Relazione progetto finale
Prototipizzazione con Arduino

Cremonini Davide - VR471360

Univr - A.A. 2022/23

Indice

Chapter 1

Introduzione

1.1 Descrizione progetto

Il progetto implementa un sistema di autenticazione di un parcheggio pensato per controllare l'accesso di dipendenti di un ufficio o studenti di un università. L'autenticazione avviene tramite lettura di codici QR associati ad ogni utente. Oltre a permettere l'autenticazione, il sistema registra i tempi di utilizzo del parcheggio di ogni utente e rende possibili consultare queste statistiche da un pannello di controllo. Tramite quest'ultimo è possibile anche ricevere notifiche di un eventuale richiesta di aiuto proveniente dal totem del parcheggio, per poter essere informati e provvedere ad assistere l'utente.

Attraverso la presenza di un display e vari led, è presente anche un interfaccia per l'utente del parcheggio che potrà, utilizzando dei pulsanti, accedere a diverse funzionalità:

- **Ingresso**: l'utente potrà impostare il circuito in modalità ingresso per permettere di autenticare la propria utenza e far alzare la sbarra corretta ed entrare all'interno del parcheggio;
- **Uscita**: l'utente potrà impostare il circuito in modalità uscita per permettere di inserire nel sistema quali tra gli utenti sta lasciando il parcheggio e far alzare la sbarra corretta;
- **Richiesta di aiuto**: l'utente potrà, in caso di necessità, aprire una richiesta di assistenza, la quale potrà essere chiusa direttamente dall'utente al totem, oppure tramite il pannello di controllo gestito da un operatore.

Questa interfaccia presenta informazioni che ne facilitano l'utilizzo informando l'utente di come operare, di ciò che sta succedendo o dei tempi di attesa necessari.

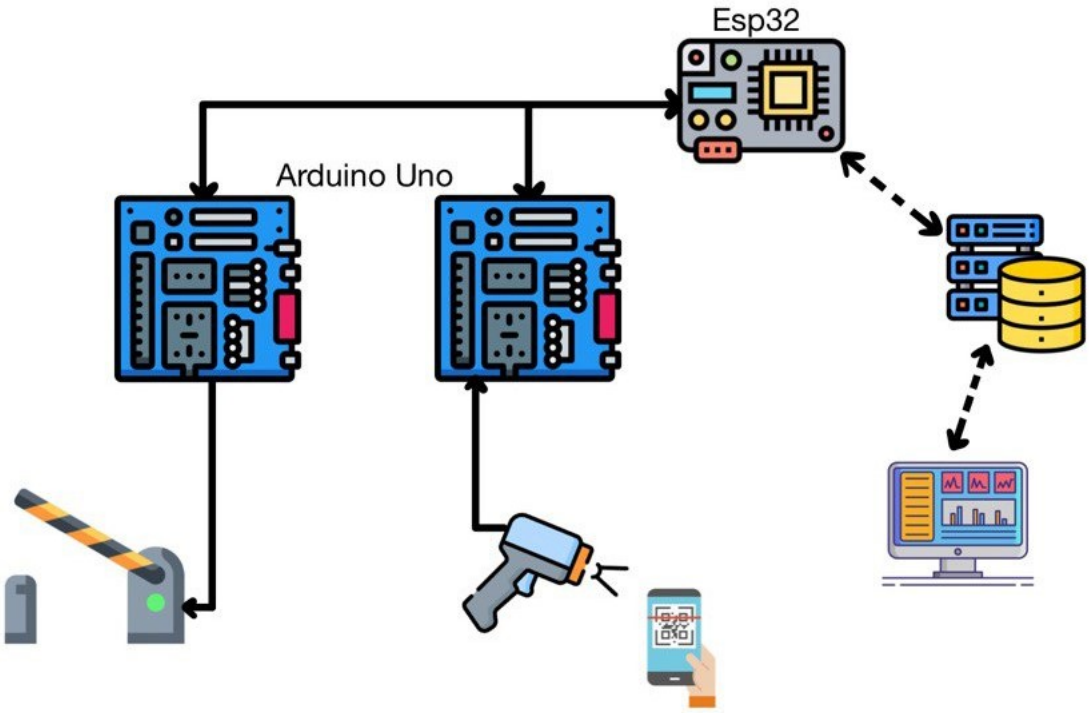
Per la realizzazione ho utilizzato 2 schede Arduino Uno e una scheda esp32, collegati in seriale (RX-TX). L'esp oltre a permettere il collegamento ad internet, si occupa di scambiare i dati con un realtime database creato con firebase.
Il resto del circuito è così diviso:

- **Arduino Uno A**: fornisce un'interfaccia all'utente e permette la lettura del qr code, inviando questo dato e la modalità selezionata dall'utente in seriale, attendendo come risposta un riscontro per poi informare l'utente tramite led e display;
- **Arduino Uno B**: svolge l'operazione di apertura e chiusura delle due sbarre relative all'ingresso e all'uscita, in base ai dati provenienti dall'esp. Questa parte di circuito controlla anche un led di errore/avviso che completa l'interfaccia per l'utente.

