FINGERPRINT BASED BIOMETRIC ATTENDANCE SYSTEM USING ARDUINO

A report submitted in partial fulfillment of the requirements

for the award of

Bachelor of Technology

In

Electronics and Communication Engineering

By

A.SURYA MOHAN KIRAN **(20BQ1A0408)**

A.MOHIT KUMAR **(20BQ1A0409)**

A.DURGA PRASAD **(20BQ1A0410)**

Open-Source Hardware Tools for Electronics Engineers

(Skill Oriented Course

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

(Approved by AICTE and permanently affiliated to JNTUK, Accredited by NBA and NAAC)

NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508

JUNE 2022

DEPARTMENT OF ECE (Electronics and Communication Engineering)

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: NAMBUR

(AUTONOMOUS)

CERTIFICATE

This is to certify that the Mini Project titled "**Fingerprint based biometric attendance system using arduino**" is a bonafide record of work done by **A.SURYAMOHANKIRAN(20BQ1A0408),A.MOHIT KUMAR(20BQ1A409),A.DURGA PRASAD(20BQ1A0410)** as part of the Skill Oriented Course **Open-Source Hardware Tools for Electronics Engineers** in partial fulfillment of the requirement of the degree for Bachelor of Technology in Electronics and Communication Engineering during the academic year 2021–22.

Mr. S. KRISHNA PRASAD
Course Coordinator

DR. M. Y. BHANU MURHT
Head of Department

DECLARATION

We, **A.SURYA MOHAN KIRAN(20BQ1A0408), A.MOHIT KUMAR(20BQ1A0409), A.DURGA PRASAD(20BQ1A0410),** hereby declare that the report entitled "**Fingerprint based biometric attendance system using arduino**" is done by us as part of the Skill Oriented Course **Open-Source Hardware Tools for Electronics Engineers** in partial fulfillment of the requirement of the degree for Bachelor of Technology in Electronics and Communication Engineering during the academic year 2021–22.

DATE:

PLACE:VVIT,NAMBUR

SIGNATURE OF THE CANDIDATES

A.SURYA MOHAN KIRAN(20BQ1A0408)

A.MOHIT KUMAR (20BQ1A0409)

A.DURGA PRASAD (20BQ1A0410)

ACKNOWLEDGEMENT

We thank Dr. Y. Mallikarjuna Reddy, Principal, Vasireddy Venkatadri Institute of Technology, Nambur, for providing us the resources for carrying out the project.

Our sincere thanks to Dr. M. Y. Bhanu Murthy, Head, Department of ECE, for his co-operation and guidance which helped us to make our project successful and complete in all aspects.

We also express our sincere thanks and are grateful to our course coordinators Mr. S. Krishna Prasad, Associate Professor, and Mrs. K. Sandhya Rani, Assistant Professor of Department of ECE, for motivating us to make our project successful and fully complete. We are grateful for their precious guidance and suggestions.

We also place our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

NAME OF THE CANDIDATES :

A.SURYA MOHAN KIRAN (20BQ1A0408)

A.MOHIT KUMAR (20BQ1A409)

A.DURGA PRASAD (20BQ1A0410)

TABLE OF CONTENTS

ABSTRACT

In industrial and domestic applications attendance registering is important at each and every moment. Many face a lot of problems due to lack of proper attendance monitoring system. In this project we use Fingerprint Sensor (R307) which senses the Fingerprint of a particular person; a buzzer and Led gets activated whenever a person places his finger on the sensor. Then the fingerprint is stored in cloud with id no. Many people can store their fingerprints. Then next time any person puts their finger on the sensor it checks there are any matching fingerprints or not. If his fingerprint matches with any of the stored fingerprints then the LCD display shows which person it is and the time & date of checking. In this model, all the fingerprints are stored each and every time someone places his finger. User can connect the system wirelessly with the cloud and monitor the process. When the app is running on the computer, data sent by R307 fingerprint module is received and stored on the cloud and displayed in serial monitor and 16*2 LCD display module. This study has mainly focused to develop IOT based biometric attendance system, that is able to keep record of attendance and count the data for daily purpose. In this project we are going to design 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 are using fingerprint Module and Arduino to take and keep attendance data and records. Attendance systems are commonly used systems to mark the presence in offices and schools. From manually marking the attendance in attendance registers to using high-tech applications and biometric systems, these systems have improved significantly. This project has a wide application in school, college, business organization, offices where marking of attendance is required accurately with time.

CHAPTER-1
INTRODUCTION

In the World of Technology, Biometrics plays an effective role in identifying Human beings. Through this project, you will develop a unique system that can identify students for attendance purpose using their fingerprints. 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. You will need an Arduino Uno board for interfacing microcontroller with the Finger Print Scanner R307/R305. So with the help of Finger Print Scanner R307/R305, we will store the finger prints of all the students and once they are stored, the Finger Print Scanner will compare the present finger print on the scanner and previously stored finger prints. If any finger print is matched, the microcontroller will print the concern data stored for the particular finger print on the LCD Display. In addition to this, we can add Wi-Fi module, to upload the data into remote cloud, so as to access the entire unit from the sole system of it from anywhere in the world. Attendance plays a major role in educational institutions. The most common means of taking attendance in the classroom is by calling out the roll numbers of students or asking the students to manually sign the attendance sheet, which is passed around during the lecture. The process of manually taking and maintaining the attendance records becomes highly cumbersome. Biometric systems have reached a sufficiently advanced stage wherein they can now be deployed in systems without hampering portability. With the recent development of various cloud based computing and storage systems, data can be securely stored and retrieved whenever required. Primarily, fingerprints and iris images are considered to be the most reliable for use in biometric 5 systems. A system that records the attendance making use of biometric scanners and stores them securely over cloud in the form of Google Spreadsheet can help resolve issues. The system consists of a fingerprint scanner which is used for ascertaining a student's identity. If the fingerprint scanned matches with records present in the database, attendance is granted to the student by updating to the.Google.Spreadsheet.

CHAPTER-2

HARDWARE AND SOFTWARE COMPONENTS

HARDWARE REQUIRMENTS:

1.ARDUINO UNO BOARD:

The Arduino Uno contains a set of analog and digital pins that are input and output pins which are used to connect the board to other components. There are a total of fourteen I/O pins placed inboard in which six are analog input pins. The board has a USB connection that can be used to a power supply to the board. The board is used for electronics projects and used to design the circuit. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Arduino UNO is the best board to get started with electronics and coding. The UNO is the most used and documented board of the whole Arduino family. The Arduino UNO board is mostly used by the beginners that can use in electronics project and do programming in this board. The board has regular innovation and a bug fix in the design of the board to make the board suitable for the project's use. The Arduino UNO board is considered as the most used board and a standard board used by the rookie in their projects. There are various features that make the board suitable for the use and preferred over other Arduino products. The selection of right Arduino products is based on user requirement but the Arduino UNO is a standard board compared to Arduino products.

