

Smart Garage Door

Corso di
Laboratorio IOT

By

Francesco Ranno - A13002382
Antonio Celiento - A13002205
Davide Flagiello - A13002377

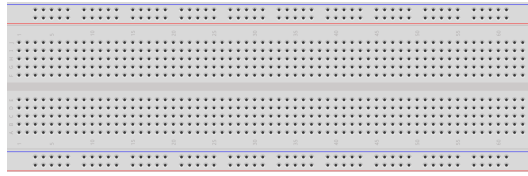


Dipartimento di Ingegneria Informatica
Università degli Studi della Campania Luigi Vanvitelli

Indice

| | | |
|----------|--------------------------------------------------|-----------|
| 1 | Introduzione | 3 |
| 2 | Componenti utilizzati e collegamenti | 3 |
| 2.1 | Componenti | 3 |
| 3 | Base di dati | 5 |
| 3.1 | Progettazione logica | 5 |
| 3.2 | Progettazione concettuale | 5 |
| 4 | Trasmissione Dati | 5 |
| 4.1 | Generazione dei Dati | 5 |
| 4.2 | Trasmissione dei Dati | 5 |
| 5 | Scenari e Casi d'uso | 6 |
| 5.1 | Requisiti funzionali | 6 |
| 5.2 | Requisiti non funzionali | 6 |
| 5.3 | Casi d'uso | 6 |
| 5.4 | Scenari | 9 |
| 6 | Codice ARDUINO | 12 |
| 6.1 | Librerie utilizzate | 12 |
| 6.2 | Codice prima esp32 | 12 |
| 6.3 | Codice seconda esp32 | 13 |
| 6.4 | Codice per apertura/chiusura Bluetooth | 15 |
| 7 | Conclusione | 16 |

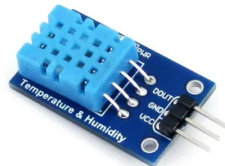
2x Breadboard



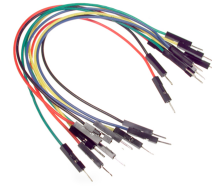
2x Servo Motor SG90



1x DHT Sensor



11x Jumper M/M



2x Freenove ESP32



1x LED Component



Tabella 1: Tabella dei componenti.

1 Introduzione

Il presente progetto propone la realizzazione di un sistema Smart Garage Door basato su ESP32 e Arduino, due tra le piattaforme più utilizzate nell'ambito dell'IOT. Attraverso l'integrazione di diverse componenti, descritte nel dettaglio poco più avanti nel documento, il sistema permetterà agli utenti di aprire, chiudere la porta del garage da remoto e anche di monitorare diversi parametri all'interno della stanza, tutto tramite un'interfaccia semplice e intuitiva. Gli utenti potranno accedere ad uno storico dettagliato dell'utilizzo attraverso l'impiego di un database per la memorizzazione di dati utili per avere una supervisione totale dell'ambiente.

2 Componenti utilizzati e collegamenti

2.1 Componenti

Nella Tabella 1 sono illustrati tutti i componenti utilizzati per questo progetto. Il componente principale è sicuramente il Servo Motor SG90 che utilizza 3 cavi: rosso, arancione e marrone. Il cavo rosso è il cavo di alimentazione positivo (+), il cavo marrone è il cavo collegato al ground (GND) (-) della nostra scheda di controllo e per ultimo il cavo arancione viene utilizzato per il segnale di controllo. Il segnale inviato tramite quest'ultimo cavo determina l'angolo di rotazione del servo motor¹. Tutti i collegamenti sono ben visibili in Figura 1 e in Figura 2.

¹Come descritto nella datasheet del componente il servo motor può rotare approssimativamente di 180° ossia 90° in senso orario oppure in senso antiorario

