OMNILOCK-SMART DOOR LOCKING SYSTEM

Bachelor of Technology

In

Computer Science and Engineering (Internet of Things)

By

Devansh Mishra	2211CS050023
K. Thanmai Reddy	2211CS050040
K. Shivani	2211CS050046

Under the esteemed guidance of

Dr. G. Anand Kumar

Head of the Department, Dept. of Cyber Security & IoT

Department of Computer Science & Engineering (Internet of Things)

School of Engineering

MALLA REDDY UNIVERSITY

Maisammaguda, Dulapally, Hyderabad, Telangana 500100

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Department of Computer Science & Engineering

(Internet of Things)

CERTIFICATE

This is to certify that the project report entitled **"OMNILOCK-SMART DOOR LOCKING SYSTEMS",** submitted by **Devansh Mishra(2211CS050023), K. Thanmai Reddy(2211CS050040), K. Shivani(2211CS050046)** of 2 year IoT – ALPHA **Department of Computer Science and Engineering, Malla Reddy University, Hyderabad**, is a record of bonafide work done by him/ her. The results embodied in the work are not submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Dr. G. Anand Kumar	Dr. G. Anand Kumar
HoD CSE(Cyber Security & IoT)	HOD CSE(Cyber Security & IoT)

External Examiner

DECLARATION

We hereby declare that the project report entitled **"OMNILOCK-SMART DOOR LOCKING SYSTEM"**, has been carried out by us. This work has been submitted to the **Department of Computer Science and Engineering (Internet of Things), MallaReddy University, Hyderabad** for the award of the degree of Bachelor of Technology. We further declare that this project work has not been submitted in full or in part for the award of any other degree in any other educational institutions.

Place:

Date:

Devansh Mishra 2211CS050023

K. Thanmai Reddy 2211CS050040

K. Shivani 2211CS050046

ACKNOWLEDGEMENT

We extend our sincere gratitude to all those who have contributed to the completion of this project report. Firstly, we would like to extend our gratitude to Dr. V. S. K Reddy, Vice-Chancellor, for his visionary leadership and unwavering commitment to academic excellence.

We would also like to express our deepest appreciation to our project guide Dr. G. Anand Kumar Associate Professor, Dept. of Cyber Security & IoT, whose invaluable guidance, insightful feedback, and unwavering support have been instrumental throughout this project for successful outcomes.

We are also grateful to Dr. G. Anand Kumar, Head of the Department of Cyber Security & IoT, for providing us with the necessary resources and facilities to carry out this project.

We extend our gratitude to our PRC-convenor, Dr. G. Latha, for giving valuable inputs and timely guidelines to improve the quality of our project through a critical review process. We thank our project coordinator, Dr. G. Latha, for his timely support.

We would like to thank Dr. Kasa Ravindra, Dean, School of Engineering, for his encouragement and support throughout my academic pursuit.

My heartfelt thanks also go to Dr. Harikrishna Kamatham, Associate Dean School of Engineering for his guidance and encouragement.

We are deeply indebted to all of them for their support, encouragement, and guidance, without which this project would not have been possible.

Devansh Mishra 2211CS050023

K. Thanmai Reddy 2211CS050040

K. Shivani 2211CS050046

ABSTRACT

By integrating cutting-edge biometrics with the capabilities of the Internet of Things (IoT), Omnilock enhances access control. This creative solution uses an Arduino microcontroller and ESP32 combination to produce a reliable, easy-to-use smart lock application. With its integration of touchpad entry, fingerprint authentication, and remote access through the Web server. Omnilock goes beyond traditional methods to provide unmatched security and convenience. Each feature of the system may operate independently due to its modular architecture, which guarantees optimal performance and a consistent user interface. Omnilock places security first, users can choose from secure login methods like a fingerprint sensor or a simple touchpad for convenient access. Additionally, Omnilock offers remote access, allowing you to unlock the door for specified individuals even when you are not physically present. A buzzer that sounds when entry is denied serves as a barrier to unwanted attempts. In the end, Omnilock bills itself as a safe access control system that can be used in homes and other locations, giving consumers a smooth and adaptable interface.

List of Diagram

List of Algorithm

List of Screenshots

Table of Contents

CHAPTER-1

INTRODUCTION

1.1 Problem Definition & Description

Traditional methods like physical keys can be easily lost or stolen, and key management can be cumbersome. Additionally, they lack features like remote access and access control for different users.

- Managing and distributing physical keys, especially for multiple individuals, can be a logistical nightmare.
- Keeping track of who has which key and ensuring everyone has the necessary access can be challenging.
- Resulting in inconvenience, cost of replacement, and potential security risks.
- Traditional locks can be susceptible to picking or forced entry.
- Traditional locks cannot be controlled remotely, making it impossible to grant or revoke access from afar.

1.2 Objectives of the Project

The primary objective of the Omnilock project is to develop an advanced Smart Door Lock System that leverages cutting-edge biometric technologies and the capabilities of the ESP32 and Arduino microcontroller to provide unparalleled security and convenience. By integrating fingerprint scanning, keypad entry, and remote access into a cohesive system, Omnilock aims to offer robust identity verification, significantly reducing the risk of unauthorized access. The project seeks to address the shortcomings of traditional access control methods by delivering a comprehensive solution that is both secure and user-friendly.

1.3 Scope of the Project

The scope of the Omnilock project encompasses the design, development, and implementation of a multi-modal authentication Smart Door Locking System. The key components and features to be included in the system are:

1. Fingerprint Sensor: Biometric authentication for secure access control based on unique fingerprint patterns.

2. Keypad Entry: Input device for users to enter PIN codes, offering an alternative access method.

3. Remote access control: Web-based interface enabling users to manage access permissions and monitor activity remotely.

4. Buzzer: Audible alert system activated upon detecting unauthorized access attempts, enhancing security measures**.**

CHAPTER-2

SYSTEM ANALYSIS

2.1 Existing System

2.1.1 Background & Literature Survey

Omnilock redefines door access with biometric authentication and remote control, meeting demands for heightened security and convenience. Recent research underscores the growing interest in smart locks, particularly in biometrics and wireless communication, positioning solutions like Omnilock at the forefront of access control innovation.

“Property Security Using a Biometric Based Door Lock System” was proposed by OnyanA and Enalume K. The paper is about the creation and implementation of a Biometric Based Door Lock System, which will automatically unlock a door when a registered fingerprint is detected. To achieve this, a fingerprint scanner R305 is used in conjunction with an ATMEGA 328 Arduino microcontroller to monitor the locking and unlocking process of a door. Access is given to the user after a recorded fingerprint is put on the sensor, the door slides open, and it closes after five seconds. The 16x2 Liquid Crystal Display (LCD) shows the individual's name along with the registered fingerprint during this process. Access is refused if an unregistered fingerprint is detected.

A paper named “Secured password-based lock system” was put forward by “Arpita Mishra, Siddharth Sharma, Sachin Dubey, and S.K. Dubey”. This strategy is focused on avoiding the opening of the entryway by obscure people. The arrangement of the domestic security benefit comprises the numeric keypad, the snare which is utilized for lifting, and a GSM module to set up reliable association for communication conferred with the MCU. The control board conferred with the gadget is utilized since the passcode gets to combination opens/closes the entryway.

“A locks which operate by a secret knock” which was put forward by “Dr.M.Siva Sangari, Dhivakar. E, Gowthaam. K”. To open and close the lock by burrowing out the thumps detected by the piezo sensor to open the entryway, a servo turns when it identifies the thumps and opens and closes the entryway.

“Android Based Smart Door Locking System” which was proposed by “Adarsh V Patil, Sreevarsha Prakash, Akshay S, Mahadevaswamy, Chandan B Patgar, Sharath Kumar A J”. The method executes on a pre-coded passcode idea. It moves forward the security prepares to halt an unathletically pulverizing done by a false.

“SMART DOOR UNLOCK SYSTEM USING FINGERPRINT” was proposed by K. Rajesh, Asst.Prof. B.VenkataRao, P.AV.S.K.Chaitanya, and A.Ruchitha Reddy. In this paper, they have used a fingerprint sensor to read one's identity to operate the door of the car automatically. They used a microcontroller to enable the door to open or close if the matching between scanned data and the existing data is correct. Comparison is done inside the fingerprint module itself and its output is given to the microcontroller. The result is displayed on an LCD whether the user is authorized or not. LCD also helps to make troubleshooting easier. The alarming option is provided to warn about unauthorized usage. The microcontroller used is PIC16F877.

2.1.2 Limitations of Existing System

Traditional door locks that just require physical keys are vanishing from use. Their weaknesses are many: keys are easily lost, stolen, or even duplicated, putting your house or place of business's security at risk. A stolen key can provide an unauthorized entrance, while a lost key can leave you stranded outside. Sharing access also becomes problematic because it frequently necessitates giving out a physical key, which is dangerous and inconvenient, particularly for temporary visitors or service providers. Moreover, carrying around several keys is inconvenient since it adds extra weight to your pockets and causes you to constantly worry about losing them. Traditional locks and keycards can be inconvenient or difficult for people with disabilities. There's no way to track who accessed the door and when with a traditional lock, making it difficult to monitor activity. Despite appearing to be a benefit, RFID technology raises additional issues. Because they rely on radio frequency transmissions, they can be copied, which could lead to unauthorized entry via intercepted communications. Omnilock provides a multi-modal authentication system that balances ease and security, surpassing the drawbacks of conventional key-based access management. Physical keys are rendered obsolete by fingerprint scanners and secure keypads. The chance of physical keys being misplaced, stolen, or duplicated is eliminated, which dramatically lowers the risk of illegal access. Omnilock's possible interaction with the web server enables remote unlocking and access management. This improves security and peace of mind by enabling real-time entryway monitoring. When access is refused, a loud alarm also sounds to discourage unwanted attempts. More control and security are ensured by doing away with the need to issue actual keys, particularly for temporary access needs like deliveries or service visits.

2.2 Proposed System

Omnilock provides a multi-layered solution to access management, addressing the drawbacks of standalone biometric systems, RFID, and classic locks. Security can be compromised by physical keys, which are readily lost or stolen. By using biometrics (fingerprint technology) and secure keypads, Omnilock removes this risk and increases the difficulty of unwanted entry using lost or stolen credentials.

Despite their ease, RFID technologies have a serious security vulnerability. Since they communicate by radio waves, they are vulnerable to copying and interception. With the correct tools, a malicious actor might be able to make a duplicate RFID card and obtain unauthorized access. Omnilock directly addresses this problem. Because biometrics are unique to each person, they greatly lessen the possibility of unwanted access through credential theft. The system may become inoperable due to a broken fingerprint scanner. The multi-modal approach of Omnilock provides users with additional techniques (remote access, keypad, or fingerprint) for user authentication, ensuring access even if one component malfunctions. The ESP32 may be utilized for remote access. However, its processing power and security features may be insufficient for the best results. Using an Arduino's advantages in communication and control, Omnilock combines the ESP32 with it to create a more robust solution.

2.2.1 Advantages of Proposed System

- The primary advantage of Omnilock is its variety of authentication options. This makes it far more difficult than with single-factor systems for unauthorized people to obtain access.
- An increased degree of security is offered by biometric verification as opposed to conventional keys or RFID cards. Biometric authentication provides an additional degree of security, even if a user's PIN code is stolen.
- Depending on their comfort level and circumstances, users can select their preferred authentication method. While fingerprint gives a possibly speedier method, the keypad offers a more comfortable option. Even if a user is not available then they can use remote access.
- Future integration with other technologies, such as smart home systems or innovative safety features, is made possible by the modular design.

