Università
degli Studi
della Campania
Luigi Vanvitelli

Università di degli Studi della Campania Luigi Vanvitelli - Dipartimento di Ingegneria

Laurea Triennale in Ingegneria Elettronica e Informatica

Laboratorio di Sviluppo di Applicazioni per IoT
***a.a.* 2023-2024**

Introduzione al Corso

Docente: Carlo Mazzocca
e-mail: carlo.mazzocca@unibo.it

Corso

- **Lezioni di laboratorio**: 24 ore
- **Lingua**: Italiano

- **Obiettivi**: Il laboratorio si propone di fornire conoscenza teorico-pratica degli elementi utili per la progettazione e implementazione di applicazioni per per l'IoT. Alla fine del corso gli studenti saranno in grado di progettare e realizzare reti di oggetti intelligenti connessi a Internet, e a sviluppare applicazioni web-oriented per l'accesso ai servizi e ai dati forniti degli oggetti

Calendario

- **Inizio corso: 21.02.2024**
- **Termine corso: 15.05.2024**
- **Mercoledì – 14:00-16:00**

Il calendario può essere soggetto a variazioni

Controllare avvisi!

Esame

Progetto di gruppo con report + presentazione

Le date d'esame saranno definite in seguito

Argomenti del Corso

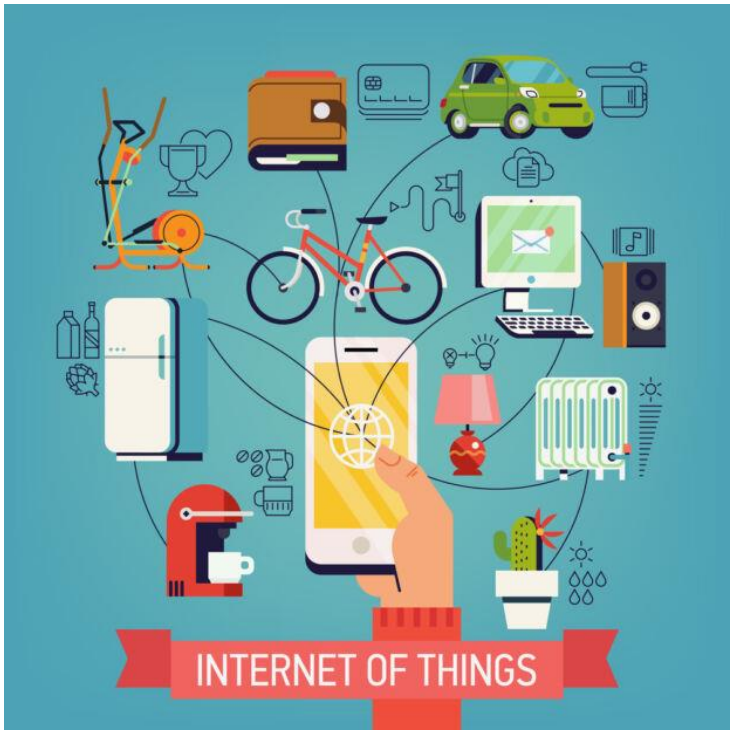
1. Internet of Things
2. Pardigmi Cloud, Edge e Fog
3. Tecnologie per lo sviluppo di applicazioni IoT
4. Principali protocolli
5. Cenni modellazione
6. Reti di sensori
7. Sviluppo di applicazioni IoT

Cosa Imparerai

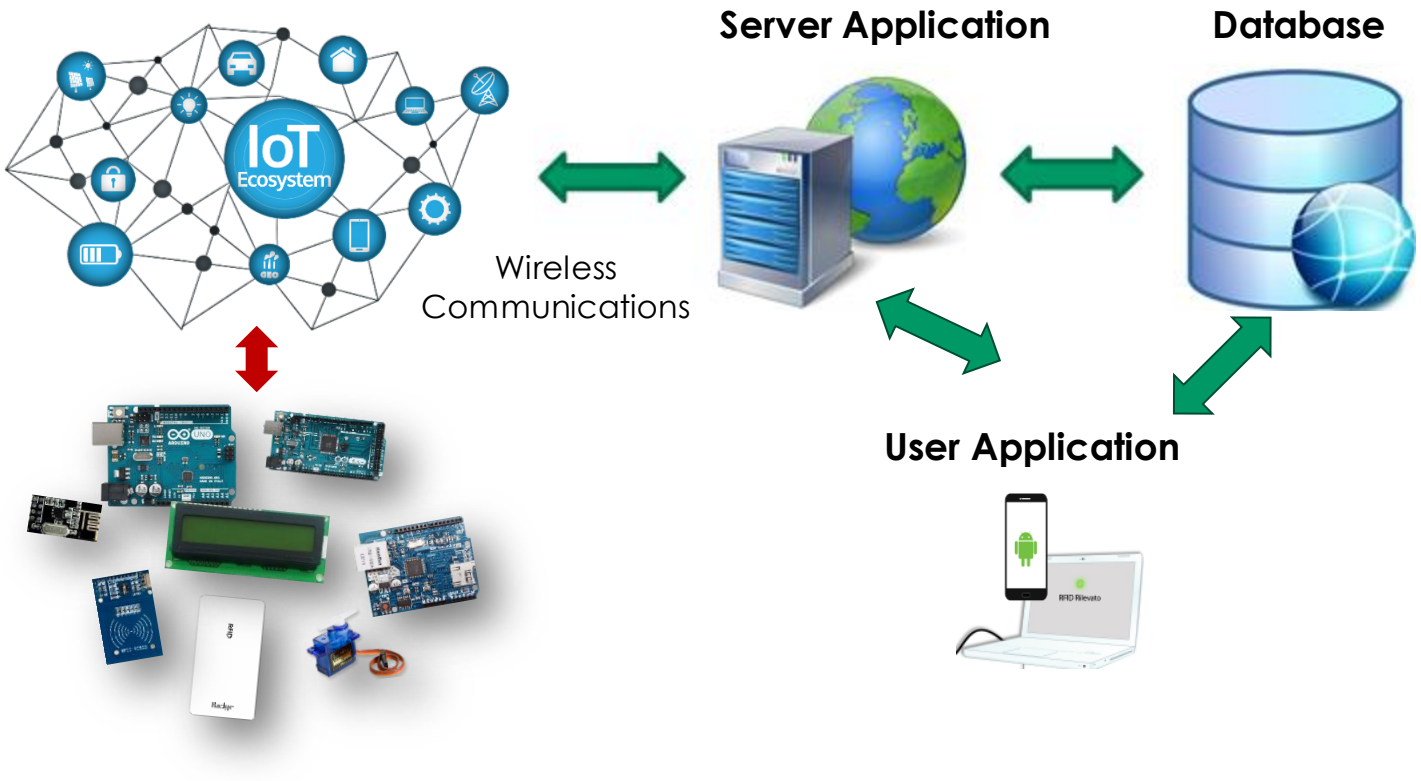
1. Concetti di base IoT
2. Piattaforme e protocolli
3. Programmare dispositivi IoT
4. Integrazione hardware e software

Internet of Things (IoT)

*"The **Internet of Things** is a global network of uniquely addressable interconnected objects based on standard communication protocols"*

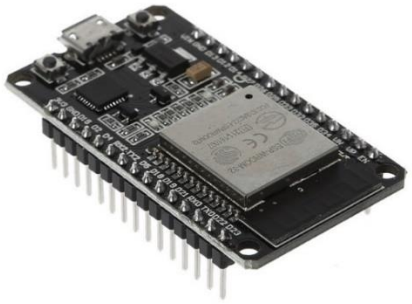


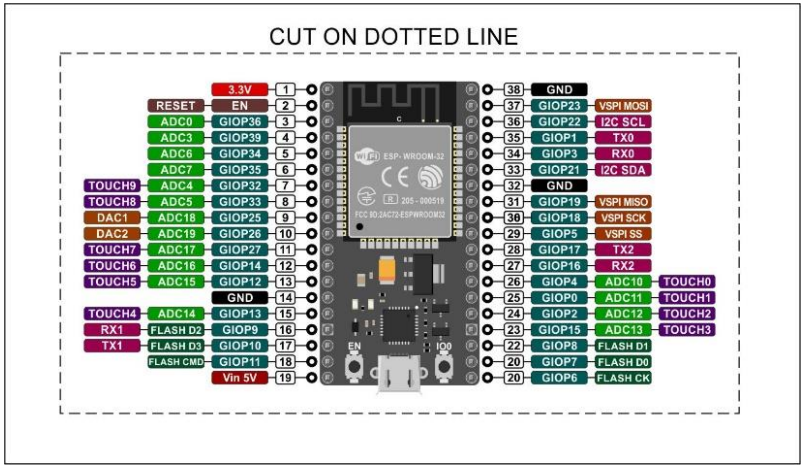
Architettura Applicazione IoT



ESP32

- Processore: Xtensa dual-core (o single-core) 32-bit LX6 microprocessore a 160 o 240 MHz and fino a 600 DMIPS
- Memoria: SRAM 520 Kb
- Basso consumo energetico
- Connettività: Wi-Fi 802-11 b/g/n e Bluetooth v4.2 BR/EDR and BLE

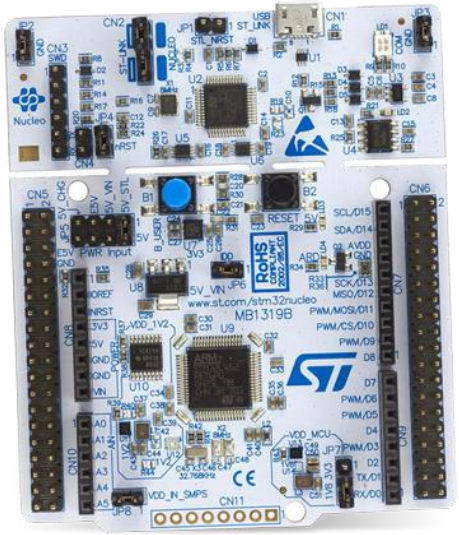


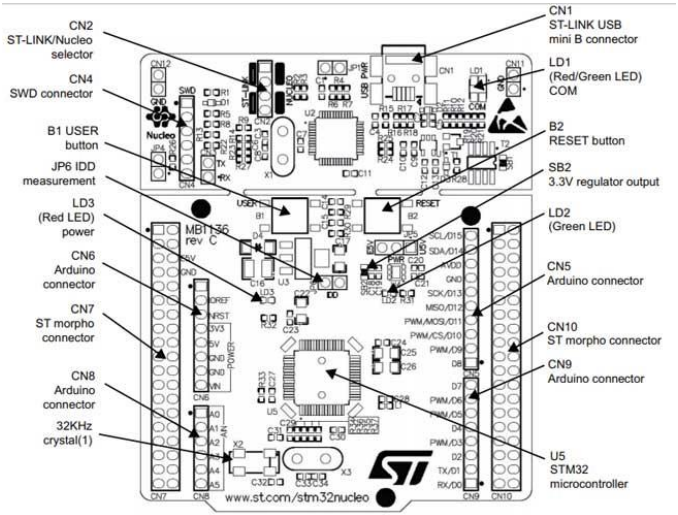


Link Amazon

STM32 Nucleo-64

- Processore: ARM 32-bit Cortex-M4 CPU
- Memoria: Memoria flash 512 Kb e SRAM 96 Kb
- Basso consumo energetico
- 12 interfacce di comunicazione (USARTs, I2C, ecc)
- 81 porte di Input/Output con supporto alle interruzioni



