# IOT BASED RASPBERRY PI SURVEILLANCE BOT

A Project Report submitted to Don Bosco Institute of Technology

In partial fulfilment of the requirement for the award of

**Degree In**

**ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

Submitted by

**JUSTINE AYROOR 06**

**SUMAN DEB 16**

**SOHAN NAIK 48**

Under the guidance of

**PROF. Iqbal**

**MINI-PROJECT II (ETL504)**

**DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATIONS**

**DON BOSCO INSTITUTE OF TECHNOLOGY**

**(2015-2016)**

Don Bosco Institute of Technology , Kurla West, Mumbai

(2015-2016)

# CERTIFICATE

The group project titled "**IOT BASED RASPBERRY PI SURVEILLANCE BOT**" was carried out by the following students:

| **NAME** | **ROLL NO** | **EXAM SEAT NO** |
|---|---|---|
| **JUSTINE AYROOR** | **06** | |
| **SUMAN DEB** | **16** | |
| **SOHAN NAIK** | **48** | |

Submitted in partial fulfilment of the requirement for the award of Degree in

**Electronics and Telecommunication Engineering**

| | | |
|---|---|---|
| **Project Guide** | **HOD** | **Principal** |

**External Examiner**

## Acknowledgment

Our very sincere thanks to our project guide **Mr.Iqbal** for motivating, co-operating and guiding us throughout the project with their effective skills and huge knowledge base. We would like to thank **Dr. Sudhakar Mande**, Head of Department, for his valuable guidance, support, suggestions and precious time in every possible way throughout the project activity.

We would like to take this opportunity to express our deep gratitude to all the staff who have given their valuable support and co-operation in spite of their busy schedule. Without their advice and help it would have been difficult to complete this work.

We wish to express our deep gratitude towards all our colleagues for their encouragement and moral support

We would like to specially thank **Mrs. Joanne Gomes** from SFIT who is a member of IEEE to conduct a workshop on raspberry pi and this inspired us to take up this project .

# TABLE OF CONTENTS

# IOT BASED RASPBERRY PI SURVEILLANCE BOT

Justine Ayroor (Roll No. 6)
Student, Bachelor of Engineering,
Department of Electronics &
Telecommunication, Don Bosco Institute of
Technology, Mumbai
justine2496@gmail.com

Suman Deb (Roll No. 16)
Student, Bachelor of Engineering,
Department of Electronics &
Telecommunication, Don Bosco Institute of
Technology, Mumbai
suman28795@gmail.com

Sohan Naik (Roll No. 48)
Student, Bachelor of Engineering,
Department of Electronics &
Telecommunication, Don Bosco Institute of
Technology, Mumbai

**Abstract—**

**The technology is a never ending process and these technologies will tend to improve the quality of and ease of living. To be able to design a product using the current technology which is beneficial to the lives of others is a huge contribution to the society. This paper presents the design and implementation of a remote controlled bot which can be controlled wirelessly to move around in the desired direction & give various desired values.**

## 1. INTRODUCTION

Raspberry Pi is a credit card sized computer used to connect the outside world using the GPIO pins. In this project we control the various sensors and motors connected to the raspberry pi to control the movement of the car and send the desired values and images/live stream on the net.

## THE CONCEPT OF THE PROJECT

Raspberry Pi uses a basic programming language called PYTHON . It is used to control the Raspberry pi to the outside world .So basically we program the raspberry pi using Python & FLASK which is a web development framework for python to display & control the bot via a HTML page. This allows us to control the bot wirelessly. The project also allows us to make use of the INTERNET OF THINGS (IOT).

## 2. RASPBERRY PI MODEL B+

### 2.1.What is RPi?

1 The **Raspberry Pi** is a series of credit card–sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools and developing countries.[5][6][7]

2 The original Raspberry Pi and Raspberry Pi 2 are manufactured in several board configurations through licensed manufacturing agreements with Newark element14 (Premier Farnell), RS Components and Egoman. These companies sell the Raspberry Pi online.[8] Egoman produces a version for distribution solely in Taiwan, which can be distinguished from other Pis by their red colouring and lack of FCC/CE marks. The hardware is the same across all manufacturers.

3 The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC),[2] which includes an ARM1176JZF-S 700 MHzprocessor, VideoCore IV GPU,[9] and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB.[3][10] The system has Secure Digital (SD) (models A and B) or MicroSD (models A+ and B+) sockets for boot media and persistent storage.[11]

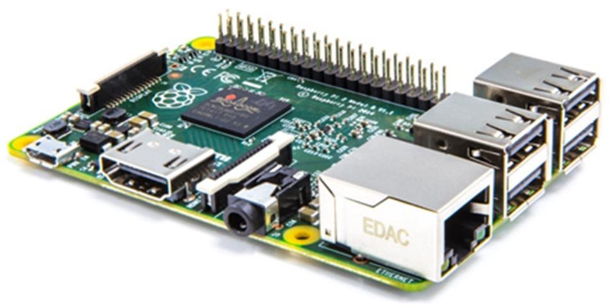
4 In 2014, the Raspberry Pi Foundation launched the Compute Module, which packages a BCM2835 with 512 MB RA

5 M and an eMMC flash chip into a module for use as a part of embedded systems.[12]

6 The Foundation provides Debian and Arch Linux ARM distributions for download.[13] Tools are available for Python as the main programming language, with support for BBC BASIC[14] (via the RISC OS image or the Brandy Basic clone for Linux),[15] C, C++, Java,[16] Perl and Ruby.[17]

7 As of 8 June 2015, about five to six million Raspberry Pis have been sold.[18][19] While already the fastest selling British personal computer, it has also shipped the second largest number of units behind the Amstrad PCW, the "Personal Computer Word-processor", which sold eight million.

8 In early February 2015, the next-generation Raspberry Pi, Raspberry Pi 2, was released.[20] The new computer board is initially available only in one configuration (model B) and features a Broadcom BCM2836 SoC, with a quad-core ARM Cortex-A7 CPU and a VideoCore IV dual-core GPU; 1 GB of RAM with remaining specifications being similar to those of the previous generation model B+. The Raspberry Pi 2 retains the same US$35 price point of the model B,[21] with the US$20 model A+ remaining on sale.



9

## 2.2. Operating System

The Raspberry Pi primarily uses Linux-kernel-based operating systems.

