Smart Doorbell System Using IOT and Raspberry pi case study

A report submitted in partial fulfillment of the requirements of

Internet Of Things(ISEC2)

In Seventh Semester By

1MS18IS412 Pranav Hegde

Under the guidance of

Ashwitha A
Associate Professor
Dept. of ISE, RIT

RAMAIAH
Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
RAMAIAH INSTITUTE OF TECHNOLOGY
(AUTONOMOUS INSTITUTE AFFILIATED TO VTU)

M. S. RAMAIAH NAGAR, M. S. R. I. T. POST,

BANGALORE – 560054

2019-2020

Introduction:

In this system, security that combines the functions of smart phone and home network system. It enables the users to monitor visitors in real-time, remotely via the IoT-based doorbell installed near the entrance door to a house. This system makes security as further autonomous by capturing the image automatically and processing the image for facial matching and uses mail communication to the server to confirm the intruder is known or unknown. And also make them as a known person by entering the OTP which is sent to the mail server.

Abstract:

Security has always been an important issue in the home or office. A remote home security system offers many more benefits apart from keeping home owners, and their property, safe from intruders.
The system is composed of the Doorbell interfaced with Raspberry pi, whoever press the doorbell, the
camera gets triggered and capture their face and it checking for their face with its database which already has registered faces, if it is an authorized person door will open, otherwise it sends an OTP with their photograph of the intruder to server mail.
Only when non authorized person entered that OTP, that face gets added to the authorized person’s database to open the
door. Smart doorbells allow home owners to receive alerts when a visitor is at the door, see who the guest is, and communicate with the visitor from a smart device. They greatly improve people’s life quality and contribute to the evolution of smart homes.

Step:1 - Purpose & Requirements

Purpose:

It enables the users to monitor visitors in real-time, remotely via the IoT-based doorbell camera.
These captured visuals are also processed for finding potentially harm causing objects.
This system makes security as further autonomous by capturing the image automatically and processing the image for facial matching and uses mail communication to the server to confirm the intruder is known or unknown.

Behavior:

The **system** is composed of a camera interfaced with Raspberry pi, whenever the doorbell is pressed, the camera gets triggered and capture their face and it verifies its database which already has registered faces. ... It enables the users to monitor visitors in real-time, remotely via the IoT-based doorbell camera.

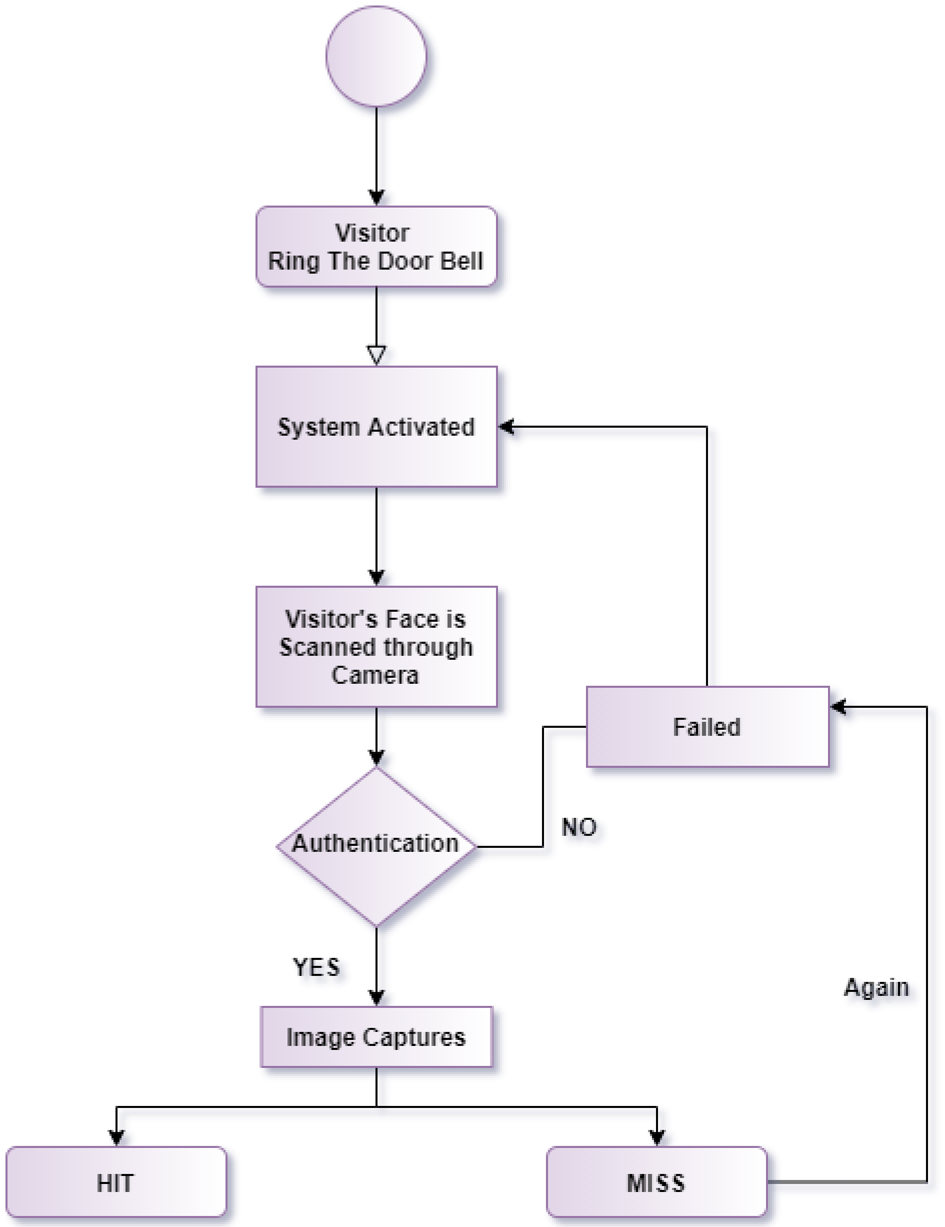
Data Analysis Requirement :

The system should perform local analysis of the data And Data Should Taken as the Person Details and His Authentication.

Security Requirement :

The system should have basic user authentication capability.

