LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title
CC5068NI– Cloud Computing & IoT

Fingerprint Door Unlock

Assessment Type
40% System Development Report

Semester
2022 Spring

Group members

London Met ID	Student Name
20049450	Pawan Raj Joshi
20049211	Robin Sah
20049157	Jitesh Sah
19031460	Sonam Sherpa
20049250	Sujen Shrestha

Assignment Due Date: May 9, 2022
Assignment Submission Date: May 9, 2022
Word Count: 2118

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

We are very grateful to our class teacher Umesh Nepal as well as Sujil Maharjan and the module leader Sugat Man Shakya who gave us a chance to work on this project. We would like to thank them for giving us valuable suggestions and ideas. We would also like to thank our college for providing us with some of the necessary resources for the project. All in all, we would like to thank everyone involved in this project and helped us with their suggestions to make the project better. Finally, we would like to thank our every friend participating in this project for proactively working and contributing to this project in every situation.

Abstract

This is our first project of the IoT (Internet of Things) which provides a lot of knowledge about the circuit and the various type of component like Arduino, motor, transistors and sensors etc. This project mainly helps to make the world digitalize and provide the additional security to the system. The project is about creating a biometric system which uses fingerprint to unlock the door. By using such system, only the authorized people can open the lock and enter the door.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The Internet of Things (IoT) is a network of physical items called "things" that are integrated with sensors, software, and other technologies in order to communicate and exchange data with other devices and systems over the internet.

This is our first project of IoT (Internet of Things) which is about fingerprint door lock system. The main components used in this project are like Arduino, fingerprint sensor, linear solenoid, battery and a transistor. This project work as Anti-Theft system because it protects the room from unauthorized users. The fingerprint door lock has a self-explanatory name, it is a system that gives authorized people entry by identifying their fingerprints, which are an excellent form of human identification. Human fingerprints are extremely detailed and one-of-a-kind. Furthermore, copying another set of fingerprints, even faking and altering them, is tough. Because fingerprints are almost always unique, fingerprint door locking systems are ideal in terms of security. In comparison to a pair of keys, you can scarcely forget your fingerprints or even have to remember it to always carry it. (Park, 2022)

1.1 Current Scenario

Fingerprint door lock system is also known as Biometric door lock system. Now a days it is used in various sector like education, banking, and home. Mostly, it is used in banking sector to safe the money vault. Within a year now it is also used in home appliances to secure and for standardization. Any method by which a person may be uniquely recognized by assessing one or more differentiating biological attributes is referred to as biometric verification. Fingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves, DNA, and signatures are all unique identifiers. We will use fingerprints for biometric verification because they are one of the few things that are unique to each individual. Using fingerprints as the key to door locks can greatly reduce the risk of unauthorized people trespassing into our homes, shops, offices, and other places because duplication of such a key is impossible. Furthermore, because we do not need to carry keys if this system is utilized instead of standard locks, this method will not

generate difficulties such as key loss. So, with the help of Arduino, we'll try to build a system with characteristics that will improve security.

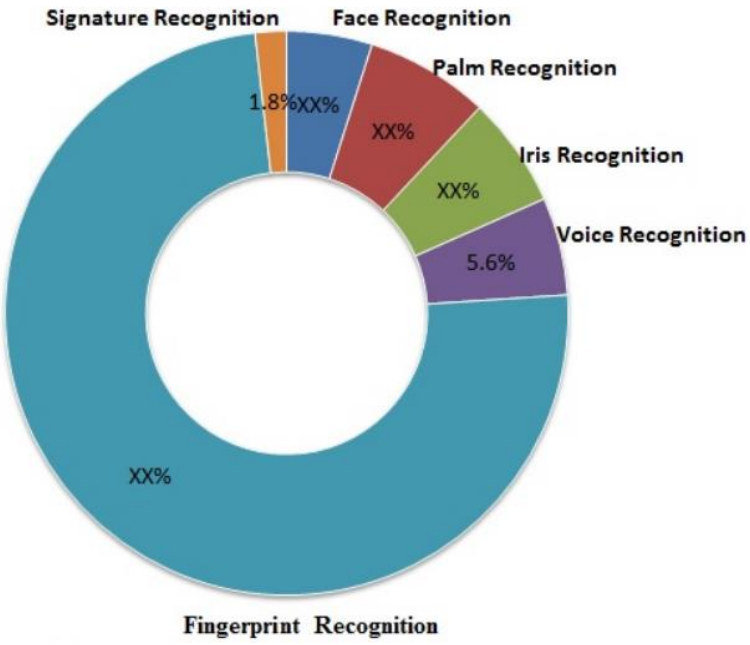


Figure 1: Biometric Statistics

According to the survey the fingerprint segment is anticipated to project a highest market share by crossing 63% in 2021 owing to the increasing demand for biometric locks in government and commercial sector. The fingerprint lock takes a processing time of 5-9 seconds and are majorly used in residential as well as commercial sectors owing to their key features such as reduced power requirements, smaller storage space and resistance to background lighting and temperature fluctuations. (Devices, 2021)

