Arduino:

```
#include <Keypad_I2C.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Buzzer.h>
#include <Servo.h>
#include <EEPROM.h>
#include <SoftwareSerial.h>

#define I2CADDR 0x20
#define LCD_ADDR 0x27

Servo servo;
SoftwareSerial ArduinoUno(10, 9);
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] = { 7, 6, 5, 4 };
byte colPins[COLS] = { 3, 2, 1, 0 };

Keypad_I2C keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS, I2CADDR);

LiquidCrystal_I2C lcd(LCD_ADDR, 16, 2);

const int motorPin = 2;
const int buzzerPin = 3;
const int ledPin = 4;
const int tempPin = A0;
const int lightSensorPin = A1;
const int buttonPin = A2;
const int ECHO_PIN = 6;
const int TRIG_PIN = 5;
const long interval = 4000;
unsigned long previousMillis = 0;
int inputCount = 0;
int timeLock = 0;
bool doorOpen = false;
const int distanceThreshold = 20;  // 20cm
```

```
int val;
const int ANALOG_THRESHOLD = 20;

String D2003 = "1234";
String D2004 = "5678";
String D2005 = "1357";


bool autoLedControl = false;

void setup() {
  Serial.begin(115200);
  Wire.begin();  // Initialize I2C communication
  keypad.begin(makeKeymap(keys));
  lcd.init();       // Initialize the LCD
  lcd.backlight();  // Turn on backlight
  lcd.clear();      // Clear the LCD screen
  lcd.setCursor(0, 0);
  servo.attach(motorPin);
  servo.write(0);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);          // LED pin as output
  pinMode(lightSensorPin, INPUT);  // Light sensor pin as input
  pinMode(buttonPin, INPUT);
  pinMode(tempPin, INPUT);  // Button pin as input

  Serial.println("Bat dau");
  if (readStringFromEEPROM(0) == "") {
    writeStringToEEPROM(0, D2003);
  } else {
    D2003 = readStringFromEEPROM(0);
  }
  if (readStringFromEEPROM(5) == "") {
    writeStringToEEPROM(5, D2004);
  } else {
    D2004 = readStringFromEEPROM(5);
  }
  if (readStringFromEEPROM(10) == "") {
    writeStringToEEPROM(10, D2005);
  } else {
    D2005 = readStringFromEEPROM(10);
  }
}
```

```
float tempRoom(int tempPin) {
  int val = analogRead(tempPin);
  float millivolts = (val / 1024.0) * 5000;
  float cel = millivolts / 10;
  return cel;
}
void loop() {
  if (Serial.available()) {
    String message = Serial.readStringUntil('\n');

    // control servo motor
    if (message.startsWith("SERVO:")) {
      int position = message.substring(6).toInt();
      servo.write(position);
      delay(3000);
      closeDoor();
    }

    // control buzzer
    if (message.startsWith("BUZZER:")) {
      int state = message.substring(7).toInt();
      digitalWrite(buzzerPin, state);
    }

    // control led
    if (message.startsWith("LED:")) {
      int state = message.substring(4).toInt();

      if (state == 1) {
        Serial.println("LED is ON");
        digitalWrite(ledPin, HIGH);
      } else if (state == 0) {
        Serial.println("LED is OFF");
        digitalWrite(ledPin, LOW);
      }
    }
     // control auto led
    if (message.startsWith("AUTOLED")) {
      if (message.startsWith("AUTOLEDON")) {
        autoLedControl = true;
      } else if (message.startsWith("AUTOLEDOFF")) {
        autoLedControl = false;
      }
    }
  }
```

```
  // Read light sensor value
  unsigned long currentMillis = millis();
  int lightValue = analogRead(lightSensorPin);
  int buttonState = digitalRead(buttonPin);
  int distance = measureDistance();
  float temp = tempRoom(tempPin);
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    Serial.print(temp);
    Serial.print(",");
    Serial.print(lightValue);
    Serial.print(",");
    Serial.print(distance);
    Serial.println();
    if (autoLedControl) {
      ledControl(lightValue);
    }
  }



  // If the button is pressed, open the door
  if (buttonState == LOW) {
    if (distance < distanceThreshold && !doorOpen) {
      String enteredPassword = getPasswordInput(false);
      String passEEPROM1 = readStringFromEEPROM(0);
      String passEEPROM2 = readStringFromEEPROM(5);
      String passEEPROM3 = readStringFromEEPROM(10);
      if (verifyPassword(enteredPassword, passEEPROM1) ||
verifyPassword(enteredPassword, passEEPROM2) || verifyPassword(enteredPassword,
passEEPROM3)) {
        lcd.clear();
        beepBuzzer();
        lcd.print("Door Opened!");
        inputCount = 0;
        openDoor();
        delay(3000);
        closeDoor();
      } else if (enteredPassword == "change") {
        lcd.clear();
        lcd.print("Change success");
        delay(2000);
      } else if (enteredPassword == "not change") {
        lcd.clear();
        lcd.print("Change failed!");