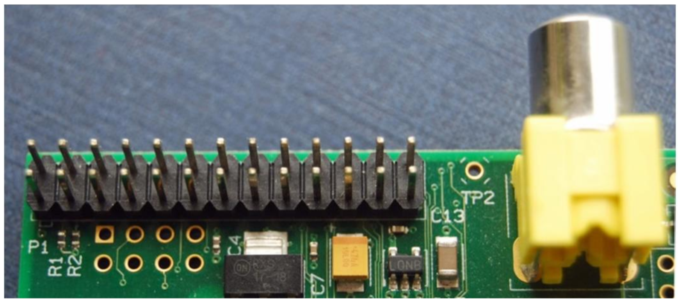
The ARM11 chip at the heart of the Pi (first generation models) is based on version 6 of the ARM. The primary supported operating system is Raspbian,[83] although it is compatible with many others. The current release of Ubuntu supports the Raspberry Pi 2,[84]while Ubuntu, and several popular versions of Linux, do not support the older[85] Raspberry Pi 1 that runs on the ARM11. Raspberry Pi 2 can also run the Windows 10 IoT Core operating system,[86] while no version of the Pi can run traditional Windows.[87] The Raspberry Pi 2 currently also supports OpenELEC and RISC OS.[88]

The install manager for the Raspberry Pi is NOOBS. The operating systems included with NOOBS are:

- Arch Linux ARM
- OpenELEC[89]
- OSMC[90] (formerly Raspbmc[91]) and the Kodi open source digital media center[92]
- Pidora (Fedora Remix)
- Puppy Linux[93]
- RISC OS[94] – is the operating system of the first ARM-based computer.
- Raspbian (recommended for Raspberry Pi 1)[95] – is maintained independently of the Foundation;[96] based on the Debian ARM hard-float (armhf) architecture port originally designed for ARMv7 and later processors (with Jazelle RCT/ThumbEE and VFPv3), compiled for the more limited ARMv6 instruction set of the Raspberry Pi 1. A minimum size of 4 GB SD card is required for the Raspbian images provided by the Raspberry Pi Foundation. There is a Pi Store for exchanging programs.[97][98]
- The Raspbian Server Edition is a stripped version with fewer software packages bundled as compared to the usual desktop computer oriented Raspbian.[99][100]
- The Wayland display server protocol enables efficient use of the GPU for hardware accelerated GUI drawing functions.[101] On 16 April 2014, a GUI shell for Weston called Maynard was released.
- PiBang Linux – is derived from Raspbian.[102]
- Raspbian for Robots – is a fork of Raspbian for robotics projects with Lego, Grove, and Arduino.[103]

## 2.3. GPIO PINS

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the edge of the board, next to the yellow video out socket.



These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Seventeen of the 26 pins are GPIO pins; the others are power or ground pins.

WHAT ARE THEY FOR? WHAT CAN I DO WITH THEM?

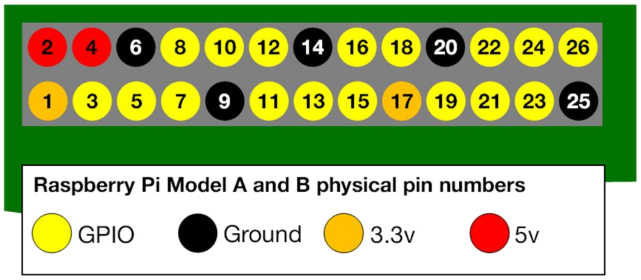
You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere** and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.

GPIO NUMBERING

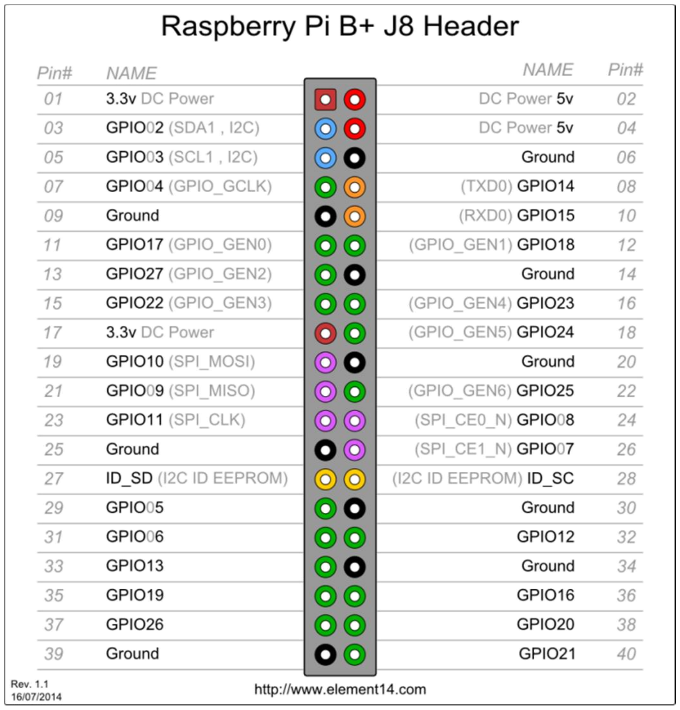
These are the GPIO pins as the computer sees them. The numbers don't make any sense to humans, they jump about all over the place, so there is no easy way to remember them. You will need a printed reference or a reference board that fits over the pins.

PHYSICAL NUMBERING

The other way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card). This is 'physical numbering' and it looks like this:



Pi B+ 40 pin GPIO looks like this:

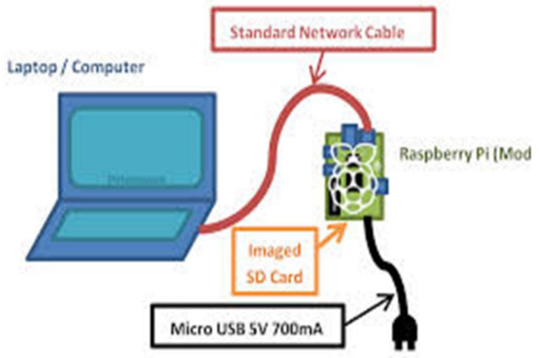


Raspberry Pi B+ J8 Header

| Pin# | NAME | NAME | Pin# |
|---|---|---|---|
| 01 | 3.3v DC Power | DC Power 5v | 02 |
| 03 | GPIO02 (SDA1 , I2C) | DC Power 5v | 04 |
| 05 | GPIO03 (SCL1 , I2C) | Ground | 06 |
| 07 | GPIO04 (GPIO_GCLK) | (TXD0) GPIO14 | 08 |
| 09 | Ground | (RXD0) GPIO15 | 10 |
| 11 | GPIO17 (GPIO_GEN0) | (GPIO_GEN1) GPIO18 | 12 |
| 13 | GPIO27 (GPIO_GEN2) | Ground | 14 |
| 15 | GPIO22 (GPIO_GEN3) | (GPIO_GEN4) GPIO23 | 16 |
| 17 | 3.3v DC Power | (GPIO_GEN5) GPIO24 | 18 |
| 19 | GPIO10 (SPI_MOSI) | Ground | 20 |
| 21 | GPIO09 (SPI_MISO) | (GPIO_GEN6) GPIO25 | 22 |
| 23 | GPIO11 (SPI_CLK) | (SPI_CE0_N) GPIO08 | 24 |
| 25 | Ground | (SPI_CE1_N) GPIO07 | 26 |
| 27 | ID_SD (I2C ID EEPROM) | (I2C ID EEPROM) ID_SC | 28 |
| 29 | GPIO05 | Ground | 30 |
| 31 | GPIO06 | GPIO12 | 32 |
| 33 | GPIO13 | Ground | 34 |
| 35 | GPIO19 | GPIO16 | 36 |
| 37 | GPIO26 | GPIO20 | 38 |
| 39 | Ground | GPIO21 | 40 |

