

Biometric Authentication

A Project as a Course requirement for

Bachelor of Computer Applications

Krishnakanth Gunta (174408)



SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)



SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

CERTIFICATE

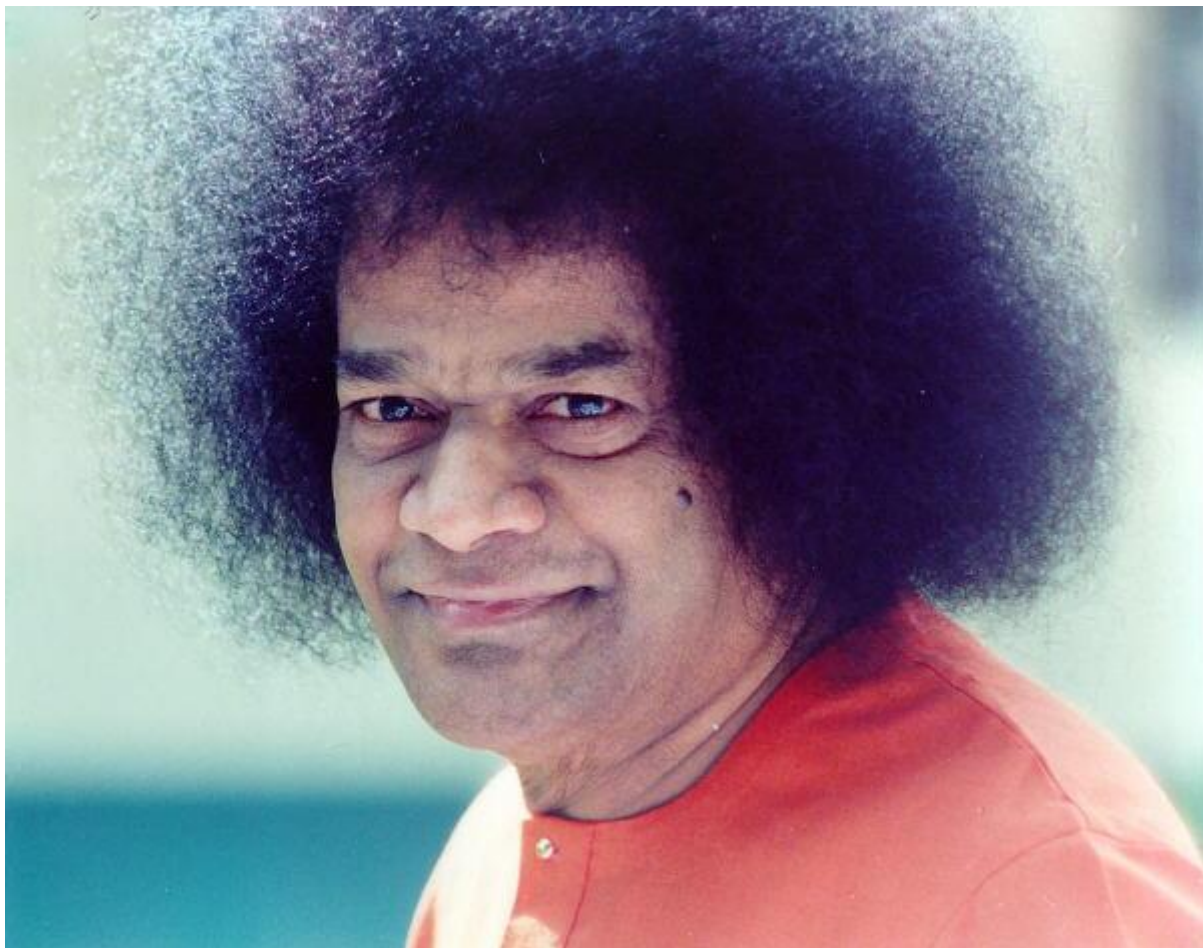
This is to certify that this Project titled **Biometric Authentication** (Finger prints) submitted by Krishnakanth Gunta, 174408, Department of Mathematics and Computer Science, Muddenahalli Campus is a bonafide record of the original work done under my supervision as a Course requirement for the Degree of Bachelor of Computer Applications.

.....

Sri V Bhaskaran,
Project supervisor.

Place: Muddenahalli

Date: 20th march 2020



Dedicating to Bhagawan Sri Sathya Sai Baba

ACKNOWLEDGEMENTS

I express my gratitude from the bottom of my heart to Bhagawan Sri Sathya Sai Baba who blessed and guided me all along this pursuit.

My heartfelt gratitude to my parents for supporting me throughout this journey.

I greatly thank my project supervisor Sri V Bhaskaran sir for providing and supporting with requirements, ideas and thoughts that led to the success of this project. If not for him the project wouldn't have completed.

I sincerely thank all the teachers for supporting me in developing this project.

I thank all my friends for supporting and encouraging me in completing this project and also my seniors for answering my queries and providing valuable inputs.

ABSTRACT

In this era of ever expanding IOT there is a dire need for proper authentication. Many tools and technologies are available to solve this need. One of them is usage of finger print sensor. It is available in market at varied prices with simple, sophisticated codes and languages used to create them. However, each one of them lacks one of the following: convenience, customization, affordability and accessibility. Most of the software that are applications of finger print sensor are platform dependent i.e. same application cannot be used with various different Operating Systems. In order to use these some driver installations are required and list of other dependencies that make life more complicated than making it simpler.

This project is making a humble attempt at developing an indigenous finger print sensor application that can be deployed to use for attendance, secure monetary transactions and upon further improvement, it can be used for maintaining log of entries. We used Arduino technology to develop this application which is platform independent, and works when provided a power source. It can be customized to the needs of our hostel in order to apply it wherever required. The usage of hardware in this application is very minimal compared to the applications available elsewhere using this technology.

