A

Mini Project Report

On

FINGERPRINT BASED EXAM HALL AUTHENTICATION

Submitted in partial fulfillment of the requirements for the award degree

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

PRESENTED BY

Varimadla Rahul - 22681A0456

Dharma Ganesh reddy -22681A0419

Kukatla Narender -22681A0432

Ganapuram Uday kiran -22681A0424

Under the Guidance of

P.THIRUPATHI

Assistant Professor

CHRISTU JYOTI INSTITUTE OF TECHNOLOGY & SCIENCE

Affiliated to JNTUH, Hyderabad

Colombo Nagar, Yeshwanthapur, Jagaon-506167, Telangana.

2024-2025

CHRISTU JYOTI INSTITUTE OF TECHNOLOGY & SCIENCE

Affiliated to JNTUH, Hyderabad

Colombo Nagar, Yeshwanthapur, Jagaon-506167, Telangana

2024-2025

Dept. ECLECTRONCS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"FINGERPRINT BASED EXAM HALL AUNTHENTICATION"** is bonafied work done by **V.Rahul(22681A0456), D.Ganesh reddy (22681A0419), K.Narender(22681A0432), G.Uday kiran(22681A0424)** in partial fulfilment of the requirements for the award of BACHELAR OF TECHNOLOGY IN **"ELECTRONICS AND COMMUNICATION ENGINEERING"** by J.N.T.University Hyderabad during the academic year 2024-2025.

Signature of Guide
P.Thirupathi
(Assistant Professor)

Signature of HOD
MR. ALLANKI SANYASI RAO
(Assoc.Prof.)

Signature of External Examiner

CHRISTU JYOTI INSTITUTE OF TECHNOLOGY & SCIENCE

Affiliated to JNTUH, Hyderabad

Colombo Nagar, Yeshwanthapur, Jagaon-506167, Telangana

2024-2025

Dept. ECLECTRONCS AND COMMUNICATION ENGINEERING

Institute Vision and Mission

VISION

- To admit and groom students from rural background and be a truly rural technical institute benefiting society and nation as a whole institute.

MISION

- The mission of the institute is to create, deliver and refine knowledge. Being a rural technical institute, our mission is to.
- Enhance our position to one of the best technical institution and to measure our performance against the highest defined standards.
- Provide highest quality learning environment to our students for the ever-greater well-being so as to equip them with highest technical and professional ethics.
- Produce engineering graduates fully equipped to meet the ever-growing needs of industry and society.

Principal Signature

CHRISTU JYOTI INSTITUTE OF TECHNOLOGY & SCIENCE

Affiliated to JNTUH, Hyderabad

Colombo Nagar, Yeshwanthapur, Jagaon-506167, Telangana

Dept. ECLECTRONCS AND COMMUNICATION ENGINEERING

2024-2025

Department Vision and Mission

VISION

- To be an established centre of excellence in Electronics and Communication Engineering facilitating youth towards professional, leadership and industrial needs.

MISSION

- Impart theoretical and practical technical of high standard with quality resources and collaborations.
- Organize trainings and activities Overall personality development in time with industrial need.
- Provide innovation towards sustainable solutions with multi discipline team work with ethics.

HOD-ECE

DECLARATION

We hereby declare that the project entitled **"Fingerprint Based Exam hall Authentication system",** which is being submitted as Mini Project in Electronics and Communication Engineering to Christu Jyothi Institute of technology & Science, is an authentic record of our genuine work one under the guidance of Mr.Allanki Sanyasi Rao, Asso.Prof. & HOD.

Project Members

Varimadla Rahul -22681A0456

Dharma Ganesh reddy -22681A0419

Kukatla Narender -22681A0432

Ganapuram Uday kiran -22681A0424

ACKNOWLEDGEMENT

We hereby express our sincere gratitude to the Management of **Christu Jyoti Institute of Technology & Science** for their kind encouragement bestowed up on us to do this Mini project.

We earnestly take the responsibility to acknowledge the following distinguished personalities who graciously allowed our project work successfully.

We express our sincere thanks to our director **Rev.Fr.D.Vijay Paul Reddy**, Principal **Mr.Dr.S. Chandrasekhar Reddy** for his encouragement, which has motivate due to strive hard to excel in our discipline of engineering.

We are greatly indebted to the professor and head of the department

Mr. Allanki Sanyasi Rao, Assistant Professor for motivation and guidance through the course of this project work. He has been responsible for providing us with lot of splendid opportunities, which has shaped our career. His advice ideas and constant support has engaged us on and helped us get through in difficult time.

We express our profound sense of appreciation and gratitude to our guide **P.Thirupathi, Assistant Professo**r for providing generous assistance, and spending many hours of valuable time with us. This excellent guidance has made the timely completion of this mini project.

We express our profound sense of appreciation and gratitude to our **Project Coordinator, K.Amarender, Asst.Prof**. For providing assistance and spending many hours of his valuable time with us.

Last but not least, we express our gratitude to Teaching and Non-Teaching staff of the Department of Electronics and communication for their needy and continuous support in technical assistance.

ABSTRACT

In educational institutions, ensuring the integrity of exams and preventing malpractice is a growing concern. Traditional methods of student authentication during exams, such as student ID cards or roll call, are prone to human errors and fraud. This paper proposes a **fingerprint-based exam hall authentication system** to improve security and ensure accurate identity verification. By leveraging biometric fingerprint recognition, the system provides a more secure, efficient, and automated solution for confirming student identity before and during exams.

The proposed system captures the student's fingerprint using a fingerprint scanner, which is then matched with a pre-enrolled fingerprint database to authenticate the student. This approach eliminates the possibility of impersonation and helps to prevent cheating. Additionally, the system records each authentication event, creating a reliable log that can be used for audit purposes.

The implementation of the fingerprint-based system not only enhances security but also streamlines the exam process by reducing the need for manual checks and enhancing the overall exam experience. Furthermore, the system can be integrated with existing exam management software to facilitate seamless operations.

CONTENTS

List of Figures

Chapter 01-INTRODUCTION

1.1. Introduction to embedded system.

An embedded system is a specialized computing system designed to perform dedicated functions or tasks within a larger system. Unlike general-purpose computers, embedded systems are optimized for specific applications and often have constraints in terms of processing power, memory, and storage. They are typically integrated with hardware and run software (often called firmware) that directly interacts with that hardware.

Embedded systems are used in a wide range of applications, from household appliances like washing machines and microwaves to critical systems in automobiles, healthcare devices, industrial machines, and consumer electronics. These systems are designed to be reliable, efficient, and real-time, often operating continuously with minimal human intervention.

Embedded systems are integral parts of modern life, from controlling household appliances to ensuring safety in automobiles. These systems are usually designed with a focus on efficiency, reliability, and specific functionality. As technology evolves, the role of embedded systems is growing, especially in the realm of the Internet of Things (IoT), where many devices are becoming interconnected and smarter.

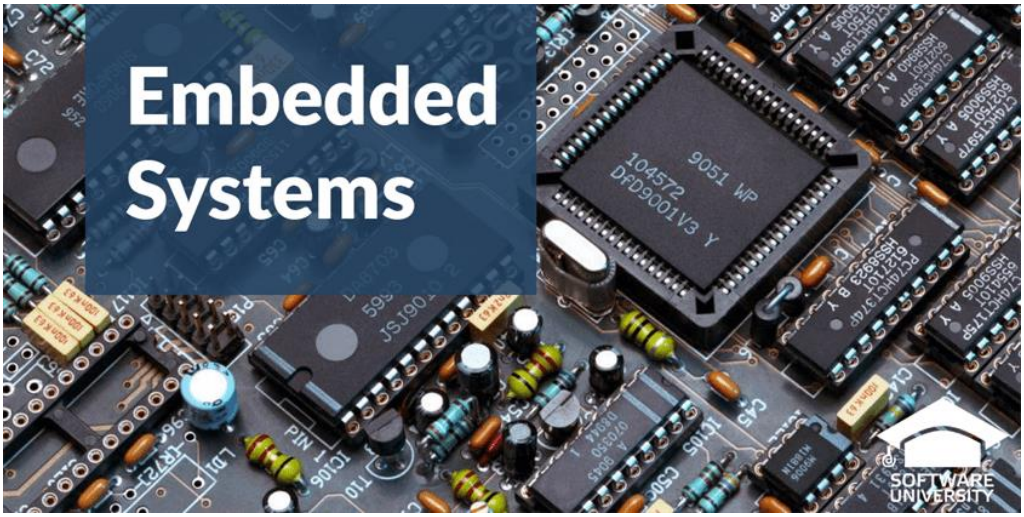


Figure: Embedded System.