Step:2 - Process Specification

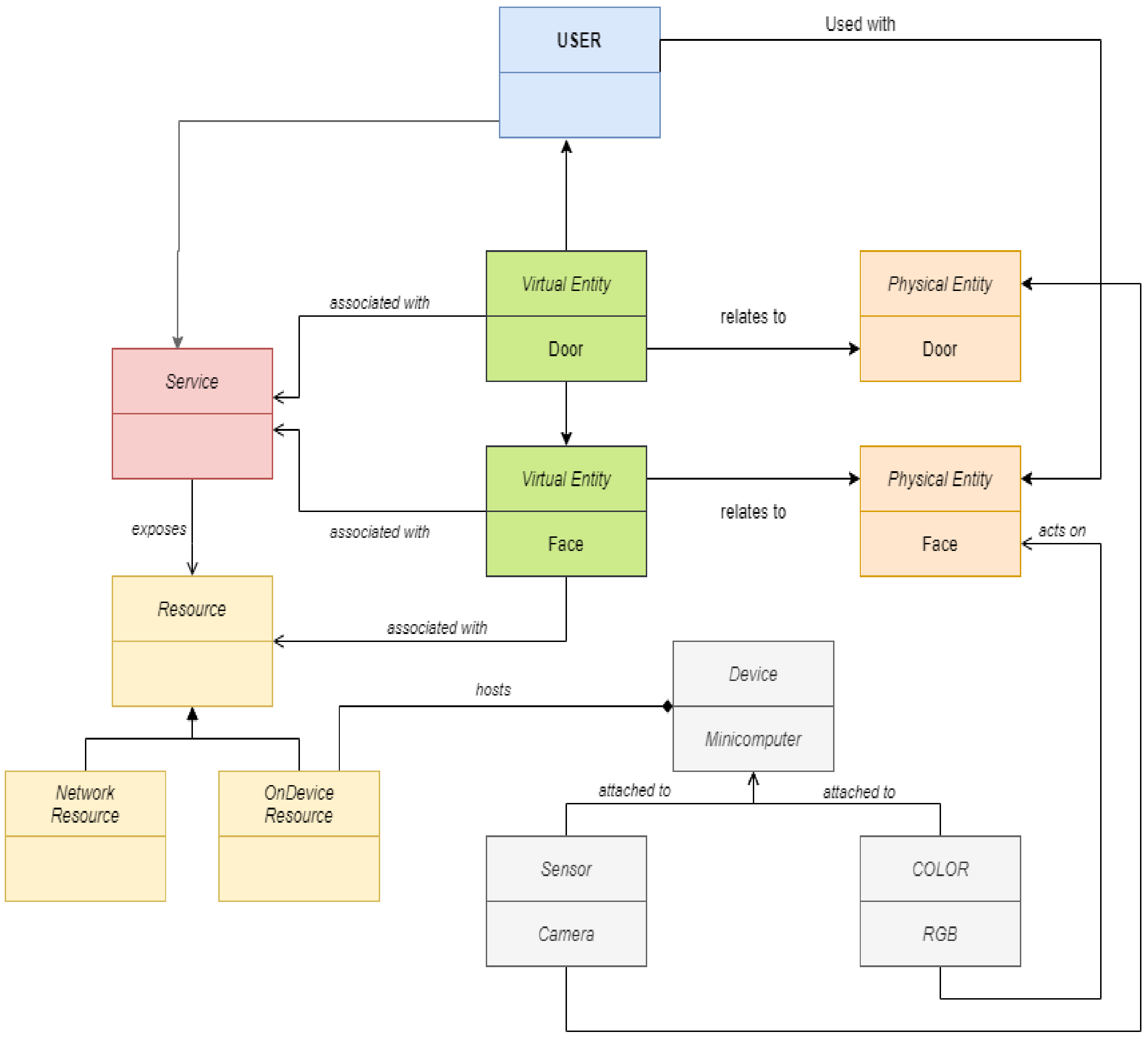


This is a feature that most people choose if they are working at night or want proper supervision in low light too.
Wireless doorbell system Every time you have a visitor, there is a notification on your phone. This takes a few seconds, and the notification light also turns on if there is any movement around the door, even if the guest does not ring the bell.

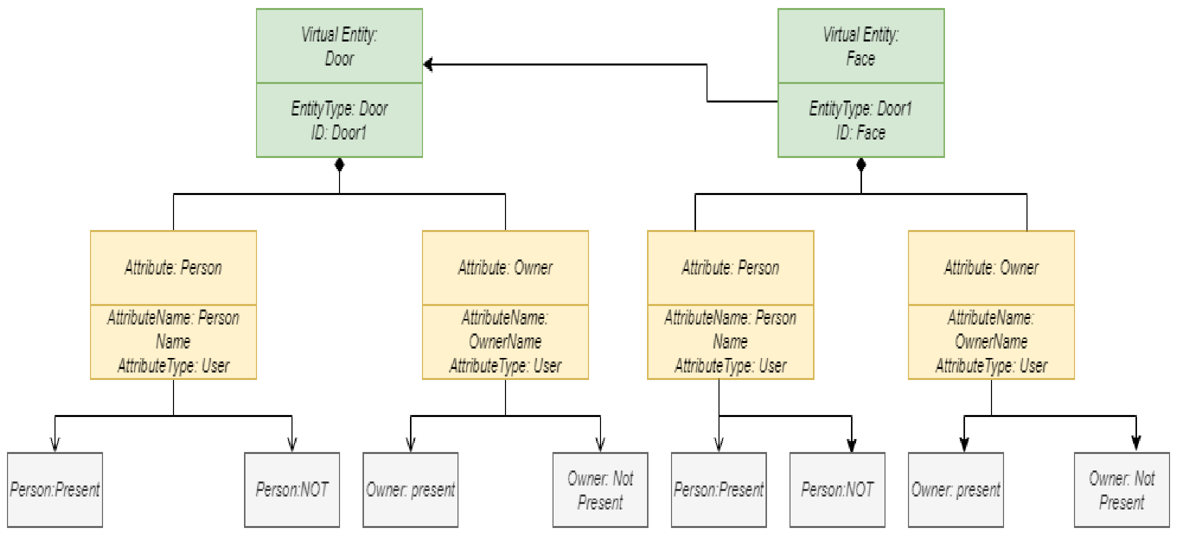
When you choose from the Image capture Doorbells, you will find that many users prefer this feature over any other Image intercom system.
First the Visitors image is taken by the camera and the image is compare with the old database,
if not there and send it to the owner and he received the image if the owner is ok with image, the owner proceed to next step.

