INTEGRATED SECURITY SYSTEM

by

GC Muhammad ul Husnain Nawaz

Capt. Hafiz Qaisar Islam

PC Hira Shahid

Submitted to the Faculty of Department of Electrical Engineering National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Telecommunication Engineering

June 2013

CERTIFICATE

It is thereby certificated that the project is dually completed by a group of EE department under the supervision of Asst. Prof. Dr. Safia Akram.

____

(Signature)

Name: Asst. Prof. Dr. Safia Akram

Date:

ABSTRACT

Biometrics is the identification of humans by their characteristics or traits. Besides it is used as a form of authentication. It is also helpful to identify and authenticate individuals at the entrance. Biometric identifiers are the distinctive, measurable characteristics used to label and describe individuals. The field of Biometrics is expanding and escalating with the increasing need of security as it is regarded as an authentic source of recognition.

This project is based on integration of biometric and non-biometric sources for ensuring automatic security system at the entrance. Biometric source involves the Facial Recognition while non-biometric is the RFID authentication. The objective of project is to develop a prototype for intelligent integrated security system based on biometric and non-biometric sources and to provide fool proof security at entrance which requires no human assistance.

This technology is widely deployed around the world but in Pakistan, it is not that common mainly because of the fact that it is very expensive and secondary there is less awareness about it among the masses. The biggest aim of choosing this project was to design a commercial system that is also cost-effective so that it can be affordable for small companies and institutions with low turnovers.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATED TO

Almighty Allah,

Faculty for their help

And our parents for their support

ACKNOWLEDGEMENT

All praise and glory to Almighty Allah (Subhanahu WaTaalaa) who gave us courage and patience to carry out this work. Peace and blessing of Allah be upon last Prophet Muhammad (Peace Be upon Him).

We are greatful and would like to express our sincere gratitude to our project supervisor Asst. Prof. Dr. Safia Akram for her invaluable guidance, continuous encouragement and constant support and appreciate her guidance from the initial to the final level that enabled us to develop an understanding of this research project. Without her advice and assistance it would be a lot tougher to completion.

We would also like to express very special thanks to our Co-Supervisor Lt. Col. Dr. Abdul Ghafoor for his suggestions and co-operation especially in dealing with image processing. A heartiest appreciation is to be given to Mr. Muhammad Sohaib, MS in Telecom. from MCS (NUST) who gave us a brand new perception about OpenCV and Emgu using C++. Special thanks to Lecturer Uzma Ehsan from Department of H & BS, MCS for her time spent on proof reading and correcting our mistakes.

Lastly we would like to extend our thanks to every person who has contributed to our final year project directly or indirectly. We would like to acknowledge their comments and suggestions, which were essential for the successful completion of this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

PIN	Personal Identification Number
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
LBPH	Local Binary Pattern Histogram
PIN	Personal Identification Number
POS	point-of-sale
CCD	Charge-Couple Device
RAM	Random Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
ROM	Read Only Memory
CPU	Central Processing Unit
RFID	Radio Frequency Identification
RF	Radio Frequency
DSRC	Dedicated Short Range Communication
DSLR	Digital Single Lens Reflex
CV	Computer Vision
Emgu	The Most General Unifier
MySql	My Sequential Query Language
EPC	Electronic Product Code
USB	Universal Serial Bus
COM	Communication
RS232	Recommended Standard 232
PCB	Printed Circuit Board
LED	Light Emitting Diode
LCD	Liquid Crystal Display

Chapter 1

Introduction

Security at the entrance gates is an issue of grave seriousness for any institution. Now-a-days, every company or institution issues some ID cards that are checked at the gates manually but the notion of using fake ID cards is always there. So, in tandem with manual checking of cards, there should be some other source of recognizing the person entering at the gates. Another major issue of concern, at the gates, is the human life i.e. life of security personnel or security guards is at a high risk. Thus there raises a calamitous need for the automated security system which requires no human assistance.

With the increasing importance of personal identification for the purpose of security in remote transactions in today's world of electronic communication, biometric identification techniques are rapidly evolving into a pervasive method for personal verification. Among the different biometrics proposed for such purpose, face recognition, fingerprints, hand prints, voice prints, retinal images, handwriting samples and etc.

1.1 Biometric sources in Security system

Biometrics is the science and technology of measuring and analyzing biological data. It measures and analyzes human body characteristics, such as DNA, fingerprints, eye retinas and irises, voice patterns, facial patterns and hand measurements, for authentication purposes. Biometrics can be classified into two classes [1]

- Physiological like face, fingerprints, hand geometry and iris Recognition
- Behavioral like signature and voice

1.1.1 Advantages and Disadvantages of Biometrics

Advantages:

Following are the advantages of Biometrics

- Increase security - provide a convenient and low-cost additional tier of security
- Reduce fraud by employing hard-to-forge technologies and materials. e.g. minimize the opportunity for ID fraud, buddy punching
- Eliminate problems caused by lost IDs or forgotten passwords by using physiological attributes.e.g. prevent unauthorized use of lost, stolen or "borrowed" ID cards
- Reduce password administration costs
- Replace hard-to-remember passwords which may be shared or observed
- Integrate a wide range of biometric solutions and technologies, customer applications and databases into a robust and scalable control solution for facility and network access
- Make it possible, automatically, to know *who* did *what, where* and *when!*
- Unequivocally link an individual to a transaction or event.

Disadvantages of a Biometric System:

Following are the disadvantages of Biometrics

- The finger print of people working in chemical industries is often affected therefore these companies should not use the finger print mode of authentication.
- It is found that with age, the voice of a person becomes different. Besides, when the person has flu or throat infection the voice changes or if there is too much

noise in the environment this method may not authenticate correctly. Therefore this method of verification is not workable all the time.

- For people affected with diabetes, the eyes get affected resulting in differences. [1]

1.2 Non Biometric sources in Security system

Non Biometrics is the science and technology of analyzing numeric data. Different non biometric sources such as Bar Code Reader, Magnetic Stripe, Smart Cards, and RFID are used for the security purposes for analyzing, identifying and authenticating persons.

1.3 Organization of Document

Motivated from the importance of the Biometric and non-Biometric sources, the aim of the present thesis is to highlight the usage of integration of Biometric and non-Biometric sources for security purposes. This thesis consists of seven chapters. The seven chapters are arranged as follows:

Chapter 1 describes the importance of biometric and non biometric sources, Chapter 2 deals with literature review of different algorithms used for facial recognition and RFID authentication. Chapter 3 deals with the selection of algorithms for face detection and recognition while Chapter 4 is about project layout of the given Integrated Security System. Chapter 5 illustrates the system design specifications, limitations and the types of different software and hardware equipment used in this project while Chapter 6 describes system development and results. Chapter 7 concludes the thesis and gives direction to future research.

Chapter 2

Literature Review

2.1 Face Detection

Face detection is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores everything else, such as buildings, trees and bodies. Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.

Face detection can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one) but in face detection, one does not have this additional information.

Early face-detection algorithms focused on the detection of frontal human faces, whereas new algorithms attempt to solve the more general and difficult problem of multi-view face detection i.e. the detection of faces that are either rotated along the axis from the face to the observer (in-plane rotation), or rotated along the vertical or left-right axis (out-of-plane rotation), or both. The new algorithms take into account variations in the image or video by factors such as face appearance, lighting, and pose. [2]

- **Neural Network-Based Face Detection, 1998 [3]**

  By Henry A. Rowley, ShumeetBaluja and Takeo Kanade.

An image pyramid is calculated in order to detect faces at multiple scales. A fixed size sub-window is moved through each image in the pyramid. The content of a sub-window is corrected for non-uniform lightning and subjected to histogram equalization. The processed content is fed to several parallel neural networks that carry out the actual face detection. The outputs are combined using logical AND, thus reducing the amount of false detections. In its first form, this algorithm also detects frontal upright faces only.

- **A Statistical Method for 3D Object Detection Applied to Faces and Cars, 2000**

  By Henry Schneiderman and Takeo Kanade.

The basic mechanics of this algorithm is also to calculate an image pyramid and scan a fixed size sub-window through each layer of this pyramid. The content of the sub-window is subjected to a wavelet analysis and histograms are made for the different wavelet coefficients. These coefficients are fed to different trained parallel detectors that are sensitive to various orientations of the object. The orientation of the object is determined by the detector that yields the highest output. Opposed to the basic Viola-Jones algorithm and the algorithm presented by Rowley et al. this algorithm also detects profile views.

- **Robust Real-Time Objection Detection, 2001**

  By Paul Viola and Michael J. Jones.

This seems to be the first article where Viola-Jones presents the sound set of ideas that constitutes the fundamentals of face detection algorithm. This algorithm only finds frontal upright faces, but in 2003 the very algorithm was presented in a variant that also detects side views. Major advantages of this algorithm are:

- It is efficient.

- The results provided by this algorithm are accurate.
- It has fast execution. Processing time is less.
- Its performance is remarkable.

2.1.1 The Viola-Jones Face Detector

Method

The basic principle of the Viola-Jones algorithm is to scan a sub-window capable of detecting faces across a given input image. The standard image processing approach would be to rescale the input image to different sizes and then run the fixed size detector through these images. This approach turns out to be rather time consuming due to the calculation of the different size images. Contrary to the standard approach Viola-Jones rescale the detector instead of the input image and run the detector many times through the image – each time with a different size. At first one might suspect both approaches to be equally time consuming, but Viola-Jones have devised a scale invariant detector that requires the same number of calculations whatever the size. This detector is constructed using a so-called integral image and some simple rectangular features reminiscent of Haar wavelets. [3]

- **The Scale Invariant Detector**

The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel.

1	1	1
1	1	1
1	1	1

Input Image

1	2	3
2	4	6
3	6	9

Integral Image

This allows for the calculation of the sum of all pixels inside any given rectangle using only four values. These values are the pixels in the integral image that coincide with the corners of the rectangle in the input image.

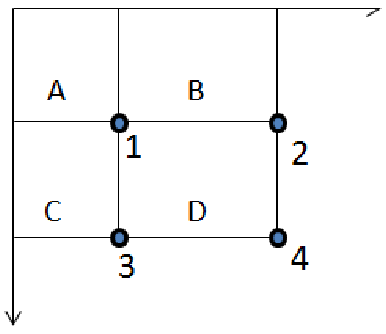


Figure 2- 1 Integral Image

Sum of all pixels in D = 1 + 4 - (2 + 3)

= A + (A + B + C + D) - (A + C + A + B)

= D

Since both rectangle B and C include rectangle A the sum of A has to be added to the calculation .It has now been demonstrated how the sum of pixels within rectangles of arbitrary size can be calculated in constant time. The Viola-Jones face detector analyzes a given sub-window using features consisting of two or more rectangles.

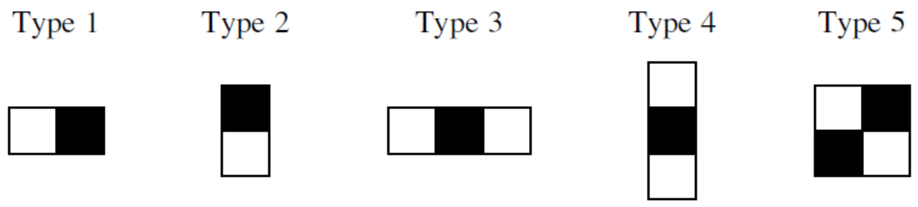


Figure 2- 2 Different types of features

Viola-Jones have empirically found that a detector with a base resolution of 25*25 pixels gives satisfactory results when allowing for all possible sizes and positions of the features .Total of approximately 160,000 different features can then be constructed. Thus, the amount of possible features vastly outnumbers the 576 pixels contained in the detector at base resolution. These features may seem simple to perform such an advanced task as face detection.

- **The Modified AdaBoost Algorithm**

AdaBoost is a machine learning boosting algorithm capable of constructing a strong classifier through a weighted combination of weak classifiers. (A weak classifier classifies correctly in only a little bit more than half the cases.) To match this terminology to the presented theory each feature is considered to be a potential weak classifier. A weak classifier is mathematically described as:

$$h(x, f, p, \theta) = \begin{cases} 1 & if pf(x) > p\theta \\ 0 & otherwise \end{cases} \qquad \text{Eq. (2.1)}$$

Where x is a 25*25 pixel sub-window, f is the applied feature, p is the polarity and θ the threshold that decides whether x should be classified as a positive (a face) or a negative (a non-face). [3]

The Cascaded Classifier

The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size. Even if an image may contains one or more faces it is obvious that an excessive large amount of the evaluated sub-windows would still be negatives (non-faces). This realization leads to a different formulation of the problem. Instead of finding faces, the algorithm should discard non-faces.

The thought behind this statement is that it is faster to discard a non-face than to find a face. With this in mind a detector consisting of only one (strong) classifier suddenly seems inefficient since the evaluation time is constant no matter what is the input. Hence, the need for a cascaded classifier arises.

The cascaded classifier is composed of stages each containing a strong classifier. The job of each stage is to determine whether a given sub-window is definitely not a face or maybe a face. When a sub-window is classified to be a non-face by a given stage, it is immediately discarded. Conversely a sub-window classified as a maybe-face is passed on to the next stage in the cascade. It follows that the more stages a given sub-window passes, the higher the chance the sub-window actually contains a face.

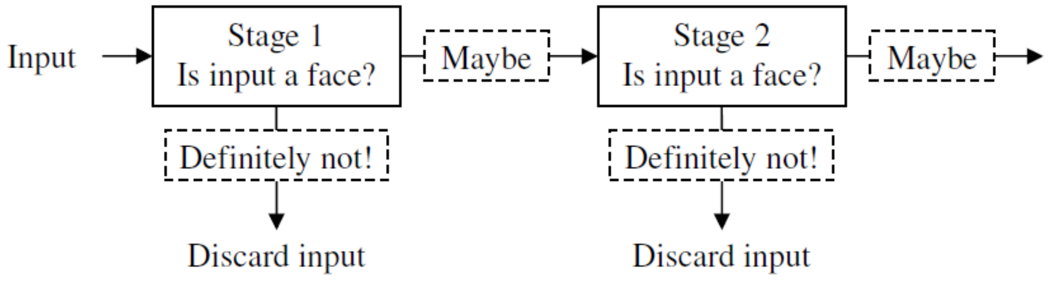


Figure 2- 3 Rejection cascade

In a single stage classifier one would normally accept false negatives in order to reduce the false positive rate. However, for the initial stages in the staged classifier false positives are not considered to be a problem since the succeeding stages are expected to sort them out. Therefore Viola-Jones prescribes the acceptance of many false positives in the initial stages. Consequently the amount of false negatives in the final staged classifier is expected to be very small. [3]

2.2 Face Recognition

After the face detection the second step starts i.e. face recognition. The detected face from the Viola Jones algorithm works as input to the face recognition.

Three algorithms were implemented for the face recognition and compared their outputs, efficiency and accuracy.

2.2.1 Advantages of Face Recognition Technology

Face recognition is the most natural biological features recognition technology according to the cognitive rule of human beings; its algorithm is ten times more complex than a fingerprint algorithm. Face recognition is featured by the following advantages compared to fingerprint: [4]

Accurate and Fast Identification

- Industrial Leading "Dual Sensor™" Facial Recognition Algorithm matches more data than fingerprint, FAR<0.0001%
- Fast as it can recognize up to 1400 users less than 1 second

High Usability and Security

- Rate of failure to enroll and acquire is less than 0.0001%, fingerprint technology will have problems for enrollment with cold, wet, desquamation, elder, and around 5% people cannot get enrolled with fingerprint technology.
- Incident becomes tractable with photo captured by camera whereas there is no evidence with finger print technology to track the incident.

User friendly design

- Contactless authentication for the ultimate in hygienic.

2.2.1 Principal Component Analysis

Firstly, Face recognition system using the Principal Component Analysis (PCA) algorithm is implemented. The algorithm is based on an Eigen faces approach which represents a PCA method in which a small set of significant features are used to describe the variation between face images.

The human face is an extremely complex and dynamic structure with characteristics that can quickly and significantly change with time. It is the primary focus of attention in social relationships and plays a major role in the transmission of identity and emotions. Therefore, face recognition is applied in many important areas such as security systems, identification of criminals, and verification of credit cards and so on. Unfortunately, many face features make development of facial recognition systems difficult. This problem is solved by the method called Principal Component Analysis. PCA is a projection technique that finds a set of projection vectors designed such that the projected data retains the most information about the original data. The most representative vectors are eigenvectors corresponding to highest eigenvalues of the covariance matrix. This method reduces the dimensionality of data space by projecting data from M-dimensional space to P-dimensional space.