Infine, ho creato un pannello di controllo implementando un interfaccia web in python, sfruttando anche flask e plotly.

1.2 Schema progetto

Di seguito uno schema ad alto livello che rappresenta il funzionamento dell'intero progetto.



1.3 Componenti utilizzati

Componenti	Quantità
Arduino Uno	2
Esp32	2
Sesore GM65	1
Schermo Oled	1
Pulsanti	2
Led	4
Resistenze	4
Micro servomotori	2

Chapter 2

Realizzazione

2.1 Processo lavorativo

Ho iniziato con la realizzazione, tra hardware e software, del sistema che permetteva di selezionare la modalità di ingresso e uscita, con un pulsante e due led. Ho poi proseguito aggiungendo un secondo pulsante implementando una terza modalità, ovvero quella della richiesta di aiuto. Tutto questo, testando fisicamente il circuito e caricando le modifiche allo sketch di partenza sull'unica scheda arduino che utilizzavo inizialmente.

Ho continuato poi, aggiungendo un display per mostrare a video varie informazioni per l'utente, e uno scanner per leggere i QR code. Questa fase non è stata per nulla rapida, è stato necessario consultare materiale informativo per apprendere l'utilizzo dei due componenti. Per lo scanner ho utilizzato una software serial in quanto, sapevo che la seriale predisposta da arduino mi sarebbe servita per la connessione con l'esp, e preferivo dedicarla a quella comunicazione. Sia per lo scanner che per il display, ho avuto problemi sul lato connessione al sistema, che facevano sfarfallare in certe occasioni il monitor o rendevano inutilizzabile lo scanner. Sono poi arrivato a una soluzione semistabile per permettere il corretto funzionamento del prototipo.

Dopo questa fase, sono passato all'inserimento del modulo esp32 nel circuito, collegandolo in seriale (RX-TX) con l'arduino e creando un nuovo sketch da caricare in esso. Anche questa fase non è stata da meno in quanto ad ostacoli. Ho iniziato creando la parte di codice che permetteva di connettere il modulo a internet per poi provare a realizzare una connessione con ThingSpeak. A questo punto ho realizzato che non faceva al mio caso e ho abbandonato questa strada per poi passare a Firebase, il quale mi permetteva di fare ciò che avevo pensato inizialmente.

Avevo fin da subito l'idea che i pin di un arduino non mi sarebbero bastati, e infatti è poi arrivato il momento in cui ho aggiunto al circuito un'altra scheda arduino, collegata anch'essa in seriale. A questa nuova scheda ho connesso due servomotori e ho creato un nuovo sketch apposito per controllarli in base a ciò che la scheda riceve dalla seriale. Successivamente ho anche aggiunto un led per permettere di gestire e informare l'utente di eventuali errori provenienti dall'esp.

Mi sono poi spostato alla creazione dell'interfaccia web sfruttando python e flask, creando il collegamento con il database di firebase a cui faceva riferimento anche l'esp. Utilizzando plotly ho reso possibile la realizzazione di un grafico che riporta le statistiche sull'utilizzo del parcheggio.

2.2 Codice

Di seguito sono riportati delle porzioni delle sezioni più salienti del codice.

Esp32

Alcune delle Funzioni:

```
//funzione per controllare se l'id che leggo da arduino corrisponde all'id che leggo nel db
int check_id(){
  Serial.println("Leggo da Arduino: "+id_a);

  path="/Utenti/"+user+"/";
  if(Firebase.RTDB.getString(&fbdo, path)){
    if(fbdo.dataType()=="string"){
      id=fbdo.stringData();
      Serial.println("Leggo da DB: "+id); //print a video per debug
      if(id==id_a){
        return 1;
      }else{
        return 0;
      }
    }
  }else{
    Serial.println("Failed! " + fbdo.errorReason());
    return 0;
  }
}
```

```
//funzione per settare a true la richiesta di aiuto nel db quando viene letto da arduino
void send_help_req(){
  if(Firebase.RTDB.setBool(&fbdo, "/Richiesta_aiuto/Stato/", true)){
    Serial.println("Richiesta di aiuto inviata con successo!");
  }else{
    Serial.println("Errore! " + fbdo.errorReason());
  }
}

//funzione per settare a false la richiesta di aiuto nel db quando viene letto da arduino
void reset_help_req(){
  if(Firebase.RTDB.setBool(&fbdo, "/Richiesta_aiuto/Stato/", false)){
    Serial.println("Stato richiesta di aiuto resettato correttamente!");
  }else{
    Serial.println("Errore! " + fbdo.errorReason());
  }
}
```

```
//funzione per salvare il current time in ingresso di un utente
void salva_time_in(){
  path="/Utenti_in/"+user+"/";
  if(Firebase.RTDB.setInt(&fbdo, path, millis())){
    Serial.println("Tempo ingresso inserito correttamente!");
  }else{
    Serial.println("Errore! " + fbdo.errorReason());
  }
}

//funzione per salvare il current time in uscita di un utente
void salva_time_out(){
  path="/Utenti_out/"+user+"/";
  if(Firebase.RTDB.setInt(&fbdo, path, millis())){
    Serial.println("Tempo ingresso inserito correttamente!");
  }else{
    Serial.println("Errore! " + fbdo.errorReason());
  }
}
```

```
//funzione per controllare lo stato della richiesta di aiuto dal db
int check_help_resolved(){
  Serial.println("Controllo help status");  //print a video per debug
  if(Firebase.RTDB.getBool(&fbdo, "/Richiesta_aiuto/Stato/")){
    if(fbdo.dataType()=="boolean"){
      help_status=fbdo.boolData();
      Serial.print("Leggo da DB: "); //print a video per debug
      Serial.println(help_status); //print a video per debug

      if(help_status){ //se true significa che non è ancora stata risolta
        return 0;
      }else{ //se false significa che è stata risolta e returno 1 al main
        return 1;
      }
    }
  }else{
    Serial.println("Errore! " + fbdo.errorReason());
    return 0;
  }
}
```

