CHAPTER 1

INTRODUCTION

In spite of numerous advantages of biometrics-based personal authentication systems over traditional security systems based on token or knowledge, they are vulnerable to attacks that can decrease their security considerably. Biometrics-based personal authentication system that use physiological (fingerprint, face) or behavioral (speech, handwriting) traits are becoming increasingly popular, compared to traditional systems that are based on tokens (key) or knowledge (password).

Fingerprint-based identification is one of the most important biometric technologies which have drawn a substantial amount of attention recently. Fingerprint technology is so common in personal identification that it has been well established. Each human has unique owns fingerprint, even the twin have different fingerprint. So fingerprint recognition is useful in security law application. The electronic lock using fingerprint recognition involves a process of verifying the user's identity by using fingerprint recognition as a key to the electronic lock. This work highlights the development of fingerprint recognition system using ARDUINO 1.6.3. to recognize the input fingerprint image from the stored samples in bmp, tif; tiff; jpg; jpeg; gif file type. Then the information of the recognized fingerprint image will be store in database for verification authorized user. These fingerprint recognition systems are based on the hypothesis that the human fingerprint is unique.

It is important to validate the individuality of fingerprint in order to use the fingerprint image for security related system. In real application, the fingerprint data is recorded by using USB fingerprint scanner and then sent to a recognizer that will check the similarity of the user's fingerprint.

1.1 OVERVIEW

The main aim of this project is to develop a secure locking system based on fingerprint scanning. In this project, microcontroller accompanied with an interface circuit has been used for opening and closing lock based on finger print which is stored in microcontroller itself so that only authorized person will access the security lock.

1.2 PROBLEM STATEMENT

- First step towards security was Lock and key system. Security protocol followed in this system was "Single key for a single lock". Initially, this system was considered to provide utmost security but soon it proved by the fact that multiple keys of a single lock can easily made.
- Another step towards security was security passwords that were used as an authenticating tool. In this a multiple set of numbers or alphabets is stored in the database for the purpose of authentication. This has a disadvantage that password can be hacked or acquired by unauthorized user by continuously trying all the possible combinations.

1.3 OBJECTIVES

The main aim of this project is to develop a secure locking system based on fingerprint scanning. In this project, microcontroller accompanied with an interface circuit has been used for opening and closing lock based on finger print which is stored in microcontroller itself so that only authorized person will access the security lock.

1.4 SCOPE OF THE PROJECT

In future, alarm will be introduced. When intruder tries to break the door, the vibration is sensed by sensor which makes an alarm. This will inform the neighbors about intruders and this will help to take further action to prevent intruder from entering. This will also send a text message to the user if a wrong fingerprint is sensed more than 5 times. Further it will also be implemented in vehicles which will reduce the motor vehicle theft.

CHAPTER 2
REVIEW OF LITERATURE

The use of fingerprint for identification has been employed in law enforcement for about a century. A fingerprint locker system using microcontroller uses fingerprint recognition system as a process of verifying the fingerprint image to open the electronic lock. This research highlights the development of fingerprint verification system using Arduino 1.6.3. Verification is completed by comparing the data of authorized fingerprint image with incoming fingerprint image. The incoming fingerprint image will first go through the extraction and filtering processes through which the information about it is obtained. Then the information of incoming fingerprint image will undergo the comparison process to compare it with authorized fingerprint image. In this work, the fingerprint module was trained to learn and identify whether the incoming fingerprint image is genuine or forgery.

2.1 PRELIMINARY INVESTIGATION

2.1.1 CURRENT SYSTEM

- i-Touchless Bio-Matic Fingerprint Door Lock

This fingerprint door lock can store up to 150 finger prints at a time and can add or delete users which is performed directly on the lock's pin pad.

2.1.2 LIMITATION OF CURRENT SYSTEM

It is very costly and power cuts, battery discharge are common issues associated with these door locking systems.

2.2 REQUIREMENT IDENTIFICATION AND ANALYSIS FOR PROJECT

Fingerprint technology is so common in personal identification that it has been well established. Each human has unique owns fingerprint, even the twin have different fingerprint. So fingerprint recognition is useful in security law application. The electronic lock using fingerprint recognition involves a process of verifying the user's identity by using fingerprint recognition as a key to the electronic lock. This work highlights the development of fingerprint recognition system using ARDUINO 1.6.3. to recognize the input fingerprint image from the stored samples in bmp, tif; tiff; jpg; jpeg; gif file type. Then the information of the recognized fingerprint image will be store in database for verification authorized user. These fingerprint recognition systems are based on the hypothesis that the human fingerprint is unique. It is important to validate the individuality of fingerprint in order to use the fingerprint image for security related system. In real application, the fingerprint data is recorded by using USB fingerprint scanner and then sent to a recognizer that will check the similarity of the user's fingerprint.

