# *Smart Doorlock Security System*

# Submitted by: Shreya Gupta(Institute of engineering and technology,Lucknow)

**Problem Statement:**

Design a smart security system that can take visitors request from a button at door and notify the owner about the visitor along with live stream from a HD Camera and take feedback from the owner using Interactive GUI to open/close the door in real time.

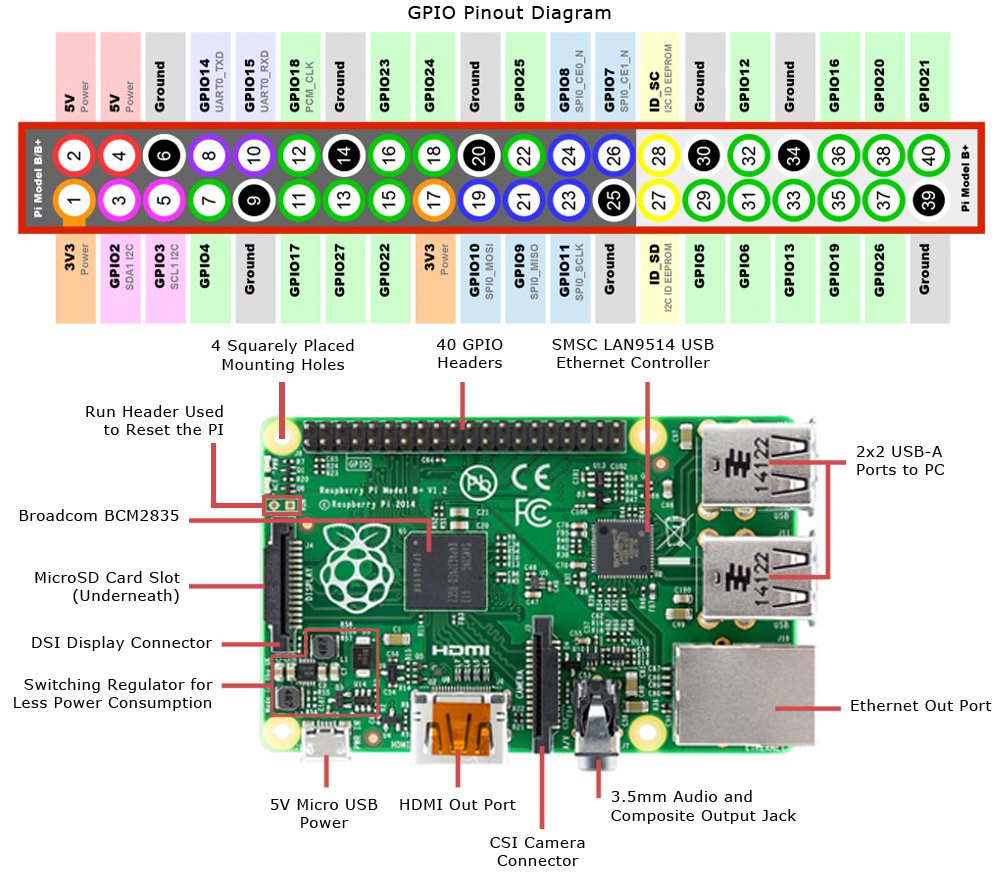
**Challenges Involved:**

1. Investigating and choosing appropriate hardware platform
2. Configuring it as a Camera streaming server
3. Running a cloud server as a medium between visitor and owner
4. Notifying the Owner about Visitors presence at the door
5. Live streaming their identity on his mobile/pc
6. Getting owners feedback to open/close the door
7. Get some voice message for the visitor from the user
8. Providing an Interactive Graphical User Interface for the owner in all the above Tasks
9. Making the system compatible on Local Area Network as well as The Internet
10. Making it practically feasible and economically viable

**OBJECTIVE 1:** Getting started with Raspberry pi and Interfacing a Web camera to it

Hardware Required: Raspberry pi board, HDMI cable,Power supply for RPi, webcamera (we are using carl zeiss tesser HD 1080p-logitech)

Pin diagram:



PROCEDURE:

(assuming that the rpi has raspbian stretch already installed follow the steps below.)

* Connect the hdmi cable from the rpi to the desktop screen and the power supply.
* After the raspbian os is launched open the terminal
* Write the following commands on terminal:
* sudo apt-get update (this command updates all the packages to the newest version)
* sudo apt-get upgrade
* Connect the webcamera to one of the usb ports of the raspberry
* To check the camera is working fine type the following commands on the terminal to install and use fswebcam. fswebcam is a simple tool for taking photos and video through a webcamera
* sudo apt-get install fswebcam
* cd /home/pi/Desktop
* fswebcam image.jpg
* the above steps will take a photo from the webcamera and save it on the desktop in a folder named image.jpg
* if you got an error in taking the photograph try the following steps:
* write “lsusb” in the terminal it will give a list of all the connected usb devices check if it shows camera on it
* if the camera is present then may be a motion file is running in the background that prevents you from taking the photo. To uninstall motion:

sudo apt-get remove motion

now try taking photo using fswebcam.