Step 3: Domain Model Specification

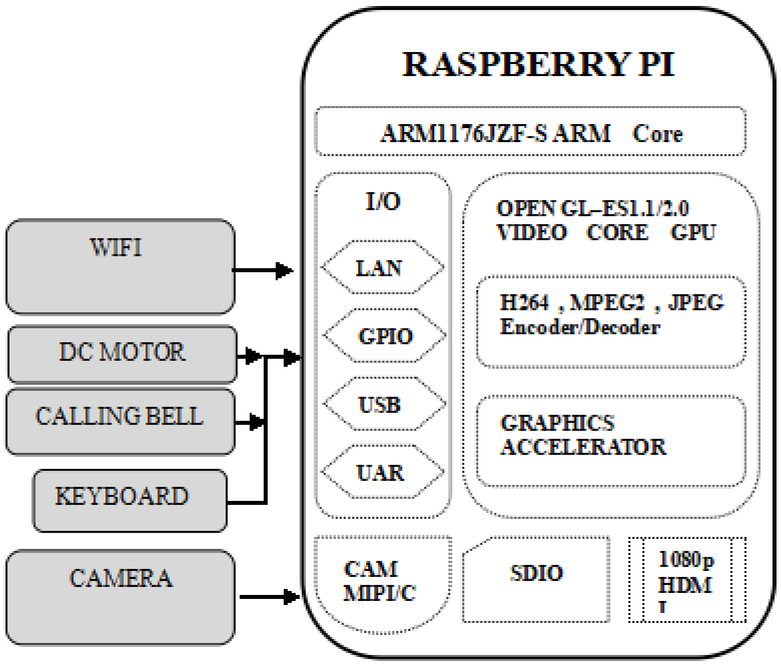
In this proposed system, we are having database of authorized person list by registering their faces by entering OTP, so that non authorized person can't able to enter the home until they entering the OTP. Whenever some person pressing calling bell switch, the camera gets triggered and capture the image of the intruder and checks that the image to the database, if that face is not matching with the database, it sends an email containing that intruder image and OTP, when intruder type the OTP by the owners' knowledge then it allows to enter.


USER
Used with
Virtual Entity
Door
associated with
relates to
Physical Entity
Door
Service
Virtual Entity
Face
associated with
relates to
Physical Entity
Face
acts on
exposes
Resource
associated with
Device
Minicomputer
hosts
Network Resource
OnDevice Resource
attached to
attached to
Sensor
Camera
COLOR
RGB

Step 4: Information Model Specification



BLOCK DIAGRAM:



Block diagram description

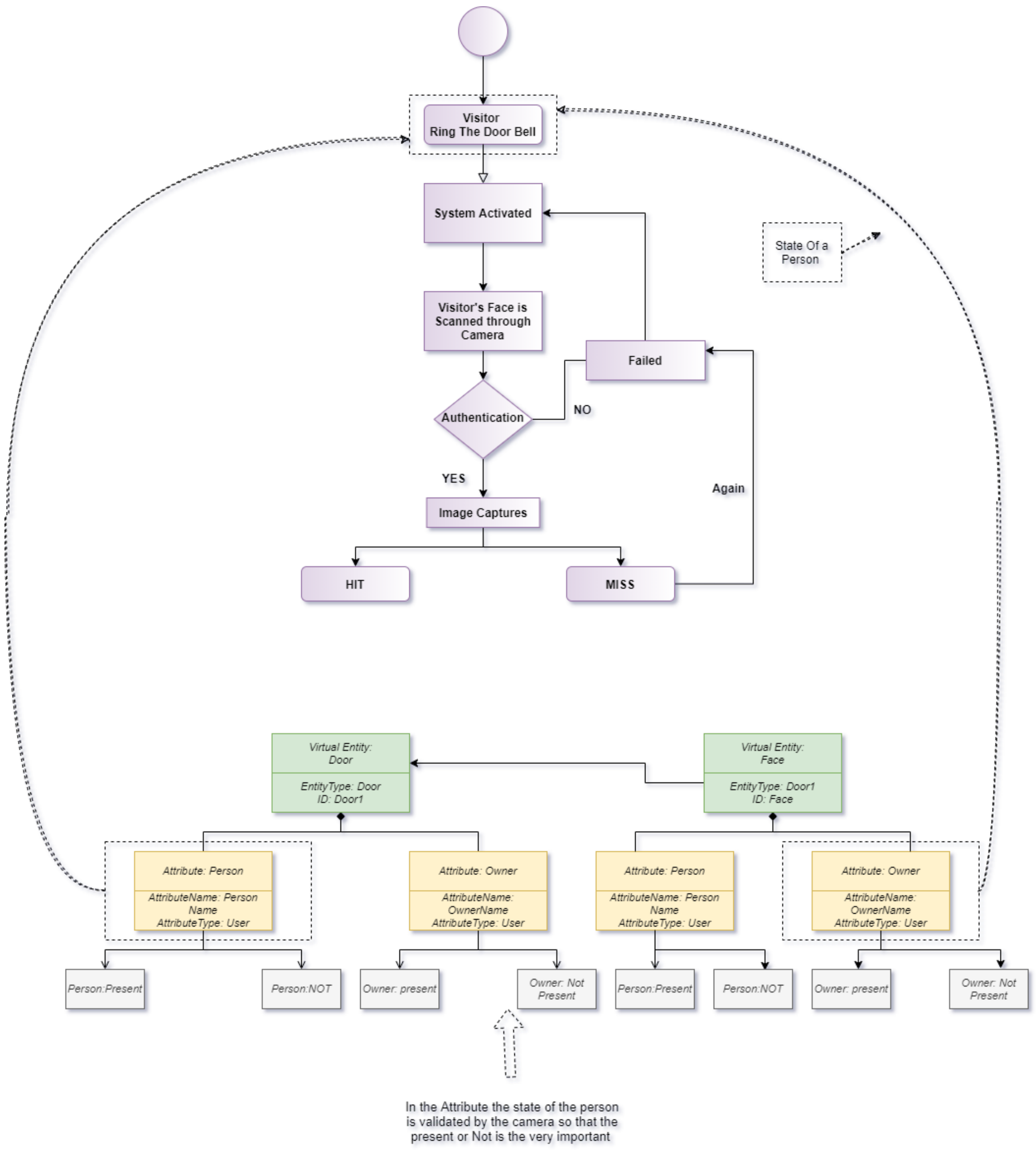
Calling bell switch which is interfaced to raspberry pi through GPIO pins, which triggers the camera when the switch gets pressed that camera is interfaced through USB. Using python OpenCV input image will be processed and authorize the user, for authorized person an alert mail will be sent along with OTP using SMTP. Now the user has to enter the OTP through keyboard which is interfaced in the USB port of the Raspberry pi.

