

School Security System Using Finger Print

A PROJECT REPORT

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY in CSE - INTERNET OF THINGS

Submitted by

VAKA.AMULYA
Roll No. 214N1A3563

POKALA.SHASIKALA
Roll No. 214N1A3552

GOLLA.HEMA SWAROOPA
Roll No. 214N1A3521

GOLLAPUDI.THIRUMALA
Roll No. 214N1A3522

Under The Guidance of

Dr. V. V. Sunil Kumar
Professor



**DEPARTMENT OF
CSE - INTERNET OF THINGS**

VISVODAYA ENGINEERING COLLEGE

KAVALI 524201, NELLORE DISTRICT, AP

**Jawaharlal Nehru Technological University Annapur, Ananthapuramu,
AP, India**

APRIL 2025

VISVODAYA ENGINEERING COLLEGE

KAVALI

BONAFIDE CERTIFICATE

Certified that this project report “**SCHOOL SECURITY SYSTEM USING FINGER PRINT**” is the bonafide work done by “**V.AMULYA (214NIA3563), P.SHASIKALA(214NIA3552), G.HEMA SWAROOPA (214NIA3521),G.THIRUMALA (214NIA3522)**”, who carried out the project under my guidance during the year 2024-25, towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Computer Science & Engineering - Internet of Things, from Jawaharlal Nehru Technological University, Anantapur. The results embodied in this report have not been submitted to any other University for the award of any degree.

Dr. V. V. Sunil Kumar

PROJECT SUPERVISOR

Department of CSE - IoT

Mr. P. ESWARAIAH

HEAD OF THE DEPARTMENT

Department of CSE-IoT

External Viva Voce conducted on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

VISVODAYA ENGINEERING COLLEGE

KAVALI

CERTIFICATE OF AUTHENTICATION

We solemnly declare that this project report, “**SCHOOL SECURITY SYSTEM USING FINGER PRINT**” is our original work, carried out under the supervision of **Dr. V. V. Sunil Kumar**, Professor, in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science & Engineering - Internet of Things from Jawaharlal Nehru Technological University, Anantapur, during the academic year 2024-25.

Furthermore, we certify that this work has not been submitted, either in part or in full, to any other department of Jawaharlal Nehru Technological University or any other university, institution, or elsewhere, nor has it been published in any form.

Signature of the Students

Date:

1. V.AMULYA (214N1A3563)

2. P. SHASIKALA(214N1A3552)

3. G. HEMA SWAROOPA (214N1A3521)

4. G. THIRUMALA (214N1A3522)

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to **VISVODAYA ENGINEERING COLLEGE, KAVALI**, for providing us with the opportunity to complete this project report.

We express our deep appreciation and indebtedness to our esteemed Chairman, **Mr. D. VIDYADHARA KUMAR REDDY**, Academic Director **Dr. D. PRATHYUSHA REDDI** and Academic In-Charge **Dr. D. LIKHITH REDDY**, for fostering an excellent academic environment.

It is our privilege to convey our sincere gratitude to our Director, Wg. Cdr. I.P.C. Reddy (Retd.), and Principal, **Dr. N. SESHIAIAH**, for granting us this opportunity to successfully complete our project report.

We are profoundly grateful to our Head of the Department, **Mr. P. ESWARAIAH**, Head & Associate Professor, Department of CSE - IoT, for his invaluable support, essential facilities, guidance, and unwavering encouragement, which played a crucial role in the timely completion of this project.

A special note of appreciation goes to our dedicated guide, **Dr.V.V.SUNIL KUMAR**, Professor, for his inspiring mentorship, continuous encouragement, and insightful guidance throughout this project.

Lastly, we express our sincere gratitude to all our lecturers, lab coordinators, non-teaching staff, and friends who directly or indirectly contributed to the successful completion of this project, offering valuable suggestions and support at every step.

ABSTRACT

The proposed school security system uses finger print sensor integrated with Arduino Uno and NodeMCU to efficiently monitor and record student attendance in real-time. Each student biometric is taken, which is scanned upon entry or exit, triggering the system to log the data. An LCD screen displays the student's name and status, while a buzzer alerts staff to any unauthorized or delayed actions. Additionally, the system send notifications to administrators and logs all attendance data onto a remote server, accessible through a user-friendly interface. This centralized system ensures enhanced security, facilitates real-time monitoring, and provides detailed attendance records for school management.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<i>List of tables</i>	<i>i</i>
	<i>List of figures</i>	<i>ii</i>
1	Introduction	1
1.1	Introduction to the project	1
1.2	Objectives	2
1.3	Existing System	2
1.3.1	Disadvantage of Existing System	2
1.4	Proposed System	3
1.4.1	Advantages of Proposed System	3
2	System Requirements	4
2.1	Software requirements	4
2.2	Hardware requirements	15
3	System Analysis	37
3.1	Architecture	37
3.2	Methodology	38
3.3	Hardware configuration	41
4	System Design	44
4.1	Introduction	44
4.2	UML Diagrams	44
4.2.1	Use Case Diagram	45
4.2.2	Class Diagram	48
4.2.3	Sequence Diagram	49
4.2.4	Activity Diagram	50

5	System Implementation	53
5.1	Implementation	53
5.2	Source code	55
5.3	Output	68
6	System Testing	71
6.1	Types of Testing	71
6.2	Test cases and Results	73
7	Conclusion & Future Enhancement	74
7.1	Conclusion	74
7.2	Future scope	74
8	References	76

List of Tables

Table No.	Table Description	Page No.
3.3.1	Components & working	41
3.3.3(a)	Main unit – pinout	42

List of Figures

Figure No.	Figure Description	Page No.
3.1	System Architecture	27
3.2.2	System flow chart	29-30
3.3.2(a)	Block diagram of main unit	31
4.1	System Design	44
4.2.1	Use case diagram	45
4.2.2	Class diagram	48
4.2.3	Sequence diagram	49
4.2.4	Activity & state diagram	50
5.3.1	Main unit Circuit	68
5.3.3	Arduino cloud dashboard	69
5.3.4	SMS alert notification	70

Chapter:1

INTRODUCTION

1.1 INTRODUCTION TO PROJECT

In today's world, ensuring the safety and security of students and staff within school premises has become increasingly important. Traditional security systems such as ID cards or attendance registers are often inefficient and prone to misuse or error. To overcome these challenges, biometric systems, especially fingerprint recognition, offer a more secure, reliable, and tamper-proof solution.

This project, "School Security System Using Fingerprint", aims to enhance school security by integrating fingerprint-based authentication for controlled access. The system allows only registered individuals—students, teachers, and staff—to enter the school premises or restricted areas by verifying their fingerprint data. It can also be used for automatic attendance tracking, thereby improving transparency and reducing administrative workload.

By using biometric technology, this project ensures accurate identification, minimizes the chances of impersonation, and creates a safer learning environment for all.

This project, School Security System Using Fingerprint, is designed to enhance the safety of students and staff by using biometric technology. Instead of traditional ID cards or manual attendance, the system uses fingerprints to allow only authorized individuals to enter the school premises. It provides a secure, reliable, and easy way to manage access and monitor attendance, helping to create a safer school environment.

OBJECTIVES

The aim of this project is to improve school security by using fingerprint-based authentication. It aims to prevent unauthorized access, ensure accurate attendance records, and reduce manual work. By implementing this system, the school can enhance safety, increase efficiency, and promote a more secure environment for students and staff.

1.2 EXISTING SYSTEM

Existing school security systems typically rely on manual attendance tracking, CCTV surveillance. Manual systems, such as roll calls or sign-in sheets, are time-consuming, prone to human error, and lack real-time tracking of student movements. CCTV cameras monitor areas but do not provide individual student tracking or entry/exit logs. These existing methods often fail to provide comprehensive security and real-time monitoring, resulting in a need for more advanced, automated systems.

1.2.1 Disadvantages of Existing System

- ID cards can be lost, forgotten, or misused by others.
- Manual attendance is time-consuming and prone to errors.
- Unauthorized persons may enter easily without proper identity checks.
- Lack of real-time monitoring and alerts.
- Difficult to maintain accurate records over time.
- No proper backup or security in case of emergencies.

1.3 PROPOSED SYSTEM

The proposed school security system focuses on efficient entry and exit monitoring using finger print sensor integrated with Arduino Uno and NodeMCU. Each student finger biometric is taken , which is scanned upon entry or exit, logging the attendance data in real-time. This system enhances security by ensuring that only authorized individuals enter or leave the premises.

1.3.1 Advantages of proposed system

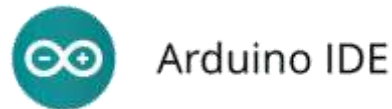
- Speeds up attendance and entry process.
- Low power consumption.
- Eliminates proxy attendance.

Chapter: 2

SYSTEM REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

2.1.1 Arduino IDE



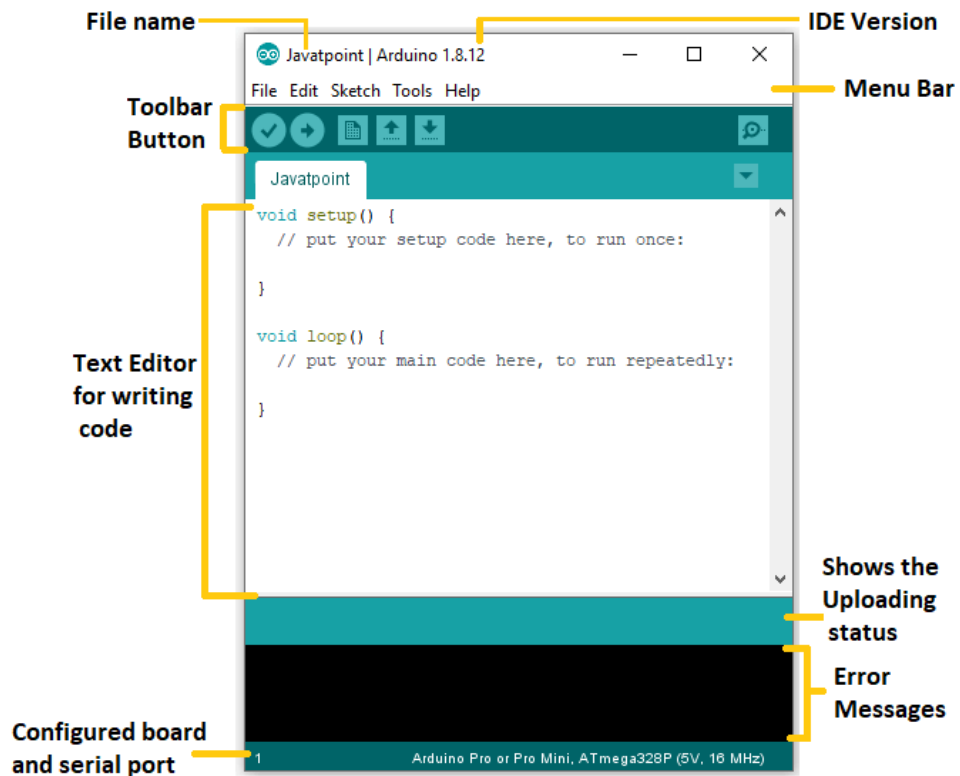
It is an official software introduced by Arduino.cc, that is mainly used for editing, compiling and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install and start compiling the code on the go.

Arduino IDE is an open-source software that is mainly used for writing and compiling the code into the Arduino Module. It is used in making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.

A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more. Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.

The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex file which is then transferred and uploaded in the controller on the board.

The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. This environment supports both C and C++ languages.



The IDE environment is mainly distributed into three sections

- Menu Bar
- Text Editor
- Output Pane

The bar appearing on the top is called Menu Bar that comes with five different options as follow

File - You can open a new window for writing the code or open an existing one. The following table shows the number of further subdivisions the file option is categorized into.

File	
New	This is used to open new text editor window to write your code
Open	Used for opening the existing written code
Open Recent	The option reserved for opening recently closed program
Sketchbook	It stores the list of codes you have written for your project
Examples	Default examples already stored in the IDE software
Close	Used for closing the main screen window of recent tab. If two tabs are open, it will ask you again as you aim to close the second tab
Save	It is used for saving the recent program
Save as	It will allow you to save the recent program in your desired folder
Page setup	Page setup is used for modifying the page with portrait and landscape options. Some default page options are already given from which you can select the page you intend to work on
Print	It is used for printing purpose and will send the command to the printer
Preferences	It is page with number of preferences you aim to setup for your text editor page
Quit	It will quit the whole software all at once

As you go to the preference section and check the compilation section, the Output Pane will show the code compilation as you click the upload button. And at the end of the compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the specific task you aim to achieve.

Edit - Used for copying and pasting the code with further modification for font

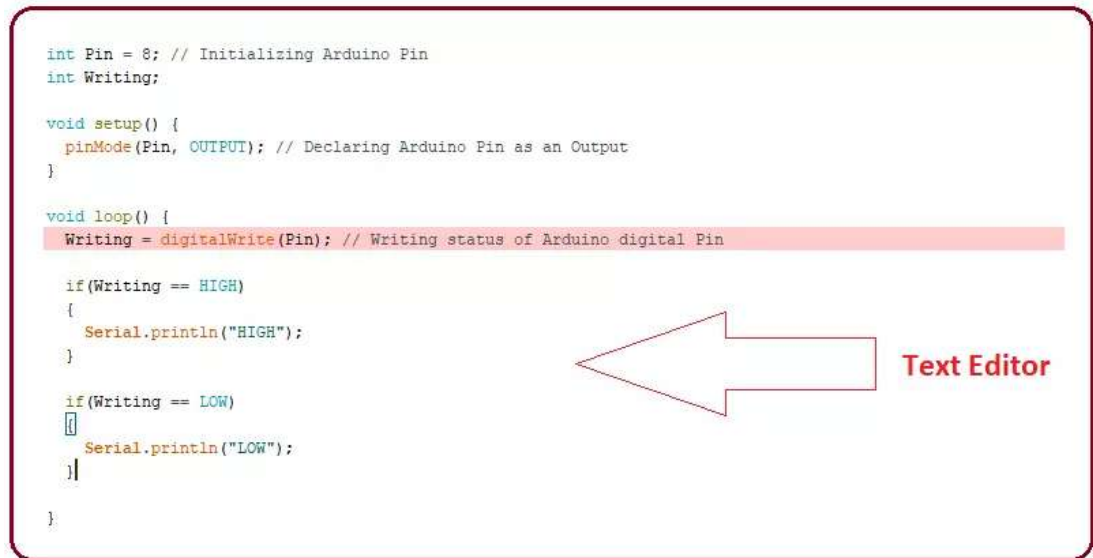
Sketch - For compiling and programming

Tools - Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.

Help - In case you are feeling skeptical about software, complete help is available from getting started to troubleshooting.

The Six Buttons appearing under the Menu tab are connected with the running program as follows.

The main screen below the Menu bard is known as a simple text editor used for writing the required code.