(NOTE: try commands “cat /etc/os-release” to check the version of raspberry pi installed

NOTE: if there is an installation problem in any file try installing the file in the root because you may not have the permissions to move or copy a any file. To work in the root type command “sudo su” or “sudo su -”. After competing installation type “exit” to return)

Note: To add a new terminal shortcut keys are ctrl+alt+T**OBJECTIVE 2** : To create a raspberry pi localhost and stream a video from raspberry pi to the local host(android and rpi on common network) and then forwarding it to web server.

PROCEDURE:

In the command prompt type in the command '**sudo apt-get install motion** ' to start the installation.

Now to make sure that the camera is correctly detected.

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

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

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

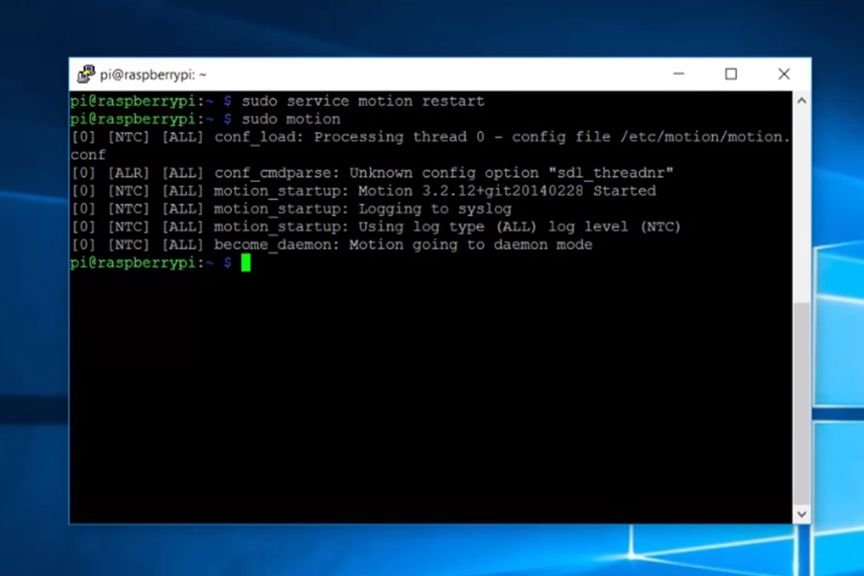
1. Make sure **'daemon' is ON**.
2. Set **'framerate'** anywhere in between **1000 to 1500**.
3. Keep '**Stream\_port'**to **8081.**
4. **'Stream\_quality'**should be**100.**
5. Change '**Stream\_localhost'**to **OFF.**
6. Change '**webcontrol\_localhost'**to **OFF.**
7. Set **'quality'** to **100.**
8. Set '**width'**& **'height'**to **640**& **480.**
9. Set **'post\_capture'**to **5.**
10. Press **ctrl + x** to exit. Type **y** to save and **enter**to confirm.

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

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

First of all your have to **restart** the motion software. To do it type in the command **'sudo service motion restart'**and press **enter.**

Again type in the command **'sudo motion'**and press **enter.**Now your server is ready



* now visit the web browser and type http://<yourip>:8081 to see the stream.
* Open your mobile and connect it to the same network(common wifi) as the raspberry and type http://<yourip>:8081 on the web browser to see the stream on mobile(you can also see the stream on ‘vlc’
* Hence a localhost for your webcam stream is created.

