IOT-ENABLED BIOMETRIC AND OTP-BASED SMART VAULT SECURITY SYSTEM

A MINI PROJECT REPORT

Submitted by

SUGANTH S N	**[6176AC22UEC140]**
THAMEEN ANSARI A	**[6176AC22UEC150]**
VIDYADHEESHA M PANDURANGI	**[6176AC22UEC162]**
YESHWANTH R	**[6176AC22UEC170]**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

ADHIYAMAAN COLLEGE OF ENGINEERING [AUTONOMOUS]

Dr. M.G.R NAGAR, HOSUR - 635 109, TAMILNADU

ANNA UNIVERSITY :: CHENNAI 600 025

NOVEMBER 2025

IOT-ENABLED BIOMETRIC AND OTP-BASED SMART VAULT SECURITY SYSTEM

A MINI PROJECT REPORT

Submitted by

SUGANTH S N [6176AC22UEC140]

THAMEEN ANSARI A [6176AC22UEC150]

VIDYADHEESHA M PANDURANGI [6176AC22UEC162]

YESHWANTH R [6176AC22UEC170]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

ADHIYAMAAN COLLEGE OF ENGINEERING [AUTONOMOUS]

Dr. M.G.R NAGAR, HOSUR - 635 109, TAMILNADU

ANNA UNIVERSITY :: CHENNAI 600 025

NOVEMBER 2025

ANNA UNIVERSITY:: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled "**IoT-Enabled Biometric and OTP based Smart Vault Security System**" is the bonafide work of **Suganth S N, Thameen Ansari A, Vidyadheesha M Pandurangi, Yeshwanth R** who carried out the research under my supervision.

SIGNATURE OF SUPERVISOR
Dr. S. SUMATHI., M.E., Ph.D.,
Professor & HOD of ECE Department,
Adhiyamaan College of Engineering
Dr. M. G. R Nagar
Krishnagiri (Dt.)
Hosur - 635109
Tamilnadu

SIGNATURE OF HOD
Dr. S. SUMATHI., M.E., Ph.D.,
Professor & HOD of ECE Department,
Adhiyamaan College of Engineering,
Dr. M. G. R Nagar
Krishnagiri (Dt.)
Hosur - 635109
Tamilnadu

Submitted for the Mini Project Viva-Voce Examination held on _____at
ADHIYAMAAN COLLEGE OF ENGINEERING, HOSUR - 635 109.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

ABSTRACT

In an era where digital intrusions and unauthorized access incidents are rapidly increasing, traditional password-based vault systems are no longer sufficient to ensure security.

This paper presents the design and development of an IoT-enabled smart vault system that utilizes dual-factor authentication based on biometric fingerprint verification and One-Time Password (OTP) validation. The system employs an ESP32 microcontroller for processing and communication, integrating an R307S Optical Fingerprint Sensor, HC-SR04 Ultrasonic Sensor, and a 12V DC Solenoid Lock. When the user approaches the vault, proximity detection activates the authentication sequence.

After successful fingerprint verification, an OTP is generated and sent to the registered mobile number using the Twilio API over Wi-Fi. The vault is unlocked only when both authentication steps are successfully validated. The proposed design enhances the reliability, scalability, and security of vault systems, making it a cost-effective solution for residential, commercial, and institutional applications.

Keywords: IoT, Biometric Authentication, ESP32, OTP Verification, Smart Vault, Security System.

ACKNOWLEDGEMENT

We wish to express my heartfelt gratitude to the department of Electronics and Communication Engineering, Adhiyamaan college of Engineering for the continues support, in technical expertise to excel to this technology world.

We take immense pleasure in thanking my honorable. beloved and respected **PRINCIPAL Dr. R. RADHAKRISHNAN, M.E., Ph.D.,** for setting up an excellent atmosphere in this institution.

We wish to express the deepest gratitude to the Head of the Department. Electronics and Communication Engineering, **Dr. S. SUMATHl, M.E., Ph.D.,** who has been inspirational and supportive throughout my project.

We would like to thank our project supervisor, **Dr. S. SUMATHl, M.E., Ph.D.**, from the bottom of our heart for her continuous guidance for the completion of the main project.

We also thank all the Teaching and Non-teaching staff who supported us in many aspects for the completion of the project.

We are also deeply indebted to our family members and friends for their co operation and suggestions towards the successful completion of the project

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	Internet of Things
OTP	One-Time Password
LCD	Liquid Crystal Display
ESP32	Espressif Systems 32-bit Microcontroller
R307	Optical Fingerprint Sensor Module
Wi-Fi	Wireless Fidelity
GSM	Global System for Mobile Communication
UART	Universal Asynchronous Receiver and Transmitter
I²C / I2C	Inter-Integrated Circuit
API	Application Programming Interface
HTTPS	HyperText Transfer Protocol Secure
SMS	Short Message Service
DC	Direct Current

LED	Light Emitting Diode
PCB	Printed Circuit Board
TX	Transmit Line
RX	Receive Line
VCC	Voltage Common Collector (Positive Supply)
GND	Ground
HC-SR04	Ultrasonic Distance Sensor Model
MCU	Microcontroller Unit
OTP Gen	One-Time Password Generation Algorithm
Auth	Authentication
ID	Identification Number
UI	User Interface
OSI	Open Systems Interconnection (Reference Model)

CHAPTER 1

INTRODUCTION

In today's digital world, traditional security systems such as mechanical locks, RFID cards, and password-based access are increasingly vulnerable to hacking and intrusion techniques. As a result, there is a growing need for advanced multi-layered security solutions that ensure reliable protection against unauthorized access.

To overcome these limitations, this project proposes an IoT-enabled Smart Vault Security System that integrates **biometric authentication** and **One-Time Password (OTP) verification** for dual-factor access control. A fingerprint sensor serves as the primary authentication, while a secure OTP is generated and sent to the registered mobile number through the Twilio API for secondary verification. The ESP32 controller handles decision-making, cloud connectivity, and user interaction through an LCD display, whereas a solenoid lock mechanism ensures secure physical locking control.

This smart vault solution enhances user safety by combining identity verification and remote authentication features. It is highly suitable for high-security applications including bank lockers, confidential document storage, home safes, and corporate vaults. By integrating IoT with dual-factor authentication, the system provides a modern, user-friendly, and tamper-resistant vault security solution.

CHAPT ER 2

LITERATURE SURVEY

In[1], The study presented by Joongjin Kook (2019) focuses on an IoT-enabled digital door lock system that incorporates OTP-based authentication to improve overall access security. The author emphasizes that integrating smartphones with dynamic OTP generation significantly minimizes unauthorized entry, outperforming conventional mechanical key and static password methods. This research highlights the growing requirement for secure, smart authentication techniques in IoT-based environments.

In[2], A comprehensive review by W. Yang et al. (2021) discusses multiple biometric authentication technologies used in IoT security. The research identifies vulnerabilities associated with single-factor authentication and concludes that combining biometrics with another verification factor strengthens data protection. This supports the adoption of multi-factor schemes like fingerprint and OTP-based verification in secure access systems.

In[3], A dual authentication system combining OTP and fingerprint verification is proposed by S. R. Patel and G. Mehta (2022). The study demonstrates that integrating fingerprint scanning with a time-bound OTP mechanism substantially reduces spoofing attempts and enhances protection. The experimental outcomes validate that biometric-OTP hybrid authentication is a highly secure technique for access control applications.

In[4], M. K. Hussain and A. Kumar (2022) introduce a smart IoT-based locker utilizing both fingerprint identification and OTP for user authentication. The design incorporates remote monitoring through Wi-Fi and ensures high-level security for valuable storage. Their study reinforces the efficiency of IoT-based smart vaults for security-critical applications.

In[5], The work by R. Paneru et al. (2023) proposes a secure locker system combining RFID, biometric recognition, and OTP verification. Through testing, the authors confirmed enhanced resistance against tampering and duplication attacks. Their results justify the adoption of OTP as an additional protection layer in smart locker solutions.

