RV College of Engineering®

(Autonomous Institution Affiliated to VTU, Belagavi)

Fingerprint Based Biometric Attendance System using Arduino

Experiential Learning Report

Submitted by:

SKANDA P R – 1RV22CS199

SUHAS RAJ H R – 1RV22CS209

SYED FARHAN ASHRAF – 1RV22CS214

TALASILA DHEERAJ – 1RV22CS216

Submitted to:

DR. ROOPA T S
Assistant Professor
Department of Mechanical Engineering

Table of Contents

Introduction:

In this project, we are going to design a Fingerprint Sensor Based Biometric Attendance System using Arduino. Simply we will be interfacing fingerprint sensor with Arduino, LCD Display & RTC Module to design the desired project. In this project, we used the fingerprint Module and Arduino to take and keep attendance data and records.

Biometric Attendance systems are commonly used systems to mark the presence in offices and schools. This project has a wide application in school, college, business organization, offices, where marking of attendance is required accurately with time. By using the fingerprint sensor, the system will become more secure for the users.

Bill of Materials:

S.N.	**Components**	**Quantity**
1	Arduino UNO Board	1
2	R305/R307 Fingerprint Sensor	1
3	DS3231/DS1307 RTC Module	1
4	16x2 LCD I2C Display	1
5	Push Buttons	4
6	Buzzer 5V	1
7	LED 5mm Red and Green	2
8	Connecting Wires	20
9	Breadboard	1

R305 Fingerprint Scanner Sensor Module:

Introduction:

This is a fingerprint sensor module with TTL UART interface for direct connections to microcontroller UART or to PC through MAX232 / USB-Serial adapter. The user can store the fingerprint data in the module and can configure it in 1:1 or 1: N mode for identifying the person.

The Fingerprint module can be directly interfaced with any microcontroller as well as Arduino Board. This optical biometric fingerprint reader with great features and can be embedded into a variety of end products like access control system, attendance system, safety deposit box, car door locking system.

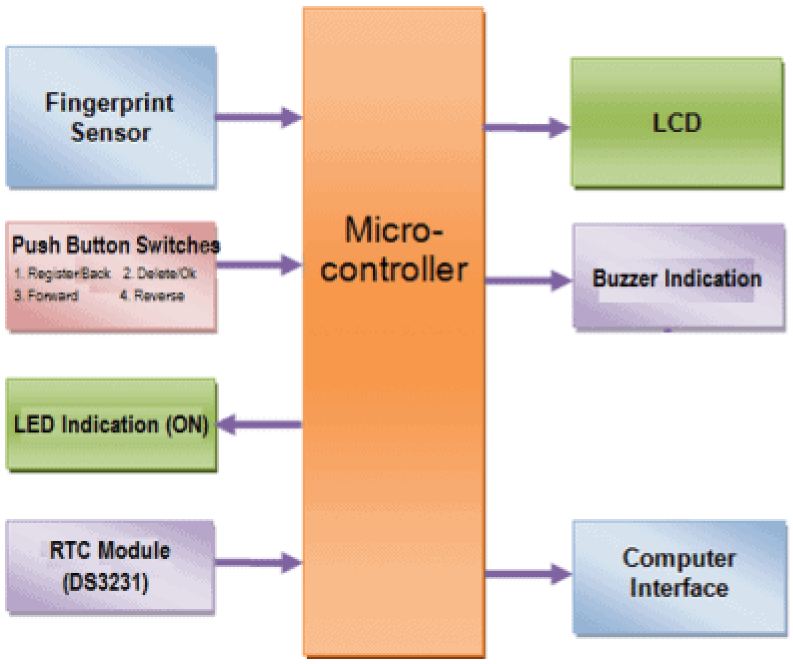
Features:

Integrated image collecting and algorithm chip together, ALL-in-One
Fingerprint can conduct secondary development & embedded into a variety of end products
Low power consumption, low cost, small size, excellent performance
Professional optical technology, precise module manufacturing techniques
Good image processing capabilities can successfully capture image up to resolution 500 dpi.

Specifications:

Fingerprint sensor type: Optical
Sensor Life: 100 million times
Static indicators: 15KVBacklight: bright green
Interface: USB1.1/UART(TTL logical level)
RS232 communication baud rate: 4800BPS~115200BPS changeable
Dimension: 553221.5mm
Image Capture Surface 15—18(mm)
Verification Speed: 0.3 sec
Scanning Speed: 0.5 sec
Character file size: 256 bytes
Template size: 512 bytes
Storage capacity: 250
Security level: 5 (1,2,3,4,5(highest))
False Acceptance Rate (FAR) :0.0001%
False Rejection Rate (FRR): 0.1%
Resolution 500 DPI
Voltage :3.6-6.0 VDC
Working current: Typical 90 mA, Peak 150mA
Matching Method: 1: N
Operating Environment Temperature: -20° to 45° centigrade.

