PASSWORD BASED SMART DOOR LOCK SECURITY SYSTEM WITH MOTION DETECTION LIGHTING

EC6020-EMBEDDED SYSTEM AND DESIGN

FINAL PROJECT REPORT

Submitted By:

ASRA S.A.F 2020/E/013

ATPUTHARAVI R. 2020/E/015

BANDARA A.H.M.V.L 2020/E/017

PROJECT TITLE: PASSWORD BASED SMART DOOR LOCK SECURITY SYSTEM WITH MOTION DETECTION LIGHTING

1. INTRODUCTION

Problem Statement

Traditional lock systems offer limited security features and lack remote accessibility. In an era where home automation is increasingly popular, there is a growing need for smart lock systems that provide enhanced security, remote control, and real-time notifications.

Solution

This project aims to design a smart, password-based door lock security system that integrates motion detection lighting and WiFi based remote access. This system will provide enhanced security and user convenience, making it suitable for homes, offices, and other secure areas.

2. PROJECT DESIGN AND IMPLEMENTATION

System Overview

The Smart Lock System addresses these issues by integrating an ESP32 microcontroller with a servo motor for locking control, a PIR sensor for motion detection, and Firebase for remote monitoring and control. The system is complemented by a Flutter mobile app that allows users to manage the lock, view status updates, and receive notifications.

Hardware Design

1. ESP32 Microcontroller:

   Controls the servo motor and reads inputs from the PIR sensor.

   Connects to Firebase and Wi-Fi for remote monitoring and control.

2. Servo Motor (SG90):

   Controls the lock mechanism (locked/unlocked positions).

3. PIR Sensor:

   Detects motion and sends input to the ESP32.

4. LCD Display:

   Provides user feedback and status messages.

5. Keypad:

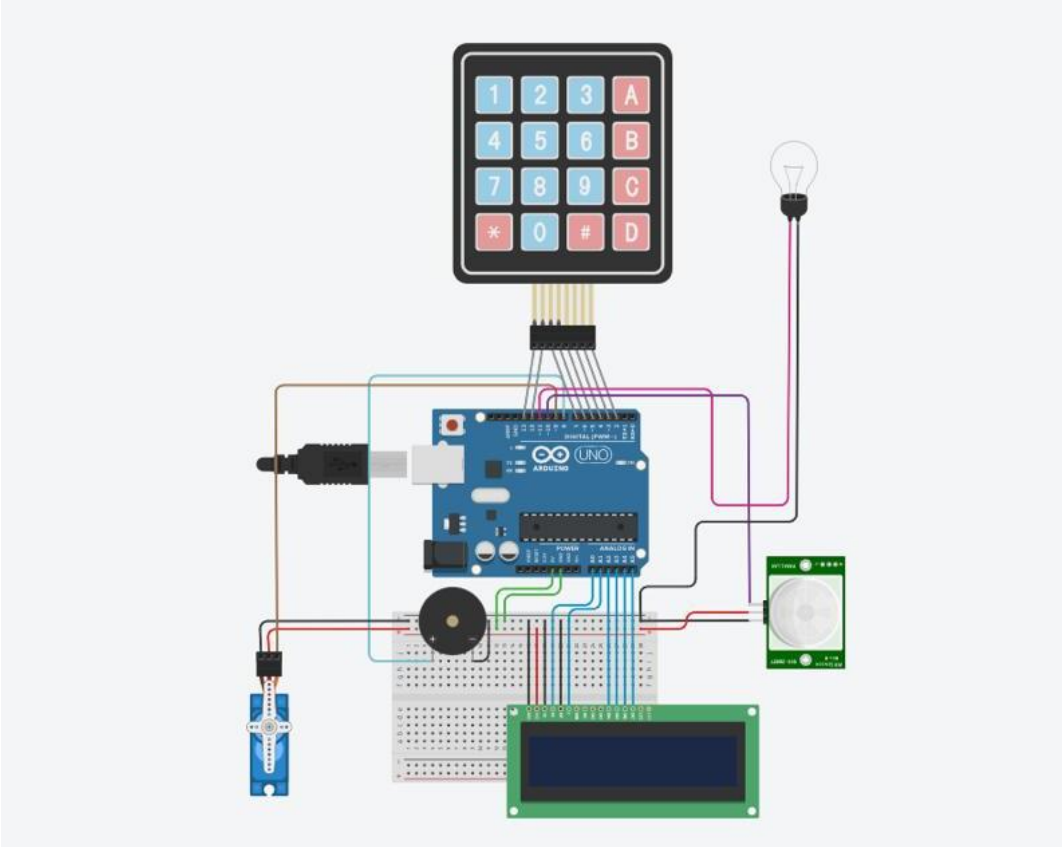
   Allows users to enter and manage passwords.