In[6], A smart vault system based on ESP32 was developed by K. L. Deshmukh and R. Patil (2023). The system offers improved wireless performance and robust data encryption compared to GSM-based solutions. Their findings establish that ESP32 provides higher operational reliability, faster authentication, and better cyber-security, making it suitable for modern smart vault systems.

In[7], Research conducted by A. S. Ma et al. (2024) evaluates fingerprint recognition as a primary identity verification method in IoT security systems. The paper concludes that biometric authentication significantly minimizes unauthorized access attempts and identity fraud, supporting the adoption of fingerprint-based security in sensitive storage systems.

In[8], A security-enhanced biometric approach is demonstrated by M. Nasr et al. (2025), where cancelable fingerprint templates are generated using EMD and quaternion representations to prevent biometric data theft. The system achieves excellent validation performance (AUC = 0.9997), ensuring safe fingerprint encryption and storage in IoT systems.

In[9], A. Alotaibi (2025) presents a detailed review on IoT authentication mechanisms and major cyber-security vulnerabilities. The author stresses the increasing demand for multi-factor authentication due to rising cyber-attacks in connected systems. This supports the adoption of fingerprint combined with OTP for high-security IoT vaults.

In [10], Finally, N. Singh and P. Agarwal (2025) propose a smart locker system that uses RFID, keypad input, and GSM-based alert notifications. However, the authors highlight limitations like SMS delays and weaker encryption standards, suggesting Wi-Fi-based systems as a more secure and efficient alternative — aligning with the use of ESP32 in the current proposed project.

CHAPTER 3

EXISTING METHOD

Existing vault and secure access systems have evolved significantly over the years, moving from basic mechanical locking techniques to electronic and biometric-based security solutions. This section provides an overview of the prevailing methodologies, highlighting their working principles, strengths, and limitations.

A. Traditional Mechanical Locking Systems

Mechanical locking methods using keys, padlocks, or combination locks have been the most common approach for securing valuables.

Working Principle: Security is achieved through physical keys or dial-based combinations.

Limitations:

- Keys can be lost, duplicated, or stolen.
- No access logs or traceability.
- Vulnerable to lock-picking, drilling, or physical tampering.

B. Password or PIN-Based Electronic Locks

These systems introduced digital input methods such as keypads to enter a PIN for unlocking.

Working Principle: Users enter a pre-set password via a keypad to gain access.

Limitations:

- Weak passwords are vulnerable to guessing or shoulder-surfing.
- Frequent reuse of passwords decreases security.
- No second-layer authentication to verify the actual user.

C. RFID-Based Authentication Systems

Radio Frequency Identification (RFID) cards and tags have become a popular

substitute for keys due to their convenience.
Working Principle: The system unlocks when a registered RFID card or tag is scanned.
Limitations:

- RFID tags can be cloned using low-cost devices.
- Cards can be misplaced or borrowed for unauthorized access.
- No real-time validation to ensure ownership.

D. Biometric-Only Authentication Systems

Biometric systems, especially fingerprint recognition, have improved security by verifying the user's unique physiological traits.
Working Principle: A fingerprint scanner captures and matches the user's fingerprint against stored templates.
Limitations:

- Sensor spoofing is possible using high-quality replicas.
- Single-layer authentication — if biometric data is compromised, there is no secondary validation.
- Lacks IoT or real-time verification for remote alerts or monitoring.

E. IoT-Enabled Smart Access Systems

Recent systems integrate wireless connectivity (Wi-Fi, GSM, or Bluetooth) to enable remote access and alerts.
Working Principle: Systems use IoT modules to communicate with mobile apps or cloud platforms to provide unlock commands, alerts, or monitoring.
Limitations:

- Many solutions rely solely on passwords or OTP via GSM, resulting in higher network costs.
- Mechanical Locks MediumLimited multi-factor authentication and security audit features.

Summary of Gaps in Existing Systems

System Type	Security Level	User Authentication Layer	Major Drawback
Mechanical Locks	Low	Physical Key	Easy to tamper or duplicate
Password PIN Locks	Medium	Knowledge-based	Password guessable or shareable
RFID Systems	Medium	Token-based	Card cloning & misplacement
Biometric Systems	High	Physiological-based	No second verification
IoT Smart Systems	High	Remote Access Enabled	Costly GSM dependency & delays

CHAPTER 4

PROPOSED METHOD

The proposed Smart Vault Security System implements a highly secured **dual-factor authentication mechanism** using fingerprint recognition and OTP verification. The ESP32 microcontroller functions as the core controller, managing biometric authentication through the R307 fingerprint sensor and initiating OTP generation via the Twilio cloud service. A 16×2 LCD display provides user guidance and real-time status updates throughout the process.

Upon a successful fingerprint match, the user receives a unique OTP on their registered mobile number, which must be entered for secondary authentication. If the OTP verification is valid, the ESP32 triggers a relay to activate a 12V solenoid lock and grant authorized access. This integrated hardware–software security approach ensures strong protection against unauthorized entry, making the system suitable for secure vaults in homes, banks, and corporate environments.

4.1 Advantages of Proposed Method

- ✓ Provides enhanced security through dual-factor authentication (Fingerprint + OTP).
- ✓ Enables real-time remote verification using cloud-based OTP delivery.
- ✓ Ensures high resistance to hacking, duplication, and unauthorized access.
- ✓ Offers a low-cost and scalable solution using ESP32 with inbuilt Wi-Fi.
- ✓ Delivers easy and user-friendly operation with LCD feedback and automated access control.

CHAPTER 5

BLOCK DIAGRAM

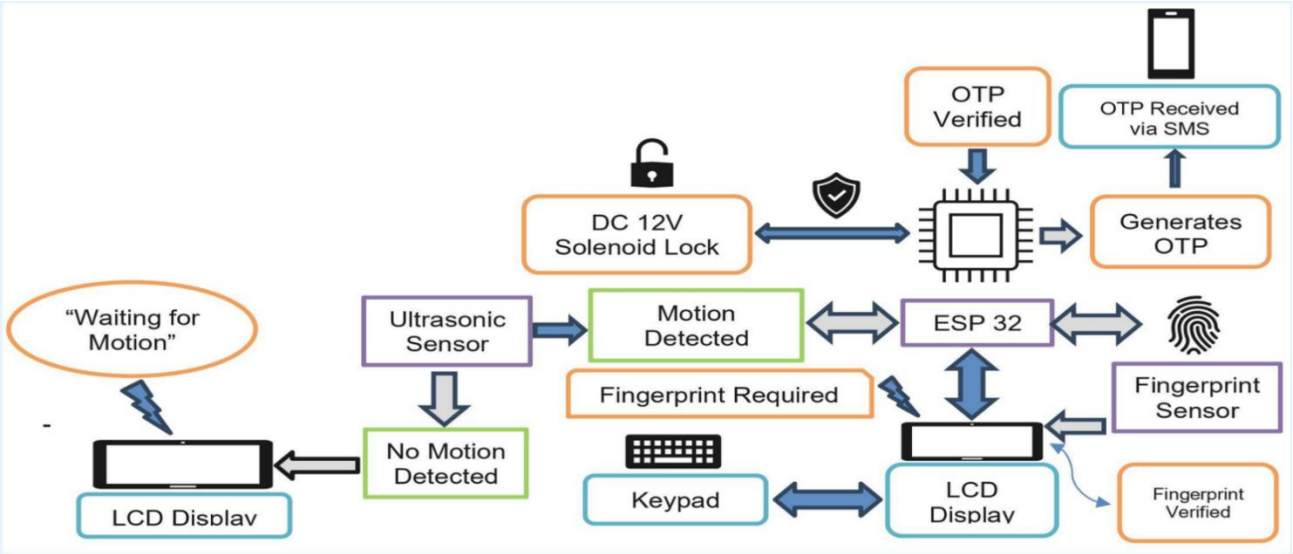

Figure 5.1 Block Diagram

The block diagram illustrates the functional flow and interaction between various hardware components used in the smart vault security system.

