**Summary**

The aim behind this project is to develop a robot to perform the act of surveillance in domestic areas. Nowadays robot plays a vital role in our day to day life activities thus reducing human labor and human error. Robots can be manually controlled. The purpose of this robot is to roam around and provide video information from the given environment and to send that obtained information to the user. In this project, one can control the robot with the help of mobile or laptop through Internet of Things (IoT) and also can get the live streaming of video both in daytime as well as at night with the help of web camera from the robot. The robot can be controlled with the help of Raspberry pi . This robot also uses various sensors that collects data and sends it to the Raspberry pi and user which used to control the robot behavior. Along with the obtained live streamed video output, user can also obtain the temperature and humidity. Fan is used when temperature is high it can be turn on. Its work both on daytime as well as night time LED light is used for watching live streaming in night time. Thus, the action of surveillance can be performed. Further advancement in our project can provide surveillance even in defence areas.

Table of Contents

[Chapter 1 | Software and Hardware Requirement Specification 1](#_Toc17811927)

[1.1 Introduction 1](#_Toc17811928)

[1.2 Brief Description 1](#_Toc17811929)

[1.3 Block Diagram 2](#_Toc17811930)

[1.4 Scope of the project 3](#_Toc17811931)

[1.5 Literature Review 3](#_Toc17811932)

[Chapter 2 | Software and hardware Requirement Analysis 4](#_Toc17811933)

[2.1 Raspberry pi 3 B+ 4](#_Toc17811934)

[2.2 Processor / SoC (System on Chip) 5](#_Toc17811935)

[2.2.1 Power source 5](#_Toc17811936)

[2.2.2 SD Card 5](#_Toc17811937)

[2.2.3 GPIO – General Purpose Input Output 5](#_Toc17811938)

[2.2.4 DSI Connector 6](#_Toc17811939)

[2.2.5 RCA Video 6](#_Toc17811940)

[2.2.6 Audio Jack 6](#_Toc17811941)

[2.2.7 USB 2.0 Port 6](#_Toc17811942)

[2.2.8 Ethernet 7](#_Toc17811943)

[2.2.9 CSI connector 7](#_Toc17811944)

[2.2.10 JTAG headers 7](#_Toc17811945)

[2.2.11 HDMI – High Definition Multimedia Interface 7](#_Toc17811946)

[2.2.12 802.11ac Wi-Fi and Bluetooth 4.1 7](#_Toc17811947)

[2.3 Advantages and disadvantages 7](#_Toc17811948)

[2.3.1 Advantages of the Raspberry Pi 7](#_Toc17811949)

[2.3.2 Disadvantages 8](#_Toc17811950)

[2.4 Internet Device 8](#_Toc17811951)

[2.5 5V Power Bank 9](#_Toc17811952)

[2.6 Ultrasonic Sensor 10](#_Toc17811953)

[2.6.1 How Ultrasonic Sensors Work. 10](#_Toc17811954)

[2.7 Web Camera 11](#_Toc17811955)

[2.8 DC Motor 11](#_Toc17811956)

[2.8.1 How Dc Motor Work 12](#_Toc17811957)

[2.9 Relay Module 12](#_Toc17811958)

[2.9.1 Types of relay module 13](#_Toc17811959)

[2.9.2 Relay Pinout 13](#_Toc17811960)

[2.9.3 Mains voltage connections 13](#_Toc17811961)

[2.9.4 Pin wiring 14](#_Toc17811962)

[2.10 5V LED 14](#_Toc17811963)

[2.11 Temperature and Humidity Sensor (DHT11) 15](#_Toc17811964)

[2.11.1 Working of DHT11 15](#_Toc17811965)

[2.11.2 Specifications of DHT11 16](#_Toc17811966)

[2.12 Servo Motor 17](#_Toc17811967)

[2.12.1 How is the servo controlled? 17](#_Toc17811968)

[2.13 Nginx Server 19](#_Toc17811969)

[2.14 uWSGI 19](#_Toc17811970)

[2.15 Motion Server 19](#_Toc17811971)

[2.16 Python 19](#_Toc17811972)

[2.17 Flask 20](#_Toc17811973)

[2.18 Thonny 20](#_Toc17811974)

[2.19 HTML/CSS 20](#_Toc17811975)

[2.20 Java Script 21](#_Toc17811976)

[2.21 Toastr 21](#_Toc17811977)

[2.22 jQuery 21](#_Toc17811978)

[2.23 Ajax 21](#_Toc17811979)

[2.24 Debian Raspberry PI Language 21](#_Toc17811980)

[2.25 Sublime Text 22](#_Toc17811981)

[2.26 Linux 22](#_Toc17811982)

[2.27 WinSCP 22](#_Toc17811983)

[2.28 VNC Viewer 22](#_Toc17811984)

[2.29 Putty 22](#_Toc17811985)

[2.30 Fritzing 23](#_Toc17811986)

[2.31 Code OF Senosr 23](#_Toc17811987)

[Chapter 3 | System Design 33](#_Toc17811988)

[3.1 The 4wd Robot Chasis 33](#_Toc17811989)

[3.2 Assemble the robot chasis 34](#_Toc17811990)

[3.3 Design the website 38](#_Toc17811991)

[3.4 Design the communication strategy 40](#_Toc17811992)

[Chapter 4 | Implementation 44](#_Toc17811993)

[4.1 Circuit Diagram 44](#_Toc17811994)

[4.2 Install Raspbian on Raspberry pi 44](#_Toc17811995)

[4.3 Port Forwarding 48](#_Toc17811996)

[4.4 Working 51](#_Toc17811997)

[4.5 Screenshot of working 54](#_Toc17811998)

[4.6 Code of Project 57](#_Toc17811999)

[4.6.1 HTML Part 57](#_Toc17812000)

[4.6.2 CSS Part 61](#_Toc17812001)

[4.7 JS Part 65](#_Toc17812002)

[Chapter 5 | System Testing 73](#_Toc17812003)

[5.1 Combining of Modules 73](#_Toc17812004)

[5.2 Unit Testing 73](#_Toc17812005)

[5.3 Discussion 73](#_Toc17812006)

[5.4 Final Product 74](#_Toc17812007)

[5.5 Future Work 74](#_Toc17812008)

[5.6 Conclusions 75](#_Toc17812009)

[5.7 References 76](#_Toc17812010)

# 

# Software and Hardware Requirement Specification

## Introduction

Raspberry pi controlled multi-environment robot with live streaming and temperature monitoring. The project aims at monitoring environmental conditions with live streaming and to measure surrounding temperature. The system consist of a small robot, free to move itself in a place is a able to measure surrounding environment quality. It can track temperature in the way to help measuring correct working temperature for machines as well as human beings E.g. In Industries. Also the environment can be visualized through the camera present on the robot. This robot could be used for in commercial spaces, schools and hospitals, warehouses and other challenging indoor and outdoor environments

For live streaming. It will allow people to ﬁnd the best place for them, for their health and it will imply more places responsible to improve the environment quality outdoor as well as indoor. The project makes use of Raspberry Pi 3 Model B + as the main MCU which is interfaced with web camera which helps in live streaming and DHT11 temperature

sensor module which helps in measuring the surrounding temperature. The web Camera captures video and DHT11 sensor helps in monitoring the temperature of the environment and the data is displayed on the server through web.

## Brief Description

This IOT base surveillance robot system mainly consists of Nine modules.

1. Sensors(DHT11)
2. Power Bank
3. Website
4. Dc motor
5. Relay Module
6. Raspberry pi 3b+
7. Horn
8. Fan
9. LED

System generally consists of sensors, power bank, website, dc Motor, Microcontroller Raspberry pi, Relay module. Dht11 sensor used for measuring the temperature and humidity. DC motors are used for control the movement of the robot such as forward , reverse , left , right and stop with the help of relay module . Relay module also used for control the fan and LED.

## Block Diagram

The following Fig 1.1 shows the block diagram of the whole system.

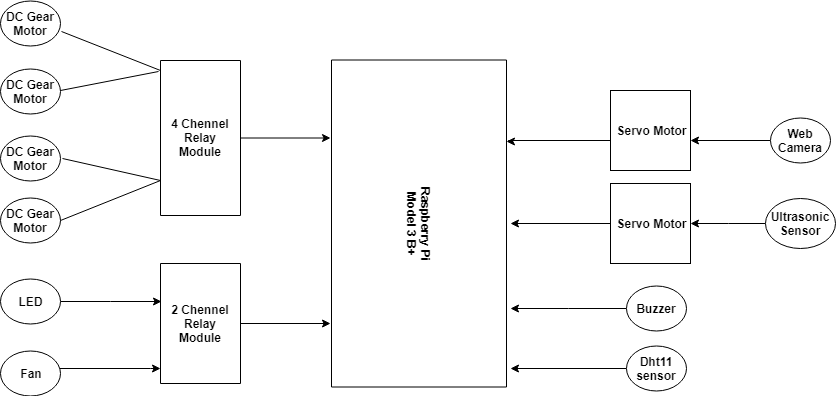


Figure 1.1 Block Diagram

1. Dc gear motor – Control the movement of robot
2. Relay module – Control the dc motor, LED and fan
3. Gateway – Raspberry pi Microcontroller
4. Servo Motor – Control the movement of camera and ultrasonic sensor
5. Dht11 sensors– Sense the temperature and humidity

## Scope of the project

The current ongoing revolution of Internet of Things (IoT), is now integrated with Robotics in various diverse fields of everyday life is making up new era i.e. Internet of Robotics (IoR). Internet of Robotics is on the mature stage of development and is currently surrounded by various challenges to be solved for more implementations, i.e., design, security, sensors, and long-range communication systems.

To fully exploit the potential of advanced technology, i.e., IoT in the coming years, more research would be done in both protocol development, energy efficiency, more sensors integration, and integration of long wireless communication modules.

## Literature Review

This project deals with the survey of Smart surveillance monitoring system using Raspberry pi. Video Surveillance is important as far as security is concerned these days. Commercial spaces, schools and hospitals, warehouses and other challenging indoor and outdoor environments require high end cameras. The current technologies require RFIDs which are costly and hence the security domain in all becomes expensive and hence there was a need to work on this. This paper describes the use of low cost single on board computer Raspberry Pi. This new technology is less expensive and in this project it is used as a standalone platform for image processing. It increases the usage of mobile technology to provide essential security to our homes and for other control applications. The proposed home security system captures information and transmits it via a 3G Dongle to a Smart Phone and Laptop using web application.

The above system helps in monitoring commercial spaces, schools and hospitals, warehouses and other challenging indoor and outdoor environments which contributes towards security. This paper helps in creating low cost monitoring system hence proving more efficient. Further implementation can be done by attaching sensors to Raspberry Pi which will help in monitoring other aspects of the environment.

# Software and hardware Requirement Analysis

## Raspberry pi 3 B+

The Micro SD card is used for installing OS and the complete project will be done with python coding. The board has specification:

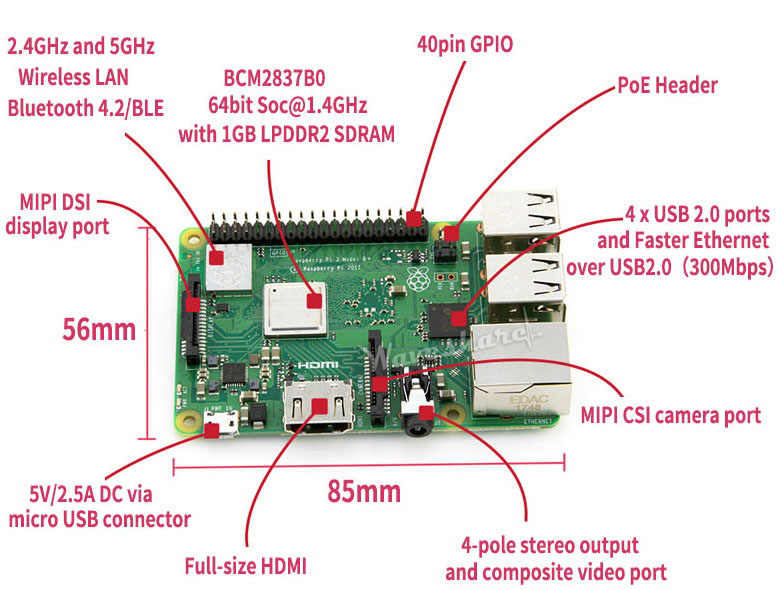


Figure 2.1 Rasberry pi

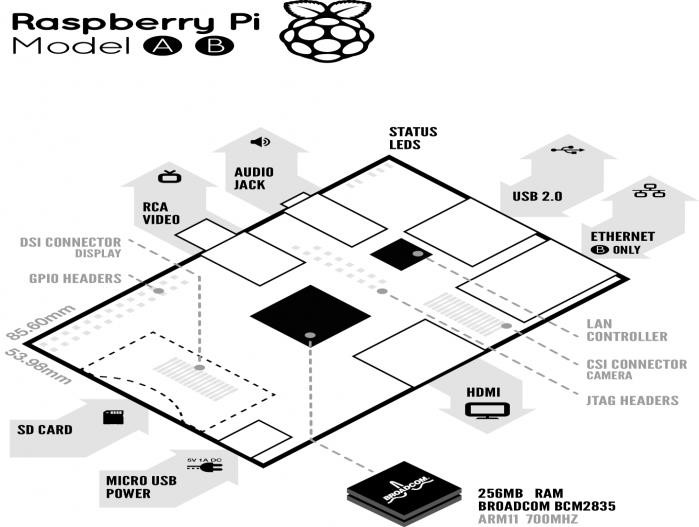


Figure 2.2 Raspberry pi sketch

## Processor / SoC (System on Chip)­­­

The Raspberry Pi has a Broadcom BCM2837 System on Chip module. It has an ARM Cortex-A53 1.4GHz processor. The Broadcom SoC used in the Raspberry Pi is equivalent to a chip used in an old smartphone (Android or iPhone). The Raspberry Pi chip operating at 700 MHz by default, will not become hot enough to need a heat sink or special cooling.

### Power source

The Pi is a device which consumes 700mA or 3W or power. It is powered by a MicroUSB charger or the GPIO header. Any good smartphone charger will do the work of powering the Pi.

### SD Card

The Raspberry Pi does not have any onboard storage available. The operating system is loaded on a SD card which is inserted on the SD card slot on the Raspberry Pi. The operating system can be loaded on the card using a card reader on any computer.

### GPIO – General Purpose Input Output

Figure 2.3 GPIO Pins

ARM Cortex-A53 1.4GHzGeneral-purpose input/output (GPIO) is a generic pin on an integrated circuit whose behavior, including whether it is an input or output pin, can be controlled by the user at run time.

GPIO capabilities may include:

GPIO pins can be configured to be input or output

GPIO pins can be enabled/disabled

Input values are readable (typically high=1, low=0)

Output values are writable/readable

Input values can often be used as IRQs (typically for wakeup events).

The production Raspberry Pi board has a 40-pin 2.54 mm (100 mil) expansion header, marked as P1, arranged in a 2x20 strip. They provide 21 GPIO pins plus access to I²C, SPI, UART), as well as +3.3 V, +5 V and GND supply lines. Pin one is the pin in the first column and on the bottom row.

### DSI Connector

The Display Serial Interface (DSI) is a specification by the Mobile Industry Processor Interface (MIPI) Alliance aimed at reducing the cost of display controllers in a mobile device. A DSI compatible LCD screen can be connected through the DSI connector, although it may require additional drivers to drive the display.

### RCA Video

RCA Video outputs (PAL and NTSC) are available on all models of Raspberry Pi. Any television or screen with a RCA jack can be connected with the RPi.



Figure 2.4 RCA Pin

### Audio Jack

A standard 3.5 mm TRS connector is available on the RPi for stereo audio output. Any headphone or 3.5mm audio cable can be connected directly. Although this jack cannot be used for taking audio input, USB mics or USB sound cards can be used.

### USB 2.0 Port

USB 2.0 ports are the means to connect accessories such as mouse or keyboard to the Raspberry Pi. There is 1 port on Model A, 2 on Model B and 4 on Model B+. The number of ports can be increased by using an external powered USB hub which is available as a standard Pi accessory.

### Ethernet

Ethernet port is available on Model B and B+. It can be connected to a network or internet using a standard LAN cable on the Ethernet port. The Ethernet ports are controlled by Microchip LAN9512

LAN controller chip.

### CSI connector

CSI – Camera Serial Interface is a serial interface designed by MIPI (Mobile Industry Processor Interface) alliance aimed at interfacing digital cameras with a mobile processor. The RPi foundation provides a camera specially made for the Pi which can be connected with the Pi using the CSI connector.