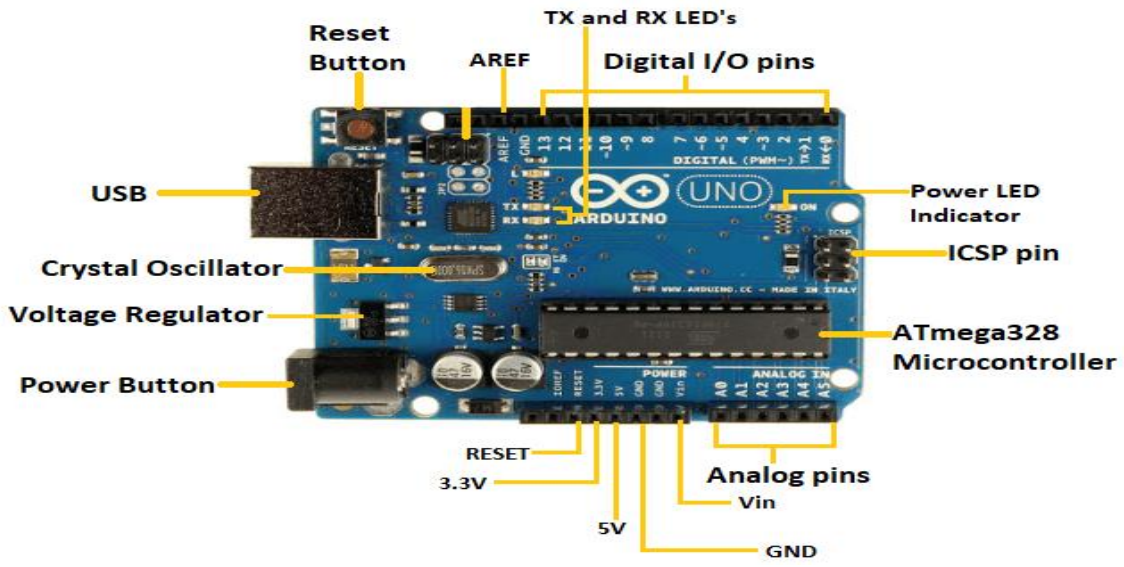


Fig.1 Arduino board

It has,

1. 14 Digital input/output pins,
2. 6 Analog inputs,
3. 16MHz Quartz crystal,
4. USB connector,
5. Power jack,
6. ICSP header, and
7. Reset button.

- ATmega328 Microcontroller- It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator**.**
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator-** The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's-** The successful flow of data is represented by the lighting of these LED's.
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.

Analog Pins- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

2.FINGERPRINT MODULE – R307:

R307 is an optical fingerprint scanner which is an upgraded version of R305. R307 has its own database which can store 1000 templates. Security level for R307 is from 1-5. This module has less false error rate, fast searching process, high speed processor, uses minutiae based algorithm to work with scanned fingerprints.

3.LCD DISPLAY:

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface. The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register.

An LCD character display is a unique type of display that can only output individual ASCII characters with fixed size. Using these individual characters then we can form a text. The number of the rectangular areas define the size of the LCD. The most popular LCD is the 16×2 LCD, which has two rows with 16 rectangular areas or

characters. It has 16 pins and each connected to arduino. We can adjust the contrast of the LCD by adjusting the voltage input at the *Vo* pin. We are using a potentiometer because in that way we can easily fine tune the contrast, by adjusting input voltage from 0 to 5V. With the I2C module, easy to connections to control the LCD.

There are 19 different functions in the LiquidCrystal library available for us to use. These functions do things like change the position of the text, move text across the screen, or make the display turn on or off. What follows is a short description of each function, and how to use it in a program.

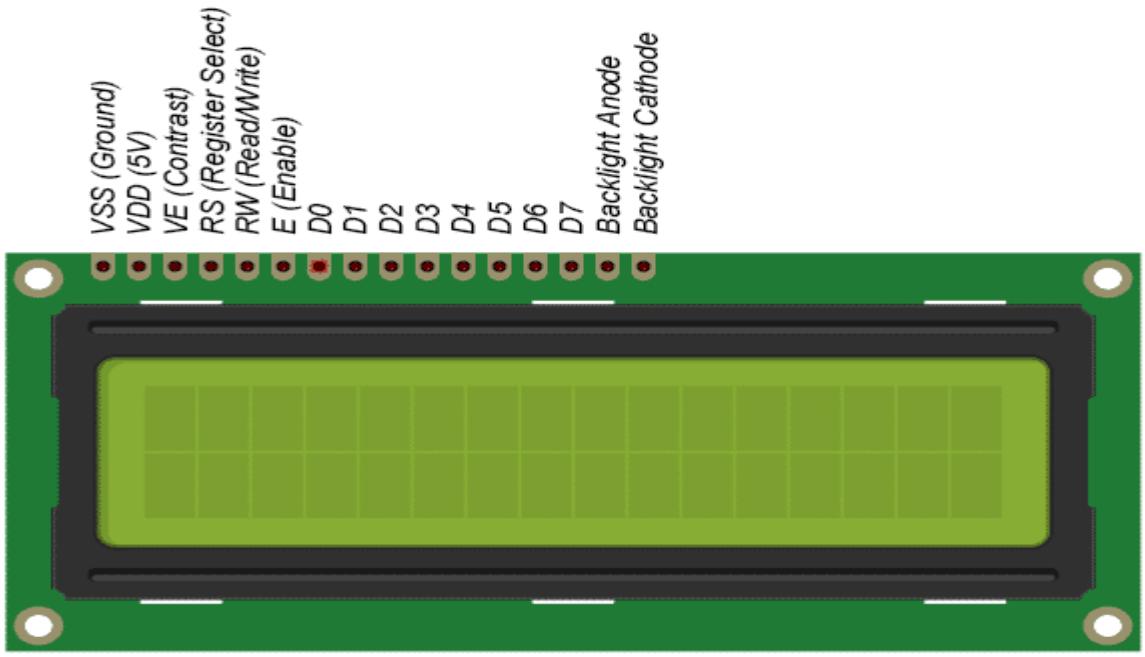
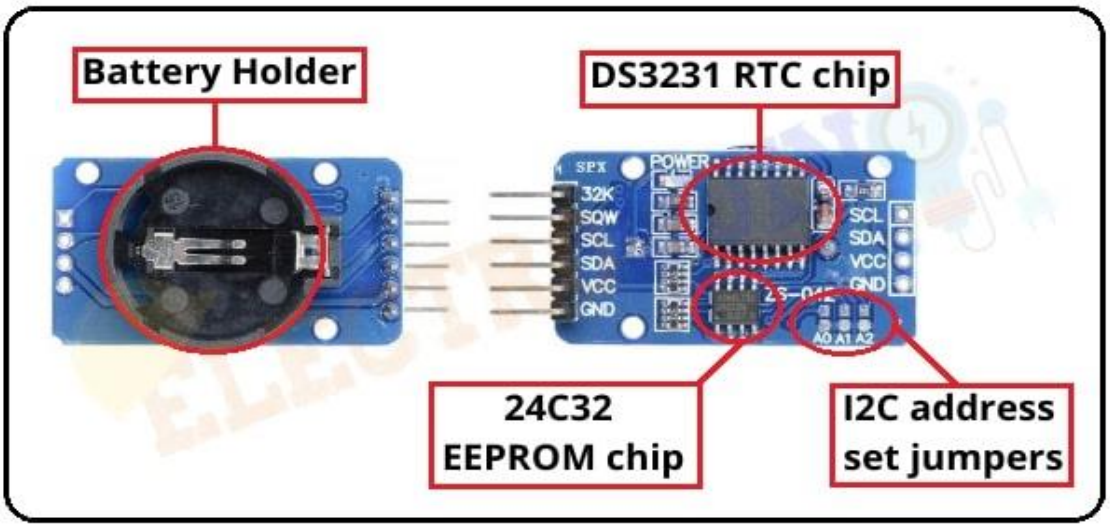


