**Practical 7 :- Interfacing GPS module with Raspberry Pi**

* **GPS** stands for Global Positioning System and used to detect the Latitude and Longitude of any location on the Earth, with exact UTC time (Universal Time Coordinated).
* GPS module is the main component in our vehicle tracking system.
* This device receives the coordinates from the satellite for each and every second, with time and date.
* **GPS module** sends the data related to tracking position in real time, and it sends so many data in NMEA format.
* NMEA format consists of sentence starts from **$GPGGA,** the coordinates, time and other useful information.
* **GPGGA** is referred to **Global Positioning System Fix Data**.
* GPGGA string contains following co-ordinates separated by commas.

# Neo 6m v2 GPS Module

* This board features the u-blox NEO-6M GPS module with antenna and built-in EEPROM. This is compatible with various flight controller boards designed to work with a GPS module.
* EEPROM is used for saving the configuration data when powered off.
* Power Supply Range: 3 V to 5 V  Default Baud Rate: 9600 bps

# Connect GPS Module to RPi

Connection is very simple. Requires 4 Female to Female Jumper Wires

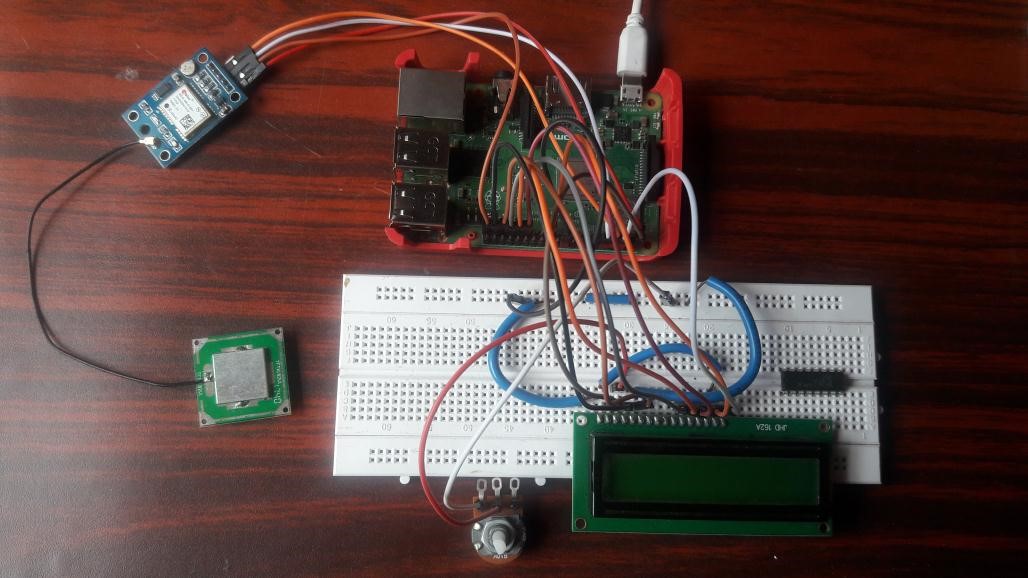
|  |  |  |  |
| --- | --- | --- | --- |
| **Neo 6m V2 GPS Board Pin** | **Raspberry Pi**  **Details**  **Physical Pin** | | **Raspberry Pi Function** |
| **VCC** | **Power** | **Pin 1** | **3.3V Power** |
| **GND** | **Common Ground** | **Pin 39** | **GND** |
| **TXD** | **Data Output** | **Pin 10** | **(UART\_RXD0) GPIO15** |
| **RXD** | **Data Input** | **Pin 8** | **(UART\_TXD0) GPIO14** |

# Connect 2x16 LCD Display to RPi

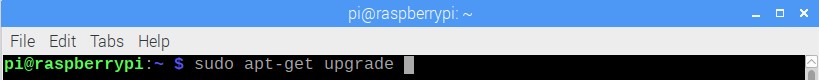
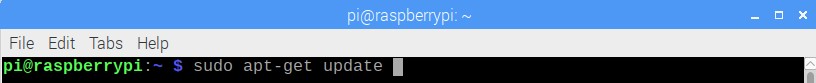
A 16x2 LCD is used for displaying all messages. A 10k Potentiometer is also used with LCD for controlling the contrast. I used following GPIO pins to connect LCD to Raspberry Pi’s GPIO Pins:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **LCD Display Board Pin** | **RPI Physical Pin** | **Raspberry Function** |
| **4** | **RS** | **37** | **GPIO26** |
| **6** | **E** | **35** | **GPIO19** |
| **11** | **D4** | **33** | **GPIO13** |
| **12** | **D5** | **31** | **GPIO6** |
| **13** | **D6** | **29** | **GPIO5** |
| **14** | **D7** | **23** | **GPIO11** |