The advantage of this approach over other face recognition systems is in its simplicity, speed and insensitivity to small or gradual changes on the face. The problem is limited to files that can be used to recognize the face. Namely, the images must be vertical frontal views of human faces.

PCA is a method for identifying patterns in the data and to express it in a way to highlight the differences and similarities. Since it is difficult to analyze the patterns in high dimensions therefore, PCA is used to reduce the dimensionality of the data thus proving useful for analysis. It generates an average face and represents all other faces as the sum and difference of this average face and all other Eigen faces.

2.2.2 Linear Discriminant Analysis (LDA)

Principal Component Analysis (PCA), which is the core of the Eigen faces method, finds a linear combination of features that maximizes the total variance in data. While this is clearly a powerful way to represent data, it doesn’t consider any classes and so a lot of discriminative information may be lost. The components identified by a PCA do not necessarily contain any discriminative information at all, so the projected samples are smeared together and a classification becomes impossible. [5]

Moreover, the accuracy of PCA was very less and the ratio of false recognition was very high. Hence Linear Discriminant Analysis method was implemented.

There major advantages of Linear Discriminant Analysis (LDA) are

- It produces accurate results comparatively better results than PCA
- Output of code not affected by minor light variations.
- Output of algorithm is not affected by minor changes in facial expressions, hence producing correct authentication.
- It consumes less time.

Linear discriminant analysis (LDA) and the related Fisher's linear discriminant are methods used in statistics, pattern recognition and machine learning to find a linear combination of features which characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier or, more commonly, for dimensionality reduction before further classification.

2.2.3 Local Binary Pattern Histogram (LBPH) Analysis

Local Binary Pattern Histogram Analysis is an advance technique as compared to PCA and LDA. It works by calculating the local binary patterns and assigning a binary number to 3x3 box of pixels.

- **Local Binary Pattern**

LBP uses concept of circular neighborhoods and linearly interpolating the pixel values allows the choice of any radius R for computations and number of pixel in the neighborhood, *P*, to form an operator, which can model large scale structure. [6]

LBP takes a 3x3 box of pixels and calculates the threshold value for each pixel by using the equation. It then calculates threshold for the 3x3 pixels. The pixels in this block are threshold values by its center pixel value, multiplied by powers of two and then summed to obtain a label for the center pixel.

As the neighborhood consists of 8 pixels, a total of $2^8 = 256$ different labels can be obtained then LBPH compares each pixel with the calculated threshold value. If the value of pixel is greater than threshold 1 is assigned otherwise 0 is assigned. Hence a binary pattern is obtained. [7]

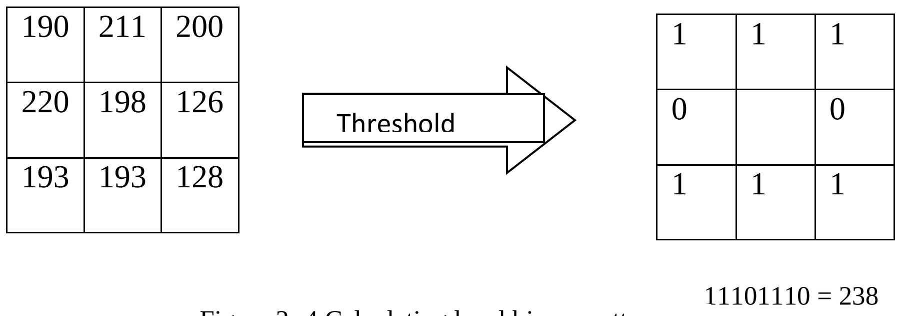


Figure 2- 4 Calculating local binary pattern

- **Image Histogram**

Histogram is a bar graph. Similar to histogram, the image histogram is graphical of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. Histogram for an image shows entire tonal distribution at a glance.

- ## Local Binary Pattern Histogram (LBPH) for Face Recognition

Generic LBP Operator

LBP using 8 pixels in a 3 × 3 pixel block, this generic formulation of the operator puts no limitations to the size of the neighborhood or to the number of sampling points.

- **Mappings of the LBP Labels**

In texture analysis applications it is favorable to have features that are invariant or vigorous to rotations of the input image. As the LBPP, R patterns are obtained by circularly sampling around the center pixel; rotation of the input image has two effects:

- Each local neighborhood is rotated into other pixel location, and within each neighborhood, the sampling points on the circle surrounding the center point are rotated into a different orientation.
- Another extension to the original operator uses so called uniform patterns.

For this, a uniformity measure of a pattern is used: U (“pattern”) is the number of bitwise transitions from 0 to 1 or vice versa when the bit pattern is considered circular. A local binary pattern is called uniform if its uniformity measure is at most 2. For example, the patterns 00000000 (0 transitions), 01110000 (2 transitions) and11001111 (2 transitions) are uniform whereas the patterns 11001001 (4 transitions) and 01010011 (6 transitions) are not. In uniform LBP mapping there is a separate output label for each uniform pattern and all the non-uniform patterns are assigned to a single label. Thus, the number of different output labels for mapping for patterns of P bits is

$$P\,(P-1)+3 \qquad \text{Eq. (2.2)}$$

For instance, the uniform mapping produces 59 output labels for neighborhoods of 8 sampling points, and 243 labels for neighborhoods of16 sampling points.

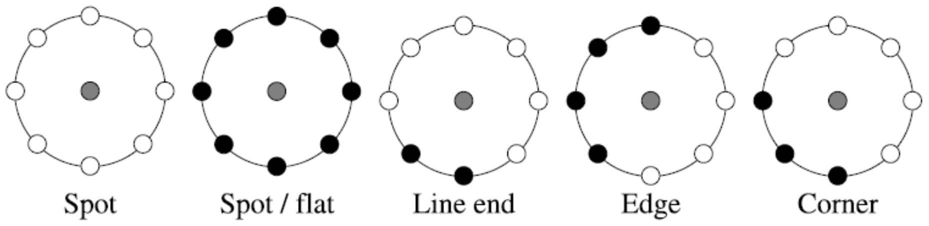


Figure 2- 5Different textures detected by LBPH

Applications of Local Binary Pattern Histogram Analysis

- The LBP has been extended to multi resolution analysis.
- The LBP and its extensions have already been applied for visual analysis.
- It is used in image retrieval.
- It is used in motion detection.
- It is used in remote sensing.
- It is used in biomedical image analysis.

Advantages of LBPH

- It provides higher recognition rate.
- Its efficiency is not affected by light variations.
- Its efficiency does not decrease if data base is increased.

Eigenfaces and Fisherfaces take a somewhat holistic approach to face recognition. They treat data as a vector somewhere in a high-dimensional image space. But high dimensionality is non-effective, so a lower-dimensional subspace is identified, where useful information is preserved. The Eigenfaces approach maximizes the total scatter, which may lead to problems if the variance is generated by an external source, because components with a maximum variance over all classes aren't necessarily useful for classification. So to preserve some discriminative information a Linear Discriminant Analysis is applied and optimized as described in the Fisherfaces method. [8]

2.3 RFID Technology

Radio-frequency identification (RFID) is the wireless non-contact use of radio-frequency electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. The tags contain electronically stored information. Some tags are powered and read at short ranges (a few meters) via magnetic fields (electromagnetic induction).

An early demonstration of RFID tags, both passive and semi-passive, was performed at the Los Alamos National Laboratory in 1973. The portable system operated at 915 MHz and used 12-bit tags. This technique is used by the majority of today's UHFID and microwave RFID tags.

The first patent to be associated with the abbreviation RFID was granted to Charles Walton in 1983. [9]

2.3.1 Advantages of RFID Technology

Although RFID is more costly than barcode, it proves to be indispensable for a variety of automated applications involving data acquisition and object identification.

- **Line-of-sight contact**

  The major advantage of all kinds of RFID system is that they work contactless and require no line of sight.

- **Robust system**

  Transponders can be read through a whole number of substances, e.g. snow, fog, ice, paint, dirt, and in difficult constructional scenarios where barcodes or other optical reading technologies would be no use at all.

- **Speed of RFID system**

  RFID transponders can be read at remarkable speed even in difficult conditions, and in most cases respond in less than 100 milliseconds.

- **Bidirectional Communication**

  The reading/writing capability of an active RFID system is also a significant advantage in interactive applications, e.g. when tracking products in process or maintenance jobs.

- **Reliability in tough environments**

  In difficult external conditions RFID has the advantage of being able to communicate contactless and without direct line-of-sight contact with the data medium. Besides, in situations where the transponder doesn't matter either -- it can be read through substances like dust, paint or ice.

- **Bulk detection**

  Active and passive systems working at HF and UHF frequencies detect a number of transponders in the field. This property is called bulk capability. In practical terms it means that every data medium need not to be scanned singly, but is automatically detected during a read operation.[10]

2.3.2 RFID Working

RFID is a dedicated short range communication (DSRC) technology. The term RFID is used to describe various technologies that use radio waves to automatically identify people or objects. RFID technology is similar to the bar code identification systems see in retail stores everyday; however one big difference between RFID and bar code technology is that RFID does not rely on the line-of-sight reading that bar code scanning requires to work.

With RFID, the electromagnetic or electrostatic coupling in the RF (radio frequency) portion of the electromagnetic spectrum is used to transmit signals. An RFID system consists of an antenna and transceiver, which read the radio frequency and transfer the information to a processing device (reader) and a transponder, or RF tag, that contains the RF circuitry and information to be transmitted. The antenna provides the means for the integrated circuit to transmit its information to the reader that converts the radio waves reflected back from the RFID tag into digital information that can then be passed on to computers that can analyze the data.

In RFID systems, the tags that hold the data are broken down into two different types. Passive tags use the radio frequency from the reader to transmit their signal. Passive tags generally have their data permanently burned into the tag when it is made, although some data may be rewritten.

Active tags are much more sophisticated and have on-board battery for power to transmit their data signal over a greater distance and power random access memory (RAM) giving them the ability to store up to 32,000 bytes of data. [11]

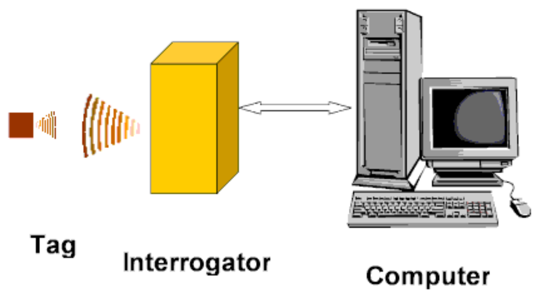


Figure 2- 6 RFID System

Chapter 3

Selection of Algorithms

3.1 Comparison and Selection of Face Detection and Recognition Algorithms

Comparison of output efficiency and accuracy of the three algorithms already discussed in Chapter 2, is given below

3.1.1 Detection Algorithm

- **Viola Jones Face Detector**

Algorithm is experimented on various face images and results are calculated.

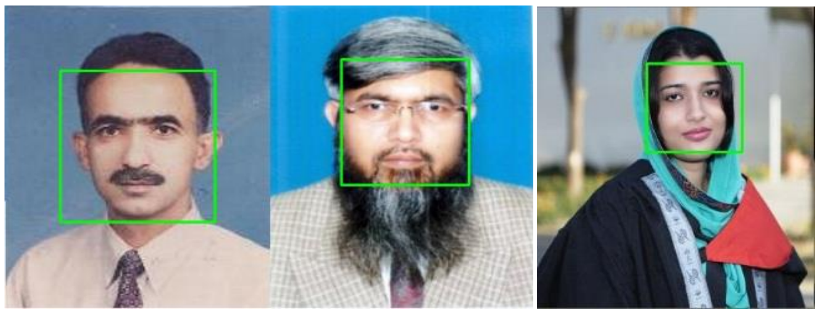
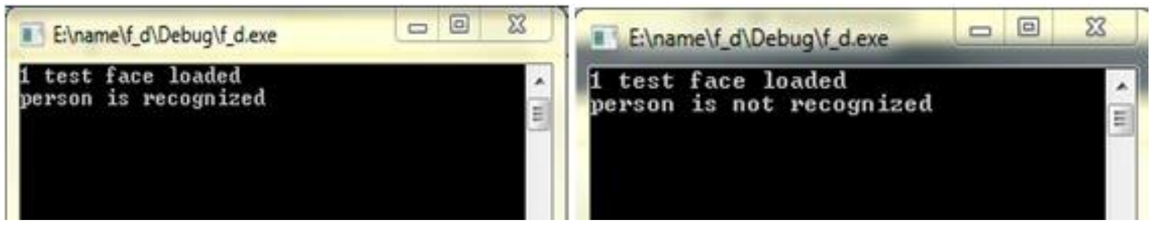


Figure 3- 1 Detected faces in images

3.1.2 Recognition Algorithm

- **Principle Component Analysis (PCA)**

  The very first algorithm implemented for face recognition was Principle Component Analysis (PCA).A data base of 10 persons was made at first. The output showed whether the unknown person is recognizes or not.


```
E:\name\f_d\Debug\f_d.exe
1 test face loaded
person is recognized
```

```
E:\name\f_d\Debug\f_d.exe
1 test face loaded
person is not recognized
```

Figure 3- 2 Output of PCA

- **Linear discriminant analysis (LDA)**

Since it is the simplest algorithm so the output of PCA is affected by minor light variations. It is the simplest algorithm. In order to achieve more accuracy Fisherface Linear Discriminant Analysis method was adopted.

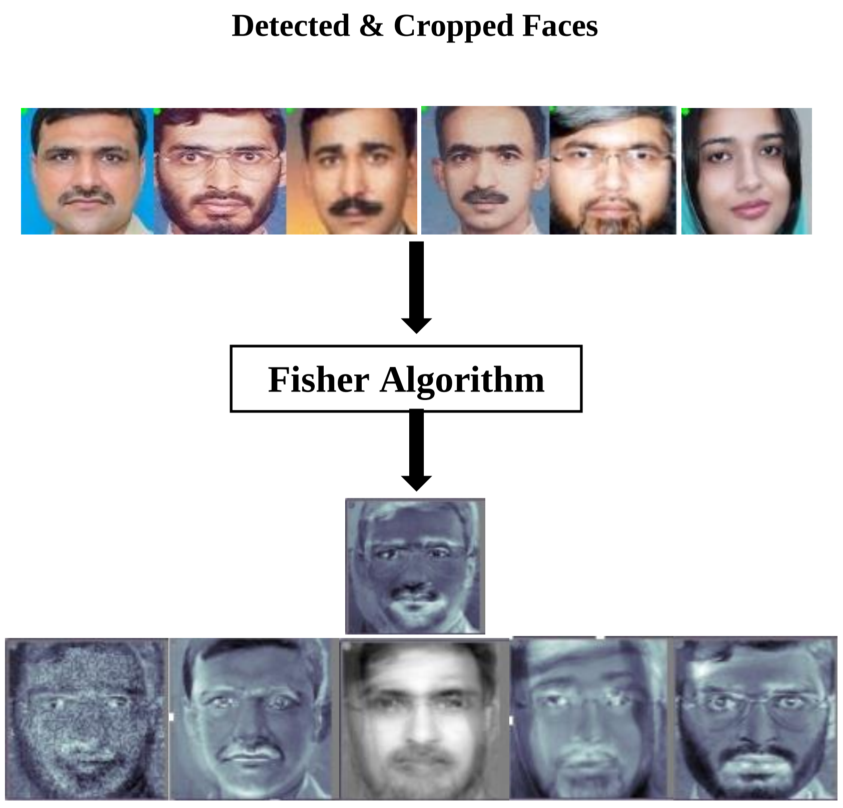


Figure 3- 3 Fisher Faces

Despite all these advantages LDA did not give satisfactory results when applied over a large set of data base images. Accuracy of algorithm became very low. Hence latest method of Local binary patters histogram was implemented.