Fig.5 LCD display

4.RTC MODULE. (DS3231):

The DS3231 RTC module is a time tracking device that gives the current time and date. The word RTC is meant **R**eal **T**ime **C**lock. The RTC module made of clock chip DS3231. This module is generally used in computers, laptops, mobiles, embedded system applications devices, etc. to provide time and date. RTC module works on the I2C protocol. The module provides details such as second, minute, hour, day of the week, day of the month, month, and year including correction for leap year. One more interesting thing It can operate either in 12 Hour or in 24 Hour format. It's can be used in projects like containing data-logging, clock-building, time stamping, timers, and alarms.



5. BREAD BOARD:

A breadboard is a solderless construction base used for developing an electronic circuit and wiring for projects with microcontroller boards like Arduino. As common as it seems, it may be daunting when first getting started with using one.

The Arduino has multiple power and ground pins that you can connect to the power rails or other rows on a breadboard. We've seen the type of breadboard originating in the past, but for the modern-day solderless breadboard, it comes in different types; full-sized, Full+, half-sized, Half+, and mini.

With the size difference, there may be variances in how the different rows and columns of wire strips are connected though the general principle should remain the same

Although the main difference comes in size, there are different shapes and colour options available as well.

A breadboard consists of two areas called strips, and are often separated from the middle portion (commonly known as ravine).

Bus strips are mainly used for power supply connections

Terminal strips are mainly used for electrical components

Each strip consist of 5 pinholes, indicating that you only can connect up to 5 components in one particular section.

6. PUSH BUTTON:

A push-button or simply button is a simple switch mechanism to control some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, although many un-biased buttons (due to their physical nature) still require a spring to return to their unpushed state.

6. CONNECTING WIRES:

For connecting arduino to bread board we use Jumper wires instead normal connecting wires. Generally, jumpers are tiny metal connectors used to close or open a circuit part. They have two or more connection points, which regulate an electrical circuit board.

Their function is to configure the settings for computer peripherals, like the motherboard. Suppose your motherboard supported intrusion detection. A jumper can be set to enable or disable it.Jumper wires are electrical wires with connector pins at each end. They are used to connect two points in a circuit without soldering.You can use jumper wires to modify a circuit or diagnose problems in a circuit. Further, they are best used to bypass a part of the circuit that does not contain a resistor and is suspected to be bad.

Jumper wires come in three versions:

- Male-to-male jumper
- Male-to-female jumper
- Female-to-female jumper

And two types of head shapes: square head and round head.The difference between each is in the endpoint of the wire. Male ends have a pin protruding and can plug into things, while female ends do not but are also used for plugging.

Moreover, a male connector is referred to as a plug and has a solid pin for centre conduction. Meanwhile, a female connector is referred to as a jack and has a centre conductor with a hole in it to accept the male pin.Male-to-male jumper wires are the most common and what you will likely use most often.

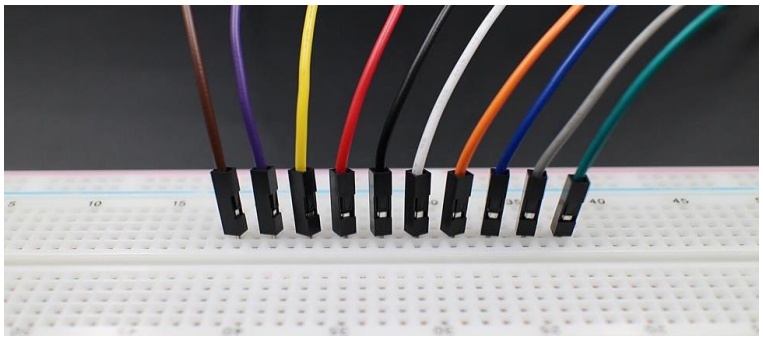


Fig.6 Bread board and connecting wires

7.USB CABLE:

The term USB stands for "Universal Serial Bus". USB cable assemblies are some of the most popular cable types available, used mostly to connect computers to peripheral devices such as cameras, camcorders, printers, scanners, and more.The USB cable standard allows for these advantages over serial cable types:

- USB cables are "Hot Pluggable", in other words you can connect and disconnect the cables while the computer is running without fear of freezing the computer
- USB cables are fast, transferring up to 480Mbps. Compare that to serial communication which transfers data at about 20Kbps
- USB cables carry power as well as signals. This allows for "USB powered" gadgets as well as recharging batteries in cameras and other USB peripherals
- USB cables are designed with several distinct connector types, making it easy to identify which plug goes into the computer and which plug goes into the peripheral device
- USB cables are a universal standard and are fairly easy to find and to afford

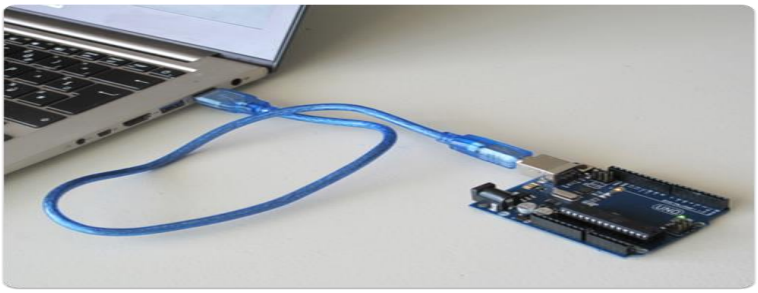


Fig.7 USB cable

8.Buzzer:

A buzzer or beeper is an audio signaling device,[1] which may be mechanical, electromechanical, or piezoelectric (*piezo* for short). Typical uses of buzzers and beepers include alarm devices, timers, train and confirmation of user input such as a mouse click or keystroke.