CONTENTS

| | |
|--|----|
| 1.INTRODUCTION..... | 14 |
| 1.1 Motivation..... | 14 |
| 1.2 Aim | 14 |
| 2.HARDWARE | 15 |
| 2.1 Arduino Uno..... | 15 |
| 2.1.1 Basic information | 15 |
| 2.1.2 Technical information | 16 |
| 2.2 Node MCU ESP8266..... | 16 |
| 2.2.1 Basic information | 16 |
| 2.2.2 Technical information | 17 |
| 2.3 Finger print scanner TTL GT-5210F12 | 17 |
| 2.3.1 Basic information | 17 |
| 2.3.2 Technical information | 17 |
| 2.4 TFT LCD Touch screen (2.4')..... | 18 |
| 2.4.1 Basic information | 18 |
| 2.4.2 Technical information | 18 |
| 3.SOFTWARE | 19 |
| 3.1 Arduino IDE (Integrated Development Environment) | 19 |
| 3.2 Libraries..... | 20 |
| 3.2.1 Finger print sensor | 20 |
| 3.2.2 TFT Touch Screen (2.4')..... | 20 |
| 3.2.3 Files | 21 |
| 3.2.4 Serial communication | 21 |
| 3.2.5 Time | 21 |
| 3.2.6 Wi-Fi | 21 |
| 4.CONNECTING HARDWARE | 22 |
| 4.1 Connections | 22 |
| 4.1.1 Connecting Arduino and ESP8266 | 22 |
| 4.1.2 Connecting ESP8266 and finger print sensor..... | 23 |
| 4.1.3 Connecting Arduino and TFT touch screen (2.4')..... | 23 |
| 5.FLOW OF EVENTS | 24 |
| 5.1 Components..... | 24 |

| | |
|---|----|
| 5.2 Events in various components | 24 |
| 5.3 Functions of events | 25 |
| 5.4 Brief Flow of project..... | 25 |
| 6.IMPLEMENTATION OF VARIOUS FUNCTIONS | 27 |
| 6.1 Finger print sensor | 27 |
| 6.1.1 Enroll()..... | 28 |
| 6.1.2 Verify()..... | 29 |
| 6.1.3 Delete() | 30 |
| 6.2 Files | 30 |
| 6.2.1 Read() | 31 |
| 6.2.2 Write()..... | 32 |
| 6.3 Time and Date | 32 |
| 6.3.1 Time and date | 33 |
| 6.4 Display..... | 33 |
| 6.4.1 Printing on screen | 33 |
| 6.4.2 Taking Touch Input..... | 34 |
| 6.5 Wi-Fi | 35 |
| 6.5.1 Connecting to Wi-Fi network..... | 35 |
| 6.5.2 Creating Web Server | 35 |
| 6.5.2 Define Handling functions..... | 36 |
| 6.5.3 Handling client requests | 36 |
| 6.5.4 HTML Page | 37 |
| 6.6 Serial Communication | 37 |
| 7. BIOMETRIC AUTHENTICATION ON TOUCH SCREEN..... | 39 |
| 7.1 Biometric authentication on Touch screen..... | 39 |
| 7.1.1 Enroll | 40 |
| 7.1.2 Verify..... | 41 |
| 7.1.3 Delete | 41 |
| 7.1.4 Time_Date | 42 |
| 7.1.5 Admin_change | 42 |
| 7.1.6 Absentees..... | 43 |
| 7.1.7 Logout | 43 |
| 8.BIOMETRIC AUTHENTICATION ON BROWSER | 44 |
| 8.1 Biometric authentication on browser..... | 44 |
| 9.FUTURE SCOPE..... | 47 |
| 10.REFERENCES..... | 48 |

1.INTRODUCTION

1.1 Motivation

Finger prints are unique for each individual human. Identifying an individual using their finger prints is being used across the globe for ease of authenticating and reduce human error in having a regular attendance log.

Our project is to create and implement a Biometric authentication system using finger prints for any event without any human intervention and making job easy.

1.2 Aim

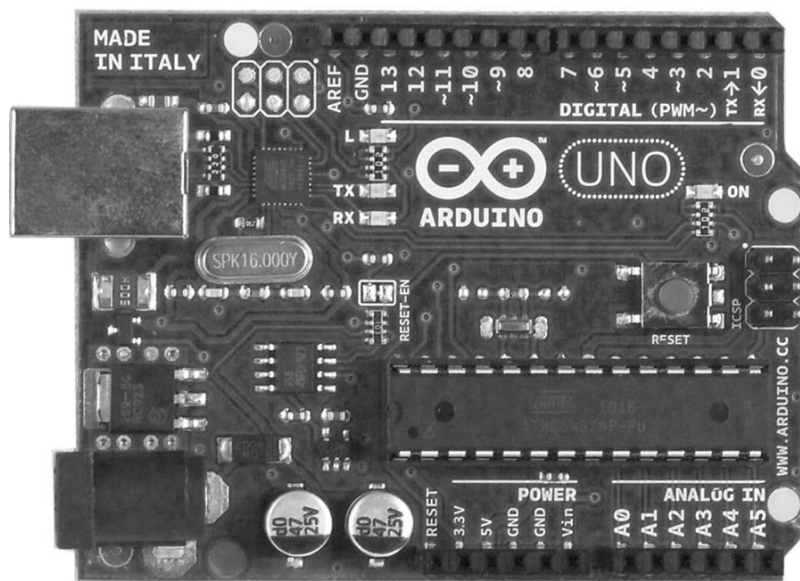
To create a Biometric authentication system which can

- ✓ Enroll a person with his registration number.
- ✓ Verify a person.
- ✓ Take attendance of people with in specified time.
- ✓ Prints all the absentees after the time expiries.
- ✓ Save all the absentees in a text file immediately after time expires and store it for further usage.
- ✓ Validate user's financial transactions and add security for user's money

2.HARDWARE

This chapter gives an overview of hardware that we used for this project.

2.1 Arduino Uno



2.1.1 Basic information

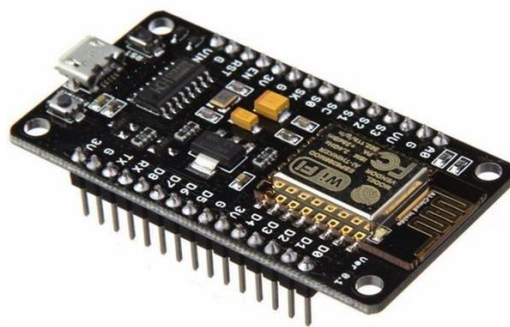
The **Arduino Uno** is an open-source micro controller based on the Microchip ATmega 328p microcontroller and developed by Arduino.cc. which was created by **Massimo Banzi** in Italy, Whose goal was to create simple low cost tools for creating digital projects by non-engineers.

The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards(shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery though it accepts voltages between 7 and 20 volts.

2.1.2 Technical information

- Microcontroller: Microchip ATmega328P
- Operating Voltage: 5 Volts
- Input Voltage: 7 to 20 Volts
- Digital I/O Pins: 14 (of which 6 can provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

2.2 Node MCU ESP8266



2.2.1 Basic information

ESP8266 is a low cost Wi-Fi Microcontroller with TCP/IP protocol and microcontroller capability, produced by Espressif Systems in Shanghai china.

ESP8266 allows to connect to a Wi-Fi network and make simple TCP/IP connections with that network using simple commands.