CHAPTER 3
PROPOSED SYSTEM

3.1 THE PROPOSAL

The proposed system grants access to authorized personnel by virtue of recognizing their unique fingerprints .Basically the system operates by scanning and converting the fingerprint data into a numerical template. Once you place your finger onto the scanner for the first time the conversion into numerical data takes place, and the fingerprint template is saved. This process is then repeated every time you want to grant someone access. The next time someone places his/her finger on the sensor, it matches the data obtained through the finger with the pre-saved values. If a match is found, access is granted and the door opens. On the other hand, if it's someone else trying to get through, access is not allowed and the door remains locked. The process is very quick and is completed in a fraction of a second.

3.2 FEASIBILITY STUDY

A feasibility study is an analysis of how successfully a system can be implemented, accounting for factors that affect it such as economic, technical and operational factors to determine its potential positive and negative outcomes before investing a considerable amount of time and money into it.

3.2.1 TECHNICAL FEASIBILITY

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the available hardware and software and how it meets the need of the proposed system. Analyzes the technical skills and capabilities of the software development team members

- Determines whether the relevant technology is stable and established

- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware.

3.2.2 ECONOMICAL FEASIBILITY

The purpose of the economic feasibility assessment is to determine the positive economic benefits of the organization that the proposed system will provide. It includes qualification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization

- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis)

- Cost of hardware, software, development team, and training.

3.2.3 OPERATIONAL FEASIBILITY

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture, and existing processes. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority

- Determines whether the solution suggested by the software development team is acceptable

- Analyzes whether users will adapt to new software.

3.3 DESIGN REPRESENTATION

3.3.1 DATA FLOW DIAGRAM

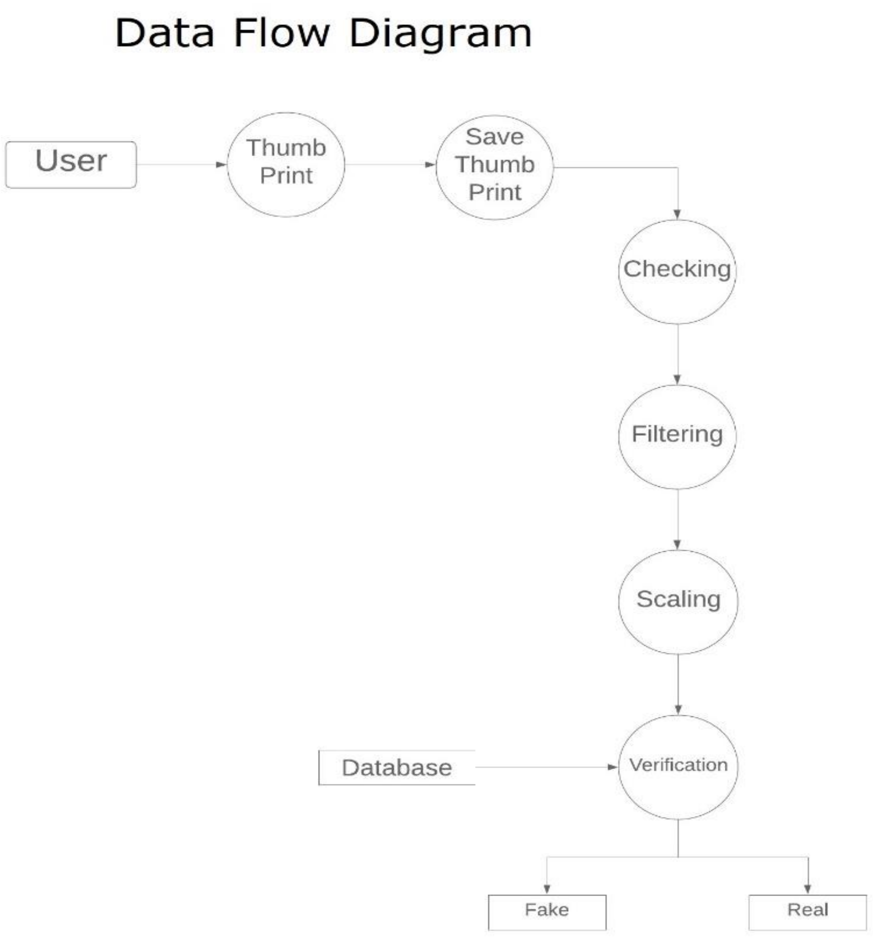


FIGURE 1

3.3.2 USE CASE DIAGRAM

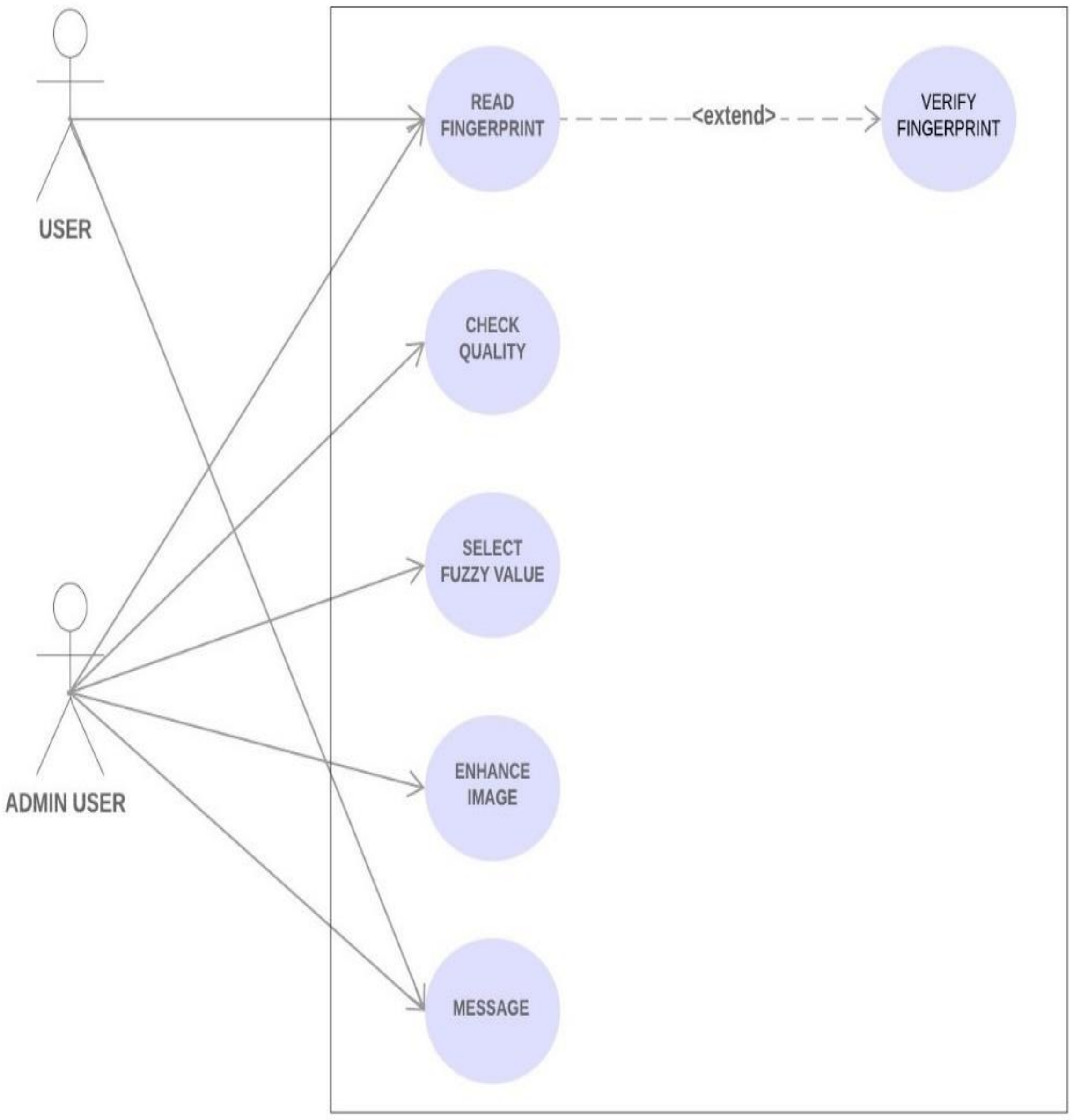
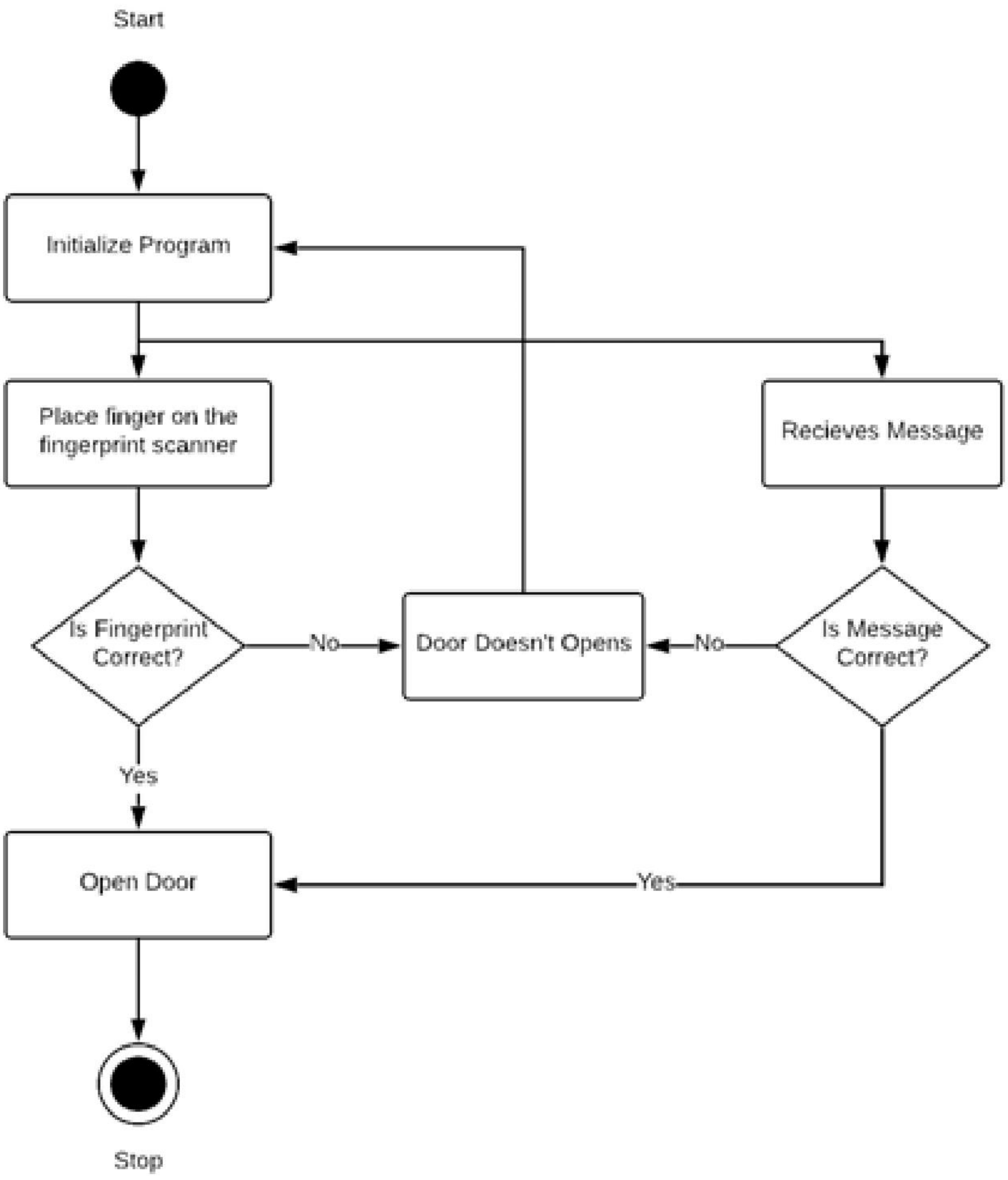


FIGURE 2

3.3.3 ACTIVITY DIAGRAM



3.3.4 SEQUENCE DIAGRAM

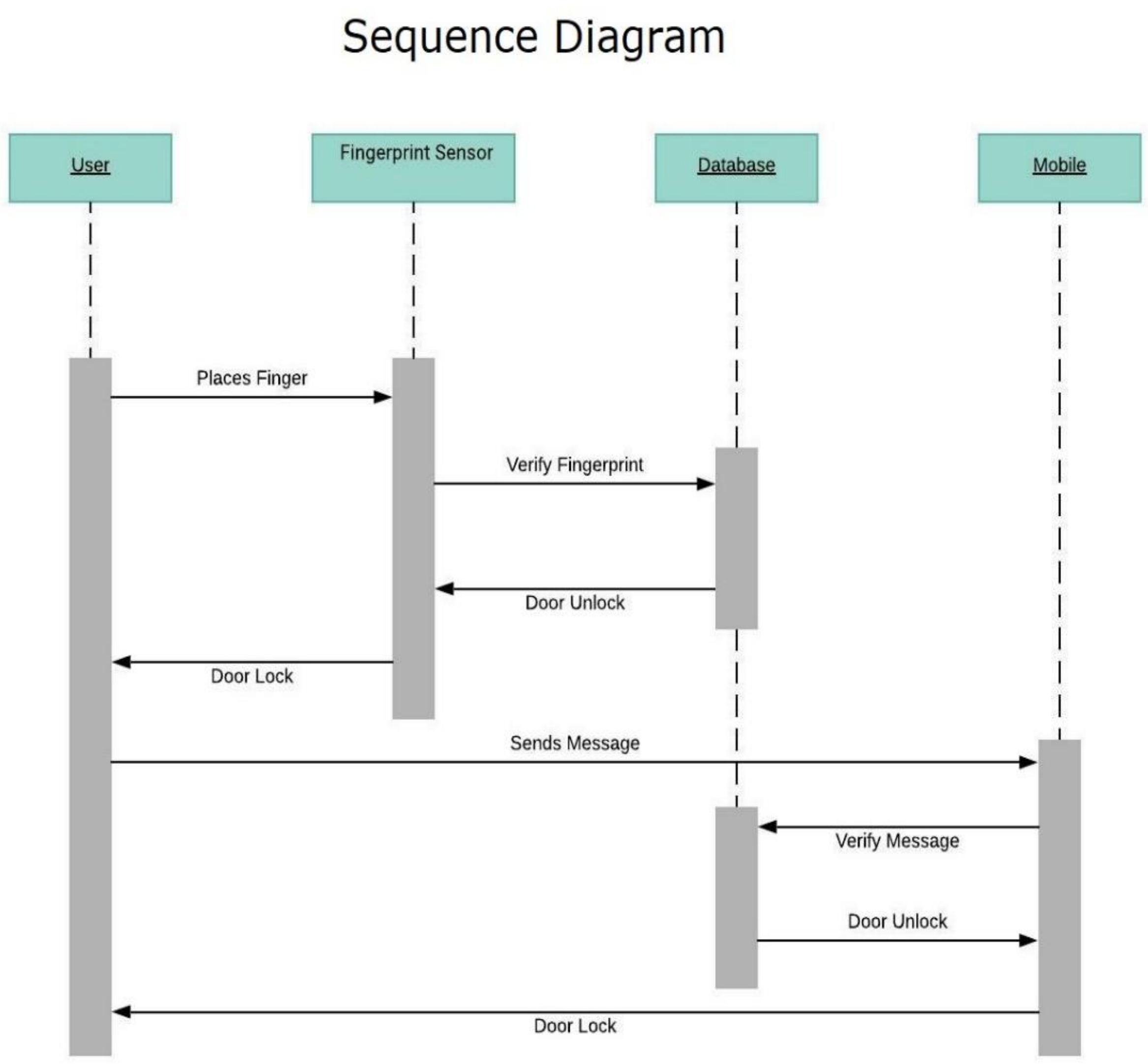


FIGURE 3

3.4 DEPLOYMENT REQUIREMENTS

There are various requirements (hardware, software and services) to successfully deploy the system. These are mentioned below :

3.4.1 HARDWARE

- ADAFRUIT FINGERPRINT SENSOR

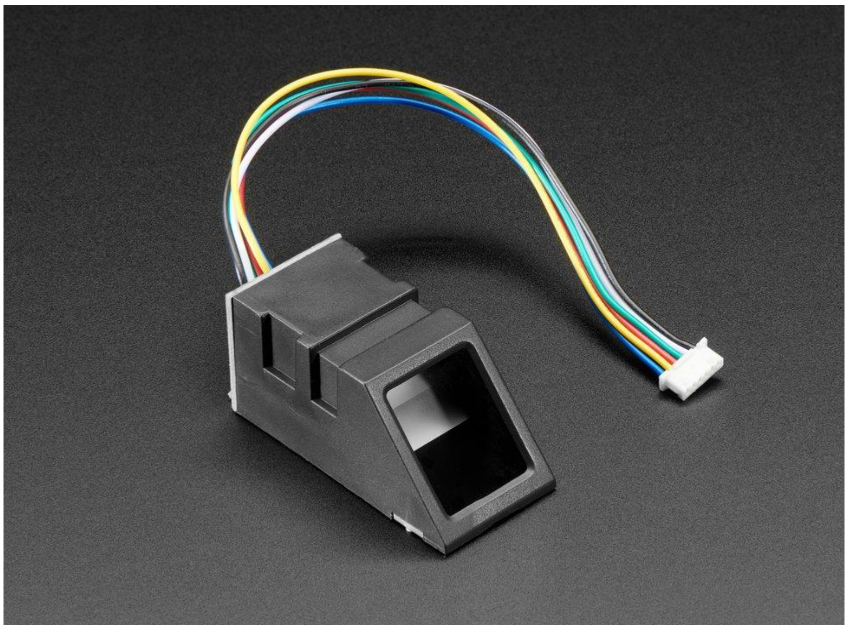


FIGURE 4

- SOLENOID 12-VOLT LOCK

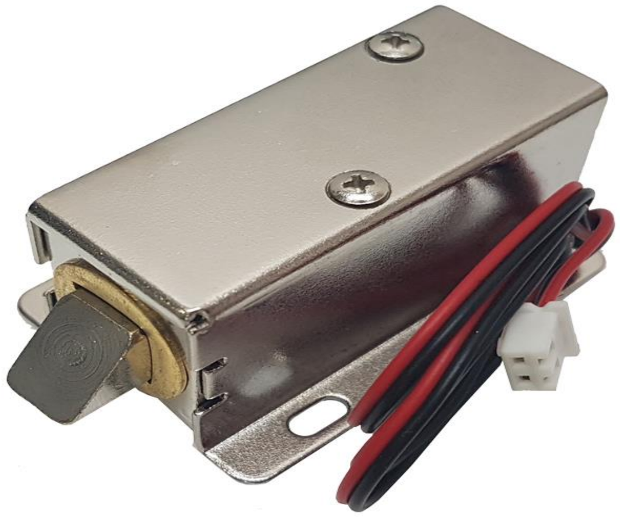


FIGURE 5

- ARDUINO UNO r3

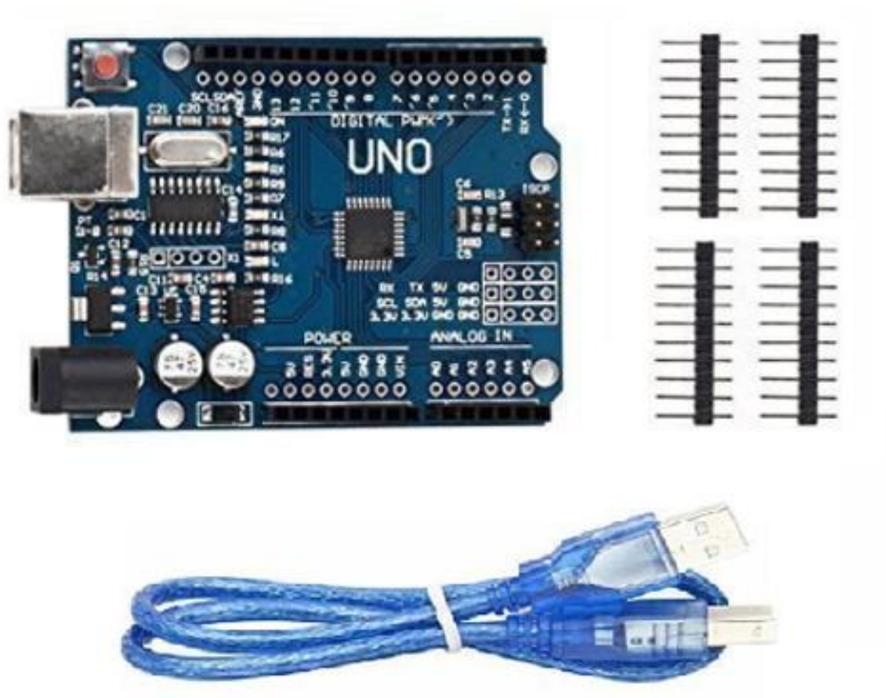


FIGURE 6

- JUMPER WIRES

FIGURE 7

- 12V ADAPTER

FIGURE 8

- RELAY

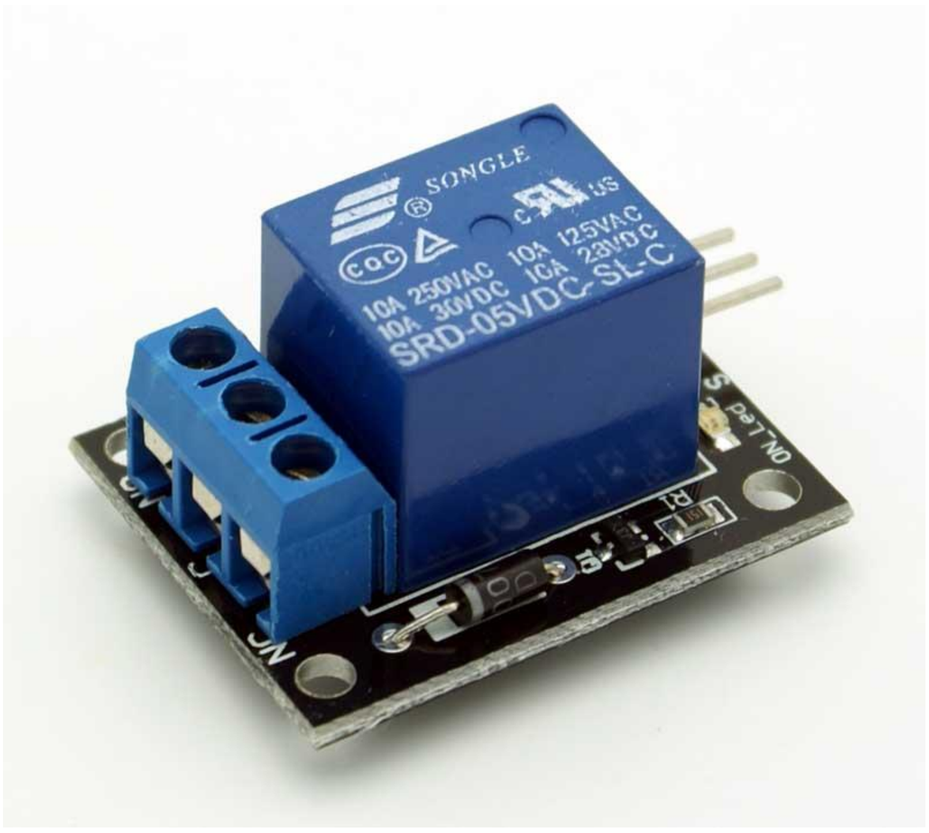


FIGURE 9

CHAPTER 4

IMPLEMENTATION

4.1 TECHNQUE USED

4.1.1 FINGERPRINT RECOGNITION

Fingerprint recognition refers to the automated method of identifying or confirming the identity of an individual based on the comparison of two fingerprints. Fingerprint recognition is one of the most well-known biometrics, and it is by far the most used biometric solution for authentication on computerized systems. The reasons for fingerprint recognition being so popular are the ease of acquisition, established use and acceptance when compared to other biometrics, and the fact that there are numerous (ten) sources of this biometric on each individual.

FIGURE 10

The three basic patterns of fingerprint ridges are the arch, the loop, and the whorl. An arch is a pattern where the ridge enters one side of the finger, then rises in the center forming an arch, and exits on the other side of the finger. With a loop the ridge enters one side of the finger, then forms a curve, and exits on the same side of the finger from which it entered. Loops are the most common pattern in fingerprints. Finally a whorl is the pattern you have when ridges form circularly around a central point.