1. **Ultrasonic Sensor**
    - Continuously monitors the presence or movement of a user in front of the vault.
    - When motion is detected within a predefined distance threshold, it sends a trigger signal to the ESP32 microcontroller.
    - If no motion is detected, the LCD displays **"Waiting for Motion"**, keeping the system in idle mode.
2. **ESP32 Microcontroller**
    - Acts as the central processing unit of the security system.
    - Receives motion triggers, fingerprint responses, and OTP input from the keypad.
    - Controls the entire authentication flow by coordinating all the connected components.

- Manages Wi-Fi connectivity to request and verify OTP through the cloud.

3. **LCD Display (16×2 with I2C)**
   - Provides real-time user guidance such as:
     - "Waiting for Motion"
     - "Fingerprint Required"
     - "Enter OTP"
     - "Access Granted" / "Access Denied"
   - Ensures user-friendly Human-Machine Interaction (HMI).
4. **Fingerprint Sensor (R307)**
   - Performs biometric authentication as the **first security layer**.
   - Sends fingerprint data to ESP32 for matching with stored templates.
   - If matched → ESP32 initiates OTP generation process.
   - If mismatched → System returns to the initial state for re-authentication.
5. **Keypad (4×4 Matrix)**
   - Used for manual entry of OTP received via SMS.
   - Sends each keypress data to ESP32 for validation against the generated OTP.
6. **OTP Generation and Verification**
   - ESP32 requests OTP using Twilio Cloud API via Wi-Fi.
   - OTP is delivered to the registered mobile number through SMS.
   - User enters the OTP through keypad.
   - ESP32 compares the entered OTP with the system-generated OTP:
     - Match → Authentication success, Mismatch → Access denied
7. **DC 12V Solenoid Lock + Relay Driver**
   - Controlled by ESP32 only when dual authentication is successful.
   - Relay isolates the 12V power supply from the low-voltage control logic.
   - Lock unlocks for a preset time duration and then returns to a locked state automatically.

CHAPTER 6

WORKING

- ✓ When a user approaches the vault, the ultrasonic sensor senses the person and then **system becomes active** and displays authentication instructions on the LCD.
- ✓ The user places their finger on the **R307 fingerprint sensor** for primary biometric verification.
- ✓ The **ESP32 compares** the scanned fingerprint with its stored database of authorized fingerprints.
- ✓ If the fingerprint is **not matched**, access is denied and the system resets.
- ✓ If the fingerprint is **successfully verified**, the ESP32 requests a secondary authentication step.
- ✓ The system communicates with the cloud server via Wi-Fi and **sends a request to Twilio API** to generate an OTP.
- ✓ A unique **OTP is delivered** to the registered user's mobile phone through SMS.
- ✓ The user **enters the OTP** using the keypad for verification.
- ✓ The ESP32 **validates the OTP** against the server-generated code.
- ✓ If the OTP is **correct**, the ESP32 **activates the relay module**, unlocking the solenoid lock to open the vault.
- ✓ If the OTP is **incorrect or expired**, access is denied and the system restarts the authentication process.

6.1 Circuit Diagram

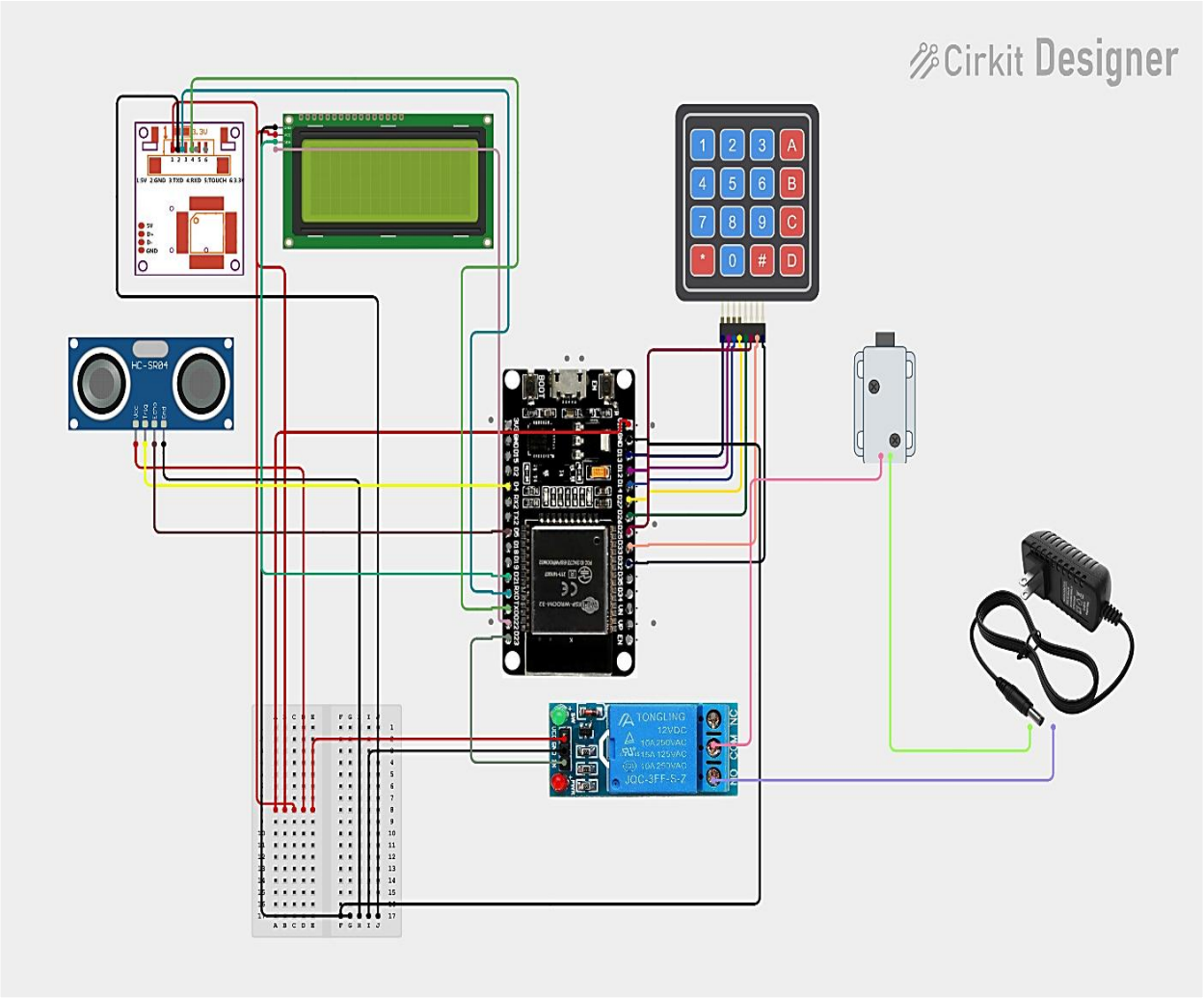

Figure 6.1 Circuit Diagram

CHAPTER 7

COMPONENTS SPECIFICATION

7.1 ESP32 Development Board (30-Pin Version)

The ESP32 is a highly integrated microcontroller developed by Espressif Systems, featuring:

✓ Dual-core Xtensa 32-bit LX6 processor

✓ Built-in Wi-Fi (2.4 GHz) & Bluetooth (v4.2 BLE)

✓ Ultra-low power consumption

✓ Multiple peripherals: UART, SPI, I2C, PWM, ADC, DAC, GPIO

It serves as the **central processing and control unit** for the smart vault. It handles fingerprint verification, OTP generation, data transmission over the internet, and relay activation.

