar to what we are going to develop. The way it operates is just by sending a simple command from the phone such as "E" to the microcontroller to stop. The downside of this car is that the car they made can only move in simple 4 directions. They should fully utilize the microcontroller by extend extra feature such as build in with autonomous feature. The microcontroller can still accept extra input such as IR sensors, ultra-sonic and etc.

Junghoon Ha aka Eric Ha, Stevanes Hermawan and Willie Pramono came up with the research titled "Parallel Parking R/C Car" 'Junghoon Ha, N.D. '. This project emphasize on making the RC car to park automatically. This requires a very powerful algorithm to make this project works and that only to rely on the IR sensors. This project can be even better with the help of ultra-sonic as just IR sensor may not be accurate enough to take in the variable needed. IR sensor itself has a very short range of detection. So to see further, we need either ultra-sonic or camera.

Journal from University of Idaho has done a research title 'Creating a low cost autonomous vehicle' autonomous car 'Richard W. Wall, N.D. '. The autonomous car in this journal is done by implementing GPS (Garmin GPS 16-LV), ultrasonic (Devantech SRF08) and the data is processed by Rabbit 2000. While the GPS is used to control the car to move to

desire location, ultrasonic is used to detect obstacle that is in the way while traveling to the destination.

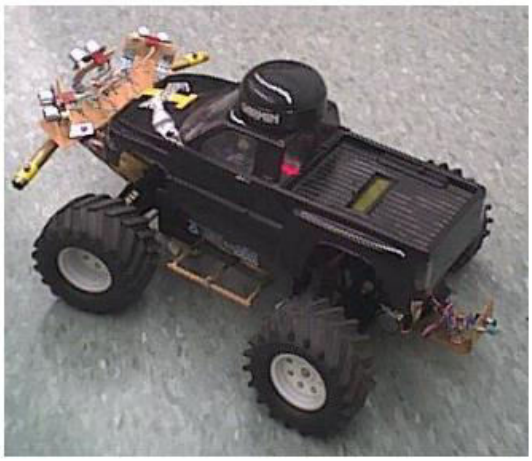

Figure 2-1: the design of the autonomous car

GPS (Garmin GPS 16-LV) is used as it provides the benefit of retrieving location information that has no bearing upon previous result. The GPS is low cost and ease to use which is the main reason it is choose to be used in this project.

For obstacle detection, ultrasonic (Devantech SRF08) is used. The reason for that is their ability to perform under low visibility and their scalability to perform underwater application. The ultrasonic range finder has a good depth resolution.

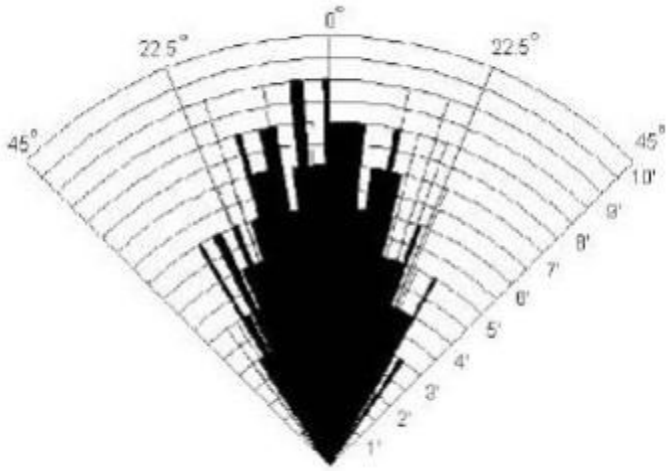

Figure 2-2: SFR08 Beam Pattern

The car is control by 2 motors. One of the motor is for steering and the other for speed. The steering motor uses Futaba s3003 while the speed motor uses standard motor that comes with the remote control car.

There are 3 things that this car will perform. First is the GPS will stream data to the microcontroller once every second while the second is the ultrasonic sensor can be pinged every 75ms and lastly the electronic speed control will receive an update once every 20 ms. As all this has different timing requirement and to ensure all this works properly, its control algorithm was implemented using a multitasking structure, which uses 3 independent loops. All this loops are executed in parallel. As the loops are running at its own pace, this ensure the data receive are the most current information.

This car can perform very well under standard condition. However, GPS was meant to be used outdoor where GPS can pick up signal. This could be difficult especially when the place is cave where the signal could not be pick up. More sensors such as infrared sensor could be included to the autonomous car to help avoid hitting into walls. The design of this car is meant for track, but not for exploration as the car has its limit making sharp turn.

Journal from University Bochum has done a research title ‘A smartphone controlled autonomous robot’ autonomous car ‘Christian Bodenstein,2015’. They offer an autonomous car which knows surrounding by using a smartphone as the brain of the car and as for its eye, they are using 2 piece of mirror place at different distance to construct a 2D picture and act from it. The camera of the phone is place to face the camera so snapshot can be done to save the image for process.

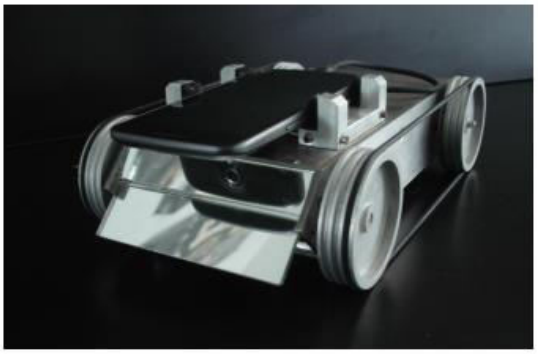

Figure 2-3: the design of the autonomous car

Other than using camera to understand its surrounding, the autonomous car is equipped with Odometry which is used to calculate its distance traveled. As the car doesn't equipped with any other sensor, Odometry plays a very important role in tracking how far the car has travel and what distance is needed to travel.

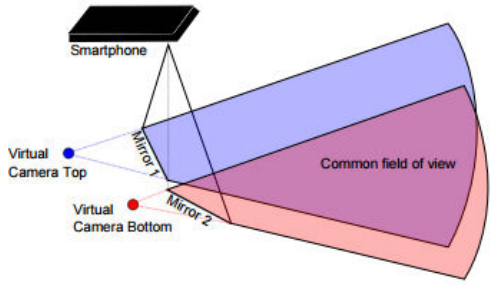

Figure 2-4: mapping concept art

This car design makes the car has the ability to make a very sharp turn. This is especially useful in exploring purpose as a lot of situation, the car is expected to make sudden turn instead a turn which needs a little distance to make the turn thanks to the rotation of the motor. The way they car know its surrounding is also quite interesting as this method uses mirror and some complex calculation, the actually manage to see the location in a 2d manner. However, since this is done with phone camera and mirror, it is only limited to open area with sufficient light as camera without light does not work perfectly.

Chapter 3: Hardware Development

3.1 Microcontroller and Development board

The microcontroller used in this project is ATmega2560. This microcontroller is developed by Atmel Company. The ATmega2560 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. Table 3-1-1 shows ATmega2560 specification.

Device	Atmel ATmega2560
Pins	86 Programmable I/O Lines
Operating Voltage	4.5 V to 5.5 V
Timer	6
PWM	Four 8-bit PWM Channels
ADC channel/resolution	16-channe, 10 bit
Communication interface	4 Programmable Serial USART

Table 3-1-1: Specification of Atmel ATmega2560

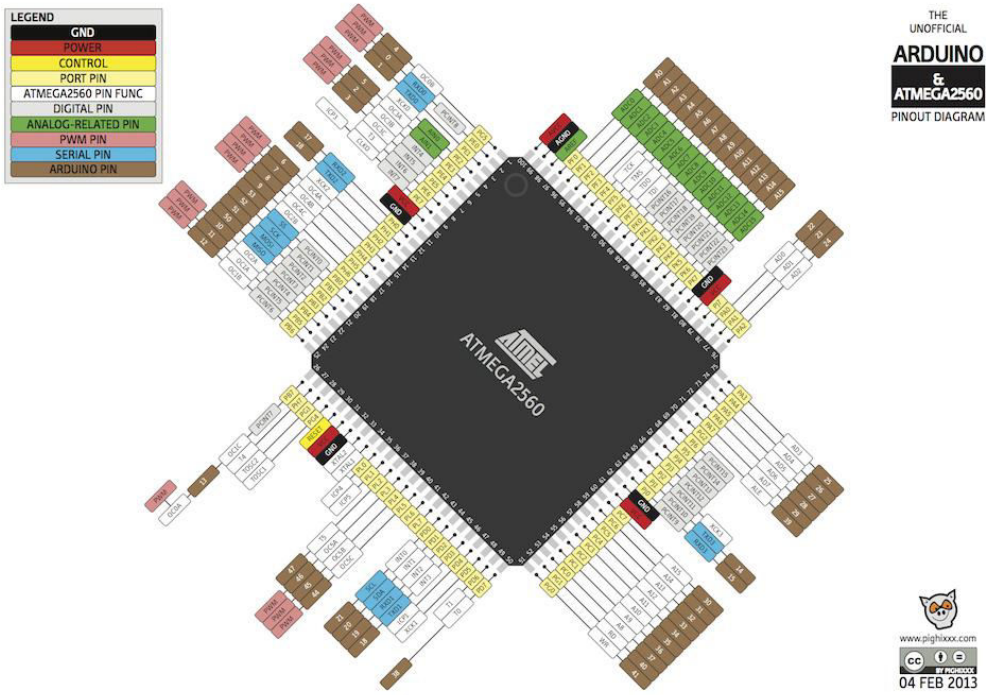

Figure 3-1-1: Pin layout of Atmel Atmega2560

The Microcontroller board used in this project is Arduino MEGA 2560. It acts as the controller and the processing unit for the RC car. It is a microcontroller based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. Table 3-1-2 shows Arduino MEGA basic specification.

Operating voltage	5V
Input Voltage(recommended)	7-12V
Input Voltage(Limit)	6-20V
Digital I/O pin	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current for 3.3V Pin	50 mA
DC Current per I/O Pin	20 mA
Clock Speed	16 MHz
Flash Memory	256 KB of which 8 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Weight	37 g
Length	101.52 mm
Width	53.3 mm

Table 3-1-2: Specification of Arduino MEGA 2560

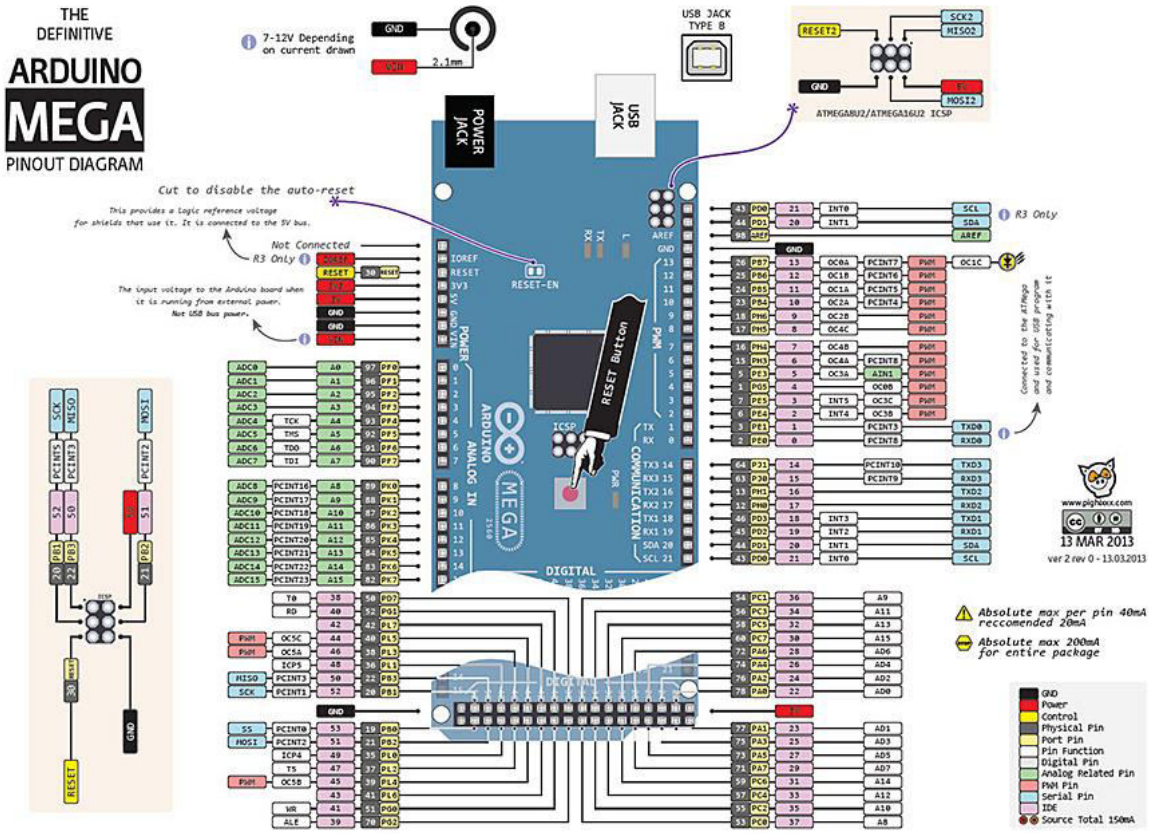

Figure 3-1-2: Pin layout of Arduino MEGA 2560

3.2 L298N motor driver

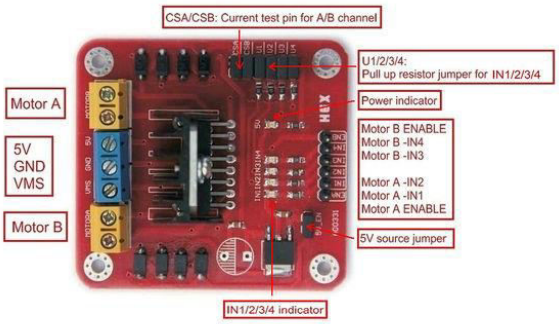

Figure 3-2-1: L298N board

Power Supply	5V~46V
Power Output Vss	5V~7V
Controlling level	Low -0.3V ~ 1.5V High 2.3V ~ Vss
Enable Signal	Low -0.3V ~ 1.5V High 2.3V ~ Vss
Max Power	25W
Logic Current	0 ~ 36mA

