INTRODUCTION :

What is IOT?

Internet of Things (IoT) is a technology where everyday physical devices are connected to the Internet, allowing them to collect, send, and receive data. These devices can interact with each other and with users remotely, making life easier, smarter, and more efficient.

IoT has revolutionized many fields like healthcare, transportation, and especially **home automation**.

How is it related to Home Automation:

Home Automation refers to the use of IoT and smart devices to monitor and control various aspects of a house — like lights, doors, fans, security systems, and environmental sensors — automatically or remotely through smartphones or computers. The goal is to create a more **comfortable, secure, energy-efficient, and convenient** living environment.

The objective of this project is to design and implement a **Smart Home Automation System** using IoT technologies.
The system allows users to:

- **Securely enter** their homes using a password-protected door lock.
- **Monitor** environmental conditions like **temperature**, **humidity**, **gas leaks**, and **fire detection**.
- **Control** home appliances such as **lights** and **fans** from anywhere through a mobile application built on **MIT App Inventor**.
- **Receive alerts** and **status updates** in real-time, ensuring safety and convenience.

A small-scale model house was built, integrating various sensors and actuators, controlled through an **Arduino** microcontroller and **WiFi communication** (ESP8266), showcasing a complete working prototype of smart home automation.

COMPONENTS USED ALONG WITH THEIR SPECIFICATIONS:

Sl. NO	NAME OF THE COMPONENT	SPECIFICATION	QUANTITY
1.	Arduino Uno	Microcontroller, 16MHz clock speed, 14 digital I/O	1
2.	Servo Motor	180° rotation, 5V operation	1
3.	LCD Display (16x2)	16 characters x 2 lines, I2C module	1
4.	Keypad	4x4 Matrix, 16 buttons	1
5.	LDR Sensor	Light-dependent resistor, analog input	1
6.	Gas Sensor (MQ-2)	Gas detection (LPG, smoke, alcohol), analog output	1
7.	Buzzer	5V Piezo buzzer for audio alerts	1
8.	Flame Sensor	Detects flame/fire presence, digital output	1
9.	WiFi Module (ESP8266)	WiFi communication module, TCP/IP protocol support	1
10.	DHT22 Sensor	Temperature and Humidity sensor, digital output	1
11.	DC Motor (Fan)	6V-12V small motor for ventilation simulation	1
12.	Power Supply/Battery	5V/9V regulated supply for components	1
13.	Jumper Wires, Breadboard	Connection and prototyping	
14.	4-Channel Relay Module	5VDC,12DC & 100mA Relay Type: Single Pole Double Throw (SPDT)	1

MOTIVATION/SOCIATAL/INDUSTRIAL IMPACT:

Motivation:

Traditional home automation systems are often expensive and inaccessible for common households. This project aims to offer a **low-cost**, **easy-to-deploy** alternative using commonly available hardware and open-source software.
This project even tries to let the user be interactive with condition status of the house.

Social/Industrial Impact:

- Smart Homes:

  In the coming years, fully automated smart homes will surely become a reality because the home automation is growing rapidly in this tech world. Due to good user convenience and accessibility, smart homes are appealing for a wide range of people all over the world. The User can check for the electricity usage, the condition of their devices and receive notifications accordingly.
- Smart Cities:

  With increasing automation and IOT, devices can share info with each other. This will help in building modern and hi-tech cities. Cities that might be free form all types of pollutions, traffic accidents, etc. problems.
- Agriculture:

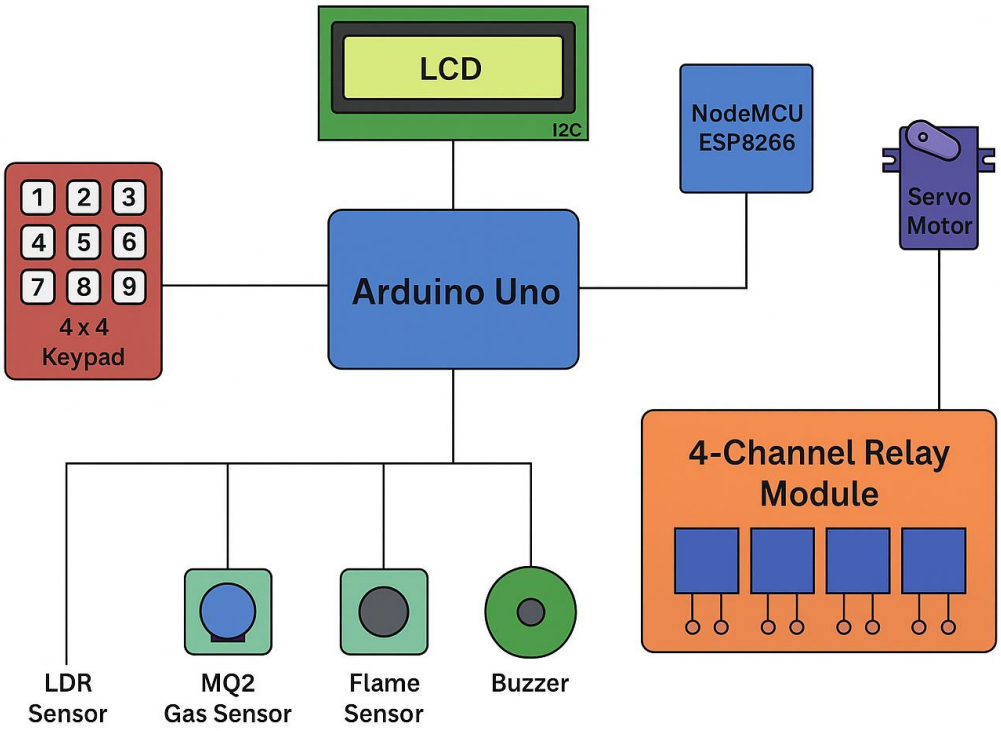
  The proposed system might be used in Agricultural as well. The various devices utilized in fields are often operated from any remote location.

Objective:

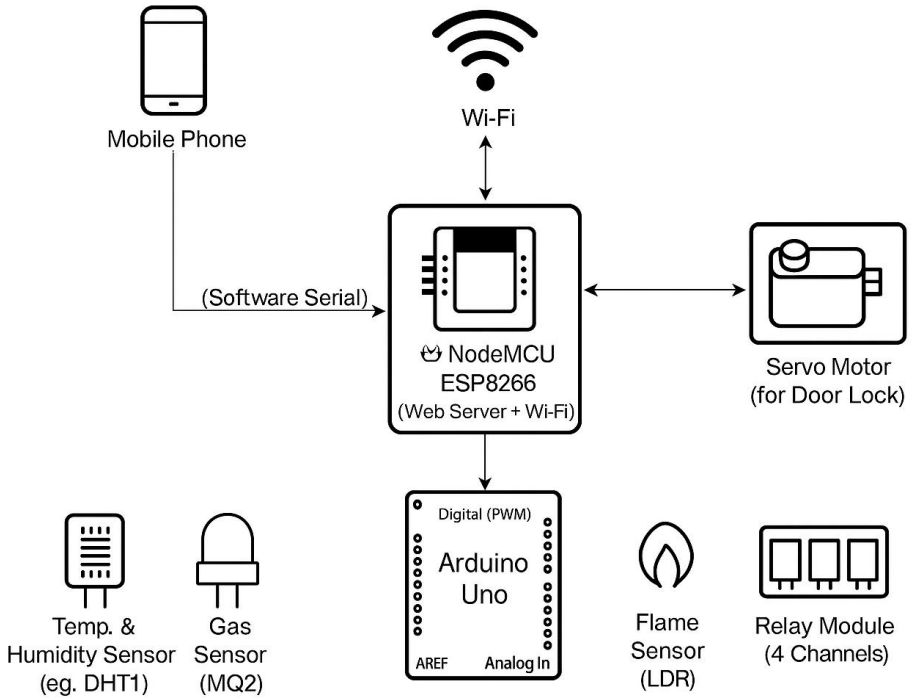
As technology advances, houses are becoming smarter, shifting from conventional switches to centralized, remote-controlled systems. Traditional wall switches make it difficult for users, especially the elderly or physically challenged, to operate appliances. Remote-controlled home automation using smartphones offers a modern, convenient solution.