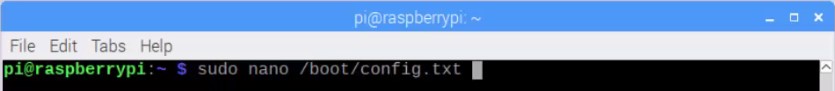
**Connect GPS module to Raspberry Pi’s GPIO Pins by using Female to Female Jumper wires Connect 2x16 LCD Display to Raspberry Pi’s GPIO Pins with the help of breadboard and jumper wires.**



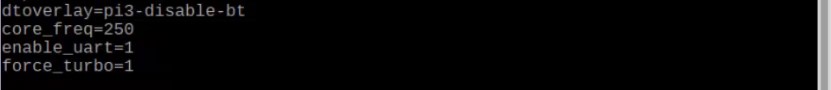
## Step 1: Update Raspberry Pi



## Step 2: edit the /boot/config.txt file



At the bottom of this file, add the above lines



The ***dtoverlay=pi3-disable-bt*** disconnects the bluetooth from the *ttyAMA0*, this is to allow us access to use the full UART power available via *ttyAMAO* instead of the mini UART ttyS0.

Save with Ctrl+X, yes and press Enter.

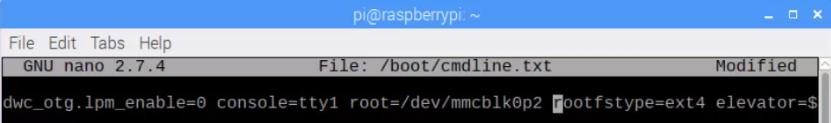
## Step 3: Before proceeding, make a copy of cmdline.txt file



## Now, edit file cmdline.txt



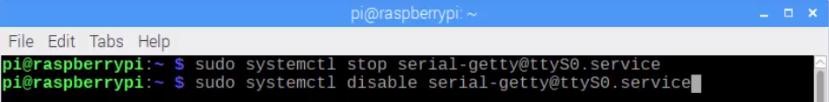
**Remove** console=serial0,115200 **and Modify** root=/dev/mmcblk0p2



Save with Ctrl+X, yes and press Enter.

**Step 4: Reboot Raspberry Pi using the command *sudo reboot***

## Step 5: Stop and disable the Pi’s serial ttyS0 service



The following commands can be used to enable it again if needed

|  |
| --- |
| sudo systemctl enable serial-getty@ttyS0.service sudo systemctl start serial-getty@ttyS0.service |

**Step 6: Reboot Raspberry Pi using the command *sudo reboot***

## Step 7: Now, Enable the ttyAMA0 service



## Verify it using *ls –l /dev* command



## Step 8: Install minicom and pynmea2

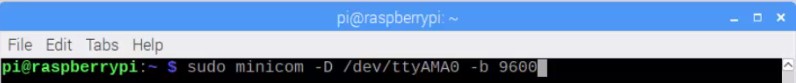
**Install minicom package which is used to connect to the GPS module and make sense of the data.**



**Install pynmea2 library which is used to parse the received data.**



**Step 9: Use minicom command to test our GPS module is working fine.**



9600 represents the baud rate at which the GPS module communicates.

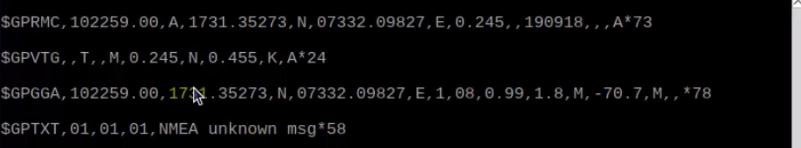
Here, we can see NMEA sentences .**NMEA format** consist several sentences, in which we only need one sentence. This sentence starts from **$GPGGA** and contains the coordinates



Sometimes we received sentences which contains **unknown msg\*58,** but this is not an error, actually it may takes some time to track your GPS module. (Even for the first time more than 20-30 minutes.)