Table 3-2-1: L298N Specification

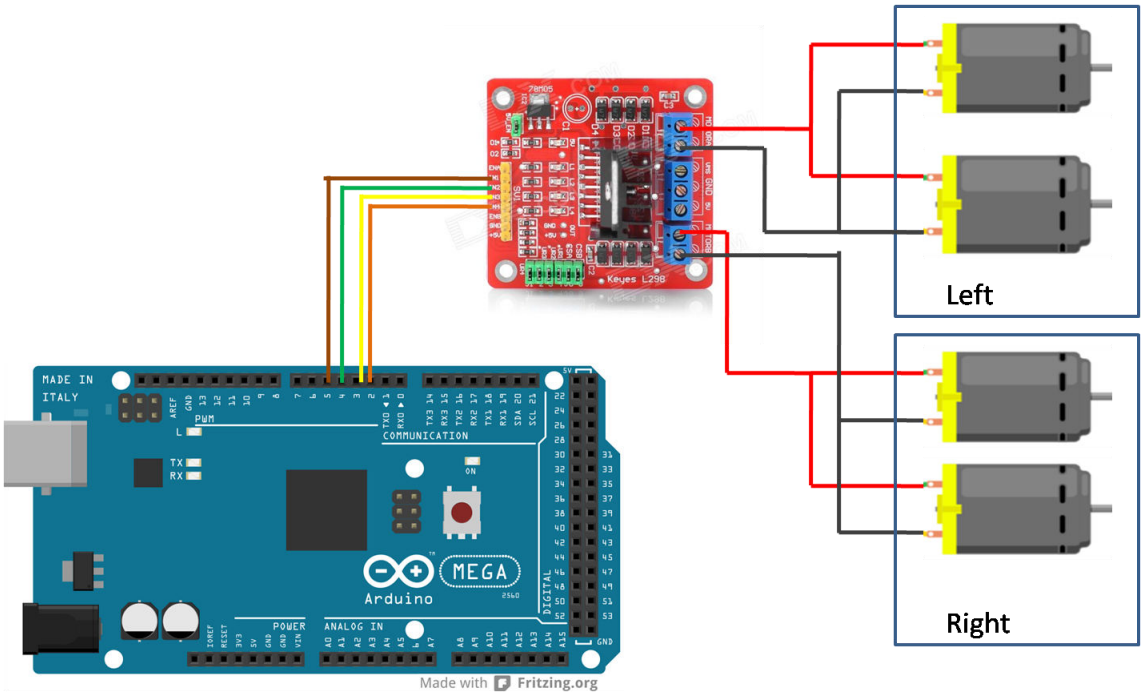

Figure 3-2-2: Arduino MEGA, DC motor and L298N schematic

The remote control car runs on 4 wheels which have a DC motor individually. Each of them can be control to turn clockwise or anti clockwise direction. For this project, the DC motor in parallel is wired to one output port as both the DC motors will move in the same direction.

To control the rotation of the DC motors, L298N is used in this project. As mentioned, the DC motor on the same side is wired to one output port. With the correct direction, it enables the RC car to move in different directions.

The L298N has 4 int pins which are used to control the movement of the motor. For an instance, pin 4(int 2) is used to make the left hand side motor to reverse while pin5 (int1) is used to make the LHS motor to rotate forward.

Arduino Pin	L298N Pin	Pin Name	movement	Motor
2	Int4	P3	Forward	RHS
3	Int3	P4	Reverse	RHS
4	Int2	P1	Reverse	LHS
5	Int1	P2	Forward	LHS

Table 3-2-2: L298N int pin

Normally if the LHS motor is set to turn right, pin P1 will be set low while P2 will be set high. When both are set to high, it will create a situation like handbrake. When both set to low, it will stop the car. For instance, the RC car does not stop instantly despite object is detected.

```
void t2_rightC(){
  digitalWrite(P1, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P3, LOW);
  digitalWrite(P4, HIGH);
}
```

Figure 3-2-3: Turn Right Code

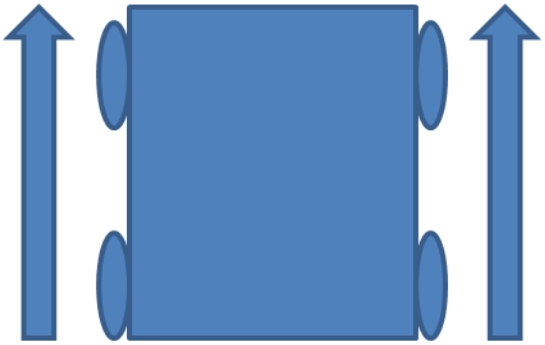

Figure 3-2-4: Wheels Movement: Forward

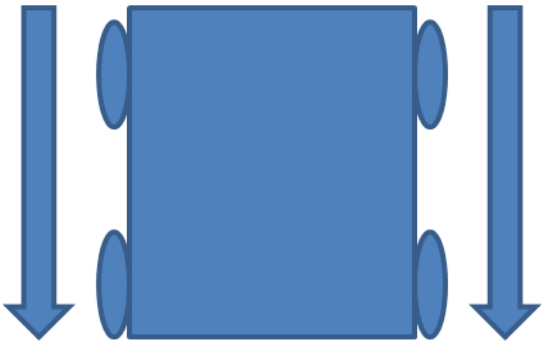

Figure 3-2-5: Wheels Movement: Backward

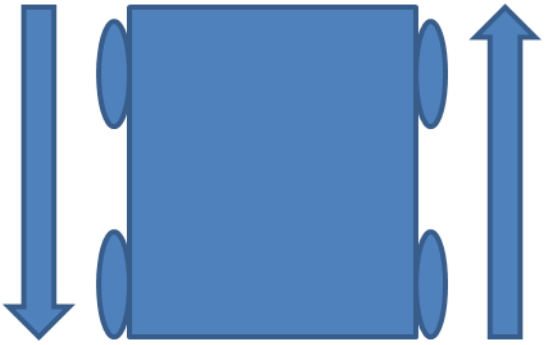

Figure 3-2-6: Wheels Movement: Turn Left

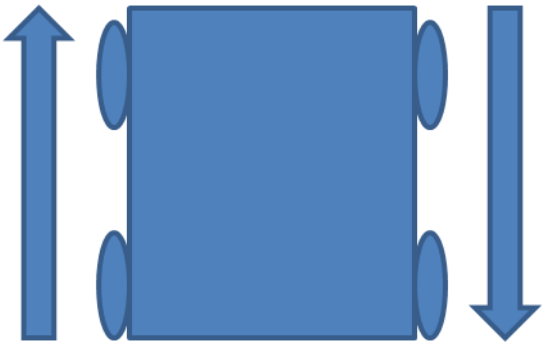

Figure 3-2-7: Wheels Movement: Turn Right

The rotation of the motor is controlled by H-bridge. This electronic circuit allows a voltage to be applied across a load in either direction. This results in the DC motors to move clockwise or anti clockwise.

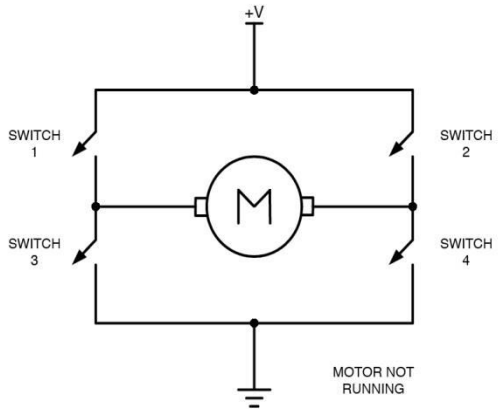

Figure 3-2-8: H-bridge working principle

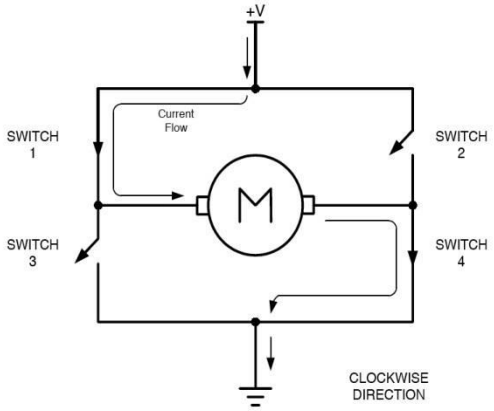

Figure 3-2-9: H-bridge (Clockwise)

To turn clock wise, switch 1 and switch 4 will be closed (turn on), results in left lead of the motor to connected to power supply while right side lead to connected to ground. Current starts to flow and the motor begin to rotate in a positive direction.

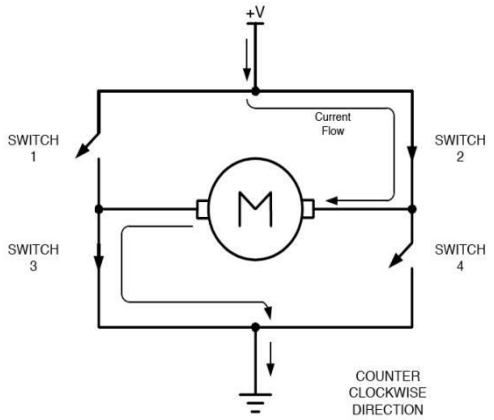

Figure 3-2-10: H-bridge (anti-clockwise)

To turn anti clockwise, switch 1 and switch 4 will be will be open(turn off), results in right lead of the motor to connected to power supply while left side lead to be connected to ground. This will cause the DC motor to energize in the reverse direction which leads to spin backward.

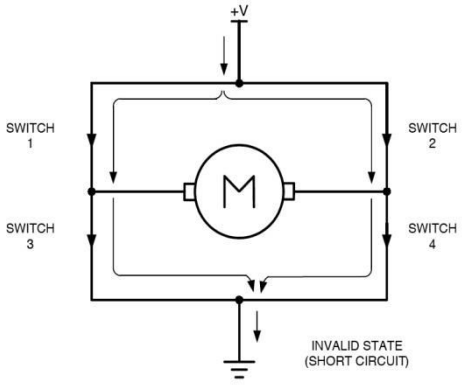

Figure 3-2-11: H-bridge (Short circuit)

All switches should never turn on at the same time as it will create a really low resistance path between power and ground which will lead to short-circuit the power supply. This will damage the h-bridge.

3.3 HC-SR04 ultrasonic

Figure 3-3-1: HC-SR04

Working Voltage	DC 5V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
Measuring Angle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion

Table 3-3-1: Specification of HC-SR04

Another feature of this project is that it is equipped with an ultrasonic mapping function. This ultrasonic mapping function is done with 1 pair of ultrasonic sensors and this pair of ultrasonic will be fixed on a short clock-like frame as shown in figure 3-3-2.

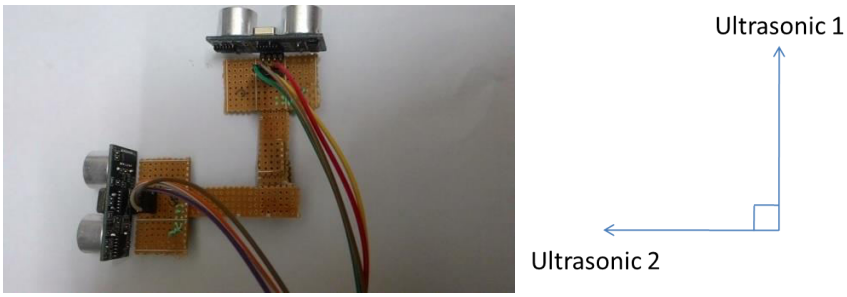

Figure 3-3-2: Ultrasonic Mapping placing

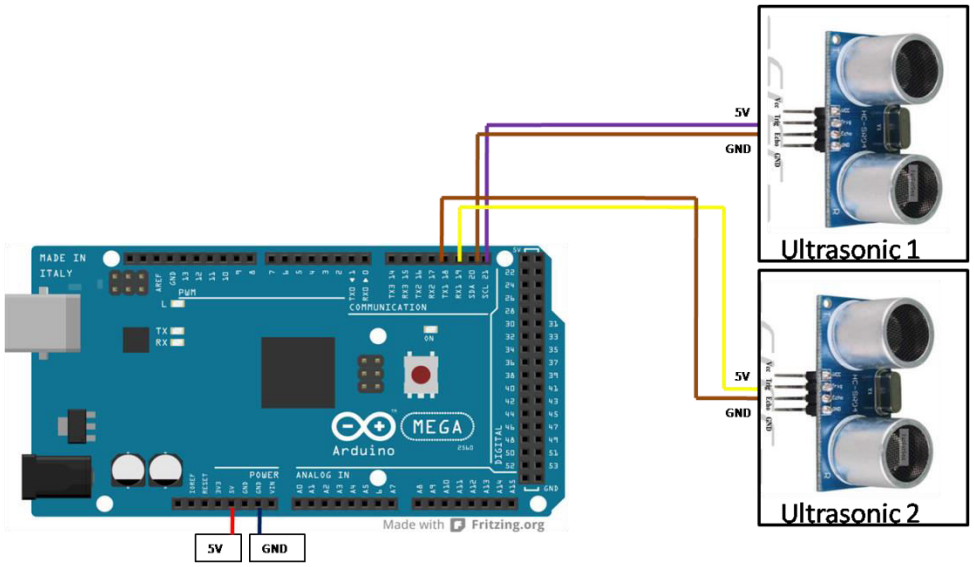

Figure 3-3-3: Schematic of Ultrasonic Mapping

The frame is done by using 2 thin and long pieces of PCB board. This function can be done with just one ultrasonic. However, to speed up the process, two sensors are used each sensor can cover 90°. This way, the servo only needs to rotate 90°instead of 180° to cover whole 180°. Ultrasonic 2 will activate first as the instruction in Arduino runs serially. After Ultrasonic 2 obtains the time taken, Ultrasonic 1 will then be executed. The rotation of the servo and the ultrasonic mapping is shown in figure 3-3-4.