Step 5: Service Specifications

Smart device. These are hardware units, e.g., domestic appliances, lights, or sensors, that can sense, actuate, process data, and communicate. Three core devices are sensors, actuators, and end-user client devices.

User. The stakeholder that uses and benefits from the services offered by the smart connected home. Typically, this represents the 31 residents that manage the different smart connected home devices and services.

Smart home devices differ in terms of their hardware and software capabilities. At one end, there are constrained devices, such as smart door with limited Chip, memory, battery, etc. Then, we find resourceful or high-capacity devices, such as gateways, Or Wifi through the Internet that are typically powered by the main supply


Visitor
Ring The Door Bell
System Activated
State Of a Person
Visitor's Face is Scanned through Camera
Failed
Authentication
NO
YES
Again
Image Captures
HIT
MISS
Virtual Entity: Door
EntityType: Door
ID: Door1
Virtual Entity: Face
EntityType: Door1
ID: Face
Attribute: Person
AttributeName: Person Name
AttributeType: User
Attribute: Owner
AttributeName: OwnerName
AttributeType: User
Attribute: Person
AttributeName: Person Name
AttributeType: User
Attribute: Owner
AttributeName: OwnerName
AttributeType: User
Person:Present
Person:NOT
Owner: present
Owner: Not Present
Person:Present
Person:NOT
Owner: present
Owner: Not Present
In the Attribute the state of the person is validated by the camera so that the present or Not is the very important

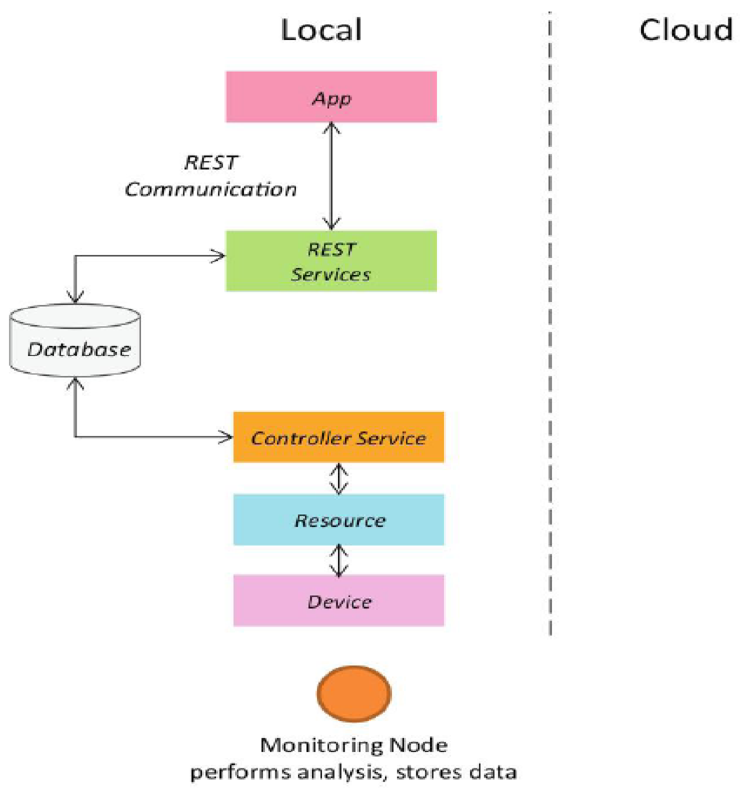
Step 6: IoT Level Specification

Hardware required

- Raspberry Pi with Raspbian OS Installed in it.
- Pi Camera or USB webcam
- Push button
- Jumper wires

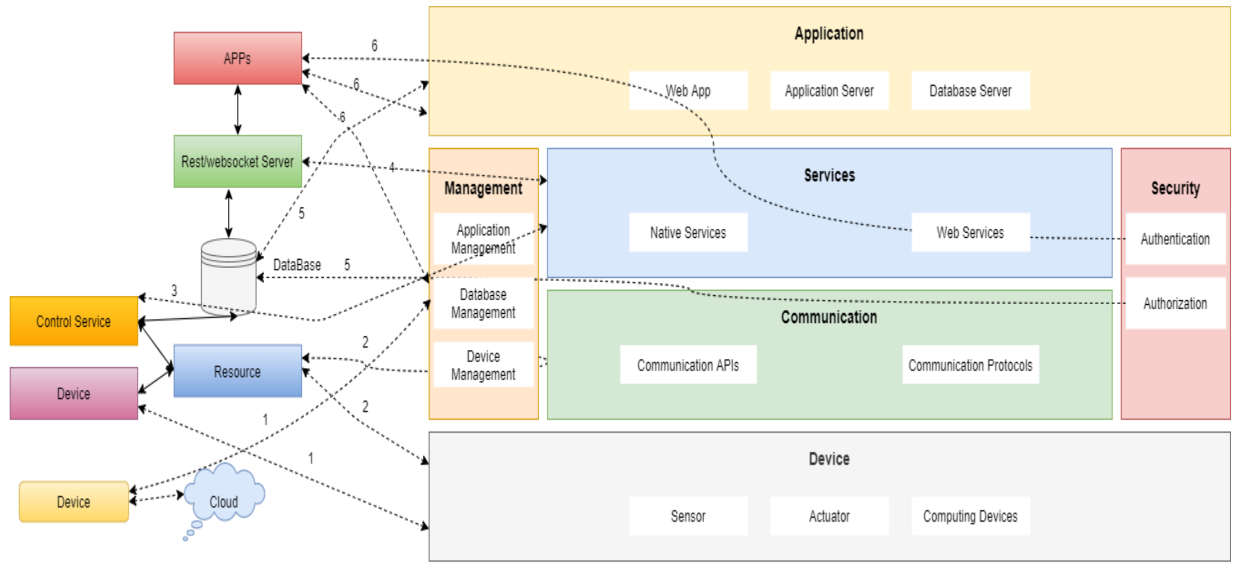
Software required

- Raspbian Jessie OS
- Python
- Open CV
- Apache
- PHP



Step 7: Functional View Specification

RFC 4949 defines data as “information in a specific physical representation, usually a sequence of symbols that have meaning; especially a representation of information that can be processed or produced by a computer,” and information as “facts and ideas, which can be represented (encoded) as various form of data”.
Thus, while the terms are related, in general information is often seen as data that has been processed into a meaningful form . However, both terms are difficult to define in a useful way, and are oftentimes used interchangeably in legislation and regulations .
Legally, any data that can be linked to a person, directly or indirectly, is in general referred to as personal data .
Personal data have been intensely debated and given attention especially by the recent entry into force of the General Data Protection Regulation (GDPR) The GDPR is fundamentally an EU regulation that aims to protect and expand EU citizens’ right to have their data processed safely and only when needed.



