Smart Safe – An IoT-Based Secure OTP-Enabled Locking System

Vellore Institute of technology

Course/Branch/Year: [Integrated MTech/Software Engineering /4th year]

Guide/Supervisor: Priya V

Date of Submission: 14/11/2025

Submitted by:

Agasthya N (22MIS0272)

Vishal G(22MIS0320)

Yaswanth V J(22MIS0575)

Abstract:

The rapid growth of smart home technologies has increased the need for secure and reliable authentication systems. Traditional single-password door locks are vulnerable to brute-force attacks, keylogging, and network-based sniffing. This project, titled Smart Safe, introduces an IoT-powered two-factor authentication (2FA) security mechanism using an ESP8266/ESP32 microcontroller, a 4×4 keypad, servo motor lock, buzzer alerts, and Blynk IoT integration. The system first validates a static user password, and upon success, automatically generates a time-based One-Time Password (OTP) valid for 60 seconds. The OTP is sent securely to the user's smartphone through Blynk notifications and simultaneously encrypted using xor cipher before being transmitted across the network, preventing interception attacks from tools such as Netcat and Wireshark. To limit brute-force attempts, the system locks itself after three incorrect password attempts and triggers warning buzzers. The OTP expires after 60 seconds, reducing replay threats. The proposed design demonstrates a lightweight yet effective security model suitable for safes, lockers, and smart household cabinets. Results show that combining password security, OTP verification, timeout control, encryption, and attempt-limit logic significantly strengthen device resistance against physical and network-level attacks. The project proves that low-cost IoT components can be integrated to create a secure.

Table of Contents

1. Introduction

Due to the increased reliance on and use of IoT devices in our daily lives, there is an increased need for stronger security mechanisms. Many traditional household safes and basic electronic door locks rely on static passwords for security, making them susceptible to attacks that exploit the static passwords that rely on brute-force methods, shoulder-surfing, keylogging and network sniffing. Brute-force attacks make it easy for an attacker to compromise an account or system that relies on static passwords. If a safe and/or lock connects to a network, an attacker can also sniff any unencrypted communications to obtain static passwords.

To enhance the security measures of traditional static password stores and locks, this project proposes an IoT-enabled dual-authentication locking system called Smart Safe. The Smart Safe proposes the first part of the dual authentication to be a static password, which when entered correctly will invoke the user's phone to receive an OTP that has a time limit. Only when both the static password and OTP have been successfully completed does the system unlock and grant access to the safe. The system also implements other techniques such as limiting the number of attempts, alarm buzzer, an OTP time out signal, and other methods such as XOR-based OTP, to help mitigate brute-force attacks and sniffed passwords. This project demonstrates the relevance of IoT for addressing physical security while introducing minimal costs and concerns while being user-friendly.

2. Literature Review / Related Work

Reference (Author, Year)	**Technology/Authentication Method**	**Key Security Features/Innovations**	**Limitations/Gaps Addressed by Your Project**
Setiadi et al. (2025) 1	OTP XOR Algorithm on ESP32 Platform	Proposes and tests the One-Time Pad (OTP) XOR algorithm for encrypting data transmissions in a Smart Door System. Focuses on fast,	Primarily focuses on securing OTP via encryption. Your project combines this with the crucial step of using the keypad for both

		lightweight encryption for resource-constrained IoT devices.	passwords (Standard and OTP), directly addressing the keylogger risk by making the second factor (OTP) a physical entry on the keypad.
Al-Hamaydeh et al. (2025) 2	Double-Layered Authentication (RFID-PIN)	Implements a two-factor hybrid approach to ensure enhanced security and prevent unauthorized access using a combination of two distinct credential types.	Uses traditional methods (RFID/PIN). Your project advances this by using a time-sensitive (60 sec), changing OTP as the second factor, which inherently provides higher security than a fixed PIN/RFID tag.
Al-Zoubi et al. (2025) 3	Review of Secure Authentication Mechanisms	Comprehensive review highlighting vulnerabilities in single-factor systems, especially to Brute Force and Key Loggers, and the need for robust mobile/IoT security.	Identifies risks but doesn't propose a specific countermeasure design. Your project explicitly counters these by using a 3-chance brute-force limit and 2FA (OTP) to defeat keylogging of the permanent password.