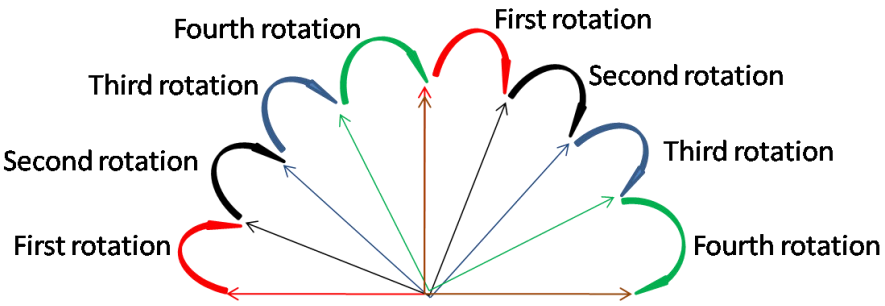

Figure 3-3-4: Ultrasonic Mapping Servo Rotation

Ultrasonic return a time taken for the ultrasound to return the echo in turn translates into depth data. The depth data will be represented in a graph.

The basic principle work of ultrasonic is that 10us will be supplied to the trigger input. After that 8 cycle burst of ultrasound (40Hz) will be emitted from the module. Once an object is detected, ultrasound will bounce back to echo to be received with the time taken for the

ultrasound to reach echo. With the time taken, distance can be calculated. figure 3-3-5 shows how ultrasonic sensor works.

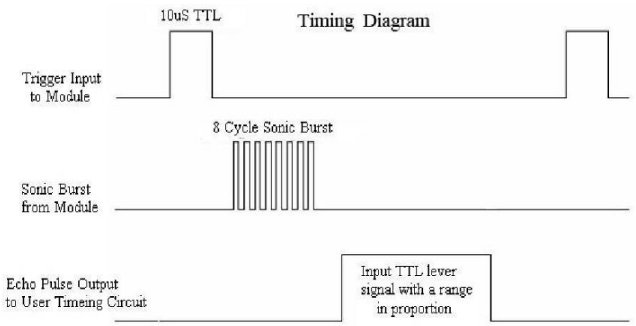

Figure 3-3-5: HC-SR04 basic principle of work

To calculate the depth, other than distance, there is still another variable missing and that is the angle of the turning. With this, we can calculate coordinate x and coordinate y for graph plotting purpose. To get that, the servo is set to turn 22.5°. The angle of turning is shown in the figure 3-3-6.

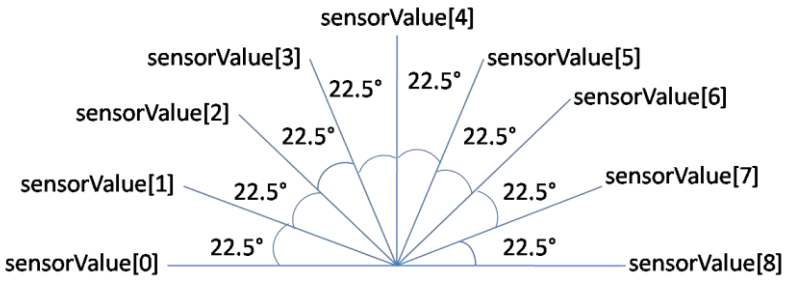

Figure 3-3-6: Servo's angle of rotation

3.4 HC-06

Bluetooth is used in this project to transmit and receive data between Android application and Arduino. To enable Bluetooth connection with the Arduino Mega, a Bluetooth module HC-06 as shown in figure 3-4-1 is used.

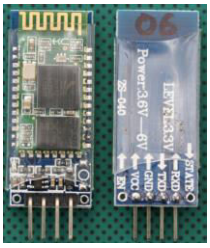

Figure 3-4-1: HC-06

Frequency	2.4GHz ISM Band
Power Supply	3.3VDC 50mA
Emission Power	≤4dBm ,class 2
Sensitivity	≤-84dBm at 0.1% BER
Speed	Asynchronous: 2.1Mbps(Max)/160kbps Synchronous: 1Mbps/1Mbps
Bluetooth protocol	Bluetooth Specification v2.0 + EDR
Signal coverage	30 ft
Default Baud Rate	9600 , 8 , 1 , n

Table 3-4-1: Specification of HC-06

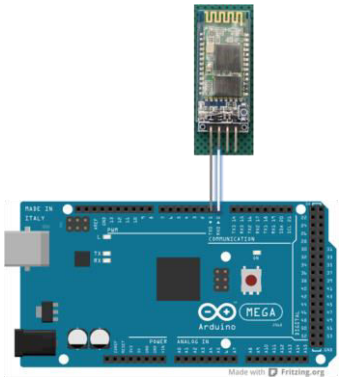

Figure 3-4-2: Schematic of connecting Bluetooth module to Arduino

3.5 Infrared Sensor

Figure 3-5-1: HC-06

Operating Voltage	DC 3.3V ~ 5V
Operating Current	Try to choose more than 1A power supply
Working temperature	-10° ~ +50°

Table 3-5-1: Specification of Infrared Sensor

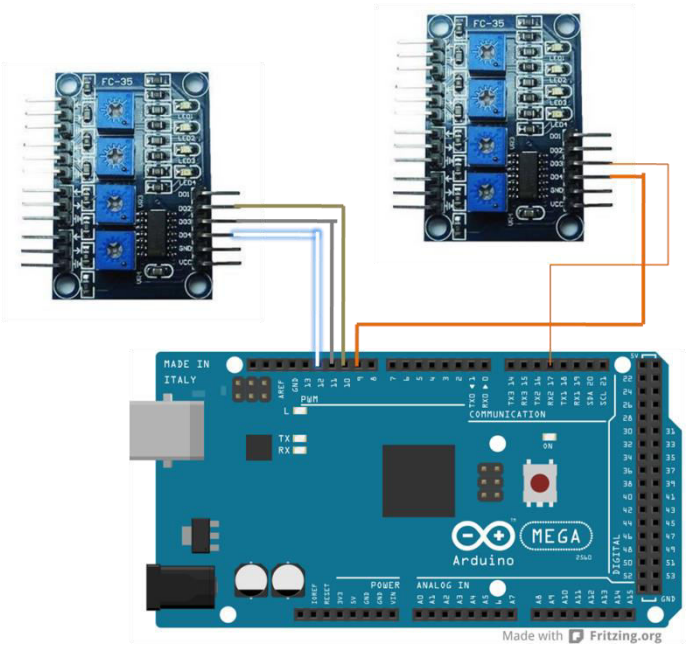

Figure 3-5-2: Schematic of IR Connect to Arduino

Five infrared sensors are placed at a few crucial angles of the RC car. The reason of using these IR sensors is to help avoid obstacle in either DPAD mode or Autonomous mode. The placing is shown as figure 3-5-3.

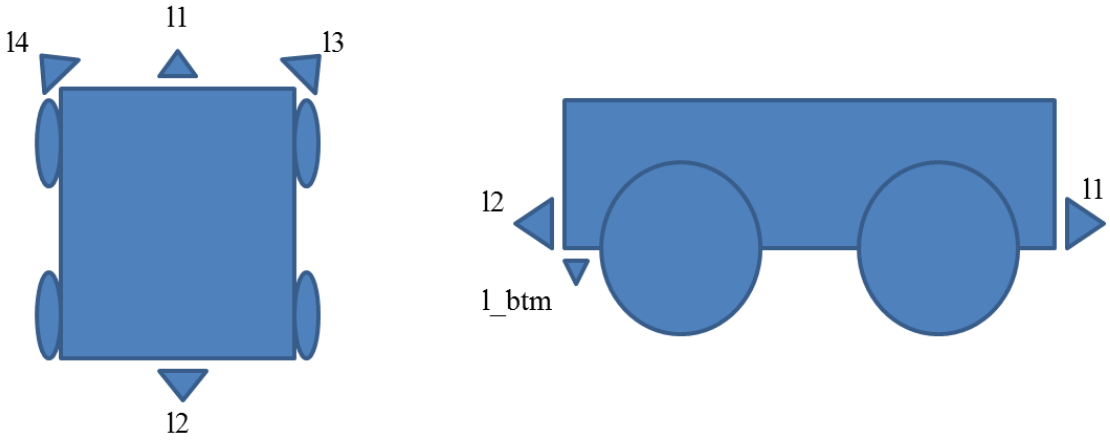

Figure 3-5-3: IR placement

3.6 Smartphone (Samsung galaxy S3)

Figure 3-6-1: Samsung Galaxy s3

Samsung Galaxy S3 specs	
Network	GSM 850/900/1800/1900 MHz, UMTS 850/900/1900/2100 MHz,
Data:	HSDPA 21 Mbps, HSUPA 5.76 Mbps; 4G [LTE is region dependent]
Form factor:	Bar design, Touchscreen
Dimensions:	136.6 x 70.6 x 8.6 mm
Weight	133 grams
Display:	4.8" (720 x 1280 pixels)
Colour output:	16.7 million colours
Screen type	Super AMOLED - Pentile Display
Glass:	Gorilla Glass
CPU:	Quad-core 32nm ARM Cortex A9 1.4 GHz processor, Exynos 4212 Quad chipset
GPU:	Mali-400MP
RAM:	1GB
OS:	Android 4.0.4 (Ice Cream Sandwich)
Memory:	16/32/64GB internal memory, MicroSD card support
Camera:	8 megapixel auto-focus camera [BIS type]
Other camera features:	Face detection, Touch to focus, image stabilization, Full HD (1080p) recording at 30fps, single LED flash, 1.9MP front facing camera, video-call capability. HD video recording from FFC
Connectivity:	Wi-Fi a/b/g/n [WiFi hotspot capability], Bluetooth 4 with HS, MicroUSB port [MHL variety]
Others:	GPS with A-GPS capability, GLONASS support, 3.5mm audio jack, TV-Out, USG OTG, NFC, Accelerometer, proximity sensor, gyroscope sensor, RGB light,Induction charging (wireless charging), NFC based WiFi direct, DLNA, Gesture controls, Voice inputs for certain features, FM radio
Misc:	TouchWiz 4.0 UI, DivX/XviD codec support, multi-touch input, Swype text input
Battery:	2100 mAh
Talktime:	n.a
Standby:	n.a

Table 3-6-1: Specification of Samsung Galaxy S3

For this project, a smartphone is required to send command and performing calculation. So the phone needed have sufficient RAM and CPU to process it. With quad-core and 1 GB RAM, it is sufficient for this project. An Android application will be written for this project and will run on this smartphone.

Chapter4: Software Development

4.1 Arduino

Arduino is used to control the rotation of the motor, to process the distance of ultrasonic and receive command from the Android application.

Arduino is an open source platform which is easy to use and a lot of library is provided. Compared to other microcontroller, it has a variety of different microcontroller build for specific purposes. Arduino itself also has its own Arduino development environment which is free and easily available online. Arduino is a development board whereas microcontroller like 8051 is just a microcontroller. Arduino simplifies the amount of hardware and software development needed to do in order to get the system running.

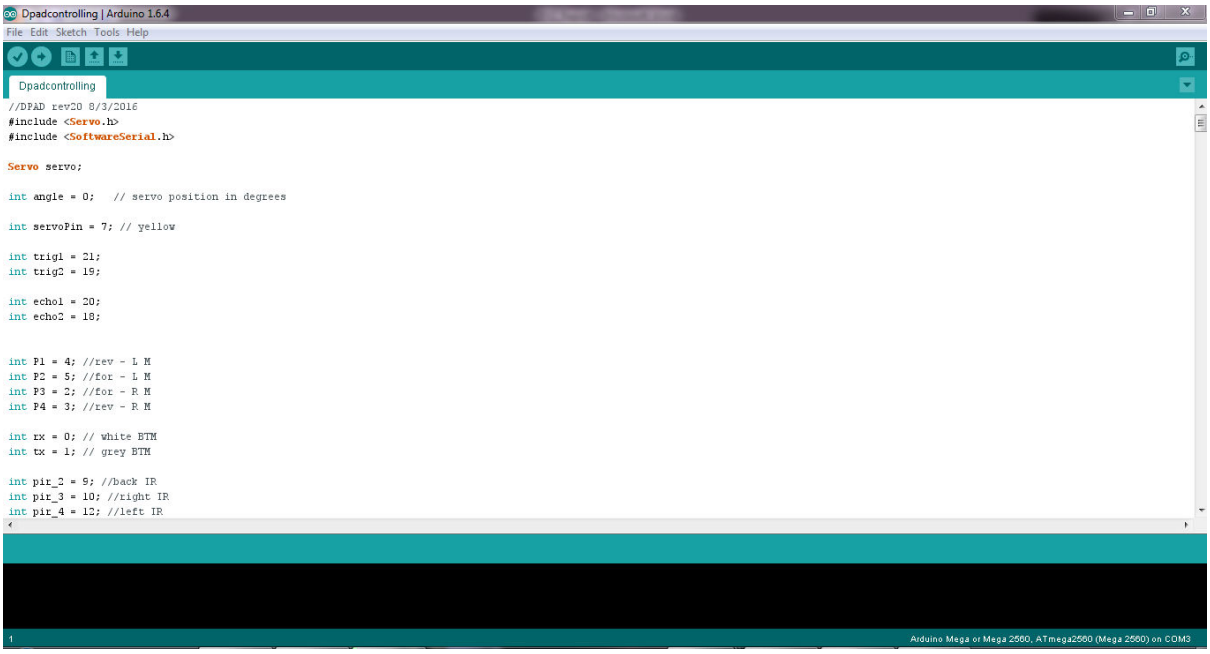

Figure 4-1-1: Arduino 1.6.4

When first used, the driver for the Arduino Mega needs to be installed by just plugging in the board. The port number needs to be set correctly before transferring the code from computer to the microcontroller.

