INTELLIGENT SECURITY SYSTEM

USING RASPBERRY PI AND ANDROID

Fairooz Zafar 13110033

Maisha Maliha 12210015

Moin Afnan 12221028

Supervisor: Professor Dr. Md. Adnan Kiber

Department of Electrical and Electronics Engineering

School of Engineering and Computer Science

BRAC University

Submitted on: 14th December2016

DECLARATION

We, hereby declare that this thesis project is completed on the basis of the knowledge we have acquired and implemented ourselves. Materials of work accomplished by others and the references gained from the official websites are mentioned by reference. This thesis, neither in whole or in part, has been previously submitted for any degree.

_____

Fairooz Zafar

_____

Maisha Maliha

_____

MoinAfnan

_____

Professor Dr. Md. Adnan Kiber

ACKNOWLEDGEMENT

This project was given to us by our thesis advisor, Dr. Md. Adnan Kiber, Professor, Department of Electrical and Electronics Engineering, University of Dhaka, after he had sat down in sessions to know what we wanted to achieve through this thesis. He finally understood that we were keen to increase our knowledge and also work on something that can be used in our daily lives and came up with this project for us.

We would like to thank the Almighty Allah (SWT) for giving us patience, curiosity, ability to work and learn.

In addition, we thank our parents for their support and help throughout.

We would like to show our gratitude towards our advisor, Dr. Md. Adnan Kiber, who has helped, supported and guided us throughout this project and kept us motivated to push our limits to achieve our goals.

We also want to thank the Acting Chairperson, Dr.Farhat M. Iftekharuddin, Department of EEE, BRAC University and all the Faculty Members and Technicians of the department who have helped us achieve our goal in this project. It would not have been possible without them.

Last but not the least, we want to thank Mr. ForhadNaim, Software Engineer- Android at Bangla Trac Communications Ltd, who has helped us with the basics of Android to come as far as we have today.

Contents

TABLE OF FIGURES

ABSTRACT

Keeping our homes and office spaces safe and secure from intruders has always been a priority in our lives and like everything else, adding technology in this aspect can make our existing security measure better and more efficient. For our thesis project, we are aiming to create an intelligent security system that will incorporate Raspberry Pi (RPi), Camera Module and Android operating system to make a real-time, user friendly and reliable form of security and surveillance system. The RPi uses Haar Cascade algorithm to extract facial features in given images to detect faces. With a machine learning approach along with Eigen faces algorithm, the program is trained to recognize the features of a face from images already fed to the system. The RPi uses opencv application to compare the new image with the images already stored in the SD Card for cross-referencing purposes. The images for cross-referencing can be copied to the SD Card with a computer or can also be uploaded via the Android application. If the system finds that the image does not match any image in the directory it uploads that image to the server set up in the RPi given that internet connection is available. On the other hand, if it matches, the RPi does not do anything. Once uploaded, the image can be viewed from the Android Application. The app will offer the user multiple options on notifications, image receiving/storing and disabling/enabling the service. By using this system, the user will get real time alert when there's activity within a predetermined area, check whether it is an intrusion or not and store photos for further investigations if required.

1 INTRODUCTION

The concept of facial recognition has been widely researched and developed since its conception in the 1960s, pioneered by Woody Bledsoe, Helen Chan Wolf, and Charles Bisson. The applications of facial recognition are abundant but its employment in the areas of biometric security has been emphasized on the most. At Super Bowl XXXV in January 2001, police in Tampa Bay, Florida used Visage facial recognition software to search for potential criminals and terrorists in attendance at the event. 19 people with minor criminal records were potentially identified. [1] Using this technology at ATM booths as a security measure is an area currently under development. Vast databases have been developed by law enforcement agencies and governments across the world for easier identification and registration purposes. For our thesis project, we have tried to take this large-scale system and compress it into a user and cost friendly system to be used in household or in small scale. This concept also has applications in other areas besides security and surveillance. This very system can be used to register attendance in classrooms or can also replace or assist in the use of ID cards for identification in universities or office spaces. If applied properly and used correctly, this design may prove to be an easy and cost friendly technological update or addition to similar existing systems.

1.1 MOTIVATION

The motivation of working on this project originated from the urge to increase the safety of the citizens of this country by using available technology. Additionally, there is always room for improvement in security and surveillance which can assist law enforcement in investigation processes, just like it is seen overseas. According to Numbeo, world's largest user contributed

database, the concern for homes being broken into and things getting stolen is 68.82 which is considered High. Implementing a system like this has the chances of deterring the number of breaking and entering offences and will definitely increase the rate of solved cases like this due to high probability of prompt action by user and easier identification of the perpetrators. Overall, combining previous knowledge with newly acquired skills over the course of the time given to assemble this project to create something that can be used to make the lives of people easier or safer was the crux behind this idea.

1.2 THESIS OUTLINE

Section 2, provides information on all the hardware used in developing this project. The information on models and specifications has been collected from verified websites of their respective manufacturers. Section 3 discusses the software used in writing and running the codes required in this system, most prominently Android studio. The project in its entirety is discussed in detail in section 4 along with diagrams. The functions and processes associated with integrating the hardware and software are separately described. The particular algorithms used in facial detection and recognition are explained in this section. The steps involved in the connection between Android Application and database are also explained in this section. A sample run of the system is presented in section 5. Other applications previously mentioned are elaborated in section 6. The total expense in setting up and running this system is tabulated in section 7. Section 8 draws a comparison between the common methods of surveillance/security in the country and this project. There are certain limitations in this design which may affect its performance or functioning. A few of those limitations and ways to solve those problems are

explained in section 9. Finally, possible improvements to this design and its applications are proposed in section 10.

2 HARDWARE INTRODUCTIONS

This project is a compilation of different hardware products and their related software components to run the hardware perfectly. These are listed and described below:

2.1 RASPBERRY PI

2.1.1 Overview

The Raspberry Pi (RPi) is a single board computer fitted with a processor, memory, input/output and other features commonly found on a functional computer. The RPi was developed by the Raspberry Pi foundation in the United Kingdom in 2012. Several models of the microcontroller have been released over the years differing in price and functionalities. All models feature a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on chip graphics processing unit (GPU, a VideoCore IV). The operating system and program memory are stored on Secure Digital (SD) Cards. USB, HDMi, Composite video output and 3.5mm audio jack is fitted on the board as well. Lower level output is provided by GPIO pins which support common protocols. Raspbian, a Debian-based Linux distribution and third party Ubuntu and Windows is provided by the foundation for download. The version used in this thesis project is Raspberry Pi 1 Model B+ released in 2014. All technical information on the Raspberry Pi and Camera Module is taken from the official website of Raspberry Pi [2].

