IOT BASED BIOMETRIC ATTENDANCE SYSTEM USING ARDUINO.

REPORT OF PROJECT SUBMITTED FOR PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
ELECTRICAL ENGINEERING

By

NAME OF THE STUDENT	REGISTRATION NO.	UNIVERSITY ROLL NO.
SANKHA SUDHA GHOSH	**171170120063**	**11701617005**
SWAPNADIP MONI	**161170110370**	**11701616016**
UPAL GANGOPADHYAY	**161170110374**	**11701616012**
MONOJEET DAS	**17117012055**	**11701617013**

UNDER THE SUPERVISION OF

MR. DIPANKAR SANTRA
ASSOCIATE PROFESSOR,
DEPARTMENT OF ELECTRICAL ENGINEERING.
RCC INSTITUTE OF INFORMATION TECHNOLOGY.

AT

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL
Maulana Abul Kalam Azad University of Technology (MAKAUT)

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
GROUND FLOOR, NEW BUILDING,
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL
PHONE: 033-2323-2463-154, FAX: 033-2323-4668
Email: hodeercciit@gmail.com, Website: http://www.rcciit.org/academic/ee.aspx

CERTIFICATE

To whom it may concern

This is to certify that the project work entitled, **IOT based Biometric Attendance system using Arduino** is the bonafide work carried out by:

1. **SANKHA SUDHA GHOSH. (11701617005)**
2. **SWAPNADIP MONI. (11701616016)**
3. **UPAL GANGOPADHYAY. (11701616012)**
4. **MONOJEET DAS. (11701617013)**

a student of B.Tech in the Dept. of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India, during the academic year 2019-20, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electrical Engineering and that this project has not submitted previously for the award of any other degree, diploma and fellowship.

____

Mr. Dipankar Santra.
Associate Professor.
Department of Electrical Engineering.
RCC Institute of Information Technology.

____

Dr. Debashish Mondal
HOD, Dept of Electrical Engineering.
RCC Institute of Information Technology.

____

Signature of the External Examiner

ACKNOWLEDGEMENT

It is our great fortune that we have got opportunity to carry out this project work under the supervision of **Mr. Dipankar Santra** in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance and limitless encouragement.

We wish to convey our gratitude to Prof. (Dr.) Debashish Mondal, HOD, Department of Electrical Engineering, RCCIIT and to the authority of RCCIIT for providing all kinds of infrastructural facility towards the research work.
Thanks to the fellow members of our group for working as a team.

1. SANKHA SUDHA GHOSH. (11701617005)
2. SWAPNADIP MONI. (11701616016)
3. UPAL GANGOPADHYAY. (11701616012)
4. MONOJEET DAS. (11701617013)

DEPARTMENT OF ELECTRICAL ENGINEERING.
B.TECH. 4TH YEAR. 7TH SEMESTER.
RCC INSTITUTE OF INFORMATION TECHNOLOGY.
BATCH OF 2016-2020.

To,

The Head of the Department

Department of Electrical Engineering

RCC Institute of Information Technology

Canal South Road. Beliaghata,

Kolkata-700015

Respected Sir,

In accordance with the requirements of the degree of Bachelor of Technology in the Department of Electrical Engineering, RCC Institute of Information Technology, We present the following thesis entitled "**IOT based Biometric Attendance system using Arduino**". This work was performed under the valuable guidance of Mr. Dipankar Santra, Associate Professor in the Dept. of Electrical Engineering.

We declare that the thesis submitted is our own, expected as acknowledge in the test and reference and has not been previously submitted for a degree in any other Institution.

Yours Sincerely,

1. SANKHA SUDHA GHOSH.	**(11701617005)**
2. SWAPNADIP MONI.	**(11701616016)**
3. UPAL GANGOPADHYAY.	**(11701616012)**
4. MONOJEET DAS.	**(11701617013)**

TABLE OF CONTENTS

Topic	Page No.
List of figures	i
List of table	ii
Abbreviations and acronyms	iii
Abstract	1
Chapter 1 (Introduction)	
1.1 Introduction.	4
1.2 R307/R305 Fingerprint sensor module.	7
1.3 Working principle of fingerprint sensor.	8
1.4 Preparing your Thingsboard account.	9
1.5 Preparing the ESP8266-01 Wi-fi Module.	11
1.6 Overview and benefits of the project.	12
1.7 Organization of thesis.	14
Chapter 2 (Literature Review)	16
Chapter 3 (Theory)	
3.1 R307 Fingerprint Sensor.	23
3.2 Operation Principle.	23
3.3 Technical Parameters.	24
3.4 Features.	25
3.5 Interface Description.	28
3.6 Working of fingerprint sensor and push buttons.	28
3.7 Implementation .	29
3.8 Pin Configuration.	31
3.9 R307 Memory and Registers.	33
3.10 R307 Communication Protocol.	35
Chapter 4 (Hardware Modeling)	
4.0 Prototype Modeling.	41

4.1 Main features of the prototype. 41

4.2 Overview of the Project. 41

4.3 Photographs of the prototype. 42

4.4 Step by step operation of the prototype. 42

4.5 Components Required. 43

4.6 Cost estimation of the prototype. 44

4.7 *Hardware Connection:*

4.7.1 Prototype hardware connection. 45

4.7.2 Detailed Hardware Description. 46

Chapter 5 (Logic & Operations)

5.1 Introduction. 49

5.2.Flow Diagram. 49

5.3 Principle & Operations. 49

5.4 Advantages of the project. 50

5.5 Disadvantages of the project. 50

5.6 Coding Logic. 51

5.7 Datasheet from Serial Monitor. 51

5.8 Circuit Diagram. 52

5.9 Photograph of the Prototype. 53

Chapter 6 (Conclusion & Future Scope)

6.1 Conclusion. 55

6.2 Results. 56

6.3 Future work & Scope. 56

6.3.1 Different ways of making and usage of biometric attendance system 57

Chapter 7 (References) 62

Appendix A (Hardware Description) 64

Appendix B (Software Coding) 77

Appendix C (Data Sheets) 94

<u>**LIST OF FIGURES:**</u>

Sl. No. ***Figures*** ***Page No.***

Sl. No.	Figures	Page No.
1.	Working of Fingerprint based Attendance System.	6
2.	Fingerprint capture technique.	9
3.	Fingerprint Sensor 2D model.	25
4.	Biometric Acceptance by Unit.	26
5.	Cross Section of Fingerprint Scanner.	27
6.	Pin Configuration.	31
7.	R307 Fingerprint Scanner PCB Side.	32
8.	R307 Schematic Diagram.	33
9.	R307 UART Frame Format.	36
10.	R307 commands and byte codes.	37
11.	Overall Setup of prototype.	42
12.	Overall Circuit Diagram for Setup.	45
13.	Hardware Setup Diagram.	46
14.	Circuit Diagram of the Setup.	47
15.	Flowchart of the System.	49
16.	Circuit Diagram of Setup.	52
17.	Actual Setup of the entire system.	53
18.	Arduino Uno.	65
19.	ATmega328/P and Arduino Uno Pin Mapping.	66
20.	R307 Fingerprint Sensor.	68
21.	LCD setup and pins.	69
22.	16*2 LCD panel.	70
23.	2D Model Of A 16*2 LCD Display.	70
24.	ESP8266 Wifi Module.	71
25.	Push Button.	72
26.	Resistor.	72
27.	Color Code Of Resistor.	73
28.	Piezo buzzer.	73
29.	Breadboard	74
30.	DS3231 RTC Chip.	74
31.	DS3231 RTC Module Pinouts	75
32.	Serial Monitor for DS3231 code.	79
33.	Pin Configuration of ATmega328/P.	97
34.	Block Diagram of ATmega328/P.	98

<u>LIST OF TABLES:</u>

Sl. No.	*Tables*	*Page No.*
1.	Difference Between R305 (Old) and R307 (New).	08
2.	Pinouts.	31
3.	System Configuration Register.	34
4.	List of Components.	43
5.	Cost Estimation table	44
6.	Data sheet of collected data on Arduino Serial Monitor.	52
7.	Comparative analysis of different systems.	60
8.	Data Sheet from Serial Monitor.	94

Abbreviations and Acronyms:

LCD –Liquid Crystal Display.

RTC- Real-Time Clock.

IC - Integrated Circuit.

PCB – Printed Circuit Board.

μC – Micro Controller.

IOT- Internet of Things.

RFID- Radio Frequency Identification.

PC- Personal Computer.

UART- Universal Asynchronous receiver- transmitter.

USB- Universal Serial Bus.

TIR- Total Internal Reflection.

LED- Light Emitting Diode.

FTIR- Frustrated Total Internal Reflection.

FPS- Frames per second.

AT- Attention.

ESP- Electronic Stability Program.

TTL- Transistor-Transistor Logic.

FAR -False Acceptance Rate.

FRR-False Recognition Rate.

DEL- Delete.

ID- Individual Details.

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.
By using the fingerprint sensor, the system will become more secure for the users.

Biometric student attendance system increases the efficiency of the process of taking student

attendance. This presents a simple and portable approach to student attendance in the form of an Internet of Things (IOT) based system that records the attendance using fingerprint based biometric scanner and stores them securely over cloud. This system aims to automate the cumbersome process of manually taking and storing student attendance records. It will also prevent proxy attendance, thus increasing the reliability of attendance records. The records are securely stored and can be reliably retrieved whenever required by the teacher.

Proper attendance recording and management has become important in today's world as attendance and achievement go hand in hand. Attendance is one of the work ethics valued by employers. Most of the educational institutions and government organizations in developing countries still use paper based attendance method for maintaining the attendance records. There is a need to replace these traditional methods of attendance recording with biometric attendance system. The unique nature of fingerprint makes it ideal for use in attendance management systems. Besides being secure, Fingerprint based attendance system will also be environment friendly. Fingerprint matching is widely used in forensics for a long time. It can also be used in applications such as identity management and access control. This review incorporates the problems of attendance systems presently in use, working of a typical fingerprint based attendance system, study of different systems, their advantages, disadvantages and comparison based upon important parameters.

CHAPTER 1

(Introduction)

INTRODUCTION

1.1 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 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.

IoT based Biometric Attendance system using Arduino and Thingsboard and Adafruit Fingerprint Sensor Library.

Few years back if you were to tell someone that the Geyser and bedroom lights in your home are connected to internet, they would be baffled and might even criticize it as over-engineered products. But today with the advent of IoT, Smart cities etc the idea no longer sounds strange, we have devices around us that have become smarter by being able to communicate with the internet.
In this project our aim is to leverage this IoT into the boring attendance system to make it smart and more effective. Most conventional attendance systems available today store the information over a micro SD card and have to be connected to software via a computer to access the information. Here, we will build a biometric attendance system using Arduino that scans for finger print and on successful identification of the person it will log the information to a cloud platform like ThingsBoard by using the ESP8266 Wi-Fi module. This information can then be displayed in the dashboard of ThingsBoard making it available for the required authorities to view and analysis information over the internet without having any direct physical access to the hardware. However the conventional Attendance system without involving IoT can also be built by following the link and Finger print sensor can be further used for many other biometric applications like Voting Machine, Security system etc.

Presently, attendance of students in most institutes is taken by the teacher on paper based attendance registers. There are various disadvantages to this approach such as data is not available for analysis because paper based registers are not uploaded to a centralized system, time taken for data collection reduces the effective lecture time and fake attendance by students. Some universities also use wall mounted RFID swipe card systems. RFID (Radio Frequency

Identification) is a wireless technology which uses electromagnetic waves for communication between RFID reader and RFID tag. Though better than paper based systems, RFID based systems also have certain problems such as the system is complex, costly and absent student's card can be swiped by other students.

Biometric techniques can be used to solve these problems. Biometric is derived from two Greek roots "bios" meaning life and "metrics" meaning measurement. Biometric technology identifies a person uniquely based on his/her characteristics which can be physiological or behavioral. Among the various biometric techniques, there are nine main biometric techniques which are widely used. These include fingerprint, face, hand vein, hand geometry, iris, retinal pattern, voice print, signature, and facial thermo grams. Comparison of different biometric techniques has shown that fingerprint biometric is a reliable, mature and legally accepted biometric technique. Therefore, Fingerprint based attendance system can be used for identification of large number of students in universities and also for attendance monitoring of employees in organizations. There are two stages of working of these systems **1)** Enrolment of fingerprints. **2)** Matching.of.Fingerprints.

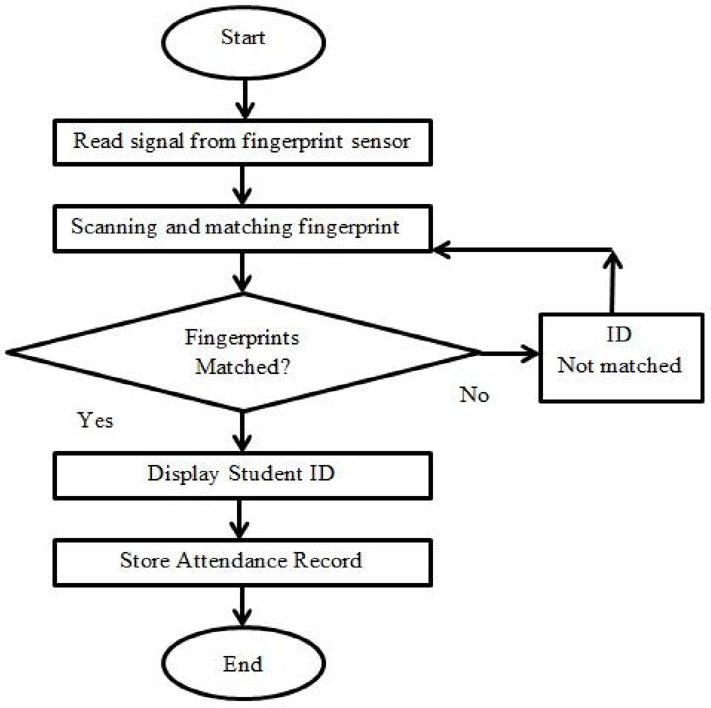


Figure 1: Working of Fingerprint based Attendance System

Pre-requisites

We will be writing two Arduino scripts for this program. One for the ESP8266-01 Module and the other is for Arduino UNO. The reason is that Arduino was not able to handle both GT511C3 sensor and ESP8266 through AT commands via software serial. Hence we will write two codes, one for Arduino in which it will communicate with the FPS and send the obtained values via software serial to ESP8266. The other code will be written for ESP8266 which will enable the module to be connected to the Thingsboard server and then will receive the values from Arduino through serial communication to update them on Thingsboard Dashboard.

1.2 R307/R305 FINGERPRINT SENSOR MODULE

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

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

Features:

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

Fingerprints are one of the many unique biometric signatures which we can use to identify people very accurately. But just by holding someone's hand and staring at their fingers can't be practical [grins]; we're not good at it. But computers are good at recognizing and matching patterns very fast and accurately. Before we can process a fingerprint pattern with a computer, we must "capture" it.

R307 Fingerprint Module consists of optical fingerprint sensor, high-speed DSP processor, high-performance fingerprint alignment algorithm, high-capacity FLASH chips and other hardware and software composition, stable performance, simple structure, with fingerprint entry, image processing, fingerprint matching, search and template storage and other functions.

Table:01

Difference Between R305 (Old) and R307 (New)

	R305	**R307**
Storage Capacity(Fingerprints)	250	1000
3.3V Operation	No	Yes
USB Operation	No	Yes
Finger Detect Output	No	Yes

These are the reasons why we used the R307 module over the R305 module.

1.3 Working principle of fingerprint sensor

The skin on the palms of our hands have a special pattern called friction ridges that help us grab things effectively without slipping. These patterns consist of ridges and valleys arranged in certain configurations and is unique for each individual. Our finger tips also have them as you can see from the above image. When a finger comes in contact with a surface, the ridges make strong contact with the surface. When we strongly grab something, the moisture, oil, dirt and dead skin cells on our finger can attach to the surface of the material, leaving an impression we call a fingerprint. Various forensic methods involving the use chemicals are used to extract such fingerprints from crime scenes and are called latent fingerprints. But an optical fingerprint scanner works a bit differently.

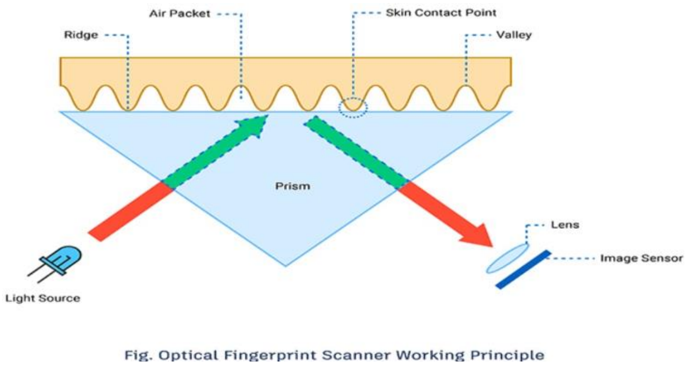


Figure 2: Fingerprint capture technique

An optical fingerprint scanner works based on the principle of **Total Internal Reflection** (TIR). In an optical fingerprint scanner, a glass prism is used to facilitate TIR. Light from an LED (usually blue color) is allowed to enter through one face of the prism at a certain angle for the TIR to occur. The reflected light exits the prism through the other face where a lens and an image sensor or the camera or reflector inside it (essentially camera) are placed.

When there's no finger on the prism, the light will be completely reflected off from the surface, producing a plain image in the image sensor. When TIR occurs, a small amount of light leaked to the external medium and it is called the **Evanescent Wave**. Materials with different *refractive indexes* (RI) interact with the evanescent wave differently. When we touch a glass surface, only the ridges make good contact with it. The valleys remain separated from the surface by air packets. Our skin and air have different RIs and thus affect the evanescent field differently. This effect is called **Frustrated Total Internal Reflection** (FTIR). This effect alters the intensities of the internally reflected light and is detected by the image sensor. The image sensor data is processed to produce a high contrast image which will be the digital version of the fingerprint.

In capacitive sensors, which are more accurate and less bulky, there's no light involved. Instead, an array of capacitive sensors are arranged on the surface of the sensor and allowed to come in contact with the finger. The ridges and air packets affect the capacitive sensors differently. The data from the sensor array can be used to generate a digital image of the fingerprint.

1.4 Preparing your Thingsboard account

There are many open source cloud platforms available today for IoT project integrations. Each platform has its own specialty, for our application I was looking for something that is good in data logging and visualization and found Thingsboard.io to suit that purpose. So let's start by setting up the Thingsboard account first. Get into thingboard.io and click on “TRY IT NOW” and then under the community edition (because it’s free) click on Live Demo. You will be taken to a sign-up page, it is a simple procedure in which you have to link and confirm your Email ID and once done you will be taken to the home page for further enrolment of the process.

On ThingsBoard we have two important terms, Assets and Devices. You can think of assets as buildings, warehouses, industry, farmland etc and devices as the sensor or devices present in that particular asset. So every asset will have one or more devices in it based on the project, here we will have one asset and one device in our asset.

Creating an Asset on Thingsboard

Let’s start by creating our first asset, by clicking on assets on our left panel and we notice all the assets related to our account, we might have none or some example assets which we can ignore. To create an asset click on the add icon on the bottom right corner of the screen which creates a pop-up promoting for Name, Asset type and Description. We give any name and type.

Once the details are entered just click on Add and our asset will be created. Note that I have named by asset as “RCCIIT EE Department”. Once the asset is created we can notice it appearing on the window.