6. LEDs:

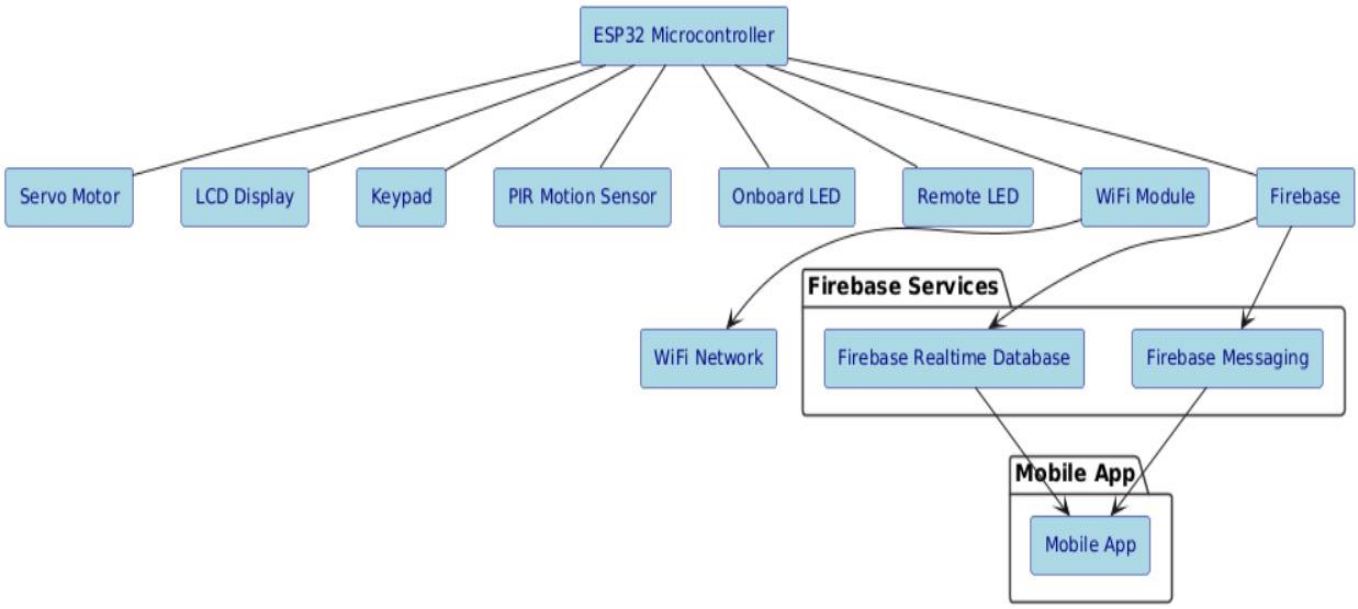
   Indicates the status of the system (onboard and remote).

7. Firebase Integration:

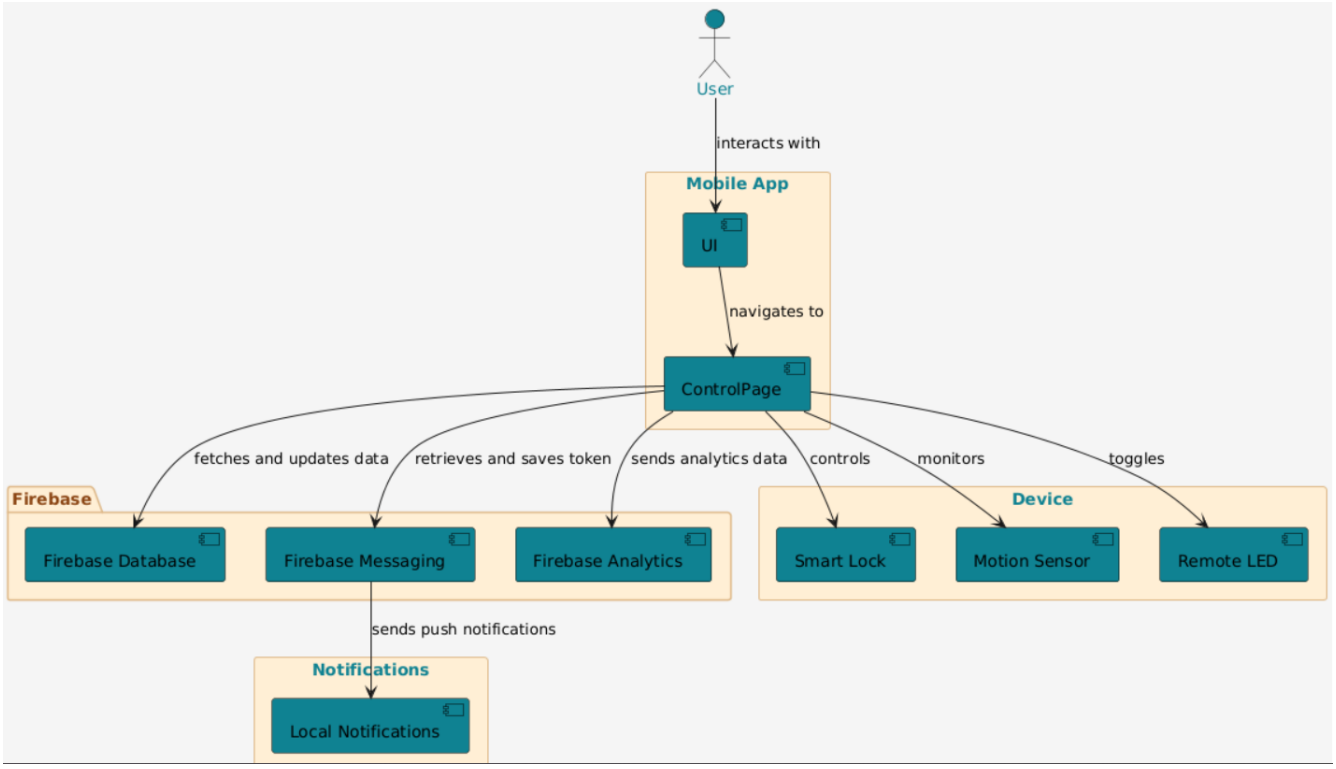
   Provides real-time data storage and retrieval.



