KUMARAGURU
Institutions

SMART DOOR LOCK SYSTEM USING MOBILE FINGERPRINT

Team members :

Swetha T	21BEC163
Abharnaa M.R	21BEC002

TABLE OF CONTENTS

INTRODUCTION:

The security aspect is the highest concern of IoT connected entities. The data can be personal, enterprise or consumer. To reach an acceptable implementation for the smart door lock (SDL), security should be taken as a major challenge.

The purpose to develop a smart door lock which is intended to offer high security, easy access, and control. A key challenge that is faced in this project is the security and privacy of the IoT systems.

The goal of the smart lock is to construct an IoT system that includes the SDL application. The system should be secure and user-friendly. The main goal has been allocated to the following subgoals:

- Constructing an architecture regarding the security and functionality.
- Establishing a reliable technique to determine if a user is in the physical proximity of the door lock using Wi-Fi.
- Attaining a proper policy to authenticate users trying to access the door.
- Creating an android application that can serve as the user endpoint.

Project Objectives :

The goal of the project is to find and assess a suitable set of components for creating a password-based smart door locking system with Arduino UNO that provides high security and quick access. The following are the specific goals:

▪ Familiarity with a smart door locking system based on a microcontroller (Arduino).

▪ Using the components to build a simple and smart door locking system.

ABSTRACT:

Mobile-based fingerprint door lock systems have become a popular and efficient way to enhance security in residential and commercial settings. These systems offer a higher level of security and convenience compared to traditional mechanical locks and keypadbased access systems.

The system utilizes the fingerprint as a unique biometric feature to grant or deny access to a particular area or room. This technology is rapidly gaining popularity because of its cost-effectiveness, ease of use, and high level of security. Here we will be using ESP8266 microcontroller development board with wifi capability.

COMPONENTS REQUIRED:

Hardware requirement:

- ESSP8266(NodeMCU)
- Relay ○ Solenoid lock ○ Jumper wires

Software requirement:

- Kodular mobile application ○ Arduino IDE

COST ESTIMATION :

S.No	component	quantity	cost
1.	ESP8266nodemcu	1	500
2.	1 channel 5v relay	1	60
	Total		560

COMPONENT DETAILS:

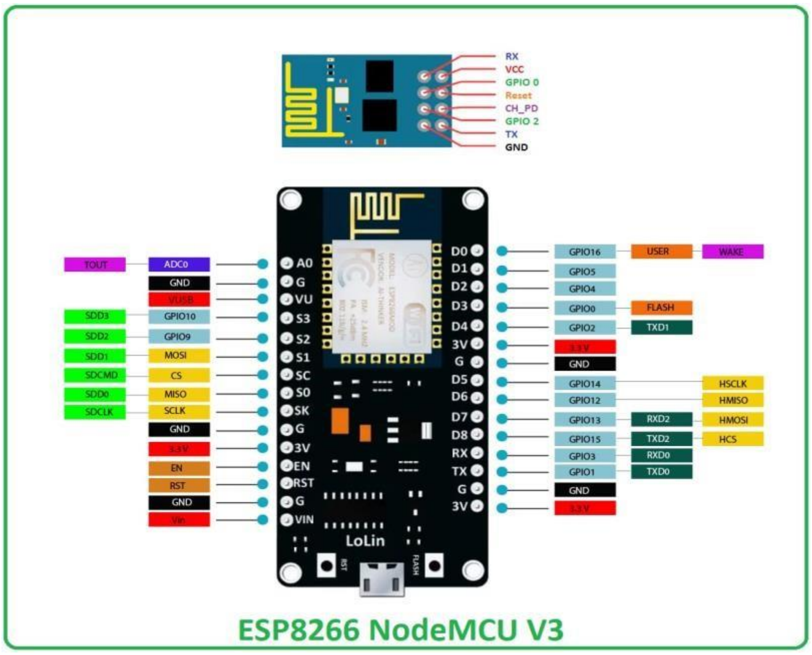
ESP8266 NODEMCU:

The ESP8266 NodeMCU is a popular development board based on the ESP8266 microcontroller. It offers a convenient way to prototype and build Internet of Things (IoT) projects. Here are the specifications of the NodeMCU board:

- Microcontroller: The NodeMCU board is based on the ESP8266 microcontroller, which is a low-cost Wi-Fi-enabled chip manufactured by Espressif Systems.
- Processor: The ESP8266 chip on the NodeMCU board is powered by a Tensilica L106 32-bit RISC processor running at 80 MHz.
- Wi-Fi: The board provides built-in 802.11 b/g/n Wi-Fi connectivity, allowing it to connect to wireless networks and communicate over the internet.
- Memory: It has 4MB (32Mb) of flash memory, which is used to store the firmware, program code, and other data.
- GPIO Pins: The NodeMCU board features a series of General-Purpose Input/Output (GPIO) pins that can be used to connect and control external devices such as sensors, actuators, and displays. It typically has about 10 GPIO pins available for use.

- Analog Input: The board offers a single analog input pin (ADC) that can measure voltages from 0 to 3.3V.

- USB Interface: It includes a micro-USB port for power supply and serial communication with the computer.



Datasheet Link : https://handsontec.com/dataspecs/module/esp8266-V13.pdf

RELAY MODULE :

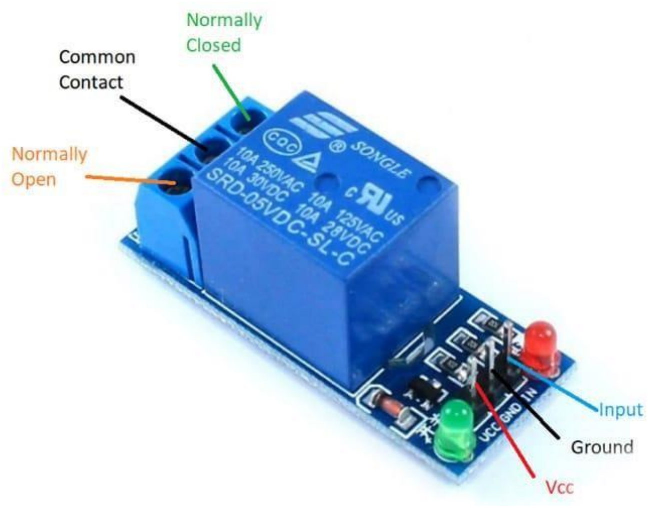
- A 5V 10A relay module is an electronic component that allows low-voltage control signals (such as 5V) to switch high-current loads (up to 10A). Here are some specifications typically associated with such a relay module:

- Input voltage: 5V DC - The module operates using a 5V direct current (DC) power supply.

- Contact rating: 10A - This indicates the maximum current that the relay contacts can handle. In this case, it can switch loads up to 10 amperes.