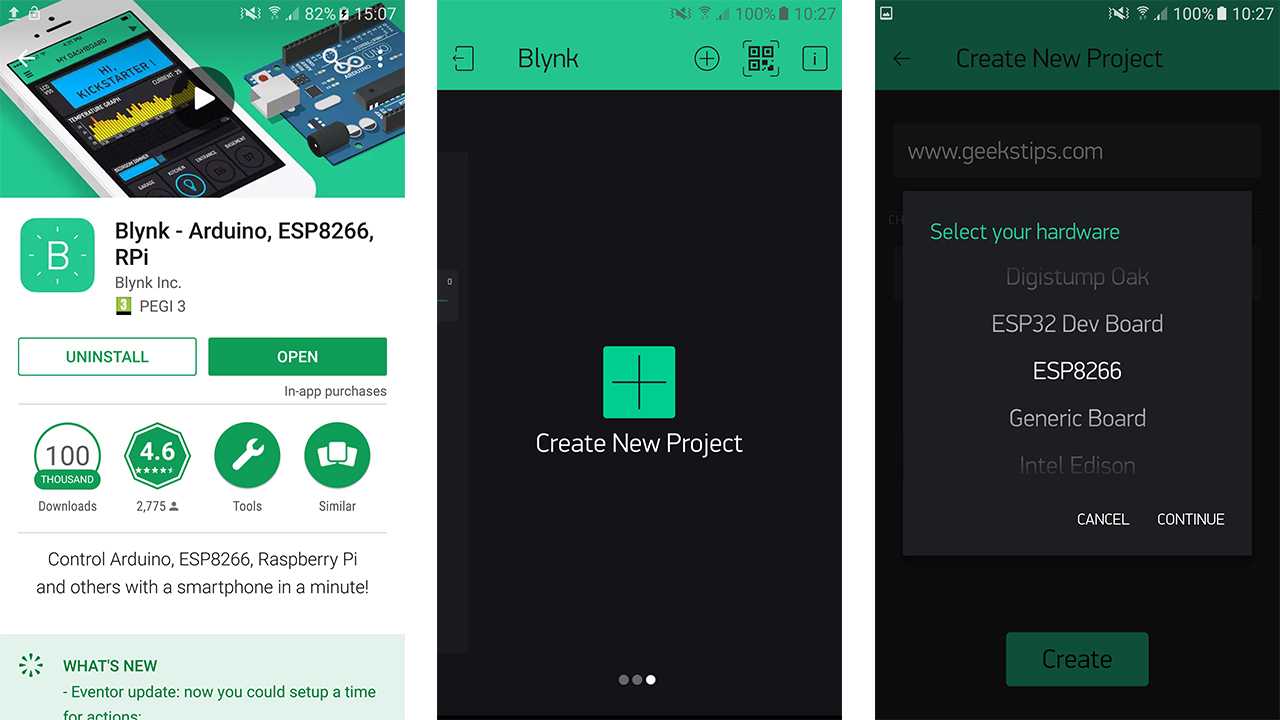


**OBJECTIVE 3**: Install nodejs and blynk for rpi

Procedure:

Download and install blynk app from play store on mobile. In order to connect Blynk App and your hardware, you need an Auth Token.

1. Create a new account in Blynk App.
2. Create a New Project. Then choose the board(raspberry pi 3 B) and connection you will use.
3. After the project was created, we will send you Auth Token over email.
4. Check your email inbox and find the **Auth Token**.



In the raspberry pi first of all, you need to install Node.js.

Before updating Node.js, please be sure to remove old versions. For this type the following commands in terminal

sudo apt-get purge node nodejs node.js -y  
sudo apt-get autoremove

## Automatic Node.js installation

Add repositories:

curl -sL <https://deb.nodesource.com/setup_6.x> | sudo -E bash -

Install Node.js:

sudo apt-get update && sudo apt-get upgrade  
sudo apt-get install build-essential nodejs -y

## Manual Node.js installation

Automatic install might not work for you, in this case you can perform manual installation.  
If uname -m gives you **armv6l** (on Raspberry Pi, usually), try this:

sudo su  
  
cd /opt  
  
wget <https://nodejs.org/dist/v6.9.5/node-v6.9.5-linux-armv6l.tar.gz> -O - | tar -xz  
mv node-v6.9.5-linux-armv6l nodejs  
  
apt-get update && apt-get upgrade  
apt-get install build-essential  
  
ln -s /opt/nodejs/bin/node /usr/bin/node  
ln -s /opt/nodejs/bin/node /usr/bin/nodejs  
ln -s /opt/nodejs/bin/npm /usr/bin/npm

(Note: Here if any of the above three commands give error replace –s with -sf)  
  
exit  
  
export PATH=$PATH:/opt/nodejs/bin/

## Check your Node.js and npm installation

pi@raspberrypi:/ $ node --version  
v6.9.5  
  
pi@raspberrypi:/ $ npm -v  
3.10.10

## Install Blynk globally

sudo npm install blynk-library -g  
sudo npm install onoff -g

Run default Blynk client (replace YourAuthToken):

export PATH=$PATH:/opt/nodejs/bin/  
unset NODE\_PATH  
blynk-client YourAuthToken

## Creating a new Node.js project with Blynk

Installing Blynk globally may not work or can be undesired.  
In this case, you need to create a new Node.js module with local Blynk library dependency.

mkdir my-awesome-project  
cd my-awesome-project  
npm init

It will prompt you for general information about your project and create a package.jsonfile (project description). Next, add Blynk to your project:

npm install blynk-library --save

You can also install onoff, if you want (allows direct pin operations):

npm install onoff --save

Now create your main script file index.js (just replace YourAuthToken):

var Blynk = require('blynk-library');  
  
var AUTH = 'YourAuthToken';  
  
var blynk = new Blynk.Blynk(AUTH);  
  
var v1 = new blynk.VirtualPin(1);  
var v9 = new blynk.VirtualPin(9);  
  
v1.on('write', function(param) {  
  console.log('V1:', param[0]);  
});  
  
v9.on('read', function() {  
  v9.write(new Date().getSeconds());  
});