1.2. Types of embedded system

i. **Standalone Embedded Systems:** These systems function independently. For example, a microwave oven or an alarm system.

ii. **Real-Time Embedded Systems:** These systems have strict timing requirements and must process inputs within a specific timeframe. An example is a car's anti-lock braking system (ABS).

iii. **Networked Embedded Systems:** These systems are connected to a network to communicate with other systems. For example, smart home devices or industrial automation systems.

iv. **Mobile Embedded Systems:** These are portable and often battery-powered devices, such as smart phones, tablets, or wearable's.

1.3. Components of embedded system

i. **Microcontroller/Microprocessor**: The "brain" of the embedded system. It processes inputs and executes control functions. Microcontrollers (MCUs) are commonly used in embedded systems due to their small size and integrated components.

ii. **Memory**: Embedded systems have different types of memory:

- **RAM (Random Access Memory)**: Temporary storage for data that is being processed.
- **ROM (Read-Only Memory)**: Permanent storage for program code and firmware that doesn't change.
- **Flash Memory**: A type of non-volatile memory often used for storing firmware or system settings.

iii. **Input/output (I/O) Interfaces**: These allow the embedded system to interact with the external world (sensors, actuators, displays, buttons, etc.). For example, a temperature sensor input might trigger a heating element output.

iv. **Power Supply**: Embedded systems often require specific power management systems to ensure they operate efficiently within power constraints.

v. **Software/Firmware**: The software in an embedded system is usually tightly coupled with the hardware and is often written in low-level programming

languages like C or assembly. It handles specific functions like reading sensor data, processing inputs, controlling outputs, etc

1.4 .Characteristics

i. **Task-Specific:** Embedded systems are designed to perform specific tasks or a set of tasks. For example, the firmware in a microwave oven or the software in a washing machine.

ii. **Real-Time Operation:** Many embedded systems must function in real-time, meaning they must process inputs and generate outputs within a strict time frame. For instance, airbag deployment systems in vehicles need to react to collisions in milliseconds.

iii. **Resource Constraints:** These systems often run on microcontrollers or microprocessors with limited computing power, memory, and storage compared to general-purpose computers.

iv. **Reliability and Stability:** Embedded systems are expected to be highly reliable and stable over long periods, often running 24/7 without frequent maintenance.

v. **Integration with Hardware:** Embedded systems are tightly integrated with their hardware. The software is typically tailored to interact directly with the hardware components, such as sensors, actuators, and communication interfaces.

vi. **Low Power Consumption:** Many embedded systems are designed to run on battery power or consume minimal electricity, especially in portable or remote applications like wearable devices or environmental sensors.

1.5. Applications

- **Consumer Electronics:** Smartphone's, smart TVs, and digital cameras use embedded systems for functionality like image processing, media playback, and connectivity.
- **Automotive:** Embedded systems control anti-lock braking systems (ABS), engine control units (ECUs), and airbag deployment in vehicles.
- **Healthcare:** Medical devices like pacemakers, insulin pumps, and wearable health monitors rely on embedded systems to manage critical health functions.

- **Industrial Automation:** Embedded systems control robotics, manufacturing processes, and machinery in factories.
- **Home Automation:** Smart home devices like thermostats, lights, and security systems use embedded systems for automation and control.
- **Telecommunications:** Embedded systems are used in network routers, base stations, and mobile communication devices.
- **Transportation:** Smart traffic lights, toll systems, and passenger information systems rely on embedded technologies for efficient operation

1.6 Vision

The **vision of embedded systems** is to create smarter, more efficient, and interconnected devices that enhance daily life. This includes:

i. **Ubiquitous Connectivity:** Enabling IoT for smarter homes, cities, and industries.
ii. **Enhanced Intelligence:** Integrating AI for real-time decision-making and optimization.
iii. **Miniaturization:** Making systems smaller, more compact, and efficient.
iv. **Low Power Consumption:** Focusing on energy efficiency, especially in battery-powered devices.

Chapter 02-Literature Survey

The **literature survey** for fingerprint-based exam hall authentication systems explores existing research, technologies, and implementations related to biometric authentication, specifically using fingerprint recognition. It highlights the trends, advantages, and challenges of using biometric systems in secure environments such as exam halls.

Biometric Authentication Systems:

Biometric authentication systems use physical characteristics, such as fingerprints, face recognition, or iris patterns, for identity verification. Among these, **fingerprint-based authentication** has emerged as the most widely used method due to its high accuracy, ease of implementation, and cost-effectiveness.

- **Fingerprint Recognition Technology:** Research has shown that fingerprint recognition systems offer an accuracy rate of over **99%** in identifying individuals, making them highly reliable in applications like access control and security. These systems extract unique features from the fingerprint, known as minutiae points (such as ridge endings and bifurcations), to create a template for comparison during the verification phase (Jain et al., 2000).

Fingerprint Authentication in Exam Hall:

In recent years, the application of fingerprint authentication in **educational environments**, particularly in **exam halls**, has been studied for its potential to enhance security, reduce cheating, and improve the efficiency of exam administration.

- **Automated Attendance and Access:** Fingerprint-based systems are increasingly being proposed for managing exam hall attendance. Research indicates that these systems can automate the process of verifying student identity at the entrance, preventing impersonation, and creating a digital log for real-time attendance tracking (Khan et al., 2016).
- **Prevention of Impersonation:** One of the primary concerns in exam settings is **impersonation**, where someone else takes the exam in place of the registered candidate. Fingerprint-based authentication eliminates this risk, as

fingerprints are unique to each individual, ensuring that only the registered student can enter the exam hall (Tayal et al., 2017).

2.1. Challenges and Considerations

While fingerprint-based authentication systems offer many benefits, several challenges must be considered when implementing them in exam halls.

- **Fingerprint Quality:** Variations in fingerprint quality due to age, cuts, or dirt on the fingers can sometimes lead to authentication failures. Research by Jain et al. (2000) discusses the need for advanced algorithms to handle such discrepancies and ensure the reliability of fingerprint recognition.
- **Privacy Concerns:** The storage of biometric data raises privacy issues. Ensuring that student fingerprint data is securely encrypted and complies with privacy regulations (e.g., GDPR) is essential. Researchers have emphasized the need for secure data storage practices (Zhang et al., 2018).
- **Cost of Implementation:** While cost-effective in the long term, the initial setup costs for fingerprint scanners and associated infrastructure may be a challenge for smaller institutions or exam centres. Some studies suggest that hybrid systems combining fingerprint recognition with other authentication methods can reduce costs while maintaining security (Sankaran et al., 2018).

2.3. Future Trends

The future of fingerprint-based authentication systems in exam halls is moving towards more integrated, scalable, and user-friendly solutions. Future developments may include:

- **Multi-factor Authentication:** Combining fingerprint recognition with other forms of biometric authentication (e.g., face recognition or smart cards) to enhance security and redundancy.

Chapter 03- Project Overview

3.1 Introduction

Fingerprint-based exam hall authentication is a security system that uses biometric fingerprint recognition to verify the identity of students entering an exam hall. It ensures that only the enrolled candidates are allowed access to the exam room, providing a secure and automated way of monitoring and managing exam attendance. The system captures unique fingerprint patterns from each student and matches them against a stored database, offering a foolproof method of identification.

This approach eliminates the risks of impersonation and ensures that the right person takes the exam. It also helps in maintaining an accurate record of student attendance, reducing the chances of manual errors and fraud during examinations. With the rise of technology and the need for secure systems in educational settings, fingerprint-based authentication is becoming an increasingly popular solution in exam halls to uphold integrity and streamline the process.