4.1.2 CAPACITIVE READERS

Capacitive readers, also referred to as CMOS readers, do not read the fingerprint using light. Instead a CMOS reader uses capacitors and thus electrical current to form an image of the fingerprint. CMOS readers are more expensive than optical readers, although they still come relatively cheap with prices starting well below 100 euro's.
An important advantage of capacitive readers over optical readers is that a capacitive reader requires a real fingerprint shape rather than only a visual image. This makes CMOS readers harder to trick.

4.2 APPLICATION USED

Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiringproject, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

4.3 PROJECT PHOTOGRAPGH

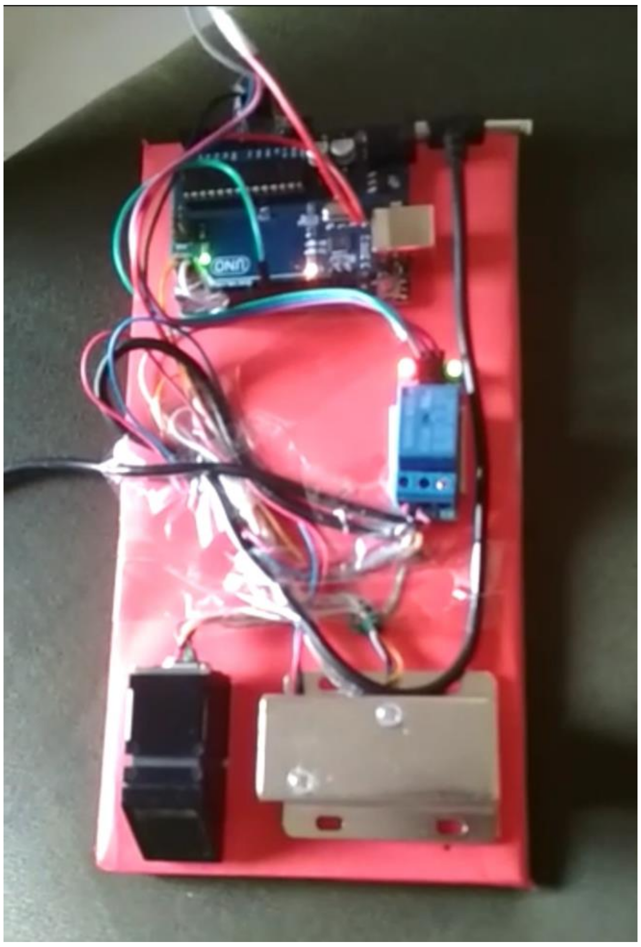


FIGURE 11

4.4 TESTING

Testing is the process of evaluation of a system to detect differences between given input and expected output and also to assess the feature of the system. Testing assesses the quality of the product. It is a process that is done during the development process. .

4.4.1 STRATEGY USED

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

The texting method used here is Black Box Testing. It is carried out to test functionality of the program. It is also called ‘Behavioral’ testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.