Loop:

```
  //ad ogni ciclo controllo se il sistema è in attesa di sblocco
  if(attesa_req){
    if(check_help_resolved()==1){ //controllo se è stato sbloccato
      Serial2.print("4"); //informo arduino dell'aiuto concluso

      attesa_req=false; //resetto lo stato di attesa
      Serial.print("Attesa terminata");
    }
  }

  //ad ogni ciclo salvo ciò che leggo da seriale
  str_from_arduino=Serial2.readString();
  //Serial.println("leggo from arduino: '"+str_from_arduino+"'");
  str_from_arduino.trim(); //rimuovo eventuali \r e \n finali

  //effettuo un controllo su ciò che ho letto così evito di entrare se è uguale o vuoto
  if(str_from_arduino!=in_prec && str_from_arduino!=""){

    //salvo ciò che ho letto per non rientrare nell'if
    in_prec=str_from_arduino;

    //scompongo la stringa formattata
    mode=str_from_arduino.charAt(0);
    user=str_from_arduino.substring(1,4);
    id_a=str_from_arduino.substring(4);
```

```
    //qui parte il controllo sulla modalità letta da seriale selezionata dall'utente
    if(mode=='h'){ //significa che da arduino arriva una richiesta di aiuto

      Serial.println("Richiesta di aiuto ricevuta da Arduino!\n");
      send_help_req(); //modifico il valore nel db
      attesa_req=true; //porto a true lo stato di attesa per effettuare il controllo sul db controllando se viene risolta

    }else if(mode=='r'){ //significa che da arduino la richiesta è stata chiusa

      reset_help_req(); //modifico il valore nel db
      attesa_req=false; //resetto lo stato di attesa
    }else if(mode=='i'){ //significa che un utente vuole entrare
      if(check_id()){ //controllo se l'id corrisponde

        salva_time_in(); //salvo current time ingresso

        Serial.println("Corrispondono! Benvenuto");
        Serial2.print("1"); //scrivo ad arduino di aprire la sbarra 1
      }else{
        Serial.println("Non corrispondono!");
        Serial2.print("3"); //segnalo ad arduino errore
      }
    }else if(mode=='o'){ //significa che un utente vuole uscire
      if(check_id()){ //controllo se l'id corrisponde
```

Arduino A

Alcune delle Funzioni:

```
67  //funzione per leggere e salvare i codici scansionati dal sensore gm65
68  void lettura_code(){
69    // while (!(ss.available())) {
70    //     //do nothing
71    // }
72    // Serial.write(ss.read());
73
74    if(ss.available()){
75      //Serial.write(ss.read()); //scan & print (momentaneo, non va bene)
76
77      String scan=ss.readString();  //leggo ciò che viene scansionato e lo salvo in una stringa
78      scan.trim(); //rimuovo eventuali \r e \n finali
79
80      code=scan.substring(0,8); //salvo i primi 8 caratteri, utili all'identificazione
81      name=scan.substring(8); //il resto è il nome dell'utente
82    }
83  }
```

```
85  //funzione per inviare stringa in seriale a esp32
86  void invio_dati(){
87    /*prima di scirvere sulla seriale, formatto la stringa nel seguente modo:
88    primo carattere->modalità (i/o/h/r)
89    resto dei caratteri->codice letto
90
91    */
92    if((mode!=' ' && code!="") || mode=='h' || mode=='r'){
93
94      output=mode+code; //formatto la stringa
95      if(output!=output_prec){ //controllo per evitare di spammare dati in seriale sempre uguali
96        Serial.println(output); //scrivo su seriale la stringa formattata che dovrà poi essere interpretata da esp32
97        output_prec=output;
98
99        //resetto i valori che ho inviato
100       code="";
101       mode=' ';
102     }
103   }
104
105 }
```

```
123 //funzione per leggere ack di esp32 da seriale
124 int leggo_risposta(){
125
126   //variabile temporanea che contiene ciò che leggo dalla seriale dall'esp
127   int ck=Serial.read()-48; //-48 perchè viene inviato come stringa
128
129   if(ck==1){ //se vale 1 sta entrando
130     frase1="Identificazione";
131     frase2="avvenuta con";
132     frase3="successo!";
133     frase4="Benvenuto "+name;
134     display_print(frase1,frase2,frase3,frase4);
135     delay(5000);
136     frase1="Sbarra in funzione";
137     frase2="Attendere ...";
138     frase4="";
139     for(int i=10;i>0;i--){
140       frase3=String(i)+" secondi rimasti";
141       display_print(frase1,frase2,frase3,frase4);
142       delay(1000);
143     }
144   }else if(ck==2){ //se vale 1 sta uscendo
```