 This is it. Run your project:

node index.js

You should see something like:

OnOff mode  
Connecting to: blynk-cloud.com 8441  
SSL authorization...  
Connected  
Authorized

## Image result for blynk rpi

## Troubleshooting

If you are trying to connect to Blynk cloud, and get an error like:

npm ERR! Error: SSL Error: CERT\_NOT\_YET\_VALID

you should use the date command to update current system time.

On the mobile open the project you created and click on the triangle on top right corner. If you don’t see a red mark on the button next to triangle means the blynk is connected now. If you see a red mark check your internet connection.

(NOTE: use “clear” command to clear the terminal at any moment

NOTE: use pip\_freeze to see the apps installed)

To port forward your raspberry pi and access it globally,

* First create an ngrok account
* Login to your account and download the zip file for linux(arm)

Or

In the terminal type **“sudo** **wget** [**https://dl.ngrok.com/ngrok\_2.0.19\_linux\_arm.zip**](https://dl.ngrok.com/ngrok_2.0.19_linux_arm.zip)”

* Unzip the file by typing “**sudo unzip ngrok\_2.0.19\_linux\_arm.zip**”.
* Next step is to embed the token in raspi. Type:

./ngrok authtoken <your\_auth\_token>

* To run ngrok:

./ngrok http 8081

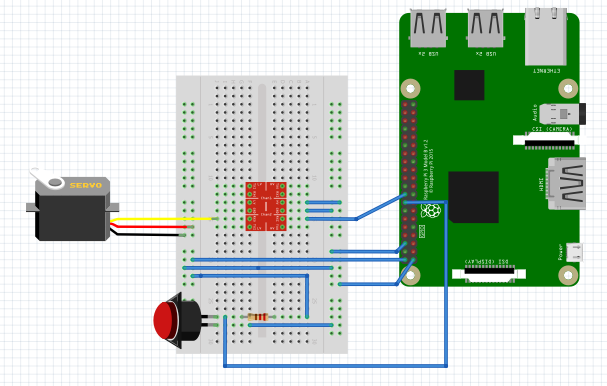
Now open web browser from a different network and type the web link generated on ngrok screen, you can see the stream.

**OBJECTIVE 4**: Create a basic python code for reading data from a button and and write a servo motor.

Material Required:

1. Raspberry pi 3 B
2. Servo motor(SG90 micro servo 9g)
3. Push button
4. Level converter
5. Breadboard
6. Jumper wires
7. Resistance(15 K)

Connections:



PROCEDURE:

Install rpi gpio for python by:

1. sudo apt-get install python-dev python-rpi.gpio

For writing a python script indentation is very important to separate blocks. Now open python IDE and type this code:

###############libraries###############

import RPi.GPIO as GPIO #library to control gpio pins of raspi

import time #library imported to give delay in the code

import os #used to give shell commands or terminal commands , the commands given with this will #be executed directly in the terminal

###############initialization##############

button=16 #define the pi no for button and servo

servo=18

GPIO.setmode(GPIO.BOARD) #there 2 modes to work with rpi GPIO.BCM mode and GPIO.BOARD mode

GPIO.setwarnings(False) #this done to ignore any warnings during code execution

GPIO.setup(button,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP) #to set the switch

GPIO.setup(servo,GPIO.OUT) #to set the servo motor

pwm=GPIO.PWM(servo,50) initiate pwm(pulse width modulation for servo control)

###########definition for angle servo###########

def setAngle(angle):

duty = angle / 18 + 2

GPIO.output(servo, True)

pwm.ChangeDutyCycle(duty)

time.sleep(1)

GPIO.output(servo, False)

pwm.ChangeDutyCycle(0)

################main program##############

while(1): #to run code for infinite time

if GPIO.input(button)==0: #if doorbell is pressed

print("button pressed")

os.system("sudo service motion start") #terminal commands for webcam to activate

os.system("sudo motion") #optional this statement is used to check whether the webam is #initialized or not

else:

print("button released")

os.system("sudo service motion stop")

time.sleep(1) #to stream video for one second

pwm.start(50) #start pwm

setAngle(0) #call function setangle and write angle 0

print("lock closed")

time.sleep(1) #to see motor turn

setAngle(180) #call function setangle and write angle 180

print("lock open")

time.sleep(1)

pwm.stop()

GPIO.cleanup()

The 2 modes mentioned above(bcm and board):



In this image if we count the pins from 1 to 40 in a sequential manner we are working in BOARD mode(usually easy to avoid wrong connections) and if we work with labeled gpio pins then we are working in BCM mode. But with nodejs we work with BCM mode only.