CHAPTER 5
CONCLUSION

5.1 CONCLUSION

Fingerprint identification enhances the security of a room and makes it possible only for some selected people to use the room. Thus by implementing this relatively cheap and easily available system on a room, one can ensure much greater security and exclusivity than that offered by a conventional lock and key. It can be deduced that the use of biometric security systems offers a much better and foolproof means of restricting the use of room by unauthorized users. The developed prototype serves as an impetus to drive future research, geared towards developing a more robust and embedded real-time fingerprint based automatic door lock systems in rooms.

5.2 LIMITATIONS OF THE WORK

- Sometimes the scanner may refuse to recognize your finger in case of scarring or cuts or abrasions.

- If someone does manage to create a copy/mold of your hand then the door can be opened.

5.3 SUGGESTION AND RECOMMENDATION FOR FUTURE WORK

- In future, alarm will be introduced. When intruder tries to break the door, the vibration is sensed by sensor which makes an alarm. This will inform the neighbors about intruders and this will help to take further action to prevent intruder from entering.
- This will also send a text message to the user if a wrong fingerprint is sensed more than 5 times.
- Further it will also be implemented in vehicles which will reduce the motor vehicle theft.

BIBLIOGRAPHY

file:///C:/Users/Owner/AppData/Local/Temp/1019-Article%20Text-1774-1-10-20171231.pdf

https://www.ijareeie.com/upload/2017/april/45_Fingerprint.pdf

https://www.iosrjen.org/Papers/vol7_issue4/Version-1/C0704011319.pdf

https://pdfs.semanticscholar.org/e825/c394931cdd6f66c56c5171e96737bd0325a4.pdf

SOURCE CODE

R307 FINGERPRINT MODULE

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <Servo.h> //Add servo library

int getFingerprintIDez();

Servo servo1; //Define servo name / object

#define servoPin 9 //Define pin number to which servo motor is connected
#define durationTime 3000 //Define the time it remains in the open position of the door lock (miliseconds)
#define servoMin 0 //Open position
#define servoMax 90 // Closed position



SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);



void setup()
{
  while (!Serial);  // For Yun/Leo/Micro/Zero/...

  Serial.begin(9600);
  Serial.println("Adafruit finger detect test");

  servo1.attach(servoPin); //Define pin number of the servo
```

```
 servo1.write(servoMax); //The position of the servo at the start of the
program

 // set the data rate for the sensor serial port
 finger.begin(57600);

 if (finger.verifyPassword()) {
  Serial.println("Found fingerprint sensor!");
 } else {
  Serial.println("Did not find fingerprint sensor :(");
  while (1);
 }
 Serial.println("Waiting for valid finger...");
}

void loop()                    // run over and over again
{
 getFingerprintIDez();
 delay(50);            //don't ned to run this at full speed.
}

uint8_t getFingerprintID() {
 uint8_t p = finger.getImage();
 switch (p) {
  case FINGERPRINT_OK:
   Serial.println("Image taken");
   break;
  case FINGERPRINT_NOFINGER:
   Serial.println("No finger detected");
   return p;
  case FINGERPRINT_PACKETRECIEVEERR:
   Serial.println("Communication error");
   return p;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK success!

p = finger.image2Tz();
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
```

```
  p = finger.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
    Serial.println("Found a print match!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    Serial.println("Did not find a match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
}

// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  servo1.write(servoMin); //If the fingerprint is correct open the door lock
  delay(durationTime); //Keep the lock open for the defined duration
  servo1.write(servoMax); //take the lock OFF again
  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
```

ENROLLING A FINGERPRINT

```
#include <Adafruit_Fingerprint.h>

// On Leonardo/Micro or others with hardware serial, use those! #0 is green wire, #1 is white
// uncomment this line:
// #define mySerial Serial1

// For UNO and others without hardware serial, we must use software serial...
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
// comment these two lines if using hardware serial
SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
```

```
 delay(100);
 Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

 // set the data rate for the sensor serial port
 finger.begin(57600);

 if (finger.verifyPassword()) {
   Serial.println("Found fingerprint sensor!");
 } else {
   Serial.println("Did not find fingerprint sensor :(");
   while (1) { delay(1); }
 }
}

uint8_t readnumber(void) {
 uint8_t num = 0;

 while (num == 0) {
   while (! Serial.available());
   num = Serial.parseInt();
 }
 return num;
}

void loop()                    // run over and over again
{
 Serial.println("Ready to enroll a fingerprint!");
 Serial.println("Please type in the ID # (from 1 to 127) you want to save
this finger as...");
 id = readnumber();
 if (id == 0) {// ID #0 not allowed, try again!
    return;
 }
```

```
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!
```

```
p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
Serial.print("Creating model for #");  Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}

Serial.print("ID "); Serial.println(id);
p = finger.storeModel(id);
if (p == FINGERPRINT_OK) {
  Serial.println("Stored!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
```

```
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}
```

TABLE OF CONTENTS

TABLE OF FIGURES