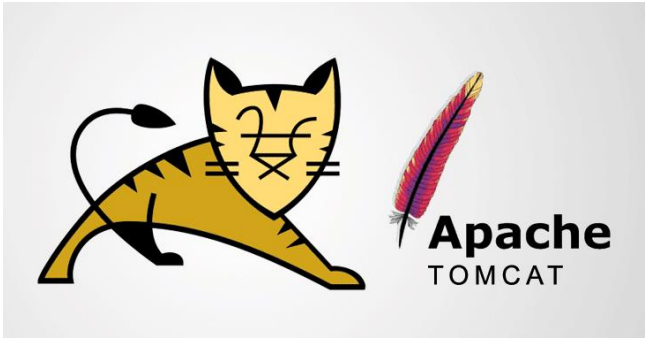


https://www.st.com/resource/en/datasheet/stm32f401re.pdf

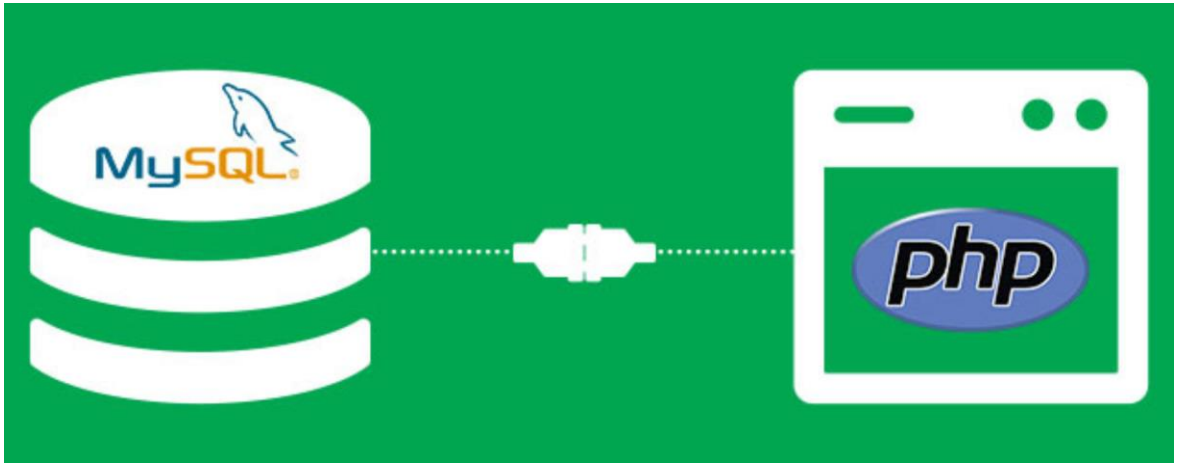
Arduino IDE


ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED,
AND SUPPORTED BY ARDUINO.CC AND
THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS
OF ARDUINO.CC on arduino.cc/credits
Sto avviando...

Applicazioni Server



Database



Cuccia automatizzata

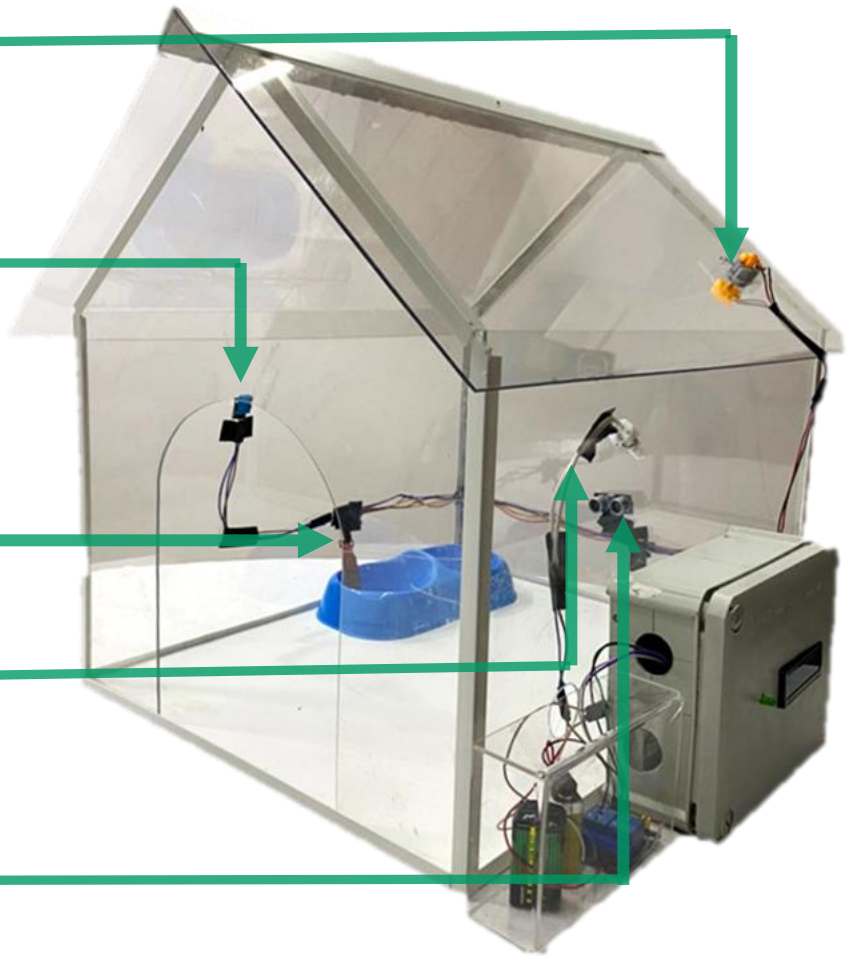
VENTOLA

SENSORE DI TEMPERATURA

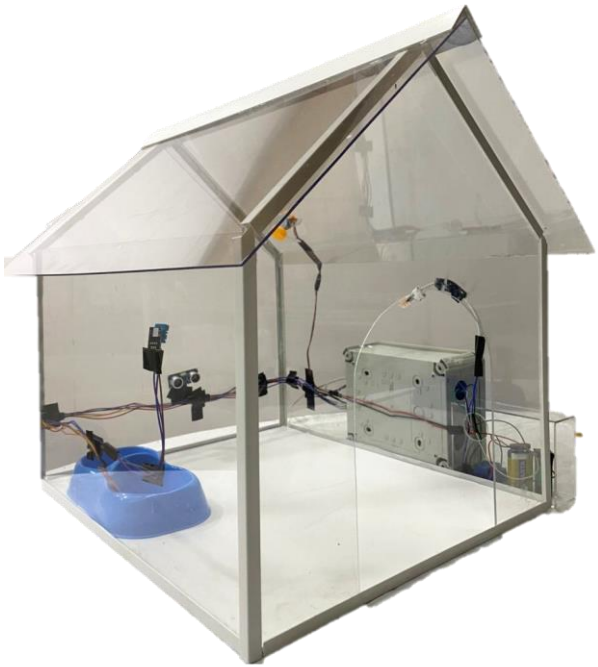
SENSORE DI LIVELLO ACQUA

LAMPADA CALDA

SENSORE DI PRESENZA

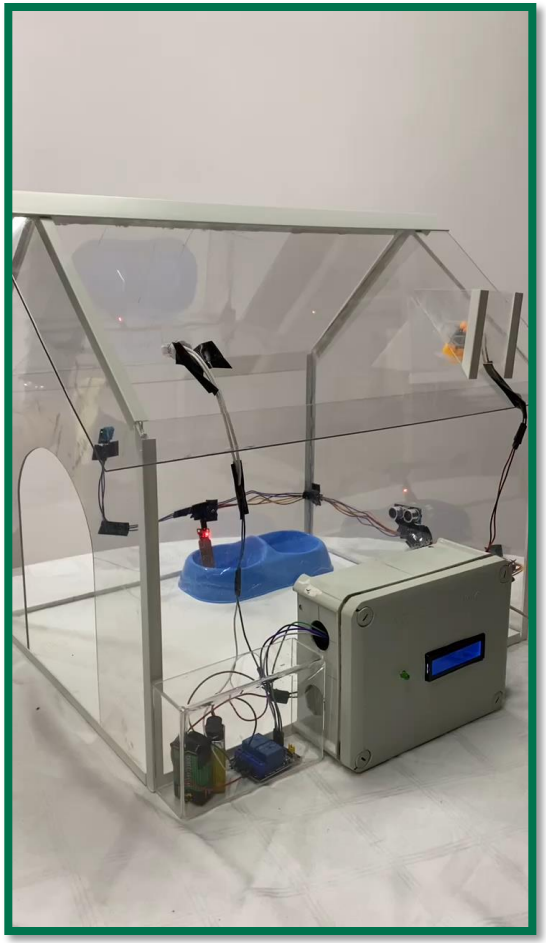


VISUALE FRONTALE	VISUALE DALL'ALTO	VISUALE 3D



CODICE ARDUINO IDE

```
void loop() {
  delay(2000);

 //SENSORE ULTRASUONI
 digitalWrite(pinTrigger,LOW);
 delayMicroseconds(2);
 digitalWrite(pinTrigger,HIGH);
 delayMicroseconds(10);
 digitalWrite(pinTrigger,LOW);

 long durata= pulseIn(pinEcho,HIGH);
 long distanza = durata/58;   // velocità del suono=0,0343 cm/micros=(1/29) cm/micros

 if(distanza>=0&&distanza<massimaDistanza){
  Serial.println("L'animale è nella cuccia, distanza dal sensore "+String(distanza)+" cm");
  digitalWrite(pinLed,HIGH);
  }else{
  Serial.println("L'animale non è nella cuccia, distanza dal sensore "+String(distanza)+" cm");
  digitalWrite(pinLed,LOW);
  }

//SENSORE DHT11
int temp= dht.readTemperature();
Serial.println("Temperatura: "+String(temp)+"°C");

lcd.setCursor(0,0);
lcd.print("                ");
lcd.setCursor(0,0);
lcd.print("Temperatura:"+String(temp)+char(0xDF)+"C");

//AVVIO SISTEMA DI CLIMATIZZAZIONE AUTOMATICO SOLO SE L'ANIMALE E' NELLA CUCCIA
if(distanza<massimaDistanza){
if(temp>25) {
  digitalWrite(pinVentola,LOW);
  digitalWrite(pinLampadina,HIGH);
}
else if(temp<20){
  digitalWrite(pinLampadina,LOW);
  digitalWrite(pinVentola,HIGH);
}else{
  digitalWrite(pinLampadina,HIGH);
  digitalWrite(pinVentola,HIGH);
  }
}else{
  digitalWrite(pinLampadina,HIGH);
  digitalWrite(pinVentola,HIGH);
  }
```