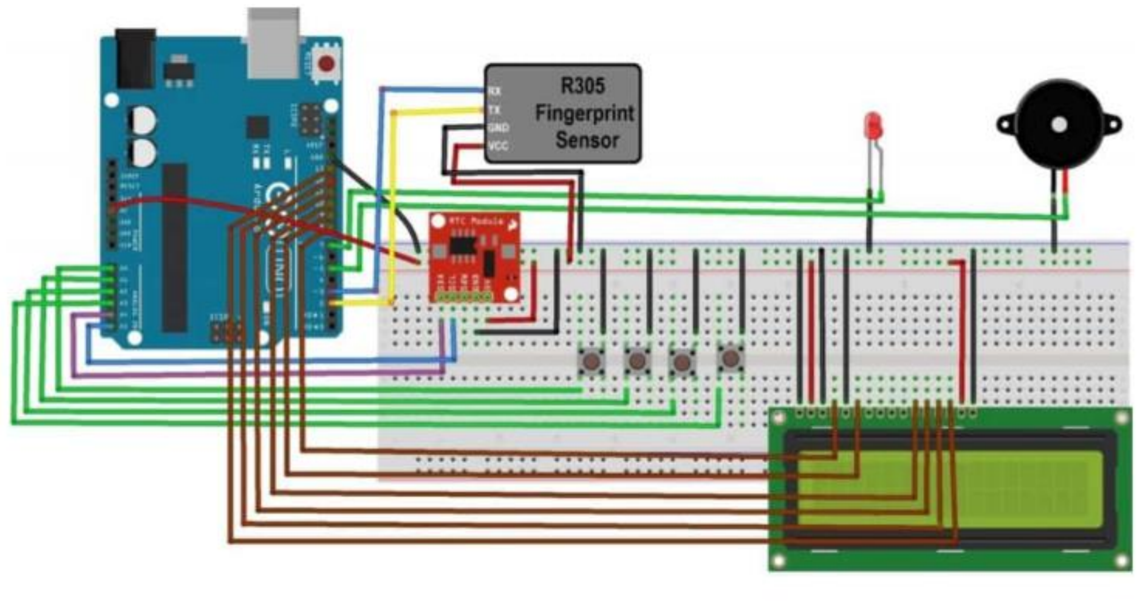
Block Diagram Biometric Attendance System:



In this Fingerprint Sensor Based Biometric Attendance System using Arduino, we used a Fingerprint Sensor module to authenticate a true person or employee by taking their finger input in the system. Here we are using 3 push buttons to register new fingerprint or delete stored fingerprint or match stored fingerprint. The 3 push buttons are used as an input unit for these tasks. Similarly, RTC Module DS3231 is used for registering scanning/entering/existing time of the user.

The LCD displays the time record and every function happening via push button. Buzzer and LED indicates different functions and happening whenever an interrupt is detected.

Circuit Diagram & Connections:



Source Code Program:

The source code/Program for Fingerprint Sensor Based Biometric Attendance System using Arduino is given below. But before that you need to add fingerprint Sensor library & DS3231 Library.

```
#include <RTClib.h>
#include "Adafruit_Fingerprint.h"
#include<EEPROM.h>
#include<LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
#include <SoftwareSerial.h>
SoftwareSerial fingerPrint(2, 3);

#include <Wire.h>
RTC_DS3231 rtc;

uint8_t id;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&fingerPrint);
```

```
#define register_back 14
#define reset 12
#define delete_ok 15
#define forward 16
#define reverse 17
#define match 5
#define indFinger 7
#define buzzer 6
#define rled 7
#define gled 8
#define records 10

int user1, user2, user3, user4, user5, user6, user7, user8, user9, user10;

DateTime now;

void setup()
{
  delay(1000);
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);
  pinMode(register_back, INPUT_PULLUP);
  pinMode(reset, INPUT_PULLUP);
  pinMode(forward, INPUT_PULLUP);
  pinMode(reverse, INPUT_PULLUP);
  pinMode(delete_ok, INPUT_PULLUP);
  pinMode(match, INPUT_PULLUP);
  pinMode(buzzer, OUTPUT);
  pinMode(rled, OUTPUT);
  pinMode(gled, OUTPUT);
  pinMode(indFinger, OUTPUT);

  digitalWrite(buzzer, LOW);
  digitalWrite(rled, LOW);
  digitalWrite(gled, LOW);

  lcd.clear();
  lcd.print(" Fingerprint ");
  lcd.setCursor(0, 1);
  lcd.print("AttendanceSystem");
  delay(2000);
  lcd.clear();
```

```
for (int i = 1000; i < 1000 + records; i++)
{
  if (EEPROM.read(i) == 0xff)
    EEPROM.write(i, 0);
}

finger.begin(57600);
Serial.begin(9600);
lcd.clear();
lcd.print("Finding Module..");
lcd.setCursor(0, 1);
delay(2000);
if (finger.verifyPassword())
{
  Serial.println("Found fingerprint sensor!");
  lcd.clear();
  lcd.print(" Module Found");
  delay(2000);
}
else
{
  Serial.println("Did not find fingerprint sensor :(");
  lcd.clear();
  lcd.print("Module Not Found");
  lcd.setCursor(0, 1);
  lcd.print("Check Connections");
  while (1);
}

if (! rtc.begin())
  Serial.println("Couldn't find RTC");

rtc.adjust(DateTime(__DATE__, __TIME__));

if (rtc.lostPower())
{
  Serial.println("RTC is NOT running!");
  rtc.adjust(DateTime(2018, 6, 7, 11, 0, 0));
}
lcd.setCursor(0, 0);
lcd.print(" Press Match to ");
lcd.setCursor(0, 1);
```

```
  lcd.print(" Start System");
  delay(3000);

  user1 = EEPROM.read(1000);
  user2 = EEPROM.read(1001);
  user3 = EEPROM.read(1002);
  user4 = EEPROM.read(1003);
  user5 = EEPROM.read(1004);
  lcd.clear();
  digitalWrite(indFinger, HIGH);
}

void loop()
{

  digitalWrite(buzzer, LOW);
  digitalWrite(rled, LOW);
  digitalWrite(gled, LOW);

  now = rtc.now();
  lcd.setCursor(0, 0);
  lcd.print("Time: ");
  lcd.print(now.hour(), DEC);
  lcd.print(':');
  lcd.print(now.minute(), DEC);
  lcd.print(':');
  lcd.print(now.second(), DEC);
  lcd.print(" ");
  lcd.setCursor(0, 1);
  lcd.print("Date: ");
  lcd.print(now.day(), DEC);
  lcd.print('/');
  lcd.print(now.month(), DEC);
  lcd.print('/');
  lcd.print(now.year(), DEC);
  lcd.print(" ");
  delay(500);
  int result = getFingerprintIDez();
  if (result > 1)
  {
    digitalWrite(indFinger, LOW);
    tone(buzzer, 100);
    delay(100);
  noTone(buzzer);
  digitalWrite(rled, LOW);
  digitalWrite(gled, HIGH);
  lcd.clear();
  lcd.print("ID:");
  lcd.print(result);
  lcd.setCursor(0, 1);
  lcd.print("Please Wait....");
  delay(1000);
  attendance(result);
  lcd.clear();
  lcd.print("Attendance ");
  lcd.setCursor(0, 1);
  lcd.print("Registered");
  delay(1000);
  digitalWrite(indFinger, HIGH);
  return;
 }
 if (result == 1) {
  digitalWrite(indFinger, LOW);
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  digitalWrite(rled, LOW);
  digitalWrite(gled, HIGH);
  lcd.clear();
  lcd.print("Teacher Found");
  lcd.setCursor(0, 1);
  lcd.print("Downloading Data");
  delay(1000);
  Serial.print("S.No. ");
  for (int i = 0; i < records; i++)
  {
   Serial.print("---------User ID");
   Serial.print(i + 1);
   Serial.print("--------");
  }
  Serial.println();
  int eepIndex = 0;
  for (int i = 0; i < 30; i++)
  {
   if (i + 1 < 10)
    Serial.print('0');
   Serial.print(i + 1);
   Serial.print(" ");
   eepIndex = (i * 7);
   download(eepIndex);
   eepIndex = (i * 7) + 210;
   download(eepIndex);
   eepIndex = (i * 7) + 420;
   download(eepIndex);
   eepIndex = (i * 7) + 630;
   download(eepIndex);
   eepIndex = (i * 7) + 840;
   download(eepIndex);
   eepIndex = (i * 7) + 1050;
   download(eepIndex);
   eepIndex = (i * 7) + 1260;
   download(eepIndex);
   eepIndex = (i * 7) + 1470;
   download(eepIndex);
   eepIndex = (i * 7) + 1680;
