# Chapter 01

## 1.1 Introduction

In this time era, security issues have surfaced and must be handled with utmost priority. Every business person strives to keep their employees, assets and office space safe and same goes for home security. A huge number of terrible incidents and crimes occur daily and the count keeps on increasing due to lack of proper security measures. The growing crime rates across various cities reflect the bitter reality. This has caused a stir among people. Many people ignore and underestimate the need for taking appropriate security measures, which may, consequently, result in burglaries or thefts which leave them financially and emotionally drained and traumatizes them for life.

There are various ways to help increase the security at door level itself and one of the solutions is to install a security system that is simple and easy to use, and can protect people from any sort of danger. If one delves deeper into the reasons as to why one might need a proper security system at their home when there are locks, then one can find himself standing in today's world and gazing ahead at the people who are fast-paced with the latest technologies available to them for better security actions.

A security system is there to protect you by raising the alarm in cases of emergency like fire, break-in, etc. But a smart solution doesn't just end there. It is not limited to this, thanks to advanced technology that proves itself useful every time. It can send you a notification when your kids get home. It can show you who is at your front door and lets you talk to the person via a video doorbell system. It

can also control things like heating or cooling with the help of a thermostat. There might be smart locks, lights and even a garage door controller. Even if you are miles away from home, you can have a peace of mind knowing that you are connected to your home with smart security system. You can get instant notifications to show you what's going on using images and video clips. If you need to give a neighbor emergency access, then you can unlock your door as easily as if you were there in person. Hence, a Smart Security System has been proposed.

## 1.2 Motivation

After estimating the need for proper security measures, this idea of an IoT based security system has been proposed. The door locks generally used at present in most places can be easily unlocked using a master key whereas this system consists of special features which require conditions to open a door lock. An advanced step towards technology making lives better, simpler and most secure is always welcome and that exactly brought up the motivation behind this project. The security system consists of two sections where each section has a criterion or condition to fulfill in order to unlock the door. Smartphones are very common these days; hence, implementation and execution of this security system won't be an issue and it won't be difficult to use either. Hence, this Smart Security System has been proposed keeping in view the need for an advanced security system.

## 1.3 Contribution Summary

The various components of the Smart Security System demonstrated in this project have been described as follows:-

- Raspberry Pi as a mini OS to make all modules work and make the system function in accordance
- Camera module attached to the Raspberry Pi to click images of the visitor
- Passive Infrared (PIR) Sensor attached to the Raspberry Pi to detect motion or movement of the visitor to activate the camera
- An electric door lock that requires 12V to operate
- The face detection program is done in OpenCV in Python using Local Binary Patterns Histograms (LBPH) Recognizer algorithm.
- Using LBPH Recognizer algorithm ensures faster execution time for the whole process as this algorithm makes sure that each face is trained separately giving complete accuracy.

# Chapter 02

## 2.1 System Architecture

We use Raspberry Pi 3 Model B to implement our security system which is shown in Fig. 2.1.

The ports required for the system are depicted in Fig. 2.2.