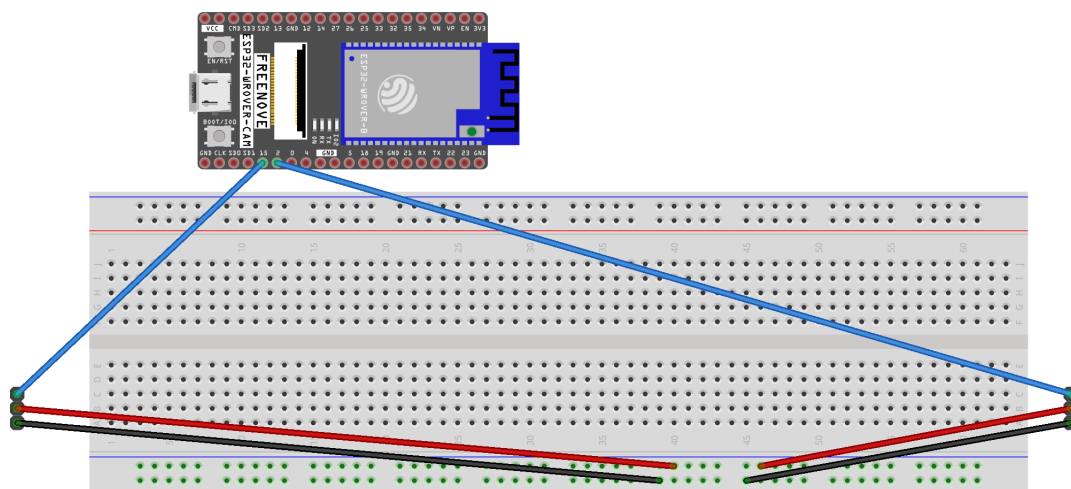


Figura 1: Collegamenti della prima esp32

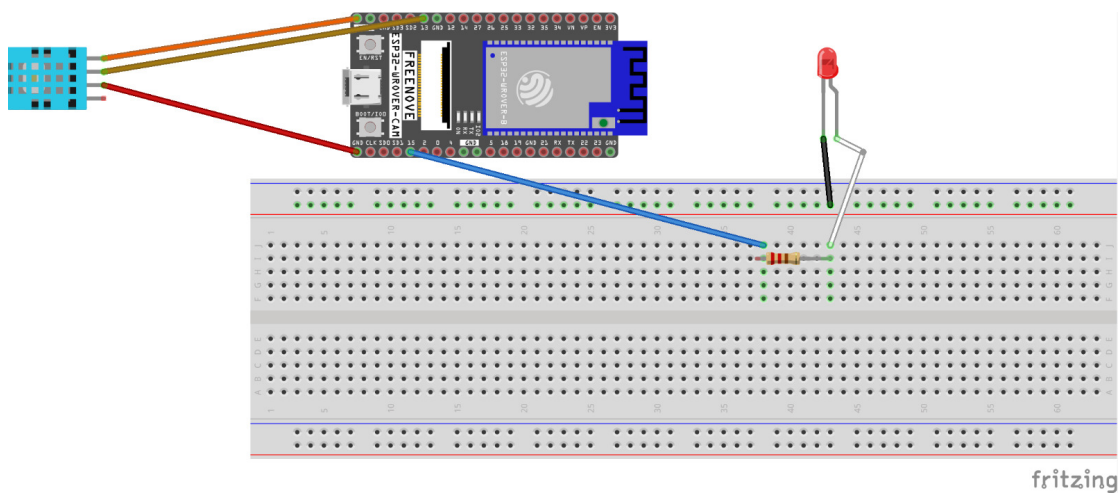


Figura 2: Collegamenti della seconda esp32

3 Base di dati

Progettare la base di dati necessaria per gestire i dati generati dai dispositivi IoT. Questa sezione deve contenere la progettazione concettuale, logica e fisica della base di dati utilizzata.

3.1 Progettazione logica

Utenti(ID, Email, Password_hash)

Valori_Sensori(ID, temperatura, timestamp, umidita)

Stato_Garage(ID, stato_luce, stato_porta, timestamp)