   download(eepIndex);
   Serial.println();
  }
 }
 if (digitalRead(reset) == LOW)
 {
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  digitalWrite(rled, HIGH);
  digitalWrite(gled, LOW);
  lcd.clear();
  lcd.print("Please Wait");
  lcd.setCursor(0, 1);
  lcd.print("Reseting.....");
  for (int i = 1000; i < 1005; i++)
   EEPROM.write(i, 0);
  for (int i = 0; i < 841; i++)
   EEPROM.write(i, 0xff);
  lcd.clear();
  lcd.print("System Reset");
  delay(1000);
 }
 checkKeys();
```

```
  delay(300);
}

void attendance(int id)
{
  int user = 0, eepLoc = 0;
  if (id == 1)
  {
    eepLoc = 0;
    user = user1++;
  }
  else if (id == 2)
  {
    eepLoc = 210;
    user = user2++;
  }
  else if (id == 3)
  {
    eepLoc = 420;
    user = user3++;
  }
  else if (id == 4)
  {
    eepLoc = 630;
    user = user4++;
  }
  else if (id == 5)
  {
    eepLoc = 0;
    user = user5++;
  }
  else if (id == 6)
  {
    eepLoc = 840;
    user = user5++;
  }
  else if (id == 7)
  {
    eepLoc = 1050;
    user = user7++;
  }
  else if (id == 8)
  {
    eepLoc = 1260;
    user = user8++;
  }
  else if (id == 9)
  {
    eepLoc = 1470;
    user = user9++;
  }
  else if (id == 10)
  {
    eepLoc = 1680;
    user = user8++;
  }
  else
    return;

  int eepIndex = (user * 7) + eepLoc;
  EEPROM.write(eepIndex++, now.hour());
  EEPROM.write(eepIndex++, now.minute());
  EEPROM.write(eepIndex++, now.second());
  EEPROM.write(eepIndex++, now.day());
  EEPROM.write(eepIndex++, now.month());
  EEPROM.write(eepIndex++, now.year() >> 8 );
  EEPROM.write(eepIndex++, now.year());

  EEPROM.write(1000, user1);
  EEPROM.write(1001, user2);
  EEPROM.write(1002, user3);
  EEPROM.write(1003, user4);
}

void checkKeys()
{
  if (digitalRead(register_back) == 0)
  {
    lcd.clear();
    lcd.print("Please Wait");
    delay(1000);
    while (digitalRead(register_back) == 0);
    Enroll();
  }

  else if (digitalRead(delete_ok) == 0)
  {
    lcd.clear();
    lcd.print("Please Wait");
    delay(1000);
    delet();
  }
}

void Enroll()
{
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  digitalWrite(rled, LOW);
  digitalWrite(gled, HIGH);
  int count = 1;
  lcd.clear();
  lcd.print("Enter Finger ID:");

  while (1)
  {
    lcd.setCursor(0, 1);
    lcd.print(count);
    if (digitalRead(forward) == 0)
    {
      count++;
      if (count > records)
        count = 1;
      delay(500);
    }

    else if (digitalRead(reverse) == 0)
    {
      count--;
      if (count < 1)
        count = records;
      delay(500);
    }
    else if (digitalRead(delete_ok) == 0)
    {
      id = count;
      getFingerprintEnroll();
      for (int i = 0; i < records; i++)
```

```
        {
          if (EEPROM.read(i) != 0xff)
          {
            EEPROM.write(i, id);
            break;
          }
        }
        return;
      }

      else if (digitalRead(register_back) == 0)
      {
        return;
      }
    }
  }

void delet()
{
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  digitalWrite(rled, HIGH);
  digitalWrite(gled, LOW);
  int count = 1;
  lcd.clear();
  lcd.print("Enter Finger ID");

  while (1)
  {
    lcd.setCursor(0, 1);
    lcd.print(count);
    if (digitalRead(forward) == 0)
    {
      count++;
      if (count > records)
        count = 1;
      delay(500);
    }

    else if (digitalRead(reverse) == 0)
    {
      count--;
      if (count < 1)
        count = records;
      delay(500);
    }
    else if (digitalRead(delete_ok) == 0)
    {
      id = count;
      deleteFingerprint(id);
      for (int i = 0; i < records; i++)
      {
        if (EEPROM.read(i) == id)
        {
          EEPROM.write(i, 0xff);
          break;
        }
      }
      return;
    }

    else if (digitalRead(register_back) == 0)
    {
      return;
    }
  }
}

uint8_t getFingerprintEnroll()
{

  int p = -1;
  lcd.clear();
  lcd.print("finger ID:");
  lcd.print(id);
  lcd.setCursor(0, 1);
  lcd.print("Place Finger");
  delay(2000);
  while (p != FINGERPRINT_OK)
  {
    p = finger.getImage();
    switch (p)
    {
      case FINGERPRINT_OK:
        digitalWrite(buzzer, HIGH);
        delay(100);
        digitalWrite(buzzer, LOW);
        digitalWrite(rled, LOW);
        digitalWrite(gled, HIGH);
        Serial.println("Image taken");
        lcd.clear();
        lcd.print("Image taken");
        break;
      case FINGERPRINT_NOFINGER:
        digitalWrite(buzzer, HIGH);
        delay(100);
        digitalWrite(buzzer, LOW);
        digitalWrite(rled, HIGH);
        digitalWrite(gled, LOW);
        Serial.println("No Finger");
        lcd.clear();