1.2 Problem statement

The traditional door locks can be easily unlocked with the help of pins, duplicate keys or by breaking the lock. This issue can be solved if there is an alternative way of locking the doors. For this purpose, we have designed a way of locking the doors using the biometric fingerprint system. By implementing this any unauthorized people cannot enter the door and it provides higher level of security as it maintains logs of people entering from the door.

1.3 Aims and Objectives

The main aim of this project is to provide the relevant and in-depth knowledge about the IOT (Internet of Things) and the providing additional layer to the security.

The Main objectives of this project which signifies the aims are given below: -

- To ensure that only authenticated people are allowed access to enter.
- To maintain a log of entries made through the device.
- To allow permission to only the required people.
- To add an additional layer of security to the system.

2. Background

2.1 System Overview

In today's world, homes, offices, stores and banks need excessive security measures for security reasons. To protect this area, the Smart-Lock system will be started. Numerous Innovative Smart Door Locks Created to lock and unlock the system. These types of locks feature fingerprints, RFID cards, pins, passwords or IOT by unlocking the system on your mobile phone. User uses. This type of lock system unlocks the system using pin number or fingerprint or RFID card to. This system does not have security pecking instructions to enhance security. To enhance security, users need to unlock the system, which is an intelligent door entry system with a fingerprint module. Both hardware and software techniques are used to design it.

This part gives a fast outline of every one of the parts used to make the fingerprint door lock/open framework. The framework’s spine is the Arduino UNO, a maximum expense, adaptable and simple to utilize programmable opensource microcontroller board that can likewise be utilized in an assortment of electronic and IOT projects (D. & Pappachan, 2019).

The project shown was powered by a 12V power supply. The Arduino microcontroller only needs 5V, but the electromagnet lock requires 12V. The Arduino UNO has a built-in 5V regulator, so you can use a common 12V power supply for the entire system. The brain of the circuit is the Arduino UNO board. It is based on 6 analog inputs, 32K flash memory, 16MHz crystal oscillator, USB connector, power jack, ICSP header, reset button and more.

The AS608 fingerprint sensor module has a UART interface that connects directly to the MCU or PC via a USB serial adapter. Users can save fingerprint data in module configuration mode for identification. An electronic door lock solenoid is basically an electromagnet consisting of a large coil of copper wire with an armature in the center. When the coil is energized, the slag is pulled into the center of the coil. This allows the solenoid to move to the end (D. & Pappachan, 2019).

