Smart Door

- Beyond just opening and closing

Internet of Things is meant by a network of physical things sending and receiving or communicating information using Internet or other communication technologies and thus enabling the monitoring coordinating or controlling process across the internet or another data network

Problem statement:

The explosive growth of the "Internet of Things" is changing our world and allowing people to innovate new designs and products. The new generation is now full of smart people using smart technology. Smart devices make life of a person easy and updated. There are hundreds of goods available today that allow us to have power over the devices without human intervention. One of the major challenging issue in our society is security. So,in this project we used IoT to create smart security system. Using set of sensors and devices we monitor the visitors of the house and provided timely information to the owner through email. The owner can forward this information to anyone in case of any emergency. We also used openCV to train the machine to detect the faces of the visitors and unlock the the door for authenticated visitors. The system don't allow unauthenticated visitors to unlock the door. The owner can give authentication to anyone by adding their image to the database. Through this model we can avoid intruders and achieve the strongest security system.

Components Required:

- Raspberry Pi
- Pi Camera
- PIR Sensor
- Electrical door lock
- LED
- Bread Board
- Resistor (1k)
- Connecting wires
- Power supply

Circuit Description:

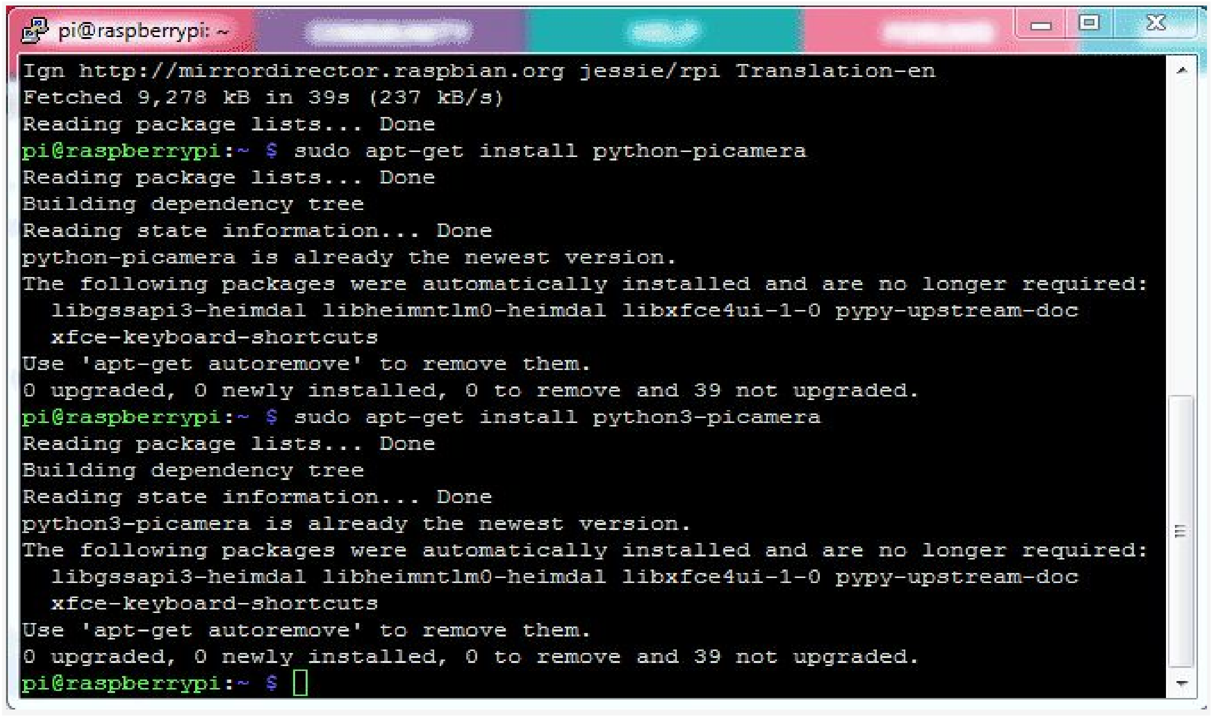
In this project we need to **connect Pi Camera module PIR sensor and electrical door lock to Raspberry Pi 3**. Pi Camera is connected at the camera slot of the Raspberry Pi and PIR is connected to GPIO pin 18. A LED is also connected to GPIO pin 17 through a 1k resistor.

