CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

Approved by AICTE | Affiliated to JNTUH | Accredited by NAAC & NBA

Department of Electronics and Communication Engineering

A.Y: 2021-2022

R18 - III-II –BTech –Sem

Project Based Learning (Hobby Project) Name:

Fingerprint Sensor Based Biometric Attendance System Using Arduino

Submitted By:

198R1A04F0	**K Mahaveera Sharma**
198R1A04F1	**K Venkata Sai Jagdishwar**
198R1A04F2	**K Preethi Naidu**
198R1A04F3	**M Lakshmana Saii**
198R1A04F8	**N Sai**
198R1A04F9	**P Asreeja Reddy**
198R1A04G0	**P Vinay Sai**
198R1A04G1	**P Rakesh**

INDEX

AGENDA:

To create a biometric attendance system based on fingerprints using Arduino.

ABSTRACT:

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 4 push buttons to register new fingerprint or delete stored fingerprint or match stored fingerprint. The 4 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 indicates different functions and happening whenever an interrupt is detected. The LED is used for power indication.

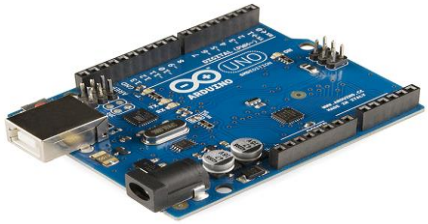
INTRODUCTION:

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.

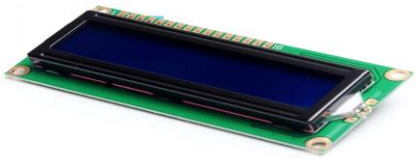
Biometric Attendance systems are commonly used systems to mark the presence in offices and schools. This project has a wide application in school, college, business organization, offices where marking of attendance is required accurately with time. By using the fingerprint sensor, the system will become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

COMPONENTS REQUIRED:

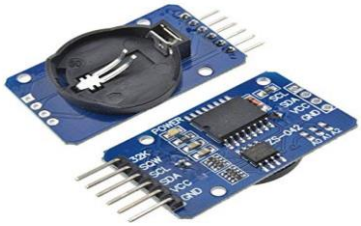
- **Arduino**



- **16x2 LCD**



- **RTC Module**



- **Fingerprint Sensor**

- **Push Buttons**

- **SPST Switch**

- **3V Battery for RTC Module**

- **4V two batteries**

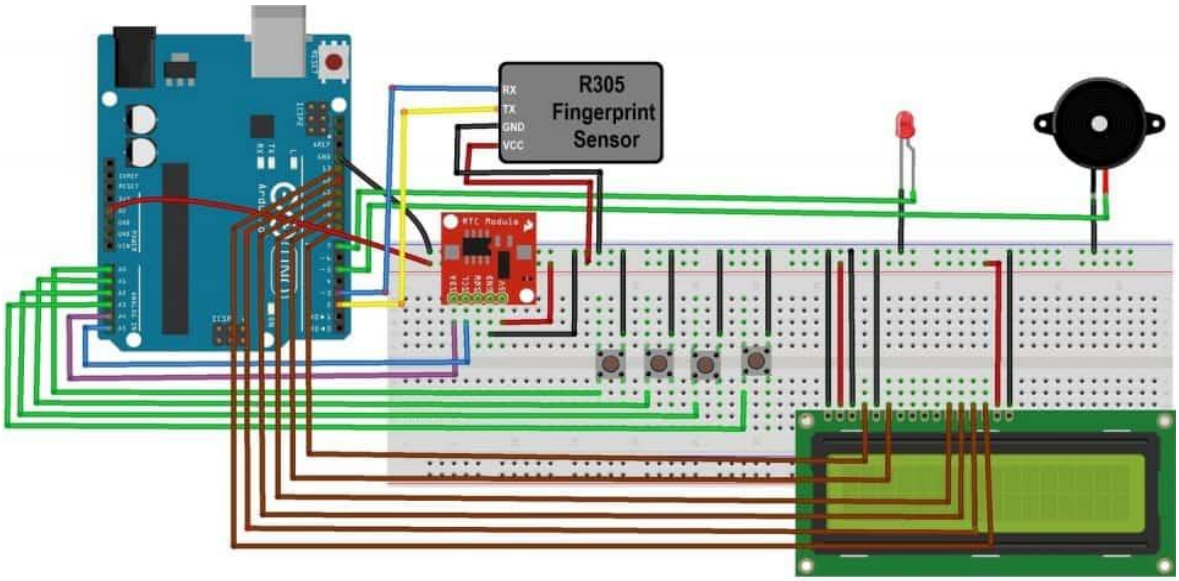
- **Connector wires**

- **Buzzer**

- **Red LED**

- **Plastic Box**

CIRCUIT DIAGRAM:



WORKING:

The “Fingerprint Sensor Based Biometric Attendance System” is basically setting up a system to record the attendance through biometric.

In this project, we have used a DS3231 RTC Module for time & date display. We used 1 LED for power indication, 1 buzzer for different function indication. We have interfaced 16*2 LCD 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 fingerprint 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. Reverse Button – Used for moving backward while selecting memory location for storing or deleting fingerprints.

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/DOWN 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.