2.2 Design Diagrams

2.2.1 Circuit Diagram

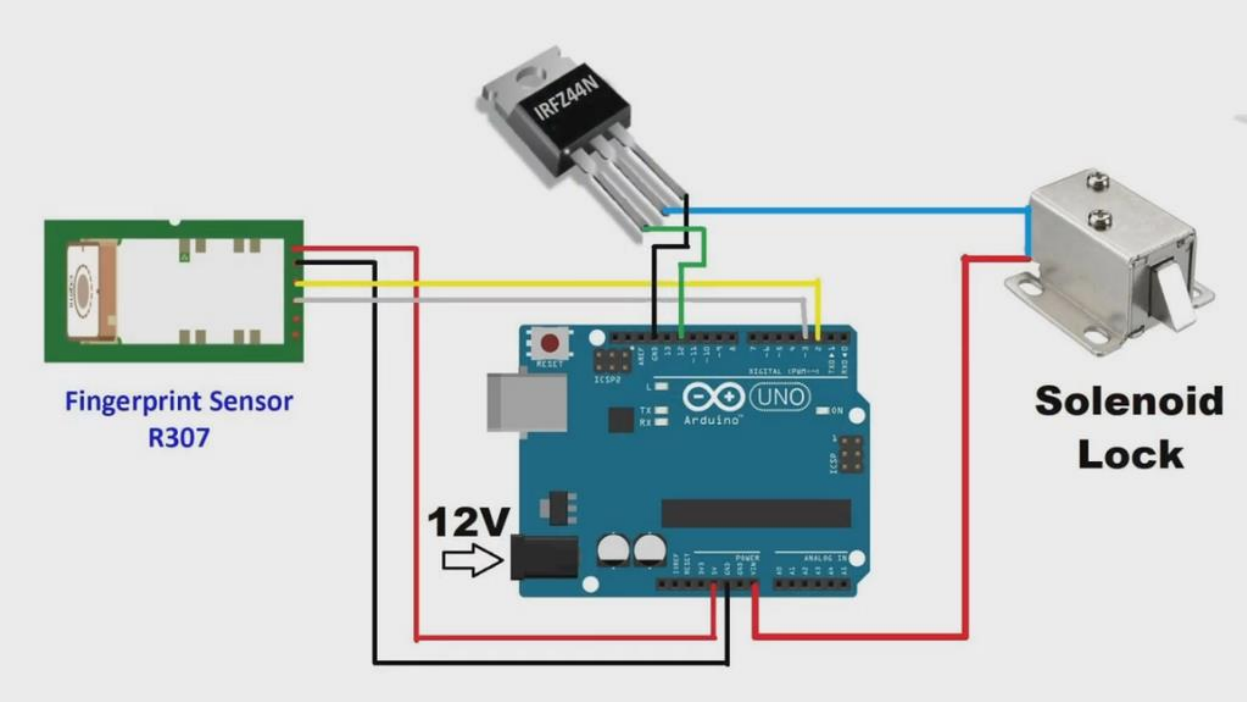


Figure 2: Circuit Diagram of the system

2.2.2 Flowchart

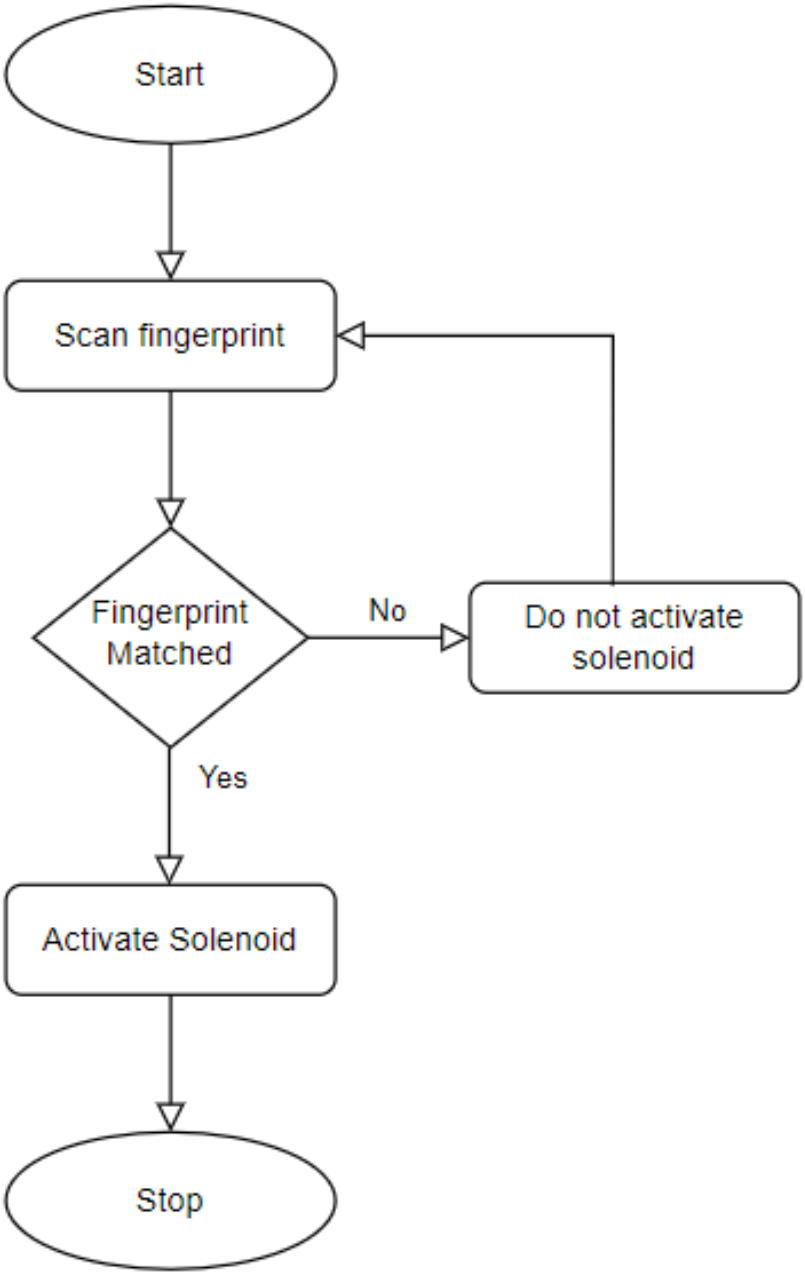


Figure 3: Flowchart of the system

2.3 Requirement Analysis

The various tools required for this project are:

1. **Arduino UNO R3:**

   It is the microcontroller required for programming the device.