Raspberry Pi Configuration and Programming Explanation:

We are using **Python language** here for the Program. Before coding, user needs to Installing & Configuring Raspbian Jessie OS in Pi

After successfully installing Raspbian OS on Raspberry Pi, we need to **install Pi camera library files** for run this project in Raspberry pi. To do this we need to follow given commands:

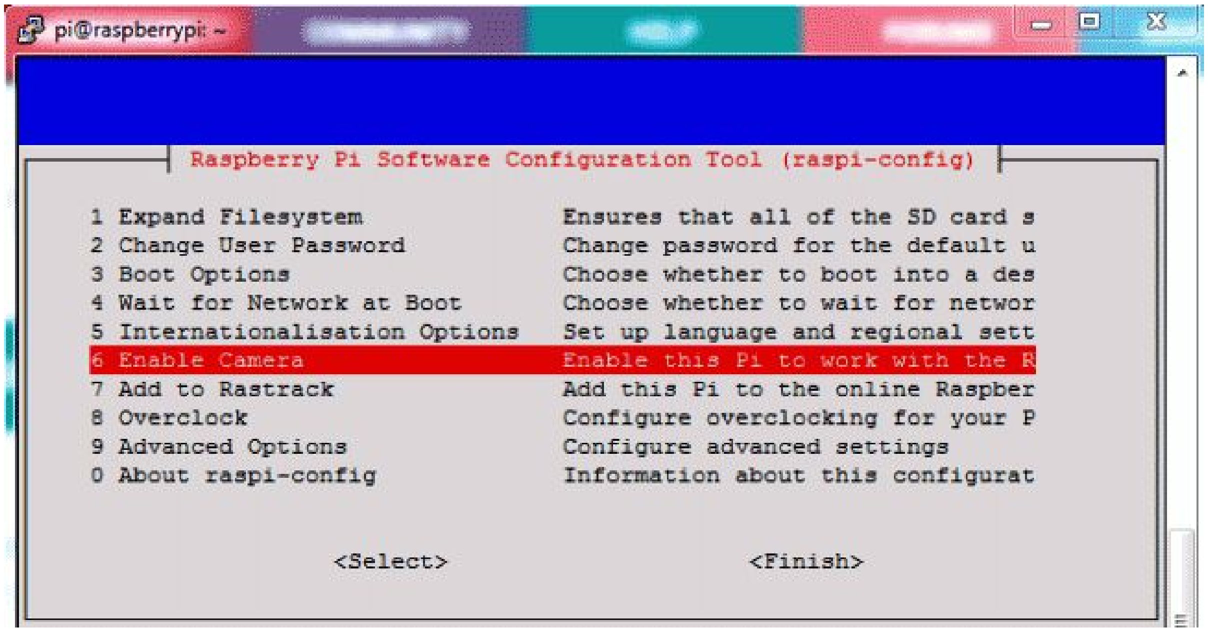
```
$ sudo apt-get install python-picamera
$ sudo apt-get install python3-picamera
```


```
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en
Fetched 9,278 kB in 39s (237 kB/s)
Reading package lists... Done
pi@raspberrypi:~ $ sudo apt-get install python-picamera
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-picamera is already the newest version.
The following packages were automatically installed and are no longer required:
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc
  xfce-keyboard-shortcuts
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
pi@raspberrypi:~ $ sudo apt-get install python3-picamera
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-picamera is already the newest version.
The following packages were automatically installed and are no longer required:
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc
  xfce-keyboard-shortcuts
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
pi@raspberrypi:~ $
```

After it, user needs to **enable Raspberry Pi Camera** by using Raspberry Pi Software Configuration Tool (raspi-config):

```
$ sudo raspi-config
```

Then select **Enable camera** and Enable it.