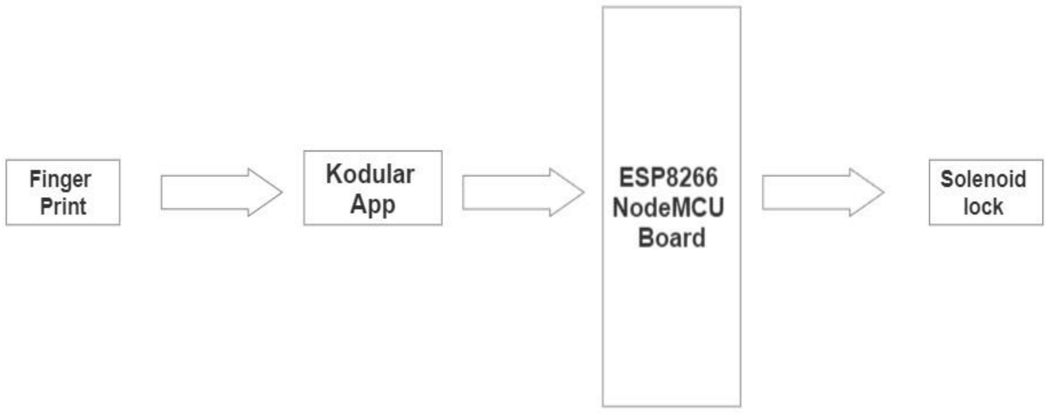
- Normally Open (NO) and Normally Closed (NC) contacts: A relay module usually has at least one normally open contact and one normally closed contact. The NO contact remains open (disconnected) when the relay is not activated, while the NC contact remains closed (connected) in the same state.

- Activation current: The module requires a certain amount of current to activate the relay coil. The exact value depends on the specific relay used in the module.

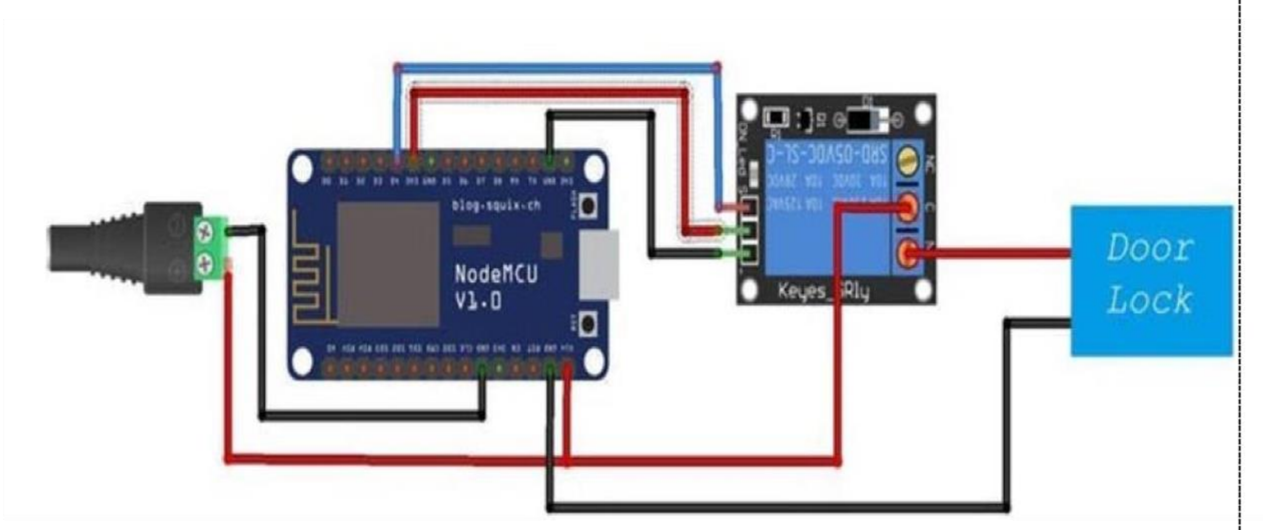


Datasheet Link : https://handsontec.com/dataspecs/relay/1Ch-relay.pdf

BLOCK DIAGRAM:



CIRCUIT DIAGRAM:



WORKING:

The smart mobile fingerprint door lock system relies on several key steps to ensure secure and efficient operation. Below is a detailed explanation of each step:

- Connection establishment: In order to establish a connection between the relay and the Kodular application, it is necessary to enter the IP address of the Nodemcu, which acts as an intermediary between the application and the door lock. This IP address allows the application to efficiently communicate with nodemcu.

- Fingerprint Identification: The user's fingerprint is identified through the fingerprint interface of the Kodular mobile app. This interface captures and analyzes the unique characteristics of the user's fingerprint for authentication.

- Fingerprint Verification: The system verifies the scanned fingerprint to ensure its authenticity. Compare the captured fingerprint with the registered fingerprints stored in the database. If the captured fingerprint matches one of the enrolled fingerprints, the verification process is successful.

- Message Broadcast: After fingerprint recognition and verification, Kodular sends a message to nodemcu using an HTTP command. This message serves as a signal to unlock the door and allow the user access.

- Match Request Response: Nodemcu receives the message from the Kodular application and processes it. Make sure the command received matches the response required to unlock the door. The desired response can be specific code or an instruction programmed in nodemcu.

- Unlock Door: If the Kodular message matches the requested response, the Nodemcu controller activates the unlocking mechanism attached to the door. This may include activation of a door unlocking relay or activation of an electronic locking system allowing user access.

- Implementation of securecommunication: Depending on the specific requirements of the smart lock system, additional security measures can be implemented.

- For example, a buzzer or alarm system can be integrated to warn users or administrators of unauthorized access attempts or potential security breaches. Encryption protocols and secure data transmission techniques can also be used to ensure the confidentiality and integrity of the communication between the mobile application and nodemcu.

CODE:

```
#include <ESP8266WiFi.h>
WiFiClient client; WiFiServer server (80);
```

```
/* WIFI settings */
Const char* ssid = “id”;   //WIFI SSID
Const char* password = “pass”;    //WIFI PASSWORD

/* data received from application */
String  data =””;
Int Relay = 5;     //D1


Void setup()
```

```
{
  pinMode(Relay, OUTPUT);   digitalWrite(Relay,LOW);    Serial.begin(9600);
connectWiFi();   server.begin();
}


Void loop()
{
   /* If the server available, run the "checkClient" function */
   Client = server.available();
   If (!client) return;      Data = checkClient ();
Serial.print(data);


  If (data == "RELAYSUCCESS")
   {
     digitalWrite(Relay,HIGH);
     }
Else{       digitalWrite(Relay,LOW);
     }
      }

Void connectWiFi()
{
  Serial.println("Connecting to WIFI");   WiFi.begin(ssid, password);


  While ((!(WiFi.status() == WL_CONNECTED)))
  {
    Delay(300);
    Serial.print("..");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("NodeMCU Local IP is : ");
  Serial.print((WiFi.localIP()));
}


String checkClient (void)
{
  While(!client.available()) delay(1);
  String request = client.readStringUntil('\r');
  Request.remove(0, 5);
  Request.remove(request.length()-9,9);
  Return request;
}
```

CODE EXPLANATION :

The code you provided is an Arduino sketch written for the ESP8266 microcontroller board (NodeMCU) to control a relay module using Wi-Fi. Let's break down the code section by section:

- #include <ESP8266WiFi.h>
  WiFiClient client;
  WiFiServer server(80);
  These lines include the necessary libraries for the ESP8266 Wi-Fi module and define the `client` and `server` objects to handle Wi-Fi client connections.

- const char* ssid = "id";   // WIFI SSID const char* password = "pass"; // WIFI PASSWORD
  These lines define the Wi-Fi network credentials, such as the SSID (network name) and password.

- String data = ""; int Relay = 5;   // D1

  Here, a `String` variable `data` is declared to store the received data from the application. An `int` variable `Relay` is defined and set to pin D1 (GPIO 5) to control the relay module.