### JTAG headers

JTAG is an acronym for ‘Joint Test Action Group', an organization that started back in the mid 1980's to address test point access issues on PCB with surface mount devices. The organization devised a Many thousands of devices now include this standardized port as a feature to allow test and design engineers to access pins.

### HDMI – High Definition Multimedia Interface

HDMI 1.3 a type a port is provided on the RPi to connect with HDMI screens.

### 802.11ac Wi-Fi and Bluetooth 4.1

It's got wireless connectivity built in Wifi and Bluetooth card Adapter.

## Advantages and disadvantages

Advantages and disadvantages of Raspberry Pi from different aspect are described below:

### Advantages of the Raspberry Pi

This microcomputer is useful for small or home based businesses that run on a smaller budget than bigger companies for you are not required to purchase any special licenses from the Raspberry Pi Foundation to use their product or if you invent new technology that embeds the product.

The product does not require the user to have extensive programming experience since it is aimed for the younger generation to learn about programming. Python, the programming language that the Pi uses, is less complex than other languages available.

The SD cards on the board can be easily switched, which allows you to change the functions of the device without spending a lot of time re-installing the software

The Raspberry Pi is perfect for adaptive technology: it is able to display images or play videos at 1080p high definition resolution. This product makes it possible to build complex and effective products at a cheaper price.

This small credit card sized product makes it easy to recycle and does not release as much carbon dioxide emissions into the environment, unlike big servers that require lots of energy and extensive cooling systems.

### Disadvantages

It does not replace your computer, since the Ethernet is only a 10/100 and the processor is not as fast, it is time consuming to download and install software and is unable to do any complex multitasking.

Not compatible with other operating systems such as Windows (There are currently 1.3 billion Windows users around the world.)

This product will not be useful for bigger businesses that already have big servers, which would already do everything that the Raspberry Pi does, so it would not be worth it to take the time to get someone to put it together.

## Internet Device

I used PTCL internet wingle but you can used any kind of internet usb wingle as well as router and switch . there are some features of PTCL wingle listed below.

* Supports 1900 MHz frequency band.
* Supports CDMA2000 1X EVDO Rev. A / B Standard
* EVDO Rev. B data service of up to 9.3Mbps on downlink / 5.4Mbps on uplink\*.
* WLAN IEEE 802.11 b/g/n.
* Supports MicroSD Card up to 32GB.
* Hi-Link technology for auto-installation and easy use.
* Plug n’ share! No need to open any software, just plug in the USB to instantly start surfing or creating your own Wi-Fi hotspot.
* EVO USB Wi Fi modem
* Speeds up of to 9.3 Mbps in 200+ cities
* Connects up to 5 Wi-Fi enabled devices simultaneously



Figure 2.5 Evo Wingle

## 5V Power Bank

Raspberry Pi require power source to turn it on. 5V Power bank is enough to power up. In project I connect Power bank with raspberry Pi, so that it can be put easily in structure.



Figure 2.6 PowerBank

## Ultrasonic Sensor

This sensor is a high performance ultrasonic range finder. It is compact and measures an amazingly wide range from 2cm to 4m. This ranger is a perfect for any robotic application, or any other projects requiring accurate ranging information. This sensor can be connected directly to the digital I/O lines of your microcontroller and distance can be measured in time required for travelling of sound signal using simple formula as below. The module works on 5VDC input and also gives an output signal directly for detection of any obstacle up to 4M. As soon as the signals are transmitted the “Echo” pin goes to high level and remains in high level until the same sound waves are received by the receiver. If the received sound waves are same as what the same sensor transmitted then the Echo pin goes to low level. If no object is detected within 5M after 30ms the Echo signal will automatically go to low level.



Figure 2.7 Ultrasonic Sensor

### How Ultrasonic Sensors Work.

Ultrasonic sound vibrates at a frequency above the range of human hearing.Transducers are the microphones used to receive and send the ultrasonic sound. Our [ultrasonic sensors](https://www.maxbotix.com/SelectionGuide/Selection-Guide.htm), like many others, use a single transducer to send a pulse and to receive the echo.  The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulses.

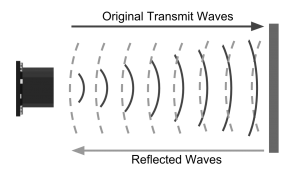


Figure 2.8 Ultrasonic sensor working

The working principle of this module is simple.  It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor.  By calculating the travel time and the speed of sound, the distance can be calculated.

## Web Camera

I used Techno HD webcam that have high quality COMS sensor 640\*480 and 12 mega pixel and support usb 2.0 as well as 1.1 but any company usb camera can be used . Pi cam also be used for that purpose that is directly connect to raspi rather than usb port



Figure 2.9 Web Camera

## DC Motor

Almost every mechanical movement that we see around us is accomplished by an electric motor. Electric machines are means of converting conventional energy. Motors take electrical energy and produce mechanical energy. Electric motor is used to power hundreds of devices we use in everyday life.



Figure 2.10 Dc motor

### How Dc Motor Work

Electric motors are the interface of electronic boards to almost every mechatronic and mechanical product including robots, drones, home appliances, etc.

A DC motor is an electrical machine devised to convert electrical energy into mechanical energy.

The main principle behind working a DC motor is the electromagnetic law according to which a current carrying conductor placed in magnetic field experiences a force, and the direction of the force is given by Fleming’s left-hand rule. In order to understand the basic principles, we need to understand the basic constructional features of a DC motor.

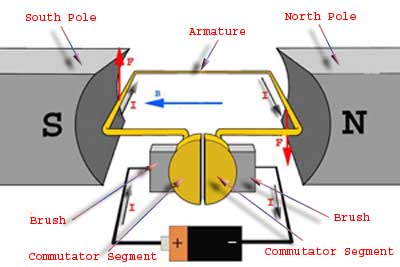
[](https://www.meee-services.com/quick-learning-dc-motors-work/)

Figure 2.11 Motor working

## Relay Module

This is a 5V 4-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equiped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller. I used this Relay board for controlling 4 DC Motor . I used Relay alternative of Motor H-bridge Module for control motor movement

This relay module has two channels (those blue cubes). There are other models with one, four and eight channels. This module should be powered with 5V, which is appropriate to use with an Arduino. There are other relay modules that are powered using 3.3V, which is ideal for ESP32, ESP8266, and other microcontrollers.

### Types of relay module

* [5V 2-channel relay module](https://makeradvisor.com/tools/5v-2-channel-relay-module-optocoupler/)
* [5V 1-channel relay module](https://www.banggood.com/custlink/vmv3RZoBhE)
* [5V 8-channel relay module](https://www.banggood.com/custlink/KKvGdZOgC9)
* [3.3V 1-channel relay module](https://www.banggood.com/custlink/vGm3dSjgWU)

### Relay Pinout

The following figure shows the relay module pinout.



Figure 2.12 Relay moudle

The six pins on the left side of the relay module connect high voltage, and the pins on the right side connect the component that requires low voltage—the Arduino pins.

### Mains voltage connections

The high-voltage side has two connectors, each with three sockets: common (COM), normally closed (NC), and normally open (NO).

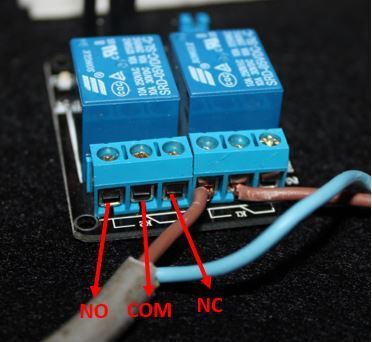


Figure 2.13 Relay connection

**COM**: common pin

**NO (Normally Open):** the normally open configuration works the other way around: the relay is always open, so the circuit is broken unless you send a signal from the Arduino to close the circuit.

If you just want to light up a lamp occasionally, it is better to use a normally-open circuit configuration.

### Pin wiring

The low-voltage side has a set of four pins and a set of three pins.

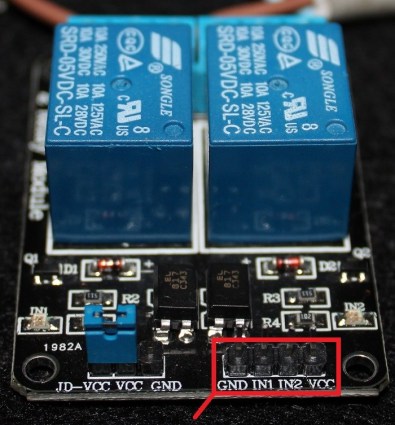


Figure 2.14 Relay Pins wiring

The set at the right consists of VCC and GND to power up the module, and input 1 (IN1) and input 2 (IN2) to control the bottom and top relays, respectively.

The second set of pins consists of GND, VCC, and JD-VCC pins. The JD-VCC pin powers the electromagnet of the relay.

## 5V LED

5v Led light strip used when light is low or off . It is use for watching the live streaming at night . its enable the user to watch live stream at day time as well as night time .

****

Figure 2.15 LED

## Temperature and Humidity Sensor (DHT11)

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old. I used DHT11 sensor for control the heat with Fan.

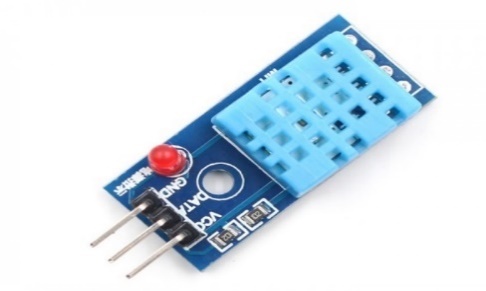
****

Figure 2.16 DHT11

### Working of DHT11

Lets start its working, it has two main part one is used for temperature measurement and other is used for humidity measurement. It also contain on IC which is used to send data to other Microcontroller. Now we discuss its working with details. First we discuss its Temperature Measurement Part.

**DHT11 Temperature Measurement Part**

For temperature measurement it consists of NTC sensor or thermistor which measure temperature.

Thermistor is basically a variable resistor which changes its resistance with change in temperature. Thermistors are made by sintering of semiconductors materials, such as ceramic or polymers in orders to large change in resistor with small change in temperature. when temperature changes there is a change in resistance of thermistor by this change we find temperature of our environment.

For better understanding lets see its picture

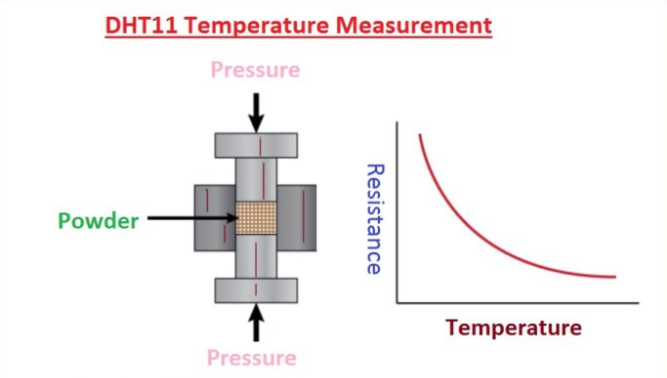


Figure 2.17 DHT Temperature Part

**DHT11 Humidity Measurement Part**

For Humidity Measurement it uses humidity measurement component, which has two electrodes with humidity holding substrate between them.

As moisture content changes in our environment, the conductivity of substrate changes or resistance between electrode changes. By this change in resistance we can find our outside environment humidity.



Figure 2.18 DHT Humidity Part

### Specifications of DHT11

These are some specification of DHT11, we discuss them with detailed.

* Its operating voltage are 3.5 v to 5.5 v.
* Its operating current are 0.3 to 5.5 mA.
* It send and receive data by serial transmission.
* Its operating temperature is -40°C to 80°C.
* Humidity range is 0% to 100%.
* It humidity and temperature resolution are 16-bit.
* Its accuracy is ±0.5°C and ±1%.
* Its sampling rate is 0.5 Hz.
* Its dimensions are 27mm x 59mm x 13.5mm (1.05″ x 2.32″ x 0.53″).
* It has four pins with 0.1″ spacing.

## Servo Motor

The servo motor is most commonly used for high technology devices in the industrial applications like automation technology. It is a self contained electrical device, that rotates parts of machine with high efficiency and great precision. Moreover the output shaft of this motor can be moved to a particular angle. Servo motors are mainly used in home electronics, toys, cars, airplanes and many more devices. I used 2 Servo motor one for movement of ultrasonic distance sensor and another one for web camera movement



Figure 2.19 Servo motor

### How is the servo controlled?

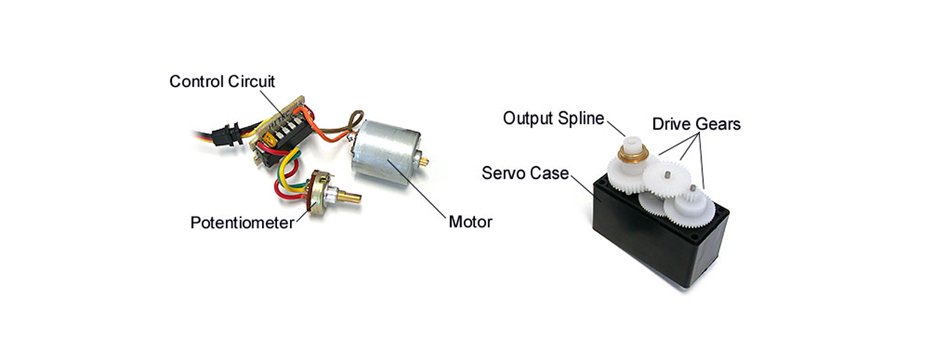


Figure 2.20 Servo working

Servos are controlled by sending an electrical pulse of variable width, or **pulse width modulation** (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the [**motor**](https://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&freeText=motor&search_type=jamecoall) determines position of the shaft, and based on the duration of the pulse sent via the control wire; the [**rotor**](https://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=cat_3540&subCategoryName=Electromechanical%20%2F%20Switches%20%2F%20Rotary&category=354055&refine=1&position=1&history=kv7hqebe%7CfreeText~rotor%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV%405hha4bcd%7Ccategory~35%5EcategoryName~category_root%5Eposition~1%5Erefine~1%5EsubCategoryName~Electromechanical%5EprodPage~50%5Epage~SEARCH%252BNAV) will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counter clockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position.

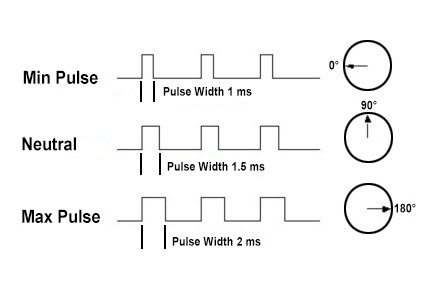


Figure 2.21 Servo Working

When these servos are commanded to move, they will move to the position and hold that position. If an external force pushes against the servo while the servo is holding a position, the servo will resist from moving out of that position. The maximum amount of force the servo can exert is called the **torque rating** of the servo.

It required number of Programming Tools , Languages and Servers to build a project. Nginx server, UWSGI server , Motion server, Python , Flask , HTML/CSS, Bootstrap, JavaScript, JQuery, Ajax, VNC server, VNC Viewer, WinSCP, and Putty.

## Nginx Server

**Nginx** is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and first publicly ­­­­­­­­­­­­­­­­­­­­­­released in 2004. A company of the same name was founded in 2011 to provide support and Nginx plus paid software. Nginx is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and first publicly released in 2004. A company of the same name was founded in 2011 to provide support and Nginx plus paid software . Nginx  is an open-source web server which server more 25% websites across the globe. I used Nginx for host my robot website

## uWSGI

**uWSGI** is a software application that "aims at developing a full stack for building hosting services". It is named after the Web Server Gateway Interface, which was the first plugin supported by the project . uWSGI is often used for serving python web application in conjuction with web servers such as Nginx which offer direct support for uWSGI's native uwsgi protocol.

## Motion Server

Motion is a highly configurable program that monitors video signals from many types of cameras.Set it up to monitor your security cameras, watch birds, check in on your pet, create timelapse videos and more. I used it for live streaming of my webcam

## Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991It is used for:

Figure 2.22 Python

* web development (server-side),
* software development,
* mathematics,
* system scripting.

**Uses of python**

Python can be used on a server to create web applications.

Python can be used alongside software to create workflows.

Python can connect to database systems. It can also read and modify files.

Python can be used to handle big data and perform complex mathematics.

Python can be used for rapid prototyping, or for production-ready software development.

## 1200px-Flask_logoFlask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[Flask](http://flask.pocoo.org/) ([source code](https://github.com/pallets/flask)) is a Python [web framework](https://www.fullstackpython.com/web-frameworks.html) built with a [small core and easy-to-extend philosophy](http://flask.pocoo.org/docs/design/). I used flask to make rest Api with python

Figure 2.23 Flask

## Thonny

Figure 2.24 Thonny

**Thonny** is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plugin.

## HTML/CSS

**Hypertext Mark-up Language**, commonly abbreviated as **HTML**, is the standard mark-up language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and web applications.