3.2 Progettazione concettuale

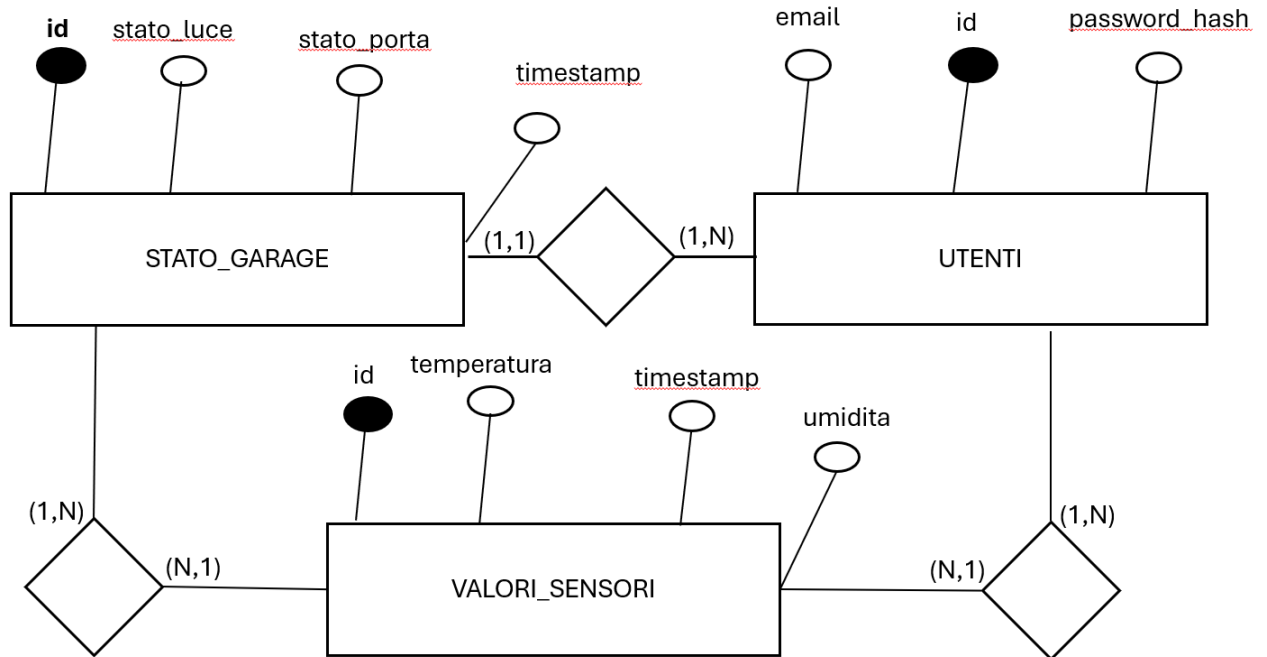


Figura 3: Progettazione concettuale

4 Trasmissione Dati

Il ciclo di vita dei dati inizia con la generazione da parte dei sensori, seguita dalla trasmissione tramite HTTP al server PHP, che infine memorizza i dati nella base di dati. La scelta del protocollo HTTP è motivata dalla sua semplicità e dalla disponibilità di supporto nelle librerie ESP32. La trasmissione dei dati avviene in maniera asincrona, attivata da specifiche richieste HTTP basate sugli eventi rilevati.

4.1 Generazione dei Dati

- **Sensore DHT11:** Misura la temperatura e l'umidità e invia i dati al microcontroller ESP32.
- **Pin di Controllo Luce (LUCE_PIN):** Controlla lo stato della luce (accesa o spenta) tramite il microcontroller ESP32.
- **Servo Motori:** Controllano l'apertura e la chiusura del garage, gestiti anch'essi dall'ESP32.

4.2 Trasmissione dei Dati

- **Connessione WiFi:** L'ESP32 si connette alla rete WiFi utilizzando le credenziali specificate (ssid e password).

- **Server Web:** L'ESP32 ospita un server web sulla porta 80, che gestisce richieste HTTP.
- **Richieste HTTP Client:** L'ESP32 invia richieste HTTP GET ai server PHP per memorizzare i dati e aggiornare gli stati.
- **Baudrate:** settato a 115200, è una velocità relativamente alta che consente una rapida trasmissione di dati tra l'ESP32 e il computer.

5 Scenari e Casi d'uso

5.1 Requisiti funzionali

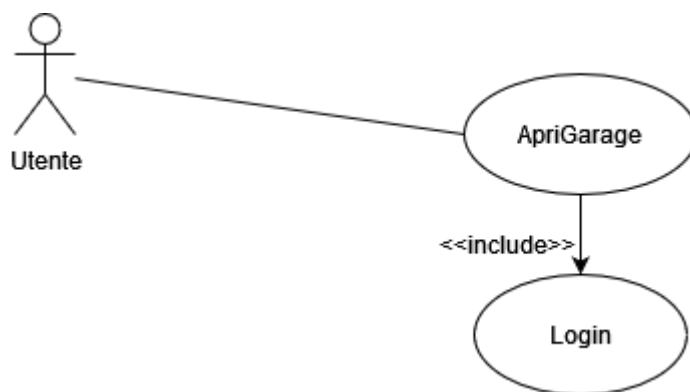
- Il sistema consentirà all'utente di aprire la porta del garage da remoto.
- Il sistema consentirà all'utente di chiudere la porta del garage da remoto.
- Il sistema consentirà all'utente di aprire la porta del garage tramite bluetooth.
- Il sistema consentirà all'utente di chiudere la porta del garage tramite bluetooth.
- Il sistema consentirà la creazione di un account per poter visualizzare i dati dei sensori all'interno del garage.
- Il sistema consentirà all'utente di accendere la luce all'interno del garage.
- Il sistema consentirà all'utente di spegnere la luce all'interno del garage.

