



**Relazione progetto finale
Prototipizzazione con Arduino**

Cremonini Davide - VR471360

Indice

1	Introduzione	3
1.1	Descrizione progetto	3
1.2	Schema progetto	4
1.3	Componenti utilizzati	4
2	Realizzazione	5
2.1	Processo lavorativo	5
2.2	Codice	6
2.3	Schema Circuito	13
2.4	Implementazione Fisica del circuito	14
2.5	Intefraccia Web	16
3	Conclusioni	17
3.1	Considerazioni finali ed eventuali sviluppi possibili	17
3.2	Bibliografia	18

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.

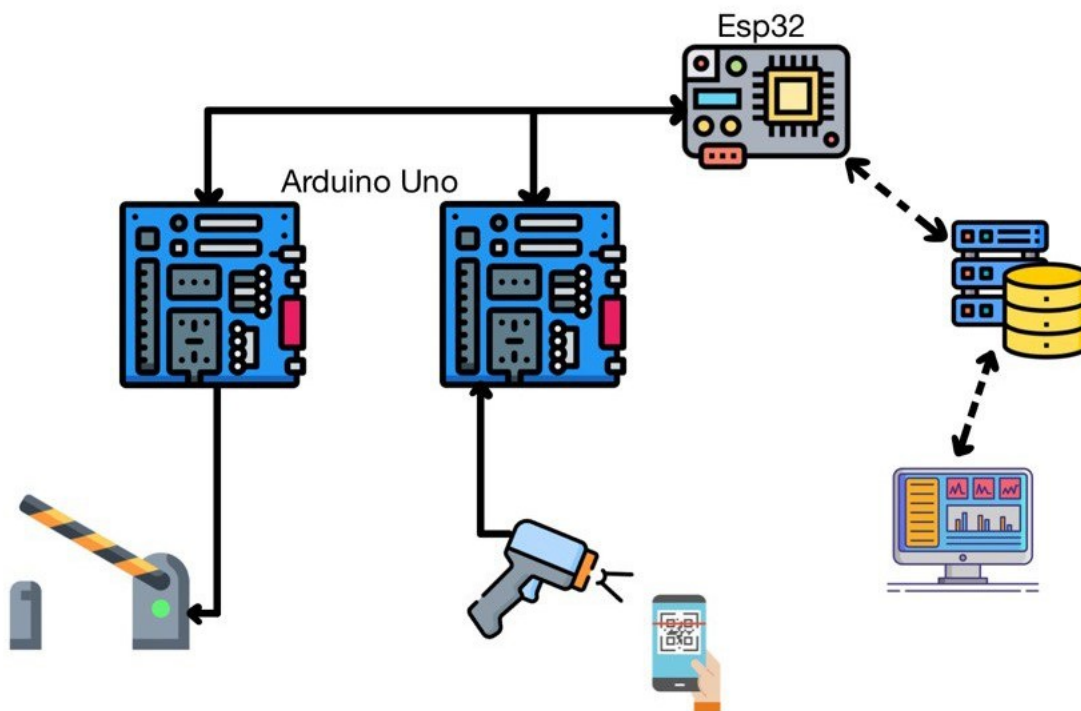
Il resto del circuito è così diviso:

- **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:

```
34 //funzione per controllare se l'id che leggo da arduino corrisponde all'id che leggo nel db
35 int check_id(){
36     Serial.println("Leggo da Arduino: "+id_a);
37
38     path="/Utenti/"+user+"/";
39     if(Firebase.RTDB.getString(&fbdo, path)){
40         if(fbdo.dataType()=="string"){
41             id=fbdo.stringData();
42             Serial.println("Leggo da DB: "+id); //print a video per debug
43             if(id==id_a){
44                 return 1;
45             }else{
46                 return 0;
47             }
48         }
49     }else{
50         Serial.println("Failed! " + fbdo.errorReason());
51         return 0;
52     }
53 }
```

```
55 //funzione per settare a true la richiesta di aiuto nel db quando viene letto da arduino
56 void send_help_req(){
57     if(Firebase.RTDB.setBool(&fbdo, "/Richiesta_aiuto/Stato/", true)){
58         Serial.println("Richiesta di aiuto inviata con successo!");
59     }else{
60         Serial.println("Errore! " + fbdo.errorReason());
61     }
62 }
63
64 //funzione per settare a false la richiesta di aiuto nel db quando viene letto da arduino
65 void reset_help_req(){
66     if(Firebase.RTDB.setBool(&fbdo, "/Richiesta_aiuto/Stato/", false)){
67         Serial.println("Stato richiesta di aiuto resettato correttamente!");
68     }else{
69         Serial.println("Errore! " + fbdo.errorReason());
70     }
71 }
```

```
73 //funzione per salvare il current time in ingresso di un utente
74 void salva_time_in(){
75     path="/Utenti_in/"+user+"/";
76     if(Firebase.RTDB.setInt(&fbdo, path, millis())){
77         Serial.println("Tempo ingresso inserito correttamente!");
78     }else{
79         Serial.println("Errore! " + fbdo.errorReason());
80     }
81 }
82
83 //funzione per salvare il current time in uscita di un utente
84 void salva_time_out(){
85     path="/Utenti_out/"+user+"/";
86     if(Firebase.RTDB.setInt(&fbdo, path, millis())){
87         Serial.println("Tempo ingresso inserito correttamente!");
88     }else{
89         Serial.println("Errore! " + fbdo.errorReason());
90     }
91 }
```

```

161 //funzione per controllare lo stato della richiesta di aiuto dal db
162 int check_help_resolved(){
163     Serial.println("Controllo help status"); //print a video per debug
164     if(Firebase.RTDB.getBool(&fbdo, "/Richiesta_aiuto/Stato/")){
165         if(fbdo.dataType()=="boolean"){
166             help_status=fbdo.boolData();
167             Serial.print("Leggo da DB: "); //print a video per debug
168             Serial.println(help_status); //print a video per debug
169
170             if(help_status){ //se true significa che non è ancora stata risolta
171                 return 0;
172             }else{ //se false significa che è stata risolta e ritorno 1 al main
173                 return 1;
174             }
175         }
176     }else{
177         Serial.println("Errore! " + fbdo.errorReason());
178         return 0;
179     }
180 }

```

Loop:

```

226 //ad ogni ciclo controllo se il sistema è in attesa di sblocco
227 if(attesa_req){
228     if(check_help_resolved()==1){ //controllo se è stato sbloccato
229         Serial2.print("4"); //informo arduino dell'aiuto concluso
230
231         attesa_req=false; //resetto lo stato di attesa
232         Serial.print("Attesa terminata");
233     }
234 }
235
236 //ad ogni ciclo salvo ciò che leggo da seriale
237 str_from_arduino=Serial2.readString();
238 //Serial.println("leggo from arduino: '"+str_from_arduino+"'");
239 str_from_arduino.trim(); //rimuovo eventuali \r e \n finali
240
241 //effettuo un controllo su ciò che ho letto così evito di entrare se è uguale o vuoto
242 if(str_from_arduino!=in_prec && str_from_arduino!=""){
243
244     //salvo ciò che ho letto per non rientrare nell'if
245     in_prec=str_from_arduino;
246
247     //scompongo la stringa formattata
248     mode=str_from_arduino.charAt(0);
249     user=str_from_arduino.substring(1,4);
250     id_a=str_from_arduino.substring(4);

```

```

259 //qui parte il controllo sulla modalità letta da seriale selezionata dall'utente
260 if(mode=='h'){ //significa che da arduino arriva una richiesta di aiuto
261
262     Serial.println("Richiesta di aiuto ricevuta da Arduino!\n");
263     send_help_req(); //modifico il valore nel db
264     attesa_req=true; //porto a true lo stato di attesa per effettuare il controllo sul db controllando se viene risolta
265
266 }else if(mode=='r'){ //significa che da arduino la richiesta è stata chiusa
267
268     reset_help_req(); //modifico il valore nel db
269     attesa_req=false; //resetto lo stato di attesa
270 }else if(mode=='i'){ //significa che un utente vuole entrare
271     if(check_id()){ //controllo se l'id corrisponde
272
273         salva_time_in(); //salvo current time ingresso
274
275         Serial.println("Corrispondono! Benvenuto");
276         Serial2.print("1"); //scrivo ad arduino di aprire la sbarra 1
277     }else{
278         Serial.println("Non corrispondono!");
279         Serial2.print("3"); //segnalo ad arduino errore
280     }
281 }else if(mode=='o'){ //significa che un utente vuole uscire
282     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 scrivere sulla seriale, formato 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){
```

```

266 //eseguo controllo su secondo stato
267 if(in_status==HIGH){
268     //se in_status è high allora imposto il circuito in modalità in
269
270     //modifico stato led
271     digitalWrite(blue,LOW);
272     digitalWrite(verde,HIGH);
273     //mostro a video lo stato
274     frase1="Hai selezionato:";
275     frase2="INGRESSO";
276     frase3="Scansiona codice";
277     frase4="per entrare";
278     //modifica char di stato
279     mode='i';
280 }else{
281     //se in_status è low allora imposto il circuito in modalità out
282
283     //modifico stato led
284     digitalWrite(blue,HIGH);
285     digitalWrite(verde,LOW);
286     //mostro a video lo stato
287     frase1="Hai selezionato:";
288     frase2="USCITA";
289     frase3="Scansiona codice";
290     frase4="per uscire";
291     //modifica char di stato
292     mode='o';
293 }
294 //fine else dell'help
295 }
296 //fine if del change
297 }
298 display_print(frase1,frase2,frase3,frase4);

```

```

300 //ad ogni ciclo eseguo un controllo sulla richiesta di aiuto(se è attiva o se è stata chiusa)
301 if(mode=='h' || mode=='r'){
302     invio_dati(); //invio h o r ad esp
303 }
304 }else{
305     lettura_code(); //leggo codice da gm65
306     invio_dati(); //invio a esp 32
307 }
308 }
309
310 //ad ogni ciclo eseguo la lettura da seriale dove troverò un eventuale ack da esp
311 rck=leggo_risposta();
312 if(rck==1 || rck==2 || rck==3){
313     reset(); //resetto il sistema e ricomincio il loop
314 }else if(rck==4){
315     help_status=LOW; //abbasso lo stato di help
316     reset(); //e resetto il sistema
317
318     //informo esp(che cambierà anche il db) e lo preparo a una nuova lettura (così permetto una nuova h)
319     mode='r';
320     invio_dati();
321 }
322 }

```

Arduino B

Alcune delle Funzioni:

```
20 //funzione per aprire la sbarra dell' ingresso
21 void ingresso(){
22     //apro
23     for(int i=0;i<90;i++){
24         servo1.write(i);
25         delay(20);
26     }
27     delay(8000);
28     //chiudo
29     for(int i=90;i>0;i--){
30         servo1.write(i);
31         delay(50);
32     }
33 }
34
35 //funzione per aprire la sbarra dell'uscita
36 void uscita(){
37     //apro
38     for(int i=90;i>0;i--){
39         servo2.write(i);
40         delay(20);
41     }
42     delay(8000);
43     //chiudo
44     for(int i=0;i<90;i++){
45         servo2.write(i);
46         delay(50);
47     }
48 }
```

Loop:

```
82 void loop() {
83     //ad ogni ciclo leggo ack da seriale che arriva da esp
84     in=Serial.read()-48;
85
86     //debug
87     if(ao!=in){ //per non printare continuamente
88         Serial.print("Leggo da seriale: ");
89         Serial.println(in);
90
91         ao=in; //per non printare continuamente
92     }
93
94     //ad ogni ciclo eseguo un controllo sull'input letto e
95     if(in==1){
96         ingresso();
97     }else if(in==2){
98         uscita();
99     }else if(in==3){
100         errore();
101     }
102 }
103 }
```

Python

Route:

```
#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)

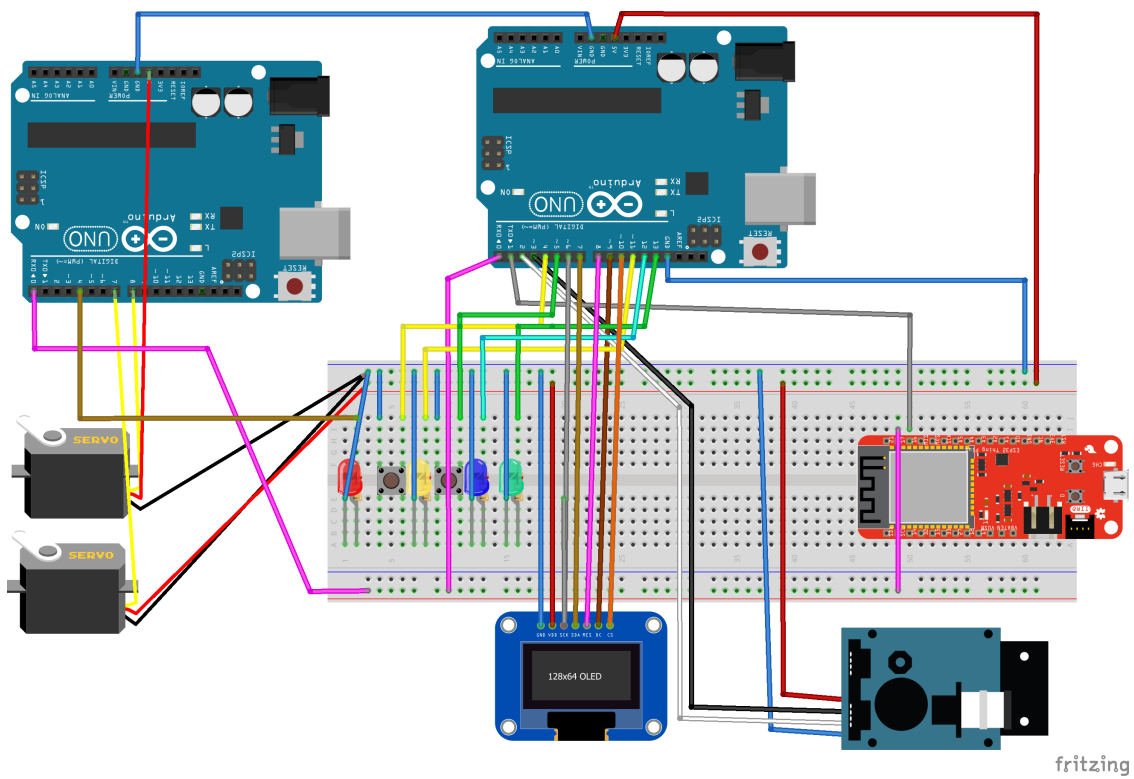
#route per la pagina con il grafico
@app.route('/graph')
def graph():
    data = get_data()
    fig = create_plot(data)
    graph = pyo.plot(fig, output_type='div')
    return render_template('graph.html', graph=graph)
```

Funzioni:

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

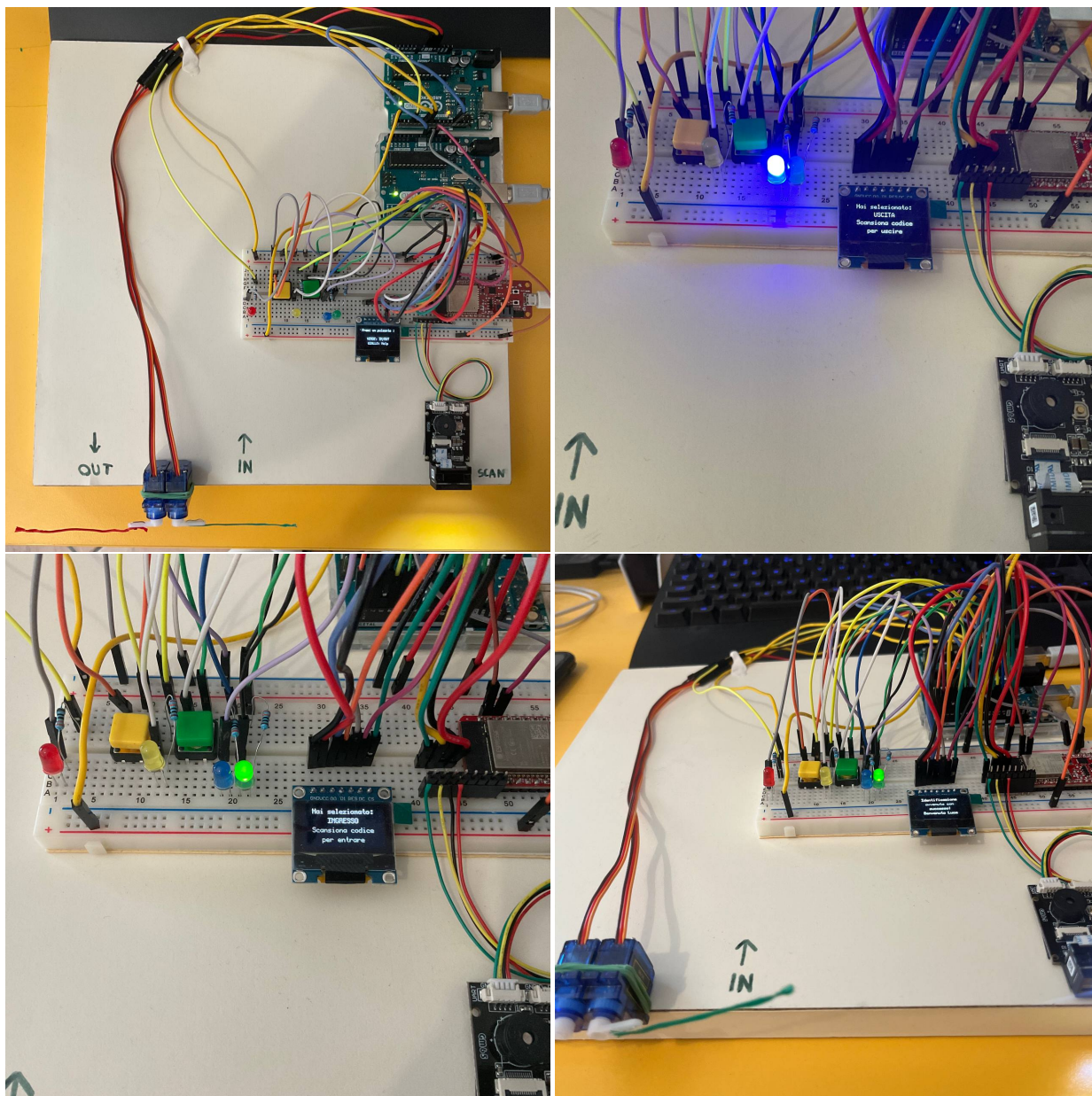
```
#funzione per creare il grafico con i dati
def create_plot(data):
    fig = go.Figure()
    fig.add_trace(go.Bar(x=list(data.keys()), y=list(data.values()), name="Statistiche parcheggio"))
    fig.update_layout(title='Statistiche sull\'utilizzo', xaxis_title='', yaxis_title='secondi')
    return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

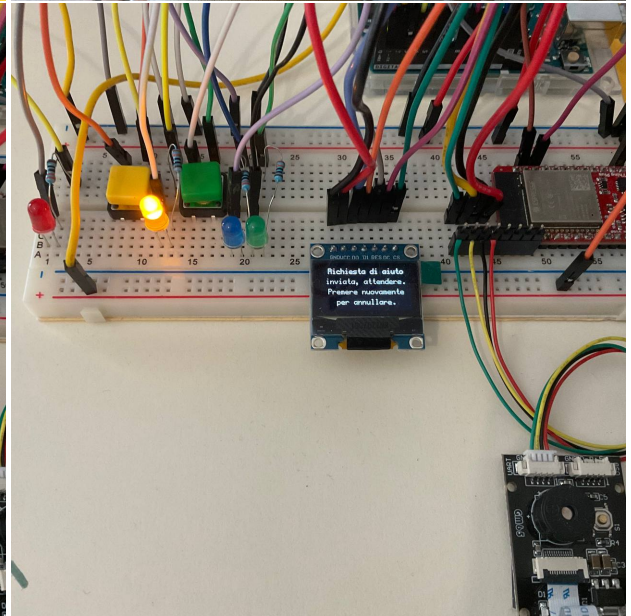
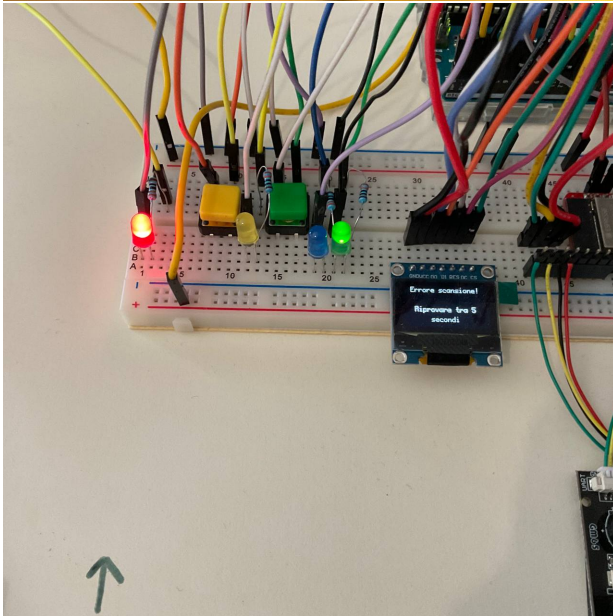
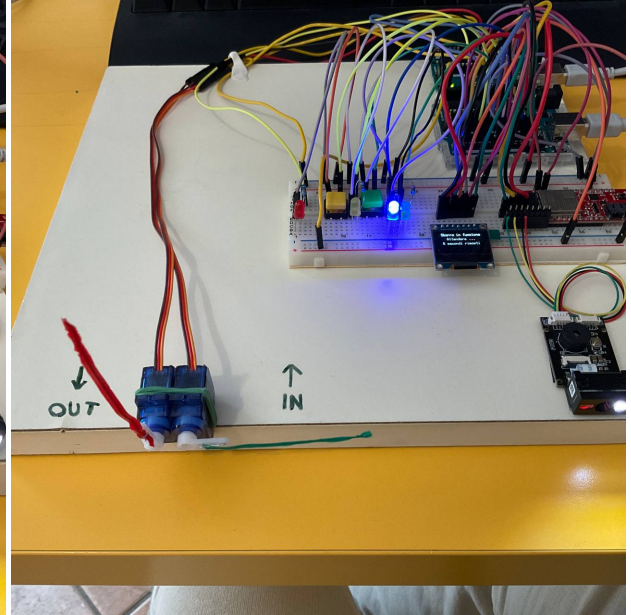
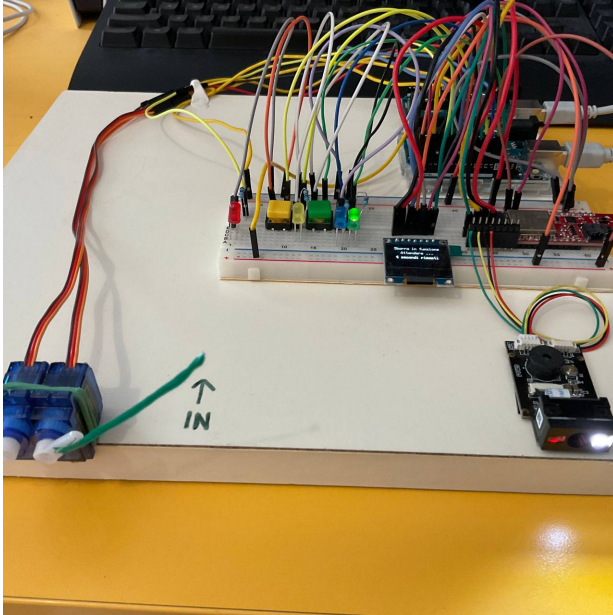
2.3 Schema Circuito



2.4 Implementazione Fisica del circuito

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





2.5 Intefraccia Web

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

Pannello di controllo parcheggio

Visualizza Statistiche

Attenzione!
Richiesta di aiuto in arrivo!

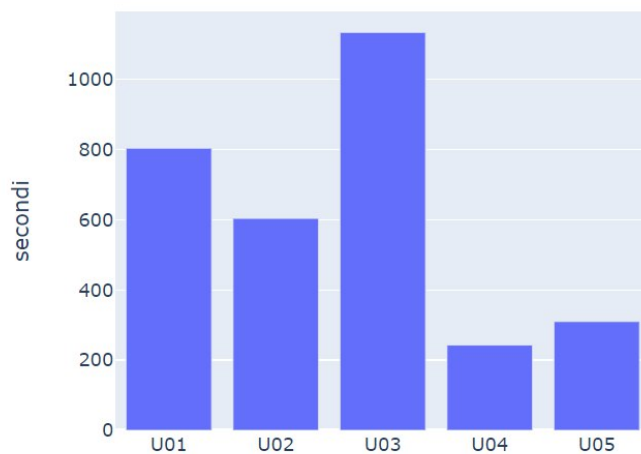
Chiudi Richiesta

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

Pannello di controllo parcheggio

Torna alla home

Statistiche sull'utilizzo



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/a092B-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 già 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/>