Figure 4: Arduino R3 Uno

2. **AS608 Optical Fingerprint Sensor:**

   It is the sensor required for taking the input from user.

Figure 5: Fingerprint sensor

3. **Linear Solenoid Lock Assembly:**

   It is an actuator required for locking and unlocking the door.

Figure 6: Linear solenoid Lock

4. **Battery:**

   It is required for providing power to the device.

Figure 7: Battery

5. **Transistor:**

   It is required to regulate the current flow.

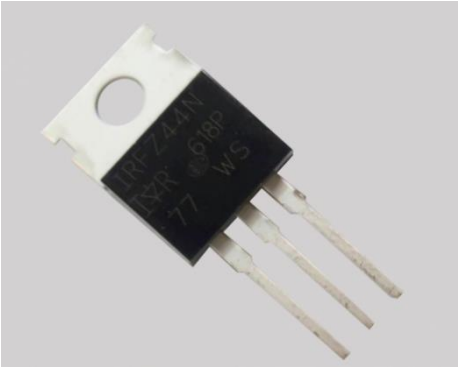


Figure 8: Transistor

6. **Battery Cap with DC Jack**

   It is required to give power to the Arduino by connecting with battery.

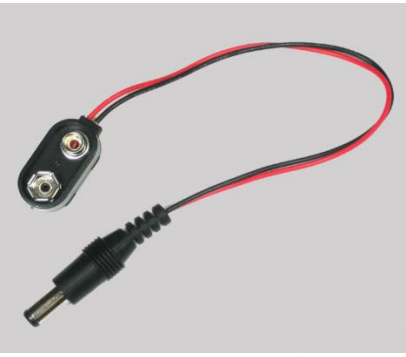


Figure 9: Battery Cap with DC Jack

3. Development

Step 1

Arduino Uno, fingerprint scanner, 9V battery, 12V solenoid, jumper wires, and transistor are the six primary pieces of hardware used in this project. To get entrance to the door, the fingerprint scanner is employed. The Arduino board, on the other hand, is a platform for testing hardware by connecting it to its I/O pins using jumper wires and uploading programs. The latch for the door is a solenoid. The transistor also assists in signal amplification, control, and generation. Finally, it is powered by a battery. The following diagram depicts all the system's components.