3.2 Existing System

Currently, several authentication methods are in place to verify the identity of students before entering exam halls. These systems vary in terms of security, accuracy, and user-friendliness. Below are some of the most common existing systems:

1. Manual Identity Verification (Paper-based)

- **Method:** This is the traditional approach where students show their ID cards, and invigilators manually check the list of candidates.

2. Smart Cards and ID card Scanners

- **Method:** Students are provided with ID cards embedded with a unique identifier (e.g., RFID or barcode). These cards are scanned upon entry to verify identity..

3. Face Recognition Systems

- **Method:** Facial recognition technology scans and identifies students based on their unique facial features..

4. Barcode/QR Code Scanning

- **Method:** Students receive a QR code or barcode that is scanned for entry to verify their identity.

5. PIN/Password-based Authentication

- **Method:** Students enter a personal identification number (PIN) or password to verify their identity.

3.2.1 Limitations of Existing System

- **Security Concerns:** Traditional methods like ID cards or PIN-based systems can be easily bypassed, leading to impersonation or fraud.
- **Time Consumption:** Manual verification and some electronic methods can be slow, leading to delays in the exam process.
- **Operational Challenges:** Systems like face recognition or card scanning require additional hardware and proper maintenance, which could incur high setup and operational costs.
- PINs or passwords can be forgotten or shared.
- Susceptible to security breaches if the password is leaked or guessed.
- Requires students to remember and enter the correct credentials.
- Can be affected by poor lighting or changes in appearance (e.g., glasses or makeup).
- Privacy concerns about storing and processing biometric data.
- May have higher setup costs and maintenance requirements.
- Time-consuming and prone to human error.
- Higher chances of impersonation or incorrect attendance records.
- Manual handling leads to delays in checking in students.

3.3 Proposed System

The proposed system for **fingerprint-based exam hall authentication** aims to enhance the security, efficiency, and integrity of the exam process by using biometric fingerprint recognition technology to verify the identity of students entering the exam hall.

Multi-Factor Authentication:

- Fingerprint + Facial Recognition: Students must pass both fingerprint and facial recognition scans to gain entry.
- Smart Card + PIN: Students use a smart card and also enter a PIN for verification.

Contactless Biometrics:

- **3D Facial Recognition:** Captures a 3D image of the face without needing to touch a sensor.
- **Iris Scanning:** Scans the iris of the eye from a distance.

3.3.1 Benefits of proposed system

1. **Enhanced Security:**
    - Fingerprints are unique to each individual, providing a highly secure method of authentication.
2. **Efficiency:**
    - The fingerprint verification process is quick and automated, reducing waiting times and ensuring smooth entry for students.
3. **Accuracy and Reliability:**
    - Fingerprint-based authentication offers a high level of accuracy and reduces the chances of human error.
4. **Real-Time Monitoring:**
    - Real-time data collection helps with tracking student attendance and monitoring exam processes, making it easier to detect irregularities.
5. **Scalability**

Chapter 04 – HARDWARE COMPONENTS

4.1 Power Supply

In embedded systems-based IoT projects, power supply management is even more critical because embedded devices are often designed to be small, low-power, and capable of running autonomously for extended periods. The choice of power supply in such projects depends on the device's use case, the complexity of the system, and the environment in which it operates.

In embedded-based IoT projects, you will find that the power requirements and solutions are more focused on **efficiency**, **compactness**, and **long-term autonomy** due to their limited resources (e.g., small microcontrollers, low-power sensors, and communication modules).

4.1.1 Power Management and Efficiency

- **Low-Power Components:** In embedded IoT systems, **low-power microcontrollers (MCUs)** and **low-power peripherals** are critical.Popular choices include:
  - ARM Cortex-M Series
  - AVR and PIC
- **Power Management ICs:** Embedded systems often need specialized **power management ICs (PMICs)** that handle functions like voltage regulation, battery charging, and power distribution. These ICs ensure that the device remains within its operating voltage and can extend battery life.
- **Sleep and Idle States:** Leveraging sleep and idle states is critical in embedded systems. These modes ensure that the device only consumes the minimum power when not actively performing tasks.
  - **Low-Power Modes (e.g., Deep Sleep, Hibernate**
  - **Intermittent Operation**
- **Wireless Communication Power Considerations:**
  - **Wi-Fi**
  - **Bluetooth Low Energy (BLE)**
  - **LoRa**
  - **Zigbee**

4.1.2 Block diagram

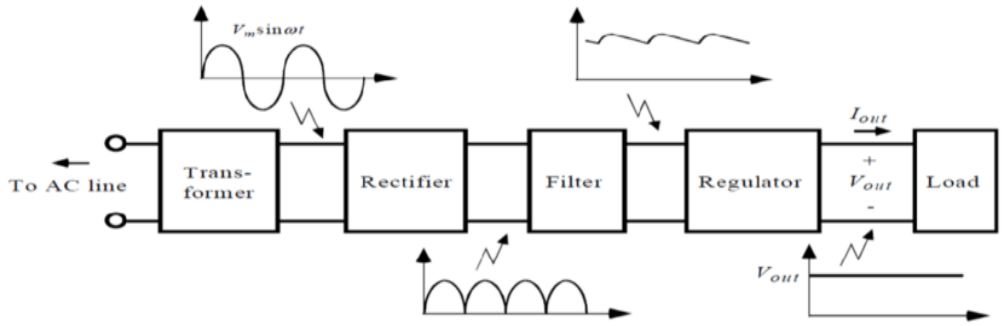


Figure- Components of power supply.

- **Transformer:** A transformer is typically used when converting AC voltage from a higher level (e.g., from an electrical outlet) to a lower, more suitable level required by the system (e.g., 5V, 12V). The transformer operates on alternating current (AC) and is often found in AC-DC power supplies**.**
- **Rectifier:** A rectifier is a key component in power supplies, particularly in converting alternating current (AC) to direct current (DC). Since many embedded and IoT systems require DC power, a rectifier plays a crucial role in ensuring that the power supplied is in the correct form.
- **Filter:** A filter is used to smooth the output from a rectifier to produce a more stable DC voltage. After the rectification process, the output is typically a pulsating DC signal, which still contains ripples due to the alternating nature of the input AC signal. The purpose of the filter is to reduce these ripples and provide a smoother DC output, suitable for powering sensitive electronic devices like those in embedded systems and IoT projects.
- **Regulator:** A voltage regulator is a crucial component in a power supply system, especially in embedded IoT-based projects, as it ensures that the voltage supplied to the system is stable and within a specified range. Voltage regulators protect sensitive components from overvoltage or under voltage conditions, which could otherwise damage the system or lead to unreliable performance.
- **Load:** The load is typically the final destination of the regulated DC voltage. It consumes the electrical energy provided by the power supply to perform its intended functions, such as processing data, sensing the environment, or transmitting signals in an IoT system.

4.2 Introduction to Arduino.

Arduino is an open-source electronics platform used for building a wide range of projects in embedded systems, robotics, IoT (Internet of Things), and automation. It provides an easy-to-use hardware and software environment that makes it accessible to both beginners and advanced engineers for creating interactive electronic devices. Arduino is popular for its simplicity, versatility, and accessibility, making it ideal for beginners and professionals alike.

The platform enables users to control sensors, motors, lights, and other devices by writing and uploading code to the microcontroller. Arduino boards are widely used in applications such as robotics, home automation, IoT (Internet of Things) projects, and educational purposes. Its open-source nature encourages a global community to collaborate, share projects, and expand its capabilities.

Figure: Arduino UNO

4.2.1 Features of Arduino

- **Microcontroller-Based:**
  - At the heart of Arduino is a microcontroller, which is a small computing device capable of processing data and controlling other electronic components**.**
- **Open-Source:**

- Both the **hardware** and **software** are open-source. This means anyone can access, modify, and share the designs and code.

- **User-Friendly:**
  - Arduino simplifies electronics and programming for beginners. It comes with a simple **IDE (Integrated Development Environment)** where users write and upload code to the board, often using a language based on **C/C++**
- **Versatility:**
  - Arduino boards can interact with a wide range of sensors, motors, displays, and other components, making it suitable for many types of applications, from simple automation to complex IoT projects.