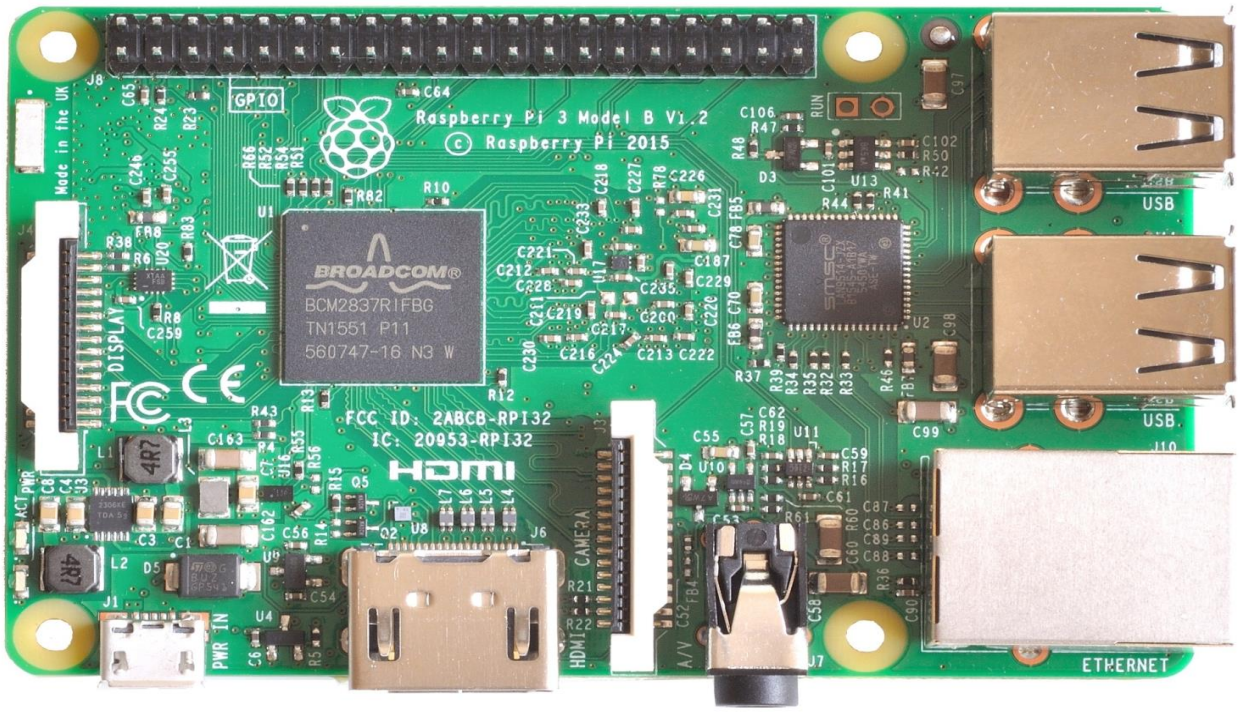


**Fig. 2.1: Raspberry Pi 3 Model B Architecture**

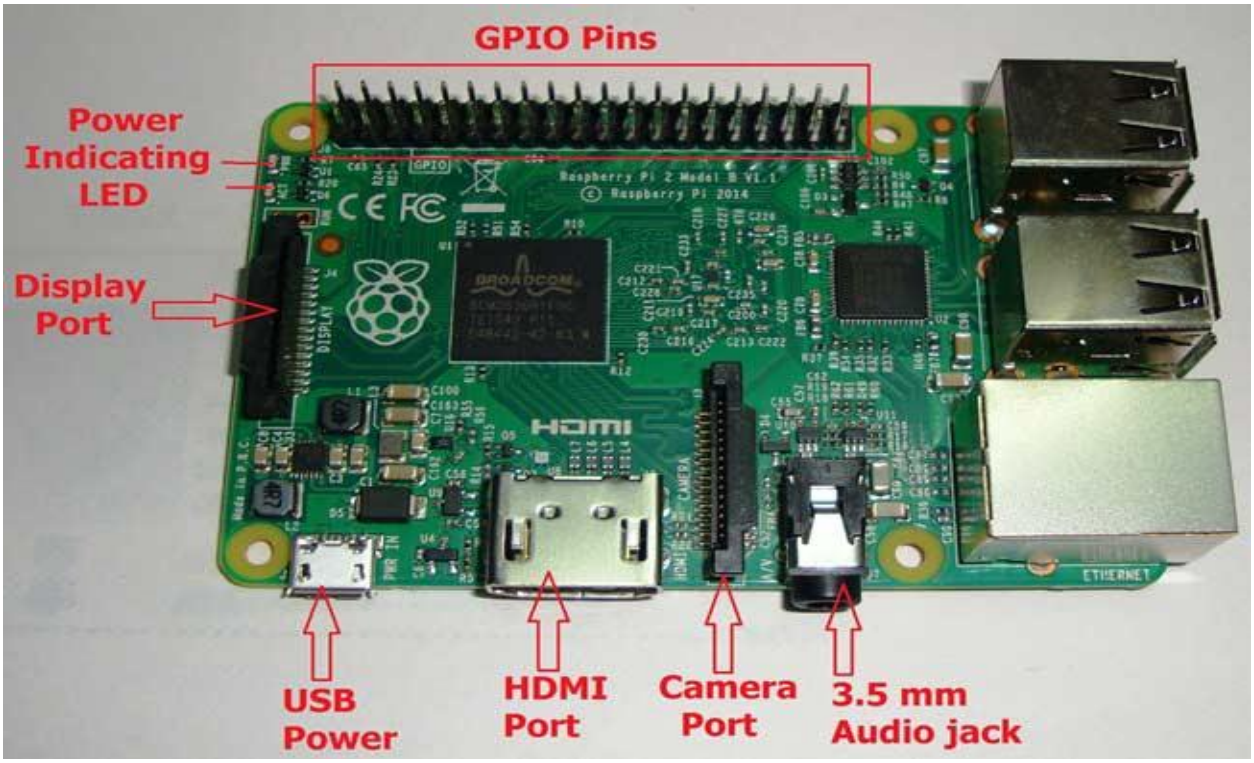


**Fig. 2.2: Raspberry Pi 3 Model B Ports Specifications**

The system design of the Raspberry Pi module as depicted in Fig. 2.3 consists of the following components:-

- PIR sensor to sense movements and activate the camera instantly
- Raspberry Pi Camera to click images of the object in front
- Power Bank as a power source for the circuit to work
- Resistor to lower the voltage level for the LED lights (from 5V to 1V or 2V)
- Micro SD Card for the Raspbian Jessie OS to be installed and for a database for the images
- LED Lights to check the system works and the connection is complete
- Breadboard for building the circuit as they require no soldering
- HDMI Cable to connect with a display screen to show the actions of the system
- Jumpers for connections in the circuit

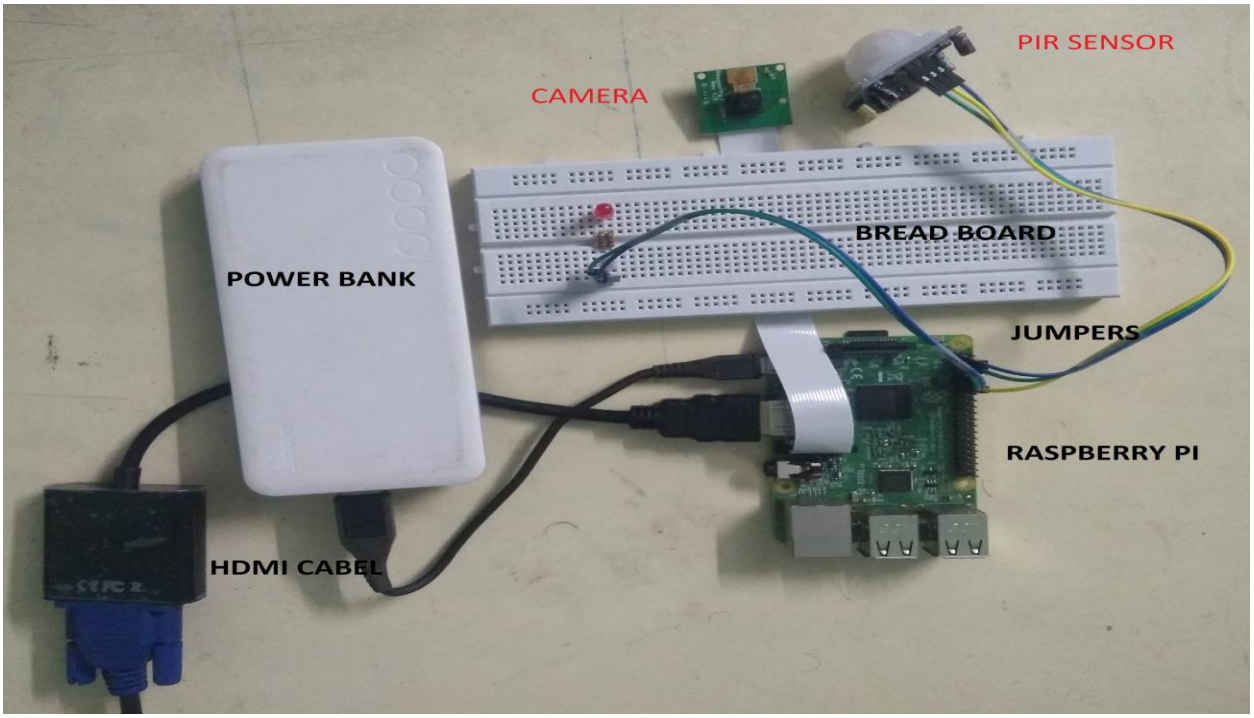


**Fig. 2.3: System Design for Smart Security Camera**

## 2.2 Local Binary Patterns Histogram (LBPH) Recognizer Algorithm

Human beings perform face recognition everyday and with practically no effort at all. However, in Computer Science, face recognition is basically a task of recognizing a person based on its facial image. Face detection is different from face recognition in the sense that face detection has the objective to find the faces (location and size) in an image and extract them to be used for face recognition, whereas face recognition is responsible for finding characteristics that best describe the image (which is cropped and generally converted to grayscale).

The face recognizer algorithm works in two modes:-

- **Verification or authentication of a facial image** – it compares the input facial image with the facial image related to the user which is requiring the authentication (1x1 comparison).
- **Identification or facial recognition** – it compares the input facial image with all facial images from a dataset with the aim to find the user that matches that face (1xN comparison).