Rev. 1.1
16/07/2014
http://www.element14.com

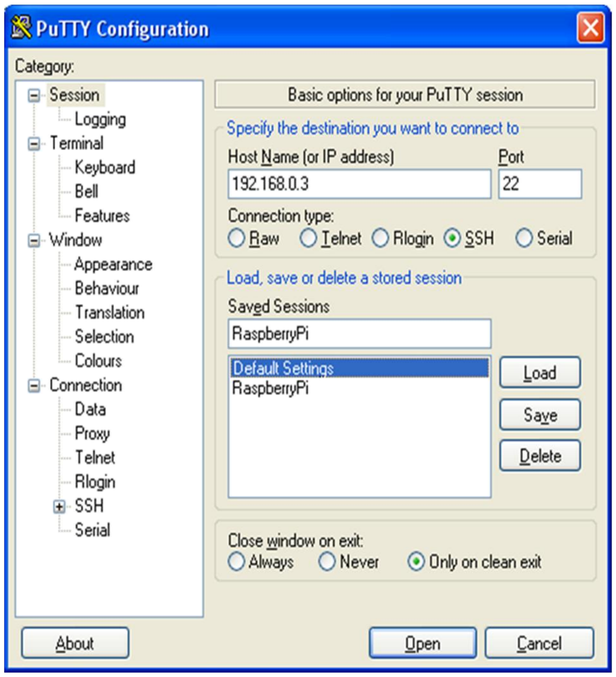
## 2.4. CONNET TO A REMOTE DESKTOP



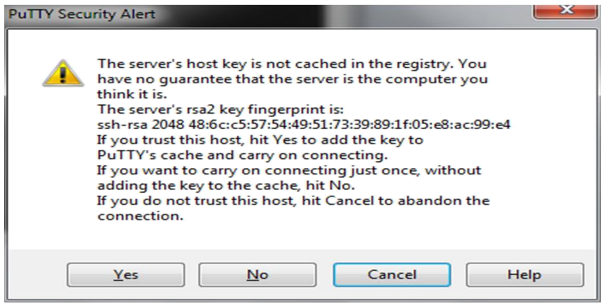
→Go to sudo nano /etc/network/interfaces
→Edit the file save it and connect it
→Static and dynamic IP to configure eth0
→Open file /etc/network/interfaces

```
$ sudo nano /etc/network/interfaces
auto lo eth0
Iface eth0 inet static
    address 192.186.1.48
    netmask 255.255.255.0
    network 192.168.1.0
    gateway 192.168.1.254
    broadcast 192.168.1.255
allow-hotplug wlan0
iface wlan0 inet manual
wpa-
roam/etc/wpa_supplicant/wpa_supplicant.c
onf

iface default inet dhcp
```

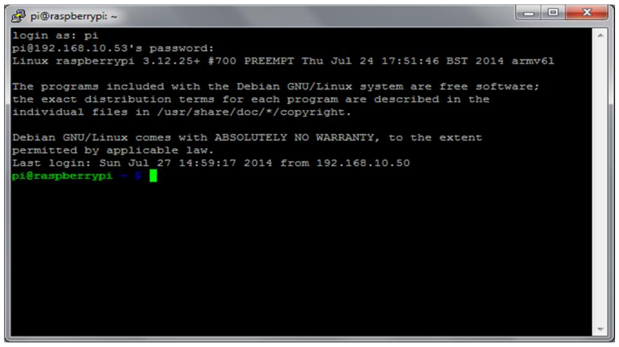
→Install puTTY.exe terminal emulator on you PC
→Insert Rpi's IP as Host name
→Give name to the connection
→Choose SSH
→Save..load..open
→It will open terminal window
→Login to Rpi
→Gain access to RPi's terminal



Host Name (or IP address): 192.168.0.3
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions: RaspberryPi
Default Settings
RaspberryPi
Close window on exit: Always Never [✓ Only on clean exit]



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 48:6c:c5:57:54:49:51:73:39:89:1f:05:e8:ac:99:e4
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.



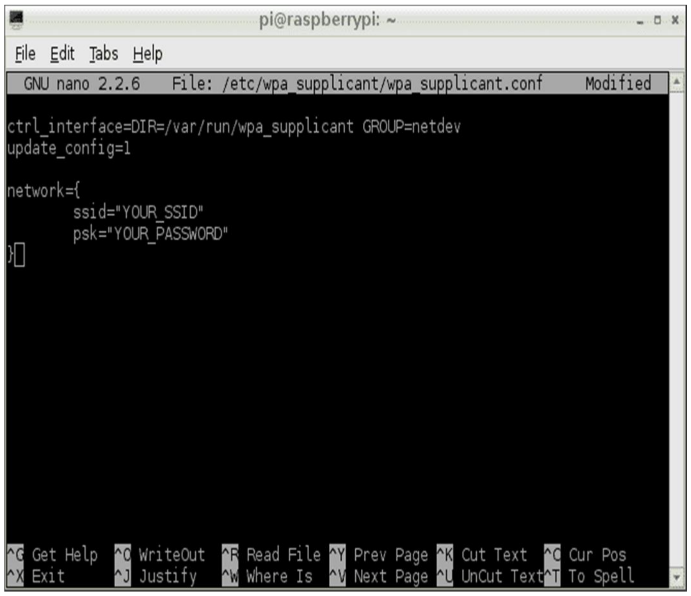
```
login as: pi
pi@192.168.10.53's password:
Linux raspberrypi 3.12.25+ #700 PREEMPT Thu Jul 24 17:51:46 BST 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 27 14:59:17 2014 from 192.168.10.50
pi@raspberrypi ~ $
```

## 2.5. CONNECT TO WIFI

→Dynamic IP to configure wifi
→Open etc/wpa_supplicant/wpa_supplicant.config

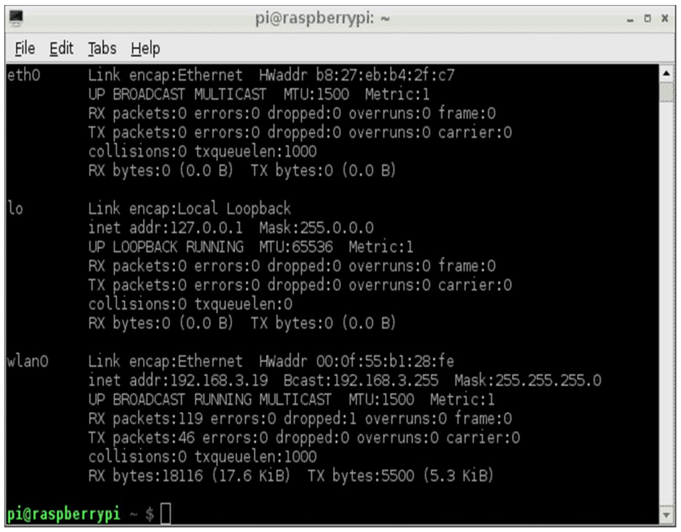
```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.config
Add the following…
Network={
      ssid="ur_APname"
      psk="ur_APpasswd"
      id_str="home"
}
```

**1. CMD (Command Line)** First we're going to make sure our Raspbian operating system is all up to date with the latest drivers. To do this we'll run the following commands `sudo apt-get update sudo apt-get upgrade sudo apt-get autoremove` We'll want to take a backup of the WIFI configuration file before we start to make changes. `sudo cp /etc/wpa_supplicant/wpa_supplicant.conf /etc/wpa_supplicant/wpa_supplicant.conf.bak` N ext up we can edit the file with the "nano" editor. `sudo nano /etc/wpa_supplicant/wpa_supplicant.conf` We want the file to look like the screenshot below. You'll need to swap "YOUR_SSID" and "YOUR_PASSWORD" for your WIFI name and password. Once done, save and close the nano editor



```
  GNU nano 2.2.6    File: /etc/wpa_supplicant/wpa_supplicant.conf      Modified

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        ssid="YOUR_SSID"
        psk="YOUR_PASSWORD"
}

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

all we need to do to get things going! Reboot your Pi with `sudo reboot` Once your Pi is back up and running, we can run `sudo ifconfig` to see if the change we have made has worked. The screenshot below shows that it has worked and that our WLAN adapter has been assigned an IP address (192.168.3.19 in this example)