Figure 2.25 HTML

**Cascading Style Sheets (CSS)** is a style sheet language used for describing the presentation of a document written in a language. Although most often used to set the visual style of web pages and user interfaces written in HTML. I used HTML and CSS both for the design of website and web page to control the Robot.

Figure 2.26 CSS

## Java Script

**JavaScript** is a high-level, dynamic, untipped, and interpreted programming language. JavaScript's typing is dynamic. JavaScript is loaded as human-readable source code. JavaScript's are prototype-based.

## Toastr

**Toastr** is a Javascript library for non-blocking notifications. jQuery is required. The goal is to create a simple core library that can be customized and extended.i used toastr library for push notification at every event

## jQuery

Figure 2.27 jQuery

**JQuery** is a cross-platform JavaScript library designed to simplify the client- side scripting of HTML. JQuery is the most popular JavaScript in use today.

## Ajax

Ajax is [asynchronous JavaScript and XML) is a set of web](https://en.wikipedia.org/wiki/XML) development techniques using many web technologies on the client-side to create asynchronous Web applications. With Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. When any button is pressed to control the robot, the whole page will not refresh and action is executed with the help of Ajax.

## Debian Raspberry PI Language

**Debian** is a Unix-like computer operating system that is composed entirely of free software, most of which is under the GNU General Public License, and packaged by a group of individuals known as the Debian Project. Raspbian Jessi Lite version 4.1 is installed in Raspberry Pi 3B+.

Figure 2.28 Debian

## Sublime Text

**Sublime Text** is a proprietary cross-platform source code editor with a Python application programming interface. It natively supports many programming languages and markup languages, zand functions can be added by users with plugins. I used sublime text 3 for make my website

Figure 2.29 Sublime

## Linux

Figure 2.30 Linux

Linux is a Unix-like and mostly POSIX-complain computer operating system (OS) assembled under the model of free and open-source software development and distribution.

## WinSCP

WinSCP (Windows Secure Copy) is a free and open- source SFTP, FTP, WebDAV and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a remote computer. Beyond this, WinSCP offers basic manager and file synchronization functionality. For secure transfers, it uses Secure Shell (SSH) and supports the SCP protocol in addition to SFTP. Raspberry Pi to Laptop file transfer WinSCP is used in project.

Figure 2.31 WinSCP

## VNC Viewer

**VNC Viewer** from **RealVNC** provides you with instant remote access to your chosen computer; a Mac, a Windows PC or a Linux machine, from anywhere in the world .I use d VNC viewer to access GUI of raspberry pi remotely

Figure 2.32 VNC

## Putty

**Putty** is an SSH and telnet client, developed originally by Simon Tat ham for the Windows platform. Putty is open source software that is available with source code and is developed and supported by a group of volunteers. IP address of Raspberry Pi have to enter and raspberry pi is ready for programming.

Figure 2.33 Putty

## Fritzing

Fritzing is an [open-source hardware initiative](https://en.wikipedia.org/wiki/Open-source_hardware)that makes electronics accessible as a creative material for anyone. Fritizing is a software tool, a community website and services in the spirit of [Processing](https://processing.org/) and [Arduino](https://arduino.cc/) or Raspberry pi, prototypes, *share*them with others, *teach*electronicsin a classroom, and layout and *manufacture* professional pcbs.

## Code OF Senosr

1. import RPi.GPIO as GPIO
2. import time
3. from flask import Flask, jsonify
4. from flask\_cors import CORS, cross\_origin
5. from flask\_httpauth import HTTPBasicAuth
6. import Adafruit\_DHT
7. from time import sleep
8. app = Flask(\_\_name\_\_)
9. cors = CORS(app)
10. auth = HTTPBasicAuth()
11. users = {
12. "Talha" : "abc",
13. "Arslan" : "123",
14. }
15. employees = [
16. {
17. 'id': 1,
18. 'name': 'Talha'
19. },
20. {
21. 'id': 2,
22. 'title': 'Arslan'
23. }
24. ]
25. app.config['CORS\_HEADERS'] = 'Content-Type'
26. TRIG = 18
27. ECHO = 22
28. servo\_pin = 40
29. servo\_pin1 = 38
30. duty\_cycle = 7.5
31. GPIO.setmode(GPIO.BOARD)
32. GPIO.setup(TRIG, GPIO.OUT)
33. GPIO.setup(ECHO, GPIO.IN)
34. GPIO.setup(servo\_pin, GPIO.OUT)
35. GPIO.setup(11, GPIO.OUT)
36. GPIO.setup(13, GPIO.OUT)
37. GPIO.setup(15, GPIO.OUT)
38. GPIO.setup(16, GPIO.OUT)
39. GPIO.setup(33, GPIO.OUT)
40. GPIO.setup(35, GPIO.OUT)
41. GPIO.setup(36, GPIO.OUT)
42. GPIO.setup(38, GPIO.OUT)
43. GPIO.setwarnings(False)
44. @auth.get\_password
45. def GetPassword(username):
46. if(username in users):
47. return users.get(username)
48. return None
49. def ResetServoToFront():
50. GPIO.output(TRIG, False)
51. pwm\_servo = GPIO.PWM(servo\_pin, 50)
52. pwm\_servo.start(7.5)
53. pwm\_servo.ChangeDutyCycle(7.5)
54. time.sleep(0.1)
55. def SetAngle(angle):
56. pwm=GPIO.PWM(servo\_pin1, 50)
57. pwm.start(0)
58. duty = angle / 18 + 2
59. GPIO.output(servo\_pin1, True)
60. pwm.ChangeDutyCycle(duty)
61. sleep(1)
62. GPIO.output(servo\_pin1, False)
63. pwm.ChangeDutyCycle(0)
64. def ReadSensorReading():
65. GPIO.output(TRIG, True)
66. time.sleep(0.00001)
67. GPIO.output(TRIG, False)
68. while GPIO.input(ECHO) == 0:
69. pulse\_start = time.time()
70. while GPIO.input(ECHO) == 1:
71. pulse\_end = time.time()
72. pulse\_duration = pulse\_end - pulse\_start
73. distance = pulse\_duration \* 17150
74. distance = round(distance, 2)
75. return distance
76. def GetSensorReadingsWithServo():
77. GPIO.output(TRIG, False)
78. pwm\_servo = GPIO.PWM(servo\_pin, 50)
79. pwm\_servo.start(duty\_cycle)
80. pwm\_servo.ChangeDutyCycle(7.5)
81. time.sleep(0.1)
82. pwm\_servo.ChangeDutyCycle(13)
83. time.sleep(0.1)
84. distincmsleft = ReadSensorReading()
85. print "distincmsleft " + str(distincmsleft)
86. time.sleep(0.1)
87. pwm\_servo.ChangeDutyCycle(10)
88. time.sleep(0.1)
89. distincmsleft1 = ReadSensorReading()
90. print "distincmsleft1 " + str(distincmsleft1)
91. time.sleep(0.1)
92. pwm\_servo.ChangeDutyCycle(7.5)
93. time.sleep(0.1)
94. distincmsstraight = ReadSensorReading()
95. print "distincmsstraight " + str(distincmsstraight)
96. time.sleep(0.1)
97. pwm\_servo.ChangeDutyCycle(5)
98. time.sleep(0.1)
99. distincmsright1 = ReadSensorReading()
100. print "distincmsright1 " + str(distincmsright1)
101. time.sleep(0.1)
102. pwm\_servo.ChangeDutyCycle(3)
103. time.sleep(0.1)
104. distincmsright = ReadSensorReading()
105. print "distincmsright " + str(distincmsright)
106. time.sleep(0.1)
107. pwm\_servo.start(duty\_cycle)
108. time.sleep(0.1)
109. return distincmsleft, distincmsleft1, distincmsstraight, distincmsright1, distincmsright
110. def HornOn():
111. GPIO.output(33, 1)
112. def HornOff():
113. GPIO.output(33, 0)
114. def FanOn():
115. GPIO.output(36, 1)
116. def FanOff():
117. GPIO.output(36, 0)
118. def Forward():
119. GPIO.output(11, 1)
120. GPIO.output(13, 0)
121. GPIO.output(15, 1)
122. GPIO.output(16, 0)
123. ResetServoToFront()
124. def LightOn():
125. GPIO.output(35, 1)
126. def LightOff():
127. GPIO.output(35, 0)
128. def Reverse():
129. GPIO.output(11, 0)
130. GPIO.output(13, 1)
131. GPIO.output(15, 0)
132. GPIO.output(16, 1)
133. ResetServoToFront()
134. def Left():
135. GPIO.output(11, 1)
136. GPIO.output(13, 0)
137. GPIO.output(15, 0)
138. GPIO.output(16, 1)
139. ResetServoToFront()
140. def Right():
141. GPIO.output(11, 0)
142. GPIO.output(13, 1)
143. GPIO.output(15, 1)
144. GPIO.output(16, 0)
145. ResetServoToFront()
146. def Brake():
147. GPIO.output(11, 1)
148. GPIO.output(13, 1)
149. GPIO.output(15, 1)
150. GPIO.output(16, 1)
151. ResetServoToFront()
152. @app.route('/api/login', methods=['POST'])
153. @cross\_origin()
154. @auth.login\_required
155. def index():
156. return jsonify({'Login':'Success'})
157. @app.route('/api/employees', methods=['GET'])
158. def get\_users():
159. return jsonify({'employees': employees})
160. @app.route('/api/forward', methods=['GET'])
161. @cross\_origin()
162. @auth.login\_required
163. def forward():
164. try:
165. Forward()
166. return jsonify({'forward': "1"})
167. except:
168. return jsonify({'forward': "0"})
169. @app.route('/api/lighton', methods=['GET'])
170. @cross\_origin()
171. @auth.login\_required
172. def lighton():
173. try:
174. LightOn()
175. return jsonify({'light': "1"})
176. except:
177. return jsonify({'light': "-1"})
178. @app.route('/api/lightoff', methods=['GET'])
179. @cross\_origin()
180. @auth.login\_required
181. def lightoff():
182. try:
183. LightOff()
184. return jsonify({'light': "0"})
185. except:
186. return jsonify({'light': "-1"})
187. @app.route('/api/reverse', methods=['GET'])
188. @cross\_origin()
189. @auth.login\_required
190. def reverse():
191. try:
192. Reverse()
193. return jsonify({'reverse': "1"})
194. except:
195. return jsonify({'reverse': "0"})
196. @app.route('/api/left', methods=['GET'])
197. @cross\_origin()
198. @auth.login\_required
199. def left():
200. try:
201. for x in range(1, 3, 1):
202. Left()
203. time.sleep(0.1)
204. Brake()
205. return jsonify({'left': "1"})
206. except:
207. return jsonify({'left': "0"})
208. @app.route('/api/right', methods=['GET'])
209. @cross\_origin()
210. @auth.login\_required
211. def right():
212. try:
213. for x in range(1, 3, 1):
214. Right()
215. time.sleep(0.1)
216. Brake()
217. return jsonify({'right': "1"})
218. except:
219. return jsonify({'right': "0"})
220. @app.route('/api/brake', methods=['GET'])
221. @cross\_origin()
222. @auth.login\_required
223. def brake():
224. try:
225. Brake()
226. return jsonify({'brake': "1"})
227. except:
228. return jsonify({'brake': "0"})
229. @app.route('/api/seearound', methods=['GET'])
230. @cross\_origin()
231. @auth.login\_required
232. def seearound():
233. try:
234. SeeAround()
235. return jsonify({'seearound': "1"})
236. except:
237. return jsonify({'seearound': "-1"})
238. @app.route('/api/hornon', methods=['GET'])
239. @cross\_origin()
240. @auth.login\_required
241. def hornon():
242. try:
243. HornOn()
244. return jsonify({'Horn': "1"})
245. except:
246. return jsonify({'Horn': "-1"})
247. @app.route('/api/hornoff', methods=['GET'])
248. @cross\_origin()
249. @auth.login\_required
250. def hornoff():
251. try:
252. HornOff()
253. return jsonify({'Horn': "0"})
254. except:
255. return jsonify({'Horn': "-1"})
256. @app.route('/api/fanon', methods=['GET'])
257. @cross\_origin()
258. @auth.login\_required
259. def fanon():
260. try:
261. FanOn()
262. return jsonify({'Fan': "1"})
263. except:
264. return jsonify({'Fan': "-1"})
265. @app.route('/api/fanoff', methods=['GET'])
266. @cross\_origin()
267. @auth.login\_required
268. def fanoff():
269. try:
270. FanOff()
271. return jsonify({'Fan': "0"})
272. except:
273. return jsonify({'Fan': "-1"})
274. @app.route('/api/humidity', methods=['GET'])
275. @cross\_origin()
276. @auth.login\_required
277. def humidity():
278. try:
279. humidity, temperature = Adafruit\_DHT.read\_retry(11, 26)
280. return jsonify({'humidity':humidity})
281. except:
282. return jsonify({'humidity':'-1'})
283. @app.route('/api/temperature', methods=['GET'])
284. @cross\_origin()
285. @auth.login\_required
286. def temperature():
287. try:
288. humidity, temperature = Adafruit\_DHT.read\_retry(11, 26)
289. return jsonify({'temperature':temperature})
290. except:
291. return jsonify({'temperature':'-1'})
292. @app.route('/api/frontsensor', methods=['GET'])
293. @cross\_origin()
294. @auth.login\_required
295. def frontsensor():
296. try:
297. return jsonify({'reading': ReadSensorReading()})
298. except:
299. return jsonify({'reading': "-1"})
300. @app.route('/api/detailsensorreadings', methods=['GET'])
301. @cross\_origin()
302. @auth.login\_required
303. def detailsensorreadings():
304. try:
305. return jsonify({'detailsensorreadings': GetSensorReadingsWithServo()})
306. except:
307. return jsonify({'detailsensorreadings': "-1"})
308. @app.route('/api/move/<int:value>' , methods=['GET','POST'])
309. @cross\_origin()
310. @auth.login\_required
311. def setangle(value):
312. try:
313. print(value)
314. SetAngle(value)
315. GPIO.cleanup()
316. return jsonify({'Angle': value})
317. except:
318. return jsonify({'Angle': "-1"})
319. if \_\_name\_\_ == '\_\_main\_\_':
320. try:
321. app.run(debug=True, host='0.0.0.0', port=9000)
322. finally:
323. print("Cleaning Up!")
324. GPIO.cleanup()

# System Design

## The 4wd Robot Chasis

This 4WD Chassis is simple yet versatile robot chassis designed specifically for students and hobbyist. Featuring large size chassis plates cut from acrylic and designed with numerous holes and mounting points, providing plenty of space to carry a PCB board and any additional components that you choose. The possibilities are only limited by your imagination.

4 Wheel Robot Chassis Kit, an easy to assemble and use robot chassis platform. The Chassis kit provides you with everything you need to give your robot a fast four wheel drive platform with plenty of room for expansion to add various sensors and controllers.

Just add your electronics - Arduino/Raspberry Pi and Motor Driver and you can start programming your robot. It offers a large space with predrilled holes for mounting sensors and electronics as per your requirement. This robot chassis lets you get your mechanical platform ready in minutes and quickstart your robot building process.Wheeled Robots are most popular robot platforms and are easy to run, maintain and use. Simple to build and program, this kit is the simplest robot platform. Highly recommended for beginners and novice users. Building robots using this wheeled kit is fun and a great learning experience too. The 4WD Kit lets you go faster, carry more weight, and carry bigger load compared to the 2WD Kit. You can build line following robots, obstacle avoiding robots, and other robots using this kit.



