الجمهورية الجزائرية الديمقراطية الشعبية
*People's Democratic Republic of Algeria*
وزارة التعليم العالي و التعليم العلمي
*Ministry of Higher Education and Scientific Research*

*The university of Biskra - Mohamed Khider*
*Faculty of Science and Technology*
*Department of Electrical Engineering*
*Division: Automatic*

*Option: Industrial Systems Engineering*

**Ref: ……**

**Final Year Project Report in view of obtaining**

**The Diploma of:**

# MASTER

# *Theme:*

## *Experimental mobile robot controlled by raspberry pi*

**Presented by:**

**GUESBAYA MOUNIR**
**June 06, 2017**

**The jury:**

| | | |
|---|---|---|
| **Mr. BENELMIR OKBA** | **M.C.B** | **Supervisor** |
| **Mrs. TERKI NADJIBA** | **M.C.A** | **President** |
| **Mr. BOUMAHREZ MOHAMED** | **Pr** | **Examiner** |

**Academic Year: 2016/2017**

الجمهورية الجزائرية الديمقراطية الشعبية
*People's Democratic Republic of Algeria*
وزارة التعليم العالي و التعليم العلمي
*Ministry of Higher Education and Scientific Research*

*The university of Biskra - Mohamed Khider*
*Faculty of Science and Technology*
*Department of Electrical Engineering*
*Division: Automatic*

***Option: Industrial Systems Engineering***

**Final Year Project Report in view of obtaining**

**The Diploma of:**

# MASTER

# *Theme:*

# *Experimental mobile robot controlled by raspberry pi*

**Presented by:** **Favorable opinion of the supervisor:**

**GUESBAYA MOUNIR** **BENELMIR OKBA** **Signature**

.

**Favorable opinion of the President of Jury:**

**TERKI NADJIBA** **Signature**

.

**Stamp and signature**

.

**الجمهورية الجزائرية الديمقراطية الشعبية**
***People's Democratic Republic of Algeria***
**وزارة التعليم العالي و التعليم العلمي**
***Ministry of Higher Education and Scientific Research***

***The university of Biskra - Mohamed Khider***
***Faculty of Science and Technology***
***Department of Electrical Engineering***
***Division: Automatic***

***Option: Industrial Systems Engineering***

# *Theme:*

## *Experimental mobile robot controlled by raspberry pi*

**Proposed and directed by: BENELMIR OKBA**

### Abstract (English and Arabic)

**المشروع هو روبوت متنقل على ثلاثة عجلات ذاتي و يدوي التحكم في آن واحد و ذلك حسب اختيارنا لطريقة تحكمنا به.**
**الهدف من هذا المشروع هو بناء آلي متنقل مخبري مصمم للأعمال المخبرية, ما يعني وجود إمكانية لتغيير طبيعة عمله (إعادة برمجته) أو زيادة طرق التحكم به سواء كانت ذاتية أو يدوية التحكم. يتجسد عقل هذا الروبوت في البطاقة البرمجية المتطورة راسبيري باي والتي يمكن اعتبارها كحاسوب صغير جدا حيث تتحكم في جميع أجهزة الاستشعار و الحركة بالتالي تتحكم في الروبوت كليا, تم استخدام أربع أنواع من الحساسات, حساس استشعار الأشعة فوق الصوتية و حساس تابع للخط الأسود للتحكم التلقائي, أما بالنسبة للتحكم اليدوي, تم استخدام مستقبل الأشعة فوق الحمراء و مستقبل الواي فاي و قد تمت عملية البرمجة باستخدام لغة بايثون الشهيرة.**
**الكلمات المفتاحية : روبوت, متنقل, روبوت بالعجلات, راسبيري باي, بايثون, التحكم عن طريق, حساس.**

**The project is a mobile robot with Manual and automatic control at the same time, according to the choice of the control mode.**
**The goal of this project is to design an experimental mobile robot for laboratory works, which means having the possibility to change the nature of his work by reprogramming or increase its control modes, whether automatic or manual control. The mind of this robot is the developed programming card Raspberry Pi that can be considered a very small computer which controls all sensors and actuators, consequently controlling the robot completely. Four sensor types that have used, an ultrasonic sensor and line follower sensor were used for the manual control, and the infrared receiver and the Wi-Fi receiver were used for the manual control, and the programming was done using the famous Python language.**

**Keywords: robot, mobile, wheeled robot, raspberry pi, python, controlled by, sensor.**

# *Dedication*

*I dedicate this work to*
*My precious family: My dear parents, my brothers' idris and seif eddine, my sisters' zohra and Sara and I ask God to protect them.*

*My esteemed teachers and everyone who I have known them in university in the last five years*

*To everyone who appreciates science and works for better*

# *Thanks*

*Above all, I thank in the first place the Almighty God for having given me courage, volition and strength to complete this work, glory and greatness for him.*

*I thank my parents a lot, my mother for the emotional support and the excessive keenness, my father for the great help materially and morally*

*I thank the honorable juries, Mr Boumahrez Mohamed and Mrs Terki Nadjiba for the help, the confidence and for giving their time for discussing my thesis*

*I thank my supervisor Benelmir Okba for his confidence, guidance and patience.*

*I thank Guesbaya Nadjib, Youcef Imad, Gheddab Abd El-ghani, Zouaoui Mohamed, Fodil Abd Assamad, Telli Islem and Abouba Kamel for their help and guidance*

*I thank all my Class friends for all good times and the unforgettable memories.*

# *Summary:*



## *Chapter 01: Generalities about robotics*

## *Chapter 02: Mobile robot hardware*

## *Chapter 03: Configuring and Programming*

## *Chapter 04: Application and Discussion*

## *List of figures:*

## *Chapter 01: Generalities about robotics*

## *Chapter 02: Mobile robot hardware*

## *Chapter 03: Configuring and Programming*



## *Chapter 04: Application and Discussion*

## *List of tables:*

## *Chapter 02: Mobile robot hardware*

## *List of Abbreviations:*

1- LED - Light-emitting diode
2- PWM - Pulse width modulation
3- VCC - voltage at the common collector
4- IEEE - Institute of Electrical and Electronics Engineers
5- Wi-Fi - Wireless Fidelity
6- HTML - Hypertext Markup Language
7- HTTP - HyperText Transfer Protocol
8- WSGI - Web Server Gateway Interface
9- RPM - Round per minute
10- AC - Alternating current
11- DC - Direct current
12- CPU - Central processing unit
13- RAM - random-access memory
14- Web APP - Web application

# General introduction

This memory provides an overview on the general principles of robotics, especially mobile robots and presents the main challenges. Robots are mainly used in the industrial environments more than the others. However, science development has brought us to this moment where we can see robotics especially mobile robots in a lot of other domains (medicine, space, exploration, military…).

The autonomy nature of mobile robot has made an influential assistance in human life, where it performs many tasks especially dangerous and sensitive ones (exploration, space, military, civil protection ...).

In this project, we will design a control system for mobile robot with two wheels, which includes four control modes that are characterized by private means (sensors, actuators, programs). These control modes can be automatic or manual since we have two manual control modes which are the Infrared control mode and the Web Application (Wi-Fi) control mode, and two autonomous control modes that are the Obstacles Avoider mode and the Line Follower mode, finally all these modes will work together in a multiple control mode.

This project aims at building an experimental mobile robot that is easy for exploitation by students at laboratory works. Therefore, it would help students transform their theoretical knowledge into a practical work.

The protocol of this project is based on laboratory sessions, which will be a suitable way to help students link between theoretical and practical work. The construct of this robot enables them to see the relation between hardware and software, learn about electronic concepts, choose the right parts, do the right wiring up process and divide the power from the alimentation source, without any risks, and a lot of other benefits.

This memory consists of four chapters, where the first chapter presents generalities about robotics, especially the wheeled mobile robot where it discusses its elements and its kinematic model, the second chapter includes detailed specifications about the hardware and software that have used to build this mobile robot, than the third chapter provides all the details about software in terms of configuration and programming, which means all the command lines, instructions and programming symbols that will be used, finally the fourth chapter where is based on five laboratory sessions which are considered as steps to connect everything together and realize the complete building of this mobile robot.

# *Chapter I: Generalities about robotics*

Robotics is a set of techniques that allows designing and producing the automated robots or machines. Robotics is one of the scientific and industrial fields where is characterized by multidisciplinary, it has many aspects: mechanical, computing, electronics...

Robotics is an infinite field in world activity where includes many other areas such as: domestic, medical, military, agricultural, exploratory, space robotics.

## I-1 Robot definition:

Robots are taking an important place in our lives, it exists in every industrial company to speed up production or to act where human can't work because of the danger.

The robot is a machine with an automatic system that can perform certain tasks instead Human, we can see this in one or more autonomous tasks , all this by sensors (ultrasonic, infrared, camera, button...) and ordered effectors by actuators (DC motor, servo motor, stepper motor, ...), we can say the robot is capable of extracting information from his environment and use this knowledge to decide how to act , this interrelationships between the machine and the environment ,it makes the machine more close to autonomy [1][2].

As we can see in the following figure:

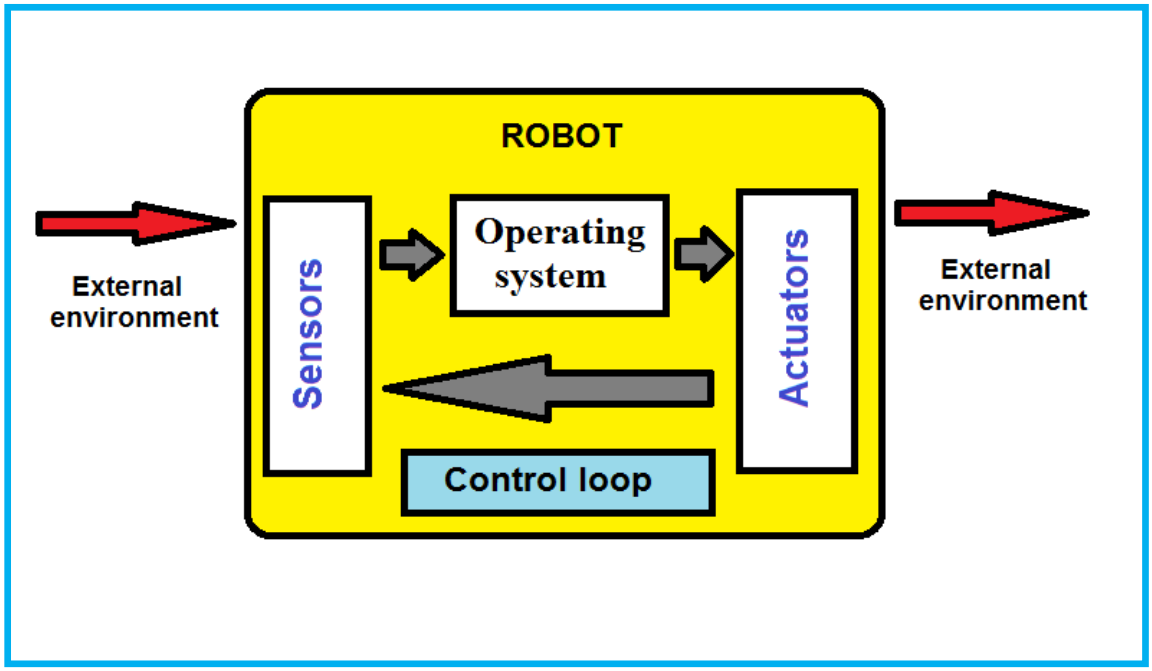



Figure I-1: Decision loop

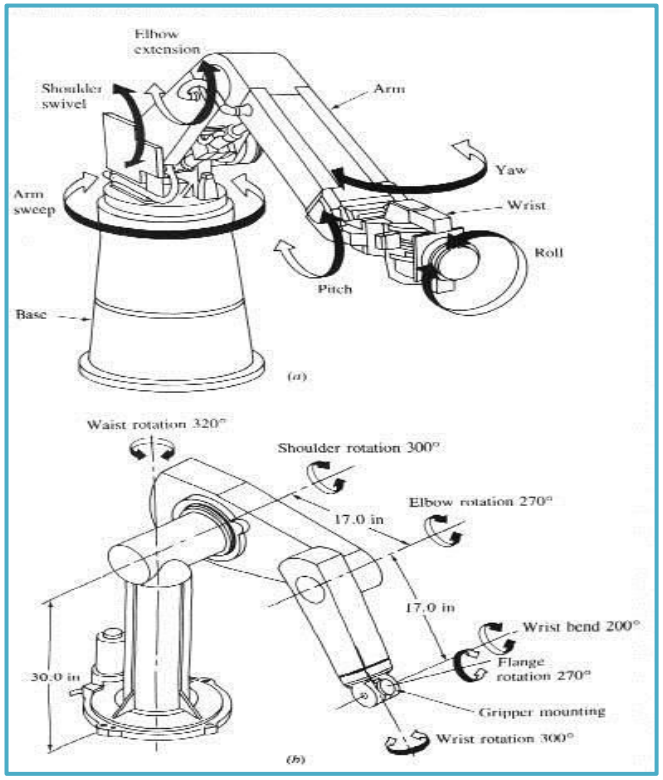
## I-2 Robots histories:

The robotic word is derived from robot, According to the Oxford English Dictionary, was used for the first time in 1921 by Karel Capek in his play R.U.R: Rossums Universal Robots [2].

The robotic word was used in print for the first time by Isaac Asimov, Russian-born American writer, in his science fiction story "Menteur!" Published in May 1941 in the history of «Astounding Science Fiction» [3].

In 1954, George Charles Devol invented the first programmable industrial robot called Unimate, was able to manipulate objects with hydraulic actuators.

Figure I-2: « Unimate » The first industrial robot




Robotics industry begins in 1960, where in 1961 the appearance of a robot on a chain for mounting is the first robot with effort control and in 1969 Victor Scheinman of Stanford University injected a 6 axes articulated arm.

In 1974 Scheinman commercialized electric robot controlled by a computer that uses pressure sensors for industrial applications.

Figure I-3: The Puma « 6 axes »

A subsequent year has seen improvements in all kinds of robots and becomes increasingly fast, precise and flexible. In 1998, for example control systems could handle up to 27 axes and synchronized control of four robots.

Between 1990 - 2000: large development of artificial intelligence and robotics, new robots appear constantly produces [4].

- 1995: Launching Robocup
- 1997: First mobile robot on Mars planetary extra « Sojourner »
- 1999: Launching of « Aibo » [5].

Figure I-4: Aibo

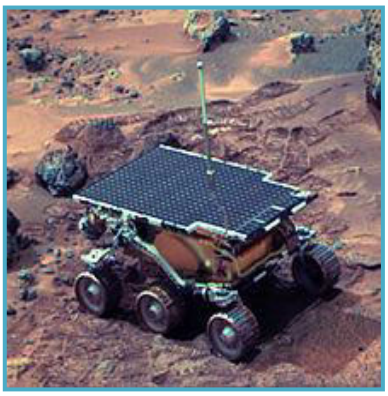


Figure I-5: Sojourner

Figure I-6: Robocup

- 2000: Launching of « Asimo »
- 2009: Robot « Nao » has used like a soccer robot

Figure I-7: Nao


Figure I-8: Asimo

## I-3 Different types of robots:

Mobile robots can be found in the manufacturing industry, military, space exploration, transportation and medical applications. Here are some mobile robots categories:

- Land Robots
- Flying Robots
- Underwater Robot
- Manipulator Robot
- Human robots [6].

## I-4 Robot elements:

There are many basic elements of the mobile robot as actuator motors which are to be controlled by the control part according to the information extracted from these sensors.

### I-4-1 Control part:

The control part of robot is the decision center. It gives orders to the operative party and receives its reports. It can consist of three parts: a computer, software and an interface.

Control system is that part of the robot that determines the robot's behavior according to its program in the software part. We have a lot of different types of electronic control units (based on microcontrollers or Processors), as we can see some examples in the following pictures:

Figure I-9-A:
Raspberry pi 2

Figure I-9-B:
Arduino Uno

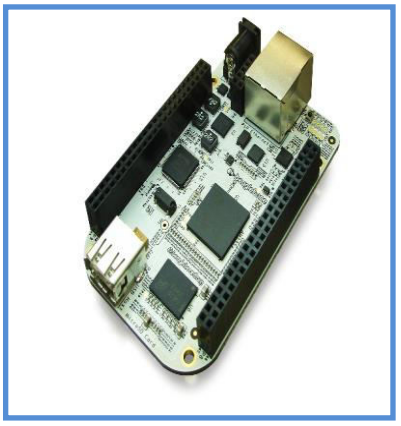


Figure I-9-C:
BeaglBone

## I-4-2 Different types of sensors:

The sensor is a sampling device information that develops from a physical quantity (incoming information) to another physical quantity of different types (outbound information: very often electric). This quantity representative of the sampled quantity, is used for purposes of measuring or controlling [7].

### I-4-2-A - Light Sensors:

Light Sensor is a device that robot can use to detect the current ambient light level, how bright or dark it is. There are a range of different types of light sensors, including Photoresistors, Photodiodes, and Phototransistors, Photographic. As follows:

Figure I-10-A: Photoresistor

Figure I-10-B: Photodiode

Figure I-10-C: Phototransistor

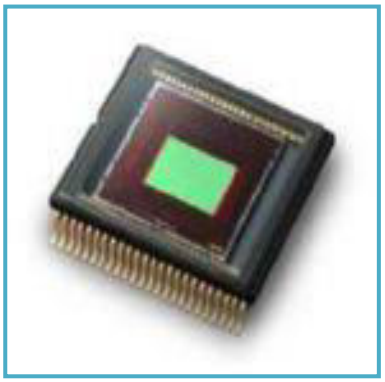


Figure I-10-D: Photographic

### I-4-2-B - Force Sensors:

Force sensors are initially rigid links between two shapes that are able to measure transmitted forces and torques. As we can see those examples of force sensors:

Figure I-11-A: Force Sensors « s »

Figure I-11-B: Pancake

Figure I-11-C: Etalon

Figure I-11-D: Miniature

### I-4-2-C - Speed Sensors:

Speed sensors are machines used to detect the speed of an object. As follows:

Figure I-12-A: Incremental encoder

Figure I-12-B: Tachometry

### I-4-2-D - Meteorological Sensors:

Meteorological sensors are used to understand and measure climate and weather. As follows:

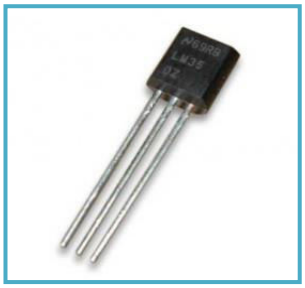


Figure I-13-A: Temperature Sensor

Figure I-13-B: Pressure Sensor

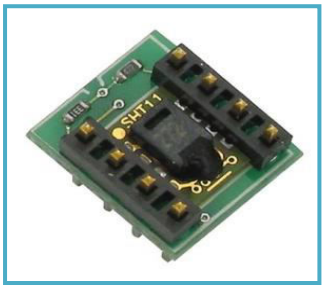


Figure I-13-C: humidity sensor

### I-4-2-E - Position sensors:

Position sensor is any device that permits position measurement. It can either be an absolute position sensor or a relative one. Position sensors can be linear, angular, or multi-axis, as we can see in the following figures:

Figure I-14-A:
Sensor hall effect

Figure I-14-B:
Sensors Capacitive

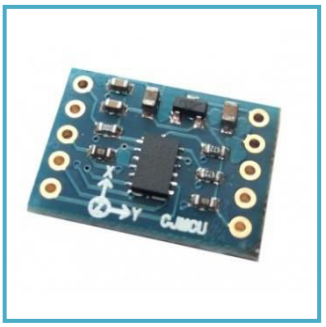


Figure I-14-C:
Accelerometer

### I-4-2-F- Voice sensors:

Sound sensors work by detecting differences in air pressure and transforming them into electrical signals. Sound sensors such as microphones usually have. As we can see in the following figure:

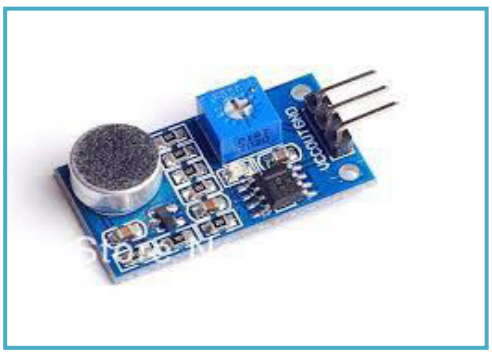


Figure I-15: LM393 Sound Detection

### I-4-3 Different types of actuators:

The actuator is the element that operates the control system to bring it from a given state to the desired state, it is an electromechanical converters designed for moving mechanical systems from electrical controls [8].

### I-4-3-A - Pneumatic actuators:

Transform pneumatic energy into mechanical energy, the use of this model are mainly for simple sequential movements. The modus operandi of this type is the use of compressed air at ~ 6 bar and allow for the cylinders whose strength can reach 50,000 N.

Pneumatic actuators are often the least expensive automation solution and performances meet the needs [9].

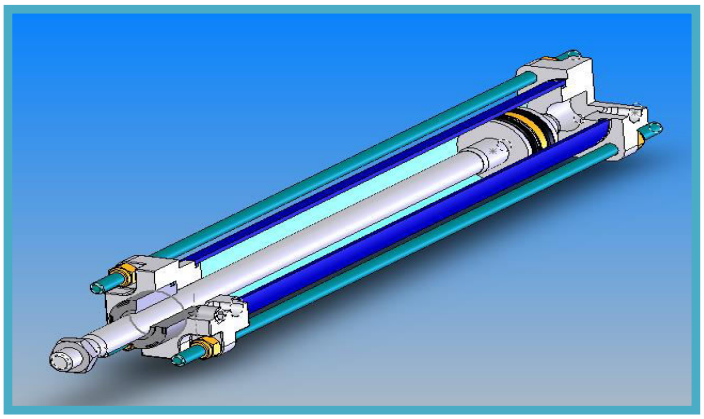


Figure I-16: Pneumatic Actuator (cylinder)

### I-4-3-B - Hydraulic actuators:

Transform the hydraulic energy into mechanical energy, such as:

- **Hydraulic cylinder:** A hydraulic cylinders is a mechanical actuator that is used to give a unidirectional force . It gets their power from pressurized hydraulic fluid , which is typically oil or water [10].
- **Pumps:** Generating a flow rate by converting mechanical energy into hydraulic energy. It is a flow generator; the principle used in hydraulic pumps is based on the volume variation between the inlet and outlet [9].

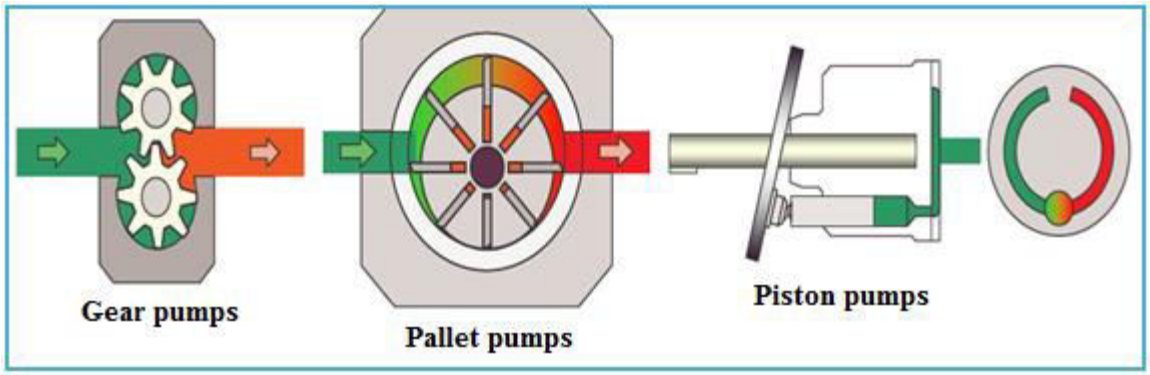



Figure I-17: Types of pumps

### I-4-3-C - Electric actuators:

Convert electrical energy (phase alternating current) to mechanical energy, we sight the generality of electric actuators in electric motors, such as:

- Servo motor
- Step-by-step engine
- DC motor (direct current) [9]

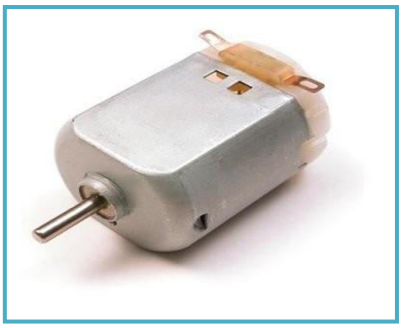


Figure I-18-A: DC motor

Figure I-18-B: Step-by-step motor

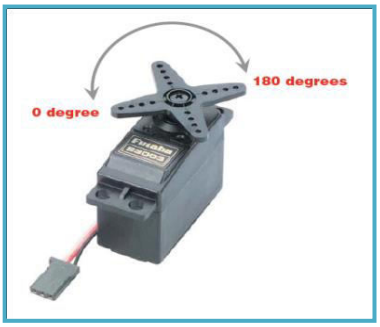



Figure I-18-C: Servo motor

## I-5 Mobile Robot:

Mobile Robotics is a multidisciplinary domain that involves many aspects such as mechanics, electronics, automation, computer science or artificial intelligence.

The mobile robot is a machine that has abilities to percept, decide and action. That allows it to act autonomously in environment according to its perception degrees [11].

There are many ways that allows mobility in a specific environment (land, air, water) and the focus will be on the land environment in this memory where there are many different types of land mobile robots which are wheeled robots, legged robots and chained robots.

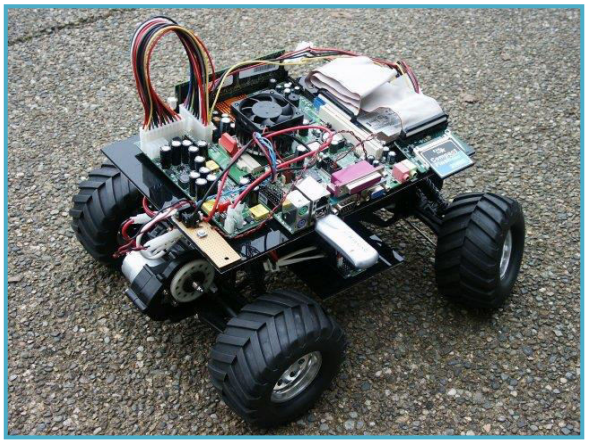


Figure I-19-A: Wheeled robot

Figure I-19-B: Legged robot

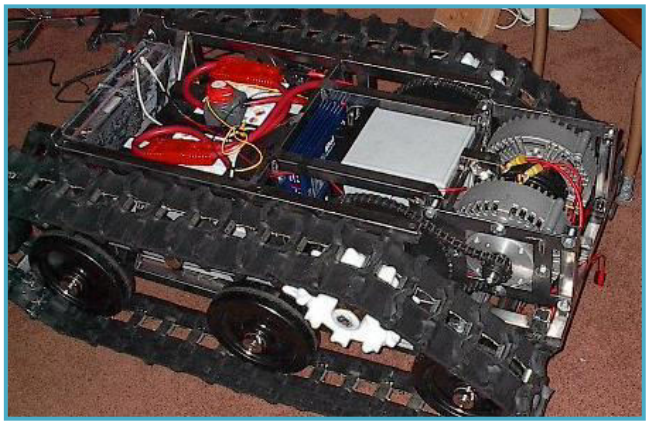
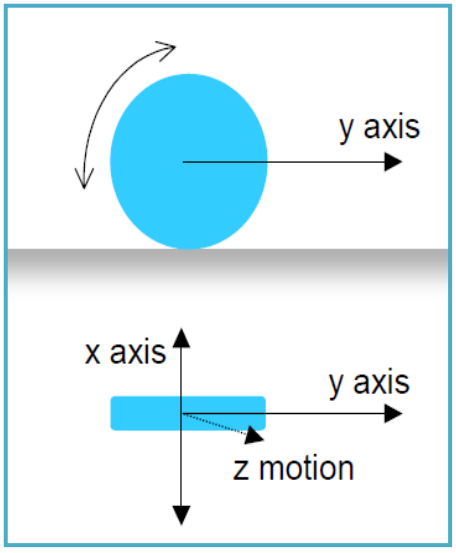


Figure I-19-C: Chained robot

## I-5-1 Wheeled Robot:




Wheeled robot is a robot that navigates around the ground using motorized wheels to propel itself. Wheeled mobile robots are widely used in many industrial and service applications, particularly when flexible motion capabilities are required on smooth grounds and surfaces.

In general, wheeled robots consume less energy and move faster than other locomotion mechanisms (legged robots or chained robots). Wheeled mobile robots are suitable for a large class of target environments in practical applications [12][13].

Figure I-20: Wheel movement

## I-5-2 Differential Drive Kinematics:

Many mobile robots use a drive mechanism known as differential drive. It consists of two drive wheels mounted on a common axis, and each wheel independently can be driven either forward or backward.

While we can vary the velocity of each wheel of the robot to perform rolling motion, the robot must rotate about a point that lies along their common left and right wheel axis. The point that the robot rotates about is known as the ICC (Instantaneous Center of Curvature).

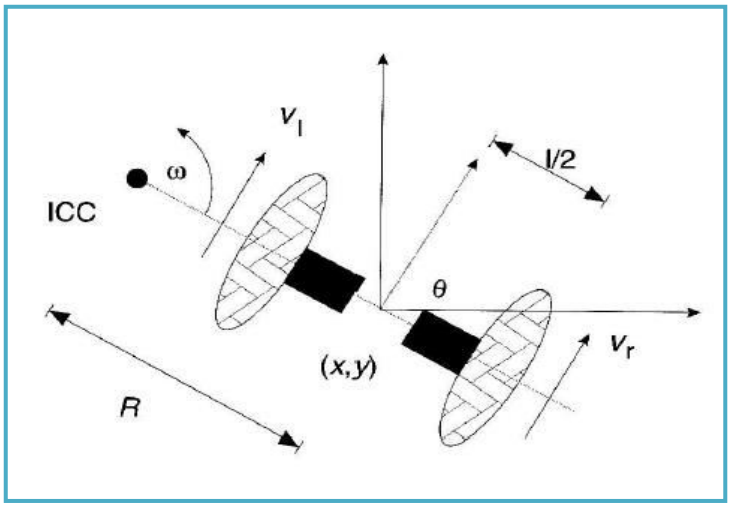



Figure I-21: Instantaneous Center of Curvature

By varying the velocities of the two wheels, we can vary the trajectories that the robot takes. Because the rate of rotation ω about the ICC must be the same for both wheels, we can write the following equations:

$$\omega\,(R + l/2) = V_r \qquad \text{(I-5-2-1)}$$

$$\omega\,(R - l/2) = V_l \qquad \text{(I-5-2-2)}$$

Where " l " is the distance between the centers of the two wheels, Vr , $V_l$ are the right and left wheel velocities along the ground , and R is the signed distance from the ICC to the midpoint between the wheels. At any instance in time, we can solve R and ω:

$$R = \frac{l}{2}\frac{V_l + V_r}{V_r - V_l};\ \omega = \frac{V_r - V_l}{l}; \qquad \text{(I-5-2-3)}$$

- There are three interesting cases with these kinds of drives.

1. If $V_l = V_r$ then we have forward linear motion in a straight line. R becomes infinite, and there is effectively no rotation that's means ω is zero.

2. If $V_l = -V_r$ then R = 0 and we have rotation about the midpoint of the wheel axis - we rotate in place.

3. If $V_l = 0$ and $V_r = X$ then we have rotation about the left wheel, in this case $R = \frac{l}{2}$ . Same is true if Vr = 0 and $V_l = X$ (rotation about the right wheel).

Differential drive vehicles are very sensitive to slight changes in velocity in each of the wheels. Small errors in the relative velocities between the wheels can affect the robot trajectory. They are also very sensitive to small variations in the ground plane, and may need extra wheels for support [14].

## I-6 Degrees of freedom:

The degree of freedom is a mechanical concept covering the possibility of movement in space, the number of degrees of freedom of a mobile robot is defined as the number of independent movement that the robot can make compared to a coordinate system determined.

We can deduce the number of degrees of freedom by the number of axes and the nature of the movement of the robot on these axes [15].

The mobile robot with two wheels has two degrees of freedom because it can move forward or backward at the level of the horizontal axis or rotate around the vertical axis. Two axes means two freedom degrees.

## ❖ Conclusion:

This chapter contains generalities about robotics and how it works generally, where robots have been developed over time and this reflected in its size differences, forms and functions according to its perceptual abilities.

It provides also an overview about mobile robots and its types especially of land-based locomotion (wheeled, legged and chained robots), finally presented the drive mechanism which is the differential drive kinematic that I used in my project.

The next chapter presents the electronic card which we can consider it as a brain and the major hardware that will be used on the wheeled mobile robot.

# *Chapter II: Mobile Robot Hardwar*

Each robot must be programmed to do what we want it; this programming is stored in the memory of processor which belongs to the electronic card which controls everything. Raspberry pi is the electronic card that will be used in this project contains ARM processor, it's a small electronic card and it will be the basis of our project.

The Raspberry Pi is a complete computer in one chip with lower performance but very small and suitable for this project where will connect all the devices to it, this means sensors and actuators which also we will talk about it in this chapter.

## II-1 Raspberry pi:

The electronic card Raspberry Pi designed to allow students and more young people to learn programming; it is also used for different projects in electronics or computer. The advantage of such a device lies precisely in the fact that the user is completely free to choose the field or the aspect that he want to use it on , Some may made a computer very mobile , others will do a web server or build a robot ..., and these are just some examples among many others [16][17].

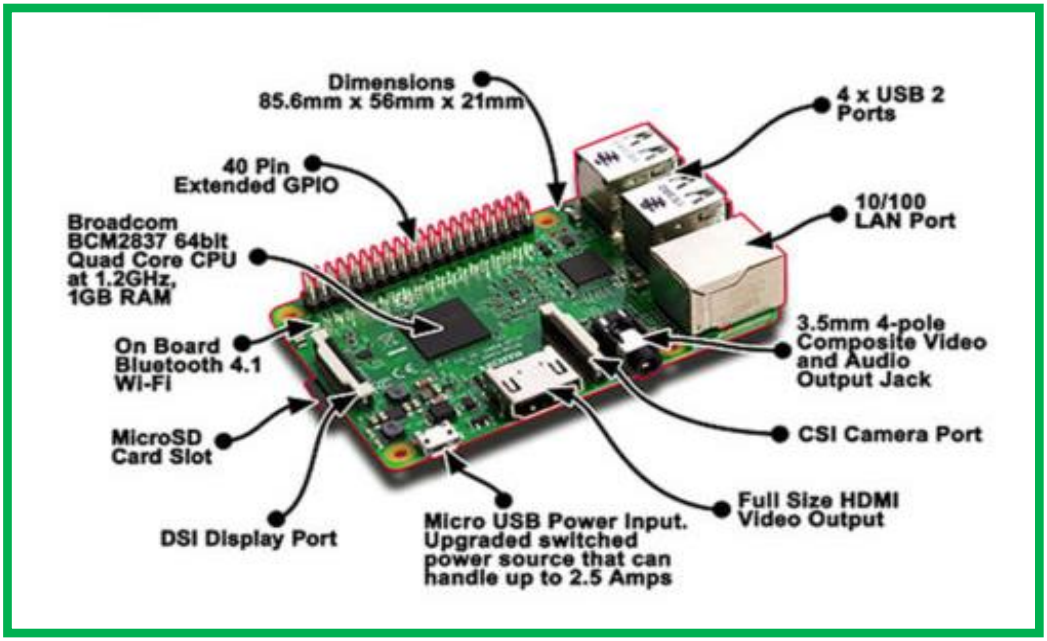



Figure II-1: Raspberry pi (model B)

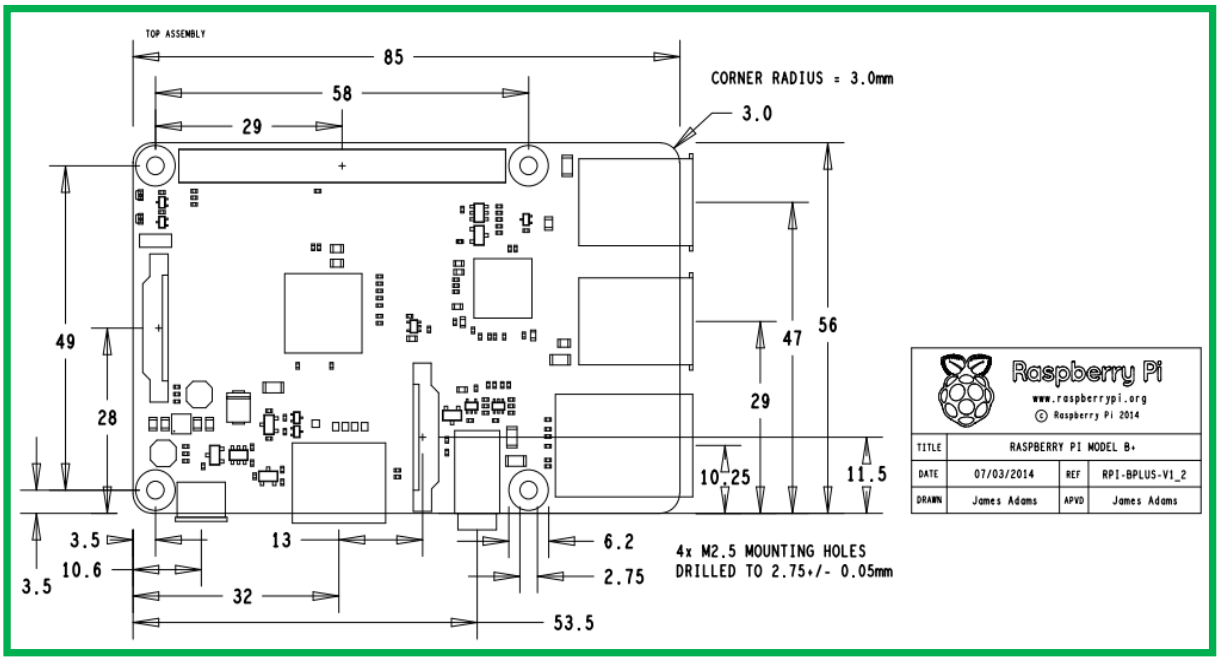



Figure II-2: Dimensions measurements of Raspberry pi 2 model B

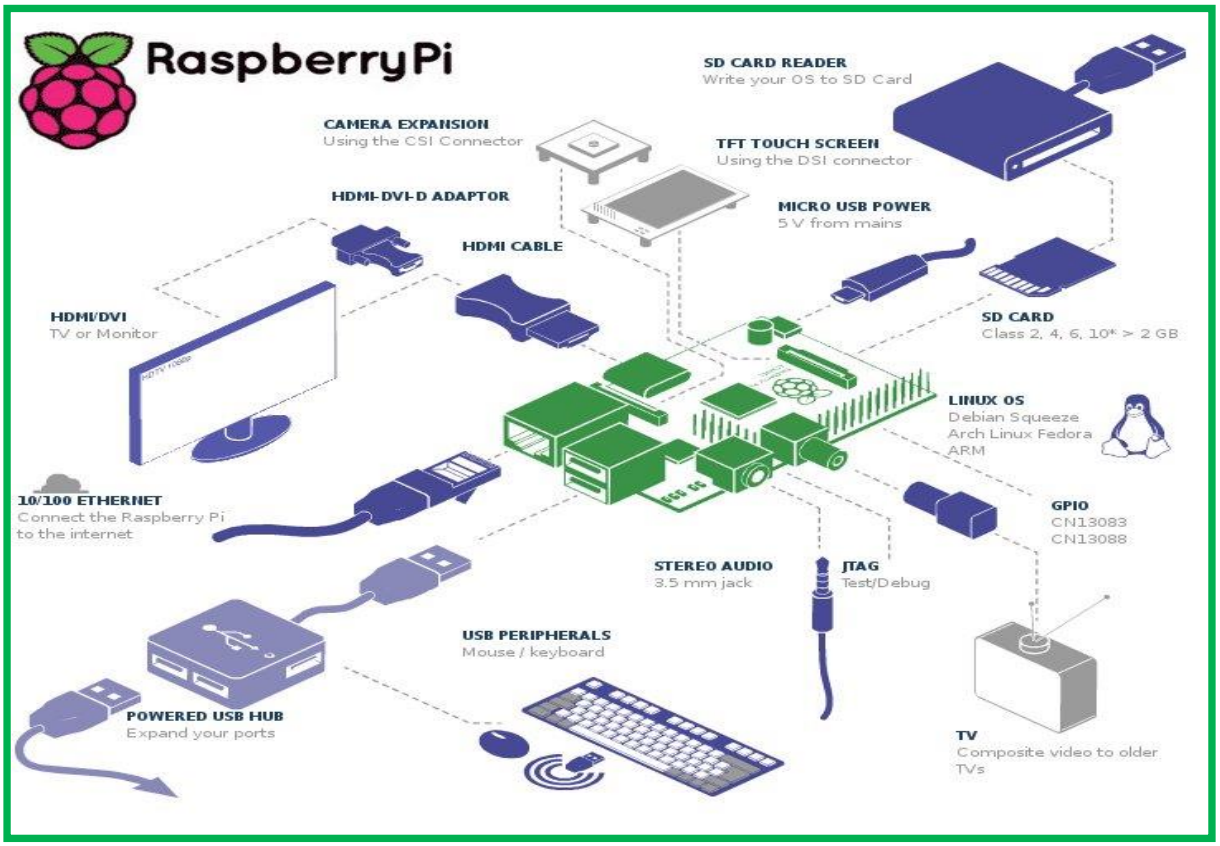



Figure II-3: Raspberry pi model B Inputs and Outputs

- We can see here some models of Raspberry Pi:

| | Raspberry Pi 3 Model B | Raspberry Pi Zero | Raspberry Pi 2 Model B | Raspberry Pi Model B+ |
|---|---|---|---|---|
| Introduction Date | 2/29/2016 | 11/25/2015 | 2/2/2015 | 7/14/2014 |
| SoC | BCM2837 | BCM2835 | BCM2836 | BCM2835 |
| CPU | Quad Cortex A53 @ 1.2GHz | ARM11 @ 1GHz | Quad Cortex A7 @ 900MHz | ARM11 @ 700MHz |
| Instruction set | ARMv8-A | ARMv6 | ARMv7-A | ARMv6 |
| GPU | 400MHz VideoCore IV | 250MHz VideoCore IV | 250MHz VideoCore IV | 250MHz VideoCore IV |
| RAM | 1GB SDRAM | 512 MB SDRAM | 1GB SDRAM | 512MB SDRAM |
| Storage | micro-SD | micro-SD | micro-SD | micro-SD |
| Ethernet | 10/100 | none | 10/100 | 10/100 |
| Wireless | 802.11n / Bluetooth 4.0 | none | none | none |
| Video Output | HDMI / Composite | HDMI / Composite | HDMI / Composite | HDMI / Composite |
| Audio Output | HDMI / Headphone | HDMI | HDMI / Headphone | HDMI / Headphone |
| GPIO | 40 | 40 | 40 | 40 |
| Price | $35 | $5 | $35 | $35 |

Table II-1: Characteristics of some Raspberry Pi models

### II-1-1 History:

Raspberry story begins in 2006 at British Cambridge University and specifically department of Computer Science, where a group of professors of computer science are discussed the problem of the educational level of students newly delegations of university the team found that the main problem in the low level new students is that they lack experience of dealing with computer components and electronic pieces on the reverse previous generations characterized by experience in the field of electronics (Wearer do not attend the faculties of computer Science only nerds of electronics).