4.2.2 Pin description

Arduino boards, such as the Arduino Uno, have multiple pins that are used to interface with external devices (like sensors, actuators, and other modules). These pins can be categorized into digital pins**,** analog pins**,** power pins, and communication pins. Below is a description of the different types of pins on an Arduino board:

I. Digital Pins

- **Function:** Digital pins can be configured as either input **or** output**.**
- **Pin Numbers:** On most Arduino boards like the Arduino Uno, there are 14 digital pins **(**numbered from 0 to 13).
- **Modes:**
  - **INPUT:** Used to read a signal (e.g., from a button or sensor).
  - **OUTPUT:** Used to send a signal (e.g., to control an LED or motor).
  - **PWM:** Some digital pins (marked with a **~** symbol, such as pins 3, 5, 6, 9, 10, 11**)** support Pulse Width Modulation (PWM**)**, which allows you to simulate analog output by rapidly switching the pin on and off.

II. Analog Pins

- **Function:** These pins can read analog signals (voltage levels ranging from 0 to 5V) and convert them into digital values.

- **Pin Numbers:** On the Arduino Uno, there are 6 analog input pins (labelled A0 to A5).
- **Resolution:** Arduino uses an ADC (Analog-to-Digital Converter) with a 10-bit resolution**,** meaning it converts an analog input (0-5V) into a range of 0 to 1023**.**

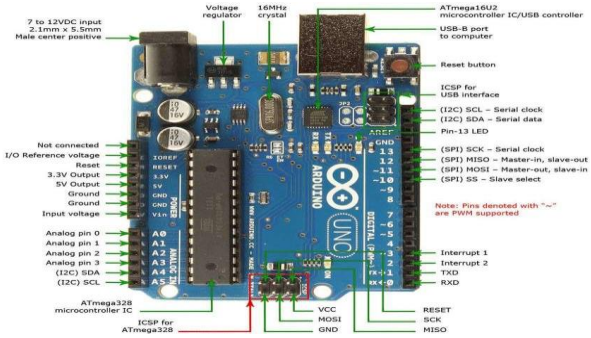


Figure: Pin diagram of Arduino

III. Power Pins

These pins are used to supply power to the board and to external components.

- **VCC (5V):** Provides a regulated **5V** supply for powering the board and external circuits.
- **3.3V:** Provides a **3.3V** supply for components that require a lower voltage.
- **GND (Ground):** There are usually two GND pins that serve as the reference point for the circuit and help complete the circuit by providing the return path for current.
- **VIN:** This pin allows you to supply an external voltage (typically 7-12V) to the Arduino board. The onboard voltage regulator steps this down to the 5V required to power the board.

IV. Reset Pin

- **Function:** The Reset pin is used to reset the Arduino board, causing it to restart the program loaded in its memory.

V. Communication Pins

These pins are used for communication between the Arduino board and other devices (e.g., computers, sensors, or other Arduino boards).

- **TX (Transmit):** This pin is used for serial communication and transmits data from the Arduino to other devices (typically connected to pin 1).
- **RX (Receiver):** This pin is used to receive serial data from other devices (typically connected to **pin 0**).
- **I2C (SDA, SCL)**: I2C is a communication protocol used for connecting multiple devices using only two wires:
  - **SDA** (Serial Data) – used for data transfer.
  - **SCL** (Serial Clock) – used for synchronizing data transfer.
  - On Arduino Uno, SDA is on A4, and SCL is on A5.
- **SPI Pins (MISO, MOSI, SCK, SS)**: The Serial Peripheral Interface (SPI) allows fast communication with other devices.
  - **MISO (Master In Slave Out)**: Used to receive data from the slave device.
  - **MOSI (Master Out Slave In)**: Used to send data to the slave device.
  - **SCK (Serial Clock)**: Used to synchronize the data transfer.
  - **SS (Slave Select)**: Used to select the slave device in SPI communication.

VI. Other Pins

- **AREF (Analog Reference)**: This pin is used to provide an external reference voltage for the analog-to-digital conversion process. This is typically set to 5V **or** 3.3V, but an external voltage reference can be used for more accurate measurements.
- **SDA and SCL**: For I2C communication (Inter-Integrated Circuit). The SDA pin is used for data transfer, and the SCL pin is for clock synchronization. These are often located on analog pins A4 and A5 on the Arduino Uno.

4.2.3 Applications of Arduino

- **IoT Projects**: Arduino can be used to build devices that communicate over the internet, such as weather stations, smart home systems, and remote monitoring systems.
- **Robotics:** Arduino is widely used in robotics projects, where it controls motors, sensors, and actuators to create autonomous robots.
- **Home Automation**: Arduino can control lights, fans, and other appliances, often integrated with sensors to automate the process.
- **Educational Projects**: It is commonly used to teach electronics, programming, and embedded systems to students of all levels.

4.3 Introduction to 16x2 LCD

A **16x2 LCD** (Liquid Crystal Display) is a popular and widely used display module that can show up to 32 characters at a time, with 16 characters per row and 2 rows in total. It's commonly used in various electronics projects, especially in microcontroller-based systems like Arduino and Raspberry Pi. The LCD works by displaying characters using a combination of pixels, which form readable text. It is commonly controlled through a simple interface and is often used in applications like showing sensor data, status messages, or user input. Its versatility, ease of use, and low cost make it a go-to choice for many developers working on embedded systems and DIY electronics projects.

Figure: 16x2 LCD

4.3.1 Pin description

- **VSS (Pin 1)**: Ground pin, connects to the ground of the power supply.
- **VDD (Pin 2)**: Power supply pin, typically connected to +5V.
- **V0 (Pin 3)**: Contrast control. This pin adjusts the contrast of the LCD screen, usually connected to a potentiometer to control the display's visibility.
- **RS (Pin 4)**: Register Select. This pin is used to switch between command mode (RS = 0) and data mode (RS = 1). In command mode, instructions are sent to the display; in data mode, actual characters are sent to be displayed.
- **RW (Pin 5)**: Read/Write. This pin controls whether the LCD will read from or write to the display. It is usually connected to ground (for write mode) when using a microcontroller like Arduino.
- **E (Pin 6)**: Enable. This pin is used to latch data into the display. A pulse on this pin indicates the controller to accept data or commands.
- **D0-D7 (Pins 7-14)**: Data pins. These are the pins that transmit the data to be displayed on the screen. In **8-bit mode**, all pins (D0-D7) are used for data transmission. In **4-bit mode**, only pins D4-D7 are used.
- **LED+ (Pin 15)**: Anode for the backlight. This pin is connected to the positive voltage for the LCD's backlight.
- **LED- (Pin 16)**: Cathode for the backlight. This pin is connected to the ground for the backlight.

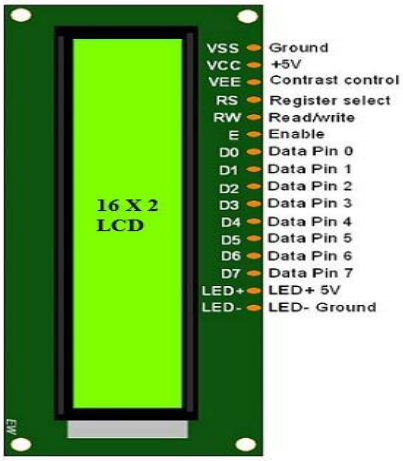


Figure: Pin diagram

4.3.2 Features of LCD

- **Character Display**: Each pixel in the display represents a character. The display can show letters, numbers, and some special symbols.
- **16 Characters per Row, 2 Rows**: The display shows 16 characters on each row and can display 2 rows of characters, totaling 32 characters at once.
- **Backlight**: Many 16x2 LCDs come with a backlight, typically controlled by adjusting the voltage, allowing for visibility in low-light conditions.
- **Standard Interface**: It usually interfaces using **I2C** or **Parallel** (4-bit/8-bit) communication, making it easy to connect with a variety of microcontrollers like Arduino, Raspberry Pi, etc.