2.3 Software & Hardware Requirements

2.3.1 Software Requirements

- Web Server
- Security Libraries
- Arduino IDE
- Windows
- Programming Language
- Drivers

2.3.2 Hardware Requirements

- Laptop with 8 GB RAM
- Fingerprint sensor-r307s
- Keypad
- Buzzer
- Esp 32
- Arduino Uno
- USB cables
- Servo motor
- Breadboard
- Jumper Wires
- LCD Display

2.4 Feasibility Study

2.4.1 Technical Feasibility

The Omnilock app demonstrates strong technical feasibility, leveraging mature technologies like fingerprint sensors and keypads, alongside seamless integration with established authentication frameworks and web servers. Compatibility with major operating systems and development platforms ensures broad accessibility, while robust security measures enhance data protection, ensuring the app's technical viability.

2.4.2 Robustness & Reliability

Omnilock guarantees reliability with stable performance, thorough error handling, and strong security measures. Rigorous testing and user feedback improve dependability, while regular updates maintain long-term reliability.

2.4.3 Economic Feasibility

Omnilock's economic feasibility lies in its cost-effectiveness, competitive pricing, and scalability. With growing market demand for smart security solutions, it offers a promising return on investment for homeowners and businesses alike.

CHAPTER-3

ARCHITECTURAL DESIGN

3.1 Modules Design

3.1.1 User Interface Module

- Responsible for giving an interface for clients to associate with the system.
- Includes functionalities such as user authentication, access request submission, and viewing system status.
- It is designed with a user-friendly interface for ease of use and accessibility.

3.1.2 Access Controller Module

- Manages access requests from users and coordinates the authentication process.
- Acts as an intermediary between the user interface and the security system.
- Responsible for initiating the authentication process and handling access control decisions based on authentication results.

3.1.3 Security System Module

- The central component is responsible for authentication and access control.
- Consists of sub-modules for various authentication methods such as fingerprint recognition, remote access, and PIN authentication.
- Implements access control policies based on authentication results and grants or denies access accordingly.

3.1.4 Authentication Module

- Handles the authentication process for verifying the identity of users.
- Includes sub-modules for different authentication methods, such as biometric authentication (fingerprint technology), traditional methods (PIN authentication), and remote access via web server.

3.1.5 Access Control Module

- Implements access control policies based on authentication results.

- Determines whether users are granted or denied access to secured areas based on authentication outcomes.
- Interfaces with the security system to enforce access control decisions and manage access permissions.
- The buzzer can be activated for a predetermined duration by a function that the Access Control Module may trigger in response to a "denied access" signal.

3.1.6 Logging & Notification Module

- Responsible for logging system events and generating notifications/alerts.
- Records access control activities, authentication attempts, and system errors for audit and monitoring purposes.
- Notifies administrators or users about security-related events, such as unauthorized access attempts or system malfunctions.

3.2 Methods & Algorithm Design

3.2.1 PIN Authentication Method

Omnilock provides a familiar and user-friendly keypad for PIN entry, allowing customers to create unique PINs during setup for a fast and dependable approach to unlocking the door. The LCD displays the lock state, which adds an added layer of security. The technology allows you to customize the length of your PIN, balancing security with user convenience. If an erroneous PIN is entered, a buzzer will sound to warn potential intruders.

3.2.2 Fingerprint Authentication Method

Omnilock has a fingerprint scanner that uses optical or capacitive sensors to collect an individual's unique biometrics. During enrollment, customers scan their fingerprints several times to generate a comprehensive blueprint for proper identification. Omnilock may additionally include liveness detecting technologies to ensure that only a real finger is used, preventing attempts to spoof the system with fake fingerprints. In the event of a failed fingerprint recognition attempt, the buzzer will sound to prevent unauthorized entry.

3.2.3 Remote Access via Web Server

Omnilock offers the convenience of remote access via a web server, which connects to a central server and allows authorized users to lock or unlock doors through a secure web application. The interface

typically displays the current lock state and allows users to send locking or unlocking instructions remotely. Secure communication protocols encrypt data and prevent unauthorized access while in remote control. In addition, the system provides notification alerts when the door is opened and closed.

3.2.4 Fingerprint Verification Algorithm

The technology used in fingerprint sensors does more than just take a picture of your fingers. The actual magic of user identification is carried out by complex algorithms that are included in these sensors. These exclusive algorithms function directly on the sensor, guaranteeing a safe and effective procedure. Since the sensor vendor closely guards the specifics of these algorithms, they are frequently private. Because it becomes harder for potential hackers to exploit the underlying workings of the algorithms, this strategy maintains security. These sensor-specific algorithms also take advantage of the hardware's capabilities to optimize processing times and power consumption. The sensor's fingerprint verification algorithms serve as the gatekeepers, carefully examining each characteristic of your fingerprint to provide safe and practical user identification.

3.2.5 String Comparison Algorithm

PIN code protection is just one more way that Omnilock puts the security of its users first. An important vulnerability would be a straightforward string comparison between the entered PIN and a stored PIN code. A hacker might simply open the door if they managed to get the stored PIN codes. Omnilock uses hashing functions to solve this issue. These cryptographic functions create a unique, fixed-length string of characters termed a hash using the user's PIN code as input. The original PIN is securely represented by this hash. It's crucial to remember that the hashing function is one-way; the original PIN code cannot be recovered by mathematically reversing the hash.

3.3 Project Architecture

3.3.1 Architectural Diagram

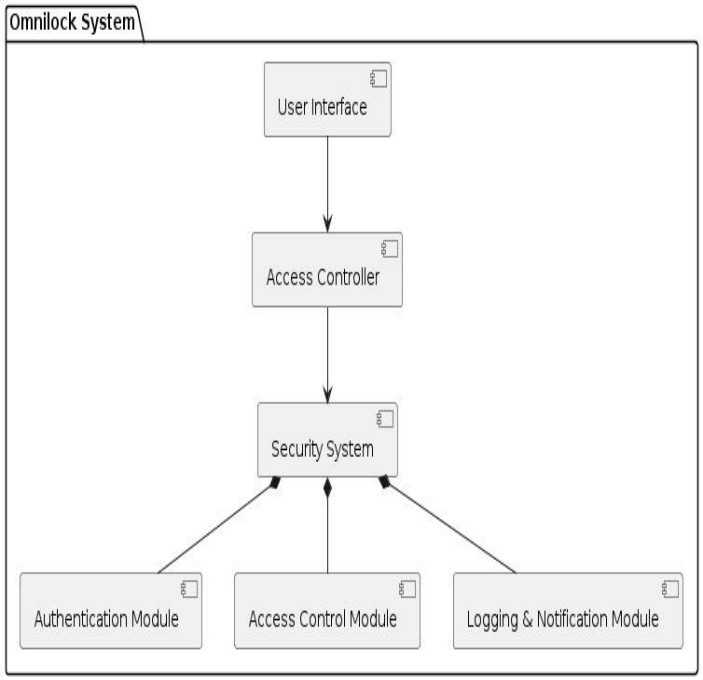


Figure 3.3.1 Architectural Diagram

- **System Overview:** Omnilock is a security system designed for access control, ensuring secure entry to facilities through advanced authentication methods.
- **Component Breakdown**:
    - User Interface: Allows users to interact with the system.
    - Access Controller: Manages access requests and coordinates with the Security System.
    - Security System: Central component responsible for authentication and access control.
    - Authentication Module: Handles biometric and PIN authentication.
    - Access Control Module: Implements access policies based on authentication results.
    - Logging & Notification Module: Responsible for logging system events and generating notifications/alerts.
- **Communication Flow**: Requests for access from users are processed through the Access Controller, which interacts with the Security System for authentication and access control.
- **Scalability and Integration**: Omnilock's modular architecture allows for scalability and integration with other security systems, accommodating various environments and security needs.
- **Benefits**: The architectural design of Omnilock ensures enhanced security, user convenience, and ease of management, making it a robust solution for access control requirements.

3.3.2 Data Flow Diagram

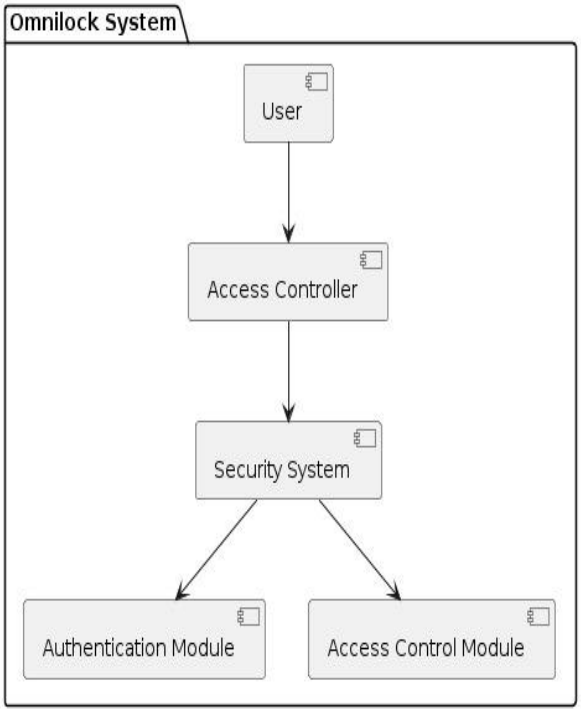


Figure 3.3.2 Data Flow Diagram

- **Data Flow Visualization**: The DFD provides a visual representation of how data flows within the Omnilock system, showing inputs, outputs, and processes involved in access control.
- **Component Interaction**: It illustrates how different components of Omnilock, such as the User, Access Controller, and Security System, interact by exchanging data during the access control process.
- **Data Transformation**: The DFD highlights how data is transformed as it moves through the system, such as user access requests being processed and authenticated by the Security System.
- **Boundary Identification**: It helps identify the boundaries of the system by illustrating external entities (e.g., users) interacting with internal components (e.g., Access Controller and Security System).
- **System Functionality**: By visualizing the flow of data between components, the DFD provides insight into the functionality of Omnilock, emphasizing its role in ensuring secure access control for various environments.

3.3.3 Class Diagram

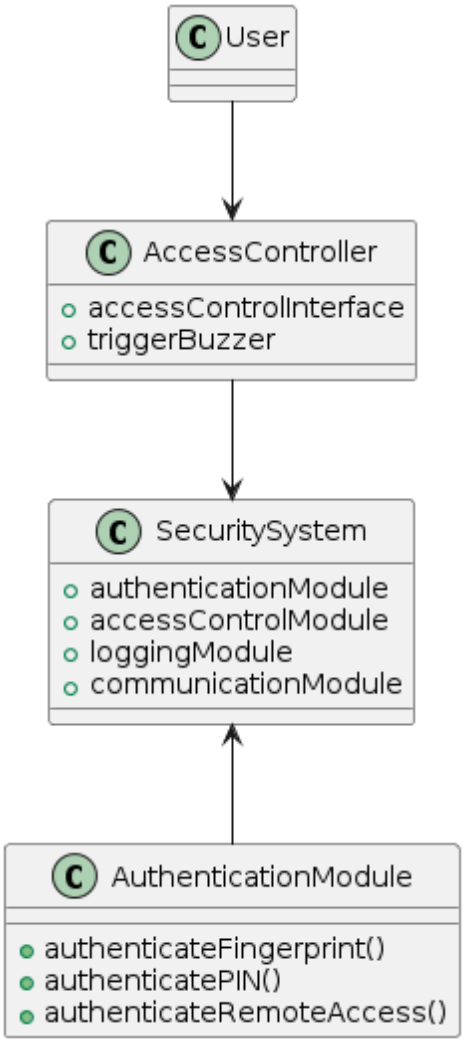


Figure 3.3.3 Class Diagram

- **Structural Representation**: The Class Diagram provides a structural representation of the Omnilock system, illustrating the classes (components) and their relationships.
- **Component Identification**: It identifies key components of Omnilock, such as the User, Access Controller, Security System, and Authentication Module, along with their attributes and methods.
- **Relationships**: The Class Diagram depicts relationships between components, such as associations, dependencies, and composition, highlighting how different parts of the system interact.
- **Abstraction of System**: By abstracting complex system architecture into manageable components and relationships, the Class Diagram facilitates understanding and communication among stakeholders.
- **Design Documentation**: It serves as a valuable documentation tool for system design, aiding in the development, maintenance, and communication of design decisions among developers and stakeholders.