- void setup() { pinMode(Relay, OUTPUT); digitalWrite(Relay, LOW); Serial.begin(9600); connectWiFi(); server.begin();
  }
  The `setup()` function is called once at the beginning. It sets the `Relay` pin as an output and initializes it with a low (OFF) state. It also starts the serial communication for debugging and calls the `connectWiFi()` function to establish a connection to the Wi-Fi network. Finally, it starts the server on port 80.

- void loop() { client = server.available(); if (!client) return; data = checkClient(); Serial.print(data);
  if (data == "RELAYSUCCESS") { digitalWrite(Relay, HIGH);
  } else { digitalWrite(Relay, LOW);
  }
  }
  The `loop()` function is called repeatedly after the `setup()` function. It checks if a client is available and assigns it to the `client` object. If there is no client available, it returns. Otherwise, it calls the `checkClient()` function to read the client's request data. The received data is printed to the serial monitor. If the received data matches "RELAYSUCCESS", it sets the `Relay` pin to a high (ON) state. Otherwise, it sets it to a low (OFF) state.

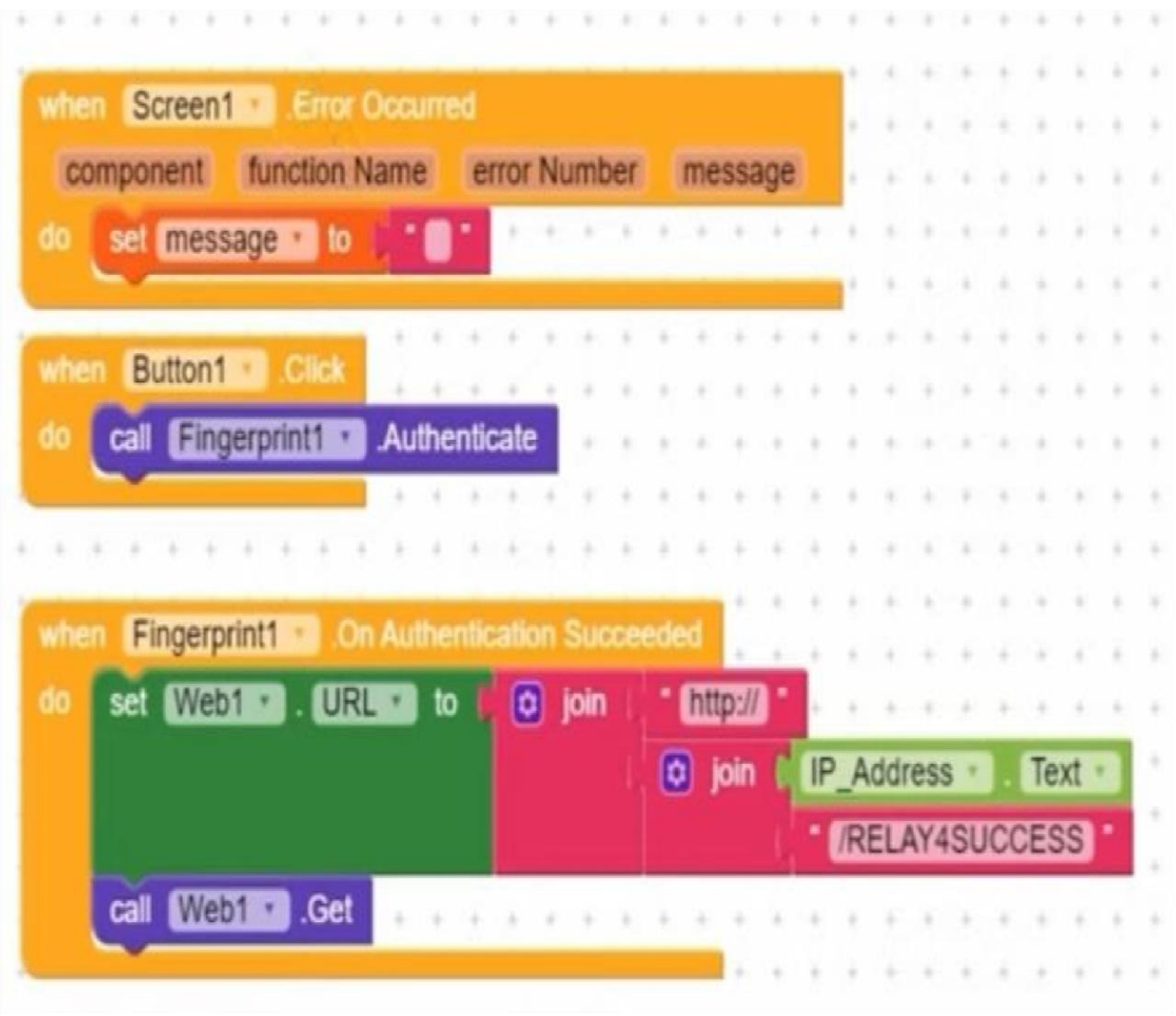
- void connectWiFi() {
  Serial.println("Connecting to WIFI");

WiFi.begin(ssid, password); while (!(WiFi.status() == WL_CONNECTED))
{ delay(300);
Serial.print(".."); }
Serial.println();
Serial.println("WiFi connected");
Serial.println("NodeMCU Local IP is: ");
Serial.print(WiFi.localIP());
}
The `connectWiFi()` function is responsible for connecting the ESP8266 to the specified Wi-Fi network. It starts the Wi-Fi connection process, waits until the connection is established, and prints the local IP address of the NodeMCU to the serial monitor.

o String checkClient() { while (!client.available())
delay(1);
String request = client.readStringUntil('\r'); request.remove(0, 5);
request.remove(request.length() - 9, 9); return request;
}
The `checkClient()` function reads the client's request data and removes any leading or trailing characters. It returns the processed request data as a `String`.

Overall, this code sets up a web server on the ESP8266 and waits for a client to send a request. When a request is received, it checks the content of the request. If it matches "

KODULAR APP DEVELOPMENT -BLOCK :



```
when Screen1 .Error Occurred
  component  function Name  error Number  message
do  set message to " "

when Button1 .Click
do  call Fingerprint1 .Authenticate

when Fingerprint1 .On Authentication Succeeded
do  set Web1 . URL to  join  " http:// "
                             join  IP_Address . Text
                                   " /RELAY4SUCCESS "
    call Web1 .Get
```

STEPS TO CREATE THE BLOCKS :

I. First the application checks for any error in the screen.

II. If any error occurs, it will not send any message. It will just present a blank screen.

III. We can modify it accordingly, if we want to display any messages like ‘an error occurred’ by using the set function from the available blocks.