Once the code is done, we can transfer the code by just pressing the upload button and the code will be directly transfer to the microcontroller. However, when transfering, it is

important to remove the RX and TX pin for the Bluetooth module as upload will fail if it is not remove.

For debugging purpose, the Arduino is equiped with a serial monitor for user to check. For instance, one is able to track what data is passing through the Bluetooth connection by writing "serial.print". This is to ensure the data transfer is correct and is successfully received at both ends.

4.2 Eclipse Juno

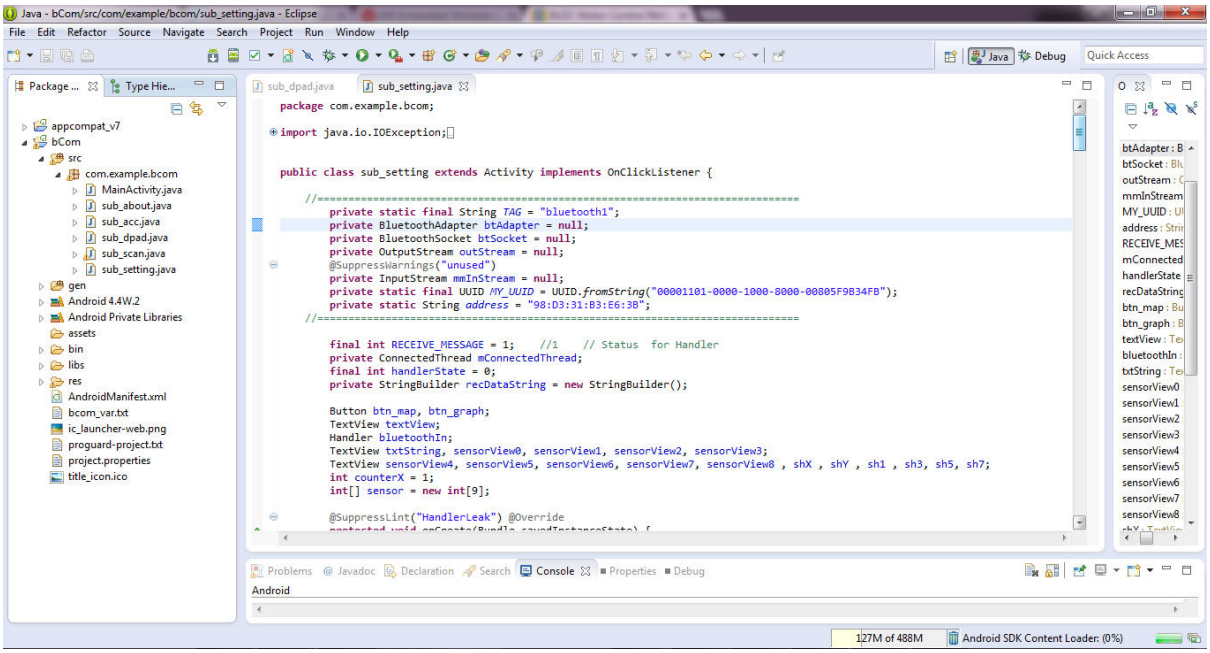

Figure 4-2-1: Eclipse Juno

Eclipse is open source software for user to do project. They also have variety of plug-in for users to download and apply for their own need. One of the plugins used in this project is the Android SDK, which allows users to write and compile Android application for smartphone.

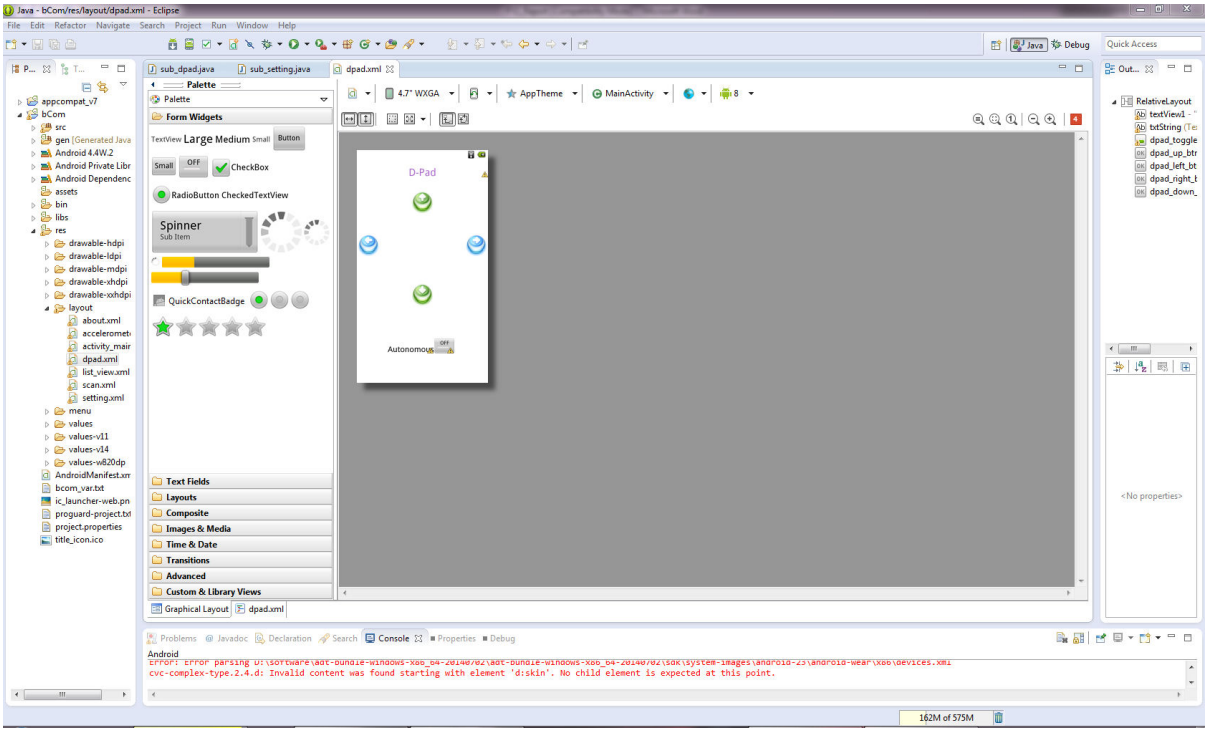

Figure 4-2-2: Eclipse Juno graphical layout tool

In this project, commands are sent and received between Android application and Arduino. The main role of this mobile application is to send simple command for DPAD controlling. At the same time, a long string from Arduino will also be received by the Android application to break down into chunks of data and then calculate the depth coordinate with a simple trigonometry algorithm.

In this project, the Android version used to develop is Android 2.2. The reason for not using the newer version available is to produce the Android application that can support older smartphone which has older Android version.

The Android application is named as bCom (Bluetooth controller on mobile). The first layout is the main menu. The layout is comprised of a few button. In the application, it is equipped with a toggle button for user to on/off the Bluetooth. The design is as shown as figure below.

Figure 4-2-3: bCom Main menu layout

When the "D-pad" button is pressed, the program will trigger a new intent which will call dpad.xml which is linked with sub_dpad.java. This is the part where the command is sent upon button pressed to the Arduino for moving forward (F), backward (B), left (L) and right (R). To activate autonomous mode, there is a Toggle Button which when pressed once, it will activate it by sending a command (Y) and when pressed the second time, it will send command (N) to stop the car. For continuous turning, users just have to press and hold the button and the car will continue to send the command until the button is released, and the car will then stop. This is done by on touch listener.

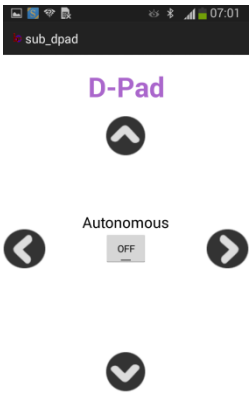

Figure 4-2-4: bCom DPAD layout

Upon clicking the mapping button in the main menu, the Android application will call the mapping layout. The use of this mapping function is to read the ultrasonic and plot them in a graph to show the shape of the location it scans. The reading will be written to the text view to allow user to check the reading for the calculation. The calculated coordinate will be store in the text view located at the lower part of the application. Figure 4-2-5 shows the layout of the mapping function.

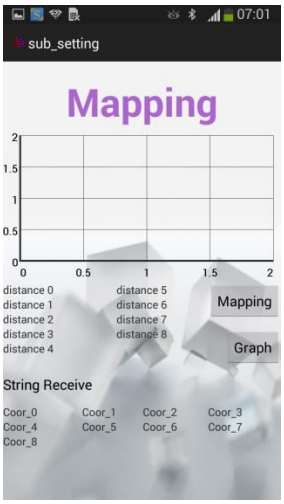

Figure 4-2-5: bCom mapping layout

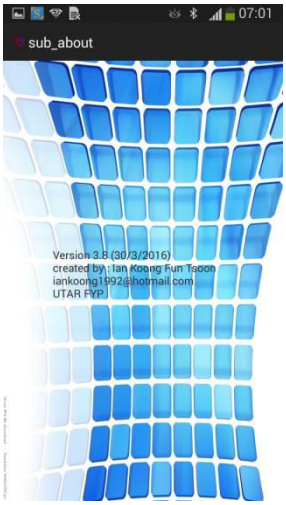

Figure 4-2-6: bCom about layout

Chapter5: System Integration

To integrate the hardware and software, all the parts are combined and the code will be burned/flashed in to the Arduino as well as the Android application into the android phone. Below shows the block diagram on how the final product will be.

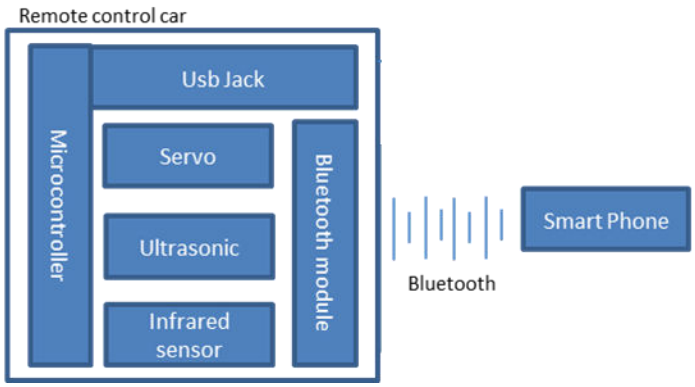

Figure 5-0-0: Block Diagram of the final product

The Bluetooth module will be the main communication channel between Arduino and Android application. The USB jack is used to flash the code and for debugging purpose. It is important to make sure the data sent and received correct as it will affect the RC car's overall performance.

5.1 Infrared sensor + Autonomous Mode

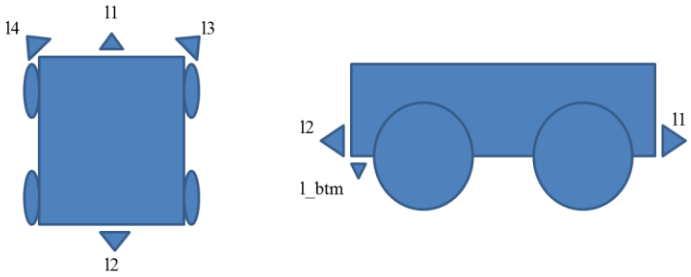

Figure 5-1-1: IR placement

To start the autonomous mode, a toggle button in the DPAD layout can be pressed. The moment it is pressed, the Android application will send a character "Y" to the Arduino and the Arduino will then check which action to perform. In this case, the car motor will be set to go forward. The code is shown in figure 5-1-2.

Command character	Action
Y	Start autonomous
N	Stop autonomous

Table 5-1-1: Command for autonomous

```
void t2_str(){
  digitalWrite(P1, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P3, HIGH);
  digitalWrite(P4, LOW);
}
```

Figure 5-1-2: Motor move forward code

Pressed the toggle button again, and the car will stop moving as the Arduino will receive an "N" which indicate stop.

In autonomous mode, the car is set to move forward by default. In order to make the car be able to move, infrared sensors are used to help navigating the car instead of users. For instance, if an object or wall is on the left hand side, and the l4 detected it, the car will automatically turn to right to avoid it. The principle of infrared sensor is that it will transmit

the IR signal in the direction it is facing and then IR signal will bounce back from the surface of the object once the object is detected.

