

DESIGN AND DEVELOPMENT OF WIRELESS FINGERPRINT RECOGNITION BASED ATTENDANCE SYSTEM



By

NC Adil Khan

Capt Atif Naveed

PC Masood Ahmed

Submitted to the Department of Electrical Engineering, Military College of Signals
National University of Sciences and Technology, Rawalpindi in partial fulfillment for
the requirements of a B.E Degree in Telecomm Engineering

ABSTRACT

The aim of project is to introducing the use of biometric base technologies in automating the attendance system for the students in different institutions. The goal of the project can be subdivided into smaller targets like fingerprint capture, fingerprint image processing, template generation and wireless transfer of data in a client server base environment. For each sub-task, various methods from literature are analyzed. From the study of the entire process, an integrated approach is proposed.

Use of biometrics based technologies is efficient personal identifiers as they make use of characteristics that are unique in each person. Among these technologies, Fingerprint recognition is universally applied. It extracts minutia- based features from images scanned by fingerprint scanners made by the different ridges and valleys on the finger-tips. The student attendance system can be very useful in an institute like Military College of Signals since it aims at eliminating all the drawbacks of attendance system.

Dedicated to

*Almighty Allah,
Faculty for their help
And Our Parents for their support and prayers*

ACKNOWLEDGMENTS

We express our profound gratitude and indebtedness to Lec Ayesha Naureen Department of Information Security, Military College of Signals for introducing the present topic and for their inspiring intellectual guidance, constructive criticism and valuable suggestions throughout the project work.

We are also thankful to all faculty members of Electrical Engineering Department, Military College of Signals specially Dr Adil Masood and Miss Aimon Aakif Department of Electrical Engineering for motivating us in improving the algorithms.

Finally we would like to thank our parents for their support and prayers for us to complete this project.

Capt Atif Naveed

NC Adil Khan

PC Masood Ahmed

Table of Contents

Chapter 1	1
Introduction.....	1
1.1 Problem Statement.....	2
1.2 Motivation and Challenges	2
1.3 Using Biometrics	2
Chapter 2	4
Literature Review.....	4
2.1 Fingerprint Module	4
2.1.1 What is Fingerprint?	4
2.1.2 Why use Fingerprints?	5
2.1.3 Fingerprint Recognition	5
2.1.4 Approach to Fingerprint Recognition	6
2.1.5 Using fingerprint Recognition System for Attendance Management	7
2.2 Wireless Transmission Module.....	7
2.2.1 Original System Structure.....	7
2.2.2 Transmitter.....	7
2.2.3 Receiver	8
2.2.4 Problems with Original Design.....	9
2.2.5 Proposed Design	9
Chapter 3	10
Fingerprint Image Processing	10
3.1 Pre-Processing.....	10
3.1.1 Image Enhancement.....	10
3.1.1.1 Histogram Equalization	10
3.1.2 Image Binarization	11
3.1.3 Image Segmentation.....	12
3.1.3.1 Block Direction Estimation.....	12
3.1.3.2 ROI Extraction by Morphological Methods	13
3.2 Minutia Extraction	14
3.2.1 Ridge Thinning	14
3.2.2 Minutia Marking	15
3.3 Post-Processing	15
3.3.1 False Minutia Removal	15
3.3.2 Unification of Terminations and Bifurcations	16
Chapter 4	17

System Design	17
4.1 Overall System Design Block Diagram.....	17
4.2 Module Design.....	18
4.2.1 Fingerprint Capture Module (FCM)	18
4.2.2 Client Application Module	19
4.2.2 Wireless Communication Module (WCM).....	20
4.2.4 Server Application Module.....	20
4.2.5 GSM Module	21
4.3 Algorithm Design.....	22
Chapter 5	23
Data Transfer and Application Development	23
5.1 Wireless Data Transfer	23
5.1.1 Enroll Data.....	23
5.1.2 Daily Attendance Data.....	23
5.2 Application Development	23
5.2.1 Client PC.....	24
5.2.2 Server PC.....	26
5.2.3 GSM Module	27
5.2.4 Database.....	28
Chapter 6	30
Experimental Setup and Testing	30
6.1 Physical Layout.....	30
6.2 Fingerprint Template Generation.....	30
6.2.1 Fingerprint Enrollment.....	30
6.2.2 Fingerprint Verification	31
6.2.3 Fingerprint Template	32
6.3 Client-Server Communication	33
6.4 Attendance management at Server.....	34
6.4.1 Enrollment in Database.....	34
6.4.2 Verification SMS using GSM Module.....	35
Chapter 7	36
7.1 Conclusion	36
7.2 Outcomes of this Project.....	36
7.3 Future Work.....	37
Appendix	38
Fingerprint Scanner Code	38

Client- Server Communication	45
Client Code	45
Server Code.....	46
GSM Module Code.....	49
Database Code	58
References	67

S.No	Figure No	Title	Page No
1	Figure 2.1	Ridge ending and Bifurcation	4
2	Figure 2.2	Verification Vs Identification	6
3	Figure 2.3	Transmitter Block Diagram	8
4	Figure 2.4	Receiver Block Diagram	8
5	Figure 2.5	Proposed System Block Diagram	9
6	Figure 3.1	Fingerprint with Original Histogram	11
7	Figure 3.2	After Histogram Equalization	11
8	Figure 3.3	Original Image	11
9	Figure 3.4	Enhanced Image	11
10	Figure 3.5	Effect of Binarization	11
11	Figure 3.6	Effect of Block Direction Estimation	13
12	Figure 3.7	Before CLOSE Operation	13
13	Figure 3.8	After CLOSE Operation	13
14	Figure 3.9	After operation OPEN	14
15	Figure 3.10	ROI + Bound	14
16	Figure 3.11	False Minutia Structures	15
17	Figure 4.1	Various Modules in the System Design	17
18	Figure 4.2	A Fingerprint Sensor	18
19	Figure 4.3	Client Application	19
20	Figure 4.4	Wireless Module	20
21	Figure 4.5	Server Application	21
22	Figure 5.1	Fingerprint Acquisition Application	24
23	Figure 5.2	Adil Fingerprint	25
24	Figure 5.3	Masood Fingerprint	25
25	Figure 5.4	Client Application	26
26	Figure 5.5	Server Application	27
27	Figure 5.6	GSM Application	28
28	Figure 5.7	Database Application	29
29	Figure 6.1	Physical Layout	30
30	Figure 6.2	Fingerprint Enrollment	31
31	Figure 6.3	Successful Fingerprint Enrollment	31
32	Figure 6.4	Successful Verification of Fingerprint	32
33	Figure 6.4	Unsuccessful Verification of Fingerprint	32
34	Figure 6.5	Template Generation	33
35	Figure 6.6	Client Application	33
36	Figure 6.7	Successful Transmission by Client	34
37	Figure 6.7	Successful Reception by Server	34

Chapter 1

Introduction

The human body has been blessed with the great privilege of having features that are exclusive and unique to each individual. These unique characteristics have led to the use of biometrics and its applications in providing better security in various fields. In few past years biometrics has gained a lot of popularity and proved itself to be a reliable mode of providing privacy, maintaining security and identifying individuals. It is widely accepted throughout the world and now a day is being used at places like private organizations, airports, hospitals, schools, colleges, corporate offices etc.

Biometrics is the study of identifying a person by his/her unique physical traits that are inherent from family and unique to only the person under study. Biometric measurement and assessment include fingerprint verification, iris recognition, palm geometry, face recognition etc. The techniques being used in biometrics can work with different levels of functionality and accuracy.

Accuracy and reliability are the two most important parameters when it comes to biometric applications. Fingerprint verification is one of the oldest known biometric techniques known but still is the most widely used because of its simplicity and good levels of accuracy. It's a well-known fact that every human being is born with a different pattern on the fingers and this feature is exploited to identify and differentiate between two different persons.

The application in an educational institute is worth noting because of the benefits it brings along with it. The fingerprint recognition and verification technique can easily replace an attendance sheet and save time wasted on calling out roll numbers in the class. A fingerprint detecting device needs to be placed in each classroom and students would be made to swipe their finger over the sensor so as to mark their presence in the class. The database would contain all the fingerprints beforehand. So, the moment a finger would be swiped, a check would be carried out.

1.1 Problem Statement

Designing a student attendance management system based on fingerprint recognition and faster way of identification that manages records for attendance in institutes and other organizations.

1.2 Motivation and Challenges

Every organization whether it be an educational institution or business organization, it has to maintain a proper record of attendance of students or employees for effective functioning of organization. Designing a better attendance management system for students so that records are maintained with ease and accuracy was an important key behind motivating this project. This would improve accuracy of attendance records because it will remove all the hassles of roll calling and will save valuable time of the students as well as teachers. Image processing and fingerprint recognition are very advanced today in terms of technology. It was our responsibility to improve fingerprint identification system.

1.3 Using Biometrics

The human body has the privilege of having features that are unique and exclusive to each individual. This exclusivity and unique characteristic has led to the field of biometrics and its application in ensuring security in various fields. Biometrics has gained popularity and has proved itself to be a reliable mode of ensuring privacy, maintaining security and identifying individuals. It has wide acceptance throughout the globe and now is being used at places like airports, hospitals, schools, colleges, corporate offices etc.

Biometrics is the very study of identifying a person by his/her physical traits that are inherent and unique to only the person concerned. Biometric measurement and assessment include fingerprint verification, iris recognition, palm geometry, face recognition etc. The above mentioned techniques work with different levels of functionality and accuracy.

Accuracy and reliability are the two most important parameters when it comes to biometric applications. Fingerprint verification is one of the oldest known biometric techniques known but still is the most widely used because of its simplicity and good

levels of accuracy. It's a well-known fact that every human being is born with a different pattern on the fingers and this feature is exploited to identify and differentiate between two different persons.

The application in an educational institute is worth noting because of the benefits it brings along with it. The fingerprint recognition and verification technique can easily replace an attendance sheet and save time wasted on calling out roll numbers in the class. A fingerprint detecting device needs to be placed in each classroom and students would be made to swipe their finger over the sensor so as to mark their presence in the class. The database would contain all the fingerprints beforehand. So, the moment a finger would be swiped, a check would be carried out with the existing database and the corresponding student would get a present mark on his attendance record maintained in a server.

The transfer of the fingerprint from the device to the server can be carried out wirelessly using certain wireless adapters which can together form a wireless network in a short range and carry out the verification process. The communication channel needs to be secured and should be kept free from interference as far as possible. For further security of the entire system and to detect illegal activities, a security camera can be installed to keep track of the enrollments made in the classroom. Biometric Identification Systems are widely used for unique identification of humans mainly for verification and identification. Biometrics is used as a form of identity access management and access control. So use of biometrics in student attendance management system is a secure approach. There are many types of biometric systems like fingerprint recognition, face recognition, voice recognition, iris recognition, palm recognition etc. In this project we used fingerprint recognition system.

Chapter 2

Literature Review

2.1 Fingerprint Module

In this section we will focus on the characteristics and benefits of using the fingerprints for our purpose of making an attendance system which can be deployed in institutions and organization for maintaining the record of attendance.

2.2.1 What is Fingerprint?

A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. The endpoints and crossing points of ridges are called minutiae. It is a widely accepted assumption that the minutiae pattern of each finger is unique and does not change during one's life. Ridge endings are the points where the ridge curve terminates and bifurcations are where a ridge splits from a single path to two paths at a Y-junction. Figure 2.1 illustrates an example of a ridge ending and a bifurcation. In this example the black pixels correspond to the ridges, and the white pixels correspond to the valleys.

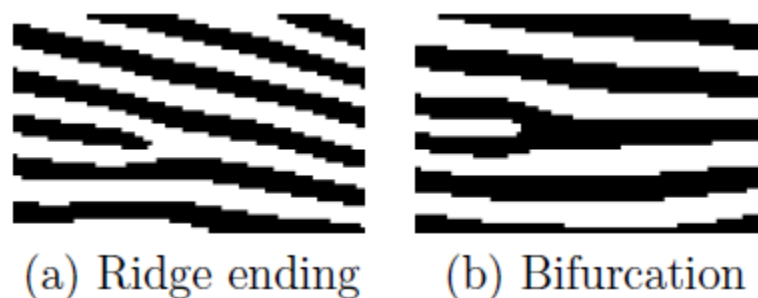


Figure 2.1: Ridge ending and Bifurcation

When human fingerprint experts determine if two fingerprints are from the same finger, the matching degree between two minutiae pattern is one of the most important factors. Thanks to the similarity to the way of human fingerprint experts and compactness of templates, the minutiae-based matching method is the most widely studied matching method.

2.1.2 Why use Fingerprints?

Fingerprints are considered to be the best and fastest method for biometric identification. They are secure to use, unique for every person and do not change in one's lifetime. Besides these, implementation of fingerprint recognition system is cheap, easy and accurate up to satisfactory level.

Fingerprint recognition has been widely used in both forensic and civilian applications. Compared with other biometrics features, fingerprint-based biometrics is the most proven technique and has the largest market shares. Not only it is faster than other techniques but also the energy consumption by such systems is too less.