Professor Eben Upton thought how he can helps new students to delve deeper into computer Science with an understanding of electronic components and science programming at the same time makes students able to manufacture and modify hardware and linked by computer.

University professor began studying the first problem appeared and it's about provision of computers that students can manipulate and dismantled parts and add other parts of it without fear of sabotage. Upton was assumed that fathers and mothers would not be happy that the sons sabotaged the expensive computers by making experiments and dismantled them from the inside.

Figure II-4: Prof Eben Upton

The solution came as a small computer easily manipulate and develop it at the same time is cheap so that can be available to all and It facilitates the learning process for all students without fear of damaging their expensive computers.

Eben Upton has led a team of Rob Mullins, Jack Lang and Alan Mycroft they cooperated on the development of minicomputer where they developed three prototypes over 5 years and has been agreed on the deployment of the third model, which became the nucleus of a stunning revolutionary educational and intellectual when it left in the final image to the world end of 2011.

The first model came out a distinct size too small and cheap, but Much smaller than fit to deal with it where he was twice the size of a small coin size almost, and it contains a port USB one outlet HDMI to connect with high quality screens, as we can see in the following figure:

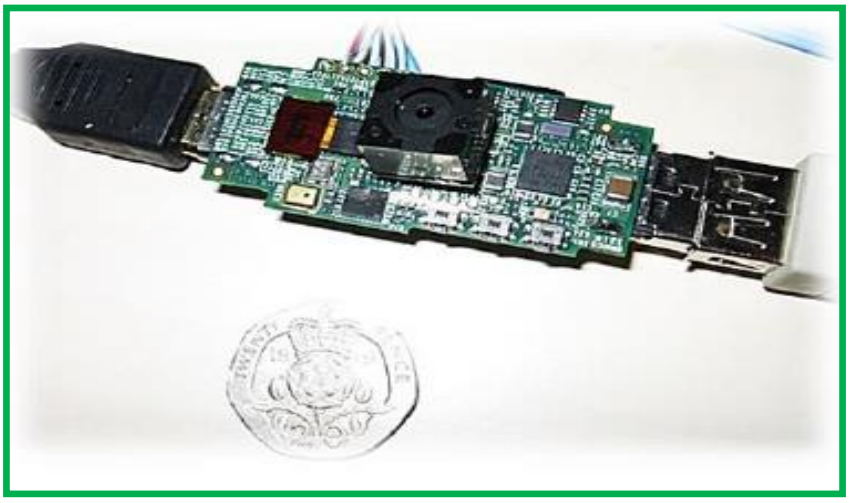


Figure II-5: The first version Raspberry Pi

The evolution of the form below to become larger than its predecessor with the addition of outlets GPIO which adds the possibility of connecting electronic components from raspberry and adding plug computer network port, the following picture shows the difference between both models.

Figure II-6: the first two models of raspi

Professor Upton wanted to make Raspberry in England not China so developments continued he added further amendments to the second model been miniaturized in size as well as the cost of manufacturing of the plate raspberry pi [17].

### II-1-2 Operating system of Raspberry Pi:

In this project we will use Raspbian-Jessie as an operating system for Raspberry pi which is one of Linux systems ready-made and there is a lot of other systems (Linux-Debian , OpenElec, Kali-Linux, Noobs, Arch-linux and who don't knows Android…), each one of these systems has its business area ( robotics , games , media , hacking… ).

### II-1-2-1 Operating System installation:

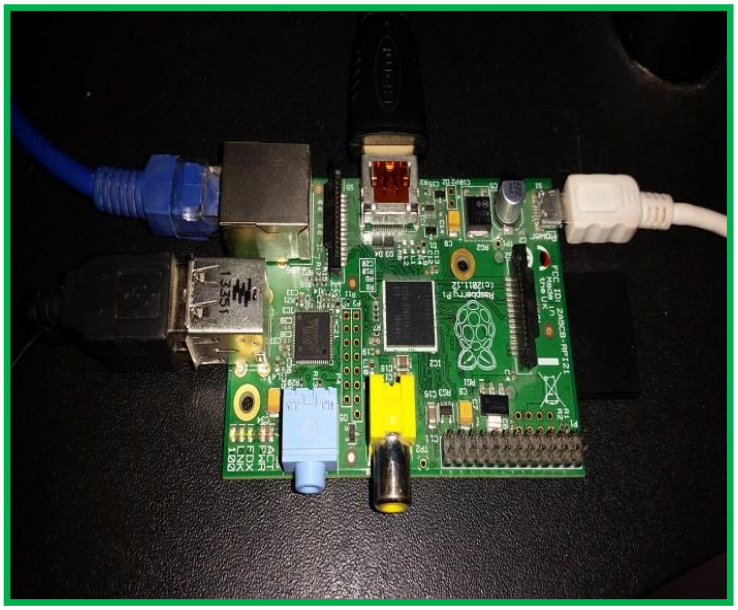


Figure II-7: Turn on the Raspberry Pi for the first time

Firstly we will need a full size SD card at least 4 GB to 32 GB maximum, type of class10 because this supports the speed of its performance.

Win32DiskImager it's a program that let us move the image of operating system (ISO form) in the SD card then we put the SD card in Raspberry.

We can run Raspberry Pi after connecting it to the screen via HDMI cable, internet via Ethernet cable, Mouse, keyboard and charger (Alimentation source 5V-1A at least), finally we will place the charger into the socket to see that Raspberry pi began installing the system automatically [17].

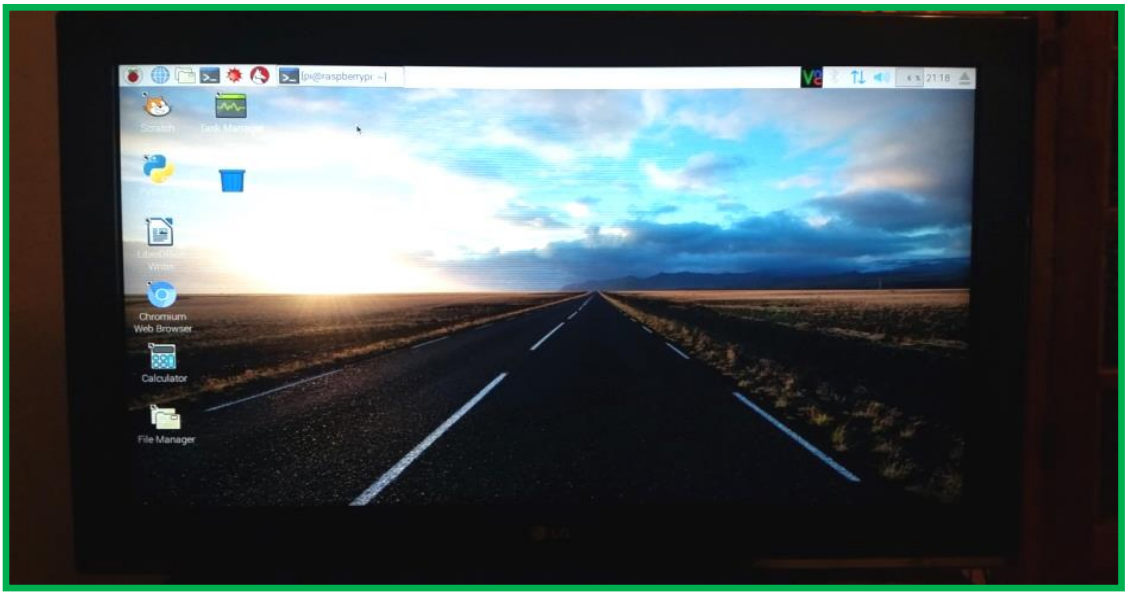



Figure II-8: Running the System for the first time (HD screen)

### II-1-3 Control on Raspberry Pi via VNC:

VNC is powerful program, easy to use and free (remote pc access software), where it will be used to display the screen of computer (Raspberry pi) via internet or local network on the screen of any other device (PC or Phone).

The program allows us to use our mouse and keyboard to control on Raspberry Pi remotely; it means that we can work on a remote computer, as if we were sitting in front of it, right from our current location.


Figure II-9: Enable the VNC

VNC allows a desktop to be viewed and controlled remotely over the Internet, VNC server must be run on the computer (RASPI) sharing the desktop, a VNC client must be run on the computer (PC or Phone) that will access the shared desktop [18].

In Rasbian-jessie system the VNC is already installed (by default), just we must enable it and that what makes it start automatically whenever raspberry pi is powered on

After that, we must install the VNC client in PC or Phone that it will access the shared desktop and logging by Contact with the same access point and write the Correct IP Address which we know it by running the command **"ifconfig"** on the command-line terminal or we can make the IP address static which is batter, so we must add the following line to "cmdline.txt" file in our SD card, which will be our new static IP address, as the following example:

```
Ip =192.168.1.7
```

After making all that successfully that means we do have full control in raspberry pi desktop, us we can see in the following two pictures:

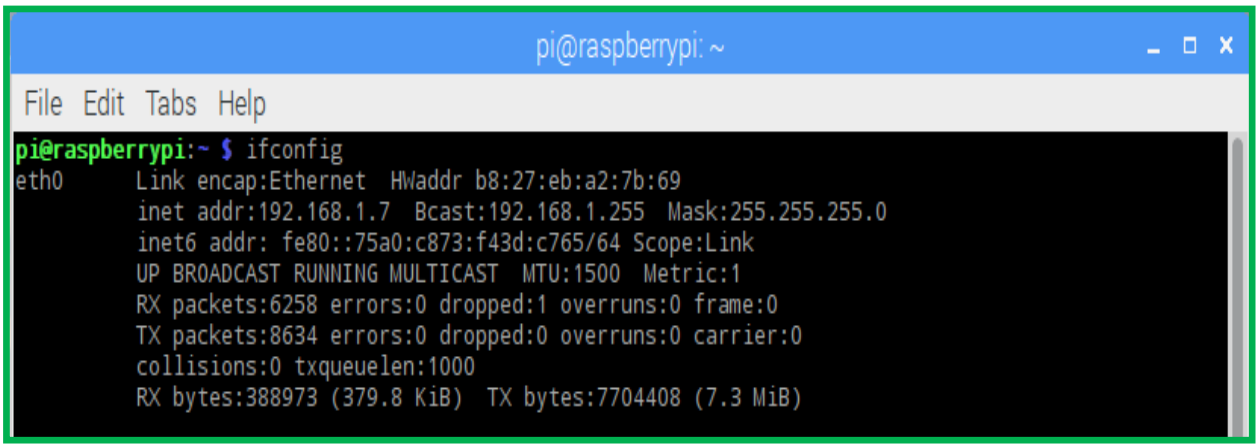



Figure II-10: IP Address (inet addr)

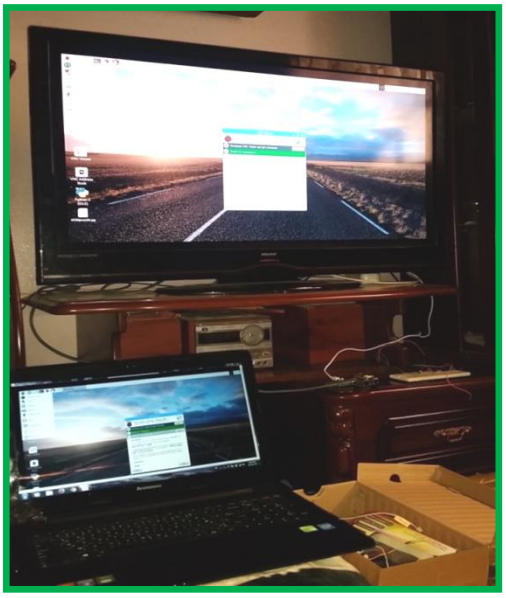


Figure II-11: Control in raspberry desktop by laptop

## II-2 Arduino Uno:

The Arduino UNO is the most used board in the family of Arduino boards. It's a microcontroller board based on the ATmega328 and it is an open-source. It has 14 digital input/output pins, where 6 of these pins can be used as PWM outputs, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. It's the latest in a series of USB Arduino boards, and the reference model for the Arduino platform [19].

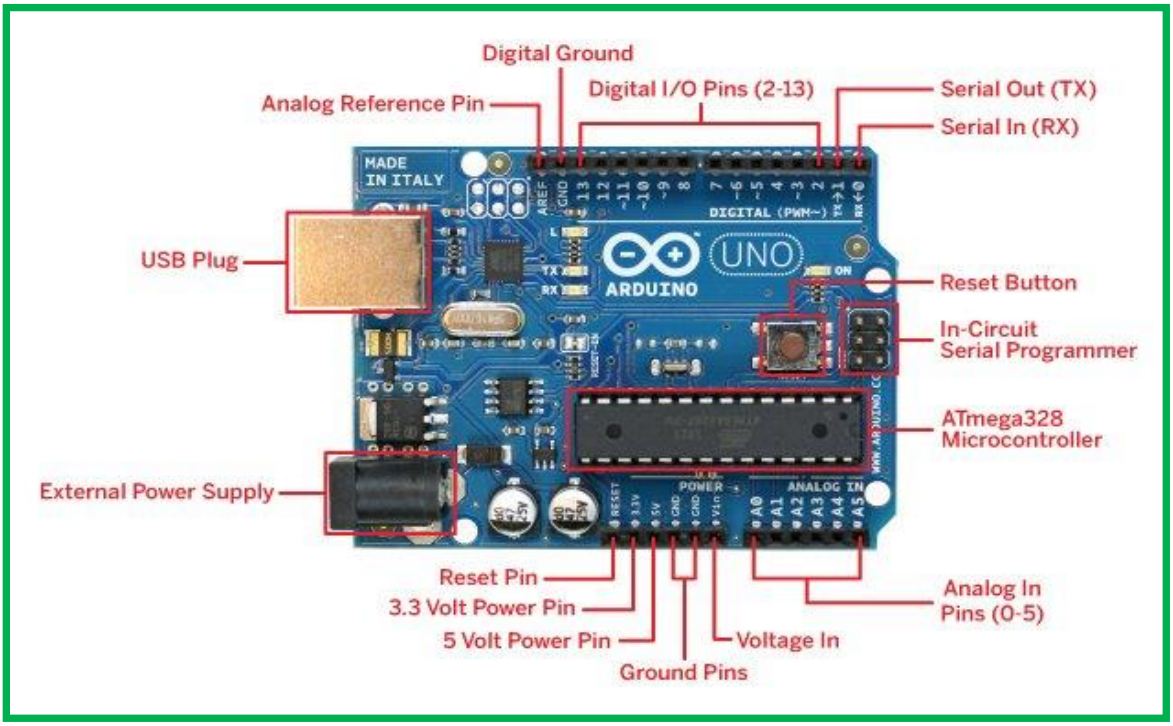



Figure II-12: Arduino Uno

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P)<br>of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

Table II-2: Technical specs of Arduino Uno

## II-2 Difference between Arduino and Raspberry Pi:

### II-2-A Advantages of Raspberry Pi over Arduino:

- **Powerfulness:** Raspberry Pi is 40 times faster than Arduino Uno Pi is capable of doing multiple tasks at a time like a computer. Building a complex project like an advanced robot controlled from a web page over internet then Pi is the best choice.
- **Networking:** Raspberry Pi contains an Ethernet port**,** through which you can directly connect to the networks. Even Internet can easily be run on Pi using some USB Wi-Fi dongles. While in Arduino, it's very difficult to connect to network.
- **Operating System:** OS can be easily switched on the single Raspberry Pi board. Pi uses SD card as flash memory to install the OS.

### II-2-B Advantages of Arduino over Raspberry Pi:

- **Simplicity:** It's very easy to interface electronic components with Arduino, with just few lines of code. While in Raspberry pi, there is much overhead (installing libraries and softwares) for simply reading those sensors. Adding that there is no RTC (Real Time Clock) or ADC (Analog Digital Converter) on Raspberry Pi but there is an RTC and an ADC on Arduino.
- **Robustness:** Raspberry Pi runs on a OS so it must be properly shut down before turning OFF the power, otherwise OS and applications may get corrupt and Pi can be damaged. While Arduino is just a plug and play device which can be turned ON and OFF at any point of time, without any risk of damage.
- **Power consumption:** Pi is a powerful hardware, it needs continuous 5v power supply and it is difficult to run it on Batteries, while Arduino needs less power can easily be powered using a battery pack.
- **Price:** Obviously Arduino is cheaper than Raspberry Pi, Arduino costs around $10-20 depending on the version, while price of Raspberry is around $35-40 [20].

## II-4 Robot Platform:

Robot platform will be like a composition of an aluminum plaque with specific size contains two geared motors with two wheels to control the trajectories that robot takes, the third wheel free just to make robot balanced, sensors we need in project and finally breadboard for any extra experiments.

### II-4-1 Bread board:

I added a breadboard to my robot because it helps to use the robot experimentally, which means this robot can contain many other experiments whether simple or complicated (Blinking Led, Integrating Arduino with raspberry pi, Motion Detector, PWM control…). That's what makes my mobile robot experimental.

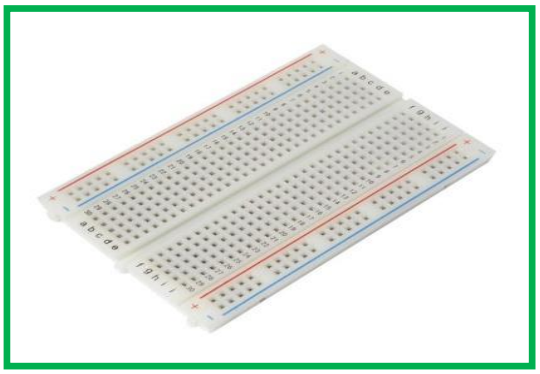


Figure II-13: Breadboard

### II-4-2 Geared DC motor:

A geared DC motor has a gear assembly (gear box) attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute. The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a geared motor makes its speed reduced to any desirable figure. These motors are, small, easy to install and ideally suited for use in a mobile robot car [21] [22].

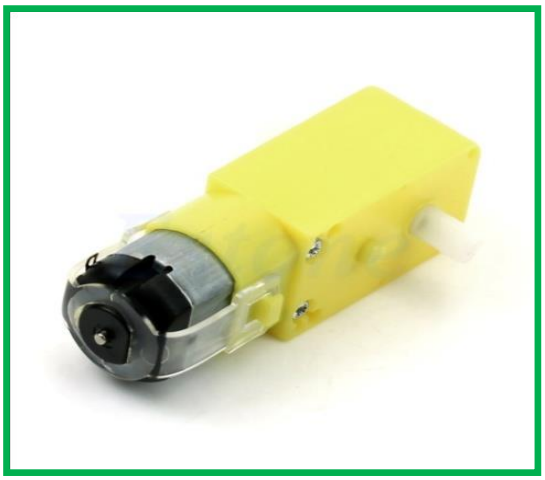


Figure II-14: Geared motor

❖ **Specification:**

- Strong magnetic with anti-interference
- Double axis gear motor
- Reduction ratio: 48:1
- Working voltage: 3V ~ 6V
- Supply current: ≤200mA at 6V, ≤150mA at 3V
- Supply speed: 200 ± 10%RPM at 6V, 90 ± 10%RPM at 3V

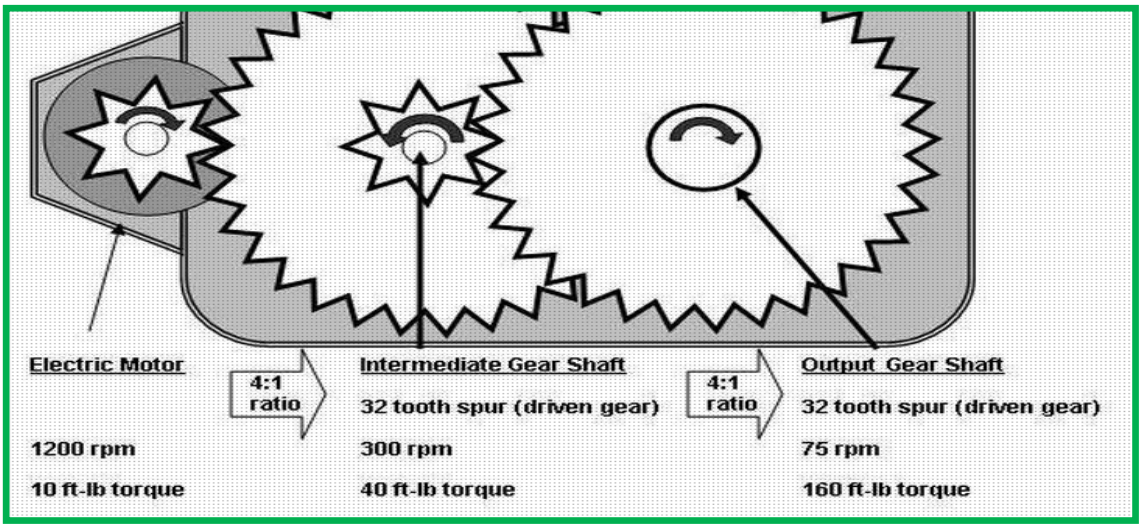



Figure II-15: Planogram of gear box (just an example)

### II-4-3 Micro Servo Motor (DXW90):

A Servo Motor is a combination of DC motor, position control system and gears, it is an electrical device which can push or rotate an object with great precision, which means rotating an object at some specific angles or distance and it is just made up of simple motor which run through servo mechanism.

Servo motors have many applications in the modern world and available in different shapes and sizes. We will be using DXW90 Servo Motor in this project, it is one of the popular and cheapest one. DXW90 is a 180 degree servo. So with this servo we can position the axis from 0 to 180 degrees.

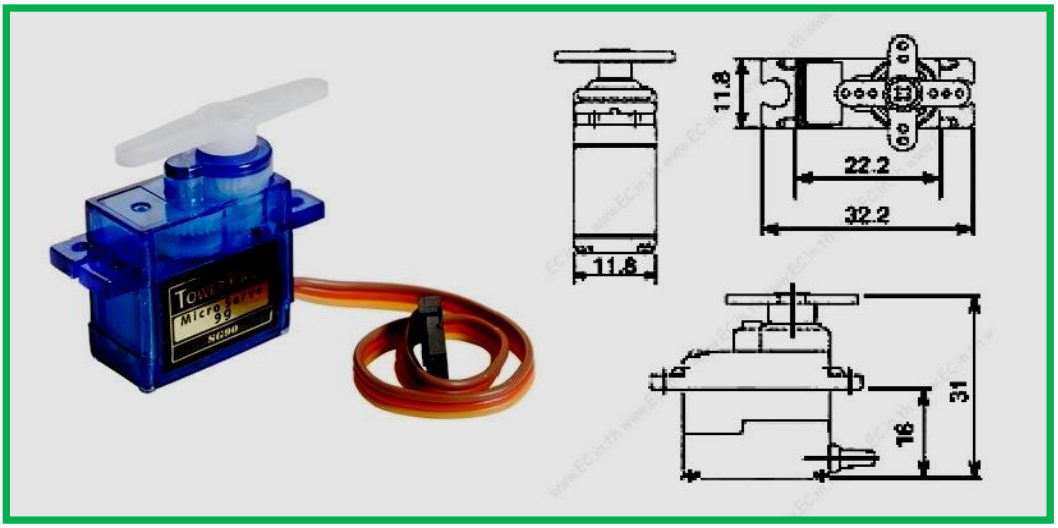



Figure II-16: DXW90 Servo Motor with dimensions measurements

A Servo Motor mainly has three wires, one is for positive voltage, another is for ground and last one is for position setting. The Red wire is connected to power, Brown wire is connected to ground and Yellow wire (or WHITE) is connected to signal.

### II-4-3-1 PWM Function:

Servo motor works on PWM (Pulse width modulation) principle, also DC motor can be controlled with PWM by L293D. The frequency of PWM signal can vary based on type of servo motor, for DXW90 the frequency of PWM signal is 50Hz. The PWM principle means that the angle of rotation is controlled by the duration of applied pulse to its Control PIN. As we can see in the following figures:

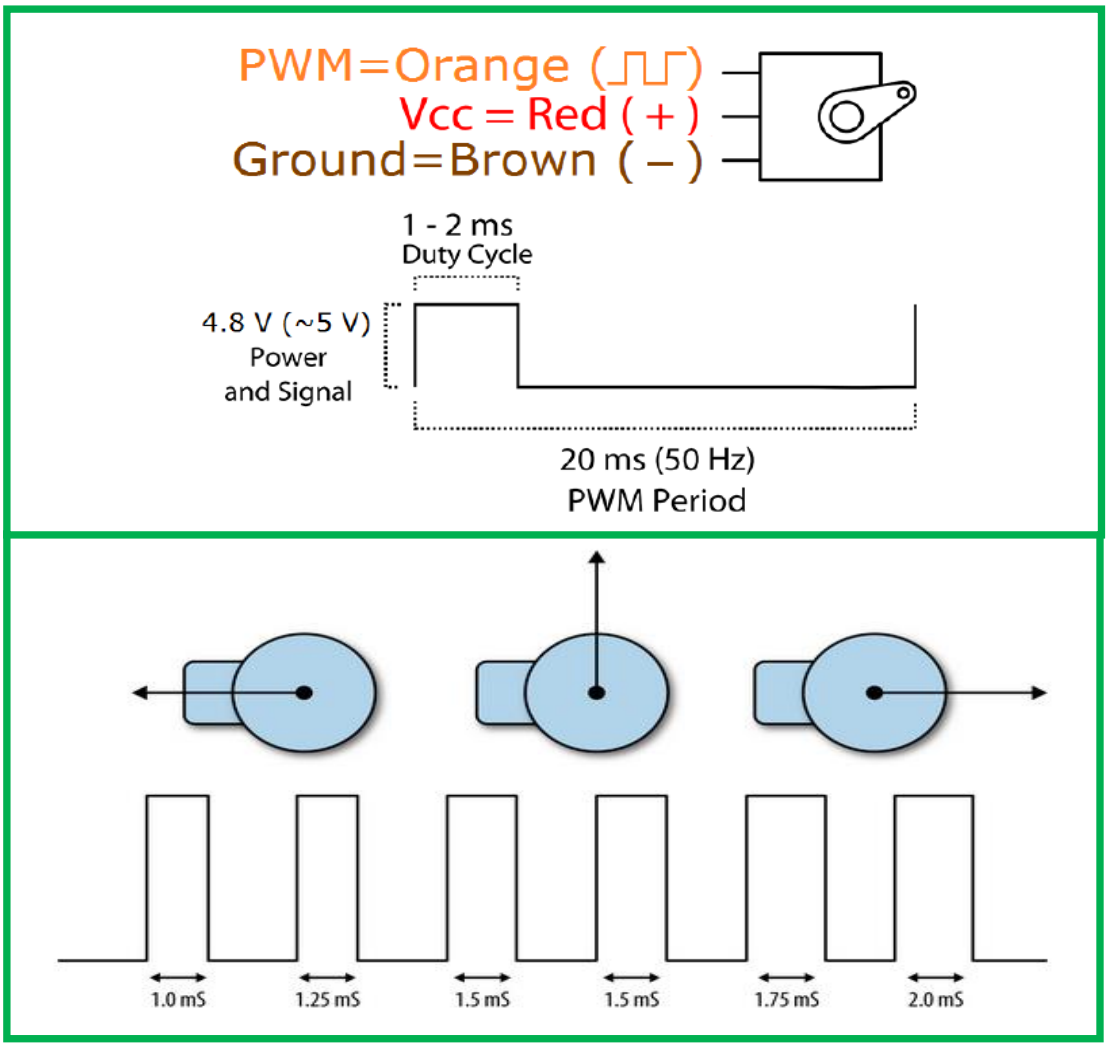



Figure II-17: Relation between rotation and Pulse duration

The table below shows the Servo Position for that particular Duty Ratio. We can get any angle in between by choosing the value accordingly. So for 45º of servo the Duty Ratio should be '5' which means 5% and for 135º of servo the Duty Ratio should be '10' which means 10%. As we can see in the following table [23]:

| POSITION | DUTY RATIO |
|---|---|
| 0º | 2.5 |
| 90º | 7.5 |
| 180º | 12.5 |

Table II-3: Effect of Duty Ratio on Servo position

### ❖ Specifications

- Weight: 9 g
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Running current with 5V supply is 220 ±50mA
- Temperature range: 0 ºC – 55 ºC

## II-4-4 Motors Driver:

### II-4-4-1 the integrated circuit L293D:

L293D is a 16-pin integrated circuit contains two H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

It works on the concept of H-bridge which finds a huge application in the area of robotics to control motor direction. This circuit uses the basic concept of transistors as a switch. Transistor with proper biasing can be used as switch; it can be used to toggle between the two states of a switch on or off. This configuration of transistor has been used in H-bridge to drive a motor in both clockwise and anticlockwise direction [24] [25].

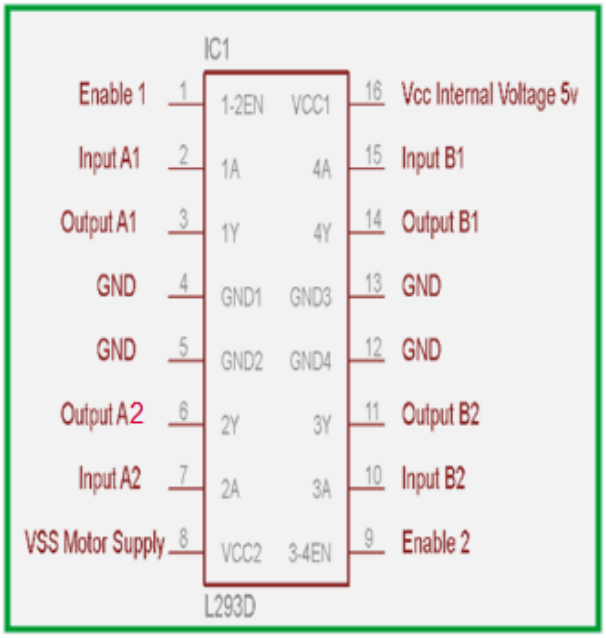



Figure II-18: L293D Pin diagram

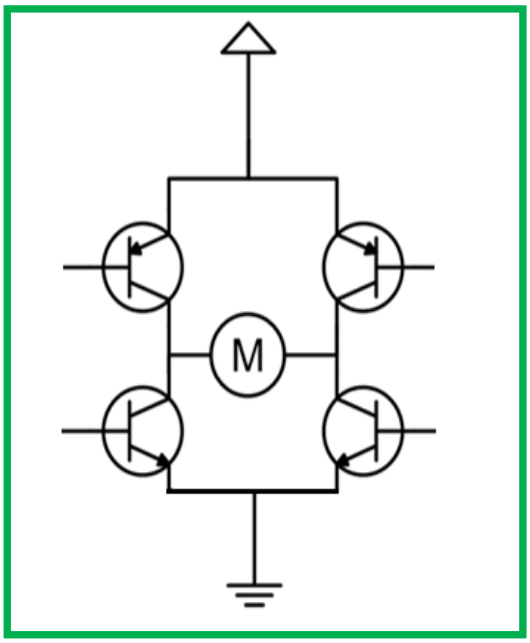



Figure II-19: H-Bridge schema

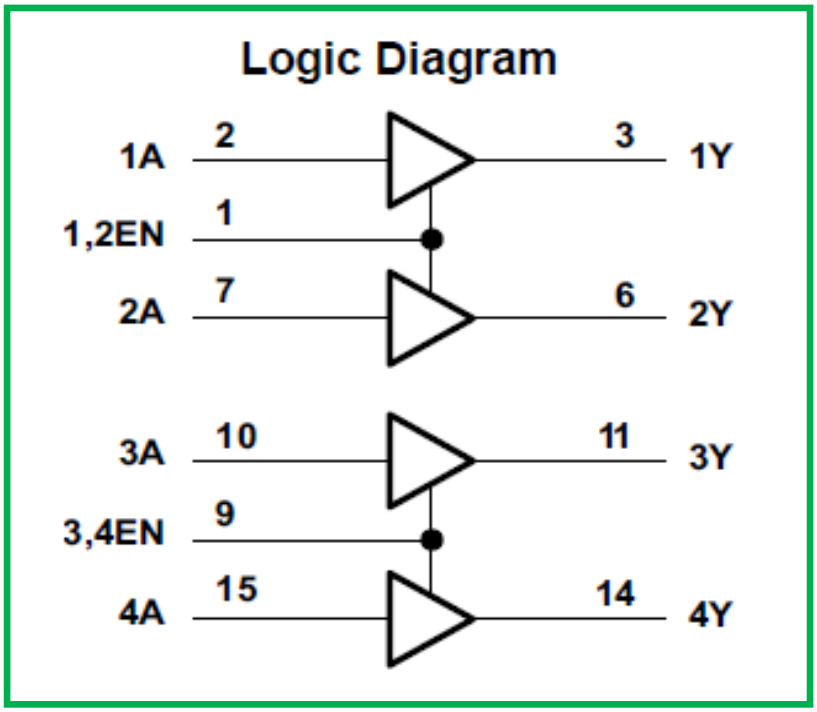



Figure II-20: Logic diagram of L293D

| Pin No | Function | Name |
|---|---|---|
| 1 | Enable pin for Motor 1; active high | Enable 1,2 |
| 2 | Input 1 for Motor 1 | Input 1 |
| 3 | Output 1 for Motor 1 | Output 1 |
| 4 | Ground (0V) | Ground |
| 5 | Ground (0V) | Ground |
| 6 | Output 2 for Motor 1 | Output 2 |
| 7 | Input 2 for Motor 1 | Input 2 |
| 8 | Supply voltage for Motors; 9-12V (up to 36V) | $V_{cc\,2}$ |
| 9 | Enable pin for Motor 2; active high | Enable 3,4 |
| 10 | Input 1 for Motor 2 | Input 3 |
| 11 | Output 1 for Motor 2 | Output 3 |
| 12 | Ground (0V) | Ground |
| 13 | Ground (0V) | Ground |
| 14 | Output 2 for Motor 2 | Output 4 |
| 15 | Input2 for Motor 2 | Input 4 |
| 16 | Supply voltage; 5V (up to 36V) | $V_{cc\,1}$ |

Table II-4: L293D Pins description

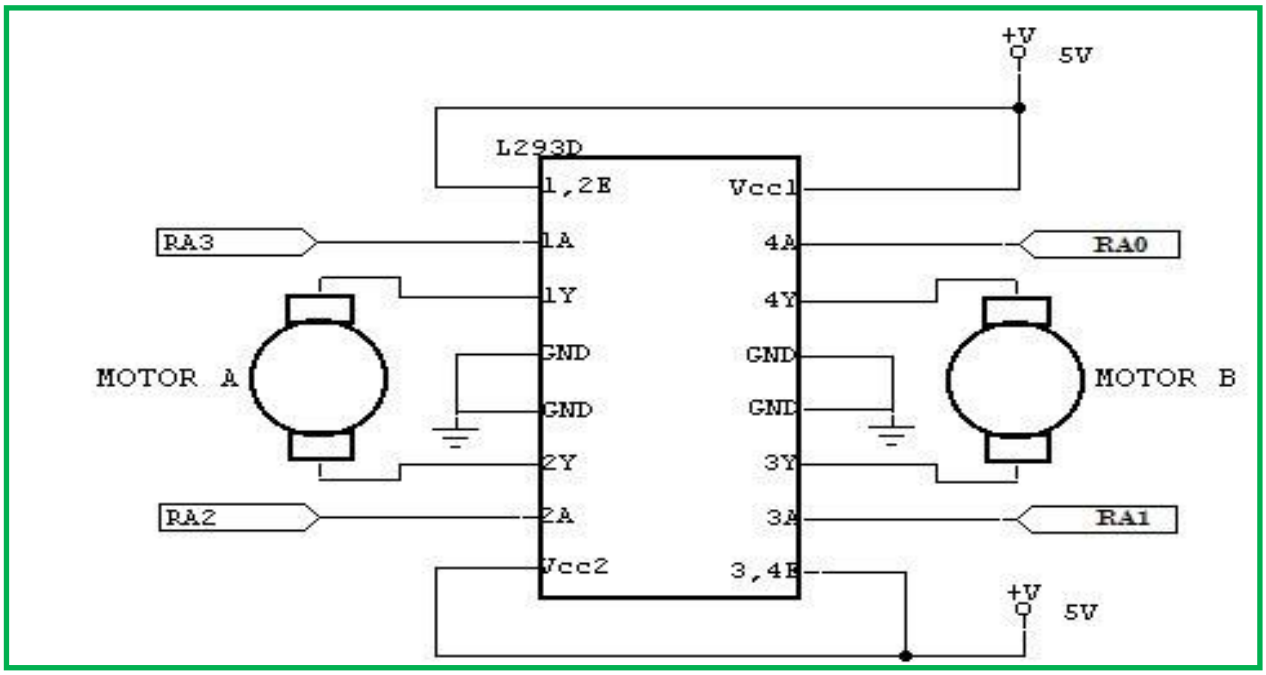



Figure II-21: Connecting DC motors to L293D

| Direction / Pins | STOP | FORWARD | RIGHT | LEFT | BACKWARD | STOP |
|---|---|---|---|---|---|---|
| Input 01 "A1" | 0 | 1 | 0 | 1 | 0 | 1 |
| Input 02 "A2" | 0 | 0 | 1 | 0 | 1 | 1 |
| Input 03 "A3" | 0 | 1 | 1 | 0 | 0 | 1 |
| Input 04 "A4" | 0 | 0 | 0 | 1 | 1 | 1 |

Table II-5: H-bridge inputs relation with direction

❖ **Specification:**

VCC is the voltage that it needs for its own internal operation 5V where L293D will not use this voltage for driving the motor. For driving the motors it has a separate provision to provide motor supply VSS where L293D will use this to drive the motor. It means if you want to operate a motor at 9V then you need to provide a Supply of 9V across VSS Motor supply.

The maximum voltage for VSS motor supply is 36V. It can supply a max current of 600mA per channel. Since it can drive motors Up to 36v hence you can drive big motors with L293D [25].

| | | | MIN | NOM | MAX | UNIT |
|---|---|---|---|---|---|---|
| | Supply voltage | $V_{CC1}$ | 4.5 | | 7 | V |
| | | $V_{CC2}$ | $V_{CC1}$ | | 36 | |
| $V_{IH}$ | High-level input voltage | $V_{CC1} \leq 7$ V | 2.3 | | $V_{CC1}$ | V |
| | | $V_{CC1} \geq 7$ V | 2.3 | | 7 | V |
| $V_{IL}$ | Low-level output voltage | | –0.3(1) | | 1.5 | V |
| $T_A$ | Operating free-air temperature | | 0 | | 70 | °C |

Table II-6: Recommended operating conditions of L293D

There is a problem which is that Raspberry pins can provide just 3.3V and L293D needs 5V to drive motors, so I will use the integrated circuit MC74HC244AN "Buffer".

### II-4-4-2 the integrated circuit MC74HC244AN "Buffer":

I will use the integrated circuit **"Buffer"** like an amplifier which will help me to convert the 3.3V of Raspberry pins to 5v which gives me the ability to drive the two DC geared motors by L293D. This device contains protection circuitry to guard against damage due to high static voltages or electric fields. This will help me for do a proper operation.

Vin and Vout should be constrained to the range GND ≤ (Vin or Vout) ≤ VCC. Unused inputs must always be tied to an appropriate logic voltage level (either GND or VCC). Unused outputs must be left open [26].