2.2.2 Technical information

- Processor: L106 32-bit Tensilica Xtensa Diamond
- Clock Speed: 80 MHz
- Architecture: RISC
- Flash memory: 4MB
- Pins: 16 GPIO pins (General purpose input/output)
- Operating Voltage: 5 Volts
- Input Voltage: 7 to 20 Volts

2.3 Finger print scanner TTL GT-5210F12



2.3.1 Basic information

GT-5210F12 finger print sensor from ADH-Tec is an on-board optical sensor which is Capable of 360° recognition. This module is the economical version which can store up to 200 different fingerprints.

2.3.2 Technical information

- Simple UART & USB communication protocol.
- 32-bit ARM Cortex M3 processor.
- Complies with USB 2.0 full-speed specification.
- Ultra-thin optical sensor.
- Capable of 360° recognition.
- Storage for 200 unique fingerprints.
- Works well with dry, moist or rough fingerprints.
- Anti-scratch with surface high hardness.

2.4 TFT LCD Touch screen (2.4')



2.4.1 Basic information

A 2.4' TFT LCD module consists of a bright backlight (4 white LEDs) and a colourful 240X320 pixels display.

It also features individual RGB pixel control giving a much better resolution than the black and white displays. Using this screen, finger presses anywhere on the screen can be detected.

2.4.2 Technical information

- 2.4' TFT LCD.
- Pixels: 240X320.
- Operating Voltage: 3.3V.
- Operating Mode: SPI and 8-bit mode.
- SD card option available for displaying bitmap images.
- Can be easily interfaced with Arduino.

3.SOFTWARE

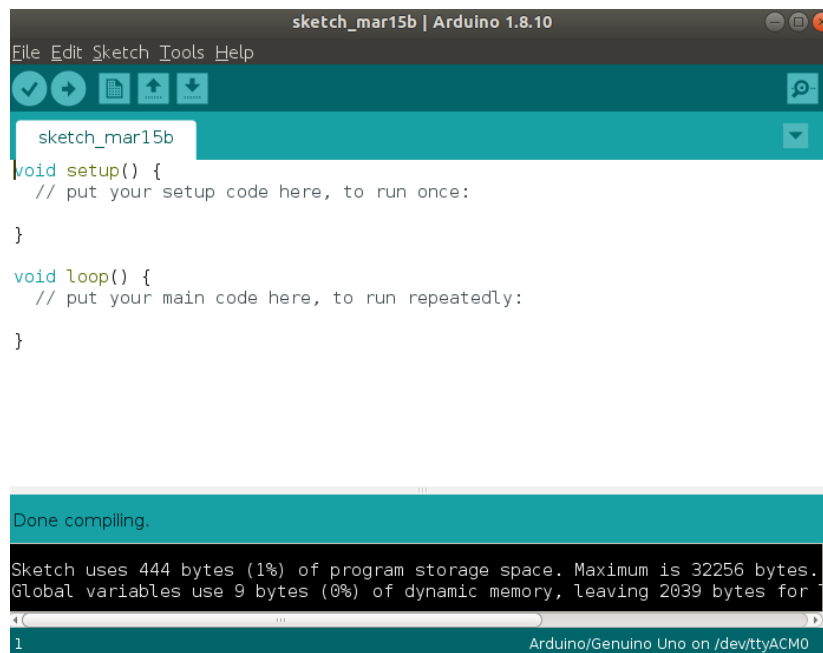
3.1 Arduino IDE (Integrated Development Environment)





The **Arduino Integrated Development Environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. 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 licence version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring.

A program in Arduino is known as a sketch. Its syntax is similar to C/C++. Every sketch will have `setup()` and a `loop()` function. `Setup()` function executes on start of sketch and then it keeps on executing the instructions that are written in `loop`. We can never break `loop()` function.



Above picture is a snapshot of Arduino IDE .On clicking  verify icon the code gets compiled and any errors will be shown in message pane (black area in above figure). On clicking  upload icon the code gets uploaded into Arduino or ESP8266 and gets executed.

3.2 Libraries

Libraries that we used in this project for hardware are as below.

3.2.1 Finger print sensor

Name: Fingerprint Scanner TTL

Author: Josh Hawley

Version: 1.1.0

3.2.2 TFT Touch Screen (2.4')

Name: SPDF5408 Library

Author: Joao Lopes

Version: 0.9.2

3.2.3 Files

Name: LittleFS

Author: Ivan Grokhotkov

Name: sdFat

Author: Bill Greiman.

3.2.4 Serial communication

Name: SoftwareSerial

Author: Arduino

Version: 1.0

3.2.5 Time

Name: Time

Author: Michael Margolis

Version: 1.6

3.2.6 Wi-Fi

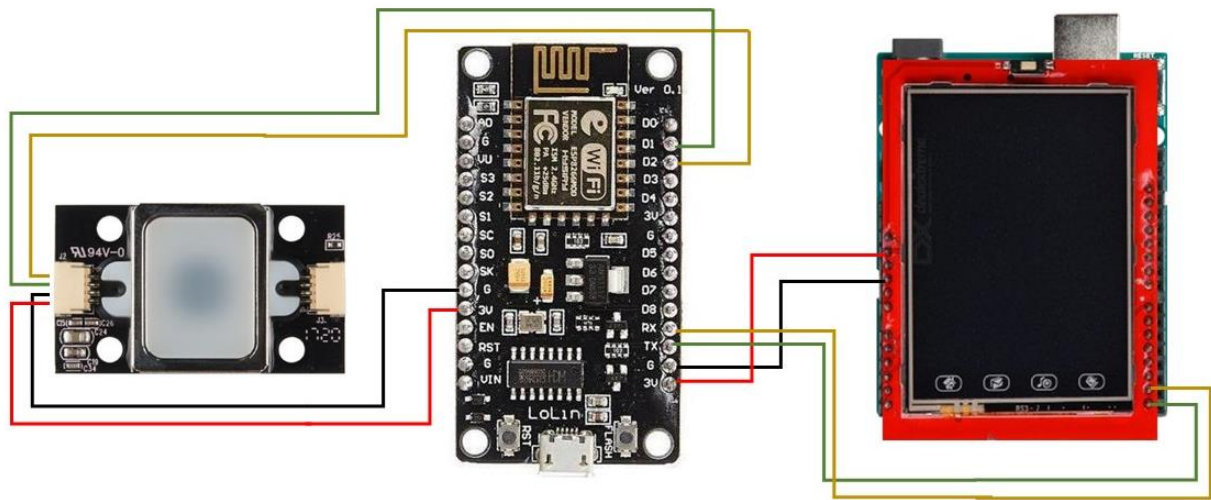
Name: ESP8266WiFi and ESP8266WebServer

Author: Ivan Grokhotkov and Wuweixin

4.CONNECTING HARDWARE

Understanding and connecting hardware is very important in these kinds of projects.