5.2 Requisiti non funzionali

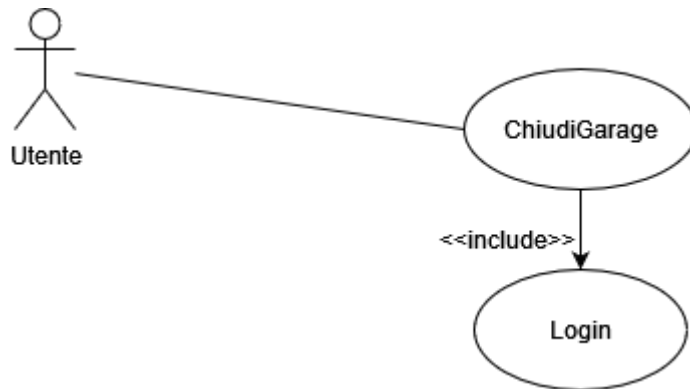
- **Affidabilità:** Il sistema deve essere affidabile, garantendo che la porta del garage si apra e si chiuda correttamente in risposta ai comandi dell'utente.
- **Facilità d'Uso:** L'interfaccia utente del sistema, inclusa l'applicazione mobile, deve essere intuitiva e di facile utilizzo per gli utenti.
- **Sicurezza:** Il sistema deve garantire la sicurezza dell'accesso al garage, impedendo l'accesso non autorizzato.