3.3.4 Use case Diagram

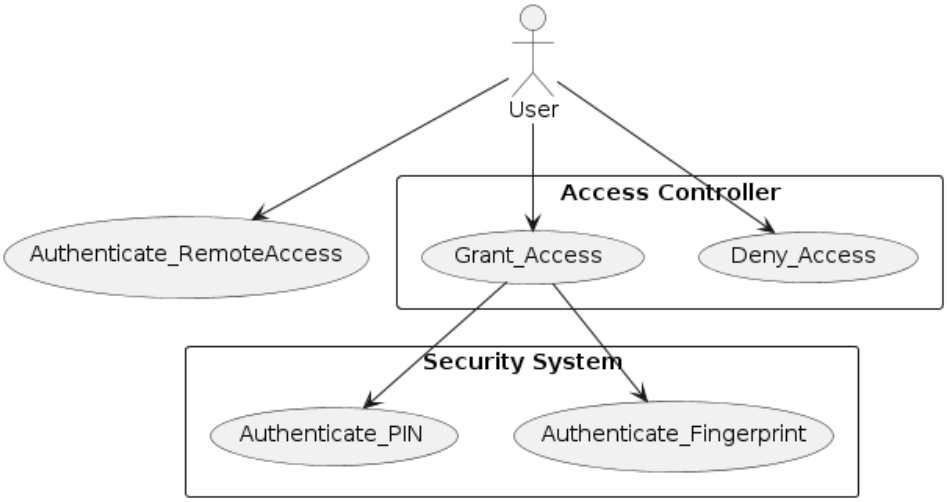


Figure 3.3.4 Use case Diagram

- **Functional Overview**: The Use Case Diagram provides a functional overview of the Omnilock system, showcasing the interactions between external actors (users) and the system.
- **User Actions**: It identifies the actions or functionalities that users can perform within the system, such as requesting access, authentication, and receiving access grants.
- **System Components**: The Use Case Diagram highlights the major system components or actors involved in the use cases, such as the User, Access Controller, and Security System.
- **Interaction Flow**: It illustrates the flow of interactions between users and the system, showcasing how users initiate actions and receive responses from the system.

3.3.5 Sequence Diagram

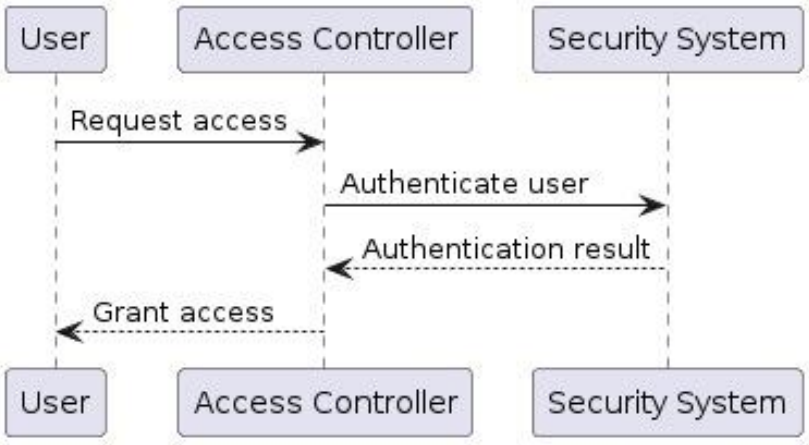


Figure 3.3.5 Sequence Diagram

- **Interaction Visualization**: The Sequence Diagram provides a visual representation of the interactions between different components of the Omnilock system, illustrating the flow of messages and actions over time.
- **Behavioral Understanding**: It helps in understanding the dynamic behavior of the system by showing the sequence of steps and communication exchanges between actors and system components during specific scenarios, such as the authentication process.
- **Temporal Ordering**: The Sequence Diagram depicts the temporal ordering of messages exchanged between actors and system components, providing insights into the chronological flow of interactions.
- **Detailed Communication**: It offers a detailed view of the communication between different components, including the types of messages exchanged and the sequence of method invocations.
- **Error Handling**: The Sequence Diagram can illustrate error handling and exception scenarios, showing how the system responds to unexpected events or errors during the execution of a sequence of actions.
- **Collaboration Visualization**: It visualizes the collaboration between actors and objects, helping stakeholders understand how different parts of the system work together to achieve specific tasks or objectives.
- **Design Validation**: Sequence Diagrams can be used to validate the design of the system by ensuring that the sequence of interactions aligns with the expected behavior and requirements of the system.

3.3.6 Activity Diagram

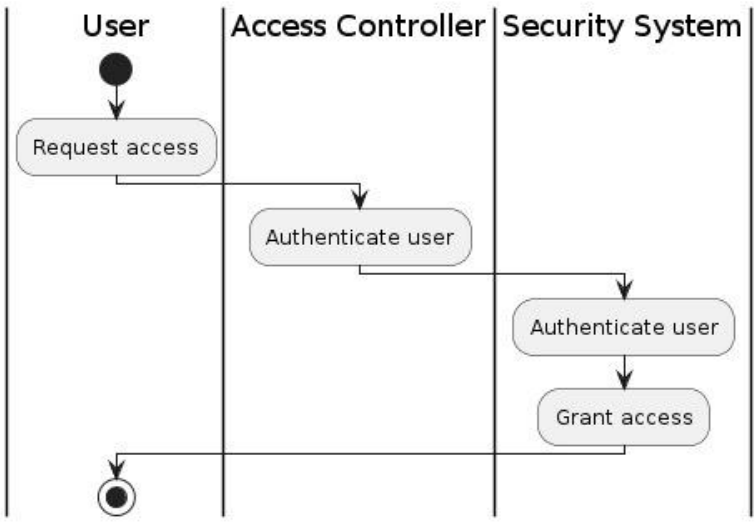


Figure 3.3.6 Activity Diagram

- **Process Visualization**: The Activity Diagram provides a visual representation of the workflow or processes involved in the access control system of Omnilock, showcasing the sequence of activities and decision points.
- **Activity Identification**: It identifies the various activities or tasks performed within the system, such as user access requests, authentication processes, and access grants.
- **Control Flow**: The Activity Diagram illustrates the control flow between activities, including decision points, branching, and parallel activities, helping stakeholders understand the logical flow of operations.
- **Concurrency and Synchronization**: It can depict concurrent activities and synchronization points within the system, highlighting how multiple tasks can be performed simultaneously or synchronized at specific points.
- **User Interaction**: The Activity Diagram shows the interaction between users and the system, illustrating how users initiate actions and navigate through different activities to accomplish specific goals, such as gaining access to secured areas.

CHAPTER-4

IMPLEMENTATION & TESTING

4.1 Coding Blocks

4.1.1 Servo Motor code

```
void ServoOpen(Servo s) {
  Serial.println("Opening servo...");
  for (int pos = s.read(); pos <= servoMax; pos += 1) {
    s.write(pos);
    Serial.print("Servo position: ");
    Serial.println(pos);
    delay(15);
  }
}
void ServoClose(Servo s) {
  Serial.println("Closing servo...");
  for (int pos = s.read(); pos >= servoMin; pos -= 1) {
    s.write(pos);
    Serial.print("Servo position: ");
    Serial.println(pos);
    delay(15);
  }
}
```

4.1.2 Fingerprint code

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <Servo.h> //Add servo library
int getFingerprintIDez();
Servo servo1; //Define servo name/object
#define servoPin 9 //Define the pin number to which the servo motor is connected
#define durationTime 3000 //Define the time it remains in the open position of the door lock (miliseconds)
#define servoMin 0 //Open position
#define servoMax 90 // Closed position
SoftwareSerial mySerial(12,13);
```

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
void setup()
{
while (!Serial); // For Yun/Leo/Micro/Zero/...
Serial.begin(9600);
Serial.println("Adafruit finger detect test");
servo1.attach(servoPin); //Define pin number of the servo
servo1.write(servoMax); //The position of the servo at the start of the program
// 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);
}
Serial.println("Waiting for valid finger...");
}
void loop() // run over and over again
{
getFingerprintIDez();
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");
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");
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
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");
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
return p;
// found a match!
Serial.print("Door is Open");
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);
}
// 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;
servo1.write(servoMin); //If the fingerprint is correct open the door lock
delay(durationTime); //Keep the lock open for the defined duration
servo1.write(servoMax); //take the lock OFF again
// found a match!
Serial.print("Found ID #"); Serial.print(finger.fingerID);
Serial.print(" with confidence of "); Serial.println(finger.confidence);
return finger.fingerID;
}
```

4.1.3 Keypad code

```
#include <Keypad.h> // the library for the 4x4 keypad
#include <LiquidCrystal_I2C.h> // the library for the i2c 1602 lcd
```

```
#include <Servo.h> // the library to control the servo motor
LiquidCrystal_I2C lcd(0x20,16,2); // gets the lcd
Servo servo;
#define Password_Length 5 // the length of the password, if the password is 4 digits long set this to 5
int Position = 0; // position of the servo
char Particular[Password_Length]; // the password length
char Specific[Password_Length] = "1234"; // the password which is called specific in the code, change this to anything you want with the numbers 0-9 and the letters A-D
byte Particular_Count = 0, Specific_Count = 0; // counts the amount of digits and and checks to see if the password is correct
char Key;
const byte ROWS = 4; // the amount of rows on the keypad
const byte COLS = 4; // the amount of columns on the keypad
char keys[ROWS][COLS] = { // sets the rowns and columns
  // sets the keypad digits
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
{'*','0','#','D'}
};
bool SmartDoor = true; // the servo
// the pins to plug the keypad into
byte rowPins[ROWS] = {2, 3, 4, 5};
byte colPins[COLS] = {6, 8, 9, 10};
Keypad myKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS); // gets the data from the keypad
// locked charcater
byte Locked[8] = {
  B01110,
  B10001,
  B10001,
  B11111,
  B11011,
  B11011,
  B11011,
  B11111
```

```
};
// open character
byte Opened[8] = {
  B01110,
  B00001,
  B00001,
  B11111,
  B11011,
  B11011,
  B11011,
  B11111
};
void setup()
{
  servo.attach(11); // attaches the servo to pin 10
  ServoClose(); // closes the servo when you say this function
  lcd.init(); // initializes the lcd
  lcd.backlight(); // turns on the backlight
  lcd.setCursor(0,0); // sets the cursor on the lcd
  lcd.print("Vector X"); // prints the text/charater
  lcd.setCursor(0,1); // sets the cursor on the lcd
  lcd.print("Arduino Lock!!!"); // prints text
  delay(4000); // waits 4 seconds
  lcd.clear(); // clears the lcd diplay
}
void loop()
{
  if (SmartDoor == 0) // opens the smart door
  {
    Key = myKeypad.getKey(); // the word key = myKeypad which gets the value
    if (Key == '#') // when the '#' key is pressed
    {
      lcd.clear(); // clears the lcd diplay
      ServoClose(); // closes the servo motor
      lcd.setCursor(2,0); // sets the cursor on the lcd
```

```
      lcd.print("Door Closed"); // prints the text to the lcd
      lcd.createChar(0, Locked); // prints the locked character
      lcd.setCursor(14,0); // sets the cursor on the lcd
      lcd.write(0); // prints the first character when you are on the door closed page
      delay(3000); // waits 3 seconds
      SmartDoor = 1; // closes the door
    }
  }
  else Open(); // keeps the door open
}
void clearData() // clears the data
{
  while (Particular_Count != 0) // counts the digits pressed
  {
    Particular[Particular_Count--] = 0; // counts how many digits
  }
  return; // returns the data
}
void ServoOpen() // opens the servo
{
  for (Position = 180; Position >= 0; Position -= 5) { // moves from 0 to 180 degrees
    servo.write(Position); // moves to the postion
    delay(15); // waits 15 milliseconds
  }
}
void ServoClose() // closes the servo
{
  for (Position = 0; Position <= 90; Position += 5) { // moves from position 0 to 180 degrees
    servo.write(Position); // moves to the position
    delay(15); // waits 15 milliseconds
  }
}
void Open() // function declarations
{
  lcd.setCursor(1,0); // sets the cursor on the lcd
```

```
  lcd.print("Enter Password:"); // prints the text

  Key = myKeypad.getKey(); // gets the keys you press from the keypad
  if (Key)
  {
   Particular[Particular_Count] = Key;
   lcd.setCursor(Particular_Count, 1); // sets the cursor on the lcd
   lcd.print("*"); // prints '*' instead of the password
   Particular_Count++; // counts the length of the password
  }
  if (Particular_Count == Password_Length - 1) // gets the length of the password
  {
   if (!strcmp(Particular, Specific)) // counts the length and checks to see if the password is correct
   {
    lcd.clear();
    ServoOpen(); // moves the servo 180 degrees
    lcd.setCursor(2,0); // sets the cursor on the lcd
    lcd.print("Door Opened");
    lcd.createChar(1, Opened);
    lcd.setCursor(14,0); // sets the cursor on the lcd
    lcd.write(1);
    lcd.setCursor(0,1); // sets the cursor on the lcd

lcd.print("Press # to Close");
    SmartDoor = 0;
   }
   else
   {
    lcd.clear();
    lcd.setCursor(0,0); // sets the cursor on the lcd
    lcd.print("Wrong Password"); // prints the text/character
    lcd.setCursor(0,1);
    lcd.print("Try Again In");
    lcd.setCursor(13,1);
    lcd.print("10");
```

```
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("09");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("08");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("07");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("06");
    delay(1000);
    lcd.setCursor(13,1);

lcd.print("05");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("04");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("03");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("02");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("01");
    delay(1000);
    lcd.setCursor(13,1);
    lcd.print("00");
    delay(1000);
    lcd.clear();
    SmartDoor = 1; // closes the smart door
  }
```

```
    clearData(); // clears the data
  }
}
```