In Python, OpenCV 2.4 onwards comes with the Face Recognizer class for face recognition. There are various algorithms available for the same, which are:-

- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms
- Scale Invariant Feature Transform
- Speed Up Robust Features

For the purpose served in this project, we implement the Local Binary Patterns Histograms Face Recognizer Algorithm because it is easier to understand, robust against monotonic grayscale transformations, works better in different environments and light conditions, can represent local features in images and has a more efficient way of detecting faces than the rest of the aforementioned algorithms. The LBPH method is explained in detail.

1. **Parameters**: LBPH uses 4 parameters:

    - **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel (usually set to 1).

    - **Neighbors**: the number of sample points to build the circular local binary pattern (usually set to 8).

    - **Grid X**: the number of cells in the horizontal direction (usually set to 8).

    - **Grid Y**: the number of cells in the vertical direction (usually set to 8).

2. **Creating the Dataset**: First, we need to create the dataset with the facial images clicked by the camera. They are turned to grayscale and stored in a folder called dataset. We store the images using an ID so the algorithm will use this information to recognize an input image and give you an output.
3. **Training the Algorithm**: Now, we need to train the algorithm. To do so, we need to use the dataset with the facial images of the people we want to recognize. Images of the same person must have the same ID. The training set is constructed as .yml file.

4. **Applying the LBP operation**: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

   The image below describes the procedure:

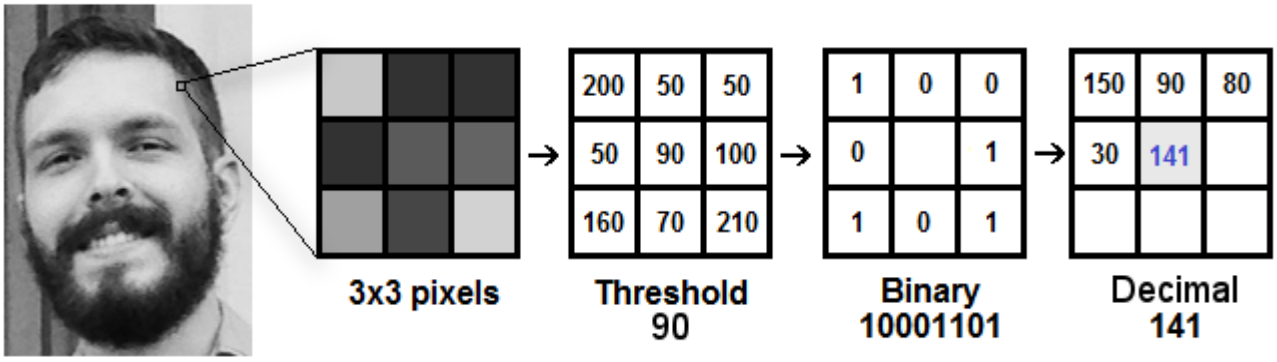


**Fig. 2.4: Local Binary Patterns (LBP) Procedure**

- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0-255).
- We need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.

- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Some people use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.

- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.

- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

5. **Extracting the Histograms**: Now, using the image generated, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, as can be seen in the following image:

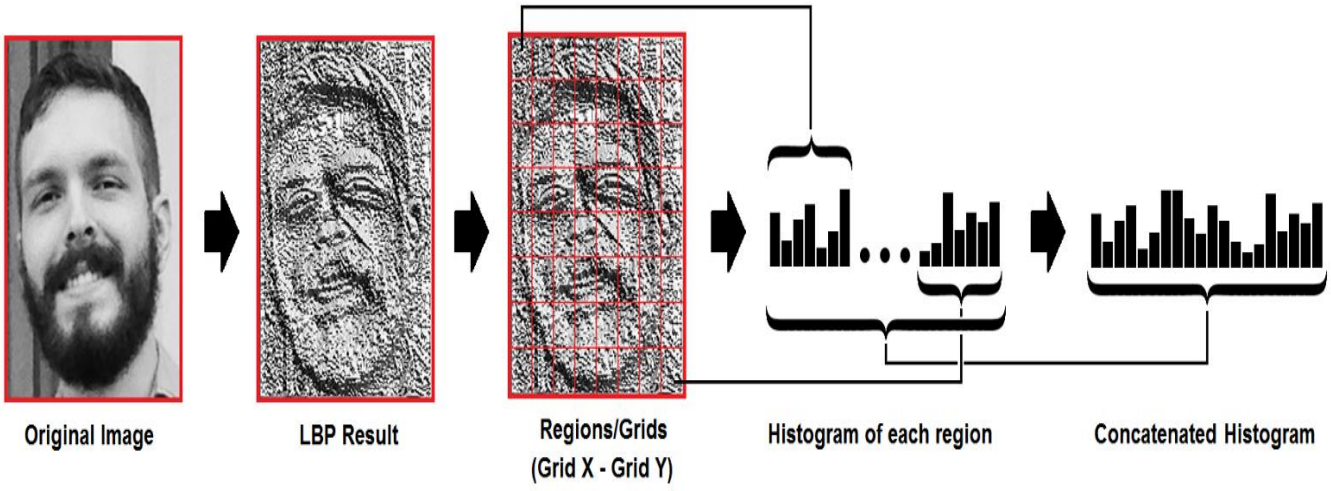


**Fig. 2.5: Histogram after Division of the LBP Image into Grids**

Based on the LBP resultant image, we can extract the histogram of each region:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0-255) representing the occurrences of each pixel intensity.

- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8*8*256=16,384 positions in the final histogram.

The final histogram represents the characteristics of the original image.

6. **Performing the face recognition**: With the algorithm already trained, each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and create a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.

- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example, Euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a 'confidence' measurement. Lower confidences are better because it means the distance between the two histograms is closer.

- We can use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. The algorithm has successfully recognized if the confidence is lower than the threshold defined.