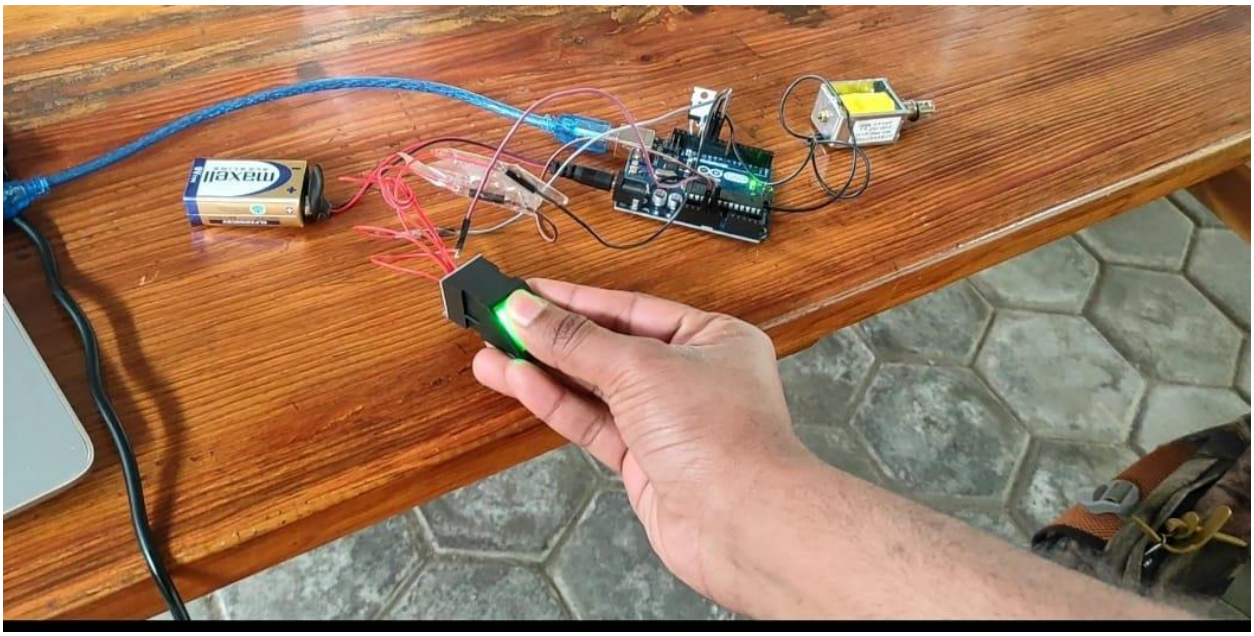


Figure 10: Hardware Components

Step 2

Tinker cad software, a sort of designing program that supports Arduino, was used to create the circuit schematic. The jumper wire connections to the Arduino I/O pins are listed below the table.

Devices used	Connection wires of devices	Arduino I/O pins
Fingerprint scanner	VCC GND Rx Tx	5v GND 11 10
Solenoid	Negative Positive	Power supply Transistor 2nd pin
Transistor	1st 3rd	GND 13

Table 1: Connection of Arduino and I/O pins

Step 3

This step consists of connecting the wires to the fingerprint scanner, solenoid, and the transistor. The fingerprint scanner has four wires and solenoid has two wires, they are connected to the Arduino I/O pins with the help of jumper wires connected to the ports as described in the table above.

Step 4

This step follows the understanding of the circuit operation of fingerprint scanner and solenoid. Here, the Arduino communicates with fingerprint scanner through UART which means, any voltage (+5/ -5), D10 pins helps to flow the data from Arduino to fingerprint sensor and D11 helps flow of data from fingerprint sensor to Arduino. The solenoid communicates with Arduino with the help of power supply ad get the flow of data from the Arduino through D13 pin.

Step 5

Finally, the code for the Arduino to receive input and generate output is written to make this project work. We designed two features in the beginning since we employed a fingerprint scanner in our project. It comprises code for enrollment and a finger verification for the fingerprint scanner. The output will show on the laptop. Each feature has its own code. The listed of coding is given below. Refer Appendix for full coding.

1. Code for enrolling the fingerprint
2. Code for verifying the fingerprint and activating the solenoid

4. Results and Findings

We've shown the components and linked them to the power source for this IoT based Fingerprint door lock using Arduino. This solution is designed to improve security by registering the owner's fingerprint into the Arduino using the fingerprint sensor, and we have provided 5v power to the Arduino via the code uploading wire. When you register yourself by placing your thumb on the fingerprint sensor, the lock will be opened. If you repeat the operation, the solenoid lock will be locked. The Solenoid lock is utilized in this project since the process of locking and unlocking takes less than 5 seconds.

4.1 Fingerprint scanner Enrolling

The fingerprints are scanned using fingerprint scanner. While scanning, the fingerprint scanner will take and store the user's information. To have a database of their fingerprint details, the user needs enroll their fingerprint. The user enters their ID, which is kept alongside their fingerprint minutiae, when the Arduino identifies a fingerprint scanner. It is then recorded in the fingerprint scanner's inbuilt flash memory. As indicated in the diagram above, the user's fingerprint minutiae is enrolled with the ID:2. The image below depicts the outcome of an enrolled fingerprint.

Figure 11: Fingerprint enrolling output

4.2 Fingerprint test and matching

The stored data must be checked when the fingerprint is enrolled to confirm that the fingerprint scanner is correct. The accuracy of the fingerprint scanner is indicated by a degree of confidence that runs from 0 to 255. On the serial monitor, the finger test output is presented in the diagram. The user ID is revealed to be ID: 2. This indicates how confident the current scanned fingerprint and those stored in memory are accurate. The user can insert their fingerprint on the command display displayed as shown in the figure.