- **Comparison with other Face Recognition Algorithms**

Eigenfaces (PCA) and Fisherfaces (LDA) take a somewhat holistic approach to face recognition. These algorithms treat their data as a vector somewhere in a high-dimensional image space. As high-dimensionality is non-effective, so a lower-dimensional subspace is identified, where (probably) useful information is preserved. The Eigenfaces approach maximizes the total scatter, which can lead to problems if the variance is generated by an external source. So to preserve some discriminative information, a Linear Discriminant Analysis is analyzed. The Fisherfaces method works great but this method doesn't really help whenever it takes at least 8(+-1) images for each person in order to get good recognition rates. [12]

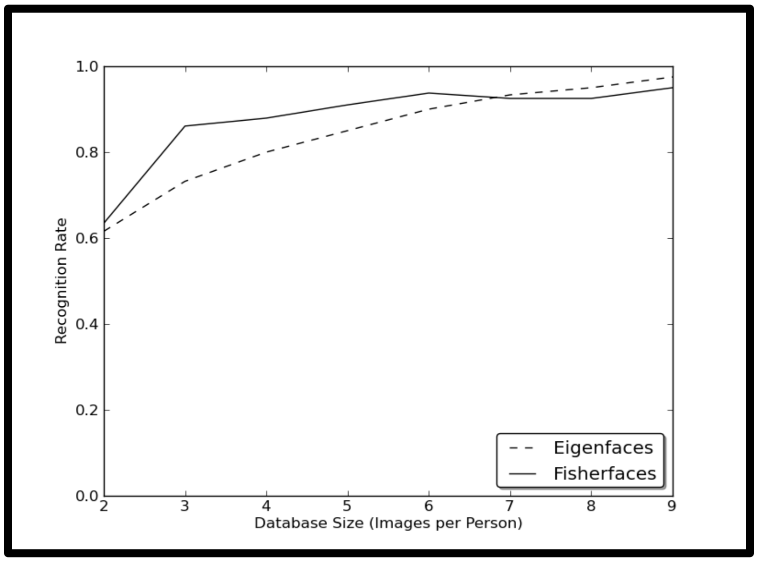


Figure 3- 4 Comparison of PCA and LDA

Keeping in view these advantages of Local Binary Pattern Histogram Analysis over other primitive face recognition algorithms LBPH was chosen to be implemented in the project.

Chapter 4

Project Layout

4.1 Project Design

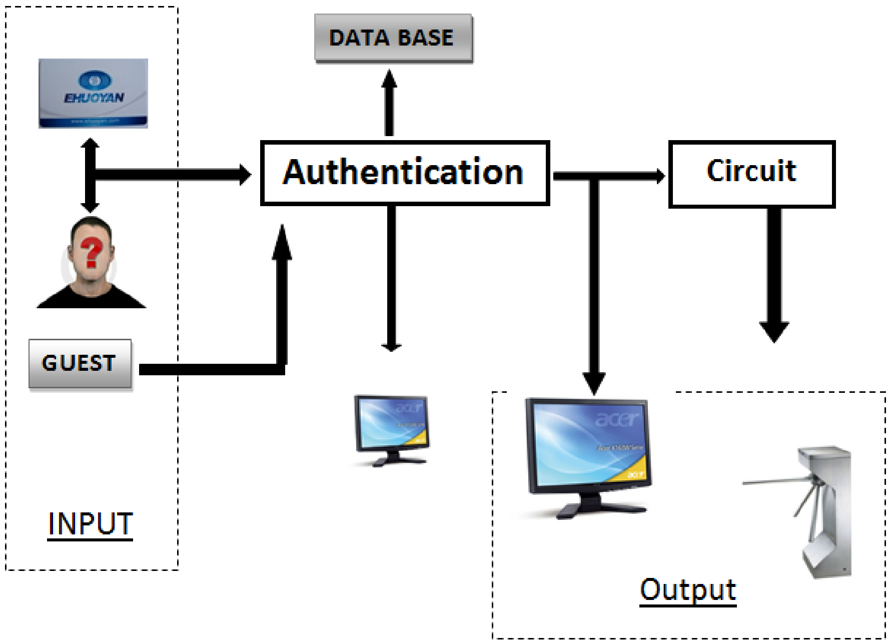


Figure 4- 1 Approach

The project has been implemented as per above design.

- When an unknown person comes for authentication,input from card reader is taken.
- If his data is confirmed by the data base then his face recognition takes place and his data is verified from the facial data base.
- If the person goes through both levels of identification and he is verified and ONLY then he is allowed to pass though the walk through gate. The gate will automatically open for him.
- From the authentication block output is fed into the hardware circuit , which controls the gate .

- Along the hardware circuit output to a display screen is also provided which shows the output that whether the unknown person is recognized and can pass through gate or not .

4.2 Objectives

4.2.1 Academic Objectives

1) To study communication between RF devices
2) To learn different interfacing techniques
3) To learn and understand basics of image processing tools and techniques
4) To acquire appropriate coding language for developing windows application form
5) To be able to learn relevant coding for Microcontroller

4.2.2 Industrial Objectives

1) To provide cost effective solution for security system at entrance.
2) To bring combination of biometric and non-biometric sources based technology in everyday use
3) To develop an intelligent integrated security system prototype that may also be used for asset management, security and attendance system.
4) To provide fool proof security at entrance through intelligent face and card recognition
5) To deliver automated security system that requires no human assistance

4.3 Limitations

- There are certain limitations of the project
- Only one person can enter at a time at the gate. Multiple entries cannot be detected and authorized at the same time.

- The person entering at the gate should maintain a specific distance from the camera for his picture to be taken correctly and get recognized correctly.
- The output of face recognition can vary in extreme weather conditions.
- Entry of person is not allowed without his RFID card.

Chapter 5

Hardware and Software Specifications

5.1 Hardware

5.1.1 Camera selection

Marking recognition accurately demands a good camera with good resolution, pan, and tilt and zoom capability. Good resolution is required for recognition of unknown persons entering at the gates.

- **DSLR (Digital Single Lens Reflex)**

Initially a Digital Single Lens Reflex camera was used for capturing the face image but there were two major drawbacks of using this. Firstly DSLR isn't economically feasible to buy. Secondly it takes pictures manually. It cannot be controlled via controller laptop or computer.

- **IP Camera**

Due to less efficiency of DSLR, IP camera was chosen for face detection but this camera also did not yield accurate results because of poor resolution. Hence facial features were not recognized accurately.

- **Webcam**

Since IP camera had poor resolution, so Logitech sphere AF webcam was chosen. It is equipped with pan, tilt and digital zoom facility. The camera offers 189-degree field of view and 102-degree tilt. It is easy to interface it with visual C++. In poorly lit conditions, the camera is able to adjust intelligently thus producing best possible image.

Due to non-availability of optical zoom it can recognize faces correctly. It can be connected to laptop or PC directly with USB interface.

5.1.2 RFID Reader selection

RFID YHY638A

The said RFID card reader has following specifications which are best suited for the project.

RFID YHY638A Specifications

- Read and write contactless smart cards
- Frequency 13.56MHz
- Typical time to read and write card <100ms
- Communication interface : USB baud rate 9600-115200 bps
- Ambient working temperature : 0C to 60 C
- Operating distance : 0- 70 mm

Figure 5- 1 Ehuoyan YHY638A RFID Reader

5.1.3 Microcontroller Module

The Hardware module of Automated Face Recognition system is controlled via central microcontroller module that is interfaced with the host system. The module does the job of controlling the gate. The microcontroller is interfaced with the host laptop or PC via USB interface. Commands are generated from there and the gate is controlled i-e opened or closed by the microcontroller circuit. A relay is used after the microcontroller for switching purposes.

5.1.4 USB Interface

Project's hardware module communicates with the host system to which the Camera is connected through the USB interface. The microcontroller has been programmed to allow USB interfacing and with minimal hardware components. A stable USB interface is accomplished that controls the entrance system module from a host computer system.

5.1.5 Selection of Gate

- **Barrier**

A barrier is a physical structure that stops entrance of unwanted persons when it is controlled by some electronic means. It is most commonly used for security purposes.

Despite being most commonly used security gate, the barrier is not adopted for this project. The opening / closing time of barrier is very large. The barrier opens slowly and closes slowly. Secondly, more than two persons can pass at a time when the barrier is open. This is highly unfavorable for this project.

Figure 5- 2 Barrier

- **Tripod gate**

Tripod gate is second option to keep unwanted person from entering while using electric means. It allows persons to enter one by one. An RFID card reader is also attached with the tripod gate. Its efficiency is better than electronic barrier and it occupies less space. Since it is costly, so this was option was also not chosen.

Figure 5- 3 Tripod Gate

- **Gate**

A Gate is specially designed for the project. This gate allows only one person to pass at a time. The structure of the gate is made of wood and aluminum. The frame of the door is wooden whereas aluminum and glass is used in door. The height of the door is 6 feet.

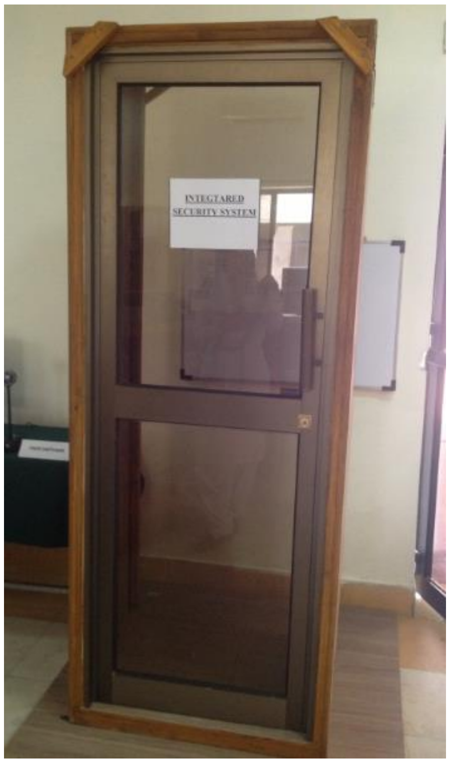


Figure 5- 4 Selected Gate

5.1.6 Mechanism for opening and closing of gate

The door of the gate has an electronic lock. A retractable mechanism has been installed in it that automatically brings the door back to its initial stage after it is opened. This causes the door of gate to close automatically after the person has passed.

If the person is recognized, a command from the host PC is generated and through the microcontroller module to unlock the gate and hence, the gate opens. The gate automatically closes after the person passes. If the unknown person is not recognized by the system, the gate remains closed.

- **Alarm system**

The gate opens if the person is recognized by the integrated security system. If an unknown person tries to enter the gate, it does not open and an alarm system is activated.

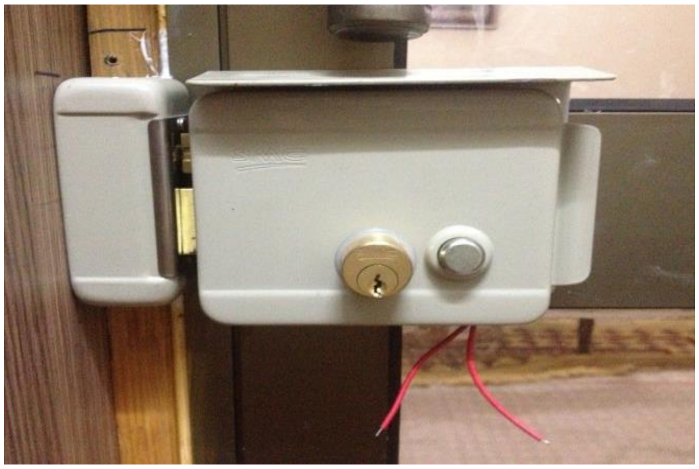


Figure 5- 5 Electronic Lock

5.2Software

5.2.1 MATLAB

MATLAB is one of the best Soft wares used for simulations as it has lot of image processing tools but it is extremely slow when it comes to processing video streams from a large number of cameras.

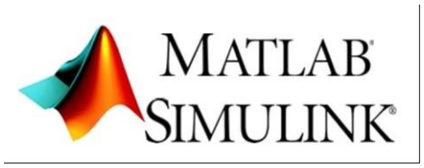


Figure 5- 6 MATLAB

5.2.2 Microsoft Visual Studio

It is used for programming and making GUI of the system. It caters for processing speed. It has tools for programming and debugging C++ code. It provides third party software compatibility.

Figure 5- 7 Visual Studio

5.2.3 OpenCV Library

In Microsoft Visual Studio, OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate it. For the project, the researchers have used OpenCV library in Microsoft Visual Studio.

5.2.4 EMGU CV Wrapper

EMGU CV is also a computer vision library used for real time processes. It has in built commands and functions used in face detection and recognition. This library has been used when developing GUI (graphical user interface) for the project.

5.2.5 Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet).

Chapter 6

Development of the Project

The project Integrated Security System is developed by segmenting the project into number of modules.

6.1 Modules of Project

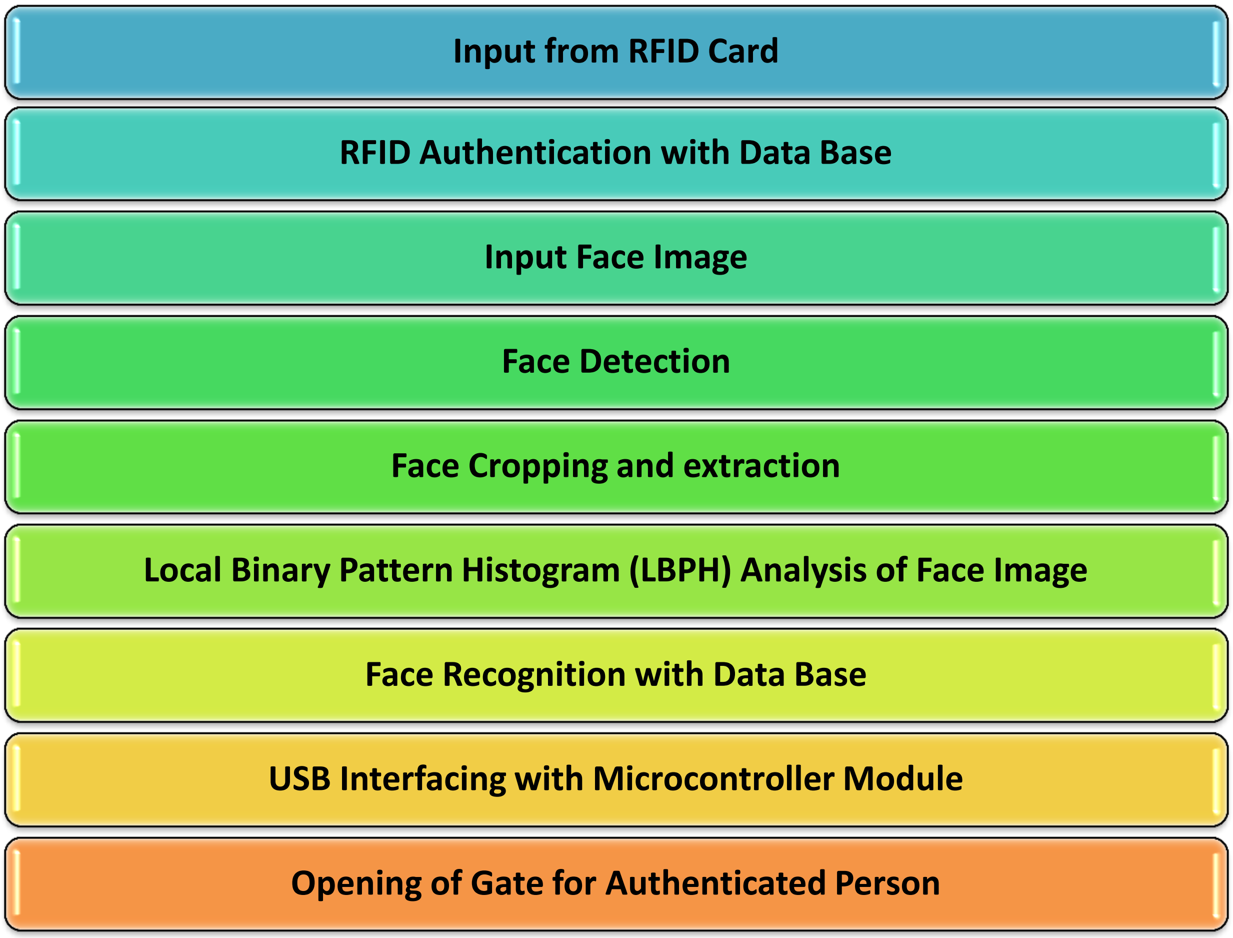

Input from RFID Card
RFID Authentication with Data Base
Input Face Image
Face Detection
Face Cropping and extraction
Local Binary Pattern Histogram (LBPH) Analysis of Face Image
Face Recognition with Data Base
USB Interfacing with Microcontroller Module
Opening of Gate for Authenticated Person

Figure 6- 1 System Modules

- **GUI for the display**

A customized GUI is created for the display at the entrance of the gates. The GUI shows the important instructions for the person entering the gate.