Loop:

```
211   //leggo pulsanti ogni ciclo
212   if(digitalRead(help_button)==LOW){ //se pulsante premuto, cambio stato
213     help_status=!help_status;
214     delay(200);
215     //change=1; //aggiunto flag per sapere se i pulsanti sono stati cambiati o no
216
217     //aggiungo if per resettare e non tornare a in\out
218     if(help_status==LOW){
219       reset();
220       mode='r'; //richiesta chiusa da pulsante che verrà inviata a esp
221     }else{
222       change=1;
223     }
224
225   }else if(digitalRead(green_button)==LOW){ //se pulsante premuto, cambio stato
226     in_status=!in_status;  //cambio stato tra ingresso e uscita
227     delay(200);
228     change=1; //flag per sapere se i pulsanti sono stati cambiati o no
229   }
```

```
234   //se pulsanti sono stati premuti e qualche stato è cambiato, rieseguo i controlli
235   if(change==1){
236
237     change=0; //una volta entrato nell'if, reimposto il flag a 0
238
239     /*3 diversi stati del circuito a seconda dei pulsanti premuti e degli stati attivi
240     -ingresso
241     -uscita
242     -richiesta di aiuto
243     */
244     if(help_status==HIGH){ //richiesta aiuto
245
246       //modifico stato led
247       digitalWrite(blue,LOW); //spengo tutto il resto
248       digitalWrite(verde,LOW);
249       digitalWrite(giallo,HIGH); //accendo giallo
250
251       //mostro a video lo stato
252       //display_print("Richiesta aiuto");
253       frase1="Richiesta di aiuto";
254       frase2="inviata, attendere.";
255       frase3="Premere nuovamente";
256       frase4="per annullare.";
257
258       //modifica char di stato
259       mode='h';
260
261     }else{
262       //no richiesta, gestisco in/out
263
264       digitalWrite(giallo,LOW); //spengo giallo
265
266       //eseguo controllo su secondo stato
267       if(in_status==HIGH){
```

```
      //eseguo controllo su secondo stato
      if(in_status==HIGH){
        //se in_status è high allora imposto il circuito in modalità in

        //modifico stato led
        digitalWrite(blue,LOW);
        digitalWrite(verde,HIGH);
        //mostro a video lo stato
        frase1="Hai selezionato:";
        frase2="INGRESSO";
        frase3="Scansiona codice";
        frase4="per entrare";
        //modifica char di stato
        mode='i';
      }else{
        //se in_status è low allora imposto il circuito in modalità out

        //modifico stato led
        digitalWrite(blue,HIGH);
        digitalWrite(verde,LOW);
        //mostro a video lo stato
        frase1="Hai selezionato:";
        frase2="USCITA";
        frase3="Scansiona codice";
        frase4="per uscire";
        //modifica char di stato
        mode='o';
      }
      //fine else dell'help
    }
    //fine if del change
  }
  display_print(frase1,frase2,frase3,frase4);
```

```
  //ad ogni ciclo eseguo un controllo sulla richiesta di aiuto(se è attiva o se è stata chiusa)
  if(mode=='h' || mode=='r'){
    invio_dati(); //invio h o r ad esp

  }else{
    lettura_code(); //leggo codice da gm65
    invio_dati(); //invio a esp 32

  }

  //ad ogni ciclo eseguo la lettura da seriale dove troverò un eventuale ack da esp
  rck=leggo_risposta();
  if(rck==1 || rck==2 || rck==3){
    reset(); //resetto il sistema e ricomincio il loop
  }else if(rck==4){
    help_status=LOW; //abbasso lo stato di help
    reset(); //e resetto il sistema

    //informo esp(che cambierà anche il db) e lo preparo a una nuova lettura (così permetto una nuova h)
    mode='r';
    invio_dati();
  }
}
```

Arduino B

Alcune delle Funzioni:

```
//funzione per aprire la sbarra dell' ingresso
void ingresso(){
  //apro
  for(int i=0;i<90;i++){
    servo1.write(i);
    delay(20);
  }
  delay(8000);
  //chiudo
  for(int i=90;i>0;i--){
    servo1.write(i);
    delay(50);
  }
}

//funzione per aprire la sbarra dell'uscita
void uscita(){
  //apro
  for(int i=90;i>0;i--){
    servo2.write(i);
    delay(20);
  }
  delay(8000);
  //chiudo
  for(int i=0;i<90;i++){
    servo2.write(i);
    delay(50);
  }
}
```