Figure 3.1 Robot chasis

## Assemble the robot chasis

Here’s what we have, after taking everything out of the packaging:

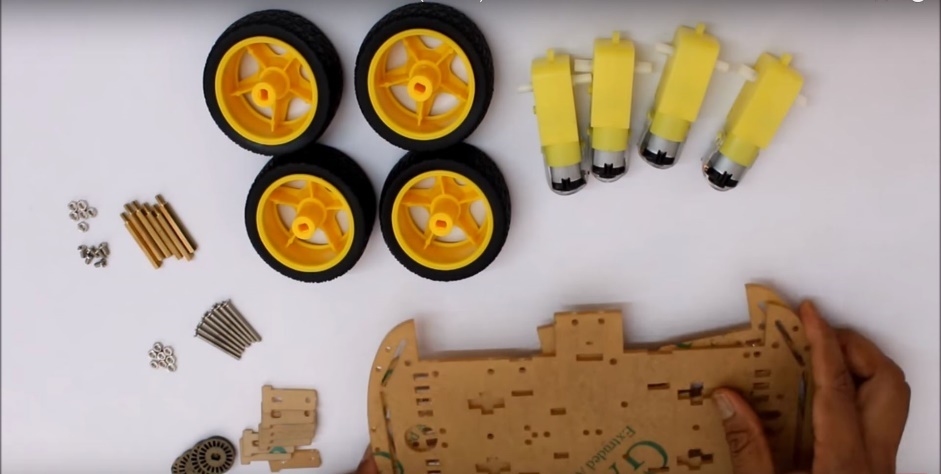
****

Figure 3.2 Robot Chasis

Firstly insert the coded disc

****

Figure 3.3 Coded Disc

Now insert the support for dc motors

****

Figure 3.4 Support insert

Now install the dc motor

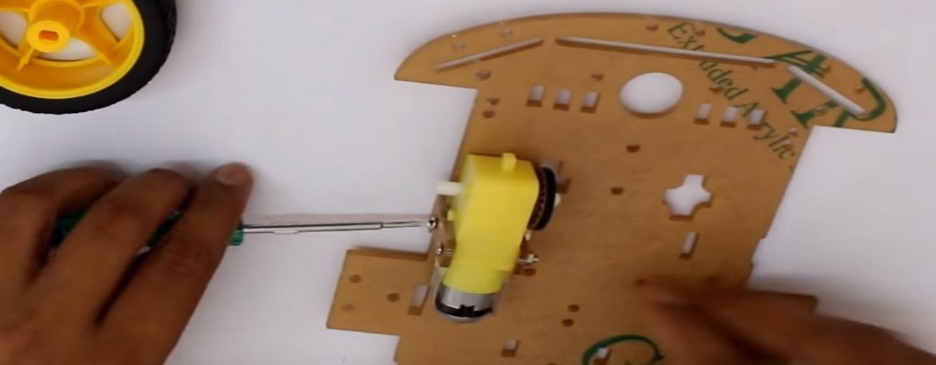
****

Figure 3.5 Motor install

After installing the remaining 3 dc motors

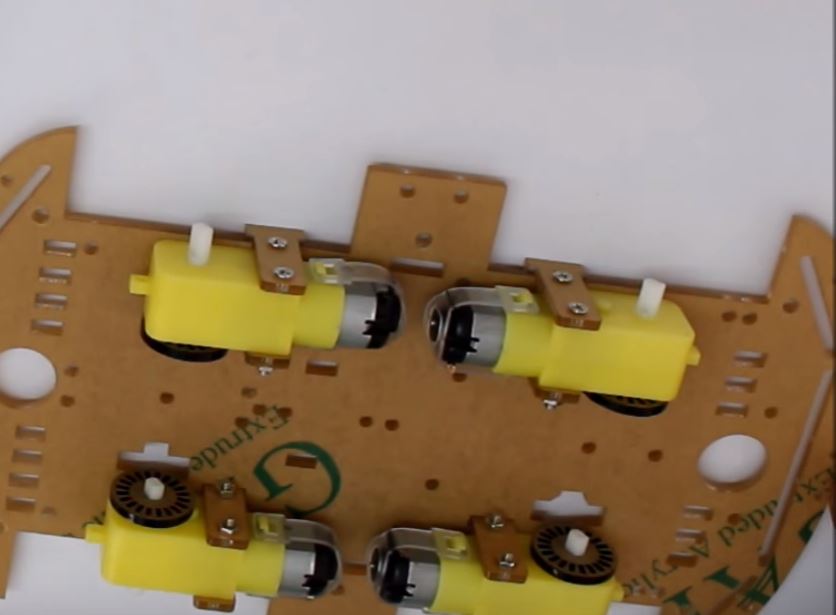
****

Figure 3.6 Motor install

Now install the copper column

****

Figure 3.7 Copper column

After installing the remaining 5 column

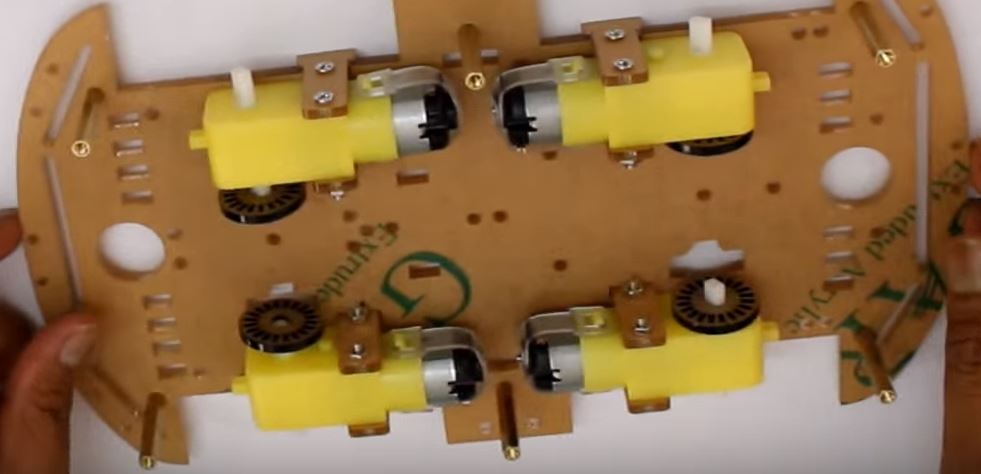
****

Figure 3.8 Copper Column

Now install the top roof

****

Figure 3.9 Top Roof

Finally tight the screws with screw driver

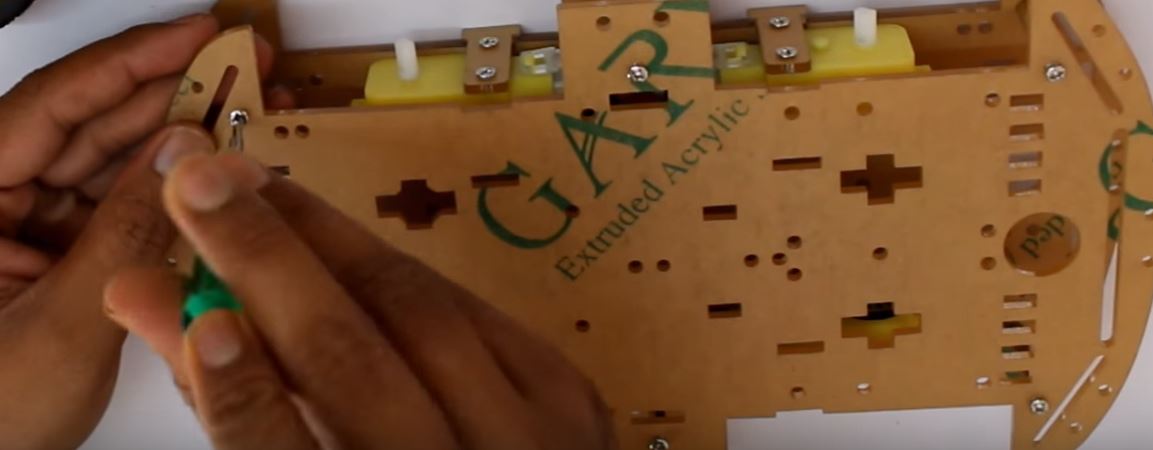
****

Figure 3.10 screw tight

Now install the wheel

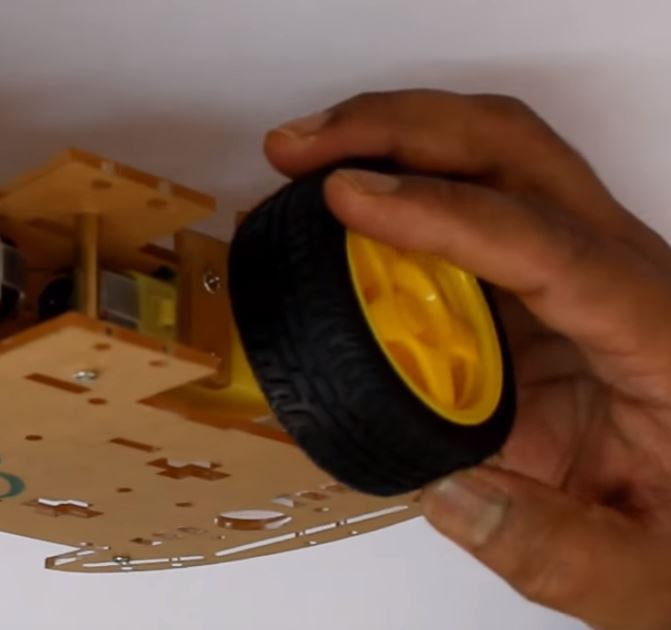
****

Figure 3.11 install wheel

After install the remaining wheel its look like this

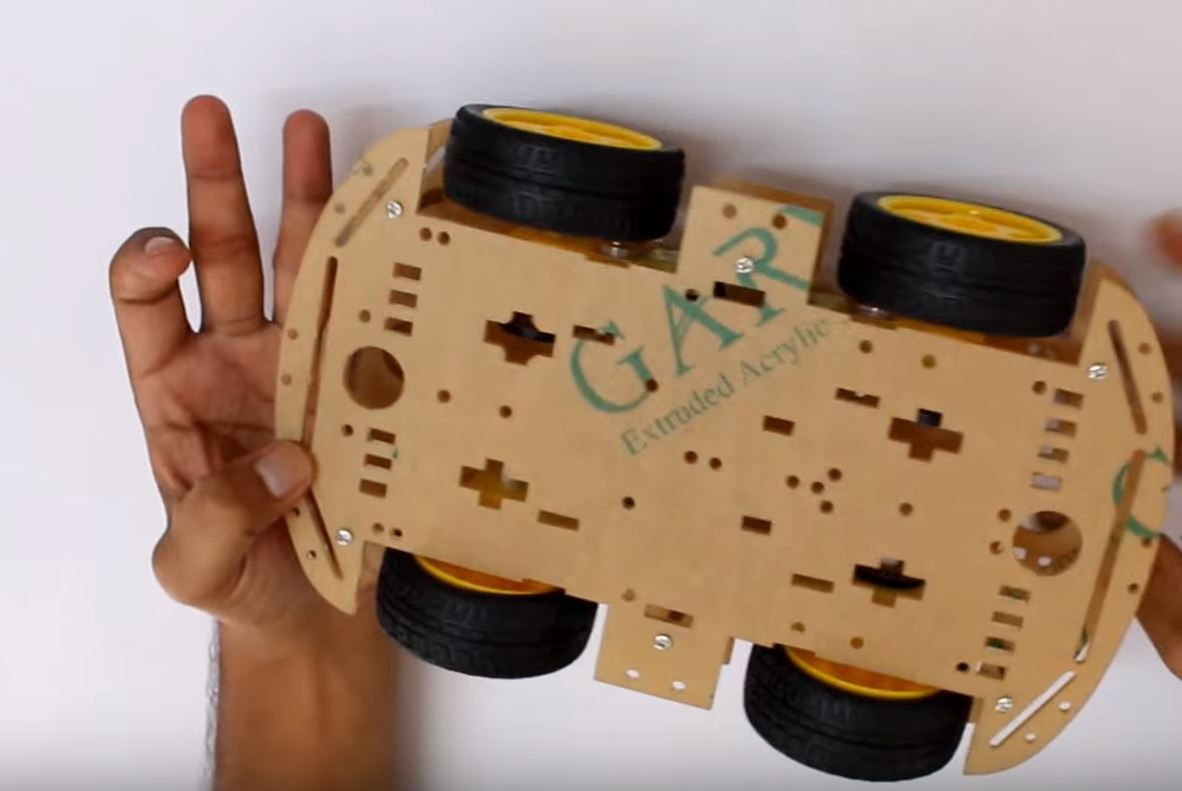
****

Figure 3.12 robot car

## Design the website

Designing good looking clean and functional Web layouts is an essential part of a Web Designers life. In this tutorial we are going to create a clean and professional Web layout in Photoshop from scratch.First I create icon in photoshop for my robot control buttons

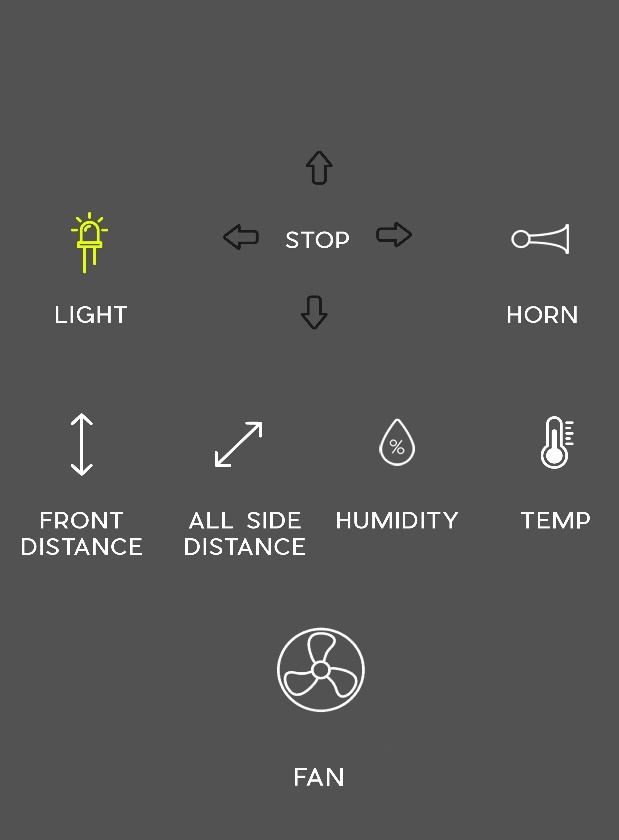


Figure 3.13

I used these icon for create my button .After some modification of these icons I make a button and save each button icon as PNG so that is used in my website with the help of coding. These are icon listed below image.

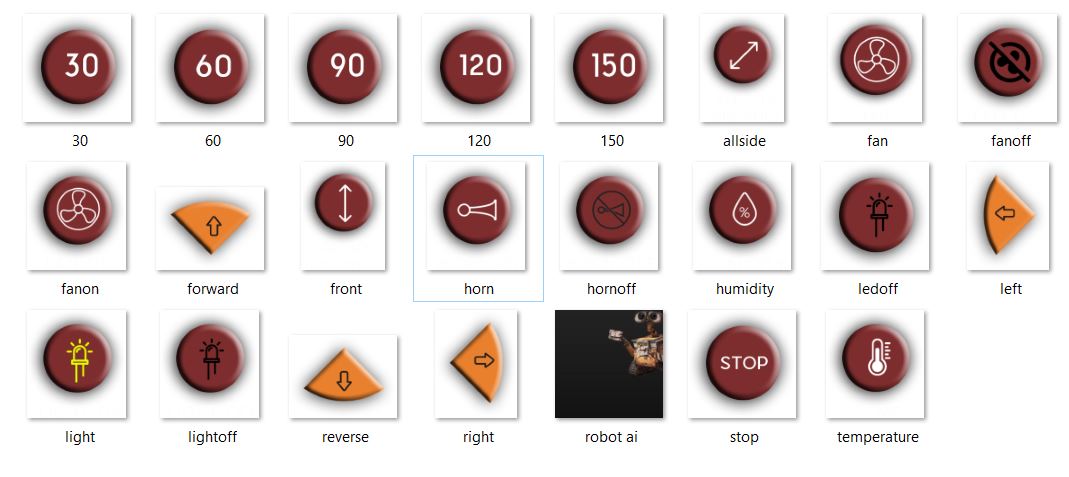


Figure 3.14

Finally after place are icons and design the actual website using photoshop its look like this.