I suggest keep your GPS module's antenna in open space (e.g. near the window) To exit from above window, Press Ctrl+A, and Press x and Enter Key.

## Step 10: The above same test can also be done using cat command



This sentence gives you Latitude found after two commas and Longitude found after four commas.

**Step 11:** Write Python script to display your current **Latitude and Longitude on LCD Display.**

**Python Script (gpsdemo.py)**

import time import serial import string import pynmea2 import RPi.GPIO as gpio

import Adafruit\_CharLCD as LCD

gpio.setmode(gpio.BCM)

lcd = LCD.Adafruit\_CharLCD(rs=26, en=19,

d4=13, d5=6, d6=5, d7=11, cols=16, lines=2)

lcd.message("MSD Gurukul\n Welcomes You") time.sleep(2)

lcd.clear()

lcd.message("GPS Demo")

time.sleep(2)

lcd.clear()

port = "/dev/ttyAMA0" # the serial port to which the pi is connected.

#create a serial object

ser = serial.Serial(port, baudrate = 9600, timeout = 0.5)

try: while 1: try:

data = ser.readline() except: print("loading")

#wait for the serial port to churn out data

if data[0:6] == '$GPGGA': # the long and lat data are always contained in the GPGGA string of the NMEA data

msg = pynmea2.parse(data)

latval = msg.lat #parse the latitude and print concatlat = "Lat:" + str(latval) print(concatlat) lcd.set\_cursor(0,0)

lcd.message(concatlat)

#parse the longitude and print

longval=msg.lon

concatlong=”Long:”+str(longval)

print(concatlong)

lcd.set\_cursor(0,1)

lcd.message(concatlong)

time.sleep(0.5)

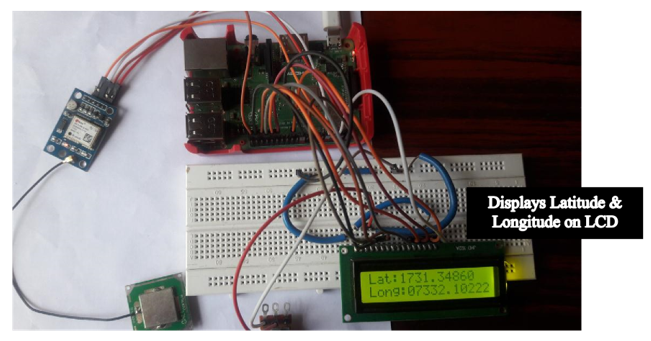
except KeyboardInterrupt:

lcd.clear()

lcd.message(“Thank You”)

time.sleep(2)

**Run Python Script on terminal using sudo command (sudo gpsdemo.py)**



**Practical 8: IoT based Web Controlled Home Automation Using Raspberry Pi.**

# Installation Manual

Controlling AC appliances with the click of buttons on a webpage using internet. It is possible to control your Home appliances from anywhere in the world. This web server can be run from any device which can run HTML applications, like Smart Phone, tablet, computer etc.

**Hardware Requirements**

1. Raspberry Pi Model B/B+ 5v Relays
2. LEDs to test.
3. Breadboard
4. AC lamp to Test
5. Jumper wires

 **Software Requirements**

1. Raspbian Stretch OS
2. WebIOPi frame work

**1. Connect your 5v Relay with Raspberry Pi's GPIO Pins using Jumper wires (FemaleFemale).**

|  |  |  |  |
| --- | --- | --- | --- |
| Relay Board Pin | Function | RPI Physical Pin | Raspberry Function |
| **+5v** | + 5V Power | 4 | 5V |
| **I/P** | Data In | 7 | GPIO 4 |
| **GND** | Ground | 6 | GND |

## LED to Test : Connect LED to Raspberry Pi using breadboard

|  |  |  |  |
| --- | --- | --- | --- |
| LED Terminal | RPI Physical Pin | LED Terminal | RPI Physical Pin |
| Positive | 37 | Negative | 39 |