```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:b4:2f:c7
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:0f:55:b1:28:fe
          inet addr:192.168.3.19  Bcast:192.168.3.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:119 errors:0 dropped:1 overruns:0 frame:0
          TX packets:46 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18116 (17.6 KiB)  TX bytes:5500 (5.3 KiB)

pi@raspberrypi ~ $
```

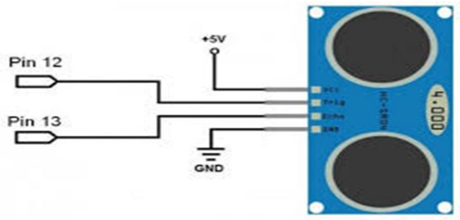
## 3. HARDWARE

### 3.1. HC-SR04 (Ultrasonic Sensor)

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work: (1) Using IO trigger for at least 10us high level signal, (2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. (3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning. Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

5V Supply
Trigger Pulse Input
Echo Pulse Output
0V Ground



WORKING:

Sound consists of oscillating waves through a medium (such as air) with the pitch being determined by the closeness of those waves to each other, defined as the frequency. Only some of the sound spectrum (the range of sound wave frequencies) is audible to the human ear, defined as the “Acoustic” range. Very low frequency sound below Acoustic is defined as “Infrasound”, with high frequency sounds above, called “Ultrasound”. Ultrasonic sensors are designed to sense object proximity or range using ultrasound reflection, similar to radar, to calculate the time it takes to reflect ultrasound waves between the sensor and a solid object. Ultrasound is mainly used because it’s inaudible to the human ear and is relatively accurate within short distances. You could of course use Acoustic sound for this purpose, but you would have a noisy robot, beeping every few seconds. . . .

A basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects. Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting object.

The HC-SR04 Ultrasonic sensor we’ll be using in this tutorial for the Raspberry Pi has four pins: ground (GND), Echo Pulse Output (ECHO), Trigger Pulse Input (TRIG), and 5V Supply (Vcc). We power the module using Vcc, ground it using GND, and use our Raspberry Pi to send an input signal to TRIG, which triggers the sensor to send an ultrasonic pulse. The pulse waves bounce off any nearby objects and some are reflected back to the sensor. The sensor detects these return waves and measures the time between the trigger and returned pulse, and then sends a 5V signal on the ECHO pin.

ECHO will be “low” (0V) until the sensor is triggered when it receives the echo pulse. Once a return pulse has been located ECHO is set “high” (5V) for the duration of that pulse. Pulse duration is the full time between the sensor outputting an ultrasonic pulse, and the return pulse being detected by the sensor receiver. Our Python script must therefore measure the pulse duration and then calculate distance from this.

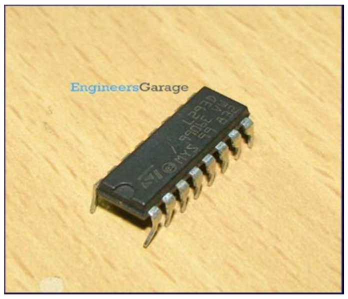
IMPORTANT. The sensor output signal (ECHO) on the HC-SR04 is rated at 5V. However, the input pin on the Raspberry Pi GPIO is rated at 3.3V. Sending a 5V signal into that unprotected 3.3V input port could damage your GPIO pins, which is something we want to avoid! We’ll need to use a small voltage divider circuit, consisting of two resistors, to lower the sensor output voltage to something our Raspberry Pi can handle.

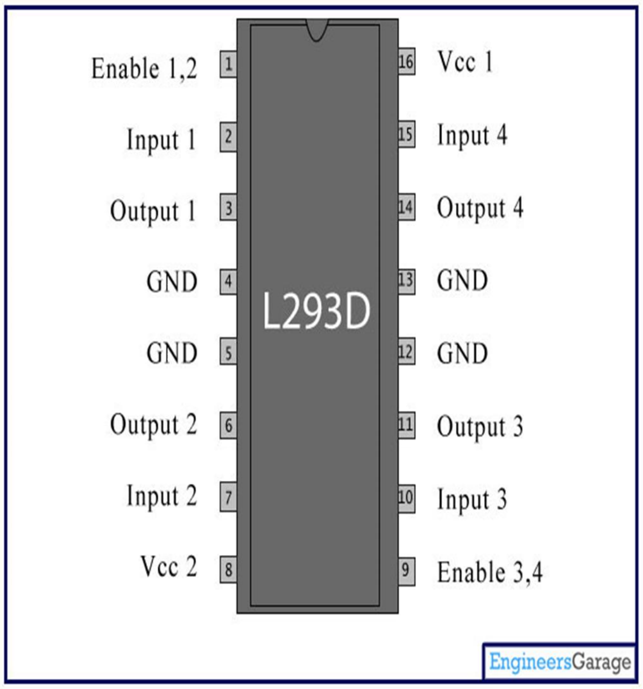
## 3.2. L293D IC (Motor Driver IC)

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt 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.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.





1) Pin Description:

| 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) | Vcc $_2$ |
| 9 | Enable pin for Motor 2; active high | Enable 3,4 |
| 10 | Input 1 for Motor 1 | Input 3 |
| 11 | Output 1 for Motor 1 | Output 3 |
| 12 | Ground (0V) | Ground |
| 13 | Ground (0V) | Ground |
| 14 | Output 2 for Motor 1 | Output 4 |
| 15 | Input2 for Motor 1 | Input 4 |
| 16 | Supply voltage; 5V (up to 36V) | Vcc $_1$ |

## 3.3. DHT-11 (Temperature & Humidity Sensor)

This is a multifunctional sensor that gives you temperature and relative humidity information at the same time. It utilizes a DHT11 sensor that can meet measurement needs of general purposes. It provides reliable readings when environment humidity condition in between 20% RH and 90% RH, and temperature condition in between 0°C and 50°C, covering needs in most home and daily applications that don't contain extreme conditions.

### *Specification:*

- Work Voltage: 3.3V ~ 5V
- Measuring Range:
  Humidity: 20% - 90% RH
  Temperature: 0 ~ 50 °C
- Accuracy:
  Humidity: ±5% RH
  Temperature: ±2°C
- Sensitivity:
  Humidity: ±1% RH
  Temperature: 1°C
- Signal Collecting Period: 2S

## 3.4. SERVO MOTOR

A **servo motor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system.

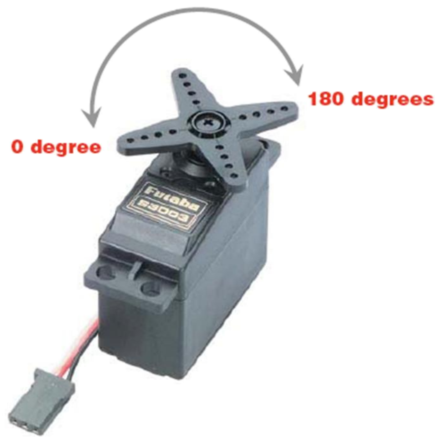
*Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.*

The function of the servo is to receive a control signal that represents a*desired output position of the servo shaft*, and apply power to its DC motor until its shaft turns to that position. It uses the position-sensing device to determine the rotational position of the shaft, so it knows which way the motor must turn to move the shaft to the commanded position. The shaft typically does *not* rotate freely round and round like a DC motor, but rather can only turn 200 degrees or so back and forth.
The servo has a 3 wire connection: power, ground, and control. The power source must be constantly applied; the servo has its own drive electronics that draw current from the power lead to drive the motor.

The control signal is pulse width modulated (PWM), but here the **duration** of the positive-going pulse determines the **position** of the servo shaft. For instance, a 1.520 millisecond pulse is the center position for a Futaba S148 servo. A longer pulse makes the servo turn to a