CODICE ARDUINO IDE

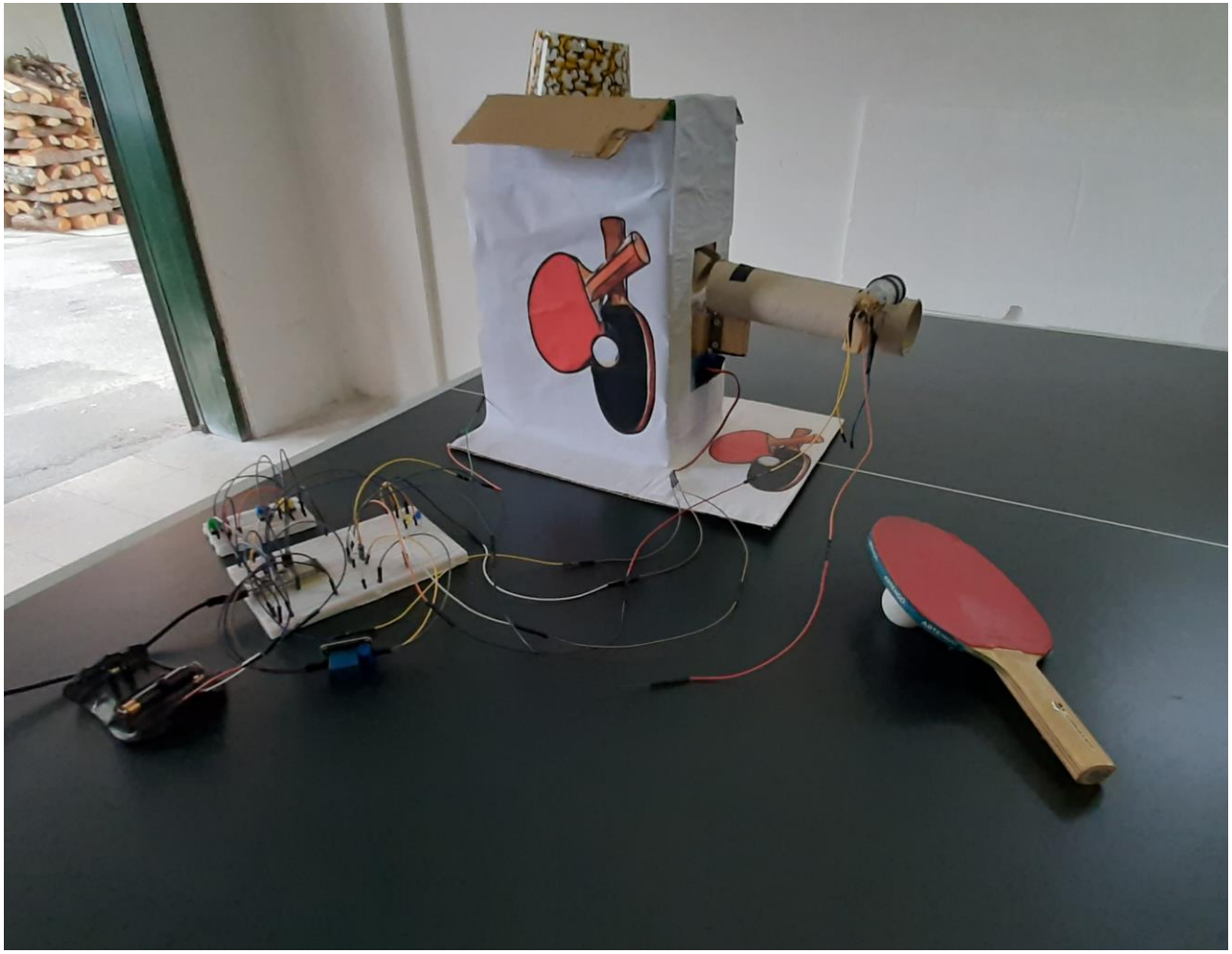
```
void loop() {
  delay(2000);

 //SENSORE ULTRASUONI
 digitalWrite(pinTrigger,LOW);
 delayMicroseconds(2);
 digitalWrite(pinTrigger,HIGH);
 delayMicroseconds(10);
 digitalWrite(pinTrigger,LOW);

 long durata= pulseIn(pinEcho,HIGH);
 long distanza = durata/58;

 if(distanza>=0&&distanza<massimaDistanza){
  Serial.println("L'animale è nella cuccia, distanza dal sensore "+String(distanza)+" cm");
  digitalWrite(pinLed,HIGH);
  }else{
  Serial.println("L'animale non è nella cuccia, distanza dal sensore "+String(distanza)+" cm");
  digitalWrite(pinLed,LOW);
  }

//SENSORE DHT11
int temp= dht.readTemperature();
Serial.println("Temperatura: "+String(temp)+"°C");

lcd.setCursor(0,0);
lcd.print("                ");
lcd.setCursor(0,0);
lcd.print("Temperatura:"+String(temp)+char(0xDF)+"C");

//AVVIO SISTEMA DI CLIMATIZZAZIONE AUTOMATICO SOLO SE L'ANIMALE E' NELLA CUCCIA
if(distanza<massimaDistanza){
if(temp>25) {
  digitalWrite(pinVentola,LOW);
  digitalWrite(pinLampadina,HIGH);
}
else if(temp<20){
  digitalWrite(pinLampadina,LOW);
  digitalWrite(pinVentola,HIGH);
}else{
  digitalWrite(pinLampadina,HIGH);
  digitalWrite(pinVentola,HIGH);
  }
}else{
  digitalWrite(pinLampadina,HIGH);
  digitalWrite(pinVentola,HIGH);
  }
```


Temperatura:21°C
Acqua: OK

Controllo da Remoto

```
#include "BluetoothSerial.h"
#define pinVentola 15
#define pinLampadina 4
String comando="";
BluetoothSerial SerialBT;
void setup() {
  pinMode(pinVentola,OUTPUT);
  pinMode(pinLampadina,OUTPUT);
  Serial.begin(9600);
  SerialBT.begin("Cuccia automatizzata");
  Serial.println("Connettiti al bluetooth"); }
void loop() {
  if (SerialBT.available()) {
    char carattere = SerialBT.read();
    if(carattere!= '\n'){
     comando+= String(carattere);
    }else{
      comando="";}
    Serial.write(carattere);
   if(comando=="accendiVentola"){
      digitalWrite(pinVentola,LOW);
    }
    else if(comando=="spegniVentola") {
    digitalWrite(pinVentola,HIGH);
   }
   else if(comando=="accendiLuce") {
    digitalWrite(pinLampadina,LOW);
   }
   else if(comando=="spegniLuce") {
    digitalWrite(pinLampadina,HIGH);
   }
   else{}
  }delay(20);}
```

Arbitro Automatizzato



Specifica Requisiti

Realizzare un arbitro automatizzato per le partite di ping-pong. Il dispositivo supporta due modalità: partita e allenamento

Nella **modalità partita** il dispositivo consente di tenere traccia dei punti dei due giocatori, indicare il giocatore che deve battere e richiedere una pallina da ping-pong, che verrà lanciata dal cannone

Nella **modalità allenamento** il dispositivo consente di scegliere un livello di difficoltà:

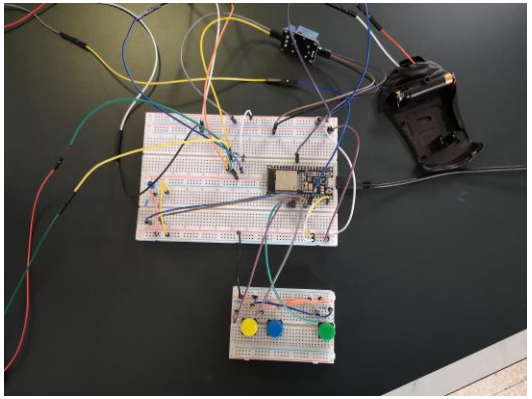
- *facile*: il cannone lancia n palline in una direzione preimpostata e fissa;
- *intermedio*: il cannone lancia n palline in diverse direzioni con velocità media;
- *difficile*: il cannone lancia n palline in diverse direzione con velocità elevata.

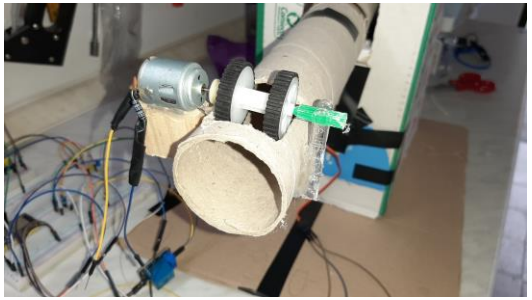
Per utilizzare tale dispositivo, è necessario utilizzare un apposito sito web, in cui ciascun giocatore è invitato ad inserire il proprio nome

Com’è realizzato?

I **sensori** utilizzati sono:

- Tre bottoni;
- Servo motore;
- 2 Motori DC;
- 2 Led;
- Relay.





Il cuore del dispositivo è un ESP32, dotato di modulo Wi-Fi integrato, tramite cui Arduino invia al database i seguenti dati:

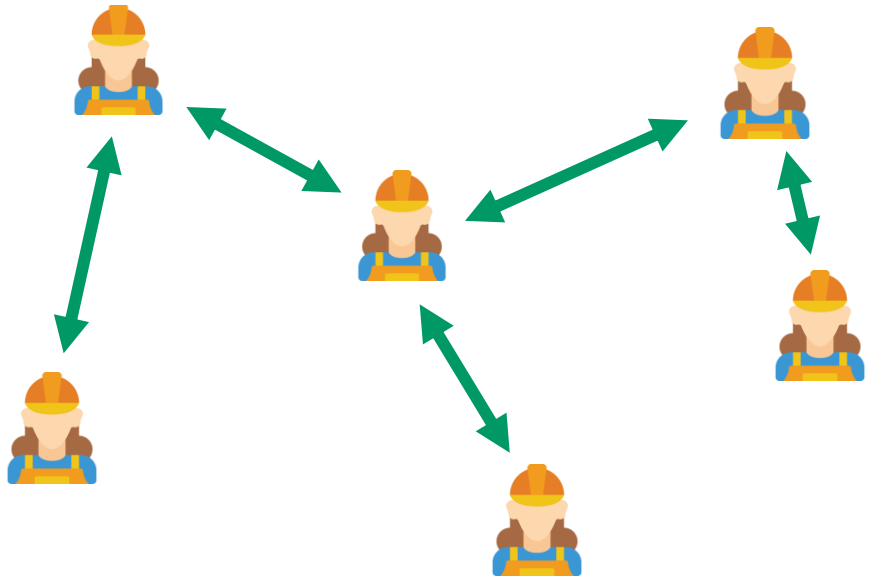
Modalità partita: punteggio Giocatore1 e Giocatore2 ad ogni incremento

Modalità allenamento: modalità scelta, posizioni del cannone, durata dell’allenamento