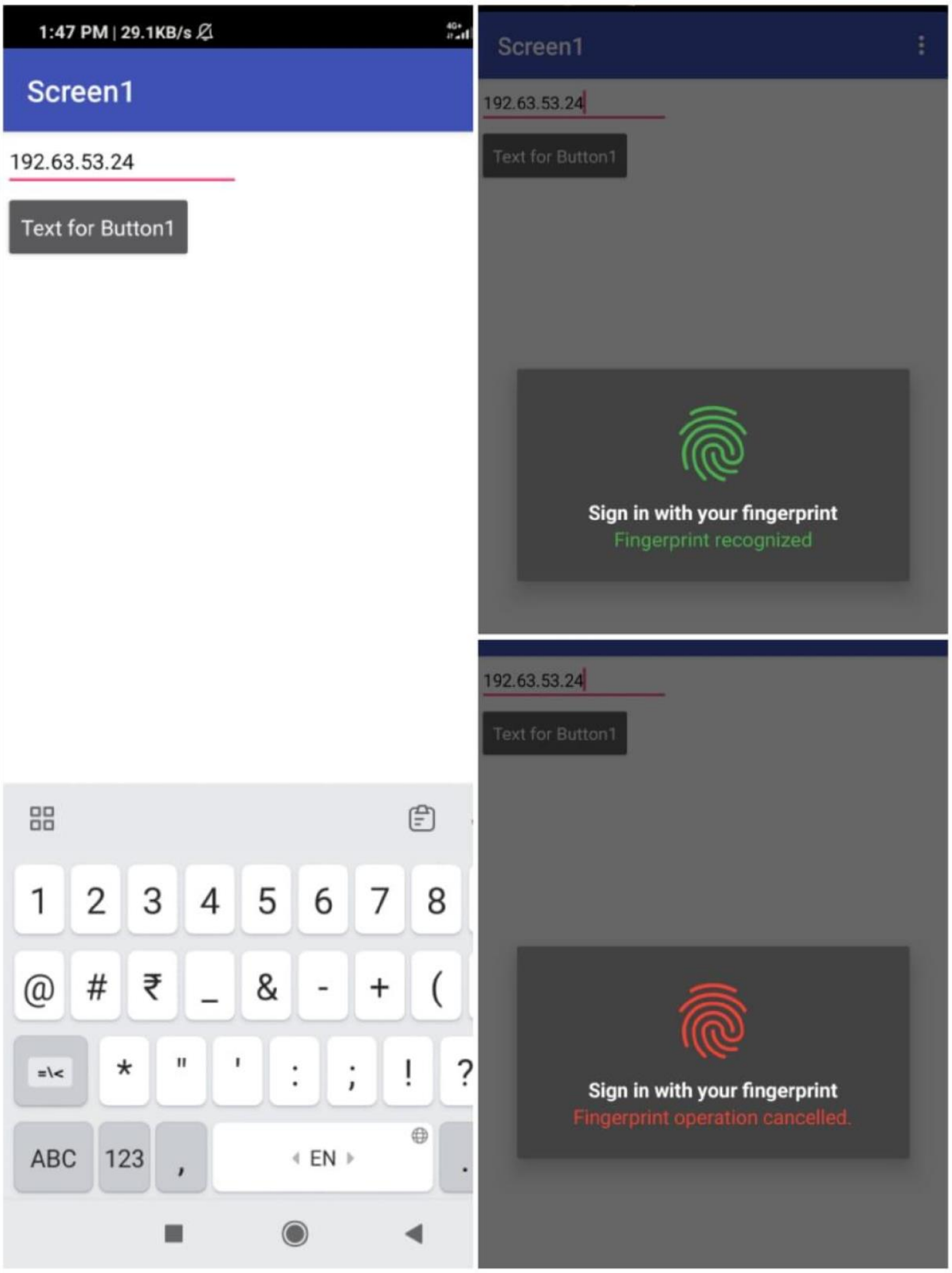
IV. Secondly, when the user wants to open the door, he should press the fingerprint icon button in the top most corner.

V. The finger print will be authenticated based on the original fingerprint saved in that particular device.

VI. When the fingerprint authentication has been succeeded, the kodular app will set up a http web URL network in port 80 which is the reserved port for http communication.

VII. It is followed up by appending the IP address i.e., the IP address of the nodemcu which has to be entered prior to the fingerprint authentication

MOBILE APPLICATION:



192.63.53.24

192.63.53.24

Sign in with your fingerprint
Fingerprint recognized

192.63.53.24

Sign in with your fingerprint
Fingerprint operation cancelled.

FIREBASE AND ITS USE CASE :

Firebase is a comprehensive mobile and web application development platform that provides developers with a wide range of tools and services to build, improve, and scale their applications. It was initially developed by Firebase Inc., a startup acquired by Google in 2014. Firebase offers a suite of backend services, SDKs (Software Development Kits), and ready-to-use features that simplify the development process and help developers focus on building high-quality applications.

KEY FEATURES AND USECASE OF FIREBASE:

1. Realtime Database: Firebase's Realtime Database is a cloud-hosted NoSQL database that enables developers to store and sync data in real time between clients and servers. It allows for seamless data synchronization and updates across multiple devices and platforms.

2. Authentication: Firebase provides an easy-to-use authentication system that allows developers to add secure user sign-up, sign-in, and management functionality to their applications.

3. Cloud Firestore: Firestore is Firebase's cloud-native, scalable NoSQL document database. It offers flexible data modeling, powerful querying, and real-time synchronization. Firestore is designed to handle large-scale applications and provides offline support, making it ideal for mobile and web applications that require offline access.

4. Cloud Storage: Firebase Storage provides secure cloud storage for usergenerated content such as images, videos, and files. It offers an easy-to-use API for uploading, downloading, and managing files, and it integrates seamlessly with other Firebase services.

5. Cloud Functions: Firebase Cloud Functions allows developers to run serverside code in response to events triggered by Firebase services or HTTPS requests. It enables developers to extend their application's functionality without managing server infrastructure.

6. Cloud Messaging: Firebase Cloud Messaging (FCM) enables developers to send push notifications to their users on mobile and web platforms. It supports targeting specific devices, user segments, or topics, making it easy to engage and re-engage users with personalized notifications.

7. Performance Monitoring: Firebase Performance Monitoring helps developers gain insights into their application's performance and user experience. It provides detailed metrics on network latency, app startup time, and other performance indicators, helping developers identify and optimize performance bottlenecks.

8. Analytics: Firebase Analytics offers comprehensive app usage analytics, allowing developers to track user behavior, measure ad performance, and gain insights into their application's usage patterns. It provides valuable data to make informed decisions and optimize the app's user experience.

9. Test Lab: Firebase Test Lab provides a cloud-based infrastructure for testing Android and iOS apps on real devices. It allows developers to run automated tests and perform compatibility testing across a wide range of devices, helping ensure app quality and reliability.

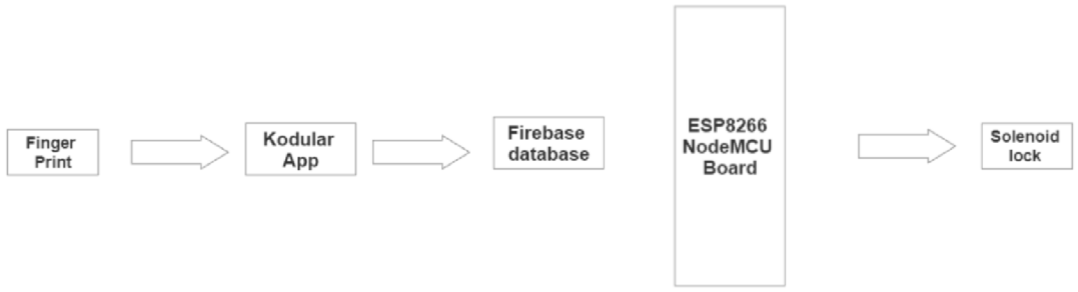
USE CASE OF FIREBASE:

1. Real-time Applications: Firebase's real-time database and Firestore are ideal for building collaborative applications such as chat apps, collaborative document editors, or multiplayer games. They enable seamless data synchronization and real-time updates across multiple clients.