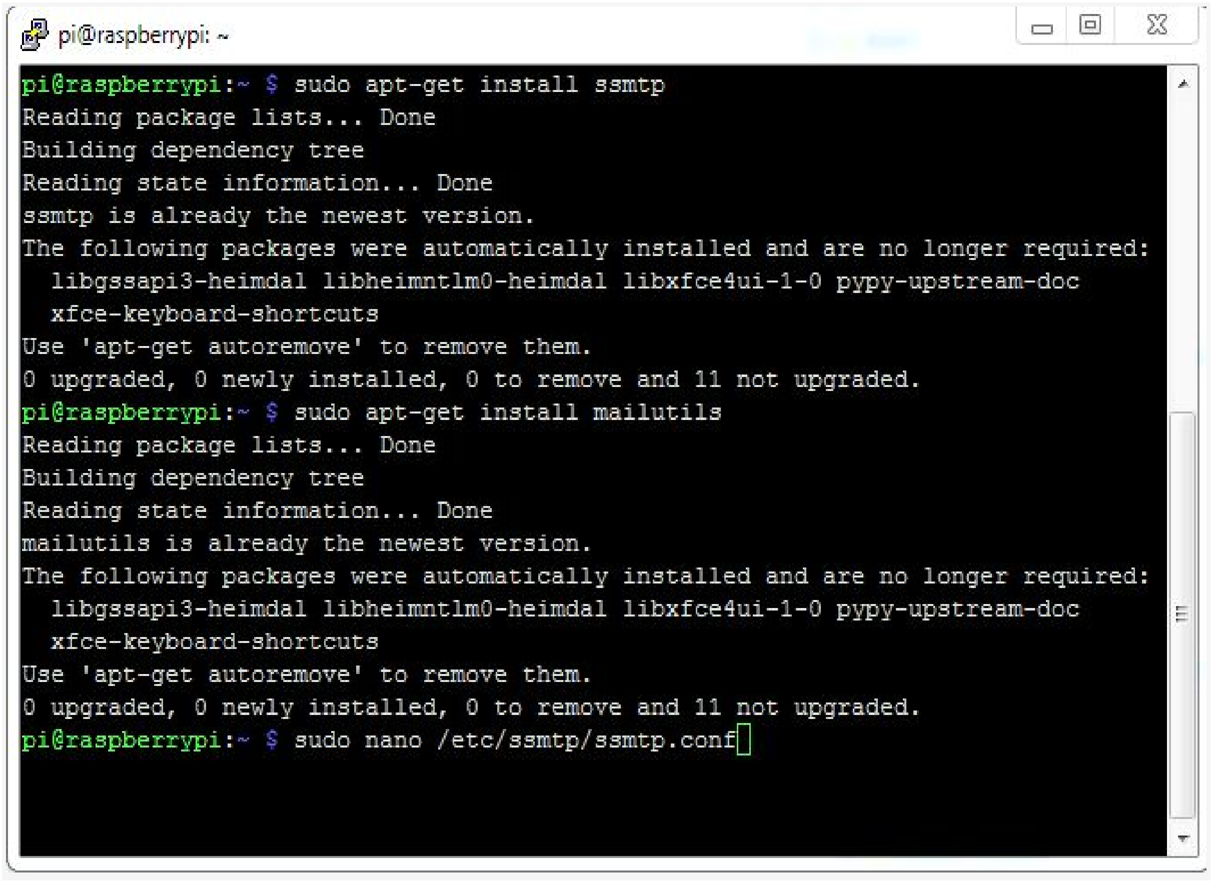


```
pi@raspberrypi: ~

    Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem              Ensures that all of the SD card s
2 Change User Password           Change password for the default u
3 Boot Options                   Choose whether to boot into a des
4 Wait for Network at Boot       Choose whether to wait for networ
5 Internationalisation Options   Set up language and regional sett
6 Enable Camera                  Enable this Pi to work with the R
7 Add to Rastrack                Add this Pi to the online Raspber
8 Overclock                      Configure overclocking for your P
9 Advanced Options               Configure advanced settings
0 About raspi-config             Information about this configurat

            <Select>                        <Finish>
```

Then user needs to **reboot Raspberry Pi**, by issuing *sudo reboot*, so that new setting can take. Now your Pi camera is ready to use.

Now after setting up the Pi Camera, we will install software for sending the mail. Here we are using *ssmtp* which is an easy and good solution for **sending mail using command line or using Python Script**. We need to install two Libraries for sending mails using SMTP:

```
sudo apt-get install ssmtp
sudo apt-get install mailutils
```