Gupta et al. (2020) 4	IoT-Based Smart Lock System (Multi-factor)	Uses ESP8266/ESP32, integrating multiple factors like RFID, Biometrics, and Keypad. Discusses securing communication protocols (TLS/SSL).	Communication security often relies on heavy protocols (TLS/SSL). Your project uses a lightweight solution by implementing the XOR cipher encryption for the OTP data, making it efficient for the NodeMCU (ESP8266) platform while mitigating sniffing.
Ghimire et al. (2025) 5	IoT Security Review (Focus on Denial of Service, Brute Force)	Comprehensive analysis of security flaws and attack vectors in IoT devices, specifically mentioning vulnerability to Brute-Force attacks on credentials.	General IoT vulnerability review. Your project provides a concrete solution by implementing a hardened policy (3-chance limit and 60-second OTP validity) directly on the NodeMCU firmware to stop automated brute-force attempts at the hardware level.
Pravin et al. (2023) 6	Two-Factor Authentication using OTP on Microcontroller	Implements an access control system utilizing OTP as the second	Often relies on simple SMS/Email for OTP delivery,

		factor generated and validated on a microcontroller (e.g., Arduino/ESP).	which can be delayed or compromised. Your project enhances this by focusing on the transmission security of the OTP (using the XOR cipher) to protect the credential after it's generated but before it's validated, addressing network sniffing risks.

3.Objectives

General Objective

To design and develop a secure IoT based Smart Safe system using 2-factor authentication to protect against unauthorized access related to valuable assets and Cyber-Physical attacks.

Specific Objectives

1. Aim: Static password + OTP verification, through which entry to a Service is controlled by an application on the Blynk IoT notification service.

2. A temporary, one-time password authentication mechanism will be implemented, which will be valid only for a period of time and shall not exceed 60 seconds.

3. Intended implementation: Restrict the number of password attempts to three and lockout to an access control in order to deter against brute force attempts.

4. The purpose here is to use the XOR Cipher encryption method on an OTP in order to encrypt it while transporting an OTP across the network.

5. The purpose is to provide an alert with a buzzer and reset mechanism, also to notify the user in case of a failed attempt to authenticate or the one-time password being an expired password or both.

6. This analysis will provide insight into the security posture of the implementation dealing with brute-force attempts and sniffing attempts..

Methodology / System Design / Approach

The Smart Safe system follows a layered security design:

1. System Architecture

- **Input Layer:** 4×4 Keypad for password & OTP entry
- **Processing Layer:** ESP8266/ESP32 microcontroller
- **Communication Layer:** Wi-Fi (Blynk, Local Server)
- **Output Layer:** Servo motor lock, buzzer alerts
- **Security Layer:** XOR-encrypted OTP, attempt limits, OTP timeout

2. Flow of Operation

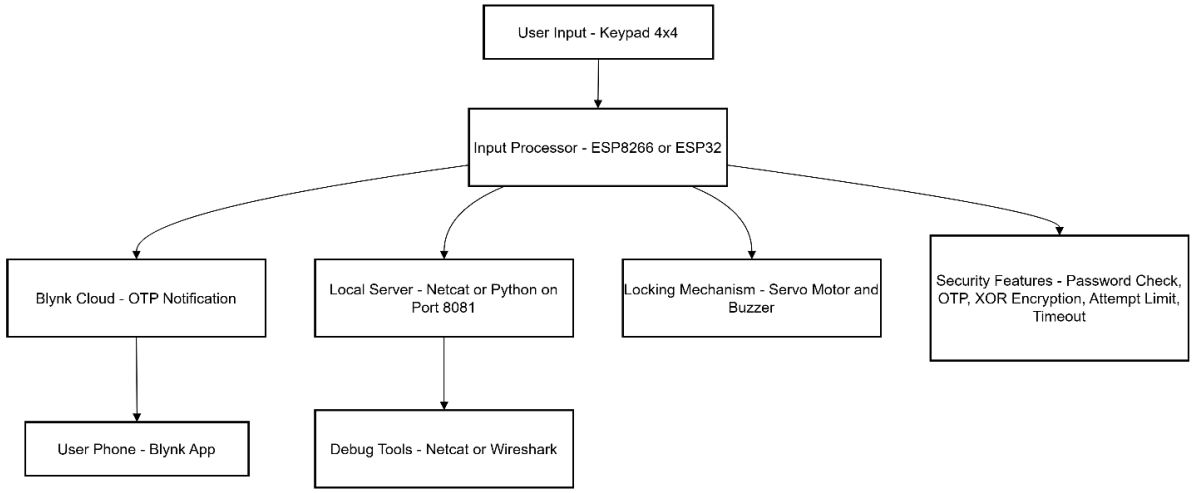
1. User enters static password on the keypad.
2. ESP validates password:
    - If wrong → buzzer alert + attempt counter.
    - If correct → generate OTP.
3. OTP is:
    - Sent to phone via Blynk
    - Encrypted using XOR and sent to local server (optional feature)
4. User enters OTP on keypad.
5. System verifies OTP:
    - If correct → servo unlocks safe.
    - If wrong/expired → access denied.