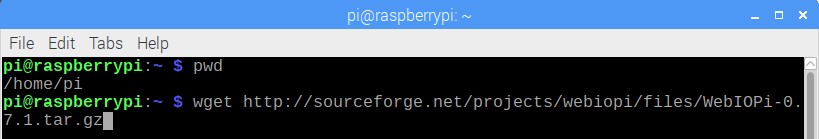


## Step 1: Download the WebIOPi Framework file

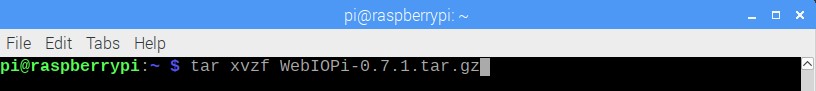
Use wget command to get the installer file of WebIOPi framework from sourceforge page

**Make sure**

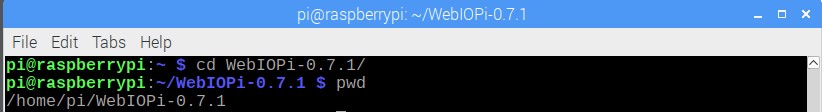
**you are in home directory.**



Extract the file using tar command.



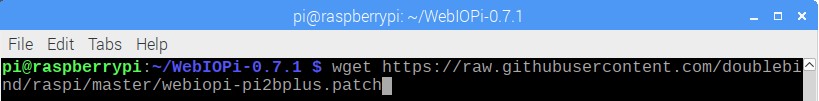
Now, Go to the WebIOPi Directory.



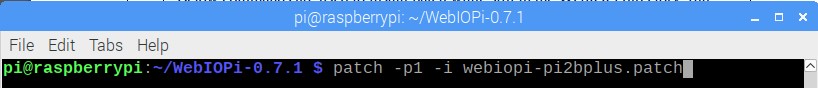
## Step 2: Install patch file

At this point before running the setup, we need to **install a patch as this version of the WebIOPi** does not work with the raspberry pi 3.

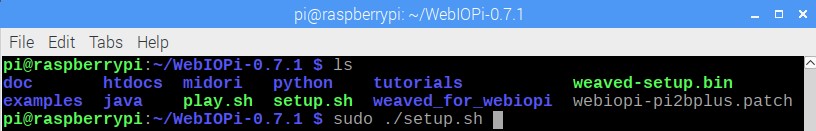
Download patch file using wget command



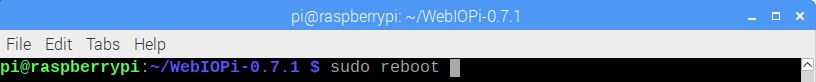
Install patch file using patch command



## Step 3: Install setup of WebIOPi framework, Run setup file

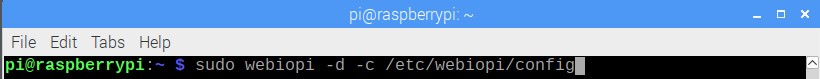


Keep saying yes if asked to install any dependencies during setup installation. When done, reboot your pi

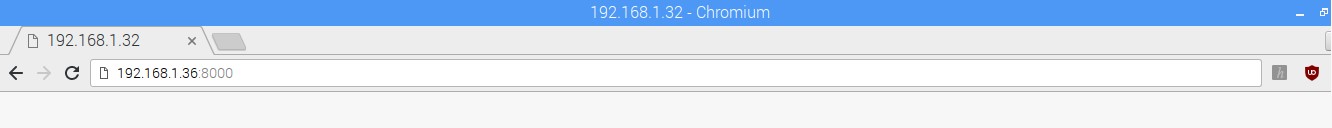


### Step 4: Test WebIOPi Installation

We will need to test our WebIOPi installation to be sure everything works fine as desired. **Run following command on terminal**



## Now, open web browser and connect to [http://PI’s](http://pi's/) IP address:8000



The system will prompt you for username and password.