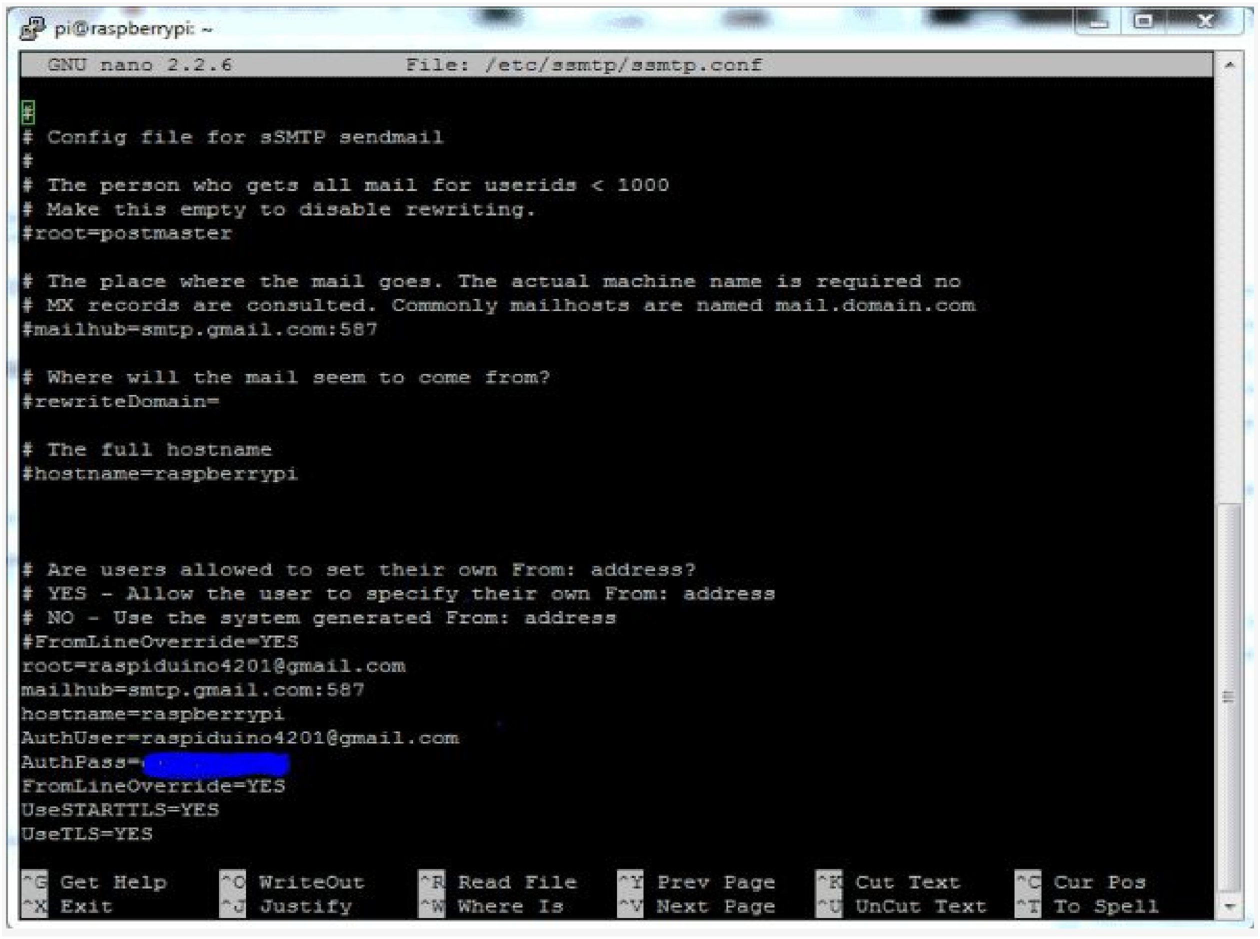


```
pi@raspberrypi:~ $ sudo apt-get install ssmtp
Reading package lists... Done
Building dependency tree
Reading state information... Done
ssmtp is already the newest version.
The following packages were automatically installed and are no longer required:
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc
  xfce-keyboard-shortcuts
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
pi@raspberrypi:~ $ sudo apt-get install mailutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
mailutils is already the newest version.
The following packages were automatically installed and are no longer required:
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc
  xfce-keyboard-shortcuts
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
pi@raspberrypi:~ $ sudo nano /etc/ssmtp/ssmtp.conf
```