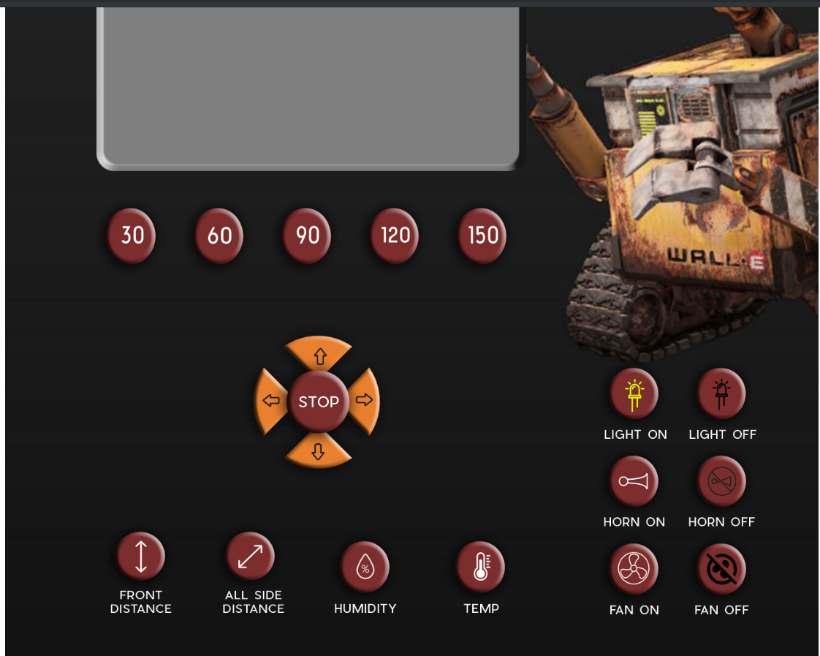


Figure 3.15

## Design the communication strategy

Communication server is one of the most important parts of Project. It is designed to link the website and the robot with the internet. Basically my project has three phase design. First is website that is coded in HTML , CSS , javascript and Bootsrap etc, and second phase is used to Get and Post data from website to webservice and vice versa , I used Ajax and jquery for that purpose . I used NGINX server to host my websites and uWSGI to invokes python objects. Third phase is webservice I use python language framework called flask for make REST API . I develop REST API for control all raspberry pi GPIO pin and every thing is control in REST API so that I had fast response

**What is REST?**

In recent years [REST](https://en.wikipedia.org/wiki/Representational_state_transfer) (REpresentational State Transfer) has emerged as the standard architectural design for web services and web APIs.

The characteri stics of a REST system are defined by six design rules:

**Client-Server**: There should be a separation between the server that offers a service, and the client that consumes it.

**Cacheable**: The server must indicate to the client if requests can be cached or not.

**Layered System**: Communication between a client and a server should be standardized in such a way that allows intermediaries to respond to requests instead of the end server, without the client having to do anything different.

**Uniform Interface**: The method of communication between a client and a server must be uniform.

**Code on demand**: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

**What is a RESTful web service?**

The REST architecture was originally designed to fit the [HTTP protocol](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) that the world wide web uses.

Central to the concept of RESTful web services is the notion of resources. Resources are represented by [URIs](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier). The clients send requests to these URIs using the methods defined by the HTTP protocol, and possibly as a result of that the state of the affected resource changes.

The HTTP request methods are typically designed to affect a given resource in standard ways:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Action** | **Examples** |
| GET | Obtain information about a resource | http://example.com/api/orders (retrieve order list) |
| GET | Obtain information about a resource | http://example.com/api/orders/123 (retrieve order #123) |
| POST | Create a new resource | http://example.com/api/orders (create a new order, from data provided with the request) |
| PUT | Update a resource | http://example.com/api/orders/123 (update order #123, from data provided with the request) |
| DELETE | Delete a resource | http://example.com/api/orders/123 (delete order #123) |

Table 3.1 HTTP Method

The REST design does not require a specific format for the data provided with the requests. In general data is provided in the request body as a [JSON](http://en.wikipedia.org/wiki/JSON) blob, or sometimes as arguments in the [query string](http://en.wikipedia.org/wiki/Query_string) portion of the URL.

Communication method is shown on below image , web clients send the request to the server by using ajax and jquery then invokes the REST API webservice to full fill the request send by the server

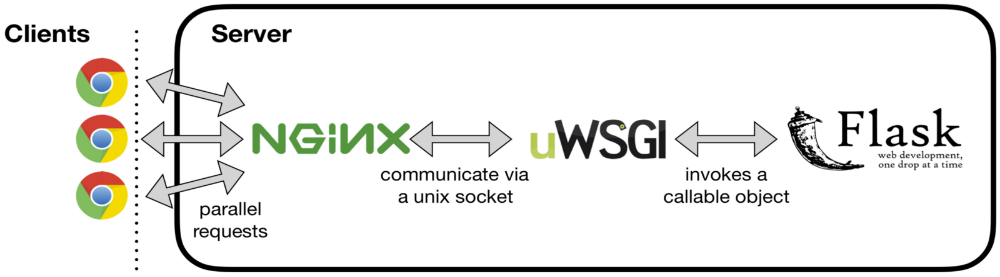


Figure 3.16 communication method

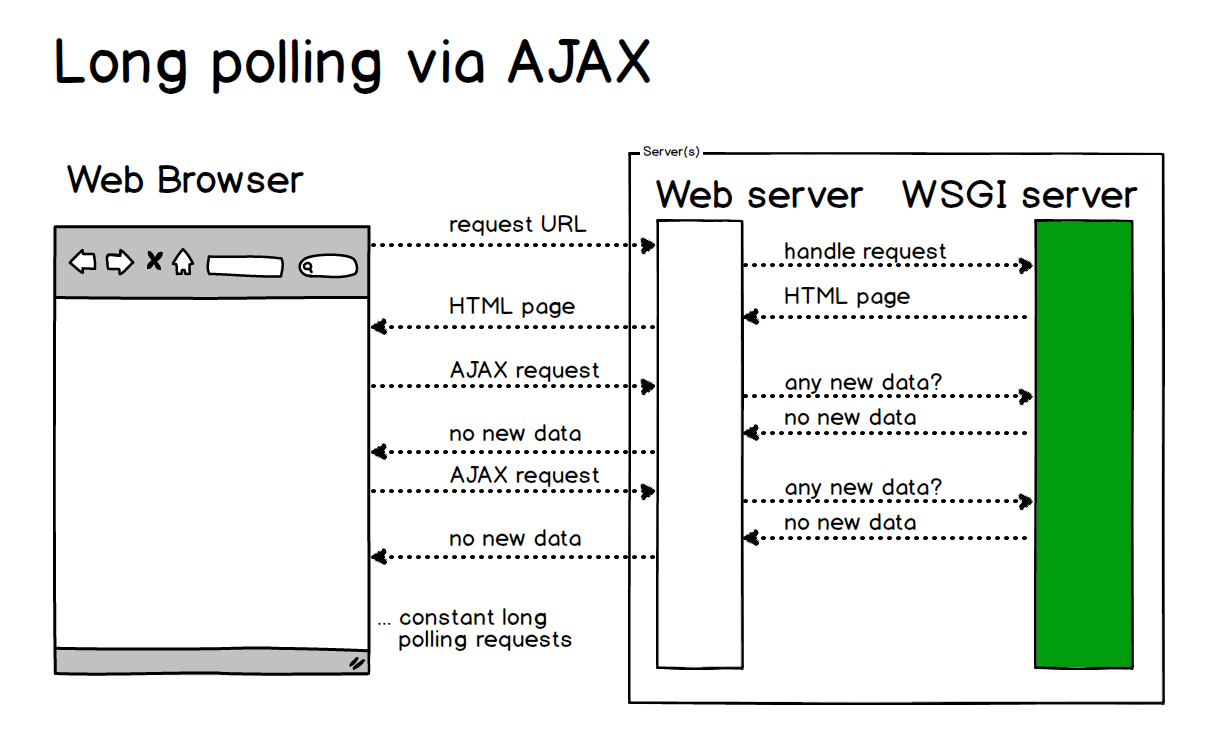


Figure 3.17 Long polling

**3.3 | Robustness**

The communication server is designed in such a way that it has the ability to smoothly direct the data from time to time depending on situation .In some cases the speed of communication is going slow due to some technical reason.

Robots are essentially a self-contained tribute to the wonders of technology. The most advanced models use fast computer processing, high-definition cameras, artificial intelligence and long-range sensors, all of which give you a pretty good idea where technology is heading. In some ways, a robot even provides a glimmer of the future car and future IT advances.

The components for a robot are all housed on one intelligent machine that connects back to a server over a high-speed network. When deployed, a robot must be engineered to act autonomously. Any flaws in the programming lead to serious repercussions. The autonomous security robot is equipped with a panoramic video surveillance system consisting of six cameras for all-round observation and a PTZ camera to track motion at a large distance. The images from all cameras are transmitted over WiFi to a Central Monitoring Station or a security officer’s laptop. The video surveillance system has a built-in DVR with a motion-detection feature. In automatic mode, the video surveillance system uses a 360-degree camera to look for sources of movement. In manual mode, the robot’s operators can control the camera at their own discretion.

The security patrol robot performs remote video surveillance using WiFi wireless data transfer technology. When using the built-in omnidirectional antennae, the data transfer distance between the robots and the base station may reach up to more than half a mile. To transmit video at a distance of several miles, it is necessary to use a directional antenna on the receiving side in the Central Monitor Station. In the presence of obstacles between the robot and the monitor station, such as buildings or heavy vehicles, using an additional robot to relay the signal is feasible.

After a huge modification in my project code I will be able to overcome these problem. I remove all less important part of code and try to sumerize the code as much as possible . In REST API I optimize my code very efficiently so that my communication was very fast . In my project speed of communication is more concern because it is a moving robot with internet .

# Implementation

## Circuit Diagram

A circuit diagram is a graphical representation of an electrical circuit. A pictorial circuit diagram uses simple images of components, while a schematic diagram shows the components and interconnections of the circuit using standardized symbolic representations.

This is whole circuit diagram of my project which things i used, where I connected it, every thing is explain in below circuit diagram

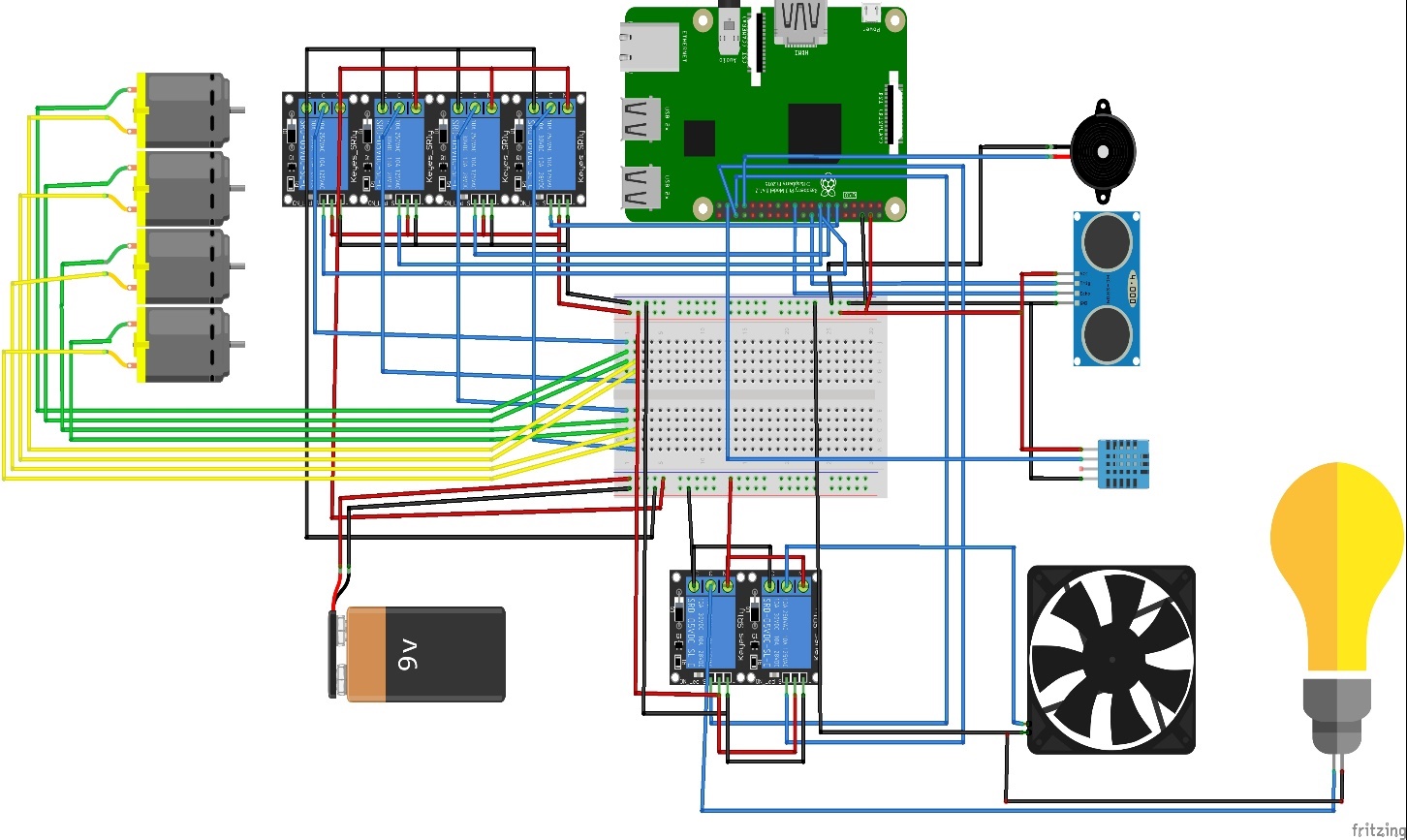


Figure 4.1 Ciruit Diagram

## Install Raspbian on Raspberry pi

These are the steps to install the Raspbian

**Step 1. Download a lite Raspbian image**

These instructions are for a **Raspbian Stretch** image that I downloaded from here:

<https://www.raspberrypi.org/downloads/raspbian/>

**Step 2. Burn the Raspbian image to the SD card**



Figure 4.2 etcher

To burn an image to the SD card you can use Etcher.

Browse to [https://etcher.io](https://etcher.io/)

Download the version for your operating system

Run the installer

To run Etcher is pretty straight forward.

Put a blank mini SD card and adapter into your machine. No need to format it. You can use a new SD card right out of the package.

**Select image** - browse to the **zip** file you downloaded from Raspbian

**Select drive** - it may find the SDHC Card automatically

Click **Flash!**

After you flash (burn) the image, Finder (Mac) or File Explorer (Windows) may have trouble seeing it. A simple fix is to pull the SD card out then plug it back in. On a Mac it should appear on the desktop with the name **boot**. On Windows it should appear in File Explorer with the name **boot**followed by a drive letter.

**Step 3. Enable ssh**

For security reasons, **ssh** is no longer enabled by default. To enable it you need to place an empty file named ssh (no extension) in the root of the boot disk.

Run **Notepad**

In a new file put in one space and nothing more

Click **File / Save As ...**

Be sure to set **Save as type** to **All Files** (so the file is NOT saved with a .txt extension)

Call the file **ssh** and save it

Close the file

If you are comfortable with the Windows command line you could try this instead

Open up a command line

Switch to the drive and root where **boot** is located:

Type: type NUL >> ssh

Verify that file **ssh** was created

**Step 4. Add network info**

Create a file in the root of **boot** called: wpa\_supplicant.conf (instructions below). Then paste the following into it (adjusting for the name of your country code, network name and network password):

country=US

ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev

update\_config=1

network={

ssid="NETWORK-NAME"

psk="NETWORK-PASSWORD"

}

**Windows instructions (wifi settings)**

Run **Notepad**

Paste in the contents above (adjusting for the name of your country code, network name and network password)

Click **File / Save As ...**

Be sure to set **Save as type** to **All Files** (so the file is NOT saved with a .txt extension)

Call the file wpa\_supplicant.conf and save it

Close the file

Step 5. Eject the micro SD card

Right-click on **boot** (on your desktop or File Explorer) and select the **Eject** option

This is a "logical" eject - meaning it closes files and preps the SD card for removal - you still have to pull the card out yourself

Step 6. Boot the Raspberry Pi

Remove the mini-SD card from the adapter and plug it into the Raspberry Pi

Plug a Micro-USB power cable into the **power** port

Give the Pi plenty of time to boot up (it can take as much as 90 seconds -- or more)

Step 7. Login over Wifi

This part assumes that **ssh** is enabled for your image and that the default user is **pi** with a password of **raspberry**.

**NOTE**: Your machine must be on the same WiFi network that you configured the Pi for.

**Install Putty**

If you already have Putty installed, skip to the next section.

Browse to: [https://www.putty.org](https://www.putty.org/)

Download the 64-bit MSI (Windows Installer)

Open it to run the installer (if asked for permission, click Yes)

Select: **Add shortcut to PuTTY on the Desktop**

**Login over WiFi using Putty**

This part assumes that **ssh** is enabled for your image and that the default user is **pi** with a password of **raspberry**.

Launch Putty

Set the **Host Name (or IP address)** field to **raspberrypi.local**

By default the **Port** should be set to **22** and **Connection type** should be set to **SSH**

Click **Open**

If you see a Security Alert select Yes

A new terminal window should appear prompting you for a user name

The default user name is: **pi**

The default password is: **raspberry**

Step 8 . Get the updates

Once connected over WiFi, the next thing you should do is run some updates:

sudo apt-get update

sudo apt-get upgrade

Install Nginx on Raspberry Pi

Begin by installing the Nginx package. In a terminal, run:

sudo apt-get install nginx

Next, start the server using:

sudo /etc/init.d/nginx start

NGINX places an HTML file under the web folder. As such, you can test if NGINX installed properly by navigating to http://localhost/ or http://[YOUR RASPBERRY PI'S IP ADDRESS]/. To view your IP address, you may run the command hostname -I.

