

ABSTRACT

WEBPAGE CONTROLLED SURVEILLANCE BOT

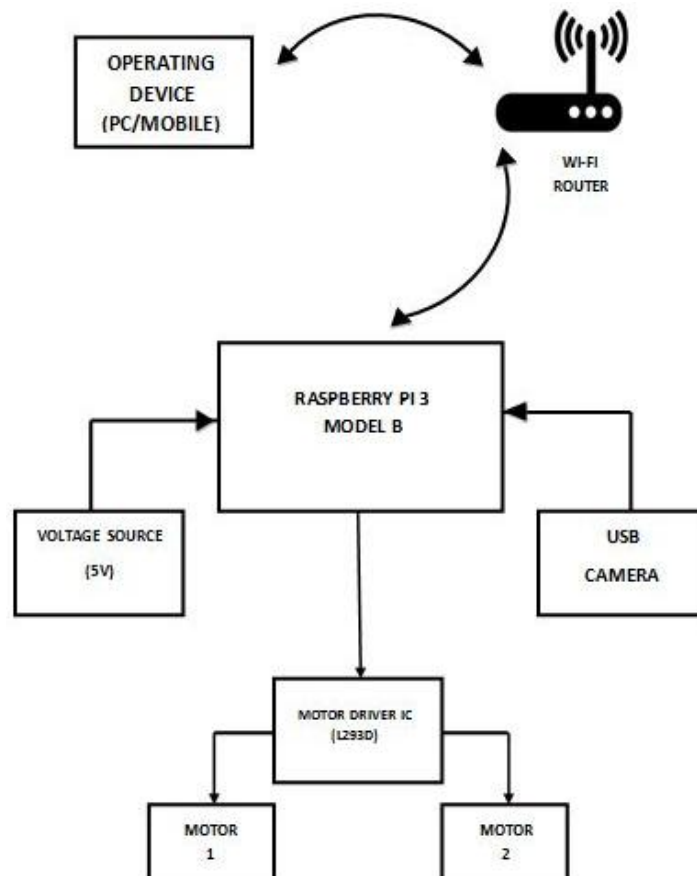
USING RASPBERRY PI

This mini-project is about designing and building a manually controlled surveillance robot using Raspberry pi. The main purpose of the robot is to roam around in a given environment while transmitting back real time data (video) to the manually hosted webpage, where program can be invoked through SSH connection (wireless real-time programming). This real time video data can then be used to move the robot around. The robot is compact and self contained with wireless transmission and reception of data using inbuilt Wi-Fi module which is controlled and viewed through webpage. Ports of the user IP address (Raspberry pi 's IP address) is used for hosting the webpage.

Keywords: Raspberry pi 3 model B, USB CAMERA, MOTION, Flask, motor driver IC (L293D).

PROJECT DESCRIPTION

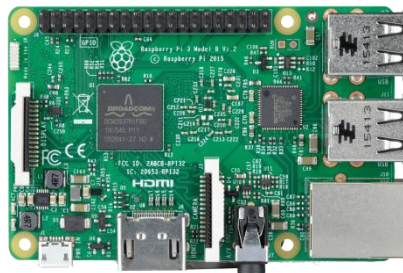
Day by day the application of wireless communication in automation field is increasing rapidly. In some applications human beings have been replaced by unmanned devices that will acquire data and relay the data back to the base. A single person can monitor and even interact with the ongoing work from a single base station. Wireless based surveillance robot is a prime concern in our day-to-day life. The approach to Wireless Network for various applications standardized nowadays. The embedded web server network consists of advanced processor ARM Cortex-A53 Raspberry Pi. It operates on RISC architecture. An embedded web server creates an easy way for monitoring & controlling the device which is at remote place. It has a web camera mounted over it, through which we will get live video feed and we can control and move this robot from a web browser over the internet. As it can be controlled using webpage, means it can also be controlled using webpage in Mobile / PC. We built a webpage in HTML which has Left, Right, Forward, Backward links, clicking on which we can move the robot in any direction. Here we used “**Motion**” for getting live Video feed from USB camera and used “**Flask**” for sending commands from webpage to Raspberry Pi using python to move the Robot.



BLOCK DIAGRAM

RASPBERRY PI 3 MODEL B :

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It has Quad-core 64-bit ARM Cortex A53 clocked at 1.2 GHz and 400MHz VideoCore.



RASPBERRY PI 3 MODEL B

USB CAMERA :

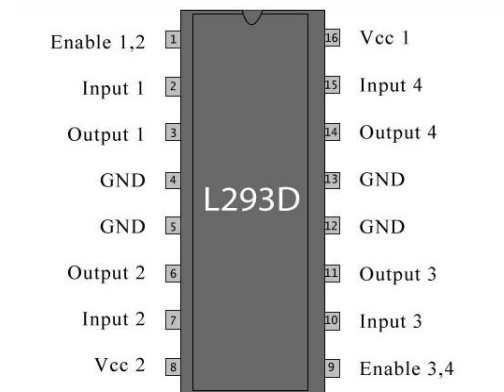
USB Cameras are imaging cameras that use USB 2.0 or USB 3.0 technology to transfer image data. USB Cameras are designed to easily interface with dedicated computer systems by using the same USB technology that is found on most computers. The accessibility of USB technology in computer systems as well as the 480 Mb/s transfer rate of USB 2.0 makes USB Cameras ideal for many imaging applications. An increasing selection of USB 3.0 Cameras is also available with data transfer rates of up to 5 Gb/s. For this project we are using Logitech Quick cam Notebook pro.



Logitech Quickcam for Notebook Pro

MOTOR DRIVER IC (ICL293D) :

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. Dual H-bridge *Motor Driver integrated circuit (IC)*. It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor.



Pin configuration of IC7293D

DC MOTORS :

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. 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



. DC MOTORS

POWER SUPPLY :

The Raspberry Pi 3 is powered by a +5.1V micro USB supply. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you connect to it. Typically, the model B uses between 700-1000mA depending on what peripherals are connected; the model A can use as little as 500mA with no peripherals attached. The maximum power the Raspberry Pi can use is 1 Amp. If you need to connect a USB device that will take the power requirements above 1 Amp, then you must connect it to an externally-powered USB hub. Here for raspberry pi 3 we use Li-on type battery of 10400 mAh of capacity.



USB POWER BANK UNIT

FLASK :

Flask is a micro framework for Python. This tool is Unicode based having built-in development server and debugger, integrated unit testing support for secure cookies and its easy to use. Here, we have created a web server using **Flask**, which provides a way to **send the commands from webpage to Raspberry Pi** to control the Robot over the network. Flask allows us to run our python scripts through a webpage and we can send & receive data from Raspberry Pi to web browser and vice versa.

