

AUTONOMOUS RC CAR CONTROL USING COMPUTER VISION

BY

CHONG WEN YANG

A REPORT

SUBMITTED 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)**

MAY 2017



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (PERAK CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	DOCUMENT
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as page 2)

(Signature of Student)

Date:

(Signature of Supervisor)

Date:

REPORT STATUS DECLARATION FORM

Title: _____

Academic Session: _____

I _____
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

Supervisor's name

Date: _____

Date: _____

AUTONOMOUS RC CAR CONTROL USING COMPUTER VISION

BY

CHONG WEN YANG

A REPORT

SUBMITTED 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)

MAY 2017

DECLARATION OF ORIGINALITY

I declare that this report entitled “**AUTONOMOUS RC CAR CONTROL USING COMPUTER VISION**” 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 : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Mr Teoh Shen Khang who has given who gives me a lot patient, support and guidance. He is always willing to teach me knowledge and answer any question about the project. Since this project require skills in embedded system, I would like to take this opportunity to thanks Mr. Lee Wai Kong and my supervisor, Mr Teoh for teaching me embedded system in previous semester.

I would like to thank my lecturer Mr Lee Wai Kong who borrow me the car base car track and other hardware components and Mr Leong Chun Farn who guide me on the lane tracking. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

More and more different types of autonomous car for different purposes were created. The most challenging part of developing an autonomous car is to achieve low cost, low power and consists many features. This project is to develop a prototype of an autonomous RC car controlled by Android smartphone. The RC car is able to navigate itself through the track and overtake any obstacles by computer vision technique. The RC car have 4 wheels which front wheels helps the RC car to turn in right or left direction and the back wheels helps the RC to move forward and backward. The back wheels DC motor controlled by a motor driver. A webcam was attached on the RC car for computer vision purpose. The ultrasonic sensor is attached at the front of the RC car to detect the obstacles when passing through the obstacles. A 2 lane track is built to test the RC car and some obstacles are placed on the track. The RC car will navigate itself and overtake all the obstacles on the track. An Android application was developed to control the RC car in manual mode or autonomous mode. If anything goes wrong in autonomous mode, user able to switch to manual mode to control the RC car. In manual mode, user able see the live view from the RC car webcam and change she speed of the RC car while in autonomous mode, the speed of the RC car are fixed. The smartphone communicate with single board computer through Wi-Fi. The single board computer hosted a TCP server. When a button is pressed in the Android application, the command data will send to the TCP server and the single board computer will send data signal to the connected components to control the movement of the RC car.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	i
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	viii
LIST OF ABBREVIATIONS	ix
Chapter 1: Introduction	1
1.1 Problem Statement	1
1.2 Project Scope	2
1.3 Background Information	2
1.4 Project Objective	3
1.5 What Have Been Achieve	3
1.6 Impact, Significance and Contribution	4
Chapter 2: Literature Review	5
Chapter 3: System Design	6
Chapter 4: Methodology	9
Chapter 5: Hardware Development	11
5.1 Chassis	11
5.2 Raspberry Pi 3 Model B	12
5.3 L298N Dual H-Bridge Motor Driver	14
5.4 Futaba S3010 Standard High-Torque BB Servo Motor	22
5.5 Ultrasonic Sensor (HC-SR04)	26
5.6 Logitech USB Camera (HD Webcam C270)	31
5.7 PlayStation 4 Controller	34
Chapter 6: Software Development	38
6.1 Raspberry Pi 3	38

6.1.1 Set up Raspberry Pi 3	38
6.1.2 Create hotspot using Raspberry Pi 3	41
6.1.3 Hotspot Script	44
6.2 Python (IDLE)	45
6.2.1 TCP Socket Server on Raspberry Pi 3 Python 2	47
6.2.2 Lane Tracking algorithms	48
6.3 Android Studio	54
6.3.1 Android Application	56
Chapter 7: System Integration	63
7.1 PS4 Controller + Servo Motor + DC Motor	63
7.2 Android Application + Manual Mode	65
7.3 Ultrasonic Sensor + Autonomous Mode	66
7.4 Overall System	66
Chapter 8: Testing and Result	69
8.1 Manual Mode	69
8.1.1 PS4 Controller with Manual Mode	69
8.1.2 Android Application with Manual Mode	70
8.2 Autonomous Mode	72
Chapter 9: Conclusion	78
Bibliography	79

LIST OF FIGURES

List of Figure	Page
Figure 3.1 System Block Diagram	6
Figure 3.2 Final Product View	7
Figure 4.1 General Steps for Prototyping Model	9
Figure 5.1.1 Top View of Chassis	11
Figure 5.2.1 Raspberry Pi 3 Model B Technical Specification	13
Figure 5.2.2 Raspberry Pi 3 Model B Pin Layout	13
Figure 5.3.1 L298N Motor Driver	14
Figure 5.3.2 H-bridge Principle	15
Figure 5.3.3 H-bridge (Clockwise direction)	15
Figure 5.3.4 H-bridge (Anti-clockwise direction)	16
Figure 5.3.5 H-bridge (Short circuit)	17
Figure 5.3.6 Connection of Raspberry Pi 3, L298N and DC motor	17
Figure 5.3.7 RC Car Move Forward Code	19
Figure 5.3.8 RC Car Move Forward	20
Figure 5.3.9 RC Car Move Backward	20
Figure 5.4.1 Futaba S3010	22
Figure 5.4.2 Connection of Raspberry Pi 3, L298N and Servo Motor	23
Figure 5.4.3 Servo Motor Test Code	24
Figure 5.4.4 RC Car Turn left	25
Figure 5.4.5 RC Car Turn Right	25
Figure 5.5.1 HC-SR04	26
Figure 5.5.2 HC-SR04 Working Principle	27
Figure 5.5.3 Connection of Raspberry Pi 3 and Ultrasonic Sensor	27
Figure 5.5.4 Voltage Divider	28
Figure 5.5.5 Ultrasonic Sensor Test Code	29
Figure 5.5.6 Ultrasonic Sensor Test Result	30
Figure 5.6.1 Logitech Webcam C270	31
Figure 5.6.2 Live View from Webcam through Browser	33
Figure 5.7.1 PS4 Controller	34

Figure 5.7.2 PS4 Controller Description	34
Figure 5.7.3 PS4 Controller Test Code	36
Figure 6.1.1 Installation of Raspbian on Raspberry Pi 3	39
Figure 6.1.2 Enable VNC	40
Figure 6.1.3 Connect to Raspberry Pi 3 using VNC Viewer	41
Figure 6.1.4 Hotspot Script	44
Figure 6.2.1 Python 2 Interface	46
Figure 6.2.2 TCP Socket Server Test Code	47
Figure 6.2.3 Flowchart of Lane Tracking	49
Figure 6.2.4 Capture Frame using OpenCV Test Code	50
Figure 6.2.5 Grayscale Image	51
Figure 6.2.6 Blurred Image	51
Figure 6.2.7 Edge Image	52
Figure 6.2.8 Masked Image	53
Figure 6.2.9 Result of Lane Tracking Algorithm	54
Figure 6.3.1 Configuration of Android Virtual Device	55
Figure 6.3.2 Android Studio Interface	55
Figure 6.3.3 Code of Autonomous Class	56
Figure 6.3.3 Code of Autonomous Layout	57
Figure 6.3.4 Code of SendMessage Class	58
Figure 6.3.5 Code of AndroidManifest.xml	59
Figure 6.3.6 Flowchart of Android Application	60
Figure 6.3.7 Android Application Menu Interface	61
Figure 6.3.8 Android Application Manual Interface	61
Figure 6.3.9 Android Application Autonomous Interface	62
Figure 6.3.10 Android Application PS4 Controller Interface	62
Figure 7.1.1 Final product Block Diagram	63
Figure 7.1.2 PS4 Controller Used Axis	63
Figure 7.1.3 Code for RC Car Move Forward and Backward	64
Figure 7.4.1 Flowchart of System Part 1	67
Figure 7.4.2 Flowchart of System Part 2	68

Figure 8.1.1 Android Application Manual Mode	70
Figure 8.2.1 Test Case 1	72
Figure 8.2.2 Test Case 2	73
Figure 8.2.3 Test Case 3	74
Figure 8.2.4 Test case 4	75
Figure 8.2.5 Test Case 5	76
Figure 8.2.6 Test Case 6	77

LIST OF TABLES

List of Table	Page
Table 5.2.1 Raspberry Pi 3 model B basic specification	12
Table 5.3.1 L298N Motor Driver Specification	14
Table 5.3.2 L298N Pin Assign	18
Table 5.3.3 DC Motor Test Case	21
Table 5.4.1 Futaba S3010 Specification	22
Table 5.4.2 Servo Motor Test Case	25
Table 5.5.1 HC-SP04 Specification	26
Table 5.6.1 Logitech Webcam C270 Specification	31
Table 5.7.1 PS4 Controller Specification	35
Table 7.2.1 Command Table	65
Table 8.1.1 PS4 Controller Manual mode Test Case	69
Table 8.1.2 Android Application Manual mode Test Case	71

LIST OF ABBREVIATIONS

USB	Universal Serial Bus
RC	Remote Control
LED	Light-emitting Diode
Wi-Fi	Wireless Fidelity
IR	Infrared Sensor
DCHP	Dynamic Host Configuration Protocol
IDE	Integrated Development Environment
DC	Direct Current
GPIO	General-purpose input/output
CSI	Camera Serial Interface
DSI	Display Serial Interface
SoC	System on Chip
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ARM	Advanced RISC Machines
MIPI	Mobile Industry Processor Interface
LCD	Liquid Crystal Display
TTL	Transistor Transistor Logic
MP	Mega Pixel
FPS	Frame Per Second
APK	Android Application Package
TCP	Transmission Control Protocol
FYP	Final Year Project
PWM	Pulse Width Modulation
RPi 3	Raspberry Pi 3
PS4	Play Station 4
JPG	Joint Photographic Experts Group
MJPG	Motion Joint Photographic Experts Group
HDMI	High-Definition Multimedia Interface
VGA	Video Graphics Array

SD	Secure Digital
GB	Gigabyte
VNC	Virtual Network Computing
PC	Personal computer
IP	Internet Protocol
WLAN	Wireless local area network
SSID	Service set identifier

Chapter 1: Introduction

1.1 Problem Statement

Nowadays, most of the RC car comes with their own remote control which binds the limitation to improve the controller. If the controller is changed to Android smartphone, an Android application can be designed in the way user likes. Other than that, Android smartphone nowadays supports a lot of feature. So, the Android application can include many feature rather than just control the RC car.

Camera is important for this project. The camera used to detect the line of the track. One of the option is to sue night vision camera which can work in the dark, but it is too expensive. So, it does not meet the low cost objective. A standard USB webcam can use in this project. In order to work in the dark, an USB webcam with build in LED can be used in this project or install the RC car with light to shine up the track but the result will not good as night vision camera.

There are few ways to communicate between RC car and controller. One of the most basic method is using radio wave which is 1 to 1. Another method is by using IR connection. But the communication range is too short for the controller to communicate with the RC car. Other than that, Wi-Fi is a better option. Wi-Fi perform more stable and larger communication range than radio wave and IR connection. Besides that, Wi-Fi connection able to stream the live view to the smartphone. Hotspot or router are needed for Wi-Fi connection. Since Raspberry Pi 3 is used in this project, Wi-Fi connection is chosen as the solution.

1.2 Project Scope

The project scope of this project is to deliver a prototype for the concept which the RC car able to navigate itself through the track with the help of computer vision and ultrasonic sensor. An Android application is created to let user control the RC using manual or autonomous control mode through Android smartphone. User able to watch the live view from the webcam that attached at the RC car in manual control mode.

A two lane track is build and one of the lane placed some obstacle that act as car. When the RC car detected the obstacle, it will move to another lane to overtake the obstacle and back to the original lane.

1.3 Background Information

An autonomous navigation system able to complete the task without the help of navigator. As years passed by, more and more different types of autonomous car for different purposes were created. One of the autonomous car nowadays is Google's self-driving vehicles. The vehicles able to detect most of the objects on the road such as animals and vehicles. Besides that, the vehicles can detect the road work in all direction from far away.

Autonomous vehicle are used many different area such as house, other planet for discovery and exploration, same narrow space like cave or mine or a building with which is on fire. . With the help of autonomous vehicle, many task had been achieved without risking our life. Each of the autonomous vehicle have different characteristics such as the ways to control the vehicle, the movement of the vehicle and the task to achieve.

With the help of computer vision, autonomous navigation system became perfect. Computer vision used the intelligent algorithms to process the digital data from videos and images. Computer vision has been started in several decades ago in academic research. Nowadays, a low cost, powerful and energy-efficient processors can be found easily which improving the use of computer vision in embedded systems.

1.4 Project Objective

This project aims to produce an android remote control car with autonomous using computer vision. The sub-objective are as below:

- To produce low cost autonomous RC car.
- To produce low power consumption autonomous RC car.
- The RC car is able to navigate itself through the track and overtake any obstacles by computer vision technique.
- Create an Android application that have both autonomous and manual control mode.

1.5 What Have Been Achieve

Below are the highlights of what have been achieved in this project so far.

- A prototype RC car was build.
- Ultrasonic sensor able to measure the distance between the RC car and obstacles.
- Able to control the servo motor to turn the direction of RC car front wheels to move left and right.
- Able to control the DC motor using L298N motor driver to move the RC car forward and backward.
- Android application is created to control the RC car.
- Android application able to communicate with Raspberry Pi 3 though Wi-Fi signal.
- Able to stream the live view using webcam though Android smartphone.
- PS4 controller able to communicate with Raspberry Pi 3 though Bluetooth and control the RC car in manual mode.
- An algorithm for lane tracking was created by computer vision technique with OpenCV library.

1.6 Impact, Significance and Contribution

By involving in building this prototype, we can understand and visualize better on how Raspberry Pi 3 work. With the help of sensors and webcam, the RC car will be able to navigate itself through the track by using computer vision and overtake obstacles at the track. This can enhances the knowledge of hardware as well as software of students, by knowing how to establish connection between Raspberry Pi 3, sensors, webcam and other hardware component. OpenCV library used to develop line detection for the track and obstacle detection. In order to communicate between Android smartphone and Raspberry Pi 3, Raspberry Pi 3 will set up as hotspot with DHCP server. When Android smartphone connect to Raspberry Pi 3, the smartphone will assigned an IP address from the DHCP server.

Programming languages that used in this project are JAVA and Python. Python software (IDE) used as the programming tool in Raspberry Pi 3 and Android Studio software (IDE) used to develop Android application by using JAVA programming language. Both of these software are easy to learn. There are many tutorials and guidelines were uploaded in the internet.

Chapter 2: Literature Review

Parth Verma did a research with the title “The Google Autonomous car”. The Google autonomous car was invented by Sebastian Thrun which was the co-inventor of the Street view mapping service. The strength of this research is that reduces human error while driving and result in decreasing possibility of accident occurring. Better fuel efficiency is achieved when more autonomous car on the road which change the driving habits. Besides that, safety features per-say was included in the autonomous car. Destination can be reached in shorter time which the car able to identify the shortest route. The weakness of this research are when the car is traveling at high speed which over 100mph, the car is hardly to differentiate the objects on the roads.

A research was did by Michal Ruzicka and Petr Masek with the title “Real Time Visual Marker Detector and Tracker Based on Computer Vision for Semi-autonomous Convoy Purpose”. This research is based control the semi-autonomous convoy by designing the computer vision method. The strength of this research is low power consumption and low cost product. The weakness of this research are using 320 x 240 low resolution of captured frames which will cause the result not accurate. Higher resolution cannot be used because of the low cost power device which does not do well in real time processing due to low frame rate. But this problem can be solve by OpenCL optimization.

Another research was done with the title “Autonomous Three-Wheeled Robot with Computer Vision System”. The research is based on three wheel robot with computer vision. The strength of this research is the product have many feature such as GPS, accelerometer and compass which can use for data collections purpose. Higher accuracy result is achieved with higher resolution and frequency. The weakness of this research is having a bad robot control method which can change to Wi-Fi control that can support further control range. Other than that, the robot cannot operate at dark surroundings. The robot may have problem with the movement of the robot which cannot avoid the obstacles with accurate turning angle of the robot.

Chapter 3: System Design

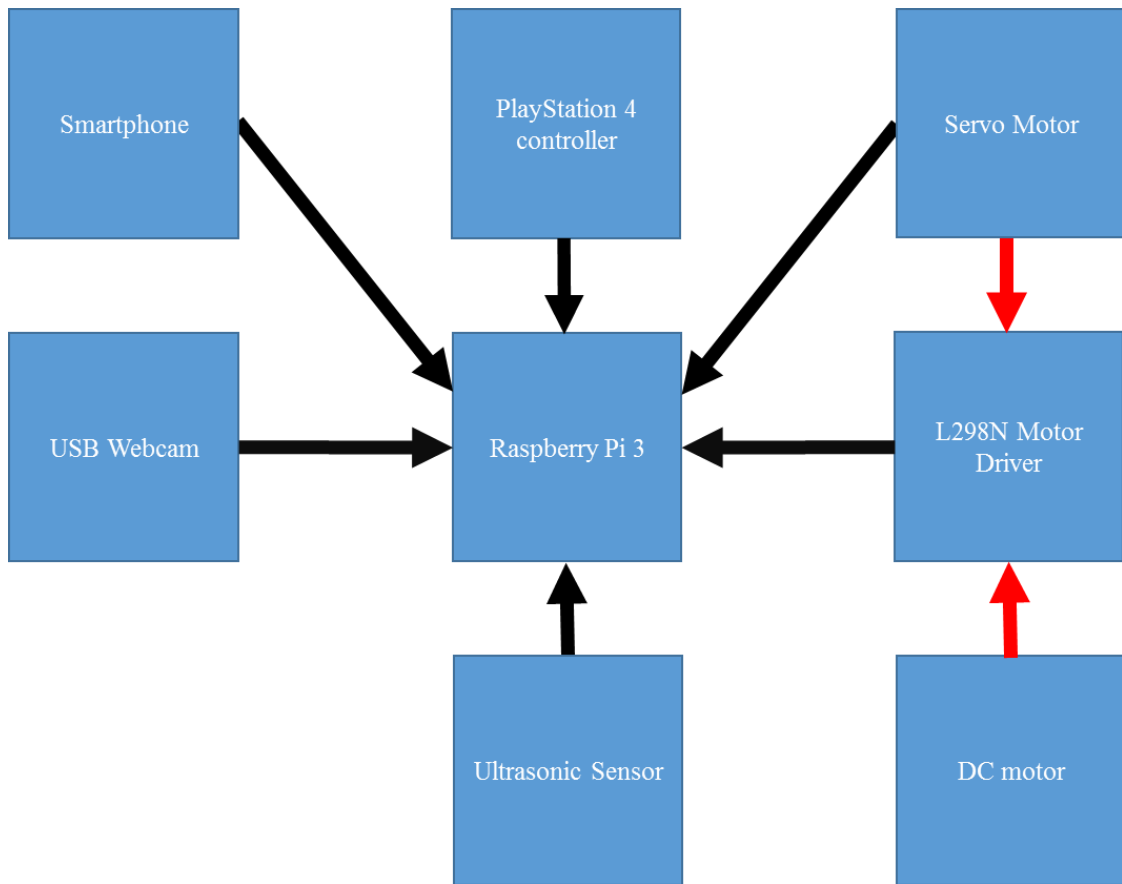


Figure 3.1 System Block Diagram

Figure 3.1 shows the block diagram of the system. The ultrasonic sensor, servo motor and L298N motor driver are connected to Raspberry Pi 3 through GPIO pin while the USB webcam was connected to Raspberry Pi 3 through USB 2.0. The Raspberry Pi 3 was powered by a 5V power bank and the DC motor powered up by a 7.2V battery shown as the red arrow. The DC motor was controlled by L298N motor driver but Raspberry Pi 3 send the control signal to the L298N motor driver and control the DC motor to turn clockwise or anti clockwise. The servo motor is powered up by 5V on board voltage regulator (red arrow) at L298N motor driver from 7.2V battery. The ultrasonic sensor is powered up by 5V from Raspberry Pi 3 5V output pin. The smartphone communicate with Raspberry Pi 3 through Wi-Fi signal and the PlayStation 4 controller communicate with Raspberry Pi 3 through Bluetooth.