As shown in the figure TFT Touch screen is connected to Arduino as a shield. Arduino Uno and ESP8266 are connected to each other and they communicate using Serial communication. Finger print sensor is connected to ESP8266.



4.1 Connections

The connections between different hardware are as below:

4.1.1 Connecting Arduino and ESP8266

Pins of Arduino 0(Rx) 1 (Tx)

Pins of ESP8266 Tx Rx

4.1.2 Connecting ESP8266 and finger print sensor

Pins of ESP8266 3.3 GND 4 5

Finger print sensor 3.3 GND Tx Rx

4.1.3 Connecting Arduino and TFT touch screen (2.4')

Stack the LCD screen on to Arduino Uno so that it connects to Arduino.

5.FLOW OF EVENTS

This chapter will provide a brief explanation of the flow of various different components and events that happen in this project.

5.1 Components

There are various components in the project.

- Finger print.
- Touch screen Display.
- Time.
- Serial communication.
- Wi-Fi.

5.2 Events in various components

These are the various events that happen in each component.

- Finger print
 - Register or Enroll.
 - Verify 1 to 1 and identify 1 to N.
 - Delete.
 - Attendance.
- Touch screen Display
 - Display.
 - Taking touch input.
 - Save log data.
- Time and date
 - Current time.
 - Maintaining time limits for an event.
- Serial communication
 - Communication between Arduino Uno and ESP8266.
- Wi-Fi
 - Connecting ESP8266 with Wi-Fi.
 - Make HTTP requests and get Responses from browser.

5.3 Functions of events

Enroll: This event will register a user with his finger print and registration number.

Verify: This event verifies a user (user's finger print) with his ID and also identifies a user (user's finger print) among N IDs.

Delete: This event deletes the finger print of specified user.

Attendance: This event will take attendance of users between specified time interval.

Display: This event will display messages on screen.

Touch input: This event will take touch input from user on touch screen as an analog signal.

Save log data: This event saves the data into SD card after time expires.

Current time: This event will calculate and give the current time.

Time limits: This event will accept the users only in given time interval.

Serial communication: This event establishes communication between Arduino and ESP8266.

HTTP: This event makes HTTP requests and receives HTTP responses.

5.4 Brief Flow of project

There are two modules in the project. First is taking biometric attendance using finger print sensor and screen. Later is Authenticating financial transactions through browser.

A person or a user gets enrolled himself using either touch screen or web browser using his registration number. He scans his finger thrice with help of

finger print sensor. The sensor makes a template using 3 fingerprints and provides an ID which is in range of 0 to 199. Then a mapping is created with registration number and ID generated by Finger print sensor so that user need not remember the ID and can use his own registration as ID.

In first module an event is defined between a time interval. User can put his finger on finger print sensor between given time interval. After time expires all the ID's of absentees will be saved to a text file with date in SD card connected to Arduino.

The other module of the project is browser oriented, where a user can authenticate his financial transactions by giving his user ID and putting his finger print. All basic functions like verifying, deleting etc. can even be done through browser.

6.IMPLEMENTATION OF VARIOUS FUNCTIONS

This chapter will provide a detailed understanding of how various functions are implemented across the different components.

6.1 Finger print sensor

This component will take finger print as input from sensor and use below functions to implement different methods.

fps.SetLED (true): Turns on the LED lights present inside the sensor. LED must to on to work with finger prints. Returns true if success else returns false.

fps.IsFigurePress(): checks whether a finger is pressed on sensor or not. Returns true if pressed else returns false.

fps.DeleteID(ID): deletes the finger print of ID. Returns true if ID is successfully deleted else returns false.

fps.Verify1_1(ID): verifies the currently pressed finger against a specific ID. Returns true if matched else false.

fps.Verify1_N(): Verifies the currently pressed finger against all enrolled fingerprints. Returns ID if finger present else returns 200.

fps is an object of **FPS_GT511C3** class.

6.1.1 Enroll()

Below snippet of code shows the core implementation of Enroll() function.

Enroll1(), Enroll2() and Enroll3() functions are used to take first, second and third finger print respectively and merges all three to make a template and stores in its memory.

```
if (capture != false)
{
    Serial.println(F("Remove finger"));
    Status = fps.Enroll1();
    while (fps.IsPressFinger() == true)
        delay(100);
    Serial.println(F("Press same finger again"));
    while (!fps.IsPressFinger())
    {
        if (breakloop())
            return -1;
    }
    capture = fps.CaptureFinger(true);
    if (capture != false && Status == 0)
    {
        Serial.println(F("Remove finger"));
        Status = fps.Enroll2();
        while (fps.IsPressFinger() == true)
            delay(100);
        Serial.println(F("Press same finger again"));
        while (!fps.IsPressFinger())
        {
            if (breakloop())
                return -1;
        }
        capture = fps.CaptureFinger(true);
        if (capture != false && Status == 0)
        {
            Serial.println(F("Remove finger"));
            Status = fps.Enroll3();
        }
    }
}
```

6.1.2 Verify()

Below snippet of code shows the implementation of verify() function.

fps.Verify1_N() returns the ID of the finger that is pressed on the sensor.

```
void Verify()
{
    bool c = false;
    int id = 300;
    while (!fps.IsPressFinger())
    {
        if (breakloop())
            return;
    }
    c = fps.CaptureFinger(true);
    if (c)
    {
        id = fps.Identify1_N();
        if (id < 200)
        {
            if (id == 0 || id == 199)
            {
                Serial.print(F("Administrator"));
                return;
            }
            Serial.print(F("Finger print found\nid : "));
            uint32_t i = readId(id);
            Serial.println(i);
        }
        else
            Serial.println(F("No match found"));
    }
    else
    {
        Serial.println(F("Failed to capture"));
        Serial.println(F("try again"));
    }
}
```

6.1.3 Delete()

Below snippet shows the implementation of delete() function.

fps.delete(ID) function deletes the finger print of the ID that is passed as parameter to it.

```
int Deleteid(String reg1){
    int id = -1;
    bool c = false;
    uint32_t reg = 0;
    for (int i = 1; i < 199; i++){
        reg = readId(i);
        if (String(reg) == reg1){
            id = i;
            modify(i);
        }
    }
    bool is_exists = fps.CheckEnrolled(id);
    if (is_exists){
        if (id == 0 || id == 199){
            Serial.println(F("change admin before delete"));
            return -1;
        }
        fps.DeleteID(id);
        Serial.print(F("Finger print id : "));
        Serial.print(reg1);
        Serial.println(F(" is deleted"));
        return 1;
    }
    else{
        Serial.print(F("No Finger Found with id :"));
        Serial.println(reg1);
    }
    return -1;
}
```

6.2 Files

Files are the most important component in this project as we are not connecting Arduino with database to save data. Files are used to store the mapping of Registration numbers with ID's that are generated by Finger print sensor.