4.1.4 Remote Access code

```
#include <WiFi.h>
#include <ESP32Servo.h>

Servo myservo;  // Create servo object to control a servo

// GPIO the servo is attached to
static const int servoPin = 13;

// Replace with your network credentials
const char* ssid = "Laptop";
const char* password = "#dev11oc04";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Decode HTTP GET value
int servoPosition = 0;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200); // Set baud rate to 115200 for the serial monitor
```

```
  myservo.attach(servoPin); // Attaches the servo on the servoPin to the servo object

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  // Improved connection feedback
  unsigned long startTime = millis();
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    // Optional: You can add a timeout for connection attempts
    if (millis() - startTime > 10000) { // 10 seconds timeout
      Serial.println("\nFailed to connect to WiFi, restarting...");
      ESP.restart(); // Restart the ESP32
    }
  }

  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop() {
  WiFiClient client = server.available(); // Listen for incoming clients

  if (client) { // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client."); // Print a message out in the serial port
```

```
  String currentLine = ""; // Make a String to hold incoming data from the client
  while (client.connected() && currentTime - previousTime <= timeoutTime) { // Loop while the client's connected
    currentTime = millis();
    if (client.available()) { // If there's bytes to read from the client,
      char c = client.read(); // Read a byte, then
      Serial.write(c); // Print it out the serial monitor
      header += c;
      if (c == '\n') { // If the byte is a newline character
        // If the current line is blank, you got two newline characters in a row.
        // That's the end of the client HTTP request, so send a response:
        if (currentLine.length() == 0) {
          // Parse GET request before sending the response
          bool servoActionTaken = false;
          String action;
          if (header.indexOf("GET /open") >= 0) {
            servoPosition = 0; // Change to the closed position
            myservo.write(servoPosition);
            Serial.println("Door Opened");
            servoActionTaken = true;
            action = "open";
          }
          if (header.indexOf("GET /close") >= 0) {
            servoPosition = 90; // Change to the open position
            myservo.write(servoPosition);
            Serial.println("Door Closed");
            servoActionTaken = true;
            action = "close";
          }

          // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
          // and a content-type so the client knows what's coming, then a blank line:
          client.println("HTTP/1.1 200 OK");
          client.println("Content-type:text/html");
          client.println("Connection: close");
```

```
client.println();

// Display the HTML web page
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:,\">");
// CSS to style the on/off buttons
client.println("<style>body { text-align: center; font-family: 'Trebuchet MS', Arial; background-color: #f7f7f7; color: #333; margin-left: auto; margin-right: auto; padding: 0 10px; }");
client.println(".button { display: inline-block; padding: 15px 25px; font-size: 24px; cursor: pointer; text-align: center; text-decoration: none; outline: none; color: #fff; background-color: #4CAF50; border: none; border-radius: 15px; box-shadow: 0 9px #999; margin: 10px; }");
client.println(".button:hover { background-color: #3e8e41 }");
client.println(".button:active { background-color: #3e8e41; box-shadow: 0 5px #666; transform: translateY(4px); }");
client.println(".button-red { background-color: #f44336; }");
client.println(".button-red:hover { background-color: #da190b; }");
client.println(".button-red:active { background-color: #da190b; box-shadow: 0 5px #666; transform: translateY(4px); }");
client.println(".container { background-color: #fff; padding: 20px; border-radius: 10px; box-shadow: 0 2px 4px rgba(0,0,0,0.1); display: inline-block; margin-top: 50px; }");
client.println("h1 { color: #5D6D7E; }");
client.println("</style>");
client.println("</head>");
client.println("<body>");
client.println("<div class=\"container\">");
client.println("<h1>OmniLock</h1>");
client.println("<button class=\"button\" onclick=\"sendData('open')\">Open</button>");
client.println("<button class=\"button button-red\" onclick=\"sendData('close')\">Close</button>");
client.println("</div>");
client.println("<script>");
client.println("function sendData(action) {");
client.println("  var xhttp = new XMLHttpRequest();");
client.println("  xhttp.open('GET', '/' + action, true);");
```

```
            client.println("  xhttp.send();");
            client.println("  if (action == 'open') { alert('Door is Opened'); }");
            client.println("  if (action == 'close') { alert('Door is Closed'); }");
            client.println("}");
            client.println("</script>");
            client.println("</body>");
            client.println("</html>");

            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // If you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // If you got anything else but a carriage return character,
          currentLine += c;      // Add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

4.1.5 Integration of Fingerprint, Keypad, and Remote Access via Web Server

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
#include <Servo.h>

#define Password_Length 5

#define servoPin 13 // Define pin number to which servo motor is connected
#define durationTime 3000 // Define the time it remains in the open position of the door lock (milliseconds)
#define servoMin 0 // Open position
#define servoMax 90 // Closed position

SoftwareSerial mySerial(9, 12);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
Servo servo1; // Define servo name / object
LiquidCrystal_I2C lcd(0x27, 16, 2); // Change the address to match your I2C address

char Data[Password_Length];
char Master[Password_Length] = "5555";
byte data_count = 0;
bool door = false;

const int buzzer = 8;

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] = {7, 6, 5, 4};
byte colPins[COLS] = {3, 2, 1, 0};
Keypad customKeypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  pinMode(buzzer, OUTPUT);
```

```
  servo1.attach(servoPin); // Define pin number of the servo
  servo1.write(servoMax); // The position of the servo at the start of the program

  lcd.init(); // Initialize the I2C LCD
  lcd.backlight(); // Turn on the backlight

  lcd.print("Protected Door");
  loading("Loading");
  lcd.clear();

  Serial.begin(9600);
  Serial.println("Adafruit finger detect test");
  // 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);
  }
  Serial.println("Waiting for valid finger...");
}

void loop() {
  if (door == true) {
    char customKey = customKeypad.getKey();
    if (customKey == '#') {
      lcd.clear();
      ServoClose();
      lcd.print("Door is closed");
      delay(3000);
      door = false;
    }
  } else {
    char fingerprintID = getFingerprintIDez();
```

```
    if (fingerprintID != -1) {
      lcd.clear();
      ServoOpen();
      lcd.print("Door is Open");
      door = true;
      delay(5000);
      loading("Waiting");
      lcd.clear();
      lcd.print("Time is up!");
      lcd.print("Door is closed");
      delay(1000);
      ServoClose();
      door = false;
    } else {
      Open();
    }
  }
}

void loading(char msg[]) {
  lcd.setCursor(0, 1);
  lcd.print(msg);

  for (int i = 0; i < 9; i++) {
    delay(1000);
    lcd.print(".");
  }
}

void clearData() {
  while (data_count != 0) {
    Data[data_count--] = 0;
  }
  return;
}
```

```
void ServoClose() {
  servo1.write(servoMax); // Closed position
}

void ServoOpen() {
  servo1.write(servoMin); // Open position
}

void Open() {
  lcd.setCursor(0, 0);
  lcd.print("Enter Password or Scan Fingerprint:");

  char customKey = customKeypad.getKey();
  if (customKey) {
    Data[data_count] = customKey;
    lcd.setCursor(data_count, 1);
    lcd.print(Data[data_count]);
    data_count++;
  }

  if (data_count == Password_Length - 1) {
    if (!strcmp(Data, Master)) {
      lcd.clear();
      ServoOpen();
      lcd.print(" Door is Open ");
      door = true;
      delay(5000);
      loading("Waiting");
      lcd.clear();
      lcd.print(" Time is up! ");
      lcd.print("Door is closed");
      delay(1000);
      ServoClose();
      door = false;
```

```
    } else {
      lcd.clear();
      lcd.print(" Wrong Password ");
      lcd.print("Access Denied");
      activateBuzzer(); // Access Denied, activate buzzer
      door = false;
    }
    delay(1000);
    lcd.clear();
    clearData();
  }
}

void activateBuzzer() {
  tone(buzzer, 1000); // Activate the buzzer
  delay(1000);        // Buzz for 1 second
  noTone(buzzer);     // Stop the buzzer
}

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");
      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");
    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!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
  Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Did not find a match");
  return p;
```

```
  } else {
    Serial.println("Unknown error");
    return p;
  }
  // found a match!
  Serial.print("Door is Open");
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
}

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) {
    activateBuzzer(); // Access Denied, activate buzzer
    return -1;
  }
  ServoOpen(); // If the fingerprint is correct open the door lock
  delay(durationTime); // Keep the lock open for the defined duration
  ServoClose(); // take the lock OFF again
  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
#include <WiFi.h>
#include <ESP32Servo.h>

Servo myservo;  // Create a servo object to control a servo

// GPIO the servo is attached to
static const int servoPin = 13;
```

```
// Replace with your network credentials
const char* ssid = "Laptop";
const char* password = "#dev11oc04";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Decode HTTP GET value
int servoPosition = 0;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200); // Set baud rate to 115200 for the serial monitor

  myservo.attach(servoPin); // Attaches the servo on the servoPin to the servo object

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  // Improved connection feedback
  unsigned long startTime = millis();
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
```

```
    Serial.print(".");
    // Optional: You can add a timeout for connection attempts
    if (millis() - startTime > 10000) { // 10 seconds timeout
      Serial.println("\nFailed to connect to WiFi, restarting...");
      ESP.restart(); // Restart the ESP32
    }
  }

  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop() {
  WiFiClient client = server.available(); // Listen for incoming clients

  if (client) { // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client."); // Print a message out in the serial port
    String currentLine = ""; // Make a String to hold incoming data from the client
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // Loop while the client's
connected
      currentTime = millis();
      if (client.available()) { // If there are bytes to read from the client,
        char c = client.read(); // Read a byte, then
        Serial.write(c); // Print it out the serial monitor
        header += c;
        if (c == '\n') { // If the byte is a newline character
          // If the current line is blank, you got two newline characters in a row.
          // That's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
```

```
// Parse GET request before sending the response
bool servoActionTaken = false;
String action;
if (header.indexOf("GET /open") >= 0) {
  servoPosition = 0; // Change to the closed position
  myservo.write(servoPosition);
  Serial.println("Door Opened");
  servoActionTaken = true;
  action = "open";
}
if (header.indexOf("GET /close") >= 0) {
  servoPosition = 90; // Change to the open position
  myservo.write(servoPosition);
  Serial.println("Door Closed");
  servoActionTaken = true;
  action = "close";
}

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
// and a content-type so the client knows what's coming, then a blank line:
client.println("HTTP/1.1 200 OK");
client.println("Content-type:text/html");
client.println("Connection: close");
client.println();

// Display the HTML web page
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:,\">");
// CSS to style the on/off buttons
client.println("<style>body { text-align: center; font-family: 'Trebuchet MS', Arial; background-color: #f7f7f7; color: #333; margin-left: auto; margin-right: auto; padding: 0 10px; }");
```

```
client.println(".button { display: inline-block; padding: 15px 25px; font-size: 24px; cursor: pointer; text-align: center; text-decoration: none; outline: none; color: #fff; background-color: #4CAF50; border: none; border-radius: 15px; box-shadow: 0 9px #999; margin: 10px; }");
client.println(".button:hover { background-color: #3e8e41 }");
client.println(".button:active { background-color: #3e8e41; box-shadow: 0 5px #666; transform: translateY(4px); }");
client.println(".button-red { background-color: #f44336; }");
client.println(".button-red:hover { background-color: #da190b; }");
client.println(".button-red:active { background-color: #da190b; box-shadow: 0 5px #666; transform: translateY(4px); }");
client.println(".container { background-color: #fff; padding: 20px; border-radius: 10px; box-shadow: 0 2px 4px rgba(0,0,0,0.1); display: inline-block; margin-top: 50px; }");
client.println("h1 { color: #5D6D7E; }");
client.println("</style>");
client.println("</head>");
client.println("<body>");
client.println("<div class=\"container\">");
client.println("<h1>OmniLock</h1>");
client.println("<button class=\"button\" onclick=\"sendData('open')\">Open</button>");
client.println("<button class=\"button button-red\" onclick=\"sendData('close')\">Close</button>");
client.println("</div>");
client.println("<script>");
client.println("function sendData(action) {");
client.println("  var xhttp = new XMLHttpRequest();");
client.println("  xhttp.open('GET', '/' + action, true);");
client.println("  xhttp.send();");
client.println("  if (action == 'open') { alert('Door is Opened'); }");
client.println("  if (action == 'close') { alert('Door is Closed'); }");
client.println("}");
client.println("</script>");
client.println("</body>");
client.println("</html>");

// The HTTP response ends with another blank line
client.println();
```

```
          // Break out of the while loop
          break;
        } else { // If you got a newline, then clear currentLine
          currentLine = "";
        }
      } else if (c != '\r') {  // If you got anything else but a carriage return character,
        currentLine += c;      // Add it to the end of the currentLine
      }
    }
  }
  // Clear the header variable
  header = "";
  // Close the connection
  client.stop();
  Serial.println("Client disconnected.");
  Serial.println("");
 }
}
```