BLOCK DIAGRAM/SYSTEM DESIGN:

Arduino block diagram



ESP8266 block diagram:



CODE (ARDUNIO AND MIT APP CODE):

Code for Arduino:

```
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Keypad.h>

#define DHTPIN 2
#define DHTTYPE DHT22
#define MQ2PIN A0
#define FLAMEPIN 3
#define LDRPIN A1
#define RELAY1 4
#define RELAY2 5
#define RELAY3 6
#define RELAY4 7
#define BUZZER 8
#define SERVO_PIN 9

DHT dht(DHTPIN, DHTTYPE);
int flameValue;
int ldrValue;
int mq2Value;
String dataString = "";
char incomingByte;

#define LCD_ADDR 0x27
#define LCD_COLS 16
#define LCD_ROWS 2

LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);


const byte ROWS = 3;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'}
};
byte rowPins[ROWS] = {10, 11, 12};
byte colPins[COLS] = {13, A2, A3};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);


Servo doorServo;
String password = "1234";
String enteredPassword = "";
bool doorOpen = false;
bool gasDetected = false;
bool buzzerActive = false;
unsigned long buzzerStartTime = 0;
const unsigned long buzzerDuration = 5000;
bool showSensorData = false;
bool alertSent = false;
```

```
void setup() {
  Serial.begin(9600);

  pinMode(RELAY1, OUTPUT);
  pinMode(RELAY2, OUTPUT);
  pinMode(RELAY3, OUTPUT);
  pinMode(RELAY4, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(FLAMEPIN, INPUT);
  pinMode(LDRPIN, INPUT);
  pinMode(MQ2PIN, INPUT);

  digitalWrite(RELAY1, HIGH);
  digitalWrite(RELAY2, HIGH);
  digitalWrite(RELAY3, HIGH);
  digitalWrite(RELAY4, HIGH);
  digitalWrite(BUZZER, LOW);

  dht.begin();
  lcd.init();
  lcd.backlight();
  lcd.print("Smart Home");
  lcd.setCursor(0, 1);
  lcd.print("System Online");
  delay(2000);
  lcd.clear();
  lcd.print("Enter Password:");

  doorServo.attach(SERVO_PIN);
  doorServo.write(90);
}

void loop() {
  if (Serial.available() > 0) {
    while (Serial.available() > 0) {
      incomingByte = Serial.read();
      if (incomingByte == '\n') {
        break;
      }
      dataString += incomingByte;
    }
    dataString.trim();
    processCommand(dataString);
    dataString = "";
  }

  readSensors();
  checkAndSendAlerts();
  controlBuzzerAndExhaust();
  sendSensorData();
  delay(1000);

  char key = keypad.getKey();
  if (key) {
    handleKeypadInput(key);
  }

  if (showSensorData) {
```

```
    displaySensorData();
    displayDoorStatus();
  } else {
    displayDoorStatus();
  }

  if (buzzerActive && (millis() - buzzerStartTime < buzzerDuration)) {
    digitalWrite(BUZZER, HIGH);
  } else {
    digitalWrite(BUZZER, LOW);
    buzzerActive = false;
  }
  alertSent = false;
}

void processCommand(String command) {
  if (command == "light1_on") {
    digitalWrite(RELAY1, LOW);
    Serial.println("Light 1 ON");
  } else if (command == "light1_off") {
    digitalWrite(RELAY1, HIGH);
    Serial.println("Light 1 OFF");
  } else if (command == "fan_on") {
    digitalWrite(RELAY2, LOW);
    Serial.println("Fan ON");
  } else if (command == "fan_off") {
    digitalWrite(RELAY2, HIGH);
    Serial.println("Fan OFF");
  } else if (command == "exhaust_on") {
    digitalWrite(RELAY3, LOW);
    Serial.println("Exhaust Fan ON");
  } else if (command == "exhaust_off") {
    digitalWrite(RELAY3, HIGH);
    Serial.println("Exhaust Fan OFF");
  } else if (command == "light_on") {
    digitalWrite(RELAY4, LOW);
    Serial.println("Light ON");
  } else if (command == "light_off") {
    digitalWrite(RELAY4, HIGH);
    Serial.println("Light OFF");
  } else if (command == "buzzer_on") {
    digitalWrite(BUZZER, HIGH);
    Serial.println("Buzzer ON");
    buzzerActive = true;
    buzzerStartTime = millis();
  } else if (command == "buzzer_off") {
    digitalWrite(BUZZER, LOW);
    Serial.println("Buzzer OFF");
    buzzerActive = false;
  } else if (command == "unlock_door") {
    doorServo.write(180);
    doorOpen = true;
    Serial.println("Door Unlocked");
    lcd.clear();
    lcd.print("Door Unlocked");
    showSensorData = true;
    if (ldrValue < 500) {
```

```
      digitalWrite(RELAY1, LOW);
      Serial.println("Light 1 ON (Dark)");
    }
  } else if (command == "lock_door") {
    doorServo.write(90);
    doorOpen = false;
    digitalWrite(RELAY1, HIGH);
    Serial.println("Door Locked");
    lcd.clear();
    lcd.print("Door Locked");
    showSensorData = false;
  } else if (command == "reset") {
    resetSystem();
  }
}

void readSensors() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  mq2Value = analogRead(MQ2PIN);
  flameValue = digitalRead(FLAMEPIN);
  ldrValue = analogRead(LDRPIN);

  if (isnan(t) || isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
}

void sendSensorData() {
  float ldrPercentage = (100.0 - (ldrValue * 100.0 / 1023.0));
  String data = String(millis()) + "," + String(dht.readTemperature()) + "," + String(dht.readHumidity()) + "," + String(mq2Value) + "," + String(flameValue) + "," + String(ldrPercentage) + "\n";
  Serial.print(data);
}

void checkAndSendAlerts() {
  if (flameValue == LOW && !alertSent) {
    Serial.println("ALERT: Flame Detected!\n");
    alertSent = true;
  }
  if (mq2Value > 250 && !alertSent) {
    Serial.println("ALERT: High Gas Concentration Detected!\n");
    alertSent = true;
  }
}

void controlBuzzerAndExhaust() {
  if (flameValue == LOW || mq2Value > 300) {
    if (!buzzerActive) {
      digitalWrite(BUZZER, HIGH);
      buzzerActive = true;
```

```
    buzzerStartTime = millis();
    Serial.println("Buzzer ON (Flame or High Gas)");
   }
  }

  if (mq2Value > 100) {
   digitalWrite(RELAY3, LOW);
   gasDetected = true;
   Serial.println("High Gas Detected! Exhaust Fan ON");
  } else if (gasDetected) {
   digitalWrite(RELAY3, HIGH);
   gasDetected = false;
   Serial.println("Gas level normal. Exhaust Fan OFF");
  }
}

void handleKeypadInput(char key) {
  if (key >= '0' && key <= '9') {
   enteredPassword += key;
   lcd.setCursor(0, 1);
   lcd.print("                ");
   lcd.setCursor(0, 1);
   lcd.print(enteredPassword);
   if (enteredPassword.length() >= password.length()) {
    if (enteredPassword == password) {
     Serial.println("Correct Password");
     lcd.clear();
     lcd.print("Door Unlocked");
     doorServo.write(180);
     doorOpen = true;
     showSensorData = true;
     if (ldrValue < 500) {
      digitalWrite(RELAY1, LOW);
      Serial.println("Light 1 ON (Dark)");
     }
     enteredPassword = "";
    } else {
     Serial.println("Incorrect Password");
     lcd.clear();
     lcd.print("Incorrect!");
     delay(2000);
     enteredPassword = "";
     lcd.clear();
     lcd.print("Enter Password:");
    }
   }
  } else if (key == '*') {
   enteredPassword = "";
   lcd.clear();
   lcd.print("Enter Password:");
  }
}

void resetSystem() {
  gasDetected = false;
```

```
  buzzerActive = false;
  digitalWrite(RELAY3, HIGH);
  digitalWrite(BUZZER, LOW);
  enteredPassword = "";
  doorOpen = false;
  doorServo.write(90);
  lcd.clear();
  lcd.print("System Reset");
  delay(2000);
  lcd.clear();
  lcd.print("Smart Home");
  lcd.setCursor(0, 1);
  lcd.print("System Online");
  Serial.println("System Reset");
}

void displaySensorData() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:" +
String(dht.readTemperature()) + "C H:" +
String(dht.readHumidity()) + "%");
  lcd.setCursor(0, 1);
  lcd.print("G:" + String(mq2Value) + " F:"
+ String(flameValue));
  lcd.setCursor(8, 0);
  lcd.print("L:" + String((100 - (ldrValue *
100.0 / 1023.0))) + "%");
}

void displayDoorStatus() {
  lcd.setCursor(8, 1);
  if (doorOpen) {
    lcd.print("Door:Open");
  } else {
    lcd.print("Door:Locked");
  }
}
```