Username: webiopi

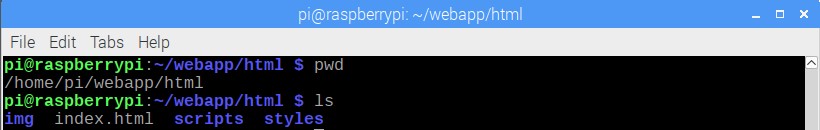
Password: raspberry

After the login, **click on the GPIO header link**. Test your LED which is connected to raspberry Pi’s GPIO Pins. In my case, I have used Physical Pin no 37 of Pi. So set it as **output. Click the pin 37 button to turn on or off the LED**. This way we can control the Raspberry Pi GPIO using WebIOPi.

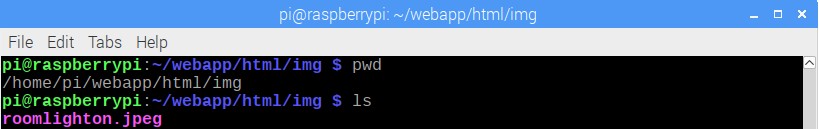
After the test, if everything worked as described, then we can go back to the terminal and stop the program with CTRL + C.

**Step 4: Building the Web Application for Raspberry Pi Home Automation**

## Create below directory structure



Place image inside **img** folder



C

reate

**smarthome.js**

file inside

**scripppts**

folder

**S**

**marthome.**

**js file**

**webiopi().ready(function() {**

**webiopi().setFunction(4,"out");**

**var content, button;**

**content = $("#content");**

**button = webiopi().createGPIOButton(4,"Room 1");**

**content.append(button);**

**})**

**;**

C

reate

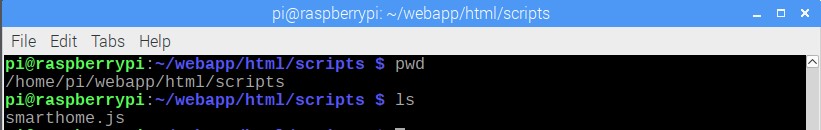
**smarthome.css**

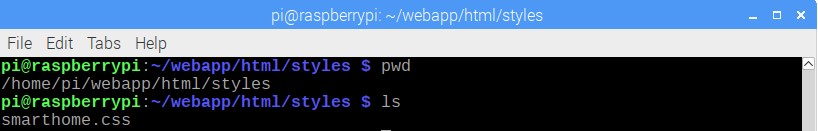
file inside

**s**

**tyles**

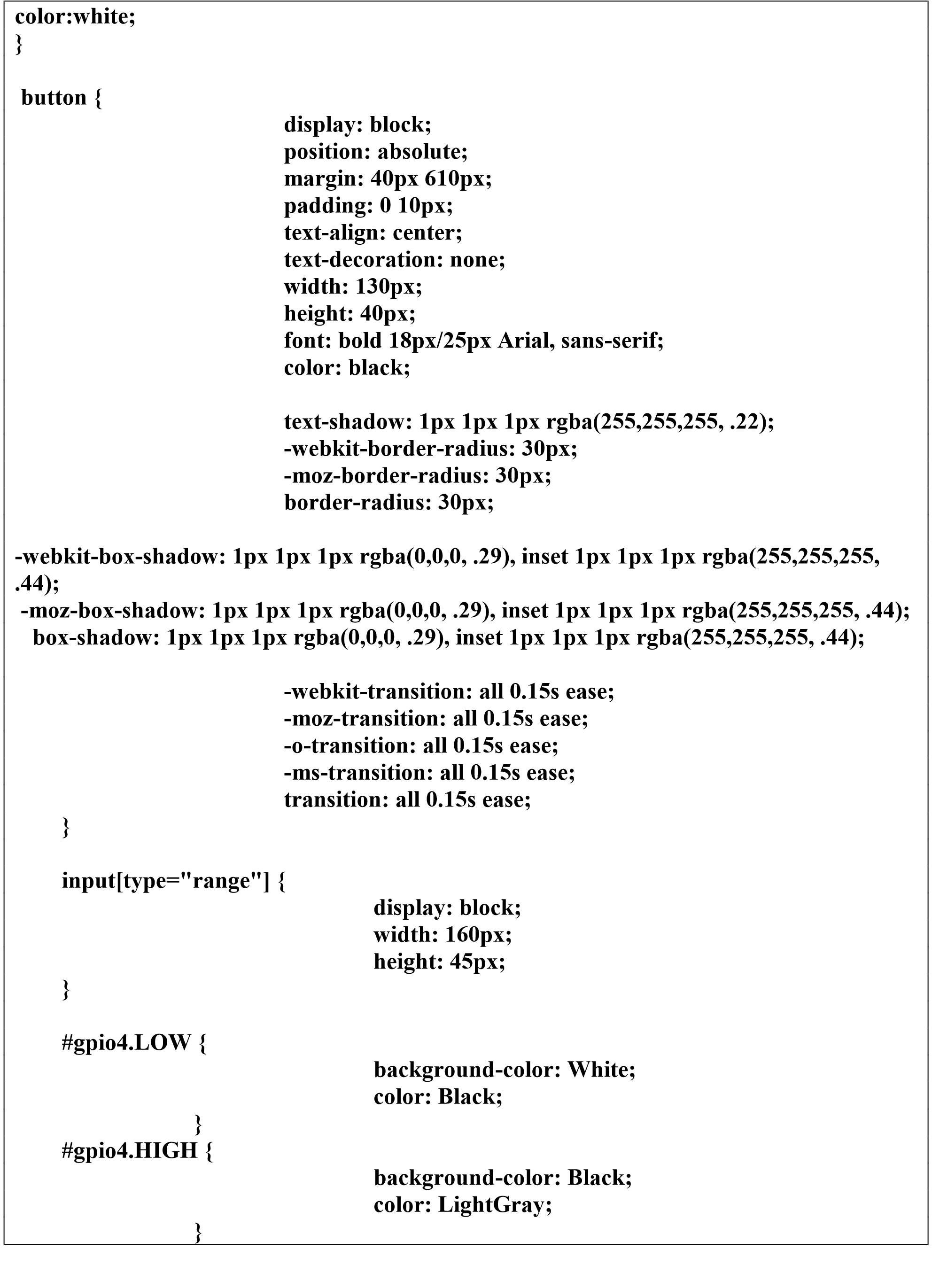
folder





## Smarthome.css file

|  |
| --- |