4.2 Execution Flow

4.2.1 User Interaction:

- The execution flow begins when a user approaches a secured area and interacts with the Omnilock system, typically through a user interface such as a touchscreen or keypad.

4.2.2 Access Request:

- The user initiates the access request by providing their credentials, which may include biometric data (e.g., fingerprint), or a personal identification number (PIN).

4.2.3 Input Processing:

- Omnilock's access controller receives and processes the input provided by the user, which may involve capturing biometric data, validating PINs, or through remote access.

4.2.4 Authentication:

- The system authenticates the user's credentials using the appropriate authentication method(s) configured for the access control system.

- Biometric authentication involves matching the captured biometric data with stored templates or reference data.
- PIN authentication verifies the correctness of the entered PIN against the stored user credentials.

4.2.5 Access Control Decision:

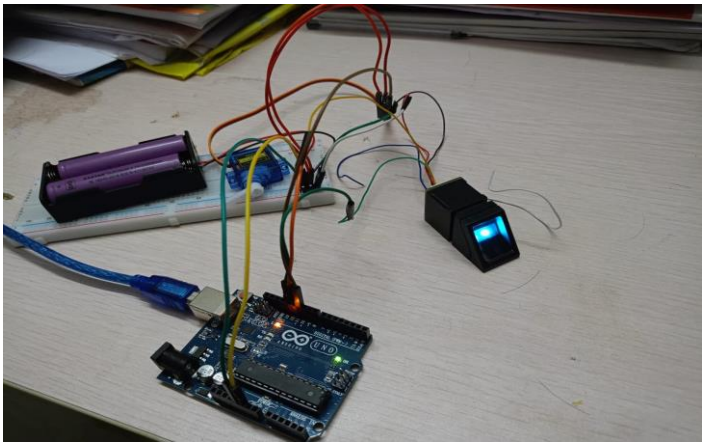
- Based on the authentication results, the system makes an access control decision to either grant or deny access to the secured area.
- If the user's credentials are successfully authenticated, access is granted, and the user is allowed entry.
- If authentication fails or if the user's credentials do not meet the access control criteria, access is denied, then the buzzer is triggered and the user is prevented from entering the secured area.

4.2.6 Logging and Notification:

- The system logs the access control event, recording details such as the user's identity, the outcome of the access control decision, and the timestamp of the event.
- Notifications or alerts may be generated to inform administrators or security personnel about access control events, especially in the case of unauthorized access attempts or security breaches.

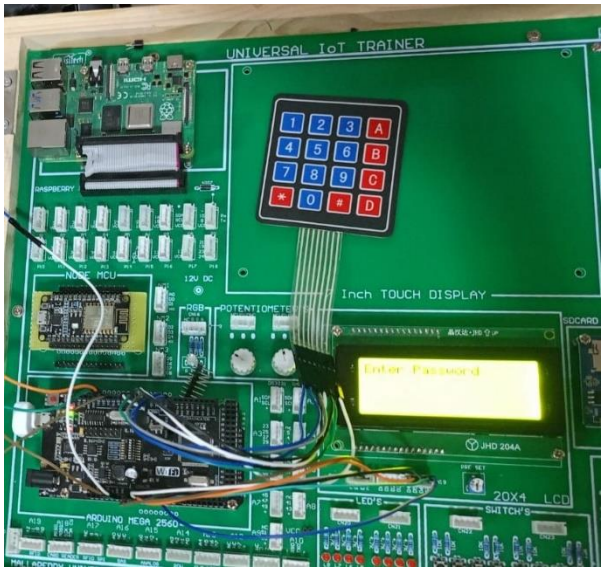
4.3 TESTING

4.3.1 Test case 1: Fingerprint



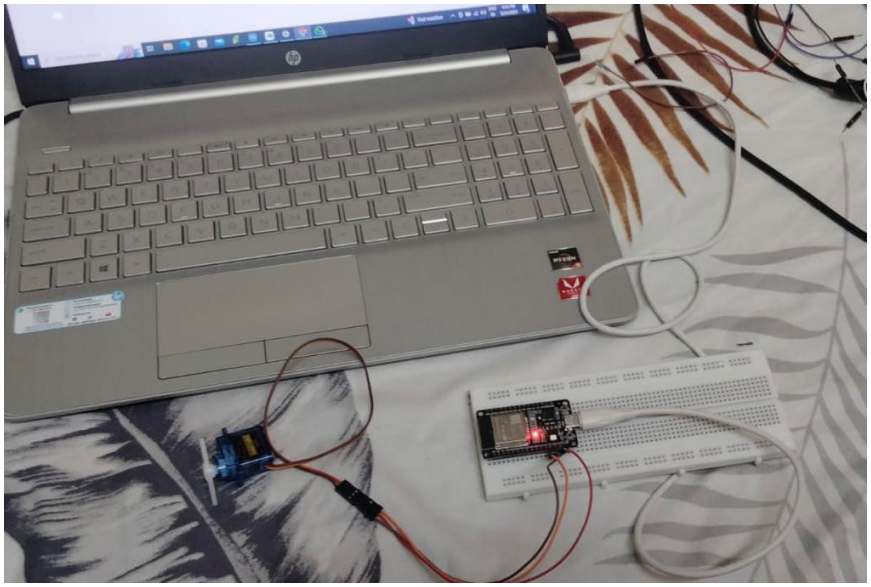
Screenshot 4.3.1 Door unlocking using Fingerprint