Code for Wi-Fi Module(ESP8622):

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <SoftwareSerial.h>

const char* ssid = "Unifi";
const char* password = "99990000";

const int port = 80;
ESP8266WebServer server(port);
SoftwareSerial arduinoSerial(D2, D3);

String timeStamp = "N/A";
String temperature = "N/A";
String humidity = "N/A";
String gasValue = "N/A";
String flameValue = "N/A";
String lightValue = "N/A";
String latestAlert = "";
String doorStatus = "Locked";

const char* automation_html = R"=====(
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="refresh" content="5">
  <title>Automation</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    .btn {
      transition: all 0.3s ease;
    }
    .btn:hover {
      transform: translateY(-2px);
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }
  </style>
</head>
<body class="bg-gray-100 flex items-center justify-center min-h-screen">
  <div class="bg-white w-full max-w-md rounded-lg shadow-lg overflow-hidden">
    <div class="bg-indigo-600 text-white text-center py-4">
      <h1 class="text-2xl font-bold">AUTOMATION</h1>
    </div>
    <div class="p-6 space-y-6">
      <div class="flex justify-center space-x-4">
        <a href="/command?command=light1_on" class="btn bg-green-500 text-white px-6 py-2 rounded-lg hover:bg-green-600">Light 1 ON</a>
```

```
      <a
href="/command?command=light1_off"
class="btn bg-red-500 text-white px-6 py-
2 rounded-lg hover:bg-red-600">Light 1
OFF</a>

    </div>

    <div class="flex justify-center space-x-
4">

      <a
href="/command?command=fan_on"
class="btn bg-green-500 text-white px-6
py-2 rounded-lg hover:bg-green-600">Fan
ON</a>

      <a
href="/command?command=fan_off"
class="btn bg-red-500 text-white px-6 py-
2 rounded-lg hover:bg-red-600">Fan
OFF</a>

    </div>

    <div class="flex justify-center space-x-
4">

      <a
href="/command?command=exhaust_on"
class="btn bg-green-500 text-white px-6
py-2 rounded-lg hover:bg-green-
600">Exhaust ON</a>

      <a
href="/command?command=exhaust_off"
class="btn bg-red-500 text-white px-6 py-
2 rounded-lg hover:bg-red-600">Exhaust
OFF</a>

    </div>

    <div class="flex justify-center space-x-
4">

      <a
href="/command?command=light_on"
class="btn bg-green-500 text-white px-6
py-2 rounded-lg hover:bg-green-
600">Light ON</a>

      <a
href="/command?command=light_off"
class="btn bg-red-500 text-white px-6 py-
2 rounded-lg hover:bg-red-600">Light
OFF</a>

    </div>

  </div>

 </div>

</body>

</html>

)=====";


const char* temperature_html = R"=====(

<!DOCTYPE html><html><head><meta
charset="UTF-8"><meta
name="viewport" content="width=device-
width"><meta http-equiv="refresh"
content="5"><title>Temperature</title><st
yle>body{font-
family:Arial;display:flex;justify-
content:center;align-
items:center;height:100vh;margin:0;backgr
ound:#f0f0f0;}div.screen{background:#fff;
width:320px;height:505px;border:1px
solid
#000;overflow:hidden;}div.header{backgr
ound:#3F51B5;color:white;text-
align:center;padding:10px;font-
size:20px;}div.content{padding:20px;text-
align:center;}div.reading{font-
size:24px;margin-top:20px;}a{text-
decoration:none;padding:5px
10px;border:1px solid
#000;color:#000;}a:hover{background:#d
dd;}</style><script>async function
f(){try{const r=await
```

```
fetch('/sensors');const d=await r.text();const[t,tp,h,g,f,l]=d.split(',');document.getElementById('t').innerText=tp;document.getElementById('h').innerText=h;}catch(e){console.error(e);}}window.onload=f;setInterval(f,5000);</script></head><body><div class="screen"><div class="header">TEMPERATURE</div><div class="content"><div class="reading">Temp: <span id="t">N/A</span>°C</div><div class="reading">Hum: <span id="h">N/A</span>%</div></div></div></body></html>

)=====";


String getDoorHtml() {

  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><meta name='viewport' content='width=device-width'><meta http-equiv='refresh' content='5'><title>Door</title>";

  html += "<style>body{font-family:Arial;display:flex;justify-content:center;align-items:center;height:100vh;margin:0;background:#f0f0f0;}div.screen{background:#fff;width:320px;height:505px;border:1px solid #000;overflow:hidden;}div.header{background:#3F51B5;color:white;text-align:center;padding:10px;font-size:20px;}div.content{padding:20px;text-align:center;}div.reading{font-size:24px;margin-top:20px;}div.buttons{margin-top:20px;}a{text-decoration:none;padding:10px 20px;margin:5px;border:1px solid #000;color:#000;}a:hover{background:#ddd;}</style>";

  html += "</head><body><div class='screen'><div class='header'>DOOR</div><div class='content'><div class='reading'>Status: " + doorStatus + "</div>";

  html += "<div class='buttons'><a href='/command?command=unlock_door'>Open</a><a href='/command?command=lock_door'>Close</a></div></div></div></body></html>";

  return html;