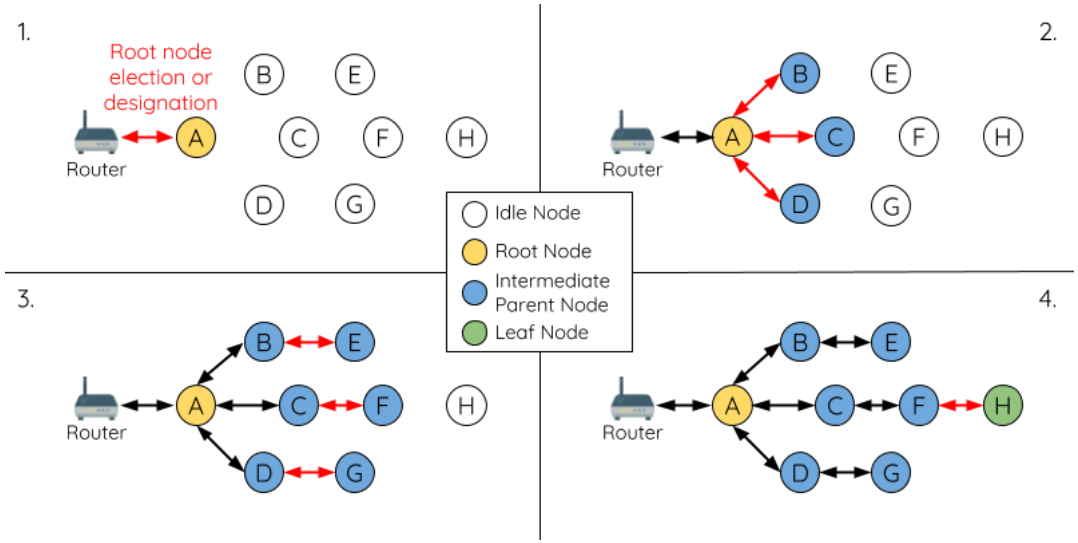
Turbina Eolica

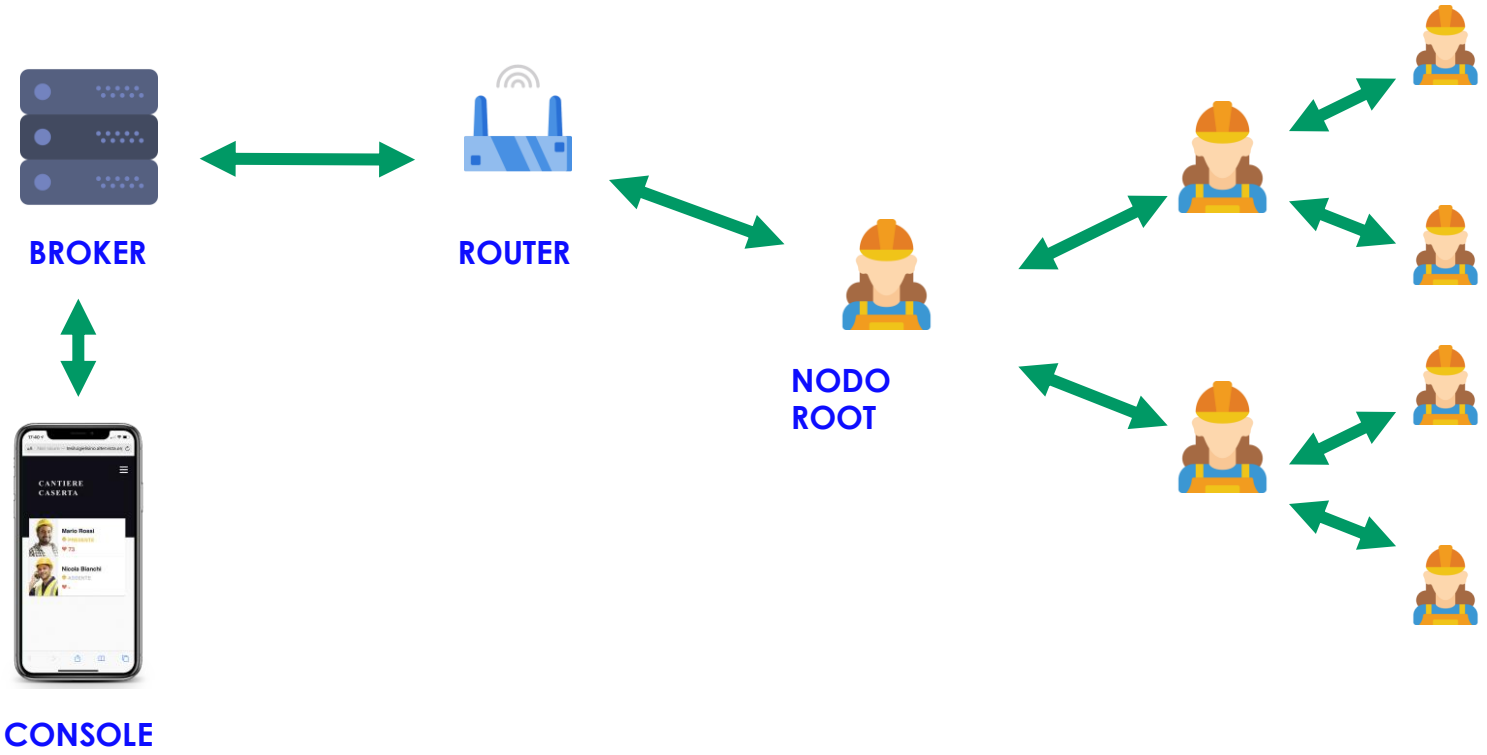
https://cirociampaglia.altervista.org/portfolio-progetti/software-gestionale-e-di-monitoraggio/11-iot-turbina-eolica

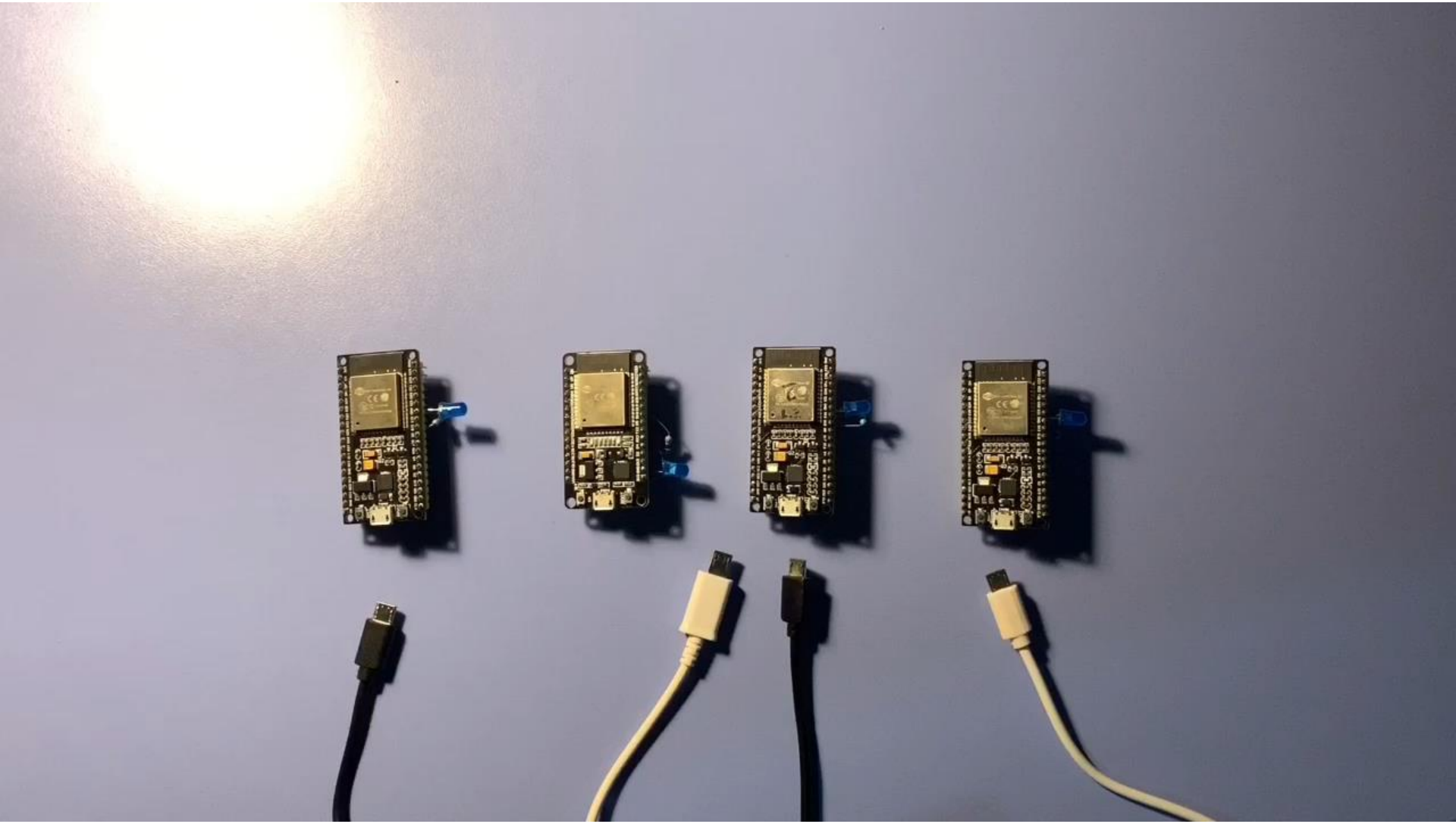
Reti Ad Hoc

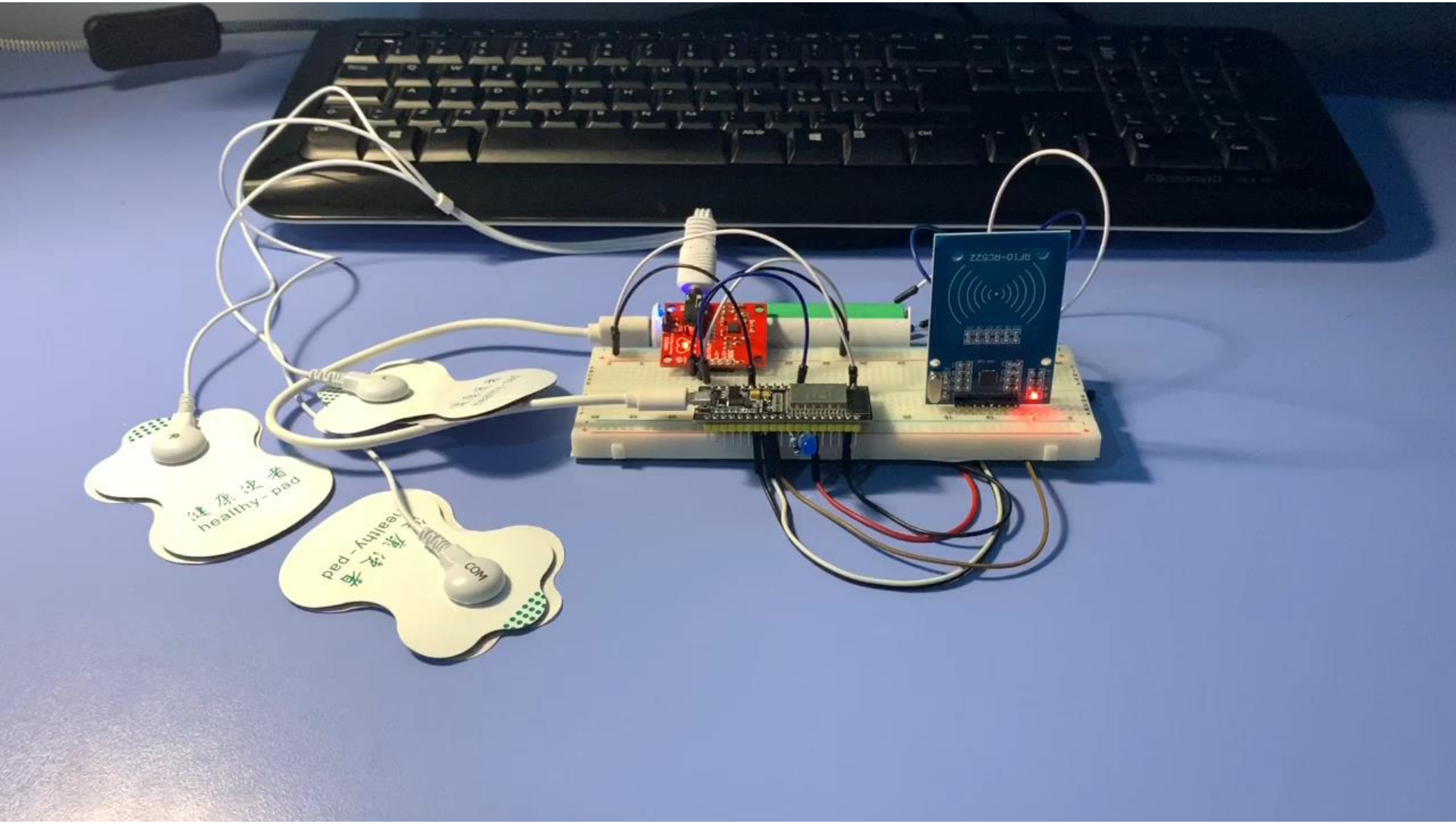


ESP-Mesh


1.
Root node election or designation
Router
A
B
C
D
E
F
G
H
2.
Router
3.
Router
4.
Router
Idle Node
Root Node
Intermediate Parent Node
Leaf Node