4.3.2 Test case 2: Keypad

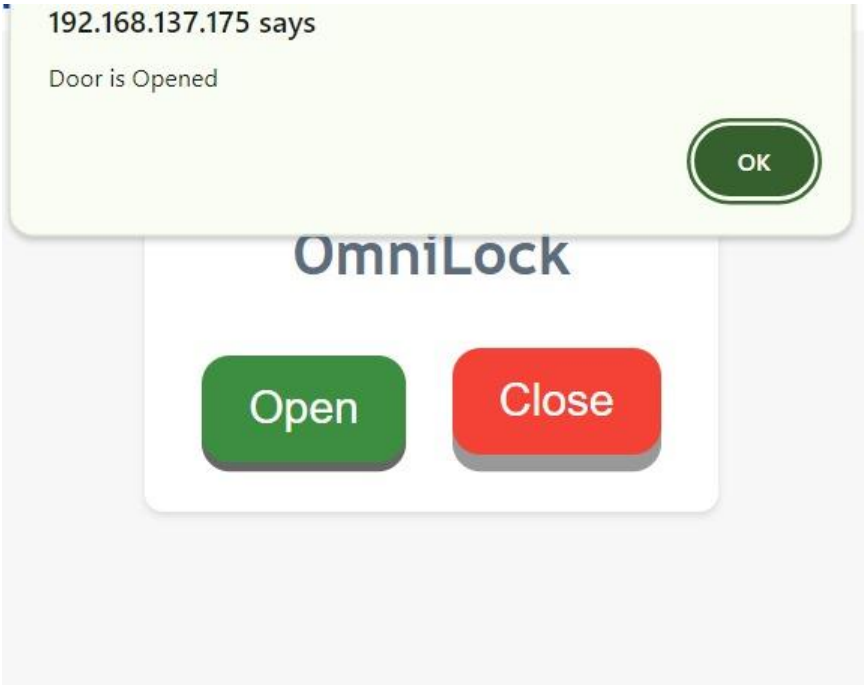


Screenshot 4.3.2 Door unlocking using Keypad

4.3.3 Test Case 3: Remote Access



Screenshot 4.3.3 ESP32 connections with servo

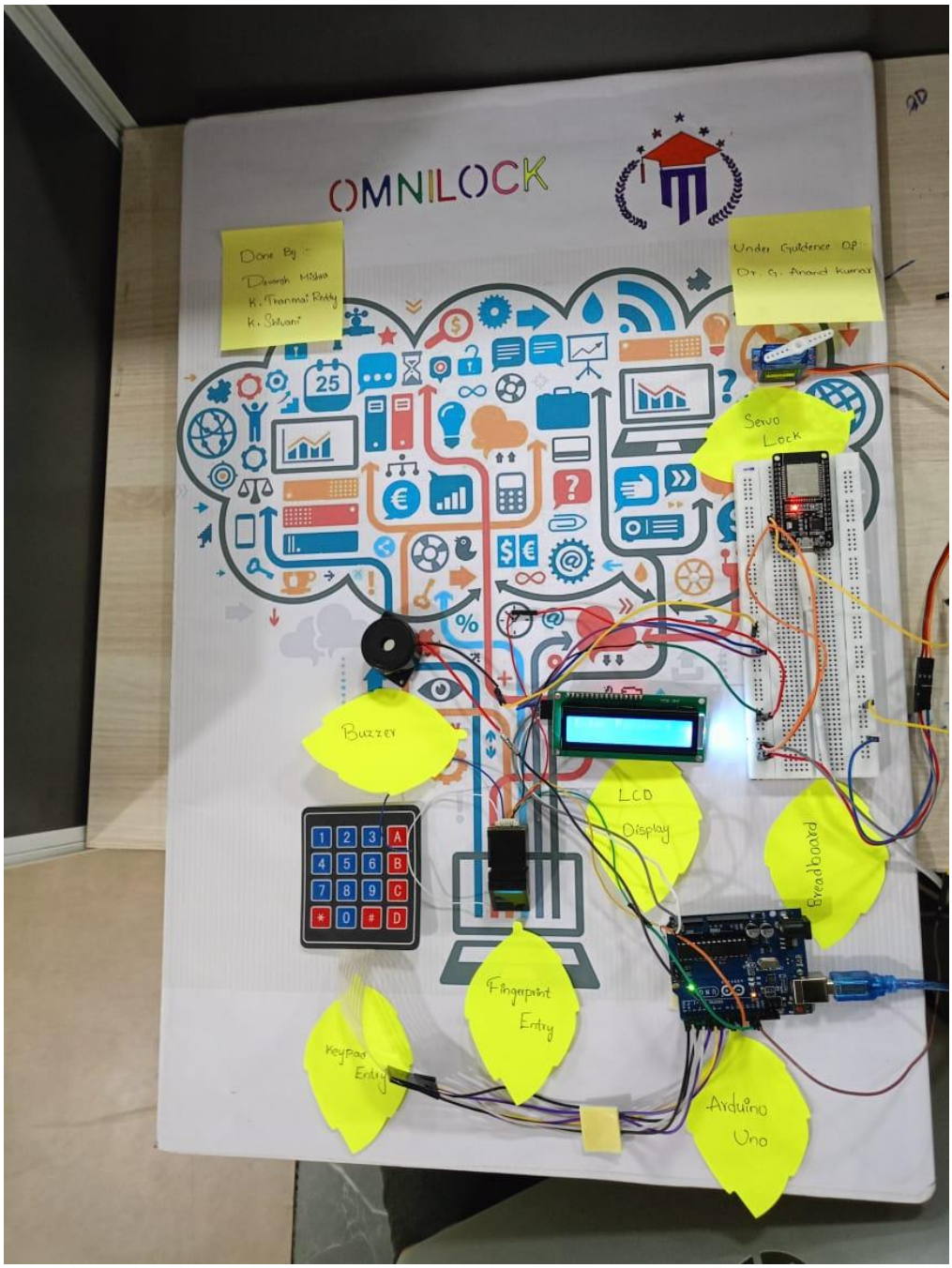


Screenshot 4.3.3 Website for Remote Access

CHAPTER-5

RESULTS

5.1 Resulting Screen



Screenshot 5.1 Integration of Fingerprint, Keypad, and Remote Access

CHAPTER-6

CONCLUSIONS & FUTURE SCOPE

6.1 Conclusions

By overcoming the drawbacks of conventional keys and single-factor authentication, Omnilock transforms access control. With its multi-modal approach, customers can select their favorite technique, be it a fingerprint scanner or secure keypad. Compared to readily misplaced, stolen, or duplicated keys or RFID cards that are susceptible to interception, this greatly improves security. Another level of comfort comes with the possibility of remote access control provided through the web server. Omnilock markets itself as a user-friendly, flexible, and safe access control system. With the use of potent microcontrollers, state-of-the-art biometrics, and the possibility of smart home integration, Omnilock users are in a new era of smart and practical door locks for homes and other locations.

6.2 Future Scope

Encouraging possibilities exist for enhanced security and user-friendliness in the future. Visualize a scenario where high-security regions require access to both a PIN code and a fingerprint scan as part of multi-factor authentication. When you get closer, geofencing might open the door for you, and time-based access might only allow certain users in during particular times. Voice authentication may even be made available as a convenient hands-free option.

BIBLIOGRAPHY

[1] Anu and Bhatia, “A smart door access system using fingerprint biometric system” International Journal of Medical Engineering and Informatics www.inderscience.com Volume No.06, Issue No.03,7 July 2014.

[2] Arpita Mishra, Siddharth Sharma, Sachin Dubey, S. K. Dubey, “PASSWORD BASED SECURITY LOCK SYSTEM” International Journal of Advanced Technology in Engineering and Science www.ijates.com Volume No.02, Issue No. 05, May 2014 ISSN (online): 2348 – 7550.

[3] Dr. M. SivaSangari, Dhivakar. E, Gowthaam. K, “Secret Knock Detecting Door Lock” Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 406-410 Received 15 April 2021; Accepted 05 May 2021.

[4] Adarsh V Patil, Sreevarsha Prakash, Akshay S, Mahadeva Swamy, Chandan B Patgar, Sharath Kumar A J, “Android Based Smart Door Locking System” International Journal of Engineering Research &

Technology (IJERT) ISSN: 2278-0181 Published by, www.ijert.org NCESC - 2018 Conference Proceedings.

[5] K. Rajesh, ASST. PROFESSOR, B. Venkata Rao, P. AV. S. K. Chaitanya, A. Ruchitha Reddy, “SMART DOOR UNLOCK SYSTEM USING FINGERPRINT” Pramana Research Journal Volume 9, Issue 3, 2019 ISSN NO: 2249-2976.

[6] A.Vadakkan, A.Babu.V.K and C.Pappachan, "DOOR LOCKING USING KEYPAD AND ARDUINO," International Research Journal of Modernization in Engineering Technology and Science, vol. 3, no. 11, p. 783, 2021.

[7] Park, Yong Tae, Pranesh Sthapit, and Jae-Young Pyun. "Smart digital door lock for the home automation." In TENCON 2009-2009 IEEE Region 10 Conference, pp. 1-6. IEEE, 2009.

[8] Kassem, Abdallah, Sami El Murr, Georges Jamous, Elie Saad, and Marybelle Geagea. "A smart lock system using Wi-Fi security." In Advances in Computational Tools for Engineering Applications (ACTEA), 2016 3rd International Conference on, pp. 222-225. IEEE, 2016.

[9] Manurung, M. J., Poningsih, P., Andani, S. R., Safii, M., & Irawan, I. (2021). “Door Security Design Using Fingerprint and Buzzer Alarm Based on Arduino”. *Journal of Computer Networks, Architecture and High Performance Computing*, *3*(1), 42-51.

[10] LiaKamelia, AlfinNoorhassan S.R, MadaSanjaya and W.S., Edi Mulyana, “DOOR-AUTOMATION SYSTEM USING BLUETOOTH-BASED ANDROID FOR MOBILE PHONE” VOL. 9, NO. 10, OCTOBER 2014 ISSN 1819-6608 ARPN Journal of Engineering and Applied Sciences.

PAPER PUBLICATION

OMNILOCK-SMART DOOR LOCKING SYSTEMS

Dr. G Anand Kumar
Computer Science & Engineering
Dept. of Cyber Security & IoT
Malla Reddy University
Hyderabad, India
anandlife@gmail.com

Dr. M V N Srujan Manohar
Computer Science & Engineering
Dept. of Cyber Security & IoT
Malla Reddy University
Hyderabad, India
srujanmanohar.edu@gmail.com

Devansh Mishra
Computer Science & Engineering
Dept. of IoT
Malla Reddy University
Hyderabad, India
2211cs050023@mallareddyuniversity.ac.in

Kalla Thanmai Reddy
Computer Science & Engineering
Dept. of IoT
Malla Reddy University
Hyderabad, India
2211cs050040@mallareddyuniversity.ac.in

Kancharla Shivani
Computer Science & Engineering
Dept. of IoT
Malla Reddy University
Hyderabad, India
2211cs050046@mallareddyuniversity.ac.in

Abstract— **By integrating cutting-edge biometrics with the capabilities of the Internet of Things (IoT), Omnilock enhances access control. This creative solution uses an Arduino microcontroller and ESP32 combination to produce a reliable, easy-to-use smart lock application. With its integration of touchpad entry, fingerprint authentication, and remote access through the Web server. Omnilock goes beyond traditional methods to provide unmatched security and convenience. Each feature of the system may operate independently due to its modular architecture, which guarantees optimal performance and a consistent user interface. Omnilock places security first, users can choose from secure login methods like a fingerprint sensor or a simple touchpad for convenient access. Additionally, Omnilock offers remote access, allowing you to unlock the door for specified individuals even when you are not physically present. A buzzer that sounds when entry is denied serves as a barrier to unwanted attempts. In the end, Omnilock bills itself as a safe access control system that can be used in homes and other locations, giving consumers a smooth and adaptable interface.**

Keywords—Arduino, ESP32, fingerprint authentication, touchpad, remote access, buzzer

I. Introduction

These days security is a major threat faced by every individual when away from home or at home. The crime of theft at home often occurs from time to time this happens because the house is often inhabited or empty by the owner so the house becomes the main target for thieves [9]. And when it comes to security frameworks, it is one of the essential concerns in this active competitive world, where humans cannot discover ways to supply security to his/her private possessions physically. Besides, the traditional door lock uses a key that can easily be opened by an unauthorized person if he/she has the right key or a duplicate [1]. There are some more hassles that people might face when using keys or smart cards. That is when our Omnilock door lock system comes into play. Adarsh V Patil highlighted that their system works on pre-decided password concepts. It increases the security level to prevent an unauthorized unlocking done by an attacker. In case the user forgets the password, the system gives the flexibility to the user to change or reset the password [4]. Our design is implemented to provide better security as users get convenient and secure access choices, such as fingerprint scanning, and keypad entry. Furthermore, remote access features for convenience. Moreover, it doesn't need any sort of keys or cards that often get lost. Omnilock elevates your security from traditional locks to a state-of-the-art access control solution. It seamlessly integrates cutting-edge technologies with user-friendly design, offering unparalleled security and convenience. With Omnilock, your door becomes a sophisticated guardian instead of just a basic keyhole. Instead of using standard keys, which are easy to steal, misplace, or duplicate, this smart lock uses fingerprint and keypad authentication. The human detection and recognition field is very significant and has undergone rapid changes with time. An important and very reliable human identification method is fingerprint identification [5]. When access is refused, an audible buzzer sounds to discourage unwanted attempts. In contrast to RFID, which is clone-prone, Omni lock's biometric method provides more security. Additionally, the servo motor ensures flawless operation with your current door, saving you money on changes. The primary objective of the Omnilock project is to develop an advanced Smart Door Lock System that leverages cutting-edge biometric technologies and the capabilities of the ESP32 and Arduino microcontroller to provide unparalleled security and convenience. Omnilock aims to offer robust identity verification by integrating fingerprint scanning and keypad entry, significantly reducing the risk of unauthorized access. Moreover, Omnilock enables remote access for user convenience. The project seeks to address the shortcomings of traditional access control methods by delivering a comprehensive solution that is both secure and user-friendly. At its core, a high-accuracy fingerprint sensor provides quick and reliable individual identification, eliminating the need for physical keys. For those who prefer a traditional approach or situations where fingerprint scanning isn't ideal, a secure keypad interface allows access using a user-defined PIN code. Furthermore, remote access through the web server provides total control over the lock even when you're not there. The powerful ESP32 microcontroller serves as the system's brain, managing data processing, communication, and general control.

II. Literature Survey

[1] "Property Security Using a Biometric Based Door Lock System" was proposed by OnyanA and Enalume K. The paper is about the creation and implementation of a Biometric Based Door Lock System, which will automatically unlock a door when a registered fingerprint is detected. To achieve this, a fingerprint scanner R305 is used in conjunction with an ATMEGA 328 Arduino microcontroller to monitor the locking and unlocking process of a door. Access is given to the user after a recorded fingerprint is put on the sensor, the door slides open, and it closes after five seconds. The 16x2 Liquid Crystal Display (LCD) shows the individual's name along with the registered fingerprint during this process. Access is refused if an unregistered fingerprint is detected.