}


void parseSensorData(String data) {

  int comma1 = data.indexOf(',');

  int comma2 = data.indexOf(',', comma1 + 1);

  int comma3 = data.indexOf(',', comma2 + 1);

  int comma4 = data.indexOf(',', comma3 + 1);

  int comma5 = data.indexOf(',', comma4 + 1);

  int comma6 = data.lastIndexOf(',');


  if (comma1 != -1 && comma2 != -1 && comma3 != -1 && comma4 != -1 && comma5 != -1 && comma6 != -1) {

    timeStamp = data.substring(0, comma1);
```

```
    temperature = data.substring(comma1 + 1, comma2);
    humidity = data.substring(comma2 + 1, comma3);
    gasValue = data.substring(comma3 + 1, comma4);
    flameValue = data.substring(comma4 + 1, comma5);
    lightValue = data.substring(comma5 + 1, data.length());
    Serial.println("Parsed at " + String(millis()) + ": Time=" + timeStamp + ", Temp=" + temperature + ", Hum=" + humidity + ", Gas=" + gasValue + ", Flame=" + flameValue + ", Light=" + lightValue);
  } else {
    Serial.println("Parse error at " + String(millis()) + ": Invalid format - " + data);
    timeStamp = "Error";
    temperature = "Error";
    humidity = "Error";
    gasValue = "Error";
    flameValue = "Error";
    lightValue = "Error";
  }
}

void handleRoot() {
  String html = "<!DOCTYPE html><html><head><title>Smart Home</title><meta http-equiv='refresh' content='5'></head><body><h1>Smart Home</h1>";
  if (latestAlert != "") html += "<div style='color:red'><strong>Alert:</strong> " + latestAlert + "</div><br>";
  html += "<p>Time: " + timeStamp + "</p><p>Temp: " + temperature + "°C</p><p>Hum: " + humidity + "%</p><p>Gas: " + gasValue + "</p><p>Flame: " + flameValue + "</p><p>Light: " + lightValue + "%</p><p><a href='/automation'>Automation</a>|<a href='/temperature'>Temperature</a>|<a href='/door'>Door</a>|<a href='/test'>Test</a></p></body></html>";
  server.send(200, "text/html", html);
}

void handleCommand() {
  String command = server.arg("command");
  Serial.println("Cmd: " + command + " at " + String(millis()));
  if (command == "relay4_on") command = "light_on";
  if (command == "relay4_off") command = "light_off";
  arduinoSerial.println(command);
  if (command == "lock_door") {
    doorStatus = "Locked";
  } else if (command == "unlock_door") {
    doorStatus = "Unlocked";
  }
```

```
  server.send(200, "text/plain", "OK");
}

void handleSensorData() {
  server.send(200, "text/plain", timeStamp + "," + temperature + "," + humidity + "," + gasValue + "," + flameValue + "," + lightValue);
}

void handleReset() {
  Serial.println("Reset cmd at " + String(millis()));
  arduinoSerial.println("reset");
  server.send(200, "text/plain", "Resetting...");
}

void handleAutomation() {
  server.send(200, "text/html", automation_html);
}

void handleTemperature() {
  server.send(200, "text/html", temperature_html);
}

void handleDoor() {
  server.send(200, "text/html", getDoorHtml());
}

void handleTest() {
  String html = "<!DOCTYPE html><html><head><title>Test Data</title></head><body><h1>Enter Test Data</h1><form action='/setdata' method='get'><input type='text' name='data' placeholder='e.g., 12345,25.5,60,100,0,75'><input type='submit' value='Set'></form><p><a href='/'>Back</a></p></body></html>";
  server.send(200, "text/html", html);
}

void handleSetData() {
  String data = server.arg("data");
  if (data.length() > 0) {
    parseSensorData(data);
    Serial.println("Test data set: " + data + " at " + String(millis()));
  }
  server.sendHeader("Location", "/");
  server.send(303);
}

void setup() {
  Serial.begin(9600);
  arduinoSerial.begin(9600);
  Serial.println("Setup started at " + String(millis()));
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

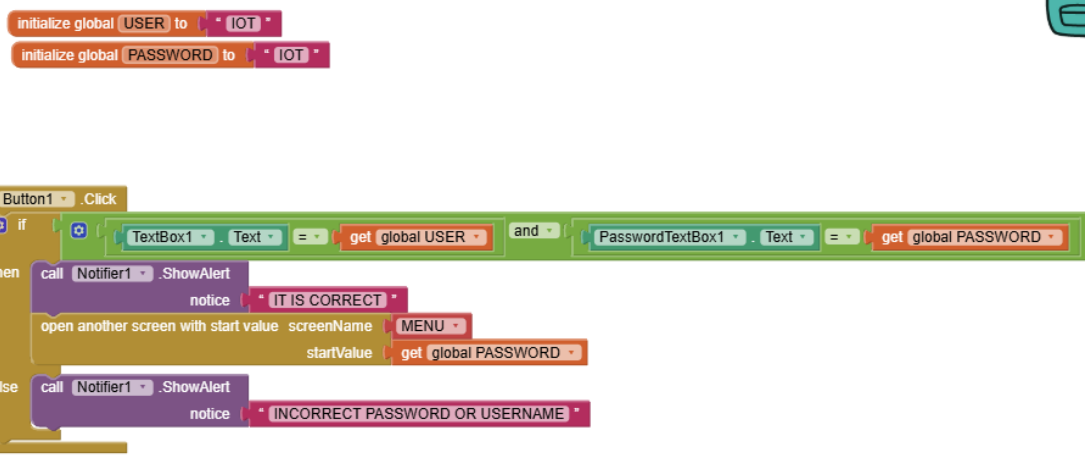
```
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected at " + String(millis()));
  Serial.println(WiFi.localIP());
  server.on("/", handleRoot);
  server.on("/command", handleCommand);
  server.on("/sensors", handleSensorData);
  server.on("/reset", handleReset);
  server.on("/automation", handleAutomation);
  server.on("/temperature", handleTemperature);
  server.on("/door", handleDoor);
  server.on("/test", handleTest);
  server.on("/setdata", handleSetData);
  server.begin();
  Serial.println("Server started at " + String(millis()));
}

void loop() {
  server.handleClient();
  while (arduinoSerial.available() > 0) {
    String receivedData = arduinoSerial.readStringUntil('\n');
    receivedData.trim();
    Serial.print("Raw data at " + String(millis()) + ": '");
    Serial.print(receivedData);
    Serial.println("'");
    if (receivedData.length() > 0) {
      if (receivedData.startsWith("ALERT:")) {
        Serial.println("Alert at " + String(millis()));
        latestAlert = receivedData;
      } else {
        parseSensorData(receivedData);
        latestAlert = "";
      }
    }
    arduinoSerial.flush();
  }
}
```

MIT APP INVENTOR:

Screen 1:

Block diagram:




12:30
MENU
DOOR LOCK
AUTOMATION
LIVE INFO
PASSWORD CHANGE

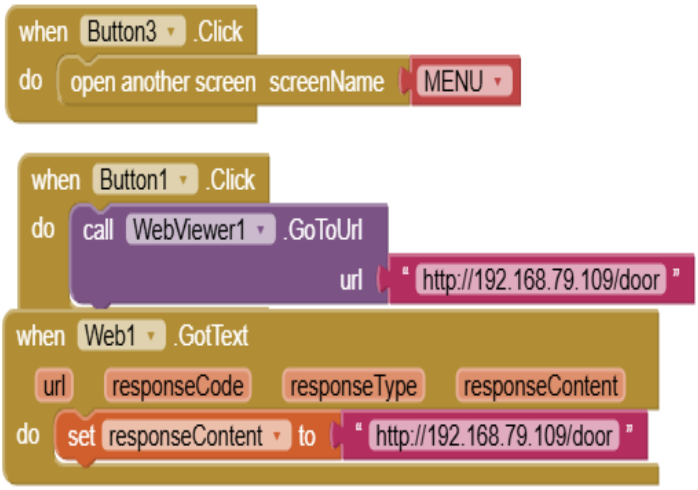

when Button1 .Click
do open another screen screenName DOOR

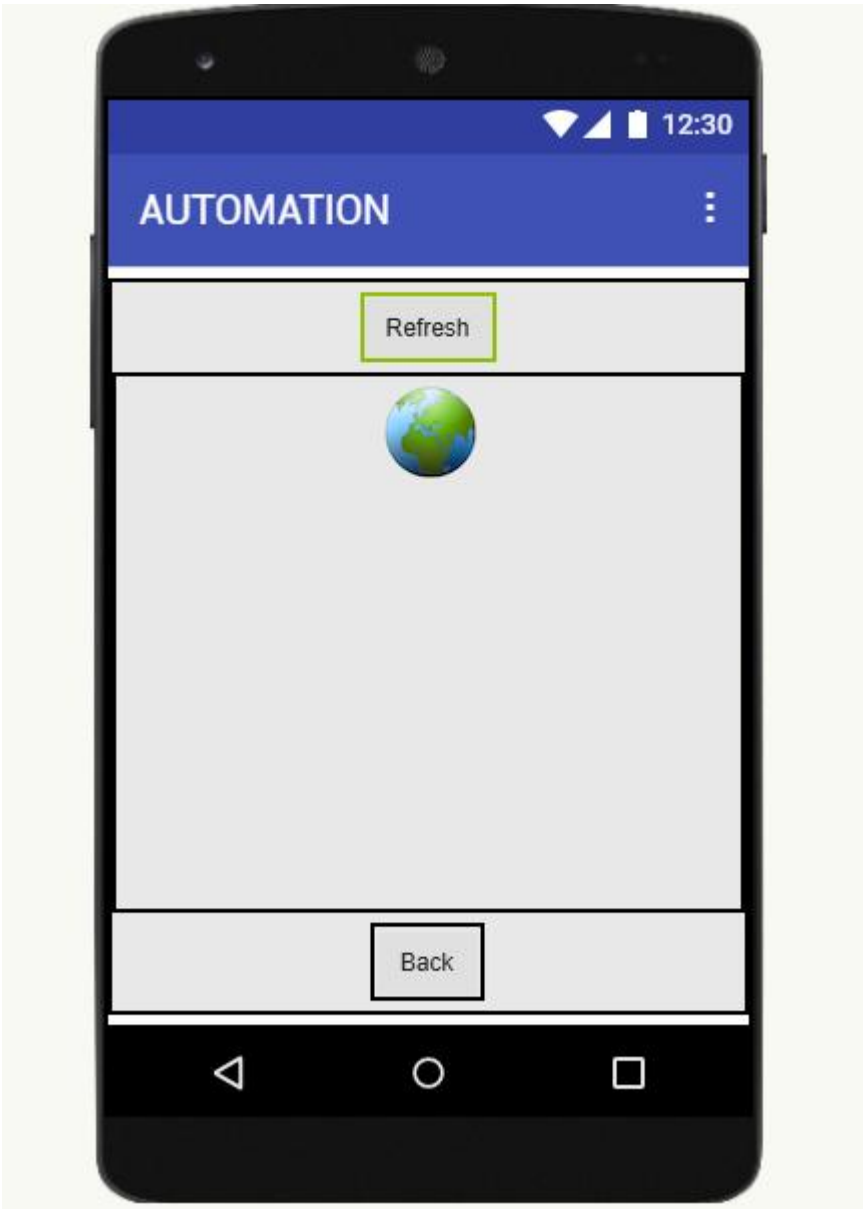

when Button2 .Click
do open another screen screenName AUTOMATION

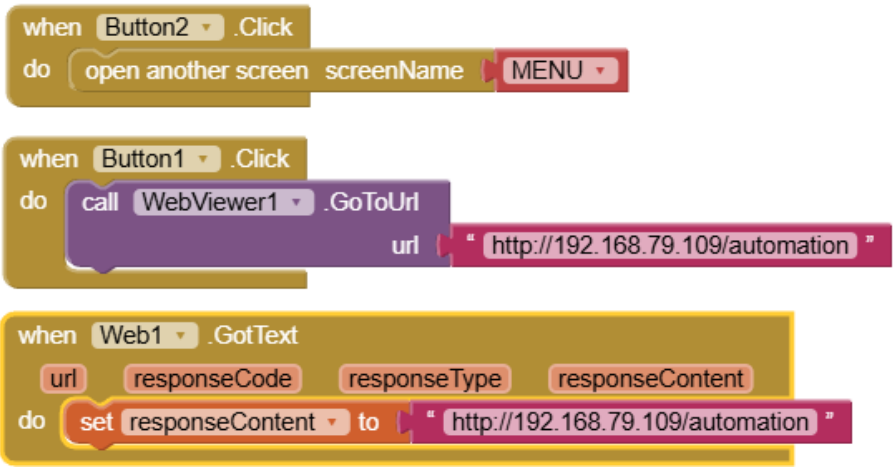

when Button3 .Click
do open another screen screenName TEMPERATURE

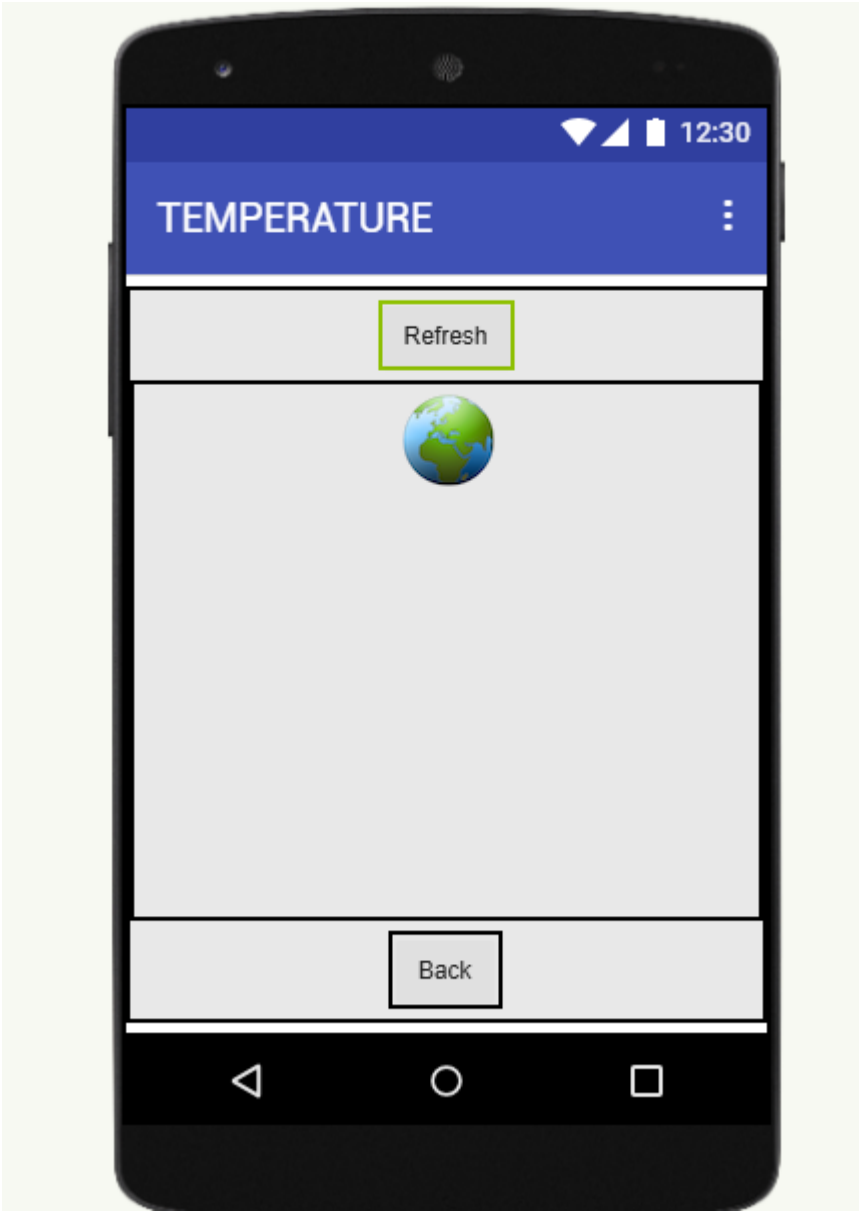

when Button4 .Click
do open another screen screenName PASSWORDCHANGE


12:30
DOOR
Refresh
Back

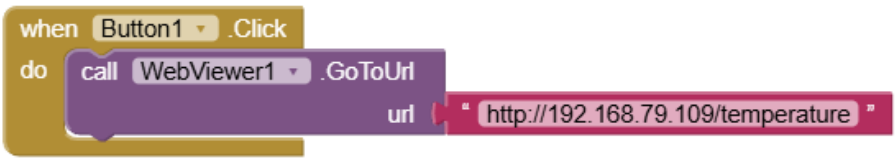

when Button3 .Click
do open another screen screenName MENU
when Button1 .Click
do call WebViewer1 .GoToUrl
url " http://192.168.79.109/door "
when Web1 .GotText
url responseCode responseType responseContent
do set responseContent to " http://192.168.79.109/door "