2.1.3 Fingerprint Recognition

Once the fingerprint is captured, the next step is the recognition procedure. The recognition procedure can be divided into

- a. Fingerprint identification
- b. Fingerprint verification

Fingerprint identification means user's identity is based on his fingerprints. The fingerprint image is stored without any information about the identity of the person. After that it is matched across a number of fingerprints in the database. The identity of the user is only retrieved when a match is found in the database. So, this is a case of one-to-n matching where one capture is compared to several others. This is widely used for criminal cases.

Fingerprint verification is different from identification as the person's identity is stored along with the fingerprint in a database. On enrolling the fingerprint, the captured image will retrieve back the identity of the person. This is however a one-to-one matching. This is used in offices like confirmation office, passport offices etc. where the identity of a person has to be checked with the one provided at a previous stage.

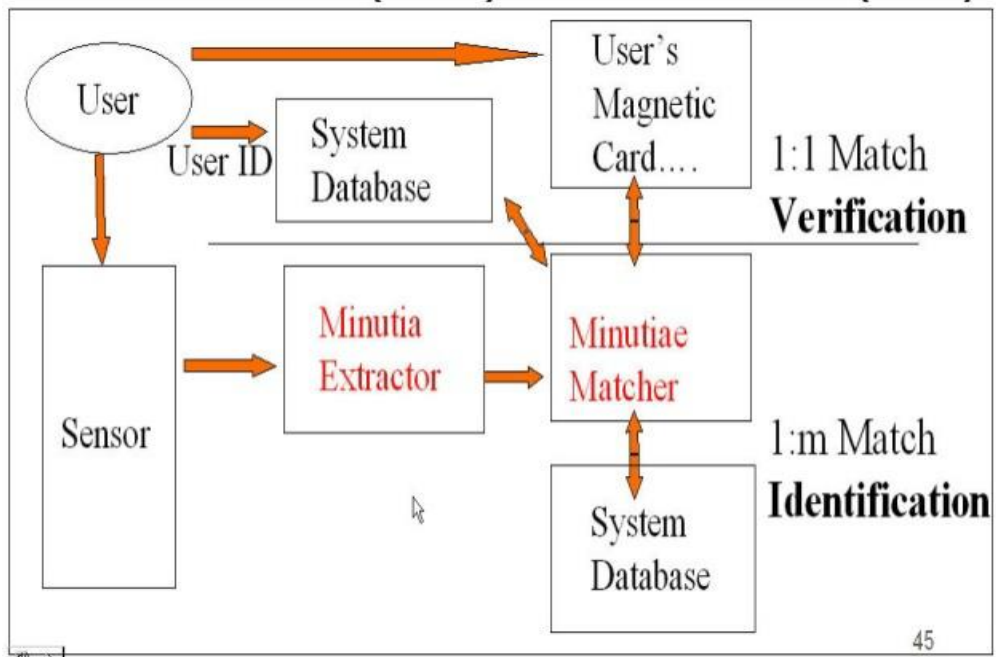


Figure 2.2: Verification Vs Identification

In the following pages, an approach to fingerprint recognition has been discussed.

2.1.4 Approach to Fingerprint Recognition

The approach that we have concentrated on in recognition of the fingerprints is the minutia based approach. In this approach the ridge bifurcations and terminations are taken into consideration for analyzing each fingerprint. The representation is based on these local features.

The scanner system uses highly complex algorithms to recognize and analyze the minutia. The basic idea is to measure the relative portion of minutia. Simply, it can be thought of as considering the various shapes formed by the minutia when straight lines are drawn between them or when the entire image is divided into matrix of square sized cells. If two fingerprints have the same set of ridge endings and bifurcations forming the same shape with the same dimension, there's a huge likelihood that they are of the same fingerprint.

So, to find a match the scanner system has to find a sufficient number of minutia patterns that the two prints have in common, the exact number being decided by the scanner programming.

2.1.5 Using fingerprint Recognition System for Attendance Management

Managing attendance records of students of an institute is a tedious task. It consumes time and paper both. To make all the attendance related work automatic and on-line, we have designed an attendance management system which could be implemented in any institution or organization. It uses a fingerprint identification system developed in this project.

This fingerprint identification system uses existing as well as new techniques in fingerprint recognition and matching. A new one to many matching algorithm for large databases has been introduced in this identification system.

2.2 Wireless Transmission Module

It is the Wi-Fi module which used for the transmission of data between Client and Server. The client and server should both at the same network.

2.2.1 Original System Structure

The system hardware includes Fingerprint sensor, PIC Microcontroller, ZigBee module, Wireless transmission GSM module.

- a. Fingerprint acquisition module is used to realize fingerprint collecting and pre-processing.
- b. ZigBee module is used to send the finger print image to computer.
- c. Attendance management workstation is used to realize fingerprint extraction and matching in order to realize attendance function.
- d. GSM module connecting to GSM network and it is use for enquire purpose.

2.2.2 Transmitter

This block consists of a finger print module that captures the data which is a person's fingerprint. The sensor forms the core part of the fingerprint module. This in turn is connected to a PIC16F877A microcontroller using RS232. The microcontroller stores

the captured data and this will send through the ZigBee transmitter module for further processing of data.

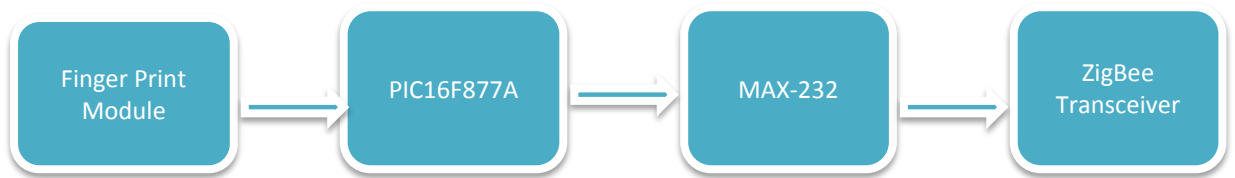


Figure 2.3: Transmitter Block Diagram

2.2.3 Receiver

The captured data is received by the ZigBee receiver module and is forwarded to PIC16F877A microcontroller. The database of the person's that has been stored in microcontroller is compared with the receive data. The microcontroller then sends the data to PC through MAX232. The data is sent to PC for a specified time interval. The PC thus sends the information to the GSM further it is sent to mobile through message on enquire.

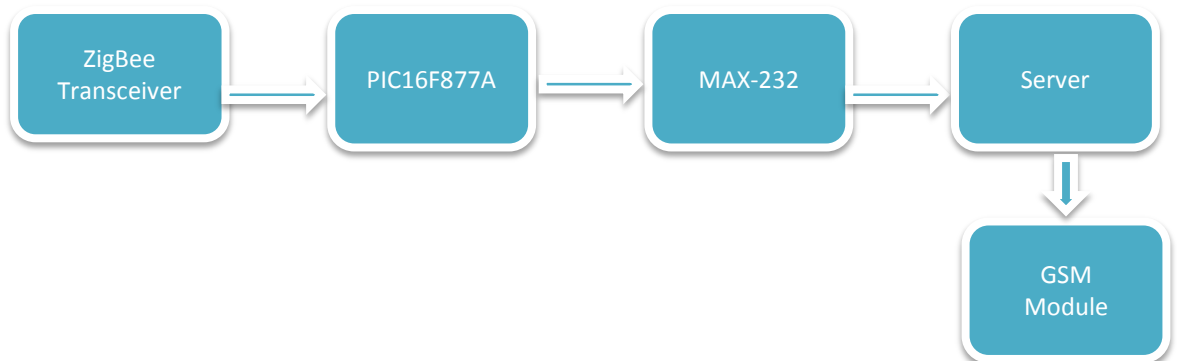


Figure 2.4: Receiver Block Diagram

2.2.4 Problems with Original Design

- a. We would have to transmit each finger print image to data base i.e. more transmissions were required with this design.
- b. Image compression technique was to be used to compress the image to reasonable size so that it can be transmitted through ZigBee.
- c. More interfacing work load than the original task to accomplish.
- d. Expensive design
- e. Time inefficient.

2.2.5 Proposed Design

After study of several research papers we came to the conclusion that we should use the existing WLAN which is already deployed in the institution to make the system cost efficient and easy to integrate with the existing environment.

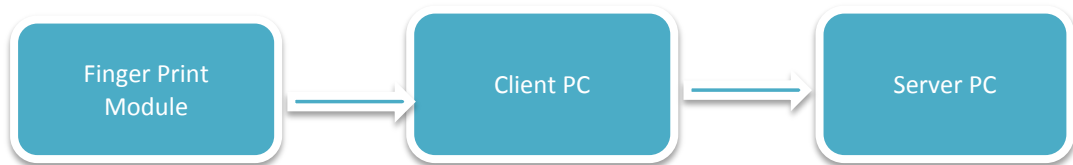


Figure 2.5: Proposed System Block Diagram

The complete system will be discussed in detail in the coming chapters.

Chapter 3

Fingerprint Image Processing

Finger print image processing can be divide into three steps .Pre-Processing, Minutia extraction, Post-Processing. These three steps are discussed here in detail.

3.1 Pre-Processing

The pre-processing can be sub divided into image enhancement, image binerization and image segmentation.

3.1.1 Image Enhancement

Image enhancement is important to clarify the image for further procedure. The fingerprint images obtained from sensor is not very good quality. Hence, enhancement methods are used for making the contrast between ridges and furrows higher and it is also important for maintaining continuity among the false broken points of ridges, which make ensure a higher accuracy for recognition of fingerprint.

Generally two types of procedures are adopted for image enhancement:

- i. Histogram Equalization
- ii. Fourier Transform.

3.1.1.1 Histogram Equalization

Histogram equalization is responsible for expanding the pixel distribution of an image in order to increase perceptual improvement. The pictorial description is given below. The fingerprint initially has a bimodal type histogram as shown in Figure 3.1. After histogram equalization is carried out, the image occupies the entire range from zero to 255, enhancing the visualization effect in the process.

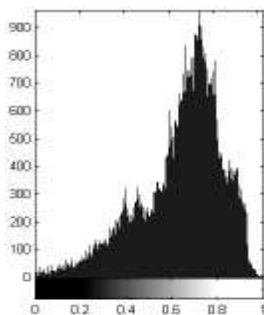


Figure 3.1: Fingerprint with Original Histogram

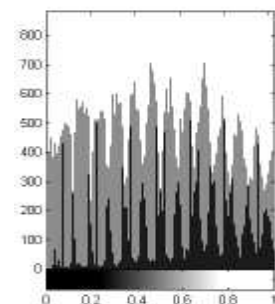


Figure 3.2: After Histogram Equalization

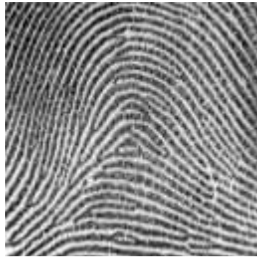


Figure 3.3: Original Image

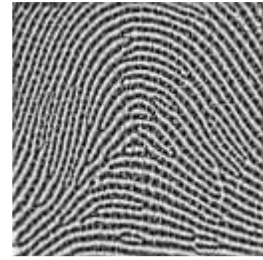


Figure 3.4: Enhanced Image

3.1.2 Image Binarization

The original image is a 8-bit gray scale image. This process transforms the original image into a 1-bit image that assigns values 0 for ridges and 1 for furrows. After binarization, the ridges appear black while the furrows appear white. Binarization changes the pixel value to 1 if the value is found to exceed the mean intensity of the current block to which it belongs.

The figure clearly depicts the effect of binarization on a normal gray scale image that has been only enhanced.

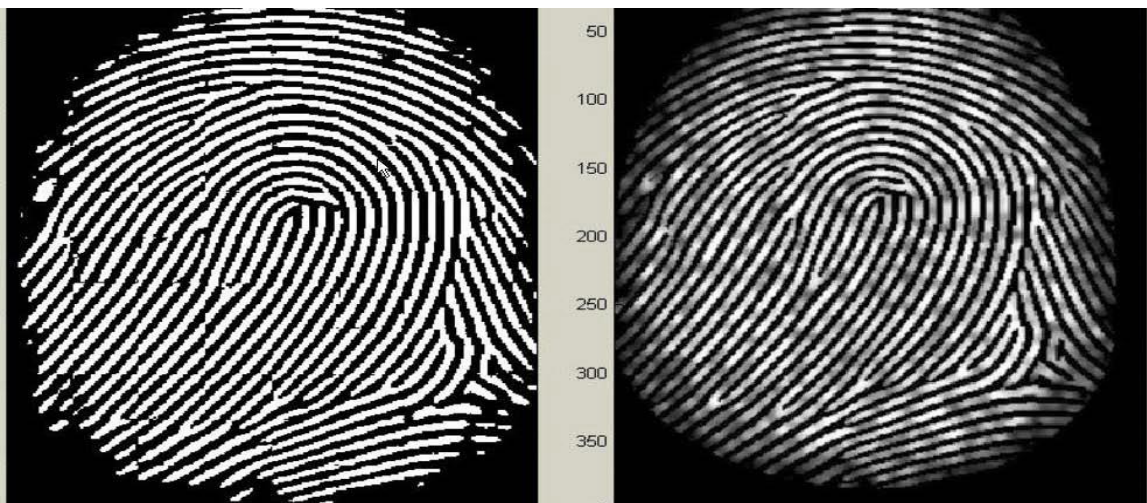


Figure 3.5: Effect of Binarization

Binarized Image

Gray image

3.1.3 Image Segmentation

For a fingerprint image, only a certain portion is important which can provide the required information and can be useful for further processing. This portion is called ROI.