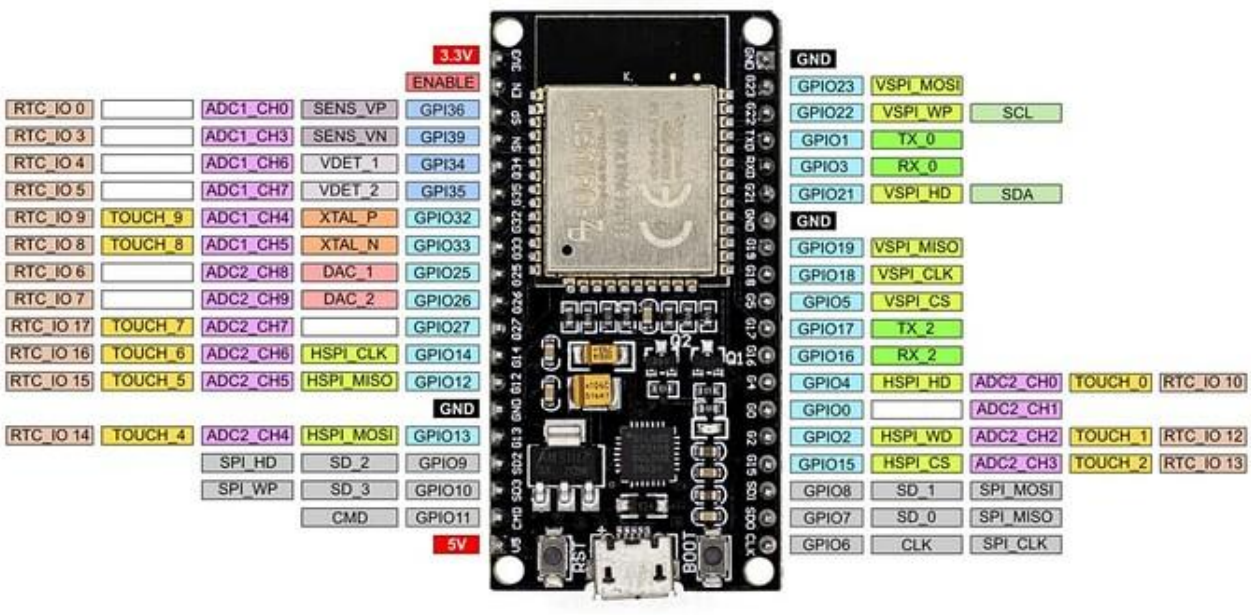

Figure 7.1 ESP32

7.1.1 Features of ESP32

1. Built-in Wi-Fi (802.11 b/g/n) and BLE
2. High processing performance (dual-core)
3. Multiple communication interfaces (UART, I2C, SPI)

7.1.2 ESP32 Pin Architecture

Pin	Type	Purpose
5V	Power	USB-based power input for the board
3V3	Power	Regulated output power to sensors
GND	Ground	Common reference ground
GPIO 16 (RX2)	UART	Receives fingerprint data (from R307 TX)
GPIO 17 (TX2)	UART	Sends commands to R307 fingerprint sensor
GPIO 21 (SDA)	I2C	LCD communication
GPIO 22 (SCL)	I2C	LCD communication
GPIO 13	Digital I/O	Relay lock control
GPIO 32,33	Digital I/O	Keypad row/column mapping

7.1.3 Communication and programming

The ESP32 microcontroller acts as the central brain of the proposed smart vault system. It manages all control and communication processes between the biometric sensor, keypad, display unit, relay module, and cloud-based OTP service. To ensure reliable and seamless operation, the ESP32 is programmed using the Arduino IDE with Wi-Fi and serial communication protocols.

7.1.4 Working Principle

- ✓ Continuously monitors sensors and keypad input
- ✓ Authenticates user identity (biometric + OTP)
- ✓ Connects to internet & Twilio server for OTP delivery
- ✓ Drives relay to lock/unlock solenoid based on verification

7.1.5 Applications

1. IoT automation.
2. Smart security systems.
3. Access control devices.

7.2 R307 Optical Fingerprint Sensor

The R307 is an Optical Fingerprint Reader Sensor Module that combines an optical image collection component and an algorithmic processing chip into a single, compact unit. This "all-in-one" design allows it to handle the entire fingerprint recognition process—from image capture to final matching—without requiring an external Digital Signal Processor (DSP) chip.

Figure 7.2 R307 Optical Fingerprint Sensor

7.2.1 Features of R307 Sensor

- ✓ Integrated Design: Combines the sensor, processor, and algorithms.
- ✓ High Resolution: Typically captures images at a resolution of 500 DPI.
- ✓ Storage Capacity: Can store a significant number of unique fingerprint templates (often up to 1000).
- ✓ Operation Modes: Supports two primary functions:

    1. Comparison (1:1): Verifies a scanned finger against a specific, stored template (e.g., checking an ID number and fingerprint).
    2. Search (1: N): Searches the entire stored database for a match against the scanned finger (e.g., general access control).

- ✓ Interface: Communicates using a TTL UART (serial) interface, making it easy to connect directly to microcontrollers like Arduino, Raspberry Pi, or ESP32. It also often includes a USB1.1 interface for PC connection.
- ✓ Low Power: It is known for its low power consumption, making it suitable for battery-operated devices.

- ✓ Secondary Development: The module is designed for secondary development, meaning it can be easily embedded into various end-products like access control systems, attendance machines, safes, and secure locks.

7.2.2 Pin Description

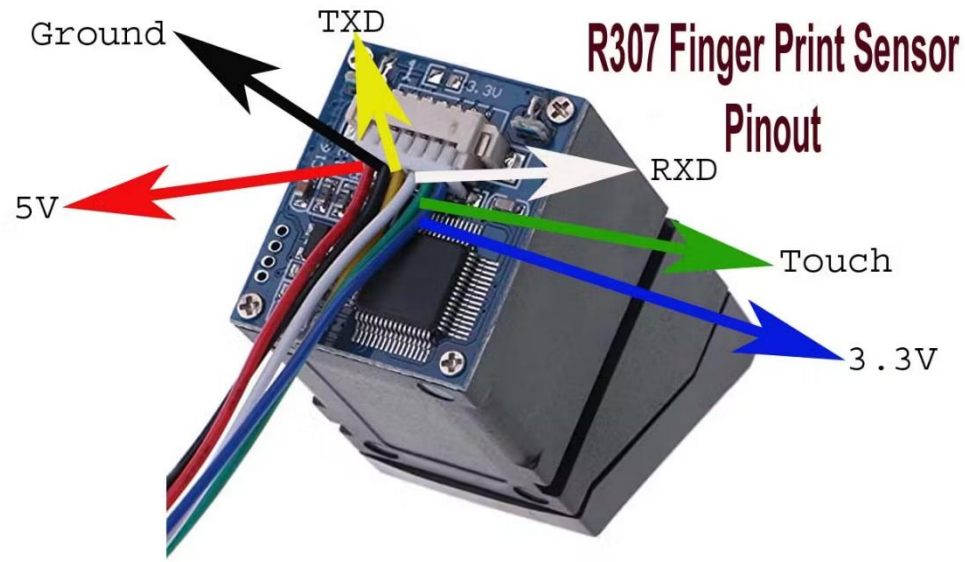

Figure 7.3 R307 Sensor Pin Diagram

7.2.3 Communication and Programming

The R307 Optical Fingerprint Module communicates with the ESP32 microcontroller using a UART (Universal Asynchronous Receiver/Transmitter) serial interface. This communication enables the microcontroller to send fingerprint enrollment or verification commands and receive recognition results.

7.2.4 Internal Storage & Template Handling

Memory Type	Purpose
Flash Memory	Stores fingerprint templates
RAM	Stores temporary fingerprint images

7.2.5 Programming Workflow Using R307

The fingerprint module is controlled using commands through fingerprints library such as:

Adafruit_Fingerprint.h or similar R307 protocol libraries

The workflow includes:

✓ **Enrolment Phase**

1. User requested to place finger on sensor
2. Sensor captures fingerprint image
3. Converts image → Feature template
4. Saves template into Flash memory at a specific ID

✓ **Verification Phase**

1. User places finger again
2. Module compares with stored templates
3. Returns:
   - Success (Matched ID)
   - Fail (Not recognized)

7.2.6 Working Process

- ✓ Captures fingerprint image using optical scanning
- ✓ Converts to a digital template
- ✓ Compares template with internal database
- ✓ Sends result to ESP32

Fingerprint matching modes:

1. 1:1 Verification ⇒ matches against a pre-linked ID
2. 1: N Identification ⇒ matches against stored database

Highly secure because fingerprint replication is extremely difficult.