Adding a Device to the Asset

Now that we have created an asset we should add a device to it. To do so click on the device tab on the left panel and then click on add icon on the bottom right corner of the screen. You will get a similar pop up in which you to name the device I have name mine as FPS main gate and device type as default.

Click on add and then you will find the device being created on the panel, click on the device that we just created and you will notice a new panel sliding in from the right. This panel will have all the information about the device, just click on copy access token to get the token value of our device, we will need this value in our Arduino program to send or receive values to this device.

Creating Device Relation for Asset

After creating our asset and device, go back to our asset tab and click on the asset that we just created. My asset is named as “RCCIIT EE Department”. This will slide in a pop-up from the right, now select relations tab in the new pop-up and over outbound relations click on add (+) symbol to get the following pop-up

Select entity type as Device and enter the name of the device that we created earlier. My device name was “RCC FPS Main Gate” which I have entered above. Finally click on Add button to add then the device relationship to asset to the network for the relation coordination.

1.5 Preparing the ESP8266-01

The ESP8266 should be operated both in AT command mode and Programming mode for this project. We can use a LM317 to regulate 3.3V for powering the ESP8266 module and connect the Tx Rx pins to FTDI board.

The toggle switch can be used to toggle the ESP8266 between AT command mode and Programming mode and push button can be pressed to reset the module. Note that the ESP8266 has to be reset every time before a code is being uploaded to it. If you are confused on how to do it you can refer to the basics of ESP8266, including how to use ESP8266 in AT command mode and Flash firmware on it.

This circuit will only be used to upload the program to ESP8266, later we will replace the FTDI board with Arduino UNO in our final set-up.

Programming ESP8266 with Arduino IDE for sending data to ThingsBoard

We have to program the ESP8266 to connect to our Wi-Fi router and sync in with our ThingsBoard device using the token address that we obtained earlier. Once that is done it should actively look for details coming from the Arduino board through its serial pins and if it gets a data it should phrase the information and send it to the Thingsboard dashboard. Before proceeding with the program makes sure you have already installed the required board details in your Arduino IDE using board manager to program ESP8266 with Arduino also install the following libraries using Sketch -> Include Library -> Manage Library. Just search for the required library and click on install.

- **PubSubClient by Nick O’Leary.**
- **WiFiEsp by bportaluri**

Once the IDE is ready we can begin the program by adding the required libraries and providing the Wifi credential and password with the Token value that we obtained earlier. Then create a Wi-Fi client that connects to the Thingsboard demo page.

Important: Make sure you change the TOKEN and Wi-Fi credentials according to your device.

Inside the setup function we will begin the serial communication at 9600 baud rate and initialize the Wi-Fi module to connect to the Wi-Fi router. Finally we will connect our Wi-Fi client to the ThingsBoard server.

Inside the loop function we will constantly check if the connection to server is active, if not we will try a reconnection using the reconnect function. If the client connection is successful the program will check for incoming data from the Serial monitor, if data is received it will converted to string and stored in the variable called Name. This variable will then be sent to the ThingsBoard server in json format using the function Send_to_Thingsboard(). If there is no serial data coming in then we will just maintain the client connection by using the line client.loop().

Thingsboard server accepts information in form of JSON, so we have to construct our payload from ESP8266 in JSON format. We can do that by simply using string cascading function in Arduino. We also name our payload as "Name". The attributes has to mention the size of the payload that we are sending to Thingsboard. Since the number of characters in each employee name will vary our payload will not have a fixed attribute value so we set it to a maximum of 100. Then to send them over ESP8266 we have to convert the String to char using the toCharArraymethod and finally send it using the client.publish option. Note the v1/devices/me/telemetry should not be changed.

Testing ESP8266 connection with ThingsBoard

Once your IDE and hardware is ready, we can check if the ESP8266 Part is working well. Using the FTDI module upload the code to ESP8266 by keeping it in programming mode. Once the code is uploaded connect it back to AT command mode and press the reset button. Then open serial monitor .

The confirmation message confirms that the ESP8266 module was able to establish communication with our device on Thingsboard server. At this stage the ESP is waiting for input through its Serial monitor, when it receives something it will send that data to the server. So, now it's time to connect the ESP with Arduino to provide the values to ESP.

1.6 Overview and benefits of the project

Remote Control Technology’s line of dependable, durable wireless remote switching systems can and will make money for us and our business. Wireless remote control benefits include:

No legal issues

obtaining access to or traversing properties with hard lines is extremely difficult.

No copper wire to steal

As the price of copper increases, so does the possibility that your wire will be stolen. Using a wireless remote system means no wire for thieves to steal.

Extended range

Unlike much of the equipment on the market, Remote Control Technology’s wireless remote equipment has long-range communication capabilities — up to 5 miles.

Eliminate the need for wire and conduit

Wire and conduit are expensive and high maintenance. Typical wear-and-tear, digging, rodent damage, theft, etc., are all examples of problems that can damage wire. RCT’s wireless remote systems put an end to these drawbacks of wired technology.

Higher profits

Wireless remote switching systems eliminate the costly, labor-intensive process of trenching and laying wire. As a result, the contractor can enjoy an increased profitability of 200 percent or more in this facet of the job.

No FCC licensing required

RCT equipment does not require FCC licensing, whereas much of the other equipment on the market does. This is a significant benefit, as the FCC licensing process alone may take up to 8 weeks.

Less maintenance and servicing

In many states a contractor is obligated by law to maintain pumping systems for up to a year after its installation. RCT switching systems eliminate a majority of these maintenance and servicing issues by automating the job. Fewer service calls mean higher profits.

Reliability and compatibility

All of the components that a contractor puts into a project must interface with one another and have the utmost reliability. RCT wireless remote equipment has proven to be highly compatible with standard equipment used in most industries, as well as offering unparalleled reliability in use with programmable logic controllers (PLCs), various switches and relays, etc.

1.7 Organization of thesis

The thesis is organized into five chapters including the chapter of introduction. Each chapter is different from the other and is described along with the necessary theory required to comprehend it.

Chapter 2 deals with the literature reviews. From this chapter we can see before our project who else works on this topic and how our project is different and advance from those projects.

Chapter 3 deals with the theory required to do the project. The basic of operation of R307/R305 Fingerprint sensor and how to interface with ARDUINO and THINGSBOARD are described there.

Chapter 4 deals with the hardware modeling of the projects. The main features, photographs, step by step operation of the prototype, component listing and the hardware interfacing of the required components are described here.

Chapter 5 describes the operation of the prototype circuit. A flow chart is presented on the actions which describes the principle of R307 Fingerprint sensor detection. Once the temperature is measured by the sensor the controller display it over a 16X2 LCD screen and send it to a remote device through Bluetooth.

Chapter 6 concludes the work performed so far. The possible limitations in proceeding research towards this work are discussed. The future work that can be done in improving the current scenario is mentioned. The future potential along the lines of this work is also discussed.

Chapter 7 References are listed in this chapter

Appendix A, B & C Hardware description, software coding and datasheets are listed here.

CHAPTER 2
(Literature Review)

*M A Muchtar, Seniman, D Arisandi, S Hasanah "***Attendance fingerprint identification system using arduino and single board computer"** *2nd International conference on computing and applied informatics 2017*

2nd International Conference on Computing and Applied Informatics 2017 IOP Publishing
IOP Conf. Series: Journal of Physics: Conf. Series 978 (2018) 012060 doi :10.1088/1742-6596/978/1/012060

2nd International Conference on Computing and Applied Informatics 2017 IOP Publishing
IOP Conf. Series: Journal of Physics: Conf. Series 978 (2018) 012060 doi :10.1088/1742-6596/978/1/012060

Abstract. Fingerprint is one of the most unique parts of the human body that distinguishes one person from others and is easily accessed. This uniqueness is supported by technology that can automatically identify or recognize a person called fingerprint sensor. Yet, the existing Fingerprint Sensor can only do fingerprint identification on one machine. For the mentioned reason, we need a method to be able to recognize each user in a different fingerprint sensor. The purpose of this research is to build fingerprint sensor system for fingerprint data management to be centralized so identification can be done in each Fingerprint Sensor. The result of this research shows that by using Arduino and Raspberry Pi, data processing can be centralized so that fingerprint identification can be done in each fingerprint sensor with 98.5 % success rate of centralized server recording.

*Nur Izzati Zainal, Khairul Azami Sidek, Teddy Surya Gunawan, Hasmah Mansor, and Mira Kartiwi "***Design and Development of portable classroom attendance system based on Arduino and fingerprint biometric**" *Department of Electrical and computer engineering,kulliyah of engineering, department of information systems , kulliyyah of information and communications technology, international Islamic university Malaysia, Kuala Lampur, Malaysia*

In this paper, the design and development of a portable classroom attendance system based on fingerprint biometric is presented. Among the salient aims of implementing a biometric feature into a portable attendance system is security and portability. The circuit of this device is strategically constructed to have an independent source of energy to be operated, as well as its miniature design which made it more efficient in term of its portable capability. Rather than recording the attendance in writing or queuing in front of class equipped with fixed fingerprint or smart card reader. This paper introduces a portable fingerprint based biometric attendance system which addresses the weaknesses of the existing paper based attendance method or long time queuing. In addition, our biometric fingerprint based system is encrypted which preserves data integrity.

*Khin San Myint, Chan Mya Mya Nyein "**Fingerprint Based Attendance System Using Arduino**" Department of Electronic Engineering, Technological University (Sagaing)*
DOI: 10.29322/IJSRP.8.7.2018.p7967 http://dx.doi.org/10.29322/IJSRP.8.7.2018.p7967
Design and Development of Portable Classroom Attendance System
Based on Arduino and Fingerprint Biometric
Design and Development of Portable Classroom Attendance System
Based on Arduino and Fingerprint Biometric
Design and Development of Portable Classroom Attendance System
Based on Arduino and Fingerprint Biometri

Abstract- Attendance system is required in many different places such as offices, companies, schools, organizations and institutions, etc. There are many attendance systems to take attendance. But, every place need to have a good system. This paper describes one of the attendance systems. The main objective of this paper is to study and construct the attendance system using fingerprint module. In this system, Arduino UNO controller and PLX DAQ tool are the main components to display the record on Excel.

*Karthik Vignesh E, Shanmuganathan S , A.Sumithra S.Kishore and P. Karthikeyan "**A Foolproof Biometric Attendance Management System**" Information Technology, Velammal College of Engineering and Technology ,Velammal College of Engineering and Technology, Madurai, Tamil Nadu, India.*

In this paper, we proposed a system which maintains the attendance records of students automatically. Manual entering of attendance in log books becomes a difficult task and it also wastes the time. Reading out the names of each student, each hour destroys the precious time. So we designed an efficient module that comprises of a fingerprint sensor to manage the attendance records of students. Our module enrolls the student's as well as staff's fingerprints. This enrolling is a onetime process and their fingerprints will be stored in the fingerprint sensor. During enrolling of fingerprints alone we require a system since it is a onetime process. You can have your own roll number as your fingerprint id which will be unique for each student and staff. After enrolling process gets completed you can disconnect the module from the system and insert a 9v battery into the module. This will provide power when the module is not connected with the system. Then the module can be taken to the class and the presence of students can be get. The presence of each students will be updated in a database and the data will be passed to the server using Wi-Fi. If a student is absent for a particular class automatically a SMS will be sent to their parents. If a student is absent continuously for more than three days a message intimating the parents to meet the HOD will be sent automatically. So everything here gets automated. Also a unique username and password for staff members are given in a website we create and the website can display the student's details, their attendance percentage which makes the work simple. Also mails and messages can be sent by the staff members using that site to intimate any urgent messages to the parents.

<u>Lia Kamelia</u>, <u>Eki Ahmad Dzaki Hamidi</u>, <u>Wahvudin Darmalaksana</u>, <u>Afit Nugraha</u>, "_Real-Time Online Attendance System Based on Fingerprint and GPS in the Smartphone"_ ***UIN Sunan Gunung Djati, Department of Electrical Engineering, Bandung, Indonesia***

Real-time online attendance method is helpful for workers who do a lot of activities outside the office or workers with multi-schedule. The attendance system using online biometric fingerprint system will reduce the problems caused by manual system usage such as lags in data management. The purpose of the research is to constructs an online presence system that combines fingerprint modules and GPS. The ZFM-20 fingerprint module is used as the system's main input as well as a security tool as an entrance to get access to the entire system. GPS module is applied to determine the user's location and sends it to the smartphone. Arduino module in the system will send a text message to the parties concerned about the user's location data automatically. Each module works well and testing the entire system showed the system work reliable according to the initial scenario. The User can access the report using SMS, website, and application on the Android smartphone. The fingerprint sensor can determine the fingerprint stored in the database with an average response time of 1.39 seconds, and GPS can determine latitude and longitude with an average error of 0.007352% and 0.0003% respectively.

A. Jain, L. Hong , S. Pankanti, and R. Bolle , "An Identity Authentication System Using Fingerprints*", Proceedings of the* ***IEEE, Vol. 85****, Issue 9, 1997, pp. 1365-1388.*

Abstract- Proper attendance recording and management has become important in today's world as attendance and achievement go hand in hand. Attendance is one of the work ethics valued by employers. Most of the educational institutions and government organizations in developing countries still use paper based attendance method for maintaining the attendance records. There is a need to replace these traditional methods of attendance recording with biometric attendance system. The unique nature of fingerprint makes it ideal for use in attendance management systems. Besides being secure, Fingerprint based attendance system will also be environment friendly. Fingerprint matching is widely used in forensics for a long time. It can also be used in applications such as identity management and access control. This review incorporates the problems of attendance systems presently in use, working of a typical fingerprint based attendance system, study of different systems, their advantages, disadvantages and comparison based upon important parameters.

Piyush Devikar, Ajit Krishnamoorthy, Aditya Bhanage, Mohit Singh Chauhan *Department of Electronics and Telecommunication Engineering, Vivekanand Education Society's Institute of Technology, Mumbai University, Mumbai, India.*

Abstract: Biometric student attendance system increases the efficiency of the process of taking student attendance. This paper presents a simple and portable approach to student attendance in the form of an Internet of Things (IOT) based system that records the attendance using fingerprint based biometric scanner and stores them securely over cloud. This system aims to automate the cumbersome process of manually taking and storing student attendance records. It will also prevent proxy attendance, thus increasing the reliability of attendance records. The records are securely stored and can be reliably retrieved whenever required by the teacher.

Dipak Gadekar(1), Sanyukta Ghorpade(2), Vishakha Shelar(3), Ajay Paithane*(4)."(1,2,3) Students, (4) Professor, Department of Electronics And Telecommunication Engineering, JSPMs Rajarshi Shahu College of Engineering, Tathwade, Pune, Maharashtra, India.*

Abstract - Authentication is one of the vital concern in this era of information system. . Inclusive of the other techniques, Human Face Recognition (HFR)is one ofthe techniques which is used for user authentication . HFR has been extensively used in many appliances asin , video conferencing,military services and attendance systems . Maintaining attendance is difficult process if it is done manually. The automated attendance system for administrating the attendance can be put into effect using the several ways of biometrics. Usage of this system can resolve the issue of fake attendance and proxies. Instead of recording the attendance in writing, taking attendance through fingerprint and face recognition will make it a hassle free process.

Niharika Yadu(1) , K Uma(2) *Research Scholar, Dept. of Electronics Engineering, BIT, Durg, C.G., India(1) Associate Professor, Dept. of Electronics Engineering, BIT, Durg, C.G., India(2)* "**A Review on Real Time IOT Based advanced E-attendance System.**"

Abstract: If we talk about the current scenario of our education system then we found that we have a lot of technologies to use but still we are following the traditional system. If we talk about the attendance system in universities and schools, lecturers did that work manually. Lecturers take the attendance and update it manually in the database. If we combine the fingerprint sensor and RFID sensor with IOT (Internet of Things) than we can do it automatically and there is no need to do it by lecturers. We can use IOT and finger print sensor for better performance. IOT data is directly store on server in real time so we can access it from anywhere and anytime which will provide us with better proficiency and flexibility.

"Biometric Student Attendance System using IoT", *Sameer Kanse , Monish Shaikh, Siddesh Gadhari , Pravin Labde , Prof. Anuprita Gawande. Department of Electronics and Telecommunication Engineering, Shivajirao S. Jondhle College of Engineering & Technology, Asangaon, Mumbai University, India.*

ABSTRACT.In the World of Technology, Biometrics plays an effective role in identifying Human beings. Through this paper, we will develop a unique system that can identify students for attendance purpose using their fingerprints. We will need an Arduino Uno board for interfacing microcontroller with the Finger Print Scanner R305. So, with the help of Finger Print Scanner 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 IP address, to access it from anywhere in the world.

CHAPTER 3
(THEORY)

3.1 R307 Fingerprint Sensor

Fingerprints are one of the many unique biometric signatures which we can use to identify people very accurately. But just by holding someone's hand and staring at their fingers can't be practical [grins]; we're not good at it. But computers are good at recognizing and matching patterns very fast and accurately. Before we can process a fingerprint pattern with a computer, we.must."capture".it.

There exists many methods to digitize fingerprints; from forensic methods to ultrasound scanning. In this tutorial, we will learn how an **Optical Fingerprint Scanner** works and how we can interface the R307 fingerprint scanner module to Arduino. R307 is an optical fingerprint scanner module from R30X series produced by a Chinese vendor called Hangzhou Grow Technology Company Limited. Other sensors in the series are R300, R301T, R302, R303, R303T, R305, R306, R308, and R311, some of which are capacitive sensors. Despite having different sensing techniques and form-factors, they all share the same interface and command set. Therefore it is easy to adapt the library that you find here for other models as well.

R307 Fingerprint Module consists of optical fingerprint sensor, high-speed DSP processor, high-performance fingerprint alignment algorithm, high-capacity FLASH chips and other hardware and software composition, stable performance, simple structure, with fingerprint entry, image processing, fingerprint matching, search and template storage and other functions.

3.2 Operation Principle

Fingerprint processing includes two parts: fingerprint enrollment and fingerprint matching (the matching can be 1:1 or 1:N).
When enrolling, user needs to enter the finger two times. The system will process the two time finger images, generate a template of the finger based on processing results and store the template. When matching, user enters the finger through optical sensor and system will generate a template of the finger and compare it with templates of the finger library. For 1:1 matching, system will compare the live finger with specific template designated in the Module; for 1:N

matching, or searching, system will search the whole finger library for the matching finger. In both circumstances, system will return the matching result, success or failure.

3.3 Technical Parameters

- Supply voltage: DC 4.2 ~ 6.0V
- Supply current: Working current: 50mA (typical) Peak current: 80mA
- Interface: UART and USB
- Fingerprint image input time: <0.3 seconds
- Window area: 14x18 mm
- Matching method: Comparison method (1: 1)
- Search method (1: N)
- Characteristic file: 256 bytes
- Template file: 512 bytes
- Storage capacity: 1000 pieces
- Security Level: Five (from low to high: 1,2,3,4,5)
- Fake rate (FAR): <0.001%
- Refusal rate (FRR): <1.0%
- Search time: <1.0 seconds (1: 1000 hours, mean value)
- Host interface: UART \ USB1.1
- Communication baud rate (UART): (9600xN) bps Where N = 1 ~ 12 (default N = 6, i.e., 57600bps)
- Working environment: Temperature: -20 ℃ - +40 ℃ Relative humidity: 40% RH-85% RH (no condensation)
- Storage environment: Temperature: -40 ℃ - +85 ℃ Relative humidity: <85% H (no condensation)
- Outline Dimensions: Split Type, 44.1*20*23.5 mm

Suitable for fingerprint lock, fingerprint safes and other purposes.