5.3 Casi d'uso

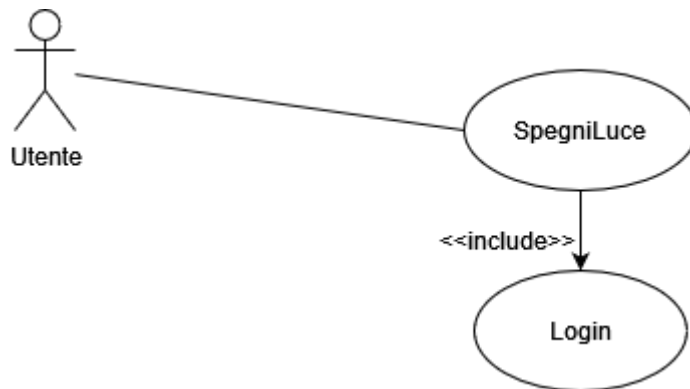
5.3.1 Apri Garage



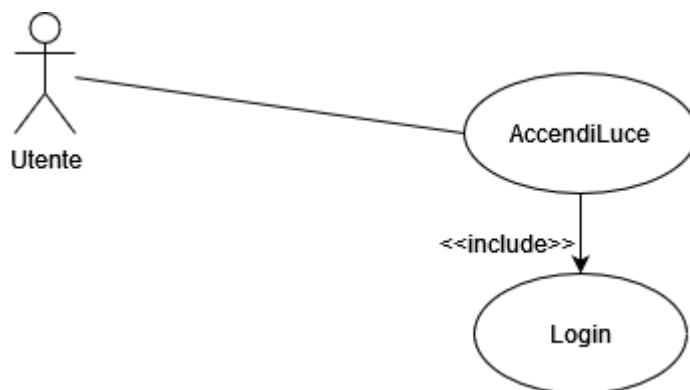
5.3.2 Chiudi Garage



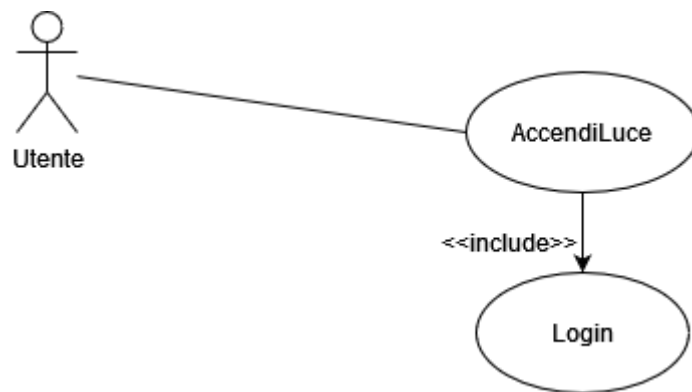
5.3.3 Visualizza Dati



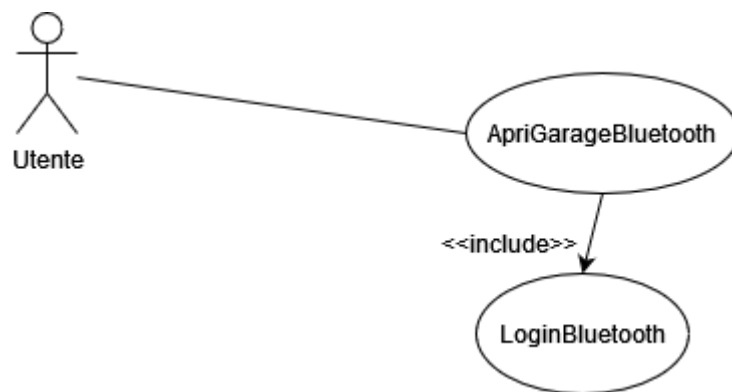
5.3.4 Accendi Luce



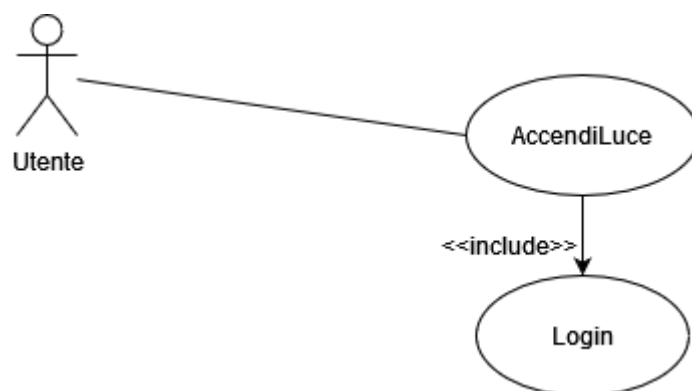
5.3.5 Spegni Luce



5.3.6 Apri Garage Bluetooth



5.3.7 Spegni Garage Bluetooth



5.4 Scenari

5.4.1

| | |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Caso d'uso | Apri Garage |
| ID | ApriGarage |
| Attori primari | Utente |
| Attori secondari | ESP32 |
| Precondizioni | L'utente si è registrato o ha fatto l'accesso |
| Sequenza degli eventi principali | <ol style="list-style-type: none">1. Il caso d'uso inizia quando l'utente preme sul pulsante "Apri"2. Il sistema invoca il codice per l'attivazione dei motori Servo3. Fisicamente la porta del garage si apre |
| Postcondizioni | La porta del garage è aperta |
| Sequenza degli eventi alternativa | <ol style="list-style-type: none">3A. Se l'invocazione del codice fallisce apparirà un pop-up con la scritta "Riprova" |

5.4.2

| | |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Caso d'uso | Stato Garage |
| ID | StatoGarage |
| Attori primari | Utente |
| Attori secondari | ESP32 |
| Precondizioni | L'utente si è registrato o ha fatto l'accesso |
| Sequenza degli eventi principali | <ol style="list-style-type: none">1. Il caso d'uso inizia quando l'utente preme sul pulsante "Visualizza Stato Garage"2. Il sistema invoca il codice per verificare lo stato attuale dei motori Servo3. Viene stampato sulla pagina web lo stato attuale del garage |
| Postcondizioni | Sulla pagina web appare lo stato del garage |
| Sequenza degli eventi alternativa | <ol style="list-style-type: none">3A. Se l'invocazione del codice fallisce apparirà come stato del garage "Non disponibile" |

5.4.3

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Caso d'uso | Chiudi Garage |
| ID | ChiudiGarage |
| Attori primari | Utente |
| Attori secondari | ESP32 |
| Precondizioni | L'utente si è registrato o ha fatto l'accesso |
| Sequenza degli eventi principali | <ol style="list-style-type: none"> 1. Il caso d'uso inizia quando l'utente preme sul pulsante "Chiudi" 2. Il sistema invoca il codice per l'attivazione dei motori Servo 3. Fisicamente la porta del garage si chiude |
| Postcondizioni | La porta del garage è chiusa |
| Sequenza degli eventi alternativa | <ol style="list-style-type: none"> 3A. Se l'invocazione del codice fallisce apparirà un pop-up con la scritta "Riprova" |