Block Diagram



Schematic Diagram



Interfacing

- ESP32 to Servo Motor: PWM signal on a defined GPIO pin.
- ESP32 to PIR Sensor: Digital input on a defined GPIO pin.
- ESP32 to LCD Display: I2C communication.
- ESP32 to Keypad: Digital input with row and column scanning.
- ESP32 to Firebase: HTTP requests for data exchange.
- ESP32 to Wi-Fi: Standard Wi-Fi communication.

Software Design

System code

The software is developed using the Arduino IDE and consists of several modules:

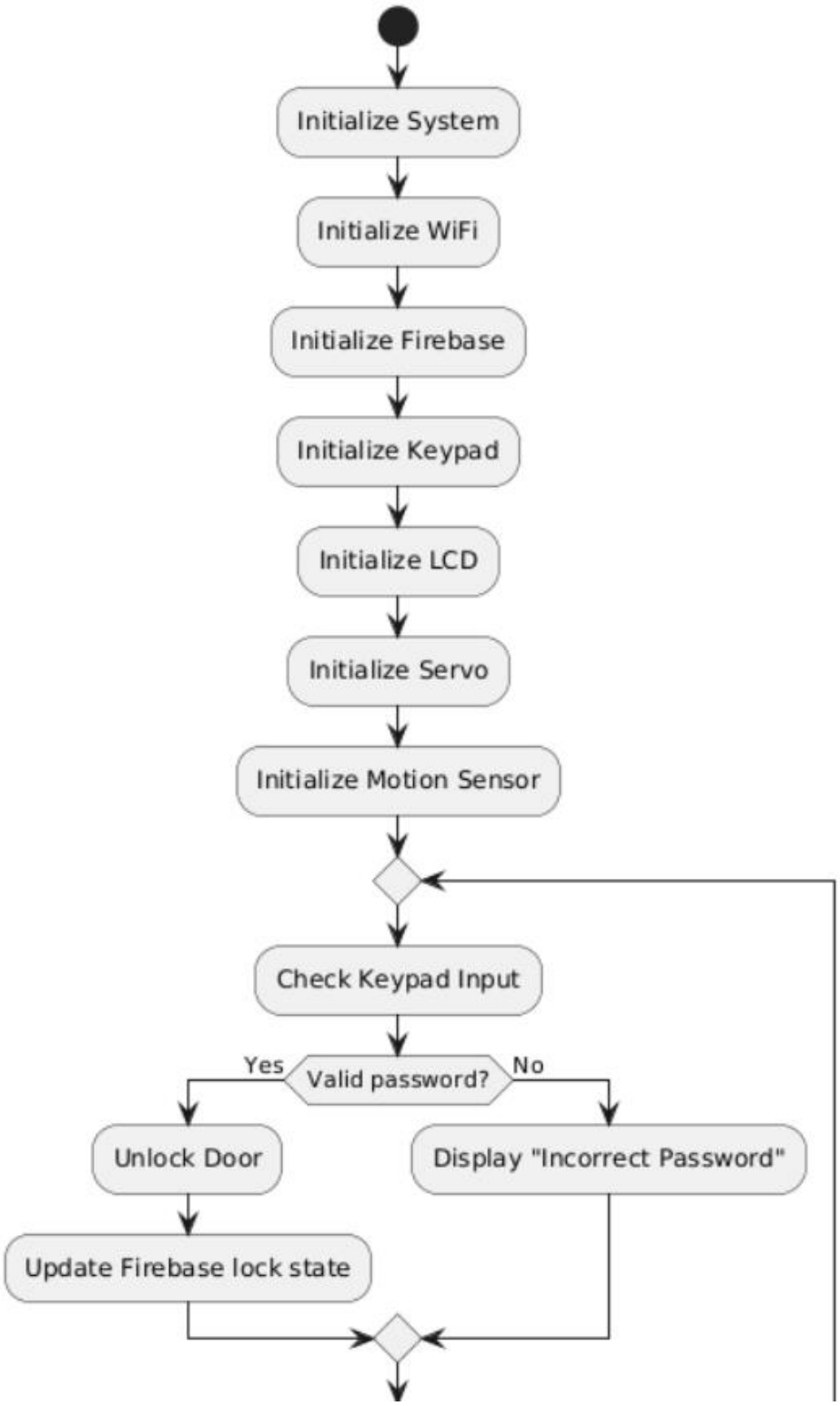
- **Keypad Input Handling**: Detects and processes password entries.
- **Motion Detection**: Monitors the PIR sensor and controls lighting.
- **WiFi and Firebase Integration**: Manages remote access and updates system status.
- **User Interface**: Updates the LCD display based on user interactions and system status.