[2] A paper named "Secured password-based lock system" was put forward by "Arpita Mishra, Siddharth Sharma, Sachin Dubey, and S.K. Dubey". This strategy is focused on avoiding the opening of the entryway by obscure people. The arrangement of the domestic security benefit comprises the numeric keypad, the snare which is utilized for lifting, and a GSM module to set up reliable association for communication conferred with the MCU. The control board conferred with the gadget is utilized since the passcode gets to combination opens/closes the entryway.

[3] "A locks which operate by a secret knock" which was put forward by "Dr.M.Siva Sangari, Dhivakar. E, Gowthaam. K". To open and close the lock by burrowing out the thumps detected by the piezo sensor to open the entryway, a servo turns when it identifies the thumps and opens and closes the entryway.

[4] "Android Based Smart Door Locking System" which was proposed by "Adarsh V Patil, Sreevarsha Prakash, Akshay S, Mahadevaswamy, Chandan B Patgar, Sharath Kumar A J". The method executes on a pre-coded passcode idea. It moves forward the security prepares to halt an unathletically pulverizing done by a false.

[5] "SMART DOOR UNLOCK SYSTEM USING FINGERPRINT" was proposed by K. Rajesh, Asst.Prof. B.VenkataRao, P.AV.S.K.Chaitanya, and A.Ruchitha Reddy. In this paper, they have used a fingerprint sensor to read one's identity to operate the door of the car automatically. They used a microcontroller to enable the door to open or close if the matching between scanned data and the existing data is correct. Comparison is done inside the fingerprint module itself and its output is given to the microcontroller. The result is displayed on an LCD whether the user is authorized or not. LCD also helps to make troubleshooting easier. The alarming option is provided to warn about unauthorized usage. The microcontroller used is PIC16F877.

[6] "DOOR LOCKING USING KEYPAD AND ARDUINO" was put forward by Annmary Vadakkan, Athulya Babu V K, Christy Pappachan, and Prof. Ani Sunny. Password-based door lock systems allow authorized persons to access restricted areas in their security pattern. It is controlled by an Arduino. The password is entered using a keypad. The entered password is compared with the known password when setting a combination password and by using 1-6 digits. A correct password opens the door and displays the status on the LCD. If the password is wrong, the door remains closed and states "WRONG PASSWORD" on LCD. The buzzer will also activate when a password is entered incorrectly an indefinite number of times.

[7] "Smart Digital Door Lock for the Home Automation" was proposed by Yong Tae Park, Pranesh Sthapit, and Jae-Young Pyun. In this paper, they proposed a smart digital door lock system for home automation. In their proposed system, a ZigBee module is embedded in the digital door lock and the door lock acts as a central main controller of the overall home automation system. This door lock system proposed here consists of an RFID reader for user authentication, a touch LCD, a motor module for door opening and closing, sensor modules for detecting the condition inside the house, a communication module, and a control module for controlling other modules.

[8] In the paper "IoT and Fingerprint Based Door Looking System" which was proposed by Prof. Mohammad Nasirruddin, Prerna Kachhwaha, Ankita Balpande, Payal Bondre, and Mrunal Gawande. This door-locking system project suggests unlocking a door using a fingerprint. One of the most reliable biometric features having a wide range of applications is fingerprint. It provides tools to enforce reliable system transaction logs as well as protect an individual's privacy. To provide access to the facility used by multiple users, the fingerprints of the authorized users are enrolled and verified. A new user can be enrolled as well and an old user can also be removed from the system.

[9] A paper named "Door Security Design Using Fingerprint and Buzzer Alarm Based on Arduino" was put forward by Mario Junianto Manurung, Poningsih, Sundari Retno Andani, Muhammad Safii, and Irawan. In this paper they used a microcontroller Arduino as the main control center, assisted by a fingerprint sensor as a process for recording and identifying fingerprints and NodeMCU to connect to a wifi network where later on the door can also be controlled using Android via a wifi network as a remote control and an alarm buzzer which will function if when the fingerprint identification process fails, the alarm will sound as a warning sign, it will be concluded to design a safety device.

[10] "DOOR-AUTOMATION SYSTEM USING BLUETOOTH-BASED ANDROID FOR MOBILE PHONE" proposed by Lia Kamelia, Alfin Noorhassan S.R, Mada Sanjaya and W.S., and Edi Mulyana. In this paper, a system called door locks automation system using Bluetooth-based Android smartphones is proposed and prototyped. First, the hardware design and software development are described, and then the design of a Bluetooth-based Smartphone application for locking/unlocking the door is presented. The hardware design for the door-lock system is the combination of an Android smartphone as the taskmaster, a Bluetooth module as the command agent, an Arduino microcontroller as the controller center/data processing center, and a solenoid as the door lock output.

III. System Analysis

A. Existing System

Traditional door locks that just require physical keys are vanishing from use. Their weaknesses are many: keys are easily lost, stolen, or even duplicated, putting your house or place of business's security at risk. A stolen key can provide an unauthorized entrance, while a lost key can leave you stranded outside. Sharing access also becomes problematic because it frequently necessitates giving out a physical key, which is dangerous and inconvenient, particularly for temporary visitors or service providers. Moreover, carrying around several keys is inconvenient since it adds extra weight to your pockets and causes you to constantly worry about losing them.

Traditional locks and keycards can be inconvenient or difficult for people with disabilities. There's no way to track who accessed the door and when with a traditional lock, making it difficult to monitor activity. Despite appearing to be a benefit, RFID technology raises additional issues. Because they rely on radio frequency transmissions, they can be copied, which could lead to unauthorized entry via intercepted communications.

Omnilock provides a multi-modal authentication system that balances ease and security, surpassing the drawbacks of conventional key-based access management. Physical keys are rendered obsolete by fingerprint scanners and secure keypads. The chance of physical keys being misplaced, stolen, or duplicated is eliminated, which dramatically lowers the risk of illegal access. Omnilock's possible interaction with the web server enables remote unlocking and access management. This improves security and peace of mind by enabling real-time entryway monitoring. When access is refused, a loud alarm also sounds to discourage unwanted attempts. More control and security are ensured by doing away with the need to issue actual keys, particularly for temporary access needs like deliveries or service visits.

B. Proposed System

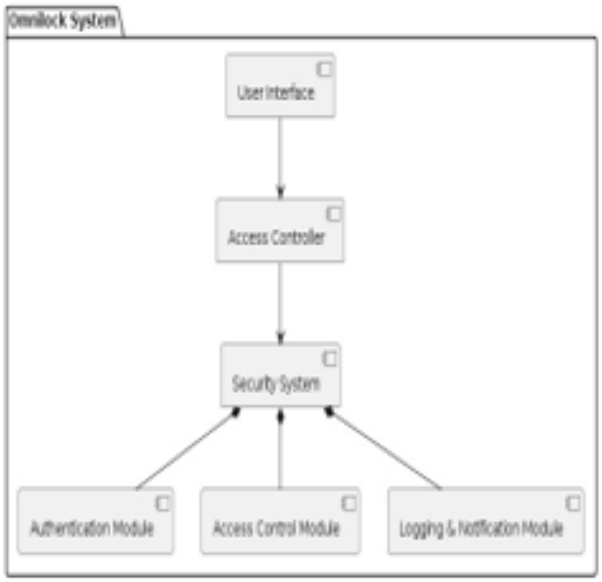
Omnilock provides a multi-layered solution to access management, addressing the drawbacks of standalone biometric systems, RFID, and classic locks. Security can be compromised by physical keys, which are readily lost or stolen. By using biometrics (fingerprint technology) and secure keypads, Omnilock removes this risk and increases the difficulty of unwanted entry using lost or stolen credentials. Despite their ease, RFID technologies have a serious security vulnerability. Since they communicate by radio waves, they are vulnerable to copying and interception. With the correct tools, a malicious actor might be able to make a duplicate RFID card and obtain unauthorized access. Omnilock directly addresses this problem. Because biometrics are unique to each person, they greatly lessen the possibility of unwanted access through credential theft. The system may become inoperable due to a broken fingerprint scanner. The multi-modal approach of Omnilock provides users with additional techniques (remote access, keypad, or fingerprint) for user authentication, ensuring access even if one component malfunctions. The ESP32 may be utilized for remote access. However, its processing power and security features may be insufficient for the best results. Using an Arduino's advantages in communication and control, Omnilock combines the ESP32 with it to create a more robust solution.

Advantages

- The primary advantage of Omnilock is its variety of authentication options. This makes it far more difficult than with single-factor systems for unauthorized people to obtain access.
- An increased degree of security is offered by biometric verification as opposed to conventional keys or RFID cards. Biometric authentication provides an additional degree of security, even if a user's PIN code is stolen.
- Depending on their comfort level and circumstances, users can select their preferred authentication method. While fingerprint gives a possibly speedier method, the keypad offers a more comfortable option. Even if a user is not available then they can use the remote access.
- Future integration with other technologies, such as smart home systems or innovative safety features, is made possible by the modular design.

IV. Methodology

A. Architecture



Algorithms Implemented: Fingerprint Verification and String Comparison.

Fingerprint Verification Algorithm:

The technology used in fingerprint sensors does more than just take a picture of your fingers. The actual magic of user identification is carried out by complex algorithms that are included in these sensors. These exclusive algorithms function directly on the sensor, guaranteeing a safe and effective procedure. Since the sensor vendor closely guards the specifics of these algorithms, they are frequently private. Because it becomes harder for potential hackers to exploit the underlying workings of the algorithms, this strategy maintains security. These sensor-specific algorithms also take advantage of the hardware's capabilities to optimize processing times and power consumption. The sensor's

fingerprint verification algorithms serve as the gatekeepers, carefully examining each characteristic of your fingerprint to provide safe and practical user identification.

String Comparison Algorithm:

PIN code protection is just one more way that Omnilock puts the security of its users first. An important vulnerability would be a straightforward string comparison between the entered PIN and a stored PIN code. A hacker might simply open the door if they managed to get the stored PIN codes. Omnilock uses hashing functions to solve this issue. These cryptographic functions create a unique, fixed-length string of characters termed a hash using the user's PIN code as input. The original PIN is securely represented by this hash. It's crucial to remember that the hashing function is one-way; the original PIN code cannot be recovered by mathematically reversing the hash.

B. Process

1) Module 1: User Interface Module

- Responsible for giving an interface for clients to associate with the system.
- Includes functionalities such as user authentication, access request submission, and viewing system status.
- Designed with a user-friendly interface for ease of use and accessibility.

2) Module 2: Access Controller Module

- Manages access requests from users and coordinates the authentication process.
- Acts as an intermediary between the user interface and the security system.
- Responsible for initiating the authentication process and handling access control decisions based on authentication results.

3) Module 3: Security System Module

- The central component is responsible for authentication and access control.
- Consists of sub-modules for various authentication methods such as fingerprint recognition, remote access, and PIN authentication.
- Implements access control policies based on authentication results and grants or denies access accordingly.

4) Module 4: Authentication Module

- Handles the authentication process for verifying the identity of users.
- Includes sub-modules for different authentication methods, such as biometric authentication (fingerprint technology), traditional methods (PIN authentication), and remote access via web server.

5) Module 5: Access Control Module

- Implements access control policies based on authentication results.
- Determines whether users are granted or denied access to secured areas based on authentication outcomes.
- Interfaces with the security system to enforce access control decisions and manage access permissions.
- The buzzer can be activated for a predetermined duration by a function that the Access Control Module may trigger in response to a "denied access" signal.