7.2.7 APLLICATION

1. Secure Access Control Systems
2. Biometric Attendance and Time Tracking
3. Smart IoT-Based Security Devices

7.3 16×2 LCD Display with I2C Interface

A 16×2 LCD is a character display that shows 2 rows × 16 characters (based on the popular HD44780 controller family). An I²C backpack (commonly using a PCF8574 I/O expander) converts the parallel interface into a 2-wire I²C interface (SDA, SCL), drastically reducing wiring to only four wires (VCC, GND, SDA, SCL). This is ideal for microcontrollers with limited GPIOs (like the ESP32).

Figure 7.4 16*2 LCD DISPLAY I2C

7.3.1 Features of 16*2 LCD Display I2C

- ✓ 16 characters per line and 2 display lines (total 32 characters).
- ✓ I2C interface reduces connection from 16 pins to only 4 wires (VCC, GND, SDA, SCL).
- ✓ Low Power Consumption (5V supply).
- ✓ Built-in character generator (CGROM) with ASCII character set.
- ✓ Backlight support for good visibility in low-light environments.
- ✓ Adjustable contrast through potentiometer (on I2C backpack module).
- ✓ Supports display features like cursor blink, character scroll, and custom

characters.

- ✓ Compatible with microcontrollers like ESP32, Arduino, etc.
- ✓ Works using LiquidCrystal_I2C library making programming simple.

7.3.2 Pin Description

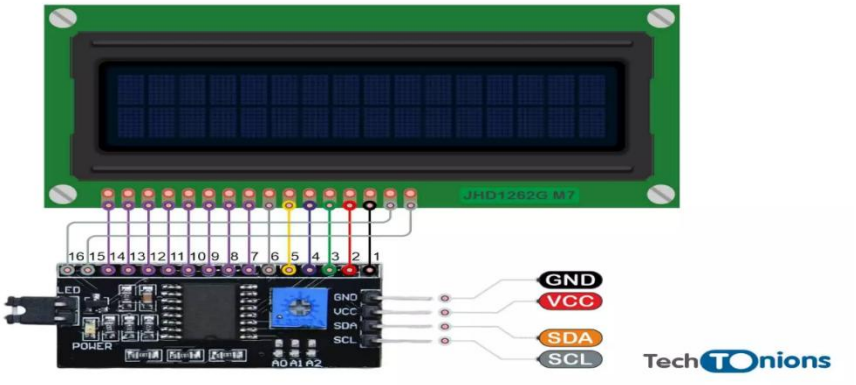

Figure 7.5 LCD I2C Pin Description

LCD VCC → 5V (or 3.3V if module/backpack supports)

LCD GND → GND

LCD SDA → ESP32 GPIO21 (SDA)

LCD SCL → ESP32 GPIO22 (SCL)

Notes:

- ✓ Many backpacks include pull-up resistors on SDA and SCL already. If not, use ~4.7k pull-ups to VCC.
- ✓ Modules are usually powered by 5V. Most PCF8574 and HD44780 tolerate 3.3V I²C logic levels, but check the specific board. If uncertain, use 5V VCC and confirm the backpack's inputs are 3.3V-friendly or use a level shifter.
- ✓ Adjust contrast with the small potentiometer on the backpack.

7.3.3 Communication and Programming

- ✓ Communicates using I²C (Inter-Integrated Circuit) protocol
- ✓ ESP32 acts as Master, LCD acts as I2C Slave
- ✓ Default I2C address is usually 0x27 or 0x3F

- ✓ Data and commands are sent serially over SDA & SCL lines

7.3.4 Programming (ESP32)

- ✓ Include Library - #include <LiquidCrystal_I2C.h>
- ✓ Create LCD object - LiquidCrystal_I2C lcd(0x27, 16, 2);
- ✓ Basic operations - lcd.init();

```
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("Hello");
```

7.3.5 Working Principle

- ✓ The module contains:
    1. LCD panel for character display
    2. HD44780 LCD controller
    3. PCF8574 I/O expander IC for I2C communication

The microcontroller sends 8-bit data serially → PCF8574 converts it into parallel form → LCD controller displays characters accordingly.

- ✓ The data sent can be:
    1. Commands (clear, cursor shift, display ON/OFF)
    2. Characters (ASCII text display)
- ✓ Internal memory used:
    1. DDRAM: Stores display characters
    2. CGROM/CG-RAM: Generates characters including custom fonts

7.3.6 Working Process in the System

Role	Description
Status Display	Shows instructions → “Scan Finger”, “Enter PIN”, “Access Granted/Denied”
User Feedback	Helps user interact with authentication process
Security Alerts	Displays warnings such as intrusion detected
Debugging	Shows sensor/module connectivity status

1. System ON → LCD displays Welcome – To enroll the finger print in the system.
2. Prompts user to Scan Fingerprint
3. On success → asks Enter PIN
4. Displays status → Unlocked or Access Denied
5. Shows Intrusion alert based on ultrasonic readings

7.3.7 Applications

1. Embedded systems display units
2. Digital meters and gauges
3. Industrial automation control panels
4. Security systems (like Smart Vault)
5. Consumer electronics (printers, switches)
6. IoT devices for live data display

7.4 Ultrasonic Sensor (HC-SR04)

The HC-SR04 Ultrasonic Sensor is a popular and cost-effective electronic module used for accurate non-contact distance measurement in embedded and automation systems. It works by transmitting high-frequency ultrasonic sound waves (40 kHz) through the TRIG pin and then receiving the reflected waves using the ECHO pin to calculate the distance of an object based on the time of flight. With a typical detection range from 2 cm to 4 meters, the sensor provides stable and reliable output, making it suitable for applications such as obstacle detection, robotic navigation, water level monitoring, and smart security systems.

Due to its simple interfacing and compatibility with microcontroller platforms like Arduino and ESP32, the HC-SR04 is widely used in academic projects and real-time IoT implementations to enhance environmental awareness and automation capability.

Figure 7.6 Ultrasonic Sensor (HC-SR04)

7.4.1 Features of Ultrasonic Sensor

- ✓ Provides accurate non-contact distance measurement
- ✓ Operating voltage: 5V DC
- ✓ Measurement Range: 2 cm to 400 cm
- ✓ Best accuracy within 30 cm to 300 cm

- ✓ Resolution: 3 mm
- ✓ Ultrasonic frequency: 40 kHz
- ✓ Built-in transmitter and receiver module
- ✓ Low cost, fast response time
- ✓ Widely used in obstacle detection and security systems

6.4.2 Pin Description

Pin Name	Function
VCC	+5V Power Supply
GND	Ground
TRIG	Sends ultrasonic pulses
ECHO	Receives reflected signal (distance measurement)

7.4.3 Communication & Programming

Communication with microcontroller is based on pulse timing.

✓ Working Signal Flow

1. ESP32 sends a 10µs pulse to TRIG
2. Ultrasonic transmitter emits 8 cycles of 40kHz wave
3. If a surface is present, the echo signal is reflected back
4. ECHO pin changes to HIGH for a duration equal to round-trip time

✓ Distance Calculation Formula

$$\boldsymbol{Distance\ (cm) = \frac{Tme(\mu s) * 0.043}{2}}$$

1. Speed of sound = 0.034 cm/µs
2. Divided by 2 → forward + reflected travel.

7.4.4 Working Principle

The sensor uses ultrasonic sound waves to detect objects.