# Chapter 03

## 3.1 Circuit Design and Workflow of the System

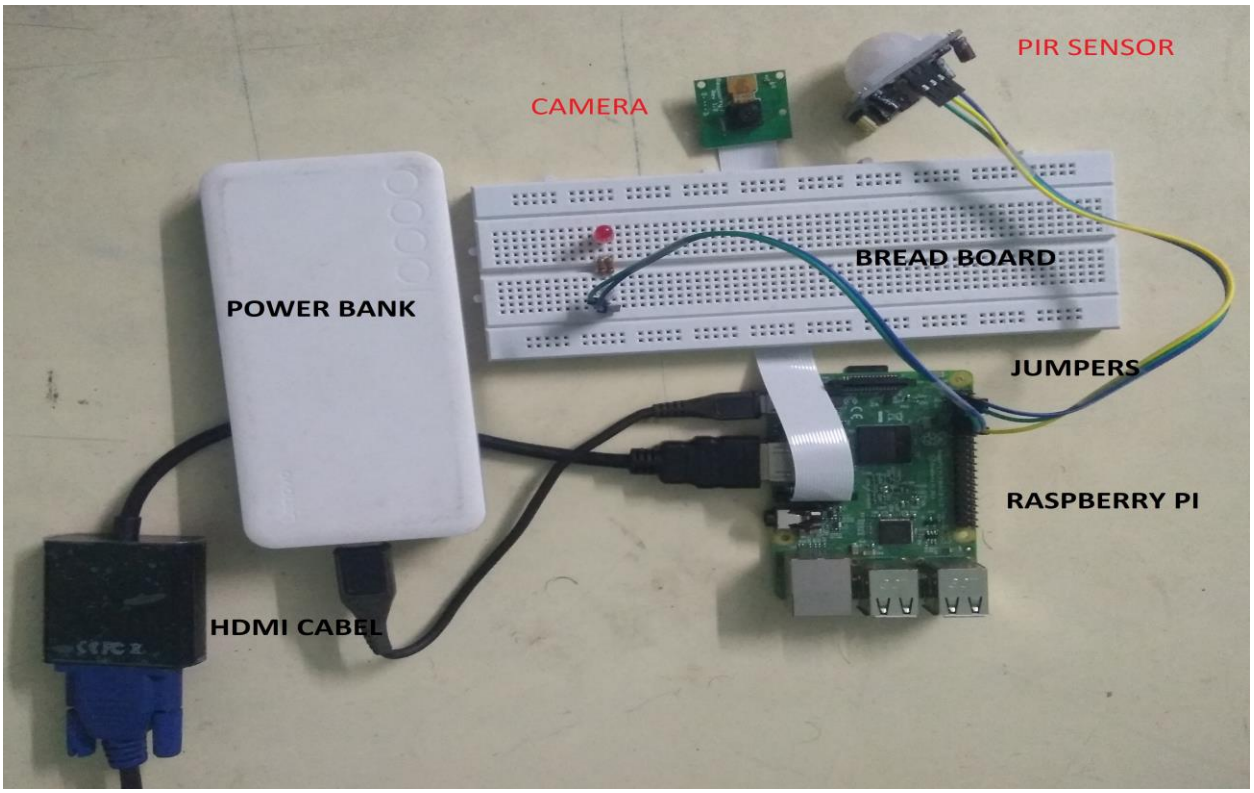


**Fig. 3.1: Circuit Design for PIR sensor and Camera module**

The circuit for Smart Security System consists of the components as labeled in the image above. The components are connected via jumpers. The Raspberry Pi is connected to a power source and the HDMI cable to view operations. The Pi camera is connected with the Raspberry Pi and the PIR sensor is connected with the breadboard and Raspberry Pi. If it detects any sort of movement, it activates the camera module which then clicks images and sends it via Gmail to the user using certain codes in Python which are saved in the Micro SD Card as it contains the OS for Raspberry Pi.

The entire process flow of the Smart Security System is depicted appropriately in the workflow diagram in Fig. 3.2.

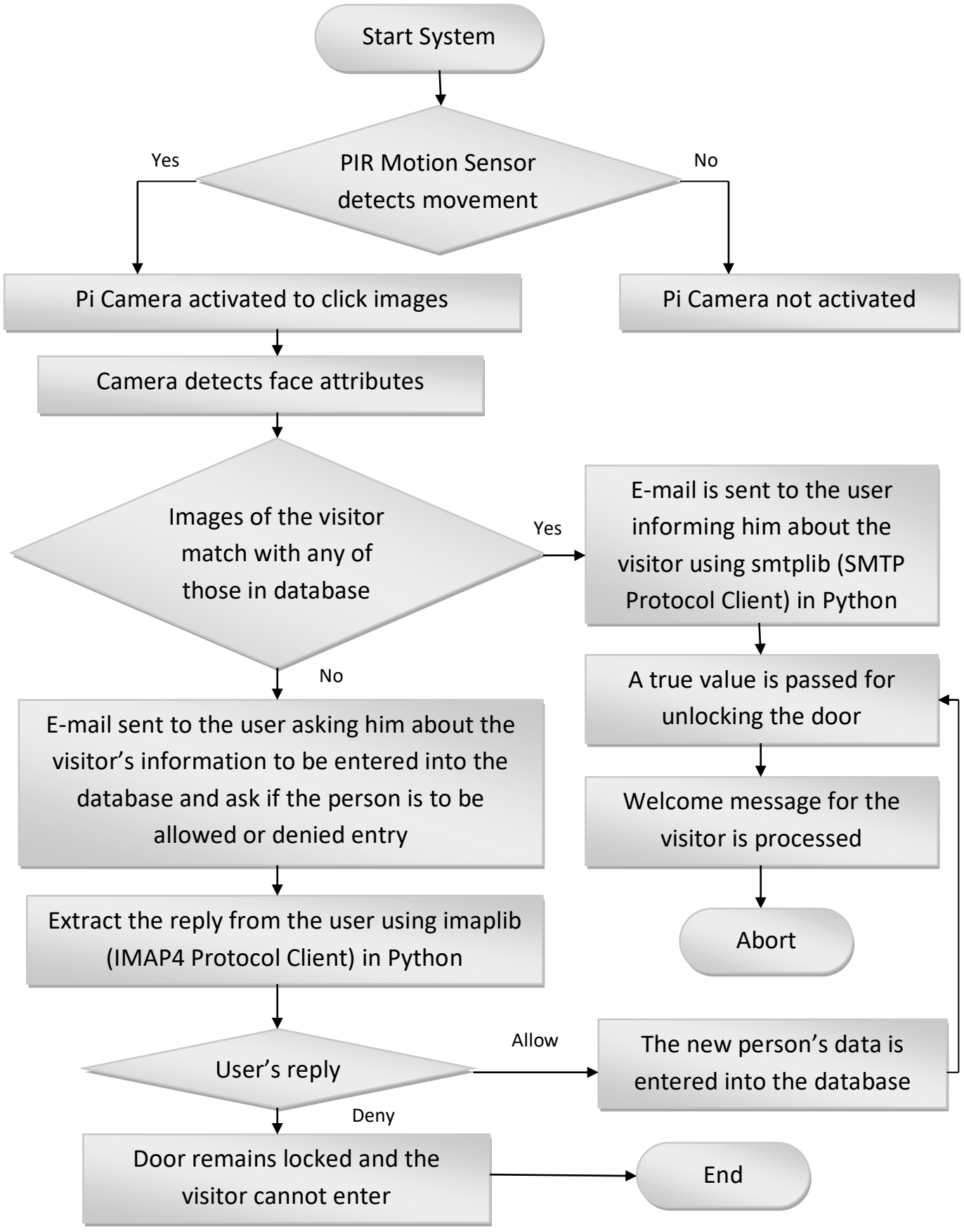


**Fig. 3.2: Workflow of Smart Security System**

# Chapter 04

## 4.1 Software Requirement

Software required:-

- Raspbian Jessie (OS)
- Python (version 2.7.14)
- OpenCV (cv2 version 3.4.0)
- PIL or Pillow modules as necessary (PIL version 5.1.0)

## 4.2 Hardware Requirement

Hardware required:-

- Raspberry Pi 3 Model B
- Pi Camera (5 MegaPixel)
- PIR (Passive Infrared) Sensor
- LED
- Breadboard
- Resistor (1K)
- Connecting Wires (Jumpers)
- Power Supply (Power Bank)
- HDMI Cable

# Chapter 05

## 5.1 Process Execution

The system begins execution from the PIR motion sensor. The Passive Infrared Sensor is an electronic sensor that measures infrared light radiating from objects in its field of view. It is used to sense movement of people, animals or other objects. It is commonly used in theft or break-in alarms and automatic activated light systems. Fig. 5.1 shows the PIR motion sensor used in this system. Any movement in the sensor's field of view will make its temperature rise from room temperature to body temperature, and then back again. This incoming infrared radiation as body heat is converted into an output voltage which triggers the detection.