3.4 Features

- Perfect function: independent fingerprint collection, fingerprint registration, fingerprint comparison (1: 1) and fingerprint search (1: N) function.
- Small size: small size, no external DSP chip algorithm, has been integrated, easy to install, less fault.
- Ultra-low power consumption: low power consumption of the product as a whole, suitable for low-power requirements of the occasion.
- Anti-static ability: a strong anti-static ability, anti-static index reached 15KV above.
- Application development is simple: developers can provide control instructions, self-fingerprint application product development, without the need for professional knowledge of fingerprinting.
- Adjustable security level: suitable for different applications, security levels can be set by the user to adjust.
- Finger touch sensing signal output, low effective, sensing circuit standby current is very low, less than 5A.

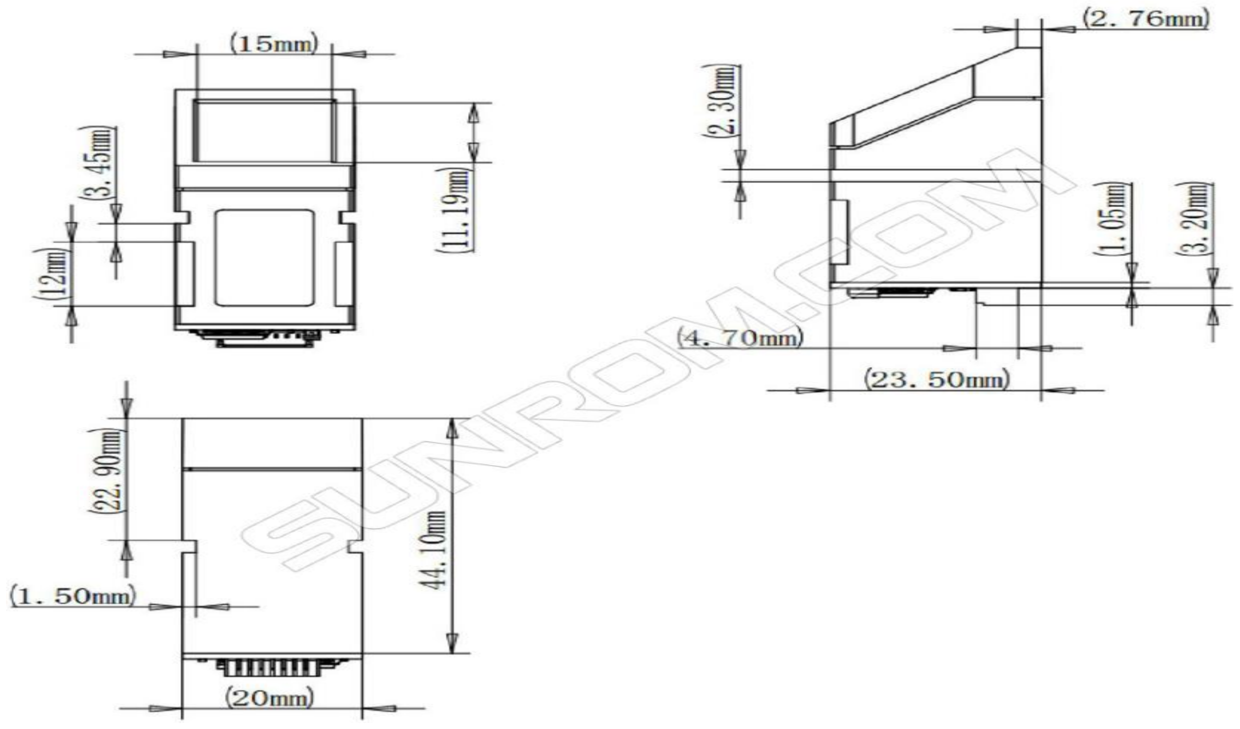


Figure 3: Fingerprint Sensor 2D model.

The skin on the palms of our hands have a special pattern called *friction ridges* that help us grab things effectively without slipping. These patterns consist of *ridges* and *valleys* arranged in certain configurations and is unique for each individual. Our finger tips also have them as you can see from the above image. When a finger comes in contact with a surface, the ridges make strong contact with the surface. When we strongly grab something, the moisture, oil, dirt and dead skin cells on our finger can attach to the surface of the material, leaving an *impression* we call a fingerprint. Various forensic methods involving the use chemicals are used to extract such fingerprints from crime scenes and are called *latent fingerprints*. But an optical fingerprint scanner works a bit differently.

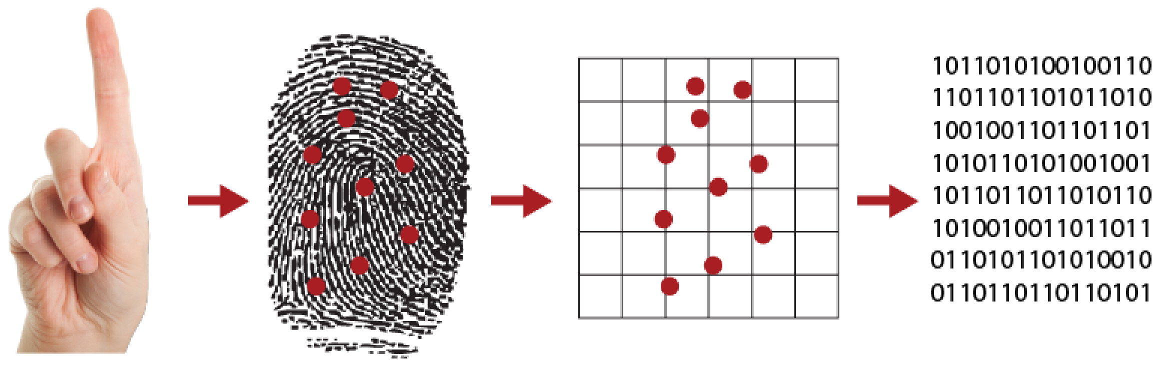


Figure 4: Biometric Acceptance by Unit.

An optical fingerprint scanner works based on the principle of Total Internal Reflection (TIR). In an optical fingerprint scanner, a glass prism is used to facilitate TIR. Light from an LED (usually blue color) is allowed to enter through one face of the prism at a certain angle for the TIR to occur. The reflected light exits the prism through the other face where a lens and an image sensor (essentially.camera).are.placed.

When there's no finger on the prism, the light will be completely reflected off from the surface, producing a plain image in the image sensor. When TIR occurs, a small amount of light leaked to the external medium and it is called the Evanescent Wave. Materials with different *refractive indexes* (RI) interact with the evanescent wave differently. When we touch a glass surface, only the ridges make good contact with it. The valleys remain separated from the surface by air packets. Our skin and air have different RIs and thus affect the evanescent field differently. This effect is called Frustrated Total Internal Reflection (FTIR). This effect alters the intensities of the internally reflected light and is detected by the image sensor. The image sensor data is processed to produce a high contrast image which will be the digital version of the fingerprint.

In capacitive sensors, which are more accurate and less bulky, there's no light involved. Instead, an array of capacitive sensors are arranged on the surface of the sensor and allowed to come in contact with the finger. The ridges and air packets affect the capacitive sensors differently. The data from the sensor array can be used to generate a digital image of the fingerprint.

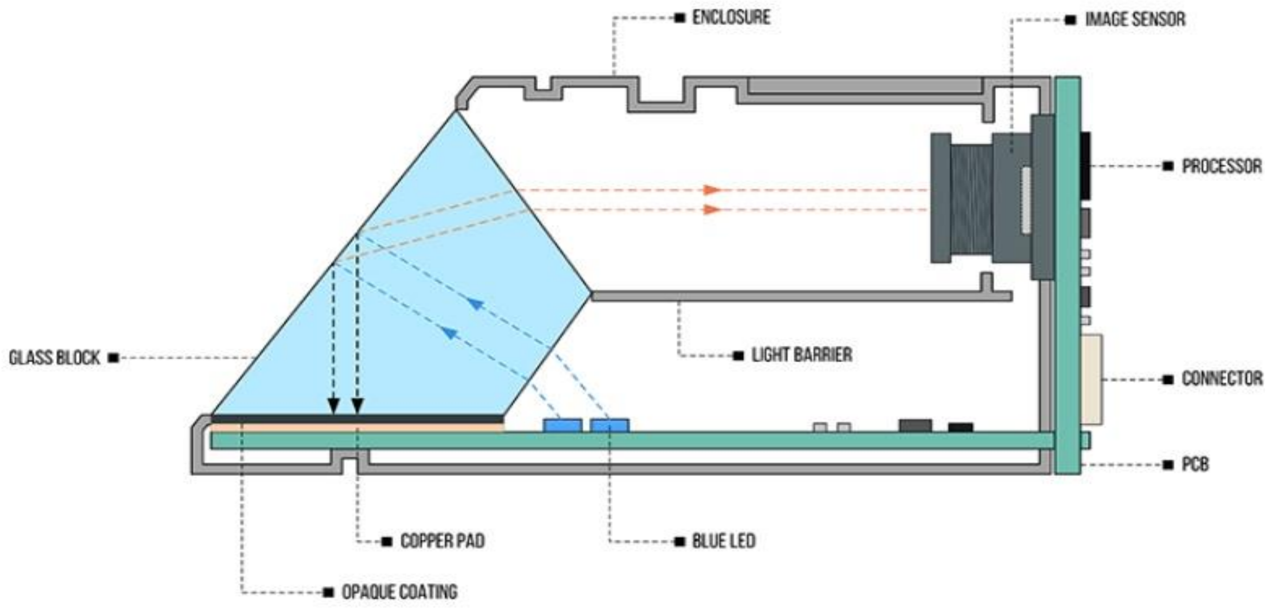


FIG. R307 FINGERPRINT SCANNER CROSS-SECTION

Figure 5: Cross Section of Fingerprint Scanner.

Above is a cross-sectional diagram that I made to better understand the construction (illustrative only, not a physically exact one). Opening the module was easy; there are four Philips screws on the back. Unscrew them and you can remove the PCB. There are two PCBs; one arranged horizontally and one vertically (shown in washed green). These PCBs are connected by solder. The four blue LEDs and the touch sense pad are on the horizontal PCB. The vertical PCB has the image sensor, the processor and connector. When inserted, the touch sense pad comes in contact with the glass block above. The image sensor is soldered and glued. Strangely, I couldn't find any lens on it. May be it doesn't need one. The enclosure has an internal barrier to separate the light from the LEDs and the light coming out of the prism. On the bottom side of the prism a black epoxy is coated which gives a high-contrast background for the fingerprint image. To access the prism, just remove the cap on the front.

3.5 Interface Description

The R307 fingerprint module has two interface TTL UART and USB2.0, USB2.0 interface can be connected to the computer; RS232 interface is a TTL level, the default baud rate is 57600 , can be changed, refer to a communication protocol ; can And microcontroller, such as ARM, DSP and other serial devices with a connection, 3.3V 5V microcontroller can be connected directly. Needs to connect the computer level conversion, level conversion note , embodiments such as a MAX232circuit.

3.6 Working of fingerprint sensor and push buttons

We use **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 enroll, Delete, UP/Down. ENROLL and DEL key has triple features. ENROLL key is used for enrollment of a new person into the system. So when the user wants to enroll new finger then he/she need to press ENROLL key then LCD asks for the ID, where user want to be store the finger print image. Now if at this time user does not want to proceed further then he/she can press ENROLL key again to go back. This time ENROLL key behave as Back key, i.e. ENROLL key has both enrollment and back function. Besides enroll key is also used to download attendance data over serial monitor. Similarly, DEL/OK key also has the same double function like when user enrolls new finger, then he/she need to select finger ID by using another two key namely UP and DOWN. Now user need to press DEL/OK key (this time this key behave like OK) to proceed with selected ID. And Del key is used for reset or delete data from EEPROM of Arduino.

Fingerprint sensor module captures finger’s print image and then converts it into the equivalent template and saves them into its memory as per selected ID by Arduino. All the process is commanded by Arduino like taking an image of finger’s print, convert it into templates and storing as ID etc. You can check some more projects using fingerprint module. Here we have added a LED which indicates that fingerprint module is ready to take an image of the finger. A buzzer is also used for various indications. Arduino is the main component of this system it is responsible for control of the whole system.

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.

Whenever user place his finger over fingerprint module then fingerprint module captures finger image, and search if any ID is associated with this fingerprint in the system. If fingerprint ID is detected then LCD will show Attendance registered and in the same time buzzer will beep once and LED will turn off until the system is ready to take input again.

Along with the fingerprint module, we have also used an **RTC module for Time and date**. Time and date are running continuously in the system. So Arduino take time and date whenever a true user places his finger over fingerprint and save them in the EEPROM at the allotted slot of memory.

Here we have created 5 user space in this system for 30 days. By pressing the RESET button in Arduino and then immediately enroll key will be responsible for downloading attendance data over serial monitor from the Arduino EEPROM Memory.

3.7 Implementation:

1. The Finger Print Sensor is interfaced with the Arduino board.
2. At the beginning, your finger will be scanned by placing your finger on the scanner

3. Once your finger is scanned, the scanner will generate template by Image Processing method which will be stored for comparing
4. Like this we will store all the templates of different people
5. So when we place our finger, the scanner will scan the finger and it will generate template and this template will be compared with previously stored templates
6. If both templates are matched, then certain people data stored will be shown on the LCD

The data regarding how many students were present on day to day basis can be updated in remote cloud rather maintain ledgers and record books and can be retrieved whenever we want.

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.

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.

3.8 Pin configuration:

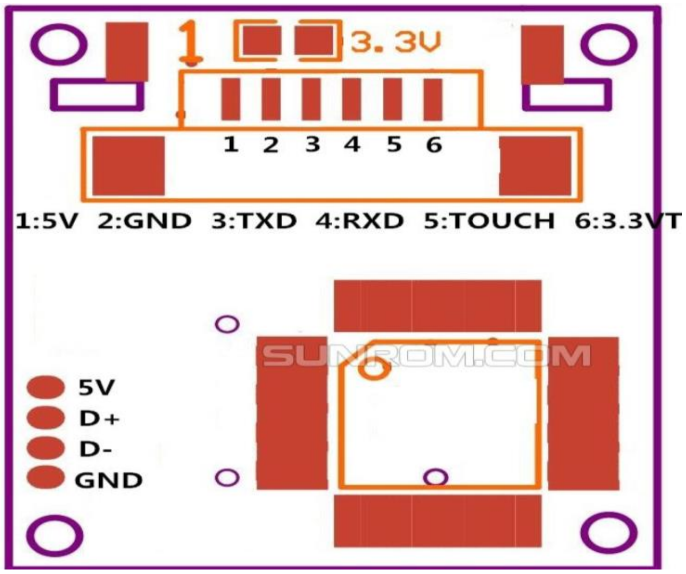


Figure 6: Pin Configuration

Table 02: PINOUTS:

PIN#	PIN NAME	DETAILS
1	5V	Regulated 5v DC
2	GND	Common Ground
3	TXD	Data Output – Connect to MCU RX
4	RXD	Data Input – Connect to MCU TX
5	TOUCH	Active Low output when there is touch on sensor by finger

6	3.3V	Use this wire to give 3.3V to sensor instead of 5V

USB Cable Connections are 5V/D+/D-/GND (Optional)

The scanner can be interfaced and powered from both 3.3V and 5V supplies. The working voltage of the scanner controller is always 3.3V. There's a 3.3V regulator on the PCB. The 5V supply you provide goes to the input of that regulator, and the 3.3V you supply bypasses the regulator and goes directly to the fingerprint scanner controller.

When you want to power the scanner from 5V and interface with a 5V microcontroller, supply the power to the pins 1 and 6, and disconnect the 3.3V jumper shown in the picture. If you want to supply 3.3V and interface the scanner with a 3.3V microcontroller such as Arduino Due, supply the power to pins 1 and 6 and short the 3.3V jumper. Improper voltages and configurations might damage the controller. So be careful with it.

The pin 6 (Touch Sense Power) is the supply voltage for the finger detection circuit. When a finger is present on the scanner, the output of pin 5 (Touch Sense) will be high. This signal can be used to initiate the scanning of the finger manually. Otherwise the scanner will wait for some time.to.detect.the.finger.

The R307 has both USB and UART interfaces. With the USB, you can directly connect the scanner to a computer and communicate with it. A virtual COM port will be created when you connect the scanner to a Windows PC. If you want to interface the scanner with a microcontroller, you can use the UART interface which supports baud rates up to 115200 bps.

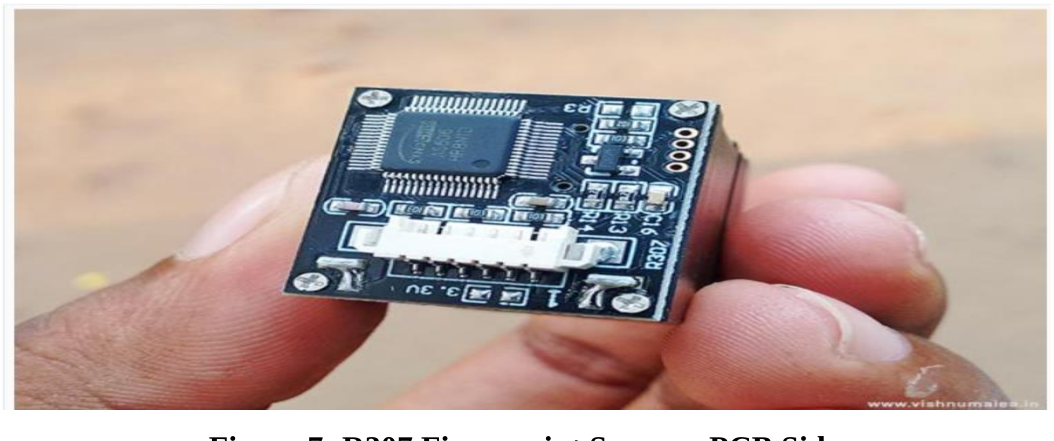


Figure 7: R307 Fingerprint Scanner PCB Side

The main controller on the PCB is AS606 from a company called Synochip. I don't know how Synochip is related to the company Hangzhou Grow. Whatever that is, the AS606 is a microcontroller with *Cordis 5+ RISC* cores and has everything needed for a performance controller.including.a DSP.

The R307 Manual is the only document to our rescue, and it is ambiguous at many places. The R30X Series Manual has some more information. The below schematic is included in the manual.