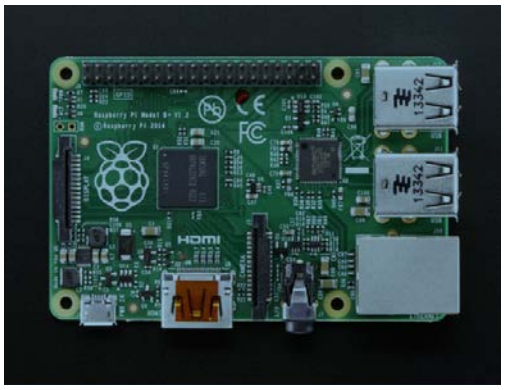


Fig. 01: Raspberry Pi Module

The Broadcom BCM2835 SoC used in the first-generation Raspberry Pi is somewhat equivalent to the chip used in first generation smart phones (its CPU is an older ARMv6 architecture), which includes a 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU), and RAM. It has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. [1] The following block diagram represents the hardware setup on the board:

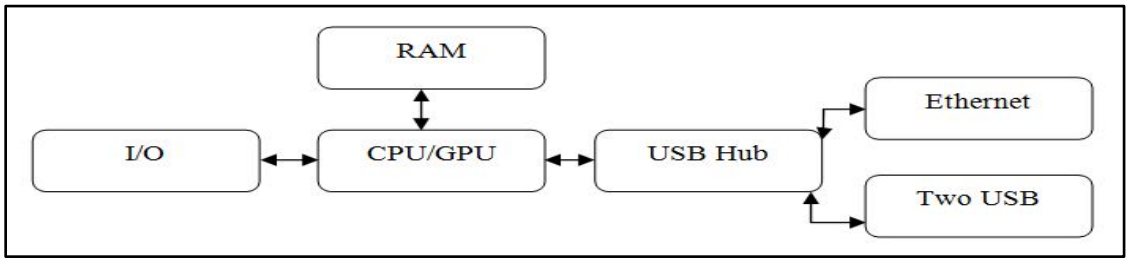


Fig. 02: Hardware on board

2.1.2 GPIO, Interfacing and other Ports

Raspberry Pi has got 40 general purpose input/output pins also known as GPIO. They are the means by which a Raspberry Pi can interact with other devices. These pins can be programmed in ways to suit the device it is been connected to. Each pin has its own designated work. The details of their functions are shown in (Fig. 03) below.

There are 4 USB 2.0 ports for connecting USB devices. More than 4 USB devices can also be connected using USB HUB.

An Ethernet port (RJ45 10/100 MBit/s) is fabricated on the board as well. It can be used to connect to the internet instead of using USB Wi-Fi dongle which results in faster internet speed. Other devices which have Ethernet connectivity can also be connected via this port.For video input, there is a 15-pin MIPI camera interface (CSI) connector through which a Raspberry Pi Camera (Fig.04) can be connected. For video output, there are numerous ports. HDMI port is used to connect to HDMI supported monitors, TRRS jack for composite video output and MIPI display interface (DSI) for raw LCD panel displays. Raspberry Pi does not have any audio line-in option. However, for audio output it has an independent analog output via a 3.5mm phone jack. The HDMI port also serves as an audio output for cables supporting both audio and video. There is a micro-SD card slot which when loaded with a memory card acts as a storage medium for the Pi and also the operating system needed to run the Raspberry Pi is stored in that card.

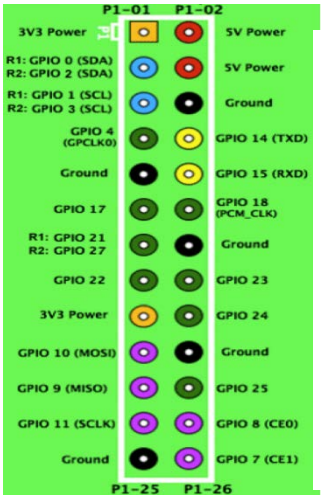


Fig. 03: Pin Configuration of Module

Fig. 04: Raspberry Pi Camera

2.1.4 Power

Raspberry Pi needs 5V, and 700mA to 2A current to operate. This is supplied by a micro USB cable or GPIO power connector. The input voltage is divided into portions of 3.3V, 2.5V and 1.8V to operate the different components. The B model typically uses 700-1000mA depending on the peripherals in use. An individual GPIO pin can tolerate up to 16mA safely while they can take 50mA safely while distributed across all the 26 GPIO pins on the board. The HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take any amount in between 100mA or over 1000mA.

The dimensions of this device are 85.60 mm × 56.5 mm and weigh about 45g.

2.2 RASPBERRY PI CAMERA MODULE

The camera being used in this project is Raspberry Pi camera module Version 1. This module is capable of capturing high definition video and still photographs. 1080p30, 720p60 and VGA90 video and still capture modes are available on this module. The connection from the module to the board is done by the CSI port. The camera is accessed through the MMAL and V4L APIs. [2]

3 SOFTWARE INTRODUCTIONS

The software components of this project are an indispensable part for our complete idea to work. To develop our application, we have used Android Studio, which is now the official development environment for the development of Android application. For the connection between the Application and the Raspberry Pi images we have used PHP as our connection script language, MySQL to serve as our database and Apache server for storing the images. The software configuration and characteristics are explained in detail in the coming sub-sections.

3.1 ANDROID STUDIO

Android Studio (AS) is the official Integrated Development Environment (IDE) for Android application development that is based on IntelliJ IDEA. IntelliJ IDEA is a JAVA Integrated Development Environment for software developments. This IDE has Git, JVM languages, Enterprise Frameworks, Mobile Development and Web Development integrated with it.

Android Studio has the features of IntelliJ IDEA along with other characteristics needed to develop the application. The highlights of the software are as below:

- AS has a flexible Gradle-based build system. Gradle uses a Domain- Specific Language (DSL) for the Android SDK (Software Development Kit) build system. In Gradle, declarative DSL is used to configure builds to support a wide variety of Android devices and App stores. With this, the developers have access to a single, authoritative build that powers both the Android Studio IDE and builds from the command-line.
- Unified environment applications for all Android Devices can be done. AS allows developers to build applications for a wide range of android version and devices. Starting with Froyo through Nougat along with the percentage of people this version of Android will reach, this enables developers to build apps for any of them. Also, the environment lets them be made for not only one brand of android, rather any android device.

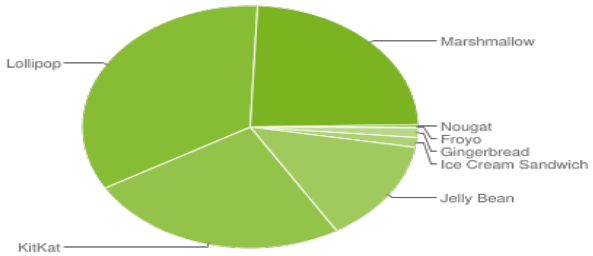


Fig. 05: User percentage of different Android Versions

Fig. 05 [4] refers to the percentage of people currently (2016) using all the versions of Android on their devices. This estimates the reach to the people which allows the developers to know its target pool. As an app made for the higher version of Android cannot be used or downloaded with devices which are running on devices on lower Android versions, this reach percentage is very important.