After installing libraries, user needs to open *ssmtp.conf* file and edit this configuration file as shown in the Picture below and then save the file. To save and exit the file, Press ‘CTRL+x’, then ‘y’ and then press ‘enter’.

```
sudo nano /etc/ssmtp/ssmtp.conf
```

```
root=YourEmailAddress
mailhub=smtp.gmail.com:587
hostname=raspberrypi
AuthUser=YourEmailAddress
AuthPass=YourEmailPassword
FromLineOverride=YES
UseSTARTTLS=YES
UseTLS=YES
```



```
  GNU nano 2.2.6             File: /etc/ssmtp/ssmtp.conf

#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
#root=postmaster

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
#mailhub=smtp.gmail.com:587

# Where will the mail seem to come from?
#rewriteDomain=

# The full hostname
#hostname=raspberrypi



# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
root=raspiduino4201@gmail.com
mailhub=smtp.gmail.com:587
hostname=raspberrypi
AuthUser=raspiduino4201@gmail.com
AuthPass=[redacted]
FromLineOverride=YES
UseSTARTTLS=YES
UseTLS=YES

^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
```

We can also **test it by sending a test mail** by issuing below command, you shall get the mail on the mentioned email address if everything is working fine:

```
echo "Hello saddam" | mail -s "Testing..." saddam4201@gmail.com
```

The Python Program of this project plays a very important role to perform all the operations. First of all, we include required libraries for email, initialize variables and define pins for PIR, LED and other components. For sending simple email, smtplib is enough but if you want to send mail in cleaner way with subject line, attachment etc. then you need to use MIME (Multipurpose Internet Mail Extensions)

```
import RPi.GPIO as gpio
import picamera
import time
import smtplib
```

```
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders
from email.mime.image import MIMEImage
```

After it, we have initialized mail and define mail address and messages:

```
fromaddr = "raspiduino4201@gmail.com"
toaddr = "saddam4201@gmail.com"
mail = MIMEMultipart()
mail['From'] = fromaddr
mail['To'] = toaddr
mail['Subject'] = "Attachment"
body = "Please find the attachment"
```

Then we have created *def sendMail(data)* function for sending mail:

```
def sendMail(data):
    mail.attach(MIMEText(body, 'plain'))
    print data
    dat='%s.jpg'%data
    print dat
    attachment = open(dat, 'rb')
    image=MIMEImage(attachment.read())
    attachment.close()
    mail.attach(image)
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(fromaddr, "your password")
    text = mail.as_string()
    server.sendmail(fromaddr, toaddr, text)
    server.quit()
```

Function *def capture_image()* is created to **capture the image of intruder** with time and date.

```
def capture_image():
    data= time.strftime("%d_%b_%Y|%H:%M:%S")
    camera.start_preview()
    time.sleep(5)
    print data
    camera.capture('%s.jpg'%data)
    camera.stop_preview()
    time.sleep(1)
    sendMail(data)
```

Then we initialized the **Picamera** with some of its settings:

```
camera = picamera.PiCamera()
camera.rotation=180
camera.awb_mode= 'auto'
camera.brightness=55
```

And now in last, we have read PIR sensor output and when its goes high Raspberry Pi calls the *capture_image()* function to capture the image of intruder and send a alert message with the picture of intruder as an attachment. We have used *sendmail()* inside *capture_image()* function for sending the mail.

```
while 1:
    if gpio.input(pir)==1:
        gpio.output(led, HIGH)
        capture_image()
        while(gpio.input(pir)==1):
            time.sleep(1)

    else:
        gpio.output(led, LOW)
        time.sleep(0.01)
```

So this how this **Raspberry Pi Security System** works, you can also use Ultrasonic sensor or IR sensor to detect the presence of burglar or intruder. Further check the **Full Code**.