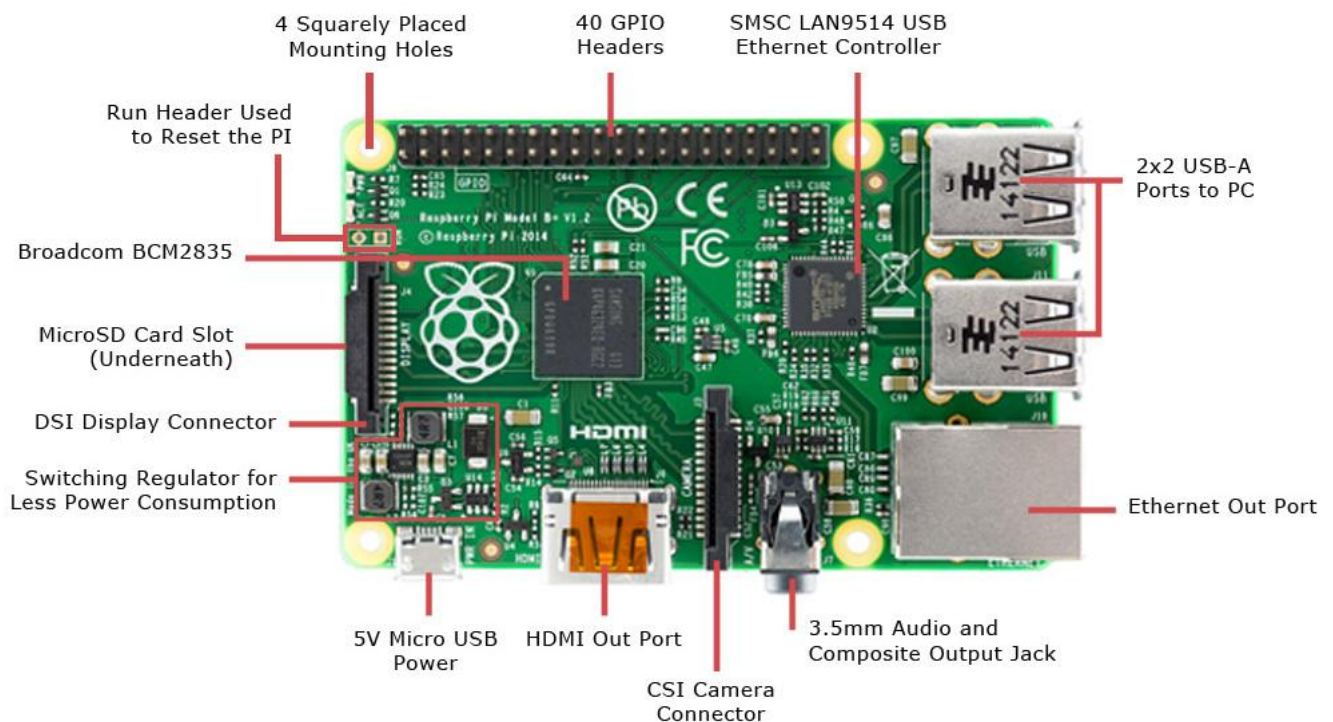
MOTION :

Motion (Surveillance Software) is free, open source motion detector CCTV software, developed for Linux. It detects the motion and start recording video of it. With 'Motion' installed in your Raspberry Pi, you can magically turn your Raspberry Pi into a Security Camera. It is used for getting live video feed, making timelapse videos and taking snapshots at regular interval. It records and saves the Video whenever it detects Motion or any disturbance in the view area. Live Video feed can be watched on the web browser by entering the IP address of Pi along with the port.

HARDWARE DESCRIPTION

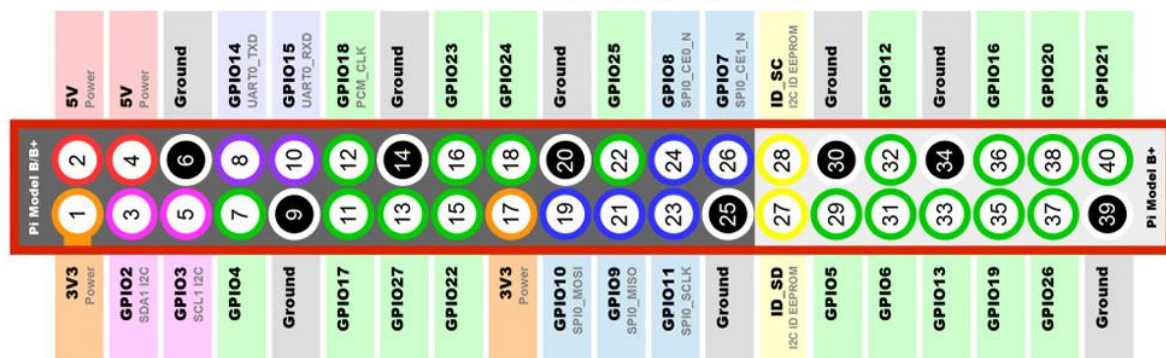
RASPBERRY PI 3 MODEL B :

The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processor, 10x faster than the first generation Raspberry Pi. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.



RASPBERRY PI 3 MODEL B

GPIO Pinout Diagram



SPECIFICATIONS

Processor	Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs with texture filtering and DMA infrastructure.
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT.
Dimensions	85 x 56 x 17mm.
Power	Micro USB socket 5V1, 2.5A.

CONNECTORS:

Ethernet	10/100 Base T Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4 Composite RCA (PAL and NTSC)
Audio Output	Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 Strip providing 27GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Push/pull Micro SDIO

Key Benefits

- Low cost
- Consistent board format
- 10x faster processing
- Added connectivity

Key Applications

- Low cost PC/tablet/laptop
- IoT applications
- Media centre
- Robotics
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Web camera
- Gaming
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)

FEATURES :

Wireless radio

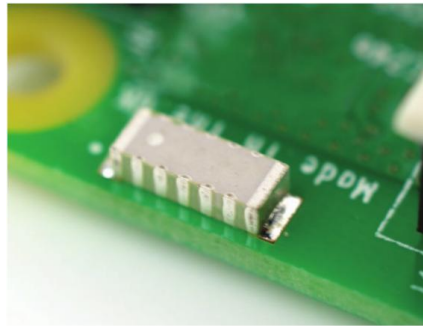
So small, its markings can only be properly seen through a microscope or magnifying glass, the Broadcom BCM43438 chip provides 2.4GHz 802.11n wireless LAN, Bluetooth Low Energy, and Bluetooth 4.1 Classic radio support. Cleverly built directly onto the board to keep costs down, rather than the more common fully qualified module approach, its only unused feature is a disconnected FM radio receiver.