Example:

When a user enrolls himself with his finger prints, finger print sensor will generate an ID which is in range 0 to 199. Here a map must be created with user's registration number and the ID generated by sensor.

Number - ID

174401 - 13

174402 - 14

The above map must be stored, so that even after power supply goes off we can use them.

In ESP8266 there is a space built-in to store files which is non-volatile in nature. We can create, read, write, modify delete those files.

myFile is an object of class **File** and LittleFS is the library that is being used for handling files in ESP8266.

There are few functions that are used in this module:

LittleFS.open(): this function opens a file in specified opening mode

myFile.read(): reads characters from the file.

myFile.print(): writes the string into file that is passed to it.

myFile.close(): closes the file that is opened.

6.2.1 Read()

```
void readFile()
{
    char c = '\0';
    File myFile = LittleFS.open("/record.txt", "r");
    if (myFile)
    {
        while (myFile.available())
        {
            c = myFile.read();
            Serial.print(c);
        }
        myFile.close();
    }
}
```

Above code shows us how basic reading of file is implemented in ESP8266.

LittleFS.open() will take 2 parameters to open a file. First is file name and second is opening mode. There are different modes in which a file can be opened. r for read, w for write and a for append etc. After doing all the operations we should close the file using myFile.close() function.

6.2.2 Write()

```
void writeFile(String s)
{
    File myFile = LittleFS.open("/record.txt", "a");
    if (myFile)
    {
        myFile.print(s);
        myFile.close();
    }
}
```

The above snippet shows the basic code for writing to a file. LittleFS.open() will take two parameters i.e. file name and opening mode. As we are appending the new contents to the file, we open it in append mode. then myFile.print(String) will append the string to the file that is opened. myFile.close() will close the file that is opened.

Storing the mapping of registration numbers with ID's is efficient in ESP8266 but storing log data in ESP8266 is not a good idea because there is limited memory available for files on ESP8266. So we used sdFat library to store log data on SD card connected to Arduino. Basic functions like read and write are common in both the file systems.

6.3 Time and Date

Time is one of the major components in the project that is used to display time. There is a function to set time and date. To retrieve date and time we use below functions.

hour(): returns the hours part of the time.

minutes(): returns minutes part of the time.

weekday(): returns weekday (1 if Sunday, 2 if Monday and so on)

day(): returns the date (if date is 24 FEB 2020 it returns 24)

month(): returns the month part.

year(): returns the year part.

6.3.1 Time and date

```
void digitalClockDisplay()
{
    int d = weekday();
    Serial.print("@ ");
    Serial.print(Day[d - 1]);
    Serial.print(", ");
    printzero(day());
    Serial.print(" ");
    int m = month();
    Serial.print(Month[m - 1]);
    Serial.print(" ");
    Serial.print(year());
    Serial.print(" ");
    Serial.print(" ");
    printzero(hour());
    printDigits(minute());
    Serial.println();
}
```

The above snippet shows the basic implementation of displaying time.

6.4 Display

This component deals with displaying messages on screen and taking touch inputs from user. These are few functions in touch screen.

6.4.1 Printing on screen

Lcd.fillScreen(COLOR): sets the screen color. COLOR is a parameter which is passed as hexadecimal code.

Lcd.setCursor(X,Y): sets the Cursor on screen to a particular point (X,Y). X and Y are coordinates.

Lcd.setTextColor(COLOR): sets the text color. COLOR is a parameter which is passed as hexadecimal code.

Lcd.setTextSize(SIZE): sets the text size. SIZE is a parameter which is passed as integer.

```
lcd.fillScreen(BLACK);  
lcd.setCursor(0, 20);  
lcd.setTextColor(WHITE);  
lcd.setTextSize(1);
```

lcd is an object of Adafruit_TFTLCD class.

6.4.2 Taking Touch Input

```
TSPoint waitOneTouch()  
{  
    TSPoint p;  
    do  
    {  
        p = ts.getPoint();  
        pinMode(XM, OUTPUT);  
        pinMode(YP, OUTPUT);  
    } while ((p.z < MINPRESSURE) || (p.z > MAXPRESSURE));  
  
    return p;  
}
```

ts is an object of TouchScreen class.

6.5 Wi-Fi

This component is used to connect ESP8266 with Wi-Fi networks.

6.5.1 Connecting to Wi-Fi network

```
//SSID and Password of your WiFi router
const char* ssid = "WIFI SSID";
const char* password = "WIFI PASSWORD";

ESP8266WebServer server(80);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}

Serial.println("\nConnected to WiFi");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
```

By giving SSID and PASSWORD of the Wi-Fi network as show above, we can connect ESP8266 to that network. To connect ESP8266 to Wi-Fi we use ESP8266WiFi library.

WiFi.begin(ssid, password) function tries to connect to the network. WiFi.status() function will return WL_CONNECTED if ESP8266 is connected to Wi-Fi network. After connecting to the network WiFi.localIP() gives the IP address.

6.5.2 Creating Web Server

```
ESP8266WebServer server(80);
```

To create a webserver we use a ESP8266WebServer library.

server is an object of class ESP8266WebServer. We should send a port number as a parameter at which it listens. So, as shown in above figure server object is created and listening at port 80.

6.5.2 Define Handling functions

```
server.on("/", handleRoot);
```

```
server.begin();
```

In setup() function we have to define functions to handle web requests present on root.

We use server.on (root,event) function to handle web requests. It takes two parameters root which specifies location and event which specifies what function to be called.

6.5.3 Handling client requests

```
void loop()  
{  
  server.handleClient();  
}
```

We must service the requests of client periodically. We have handleClient() function to take care of clients. As said in introduction every sketch has a loop function which cannot be broken. So, we put handleClient() function inside loop() to call the function again and again to handle client requests.

6.5.4 HTML Page

```
const char HTML_Page[] = R"=====  
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <title>HTML page</title>  
</head>  
  
<body>  
  <p>Hello word</p>  
</body>  
  
</html>  
  
)=====";
```

The html page that is displayed on the browser will be stored in a string as shown in above image.

```
void handleRoot()  
{  
  server.send(200, "text/html", HTML_Page);  
}
```

server.send() function sends a response for the request. The parameters are code, content Type and content in form of a string.

So, code is 200(OK), content type is text/HTML and the string that was in previous figure HTML_Page.