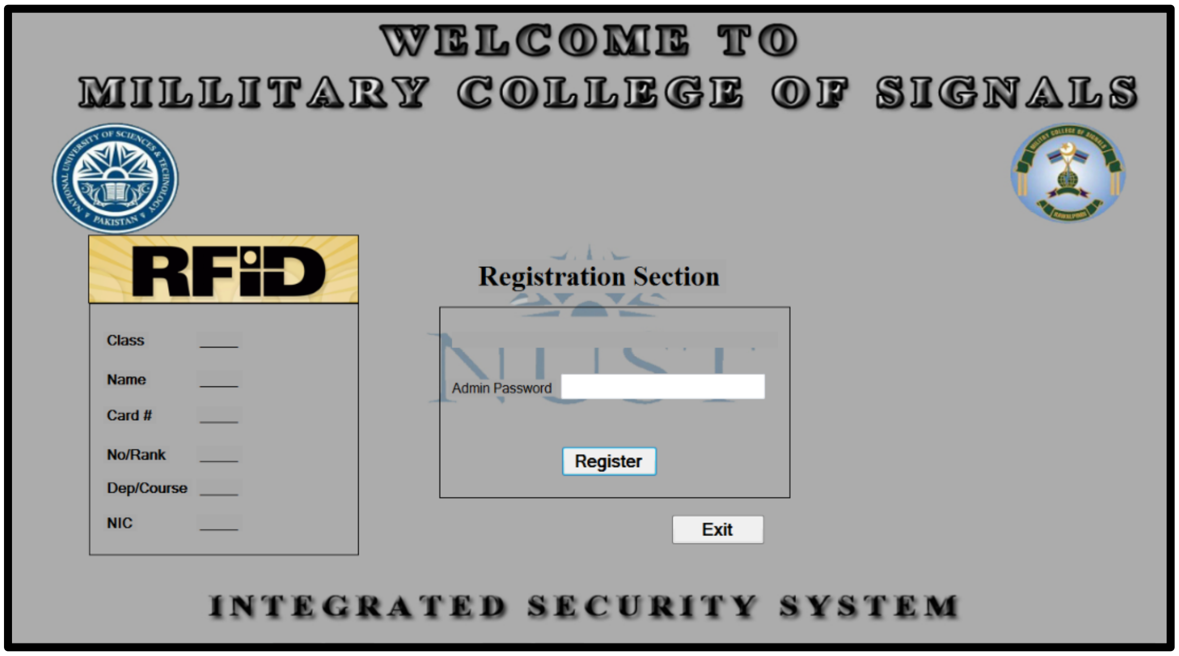


Figure 6- 2 GUI for user

6.1.1Input from RFID Card

When a person enters the gate where Integrated Security System is installed, he has to swipe his card at the RFID card reader for RFID card authentication. FRID cards are issued to each person whose data is stored in the data base.

- **MySql Data Base**

Personal details of each person who are issued a RFID Card are stored and maintained in online MySql data base. This data base gets automatically updated when new person is registered.

6.1.2 RFID Authentication with Data Base

After the person puts his RFID card on the card reader RFID authentication takes place. If the person is registered in the data base, his following details are displayed on the display.

- Name
- Card Number
- Rank
- Course / Dept.
- CNIC Number

These details are entered against each person at the time of registration. If the person is authenticated by the system, his details are displayed.

- **Card Number**

Each RFID card is assigned a different card number. This is a 16 digit hex decimal number that differentiates each card from the other. Each person in the data base holds a unique card number. The face images used for face recognition are stored against this card number in the data base.

6.1.3 Input Face Image

After RFID card authentication, the next step is the face detection and recognition. In this automated integrated security system every step is automated after the card authentication. Image of person is captured automatically after 5 seconds after the card authentication. This time is given to the person to adjust himself and face the camera. The time of 5 seconds can be changed (increased or decreased) as per requirement. This input face image is further used for face detection and then face recognition.

6.1.4 Face Detection

For face recognition, face detection is required. First face detection is done by Viola Jones.

- **Output of Face Detection**

Viola Jones is implemented on each image that is captured from the camera. This algorithm detects the face from the complete image and makes a box around the detected image.

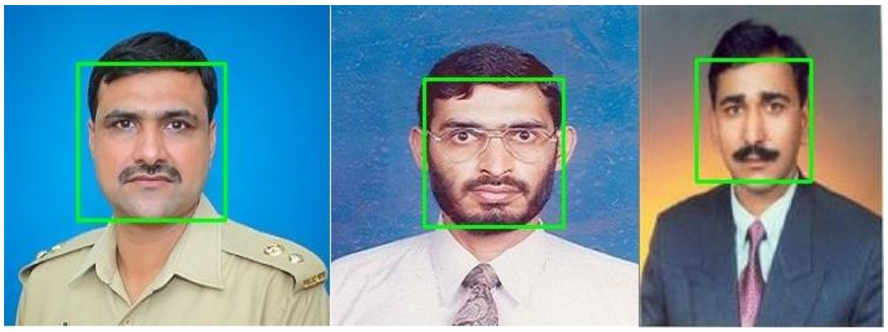


Figure 6- 3 Detected faces in images

6.1.5Cropping and Resizing Of Detected Face

The detected image is cropped to eliminate the background. Each image is then resized to 100x100 pixels. This resize is done to make the pixels homogenous and every image that is taken has same dimensions. This cropped face image gives accuracy of 95%. If the pixels are increased, the processing time of the code also increases. Hence, 100 x 100 pixels are selected. This cropped image is then used in the face recognition process.

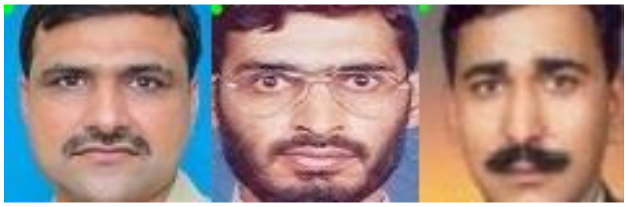


Figure 6- 4 Cropped and resized images

6.1.6Local Binary Pattern Histogram Analysis

Local Binary Pattern Histogram analysis is done for face recognition. In real life scenario false recognitions may occur due to light variations. LBPH analysis encounters this issue by calculating local binary patterns and assigning a binary value to each 3x3 box of pixels. This value is compared for recognition.

Slight variations in expressions also affect the output of the previous algorithms like Eigen Face and Fisher Face methods. LBPH analysis is not affected by slight variations and correct output is given.

The efficiency of PCA (Fisher Face) and LDA (Eigen Face) decreases as the number of face images in data base increases. Efficiency of LBPH is not affected by the number of images in the data base.

6.1.7 Face Recognition with Data Base

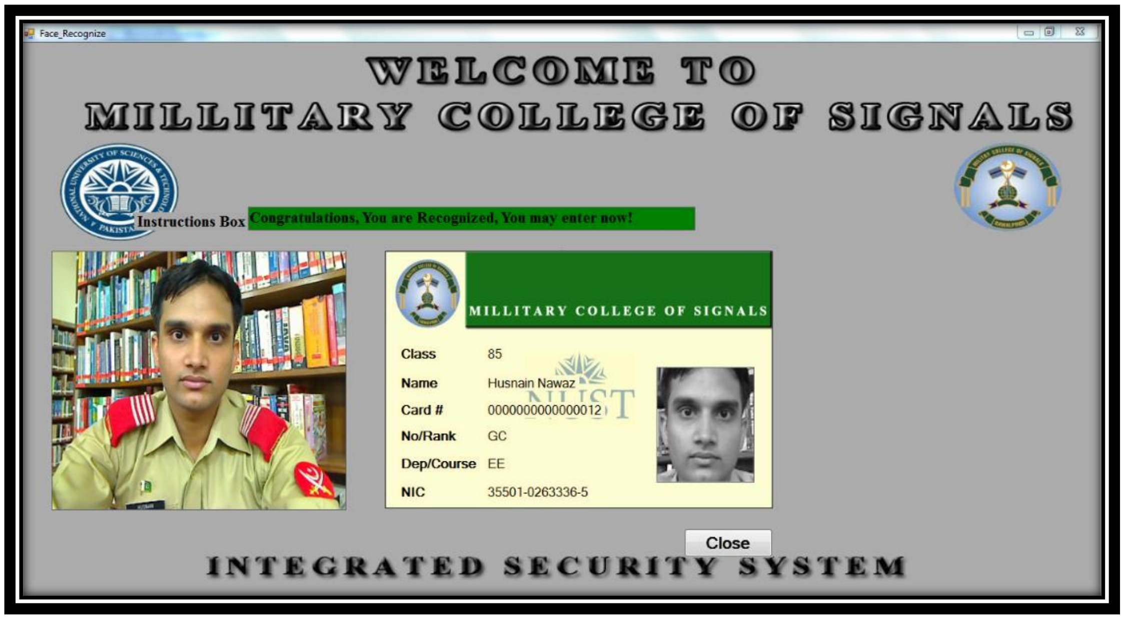
After face detection, the input face image is used for face recognition by Local Binary Pattern Histogram analysis. This image is compared with image in the data base which is taken during the registration phase. LBPH is implemented on the input image to match this image with the images present in the data base against the RFID card number.

- **AND Operation**

AND logic is implemented for the validation of unknown persons. If the person entering the gate has his RFID card authenticated correctly and his face recognized correctly, only then he is rendered authorized by the system and he may pass. If the person does not have his RFID card or his image is not recognized in the data base, he cannot pass through the system.

- **Authorized Person**

After successful card authentication and face recognition the personal details of the said person are shown on the display. And image of person is also shown who is recognized. The person is now authorized and he may pass through the gate. A note saying "Congratulations, You are Recognized, You may enter now! "is displayed on the display screen.



Congratulations, You are Recognized, You may enter now!

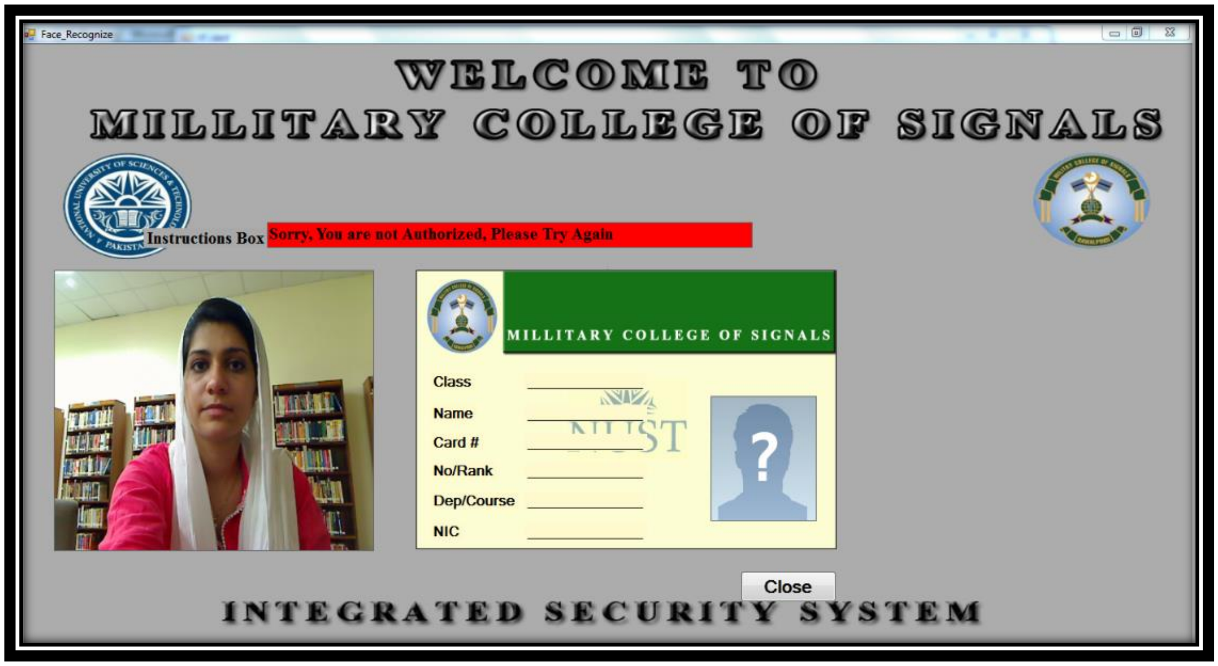
MILLITARY COLLEGE OF SIGNALS

Class	85
Name	Husnain Nawaz
Card #	0000000000000012
No/Rank	GC
Dep/Course	EE
NIC	35501-0263336-5

Figure 6- 5 Authorized Person GUI

- **Unauthorized Person**

If an unknown person is not authorized by face recognition, or he has the wrong card he is not authorized by the system. A note saying “Sorry, You are not authorized, Please Try Again” is displayed on the display screen which is placed in front of the person.


Face_Recognize
WELCOME TO
MILLITARY COLLEGE OF SIGNALS
Instructions Box
MILLITARY COLLEGE OF SIGNALS
Class
Name
Card #
No/Rank
Dep/Course
NIC
Close
INTEGRATED SECURITY SYSTEM

Sorry, You are not Authorized, Please Try Again

Figure 6- 6 Unauthorized Person GUI

6.2 Registration process

One time registration is required for each person who needs to be entered, in the data base. In case of a new person, the registration process is also done after which he may pass through the Integrated Security System. A separate section for registration is made that can only be accessed by the control room personnel. The registration process is completed in number of steps which are explained on the following page.

6.2.1 Registration password

In order to access the registration section and admin password is required. This password is only available to person in the control room who is required to perform the registration process. This password can also be changed as per requirement.

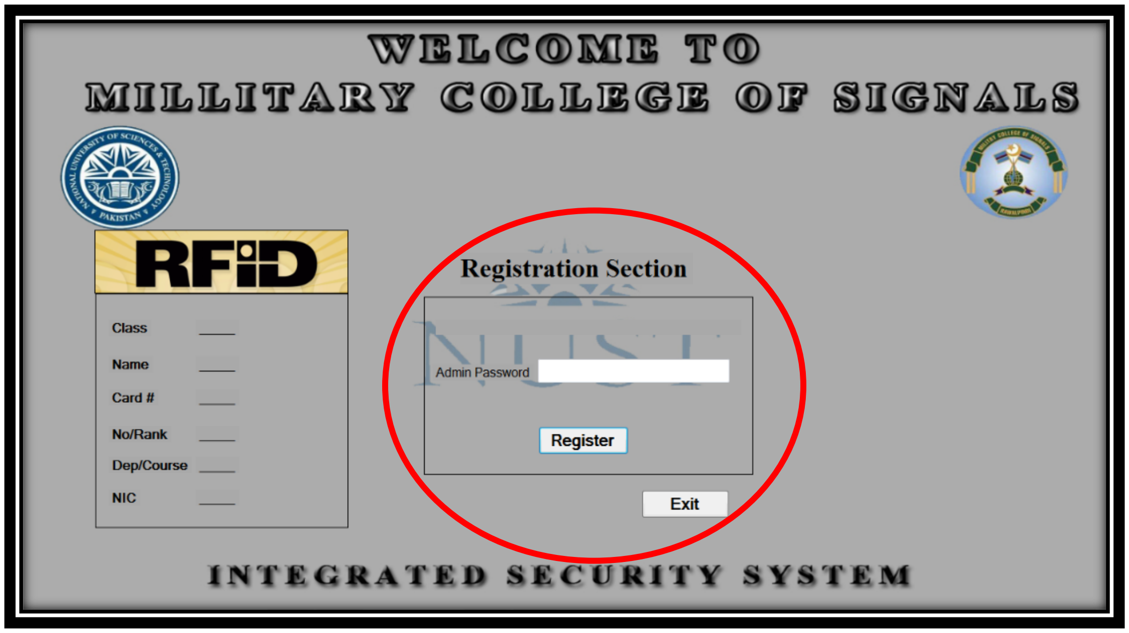
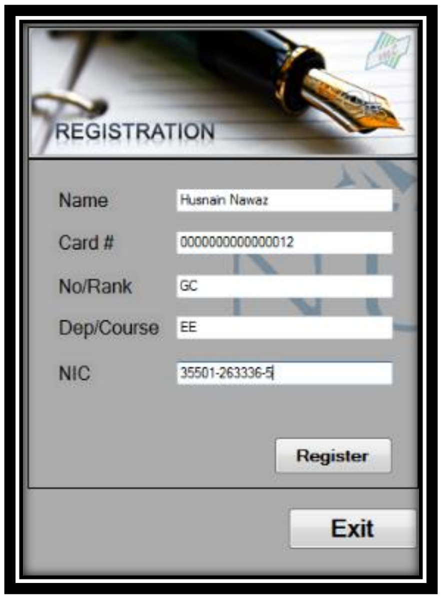


Figure 6- 7 Registration section

6.2.2 Entering details of person

After access to the registration section, personal details of new entry are entered. These details are required to correctly identify each person. The details include name, course, rank, CNIC number and card number. More details may also be added if we want to.