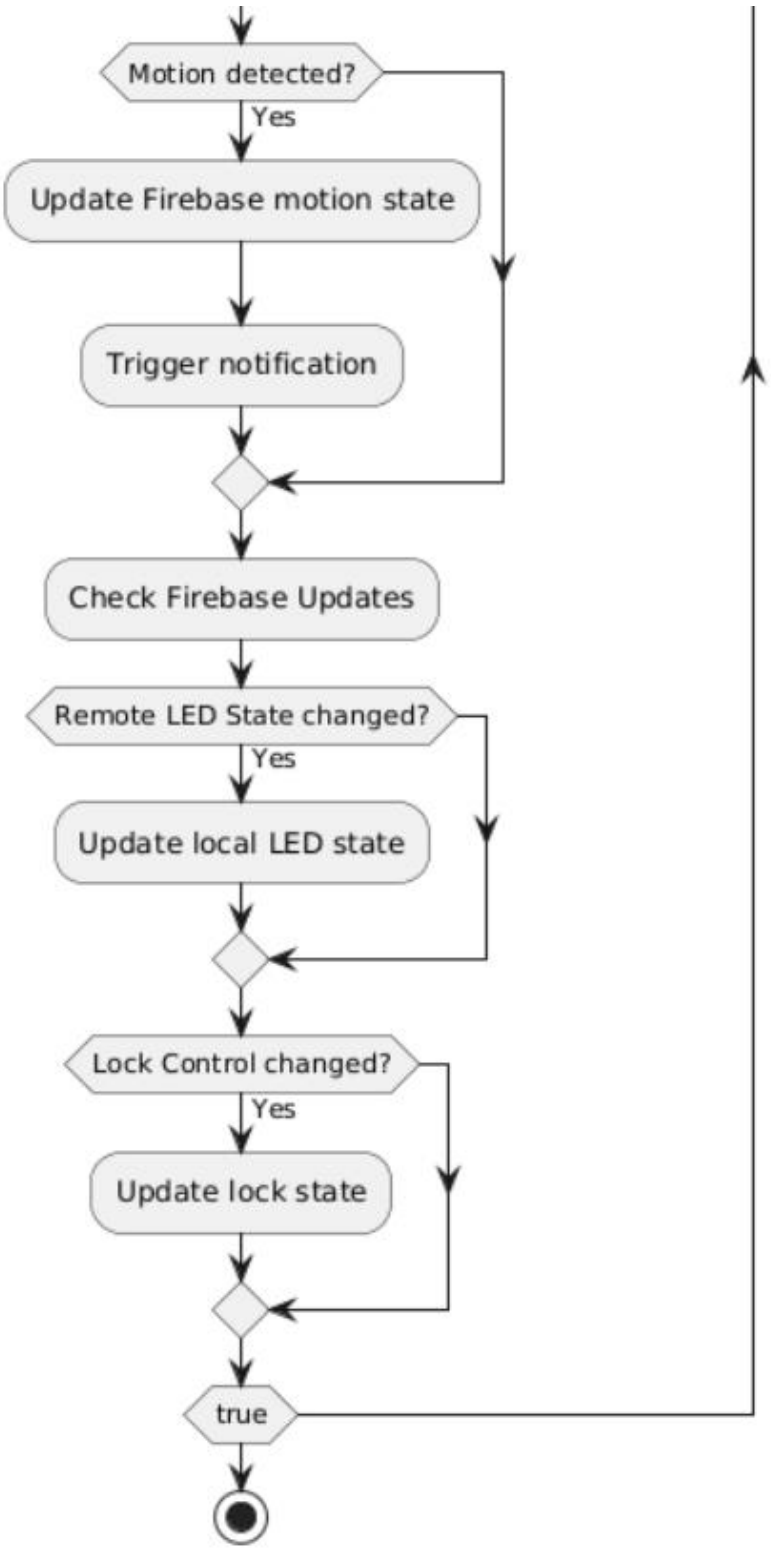
Flutter App Code

- **Firebase Initialization**: Connects the app to Firebase services.
- **Notification Handling**: Displays local notifications when messages are received.
- **Control Interface**: Provides user controls for the lock, LED, and password management.
- **Real-time Updates**: Reflects changes from Firebase in the app's UI.