```

```
        delay(2000);
      } else if (enteredPassword == "Timeout! No input.") {
        lcd.clear();
        lcd.print("Timeout!");
        delay(1000);
      } else {
        inputCount++;
        if (inputCount >= 3) {
          lockKeypad();
        } else {
          lcd.clear();
          lcd.print("Wrong password!");
          delay(1000);
          lcd.clear();
        }
      }
    } else if (distance > distanceThreshold) {
      lcd.clear();
    }
    delay(100);
  } else {
    beepBuzzer();
    openDoor();
    lcd.print("Door Opened!");
    delay(3000);  // Keep the door open for 5 seconds
    closeDoor();
    lcd.clear();
    lcd.print("Door Closed!");
    delay(2000);
  }
}

int ledControl(int lightValue) {
  if (lightValue < ANALOG_THRESHOLD) {
    digitalWrite(ledPin, HIGH);  // turn on LED
  } else {
    digitalWrite(ledPin, LOW);
  }
}

int measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  int distance = duration / 29 / 2;
  return distance;
}

void writeStringToEEPROM(int addrOffset, String strToWrite) {
  byte len = strToWrite.length();
  EEPROM.write(addrOffset, len);
  for (int i = 0; i < len; i++) {
    EEPROM.write(addrOffset + 1 + i, strToWrite[i]);
  }
}

String readStringFromEEPROM(int addrOffset) {
  int newStrLen = EEPROM.read(addrOffset);
  char data[newStrLen + 1];
  for (int i = 0; i < newStrLen; i++) {
    data[i] = EEPROM.read(addrOffset + 1 + i);
  }
  data[newStrLen] = '\0';
  return String(data);
}

boolean verifyPassword(String enteredPassword, String password) {
  return enteredPassword.indexOf(password) != -1;
}

String getPasswordInput(boolean changePass) {
  lcd.clear();
  unsigned long startTime = millis();
  unsigned long timeout = 20000;
  if (changePass == false) {
    lcd.print("Enter password :");
  } else {
    lcd.print("New password: ");
  }
  String lcdPass = "";
  String input = "";
  char key;
  while (true) {
    key = keypad.getKey();
    if (key) {
      if (key == '#') {
        break;
```

```
      } else if (key == 'C') {
        boolean success = changePassword();
        if (success) {
          return "change";
        } else {
          return "not change";
        }
      } else if (key == 'D') {
        lcdPass = "";
        input = "";
        lcd.clear();
        lcd.print("Enter password :");
      } else if (key == 'B') {
        writeStringToEEPROM(0, "1234");
        writeStringToEEPROM(5, "5678");
        writeStringToEEPROM(10, "1357");
      } else {
        lcd.setCursor(0, 1);
        lcdPass += "*";
        lcd.print(lcdPass);
        input += key;
      }
    }
    if (millis() - startTime > timeout) {
      return "Timeout! No input.";
    }
  }
  return input;
}

boolean changePassword() {
  lcd.clear();
  lcd.print("Before password:");

  String lcdPass = "";
  String input = "";
  char key;

  while (true) {
    key = keypad.getKey();
    if (key) {
      if (key == '#') {
        break;
      } else {
        lcd.setCursor(0, 1);
```

```
        lcdPass += "*";
        lcd.print(lcdPass);
        input += key;
      }
    }
  }
  String passEEPROM = readStringFromEEPROM(0);
  String newPass = "";
  if (input == D2003 || input == passEEPROM) {
    newPass = getPasswordInput(true);
    D2003 = newPass;
    writeStringToEEPROM(0, D2003);
    return true;
  } else if (input == D2004) {
    newPass = getPasswordInput(true);
    D2004 = newPass;
    writeStringToEEPROM(5, D2004);
    return true;
  } else if (input == D2005) {
    newPass = getPasswordInput(true);
    D2005 = newPass;
    writeStringToEEPROM(10, D2005);
    return true;
  } else {
    lcd.clear();
    lcd.print("Invalid password");
    delay(2000);
  }
  return false;
}

void incorrectPass() {
  lcd.print("Lock surcurity!");
  tone(buzzerPin, 1000, 3000);
  delay(2000);
  noTone(buzzerPin);
  lcd.setCursor(0, 0);
}

void openDoor() {
  servo.write(180);
  doorOpen = true;
}

void closeDoor() {
```

```
  servo.write(0);
  doorOpen = false;
}

void lockKeypad() {
  lcd.clear();
  lcd.print("Keypad locked!");
  String countDown = "";
  tone(buzzerPin, 1000, 3000);

  lcd.setCursor(0, 1);
  timeLock += 3000;

  for (int i = timeLock / 1000; i > 0; i--) {
    lcd.setCursor(0, 1);
    if (i < 10) {
      lcd.print("0");
    }
    lcd.print(i);
    delay(1000);
  }
  lcd.clear();
  lcd.print("Unlocking...");
  delay(1000);
}
void beepBuzzer() {
  digitalWrite(buzzerPin, HIGH);
  delay(500);  // Đợi 0,5 giây
  digitalWrite(buzzerPin, LOW);
  delay(500);  // Đợi 0,5 giây
}
```