1. TRIG → Sends sound pulse
2. Sound hits an object and returns
3. ECHO → Measures the time taken for return signal

Based on time delay → calculates distance

1. No physical contact required
2. Works even in dark environments (unlike IR sensors)

7.4.5 Working Process in the System

In your **Smart Vault Security System**, ultrasonic sensor plays a major role in:

Purpose	Functionality
Intrusion Detection	Detects forced entry or door opening
Security Alert	Triggers alarm/notification on suspicious motion
Vault Tamper Monitoring	Measures distance changes inside the vault door

1. Vault remains **closed** → stable distance reading
2. Vault remains **closed** → stable distance reading

- ✓ If intruder tries to **open the door** → sudden distance change
- ✓ System activates:
    1. LCD **warning message**
    2. Buzzer / Relay lock reinforcement
    3. Security event logged
- ✓ Enhances the overall **anti-theft protection** of the vault

7.4.6 Applications

1. Automated door control
2. Object presence detection
3. Level measurement (tank, water level monitoring)
4. Robotics for obstacle avoidance

7.5 Single Channel 5V Relay Module

The Single Channel 5V Relay Module is an electromechanical switching device used to control high-voltage or high-current loads using low-power control signals from microcontrollers such as Arduino, ESP32, and Raspberry Pi. It acts as an interface between the low-voltage digital world and high-power electrical devices, enabling safe isolation through its built-in optocoupler and switching mechanism.

When triggered by a 5V input, the relay coil energizes and activates the internal switch, allowing control of appliances like lights, motors, pumps, and fans through automation or remote operation. Due to its ease of use, electrical isolation, and reliable performance, the 5V Single Channel Relay Module is widely used in IoT applications, home automation projects, industrial control systems, and smart security systems.

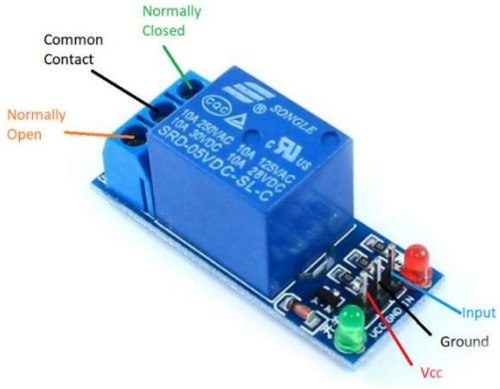

Figure 7.7 Single Channel 5V Relay Module

7.5.1 Features of Single Channel 5V Relay Module

- ✓ Supports AC or DC load switching
- ✓ Control signal: **5V logic**
- ✓ Can switch **high-power devices** like motors, solenoids, lamps, etc.
- ✓ Built-in **flyback diode** for coil protection
- ✓ **Optocoupler isolation** in some relay modules for electrical safety
- ✓ Supports **NO (Normally Open)** and **NC (Normally Closed)** contact.

7.5.2 Pin Description

Relay Pin	Function
VCC	+5V supply to relay coil
GND	Ground connection
IN	Control signal from ESP32
COM	Common terminal → Load connection
NO	Normally open output (active ON switching)
NC	Normally closed output (active OFF switching)

7.5.3 Communication & Programming

Relay works on digital switching:

ESP32 Output	Relay Condition
LOW	Relay OFF → Load disconnected
HIGH	Relay ON → Load Activated

Code Example

```
digitalWrite(RELAY_PIN, HIGH);  // Unlock solenoid
delay(5000);
digitalWrite(RELAY_PIN, LOW);   // Lock again
```

7.5.4 Working Principle

- ✓ ESP32 sends ON/OFF signal to relay IN pin
- ✓ Relay coil energizes → switches mechanical contact
- ✓ Electrical connection changes from NO → COM
- ✓ Load (solenoid) gets power → performs required action

7.5.5 Working Process in System

Relay is used to control access to the Smart Vault:

- ✓ Relay ON → Solenoid lock energized → Vault unlocked
- ✓ Relay OFF → Lock de-energized → Vault secured

Acts as the final execution device for access approval

7.5.6 Applications

1. Smart Home Automation
2. Industrial Control Panels
3. Access Control Systems
4. Motor and Actuator Control
5. High-power device switching in IoT projects

7.6 12V Solenoid Lock

The Single Channel 5V Relay Module is an electromechanical switching device used to control high-voltage or high-current loads using low-power control signals from microcontrollers such as Arduino, ESP32, and Raspberry Pi.

It acts as an interface between the low-voltage digital world and high-power electrical devices, enabling safe isolation through its built-in optocoupler and switching mechanism. When triggered by a 5V input, the relay coil energizes and activates the internal switch, allowing control of appliances like lights, motors, pumps, and fans through automation or remote operation. Due to its ease of use, electrical isolation, and reliable performance, the 5V Single Channel Relay Module is widely used in IoT applications, home automation projects, industrial control systems, and smart security systems.

Figure 7.8 12V Solenoid Lock

7.6.1 Features of 12V Solenoid Lock

- ✓ High-security **electromagnetic locking mechanism**
- ✓ Operating Voltage: **12V DC**
- ✓ High mechanical strength and durability
- ✓ Low idle current consumption
- ✓ Locking method: **spring-loaded plunger**
- ✓ Compact and easy to mount inside vault/door systems

7.6.2 Pin Description

Terminal	Function
+12V	Power input (Positive)
GND	Power input (Negative through relay)

Solenoid **must not** be connected directly to ESP32 → Requires Relay module for isolation & current support

7.6.3 Working Principle

- ✓ When **no power**, the plunger **remains extended** → Vault Locked
- ✓ When power applied:

1. Electromagnet activates

2. Plunger retracts

3. Door/Vault opens

- ✓ Known as **Fail-Secure Locking** (Secured when power is removed)

7.6.4 Working Process in System

- ✓ The solenoid lock ensures final physical security:

1. Relay OFF → Lock stays engaged → Unauthorized access blocked

2. Relay ON (after successful authentication):

- ✓ Power flows to solenoid

- ✓ Plunger retracts → Access granted

It plays the key role of granting or denying entry after successful fingerprint & PIN verification.

7.6.5 Applications

1. ATM and bank locker systems
2. Safe boxes and vaults
3. Office and apartment access control
4. Smart home security devices
5. Industrial lockers and cabinets

CHAPTER 8

PROGRAMMING IN EMBEDDED SYSTEMS

As mentioned earlier, Embedded Systems consists of both Hardware and Software. If we consider a simple Embedded System, the main Hardware Module is the Processor. The Processor is the heart of the Embedded System and it can be anything like a Microprocessor, Microcontroller, DSP, CPLD (Complex Programmable Logic Device) and FPGA (Field Programmable Gated Array).

All these devices have one thing in common: they are programmable i.e. we can write a program (which is the software part of the Embedded System) to define how the device actually works.

Embedded Software or Program allow Hardware to monitor external events (Inputs) and control external devices (Outputs) accordingly. During this process, the program for an Embedded System may have to directly manipulate the internal architecture of the Embedded Hardware (usually the processor) such as Timers, Serial Communications Interface, Interrupt Handling, and I/O Ports etc.

From the above statement, it is clear that the Software part of an Embedded System is equally important to the Hardware part. There is no point in having advanced Hardware Components with poorly written programs (Software).

There are many programming languages that are used for Embedded Systems like Assembly (low-level Programming Language), C, C++, JAVA (high-level programming languages), Visual Basic, JAVA Script (Application-level Programming Languages), etc. In the process of making a better embedded system, the programming of the system plays a vital role and hence, the selection of the Programming Language is very important.

EMBEDDED C

Embedded C is one of the most popular and most commonly used Programming Languages in the development of Embedded Systems. So, in this article, we will see some of the Basics of Embedded C Program and the Programming Structure of Embedded C.

Embedded C is perhaps the most popular languages among Embedded Programmers for programming Embedded Systems. There are many popular programming languages like Assembly, BASIC, C++ etc. that are often used for developing Embedded Systems but Embedded C remains popular due to its efficiency, less development time and portability.

Before digging in to the basics of Embedded C Program, we will first take a look at what an Embedded System is and the importance of Programming Language in Embedded Systems.