4.3.3 How it works

- To send information to the LCD, data or commands are transmitted through the data pins and the control pins (RS, RW, and E).
- **In 4-bit mode**, only the upper 4 bits (D4-D7) are used for communication, which reduces the number of pins required for communication.
- **In 8-bit mode**, all 8 data pins (D0-D7) are used for communication, providing faster data transfer.

4.3.4 Applications of LCD

I. Arduino Projects

- Displaying real-time sensor data (temperature, humidity, pressure, etc.)
- Showing output from user inputs like buttons or potentiometers.
- Providing feedback in robotics projects, such as status or error messages.

II. Home Automation

- Displaying system status, such as lights, temperature, and humidity.
- Controlling home devices like fans, thermostats, or security systems through feedback on the screen.

III. Industrial Equipment

- Showing error codes, machine status, and operational data for operators in factories and production lines.

- Displaying settings, configurations, or calibration data for equipment.

IV. Embedded Systems

- Providing user interface feedback in small, low-power embedded systems.
- Displaying data from sensors, readings, or any other output from a microcontroller in embedded applications.

4.4 Introduction to fingerprint sensor

A **fingerprint sensor** is a biometric device that captures and analyzes a person's fingerprint to identify or authenticate them. It works by scanning the unique patterns of ridges and valleys found in a person's fingerprints, which are distinct to each individual. These sensors are commonly used for security purposes, replacing traditional methods like passwords or PIN codes for access control.

Figure: Fingerprint sensor

4.4.1 Features

- **Optical Sensing**: The R307 uses an optical sensor, which captures a detailed image of the fingerprint and then processes it for matching and identification.
- **High Accuracy**: The R307 provides reliable fingerprint scanning, with a high false rejection rate (FRR) and false acceptance rate (FAR), ensuring secure and accurate identification.

- **Standalone Operation**: The sensor module can function independently, meaning it doesn't require an external controller for fingerprint recognition. It can store and compare fingerprint data directly on the module itself.
- **Storage Capacity**: The R307 module can store up to **1000** fingerprints, allowing it to be used in systems that need to accommodate multiple users.
- **Compact Design**: It has a compact form factor, making it easy to integrate into a variety of devices and systems.
- **Interface**: The R307 typically uses **UART (serial)** communication to interface with microcontrollers like Arduino or Raspberry Pi. It also supports **TTL-level communication**, which is easy to use in embedded systems.

4.4.2 Pin Description

The R307 fingerprint sensor typically has the following pins:

I. **VCC**: Power supply pin (typically 5V).
II. **GND**: Ground pin.
III. **TX**: Transmit pin for communication with the microcontroller (UART).
IV. **RX**: Receive pin for communication with the microcontroller (UART).
V. **IO (optional)**: Interface pin used for specific configuration or communication modes (if applicable).

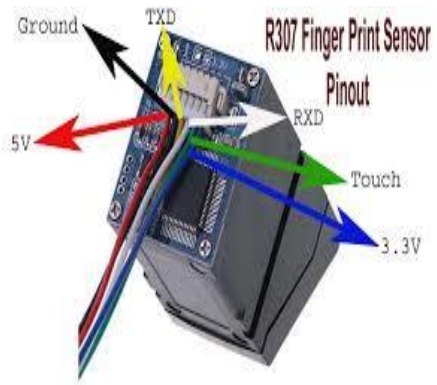


Figure: Pin diagram.

4.4.3 Applications

- **Access Control Systems**: The R307 is commonly used in security systems for granting or denying access to buildings, rooms, or devices based on fingerprint authentication.
- **Time and Attendance Systems**: It's used to track employee attendance by fingerprint recognition.
- **Personal Security Systems**: Can be used in home security systems or for protecting personal devices.
- **Embedded Projects**: Ideal for DIY electronics projects that require fingerprint-based user authentication.

4.5 Buzzer

A **buzzer** is a device commonly used to produce a loud, attention-grabbing sound, often to signal an alert or to indicate that something important or urgent is happening.

Figure: Buzzer

4.5.1 Types of Buzzer.

- **Electromechanical Buzzers**: Use mechanical parts to create sound. Common in older devices and alarms.
- **Piezoelectric Buzzers**: Use piezoelectric materials to generate sound, energy-efficient, and found in modern electronics like microwaves and smoke detectors.
- **Magnetic Buzzers**: Use an electromagnet to vibrate a diaphragm, offering reliable and adjustable sound.
- **Mechanical Buzzers**: Use mechanical components to strike a diaphragm, often producing a distinctive buzzing sound in industrial applications.

- **Active Buzzers**: Have a built-in oscillator to generate sound when power is applied, often used in alarms and clocks.
- **Passive Buzzers**: Require an external frequency signal to produce sound, offering customizable tones.
- **Rotary Buzzers**: Use a rotating mechanism to create continuous sound, used in specific industrial or machinery applications.
- **Digital/Tone Generators**: Use digital signals to generate customizable tones, often found in toys and digital clocks.

4.5.2 Features of buzzer.

- **Sound Generation**: The primary function of a buzzer is to produce sound, typically in the form of a beep or buzzing noise, to alert or attract attention.
- **Power Supply**: Buzzers require an electrical power source to operate, either from direct current (DC) or alternating current (AC), depending on the type of buzzer.
- **Frequency and Tone**: Buzzers can generate different frequencies and tones, ranging from low to high-pitched sounds. Some buzzers allow customization of these tones.
- **Durability**: Buzzers are built to be durable, often made from materials that can withstand frequent use in various environments, such as industrial or household settings.
- **Volume**: Buzzers vary in volume, with some producing loud sounds to be heard over long distances, while others may have a softer, more localized sound.

4.5.3 Applications.

- **Smoke Detectors**: Buzzers alert people to the presence of smoke or fire.
- **Security Systems**: Buzzers signal unauthorized access or security breaches.
- **Kitchen Timers**: Buzzers notify when cooking or baking time is up.
- **Alarm Clocks**: Buzzers wake people up at a set time.
- **Machine Signals**: Buzzers indicate faults or the completion of tasks in industrial equipment.

- **Quiz Shows**: Buzzers are used by contestants to signal answers in competitions.
- **Microwaves**: Buzzers indicate when cooking is complete.
- **Car Systems**: Buzzers alert drivers to seat belt reminders or door issues.
- **Patient Monitors**: Buzzers sound alarms for critical changes in medical devices.
- **Hearing Aids**: Buzzers notify users of low battery levels or settings.

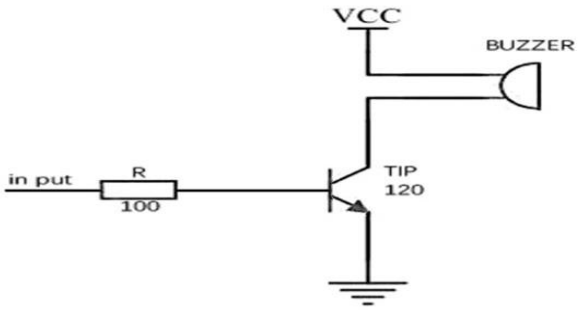


Figure: Circuit of Buzzer

4.6 Jumper wires

Connecting wires are electrical conductors used to establish a pathway for electrical current to flow between different components in a circuit. They are typically made of materials like copper or aluminium due to their excellent conductivity. These wires play a crucial role in ensuring the proper functioning of electrical systems by connecting power sources (like batteries or power supplies) to devices (such as motors, sensors, or buzzers), allowing the transmission of electrical signals or power.

- **Flexible and Easy to Use:** Jumper wires are typically easy to bend and move, making them convenient for quickly modifying circuits.
- **Pre-Striped or Pre-attached Connectors:** Jumper wires usually have connectors on both ends (male or female), which can fit into **breadboards**, **Arduino boards**, and **other components** without needing additional soldering.

- **Short Length:** The wires are typically quite short (often between 5 to 30 cm), making them suitable for compact and temporary connections.
- **Color-coded:** Jumper wires come in different colors to help identify different connections in a circuit, reducing the risk of errors.