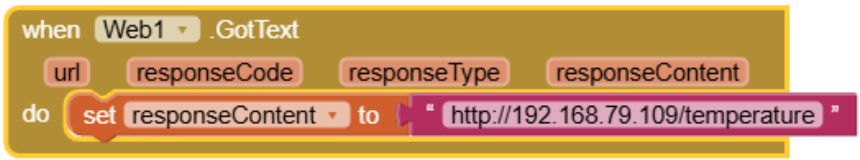

12:30
AUTOMATION
Refresh
Back

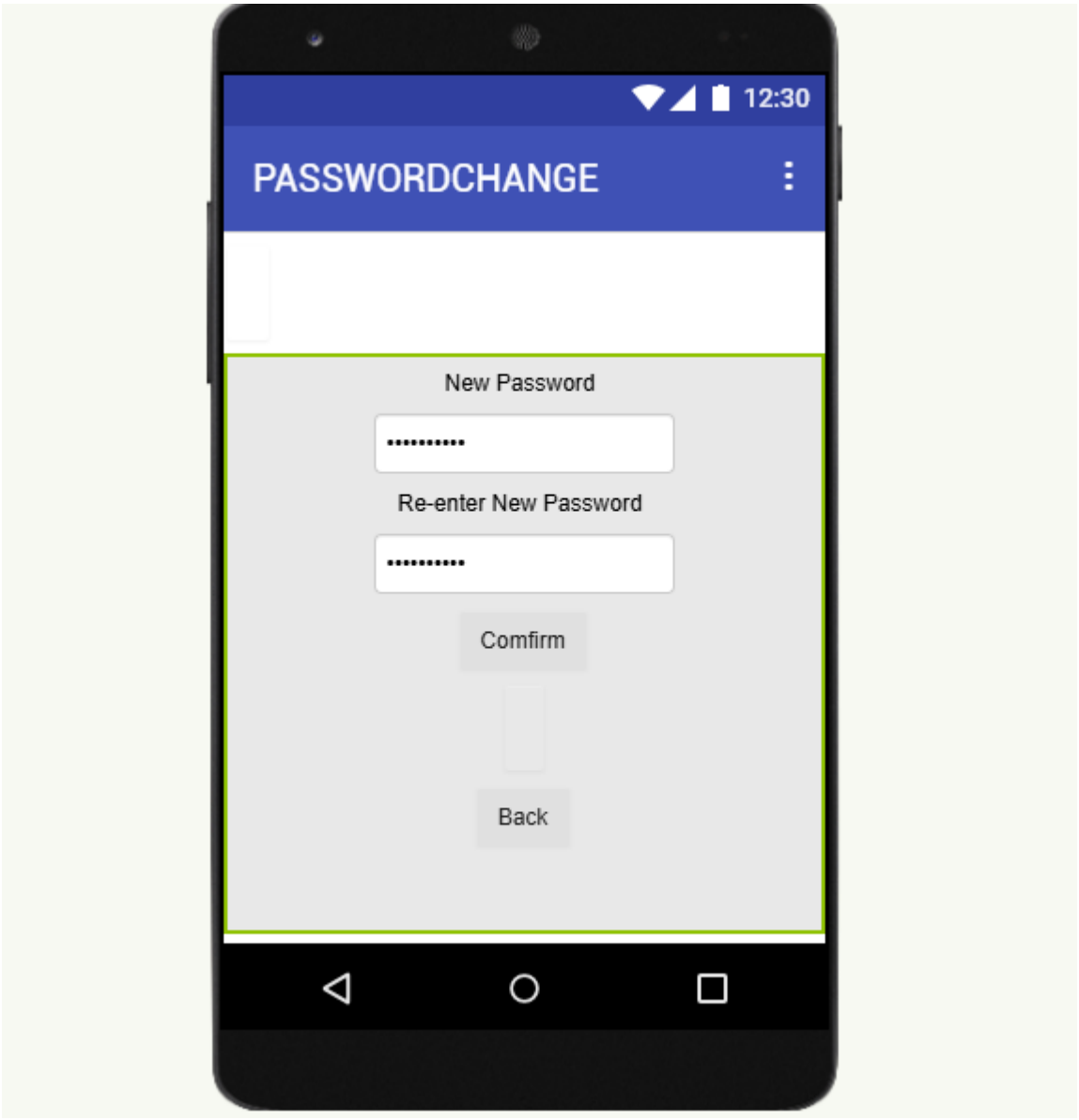

when Button2 .Click
do open another screen screenName MENU
when Button1 .Click
do call WebViewer1 .GoToUrl
url " http://192.168.79.109/automation "
when Web1 .GotText
url responseCode responseType responseContent
do set responseContent to " http://192.168.79.109/automation "