Piezoelectric buzzers, or piezo buzzers, as they are sometimes called, were invented by Japanese manufacturers and fitted into a wide array of products during the 1970s to 1980s. This advancement mainly came about because of cooperative efforts by Japanese manufacturing companies. In 1951, they established the Barium Titanate Application Research Committee, which allowed the companies to be "competitively cooperative" and bring about several piezoelectric innovations and inventions.

SOFTWARE REQUIRMENTS:

ARDUINO IDE:

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.To download IDE the following steps are involved

Step 1 − First you must have your Arduino board and a USB cable:

In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB

Step 2 − Download Arduino IDE Software:

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

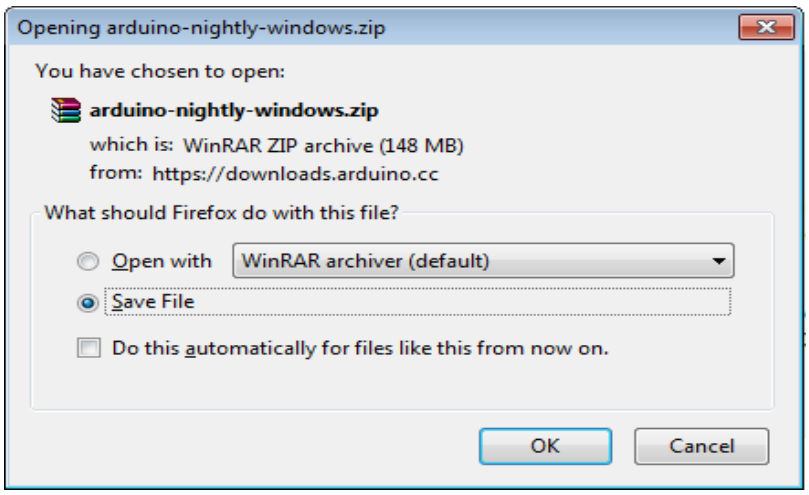


Fig.8 Zip file of Arduino IDE

Step 3 − Power up your board:

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection.

The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE:

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

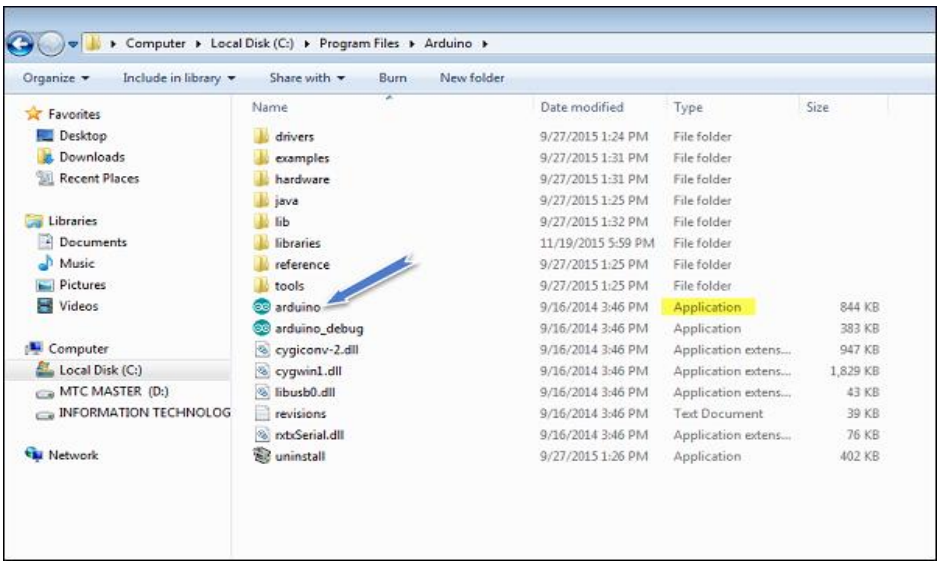


Fig.9 Arduino IDE software

Step 5 − Open your first project:

Once the software starts, you have two options −

- Create a new project.
- Open an existing project example.

To create a new project, select File → New.

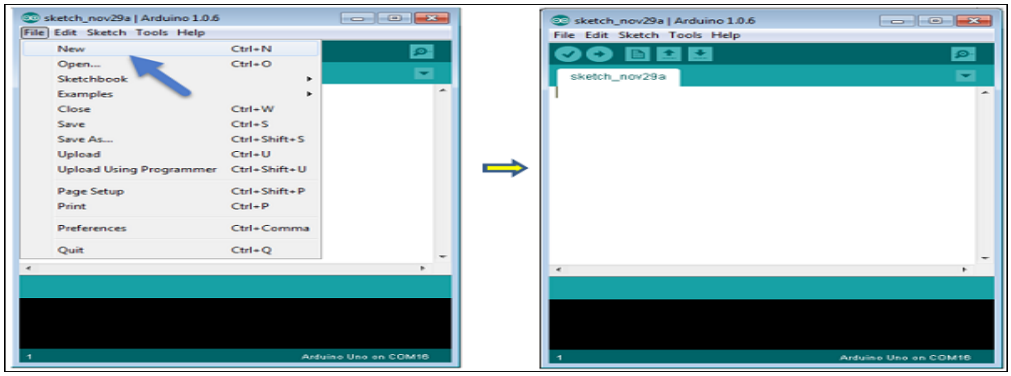


Fig .10 Arduino IDE Home page

Step 6 − Select your Arduino board:

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools → Board and select your board.