clockwise-from-center position, and a shorter pulse makes the servo turn to a counter-clockwise-from-center position.
The servo control pulse is repeated every 20 milliseconds. In essence, every 20 milliseconds you are telling the servo, "go here."

To recap, there are two important differences between the control pulse of the servo motor versus the DC motor. First, on the servo motor, duty cycle (on-time vs. off-time) has *no meaning* whatsoever—all that matters is the absolute duration of the positive-going pulse, which corresponds to a commanded output position of the servo shaft. Second, the servo has its own power electronics, so *very little power* flows over the control signal. All power is draw from its power lead, which must be simply hooked up to a high-current source of 5 volts.
Contrast this to the DC motor. On the Handy Board, there are specific motor driver circuits for four DC motors. Remember, a DC motor is like a light bulb; it has no electronics of its own and it requires a large amount of drive current to be supplied to it. This is the function of the L293D chips on the Handy Board, to act as large current switches for operating DC motors.



## 3.5. DC MOTOR

A **DC motor** is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line.

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

DC motors are also the basic motor used in a servo-motor. A servo-motor is comprised of a dc motor, an encoder, an amplifier, a controller, communication software, and IO's

## 3.6. PICAMERA

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion and other video cleverness. You can also use the libraries we bundle with the camera to create effects.

If you're interested in the nitty-gritty, you'll want to know that the module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library.

The camera module is very popular in home security applications, and in wildlife camera traps.

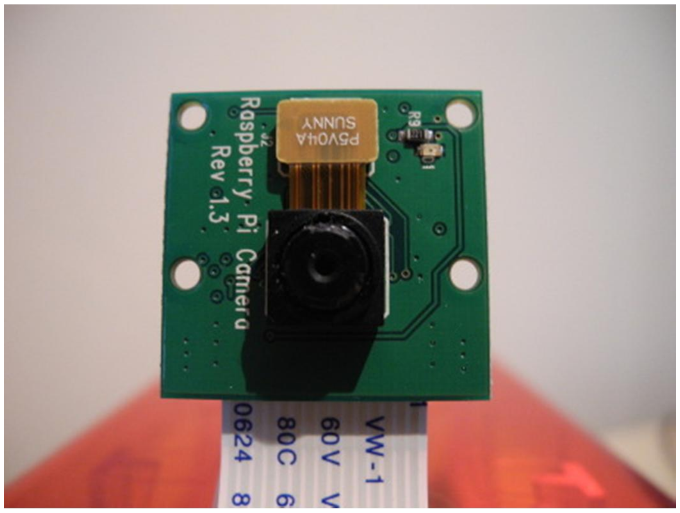
You can also use it to take snapshots.

***Features***

- 5MP sensor
- Wider image, capable of 2592x1944 stills, 1080p30 video
- 1080p video supported
- CSI
- Size: 25 x 20 x 9 m

***Camera Details***

The camera consists of a small (25mm by 20mm by 9mm) circuit board, which connects to the Raspberry Pi's Camera Serial Interface (CSI) bus connector via a flexible ribbon cable. The camera's image sensor has a native resolution of five megapixels and has a fixed focus lens. The software for the camera supports full resolution still images up to 2592x1944 and video resolutions of 1080p30, 720p60 and 640x480p60/90.



## 3.7. RC Car Chassis

Ready made steel car chassis was purchased, easily available at various electronics stores or available online on the internet.

# 4. SOFTWARE

## 4.1. Python

**Phyton** is a widely used general-purpose, high-level programming language.[19][20] Its design philosophy emphasizes code readability, and its syntax allows programmers to express conce pts in fewer lines of code than would be possible in languages such as C++ orJava.[21][22] The language provides constructs intended to enable clear programs on both a small and large scale.[23]

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or proceduralstyles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.[24]

Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Using third-party tools, such as Py2exe or Pyinstaller,[25] Python code can be packaged into stand-alone executable programs for some of the most popular operating systems, allowing the distribution of Python-based software for use on those environments without requiring the installation of a Python interpreter.

CPython, the reference implementation of Python, is free and open-source software and has a community-based development model, as do nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation.

## 4.2. FLASK

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!

Latest Version: 0.10.1

Not only can you use the Raspberry Pi to get data from servers via the internet, but your Pi can also act as a server itself. There are many different web servers that you can install on the Raspberry Pi. Traditional web servers, like Apache or lighttpd, serve the files from your board to clients. Most of the time, servers like these are sending HTML files and images to make web pages, but they can also serve sound, video, executable programs, and much more.

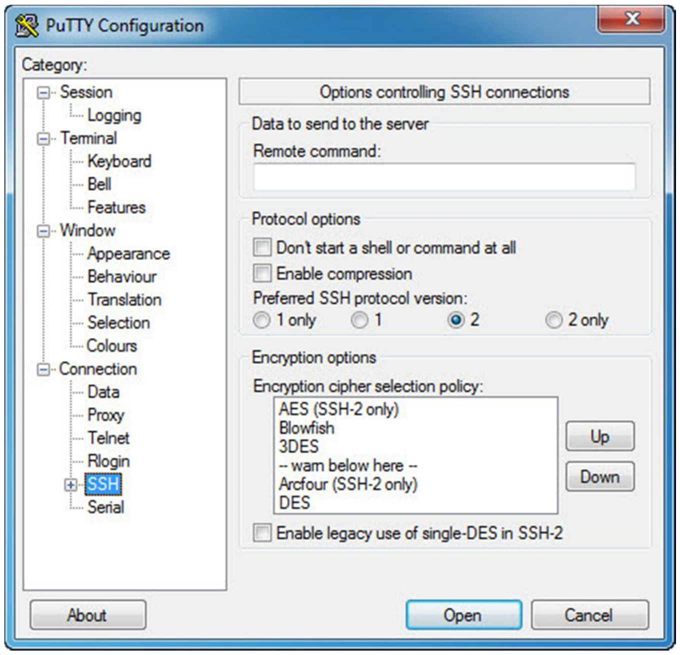
However, there's a new breed of tools that extend programming languages like Python, Ruby, and JavaScript to create web servers that dynamically generate the HTML when they receive HTTP requests from a web browser. This is a great way to trigger physical events, store data, or check the value of a sensor remotely via a web browser. You can even create your own JSON API for an electronics project!

Flask Basics

We're going to use a Python web framework called Flask to turn the Raspberry Pi into a dynamic web server. While there's a lot you can do with Flask "out of the box," it also supports many different extensions for doing things such as user authentication, generating forms, and using databases. You also have access to the wide variety of standard Python libraries that are available to you.

## 4.3. PUTTY

PuTTY is a free implementation of SSH and Telnet for Windows and Unix platforms, along with an `xterm` terminal emulator. It is written and maintained primarily by Simon Tatham.
SSH, Telnet and Rlogin are three ways of doing the same thing: logging in to a multi-user computer from another computer, over a network.

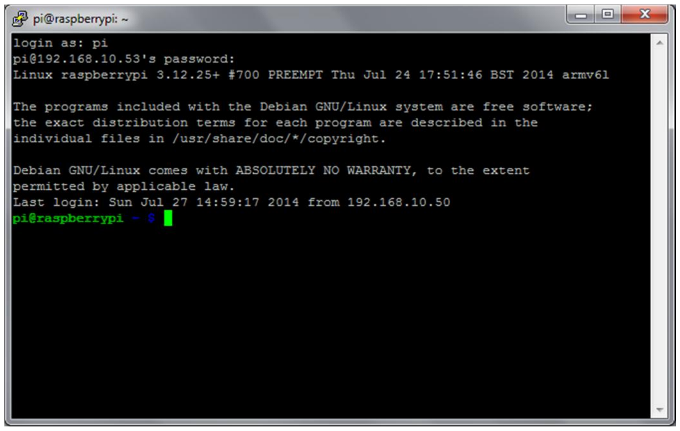


Options controlling SSH connections

Data to send to the server
Remote command:

Protocol options
☐ Don't start a shell or command at all
☐ Enable compression
Preferred SSH protocol version:
○ 1 only ○ 1 ● 2 ○ 2 only

Encryption options
Encryption cipher selection policy:
AES (SSH-2 only)
Blowfish
3DES
-- warn below here --
Arcfour (SSH-2 only)
DES
☐ Enable legacy use of single-DES in SSH-2