Loop:

```
void loop() {
  //ad ogni ciclo leggo ack da seriale che arriva da esp
  in=Serial.read()-48;

  //debug
  if(ao!=in){ //per non printare continuamente
    Serial.print("Leggo da seriale: ");
    Serial.println(in);

    ao=in; //per non printare continuamente
  }

  //ad ogni ciclo eseguo un controllo sull'input letto e
  if(in==1){
    ingresso();
  }else if(in==2){
    uscita();
  }else if(in==3){
    errore();
  }

}
```

Python

Route:

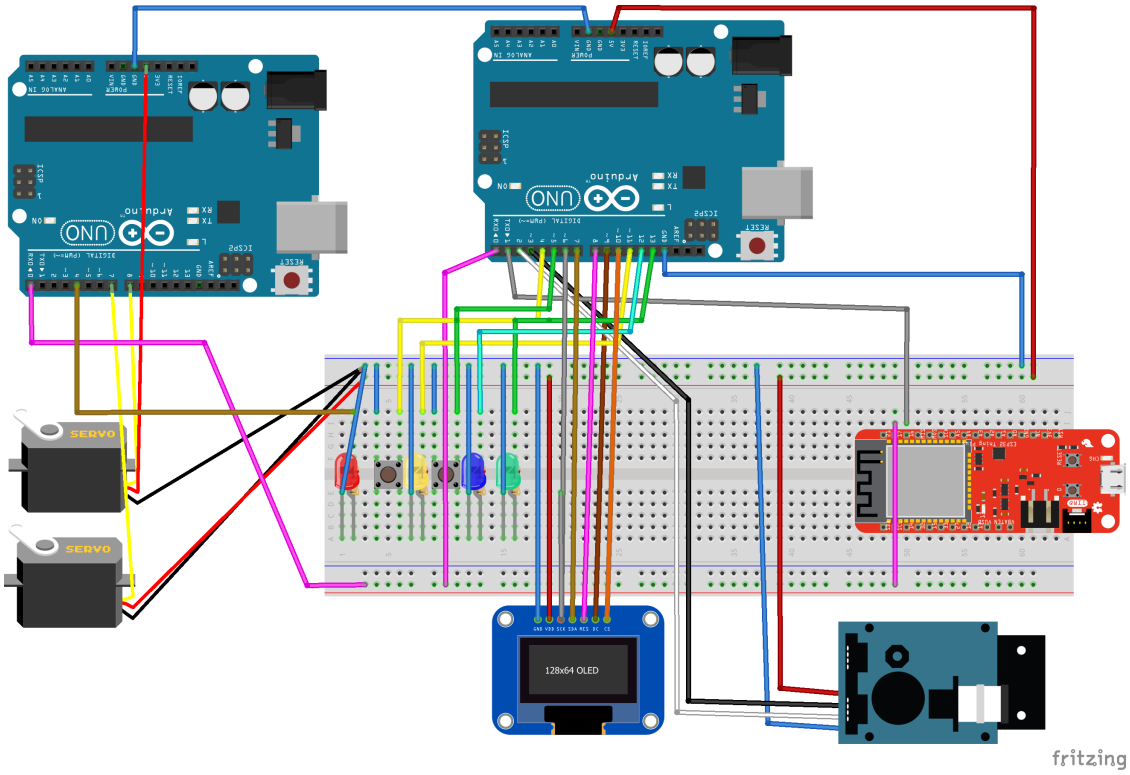
```python
#route per la pagina principale
@app.route('/')
def home():
    ref = db.reference('Richiesta_aiuto/Stato')
    status = ref.get()

    return render_template('index.html', status=status)

#route per modificare lo stato della richiesta
@app.route('/change_status', methods=['POST'])
def change_status():
    ref = db.reference('Richiesta_aiuto/Stato')
    ref.set(not ref.get())
    return redirect(url_for('home'))

#route per controllare lo stato della richiesta nel db, cosi posso far apparire la richiesta solo se true
@app.route('/check_status')
def check_status():
    ref = db.reference('Richiesta_aiuto/Stato')
    status = ref.get()
    return str(status)

#route per la pagina con il grafico
@app.route('/graph')
def graph():
    data = get_data()
    fig = create_plot(data)
    graph = pyo.plot(fig, output_type='div')
    return render_template('graph.html', graph=graph)
```

Funzioni:

```python
#funzione per leggere i dati dal db
def get_data():
    ref = db.reference('Utenti_time')
    data = ref.get()
    return data
```