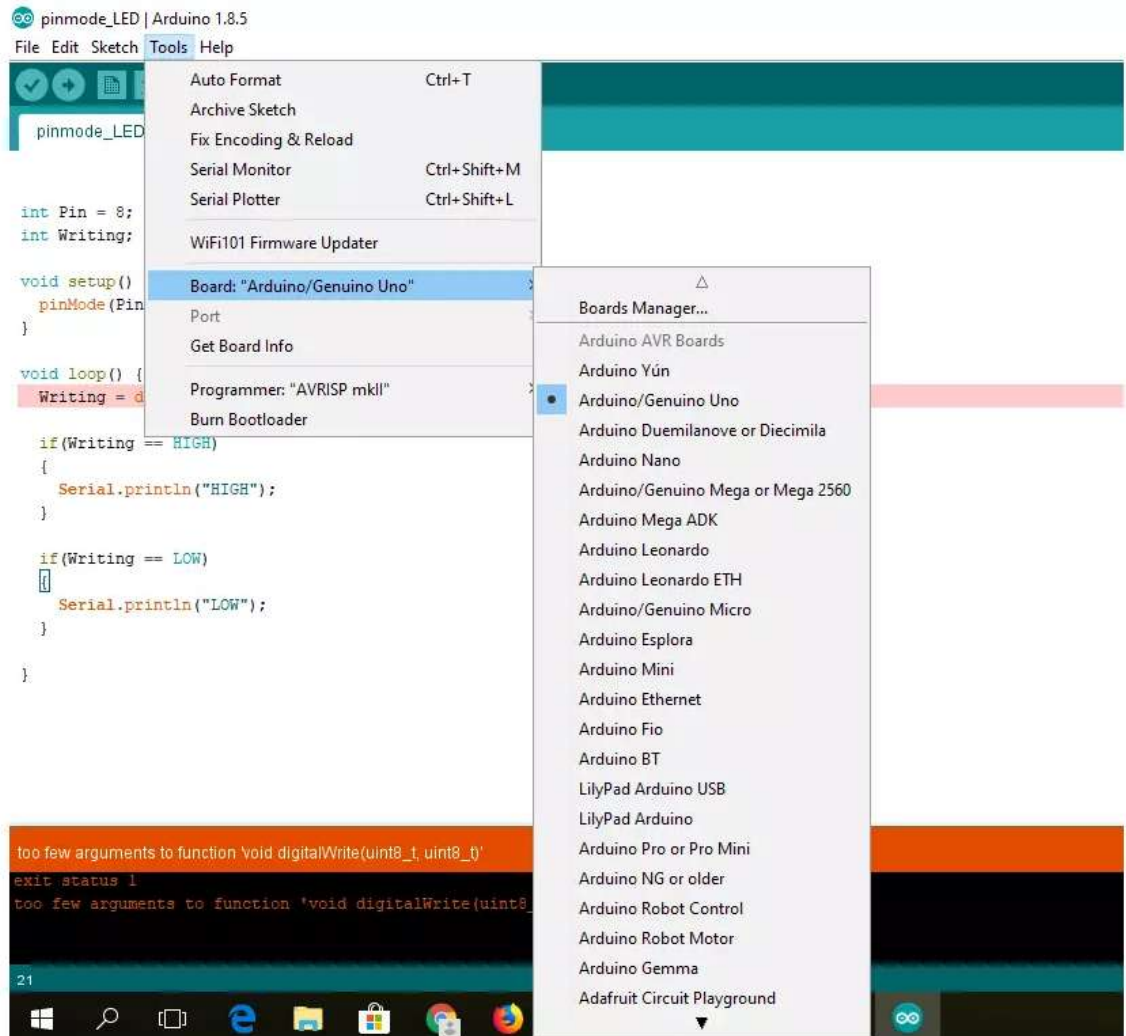
The bottom of the main screen is described as an Output Pane that mainly highlights the compilation status of the running code: the memory used by the code, and errors that occurred in the program. You need to fix those errors before you intend to upload the hex file into your Arduino Module.



More or less, Arduino C language works similar to the regular C language used for any embedded system microcontroller, however, there are some dedicated libraries used for calling and executing specific functions on the board.

Arduino Libraries:

Libraries are very useful for adding extra functionality into the Arduino Module. There is a list of libraries you can check by clicking the Sketch button in the menu bar and going to Include Library.



As you click the Include Library and Add the respective library it will be on the top of the sketch with a `#include` sign. Suppose, we Include the EEPROM library, it will appear on the text editor as `#include <EEPROM.h>`

Most of the libraries are preinstalled and come with the Arduino software. However, you can also download them from external sources.

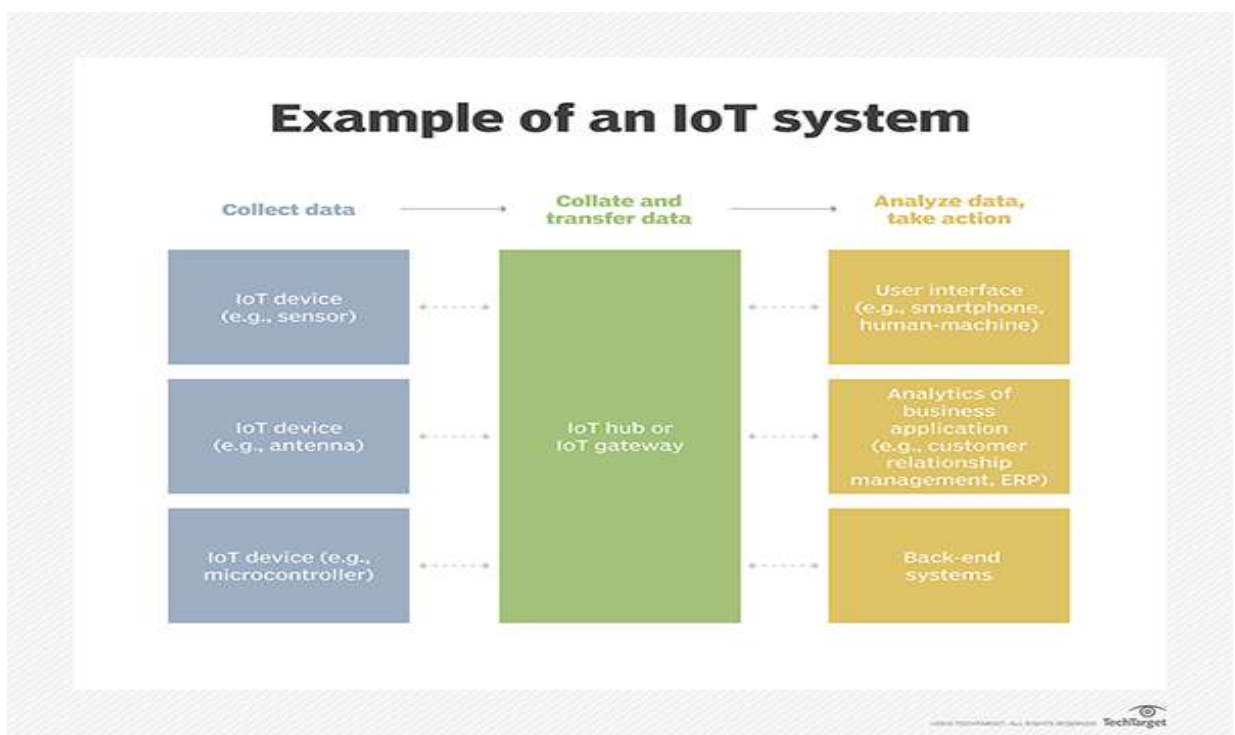
How IoT works:

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. [IoT devices](#) share the sensor data they collect by connecting to an [IoT gateway](#) or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act

on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data.

The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

IoT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic.



2.1.2 BLYNK APP:

Blynk is an Internet of Things Platform aimed to simplify building mobile and web applications for the Internet of Things. Easily connect 400+ hardware models like Arduino, ESP8266, ESP32, Raspberry Pi and similar MCUs and drag-n-drop IOT mobile apps for iOS and Android in 5 minutes

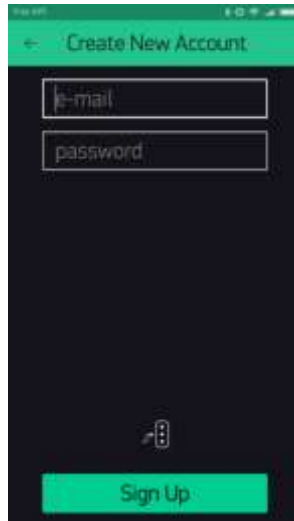


Blynk is a Platform with IOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

There are three major components in the platform:

- **Blynk App** - allows to you create amazing interfaces for your projects using various widgets we provide.
- **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your [private Blynk server](#) locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands

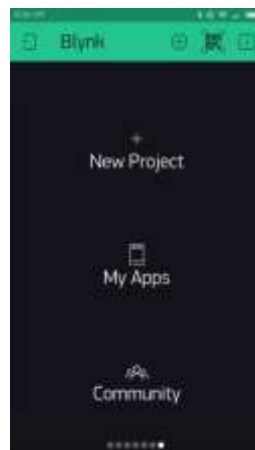


You'll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from <https://github.com/blynkkk/blynk-library/releases> , and follow along with the directions there to install the required libraries.

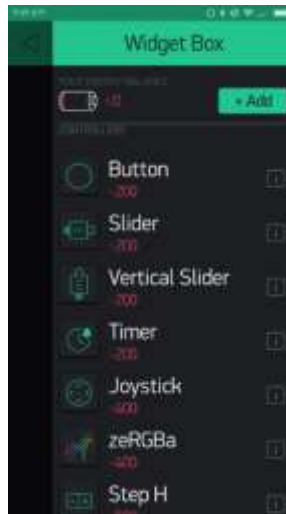
Create a Blynk Project :

Click the “Create New Project” in the app to create a new Blynk app. Give it any name. Blynk works with hundreds of hardware models and connection types. Select the Hardware type. After this, select connection type. In this project we have select WiFi connectivity

The **Auth Token** is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the “E-mail” button to send it to yourself.



Add Widgets To The Project:



Then you'll be presented with a blank new project. To open the widget box, click in the project window to open.

We are selecting a button to control Led connected with NodeMCU.

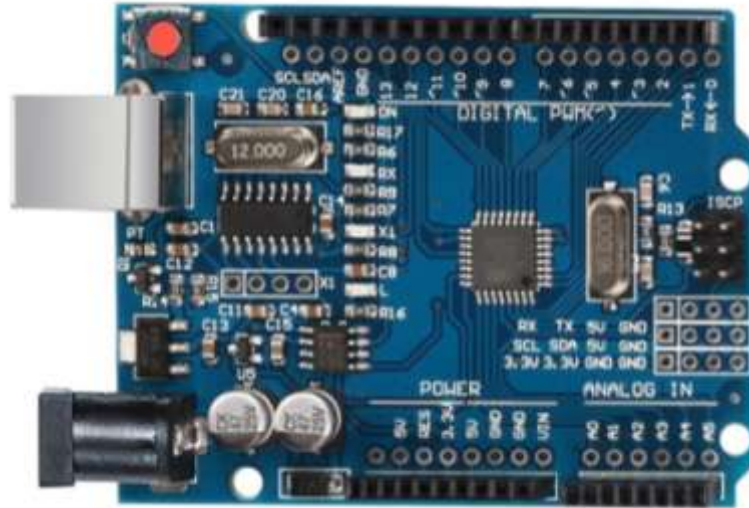
1. Click on Button.
2. Give name to Button say led.
3. Under OUTPUT tab- Click pin and select the pin to which led is connected to NodeMCU, here it is digital pin 2, hence select digital and under pin D2. And Click continue.

Under MODE tab- Select whether you want this button as “push button” or “Switch”.

You have successfully created a GUI for Arduino.

2.2 HARDWARE REQUIREMENTS

2.2.1 Arduino UNO:



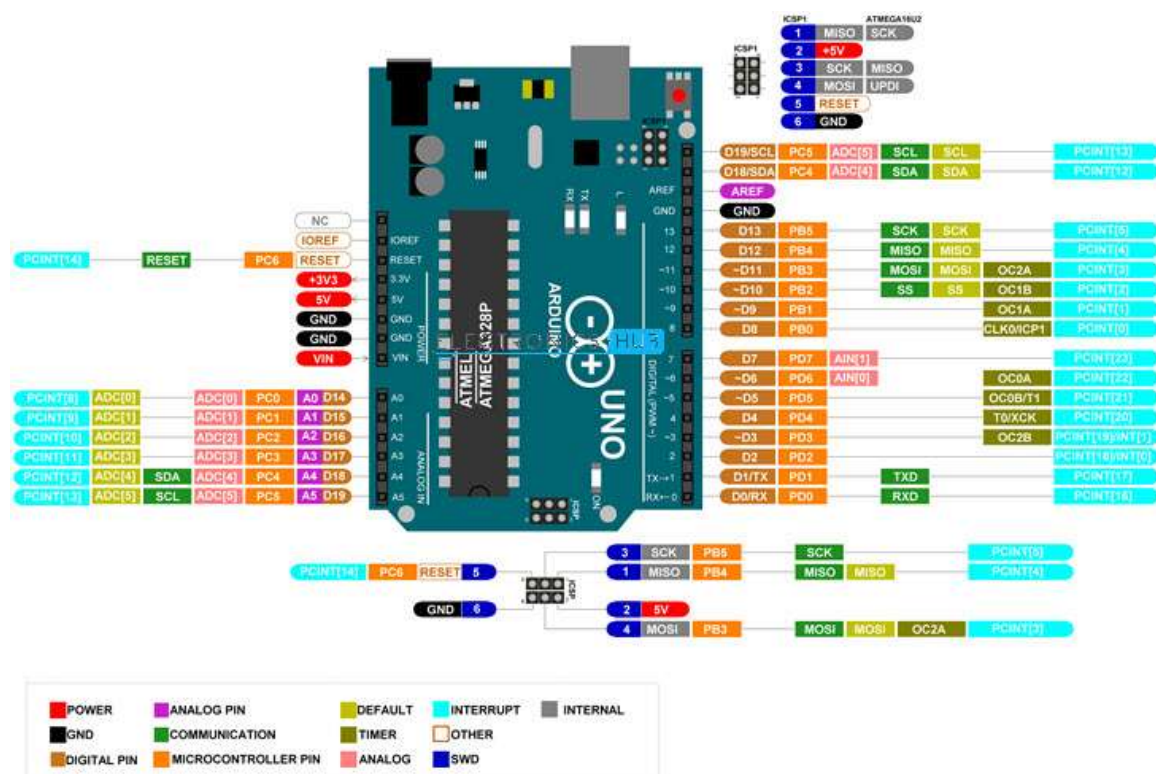
The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to 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. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards

The **ATmega328** is one kind of single-chip microcontroller formed with Atmel within the **megaAVR family**. The architecture of this Arduino Uno is a customized Harvard architecture with 8 bit **RISC processor** core. **Uno** include Arduino Pro Mini, Arduino Nano, Arduino Due, Arduino Mega, and Arduino Leonardo.