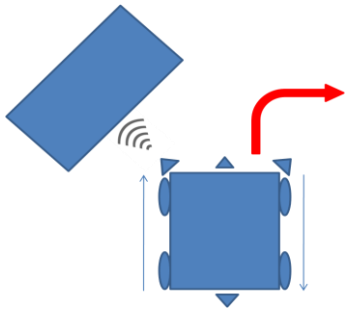

Figure 5-1-3: Left hand side IR detected obstacle

```
void t2_right(int timeDelay){
  digitalWrite(P1, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P3, LOW);
  digitalWrite(P4, HIGH);
  delay(timeDelay);
  digitalWrite(P1, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P4, LOW);
}
```

Figure 5-1-4: code for turning left with certain ms

```
void t2_rev(int timeDelay){
  digitalWrite(P1, HIGH);
  digitalWrite(P2, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P4, HIGH);
  delay(timeDelay);
  digitalWrite(P1, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P4, LOW);
}
```

Figure 5-1-5: code for reverse with certain ms

l4 is used to detect object or walls on the left hand side while moving forward. When l4 detected object, the wheel will make a short reverse for 100ms, then make a right turn (200ms). The same will happen when l1 and l4 pick up values at the same time.

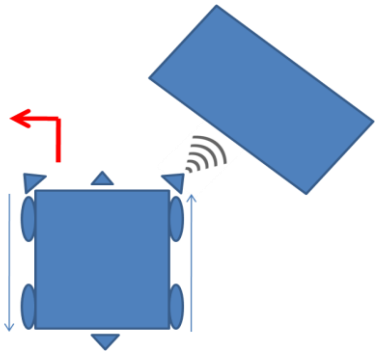

Figure 5-1-6: Right hand side IR detected obstacle

```
void t2_left(int timeDelay){
  digitalWrite(P1, HIGH);
  digitalWrite(P2, LOW);
  digitalWrite(P3, HIGH);
  digitalWrite(P4, LOW);
  delay(timeDelay);
  digitalWrite(P1, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P4, LOW);
}
```

Figure 5-1-7: code for turning left with certain ms

l3 is used to detect object or walls on the right hand side while moving forward. When l3 detected object, the wheel will make a short reverse for 100ms, then make a left turn (200ms). The same will happen when l3 and l1 pick up value at the same time.

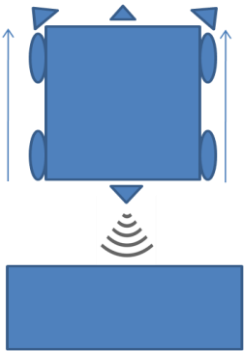

Figure 5-1-8: Rare IR detected obstacle

```
void t2_strTime(int time){
 digitalWrite(P1, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P3, HIGH);
  digitalWrite(P4, LOW);
  delay(time);
  digitalWrite(P1, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P4, LOW);
}
```

Figure 5-1-9: code for go straight with certain ms

l2 is used to detect obstacle from behind. If the infrared detected something from the back, the car will stop and move forward for 400ms.

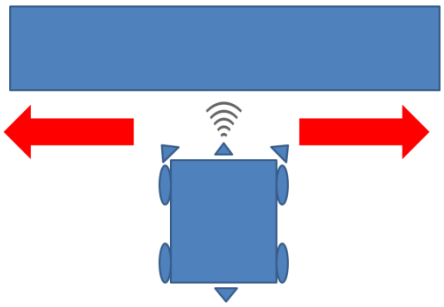

Figure 5-1-10: Front IR detected obstacle

l1 is used to detect when the car meets a T junction. When this happen, the first thing the car will do is to trigger the car to reverse some distance for 800 ms.
Then ultrasonic sensors will be triggered, and it will read the distance of both left and right. The decision is made by comparing distance from both sides and will choose the longest distance to turn to.

To turn the servo to the right, the servo will be set write with value 0. Then ultrasonic sensor will be activated and scanning is performed. To turn the servo to the most left, the servo will be set write with value 85. Then the values will stored as distance1 and distance2 and then compared to find the longest distance. Then the car will set to turn to the direction with the longest reading. The code for turning is set to turn only 200 ms to ensure not to turn too much. The code to choose the longest distance is as shown figure 5-1-11.

```
if(distance1>distance2)
{
  t2_right(200);
  if(count<1){
    t2_strTime(100);
  }
  else{
    //t_straight();
    t2_str();
  }

}
else if (distance1<distance2){
  //t_left(scroll_speed,200);
  t2_left(200);
  if(count<1){
    //t_forward(scroll_speed,100);
    t2_strTime(100);
  }
  else{
    //t_straight();
    t2_str();
  }
}
else{
 t2_rev(300);
```

Figure 5-1-11: Compare distance code

```
    else if(l_btm == 1){
      t2_stop();
      t2_rev(100);
      t2_left(800);

      if(count < 1){
        t2_strTime(200);
      }
      else{
        t2_str();
      }

    /    Serial.println(count);

     }
```

Figure 5-1-12: Code to run upon bottom IR lost reading

l_btm is used as an anti-fall method. There is scenario where the car might fall from stairs and this IR is used to prevent that. Upon receiving signal, the Arduino will start comparing through if else case. The code for this is to stop the car instantly by setting the entire pin to high, which act as a hand brake. The car will then reverses for 100ms and steer 180° to go against the stairs.

5.2 Direction pad

For controlling with DPAD, the Arduino receives commands from smartphone in string form. For an instance, "F" is for moving forward. Upon detecting button press on the smartphone, the Android application will send a character to the Arduino through the Bluetooth. Once the Arduino receive the character command, it will check the if else case and react. table 5-2-1 shows all the command character for different purposes.

Commmand	Command Character
Forward	F
Reverse	B
Left	L
Right	R
Stop	O
Continous Forward	S
Continous Reverse	Z
Continous Left	P
Continous Rght	K

Table 5-2-1: Command table

Since this is a RC car, the button needs extra work to model the touch and hold scenario. To fix that, the button is implemented with on touch listener. This listener will detect the touch on the screen. While the user is still pressing, the Android application will continuously send the command to Arduino till the user release the button. In the Arduino, while Android application continues to send the same command, it will continue to trigger the command code. When the finger is released from the screen, Android application will send another command “O” to Arduino which will call the stop function. The code is shown in figure 5-2-1.

```
btn_up = (Button) findViewById(R.id.dpad_up_btn);
btn_up.setOnClickListener(this);
btn_up.setOnTouchListener(new OnTouchListener() {
    @SuppressLint("ClickableViewAccessibility") @Override
    public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_DOWN) {
            mConnectedThread.write("S");
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            mConnectedThread.write("0");
        }
        return false;
    }
});
```

Figure 5-2-1: Android application On touch listener code

```
if(incomingByte == 'S') {
  t2_str();
  if(count<1){
  }
  else{
    t2_str();
  }
}
```

Figure 5-2-2: Arduino command receive code for going straight

```
if(incomingByte == '0') {
  t2_stop();
}
```

Figure 5-2-3: Arduino command receive code for stop instantly

Each function code comes in two versions. One is time delay and one without time delay. The code with time delay is used for autonomous while without time delay is code for normal DPAD control usage. Figure 5-2-4 shows code for turning right continuously.

```
void t2_rightC(){
  digitalWrite(P1, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P3, LOW);
  digitalWrite(P4, HIGH);
}
```

Figure 5-2-4: Arduino sample code for turning right without setting time

5.3 Ultrasonic + Servo + Depth Mapping

As for the mapping, as the button is pressed, the Android will transfer "M" character to the Arduino will run the code as shown in the figure below.

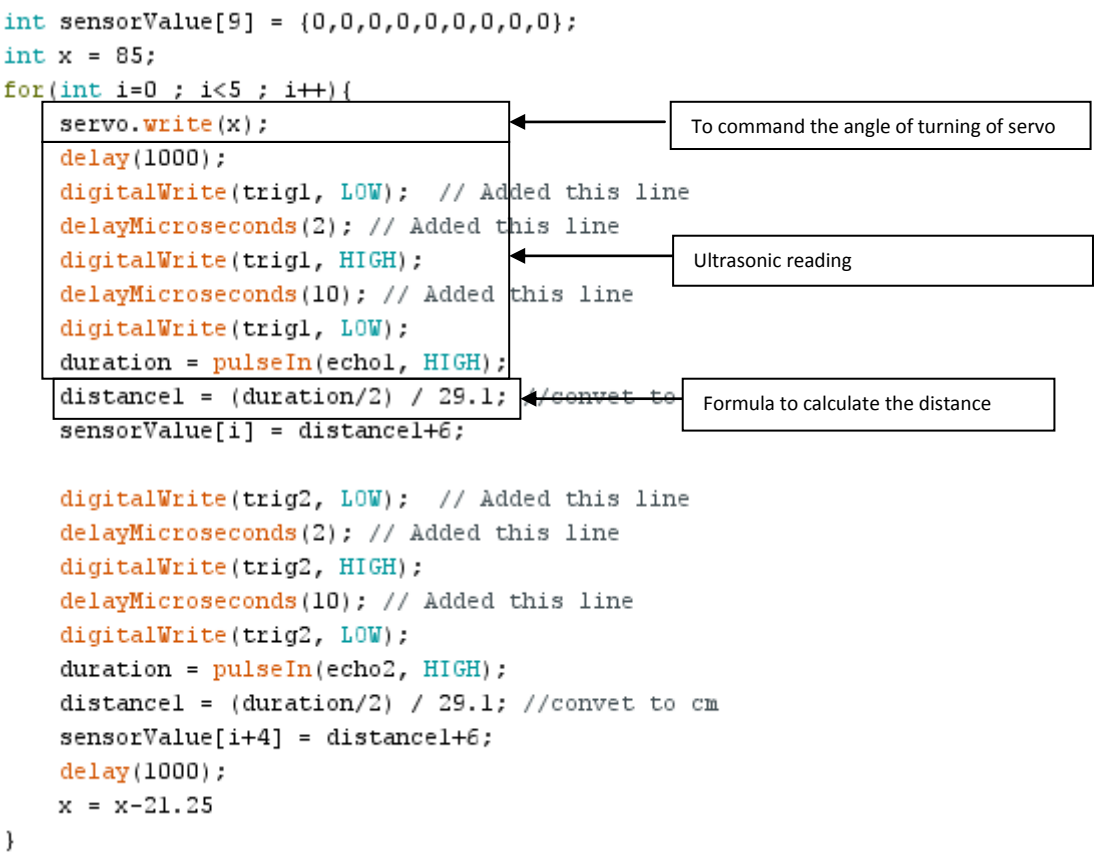

Figure 5-3-1: Arduino servo rotation and distance reading code

In the code, the "x" represents the PWM for the servo to turn. The total angle turn is 90 and is divided by 4. However, calibrating the servo motor, it was found that the angle input for turning 90° is by inserting 85 as PWM. So the PWM is from 85 to 0.

PWM for Servo turning (angle)	Reading
85(90°)	sensorValue[0] , sensorValue[4]
63.75(67.5°)	sensorValue[1] , sensorValue[5]
42.5(45°)	sensorValue[2] , sensorValue[6]
21.25(22.5°)	sensorValue[3] , sensorValue[7]
0(0°)	sensorValue[4] , sensorValue[8]

Table 5-3-1: Servo turning and ultrasonic reading

Ultrasonic sensors reading returns the time taken for the ultrasound to bounce back the echo. Thus, to translate it into distance, the formula “distance1 = (duration/2) / 29.1” is used. Once the reading is taken, it will be stored in an array named sensorValue.

The sensorValue will then be sent one by one to Bluetooth via a for-loop. The value will be concatenated to a string in the Android application. The code is as shown in figure 5-3-2.

```
Serial.print('#');
  for(int k=0; k<9; k++)
  {
    Serial.print(sensorValue[k]);
    Serial.print('+');

  }
 Serial.print('~'); //used as an end of transmission character - used in app for string length
 Serial.println();
 delay(10);

}
```

Figure 5-3-2: Sending data to Android with for loop

The first thing to “serial.print” is a “#” which is used to detect the starting character of the string. A “+” is used as the delimiter to adjacent values. The string received by the Android application is as shown as the figure below.

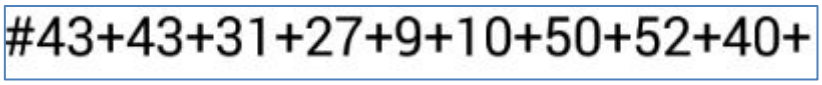

Figure 5-3-3: Android: String received

Since the value is transfer serially, the data needs to be appended to form a string. When there is incoming data from the Arduino, the data will be stored into a string variable, then append into a string builder. The reason of checking the beginning and ending of strings id to ensure that all data is fully received before proceeding to the calculation part.

```
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {                    //if message is what we want
            String readMessage = (String) msg.obj;                                // msg.arg1 = bytes from connect threa
            recDataString.append(readMessage);                                    //keep appending to string until ~
```

Figure 5-3-4: Android: receive data code + append

Once the Android application successfully receives the whole string, it will be broken down with a simple for loop by using "+" as the delimiter. The points will be written into the text view for user to check the distance. The for-loop for breaking up the string is as shown in figure 5-3-5.

```
if (recDataString.charAt(0) == '#')                    //if it starts with # we know it is what we are looking for
{

    for (int y=0 ; y<10 ; y++ ){
        for(int i = 1 ; i < recDataString.length() ; i++){
            if((recDataString.charAt(i) == '+')){
                int pivot = i;
                sensor[y] = Integer.parseInt(recDataString.substring(counterX, pivot));
                counterX = pivot+1;
                y++;
            }
        }
    }
```

Figure 5-3-5: Android: Break Down string

As the transfer data is a string, the data needs to be converted into integer format by using Integer.parseInt () before storing them into an integer array name sensor. The for loop detects "#" as the starting character of the string to be read and then checks "+" as a delimiter of each value. The string also ends with a "+" to show it is the last value.