5.4.4

| | |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Caso d'uso | Visualizza attività |
| ID | VisualizzaAttività |
| Attori primari | Utente |
| Attori secondari | ESP32 |
| Precondizioni | L'utente si è registrato o ha fatto l'accesso |
| Sequenza degli eventi principali | <ol style="list-style-type: none"> 1. Il caso d'uso inizia quando l'utente preme sul pulsante "Visualizza Attività" 2. Il sistema invoca il codice per verificare le credenziali dell'utente 3. Verrà stampata sulla pagina web una tabella contenente tutte le informazioni relative all'attività |
| Postcondizioni | La porta del garage è chiusa |
| Sequenza degli eventi alternativa | <ol style="list-style-type: none"> 2A. Se le informazioni dell'utente sono già presenti in cache il sistema raccoglierà username e password da quest'ultima 2B. Se le informazioni dell'utente sono errate apparirà il messaggio di errore "Login fallito" |

5.4.5

| | |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Caso d'uso | ApriGarageBluetooth |
| ID | ApriGarageBluetooth |
| Attori primari | Utente |
| Attori secondari | ESP32 |
| Precondizioni | L'utente si è collegato tramite bluetooth all'esp32 |
| Sequenza degli eventi principali | <ol style="list-style-type: none"> 1. L'utente digita la password sul terminale bluetooth 2. Il sistema invoca il codice per verificare le credenziali dell'utente 3. Fisicamente la porta del garage si apre |
| Postcondizioni | La porta del garage è aperta |
| Sequenza degli eventi alternativa | <ol style="list-style-type: none"> 2A. Se le informazioni dell'utente non sono corrette verrà stampato a video "password errata" |

5.4.6

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Caso d'uso | ChiudiGarageBluetooth |
| ID | ChiudiGarageBluetooth |
| Attori primari | Utente |
| Attori secondari | ESP32 |
| Precondizioni | L'utente si è collegato tramite bluetooth all'esp32 |
| Sequenza degli eventi principali | <ol style="list-style-type: none"> 1. L'utente digita la password sul terminale bluetooth 2. Il sistema invoca il codice per verificare le credenziali dell'utente 3. Fisicamente la porta del garage si chiude |
| Postcondizioni | La porta del garage è chiusa |
| Sequenza degli eventi alternativa | <ol style="list-style-type: none"> 2A. Se le informazioni dell'utente non sono corrette verrà stampato a video "password errata" |

6 Codice ARDUINO

Il codice è suddiviso in due programmi principali ognuno caricato sulla propria esp32, più avanti sono riportati i codici Arduino mentre tutto il codice è reperibile sul nostro github: [Link](#)

6.1 Librerie utilizzate

Con riferimento alla prima esp (servo motori)

- ESP32Servo.h: Libreria per controllare i servo motori con la ESP32.
- WiFi.h: Permette la connessione a una rete WiFi.
- HTTPClient.h: Permette di fare richieste HTTP.
- WebServer.h: Permette di creare un server web sulla ESP32.

Con riferimento alla seconda esp (luce e sensore DHT):

- WiFi.h
- HTTPClient.h
- WebServer.h
- DHT.h: Libreria per interfacciarsi con il sensore DHT11 (temperatura e umidità).