Wireless radio

Antenna

There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board, in order to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.



Antenna

SoC

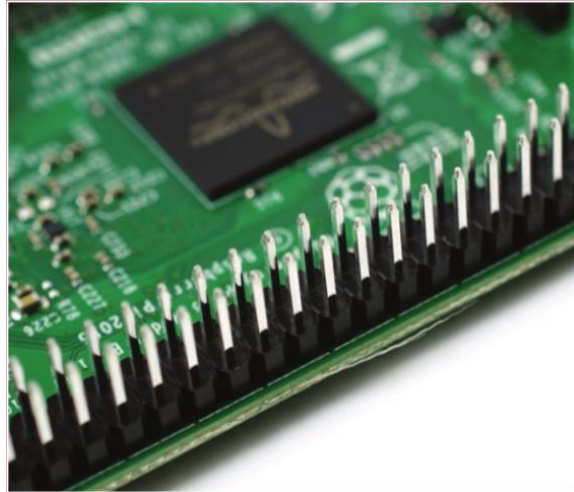
Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a Video Core IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.



SOC

GPIO

The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pis going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO's pins, but that's handled internally by the operating system.



GPIO PINS

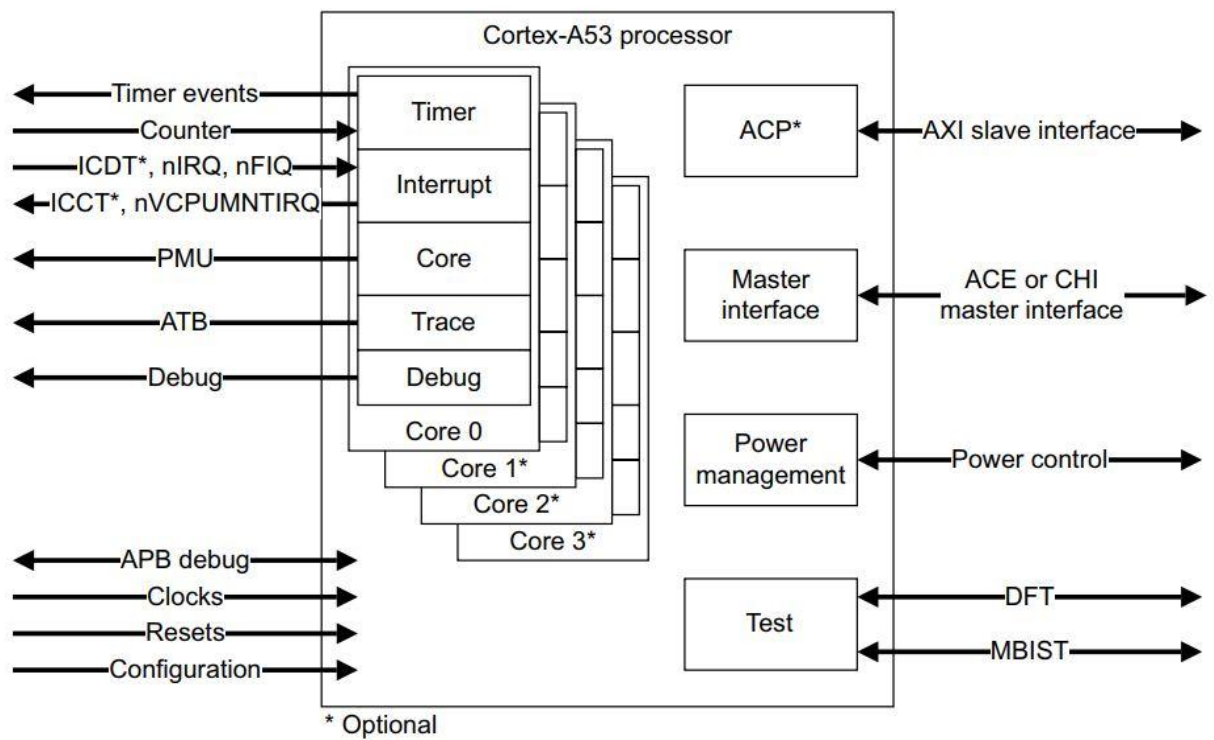
PROCESSOR

ARM CORTEX A53

The Cortex-A53 processor is a mid-range, low-power processor that implements the ARMv8-A architecture. The Cortex-A53 processor has one to four cores, each with an L1 memory system and a single shared L2 cache. The Cortex-A53 processor implements the ARMv8-A architecture.

This includes:

- Support for both AArch32 and AArch64 Execution states.
- Support for all exception levels, EL0, EL1, EL2, and EL3, in each execution state.
- The A32 instruction set, previously called the ARM instruction set.
- The T32 instruction set, previously called the Thumb instruction set.
- The A64 instruction set.



ARM CORTEX A53 ARCHITECTURE

INTERFACES

The Cortex-A53 processor has the following external interfaces:

- Memory interface that implements either an ACE or CHI interface.
- Optional Accelerator Coherency Port (ACP) that implements an AXI slave interface.
- Debug interface that implements an APB slave interface.
- Trace interface that implements an ATB interface.
- CTI. • Design for Test (DFT).
- Memory Built-In Self Test (MBIST).
- Q-channel, for power management.

SOFTWARE DESCRIPTION

FLASK

Its ability to combine modern web frameworks with hardware and electronics is one of the strengths of the Pi.

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.

In order to install Flask, you'll need to have pip installed. If you haven't already installed pip, it's easy to do:

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

After pip is installed, you can use it to install Flask and its dependencies:

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

To test the installation, create a new file called hello-flask.py with the code from below.

Python hello-flask.py

```
from flask import Flask
```

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

```
@app.route("/")
```

```
def hello():
```

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

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

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

Here's a breakdown of each line of code:

```
from flask import Flask
```

Load the Flask module into your Python script

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

Create a Flask object called app

```
@app.route("/")
```

```
def hello():
```

Run the code below this function when someone accesses the root URL of the server

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

Send the text "Hello World!" to the client's web browser

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

If this script was run directly from the command line

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

Have the server listen on port 80 and report any errors.

NOTE: Before you run the script, you need to know your Raspberry Pi's IP address. You can run `ifconfig` to find it. An alternative is to install `avahi-daemon` (run `sudo apt-get install avahi-daemon` from the command line). This lets you access the Pi on your local network through the address `http://raspberrypi.local`. If you're accessing the Raspberry Pi web server from a Windows machine, you may need to put Bonjour Services on it for this to work.