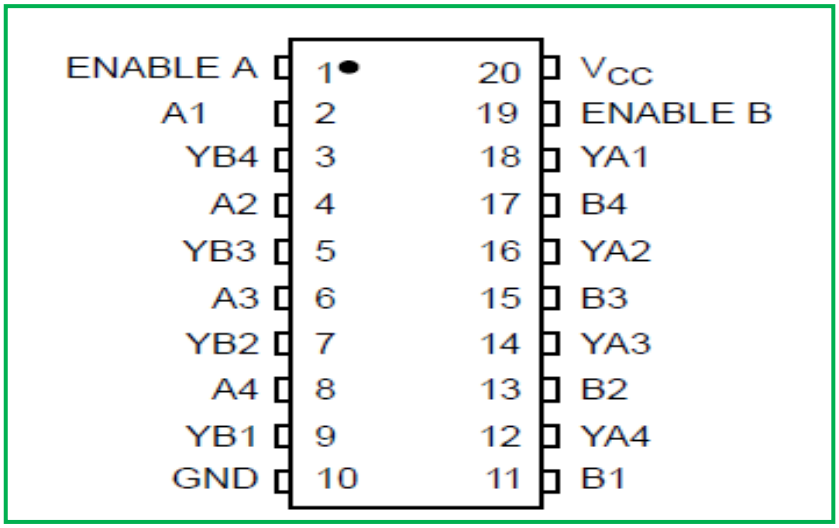



Figure II-22: MC74HC244AN Pins

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $V_{CC}$ | DC Supply Voltage (Referenced to GND) | 2.0 | 6.0 | V |
| $V_{in}$, $V_{out}$ | DC Input Voltage, Output Voltage (Referenced to GND) | 0 | $V_{CC}$ | V |
| $T_A$ | Operating Temperature, All Package Types | –55 | +125 | °C |

Table II-7: Recommended operating conditions of MC74HC244AN

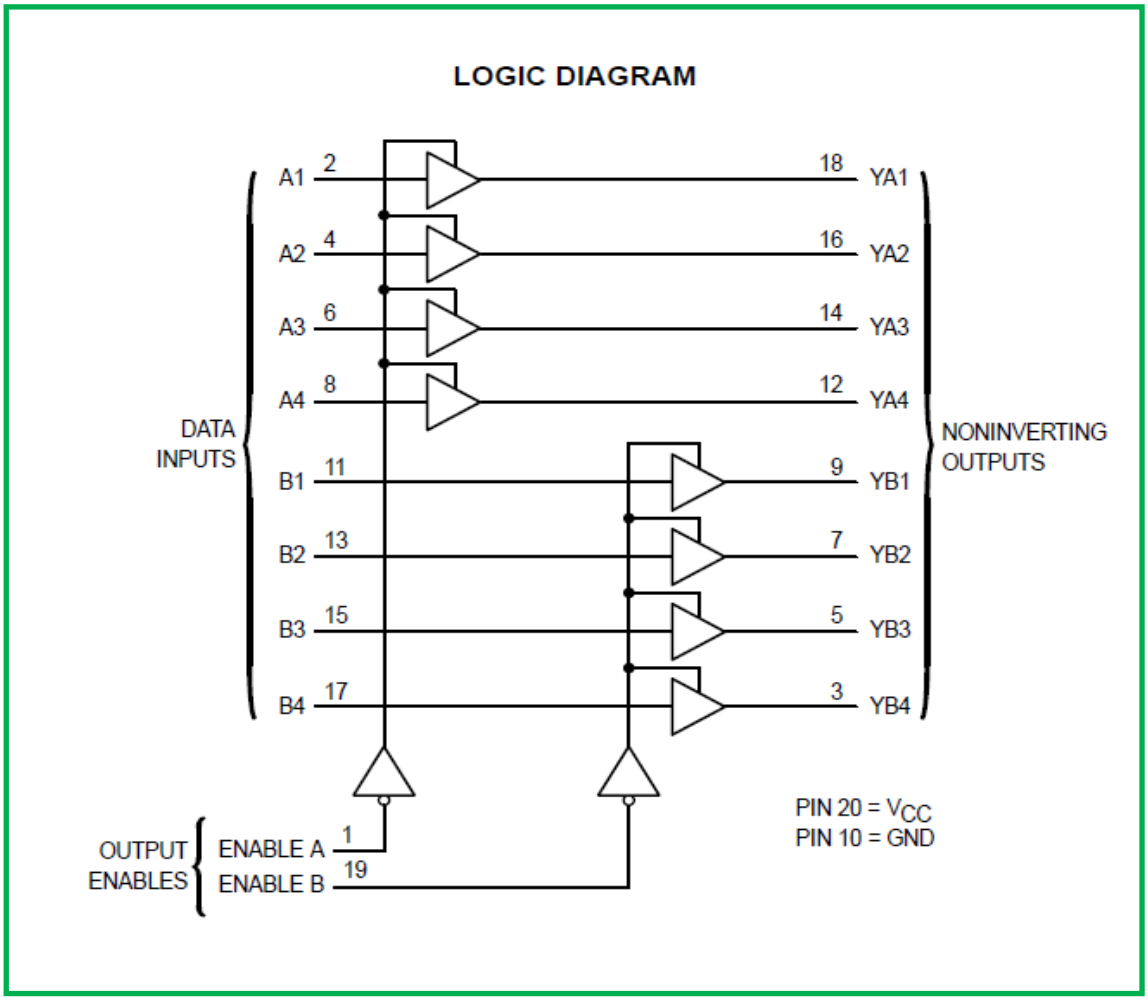



Figure II-23: Logic diagram of MC74HC244AN

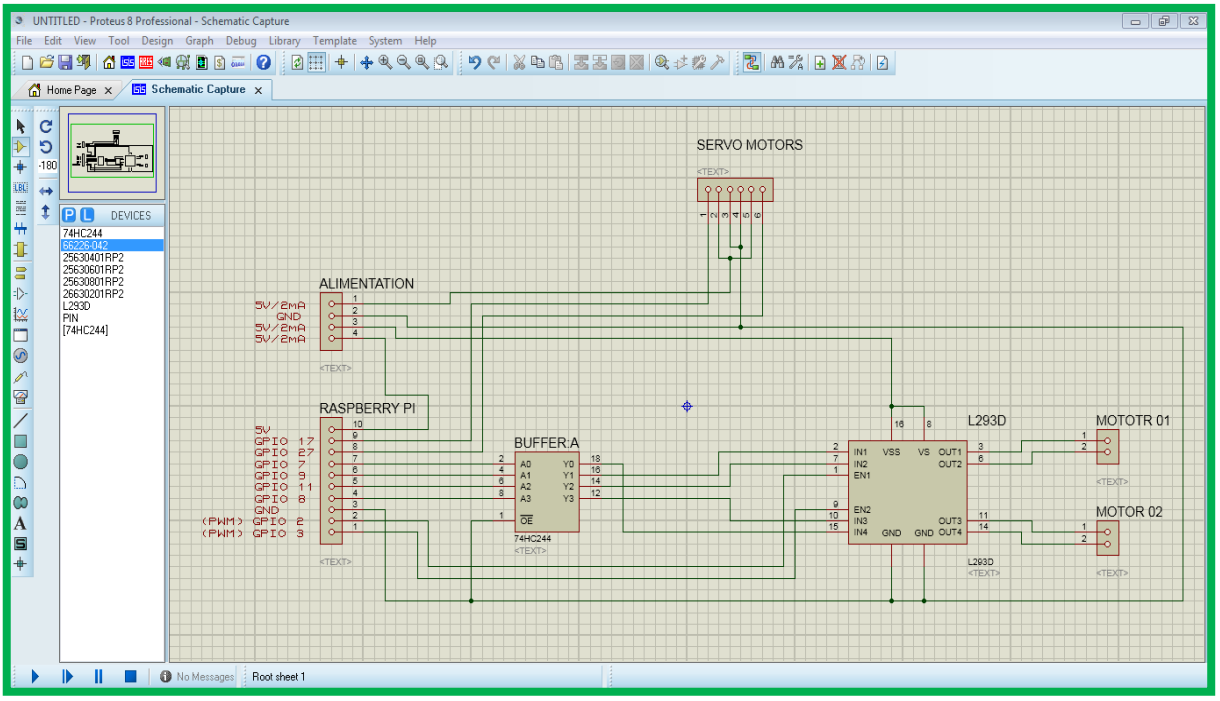



Figure II-24: Creating the electric schema of control circuit with Proteus

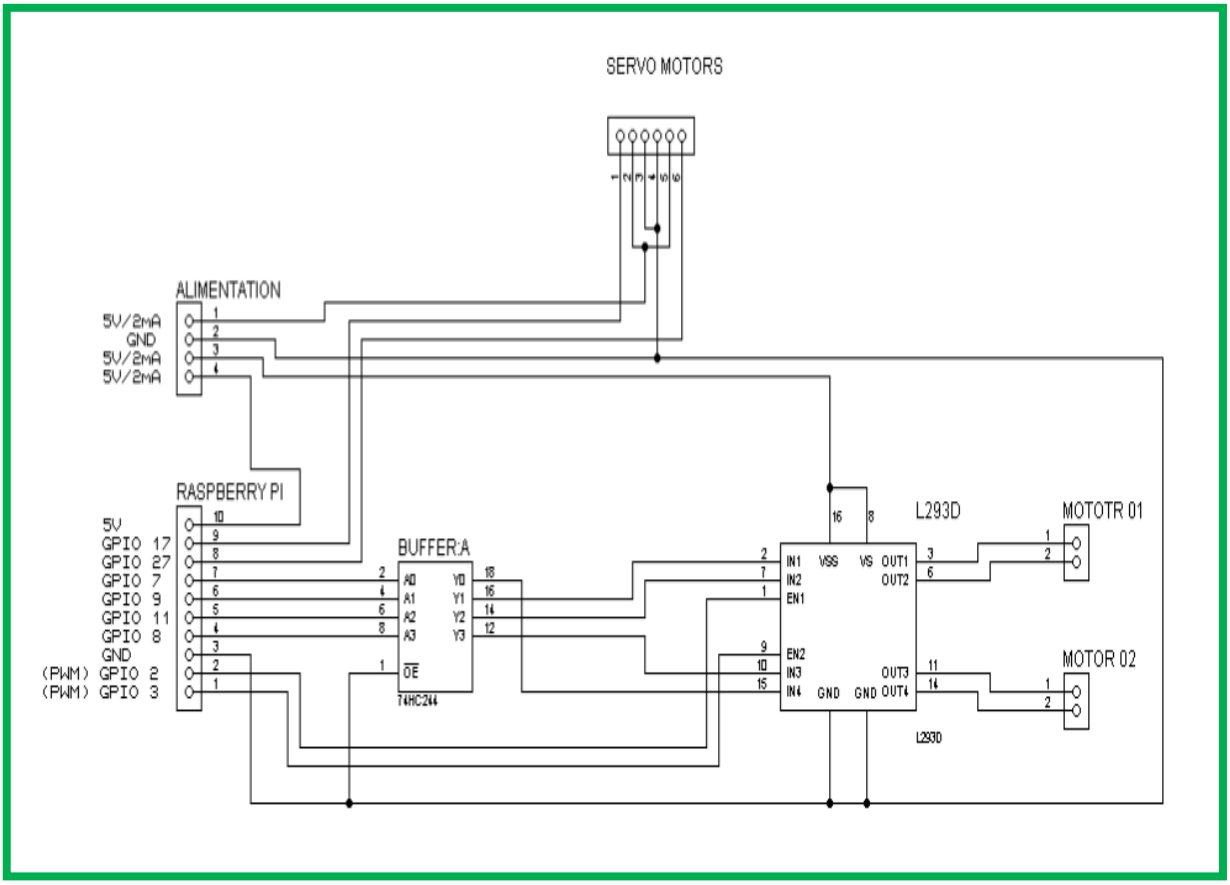



Figure II-25: Electric Schema of control circuit

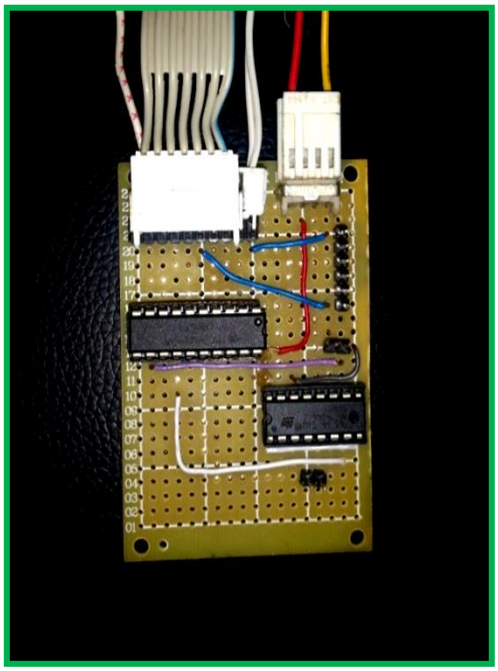


Figure II-26: Printed circuit board "MC74HC244AN+L293D"

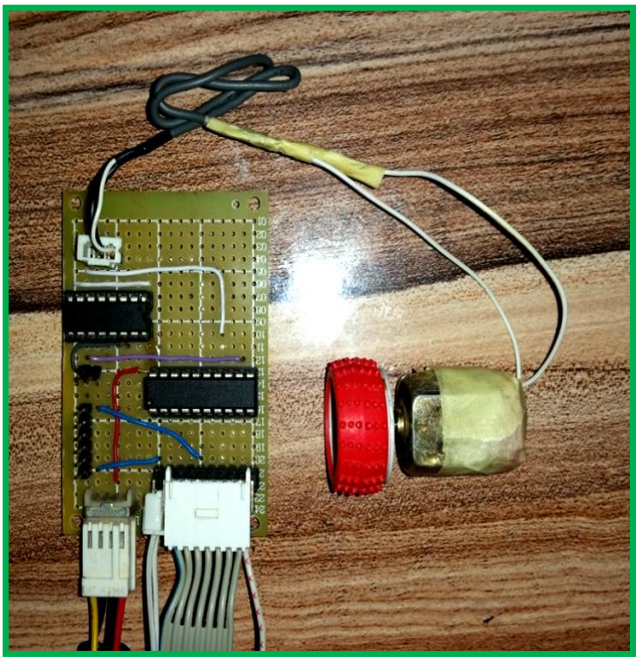


Figure II-27: Connecting DC motor to motors driver "L293D"

## II-5 Infrared light:

Many electronic circuits detect the invisible light that's commonly called infrared; it's a light where its frequency is just below the range of visible red light and specifically falls between 1 THz to 400 THz (one THz is one trillion cycles per second).

The first wireless IR remote control was developed by Zenith in 1955. It used ordinary visible light, could turn the TV on or off, and could change channels [27].

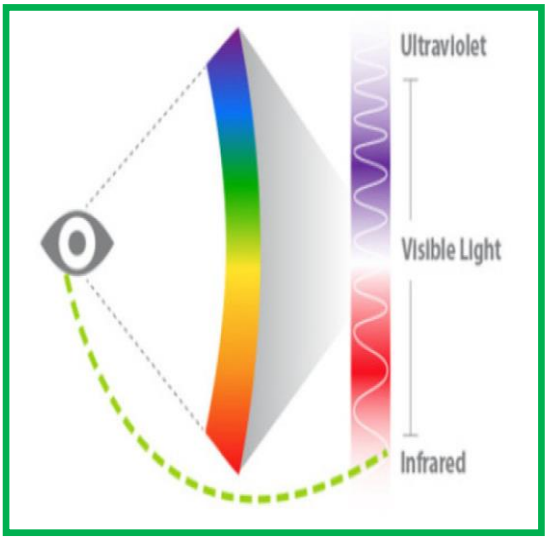



Figure II-28: IR invisibility

### II-5-1 Infrared Receiver:

Infrared receiver or IR receiver is the part of a device that receives infrared commands from a remote control. It is hardware that transmits information from an infrared remote control to another device by receiving and decoding signals.

The receiver outputs a code to uniquely identify the infrared signal that it receives; this code is used in order to convert signals from the remote control into a format that can be understood by the other device [28].

### II-5-1-1 Types of Infrared Receivers:

There are many different kinds of infrared receivers and the most common types categorized by supply voltage, carrier frequency, transmission distance, Operating Temperature Range, Directivity and supply current…, where in this project we will use the TSOP38238 IR receiver.

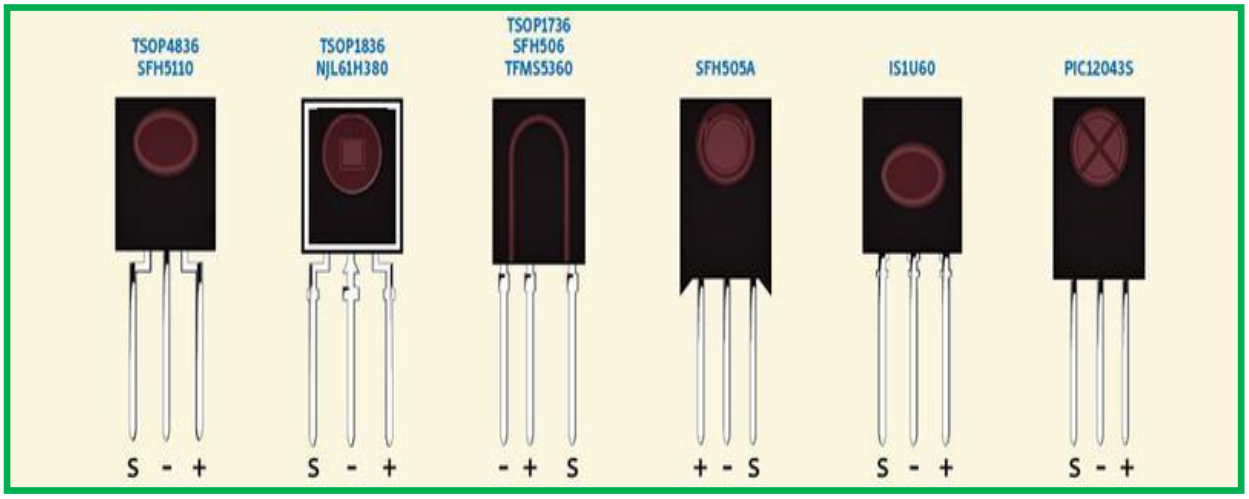



Figure II-29: Types of Infrared Receivers

❖ **specifications**:

- Carrier Frequency: 38kHz
- Very low current supply
- Supply Voltage Range: 2.5V to 5.5V
- Supply Current: 350µA
- Transmission distance: 45m
- Directivity: 45°
- Operating Temperature Range: -25°C to +85°C

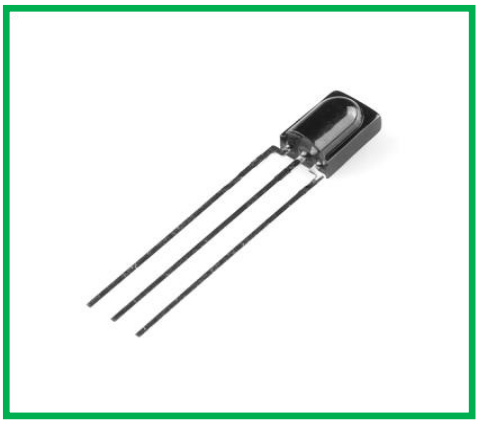


Figure II-30: TSOP382 IR receiver

TSOP38238 IR sensor tuned to 38 KHz which means 38000 cycles per second, transmission distance 45m and directivity 45°, so it is suitable for receiving commands from our remote control and Runs at 3V to 5V which is also suitable to use on Raspberry pi [29][30].

### II-5-1-2 Applications for Infrared Receivers:

Infrared receivers can often be found in consumer products such as home entertainment systems (Video equipment, DVD, Blu-Ray players and audio amplifiers). Infrared receivers can also be found in the industrial, military, aerospace robots and photography markets.

Infrared light is used also to detect objects that we can't see in visible light. One common application of this is night vision, According to a principal of physics called Planck's law; I can say devices that can detect infrared light can literally see in the dark [29].

### II-5-1-3 Obstacles Detection using Infrared:

IR can be used to detect obstacles that are further away even when it's dark by making the IR LED headlights brighter. We can increase its detection range by resisting electric current less, a smaller resistor allows more current to flow through the IR LED. More current is what causes it to glow more brightly, which means a wider detection range, also means wider control range and very less hampering if there is obstacles [31]. For example using two IR emitters and two IR receivers in the each corner with a specific angle to give the best receiving, as we can see in the following figure:

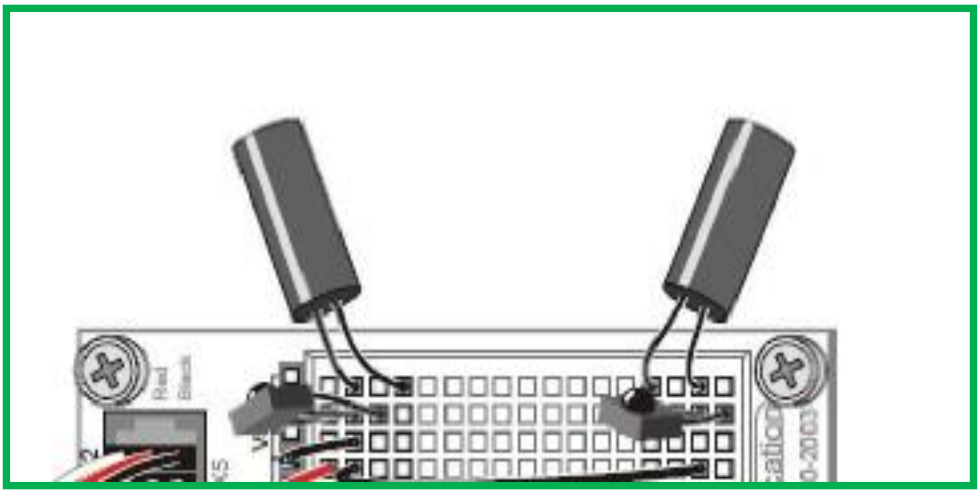


Figure II-31: Two IR components to detect obstacles

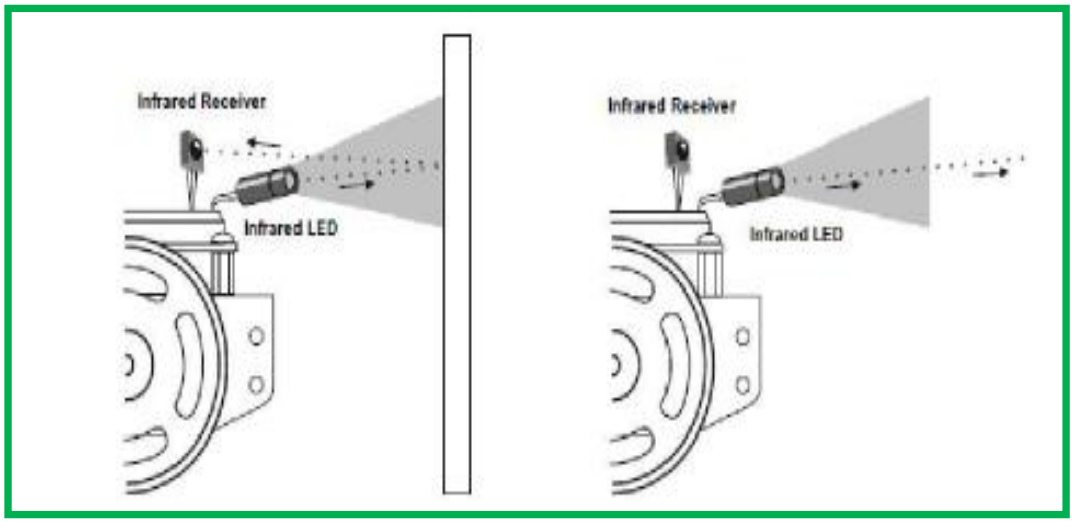



Figure II-32: Reflecting the IR light on an obstacle

## II-6 Ultrasonic Sensor:

### II-6-1 Definition:

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object. We use ultrasonic waves because they are relatively accurate across short distances and don't cause disturbances as they are inaudible to human ear [32].

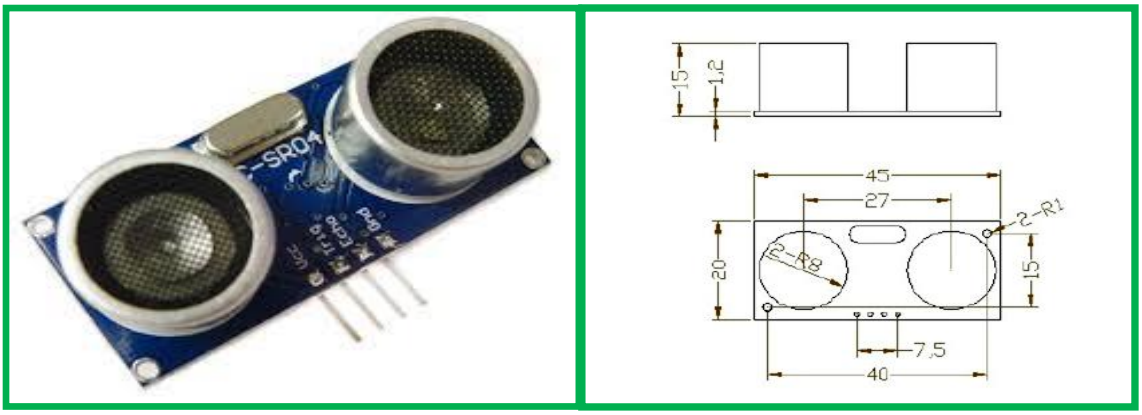



Figure II-33: Ultrasonic sensor HC-SR04

HC-SR04 is a commonly used module for distance measurement from 2cm to 400 cm, since it is known that sound travels through air at about 344 m/s (1129 ft/s), I can take the time for the sound wave to return and multiply it by 344 meters to find the total round-trip distance of the sound wave. Round-trip means that the sound wave traveled 2 times the distance to the object before it was detected by the sensor; it includes the trip from the sonar sensor to the object and the trip from the object to the Ultrasonic sensor (after the sound wave bounced off the object), to find the distance to the object, simply divide the round-trip distance in half.

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2} \quad \text{(II.6.1)}$$

It is important to understand that some objects might not be detected by ultrasonic sensors. This is because those objects are shaped or positioned in such a way that the sound wave bounces off the object, but are deflected away from the Ultrasonic sensor. It is also possible for the object to be too small to reflect enough of the sound wave back to the sensor to be detected. Other objects can absorb the sound wave all together (cloth, carpeting, etc), which means that there is no way for the sensor to detect them accurately. These are important factors to consider when designing and programming a robot using an ultrasonic sensor [33].

### II-6-2 Working way of HC-SR04:

HC-SR04 ultrasonic distance sensor module has four pins:

- VCC – 5V input power
- TRIG – Trigger Input
- ECHO – Echo Output
- GND – Ground

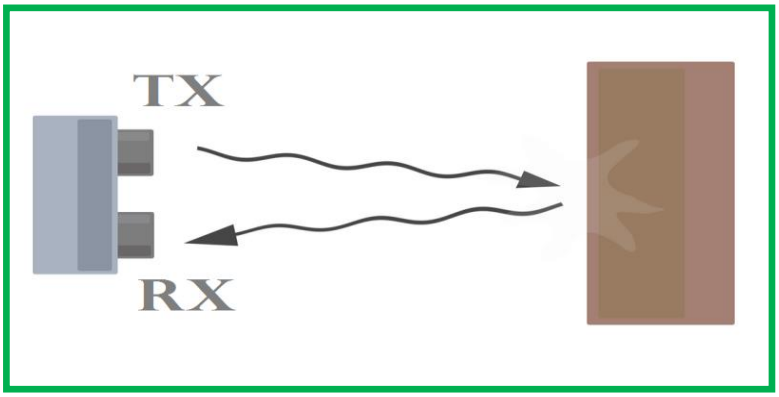



Figure II-34: Ultrasonic Module Operation

HC-SR04 function summarized in the following stages:

1. Provide trigger signal to TRIG input, it requires a HIGH signal of at least 10µS duration.
2. This enables the module to transmit eight 40 KHz ultrasonic burst.
3. If there is an obstacle in-front of the module, it will reflect those ultrasonic waves
4. If the signal comes back, the ECHO output of the module will be HIGH for duration of time taken for sending and receiving ultrasonic signals. The pulse width ranges from 150µS to 25mS depending upon the distance of the obstacle from the sensor and it will be about 38ms if there is no obstacle [34].

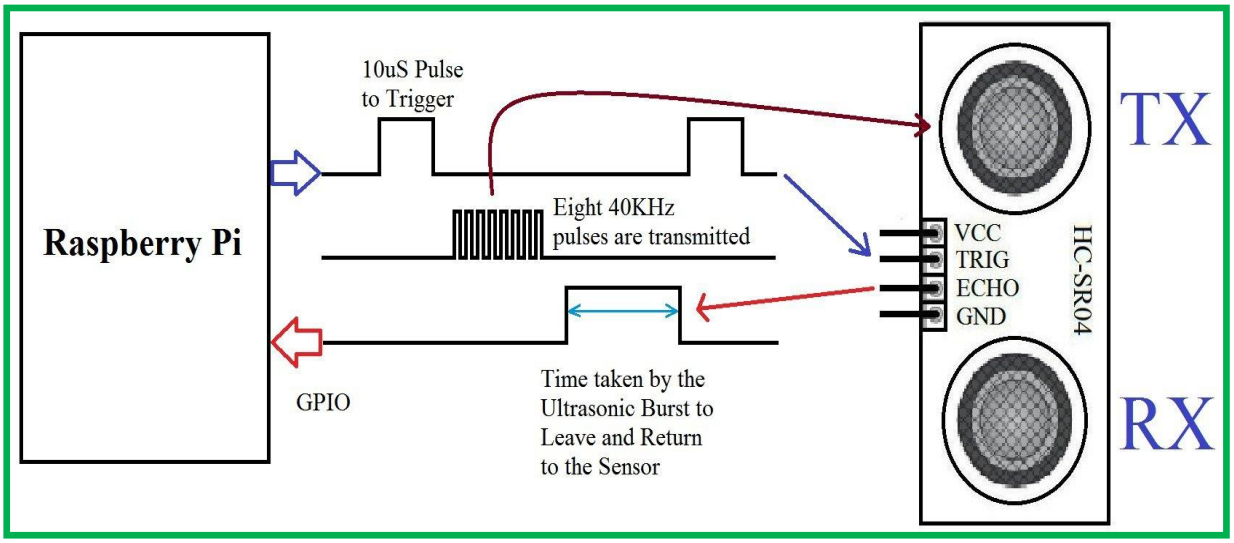



Figure II-35: Raspberry pi interaction with HC-SR04

| | |
|---|---|
| **Working Voltage** | **DC 5 V** |
| **Working Current** | **15mA** |
| **Working Frequency** | **40Khz** |
| **Max Range** | **4m** |
| **Min Range** | **2cm** |
| **MeasuringAngle** | **15 degree** |
| **Trigger Input Signal** | **10uS TTL pulse** |
| **Echo Output Signal** | **Input TTL lever signal and the range in proportion** |
| **Dimension** | **45*20*15mm** |

Table II-8: Recommended operating conditions of HC-SR04

## II-7 Web Application:

### II-7-1 Definition:

In computing, a web application or web app is a client–server software application in which the client (or user interface) runs in a web browser. It is an application program that is stored on a remote server and the data for a Web app may be stored locally or on the Web, or in both locations and the delivering will be through a browser interface.

### II-7-2 HTML:

The definition of HTML is Hypertext Markup Language, is a computer language devised to allow website creation where the HTML code determines the visual layout and executed via browser. This website can then be viewed by anyone else connected to the internet or without internet means only you can see this website by Wi-Fi network just as I did in my web app [35].


- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

### II-7-3 Wi-Fi:

Wi-Fi is short for Wireless Fidelity, it's the name of a popular wireless networking technology that uses radio waves to transmit information across a network and provide wireless high-speed Internet and network connections, utilizes one of the IEEE 802.11 wireless standards to achieve a wireless connection to a network.

A Wi-Fi connection is established using a wireless adapter to create hotspots areas in the vicinity of a wireless router that are connected to the network and allow users to access internet services. The nature of its work is based on providing wireless connectivity to your devices by emitting frequencies between 2.4GHz - 5GHz, according to the amount of data on the network [36].

### II-7-3-1 Wi-Fi frequencies and speed:

The 802.11 networking standards will somewhat vary depending mostly on the user's needs, as follows:

- The 802.11a will transmit data at a frequency level of 5GHz. We can transmit a maximum of 54 megabits of data per second.
- The 802.11b will transmit data at a frequency level of 2.4GHz. We can transmit a maximum of 11 megabits of data per second.
- The 802.11g will transmit data at 2.4GHz but can transmit a maximum of 54 megabits of data per second.
- The more advanced 802.11n can transmit a maximum of 140 megabits of data per second and uses a frequency level of 5GHz [36].

### II-7-3-2 Wireless Range:

The range of a Wi-Fi computer network depends on the number, type of wireless access points and wireless routers used to build it.

- A traditional home network having one wireless router can cover a single-family dwelling but often not much more.
- Business networks with grids of access points can cover large office buildings.
- We can find in some cities wireless hotspots spanning several square kilometers [37].

## II-7-4 GPIO Web service and Displaying Camera:

In this project the GPIO pins will be remote-controlled with Wi-Fi and all that by creating a web application, which means creating a web site interface by HTML that contains many buttons (forward, backward, right, left, stop…) to take control in GPIO pins therefore taking control in my mobile robot. Plus, there is a camera on the robot which specifically is Android phone camera, for live viewing.

Displaying the camera depends on an android application (IP Camera), which work with Wi-Fi and its function is creating a web app to run the live viewing on it and making a lot of video settings, the IP address of this web app will be given by the android application itself. Our task is to extract the IP address of the live viewing without all the complicated video settings and putting it into our web app program, by doing these steps correctly the live show will work.

As we can see data of the web service and viewing interface in the following figures:

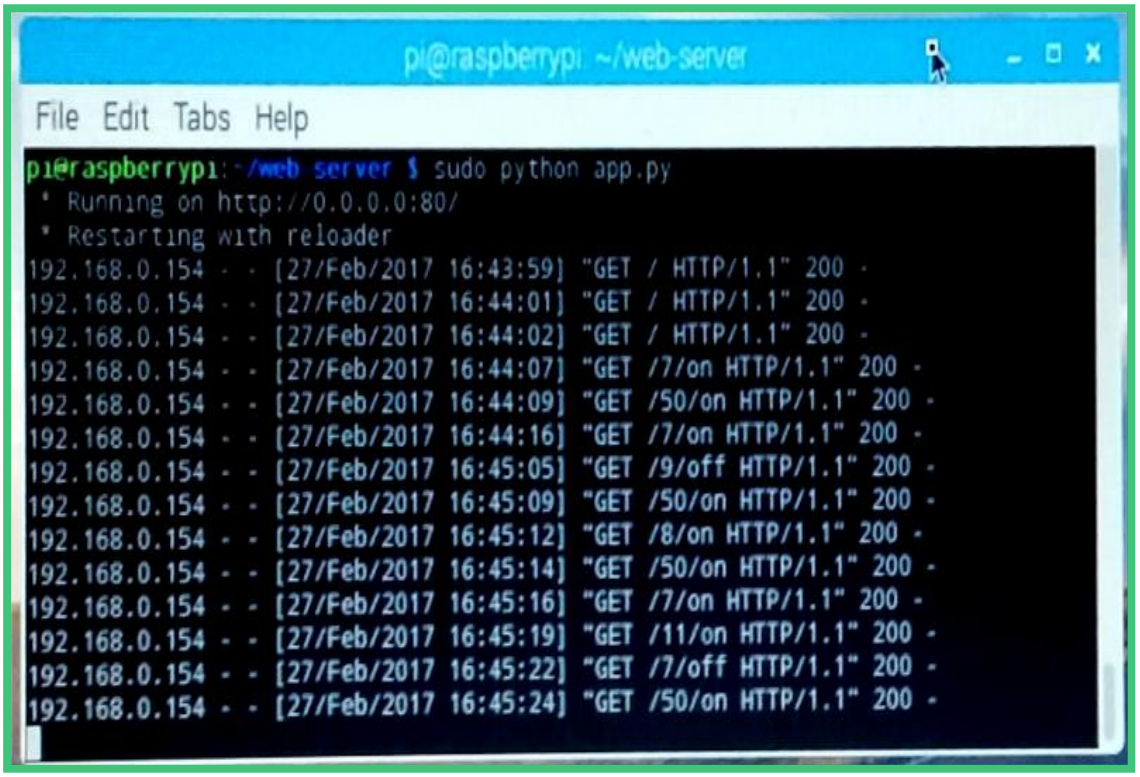



Figure II-36: Data of the web app after making some actions


Figure II-37: Interface created with HTML on a browser

## II-8 Line follower:

Line follower robot is an autonomous robot which follows either black line in white area or white line in black area. Robot must be able to detect particular line and keep following it; also we can say that the line follower robot is a robot which follows a certain path controlled by a feedback mechanism [38].

### II-8-1 TCRT5000 Line follower sensor:

The Line Follower TCRT5000 reflective sensor, is a good add-on for this robot because it gives the ability to detect lines or nearby objects. The sensor works by sending an infrared light by the infrared LED which is the emitter and receiving it by the phototransistor which is the receiver after the reflecting on the white object, also includes a package which blocks visible light. It can detect transitions from light to dark or even objects directly in front of it, but in this project, it will be used for Black line detection [39].

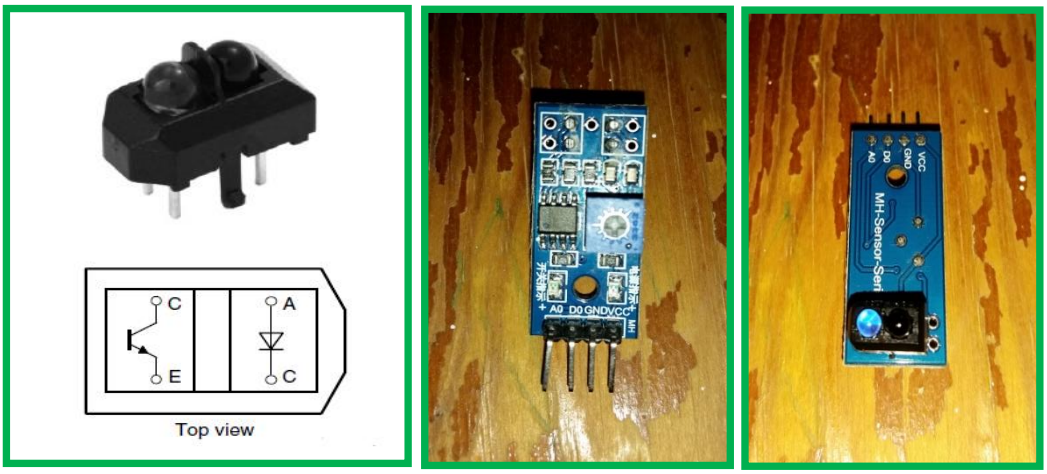



Figure II-38: TCRT5000 Line Follower sensor

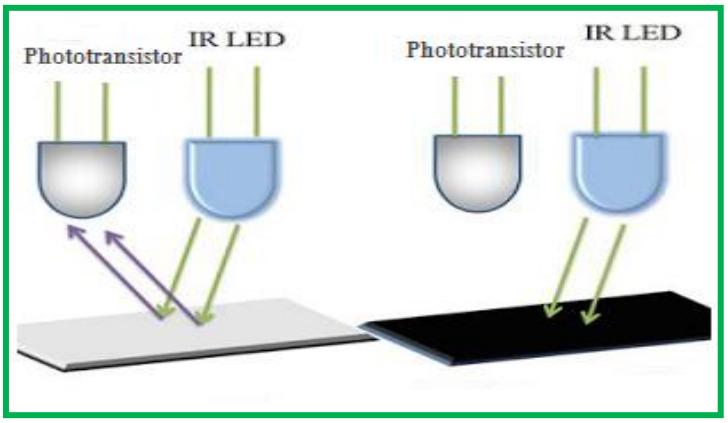



Figure II-39: IR reflection on white floor

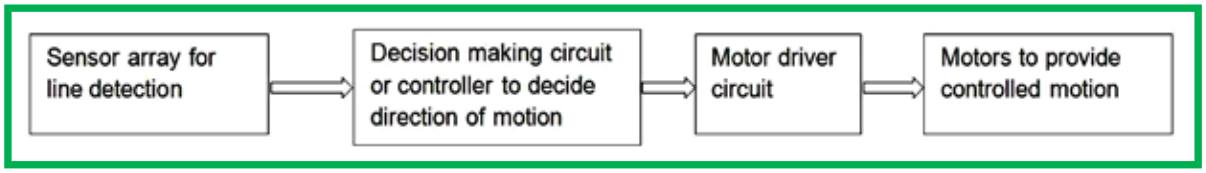



Figure II-40: Block diagram

## ❖ Specifications:

- Detector type: phototransistor
- Emitter type: Infrared LED
- Dimensions (L x W x H in mm): 10.2 x 5.8 x 7
- Voltage supply 5 V
- Current supply 60 mA
- Operating temperature 25 °C

### II-8-2 Number of cells:

The first aspect that affects the precision and the quality of the line follower sensor detection is the number of cells. The line follower sensor that will be used is composed of one cell which composed of a sender and a receiver. Some users may be used only two cells to know whether if the line is at the left or at the right of the robot, but in this robot we will use three sensors which means three cells, this will provide a good detection and precise result with fewer errors [40].

## II-9 Alimentation source:

The robot will be operated using an external alimentation source (Samsung power bank) which contains three USB outputs and each one can provides a supply of 5V ~ 2000mA. Its storage capacity is 20000 mAh which is acceptable for this project due to the multiplicity of devices that will be used on this robot and it can be changed with another source maybe with big capacity to live longer.


Figure II-41: Power bank 20000mAH (Samsung)

## ❖ Conclusion:

This chapter presents the important part and the base of this project which is the electronic card Raspberry Pi where is considered as a master and the Arduino Uno board is considered as a slave. This chapter describes how these two cards can be very useful, effective, enjoyable and very interesting, also the differences between them.

It describes also the actuators at first, where presents the wheeled motors that will be used and clarify its work way. Secondly presents the sensors that will be used, shows its characteristic and its function accurately in how it will make the robot able to do important specific tasks.

The next chapter represents an overview about configuration, programming, which language that will be used and explains its important codes.

# Chapter III: Configuring and Programming

In terms of communication with robots, the programming is the process of developing and implementing various sets of instructions to enable a robot to do a certain task or many tasks, these instructions are considered computer programs and help the computer which is Raspberry pi to apply the orders and realize the tasks exactly and smoothly.

The program is a set of instructions that tells the robot to do various things, it's a directed orders. These instractions differs according to the programming language that will be chosen.

## III-1- Language of programming Python:

Python is a computer programming language that lets you work more quickly than other programming languages. It is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools and libraries.


Professionally, Python is great for web development, data analysis, artificial intelligence, and scientific computing. Many developers have also used Python to build productivity tools, games, robots and desktop apps. So there are plenty of resources to help us learn how to do those as well.

The problem which will be an obstruct in Python is that there is no ensure reacting in the real time where it's a negative point in Python but despite this, Python is really flexible which means there are no hard rules on how to build features, it handles a lot of complexity for us, more forgiving of errors, so we will still be able to compile and run our program until we hit the problematic part and allows beginners to focus on learning programming concepts and not have to worry about too many details. [41]

### III-1-1 GPIO Control:

The Raspberry Pi offers up its GPIO over a standard male header on the board. Over the years the header has expanded from 26 pins to 40 pins while maintaining the original pin out, as we can see in the following figure:

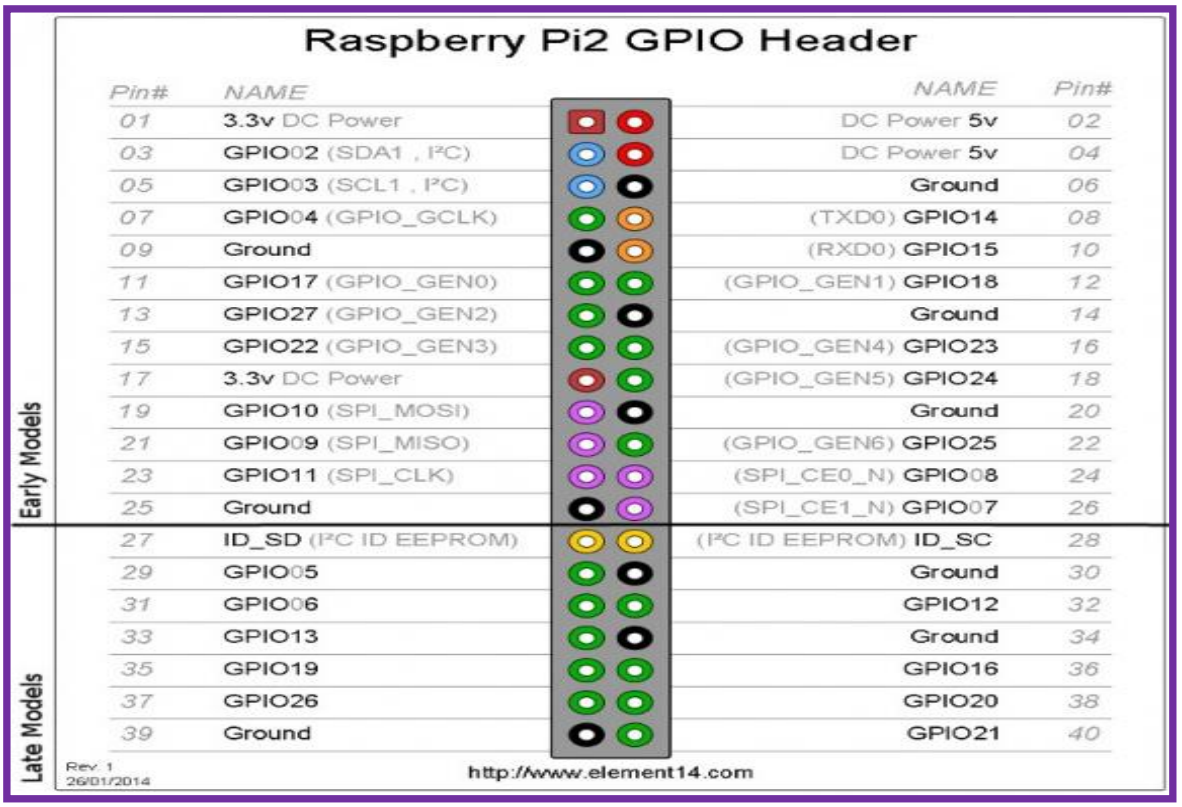



Figure III-1: Raspberry pi GPIO header

### III-1-1-1 Python RPi.GPO module:

We will use the RPi.GPIO module (library) as the driving force behind Python programs; it handles interfacing with the pins. The operating system RASBIAN has the RPi.GPIO library pre-installed, so don't need to download anything to get started.