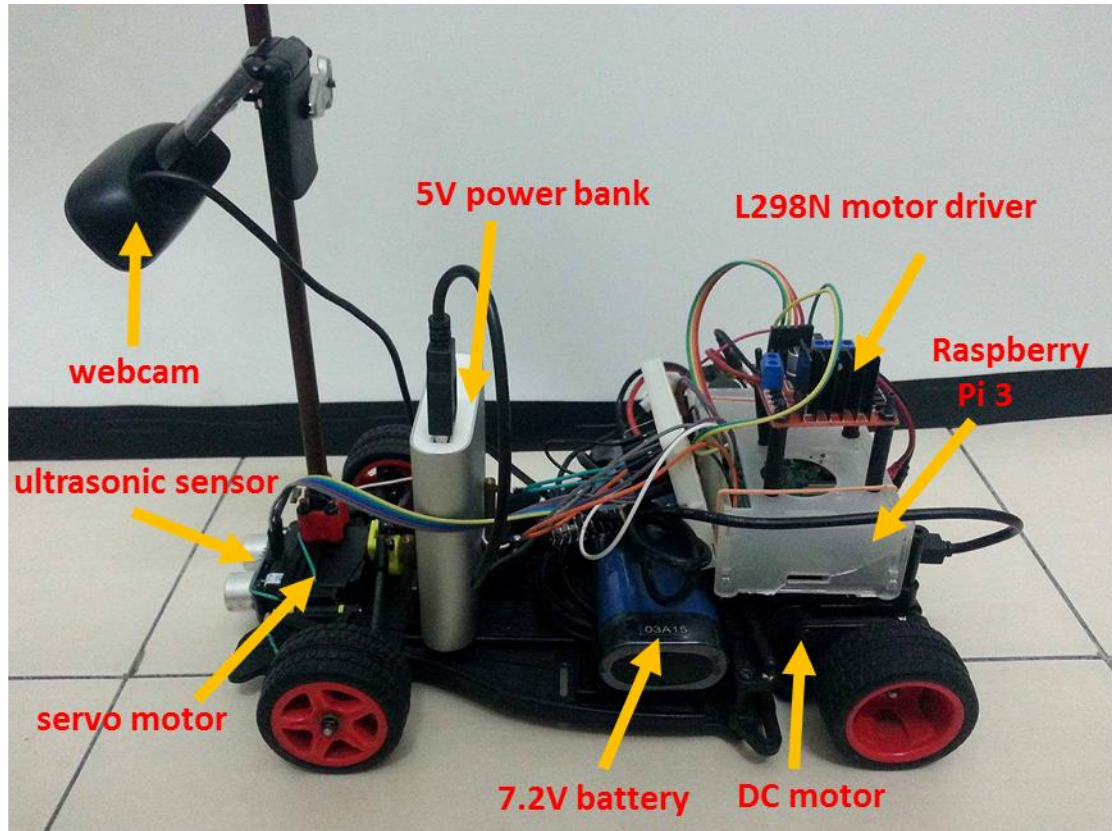


Figure 3.2 Final Product View

The aim of this project is to build an autonomous RC car prototype. First of all, a car base is needed for hardware part. The car base will hold up all the other hardware components to build the autonomous RC car. The car base have 4 wheels in which both front wheels will help the RC car to turn left and right with the help of servo motor. But these front wheels dint have DC motor attached. Each of the DC motor are attached to the both of the back wheels. These back wheels responsible to move the car forward and backward. Both of these DC motor are powered by a 7.2V battery. In order to move the car forward and backward, L298N motor driver is used.

Raspberry Pi 3 is used in this project as the single board computer. The movement of the RC car will controlled by the Raspberry Pi 3. Besides that, a USB webcam is needed in this project. The webcam will connect to Raspberry Pi 3 in order to perform computer vision with the help of OpenCV library. For autonomous mode, an algorithm of lane tracking for the track are written in Python programming language. The algorithm captured the image by using the webcam and process the image to determine the angle of the servo motor to turn left or right. The position and the angle

of the webcam are important which allow the lane tracking algorithm to get the best result. An ultrasonic sensor is attached in front the car base. When the RC car is moving and detected obstacle in certain range, the RC car will move to another lane of the track. After that, the RC car will move back to the original lane after pass by the obstacle. Therefore, the RC car able to move in the two lane track and overtake obstacles. There are two manual mode to control the RC car in this project. One of it is control the RC car through smartphone and the other one is through joystick (PS4 controller). Pygame library is used to allow the joystick to communicate with Raspberry Pi 3 through Bluetooth. This is an extra feature that allow user to control the RC car in more realistic way.

For software parts, Python (IDE) software and Android studio are used in this project. Android studio is used to develop an Android application in Java programming language for letting user to control the RC car manually or autonomous with live view from the RC car USB webcam. The L298N motor driver, ultrasonic sensor and servo motor and USB webcam are controlled by Raspberry Pi 3 which the coding will be in Python programming language. The Raspberry Pi 3 will act as the TCP server and the smartphone will act as the TCP client. So, all of the user input data send from the Android application will received by Raspberry Pi 3 TCP server through the Raspberry Pi 3 hotspot and this will make the project portable without connect through a router . In order to send data from Android application to Raspberry Pi 3, socket library is needed.

Chapter 4: Methodology

Prototyping model was used as the methodology for this project. A prototype is the demo version of the final product and the prototype act as a sample to test the process. Prototype model is defined as a method of system development. Prototype model consists the following phases as Figure 4.1.

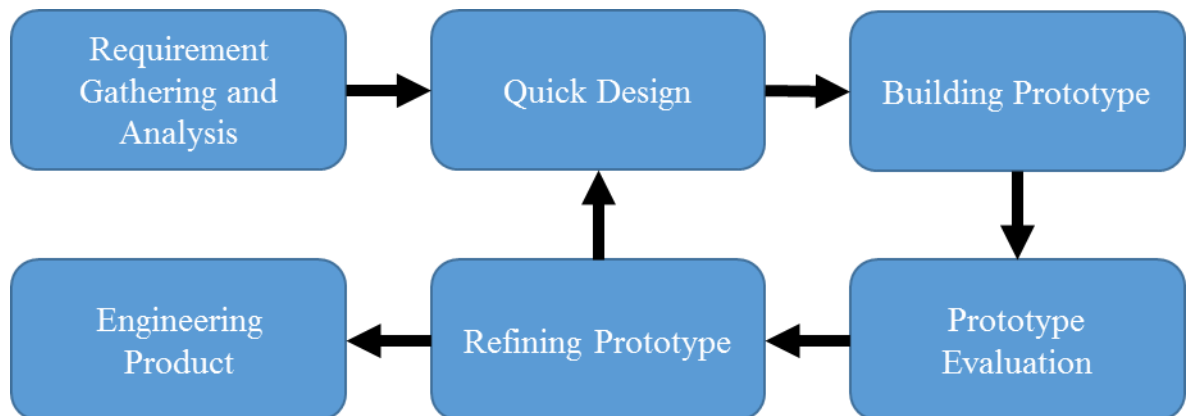


Figure 4.1 General Steps for Prototyping Model

Explanation for each stage in prototyping model:

- **Requirement Gathering and Analysis:** Before a project is started, all the requirement must be gathered and analyze all the gathered requirement of the system.
- **Quick Design:** A quick design about the system has been generated after the requirement gathering and analysis stage. The important components of the system must be included in the design. By using the quick design, user will able to know how the system works.
- **Building Prototype:** Before the prototype is developed, all of the information that get from the previous stage design is gathered and modified. Those information become the minimum requirement for the system.
- **Prototype Evaluation:** In the process of developing the prototype, the prototype is evaluated with the feedback of user.

- **Refining Prototype:** In this phase, the prototype will be modified depends on the requirement of the user, after the user done assess the prototype. If the user evaluation is good, the finalized system will be created with the finalized prototype specification.
- **Engineering Product:** In this stage, the evaluation and testing must be done on the final product to minimize downtime with the help of life time maintenance.

In this project, the first step is to analyses all the requirements for the project. The hardware components and the software tools are listed down. All of the hardware components datasheet has been studied. Then in quick design stage, the system quick design is created by drawing out the block diagram of the system and flowchart of the system. After that, a prototype RC car was built by using the quick design in the previous stage. Then, user evaluate the prototype RC car and the suggestion and comments are collected. In refining prototype stage, if the prototype RC car does not meet the requirement of customer, the process will back to second stage quick design to redesign the system to fulfill customer requirement. If the customer satisfied with the developed prototype RC car, then it will went to last stage. In final stage, the RC car is evaluated and tested with all possible situations to minimize downtime.

Chapter 5: Hardware Development

5.1 Chassis

The chassis is the body of the RC car where all the components, webcam, battery, power bank, servo motor, Raspberry Pi 3, L298N motor driver and ultrasonic sensor are mounted on the chassis. The chassis came with 2 DC motors which attached to the back wheels and front wheels will use to turn the direction of the RC car.

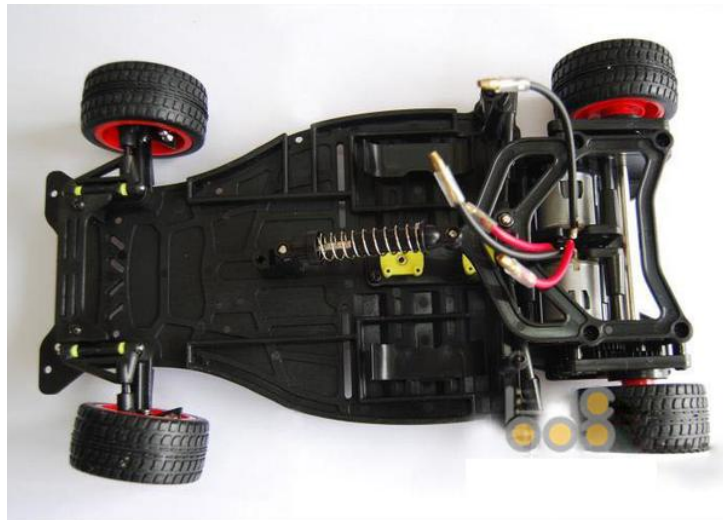


Figure 5.1.1 Top View of Chassis

5.2 Raspberry Pi 3 Model B

The development board used in this project is Raspberry Pi 3 model B. It acts as the controller and processing unit for the RC car. It is a single board computer based on the Broadcom BCM2837. It has 40 GPIO pins, HDMI video output, 4-pole composite output jack for video and audio, USB2.0 ports, CSI camera port, micro USB power input, , Ethernet port, DSI display port, microSD card slot, on board Bluetooth 4.1 and on board Wi-Fi. It used to control most of the hardware components used in this project. Table 5.2.1 shows Raspberry Pi 3 model B basic specification.

Architecture	ARMv8-A (64/32bit)
SoC	Broadcom BCM2837
CPU	1.2 GHz 64-bit quad-core ARM Cortex-A53
GPU	400MHz VideoCore IV
Memory (SDRAM)	1GB (shared with GPU)
USB 2.0 port	4
Video output	MIPI display interface (DSI) for raw LCD panels, HDMI (rev 1.3), composite video (3.5 mm TRRS jack)
On-board network	10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1
Power ratings	800mA (4.0W)
Operation Power	5V via micro USB
Dimension	85.60 mm x 56.5 mm x 17 mm
Weight	45g

Table 5.2.1 Raspberry Pi 3 Model B Basic Specification

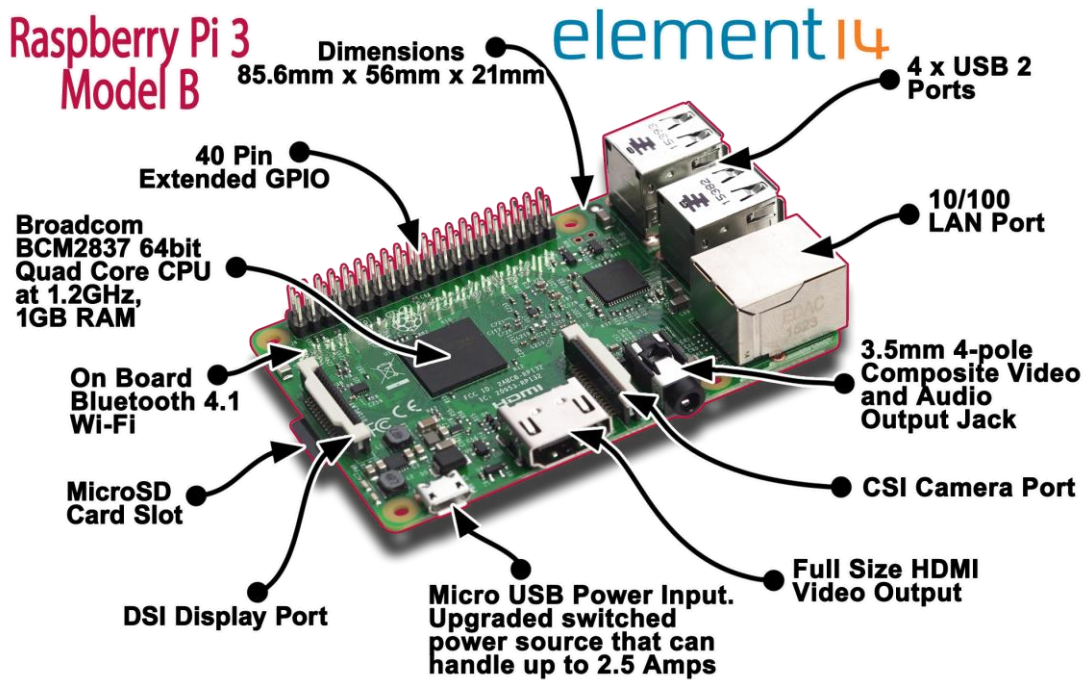


Figure 5.2.1 Raspberry Pi 3 Model B Technical Specification

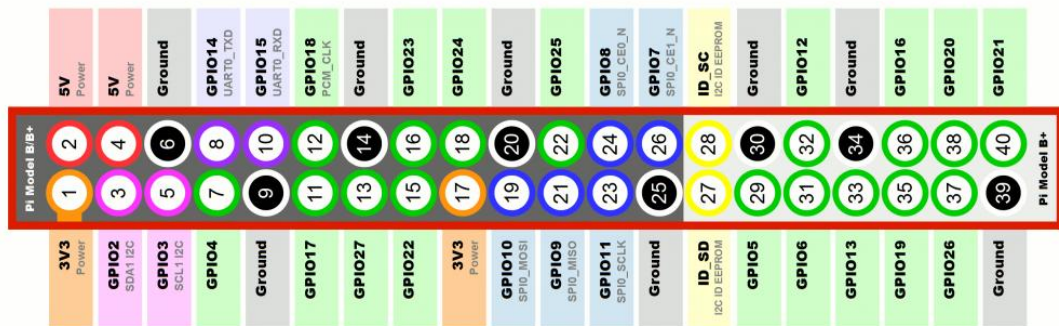


Figure 5.2.2 Raspberry Pi 3 Model B Pin Layout

5.3 L298N Dual H-Bridge Motor Driver

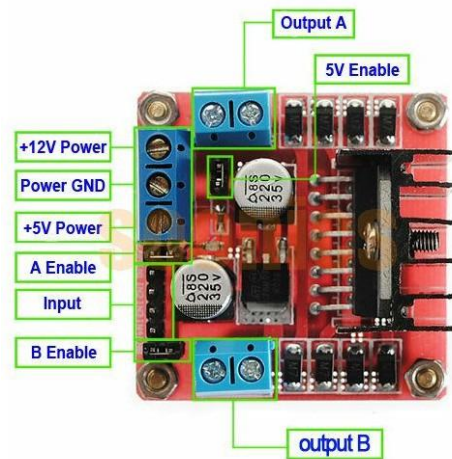


Figure 5.3.1 L298N Motor Driver

Operating Voltage	5V ~ 35V
Logic Power Output Vss	5V ~ 7V
Logical Current	0 ~36mA
Drive Current	2A (Max single bridge)
Max Power	25W
Controlling Level	Low: -0.3V ~ 1.5V High: 2.3V ~ Vss
Enable Signal	Low: -0.3V ~ 1.5V High: 2.3V ~ Vss

Table 5.3.1 L298N Motor Driver Specification

The direction rotation of the DC motor is controlled by H-bridge. The DC motor able to move in clockwise or anti clockwise when there is voltage applied across a load in either direction. Figure 5.3.2 show the H-bridge principle.

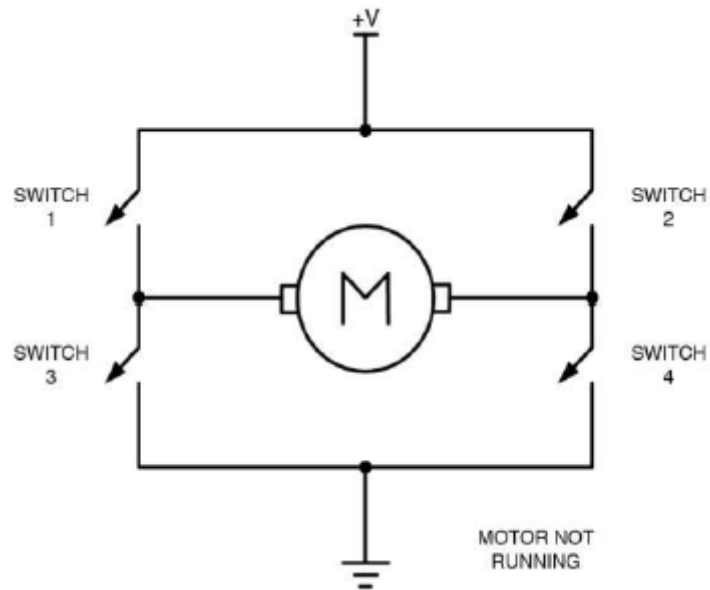


Figure 5.3.2 H-bridge Principle

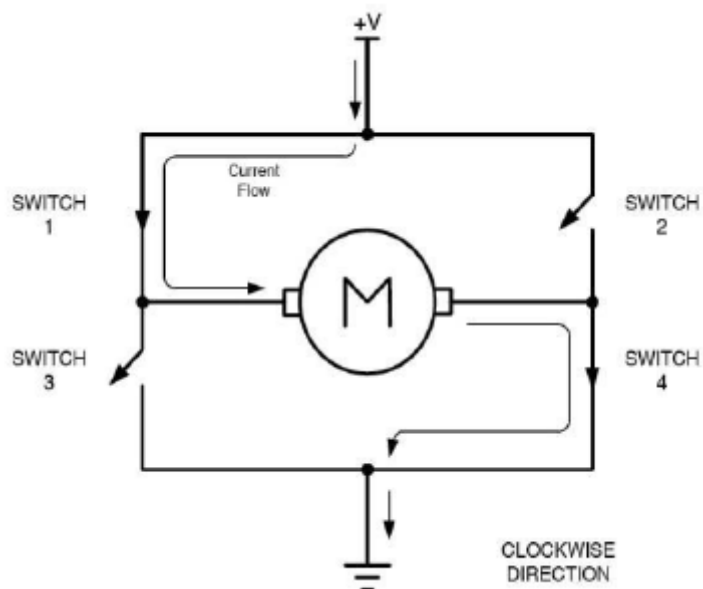


Figure 5.3.3 H-bridge (Clockwise direction)

To turn in clockwise direction, switch 1 and switch 4 are connected to the circuit. The current pass through switch 1 then through DC motor then pass through switch 4 then to the ground. Current starts to flow and the DC motor begin to rotate in a positive direction.

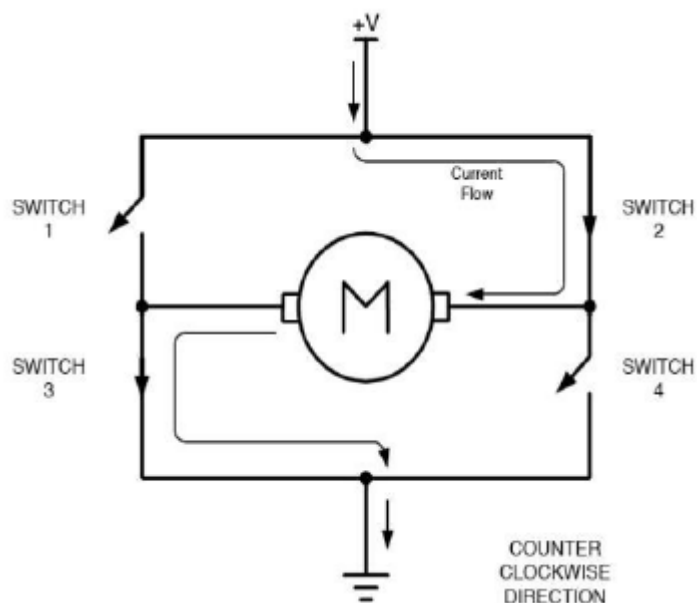


Figure 5.3.4 H-bridge (Anti-clockwise direction)

To turn in anti-clockwise direction, switch 2 and switch 3 will be will are connected to the circuit. The current pass through switch 2 then through DC motor then pass through switch 3 then to the ground. Therefore, this situation will cause the DC motor to energize in the reverse direction which leads to spin backward.

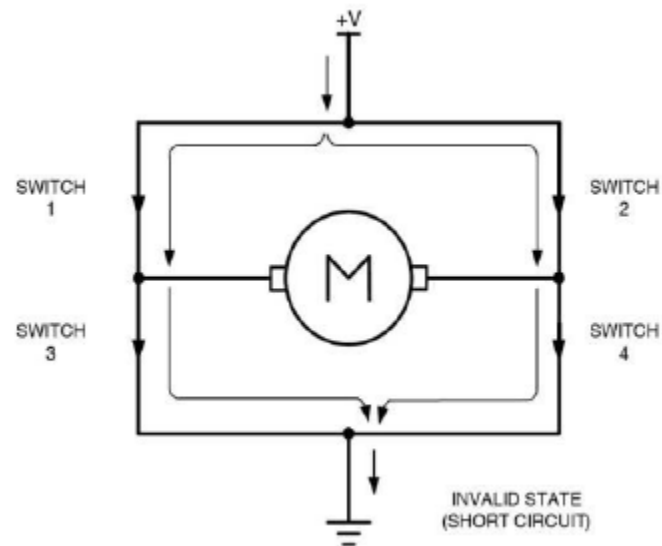


Figure 5.3.5 H-bridge (Short circuit)

When short circuit situation occur, all switches are connected to the circuit and the current flow through all the switches and then to the ground. Therefore, it will damage the H-bridge.