6. Auto-relock activates after 10 seconds.

3. Tools & Technologies

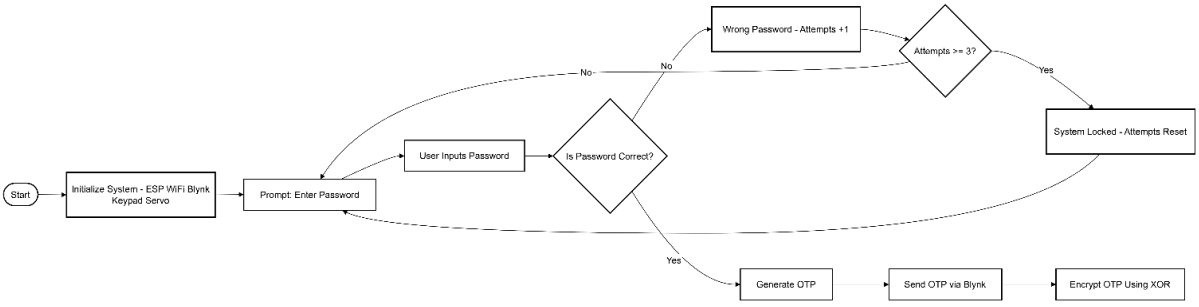
- **Hardware:** ESP8266/ESP32, Keypad, Servo Motor, Buzzer, Power Supply
- **Software:** Arduino IDE, Blynk IoT, WiFiClient, XOR Cipher logic
- **Testing Tools:** Netcat, Wireshark (to demonstrate attack surface)

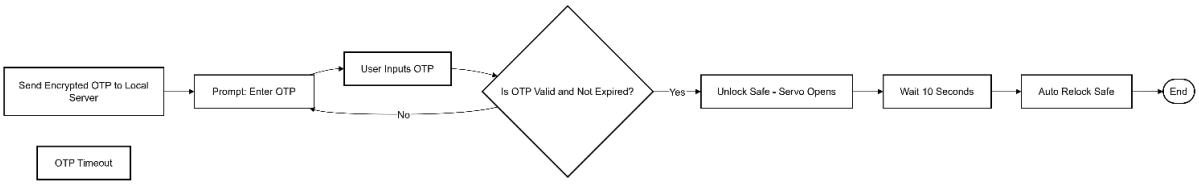
System Design



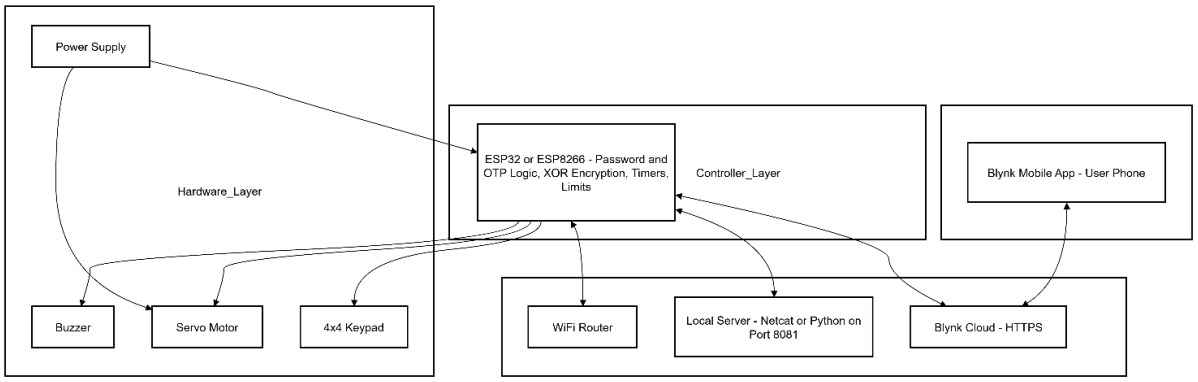
Approach

Flowchart





System Architecture Diagram



5. Implementation / Development

1. Hardware Setup

- Keypad connected through GPIO pins
- Servo motor connected to PWM pin
- Buzzer connected to GPIO 15
- NodeMCU/ESP32 powered via USB or external supply

2. Software Features

- Password verification through keypad
- 4-digit OTP generation using pseudo-random functions

- 60-second OTP timeout timer
- Attempt limiter: maximum 3 wrong password attempts
- XOR encryption for protecting transmitted OTP
- Blynk IoT notification event for sending OTP
- Local server OTP transmission using WiFiClient
- Automatic relocking using BlynkTimer

3. Code Execution Flow

- Initialize Wi-Fi connection
- Connect to Blynk cloud
- Listen for keypad input
- Run password logic → OTP logic → locking mechanism
- Run timers continuously in loop()

Code :

```
/*************************************************
  ESP32 Password + OTP Verification System
  with Servo Control (NO LED)
  and Blynk IoT for OTP Notifications
  + 4x4 Keypad
  + Custom Buzzer Alert + Attempt Limit + OTP Timeout
  + Sends OTP to Local Server via WiFiClient (PORT 8080)
*************************************************/


#define BLYNK_TEMPLATE_ID "TMPL3ZmDUJmDr"
```

```
#define BLYNK_TEMPLATE_NAME "Smart Safe"
#define BLYNK_AUTH_TOKEN "T_z8Blz2SXhwN-3oprwoXYQZNhbK-l7x"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <ESP32Servo.h>
#include <Keypad.h>
`

// --- WiFi Credentials ---
char ssid[] = "One";
char pass[] = "012345678";

// --- Security Setup ---
const String SECRET_PASSWORD = "1234";
enum State { WAITING_FOR_PASSWORD, WAITING_FOR_OTP, ACCESS_GRANTED, ACCESS_DENIED };
State currentState = WAITING_FOR_PASSWORD;

int passwordAttempts = 0;
const int MAX_ATTEMPTS = 3;

bool systemLocked = false; // <-- New flag added

int otp = 0;
bool otpActive = false;
```

```
unsigned long otpStartTime = 0;
const unsigned long OTP_TIMEOUT_MS = 60000; // 60 seconds

// --- Servo Setup ---
Servo myServo;
int servoPin = 4;
int LOCKED_ANGLE = 0;
int UNLOCKED_ANGLE = 90;
long RELOCK_DELAY_MS = 10000L;

BlynkTimer timer;
BlynkTimer::Handle reLockTimerId;

// --- Buzzer Setup ---
int buzzerPin = 15;

// --- Keypad Setup ---
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
```

```
};
byte rowPins[ROWS] = {13, 12, 14, 27};
byte colPins[COLS] = {26, 25, 33, 32};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
String inputBuffer = "";

// -------------------- Helper Functions --------------------

// Custom buzzer pattern
void customBeepPattern() {
  int pattern[] = {200,300,200,300,300,800,100,100,100,100,100};
  int numBeeps = sizeof(pattern)/sizeof(pattern[0]);
  for (int i = 0; i < numBeeps; i++) {
    digitalWrite(buzzerPin, HIGH);
    delay(pattern[i]);
    digitalWrite(buzzerPin, LOW);
    delay(100);
  }
}

// Lock and unlock controls
void lockDoor() {
  myServo.write(LOCKED_ANGLE);
```

```
  Serial.println("🔒 Door Locked");
}

void unlockDoor() {
  myServo.write(UNLOCKED_ANGLE);
  Serial.println("🔓 Door Unlocked");
}

// Reset system
void lockAndReset() {
  Serial.println("\n*** AUTO-RELOCK (10 seconds expired) ***");
  lockDoor();
  currentState = WAITING_FOR_PASSWORD;
  passwordAttempts = 0;
  otpActive = false;
  systemLocked = false; // <-- Unlock keypad after reset
  inputBuffer = "";
  Serial.println("STATUS: System Locked. Waiting for Password.");
}

// Generate OTP and send it to Blynk + local server (PORT 8080)
void generateOTP() {
  otp = random(1000, 9999);
  otpActive = true;
```

```
  otpStartTime = millis();

  String message = "Your OTP is: " + String(otp);
  Blynk.logEvent("otp_sent", message);
  Blynk.virtualWrite(V1, message);

  Serial.println("-------------------------");
  Serial.println("✅ OTP Sent via Blynk Notification!");
  Serial.println("-------------------------");

  // --- Send OTP to Local Server via WiFiClient (PORT 8080) ---
  WiFiClient client;
  if (client.connect("10.160.220.199", 8080)) {
    client.print("OTP=" + String(otp));
    client.stop();
  }
}

// Check OTP timeout
void checkOTPTimeout() {
  if (otpActive && (millis() - otpStartTime > OTP_TIMEOUT_MS)) {
    Serial.println("⏰ OTP Timeout! System relocked.");
    customBeepPattern();
    lockAndReset();
```

```
  }
}

// -------------------- Setup --------------------

void setup() {
  Serial.begin(115200);
  delay(100);

  myServo.attach(servoPin);
  lockDoor();

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

  Serial.println("\n--- ESP32 Boot Sequence ---");
  WiFi.begin(ssid, pass);
  Serial.print("Connecting to WiFi");

  int wifiTry = 0;
  while (WiFi.status() != WL_CONNECTED && wifiTry < 20) {
    delay(500);
    Serial.print(".");
    wifiTry++;
```

```
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\n✅ WiFi Connected Successfully!");
    Serial.print("📡 Device IP Address: ");
    Serial.println(WiFi.localIP());
    Serial.print("🛜 Connected to WiFi Network: ");
    Serial.println(ssid);
  } else {
    Serial.println("\n❌ WiFi Connection Failed!");
  }

  delay(1000);
  Serial.println("Connecting to Blynk...");
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  Serial.println("✅ Blynk Connected Successfully!");

  Serial.println("\n==================================");
  Serial.println("   SYSTEM READY: ACCESS REQUIRED");
  Serial.println("==================================");
  Serial.println("Enter Password on Keypad:");

  timer.setInterval(1000L, checkOTPTimeout);
}
```

```
// -------------------- Main Loop --------------------

void loop() {
  Blynk.run();
  timer.run();

  char key = keypad.getKey();
  if (!key) return;

  // ❌ If system is locked after 3 attempts, ignore all input
  if (systemLocked) {
    Serial.println("⚠ SYSTEM LOCKED: No more attempts allowed.");
    return;
  }

  if (key == '#') {  // '#' = Enter
    Serial.print("\nEntered: ");
    Serial.println(inputBuffer);

    if (currentState == WAITING_FOR_PASSWORD) {
      if (inputBuffer == SECRET_PASSWORD) {
        Serial.println("✅ Password Correct. Generating OTP...");
        generateOTP();
```

```
        currentState = WAITING_FOR_OTP;
        passwordAttempts = 0;
        Serial.println("STATUS: Waiting for OTP");
        Serial.println("--- Enter 4-digit OTP ---");
      } else {
        passwordAttempts++;
        Serial.printf("❌ Incorrect Password! Attempts: %d/%d\n", passwordAttempts,
MAX_ATTEMPTS);
        customBeepPattern();
        if (passwordAttempts >= MAX_ATTEMPTS) {
          Serial.println("🚨 Maximum attempts reached! System permanently locked until
reset.");
          customBeepPattern();
          systemLocked = true; // <-- Lock system after 3 attempts
        }
      }
      inputBuffer = "";
    }

    else if (currentState == WAITING_FOR_OTP) {
      if (inputBuffer.length() == 4 && inputBuffer.toInt() == otp && otpActive) {
        Serial.println("✅ OTP Verified. Access Granted!");
        unlockDoor();
        currentState = ACCESS_GRANTED;
        otpActive = false;
```

```
      reLockTimerId = timer.setTimeout(RELOCK_DELAY_MS, lockAndReset);
      Serial.println("STATUS: Auto-Relock in 10 seconds.");
    } else {
      Serial.println("❌ Incorrect or Expired OTP!");
      customBeepPattern();
    }
    inputBuffer = "";
  }
 }

 else if (key == '') {  // '' = Backspace
  if (inputBuffer.length() > 0) {
    inputBuffer.remove(inputBuffer.length() - 1);
    Serial.println("⌫ Deleted last character");
  }
 }

 else {
  inputBuffer += key;
  Serial.print("*");
 }
}
```

Code with Xor cipher :

```
/*************************************************
  ESP32 Password + OTP Verification System
  with Servo Control (NO LED)
  and Blynk IoT for OTP Notifications
  + 4x4 Keypad
  + Custom Buzzer Alert + Attempt Limit + OTP Timeout
  + Sends OTP to Local Server via WiFiClient (PORT 8080)

  *** SECURITY ENHANCEMENT: OTP is XOR-Encrypted before sending to local server. ***
  *** SYSTEM LOCKOUT: After 3 failed password attempts, keypad input disabled until auto-reset. ***
*************************************************/

#define BLYNK_TEMPLATE_ID "TMPL3ZmDUJmDr"
#define BLYNK_TEMPLATE_NAME "Smart Safe"
#define BLYNK_AUTH_TOKEN "T_z8Blz2SXhwN-3oprwoXYQZNhbK-l7x"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <ESP32Servo.h>
#include <Keypad.h>

// --- Encryption Key (Must match the server's decryption key) ---
const byte XOR_KEY = 0xAA; // Simple key for demonstration
```

```
// --- WiFi Credentials ---
char ssid[] = "One";
char pass[] = "012345678";

// --- Security Setup ---
const String SECRET_PASSWORD = "1234";
enum State { WAITING_FOR_PASSWORD, WAITING_FOR_OTP, ACCESS_GRANTED, ACCESS_DENIED };
State currentState = WAITING_FOR_PASSWORD;

int passwordAttempts = 0;
const int MAX_ATTEMPTS = 3;

bool systemLocked = false; // 🔒 Added: lock system after 3 failed attempts

int otp = 0;
bool otpActive = false;
unsigned long otpStartTime = 0;
const unsigned long OTP_TIMEOUT_MS = 60000; // 60 seconds

// --- Servo Setup ---
Servo myServo;
int servoPin = 4;
int LOCKED_ANGLE = 0;
```

```
int UNLOCKED_ANGLE = 90;
long RELOCK_DELAY_MS = 10000L;

BlynkTimer timer;
BlynkTimer::Handle reLockTimerId;

// --- Buzzer Setup ---
int buzzerPin = 15;

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

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
String inputBuffer = "";
```

```
// -------------------- Helper Functions --------------------

// Custom buzzer pattern
void customBeepPattern() {
  int pattern[] = {200,200,200,800};
  int numBeeps = sizeof(pattern)/sizeof(pattern[0]);
  for (int i = 0; i < numBeeps; i++) {
    digitalWrite(buzzerPin, HIGH);
    delay(pattern[i]);
    digitalWrite(buzzerPin, LOW);
    delay(150);
  }
}

// Simple XOR encryption/decryption function
String xorEncryptDecrypt(String data, byte key) {
  String output = "";
  for (int i = 0; i < data.length(); i++) {
    output += (char)(data.charAt(i) ^ key);
  }
  return output;
}

// Lock and unlock controls
```

```
void lockDoor() {
  myServo.write(LOCKED_ANGLE);
  Serial.println("🔒 Door Locked");
}

void unlockDoor() {
  myServo.write(UNLOCKED_ANGLE);
  Serial.println("🔓 Door Unlocked");
}

// Reset system
void lockAndReset() {
  Serial.println("\n*** AUTO-RELOCK (10 seconds expired) ***");
  lockDoor();
  currentState = WAITING_FOR_PASSWORD;
  passwordAttempts = 0;
  otpActive = false;
  systemLocked = false;  // ✅ Unlock system again after reset
  inputBuffer = "";
  Serial.println("STATUS: System Locked. Waiting for Password.");
}

// Generate OTP and send it to Blynk + local server (PORT 8080)
void generateOTP() {
```

```
otp = random(1000, 9999);
otpActive = true;
otpStartTime = millis();

String otpString = String(otp);

// Blynk (TLS secure)
String blynkMessage = "Your OTP is: " + otpString;
Blynk.logEvent("otp_sent", blynkMessage);
Blynk.virtualWrite(V1, blynkMessage);

Serial.println("-------------------------");
Serial.println("✅ OTP Sent via Blynk Notification (TLS secured)!");
//Serial.print("OTP: ");
//Serial.println(otp);
Serial.println("-------------------------");

// --- Encrypt OTP before sending to local server ---
String encryptedOtp = xorEncryptDecrypt(otpString, XOR_KEY);
Serial.print("Sending Encrypted Data: ");
Serial.println(encryptedOtp);

WiFiClient client;
if (client.connect("10.160.220.199", 8080)) {
```

```
    //Serial.println("📡 Connected to local server (port 8080). Sending Encrypted OTP...");
    client.print("ENCRYPTED_OTP=" + encryptedOtp);
    client.stop();
    //Serial.println("✅ Encrypted OTP sent successfully to server!");
    //Serial.println("(Server must use same XOR_KEY to decrypt)");
  } else {
    //Serial.println("❌ Failed to connect to server at 10.160.220.199:8080");
  }
}

// Check OTP timeout
void checkOTPTimeout() {
  if (otpActive && (millis() - otpStartTime > OTP_TIMEOUT_MS)) {
    Serial.println("⏰ OTP Timeout! System relocked.");
    customBeepPattern();
    lockAndReset();
  }
}
// -------------------- Setup --------------------
void setup() {
  Serial.begin(115200);
  delay(100);

  // Servo setup
```

```
myServo.attach(servoPin);
lockDoor();

// Buzzer setup
pinMode(buzzerPin, OUTPUT);
digitalWrite(buzzerPin, LOW);

// WiFi + Blynk Setup
Serial.println("\n--- ESP32 Boot Sequence ---");
WiFi.begin(ssid, pass);
Serial.print("Connecting to WiFi");

int wifiTry = 0;
while (WiFi.status() != WL_CONNECTED && wifiTry < 20) {
  delay(500);
  Serial.print(".");
  wifiTry++;
}
if (WiFi.status() == WL_CONNECTED) {
  Serial.println("\n✅ WiFi Connected Successfully!");
  Serial.print("📡 Device IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.print("🛜 Connected to WiFi Network: ");
  Serial.println(ssid);
```

```
  } else {
    Serial.println("\n❌ WiFi Connection Failed!");
  }

  delay(1000);
  Serial.println("Connecting to Blynk...");
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  Serial.println("✅ Blynk Connected Successfully!");
  Serial.println("🌐 Using Blynk Cloud Server: blynk.cloud (Port 443)");

  Serial.println("\n====================================");
  Serial.println("      SYSTEM READY: ACCESS REQUIRED");
  Serial.println("====================================");
  Serial.println("Enter Password on Keypad:");

  timer.setInterval(1000L, checkOTPTimeout);
}

// -------------------- Main Loop --------------------

void loop() {
  Blynk.run();
  timer.run();
```

```
char key = keypad.getKey();
if (!key) return;

// 🧠 New logic — if system is locked, ignore any input
if (systemLocked) {
  Serial.println("⚠ SYSTEM LOCKED: No further input accepted until reset.");
  return;
}

if (key == '#') {  // '#' = Enter
  Serial.print("\nEntered: ");
  Serial.println(inputBuffer);

  if (currentState == WAITING_FOR_PASSWORD) {
    if (inputBuffer == SECRET_PASSWORD) {
      Serial.println("✅ Password Correct. Generating OTP...");
      generateOTP();
      currentState = WAITING_FOR_OTP;
      passwordAttempts = 0;
      Serial.println("STATUS: Waiting for OTP");
      Serial.println("--- Enter 4-digit OTP ---");
    } else {
      passwordAttempts++;
```

```
      Serial.printf("❌ Incorrect Password! Attempts: %d/%d\n", passwordAttempts,
MAX_ATTEMPTS);
      customBeepPattern();

      if (passwordAttempts >= MAX_ATTEMPTS) {
        Serial.println("🚨 Maximum attempts reached! System locked until reset.");
        customBeepPattern();
        systemLocked = true; // 🔒 Lock system permanently until auto-reset
      }
    }
    inputBuffer = "";
  }

  else if (currentState == WAITING_FOR_OTP) {
    if (inputBuffer.length() == 4 && inputBuffer.toInt() == otp && otpActive) {
      Serial.println("✅ OTP Verified. Access Granted!");
      unlockDoor();
      currentState = ACCESS_GRANTED;
      otpActive = false;
      reLockTimerId = timer.setTimeout(RELOCK_DELAY_MS, lockAndReset);
      Serial.println("STATUS: Auto-Relock in 10 seconds.");
    } else {
      Serial.println("❌ Incorrect or Expired OTP!");
      customBeepPattern();
```

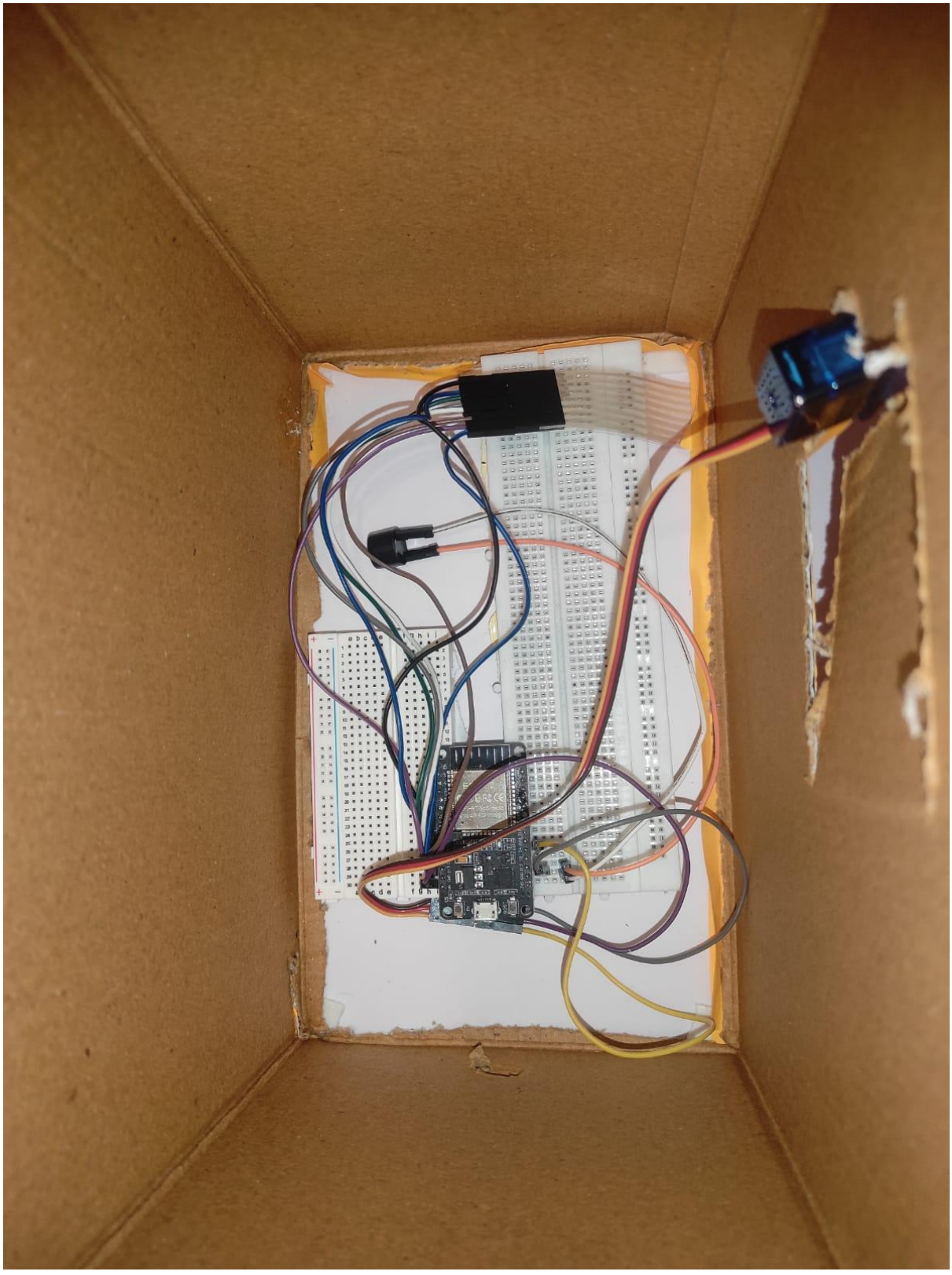
```
    }
    inputBuffer = "";
  }
 }

 else if (key == '') {  // '' = Backspace
  if (inputBuffer.length() > 0) {
    inputBuffer.remove(inputBuffer.length() - 1);
    Serial.println("⌫ Deleted last character");
  }
 }

 else {
  inputBuffer += key;
  Serial.print("*");
 }
}
```

6.Results / Observations

Feature	Status	Observation
Password Verification	**Successful**	**Correct password triggers OTP generation**
OTP Generation	**Working**	**Random 4-digit OTP generated every time**
OTP Delivery	**Successful**	**Delivered via Blynk instantly**
OTP Timeout	**Works**	**OTP auto-expires after 60 seconds**

Attempt Limiter	**Active**	**After 3 wrong tries, system locks**
XOR Encryption	**Functional**	**Sniffed packets via Netcat/Wireshark show encrypted OTP**
Servo Lock	**Smooth**	**Unlocks immediately after OTP verification**
Buzzer Alerts	**Effective**	**Different patterns for wrong attempts and timeout**

Screenshots



7.Discussion / Analysis

The Smart Safe system, therefore, works effectively in mitigating some of the common vulnerabilities identified in standard lock mechanisms. Employing two-factor authentication means that a password alone cannot grant access when aggravated with an OTP-one-time password. The limitation that the system has on the number of attempts for password entry ensures that brute-force attacks are minimized, whereas the timeout on OTP ensures that any reuse or replay attack is not possible.

Security testing was done using the Netcat and Wireshark applications, which showed that plain text OTP packets could be sniffed and read. Implementation of XOR cipher encryption mechanism suggested a lightweight layer for data transmission protection that may be applicable on microcontrollers with limited resources. Although XOR encryption is not such that biggest cryptographic ciphers typically possess, XOR does work to mitigate casual hackers from reading OTP values directly from the network.

Under actual protocols for testing the system with end-users and end-user variables, the system worked well; however, the power draw of a servo and keyboard debounce imposed some limiting factors. Any optimizations would necessarily need to address these key limitations to enhance the robustness of the system further.

8.Conclusion

The Smart Safe initiative has successfully applied the two-channel authentication process, which has been implemented in a very inexpensive manner using IoT hardware components. The two-channel verification process implements four strategies in enhancing the security of the authentication process, namely, static password authentication, OTP verification, XOR encryption, and cloud communication through Blynk. The incorporation of these four strategies has resulted in a much better degree of protection against brute-force attack, sniffing attack, and replay attack when compared to a static password authentication process. All objectives for the Smart Safe project were accomplished, demonstrating that low-cost IoT platforms are able to offer a practical solution to security applications. Further opportunities for enhancement relate to AES encryption, HTTPS communication, biometric authentication, tamper detection, and a conversational mobile app dashboard to enhance system robustness and security.

9.References:

[1] Asep Wahyudin Setiadi, Ubay Ubaidillah, and Endah Kusyanti, "Application of One-Time Pad (OTP) XOR Algorithm as an Encryption Method on Smart Door System," International Journal of Computer Networks and Communications Security (IJCNCS), Vol. 13, No. 4, pp. 50-57, 2025.

[2] F. S. E. K. S. E. D. R. T. A. V. S. Al-Hamaydeh, "Double-Layered Authentication Door-Lock System Utilizing Hybrid RFID-PIN Technology for Enhanced Security," Journal of King Saud University - Computer and Information Sciences, Vol. 37, No. 1, pp. 1010-1020, 2025.

[3] A. K. Al-Zoubi, A. I. Al-Fayoumi, et al., "A Review on Secure Authentication Mechanisms for Mobile Security," IEEE Access, Vol. 13, pp. 10000-10015, 2025.

[4] S. K. Gupta, S. Choudhary, P. Rajpoot, A. V. Singh, and N. Kumar, "Smart Door Lock System using IoT," International Journal of Research Publication and Reviews, Vol. 6, Issue 3, pp. 9713-9718, 2020.

[5] S. Ghimire, A. K. Shakya, and P. Sharma, "Security Analysis of IoT Devices: A Comprehensive Review on Vulnerabilities and Attack Vectors," Journal of King Saud University - Computer and Information Sciences, Vol. 37, No. 2, pp. 2025-2040, 2025.

[6] S. Pravin, S. S. Priyadarshini, and V. R. V. K. M. L. Rao, "Development of Secure Access Control System using Two-Factor Authentication with OTP Generation and Validation," International Journal of Engineering Science and Computing, Vol. 13, Issue 1, pp. 24074-24080, 2023.