The **features of Arduino Uno ATmega328** includes the following.

- The operating voltage is 5V
- The recommended input voltage will range from 7v to 12V
- The input voltage ranges from 6v to 20V
- Digital input/output pins are 14
- Analog i/p pins are 6
- DC Current for each input/output pin is 40 mA
- DC Current for 3.3V Pin is 50 mA
- Flash Memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK Speed is 16 MHz

Pin Configuration:



Power Supply: The Arduino Uno power supply can be done with the help of a USB cable or an external power supply. The external power supplies mainly include AC to DC adapter

otherwise a battery. The adapter can be connected to the Arduino Uno by plugging into the power jack of the Arduino board. Similarly, the battery leads can be connected to the Vin pin and the GND pin of the POWER connector. The suggested voltage range will be 7 volts to 12 volts.

Vin: This is the input voltage pin of the Arduino board used to provide input supply from an external power source.

5V: This pin of the Arduino board is used as a regulated power supply voltage and it is used to give supply to the board as well as onboard components.

3.3V: This pin of the board is used to provide a supply of 3.3V which is generated from a voltage regulator on the board

GND: This pin of the board is used to ground the Arduino board.

Reset: This pin of the board is used to reset the microcontroller. It is used to Reset the microcontroller.

Analog Pins: The pins A0 to A5 are used as an analog input and it is in the range of 0-5V.

Digital Pins: The pins 0 to 13 are used as a digital input or output for the Arduino board.

Serial Pins: These pins are also known as a UART pin. It is used for communication between the Arduino board and a computer or other devices. The transmitter pin number 1 and receiver pin number 0 is used to transmit and receive the data resp.

External Interrupt Pins: This pin of the Arduino board is used to produce the External interrupt and it is done by pin numbers 2 and 3.

PWM Pins: This pins of the board is used to convert the digital signal into an analog by varying the width of the Pulse. The pin numbers 3,5,6,9,10 and 11 are used as a PWM pin.

SPI Pins: This is the Serial Peripheral Interface pin; it is used to maintain SPI communication with the help of the SPI library. SPI pins include:

- **SS:** Pin number 10 is used as a Slave Select
- **MOSI:** Pin number 11 is used as a Master Out Slave In

- **MISO:** Pin number 12 is used as a Master In Slave Out
- **SCK:** Pin number 13 is used as a Serial Clock

LED Pin: The board has an inbuilt LED using digital pin-13. The LED glows only when the digital pin becomes high.

AREF Pin: This is an analog reference pin of the Arduino board. It is used to provide a reference voltage from an external power supply.

PIN DESCRIPTION:

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	<p>Vin: Input voltage to Arduino when using an external power source.</p> <p>5V: Regulated power supply used to power microcontroller and other components on the board.</p> <p>3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.</p> <p>GND: ground pins.</p>
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

HOW TO USE ARDUINO BOARD:

The 14 digital input/output pins can be used as input or output pins by using `pinMode()`, `digitalRead()` and `digitalWrite()` functions in arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11:** These pins provide an 8-bit PWM output by using `analogWrite()` function.
- **SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK):** These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e. 1024 different values. They measure from 0 to 5 volts but this limit can be increased by using AREF pin with `analogReference()` function.

- Analog pin 4 (SDA) and pin 5 (SCA) also used for TWI communication using Wire library.

Arduino Uno has a couple of other pins as explained below:

- **AREF:** Used to provide reference voltage for analog inputs with `analogReference()` function.
- **Reset Pin:** Making this pin LOW, resets the microcontroller.

Arduino Uno to ATmega328 Pin Mapping:

When ATmega328 chip is used in place of Arduino Uno, or vice versa, the image below shows the pin mapping between the two.

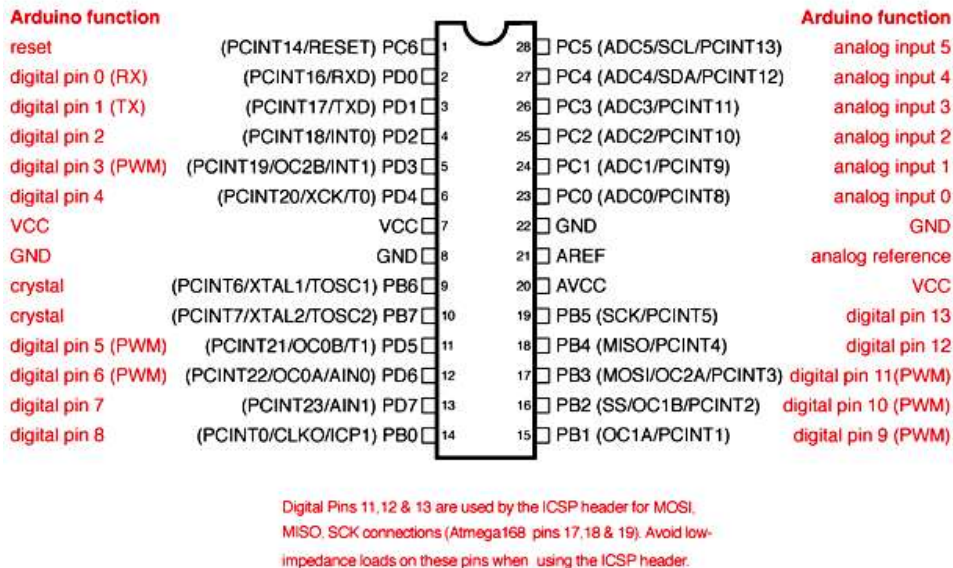


Fig 4.3 : AT mega 328 pin

Applications

- Prototyping of Electronics Products and Systems
- Multiple DIY Projects.
- Easy to use for beginner level DIYers and makers.

Projects requiring Multiple I/O interfaces and communications

2.2.3 NodeMCU:



NodeMCU is an open-source firmware and development kit that helps you to prototype or build IoT products. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The firmware uses the Lua scripting language. It is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266.

The NodeMCU_ESP8266 has 30 pins in total out of which there are 17 GPIO pins. GPIO stands for General Purpose Input Output. There are the 9 digital pins ranging from D0-D8 and there is only one analog pin A0, which is a 10-bit ADC. The board has a 2.4 GHz antenna for a long-range of network and the CP2102 is the USB to TTL converter.

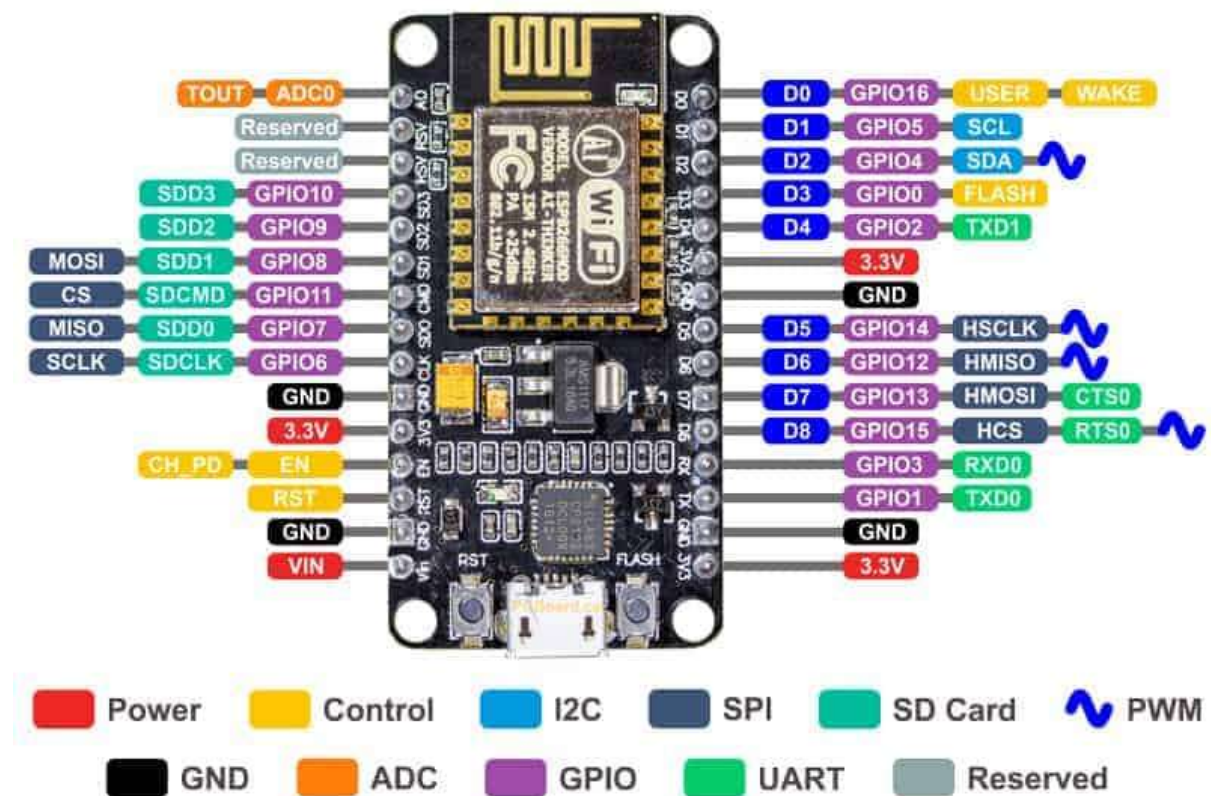
There's also 128 KB RAM and 4MB of Flash memory. The ESP8266 Integrates 802.11b/g/n HT40 Wi-Fi transceiver, so it can not only connect to a Wi-Fi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8266 NodeMCU even more versatile.

ESP8266 has many applications when it comes to the IoT. Here are just some of the functions the chip is used for:

- **Networking:** The module's Wi-Fi antenna enables embedded devices to connect to routers and transmit data
- **Data Processing:** Includes processing basic inputs from analog and digital sensors for far more complex calculations with an RTOS or Non-OS SDK

- **P2P Connectivity:** Create direct communication between ESPs and other devices using IoT P2P connectivity
- **Web Server:** Access pages written in HTML or development languages.

Pinout configuration:



- ❖ **Power Pins** There are four power pins. **VIN** pin and three **3.3V** pins.
 - **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the **VIN** pin
 - **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.

- ❖ **GND** are the ground pins of NodeMCU/ESP8266

- ❖ **I2C Pins** are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

- ❖ **GPIO Pins** NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

- ❖ **ADC Channel** The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.


- ❖ **UART Pins** NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

- ❖ **SPI Pins** NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
 - 4 timing modes of the SPI format transfer
 - Up to 80 MHz and the divided clocks of 80 MHz
 - Up to 64-Byte FIFO

- ❖ **SDIO Pins** NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

- ❖ **PWM Pins** The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μ s to 10000 μ s (100 Hz and 1 kHz).

- ❖ **Control Pins** are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
 - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
 - **RST:** RST pin is used to reset the ESP8266 chip.
 - **WAKE:** Wake pin is used to wake the chip from deep-sleep.

- ❖  Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
 - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
 - **RST:** RST pin is used to reset the ESP8266 chip.
 - **WAKE:** Wake pin is used to wake the chip from deep-sleep.

USB to Serial Converter – CP2102 or CH340G

Incorporated into each NodeMCU is a USB to Serial Converter. The official design is based on the CP2102 chipset and offers the best compatibility. Genuine boards use the CP2102 chipset including the officially licensed Amica NodeMCU modules. The other common USB to Serial Converter used is the CH340G which is common on the lower-priced modules including the LoLin units. Other designs may use drivers including the FTDI chipset, but those designs are rare.

Depending on the Operating System you are using with the NodeMCU, the appropriate driver must be installed. Generally, Windows 10 immediately recognizes the CP2102 chipset while the CH340G may require separate installation.



- Drivers for the CP2102 are available for **download** from the [Silicon Labs support site](#). Drivers constantly evolve and ensuring and installing the most recent version in your development environment minimum issues. Drivers are available for Windows, Mac, Linux, and Android. *We also have a local copy of the CP2102 drivers (v10.1.8) available locally for [download](#). You are always best to visit the original manufacturer to ensure you are receiving the most recent versions of the driver.*



- WCH maintain and update the drivers for the CH340G on a regular basis. Versions of the driver are also available for Windows, Mac, Linux, and Android. Visit their [Driver Download](#) page. We also have a local copy of the CH340G drivers (version 3.5) available locally for [download](#). *You are always best to visit the original manufacturer to ensure you are receiving the most recent versions of the driver.*

We have experienced situations where both CP2102 and CH340G devices have not functioned or been recognized as expected. The solution was as simple as uninstalling the old driver and installing the most recent version.

FEATURES	NODEMCU
Microcontroller	ESP-8266 32-bit
NodeMCU Model	Amica
NodeMCU Size	49mm x 26mm
Carrier Board Size	n/a
Pin Spacing	0.9" (22.86mm)
Clock Speed	80 MHz
USB to Serial	CP2102
USB Connector	Micro USB
Operating Voltage	3.3V
Input Voltage	4.5V-10V
Flash Memory/SRAM	4 MB / 64 KB
Digital I/O Pins	11
Analog In Pins	1
ADC Range	0-3.3V
UART/SPI/I2C	1 / 1 / 1
WiFi Built-In	802.11 b/g/n
Temperature Range	-40C - 125C

2.2.4 FINGER PRINT SENSOR:

The basic function of every type of scanner is **to obtain an image of a person's fingerprint and find a match for it in its database**. The measure of the fingerprint image quality is in dots per inch (DPI). Optical scanners take a visual image of the fingerprint using a digital camera.

Fingerprints' **capacitive touch sensors** are based on patented proprietary technology, which offers several advantages such as high image quality, and 256 gray-scale values from every single pixel element. The sensors contain small capacitive plates, each with their own electrical circuit embedded in the chip.

Although every fingerprint is different, they're all variations on three broad categories: **the arch, which looks a bit like a cross-section of a hill; the loop, which is teardrop-shaped; and the whorl, which is reminiscent of a whirlpool.**



- 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.
- This is the R307 Optical Fingerprint Reader Sensor Module. R307 fingerprint module is a fingerprint sensor with a TTL UART interface for direct connections to microcontroller UART or to PC through MAX232 / USB-Serial adapter.
- Operating voltage (v) - 4.2 to 6 VDC. Current consumption - $\leq 75\text{mA}$. Verification Speed - 0.2 sec.
- Scanning Speed - 0.3 sec. Character file size - 256 bytes. Template size - 512 bytes.

2.2.5 16x2 I2C LCD DISPLAY:

A **liquid-crystal display (LCD)** is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made from a matrix of small pixels, while other displays have larger elements. LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement. For example, a character positive LCD with a backlight will have black lettering on a background that is the color of the backlight, and a character negative LCD will have a black background with the letters being of the same color as the backlight. Optical filters are added to white on blue LCDs to give them their characteristic appearance.