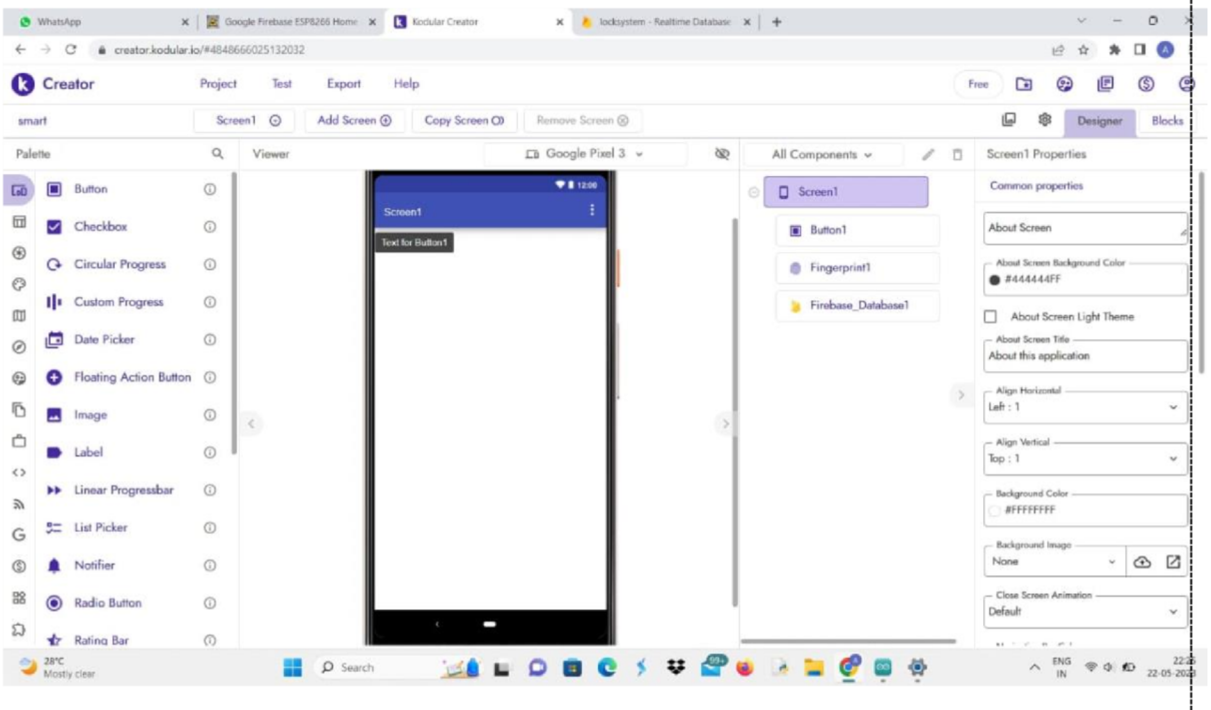
2. User Authentication and Authorization: Firebase Authentication simplifies the process of adding user authentication to applications, allowing developers to secure user data, manage user accounts, and provide personalized experiences based on user profiles.

3. Mobile and Web App Development: Firebase provides a comprehensive set of tools and services that streamline the development process, from data storage and synchronization to cloud functions and push notifications. It simplifies backend development and allows developers to focus on building front-end features.

BLOCK DIAGRAM :



KODULAR CODE BLOCK :