If installing NGINX on the Raspberry Pi went as planned, you'll see a friendly "Welcome to nginx" message when you navigate to http://localhost/ in a browser.

By default, NGINX stores its web page location at /var/www/html. Here, you may edit the index.nginx-debian.html to change the default web page. You can check the default page at /etc/nginx/sites-available.

Install Motion Server

Type in the command 'sudo apt-get install motion ' to start the installation.

Now to make sure that the camera is correctly detected.

Type in the command 'lsusb' and enter. You should see the name of your camera. If it is NOT there, then there is some problem in your camera or the camera is not supported in 'motion'.

After the installation is complete, type in the command ' sudo nano /etc/motion/motion.conf' and press enter.

Then you have to change some settings in the .conf file. It might be difficult sometimes to find the settings but use 'ctrl + w' to find it. So follow the steps:

Make sure 'daemon' is ON.

Set 'framerate' anywhere in between 1000 to 1500.

Keep 'Stream\_port' to 8081.

'Stream\_quality' should be 100.

Change 'Stream\_localhost' to OFF.

Change 'webcontrol\_localhost' to OFF.

Set 'quality' to 100.

Set 'width' & 'height' to 640 & 480.

Set 'post\_capture' to 5.

Press ctrl + x to exit. Type y to save and enter to conform.

Again type in the command 'sudo nano /etc/default/motion ' and press enter.

Set ' start\_motion\_daemon ' to yes. Save and exit.

## Port Forwarding

A port forward is a way of making a computer on your home or business network accessible to computers on the internet, even though they are behind a router. It is commonly used in gaming, security camera setup, voice over ip, and downloading files. After you have forwarded a port you are said to have an open port.

setting up port forwarding is quite similar to how its done in any other router. Port forwarding allows your router to correctly redirect external inbound service requests to the correct internal computer on your network. Setting up a router to port forward can be accomplished in a few steps.

Port forwarding is a very easy task and can be done using the following steps:

Open the administration panel of your modem. Usually the IP of your modem is http://192.168.1.1  or you can try [http://192.168.0.1](http://192.168.0.1/) if the other doesn’t work for you.

Enter username and password once prompted. The default login details are username: admin and password: admin.Once logged in you will see the following page

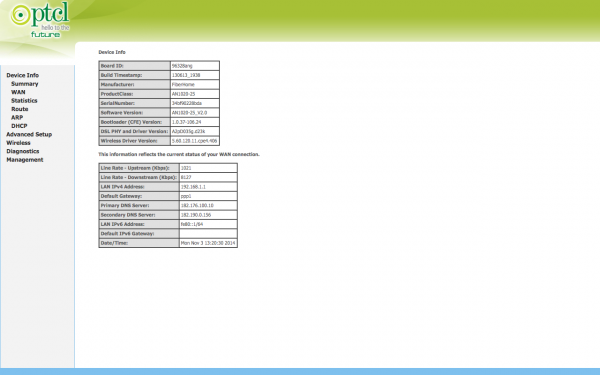


Figure 4.3 Ptcl login

Use the navigation on the left and click **Advanced Setup**.

Next click **NAT**, and by default Virtual Servers should be selected. If it is not click on Virtual Servers.

Click Add button to proceed with port forwarding.

Once you click Add, a new page asking for details will open. For your reference I have attached a screenshot of the page

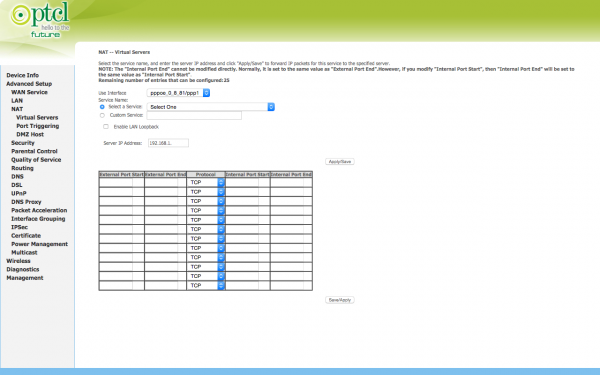


Figure 4.4 Ptcl NAT

You can select a service from the drop down or select Custom Service. In case you want Custom Service, just click on the radio button next to Custom Service. Enter any name you

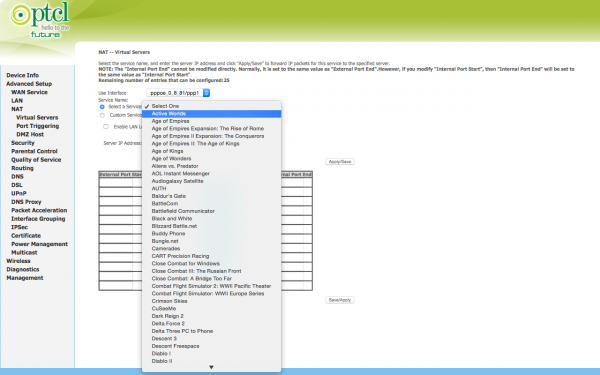


Figure 4.5 PTCl Custom

want to give to this service in the input field next to the Custom Service

Now write the IP address of the device on which the service is running. For example if you are running wamp on your desktop, enter the IP address of your desktop. This will be internal ip which should look something like

192.168.1.120.

In the External Port Start and External Port End enter the desired port.

Next select appropriate protocol. The available protocols are TCP or UDP or TCP/UDP.

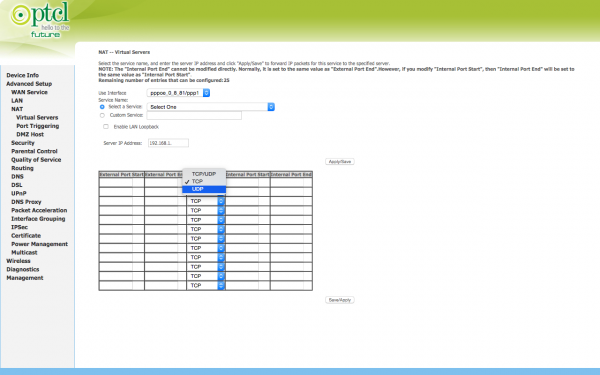


Figure 4.6 PTCL

Once you are done entering details press Apply/Save.

## Working

Open the putty software , give the ip address with port 22

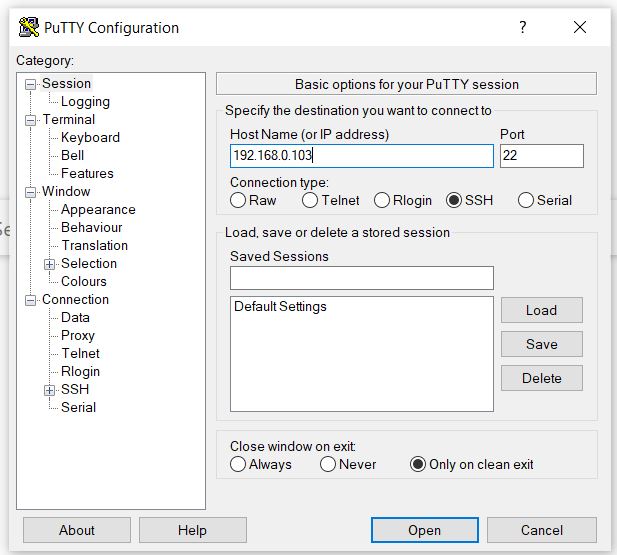


Figure 4.7 Putty

Now click the open button

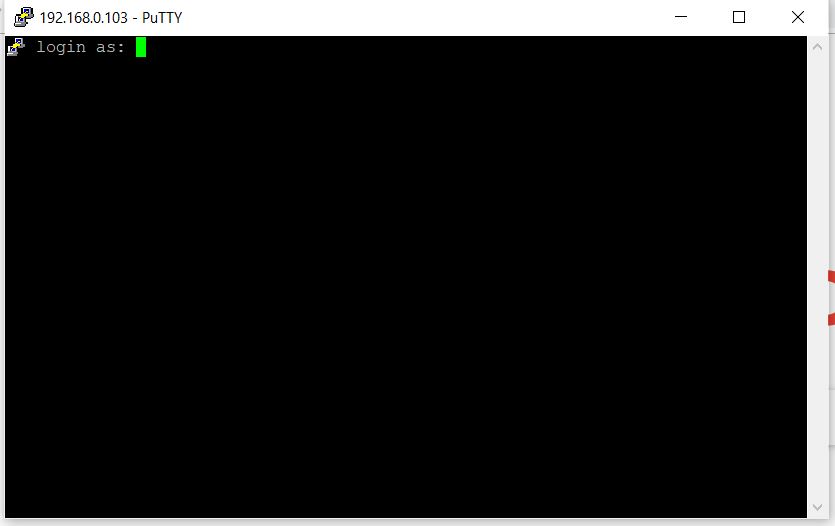


Figure 4.8 CMD

It ask the user name give the user name and hit enter

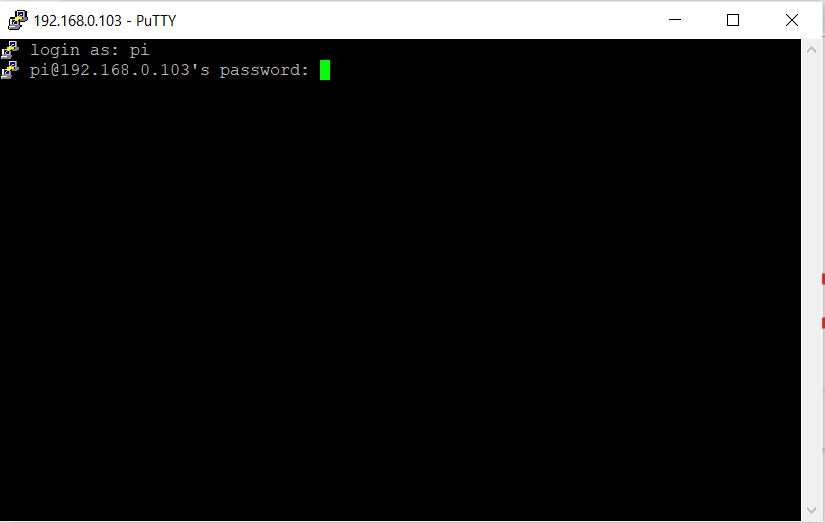


Figure 4.9 CMD login

Now enter the password and hit enter

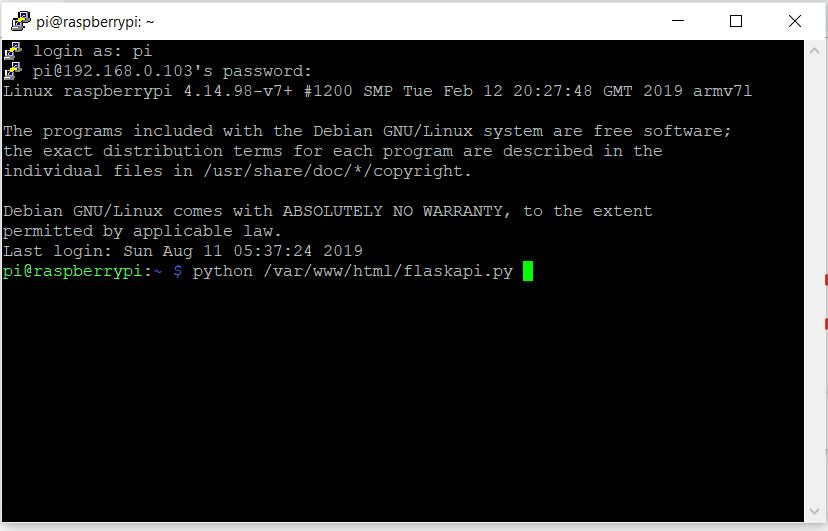


Figure 4.10 CMD

Give the location of the rest api and run that program after run open the url 192.168.0.103/login.html that is look like this