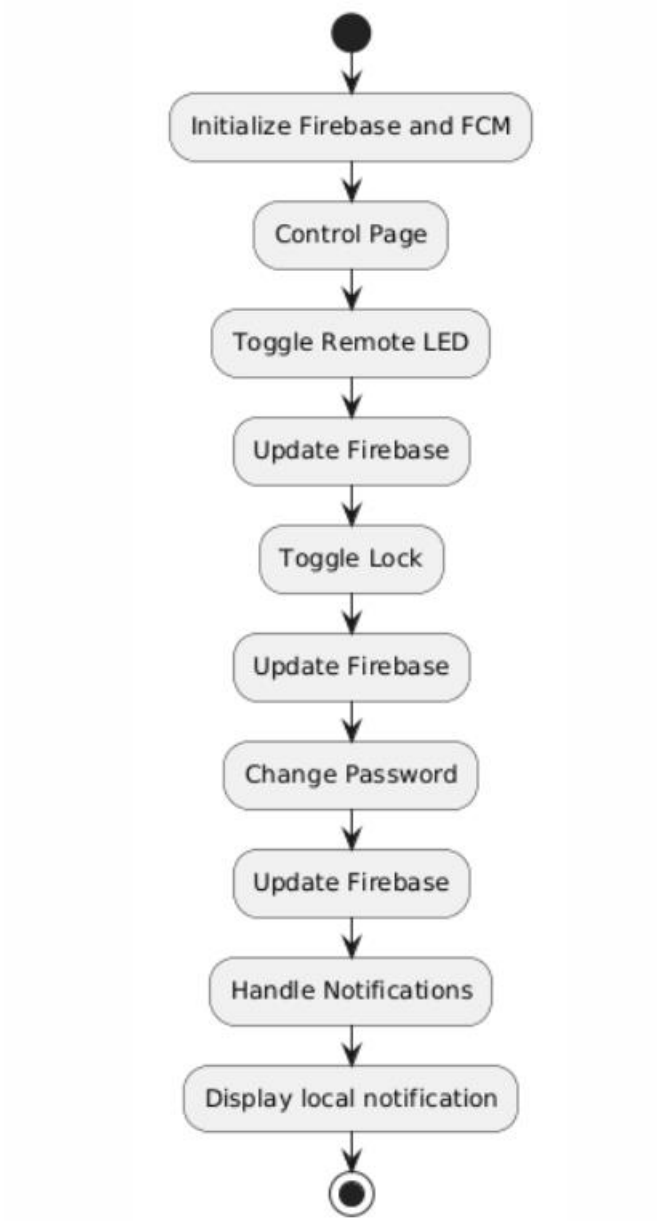
Diagrams

- **Flowchart**: Illustrates the logical flow of the program, including password verification, motion detection, and remote control.


Initialize System
Initialize WiFi
Initialize Firebase
Initialize Keypad
Initialize LCD
Initialize Servo
Initialize Motion Sensor
Check Keypad Input
Valid password?
Yes
No
Unlock Door
Display "Incorrect Password"
Update Firebase lock state

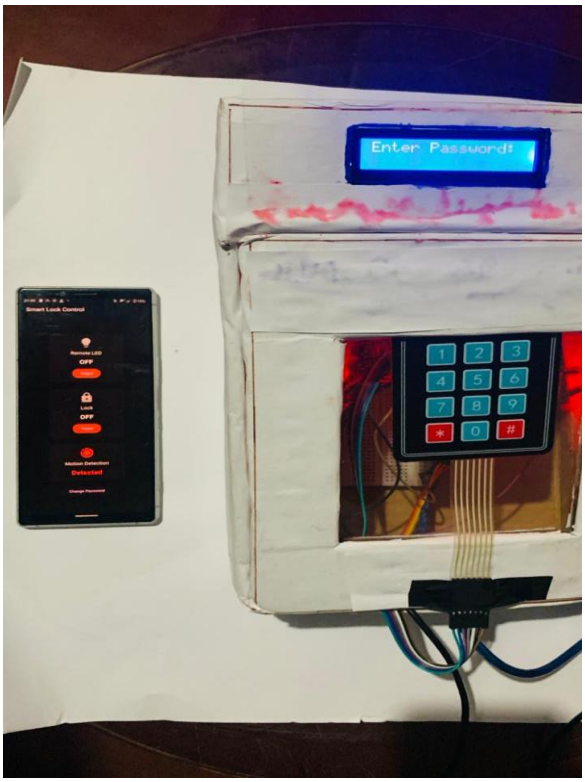

Motion detected?
Yes
Update Firebase motion state
Trigger notification
Check Firebase Updates
Remote LED State changed?
Yes
Update local LED state
Lock Control changed?
Yes
Update lock state
true

Flow chart for mobile app



4. Implementation

- Hardware Setup: Assembled the components as per the schematic, tested servo control, sensor inputs, and display outputs.
- Firmware Development: Developed and tested the ESP32 firmware for hardware control and Firebase communication.
- App Development: Created the Flutter app for user interaction, integrated Firebase services, and tested notifications.
- Testing: Conducted integration testing to ensure seamless operation between hardware and software components.