BROKER
ROUTER
NODO
ROOT
CONSOLE
CANTIERE
CASERTA

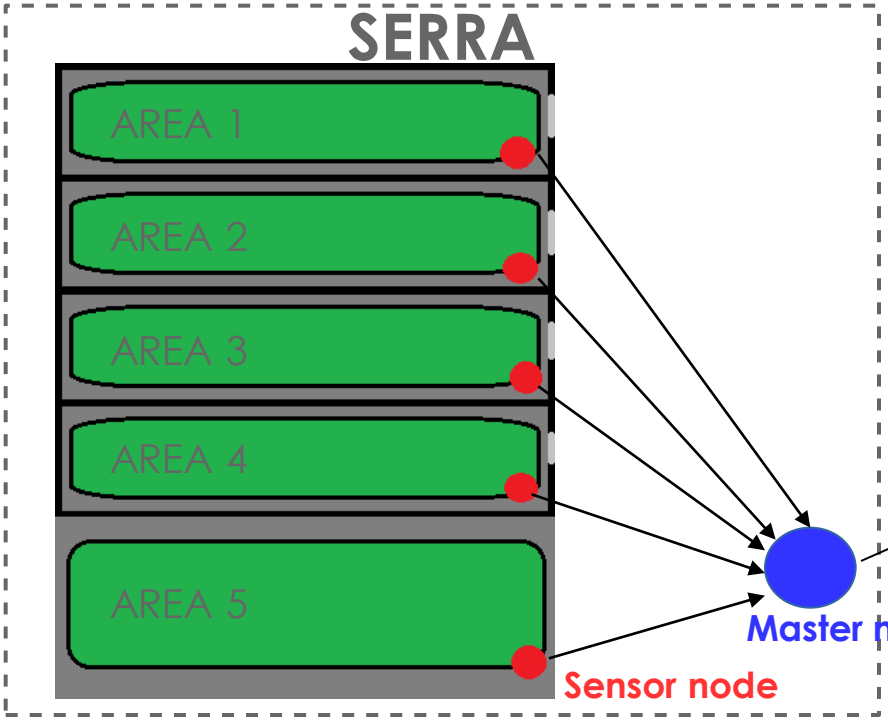



healthy - pad
COM

Gestione Serre Agricole

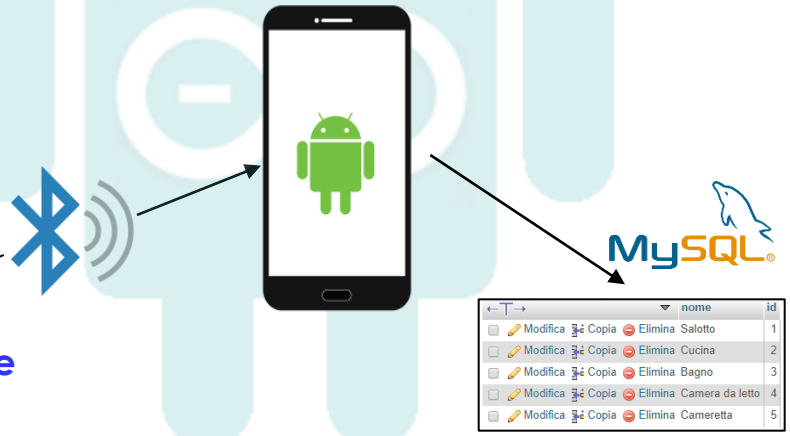
Rete di sensori wireless

- I sensor node acquisiscono i parametri dai sensori e li inviano al master node;
- Il master node riceve i parametri e li invia all'applicazione Android.

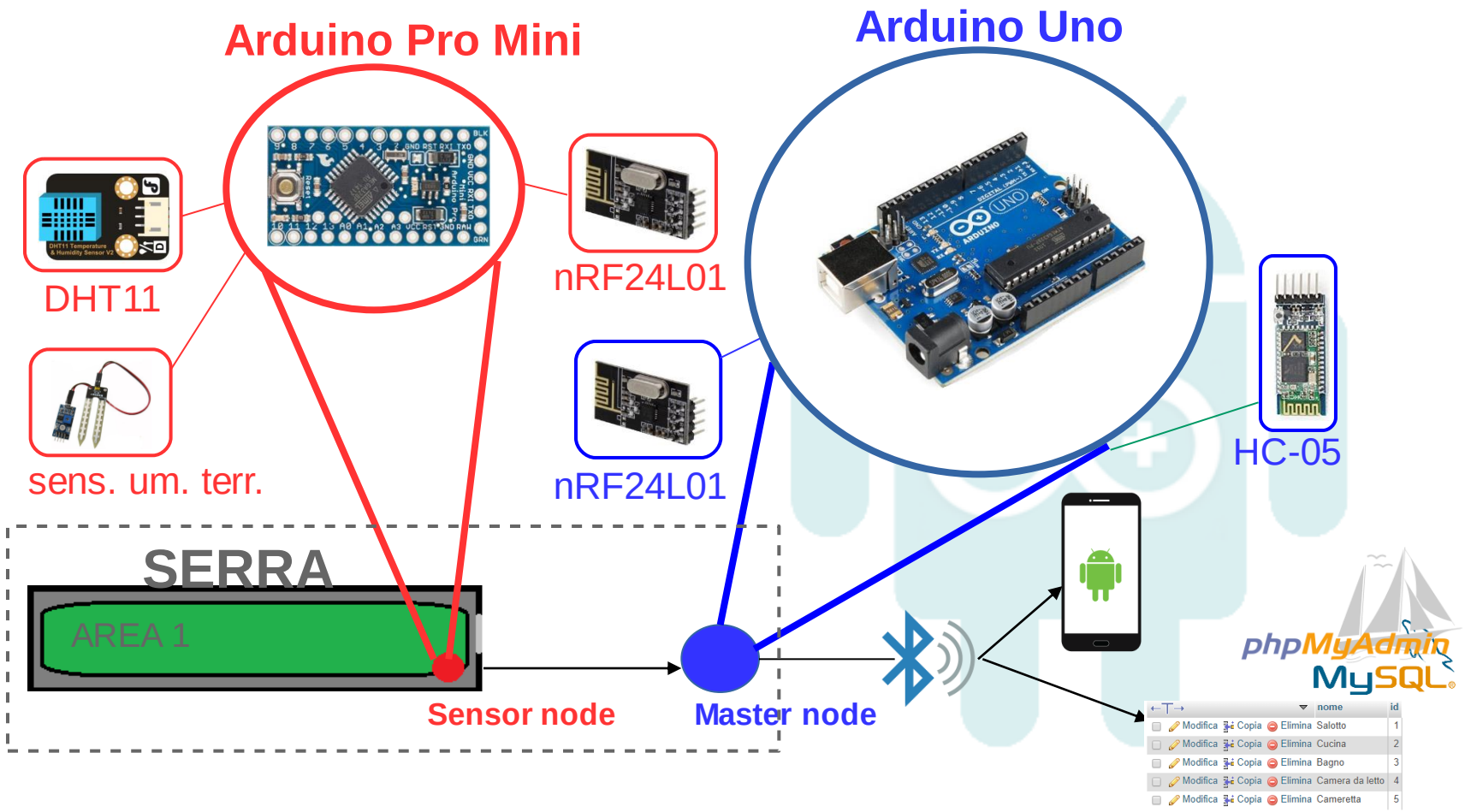


Applicazione Android

- Comunica con il nodo master della rete tramite il protocollo bluetooth;
- Monitora e controlla i parametri ambientali della serra.



Architettura Hardware



Layout Grafico

Selezione area

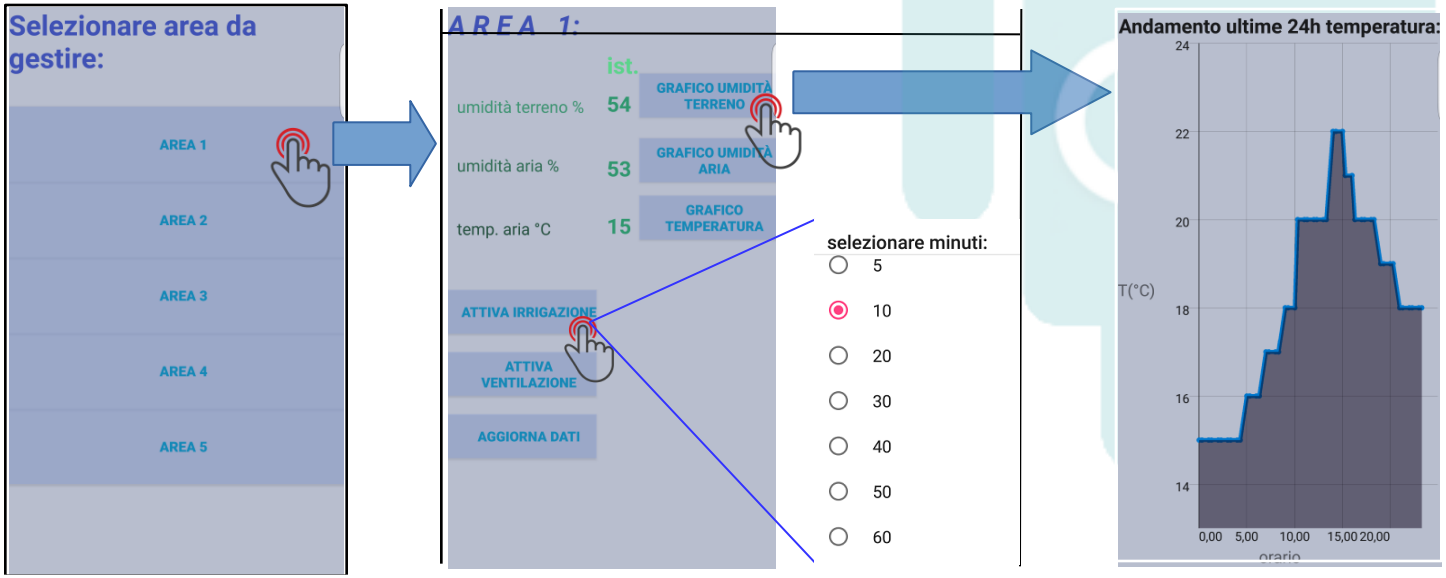
Consente di selezionare un area della serra della quale si vogliono controllare i parametri.

Gestione area

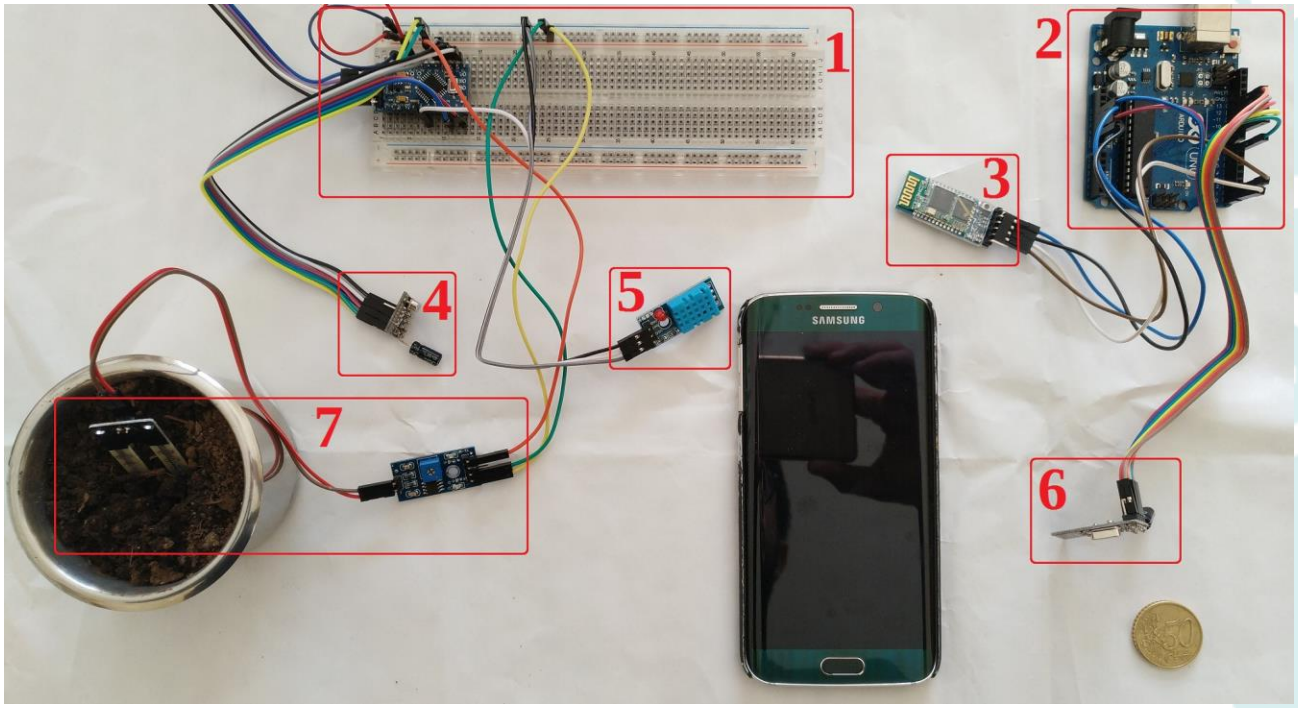
Visualizza i parametri della serra e permette di attivare gli impianti di irrigazione e ventilazione.

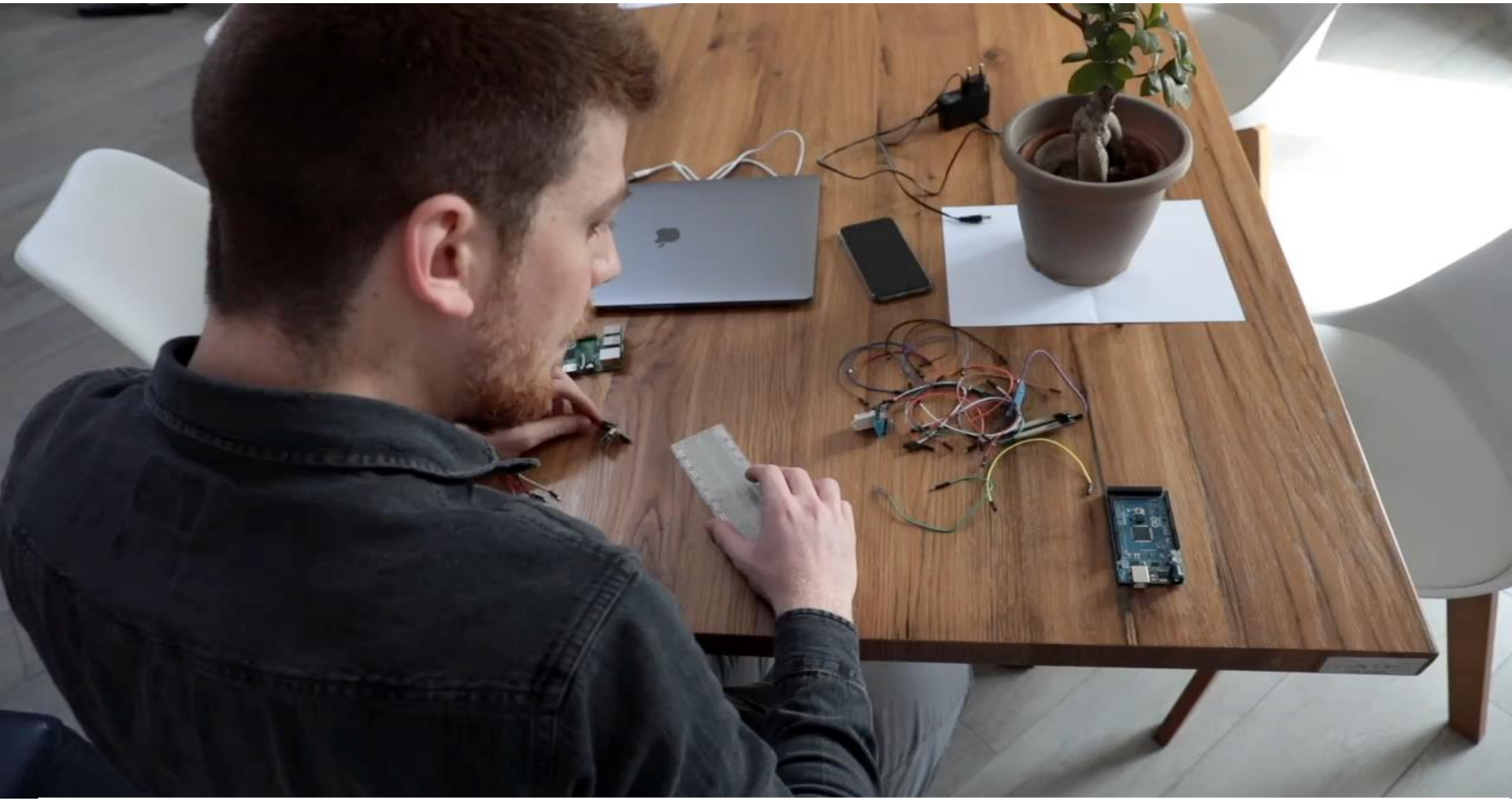
Grafico parametri

Visualizza il grafico dell'andamento di un parametro nelle ultime ventiquattro ore.