**Fig. 5.1: PIR (Passive Infrared) Motion Sensor**

When the PIR sensor detects movement in its field of view, it activates the Pi Camera module of 5 mega pixels as shown in Fig. 5.2. The camera clicks the image of the person in front as shown in Fig. 5.3.

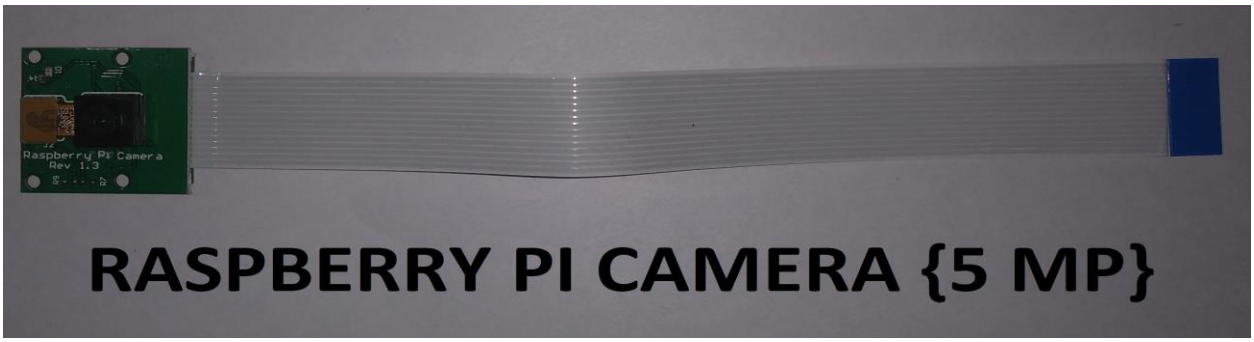


**Fig. 5.2: Pi Camera**

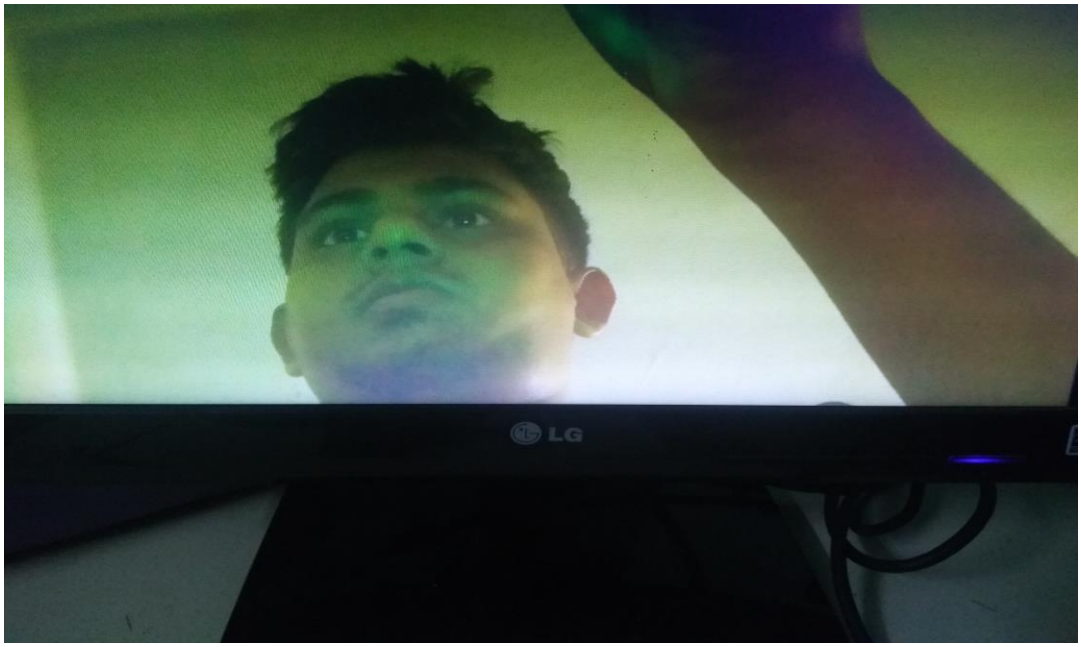

**Fig. 5.3: Pi Camera module working and clicking images**

After the images have been clicked, the camera extracts the features of the face after locating the face. These extracted features are then compared with the images of the faces stored in the database (of people already known to the

user/resident). Here, the frontal face detection scene comes into view which has been discussed in detail in section 5.2. For detecting and recognizing the face, the algorithm used is Local Binary Patterns Histograms (LBPH) Face Recognizer Algorithm.

If the face of the user matches with those in the system, the door lock will open. At the same time, an e-mail is sent to the user or resident indicating that XYZ has entered the house along with the images clicked. As such, the user can keep track of who visits his house and when. The e-mail is sent using Gmail and SMTP (Simple Mail Transfer Protocol) Protocol Client. The Python code imports smtplib module which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener or daemon. The smtplib includes the class SMTP, which is useful for communicating with mail servers to send mail. The connection made using port 587 in SMTP server along with the code is explained in Fig. 5.4. As a true value is passed for the lock to be unlocked, the process ends and the person gets in.

```
mail.attach(image)
server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login("kp.komalparikh1@gmail.com", "komalparikh1*")
text = mail.as_string()
server.sendmail("kp.komalparikh1@gmail.com", "kp.komalparikh7@gmail.com", text)
server.quit()
```

**Fig. 5.4: SMTP Connection Establishment using port 587**

If the face of the person does not match with any of those in the database, then an e-mail is sent to the user or resident asking for permission. If the user replies