In order to us RPi.GPIO throughout the rest of Python script, we must put the following line at the top of program lines:

```
import RPi.GPIO as GPIO
```

### III-1-1-2 Pin Numbering Declaration:

We have two pin-numbering schemes:

- **GPIO.BOARD – Board numbering scheme:** The pin numbers follow the pin numbers on header as we see on the previous figure.
- **GPIO.BCM – Broadcom chip-specific pin numbers:** These pin numbers follow the lower-level numbering system defined by the Raspberry Pi's Broadcom-chip brain as we see on the previous figure. [42]

- There is two lines that can be used to specify in our code which number-system is being used, by use "GPIO.setmode ()" function. As we can see the following example:

```
#set up GPIO using BCM numbering
GPIO.setmode(GPIO.BCM)
#setup GPIO using Board numbering
GPIO.setmode(GPIO.BOARD)
```

Figure III-2: GPIO numbering codes

## III-2 Programming of IR receiver:

In order to write the program that makes the robot move according to what programmed it to do when receive the IR signals, will download the LIRC and PYLIRC libraries.

These libraries are mature and stable open source libraries that provide the ability to send and receive IR commands. In this project we will use it just to receive IR commands.

### III-2-1 setting up LIRC on the Raspberry:

- Initially we will install and configure LIRC and PYLIRC and run these libraries on Raspberry Pi, firstly by writing the following two lines in Command line terminal :

```
sudo apt-get install lirc
```

```
sudo apt-get install python-pylirc
```

- We must add this to /etc/modules file:

```
lirc_dev
lirc_rpi gpio_in_pin=23
```

- Adding these two lines means that will use GPIO 23 like a pin-in to receive information from IR receiver.

- We must Change /etc/lirc/hardware.conf file to:

```
######################################################
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS="--uinput"

# Don't start lircmd even if there seems to be a good config file
# START_LIRCMD=false

# Don't start irexec, even if a good config file seems to exist.
# START_IREXEC=false

# Try to load appropriate kernel modules
LOAD_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.
DRIVER="default"
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
######################################################
```

- Now we will restart lircd so it picks up these changes:

```
sudo /etc/init.d/lirc stop
sudo /etc/init.d/lirc start
```

- We must Edit /boot/config.txt file and add these commands:

```
dtoverlay=lirc-rpi,gpio_in_pin=23
```

- Adding this line means that GPIO23 will always used like a PININ to receive information from IR receiver, even after reboot the Raspberry Pi.

- Must reboot the Raspberry Pi after making this change [43].

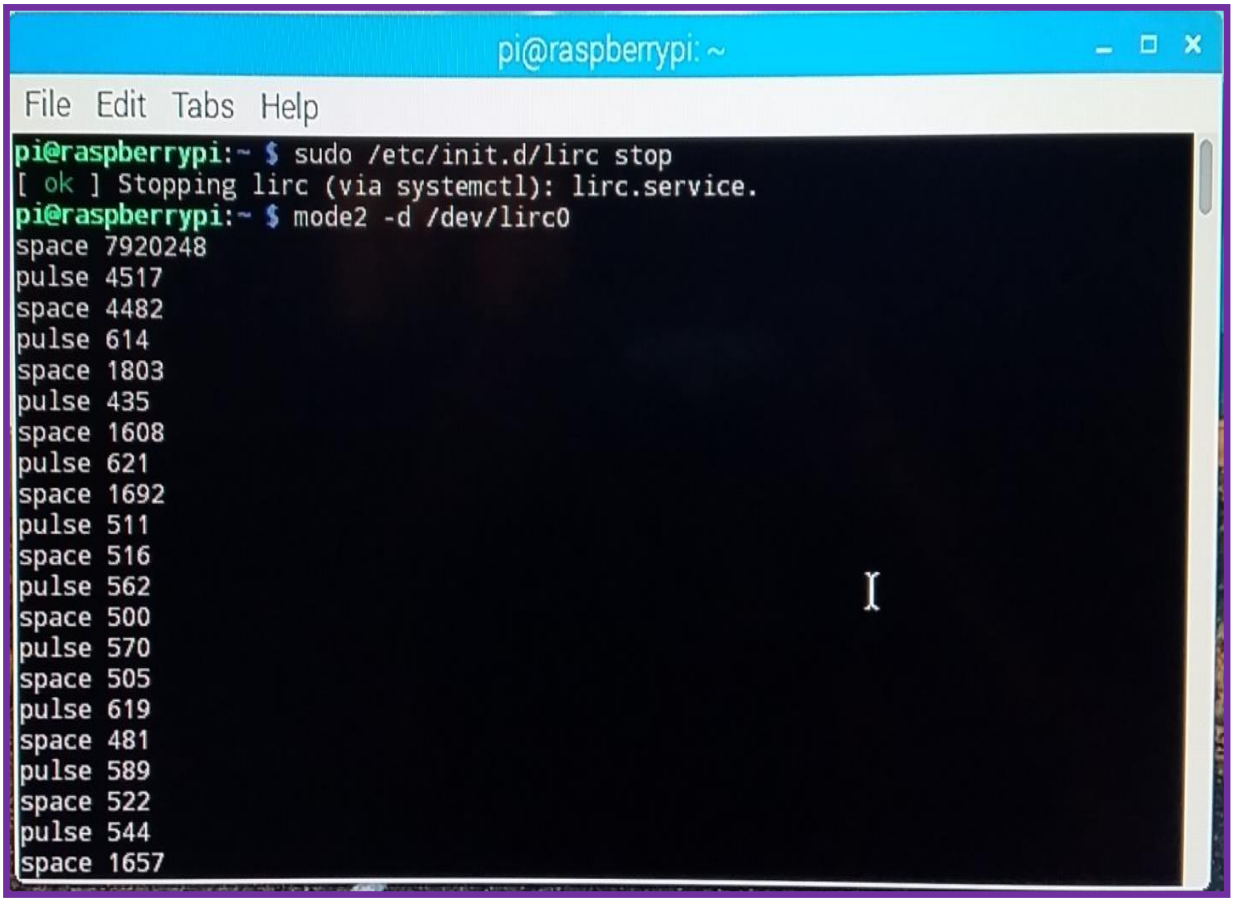
### III-2-2 Testing the IR receiver:

After wiring up the IR receiver (Chapter 04) and reboot the Raspberry Pi.

- We need to run these two commands to stop lircd and start outputting raw data from the IR receiver:

```
sudo /etc/init.d/lirc stop
mode2 -d /dev/lirc0
```

- After this when we will Point the remote control at the IR receiver, pressing some buttons and we should see something like this:



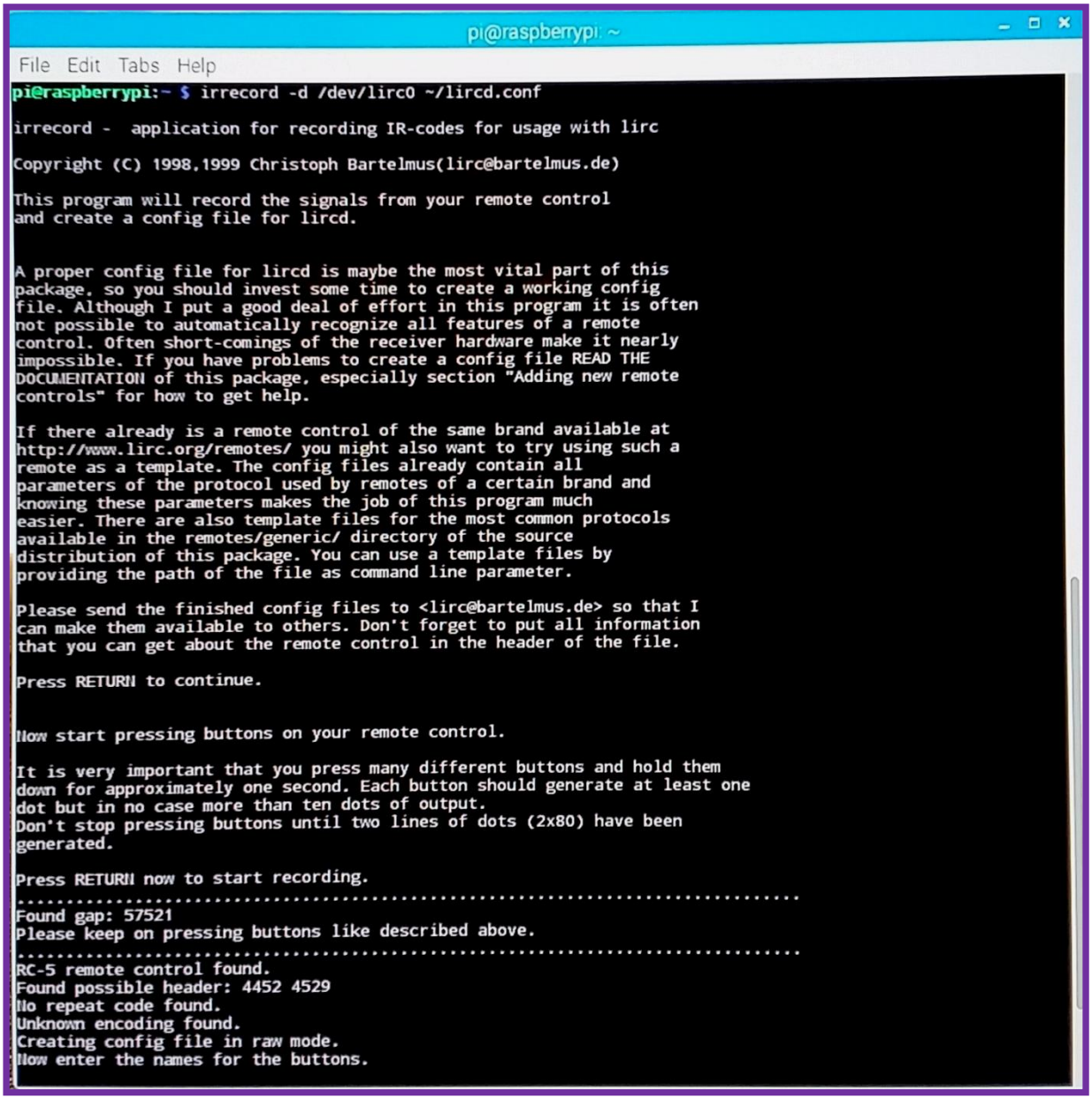

- That's means our IR receiver works and receives the signals correctly.

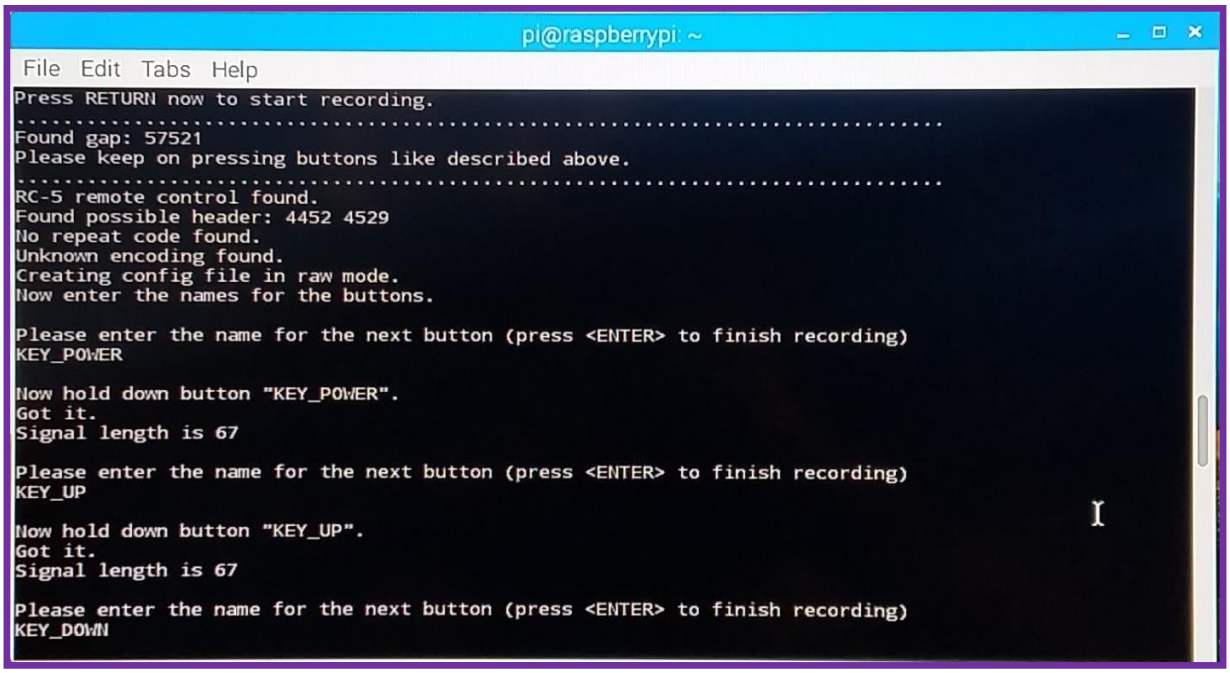
### III-2-3 Record IR codes from your remote:

- We need to create a configuration file and "irrecord" which it will help me to discover the IR codes used by the remote and assist to create this file which will be used by LIRC [44].

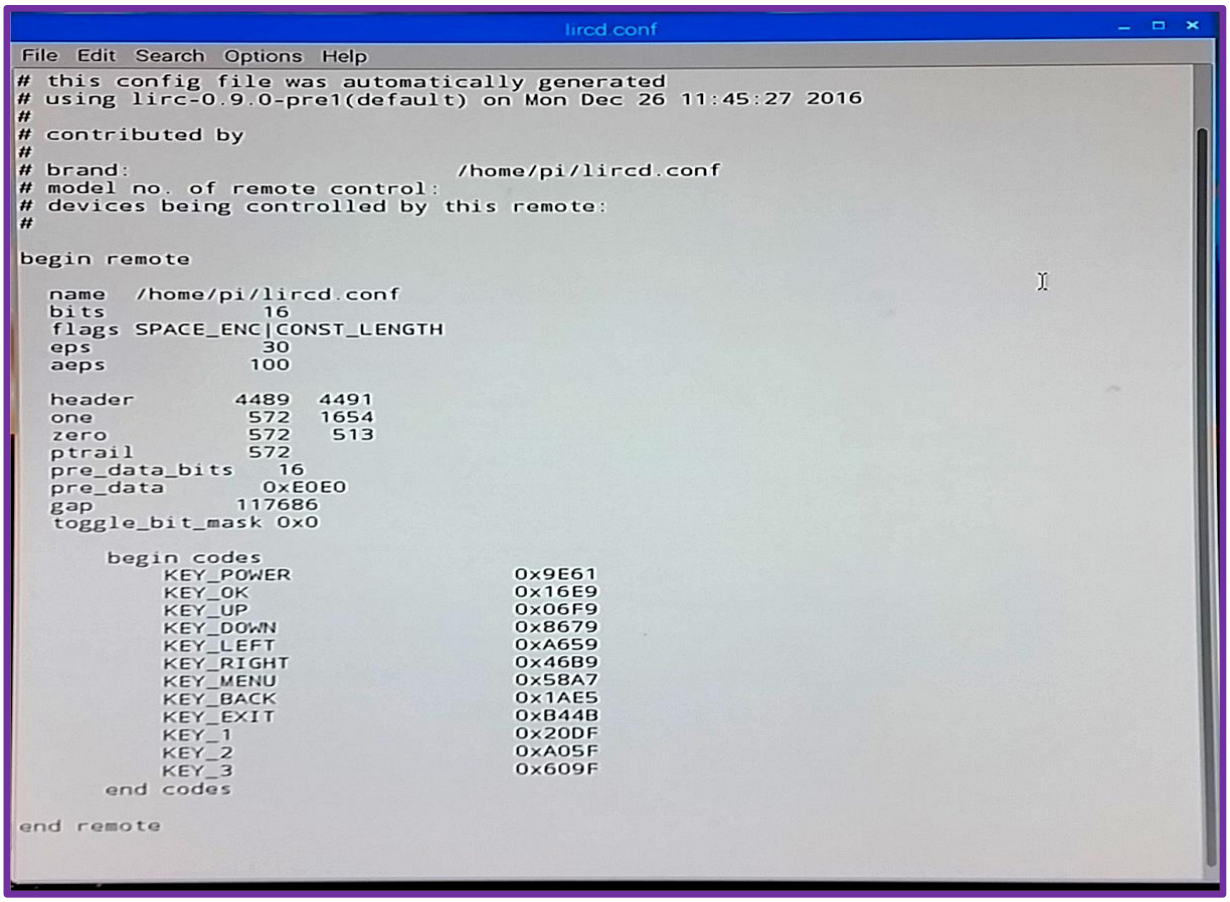
```
pi@raspberrypi ~ $ irrecord -d /dev/lirc0 ~/lircd.conf
```

- After running of this command line, "irrecord" will show detailed instructions on how to setup your remote. As we see in the following lines:
- Now, we will enter the name of each button in remote control:






- Those names will be registered in the configuration file "/home/pi/lircd.conf" like we see in the following picture:

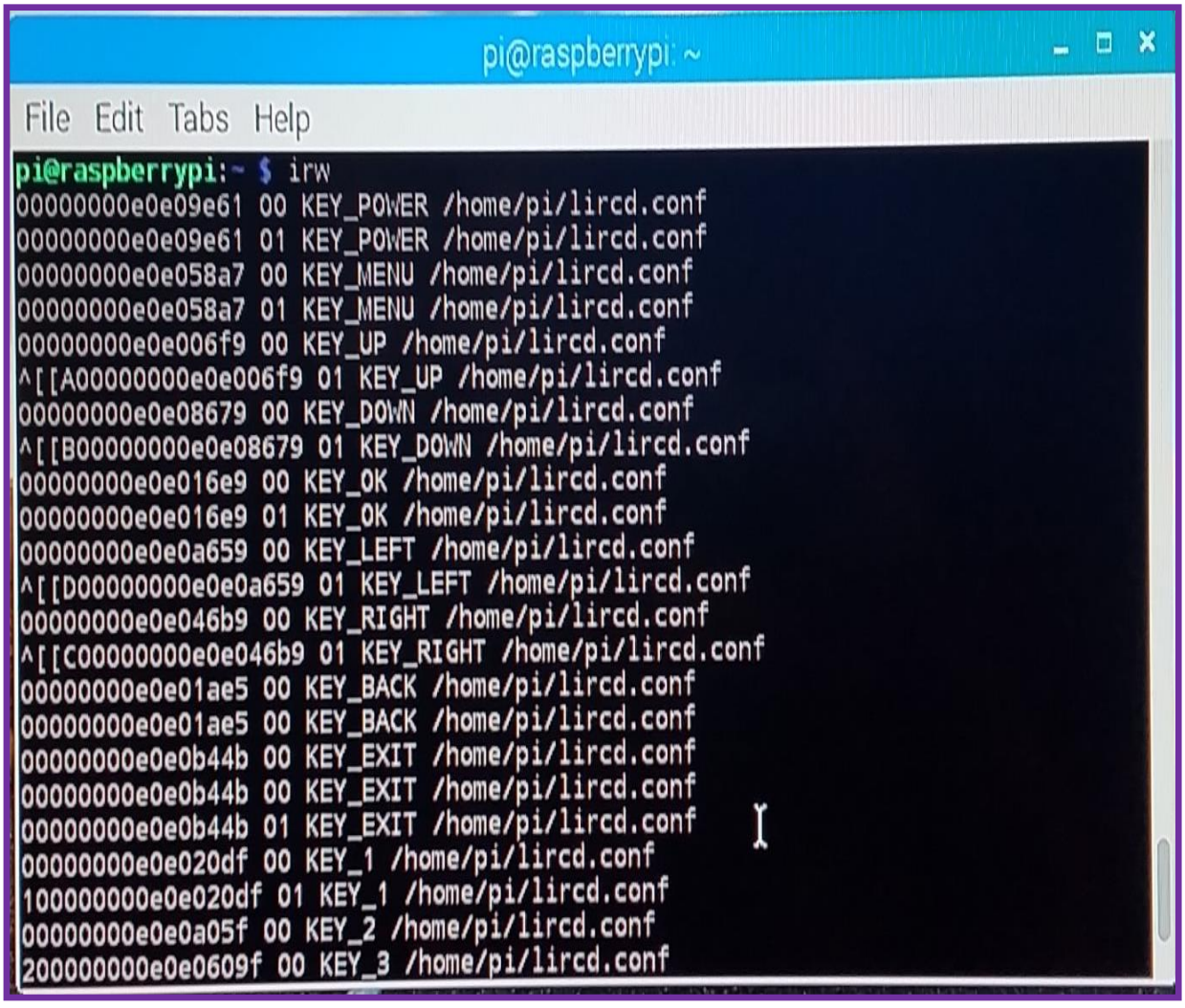


- Now we will replace the existing conf file (which is most likely empty) with the new one which just created and restart LIRC with these following commands :

```
pi@raspberrypi ~ $ sudo cp lircd.conf /etc/lirc/lircd.conf
```

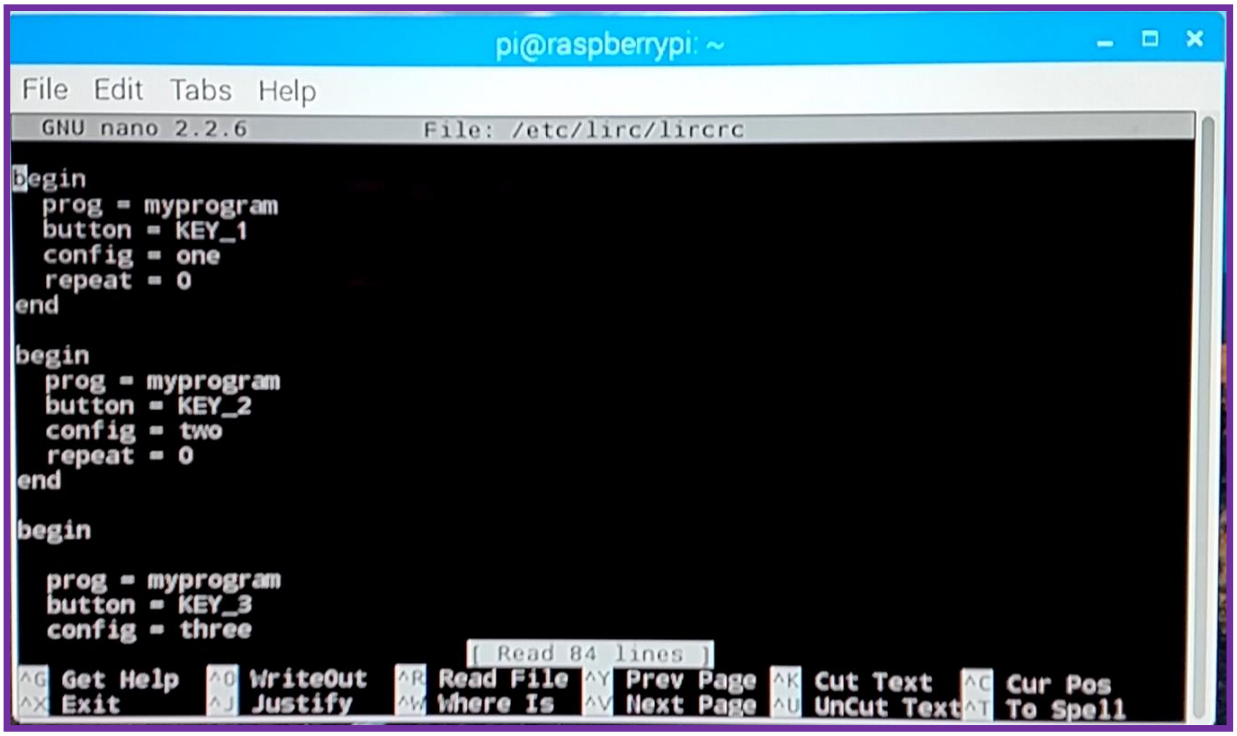
```
pi@raspberrypi ~ $ sudo /etc/init.d/lirc restart
```

- We can test these buttons by running the command “irw” on the command line terminal and press some of these buttons:




- As we can see here, the buttons works correctly.

- The final step is to add some lines to another two configuration files “etc/lirc/lircrc” and “/root/.lircrc” to define the name of the remote and its buttons and we can insert many of remotes if we want to [45]. As follows:




- Finally, now we can write the python codes to control in our robot by IR.

### III-2-4 Python Codes:

The basic codes that will be used in the IR program are:

- Loading the GPIO module into Python script, as follows:

```
Import RPi.GPIO as GPIO
```

- Loading the Pylirc module into Python script, as follows:

```
Import pylirc
```

- Loading the time module into Python script, as follows:

```
Import time
```

- Getting the list of buttons names from configuration file, as follows:

```
Pylirc.init("myprogram")
```

- Using while loop for repeating sections of code, "while true" means running the loop for infinite times, as follows:

```
While True:
```

- The next line is to read the frequencies coming from the IR sender:

```
Pylirc.nextcode ()
```

- The "if" statement is used to conditionally execute a statement or a block of statements. Conditions can be true or false, execute a specific order when the condition is true [45]:

```
if (m == ['mode']) :
      GPIO.output(18, GPIO.HIGH)
```

## III-3 Programming of Ultrasonic (Obstacles avoidance):

### III-3-1 Python Codes:

The basic codes that will be used in the program of obstacles avoidance are:

- Loading the GPIO and time libraries and choosing the pin numbering into Python script, as the following three lines:

```
# Import required Python libraries
import time
import RPi.GPIO as GPIO

# Use BCM GPIO references
# instead of physical pin numbers
GPIO.setmode(GPIO.BCM)
```

- Setting up pins as an output or input:

```
# Set pins as output and input
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)  # Trigger
GPIO.setup(GPIO_ECHO,GPIO.IN)      # Echo
```

- Using while loop for repeating sections of code until a defined condition is met, in case when having True as a condition without break line ensures that the code runs to infinity, as follows:

```
while True:
```

- We will use time instruction for delay the next instruction. We have two seconds as a delay in the following example:

```
  time.sleep(2)                          #Delay of 2 seconds
```

  - In the next case the time instruction will be used between lines to measure time taken to perform a specific task (from start sending signals until receiving it), as we can see in the following lines:

```
while GPIO.input(GPIO_ECHO)==0:
  start = time.time()

while GPIO.input(GPIO_ECHO)==1:
  stop = time.time()
```

- Finally, will calculate the distance between the obstacle and the robot with the following instructions [46], knowing that the speed of sound equal about 34000 cm/s , as follows:

```
# Calculate pulse length
elapsed = stop-start

# Distance pulse travelled in that time is time
# multiplied by the speed of sound (cm/s)
distance = elapsed * 34000

# That was the distance there and back so halve the value
distance = distance / 2
```

## III-4 Programming of Servo Motor:

### III-3-1 Python Codes:

- Loading the GPIO module into Python script, as follows:

```
Import RPi.GPIO as GPIO
```

- Loading the time module into Python script, as follows:

```
Import time
```

- Setting PIN12 or GPIO18 as output pin. We will get PWM output from this pin, as follows:

```
GPIO.setup (18, GPIO.OUT)
```

- After setting the output pin, we need to setup the pin as PWM output pin, as follows:

```
P = GPIO.PWM (output channel, frequency of PWM signal)
```

  - The above command is for setting up the channel and also for setting up the frequency of the Channel". 'P' here is a variable it can be anything. We are using GPIO18 as the PWM "Output channel. "Frequency of PWM signal" we will choose 50, as DXW90 working frequency is 50Hz [47].

- Below command is used to start PWM signal generation. 'DUTYCYCLE' is for setting the 'Turn On' ratio as explained before for example "7.5" as 'DUTYCYCLE', as we can see here:

```
p.start(DUTYCYCLE)
```

- We will use a delay between each movement, we can see a delay of one second in the following example:

```
time.sleep (1)
```

## III-5 Programming of web application:

### III-5-1 Flask:

We will use a Python web framework called Flask to turn the Raspberry Pi into a dynamic web server. Flask is a micro framework for Python based on Werkzeug which is the python WSGI (Web Server Gateway Interface) Utility Library and Jinja2 which is a full featured template engine for Python [48].

- To install Flask we need to have pip installed so we will run the following two commands install pip and flask:

```
pi@raspberrypi ~ $ sudo pip install flask
```

```
pi@raspberrypi ~ $ sudo apt-get install python-pip
```

### III-5-2 Python Codes:

The basic codes that will be used in this program are:

- Loading the GPIO module into Python script, as follows:

```
import RPi.GPIO as GPIO
```

- Loading the GPIO module into Python script, as follows:

```
Import time
```

- Loading the Flask, render template and request modules into Python script, as follows:

```
from flask import Flask, render_template, request
```

- Creating a Flask object called “app” by the following line:

```
app = Flask(__name__)
```

- Running the code below this function when someone accesses the root URL of the, as follows:

```
@app.route("/")
    def hello():
```

- Sending the text "Hello World!" to the client's web browser, as follows:

```
return "Hello World!"
```

- The meaning of the Next two lines is:
  - "if this script was run directly from the command line", as follows:

```
if __name__ == "__main__":
```

  - "Have the server listen on port 80 and report any errors", as follows:

```
app.run(host='0.0.0.0', port=80, debug=True)
```

❖ **Templates:**

If we want to send the browser a site formatted in proper HTML, it doesn't make a lot of sense to put all the HTML code into Python script. Flask uses a template engine called Jinja2 so that we can use separate HTML files with place holders for spots where we want dynamic data to be inserted [48].

- The next lines enable us to create a dictionary of variables (a set of keys) to pass into the template, as follows:

```
templateData = {
   'title' : 'HELLO!',
   'time': timeString
   }
```

- The "main.html" is file that contains the html code. The meaning of the next line is return the "main.html" template to the web browser using the variables in the "templateData" dictionary, as follows:

```
return render_template('main.html', **templateData)
```

### III-5-3 HTML Codes:

- All HTML documents must start with a document type declaration: <!DOCTYPE html> and the HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between, as follows:

```
<tagname>Content goes here...</tagname>
```

  - The HTML **element** is everything from the start tag to the end tag.

- The <head> element is a container for all the head elements and it can include a title for the document, scripts, styles, meta information, and more.
- HTML elements always be nested (elements can contain elements) and all HTML documents consist of nested HTML elements.
- The HTML document itself begins with **<html>** and ends with **</html>**, the **<html>** element defines the whole document. The element content is another HTML element (the <**body**> element).
- The visible part of the HTML document is between **<body>** and **</body>**, the **<body>** element defines the document body. The element content is other HTML elements**.**
- HTML headings are defined between **<h1>** and **</h1>** with the **<h1>** to **<h6>** tags, depending on the size of writing that we want.
- HTML paragraphs are defined between ** ** and ** .**
- **<br>** is an empty element without a closing tag.
- HTML links are defined between **<a>** and **</a>**, as follows:

```
<a href="http://univ-biskra.dz">This is a link</a>
```

  - The link's destination is specified in the href attribute.

- HTML Attributes are used to provide additional information about HTML elements.
  - All HTML elements can have **attributes**
  - Attributes provide **additional information** about an element
  - Attributes are always specified in **the start tag**
  - Attributes usually come in name/value pairs like: **name="value"**

- CSS is a language that describes the style of an HTML document and how HTML elements should be displayed.
- Setting the style of an HTML element, can be done with the **style attribute**, the HTML style attribute has the following **syntax**:

```
<tagname style="property:value;">
```

  - This **property** is a CSS property and the **value** is a CSS value.

- The **background-color** property defines the background color for an HTML element.

```
<body style="background-color:powderblue;">
```

- The **color** property defines the text color for an HTML element:

```
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

- Using the HTML **border** attribute allows us to make a quick border around the table but when the value equal zero `<TABLE BORDER="0">`, it will be no border around which make the buttons horizontally in the same line, as follows:

```
<TABLE BORDER="0">
<TR>
<TD><FORM>Button1</TD></FORM>
<TD><FORM>Button2</TD></FORM>
</TR>
</TABLE>
```

- The  tag defines a division or a section in an HTML document, is used to group block-elements to format them with CSS, as follows:

```
<div style="color:#0000FF">
  <h3>This is a heading</h3>
  <p>This is a paragraph.</p>
</div>
```

- Django's template language is designed to strike a balance between power and ease; it's designed to feel comfortable to those used to working with HTML.
- The **{% if %}** tag is for boolean tests, it's a tag of Django's template language, it evaluates a variable and if that variable is "true" the contents of the block are displayed, as follows:

```
{% if pin[7].state == true %}
<p> LED is ON </p>
{% else %}
<p> LED is OFF </p>
{% endif %}
```

- The class attribute is mostly used to point to a class in a style sheet, it will be used for chose the display style of buttons, as the following examples:

```
<button type="button" class="btn btn-primary"> HOME </button>
<button type="button" class="btn btn-default"> HOME </button>
```

The **<iframe>** tag specifies an inline frame. The inline frame is used to embed another document within the current HTML document. This tag will be used to adding the camera document to the web app interface to have a live viewing, by adding the IP address of this camera which appears inside the following line [49]:

```
<iframe src="http://192.168.43.1:8080/browserfs.html">
</iframe>
```

## III-6 Arduino Uno configuration on Raspberry Pi:

- We can easily emerge Arduino Uno with Raspberry Pi by "Nanpy" tool following these steps, as follows:

- Firstly, we will download the zipped folder which contains all the required programming documents and libraries, by executing the following lines on terminal line command:

```
wget https://pypi.python.org/packages/source/n/nanpy/nanpy-v0.8.tar.gz
```

- The following line will be used unzip the folder:

```
tar xvf nanpy-v0.8.tar.gz
```

- We will enter the extracted folder to install Nanpy, as follows:

```
cd nanpy-0.8
sudo python setup.py install
```

- Now we must download the Nanpy Firmware form the browser of Raspberry Pi, as follows:

```
https://github.com/nanpy/nanpy
```

- We need just the Firmware folder from the downloaded file which we will add it to the first unzipped Nanpy File after deleting the current one.

- After this we will enter to the Firmware file to display all kinds of Arduino boards that supported from Nanpy environment and to choose Arduino Uno, by the following lines:

```
cd firmware
make boards
```

```
Available values for BOARD:
uno          Arduino Uno
atmega328    Arduino Duemilanove w/ ATmega328
diecimila    Arduino Diecimila or Duemilanove w/ ATmega168
nano328      Arduino Nano w/ ATmega328
```

```
export BOARD = UNO
make
make upload
```

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.02s

avrdude: Device signature = 0x1e9801
avrdude: reading input file "Nanpy.hex"
avrdude: writing flash (31174 bytes):

Writing | ################################################## | 100% 4.66s

avrdude: 31174 bytes of flash written

avrdude: safemode: Fuses OK

avrdude done.  Thank you.

pi@raspberrypi ~/nanpy-0.8/firmware $
```

- As we can see above the Arduino Uno has configured successfully and the programming will by Python [17][50][51].

## III-7 Programming of Line Follower Mode:

### III-7-1 Python Codes:

- Loading the GPIO module into Python script, as follows:

```
Import RPi.GPIO as GPIO
```

- Loading the time module into Python script, as follows:

```
Import time
```

- Setting GPIO X as an input pin. We will get the D0 information from this input pin, as follows:

```
GPIO.setup (X, GPIO.IN)
```

- The "if" statement is used to conditionally execute a statement or a block of statements. Conditions can be true or false and executing the specific order will be when the condition is true [17]:

```
If (GPIO.input(X) ==True):
      Print "Black"
```

## ❖ Conclusion:

This chapter represents an overview about configuration of IR receiver and Arduino Uno, the programming language HTML and the programming language Python and how the robot does its tasks step by step depending on this configuration and Python codes.

Python is easy programming language and very enjoyable in this domain, finding examples can be easy too with Python because it's famous and supported by a lot of programmers around the world.

The next chapter which is the final one represents a practical overview, where all the tasks that this mobile robot can do will be applied and discussed.

# *Chapter IV: Application and Discussion*

Robots are used in all life aspects: the medical, the manufacturing as the auto industries, military and all manner of exploration vehicles as space exploration. Building and programming, a robot is a combination of mechanics, electronics, and problem solving aspects. What we are about to learn while doing the activities in this project will be relevant to real-world applications that use robotic control, the only differences being the size, sophistication, principles, program listings and the circuits you will use although sometimes the same as industrial applications developed by engineers.

The goal of this project is to get students interested in and excited about the fields of engineering, mechatronics, and software development as they design, construct, and program an autonomous or manual-controlled robot.

The way that will be chosen to get students interested in this domain is by doing lab sessions; this part contains series of hands-on activities that introduce students to basic robotic concepts using our pedagogic mobile robot "Foks-Bot", which is the experimental mobile robot that we will build and we have talked about it in the previous chapters. Its name comes from "Friend of knowledge seeker", and we will see how and what it was designed for [31].

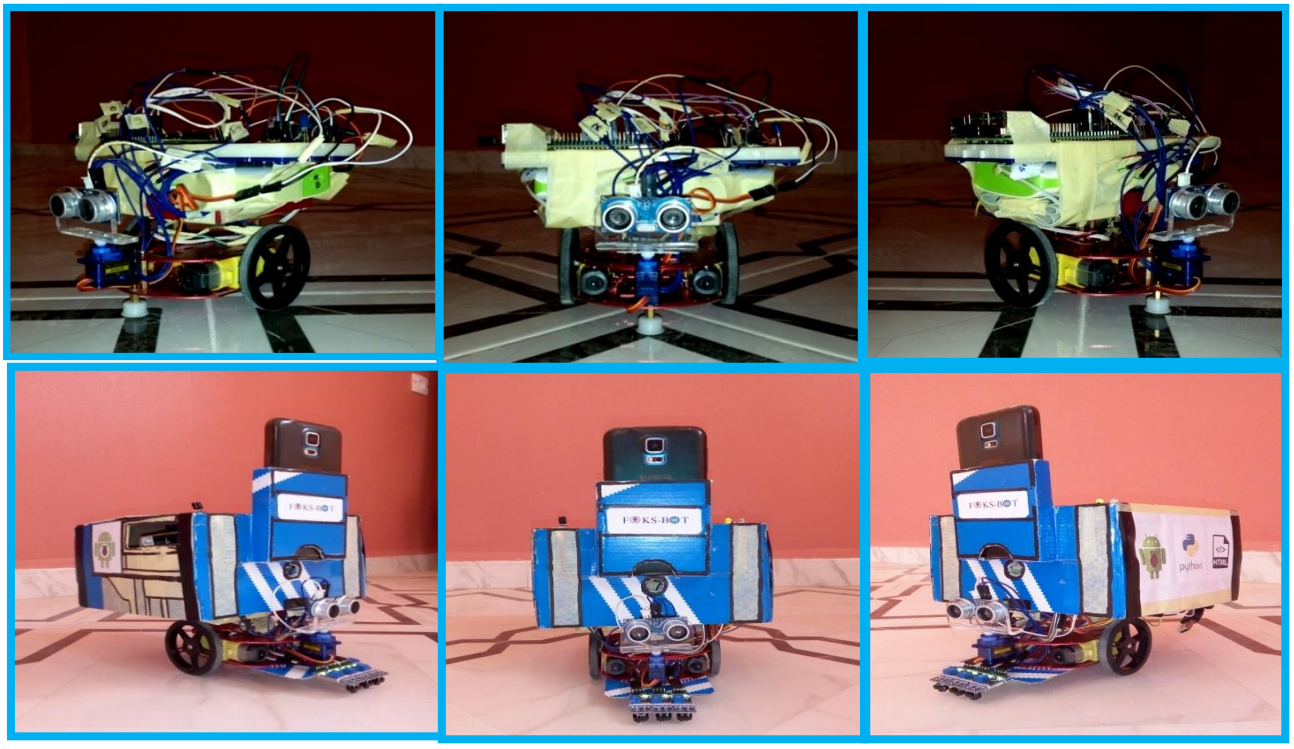



Figure IV-1: Foks-Bot

# First Lab Session

## Control by Infrared

The student can use infrared signals to control Foks-Bot " wireless control " by providing an IR sensor (IR receiver), connecting it and finally programming raspberry pi which is the brain of Foks-Bot to receive and decode the signals coming from the remote control (IR sender). The way it functions has been explained in the second chapter. The infrared can also be used to detect obstacles, which gives the ability to make decisions to avoid them [28].

### IV-1-1 Objectives:

- Wiring up the IR receiver.
- Configuring and programming the IR receiver on Raspberry pi.
- Testing and discussing IR control.

### IV-1-2 Parts list:

- Raspberry pi.
- Robot chassis with two controlled wheels by Motors driver.
- IR receiver.
- IR sender (IR remote control).
- Connecting wires Male/Female.

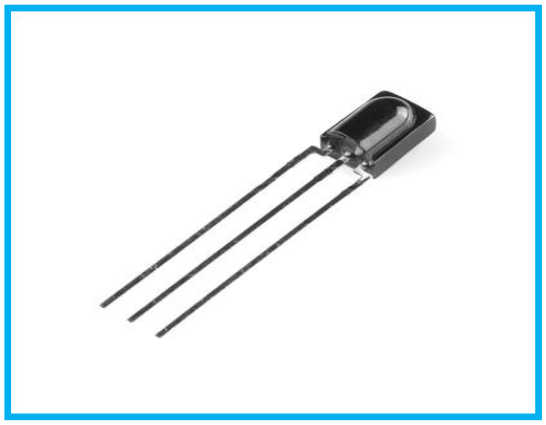


Figure IV-2: IR receiver (TSOP38238)

Figure IV-3: Robot chassis

### IV-1-3 Task 01: **Wiring up the IR receiver**

In this activity we will connect the IR receiver to Raspberry Pi, which is very easy as there are only 3 pins on the sensor: GND, Vs and Output. We will connect the Output to GPIO 23, Vs to 5V (5V pin or External alimentation source) and GND to Ground pin. As we can see in the following figure:

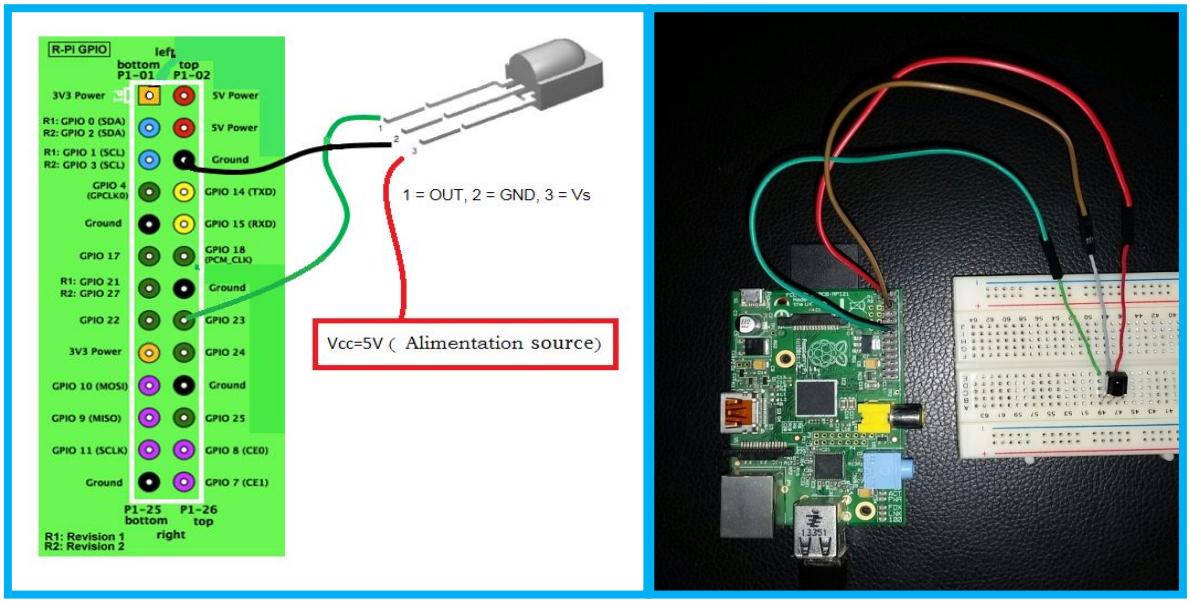



Figure IV-4: Connecting TSOP382 IR receiver on Raspberry pi

### IV-1-4 Task 02: **Configuring and programming IR receiver on Raspberry pi**

In this activity we will configure Raspberry pi to receive and decode the IR signals by IR receiver and we will made a program to control in Foks-Bot (Direction control) by sending specific IR signals from the IR sender (TV, DVD,AC remote control…) that we have chosen in configuration part. In my case I choose my smart phone Samsung Galaxy-S5 which contains an IR sender.