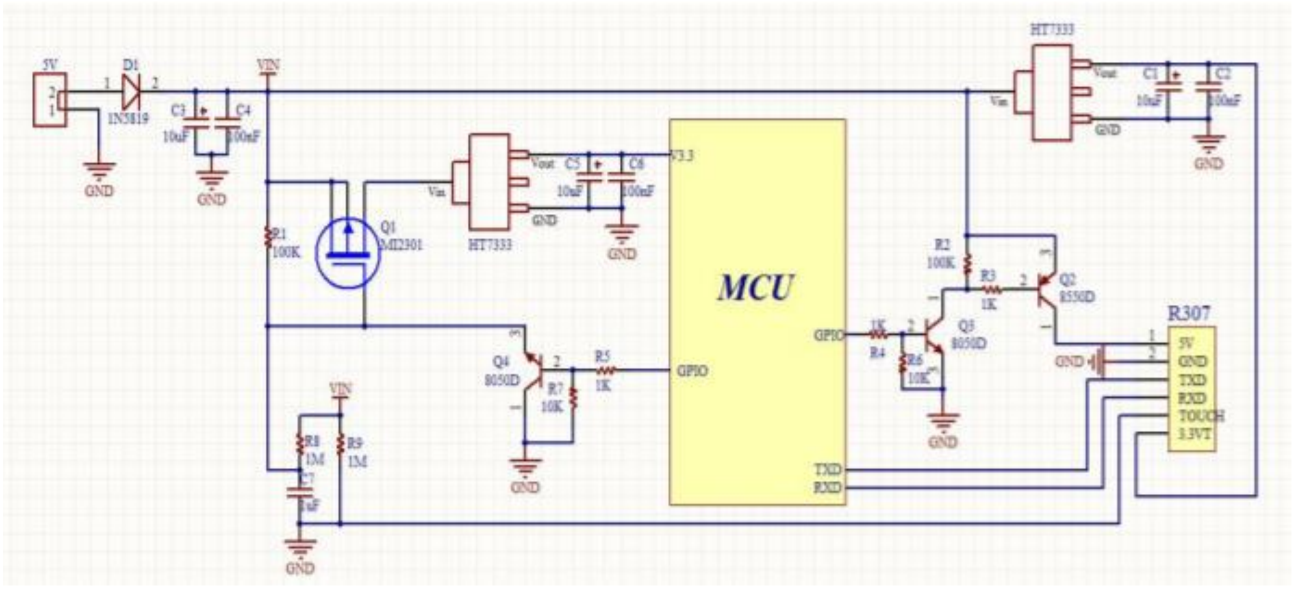


Figure 8:R307 Schematic Diagram.

3.9 R307 Memory and Registers

1. **Notepad** - This is a 512 bytes of the non-volatile flash memory. It is logically divided into 16 pages with 32 bytes each. Instructions GR_WriteNotepad and GR_ReadNotepad can be used to access this memory. When writing a page, it is taken as a whole and the contents are replaced.

2. **Image Buffer** - Image buffer is used to store a BMP image of size **256 x 288**, each pixel occupying a byte. This buffer is part of the RAM and the contents are lost when power is lost.

3. **Character File Buffer** - A *character file* is a processed high contrast image of a fingerprint. Two character files from two consecutive scans are combined to form a *template file* which is the

final version of the fingerprint that is stored in the *fingerprint library* (not to be confused with the Arduino library. Fingerprint library is the memory used to store up to 1000 fingerprints). The two character file buffers are CharBuffer1 and CharBuffer2 each with size of 512 bytes.

4. **Fingerprint Library** - This is a section of the flash memory where 1000 fingerprint templates can be stored. Templates are arranged sequentially with numbering from 0 to N-1 (The manual says 0-N) where N is the capacity of the library determined by the size of the flash memory. There are instructions to store, process and delete templates from this memory. They will be explained later.

5. **System Configuration Register -** This is a 16-bytes long register bank containing operating parameters and status. Except the device address which takes up 4 bytes, rest of the parameters are 2 bytes (a word) in length. The command ReadSysPara can be used to read, and command SetSysPara can be used to write this register bank. **(TABLE: 03)**

NAME	**DESCRIPTION**	**OFFSET(Word)**	**SIZE(Word)**
Status Register	Contents of system status register	0	1
SystemIdentifier Code	Fixed Register: 0*0009	1	1
Library Size	Fingerprint library size	2	1
Security Level	Security Level (1,2,3,4,5)	3	1
Device Address	32-bit device address	4	2
Data Packet Size	Size code (0,1,2,3)	6	1
Baud Multiplier	N (baud=9600*N bps)	7	1

The *Status Register* indicates the current operation status of the module, and comprises of the following :

Bit Number	**15-4**	**3**	**2**	**1**	**0**
Description	*Reserved*	*imgBufStat*	*PWD*	*Pass*	*Busy*

where *Busy* = 1: system is executing commands; 0: system is free, *Pass* = 1: found a matching fingerprint; 0: fingerprint not found, *PWD* = 1: handshaking password verified; 0: password not verified, *ImgBufStatus* = 1: image buffer contains valid image; 0: image buffer does not have a valid image.

System Identifier Code is a fixed value that determines the type of module. Its value is 0x0009 for.R307.

Library Size is the number of templates that can be stored in the module. The maximum value for this.parameter.is.1000.for.R307.

Security Value determines the threshold for fingerprint searching and matching. Its value can be from 1-5. When it is 1, the **FAR** (False Acceptance Rate) is the highest and **FRR** (False Recognition Rate) is the lowest. FAR is simply the number that determines how likely the module will identify a weakly matched fingerprint as positive. FRR is how likely the module will identify a wrong fingerprint as negative. At level 5, the FAR is the lowest and FFR is the highest. In this setting, it can be difficult to match your fingerprint.

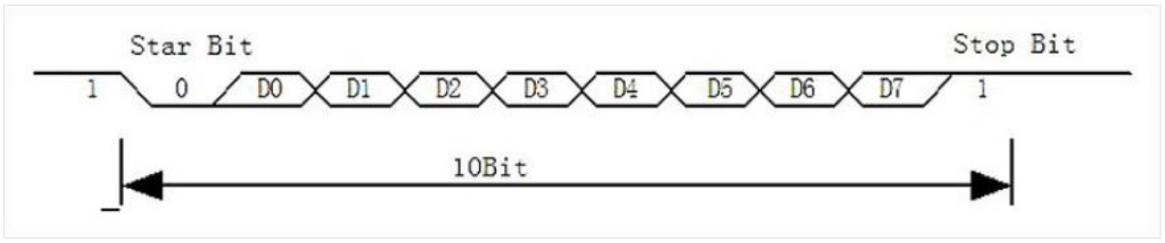
Device Address is a 32-bit value that holds the address of the module. The correct address is needed to communicate with the module. If you don't send the correct address, the module won't execute any commands. Device address can be modified with the command SetAddr. The factory programmed address is 0xFFFFFFFF. There's no methods specified in the manual to reset the address to default, so keep the address safe if you ever change it.

Data Packet Size determines the maximum length of data content in a single packet. Its value can be 0-3 where 0 = 32 bytes, 1 = 64 bytes, 2 = 128 bytes and 3 = 256 bytes.

Baud Multiplier sets the UART communication speed of the module. The minimum speed is 9600bps and can be set to up to 12 times of that which is 115200bps. The multiplier value N can be from 1-12 and the effective speed will be (9600*N) bps. The default baudrate is 57600bps.

3.10 R307 Communication Protocol

Both UART and USB interfaces use a common serial communication protocol based on a packet format (the manual refers packets as "packages"). All data and commands are to be sent as data packets and all responses from the module will also be packets. So we need to frame data and commands as packets before sending out, and must extract data from response packets. The UART frame format is 10 bit with 1 start bit, 1 stop bit and 8 data bits.



R307 UART frame format

Figure 9: R307 UART Frame Format

The Packet format is as follows(length in bytes is shown in brackets)

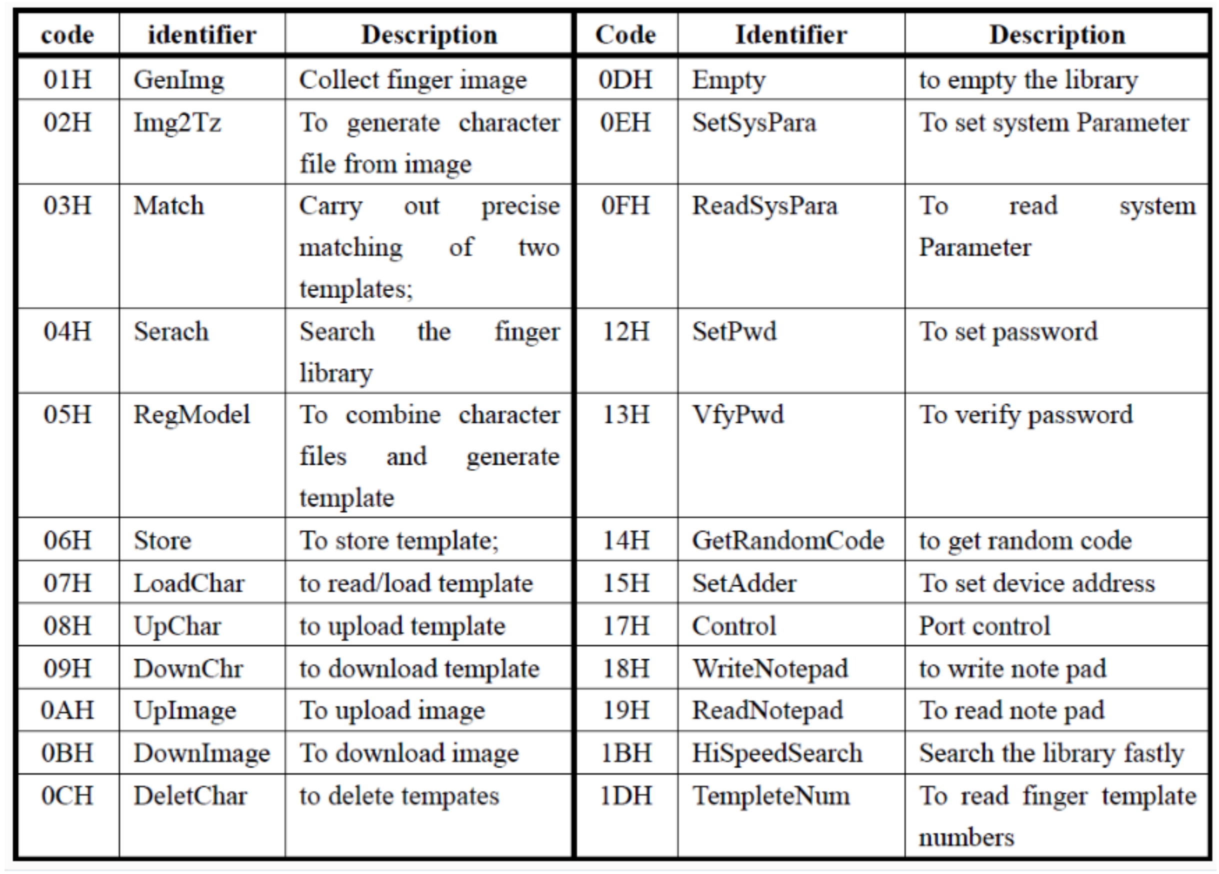
Header (2)	Address (4)	Packet Identifier(1)	Packet Length(2)	Packet Content (Instruction/Data/Parameter)	Check sum (2)

If you feel confused, the UART frame is how a byte of data is transferred via the UART interface. A packet is a group of many such bytes (or frames). Let's see the definitions of each part.

1. Header : This indicates the start of a packet. It has to be the fixed value 0xEF01. It is 2 bytes long and the high byte is always transferred first.
2. Address : This is the 32-bit address of the scanner module. The module will accept instructions only if the address we are sending matches the address stored in the module. The default address is 0xFFFFFFFF and can be modified with SetAddr instruction.
3. Packet Identifier : This determines what type of packet we're sending or receiving. It is 1 byte long and depending on the value the packet types can be,
    - 0x01 : The packet contains a command.
    - 0x02 : Data packet. A data packet must be followed by a command packet or acknowledge packet.
    - 0x07 : Acknowledge packet. It is sent by the module in response to a command.
    - 0x08 : End of data transfer packet. When we send large volume data such as an image, the data transfer will be terminated by this packet.

4. Packet Length : This is the total length of *Packet Content* and *Checksum* in bytes. Maximum length is 256 bytes and high byte is transferred first.
5. Packet Content : This can be data/command/parameters etc. of varying length. The Packet Length is the value that specifies the length of the data here in bytes.
6. Checksum : This is the arithmetic sum of all bytes in *Packet Identifier*, *Packet Length* and *Packet Content*. Overflowing bits are ignored. High byte is always transferred first.

In order make the fingerprint scanner work, we must send *instructions* or *commands* in the form of packets. Each instruction is simply a 1-byte code that we must include in the packet. The module responds to each instruction with an *acknowledgment packet* that describes the result and status of command execution. Each instruction has a set of expected response codes found in the ACK packet that are called confirmation codes. Instructions and their byte codes are grouped according to their functions as shown below,



code	identifier	Description	Code	Identifier	Description
01H	GenImg	Collect finger image	0DH	Empty	to empty the library
02H	Img2Tz	To generate character file from image	0EH	SetSysPara	To set system Parameter
03H	Match	Carry out precise matching of two templates;	0FH	ReadSysPara	To read system Parameter
04H	Serach	Search the finger library	12H	SetPwd	To set password
05H	RegModel	To combine character files and generate template	13H	VfyPwd	To verify password
06H	Store	To store template;	14H	GetRandomCode	to get random code
07H	LoadChar	to read/load template	15H	SetAdder	To set device address
08H	UpChar	to upload template	17H	Control	Port control
09H	DownChr	to download template	18H	WriteNotepad	to write note pad
0AH	UpImage	To upload image	19H	ReadNotepad	To read note pad
0BH	DownImage	To download image	1BH	HiSpeedSearch	Search the library fastly
0CH	DeletChar	to delete tempates	1DH	TempleteNum	To read finger template numbers

R307 commands and their byte codes

Figure 10: R307 commands and byte codes

Following is the list of confirmation codes.

- **0x00** - Command execution complete
- **0x01** - Error when receiving data package
- **0x02** - No finger on the sensor
- **0x03** - Failed to enroll the finger
- **0x04** - Failed to generate character file due to the over-disorderly fingerprint image
- **0x05** - Failed to generate character file due to the over-wet fingerprint image
- **0x06** - Failed to generate character file due to the over-disorderly fingerprint image
- **0x07** - Failed to generate character file due to lack of character point or over-smallness of fingerprint image
- **0x08** - Finger doesn't match
- **0x09** - Failed to find a matching finger
- **0x0A** - Failed to combine the character files
- **0x0B** - Addressing PageID is beyond the finger library
- **0x0C** - Error when reading template from library or the template is invalid
- **0x0D** - Error when uploading template
- **0x0E** - Module can't receive the following data packages
- **0x0F** - Error when uploading image
- **0x10** - Failed to delete the template
- **0x11** - Failed to clear finger library
- **0x13** - Wrong password
- **0x15** - Failed to generate the image
- **0x18** - Error when writing flash
- **0x19** - No definition error
- **0x21** - Password not verified
- **0x1A** - Invalid register number
- **0x1B** - Incorrect configuration of register
- **0x1C** - Wrong notepad page number
- **0x1D** - Failed to operate the communication port

- **0x41** - No finger on sensor when add fingerprint on second time
- **0x42** - Failed to enroll the finger for second fingerprint scan
- **0x43** - Failed to generate character file due to lack of character point or over-smallness of fingerprint image for second fingerprint scan
- **0x44** - Failed to generate character file due to the over-disorderly fingerprint image for second fingerprint scan
- **0x45** - Duplicate fingerprint
- Others - System reserved

The confirmation code 0x21 is not listed in the manual but I found when working with the module. It will be sent when we try to execute commands without verifying the password first. All the commands and their response codes are explained in detail in the manual. So it would be redundant to do it here. But let's look at the VfyPwd instruction and its corresponding packet for an example.

2 bytes	**4 bytes**	**1 byte**	**2 bytes**	**1 byte**	**4 bytes**	**2 bytes**
Header	**Address**	**Packet Identifier**	**Packet Length**	**Instruction Code**	**Data (Password)**	**Checksum**
0*EF01	0*FFFFFFFF	0*01	0*07	0*13	0*6F6F6F6F	0*01D7

Above is the packet format for verifying password. *Address* is assumed to be default. We're sending out a command, and so the *Packet Identifier* has to be 0x07. The *Packet Length* will be always 7 for this instruction. The *Packet Content* is split into Instruction Code and Password. Password is 0x6F6F6F6F for our example. If we calculate the *Checksum* of the bytes, it will be 0x01D7 which is two bytes long of overall byte i.e., 2bytes long in size.

If the address and password are correct and the packet is correctly formatted, the module will respond with the following acknowledgment packet

2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
Header	Address	Packet Identifier	Packet Length	Confirmation Code	Checksum
0*EF01	0*FFFFFFFF	0*01	0*03	0*00	0*04

The confirmation code will be 0x00 if our password was correct and 0x13 if it was not.

CHAPTER 4

(Hardware Modeling)

4.0 Prototype Modelling

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.

4.1 Main features of the prototype

The features of the developed prototype are:

- LCD display(showing the username and time & date)
- Upto 256 fingerprints can be stored and checked when needed
- Fingerprint is stored in cloud digitally
- After fingerprint checking data is displayed in a serial monitor from Thingsboard account
- Fingerprint data can be stored and deleted as many times as one wants
- Buzzer and LCD indicates fingerprint is stored and checked
- The date and time is shown when fingerprint is stored and checked along with username
- Cost Effective(Rs 2000/- approx)

4.2 Overview of the Project

Working of this **fingerprint attendance system project** is fairly simple. First of all, the user needs to enrol 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.

4.3 Photographs of the prototype:

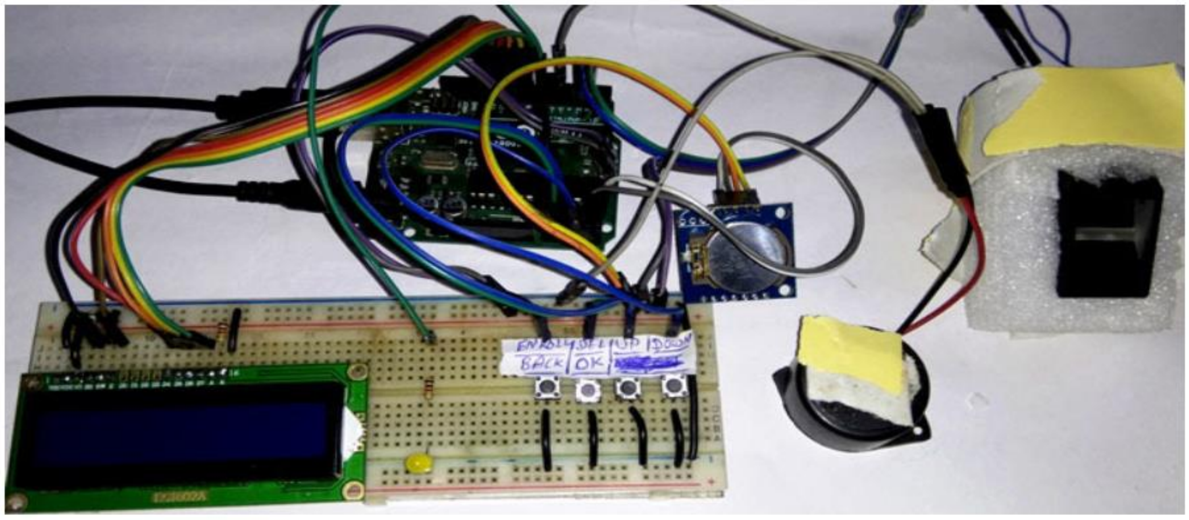


Figure 11: Overall Setup of prototype

4.4 Step by step operation of the prototype

1. Connect the dc power jack to the prototype board(5v dc)

2. In this **fingerprint attendance system circuit**, we used **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 enroll, Delete, UP/Down. ENROLL and DEL key has triple features.

3. ENROLL key is used for enrollment of a new person into the system. So when the user wants to enroll new finger then he/she need to press ENROLL key then LCD asks for the ID, where user want to be store the finger print image.

4. Now if at this time user does not want to proceed further then he/she can press ENROLL key again to go back. This time ENROLL key behave as Back key, i.e. ENROLL key has both enrollment and back function. Besides enroll key is also used to download attendance data over serial monitor.

5. Similarly, DEL/OK key also has the same double function like when user enrolls new finger, then he/she need to select finger ID by using another two key namely UP and DOWN. Now user need to press DEL/OK key (this time this key behave like OK) to proceed with selected ID. And Del key is used for reset or delete data from EEPROM of Arduino.