“accept” or “allow,” then a true value will be passed for the lock to be unlocked else the person is denied access to enter the house. At this stage, another action takes place, i.e. data entry. The user can allow the person to enter and also, get his name and images in the database so that next time, such permission e-mails are not sent and the person is allowed hassle-free entry into the house. Thus, extracting information from the e-mail reply is done with the help of IMAP4 (Internet Message Access Protocol) Protocol Client. The Python code imports imaplib which implements a client for communicating with IMAP version 4 servers. The IMAP protocol defines a set of commands sent to the server and the responses delivered back to the client. It basically has three client classes for communicating with servers – IMAP4 which uses clear text sockets, IMAP4_SSL which uses encrypted communication over SSL sockets, and IMAP4_stream which uses standard input and standard output of an external command. The connection establishment using imaplib is shown in Fig. 5.5.

```
mail = imaplib.IMAP4_SSL(SMTP_SERVER)
mail.login(FROM_EMAIL,FROM_PWD)
mail.select('inbox')

print "mail inbox selected"
```

**Fig. 5.5: Connection establishment for IMAP4**

To explain the terms mentioned in the code, Fig. 5.6 has been provided, which shows the values being set up for the server login.

```
ORG_EMAIL   = "@gmail.com"
FROM_EMAIL  = "kp.komalparikh1" + ORG_EMAIL
FROM_PWD    = "komalparikh1*"
SMTP_SERVER = "imap.gmail.com"
SMTP_PORT   = 587
```

**Fig. 5.6: Setting Values for imaplib connection**

For getting the latest email ID or the first email ID, etc., we can use imaplib. Then some Python code is used for extracting information from the Gmail reply by the user. We print information retrieved from the final e-mail sent by the user, i.e. the reply whether to allow or deny the person to enter into the premises. We need to extract the actual content from the message body and ignore the remaining clipped message. If the message body says “accept” or “allow” or “yes,” then a true value is passed to unlock the lock else it remains locked. Code for the same is explained in Fig. 5.7.

```
x.lower()
if "accept" in x or "allow" in x or "yes" in x:
    print "true"
else:
    print "false"
```

**Fig. 5.7: Printing True or False values based on the e-mail reply**

These Boolean values can then be accessed to send a power voltage to an electronic lock for unlocking (true) or locking, which means no action on it (false) in future work implementation.

The entire circuit working with the LED lights on as well as the Pi Camera light on is shown below.

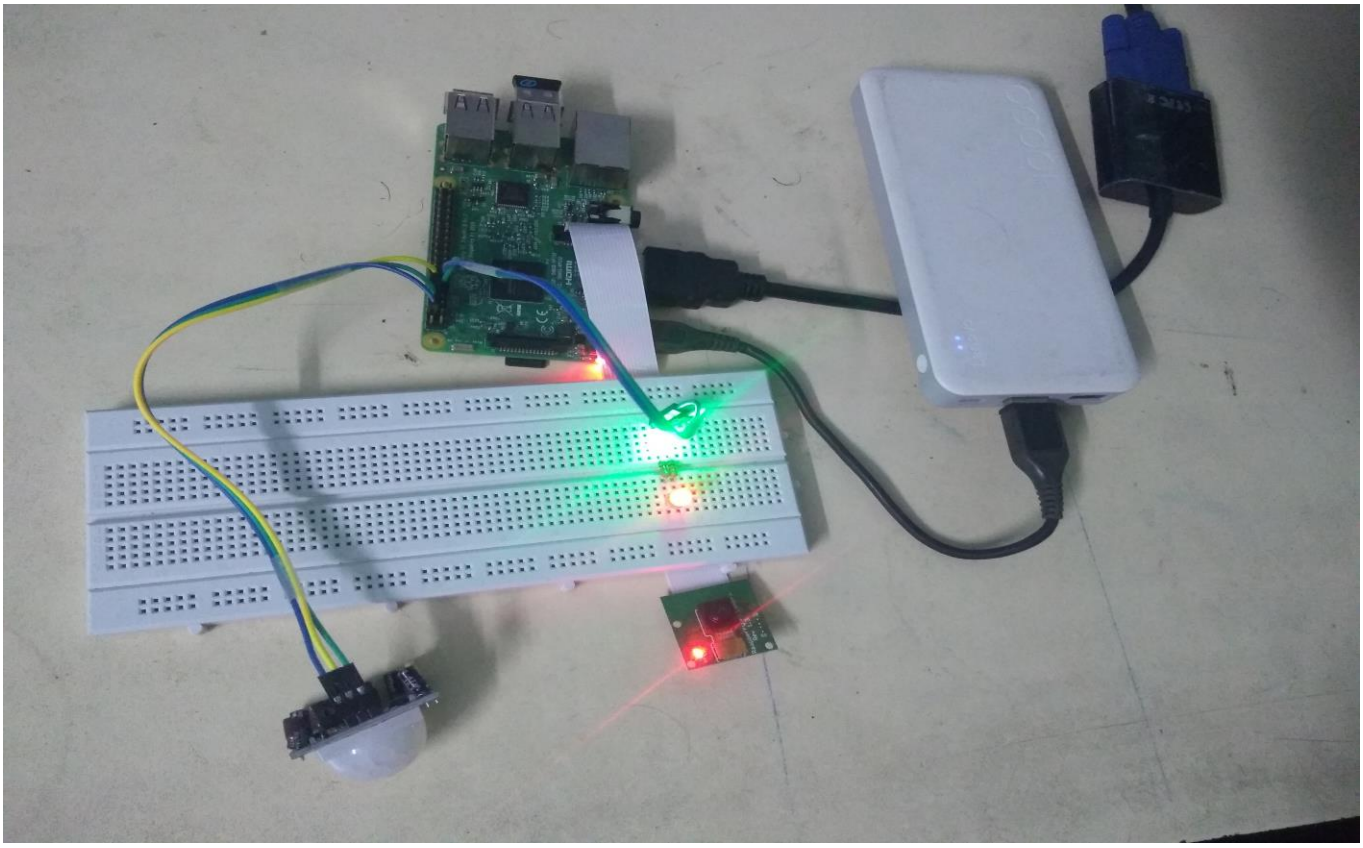

**Fig. 5.8: Working Circuit for Smart Security Camera**

## 5.2 Face Detection

This part of the system begins once the Pi camera is activated on sensing a movement in front of the PIR motion sensor. The camera then detects the face and the face attributes are extracted by the system. The system will match them with the photos of the faces stored in the database. The faces stored in the database have their name and identity saved in the database. If there is a match found, then the system will store the name with date and time of the specific day when the match was found. Also, an email is sent to the user informing him about

the same. This helps to keep updated information about people entering and leaving the place.

The whole process for training the recognizer can be divided in the following major steps:-

- The first step is to find the database of faces with several images for each individual person.
- The next step is to detect faces that are stored in the database images and use them to train the face recognizer.
- The last step is to test the face recognizer so that it is trained enough to recognize faces.

We also need to import the following modules:-

- **cv2** for face detection and recognition. OpenCV module contains all the necessary functions
- **os** used to maneuver with image and directory names. It is first used to extract the image names in the database directory. From these names, individual numbers are extracted, which will be used as respective labels for the face in that image.
- **Image** module from Python Imaging Library (PIL) to read the image in grayscale format because the dataset images are in gif format and as of now, OpenCV does not support gif format.
- **numpy** to store images in numpy arrays.