6.6 Serial Communication

Serial communication is done using a protocol called UART (Universal Asynchronous Receiver/Transmitter). In the serial communication method, we send one bit at time, sequentially, over a communication channel. If you have 32 bits of data, the data will send one by one bit. Speed of data sending is depending on a serial speed, called baud rate. For instance, Arduino UNO has UART pins on digital pin 0 and 1.

```

#include <SoftwareSerial.h>
SoftwareSerial ESPserial(0, 1);

void setup()
{
    Serial.begin(115200);
    ESPserial.begin(115200);
}

void loop()
{
    if ( ESPserial.available() )
    {
        Serial.write( ESPserial.read() );
    }

    if ( Serial.available() )
    {
        ESPserial.write( Serial.read() );
    }
}

```

Above image shows the basic implementation of serial communication between Arduino and ESP8266. Arduino will send data to ESP8266 using ESPserial.write() and ESP8266 will send data to Arduino using Serial.write().

7. BIOMETRIC AUTHENTICATION ON TOUCH SCREEN.

This chapter will provide a detailed understanding of biometric authentication using touch screen.

7.1 Biometric authentication on Touch screen

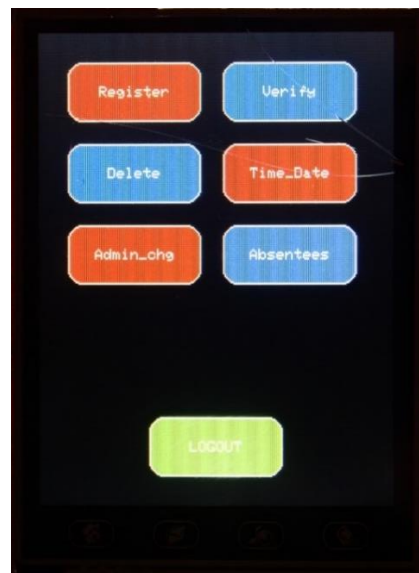


Above image shows the default screen that is displayed on touch screen. On powering the Arduino this default page is displayed on screen in which it will be printing the time. This mode is known as attendance mode.

In this mode it will be waiting for users to put their fingers on the sensor. If a valid user puts his finger on the finger print sensor, then the registration number is displayed on the screen and the user's attendance is taken. If a non-user puts his finger on finger print sensor, then the message "no user" is displayed on the screen. If the finger moves too fast that the sensor could not capture the finger, then the message "Failed to capture" is displayed on screen.

After the given time limit is crossed, then a message “Saving in progress...” is displayed on screen and all the registration numbers of absentees will be stored in a text file which will be stored on SD card.

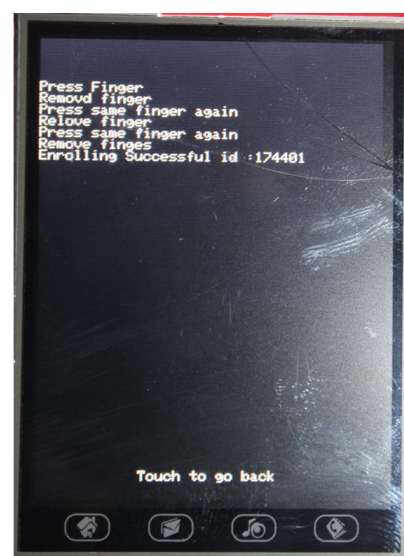
Any time if admin (super user) puts his finger on the sensor a menu is displayed on screen as shown in below figure.



The admin menu consists of Register, Verify, Delete, Time_date, Admin change, absentees and logout buttons.

7.1.1 Enroll

On clicking Register button a keypad is displayed as shown below.



Admin will give the registration number and the click 'S' button on keypad which will redirect to Enroll function where user puts his finger thrice and gets enrolled. If ID is invalid, then it will show an error message.

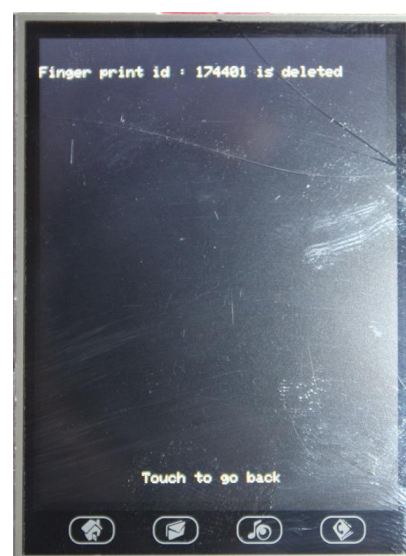
7.1.2 Verify

On clicking verify button, a message "put your finger on sensor" will be displayed on screen. If a valid user puts his finger, then his registration number is displayed or else message "no user" will be displayed.



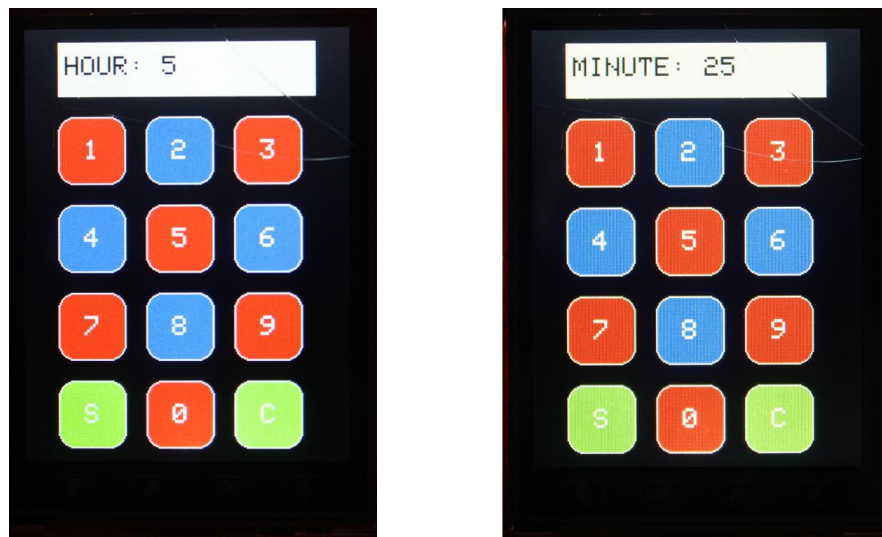
7.1.3 Delete

On clicking Delete button, keypad is displayed on the screen. Admin will enter the number that is to be deleted and it will redirect to delete function and deletes the finger prints of specified user. If ID is invalid, then it will show an error message.



7.1.4 Time_Date

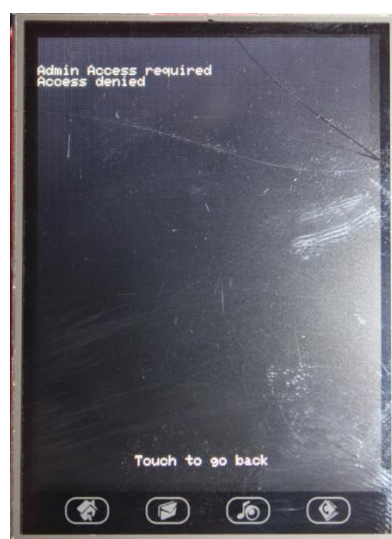
On clicking Time_date button a new keypad will be displayed on screen as shown below.