Software Development

The code is written in C++ using the Arduino IDE. Key libraries used include:

- **WiFi.h**: For WiFi connectivity.
- **FirebaseESP32.h**: For Firebase integration.
- **ESP32Servo.h**: For controlling the servo motor.
- **Wire.h**: For I2C communication with the LCD.
- **LiquidCrystal_I2C.h**: For LCD control.
- **Keypad.h**: For keypad input handling.

3. CHALLENGES AND SOLUTIONS

Challenges Faced

- Lack of component availability
- **Time Constraints** : Delays in implementation due to the strike, making it difficult for the group to work together
- **WiFi Connectivity Issues**: Ensured stable connection by implementing periodic checks and reconnections.
- **Component Integration**: Addressed hardware compatibility issues through careful selection and testing.
- **Password Security**: Stored passwords securely using EEPROM.

Solutions

- Placed advance orders for unavailable components
- Implemented robust error handling and recovery mechanisms.
- Used modular software design for easier debugging and integration.
- Conducted extensive testing to ensure reliable operation under various conditions.

4. Timeline

	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13
Literature review, System design							
Hardware assembly and circuit integration.							
Software development for password verification and UI							
Integration of motion detection and lighting							
System testing and debugging.							
Prototype finalization and documentation.							
Project report and final presentation preparation.							

The project was completed over 7 weeks

5. Components and Cost

PURCHASE COMPONENTS	QUANTITY	UNIT PRICE (LKR)	TOTAL PRICE (LKR)	DISTRIBUTOR
1602 16X2 Blue backlight LCD Display	1	480.00	480.00	tronic.lk
Arduino UNO	1	1150.00	1150.00	tronic.lk
Rotation SG90 servo motor	1	350.00	50.00	tronic.lk
3V mini buzzer	1	60.00	60.00	tronic.lk
HC-SR501 PIR sensor module	1	280.00	280.00	tronic.lk
Bread Board	1	200.00	200.00	tronic.lk
Jumper Wires	3	150.00	450.00	tronic.lk
Esp 32 microcontroller	1	1290.00	1290.00	tronic.lk
Keypad module	1	200.00	200.00	tronic.lk
Total			4460.00	

The total cost of the project was approximately 4500 LKR.

6.Reflection on Applied Knowledge

The project provided practical experience in integrating hardware with software using real-time databases and mobile applications. It enhanced understanding of microcontroller programming, IoT communication, and mobile app development, applying knowledge from embedded systems and software engineering courses.

7.Conclusion

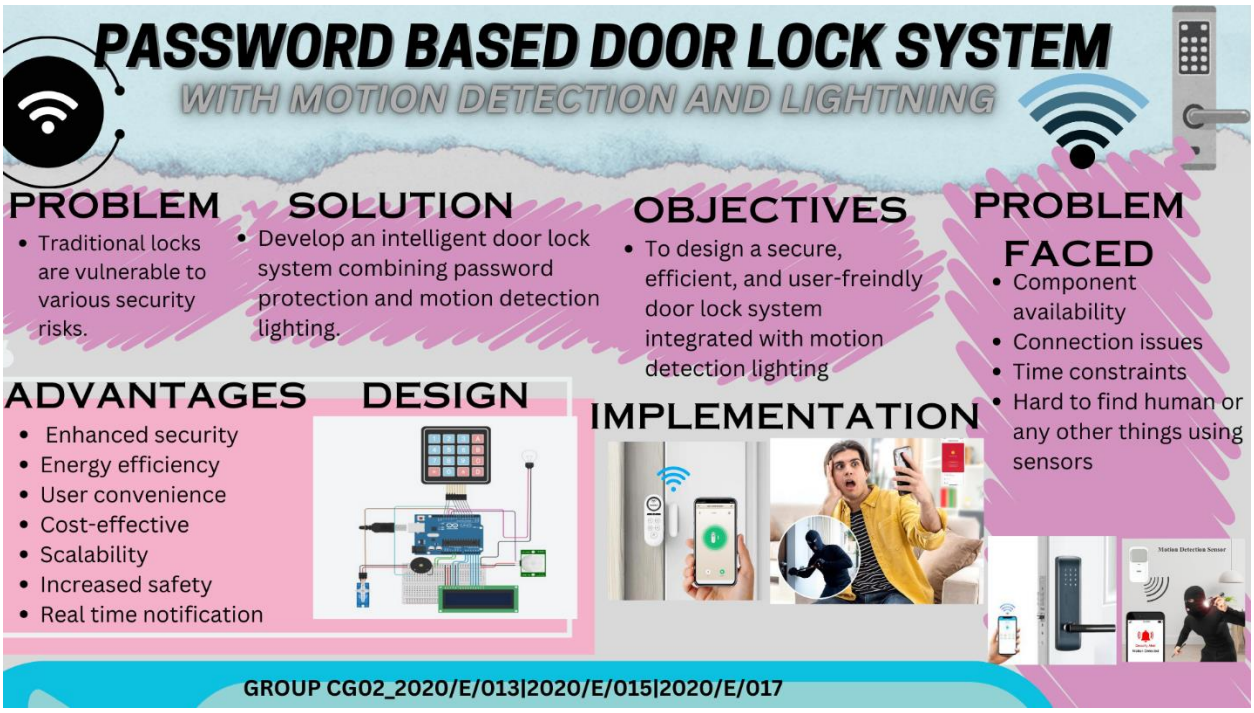
The Smart Lock System effectively combines hardware and software to provide a secure and user-friendly locking solution. The integration of Firebase and Flutter offers real-time control and monitoring capabilities, showcasing the potential of modern IoT solutions.