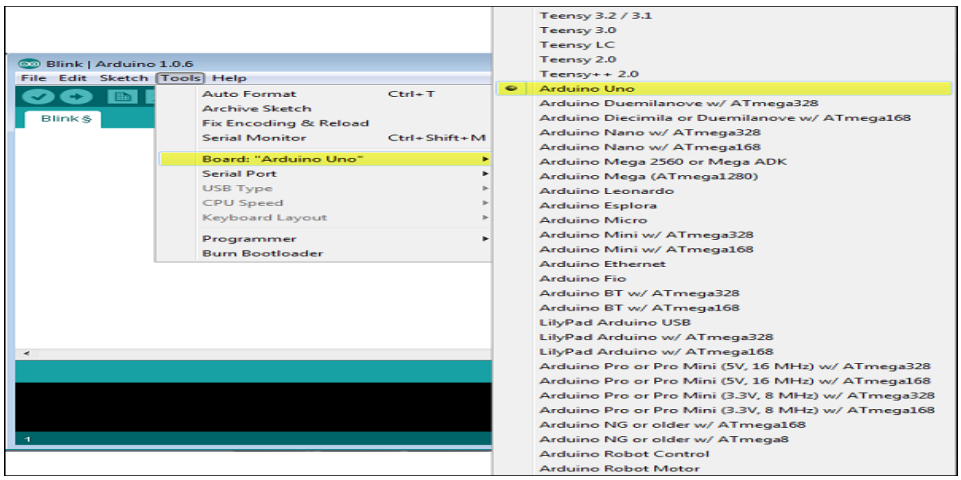


Fig.11 Selection of Arduino board

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

Step 7 − Select your serial port:

Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

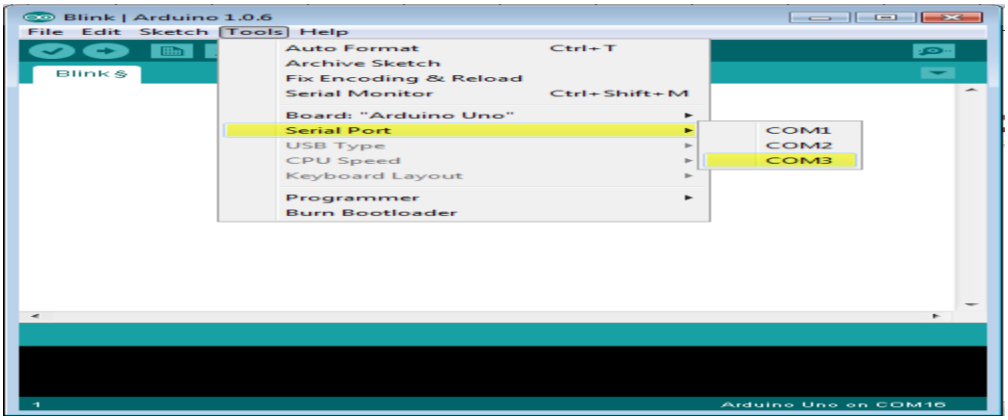


Fig.12 Selection of port in Arduino IDE

CHAPTER-3
WORKING AND IMPLEMENTATION

Intially check all the equipments whether they are working or not one by one.connect the circuit carefully.

The total system is divided in several sub systems, like

- R 307 interfacing
- Serial Monitor interfacing
- connect to Thingsboard
- LCD interfacing

The operation of the whole circuit is depending on every sections performance

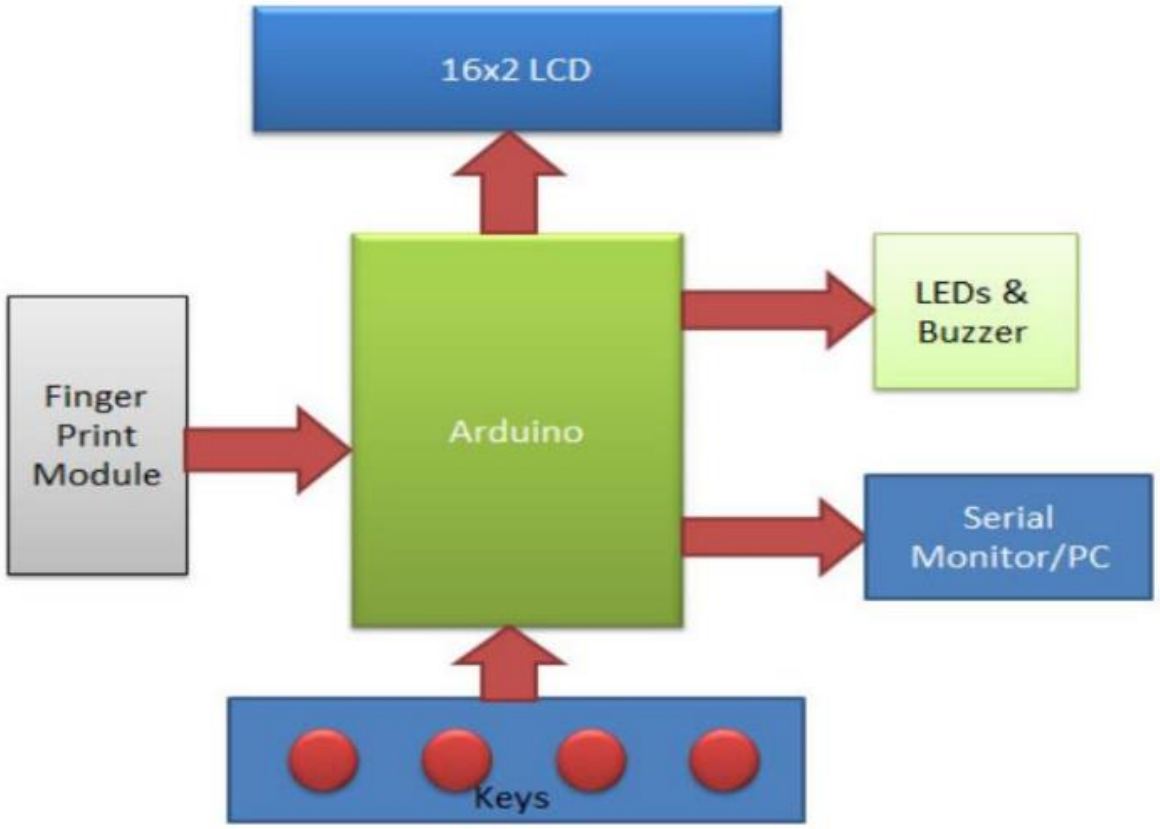


Fig.13 :Flow Diagram

The working of the Fingerprint Sensor Based Biometric Attendance System. 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.