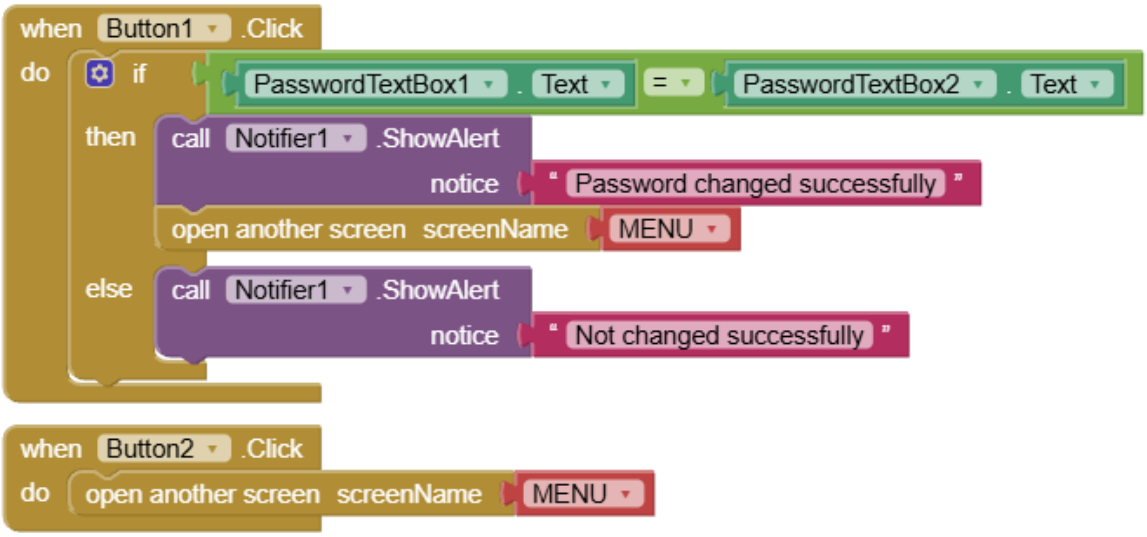

12:30
TEMPERATURE
Refresh
Back


when Button2 .Click
do open another screen screenName MENU

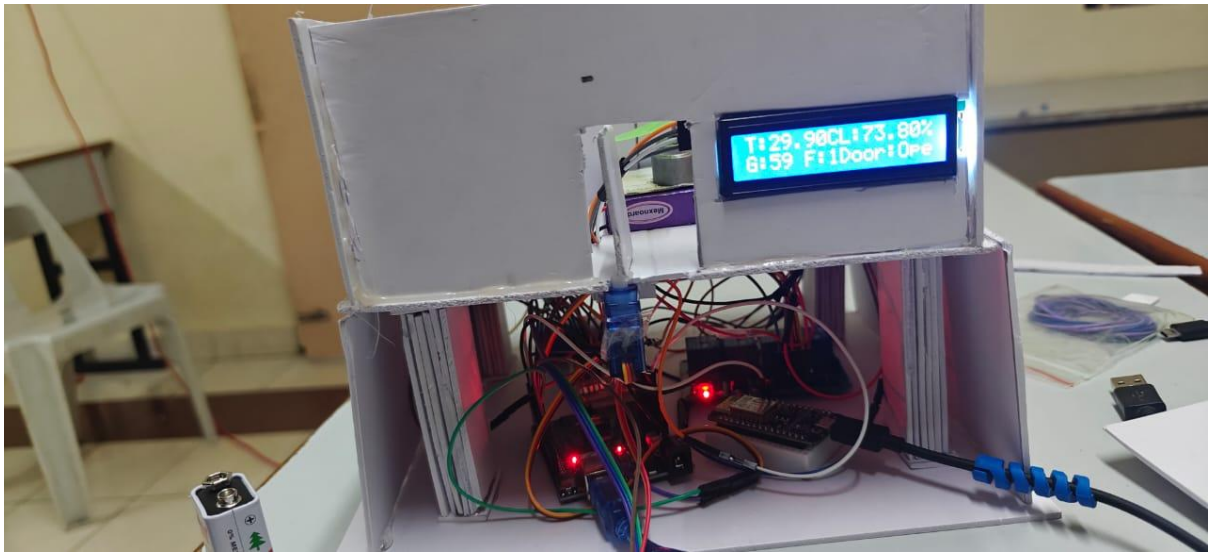

when Button1 .Click
do call WebViewer1 .GoToUrl
url " http://192.168.79.109/temperature "

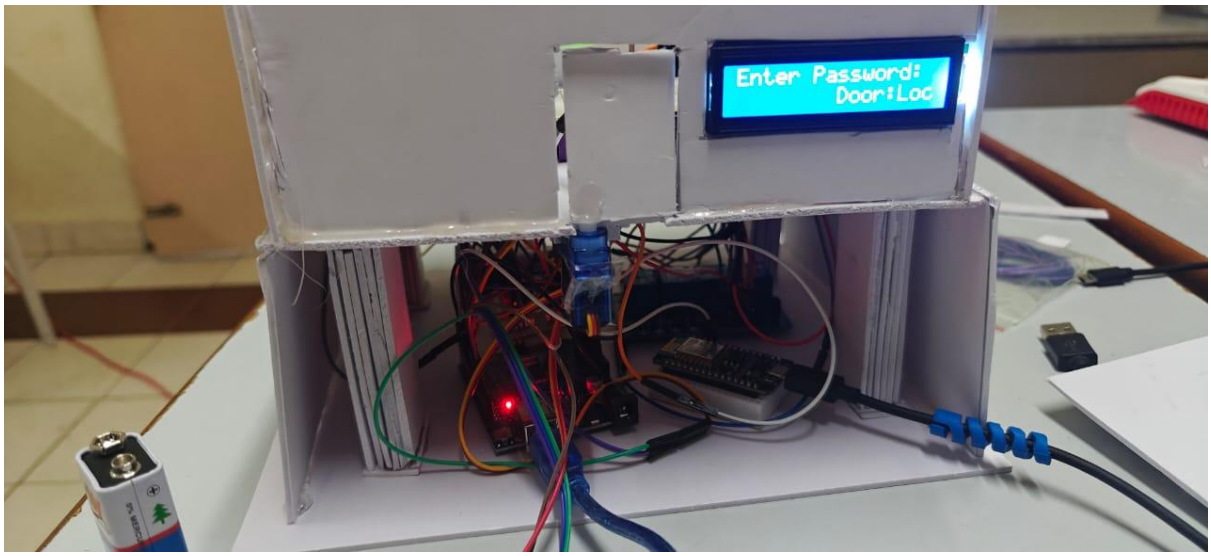

when Web1 .GotText
url responseCode responseType responseContent
do set responseContent to " http://192.168.79.109/temperature "