Common properties

About Screen

About Screen Background Color: #444444FF

☐ About Screen Light Theme

About Screen Title: About this application

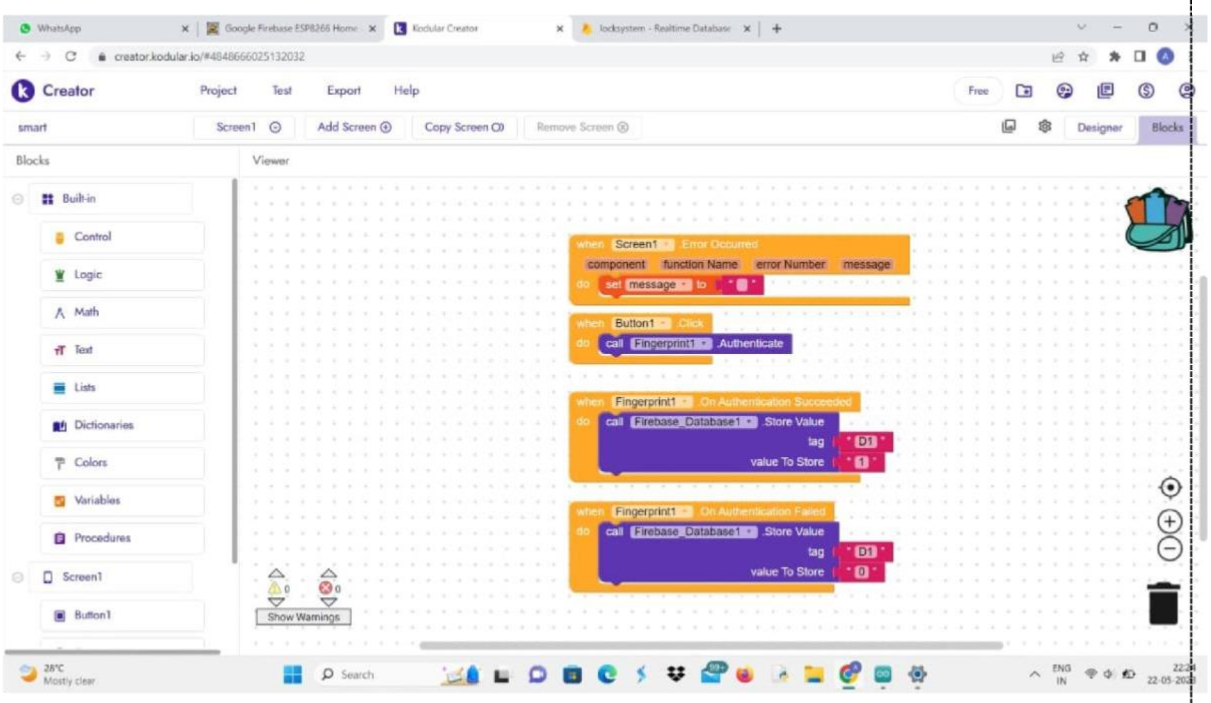
Align Horizontal: Left : 1

Align Vertical: Top : 1

Background Color: #FFFFFFFF

Background Image: None

Close Screen Animation: Default



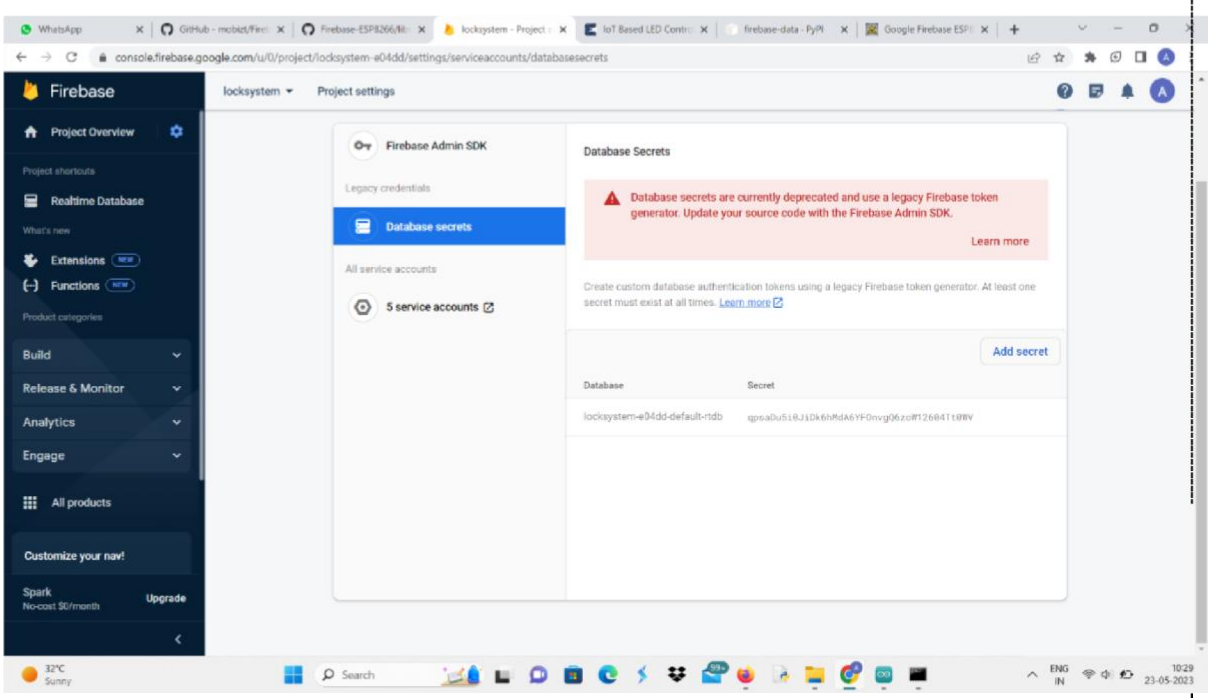
```
when Screen1 .Error Occurred
  component  function Name  error Number  message
do  set message to " "

when Button1 .Click
do  call Fingerprint1 .Authenticate

when Fingerprint1 .On Authentication Succeeded
do  call Firebase_Database1 .Store Value
                            tag  "D1"
                  value To Store  "1"

when Fingerprint1 .On Authentication Failed
do  call Firebase_Database1 .Store Value
                            tag  "D1"
                  value To Store  "0"
```

FIREBASE REALTIME DATA BASE:



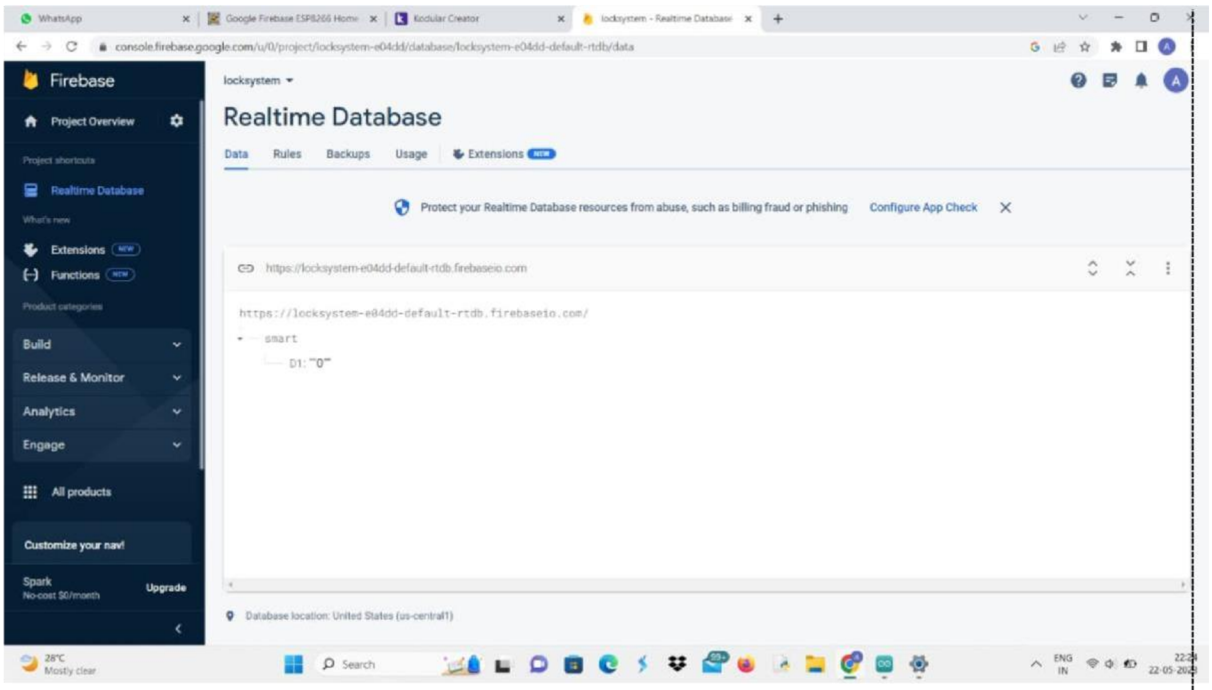
Database Secrets

Database secrets are currently deprecated and use a legacy Firebase token generator. Update your source code with the Firebase Admin SDK.

Learn more

Create custom database authentication tokens using a legacy Firebase token generator. At least one secret must exist at all times. Learn more

Database	Secret
locksystem-e04dd-default-rtdb	qpsaDu5i0JiDk6hMdA6YFOnvgQ6zoM12604Tt0WV



Realtime Database

```
https://locksystem-e04dd-default-rtdb.firebaseio.com/
- smart
   D1: "0"
```

To create a database in Firebase, follow these step-by-step instructions:

Step 1: Set up a Firebase Project

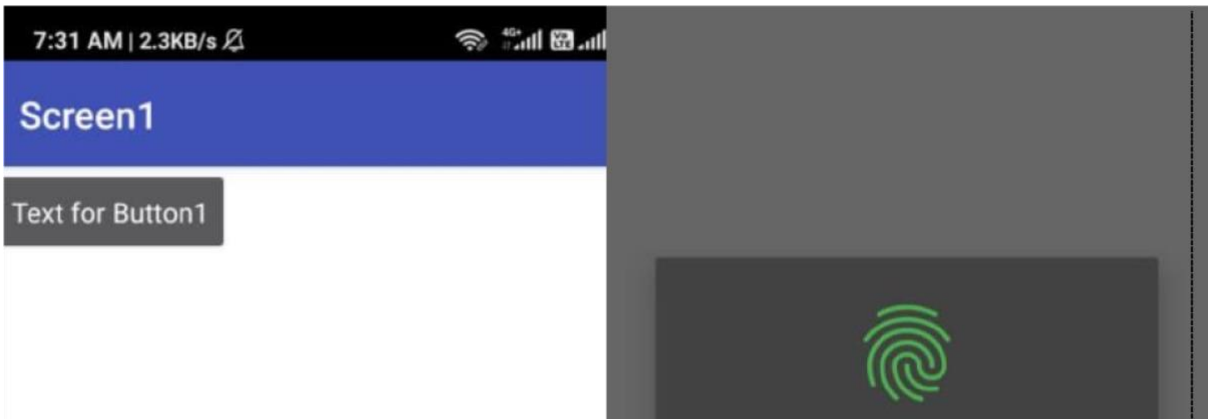
- Go to the Firebase website (https://firebase.google.com) and sign in with your Google account.
- Click on "Go to Console" in the top-right corner to access the Firebase Console.
- Click on the "Add project" button to create a new Firebase project.
    - Enter a name for your project and select your desired location.
    - Enable Google Analytics for your project (optional).