ADAFRUIT FINGERPRINT SENSOR LIBRARY:

This is a library for our optical Fingerprint sensor,designed specifically to work with the Adafruit Fingerprint sensor .These displays use TTL Serial to communicate, 2 pins are required to interface Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!This is a library for our optical Fingerprint sensor

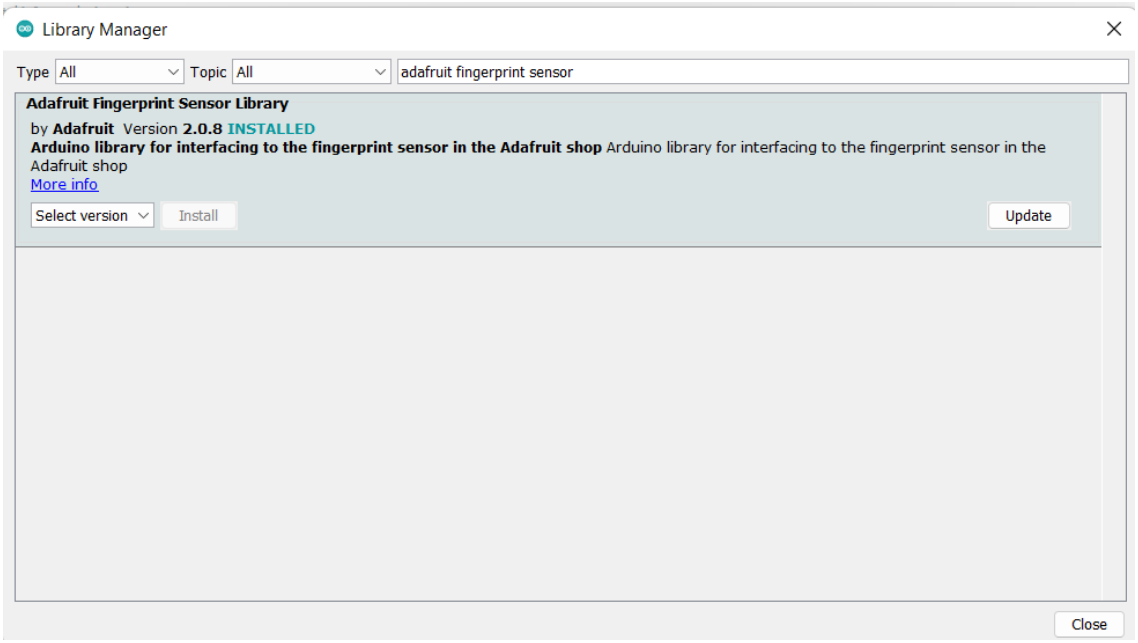


Fig.14 Installing Libraries

After it, we have to write code for downloading attendance data. After that, we have to write code for clearing attendance data from EEPROM. After it, we initiate finger print

module, showing welcome message over LCD and also initiated RTC module. After it, in loop function, we have read RTC time and displayed it on LCD After it, waiting for the finger print to take input and compare captured image ID with stored IDs. If a match occurs then proceed with next step. And checking enrol/del keys as well Given Function is used to taking finger print image and convert them into the template and save as well by selected ID into the finger print module memory. Given function is used for storing attendance time and date in the allotted slot of EEPROM Given function is used to fetching data from EEPROM and send to serial monitor.

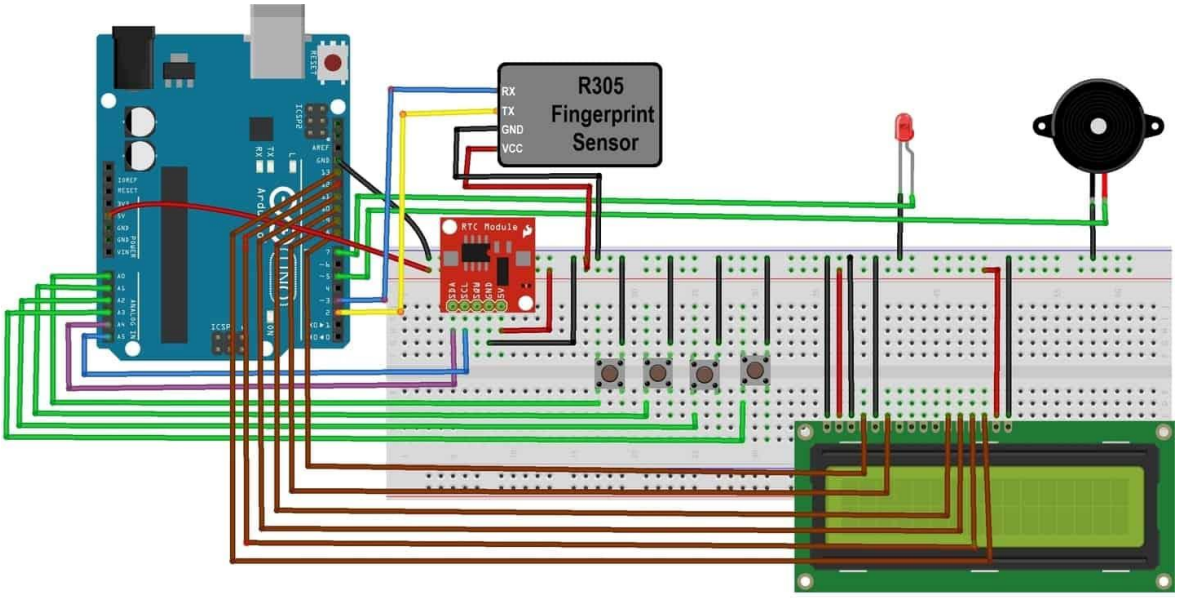


Fig.15:Circuit Diagram of Setup

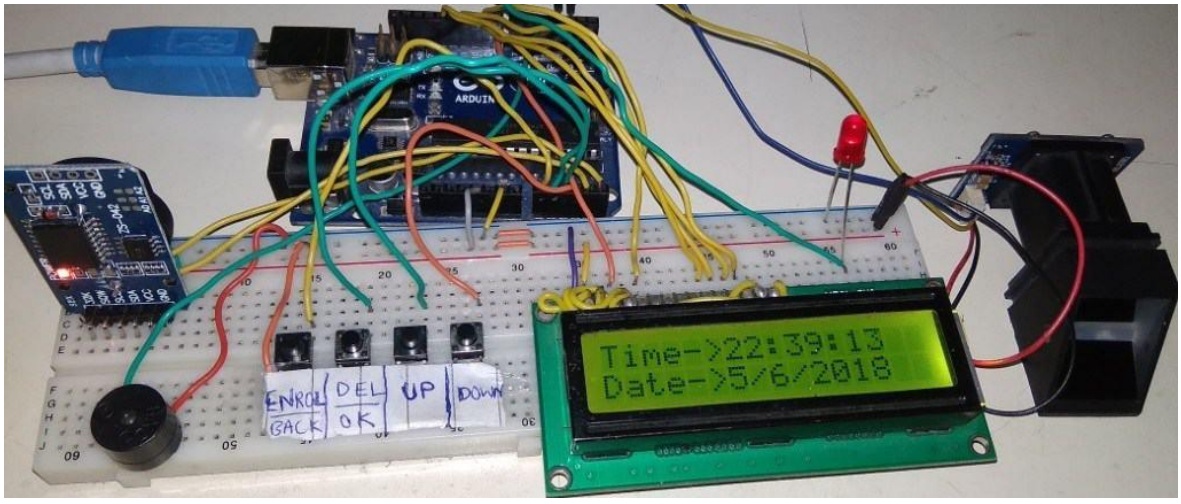


Fig. : Actual Setup of the entire system

CHAPTER-4

RESULT AND DISCUSSION

The experimental model was made following the circuit diagram and the desired results were obtained. Every time someone places his finger on the sensor the sensor reads the data and stores it in the cloud. Next time someone wants to check the fingerprint he/she places the finger on the sensor. The sensor reads the data and searches and cross-checks the data with stored fingerprints. If it matches with any of them then it displays the username, date and time. If not then says fingerprint doesn't match .That's how the whole system works.