6.2 Codice prima esp32

```
1 #include <ESP32Servo.h>
2 #include <WiFi.h>
3 #include <HTTPClient.h>
4 #include <WebServer.h>
5
6 // Credenziali WiFi
7 const char* ssid_router = "Hotspot 374";
8 const char* password_router = "123456dK";
9
10 // Indirizzo IP del server PHP
11 const char* serverName = "http://192.168.212.223"; // Sostituisci con l'IP del tuo
    server
12
13 // Pin dei servo motori
14 #define SERVO1_PIN 15
15 #define SERVO2_PIN 2
16
17 Servo servo1;
18 Servo servo2;
19 int posVal = 0;
20 int posVal2 = 0;
21
22 // Server web sulla porta 80
23 WebServer server(80);
24
25 void setup() {
26     Serial.begin(115200);
27
28     // Configurazione dei servo motori
29     servo1.setPeriodHertz(50);
30     servo2.setPeriodHertz(50);
31     servo1.attach(SERVO1_PIN, 500, 2500);
32     servo2.attach(SERVO2_PIN, 500, 2500);
33
34     // Connessione alla rete WiFi
35     WiFi.begin(ssid_router, password_router);
36     Serial.println("Connessione WiFi in corso...");
37     while (WiFi.status() != WL_CONNECTED) {
38         delay(500);
39         Serial.print(".");
40     }
41     Serial.println("\nConnesso alla rete WiFi");
42     Serial.print("Indirizzo IP: ");
43     Serial.println(WiFi.localIP());
```

```

45 // Configurazione delle route del server
server.on("/apri-garage", handleApriGarage);
47 server.on("/chiudi-garage", handleChiudiGarage);

49 // Avvio del server
server.begin();
51 Serial.println("Server HTTP avviato");
}

53 void loop() {
55     server.handleClient();
57 }

void handleApriGarage() {
59     while (posVal < 100 && posVal2 < 100) {
        servo1.write(100 - posVal); // Muove il servo1 in senso opposto
61         servo2.write(posVal2);
        posVal++;
63         posVal2++;
        delay(20);
65     }
    inviaStatoGarage("aperta");
67     server.send(200, "text/plain", "Garage aperto");
}

69 void handleChiudiGarage() {
71     while (posVal > 0 && posVal2 > 0) {
        servo1.write(100 - posVal); // Muove il servo1 in senso opposto
73         servo2.write(posVal2);
        posVal--;
75         posVal2--;
        delay(20);
77     }
    inviaStatoGarage("chiusa");
79     server.send(200, "text/plain", "Garage chiuso");
}

81 void inviaStatoGarage(const char* stato) {
83     if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
85         String url = String(serverName) + "/aggiorna-stato-garage.php?stato_garage=" +
            stato;
        http.begin(url.c_str());
87         int httpResponseCode = http.GET();
        if (httpResponseCode > 0) {
89             String response = http.getString();
            Serial.println(httpResponseCode);
            Serial.println(response);
91         } else {
93             Serial.print("Errore nella richiesta HTTP: ");
            Serial.println(httpResponseCode);
95         }
        http.end();
97     }
}

```

6.3 Codice seconda esp32

```

#include <WiFi.h>
2 #include <HTTPClient.h>

4

6
#include <WebServer.h>
8

10
#include "DHT.h"
12
#define DHTPIN 13

```

```

14 #define DHTTYPE DHT11
16 #define LUCE_PIN 15

18 const char* ssid = "Hotspot 374";
const char* password = "123456dK";
20 const char* serverName = "http://192.168.212.223/save_data.php"; // IP del computer

22 WebServer server(80);
DHT dht(DHTPIN, DHTTYPE);

24 String stato_luce = "OFF"; // Dichiarazione della variabile stato_luce

26 void setup() {
28     Serial.begin(115200);
    setup_wifi();

30     server.on("/luce", handleLuce);
32     server.on("/dati_sensori", handleDatiSensori);
    server.on("/stato", handleStato);

34     server.begin();

36     pinMode(LUCE_PIN, OUTPUT);
38     dht.begin();
}

40 void setup_wifi() {
42     delay(10);
    Serial.println();
44     Serial.print("Connecting to ");
    Serial.println(ssid);

46     WiFi.begin(ssid, password);

48     while (WiFi.status() != WL_CONNECTED) {
50         delay(500);
        Serial.print(".");
52     }

54     Serial.println("");
    Serial.println("WiFi connected");
56     Serial.println("IP address: ");
    Serial.println(WiFi.localIP()); // Stampa l'indirizzo IP della ESP32
58 }

60 void handleLuce() {
    String cmd = server.arg("cmd");
62     if (cmd == "accendi") {
        digitalWrite(LUCE_PIN, HIGH);
64         stato_luce = "ON";
        server.send(200, "text/plain", "Luce accesa");
66     } else if (cmd == "spegni") {
        digitalWrite(LUCE_PIN, LOW);
68         stato_luce = "OFF";
        server.send(200, "text/plain", "Luce spenta");
70     } else {
        server.send(400, "text/plain", "Comando non valido");
72     }
}

74 void handleDatiSensori() {
    float h = dht.readHumidity();
    float t = dht.readTemperature();
76     if (isnan(h) || isnan(t)) {
        server.send(500, "text/plain", "Errore nella lettura del sensore DHT");
78         return;
    }
82     String data = "Temperatura: " + String(t) + " C , Umidit : " + String(h) + "%";
    server.send(200, "text/plain", data);
84     sendDataToServer(t, h);
}

86 void handleStato() {
88     String stato = "{\"luce\":\"" + stato_luce + "\"}";
    server.send(200, "application/json", stato);