- Click on "Create project" to proceed.

Step 2: Set up Firebase Database

- In the Firebase Console, select your newly created project.
- In the left-side navigation menu, click on "Database" under the "Develop" section.
- Choose "Create Database" on the Database page.
- Select the "Start in test mode" option to set up security rules for your database later. If you're building a production application, you should review Firebase security rules and set up appropriate access controls.
- Choose a location for your database (e.g., "us-central1") and click on "Next."

KODULAR APP INTERFACE :


7:31 AM | 2.3KB/s
Screen1
Text for Button1

KODULAR AND FIREBASE INTERFACE :

To interface Firebase Database with Kodular app and NodeMCU in Arduino IDE, follow these steps:

Step 1: Set up Firebase Database

1. Follow the instructions mentioned earlier to create a Firebase project and set up the database.

2. Make note of your Firebase project credentials, including the API key, database URL, and other necessary details.

Step 2: Kodular App Setup

1. Create a new project in Kodular or open an existing project.

2. Drag and drop the necessary components to your app screen. For Firebase integration, you will typically need the Firebase component and other UI components as per your app requirements.

3. Configure the Firebase component by entering your Firebase project credentials (API key, database URL, etc.).

4. Use the appropriate blocks provided by the Firebase component to perform database operations such as reading or writing data. For example, you can use blocks like "Get Value" or "Set Value" to interact with the Firebase Database.

Step 3: NodeMCU Setup

1. Connect your NodeMCU to your computer and open the Arduino IDE.

2. Install the necessary libraries for Firebase integration. The two popular libraries for Firebase integration with NodeMCU are "FirebaseArduino" and "Firebase ESP8266."

3. Install the libraries by going to "Sketch" -> "Include Library" -> "Manage Libraries" and searching for the respective libraries. Click on "Install" to add them to your Arduino IDE.

4. Import the required libraries into your Arduino sketch by including the appropriate headers at the beginning of your code. For example:

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
```

5. Configure the NodeMCU to connect to your Wi-Fi network by providing your network SSID and password in your code. For example:

```
#define WIFI_SSID "your_wifi_ssid"
#define WIFI_PASSWORD "your_wifi_password"
```

6. Initialize the Firebase connection by providing your Firebase project credentials and database URL. For example:

```
#define FIREBASE_HOST "your_project_id.firebaseio.com"
#define FIREBASE_AUTH "your_firebase_auth"
```

7. Use the FirebaseArduino library functions to interact with the Firebase Database. For instance, you can use functions like `Firebase.set()` or `Firebase.push()` to write data to the database, or `Firebase.getString()` or `Firebase.getInt()` to read data from the database.

Step 4: Upload Code to NodeMCU

1. Connect your NodeMCU to your computer via USB.

2. Select the appropriate board and port in the Arduino IDE from the "Tools" menu.

3. Copy your NodeMCU code into the Arduino IDE.

4. Verify that there are no syntax errors, and then click on the "Upload" button to compile and upload the code to your NodeMCU.

With these steps, you should be able to interface the Firebase Database with your Kodular app and NodeMCU using the Arduino IDE.

FIREBASE CODE :

```
#include <FirebaseESP8266.h>

#include  <ESP8266WiFi.h>

#define ssid "Redmi Note 10S"  //WiFi SSID

#define password "00000swetha"  //WiFi Password

#define FIREBASE_HOST "locksystem-e04dd-default-rtdb.firebaseio.com"

#define FIREBASE_AUTH "qpsaDu5i0JiDk6hMdA6YFOnvgQ6zoW12604Tt0WV"

FirebaseData fbdo; int Relay =D6 ;

String a="";


void setup() {

 Serial.begin(9600);

   WiFi.begin (ssid, password);    while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println ("");
  Serial.println ("WiFi Connected!");

  Firebase.begin(FIREBASE_HOST,FIREBASE_AUTH);