Using keypad, we can set Hours, minutes, date, month and year. The above image is of setting hours and minutes. If Hours, Minutes, Year, Day, and Month are invalid then an error message displayed.

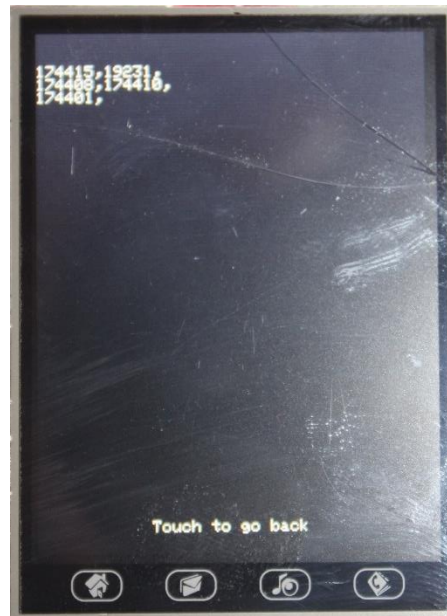
7.1.5 Admin_change

On clicking Admin change button, it will ask for admin access if access is granted then it enrolls a new person as an admin and removes the existing admin else an error message “Access denied” is displayed as shown in below figure.



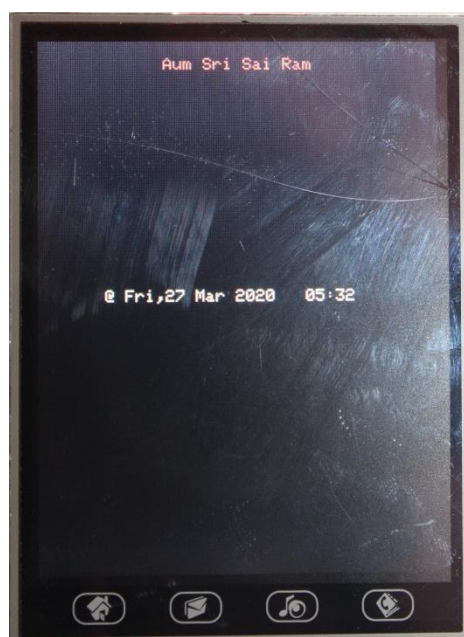
7.1.6 Absentees

On clicking Absentees button, screen will display all the registration numbers of absentees as shown below.



7.1.7 Logout

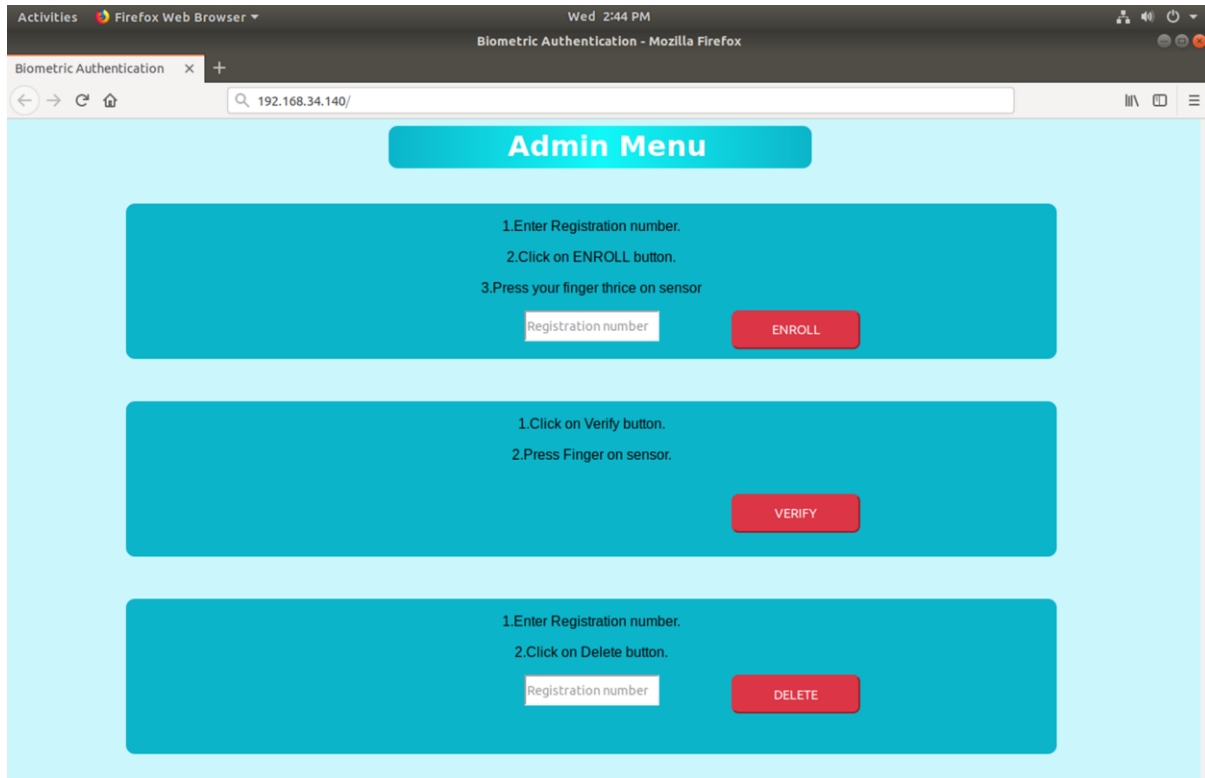
On clicking logout button, the screen and program goes into default mode i.e. attendance mode as shown below.



8. BIOMETRIC AUTHENTICATION ON BROWSER

This chapter will provide a detailed understanding of authentication on browser.

8.1 Biometric authentication on browser

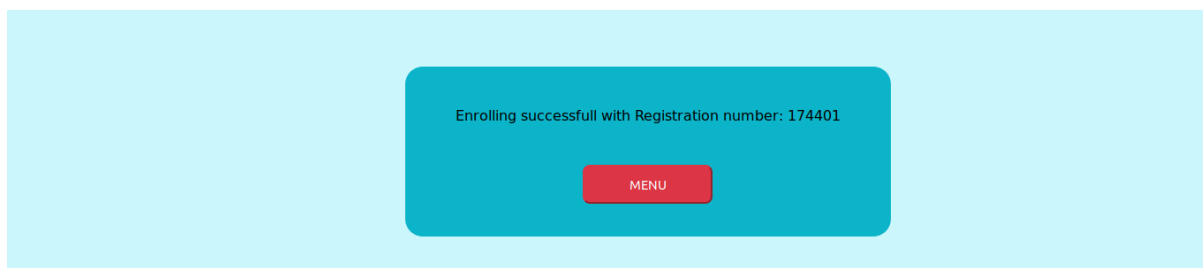


Above image shows the UI of browser implementation of Admin menu. On clicking the IP address of server the above page will open on browser.