LCDs are used in a wide range of applications, including LCD televisions, computer monitors, instrument panels, aircraft cockpit displays, and indoor and outdoor signage. Small LCD screens are common in portable consumer devices such as digital cameras, watches, calculators, and mobile telephones, including smartphones. LCD screens are also used on consumer electronics products such as DVD players, video game devices and clocks. LCD screens have replaced heavy, bulky cathode ray tube (CRT) displays in nearly all applications. LCD screens are available in a wider range of screen sizes than CRT and plasma displays, with LCD screens available in sizes ranging from tiny digital watches to very large television receivers.

LCDs are slowly being replaced by OLEDs, which can be easily made into different shapes, and have a lower response time, wider color gamut, virtually infinite color contrast and viewing angles, lower weight for a given display size and a slimmer profile (because OLEDs use a single glass or plastic panel whereas LCDs use two glass panels; the thickness of the panels increases with size but the increase is more noticeable on LCDs) and potentially lower power consumption (as the display is only "on" where needed and there is no backlight). OLEDs, however, are more expensive for a given display size due to the very expensive electroluminescent materials or phosphors that they use. Also due to the use of phosphors, OLEDs suffer from screen burn-in and there is currently no way to recycle OLED displays,

whereas LCD panels can be recycled, although the technology required recycling LCDs is not yet widespread. Attempts to increase the lifespan of LCDs are quantum dot displays, which offer similar performance to an OLED display, but the quantum dot sheet that gives these displays their characteristics cannot yet be recycled.

Since LCD screens do not use phosphors, they rarely suffer image burn-in when a static image is displayed on a screen for a long time, e.g., the table frame for an airline flight schedule on an indoor sign. LCDs are, however, susceptible to image persistence. The LCD screen is more energy-efficient and can be disposed of more safely than a CRT can. Its low electrical power consumption enables it to be used in battery-powered electronic equipment more efficiently than a CRT.

A liquid-crystal display (LCD) is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. Here, in this we're going to use a monochromatic 16x2 alphanumeric LCD. 16x2 means that 20 characters can be displayed in each of the 4 rows of the 16x2 LCD, thus a total of 80 characters can be displayed at any instance of time. LCD accepts two types of signals, one is data, and another is control. These signals are recognized by the LCD module from status of the RS pin. Now data can be read also from the LCD display, by pulling the R/W pin high. As soon as the E pin is pulsed, LCD display reads data at the falling edge of the pulse and executes it, same for the case of transmission.

This is a basic 20 character by 4 line display. Utilizes the extremely common HD44780 parallel interface chipset. Interface code is freely available. You will need ~11 general I/O pins to interface to this LCD screen. Includes LED backlight. 20 characters wide, 4 rows character LCD module, SPLC780C controller, 6800 4/8-bit parallel interface, single led backlight with yellow green colour included can be dimmed easily with a resistor or PWM, STN-LCD positive, dark blue text on the yellow green colour, wide operating temperature range, ROHS compliant, built in character set supports English/Japanese text.

The full character set. Optional 3.3v or 5v power supply and optional pin header connection it's easily controlled by MCU such as 8051, PIC, AVR, ARDUINO, ARM and Raspberry PI. IT can be used in any embedded systems, industrial device, and security, medical and hand-held equipment.

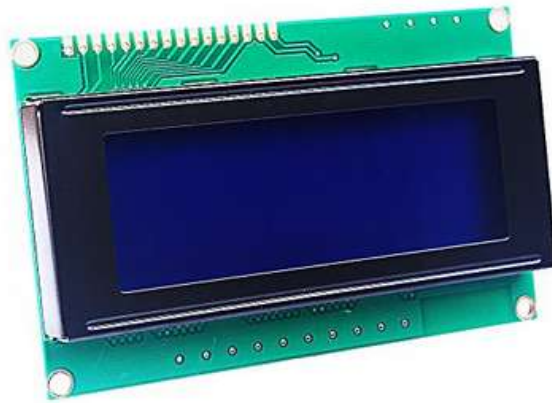
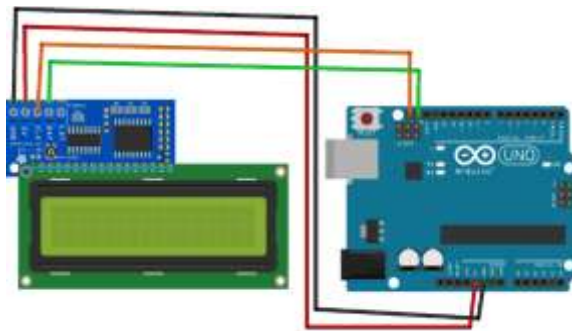


Fig 2.2.5: 16x2 LCD display

Interfacing of 16x2 I2C LCD with Arduino Mega:



2.2.6. SERVO MOTOR:

A servomotor (or servo motor) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.



Servomotors are generally used as a high-performance alternative to the stepper motor. Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate, but for this the controller needs to 'know' the position of the stepper motor on power up. Therefore, on first power up, the controller will have to activate the stepper motor and turn it to a known position, e.g. until it activates an end limit switch. This can be observed when switching on an inkjet printer; the controller will move the ink jet carrier to the extreme left and right to establish the end positions. A servomotor will immediately turn to whatever angle the controller instructs it to, regardless of the initial position at power up.

The lack of feedback of a stepper motor limits its performance, as the stepper motor can only drive a load that is well within its capacity, otherwise missed steps under load may lead to positioning errors and the system may have to be restarted or recalibrated. The encoder and controller of a servomotor are an additional cost, but they optimise the performance of the overall system (for all of speed, power and accuracy) relative to the capacity of the basic motor. With larger systems, where a powerful motor represents an increasing proportion of the system cost, servomotors have the advantage.

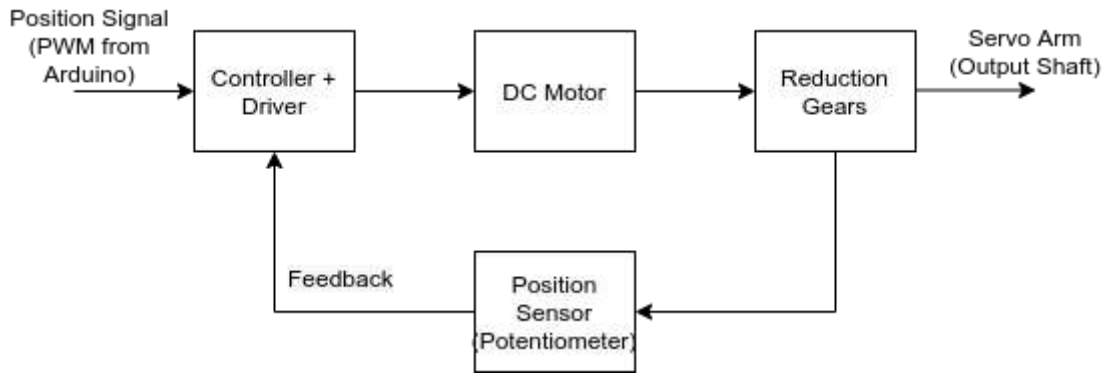
There has been increasing popularity in closed loop stepper motors in recent years.^[citation needed] They act like servomotors but have some differences in their software control to get smooth motion. The main benefit of a closed loop stepper motor is its relatively low cost. There is also no need to tune the PID controller on a closed loop stepper system.^[5]

Many applications, such as laser cutting machines, may be offered in two ranges, the low-priced range using stepper motors

Servos

- Standard servos are used to maintain a certain angle, not meant for full rotation
- Uses closed-loop feedback control
- Servos are used in applications requiring high torque, accurate rotation within a limited angle such as Robotic arms, valve control, rudder control etc.





SG90 Micro Servo:



- SG90 is a servo motor which operates based on PWM control signals
- The servo maintains a certain angle (position) based on the width of the pulse fed in through a signal input
- Some technical specifications
- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- PWM frequency = 50Hz
- Pin configuration: Yellow / Light Orange / White (Signal), Red / Dark Orange (+5V), Brown/Black (Ground)

Note : The pulse width in the image does not correspond to SG90, for illustration of the concept only.

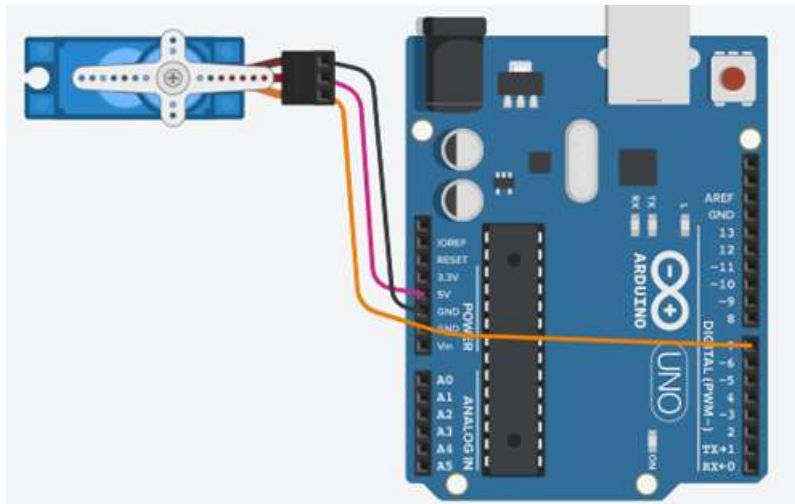
Image courtesy: Stefan Tauner

544 – 1500 – 2400 us

0° – 90° – 180°

Servo Library

- This library allows an Arduino board to control servo motors
- Standard servos allow the shaft to be positioned at various angles, usually between 0° and 180°
- Any digital pin on UNO can be used, not necessarily those supporting PWM. However, note that using Servo library disables `analogWrite()` functionality on pins 9 and 10
- `attach(int)` - attach a servo to an I/O pin, e.g., `servo.attach(pin)`, `servo.attach(pin, min, max)`
 - `servo`: a variable of type `Servo`, `pin`: pin number, default values: `min` = 544 us, `max` = 2400 us
- `write(int)` - write a value to the servo to control its shaft accordingly
- `detach()` - stop an attached Servo from pulsing its I/O pin



Caution : Do not overload the servo. The servo and your battery / power source could be damaged if servo is overloaded.

Do not power the servo from a 9V battery. Most servos can't take > 6V.

Continuous Rotation Servos

- Continuous rotation servos are standard servos modified to perform *open loop speed control* (instead of *closed loop position control*)
- Rotation speed and direction are controlled through PWM signals (pulse width) for continuous rotation servos, just like how the position is controlled for standard servos
- Effectively, continuous servos are DC motors with integrated motor drivers and reduction gears in a compact, inexpensive package, rather than true 'servo' motors
- Continuous rotation servos allow the rotation of the shaft to be set to various speeds
 - Electrical connections are identical to that of a standard servo.
 - The original servo library can be used; e.g., `servo.write(angle)`, `angle` = 0 to 180 → 0: full speed in one direction, 180: full speed in the other, and around 90: no movement
- FS90R (also known as SG90 continuous / 360° full rotation) operating speed: 110RPM (4.8V); 130RPM (6V)

BUZZER

A buzzer typically produces a continuous, loud, and often unpleasant sound. It is commonly used as an alarm or warning signal in industrial settings or emergencies. In general, the terms "buzzer" and "beeper" are often used interchangeably, and their specific meanings can vary depending on the context in which they are used.

The basic structure of a buzzer consists of a coil of wire, a magnet, and a diaphragm or plate. When an electric current is passed through the coil, it creates a magnetic field that causes the diaphragm to vibrate, producing sound waves that create the buzzing or alarm sound.

Buzzer devices are available in a variety of types and sizes, including piezoelectric buzzers, magnetic buzzers, and electromechanical buzzers. Piezoelectric buzzers are commonly used in small electronic devices and produce a high-pitched sound. Magnetic buzzers are larger and produce a lower-pitched sound. Electromechanical buzzers are similar to magnetic buzzers but also include a mechanical component that creates an additional vibration and louder sound.

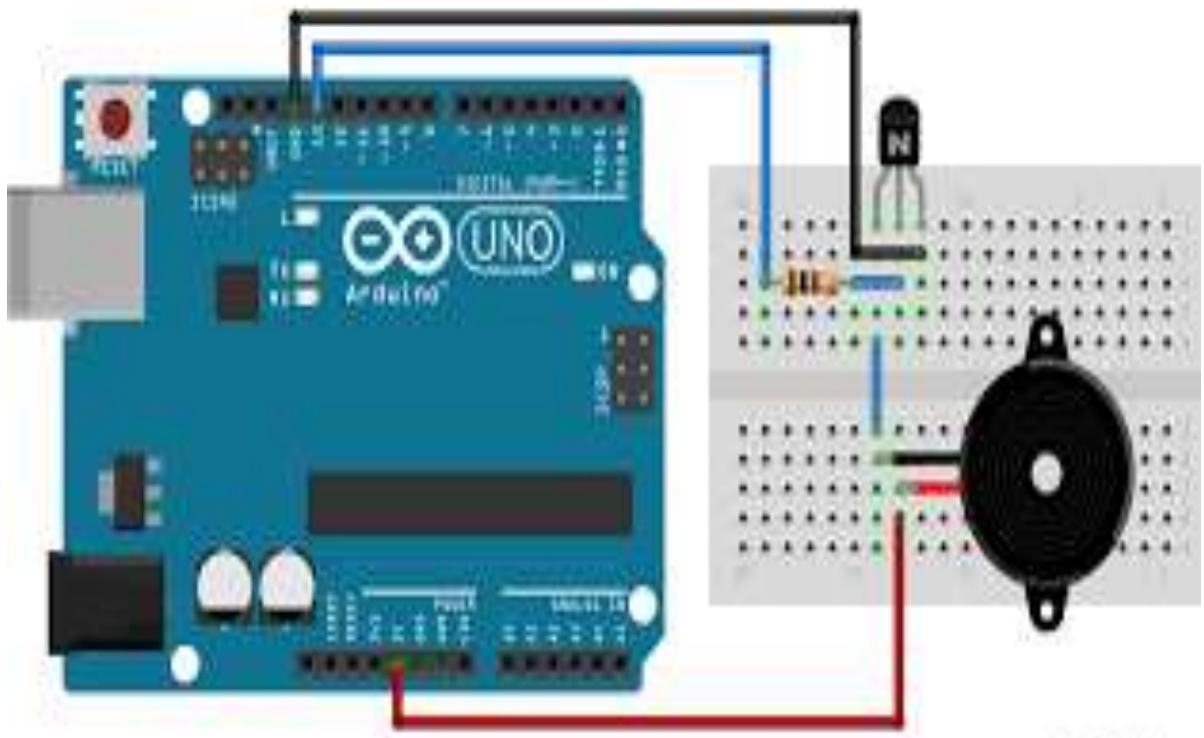
Buzzer devices can be found in various applications, such as security systems, smoke alarms, industrial machinery, and emergency warning systems. They are also used in games and toys to create sound effects, such as in quiz games, timers, and electronic musical instruments.