1. IoT device maps to the Device FG (sensors, actuators, devices, computing devices) and the Management FG (device management)

2. Resources map to the device FG (on-device resource) and Communication FG (communication APIs and protocols)

3. Controller service maps to the services FG (native service). Web services map to Services FG (web services)

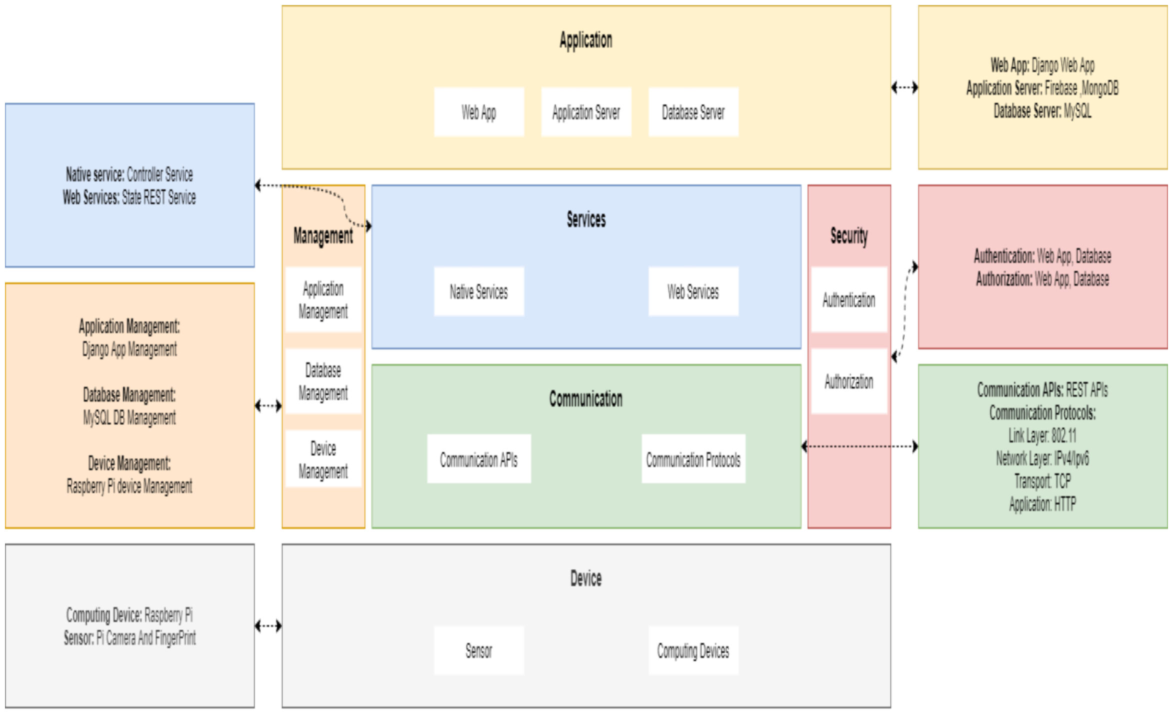
4. Web services map to services FG (web services)

5. Database maps to the Management FG (database management) and security FG (database security)

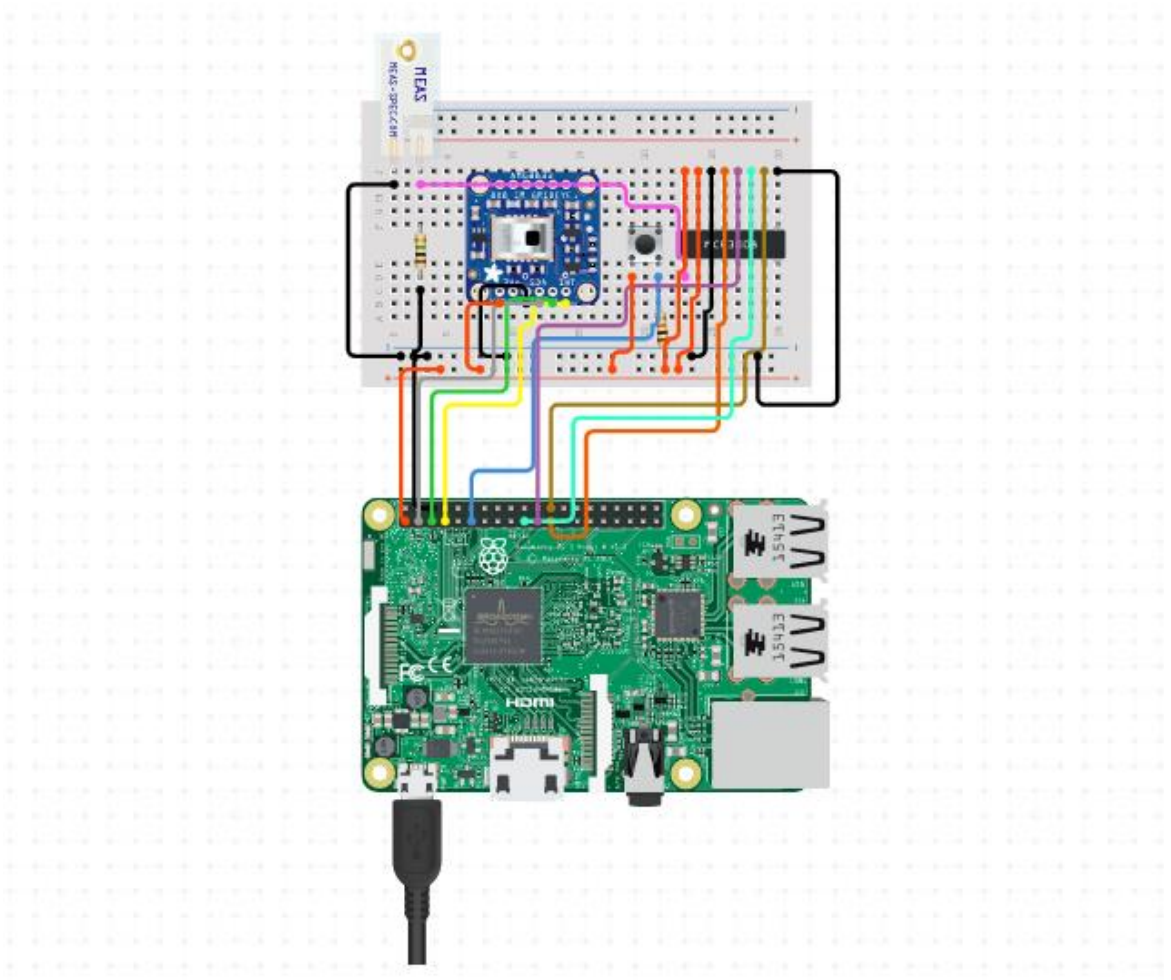
6. Application maps to the Application FG (web application, application and database servers), management FG (app management) and security FG (app security)