For more python codes reference watch videos by Paul MCWhorter <https://www.youtube.com/watch?v=J4fhE4Pp55E&list=PLGs0VKk2DiYypuwUUM2wxzcI9BJHK4Bfh>

**Objective 5:** Send a text message from android blynk and the raspberry pi should speak out the thext for you.

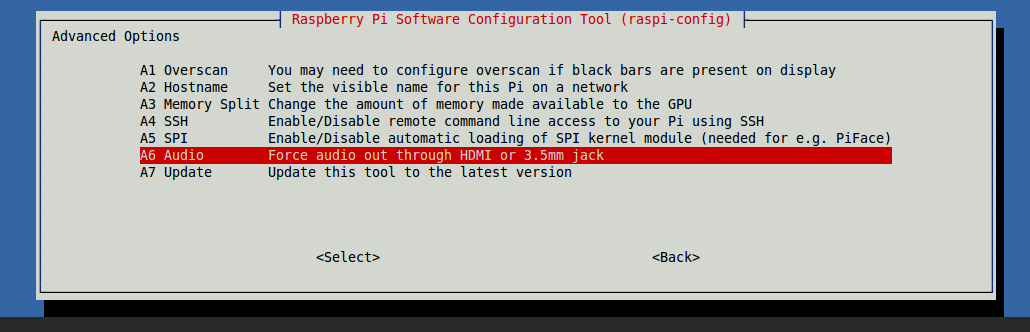
Material Required:

1. Raspberry pi
2. Speakers

Procedure:

First we will enable the audio port of raspberry pi and check the voice feature is working properly or not.

* Open the terminal and type ‘sudo raspi-config’
* goto ‘Advanced Options’
* select ‘Audio’
* among the options choose ‘auto’ and press enter



To check if the audio is working, type following commands on terminal:

* aplay /usr/share/sounds/alsa/\*

If you are able to hear the sounds like “***Front Center***”,”***Front Left***”, “***Front Right***” and so on, your sound is working!

Next, install eSpeak.  Run the following in terminal to install espeak:

* sudo apt-get install espeak

After eSpeak has been successfully installed on the Raspberry Pi, run the following command to test eSpeak:

* espeak "Text you wish to hear back" 2>/dev/null

Reference: https://www.dexterindustries.com/howto/make-your-raspberry-pi-speak/

**OBJECTIVE 6:**  Using blynk with python

PROCEDURE: Blynk server cannot be used directly by invoking blynk library. For this purpose we need to write a code in nodejs(javascript), or in c++ or by installing blynkapi for python. We will use blynkapi. You can alse download blynkapi library from and learn to use built in commands with python.

For nodejs coding refer to the following links for the project:

* <http://www.instructables.com/id/Blynk-JavaScript-in-20-minutes-Raspberry-Pi-Edison/>
* <https://github.com/fivdi/pigpio>
* <https://medium.freecodecamp.org/node-js-child-processes-everything-you-need-to-know-e69498fe970a> ,for child process i.e. if we want to execute terminal commands from our script itself(for webcam control) it is used. Exec() worked for me , you can try other methods.

to execute a nodejs program save the file by the extension .js and run it in the terminal by typing “node <project\_name>.js

For blynk api python installation follow these steps:

* <https://github.com/xandr2/blynkapi>
* If you donot have pip installed already follow these steps for installing pip:
  + - sudo su –
    - curl [https://bootstrap.pypa.io/get-pip.py -o get-pip.py](https://bootstrap.pypa.io/get-pip.py%20-o%20get-pip.py)
    - sudo python get-pip.py –force-reinstall
    - then install blynkapi

another is to use the source codes for blynkapi library without installing any additional library. We will be using this method for our task. Write the open the terminal and write the following codes in the terminal. For reference and understanding of the code visit ‘**https://blynkapi.docs.apiary.io/#**’

**Code (for python 2):**

###################libraries###################################

from subprocess import call

from urllib2 import Request, urlopen

import RPi.GPIO as GPIO

import time

import os

##########################setup##############################

button=16

servo=18

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(button,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(servo,GPIO.OUT)

cmd\_beg= 'espeak '

cmd\_end= ' 2>/dev/null' # To dump the std errors to /dev/null

pwm=GPIO.PWM(servo,50)

values = """

{

"body": "someone at the door"

}

"""

headers = {

'Content-Type': 'application/json'

}

###########definition for angle servo###########

def SetAngle(angle):

duty = angle / 18 + 2

GPIO.output(servo, True)

pwm.ChangeDutyCycle(duty)

time.sleep(1)