6) Module 6: Logging & Notification Module

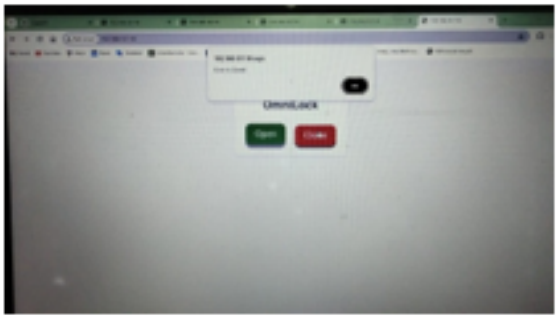
- Responsible for logging system events and generating notifications/alerts.
- Records access control activities, authentication attempts, and system errors for audit and monitoring purposes.
- Notifies administrators or users about security-related events, such as unauthorized access attempts or system malfunctions.

V. Results

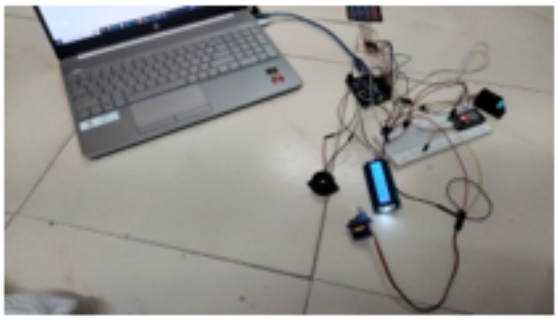
A. Figure 1: Integration with Fingerprint sensor

B. Figure 2: Integration of Keypad

C. *Figure 3: Remote Access*



D. *Figure 4: Integration of Fingerprint, Keypad, and Remote Access*



VI. Conculsion

By overcoming the drawbacks of conventional keys and single-factor authentication, Omnilock transforms access control. With its multi-modal approach, customers can select their favorite technique, be it a fingerprint scanner or secure keypad. Compared to readily misplaced, stolen, or duplicated keys or RFID cards that are susceptible to interception, this greatly improves security. Another level of comfort comes with the possibility of remote access control provided through the web server. Omnilock markets itself as a user-friendly, flexible, and safe access control system. With the use of potent microcontrollers, state-of-the-art biometrics, and the possibility of smart home integration, Omnilock users are in a new era of smart and practical door locks for homes and other locations.

VII. Future Work

Encouraging possibilities exist for enhanced security and user-friendliness in the future. Visualize a scenario where high-security regions require access to both a PIN code and a fingerprint scan as part of multi-factor authentication. When you get closer, geofencing might open the door for you, and time-based access might only allow certain users in during particular times. Voice authentication may even be made available as a convenient hands-free option.

VIII. References

[1] OnyanA and Enalume K, "Property Security Using a Biometric Based Door Lock System" – 2018.

[2] Arpita Mishra, et al., "PASSWORD BASED SECURITY LOCK SYSTEM" International Journal of Advanced Technology in Engineering and Science www.ijates.com Volume No.02, Issue No. 05, May 2014 ISSN (online): 2348 – 7550.

[3] Dr. M. SivaSangari, et al., "Secret Knock Detecting Door Lock" Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 406-410 Received 15 April 2021; Accepted 05 May 2021.

[4] Adarsh V Patil, et al., "Android Based Smart Door Locking System" International Journal of Engineering Research & Technology (IJERT)

[5] K. Rajesh, et al., "SMART DOOR UNLOCK SYSTEM USING FINGERPRINT" Pramana Research Journal Volume 9, Issue 3, 2019 ISSN NO: 2249-2976.

[6] A.Vadakkan, et al., "DOOR LOCKING USING KEYPAD AND ARDUINO," International Research Journal of Modernization in Engineering Technology and Science, vol. 3, no. 11, p. 783, 2021.

[7] Park, et al., "Smart digital door lock for the home automation." In TENCON 2009-2009 IEEE Region 10 Conference, pp. 1-6. IEEE, 2009.

[8] NasarruddinM, et al., "IOT and fingerprint based door locking system" – 2018.

[9] Manurung, M. J., et al., (2021). "Door Security Design Using Fingerprint and Buzzer Alarm Based on Arduino". *Journal of Computer Networks, Architecture and High Performance Computing*, *3*(1), 42-51.

[10] LiaKamelia, et al., "DOOR-AUTOMATION SYSTEM USING BLUETOOTH-BASED ANDROID FOR MOBILE PHONE" VOL. 9, NO. 10, OCTOBER 2014 ISSN 1819-6608 ARPN Journal of Engineering and Applied Sciences.

ACKNOWLEDGMENT OF IEEE

IEEE MYSORE SUBSECTION FLAGSHIP CONFERENCE MYSURUCON 2024 : Submission (852) has been created.

1 message

Microsoft CMT <email@msr-cmt.org> Wed, Jun 5, 2024 at 11:42
Reply-to: Microsoft CMT - Do Not Reply <noreply@msr-cmt.org>
To: 2211cs050023@mallareddyuniversity.ac.in

Hello,

The following submission has been created.

Track Name: Track-2: Defence and Security

Paper ID: 852

Paper Title: OMNILOCK-SMART DOOR LOCKING SYSTEMS

Abstract:
By integrating cutting-edge biometrics with the capabilities of the Internet of Things (IoT), Omnilock enhances access control. This creative solution uses an Arduino microcontroller and ESP32 combination to produce a reliable, easy-to-use smart lock application. With its integration of touchpad entry, fingerprint authentication, and remote access through the Web server. Omnilock goes beyond traditional methods to provide unmatched security and convenience. Each feature of the system may operate independently due to its modular architecture, which guarantees optimal performance and a consistent user interface. Omnilock places security first, users can choose from secure login methods like a fingerprint sensor or a simple touchpad for convenient access. Additionally, Omnilock offers remote access, allowing you to unlock the door for specified individuals even when you are not physically present. A buzzer that sounds when entry is denied serves as a barrier to unwanted attempts. In the end, Omnilock bills itself as a safe access control system that can be used in homes and other locations, giving consumers a smooth and adaptable interface.

Created on: Wed, 05 Jun 2024 06:12:42 GMT

Last Modified: Wed, 05 Jun 2024 06:12:42 GMT

Authors:
- anandlife@gmail.com (Primary)
- srujanmanohar.edu@gmail.com
- 2211cs050023@mallareddyuniversity.ac.in
- 2211cs050046@mallareddyuniversity.ac.in
- 2211cs050040@mallareddyuniversity.ac.in

Secondary Subject Areas: Not Entered
Submission Files: Omnilock ad_ieee final.pdf (320 Kb, Wed, 05 Jun 2024 06:11:38 GMT)

Submission Questions Response: Not Entered

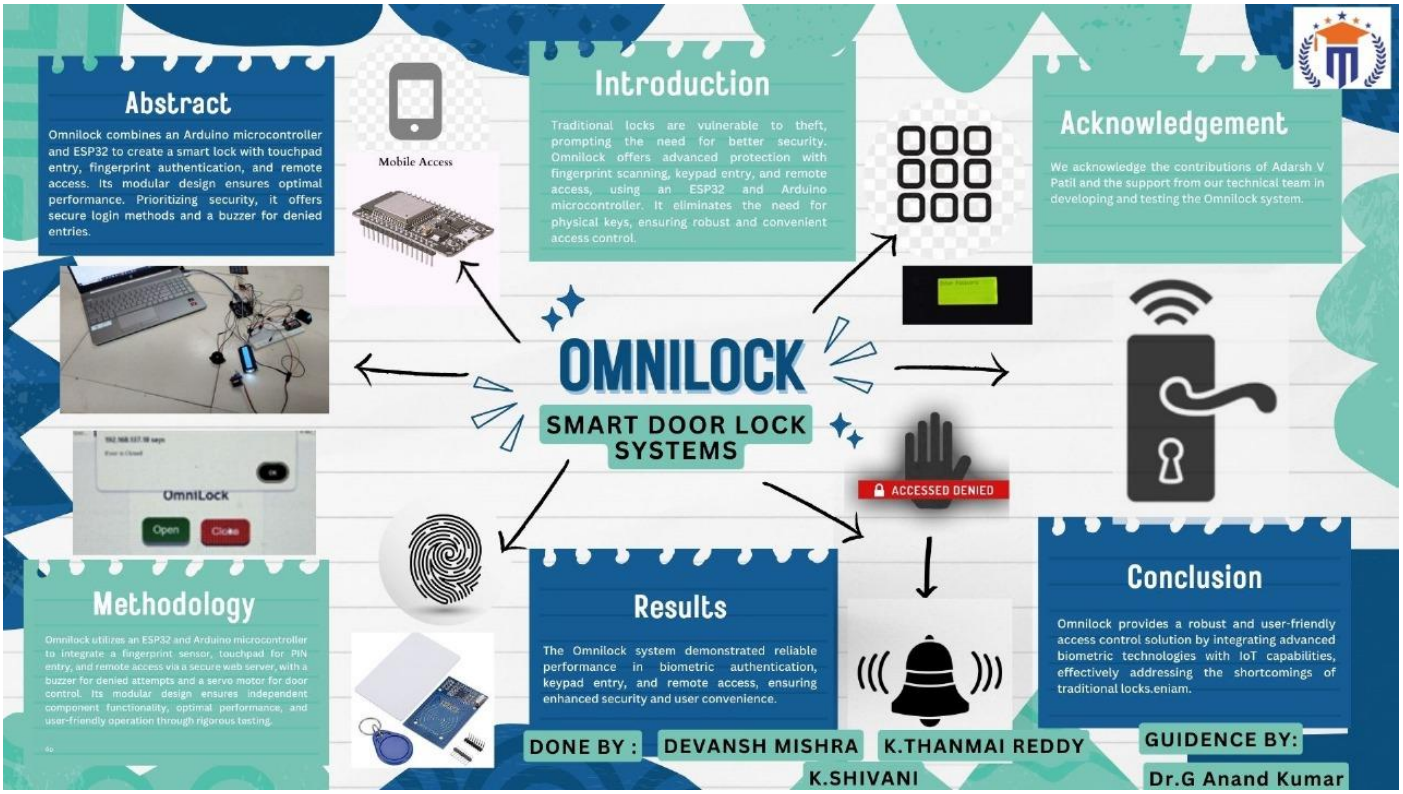
Thanks,
CMT team.

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our Privacy Statement.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

POSTER


Abstract
Omnilock combines an Arduino microcontroller and ESP32 to create a smart lock with touchpad entry, fingerprint authentication, and remote access. Its modular design ensures optimal performance. Prioritizing security, it offers secure login methods and a buzzer for denied entries.
Mobile Access
Introduction
Traditional locks are vulnerable to theft, prompting the need for better security. Omnilock offers advanced protection with fingerprint scanning, keypad entry, and remote access, using an ESP32 and Arduino microcontroller. It eliminates the need for physical keys, ensuring robust and convenient access control.
Acknowledgement
We acknowledge the contributions of Adarsh V Patil and the support from our technical team in developing and testing the Omnilock system.
OMNILOCK
SMART DOOR LOCK SYSTEMS
OmniLock
Open
Close
ACCESSED DENIED
Methodology
Omnilock utilizes an ESP32 and Arduino microcontroller to integrate a fingerprint sensor, touchpad for PIN entry, and remote access via a secure web server, with a buzzer for denied attempts and a servo motor for door control. Its modular design ensures independent component functionality, optimal performance, and user-friendly operation through rigorous testing.
Results
The Omnilock system demonstrated reliable performance in biometric authentication, keypad entry, and remote access, ensuring enhanced security and user convenience.
Conclusion
Omnilock provides a robust and user-friendly access control solution by integrating advanced biometric technologies with IoT capabilities, effectively addressing the shortcomings of traditional locks.eniam.
DONE BY : DEVANSH MISHRA K.THANMAI REDDY K.SHIVANI
GUIDENCE BY:
Dr.G Anand Kumar