CODE:

```
import RPi.GPIO as GPIO
import time
import picamera
import smtplib
import face_recognition
from time import sleep

from email.mime.multipart import MIMEMultipart
#from email.MIMEText import MIMEText
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from email.mime.image import MIMEImage

fromaddr = "smartdoorgmrit789@gmail.com"    # change the email address
accordingly
toaddr = "vamsikrishnapapana@gmail.com"     # Destination mail address

mail = MIMEMultipart()

mail['From'] = fromaddr
mail['To'] = toaddr
mail['Subject'] = "Attachment"
body = "Please find the attachment"

class Motor(object):
    def __init__(self, pins):
        self.P1 = pins[0]
        self.P2 = pins[1]
        self.P3 = pins[2]
        self.P4 = pins[3]
        self.deg_per_step = 360.0 / 512  # for half-step drive (mode 3)
        print(self.deg_per_step)
        self.steps_per_rev = 512  # 4096
        self.step_angle = 0  # Assume the way it is pointing is zero
degrees
        print(self.deg_per_step, self.steps_per_rev, self.step_angle)
        self._T = 0.005
        for p in pins:
            GPIO.setup(p, GPIO.OUT)
            GPIO.output(p, 0)

    def move_to(self, angle):
        """Take the shortest route to a particular angle (degrees)."""
        # Make sure there is a 1:1 mapping between angle and stepper angle
        target_step_angle = int(angle / self.deg_per_step)
        steps = target_step_angle - self.step_angle
        steps = (steps % self.steps_per_rev)
        print(self.deg_per_step, target_step_angle, self.step_angle,
steps)
        if steps > self.steps_per_rev / 2:
            steps -= self.steps_per_rev
            self._move_acw(-steps)
        else:
            self._move_cw(steps)
        self.step_angle = target_step_angle

    def __clear(self):
        GPIO.output(self.P1, 0)
        GPIO.output(self.P2, 0)
        GPIO.output(self.P3, 0)
        GPIO.output(self.P4, 0)

    def _move_acw(self, big_steps):
        self.__clear()
        for i in range(big_steps):
            #print(i)
            GPIO.output(self.P3, 0)
            GPIO.output(self.P1, 1)
            sleep(self._T * 2)
            GPIO.output(self.P2, 0)
            GPIO.output(self.P4, 1)
            sleep(self._T * 2)
            GPIO.output(self.P1, 0)
            GPIO.output(self.P3, 1)
            sleep(self._T * 2)
            GPIO.output(self.P4, 0)
            GPIO.output(self.P2, 1)
            sleep(self._T * 2)

    def _move_cw(self, big_steps):
        self.__clear()
        for i in range(big_steps):
            GPIO.output(self.P4, 0)
            GPIO.output(self.P2, 1)
            sleep(self._T * 2)
            GPIO.output(self.P1, 0)
            GPIO.output(self.P3, 1)
            sleep(self._T * 2)
            GPIO.output(self.P2, 0)
            GPIO.output(self.P4, 1)
            sleep(self._T * 2)
            GPIO.output(self.P3, 0)
            GPIO.output(self.P1, 1)
            sleep(self._T * 2)
```

```
def sendMail(data):
    mail.attach(MIMEText(body, 'plain'))
    dat='%s.jpg'%data
    print (dat)
    attachment = open(dat, 'rb')
    image=MIMEImage(attachment.read())
    attachment.close()
    mail.attach(image)
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(fromaddr, "project@123")
    text = mail.as_string()
    print("Processing Image")
    known_image =
face_recognition.load_image_file("/home/pi/Documents/vamsi.jpeg")
    unknown_image =
face_recognition.load_image_file("/home/pi/Documents/img.jpg")
    biden_encoding = face_recognition.face_encodings(known_image)[0]
    unknown_encoding = face_recognition.face_encodings(unknown_image)[0]
    results = face_recognition.compare_faces([biden_encoding],
unknown_encoding)
    if results[0]==True:
        print("Welcome Home")
        GPIO.setmode(GPIO.BOARD)
        m = Motor([11,13,15,16])
        m.move_to(-90)
        sleep(1)
        m.move_to(0)
        sleep(1)
    else:
        print("Intruder...")
        server.sendmail(fromaddr, toaddr, text)
        server.quit()

def capture_image():
    data="img"
    camera.start_preview()
    time.sleep(5)
    camera.capture('%s.jpg'%data)
    camera.stop_preview()
    time.sleep(1)
    sendMail(data)

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.IN)         #Read output from PIR motion sensor
GPIO.setup(3, GPIO.OUT)         #LED output pin

GPIO.output(3, 0)
camera = picamera.PiCamera()
```

```
camera.rotation=0
camera.awb_mode= 'auto'
camera.brightness=50
while True:
  i=GPIO.input(18)
  print(i)
  if i==1:                 #When output from motion sensor is LOW
     print ("Motion Detected")
     GPIO.output(3, 1)
     capture_image()
     time.sleep(1)

    #Turn OFF LED
#     time.sleep(0.1)
  elif i==0:               #When output from motion sensor is HIGH
      print ("No Motion detected")
      GPIO.output(3, 0)
      time.sleep(0.01)#Turn ON LED
#      time.sleep(0.1)
```