Now that the 9 values needed to calculate the coordinate are acquired, we can now start to calculate the coordinate of each point by using trigonometry except the 0,4,9 value as sensor[0] represents x-axis minimum value, sensor[8] represents maximum value of x-axis and sensor[4] represent y-axis maximum value.

Since the value we get from the ultrasonic sensor is the distance of depth relative to the sensor, it needs to be transformed to the correct coordinates in Cartesian planes. This is especially crucial when the ultrasonic sensor orientation is not parallel to the car direction. The coordinate can be calculated with the formula below to obtain the correct coordinate.

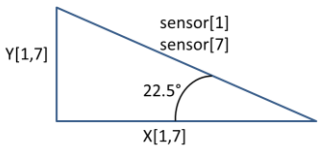

Figure 5-3-6: trigonometry diagram for sensor[1,7]

- Coordinate for (x1,y1)
    - x1 = ((COS (22.5))*(sensor[1]) *-1)
    - y1 = (SIN(22.5))*(sensor[1])

- Coordinate for (x7,y7)
    - x7 = (COS (22.5))*(sensor[7])
    - y7 = (SIN (22.5))*(sensor[7])
- Since x1 is on the left hand side of the chart, value is multiply by -1 to plot the graph correctly.

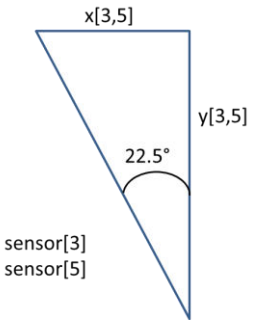

Figure 5-3-7: trigonometry diagram for sensor[3,5]

- Coordinate for (x3,y3)
    - x3 = ((SIN (22.5))*(sensor[3]) *-1)
    - y3 = (COS (22.5))*(sensor[3])
- Coordinate for (x5,y5)
    - x5 = (SIN (22.5))*(sensor[5])
    - y5 = (COS (22.5))*(sensor[5])
- Since x3 is on the left hand side of the chart, value is multiply by -1 to plot the graph correctly.

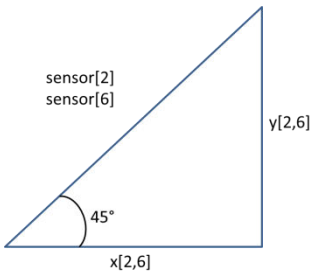

Figure 5-3-8: trigonometry diagram for sensor[2,6]

- Coordinate for (x2,y2)
    - x2 = ((COS(45))*(sensor[2]) *-1)
    - y2 = (SIN (45))*(sensor[2])
- Coordinate for (x6,y6)
    - x6 = (COS (45))*(sensor[6])
    - y6 = (SIN (45))*(sensor[6])
- Since x2 is on the left hand side of the chart, value is multiply by -1 to plot the graph correctly.

Chapter6: Final Product

6.1 Remote Control Car

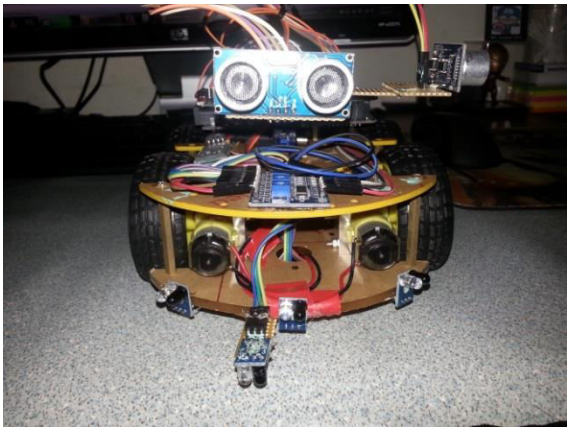

Figure 6-1-1: Front view of remote control car

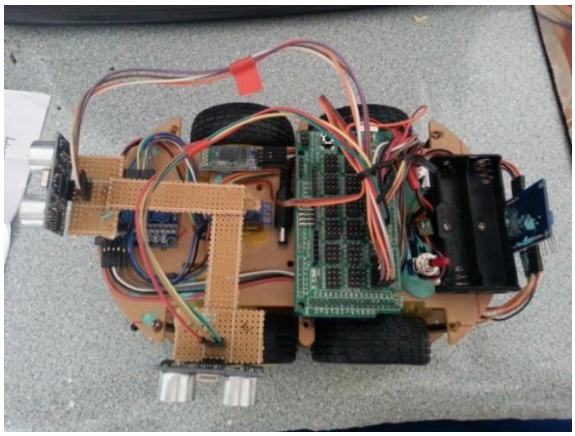

Figure 6-1-2: top view of remote control car

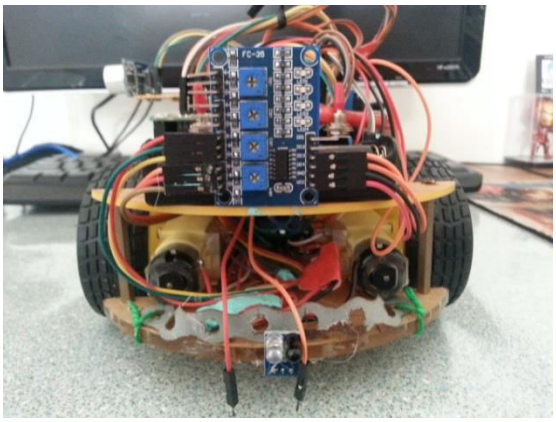

Figure 6-1-3: rare view of remote control car

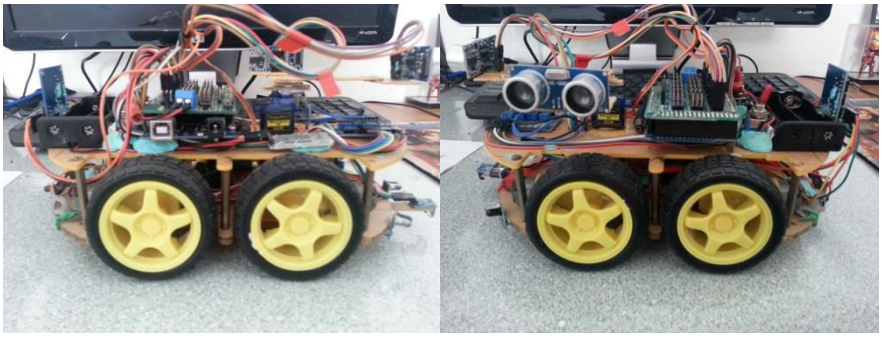

Figure 6-1-4: side view of remote control car

6.2 Android Application

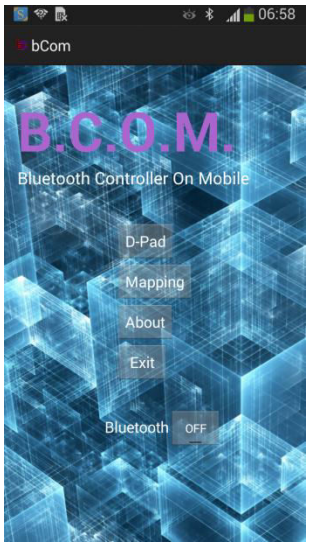

Figure 6-2-1: main menu

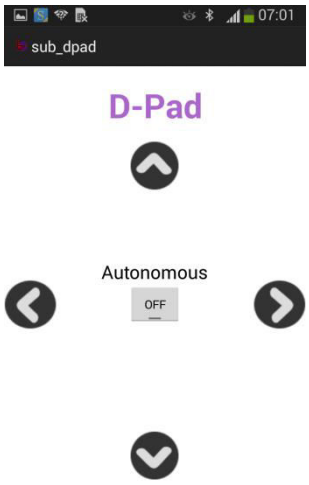

Figure 6-2-2: DPAD + autonomous

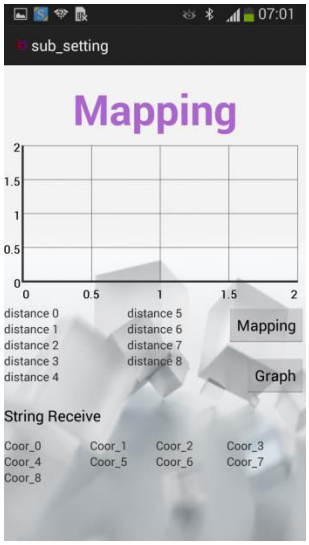

Figure 6-2-3: Mapping

Figure 6-2-3: About

Chapter7: Testing and Result

7.1 DPAD + IR

7.1.1 DPAD + IR + testing

To test its application, the car is placed in an area with obstacles. The user should be able to maneuver the car from the area without hitting any obstacle. The main point of this test is to ensure the car can turn or go straight or reverse as long as the user wants. At the same time, it is done to check whether the IR is working as the IR is used to avoid the car from hitting anything even if it is forced to hit.

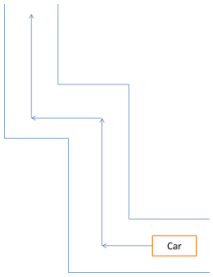

Figure 7-1-1-1: track for testing

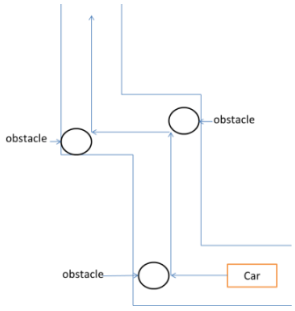

Figure 7-1-1-2: obstacle testing

The second DPAD test is to drive the car to the stairs to ensure the IR for detecting stairs work before allowing it to run in autonomous mode.

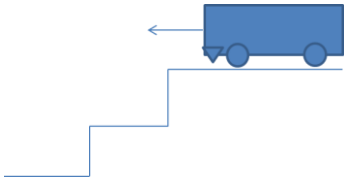

Figure 7-1-1-3: stairs testing

7.1.2 DPAD + IR + Result

The car is able to move more smoothly compared to during FYP1 as the code is modified to use on touch listener which enables finer grain of control. The car now also stop more rapidly due to the handbrake imitator which allows the car to stop before hitting onto anything especially when the user cannot control it properly.

The part to test near the stairs also yields positive result, as upon reaching the stairs, the IR that keeps detecting the floor will suddenly lost detection of the floor. This causes the car to stop instantly and will trigger the car to reverse and make a 180° turn to go away from the stairs.

No	Test case	Result and discussion
1.0	• Dpad up button is pressed and release instantly	• The car move forward for 200ms then stops
1.1	• Dpad up button is pressed and release instantly • (l4) / (l4 & l1) / (l4 & l2)	• The car move forward for 200ms then stops • car will turn right if there is something on the left
1.2	• Dpad up button is pressed and release instantly • (l3) / (l3 & l1) /(l3 & l2)	• The car move forward for 200ms then stops • car will turn left if there is something on the right
1.3	• Dpad up button is pressed and release instantly • (l1)	• The car move forward for 200ms then stops • car reverse for 800ms if there is something in front
2.0	• Dpad down button is pressed and release instantly	• The car move backward for 400ms then stops
2.1	• Dpad down button is pressed and release instantly • (l4) / (l4 & l1) / (l4 & l2)	• The car move backward for 400ms then stops • car will turn right if there is something on the left

2.2	• Dpad down button is pressed and release instantly • (l3) / (l3 & l1) /(l3 & l2)	• The car move backward for 400ms then stops • car will turn left if there is something on the right
2.3	• Dpad down button is pressed and release instantly • (l2)	• The car move backward for 200ms then stops • car move forward for 400ms if there is something at the back
3.0	• Dpad left button is pressed and release instantly	• the car will turn left for 100ms
3.1	• Dpad left button is pressed and release instantly • (l4) / (l4 & l1) / (l4 & l2)	• The car turn left for 100ms then stops • car will turn right if there is something on the left
4.0	• Dpad right button is pressed and release instantly	• The car turn right for 100ms then stops
4.1	• Dpad right button is pressed and release instantly • (l3) / (l3 & l1) / (l3 & l2)	• The car turn right for 100ms then stops • car will turn left if there is something on the right
5.0	• Dpad up button is pressed for some time	• The car move forward until the button is release.
5.1	• Dpad up button is pressed for some time • (l4) / (l4 & l1) / (l4 & l2)	• The car move forward until the button is release. • car will turn right if there is something on the left
5.2	• Dpad up button is pressed for some time • (l3) / (l3 & l1) /(l3 & l2)	• The car move forward until the button is release. • car will turn left if there is something on the right

5.3	• Dpad up button is pressed for some time • (l1)	• The car move forward until the button is release. • car reverse for 800ms if there is something in front
6.0	• Dpad down button is pressed for some time	• The car move backward until the button is release.
6.1	• Dpad down button is pressed for some time • (l4) / (l4 & l1) / (l4 & l2)	• The car move backward until the button is release. • car will turn right if there is something on the left
6.2	• Dpad down button is pressed for some time • (l3) / (l3 & l1) /(l3 & l2)	• The car move backward until the button is release • car will turn left if there is something on the right
6.3	• Dpad down button is pressed for some time • (l2)	• The car move backward until the button is release • car move forward for 400ms if there is something at the back
7.0	• Dpad left button is pressed for some time	• The car turn left until the button is release
7.1	• Dpad left button is pressed for some time • (l4) / (l4 & l1) / (l4 & l2)	• The car turn left until the button is release • car will turn right if there is something on the left
8.0	• Dpad right button is pressed for some time	• The car turn right until the button is release
7.1	• Dpad right button is pressed for some time	• The car turn right until the button is release

	• (l3) / (l3 & l1) / (l3 & l2)	• car will turn left if there is something on the right
9.0	• Dpad autonomous button is pressed every odd time	• The car move forward until button is pressed again
10.0	• Dpad autonomous button is pressed every even time	• stops the car immediately
11.0	• l_btm did not detect anything	• reverse • make a left 180° turn

Table 7-1-2: test case for dpad

7.2 Autonomous + IR + Ultrasonic

7.2.1 Autonomous + IR + Ultrasonic testing

To test the autonomous function, a simple track is set to see if the car will avoid the obstacles. By default, the autonomous mode will set the car moving forward. What help the car to maneuver are the IR sensors and the ultrasonic sensors. As mentioned, there are 5 IR installed on the car including the one which prevent the car falling from higher location. The rest of them are tested when the car moves near a wall.