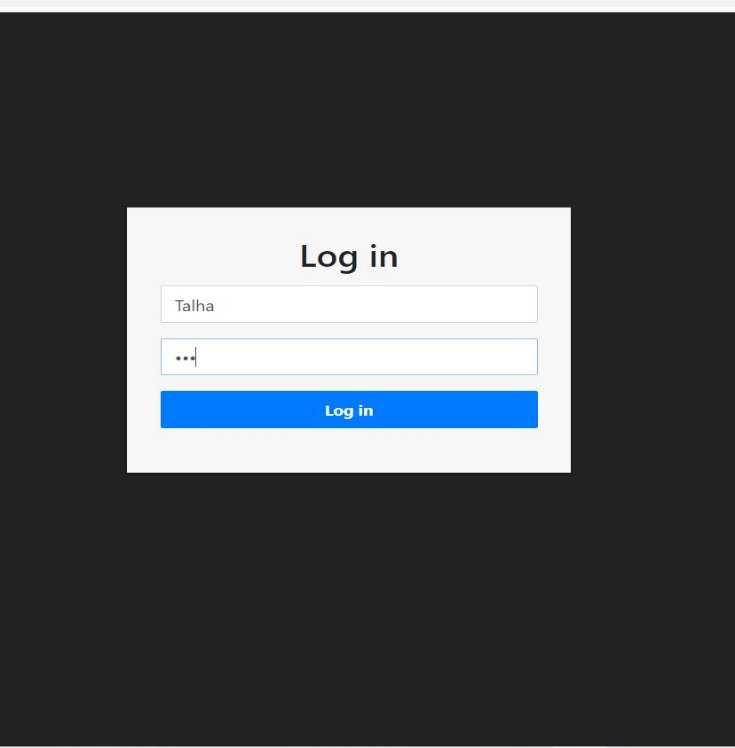


Figure 4.11 Login page

Give username and password and hit login button

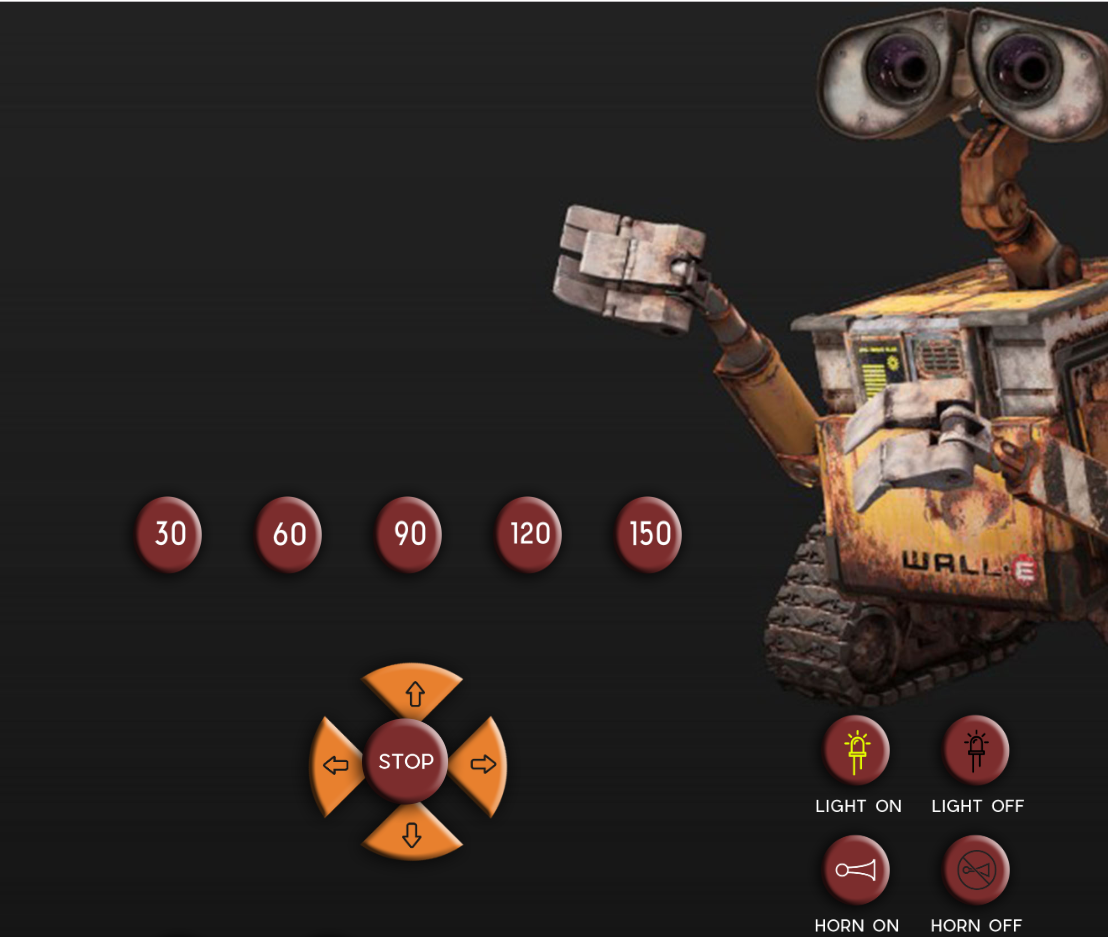


Figure 4.12 Dashboard

## Screenshot of working

These are screenshot of the working of project



Figure 4.1 Light on

Figure 4.2 Light on



Figure 4.3 camera move

Figure 4.4 move 150

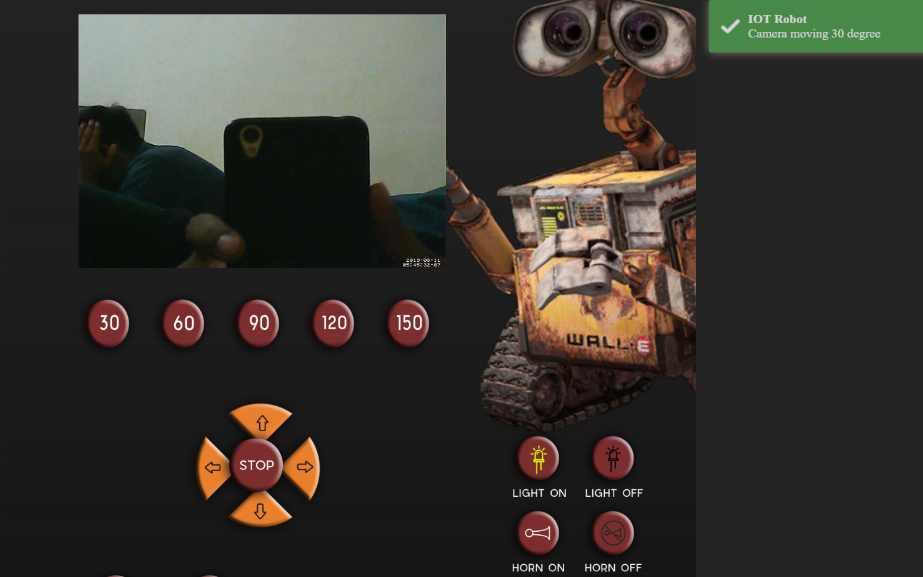
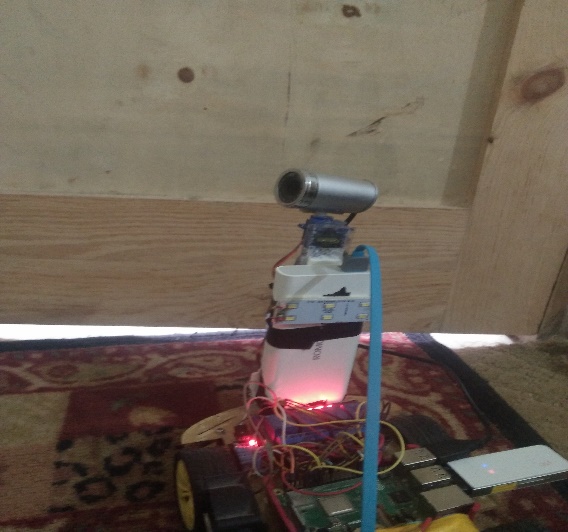


Figure 4.5 Camera move

Figure 4.6 Camera move

Figure 4.7 move 30

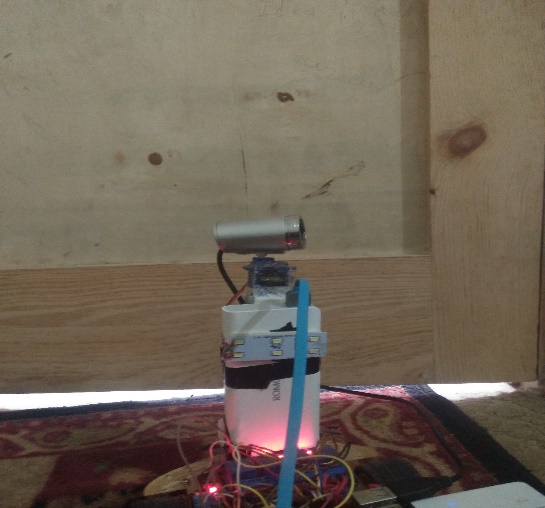
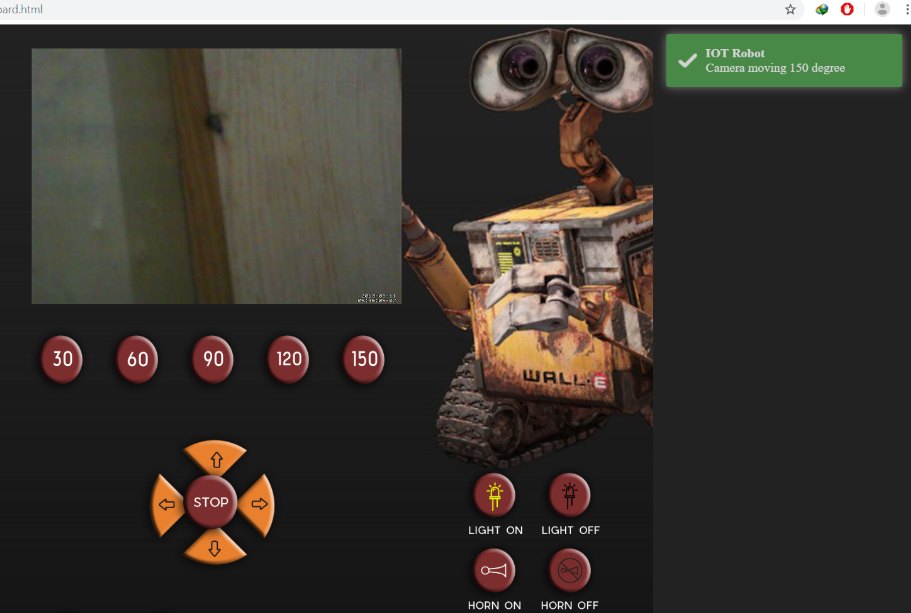


Figure 4.8 move 150

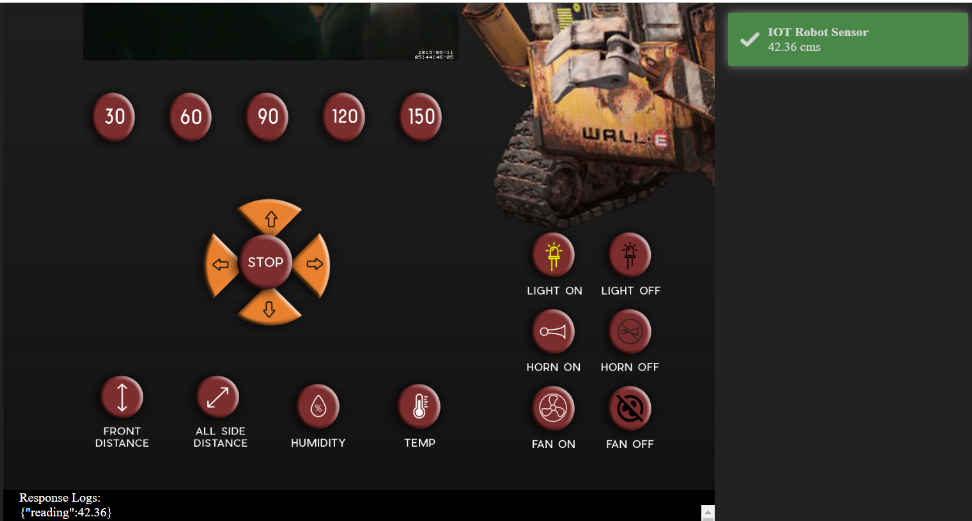


Figure 4.13 Front distance

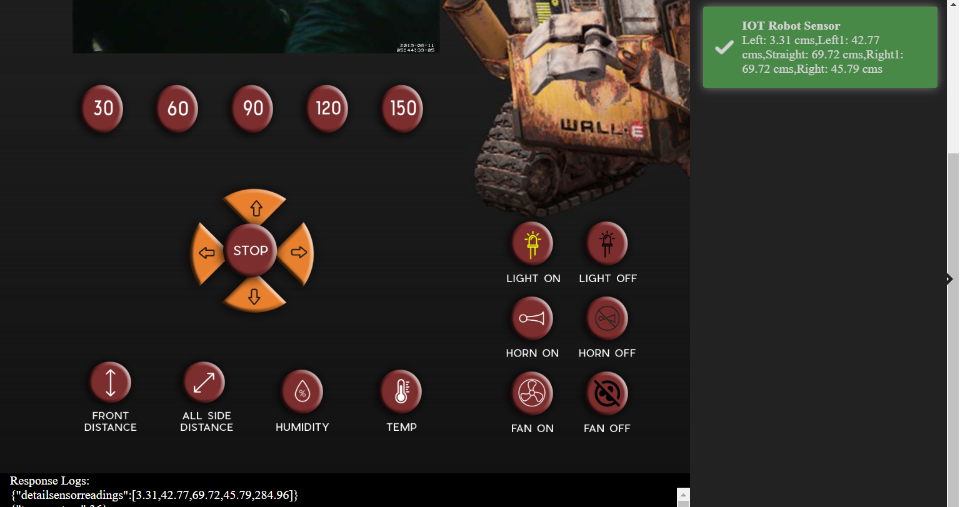


Figure 4.14 All side distance

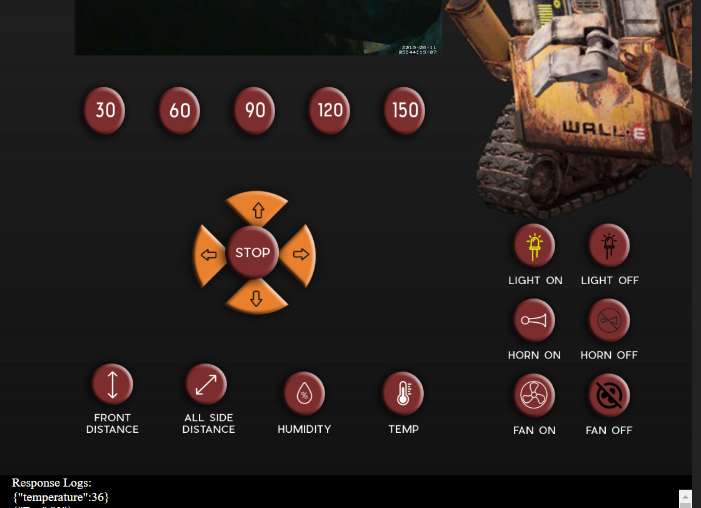
­­­­­­­

Figure 4.15 Temperature

## Code of Project

### HTML Part

**Login.html**

1. <!DOCTYPE html>
2. <!-- saved from url=(0049)http://getbootstrap.com/docs/4.0/examples/signin/ -->
3. <html lang="en"><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4. <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
5. <meta name="description" content="">
6. <meta name="author" content="">
7. <link rel="icon" href="http://getbootstrap.com/favicon.ico">
8. <title>Login IOT Robot</title>
9. <!-- Bootstrap core CSS -->
10. <link href="css/bootstrap.min.css" rel="stylesheet">
11. <!-- Custom styles for this template -->
12. <link href="css/signin.css" rel="stylesheet">
13. <link href="css/toastr.min.css" rel="stylesheet">
14. </head>
15. <body>
16. <div class="login-form">
17. <form>
18. <h2 class="text-center">Log in</h2>
19. <div class="form-group">
20. <input type="text" id="inputUsername" class="form-control" placeholder="Username" required="required">
21. </div>
22. <div class="form-group">
23. <label for="inputPassword" class="sr-only">Password</label>
24. <input type="password" id="inputPassword" class="form-control" placeholder="Password" required="required">
25. </div>
26. <div class="form-group">
27. <button class="btn btn-primary btn-block" onclick="LoginClicked(); return false;">Log in</button>
28. </div>
29. </form>
30. <!-- Bootstrap core JavaScript
31. ========================================== -->
32. <!-- Placed at the end of the document so the pages load faster -->
33. <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
34. <script src="js/jquery-3.2.1.min.js"></script>
35. <script src="js/popper.min.js.download"></script>
36. <script src="js/bootstrap.min.js.download"></script>
37. <script src="js/toastr.min.js"></script>
38. <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
39. <script src="js/ie10-viewport-bug-workaround.js.download"></script>
40. <script src="js/global.js"></script>
41. <script src="js/login.js"></script>
42. </body></html>

**Dashboard.html**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Robot</title>
6. <link href="css/bootstrap.min.css" rel="stylesheet">
7. <!-- Custom styles for this template -->
8. <link href="css/toastr.min.css" rel="stylesheet">
9. <link href="css/dashboard.css" rel="stylesheet">
10. </head>
11. <body>
12. <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
13. <a class="navbar-brand" href="#">Raspberry Pi IOT Robot</a>
14. <button class="navbar-toggler d-lg-none" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">
15. <span class="navbar-toggler-icon"></span>
16. </button>
17. <div class="collapse navbar-collapse" id="navbarsExampleDefault">
18. <ul class="navbar-nav mr-auto">
19. <!-- <li class="nav-item active">
20. <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
21. </li> -->
22. </ul>
23. <div>
24. <a href="login.html">Logout</a>
25. </div>
26. </div>
27. </nav>
28. <div class="outer">
29. <div class="live">
30. <img id="streamimg" class="img-responsive img-rounded" src= "http://192.168.10.15:8081?action=stream" />
31. </div>
32. <div class="row1">
33. <button onclick="Angle30()" style="margin-left: 90px;"><img class="b" src="images/30.png" ></button>
34. <button onclick="Angle60()"><img class="b" src="images/60.png" ></button>
35. <button onclick="Angle90()"><img class="b" src="images/90.png" ></button>
36. <button onclick="Angle120()"><img class="b" src="images/120.png" ></button>
37. <button onclick="Angle150()"><img class="b" src="images/150.png" ></button>
38. </div>
39. <div class="center">
40. <div class="forward"> <button onclick="Forward()"><img src="images/forward.png" alt=""></button> </div>
41. <div class="reverse"> <button onclick="Reverse()"><img src="images/reverse.png" alt=""></button> </div>
42. <div class="right"> <button onclick="Right()"><img src="images/right.png" alt=""></button> </div>
43. <div class="left"> <button onclick="Left()"><img src="images/left.png" alt=""></button> </div>
44. <div class="stop"> <button onclick="Brake()"><img src="images/stop.png" alt=""></button></div>
45. </div>
46. <div class="col">
47. <button onclick="LedOn()"><img class="b" src="images/light.png"></button>
48. <button onclick="LedOff()"><img class="b" src="images/lightoff.png"></button>
49. <br>
50. <button onclick="HornOn()"><img class="b" src="images/horn.png"></button>
51. <button onclick="HornOff()"><img class="b" src="images/hornoff.png"></button>
52. <br>
53. <button onclick="FanOn()"><img class="b" src="images/fanon.png"></button>
54. <button onclick="FanOff()"><img class="b" src="images/fanoff.png"></button>
55. </div>
56. <div class="row">
57. <button onclick="FrontSensor()" style="margin-left: 100px; "><img class="c" src="images/front.png" ></button>
58. <button onclick="AllSideSensor()" style="padding-left: 30px; "><img class="c" src="images/allside.png" style="margin-right: 30px"></button>
59. <button onclick="Humidity()" ><img class="b" src="images/humidity.png" style="margin-right: 30px"></button>
60. <button onclick="Temperature()"><img class="b" src="images/temperature.png" style="margin-right: 30px"></button>
61. </div>
62. <div class="row2">
63. <main role="main" style="background:black; margin-top:50px; color:white;padding-left:20px;">
64. <div>Response Logs:
65. </div>
66. <div id="response" style="height:200px;overflow-y:auto;">
67. </div>
68. </main>
69. </div>
70. </div>
71. <script src="js/jquery-3.2.1.min.js"></script>
72. <script src="js/popper.min.js.download"></script>
73. <script src="js/bootstrap.min.js.download"></script>
74. <script src="js/toastr.min.js"></script>
75. <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
76. <script src="js/ie10-viewport-bug-workaround.js.download"></script>
77. <script src="js/global.js"></script>
78. <script src="js/custom.js"></script>
79. </body>
80. </html>