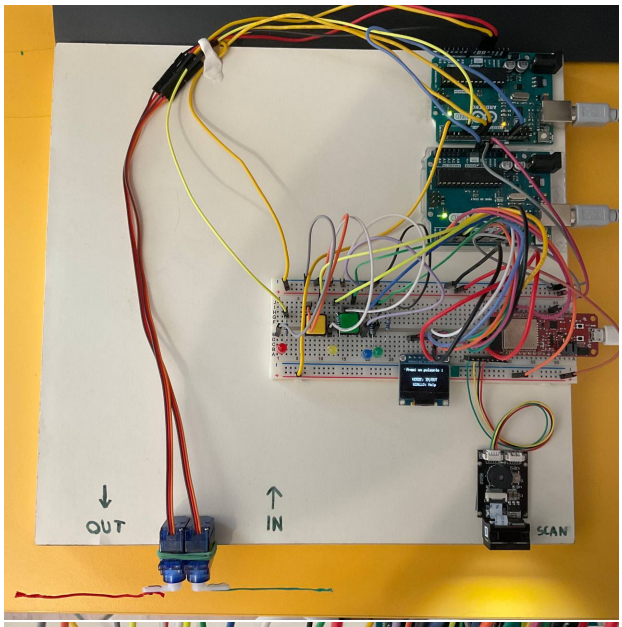
```python
#funzione per creare il grafico con i dati
def create_plot(data):
    fig = go.Figure()
    fig.add_trace(go.Bar(x=list(data.keys()), y=list(data.values()), name="Statistiche parcheggio"))
    fig.update_layout(title='Statistiche sull\'utilizzo', xaxis_title='', yaxis_title='secondi')
    return fig
```