#### IV-1-4-1Infrared Sender:

In this project we will use a smart phone (Samsung Galaxy S5) as a remote control, because it contains an IR sender (infrared source), where we will use the application "Peel Smart Remote" which contains many remotes. We will then choose a certain remote to make the programming. As we can see in the following figures:

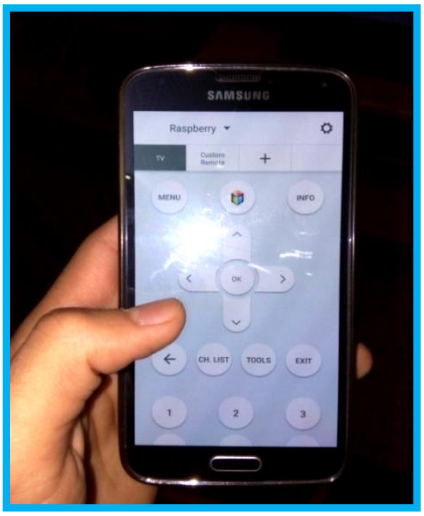



Figure IV-5: Peel Smart Remote interface


Figure IV-6: Phone IR sender

When we point the remote control at the mobile robot and we push a certain button, it makes the infrared emitter run and encode a messagein the form of a signal. The IR receiver picks up this signal, decodes the message, and does whatever the message directs: goes forward/backward, turns right/left or stops.

### IV-1-4-2 Configuration:

We have configured the raspberry pi in the previous chapter (Chapter 03), as we can see the final step of configuring where the buttons are well-known by raspberry pi in the following figure:

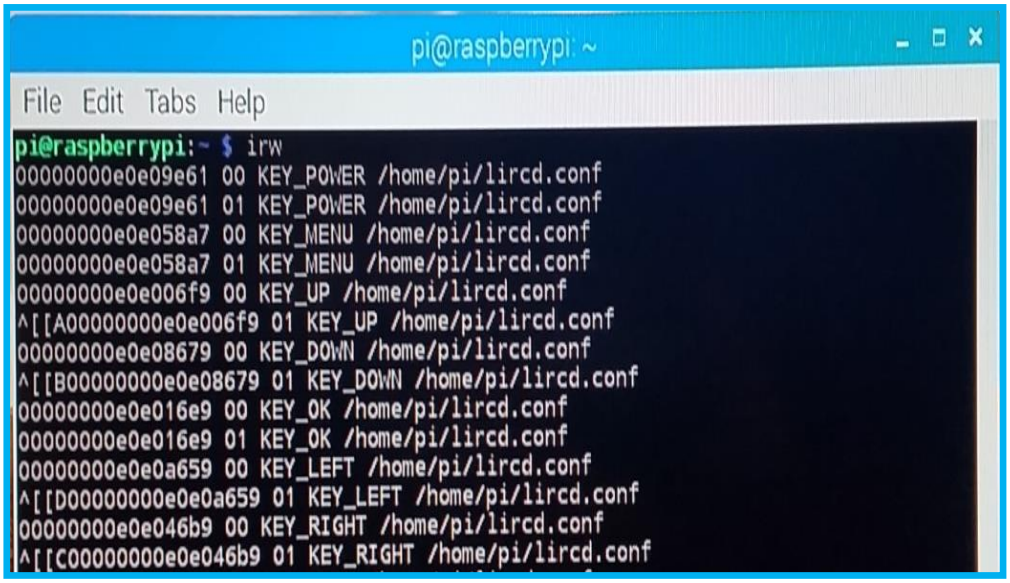



Figure IV-7: Testing the buttons that already configured

### IV-1-4-3 Designing the organigram of python program:

We will use the instructions mentioned in the previous chapter (Chapter 03) to create the python program (Appendix) after designing its organigram follows:

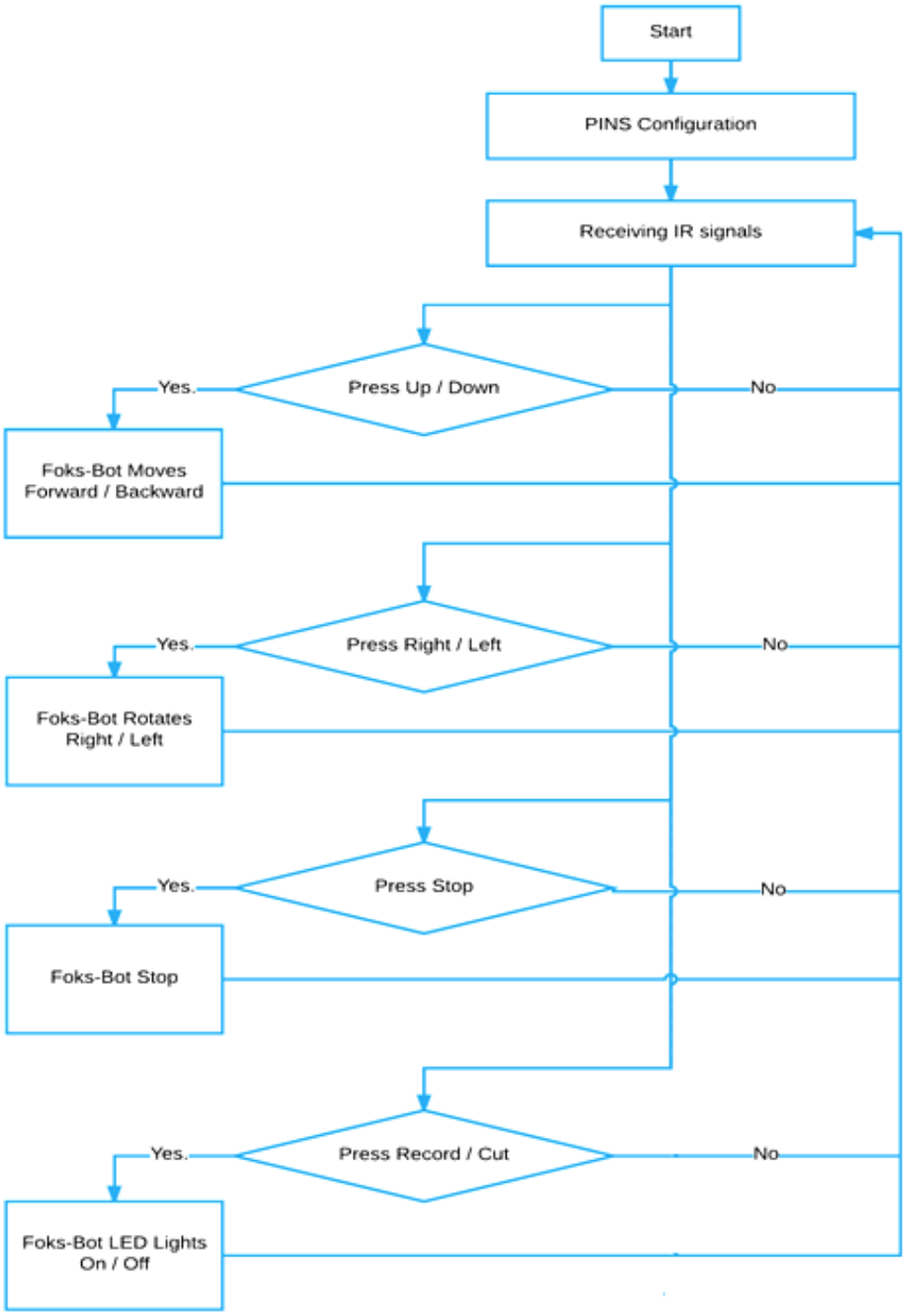



Figure IV-8: Organigram of IR control

## IV-1-5Task 03: Testing and discussing the IR control

### IV-1-5-A Testing the IR control:

When we point the remote control at Fosk-Bot robot and press a button, it makes the infrared light source run and encode a message on it in the form of signals. The IR receiver picks up these signals, decodes the message, and does whatever the message directs:moving forward/ backward, turning right/ left,stop and turn the diode led on/off, as we can see in the following figures:

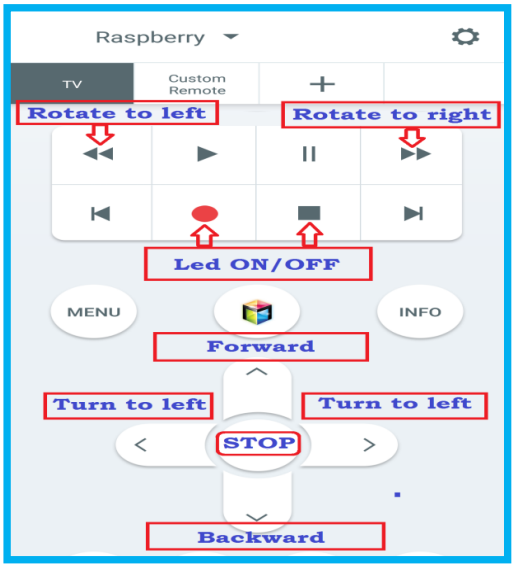



Figure IV-9: Tasks of remote control buttons

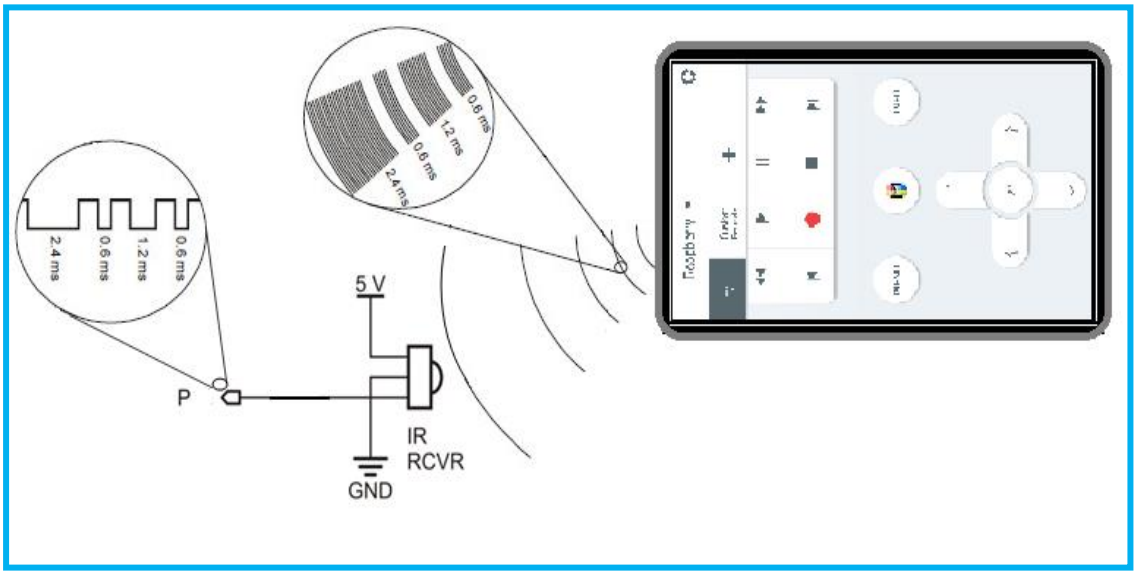



Figure IV-10: Sending and receiving signals


Figure IV-11:Area of experiments and the led lightis off

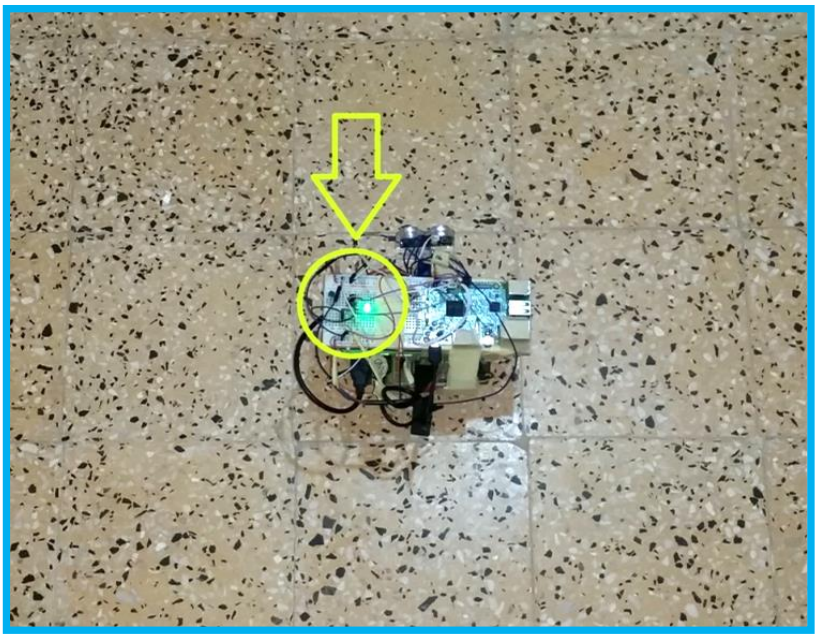


Figure IV-12: Lighting the green LED


Figure IV-13: Foks-Bot moves forward/ backward


Figure IV-14: Foks-Bot rotates 90º and moves to right

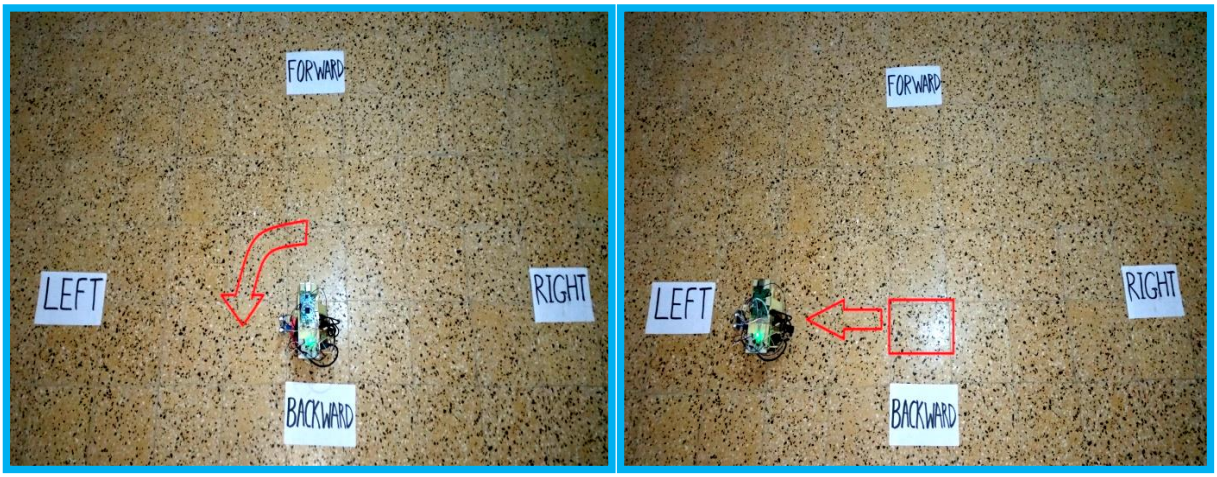



Figure IV-15: Foks-Bot rotates 90º and moves to left

### IV-1-5-A Discussing the IR control:

#### ❖ Advantages:

- IR control is a wireless control.
- Recommended operating conditions are 5V and a very low current supply 3mA, which means very low power consumption.
- Inexpensive as IR receiver are available everywhere (TV, DVD-Player, MP3-Player, Wall fan, TV channels receiver, etc.).
- Transmission distance is 45m and Directivity is 45°, which is good for a local mobile robot.
- We can use IR to detect obstacles (Chapter 02) as an improvement for the obstacles avoidance skill, which means using it with ultrasonic to achieve the best results.

#### ❖ Disadvantages:

- Existence of an obstacle between the IR emitter and receiver could hamper the control on Foks-Bot.
- Very short delay at responding because there is no timer in raspberry pi which means no tempo real (Real-time response) or maybe it is caused by the weakness of IR LED light (IR emitter).

# Second Lab Session

## Autonomous Control by Ultrasonic

Autonomous control means that Foks-Bot will have a relative independency in making its decisions with regard to taking trajectories. The feature encapsulates in the fact that Foks-Bot can avoid the obstacles while moving and choosing its path.

HC-SR04 Ultrasonic sensor is the mean that gives the ability to know whether or not there is an obstacle ahead of the robot. Supported with servo motor DXW90 for directing the sensor, its working method is by sending sound waves and receiving them when they bounce back after hitting the obstacle, its working method has been explained in the second chapter [32].

### IV-2-1Objectives:

- Wiring up ultrasonic sensor.
- Programming the ultrasonic sensor on Raspberry pi.
- Testing and discussing the autonomous control.

### IV-2-2 Parts list:

- Raspberry pi.
- Robot chassis with two wheels controlled by Motors driver.
- Ultrasonic sensor HC-SR04.
- HC-SR04 support.
- Male/Female connecting wires.
- Resistors 330 Ω ohm and 470 Ω.
- Breadboard.

Figure IV-16: Ultrasonic sensor HC-SR04

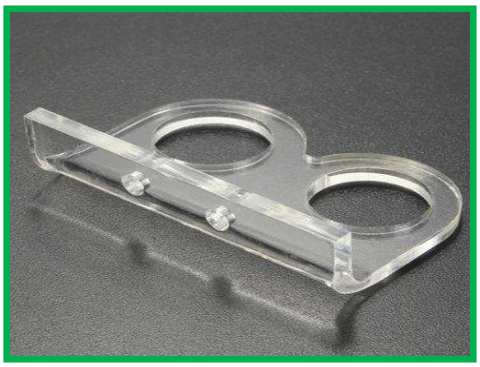


Figure IV-17: Support of HC-SR04

### IV-2-3 Task 01: **Wiring up the Ultrasonic sensor**

In this activity we will connect the ultrasonic sensor to Raspberry Pi, as follows:

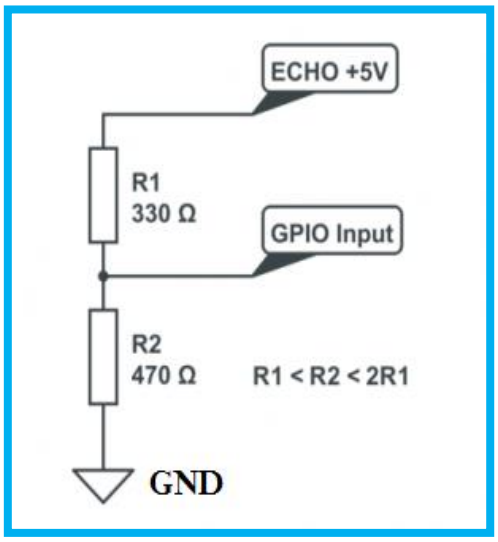
### IV-2-3-1 Connection Diagram:

We willconnectHC-SR04 ultrasonic sensor to Raspberry Pi as follows:

- VCC to Pin 2 (5V).
- GND to Pin 6 (GND).
- TRIG to Pin 13 (GPIO27).
- We connect the 330Ω resistor to ECHO. On its end we connect it to Pin 15 (GPIO22). Through a 470Ω resistor, we also connect it to Pin 6 (GND).

The ECHO output is 5v. The input pin of Raspberry GPIO is rated at 3.3v. So 5v cannot be directly given to the unprotected 3.3v input pin. Therefore we use a voltage divider circuit using appropriate resistors to bring down the voltage to 3.3V.

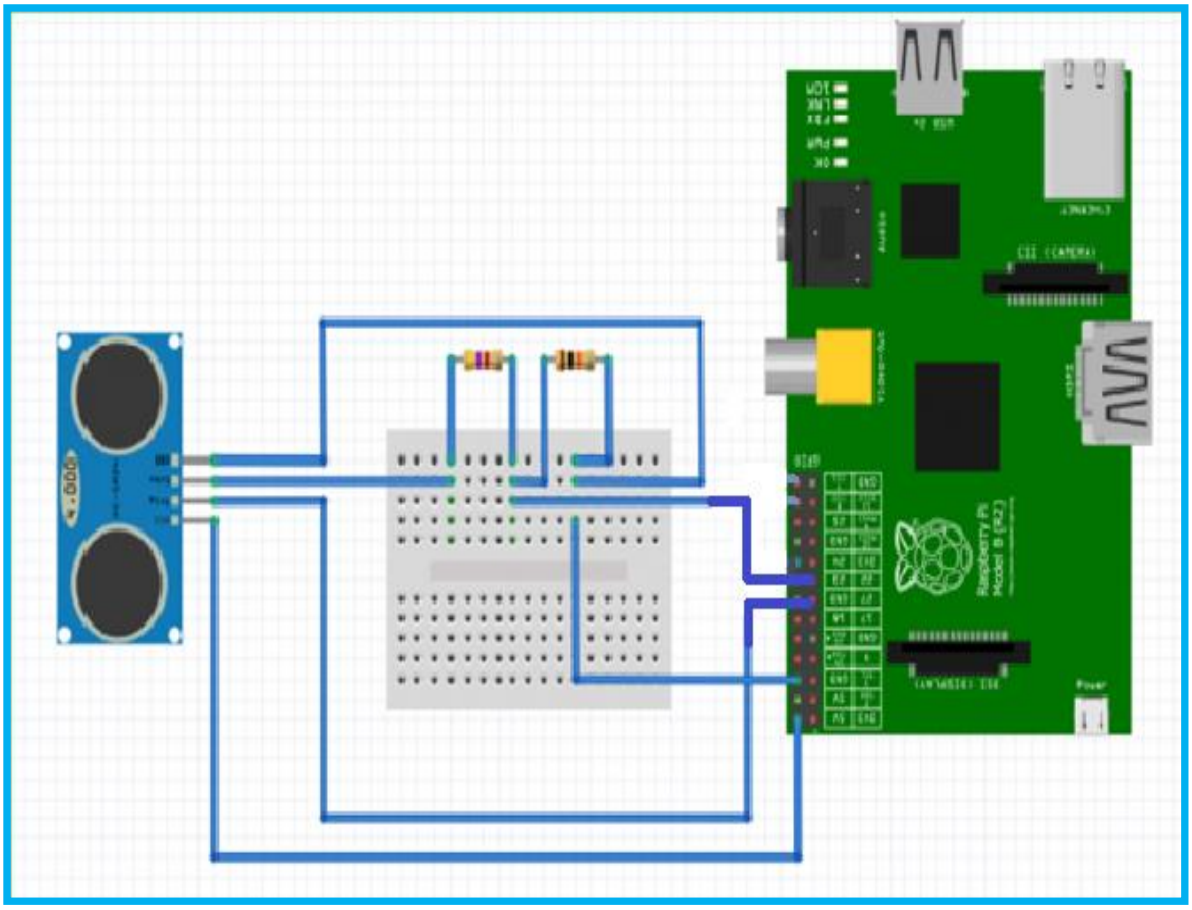
Ideally R2 must be between R1 and the double value of R1. In this exemplary circuit, we used 330 Ω and 470 Ω resistors, as follows (Figure 4-18) [33].




FigureIV-18: 5V to 3.3V Voltage Divider

- The following equation can be used to calculate the out voltage values:

$$V_{out} = V_{in} \times R2 / (R1+R2) = 5 \times 470 / (330+470) = 2.93\ V \quad \text{(IV.2.3)}$$



FigureIV-19: Connecting HC-SR04 Circuit with Raspberry Pi

### IV-2-4 Task 02: **Programming ultrasonic on Raspberry pi**

In this activity we will create a program to realize the autonomous control, which means that Foks-Bot will take control on itself. It is a direction control with feature of automatic obstacles-avoidance and decision-making about the right path. We will use the instructions mentioned in the previous chapter (Chapter 03) to create the python program (Appendix), and we will start by designing an organigram for this program, as follows:

### IV-2-4-1 Designing the organigram of python program:

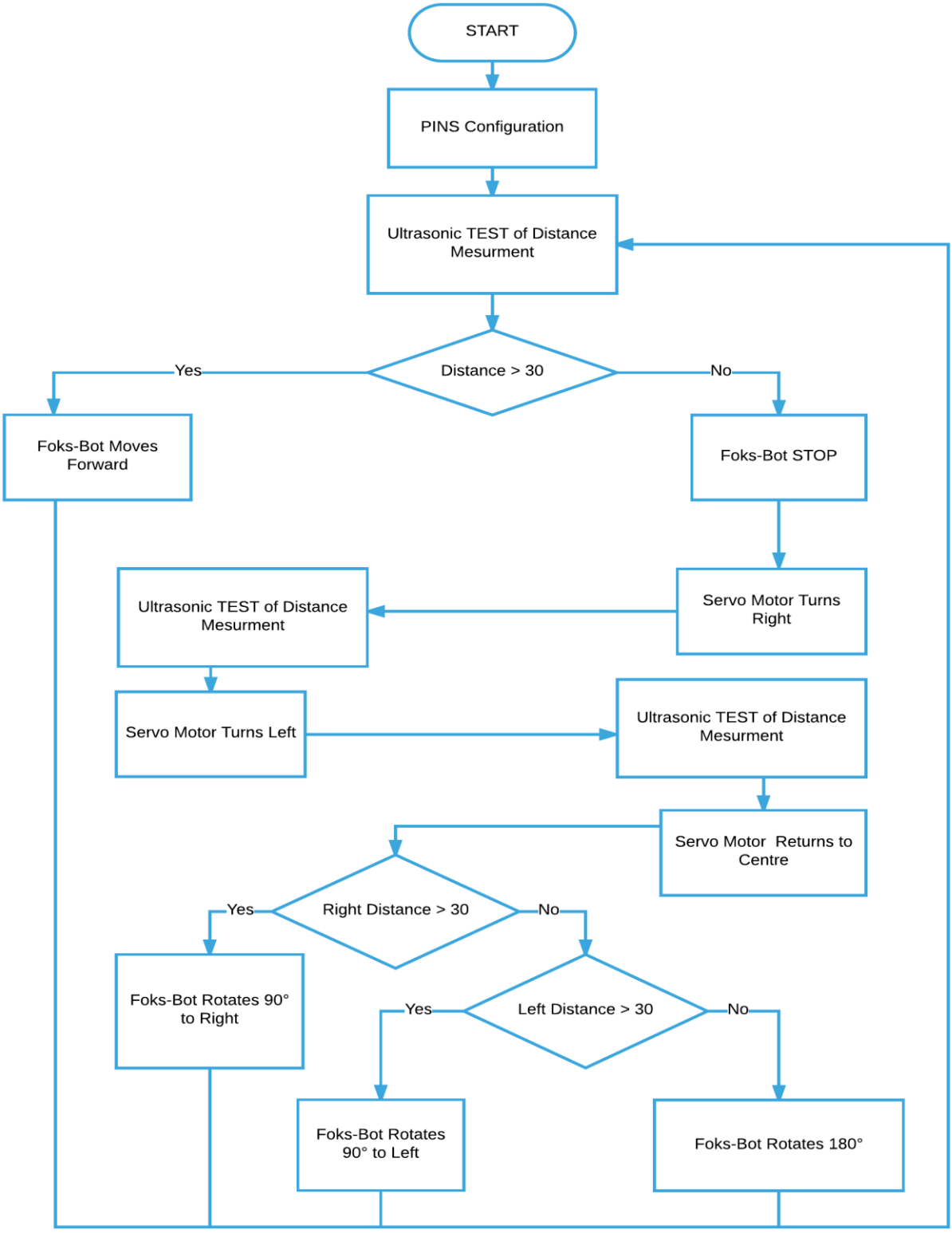



Figure IV-20: Organigram of the autonoumous control by Ultrasonic

### IV-2-5 Task 03: Testing and discussing the Autonomous Control

#### IV-1-5-A Testing the Autonomous control:

When we activate the autonomous control mode we will notice Foks-Bot moving randomly. Given that this is not the purpose, we will see it avoid the obstacles automatically and chose the right trajectory, as we can see in the following figures:

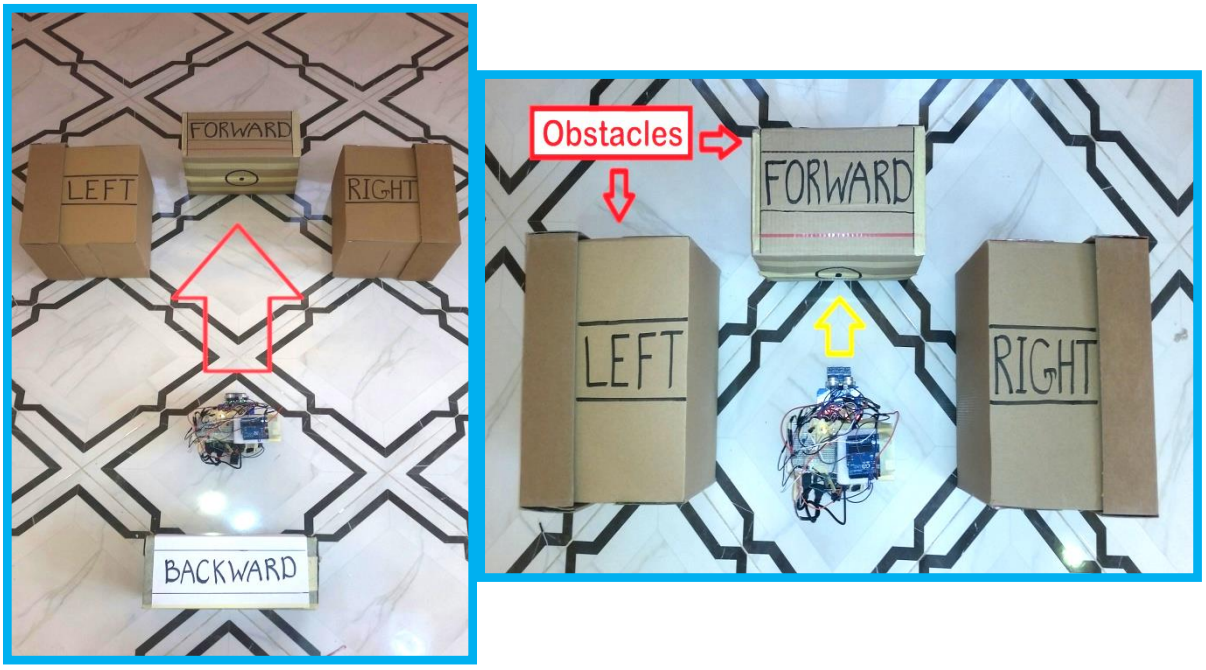



Figure IV-21:Foks-Bot moves Forward and detects an obstacle


Figure IV-22: Servo motor Rotates 90º and Ultrasonic detects obstacles Right/Left


Figure IV-23: Foks-Bot Rotates 180ºthen moves Forward

### IV-1-2-A Discussing the Autonomous control:

#### ❖ Advantages:

- Work is by signals, which means no physical contact exists, unlike sensitive whiskers which work by physical contact, so this will be a good option in many life aspects.
- Recommended operating conditions is 5V, and the current supply is very low at 15mA, which implies very low power consumption.
- Transmission distance is 4m, and Directivity is15°, which is acceptable compared to the good IR characteristics.
- Using the DXW90 servo motor to realize control in direction of the sensor (left/right) is better than directing the whole robot.

#### ❖ Disadvantages:

- Sometimes, the sensors cannot detect the obstacles if the obstacle shape is random or if it contains a lot of bends or corners

- Four meters is a small detection distance, in case we want to program Foks-Bot to do a navigation process.
- The sensors can't detect the obstacles if the obstacle face is flat but not parallel to it, and if it forms an obtuse angle with the sensor, as the follows figure:

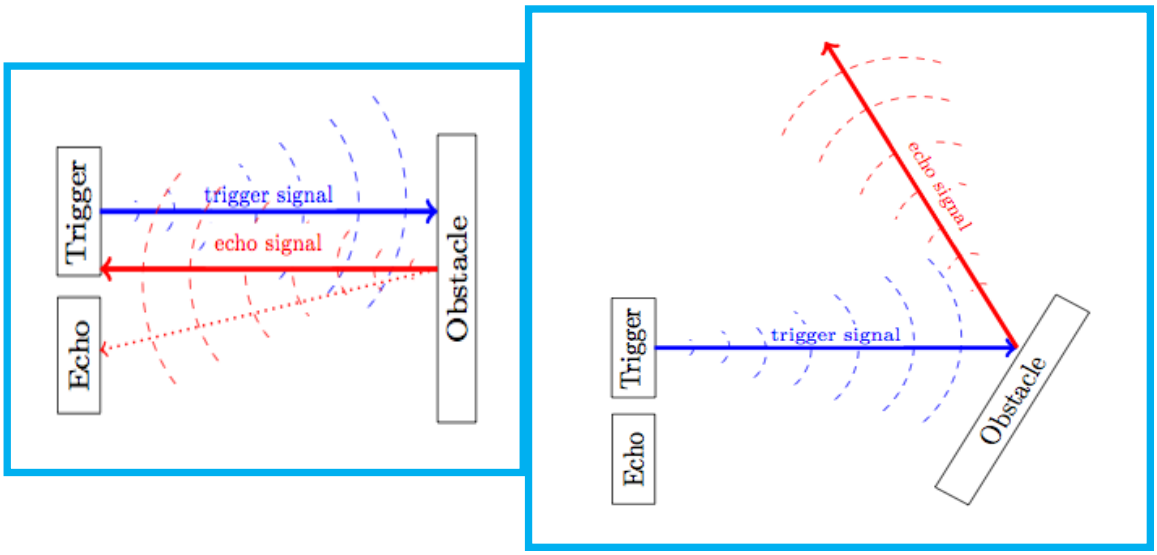



Figure IV-24: The Ultrasonic blind detection angle

- We can solve this issue with more Ultrasonic sensors, three sensors is good, one in the middle and two on both sides.

# Third Lab Session

## Control by Wi-Fi

In the first lab, we used session IR to control Foks-Bot. In this one, we will use Wi-Fi to control it. Wi-Fi uses radio waves to transmit information across a network (see chapter two).

Wi-Fi is better than IR in all aspects, signals transition speed, control in range with less worry about obstacles and the abundance of information about the status of our robot. All this will be realized firstly by providing a Wi-Fi adapter on raspberry pi, secondly by creating a web application, and finally by testing this web app and discussing Foks-Bot's interactions [36].

### IV-3-1 Objectives:

- Creating a web application.
  - Designing an Interface with HTML.
  - Creating a Python Program.
- Testing and discussing the web app and Foks-Bot interactions.

### IV-3-2 Parts list:

- Raspberry pi.
- Robot chassis with two wheels controlled by Motors driver.
- Wi-Fi adapter for raspberry pi 1 and 2, but raspberry pi 3 contains a Wi-Fi by default.

Figure IV-25: Wi-Fi Adapter (802.11)

### IV-3-3 Task 01: **Creating a web application**

In this activity we will create a web app composed of two steps: first, creating a python program after designing its organigram and secondly, designing an interface for this program.

#### IV-3-3-1Designing the organigram of python program:

We will use the instructions mentioned in Chapter 03 to create the python program that will control Foks-Bot (Appendix) after designing its organigram. The following figure summarizes the steps:

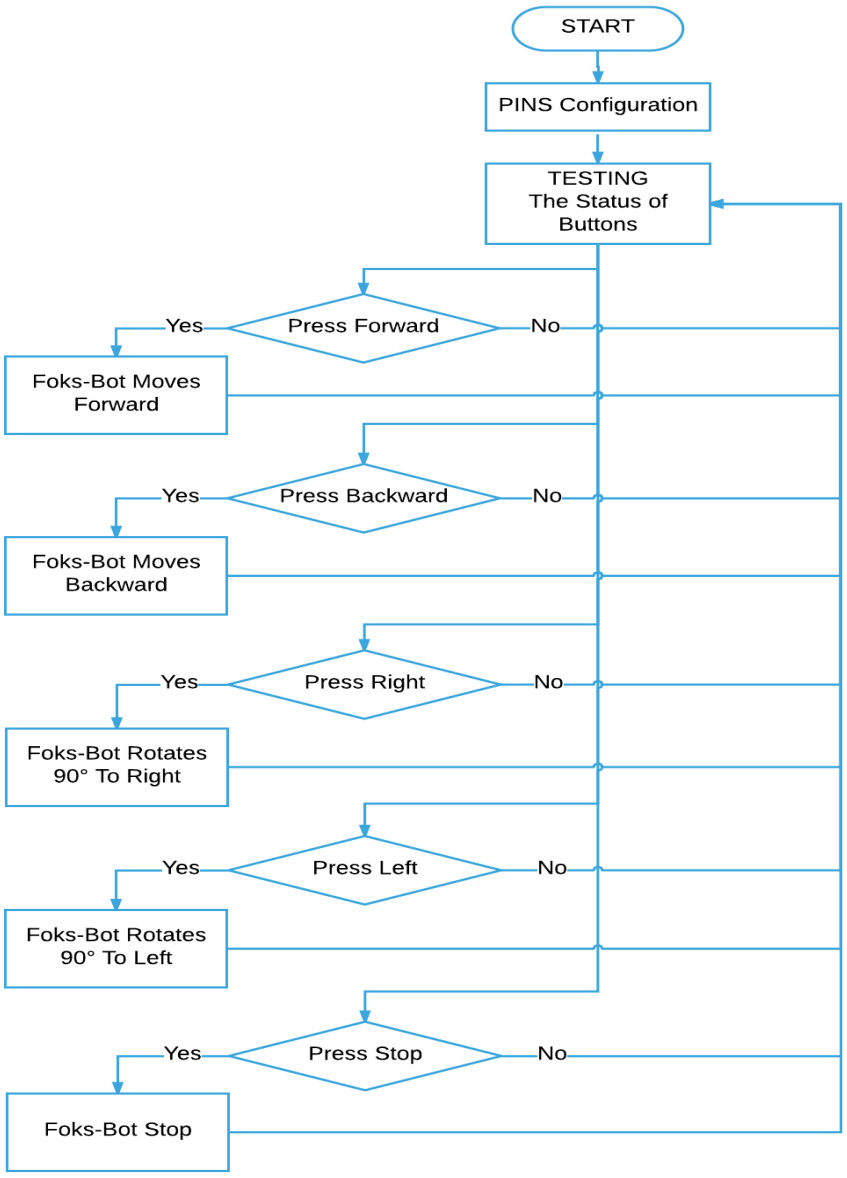



Figure IV-26: Organigram of python program

### IV-3-3-2Designing an interface with HTML:

We will use the instructions mentioned in Chapter 03 to design a python program for the interface of the web app that will be displayed on the browser (Appendix).