CHAPTER 9

RESULTS AND DISCUSSIONS

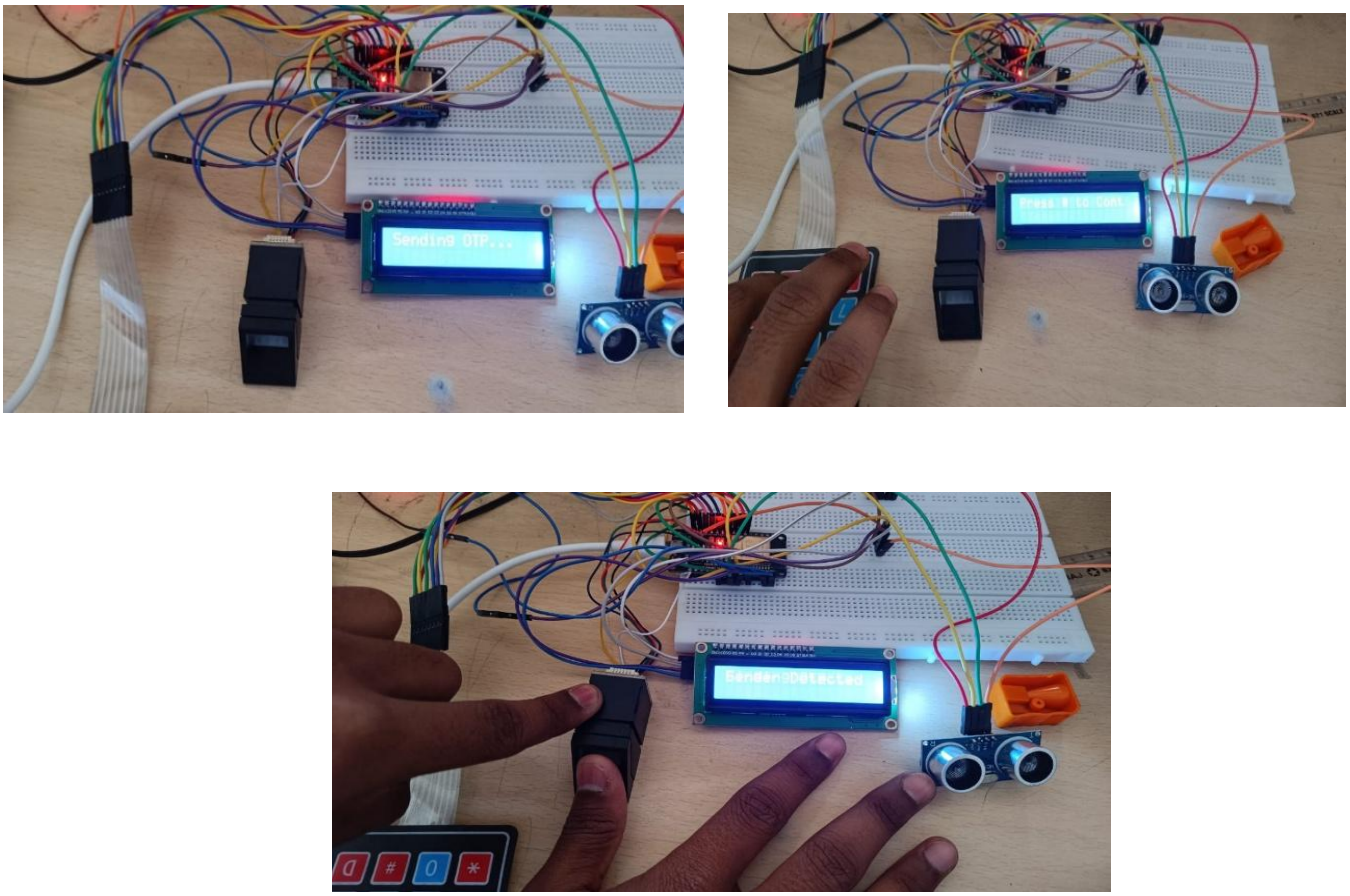

Figure 9.1: Hardware Output

The proposed IoT-enabled Smart Vault Security System with dual authentication using Fingerprint and OTP was successfully developed and tested under different real-time conditions to evaluate its performance, accuracy, reliability, and operational efficiency. The results confirm that the system offers an enhanced security mechanism by integrating both biometric and OTP verification before granting access to the vault.

Hardware Analysis & Results

The hardware components, including the ESP32 microcontroller, R307 fingerprint sensor, 16×2 I2C LCD, relay driver, and solenoid lock, operated effectively as per design requirements. The R307 fingerprint sensor demonstrated quick authentication with high accuracy, enabling successful verification within 1–2 seconds for registered users. The solenoid lock responded immediately after OTP confirmation, ensuring secure unlocking without mechanical delays. The ESP32 provided stable Wi-Fi communication and faster processing compared to traditional microcontrollers like Arduino UNO or GSM-based systems. Power consumption was found to be minimal and reliable for continuous operation. Overall, the hardware integration ensured smooth functionality, physical security, and robust system performance.

Software Analysis & Results

On the software side, fingerprint enrollment, storage, and matching algorithms worked efficiently, allowing secure biometric authentication. Once the fingerprint was verified, the system generated and sent OTP through the Twilio Cloud API to the registered mobile number. The time-bound OTP verification prevented misuse and ensured valid access control. The LCD display successfully showed user notifications such as “Place Finger,” “Access Granted,” and “OTP Verified,” improving user interaction. The backend communication with cloud services was secure, providing encrypted data flow between the ESP32 and server. The overall software integration enhanced usability, remote authentication reliability, and smart monitoring features.

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

The IoT-Enabled Biometric and OTP-Based Smart Vault Security System was successfully developed to provide a highly secure and technologically advanced access control mechanism. By integrating fingerprint authentication with OTP verification, the system ensures multi-level security, making unauthorized access extremely difficult. The use of ESP32 enables real-time wireless communication and remote OTP delivery through cloud services, while the relay-controlled solenoid lock ensures strong physical security. The ultrasonic sensor intelligently activates the system only in the presence of a user, enhancing both efficiency and power management. The overall results confirm that the developed prototype is reliable, user-friendly, and capable of replacing traditional lock mechanisms in high-security environments such as financial vaults, confidential document storage, and industrial access control systems.

Future Enhancement

- ✓ Integration of a **mobile application** for remote vault monitoring and access control
- ✓ Cloud-based storage for **access logs and user management**
- ✓ Addition of **GSM backup** to ensure OTP delivery even without Wi-Fi
- ✓ Implementation of **facial recognition / RFID / NFC** as additional authentication layers
- ✓ Incorporation of **GPS tracking** for mobile vault systems
- ✓ Use of **battery backup** to maintain operation during power failures
- ✓ Adding **tamper detection sensors** to detect forced entry or physical attacks
- ✓ Enhanced **encryption methods** for secure data transmission via IoT
- ✓ Real-time alerts through **email, WhatsApp, or push notifications**
- ✓ Miniaturized and rugged hardware design for **commercial deployment**

REFERENCES

[1] J. Kook, "Design and Implementation of an OTP-Based IoT Digital Door-Lock System," Int. J. Eng. Res. Technol. (IJERT), vol. 12, no. 11, pp. 112–118, 2019.

[2] W. Yang, X. Hu, J. Guo, N. Yan, and L. Wu, "Biometrics for Internet-of-Things Security: A Review," Elsevier / PMC, pp. 1–15, 2021.

[3] S. R. Patel and G. Mehta, "Dual Authentication Method for Secure Access Using Biometrics and OTP," Int. Journal of Computer Applications, vol. 182, no. 32, pp. 18–24, 2022

[4] M. K. Hussain and A. Kumar, "IoT-Based Smart Locker with Biometric and OTP Authentication System," *IEEE Int. Conf. Electronics, Communication and Control Systems*, pp. 85–90, 2022.

[5] R. Paneru, S. Shah, and P. Paneru, "An IoT Based Smart Security Locker System with OTP Verification," in *Proc. Int. Conf. Adv. Comput. Tech.*, pp. 245–250, 2023.

[6] K. L. Deshmukh and R. Patil, "IoT Enabled Smart Vault System Using ESP32," in *Proc. Int. Conf. Smart Technologies for Secure IoT*, pp. 150–156, 2023.

[7] A. S. Ma, J. Choi, and H. Kim, "Fingerprint Based Authentication in IoT Security Systems," *IEEE Trans. Industrial Electronics*, vol. 70, no. 3, pp. 2250–2260, 2024.