        lcd.print("No Finger Found");
        break;
      case FINGERPRINT_PACKETRECIEVEERR:
        digitalWrite(buzzer, HIGH);
        delay(100);
        digitalWrite(buzzer, LOW);
        digitalWrite(rled, HIGH);
        digitalWrite(gled, LOW);
        Serial.println("Communication error");
        lcd.clear();
        lcd.print("Comm Error");
        break;
      case FINGERPRINT_IMAGEFAIL:
        digitalWrite(buzzer, HIGH);
        delay(100);
        digitalWrite(buzzer, LOW);
        digitalWrite(rled, HIGH);
        digitalWrite(gled, LOW);
        Serial.println("Imaging error");
        lcd.clear();
        lcd.print("Imaging Error");
        break;
      default:
        digitalWrite(buzzer, HIGH);
        delay(100);
        digitalWrite(buzzer, LOW);
        digitalWrite(rled, HIGH);
```

```
      digitalWrite(gled, LOW);
      Serial.println("Unknown error");
      lcd.clear();
      lcd.print("Unknown Error");
      break;
  }
}



p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, LOW);
    digitalWrite(gled, HIGH);
    Serial.println("Image converted");
    lcd.clear();
    lcd.print("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    Serial.println("Image too messy");
    lcd.clear();
    lcd.print("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    Serial.println("Communication error");
    lcd.clear();
    lcd.print("Comm Error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    Serial.println("Could not find fingerprint features");
    lcd.clear();
    lcd.print("Feature Not Found");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    Serial.println("Could not find fingerprint features");
    lcd.clear();
    lcd.print("Feature Not Found");
    return p;
  default:
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    Serial.println("Unknown error");
    lcd.clear();
    lcd.print("Unknown Error");
    return p;
}

Serial.println("Remove finger");
lcd.clear();
lcd.print("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");
lcd.clear();
lcd.print("Place Finger");
lcd.setCursor(0, 1);
lcd.print(" 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");
      return;
  }
}



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;
}


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!");
  lcd.clear();
  lcd.print(" Finger Stored!");
  delay(2000);
} 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;
}
```

```
}

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)
  {
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    lcd.clear();
    lcd.print("Finger Not Found");
    lcd.setCursor(0, 1);
    lcd.print("Try Later");
    delay(2000);
    return -1;
  }
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  digitalWrite(rled, LOW);
  digitalWrite(gled, HIGH);// found a match!
  Serial.print("Found ID #");
  Serial.println(finger.fingerID);
  return finger.fingerID;
}

uint8_t deleteFingerprint(uint8_t id)
{
  uint8_t p = -1;
  lcd.clear();
  lcd.print("Please wait");
  p = finger.deleteModel(id);
  if (p == FINGERPRINT_OK)
  {
    digitalWrite(buzzer, HIGH);
    delay(100);
    digitalWrite(buzzer, LOW);
    digitalWrite(rled, HIGH);
    digitalWrite(gled, LOW);
    Serial.println("Deleted!");
    lcd.clear();
    lcd.print("Finger Deleted");
    lcd.setCursor(0, 1);
    lcd.print("Successfully");
    delay(1000);
  }

  else
  {
    Serial.print("Something Wrong");
    lcd.clear();
    lcd.print("Something Wrong");
    lcd.setCursor(0, 1);
    lcd.print("Try Again Later");
    delay(2000);
    return p;
  }
}

void download(int eepIndex)
{
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  digitalWrite(rled, LOW);
  digitalWrite(gled, HIGH);
  if (EEPROM.read(eepIndex) != 0xff)
  {
    Serial.print("T->");
    if (EEPROM.read(eepIndex) < 10)
      Serial.print('0');
    Serial.print(EEPROM.read(eepIndex++));
    Serial.print(':');
    if (EEPROM.read(eepIndex) < 10)
```

```
      Serial.print('0');
    Serial.print(EEPROM.read(eepIndex++));
    Serial.print(':');
    if (EEPROM.read(eepIndex) < 10)
      Serial.print('0');
    Serial.print(EEPROM.read(eepIndex++));
    Serial.print(" D->");
    if (EEPROM.read(eepIndex) < 10)
      Serial.print('0');
    Serial.print(EEPROM.read(eepIndex++));
    Serial.print('/');
    if (EEPROM.read(eepIndex) < 10)
      Serial.print('0');
    Serial.print(EEPROM.read(eepIndex++));
    Serial.print('/');
    Serial.print(EEPROM.read(eepIndex++) << 8 | EEPROM.read(eepIndex++));
  }
  else
  {
    Serial.print("-------------------------");
  }

  Serial.print(" ");
}
```