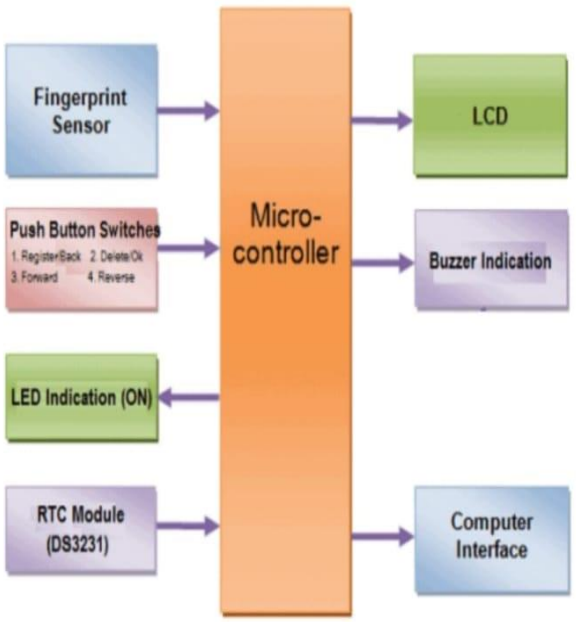
Deleting Stored Fingerprint:

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

Downloading Data:

Simply click on Register/Back Button and reset the button together. At this movement, the serial monitor should be opened.

BLOCK DIAGRAM:



PROGRAM:

```
#include <Wire.h>
#include "RTClib.h" //library file for DS3231 RTC Module
RTC_DS3231 rtc;
uint8_t id;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&fingerPrint);
#define register_back 14
#define delete_ok 15
#define forward 16
#define reverse 17
#define match 5
#define indFinger 7
#define buzzer 5
#define records 10 // 10 for 10 user
int user1,user2,user3,user4,user5,user6,user7,user8,user9,user10;
DateTime now;
void setup() {
delay(1000);
lcd.begin(16,2);
Serial.begin(9600);
pinMode(register_back, INPUT_PULLUP);
pinMode(forward, INPUT_PULLUP);
pinMode(reverse, INPUT_PULLUP);
pinMode(delete_ok, INPUT_PULLUP);
```

```
pinMode(match, INPUT_PULLUP);
pinMode(buzzer, OUTPUT);
pinMode(indFinger, OUTPUT);
digitalWrite(buzzer, LOW);
if(digitalRead(register_back) == 0) {
digitalWrite(buzzer, HIGH);
delay(500);
digitalWrite(buzzer, LOW);
lcd.clear();
lcd.print("Please wait !");
lcd.setCursor(0,1);
lcd.print("Downloding Data");
Serial.println("Please wait");
Serial.println("Downloding Data..");
Serial.println();
Serial.print("S.No. ");
for(int i=0;i<records;i++) {
digitalWrite(buzzer, HIGH);
delay(500);
digitalWrite(buzzer, LOW);
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(delete_ok) == 0) {
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);
} lcd.clear();
lcd.print(" Fingerprint ");
lcd.setCursor(0,1);
lcd.print("Attendance System");
delay(2000);
lcd.clear();
digitalWrite(buzzer, HIGH);
delay(500);
digitalWrite(buzzer, LOW);
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(F(_DATE), F(TIME_)));
if (rtc.lostPower()) {
```

```
Serial.println("RTC is NOT running!");
// following line sets the RTC to the date & time this sketch was compiled
rtc.adjust(DateTime(2018, 6, 7, 11, 0, 0));
// This line sets the RTC with an explicit date & time, for example to set
// June 7, 2018 at 11am you would call:
// 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() {
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>0) {
digitalWrite(indFinger, LOW);
digitalWrite(buzzer, HIGH);
delay(100);
digitalWrite(buzzer, LOW);
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;
} checkKeys();
delay(300);
} // dmyyhms - 7 bytes
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 if(id == 5) // fifth user {
eepLoc=840;
user=user5++;
}*/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);
// EEPROM.write(4,user5); // figth user
} 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() {
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() {
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:
Serial.println("Image taken");
lcd.clear();
lcd.print("Image taken");
break;
case FINGERPRINT_NOFINGER:
Serial.println("No Finger");
lcd.clear();
lcd.print("No Finger Found");
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
lcd.clear();
lcd.print("Comm Error");
break;
case FINGERPRINT_IMAGEFAIL:
Serial.println("Imaging error");
lcd.clear();
lcd.print("Imaging Error");
break;
default:
Serial.println("Unknown error");
lcd.clear();
lcd.print("Unknown Error");
```

```
break;
} }
// OK success!
p = finger.image2Tz(1);
switch (p) {
case FINGERPRINT_OK:
Serial.println("Image converted");
lcd.clear();
lcd.print("Image converted");
break;
case FINGERPRINT_IMAGEMESS:
Serial.println("Image too messy");
lcd.clear();
lcd.print("Image too messy");
return p;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println("Communication error");
lcd.clear();
lcd.print("Comm Error");
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
lcd.clear();
lcd.print("Feature Not Found");
```

```
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println("Could not find fingerprint features");
lcd.clear();
lcd.print("Feature Not Found");
return p;
default:
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;
} }
// 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!");
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) {
lcd.clear();
lcd.print("Finger Not Found");
lcd.setCursor(0,1);
lcd.print("Try Later");
delay(2000);
return -1;
}
// found a match!
Serial.print("Found ID #");
```

```
Serial.print(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) {
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) {
```

```
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(" ");}
```

FEATURES/ADVANTAGES:

- 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

CONCLUSION:

- By Using this model, we can store the fingerprint data in the module and can configure it in 1:1 or 1: N mode for identifying the person.
- By this model, time and manpower are saved.