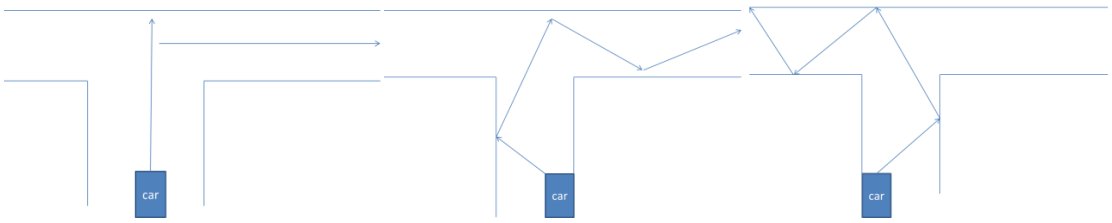

Figure 7-2-1-1: tracks for testing

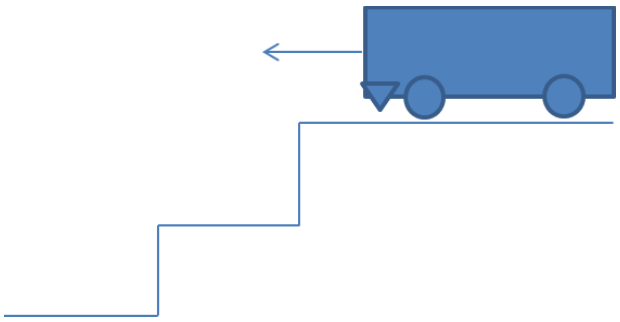

Figure 7-2-1-2: test on stairs

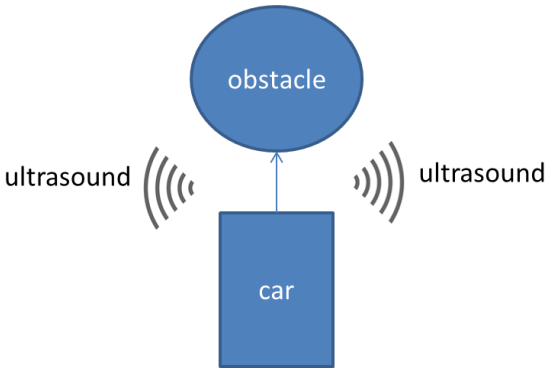

Figure 7-2-1-3: face obstacle testing

7.2.2 Autonomous + IR + Ultrasonic Result

The result for the autonomous car is a success as the car is able to escape from a small track with the help of IR. When the car comes to a T junction, the car will reverse and use ultrasonic sensors to find a way out of the place.

No	Test Case	Result & discussion
1	l1	• trigger ultrasonic sensors make decision to turn left or right
2	l2	• move forward
3	l4	• turn right
4	l3	• turn left
5	l3 l1	• turn right
6	l4 l1	• turn left
7	l3 l2	• turn right
8	l4 l2	• turn left
9	l4 l3 l1	• reverse
10	l_btm	• reverse • make a left 180° turn
11	l1 left and right distance < 50	• reverse • make a left 180° turn

Table 7-2-2: test case for autonomous

7.3 Mapping

7.3.1 Mapping Testing

To test the mapping algorithm, the car is place in several types of simple location and since it is a prototype, the first test is a rectangular area. The expected result is shown as figure below.

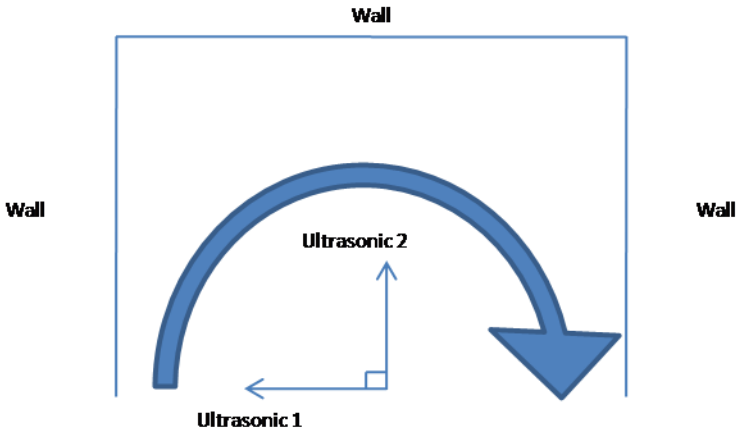

Figure 7-3-1-1: rectangular angle scan

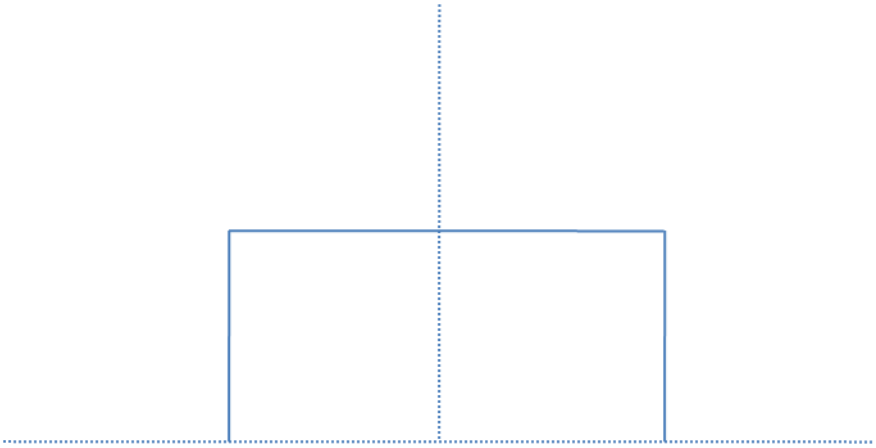

Figure 7-3-1-2: rectangular angle expected result

The second area to test is a triangular angle. The expected result is as shown as figure below.

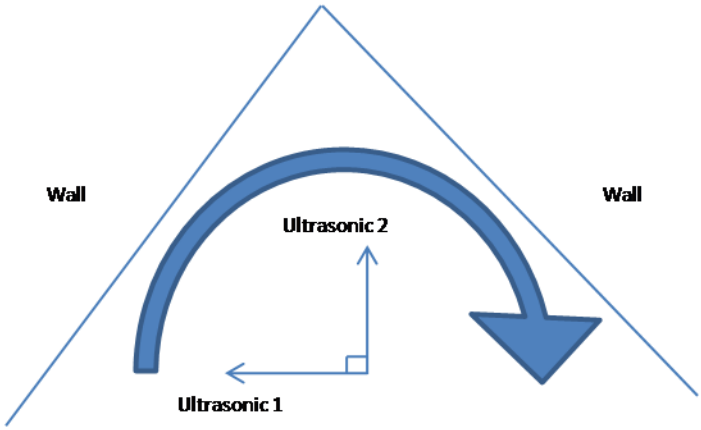

Figure 7-3-1-3: triangular angle scan

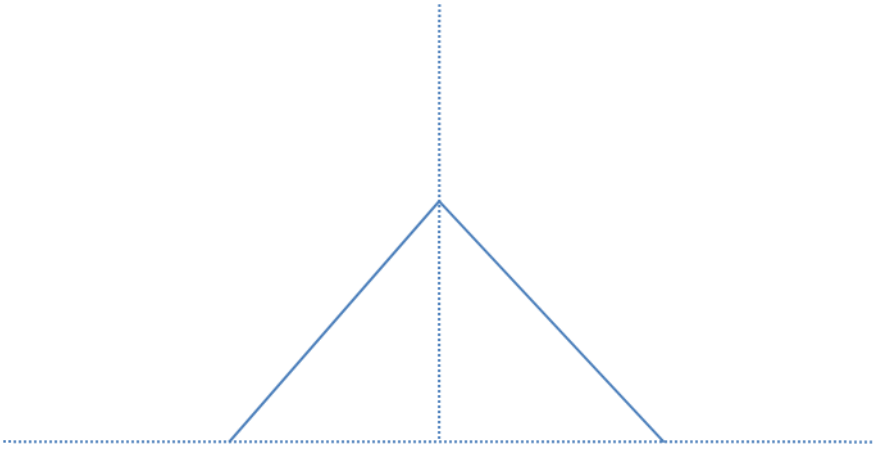

Figure 7-3-1-4: triangular expected result

The third area to test is a rectangular angle with obstacle. The expected result is as shown as figure below.