```

```

90 }

92 void sendDataToServer(float temperatura, float umidita) {
93     if (WiFi.status() == WL_CONNECTED) {
94         HTTPClient http;
95         String serverPath = String(serverName) + "?temperatura=" + String(temperatura)
96         + "&umidita=" + String(umidita);

97         http.begin(serverPath.c_str());
98         int httpResponseCode = http.GET();

100         if (httpResponseCode > 0) {
101             String response = http.getString();
102             Serial.println(httpResponseCode);
103             Serial.println(response);
104         } else {
105             Serial.print("Errore nella richiesta HTTP: ");
106             Serial.println(httpResponseCode);
107         }
108         http.end();
109     } else {
110         Serial.println("WiFi Disconnesso");
111     }
112 }

114 void loop() {
115     server.handleClient();
116 }

```

6.4 Codice per apertura/chiusura Bluetooth

```

#include <ESP32Servo.h>
2 #include "BluetoothSerial.h"
#include "string.h"

4 BluetoothSerial SerialBT;
6 char buffer[20]; //creo un buffer di caratteri che mi servirà per leggere da
    serial al bluetooth
static int count = 0;
8 Servo myservo; // creo un oggetto servo per controllare il primo servo
Servo myservo2; // creo un oggetto servo per controllare il primo servo
10 int servoPin = 15; // definisco il pin del primo servo
int servoPin2 = 2; //definisco il pin del secondo servo

12 int posVal = 0; // variable che uso per conservare la posizione del servo
14 int posVal2 = 0;

16 void setup() {
17     myservo.setPeriodHertz(50); // periodo standard 50 hz servo
18     myservo2.setPeriodHertz(50); // periodo standard 50 hz servo
    // pinMode(BUZZER_PIN, OUTPUT);
20     myservo.attach(servoPin, 500, 2500); // collego il servo al rispettivo pin e
        stabilisco col secondo la pulsazione in microsecondi 544 corrisponde al minimo
        ossia 0 gradi
    myservo2.attach(servoPin2, 500, 2500); // con il terzo parametro invece ho il
        massimo ossia 180 gradi 2500
22     SerialBT.begin("ESP32test"); //Bluetooth device name
    Serial.begin(115200); //Baudrate
24     Serial.println("\nConnessione bluetooth disponibile");
}

26 void loop() {
28     while(SerialBT.available())
    {
30         buffer[count] = SerialBT.read();
        count++;
32     }
    if(count>0){
34         Serial.print(buffer);
        if(strncmp(buffer, "open", 4)==0){//se la stringa inviata tramite bluetooth
            open allora procedo a muovere i servo

```

```

36     while(posVal<100 && posVal2<100) { // vado da 0 gradi fino a 100 gradi con
un passo di 1 grado alla volta con entrambi i motori servo
38     myservo.write(posVal);
myservo2.write(posVal2);
posVal++;
40     posVal2++;
delay(20); // prima di aumentare aspetto 20ms per far
raggiungere la posizione ad entrambi i servo
42 }

44 }
if(strncmp(buffer,"close",5)==0){ //se la stringa inviata tramite bluetooth
close allora procedo a muovere i servo
46     while(posVal > 0 && posVal2>0){ // ritorno da 100 gradi a 0 gradi con un
passo di 1 grado alla volta
myservo.write(posVal);
48     myservo2.write(posVal2);
posVal--;
50     posVal2--;
delay(20);
52 }

54 }
count=0;
56     memset(buffer,0,20); //uso memset per svuotare il buffer
}

58 }

```

7 Conclusione

Degli eventuali sviluppi futuri del progetto possono essere:

- **Crittografia delle comunicazioni:** Utilizzare HTTPS anziché HTTP per garantire che i dati trasmessi siano crittografati.
- **Automazione:** Implementare logiche di automazione basate su condizioni specifiche, ad esempio accendere la luce automaticamente quando l'umidità supera una certa soglia.
- **Risparmio energetico:** Tramite l'implementazione di Mosquitto.
- **App mobile:** Sviluppare un'app mobile dedicata per un accesso e controllo più intuitivi.