.

8. References

1. Arduino Official Documentation: https://www.arduino.cc/reference/en/
2. Karan, S., & Gupta, P. (2018). Arduino-based home automation and security system. International Journal of Advanced Research in Computer and Communication Engineering, 7(5), 58-62.
3. Sharma, R., & Sharma, A. (2017). Motion-activated lighting system using PIR sensor and Arduino. International Journal of Advanced Research in Computer Science and Software Engineering, 7(5), 244-248.
4. Bhavsar, A., & Bhuva, B. (2020). Password-based door lock security system using Arduino and keypad. International Journal of Engineering Research and Technology, 9(4), 1-5.

Minimized version of the poster



Appendix

System code

```
include <WiFi.h>
#include <FirebaseESP32.h>
#include <addons/TokenHelper.h>
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Preferences.h>

// WiFi credentials
const char* ssid = "Redmi 8A";
const char* password = "45362718";

// Firebase credentials
#define FIREBASE_HOST "slock-37e7b-default-rtdb.asia-southeast1.firebasedatabase.app"
#define FIREBASE_AUTH "k8gbHELXxv8DRD3qg5R6WpuwuVg4fbtKJIFGiopX"

// Define Firebase Data object
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Servo setup
Servo sg90;
#define PIN_SG90 13
const int lockedPosition = 0;
const int unlockedPosition = 90;

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

// Keypad setup
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
byte rowPins[ROWS] = {4, 16, 17, 5};
byte colPins[COLS] = {18, 19, 23};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Password setup
String passwordLock = "1234"; // Default password
String input = "";

// PIR sensor and LED setup
#define PIR_PIN 14
#define ONBOARD_LED_PIN 2
#define REMOTE_LED_PIN 15

// State variables
bool remoteLedState = false;
bool lockState = false;
bool motionDetected = false;

// Timers for periodic tasks
unsigned long lastWiFiCheck = 0;
unsigned long lastFirebaseUpdate = 0;
unsigned long lastMotionCheck = 0;
const unsigned long wifiCheckInterval = 30000; // Check WiFi every 30 seconds
const unsigned long firebaseUpdateInterval = 5000; // Update Firebase every 5 seconds
const unsigned long motionCheckInterval = 1000; // Check motion every second

// Preferences for storing data
Preferences preferences;

void setup() {
```

```
  Serial.begin(115200);
  preferences.begin("smartlock", false);

  pinMode(ONBOARD_LED_PIN, OUTPUT);
  pinMode(REMOTE_LED_PIN, OUTPUT);
  pinMode(PIR_PIN, INPUT);

  // Servo setup
  sg90.setPeriodHertz(50);
  sg90.attach(PIN_SG90, 500, 2400);

  // LCD setup
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.print("Initializing...");

  // Load saved password
  passwordLock = preferences.getString("password", passwordLock);

  // Keypad setup - Set debounce time to 0 for faster response
  keypad.setDebounceTime(0);

  // Initial connection
  connectWiFi();
  initFirebase();

  // Lock the door initially
  lockDoor();

  lcd.clear();
  lcd.print("Enter Password:");
}

void loop() {
  unsigned long currentMillis = millis();

  // Priority 1: Check keypad input
  checkKeypad();

  // Priority 2: Handle motion detection
  if (currentMillis - lastMotionCheck >= motionCheckInterval) {
    checkMotion();
    lastMotionCheck = currentMillis;
  }

  // Lower priority tasks
  if (currentMillis - lastWiFiCheck >= wifiCheckInterval) {
    checkWiFiConnection();
    lastWiFiCheck = currentMillis;
  }

  if (currentMillis - lastFirebaseUpdate >= firebaseUpdateInterval) {
    if (Firebase.ready()) {
      updateFirebaseStates();
      handleFirebaseData();
    }
    lastFirebaseUpdate = currentMillis;
  }

  // No delay in the main loop to keep it responsive
}

void connectWiFi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(500);
    Serial.print(".");
    attempts++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nFailed to connect to WiFi");
  }
}

void initFirebase() {
  config.database_url = FIREBASE_HOST;
  config.signer.tokens.legacy_token = FIREBASE_AUTH;

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  Serial.println("Connecting to Firebase...");
  int attempts = 0;
  while (!Firebase.ready() && attempts < 20) {
    Serial.print(".");
    delay(500);
    attempts++;
  }
  if (Firebase.ready()) {
    Serial.println("\nConnected to Firebase");
  } else {
    Serial.println("\nFailed to connect to Firebase");
  }
}

void checkKeypad() {
  if (keypad.getKeys()) {
    for (int i = 0; i < LIST_MAX; i++) {
      if (keypad.key[i].stateChanged) {