This process of segmentation is carried out in two steps. The first step is block direction estimation and the next ROI extraction by morphological methods. The details of the two steps are as follows

3.1.3.1 Block Direction Estimation

The block direction for every block of the image is estimated. The algorithm is:

- i. Calculation of gradient values for x-direction (p_x) and y-direction (p_y) for each pixel of the block using two Sobel filters.
- ii. Obtaining Least Square Approximation of block direction for each block using the following formula.

$$\tan 2\beta = 2 \sum \sum (p_x * p_y) / \sum \sum (p_x^2 - p_y^2)$$

Considering the gradient values p_x and p_y as cosine value and sine value respectively, the tangent value of block direction can be estimated as given by the following formula:

$$\tan 2\theta = 2 \sin\theta \cos\theta / (\cos^2\theta - \sin^2\theta)$$

The blocks with insignificant information are discarded as mentioned above using the following formula.

$$E = \{2 \sum \sum (p_x * p_y) + \sum \sum (p_x^2 - p_y^2)\} / W * W * \sum \sum (p_x^2 + p_y^2)$$

If certainty level E is found to be less than a threshold, then it is considered as a background block. A direction map is depicted in the figure below.

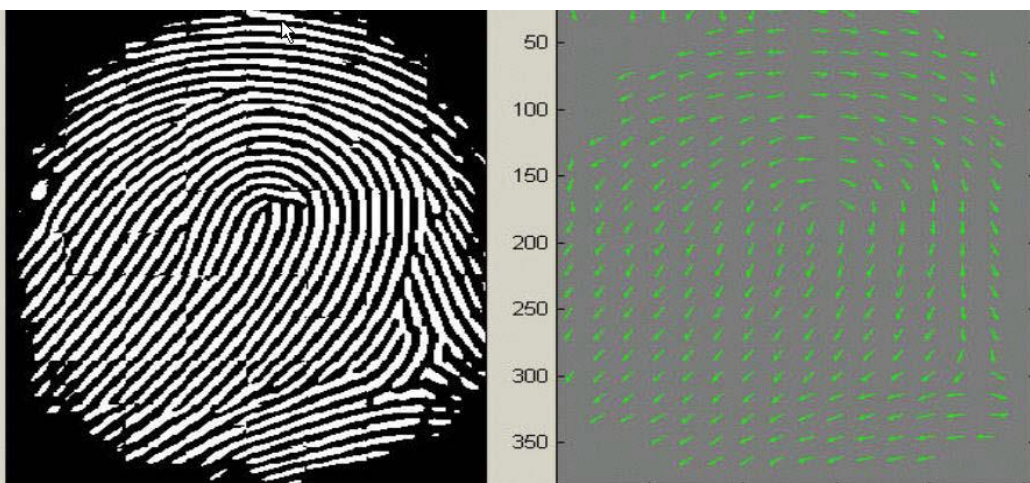


Figure 3.6: Effect of Block Direction Estimation
Direction Map (Right)

3.1.3.2 ROI Extraction by Morphological Methods

Morphological operations divide into two operations “OPEN” and “CLOSE” . The OPEN operation (Figure 3.1.3.3) has capability to inflict enhancement of an image and removal of peaks caused by noise while the CLOSE operation (Figure3.1.3.2) is effective in shrinking images so as to remove small cavities.

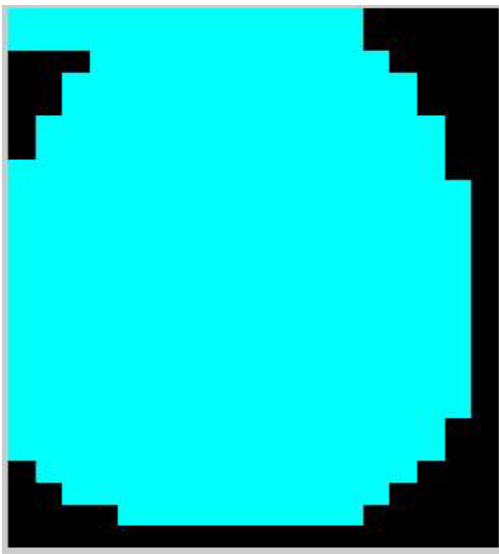


Figure 3.7: Before CLOSE Operation

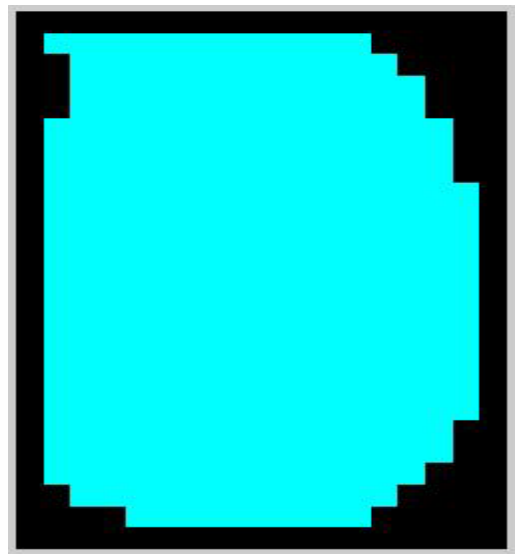


Figure 3.8: After CLOSE Operation

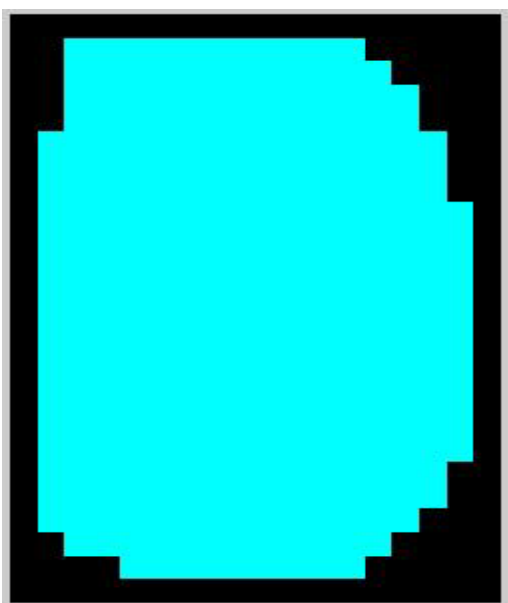


Figure 3.9: After operation OPEN

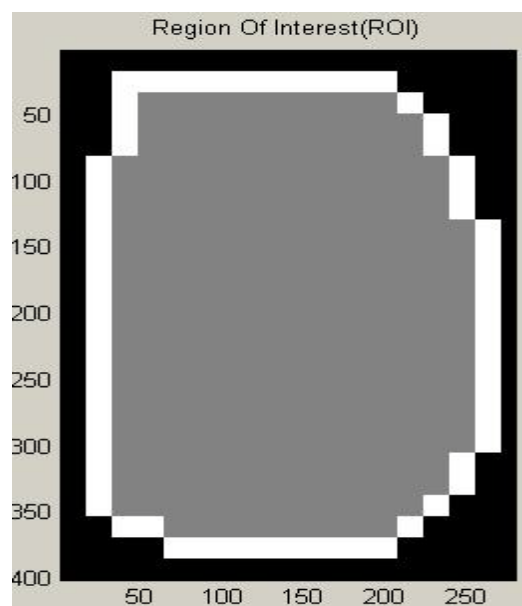


Figure 3.10: ROI + Bound

The bound is the remnant of the closed area out of the opened area. Then the algorithm remove those extreme left, right, upper and bottom blocks out of the calculated area so that we get the bounded region only containing the bound and inner area.

3.2 Minutia Extraction

The minutia extraction process divides into ridge thinning and minutia marking

3.2.1 Ridge Thinning

In ridge thinning process we get rid of repetitive pixels of ridges until the ridges are just one pixel wide. An iterative thinning algorithm is used. In every scan of the full image, the algorithm count repetitive pixels in each small image window. Finally all those marked pixels are removed after several scans. It can extract thinned ridges directly from gray-level fingerprint images. The method traces the ridges with highest gray intensity value..

3.2.2 Minutia Marking

This follows the ridge thinning process. The mechanism behind the minutia marking process is described as follows. For every 3x3 window, if the pixel at the middle is one and has exactly three single-value neighbors, then the pixel is a ridge branch. If the pixel at the middle is 1 and has only one single-valued neighbor, then it means the central pixel is ridge ending.

The mean ridge width D is calculated at this point. The mean inter-ridge width is the mean distance between two nearby ridges. The method to approximate the D is easy. A row of the thinned ridge is scanned and the pixels with value one re summed up. Then the row length is divided with the summation above to get inter ridge width. For better results, such row scans are performed several times and column scans too are conducted. Finally the mean of all the widths are calculated to get the D .

3.3 Post-Processing

Post processing is the final step in this fine tune the image by processes like removing false minutia and unifying terminations and bifurcations.

3.3.1 False Minutia Removal

The preprocessing & minutia-extraction stage does not yield the final processed fingerprint image. False minutia such as false ridge breaks because of lack of ink and also ridge cross-connections from ink spill are still present. Also the earlier steps in processing themselves allow some errors. False minutiae can significantly affect accuracy of matching. So mechanisms to remove them are important.

False minutia can be of different types as follows

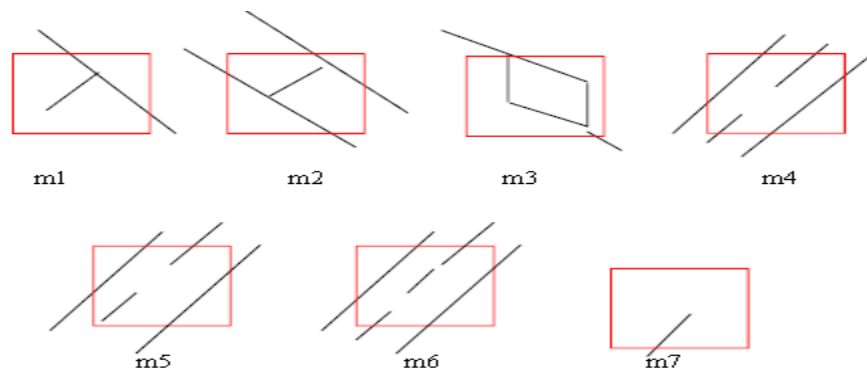


Figure 3.11 False Minutia Structures

m1 is case of a spike entering a ridge. In the m2, a spike connects falsely two valleys. m3 has two branching almost in the same ridge. Moving to m4, the two broken points on the ridge have almost the same orientation and also a short distance. m5 is alike the m4 case with the difference of one part being so short that another end is generated. m6 adds to the m4 case but with the extra condition that a third ridge is found at the center of the parts of the broken ridge. The procedures for removal of false minutia are as follows:

1. If the distance separating a bifurcation and termination is found to be less than D and two minutia belong to one ridge (m1) both of them are eliminated. Where D is the mean inter-ridge distance portraying the mean distance between two parallel nearby ridges.
2. If the width between two bifurcations is found less than D and they belong to one ridge, the two bifurcations are removed (m2, m3).
3. If two endings are within some predetermined distance D and their respective directions match with a small angle variation. And they support the condition that no any other ending is located between the two endings. Then the two terminations are considered to be false and are removed. (m4, m5, m6).

4. If two endings are located in a ridge with width less than D , the two are removed (m7).

3.3.2 Unification of Terminations and Bifurcations

Unification representation is used to avoid interference because of different data acquisition system conditions such as impression pressure. This representation is adopted for both termination and bifurcation. Hence, each individual minutia is characterized by the following parameters:

- 1) x-coordinate
- 2) y-coordinate
- 3) Orientation

Chapter 4

System Design

Fingerprint attendance system can be divided into four different modules. They are:-

1. Fingerprint Capture Module
2. Wireless Module
3. PC based Server-Client Software Management Module
4. GSM Module

The module-wise approach to the design of the system helps in better understanding of the individual function levels.