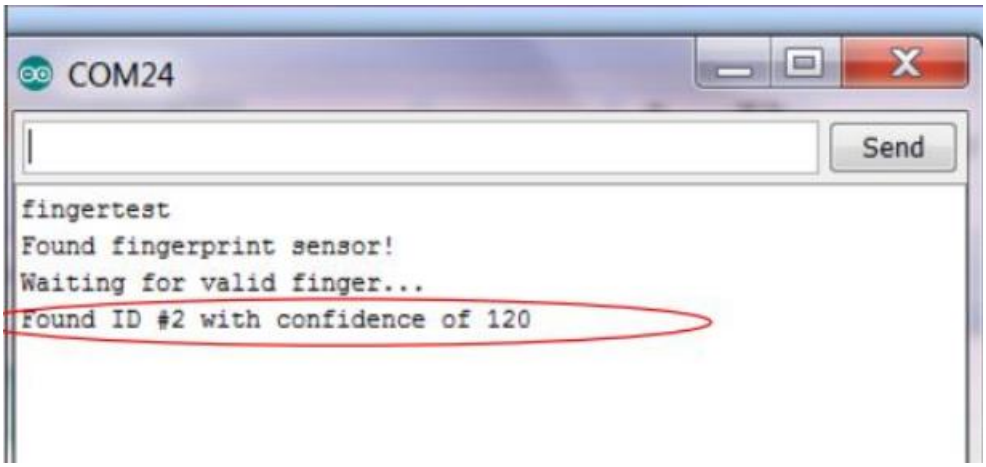


Figure 12: Fingerprint test output with confidence level at serial monitor

4.3 Accuracy Test

The accuracy test determines the security level of a fingerprint scanner. The graphic below depicts the accuracy test result based on confidence level. Five persons were tested, with each receiving four fingerprints. The thumbprints of the left and right hands, as well as the index fingers of the left and right hands, were scanned. The left thumbprint had the highest accuracy, at 70%, followed by the right thumbprint, right index finger, and left index finger, at 67.2 percent, 41.4 percent, and 30%, respectively.

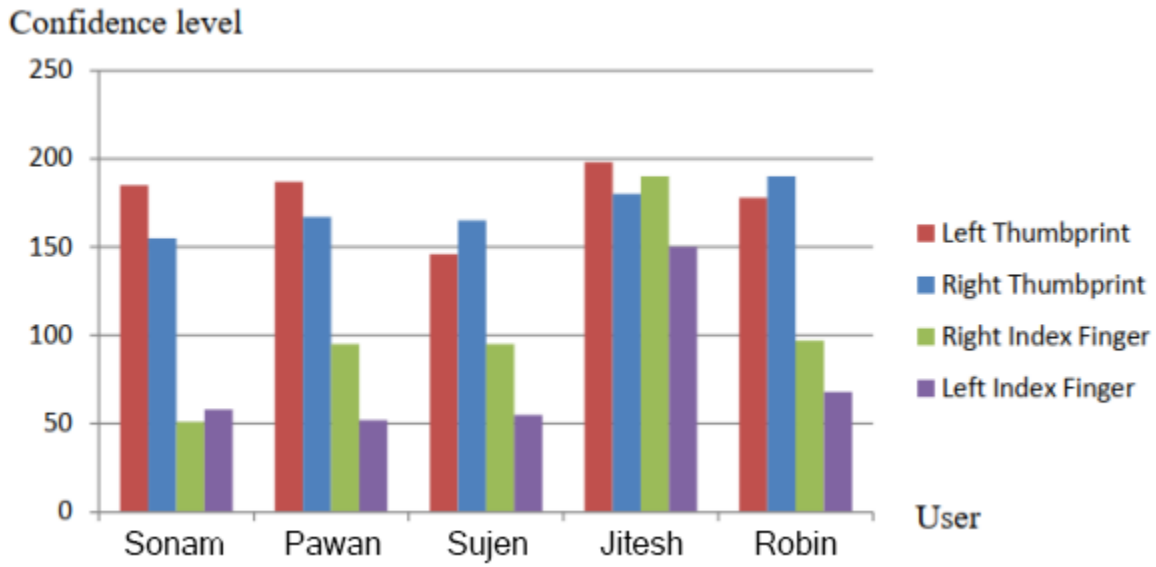


Figure 13: Results of accuracy test

5. Future Works

The system is fully functional as per our requirement. However, there are many things which can be improved in the current system to make it even more efficient and effective. Some of the things which we have planned to implement in the future is the addition of LCD screen which displays the relevant information of the person with their details such as ID, time of login as well as error message when an invalid fingerprint has been placed. Also, we would like to add sound effects which greets the user if the correct fingerprint has been placed as well as notify the users when an invalid fingerprint has been placed.