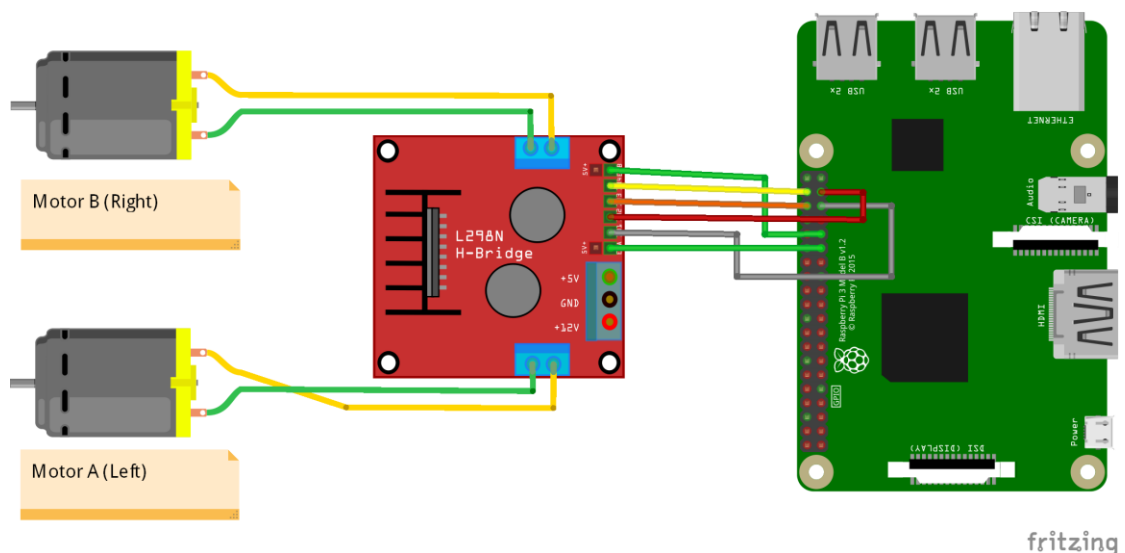


Figure 5.3.6 Connection of Raspberry Pi 3, L298N and DC motor

The remote control car only used 2 DC motor to move the car. The DC motor are attached to the both of the RC car back wheels. The front wheels are used to turn left or right. Each of the back wheels can be control to turn in clockwise and anti-clockwise direction.

In order to control the rotation and PWM of the DC motor, L298N motor driver is used. L298N has 4 int pins and 2 enable pins. The int pins are used to control the movement of the motor and the enable pins are used to control the PWM of the motor. Control the PWM of the motor only available in manual mode, which user control the RC car by Android smartphone or PS4 controller. In autonomous mode, the PWM of the motor are set to specific value. Table 5.3.2 shows the pins assigned to L298N.

L298N Pin	RPi 3 GPIO Pin	Pin Name	Motor	Movement/ PWM
Int 1	19	MOTOR1fwd	A (LHS)	Forward
Int 2	26	MOTOR1rev	A (LHS)	Reverse
Int 3	16	MOTOR2fwd	B (RHS)	Forward
Int 4	20	MOTOR2rev	B (RHS)	Reverse
A Enable	5	MOTOR1_PWM	A (LHS)	PWM
B Enable	6	MOTOR2_PWM	B (RHS)	PWM

Table 5.3.2 L298N Pin Assign

The RPi 3 GPIO pin can be assign to other pin refer to figure 5.2.2 Raspberry Pi 3 Model B Pin Layout. Figure 5.6 shows the Python code to control the DC motor using L298N motor driver.


```

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

MOTOR1fwd_GPIO = 19
MOTOR1rev_GPIO = 26
MOTOR2fwd_GPIO = 16
MOTOR2rev_GPIO = 20
MOTOR1_PWM = 5
MOTOR2_PWM = 6

GPIO.setup(MOTOR1fwd_GPIO,GPIO.OUT)
GPIO.setup(MOTOR1rev_GPIO,GPIO.OUT)
GPIO.setup(MOTOR2fwd_GPIO,GPIO.OUT)
GPIO.setup(MOTOR2rev_GPIO,GPIO.OUT)
GPIO.setup(MOTOR1_PWM,GPIO.OUT)
GPIO.setup(MOTOR2_PWM,GPIO.OUT)

Motor1 = GPIO.PWM(MOTOR1_PWM, 100)
Motor2 = GPIO.PWM(MOTOR2_PWM, 100)

def fwd():
    GPIO.output(MOTOR1fwd_GPIO,GPIO.HIGH)
    GPIO.output(MOTOR1rev_GPIO,GPIO.LOW)
    GPIO.output(MOTOR2fwd_GPIO,GPIO.HIGH)
    GPIO.output(MOTOR2rev_GPIO,GPIO.LOW)
    Motor1.ChangeDutyCycle(power)
    Motor2.ChangeDutyCycle(power)

```

Figure 5.3.7 RC Car Move Forward Code

There are many library can be used to control the PWM of the DC motor. The library that used to control the DC motor in this project is RPi.GPIO.

Command to install RPi.GPIO:

- `sudo apt-get install RPi.GPIO`

This library only allowed input value from 0 to 100. Firstly, the library is imported. Then the GPIO mode must set to GPIO.BCM. After that, all the pins are assigned. The fwd() function is used to control the PWM of the DC motor. The variable named power is used to pass into ChangeDutyCycle() function with the range of 0 to 100. The higher the value the faster the RC car move forward.

When the LHS and RHS motor is set to move forward, pin MOTOR1fwd and MOTOR2fwd are set to high and pin MOTOR1rev and MOTOR2rev are set to low. Then both of the pin MOTOR1_PWM and MOTOR2_PWM are set to certain value to change the duty cycle. So, the RC car able to move in different speed.

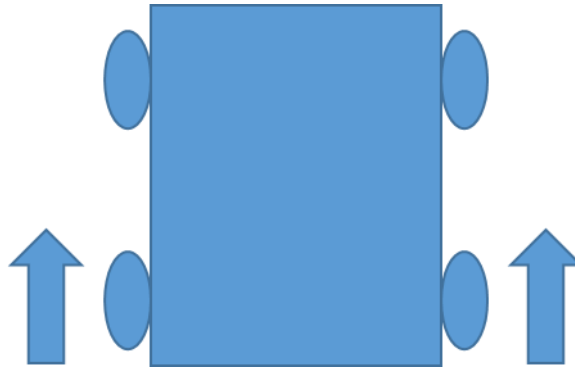


Figure 5.3.8 RC Car Move Forward

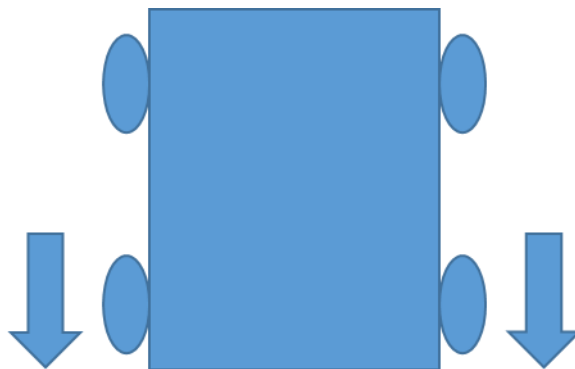


Figure 5.3.9 RC Car Move Backward

No	Test case	Result and discussion
1	GPIO.output(MOTOR1fwd_GPIO,GPIO.HIGH) GPIO.output(MOTOR1rev_GPIO,GPIO.LOW) GPIO.output(MOTOR2fwd_GPIO,GPIO.HIGH) GPIO.output(MOTOR2rev_GPIO,GPIO.LOW) Motor1.ChangeDutyCycle(50) Motor2.ChangeDutyCycle(50)	RC car move forward with 50% speed.
2	GPIO.output(MOTOR1fwd_GPIO,GPIO.HIGH) GPIO.output(MOTOR1rev_GPIO,GPIO.LOW) GPIO.output(MOTOR2fwd_GPIO,GPIO.HIGH) GPIO.output(MOTOR2rev_GPIO,GPIO.LOW) Motor1.ChangeDutyCycle(100) Motor2.ChangeDutyCycle(100)	RC car move forward with 100% speed.
3	GPIO.output(MOTOR1fwd_GPIO,GPIO.LOW) GPIO.output(MOTOR1rev_GPIO,GPIO.HIGH) GPIO.output(MOTOR2fwd_GPIO,GPIO.LOW) GPIO.output(MOTOR2rev_GPIO,GPIO.HIGH) Motor1.ChangeDutyCycle(100) Motor2.ChangeDutyCycle(100)	RC car move backward with 100% speed.
4	GPIO.output(MOTOR1fwd_GPIO,GPIO.LOW) GPIO.output(MOTOR1rev_GPIO,GPIO.HIGH) GPIO.output(MOTOR2fwd_GPIO,GPIO.LOW) GPIO.output(MOTOR2rev_GPIO,GPIO.HIGH) Motor1.ChangeDutyCycle(60) Motor2.ChangeDutyCycle(60)	RC car move backward with 60% speed.

Table 5.3.3 DC Motor Test Case

5.4 Futaba S3010 Standard High-Torque BB Servo Motor



Figure 5.4.1 Futaba S3010

Operating Voltage	4.8V ~ 6.0V
Speed	0.20 sec/60 ° at 4.8V 0.16 sec/60 ° at 6.0V
Torque	(5.2 kg-cm) at 4.8V (6.5 kg-cm) at 6.0V
Dimensions	40mm x 20mm x 38mm
Weight	41g

Table 5.4.1 Futaba S3010 Specification

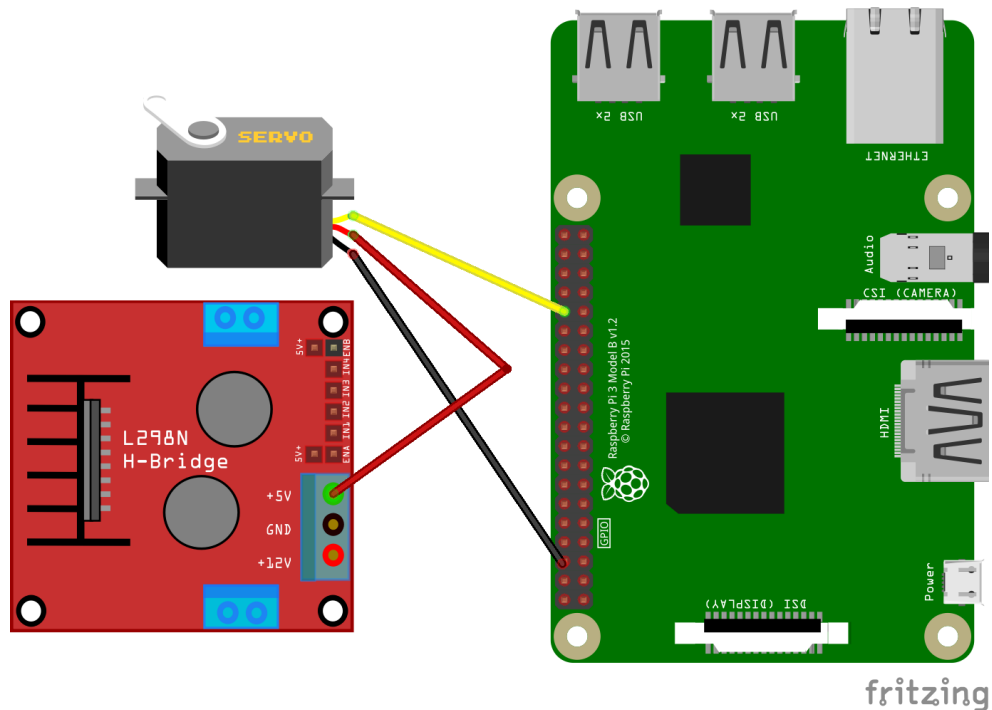


Figure 5.4.2 Connection of Raspberry Pi 3, L298N and Servo Motor

Normally the servo motor is operated by Raspberry Pi 3. After done some research from internet, powering a servo motor from Raspberry Pi 3 5V pin is a bad connection. Servo motor are draining more power to operate which will permanently damage the Raspberry Pi 3 board. The L298N motor driver have 5V voltage regulator. So, it able to convert 7.2V to 5V. Since L298N motor driver have an external 5V supply, the servo motor is connected to L298N motor driver. Therefore, the 7.2V will supplying power to DC motor and servo motor. The servo motor ground is connected to Raspberry Pi 3 and the signal of servo motor is connected to GPIO 12. The pin name was SERVO_GPIO.

```

from subprocess import call
call("sudo pigpiod", shell=True)
import RPi.GPIO as GPIO
import pigpio

GPIO.setmode(GPIO.BCM)
pi = pigpio.pi()
SERVO_GPIO = 12
GPIO.setup(SERVO_GPIO, GPIO.OUT)
try:
    while True:
        position = int(raw_input("enter servo value: "))
        pi.set_servo_pulsewidth(SERVO_GPIO, (position))
except KeyboardInterrupt:
    GPIO.cleanup()

```

Figure 5.4.3 Servo Motor Test Code

Command to install pigpio library:

- `wget abyz.co.uk/rpi/pigpio/pigpio.zip`
- `unzip pigpio.zip`
- `cd PIGPIO`
- `make`
- `make install`

The pigpio library was used. In order to for the pigpio library work properly, a sub process is called “sudo pigpiod”. The original PWM library used `ChangeDutyCycle()` function. The value can be use are from 0 to 100 but for pigpio library is using `set_servo_pulsewidth()` and the input value is from 500 to 2500. Therefore, `set_servo_pulsewidth()` is more sensitive than `ChangeDutyCycle()`. More accurate results is generated by using pigpio library.

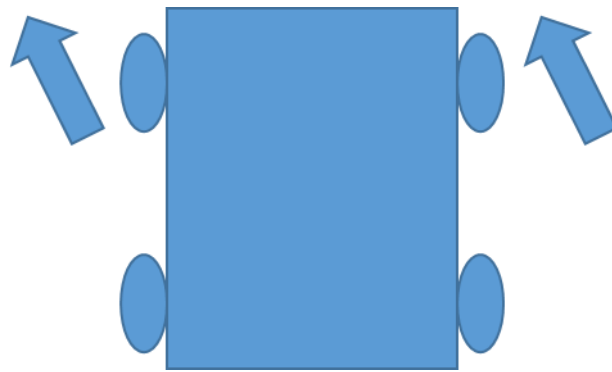


Figure 5.4.4 RC Car Turn left

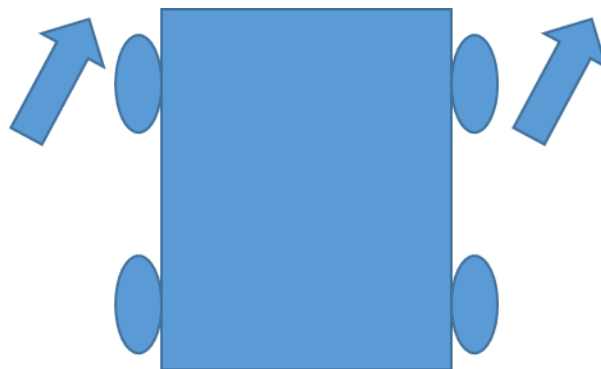


Figure 5.4.5 RC Car Turn Right

No	Test case	Result and discussion
1	<code>pi.set_servo_pulsewidth(SERVO_GPIO, 1595)</code>	RC car point in middle.
2	<code>pi.set_servo_pulsewidth(SERVO_GPIO, 1900)</code>	RC car turn right with maximum angle.
3	<code>pi.set_servo_pulsewidth(SERVO_GPIO, 1350)</code>	RC car turn left with maximum angle.

Table 5.4.2 Servo Motor Test Case

Since the servo motor is placed horizontally at the car base the input value should not be 0. After testing for several time, the input value to make the RC car pointing middle is 1595. So, the RC car can move in straight line. The input value to make the RC car turn to right direction is between 1596 and 1900. For the RC car to turn left direction, the input value will be between 1350 and 1594.

5.5 Ultrasonic Sensor (HC-SR04)



Figure 5.5.1 HC-SR04

Operating Voltage	5V
Operating Current	15mA
Operating Frequency	40kHz
Maximum Range	4m
Minimum Range	2cm
Measuring Angle	15 degree
Input Trigger Signal	10us TTL pulse
Output Echo Signal	Output TTL level signal, proportional with range
Dimension	45mm x 20mm x 15mm

Table 5.5.1 HC-SP04 Specification

The ultrasonic sensor works when the trigger input supplied with 10us. 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 5.5.2 shows how ultrasonic sensor works.

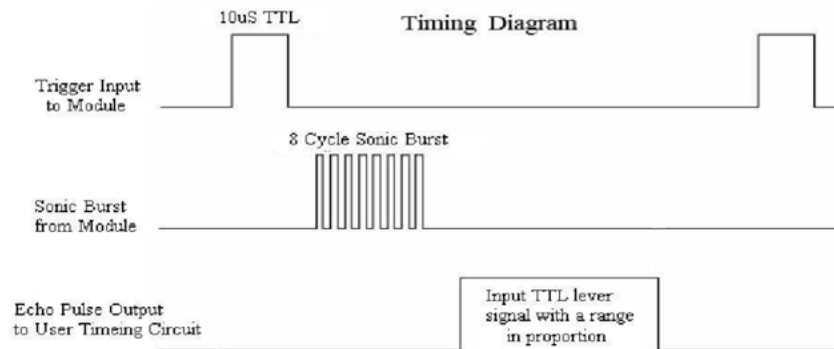


Figure 5.5.2 HC-SR04 Working Principle

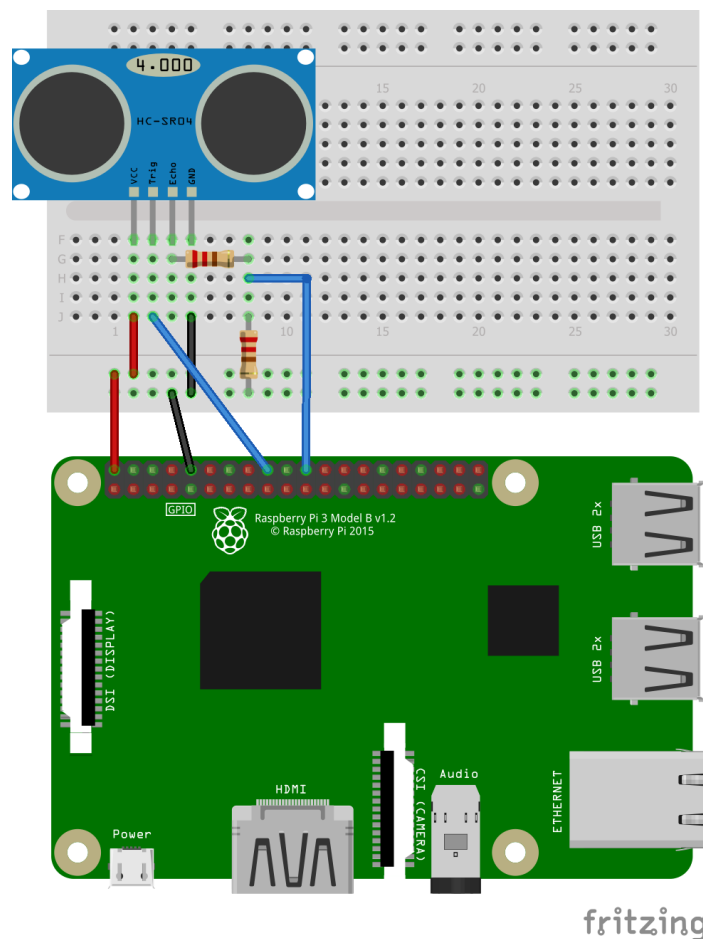


Figure 5.5.3 Connection of Raspberry Pi 3 and Ultrasonic Sensor

The signal outputs from ultrasonic sensor is 5V. The signal need to be converted from 5V to 3.3V to avoid permanent damage to Raspberry Pi 3. Voltage divider is used. V_{in} (ECHO) must be decreased from 5V to V_{out} 3.3V.