Now you're ready to run the server, which you'll have to do as root:

```
pi@raspberrypi ~ $ sudo python hello-flask.py
```

* Running on `http://0.0.0.0:80/`

* Restarting with reloader

From another computer on the same network as the Raspberry Pi, type your Raspberry Pi's IP address into a web browser. If your browser displays "Hello World!", you know you've got it configured correctly. You may also notice that a few lines appear in the terminal of the Raspberry Pi:

```
10.0.1.100 - - [19/Apr/2018 00:31:31] "GET / HTTP/1.1" 200 -
```

```
10.0.1.100 - - [19/Apr/2018 00:31:31] "GET /favicon.ico HTTP/1.1" 404 -
```

The first line shows that the web browser requested the root URL and our server returned HTTP status code 200 for "OK." The second line is a request that many web browsers send automatically to get a small icon called a *favicon* to display next to the URL in the browser's address bar. Our server doesn't have a `favicon.ico` file, so it returned HTTP status code 404 to indicate that the URL was not found.

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

If you've still got **hello-flask.py** running, press Control-C to kill it.

MOTION

Motion (Surveillance Software) is free, open source motion detector CCTV software, developed for Linux. It detects the motion and start recording video of it. With 'Motion' installed in your Raspberry Pi, you can magically **turn your Raspberry Pi into a Security Camera**. It is used for getting live video feed, making time lapse videos and taking snapshots at regular interval. It records and saves the Video whenever it detects Motion or any disturbance in the view area. Live Video feed can be watched on the web browser by entering the IP address of Pi along with the port.

Installing and Configuring 'Motion' for getting Video feed:

Step 1: First run the below command to **update the Raspbian OS** on Raspberry Pi:

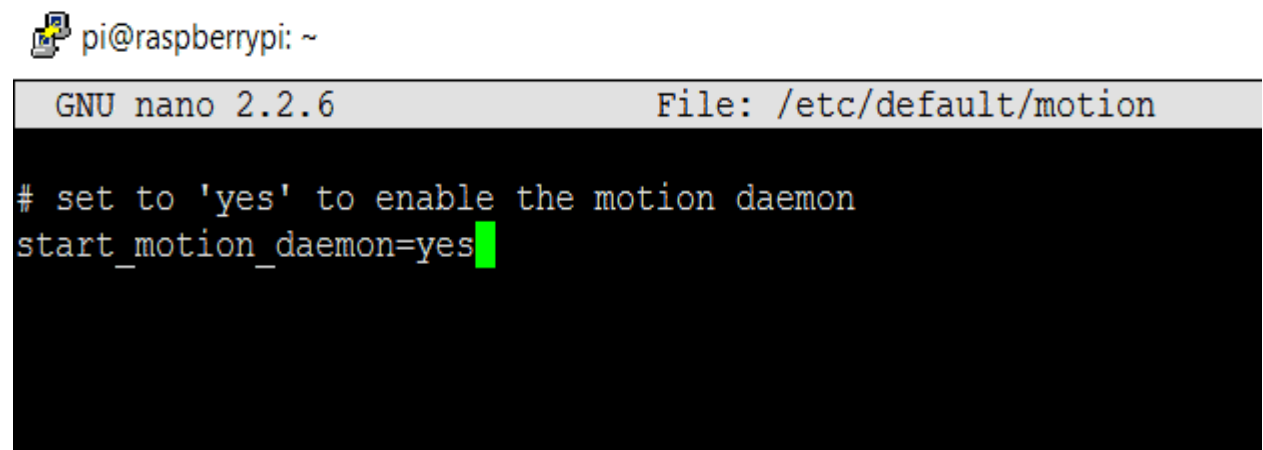
```
sudo apt-get update
```

Step 2: Then **install 'Motion' Library** by using below command:

```
sudo apt-get install motion
```

Step 3: Now **set Motion daemon to yes** by editing the file: `/etc/default/motion` so that it will be always running. Edit this file using 'nano' editor with 'sudo' like given below:

```
sudo nano /etc/default/motion
```



```
pi@raspberrypi: ~  
GNU nano 2.2.6 File: /etc/default/motion  
# set to 'yes' to enable the motion daemon  
start_motion_daemon=yes
```

Then save the file by pressing 'CTRL+X', then 'Y' and the Enter.

Step 4: Now we need to **set the permission for the Target Directory** (*/var/lib/motion/*), in which Motion saves all the Video recordings and picture files. We need to set ‘Motion’ as owner of this directory by issuing below command:

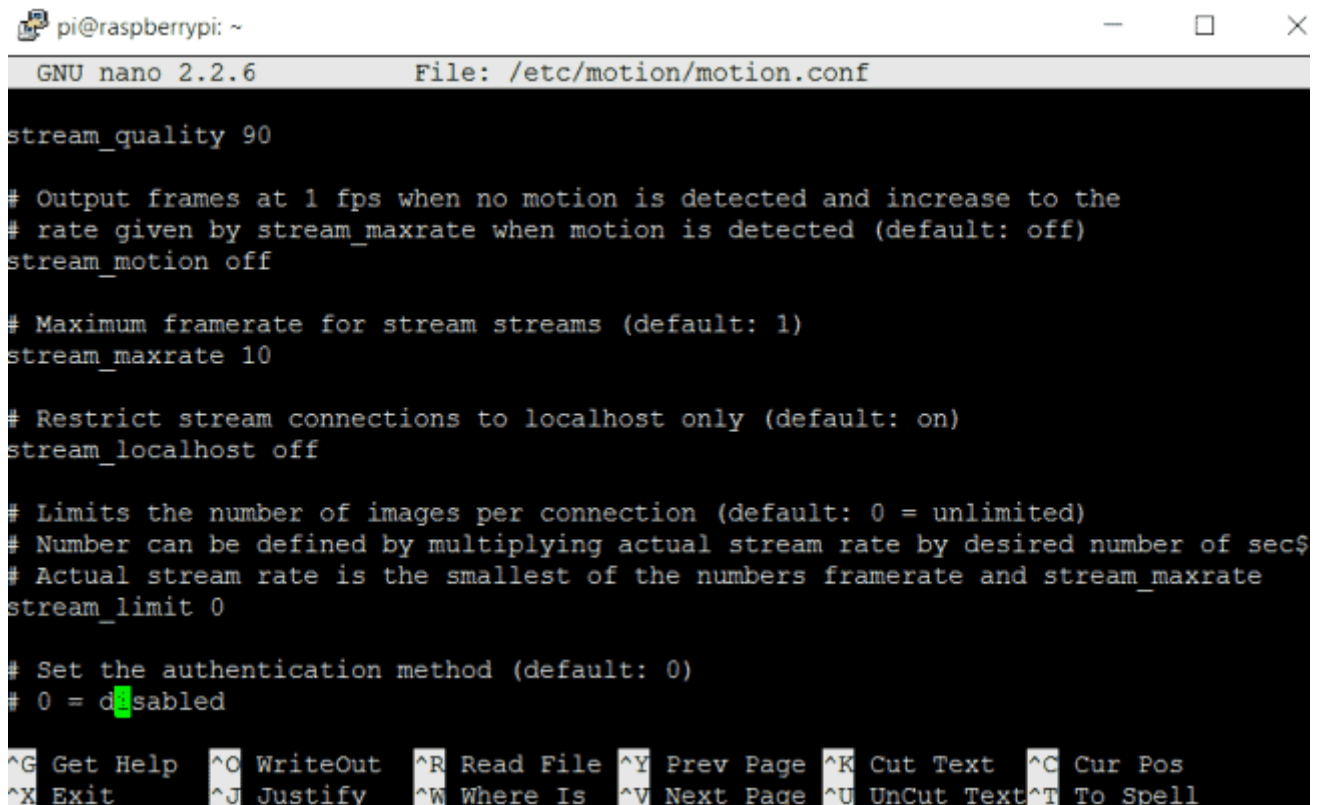
sudo nano /var/lib/motion/

This permission is necessary otherwise you will get error, when you check Motion service Status.

You can check service status by using this command: *sudo service motion status*

Step 5: Now we are almost done, only we need to change one config option in Motion configuration file (*/etc/motion/motion.conf*) which is *stream_localhost off*. We have to **turn off this local host streaming**, otherwise we will not be able to access the Video feed on our network and it will be only accessible from the Raspberry Pi itself. To doing so, edit the Motion Configuration file with ‘nano’ editor and turn it off, like shown below:

sudo nano /etc/motion/motion.conf



```
pi@raspberrypi: ~
GNU nano 2.2.6      File: /etc/motion/motion.conf

stream_quality 90

# Output frames at 1 fps when no motion is detected and increase to the
# rate given by stream_maxrate when motion is detected (default: off)
stream_motion off

# Maximum framerate for stream streams (default: 1)
stream_maxrate 10

# Restrict stream connections to localhost only (default: on)
stream_localhost off

# Limits the number of images per connection (default: 0 = unlimited)
# Number can be defined by multiplying actual stream rate by desired number of sec$
# Actual stream rate is the smallest of the numbers framerate and stream_maxrate
stream_limit 0

# Set the authentication method (default: 0)
# 0 = disabled

^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
```

Now we are done and ready to get our live feed from the USB web camera connected to Pi. Just start (or restart) the Motion service using below command and **open your Raspberry Pi's IP, with port 8081**, in your browser (like 192.168.43.199:8081). In this project we have embed this IP in our HTML code in *img src* tag.

sudo service motion restart

And you will see the live feed from your web camera. Here we have used a low cost USB web camera which worked smoothly with our Raspberry Pi, but you can further use a good quality camera for better resolution. As it will show in browser, you can use any device, to watch the feed, which supports web browser like Mobile, tablet etc.

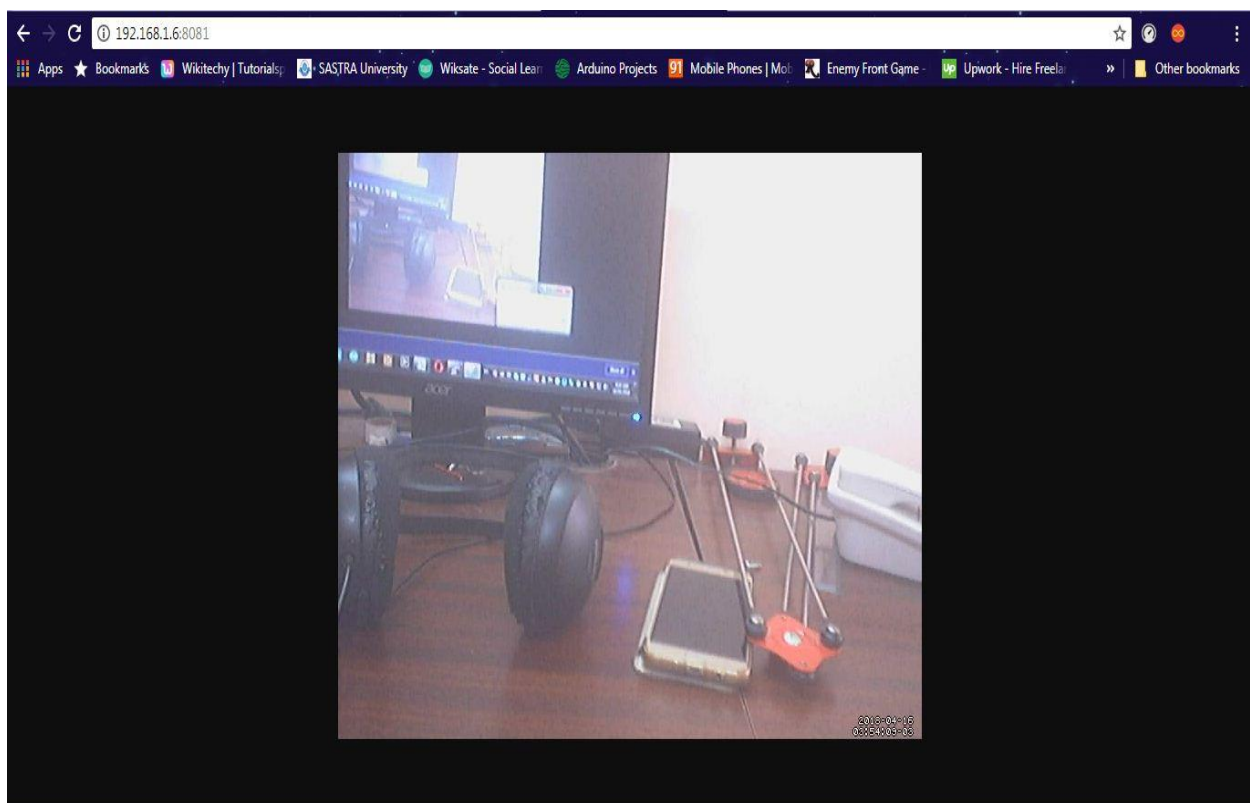
Try rebooting the Raspberry Pi as a troubleshooting step when necessary:

sudo reboot

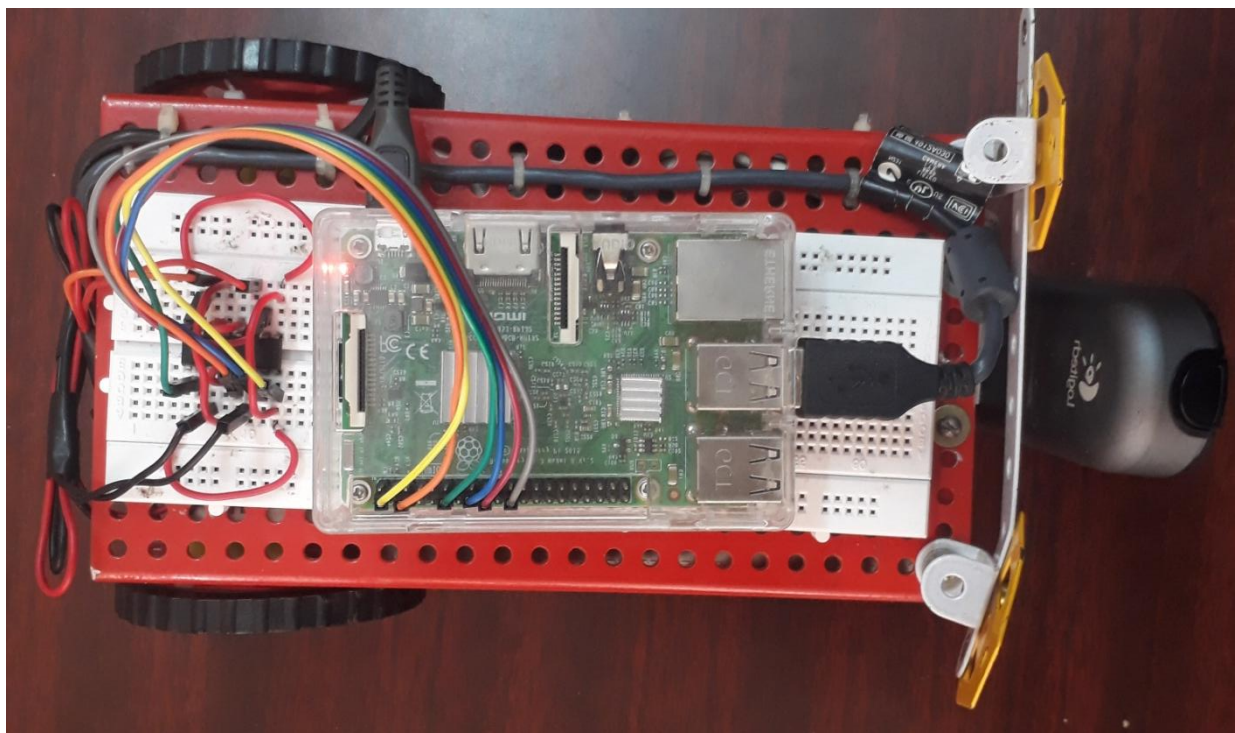
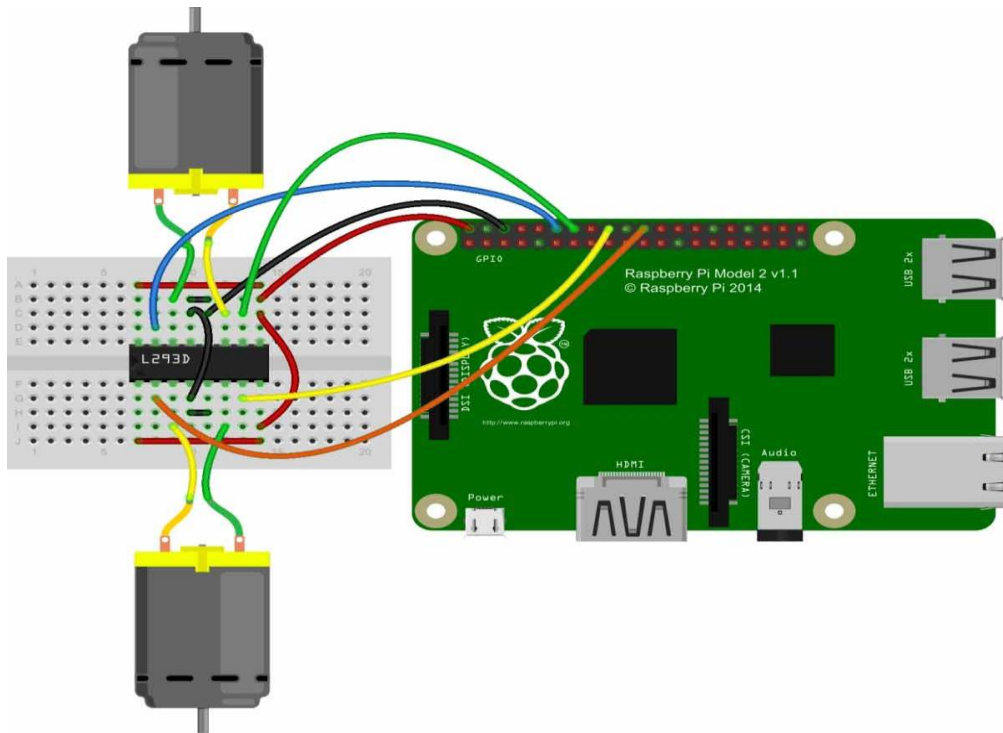
This is all about **using Motion for our Surveillance Robot**.