Step 8: Operational View Specification

Security is major concern nowadays and today we have all types of surveillance and security system available in the market.
But they are very expensive and sometimes create problems which we can't solve.
Previously we built a surveillance camera which can stream live video on IoT cloud and today we will build a low cost Raspberry Pi based Smart Wi-Fi doorbell.
This system will send the picture of visitor on email when the door bell switch is pressed. A PiCamera is attached with raspberry pi to take the picture, although a USB webcam can also be used if you don't have PiCamera.
This system can be installed at the main door of your home or office and can be monitored from anywhere in the world over internet.



Step 9: Device & Component Integration



Step 10: Application Development

Here, we will use SSH to access our Raspberry Pi on the laptop. If you have monitor then it will be very easy to start with but if you don’t have a monitor then setup raspberry pi in headless mode or use VNC server to get Raspberry Pi desktop on Laptop.

Raspberry Pi is very popular for building IoT based projects as it has all the necessary support for Internet of Things. It is a palm size computer having inbuilt Wi-Fi, Bluetooth, USB port, Audio/video port, HDMI port, camera port etc. You can check all the Raspberry Pi based Iot Projects here.

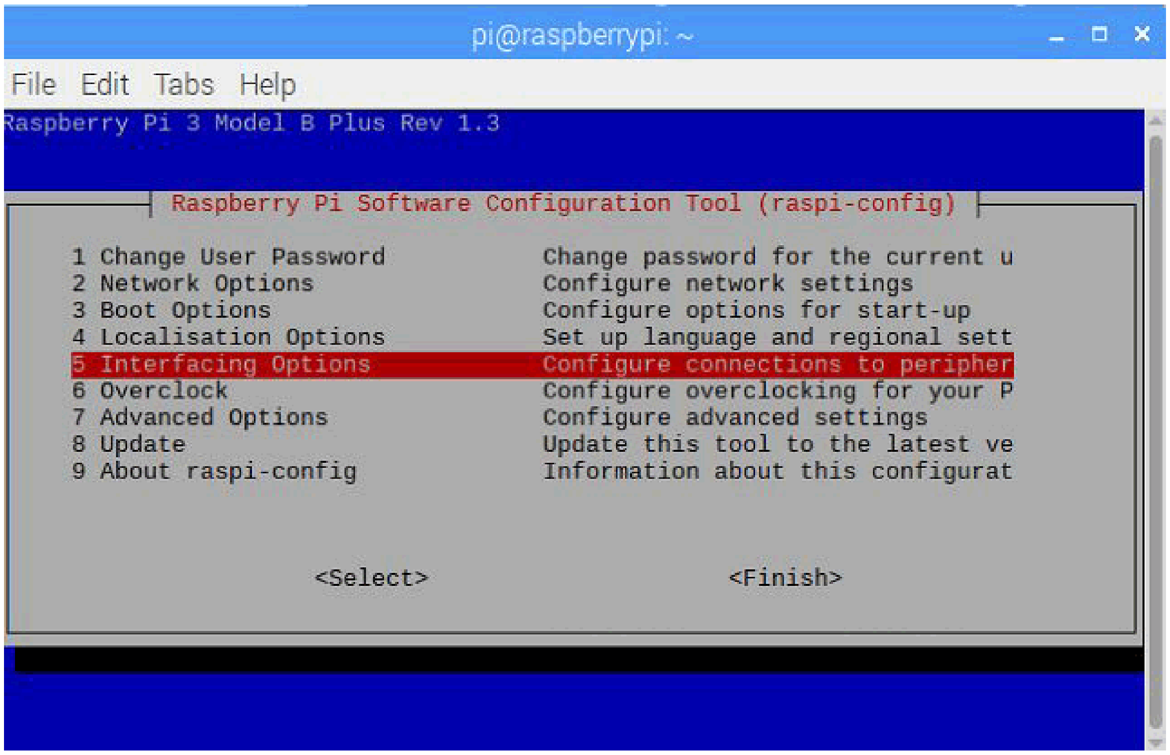
A Django application with a MySQL database, hosted on a cloud service exposing 2 endpoints for the purposes
described above.
● The first endpoint would take an image as an input, get a prediction from a trained
Convolutional Neural Network, and return the same as the HTTP response.
● The second endpoint would receive the state of the car, which is stored in the
MySQL database, which can be used for a real-time dashboard.

Implementation:

Setup Pi Camera or USB webcam with Raspberry Pi

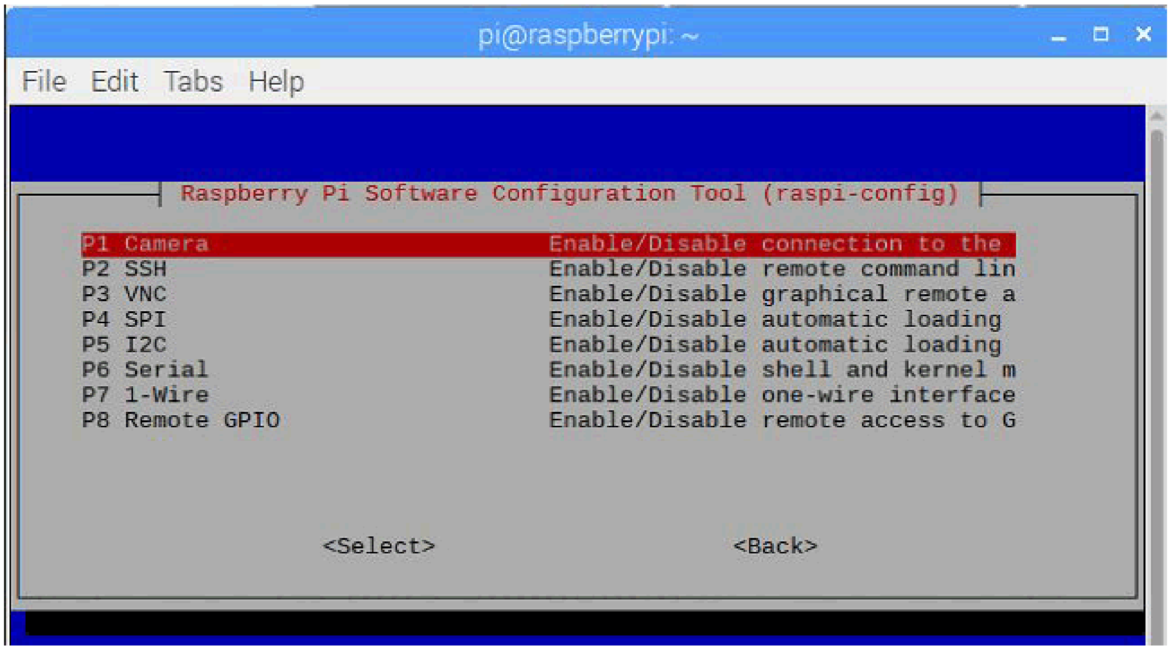
PiCamera:

1. If you are using picam then you have to enable camera interfacing from raspi-config. Run the sudo raspi-config and go to Interfacing options.



```
Raspberry Pi 3 Model B Plus Rev 1.3

         ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

   1 Change User Password            Change password for the current u
   2 Network Options                 Configure network settings
   3 Boot Options                    Configure options for start-up
   4 Localisation Options            Set up language and regional sett
   5 Interfacing Options             Configure connections to peripher
   6 Overclock                       Configure overclocking for your P
   7 Advanced Options                Configure advanced settings
   8 Update                          Update this tool to the latest ve
   9 About raspi-config              Information about this configurat


                  <Select>                        <Finish>
```

2. Then select the Camera option and Enable it on the next window and restart the Pi.



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera           Enable/Disable connection to the
P2 SSH              Enable/Disable remote command lin
P3 VNC              Enable/Disable graphical remote a
P4 SPI              Enable/Disable automatic loading
P5 I2C              Enable/Disable automatic loading
P6 Serial           Enable/Disable shell and kernel m
P7 1-Wire           Enable/Disable one-wire interface
P8 Remote GPIO      Enable/Disable remote access to G
```

3. Now, test the camera by capturing a photo using below command.

USB Camera:

If you are using USB webcam then you have to install some packages to enable the webcam functionalities. Install the package using below command

```
sudo apt-get install fswebcam
```

Installing SMTP on Raspberry Pi to send Emails

1.Simple Message Transfer Protocol (SMTP) is the communication protocol which is used to send e-mails. It is nice and easy solution to send e-mails using command line or python scripts. So we need to install some libraries and packages on Raspberry Pi to send email using SMTP

2. Now, install SMTP library packages using following commands

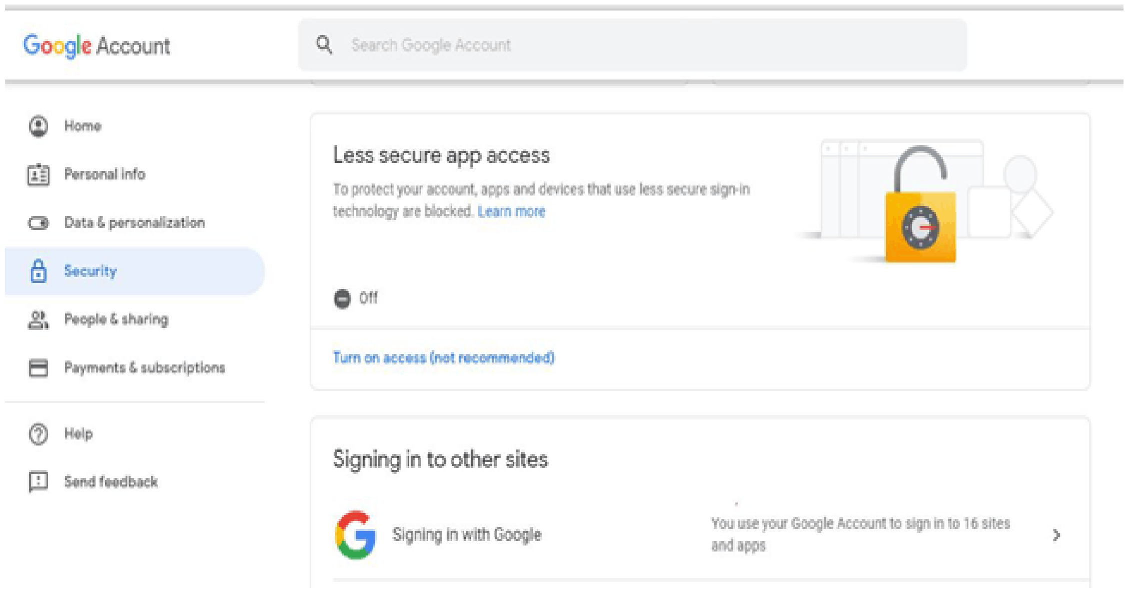
```
sudo apt-get install ssmtp