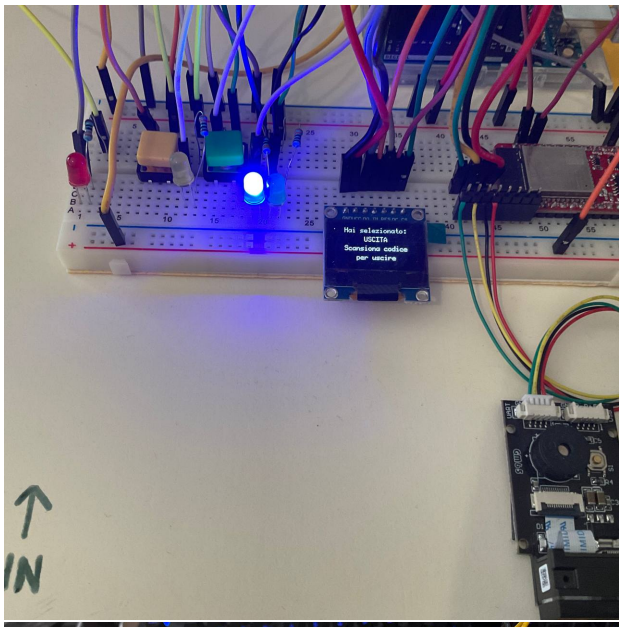
2.3 Schema Circuito

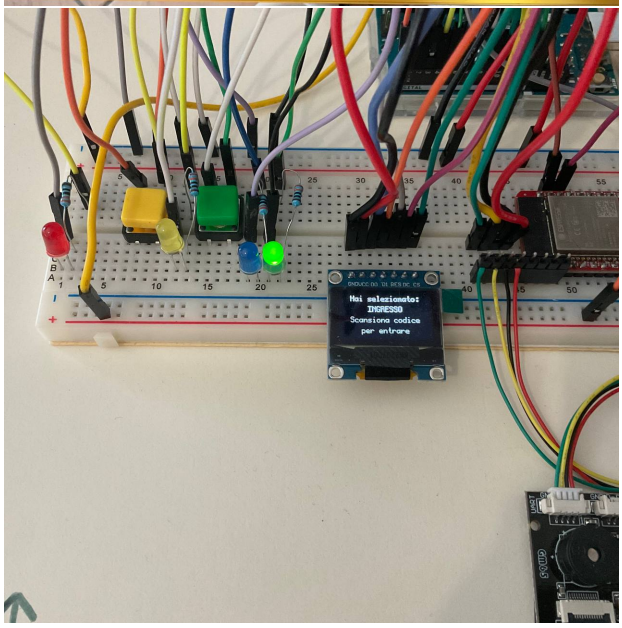


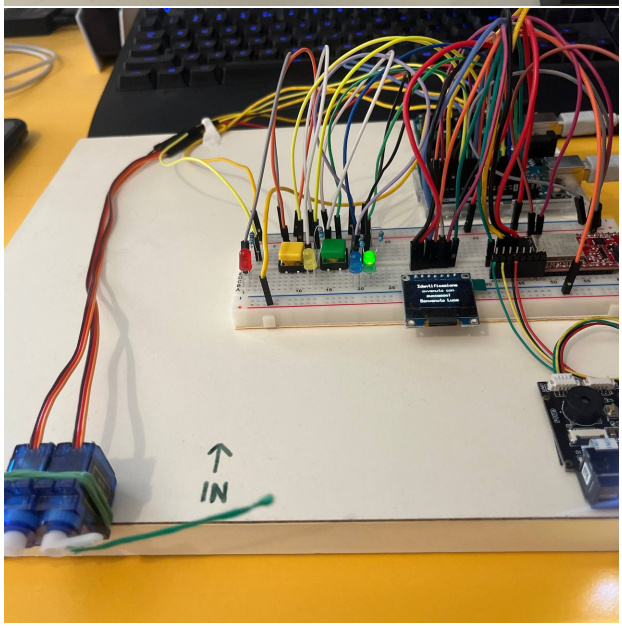
2.4 Implementazione Fisica del circuito

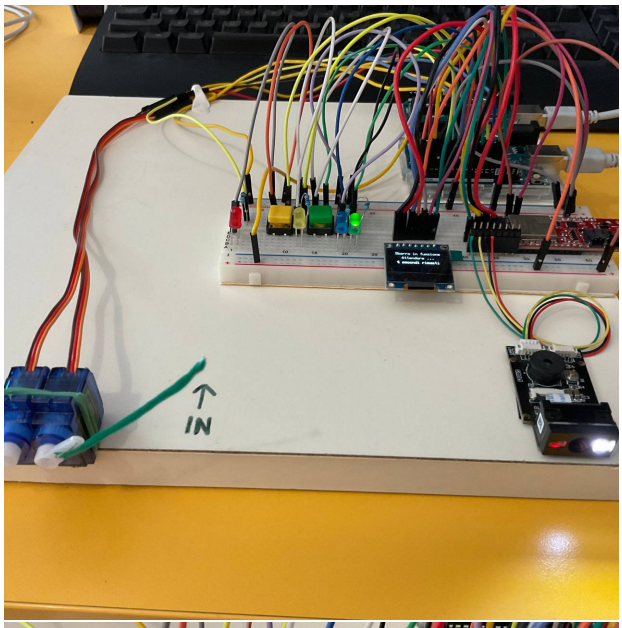
Di seguito una serie di foto del progetto implementato realmente fisicamente con alcuni scatti che evidenziano determinate situazioni.