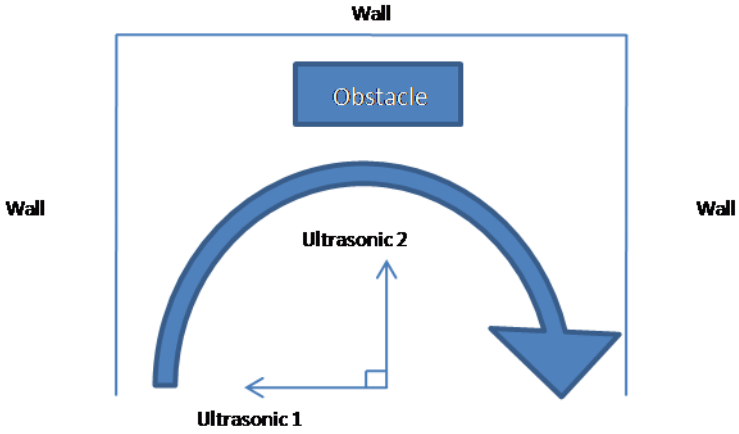

Figure 7-3-1-5: rectangular angle with obstacle scan

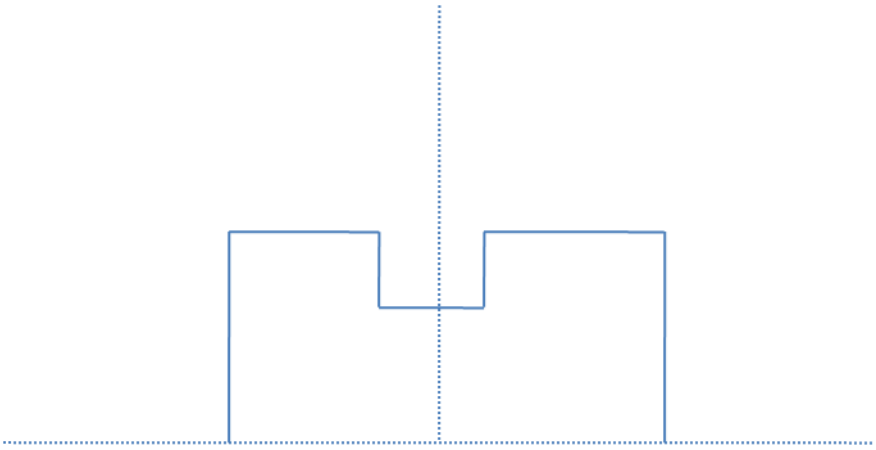

Figure 7-3-1-6: rectangular angle with obstacle scan expected result

7.3.2 Mapping Result

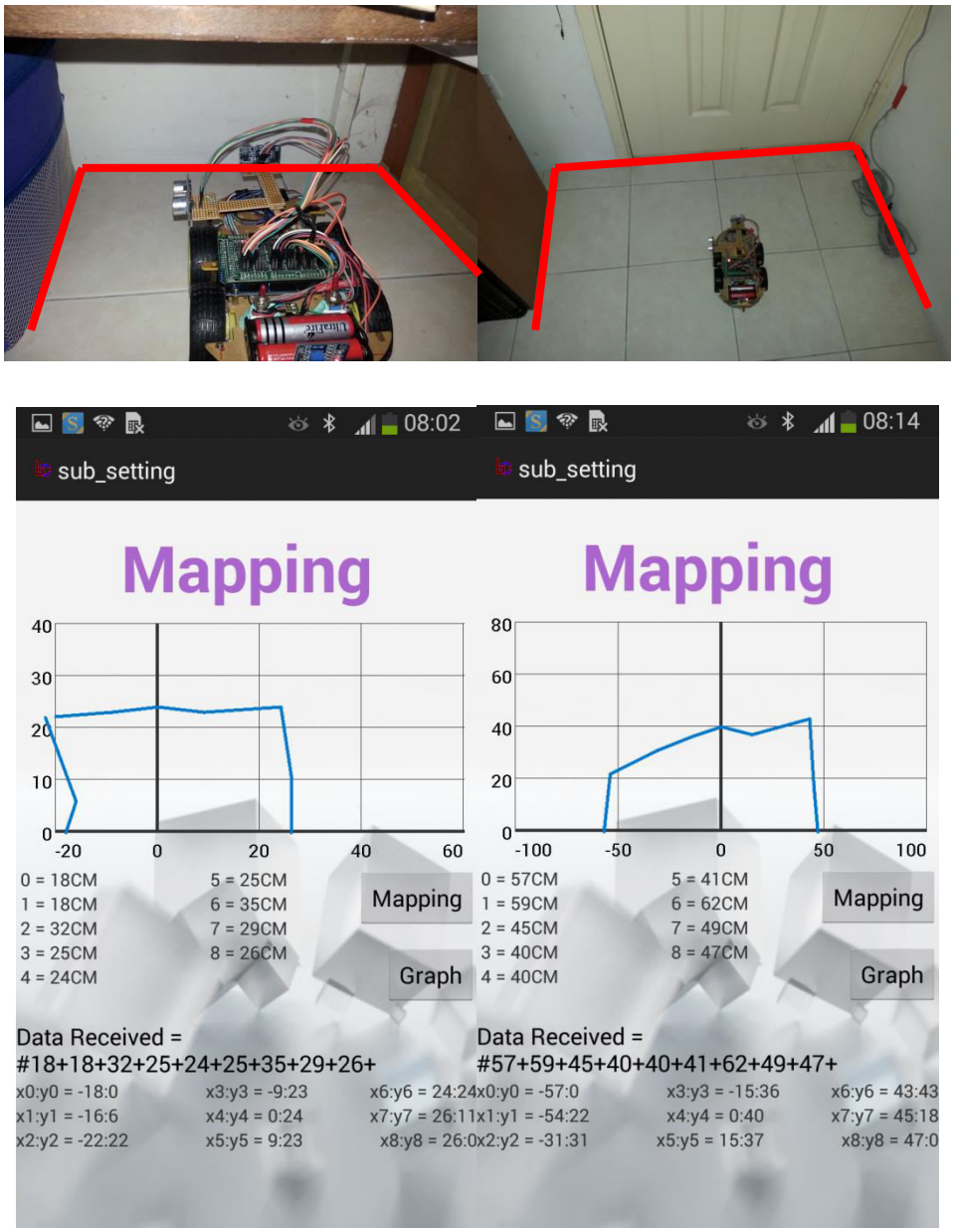

Figure 7-3-2-1: rectangular angle scan result

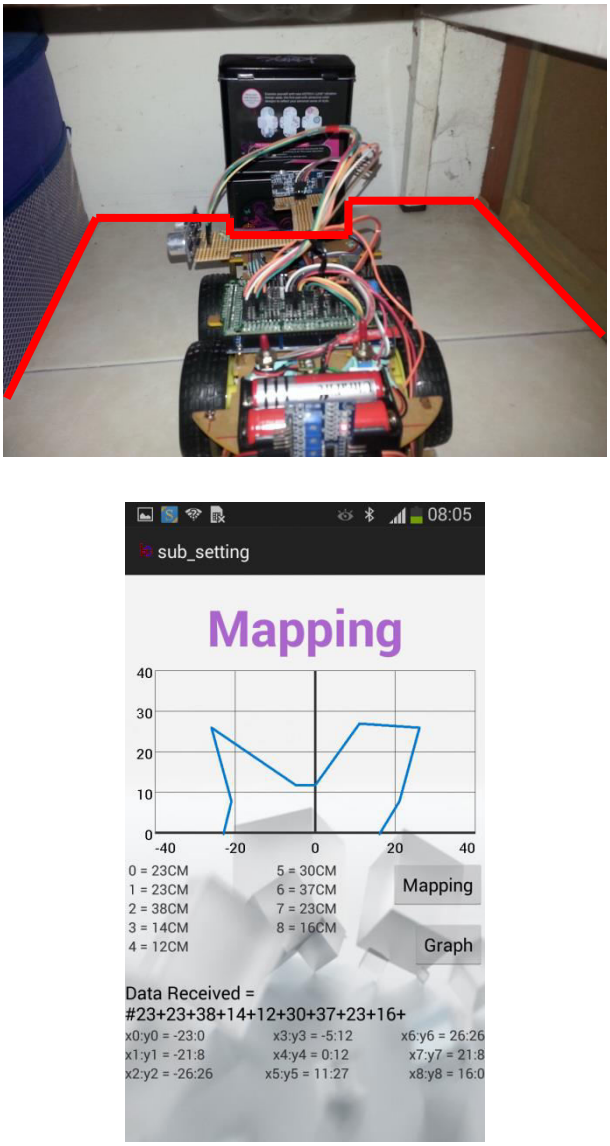

Figure 7-3-2-2: rectangular angle with obstacle scan result

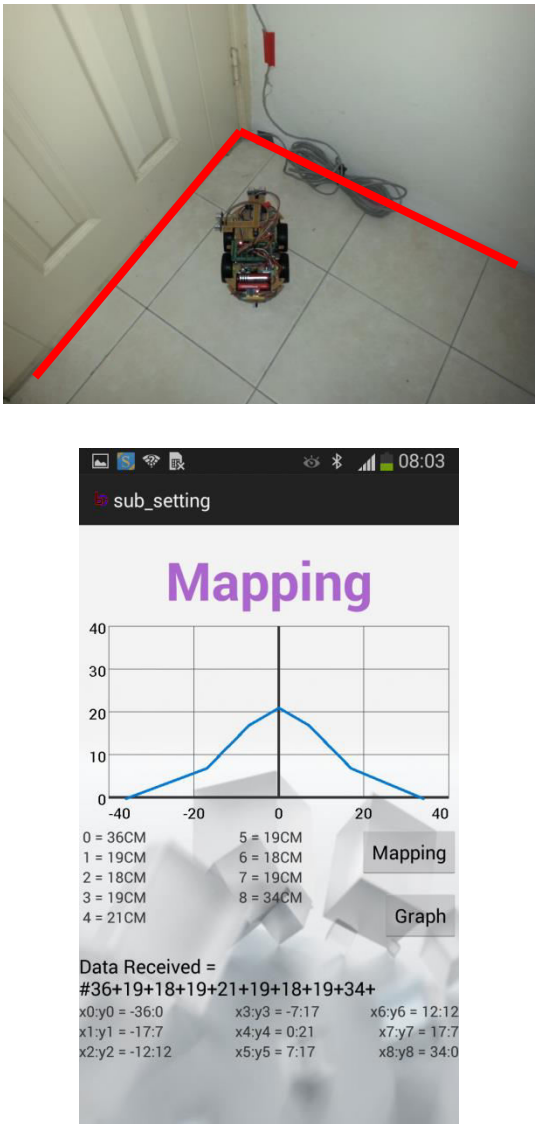

Figure 7-3-2-3: triangular angle scan result

Chapter8: Conclusion and Discussion

Most objectives have been successfully achieved except for the ultrasonic mapping which still needs further refinement as test result shows it is still far from perfect. However, it did manage to draw out the shape needed. For more accurate result, the numbers of reading can be increase. This will have more scan points and higher resolution of shape can be detected.

Since this is a prototype, the car used is not perfect. In order to travel in bumpy roads such as cave, the car would need to be modified for to suit such purpose. The current car works fine in normal circumstances. In cave, it might not be able to move as well as the car is not equipped with suspension to travel bumpy road. The power source for this car can also be changed to sustain longer usage.

As for the DPAD, the code has been refined and it is controlled by smartphone via Bluetooth. The reason of using Bluetooth here is that all smartphone is equipped with Bluetooth module and Bluetooth module may not be interrupted so easily as the Bluetooth is set to only bond with the smartphone itself. However, for future improvement, it can be changed to Wi-Fi connection. Autonomous mode code has also been refined to handle all condition as listed in the test case section.

The car can be used to assist in hazardous situation. In this kind of situation, extra function such as robotic hand might be useful. To detect gas leakage is one of the examples. User can use the robotic arm to fix it instead of having human to go to hazardous location.

Chapter9: Reference

1. Modular Circuits (no date), H-Bridges-the basic [online].
   Available from: <http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/>
   (14th February 2016)
2. Richard W.Wall, Jerry Bennett, Greg Eis, Kevin Lichy, Elizabeth Owings (no date), 'Creating a Low Cost Autonomous Vehicle', University of Idaho,
   Available from:
   <http://isl.ecst.csuchico.edu/DOCS/Logs/Michael/Files/web_link_files/low_cost.pdf>
   (14th February 2016)
3. Christian Bodenstein, Michael tremer, Jonathan Overhoff, Rolf P.Wurtz (2015) 'A smartphone-controlled Autonomous Robot', 12th international Conference on Fuzzy System and Knowledge, Discovery(FSKD'15),
   Available from: <http://www.ini.rub.de/uploads/document/attachment/352/android-robot.pdf >
   (14th February 2016)
4. R.A.Ramlee, M.H.Leong, R.S.S.Singh, M.M.Ismail, M.A.Othman, H.A.Sulaiman, M.H.Misran and M.A.Meor Said (2013) 'Bluetooth Remote Home Automation System Using Android Application', The international Journal Of Engineering And Science, vol 2, 11 January, p.1.

   Available from: <http://eprints2.utem.edu.my/11005/1/X02101490153.pdf >

   (8 July 2015)

5. Antoine Monmarche (2011) 'RC Car Controlled by WiFi with an Android Smartphone', 7 April, p.4.

   Available from: <http://todoelectronica.com/documentos/carcontroller.pdf >

   (8 July 2015)

6. Nicolas Oros, Jeffrey L.Krichmar (2013) 'SmartPhone Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers', Center for Embedded Computer Systems University of Califonia, Irvine, 26th November, p.2.

Available from: <http://cecs.uci.edu/files/2013/12/TR-13-16.pdf >

(8 July 2015)

7. Jeff Tyson (no date) How Radio Controlled Toys Work. How Stuff Works [online]. Available at http://electronics.howstuffworks.com/rc-toy.htm .

   (8th July 2015)

8. Panther Products (no date). What is Bluetooth. Panther Products[online]. Available at http://www.pantherproducts.co.uk/Articles/Communication/Bluetooth.shtml. (9th July 2015, 10:20pm).

9. Tolik777 (no date). Simple RC car for beginners (Android control over Bluetooth). Instructables [online].

   Available at: <http://www.instructables.com/id/Simple-RC-car-for-beginners-Android-control-over-/ >

   (6th July 2015, 12:01am)

10. Ritika Pahuja, Narender Kumar (2014) 'Android Mobile Phone Controlled Bluetooth RobotUsing 8051 Microcontroller'

    Available at: <http://www.ijser.in/archives/v2i7/SjIwMTMzMjQ=.pdfLast>

    (20th July 2015)

11. Chomtip Pornpanomchai, Member, IACSIT, and Phichate Sukklay (2011) 'Operating Radio-Controlled Cars by a Computer'

    Available at: <http://www.ijetch.org/papers/227-JT322.pdf >

    (20th July 2015)

12. Junghoon Ha aka Eric Ha, Stevanes Hermawan, Willie Pramono (no date) 'Parallel Parking R/C Car'

    Available at: <http://eecs.richardnelson.com/2006-07/CpE_Projects/RC%20Car%20Parallel%20Parking/Parallel_Parking_RC_Car.pdf>

    (20th July 2015)

fyp2

ORIGINALITY REPORT

9%	6%	2%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Teknikal Malaysia Melaka Student Paper	1%
2	grietinfo.in Internet Source	1%
3	Submitted to Heriot-Watt University Student Paper	1%
4	Submitted to Visvesvaraya Technological University Student Paper	<1%
5	Submitted to University of Northumbria at Newcastle Student Paper	<1%
6	www.modularcircuits.com Internet Source	<1%
7	Submitted to Higher Education Commission Pakistan Student Paper	<1%
8	Submitted to Ajman University of Science and Technology Student Paper	<1%

9	sunstorm.com.au Internet Source	<1%
10	Submitted to Multimedia University Student Paper	<1%
11	Submitted to Oklahoma State University Student Paper	<1%
12	www.elecfreaks.com Internet Source	<1%
13	widuri.raharja.info Internet Source	<1%
14	Submitted to University of Sunderland Student Paper	<1%
15	Xu, Zhaohong, Tiansheng Lu, and Xuyang Wang. "Inertia Matching Manipulability and Load Matching Optimization for Humanoid Jumping Robot", International Journal of Advanced Robotic Systems, 2008. Publication	<1%
16	Submitted to Institute of Technology Blanchardstown Student Paper	<1%
17	www.docstoc.com Internet Source	<1%
18	www.iit.tsukuba.ac.jp Internet Source	<1%
19	Submitted to University of South Alabama	

	Student Paper	<1%
20	Submitted to University of Houston System Student Paper	<1%
21	Submitted to Universiti Tenaga Nasional Student Paper	<1%
22	Submitted to University of Florida Student Paper	<1%
23	Submitted to University of Nottingham Student Paper	<1%
24	www.robots-and-androids.com Internet Source	<1%
25	www.robotshop.com Internet Source	<1%
26	Submitted to Nicholls State University Student Paper	<1%
27	www.famosastudio.com Internet Source	<1%
28	edoc.ub.uni-muenchen.de Internet Source	<1%
29	www.oddwires.com Internet Source	<1%
30	Lattice Theory Special Topics and Applications, 2014. Publication	<1%

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	< 2 WORDS
EXCLUDE BIBLIOGRAPHY	ON		