1. **Training Recognizer for Face Detection**: First, an image database has been created using a code for dataset creation which makes use of the module using Haarcascade's .xml file which is in-built in OpenCV module. The database contains faces of the users with whom the face on the camera can be matched. In each image, the individual has a different facial expression. As such, there will be several images for the first individual. The database will use these images of each individual to train the recognizer. The algorithm used in this program is Local Binary Patterns Histograms (LBPH). In LBPH algorithm, each image is analyzed independently. The LBPH method is simpler as it characterizes each image in the dataset locally. When a new unknown image is provided through camera, same analysis is performed on it and the result is compared with each of the images stored in the dataset. The algorithm works by characterizing the local patterns in each location in the image and thus it analyzes the image.

2. **Applying Local Binary Patterns Histograms Face Recognizer Algorithm**: The function in LBPH which prepares the training set has a function called get_images_and_labels which takes the absolute path to the image database as input argument and returns a tuple of two lists, one containing the detected faces and the other containing the corresponding label for that face. After preparing the training set, the get_images_and_labels function with the path of the database directory is passed. This path must be the absolute path. This function returns the features of the images and labels or captions of the images which will be used to train the face

recognizer afterwards. To perform the training, the function, FaceRecognizer.train is used. It requires two arguments – the first is the knowledge about features which in this case are the images of faces and second is the corresponding labels assigned to the images which in this case are the individual numbers that are extracted from the image names.

3. **Extracting Features from Images**: The first step is to detect the face in each image. As it gets the region of interest (ROI) containing the face in the image, it will use it for training the recognizer. For the purpose of face detection, the HaarCascade provided by OpenCV is used. For detecting the face, haarcascade_frontalface_default.xml is used. The cascade is loaded using the module called cv2. Then the function called CascadeClassifier takes the path to the cascade xml file where it is copied in the current working directory, to use the relative path.

4. **Displaying Result as Confidence**: As the detection of faces is done accordingly, the result of the detection is displayed in Python in a way where the confidence of detection is mentioned. The confidence tells with what accuracy the face of the person has been detected. The less the confidence found, the more the accuracy is.

5. **Experimental Results in Face Detection and Recognition**: Recognizer tests the result by using the images with .sad extension. It reads in grayscale

format and detects faces in it for all images in the list present in the database. Having the region of interest (ROI) containing the faces, the ROI is passed to the function called FaceRecognizer.predict function which will assign it a label. Here the confidence of the result taken into consideration. The label is an integer that is one of the individual numbers that is assigned to the faces earlier and it is stored in nbr_predicted. The more the value of confidence variable is, the less the recognizer has confidence in the recognition. A confidence value of 0.0 is a perfect recognition. The images below, i.e. Fig. 5.9, Fig. 5.10, Fig. 5.11 and Fig. 5.12 show the face being accurately detected and recognized.