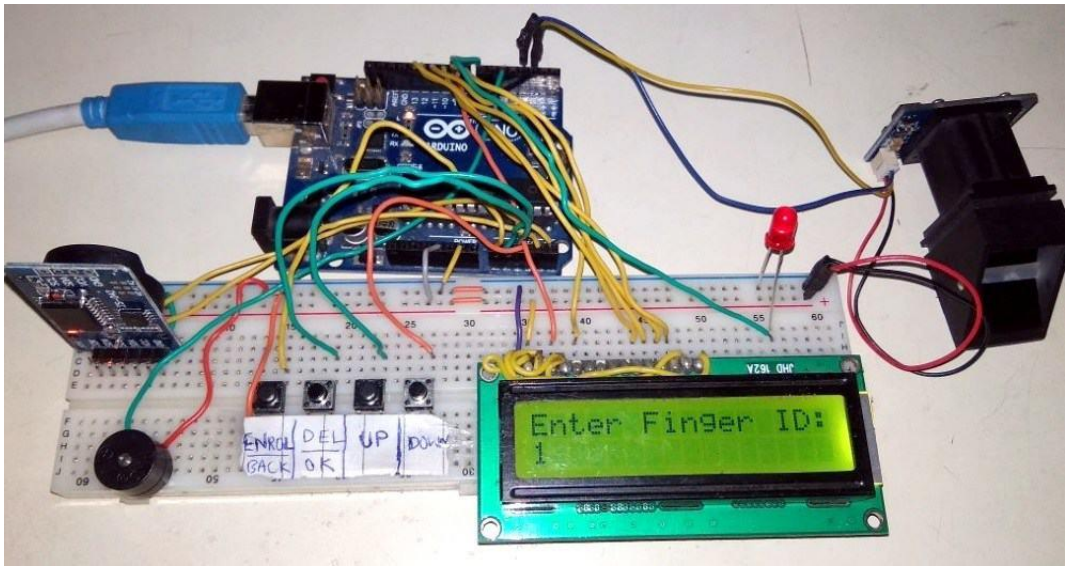


Fig.16

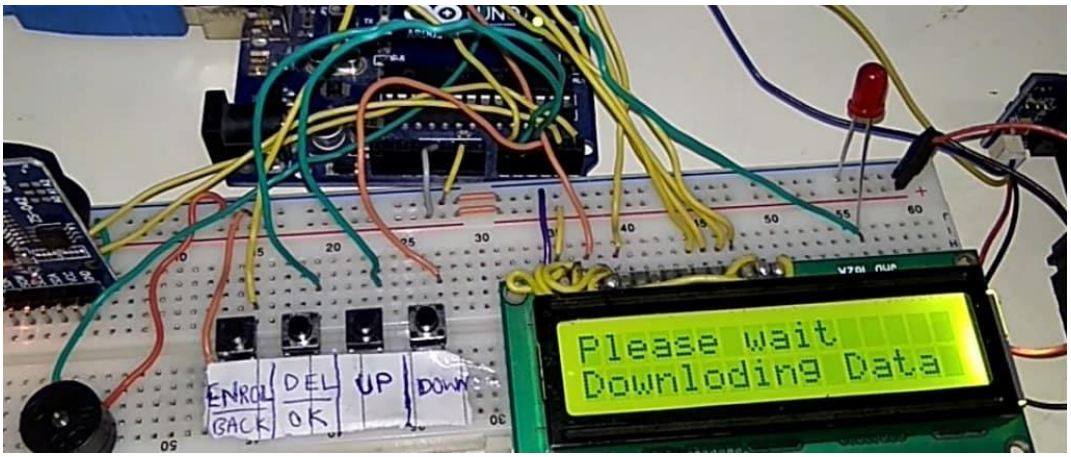
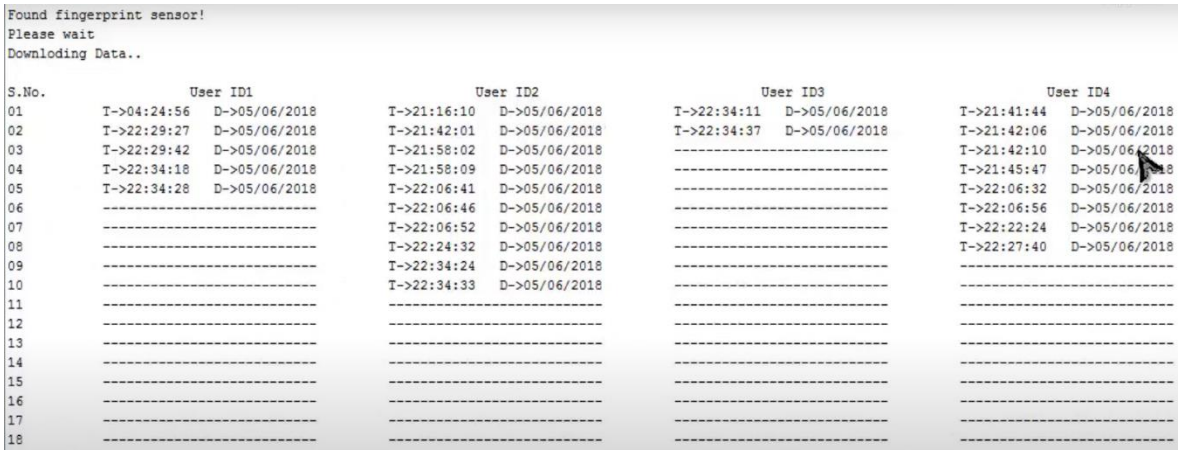