        switch (keypad.key[i].kstate) {
          case PRESSED:
            handleKeypadInput(keypad.key[i].kchar);
            break;
        }
      }
    }
  }

void handleKeypadInput(char key) {
  if (key == '#') {
    checkPassword();
  } else if (key == '*') {
    input = "";
    updateLCD();
  } else {
    input += key;
    updateLCD();
  }
}

void updateLCD() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");
  lcd.setCursor(0, 1);
  for (int i = 0; i < input.length(); i++) {
    lcd.print('*');
  }
}

void checkPassword() {
  if (input == passwordLock) {
```

```
  if (input == passwordLock) {
    lcd.clear();
    lcd.print("Access Granted");
    unlockDoor();
    updateFirebaseLockState(true);
    sendNotification("Door Unlocked", "Someone has entered the home.");
    delay(5000);
    lockDoor();
    updateFirebaseLockState(false);
    lcd.clear();
    lcd.print("Door Locked");
    delay(1000);
  } else {
    lcd.clear();
    lcd.print("Access Denied");
    delay(2000);
  }
  input = "";
  updateLCD();
}

void unlockDoor() {
  sg90.write(unlockedPosition);
  lockState = true;
}

void lockDoor() {
  sg90.write(lockedPosition);
  lockState = false;
}

void checkMotion() {
  bool currentMotionState = digitalRead(PIR_PIN) == HIGH;
  digitalWrite(ONBOARD_LED_PIN, currentMotionState ? HIGH : LOW);
  if (currentMotionState != motionDetected) {
    motionDetected = currentMotionState;
    updateFirebaseMotionState(motionDetected);
    if (motionDetected) {
      sendNotification("Motion Detected", "Movement detected near the smart lock.");
    }
  }
}

void updateFirebaseLockState(bool isUnlocked) {
  if (Firebase.setBoolAsync(fbdo, "/lockState", isUnlocked)) {
    Serial.println("Firebase lock state update queued successfully");
  } else {
    Serial.println("Failed to queue Firebase lock state update");
    Serial.println("REASON: " + fbdo.errorReason());
  }
}

void updateFirebaseMotionState(bool detected) {
  if (Firebase.setBoolAsync(fbdo, "/motionDetected", detected)) {
    Serial.println("Firebase motion state update queued successfully");
  } else {
    Serial.println("Failed to queue Firebase motion state update");
    Serial.println("REASON: " + fbdo.errorReason());
  }
}

void updateFirebaseStates() {
  FirebaseJson json;
  json.set("lockState", lockState);
  json.set("remoteLedState", remoteLedState);
  json.set("motionDetected", motionDetected);

  if (Firebase.updateNodeAsync(fbdo, "/", json)) {
    Serial.println("Firebase states update queued successfully");
  } else {
    Serial.println("Failed to queue Firebase states update");
    Serial.println("REASON: " + fbdo.errorReason());
  }
}

void handleFirebaseData() {
  if (Firebase.getBool(fbdo, "/remoteLedState")) {
    bool currentRemoteLEDState = fbdo.boolData();
    if (currentRemoteLEDState != remoteLedState) {
      remoteLedState = currentRemoteLEDState;
      digitalWrite(REMOTE_LED_PIN, remoteLedState ? HIGH : LOW);
      Serial.println(remoteLedState ? "Remote LED ON" : "Remote LED OFF");
    }
  }

  if (Firebase.getBool(fbdo, "/lockControl")) {
    bool shouldUnlock = fbdo.boolData();
    if (shouldUnlock != lockState) {
      if (shouldUnlock) {
        unlockDoor();
        updateFirebaseLockState(true);
        sendNotification("Door Unlocked", "Door unlocked via app.");
      } else {
        lockDoor();
        updateFirebaseLockState(false);
        sendNotification("Door Locked", "Door locked via app.");
      }
    }
  }

  if (Firebase.getString(fbdo, "/newPassword")) {
    String newPassword = fbdo.stringData();
    if (newPassword != "" && newPassword != passwordLock) {
      passwordLock = newPassword;
      preferences.putString("password", passwordLock);
      preferences.putString("password", passwordLock);
      Firebase.setString(fbdo, "/newPassword", "");
      sendNotification("Password Changed", "The smart lock password has been updated.");
    }
  }
}

void sendNotification(const char* title, const char* body) {
  FirebaseJson json;
  json.add("title", title);
  json.add("body", body);

  if (Firebase.pushAsync(fbdo, "/notifications", json)) {
    Serial.println("Notification queued successfully");
  } else {
    Serial.println("Failed to queue notification");
    Serial.println("REASON: " + fbdo.errorReason());
  }
}

void checkWiFiConnection() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi connection lost. Reconnecting...");
    WiFi.disconnect();
    WiFi.begin(ssid, password);
    // Don't wait here, check status in next iteration
  }
}
```