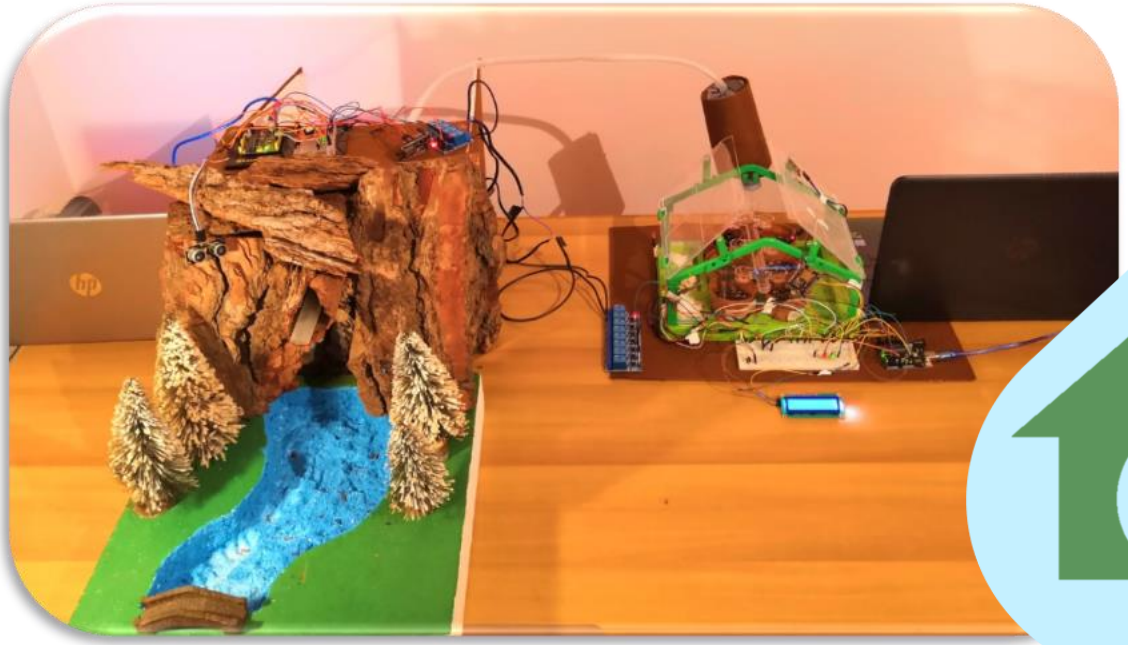


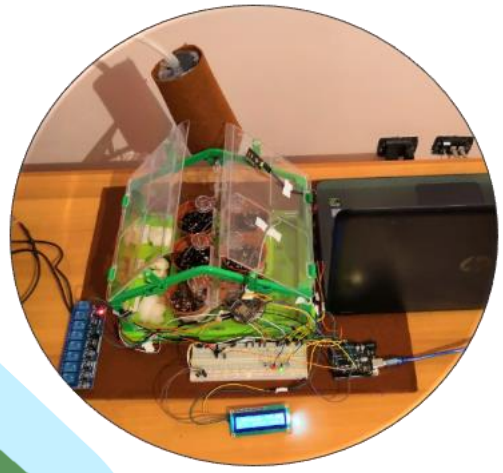
Prototipo





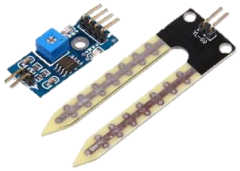
GreenHouse

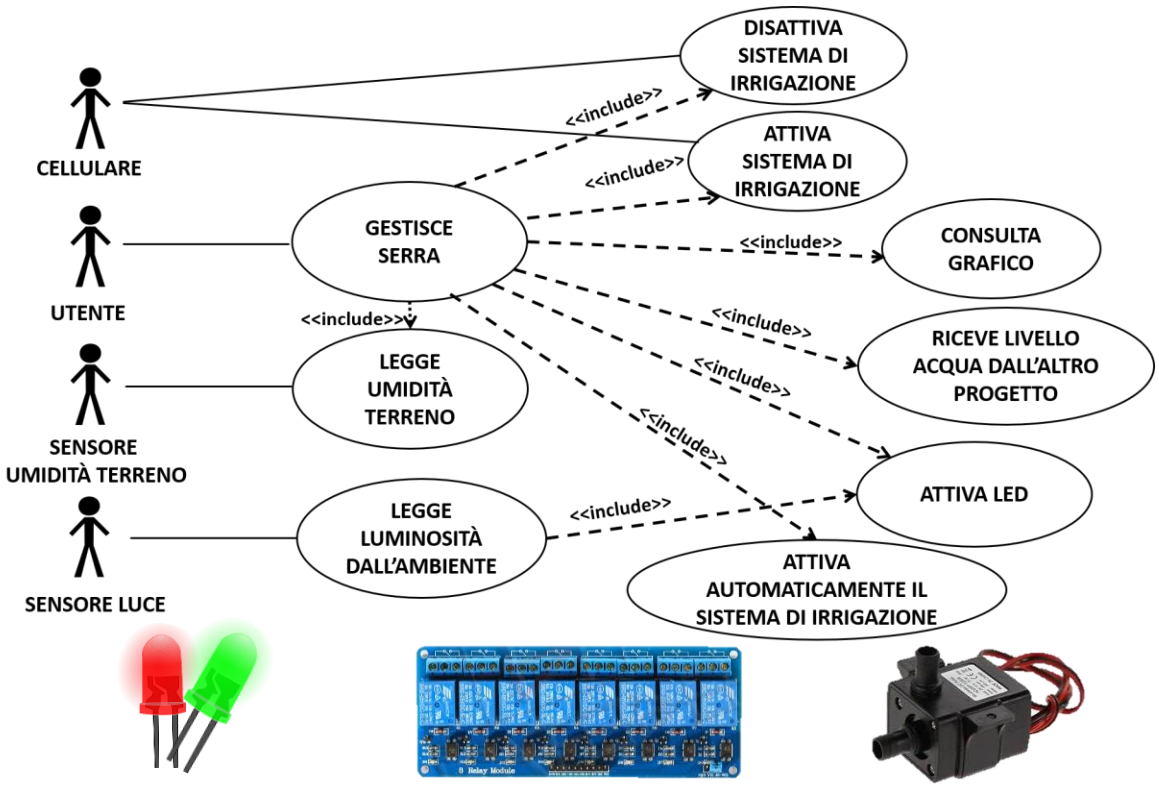




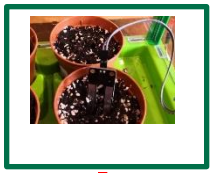
GreenHouse

Componenti




CELLULARE
UTENTE
SENSORE UMIDITÀ TERRENO
SENSORE LUCE
DISATTIVA SISTEMA DI IRRIGAZIONE
ATTIVA SISTEMA DI IRRIGAZIONE
GESTISCE SERRA
CONSULTA GRAFICO
LEGGE UMIDITÀ TERRENO
RICEVE LIVELLO ACQUA DALL'ALTRO PROGETTO
ATTIVA LED
LEGGE LUMINOSITÀ DALL'AMBIENTE
ATTIVA AUTOMATICAMENTE IL SISTEMA DI IRRIGAZIONE
<<include>>

Funzionalità

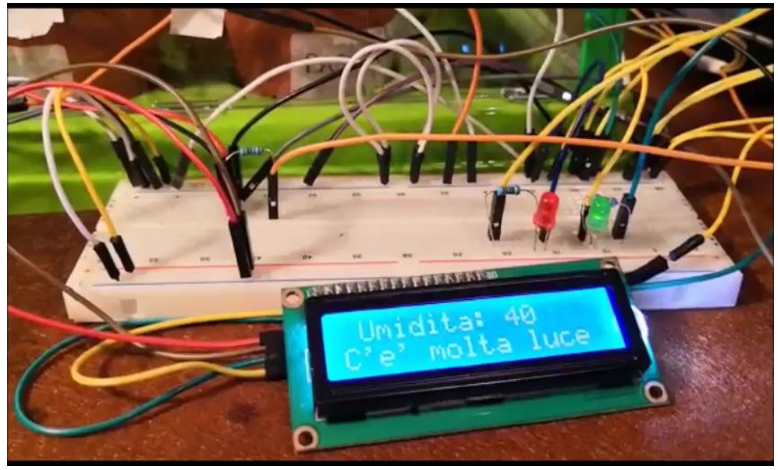


Collegato a

```
moistureValue = analogRead(MOISTURE_PIN);
```

Comunicazione seriale

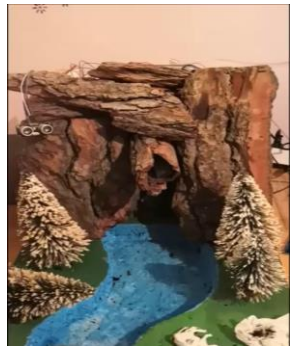
```
if( umiditaTerreno > regionecritica) i++;
    if(i==5){
        digitalWrite(8, HIGH);//IRR. ATTIVATA
        digitalWrite(11,HIGH);//rosso
        digitalWrite(13,LOW);//verde
        i=0; stoop==true;
          }}else{
         if(stoop==true){
         digitalWrite(8, LOW);//IRR. DISATTIVATA
         digitalWrite(13,HIGH);//verde
         digitalWrite(11,LOW);//ROSSO
                          }}
```




```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

t = now(); // in secondi
n = atof(t.c_str());
n=n*1000; // in millisecondi
root["xValue"]=n;
root["yValue"]=moistureValue;
String name = Firebase.push("/datigrafico",root);
```

Firebase

- datigrafico
  - -LVn-NPHspSn3EDLPo4u
    - xValue: 154704420000
    - yValue: 126
  - -LVn-NwJLIMo7VpXrC5W
    - xValue: 154704600000
    - yValue: 133
  - -LVn-OTIPqvCEZ3j90Rc
    - xValue: 154704780000
    - yValue: 116



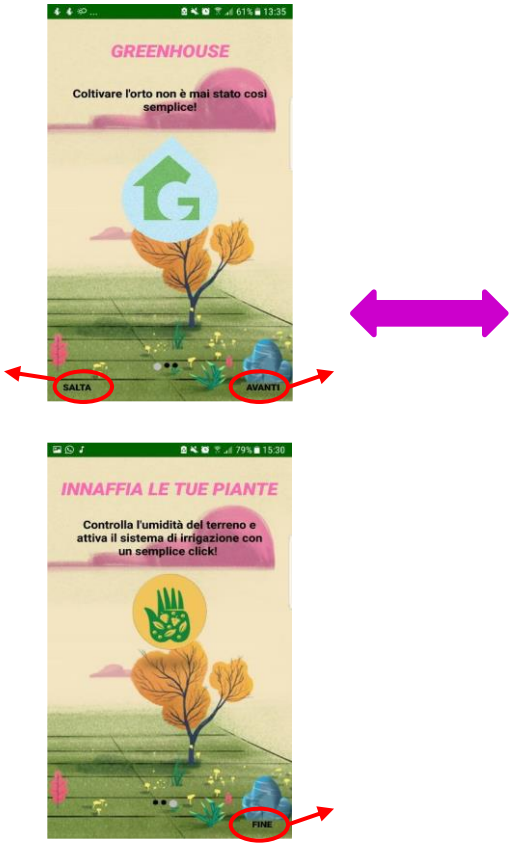
Funzionalità

```
int luce = analogRead(OUT_Pin);
if(luce<minLux){analogWrite(6,255); // intensità elevata
                lcd.setCursor(0,1);
                lcd.print("E' scuro");
                }else if(luce<medLux){
                     analogWrite(6,50);// intesità media
                     lcd.setCursor(0,1);
                     lcd.print("C'e' poca luce  ");
                     }else{analogWrite(6,LOW); // led spenti
                           lcd.setCursor(0,1);
                           lcd.print("C'e' molta luce");
                                 }
```

Applicazione Android

```
private ViewPager mSlideViewPager;
private SliderAdapter sliderAdapter;
```

```
sliderAdapter = new SliderAdapter( context: this);
mSlideViewPager.setAdapter(sliderAdapter);
```



È possibile aggiungere nuove coltivazioni specificando:

- Nome
- Data di semina e di raccolto mediante calendario;
- Eventuale immagine da galleria;
- Antenna Bluetooth necessaria ad attivare il sistema di irrigazione

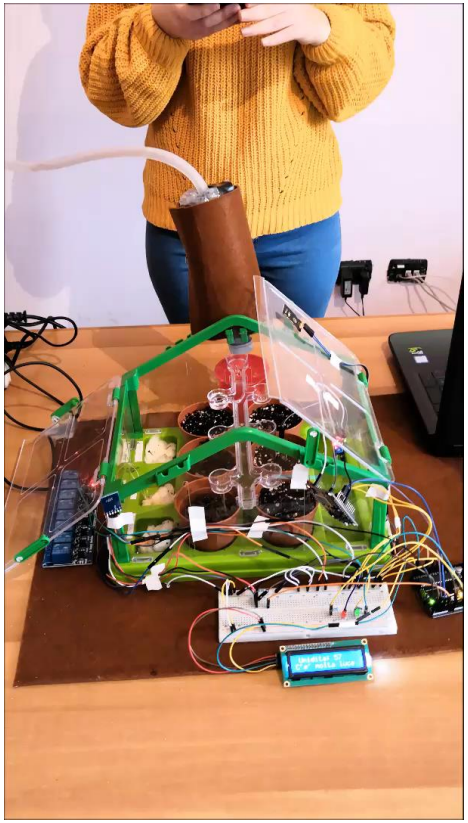
```
public class Coltivazione implements Serializable {
    String idColtivazione, nome, image,orto, bluetooth;
    long dataSemina, dataRaccolto;
```


```
btSocket.getOutputStream().write("1".toString().getBytes());
Toast.makeText(c, text: "Acceso", Toast.LENGTH_SHORT).show();
motoreOff.setEnabled(true);
motoreOn.setEnabled(false);
```

```
if(Serial.available() > 0) {
                command=Serial.read();
                if(command == '1' ){
                 digitalWrite(8, HIGH);
                 digitalWrite(11,HIGH);
                 digitalWrite(13,LOW);
                 stoop=false;}
```

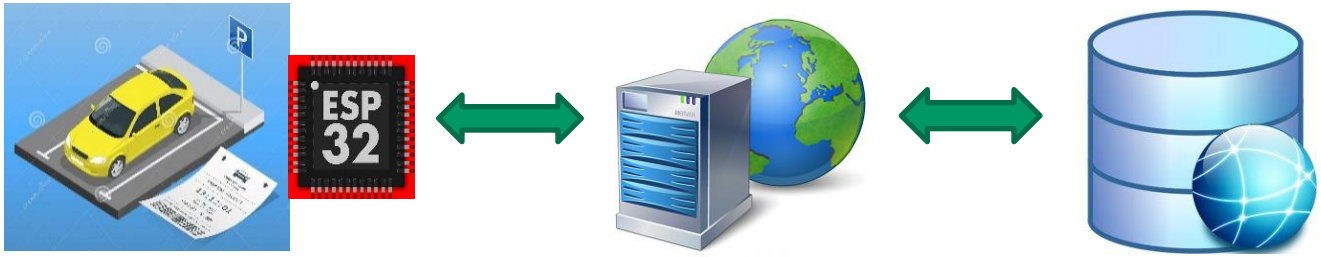
```
btSocket.getOutputStream().write("0".toString().getBytes());
Toast.makeText(c, text: "Spento", Toast.LENGTH_SHORT).show();
motoreOff.setEnabled(false);
motoreOn.setEnabled(true);
```

```
if(command =='0'){
          stoop=true;
          digitalWrite(8, LOW);
          digitalWrite(13,HIGH);
          digitalWrite(11,LOW);
                              }
```



Controllo e gestione di un parcheggio

L'architettura del sistema prevede un modello in scala del parcheggio, gestito da un microcontrollore che comunica con un Web Server, il quale a sua volta colloquia con un DBMS per la persistenza delle informazioni.