```
login as: pi
pi@192.168.10.53's password:
Linux raspberrypi 3.12.25+ #700 PREEMPT Thu Jul 24 17:51:46 BST 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 27 14:59:17 2014 from 192.168.10.50
pi@raspberrypi ~ $
```

Multi-user operating systems, such as Unix and VMS, usually present
a command-line interface to the user, much like the `Command Prompt'
or `MS-DOS Prompt' in Windows. The system prints a prompt, and you
type commands which the system will obey.

Using this type of interface, there is no need for you to be sitting
at the same machine you are typing commands to. The commands,
and responses, can be sent over a network, so you can sit at one computer and give commands to another one, or even to more than one.

SSH, Telnet and Rlogin are _network protocols_ that allow you to do
this. On the computer you sit at, you run a _client_, which makes a network connection to the other computer (the _server_). The network
connection carries your keystrokes and commands from the client to the server, and carries the server's responses back to you.

These protocols can also be used for other types of keyboard-based interactive session. In particular, there are a lot of bulletin
boards, talker systems and MUDs (Multi-User Dungeons) which support
access using Telnet. There are even a few that support SSH.

## 4.4. UWSGI (Application Server)

The uWSGI project aims at developing a full stack for building hosting services.

Application servers (for various programming languages and protocols), proxies, process managers and monitors are all implemented using a common api and a common configuration style.

Thanks to its pluggable architecture it can be extended to support more platforms and languages.

Currently, you can write plugins in C, C++ and Objective-C.

The “WSGI” part in the name is a tribute to the namesake Python standard, as it has been the first developed plugin for the project.

Versatility, performance, low-resource usage and reliability are the strengths of the project (and the only rules followed).

uWSGI is a deployment option on servers like nginx, lighttpd, and cherokee; seeFastCGI and Standalone WSGI Containers for other options. To use your WSGI application with uWSGI protocol you will need a uWSGI server first. uWSGI is both a protocol and an application server; the application server can serve uWSGI, FastCGI, and HTTP protocols.

## 4.5. NGINX (Web Server)

**Nginx** (pronounced "engine x") is a web server. It can act as a reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAPprotocols, as well as a load balancer and an HTTP cache.

Created by Igor Sysoev in 2002, Nginx runs on Unix, Linux, BSD variants, OS X, Solaris, AIX, HP-UX, and Windows.[5] Released under the terms of a BSD-like license, Nginx is free and open source software.

Nginx can be deployed to serve dynamic HTTP content on the network using FastCGI, SCGI handlers for scripts, WSGI application servers or Phusion Passenger modules, and it can serve as a software load balancer.[8]

Nginx uses an asynchronous event-driven approach to handling requests, similar to Apache HTTP Server Event MPM model. Nginx's modular event-driven architecture[9] can provide more predictable performance under high loads.[10]

According to Netcraft's October 2015 Web Server Survey,[11] Nginx was found to be the second most widely used web server across all "active" sites (15.33% of surveyed sites) and for the top million busiest sites (23.66% of surveyed sites). According to W3Techs, it was used by 29.7% of the top 1 million websites, 39.5% of the top 100,000 websites, and by