6. Conclusion

Security has always been and continues to be a source of concern in our homes, offices, businesses, and other public places. Everyone is afraid of an uninvited visitor entering their house or business without their permission. A standard door can be equipped with locks that can be broken with the use of a different key. Alternatives to this approach include the use of a password or pattern in the locks, which has the risk of being disclosed and allowing the lock to be opened. As a result, integrating a door lock with biometrics can be a solution to such issues.

7. References

D., D. & Pappachan, K. C., 2019. *Arduino Projects: Fingerprint Door Unlock System.* [Online]
Available at: https://www.electronicsforu.com/electronics-projects/hardware-diy/arduino-fingerprint-door-unlock-system
[Accessed 02 May 2022].

Devices, E. a. S., 2021. [Online]
Available at: https://www.researchnester.com/reports/smart-door-lock-market/1260

Park, T. I., 2022. [Online]
Available at: https://www.betechlock.com/The-benefits-of-fingerprint-door-locks.html

8. Appendix

8.1 Appendix A: Source Code

8.1.1 Code for enrolling fingerprint

```
#include <Adafruit_Fingerprint.h>


#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)
// 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)
// Set up the serial port to use softwareserial..
SoftwareSerial mySerial(10, 11);

#else
// On Leonardo/M0/etc, others with hardware serial, use hardware serial!
// #0 is green wire, #1 is white
#define mySerial Serial1

#endif


Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");
```

```
  pinMode(13,OUTPUT);

  // set the data rate for the sensor serial port
  finger.begin(57600);
  delay(5);
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }

  Serial.println(F("Reading sensor parameters"));
  finger.getParameters();
  Serial.print(F("Status: 0x")); Serial.println(finger.status_reg, HEX);
  Serial.print(F("Sys ID: 0x")); Serial.println(finger.system_id, HEX);
  Serial.print(F("Capacity: ")); Serial.println(finger.capacity);
  Serial.print(F("Security level: ")); Serial.println(finger.security_level);
  Serial.print(F("Device address: ")); Serial.println(finger.device_addr, HEX);
  Serial.print(F("Packet len: ")); Serial.println(finger.packet_len);
  Serial.print(F("Baud rate: ")); Serial.println(finger.baud_rate);

  finger.getTemplateCount();

  if (finger.templateCount == 0) {
    Serial.print("Sensor doesn't contain any fingerprint data. Please run the 'enroll' example.");
  }
  else {
    Serial.println("Waiting for valid finger...");
```

```
    Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates");
  }
}

void loop()                     // run over and over again
{
  getFingerprintIDez();
  delay(50);            //don't need 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.fingerSearch();
  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);

  return finger.fingerID;
}

// 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;

  digitalWrite(13,HIGH);
  delay(5000);
  digitalWrite(13,LOW);
```

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

8.1.2 Code for verifying the fingerprint and activating the solenoid

```
 /*
  AS608-Optical-Fingerprint-Sensor-enroll
  Home
    based on Adafruit Library

*/
#include <Adafruit_Fingerprint.h>

SoftwareSerial mySerial(10, 11); // TX/RX

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:
      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:
```

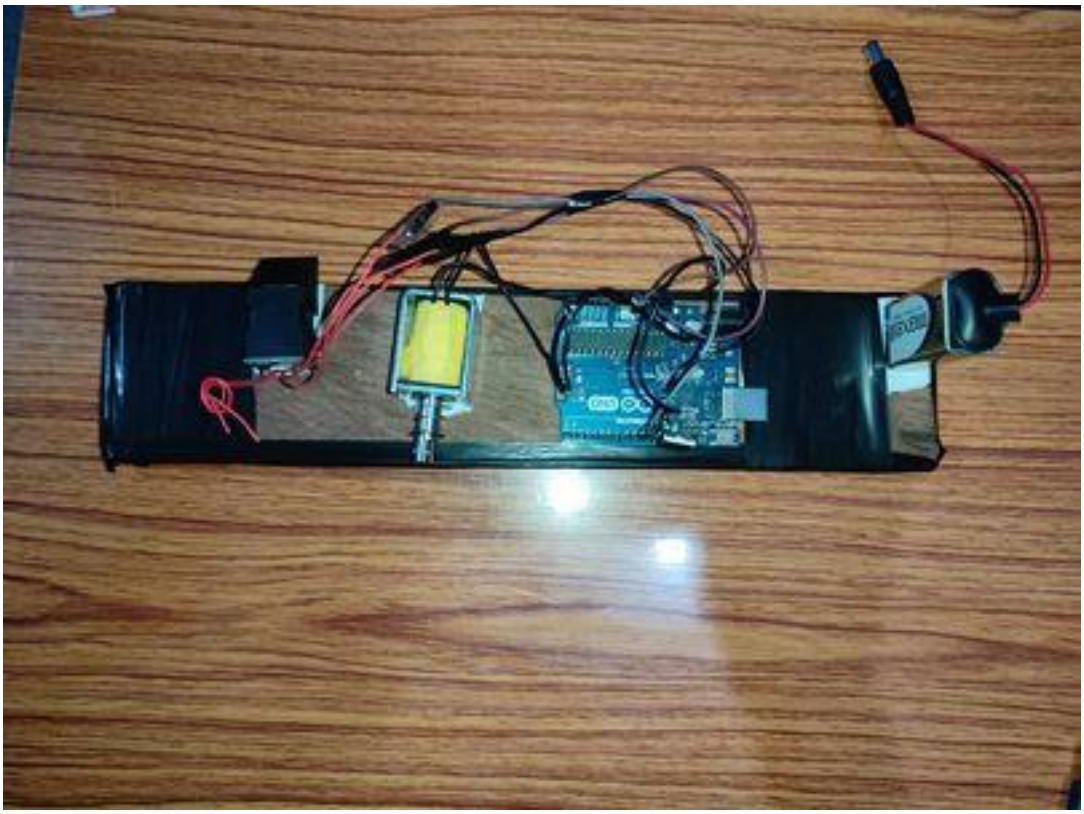
```
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could 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) {
```