Fig.17

We use the serial monitor to check different patterns of fingerprint and different login and biometric usage for different IDs and hence the serial monitor gives us the different login time for different fingerprint patterns.



```
Found fingerprint sensor!
Please wait
Downloding Data..

S.No.          User ID1                  User ID2                  User ID3                  User ID4
01        T->04:24:56  D->05/06/2018   T->21:16:10  D->05/06/2018   T->22:34:11  D->05/06/2018   T->21:41:44  D->05/06/2018
02        T->22:29:27  D->05/06/2018   T->21:42:01  D->05/06/2018   T->22:34:37  D->05/06/2018   T->21:42:06  D->05/06/2018
03        T->22:29:42  D->05/06/2018   T->21:58:02  D->05/06/2018   ---------------------------   T->21:42:10  D->05/06/2018
04        T->22:34:18  D->05/06/2018   T->21:58:09  D->05/06/2018   ---------------------------   T->21:45:47  D->05/06/2018
05        T->22:34:28  D->05/06/2018   T->22:06:41  D->05/06/2018   ---------------------------   T->22:06:32  D->05/06/2018
06        ---------------------------   T->22:06:46  D->05/06/2018   ---------------------------   T->22:06:56  D->05/06/2018
07        ---------------------------   T->22:06:52  D->05/06/2018   ---------------------------   T->22:22:24  D->05/06/2018
08        ---------------------------   T->22:24:32  D->05/06/2018   ---------------------------   T->22:27:40  D->05/06/2018
09        ---------------------------   T->22:34:24  D->05/06/2018   ---------------------------   ---------------------------
10        ---------------------------   T->22:34:33  D->05/06/2018   ---------------------------   ---------------------------
11        ---------------------------   ---------------------------   ---------------------------   ---------------------------
12        ---------------------------   ---------------------------   ---------------------------   ---------------------------
13        ---------------------------   ---------------------------   ---------------------------   ---------------------------
14        ---------------------------   ---------------------------   ---------------------------   ---------------------------
15        ---------------------------   ---------------------------   ---------------------------   ---------------------------
16        ---------------------------   ---------------------------   ---------------------------   ---------------------------
17        ---------------------------   ---------------------------   ---------------------------   ---------------------------
18        ---------------------------   ---------------------------   ---------------------------   ---------------------------
```

Fig.18: Monitoring the result

CHAPTER-5

SUMMARY AND FUTURE SCOPE

SUMMARY:

Here we have developed a Biometric fingerprint based attendance system using Arduino. In this project we have used R307 fingerprint sensor which reads the Fingerprint and stores in the form of digital data. A buzzer is activated and LED blinks then LCD panel shows that data is stored along with username, date and time. Working of this fingerprint attendance system project is fairly simple. First of all, the user needs to enroll fingerprints of the user with the help of push buttons. To do this, user need to press ENROLL key and then LCD asks for entering ID for the fingerprint to save it in memory by ID name. So now user needs to enter ID by using UP/DOWN keys. After selecting ID, user needs to press OK key (DEL key). Now LCD will ask to place finger over the fingerprint module. Now user needs to place his finger over finger print module and then the module takes finger image. Now the LCD will say to remove finger from fingerprint module, and again ask to place finger again. Now user needs to put his finger again and module takes an image and convert it into templates and stores it by selected ID into the finger print module's memory. Now the user will be registered and he/she can feed attendance by putting their finger over fingerprint module. By the same method, all the users will be registered into the system. Now if the user wants to remove or delete any of the stored ID or fingerprint, then he/she need to press DEL key. Once delete key is pressed LCD will ask to select ID that need to be deleted. Now user needs to select ID and press OK key (same DEL key). Now LCD will let you know that fingerprint has been deleted successfully. The traditional process of manually taking and maintaining student attendance is highly inefficient and time consuming. The attendance monitoring system based on biometric authentication has a potential to streamline the whole process. An Internet of Things (IoT) based portable biometric attendance system can prove to be of great value to educational institutions in this regard as it proves to be highly efficient and secure. The cost involved in making this system is quite less, when compared to conventional biometric attendance system. The use of cloud computing to store the attendance records makes all the data easy to access and retrieve as end when required by the teachers. The use of fingerprint scanner ensures the reliability of the attendance record. The system, due to its lack of complexity, proves to be easy to use and user friendly. 56 The system can be improved by encasing it in a plastic covering. This would make it more compact and easy to use in a classroom setting. The system

can be configured to enable lecturewise attendance taking. It can further be improved to automatically calculate attendance percentages of students and intimate the teachers if a student's attendance is below a certain percentage. It can also be modified to fit the corporate environment. The traditional process of manually taking and maintaining student attendance is highly inefficient and time consuming. The attendance monitoring system based on biometric authentication has a potential to streamline the whole process. An Internet of Things (IoT) based portable biometric attendance system can prove to be of great value to educational institutions in this regard as it proves to be highly efficient and secure. The cost involved in making this system is quite less, when compared to conventional biometric attendance system. The use of cloud computing to store the attendance records makes all the data easy to access and retrieve as end when required by the teachers. The use of fingerprint scanner ensures the reliability of the attendance record. The system, due to its lack of complexity, proves to be easy to use and user friendly.

Future Scope of the Project:

Biometric attendance system using Arduino uno is very useful for many industries and offices. It's easy, cost effective and works very well. Hence the future scope of this technology is wide spread and quite essential in both domestic and industrial applications. This system could be connected to a server and daily records stored and maintained. The flash of the R-305 module can store up to 250 image scans, implying that such a system can easily accommodate the needs of a classroom.

REFERENCES

[1] Biometric Systems - Technology, Design and Performance Evaluation By - James Wayman, AnilJain, Davide Maltoni and Dario Maio (Eds). Springer Publications.

[2] On Board Manual for organization to install Aadhar – enabled biometric attendance system. Issued from : National Informatics Centre (NIC). Govt. Of India. Nov, 2014.

[3] Open source site for code help, www.Circuitoday.com

[4] Connection assistance from www.arduino.com

[5] en.wikipedia.org/wiki/Arduino

[6] FINGERPRINT BASED LICENSE CHECKING FOR AUTO-MOBILES. IEEE Conference Paper. By - J.Angeline Rubella, M.Suganya B.Santhosh Kumar, K.R.Gowdham, M.Ranjithkumar, K.Senathipathi.

[7] Open source code assistance from www.stackexchange.com

[8] A Robust Embedded Biometric Authentication System Based on Fingerprint and Chaotic Encryption. Elsevier Journel. By-M.A. Murillo-Escobar and C. CruzHernández

APPENDIX (CODE)

```
#include "Adafruit_Fingerprint.h" //fingerprint library header file
#include<EEPROM.h> //command for storing data
#include<LiquidCrystal.h> //lcd header file
LiquidCrystal lcd(8,9,10,11,12,13);
#include <SoftwareSerial.h>
SoftwareSerial fingerPrint(2, 3);
#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("  ");
}
```