GPIO.output(servo, False)

pwm.ChangeDutyCycle(0)

pwm.start(50)

SetAngle(0)

print("Gate locked startng setup")

##################main###########################################

while(1):

if GPIO.input(button)==0:

print("button pressed")

requesta = Request('http://blynk-cloud.com/auth\_token/notify', data=values, headers=headers)

#replace auth\_token with your authentication token

response\_bodya = urlopen(requesta).read()

#print response\_bodya

os.system("sudo service motion start")

print("sending notifiiaction to user")

time.sleep(5)

else:

print("button released")

os.system("sudo service motion stop")

time.sleep(.01)

request = Request('http://blynk-cloud.com/auth\_token/get/V3')

response\_body = urlopen(request).read()

#print response\_body

if response\_body == '["12345"]':

SetAngle(180)

print("Password matched")

print("lock opened")

else:

SetAngle(0)

print("Password not matched")

print("lock closed")

time.sleep(.05)

request = Request('http://blynk-cloud.com/auth\_token/get/V1')

response\_body = urlopen(request).read()

print("message from user")

print response\_body

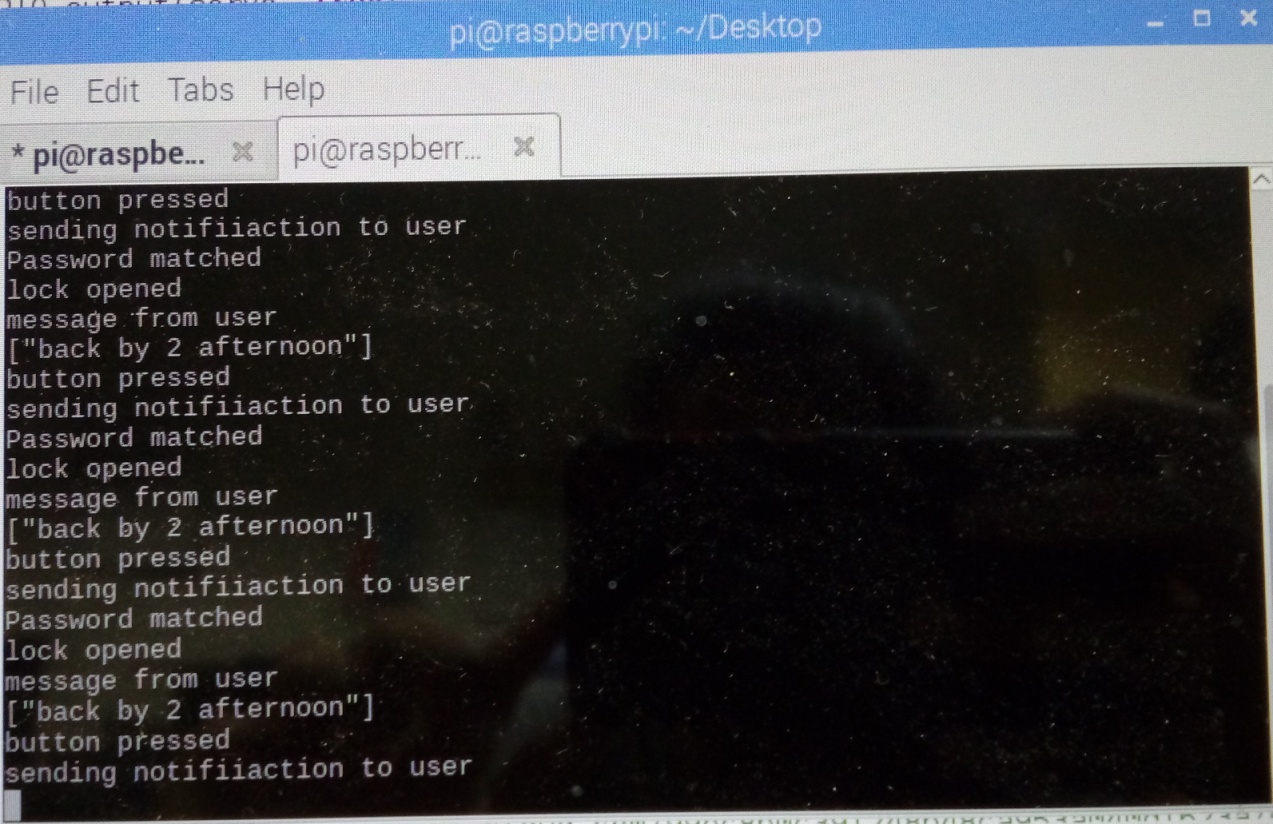
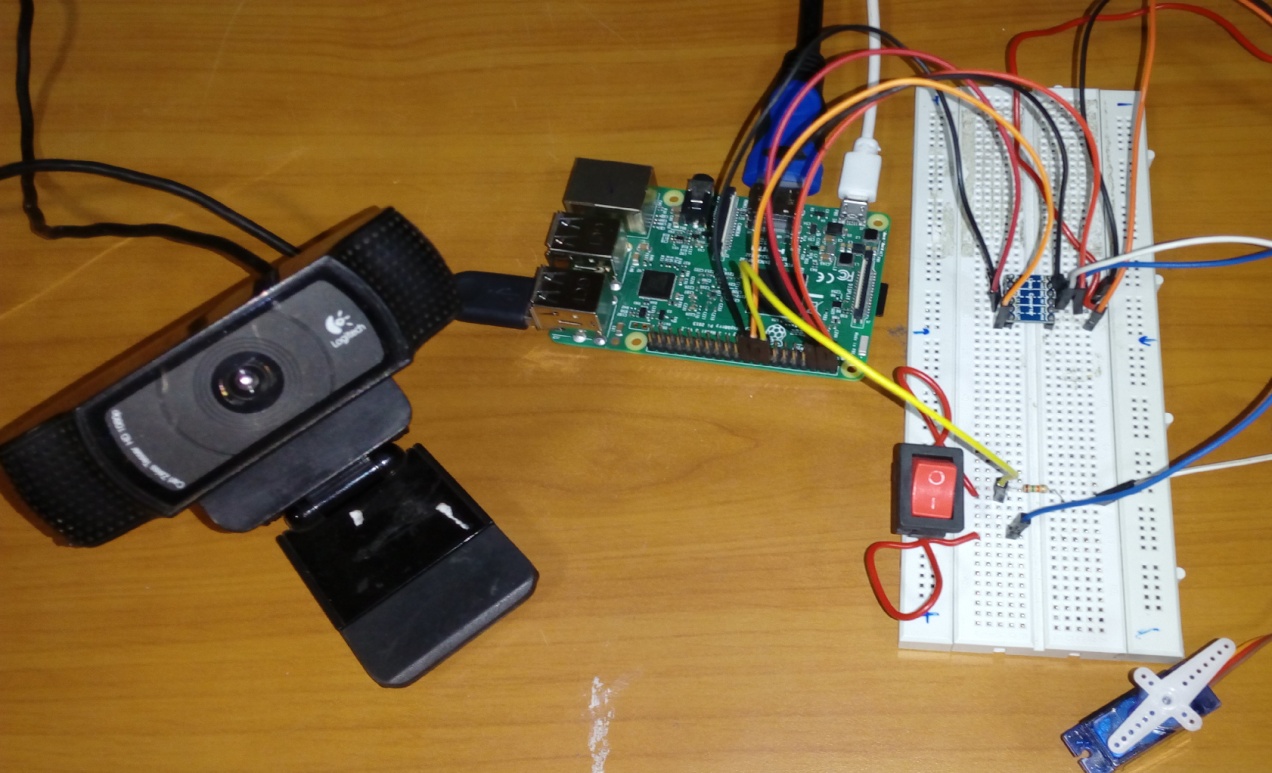
call([cmd\_beg+response\_body+cmd\_end], shell=True)