Name	Husnain Nawaz
Card #	0000000000000012
No/Rank	GC
Dep/Course	EE
NIC	35501-263336-5

Figure 6- 8 Entering details of person

6.2.3 Card number

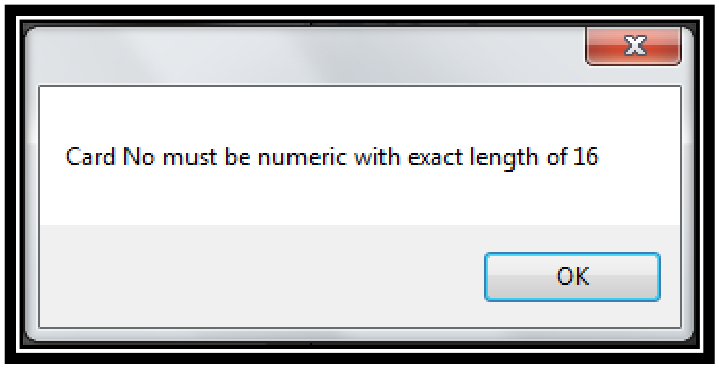
A 16 digit hex decimal card number is assigned to each RFID card belonging to each person. This number is not repeated for any two persons.

- **Avoiding fake registrations**

Certain measures are implemented to avoid any fake registrations.

- **Card number to be exact 16 digits**

If card number of less than 16 digits is entered, a message is displayed and the details are not updated in the data base.

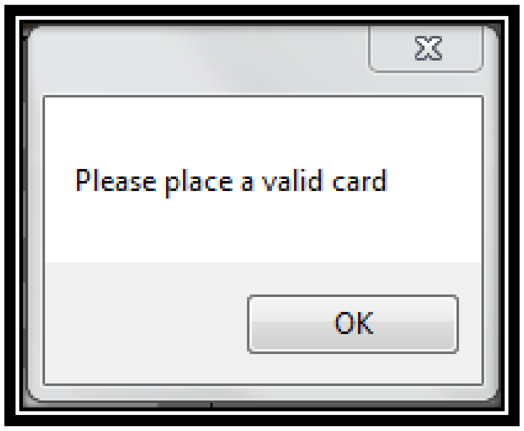


Card No must be numeric with exact length of 16

Figure 6- 9 Output message when wrong card number is entered

- **Placement of RFID card on the card reader**

RFID card should be placed on the card reader. If someone tries to register without a proper RFID card a message is displayed on display screen and proper registration is not complete until an RFID card is placed on the card reader.

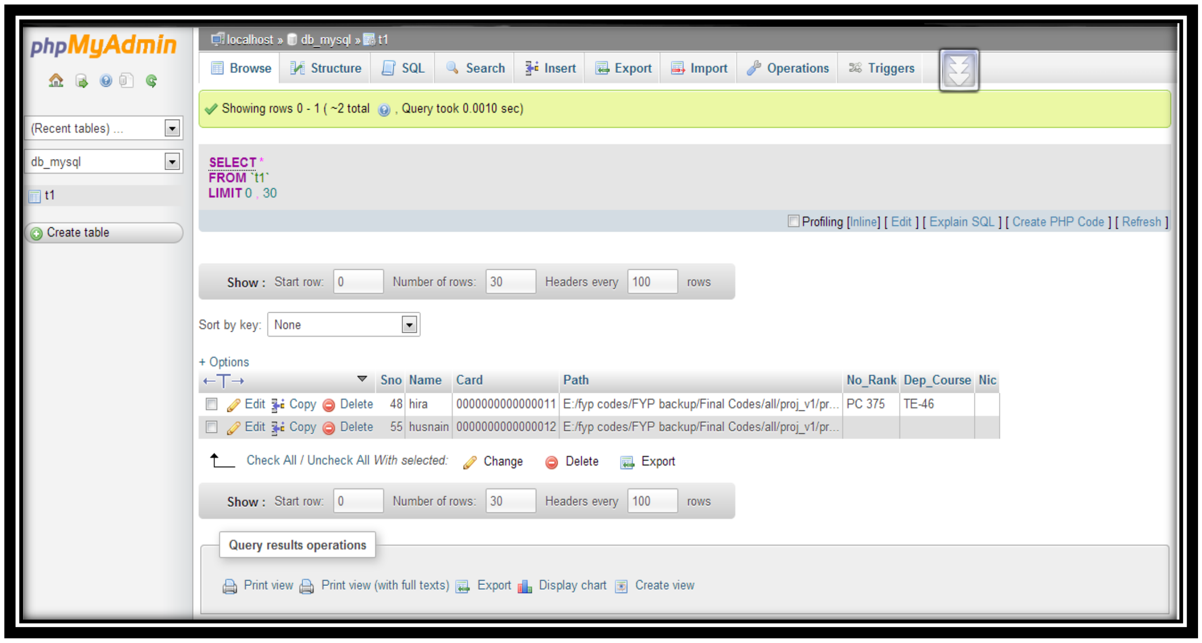


Please place a valid card

Figure 6- 10 Output message when RFID card is not placed on card reader

6.2.4Update of details in MySql

Until this point the RFID card details are entered. The details are automatically updated in the MySql data base.