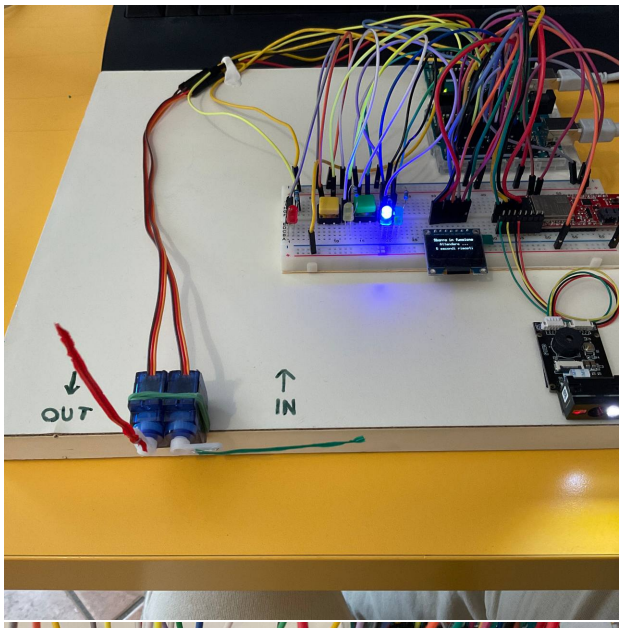


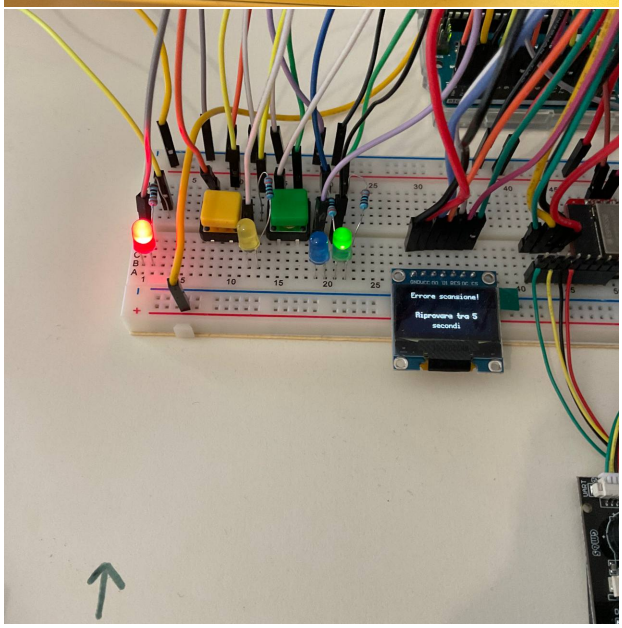


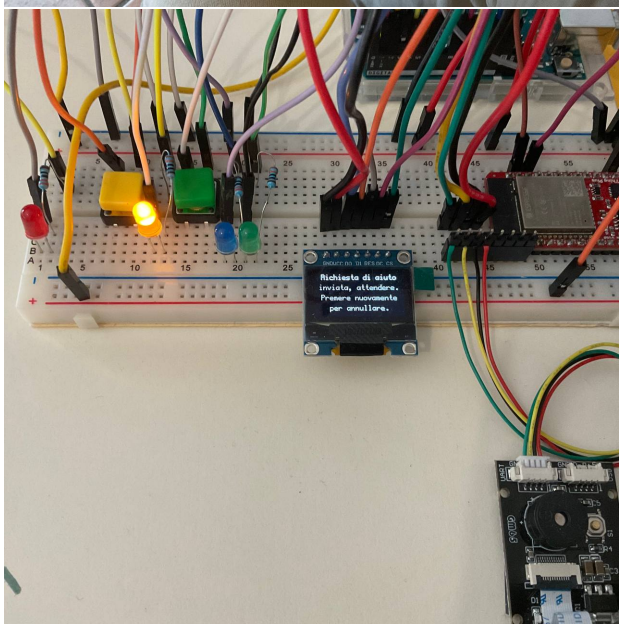





IN


OUT
IN

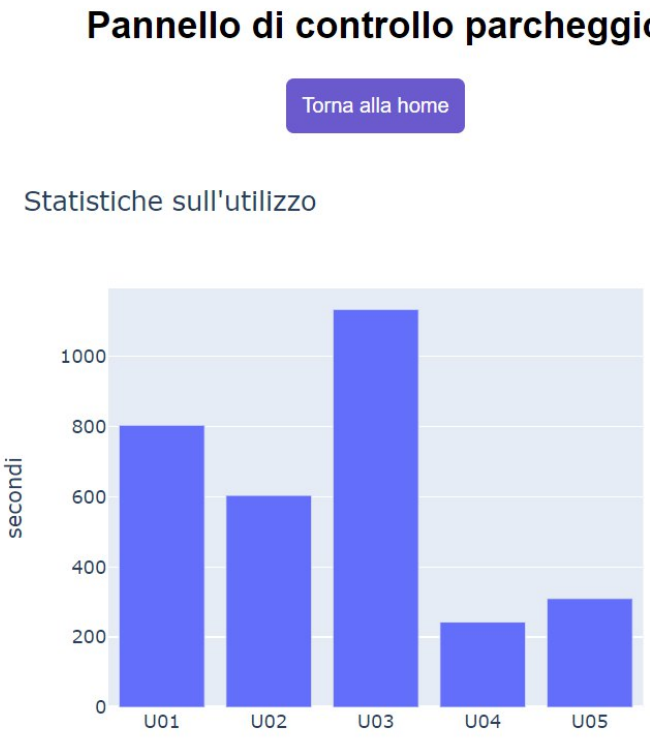

Errore scansione!
Riprovare tra 5
secondi


Richiesta di aiuto
inviata, attendere.
Premere nuovamente
per annullare.

2.5 Interfaccia Web

Pagina principale nella quale puó comparire una notifica che segnala una richiesta di aiuto.

Pagina dove é possibile visualizzare i dati sotto forma di grafici rappresentati con plotly.



Chapter 3

Conclusioni

3.1 Considerazioni finali ed eventuali sviluppi possibili

Durante la realizzazione del progetto ho deciso di modificare l'idea iniziale alcune volte per permettermi di ultimare il progetto, questo dovuto a ostacoli incontrati o mancanza di materiale, ma restano possibili diversi sviluppi che avevo ideato per diramare il progetto.

Per esempio la possibilità di fermare la sbarra se un veicolo si trova sottostante nella fase di chiusura, avevo implementato via software il controllo ma il sensore di prossimità che avevo a disposizione sembrava non funzionare perciò dopo diversi tentativi inutili ho abbandonato l'idea.

Un'altra cosa che avrei voluto implementare è quella di salvare nel database l'orario esatto nel momento di ingresso e uscita degli utenti, questo sarebbe stato possibile tramite un orologio rtc che non possedevo. Ho però ovviato a questo problema tramite l'utilizzo della funzione millis() che, tramite una differenza, riesco a sfruttare per trovare il tempo trascorso all'interno del parcheggio.

Un ultimo possibile sviluppo che vorrei citare è la possibilità di incrementare le analisi statistiche possibili grazie ai dati che il sistema può raccogliere. Io mi sono concentrato sulla parte del circuito e del codice arduino per realizzare il prototipo nel migliore dei modi, implementando la parte dell'interfaccia in maniera riduttiva per mostrare che è possibile controllare il circuito tramite web e visualizzare i dati che vengono salvati nel database.

Infine quindi, consapevole che potrebbe essere realizzato molto di più attorno a questo progetto, mi ritengo soddisfatto del risultato finale e delle capacità che mi ha permesso di sviluppare lavorandoci.

3.2 Bibliografia

Per apprendere l'utilizzo di alcuni componenti, come il sensore gm65, ho utilizzato alcune guide di questo canale youtube:

`https://youtube.com/@zeppelinmaker`

Per l'idea su come utilizzare firebase interfacciato con esp32 mi sono ispirato a questo progetto, specialmente la parte dove con il database controllano lo stato di un led, per tramutare il suo utilizzo a ciò che più faceva comodo a me:

`https://youtu.be/aO92B-K4TnQ`

Per mostrare a video sul display oled ho fatto uso della libreria u8g, consultando la documentazione presente su github:

`https://github.com/olikraus/u8glib/wiki/userreference`

Per la realizzazione dell'interfaccia avevo gia qualche conoscenza di flask ma ho utilizzato, come anche per plotly, diverse guide e forum ora difficili da trovare e citare tutti. Tra i tanti mi ha aiutato a risolvere alcune necessità stackoverflow, che mi è stato utile anche per altri problemi sul lato arduino, per il quale ho utilizzato anche il forum ufficiale.

`https://stackoverflow.com/`

`https://forum.arduino.cc/`