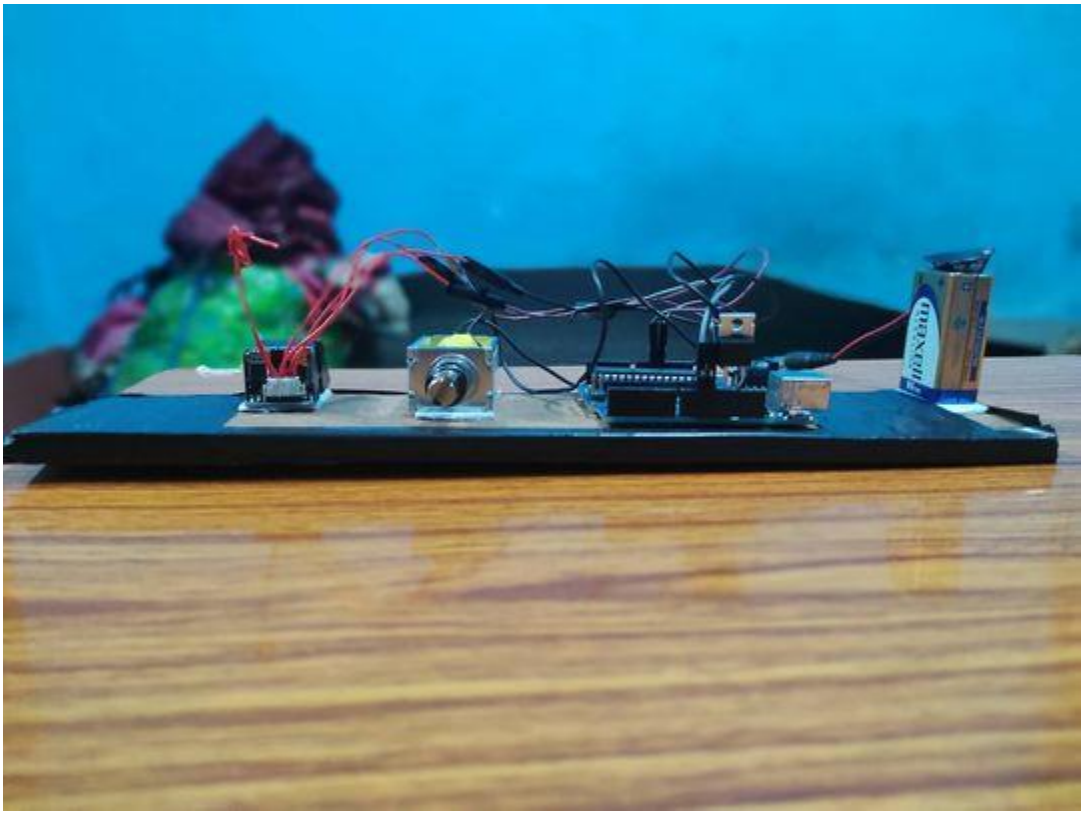
```
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    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;
  }
}
```

8.2 Appendix B: Screenshots of the system





8.3 Appendix C: Design Diagrams