6. So next time when someone checks his fingerprint on the fingerprint sensor if any stored fingerprint matches with any of the stored fingerprints then the LCD display shows the username and time & date of the checking and register the data. Thus attendance is stored.

4.5 Components Required : Table 04:

List of Components:

Sl No.	**Component**	**Quantity**
1	Arduino	1
2	Fingerprint Module – R307	1
3	Push Button	4
4	LEDs	1
5	1k Resistor	2
6	2.2k Resistor	1
7	Power Supply	

8	Connecting Wires	35
9	Box	1
10	Buzzer	1
11	16*2 LCD	1
12	Bread Board	1
13	RTC Module(DS3231)	1

4.6 Cost estimation of the prototype: Table 05:

Cost Estimation table:

List of Components

Sl No.	**Component**	**Quantity**	**Price(in rupees)**
1	Arduino	1	380
2	Fingerprint Module – R307	1	1600
3	Push Button	4	20
4	LEDs	1	40 per set
5	1k Resistor	2	20 per box
6	2.2k Resistor	1	20 per box
7	Power Supply		250 for 3 piece
8	Connecting Wires	35	225 for 3 pair
9	Box	1	------------------
10	Buzzer	1	30 for 2 pc
11	16*2 LCD	1	250
12	Bread Board	1	100
13	RTC Module(DS3231)	1	160
			Total- ***3095/-(approx)***

4.7 Hardware Connection

4.7.1 Prototype hardware connection:

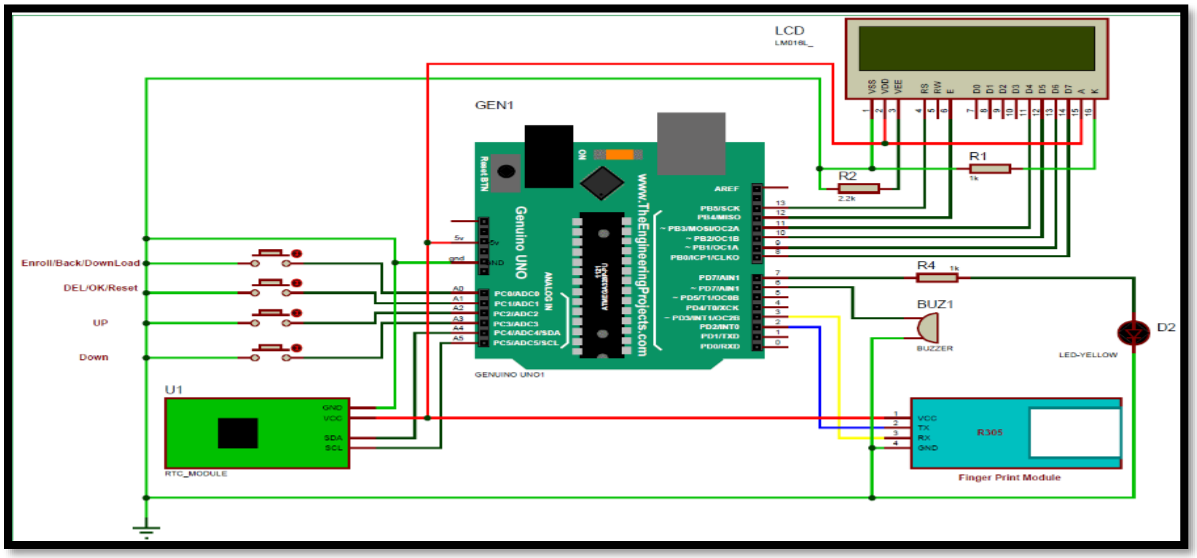


Figure 12: Overall Circuit Diagram for Setup

HARDWARE IMPLEMENTATION:

- **ARDUINO UNO:** Arduino UNO is used here to control the operations involved in taking attendance. The four operations that are to be performed are to enroll, verify, delete and reset. Arduino is chosen here as it is easy to use, code, handle and has many modules which add on features to Arduino board.

- **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.

- **16X2 LCD DISPLAY:** LCD Display is used here to provide messages to the user to have a better interaction with the device. LCD Display has greenlight in background with characters displayed on them in black. Characters are displayed in 7X5 matrix.

- **Wi-Fi Module:** ESP8266 Wi-Fi module is generally used to establish the wireless communication between the devices. But this module is not capable of 5-3V logic shifting and will require an external logic level converter.

- **Push Button.**
- **LED Lights.**
- **1K Resistor.**
- **2.2K resistor.**
- **Power.**
- **Connecting wires.**
- **Box.**
- **Buzzer.**
- **Bread Board.**
- **RTC Module. (DS3231)**

7.4.2 Detailed Hardware Description

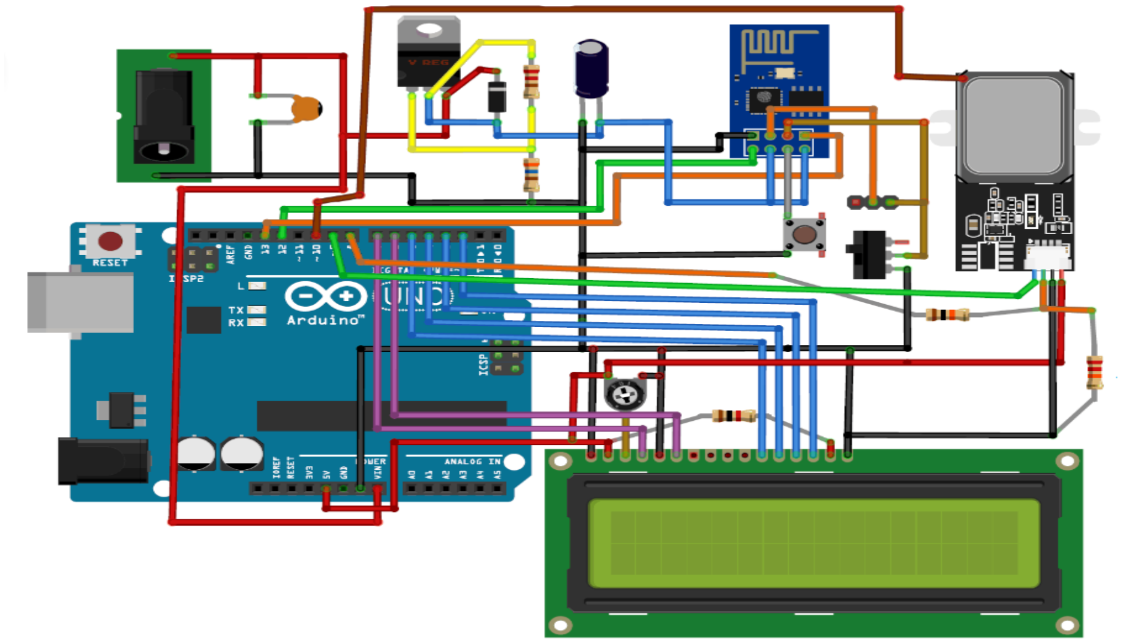


Figure 13: Hardware Setup Diagram

It has Arduino for controlling all the process of the project, push button for enrolling, deleting, selecting IDs and for attendance, a buzzer for alerting, LEDs for indication and LCD to instruct user and showing the resultant messages.
As shown in the circuit diagram, a push button is directly connected to pin A0(ENROL), A1(DEL), A2(UP), A3(DOWN) of Arduino with respect to the ground And Yellow LED is connected at Digital pin D7 of Arduino with respect to ground through a 1k resistor. Fingerprint module's Rx and Tx directly connected at Serial pin D2 and D3 (Software Serial) of Arduino. 5v supply is used for powering finger print module taken from Arduino board. A buzzer is also connected at pin A5. A 16x2 LCD is configured in 4-bit mode and its RS, EN, D4, D5, D6, and D7 are directly connected at Digital pin D13, D12, D11, D10,D9, and D8 of Arduino.

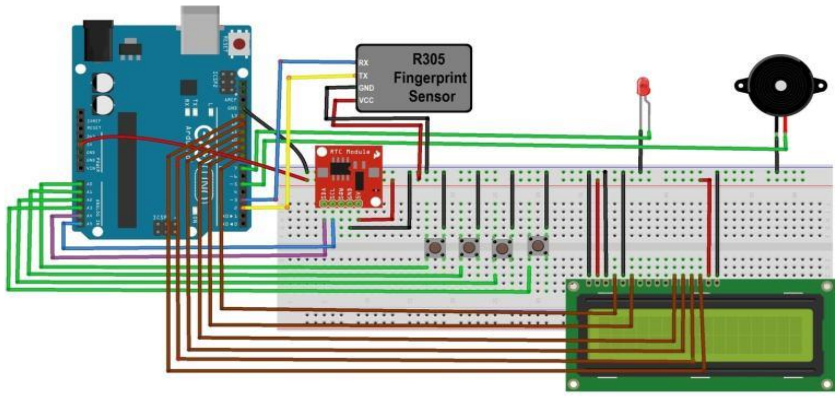


Figure 14: Circuit Diagram of the Setup.

SOFTWARE IMPLEMENTATION:

1. **Arduino IDE :** You will be needing Arduino IDE software to write and upload the programming logic onto the Arduino Uno board.

2. **Thinkboard :** Also, you need to create an account in the Thinkboad IoT platform to integrate the system onto the cloud and store the data online. (WORKS AS A SERIAL MONITOR).

3. **ADAFRUIT FINGERPRINT SENSOR LIBRARY** is used for downloading the data to the serial monitor of the Arduino IDE and for refined capture of the fingerprints.

CHAPTER 5
(Logic & Operation)

5.1 INTRODUCTION

After assembling the system, what remains is to observe its operation and efficiency of the system. 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

5.2.Flow Diagram

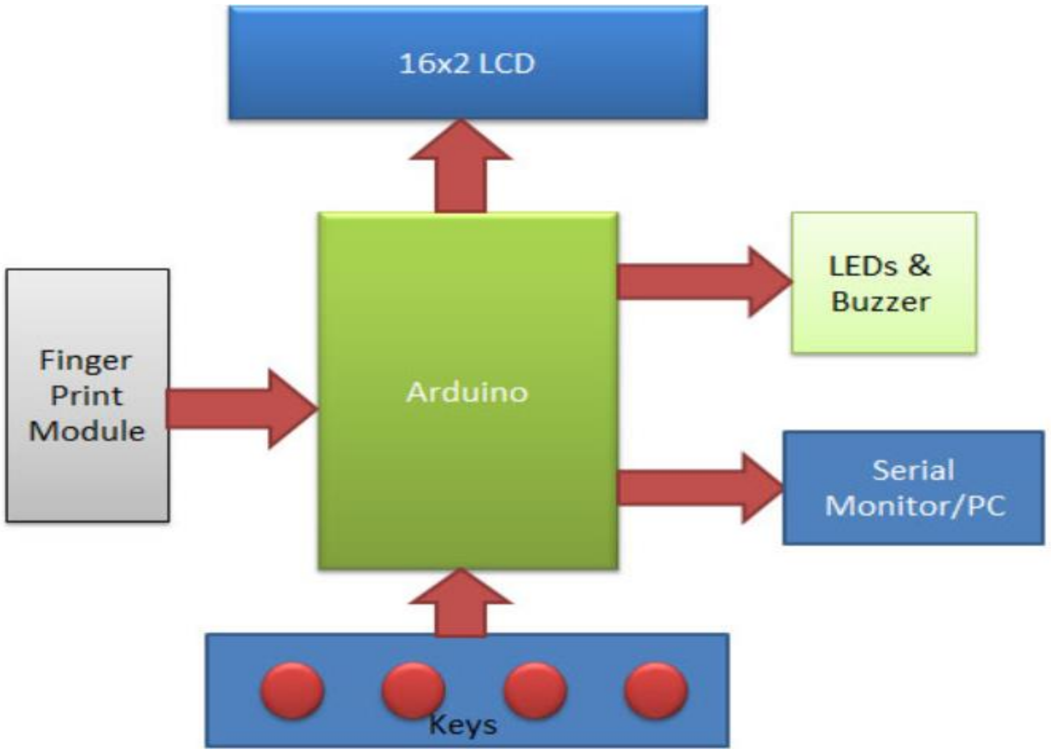


Figure 15: Flowchart of the System

5.3 Principle & Operations

Few years back if you were to tell someone that the Geyser and bedroom lights in your home are connected to internet, they would be baffled and might even criticize it as over-engineered products. But today with the advent of IoT, Smart cities etc the idea no longer sounds strange, we have devices around us that have become smarter by being able to communicate with the internet.

In this project our aim is to leverage this IoT into the boring attendance system to make it smart and more effective. Most conventional attendance systems available today store the information over a micro SD card and have to be connected to software via a computer to access the information. Here, we will build a biometric attendance system using Arduino that scans for finger print and on successful identification of the person it will log the information to a cloud platform like ThingsBoard by using the ESP8266 Wi-Fi module. This information can then be displayed in the dashboard of ThingsBoard making it available for the required authorities to view and analysis information over the internet without having any direct physical access to the hardware. However the conventional Attendance system without involving IoT can also be built by following the link and Finger print sensor can be further used for many other biometric applications like Voting Machine, Security system etc.

5.4 Advantages of the project

● Very accurate fingerprint reading & storing
● Cost Effective
● Can be installed in small spaces
● Fingerprint is stored via cloud
● Can store up to 1000 fingerprints
● LCD display for username, date and time of the operation.
● Alarm signal for attention of the observer.

5.5 Disadvantages of the project

● Limited number of fingerprints are stored
● As the whole thing is connected to internet so any problem related to internet can cause disruption to the whole system
● As fingerprints are stored primarily in EEPROM so that can create some problems

5.6 Coding Logic

The **fingerprint attendance system code for arduino** is given in the subsequent sections. Although the code is explained well with comments, we are discussing here few important parts of the code. We used fingerprint library for **interfacing finger print module with Arduino board**.

First of all, we include the header file and defines input and output pin and define the macro and declared variables. After this, in setup function, we give direction to defined pin and initiate LCD and finger print module

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.

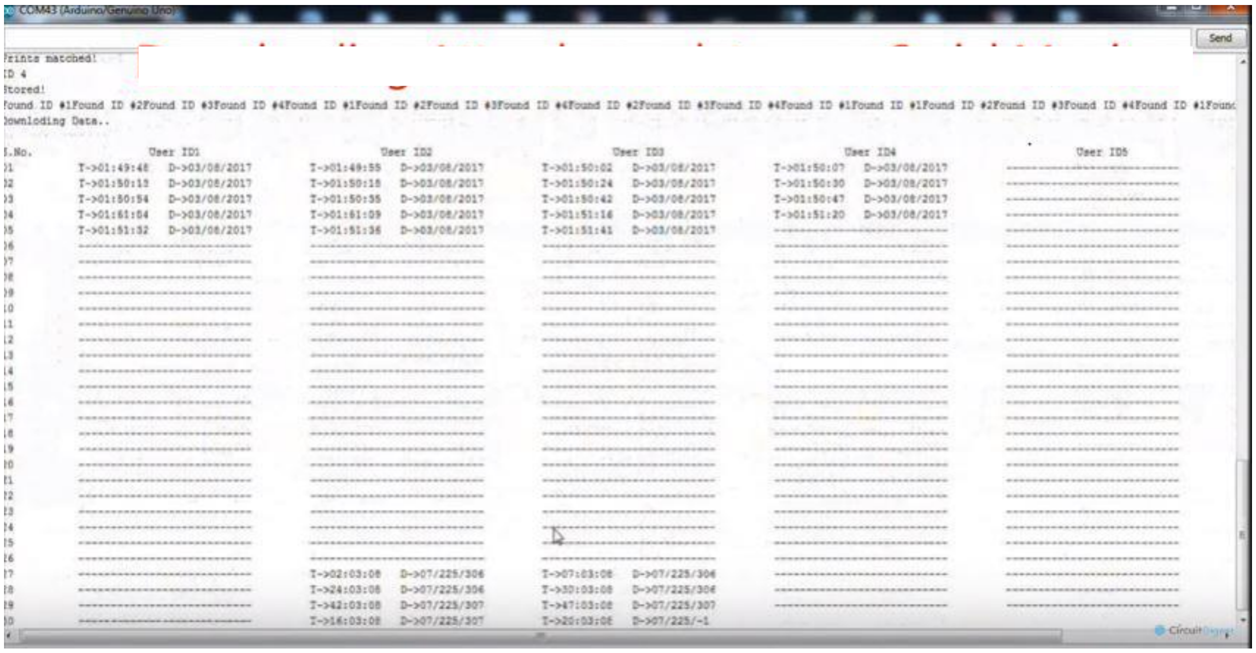
Given function is used for storing attendance time and date in the allotted slot of EEPROM

Given function is used to fetching data from EEPROM and send to serial monitor.

5.7 Datasheet from Serial Monitor:

Using the Adafruit Fingerprint Sensor library, we add the library for the fingerprint sensor to sense different patterns of fingerprint onto the arduino unit for checking and enrolment is done.