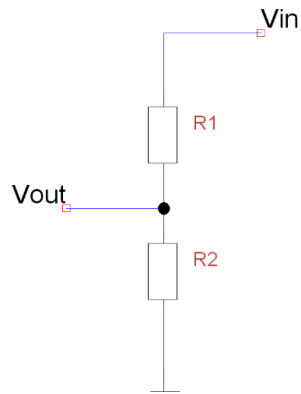


Figure 5.5.4 Voltage Divider

By using the formula

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2}$$

$$\frac{3.3}{5} = \frac{R_2}{1000 + R_2}$$

$$0.66(1000 + R_2) = R_2$$

$$0.34R_2 = 660$$

$$R_2 = 1941 \approx 2000$$

Therefore, for resistor 1 is 1000ohm and resistor 2 is 2000ohm. The trigger pin is assigned to GPIO 24 and echo pin is assigned to GPIO 25. The ultrasonic sensor output echo always low until been triggered.

```

import time
GPIO.setmode(GPIO.BCM)

TRIG = 23
ECHO = 24

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

try:
    while True:

        GPIO.output(TRIG, False)
        time.sleep(2)

        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        while GPIO.input(ECHO)==0:
            pulse_start = time.time()

        while GPIO.input(ECHO)==1:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        distance = pulse_duration * 17150
        distance = round(distance, 2)

        if distance > 2 and distance < 400:
            print "Distance:",distance - 0.5,"cm"
        else:
            print "Out Of Range"

```

Figure 5.5.5 Ultrasonic Sensor Test Code

Time library is needed to test the ultrasonic sensor. First, the TRIG set as low and delay for 2 seconds to let the sensor to settle. The ultrasonic sensor requires a short 10uS pulse to trigger the module, which will cause the sensor to start the ranging program (8 ultrasound bursts at 40 kHz) in order to obtain an echo response. So, to create the trigger pulse, the trigger pin was set to high for 10uS then set it low again. After that, the ECHO is checked whether is in low condition and save the last known time of the low pulse. In the next step, the ECHO is been checked again whether is in high condition and save the last known time of the high pulse. The pulse_duration

variable is used to store the time used. Then, the pulse_duration been converted to distance by multiplying 17150.

The formula to calculate the distance is

$$\text{Distance} = \text{Speed} * \text{Time}$$

The variable pulse_duration stored time is the time travel from the object and back again to the ultrasonic sensor. Therefore, the time will be divided by 2.

Speed of sound at sea level = 343 m/s or 34300 cm/s

$$\text{Distance} = \text{Speed} * \text{Time} / 2$$

$$\text{Distance} = 34300 * \text{Time} / 2$$

$$\text{Distance} = 17150 * \text{Time}$$

After calculate the distance the result is rounded to 2 decimal places. For accurate distance, calibration of 0.5 cm is added to the final result.

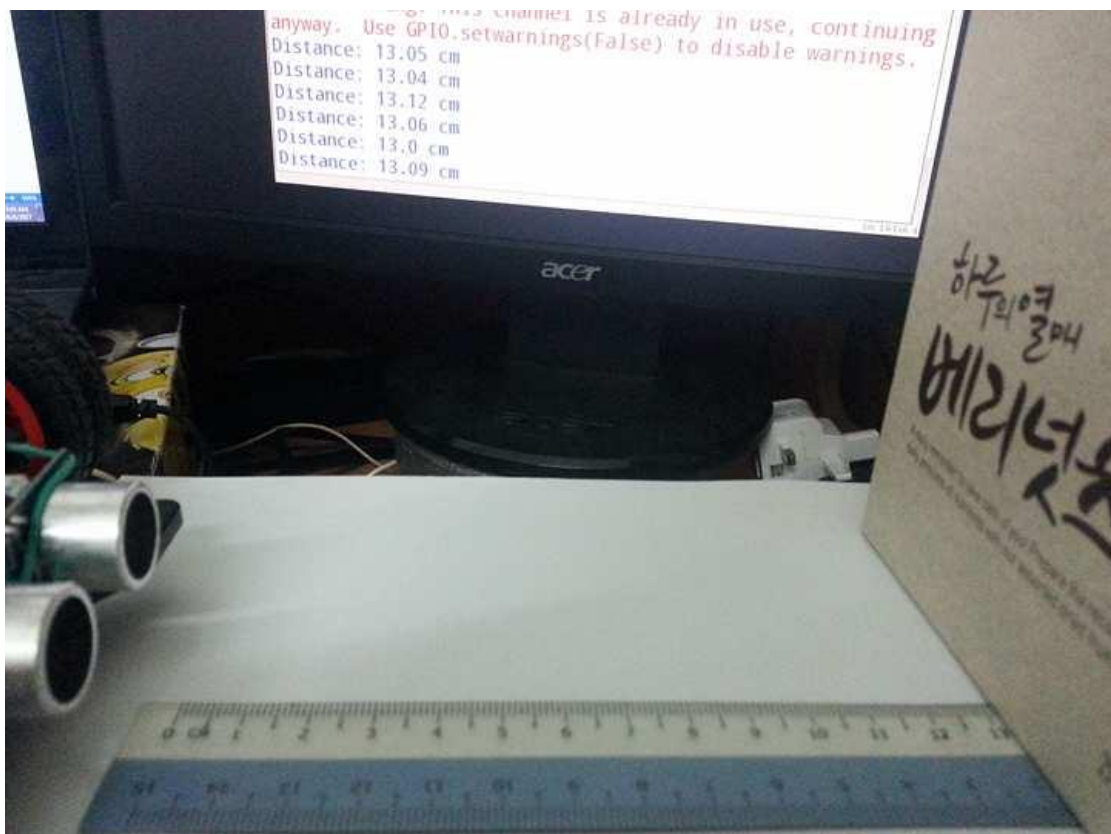


Figure 5.5.6 Ultrasonic Sensor Test Result

5.6 Logitech USB Camera (HD Webcam C270)



Figure 5.6.1 Logitech Webcam C270

Focus Type	Fixed
Field of View (FOV)	60 °
Image Capture (4:3 SD)	320x240, 640x480 1.2 MP, 3.0 MP
Image Capture (16:9 W)	360p, 480p, 720p
Video Capture (4:3 SD)	320x240, 640x480, 800x600
Video Capture (16:9 W)	360p, 480p, 720p
Frame Rate (max)	30fps at 640x480

Table 5.6.1 Logitech Webcam C270 Specification

The webcam is connected to Raspberry Pi 3 through the USB 2.0 port. No driver is needed to make the webcam operate. But library is needed to view the image or video from the webcam through the raspberry Pi 3. In order to use the USB webcam in Raspberry Pi 3, MJPG-Streamer library need to be installed.

Step to set up MJPG-Streamer:

1. Install the three libraries that MJPG-Streamer uses:

- `sudo apt-get install libjpeg8-dev imagemagick libv4l-dev`

2. Since the videodev.h file has been replaced with videodev2.h. Manual modification need to be done.

- `sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h`

3. Download MJPG-Streamer

- `wget http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-streamer-code-182.zip`

4. Unzip the MJPG-Streamer source code and build it.

- `unzip mjpg-streamer-code-182.zip`
- `cd mjpg-streamer-code-182/mjpg-streamer`
- `make mjpg_streamer input_file.so output_http.so`

5. Install MJPG-Streamer

- `sudo cp mjpg_streamer /usr/local/bin`
- `sudo cp output_http.so input_file.so /usr/local/lib/`
- `sudo cp -R www /usr/local/www`

6. Export the path

- `export LD_LIBRARY_PATH=/usr/local/lib/`
- `source ~/.bashrc`

7. Start MJPG-Streamer

- `/usr/local/bin/mjpg_streamer -i "/usr/local/lib/input_uvc.so" -o "/usr/local/lib/output_http.so -w /usr/local/www"`

8. View the Webcam Live-stream

- `http://<raspberry_pi_3's_ip_address>:8080`

After installation, the command “input_file.so” is choose to stream the video. For “input_uvc.so” command, which is used for USB webcam, it copies JPG pictures from webcam to one or more output source. Therefore lower CPU usage is achieved for streaming larger resolution images at higher frame rate. For “input_file.so” command, which used for camera module have the similar feature with “input_uvc.so” but copies the JPG pictures from a directory. And the “output_http.so” command will stream the files to a web server. Besides that, the resolution can be changed but need to be supported by the webcam.

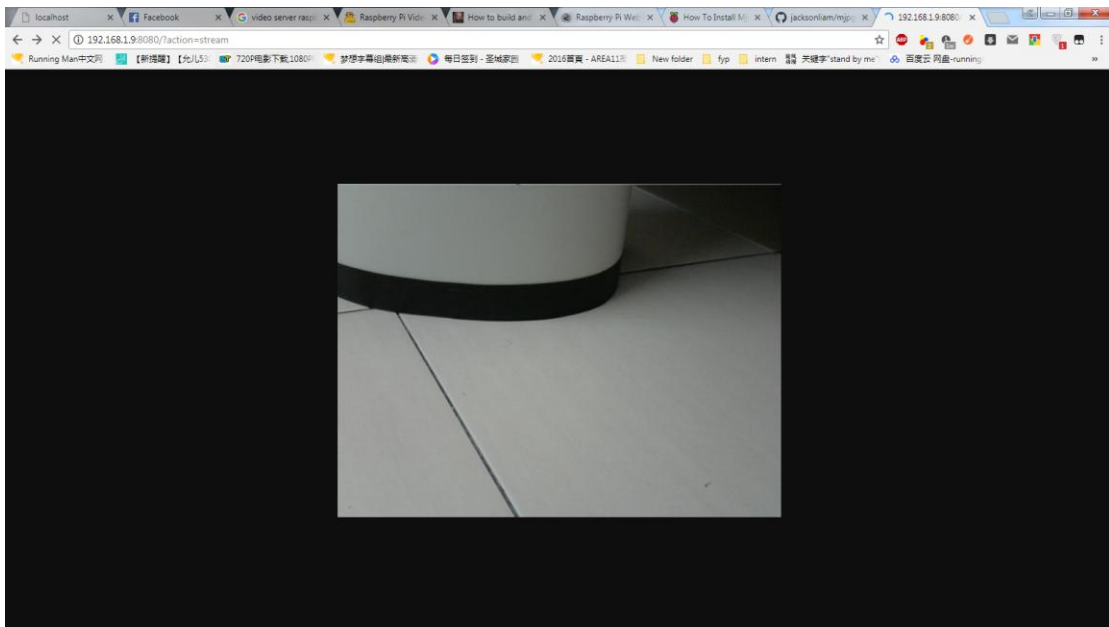


Figure5.6.2 Live View from Webcam through Browser

5.7 PlayStation 4 Controller



Figure 5.7.1 PS4 Controller

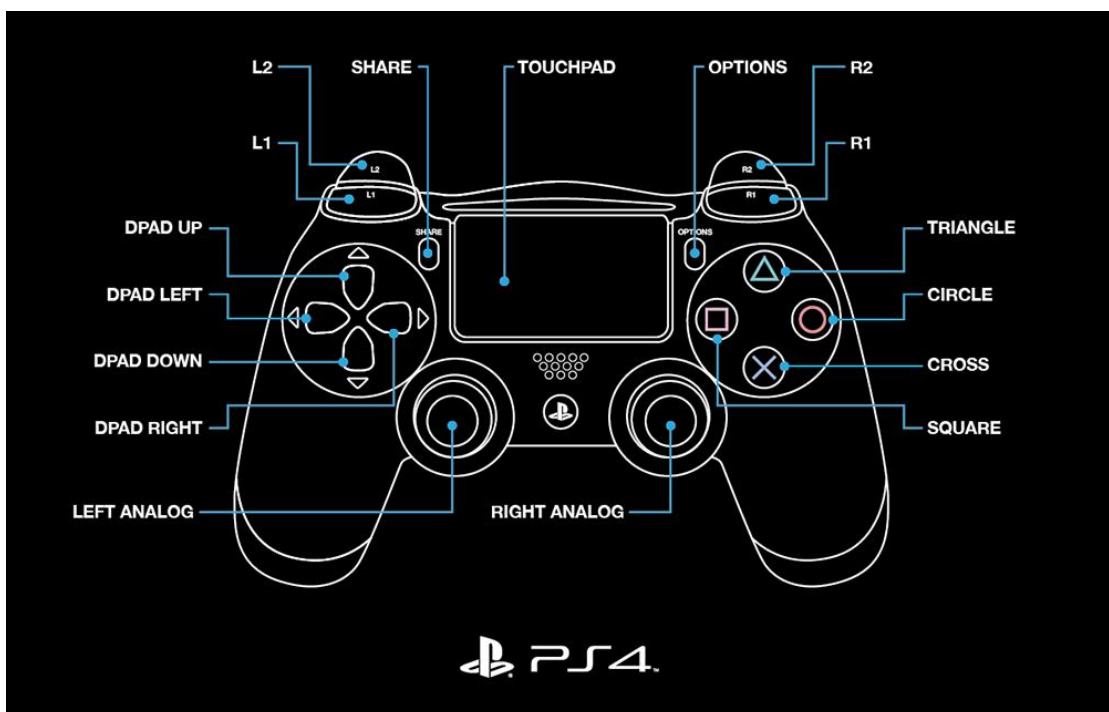


Figure 5.7.2 PS4 Controller Description

Touch Pad	2 Point Touch Pad, Click Mechanism, Capacitive Type
Bluetooth	Bluetooth Ver2.1 + EDR
Ports	USB (Micro B), Extension Port, Stereo Headset Jack
Motion Sensor	Six-axis motion sensing system (three-axis gyroscope, three-axis accelerometer)
External Dimensions	162mm x 52mm x 98mm
Weight	210g

Table 5.7.1 PS4 Controller Specification

Step to set up Bluetooth joystick in Raspberry Pi 3:

1. Install or update Bluetooth

- `sudo apt-get install pi-bluetooth`

2. Install ds4drv

- `sudo apt-get install python-dev python-setuptools`
- `sudo pip install ds4drv`

3. Install pygame

- `sudo apt-get install python-pygame`

PS4 controller is one of the option used to control the RC car in manual mode. The PS4 controller is connected to Raspberry Pi 3 through Bluetooth or USB cable. A library named “ds4drv” is installed in order to make the PS4 controller works with Bluetooth and library named “pygame” to get input from PS 4 controller. By using the library, the light bar color of the PS4 controller can be changed. First of all, a test code was written in Python to test PS4 controller to get all the axis and button initialization.

```

from subprocess import call
call("sudo ds4drv &", shell=True)
import pygame

pygame.init()
pygame.joystick.init()
joysticks = pygame.joystick.get_count()
if joysticks:
    print str(joysticks) + " joystick(s) detected!"
    for i in range(joysticks):          # Initialize each joystick
        joystick = pygame.joystick.Joystick(i)
        joystick.init()
        name = joystick.get_name()
        print "Joystick " + str(i) + " name: " + name

while True:
    if joystick.get_axis(0):
        print("left analog: left and right")
    if joystick.get_axis(1):
        print("left analog: up and down")
    if joystick.get_axis(2):
        print("right analog: left and right")
    if joystick.get_axis(3):
        print("right analog: up and down")
    for event in pygame.event.get():
        if event.type == pygame.JOYBUTTONDOWN:
            if joystick.get_button(0):
                print("square pressed")
            elif joystick.get_button(1):
                print("cross pressed")
            elif joystick.get_button(2):
                print("circle pressed")
            elif joystick.get_button(3):
                print("triangle pressed")
            elif joystick.get_button(4):
                print("L1 pressed")
            elif joystick.get_button(5):
                print("R1 pressed")
            elif joystick.get_button(6):
                print("L2 pressed")
            elif joystick.get_button(7):
                print("R2 pressed")
            elif joystick.get_button(8):
                print("SHARE pressed")
            elif joystick.get_button(9):
                print("OPTIONS pressed")
            elif joystick.get_button(10):
                print("left Axis pressed")
            elif joystick.get_button(11):
                print("right Axis pressed")
            elif joystick.get_button(12):
                print("PS pressed")
            elif joystick.get_button(13):
                print("Touch pad pressed")

```

Figure 5.7.3 PS4 Controller Test Code

A sub process with command “sudo ds4drv” was started. Then the pygame and pygame joystick module was initialized. Now, the pygame is waiting for the connection with PS4 controller through Bluetooth. The Share and PS button was pressed at the same time on the PS4 controller to enter pairing mode. The light bar will start blinking. If successfully paired with Raspberry Pi 3, the light bar will stop blinking. When detected the PS4 controller, the joystick is initialized by printing out the number of joystick and the name of the joystick. All of the button is pressed and the left analog and right analog are tested. From the test code, pygame initialized all of the button and analog as number. So, it is hard to identify them. The only way was test all the button and analog one by one. For the analog, up and down or left and right are shared the same axis. For example, if left analog axis 0 was used which was the left and right direction. The value will be from negative one (left side) to positive one (right side). After finish using the PS4 controller, the ds4drv must be stopped by using command “sudo killall ds4drv”. The test code from figure 5.7.3 was only for PS4 controller. If other controller was used, the pygame may initialized the button and analog to other value.

The PS4 controller is connected to Raspberry Pi 3 through Bluetooth and the test code is executed, but when button is pressed there is no result printed. Try to unpair the PS4 controller from Raspberry Pi 3.

Step to unpair PS4 controller:

1. Open terminal and execute the command:
 - bluetoothctl
2. A list of the Bluetooth device and their corresponding MAC address that paired to Raspberry Pi 3 is shown. Search for the PS4 controller MAC address. Type the command to unpair the PS4 controller.
 - remove aa:bb:cc:dd:ee:ff

Replace aa:bb:cc:dd:ee:ff with the MAC address of the PS4 controller.

Try to pair the PS4 controller to the Raspberry Pi 3 and execute the test code again. Repeat the steps until the Python 2 console print the result when button is pressed.

Chapter 6: Software Development

6.1 Raspberry Pi 3

6.1.1 Set up Raspberry Pi 3

Raspberry Pi 3 does not come with operating system. In order to use it, user must install the operating system by themselves.

Required materials:

1. Raspberry Pi 3
2. Power source with 5V and 2A or 2.5A
3. HDMI cable
4. A monitor or TV with HDMI port / A monitor or TV with VGA port (VGA to HDMI converter is needed)
5. Class 10 SD card with minimum 4GB size (recommended card size is 8GB)
6. USB mouse and keyboard

Step to set up Raspberry Pi 3:

1. Downlaod NOOBS (easy operating system installer which contains Raspbian) from <https://www.raspberrypi.org/downloads/noobs/>
2. Unzip the file downloaded in step 1 and copy into the class 10 SD card.
3. Insert the SD card to Raspberry Pi 3 and connect the mouse, keyboard and monitor/TV.
4. Plug the power source to turn on Raspberry Pi 3.
5. After boot up, a window will appear with a list of different operating systems that can be install. Select Raspbian then click install.

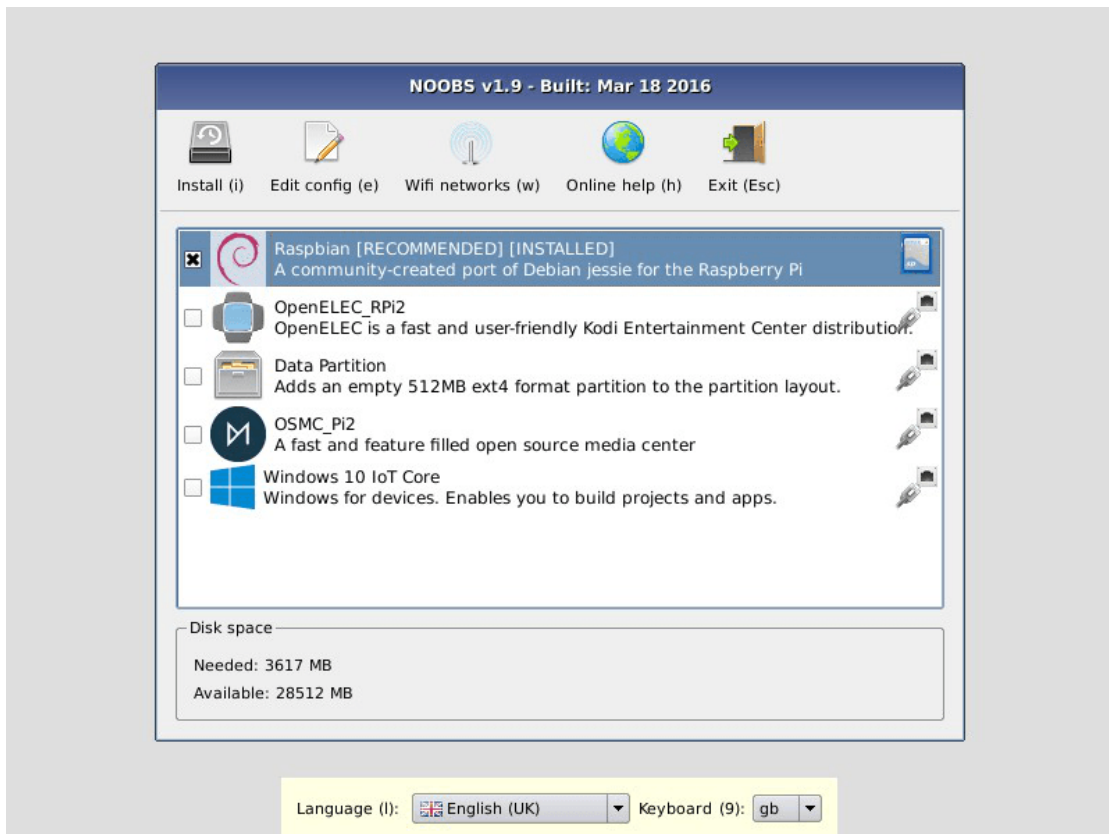


Figure 6.1.1 Installation of Raspbian on Raspberry Pi 3

6. After the installation, the Raspbian is boot up. Connect the Raspberry Pi 3 to internet.
7. Run the codes in terminal to update the Raspbian to latest version
 - `sudo apt-get update`
 - `sudo apt-get upgrade`
8. Go to preferences, then select Raspberry Pi Configuration, then select Interfaces tab enable VNC. Other feature can be enable too which depend on the user.