ESP8266:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <SoftwareSerial.h>
String URL = "http://api.thingspeak.com/update?api_key=LLI62EG4A7RXEM91&field1=";

const char *ssid = "iot";           // Enter your WIFI SSID
```

```
const char *password = "12345678";  // Enter your WIFI Password

#define BOTtoken "7169920821:AAEvMk3EmQywc1e-AZQU6VVSaP-cQ34MULI"  // Enter the
bottoken you got from botfather
#define CHAT_ID "1241961204"                                        // Enter your
chatID you got from chatid bot

SoftwareSerial ESP82666(D2, D3);

X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
unsigned long lastTime = 0;     // initialize it in setup()
unsigned long interval = 3000;  // the time we need to wait
int botRequestDelay = 100;
unsigned long lastTimeBotRan;
void setup() {
  Serial.begin(115200);
  pinMode(D2, INPUT);
  pinMode(D3, OUTPUT);
  configTime(0, 0, "pool.ntp.org");
  client.setTrustAnchors(&cert);
  WiFi.disconnect();
  delay(1000);
  Serial.print("Start connection");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while ((!(WiFi.status() == WL_CONNECTED))) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  bot.sendMessage(CHAT_ID, "Wifi Connected!", "");
  bot.sendMessage(CHAT_ID, "System has Started!!", "");
  lastTime = millis();
}

void loop() {
  if (millis() > lastTimeBotRan + botRequestDelay) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while (numNewMessages) {
```

```
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
  while (Serial.available() == 0) {
    ;  // wait until data is fully available
  }
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    int comma1 = data.indexOf(',');
    int comma2 = data.lastIndexOf(',');
    if (comma1 != -1 && comma2 != -1 && comma1 != comma2) {
      float temp = data.substring(0, comma1).toFloat();
      int light = data.substring(comma1 + 1, comma2).toInt();
      int distance = data.substring(comma2 + 1).toInt();
      // Serial.print("nhietdo: ");
      // Serial.println(temp);
      // Serial.print("anh sang: ");
      // Serial.println(light);
      // Serial.print("khoangcach: ");
      // Serial.println(distance);
      sendData(temp, light, distance);
      if (distance <= 10 && distance >= 0) {
        bot.sendMessage(CHAT_ID, "Co nguoi den", "");
      }
      if (temp > 80) {
        bot.sendMessage(CHAT_ID, "Nhiet do phong dang cao", "");
      }
    }
  }
}

void sendData(float temp, int light, int distance) {
  WiFiClient client;
  HTTPClient http;
  String newUrl = URL + String(temp) + "&field2=" + String(light) + "&field3=" +
String(distance);
  http.begin(client, newUrl);
  int responsecode = http.GET();
  String data = http.getString();
  http.end();
}
void handleNewMessages(int numNewMessages) {
  for (int i = 0; i < numNewMessages; i++) {
```

```
    // Chat id of the requester
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID) {
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }

    // Print the received message
    String text = bot.messages[i].text;
    String from_name = bot.messages[i].from_name;

    if (text == "/start") {
      String welcome = "Welcome, " + from_name + ".\n";
      welcome += "Use the following commands to control your outputs.\n\n";
      welcome += "/led_on to turn GPIO ON \n";
      welcome += "/led_off to turn GPIO OFF \n";
      welcome += "/cuamo to open door \n";
      welcome += "/ledautoon to open door \n";
      welcome += "/ledautooff to open door \n";
      welcome += "/state to request current GPIO state \n";
      bot.sendMessage(chat_id, welcome, "");
    }

    if (text == "/led_on") {
      bot.sendMessage(chat_id, "LED state set to ON", "");
      Serial.println(String("LED:") + HIGH);
    }

    if (text == "/led_off") {
      bot.sendMessage(chat_id, "LED state set to OFF", "");
      Serial.println(String("LED:") + LOW);
    }

    if (text == "/cuamo") {
      bot.sendMessage(chat_id, "Mo cua", "");
      Serial.println(String("SERVO:") + 180);
    }

    if (text == "/ledautoon") {
      bot.sendMessage(chat_id, "Led auto on", "");
      Serial.println(String("AUTOLEDON"));
    }
    if (text == "/ledautooff") {
      bot.sendMessage(chat_id, "Led auto off", "");
      Serial.println(String("AUTOLEDOFF"));
```

```
    }
  }
}
```