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.



```
COM43 (Arduino/Genuino Uno)
                                                                                                    Send
Prints matched!
ID 4
Stored!
Found ID #1Found ID #2Found ID #3Found ID #4Found ID #1Found ID #2Found ID #3Found ID #4Found ID #2Found ID #3Found ID #4Found ID #1Found ID #1Found ID #2Found ID #3Found ID #4Found ID #1Foun
Downloding Data..

S.No.          User ID1                         User ID2                         User ID3                         User ID4                         User ID5
01       T->01:49:48   D->03/08/2017      T->01:49:55   D->03/08/2017      T->01:50:02   D->03/08/2017      T->01:50:07   D->03/08/2017      ---------------------------
02       T->01:50:13   D->03/08/2017      T->01:50:18   D->03/08/2017      T->01:50:24   D->03/08/2017      T->01:50:30   D->03/08/2017      ---------------------------
03       T->01:50:54   D->03/08/2017      T->01:50:35   D->03/08/2017      T->01:50:42   D->03/08/2017      T->01:50:47   D->03/08/2017      ---------------------------
04       T->01:61:04   D->03/08/2017      T->01:61:09   D->03/08/2017      T->01:51:16   D->03/08/2017      T->01:51:20   D->03/08/2017      ---------------------------
05       T->01:51:32   D->03/08/2017      T->01:51:36   D->03/08/2017      T->01:51:41   D->03/08/2017      ---------------------------      ---------------------------
06       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
07       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
08       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
09       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
10       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
11       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
12       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
13       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
14       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
15       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
16       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
17       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
18       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
19       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
20       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
21       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
22       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
23       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
24       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
25       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
26       ---------------------------      ---------------------------      ---------------------------      ---------------------------      ---------------------------
27       ---------------------------      T->02:03:08   D->07/225/306      T->07:03:08   D->07/225/306      ---------------------------      ---------------------------
28       ---------------------------      T->24:03:08   D->07/225/306      T->30:03:08   D->07/225/306      ---------------------------      ---------------------------
29       ---------------------------      T->42:03:08   D->07/225/307      T->47:03:08   D->07/225/307      ---------------------------      ---------------------------
30       ---------------------------      T->16:03:08   D->07/225/307      T->20:03:08   D->07/225/-1
```

Table 06: Data sheet of collected data on Arduino Serial Monitor

5.8 Circuit Diagram

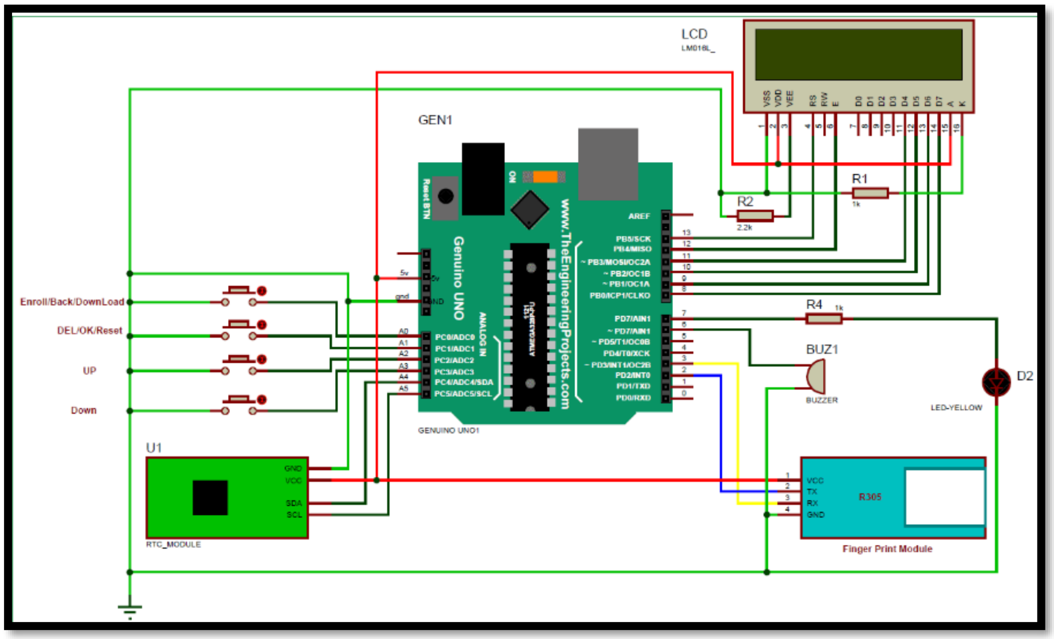


Figure 16: Circuit Diagram of Setup

5.9 Photograph of the Prototype:

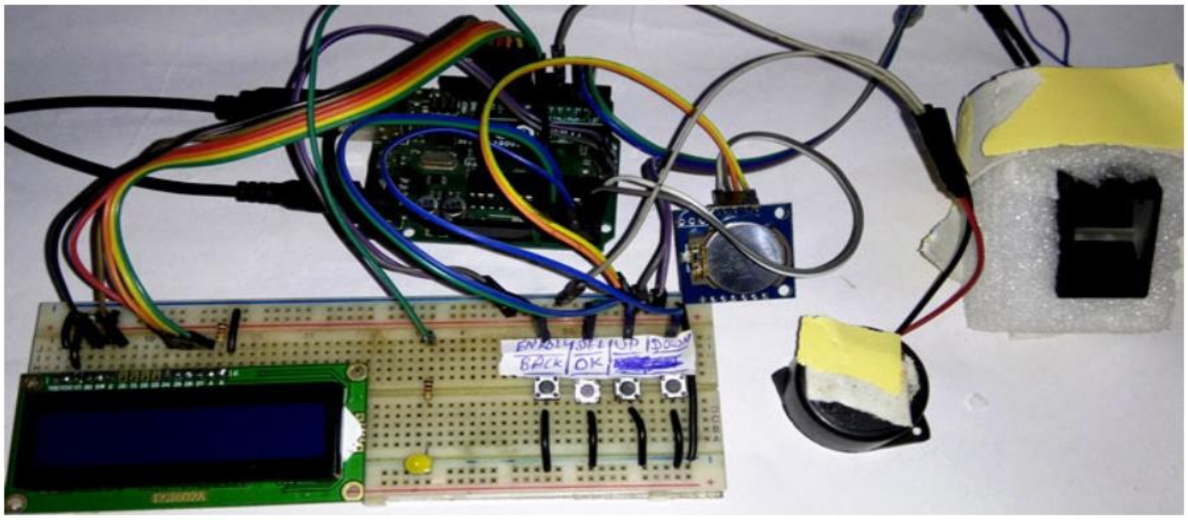


Figure 17: Actual Setup of the entire system

CHAPTER 6

(Conclusion & Future Scope)

6.1 CONCLUSION

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.

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 lecture-wise 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.

6.2 RESULTS

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.

6.3 FUTURE WORK

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.

6.3.1 DIFFERENT WAYS OF MAKING AND USAGE OF BIOMETRIC ATTENDANCE SYSTEM:

8051 microcontroller, R307 optical fingerprint sensor and LabVIEW.

Advantages:

1. User friendly (LabVIEW graphical interface)
2. High speed
3. Efficient and low cost embedded platform
4. Low power consumption
5. Attendance report generation using LabVIEW

Disadvantages:

1. For small databases only
2. Limited Functionality
3. Two microcontrollers are used

Internet of things :

Advantages:

1. More secure (Additional Vein recognition)
2. Internet of things (Everything can be done online)
3. Information can be accessed anywhere

Disadvantages:

1. Complex software system architecture.
2. High cost

GSM and ZigBee

Advantages:

1. Easy to use.
2. Portable (Wireless)

3. Low power consumption

4. Additional functionality (due to GSM)

Disadvantages:

1. Less range of ZigBee (10-20 metres)
2. Low data rate of ZigBee.
3. High cost (both GSM and Zigbee)

<u>RFID and Android</u>

Advantages:

1. More secure (RFID+ Biometrics)
2. More functionality
3. System can be accessed remotely (via android application)
4. Attendance performance graph will be generated
5. RFID cards can serve as library card, mess card
6. RFID cards are difficult to tamper
7. Free bulk SMS service used instead of GSM (Lower cost)

Disadvantages:

1. Complex software design
2. Android application development difficult.

<u>ZigBee, DSP and MATLAB</u>

Advantages:

1. Low power consumption
2. Good accuracy (Gabor image enhancement).
3. Flexible user modes
4. High resolution of fingerprint sensor
5. Wireless (portable)
6. Low cost

Disadvantages:

1. Three different supply voltages required (3.3V, 5V, 12V)

Cryptography

Advantages:

1. User friendly design
2. Portable
3. Small size
4. Security enhancement (encryption of data)
5. Faster than fixed fingerprint reader

Disadvantages:

1. Limited battery life
2. Limited Functionality

RFID, GSM and .Net

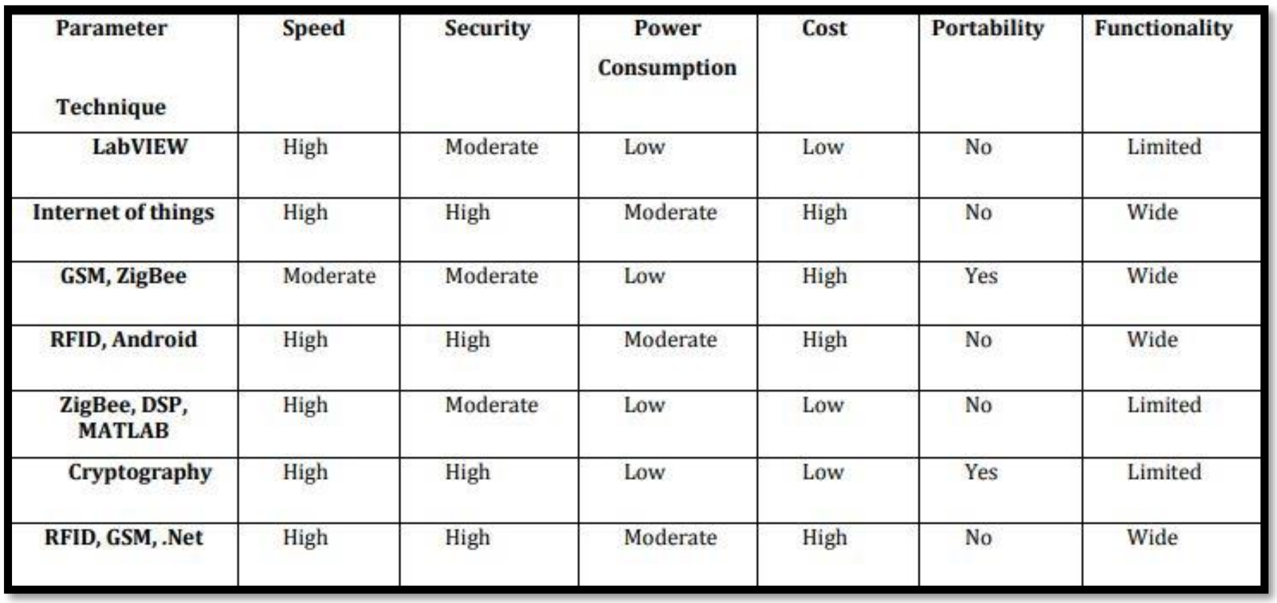
Advantages:

1. More Secure due to RFID and biometrics
2. Complete system is automated
3. Small size of RFID cards
4. Fast processing speed
5. No line of sight required for RFID
6. Many tags can be read simultaneously
7. .net framework simplifies debugging

Disadvantages:

1. Software design is difficult.
2. System should always be kept ON
3. Costly

COMPARATIVE ANALYSIS OF FINGERPRINT ATTENDANCE SYSTEMS:



Parameter / Technique	Speed	Security	Power Consumption	Cost	Portability	Functionality
LabVIEW	High	Moderate	Low	Low	No	Limited
Internet of things	High	High	Moderate	High	No	Wide
GSM, ZigBee	Moderate	Moderate	Low	High	Yes	Wide
RFID, Android	High	High	Moderate	High	No	Wide
ZigBee, DSP, MATLAB	High	Moderate	Low	Low	No	Limited
Cryptography	High	High	Low	Low	Yes	Limited
RFID, GSM, .Net	High	High	Moderate	High	No	Wide

Table 07: Comparative analysis of different systems

CHAPTER 7
(References)

- https://ijarcce.com/wp-content/uploads/2016/11/IJARCCE-ICRITCSA-3.pdf
- www.irjet.net/uploads/2016/05/ IRJET-V3I5237.pdf
- https://www.skyfilabs.com/project-ideas/biometric-attendance-system-with-iot
- https://www.how2electronics.com/fingerprint-sensor-based-biometric-attendance-system/
- https://circuitdigest.com/microcontroller-projects/fingerprint-attendance-system-using-arduino-uno
- Wang and Jingli , "The Design of Teaching Management System in Universities Based on Biometrics Identification and the Internet of Things Technology", IEEE 10th International Conference on Computer Science & Education (ICCSE), Cambridge University, UK July 22-24, 2015, pp. 979-982.

APPENDIX A
(Hardware Description)

ARDUINO UNO- **Arduino Uno** is a microcontroller board based on the ATmega328P datasheet. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. 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.. You can tinker with your Uno without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

TECHNICAL SPECIFICATIONS:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 Ma
DC Current for 3.3V Pin	50 Ma
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13

Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figure 18: ARDUINO UNO

ATmega328/P Microcontroller: The high-performance Microchip picoPower 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.

By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.

Parameters:

Program Memory Type: Flash

Program Memory Size (KB): 32

CPU Speed (MIPS/DMIPS): 20

SRAM (B): 2,048

Data EEPROM/HEF (bytes): 1024

Digital Communication Peripherals: 1-UART, 2-SPI, 1-I2C

Capture/Compare/PWM Peripherals: 1 Input Capture, 1 CCP, 6PWM

Timers: 2 x 8-bit, 1 x 16-bit

Number of Comparators: 1

Temperature Range (°C): -40 to 85

Operating Voltage Range (V): 1.8 to 5.5

Pin Count: 32

Low Power: Yes

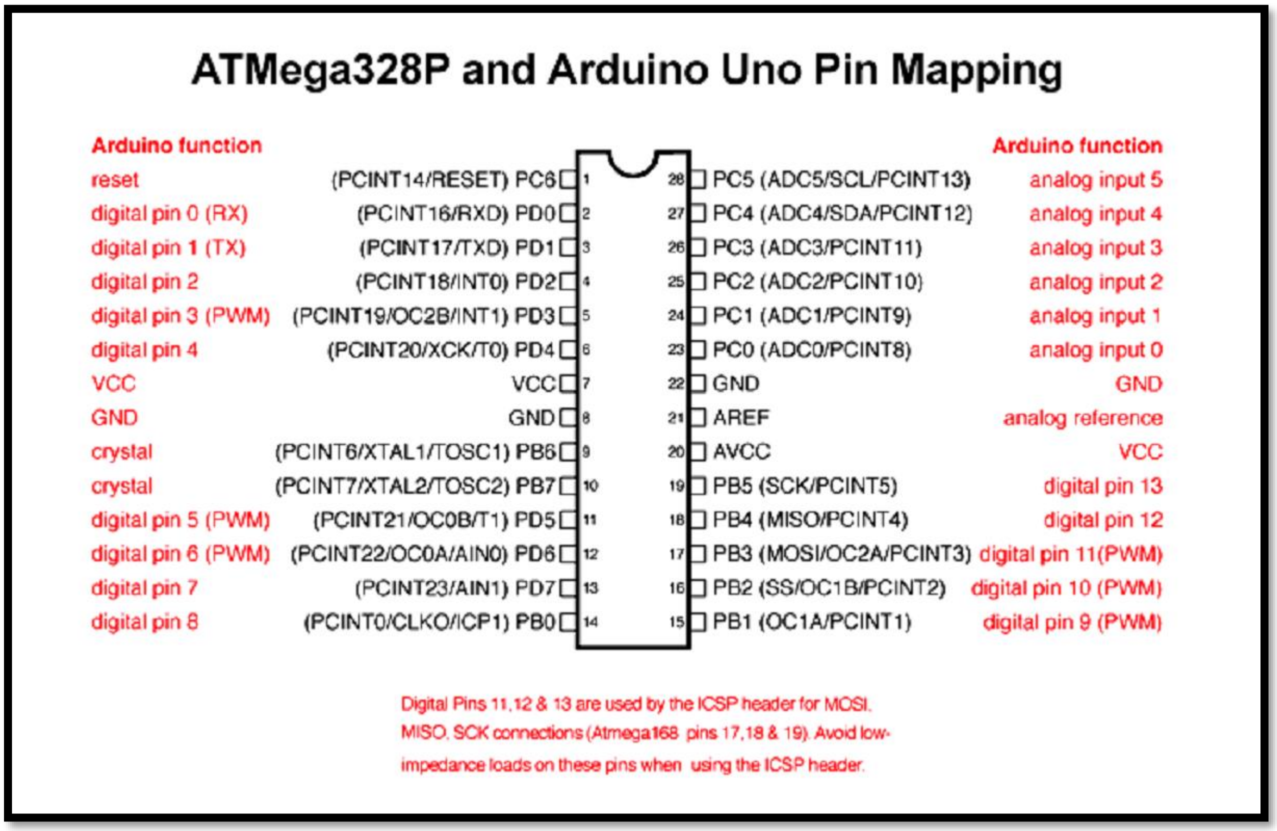

ATMega328P and Arduino Uno Pin Mapping
Arduino function
Arduino function
reset (PCINT14/RESET) PC6 1
28 PC5 (ADC5/SCL/PCINT13) analog input 5
digital pin 0 (RX) (PCINT16/RXD) PD0 2
27 PC4 (ADC4/SDA/PCINT12) analog input 4
digital pin 1 (TX) (PCINT17/TXD) PD1 3
26 PC3 (ADC3/PCINT11) analog input 3
digital pin 2 (PCINT18/INT0) PD2 4
25 PC2 (ADC2/PCINT10) analog input 2
digital pin 3 (PWM) (PCINT19/OC2B/INT1) PD3 5
24 PC1 (ADC1/PCINT9) analog input 1
digital pin 4 (PCINT20/XCK/T0) PD4 6
23 PC0 (ADC0/PCINT8) analog input 0
VCC VCC 7
22 GND GND
GND GND 8
21 AREF analog reference
crystal (PCINT6/XTAL1/TOSC1) PB6 9
20 AVCC VCC
crystal (PCINT7/XTAL2/TOSC2) PB7 10
19 PB5 (SCK/PCINT5) digital pin 13
digital pin 5 (PWM) (PCINT21/OC0B/T1) PD5 11
18 PB4 (MISO/PCINT4) digital pin 12
digital pin 6 (PWM) (PCINT22/OC0A/AIN0) PD6 12
17 PB3 (MOSI/OC2A/PCINT3) digital pin 11(PWM)
digital pin 7 (PCINT23/AIN1) PD7 13
16 PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
digital pin 8 (PCINT0/CLKO/ICP1) PB0 14
15 PB1 (OC1A/PCINT1) digital pin 9 (PWM)
Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Figure 19: ATmega328/P and Arduino Uno Pin Mapping

<u>FINGERPRINT SCANNER (R307):</u> R307 Fingerprint Module consists of optical fingerprint sensor, high-speed DSP processor, high-performance fingerprint alignment algorithm, high-capacity FLASH chips and other hardware and software composition, stable performance, simple structure, with fingerprint entry, image processing, fingerprint matching, search and template storage and other functions.

Features:

- Perfect function: independent fingerprint collection, fingerprint registration, fingerprint comparison (1: 1) and fingerprint search (1: N) function.
- Small size: small size, no external DSP chip algorithm, has been integrated, easy to install, less fault.
- Ultra-low power consumption: low power consumption of the product as a whole, suitable for low-power requirements of the occasion.
- Anti-static ability: a strong anti-static ability, anti-static index reached 15KV above.
- Application development is simple: developers can provide control instructions, self-fingerprint application product development, without the need for professional knowledge of fingerprinting.
- Adjustable security level: suitable for different applications, security levels can be set by the user to adjust.
- Finger touch sensing signal output, low effective, sensing circuit standby current is very low, less than 5uA.

Interface Description:

The R307 fingerprint module has two interface TTL UART and USB2.0, USB2.0 interface can be connected to the computer; RS232 interface is a TTL level, the default baud rate is 57600 , can be changed, refer to a communication protocol ; can And microcontroller, such as ARM, DSP and other serial devices with a connection, 3.3V 5V microcontroller can be connected directly. Needs to connect the computer level conversion, level conversion note , embodiments such as a MAX232 circuit.

About the module's power supply:

Fingerprint module board marked with 3.3V - 2 contacts short circuit, you can use DC3.3V .

Technical Parameters:

- Supply voltage: DC 4.2 ~ 6.0V
- Supply current: Working current: 50mA (typical) Peak current: 80mA
- Fingerprint image input time: <0.3 seconds
- Window area: 14x18 mm
- Matching method: Comparison method (1: 1)
- Search method (1: N)
- Characteristic file: 256 bytes
- Template file: 512 bytes
- Storage capacity: 1000 pieces
- Security Level: Five (from low to high: 1,2,3,4,5)
- Fake rate (FAR): <0.001%
- Refusal rate (FRR): <1.0%
- Search time: <1.0 seconds (1: 1000 hours, mean value)
- Host interface: UART \ USB1.1
- Communication baud rate (UART): (9600xN) bps Where N = 1 ~ 12 (default N = 6, ie 57600bps)
- Working environment: Temperature: -20 ℃ - +40 ℃ Relative humidity: 40% RH-85% RH (no condensation)
- Storage environment: Temperature: -40 ℃ - +85 ℃ Relative humidity: <85% H (no condensation)

  Suitable for fingerprint lock, fingerprint safes and other purposes.