### IV-3-4 Task 02: **Testing and discussing the web app and Foks-Bot interactions**

### IV-1-5-A Testing the Web App control:

Will run the web app by creating the right IP address on browser.The latter is in a device that must be connected to the same router connectedto Foks-Bot. This give as the ability to take control on Foks-Bot, it controls direction by choosing the path that robot takes, we will apply this by the buttons which create in the interface like moving forward/ backward, turning right/ left and stop, as we can see in the following figures:


Figure IV-27: The interface created with HTML

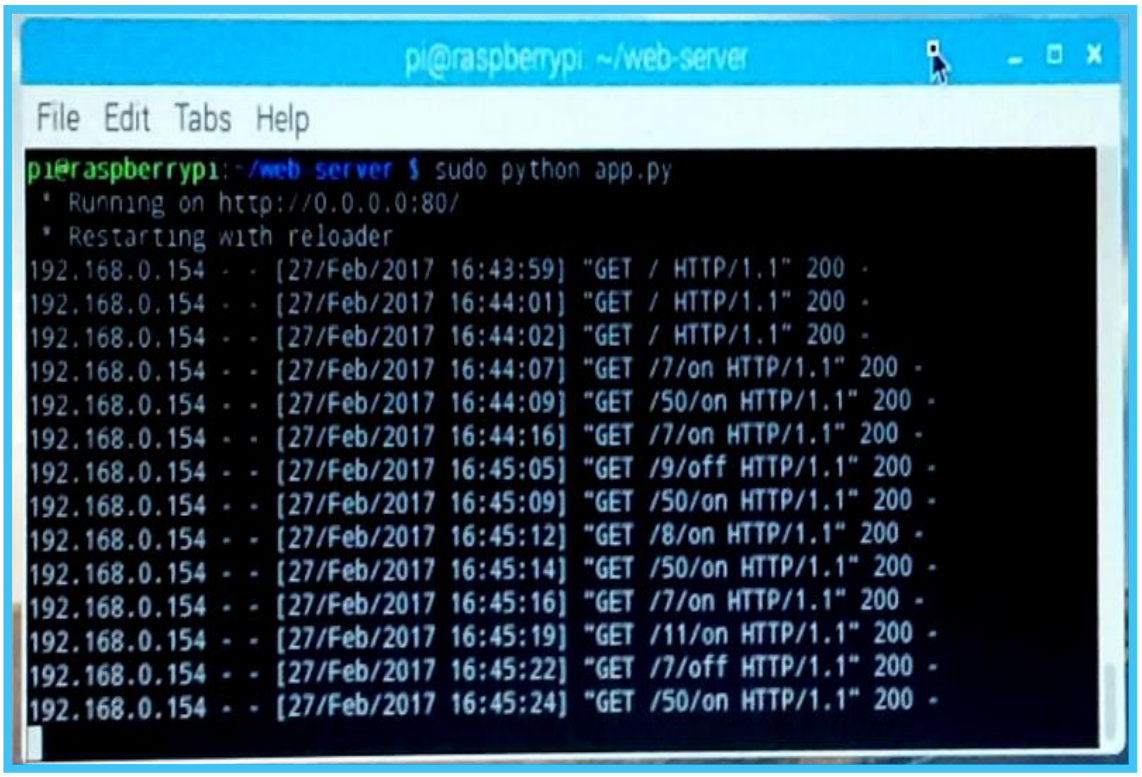



Figure IV-28: Data of the web app after making some actions

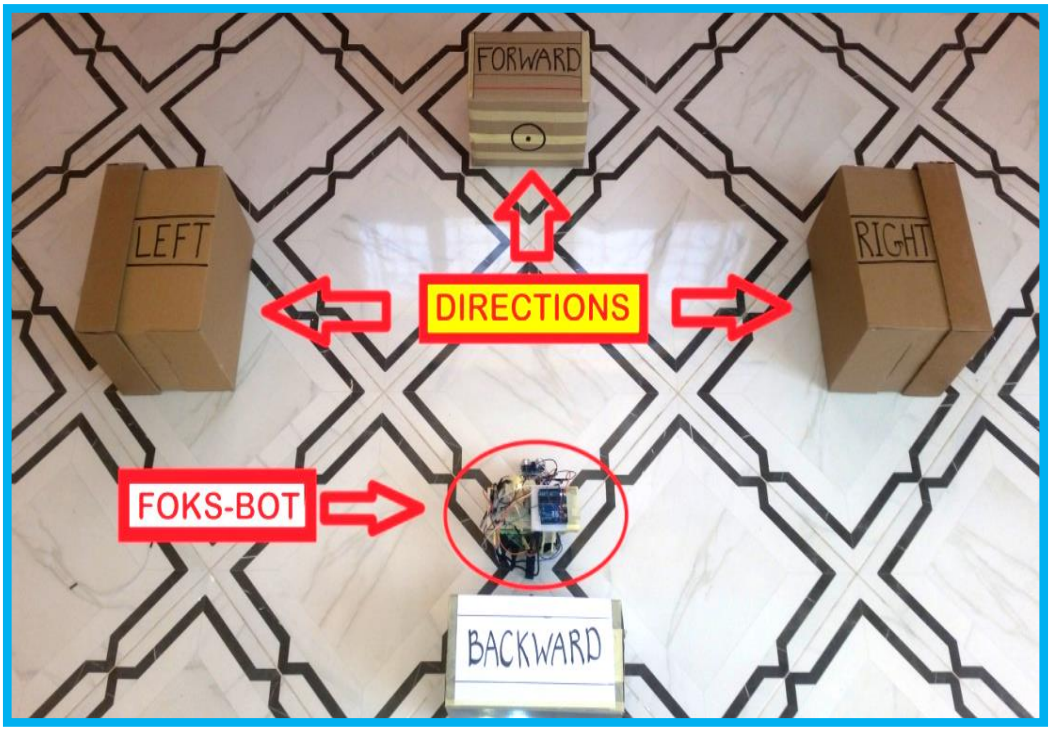



Figure IV-29: Area of experiments


Figure IV-30: Foks-Bot Moves forward/ backward

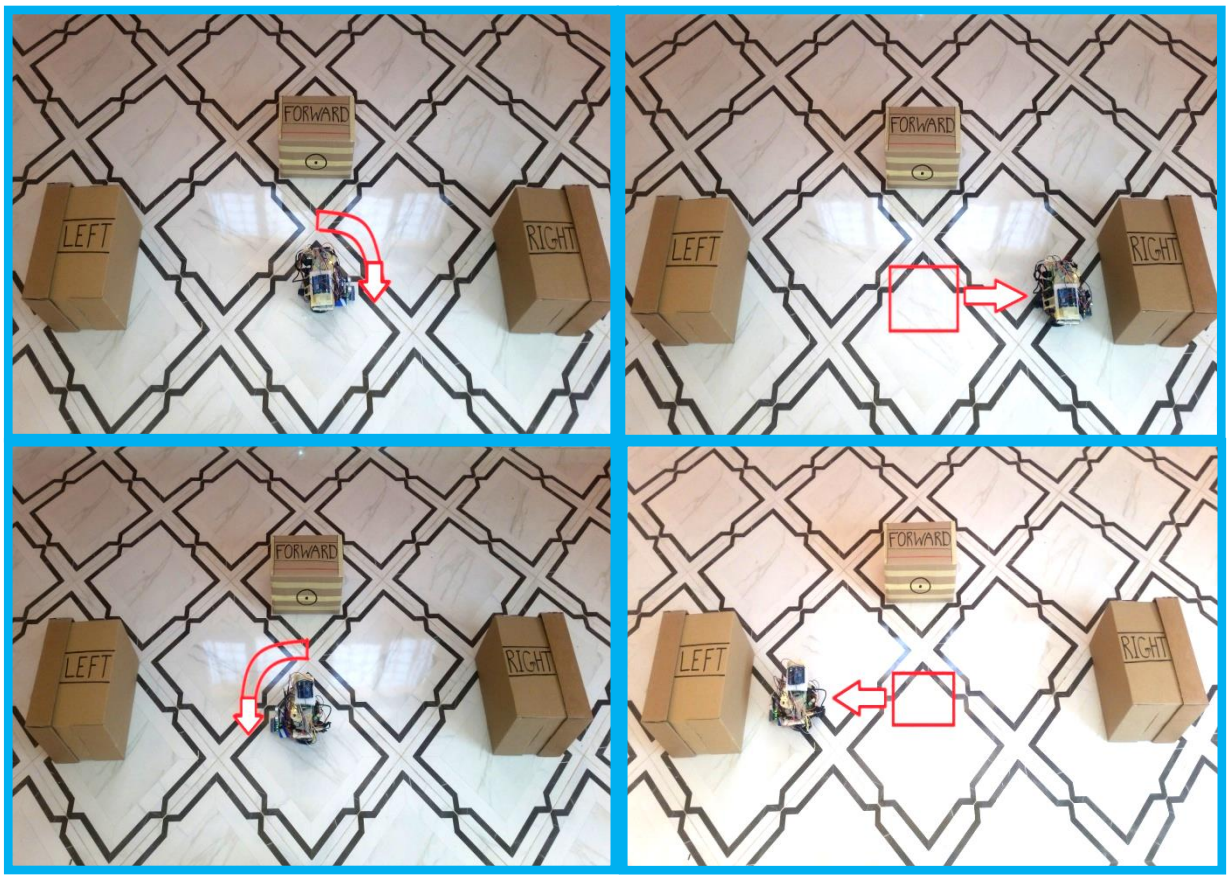



Figure IV-31: Foks-Bot Rotates then moves to right/ left

### IV-1-3-A Discussing the Web App control:

#### ❖ Advantages:

- Web App control is a wireless control with Wi-Fi.
- Using the phone camera to achieve the live-viewing on the Web App interface makes the control very easy with fewer concerns. Moreover, it is inexpensive.
- The existence of obstacles between the Wi-Fi emitter and receiver causes fewer problems compared to the IR control.
- The control range is very large and expandable.
- The ability to develop the control from being local to larger ones such as global using the internet.
- Using a frequency level of 5GHz means very high speed of transmission compared to IR control.

#### ❖ Disadvantages:

- We can see the updated information only when the page is refreshed; knowing that the solution exists but it is a matter of time restrictions
- We have circumvented the Python program to achieve my goal, but it cost me many unused pins, definitely there is a logical solution for this, but it is always a matter of time restrictions.

# Fourth Lab Session

## Autonomous control by Line follower

In the second lab session, we used Ultrasonic sensor to make Foks-Bot move autonomously. In this chapter, we will use The TCRT5000Line follower sensor based on infrared emitter and phototransistor. its working method has been explained in the second chapter, we can summarized it by saying that this sensor will make Foks-Bot follow the black line that will be created on a white floor, which means making it able to sense and differentiate between the black and white colors [39].

This Lab session is different from the others because we will merge Arduino Uno with Raspberry Pi by USB cable type B as hardware connectivity and the software connectivity will be Nanpy-v0.8 tool with some important configurations as we have seen in Chapter 03.To achieve the Autonomous control by Line follower sensors, we can easily do the configuration process of merge, where Raspberry Pi will be considered as a master and Arduino as a slave, this merging will be a big step in this project. We will also achieve this autonomous control just with Raspberry pi and compare our results [17].

### IV-4-1 Objectives:

- Wiring up the line follower sensor with Arduino Uno and then again with Raspberry Pi.
- Programming the line follower sensor on Raspberry pi.
- Testing and discussing the autonomous control.

### IV-4-2 Parts list:

- Raspberry pi.
- Arduino Uno.
- USB Cable type B.
- Robot chassis with two controlled wheels by Motors driver.
- 3 X TCRT5000 Line follower sensor.
- Connecting wires Male/Female.
- Resistors 3 X 10 kΩ.

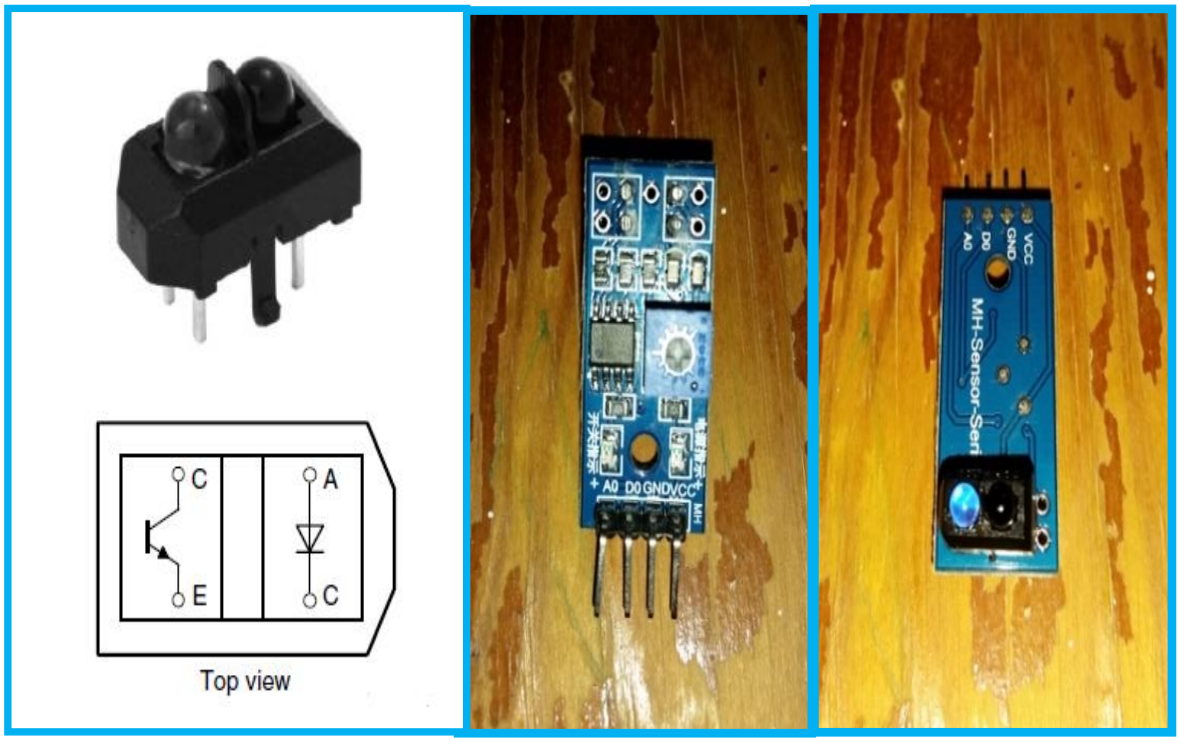



Figure IV-32: TCRT5000 the line follower sensor

### IV-4-3 Task 01: Wiring up the line follower sensor with Arduino Uno then again with Raspberry Pi

The sensor has 4 pins connected to Arduino Uno or Raspberry Pi. VCC (5V) pin and GND pin will be connected in terms of two outputs: A0 to read the information analogically and D0 to read the information digitally. Furthermore, in this activity we will use D0 digital output because we deal with situation of all or nothing, the outputs will be connected with 10kΩ resistor before connecting it with Arduino Uno or Raspberry Pi pins to reduce the incoming current. The wiring will be via female to female and male to female jumper wires, as we can see in the following figure:

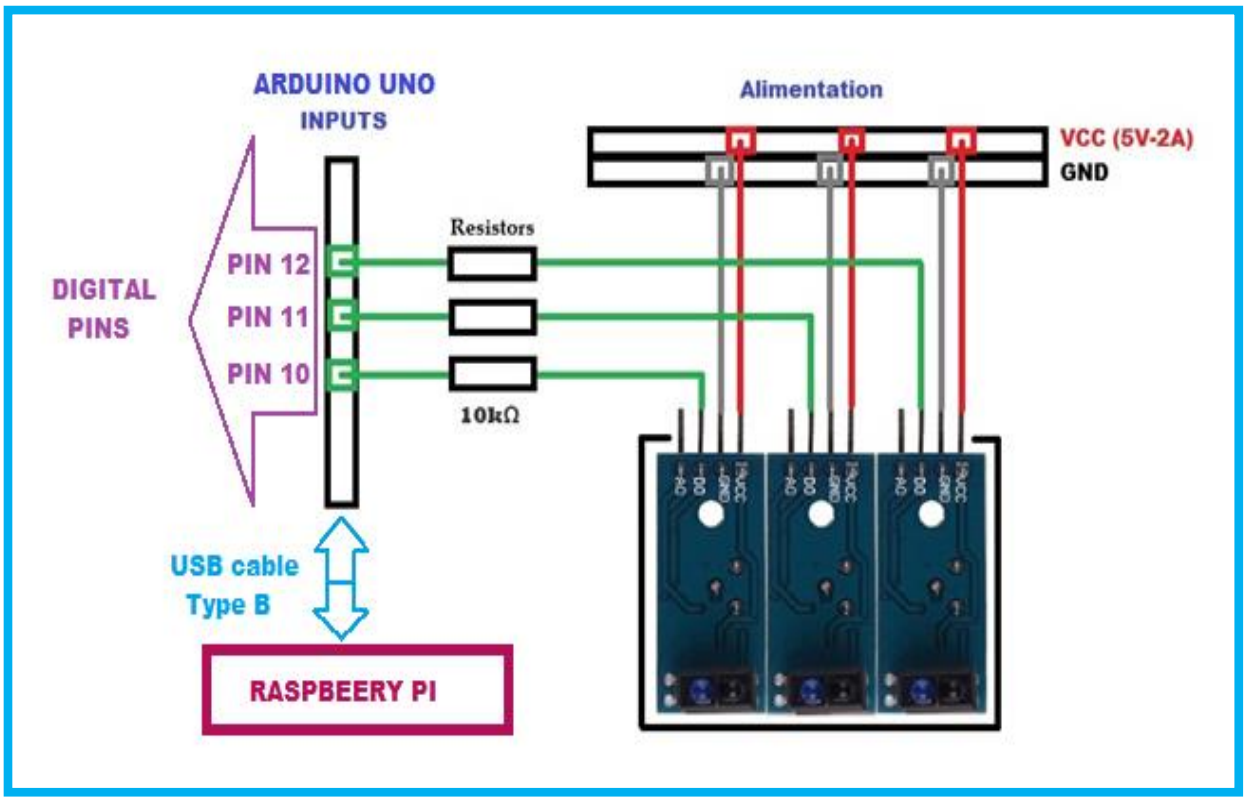



Figure IV-33: Wiring up the line follower sensorin case of merging Arduino Uno

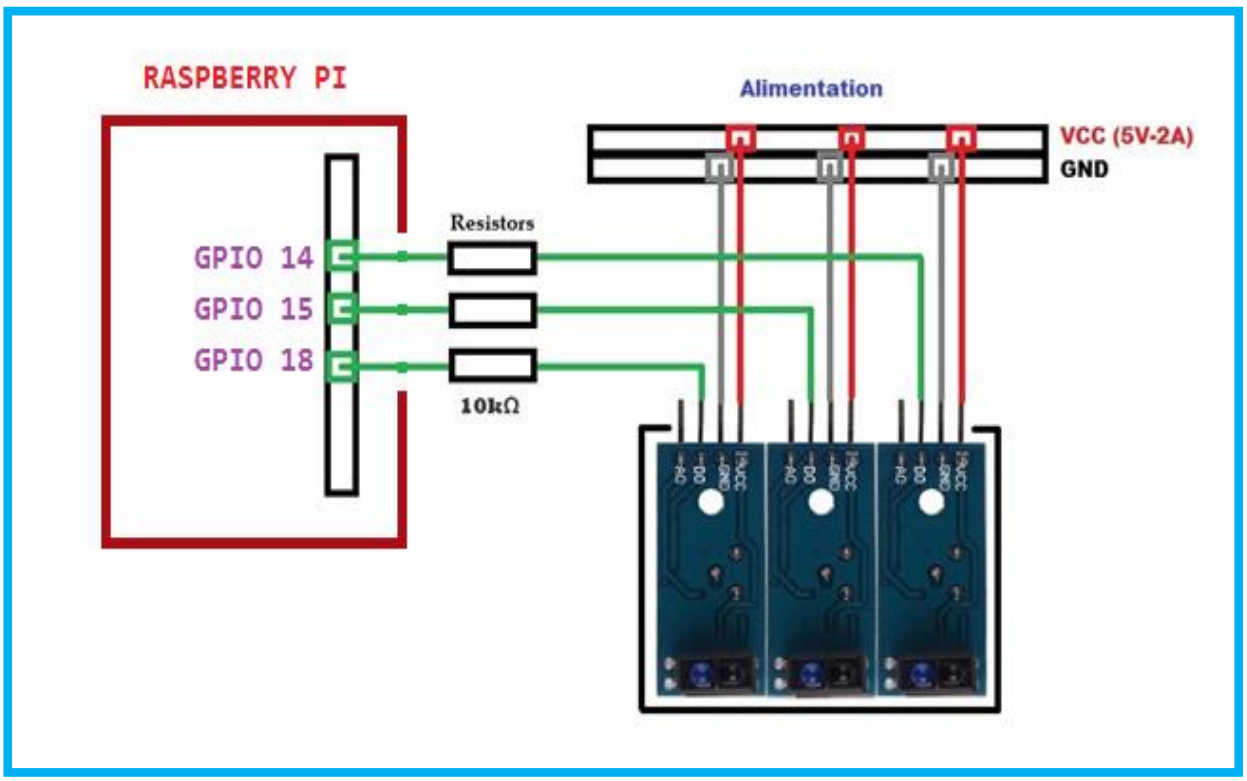



Figure IV-34: Wiring up the line follower sensor just with Raspberry Pi

### IV-3-4 Task 02: Programming the line follower sensor on Raspberry pi

#### IV-3-4-A Designing the organigram of python program:

In this activity we will create a program to realize the autonomous control with the line follower sensor as we did in the second lab session with the ultrasonic sensor, but the difference now is that the direction control with feature of black line tracking and of course making decision about the right path automatically. We will use the instructions that mentioned in the previous chapter (Chapter 03) to create the python program (Appendix), and we will start from designing an organigram for this program, as follows:

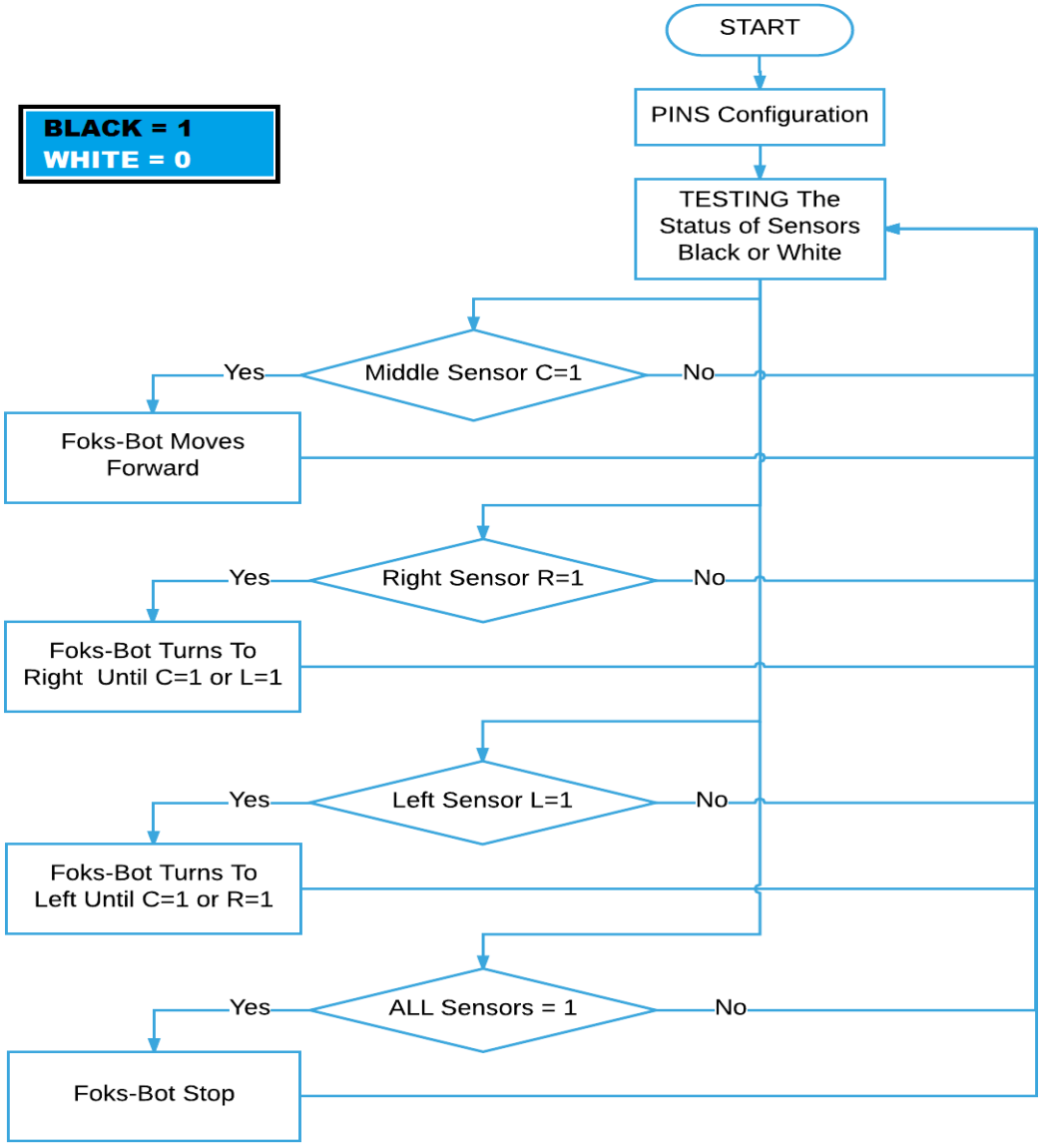



Figure IV-35: Organigram of Autonomous control by Line follower

### IV-3-4 Task 03:Testing and discussing the autonomous control

#### IV-1-5-A Testing the autonomous control:

When we activate the autonomous control mode, Foks-Bot will follow automatically the black line created on a white floor in the form of a smooth trajectory, the purpose has achieved successfully in two processes: with Arduino Uno, and then again with Raspberry Pi, as we can see in the following figures:

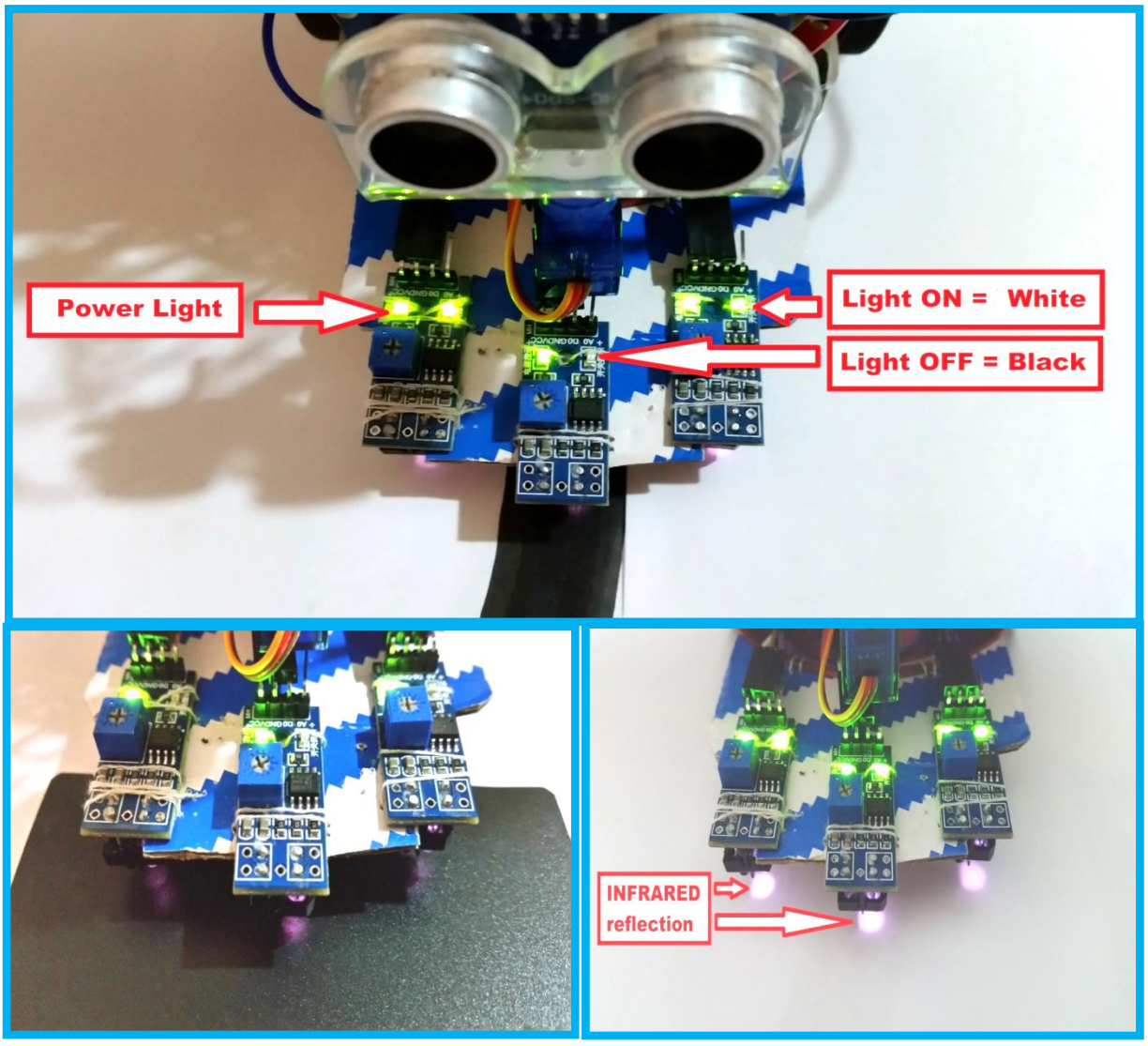



Figure IV-36:Sensors differentiate between black and white

- As we can see that the IR reflection is visible for camera but invisible for human eye, also we can note that IR reflection is brighter on white than black.

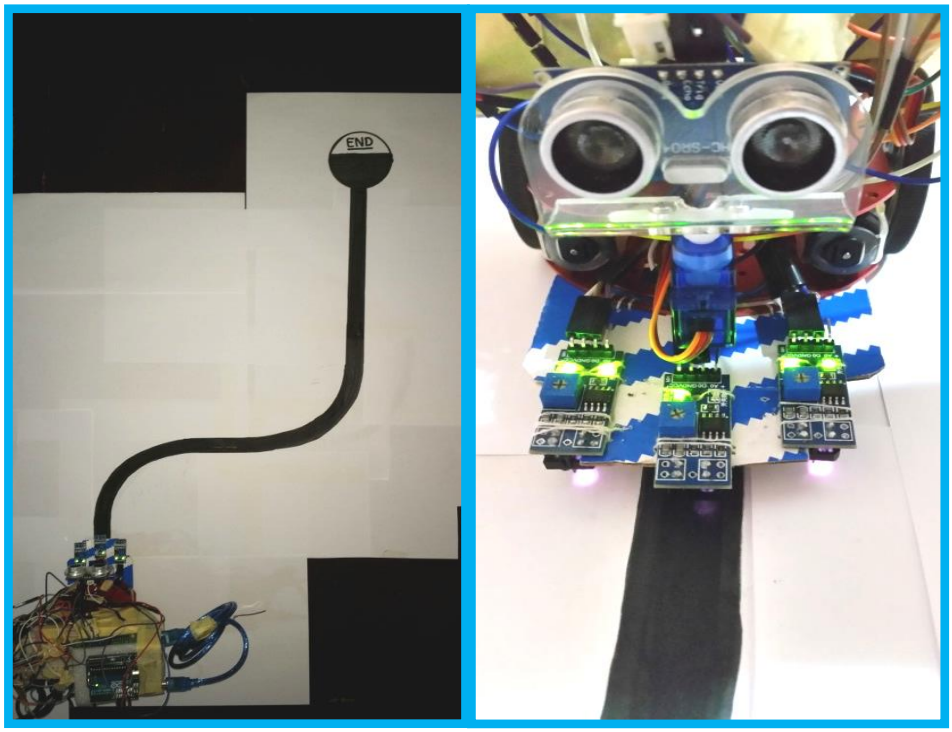



Figure IV-37: Middle sensordetects the black line and Foks-Bot moves forward

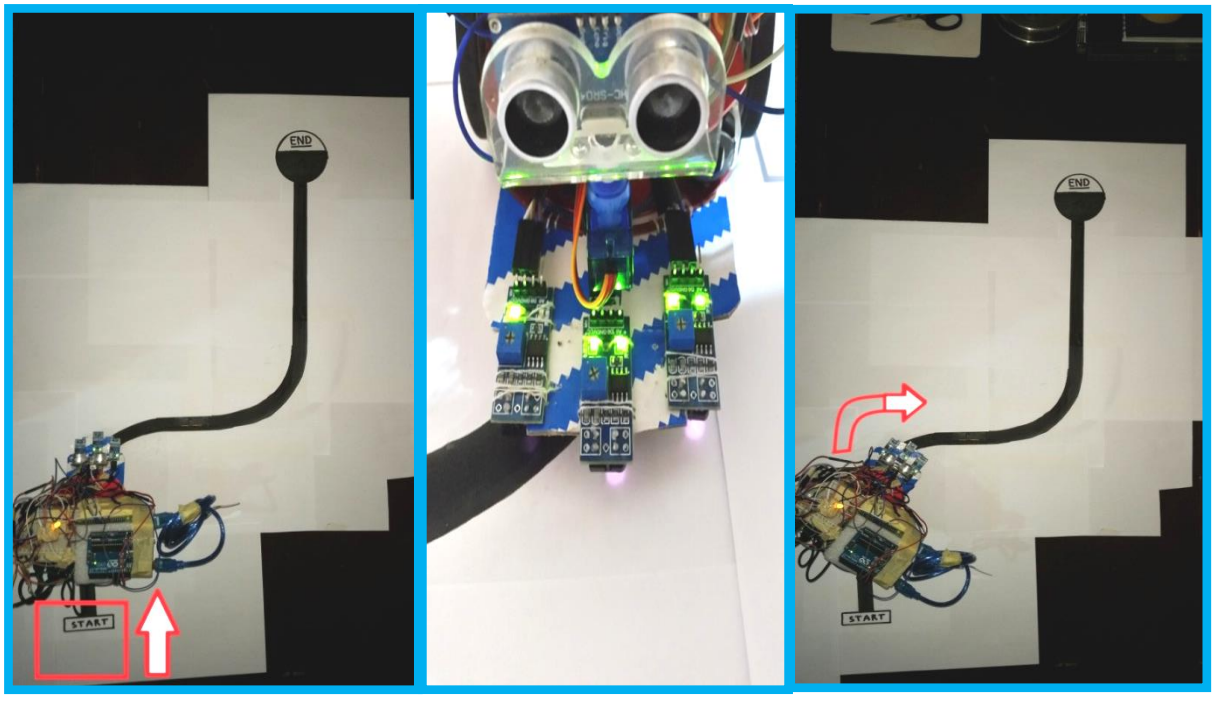



Figure IV-38: Right sensor detects the black line and Foks-Bot turns to right

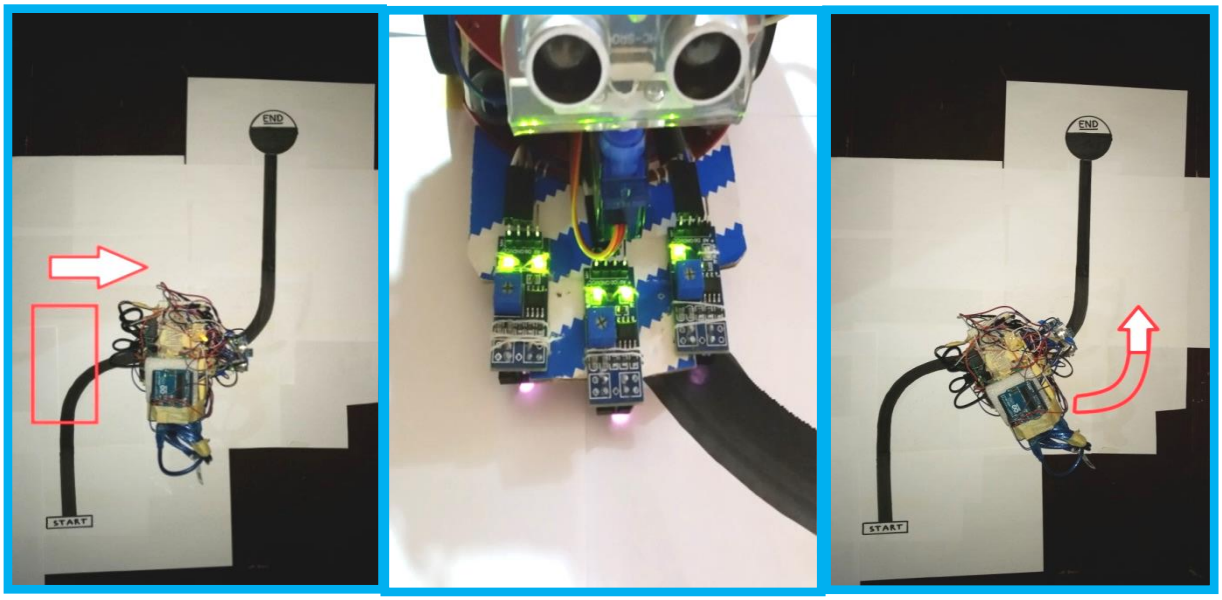



Figure IV-39: Left sensor detects the black line and Foks-Bot turns to Left

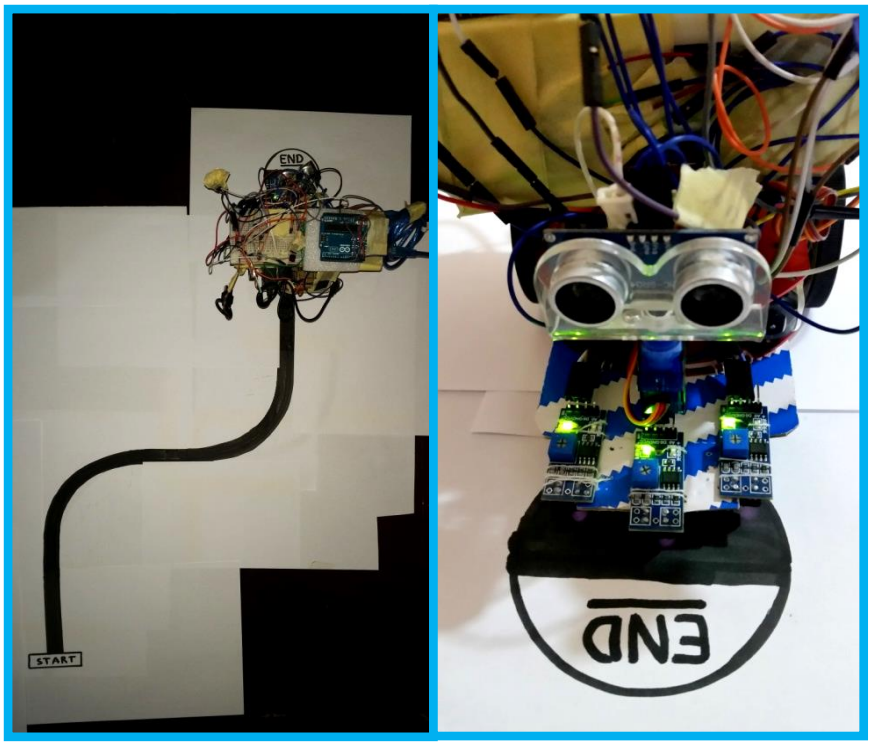



Figure IV-40: All sensor detects black and Foks-Bot Stop

- Obviously we can't see the difference between results of these processes (With or without merging Arduino Uno) in these following figures but they will be mentioned in the discussion part.

### IV-1-4-A Discussing the autonomous control:

#### ❖ Advantages:

- It works by signals, which means no physical contact takes place (No wire connection), so this will be a good option in many life aspects.
- Recommended operating conditions is 5V and 3x60mA, which is an acceptable consumption.
- Foks-Bot reacts better with less delay without merging Arduino Uno, just Raspberry Pi.

#### ❖ Disadvantages:

- Foks-Bot does not follow the line smoothly because of the simplicity of the program using the digital reading (All or nothing) instead of analog reading.
- Foks-Bot can detect specific kinds of black color such as the black sticky tape and the black marker pen which we have used both where as the printed black color will not be detected.
- Foks-Bot reacts with a delay in case of merging Arduino Uno despite of the fact that Arduino Uno includes timers unlike Raspberry pi.
- There is an annoying error, which we could not fix because of time constraints, that appears suddenly, in case of merging of Arduino Uno. Despite the correct working of the line follower mode; it makes the merging as a negative step regarding this case. We can see this error in the following figure:

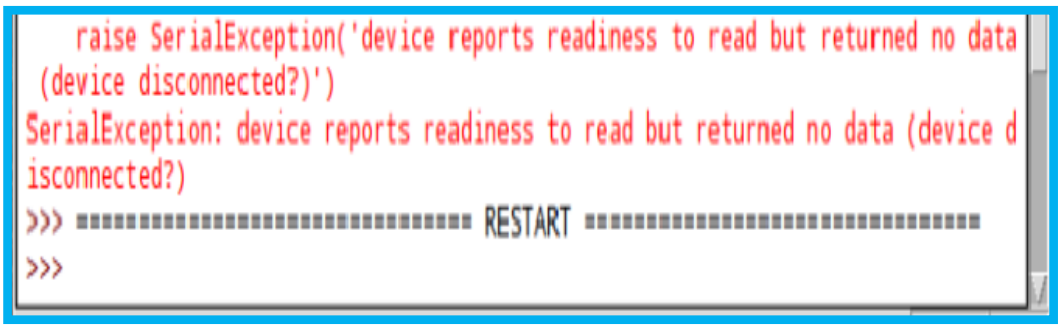



Figure IV-41: Line follower Error in case of merging Arduino Uno

# Fifth Lab Session

## Multiple Controls

After applying the four lab sessions successfully, we can consider this session as the final lab session where it will be as an assembling process for all the previous lab sessions. This means that Foks-Bot will be controlled with the four control ways at the same time: manually by IR and Wi-Fi, automatically by Ultrasonic and Line follower mode. All these methods represent a multiple control method that will be the final step of this project.

### IV-4-1 Objectives:

- Wiring up all the sensors correctly.
- Programming all modes to work together.
- Testing and discussing the multiple control.