Working Explanation:

In this project, we have used a DS3231 RTC Module for time & date display. We used 2 LED for success or failure indication, and 1 buzzer for different function indications. We have interfaced 16*2 LCD_I2C which displays everything whenever the finger is placed or removed, or registering attendance or downloading data.

We have used 4 push buttons which are used to control the entire system. The functions of each button are:

1. **Register/Back Button** – Used for enrolling new fingerprints as well as reversing the back process or going back.
2. **Delete/OK Button** – This Button is used for deleting the earlier stored fingerprint system as well as granting access as an OK selection.
3. **Forward Button** – Used for moving forward while selecting the memory location for storing or deleting fingerprints.
4. **Reset button** – Used to clear the EEPROM memory of the Arduino so that newer attendance record can be downloaded.

Enrolling New Fingerprint

To enroll New Fingerprint, click on the Enroll button. Then select the memory location where you want to store your fingerprint using the UP button. Then click on OK. Put your finger and remove your finger as the LCD instructs. Put your finger again. So finally, your fingerprint gets stored.

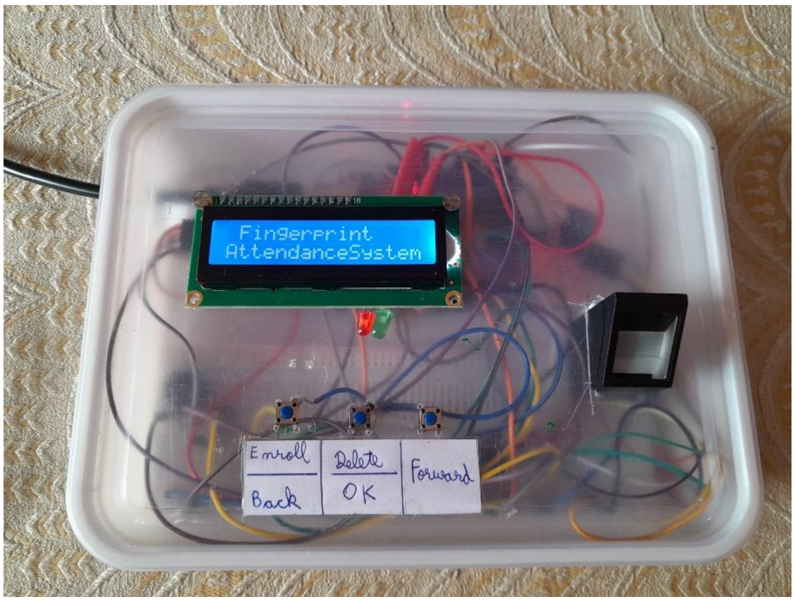
Deleting Stored Fingerprint

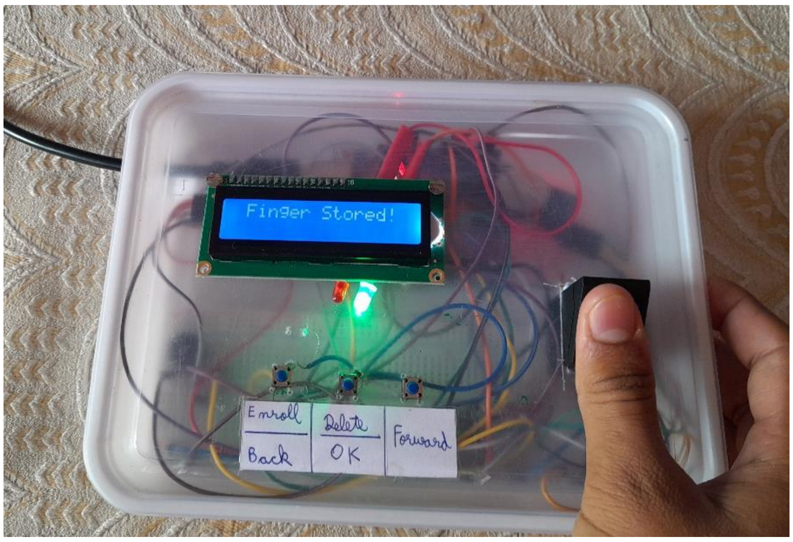
To delete the fingerprint which is stored, click on DEL Button. Then select the memory location where your fingerprint was stored earlier using the UP button. Then click on OK. So finally, your fingerprint is deleted.