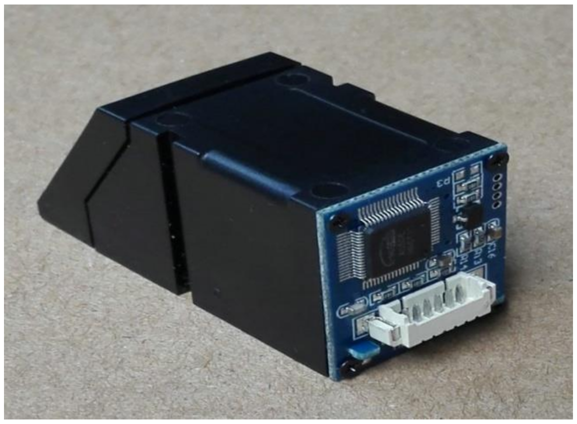
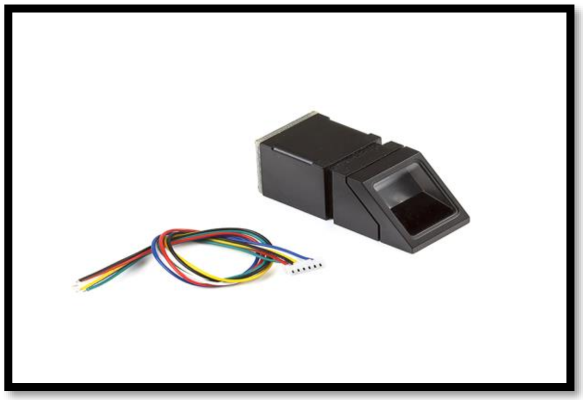



Figure 20: R307 FINGERPRINT SENSOR

16*2 LCD DISPLAY:

- Operating Voltage is 4.7V to 5.3V
- Current consumption is 1mA without backlight
- Alphanumeric LCD display module, meaning can display alphabets and numbers
- Consists of two rows and each row can print 16 characters.
- Each character is build by a 5×8 pixel box
- Can work on both 8-bit and 4-bit mode
- It can also display any custom generated characters
- Available in Green and Blue Backlight.

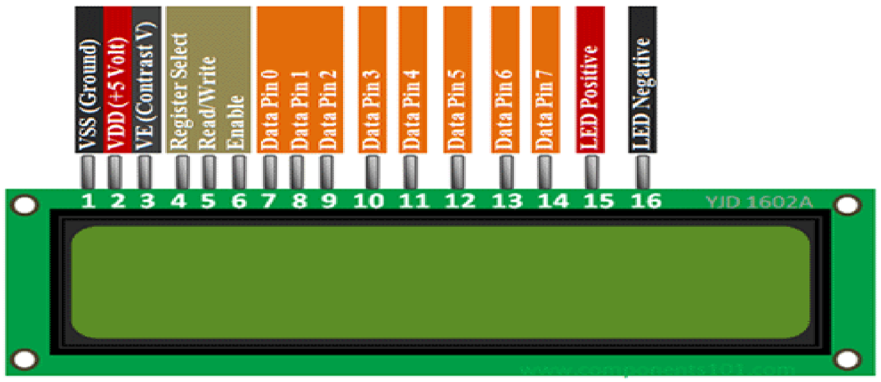


Figure 21: LCD setup and pins

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. Most of us would have come across these displays in our day to day life, either at PCO's or calculators. The appearance and the pinouts have already been visualized above now let us get a bit technical.

16×2 LCD is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. but the most used one is the 16×2 LCD. So, it will have (16×2=32) 32 characters in total and each character will be made of 5×8 Pixel Dots.

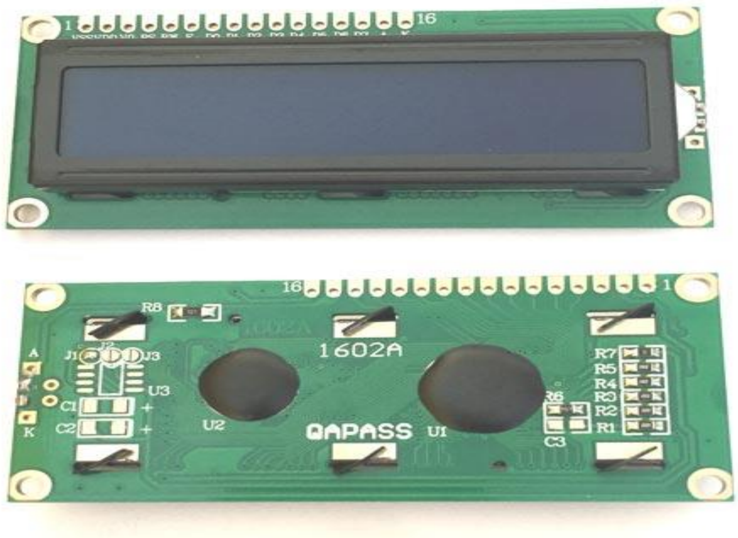


Figure 22: 16*2 LCD panel

Now, we know that each character has (5×8=40) 40 Pixels and for 32 Characters we will have (32×40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels. Hence it will be a hectic task to handle everything with the help of MCU, hence an **Interface IC like HD44780** is used, which is mounted on the backside of the LCD Module itself. The function of this IC is to get the **Commands and Data** from the MCU and process them to display meaningful information onto our LCD Screen.

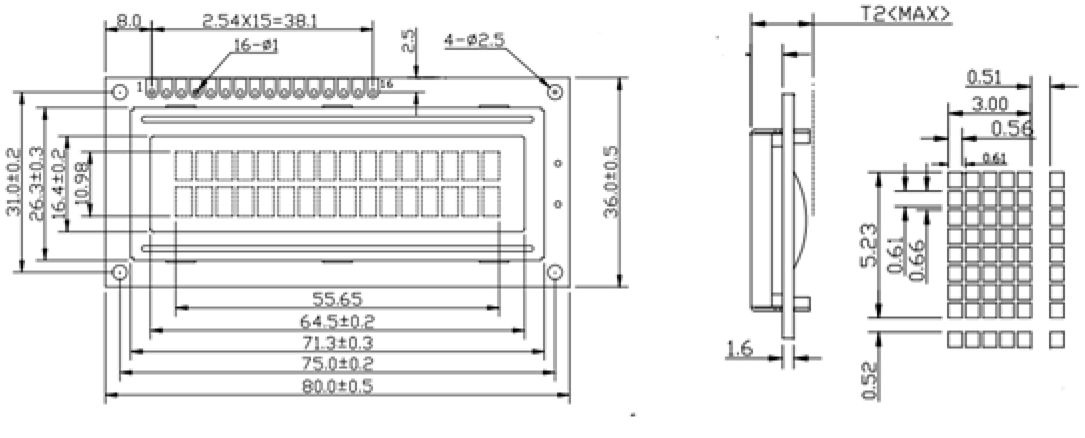


Figure 23: 2D Model Of A 16*2 Lcd Display

WiFi Module - ESP8266: The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that's just out of the box)! The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts.

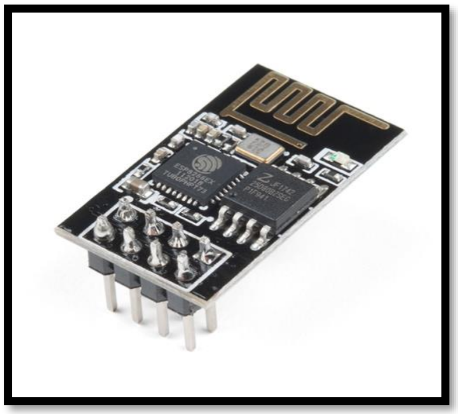


Figure 24: ESP8266 Wifi Module

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 un-pushed state.

Figure 25:Push Button

<u>Resistor:</u>

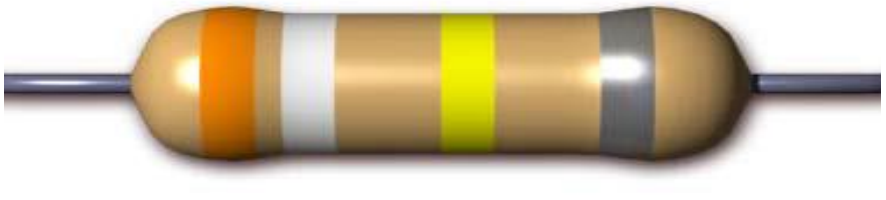


Figure 26: Resistor

Resistance is the opposition of a material to the current. It is measured in Ohms Ω. All conductors represent a certain amount of resistance, since no conductor is 100% efficient. To control the electron flow (current) in a predictable manner, we use resistors. Electronic circuits use calibrated lumped resistance to control the flow of current. Broadly speaking, resistor can be divided into two groups viz. fixed & adjustable (variable) resistors. In fixed resistors, the value is fixed & cannot be varied. In variable resistors, the resistance value can be varied by an adjuster knob. It can be divided into (a) Carbon composition (b) Wire wound (c) Special type. The most common type of resistors used in our projects is carbon type. The resistance value is normally indicated by color bands. Each resistance has four colors, one of the band on either side will be gold or silver, this is called fourth band and indicates the tolerance, others three band will give the value of resistance (see table). For example if a resistor has the following marking on it say red, violet, gold. Comparing these colored rings with the color code, its value is 27000 ohms or 27 kilo ohms and its tolerance is ±5%. Resistor comes in various sizes (Power rating).The bigger the size, the more power rating of 1/4 watts. The four color rings on its body tells us the value of resistor.value.

Color Code of the resistor:

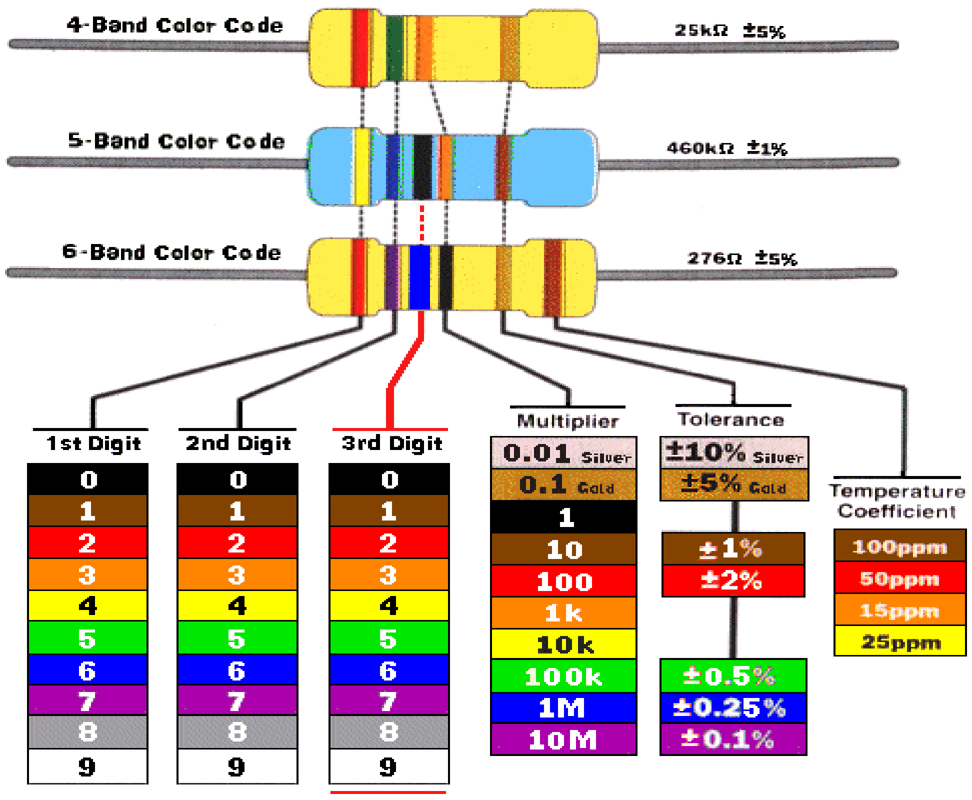


Figure 27: Color Code Of Resistor

<u>Piezo buzzer:</u>

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke. A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Figure 28: Piezobuzzer

BREADBOARD: A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to **prototype** (meaning to build and test an early version of) an electronic circuit, like this one with a battery, switch, resistor, and an LED (light-emitting diode). Modern breadboards are made from plastic, and come in all shapes, sizes, and even different colors. While larger and smaller sizes are available, the most common sizes you will probably see are "full-size," "half-size," and "mini" breadboards. Most breadboards also come with tabs and notches on the sides that allow you to snap multiple boards together. However, a single half-sized breadboard is sufficient for many beginner-level projects.

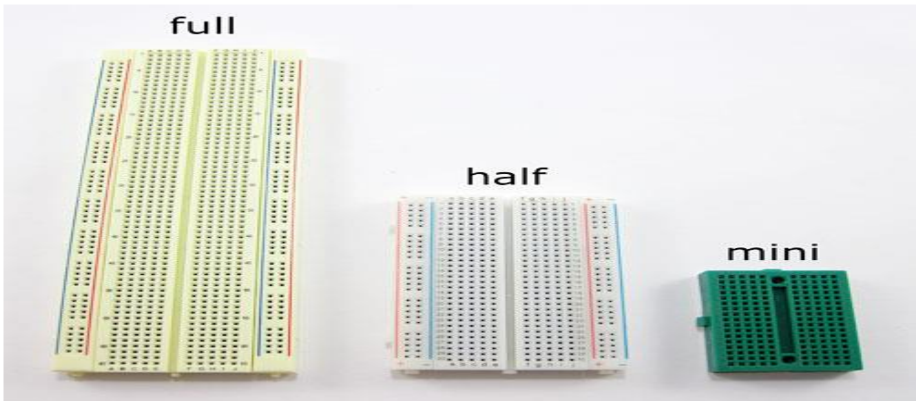


Figure 29: Breadboard

DS3231 RTC MODULE: At the heart of the module is a low-cost, extremely accurate RTC chip from Maxim – DS3231. It manages all timekeeping functions and features a simple two-wire I2C interface which can be easily interfaced with any microcontroller of your choice. The chip maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year (valid up to 2100).

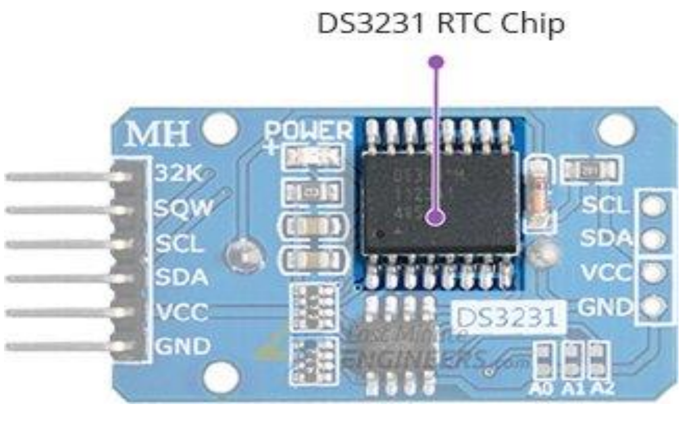


Figure 30: DS3231 RTC Chip

The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. It also provides two programmable time-of-day alarms.

The other cool feature of this board comes with SQW pin, which outputs a nice square wave at either 1Hz, 4kHz, 8kHz or 32kHz and can be handled programmatically. This can further be used as an interrupt due to alarm condition in many time-based applications.

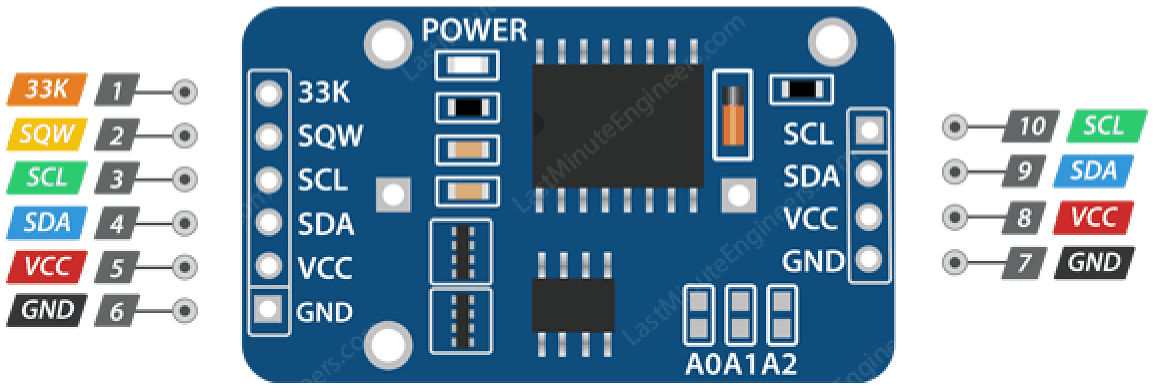


Figure 31: DS3231 RTC Module Pinouts

33K pin outputs the stable(temperature compensated) and accurate reference clock.

SQW pin outputs a nice square wave at either 1Hz, 4kHz, 8kHz or 32kHz and can be handled programmatically. This can further be used as an interrupt due to alarm condition in many time-based applications.

SCL is a serial clock pin for I2C interface.

SDA is a serial data pin for I2C interface.

VCC pin supplies power for the module. It can be anywhere between 3.3V to 5.5V.

GND is a ground pin.

APPENDIX B

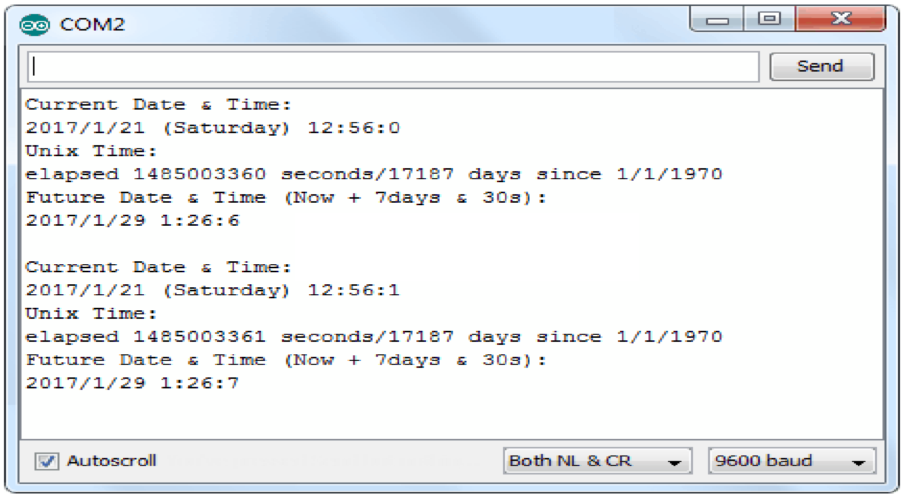
(Software coding)

CODE FOR THINGSBOARD SETUP:

```
#include <PubSubClient.h>
//http://pubsubclient.knolleary.net/
#include <ESP8266WiFi.h>
//https://github.com/bportaluri/WiFiEsp
#define WIFI_AP "RCCIIT"
#define WIFI_PASSWORD
"projectgroup6"
#define TOKEN "IFhm5ggJVsEpokiIoQ"
char thingsboardServer[] =
"demo.thingsboard.io";
WiFiClient wifiClient;

void setup()
{
  Serial.begin(9600);
  delay(10);
  InitWiFi();
  client.setServer( thingsboardServer, 1883 );
  lastSend = 0;
}

void loop()
{
  if ( !client.connected() ) {
    reconnect();
  }
  if ( Serial.available() ) { // Update and send
only after 1 seconds
    char a = Serial.read();
    Name = Name + String(a);
    if (a == 13) //check for new line
    {
    Name.trim(); //Remove /n or /r from the
incomind data
    Serial.println(Name);
    Send_to_Thingsboard();
    Name =""; //clear the string if new line is
detected
    }
  }


  client.loop();
}

void Send_to_Thingsboard()
{
  Serial.println("Collecting temperature
data.");

  String Employee_Name = Name;

  String payload = "{";
  payload += "\"Name\":"; payload +=
Employee_Name;
  payload += "}";

  char attributes[100];
  payload.toCharArray( attributes, 100 );
  client.publish( "v1/devices/me/telemetry",
attributes );
  Serial.println( attributes );
}
```

CODE FOR DS3231 LIBRARY SETUP: RTC MODULE.

```
#include <Wire.h>
#include "RTClib.h"

RTC_DS3231 rtc;

char daysOfTheWeek[7][12] = {"Sunday",
"Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};

void setup ()
{
  Serial.begin(9600);
  delay(3000); // wait for console opening

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

  if (rtc.lostPower()) {
    Serial.println("RTC lost power, lets set
the time!");

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

    // Following line sets the RTC with an
explicit date & time
    // for example to set June 20 2020 at
12:56 you would call:
    // rtc.adjust(DateTime(2020, 6, 20, 12, 56,
0));
  }
}

void loop ()
{
    DateTime now = rtc.now();

    Serial.println("Current Date & Time: ");
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" (");

Serial.print(daysOfTheWeek[now.dayOfThe
Week()]);
    Serial.print(") ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    Serial.println("Unix Time: ");
    Serial.print("elapsed ");
    Serial.print(now.unixtime());
    Serial.print(" seconds/");
    Serial.print(now.unixtime() / 86400L);
    Serial.println(" days since 1/1/1970");
    // calculate a date which is 7 days & 30
seconds into the future
    DateTime future (now +
TimeSpan(7,0,0,30));

    Serial.println("Future Date & Time (Now
+ 7days & 30s): ");
    Serial.print(future.year(), DEC);
    Serial.print('/');
    Serial.print(future.month(), DEC);
    Serial.print('/');
    Serial.print(future.day(), DEC);
    Serial.print(' ');
    Serial.print(future.hour(), DEC);
    Serial.print(':');

    Serial.print(future.minute(), DEC);
    Serial.print(':');
    Serial.print(future.second(), DEC);
    Serial.println();

    Serial.println();
    delay(1000);
}
```



```
Current Date & Time:
2017/1/21 (Saturday) 12:56:0
Unix Time:
elapsed 1485003360 seconds/17187 days since 1/1/1970
Future Date & Time (Now + 7days & 30s):
2017/1/29 1:26:6

Current Date & Time:
2017/1/21 (Saturday) 12:56:1
Unix Time:
elapsed 1485003361 seconds/17187 days since 1/1/1970
Future Date & Time (Now + 7days & 30s):
2017/1/29 1:26:7
```

Figure 32: Serial Monitor for DS3231 code.