4.6.1 Types of jumper wires.

- **Male-to-Male Jumper Wires:**
  - **Description**: These have **male pins** on both ends.
  - **Use**: Often used to connect components on a **breadboard** or between **Arduino** boards and shields.
- **Male-to-Female Jumper Wires:**
  - **Description**: One end has a **male pin**, and the other has a **female socket**.
  - **Use**: Used to connect a male pin (like on a microcontroller) to a female header (on a sensor or another board), or when you need to connect a component with a male pin to a breadboard.
- **Female-to-Female Jumper Wires:**
  - **Description**: Both ends have **female sockets**.
  - **Use**: Typically used to connect two male pins or components that need to be joined together with a female header on both ends.

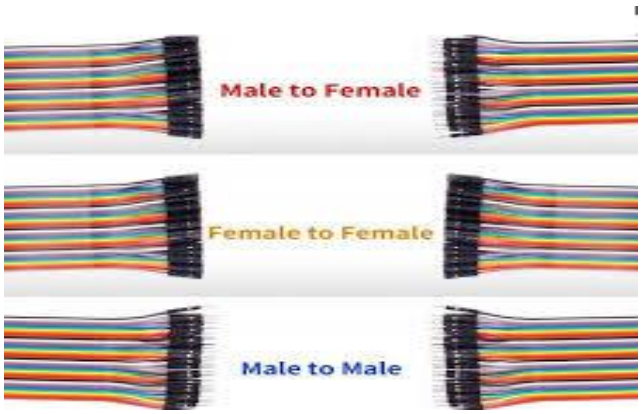


Figure: Jumper wires

4.6.2 Applications.

- **Prototyping:** Used to quickly connect components on a breadboard, allowing for easy circuit design and testing without soldering.
- **Microcontroller Projects:** Essential for connecting **Arduino**, **Raspberry Pi**, or other microcontrollers to sensors, actuators, and other electronic modules.
- **Educational Purposes:** Widely used in schools and universities for hands-on learning in electronics and circuit design.
- **Circuit Troubleshooting:** Used for testing and modifying circuits, as they allow quick changes without permanent connections.
- **Robotics:** Common in connecting motors, sensors, and microcontrollers in robotics projects during the prototyping phase.
- **Embedded Systems:** In development and testing of embedded systems to connect different parts of a system before final assembly.

Chapter 05-WORKING PRINCIPLE

The **fingerprint-based exam hall authentication system** operates using biometric fingerprint recognition to verify the identity of students entering an exam hall. Below is a step-by-step breakdown of how the system works.

1. Student Enrolment

- **Fingerprint Capture:** When a student registers for the exam, their fingerprint is captured using a **fingerprint scanner**. This scanner captures the unique ridges and patterns in the student's fingerprint.
- **Data Processing:** The fingerprint data is processed and converted into a digital template (a mathematical representation of the fingerprint). This template, which is unique to the individual, is then stored in a secure database along with the student's other details (e.g., name, student ID, course).
- **Secure Storage:** The fingerprint data is encrypted to protect student privacy and ensure security. Only authorized personnel have access to this data.

2. Entry to Exam Hall

- **Fingerprint Scanning at Entry:** On the day of the exam, as the student arrives at the exam hall, they are directed to a **fingerprint scanner** placed at the entrance.
- **Fingerprint Enrolment Check:** The student places their finger on the fingerprint scanner. The scanner reads the fingerprint and converts it into a digital template for comparison.

3. Verification Process

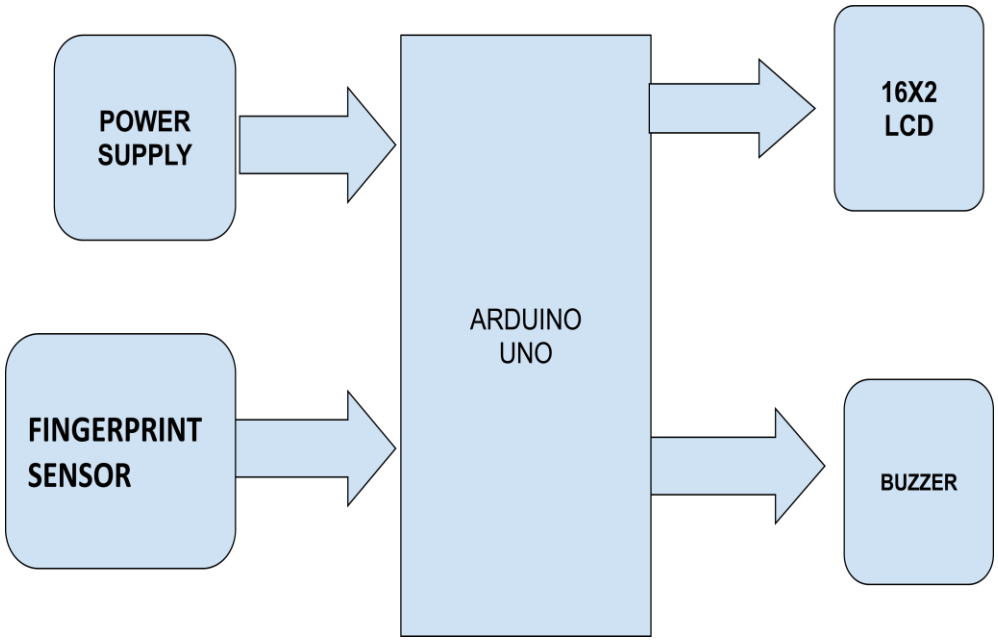
- **Matching with Stored Template:** The system compares the scanned fingerprint with the template stored in the database. The comparison looks for **minutiae points** (unique features like ridge endings, bifurcations, etc.) to find an exact match.
- **Authentication Decision:**

- **Match Found:** If the fingerprint matches a record in the database, the system confirms the student's identity, grants access, and logs the time of entry.
- **No Match:** If the fingerprint does not match any record, the student is either denied entry or prompted for assistance (e.g., a manual check can be done by staff if there's an issue).

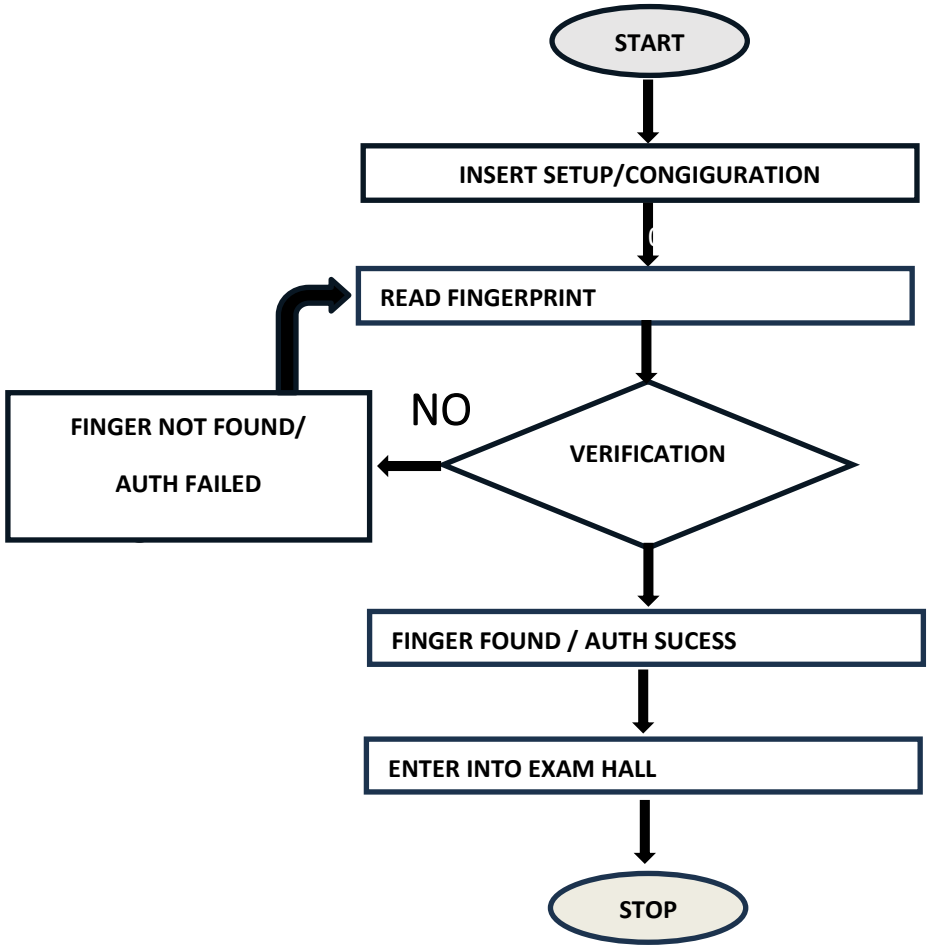
System Flow Summary:

i. **Enrolment:** Capture and store fingerprint data of the student.
ii. **Entry Verification:** Scan the student's fingerprint upon arrival at the exam hall.
iii. **Matching:** The scanned fingerprint is compared with the stored database.
iv. **Authentication:** If the fingerprint matches, the student is granted access and logged. If not, further verification occurs.

5.2 Block Diagram



5.2 Flow Chart



5.3 Schematic Diagram

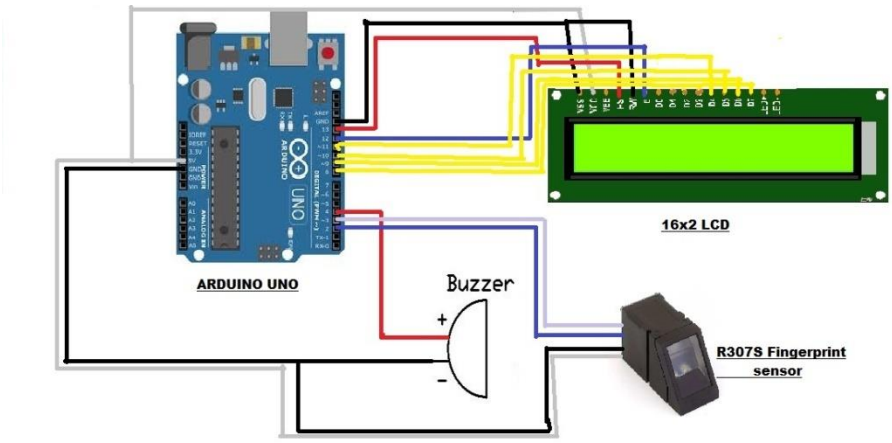


Figure: Schematic Diagram

Chapter 06- Software Used

6.1 Arduino IDE

The **Arduino IDE (Integrated Development Environment)** is the official software used to write and upload programs (called "sketches") to Arduino boards. It is an open-source application, available for **Windows, macOS, and Linux**, and is designed to make programming microcontrollers like the Arduino Uno accessible and straightforward, even for beginners.

What is Arduino IDE?

The Arduino IDE is where you write and edit code (called **sketches**) for your Arduino projects. It provides an interface for:

- Writing code.
- Compiling the code to check for errors.
- Uploading the code to the Arduino board.
- Communicating with the Arduino board during the execution of the code.

Key Components of the Arduino IDE:

- **Editor Window**: This is where you write your code. It includes syntax highlighting and error detection.
- **Toolbar**: Contains buttons for basic operations like uploading the code, verifying (compiling) the code, and selecting the correct board and port.
- **Message Area**: Displays status messages like errors or successful uploads.
- **Console**: Shows logs, warnings, and error messages.

Arduino IDE Structure:

Arduino code is typically organized into two main parts:

- **Setup ()**: A function that runs once when the program starts. It's used to initialize settings (e.g., pin modes).

- **Loop ()**: A function that runs continuously after setup(). This is where the main functionality of your program is written (e.g., reading sensors or controlling actuators).

Basic Workflow in Arduino IDE:

1. **Write Code**: Open the IDE and write your program in the editor window. A basic program might blink an LED on the Arduino board.
2. **Verify/Compile**: Click the checkmark icon to ensure the code has no errors and can be compiled.
3. **Upload Code**: Once compiled successfully, click the right arrow icon to upload the program to the Arduino board.
4. **Monitor**: You can use the Serial Monitor in the IDE to display data sent from the Arduino to your computer (such as sensor values).

Programming Language:

The Arduino IDE uses **C++**, but with simplified syntax and additional functions to make programming easier for beginners. It has pre-defined functions like digitalWrite (), analogRead (), etc., that interact directly with the hardware.

Installing and Setting Up the Arduino IDE:

- **Download**: You can download the Arduino IDE for free from the official website: https://www.arduino.cc/en/software.
- **Install**: Follow the installation instructions for your operating system (Windows, macOS, Linux).
- **Board and Port Selection**: In the IDE, you'll need to select the Arduino board model you're using and the port the board is connected to.

Additional Features of Arduino IDE:

- **Libraries**: You can include libraries to simplify coding for sensors, motors, and other hardware components.
- **Serial Monitor**: Useful for debugging and printing data from your Arduino to the computer.

Figure: Arduino IDE

Chapter 07- Expected results

Successful authentication:

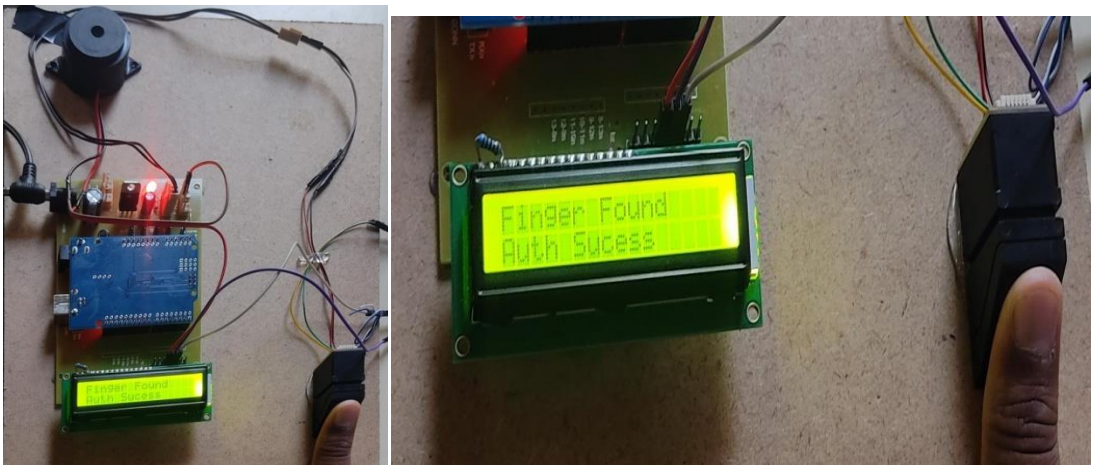


Figure: Successful Authentication o/p

Failed Authentication:

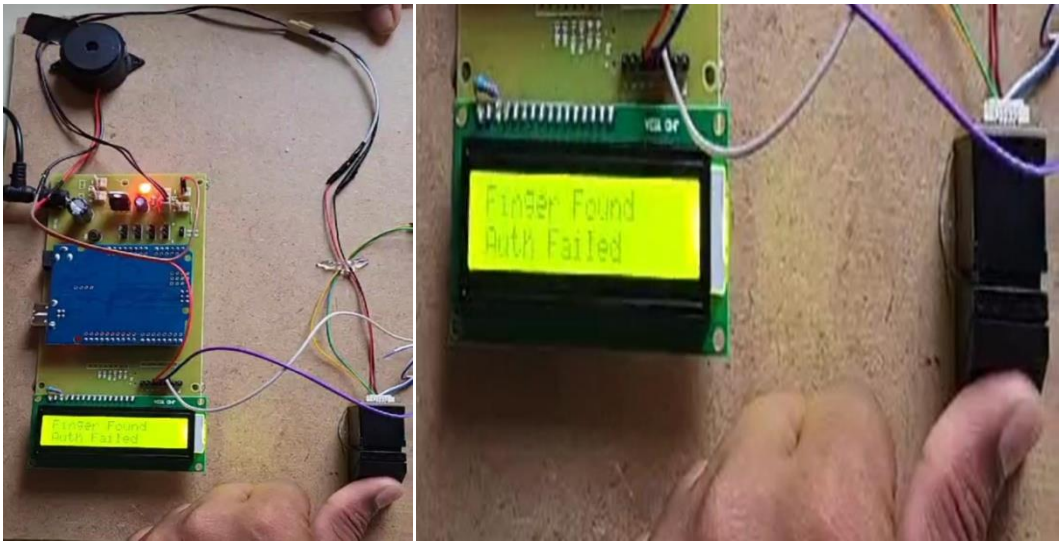


Figure: Failed Authentication o/p

Chapter 07-Advantages and Applications

7.1 Advantages

Fingerprint-based authentication for exam halls offers several significant advantages, providing a secure and efficient way to verify the identity of candidates. Below are the key benefits of using fingerprint-based authentication in exam halls:

- **Enhanced Security**: Prevents impersonation and fraud by ensuring only the registered candidate can attend the exam,
- **Accuracy**: Provides reliable, error-free identification since fingerprints are unique to each individual.
- **Faster Check-In**: Speeds up the verification process, reducing wait times and administrative workload.
- **Cost-Effective**: Reduces the need for manual checks, paperwork, and administrative resources.
- **Prevents Proxy Exams**: Eliminates the possibility of proxy exam-taking by verifying the identity of each candidate.
- **Data Integrity**: Ensures secure, tamper-proof attendance records.
- **Improved Candidate Experience**: Simplifies the process, as candidates only need to scan their fingerprints**.**

7.2 Applications

- E**xam Halls**: Verifies candidates' identity and tracks attendance during exams. **Access Control**: Secures entry to buildings and restricted areas.
- **Banking**: Used for ATM transactions and mobile banking app security.
- **Smart Devices**: Unlocks smartphones, tablets, and laptops.
- **Healthcare**: Ensures accurate patient identification and secure access to medical records.
- **Government**: Used in criminal identification, national IDs, and immigration controls.
- **Workplace**: Tracks employee attendance and integrates with payroll systems

Chapter 08-Conclusion and Future Scope

8.1 Conclusion

In conclusion, fingerprint-based exam hall authentication offers a promising solution to enhance the security and integrity of examination processes. By leveraging the unique nature of fingerprints, this method can significantly reduce instances of impersonation and ensure that only authorized candidates participate. The speed and accuracy of fingerprint scanning make the process more efficient compared to traditional verification methods, improving overall exam management. However, challenges such as potential false matches, privacy concerns, and the costs of implementation must be addressed to ensure its effectiveness. Despite these hurdles, fingerprint authentication stands as a valuable tool for maintaining fairness and security in exams, making it a viable option for modern educational institutions.

8.2 Future Scope.

The future scope of fingerprint-based exam hall authentication includes the following potential developments:

1. **Integration with Other Biometric Systems**: Combining fingerprint authentication with other biometric methods like facial recognition or iris scanning for enhanced security and accuracy.
2. **Cloud-Based Systems and Data Security**: Secure storage and processing of biometric data on cloud platforms, enabling centralized monitoring and real-time analytics of exam security across multiple locations.
3. **Automated Monitoring and AI Integration**: Integration of artificial intelligence to detect suspicious behaviors, anomalies, or discrepancies in real-time, adding an extra layer of security during exams.
4. **Increased Accessibility and Affordability**: As biometric technology becomes more affordable, fingerprint authentication could be adopted by institutions globally, including those in developing regions, enhancing accessibility.

REFERENCE:

1. **Fingerprint Authentication Basics**

- **"Handbook of Fingerprint Recognition"** by Maltoni et al.
  - Covers fingerprint recognition principles and techniques.
- **"Introduction to Biometrics"** by Jain et al.
  - A great starting point for understanding biometric systems, including fingerprints.

2. Fingerprint Matching Algorithms

- **"Fingerprint Matching Using a Hybrid Scheme"** by Lee and Hsu.
  - Discusses fingerprint matching techniques for enhanced security.

3. Biometric Authentication for Exams

- **"Biometric Authentication in e-Exams: A Review"** by Ekinci and Kose.
  - Focuses on using biometric systems (like fingerprints) to secure exams.

4. Practical Systems

- **Neurotechnology VeriFinger SDK** and **MorphoSmart SDK**.
  - Software tools to implement fingerprint recognition in your project.

APPENDIX:

Enrolment Code

```
#include <Adafruit_Fingerprint.h>
#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 (2, 3);
#else
// On Leonardo/M0/etc, others with hardware serial, use hardware serial!
// #0 is green wire, #1 is white
#define mySerial Serial1
#endif
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
uint8_t id;
void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

  // set the data rate for the sensor serial port
  finger.begin(57600);
 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);
}
```

```
uint8_t readnumber(void) {
  uint8_t num = 0;

  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}
void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println("Please type in the ID # (from 1 to 127) you want to save this finger as...");
  id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (! getFingerprintEnroll() );
}
uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  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");
    break;
  }
}

// OK success!

p = finger.image2Tz(1);
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;
}
Serial.println("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");
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");
    break;
  }
}

// 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!");
 } 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;
 }

 return true;
}
```

Authentication code

```
#include <Adafruit_Fingerprint.h>
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin
```

```
// with the arduino pin number it is connected to
const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
SoftwareSerial mySerial(2, 3);
#define buzzer 4
int prevLcdFlag = 0;
int flag = 4;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
void beep(int x = 1) {
  for (uint8_t i = 0; i < x; i++) {
    digitalWrite(buzzer, HIGH);
    delay(1000);
    digitalWrite(buzzer, LOW);
    delay(500);
  }
}
void setup() {
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  beep();
  while (!Serial)
    ;  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Initializing Sensor!");
  // 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); }
  }
  lcd.setCursor(0, 1);
  lcd.print("Sensor Found");
  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");
  }
  // updateLcdData("sadfdsf");
}
void updateLcdData(String text) {
  lcd.clear();
  lcd.print(text);
  delay(2000);
}
void updateLcd(int x) {
  lcd.clear();
  delay(100);
  if (x == 1) {
    lcd.print("Finger Found");
    lcd.setCursor(0, 1);
    lcd.print("Auth Sucess");
    beep();
    delay(2000);
  } else if (x == 2) {
    lcd.print("Finger Found");
    lcd.setCursor(0, 1);
    lcd.print("Auth Failed");
    beep(3);
```

```
    delay(2000);
  } else if (x == 3) {
    lcd.print("communication Error ");

    beep(3);
  } else if (x == 4) {
    lcd.print("Please Place Finger");

    // beep(3);
  }
  delay(1000);
}
void loop()  // run over and over again
{
  getFingerprintID();
  updateLcd(flag);
  // if (flag == 1) {

  //   updateLcd(1);

  // } else if (flag == 2) {
  //   lcd.print("Finger Found");
  //   lcd.setCursor(0, 1);
  //   lcd.print("Auth Failed");
  //   // updateLcd(2);
  // } else {
  //   updateLcd(3);
  // }
  delay(50);  //don't ned to run this at full speed.
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
      flag = 4;
      // updateLcdData("No finger detected");
      return p;
```

```
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK success!

p = finger.image2Tz();
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");
    flag = 3;
    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;
    flag = 0;
}

// OK converted!
p = finger.fingerSearch();
if (p == FINGERPRINT_OK) {
  flag = 1;
  Serial.println("Found a print match!");
```

```
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    // flag = 3;
    Serial.println("Communication error");
    updateLcdData("Communication error");

    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    flag = 2;
    Serial.println("Did not find a match");
    return p;
  } else {
    Serial.println("Unknown error");
    // flag = 4;
    return p;
  }

  // found a match!
  Serial.print("Found ID #");
  Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
  Serial.println(finger.confidence);

  return finger.fingerID;
}

// returns -1 if failed, otherwise returns ID #
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) return -1;

  // found a match!
  Serial.print("Found ID #");
  Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
  Serial.println(finger.confidence);
  return finger.fingerID;
}
```