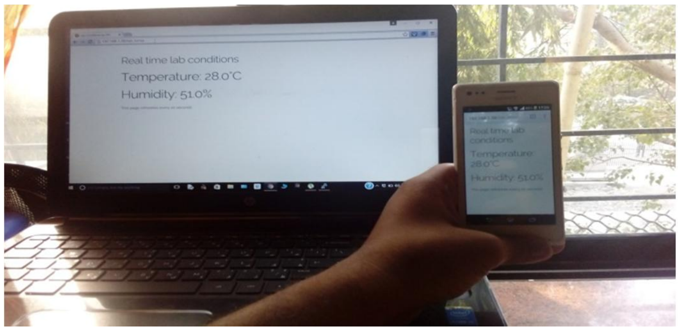
47.6% of the top 10,000 websites.[12] According to BuiltWith, it is used on 36.6% of the top 10,000 websites, and its growth within the top 10k, 100k and 1 million segments increased.[13] Wikipedia uses Nginx as its SSL termination proxy.[14] As of OpenBSD release 5.2 (1 November 2012), Nginx became part of the OpenBSD base system, providing an alternative to the system's fork of Apache 1.3, which it was intended to replace,[15] but it was later replaced by OpenBSD's own httpd

### 4.6. SKELETON CSS

SKELETON CSS
Light as a feather at ~400 lines & built with mobile in mind.
Styles designed to be a starting point, not a UI framework.
Quick to start with zero compiling or installing necessary.

**CSS frameworks** are pre-prepared software frameworks that are meant to allow for easier, more standards-compliant web design using the Cascading Style Sheets language. Most of these frameworks contain at least a grid. More functional frameworks also come with more features and additional JavaScript based functions, but are mostly design oriented and unobtrusive. This differentiates these from functional and full JavaScript frameworks.

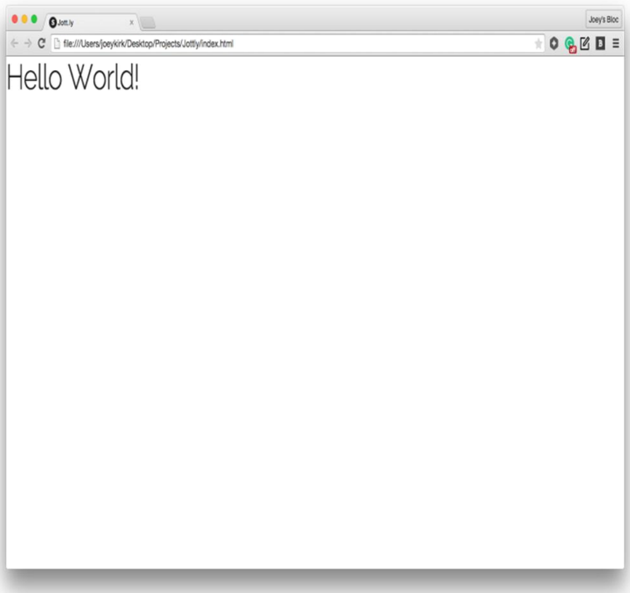
Two notable and widely used examples are Bootstrap or Foundation.



CSS frameworks offer different modules and tools:

- reset style sheet
- grid especially for responsive web design
- web typography
- set of icons in sprites or icon fonts
- styling for tooltips, buttons, elements of forms
- parts of graphical user interfaces like accordion, tabs, slideshow or moda l windows (Lightbox)
- equalizer to create equal height content
- often used css helper classes (*left*, *hide*)

Bigger frameworks use a CSS interpreter like LESS or SASS.



Hello World!

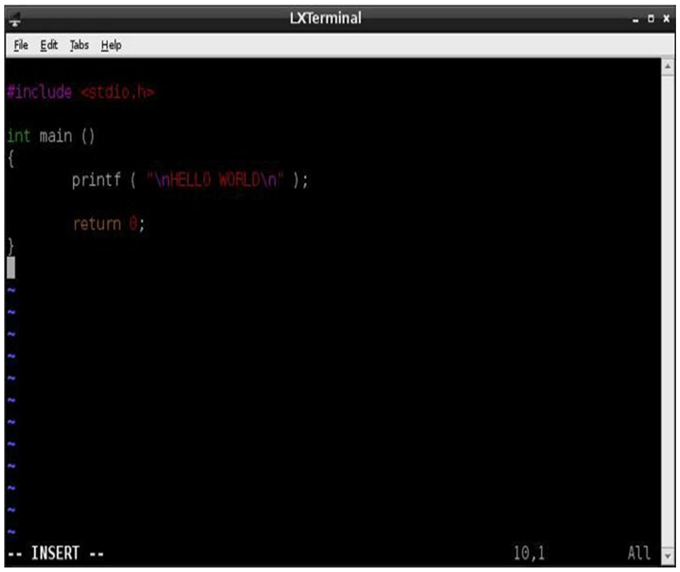
## 4.7. VIM

**Vim** (/vɪm/;[3] a contraction of **Vi IMproved**) is a clone of Bill Joy's vi editor for Unix. It was written by Bram Moolenaar based on source for a port of the Stevie editor to the Amiga[4] and first released publicly in 1991. Vim is designed for use both from acommand-line interface and as a standalone application in a graphical user interface. Vim is free and open source softwareand is released under a license that includes some charityware clauses, encouraging users who enjoy the software to consider donating to children in Uganda.[5] The license is compatible with the GNU General Public License.

Although Vim was originally released for the Amiga, Vim has since been developed to be cross-platform, supporting many other platforms. In 2006, it was voted the most popular editor amongst *Linux Journal* readers.

Like vi, Vim's interface is not based on menus or icons but on commands given in a text user interface; its GUI mode, **gVim**, adds menus and toolbars for commonly used commands but the full functionality is still expressed through its command line mode. Vi (and by extension Vim) tends to allow a typist to keep their fingers on the home row, which can be an advantage for a touch typist.[21]

Vim has a built-in tutorial for beginners (accessible through the "vimtutor" command). There is also the Vim Users' Manual that details Vim's features. This manual can be read from within Vim, or found online.[22][23]



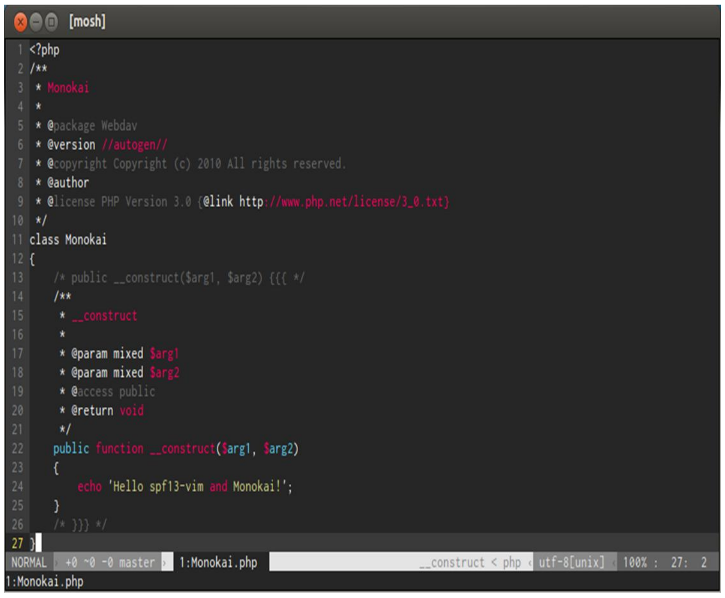
```
#include <stdio.h>

int main ()
{
        printf ( "\nHELLO WORLD\n" );

        return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                        10,1          All
```

Vim also has a built-in help facility (using the `:help` command) that allows users to query and navigate through commands and features.

Vim has a vi compatibility mode but when not in this mode Vim has many enhancements over vi.[26] However, even in compatibility mode, Vim is not 100% compatible with vi as defined in the Single Unix Specification[27] and POSIX (e.g., Vim does not support vi's open mode, only visual mode). Vim has nevertheless been described as “very much compatible with Vi”.[28]

Some of Vim's enhancements include completion, comparison and merging of files (known as vimdiff), a comprehensive integrated help system, extended regular expressions,scripting languages (both native and through alternative scripting interpreters such as Perl, Python, Ruby, Tcl, etc.) including support for plugins, a graphical user interface (known as gvim), limited integrated development environment-like features, mouse interaction (both with and without the

GUI), folding, editing of compressed or archived files in gzip,bzip2, zip, and tar format and files over network protocols such as SSH, FTP, and HTTP, session state preservation, spell checking, split (horizontal and vertical) and tabbed windows, Unicode and other multi-language support, syntax highlighting, trans-session command, search and cursor position histories, multiple level and branching undo/redohistory which can persist across editing sessions, and visual mode.

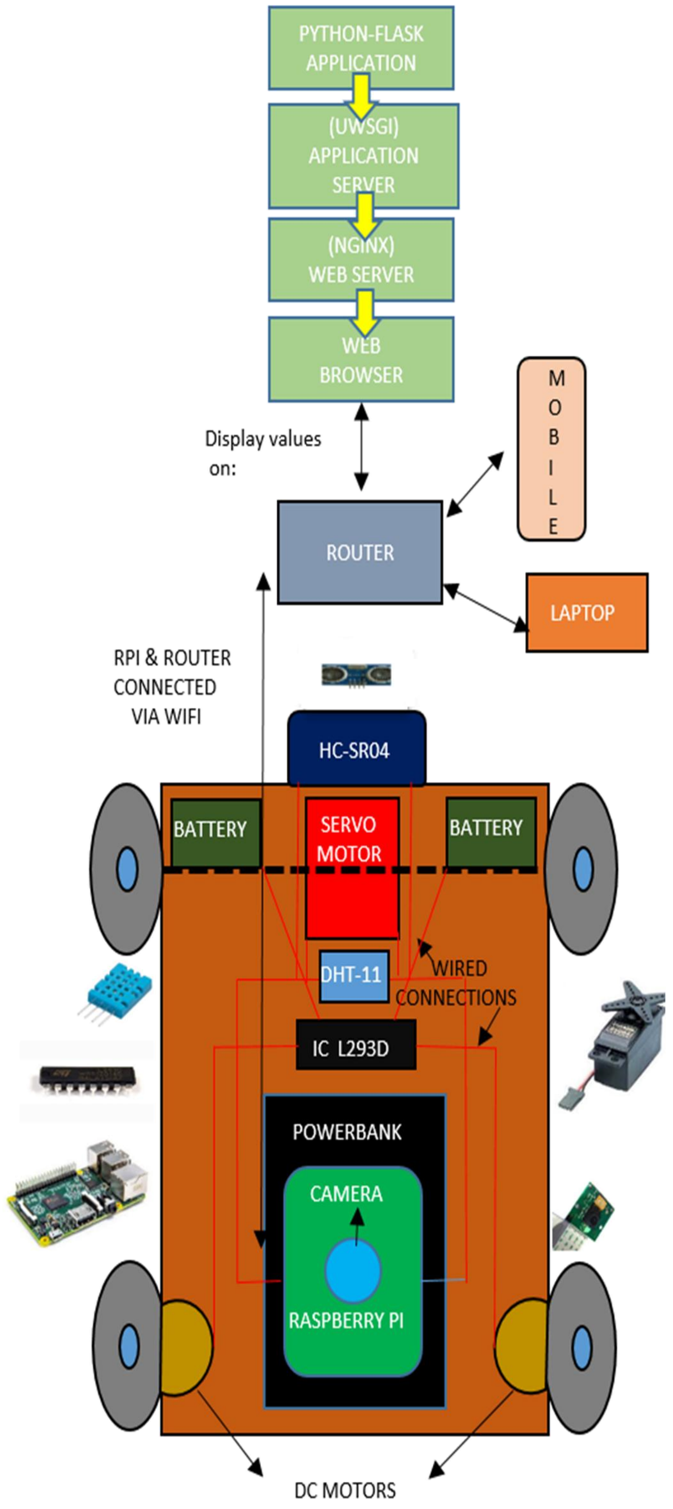


```
[mosh]
<?php
/**
 * Monokai
 *
 * @package Webdav
 * @version //autogen//
 * @copyright Copyright (c) 2010 All rights reserved.
 * @author
 * @license PHP Version 3.0 {@link http://www.php.net/license/3_0.txt}
 */
class Monokai
{
    /* public __construct($arg1, $arg2) {{{ */
    /**
     * __construct
     *
     * @param mixed $arg1
     * @param mixed $arg2
     * @access public
     * @return void
     */
    public function __construct($arg1, $arg2)
    {
        echo 'Hello spf13-vim and Monokai!';
    }
    /* }}} */
}
NORMAL  +0 ~0 -0 master  1:Monokai.php            __construct < php  utf-8[unix]  100% :  27:  2
1:Monokai.php
```

Vim has a vi compatibility mode but when not in this mode Vim has many enhancements over vi.[26] However, even in compatibility mode, Vim is not 100% compatible with vi as defined in the Single Unix Specification[27] and POSIX (e.g., Vim does not support vi's open mode, only visual mode). Vim has nevertheless been described as “very much compatible with Vi”.[28]

Some of Vim's enhancements include completion, comparison and merging of files (known as vimdiff), a comprehensive integrated help system, extended regular expressions,scripting languages (both native and through alternative scripting interpreters such as Perl, Python, Ruby, Tcl, etc.) including support for plugins, a graphical user interface (known as gvim), limited integrated development environment-like features, mouse interaction (both with and without the GUI), folding, editing of compressed or archived files in gzip,bzip2, zip, and tar format and files over network protocols such as SSH, FTP, and HTTP, session state preservation, spell checking, split (horizontal and vertical) and tabbed windows, Unicode and other multi-language support, syntax highlighting, trans-session command, search and cursor position histories, multiple level and branching undo/redohistory which can persist across editing sessions, and visual mode.

## 5. FLOW CHART

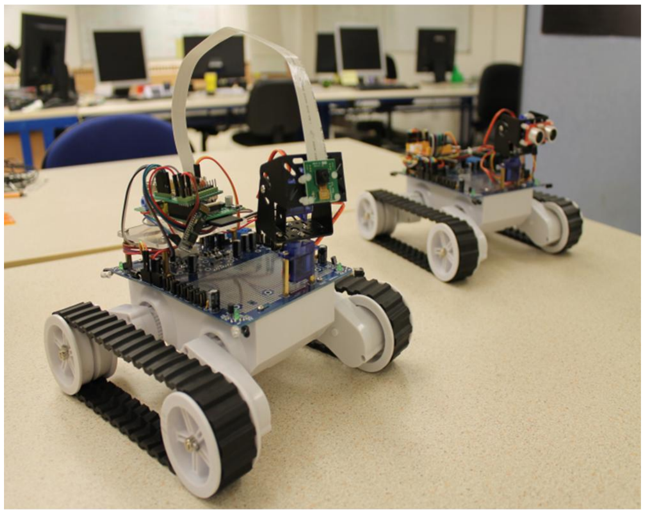

PYTHON-FLASK APPLICATION
(UWSGI) APPLICATION SERVER
(NGINX) WEB SERVER
WEB BROWSER
Display values on:
MOBILE
ROUTER
LAPTOP
RPI & ROUTER CONNECTED VIA WIFI
HC-SR04
BATTERY
SERVO MOTOR
BATTERY
DHT-11
WIRED CONNECTIONS
IC L293D
POWERBANK
CAMERA
RASPBERRY PI
DC MOTORS

**6. DESIGN COST TABLE**

| Hardware | Number | Cost |
|---|---|---|
| Raspberry Pi kit | 1 | 2500 |
| HC-SR04 | 1 | 80 |
| L293D IC | 2 | 70 |
| Servo motor | 1 | 500 |
| Dc motors | 4 | 80 |
| Rpi Camera | 1 | 2000 |
| RC chassis | 1 | 100 |

Total= 4330

## 7. FUTURE SCOPE



→Military applications
→Specific designs can be used on barren lands for various purposes
Example: (getting live feeds without physical presence) and scientific applications .
→Getting feeds from nuclear affected places restricting human movements.

## 8.CONCLUSION

→Hence we can control the bot using IOT and portable RPi.
→We can also get live feeds (images and videos) and also sense the temperature and humidity of a specific area as well as get the approx. distance between the bot and obstacle in front on a browser.
→We learned python and html along with various other softwares such as NGINX and UWSGI used to link the code to the page for our codes to run and execute.

## 9.REFERENCE:

- http://en.wikipedia.org/wiki/Raspberry_Pi
- http://elinux.org/Rpi_Datasheet_201_Raspberry_Pi_Computer
- http://elinux.org/RPi_Hardware
- http://elinux.org/R-Pi_Hub
- http://www.element14.com/community/docs/DOC-43016/l/broadcom-datasheet-for-bcm2835-soc-used-in-raspberry-pi
- http://www.raspberrypi.org/help/
- http://www.cpdforteachers.com/resources