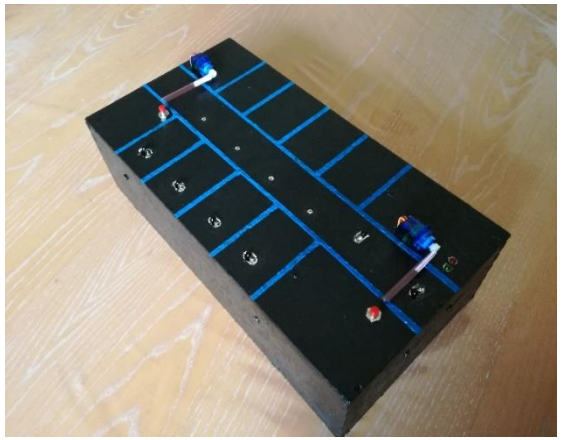


Architettura

Il parcheggio è dotato di quattro posti e in corrispondenza di ognuno di essi è collocato:

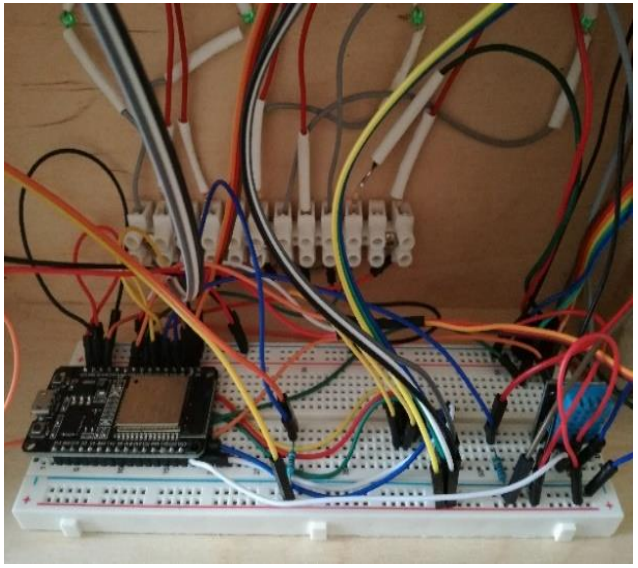
- un sensore di posizione, che rileva l'eventuale presenza di un'automobile parcheggiata;
- un led, per monitorare lo stato.

I limiti sono definiti da una sbarra e da un pulsante (simulazione il ritiro e il pagamento di un biglietto)

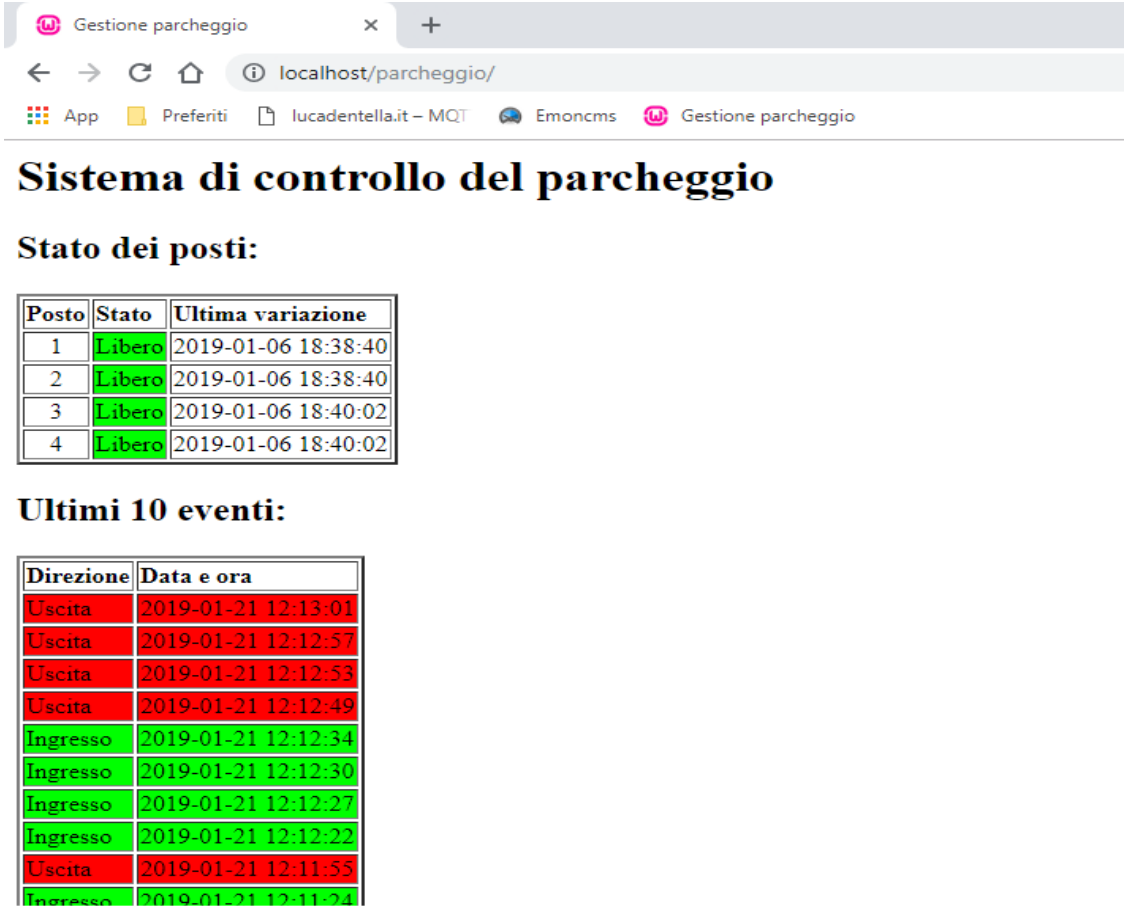


Prototipo

All'interno della struttura del parcheggio è stata installata una breadboard che comprende il microcontrollore e i collegamenti con i dispositivi di I/O



Pagina di monitoraggio



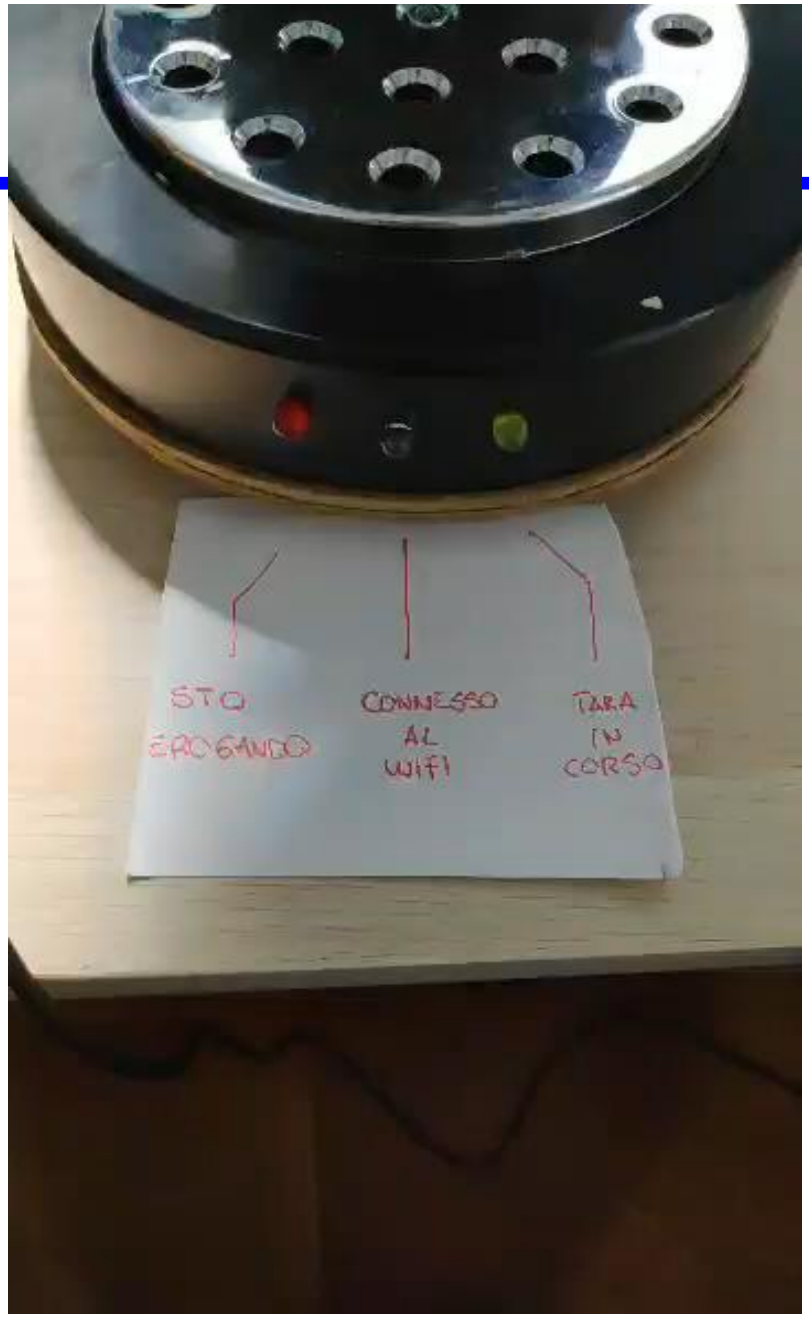
Distributore Automatico

Attraverso la stampante 3D è stato realizzato un prolungamento del fusto del distributore per contenere il motore e aumentare la distanza dalla base

Per accoppiare l'albero del motore all'albero del mulino, sempre attraverso stampa 3D, è stato prodotto il coupling

Attraverso invece due sbarre in ferro è stata stabilizzata la struttura

La base in plastica è accompagnata da due lastre di cartone, pensate per un agevole posizionamento della cella di carico sotto la ciotola


STO
EROGANDO
CONNESSO
AL
WIFI
TARA
IN
CORSO