sudo apt-get install mailutils
```

Modify Security Settings in Google Mail account

Google does not allow to send and receive e-mails which contains Python code. So, we have to update some security settings in Google account. Follow below steps to enable "***Allow less secure apps***" permission.

1. Login to your Gmail account by entering your login credentials.

2. Click on profile picture and then click on "Google account".



Google Account

Search Google Account

Less secure app access

To protect your account, apps and devices that use less secure sign-in technology are blocked. Learn more

Off

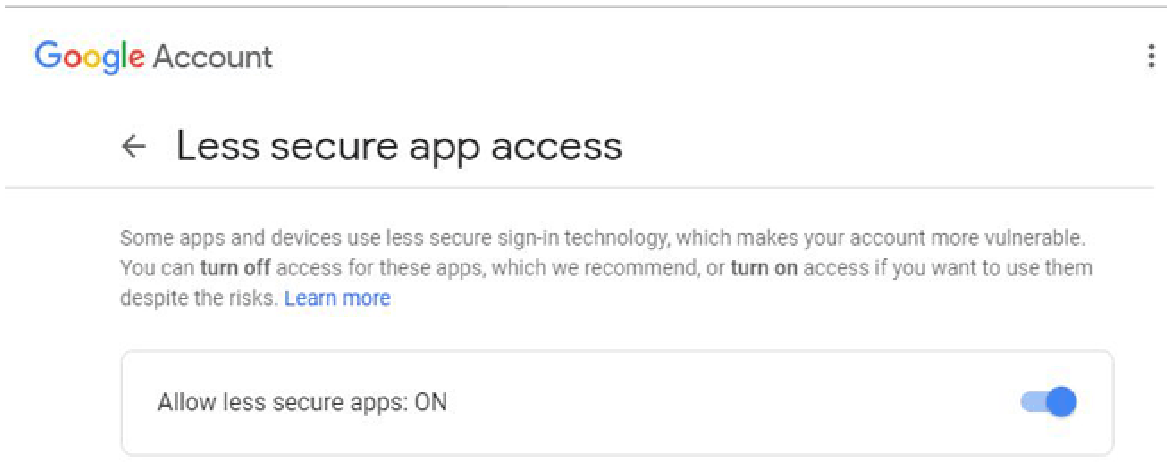
Turn on access (not recommended)

Signing in to other sites

Signing in with Google

You use your Google Account to sign in to 16 sites and apps

3. Under Security tab you will find ***Less secure app access***. Turn it ON by Clicking on "***Allow less secure apps***".



Google Account

Less secure app access

Some apps and devices use less secure sign-in technology, which makes your account more vulnerable. You can **turn off** access for these apps, which we recommend, or **turn on** access if you want to use them despite the risks. Learn more

Allow less secure apps: ON

Code and Explanation

Complete Python code and a demo video are given at the end of this tutorial. Here we are explaining the code to understand the working.

Just a quick recall, here we are writing a python script to send mail with the photo of visitor as an attachment whenever the doorbell switch is pressed. The Pi camera will capture the photo and send it to the owner of the house over the e-mail.

1. Open your favourite text editor in Raspberry pi and import all the important libraries for picamera, Rpi GPIOs, SMTP, time.

```
import os

import picamera

import RPi.GPIO as GPIO

import smtplib

from time import sleep
```

2. Now, import all the modules required to send e-mail. For writing plain text, including attachment and subject we need a separate module which composes a whole mail.

```
from email.MIMEMultipart import MIMEMultipart

from email.MIMEText import MIMEText

from email.MIMEBase import MIMEBase

from email import encoders
```

3. Assign your e-mail id, receiver e-mail id and password in a variable as shown.

```
sender = 'sender@gmail.com'

password = 'gmail password'

receiver = 'receiver@gmail.com'
```

4. In order to save the captured photo in a directory and give different names to them assign a folder and a prefix name.

```
dir = './visitors/'

prefix = 'photo'
```

5. Set pin mode and pin number to attach a push button which act as a doorbell switch.

```
GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(15, GPIO.IN)
```

6. Now, **make a function to capture photos**. In this function we have to check for the directory if it already exists or not. If not then make the directory.

```
def capture_img():

if not os.path.exists(dir):

        os.makedirs(dir)
```

Assign a file name and sort it using *glob* and find the largest ID of existing images and Start new images after this ID value.

```
 files = sorted(glob.glob(os.path.join(dir, prefix + '[0-9][0-9][0-9].jpg')))

 count = 0
```

Grab the count from the last filename.

```
 if len(files) > 0:

     count = int(files[-1][-7:-4])+1
```

Now, capture the photo and give it a unique name and save it in the defined folder.

```
filename = os.path.join(dir, prefix + '%03d.jpg' % count)

with picamera.PiCamera() as camera:

pic = camera.capture(filename)
```

7. Now, make another **function to send mail**. In this function we will attach subject, body and attachments and then send all the content to the receiver using SMTP.

```
def send_mail():
    msg = MIMEMultipart()
    msg['From'] = sender
    msg['To'] = receiver
    msg['Subject'] = New Visitor'
    body = 'Picture is Attached.'
    msg.attach(MIMEText(body, 'plain'))
    attachment = open(filename, 'rb')
..
..
    msg.attach(part)
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(sender, password)
    text = msg.as_string()
    server.sendmail(sender, receiver, text)
    server.quit()
```

Now finally, read the push button value and when its goes high, Raspberry Pi calls the ***capture_img()*** function to capture the image of visitor and send a alert email with the picture of visitor as an attachment. Here ***send_mail()*** is used inside the ***capture_img()*** function for sending the mail.

Code

```
import os
import glob
import picamera
import RPi.GPIO as GPIO
import smtplib
from time import sleep

from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders

sender = 'PranavHegde@gmail.com'
password = '********'
receiver = 'abc@gmail.com

DIR = './Visitors/'
prefix = 'image'

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(15, GPIO.IN)

def send_mail(filename):
    msg = MIMEMultipart()
    msg['From'] = sender
    msg['To'] = receiver
    msg['Subject'] = 'Visitor'

    body = 'Find the picture in attachments'
    msg.attach(MIMEText(body, 'plain'))
    attachment = open(filename, 'rb')
    part = MIMEBase('application', 'octet-stream')
    part.set_payload((attachment).read())
    encoders.encode_base64(part)
    part.add_header('Content-Disposition', 'attachment; filename= %s' % filename)
    msg.attach(part)
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(sender, password)
    text = msg.as_string()
    server.sendmail(sender, receiver, text)
    server.quit()

def capture_img():
    print ‘Capturing’
    if not os.path.exists(DIR):
        os.makedirs(DIR)
    files = sorted(glob.glob(os.path.join(DIR, prefix + '[0-9][0-9][0-9].jpg')))
    count = 0

    if len(files) > 0:
        count = int(files[-1][-7:-4])+1
    filename = os.path.join(DIR, prefix + '%03d.jpg' % count)
    with picamera.PiCamera() as camera:
        pic = camera.capture(filename)
    send_mail(filename)

while True:
    in = GPIO.input(11)
    if in == 0:
        print "Waiting", i
        sleep(0.4)
    elif in == 1:
        print "Captured->Sending", i
        capture_img()
```

CONCLUSION

This system can also be reconfigured to detect the intruder who are not pressing the doorbell, whose face get captured and follows the same authorization process to know the intruder who tries to open the door. Which can be very useful to monitor the home remotely.

And also, for the owner the system will behave in another manner where it will not perform the same process for the owner i.e. processed for an unknown person it will just recognize the voice and face and the door will be opened.