### IV-4-2 Parts List:

- Raspberry pi.
- Robot chassis with two wheels controlled by Motors driver.
- Large Breadboard.
- Connecting wires Female/Female and Male/Female.
- Resistors.
- Sensors:
  - TSOP38238
  - HC-SR04
  - Wi-Fi Adapter 802.11 (For Raspberry Pi 1 and 2, but 3 contains Wi-Fi by default)
  - TCRT5000

### IV-5-3 Task 01: **Wiring up all the sensors correctly**

In this activity we will connect all the sensors to Raspberry Pi as we did in the previous lab sessions and we will notice that the multiplicity of sensors means more power consumption, so we need a power source with good capacity, as we can see in the second chapter. We will use a Samsung 20000mAh power bank which is suitable for this multi-functionality. The connecting process will be as follows:

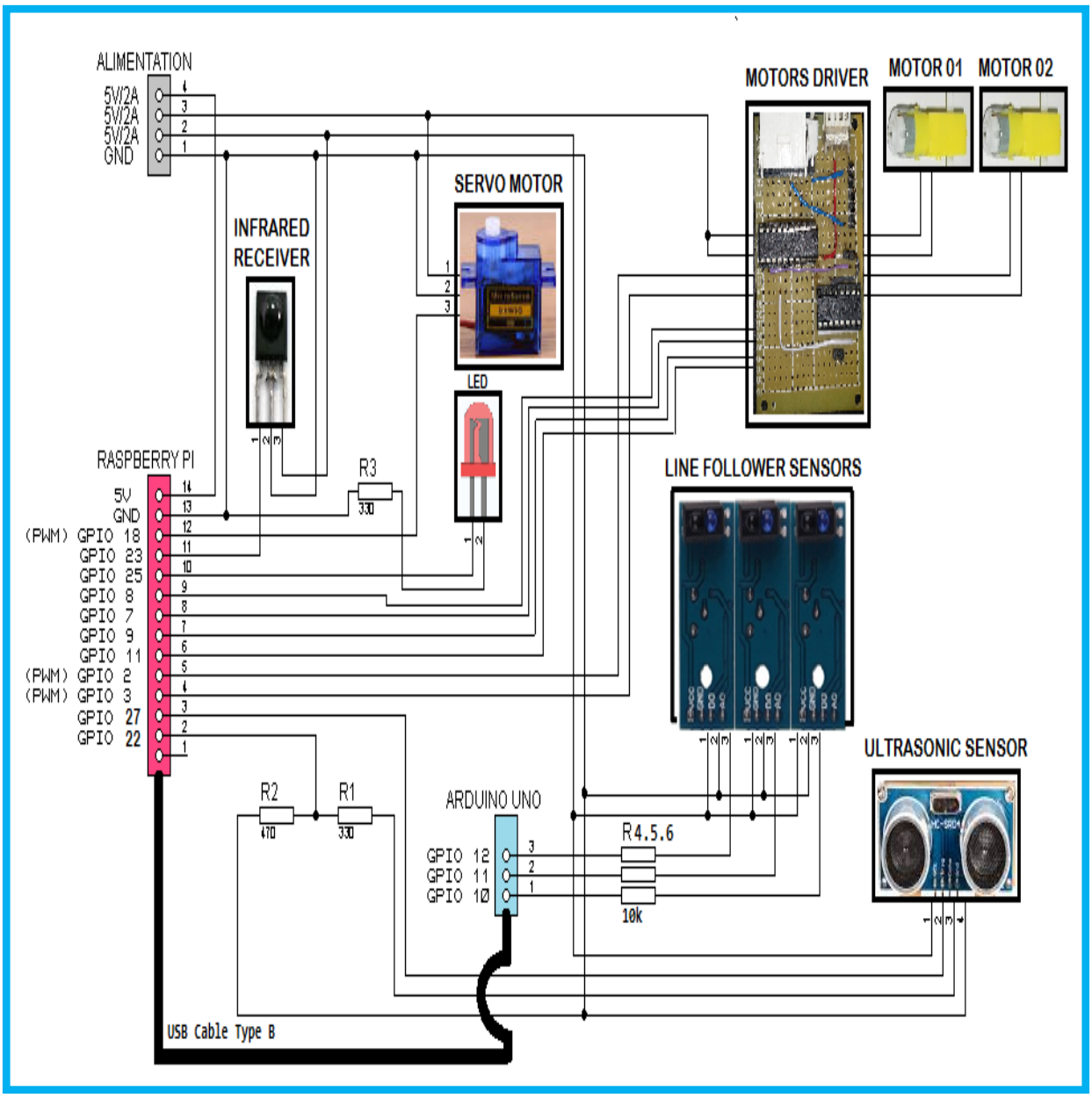



Figure IV-42: Connecting all sensors in case of merging Arduino Uno

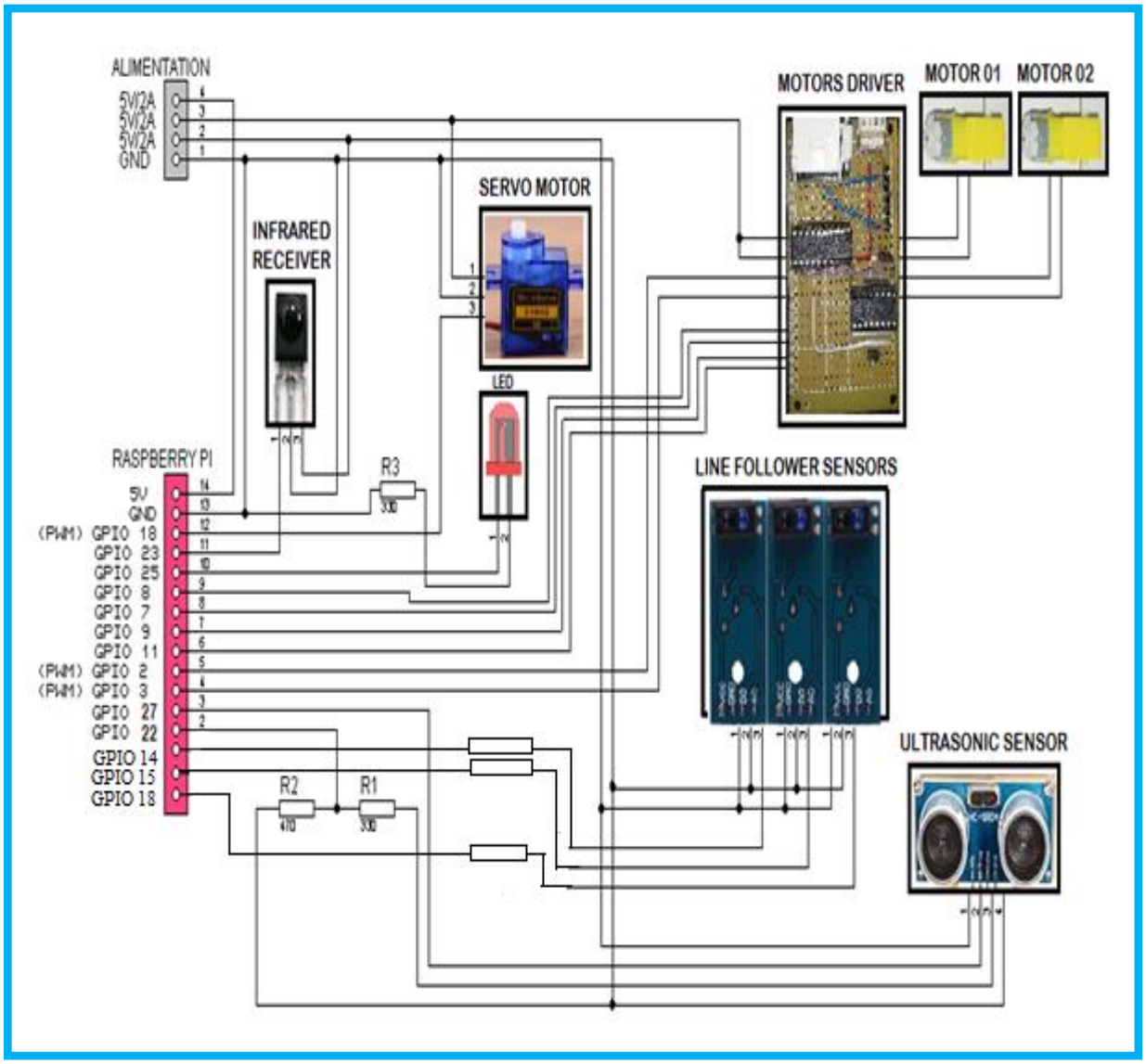



Figure IV-43: Connecting all sensors without merging Arduino Uno

### IV-5-4 Task 02: **Programming all modes to work together**

### IV-3-4-A Designing the organigram of python program:

In this activity we will create a python program to realize the multiplicity of control ways on Foks-Bot. This will be done by the integration of all python programs of the previous sessions to work in a nested way. We will start from designing an organigram for this program, as follows:

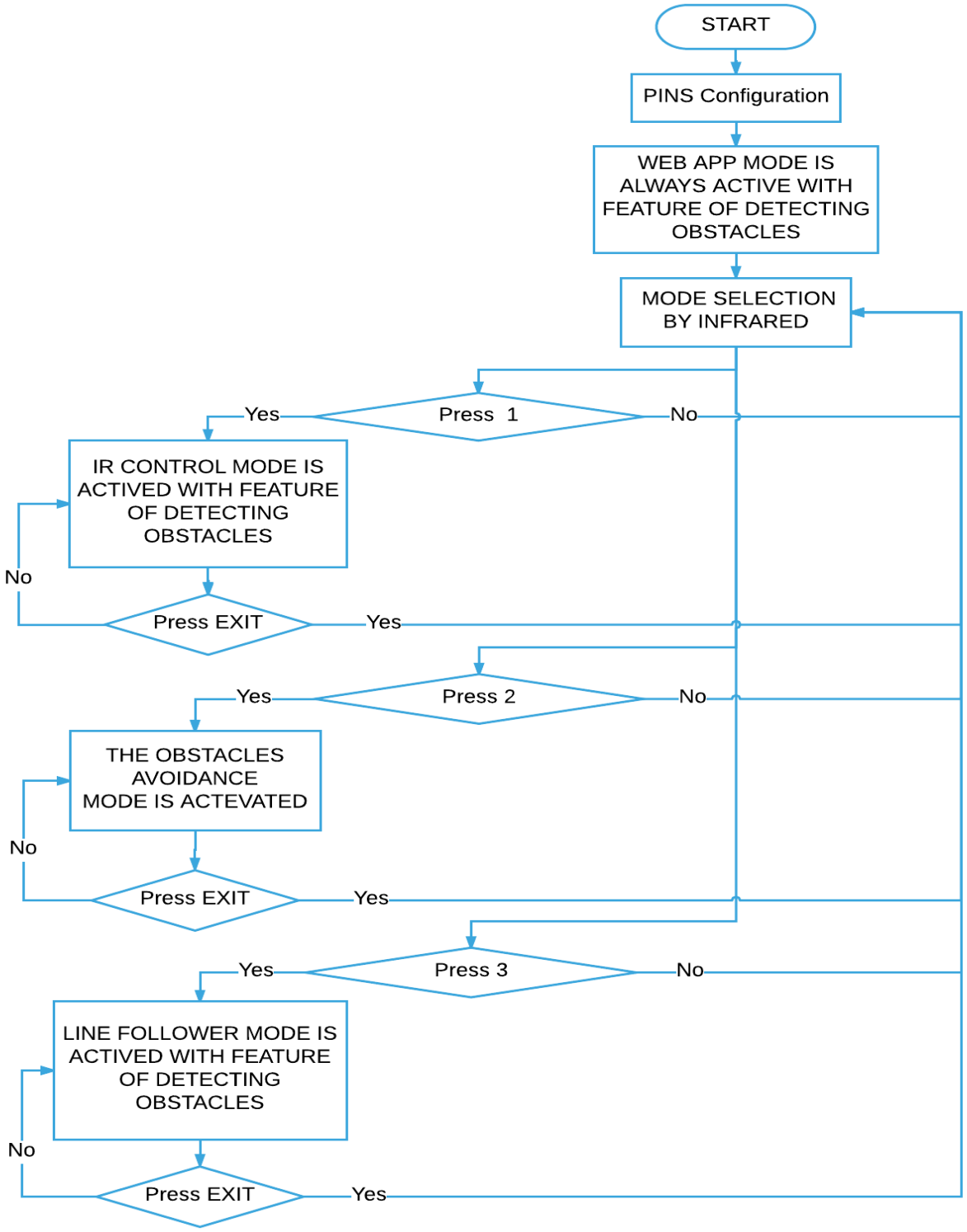



Figure IV-44: Organigram of the multiple controls

After this integration processes, there remains the last step, which is one of the best features of Raspberry Pi considering that it is based on a processor not a microcontroller, which is the activation of multiple programmes in the same time. We will activat the two integrated programmes (the Web App with displaying the Ultrasonic status and the multiple control program). We will do this by activating the web app in the terminal cammand line and the multiple control program in python.

### IV-5-5 Task 03: **Testing and discussing the multiple control**

#### IV-1-5-A Testing the multiple control:

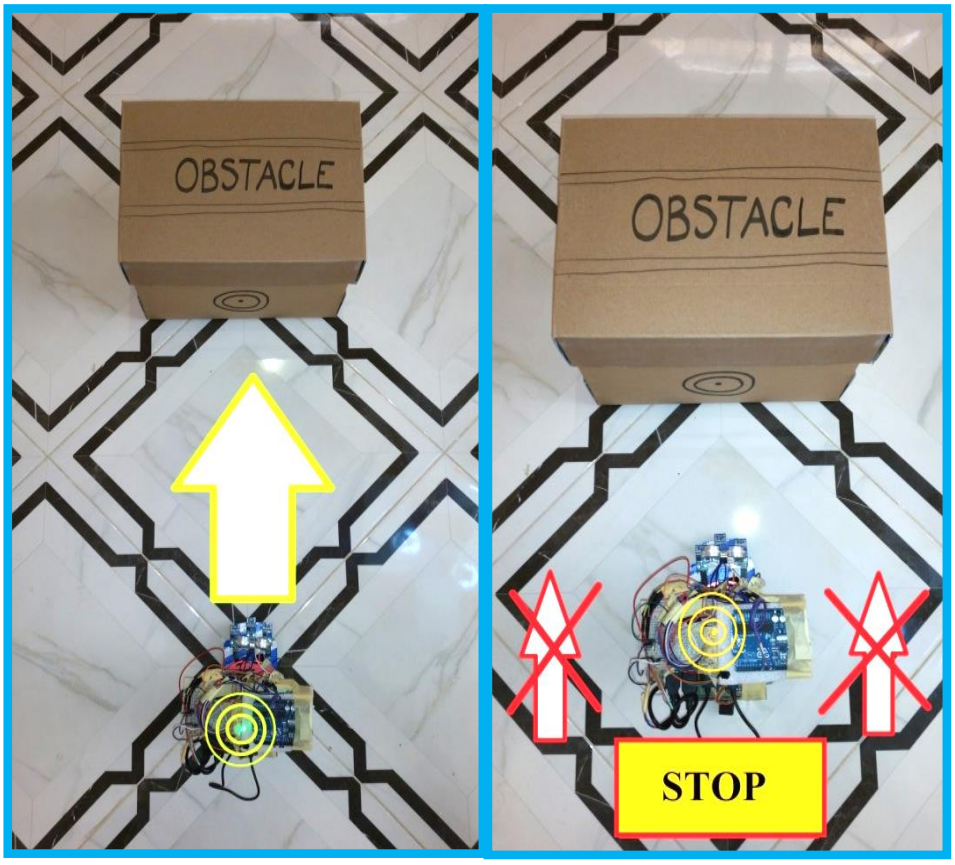



Figure IV-45: IR Mode with feature of detecting obstacles

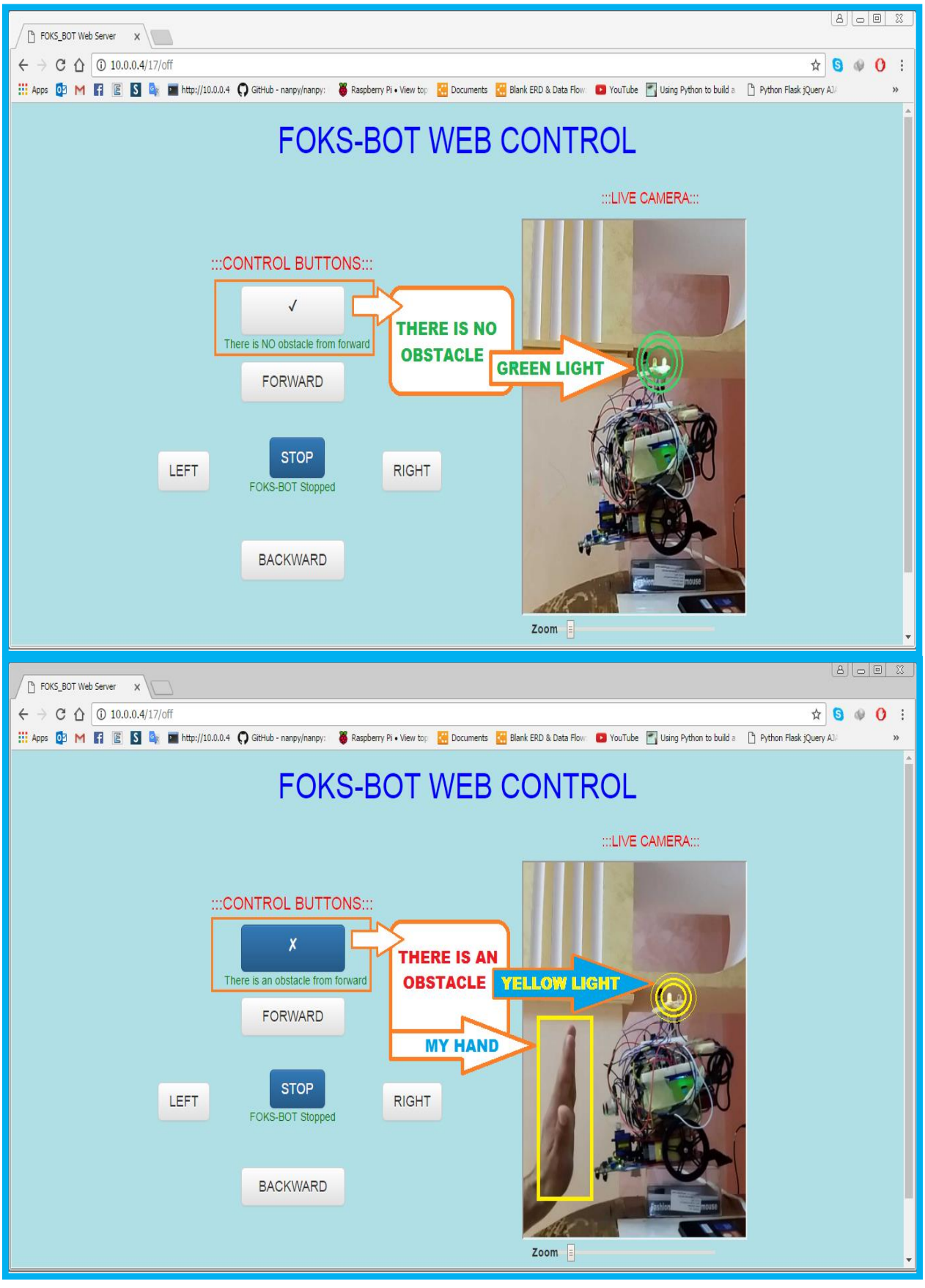



Figure IV-46: Integration of Ultrasonic state in the Web App

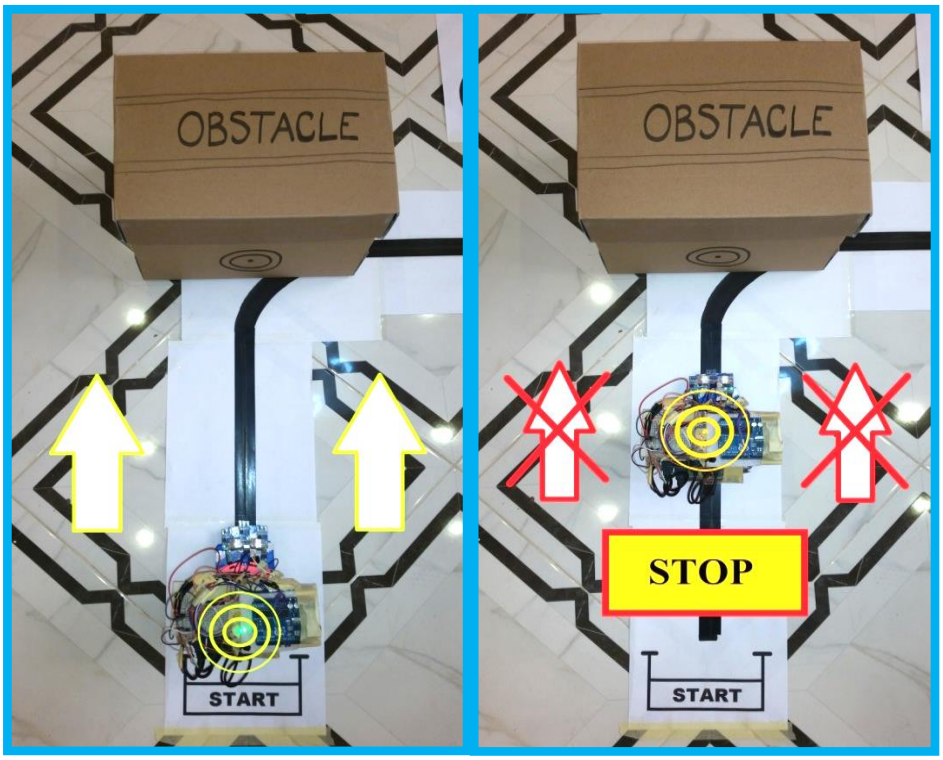



Figure IV-47: Line Follower Mode with feature of detecting obstacles

### IV-1-5-A Discussing the multiple control:

#### ❖ Advantages:

- Using ultrasonic sensor to detect obstacles with the IR control mode and the line Follower mode is a good improvement.
- IR has used as a controller to change the modes as we desire. It would be better if I Used the Web App for this work, but it is a matter of time constraints.
- Using Raspberry Pi without merging Arduino for the multiple controls is better because it helps avoid some annoying errors that we could not fix because of time constraints.
- Merging all these modes and making all these work together, give to Foks-Bot the feature of tasks multiplicity.

### ❖ Disadvantages:

- During using multiple controls the processor temperature becomes high, which makes Raspberry Pi reboot randomly.
- We can see in this mode, the defect of Raspberry Pi and Python which is there is no reacting in the real time and this causes Foks-Bot to react with a remarkable delay.
- There are two annoying errors, which we could not fix because of time constraints, appearing suddenly in case of merging of Arduino Uno and despite the correct working of the multiple control mode, it makes the merging as a disadvantageous step regarding this case. we can see these errors in the following figure:

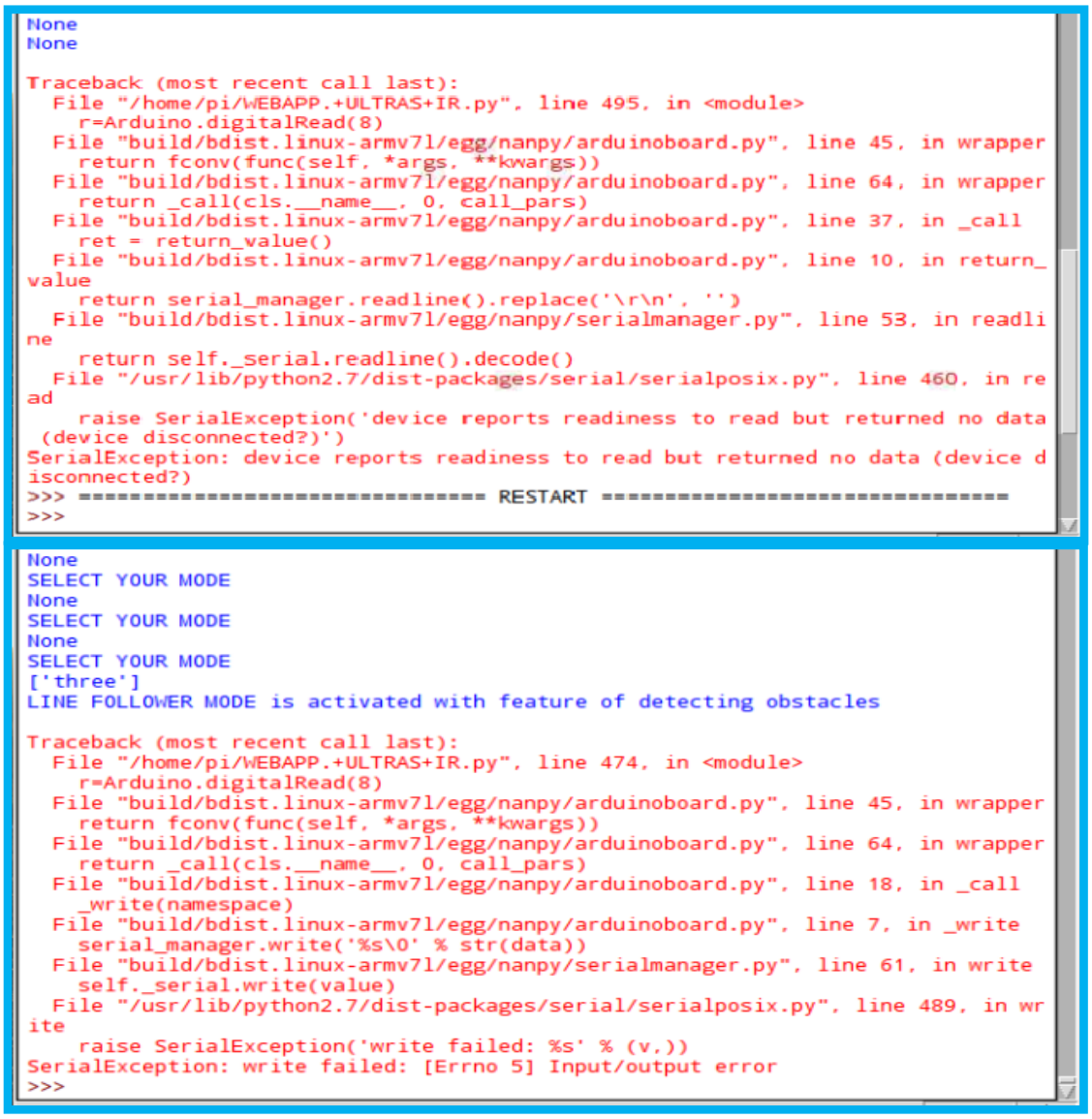



Figure IV-48: Multipl control Errors in case of merging Arduino Uno

## ❖ Conclusion:

This Chapter presents a series of lab sessions which contain hands-on activities and tasks where we have built Foks-Bot.

These lab sessions are considered as a good way to help students link between theoretical and practical work and learn about how to build this pedagogic mobile robot “Foks-Bot”. These lab sessions contain the objectives that must to be reached, parts, sensors and actuators needed. All building steps have been explained in many tasks according to the laboratory session.

Each lab session contains a part of testing and discussing how Foks-Bot reactions are, how it performs the specific tasks and what the advantages and disadvantages are. Doing these tests, discussions and notes is what gives us the opportunity to be able to develop, improve and succeed.

# General Conclusion

The objective of this project has been successfully achieved, which is designing an experimental mobile robot controlled by Raspberry Pi for student laboratory works. This robot contains multiple control modes and is able to embrace other experiments.

Multiple control methods have been realized using the electronic card Raspberry Pi as a control unit, some actuators “Two DC motors and One Servo motor” and a set of sensors which allow our robot “Foks-Bot” to detect obstacles with distance calculation “Ultrasonic sensor” and detect the black line “Tracking sensor” which are called autonomous control methods. Moreover, it allows receiving infrared signals “IR receiver” and Wi-Fi signals “Wi-Fi adapter” which are the manual control methods, a breadboard has been also added to Foks-Bot for any other experiments.

Foks-Bot has become ready for use by students in order to do the designed functions. This modest project has clarified many issues regarding practical works, where we have learned a lot of electric, electronic, mechanic and robotic concepts. Foks-Bot helps us develop our own view about the theoretical work and how to link it to the practical work. All this has a good effect on us and we hope it will be more effective on students.

At the end, this robot paves the way for further research and improvements, where the proposed developments are:

- Use of Artificial intelligence in the first place
- The movement accuracy can be developed
- Image processing
- Navigation
- Maze solver

# *References*

## *Chapter 01: Generalities about robotics*

[1] Angar Rachid, Robot mobile expérimentale, Master degree, Micro-Computing Instrumentation, University Mohamed khider biskra, May 31, 2016

[2] Laêtitia Matignon, Introduction of robotic, Licence 1st year, GREYC-CNRS University of Caen, France, 2011/2012: http://liris.cnrs.fr/laetitia.matignon/index/coursL1robotique.pdf

[3] Isaac Asimov, Menteur! Book, Analog Science Fiction and Fact, May 1941

[4] FANUC ROBOTICS, Introduction about robotics
http://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/4836/4836-fanuc-introduction-la-robotique-juin-2014.pdf

[5] AIBO the robot dog, Fukuok Japan: http://www.sony-aibo.com/

[6] Introduction about Mobile Robots: http://www.sitedunxt.fr/articles/print.php?id=24

[7] Dr Bensaid Samir, Course Sensors and actuators, Electrical engineering, University of bouira, 2014

[8] Introduction about actuator: http://dondon.vvv.enseirb-matmeca.fr/RSIcapteur/a.pdf

[9] O.VITRY, P ROUSSET, PRINCIPE, CARACTERISTIQUES ET MAINTENANCE, LES ACTIONNEURS, High school « Léon de Lepervanche »
http://lpmei.com/cd_bac_mei/eleve/cours/Automatisme/223%20Les%20actionneurs.pdf

[10] Hydraulic cylinder: https://en.wikipedia.org/wiki/Hydraulic_cylinder

[11] David Filliat, Mobile robotic, National School of Advanced Technologies, ParisTech, April 2012:

[12] Bruno Siciliano Prof & Oussama Khatib Prof, spring handbook of robotics, 2008

[13] Wiki books, Robotics Wheeled Robots, 2016
https://en.wikibooks.org/wiki/Robotics/Types_of_Robots/Wheeled

[14] Dudek and Jenkin, NOTES about Differential Drive Kinematics, Computational Principles of Mobile Robotics

[15] Fred, Mobile robots, March 15, 2012
http://www.sitedunxt.fr/articles/articles-7-24-5+les-robots-mobiles.php

## *Chapter 02: Mobile robot hardware*

[16] Pyo, Introduction about Raspberry Pi, April 21, 2016
http://www.pexiweb.be/introduction-au-raspberry-pi/
[17] Abd Allah Ali Abd Allah, Simply Raspberry pi, March 2014
[18] Ultra VNC remote access tools http://www.uvnc.com/
[19] Technical Document of Arduino Uno
[20] Jayant, Arduino vs Raspberry Pi, Differences between the two
https://circuitdigest.com/article/arduino-vs-raspberryp-pi-difference-between-the-two
[21] Pauline Gill, Gear DC Motor http://ourpastimes.com/gear-motor-5313147.html
[22] Arpit Jain, Working way of geared DC motor
http://www.engineersgarage.com/insight/how-geared-dc-motor-works
[23] Dilip Raja, Servo Motor Control with Raspberry Pi
https://circuitdigest.com/microcontroller-projects/raspberry-pi-servo-motor-control
[24] RakeshRon, L293D Motor Driver, AUG 16, 2013 http://www.rakeshmondal.info/L293D-Motor-Driver
[25] Kushagra, L293D http://www.engineersgarage.com/electronic-components/l293d-motor-driver-ic
[26] Technical Document of MC74HC244A « buffer »
[27] Doug Lowe, Electronics All-in-One for Dummies, Feb 2012
http://www.dummies.com/programming/electronics/electronics-introduction-to-infrared-light/
[28] Infrared Receiver TSOP38238
https://www.modmypi.com/electronics/sensors/ir-infrared-receiver-tsop38238
[29] Infrared Receivers
http://www.futureelectronics.com/en/optoelectronics/infrared-receivers.aspx
[30] Technical Document of TSOP38
[31] Practical Workbook Artificial Intelligence & Robotics, Department of Computer & Information Systems Engineering NED University of Engineering & Technology, Karachi, Pakistan
[32] Ultrasonic Sensor
http://www.education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html

[33] Vivek Kartha, Interfacing HC-SR04 Ultrasonic Sensor with Raspberry PI
https://electrosome.com/hc-sr04-ultrasonic-sensor-raspberry-pi/
[34] MATT, Ultrasonic Distance Measurement Using Python, DEC 30, 2012
http://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/
[35] Ross Shannon, HTML: http://www.yourhtmlsource.com/starthere/whatishtml.html
[36] WI-FI, Feb 2016: http://ccm.net/faq/298-what-is-wifi-and-how-does-it-work
[37] Bradley Mitchell, Range of WI-FI
https://www.lifewire.com/range-of-typical-wifi-network-816564
[38] Tarun Agarwal, Line Follower Robots, Controlling, Working Principle and Applications
https://www.elprocus.com/line-follower-robot-basics-controlling/
[39] Technical Document of TCRT5000
[40] Line tracking sensors and algorithms: https://www.ikalogic.com/line-tracking-sensors-and-algorithms/

## *Chapter 03: Configuring and Programming*

[41] Learning Python: http://www.bestprogramminglanguagefor.me/why-learn-python
[42] Raspberry GPIO: https://learn.sparkfun.com/tutorials/raspberry-gpio/python-rpigpio-api
[43] Alex Bain, Setting Up LIRC on the Raspberry Pi, JAN 6, 2013
http://alexba.in/blog/2013/01/06/setting-up-lirc-on-the-raspberrypi/
[44] Mark Williams, Control the GPIO on a Raspberry Pi with an IR Remote
http://ozzmaker.com/how-to-control-the-gpio-on-a-raspberry-pi-with-an-ir-remote/
[45] Interfacing Lirc with Python, MAR 14, 2014
http://notepad.xavierdetourbet.com/raspberry-interfacer-lirc-avec-python/
[46] Read a HC-SR04 ultrasonic sensor with a Raspberry Pi, MAI 17, 2013
http://www.manuel-esteban.com/lire-un-capteur-ultrason-hc-sr04-avec-un-raspberry-pi/
[47] Dilip Raja, Servo Motor Control with Raspberry Pi
https://circuitdigest.com/microcontroller-projects/raspberry-pi-servo-motor-control
[48] Matt Richardson, Creative Technologist, Serving Raspberry Pi with Flask
http://mattrichardson.com/Raspberry-Pi-Flask/
[49] Web developers, site of a Norwegian software development and consulting company
https://www.w3schools.com/

## References

[50] Using Arduino board with Raspberry Pi: https://github.com/nanpy/nanpy

[51] Using an Arduino as a slave I/O board, Jun 12, 2013
https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=46881&p=368522&hilit=Arduino#p368522

# Appendix A

❖ **Python Program of Infrared mode:**

```
import RPi.GPIO as GPIO
import pylirc, time


GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)

GPIO.setup(2,GPIO.OUT)
GPIO.setup(3,GPIO.OUT)
GPIO.setup(4,GPIO.OUT)

P = GPIO.PWM(2,50)
P.start(91)

p = GPIO.PWM(3,50)
p.start(100)


GPIO.output(18, GPIO.LOW)
GPIO.output(9, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(8, GPIO.LOW)
GPIO.output(7, GPIO.LOW)
GPIO.output(25, GPIO.LOW)

s = pylirc.init("myprogram")

while True :

   m = pylirc.nextcode()
   print m

   if (m == ['record']) :
      GPIO.output(25, GPIO.HIGH)

   if (m == ['cut']) :
      GPIO.output(25, GPIO.LOW)

```

```
44
45      if (m == ['up']) :
46          GPIO.output(9, GPIO.LOW)
47          GPIO.output(11, GPIO.LOW)
48          GPIO.output(8, GPIO.LOW)
49          GPIO.output(7, GPIO.LOW)
50          time.sleep(0.001)
51          GPIO.output(9, GPIO.HIGH)
52          GPIO.output(11, GPIO.LOW)
53          GPIO.output(8, GPIO.HIGH)
54          GPIO.output(7, GPIO.LOW)
55
56      if (m == ['down']) :
57          GPIO.output(9, GPIO.LOW)
58          GPIO.output(11, GPIO.LOW)
59          GPIO.output(8, GPIO.LOW)
60          GPIO.output(7, GPIO.LOW)
61          time.sleep(0.001)
62          GPIO.output(9, GPIO.LOW)
63          GPIO.output(11, GPIO.HIGH)
64          GPIO.output(8, GPIO.LOW)
65          GPIO.output(7, GPIO.HIGH)
66
67      if (m == ['left']) :
68          GPIO.output(9, GPIO.LOW)
69          GPIO.output(11, GPIO.LOW)
70          GPIO.output(8, GPIO.LOW)
71          GPIO.output(7, GPIO.LOW)
72          time.sleep(0.001)
73          GPIO.output(9, GPIO.LOW)
74          GPIO.output(11, GPIO.HIGH)
75          GPIO.output(8, GPIO.HIGH)
76          GPIO.output(7, GPIO.LOW)
77          time.sleep(0.367)
78          GPIO.output(11, GPIO.LOW)
79          GPIO.output(8, GPIO.LOW)
80
81      if (m == ['right']) :
82          GPIO.output(9, GPIO.LOW)
83          GPIO.output(11, GPIO.LOW)
84          GPIO.output(8, GPIO.LOW)
85          GPIO.output(7, GPIO.LOW)
86          time.sleep(0.001)
87          GPIO.output(9, GPIO.HIGH)
88          GPIO.output(11, GPIO.LOW)
89          GPIO.output(8, GPIO.LOW)
90          GPIO.output(7, GPIO.HIGH)
91          time.sleep(0.38)
92          GPIO.output(9, GPIO.LOW)
93          GPIO.output(7, GPIO.LOW)
94
95      if (m == ['ok']) :
96          GPIO.output(9, GPIO.LOW)
97          GPIO.output(11, GPIO.LOW)
98          GPIO.output(8, GPIO.LOW)
99          GPIO.output(7, GPIO.LOW)
```

```
100
101
102     if (m == ['l']) :
103         GPIO.output(9, GPIO.LOW)
104         GPIO.output(11, GPIO.LOW)
105         GPIO.output(8, GPIO.LOW)
106         GPIO.output(7, GPIO.LOW)
107         time.sleep(0.001)
108         GPIO.output(9, GPIO.LOW)
109         GPIO.output(11, GPIO.HIGH)
110         GPIO.output(8, GPIO.HIGH)
111         GPIO.output(7, GPIO.LOW)
112
113     if (m == ['r']) :
114         GPIO.output(9, GPIO.LOW)
115         GPIO.output(11, GPIO.LOW)
116         GPIO.output(8, GPIO.LOW)
117         GPIO.output(7, GPIO.LOW)
118         time.sleep(0.001)
119         GPIO.output(9, GPIO.HIGH)
120         GPIO.output(11, GPIO.LOW)
121         GPIO.output(8, GPIO.LOW)
122         GPIO.output(7, GPIO.HIGH)
123
124     if (m == ['volumeup']) :
125         GPIO.output(9, GPIO.LOW)
126         GPIO.output(11, GPIO.LOW)
127         GPIO.output(8, GPIO.LOW)
128         GPIO.output(7, GPIO.LOW)
129         time.sleep(0.001)
130         GPIO.output(9, GPIO.HIGH)
131         GPIO.output(11, GPIO.LOW)
132         GPIO.output(8, GPIO.LOW)
133         GPIO.output(7, GPIO.HIGH)
134         time.sleep(0.1)
135         GPIO.output(9, GPIO.LOW)
136         GPIO.output(7, GPIO.LOW)
137
138
139     if (m == ['volumedown']) :
140         GPIO.output(9, GPIO.LOW)
141         GPIO.output(11, GPIO.LOW)
142         GPIO.output(8, GPIO.LOW)
143         GPIO.output(7, GPIO.LOW)
144         time.sleep(0.001)
145         GPIO.output(9, GPIO.LOW)
146         GPIO.output(11, GPIO.HIGH)
147         GPIO.output(8, GPIO.HIGH)
148         GPIO.output(7, GPIO.LOW)
149         time.sleep(0.1)
150         GPIO.output(11, GPIO.LOW)
151         GPIO.output(8, GPIO.LOW)
152
153
154     if (m == ['exit']) :
155         GPIO.output(9, GPIO.LOW)
156         GPIO.output(11, GPIO.LOW)
157         GPIO.output(8, GPIO.LOW)
158         GPIO.output(7, GPIO.LOW)
159         break
160
```

## ❖ Python Program of Obstacles avoider mode:

```
# -*- coding: utf-8 -*-
# Import required Python libraries
import time
import RPi.GPIO as GPIO
import pylirc

# Use BCM GPIO references
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
# Define GPIO to use on Pi
TRIG = 27
ECHO = 22
# Set pins as output and input
GPIO.setup(TRIG,GPIO.OUT)      # Trig
GPIO.setup(ECHO,GPIO.IN)       # Echo
GPIO.setup(9, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)

GPIO.output(9, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(8, GPIO.LOW)
GPIO.output(7, GPIO.LOW)
#IR buttons configuration
s = pylirc.init("myprogram")
#Servo ready
GPIO.setup(18, GPIO.OUT)
p = GPIO.PWM(18,50)
p.start(6.2)
time.sleep(0.5)
p.ChangeDutyCycle(6.2)
time.sleep(1)
p = GPIO.PWM(18,1)

print "Ultrasonic Measurement"
#Set TRIG as LOW
GPIO.output(TRIG, False)
print "Waitng For Sensor To Settle"
time.sleep(1)

while True:
    m = pylirc.nextcode()
    print m
    if (m == ['power']) :
        while True:
                    #Obstacle Test
            GPIO.output(TRIG, False)                 #Set TRIG as LOW
            print "Waitng For Sensor To Settle"
            time.sleep(0.1)                          #Delay of 2 seconds

            GPIO.output(TRIG, True)                  #Set TRIG as HIGH
            time.sleep(0.00001)                      #Delay of 0.00001 seconds
            GPIO.output(TRIG, False)                 #Set TRIG as LOW

            while GPIO.input(ECHO)==0:               #Check whether the ECHO is LOW
                pulse_start = time.time()            #Saves the last known time of LOW

            while GPIO.input(ECHO)==1:               #Check whether the ECHO is HIGH
                pulse_end = time.time()              #Saves the last known time of HIGH

            pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable

            distance = pulse_duration * 17150        #Multiply pulse duration by 17150
            distance = round(distance, 2)
            print "Distance:",distance - 0.5,"cm"
```

```
68
69              if distance > 2 and distance < 400:
70              #Check whether the distance is within range
71                  print "Distance:",distance - 0.5,"cm"
72                  #Print distance with - 0.5 cm for calibration
73              else:
74                  print "Out Of Range"                    #display out of range
75              #End Test
76
77              n = pylirc.nextcode()
78              print n
79
80              if (n == ['record']) :
81                    GPIO.output(25, GPIO.HIGH)
82
83              if (n == ['cut']) :
84                    GPIO.output(25, GPIO.LOW)
85
86      ##
87              if (distance>30) :
88                     GPIO.output(9, GPIO.HIGH)
89                     GPIO.output(11, GPIO.LOW)
90                     GPIO.output(8, GPIO.HIGH)
91                     GPIO.output(7, GPIO.LOW)
92      ##
93              if (distance<30) :
94                     GPIO.output(9, GPIO.LOW)
95                     GPIO.output(11, GPIO.LOW)
96                     GPIO.output(8, GPIO.LOW)
97                     GPIO.output(7, GPIO.LOW)
98                     time.sleep(1)
99
100                   p = GPIO.PWM(18,50)
101                   p.start(6.2)
102                   time.sleep(0.1)
103                   p.ChangeDutyCycle(2.6)
104                   time.sleep(1)
105                   p = GPIO.PWM(18,1)
106
107     ##
108                   #Obstacle Test
109                   GPIO.output(TRIG, False)
110                   print "Waitng For Sensor To Settle"
111                   time.sleep(0.1)
112
113                   GPIO.output(TRIG, True)
114                   time.sleep(0.00001)
115                   GPIO.output(TRIG, False)
116
117                   while GPIO.input(ECHO)==0:
118                       pulse_start = time.time()
119
120                   while GPIO.input(ECHO)==1:
121                       pulse_end = time.time()
122
123                   pulse_duration = pulse_end - pulse_start
124
125                   distance = pulse_duration * 17150
126                   distance = round(distance, 2)
127                   print "Distance:",distance - 0.5,"cm"
```

```
128
129             if distance > 2 and distance < 400:
130             #Check whether the distance is within range
131                 print "Distance:",distance - 0.5,"cm"
132                 #Print distance with - 0.5 cm for calibration
133             else:
134                 print "Out Of Range"                      #display out of range
135             #End Test
136
137             if (distance<30) :
138                 oright=1
139             else:
140                 oright=0
141
142             print 'oright=',oright
143
144             p = GPIO.PWM(18,50)
145             p.start(2.8)
146             time.sleep(0.1)
147             p.ChangeDutyCycle(10.3)
148             time.sleep(1)
149             p = GPIO.PWM(18,1)
150
151             #Obstacle Test
152             GPIO.output(TRIG, False)                 #Set TRIG as LOW
153             print "Waitng For Sensor To Settle"
154             time.sleep(0.1)                            #Delay of 2 seconds
155
156             GPIO.output(TRIG, True)                  #Set TRIG as HIGH
157             time.sleep(0.00001)                      #Delay of 0.00001 seconds
158             GPIO.output(TRIG, False)                 #Set TRIG as LOW
159
160         while GPIO.input(ECHO)==0:
161             pulse_start = time.time()
162
163         while GPIO.input(ECHO)==1:
164             pulse_end = time.time()
165
166         pulse_duration = pulse_end - pulse_start
167
168         distance = pulse_duration * 17150
169         distance = round(distance, 2)
170
171         if distance > 2 and distance < 400:
172         #Check whether the distance is within ran
173             print "Distance:",distance - 0.5,"cm"
174             #Print distance with - 0.5 cm for cal
175         else:
176             print "Out Of Range"                ibration
177         #End Test
178
179         if (distance<30) :
180             oleft=1
181         else:
182             oleft=0
183
184         print 'oleft=',oleft
185
186         p = GPIO.PWM(18,50)
187         p.start(10)
188         time.sleep(0.1)
189         p.ChangeDutyCycle(6.2)
190         time.sleep(1)
191         p = GPIO.PWM(18,1)
192
```

```
192
193                     if oright==0 :
194                         if oleft==1:
195                             GPIO.output(9, GPIO.HIGH)
196                             GPIO.output(11, GPIO.LOW)
197                             GPIO.output(8, GPIO.LOW)
198                             GPIO.output(7, GPIO.HIGH)
199                             time.sleep(0.38)
200                             GPIO.output(9, GPIO.LOW)
201                             GPIO.output(11, GPIO.LOW)
202                             GPIO.output(8, GPIO.LOW)
203                             GPIO.output(7, GPIO.LOW)
204                             time.sleep(1)
205
206                     if oleft==0:
207                         if oright==1:
208                             GPIO.output(9, GPIO.LOW)
209                             GPIO.output(11, GPIO.HIGH)
210                             GPIO.output(8, GPIO.HIGH)
211                             GPIO.output(7, GPIO.LOW)
212                             time.sleep(0.367)
213                             GPIO.output(9, GPIO.LOW)
214                             GPIO.output(11, GPIO.LOW)
215                             GPIO.output(8, GPIO.LOW)
216                             GPIO.output(7, GPIO.LOW)
217                             time.sleep(1)
218
219                     if oright==1:
220                         if oleft==1:
221                             GPIO.output(9, GPIO.LOW)
222                             GPIO.output(11, GPIO.HIGH)
223                             GPIO.output(8, GPIO.LOW)
224                             GPIO.output(7, GPIO.HIGH)
225                             time.sleep(1.5)
226                             GPIO.output(9, GPIO.LOW)
227                             GPIO.output(11, GPIO.LOW)
228                             GPIO.output(8, GPIO.LOW)
229                             GPIO.output(7, GPIO.LOW)
230                             time.sleep(1)
231                             GPIO.output(9, GPIO.HIGH)
232                             GPIO.output(11, GPIO.LOW)
233                             GPIO.output(8, GPIO.LOW)
234                             GPIO.output(7, GPIO.HIGH)
235                             time.sleep(0.70)
236                             GPIO.output(9, GPIO.LOW)
237                             GPIO.output(11, GPIO.LOW)
238                             GPIO.output(8, GPIO.LOW)
239                             GPIO.output(7, GPIO.LOW)
240                             time.sleep(1)
241
242                 if (n == ['exit']) :
243                     GPIO.output(9, GPIO.LOW)
244                     GPIO.output(11, GPIO.LOW)
245                     GPIO.output(8, GPIO.LOW)
246                     GPIO.output(7, GPIO.LOW)
247                     break
248
```