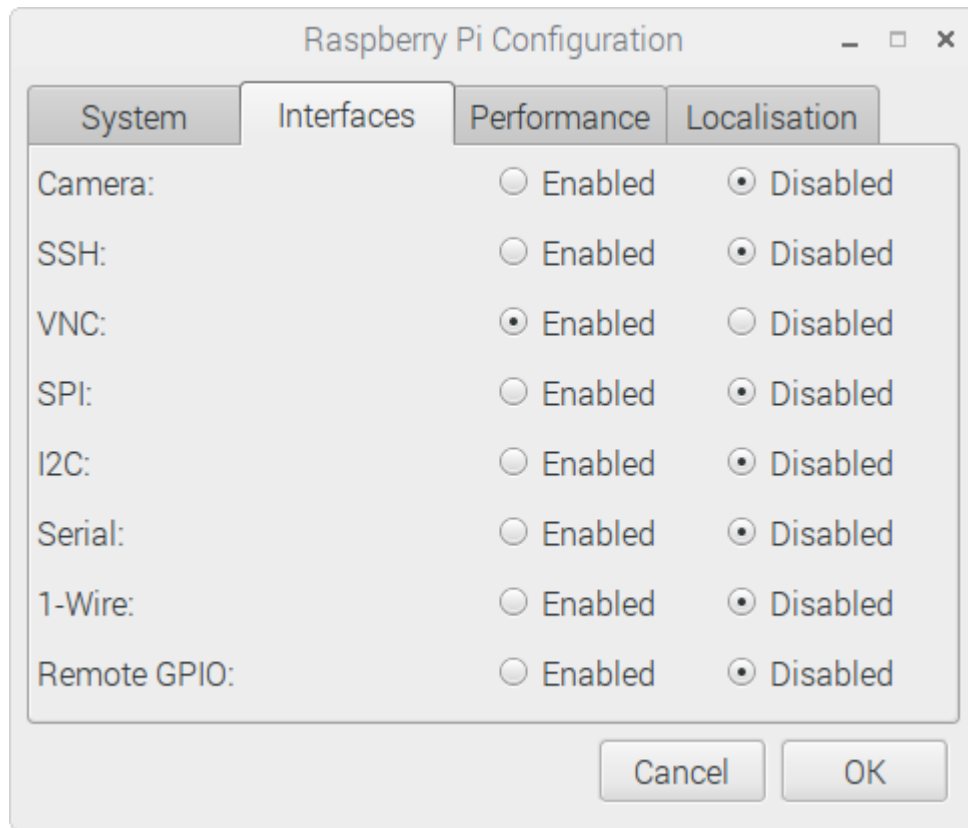


Figure 6.1.2 Enable VNC

9. Download and install VNC viewer on PC or laptop
<https://www.realvnc.com/en/connect/download/vnc/>
10. Get the IP address of the Raspberry Pi 3 that connected to the same network with the PC or laptop.
11. Key in the IP address of the Raspberry Pi 3 and fill in the username and password. The default username is "pi" and password is "raspberrypi". Click remember password and OK.

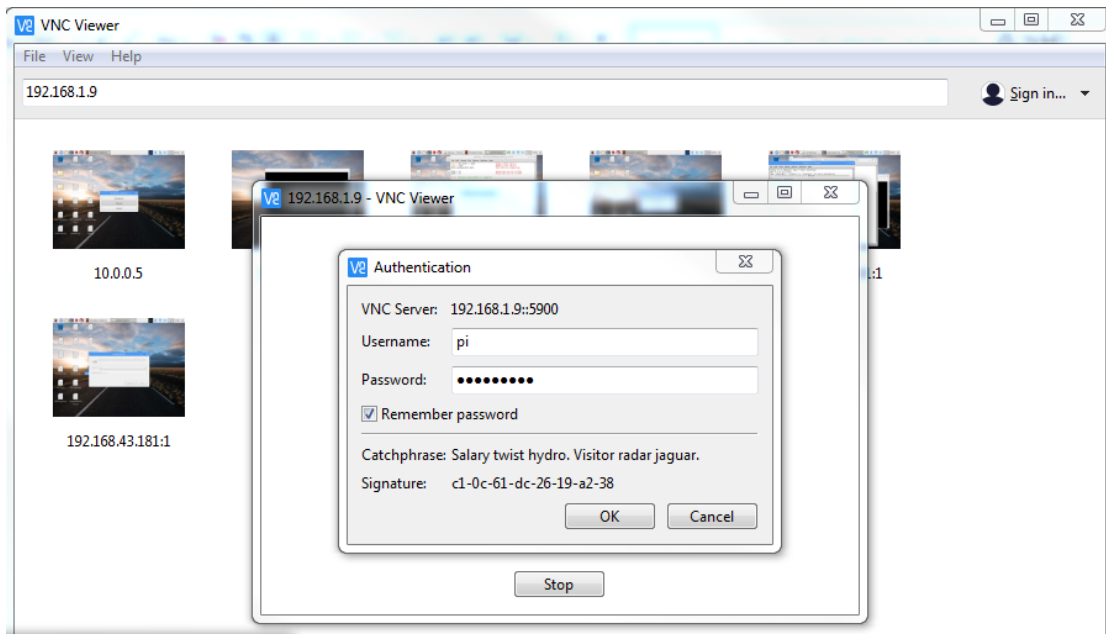


Figure 6.1.3 Connect to Raspberry Pi 3 using VNC Viewer

12. User able to control the Raspberry Pi 3 from PC or laptop.
13. Remove the HDMI cable, mouse and keyboard from Raspberry Pi 3. For the next boot, Raspberry Pi 3 will automatic connect to the network that user connected in the first boot.

6.1.2 Create hotspot using Raspberry Pi 3

To create a portable project, a router that used to connect Android smartphone and Raspberry Pi 3 for sending command from Android smartphone and live view at Android smartphone was replaced by Raspberry Pi 3 hotspot. Therefore, user does not have to bring along the router to control the RC car.

Step to set up the Raspberry Pi 3 hotspot:

1. Open a terminal and install hostapd.
 - `sudo apt-get install hostapd`
2. Install dnsmasq.
 - `sudo apt-get install dnsmasq`

3. The `hostapd` and `dnsmasq` was set to run when Raspberry Pi 3 is boot up. The automatic startup of these two programs need to be disabled.
 - `sudo systemctl disable hostapd`
 - `sudo systemctl disable dnsmasq`
4. Configure the `hostapd`
 - `sudo nano /etc/hostapd/hostapd.conf`
5. Paste the setting below to the `hostapd.conf` file.
 - `interface=wlan0`
`driver=nl80211`
`ssid=RPI3wifi`
`hw_mode=g`
`channel=6`
`wmm_enabled=0`
`macaddr_acl=0`
`auth_algs=1`
`ignore_broadcast_ssid=0`
`wpa=2`
`wpa_passphrase=1234567890`
`wpa_key_mgmt=WPA-PSK`
`wpa_pairwise=TKIP`
`rsn_pairwise=CCMP`

The `ssid` is the hotspot name and `wpa_passphrase` is the hotspot password. The minimum length for the password was 8 characters. User can change `ssid` and the `wpa_passphrase`.

6. Save the changes been done in step 5 by pressing `CTRL` and `X` then hit `ENTER`.
7. Update the file location of the config file been stored.
 - `sudo nano /etc/default/hostapd`

8. Apply the changes

- Change:

```
#demon_conf=""
```

to

```
demon_conf="/etc/hostapd/hostapd.conf"
```

9. Save the changes been done in step 8 by pressing CTRL and X then hit ENTER.

10. Configure dnsmasq

- `sudo nano /etc/dnsmasq.conf`

11. Go to the bottom of the file and add the following lines.

- #Pi3Hotspot Config

```
#stop DNSmasq from using resolv.conf
```

```
no-resolv
```

```
#Interface to use
```

```
interface=wlan0
```

```
bind-interfaces
```

```
dhcp-range=10.0.0.3,10.0.0.20,12h
```

The dhcp-range can change to other IP range. For example, 192.168.1.2, 192.168.1.10. 12h. The maximum user that able connect to Raspberry Pi 3 are 9 users. The 12h is after 12 hours the IP address no longer belong to the device and a new IP address will be assign to the device again.

12. Save the changes been done in step 11 by pressing CTRL and X then hit ENTER.

13. Update interfaces file

- `sudo nano /etc/network/interfaces`

14. Edit the following lines as below

- `auto lo wlan0`

```
iface lo inet loopback
```

```
allow-hotplug wlan0
```

```
iface wlan0 inet manual
```

```
# wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

15. Save the changes been done in step 14 by pressing CTRL and X then hit ENTER.

6.1.3 Hotspot Script

The Raspberry Pi 3 hotspot will be created if there was no network that Raspberry Pi 3 connected before in range.

```
createAdHocNetwork()
{
    echo "Creating RPI Hotspot network"
    ifconfig wlan0 down
    ifconfig wlan0 10.0.0.5 netmask 255.255.255.0 up
    service dnsmasq start
    service hostapd start
    echo " "
    echo "Hotspot network created"
    echo " "
}

macAddresses=('a8:9d:d2:d6:f1:5c')
connected=false
for macAddress in "${macAddresses[@]}"
do
    if iw wlan0 scan ap-force | grep $macAddress > /dev/null
    then
        echo "Starting supplicant for WPA/WPA2"
        wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant/wpa_supplicant.conf >$conf > /dev/null 2>&1
        echo "Obtaining IP from DHCP"
        if dhclient -1 wlan0
        then
            echo "Connected to WiFi"
            connected=true
            break
        else
            echo "DHCP server did not respond with an IP lease (DHCPOFFER)"
            wpa_cli terminate
            break
        fi
    else
        echo "Not in range, WiFi with MAC address:" $macAddress
    fi
done

if ! $connected; then
    createAdHocNetwork
fi
```

Figure 6.1.4 Hotspot Script

createAdHocNetwork() function was used to create the hotspot at Raspberry Pi 3. First, disable the Wi-Fi interface. Then configure it with IP address 10.0.0.5 with subnet mask 255.255.255.0 and enable the Wi-Fi interface. The IP address 10.0.0.5 can be change to other IP address as previously the dhcp created is in range of IP address 10.0.0.3 to 10.0.0.20. After that start the dnsmasq and hotapd service.

The macAddress variable is used to store the MAC address of the router or device. Multiple MAC address can be store in the variable with a space between the MAC address. For example,

```
macAddress=('XX:XX:XX:XX:XX:XX' 'YY:YY:YY:YY:YY:YY')
```

The MAC address can be changed to SSID, but not recommended. If the SSID contain special character, the script will not able to work properly.

If Raspberry Pi 3 detected those MAC address the hotspot will not be created and connect to the network (connected = true). If Raspberry Pi 3 does not detected those MAC address the hotspot will be created by calling createAdHocNetwork() function (connected = false).

After done the hotspot script, the script needed to put into the startup script. There are many ways to run the hotspot script in Raspberry Pi 3. One of the way was using the rc.local file.

Step to put the hotspot script into rc.local file:

1. Update the rc.local file
 - `sudo nano /etc/rc.local`
2. Go to the bottom of the file which the last line was “exit 0” and paste the hotspot script code before the “exit 0”
3. Save the changes been done in step 2 by pressing CTRL and X then hit ENTER.

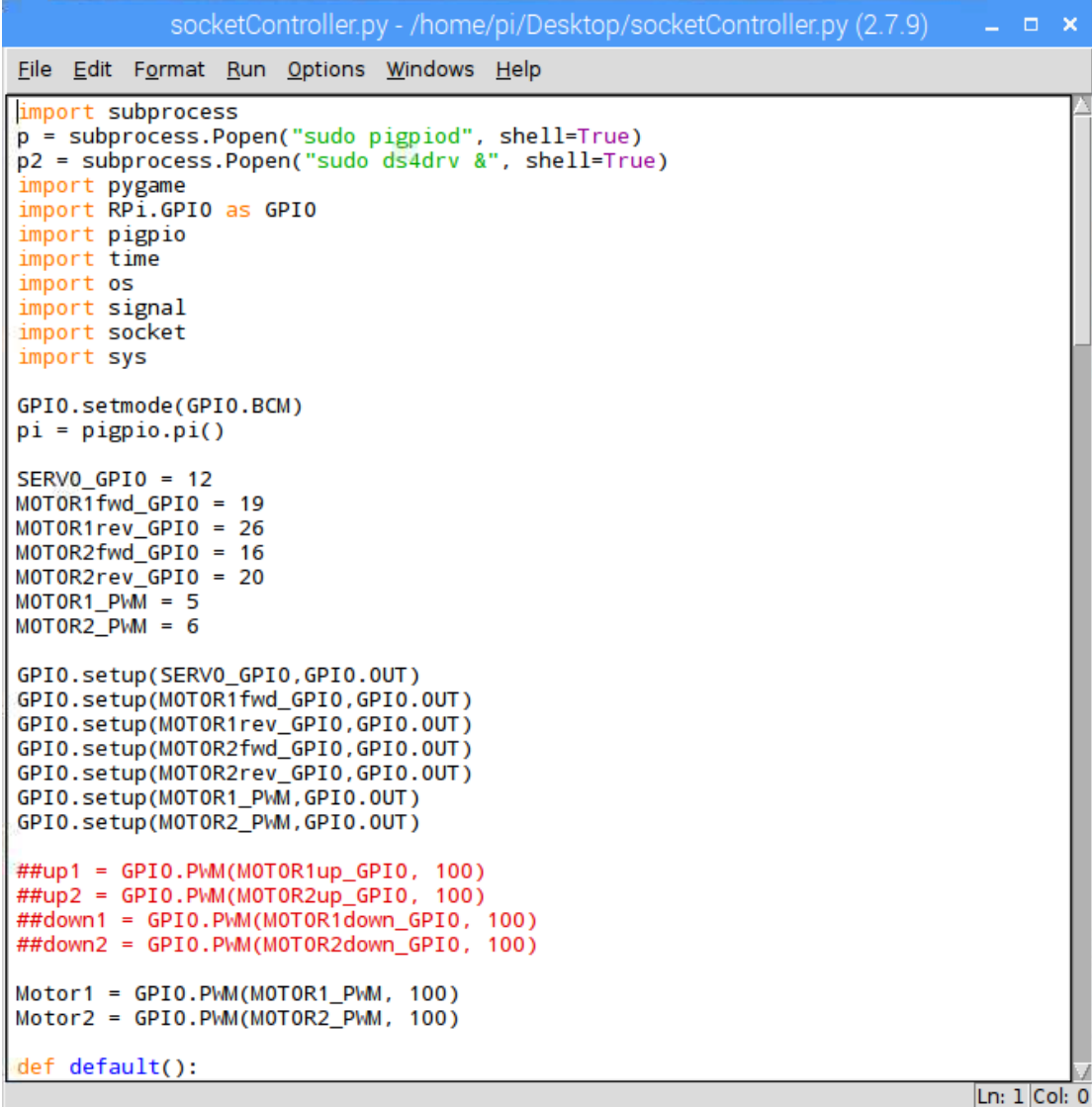
To test the script, user can change the MAC address using the macAdress variable to other value. So, the hotspot script not able to detect the network in range and the hotspot was created. Then, user able to control the Raspberry Pi 3 using the VNC viewer with the IP address of 10.0.0.5.

6.2 Python (IDLE)

Python 2 and 3 are included in Raspberry Pi 3. In year 2000, Python 2 was released and Python 3 was released in year 2008. Since Python 2 was released 8 years earlier than Python 3, most of the libraries was included in Python 2 while Python 3 have not ported yet. So, Python 2 is still mostly used nowadays, rather than Python 3. The different between Python 2 and Python 3 are some basic keys like print, integer division and raw input. Therefore, Python 2 was chosen to use in this project. Besides that, OpenCV a library that use for image processing is not included in Raspberry pi 3 Python 3 IDE. I have tried to install OpenCV on Python 3 and it works but it only can use through the terminal without the IDE which cause hard to debug and you will need to install all other library as well.

Install or update Python 2 on Raspberry Pi 3 with the command:

- `sudo apt-get install python-pip`



```

socketController.py - /home/pi/Desktop/socketController.py (2.7.9)
File Edit Format Run Options Windows Help

import subprocess
p = subprocess.Popen("sudo pigpiod", shell=True)
p2 = subprocess.Popen("sudo ds4drv &", shell=True)
import pygame
import RPi.GPIO as GPIO
import pigpio
import time
import os
import signal
import socket
import sys

GPIO.setmode(GPIO.BCM)
pi = pigpio.pi()

SERVO_GPIO = 12
MOTOR1fwd_GPIO = 19
MOTOR1rev_GPIO = 26
MOTOR2fwd_GPIO = 16
MOTOR2rev_GPIO = 20
MOTOR1_PwM = 5
MOTOR2_PwM = 6

GPIO.setup(SERVO_GPIO, GPIO.OUT)
GPIO.setup(MOTOR1fwd_GPIO, GPIO.OUT)
GPIO.setup(MOTOR1rev_GPIO, GPIO.OUT)
GPIO.setup(MOTOR2fwd_GPIO, GPIO.OUT)
GPIO.setup(MOTOR2rev_GPIO, GPIO.OUT)
GPIO.setup(MOTOR1_PwM, GPIO.OUT)
GPIO.setup(MOTOR2_PwM, GPIO.OUT)

##up1 = GPIO.PWM(MOTOR1up_GPIO, 100)
##up2 = GPIO.PWM(MOTOR2up_GPIO, 100)
##down1 = GPIO.PWM(MOTOR1down_GPIO, 100)
##down2 = GPIO.PWM(MOTOR2down_GPIO, 100)

Motor1 = GPIO.PWM(MOTOR1_PwM, 100)
Motor2 = GPIO.PWM(MOTOR2_PwM, 100)

def default():

```

Figure 6.2.1 Python 2 Interface

Since Python programming language is used, the alignment of all of the codes was important. If the codes are not in alignment will causes error when run the codes. The Python IDLE provided a way to solve the alignment problem. Select Format tab then select Untabify Region or use the shortcut key ALT + 6. The comment (ALT + 3) and uncomment region (ALT + 4) are useful which user does not have to comment out the code one by one. The shortcut key to run the code is F5. Besides that, user are allowed to change the font size and font type of the code through Options tab then select

Configure IDLE. The other way to run the code was through terminal. First, open the terminal and change to the directory that stored the python code file. Then, type the command “python2 xxx.py” and hit ENTER to run the code. The “xxx” replaced with the python file name with extension “.py”. This method is not recommended since it was hard to debug if there was any problems in the code.

6.2.1 TCP Socket Server on Raspberry Pi 3 Python 2

The TCP socket server was used to communicate with Android application through the Wi-Fi transmission. Raspberry Pi 3 act as the server and Android smartphone act as the client. User send the command from Android smartphone to Raspberry Pi 3.

