**H.K.E SOCIETYS**

**POOJYA DODDAPPA APPA COLLEGE OF ENGINEERING**

**KALABURAGI- 585102**

**(An Autonomous Institution, Affiliated to VTU Belgaum, and approved By AICTE)**

****

***A PROJECT REPORT***

***ON***

**“SMART SECURITY SYSTEM”**

**SUBMITTED TO THE**

**POOJYA DODDAPPA APPA COLLEGE OF ENGINEERING KALABURAGI**

**(An Autonomous Institution, Affiliated to VTU Belgaum,and approved By AICTE)**

***BACHELOR OF ENGINEERING***

***IN***

***ELECTRONICS AND COMMUNICATION ENGINEERING***

**Submitted By:**

**ANKUSH MAHARAJ 3PD16EC018**

**ARAVIND J 3PD16EC021**

**MOHAMMED IKRAMUDDIN 3PD16EC052**

**MOHAMMED RAHIL SHIRAZ 3PD16EC053**

**Under the guidance of**

***Prof SRIDHAR K***

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**PDA COLLEGE OF ENGINEERING**

**KALABURAGI**

**2019-2020**

**H.K.E SOCIETYS**

**POOJYA DODDAPPA APPA COLLEGE OF ENGINEERING**

**KALABURAGI- 585102**

**(An Autonomous Institution, Affiliatedto VTU Belgaum, and approved By AICTE)**

****

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**CERTIFICATE**

**This is to certify that *ANKUSH MAHARAJ (3PD16EC018), ARAVIND J (3PD16EC021), MOHAMMED IKRAMUDDIN (3PD16EC052) & MOHAMMED RAHIL SHIRAZ (3PD16EC053)* of B.E VIII Semester of Electronics and communication engineering has satisfactorily completed seminar report on “SMART SECURITY SYSTEM” during the academic year 2019-2020 as prescribed by Vishveshwariah Technological University, Belgaum.**

**Principal Guide H.O.D**

**Dr.S.S.Hebbal Prof. Sridhar K Dr. Raju Yanamshetty**

**Examiners:**

**1. 2.**

Acknowledgement

We express our deep sense of gratitude and indebtedness to my esteemed institute “PDA COLLEGE OF ENGINEERING” KALABURAGI which has provided me an opportunity to fulfill the most cherished desire to reach my goal.

We express our foremost gratitude to our guide Prof. Sridhar K for his constant support and valuable guidance.

We are thankful to Dr. RAJU YANAMSHETTY Professor and Head of the department of Electronics and Communication engineering, for giving permission to carry out this seminar in the college.

We express our heartfelt thanks to the staff members of Electronics and Communication engineering department, who helped me in completion of this seminar directly and indirectly within the scheduled period.

Finally we are very much thankful to my beloved parents and to my dear friends.

**ANKUSH MAHARAJ 3PD16EC018**

**ARAVIND J 3PD16EC021**

**MOHAMMED IKRAMUDDIN 3PD16EC052**

**MOHAMMED RAHIL SHIRAZ 3PD16EC053**

**CONTENTS**

**SL No. PARTICULARS PAGE NO.**

**ABSTRACT 1**

1. **INTRODUCTION 2**
2. **LITERATURE SURVEY 3**
3. **PROBLEM STATEMENT 4**
4. **OBJECTIVES 5**
5. **BLOCK DIAGRAM OF SECURITY**

**SYSTEM 6**

1. **HARDWARE COMPONENTS 7**
2. **CIRCUIT CONNECTION 18**
3. **WORKING OF SYSTEM 19**
4. **SOFTWARE DESIGN 20**
5. **FLOWCHARTS 45**
6. **CODE IMPLEMENTATION 49**
7. **ADVANTAGES& APPLICATION 56**
8. **REFERENCES 57**

**ABSTRACT**

Today we are facing security issues in every aspect. So we have to resolve these issues by using updated technology. In this project, we are using the Face recognition module to capture human images and to compare with stored database images. If it matches with the authorized user then the system will unlock the door by an electromagnetic lock. The need for facial recognition system that is fast and accurate that continuously increasing which can detect intruders and restricts all unauthorized users from highly secured areas and aids in minimizing human error.

Face recognition is one of the most Secured System than biometric pattern recognition technique which is used in a large spectrum of applications. The time and accuracy factor is considered about the major problem which specifies the performance of automatic face recognition in real-time environments. Various solutions have been proposed using multicore systems. By considering the present challenge, this provides the complete architectural design and proposes an analysis for a real-time face recognition system with LBPH Algorithm. In this algorithm, it converts the image from color to greyscale image and divides into pixels and it will be allocated in a matrix form and those images will be stored in the database. If an image is detected then Raspberry Pi will send power to the motor driver unit then the electromagnetic lock will unlock the door and it will lock again when there is no power supply to that unit.

**INTRODUCTION**

In today’s world by using smart devices we are make our needs smart. By following trends and updates we have to consider and remove drawbacks in existing system and add more features and updates. Face detection system is more complex because of unstable characteristics. **Example:** let us consider glasses and beard will show some impact to detect the faces. So by considering the different angles and multiple images of faces and it will influence on detection process. The study of OpenCV and its inbuilt library functions helps to generate a code will do correct and authentic facial recognition system with new and more efficient use of hardware. Human body will identified as an input within environment by capturing image through the camera. The images will run through Raspberry pi3 and check with the stored data base, in this case, used a memory card.

This door unlocking system mainly uses facial recognition. The camera is used to detect the images and the images are sending to the database. If the image matches with the admin's image then the door is unlocked and if it does not match the image an alert mail is sent to prescribed mailhub. This will control the motor driver to lock and unlock the door.

**LITERATURE SURVEY**

Over the years various biometric features have been used for the purpose of human recognition like fingerprint, palm print, hand geometry, iris, speech, signature etc. The problem with all of these is they need active co-operation of person while face recognition is a process does not require active co-operation of a person so without instructing the person can recognize the person. So, face recognition is much more advantageous compared to the other biometrics. Face recognition has a high identification or recognition rate of greater than 90% for huge face databases with well-illumination conditions.

Face Recognition has become one of the most biometrics authentication techniques from the past few years. It is an interesting and successful application of Pattern recognition and Image analysis. Face recognitionsystem has two main tasks: **verification and identification.**

Face verification means a 1:1 match that compares a faceimages against a template face images whose identity beingclaimed. Face identification means a 1:N problem that compares a query face image against all image templates in a face database. Machine recognition of faces is graduallybecoming very important due to its wide range of commercial and law enforcement applications, which include forensic identification, access control, border surveillance and human interactions and availability of low cost recording devices.

**PROBLEM STATEMENT**

In the world of emerging technology, security became an essential component in day to day life. Information theft, lack of security and violation of privacy etc. are the essential components which are needed to be protected. Using smart secure systems for door lock and unlocking became popular nowadays. This is system is being adapted by many countries and first grade countries such as USA, Japan etc. already makes use of this system. Most security systems available in market are complicated & their installation is costly.

This project will implement a simple anti-theft security system for any remote area. This project also demonstrates the idea of port programming and PC based control system. Usage of computer software to control security system was widely accepted, and only authorized person will have access to the system which determines the room security. This feature can avoid unauthorized person to take control of the system or gain access to that remote area.

**OBJECTIVES**

* To develop programming and software using any available software to program the security system for the door with autolock feature.
* To study and describe how the Raspberry Pi can be interfaced with a motion detector and Pi camera.
* To study how a Raspberry Pi can be programmed so as to be able to send an email to a prescribed mailhub if any unauthorized person is detected,and the door is automatically locked.
* To develop and build a prototype of the surveillance system based on the Raspberry Pi SBC.

**BLOCK DIAGRAM OF SECURITY SYSTEM**

L298n

Motor

Driver

**Raspberry Pi**

**3B+ module**

Power

Supply

DC

Motor

Monitor

or

Laptop

Pi Camera

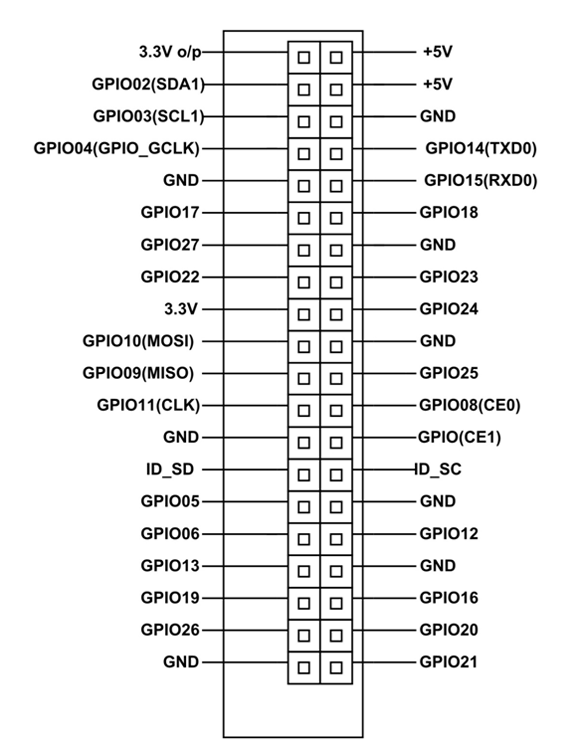
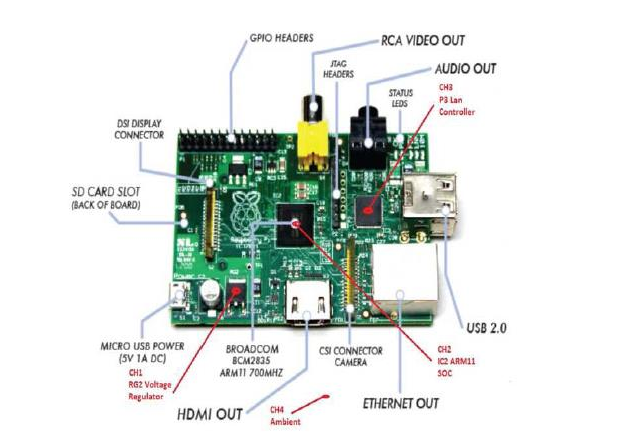
PIR Sensor

HC-SR501

**HARDWARE COMPONENTS**

1. Raspberry Pi(3B+ model)
2. Pi Camera
3. PIR Sensor(HC-SR501)
4. DC Motor
5. L298n Motor Driver
6. Monitor/Laptop

**Raspberry Pi:**

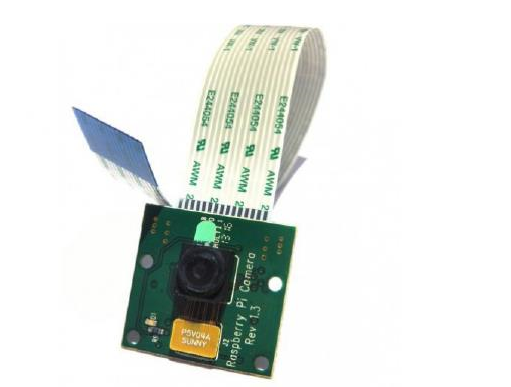


* The Raspberry Pi is a single board "credit card-sized" computer that can be used for electronics projects.
* It is also a fully-fledged computer and can do many things that a desktop PC does such as spreadsheets, word processing, browsing the Internet and playing games. It also plays high-definition video.
* Is able to run an entire Operating System (both Linux-based and Windows-based).
* In addition to computing power, later models of the Raspberry Pi have wireless networking and bluetooth connectivity built-in.
* The GPIO (General Purpose Input Output) pins allow sensors and other electronics and peripherals to be connected to the Raspberry Pi. These can be controlled programmatically by using a programming language such as Python.
* **Pin Configuration:** The Raspberry pi module is consisting of 40 pins. It has several GPIO pins, The GPIO (General Purpose Input Output) pins allow sensors and other electronics and peripherals to be connected to the Raspberry Pi.

**Specification of Raspberry Pi:**

* SoC: Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
* RAM: 1GB LPDDR2 SDRAM
* Networking: Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
* Storage: Micro-SD
* GPIO: 40-pin GPIO header, populated
* Ports: HDMI, 3.5mm analogue audio-video jack, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
* Dimensions: 82mm x 56mm x 19.5mm, 50g

**Pi Camera Module:**

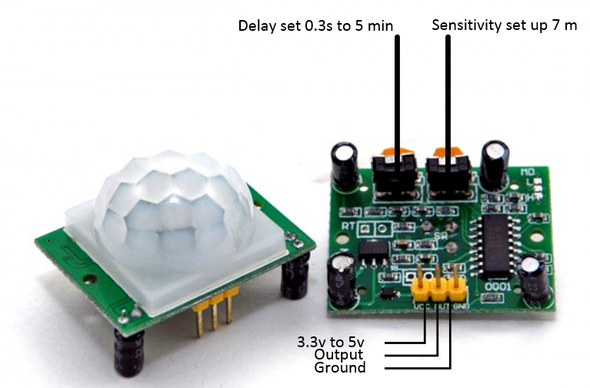


* The Camera Board on the Raspberry Pi is a small printed circuit board with a camera on it. The PCB is connected to a ribbon cable which connects to the Pi itself on its own port. The ribbon can be extendable.
* The camera on the board is very small (5MP camera). As for now it is the only Camera made specifically for the Pi therefore these specifications cannot be updated. Since it uses 250mA, externally powering the Pi should be sufficient enough for the camera.
* Specific configuration settings are required to initialize the camera plus a Python script to enable it to take pictures.

**Specifications of PI Camera:**

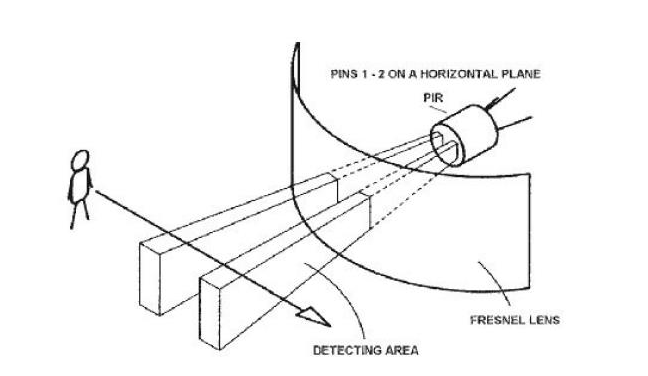
* Image Sensor: OmniVision OV5647
* Resolution: 5-megapixel
* Still picture resolution: 2592 x 1944 pixles.
* Video Modes: 1080p30, 720p60 & 640x480p60/90
* Connection to Raspberry Pi: 15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
* Temp range Operating: -20º to 60º Stable image: -20º to 60º
* Lens size: 1/4”
* Dimensions: 23.86 x 25 x 9mm
* Weight: 3g

**Passive Infrared Sensor(PIR):**



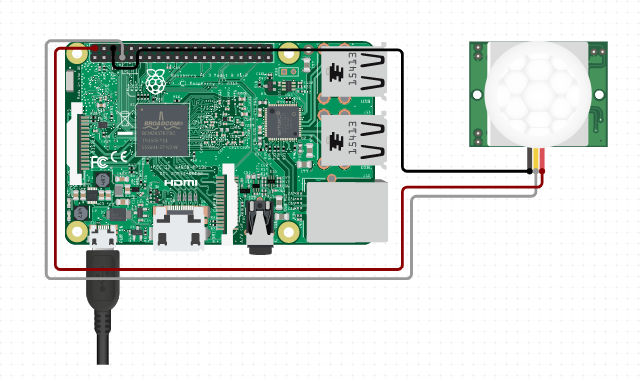
* A Passive Infrared Sensor(PIR) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view.
* They are most often used in PIR-based motion detectors.
* PIR sensors are commonly used in security alarms and automatic lighting applications.
* PIR sensors detect general movement, but do not give information on who or what moved.

**Principle Operation of a PIR Sensor:**



* An individual PIR sensor detects changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor.
* When an object, such as a human, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again.
* The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection.
* Objects of similar temperature but different surface characteristics may also have a different infrared emission pattern, and thus moving them with respect to the background may trigger the detector as well.

**Reading PIR Sensors:**



* The PIR acts as a digital output so all you need to do is listen for the pin to flip high (detected) or low (not detected).
* Power the PIR with 5V and connect ground to ground. Then connect the output to a digital pin i.e. the GPIO pin of the RPI device.
* A C/Python code can then be used to read a channel from the PIR sensor.

**Specifications of PIR Sensor:**

* Product Type: HC--SR501 Body Sensor Module
* Operating Voltage Range: 5 to 20V
* Quiescent Current: <50uA
* Level output: High 3.3 V /Low 0V
* Board Dimensions: 32mm\*24mm
* Operation Temp.: -15-+70 degrees
* Lens size sensor Diameter: 23mm(Default)

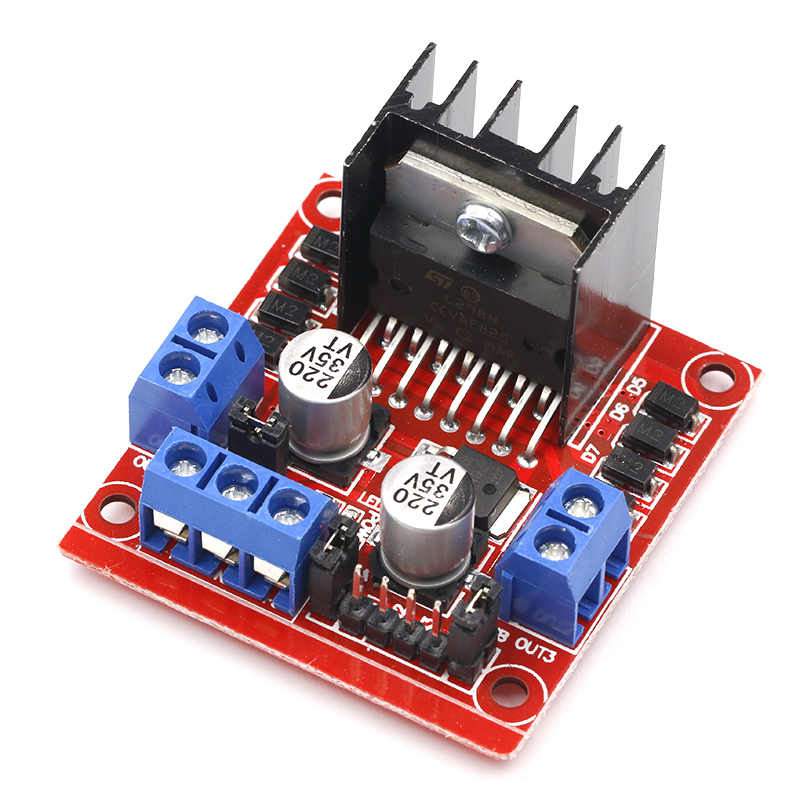
**DC Motor:**



**Features:**

* Low density: lightweight, low inertia.
* Capability to absorb shock and vibration as a result of elastic compliance.
* Ability to operate with minimum or no lubrication, due to inherent lubricity.
* The relatively low coefficient of friction.
* Operating Voltage(VDC): 3~12
* Shaft Length (mm): 8.5
* Shaft Diameter (mm): 5.5 (Double D-type)
* No Load Current: 40-180mA.
* Rated Speed(After Reduction): 300 RPM
* Rated Torque: 0.35 Kgcm

**L298n Motor Driver:**

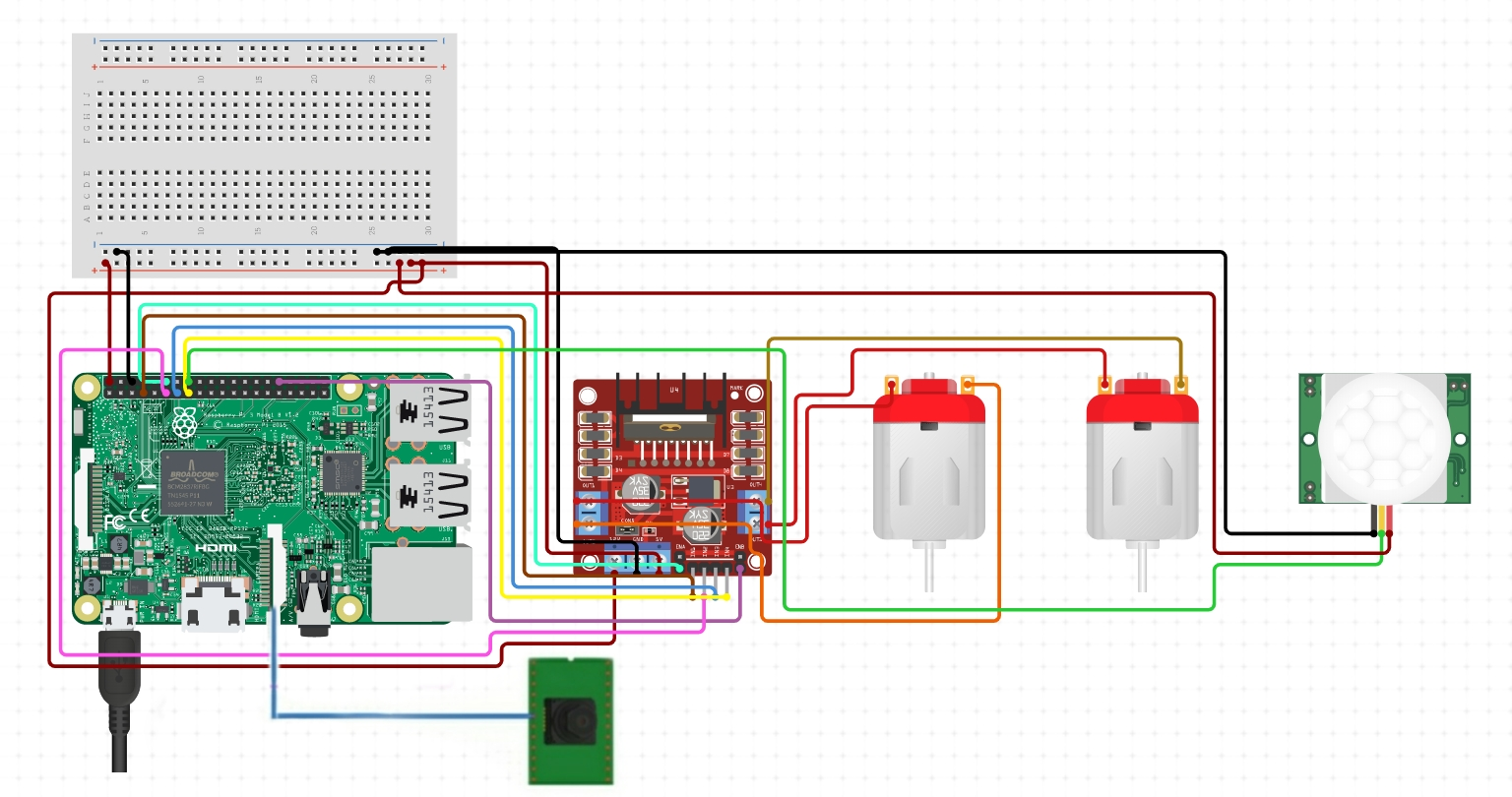


**Features:**

* Operating supply voltage up to 46V
* Total Dc current up to 4amp (each channel can carry upto 2amp)
* Low saturation voltage
* Over temperature protection
* Logical ”0” input voltage up to 1.5 v (high noise immunity)
* PTR connector for easy connection.

L98 Motor Driver Board It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepper motors. Two enable inputs are provided to enable or disable the device independently of the input signals. An additional supply input is provided so that the logic works at the lower level.

**CIRCUIT CONNECTION**

****

The Raspberry pi module is interfaced with the PIR sensor and the PI camera, the PIR sensor has 3 pins namely VCC(5V), GND, OUT. The VCC pin is connected to 5V pin of pi module, GND pin with GND of pi and the OUT pin of PIR is connected to any one of the GPIO pin of PI module.

The PI camera is connected to the PI module with the help of the ribbon like connecters to the CSI connector camera pins. The DC motor is connected to the Pi module with the help of its wires.

**Working:**

* Raspberry Pi is interfaced with Pi Camera module, PIR sensor and is given an internet connection. Firstly we create a database of authorised persons and store it in system. Now whenever a person is detected in the specified region the PIR sensor detects and sends a HIGH signal to the Pi.
* After receiving the signal the Pi camera is turned ON and starts capturing the images and then it compares the face of the person with the stored data in the database. If the face doesnot match with the authorised users then an alert Email containing the image of the intruder is sent using MIME with SMTP protocols to the specified mailhub& the door remains locked.
* If authorised user is recognised then the DC motor & L298n driver motor runs and opens the door.
* Recognition and sending of alert mail notification is done by using a Python script.

**SOFTWARE DESIGN**

In computer science, face recognition is basically the task of recognizing a person based on its facial image. It has become very popular in the last two decades, mainly because of the new methods developed and the high quality of the current videos/cameras.

Note that face recognition is different of face detection:

* **Face Detection**: it has the objective of finding the faces (location and size) in an image and probably extract them to be used by the face recognition algorithm.
* **Face Recognition**: with the facial images already extracted, cropped, resized and usually converted to grayscale, the face recognition algorithm is responsible for finding characteristics which best describe the image.

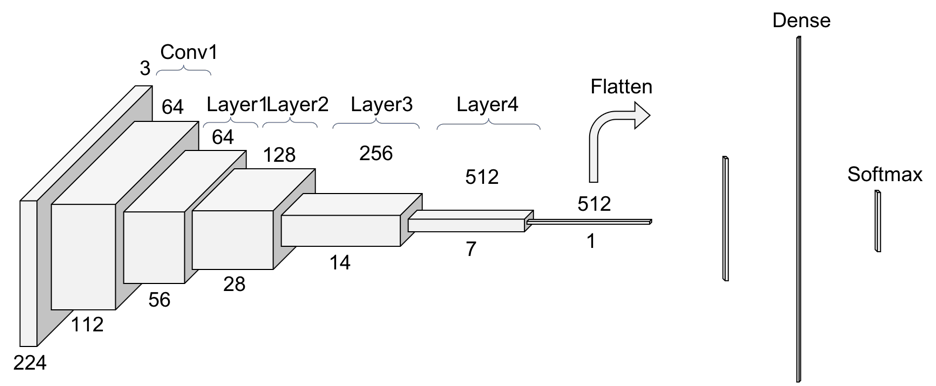
The face recognition systems can operate basically in two modes:

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

**FACE DETECTION**

**Convolutional Neural Networks:**

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.



There are four main operations in the ConvNet:

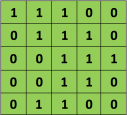
1. Convolution..
2. Non Linearity (ReLU).
3. Pooling or Sub Sampling.
4. Classification (Fully Connected Layer).

These operations are the basic building blocks of every Convolutional Neural Network. We will try to understand the intuition behind each of these operations below.

**Convolution Step:**

ConvNets derive their name from the “convolution” operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

Consider a **5 x 5 image** whose pixel values are only 0 and 1 (note that for a grayscale image, pixel values range from 0 to 255, the green matrix below is a special case where pixel values are only 0 and 1):

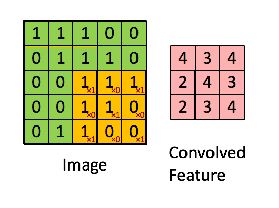


**Fig. Pixel values of a 5x5 image.**

Also, consider another **3 x 3 matrix** as shown below:

Screen Shot 2016-07-24 at 11.25.24 PM

Then, the Convolution of the 5 x 5 image and the 3 x 3 matrix can be computed as shown:



**Fig.: The Convolution operation. The output matrix is called Convolved Feature or Feature Map.**

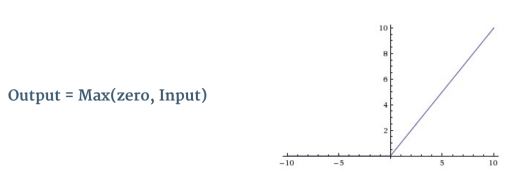
In CNN terminology, the 3×3 matrix is called a **‘filter‘** or **‘kernel’** or ‘feature detector’ and the matrix formed by sliding the filter over the image and computing the dot product is called the ‘Convolved Feature’ or **‘Activation Map’** or the **‘Feature Map‘**. It is important to note that filters acts as feature detectors from the original input image.

It is evident from the figure above that different values of the filter matrix will produce different Feature Maps for the same input image.

we can perform operations such as Edge Detection, Sharpen and Blur just by changing the numeric values of our filter matrix this means that different filters can detect different features from an image, for example edges, curves etc.

**Non-Linearity ReLu Activation:**

An additional operation called ReLU has been used after every Convolution operation. ReLU stands for **Rectified Linear Unit** and is a non-linear operation. Its output is given by:



**Fig. the ReLU operation**

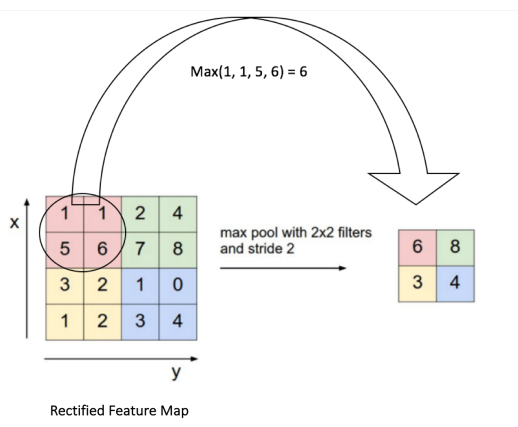
ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear. (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

**The Pooling Step:**

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: **Max, Average, Sum** etc.

In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window.



**Figure shows an example of Max Pooling operation on a Rectified Feature map (obtained after convolution + ReLU operation) by using a 2×2 window.**

We slide our 2 x 2 window by 2 cells (also called ‘stride’) and take the maximum value in each region. As shown in Figure, this reduces the dimensionality of our feature map.

So far we have seen how Convolution, ReLU and Pooling work. It is important to understand that these layers are the basic building blocks of any CNN.

Together these layers extract the useful features from the images, introduce non-linearity in our network and reduce feature dimension while aiming to make the features somewhat equivariant to scale and translation.

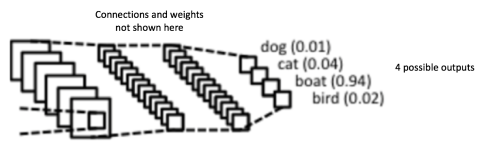
The output of the 2nd Pooling Layer acts as an input to the Fully Connected Layer.

**Fully Connected Layer:**

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer.The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

For example, the image classification task we set out to perform has four possible outputs as shown in Figure below

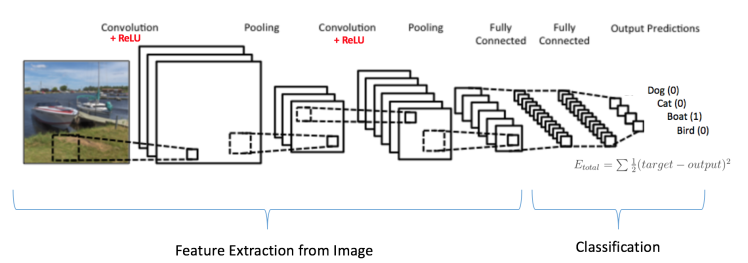


**Fig. Fully Connected Layer -each node is connected to every other node in the adjacent layer.**

**Putting it all together – Training using Backpropagation:**

As discussed above, the Convolution + Pooling layers act as Feature Extractors from the input image while Fully Connected layer acts as a classifier.

Taking an example of image of a boat: Note that in fig below, since the input image is a boat, the target probability is 1 for Boat class and 0 for other three classes, i.e.



**Fig.: Training the ConvNet**

The overall training process of the Convolution Network may be summarized as below:

1. We initialize all filters and parameters / weights with random values.
2. The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.

* Lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]
* Since weights are randomly assigned for the first training example, output probabilities are also random.

1. Calculate the total error at the output layer (summation over all 4 classes)

* Total Error = ∑ ½ (target probability – output probability) ²

1. Use Backpropagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.

* The weights are adjusted in proportion to their contribution to the total error.
* When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].
* This means that the network has learnt to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.
* Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

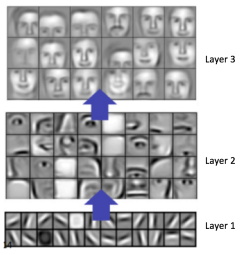
1. Repeat steps 2-4 with all images in the training set.

The above steps train the ConvNet – this essentially means that all the weights and parameters of the ConvNet have now been optimized to correctly classify images from the training set.

When a new (unseen) image is input into the ConvNet, the network would go through the forward propagation step and output a probability for each class. If our training set is large enough, the network will (hopefully) generalize well to new images and classify them into correct categories.

**Visualizing Convolutional Neural Networks:**

In general, the more convolution steps we have, the more complicated features our network will be able to learn to recognize. For example, in Image Classification a ConvNet may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to deter higher-level features, such as facial shapes in higher layers. This is demonstrated in Figure below –



**Fig. Learned features from a Convolutional Deep Network.**

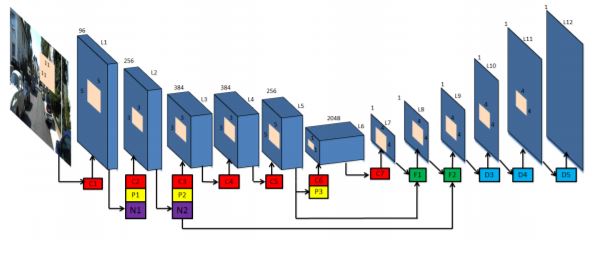
**Residual Neural Network (ResNet):**

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. **This model was the winner of ImageNet challenge in 2015**. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

However, increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient extremely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

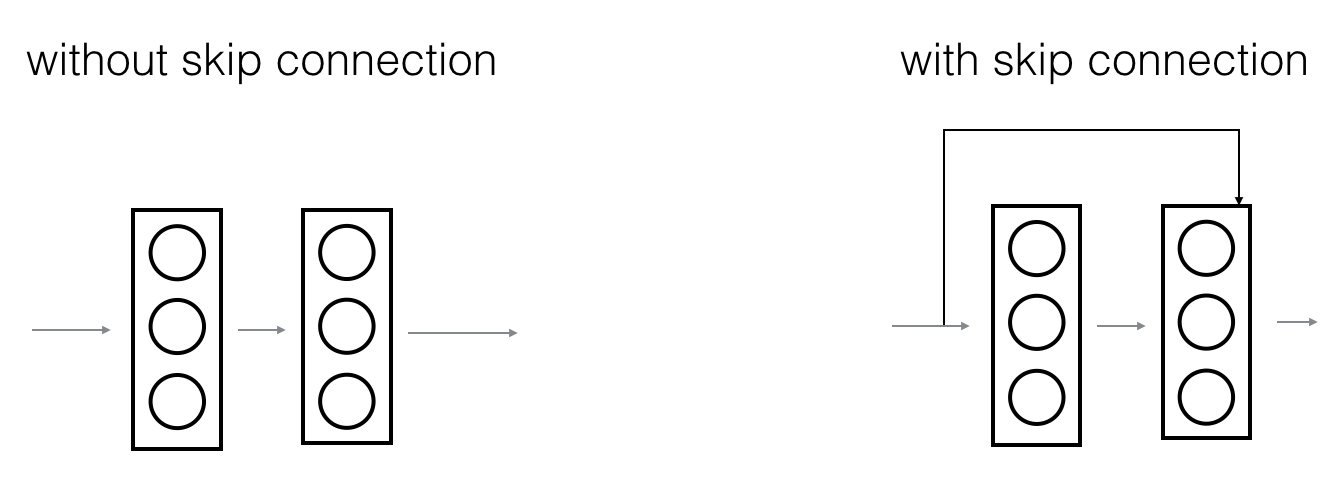
**Skip Connection - The Strength of ResNet:**

ResNet first introduced the concept of skip connection.Skip architecture as the name suggests skips some layer in the neural network and feeds the output of one layer as the input to the next layer as well as some other layer (instead of only the next layer ).



**Fig. A network architecture showing skip connections.**

There was some information that was captured in the initial layers and was required for reconstruction during the up-sampling done using the FCN layer. If we would not have used the skip architecture that information would have been lost. So the information that we had in the primary layers can be fed explicitly to the later layers using the skip architecture.



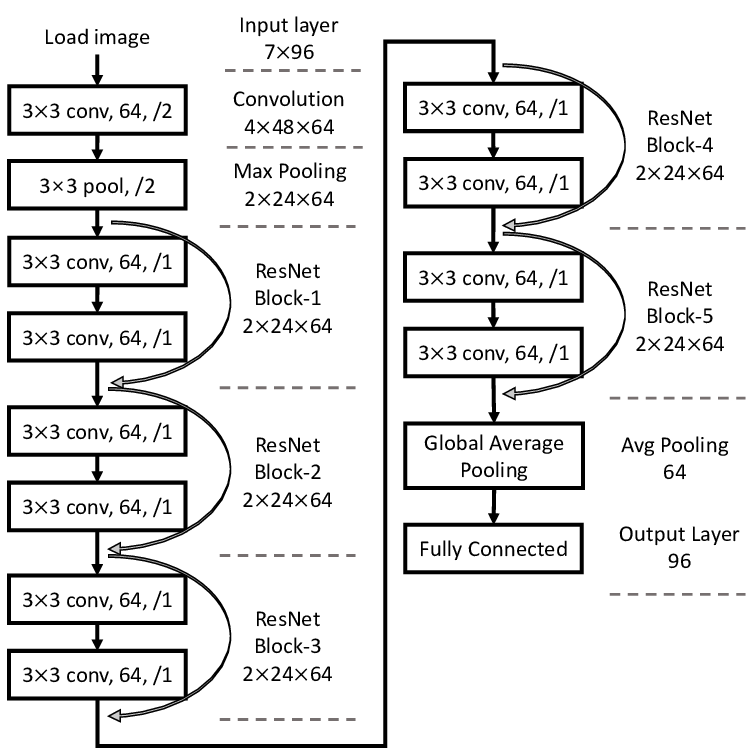
**Fig. CNN without skip connection Fig. CNN with skip connection.**

With the help of residual blocks we can increase the number of hidden layers as much as one desires without worrying about the vanishing / exploding gradients problem. Residual blocks enables the network to preserve what it had learnt previously (if there’s nothing to learn) by having an identity mapping weight function where the output is equal to the input, preserving what the neural network has learnt by not applying diminishing transformations or if the layer is able to learn something it’ll add on to what the network has learnt.

**Why do Skip Connections work?**

1. They mitigate the problem of vanishing gradient by allowing this alternate shortcut path for gradient to flow through.
2. They allow the model to learn an identity function which ensures that the higher layer will perform at least as good as the lower layer, and not worse.

**ResNet Architecture:**



The above figure represents a ResNet Architecture which takes an image of 224x224 as input and undergoes Convolution steps as discussed previously.

The ResNet-12 model consists of 12 blocks of convolution, Identity block and Max Pooling layers. The ResNet-12 has over 23 million trainable parameters.

Hence we use ResNet Architecture of CNN for face detection.

**FACE RECOGNITION**

**Local Binary Pattern Histogram(LBPH):**

It is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

Using the LBP combined with histograms we can represent the face images with a simple data vector.

**Training the Algorithm:**

First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.

**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 shows this procedure:



Based on the image above, let’s break it into several small steps so we can understand it easily:

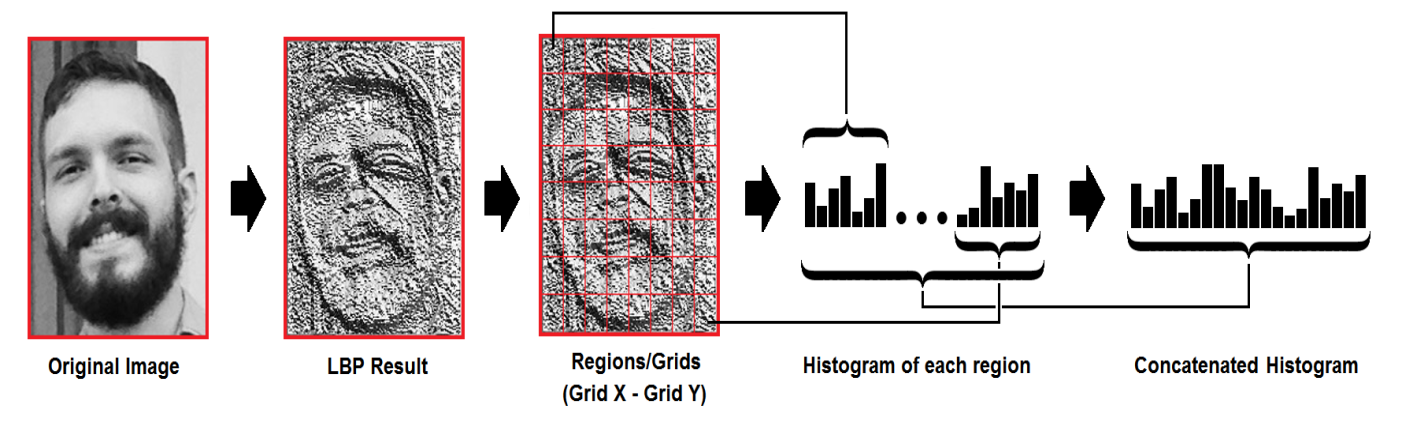
* Suppose we have a facial image in grayscale.
* We can get part of this image as a window of 3x3 pixels.
* It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
* Then, we need to take the central value of the matrix to be used as the threshold.
* This value will 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). Note: some authors 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.

**Parameters**: the LBPH uses 4 parameters:

1. **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
2. **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
3. **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
4. **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

**Extracting the Histograms**:

Now, using the image generated in the last step, 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:



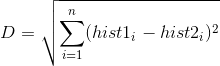
Based on the image above, we can extract the histogram of each region as follows:

* 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 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

**Performing the face recognition**:

In this step, the algorithm is 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 creates 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** (which is quite known) based on the following formula:



* 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 then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

**PYTHON PROGRAMMING LANGUAGE**

**Python** is an interpreted, high-level, general-purpose programming language. Created by **Guido van Rossum** and first released in **1991**, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming.

**Python Libraries:**

Python's large standard library, commonly cited as one of its greatest strengths,provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, faster mathematical calculations, Deep learning neural networks and for various computer vision tasks.

The following are the libraries we use for this project:

1. **OpenCV**:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

This library can be installed by running **pip install opencv-python** and **pip install opencv-contrib-python**.

1. **Numpy**:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

This library can be installed by running**pip install numpy.**

1. **OS:**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

1. **smtplib:**

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides 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 daemon.

1. **email and MIME handling package:**

The central component of the package is an “object model” that represents email messages. An application interacts with the package primarily through the object model interface defined in the message sub-module. The application can use this API to ask questions about an existing email, to construct a new email, or to add or remove email subcomponents that themselves use the same object model interface. That is, following the nature of email messages and their MIME subcomponents, the email object model is a tree structure of objects that all provide the EmailMessage API.

MIME types are used in MIME entities, as in email or HTTP traffic.It is useful at times to have information available about MIME types (or, inversely, about files). A MIME Type stores the known information about one MIME type.

**E-mail Notification:**

* In order to allow for email notifications to send, the OS needs a program that allows for emails to be sent. Simple Mail Transfer Protocol (SMTP) is a program that allows a system to deliver an email from a local computer to a mailhost. It does not receive mail but can send out mail. SMTP is ideal for situations where alerts are needed to be sent, therefore it is useful when sending notifications.
* A python script can be used to achieve this. It may just send a notification without image or can be modified to send an attachment along with the alarm message. For SMTP to support transmission of an attached file, Multipurpose Internet Mail Extension (MIME) is required.

**Simple Mail Transfer Protocol (SMTP):**

* SMTP is a push protocol and is used to send the mail whereas POP (post office protocol) or IMAP (internet message access protocol) are used to retrieve those mails at the receiver’s side.
* **Communication between sender and the receiver :**  
  The senders, user agent prepare the message and send it to the MTA. The MTA functioning is to transfer the mail across the network to the receivers MTA. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.
* **SENDING EMAIL:**  
  Mail is sent by a series of request and response messages between the client and a server. The message which is sent across consists of a header and the body. A null line is used to terminate the mail header. Everything which is after the null line is considered as the body of the message which is a sequence of ASCII characters. The message body contains the actual information read by the receipt.
* **RECEIVING EMAIL:**  
  The user agent at the server side checks the mailboxes at a particular time of intervals. If any information is received it informs the user about the mail. When the user tries to read the mail it displays a list of mails with a short description of each mail in the mailbox. By selecting any of the mail user can view its contents on the terminal.

**Multipurpose Internet Mail Extension (MIME) Protocol:**

* MIME is a kind of *add on or a supplementary protocol* which allows non-ASCII data to be sent through SMTP. It allows the users to exchange different kinds of data files on the Internet: audio, video, images, application programs as well.
* **Features of MIME –**

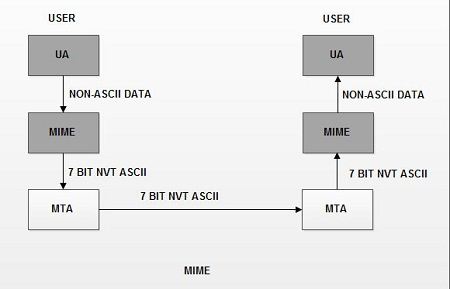
1. It is able to send multiple attachments with a single message.
2. Unlimited message length.
3. Binary attachments (executables, images, audio, or video files) which may be divided if needed.
4. MIME provided support for varying content types and multi-part messages.

**Why do we need MIME?**

Due to following Limitations of Simple Mail Transfer Protocol (SMTP) we use MIME:

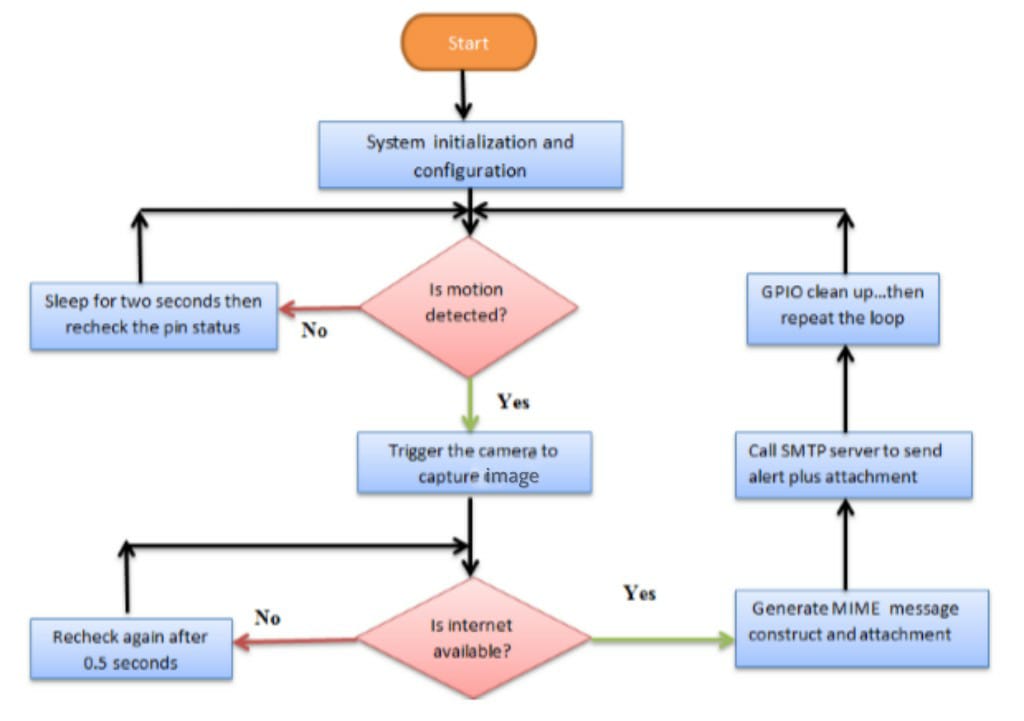
1. SMTP has a very simple structure
2. It’s simplicity however comes with a price as it only send messages in NVT 7-bit ASCII format.
3. It cannot be used for languages that do not support 7-bit ASCII format such as- French, German, Russian, Chinese and Japanese, etc. so it cannot be transmitted using SMTP. So, in order to make SMTP more broad we use MIME.
4. It cannot be used to send binary files or video or audio data.

**WORKING OF MIME**



* Suppose a user wants to send an email through user agent and it is in a non-ASCII format so there is a MIME protocol which converts it into 7-bit NVT ASCII format.
* Message is transferred through e-mail system to the other side in 7-bit format now MIME protocol again converts it back into non-ASCII code and now the user agent of receiver side reads it and then information is finally read by the receiver.
* MIME header is basically inserted at the beginning of any e-mail transfer.

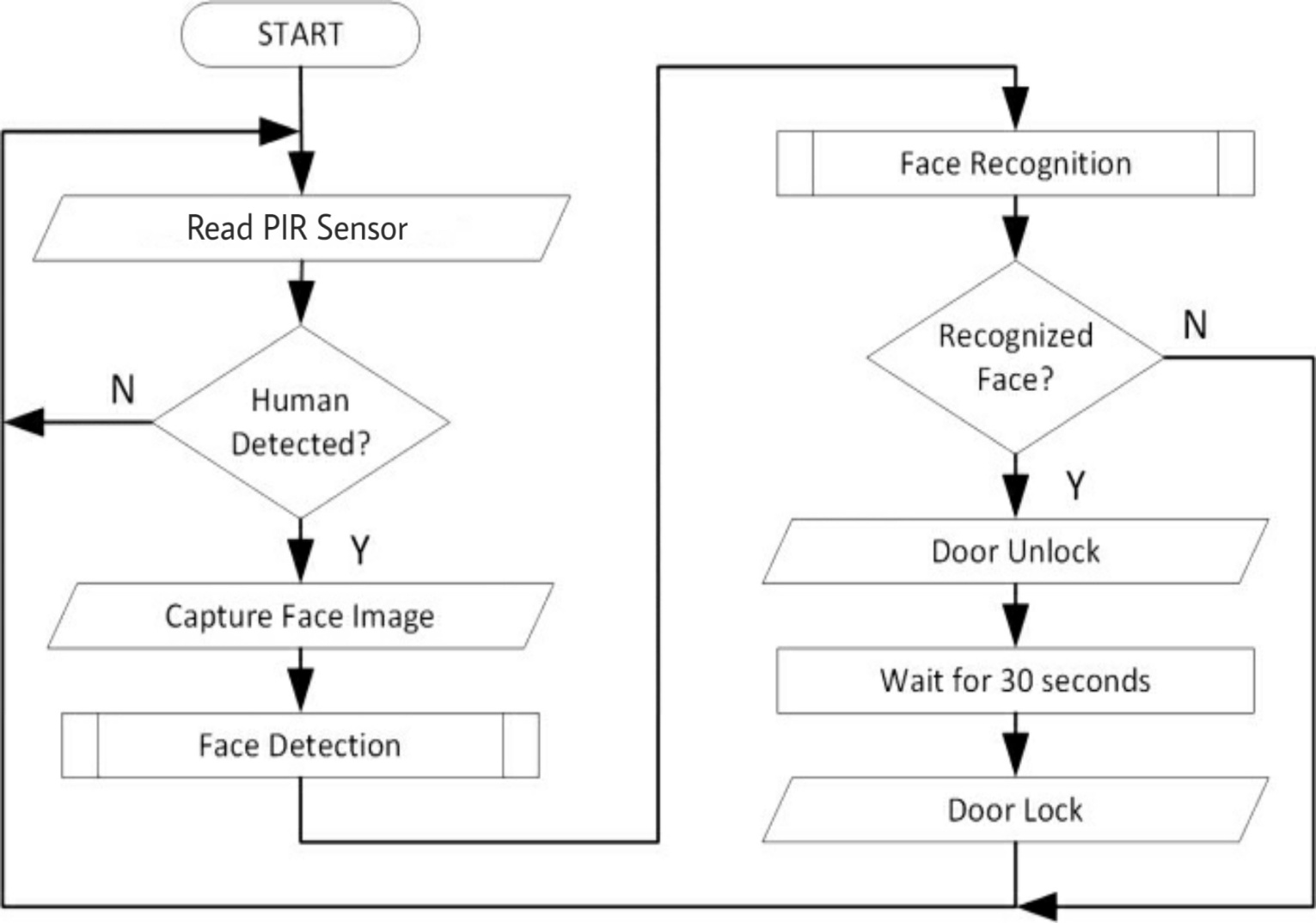
**FLOWCHART FOR OBJECT DETECTION & E-MAIL NOTIFICATION**



**ALGORITHM FOR OBJECT DETECTION & E-MAIL NOTIFICATION**

* Upon restart of the system, send out email with boot IP assigned to a mailhost.
* Check the status of the GPIO pin. If the pin is LOW, GPIO output pin 13 should remain LOW and the system is idle. Else if the pin suddenly goes HIGH. Interpret this as an interrupt event.
* While the value of the input GPIO pin is HIGH (interrupt event), set pin 13 to be HIGH. This instance blinks the LED. Call the function that starts the Pi Camera.
* Camera takes a 10 seconds video and save it in a file.
* The system checks whether the internet is enabled on the Raspberry Pi.
* If internet, send email to a prescribed mailhost. If no internet, wait for 5 seconds then check again.
* Reset the PIR sensor pin to LOW and recheck again the status after 2 seconds. This should return the program to the main loop.

**FLOWCHART FOR DOOR UNLOCKING**



**ALGORITHM FOR DOOR UNLOCKING**

1. Read PIR Sensor.

2. If any human is detected then capture image and run face recognition.

3. If captured face is recognized then unlock the door or else keep it locked and keep reading PIR sensor.

4. Wait for a specified time and lock the door.

5. Repeat.

**CODE IMPLEMENTATION**

1. **datasetcreate.py:**

This program is used for creating the dataset of known faces for training process of the LBPH algorithm.

Here the function detectface detects the faces on camera using CNN and saves those faces in a training folder with unique identity numbers for each person. Camera captures the entire frame but for this purpose our region of interest is the face hence we use detectface function.

|  |
| --- |
|  |
|  | import cv2  importnumpyas np |
|  | import time |
|  | importos |
|  |  |
|  | defdetectface(img,model): |
|  | (h,w) =img.shape[:2] |
|  | blob =cv2.dnn.blobFromImage(cv2.resize(img,(300,300)),  scalefactor=1.0,size=(300,300),mean=(104.0,177.0,123.0)) |
|  |  |
|  | model.setInput(blob) |
|  | detections =model.forward() |
|  | foriinrange(detections.shape[2]): |
|  | confidence = detections[0,0,i,2] |
|  | if confidence >0.5: |
|  | box = detections[0,0,i,3:7] \*np.array([w,h,w,h]) |
|  | (startX, startY, endX, endY) =box.astype("int") |
|  | face =img[startY:endY,startX:endX] |
|  | coordinates = [startX, startY, endX, endY] |
|  | returncoordinates,face |
|  |  |
|  | if \_\_name\_\_ =='\_\_main\_\_': |
|  | cam = cv2.VideoCapture(0) |
|  | count =0 |
|  | id =input('Enter the id:') |
|  | dirpath=os.path.dirname(\_\_file\_\_) |
|  | modelFile=f"{dirpath}/model/res10\_300x300\_ssd\_iter\_140000.caffemodel" |
|  | configFile=f"{dirpath}/model/deploy.prototxt.txt" |
|  | model = cv2.dnn.readNetFromCaffe(configFile, modelFile) |
|  | whileTrue: |
|  | \_,frame=cam.read() |
|  | try: |
|  | coordinates, face =detectface(frame,model) |
|  | except: |
|  | print('No face detected') |
|  | continue |
|  | writepath=os.path.join('DATA','user') |
|  | cv2.imwrite(f"{dirpath}/{writepath}{id}.{str(count)}.jpg", face) |
|  | startX,startY,endX,endY= [i-100foriin coordinates] |
|  | cv2.rectangle(frame,(startX+30, startY+30),(endX+30,endY+30),(0,255,0),2) |
|  | count +=1 |
|  | print(count) |
|  | cv2.imshow("FACE",frame) |
|  | if cv2.waitKey(1) & count==200: |
|  | break |
|  | time.sleep(0.2) |
|  | cam.release() |
|  | cv2.destroyAllWindows() |

1. **trainer.py:**

This program is used to train the dataset we create with their respective labels/id. The function getImageswithid takes the path of dataset folder as input and extracts the face features with their respective ids and returns them. Then the recognizer is trained and saved for further easy use.

|  |
| --- |
|  |
|  | import os  import cv2 |
|  | importnumpyas np |
|  | fromPILimportImage |
|  |  |
|  | recognizer =cv2.face.LBPHFaceRecognizer\_create() |
|  | path =r"DATA" |
|  | defgetImageswithid(path): |
|  | imagepaths= [os.path.join(path,f) for f inos.listdir(path)] |
|  | faces = [] |
|  | IDs= [] |
|  | forimagepathinimagepaths: |
|  | faceimg=Image.open(imagepath).convert('L') #convert to gray |
|  | facenp=np.array(faceimg,'uint8') |
|  | ID=int(os.path.split(imagepath)[-1].split('.')[1]) |
|  | faces.append(facenp) |
|  | print(ID) |
|  | IDs.append(ID) |
|  | cv2.imshow("training",facenp) |
|  | cv2.waitKey(1) |
|  | returnnp.array(IDs),faces |
|  |  |
|  | IDs, faces =getImageswithid(path) |
|  | recognizer.train(faces, IDs) |
|  | recognizer.save(r"recognizer\trainingdata.yml") |
|  | print('Saved') |

1. **raspFunctions.py:**

This program consists of various functions required for Door Unlocking system and this file will be imported into the main.py file, hence we use object oriented programming system to reduce the code complexity.

|  |
| --- |
|  |
|  | import os  import cv2 |
|  | import numpy as np |
|  | importRPi.GPIOasGPIO |
|  | import time |
|  | import datetime |
|  | import smtplib |
|  | from email.mime.multipart importMIMEMultipart |
|  | from email.mime.image importMIMEImage |
|  | from email.mime.text importMIMEText |
|  |  |
|  | classRaspFunctions: |
|  |  |
|  | record = {'Rahil':0,'Ankush':0,'Ikram':0,'Unknown':0} |
|  |  |
|  | dirpath =os.path.dirname(\_\_file\_\_) |
|  | modelFile =f"{dirpath}/model/res10\_300x300\_ssd\_iter\_140000.caffemodel" |
|  | configFile =f"{dirpath}/model/deploy.prototxt.txt" |
|  | model = cv2.dnn.readNetFromCaffe(configFile, modelFile) |
|  |  |
|  | rec = cv2.face.LBPHFaceRecognizer\_create() |
|  | rec.read(f"{dirpath}/recognizer/trainingdata.yml") |
|  |  |
|  | defsetGPIO(self): |
|  | GPIO.setwarnings(False) |
|  | GPIO.setmode(GPIO.BOARD) |
|  | GPIO.setup(11, GPIO.IN) #PIR |
|  | GPIO.setup(26, GPIO.OUT) #DC motor |
|  |  |
|  | defreadPIR(self): |
|  | self.detectedCount =0 |
|  | whileTrue: |
|  | i =GPIO.input(11) |
|  | if i ==0: |
|  | continue |
|  | elif i ==1: |
|  | self.detectedCount +=1 |
|  |  |
|  | if detectedCount >30: |
|  | returnTrue |
|  |  |
|  | defreadCamera(self): |
|  | self.cap = cv2.VideoCapture(0) |
|  | self.counts = {'Rahil':0, 'Ankush':0, 'Ikram':0, 'Unknown':0} |
|  | self.maxcount =0 |
|  | self.person =None |
|  | print('Detecting Face') |
|  |  |
|  | whileTrue: |
|  | \_, self.frame = self.cap.read() |
|  | try: |
|  | self.res, self.text, self.pts = self.detectface(self.frame) |
|  | except: |
|  | print('No face detected') |
|  | continue |
|  |  |
|  | self.counts[self.res] +=1 |
|  | self.personName =max(self.counts,key=self.counts.get) |
|  | self.maxcount = self.counts[self.personName] |
|  | if self.maxcount >30: |
|  | break |
|  | cv2.rectangle(self.frame,(self.pts[0], self.pts[1]),(self.pts[2],self.pts[3]),(0,255,0),2) |
|  | cv2.rectangle(self.frame,(self.pts[0], self.pts[1]),(self.pts[2],self.pts[1]+20),(0,255,0),cv2.FILLED) |
|  | cv2.putText(self.frame,self.res,(self.pts[0], self.pts[1]+12),cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, 0, 1) |
|  | cv2.imwrite("detected.jpg", self.frame) |
|  | self.cap.release() |
|  | self.Authenticate(self.personName) |
|  |  |
|  | defAuthenticate(self,personName): |
|  | if self.personName in ['Rahil','Ankush','Ikram']: |
|  | self.openDoor() |
|  | print(f'Welcome {self.personName}!!') |
|  | RaspFunctions.record[self.personName] +=1 |
|  | else: |
|  | print('Access Denied') |
|  | RaspFunctions.record['Unknown'] +=1 |
|  | self.closeDoor() |
|  | self.sendNotification(personName) |
|  |  |
|  | defopenDoor(self): |
|  | GPIO.output(26,1) |
|  |  |
|  | defcloseDoor(self): |
|  | GPIO.output(26,0) |
|  |  |
|  | defsendNotification(self,personName): |
|  | sender ='SENDER EMAIL' |
|  | receiver ='RECEIVER EMAIL' |
|  | password ='SENDER PASSWORD' |
|  |  |
|  | message =f"Unknown Person Detected at {datetime.datetime.now()}" |
|  | server =smtplib.SMTP\_SSL(host='smtp.gmail.com',port=465) |
|  | server.login(sender,password) |
|  |  |
|  | msg=MIMEMultipart() |
|  | msg['From'] = sender |
|  | msg['To'] = receiver |
|  | msg['Subject'] ='Message from Door Unlocking System' |
|  | msg.attach(MIMEImage(open(r'detected.jpg','rb').read(),name='detectedface')) |
|  | msg.attach(MIMEText(message,'plain')) |
|  |  |
|  | server.send\_message(msg) |
|  | print('Email sent') |
|  | server.quit() |
|  |  |
|  | defdetectface(self,img): |
|  | self.img=img |
|  | (h,w) =img.shape[:2] |
|  | blob = cv2.dnn.blobFromImage(cv2.resize(img,(300,300)),scalefactor=1.0,size=(300,300), |
|  | mean=(104.0,177.0,123.0)) |
|  | self.model.setInput(blob) |
|  | detections =self.model.forward() |
|  | foriinrange(detections.shape[2]): |
|  | confidence = detections[0,0,i,2] |
|  | if confidence >0.5: |
|  | box = detections[0,0,i,3:7] \*np.array([w,h,w,h]) |
|  | (startX, startY, endX, endY) =box.astype("int") |
|  | pts = (startX, startY, endX, endY) |
|  | face =img[startY:endY,startX:endX] |
|  | gray= cv2.cvtColor(face, cv2.COLOR\_BGR2GRAY) |
|  | res, text =self.recognizeface(gray) |
|  | return res, text, pts |
|  |  |
|  | defrecognizeface(self,face): |
|  | self.face= face |
|  | self.id =0 |
|  | self.id, self.sim=self.rec.predict(self.face) |
|  | ifself.sim<85: |
|  | if self.id ==1: |
|  | self.id ='Rahil' |
|  | elif self.id ==2: |
|  | self.id ='Ankush' |
|  | elif self.id ==3: |
|  | self.id ='Ikram' |
|  | else: |
|  | self.id ='Unknown' |
|  | text ='{} {:.2f}'.format(self.id,self.sim) |
|  | return self.id, text |

1. **main.py:**

Finally the main.py is used to integrate all the Raspberry Pi functions and import them in a single file. This is the file which will be running in real-time calling the Raspberry function within itself.

After each execution we get the Record of persons detected and the number of executions done.

|  |
| --- |
|  |
|  | from time import sleep |
|  | from raspFunctions import \* |
|  | defmain(): |
|  | raspberry =RaspFunctions() |
|  | raspberry.setGPIO() |
|  | ifraspberry.readPIR(): |
|  | raspberry.readCamera() |
|  | print(f"Record-->{RaspFunctions.record}") |
|  |  |
|  | if \_\_name\_\_ =='\_\_main\_\_': |
|  | num=0 |
|  | whileTrue: |
|  | main() |
|  | num+=1 |
|  | print(f'Number of executions={num}') |
|  | time.sleep(5) |

**ADVANTAGES**

1. It can be used to monitor private properties from wherever they are in the world.
2. Maintain records of authorised users.
3. Helps user to take instant right decisions whenever an intruder is found.
4. This comes at a low cost compared to CCTV.
5. Door locking and unlocking can be controlled from anywhere in the world.

**DISADVANTAGES**

1. The camera can be vulnerable i.e can be stolen/damaged.
2. Privacy is an issue.

**APPLICATIONS**

1. It provides security for houses, industries & educational institutes etc.
2. It can be used to keep track on the users.
3. It can be used to monitor any remote area.

**REFERENCES**

1. M. Peter and H. David, “Learn Raspberry Pi” Apress, 2012.
2. P. S. Dhake and B. Sumedha S., “Embedded Surveillance System Using PIR Sensor.,” vol. No. 02, no. 3, 2014.
3. J. D., “Real Time Embedded Network and SMS Alerting system,” Jun. 2014
4. Jigar M. Pandya, Devang Rathod, Jigna J. Jadav,” A Survey of

Face Recognition approach”, International Journal of EngineeringResearch and Applications (IJERA) ISSN: 2248-9622www.ijera.com Vol. 3, Issue 1, January -February 2013, pp.632-635.

1. Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of Statistical Learning - Data Mining, Inference, and Prediction. Berlin: Springer-Verlag.
2. Mitchell, T. (1997). Machine Learning. New York: Mc Graw-Hill.
3. Russel, S. and Norvig, P. (2003). Artifiical Intelligence: A Modern Approach. 2nd Edition. New York: Prentice-Hall.
4. Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. Neural Networks 61: 85-117.
5. Adrian Rosebrock (2018). Face detection with OpenCV and deep learning. www.pyimagesearch.com
6. Arjun Krishna Babu (2016). Send Emails Using Python.

freecodecamp.org