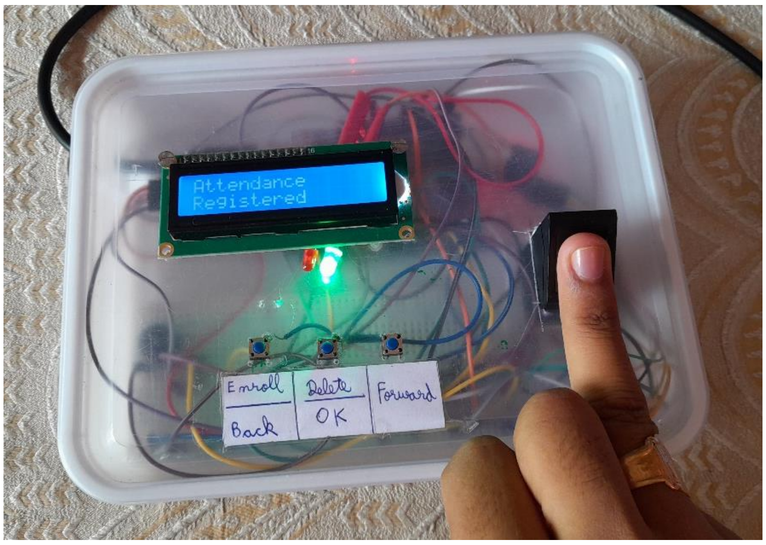
Downloading Data:

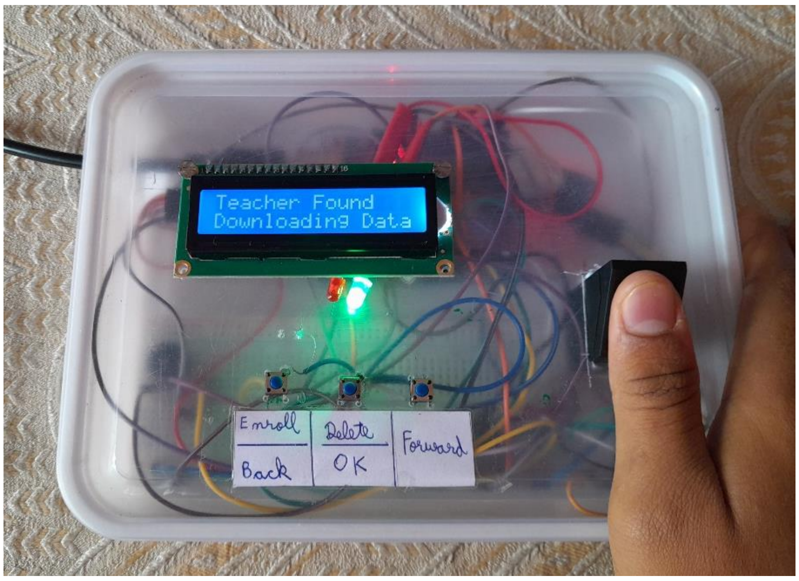
Register a teacher fingerprint as ID-1. Whenever teacher logins, the attendance data will get printed. At this movement, the serial monitor should be opened.

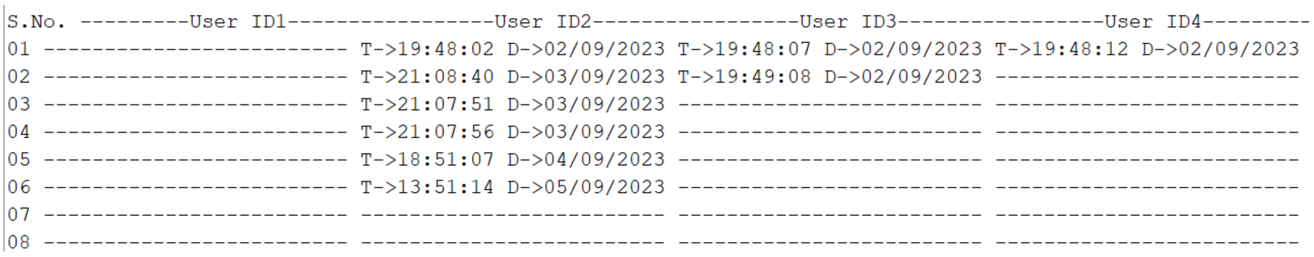
Pictures of the project:



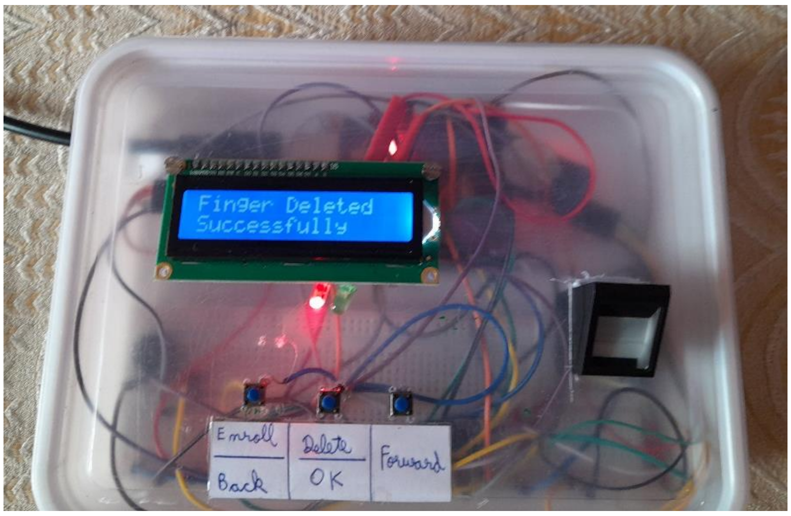








```
S.No. ---------User ID1-----------------User ID2-----------------User ID3-----------------User ID4---------
01 ------------------------- T->19:48:02 D->02/09/2023 T->19:48:07 D->02/09/2023 T->19:48:12 D->02/09/2023
02 ------------------------- T->21:08:40 D->03/09/2023 T->19:49:08 D->02/09/2023 -------------------------
03 ------------------------- T->21:07:51 D->03/09/2023 ------------------------- -------------------------
04 ------------------------- T->21:07:56 D->03/09/2023 ------------------------- -------------------------
05 ------------------------- T->18:51:07 D->04/09/2023 ------------------------- -------------------------
06 ------------------------- T->13:51:14 D->05/09/2023 ------------------------- -------------------------
07 ------------------------- ------------------------- ------------------------- -------------------------
08 ------------------------- ------------------------- ------------------------- -------------------------
```



Demerits of Project:

While fingerprint-based attendance systems have their advantages, they also come with several demerits or drawbacks that organizations should consider when implementing such a system:

1. Privacy Concerns: - Fingerprint data is considered sensitive personal information. Storing and processing this data can raise privacy concerns among students who may be worried about the misuse of their biometric information.

2. Hygiene and Health Concerns: - Fingerprint scanners can potentially become a breeding ground for germs and viruses, especially in high-traffic areas. In the context of infectious disease outbreaks, such as COVID-19, this can pose health risks.

3. False Rejections and False Acceptances: - Fingerprint recognition systems are not perfect and can sometimes produce false rejections (legitimate users being denied access) or false acceptances (unauthorized users gaining access).

4. Environmental Factors: - Environmental conditions, such as dirty or wet fingers, can affect the accuracy of fingerprint recognition.

5. Cost: - Implementing a fingerprint-based attendance system can be expensive, involving the purchase of specialized hardware and software, as well as maintenance costs.

6. Invasive Nature: - Some students may feel uncomfortable with the invasive nature of fingerprint scanning, as it involves physical contact with a device that captures their biometric data.

7. Security Risks: - While biometrics are generally considered secure, they are not immune to hacking or spoofing attempts. Biometric data can be stolen or manipulated, potentially leading to security breaches.

8. Backup and Redundancy: - Fingerprint systems should have backup mechanisms in place to handle situations where the primary system fails, or a student's fingerprint cannot be recognized.

9. Maintenance and Support: - Fingerprint scanners require regular maintenance to ensure accuracy. Technical issues may also arise, necessitating timely support and repairs.

References:

[1] Jain, A. K., Ross, A., & Prabhakar, S. (2004). A Survey of Fingerprint Recognition Techniques. Proceedings of the IEEE.

[2] Saravanan, K., & Nithya, A. (2014). Fingerprint-Based Biometric Attendance System. International Journal of Advanced Research in Computer Engineering & Technology.

[3] Biswas, A., Mukherjee, J., & Mukherjee, S. (2013). Fingerprint-Based Attendance Management System. International Journal of Emerging Technology and Advanced Engineering.

[4] Rajasekaran, K. S., Ganesan, R., & Dhenakaran, S. S. (2013). Biometric Attendance System Using Fingerprint Recognition. International Journal of Computer Applications.

[5] Meherun, H. M., & Reaz, M. B. I. (2011). Development of an Automated Fingerprint Attendance System. Proceedings of the 2011 International Conference on Electrical and Computer Engineering (ICECE).