### CSS Part

**Login.css**

1. .login-form {
2. width: 400px;
3. margin: 200px auto;
4. }
5. .login-form form {
6. margin-bottom: 15px;
7. background: #f7f7f7;
8. box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
9. padding: 30px;
10. }
11. .login-form h2 {
12. margin: 0 0 15px;
13. }
14. .form-control, .btn {
15. min-height: 38px;
16. border-radius: 2px;
17. }
18. .btn {
19. font-size: 15px;
20. font-weight: bold;
21. }
22. body{
23. background-color: #222222;;
24. }

**Dashboard.css**

1. .outer{
2. width: 890px;
3. height: 900px;
4. margin-left: 300px;
5. margin-right: 200px;
6. background-image: url("../images/robot ai.jpg");
7. background-size: cover;
8. }
9. body
10. {
11. padding: 0;
12. margin: 0;
13. }
14. .live img {
15. padding-left: 100px;
16. padding-top: 30px;
17. width: 470px;
18. height: 325px;
19. }
20. .forward {
21. position: absolute;
22. top: 68%;
23. }
24. .stop {
25. position: absolute;
26. }
27. body,button {
28. border: none;
29. }
30. button {
31. background-color: transparent;
32. cursor: pointer;
33. overflow: hidden;
34. }
35. .reverse {
36. position: absolute;
37. top: 82%;
38. }
39. .forward button , .reverse button , .reverse button img , .forward button img {
40. width: 110px;
41. }
42. .left button, .right button , .left button img , .right button img{
43. height: 110px;
44. }
45. .left {
46. position: absolute;
47. right: 59%;
48. }
49. .right {
50. position: absolute;
51. right: 52%;
52. }
53. .center{
54. margin-left: 270px;
55. margin-top: 80px;
56. }
57. .stop button, .stop button img {
58. height: 100px;
59. }
60. body {
61. background-color: #222222;}
62. .b {
63. height: 90px;
64. width: 80px;
65. }
66. .c{
67. height: 100px;
68. width: 80px;
69. }
70. .row {
71. padding-top: 180px;
72. }
73. .col{
74. position: absolute;
75. margin-left: 640px;
76. margin-top: 0px;
77. }
78. .row1{
79. padding-top: 22px;
80. }
81. button:focus {outline:0;}

## JS Part

**Login.js**

1. function LoginClicked(){
2. var username = $("#inputUsername").val();
3. var password = $("#inputPassword").val();
4. $.ajax({
5. type: "POST",
6. url: baseuri + "/api/" + "login",
7. dataType: "json",
8. headers: {
9. "Authorization": "Basic " + btoa(username + ":" + password)
10. },
11. success: function (data ){
12. setCookie("username",username,1);
13. setCookie("password",password,1);
14. location.href = "dashboard.html";
15. toastr.success("Login Successful", 'IOT Robot Sensor');
16. },
17. error: function(xhr, status, error) {
18. if(error == "UNAUTHORIZED")
19. {
20. toastr.error("Username/Password incorrect");
21. }
22. else
23. {
24. toastr.error("Status: " + status + " Error: " + error, "Oops!");
25. }
26. }
27. });
28. }

**Global.js**

1. var baseuri = '192.168.1.15:9000';
2. function setCookie(cname, cvalue, exdays) {
3. var d = new Date();
4. d.setTime(d.getTime() + (exdays \* 24 \* 60 \* 60 \* 1000));
5. var expires = "expires="+d.toUTCString();
6. document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
7. }
8. function getCookie(cname) {
9. var name = cname + "=";
10. var ca = document.cookie.split(';');
11. for(var i = 0; i < ca.length; i++) {
12. var c = ca[i];
13. while (c.charAt(0) == ' ') {
14. c = c.substring(1);
15. }
16. if (c.indexOf(name) == 0) {
17. return c.substring(name.length, c.length);
18. }
19. }
20. return "";
21. }
22. function checkCookie() {
23. var user = getCookie("username");
24. if (user != "") {
25. alert("Welcome again " + user);
26. } else {
27. user = prompt("Please enter your name:", "");
28. if (user != "" && user != null) {
29. setCookie("username", user, 365);
30. }
31. }
32. }

**Custom.js**

1. var username = "";
2. var password = "";
3. $(document).ready(function(){
4. username = getCookie("username");
5. password = getCookie("password");
6. if (username != "") {
7. toastr.success("Welcome again " + username);
8. } else {
9. location.href = "login.html";
10. }
11. var width = $( window ).width();
12. var height = $( window ).height();
13. if(width < 640){
14. $("#streamimg").width(width - 60);
15. }
16. });
17. function LedOn(){
18. CallControlApi("lighton");
19. }
20. function LedOff(){
21. CallControlApi("lightoff");
22. }
23. function Forward(){
24. CallControlApi("forward");
25. }
26. function Left(){
27. CallControlApi("left");
28. }
29. function Brake(){
30. CallControlApi("brake");
31. }
32. function Right(){
33. CallControlApi("right");
34. }
35. function Reverse(){
36. CallControlApi("reverse");
37. }
38. function HornOn(){
39. CallControlApi("hornon");
40. }
41. function HornOff(){
42. CallControlApi("hornoff");
43. }
44. function FanOn(){
45. CallControlApi("fanon");
46. }
47. function FanOff(){
48. CallControlApi("fanoff");
49. }
50. function Angle0(){
51. CallMovementApi(0);
52. }
53. function Angle45(){
54. CallMovementApi(45);
55. }
56. function Angle90(){
57. CallMovementApi(90);
58. }
59. function Angle135(){
60. CallMovementApi(135);
61. }
62. function Angle180(){
63. CallMovementApi(180);
64. }
65. function FrontSensor(){
66. $.ajax({
67. type: "GET",
68. url: baseuri + "/api/" + "frontsensor",
69. dataType: "json",
70. headers: {
71. "Authorization": "Basic " + btoa(username + ":" + password)
72. },
73. success: function (data){
74. $("#response").prepend(JSON.stringify(data) + '</br>');
75. toastr.success(data.reading + " cms", 'IOT Robot Sensor');
76. },
77. error: function(xhr, status, error) {
78. toastr.error("Status: " + status + " Error: " + error, "Oops!");
79. }
80. });
81. }
82. function Humidity(){
83. $.ajax({
84. type: "GET",
85. url: baseuri + "/api/" + "humidity",
86. dataType: "json",
87. headers: {
88. "Authorization": "Basic " + btoa(username + ":" + password)
89. },
90. success: function (data){
91. $("#response").prepend(JSON.stringify(data) + '</br>');
92. toastr.success(data.reading + " cms", 'IOT Robot Sensor');
93. },
94. error: function(xhr, status, error) {
95. toastr.error("Status: " + status + " Error: " + error, "Oops!");
96. }
97. });
98. }
99. function Temperature(){
100. $.ajax({
101. type: "GET",
102. url: baseuri + "/api/" + "temperature",
103. dataType: "json",
104. headers: {
105. "Authorization": "Basic " + btoa(username + ":" + password)
106. },
107. success: function (data){
108. $("#response").prepend(JSON.stringify(data) + '</br>');
109. toastr.success(data.reading + " cms", 'IOT Robot Sensor');
110. },
111. error: function(xhr, status, error) {
112. toastr.error("Status: " + status + " Error: " + error, "Oops!");
113. }
114. });
115. }
116. function AllSideSensor(){
117. $.ajax({
118. type: "GET",
119. url: baseuri + "/api/" + "detailsensorreadings",
120. dataType: "json",
121. headers: {
122. "Authorization": "Basic " + btoa(username + ":" + password)
123. },
124. success: function (data){
125. $("#response").prepend(JSON.stringify(data) + '</br>');
126. toastr.success("Left: " + data.detailsensorreadings[0] + " cms,"
127. + "Left1: " + data.detailsensorreadings[1] + " cms,"
128. + "Straight: " + data.detailsensorreadings[2] + " cms,"
129. + "Right1: " + data.detailsensorreadings[2] + " cms,"
130. + "Right: " + data.detailsensorreadings[3] + " cms"
131. , 'IOT Robot Sensor');
132. },
133. error: function(xhr, status, error) {
134. toastr.error("Status: " + status + " Error: " + error, "Oops!")
135. }
136. });
137. }
138. function CallControlApi(control){
139. $.ajax({
140. type: "GET",
141. url: baseuri + "/api/" + control,
142. dataType: "json",
143. headers: {
144. "Authorization": "Basic " + btoa(username + ":" + password)
145. },
146. success: function (data){
147. $("#response").prepend(JSON.stringify(data) + '</br>');
148. var msg = '';
149. msg = control + "Done";
150. toastr.success(msg, 'IOT Robot');
151. },
152. error: function(xhr, status, error) {
153. toastr.error('Error in calling: ' + control + " Status: " + status + " Error: " + error, "Oops!")
154. }
155. });
156. }
157. function CallMovementApi(control){
158. $.ajax({
159. type: "GET",
160. url: baseuri + "/api/move/"+ control,
161. dataType: "json",
162. data : {
163. "value": control
164. },
165. headers: {
166. "Authorization": "Basic " + btoa(username + ":" + password)
167. },
168. success: function (data){
169. $("#response").prepend(JSON.stringify(data) + '</br>');
170. var msg = '';
171. msg = 'Camera moving ' + control + ' degree';
172. toastr.success(msg, 'IOT Robot');
173. },
174. error: function(xhr, status, error) {
175. toastr.error('Error in calling: ' + control + " Status: " + status + " Error: " + error, "Oops!")
176. }
177. });
178. }

# System Testing

For almost every system the last stage before its completion, is to integrate all the modules end to end. Usually this stage is the most difficult one, as many errors appear during the integration. Testing the system and debugging it usually takes more time than its fabrication. This chapter explains the integration, testing, debugging and the results of the overall system.

## Combining of Modules

This Project has four modules. Sensor , Servo motor and dc motors , Communication Server and website on the user end. All these modules first test independently called unit testing . when all module work fine alone then its time to combine the modules and test the whole system .

## Unit Testing

Carrying out unit test is kind of a pre-release of the system. Performing unit test in presence of supervisor ensures that final product had met all requirements. Unit test results listed described in below showing table:

|  |  |  |
| --- | --- | --- |
| **No.** | **Test Case Description** | **Test Result** |
| 1 | Webcam image display | Accepted |
| 2 | Move Forward | Accepted |
| 3 | Move Reverse | Accepted |
| 4 | Turn Left | Accepted |
| 5 | Turn Right | Accepted |
| 6 | Stop | Accepted |
| 7 | Fan | Accepted |
| 8 | LED | Accepted |
| 9 | Ultrasonic sensor reading | Accepted |
| 10 | DHT11 Sensor reading | Accepted |

## Discussion

During the whole period of the project I gained of lot of knowledge on the raspberry pi, motors and programming in python and javascript. If we talk about the achievements out of the project when starting to do the project it was to control the motors using raspberry pi on a robot and transmit that data via any wireless technology to another device and able to collect the data and control the robot or raspberry pi in a real time instance. Out of which all the work was completed. The main achievements that I gained out of the project were that I got to learn programming in python and could learn how to program a user interface web page. Another main achievement was I could learn and understand the raspberry pi technology, the wide applications of raspberry pi and IoT. There are lots of many other areas where the raspberry pi could be used for robotic applications and that are the reasons for me to choose this project.

## Final Product

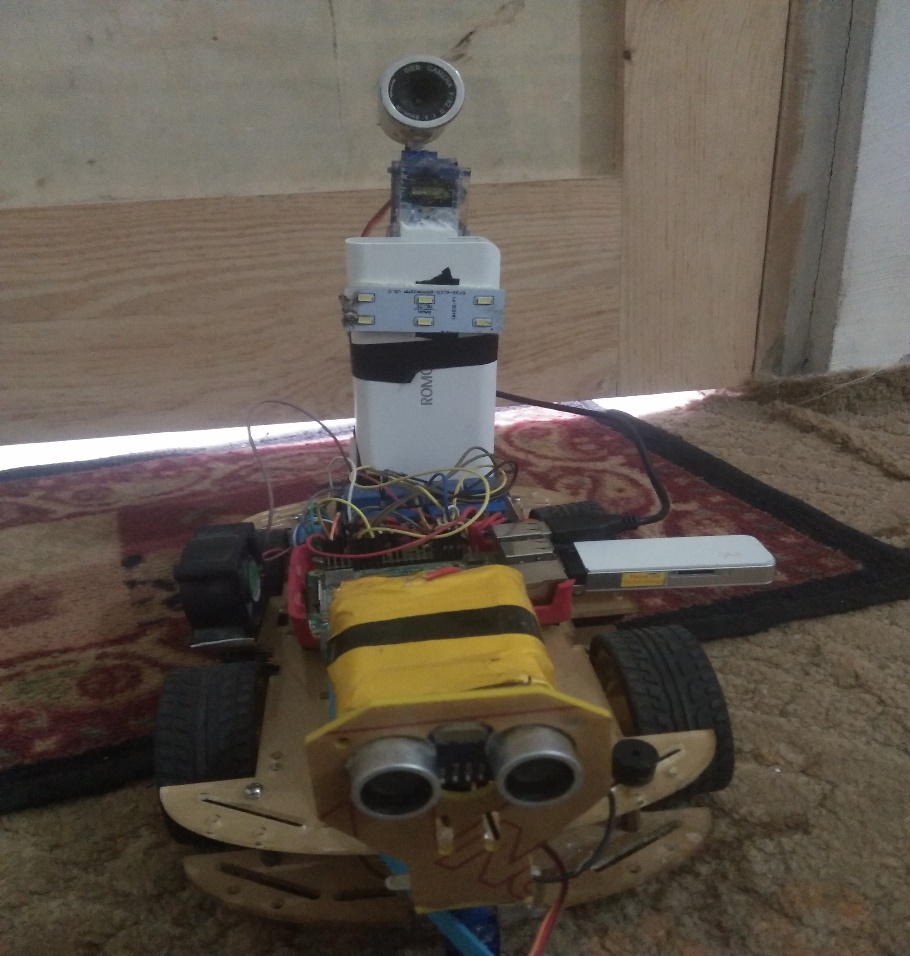


Table 5.1 Final Product

## Future Work

In the future this raspberry pi technology can be used in various different fields of work. The buddy robot can be made autonomous with the help of more sensor, gyroscope, compass and a GPS. So that it can be set to a target or a specific area where in can monitor. The robot can also be developed into an advanced robot toy for young people. Others future works described below:

 Face recognition: All the family members face images are stored in controller when an unknown person will come at door, it will create alert and click the image and send it to user.

 In changing the Mechanical design work using the same concept, different functions as Open the door, Turn on/off switch, bring newspaper for user, etc work can be done.

 Adding the Pneumatics design in Mechanical design robot can walk, go up and down and it will be control from anywhere at any time.

## Conclusions

To get to the aim of a project there will be always a set of objectives, to achieve that objectives we need to know how where and with what resource is the step towards completing the objectives taken. Now in this project too to get to the aim of the project there was a set of objectives, which gradually changed as the project research was completed and then while testing a certain technology the objectives again changed due to the failure of the method. Now the first thing of the project is a good research, I had to do a wide and a strong research before I started to put my objectives as this technology was new in market.

The research for the project was done using Advanced Google search and also from the search engines available in the student portal like tutorials, pi4j, w3school and raspberry pi.

The Google advanced search is the one that was more widely used as it is a new technology and there are very less articles or journals published regarding the raspberry pi technology and IoT.

Each stage of the project was tested after every part of it was completed and then moved on to the next one. During the course of the project I gained knowledge of Java I also gained knowledge of the raspberry pi technology and what the small computer is capable of. After knowing the capabilities of raspberry pi and the applications it could have in the field of robotics, and IoT it actually has made me to think of doing more research work on the raspberry pi for the robotic and IoT applications.

The challenges that I faced during the course of the project were that of the time constrain, as I had to learn about the raspberry pi and then learn programming in python and HTML. Then during the programming of the server client interfaces the problems of calling functions with a button press. One of the main challenges that No output comes when some functions are called from software side. Other than the small problem the buddy robot works fine and meets all its purpose.

If given an opportunity to work again on the same technology i.e. the raspberry pi and IoT technology or on a project like this where the raspberry pi is used for any kind off application I would be happy to take it up.

## References

* <https://docs.python.org/3/>
* <https://www.raspberrypi.org/education/>
* <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
* <https://stackoverflow.com/>
* [https://www.quora.com](https://www.quora.com/)
* <https://api.jquery.com/jquery.ajax/>
* <https://www.w3schools.com/jquery/jquery_ref_ajax.asp>
* https://www.codecademy.com/learn/javascript
* <https://codeseven.github.io/toastr/>
* <https://uwsgidocs.readthedocs.io/en/latest/>
* <https://www.realvnc.com/en/connect/download/viewer/>