```
import sys
import socket

HOST = '10.0.0.5' #this is your localhost
PORT = 8800

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.bind((HOST, PORT))
except socket.error as err:
    print 'Bind Failed, Error Code: ' + str(err[0]) + ', Message: ' + err[1]
    sys.exit()

print 'Socket Bind Success!'
s.listen(10)

try:
    while True:
        conn, addr = s.accept()
        print 'Connect with ' + addr[0] + ':' + str(addr[1])
        mode = conn.recv(64)
        print (mode)
except KeyboardInterrupt:
    s.close()
```

Figure 6.2.2 TCP Socket Server Test Code

First, import sys and socket library. HOST variable was the IP address of the Raspberry Pi 3 and the port used to communicate between Android smartphone and Raspberry Pi 3 was 8800. The port value can be changed. The socket.socket was used to create the socket and the AF_INET was the address format which was IP address. The SOCK_STREAM was the connection based on byte streams. Then bind() function was used to bind the socket to the server IP address with the port of 8800. After that, the socket was checked whether successfully bind or not. The server started the TCP listener to listen from the client which was the Android smartphone. The connection from the Android smartphone was accepted and receive the command using recv() function. Lastly, print out the command get from Android smartphone. The socket must be closed after using it.

6.2.2 Lane Tracking algorithms

OpenCV library is needed for image processing purpose. Therefore, OpenCV library need to be installed in Python 2.

Command to install or update OpenCV library:

- `sudo apt-get install python-opencv`

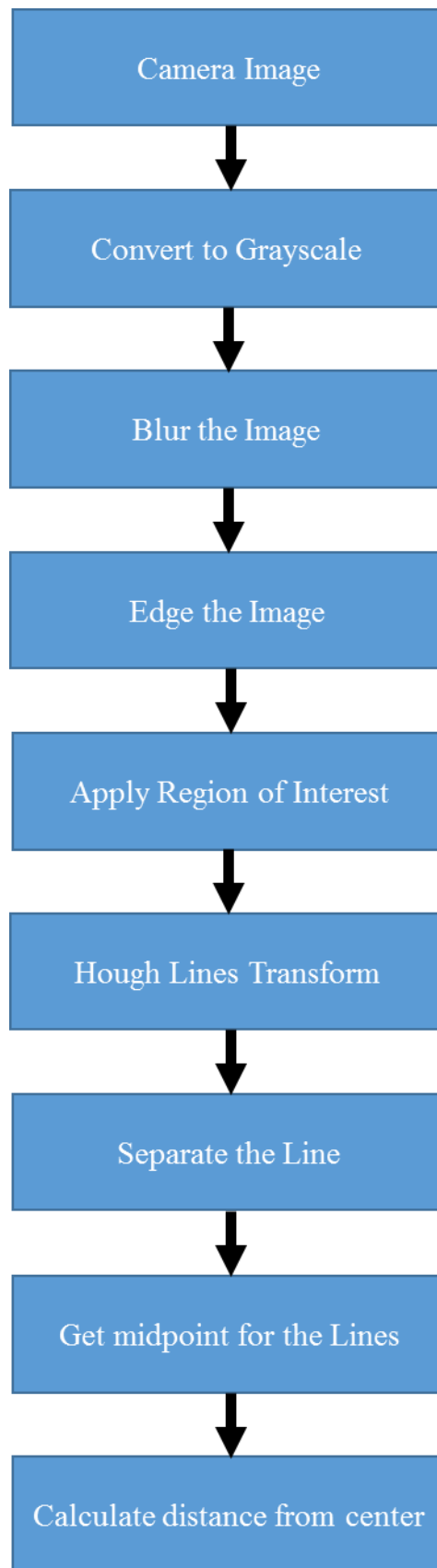


Figure 6.2.3 Flowchart of Lane Tracking

```

import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Display the resulting frame
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cv2.imwrite('c3.jpg',frame)
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()

```

Figure 6.2.4 Capture Frame using OpenCV Test Code

First, import the OpenCV library. Then use VideoCapture() function from OpenCV library. The 0 that pass into the VideoCapture() function is the ID of the webcam. Multiple webcam can be used by just passing other value like 1 which will be using the second webcam to capture. After that capture the video frame by frame using the cap.read() function. The output, frame was the image need for image processing. The imshow() function was used to show the frame captured. So, it will looks like a video. The frame can be stored by pressing “q” which will save the frame as “c3.jpg” by using the imwrite() function. When “q” was pressed, the program stored the image and stop the program. The capture need to be released for the next usage. So, cap.release() function and destroyAllWindows() function are executed.

After get the image from webcam, the image will be converted to grayscale using cv2.cvtColor(img, cv2.COLOR_BGR2GRAY).



Figure 6.2.5 Grayscale Image

Then the function `cv2.GaussianBlur(img, (kernel_size, kernel_size), 0)` was used to blur the image to reduce the noisy lines in the image. The `kernel_size` used was based on the input image.



Figure 6.2.6 Blurred Image

Then the `cv2.Canny(img, low_threshold, high_threshold)` was used to get the edge of all the lines in the image. The `low_threshold` and `high_threshold` are the values used to filter the lines in the image.

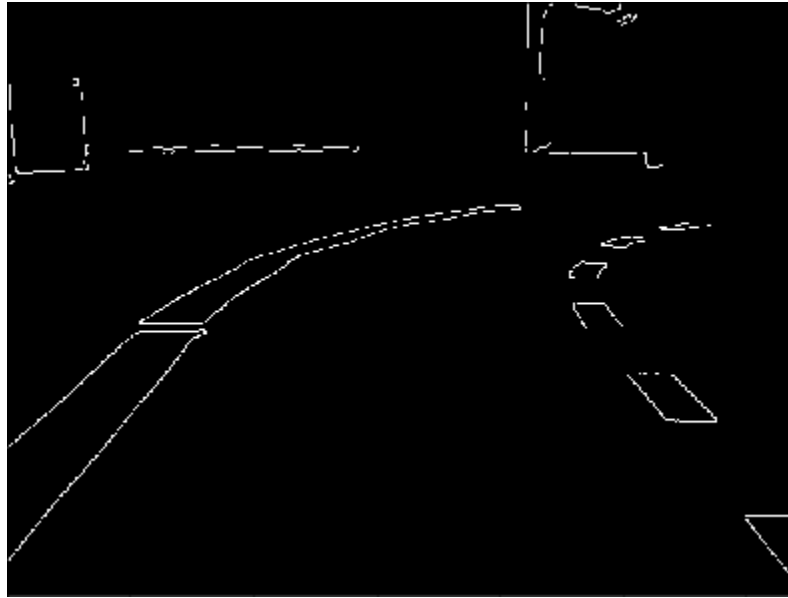


Figure 6.2.7 Edge Image

A mask was used to get the area of interest. Only part of the image been used for the lane tracking algorithm to reduce the time for processing the image.

```

bot_left = [0, 480]
bot_right = [640, 480]
apex_right = [640, 390]
apex_left = [0, 390]
v = [np.array([bot_left, bot_right, apex_right, apex_left], dtype=np.int32)]
mask = region_of_interest(edge, v)

```

The mask with the coordinates of the four corners to form a box shape. Then apply the mask to the image and get the result as in figure 6.2.6.

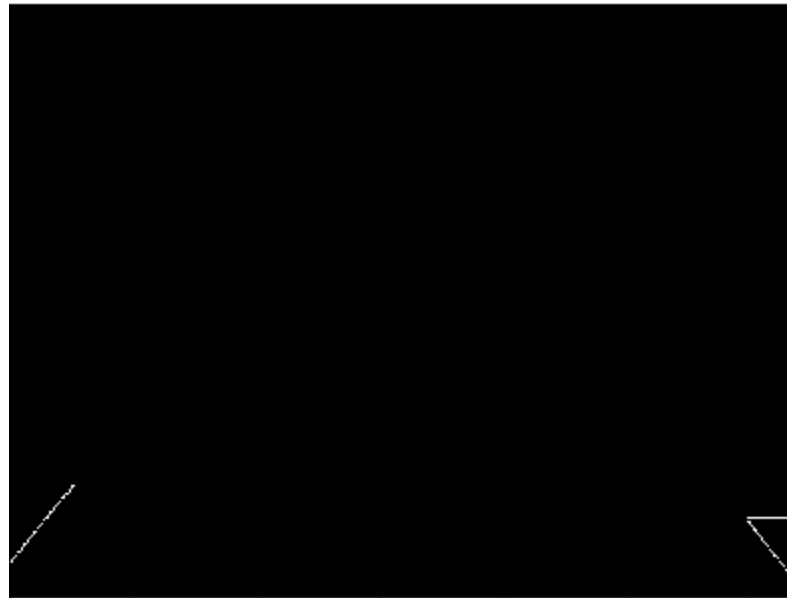


Figure 6.2.8 Masked Image

After that, the `cv2.HoughLinesP()` function was used to store all the lines coordinates to an array. The `minLineLength` and `maxLineGap` was important as `minLineLength` value is the minimum length of the lines can store and the `maxLineGap` value is the maximum gap of between the lines.

```
lines = cv2.HoughLinesP(mask, 0.5, np.pi/180, 25, np.array([]),  
                        minLineLength=40, maxLineGap=200)
```

Then the function `separate_line()` was used to split the line into left and right part for further usage.

```
right_lines, left_lines = separate_lines(lines)
```

The `right_lines` and `left_lines` arrays was used to calculate the midpoint between the lines. All the values in the `right_lines` arrays and `left_line` array been add up to get the average then divided by 2 to get the midpoint.

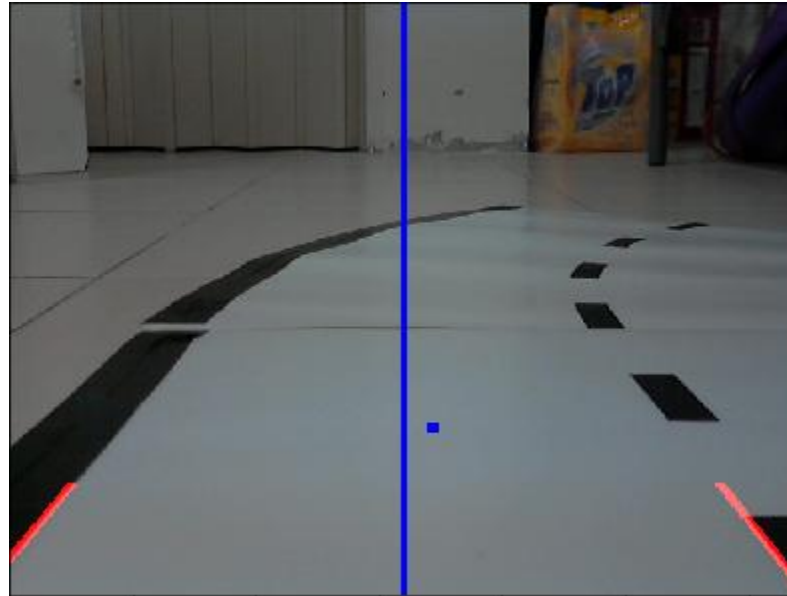


Figure 6.2.9 Result of Lane Tracking Algorithm

The blue line in the middle was the center of the track and the red lines was the lines retrieve from the input image. The blue dot was the midpoint between the left line and right line. By using the blue dot and the center of the track, the angle to turn the servo motor can be calculated and the RC car can find its way back to the center of the track.

6.3 Android Studio

There are few IDE for Android which support drag and drop feature which can help for those beginner build their first Android application. In this project Android studio was chosen, because Android studio was the official IDE for Android. There are few features in Android studio. The first feature was instant run which without building a new APK by just changing the code to the running Android application. Besides that, Android studio support C++ programming language. It also have an emulator that is fast and have many features. Other than that, importing Google code samples from GitHub was allowed. Android studio can be used to develop Android application for all Android device like android tablets, Android smartphone, Android wear and more. Android studio have simulation feature. Once the application been done, simulation can be used to test the application. User allowed to change the setting of the virtual device such as screen size, RAM size, storage, Android version, number of core to run the simulation.

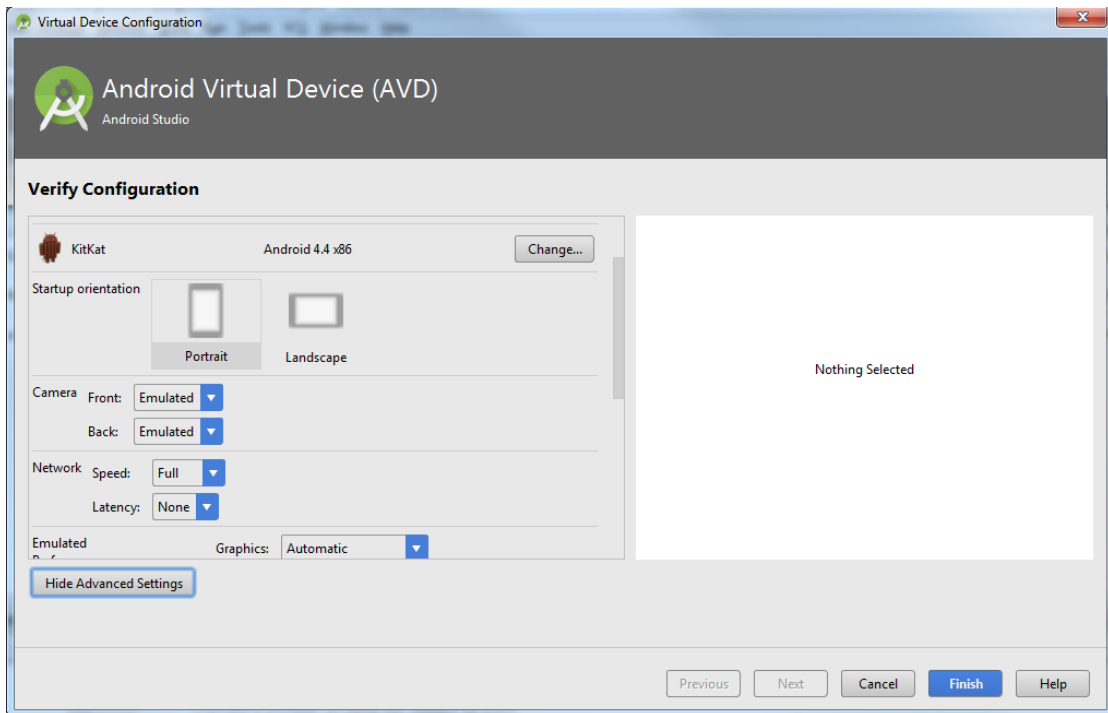


Figure 6.3.1 Configuration of Android Virtual Device

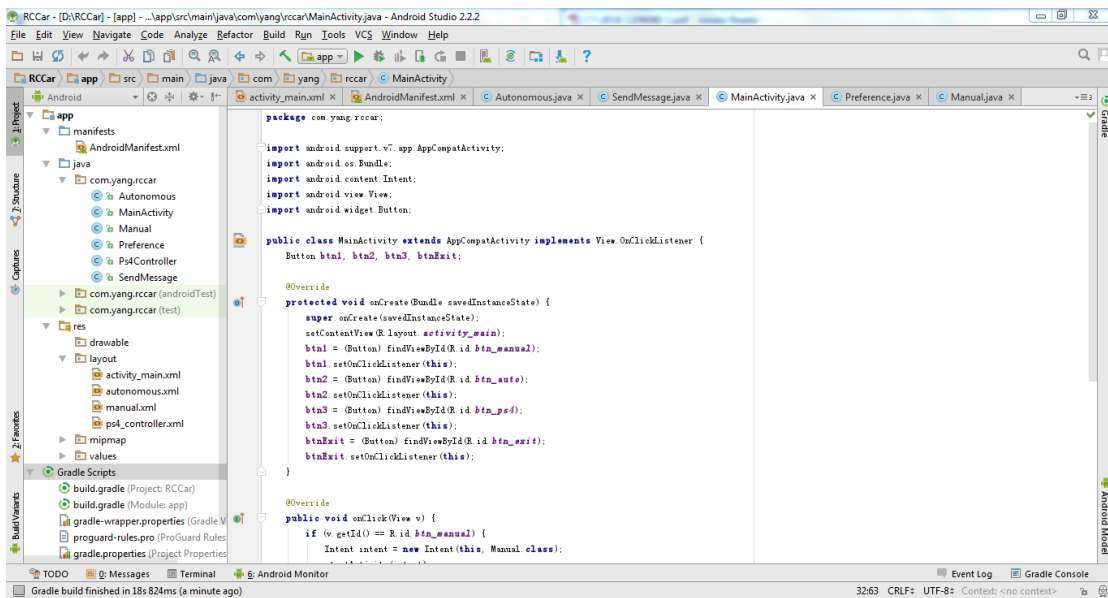


Figure 6.3.2 Android Studio Interface

6.3.1 Android Application

The Android application was created. It contain four pages which was main menu, manual mode, autonomous mode, and PS4 controller mode. Each of the pages must have its own class and layout interface. All the classes was written in JAVA programming language and the layout interface was XML.

```
public class Autonomous extends AppCompatActivity {  
  
    Button stop_btn;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.autonomous);  
  
        stop_btn = (Button) findViewById(R.id.btn_stop);  
        stop_btn.setOnClickListener((v) -> {  
            new SendMessage().execute("back");  
            finish();  
        });  
    }  
    @Override  
    public void onBackPressed() {  
        new SendMessage().execute("back");  
        //return;  
        super.onBackPressed();  
    }  
}
```

Figure 6.3.3 Code of Autonomous Class

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/output_autonomous"
    android:name="com.yang.recar.Autonomous"
    android:background="@drawable/child_background"
    tools:layout="@layout/autonomous">

    <Button
        android:id="@+id/btn_stop"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:textSize="20sp"
        android:alpha="0.85"
        android:text="Stop" />

</LinearLayout>

```

Figure 6.3.3 Code of Autonomous Layout

Design the layout first before start to write the code for the class. Each of the layout have their own id (tools:layout="@layout/autonomous"). The id was used to start the layout using the class (setContentView(R.layout.autonomous;)). It is similar with a button or others. First, a button with the text “stop” was created in the layout with the id “btn_stop” shown in figure 6.3.3. Then a variable button was created at the class with the name “stop_btn”. After that, the button variable was assigned by using the id created in the layout (stop_btn = (Button) findViewById(R.id.btn_stop). Lastly, the button listener (stop_btn.setOnClickListener) was used to done the job when the button is pressed. Inside the listener will be the code written that have to be done which was sending the command to TCP socket server at Raspberry Pi 3.

An extra class was created which was used to send the command to TCP socket server. The class name was “SendMessage”.

```

public class SendMessage extends AsyncTask<String, Void, Void>{
    private Exception exception;
    Preference p = new Preference();
    @Override
    protected Void doInBackground(String... params) {
        try {
            try {
                Socket socket = new Socket(p.getIpAddr(), p.getSocketPort());
                PrintWriter outToServer = new PrintWriter(
                    new OutputStreamWriter(socket.getOutputStream()));
                outToServer.print(params[0]);
                outToServer.flush();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } catch (Exception e){
            this.exception = e;
            return null;
        }
        return null;
    }
}

```

Figure 6.3.4 Code of SendMessage Class

It was similar with the TCP socket server created at Raspberry Pi 3. In this class the `java.net.Socket` library was used to send the command. First, the socket was created by getting the IP address of the Raspberry Pi 3 and the port used to transmit the command. The “`params[0]`” was the variable used to store the command. Then the `flush()` function was used to send the commands from TCP socket client to server.

Since the Android application was using Wi-Fi as transmission signal, the Android application must have permission to use the Wi-Fi. Therefore permission must be set in the `AndroidManifest.xml` file.


```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.yang.rccar">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="RC Car"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Manual" android:screenOrientation="landscape"
            android:configChanges="orientation|keyboardHidden" />
        <activity android:name=".Autonomous" android:screenOrientation="landscape"
            android:configChanges="orientation|keyboardHidden" />
        <activity android:name=".Ps4Controller" android:screenOrientation="landscape"
            android:configChanges="orientation|keyboardHidden" />
    </application>

</manifest>

```

Figure 6.3.5 Code of AndroidManifest.xml

The first highlighted box show the permission allowed to use in this Android application. The second highlighted box show the pages of the Android application. All of the pages must manually include in this file except the main menu which automatically included at this file. If the pages does not in this AndroidManifest.xml file. The Android application will crashed.

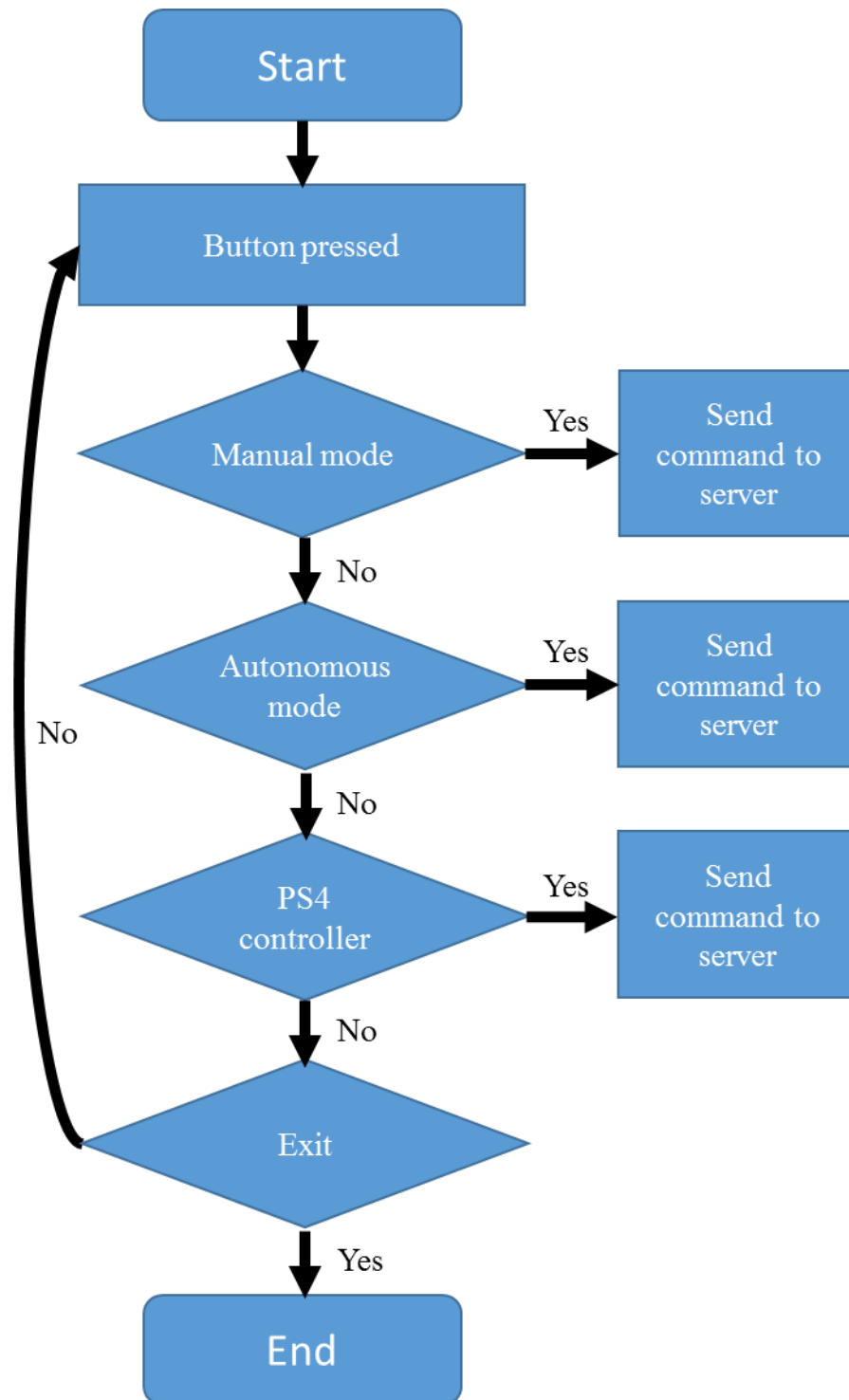


Figure 6.3.6 Flowchart of Android Application

When the Android application is started, there will be MANUAL, AUTONOMOUS, PS4 CONTROLLER and EXIT button. If MANUAL, AUTONOMOUS or PS4 CONTROLLER buttons was pressed, the Android application will send specific commands to Raspberry Pi 3 to run the part of the codes.