- Availability of Code Templates and GitHub integration is a great feature of AS which allows it to build common applications and also import sample code from existing projects available on GitHub [5]. GitHub is a Git repository hosting service which is web-based. Git is a version control system (VCS) required for software development. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features such as bug tracking, feature requests, task management, and wikis for every project [6].
- AS has newly included Lint tools to increase performance, usability, compatibility of version and other problems. Android Lint is a new tool that scans Android project sources for potential bugs which is available both as a command line tool and also integrated with IntelliJ [7].

Built-in support for Google Cloud Platform is another feature that makes it easier to integrate Google Cloud Messaging, presently known as Firebase Cloud Messaging [8]. This is a service that is used to send notifications to and from the android device to an application server, use Firebase Realtime Database, Authentication etc. as in Fig. 06 [8].

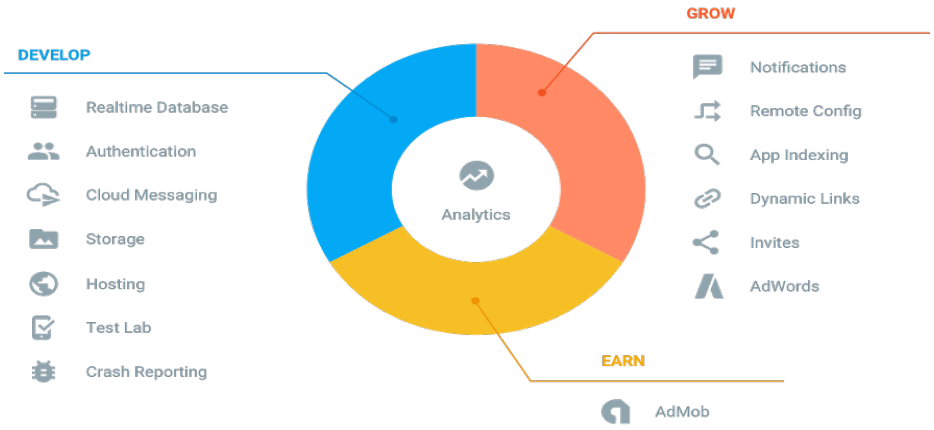


Fig. 06: Google Cloud Platform for App Development

For this project, we completed our application under the Android Operating system as in terms of sales Android has 65.9% of the market so this project can have more reach.

3.2 PHP

To connect the webserver where we have stored the images after the facial recognition to the android application, PHP (acronym for Hypertext Preprocessor) has been used. PHP is a widely used open source scripting language used mostly for web development and HTML [9]. PHP is the most efficient script as for this project, it controls the uploading from the application to the server and also downloading images uploaded by the Raspberry Pi. It also connects the server to our SQL (Structured Query Language) database, in this case we are using MySQL database.

3.3 MySQL DATABASE

Database is a collection of information that can be accessed by various end points and can be modified as required by the user. MySQL software is a database server and is a trademark of Oracle Corporation. With MySQL, the software used to manage the database is PhpMyAdmin, the most popular PHP applications and MySQL administration tools, with a large community of users and contributors [10].

3 INTEGRATED SYSTEM

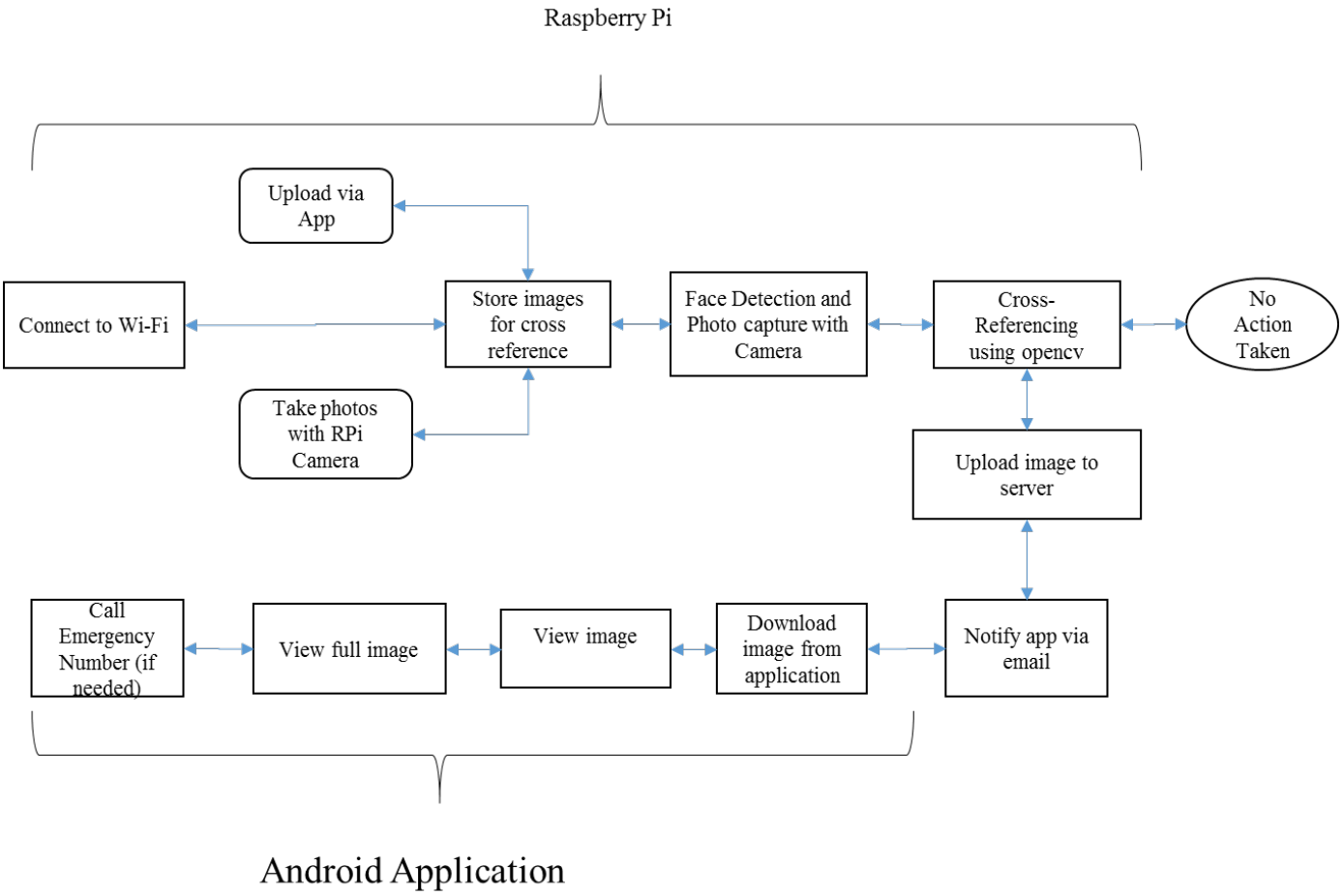


Fig. 07: Block Diagram for Integrated System

4.1 RASPBERRY PI, CAMERA MODULE and DATABASE

The working process of our project begins with detection of a face from the video feed of the camera module, which prompts it to take a still photograph of that instant. The photograph is processed by the Raspberry Pi and sent to a server if the face that is detected in the photograph doesn't match with any of the faces on the photograph stored in a previously loaded database. The is the first section of the project which required codes and libraries written in python programming language to be run on the Raspberry Pi.