Showing rows 0 - 1 (~2 total , Query took 0.0010 sec)

```
SELECT *
FROM `t1`
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	Sno	Name	Card	Path	No_Rank	Dep_Course	Nic
Edit Copy Delete	48	hira	0000000000000011	E:/fyp codes/FYP backup/Final Codes/all/proj_v1/pr...	PC 375	TE-46	
Edit Copy Delete	55	husnain	0000000000000012	E:/fyp codes/FYP backup/Final Codes/all/proj_v1/pr...			

Check All / Uncheck All *With selected:* Change Delete Export

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Figure 6- 11Updating of MySql database

6.2.5 Training images

Training images of each person are required for face recognition. These are required only one time when completing the registration. Greater the number of training faces greater is the efficiency of the algorithm but large number of training images requires more time for computation.

8 training images are taken for each person. Each image is taken with different facial expression. Looking right, looking left, looking up, looking down, looking straight, smiling face and frowning etc. These 8 images are stored in the data base against the details of each person in the data base. They are used to compare the face of person with the input face image that is taken after the RFID card authentication.

After these steps are performed the new person is issued his personal RFID card and he is allowed to enter through the system now.

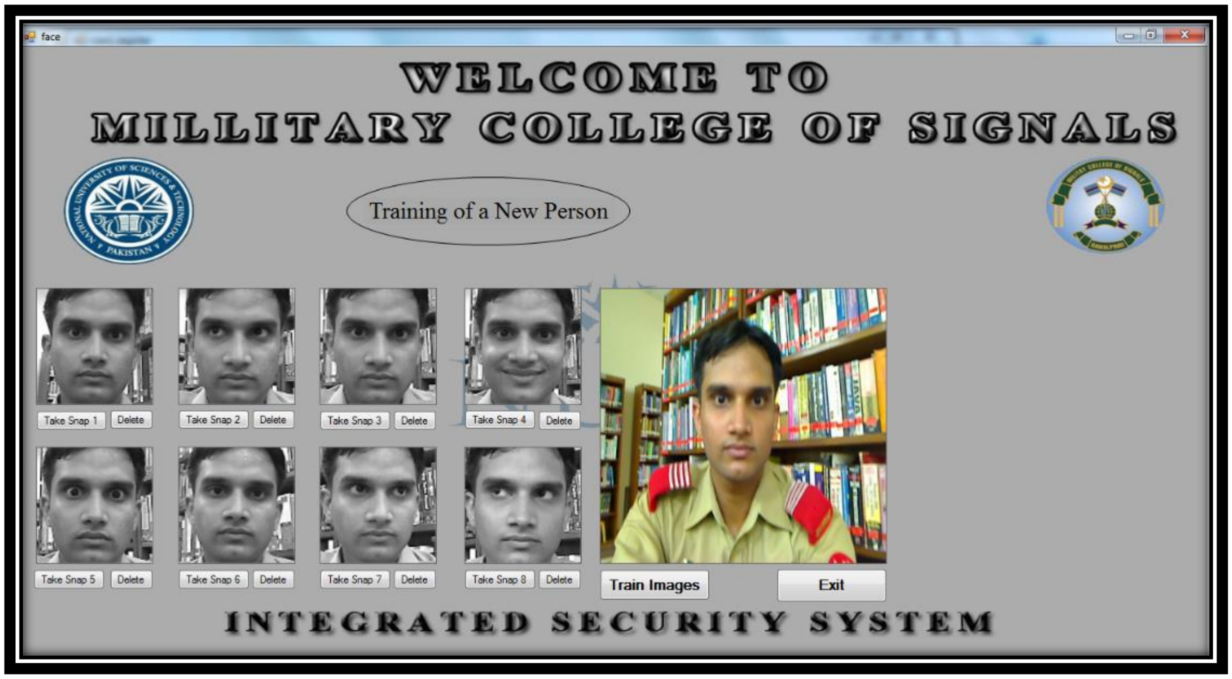


Figure 6- 12 Training Images

6.3 USB Interfacing with Microcontroller Module

If the person is declared authorized by the system he may pass through the gate. The hardware gate is interfaced to the control PC with the help of a microcontroller circuit. The circuit is connected to the control PC by USB to serial port connection.

A separate code is run that initializes the USB port. Port 11 has been selected for the project with baud rate of 9600 bps to send data from USB to serial port. The serial port is connected to the hardware circuit and command is generated. This command is passed to the circuit which causes the electronic lock to open again.

6.4 Opening of Gate for Authenticated Person

6.4.1 USB to serial connection

Command from host PC is sent from the PC to the electronic circuit via a USB to serial interface. This command is forwarded to the gate via a series of components explained below.DB9 cable is used which has a USB port at one end and serial interface at the other end.

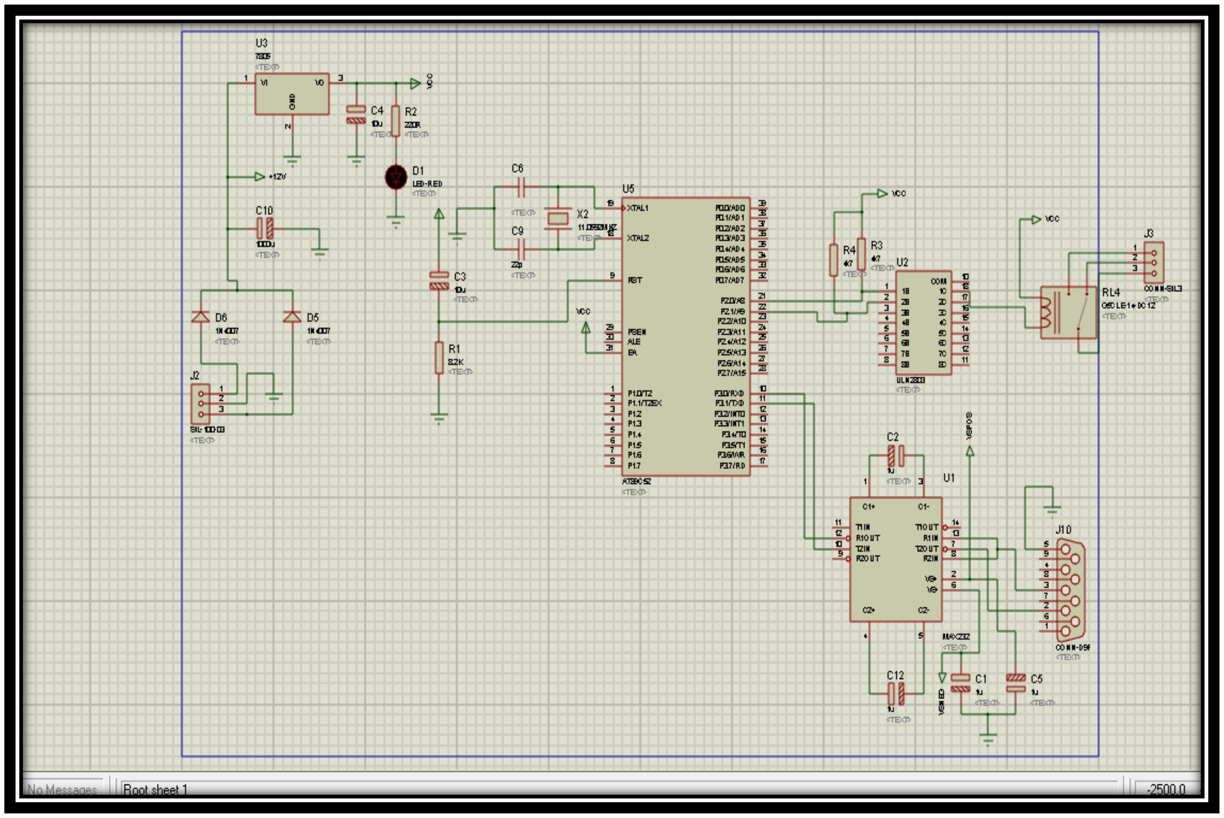


Figure 6- 13 Hardware Circuit Simulation

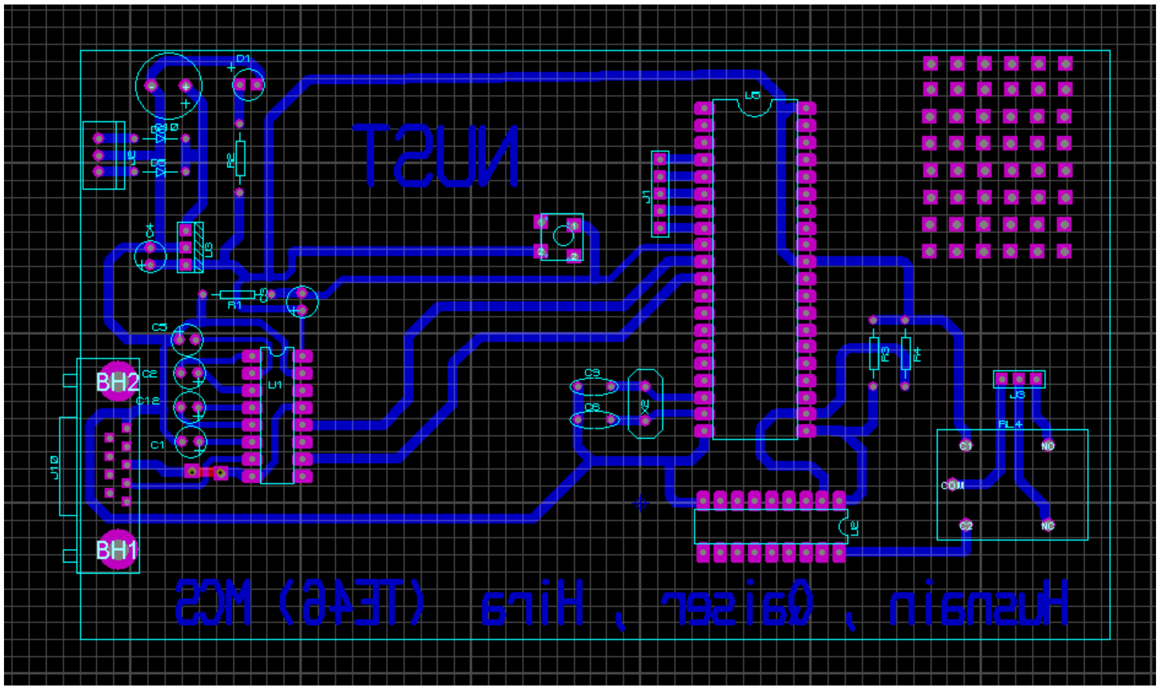


Figure 6- 14 PCB Layout

6.4.2 Power Supply

A step down transformer is used at the start of the circuit. Since the microcontroller based circuit does not work on 220 volts AC so it needs to be convert it to DC voltage. First 220 volts AC are converted to 24 volts AC and then to 24 volts DC.

6.4.3 Rectification Circuit

After getting 24 volts AC, a rectifier circuit is used. It consists of number of diodes. These diodes perform function of rectification. After rectification, 17Volts(RMS) DC is provided.

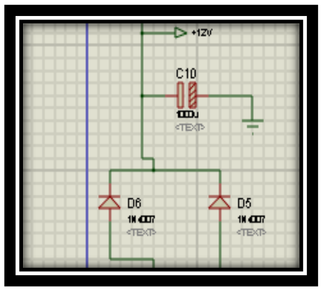


Figure 6- 15 Rectification Circuit

6.4.4 Capacitors

After the rectifier diode capacitors are attached to block any unwanted AC supply to enter the microcontroller.

6.4.5 Regulator

The 17 volts DC is fed input to the regulator. Regulator 7805 is used in the circuit. It converts 17 volts DC to 5 volts DC.17 volts DC is too high for the IC MAX 232.

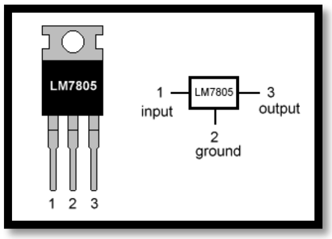


Figure 6- 16 Regulator 7805

6.4.6Pull up Resistors

Pull up resistors are used after the regulator to maintain a specific level of voltage in case any high impedance is introduced.

6.4.7MAX 232

Max 232 is used as a voltage level converter. Its basic function is to convert signal from serial port to logic which is used by the microcontroller. The microcontroller use TTL logic. It acts as interface between RS-232 serial ports to TTL logic. TTL logic works on 0 and 5volts.The MAX232 is a two pairs of driver/receiver and converts the RX, TX, CTS and RTS signals.

The drivers are used to give RS-232 voltage level outputs of ± 7.5 V from + 5 V supply. The receivers decrease RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels.

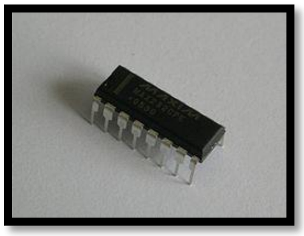


Figure 6- 17 MAX232 IC

6.4.8 AT89C52 Microcontroller

The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K Bytes of Flash programmable and erasable read only memory (PEROM). Advantage of this device is that it is compatible with 80C51 and 80C52. It lets the program memory to be reprogrammed in-system. The Atmel AT89C52 is a powerful microcomputer which provides basic purposes. It is easy to operate and is highly-flexible in terms of its functionality.

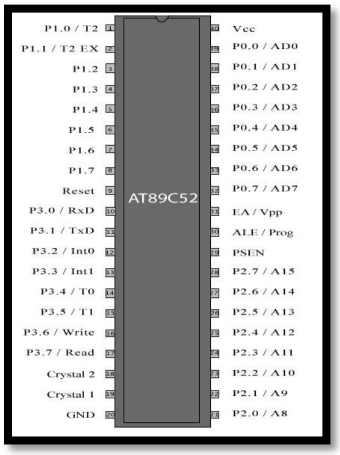


Figure 6- 18 AT89C52 Microcontroller

AT89C52 Microcontroller has

- 2 External interrupts
- 3 Timer interrupts
- 1 Serial interrupt

The timers in 89C52 perform the functions of providing time delay, frequency generation and sequencing. It has a TMOD register that acts as a counter. AT89C52 has 8K Bytes ROM that also provides reusable RAM and ROM.

Port 1, 2, 3 are used for input/output. Reset circuit is connected to pin 9 of the microcontroller.

Keeping all these advantages in mind AT89C52 was selected to be used in circuit. Pin 20 is attached to ground, and pin 40 to +5 voltage source. Input from MAX 232 is given to pin 10 and 11 of the microcontroller. These are used to receive data from serial port and send data to serial port respectively.

Data from PC is sent to the hardware circuit through serial port.AT89C52 takes input from the serial port through the pins 10 and 11.

6.4.9 Initializing serial port

Serial port is initialized and input from the host PC is sent to the circuit through the serial port.

- **Baud rate**

Baud rate is set to 9600 bps. This baud rate is also set while sending data to the serial port from the PC to the circuit.

6.4.10 Oscillator

An 11.05 MHZ oscillator is used with the microcontroller. It is connected to pin 18 and 19 of the microcontroller. It is used for frequency generation and providing time delay.

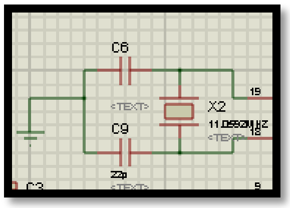


Figure 6- 19 Oscillator Circuit

6.4.11 ULN2803 High Voltage, Darlington Transistors Array

IC ULN2803 is used for amplification. This eight NPN Darlington has connected sequence of transistors which are used for interfacing between low logic level digital circuitry (NMOS, PMOS / CMOS) and the higher current/voltage requirements and electronic applications.

It uses Darlington pair for amplification. Darlington pair is designed to amplify signal so it can be detected in next stage of the circuit. Output from microcontroller is given into this IC 2803.

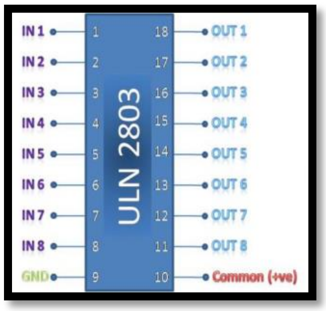


Figure 6- 20 ULN2803 IC

This IC is used after the microcontroller AT89C52 .Input to the IC ULN2803 is given through pins 1B and 2B from the output pins of the microcontroller

The low voltage signal from the microcontroller is converted to high voltage signal for the next stage of the circuit that is the relay. This IC is used for the interfacing between microcontroller and the relay.

Another advantage of IC 2803 is to protect the circuit from back emf. It protects the back emf to enter the microcontroller as it is dangerous to the proper functioning of the microcontroller.

6.4.12 Relay G2R-14-DC12

Relay is connected from the output pin 1C of ULN2803. Relay performs the purpose of switching. It is initially in “Normally Open” state. When input to the microcontroller is high, this signal is transmitted to the relay through the ULN2803 IC. The relay then changes it state.

Figure 6- 21 Relay

The relay is in open state initially. When high input is transmitted to the relay from the microcontroller via IC ULN2803 the relay switches and causes 12volts supply to be transmitted to the input of the electronic gate. Hence, the lock opens without any human assistance and the person can pass through the gate.

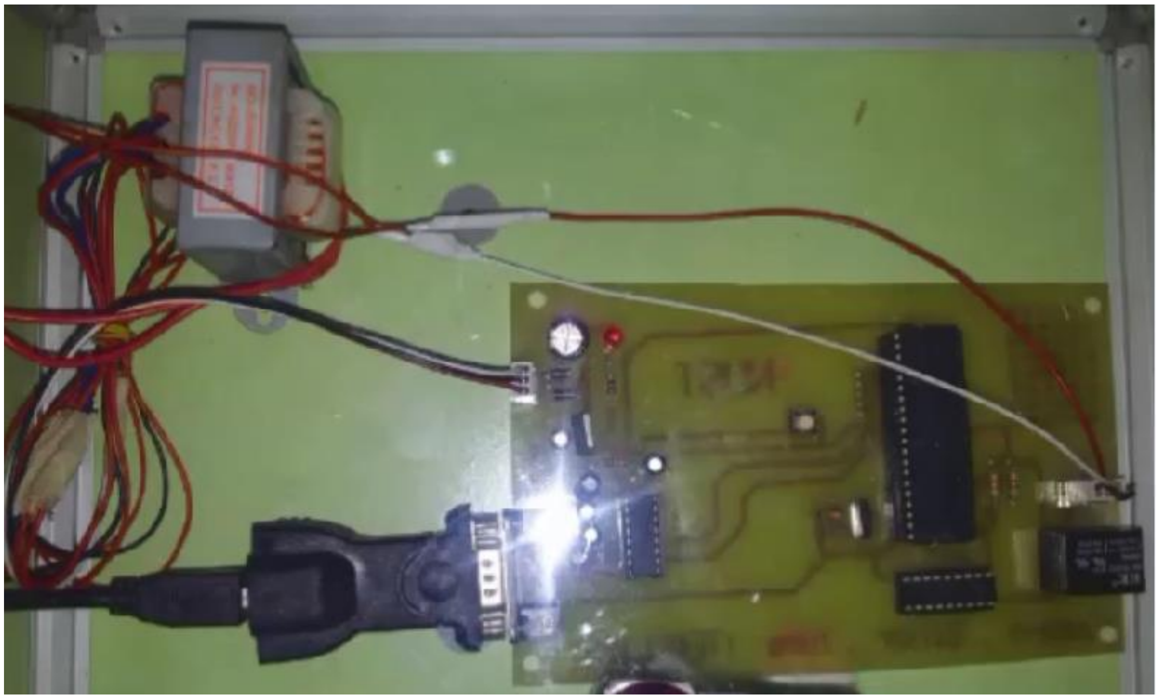


Figure 6- 22 Hardware Circuit

Chapter 7

Future Research and Conclusions

7.1 Analysis

We defined certain specifications of the project. Here we shall analyze whether we have delivered our systems to fulfill those aims.

7.1.1 Real Time System

The project provides real time system. It can be implemented practically at the entrance and used for in effect security.

7.1.2 Least False Recognitions

The project uses RFID card and face recognition for authentication. Integration of these two provide least chances of false recognitions.

7.1.3 Time Saving

This is saved by using Integrated Security System. Manual checking at the gates can be troublesome, hectic and time consuming.

7.1.4 Automated System

A complete automated system is delivered that requires no human assistance at the gates.

7.1.5 Loss of Human Life Is Avoided

In case of any security attack the loss is human life is avoided. Manual systems posed great threat to the security guards at the entrance of gates.

7.1.6 No Limitation on the Number of Entries in Data Base

There are no limitations on the number of entries that can be added to the database. The length of database can be increased or decreased as per our requirements.

7.1.7 No Tampering

As the system can only be accessed by the authorized persons who are present in the control room no tampering of database is possible by some unknown person.

7.2 Conclusion

The project Integrated Security System provides fool proof security at the entrance. We have implemented this project at the entrance of the gate. It encompasses proficient use of bio-metric and non-biometric sources. A customized gate is designed for the project that has automatically opening and closing system.

The project effectively uses the modern computer vision techniques for face recognition which provides 90% accuracy. The time for complete authentication of person can be changed as per requirements. It provides a user friendly interface for the person entering at the gate.

7.3 Future Enhancements

The said project may be used as a base for future research. The researchers may integrate it with other schemes in modules to provide enhance security measures. In this project face recognition as a biometric source of identification has been used but other researchers may also incorporate iris identification and palm recognition with the existing system. Besides, different non biometric sources like signature recognition and QR Codes may also be integrated. A centralized database of entire institute may also be maintained by recording the entrance time and departure time of each individual in an institute. This can also be used to control access of individuals in an institute or department.

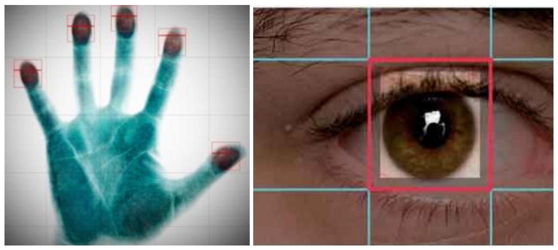


Figure 7- 1 Increased Biometric inputs

Figure 7- 2 Signature Authentication

BIBLIOGRAPHY

[1] http://www.slideshare.net/shweta-sharma99/biometrics-11109566

[2] http://www.technovelgy.com/ct/Technology-Article.asp?ArtNum=12

[3] Implementing the Viola-Jones Face Detection Algorithm by Ole Helvig Jensen

[4] http://www.hanvon.com/en/products/FaceID/downloads/FAQ.html

[5] http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#fisherfaces

[6] Multi-scale Local Binary Pattern Histogram for Face Recognition Chi Ho CHAN

[7] Ojala, T.,Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. 24(7), 971–987 (2002)

[8] http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

[9] http://en.wikipedia.org/wiki/Radio-frequency_identification

[10] http://www.brooks.com/applications-by-industry/rfid/rfid-basics/rfid-advantages

[11] http://www.webopedia.com/DidYouKnow/Computer_Science/2005/rfid.asp

[12] Computational and performance aspects of PCA-based face-recognition algorithms by Hyeonjoon moon and P Jonathan Phillips

APPENDIX A-1

RFID Authentication Code

```
#pragmaonce

#include"Face_Recognize.h"
#include"card_register.h"

namespace proj_v1 {

	usingnamespace System::Collections::Generic;
	usingnamespace System::ComponentModel;
	usingnamespace System::Text;
	usingnamespace System;
	usingnamespace System::ComponentModel;
	usingnamespace System::Collections;
	usingnamespace System::Windows::Forms;
	usingnamespace System::Data;
	usingnamespace System::Drawing;
	using System::String;
	using System::Byte;
	//Thread ^currentTimeThread;
	//#using <MasterRD.dll>
	usingnamespace System::Runtime::InteropServices;
	usingnamespace MySql::Data::MySqlClient;


	publicrefclass rf_card : public System::Windows::Forms::Form
	{
	//[DllImport("E:\\sohaib\\vc proj\\RF card 
reader\\RF_card\\MasterCom.dll")];
	[DllImport("kernel32.dll")] staticvoid Sleep(int num);
	[DllImport("MasterRD.dll")] staticint rf_init_com(int port, 
long baud);
	[DllImport("MasterRD.dll")] staticint rf_ClosePort();
	[DllImport("MasterRD.dll")] staticint 
rf_M1_read(unsignedshort icdev, unsignedchar Block, IntPtr pData, 
unsignedchar* pLen);
	[DllImport("MasterRD.dll")] staticint rf_antenna_sta(short 
icdev, unsignedchar mode);
	[DllImport("MasterRD.dll")] staticint rf_init_type(short 
icdev, unsignedchar type);
	[DllImport("MasterRD.dll")] staticint rf_request(short 
icdev, unsignedchar mode, unsignedshort* pTagType);
```

```
	[DllImport("MasterRD.dll")] staticint rf_anticoll(short 
icdev, unsignedchar bcnt, IntPtr pSnr, unsignedchar* pRLength);
	[DllImport("MasterRD.dll")] staticint rf_select(short icdev, 
IntPtr pSnr, unsignedchar srcLen, unsignedchar* Size);
	[DllImport("MasterRD.dll")] staticint 
rf_M1_authentication2(short icdev, unsignedchar mode, 
unsignedchar Block, IntPtr pLen);
	[DllImport("MasterRD.dll")] staticint rf_M1_write(short 
icdev, Byte Block, IntPtr pLen);



	private: Microsoft::VisualBasic::PowerPacks::ShapeContainer^  
shapeContainer1;
	private: Microsoft::VisualBasic::PowerPacks::RectangleShape^  
rectangleShape1;
	private: Microsoft::VisualBasic::PowerPacks::RectangleShape^  
rectangleShape2;
	private: Microsoft::VisualBasic::PowerPacks::RectangleShape^  
rectangleShape4;

		delegatevoid CurrentTimeDelegate();
		Delegate^ CurrentTime;
		Thread ^currentTimeThread;

	bool found;
	bool registeration;
	private: System::Windows::Forms::Button^  closebtn;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::TextBox^  textBox2;


			bool closing;
	public:

		rf_card(void)
		{
			InitializeComponent();
			//initializereader();
			found=false;
			registeration=false;
			CurrentTime = 
gcnewCurrentTimeDelegate(this,&rf_card::UpdateCurrentTime);
			//
			//TODO: Add the constructor code here
			//
```

```
            }

            void UpdateCurrentTime()
         {
                       initializereader();
                       int status=1;  int i;
                       unsignedint secnr = 0x02;

                       // AUTHENTICATE
                       IntPtr keyBuffer =
Marshal::AllocHGlobal(1024);


                       //char st[12]=
{'f','f','f','f','f','f','f','f','f','f','f','f'};
                       //unsigned char* bytesKey =
ToDigitsBytes(&st[0]);
                       //int leng = sizeof(st) / 2 + (((sizeof(st)
% 2) > 0) ? 1 : 0);
                       //int leng=6;
                       unsignedshort icdev = 0x00;
                       unsignedchar mode = 0x60;

                       //unsigned char abc[10];
                       //for (i=0;i<leng;i++)
                       //    abc[i]= (unsigned char) *(bytesKey+i);

                       for (i=0;i<6;i++)
                            Marshal::WriteByte(keyBuffer, i, 255);
                       status = rf_M1_authentication2(icdev, mode,
(Byte)(secnr * 4), keyBuffer);
                       Marshal::FreeHGlobal(keyBuffer);
                       if (status != 0)
                       {
                            //MessageBox("rf_M1_authentication2
failed!!");

                            return;
                       }


                       IntPtr dataBuffer =
Marshal::AllocHGlobal(1024);
                  //    for (i=0;i<=3;i++)
                       //{
                           i=1;
                            int j;
```

```
                                        unsignedchar* cLen = newunsignedchar;
                                        //unsigned int* temp = (unsigned
char*) cLen;
                                        //cLen = Convert::ToByte(cLen);

                                        status =
rf_M1_read(icdev,(Byte)((secnr * 4) + i), dataBuffer,  cLen);
                                        if ( (status != 0) | (*cLen != 16))
                                        {