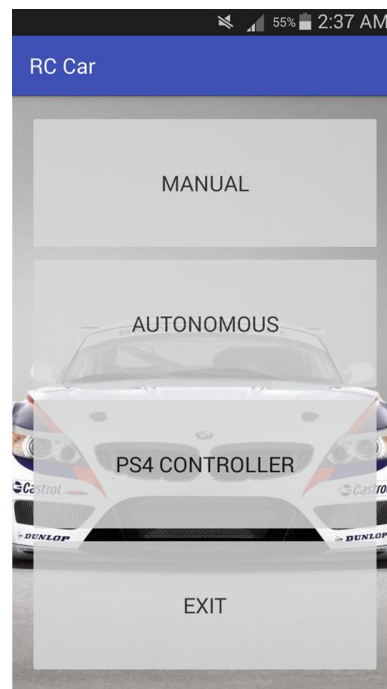


Figure 6.3.7 Android Application Menu Interface



Figure 6.3.8 Android Application Manual Interface

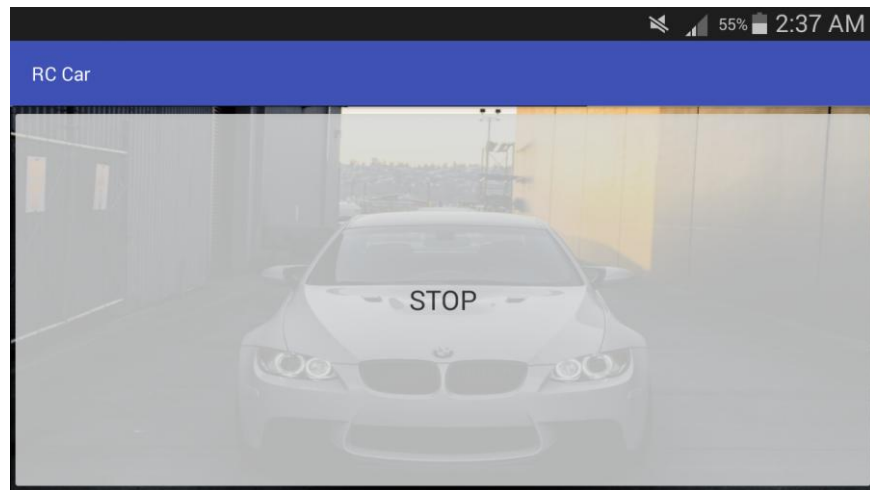


Figure 6.3.9 Android Application Autonomous Interface



Figure 6.3.10 Android Application PS4 Controller Interface

Chapter 7: System Integration

All the parts of the hardware and software are combined, the Python codes will be executed at Raspberry Pi 3 and Android application will be installed into the Android smartphone.

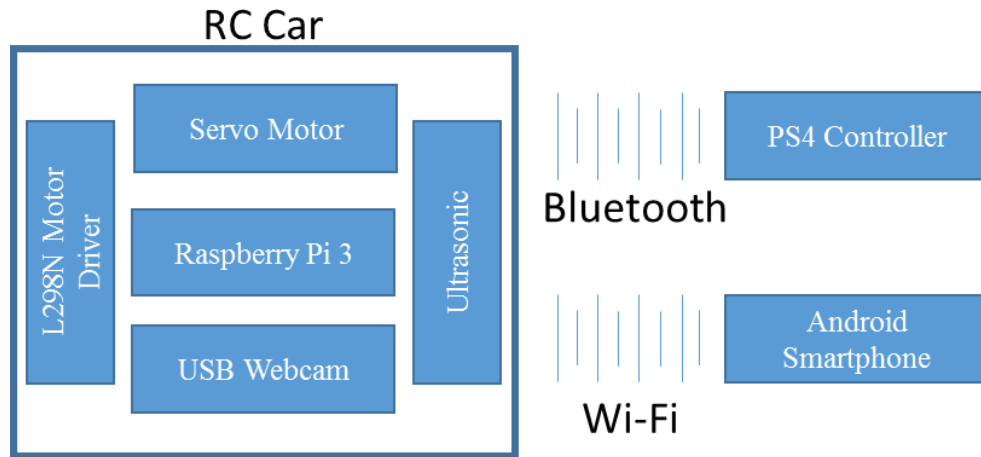


Figure 7.1.1 Final product Block Diagram

7.1 PS4 Controller + Servo Motor + DC Motor

In hardware development chapter, all of the buttons and axis of the PS4 controller been review. Not all of the button and axis been used. In this project, the PS4 controller only used for manual mode. So, the PS4 controller will control the movement of the RC car. Only the left analog up down and right analog left right been used.

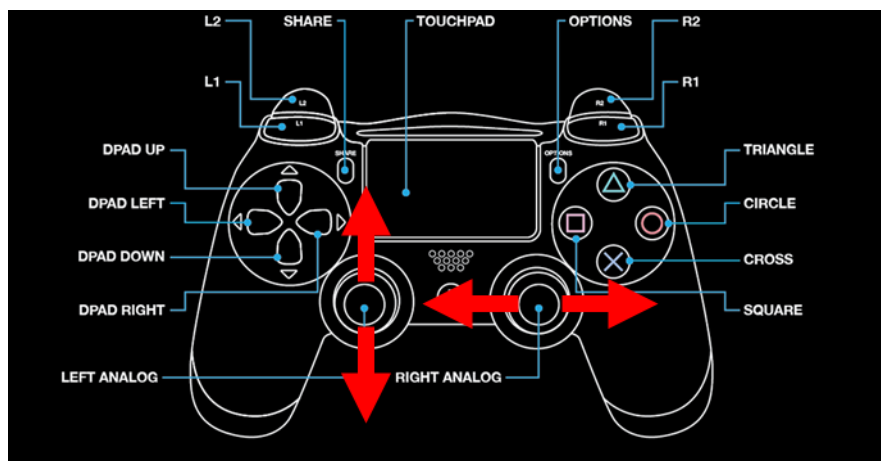


Figure 7.1.2 PS4 Controller Used Axis

```

if joystick.get_axis(1) < -0.1:    #move froward
    power = (-joystick.get_axis(1))*70
    fwd(power)

elif joystick.get_axis(1) > 0.1:    #move backward
    power = (joystick.get_axis(1))*70
    rev(power)
else:
    default()

```

Figure 7.1.3 Code for RC Car Move Forward and Backward

The axis for left analog up and down was axis 1 and the values was between -1 to 1. Therefore, if the value get from axis 1 is less than 0. This formula was used to move the RC car forward.

$$power = (-joystick.get_axis(1))*70$$

The value get from axis 1 was converted to positive and multiply with 70 which was the maximum speed. The value 70 can change to other value from 0 to 100 to set the maximum speed of the RC car.

For moving backward, just check the value get from axis 1 whether greater than 0 and use back the same formula without the negative sign. If the axis was not pushed, the RC car will be in initial state (does not move).

```

position = ((joystick.get_axis(2))*300+MID)
if position < MIN:
    position = MIN
elif position > MAX:
    position = MAX
if MIN <= position <= MAX:
    pi.set_servo_pulsewidth(SERVO_GPIO, position)

```

Figure 7.1.4 Code for RC Car Turn Left and Right

The axis for right analog left and right was axis 2 and the values was between -1 to 1. Before that the servo motor been tested in hardware development stage. The values for the RC car to turn maximum right angle was 1900, turn maximum left angle was 1350 and point to middle was 1595. Based on these value, a formula was generated for turning the RC car.

$$position = ((joystick.get_axis(2))*300+MID)$$

The formula will get maximum of $300 + 1595$ or minimum of $-300 + 1595$. Then the value was used to check if less than the minimum angle or exceeded the maximum angle. If less then it will set as the minimum angle or if exceed it will set as maximum angle. Lastly, pass the value to the `pi.set_servo_pulsewidth()` function to tune the servo motor.

7.2 Android Application + Manual Mode

For controlling the RC car with Android application, the Raspberry Pi 3 receive the commands from Android smartphone in string form. Upon detecting seek bar been pushed or button pressed on the smartphone, the Android application will send the command to Raspberry Pi 3 through Wi-Fi. Once the Raspberry Pi 3 received the command, it will check the if else cases and run the specific part of the codes. Table 7.2.1 shows all the commands for different purpose.

Command	Command String
Move forward with half speed	halfFwd
Move forward with full speed	fullFwd
Move backward with half speed	halfRev
Move backward with full speed	fullRev
Stop the RC car	stop
Turn to middle	middle
Turn right with half of maximum angle	halfRight
Turn right with maximum angle	fullRight
Turn left with half of maximum angle	halfLeft
Turn left with maximum angle	fullLeft
Stop the RC car and set front wheels to middle	back

Table 7.2.1 Command Table

Every time the seek bar been pushed, the Android application will send the command to Raspberry Pi 3. If the seek bar been pushed and hold, it won't send the command to Raspberry Pi 3 until it detected the seek bar values been changed. This will reduce the overflow command send to Raspberry Pi 3.

7.3 Ultrasonic Sensor + Autonomous Mode

In this project, the ultrasonic sensor was used to detect the obstacle. In order to move the RC car on the track and detect obstacle, the ultrasonic sensor been combined with lane tracking algorithm. Both of the ultrasonic sensor and the lane tracking codes was executed together. Previously, both of the codes was run in the same thread which causes the lane tracking code does not perform well. The ultrasonic sensor have to detect the obstacle which used more time to execute the code which make the lane tracking code run slower and the RC car went out of the track. For better result, both of these codes was executed at different threads, so it can run simultaneously without slowing down each other. When the ultrasonic sensor detected an obstacle it will stop the lane tracking code and move to right lane and the lane tracking code will be started again. After overtake the obstacle, the RC car will move back to the left lane and continue to complete the track.

7.4 Overall System

The Android application have to work together with Raspberry Pi 3. The Python code was started first at Raspberry Pi 3 to receive the command from Android application.

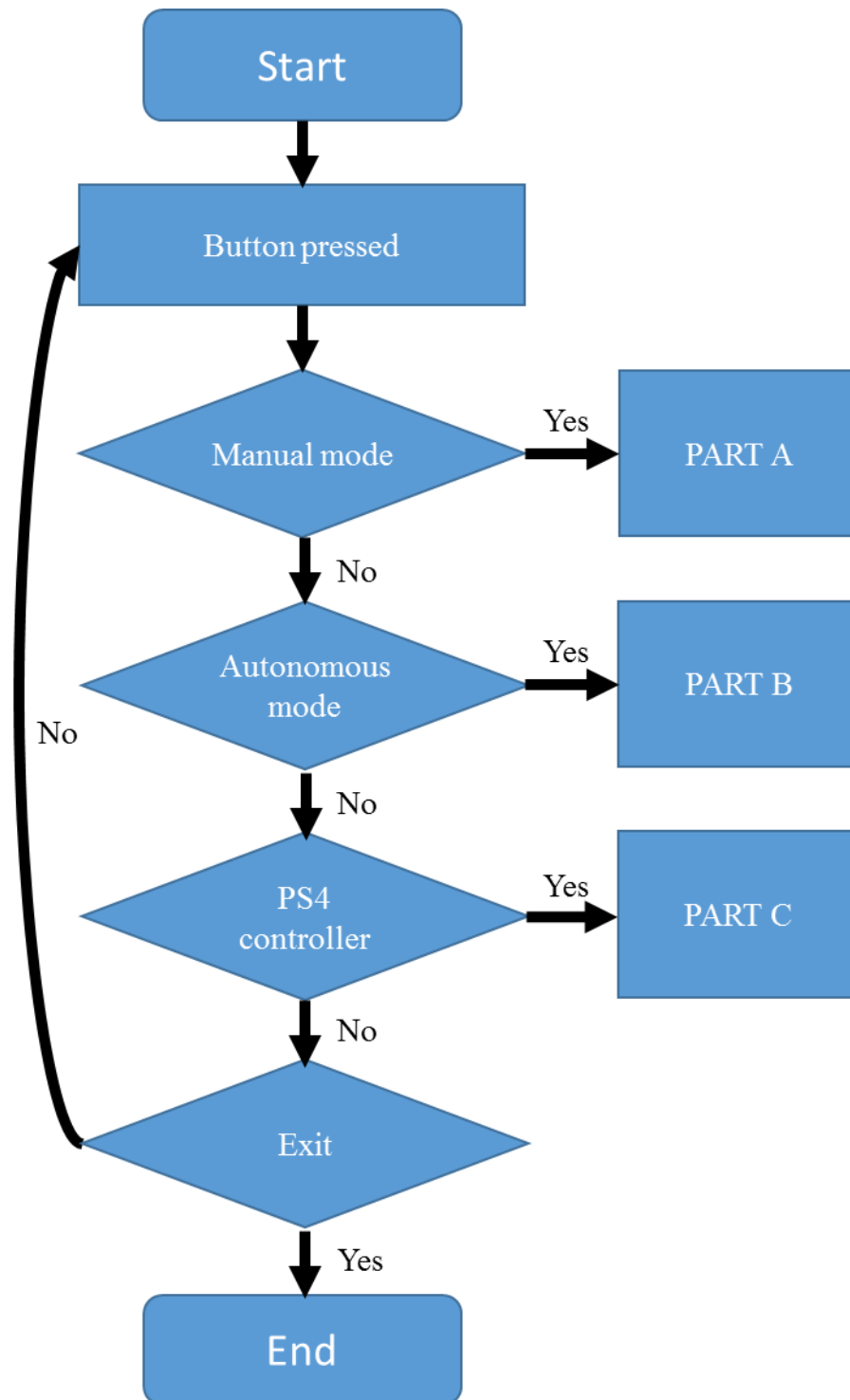


Figure 7.4.1 Flowchart of System Part 1

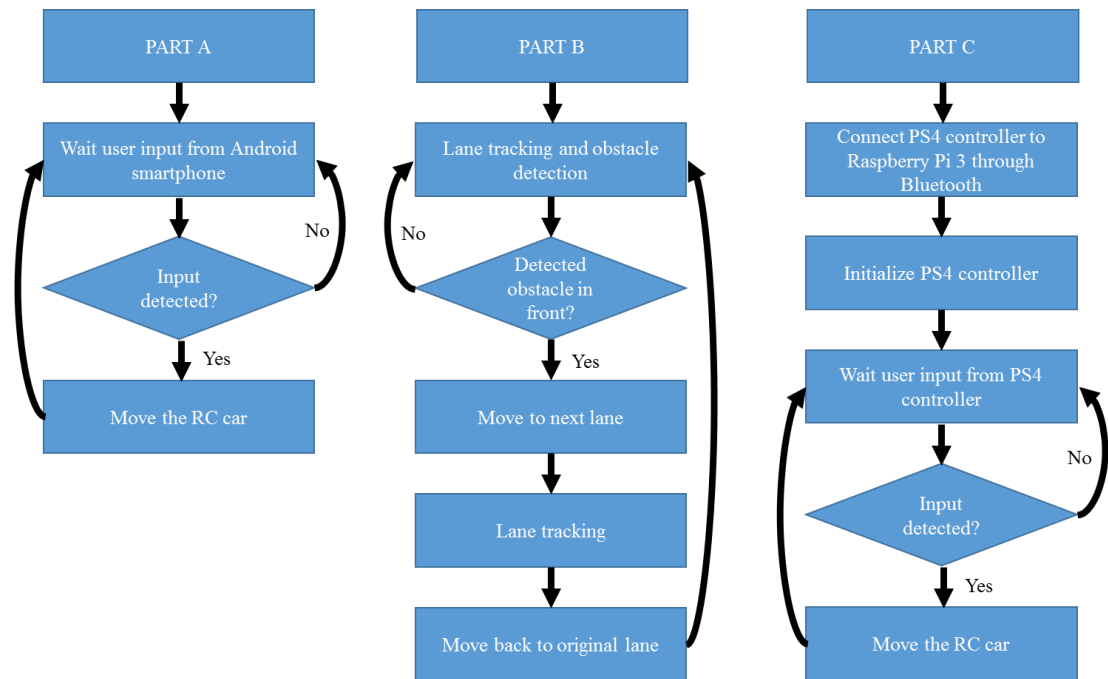


Figure 7.4.2 Flowchart of System Part 2

When the Android application is started, there will be MANUAL, AUTONOMOUS, PS4 CONTROLLER and EXIT button. When MANUAL button is pressed, two seek bars are located at bottom left and right of the interface which used to control the movement of the RC car. When AUTONOMOUS button is pressed, the RC car started to move on the track using the lane tracking algorithm and detect obstacle in the front of the RC car using ultrasonic sensor. When detected obstacle, the RC car will move to next lane and continue detect the lane of the track. The RC car will move back to original lane after pass by the obstacle and the step will be repeated again. When PS4 CONTROLLER pressed, user needed to connect the PS4 controller to Raspberry Pi 3 through Bluetooth. After successfully paired, the PS4 controller is initialized. Then user can use the controller to control the RC car manually. Finally when the EXIT button is pressed, the Android application is terminated.

Chapter 8: Testing and Result**8.1 Manual Mode****8.1.1 PS4 Controller with Manual Mode**

In order to ensure the PS4 controller perform well in manual mode, several test case has been define and following table shows the result of each test case.

No	Test case	Result and discussion
1	Press the share and PS button	Put the controller into pairing mode.
1	Push left analog of PS4 controller to up	RC car move forward with the speed control by amount of the analog been pushed up.
2	Push left analog of PS4 controller to down	RC car move backward with the speed control by amount of the analog been pushed down.
3	Push right analog of PS4 controller to left	RC car turn left with the angle control by amount of the analog been pushed left.
4	Push right analog of PS4 controller to right	RC car turn right with the angle control by amount of the analog been pushed right.
5	Push left analog of PS4 controller to up and push right analog of PS4 controller to right	RC car move forward in a circle (clockwise direction)
6	Push left analog of PS4 controller to down and push right analog of PS4 controller to right	RC car move backward in a circle (anticlockwise direction)

Table 8.1.1 PS4 Controller Manual mode Test Case

8.1.2 Android Application with Manual Mode

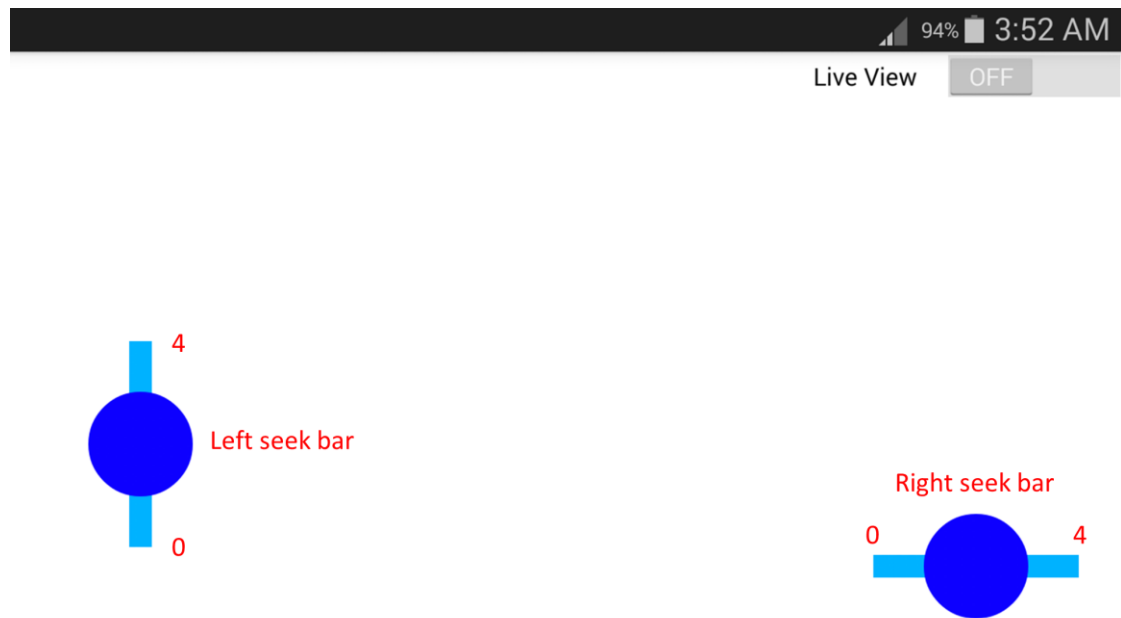


Figure 8.1.1 Android Application Manual Mode

No	Test case	Result and discussion
1	Press the live view switch	<ul style="list-style-type: none"> - If the live view switch before press is off, then live view is turn on and get the live stream from webcam. - If the live view switch before press is on, then live view is turn off and stop the live stream from webcam.
2	Push the left seek bar up which is 3	RC car move forward with 50% speed.
3	Push the left seek bar up which is 4	RC car move forward with 100% speed.
4	Push the left seek bar down which is 0	RC car move backward with 100% speed.