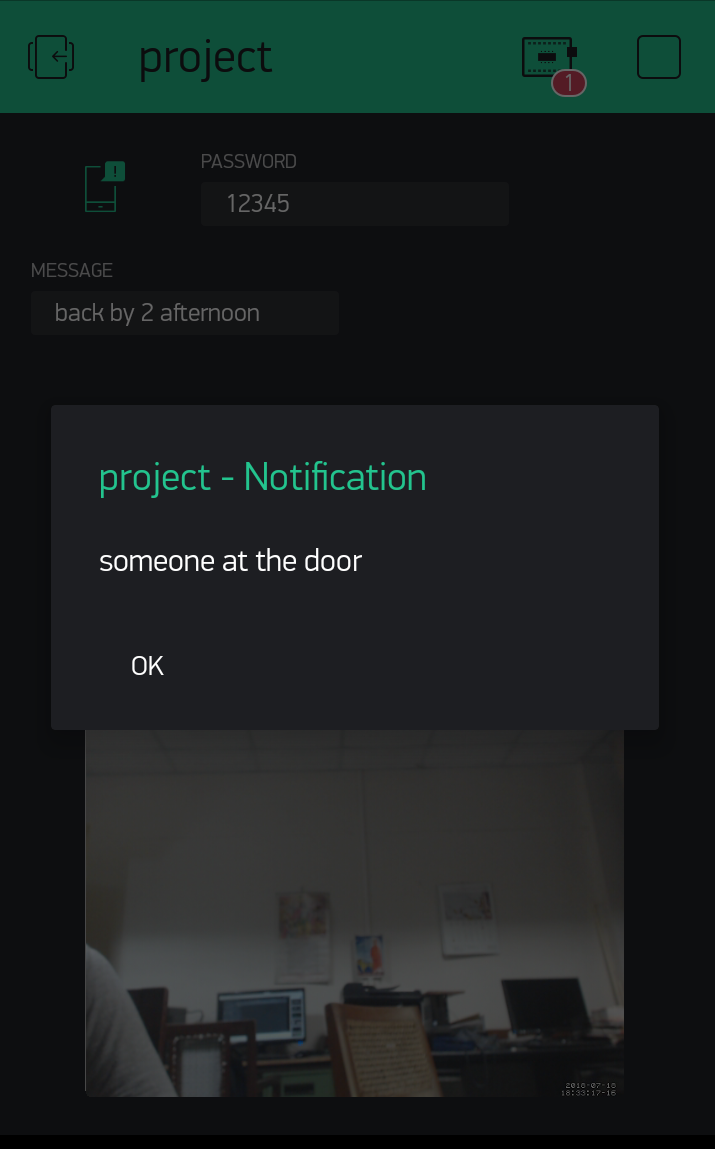
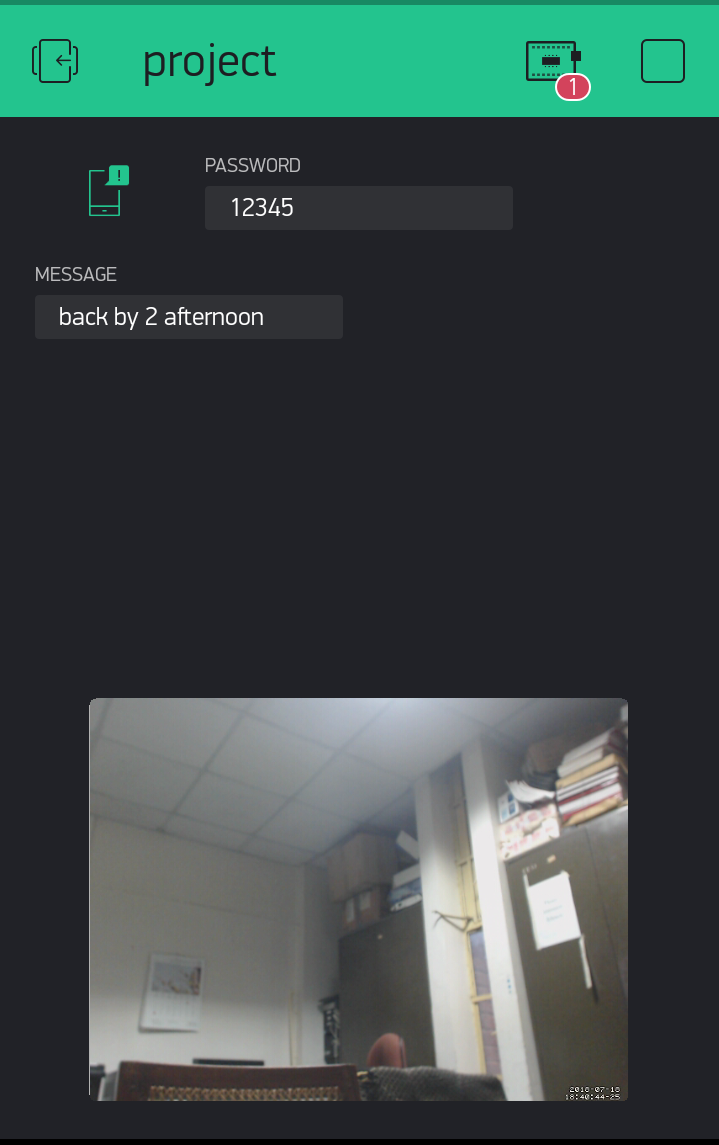
pwm.stop()

GPIO.cleanup()

NOTE: If on running the code you get an error ‘Bad request’ that means the project on the blynk app is not running so it is not able to send notifications. So run the blynk project on your phone.

***OUTPUT IMAGES:***





**CONSTRAINTS**

1.Hardware Limitations:

* Out of many possible hardware boards,most of them are incompatible for our tasks as they do not support interfacing HD cameras with their limited processing power (Eg:Arduino).Raspbery Pi served our purpose here.

2.Software compatibility:

* This project extensively involves downloading and installing library packages from the Terminal Interface.So one has to be careful about the software versions and their sources so at to make them compatible with the hardware used.
* Raspberry pi could not run the code on booting if the code runs on an infinite loop. Someone has to manually run the code everytime.

3.Network issues:

* Eventhough everything is setup properly,network delays and frequent connection loss may impact the streaming and user feedback.