        Marshal::FreeHGlobal(dataBuffer);
                                                rf_ClosePort();
                                                return;
                                        }
                                        Byte bytesData[15];
                                        for (j=0;j<=14;j++)
                                                bytesData[j] =
Marshal::ReadByte(dataBuffer,j);
                                        //Dim cnic As Int64
                                //      if (i == 0)
                                //              textBox1->Text =
ToHexString(bytesData)->ToString();
                                //      if (i == 1)
                                        String ^card = ToHexString(bytesData)-
>ToString();

                                        card = card->Remove(16);
                                                c_no->Text = card;

                                        delete cLen;
                        //      }
                                Marshal::FreeHGlobal(dataBuffer);
                                //change 4
                                cli::array<String^,1> ^arr = gcnew
cli::array<String^,1>(7);
                                String^ ab=
"server=localhost;Database=db_mysql;User ID=root;Password=''";
                                MySqlConnection^ conSQL =
gcnewMySqlConnection(ab);
                                conSQL->Open();
                                MySqlCommand ^command = conSQL-
>CreateCommand();
                                command->CommandText = "SELECT * FROM t1
WHERE Card='"+card+"';";
                                MySqlDataReader^ Reader;
                                Reader = command->ExecuteReader();
                                if (Reader->Read())
                                 {
                                      found=true;
                                       i=0;
                                      while(i<Reader->FieldCount)

                                       {
                                           arr[i] = Reader->GetString(i);
                                           i++;
                                       }
                                 }
                                // add more
                                p_no->Text=arr[0];
                                p_name->Text=arr[1];
                                r_no->Text=arr[4];
                                d_no->Text=arr[5];
                                nic_no->Text=arr[6];
                                //    add more
                                 Reader->Close();
                                conSQL->Close();
                                if (found==true)
                                {
                                      this->currentTimeThread->Abort();
                                      this->currentTimeThread->Join();
                                      Form^ face_rec= gcnew
Face_Recognize(Convert::ToInt32(p_no->Text));
                                      face_rec->Show();
                                }
                                //~Face_Recognize();
                                found = false;
                                Thread::CurrentThread->Sleep(2000);
                                //rf_ClosePort();
                  }

void DisplayCurrentTime()
     {
                     while (true)
                     {
                                Thread::CurrentThread->Sleep(100);
                                this->Invoke(CurrentTime);
                                if(closing==true)
                                      break;
                         }
              }
                String^ ToHexString(Byte bytes[])
                     {
```

```
                    staticchar hexDigits[] = { '0', '1', '2',
'3', '4', '5', '6', '7',
                    '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
                    char* chars = newchar[15*2];
                    for (int i = 0; i < 15; i++)
                    {
                          int b = bytes[i];
                          chars[i * 2] = hexDigits[b >> 4];
                          chars[i*2+1] = hexDigits[b & 0xF];
                    }
                    returngcnew String(chars);
              }



        void initializereader()
        {

              short icdev = 0x00;
              int status;

              status = rf_init_com(2, 14400);
              unsignedchar type = Convert::ToByte('A');//
    'mifare one
              unsignedchar mode = 0x52;
              unsignedshort*  TagType = newunsignedshort;
              unsignedchar bcnt = 0x04;
              // 'mifare
              IntPtr pSnr;
              unsignedchar* len = newunsignedchar;
              unsignedchar* size = newunsignedchar;

        //    If Not bConnectedDevice Then
              pSnr = Marshal::AllocHGlobal(1024);
              int i;
              for (i=0;i<=1;i++)
              {
                    status = rf_antenna_sta(icdev, 0);
                    if (status != 0)
                          continue;
                    Sleep(20);
                    status = rf_init_type(icdev, type);
                    if (status != 0)
                          continue;
                    Sleep(20);
                    status = rf_antenna_sta(icdev, 1);
```

```
                        if (status != 0)
                              continue;
                        Sleep(50);
                        status = rf_request(icdev, mode, TagType);
                        if (status != 0)
                              continue;
                        status = rf_anticoll(icdev, bcnt, pSnr,
len);
                        if (status != 0)
                              continue;
                        status = rf_select(icdev, pSnr, *len,
size);
                        if (status != 0)
                              continue;
                        unsignedint* szBytes =
newunsignedint[*len+1];
                        String^ str =
Marshal::PtrToStringAnsi(pSnr);
                        int size_str = str->Length;
                  }
                  Marshal::FreeHGlobal(pSnr);
            }



            Byte GetHexBitsValue(Byte ch)
            {
                  Byte sz = 0;
                  if (ch <= '9'&& ch >= '0')
                        sz = (Byte)(ch - 0x30);
                  if (ch <= 'F'&& ch >= 'A')
                        sz = (Byte)(ch - 0x37);
                  if (ch <= 'f'&& ch >= 'a')
                        sz = (Byte)(ch - 0x57);
                  return sz;
            }




            unsignedchar* ToDigitsBytes(unsignedchar* theHex)
            {
                  //int leng = (sizeof(theHex) / 2) +
(((sizeof(theHex) % 2) > 0) ? 1 : 0);
                  //unsigned char temp[leng];
                  int leng = 16;
```

```
                        unsignedchar* bytes = newunsignedchar[];
                        for (int i = 0; i < leng; i++)
                        {
                                char lowbits = theHex[i*2];
                                char highbits;
                                if ((i * 2 + 1) < 32)
                                        highbits = theHex[i * 2 + 1];
                                else
                                        highbits = '0';
                                int a =
(int)GetHexBitsValue((Byte)lowbits);
                                int b =
(int)GetHexBitsValue((Byte)highbits);
                                *(bytes+i) = (Byte)((a << 4) + b);
                        }
                        //unsigned char* bytes = &temp;
                        return bytes;
                }



        protected:
                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                ~rf_card()
                {
                        rf_ClosePort();
                        if (components)
                        {
                                delete components;
                        }
                }

private: System::Windows::Forms::Button^  button2;
private: System::Windows::Forms::Label^  label4;
private: System::Windows::Forms::Label^  label5;
private: System::Windows::Forms::Label^  label6;
private: System::Windows::Forms::Label^  p_no;
private: System::Windows::Forms::Label^  c_no;
private: System::Windows::Forms::Label^  p_name;



        protected:
```

```
     private:
          /// <summary>
          /// Required designer variable.
          /// </summary>
          System::ComponentModel::Container ^components;

#pragmaregion Windows Form Designer generated code
          /// <summary>
          /// Required method for Designer support - do not
modify
          /// the contents of this method with the code editor.
          /// </summary>
          void InitializeComponent(void)
          {
               System::ComponentModel::ComponentResourceManager^
resources = (gcnew
System::ComponentModel::ComponentResourceManager(rf_card::typeid)
);
#pragmaendregion
     private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e)
                {
                }
     private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e)
                {
                    if (passt->Text=="mcsnust")
                     {
                    this->currentTimeThread->Abort();
               this->currentTimeThread->Join();
                                Form^ card_reg= gcnew
card_register();
                                card_reg->Show();
                     }
                    // else
                         //
     MessageBox::Show("Sorry, You Have entered wrong password");
                    this->textBox2->Text="Sorry, You have
entered wrong Password";

          }

private: System::Void closebtn_Click(System::Object^  sender,
System::EventArgs^  e)
           {
               //bool asb=cam->Grab();
               closing=true;

               //
               this->currentTimeThread->Abort();
               this->currentTimeThread->Join();
               this->Close();
           }
private: System::Void rf_card_Load(System::Object^  sender,
System::EventArgs^  e)
           {
           }
private: System::Void rf_card_Enter(System::Object^  sender,
System::EventArgs^  e)
           {
           }
private: System::Void rf_card_Activated(System::Object^  sender,
System::EventArgs^  e) {
               currentTimeThread = gcnew Thread(gcnew
ThreadStart(this,&rf_card::DisplayCurrentTime));
//starts the thread
               Thread::CurrentThread->Sleep(100);
               currentTimeThread->Start();
           }
};
}
```

APPENDIX A-2

Face Detection Code

```
#pragmaonce

namespace proj_v1 {

	usingnamespace System;
	usingnamespace System::ComponentModel;
	usingnamespace System::Collections;
	usingnamespace System::Windows::Forms;
	usingnamespace System::Data;
	usingnamespace System::Drawing;
	using System::String;
	usingnamespace System::Threading;
	usingnamespace System::Runtime::InteropServices;
	usingnamespace System::IO;
	usingnamespace Emgu;
	usingnamespace Emgu::CV;
	usingnamespace Emgu::CV::Structure;
	usingnamespace Emgu::CV::CvEnum;
	usingnamespace Emgu::CV::Features2D;
	usingnamespace Emgu::CV::Util;
	usingnamespace Emgu::CV::GPU;
	usingnamespace MySql::Data::MySqlClient;
	/// <summary>
	/// Summary for face
	/// </summary>

	public:

		face(int sno)
		{
			InitializeComponent();
			cam = gcnew Emgu::CV::Capture(1);
			im_width = im_height =100;
			closing=false;
			serial=sno;
			faceno=0;
			total_faces=8;
			CurrentTime = 
gcnewCurrentTimeDelegate(this,&face::UpdateCurrentTime);
			RGB_img = gcnew Emgu::CV::Image<Bgr, Byte>(cam-
>Width,cam->Height);
```

```
                faces = gcnew cli::array<Emgu::CV::Image<Gray,
Byte>^,1>(8);
                //
                //TODO: Add the constructor code here
                //
           }


     void UpdateCurrentTime()
        {
                RGB_img = cam->QueryFrame();
                this->imageBox1->Image=RGB_img;
        }

      Emgu::CV::Image<Gray, Byte> ^ detect_face()
      {
           Emgu::CV::Image<Gray, Byte> ^ gray_img = RGB_img-
>Convert<Gray,Byte>();
           String ^ haar_str="C:\\Emgu\\emgucv-windows-universal-
gpu
2.4.9.1847\\opencv\\data\\haarcascades\\haarcascade_frontalface_d
efault.xml";
           Emgu::CV::HaarCascade ^ haar = gcnew
Emgu::CV::HaarCascade(haar_str);

     cli::array<cli::array<Emgu::CV::Structure::MCvAvgComp>^,1> ^
st1 = gray_img->DetectHaarCascade(haar);
           Rectangle ^ im_rect = st1[0][0].rect;
           gray_img->ROI = *im_rect;
           Emgu::CV::Image<Gray, Byte> ^ face_img = gray_img-
>Resize(im_width,im_height,Emgu::CV::CvEnum::INTER::CV_INTER_CUBI
C);
           return face_img;
      }

     void DisplayCurrentTime()
     {
                while (true)
                {
                     Thread::CurrentThread->Sleep(100);
                     this->Invoke(CurrentTime);
                     if(closing==true)
                          break;
           }
      }
```

```
	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~face()
		{
			if (components)
			{
				delete components;
			}
		}

	private: System::Windows::Forms::Button^  button2;
	private: System::ComponentModel::IContainer^  components;
	protected:

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>


#pragmaregion Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not
modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->components = (gcnew
System::ComponentModel::Container());
			System::ComponentModel::ComponentResourceManager^
resources = (gcnew
System::ComponentModel::ComponentResourceManager(face::typeid));
			this->button2 = (gcnew
System::Windows::Forms::Button());
			this->closebtn = (gcnew
System::Windows::Forms::Button());
			this->imageBox1 = (gcnew
Emgu::CV::UI::ImageBox());
			this->snapbox1 = (gcnew
Emgu::CV::UI::ImageBox());
			this->snapbox2 = (gcnew
Emgu::CV::UI::ImageBox());
```

```
                    this->snapbox4 = (gcnew
Emgu::CV::UI::ImageBox());
                    this->snapbox3 = (gcnew
Emgu::CV::UI::ImageBox());
                    this->snapbtn1 = (gcnew
System::Windows::Forms::Button());
                    this->snapbtn2 = (gcnew
System::Windows::Forms::Button());
                    this->snapbtn4 = (gcnew
System::Windows::Forms::Button());
                    this->snapbtn3 = (gcnew
System::Windows::Forms::Button());
                    this->snapbtn5 = (gcnew
System::Windows::Forms::Button());
                    this->snapbox5 = (gcnew
Emgu::CV::UI::ImageBox());
                    this->delbtn1 = (gcnew
System::Windows::Forms::Button());
                    this->delbtn2 = (gcnew
System::Windows::Forms::Button());
                    this->delbtn3 = (gcnew
System::Windows::Forms::Button());
                    this->delbtn4 = (gcnew
System::Windows::Forms::Button());
                    this->delbtn5 = (gcnew
System::Windows::Forms::Button());
                    this->label1 = (gcnew
System::Windows::Forms::Label());
                    this->shapeContainer1 = (gcnew
Microsoft::VisualBasic::PowerPacks::ShapeContainer());
                    this->ovalShape1 = (gcnew
Microsoft::VisualBasic::PowerPacks::OvalShape());
                    this->snapbox6 = (gcnew
Emgu::CV::UI::ImageBox());
                    this->snapbox7 = (gcnew
Emgu::CV::UI::ImageBox());
                    this->snapbox8 = (gcnew
Emgu::CV::UI::ImageBox());
                    this->snapbtn6 = (gcnew
System::Windows::Forms::Button());
                    this->snapbtn7 = (gcnew
System::Windows::Forms::Button());
                    this->snapbtn8 = (gcnew
System::Windows::Forms::Button());
                    this->delbtn6 = (gcnew
System::Windows::Forms::Button());
```

```
                 this->delbtn7 = (gcnew
System::Windows::Forms::Button());
                 this->delbtn8 = (gcnew
System::Windows::Forms::Button());

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->imageBox1))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox1))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox2))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox4))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox3))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox5))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox6))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox7))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->snapbox8))->BeginInit();
                 this->SuspendLayout();
                 //
                 // button2
                 //
                 this->button2->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                       static_cast<System::Byte>(0)));
                 this->button2->Location =
System::Drawing::Point(665, 602);
                 this->button2->Name = L"button2";
                 this->button2->Size = System::Drawing::Size(127,
36);
                 this->button2->TabIndex = 1;
```

```
                this->button2->Text = L"Train Images";
                this->button2->UseVisualStyleBackColor = true;
                this->button2->Click += gcnew
System::EventHandler(this, &face::button2_Click);
                //
                // closebtn
                //
                this->closebtn->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Bold));
                this->closebtn->Location =
System::Drawing::Point(869, 601);
                this->closebtn->Name = L"closebtn";
                this->closebtn->Size = System::Drawing::Size(127,
39);
                this->closebtn->TabIndex = 2;
                this->closebtn->Text = L"Exit";
                this->closebtn->UseVisualStyleBackColor = true;
                this->closebtn->Click += gcnew
System::EventHandler(this, &face::closebtn_Click);
                //
                // imageBox1
                //
                this->imageBox1->BorderStyle =
System::Windows::Forms::BorderStyle::FixedSingle;
                this->imageBox1->Location =
System::Drawing::Point(665, 278);
                this->imageBox1->Name = L"imageBox1";
                this->imageBox1->Size =
System::Drawing::Size(331, 318);
                this->imageBox1->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                this->imageBox1->TabIndex = 2;
                this->imageBox1->TabStop = false;
                //
                // snapbox1
                //
                this->snapbox1->BackgroundImage =
(cli::safe_cast<System::Drawing::Image^  >(resources-
>GetObject(L"snapbox1.BackgroundImage")));
                this->snapbox1->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Stretch;
                this->snapbox1->BorderStyle =
System::Windows::Forms::BorderStyle::FixedSingle;
                this->snapbox1->Location =
System::Drawing::Point(15, 278);
```

```
                    this->snapbox1->Name = L"snapbox1";
                    this->snapbox1->Size = System::Drawing::Size(135,
135);
                    this->snapbox1->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                    this->snapbox1->TabIndex = 2;
                    this->snapbox1->TabStop = false;
                    //
                    // snapbox2
                    //
                    this->snapbox2->BackgroundImage =
(cli::safe_cast<System::Drawing::Image^  >(resources-
>GetObject(L"snapbox2.BackgroundImage")));
                    this->snapbox2->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Stretch;
                    this->snapbox2->BorderStyle =
System::Windows::Forms::BorderStyle::FixedSingle;
                    this->snapbox2->Location =
System::Drawing::Point(179, 278);
                    this->snapbox2->Name = L"snapbox2";
                    this->snapbox2->Size = System::Drawing::Size(135,
135);
                    this->snapbox2->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                    this->snapbox2->TabIndex = 3;
                    this->snapbox2->TabStop = false;
                    //
                    // snapbox4
                    //
                    this->snapbox4->BackgroundImage =
(cli::safe_cast<System::Drawing::Image^  >(resources-
>GetObject(L"snapbox4.BackgroundImage")));
                    this->snapbox4->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Stretch;
                    this->snapbox4->BorderStyle =
System::Windows::Forms::BorderStyle::FixedSingle;
                    this->snapbox4->Location =
System::Drawing::Point(509, 278);
                    this->snapbox4->Name = L"snapbox4";
                    this->snapbox4->Size = System::Drawing::Size(135,
135);
                    this->snapbox4->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                    this->snapbox4->TabIndex = 5;
                    this->snapbox4->TabStop = false;
                    //
```