## ❖ Python Program of Web App mode (Wi-Fi control):

```
import RPi.GPIO as GPIO
import time
from flask import Flask, render_template, request
app = Flask(__name__)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Create a dictionary called pins to store the pin number, name, and pin state:
pins = {
   19 : {'name' : 'FARWARD', 'state' : GPIO.LOW},
   16 : {'name' : 'BACKWARD', 'state' : GPIO.LOW},
   26 : {'name' : 'RIGHT', 'state' : GPIO.LOW},
   20 : {'name' : 'LEFT', 'state' : GPIO.LOW},
   21 : {'name' : 'obs', 'state' : GPIO.LOW},
   17 : {'name' : 'STOP', 'state' : GPIO.LOW},
   9 : {'name' : 'FARWARD', 'state' : GPIO.LOW},
   11 : {'name' : 'BACKWARD', 'state' : GPIO.LOW},
   8 : {'name' : 'RIGHT', 'state' : GPIO.LOW},
   7 : {'name' : 'LEFT', 'state' : GPIO.LOW}
   }

# Set each pin as an output and make it low:
for pin in pins:
   GPIO.setup(pin, GPIO.OUT)
   GPIO.output(pin, GPIO.LOW)
@app.route("/")
def main():
   # For each pin, read the pin state and store it in the pins dictionary:
   for pin in pins:
      pins[pin]['state']= GPIO.input(pin)
   # Put the pin dictionary into the template data dictionary:
   templateData = {
      'pins' : pins
      }
   # Pass the template data into the template main.html and return it to the user
   return render_template('main.html', **templateData)


# The function below is executed when someone requests a URL with the pin number
@app.route("/<changePin>/<action>")
def action(changePin, action):
   # Convert the pin from the URL into an integer:
   changePin = int(changePin)
   # Get the device name for the pin being changed:
   deviceName = pins[changePin]['name']
   # If the action part of the URL is "on," execute the code indented below:

   if deviceName == 'obs':
      GPIO.output(19, GPIO.LOW)
      GPIO.output(16, GPIO.LOW)
      GPIO.output(26, GPIO.LOW)
      GPIO.output(20, GPIO.LOW)
      GPIO.output(17, GPIO.HIGH)
      GPIO.output(9, GPIO.LOW)
      GPIO.output(11, GPIO.LOW)
      GPIO.output(8, GPIO.LOW)
      GPIO.output(7, GPIO.LOW)
      message = "Turned " + deviceName + " on."
```

```
59
60      if deviceName == 'FARWARD':
61          # Set the pin high:
62          GPIO.output(19, GPIO.HIGH)
63          GPIO.output(16, GPIO.LOW)
64          GPIO.output(26, GPIO.LOW)
65          GPIO.output(20, GPIO.LOW)
66          GPIO.output(17, GPIO.LOW)
67          GPIO.output(9, GPIO.LOW)
68          GPIO.output(11, GPIO.LOW)
69          GPIO.output(8, GPIO.LOW)
70          GPIO.output(7, GPIO.LOW)
71          time.sleep(0.1)
72          GPIO.output(9, GPIO.HIGH)
73          GPIO.output(11, GPIO.LOW)
74          GPIO.output(8, GPIO.HIGH)
75          GPIO.output(7, GPIO.LOW)
76          # Save the status message to be passed into the template:
77          message = "Turned " + deviceName + " on."
78
79      if deviceName == 'BACKWARD' :
80          GPIO.output(19, GPIO.LOW)
81          GPIO.output(16, GPIO.HIGH)
82          GPIO.output(26, GPIO.LOW)
83          GPIO.output(20, GPIO.LOW)
84          GPIO.output(17, GPIO.LOW)
85          GPIO.output(9, GPIO.LOW)
86          GPIO.output(11, GPIO.LOW)
87          GPIO.output(8, GPIO.LOW)
88          GPIO.output(7, GPIO.LOW)
89          time.sleep(0.1)
90          GPIO.output(9, GPIO.LOW)
91          GPIO.output(11, GPIO.HIGH)
92          GPIO.output(8, GPIO.LOW)
93          GPIO.output(7, GPIO.HIGH)
94          message = "Turned " + deviceName + " on."
95
96      if deviceName == 'RIGHT' :
97          GPIO.output(19, GPIO.LOW)
98          GPIO.output(16, GPIO.LOW)
99          GPIO.output(26, GPIO.HIGH)
100         GPIO.output(20, GPIO.LOW)
101         GPIO.output(17, GPIO.LOW)
102         GPIO.output(9, GPIO.LOW)
103         GPIO.output(11, GPIO.LOW)
104         GPIO.output(8, GPIO.LOW)
105         GPIO.output(7, GPIO.LOW)
106         time.sleep(0.1)
107         GPIO.output(9, GPIO.HIGH)
108         GPIO.output(11, GPIO.LOW)
109         GPIO.output(8, GPIO.LOW)
110         GPIO.output(7, GPIO.HIGH)
111         time.sleep(0.38)
112         GPIO.output(9, GPIO.LOW)
113         GPIO.output(7, GPIO.LOW)
114         message = "Turned " + deviceName + " on."
115
```

```
115     if deviceName == 'LEFT' :
116         GPIO.output(19, GPIO.LOW)
117         GPIO.output(16, GPIO.LOW)
118         GPIO.output(26, GPIO.LOW)
119         GPIO.output(20, GPIO.HIGH)
120         GPIO.output(17, GPIO.LOW)
121         GPIO.output(9, GPIO.LOW)
122         GPIO.output(11, GPIO.LOW)
123         GPIO.output(8, GPIO.LOW)
124         GPIO.output(7, GPIO.LOW)
125         time.sleep(0.1)
126         GPIO.output(9, GPIO.LOW)
127         GPIO.output(11, GPIO.HIGH)
128         GPIO.output(8, GPIO.HIGH)
129         GPIO.output(7, GPIO.LOW)
130         time.sleep(0.367)
131         GPIO.output(11, GPIO.LOW)
132         GPIO.output(8, GPIO.LOW)
133         message = "Turned " + deviceName + " on."
134
135     if deviceName == 'STOP' :
136         GPIO.output(19, GPIO.LOW)
137         GPIO.output(16, GPIO.LOW)
138         GPIO.output(26, GPIO.LOW)
139         GPIO.output(20, GPIO.LOW)
140         GPIO.output(17, GPIO.HIGH)
141         GPIO.output(9, GPIO.LOW)
142         GPIO.output(11, GPIO.LOW)
143         GPIO.output(8, GPIO.LOW)
144         GPIO.output(7, GPIO.LOW)
145         message = "Turned " + deviceName + " on."
146
147     # For each pin, read the pin state and store it in the pins dictionary:
148     for pin in pins:
149         pins[pin]['state'] = GPIO.input(pin)
150
151     # Along with the pin dictionary, put the message into the template data
152     templateData = {
153         'pins' : pins
154     }
155
156     return render_template('main.html', **templateData)
157
158 if __name__ == "__main__":
159     app.run(host='0.0.0.0', port=80, debug=True)
160
```

## ❖ HTML Program of Web App Interface:

```
<!DOCTYPE html>
<head>

   <title>FOKS_BOT Web Server</title>
   <!-- Latest compiled and minified CSS -->
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au+a
   <!-- Optional theme -->
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css" integrity="sha384-fLW2N01lMqj
   <!-- Latest compiled and minified JavaScript -->
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PR



</head>

<body style="background-color:powderblue;">
   <h1 style="text-align:center;color:blue;font-size:300%;">FOKS-BOT WEB CONTROL</h1>
   <br>
   <TABLE BORDER="0" style="margin-left: 120px">
   <TR>
   <FORM><TD>
   <p style="color:red;font-size:150%;margin-left: 180px;">:::CONTROL BUTTONS:::</p>
   {% if pins[9].state == true %}
      <div class="row"><div class="col-md-4">
      <a  href="/{{9}}/off" class="btn btn-block btn-lg btn-default" style="margin-left: 225px" role="button">FORWARD</a></div></div>
   {% else %}
      <div class="row"><div class="col-md-4">
      <a href="/{{9}}/on" class="btn btn-block btn-lg btn-default" style="margin-left: 225px" role="button">FORWARD</a></div></div>
   {% endif %}
   <br>
   <br>

   <TABLE BORDER="0">
   <TR>
   <TD><FORM>
   {% if pins[7].state == true %}
      <div class="row"><div class="col-md-2">
      <a href="/{{7}}/off" class="btn btn-inline btn-lg btn-default" style="margin-left: 100px" role="button">  LEFT  </a></div></div>
   {% else %}
      <div class="row"><div class="col-md-2">
      <a href="/{{7}}/on" class="btn btn-inline btn-lg btn-default" style="margin-left: 100px" role="button">  LEFT  </a></div></div>
   {% endif %}
   </FORM></TD>
```

```
<TD><FORM>
{% if pins[17].state == true %}
  <div class="row"><div class="col-md-2">
  <a href="/{{17}}/off" class="btn btn-inline btn-lg btn-default" style="margin-left: 90px" role="button">  STOP  </a></div></div>
{% else %}
  <div class="row"><div class="col-md-2">
  <a href="/{{17}}/on" class="btn btn-inline btn-lg btn-default" style="margin-left: 90px" role="button">  STOP  </a></div></div>
{% endif %}
</FORM></TD>
<TD><FORM>
{% if pins[8].state == true %}
  <div class="row"><div class="col-md-2">
  <a href="/{{8}}/off" class="btn btn-inline btn-lg btn-default" style="margin-left: 70px" role="button">  RIGHT  </a></div></div>
{% else %}
  <div class="row"><div class="col-md-2">
  <a href="/{{8}}/on" class="btn btn-inline btn-lg btn-default" style="margin-left: 70px" role="button">  RIGHT  </a></div></div>
{% endif %}
</FORM></TD>
</TR>
</TABLE>
<br>
<br>
{% if pins[11].state == true %}
  <div class="row"><div class="col-md-4">
  <a href="/{{11}}/off" class="btn btn-block btn-lg btn-default" style="margin-left: 225px" role="button">BACKWARD</a></div></div>
{% else %}
  <div class="row"><div class="col-md-4">
  <a href="/{{11}}/on" class="btn btn-block btn-lg btn-default" style="margin-left: 225px" role="button">BACKWARD</a></div></div>
{% endif %}
</FORM></TD>

<TD><FORM>
<p style="color:red;font-size:150%;margin-left: 240px;">:::LIVE CAMERA:::</p>
<div>
<iframe width="338" height="450" style="margin-left: 120px" src="http://192.168.43.1:8080/browserfs.html">
</iframe>
</div>

  <div class="form-group">
     <label for="range_zoom" style="margin-left: 120px"  class="col-xs-1 control-label">Zoom </label>
     <div class="col-xs-7">
       <input id="range_zoom" class="col-xs-5" type="range" min="0" max="20" value="0">
     </div>
     <label for="range_zoom" id="range_zoom_label" class="control-label col-xs-1"></label>
   </div>
</FORM></TD>
</TR>
</TABLE>

<br>
<p style="text-align:center;color:blue;font-size:250%;">::: DESIGNED BY GUESBAYA MOUNIR :::</p>
</body>
</html>
```

## ❖ Python Program of Line Follower mode with Raspberry Pi:

```
import RPi.GPIO as GPIO
import time
import pylirc

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(14, GPIO.IN)
GPIO.setup(15, GPIO.IN)
GPIO.setup(18, GPIO.IN)

GPIO.setup(9, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(2,GPIO.OUT)
GPIO.setup(3,GPIO.OUT)

P = GPIO.PWM(2,50)
P.start(80)

p = GPIO.PWM(3,50)
p.start(75)

s = pylirc.init("myprogram")

GPIO.output(9, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(8, GPIO.LOW)
GPIO.output(7, GPIO.LOW)

d=0
g=0
M=0
while True :

    m = pylirc.nextcode()
    print m

    if (m == ['mode']) :
        while True:
                r=GPIO.input(14)
                c=GPIO.input(15)
                l=GPIO.input(18)
                print c
                m = pylirc.nextcode()

                if   (m == ['exit']):
                        print "EXIT STOP"
                        GPIO.output(9, GPIO.LOW)
                        GPIO.output(11, GPIO.LOW)
                        GPIO.output(8, GPIO.LOW)
                        GPIO.output(7, GPIO.LOW)
                        break

                if (m == ['up']):
                        GPIO.output(9, GPIO.HIGH)
                        GPIO.output(11, GPIO.LOW)
                        GPIO.output(8, GPIO.HIGH)
                        GPIO.output(7, GPIO.LOW)
                        time.sleep(0.3)
                        GPIO.output(9, GPIO.LOW)
                        GPIO.output(11, GPIO.LOW)
                        GPIO.output(8, GPIO.LOW)
                        GPIO.output(7, GPIO.LOW)
```

```
66
67                  if l==0:
68                      g=0
69                  else:
70                      g=1
71                      print "g=",g
72
73                  if r==0:
74                      d=0
75                  else:
76                      d=1
77                      print "d=",d
78
79                  if c==1:
80                      d=0
81                      g=0
82                      M=1
83
84                  if  (g==1) and (l==1):
85                          print "LEFT Black"
86                          GPIO.output(9, GPIO.LOW)
87                          GPIO.output(11, GPIO.LOW)
88                          GPIO.output(8, GPIO.LOW)
89                          GPIO.output(7, GPIO.LOW)
90                          time.sleep(0.1)
91                          GPIO.output(9, GPIO.LOW)
92                          GPIO.output(11, GPIO.LOW)
93                          GPIO.output(8, GPIO.HIGH)
94                          GPIO.output(7, GPIO.LOW)
95                          time.sleep(0.1)
96                          GPIO.output(9, GPIO.LOW)
97                          GPIO.output(11, GPIO.LOW)
98                          GPIO.output(8, GPIO.LOW)
99                          GPIO.output(7, GPIO.LOW)
100                         g=1
101                         d=0
102                         M=0
103
104                 if  (c==1):
105                     if (l==1):
106                         print "LEFT Black"
107                         GPIO.output(9, GPIO.LOW)
108                         GPIO.output(11, GPIO.LOW)
109                         GPIO.output(8, GPIO.LOW)
110                         GPIO.output(7, GPIO.LOW)
111                         time.sleep(0.1)
112                         GPIO.output(9, GPIO.LOW)
113                         GPIO.output(11, GPIO.LOW)
114                         GPIO.output(8, GPIO.HIGH)
115                         GPIO.output(7, GPIO.LOW)
116                         time.sleep(0.1)
117                         GPIO.output(9, GPIO.LOW)
118                         GPIO.output(11, GPIO.LOW)
119                         GPIO.output(8, GPIO.LOW)
120                         GPIO.output(7, GPIO.LOW)
121                         g=1
122                         d=0
123                         M=0
124
125                 if (d==1) and (r==1):
126                         print "RIGHT Black"
127                         GPIO.output(9, GPIO.LOW)
128                         GPIO.output(11, GPIO.LOW)
129                         GPIO.output(8, GPIO.LOW)
130                         GPIO.output(7, GPIO.LOW)
131                         time.sleep(0.1)
132                         GPIO.output(9, GPIO.HIGH)
```

```
132                     GPIO.output(9, GPIO.HIGH)
133                     GPIO.output(11, GPIO.LOW)
134                     GPIO.output(8, GPIO.LOW)
135                     GPIO.output(7, GPIO.LOW)
136                     time.sleep(0.1)
137                     GPIO.output(9, GPIO.LOW)
138                     GPIO.output(11, GPIO.LOW)
139                     GPIO.output(8, GPIO.LOW)
140                     GPIO.output(7, GPIO.LOW)
141                     d=1
142                     g=0
143                     M=0
144
145                 if (c==1):
146                     if (r==1):
147                         print "RIGHT Black"
148                         GPIO.output(9, GPIO.LOW)
149                         GPIO.output(11, GPIO.LOW)
150                         GPIO.output(8, GPIO.LOW)
151                         GPIO.output(7, GPIO.LOW)
152                         time.sleep(0.1)
153                         GPIO.output(9, GPIO.HIGH)
154                         GPIO.output(11, GPIO.LOW)
155                         GPIO.output(8, GPIO.LOW)
156                         GPIO.output(7, GPIO.LOW)
157                         time.sleep(0.1)
158                         GPIO.output(9, GPIO.LOW)
159                         GPIO.output(11, GPIO.LOW)
160                         GPIO.output(8, GPIO.LOW)
161                         GPIO.output(7, GPIO.LOW)
162                         d=1
163                         g=0
164                         M=0
165
166                 if (M==1):
167                     if (r==0):
168                         if (l==0):
169                             print "Middel Black"
170                             GPIO.output(9, GPIO.HIGH)
171                             GPIO.output(11, GPIO.LOW)
172                             GPIO.output(8, GPIO.HIGH)
173                             GPIO.output(7, GPIO.LOW)
174
175
176
177                 if (c==1):
178                     if (r==1):
179                         if (l==1):
180                             print "ALL BLACK STOP"
181                             GPIO.output(9, GPIO.LOW)
182                             GPIO.output(11, GPIO.LOW)
183                             GPIO.output(8, GPIO.LOW)
184                             GPIO.output(7, GPIO.LOW)
185                             break
186
187
```

## ❖ Python Program of Line Follower mode with Raspberry Pi and Arduino Uno:

```
import RPi.GPIO as GPIO
import time
import pylirc
import serial
ser = serial.Serial('/dev/ttyACM0',  115200, timeout = 0.1)

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(13, GPIO.IN)
GPIO.setup(6, GPIO.IN)
GPIO.setup(5, GPIO.IN)

GPIO.setup(9, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(2,GPIO.OUT)
GPIO.setup(3,GPIO.OUT)

P = GPIO.PWM(2,50)
P.start(80)

p = GPIO.PWM(3,50)
p.start(75)

s = pylirc.init("myprogram")

GPIO.output(9, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(8, GPIO.LOW)
GPIO.output(7, GPIO.LOW)

from nanpy import Arduino
from nanpy import serial manager

Arduino.pinMode(13, Arduino.OUTPUT)
Arduino.pinMode(12, Arduino.INPUT)
Arduino.pinMode(11, Arduino.INPUT)
Arduino.pinMode(8, Arduino.INPUT)
d=0
g=0

while True :

    m = pylirc.nextcode()
    print m

    if (m == ['mode']) :
        while True:
                r=Arduino.digitalRead(8)
                c=Arduino.digitalRead(11)
                l=Arduino.digitalRead(12)

                m = pylirc.nextcode()

                if   (m == ['exit']):
                        print "EXIT STOP"
                        GPIO.output(9, GPIO.LOW)
                        GPIO.output(11, GPIO.LOW)
                        GPIO.output(8, GPIO.LOW)
                        GPIO.output(7, GPIO.LOW)
                        break

                if (m == ['volumeup']):
                        GPIO.output(9, GPIO.HIGH)
                        GPIO.output(11, GPIO.LOW)
                        GPIO.output(8, GPIO.HIGH)
                        GPIO.output(7, GPIO.LOW)
                        time.sleep(0.3)
                        GPIO.output(9, GPIO.LOW)
                        GPIO.output(11, GPIO.LOW)
                        GPIO.output(8, GPIO.LOW)
```

```
76
77                  if l==0:
78                      g=0
79                  else:
80                      g=1
81                      print "g=",g
82
83                  if r==0:
84                      d=0
85                  else:
86                      d=1
87                      print "d=",d
88
89                  if  (g==1):
90                          print "LEFT Black"
91                          GPIO.output(9, GPIO.LOW)
92                          GPIO.output(11, GPIO.LOW)
93                          GPIO.output(8, GPIO.LOW)
94                          GPIO.output(7, GPIO.LOW)
95                          time.sleep(0.1)
96                          GPIO.output(9, GPIO.LOW)
97                          GPIO.output(11, GPIO.LOW)
98                          GPIO.output(8, GPIO.HIGH)
99                          GPIO.output(7, GPIO.LOW)
100                         g=1
101                         d=0
102
103                 if (d==1):
104                         print "RIGHT Black"
105                         GPIO.output(9, GPIO.LOW)
106                         GPIO.output(11, GPIO.LOW)
107                         GPIO.output(8, GPIO.LOW)
108                         GPIO.output(7, GPIO.LOW)
109                         time.sleep(0.1)
110                         GPIO.output(9, GPIO.HIGH)
111                         GPIO.output(11, GPIO.LOW)
112                         GPIO.output(8, GPIO.LOW)
113                         GPIO.output(7, GPIO.LOW)
114                       d=1
115                       g=0
116
117                 if (c==1):
118                     if (r==0):
119                         if (l==0):
120                             print "Middel Black"
121                             GPIO.output(9, GPIO.HIGH)
122                             GPIO.output(11, GPIO.LOW)
123                             GPIO.output(8, GPIO.HIGH)
124                             GPIO.output(7, GPIO.LOW)
125
126                 if (c==1):
127                     if (r==1):
128                         if (l==1):
129                             print "ALL BLACK STOP"
130                             GPIO.output(9, GPIO.LOW)
131                             GPIO.output(11, GPIO.LOW)
132                             GPIO.output(8, GPIO.LOW)
133                             GPIO.output(7, GPIO.LOW)
134                             break
135
136
```

## ❖ Python Program of Multiple control mode:

```
# -*- coding: utf-8 -*-
# Import required Python libraries
import time
import RPi.GPIO as GPIO
import pylirc

from nanpy import Arduino
from nanpy import serial_manager
from time import sleep

# Use BCM GPIO references
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Set Arduino Uno pins as output and input
Arduino.pinMode(12, Arduino.OUTPUT)
##Arduino.pinMode(11, Arduino.INPUT)
##Arduino.pinMode(8, Arduino.INPUT)


# Set Rspi LINE FOLLOWER pins as  input
GPIO.setup(14, GPIO.IN)
GPIO.setup(15, GPIO.IN)
GPIO.setup(18, GPIO.IN)


# Define GPIO to use on Pi
TRIG = 27
ECHO = 22

# Set Raspberry Pi pins as output and input
GPIO.setup(TRIG,GPIO.OUT)      # Trig
GPIO.setup(ECHO,GPIO.IN)       # Echo
GPIO.setup(21, GPIO.OUT)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(19, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(26, GPIO.OUT)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)

d=0
g=0
M=0

# Set PWM pins as output or input
GPIO.setup(2,GPIO.OUT)
GPIO.setup(3,GPIO.OUT)

p = GPIO.PWM(2,50)
p.start(100)

P = GPIO.PWM(3,50)
P.start(100)

GPIO.output(9, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(8, GPIO.LOW)
GPIO.output(7, GPIO.LOW)

s = pylirc.init("myprogram")

GPIO.output(24, GPIO.LOW)
m = GPIO.PWM(24,50)
m.start(6.2)
time.sleep(0.5)
m.ChangeDutyCycle(6.2)
time.sleep(1)
```

```
m.ChangeDutyCycle(6.2)
time.sleep(1)
m = GPIO.PWM(24,1)


GPIO.output(TRIG, False)                 #Set TRIG as LOW
##Waitng For Sensor To Settle
time.sleep(1)

def distance():
    #Obstacle Test
    GPIO.output(TRIG, False)                 #Set TRIG as LOW
    print "Waitng For Sensor To Settle"
    time.sleep(0.1)                            #Delay of 2 seconds
    GPIO.output(TRIG, True)                  #Set TRIG as HIGH
    time.sleep(0.00001)                      #Delay of 0.00001 seconds
    GPIO.output(TRIG, False)                 #Set TRIG as LOW

    while GPIO.input(ECHO)==0:               #Check whether the ECHO is LOW
        pulse_start = time.time()              #Saves the last known time of LOW pulse

    while GPIO.input(ECHO)==1:               #Check whether the ECHO is HIGH
        pulse_end = time.time()                #Saves the last known time of HIGH pulse

    pulse_duration = pulse_end - pulse_start #Get pulse duration to a variab

    distance = pulse_duration * 17150        #Multiply pulse duration by 17150 to get distance
    distance = round(distance, 2)
    print "Distance:",distance - 0.5,"cm"

    if distance > 2 and distance < 400:
        print "Distance:",distance - 0.5,"cm"  #Print distance with 0.5 cm calibration
    else:
        print "Out Of Range"                   #display out of range
    #End Test
    return distance

while True:
    print "SELECT YOUR MODE"
    m = pylirc.nextcode()
    print m
    if (m == ['one']) :
        print "IR MODE is activated with feature of detecting obstacles"
        p = GPIO.PWM(2,50)
        p.start(90)

        P = GPIO.PWM(3,50)
        P.start(80)

        while True:
            n = pylirc.nextcode()
            print n

            if (distance() < 30) :
                GPIO.output(21, GPIO.HIGH)
                GPIO.output(19, GPIO.LOW)
                GPIO.output(16, GPIO.LOW)
                GPIO.output(26, GPIO.LOW)
                GPIO.output(20, GPIO.LOW)
                GPIO.output(25, GPIO.LOW)
                GPIO.output(17, GPIO.HIGH)
                GPIO.output(9, GPIO.LOW)
                GPIO.output(11, GPIO.LOW)
                GPIO.output(8, GPIO.LOW)
                GPIO.output(7, GPIO.LOW)
            else:
                GPIO.output(21, GPIO.LOW)
                GPIO.output(25, GPIO.HIGH)
```

```
142
143             if (n == ['up']) :
144                 GPIO.output(9, GPIO.LOW)
145                 GPIO.output(11, GPIO.LOW)
146                 GPIO.output(8, GPIO.LOW)
147                 GPIO.output(7, GPIO.LOW)
148                 time.sleep(0.1)
149                 GPIO.output(9, GPIO.HIGH)
150                 GPIO.output(11, GPIO.LOW)
151                 GPIO.output(8, GPIO.HIGH)
152                 GPIO.output(7, GPIO.LOW)
153
154             if (n == ['down']) :
155                 GPIO.output(9, GPIO.LOW)
156                 GPIO.output(11, GPIO.LOW)
157                 GPIO.output(8, GPIO.LOW)
158                 GPIO.output(7, GPIO.LOW)
159                 time.sleep(0.1)
160                 GPIO.output(9, GPIO.LOW)
161                 GPIO.output(11, GPIO.HIGH)
162                 GPIO.output(8, GPIO.LOW)
163                 GPIO.output(7, GPIO.HIGH)
164
165             if (n == ['left']) :
166                 GPIO.output(9, GPIO.LOW)
167                 GPIO.output(11, GPIO.LOW)
168                 GPIO.output(8, GPIO.LOW)
169                 GPIO.output(7, GPIO.LOW)
170                 time.sleep(0.1)
171                 GPIO.output(9, GPIO.LOW)
172                 GPIO.output(11, GPIO.HIGH)
173                 GPIO.output(8, GPIO.HIGH)
174                 GPIO.output(7, GPIO.LOW)
175                 time.sleep(0.54)
176                 GPIO.output(11, GPIO.LOW)
177                 GPIO.output(8, GPIO.LOW)
178
179             if (n == ['right']) :
180                 GPIO.output(9, GPIO.LOW)
181                 GPIO.output(11, GPIO.LOW)
182                 GPIO.output(8, GPIO.LOW)
183                 GPIO.output(7, GPIO.LOW)
184                 time.sleep(0.1)
185                 GPIO.output(9, GPIO.HIGH)
186                 GPIO.output(11, GPIO.LOW)
187                 GPIO.output(8, GPIO.LOW)
188                 GPIO.output(7, GPIO.HIGH)
189                 time.sleep(0.58)
190                 GPIO.output(9, GPIO.LOW)
191                 GPIO.output(7, GPIO.LOW)
192
193             if (n == ['ok']) :
194                 GPIO.output(9, GPIO.LOW)
195                 GPIO.output(11, GPIO.LOW)
196                 GPIO.output(8, GPIO.LOW)
197                 GPIO.output(7, GPIO.LOW)
198
199
200             if (n == ['l']) :
201                 GPIO.output(9, GPIO.LOW)
202                 GPIO.output(11, GPIO.LOW)
203                 GPIO.output(8, GPIO.LOW)
204                 GPIO.output(7, GPIO.LOW)
205                 time.sleep(0.1)
206                 GPIO.output(9, GPIO.LOW)
207                 GPIO.output(11, GPIO.HIGH)
208                 GPIO.output(8, GPIO.HIGH)
209                 GPIO.output(7, GPIO.LOW)
210
211             if (n == ['r']) :
212                 GPIO.output(9, GPIO.LOW)
213                 GPIO.output(11, GPIO.LOW)
214                 GPIO.output(8, GPIO.LOW)
215                 GPIO.output(7, GPIO.LOW)
```

```
216                     time.sleep(0.1)
217                     GPIO.output(9, GPIO.HIGH)
218                     GPIO.output(11, GPIO.LOW)
219                     GPIO.output(8, GPIO.LOW)
220                     GPIO.output(7, GPIO.HIGH)
221
222                 if (n == ['volumeup']) :
223                     GPIO.output(9, GPIO.LOW)
224                     GPIO.output(11, GPIO.LOW)
225                     GPIO.output(8, GPIO.LOW)
226                     GPIO.output(7, GPIO.LOW)
227                     time.sleep(0.1)
228                     GPIO.output(9, GPIO.HIGH)
229                     GPIO.output(11, GPIO.LOW)
230                     GPIO.output(8, GPIO.LOW)
231                     GPIO.output(7, GPIO.HIGH)
232                     time.sleep(0.1)
233                     GPIO.output(9, GPIO.LOW)
234                     GPIO.output(7, GPIO.LOW)
235
236
237                 if (n == ['volumedown']) :
238                     GPIO.output(9, GPIO.LOW)
239                     GPIO.output(11, GPIO.LOW)
240                     GPIO.output(8, GPIO.LOW)
241                     GPIO.output(7, GPIO.LOW)
242                     time.sleep(0.1)
243                     GPIO.output(9, GPIO.LOW)
244                     GPIO.output(11, GPIO.HIGH)
245                     GPIO.output(8, GPIO.HIGH)
246                     GPIO.output(7, GPIO.LOW)
247                     time.sleep(0.1)
248                     GPIO.output(11, GPIO.LOW)
249                     GPIO.output(8, GPIO.LOW)
250
251                 if (n == ['record']) :
252                     Arduino.digitalWrite(12, Arduino.HIGH)
253
254                 if (n == ['cut']) :
255                     Arduino.digitalWrite(12, Arduino.LOW)
256
257
258                 if (n == ['exit']) :
259                     GPIO.output(21, GPIO.LOW)
260                     GPIO.output(25, GPIO.LOW)
261                     GPIO.output(9, GPIO.LOW)
262                     GPIO.output(11, GPIO.LOW)
263                     GPIO.output(8, GPIO.LOW)
264                     GPIO.output(7, GPIO.LOW)
265                     break
266
267     if (m == ['two']) :
268         print "OBSTACLES AVOIDANCE MODE is activated"
269         p = GPIO.PWM(2,50)
270         p.start(100)
271
272         P = GPIO.PWM(3,50)
273         P.start(93)
274
275         while True:
276             #Obstacle Test
277             GPIO.output(TRIG, False)                 #Set TRIG as LO
278             print "Waitng For Sensor To Settle"
279             time.sleep(0.1)                            #Delay of 2 s
280
281             GPIO.output(TRIG, True)                  #Set TRIG as HI
282             time.sleep(0.00001)                      #Delay of 0.000
283             GPIO.output(TRIG, False)                 #Set TRIG as LO
284
285             while GPIO.input(ECHO)==0:               #Check whether
286                 pulse_start = time.time()              #Saves the la
287
```

```
287
288             while GPIO.input(ECHO)==1:                  #Check whethe
289                 pulse_end = time.time()                   #Saves the
290
291             pulse_duration = pulse_end - pulse_start #Get pulse du
292
293             distance = pulse_duration * 17150           #Multiply pul
294             distance = round(distance, 2)
295             print "Distance:",distance - 0.5,"cm"
296
297             if distance > 2 and distance < 400:         #Check whethe
298                 print "Distance:",distance - 0.5,"cm"   #Print dist
299             else:
300                 print "Out Of Range"                      #display ou
301             #End Test
302
303             n = pylirc.nextcode()
304             print n
305
306             if (distance>30) :
307                 GPIO.output(21, GPIO.LOW)
308                 GPIO.output(19, GPIO.LOW)
309                 GPIO.output(16, GPIO.LOW)
310                 GPIO.output(26, GPIO.LOW)
311                 GPIO.output(20, GPIO.LOW)
312                 GPIO.output(25, GPIO.HIGH)
313                 GPIO.output(17, GPIO.HIGH)
314                 GPIO.output(9, GPIO.HIGH)
315                 GPIO.output(11, GPIO.LOW)
316                 GPIO.output(8, GPIO.HIGH)
317                 GPIO.output(7, GPIO.LOW)
318
319             if (distance<30) :
320                     GPIO.output(9, GPIO.LOW)
321                     GPIO.output(11, GPIO.LOW)
322                     GPIO.output(8, GPIO.LOW)
323                     GPIO.output(7, GPIO.LOW)
324                     GPIO.output(21, GPIO.HIGH)
325                     GPIO.output(25, GPIO.LOW)
326
327                     time.sleep(1)
328
329                     m = GPIO.PWM(24,50)
330                     m.start(6.2)
331                     time.sleep(0.1)
332                     m.ChangeDutyCycle(2.6)
333                     time.sleep(1)
334                     m = GPIO.PWM(24,1)
335
336
337                     #Obstacle Test
338                     GPIO.output(TRIG, False)
339                     print "Waitng For Sensor To Settle"
340                     time.sleep(0.1)
341
342                     GPIO.output(TRIG, True)
343                     time.sleep(0.00001)
344                     GPIO.output(TRIG, False)
345
346                     while GPIO.input(ECHO)==0:
347                         pulse_start = time.time()
348
349                     while GPIO.input(ECHO)==1:
350                         pulse_end = time.time()
351
352                     pulse_duration = pulse_end - pulse_start
353
354                     distance = pulse_duration * 17150
355                     distance = round(distance, 2)
356                     print "Distance:",distance - 0.5,"cm"
357
358                     if distance > 2 and distance < 400:
359                         print "Distance:",distance - 0.5,"cm"
360                     else:
```

```
359                 print "Distance:",distance - 0.5,"cm"  #Pri
360             else:
361                 print "Out Of Range"                    #dis
362             #End Test
363
364             if (distance<30) :
365                 oright=1
366             else:
367                 oright=0
368
369             print 'oright=',oright
370
371             m = GPIO.PWM(24,50)
372             m.start(2.8)
373             time.sleep(0.1)
374             m.ChangeDutyCycle(10.3)
375             time.sleep(1)
376             m = GPIO.PWM(24,1)
377
378             #Obstacle Test
379             GPIO.output(TRIG, False)                    #Set T
380             print "Waitng For Sensor To Settle"
381             time.sleep(0.1)                               #Del
382
383             GPIO.output(TRIG, True)                     #Set T
384             time.sleep(0.00001)                         #Delay
385             GPIO.output(TRIG, False)                    #Set T
386
387             while GPIO.input(ECHO)==0:                  #Check
388                 pulse_start = time.time()                 #Sav
389
390             while GPIO.input(ECHO)==1:                  #Check
391                 pulse_end = time.time()                   #Sav
392
393             pulse_duration = pulse_end - pulse_start #Get p
394
395         distance = pulse_duration * 17150           #Multipli
396         distance = round(distance, 2)               #Round unc
397
398         if distance > 2 and distance < 400:          #Check
399             print "Distance:",distance - 0.5,"cm"  #Prin
400         else:
401             print "Out Of Range"                      #disp
402         #End Test
403
404         if (distance<30) :
405             oleft=1
406         else:
407             oleft=0
408
409         print 'oleft=',oleft
410
411         m = GPIO.PWM(24,50)
412         m.start(10)
413         time.sleep(0.1)
414         m.ChangeDutyCycle(6.2)
415         time.sleep(1)
416         m = GPIO.PWM(24,1)
417
418         if oright==0 :
419             if oleft==1:
420                 GPIO.output(9, GPIO.HIGH)
421                 GPIO.output(11, GPIO.LOW)
422                 GPIO.output(8, GPIO.LOW)
423                 GPIO.output(7, GPIO.HIGH)
424                 time.sleep(0.51)
425                 GPIO.output(9, GPIO.LOW)
426                 GPIO.output(11, GPIO.LOW)
427                 GPIO.output(8, GPIO.LOW)
428                 GPIO.output(7, GPIO.LOW)
429                 time.sleep(1)
430
431         if oleft==0:
432             if oright==1:
```

```
                    if oleft==0:
                        if oright==1:
                            GPIO.output(9, GPIO.LOW)
                            GPIO.output(11, GPIO.HIGH)
                            GPIO.output(8, GPIO.HIGH)
                            GPIO.output(7, GPIO.LOW)
                            time.sleep(0.54)
                            GPIO.output(9, GPIO.LOW)
                            GPIO.output(11, GPIO.LOW)
                            GPIO.output(8, GPIO.LOW)
                            GPIO.output(7, GPIO.LOW)
                            time.sleep(1)

                    if oright==1:
                        if oleft==1:
                            GPIO.output(9, GPIO.LOW)
                            GPIO.output(11, GPIO.HIGH)
                            GPIO.output(8, GPIO.LOW)
                            GPIO.output(7, GPIO.HIGH)
                            time.sleep(1.5)
                            GPIO.output(9, GPIO.LOW)
                            GPIO.output(11, GPIO.LOW)
                            GPIO.output(8, GPIO.LOW)
                            GPIO.output(7, GPIO.LOW)
                            time.sleep(0.5)
                            GPIO.output(9, GPIO.HIGH)
                            GPIO.output(11, GPIO.LOW)
                            GPIO.output(8, GPIO.LOW)
                            GPIO.output(7, GPIO.HIGH)
                            time.sleep(0.85)
                            GPIO.output(9, GPIO.LOW)
                            GPIO.output(11, GPIO.LOW)
                            GPIO.output(8, GPIO.LOW)
                            GPIO.output(7, GPIO.LOW)
                            time.sleep(0.5)

            if (n == ['exit']) :
                GPIO.output(9, GPIO.LOW)
                GPIO.output(11, GPIO.LOW)
                GPIO.output(8, GPIO.LOW)
                GPIO.output(7, GPIO.LOW)
                break


    if (m == ['three']) :
        print "LINE FOLLOWER MODE is activated with feature of detecting obstacles"
        P = GPIO.PWM(2,50)
        P.start(100)
        p = GPIO.PWM(3,50)
        p.start(100)

        d=0
        g=0
        M=0

        while True :
        ##          if (distance() <30) :
        ##              print "D<30 STOP"
        ##              GPIO.output(9, GPIO.LOW)
        ##              GPIO.output(11, GPIO.LOW)
        ##              GPIO.output(8, GPIO.LOW)
        ##              GPIO.output(7, GPIO.LOW)
        ##              GPIO.output(21, GPIO.HIGH)
        ##              GPIO.output(25, GPIO.LOW)
        ##
        ##          else:
        ##              GPIO.output(21, GPIO.LOW)
        ##              GPIO.output(19, GPIO.LOW)
        ##              GPIO.output(16, GPIO.LOW)
        ##              GPIO.output(26, GPIO.LOW)
        ##              GPIO.output(20, GPIO.LOW)
        ##              GPIO.output(25, GPIO.HIGH)
        ##              GPIO.output(17, GPIO.HIGH)
```

```
504
505
506                     r=GPIO.input(14)
507                     c=GPIO.input(15)
508                     l=GPIO.input(18)
509
510                     m = pylirc.nextcode()
511                     print m
512                     if   (m == ['exit']):
513                                 print "EXIT STOP"
514                                 GPIO.output(9, GPIO.LOW)
515                                 GPIO.output(11, GPIO.LOW)
516                                 GPIO.output(8, GPIO.LOW)
517                                 GPIO.output(7, GPIO.LOW)
518                                 break
519
520                     if (m == ['up']):
521                                 GPIO.output(9, GPIO.HIGH)
522                                 GPIO.output(11, GPIO.LOW)
523                                 GPIO.output(8, GPIO.HIGH)
524                                 GPIO.output(7, GPIO.LOW)
525                                 time.sleep(0.3)
526                                 GPIO.output(9, GPIO.LOW)
527                                 GPIO.output(11, GPIO.LOW)
528                                 GPIO.output(8, GPIO.LOW)
529                                 GPIO.output(7, GPIO.LOW)
530
531                     if l==0:
532                                 g=0
533                     else:
534                                 g=1
535                                 print "g=",g
536
537                     if r==0:
538                                 d=0
539                     else:
540                                 d=1
541                                 print "d=",d
542
543                     if c==1:
544                                 d=0
545                                 g=0
546                                 M=1
547
548                     if   (g==1) and (l==1):
549                                 print "LEFT Black"
550                                 P.ChangeDutyCycle(20)
551                                 p.ChangeDutyCycle(85)
552                                 g=1
553                                 d=0
554                                 M=0
555
556                     if   (c==1):
557                                 if (l==1):
558                                     print "Middel and LEFT Black=TURN LEFT"
559                                     P.ChangeDutyCycle(20)
560                                     p.ChangeDutyCycle(85)
561                                     g=1
562                                     d=0
563                                     M=0
564
565                     if (d==1) and (r==1):
566                                 print "RIGHT Black"
567                                 P.ChangeDutyCycle(90)
568                                 p.ChangeDutyCycle(20)
569                                 d=1
570                                 g=0
571                                 M=0
572
573                     if (c==1):
574                                 if (r==1):
575                                     print "Middel and RIGHT Black=TURN RIGH
576                                     P.ChangeDutyCycle(90)
577                                     p.ChangeDutyCycle(20)
```

```
576                                 P.ChangeDutyCycle(90)
577                                 p.ChangeDutyCycle(20)
578                                 d=1
579                                 g=0
580                                 M=0
581
582                         if (M==1):
583                                 if (r==0):
584                                     if (l==0):
585                                         print "Middel Black=Forward"
586                                         P.ChangeDutyCycle(90)
587                                         p.ChangeDutyCycle(83)
588                                         GPIO.output(9, GPIO.HIGH)
589                                         GPIO.output(8, GPIO.HIGH)
590
591
592
593                         if (c==1):
594                                 if (r==1):
595                                     if (l==1):
596                                         print "ALL BLACK STOP"
597                                         GPIO.output(9, GPIO.LOW)
598                                         GPIO.output(8, GPIO.LOW)
599                                         break
600
601
602
```