| **body {**  **background-color:#fff; background-repeat:no-repeat; background-position:center; background-size:cover;**  **font: bold 18px/25px Arial, sans-serif;**    **}**    **h1 {**  **font: bold 40px Arial, sans-serif; background-color:#000;** |



Create **index.html** file inside **html** folder

**<html>**

**<head>**

**<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">**

**<meta name="mobile-web-app-capable" content="yes">**

**<meta name="viewport" content = "height = device-height, width = device-width, user-scalable = no" /> <title>Smart Home</title>**

**<script type="text/javascript" src="/webiopi.js"></script>**

**<script type="text/javascript" src="/scripts/smarthome.js"></script>**

**<link rel="stylesheet" type="text/css" href="/styles/smarthome.css">**

**<link rel="shortcut icon" sizes="196x196" href="/img/roomlighton.jpeg" /> </head>**

**<body>**

**<h1 align=center>Web Based Controlled Home Automation using Raspberry Pi </h1>**

**<div id="content" align="center"></div>**

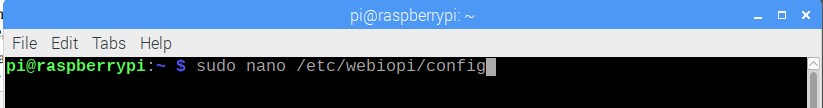
**<center><img src="/img/roomlighton.jpeg" height="500px" width="800px" id='pic1' /></center>**

**</body>**

**</html>**

### Step 5: WebIOPi Server Edits for Home Automation

We need to **edit the config file of the webiopi** service so it’s pointed to use data from our html folder instead of the config files that came with it.