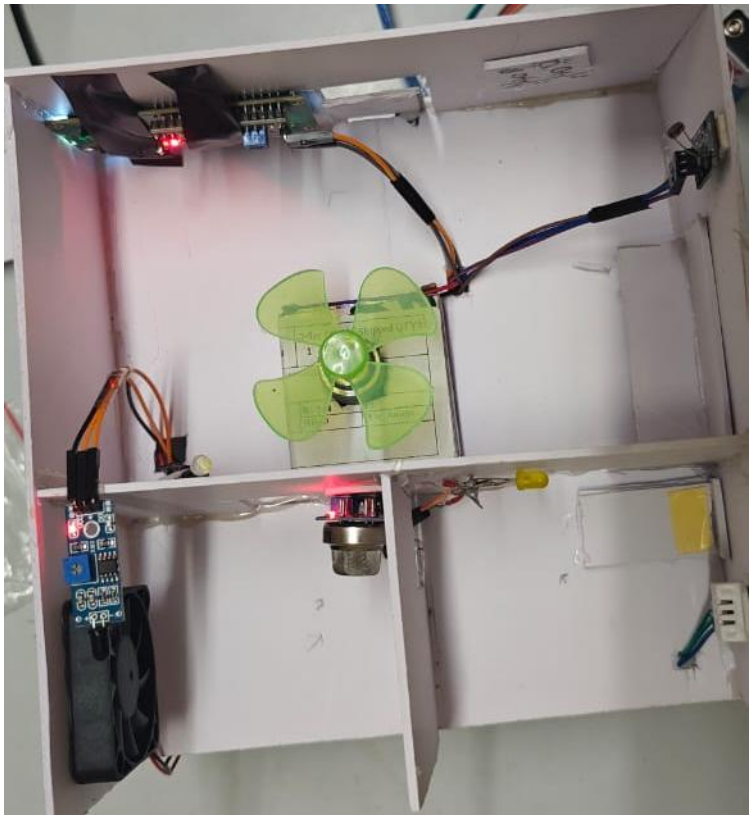

12:30
PASSWORDCHANGE
New Password
Re-enter New Password
Comfirm
Back

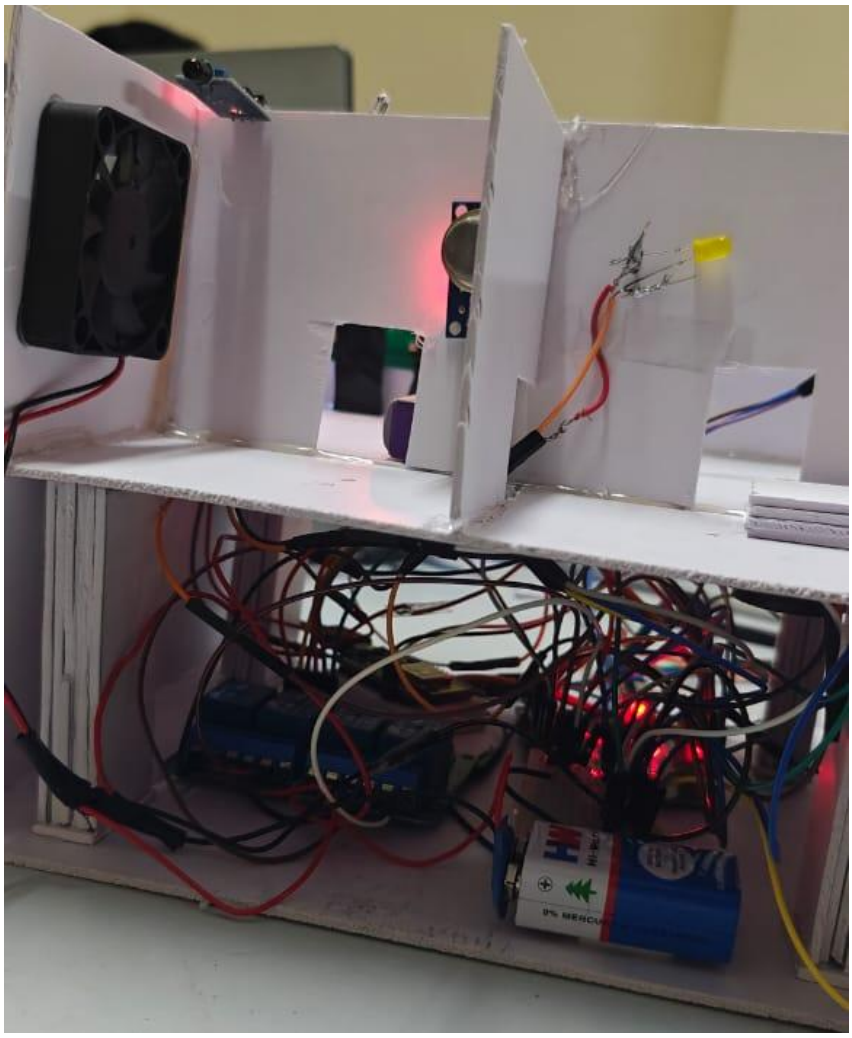

when Button1 .Click
do if PasswordTextBox1 . Text = PasswordTextBox2 . Text
then call Notifier1 .ShowAlert
notice " Password changed successfully "
open another screen screenName MENU
else call Notifier1 .ShowAlert
notice " Not changed successfully "
when Button2 .Click
do open another screen screenName MENU

Working Circuits:









OBSERVATION TABLE:

S. No	Scenario	Expected Output	Observed Output
1.	Correct password entered	Servo motor unlocks door	Door unlocked successfully
2.	Wrong password entered	Door remains locked, buzzer alert	Buzzer sounds, door stays locked
3.	Flame detected	Buzzer alert	Immediate buzzer
4.	Gas leak detected	Buzzer alert + exhaust fan	Immediate buzzer + exhaust fan turns immediately
5.	LDR	Lights turned on/off when it operated in the app	Lights are activated/inactivated
6.	Temperature/Humidity measured	Displayed in mobile app and in LCD	Real-time values shown
7.	Fan control from app	Motor turns ON/OFF	Motor successfully controlled
8.	Manual lock/unlock from app/keypad	Door locks/unlocks via app	Door responded correctly

RESULT ANALYSIS:

- The smart home prototype was able to successfully detect environmental changes (gas, flame, light) and respond appropriately by triggering alarms or controlling appliances.
- The mobile app provided seamless control and monitoring over Wi-Fi, allowing users to remotely operate and check the status of devices.
- Password-based door access added an extra layer of security to the home entrance.
- Real-time updates of temperature and humidity were accurately displayed on both the LCD and mobile app.

Overall, the project achieved its intended objectives of security, automation, monitoring, and remote access

CONCLUSION:

The Smart Home Automation System based on IoT principles has proven to be a **reliable**, **cost-effective**, and **scalable solution** for modern smart homes.
The use of Arduino and Wi-Fi module allowed for **easy prototyping** and **real-time control**.
The integration of security features (password access, fire, and gas detection) makes it highly suitable for real-world applications.
This project lays the foundation for further expansions into full-fledged **home automation ecosystems** involving AI, voice control, and cloud integration.

Problems Faced During Implementation:

1. Serial Communication Issues (NodeMCU ↔ Arduino)

- NodeMCU has only **one hardware Serial** (used for USB uploading), so using **SoftwareSerial** was tricky.
- **Baud Rate mismatch** between Arduino and NodeMCU caused garbage data or no communication.
- **Voltage mismatch**: Arduino works on **5V** logic, NodeMCU works on **3.3V** logic → direct connection sometimes damaged pins or caused unreliable communication (needed a **resistor voltage divider** or **level shifter**).
- **Data loss** if the Serial buffer overflowed (too many characters sent at once without delay).
- NodeMCU rebooting because of **serial noise** or **uncontrolled characters** received.

2. Webserver Problems (Mobile ↔ NodeMCU)

- Hosting the web page inside the limited memory (4MB or less) of NodeMCU caused **lagging** or **page crashes**.
- Mobile sometimes could **not find NodeMCU WIFI server** if IP changed (needed to set a **static IP**).
- If NodeMCU rebooted or reset, **server went down temporarily**, making mobile unable to connect.
- Multiple mobile connections overloaded NodeMCU.
- **CORS or browser caching** problems when testing repeatedly.

3. Sensor Reading and Control Issues (Arduino)

- If Arduino was **busy reading sensors**, it sometimes **missed incoming Serial commands** from NodeMCU
- Servo motor or relays **caused voltage dips** and Arduino reset if power supply was not stable (especially during multiple relay ON/OFF at once).
- Some sensors (like MQ2 Gas Sensor) needed **long warm-up time**; incorrect values early after boot.

4. Power Supply Problems

- Both NodeMCU and Arduino drawing power from USB caused **brownouts** and **unpredictable resets**.
- Needed a **separate 5V adapter** or a **stable power source** for relays and sensors.
- Ground (GND) between NodeMCU and Arduino must be **properly connected**, otherwise serial communication fails.

5. Programming and Flashing Conflicts

- While uploading code to NodeMCU, serial communication cable with Arduino must be **disconnected** — otherwise, upload errors occur.
- Sometimes ESP8266 firmware caused unexpected **watchdog timer resets** if Serial communication code wasn't optimized.

Solutions:

- Used **SoftwareSerial** carefully with lower baud rate (9600 bps).
- Added **voltage dividers** between Arduino TX → NodeMCU RX.
- Kept **serial data short and simple** (ex: "R1_ON" instead of long strings).
- Implemented **error checking** while reading Serial.
- Used **millis()** for non-blocking code instead of **delay()**.
- Used **separate 5V 2A supply** for Arduino and Relays.
- Set **Static IP** for NodeMCU to avoid mobile app disconnection.
- Created **simple HTML pages** to reduce NodeMCU webserver load.

Future Scope:

1. Full Wi-Fi Automation without Arduino

- Replace Arduino with NodeMCU alone by connecting sensors and relays directly.
- This reduces hardware complexity, cost, and power consumption.

2. Cloud Integration

- Connect the system to **Firebase**, **Blynk**, or **ThingSpeak** to monitor and control devices over the Internet (not just within the same Wi-Fi network).
- Implement **data logging** for temperature, gas levels, moisture, etc.

3. Mobile App Development

- Build a dedicated **Android/iOS app** (using **MIT App Inventor**, **Flutter**, etc.) for more professional control instead of using a web browser.

4. Voice Assistant Integration

- Integrate the system with **Google Assistant**, **Alexa**, or **Siri** for voice control of home devices.

5. Security Enhancements

- Add **authentication and encryption** to the webserver to prevent unauthorized access.
- Implement **fire alerts** and **intruder detection** with SMS or email notifications.

6. Energy Efficiency Improvements

- Use **automatic scheduling** for fans, lights, water pumps based on real-time sensor data to save energy.
- Introduce **machine learning models** to predict usage patterns.

7. Expandability

- Add more modules like **CCTV camera monitoring**, **doorbell camera**, **RFID access control**, or **solar panel integration**.

8. OTA (Over-the-Air) Updates

- Implement **OTA firmware update** feature so that NodeMCU can receive software updates remotely without USB connection.