The Admin menu consists of options Enroll, Verify, Delete, Change admin and show all ID's.

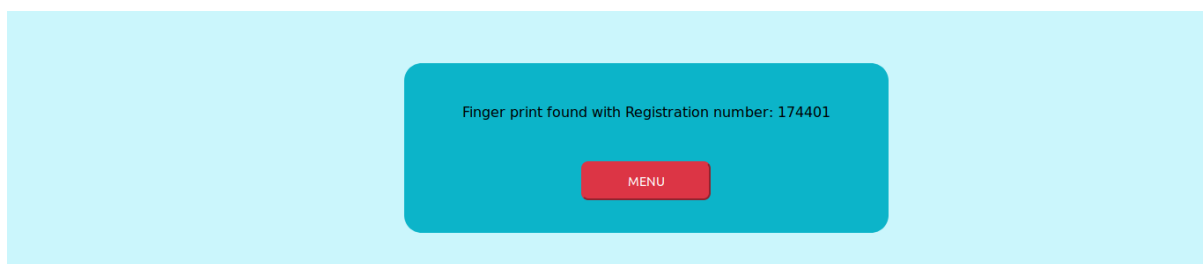
Enroll - Registers a user with registration number given in the input box. If registration number already exists, then it will display an error message "Registration number already exists". If Fingerprint already exists, it will display an error message Fingerprint already exists. If Enroll failed due to any reason, then it will display an error message "Enroll failed". If no input is given to sensor within ten seconds, then an error message "Time expired" is displayed.

On successfully enrolling message is displayed as shown below.



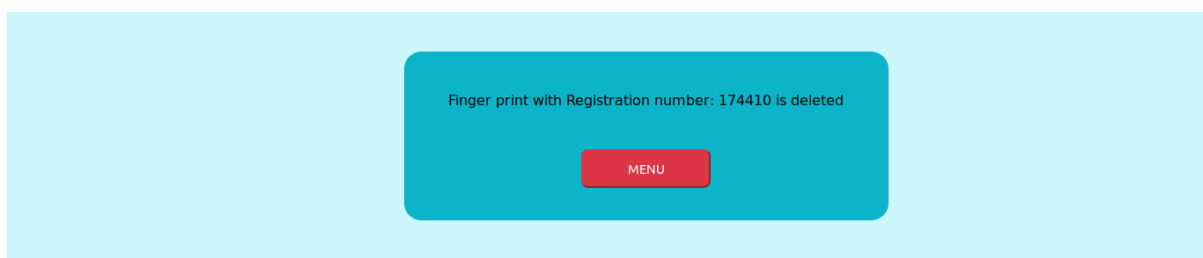
Verify - Verifies whether a user exists or not if exists user's registration number is displayed else a message "No match found" is displayed. If no input is given to sensor within ten seconds, then an error message "Time expired" is displayed.

When a user exists a message is displayed as shown below.



Delete - Deletes the user with registration number passed to it. If registration number exists, it will delete else an error "no user found with that registration number" is displayed.

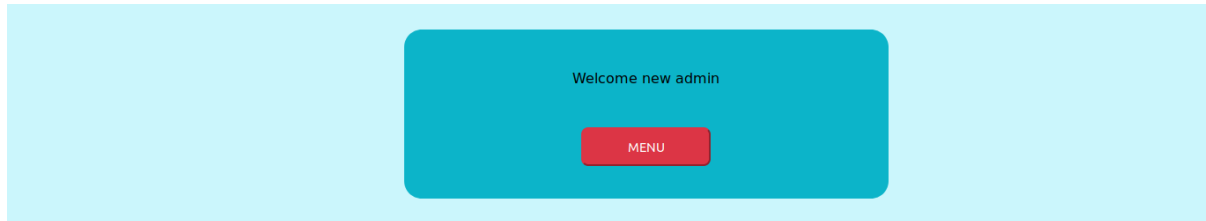
On successfully deleting below message is displayed.



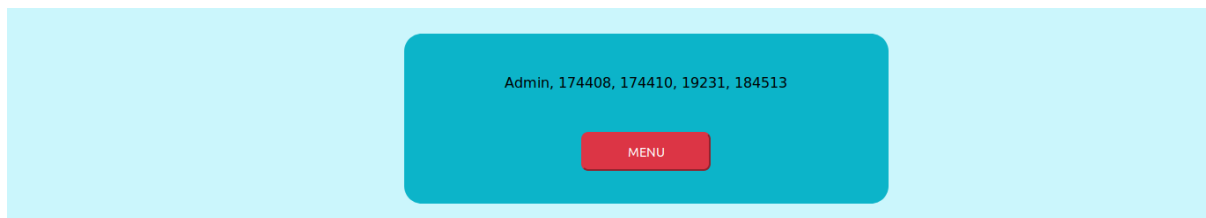
Change Admin - Changes the admin to a new person. If an existing user tries to enroll an error message is displayed. For enrolling new admin, internally we use

Enroll method. If no input is given to sensor within ten seconds, then an error message “Time expired” is displayed.

On successfully changing admin a message is displayed as shown below.



Show ID's - shows all the users registration numbers as shown below.



9.FUTURE SCOPE

9.1 Future Scope

The transaction module in the project which is authenticating financial transactions, can be integrated in HSBC (hostel's e-wallet) to add security for user's money.

The Biometric authentication on touch screen can be used for taking attendance for suprabhatham which happens on daily basis in our hostel.

This system can be implemented in any place where ever there is a need for a log register like library, Lab etc. to reduce human work and also error.

RTC (real time clock) can be included to this project to reduce the task of setting time whenever it is powered on.

10.REFERENCES

10.1 Books

Kolbans book on ESP8266 by Neil Kolban

Arduino Cookbook by Michael Margolies

NodeMCU ESP8266 communication methods and protocols by Manoj R. Thakur

Arduino development cookbook by Cornel Amariei

10.2 Online resources

Course on IOT by professor Ian G. Harris, University of California, Irvine.

www.arduino.cc

en.wikipedia.org

www.github.com

www.circuits4you.com

www.randomnerdtutorials.com

www.circuitbasics.com

www.instructables.com

www.stackoverflow.com