The basic structure of a buzzer consists of a coil of wire, a metal diaphragm, and a magnet. When an electrical current flows through the coil, it creates a magnetic field which pulls the metal diaphragm towards the magnet. This movement causes the diaphragm to vibrate, producing a sound wave that creates the buzzing sound.

Buzzer types can vary based on the application, such as electromagnetic buzzers, piezoelectric buzzers, and mechanical buzzers. Electromagnetic buzzers are typically used in alarm systems, doorbells, and timers, while piezoelectric buzzers are used in electronic devices such as watches, calculators, and mobile phones. Mechanical buzzers are more commonly used in older technologies, such as mechanical clocks and traditional doorbells.

Buzzer sound can be characterized by its frequency and loudness. The frequency of a buzzer determines the pitch of the sound it produces, with higher frequencies producing higher-pitched sounds. The loudness of a buzzer is determined by its amplitude, which is the strength of the sound wave produced. Buzzer sounds can also be modulated to create different patterns or rhythms.

Overall, buzzers are simple yet essential components in various electronic devices, providing a distinct and attention-grabbing sound to alert users of important information or events.



fritzing



Pin1 - VCC
Pin2 - GND

Chapter: 3 SYSTEM ANALYSIS

3.1 ARCHITECTURE

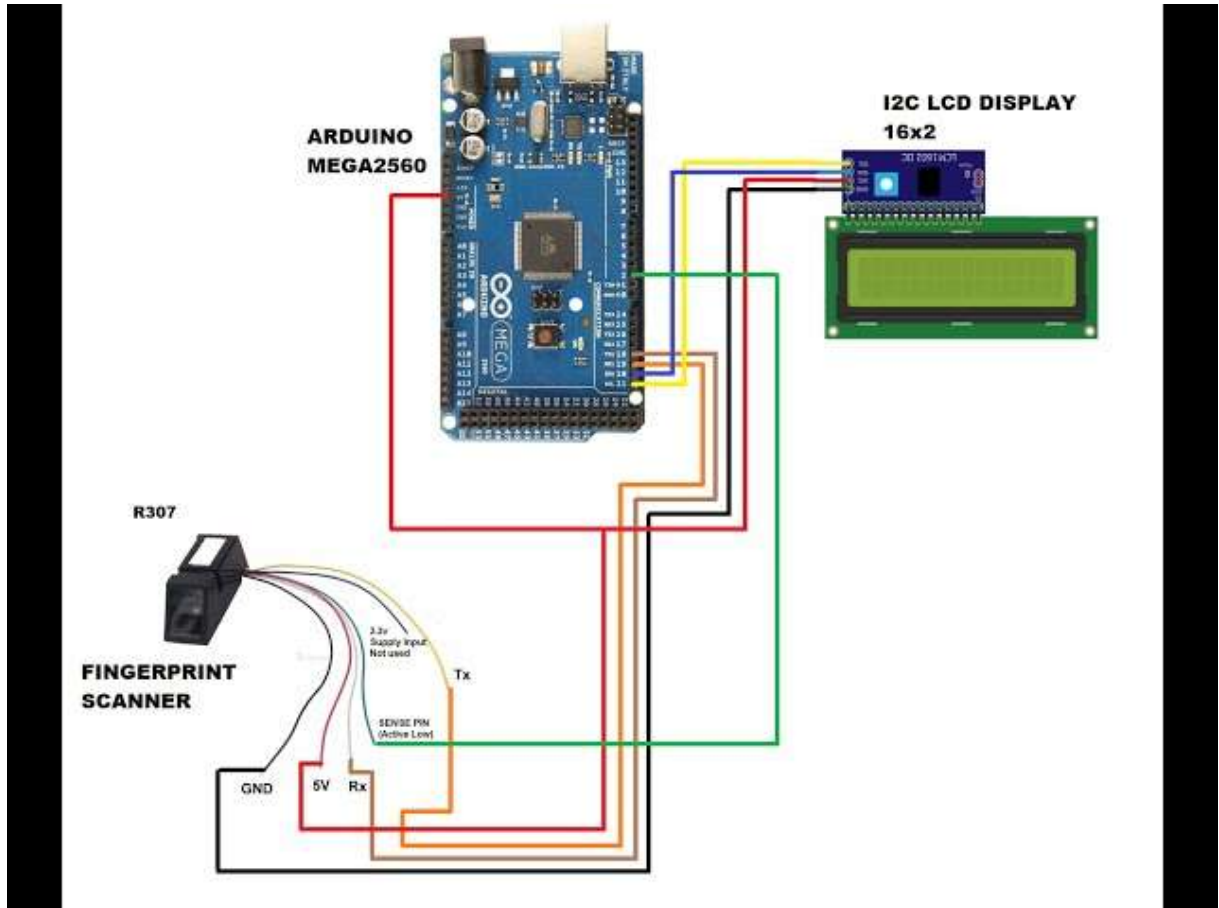


Fig 3.1: System Architecture

Develop a comprehensive design that outlines how each hardware component will interact. The fingerprint sensor will authenticate individuals, the Arduino UNO will process this data, the servo motor will manage door operations, the LCD will display status messages, and the Wi-Fi module will facilitate data transmission. Creating detailed circuit diagrams and flowcharts aids in visualizing the system's operation and guides the implementation process.

3.2 METHEDOLOGY

3.2.1 Principle of Operation

A fingerprint sensor captures the unique patterns of ridges and valleys on an individual's fingertip. This process involves scanning the finger to obtain a clear and detailed image.

The captured fingerprint image is processed to identify distinctive features, such as loops, whorls, and arches, known as Level 1 features. Advanced algorithms may further analyze minutiae points (Level 2 features) and pore patterns (Level 3 features) to create a comprehensive digital representation of the fingerprint.

The extracted features are converted into a digital template—a mathematical representation of the fingerprint. This template is securely stored in the system's database for future reference. Importantly, the original fingerprint image is not retained, ensuring user privacy.

During authentication, the system captures a new fingerprint scan and extracts its features to create a template. This new template is compared against the stored templates in the database. If a match is found, the individual's identity is verified. This matching process is typically swift, often completing in less than a second.

Upon successful verification, the system triggers mechanisms such as unlocking doors or recording attendance. If the fingerprint does not match any stored templates, access is denied, and the attempt may be logged for security purposes.

3.2.2 System work flow

Implementing a fingerprint-based school security system involves a systematic workflow to ensure secure and efficient access control. The process begins with Fingerprint Enrollment, where authorized individuals, such as students and staff, place their fingers on the fingerprint sensor. The sensor captures the unique patterns of ridges and valleys on the fingertip, processes this data to create a digital template, and securely stores it in the system's database, linking it to the individual's identity.

During the Authentication Process, when an individual seeks access, they place their finger on the sensor. The system captures the fingerprint, converts it into a digital template, and compares it against the stored templates in the database. If a match is found, the system proceeds to the next step; otherwise, access is denied.

In the Access Control Execution phase, upon successful authentication, the system sends a signal to the servo motor to unlock the door, granting access. If authentication fails, the door remains locked, ensuring unauthorized individuals cannot enter.

The system provides immediate feedback through an LCD Display, showing messages like "Access Granted" or "Access Denied," informing users of their authentication status. For Data Logging and Monitoring, each access attempt, whether successful or not, is recorded with details such as timestamp and user ID. The Wi-Fi module transmits these logs to a remote server or cloud platform, enabling administrators to monitor access patterns and identify any security concerns in real-time.

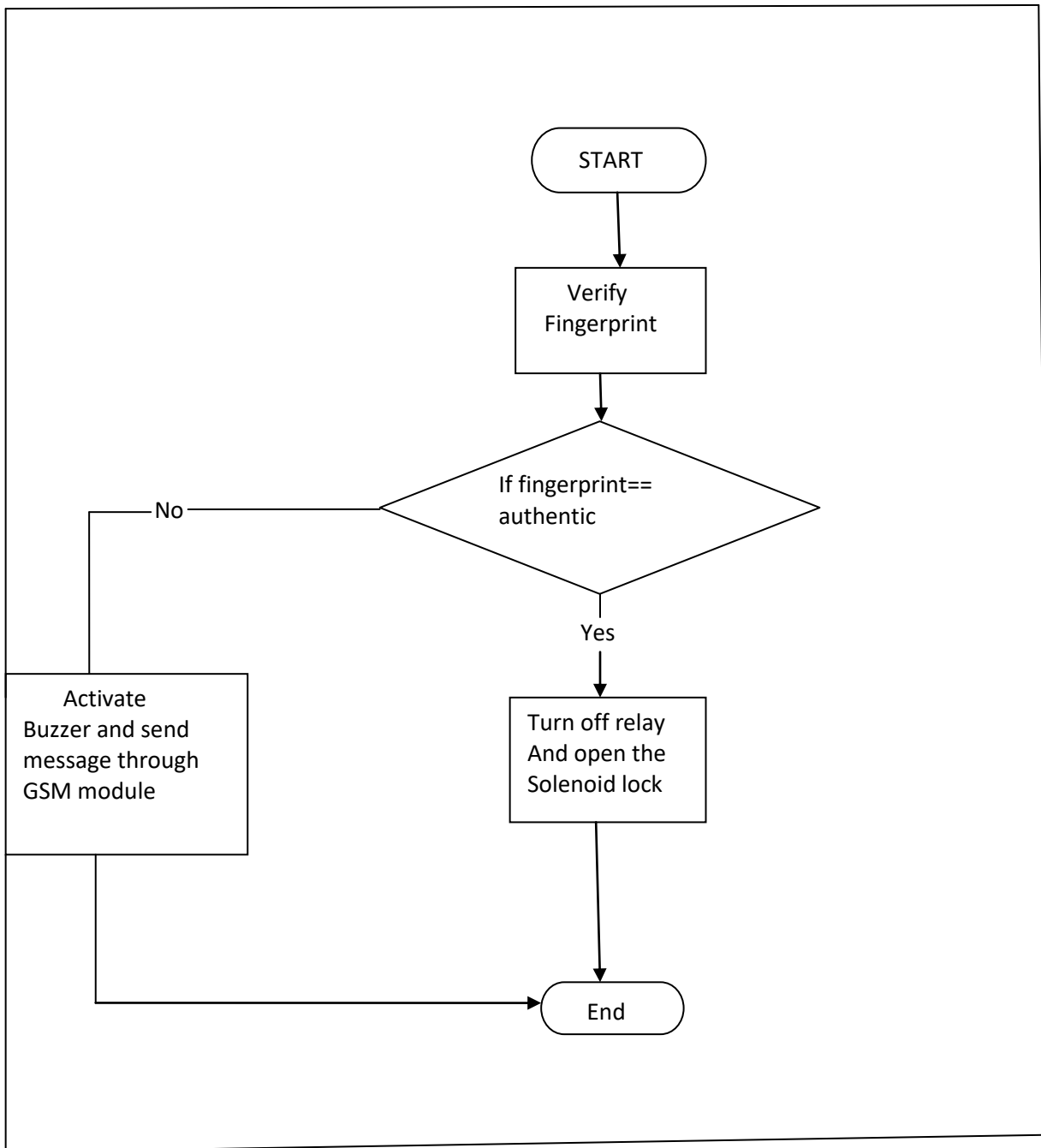


Fig3.2.2. System Work Flow

3.3 HARDWARE CONFIGURATION

3.3.1 Components and working

S No.	Components	Specification	Working
1	Arduino Uno	-	Main unit core component
2	NodeMCU	-	Wifi module, server communication
3	FingerPrint Sensor	-	Captures and processes fingerprint images for authentication.
4	ServoMotor	Dc servomotor	Controls door locking.
5	LCD Display	16x2 display	Provides real-time system messages to user
6	Wi-Fi Module	ESP8266	Enables wireless communication for remote monitoring.
7	Buzzer	Standard piezoelectric	Emits audible alert for unauthorized access attempts or system errors.

Table 3.3.1: Components & working

3.3.2 Construction

Constructing a fingerprint-based school security system involves integrating various hardware components to create a cohesive and functional setup. The primary components include a fingerprint sensor, microcontroller (such as Arduino UNO), servo motor, LCD display, Wi-Fi module, and optional peripherals like a buzzer and keypad. Each component plays a vital role in ensuring the system's efficiency and reliability.

The construction process begins with establishing a stable power supply to ensure consistent operation of all components. The microcontroller serves as the central hub, managing data flow and controlling peripheral devices. The fingerprint sensor captures and processes fingerprint images for authentication, while the servo motor controls the door's locking and unlocking mechanism based on verification results. The LCD display provides real-time feedback to users, displaying messages such as "Access Granted" or "Access

Denied." The Wi-Fi module enables wireless communication, allowing for remote monitoring and data logging. Optional components like a buzzer can emit audible alerts for unauthorized access attempts, and a keypad can facilitate administrative functions such as enrolling new fingerprints.

During assembly, it's crucial to connect each component to the microcontroller correctly, ensuring proper wiring for power and data communication. Secure all components within a protective enclosure to prevent tampering and environmental damage. Mount the fingerprint sensor and LCD display at accessible locations for users. After assembly, upload the control program to the microcontroller, incorporating libraries for the fingerprint sensor, LCD display, and Wi-Fi module. Conduct thorough testing of each component individually and as an integrated system to verify seamless operation.