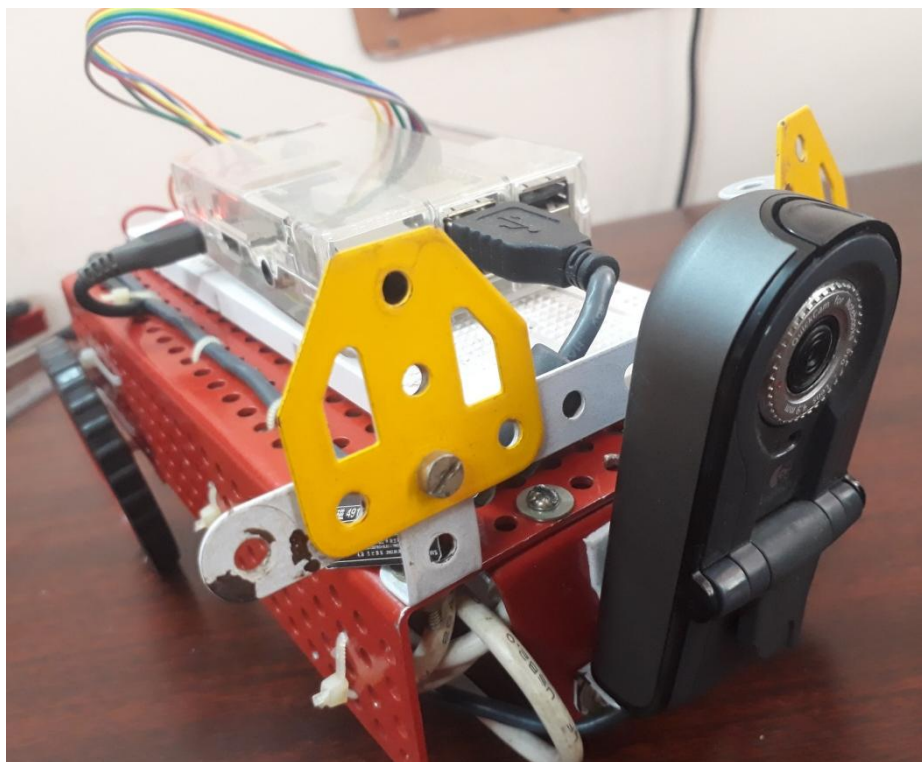
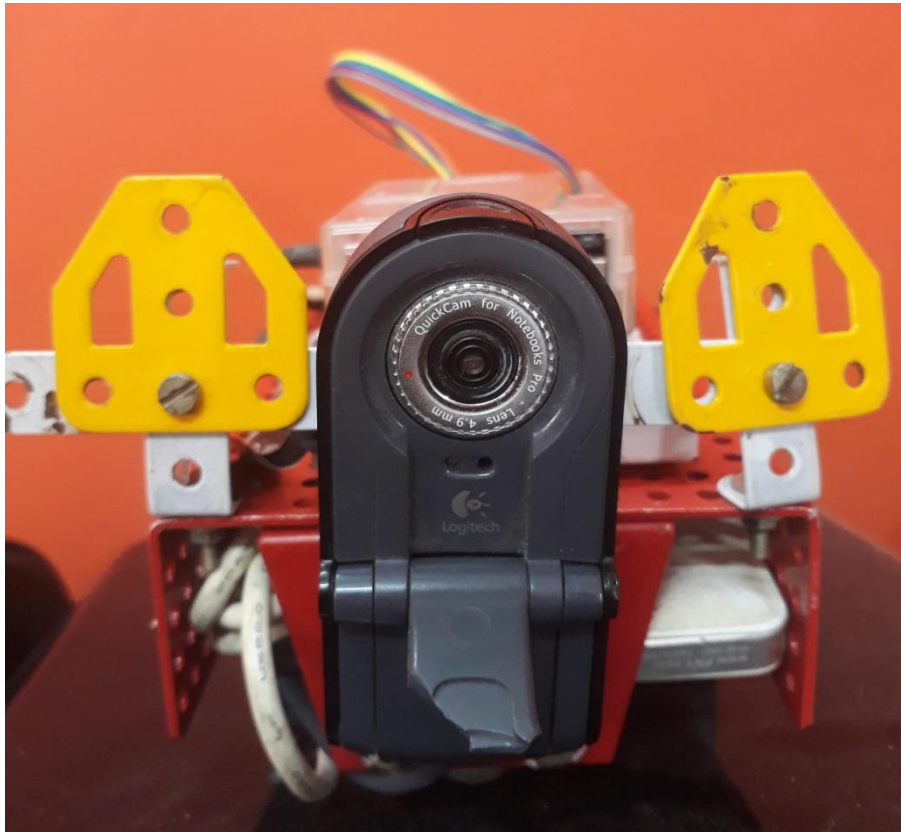
Note: If you are Raspberry Pi model below the version 3, then you may need a Wi-Fi dongle to wirelessly connect raspberry Pi to router.

Go to browser and type 192.168. xxx . xxx :8081, the live feed of the camera connected to raspberry pi displayed.



CIRCUIT DIAGRAM:





HTML code for webpage:

A web page is created using HTML language for displaying control links (Left, Right, Forward, backward) to move the Robot from web browser. We have used **jQuery script to call the functions in our Python Program**. There are five functions in Python Code to move the Robot Left, Right, Forward, Backward and to stop it. These functions will be executed by clicking on the Control Links on webpage and motors will move depending on the link being clicked. Here we have written the code in such way that Robot will move in certain direction while **clicking and holding the link**, and as soon as we **release the mouse button Robot will stop**. Below is the HTML code for webpage including the jQuery:

[illegible]

```

$("#up").on("mousedown", function() {
    $.get('/up_side');
}).on('mouseup', function() {
    $.get('/stop');
});
$("#left").on("mousedown", function() {
    $.get('/left_side');
}).on('mouseup', function() {
    $.get('/stop');
});
$("#right").on("mousedown", function() {
    $.get('/right_side');
}).on('mouseup', function() {
    $.get('/stop');
});
});
</script>

</body>
</html>

```

Note:

- Change the IP address in the code to which the raspberry pi connected.
- Note down the file name saved with .html extension for further use in python code.

PYTHON CODE :

```
from flask import Flask
from flask import render_template, request
import RPi.GPIO as GPIO
import time

app = Flask(__name__)

m11=18
m12=23
m21=24
m22=25

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(m11, GPIO.OUT)
GPIO.setup(m12, GPIO.OUT)
GPIO.setup(m21, GPIO.OUT)
GPIO.setup(m22, GPIO.OUT)
GPIO.output(m11 , 0)
GPIO.output(m12 , 0)
GPIO.output(m21, 0)
GPIO.output(m22, 0)
print "DOne"

a=1
@app.route("/")
def index():
    return render_template(' xxx .html') //replace xxx with filename of html code

@app.route('/left_side')
def left_side():
    data1="LEFT"
    GPIO.output(m11 , 0)
    GPIO.output(m12 , 0)
    GPIO.output(m21 , 1)
    GPIO.output(m22 , 0)
    return 'true'

@app.route('/right_side')
def right_side():
    data1="RIGHT"
    GPIO.output(m11 , 1)
    GPIO.output(m12 , 0)
    GPIO.output(m21 , 0)
```

```

GPIO.output(m22 , 0)
return 'true'

@app.route('/up_side')
def up_side():
    data1="FORWARD"
    GPIO.output(m11 , 1)
    GPIO.output(m12 , 0)
    GPIO.output(m21 , 1)
    GPIO.output(m22 , 0)
    return 'true'

@app.route('/down_side')
def down_side():
    data1="BACK"
    GPIO.output(m11 , 0)
    GPIO.output(m12 , 1)
    GPIO.output(m21 , 0)
    GPIO.output(m22 , 1)
    return 'true'

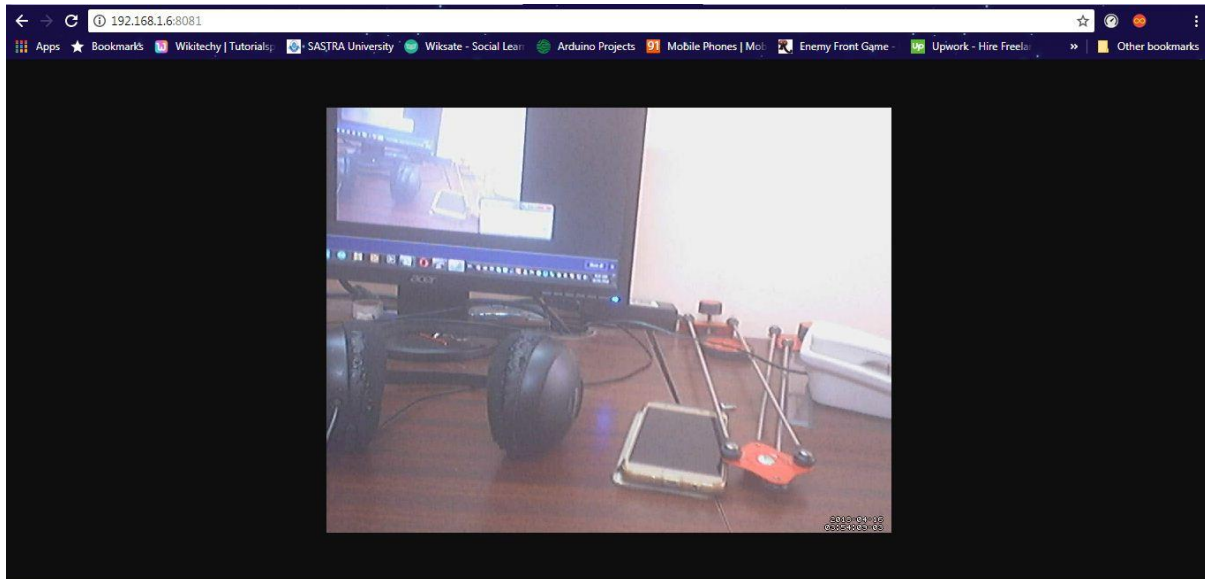
@app.route('/stop')
def stop():
    data1="STOP"
    GPIO.output(m11 , 0)
    GPIO.output(m12 , 0)
    GPIO.output(m21 , 0)
    GPIO.output(m22 , 0)
    return 'true'

if __name__ == "__main__":
    print "Start"
    app.run(host='192.168.xxx.xxx',port=5010)

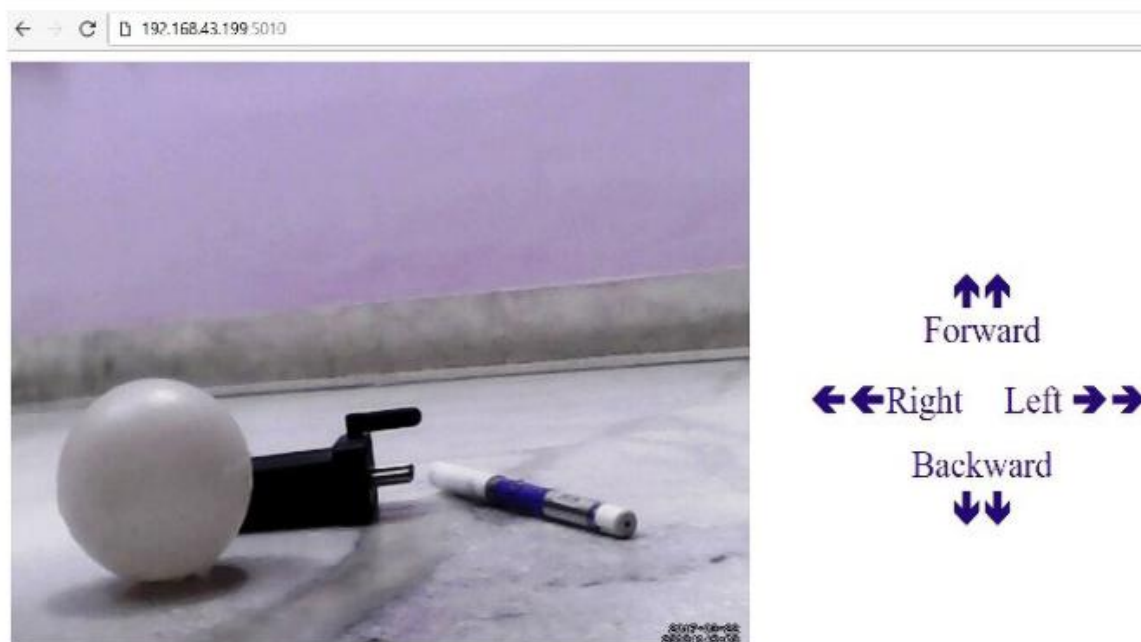
```


OUTPUT

LIVE VIDEO FEED OUTPUT IN PORT 8081



CONTROL INPUTS IN PORT 5010



CONCLUSION :

This system is designed and built to roam around in a given environment while transmitting back real time data (video) to the manually hosted webpage, where program can be invoked through SSH connection (wireless real-time programming). This real time video data can then be used to move the robot around.

BIBLIOGRAPHY

- (1) Chinmaya Kaundanya , Omkar Pathak , Akash Nalawade & Sanket Parode, “Smart Surveillance System using Raspberry Pi and Face Recognition”, IJARCCCE, Vol. 6, Issue 4 ISO 3297:2007 Certified, April 2017.
- (2) Sanjana Prasad, “Smart Surveillance Monitoring System Using Raspberry PI and PIR Sensor” ,IJCSIT International Journal of Computer Science and Information Technologies, Vol. 5, ISSN 0975-9646, 2014.
- (3) Python in raspberry pi
<https://www.raspberrypi.org/documentation/usage/python/>
- (4) Flask usage
<http://mattrichardson.com/Raspberry-Pi-Flask/>
- (5) Flask configuration
<https://projects.raspberrypi.org/en/projects/python-web-server-with-flask>
- (6) Putty configuration
<https://www.raspberrypi.org/documentation/remote-access/ssh/windows.md>
- (7) MOTION installation youtube link
<https://www.youtube.com/watch?v=3m29S2rbqBw&list=LLiKjtvdvWn9aOFeyE37oc8Q&index=111&t=0s>