CODE FOR EEPROM READ/WRITE:

```
#include <Wire.h>

void setup()
{
  char somedata[] = "rcciitprojectgroup6";
// data to write
  Wire.begin(); // initialise the connection
  Serial.begin(9600);
  Serial.println("Writing into memory...");

      // write to EEPROM
  i2c_eeprom_write_page(0x57, 0, (byte
*)somedata, sizeof(somedata));
```

```
    delay(100); //add a small delay
    Serial.println("Memory written");
}

void loop()
{
    Serial.print("Reading memory: ");
    int addr=0; //first address

    byte b = i2c_eeprom_read_byte(0x57, 0);

    while (b!=0)
    {
        Serial.print((char)b); //print content to
serial port
        addr++; //increase address
        b = i2c_eeprom_read_byte(0x57, addr);
//access an address from the memory
    }
    Serial.println(" ");
    delay(2000);
}

void i2c_eeprom_write_byte( int
deviceaddress, unsigned int eeaddress, byte
data ) {
    int rdata = data;
    Wire.beginTransmission(deviceaddress);
    Wire.write((int)(eeaddress >> 8)); // MSB
    Wire.write((int)(eeaddress & 0xFF)); //
LSB
    Wire.write(rdata);
    Wire.endTransmission();
}

void i2c_eeprom_write_page( int
deviceaddress, unsigned int eeaddresspage,
byte* data, byte length ) {
    Wire.beginTransmission(deviceaddress);
    Wire.write((int)(eeaddresspage >> 8)); //
MSB
    Wire.write((int)(eeaddresspage & 0xFF));
// LSB
    byte c;
    for ( c = 0; c < length; c++)
        Wire.write(data[c]);
    Wire.endTransmission();
}

byte i2c_eeprom_read_byte( int
deviceaddress, unsigned int eeaddress ) {
    byte rdata = 0xFF;
    Wire.beginTransmission(deviceaddress);
    Wire.write((int)(eeaddress >> 8)); // MSB
    Wire.write((int)(eeaddress & 0xFF)); //
LSB
    Wire.endTransmission();
    Wire.requestFrom(deviceaddress,1);
    if (Wire.available()) rdata = Wire.read();
    return rdata;
}

void i2c_eeprom_read_buffer( int
deviceaddress, unsigned int eeaddress, byte
*buffer, int length ) {
    Wire.beginTransmission(deviceaddress);
    Wire.write((int)(eeaddress >> 8)); // MSB
    Wire.write((int)(eeaddress & 0xFF)); //
LSB
    Wire.endTransmission();
    Wire.requestFrom(deviceaddress,length);
    int c = 0;
    for ( c = 0; c < length; c++ )
       if (Wire.available()) buffer[c] =
Wire.read();
}
```

SOURCE CODE & ADAFRUIT FINGERPRINT SENSOR LIB AND RTC LIB 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); //for tx/rx
communication between arduino & r307
fingerprint sensor

#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,us
er8,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(2020, 6, 7, 11, 0, 0));
// This line sets the RTC with an explicit
date & time, for example to set
// June 7, 2020 at 11am you would call:
// rtc.adjust(DateTime(2020, 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;
}
```

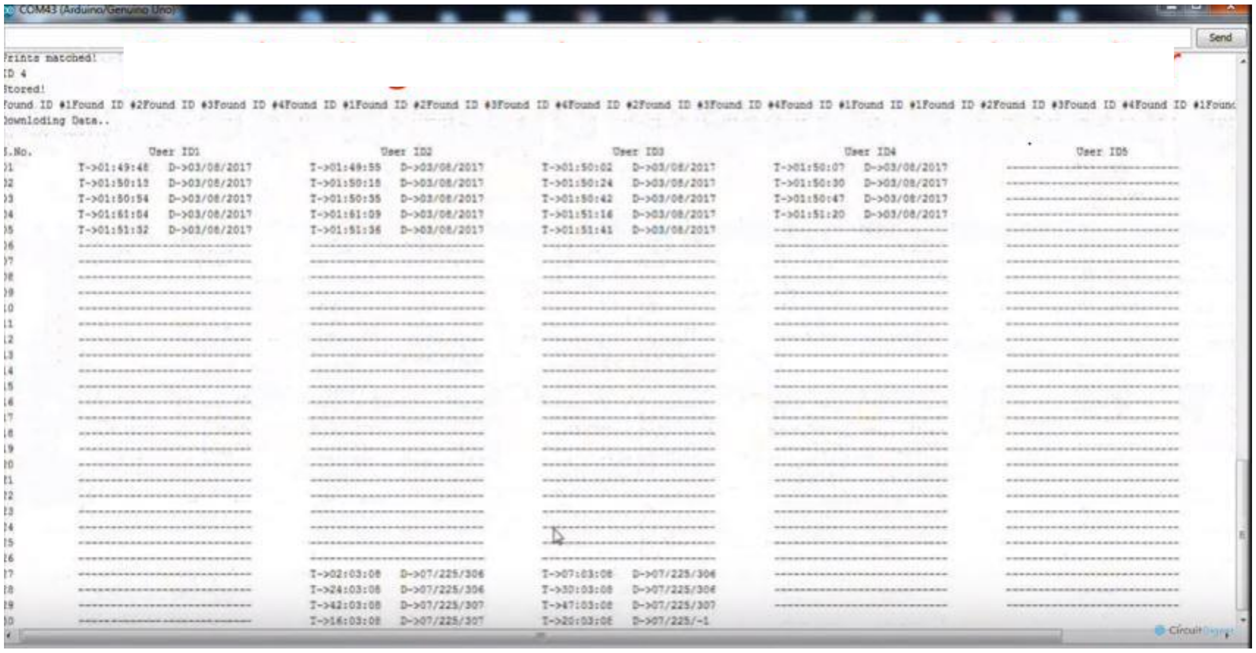
```
// OK converted!
Serial.print("Creating model for #");
Serial.println(id);

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(" ");
}
```

APPENDIX C

(Data sheets)

Datasheet from Serial Monitor:



```
rints matched!
ID 4
tored!
ound ID #1Found ID #2Found ID #3Found ID #4Found ID #1Found ID #2Found ID #3Found ID #4Found ID #2Found ID #3Found ID #4Found ID #1Found ID #1Found ID #2Found ID #3Found ID #4Found ID #1Founc
ownloding Data..

.No.          User ID1                     User ID2                     User ID3                     User ID4                     User ID5
01    T->01:49:48   D->03/08/2017   T->01:49:55   D->03/08/2017   T->01:50:02   D->03/08/2017   T->01:50:07   D->03/08/2017   --------------------------
02    T->01:50:13   D->03/08/2017   T->01:50:18   D->03/08/2017   T->01:50:24   D->03/08/2017   T->01:50:30   D->03/08/2017   --------------------------
03    T->01:50:54   D->03/08/2017   T->01:50:35   D->03/08/2017   T->01:50:42   D->03/08/2017   T->01:50:47   D->03/08/2017   --------------------------
04    T->01:61:04   D->03/08/2017   T->01:61:09   D->03/08/2017   T->01:51:16   D->03/08/2017   T->01:51:20   D->03/08/2017   --------------------------
05    T->01:51:32   D->03/08/2017   T->01:51:36   D->03/08/2017   T->01:51:41   D->03/08/2017   --------------------------   --------------------------
06    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
07    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
08    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
09    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
10    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
11    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
12    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
13    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
14    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
15    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
16    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
17    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
18    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
19    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
20    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
21    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
22    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
23    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
24    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
25    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
26    --------------------------   --------------------------   --------------------------   --------------------------   --------------------------
27    --------------------------   T->02:03:08   D->07/225/306   T->07:03:08   D->07/225/306   --------------------------   --------------------------
28    --------------------------   T->24:03:08   D->07/225/306   T->30:03:08   D->07/225/306   --------------------------   --------------------------
29    --------------------------   T->42:03:08   D->07/225/307   T->47:03:08   D->07/225/307   --------------------------   --------------------------
30    --------------------------   T->16:03:08   D->07/225/307   T->20:03:08   D->07/225/-1
```

Table 08: Data Sheet from Serial Monitor

- The Adafruit Fingerprint Sensor library is installed and whenever user places finger over module then it captures finger image, and search if any ID is associated with the fingerprint in the system. If fingerprint ID is detected then LCD will show Attendance registered and in the same time buzzer will beep once and LED will turn off until the system is ready to take input again.

- Along with the module, we have also used an **RTC module for Time and Date,** so the Arduino takes time and date whenever a user places his finger over module and save them in the EEPROM at the allotted slot of memory.

- We have created 10 user space in this system for 30 days. By pressing the RESET button in Arduino and then immediately enroll key will be responsible for downloading attendance data over serial monitor from the Arduino EEPROM Memory.

ATmega
328P

8-bit AVR Microcontroller with 32K Bytes In-System

Programmable Flash

DATASHEET

Features

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^{2}C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

- I/O and packages
  - 23 programmable I/O lines
  - 32-lead TQFP, and 32-pad QFN/MLF
- Operating voltage:
  - 2.7V to 5.5V for ATmega328P
- Temperature range:
  - Automotive temperature range: –40°C to +125°C
- Speed grade:
  - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
  - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
  - Active mode: 1.5mA at 3V - 4MHz
  - Power-down mode: 1μA at 3V

1. **PIN CONFIGURATION:**

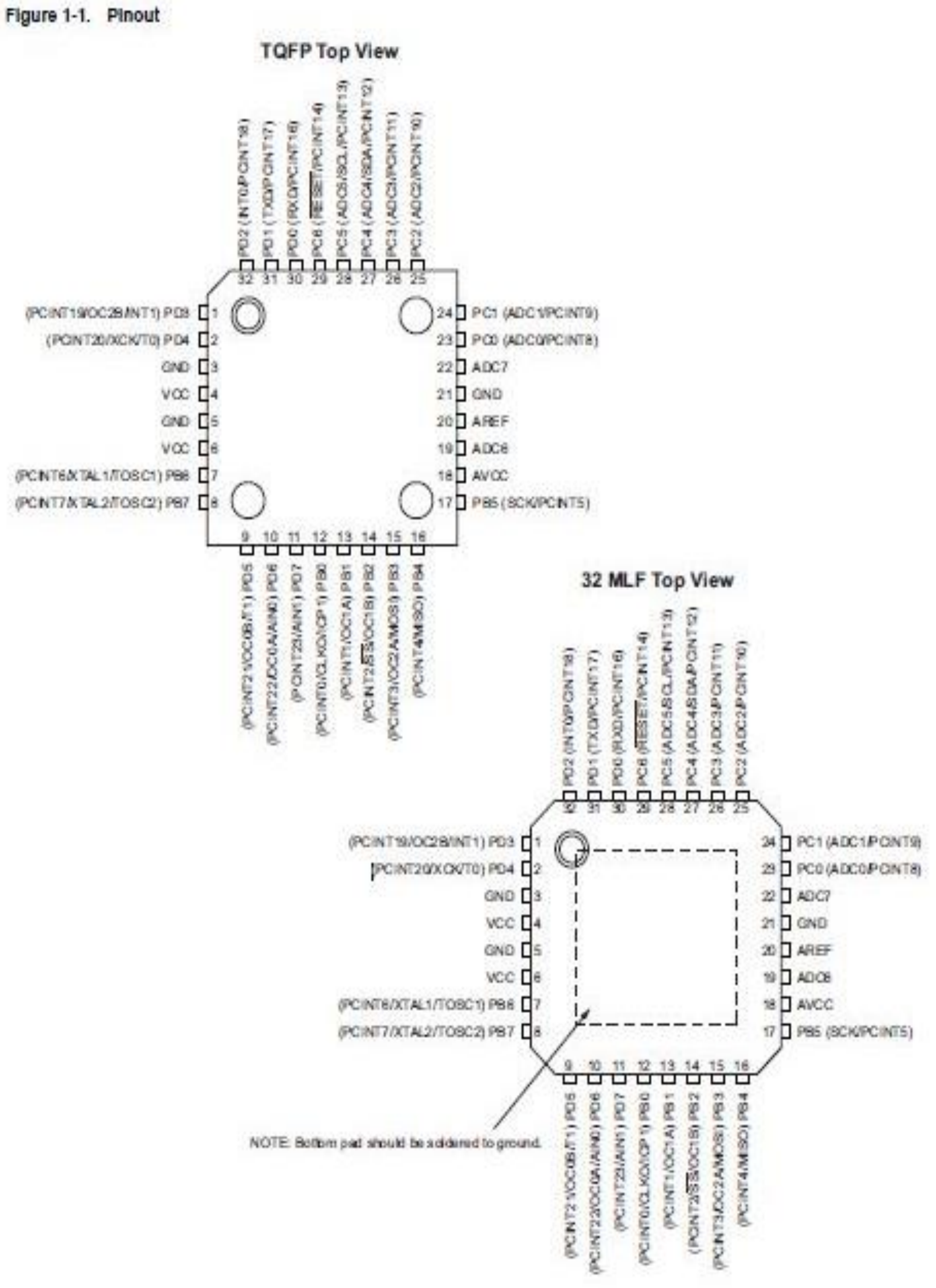


Figure 33: Pin Configuration of ATmega328/P

2. Overview

The Atmel® ATmega328P is a low-power CMOS 8-bit microcontroller based on the

AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328P achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

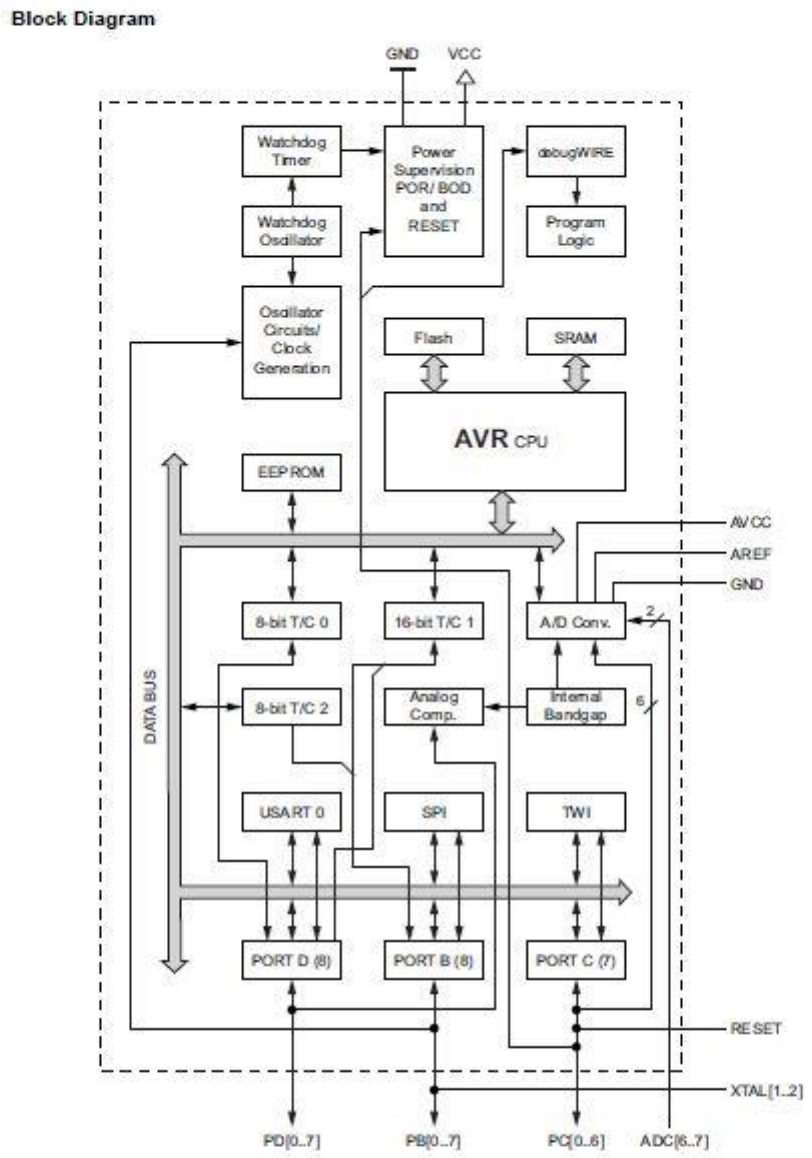


Figure 34: Block Diagram of ATmega328/P

The AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the arithmetic logic unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The Atmel® ATmega328P provides the following features: 32K bytes of in-system programmable flash with read-while-write capabilities, 1K bytes EEPROM, 2K bytes

SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte- oriented 2-wire serial interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire serial interface, SPI port, and interrupt system to continue functioning. The power-down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset. In power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC noise reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel high density non-volatile memory technology. The on-chip ISP flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional non-volatile memory programmer, or by an on-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the application flash memory. Software in the boot flash section will continue to run while the application flash section is updated, providing true read-while-write operation. By combining an 8-bit RISC CPU with
in-system self-programmable flash on a monolithic chip, the Atmel ATmega328P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega328P AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

ATmega328/P Entire Datasheet:
[http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf]