EEC-243 Microprocessors

Project Report

Smart Door Lock and Lighting System

Authors:
Team members-ID:

1. AbdulRahman Mamdouh Khalil- 19015935
2. Ahmed Muhammed Saeed- 19015328
3. Amr Ramadan Agami- 19016110
4. Mariam Ahmed Fathy- 19016627
5. Nada Atia Eid- 19016780

Supervisors:
Dr. Nayera Sadek
Eng. Nour

May 21, 2022(Spring)

Contents

Abstract

Doors play an important role in home security. To secure the house, the occupants of the house will always have the door locked. This project presents application Door Security System which is based on Android application using Internet of Things (IoT) technology of ESP32 microcontroller to set the password of the door, controlling the door and increasing security in a house. In addition, controlling the home lightning system by indicating the door status which makes our lives easier, more convenient, and more comfortable.

Introduction

Over the past decade, Internet of Things (IoT) and smart home technology have become increasingly integrated into our everyday lives. Smart homes filled with connected products are loaded with possibilities to make our lives easier, more convenient, and more comfortable. It gives a brand new level of control over our household. Not only switching on and off our appliances, but also controlling the full range of functionality on mobile phones.

The IoT makes the shift from functionality to connectivity and data-driven decision making, meaning that a device can become more useful if it is interconnected with other devices. However, the IoT is not simply a bunch of devices and sensors connected to each other in a wired or wireless network – it is a dense integration of the virtual and the real world, where the communication between people and devices takes place. It can be considered an interwoven medium of networks of different sizes, that makes up a large global network.
This project focuses on the security of smart homes brought by the integration with mobile phone as the main controller.

List of Components and budget

The used components showed in Figure 1.

- ESP32 WROOM -DA (250 EGP)
  - 2.4 GHz WiFi + Bluetooth® + Bluetooth LE module.
  - Built around ESP32 series of SoCs, Xtensa® dualcore 32bit LX6 microprocessor.
  - 8 MB flash.
  - 24 GPIOs, rich set of peripherals.
  - Onboard dual PCB antennas
- SG90 micro Servo Motor (65 EGP)
  - Operating speed: 0.12second/ 60degree (4.8V no load)
  - Stall Torque (4.8V): 17.5oz /in (1kg/cm)
  - Operating voltage: 3.0V - 7.2V
- 4x4 keypad Module (20 EGP)
- Wires (0.5 EGP)
- 2 Resistors (0.5 EGP)
- 2 Leds (7 EGP)
- Breadboard (20 EGP)
- USB cable (10 EGP)
- Battery 9v (15 EGP)

Total cost = 388 EGP

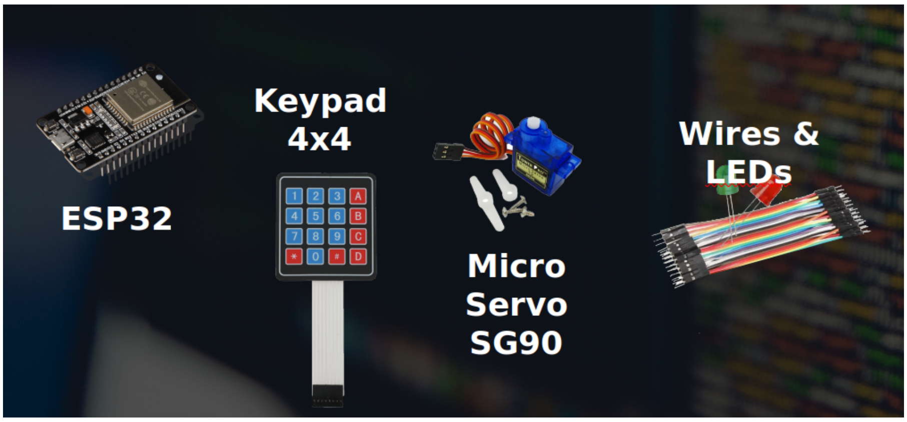


Figure 2.1: Circuit Components.

Hardware

ESP connection

The ESP32 is actually a series of microcontroller chips produced by Espressif Systems in Shanghai. It is available in a number of low-cost modules.
Its chip comes with 48 pins with multiple functions. Not all pins are exposed in all ESP32 development boards, and some pins cannot be used.

Note: The boards share many features but they don't have the same pinouts. In our experiment, It will ll be referring to the pin function (i.e. GPIO), as shown in Figure 2 instead of an actual pin number.

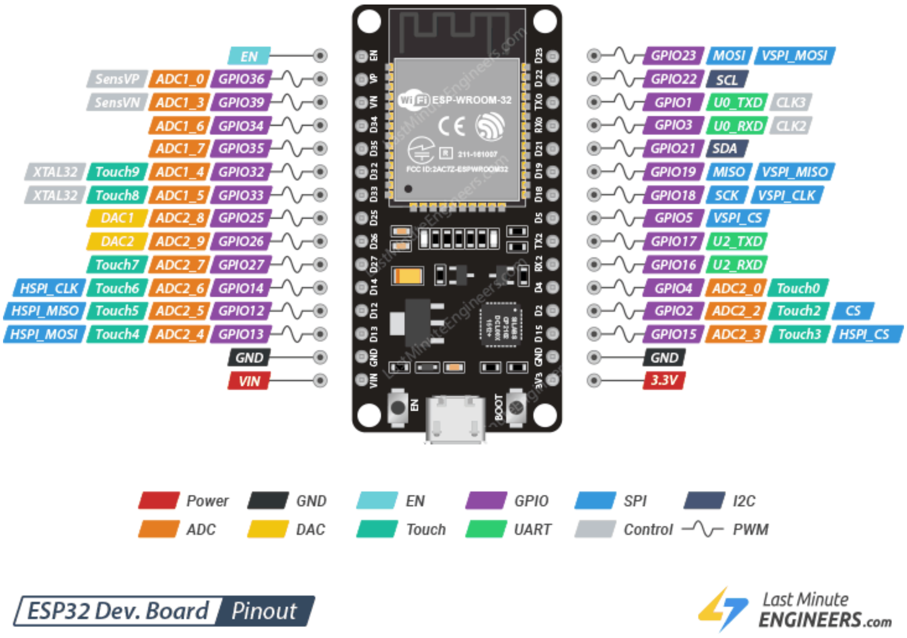


Figure 3.1: ESP32 WROOM Pinout.

Servo Motor

The servo motor is operated at +5V. It can rotate from 0° to 180° due to their gear arrangement, but in this experiment it only rotates 90° and -90° to open or lock the door.

Configuration

- Brown -> Ground wire connected to the ground of system.
- Red -> Powers the motor +5V is used.
- Orange -> PWM signal is given in through this wire to drive the motor.

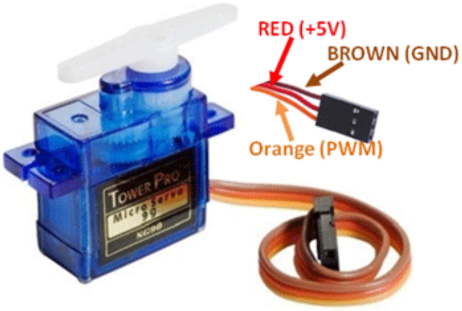


Figure 3.2: Servo Motor Pinout.

Keypad

A 4X4 KEYPAD is used as an input device due to entering the the door password , it have 8 terminals. In them four are ROWS of matrix and four are COLUMNS of matrix. These 8 PINS are driven out from 16 buttons present in the MODULE. Those 16 alphanumeric digits on the MODULE surface are the 16 buttons arranged in matrix formation.

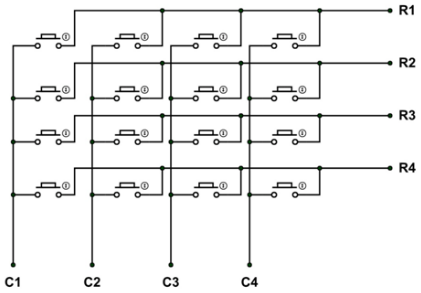


Figure 3.3: The internal structure of 4X4 KEYPAD MODULE.

All of these components are connected to the ESP with the correct pins as shown in Figures 3.4 and 3.5

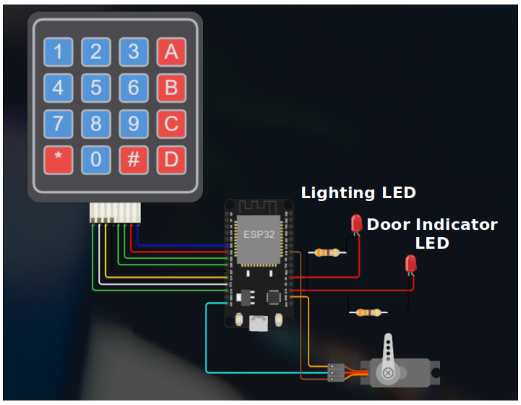
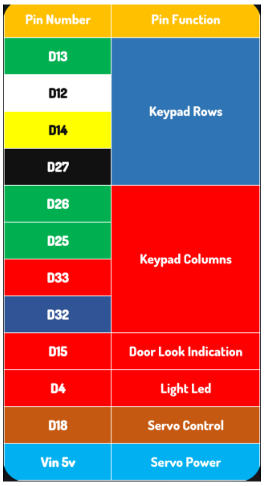


Figure 3.4: Hardware Schematic



Pin Number	Pin Function
D13	Keypad Rows
D12	
D14	
D27	
D26	Keypad Columns
D25	
D33	
D32	
D15	Door Look Indication
D4	Light Led
D18	Servo Control
Vin 5v	Servo Power

Figure 3.5: Connected Pins.

Software

Arduino IDE

Arduino setup and WiFi Communication

There are several dependecies to program the circuit

- ESP32 Board jason file, it is downloaded from ESP32 package
- ServoESP32 library
- Keypad library

After installing the required board and libraries, It is the code time,The ESP32 can be programmed using many different development environments. Code can be written in C++ (like the Arduino) or in MicroPython, here the arduino is used to program it.

As a communication system the mobile app is considered the client and the ESP is the server. In order for a client to be able to reach the HTTP server, the router is not needed but rather to the WiFi network hosted by the ESP32, it works as the access point .

The possibility of setting a HTTP server on the ESP32 working as soft AP is very useful since in real application scenarios, an IoT device may be deployed in a WiFi network to which the credentials are not known in code compile time.

The ESP32 operate as soft AP when connected for the first time, starting a HTTP server that serves a configuration HTML webpage for the user to input the name and password of the WiFi network to which the device should connect to to be able to reach the Internet and operate.

<WiFiServer server(80);> This class creates a server that listens for incoming connections on the HTTP port 80.

<WiFi.softAP(ssid, password);> To start the soft AP, we simply need to call the softAP method of the WiFi external variable (this is the same variable we use to connect the ESP32 to a WiFi network). This method receives as first input the name of the WiFi network we want to set and as second input its password. Just as a note, setting a password is not mandatory and we could have not specify it if we wanted our Access Point to be open.

<Serial.begin(115200);> Moving on to the setup function, start it by opening a serial connection needed to print the IP of the ESP32 for the client to be able to reach it.

Finally, we will need to know the ESP32 IP, in order for the client connected to its network to be able to send requests to it. We can obtain the IP by calling the softAPIP method on the same WiFi variable.

In the void setup function, the function will try to connect to WiFi. This process executes in a loop, which means it runs until there is a connection to WiFi.

In void loop, itCheck if a client has connected. It Wait until the client sends some data and performs tasks according to input. Then the connection disabled after it is performed.

The tasks is the requests arrived from the client (mobile app), for example, if the client open the door with the correct pasword, the HTTP request will end with "Led/1", it causes the red led to turns on and the motor rotate 90 degrees. Another method to control the door is the physical keypad that connected directly with the ESP.

MIT App Inventor

MIT App Inventor is an open-source website for Android. It was originally created by Google but is now maintained by the Massachusetts Institute of Technology (MIT). MIT App Inventor can easily create applications for Android. The MIT App Inventor uses a GUI (graphical interface), in which users can drag and drop visual objects to create apps that can be easily runon Android devices.
The phone app consist of three sections :

1. Unlock the door

here is where the user enters the password to unlock the door
the keypad has the alphanumeric inputs (0 to 9) in addition to A,B,C ,also it has clear (Cl) to delete all text written in the textbook and set it to empty string, enter (En) to confirm the password and sends a request to open the door in case the password is right, and (D) is used to send a request to close the door.
the unlock request : http/192.168.4.1/door/1
the close request : http/192.168.4.1/door/0
also there is a "see password" button to show and hide password when touch it

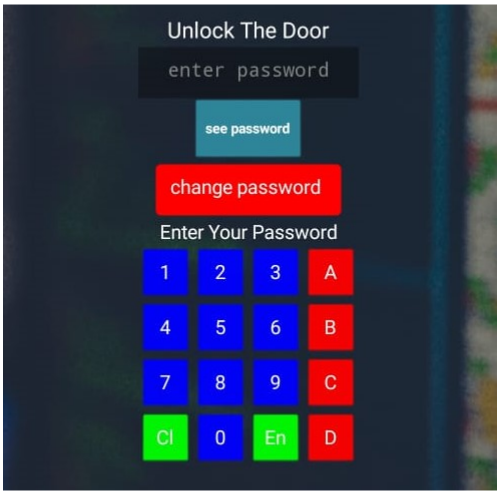
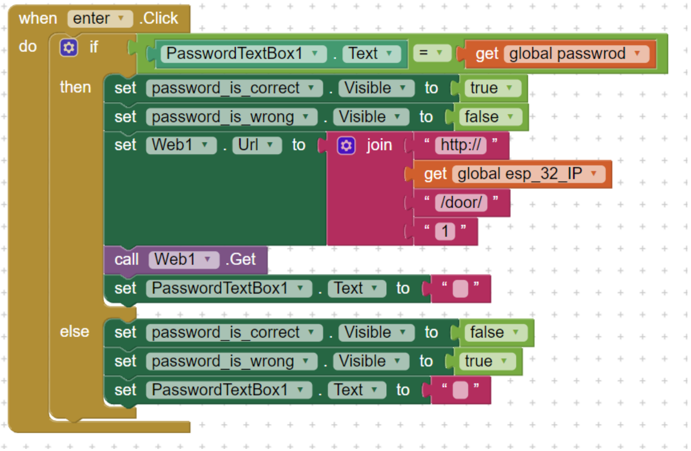


Figure 4.1: Unlock the door app section

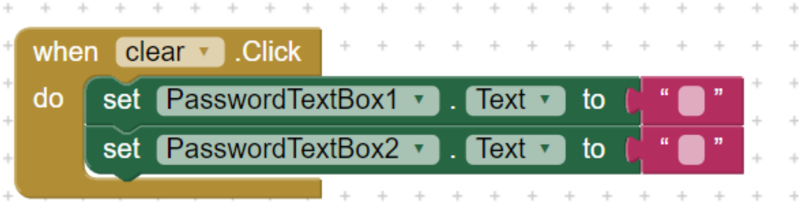
2. Change the password

When the change password button is pushed, the user is moved to another text box where he can input the new password and it is saved to "password" global variable. when



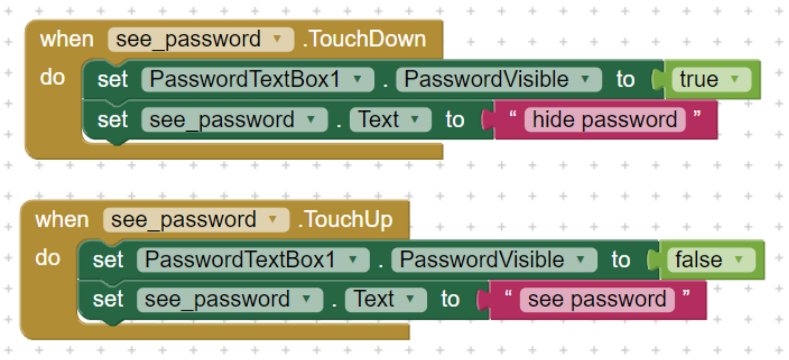
```
when enter .Click
do  if    PasswordTextBox1 . Text = get global passwrod
    then  set password_is_correct . Visible to true
          set password_is_wrong . Visible to false
          set Web1 . Url to join " http:// "
                                 get global esp_32_IP
                                 " /door/ "
                                 " 1 "
          call Web1 .Get
          set PasswordTextBox1 . Text to " "
    else  set password_is_correct . Visible to false
          set password_is_wrong . Visible to true
          set PasswordTextBox1 . Text to " "
```

Figure 4.2: The "enter" button function and the WiFi request



```
when clear .Click
do  set PasswordTextBox1 . Text to " "
    set PasswordTextBox2 . Text to " "
```

Figure 4.3: The "clear" button function



```
when see_password .TouchDown
do  set PasswordTextBox1 . PasswordVisible to true
    set see_password . Text to " hide password "

when see_password .TouchUp
do  set PasswordTextBox1 . PasswordVisible to false
    set see_password . Text to " see password "
```

Figure 4.4: The "see password" button function when touched up and down

"authenticate" is pushed, the user is returned back to where he can enter his password again to unlock the door.

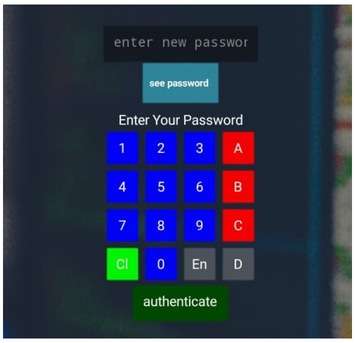


Figure 4.5: Change the password app section

3. Lighting system

The lighting system control is done by two buttons "light on" "light of" both buttons sends a requests to the WiFi server to turn on/off lights
turn on light request : http/192.168.4.1/Led/1
turn off light request : http/192.168.4.1/Led/0

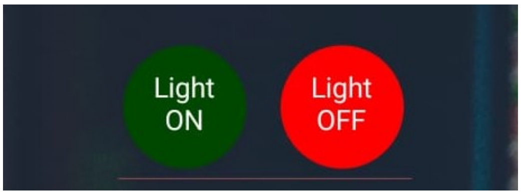
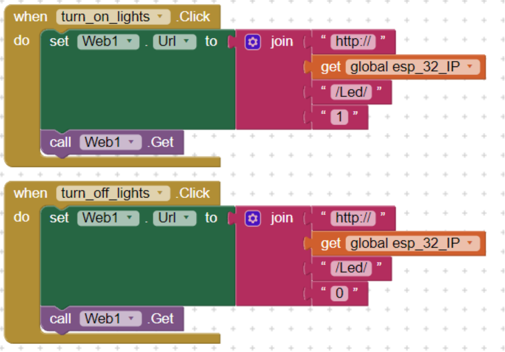


Figure 4.6: Control lighting system app section



```
when turn_on_lights .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Led/ "
                             " 1 "
    call Web1 .Get

when turn_off_lights .Click
do  set Web1 . Url to  join  " http:// "
                             get global esp_32_IP
                             " /Led/ "
                             " 0 "
    call Web1 .Get
```

Figure 4.7: The lighting system control buttons function.

Risks and Challenges

DC motor with H-bridge circuit

A DC motor controlled by H-Bridge circuit could be used instead of the servo motor. The usage of H-Bridge is to control the direction of the motor and the speed is controlled by Pulse-width modulation (PWM) signal. The motor speed is proportional to the duty cycle of the PWM.

- The setup of the motor on the arduino IDE is not stable enough to be reliable, the output degree of rotation was not the desired on.

Change password in the application

- Changing the password is working but depends on that the app is open all the time ,once you close the app the saved password will be removed and each time you open the app the password will be set to the value of the global variable password.

To solve this problem the app should have a database base where it save the latest password and take it back each time the app opend.

- To change the password we couldn't use the same textbox to unlock the door as this would cause conflict between the data while changing the password used to unlock the door.

So we doublicated the textbox to make another one separated to carry the new password so that you can see that the most of the functions happen when clicking change password or authenticate is visible enable and disable.

Servo rotation

It was a bug in the arduino code, at the first time the user enter the correct password, the motor rotates correctly in the desired direction, when he entered the correct password again the motor gets back to its first position and rotates 90 degrees again.

To solve this problem a flag of global variable is used (doorOpne) that indicates the door status, due to not repeat the the rotation after every correct password entry.

Arduino Libraries setup

Different Servo library (ESP32 Servo) that not support ESP32 WROOM-DA board at the first time, it is name is very similar to the used one (ServoESP32) that operates correctly without any errors.

Conclusions and Future Work

Conclusion

These days, You can improve your home or office security by adding technology. Using smartphones and devices become very popular for people and it is a good option for us to make our stuffs smarter.

This project shows a low-cost secure technology that can be used to save our lives with very simple components and in easiest way. it is also a user friendly because it came with an Android GUI application that control the door lock and the building's lightening system.

Future Work

- The usage of camera module came with ESP32 , so the user can find out who is visiting.
- Smart door face detection, that can recognise the owner and open the door without entering any passwords.
- New emergency call feature to directly call the police.
- Two-way communication system between the owner of the door and the guess.
- Secure and encrypt the messages between the application and the ESP, to build a safe communication from hackers.
- View the door access history logs and build our data.
- Database system in order to change the password easily and save the logs.
- Application lock with human fingerprint.
- Alarm when any attacks happened.

Appendix

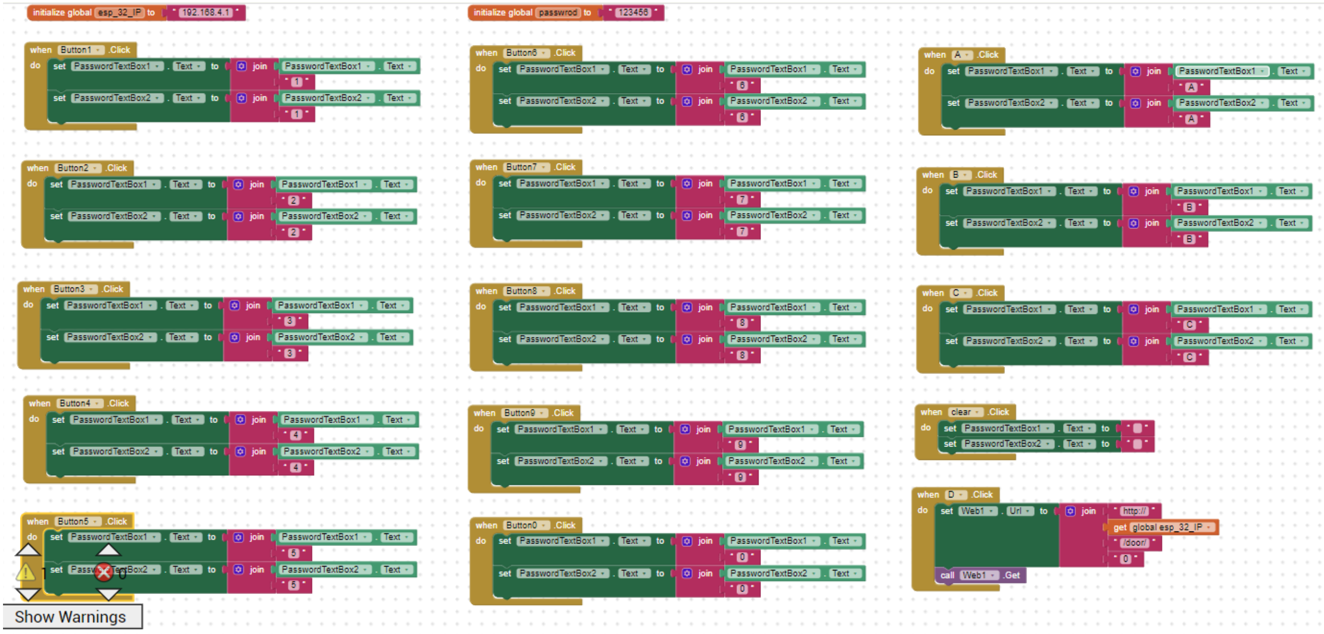
Arduino Code

GitHub repo that contains the source code of arduino and the mobile application files (.aia) and (.apk):Link

```
// Door Lock and Lightening System
//
//
//

//////////////// WiFi Librarires ///////////
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

////////////// Keypad Library //////////////
#include <Keypad.h>

////////////// Servo Library //////////////
#include <Servo.h>



#define LED_BUILTIN 4  // Light Sys. Led //Green//
#define LED   15 // Door Indecation LED  // Red//

/////////////// Keypad definitions /////////
#define ROW_NUM     4  // four rows
#define COLUMN_NUM  4  // four columns

int pos = 0;    // variable to store the servo position

int DoorOpen = 0; //idicatour -- not repeat the request

///////////////// WiFi SetUp /////////////
// Set these to your desired credentials.
const char *ssid = "Door Lock";
const char *password = "12345678";
WiFiServer server(80);


/////////////// Keypad SetUp ///////////////
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte pin_rows[ROW_NUM] = {13, 12, 14, 27};    // GIOP19, GIOP18, GIOP5, GIOP17 connect to the row pins
byte pin_column[COLUMN_NUM] = {26, 25, 33, 32}; // GIOP16, GIOP4, GIOP0, GIOP2 connect to the column pins

Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM, COLUMN_NUM );

const String password_1 = "123"; // change your password here
const String password_2 = "1234";  // change your password here
const String password_3 = "12345";  // change your password here
String input_password;


Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards



void setup() {

  pinMode(LED_BUILTIN, OUTPUT);
   pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW ); // lock the door

  Serial.begin(115200);
  Serial.println("Hello, Mo3ed!");

  Serial.println();
  Serial.println("Configuring access point...");



  // You can remove the password parameter if you want the AP to be open.
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.begin();
  Serial.println("Server started");

   myservo.attach(18);                       // attaches the servo on pin 18 to the servo object
   input_password.reserve(32);               // maximum input characters is 32
}

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

  if (client)
  {                                           // if you get a client,
    Serial.println("New Client.");            // print a message out the serial port
    String currentLine = "";                  // make a String to hold incoming data from the client
    while (client.connected())
    { // loop while the client's connected
      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
        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)
          {
                                            // 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();

                                            // the content of the HTTP response follows the header:
            client.print("Click <a href=\"/H\">here</a> to turn ON the LED.<br>");
            client.print("Click <a href=\"/L\">here</a> to turn OFF the LED.<br>");

                                            // 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
          }



        // Check to see if the client request was "GET /H" or "GET /L":
        if (currentLine.endsWith("Led/1"))
        {
          digitalWrite(LED_BUILTIN, HIGH);              // GET Led/1 turns the LED on
        }
        if (currentLine.endsWith("Led/0"))
        {
          digitalWrite(LED_BUILTIN, LOW);               // GET Led/1 turns the LED off
        }

        ////////////// Door Open ////////////////////
        if (currentLine.endsWith("door/1"))
        {
          digitalWrite(LED, HIGH);
          if(DoorOpen == 0)
          {
            for (pos = 0; pos <= 90; pos += 1)
            {
               myservo.write(pos);               // tell servo to go to position in variable 'pos'
               delay(10);
            }
            DoorOpen = 1;
          }
        }


        ////////////// Door Close ////////////////////
        if (currentLine.endsWith("door/0"))
        {
          digitalWrite(LED, LOW );
          if(DoorOpen == 1)
          {
            for (pos = 90; pos >= 0; pos -= 1)
            {
            myservo.write(pos);             // tell servo to go to position in variable 'pos'
            delay(10);
            }
            DoorOpen = 0;

          }
        }


      }
    }

    client.stop();                                 // close the connection:
    Serial.println("Client Disconnected.");
  }




  /////////////////// keypad //////////////////////

  char key = keypad.getKey();

  if (key)
  {
    Serial.println(key);
     /////////////// Door Close ////////////////////
    if (key == 'D')
    {
      if(DoorOpen == 1)
      {
      digitalWrite(LED, LOW );
       for (pos = 90; pos >= 0; pos -= 1)
       {                                    // goes from 90 degrees to 0 degrees
           myservo.write(pos);              // tell servo to go to position in variable 'pos'
```

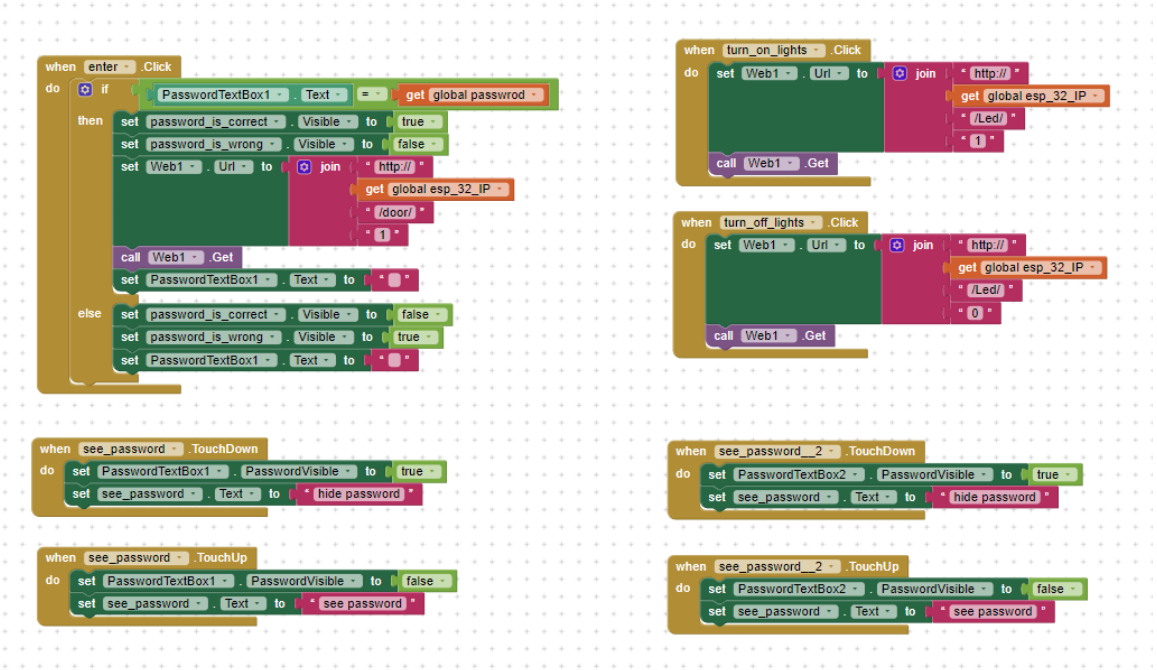
```
           delay(10);                     // waits 15ms for the servo to reach the position
       }
      Serial.println("DOOR LOOKED");
      DoorOpen = 0;
      }
    }

     //////////////// Reset the password ////////////////////////////

    else if (key == '*')
    {
      input_password = "";                       // reset the input password
      Serial.println("reset the input password");
    }


     /////////////////// Check the password ///////////////////////
    else if (key == '#')
    {
      if (input_password == password_1 || input_password == password_2 || input_password == password_3)
      {
        digitalWrite(LED, HIGH);
        Serial.println("Valid Password => unlock the door");
        if(DoorOpen == 0)
        {
           for (pos = 0; pos <= 90; pos += 1)
            {                               // goes from 0 degrees to 90 degrees
             myservo.write(pos);            // tell servo to go to position in variable 'pos'
             delay(10);
            }

             DoorOpen = 1;
        }

      }
      else
      {
        Serial.println("Invalid Password => Try again");
      }

      input_password = ""; // reset the input password
    }
    else
    {
      input_password += key; // append new character to input password string
    }
  }


}

// Code
```

Mobile Application



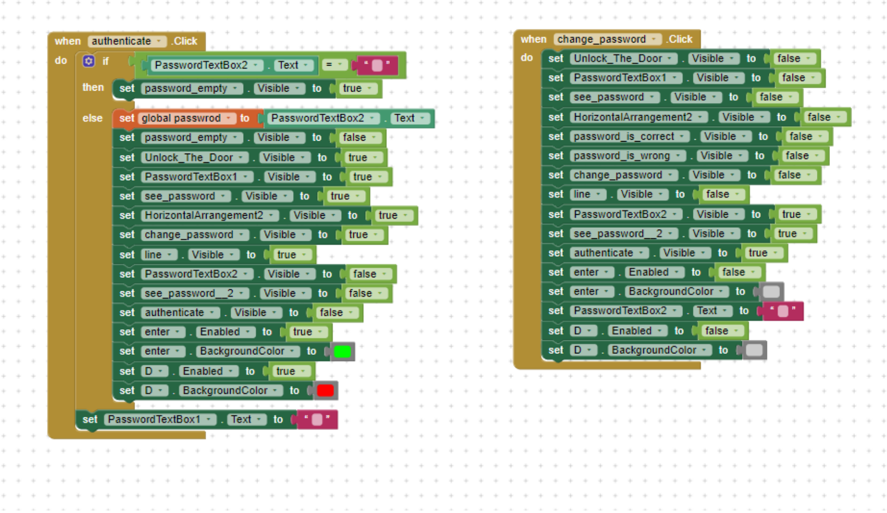
```
initialize global esp_32_IP to "192.168.4.1"

when Button1 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "1"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "1"

when Button2 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "2"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "2"

when Button3 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "3"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "3"

when Button4 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "4"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "4"

when Button5 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "5"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "5"

initialize global passwrod to "123456"

when Button6 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "6"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "6"

when Button7 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "7"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "7"

when Button8 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "8"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "8"

when Button9 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "9"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "9"

when Button0 .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "0"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "0"

when A .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "A"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "A"

when B .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "B"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "B"

when C .Click
do  set PasswordTextBox1 . Text to join PasswordTextBox1 . Text
                                        "C"
    set PasswordTextBox2 . Text to join PasswordTextBox2 . Text
                                        "C"

when clear .Click
do  set PasswordTextBox1 . Text to " "
    set PasswordTextBox2 . Text to " "

when D .Click
do  set Web1 . Url to join "http://"
                           get global esp_32_IP
                           "/door/"
                           "0"
    call Web1 .Get
```

Figure 7.1: First block diagram of the application.



```
when enter .Click
do  if  PasswordTextBox1 . Text = get global passwrod
    then set password_is_correct . Visible to true
         set password_is_wrong . Visible to false
         set Web1 . Url to join "http://"
                                get global esp_32_IP
                                "/door/"
                                "1"
         call Web1 .Get
         set PasswordTextBox1 . Text to " "
    else set password_is_correct . Visible to false
         set password_is_wrong . Visible to true
         set PasswordTextBox1 . Text to " "

when see_password .TouchDown
do  set PasswordTextBox1 . PasswordVisible to true
    set see_password . Text to "hide password"

when see_password .TouchUp
do  set PasswordTextBox1 . PasswordVisible to false
    set see_password . Text to "see password"

when turn_on_lights .Click
do  set Web1 . Url to join "http://"
                           get global esp_32_IP
                           "/Led/"
                           "1"
    call Web1 .Get

when turn_off_lights .Click
do  set Web1 . Url to join "http://"
                           get global esp_32_IP
                           "/Led/"
                           "0"
    call Web1 .Get

when see_password__2 .TouchDown
do  set PasswordTextBox2 . PasswordVisible to true
    set see_password . Text to "hide password"

when see_password__2 .TouchUp
do  set PasswordTextBox2 . PasswordVisible to false
    set see_password . Text to "see password"
```

Figure 7.2: Second block diagram of the application.



```
when authenticate .Click
do  if  PasswordTextBox2 . Text = " "
    then  set password_empty . Visible to true
    else  set global passwrod to PasswordTextBox2 . Text
          set password_empty . Visible to false
          set Unlock_The_Door . Visible to true
          set PasswordTextBox1 . Visible to true
          set see_password . Visible to true
          set HorizontalArrangement2 . Visible to true
          set change_password . Visible to true
          set line . Visible to true
          set PasswordTextBox2 . Visible to false
          set see_password__2 . Visible to false
          set authenticate . Visible to false
          set enter . Enabled to true
          set enter . BackgroundColor to 🟩
          set D . Enabled to true
          set D . BackgroundColor to 🟥
    set PasswordTextBox1 . Text to " "
```

```
when change_password .Click
do  set Unlock_The_Door . Visible to false
    set PasswordTextBox1 . Visible to false
    set see_password . Visible to false
    set HorizontalArrangement2 . Visible to false
    set password_is_correct . Visible to false
    set password_is_wrong . Visible to false
    set change_password . Visible to false
    set line . Visible to false
    set PasswordTextBox2 . Visible to true
    set see_password__2 . Visible to true
    set authenticate . Visible to true
    set enter . Enabled to false
    set enter . BackgroundColor to ⬜
    set PasswordTextBox2 . Text to " "
    set D . Enabled to false
    set D . BackgroundColor to ⬜
```

Figure 7.3: First block diagram of the application.

User Guide

Smart door lock and lighting system

Preface

Our system consists of Esp32, a keypad, and an Application, which interact with each other and with the user to achieve Its mission, which is the ability to control the lock of the door and light system. The user should be able to control the system lock through two different methods:

- Direct Control (Keypad).
- Wireless Control (Mobile Application).

And to control the lighting system through:

- Wireless Control (Mobile Application).

System lock control

Keypad

- The numbers are used to input the password.
- The ‘’ to submit the input password.
- The ‘*’ to clear the input password.

Application

- The numbers are used to input the password.
- The “En” to submit the input password.
- The “Cl” to clear the input password.

Facilities

For a more comfortable using experience, we have added a change password facility.
Using this facility by clicking on the “Change password” button, then entering your new password.
For adding more reliability we have added three master keys you can use by using the keypad.

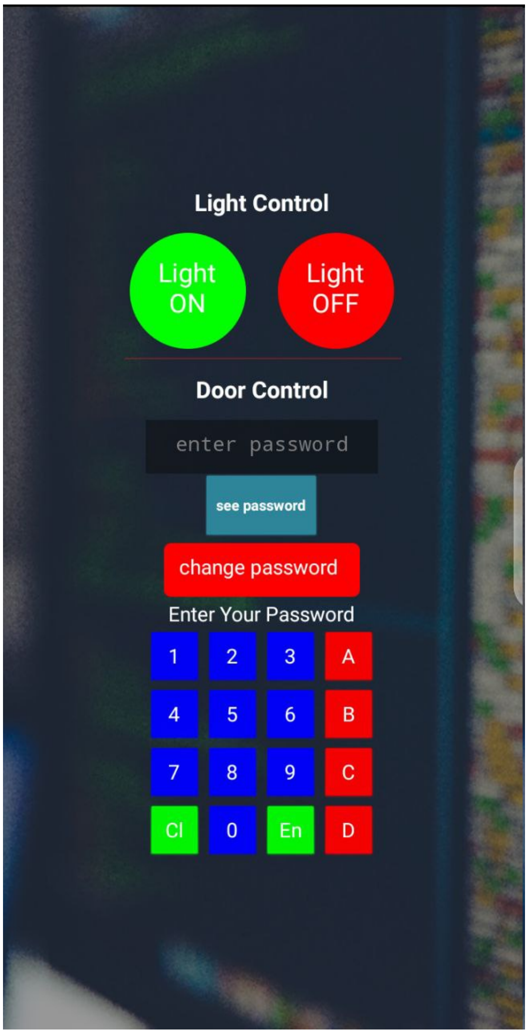


Figure 8.1: Application user Interface.

System lighting control

- By clicking on the “light on” button: house lights turn on.
- By clicking on the “light off” button: house lights turn off.

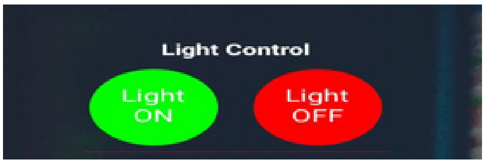


Figure 8.2: Application user Interface (Lights).