4.1 Overall System Design Block Diagram

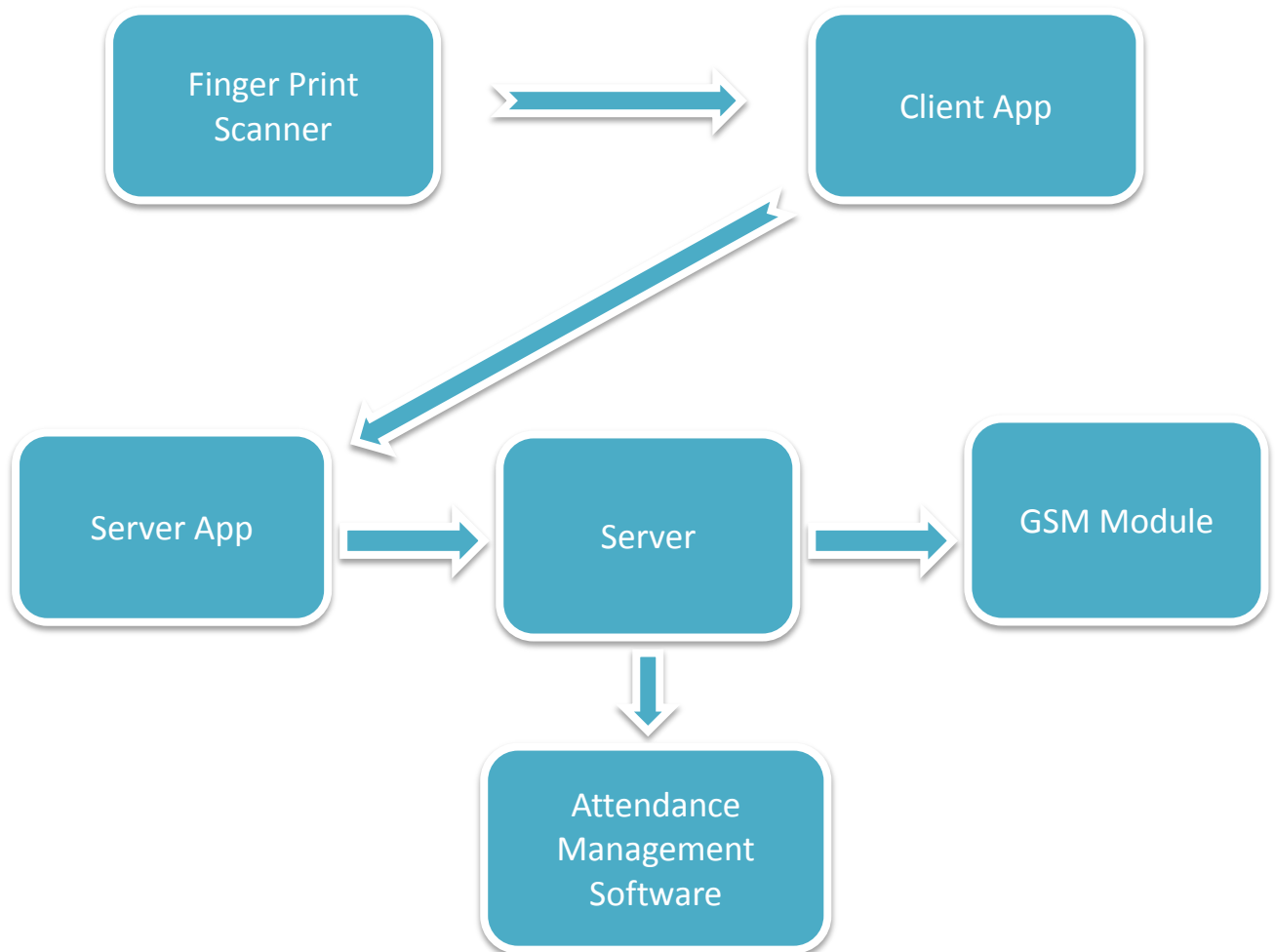


Figure. 4.1: Block Diagram showing the various Modules in the System Design

4.2 Module Design

System modular approach helps in simplifying the design problem. The three modules, i.e. Fingerprint Capture Module, Client Application and Wireless Module form the Client Hardware Modules and Server Application and GSM Module form part of the Server Modules. The modules and their roles are explained below:-

4.2.1 Fingerprint Capture Module (FCM)

The fingerprint capture module is a fingerprint sensor device. It is an electronic device that captures scan of the fingerprint pattern. Then a number of algorithms are applied to the scan to convert it into a biometric template. Generally optical sensors are used because of their ability to produce a clear image, even though other sensors are still in use such as ultrasonic and capacitive sensors.



Figure 4.2: A Fingerprint Sensor

Characteristics of Fingerprint Scanner

Features	Values
Supply Voltage	12-15 VDC
Operating Current	120mA max
Peak Current	150mA max
Interface Protocol	Standard Serial Interface(TTL)
Window Area	14mm x 18mm
Fingerprint Resolution	500dpi

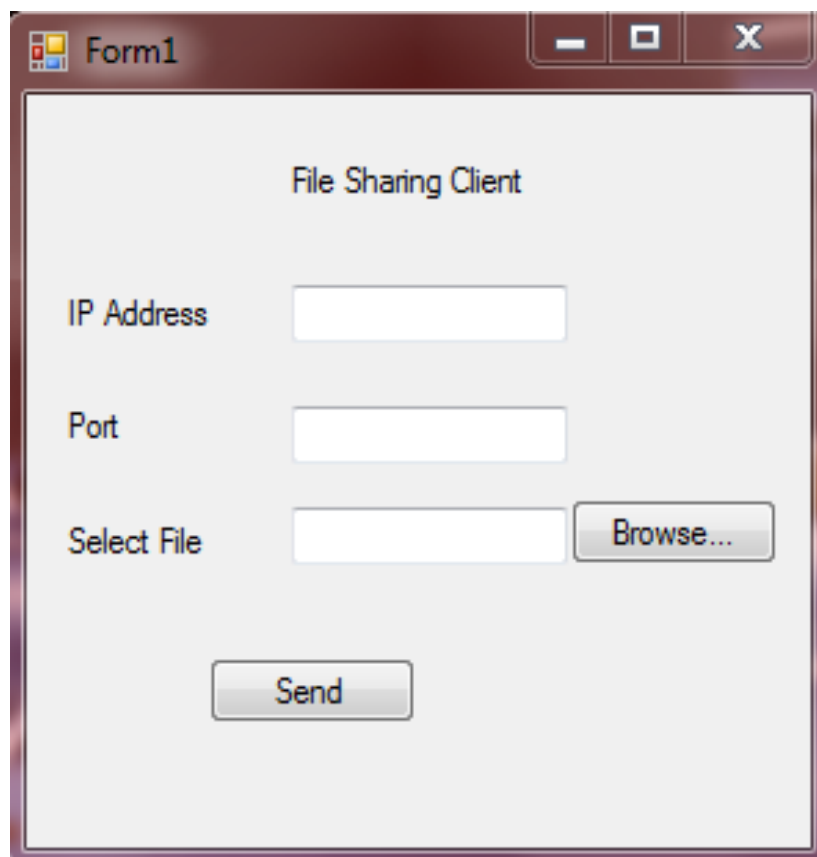
Table 4.1: Characteristics of Fingerprint Scanner

4.2.2 Client Application Module

The Client Application is developed for the transmission of Fingerprint template generated by the fingerprint scanner. The main features of the application are:-

1. It is programmed in C# and runs in classrooms or other desired places.
2. Client App asks for IP address of the server and port number on which App is running.
3. It also asks for the fingerprint template of the student taken using fingerprint scanner, to be transmitted to server for attendance management.

GUI of Client Application



The screenshot displays a Windows-style window titled "Form1". The main content area is titled "File Sharing Client". It contains three input fields: "IP Address", "Port", and "Select File". The "Select File" field is accompanied by a "Browse..." button. Below these fields is a "Send" button. The window has standard minimize, maximize, and close controls in the top right corner.

Figure 4.3: Client Application

4.2.2 Wireless Communication Module (WCM)

To access the server wirelessly, client server app is designed to run on local network in this case WLAN network. In order to make our system cost effective we are using WLAN network (Wi-Fi) which is already established in most of the organizations and institutions.



Figure 4.4: Wireless Module

Both the Client and the server are connected to the same network and client application is running on the client side which will be used to transfer the fingerprint template generated by the fingerprint scanner to the server for attendance management.

4.2.4 Server Application Module

The Server Application Module is developed for the reception of fingerprint templates being transmitted by the Client System using WLAN. The key features of Server Application are:-

1. It is programmed in C# and runs on Server containing fingerprint database.
2. Server App requires port number on which application is running.
3. It receives the finger template sent using Wi-Fi network form Client Application running in classroom.

GUI of Server Application

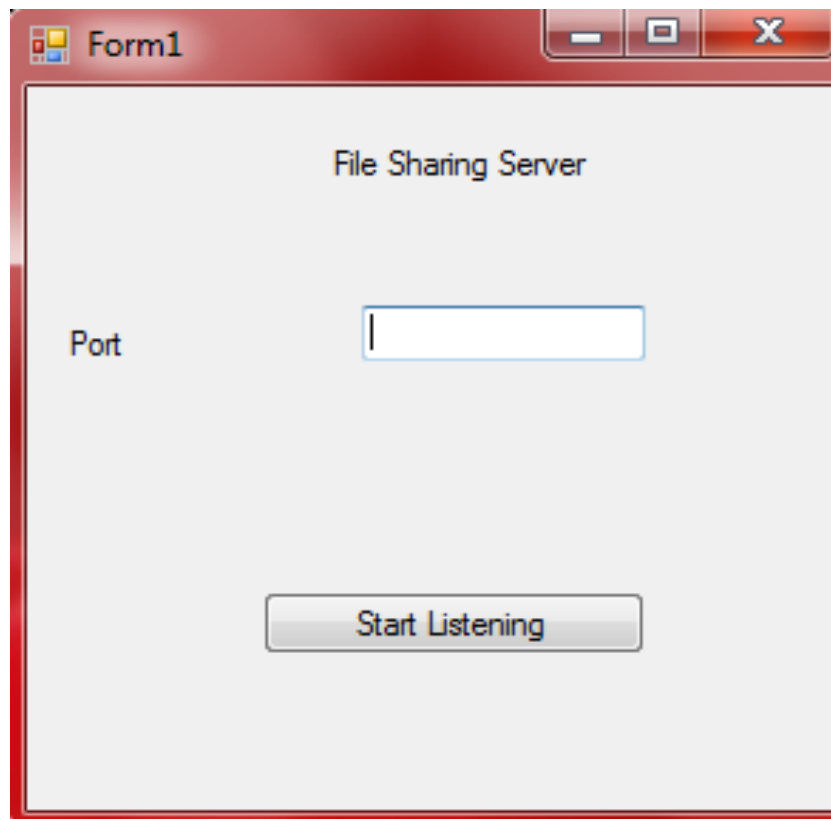


Figure 4.5: Server Application

4.2.5 GSM Module

GSM Module is interfaced with the server to send the verification SMS to the students so that they should also be intimated once their attendance has been marked and it is also useful in the cases where due to some reason their attendance cannot be marked at the first attempt. Key features of GSM Module are:-

1. SIM 900D GSM module is used to send verification message to user.
2. This module is interfaced with Server containing fingerprint database.
3. When fingerprint template is matched with the database entry, it pop ups for verification application to send message to user.
4. GUI has been developed for user interface which is further linked to the database.

4.3 Algorithm Design

Software is responsible for the implementing the following functions:-

- Fingerprint Capture
- Fingerprint Image Processing
- Wireless Data Transfer
- Updating the database and attendance sheets
- Maintenance of GUI to Student Attendance System

Chapter 5

Data Transfer and Application Development

In this chapter we will discuss the transfer of data from client to server in a wireless environment and applications developed for data transmission in a client-server environment.

5.1 Wireless Data Transfer

After the fingerprint image has been processed, the data is to be transferred to the central server through a wireless channel. The data packet is to be coded into an encrypted form due to the sensitive nature of the information it carries. The data communicated to the server is broadly classified into two types:

- a. Enroll Data
- b. Daily Attendance Data

5.1.1 Enroll Data

This data is initially obtained when adding the new students to the institute database. Along with Personal Identification Numbers (PIN), student-specific data such as degree program, date of birth (DOB), student picture & signature, the database is provided with a biometric template consisting of a processed image of the fingerprint.

5.1.2 Daily Attendance Data

Once all the students are enrolled into the institute's Student Attendance System, the daily work of each Client is to accumulate the attendance data for each student of a particular classroom and transmit the data to the Central Server System (CSS).

5.2 Application Development

The actual testing for the design of the wireless fingerprint based student attendance system was carried out in Communications Lab. Department of Electrical

Engineering. The experimental setup consists of both software based platform and hardware module in an integrated development environment. The various components of the testing environment are:-

- a. Client PC
- b. Server PC
- c. Wireless Access Point
- d. GSM Module
- e. Visual Studio 2012

5.2.1 Client PC

If we talk about any institution now a days each and every classroom has a PC installed in it for delivering the lectures to the students. We made use of same PC for our project as to make it more cost efficient and to merge it with the existing setup. The only requirement we are having is of wireless connectivity for that purpose we can use any of the Wi-Fi device which is compatible with the PCs installed at the class rooms. For our experimental purpose we have used the wireless access point manufactured by TP-Link.

The Client PC has the fingerprint scanner connected to it through USB interface which will be used to get the fingerprint of the student or employee of the organization. Results generated by the fingerprint scanner during the experiments are as follows:-

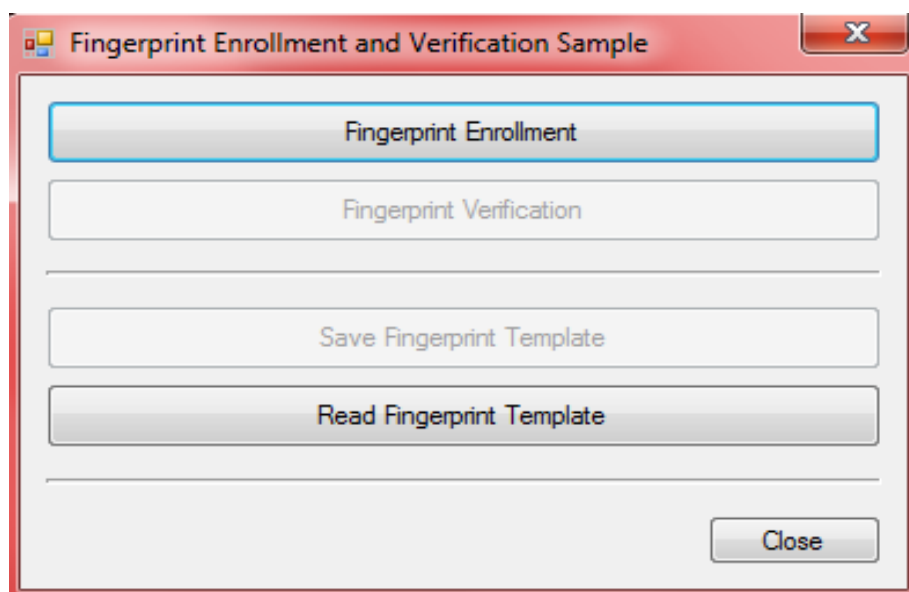


Figure 5.1: Fingerprint Acquisition Application

Fingerprint acquisition application has been developed which is used to generate the fingerprint template which will be further used for attendance management purpose.

In this GUI we have four options:-

- a. Fingerprint Enrollment
- b. Fingerprint Verification
- c. Save Fingerprint Template
- d. Read Fingerprint Template



Figure 5.2: Adil Fingerprint



Figure 5.3: Masood Fingerprint

Second thing is the client file sharing application installed and running on it. The Client Application has following requirements:-

- a. IP address of the Server
- b. Port Number of Server
- c. Fingerprint template generated by fingerprint scanner

The running Client Application is shown in Figure 5.1 which is connected to the server through Port No 4000.

Figure 5.4: Client Application

The first field is of IP address of the Server whom the client wants to communicate using WLAN. The second field is of Port Number of the server on which the server is listening. The third field is of the location of the template file which needs to be transmitted to the server.

5.2.2 Server PC

Server is the central controller which is managing the attendance record of the student or employees in case of an organization. Server file sharing application needs to be installed on the server and GSM module is also connected to the server. The server is also connected to the client PC through Wi-Fi.

The Server PC has the server file sharing application installed and running on it. The Server Application has the requirement of Port Number of Server.

It automatically stores the fingerprint template received from the client to one of the directory which was predefined during programming of the application. This template is then linked to the database for attendance management purpose.

The running Server Application is shown in Figure 5.2 which is connected to the server through Port No 4000.

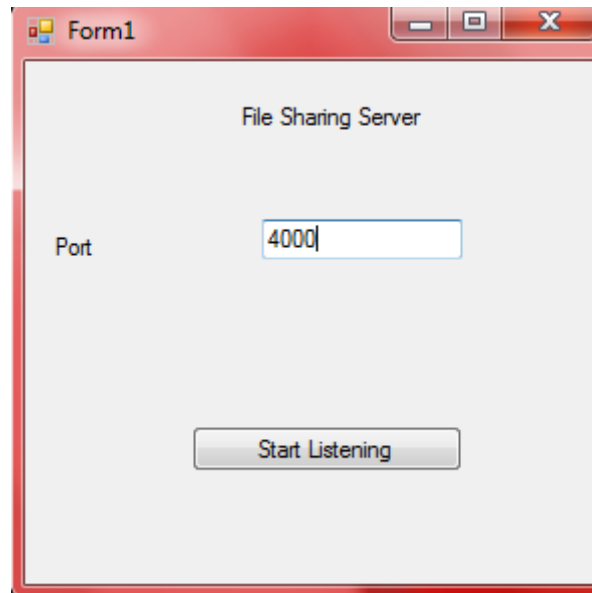


Figure 5.5: Server Application

The only thing required by the user in this application is the port number on which the client is communicating to the server, once we enter the port number it will start listening to the client continuously until the application is closed.

5.2.3 GSM Module

The GSM module is interfaced to the server PC for sending the verification SMS to the student in case of institution or employee in case of an organization. The GSM module is connected to the Server through serial interface and the application has been developed to send an verification SMS to the users, the application is further linked to the data base of the institution or organization, once the fingerprint template will be received the SMS application will automatically be called do that the verification SMS can be sent to the user.

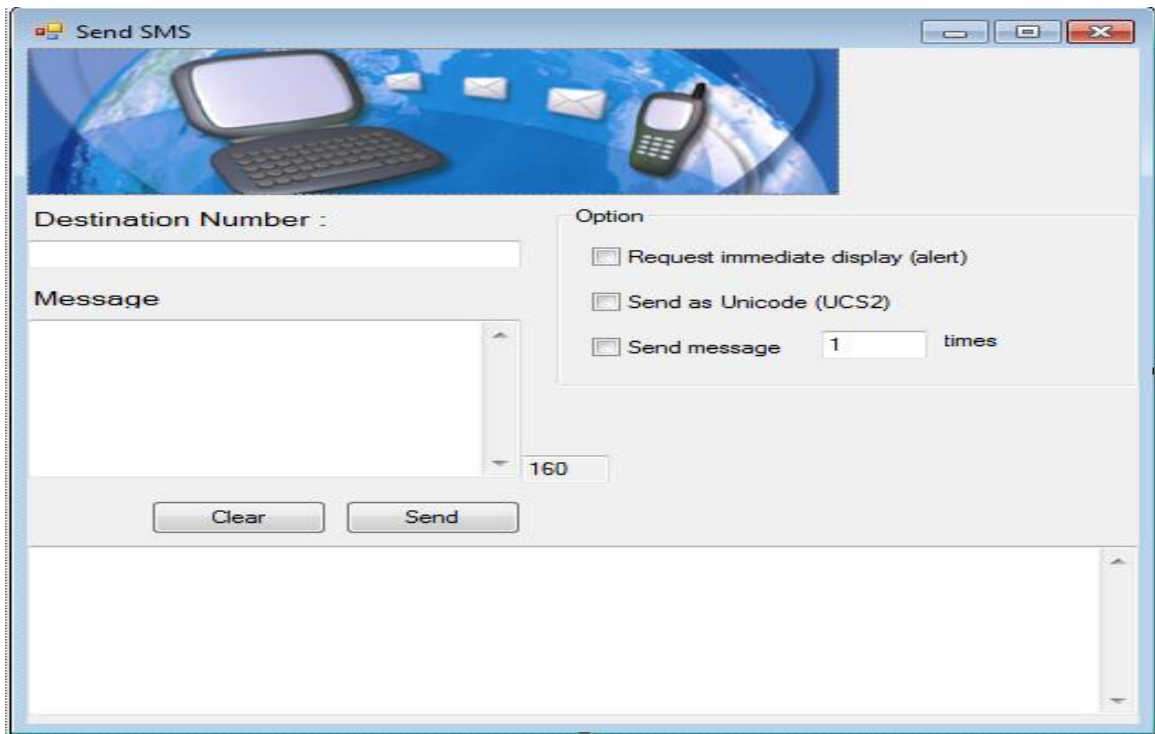


Figure 5.6: GSM Application

The application requires the destination number on which we want to send the verification SMS, these numbers will be stored in the database during the enrolment purposes and will be used for the purpose of sending the verification SMS to the user once his/her attendance will be marked. The message field will contain the Text which we want to send in the verification message.

5.2.4 Database

The database management is done on the server. The database is made in such a way that it can enroll the new students or employees and later on it can be used to maintain their attendance as well. The key features of database are

- a. Database on Server contains student's data and their fingerprints templates.
- b. Software in C# which match receiving fingerprint with existing one and mark attendance.
- c. Graphical User
- d. Interface (GUI) has developed to register fingerprint template and matching details.

The image shows a graphical user interface (GUI) window titled "Form1". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is light gray and contains the following elements:

- Four text input fields stacked vertically, each with a label to its left: "First Name", "Last Name", "Phone No", and "File Name".
- An "Upload" button located below the "File Name" field.
- A text input field labeled "Enter File name to be Matched in Database" below the "Upload" button.
- A text input field labeled "Select file to compare with" below the "Enter File name..." field.
- A "Select" button located below the "Select file to compare with" field.
- A "Match Files" button located below the "Select" button.

Figure 5.7: Database Application

This GUI is used first to enroll the student with following details:-

- a. First Name
- b. Last Name
- c. Phone No
- d. File Name (Name of Fingerprint template)

Upload option is use to save the details of the student in the database from where it can be accessed at later stage for attendance management purposes.

For verification and attendance marking purpose the lower fields are used which include:-

- a. File Name to be matched
- b. File to compare with

“File name to be matched in database” is the template of the user which was uploaded during enrollment process and “File to compare with” is the new fingerprint template received during attendance marking process. Once both the files match the attendance of the particular student/employee will be marked.

Chapter 6

Experimental Setup and Testing

6.1 Physical Layout

The actual testing of the system was carried out in the Communication Lab of Electrical Engineering Department. The Figure 6.1 shows the physical layout of the project. Due to compact space the Client PC (Laptop) and the Server PC (Desktop) both are placed side by side but both are connected through WLAN. Access Point of TP-Link is used for wireless communication.



Figure 6.1: Physical Layout

6.2 Fingerprint Template Generation

In this section we will discuss the generation of the fingerprint template and its transmission to the server.

6.2.1 Fingerprint Enrollment

Once the user is getting himself/herself enrolled for the first time the scanner will require the fingerprint four times to properly extract the features of the individual so that the chances of false detection can be reduced. The application developed for the

fingerprint scanner has four Tabs which can be used for various purposes, the first one is of “Fingerprint Enrollment”. The user will press his/her finger four times against the scanner to get his fingerprint enrolled. The Figure 6.2 shows the enrollment process of the user.



Figure 6.2: Fingerprint Enrollment

Once the user will press the finger against the scanner four times properly and scanner was able to get the fingerprint it will give the message for successful capturing of the fingerprint. Figure 6.3 shows the successful capturing of the image for enrollment purpose.

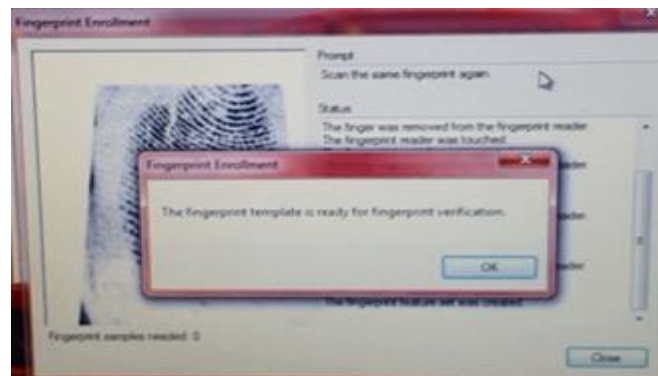


Figure 6.3: Successful Fingerprint Enrollment

6.2.2 Fingerprint Verification

Now at this stage we are ready to verify the finger print so as to check the accuracy of the scanner working, the option has been given in the application to get the fingerprint verified at this stage as well. Figure 6.4 shows the successful verification of the fingerprint. It is also worth mentioning that the scanner is able to verify the image even if the finger is placed 180 degree inverted on the scanner for verification.



Figure 6.4: Successful Verification of Fingerprint

Figure 6.5 shows the unsuccessful verification of the fingerprint it happens if we place any other finger other than which is enrolled even if we place the thumb or same finger of the other hand it will be unsuccessful.



Figure 6.4: Unsuccessful Verification of Fingerprint

6.2.3 Fingerprint Template

Once the fingerprint is verified its template can be generated using the same application so that it can be enrolled in the database which will be further used for attendance management purpose. Figure 6.5 shows the GUI which can be used for template generation purpose, once we will click on the “Save Fingerprint Template” it will ask for the location where we want to save the template and will generate and save the template of the finger print at that particular location.

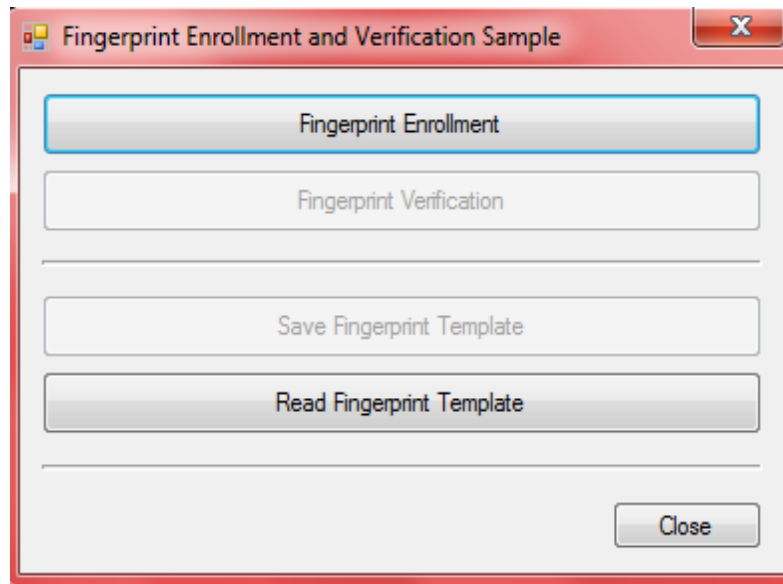


Figure 6.5: Template Generation

6.3 Client-Server Communication

Once the fingerprint template is generated it is ready to be transmitted to the server for enrollment in the database. Client and Server are communicated wirelessly through WLAN environment. A Client application is installed and running on the client PC where it has the IP address of the server and the port number on which the client want to transmit the fingerprint template. The last field is of the location of the fingerprint template which needs to be transmitted. Figure 6.6 shows the running client application ready for the transmission of fingerprint template.

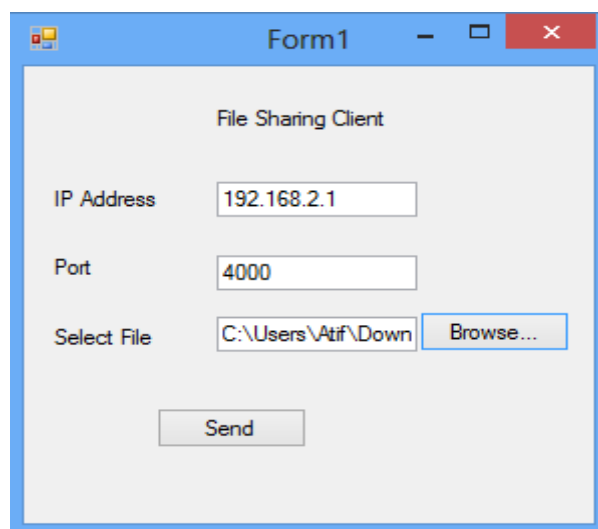


Figure 6.6: Client Application

Once we will press the send button the template will be transmitted to the server through WLAN. Figure 6.7 shows the successful transmission of the fingerprint template.

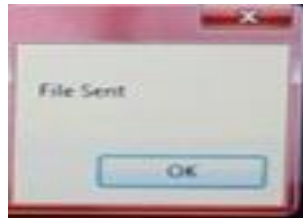


Figure 6.7: Successful Transmission by Client

6.4 Attendance Management at Server

Once the template is received by the server the rest of the task is of server to use the finger print for enrollment or for attendance management purpose as required by the system. Server has to perform several tasks like enrollment, attendance management and sending verification SMS to the user on attendance marking.

6.4.1 Enrollment in Database

Once the fingerprint template is successfully received by the server it will make an entry against that particular user in the database for the first time later on once the same template will be received by the server it will check through the entries and mark the attendance of that particular user. Figure 6.8 shows the successful reception of the fingerprint template by the server.

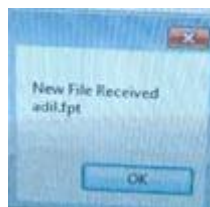


Figure 6.7: Successful Reception by Server

6.4.2 Verification SMS using GSM Module

Once the match will be found against the fingerprint template in the existing database the attendance of that particular user will be marked and database is linked to the SMS application which will use the GSM module to send the verification SMS to the user . The message is predefined which will be sent to the mobile number entered in the database during enrollment process.

Chapter 7

7.1 Conclusion

This project mainly comprised of development of attendance management system and fingerprint identification system. Attendance management is very helpful in saving valuable time of students and teachers, paper and generating report at required time. This project presented a framework using which attendance management can be made automated and on-line. A general implementable approach to attendance management was proposed using WLAN. Further, an idea for using portable devices along with wireless LAN or mobile 3G network was suggested. Fingerprint Identification System used for student identification is faster in implementation than any other fingerprint identification systems. For fingerprint recognition, prevalent enhancement techniques like Gabor filters, minutiae extraction using Crossing Number concept followed by spurious and boundary minutiae removal, fingerprint classification, reference point detection, etc. are employed.

7.2 Outcomes of this Project

1. A Scientific approach was developed during project work.
2. Skills and self-confidence in coding and working with softwares like C# and Visual Studio were developed.
3. An applicable attendance management system was designed for educational institutions and other organizations. Ideas were presented for making whole system online using WLAN technology.
4. An improved and faster fingerprint identification system was developed for student identification purpose.
5. The future expectations from this project is to actually implement such system for one or more classes if sufficient funds are provided to us.
6. Our fingerprint identification system can be used in implementation in Military College of Signals.

7.3 Future Work

There is a lot of scope in the field of biometrics application at the work place. The attendance system using fingerprint recognition can be of real use if certain nuances are taken into consideration. The wireless channel used was limited to a short range and hence the system could only be tested in the lab. For a greater range and more versatile application, a different channel could be considered which would ensure faster data transfer and provide better flexibility. The security aspect of transmission can be worked upon since data security in case of sensitive data transfer is highly essential.

Finally, the proposed model for client-server software management system can be materialized using cost effective products offered in the market.

Appendix

Most of the coding for the software development is done in C sharp. Coding is done for the following modules:-

1. Fingerprint Scanner Module
2. Client- Server Communication Module
3. GSM Module
4. Database Development

The code for each of the module will be explained in detail in subsequent topics.

Fingerprint Scanner Code

The code is written in C#. The code for the Fingerprint Scanner is given below

```
clear;
clc;
close all;
global imagine n_bands h_bands n_arcs h_radius h_lato
n_sectors matrice num_disk

n_bands=4;
h_bands=20;
n_arcs=16;
h_radius=12;
h_lato=h_radius+(n_bands*h_bands*2)+16;
if mod(h_lato,2)==0
    h_lato=h_lato-1;
end
n_sectors=n_bands*n_arcs;
matrice=zeros(h_lato);
for ii=1:(h_lato*h_lato)
    matrice(ii)=whichsector(ii);
end
num_disk=8;
% 1--> add database
% 0--> recognition
%ok=0;
chos=0;
possibility=7;

messaggio='Insert the number of set: each set determines a
class. This set should include a number of images for
each person, with some variations in expression and in
the lighting.';
```

```

while chos~=possibility,
    chos=menu('Fingerprint Recognition System','Select
image and add to database','Select image for fingerprint
recognition','Info','Delete database',...
    'Fingerprint image: visualization','Gabor Filter:
visualization','Exit');
    %-----
    %-----
    %-----
    %-----
    % Calculate FingerCode and Add to Database
    if chos==1
        clc;
        close all;
        selezionato=0;
        while selezionato==0

[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;
*.jpeg;*.gif','IMAGE Files
(*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'),'Chose
GrayScale Image');
            if namefile~=0

[img,map]=imread(strcat(pathname,namefile));
                selezionato=1;
            else
                disp('Select a grayscale image');
            end
            if (any(namefile~=0) && (~isgray(img)))
                disp('Select a grayscale image');
                selezionato=0;
            end
        end

        immagine=double(img);

        if isa(img,'uint8')
            graylevmax=2^8-1;
        end
        if isa(img,'uint16')
            graylevmax=2^16-1;
        end
        if isa(img,'uint32')
            graylevmax=2^32-1;
        end
        fingerprint = immagine;

        N=h_lato;

```

```

[BinarizedPrint, XofCenter, YofCenter]=centralizing(fingerp
rint,0);

[CroppedPrint]=cropping(XofCenter, YofCenter, fingerprint);

[NormalizedPrint, vector]=sector_norm(CroppedPrint,0);

    for (angle=0:1:num_disk-1)
        gabor=gabor2d_sub(angle, num_disk);

ComponentPrint=conv2fft(NormalizedPrint, gabor, 'same');
    [disk, vector]=sector_norm(ComponentPrint,1);
    finger_code1{angle+1}=vector(1:n_sectors);
    end

    img=imrotate(img, 180/(num_disk*2));
    fingerprint=double(img);

[BinarizedPrint, XofCenter, YofCenter]=centralizing(fingerp
rint,0);

[CroppedPrint]=cropping(XofCenter, YofCenter, fingerprint);

[NormalizedPrint, vector]=sector_norm(CroppedPrint,0);

    for (angle=0:1:num_disk-1)
        gabor=gabor2d_sub(angle, num_disk);

ComponentPrint=conv2fft(NormalizedPrint, gabor, 'same');
    [disk, vector]=sector_norm(ComponentPrint,1);
    finger_code2{angle+1}=vector(1:n_sectors);
    end
    % FingerCode added to database
    if (exist('fp_database.dat')==2)
        load('fp_database.dat', '-mat');
        fp_number=fp_number+1;
        data{fp_number,1}=finger_code1;
        data{fp_number,2}=finger_code2;
        save('fp_database.dat', 'data', 'fp_number', '-
append');
    else
        fp_number=1;
        data{fp_number,1}=finger_code1;
        data{fp_number,2}=finger_code2;
        save('fp_database.dat', 'data', 'fp_number');
    end

    message=strcat('FingerCode was succesfully added
to database. Fingerprint no. ', num2str(fp_number));
    msgbox(message, 'FingerCode DataBase', 'help');

```



```

end
%-----
-----
%-----
-----
%-----
-----
% Fingerprint recognition
if chos==2
    clc;
    close all;
    selezionato=0;
    while selezionato==0

[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;
*.jpeg;*.gif','IMAGE Files
(*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'),'Chose
GrayScale Image');
        if namefile~=0

[img,map]=imread(strcat(pathname,namefile));
            selezionato=1;
            else
                disp('Select a grayscale image');
            end
            if (any(namefile~=0) && (~isgray(img)))
                disp('Select a grayscale image');
                selezionato=0;
            end
        end

        immagine=double(img);

        if isa(img,'uint8')
            graylevmax=2^8-1;
        end
        if isa(img,'uint16')
            graylevmax=2^16-1;
        end
        if isa(img,'uint32')
            graylevmax=2^32-1;
        end
        fingerprint = immagine;

        N=h_lato;

[BinarizedPrint,XofCenter,YofCenter]=centralizing(fingerp
rint,0);

[CroppedPrint]=cropping(XofCenter,YofCenter,fingerprint);

[NormalizedPrint,vector]=sector_norm(CroppedPrint,0);

```

```

% memoria per feature vector d'ingresso
vettore_in=zeros(num_disk*n_sectors,1);
for (angle=0:1:num_disk-1)
    gabor=gabor2d_sub(angle,num_disk);

ComponentPrint=conv2fft(NormalizedPrint,gabor,'same');
    [disk,vector]=sector_norm(ComponentPrint,1);
    finger_code{angle+1}=vector(1:n_sectors);

vettore_in(angle*n_sectors+1:(angle+1)*n_sectors)=finger_
code{angle+1};
    end

% FingerCode of input fingerprint has just been
calculated.
% Checking with DataBase
if (exist('fp_database.dat')==2)
    load('fp_database.dat','-mat');
    %---- alloco memoria -----
-----
    %...
    vettore_a=zeros(num_disk*n_sectors,1);
    vettore_b=zeros(num_disk*n_sectors,1);
    best_matching=zeros(fp_number,1);
    valori_rotazione=zeros(n_arcs,1);
    % start checking -----
-----
    for scanning=1:fp_number
        fcode1=data{scanning,1};
        fcode2=data{scanning,2};
        for rotazione=0:(n_arcs-1)
            p1=fcode1;
            p2=fcode2;
            % ruoto i valori dentro disco
            for conta_disco=1:num_disk
                discol=p1{conta_disco};
                disco2=p2{conta_disco};
                for old_pos=1:n_arcs

new_pos=mod(old_pos+rotazione,n_arcs);
                    if new_pos==0
                        new_pos=n_arcs;
                    end
                    for conta_bande=0:1:(n_bands-
1)

discolr(new_pos+conta_bande*n_arcs)=discol(old_pos+conta_
bande*n_arcs);

```

```

disco2r(new_pos+conta_bande*n_arcs)=disco2(old_pos+conta_
bande*n_arcs);
        end
        end
        p1{conta_disco}=disco1r;
        p2{conta_disco}=disco2r;
    end
    % ruoto i dischi circolarmente
    for old_disk=1:num_disk

new_disk=mod(old_disk+rotazione,num_disk);
        if new_disk==0
            new_disk=num_disk;
        end
        pos=old_disk-1;

vettore_a(pos*n_sectors+1:(pos+1)*n_sectors)=p1{new_disk}
;

vettore_b(pos*n_sectors+1:(pos+1)*n_sectors)=p2{new_disk}
;

        end
        d1=norm(vettore_a-vettore_in);
        d2=norm(vettore_b-vettore_in);
        if d1<d2
            val_minimo=d1;
        else
            val_minimo=d2;
        end

valori_rotazione(rotazione+1)=val_minimo;
        end

[minimo,posizione_minimo]=min(valori_rotazione);
        best_matching(scanning)=minimo;
    end

[distanza_minima,posizione_minimo]=min(best_matching);
        beep;
        message=strcat('The nearest fingerprint
present in DataBase which matchs input fingerprint is :
',num2str(posizione_minimo),...
        ' with a distance of :
',num2str(distanza_minima));
        msgbox(message,'DataBase Info','help');

    else
        message='DataBase is empty. No check is
possible.';
        msgbox(message,'FingerCode DataBase
Error','warn');
    end
end

```

```

end % fine caso 2
if chos==3
    clc;
    close all;
    helpwin fprec;
end % fine caso 3
if chos==4
    clc;
    close all;
    if (exist('fp_database.dat')==2)
        button = questdlg('Do you really want to
remove the Database?');
        if strcmp(button,'Yes')
            delete('fp_database.dat');
            msgbox('Database was succesfully removed
from the current directory.','Database removed','help');
        end
    else
        warndlg('Database is empty.',' Warning )
    end
end % fine caso 4
if chos==5
    clc;
    close all;
    selezionato=0;
    while selezionato==0

[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;
*.jpeg;*.gif','IMAGE Files
(*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'),'Chose
GrayScale Image');
        if namefile~=0

[img,map]=imread(strcat(pathname,namefile));
            selezionato=1;
        else
            disp('Select a grayscale image');
        end
        if (any(namefile~=0) && (~isgray(img)))
            disp('Select a grayscale image');
            selezionato=0;
        end
    end
    figure('Name','Selected image');
    imshow(img);
end % fine caso 5
if chos==6
    clc;
    close all;
    figure('Name','Gabor Filter');
    mesh(gabor2d_sub(0,num_disk));
end % fine caso 6
end % fine while

```

Client- Server Communication

The code is written in C#. The code for Client-Server Communication Module is given as follows.

Client Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;
using System.Net.NetworkInformation;
using System.IO;

namespace form
{
    public partial class Form1 : Form
    {
        private static string shortFileName = "";
        private static string fileName = "";
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void browse_Click(object sender,
EventArgs e)
        {
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.Title = "File Sharing Client";
            dlg.ShowDialog();
            file.Text = dlg.FileName;
            fileName = dlg.FileName;
            shortFileName = dlg.SafeFileName;
        }

        private void send_Click(object sender, EventArgs e)
        {
            string ipAddress = ip.Text;

```

```

        int port = int.Parse(port1.Text);
        string fileName = file.Text;
        Task.Factory.StartNew(() =>
SendFile(ipAddress, port, fileName, shortFileName));
        MessageBox.Show("File Sent");
    }
    public void SendFile(string remoteHostIP, int
remoteHostPort,
        string longFileName, string shortFileName)
    {
    try
    {
    if (!string.IsNullOrEmpty(remoteHostIP))
    {
    byte[] fileNameByte =
Encoding.ASCII.GetBytes(shortFileName);
    byte[] fileData = File.ReadAllBytes(longFileName);
    byte[] clientData = new byte[4 + fileNameByte.Length +
fileData.Length];
    byte[] fileNameLen =
BitConverter.GetBytes(fileNameByte.Length);
        fileNameLen.CopyTo(clientData, 0);
        fileNameByte.CopyTo(clientData, 4);
        fileData.CopyTo(clientData, 4 + fileNameByte.Length);
        TcpClient clientSocket = new TcpClient(remoteHostIP,
remoteHostPort);
        NetworkStream networkStream = clientSocket.GetStream();
        networkStream.Write(clientData, 0,
clientData.GetLength(0));
        networkStream.Close();
    }
    }
    catch
    {
    }
    }
    }
}

```

Server Code

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;

```

```

using System.Net.Sockets;
using System.Net.NetworkInformation;
using System.IO;
using System.Threading.Tasks;

namespace FileSharingServer
{
    public partial class Form1 : Form
    {
        public delegate void
FileRecievedEventHandler(object source, string fileName);
        public event FileRecievedEventHandler
NewFileRecieved;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.NewFileRecieved += new
FileRecievedEventHandler(Form1_NewFileRecieved);
        }

        private void Form1_NewFileRecieved(object sender,
string fileName)
        {
            this.BeginInvoke(new Action(delegate()
{
                MessageBox.Show("New File Received\n" +
fileName);

System.Diagnostics.Process.Start("explorer", @"c:\");
            }));
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int port = int.Parse(port1.Text);
            Task.Factory.StartNew(() =>
HandleIncomingFile(port));
            MessageBox.Show("Listening on port" + port);
        }
        public void HandleIncomingFile(int port)
        {
            try
            {
                TcpListener tcpListener = new
TcpListener(port);
                tcpListener.Start();
                while (true)
                {

```

```

        Socket handlerSocket =
tcpListener.AcceptSocket();
        if (handlerSocket.Connected)
        {
            string fileName = string.Empty;
            NetworkStream networkStream = new
NetworkStream(handlerSocket);
            int thisRead = 0;
            int blockSize = 1024;
            Byte[] dataByte = new
Byte[blockSize];

            lock (this)
            {
                string folderPath = @"c:\";
                int receivedBytesLen =
handlerSocket.Receive(dataByte);
                int fileNameLen =
BitConverter.ToInt32(dataByte, 0);
                fileName =
Encoding.ASCII.GetString(dataByte, 4, fileNameLen);
                Stream fileStream =
File.OpenWrite(folderPath + fileName);
                fileStream.Write(dataByte, 4
+ fileNameLen, (1024 - (4 + fileNameLen)));
                while (true)
                {
                    thisRead =
networkStream.Read(dataByte, 0, blockSize);

                    fileStream.Write(dataByte, 0, thisRead);
                    if (thisRead == 0)
                        break;
                }
                fileStream.Close();
            }
            if (NewFileRecieved != null)
            {
                NewFileRecieved(this,
fileName);
            }
            handlerSocket = null;
        }
    }
}
catch { }
}
}
}

```


GSM Module Code

The code is written in C#. The code for GSM module for sending the SMS is given below.

```

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

using GsmComm.PduConverter;
using GsmComm.GsmCommunication;

namespace SMS
{
    /// <summary>
    /// Summary description for Send.
    /// </summary>
    public class Send : System.Windows.Forms.Form
    {
        private System.Windows.Forms.PictureBox
pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox
txt_destination_numbers;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox
txt_text_remaining;
        private System.Windows.Forms.Button
btnSendMessage;
        private System.Windows.Forms.Button BtnClear;
        private System.Windows.Forms.TextBox
txt_message;
        private System.Windows.Forms.TextBox txtOutput;
        private System.Windows.Forms.CheckBox
chkUnicode;
        private System.Windows.Forms.CheckBox chkAlert;
        private System.Windows.Forms.GroupBox
groupBox1;
        private System.Windows.Forms.Label label19;
        private System.Windows.Forms.TextBox
txtSendTimes;
        private System.Windows.Forms.CheckBox
chkMultipleTimes;
        /// <summary>
        /// Required designer variable.
        /// </summary>

```

```

        private System.ComponentModel.Container
components = null;

        private delegate void SetTextCallback(string
text);

        public Send()
        {
            //
            // Required for Windows Form Designer
support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after
InitializeComponent call
            //
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing
)
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do
not modify
        /// the contents of this method with the code
editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Resources.ResourceManager resources
= new System.Resources.ResourceManager( typeof( Send ) );
            this.pictureBox1 = new
System.Windows.Forms.PictureBox();
            this.labell = new
System.Windows.Forms.Label();
            this.txt_destination_numbers = new
System.Windows.Forms.TextBox();

```

```

        this.label2 = new
System.Windows.Forms.Label();
        this.txt_message = new
System.Windows.Forms.TextBox();
        this.txt_text_remaining = new
System.Windows.Forms.TextBox();
        this.btnSendMessage = new
System.Windows.Forms.Button();
        this.BtnClear = new
System.Windows.Forms.Button();
        this.txtOutput = new
System.Windows.Forms.TextBox();
        this.chkUnicode = new
System.Windows.Forms.CheckBox();
        this.chkAlert = new
System.Windows.Forms.CheckBox();
        this.groupBox1 = new
System.Windows.Forms.GroupBox();
        this.txtSendTimes = new
System.Windows.Forms.TextBox();
        this.chkMultipleTimes = new
System.Windows.Forms.CheckBox();
        this.label19 = new
System.Windows.Forms.Label();
        this.groupBox1.SuspendLayout();
        this.SuspendLayout();
        //
        // pictureBox1
        //
        this.pictureBox1.BackgroundImage =
((System.Drawing.Image) (resources.GetObject("pictureBox1.
BackgroundImage")));
        this.pictureBox1.Location = new
System.Drawing.Point(0, 0);
        this.pictureBox1.Name = "pictureBox1";
        this.pictureBox1.Size = new
System.Drawing.Size(368, 104);
        this.pictureBox1.TabIndex = 0;
        this.pictureBox1.TabStop = false;
        //
        // label1
        //
        this.label1.Font = new
System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.label1.Location = new
System.Drawing.Point(0, 112);
        this.label1.Name = "label1";
        this.label1.Size = new
System.Drawing.Size(144, 24);
        this.label1.TabIndex = 1;
        this.label1.Text = "Destination Number :";

```

```

//
// txt_destination_numbers
//
this.txt_destination_numbers.Location =
new System.Drawing.Point(0, 136);
this.txt_destination_numbers.Name =
"txt_destination_numbers";
this.txt_destination_numbers.Size = new
System.Drawing.Size(224, 20);
this.txt_destination_numbers.TabIndex = 2;
this.txt_destination_numbers.Text = "";
//
// label2
//
this.label2.Font = new
System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.label2.Location = new
System.Drawing.Point(0, 168);
this.label2.Name = "label2";
this.label2.Size = new
System.Drawing.Size(72, 16);
this.label2.TabIndex = 3;
this.label2.Text = "Message :";
//
// txt_message
//
this.txt_message.Location = new
System.Drawing.Point(0, 192);
this.txt_message.Multiline = true;
this.txt_message.Name = "txt_message";
this.txt_message.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
this.txt_message.Size = new
System.Drawing.Size(224, 112);
this.txt_message.TabIndex = 4;
this.txt_message.Text = "";
this.txt_message.TextChanged += new
System.EventHandler(this.textBox1_TextChanged);
//
// txt_text_remaining
//
this.txt_text_remaining.BackColor =
System.Drawing.SystemColors.Control;
this.txt_text_remaining.Enabled = false;
this.txt_text_remaining.Location = new
System.Drawing.Point(224, 288);
this.txt_text_remaining.Name =
"txt_text_remaining";
this.txt_text_remaining.Size = new
System.Drawing.Size(40, 20);
this.txt_text_remaining.TabIndex = 5;

```

```

        this.txt_text_remaining.Text = "160";
        //
        // btnSendMessage
        //
        this.btnSendMessage.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.btnSendMessage.Location = new
System.Drawing.Point(144, 320);
        this.btnSendMessage.Name =
"btnSendMessage";
        this.btnSendMessage.Size = new
System.Drawing.Size(80, 24);
        this.btnSendMessage.TabIndex = 16;
        this.btnSendMessage.Text = "Send";
        this.btnSendMessage.Click += new
System.EventHandler(this.btnSendMessage_Click);
        //
        // BtnClear
        //
        this.BtnClear.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.BtnClear.Location = new
System.Drawing.Point(56, 320);
        this.BtnClear.Name = "BtnClear";
        this.BtnClear.Size = new
System.Drawing.Size(80, 24);
        this.BtnClear.TabIndex = 17;
        this.BtnClear.Text = "Clear";
        this.BtnClear.Click += new
System.EventHandler(this.BtnClear_Click);
        //
        // txtOutput
        //
        this.txtOutput.Location = new
System.Drawing.Point(0, 352);
        this.txtOutput.Multiline = true;
        this.txtOutput.Name = "txtOutput";
        this.txtOutput.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
        this.txtOutput.Size = new
System.Drawing.Size(504, 120);
        this.txtOutput.TabIndex = 56;
        this.txtOutput.Text = "";
        //
        // chkUnicode
        //
        this.chkUnicode.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.chkUnicode.Location = new
System.Drawing.Point(16, 56);
        this.chkUnicode.Name = "chkUnicode";
        this.chkUnicode.Size = new
System.Drawing.Size(208, 24);

```

```

        this.chkUnicode.TabIndex = 58;
        this.chkUnicode.Text = "Send as Unicode
(UCS2) ";
        //
        // chkAlert
        //
        this.chkAlert.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.chkAlert.Location = new
System.Drawing.Point(16, 24);
        this.chkAlert.Name = "chkAlert";
        this.chkAlert.Size = new
System.Drawing.Size(176, 24);
        this.chkAlert.TabIndex = 57;
        this.chkAlert.Text = "Request immediate
display (alert) ";
        //
        // groupBox1
        //

        this.groupBox1.Controls.Add(this.chkUnicode);

        this.groupBox1.Controls.Add(this.chkAlert);

        this.groupBox1.Controls.Add(this.txtSendTimes);

        this.groupBox1.Controls.Add(this.chkMultipleTimes);
        this.groupBox1.Controls.Add(this.label19);
        this.groupBox1.Location = new
System.Drawing.Point(240, 112);
        this.groupBox1.Name = "groupBox1";
        this.groupBox1.Size = new
System.Drawing.Size(264, 128);
        this.groupBox1.TabIndex = 59;
        this.groupBox1.TabStop = false;
        this.groupBox1.Text = "Option";
        //
        // txtSendTimes
        //
        this.txtSendTimes.Location = new
System.Drawing.Point(120, 88);
        this.txtSendTimes.Name = "txtSendTimes";
        this.txtSendTimes.Size = new
System.Drawing.Size(48, 20);
        this.txtSendTimes.TabIndex = 61;
        this.txtSendTimes.Text = "1";
        //
        // chkMultipleTimes
        //
        this.chkMultipleTimes.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.chkMultipleTimes.Location = new
System.Drawing.Point(16, 88);

```

```

        this.chkMultipleTimes.Name =
"chkMultipleTimes";
        this.chkMultipleTimes.TabIndex = 60;
        this.chkMultipleTimes.Text = "Send
message";

        //
        // label19
        //
        this.label19.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.label19.Location = new
System.Drawing.Point(176, 88);
        this.label19.Name = "label19";
        this.label19.Size = new
System.Drawing.Size(72, 23);
        this.label19.TabIndex = 62;
        this.label19.Text = "times";
        this.label19.TextAlign =
System.Drawing.ContentAlignment.MiddleLeft;
        //
        // Send
        //
        this.AutoScaleBaseSize = new
System.Drawing.Size(5, 13);
        this.ClientSize = new
System.Drawing.Size(504, 478);
        this.Controls.Add(this.groupBox1);
        this.Controls.Add(this.txtOutput);
        this.Controls.Add(this.BtnClear);
        this.Controls.Add(this.btnSendMessage);

        this.Controls.Add(this.txt_text_remaining);
        this.Controls.Add(this.txt_message);
        this.Controls.Add(this.label2);

        this.Controls.Add(this.txt_destination_numbers);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.pictureBox1);
        this.Name = "Send";
        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
        this.Text = "Send SMS";
        this.Load += new
System.EventHandler(this.Send_Load);
        this.groupBox1.ResumeLayout(false);
        this.ResumeLayout(false);

    }
    #endregion

    private void textBox1_TextChanged(object sender,
System.EventArgs e)
    {

```

```

        int
        remaining=int.Parse(txt_text_remaining.Text.Trim());
        remaining-=1;

        txt_text_remaining.Text=remaining.ToString();
    }

    private void BtnClear_Click(object sender,
System.EventArgs e)
    {
        txt_message.Text="";
        txt_message.Focus();
    }

    private void btnSendMessage_Click(object sender,
System.EventArgs e)
    {
        Cursor.Current = Cursors.WaitCursor;

        try
        {
            // Send an SMS message
            SmsSubmitPdu pdu;
            bool alert = chkAlert.Checked;
            bool unicode = chkUnicode.Checked;

            if (!alert && !unicode)
            {
                // The straightforward version
                pdu = new
SmsSubmitPdu(txt_message.Text,
txt_destination_numbers.Text, ""); // "" indicate SMSC No
            }
            else
            {
                // The extended version with dcs
                byte dcs;
                if (!alert && unicode)
                    dcs =
DataCodingScheme.NoClass_16Bit;
                else if (alert && !unicode)
                    dcs =
DataCodingScheme.Class0_7Bit;
                else if (alert && unicode)
                    dcs =
DataCodingScheme.Class0_16Bit;
                else
                    dcs = DataCodingScheme.NoClass_7Bit; // should
never occur here

                pdu = new SmsSubmitPdu(txt_message.Text,
txt_destination_numbers.Text, "", dcs);
            }
        }
    }

```



```

        // Send the same message multiple
times if this is set
        int times = chkMultipleTimes.Checked
? int.Parse(txtSendTimes.Text) : 1;

        // Send the message the specified
number of times
        for (int i=0;i<times;i++)
        {

            CommSetting.comm.SendMessage(pdu);
            Output("Message {0} of {1}
sent.", i+1, times);
            Output("");
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

        Cursor.Current = Cursors.Default;
    }

    private void Output(string text)
    {
        if (this.txtOutput.InvokeRequired)
        {
            SetTextCallback stc = new
SetTextCallback(Output);
            this.Invoke(stc, new object[] { text
});
        }
        else
        {
            txtOutput.AppendText(text);
            txtOutput.AppendText("\r\n");
        }
    }

    private void Send_Load(object sender,
System.EventArgs e)
    {
        chkMultipleTimes.Checked=true;
    }

    private void Output(string text, params
object[] args)
    {
        string msg = string.Format(text, args);

```

```
        Output(msg);  
    }  
}  
}
```

Database Code

The application is developed using C#. Code for database is given below.

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
using System.Data.SqlClient;  
using System.Configuration;  
using System.IO;  
using System.Diagnostics;  
  
namespace test  
{  
    public partial class Form1 : Form  
    {  
        string a = null;  
        string b = null;  
        string c = null;
```

```
public Form1()
{
    InitializeComponent();
}

private void button5_Click(object sender,
EventArgs e)
{
    OpenFileDialog fdlg = new OpenFileDialog();
    fdlg.Title = "C# Corner Open File Dialog";
    fdlg.InitialDirectory = @"c:\";
    fdlg.Filter = "All files (*.*)|*.*|All files
(*.*)|*.*";
    fdlg.FilterIndex = 2;
    fdlg.RestoreDirectory = true;
    if (fdlg.ShowDialog() == DialogResult.OK)
    {
        // label2.Text =
Path.GetDirectoryName(fdlg.FileName);

    }

    const int BYTES_TO_READ = sizeof(Int64);
    byte[] buff2 = new byte[BYTES_TO_READ];
    string filePath = (@fdlg.FileName);
    string filename2 =
Path.GetFileName(filePath);
    FileStream fs2 = new FileStream(filePath,
FileMode.Open, FileAccess.Read);
    BinaryReader br2 = new BinaryReader(fs2);
```

```

        Byte[] bytes =
br2.ReadBytes((Int32)fs2.Length);

        long numBytes = new
FileInfo(filePath).Length;

        br2.Close();

        fs2.Close();

        string strQuery = "insert into
Information(FName, LName,PhoneNo,FileName,Data) values
(@FName,@LName,@PhoneNo,@FileName,@Data)";

        SqlCommand cmd = new SqlCommand(strQuery);

        cmd.Parameters.Add("@Fname",
SqlDbType.VarChar).Value = textBox2.Text;

        cmd.Parameters.Add("@Lname",
SqlDbType.VarChar).Value = textBox3.Text;

        cmd.Parameters.Add("@PhoneNo",
SqlDbType.VarChar).Value = textBox4.Text;

        cmd.Parameters.Add("@FileName",
SqlDbType.VarChar).Value = textBox5.Text;

        cmd.Parameters.Add("@Data",
SqlDbType.Binary).Value = bytes;

        InsertUpdateData(cmd);

    }

    private Boolean InsertUpdateData(SqlCommand cmd)
    {

        SqlConnection con = new
SqlConnection(@"Server=(LocalDB)\v11.0; Integrated
Security=true ;AttachDbFileName=F:\Project\adnan.mdf");

        cmd.CommandType = CommandType.Text;

        cmd.Connection = con;

```

```
try
{
    con.Open();
    MessageBox.Show("Done");
    cmd.ExecuteNonQuery();
    return true;
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString());
    return false;
}
finally
{
    con.Close();
    con.Dispose();
}
}

private void button6_Click(object sender,
EventArgs e)
{
    int i = 0;
    int b = 0;

    string strQuery = "Select * from Information
where FileName='" + textBox1.Text + "'";

    SqlCommand cmd = new SqlCommand(strQuery);

    SqlConnection con = new
SqlConnection(@"Server=(LocalDB)\v11.0; Integrated
Security=true ;AttachDbFileName=F:\Project\adnan.mdf");
```

```
cmd.CommandType = CommandType.Text;
cmd.Connection = con;

byte[] buff2 = new byte[5000000];
string filePath = (@textBox6.Text);
string filename2 =
Path.GetFileName(filePath);

FileStream fs2 = new FileStream(filePath,
FileMode.Open, FileAccess.Read);

BinaryReader br2 = new BinaryReader(fs2);

Byte[] bytes =
br2.ReadBytes((Int32)fs2.Length);

Byte[] a = br2.ReadBytes((Int32)fs2.Length);

long numBytes = new
FileInfo(filePath).Length;

buff2 = br2.ReadBytes((int)numBytes);

try
{
    con.Open();

    SqlDataReader dr =
cmd.ExecuteReader();

    while (dr.Read())
    {

        a = (Byte[])dr[5];

        MessageBox.Show("1");

    }

}

catch (Exception ex)
{
```

```
        MessageBox.Show(ex.ToString());
    }
    finally
    {
        con.Close();
        con.Dispose();
    }

    var md5 = new
System.Security.Cryptography.MD5Cng();

    var md5File1 = md5.ComputeHash(a);
    var md5File2 = md5.ComputeHash(bytes);
    for ( i = 0; i < md5File2.Length; ++i)
    {
        if (md5File1[i] == md5File2[i])
        {
            MessageBox.Show("Match");

            Process.Start("C:\\Users\\Adil
Khan\\Desktop\\SMS\\SMS\\bin\\Debug\\SMS.exe");

            check();

            break;
        }
        else
        {
            MessageBox.Show("No Match");

            break;
        } // label2.Text = "no match";
    }
}
```

```
    }  
  
    private void check()  
    {  
  
        string strQuery = "Select * from Information  
where FileName='" + textBox1.Text + "'";  
  
        SqlCommand cmd = new SqlCommand(strQuery);  
  
        SqlConnection con = new  
SqlConnection(@"Server=(LocalDB)\v11.0; Integrated  
Security=true ;AttachDbFileName=F:\Project\adnan.mdf");  
  
        cmd.CommandType = CommandType.Text;  
  
        cmd.Connection = con;  
  
        try  
        {  
  
            con.Open();  
  
            SqlDataReader dr = cmd.ExecuteReader();  
  
            while (dr.Read())  
            {  
  
                a = dr[1].ToString();  
  
                b = dr[2].ToString();  
  
                c = dr[3].ToString();  
  
            }  
  
        }  
  
        catch (Exception ex)  
        {  
  
            MessageBox.Show(ex.ToString());  
  
        }  
  
        finally  
        {
```



```
        con.Close();
        con.Dispose();
    }

    Form2 frm = new Form2();
    frm._textBox = _textBox1;
    frm._textBox1 = _textBox2;
    frm._textBox2 = _textBox3;
    frm.Show();
}

public string _textBox1
{
    get { return a; }
}

public string _textBox2
{
    get { return b; }
}

public string _textBox3
{
    get { return c; }
}

private void textBox1_TextChanged(object sender,
EventArgs e)
{
}

private void button1_Click(object sender,
EventArgs e)
```

```
{

    OpenFileDialog fdlg = new OpenFileDialog();
    fdlg.Title = "C# Corner Open File Dialog";
    fdlg.InitialDirectory = @"c:\";
    fdlg.Filter = "All files (*.*)|*.*|All files
(*.*)|*.*";

    fdlg.FilterIndex = 2;
    fdlg.RestoreDirectory = true;
    if (fdlg.ShowDialog() == DialogResult.OK)
    {

        textBox6.Text
=Path.GetDirectoryName(fdlg.FileName) + "\\\" +
Path.GetFileName(fdlg.FileName);

    }

}

private void textBox6_TextChanged(object sender,
EventArgs e)
{

}

private void button2_Click(object sender,
EventArgs e)
{

}

}
```

References

- [1] Zhang Yongqiang and Liu Ji ,*The design of wireless fingerprint attendance system, Proceedings of ICCT '06, International Conference on Communication Technology, 2006.*
- [2] Younhee Gil, *Access Control System with high level security using fingerprints,IEEE the 32nd Applied Imagery Pattern Recognition Workshop (AIPR '03)*
- [3] Jain, A.K., Hong, L., and Bolle, R.(1997), “*On-Line Fingerprint Verification,*” *IEEE Trans. On Pattern Anal and Machine Intell, 19(4), pp. 302-314.*
- [4] D.Maio and D. Maltoni. *Direct gray-scale minutiae detection in fingerprints. IEEE Trans. Pattern Anal. And Machine Intell., 19(1):27-40, 1997.*
- [5] Lee, C.J., and Wang, S.D.: *Fingerprint feature extration using Gabor filters, Electron. Lett., 1999, 35, (4), pp.288-290.*
- [6] L. Hong, Y. Wan and A.K. Jain, "Fingerprint Image Enhancement: Algorithms and Performance Evaluation", *IEEE Transactions on PAMI ,Vol. 20, No. 8, pp.777-789, August 1998.*
- [7] SPRA894A, *Texas Instruments, DSP for Smart Biometric Solutions*
- [8] *User Manual, DWA-510*
- [9] SPRAA23, *Texas Instruments, FADT2 Quick Start Guide*
- [10] TMS320C6713 *DSK Technical Reference, (506735-0001 Rev. B)*
- [11] FVC2002. <http://bias.csr.unibo.it/fvc2002/>
- [12] *Fingerprint Recognition System by Luigi Rosa, (http://www.mathworks.it/matlabcentral/fileexchange/4239)*
- [13] Shlomo Greenberg, Mayer Aladjem, Daniel Kogan and Itshak Dimitrov, *Fingerprint Image Enhancement using Filtering Techniques*