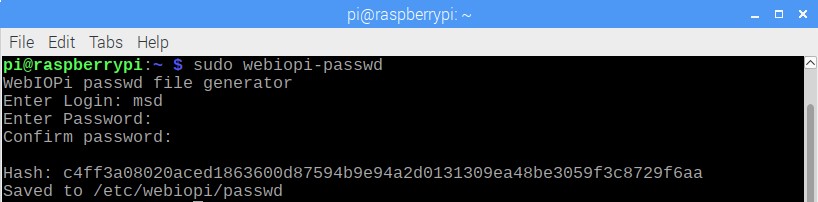


Under http section of the config file, comment out doc-root line and change the path to your project file



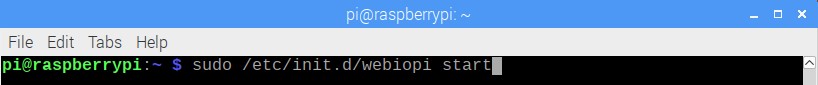
### Step 6: Change the username & password of the WebIOPi service

Note that you can change the password of the WebIOPi service using the command if you want. Or skip this step.

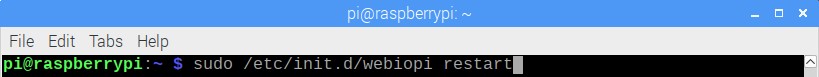


**Step 7: Run the WebIOPi service**

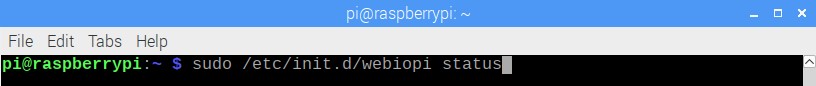
### Start the WebIOPi service



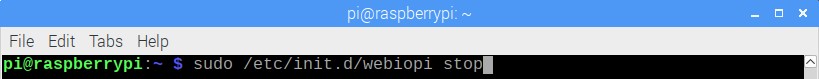
### Restart the WebIOPi service



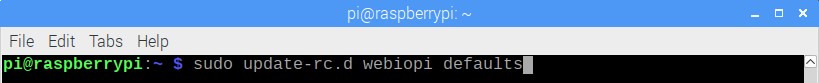
### Status the WebIOPi service



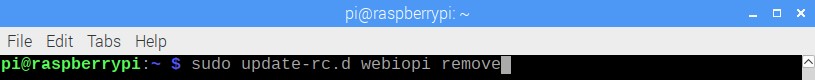
### Stop the WebIOPi service



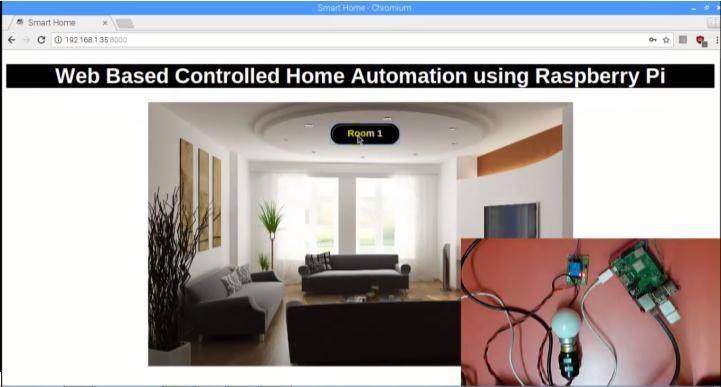
**To setup WebIOPi to run at boot, use:**

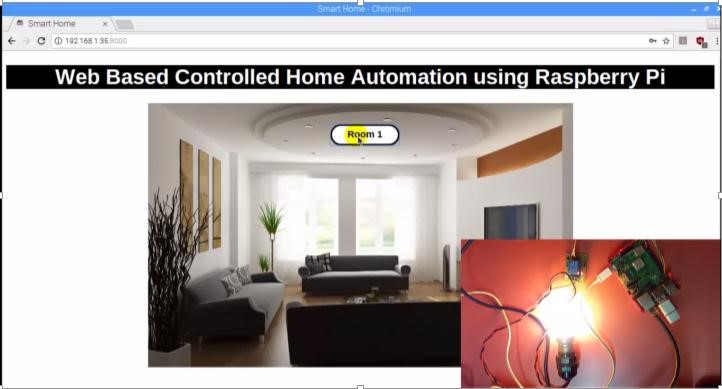


### To reverse and stop it from running at boot, use



**When you Click on button Room1 ,it switch on AC Lamp, again clicking on same button ,it will switch off AC Lamp.**





**Note : It is possible to open same URL** [**http://PI’s**](http://pi's/) **IP address:8000 on smart phone , tablet over local network.**