```
                // snapbox3
                //
                this->snapbox3->BackgroundImage =
(cli::safe_cast<System::Drawing::Image^  >(resources-
>GetObject(L"snapbox3.BackgroundImage")));
                this->snapbox3->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Stretch;
                this->snapbox3->BorderStyle =
System::Windows::Forms::BorderStyle::FixedSingle;
                this->snapbox3->Location =
System::Drawing::Point(342, 278);
                this->snapbox3->Name = L"snapbox3";
                this->snapbox3->Size = System::Drawing::Size(135,
135);
                this->snapbox3->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                this->snapbox3->TabIndex = 4;
                this->snapbox3->TabStop = false;
                //
                // snapbtn1
                //
                this->snapbtn1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(0)));
                this->snapbtn1->Location =
System::Drawing::Point(15, 419);
                this->snapbtn1->Name = L"snapbtn1";
                this->snapbtn1->Size = System::Drawing::Size(82,
23);
                this->snapbtn1->TabIndex = 6;
                this->snapbtn1->Text = L"Take Snap 1";
                this->snapbtn1->UseVisualStyleBackColor = true;
                this->snapbtn1->Click += gcnew
System::EventHandler(this, &face::snapbtn1_Click);

#pragmaendregion
/*
     private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
    }

*/
     private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e)
```

```
      {
            if ( (!faces[0]) | (!faces[1]) | (!faces[2]) |
(!faces[3]) | (!faces[4])| (!faces[5])| (!faces[6])| (!faces[7])
)
            {
                  MessageBox::Show("Insufficient Training Images");
                  return;
            }

            // Save Current images to db
            String ^ img_path = String::Concat("D:/Final
Code/all_auto/proj_v1/proj_v1/images/",serial);
            IO::Directory::CreateDirectory(img_path);
            for (int i=0;i<this->total_faces;i++)
            {
                  String
^paths=String::Concat(img_path,"/",i,".jpg");
                  faces[i]->Save(paths);
            }
      //    Emgu::CV::FisherFaceRecognizer ^ fish_rec = gcnew
CV::FisherFaceRecognizer(0,123);
      //    cli::array<int,1> ^ labels = gcnew array <int,1>
(1000);
      //    array<CV::Image<Gray,Byte>^,1> ^imgs = gcnew
array<CV::Image<Gray,Byte>^,1>(1000);
            String^ ab= "server=localhost;Database=db_mysql;User
ID=root;Password=''";
            MySqlConnection^ conSQL = gcnewMySqlConnection(ab);
            conSQL->Open();
            MySqlCommand ^command = conSQL->CreateCommand();

            command->CommandText = "UPDATE  `db_mysql`.`t1` SET
`Path` =  '"+img_path+"' WHERE  `t1`.`Sno` ='"+this->serial+"';";
            command->ExecuteNonQuery();

            MessageBox::Show("Training is Completed, Thank You for
Registering in INTEGRATED SECURITY SYSTEM'S Data Base");
 }


private: System::Void face_Load(System::Object^  sender,
System::EventArgs^  e) {
                  currentTimeThread = gcnew Thread(gcnew
ThreadStart(this,&face::DisplayCurrentTime));
//starts the thread
                  Thread::CurrentThread->Sleep(100);
```

```
                    currentTimeThread->Start();
               }
private: System::Void closebtn_Click(System::Object^  sender,
System::EventArgs^  e) {
                    closing=true;
                    this->currentTimeThread->Abort();
                    this->currentTimeThread->Join();
                    this->Close();
               }
private: System::Void snapbtn1_Click(System::Object^  sender,
System::EventArgs^  e)
               {
                    faces[0] = gcnew Emgu::CV::Image
<Gray,Byte>(im_width,im_height);
                    faces[0] = this->detect_face();
                    this->snapbox1->Image = faces[0];
               }
private: System::Void snapbtn2_Click(System::Object^  sender,
System::EventArgs^  e)
               {
                    faces[1] = this->detect_face();
                    this->snapbox2->Image = faces[1];
               }
private: System::Void snapbtn3_Click(System::Object^  sender,
System::EventArgs^  e)
               {
                    faces[2] = this->detect_face();
                    this->snapbox3->Image = faces[2];
               }
private: System::Void snapbtn4_Click(System::Object^  sender,
System::EventArgs^  e)
               {
                    faces[3] = this->detect_face();
                    this->snapbox4->Image = faces[3];
               }
private: System::Void snapbtn5_Click(System::Object^  sender,
System::EventArgs^  e)
               {
                    faces[4] = this->detect_face();
                    this->snapbox5->Image = faces[4];
               }
private: System::Void delbtn1_Click(System::Object^  sender,
System::EventArgs^  e)
               {
                    faces[0]= nullptr;
                    this->snapbox1->Image = faces[0];
```

```
                }
    };
    }
```

APPENDIX A-3

Face Recognition Code

```
#pragmaonce

usingnamespace System;
usingnamespace System::ComponentModel;
usingnamespace System::Collections;
usingnamespace System::Windows::Forms;
usingnamespace System::Data;
usingnamespace System::Drawing;
	using System::String;
	usingnamespace System::Threading;
	usingnamespace System::Runtime::InteropServices;
	usingnamespace System::IO;
	usingnamespace Emgu;
	usingnamespace Emgu::CV;
	usingnamespace Emgu::CV::Structure;
	usingnamespace Emgu::CV::CvEnum;
	usingnamespace Emgu::CV::Features2D;
	usingnamespace Emgu::CV::Util;
	usingnamespace Emgu::CV::GPU;
	usingnamespace MySql::Data::MySqlClient;
	usingnamespace System::IO::Ports;

namespace proj_v1 {

	/// </summary>
	publicrefclass Face_Recognize : public
System::Windows::Forms::Form
	{
	public:
		//findPorts();
		int serial, total_faces;
		bool closing;
		delegatevoid CurrentTimeDelegate();
		Delegate^ CurrentTime;
		Thread ^currentTimeThread;
		Emgu::CV::Capture ^cam;
		Emgu::CV::Image<Bgr, Byte> ^RGB_img;
		int im_width, im_height;

	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label3;
	private: System::Windows::Forms::Label^  norank;
	private: System::Windows::Forms::Label^  depco;




	private: Microsoft::VisualBasic::PowerPacks::ShapeContainer^
shapeContainer1;
	private: Microsoft::VisualBasic::PowerPacks::RectangleShape^
rectangleShape1;


	private: System::IO::Ports::SerialPort^  serialPort1;
	private: System::Windows::Forms::Timer^  timer1;
	private: System::Windows::Forms::Label^  label7;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::Label^  p_no;
	private: System::Windows::Forms::Label^  p_name;
	private: System::Windows::Forms::Label^  nicno;

	public:
	private: System::Windows::Forms::Button^  closebtn;
	public:

		Face_Recognize(int sno)
		{
			InitializeComponent();
//			findPorts();
			cam = gcnew Emgu::CV::Capture(1);

			im_width = im_height =100;
			closing=false;
			serial=sno;
			total_faces = 8;
			CurrentTime =
gcnewCurrentTimeDelegate(this,&Face_Recognize::UpdateCurrentTime)
;
			RGB_img = gcnew Emgu::CV::Image<Bgr, Byte>(cam-
>Width,cam->Height);
			//
			//TODO: Add the constructor code here
			//
		}

void UpdateCurrentTime()
	{
```

```
                    RGB_img = cam->QueryFrame();
                    this->imageBox1->Image=RGB_img;
            }

        Emgu::CV::Image<Gray, Byte> ^ detect_face()
        {
            Emgu::CV::Image<Gray, Byte> ^ gray_img = RGB_img-
>Convert<Gray,Byte>();
            String ^ haar_str="C:\\Emgu\\emgucv-windows-universal-
gpu
2.4.9.1847\\opencv\\data\\haarcascades\\haarcascade_frontalface_d
efault.xml";
            Emgu::CV::HaarCascade ^ haar = gcnew
Emgu::CV::HaarCascade(haar_str);

        cli::array<cli::array<Emgu::CV::Structure::MCvAvgComp>^,1> ^
st1 = gray_img->DetectHaarCascade(haar);
            if (st1[0]->Length<1)
                returngcnew Emgu::CV::Image<Gray, Byte>(100,100);
            Rectangle ^ im_rect = st1[0][0].rect;
            gray_img->ROI = *im_rect;
            Emgu::CV::Image<Gray, Byte> ^ face_img = gray_img-
>Resize(im_width,im_height,Emgu::CV::CvEnum::INTER::CV_INTER_CUBI
C);
            return face_img;
        }

        void DisplayCurrentTime()
        {
                while (true)
                {
                    Thread::CurrentThread->Sleep(100);
                    this->Invoke(CurrentTime);
                    if(closing==true)
                        break;
              }
        }
        protected:
            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            ~Face_Recognize()
            {
                if (components)
                {
                    delete components;
        }
        delete cam;
}
```

APPENDIX A-4

New Entry Registration Code

```
#pragmaonce
#include"face.h"

namespace proj_v1 {

	usingnamespace System;
	usingnamespace System::ComponentModel;
	usingnamespace System::Collections;
	usingnamespace System::Windows::Forms;
	usingnamespace System::Data;
	usingnamespace System::Drawing;
	using System::String;
	using System::Byte;
	//Thread ^currentTimeThread;
	//#using <MasterRD.dll>
	usingnamespace System::Runtime::InteropServices;
	usingnamespace MySql::Data::MySqlClient;

	/// <summary>
	/// Summary for card_register
	/// </summary>
	publicrefclass card_register : public
System::Windows::Forms::Form
	{
			//[DllImport("E:\\sohaib\\vc proj\\RF card
reader\\RF_card\\MasterCom.dll")];
	[DllImport("kernel32.dll")] staticvoid Sleep(int num);
	[DllImport("MasterRD.dll")] staticint rf_init_com(int port,
long baud);
	[DllImport("MasterRD.dll")] staticint rf_ClosePort();
	[DllImport("MasterRD.dll")] staticint
rf_M1_read(unsignedshort icdev, unsignedchar Block, IntPtr pData,
unsignedchar* pLen);
	[DllImport("MasterRD.dll")] staticint rf_antenna_sta(short
icdev, unsignedchar mode);
	[DllImport("MasterRD.dll")] staticint rf_init_type(short
icdev, unsignedchar type);
	[DllImport("MasterRD.dll")] staticint rf_request(short
icdev, unsignedchar mode, unsignedshort* pTagType);
	[DllImport("MasterRD.dll")] staticint rf_anticoll(short
icdev, unsignedchar bcnt, IntPtr pSnr, unsignedchar* pRLength);
```

```
      [DllImport("MasterRD.dll")] staticint rf_select(short icdev,
IntPtr pSnr, unsignedchar srcLen, unsignedchar* Size);
      [DllImport("MasterRD.dll")] staticint
rf_M1_authentication2(short icdev, unsignedchar mode,
unsignedchar Block, IntPtr pLen);
      [DllImport("MasterRD.dll")] staticint rf_M1_write(short
icdev, Byte Block, IntPtr pLen);
      private: System::Windows::Forms::Button^  closebtn;

      public:
      //    bool closing;

            card_register(void)
            {
                  InitializeComponent();
                  //
                  //TODO: Add the constructor code here
                  //
            }

                        String^ ToHexString(Byte bytes[])
                  {
                        staticchar hexDigits[] = { '0', '1', '2',
'3', '4', '5', '6', '7',
                        '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
                        char* chars = newchar[15*2];
                        for (int i = 0; i < 15; i++)
                        {
                              int b = bytes[i];
                              chars[i * 2] = hexDigits[b >> 4];
                              chars[i*2+1] = hexDigits[b & 0xF];
                        }
                        returngcnew String(chars);
                  }



            void initializereader()
            {

                  short icdev = 0x00;
                  int status;

                  status = rf_init_com(2, 14400);
                  unsignedchar type = Convert::ToByte('A');//
      'mifare one
```

```
                unsignedchar mode = 0x52;
                unsignedshort*  TagType = newunsignedshort;
                unsignedchar bcnt = 0x04;
                // 'mifare
                IntPtr pSnr;
                unsignedchar* len = newunsignedchar;
                unsignedchar* size = newunsignedchar;

           /*    If Not bConnectedDevice Then
MessageBox.Show("Not connect to device!!", "error",
MessageBoxButtons.OK, MessageBoxIcon.[Error])
            Return
        End If
*/
                pSnr = Marshal::AllocHGlobal(1024);
                int i;
                for (i=0;i<=1;i++)
                {
                     status = rf_antenna_sta(icdev, 0);
                     if (status != 0)
                          continue;
                     Sleep(20);
                     status = rf_init_type(icdev, type);
                     if (status != 0)
                          continue;
                     Sleep(20);
                     status = rf_antenna_sta(icdev, 1);
                     if (status != 0)
                          continue;
                     Sleep(50);
                     status = rf_request(icdev, mode, TagType);
                     if (status != 0)
                          continue;
                     status = rf_anticoll(icdev, bcnt, pSnr,
len);
                     if (status != 0)
                          continue;
                     status = rf_select(icdev, pSnr, *len,
size);
                     if (status != 0)
                          continue;
                     unsignedint* szBytes =
newunsignedint[*len+1];
                     String^ str =
Marshal::PtrToStringAnsi(pSnr);
                     int size_str = str->Length;
```

```
		}
		Marshal::FreeHGlobal(pSnr);
	}


	Byte GetHexBitsValue(Byte ch)
	{
		Byte sz = 0;
		if (ch <= '9'&& ch >= '0')
			sz = (Byte)(ch - 0x30);
		if (ch <= 'F'&& ch >= 'A')
			sz = (Byte)(ch - 0x37);
		if (ch <= 'f'&& ch >= 'a')
			sz = (Byte)(ch - 0x57);
		return sz;
	}



	unsignedchar* ToDigitsBytes(unsignedchar* theHex)
	{
		//int leng = (sizeof(theHex) / 2) + 
(((sizeof(theHex) % 2) > 0) ? 1 : 0);
		//unsigned char temp[leng];
		int leng = 16;
		unsignedchar* bytes = newunsignedchar[];
		for (int i = 0; i < leng; i++)
		{
			char lowbits = theHex[i*2];
			char highbits;
			if ((i * 2 + 1) < 32)
				highbits = theHex[i * 2 + 1];
			else
				highbits = '0';
			int a = 
(int)GetHexBitsValue((Byte)lowbits);
			int b = 
(int)GetHexBitsValue((Byte)highbits);
			*(bytes+i) = (Byte)((a << 4) + b);
		}
		//unsigned char* bytes = &temp;
		return bytes;
	}
```

```
protected:
	/// <summary>
	/// Clean up any resources being used.
	/// </summary>
	~card_register()
	{
		//rf_ClosePort();
		if (components)
		{
			delete components;
		}
	}
```

APPENDIX A-5

Microcontroller Code

```
#include<stdio.h>
#include<regx52.h>
bit i,j;
sbit LOCK=P2^0;                   //Relay for Solinoid LOCk  s connected with P1^0 pin.
char t;
void serl() interrupt 4
  {if (RI)
        {
                         t=SBUF;
if (t=='1')
                        {
                          i=1;
                       printf("One REcvd ");
                        }
RI=0;
        }
 }
void delay(unsigned int t)
{
unsigned int x1,y1;
for(x1=0;x1<t;x1++)
for(y1=0;y1<498;y1++);
}
void baud_rate_init()
{
        TMOD=0x20;// Use Timer 1 8-bit Auto Reload mode
        TH1=0xFD; //Baud Rate 9600
        SCON=0X50;// 8 bit,1 stop bit
        TR1=1;   // Start Timer 1 for Serial communication
        TI=1;
}
void main()
{
EA=1;                  // Enable All Interupts
ES=1;                                  // Enable Serial Interupt
LOCK=1;
i,j=0;
baud_rate_init();
printf(" Welcome ");
while(1)
{if(i==1)
{  LOCK=0;
delay(25);
LOCK=1;
i=0;
                        }
        }
}
```