5	Push the left seek bar down which is 1	RC car move backward with 50% speed.
6	Push the left seek bar middle which is 2	RC car does not move.
7	Push the right seek bar left which is 0	RC car turn left with maximum angle.
8	Push the right seek bar left which is 1	RC car turn left with 50% angle to maximum angle.
9	Push the right seek bar right which is 4	RC car turn right with maximum angle.
10	Push the right seek bar right which is 3	RC car turn right with 50% angle to maximum angle.
11	Push the right seek bar middle which is 2	RC car point to middle (no turning)

Table 8.1.2 Android Application Manual mode Test Case

8.2 Autonomous Mode

In this section, the autonomous mode of the RC car will be discussed.

Test case 1

The RC car been placed on the middle of the left lane of the track.

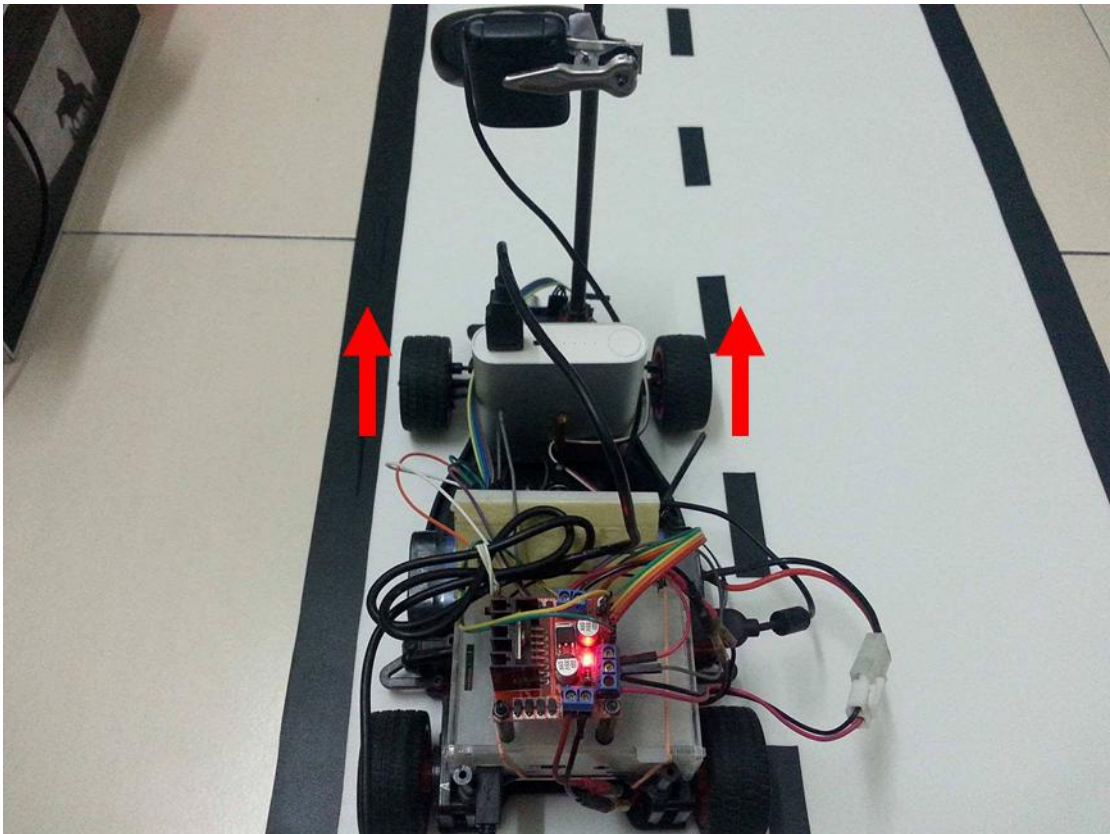


Figure 8.2.1 Test Case 1

The diagram showed that the front wheels of the RC car was parallel with the track when it detected a straight lane.

Test case 2

The RC car been placed on the left lane, left line of the track.

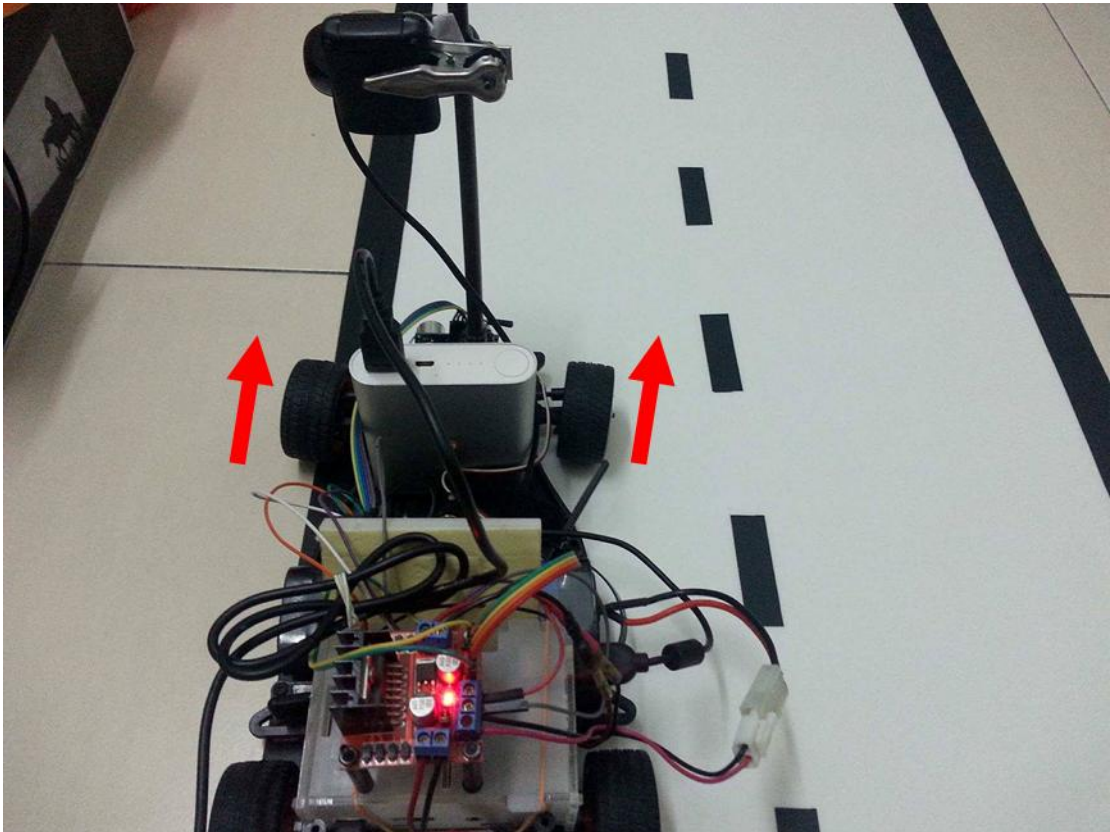


Figure 8.2.2 Test Case 2

The diagram showed that the RC car tried to move back to the center of the left lane by turning the front wheels to right. The angle of turning depend on the distance between the RC car and the center of the left lane.

Test case 3

The RC car been placed on the left lane, right line (dotted line) of the track.

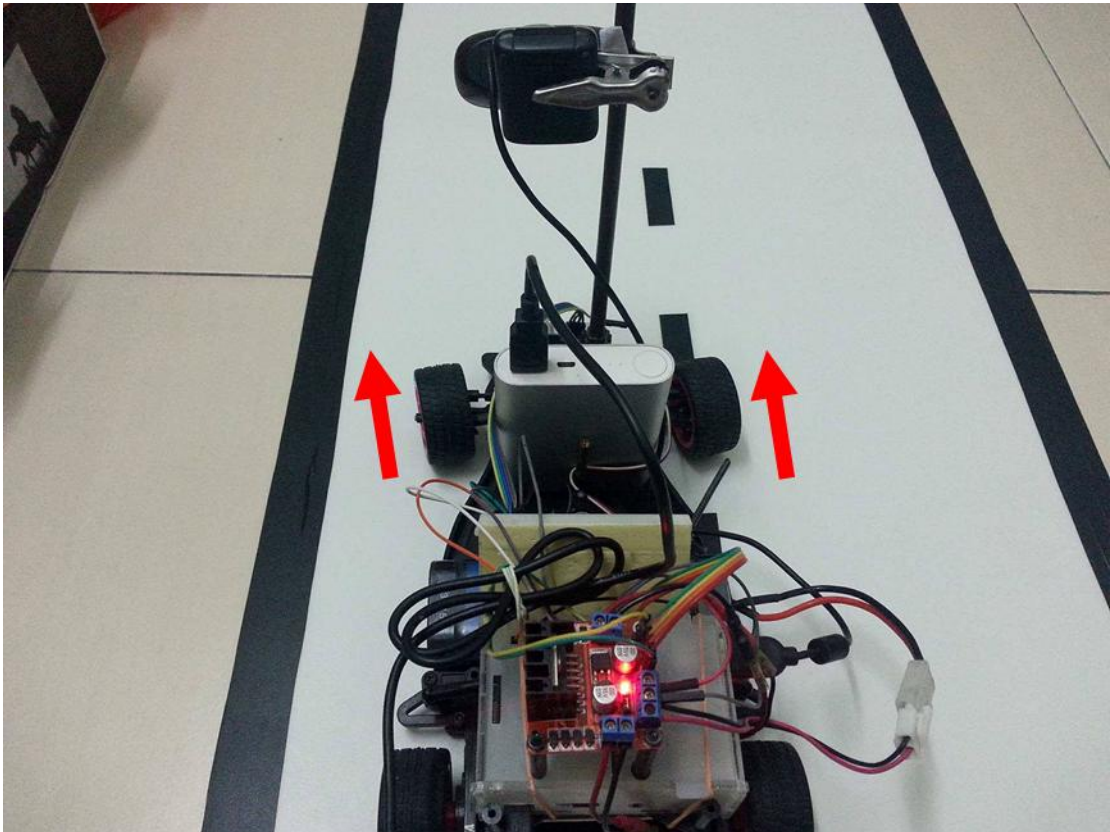


Figure 8.2.3 Test Case 3

The diagram showed that the RC car tried to move back to the center of the left lane by turning the front wheels to left. The angle of turning depend on the distance between the RC car and the center of the left lane.

Test case 4

The RC car been placed on the left lane, curve lane of the track.

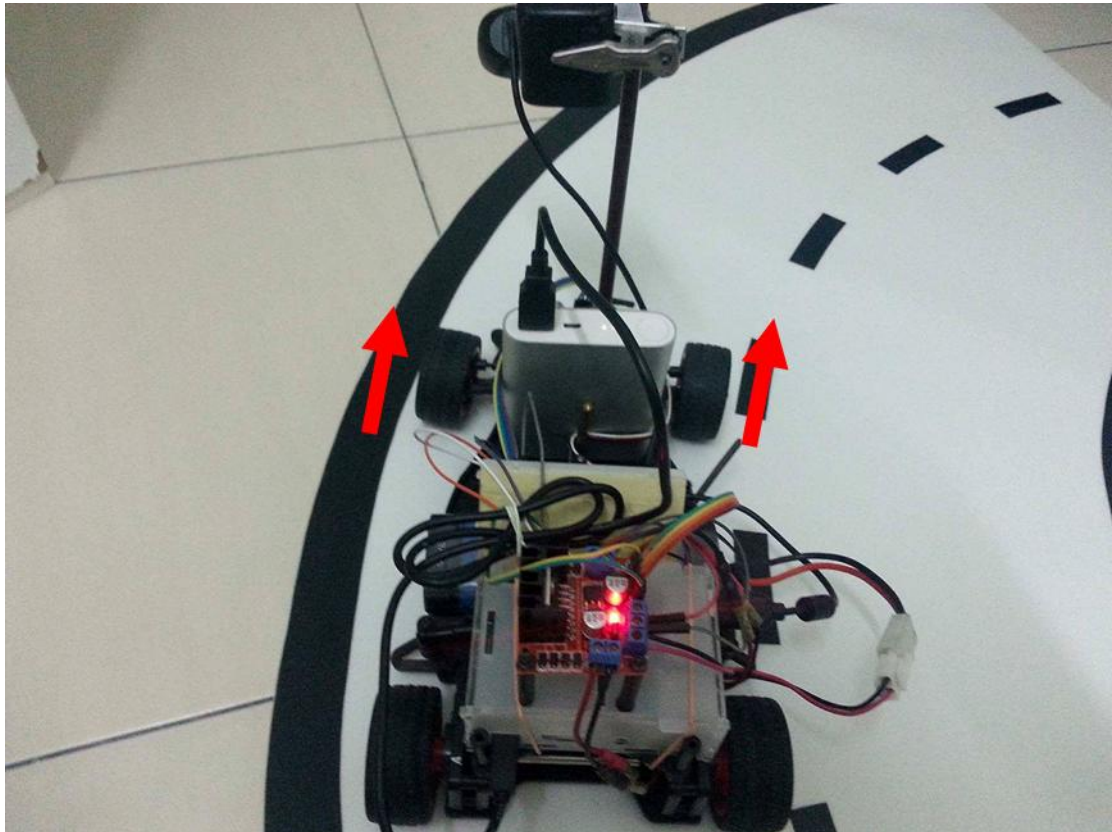


Figure 8.2.4 Test case 4

The diagram showed that the RC car turned to right when it detected a curve lane. The angle of turning depend on the curvature of the lane.

Test case 5

An obstacle was placed on the left lane of the track. The RC car been placed behind the obstacle.



Figure 8.2.5 Test Case 5

The diagram showed that when the ultrasonic detected the obstacle in the range of 30cm, the RC car will turn right and move to next right lane to overtake the obstacle.

Test case 6

After overtake the obstacle situation.

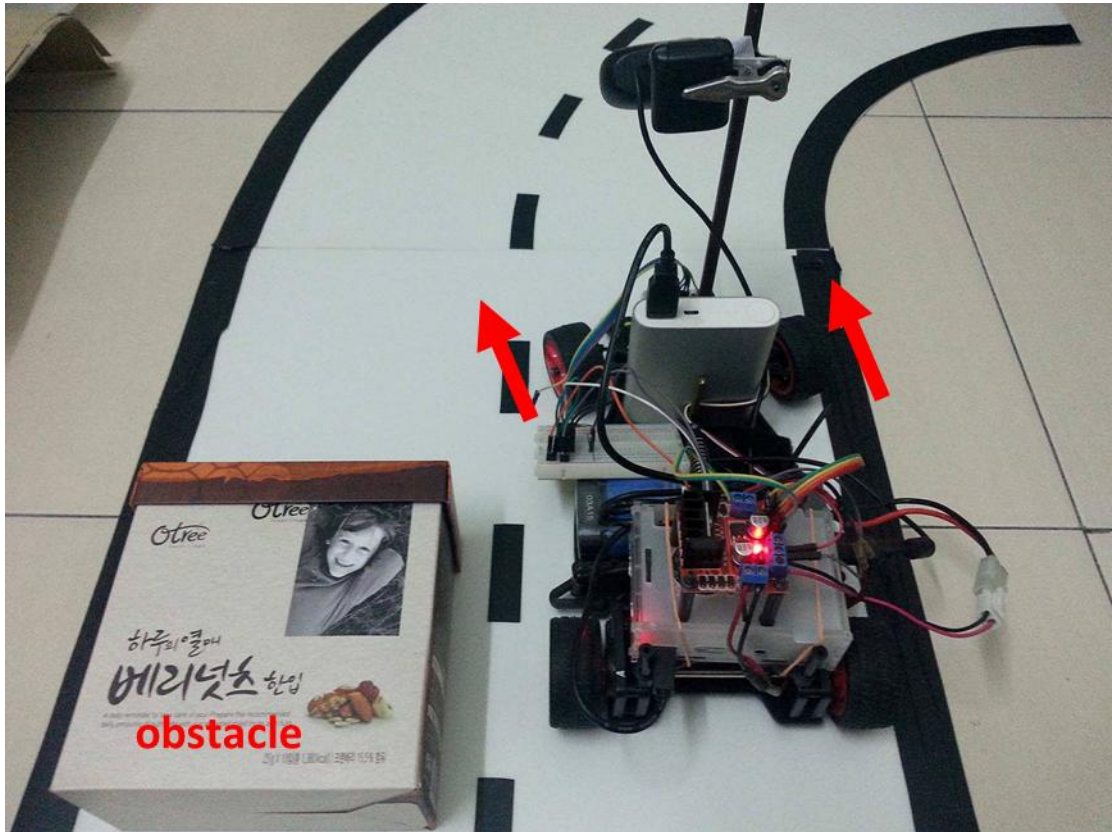


Figure 8.2.6 Test Case 6

The diagram showed that the RC car move back to the left lane after overtake the obstacle.

Chapter 9: Conclusion

In FYP 1, all of the hardware components was mounted on the RC car and tested with some basics coding to ensure that all of the hardware components are working properly. Some of the objective has been achieved which are low power consumption prototype, low cost prototype, created an version 1 Android application to control RC car in manual mode which control by Android smartphone and PS4 controller via Bluetooth. The Android application able to communicate with Raspberry Pi 3. For the objective RC car navigate itself through the track and obstacles avoidance using computer vision technique will be focus in FYP 2. There will be extra features added in FYP 2 and the Android application will be updated.

In FYP 2, the objective which RC car is able to navigate itself through the track and overtake any obstacles by computer vision technique is achieved but the algorithm for the lane tracking still needs further refinement as test result shows it is still far from perfect. The lane tracking algorithm and ultrasonic sensor are working in the same core which causes the RC car move much slower and sometime it may cause the lane tracking algorithm stop working. However, the RC car manage to complete the track with the lane tracking algorithm. For letting the RC car complete the track more smooth and faster, a better lane tracking algorithm is needed and a more powerful processor need to be used for image processing. As Raspberry Pi 3 processing power is not powerful and the program was execute in one core.

Since this is a prototype, the RC car used is not perfect. The RC car components Raspberry Pi 3 and Motor driver L298N may overheat if operate for a long time. Because in autonomous mode, the Raspberry Pi need to keep process the image capture from the webcam. Besides that, the autonomous mode not able to operate at a dark surrounding or there are many things beside the track. It will disturb the lane tracking algorithm.

The Android application has been improved with a better interface and able to detect the mac address of the Raspberry Pi for validation purpose. Wi-Fi connection is much better than Bluetooth, as Wi-Fi connection is more stable and the transmission range is longer and more features can be added like live view of the webcam from mobile device.

Bibliography

1. Alex (2013) How to use soft PWM in RPi.GPIO 0.5.2a pt 2 – led dimming and motor speed control. RasPi.TV [online]. Available from:
<http://raspi.tv/2013/how-to-use-soft-pwm-in-rpi-gpio-pt-2-led-dimming-and-motor-speed-control>
2. Android Studio (no date). Available from:
<https://developer.android.com/studio/index.html#features>
3. Ashutosh (2016) Simple Pi Robot. hackster.io [online]. Available from:
https://www.hackster.io/Sam_ashu/simple-pi-robot-8270b5
4. Cerin (2016) How to unpair bluetooth device from the command line. Available from:
<https://askubuntu.com/questions/758586/how-to-unpair-bluetooth-device-from-the-command-line>
5. Futaba S3010 Standard High-Torque BB Servo (no date). Available from:
<http://www.gpdealera.com/cgi-bin/wgainf100p.pgm?I=FUTM0043>
6. HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi (2014). Available from:
<https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi>
7. Jacob Salmela (2014) Raspberry Pi Webcam Over the Internet Using MJPG-Streamer. Available from:
<https://jacobsalmela.com/2014/05/31/raspberry-pi-webcam-using-mjpg-streamer-over-internet/>
8. Logitech HD Webcam C270 Technical Specifications (no date). Available from:
http://support.logitech.com/en_us/article/17556
9. Meet Android Studio (no date). Available from:
<https://developer.android.com/studio/intro/index.html>
10. Michal Ruzicka, Petr Masek. Real Time Visual Marker Detector and Tracker Based on Computer Vision for Semi-autonomous Convoy Purpose.

Bibliography

11. More on Python (no date). Available from:
<https://www.raspberrypi.org/documentation/usage/python/more.md>
12. Parth Verma (2012) The Google Autonomous Car. Available from:
<https://sites.google.com/site/parthvermapaper/home/the-google-car>
13. Putov Viktor Vladimirovich, Putov Anton Viktorovich, Ignatiev Konstantin Vasil'evich. Autonomous Three-Wheeled Robot with Computer Vision System. pp 262
14. Raspberry Pi. Available from:
https://en.wikipedia.org/wiki/Raspberry_Pi
15. RetroPie-Setup (no date). GitHub [online]. Available from:
<https://github.com/retropie/retropie-setup/wiki/PS4-Controller>
16. Roboberry (2017) Raspberry PI 3 Auto WiFi Hotspot if no Internet.
Available from:
<http://www.raspberrypi.com/network/item/320-rpi3-auto-wifi-hotspot-if-no-internet-olddscript>
17. What is Embedded Vision? (no date). Available from:
<https://www.embedded-vision.com/what-is-embedded-vision>