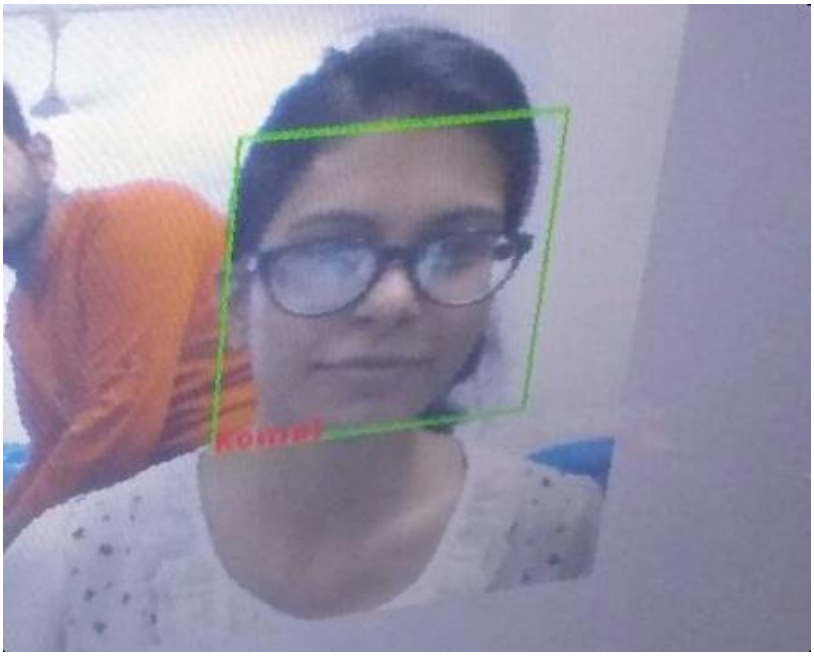


**Fig. 5.9: Face Detection and Recognition**

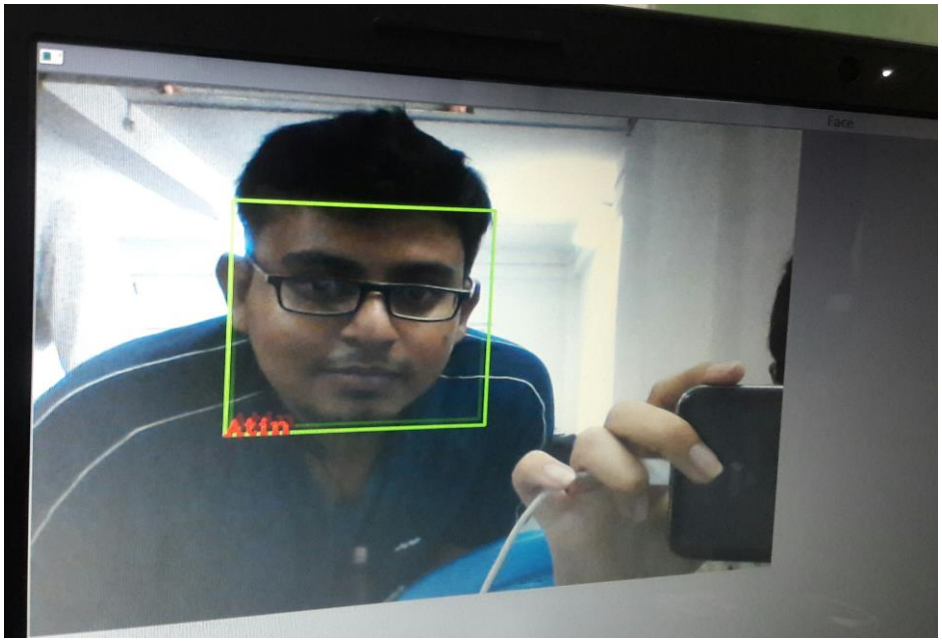


**Fig. 5.10: Face Detection running correctly**

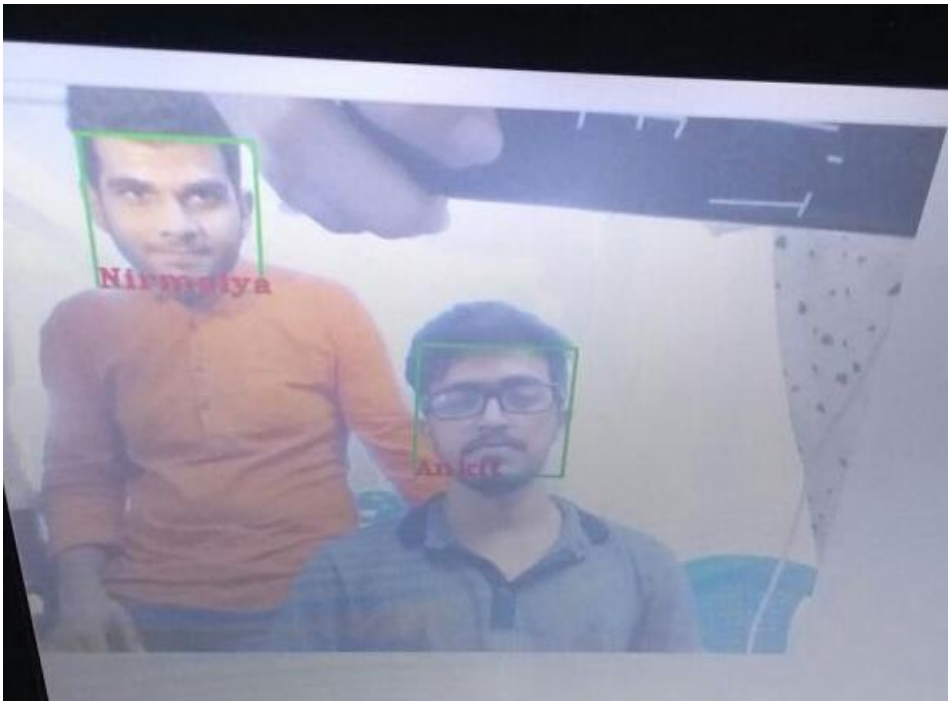

**Fig. 5.11: Face Detection with utmost Accuracy for multiple people**

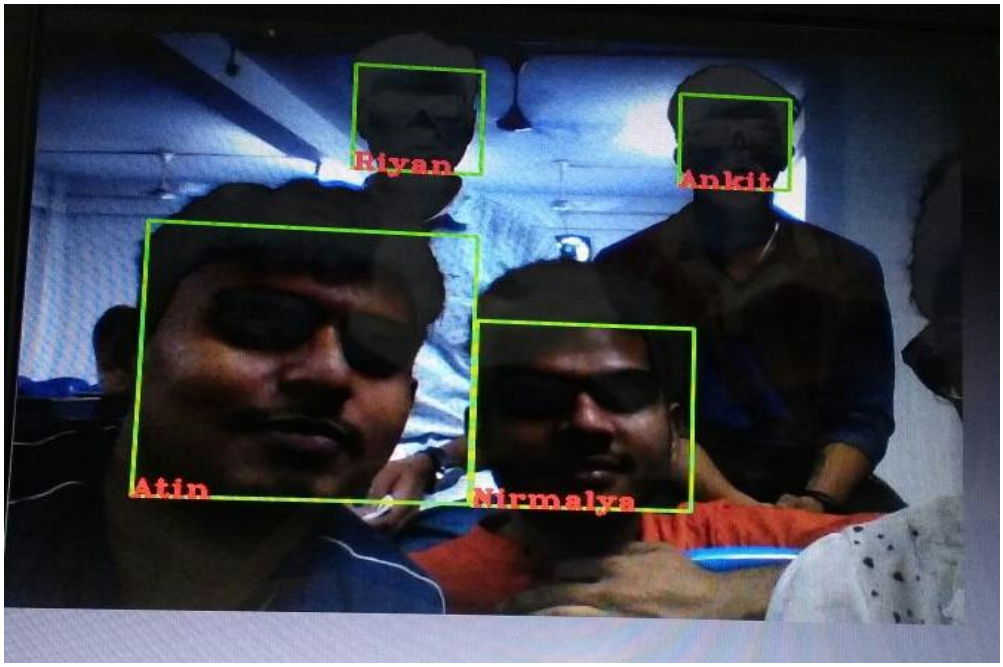


**Fig. 5.12: Face Detection working fine for all images (multiple faces) in the database with accuracy more than 90%**

# Chapter 06

## 6.1 Conclusion

In this project, a Smart Security System has been proposed based on the PIR sensor's camera activation in Raspberry Pi, face detection and face recognition, and e-mail confirmation for an extra level of security before the door is unlocked. The PIR sensor detects movement at the door which activated the camera to click images of the visitor. Thereafter, face detection and recognition part comes where the test result was able to provide a complete accuracy. Local Binary Patterns Histograms (LBPH) Face Recognizer Algorithm is run to identify the

person. The algorithm analyzes each face in the training set separately and independently and hence, is the optimum choice. Because of its accuracy and its capability to produce recognition results accurately, the extra level of security is not required in this case and the door is unlocked.

However, the images are still sent to the user or resident via Gmail using SMTP Protocol Client and his response is recorded via IMAP4 Protocol Client. This ensures double checking for a new user.

This system is a security solution for all masses due to its affordability and lack of expertise on the user end. So, instead of spending on a costly unit of surveillance, this is a better option as it can be regarded as a practical and “smart” solution keeping the budget in mind.

## 6.2 Future Work

The following sections form a basis of potential future inclusion in this project:-

- Electronic Lock – It is required for the purpose of putting this project to practical approach, an electronic lock is required which can show when the door will unlock itself and where it will not based on the voltage level passed to it which is based on the Boolean values generated by our code.
- Concept of digital signature – For the purposes of receiving a courier, if a signature is required, then it is possible via digital signature and the delivery person can get it printed on his address sheet. User authorization is provided with the help of Gmail using SMTP Protocol Client and IMAP4 Protocol Client. However, it will require a lot of hardware and the challenge is to cut down the cost of this marketable product.

- Animal detection – There can be all sorts of movements in front of the PIR sensor to activate the camera module. In future, we plan on implementing the detection of one's pets too, so that the user is aware of their whereabouts and can take care of certain situations.

- Video Doorbell – As the name suggests, the user can talk to the person at the door and deliver some useful information even if one is not at home. However, the main challenge again, is to cut down the cost of the whole system.

- Alarms – This is a cliché implementation, but a useful one and we plan on cutting down the costs on fire or smoke detectors installed along with the alarms in the Smart Security System. There can be different alarm tones for different situations like fire, burglary, etc.

- Smart Lights on the front door – The lights can be activated if the PIR sensor detects motion at the door.