[8] M. Nasr, M. Fouad, and M. H. Al-Shamni, "Cancelable Biometric Authentication Leveraging EMD and Quaternion Representations for IoT Security," *Scientific Reports (Nature)*, vol. 15, pp. 1–14, 2025.

[9] A. Alotaibi, "A Review of the Authentication Techniques for Internet of Things," *Sensors Journal (MDPI)*, vol. 14, no. 6, pp. 1–22, 2025.

[10] N. Singh and P. Agarwal, "Smart Locker System Using RFID, Keypad and GSM," in *Proc. Int. Conf. Computing, Communication and Automation (CCCA)*, pp. 301–305, 2025

ANNEXURE I

CODING

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Adafruit_Fingerprint.h>
#include <base64.h>

// Twilio credentials
String account_sid = "AC5285d52d314ac78c4aa11cb9ceb4869b";
String auth_token  = "16d885b8c9c2aacf088857d0e22f0c58";
String from_number = "+19206206896";
String to_number   = "+918525951333";

// WiFi credentials
const char* ssid = "Redmi_12";
const char* password = "123456789";

// LCD setup
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Fingerprint setup
HardwareSerial mySerial(2);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

// Keypad setup
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {13, 12, 14, 27};
byte colPins[COLS] = {26, 25, 33, 32};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
// Relay pin
#define RELAY_PIN 23

// Ultrasonic pins
#define TRIG_PIN 4
#define ECHO_PIN 5

bool systemActive = false;

// Variables
String generatedOTP = "";
String enteredOTP = "";
int enrolledID = -1;

// Function Declarations
bool sendSMS(String message);
String generateOTP();
int getEnrollID();
void enrollFingerprint(int id);
void verifyFingerprint();
void handleOTP();
void getOTPInput();

// Distance Calculation
long getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
  long distance = duration * 0.034 / 2;
  return distance;
}

// --- Send SMS via Twilio ---
bool sendSMS(String message) {
  WiFiClientSecure client;
  client.setInsecure();
```

```
  if (!client.connect("api.twilio.com", 443)) {
    Serial.println("❌ Connection to Twilio failed!");
    return false;
  }

  String url = "/2010-04-01/Accounts/" + account_sid + "/Messages.json";
  String postData = "To=" + to_number + "&From=" + from_number + "&Body=" +
message;
  String auth = base64::encode(account_sid + ":" + auth_token);

  client.println("POST " + url + " HTTP/1.1");
  client.println("Host: api.twilio.com");
  client.println("Authorization: Basic " + auth);
  client.println("Content-Type: application/x-www-form-urlencoded");
  client.print("Content-Length: ");
  client.println(postData.length());
  client.println();
  client.println(postData);

  while (client.connected()) {
    String line = client.readStringUntil('\n');
    if (line.startsWith("HTTP/1.1 201")) {
      Serial.println("✅ OTP sent successfully!");
      return true;
    } else if (line.startsWith("HTTP/1.1 400") || line.startsWith("HTTP/1.1 401")) {
      Serial.println("❌ Failed to send OTP via Twilio");
      return false;
    }
  }
  return false;
}

// --- Generate OTP ---
String generateOTP() {
  int otp = random(1000, 9999);
  return String(otp);
}

// SETUP
void setup() {
```

```
  Serial.begin(115200);
  mySerial.begin(57600, SERIAL_8N1, 16, 17);
  finger.begin(57600);

  lcd.init();
  lcd.backlight();

  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  lcd.setCursor(0, 0);
  lcd.print("Connecting WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    lcd.print(".");
  }

  lcd.clear();
  lcd.print("WiFi Connected");
  delay(1000);

  if (finger.verifyPassword()) {
    Serial.println("✅ Fingerprint sensor OK");
  } else {
    Serial.println("❌ Fingerprint sensor not found!");
    while (1);
  }

  lcd.clear();
  lcd.print("Waiting Object");
}

// MAIN LOOP
void loop() {

  long distance = getDistance();
```

```
if (distance <= 20) {
  if (!systemActive) {
    lcd.clear();
    lcd.print("Object Detected");
    delay(1500);
    lcd.clear();
    lcd.print("Press A to Enroll");
    systemActive = true;
  }
}
else {
  systemActive = false;
  lcd.clear();
  lcd.print("Waiting Object");
  delay(500);
  return;
}

char key = keypad.getKey();

if (key == 'A') {
  lcd.clear();
  lcd.print("Enter ID 1-127");
  int id = getEnrollID();
  if (id > 0) {
    enrollFingerprint(id);
    enrolledID = id;
  } else {
    lcd.clear();
    lcd.print("Invalid ID");
    delay(2000);
  }
  lcd.clear();
  lcd.print("Press # to Start");
}

if (key == '#') {
  if (enrolledID == -1) {
    lcd.clear();
    lcd.print("Enroll First!");
    delay(2000);
```

```
      lcd.clear();
      lcd.print("Press A to Enroll");
      return;
    }
    lcd.clear();
    lcd.print("Place Finger");
    verifyFingerprint();
  }
}

// --- Get ID for Enrollment ---
int getEnrollID() {
  String idStr = "";
  lcd.setCursor(0,1);
  lcd.print("ID: ");
  while (true) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      idStr += key;
      lcd.print(key);
    } else if (key == '#') break;
  }
  return idStr.toInt();
}

// --- Enroll Finger ---
void enrollFingerprint(int id) {
  int p = -1;
  lcd.clear();
  lcd.print("Place Finger #1");

  while ((p = finger.getImage()) != FINGERPRINT_OK);
  finger.image2Tz(1);

  lcd.clear();
  lcd.print("Remove Finger");
  delay(2000);
  while (finger.getImage() != FINGERPRINT_NOFINGER);

  lcd.clear();
  lcd.print("Place Finger #2");
```

```
  while ((p = finger.getImage()) != FINGERPRINT_OK);
  finger.image2Tz(2);

  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
    finger.storeModel(id);
    lcd.clear();
    lcd.print("Enroll Success!");
    Serial.println("✅ Enrolled ID: " + String(id));
  } else {
    lcd.clear();
    lcd.print("Enroll Failed!");
  }
  delay(2000);
}

// --- Finger Authentication ---
void verifyFingerprint() {
  int p = finger.getImage();
  if (p != FINGERPRINT_OK) return;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK) return;

  p = finger.fingerSearch();
  if (p == FINGERPRINT_OK && finger.fingerID == enrolledID) {
    lcd.clear();
    lcd.print("Match Found!");
    delay(1000);
    handleOTP();
  } else {
    lcd.clear();
    lcd.print("Not Matched!");
    delay(2000);
    lcd.clear();
    lcd.print("Press # to Try");
  }
}

// --- Handle OTP ---
void handleOTP() {
```

```
  generatedOTP = generateOTP();
  Serial.println("Generated OTP: " + generatedOTP);
  lcd.clear();
  lcd.print("Sending OTP...");

  bool sent = sendSMS("Your OTP is " + generatedOTP);
  delay(1000);

  if (!sent) {
    lcd.clear();
    lcd.print("SMS Failed!");
    delay(2000);
    lcd.clear();
    lcd.print("Press # to Retry");
    return;
  }

  lcd.clear();
  lcd.print("Enter OTP:");
  enteredOTP = "";
  getOTPInput();
}

// --- Read OTP Input ---
void getOTPInput() {
  while (true) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      enteredOTP += key;
      lcd.setCursor(enteredOTP.length() - 1, 1);
      lcd.print("*");
    }

    if (enteredOTP.length() == 4) {
      if (enteredOTP == generatedOTP) {
        lcd.clear();
        lcd.print("OTP Correct!");
        digitalWrite(RELAY_PIN, HIGH);
        lcd.clear();
        lcd.print("Door Unlocked");
        delay(60000);
```

```
      digitalWrite(RELAY_PIN, LOW);
      lcd.clear();
      lcd.print("Door Locked");
      delay(2000);
    } else {
      lcd.clear();
      lcd.print("Wrong OTP");
      delay(2000);
    }
    lcd.clear();
    lcd.print("Press # to Start");
    break;
  }
}
```