pinMode(Relay,OUTPUT);//initialize the Device OUTPUT
} void loop() {   if (Firebase.getString(fbdo, "/smart/D1")) {     a = fbdo.stringData();
    a.remove(0,2);
    a.remove(1,2);     Serial.print(a);
  }   if (a=="1"){     digitalWrite(Relay,HIGH);
  }   if(a=="0"){     digitalWrite(Relay,LOW);
  }
          }
```

CODE EXPLANATION :

This code is written for an ESP8266 microcontroller that is connected to a Wi-Fi network and interacts with a Firebase Realtime Database. Let's go through the code step by step to understand its functionality:

1. The necessary libraries are included:

- `FirebaseESP8266.h`: This library provides the Firebase API for the ESP8266 microcontroller.
- `ESP8266WiFi.h`: This library allows the ESP8266 to connect to a Wi-Fi network.

2. The Wi-Fi network credentials are defined:

- `ssid`: It represents the Wi-Fi network name (SSID) to which the ESP8266 will connect.

- `password`: It represents the password for the Wi-Fi network.

3. The Firebase configuration parameters are defined:

- `FIREBASE_HOST`: It specifies the URL of the Firebase Realtime Database to which the ESP8266 will connect.

- `FIREBASE_AUTH`: It represents the authentication token for accessing the Firebase Realtime Database. This token ensures secure communication between the ESP8266 and Firebase.

4. The `FirebaseData` object `fbdo` is declared. This object is used to interact with the Firebase Realtime Database.

5. The variable `Relay` is assigned the value `D6`, which represents the GPIO pin on the ESP8266 that controls a relay or an output device.

6. In the `setup()` function:

- The serial communication is initiated at a baud rate of 9600.

- The ESP8266 attempts to connect to the specified Wi-Fi network using the provided credentials.

- The program waits until the Wi-Fi connection is established (`WL_CONNECTED` status).

- The Firebase connection is established by calling `Firebase.begin()` with the provided Firebase configuration parameters.

- The `Relay` pin is configured as an output pin using `pinMode()`.

7. In the `loop()` function:
- The code checks if there is a new string value available in the Firebase Realtime Database under the path "/smart/D1" using `Firebase.getString()`. The data is retrieved and stored in the `a` variable.

- The code removes the first two characters and the third and fourth characters from the `a` variable using `a.remove()` to extract the actual value.

- The extracted value is printed to the serial monitor using `Serial.print()`.
- If the extracted value is "1", the `Relay` pin is set to `HIGH` using `digitalWrite()`, turning on the connected device.
- If the extracted value is "0", the `Relay` pin is set to `LOW` using `digitalWrite()`, turning off the connected device.

FEATURES IN THIS SMART DOOR LOCK :

1. Mobile fingerprint :

Using mobile fingerprint for a smart door lock system offers several benefits:

1. Enhanced Security: Mobile fingerprints provide a highly secure authentication method. Fingerprint patterns are unique to individuals, making it difficult for unauthorized users to gain access. Compared to traditional key-based or password-based systems, mobile fingerprints offer a higher level of security.

2. Convenience: Mobile fingerprint authentication is convenient for users. They don't need to remember or carry physical keys or passwords. Their fingerprint is always with them on their mobile device, making it easy to authenticate and unlock the door with a simple touch.

3. Accessibility: Mobile fingerprints make the door lock system more accessible to users with physical disabilities.

4. Integration with Mobile Devices: Most people carry their mobile devices with them wherever they go. Integrating the door lock system with mobile fingerprints leverages the ubiquity of mobile devices and utilizes their built-in biometric sensors, eliminating the need for additional hardware.

5. Remote Access: With mobile fingerprints, users can remotely grant access to authorized individuals. For example, if a family member or friend needs access to the house while the owner is away, the owner can remotely provide temporary access permissions using a mobile app, ensuring convenience and security.

7. Integration with Smart Home Systems: Mobile fingerprint authentication can integrate with other smart home devices and systems. For example, when the door is unlocked using a mobile fingerprint, it can trigger other actions like turning on lights, adjusting temperature, or disabling security alarms, providing a seamless smart home experience.

It's important to note that while mobile fingerprints offer significant advantages, no security system is foolproof. It's always recommended to have additional security measures in place and regularly update and monitor the smart door lock system to ensure the highest level of security.

2.IP address Vs Firebase Database:

The use of IP addresses for NodeMCU and Firebase Realtime Database differs based on their specific functionalities. Let's explore each case:

1. NodeMCU IP Address:

- When working with NodeMCU, the IP address is typically used to establish a network connection and communicate with other devices on the same network.
- NodeMCU connects to a Wi-Fi network and obtains an IP address assigned by the network's DHCP server.

- The IP address allows the NodeMCU to send and receive data over the network, enabling communication with other devices, including accessing the Firebase Realtime Database.

2. Firebase Realtime Database:

- The Firebase Realtime Database does not require the use of IP addresses directly. Instead, Firebase provides a URL endpoint for accessing the Realtime Database. This URL is specific to your Firebase project and does not involve manually dealing with IP addresses.

- To interact with the Firebase Realtime Database from NodeMCU, you would typically use the FirebaseArduino or Firebase ESP8266 libraries. These libraries abstract the network communication details, including handling the connection to the Firebase Realtime Database endpoint using the provided URL.

- The process involves initializing the Firebase connection with your project credentials and database URL (e.g., "your-project-id.firebaseio.com"), rather than dealing with IP addresses directly. The libraries handle the underlying network connection and communication with the Firebase Realtime Database.

Overall, while NodeMCU requires an IP address to establish a network connection, when interacting with the Firebase Realtime Database, you can use the Firebase-specific URL endpoint instead of dealing with IP addresses directly.

ADVANTAGES:

1. Enhanced Security: Smart lock systems often employ advanced authentication methods such as biometric recognition (e.g., fingerprint or facial recognition), PIN codes, or digital keys, which can significantly enhance security compared to traditional lock systems. These advanced authentication mechanisms reduce the risk of unauthorized access and provide greater control over who can enter a specific space.

2. Convenience and Flexibility: Smart lock systems offer convenience and flexibility in managing access. They eliminate the need for physical keys and allow users to unlock doors remotely using smartphones or other authorized

devices. This feature is particularly useful for granting temporary access to guests or service providers without the need for physical presence.

3. Remote Monitoring and Management: Smart lock systems can be integrated with other smart home or building automation systems, allowing remote monitoring and management of access control. This capability enables real-time monitoring of door activity, the ability to lock or unlock doors remotely, and receiving notifications or alerts when someone enters or exits a space. It provides peace of mind and enables efficient management of access control in various scenarios.

4. Audit Trail and Accountability: Smart lock systems often provide an audit trail feature that records entry and exit events. This information can be valuable for security purposes, tracking employee attendance, or investigating any security breaches. The audit trail enhances accountability and helps in identifying potential security issues or irregularities.

5. Integration with Other Smart Devices: Smart lock systems can integrate with other smart devices and systems, creating a cohesive and interconnected smart ecosystem. For example, integration with security cameras, alarm systems, or home automation systems can provide a comprehensive security solution and enable automation based on access events (e.g., turning on lights upon unlocking the door).

6. Scalability and Customization: Smart lock systems offer scalability, allowing the addition of multiple locks and access points within a project. They can be customized to meet specific requirements, such as defining access schedules for different individuals or groups, setting up temporary access codes, or implementing multi-factor authentication methods.

7. Energy Efficiency: Some smart lock systems incorporate energy-saving features such as auto-locking mechanisms or power-saving modes. These features help conserve energy by automatically locking doors after a specified period of inactivity or optimizing power consumption when the lock is idle.

8. Integration with Cloud Services: Smart lock systems can leverage cloud services, enabling seamless synchronization and backup of access control data. Cloud integration ensures data reliability, accessibility from anywhere, and the ability to manage access control remotely.

Overall, smart lock systems offer improved security, convenience, and flexibility, along with the ability to integrate with other smart devices and systems. They are poised to play a significant role in future projects, ranging from residential homes and commercial buildings to hospitality and industrial applications.

APPLICATIONS OF SMART LOCK :

Mobile fingerprint door locks have several uses and provide a number of advantages. The following are some typical uses for mobile fingerprint door locks:

- Residential Security: To improve home security, mobile fingerprint door locks can be employed in residential settings. Instead of using traditional keys, homeowners can unlock doors using their smartphones or registered fingerprints. This is practical and reduces the possibility of keys being misplaced or stolen.

- Mobile fingerprint door locks are frequently used in workplaces, corporations, and commercial buildings. They provide improved access management and security. Access may be easily regulated and monitored remotely, and authorised workers can enter using their registered fingerprints or mobile devices.

- Mobile fingerprint door locks are common in the hospitality sector, especially in hotels and resorts. There is no need for real keys or keycards because guests can access their rooms using their registered fingerprints or mobile devices. Convenience and an additional layer of security are provided by this.

- Healthcare Facilities: To ensure restricted access to sensitive locations like patient rooms, medicine storage spaces, and laboratories, mobile fingerprint door locks can be used in healthcare settings. To increase privacy and security, authorised staff employees can manage access using their mobile devices or fingerprints.

- Schools, colleges, and universities can use mobile fingerprint door locks to control access to classrooms, laboratories, administrative offices, and other restricted areas. They offer a productive means of controlling access permissions and boosting campus security.

CONCLUSION:

Smart mobile fingerprint door locks are a innovation in the field of access control technology. By combining biometric authentication, mobile integration, and remote management features, these locks offer enhanced security, convenience, and flexibility across various applications. The benefits of keyless entry, secure biometric authentication, and remote access management make them an attractive choice for residential, commercial, and institutional settings. As technology continues to advance, mobile fingerprint door locks are expected to play an increasingly significant role in ensuring secure access and providing peace of mind for individuals and organizations alike.

REFERENCE :

1. https://smartloock-15eb6-default-rtdb.firebaseio.com/

2. https://www.electronicsforu.com/electronics-projects/otp-based-smart-wirelesslocking-system

3. https://blynk.hackster.io/adarsh-bhola/iot-based-door-lock-system-4a40f0

4. https://iotdesignpro.com/projects/iot-based-smart-door-lock-system-usingnodemcu

5. https://circuitdigest.com/microcontroller-projects/cell-phone-controlledfingerprint-solenoid-door-lock-using-arduino