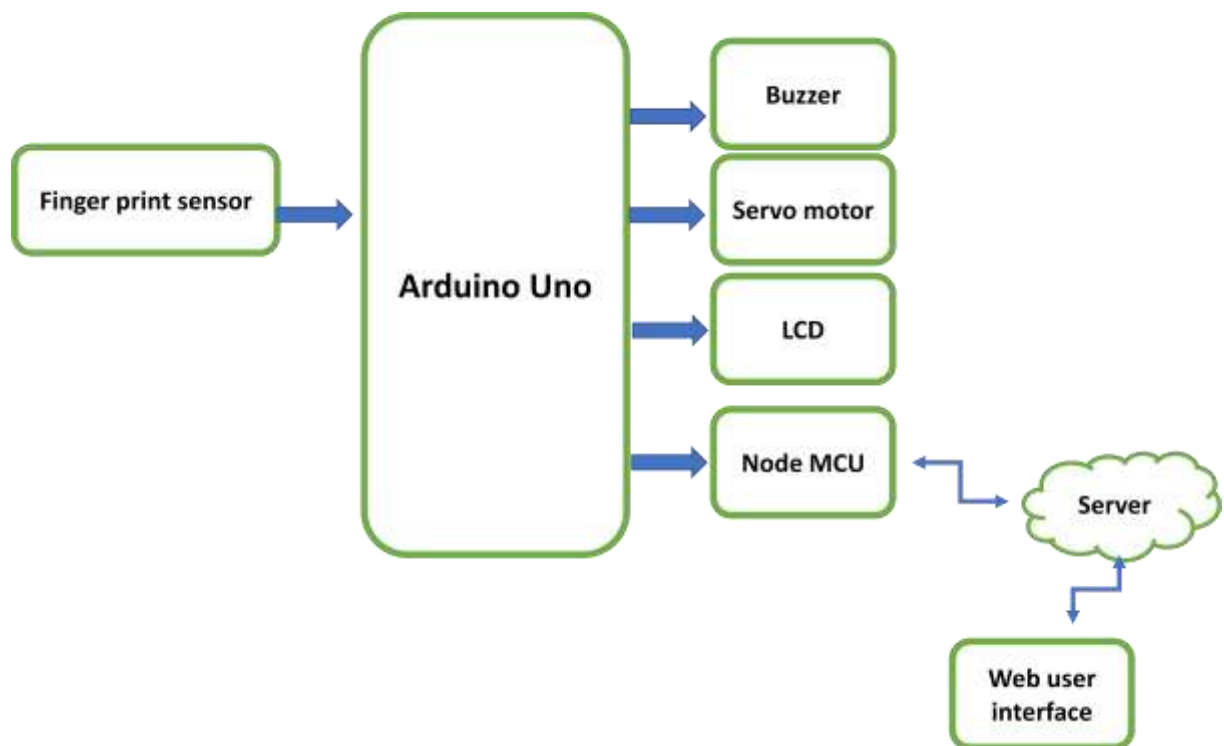


Fig 3.3.2: block diagram

3.3.2 Pinout configuration

Components	Arduino Pin
FingerPrint Sensor	
VCC	5v
GND	GND
TX(Transmit)	Digital Pin 2
RX(Receiver)	Digital Pin 3
LCD Display(I2C)	
VCC	5V
GND	GND
SDA(Data Line)	A4
SCL(Clock Line)	A5
Servo Motor	
Control signal	Digital Pin 7
Power	5V
Ground	GND
NodeMCU	
6	Rx
5	Tx
LED Indicators	
LED 1	Digital Pin 8
LED 2	Digital Pin 9
LED 3	Digital Pin 10
LED 4	Digital Pin 11

Table 3.3.3(a): Main unit – Pinout config

Chapter: 4

SYSTEM DESIGN

4.1 INTRODUCTION

A fingerprint-based school security system is designed to enhance security, automate attendance tracking, and regulate access control within school premises. The system integrates biometric authentication, database management, and real-time monitoring to ensure only authorized individuals can enter restricted areas.

The system consists of hardware components such as fingerprint scanners, microcontrollers, and servers, along with software modules for authentication, data processing, and reporting. It verifies student and staff identities by matching their fingerprints with stored records, thereby preventing unauthorized access and ensuring efficient attendance management.

4.2 UML DIAGRAMS

The Unified Modelling Language (UML) can be a customary visual modelling language purported to be used for modelling business and similar processes, analysis, design, and implementation of software-based systems.

UML is a typical language for business analysts, software system package architects and developers accustomed describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software system package systems.

UML can be described as a general-purpose visual modeling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design.

There square measure a pair of broad categories of diagrams thus square measure all over again divided into sub-categories:

- Structural Diagrams
- Behavioral Diagrams

Structural Diagrams:

The structural diagrams represent the static facet of the system. These static aspects represent those components of a diagram that forms the most structure and so stable. These static components are represented by categories, interfaces, objects, parts and nodes. The four structural diagrams are:

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Behavioral Diagrams:

Any system will have 2 aspects, static and dynamic. Therefore, a model is taken into account as complete once each the aspects area unit lined totally. Behavioral diagrams essentially capture the dynamic side of a system. Dynamic side are often any delineated because the changing/moving components of a system. UML has the subsequent 5 forms of activity diagrams:

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

4.2.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case

diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Another key aspect of use case diagrams is their versatility in capturing various scenarios and user roles within a system. Each use case represents a specific functionality or task that the system must perform to satisfy a user's goal. Actors, on the other hand, represent the roles played by users or external systems interacting with the software. This flexibility allows use case diagrams to capture a wide range of user-system interactions, from basic actions like logging in to complex processes involving multiple actors and use cases. Moreover, use case diagrams facilitate the identification of both primary and alternative flows of actions, enabling developers to anticipate different user paths and design the system to accommodate them effectively.

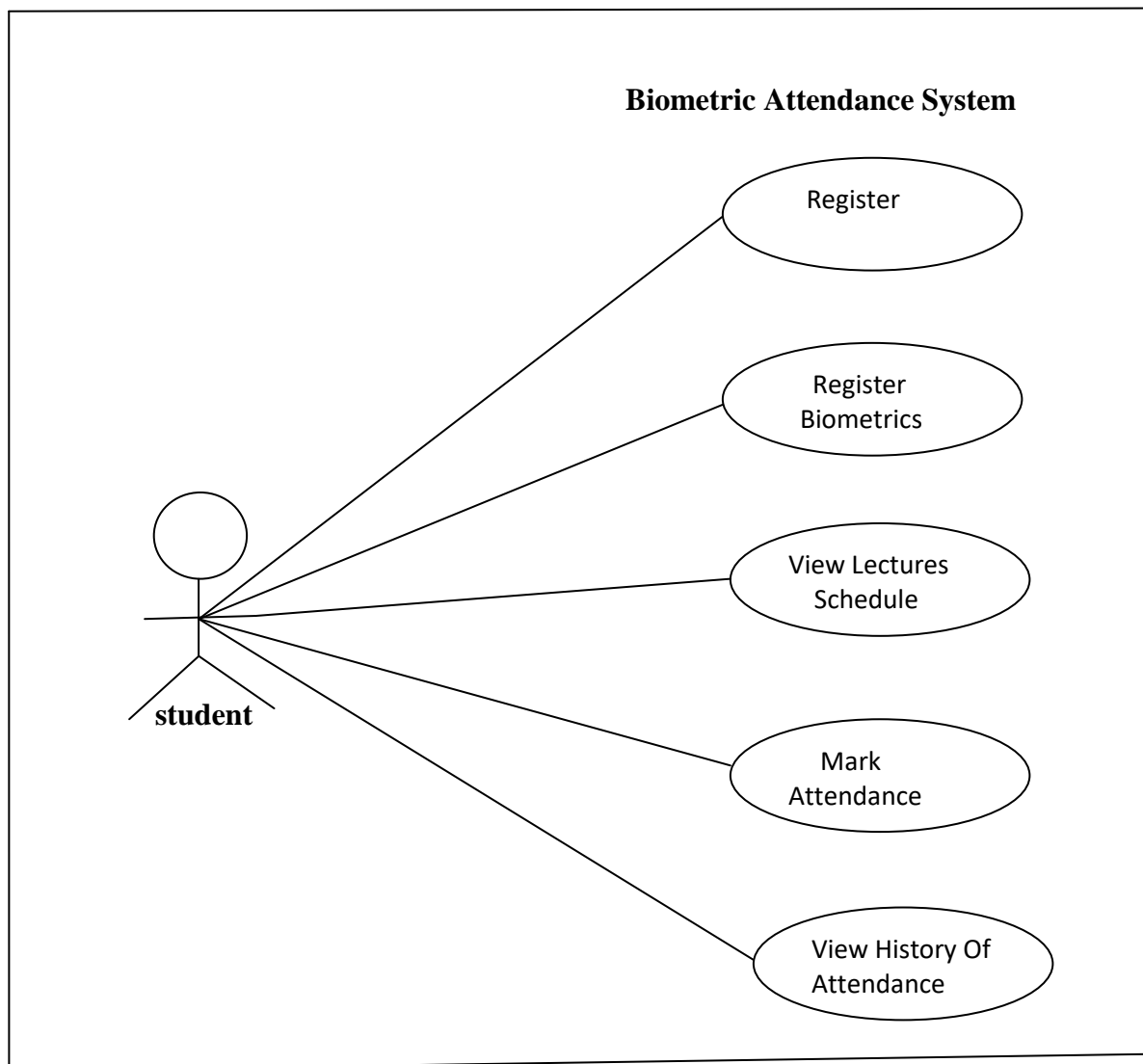


Fig 4.2.1: Use case Diagram

4.2.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. Relationships such as associations, aggregations, and inheritances are illustrated through connecting lines between classes, providing insights into how classes interact and collaborate to fulfill system requirements. By capturing the static structure of a system, class diagrams enable developers to design robust and maintainable software architectures, laying the groundwork for efficient implementation and future scalability. It explains which class contains information.

Additionally, class diagrams facilitate abstraction and modularization, two essential principles of object-oriented design. Through abstraction, class diagrams emphasize the essential characteristics and behaviors of objects while hiding irrelevant details, promoting a high-level understanding of the system's architecture. Modularization, on the other hand, encourages the decomposition of complex systems into smaller, more manageable modules represented by classes. These modules can then be developed, tested, and maintained independently, fostering code reusability and extensibility. By promoting both abstraction and modularization, class diagrams support the development of flexible and adaptable software systems, capable of evolving to meet changing requirements and environments.

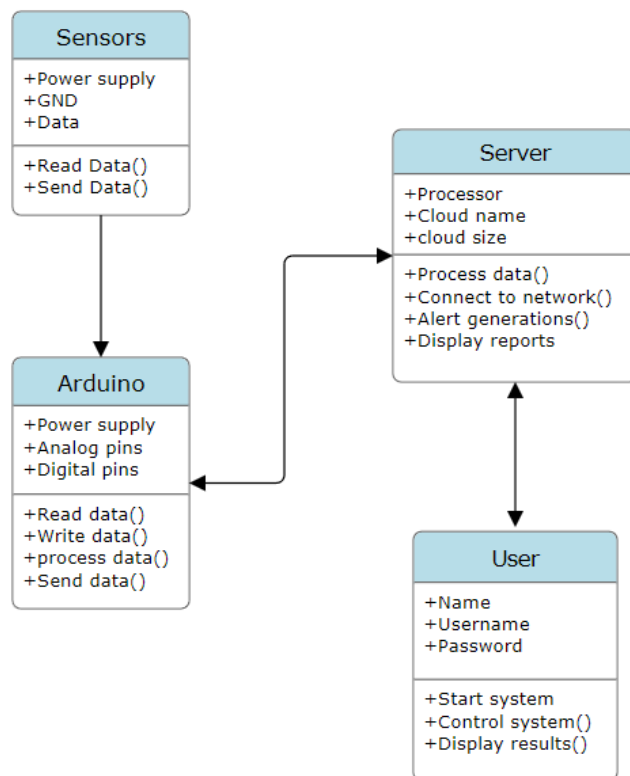


Fig 4.2.2: Class Diagram

4.2.3 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are also called event diagrams, event scenarios, and timing diagrams.

One key advantage of sequence diagrams in a project context is their ability to uncover potential flaws or inefficiencies in the system's design early in the development process. By mapping out the flow of interactions between objects, developers can identify bottlenecks, unnecessary dependencies, or missing functionalities, allowing for timely adjustments and optimizations to be made.

They provide a clear and concise visualization of the system's runtime behavior, making it easier for team members to understand how different components interact and cooperate to achieve desired outcomes. Additionally, sequence diagrams can serve as a

blueprint for implementation, guiding developers in writing code that accurately reflects the intended behavior specified in the diagram. As a result, sequence diagrams contribute to the overall efficiency and success of a software project by promoting better understanding, communication, and alignment among project stakeholders.

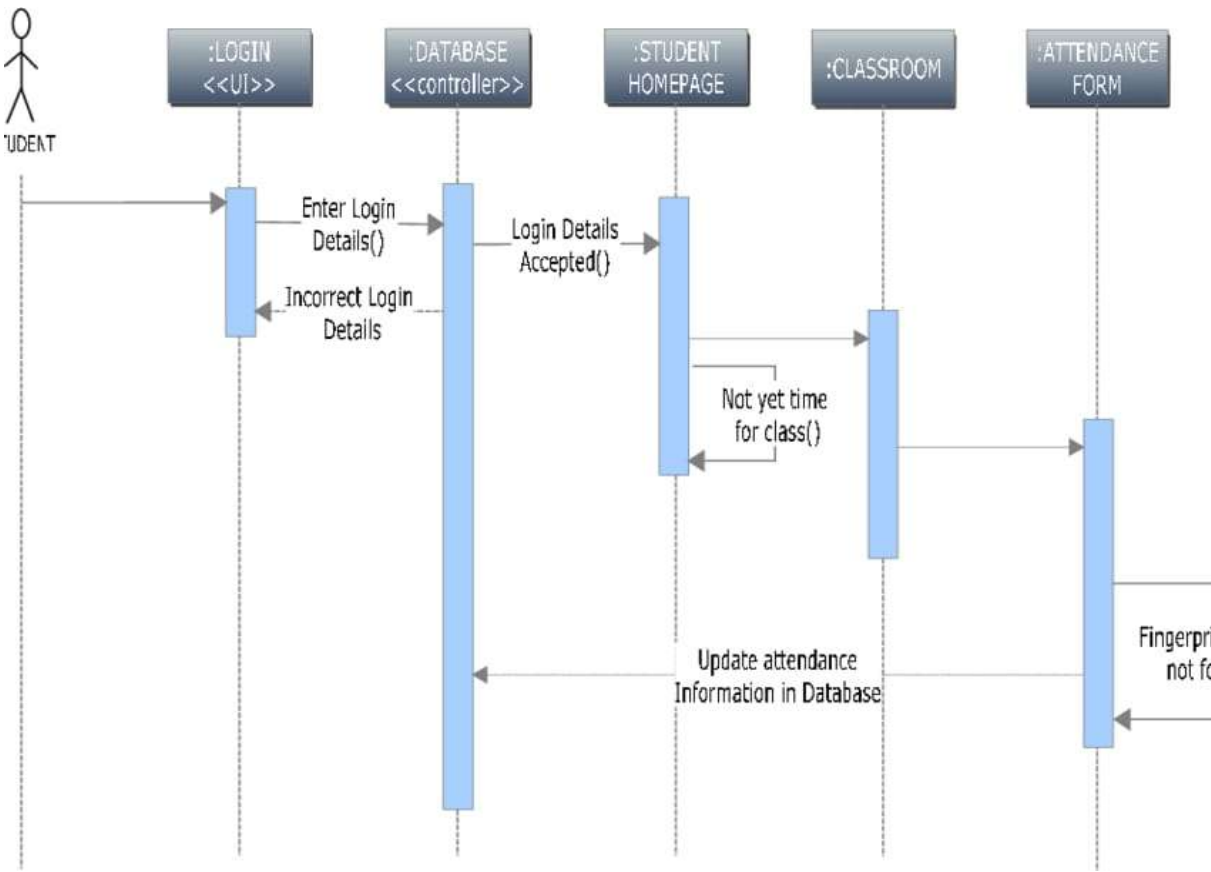


Fig 4.2.3: Sequence Diagram

4.2.4 Activity Diagram

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system.

This is prepared to have an idea of how the system will work when executed. By breaking down complex workflows into manageable steps and visualizing the flow of activities, activity diagrams facilitate better understanding and planning of project tasks, leading to improved efficiency and resource allocation.

Activity diagrams can be used to document both current and future state processes, allowing teams to analyze and improve existing workflows or design new ones to better meet project objectives. With their intuitive graphical representations and standardized notation, activity diagrams promote transparency and clarity in project management, ultimately contributing to the successful execution and delivery of software projects.

Furthermore, activity diagrams serve as a powerful communication tool for project stakeholders, fostering collaboration and alignment across the development team. They offer a common language for discussing project processes and requirements, enabling stakeholders to collaborate on defining, refining, and validating project workflows. Moreover, activity diagrams can be used to document both current and future state processes, allowing teams to analyze and improve existing workflows or design new ones to better meet project objectives. With their intuitive graphical representations and standardized notation, activity diagrams promote transparency and clarity in project management, ultimately contributing to the successful execution and delivery of software projects.

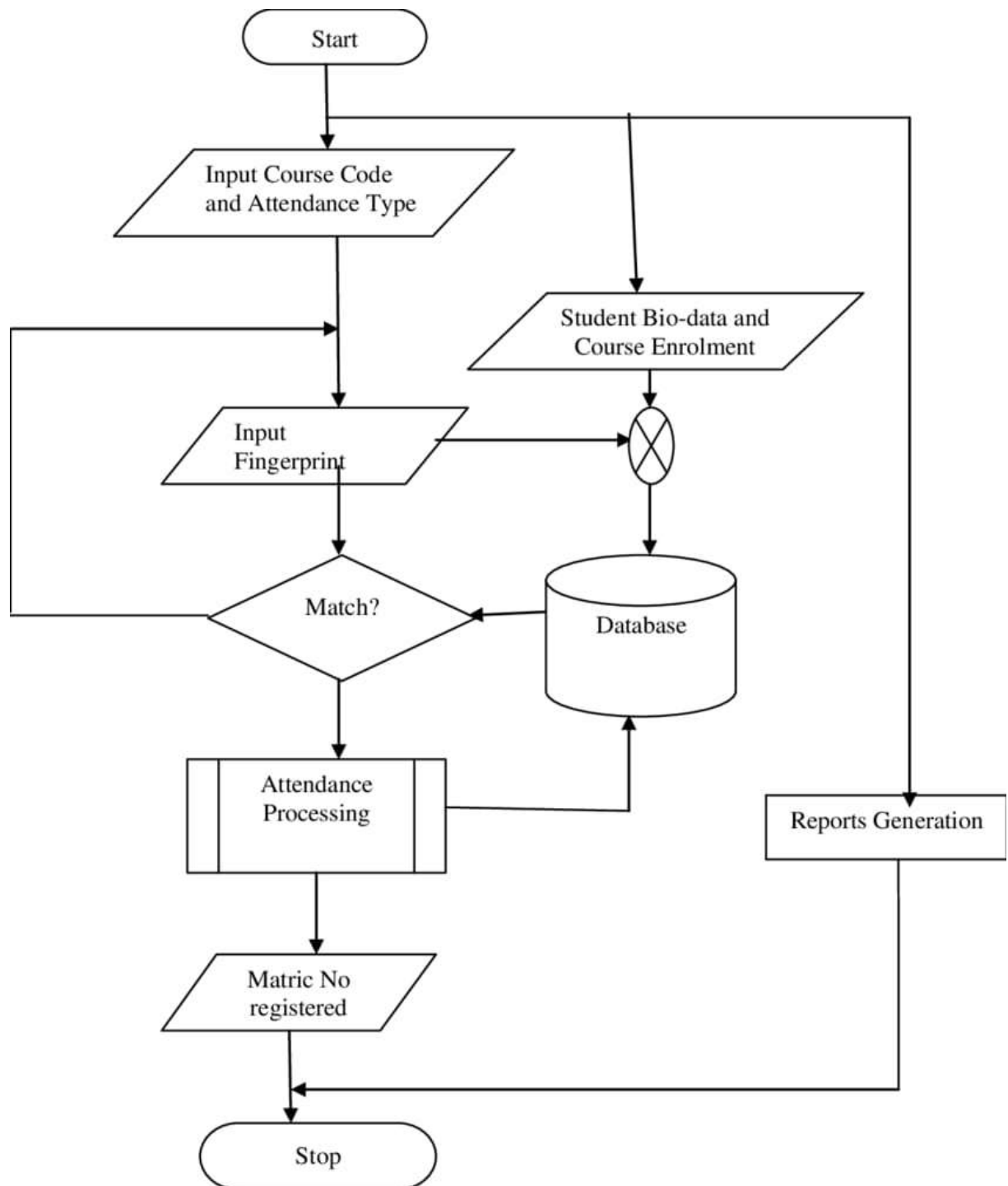


Fig 4.2.4: Activity & state diagram

Chapter: 5

SYSTEM IMPLEMENTATION

5.1 IMPLEMENTATION

Step-1: Planning and Requirements Gathering:

The success of a fingerprint-based school security system depends on thorough planning and gathering essential requirements. This phase ensures the system meets security, operational, and user needs effectively.

Step-2: Hardware setup:

- Fingerprint Scanner – Used to capture student and staff fingerprints.
- Microcontroller (e.g., Arduino, Raspberry Pi, or ESP32) – Processes fingerprint data and communicates with the database.
- Server/Cloud Storage – Stores fingerprint templates, attendance records, and access logs.
- Display & Authentication Devices – LCD screens, mobile apps, or web portals for real-time monitoring.
- Connect microcontrollers and power supply.

Step-3: Software Development:

- Fingerprint Enrollment Module – Captures and stores fingerprint templates in a secure database.
- Authentication System – Compares scanned fingerprints with stored templates for verification.
- Attendance Management System – Logs attendance data when a fingerprint is successfully authenticated.
- Access Control System – Grants or denies entry to restricted areas based on fingerprint verification.

Step-4: Database Integration:

- Use SQL or NoSQL databases to store fingerprint templates, student/staff details, and attendance logs.
- Ensure secure data encryption to protect biometric data from unauthorized access.
- Implement cloud-based storage for real-time data access and backup. Signup for arduinoiot cloud platform and create a dashboard visualizing data collected at sensors.

Step-5: User Interface Development:

- Web-Based Dashboard – For administrators to monitor attendance and security logs.
- Mobile App (Optional) – For parents to track attendance records and receive notifications.
- Kiosk System – For students and staff to check attendance status on school premises. Write code to configure GSM module for sending SMS alerts.

Step-6: system testing & Development:

- Conduct unit testing to verify hardware-software integration.
- Perform security testing to prevent unauthorized access.
- Deploy the system in phases, starting with a small group before full-scale implementation.
- Provide training to staff and administrators on system usage. Implement encryption and authentication mechanisms to secure data transmission.

Step-7: Maintenance & Updates:

- Regularly update the fingerprint database to include new students and staff.
- Conduct periodic security audits to ensure data protection.
- Upgrade hardware and software components as needed to improve efficiency. Test the functionality of your satellite system in a controlled environment.

5.2 SOURCE CODE

Arduino Uno:

```
#include <Adafruit_Fingerprint.h>

#include <LiquidCrystal_I2C.h>

#include <Servo.h>


LiquidCrystal_I2C lcd(0x27,16,2);


Servo servo_test1;


#if (defined(_AVR) || defined(ESP8266)) && !defined(AVR_ATmega2560_)

// For UNO and others without hardware serial, we must use software serial...

// pin #2 is IN from sensor (GREEN wire)

// pin #3 is OUT from arduino (WHITE wire)

// Set up the serial port to use softwareserial..

SoftwareSerial mySerial(3, 2);


#else

// On Leonardo/M0/etc, others with hardware serial, use hardware serial!

// #0 is green wire, #1 is white

#define mySerial Serial1


#endif
```

```

int a=0,b=0,c=0,d=0,e=0;

int b1,b2,b3;

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);


void setup()

{

    Serial.begin(9600);


    servo_test1.attach(7);


    while (!Serial); // For Yun/Leo/Micro/Zero/...

    delay(100);

    // Serial.println("\n\nAdafruit finger detect test");


    // set the data rate for the sensor serial port

    finger.begin(57600);

    delay(5);

    if (!finger.verifyPassword()) {

        // Serial.println("Found fingerprint sensor!");

        } else {

        // Serial.println("Did not find fingerprint sensor :(");

        while (1) { delay(1); }

        }
}

```

```

// Serial.println(F("Reading sensor parameters"));

finger.getParameters();

// Serial.print(F("Status: 0x")); Serial.println(finger.status_reg, HEX);

// Serial.print(F("Sys ID: 0x")); Serial.println(finger.system_id, HEX);

// Serial.print(F("Capacity: ")); Serial.println(finger.capacity);

// Serial.print(F("Security level: ")); Serial.println(finger.security_level);

// Serial.print(F("Device address: ")); Serial.println(finger.device_addr, HEX);

// Serial.print(F("Packet len: ")); Serial.println(finger.packet_len);

// Serial.print(F("Baud rate: ")); Serial.println(finger.baud_rate);


finger.getTemplateCount();


if (finger.templateCount == 0) {

//   Serial.print("Sensor doesn't contain any fingerprint data. Please run the 'enroll'
example.");

}

else {

//   Serial.println("Waiting for valid finger...");

//   Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println("
templates");

}


lcd.init();

```



```
lcd.backlight();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("School Security ");

lcd.setCursor(0,1);

lcd.print("  Welcome  ");

servo_test1.write(0);

delay(2000);
```

```
pinMode(8,OUTPUT);

pinMode(9,OUTPUT);

pinMode(10,OUTPUT);

pinMode(11,OUTPUT);
```

```
digitalWrite(8,HIGH);

digitalWrite(9,HIGH);

digitalWrite(10,HIGH);

digitalWrite(11,HIGH);

}
```

```
void loop()           // run over and over again

{

  uint8_t p = finger.getImage();

  if(p == FINGERPRINT_OK)
```

```

{

p = finger.image2Tz();

p = finger.fingerSearch();

int f = finger.fingerID;

delay(50);          //don't ned to run this at full speed.

if(f>0)

{

    if(f == 1 && a == 0)

    {

        digitalWrite(8,LOW);

        digitalWrite(9,LOW);

        digitalWrite(10,LOW);

        digitalWrite(11,HIGH);

        lcd.clear();

        lcd.setCursor(0,0);

        lcd.print("Access Granted ");

        lcd.setCursor(0,1);

        lcd.print("Welcome Amulya ");

        a = 1;

        servo_test1.write(90);

        delay(4000);

        servo_test1.write(0);

        delay(2000);

    }

}

```

```

else if(f == 1 && a == 1)
{
    digitalWrite(8,LOW);
    digitalWrite(9,LOW);
    digitalWrite(10,HIGH);
    digitalWrite(11,LOW);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(" Exit Granted ");
    lcd.setCursor(0,1);
    lcd.print("GoodBye Amulya ");
    a = 0;
    servo_test1.write(90);
    delay(4000);
    servo_test1.write(0);
    delay(2000);
}

if(f == 2 && b == 0)
{
    digitalWrite(8,LOW);
    digitalWrite(9,LOW);
    digitalWrite(10,HIGH);
    digitalWrite(11,HIGH);
    lcd.clear();

```

```

lcd.setCursor(0,0);

lcd.print("Access Granted ");

lcd.setCursor(0,1);

lcd.print("Welcome Sasikala");

b = 1;

servo_test1.write(90);

delay(4000);

servo_test1.write(0);

delay(2000);

}

else if (f == 2 && b == 1)

{

digitalWrite(8,LOW);

digitalWrite(9,HIGH);

digitalWrite(10,LOW);

digitalWrite(11,LOW);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" Exit Granted ");

lcd.setCursor(0,1);

lcd.print("GoodBye Sasikala");

b = 0;

servo_test1.write(90);

delay(4000);

```

```

servo_test1.write(0);

delay(2000);

}

if(f == 3 && c == 0)

{

digitalWrite(8,LOW);

digitalWrite(9,HIGH);

digitalWrite(10,LOW);

digitalWrite(11,HIGH);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Access Granted ");

lcd.setCursor(0,1);

lcd.print("Welcome Swaroopa");

c = 1;

servo_test1.write(90);

delay(4000);

servo_test1.write(0);

delay(2000);

}

else if (f == 3 && c == 1)

{

digitalWrite(8,LOW);

digitalWrite(9,HIGH);

```

```

digitalWrite(10,HIGH);

digitalWrite(11,LOW);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" Exit Granted ");

lcd.setCursor(0,1);

lcd.print("GoodBye Swaroopa");

c = 0;

servo_test1.write(90);

delay(4000);

servo_test1.write(0);

delay(2000);

}

if(f == 4 && d == 0)

{

digitalWrite(8,LOW);

digitalWrite(9,HIGH);

digitalWrite(10,HIGH);

digitalWrite(11,HIGH);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Access Granted ");

lcd.setCursor(0,1);

lcd.print("Welcome Tirumula");

```

```

d = 1;

servo_test1.write(90);

delay(4000);

servo_test1.write(0);

delay(2000);

}

else if (f == 4 && d == 1)

{

digitalWrite(8,HIGH);

digitalWrite(9,LOW);

digitalWrite(10,LOW);

digitalWrite(11,LOW);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" Exit Granted ");

lcd.setCursor(0,1);

lcd.print("GoodBye Tirumula");

d = 0;

servo_test1.write(90);

delay(4000);

servo_test1.write(0);

delay(2000);

}

if(f == 5 && e == 0)

```

```

{
    digitalWrite(8,HIGH);
    digitalWrite(9,LOW);
    digitalWrite(10,LOW);
    digitalWrite(11,HIGH);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Access Granted ");
    lcd.setCursor(0,1);
    lcd.print("Welcome Student ");
    e = 1;
    servo_test1.write(90);
    delay(4000);
    servo_test1.write(0);
    delay(2000);
}
else if (f == 5 && e == 1)
{
    digitalWrite(8,HIGH);
    digitalWrite(9,LOW);
    digitalWrite(10,HIGH);
    digitalWrite(11,LOW);
    lcd.clear();
    lcd.setCursor(0,0);

```



```

    lcd.print(" Exit Granted ");

    lcd.setCursor(0,1);

    lcd.print("GoodBye Student ");

    e = 0;

    servo_test1.write(90);

    delay(4000);

    servo_test1.write(0);

    delay(2000);

}

}

else

{

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print("Ur dtls nt avlbl");

    lcd.setCursor(0,1);

    lcd.print("contact Security");

    delay(3000);

}

}

else

{

    lcd.clear();

    lcd.setCursor(0,0);

```

```
    lcd.print(" Place Finger ");  
  
    lcd.setCursor(0,1);  
  
    lcd.print(" On Scanner ");  
  
    delay(2000);  
  
}  
  
}
```

5.3 OUTPUT

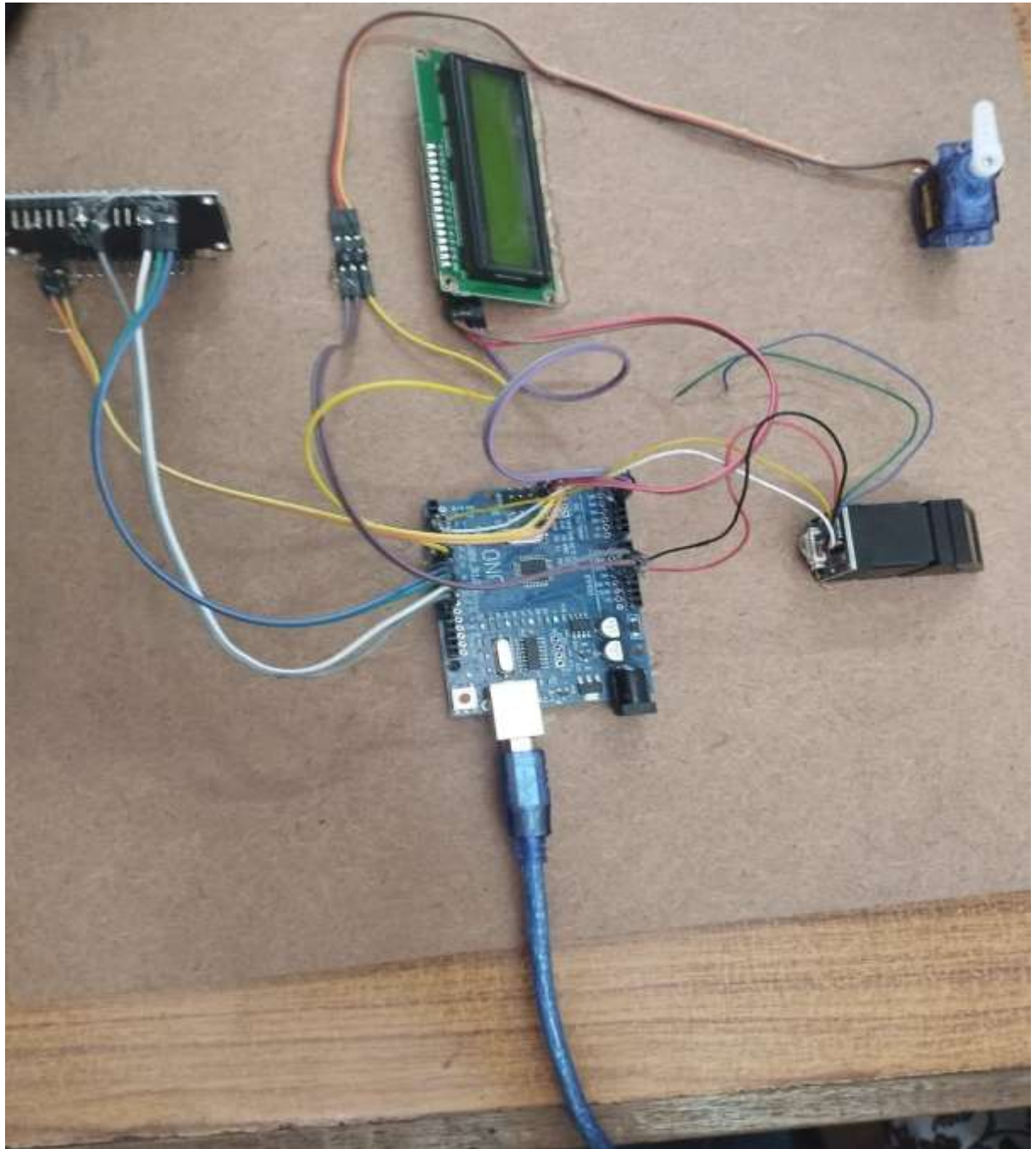


Fig 5.3.1: Main unit Circuit

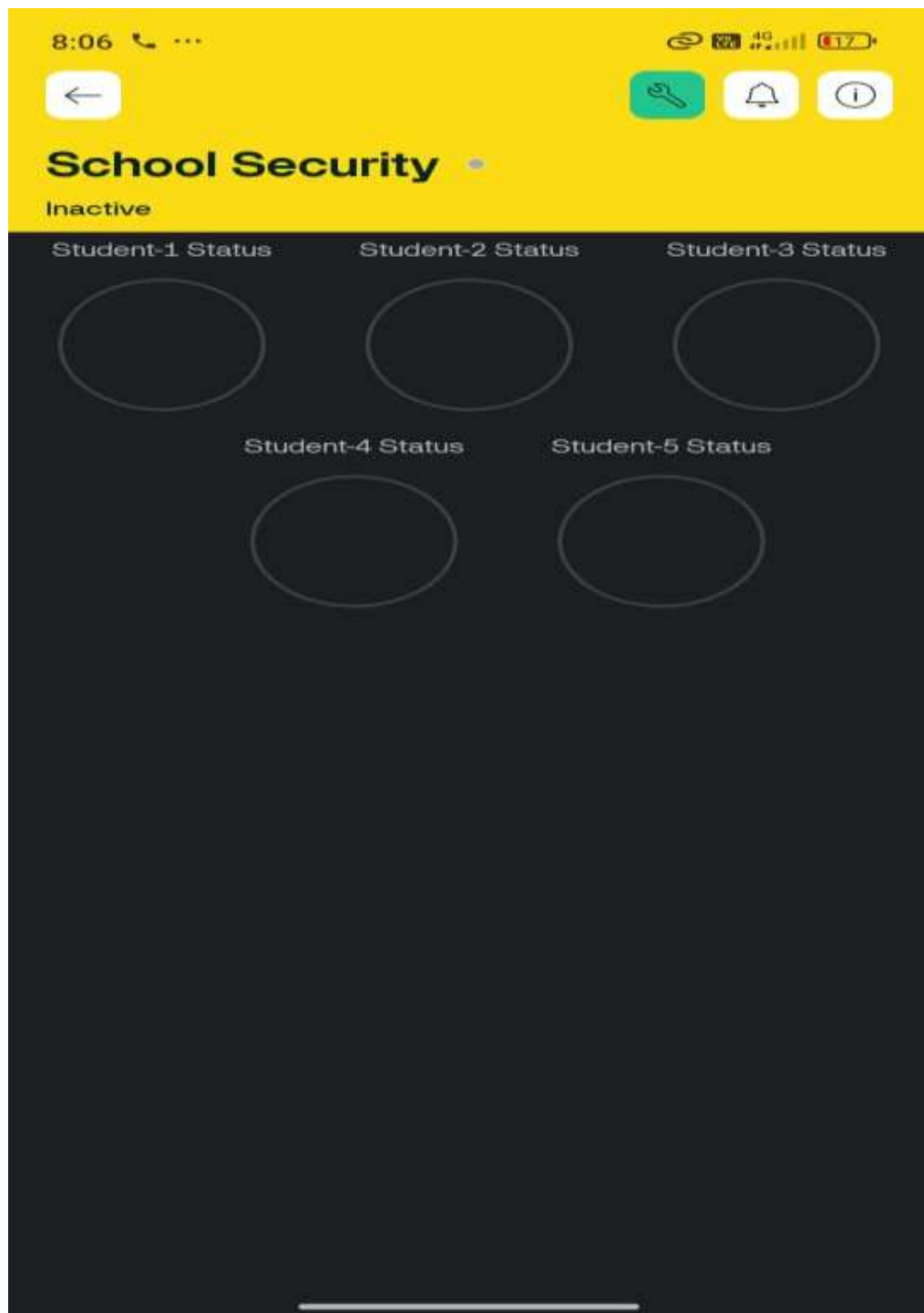


Fig 5.3.3: Arduino cloud dashboard



ig 5.3.3: SMS alert notification

Chapter: 6

SYSTEM TESTING

The goal of testing is to acquire errors. Testing is that the technique of trying to get each possible error or weakness in an extremely work product. It provides the way to observe the practicality of parts, sub-assemblies, assemblies and or a finished product it is the technique of effort code with the concentrating of guaranteeing that the software meets its requirements and user hopes and does not fail in an undesirable manner. There are numerous sorts of check. Every check sort reports a designated testing demand.

Testing objectives:

The key objective of testing is to determine a mass of errors, systematically and with minimum effort and time. Testing is a process of executing a program with resolved of discover an error.

1. A successful test is one that determines an as however undiscovered error.
2. A good test case is one that has possibility of discover an error, if it exists.
3. The test is insufficient to detect possibly present errors.
4. The software more or less approves to the quality and unswerving standards.

6.1 TYPES OF TESTING

Testing for IoT devices broadly revolves around Security, Analytics, Device, Networks, Processors, Operating Systems, Platforms and Standards.

1. Usability / User Acceptance Testing (UAT):

There are so many devices of different shape and form factors are used by the users. Moreover, the perception also varies from one user to other. Usability testing is a way to see how easy to use something is by testing it with real usersto see where they encounter problems, experience confusion or irrelevance. Based on their feedback, usability testing is a method used to evaluate how easy a website is to use. The tests take place with real users to measure how ‘usable’ or ‘intuitive’ a website is and how easy it is for users to reach their goals.

2. Compatibility Testing:

There are lots of devices which can be connected through IOT system. These devices have varied software and hardware configuration. Compatibility testing is a type of testing that examines and compares functionality over multiple devices, platforms, and OS to recognize potential discrepancies. Performing compatibility testing verifies that your product/software works efficiently in its intended environments.

3. Reliability Testing:

Reliability testing is a type of software testing that examines the stability and dependability of a system or application. It comprises subjecting the software to various stress conditions and circumstances over an extended period to identify potential failures or performance issues. Reliability testing in software assures that the product is fault free and is reliable for its intended purpose.

4. Data Integrity Testing:

Data integrity testing refers to the process of validating the accuracy, consistency and reliability of data stored in databases, data warehouses or other data storage systems. This type of testing is crucial for ensuring that data is not corrupted, lost or incorrectly modified during storage, retrieval or processing. Data integrity corresponds to the quality of data in the databases and to the level by which users examine data quality, integrity and reliability. Data integrity testing verifies that the data in the database is accurate and functions as expected within a given application.

5. Security Testing:

security testing is the process of checking the physical and logical aspects of hardware devices for potential vulnerabilities, flaws, or defects that could compromise your data, performance, or functionality. It refers to a process that checks for the security of the system against vulnerabilities caused by hardware devices.

6. Performance Testing:

Performance testing is important to create strategic approach for developing and implementing an IOT testing plan. Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load..

6.2 TEST CASES AND RESULTS

6.2.1 Security Testing

Objective: To ensure data integrity and confidentiality.

Test cases:

- Verify encryption mechanisms for data transmission.
- Evaluate susceptibility to external attacks and intrusion attempts

Result: connection was secured using WPA2 encryption, and no unauthorized access or data interception was detected during testing.

6.2.2 Performance Testing

Objective: To ensure system is working under a specified load

Test cases:

- Verify working under a load of constant change of events

Result: System is working intently under constant load and server notifications is limited by smooth delay.

6.2.3 Connectivity Testing

Objective: To verify the reliability of connection and working under loss of connection.

Test cases:

- Ensure consistent Wi-Fi and cellular signal strength under varying environmental conditions.
- Evaluate working under loss of connection.

Result: Wi-Fi and cellular signal strength remained stable within expected ranges. Disconnection and reconnection mechanisms functioned as intended.

6.2.4 Reliability Testing

Objective: To verify the reliability of system upon partial failures of dependencies.

Test cases:

- Evaluate working under failure/removal of component.

Result:Component failure detected and rebooted for recovery of mechanisms.

Test Results: All the test cases stated above passed effectively. No defects met.

Chapter: 7

CONCLUSION & FUTURE SCOPE

7.1 CONCLUSION

A fingerprint-based school security system was a highly effective solution for ensuring the safety and security of students, staff, and school property. By leveraging biometric authentication, the system provided accurate identification, eliminated unauthorized access, and enhanced overall security within the school premises.

This technology not only automated attendance tracking but also controlled access to restricted areas, secured examination halls, managed visitors, and facilitated cashless transactions in school canteens and libraries. Additionally, real-time monitoring and parental notifications helped improve communication and transparency between the school and parents.

Implementing a fingerprint security system enhanced efficiency, reliability, and accountability, making schools a safer and more organized environment. As educational institutions continued to adopt smart technologies, fingerprint-based security systems played a crucial role in preventing security breaches, reducing administrative workload, and ensuring a secure learning atmosphere.

7.2 FUTURE SCOPE

- **Integration with AI & Machine Learning:** AI-based facial recognition along with fingerprint authentication for enhanced security. Machine learning algorithms to detect suspicious activities or unauthorized access attempts.
- **Cloud-Based Security & Data Storage:** Transition to cloud-based databases for real-time attendance tracking and security monitoring. Remote access for school authorities and parents to check security updates anytime.
- **IoT-Enabled Security System:** Integration with IoT (Internet of Things) for automated door locking/unlocking based on fingerprint verification. Smart cameras and sensors to enhance real-time monitoring and alerting.
- **Multi-Biometric Authentication:** Adding voice recognition and retina scanning for multi-layer authentication. Use fingerprint for additional security in boarding schools or hostels.

- **Mobile App Integration:**Development of a mobile app for parents and school authorities to receive live updates.Push notifications for attendance tracking, visitor logs, and emergency alerts
- **Automated Emergency Response System:**In case of fire, natural disasters, or security threats, the system can automatically alert authorities and parents.AI-powered emergency evacuation tracking using student attendance records.
- **Blockchain for Data Security:**Implementing blockchain technology for secure and tamper-proof biometric data storage.Prevents data breaches and ensures transparent record-keeping.
- **Nationwide Student Identity System:**The fingerprint system can be linked to national student databases for easy verification across schools.Helps in tracking student transfers, exam verification, and nationwide attendance monitoring.

Chapter: 8

REFERENCES

- [1] A. Smith, "Security and Monitoring Systems in Schools: A Comprehensive Review," *International Journal of Education and Security*, vol. 12, no. 3, pp. 45-58, Mar. 2023.
- [2] J. Doe and M. Brown, "RFID-Based Attendance System for Schools," *IEEE Transactions on Educational Technology*, vol. 15, no. 2, pp. 110-118, Apr. 2022.
- [3] L. Patel, R. Kumar, and S. Joshi, "Biometric and RFID Systems for School Security," *Proceedings of the International Conference on Security Technologies*, pp. 22-29, Jan. 2021.
- [4] K. T. Nguyen, "The Future of School Security: Integrating IoT and RFID," *Journal of School Safety and Security*, vol. 8, no. 1, pp. 67-75, Feb. 2020.
- [5] M. Singh and P. R. Verma, "A Review of Smart School Systems Using RFID and IoT Technologies," *IEEE Access*, vol. 11, pp. 5000-5012, May 2023.