4.1.1 Python and Python IDLE:

The codes used in this system for running the RPi and all of the image processing has been written in Python language in its entirety. Python is a widely used high level general purpose programming language. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Most Python implementations (including CPython) can function as a command line interpreter, for which the user enters statements sequentially and receives the results immediately.[11]Python IDLE is a development environment with abilities beyond that such as auto completion and syntax highlighting. This project required the use of NumPy and Open CV integrated with Python IDLE to function.

NumPy is the fundamental package for scientific computing with Python. It contains among other things: [12]

- a powerful N-dimensional array object

- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code

4.1.2 OpenCV

(Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The areas of Facial Recognition and Motion Understanding applications were the main focus of using OpenCV. OpenCV is written in C++ and its primary interface is in C++. There are bindings in Python, Java and MATLAB/OCTAVE. This system uses Open CV imported to Python IDLE to run on Raspberry Pi.

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import numpy
>>> print(numpy.__version__)
1.8.1
>>> import cv2
>>> print cv2.__version__
3.1.0
>>>
```

Fig. 08: Importing NumPy and OpenCV

The very first process in this system is object identification. The camera module is to detect a face from a running feed and capture a photograph in that instant. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images which is then used to detect objects in other images. For detecting a face, the program needs images with faces in them (positive images) and images without faces (negative images). The program then extracts the features from them. Each feature

is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle. [13]

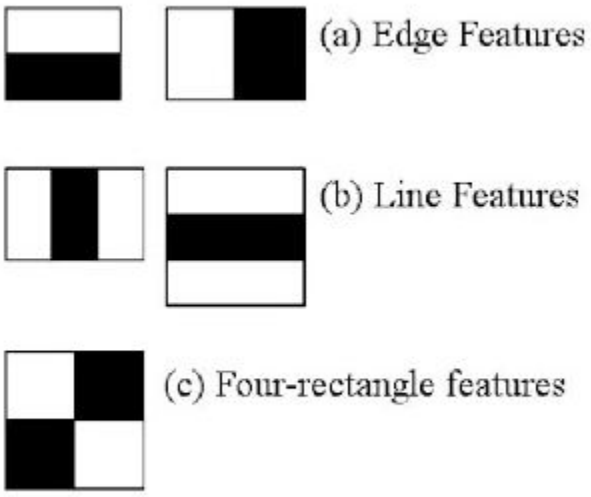


Fig. 09: Extractable Features [13]

Plenty of features are calculated by using the possible sizes and location of each kernel. This is done by using integral images which simplifies the calculation of pixels to an operation that requires only four pixels. Among the huge number of features that can be calculated, not all of them are useful for detection of objects (faces) in every case. The most relevant features need to be extracted from the collection. This is done by applying all the features to all of the training images. The features with the least error rate are chosen which means they are the features that best classifies the face and non-face images. Even after further classification, the final number of features that is needed to detect a face in an image is still very large which makes the process time consuming. However, a solution to this problem is possible because in an image, most of the image region is non-face region. For this, the concept of **Cascade of Classifiers** has been introduced. The features are grouped into different stages of classifiers and applied one-by-one on the window. If the first stage is a failure, the window is discarded entirely. Remaining

features on it are not taken into consideration. If it passes, the remaining stages of features are applied and the process is completed. The window which passes all stages is a face region. The steps to complete this process in Open CV are simplified in the following block diagram:

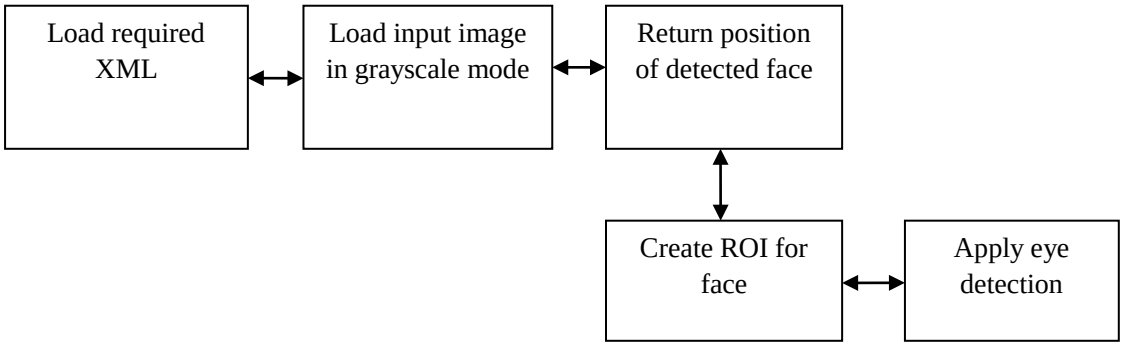


Fig. 10: Haar Cascade process

Once a face region is detected, the image is immediately passed on to the next step of the process to identify if the face detected is that of a permissible individual or an intruder.

Facial Recognition is a computer application or process which is capable of identifying a person from an image or video feed. This is achieved by cross examining a given picture with an already available image on a database. The FaceRecognizer class for face recognition that comes with OpenCV includes the following algorithms:

- Eigenfaces (see createEigenFaceRecognizer())
- Fisherfaces (see createFisherFaceRecognizer())
- Local Binary Patterns Histograms (see createLBPHFaceRecognizer())

[14] The algorithm used for this system is the Eigenfaces method. The Eigenfaces method described took a holistic approach to face recognition: A facial image is a point from a high-dimensional image space and a lower-dimensional representation is found, where classification becomeseasy. The lower-dimensional subspace is found with Principal Component Analysis,

which identifies the axes with maximum variance. While this kind of transformation is optimal from a reconstruction standpoint, it doesn't take any class labels into account. Imagine a situation where the variance is generated from external sources, let it be light. The axes with maximum variance do not necessarily contain any discriminative information at all, hence a classification becomes impossible. So, a class-specific projection with a Linear Discriminant Analysis was applied toface recognition. The basic idea is to minimize the variance within a class, while maximizing the variance between the classes at the same time. [14]

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a random vector with observations $x_i \in R^d$.

1. Compute the mean μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. Compute the Covariance Matrix *S*

$$S = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T,$$

3. Compute the eigenvalues λ_i and eigenvectors v_i of S

$$Sv_i = \lambda_i v_i, i = 1, 2, \ldots, n$$

4. Order the eigenvectors descending by their eigenvalue. The k principal components are the eigenvectors corresponding to the $k$largest eigenvalues.

The k principal components of the observed vector x are then given by:

$$y = W^T(x - \mu)$$

where $W = (v_1, v_2, \ldots, v_k)$.

The reconstruction from the PCA basis is given by:

$$x = Wy + \mu$$

where $W = (v_1, v_2, \ldots, v_k)$. [4]

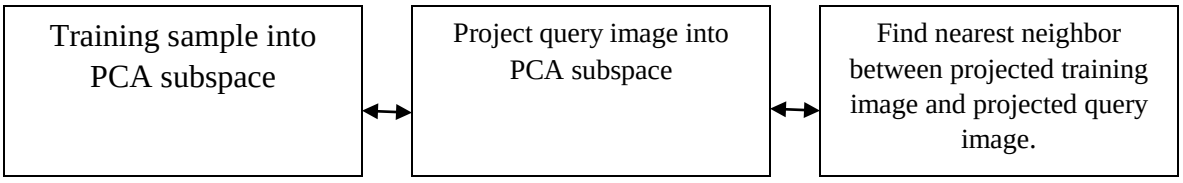
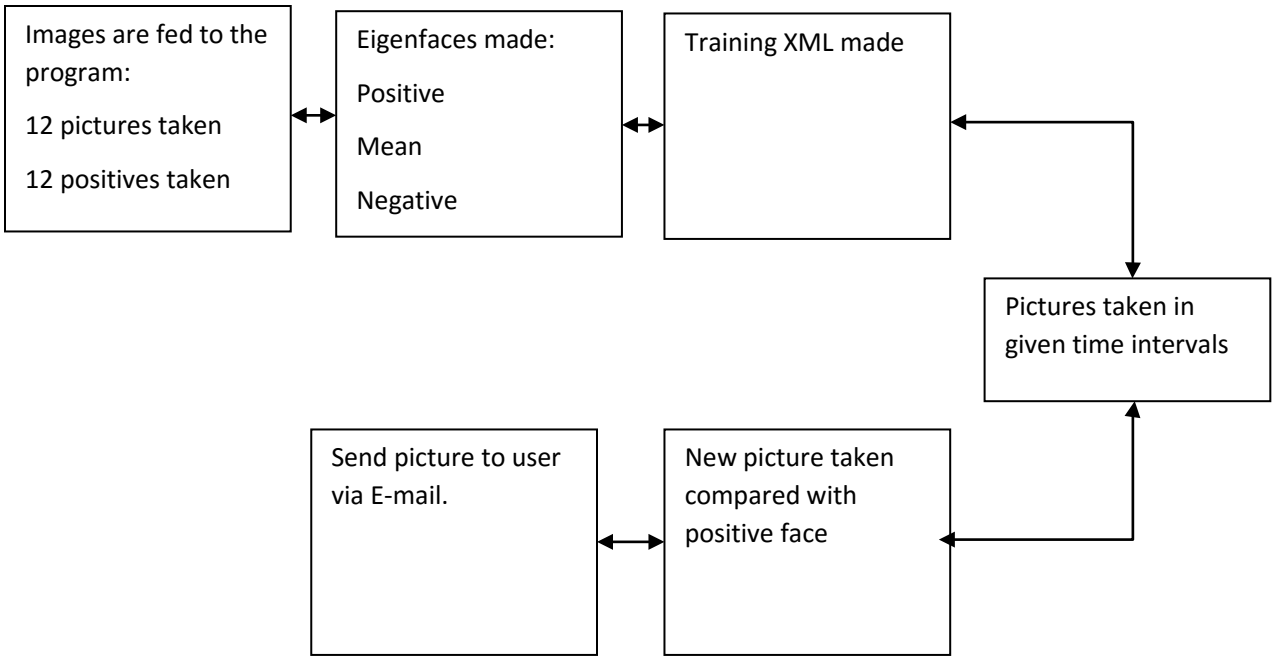


Fig. 11: Eigenface process

Eigenface in OpenCV:



At this point, the system knows if there is an individual who has not been given permission to enter is present in its area. The photo is stored in the server which sends a notification to the user

stating the present situation. The processes involved in that step are explained in details further in the next subsection.

4.2 ANDROID and DATABASE

After the Raspberry Pi and Camera Module has completed its work, comes the part of the Android Application we are using so that the user gets real time access to the image information in the database. This is done in a number of steps. Other than the RPi uploading image to the database, the directory to which the Opencv in the RPi is referring to compare the new images taken, can be uploaded using the application. The application allows the user to take a picture from the gallery and upload it the referencing directory. The directory to which the RPi is uploading the images are connected to the application via PHP which enables the app to import or download the images available in the database, given that internet connection is available. The following sections contain the complete detail on how the connections are made and the process for the application working.

4.2.1 Server Side Connections:

Database Table Set Up: In the database, a table was created with the structure of ‘id’, ‘image’ and ‘time’ in the rows. The ‘id’ (which is an integer will a maximum of 11 characters) is set to be in auto increment so whenever a new image, the ‘id’ can increase automatically and stands to be a primary key for unique identification of each image. The ‘image’ is set to be at variable characters (varchar 200) which contains the URL to the image. The original image gets stored in a folder on the server and the URL in the database. The ‘time’ row contains the Time

Stamp of each of the uploaded images and is in sync with the server time. This server time can be altered as needed in case the user is at a different time zone. The database setup is completed here after which the step moves to PHP scripting.

PHP Scripts: To make the connection between the database to server and server to android application, we need a total of three scripts that will complete the connecting to database, upload and downloading of images.

Connection Script: This script is named to be dbConnect.php in which the database information where the host name (Php MyAdmin URL for the server), username and password (to login) and the database name in which the table is made are introduced with the 'define' command. The make connection and the 'die' if unable to connect command is given for the script to work.

Image Upload Script: The script below handles uploads and is named as upload.php. It will store the images sent to a directory named 'uploads', which has been already created on the server we are using. This script calls the 'POST' Method to post an image according to the given code:

Pseudo Code:

```
1. Server 'REQUEST_METHOD'='POST';
2. Post 'image';
3. Connect to dbConnect.php;
4. While loop for getting 'id'
                end loop
5. Set upload directory
6. Sets URL of upload directory
7. Insert image into *table name*,    *row name* and *image file*;
8. Echo if image upload successful
                end method
9. Else, echo 'error'
```

Image Download Script: In this PHP script, we use JSON to import the image URLs from the database. The following code works to accomplish this:

Pseudo Code:

1. Connects to dbConnect.php
2. Selects image from *table name*
3. Stores result in an array
4. While loop starts
   Pushes array
   Imports image URL
   End loop
5. Echo json_encode for result
6. Conncetion closed

This completes our server side scripting and named as getAllImages.php. All the three scripts are now uploaded to the server where the initial ‘uploads’ directory was created and the URLs of the scripts are noted as they will be further introduced in our Application so that it can access the scripts and the directory.

4.2.2 Application Side: The Android Application created for this project is fairly simple and comprises of only the things necessary for the purpose of uploading to and downloading from the database. The application contains various methods, intents and objects needed to complete the task. In this paper, we have described our application as the buttons are placed on the first page (activity of the application). This application uses various Intent Objects to call on various actions required to complete the needed actions by the buttons. Intent Object is an abstract description of an operation to be performed. This object can be used to do multiple actions like starting an activity, sending broadcast messages and even communicate with a background

service [15]. The following image (Fig. 07) shows the interface of the application and the buttons that are explained below:

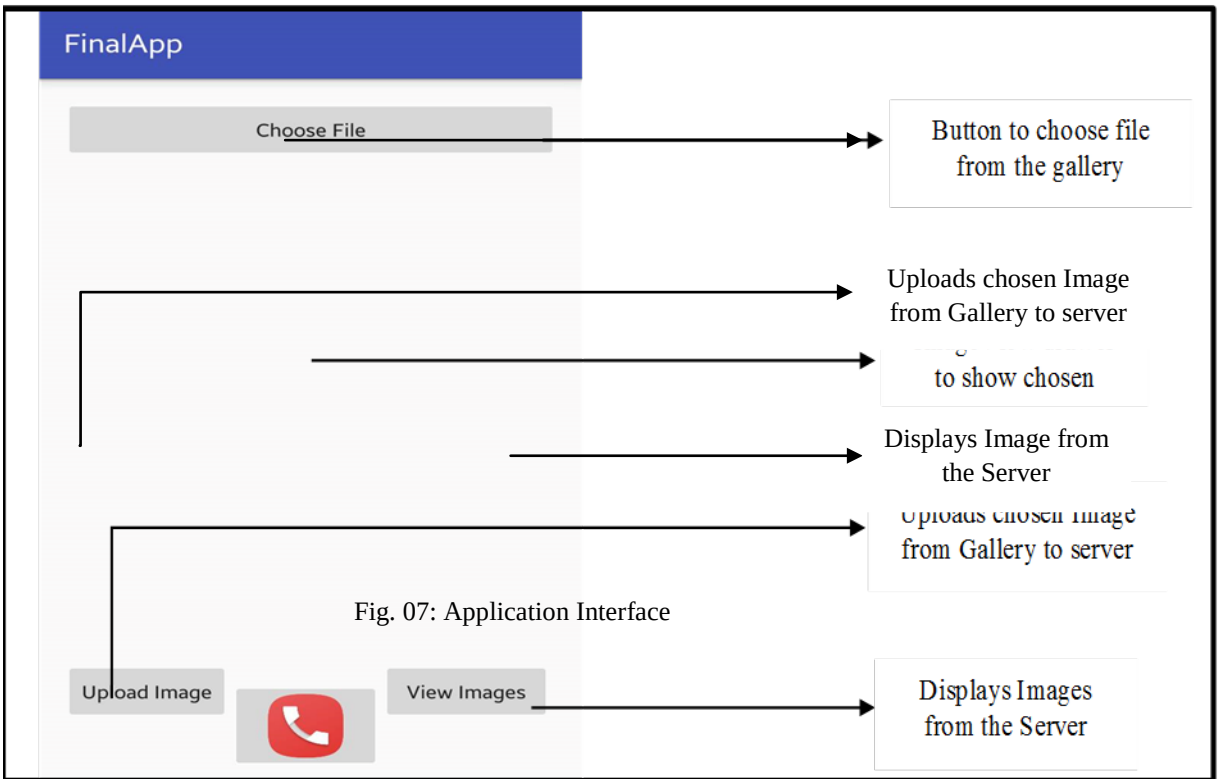


Fig. 07: Application Interface

Fig. 12: Application Interface

- **Choose File Button:** For making it easier for the user to upload photos to the server, this button will be used so that pictures from the phone's existing gallery can be used for the facial recognition. This button is used to choose the file from the gallery and display it in the Image View path of the application where the user can see the image that they have chosen. Here, firstly the button created is connecting to the On Click Listener method so

that the button works on click. The Intent created for this is in a private void where the action set is ACTION_GET_CONTENT and PICK_IMAGE_REQUEST. After these actions are completed, the file path is stored along with the data in the Bitmap. Bitmap is an image file format or an organization of memory used to store digital images [16]. Then the data is displayed in the Image View log area for display. The following code displays the complete method for this button.

Pseudo:

```
1. Add Button;
2. Call Intent ACTION_GET_CONTENT;
3. Start an Activity for Result;
4. Call PICK_IMAGE_REQUEST;
5. On success get file path and data;
6. Store in bitmap;
7. Show data in ImageView;
8. Return
```

- **Upload Image Button:** The server to application connections comes in this part of the application using the dbConnect.php file and the upload.php file. The stored path and data file stored in the Bitmap is called in this step where the image is encoded in strings using base64. Base64 is a binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation[17].The RequestHandler.java file in the application works with the connection to the HttpUrl which is proved in the Main Activity as UPLOAD_URL. The java file calls the for the connection method with URLs and also completes the POST method request required to complete the upload procedure. In the Upload Button, the encoded image is called along with the upload URL where the post method request works to upload the encoded image

to the server. This application uses Toast to post messages if the image has been uploaded successfully. A toast provides simple feedback about an operation in a small popup [18]. The given code below describes the complete process of an image upload.

Pseudo Code:

```
1. Encode Image using base64;
2. Store in bitmap;
3. On button press call RequestHandler;
4. Show progress dialog with a message;
5. Get data information from bitmap;
6. Upload Image to data;
7. Use Toast to show if Upload Successful;
8. Return result;
```

- **View Image Button:** This button is associated with three of the Java classes needed to complete the task. The first one is used for this button to access the custom list view that has been created as the layout for displaying the images. This Java class uses the get View command to get the encoded image in a scaled manner to set it in the list that has been created in the application. The next Java class uses JSON objects to import the encoded data file from the bitmap. JSON (JavaScript Object Notation) is a lightweight data-interchange format [19].The imported string is stored as an array and when called decodes the existing data file with the image URL already placed before. The last Java class uses the get All Images phpto import the image URLs from the database one by one and connects to the custom list Java class to save the images on the list view activity and

made available for viewing. Other than this, the application is also able to show the full image when clicked on any of the images in the list to open another activity where the whole image is available. The code for the activity is given below:

Pseudo Code:

```
1.  Onclick start CustomListview;
2.  Get scaled encoded image using getView;
3.  Import encoded data file using JSON Object;
4.  Store the imported String in an array;
5.  On call decode the file  with base64 Using image
    URL ;
6.  Import each image URLs from database ;
7.  Connects to the custom list;
8.  Save images to the listView;
9.  Show images in CustomListView;
10. Return
```

- **Emergency Call Button:** If any unwanted activity is noticed by the user, the emergency call button will come in handy to make a phone call to a number that the user can choose during installation. The number to be called is set in the code, which for this project is only one number. In other cases, it can be set to multiple numbers as needed by the user. The call button is done using implicit Intent with appropriate actions.

  We are using ACTION_CALL to trigger the built-in phone functionality that Android devices provide automatically. The pseudo code give below completes the calling action.

Pseudo Code:

1. Add Phone calling permission to App;
2. Add Button;
3. Connect Button to work when Clicked;
4. Call Intent ACTION_CALL;
5. Set Phone number;
6. If permission checks out, make call
7. Return

Instead of ACTION_CALL, we can also use the Intent for ACTION_DIAL, so that when the button is pressed the number opens in the device phone dialer and the user can use this. As this is only a prototype, we have used the ACTION_CALL Intent for simplicity.

6SETUP AND EXPENSE

Setting up and running the system is a fairly simple and is needed to be done only once. The user chooses the area that needs to be under surveillance first. The camera module connected with the board is placed strategically near the point of entry to capture the best angles of visitors' faces. It is definitely preferable that the area under surveillance be well illuminated and there isn't any obstruction between the camera and the point of entry for the visitors. Board power is a direct connection provided by main power supply or batteries. The programs required for the functioning of this system can be made available online or can come pre-installed on the RPi. This step can prove to be difficult because it will involve thorough and extensive instructions the

users have to follow. The user needs to input information about the network the board will be connected to. However, once the program is running the system will be up and online and should keep on running without delay.

Fig. 13a : Raspberry Pi Setup

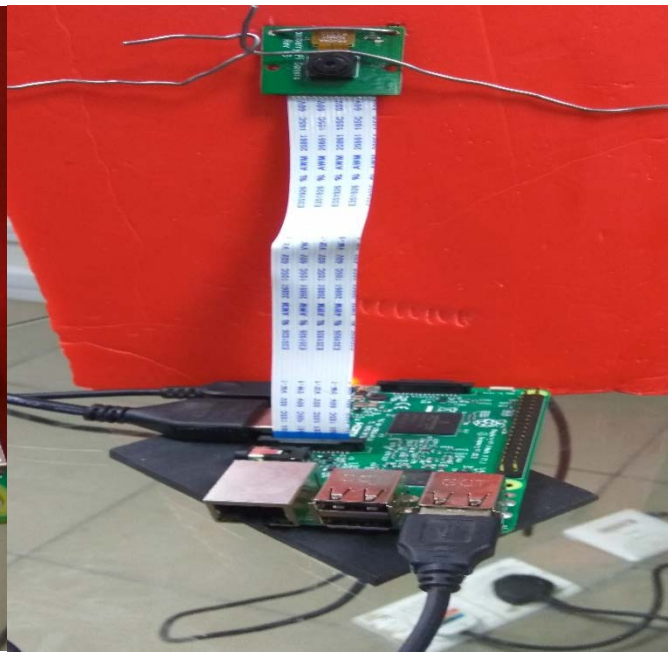


Fig. 13b: Complete connection with Camera Module

The expenditure in setting up and running this system is illustrated in the following table:

	Hardware	Function	Price (BDT)
Setup	Raspberry Pi Model Raspberry Pi 1 Model B+	Processing and uploading the images.	2500
	Raspberry Camera Module	Taking photographs.	2250
	SD Card	Storing OS for Pi board and images taken by camera.	1000

Fig. 14: Table for Expense

The total expense of setup may vary depending on the specifications of the hardware listen above. Latest Raspberry Pi boards are capable of handling more complicated processes and hence cost more. Likewise, camera modules with image sensors of higher quality and SD cards of bigger size will increase the overall cost of the project but also add to the efficiency of the system. Additional cost only includes the network connectivity which may have a monthly bill or usage limitation.

7 RELATED WORKS

1. Raspberry Pi Security System with Motion Detection / Camera made by Max Williams. https://www.hackster.io/FutureSharks/raspberry-pi-security-system-with-motion-detection-camera-bed172

   A design for a security system based on Raspberry Pi similar to this project was made by Max Williams. The features of that project are as follows:

   - Motion triggered image capture.
   - Mobile notifications with photos.
   - Arms or disarms automatically.
   - Remote disable option or queried for status using Telegram.

   The hardware used for this design includes:

   - Raspberry Pi 1 Model A+.
   - Raspberry Pi Camera Module.
   - PIR Motion Sensor (generic)

- USB WLAN/WiFi Adapter

RaspbianLunix and Telegram Bot were used to run this project.

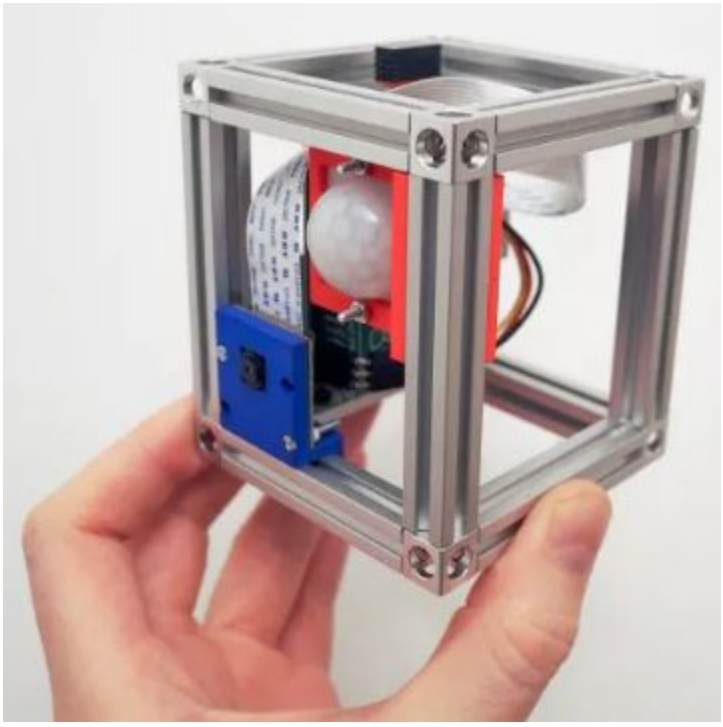


Fig. 15aRaspberry Pi based home security system

This design includes a way of automatically detecting the users presence via ARP scan and passive (packet capture) detection over Wi-Fi. However, the notifications with the image are sent over Telegram rather than a separate app for the purpose of using this security system only. The Telegram bot is used for enable/disable and status controls. The main difference and disadvantage in this design is that the phone/mobile terminal is required to be connected to the same Wi-Fi connection as the Pi board. So, the user will not get updates from this system when out of coverage of that particular Wi-Fi network.

2. Home Security Email Alert System using Raspberry Pi by jayakarthigeyan

The second related project detects activity within a given space using PIR sensors and sends an E-mail to the user afterwards. The design requires the following components:

- Raspberry Pi running Raspbian
- USB Camera
- PIR Sensor
- USB Wi-Fi Module for internet access (optional, if you connect your Ethernet cable for internet, USB Wi-Fi Module is not needed)
- Power Adapter to power the Raspberry Pi.

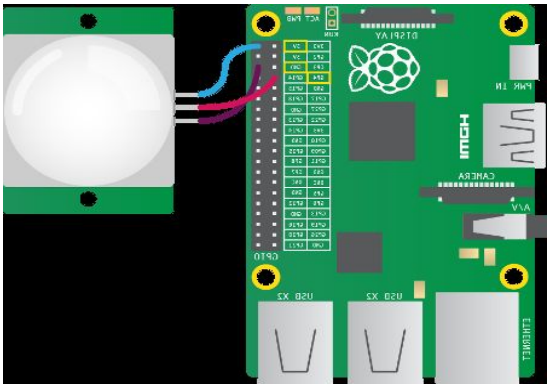


Fig. 15b: Connection between PIR and Pi

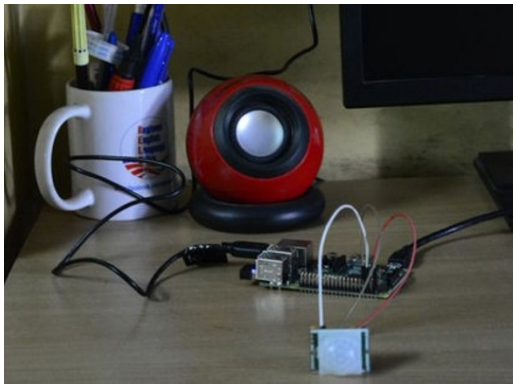


Fig. 15c Final Set up

This design is simpler and cheaper compared the one described in the previous section. The disadvantage of this design is that the sensor will detect any and sort of movement that occurs in front of it which may send intrusion alert emails to the user when it is not necessary.

8 COMPARISON

The most conventional methods of security and surveillance in Bangladesh are security guards and CCTV cameras. This section draws a comparison between these methods and the system described in this paper. Security guards are easily available and most commonly used form of security and surveillance. On average, a security guard in Dhaka city costs 10000 Taka. They are available through agencies or independently. An intelligent system is favorable for surveillance because security guards can commit errors, consciously or unconsciously, in identifying individuals. Security guards can be bribed or threatened to leak information about the families or offices they are guarding. However, an intelligent system cannot replace or compete with human intuition. Additionally, guards in most households or offices are responsible for a wider range of work including opening gates or fetching goods from shops. In recent years, CCTV (close circuit television) camera has gained popularity especially in large housing complexes and office spaces. It is capable of providing a continuous stream of video of a given area. All activity that occurs in that area can be recorded and stored for investigation if needed. CCTV footage has been instrumental in collecting evidence and solving many crimes like theft and robbery. The price of a CCTV camera of good resolution begins from 4500 Taka. But a camera on its own is not a valid or sound security measure as it cannot notify the users immediately of any disturbances unless the feed is being viewed in real time or there is an additional alarm system set up. The intelligent system using raspberry pi has the ability to not only provide a video feed just like CCTV camera but detect intrusion by strangers and notify the users almost instantly. A system like that is an improvement in surveillance measures appropriate for the ongoing advancement in technology in Bangladesh.

9 OTHER APPLICATIONS

The basic idea of the intelligent security system has possible application in other areas. The use of facial recognition can be brought to keeping attendance or for plain identification purposes. With improvements mentioned above and adding required components, the system can be installed in classrooms for keeping track of attendance of students. A database needs to be made with all the photos of the students enrolled in a given class. When a student enters the class, the facial recognition system can detect his presence and update the attendance table for that day with that particular student's name. The camera needs to be placed in a position from where it can take proper pictures of students entering. The instructor can control arming and disarming the system with an application or keypad pass code. Another possible application of this system is replacing the process of checking ID cards to permit students into campus or employees into office spaces. The process described earlier can be used along with a locked door or turnstile while under supervision of guards for added layers of security.

10 LIMITATIONS

The Raspberry Pi based security system was formulated in a short period of time while also under a limitation of funding. Despite being a complete and functioning system, there are multiple performance limitations in the design. The limitations, however, can be overcome by adding new features and components which unfortunately may increase the total expense of the

project. The most significant problems that may emerge while using this system were identified and analyzed. The problems, their solutions and ways to further improve this design are summarized in the following points:

1) **Load Shedding:** A prevalent problem in the country, load shedding is the obvious and recurring problem with using a system based on Raspberry Pi as it is powered through main line electricity. Moreover, Wi-Fi connectivity may be unavailable during load shedding. Running the board on battery is possible with low increase in expense because the board requires. However, running a Wi-Fi router is not feasible just on pencil batteries. Wi-fi router can be run on IPS (Instant Power Supply) during load shedding which will ensure uninterrupted connectivity.
2) **Photo Quality:**In order to identify the faces in the photographs taken, there are particular features that need to be prominently visible in them for the Pi to process. If for some reason the required pointers are not available, the RPi will not be able to perform recognition. This complication will occur most in low to no light and during night time. The solution to this problem is to illuminate the target area of the system while also using high resolution camera with the board.

11 POSSIBLE IMPROVEMENTS

There is significant room for improvement in designs based on micro-controllers like Raspberry pi because of the wide array of attachable modules for performing different functions. Options available in the Android application used for this system can be constantly upgraded according to

the needs of the users. Few such ideas on developing this system into a more user friendly and efficient one were formulated during the designing of this system.

1) **Push Notification:** The app can be coded to receive push notifications sent from the server so that the process of checking the photos from the e-mail or server is easier and faster.
2) **App gets camera access:** Direct access to the camera from the application will allow the user to upload photo of permissible individuals to the server without going through gallery application of the phone.
3) Live feed: The camera module and RPi is capable of streaming live video feed to the user given the required internet speed. This feed can also be recorded for viewing or investigation purposes.
4) Automatic Arming and Disarming: Using active (ARP scan) and passive (packet capture) detection over the Wi-Fi adapter based on knowing the MAC addresses of the mobile phones carried by the user the system can be taught to arm and disarm automatically like previously discussed for a related project.

12 CONCLUSION

In our paper, we have tried to come up with a prototype of an intelligent security system which has reliability and mobility. With our project and paper, we have tried to integrate Raspberry Pi and Android Application to achieve a real-time security system. There are existing models of security but none of those let the user receive real time image of the intruder especially in Bangladesh. The other applications are also very much required and practical for the context of educational institutions and work places to reduce the hassle everyone is facing now days. In this era of extreme chaos, we need one less thing to worry about and that is about our security. It is time to move forward and include more features and mobility to our security systems so we can get updates on the go. This security system is only a prototype which can be developed more for commercial use and also has immense application especially in the context of Bangladesh in recent times. Though our project is now seen to be a bit expensive but which more time the costs can be reduced much more so that this can be a practical system so more people can have access to an advanced security model.

13 REFERENCES

1. https://en.wikipedia.org/wiki/Facial_recognition_system#History

2. https://raspberrypi.org

3. https://developer.android.com/studio/intro/index.html 02 December 2016

4. https://developer.android.com/about/dashboards/index.html retrieved on 02 December 2016.

5. https://gradle.org/the-new-gradle-android-build-system/ retrieved on 02 December 2016

6. https://help.github.com/ retrieved on 02 December 2016

7. http://tools.android.com/tips/lint retrieved on 02 December 2016

8. https://firebase.google.com/ retrieved on 02 December 2016

9. http://php.net/manual/en/intro-whatis.php retrieved on 06 December 6, 2016

10. https://developer.android.com/reference/android/content/Intent.html retrieved on 06 December 7, 2016

11. https://en.wikipedia.org/wiki/Python_(programming_language)

12. http://www.numpy.org/

13. http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

14.http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#face-recognition-with-opencv

15. https://developer.android.com/reference/android/graphics/Bitmap.html retrieved on 10 December, 2016

16. https://developer.android.com/reference/android/util/package-summary.html retrieved on 10 December, 2016

17. https://developer.android.com/guide/topics/ui/notifiers/toasts.html retrieved on 10 December, 2016

18. www.json.org/ retrieved on 10 December, 2016