PROGETTO del CORSO: PROTIPIZZAZIONE CON ARDUINO

Christian Muzzin (VR455988)
Mirko Morosato (VR486157)

maggio 2023

Indice

Capitolo 1

INTRODUZIONE

1.1 Obbiettivi del Progetto

Realizzazione di un sistema di sicurezza e controllo per porte interne all'abitazione, con presenza di tastierino numerico per l'autenticazione, insieme ad un sistema di controllo della porta via Telegram.

1.2 Descrizione generale del progetto

Il progetto consiste in un dispositivo che controlla l'apertura e la chiusura di una porta con un sistema di allarme.

Più nel dettaglio: Due dispositivi Arduino sono collegati tra loro attraverso connessione wireless e il secondo dispositivo è collegato anche ad un ESP32 che si collega al Bot di Telegram.

Il primo dispositivo presenta una tastiera numerica ove bisogna inserire una password che verrà verificata dallo stesso Arduino e scritta nel display, in caso di errore multiplo verrà attivato un all'arme presente nel secondo Arduino.

Il secondo dispositivo presenta: un sensore di prossimità per aprire e chiudere la porta, un display che indica lo stato della porta, un Buzzer che viene utilizzato come allarme e un ESP32 che permette di interfacciarsi con Telegram per controllare la porta anche attraverso lo smartphone.

Capitolo 2

PROGETTAZIONE DEL CIRCUITO

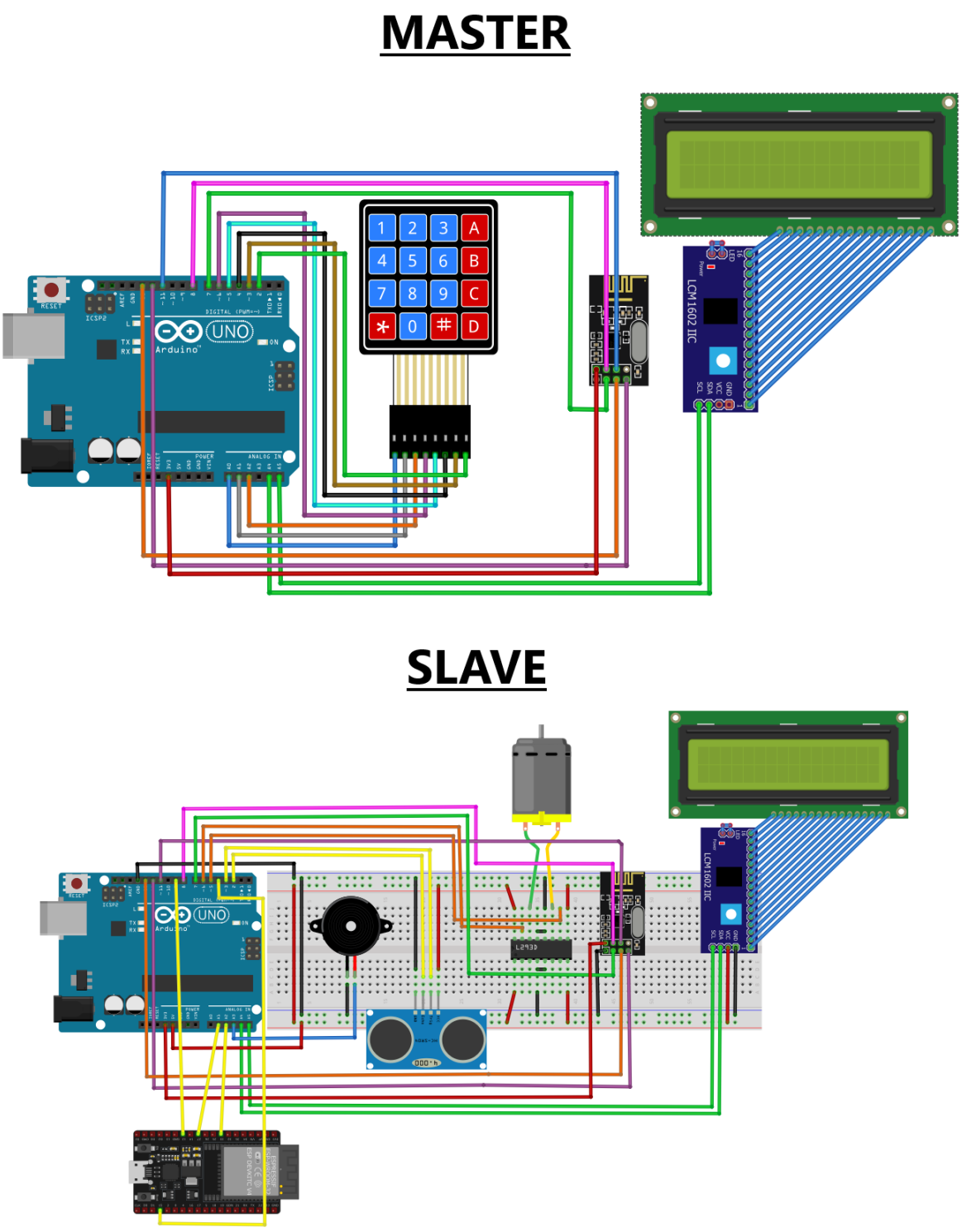
2.1 Materiali utilizzati

Di seguito è riportata una tabella che elenca tutti i componenti utilizzati nel progetto, insieme alla relativa quantità e utilizzo:

Nome Componente	**Unità Utilizzate**	**Descrizione Utilizzo**
Arduino UNO	2	
ESP32	1	Per il collegamento a Telegram.
Buzzer	1	Allarme, in caso di 3 errori consecutivi della password.
Display LCD 16x2 (I2C)	2	Display Master: per visualizzare la password e stato della porta. Display Slave: per la visualizzazione dello stato della porta.
Tastierino 4x4	1	Tastierino numerico per autenticazione, apertura e chiusura porta.
Sensore HC-SR04	1	Sensore di prossimità per apertura e chiusura porta.
NRF 24L01	2	Moduli Wireless per la comunicazione tra Arduini.
L293D	1	Driver per motore.
Motorino da 3 a 6 V	1	Motorino per apertura e chiusura della porta.
Cavi	∞	Per i collegamenti.

2.2 Schema elettrico

Di seguito è riportato lo schema elettrico del progetto:



2.3 Descrizione dei componenti

Di seguito è riportata una breve descrizione dei componenti utilizzati nel progetto:

- **Arduino UNO**: scheda di sviluppo basata su microcontrollore ATmega328P. È dotata di porte I/O digitali e analogiche, può essere programmata tramite un'interfaccia USB e supporta una vasta gamma di sensori e attuatori.
- **ESP32**: scheda di sviluppo basata su un processore dual-core a 32 bit. È dotata di connettività WiFi e Bluetooth, porta USB, porte I/O digitali e analogiche, e può essere programmata tramite un'interfaccia USB.
- **Buzzer**: un dispositivo elettronico che produce un suono continuo o intermittente, utilizzato ad esempio come allarme o segnalatore acustico.
- **Display LCD 16x2 (I2C)**: è un tipo di display a cristalli liquidi che utilizza un protocollo di comunicazione seriale chiamato I2C (Inter-Integrated Circuit) per interfacciarsi con la scheda di controllo, come ad esempio Arduino. L'interfaccia I2C richiede solo due fili per la comunicazione (oltre ai fili per l'alimentazione): il filo di dati (SDA) e il filo di clock (SCL). Questo permette di ridurre il numero di connessioni tra il display e la scheda di controllo rispetto a una configurazione standard in cui sarebbero necessarie molte connessioni separate.
- **Tastierino 4x4**: un dispositivo di input composto da una matrice di 16 tasti, organizzati in 4 righe e 4 colonne. Viene utilizzato per l'interazione con l'utente, ad esempio per l'inserimento di password o selezione di opzioni.
- **Sensore HC-SR04**: un sensore ad ultrasuoni utilizzato per la misura della distanza. È dotato di due trasduttori, uno per la generazione del segnale ad ultrasuoni e uno per la ricezione del segnale riflettuto.
- **NRF 24L01**: un modulo radio a 2,4 GHz utilizzato per la comunicazione wireless tra dispositivi. È dotato di un'antenna integrata e di un'interfaccia SPI per la connessione alla scheda di controllo.
- **L293D**: un driver di ponte H utilizzato per il controllo di motori DC. È dotato di quattro canali di uscita e può fornire fino a 600 mA di corrente.
- **Motorino da 3 a 6 V**: un piccolo motore elettrico utilizzato per la realizzazione di progetti di robotica e automazione. Viene alimentato a bassa tensione (3-6 V) e può essere controllato tramite il driver di ponte H.
- **Cavi**: i cavi sono utilizzati per collegare i vari componenti del circuito tra loro. Possono essere dotati di diversi tipi di connettori, ad esempio a pinza o a morsetto.

2.4 Formule

Formule utili per Arduino:

1. Calcolo della media dei valori letti da un sensore analogico utilizzando Arduino:

$$\text{Media} = \frac{1}{N} \int_{t_0}^{t_0+T} \text{analogRead}(A0)\, dt$$

2. Calcolo dell'energia consumata da un dispositivo elettronico controllato da Arduino:

$$\text{Energia} = \int_{t_1}^{t_2} \text{Potenza}\, dt$$

3. Controllo PID (Proporzionale-Integrativo-Derivativo) di un sistema utilizzando Arduino:

$$\text{U}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

Capitolo 3

PROGRAMMAZIONE DI ARDUINO

3.1 Codice del Master, Slave ed ESP32

Codice del Master:

```
//librerie per i moduli NRF24L01
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
//altre librerie
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include<SPI.h>

RF24 radio (7, 8);
const byte address[6]= "00001";

//configurazione tipo lcd
LiquidCrystal_I2C lcd(0x27,16,2);

//per tastierino
const byte ROWS = 4;
const byte COLS = 4;


const byte PASSWORDLENGTH = 5;
char password[PASSWORDLENGTH] = "12CD";
char chiudi[PASSWORDLENGTH] = "####";
char data[PASSWORDLENGTH];

int counter = 0;
int messaggio = 0;
int contaErrori = 0;
```

```
int i = 0;

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte pinROWS[ROWS] = {A0, A1, A2, 6};
byte pinCOLS[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), pinROWS, pinCOLS, ROWS, COLS);

void setup(){
  Serial.begin(9600);
  pinMode(pinLED,OUTPUT);
  lcd.begin(16,2); //Init with pin default ESP8266 or ARDUINO
  lcd.init();
  lcd.backlight(); //accende la retroilluminazione

  // Inizializza il modulo radio
  radio.begin();
  // Imposta il livello di potenza di trasmissione
  radio.setPALevel(RF24_PA_MIN);
  // Imposta la velocità di trasmissione dati
  radio.setDataRate(RF24_250KBPS);
  // Imposta il canale radio
  radio.setChannel(108);
  // Imposta l'indirizzo del mittente
  radio.openWritingPipe(address);
}

void loop()
{
  lcd.setCursor(0,0);
  char customKey = customKeypad.getKey();
  if (customKey)
  {
          data[counter] = customKey;
    lcd.setCursor(0,0);
    lcd.print(data);
    delay(100);
    counter++;

    //controllo password inserita
    if(counter==PASSWORDLENGTH-1)
    {
```

```
78            lcd.clear();
79      //guardo se la password messa è quella per aprire la porta
80      if(!strcmp(data,password))
81            {
82                        Serial.println("Password Corretta");
83                lcd.print("Password giusta");
84                messaggio=1;
85                digitalWrite(pinLED,HIGH);
86                delay(1000);
87                      digitalWrite(pinLED,LOW);
88                contaErrori=0;
89            }
90      //in caso non lo sia vedo se è quella per chiuderla
91      else if(!strcmp(data,chiudi))
92      {
93                Serial.println("Porta Chiusa");
94                lcd.print("Porta Chiusa");
95                messaggio=2;
96                digitalWrite(pinLED,HIGH);
97                delay(1000);
98                      digitalWrite(pinLED,LOW);
99      }
100     //se non è nanche quella allora..
101           else
102     {
103       Serial.println("Password Errata");
104             //se hai gia sbagliato fa partire l'allarme
105             if(contaErrori>1)
106       {
107         Serial.println("allarme");
108                lcd.print("Password Errata");
109                                lcd.setCursor(0, 1);
110                                lcd.print("blocco 10 secondi");
111         messaggio=0;
112         for(i=0;i<5;i++)
113               {
114                       digitalWrite(pinLED,HIGH);
115                              delay(1000);
116                                     digitalWrite(pinLED,LOW);
117                 delay(1000);
118                     }
119               contaErrori=0;
120       }
121       //altrimenti aumenta il contatore errori e puoi reinserirla
122       else
123       {
124         Serial.println("Psbaglio Errata");
125         lcd.print("Password Errata");
```

```
                delay(2000);
                contaErrori++;
        }
      }
            counter = 0;
            lcd.clear();
            for(i=0;i<4;i++)
      {
              data[i]=0;
      }
    }
  }
  //invia il messaggio
  Serial.print("messaggio: ");
  Serial.print(messaggio);
  Serial.println();
  radio.write(&messaggio,sizeof(messaggio));
  //pone il messaggio a 3 (dice allo slave di stare in posizione "fermo")
  messaggio=3;
}
```

Codice del Slave:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include<SPI.h>
#include<nRF24L01.h>
#include<RF24.h>
#include <RF24_config.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#define        trigger                3
#define        echo                2
#define m1p                        6        //avanti
#define m1n                        5        //indietro

//stato della porta (input a esp32)
#define ledblu1 4//ledVERDE
#define ledblu2 9//ledROSSO
//input da esp 32
//aprire porta
#define ledverde A2
//chiudere porta
#define ledrosso A1

#define buzzer A3

const int soglia=512;
```

```
25
26  RF24 radio(7,8);
27  const byte address[6] = "00001";
28
29  int statoPorta=0;//default porta aperta
30  int messaggio=3;
31  int i=20;//i = potenza motore
32
33  void setup()
34  {
35    pinMode(trigger,OUTPUT);
36    pinMode(echo,INPUT);
37    pinMode(m1p,OUTPUT);
38    pinMode(m1n,OUTPUT);
39    pinMode(buzzer,OUTPUT);
40    //
41    pinMode(ledblu1,OUTPUT);
42    pinMode(ledblu2,OUTPUT);
43    pinMode(ledrosso,INPUT);
44    pinMode(ledverde,INPUT);
45    //
46    lcd.begin(16,2); //Init with pin default ESP8266 or ARDUINO
47    lcd.init();
48    lcd.backlight(); //accende la retroilluminazione
49    Serial.begin(9600);
50    delay(2000);
51
52    //inizio parte wirless slave
53    // Inizializza il modulo radio
54    radio.begin();
55
56    // Imposta il livello di potenza di trasmissione
57    radio.setPALevel(RF24_PA_MIN);
58
59    // Imposta la velocità di trasmissione dati
60    radio.setDataRate(RF24_250KBPS);
61
62    // Imposta il canale radio
63    radio.setChannel(108);
64
65    // Imposta l'indirizzo del destinatario
66    radio.openReadingPipe(0, address);
67
68    // Abilita la ricezione dei messaggi
69    radio.startListening();
70  }
71  //funzione per sensore distanza
72  int getdist(){
```

```
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger,LOW);
  long durata = pulseIn(echo,HIGH);
  int distanza = durata*0.034/2;
  return distanza;
}

//funzione apertura porta
void apri()
{
  lcd.clear();
  digitalWrite(m1p, i);            //i= potenza motore(velocità)
  digitalWrite(m1n, LOW);
  lcd.print("apertura porta");
  delay(500);//da testare, tempo per girare chiave
  statoPorta=1;
  digitalWrite(ledblu1,HIGH);
  digitalWrite(ledblu2,LOW);
}

//funzione apertura porta
void chiudi()
{
  lcd.clear();
  digitalWrite(m1p, LOW);
  digitalWrite(m1n, i);               //i= potenza motore(velocità)
  lcd.print("chiusura porta");
  delay(500);//da testare, tempo per girare chiave
  statoPorta=0;
  digitalWrite(ledblu1,LOW);
  digitalWrite(ledblu2,HIGH);
}

//funzione che fa scattare l'allarme
void errore()
{
  lcd.clear();
  digitalWrite(m1p, LOW);
  digitalWrite(m1n, LOW);
  lcd.print("tentato ingr");
  while(messaggio==0)
  {
    radio.read(&messaggio, sizeof(messaggio));
          for(int j=0;j<3;j++){
          digitalWrite(buzzer, HIGH);
```

```
121     delay(500);
122     digitalWrite(buzzer, LOW);
123           delay(500);
124           }
125   }
126   digitalWrite(ledblu1,HIGH);
127   digitalWrite(ledblu2,HIGH);
128 }
129
130 //stato in cui sta se non avvengono azioni
131 void fermo()
132 {
133   lcd.clear();
134   digitalWrite(m1p, LOW);
135   digitalWrite(m1n, LOW);
136
137   lcd.print(".");
138   delay(500);
139   lcd.clear();
140   lcd.print("..");
141   delay(500);
142   lcd.clear();
143   lcd.print("...");
144   delay(200);
145 }
146
147 void loop()
148 {
149   if(radio.available())
150   {
151           radio.read(&messaggio, sizeof(messaggio));
152     int ostacolo = getdist();//calcolo_distanza();
153            int apertura=analogRead(ledverde);
154     int chiusura=analogRead(ledrosso);
155     if (ostacolo<5 && statoPorta==0)
156     {
157             apri();
158       Serial.println("apri ostacolo");
159       delay(1000);
160     }
161     else if (ostacolo<5 && statoPorta==1)
162     {
163       chiudi();
164       Serial.println("chiudi ostacolo");
165       delay(1000);
166     }
167     else if((messaggio==1) && statoPorta==0)
168     {
```

```
            apri();
      Serial.println("apri password");
    }else if((apertura>soglia)&&(chiusura<soglia)&&statoPorta==0)
    {
      apri();
      Serial.println("apri bot");
    }
          else if((messaggio==2) && statoPorta==1)
    {
            chiudi();
      Serial.println("chiudi password");
    }
    else if((apertura<soglia)&&(chiusura>soglia)&&statoPorta==1)
    {
      chiudi();
      Serial.println("chiudi bot");
    }
          else if(messaggio==0)
    {
            errore();
      Serial.println("errore");
    }
          else
    {
            fermo();
    }
  }
}
```

Codice dell'ESP32:

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
/*
char* ssid = "TIM-nuovarete";
char* password = "123456789abc";
*/
char* ssid = "Oppo";
char* password = "z7digay3";
char* botToken = "6080515741:AAG_E3pzqBO33FovHK05FklEqi7E9byVzWo";
//////////////////////////////////////
#define apertura  33//output porta aperta
#define chiusura  27//output porta chiusa
#define Pindato1  15//input porta 1
#define Pindato2  12//input porta 2
int vet[10]={0,0,0,0,0,0,0,0,0,0};
```

```
int pinval1 = 0;
int pinval2 = 0;
int mem1=0;
int mem2=0;
String stringastorico = "";
////////////////////////////////////////
void setup() {
  ////////////////////////////
  pinMode(apertura, OUTPUT);
  pinMode(chiusura, OUTPUT);
  pinMode(Pindato1, INPUT);
  pinMode(Pindato2, INPUT);
  ////////////////////////////
  Serial.begin(115200);
  delay(1000);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Ottieni chat_id dell'utente che ha inviato l'ultimo messaggio al bot
  String url = "https://api.telegram.org/bot" + String(botToken) + "/getUpdates";
  HTTPClient http;
  http.begin(url);
  int httpCode = http.GET();
  if (httpCode > 0) {
    String payload = http.getString();
    Serial.println(payload); // stampa la risposta dell'API per il debug
    // Analizza la risposta JSON dell'API e ottieni il chat_id
    int chat_id = payload.substring(payload.indexOf("\"chat\":{\"id\":") + 13, payload.indexOf(",
    Serial.println("chat_id: " + String(chat_id)); // stampa il chat_id per il debug
  }

  delay(1000);
}

void loop() {
  //String x= recive();
///////////////cambia da uttore a stringa per storico
  pinval1=digitalRead(Pindato1);
  pinval2=digitalRead(Pindato2);
  if(pinval1!=mem1 || pinval2!=mem2)
  {
    mem1=pinval1;
    mem2=pinval2;
    for (int i=9;i>0;i--)
```

```
65      {
66        vet[i]=vet[i-1];
67      }if (pinval1==HIGH && pinval2==LOW)
68      {vet[0]=1;}
69      else if (pinval1==LOW && pinval2==HIGH)
70      {vet[0]=2;}
71      else if (pinval1==HIGH && pinval2==HIGH)
72      {vet[0]=3;}
73      else if(pinval1==LOW && pinval2==LOW)
74      {vet[0]=4;}
75    }
76    stringastorico= "";
77    for (int i = 0; i < 10; i++) {
78      stringastorico += String(vet[i]) + " ";
79    }
80  ////////////////////
81    // Controlla se ci sono nuovi messaggi
82    String msgLast = recive();
83    Serial.println("Ultimo messaggio: " + msgLast);
84
85    if(msgLast == "apri") {
86      //TODO
87      send("-1001934292459", "Porta Aperta");
88      digitalWrite(apertura, HIGH);
89      digitalWrite(chiusura, LOW);
90      delay(1000);
91    }
92    else if(msgLast == "chiudi") {
93      //TODO
94      send("-1001934292459", "Porta Chiusa");
95      digitalWrite(chiusura, HIGH);
96      digitalWrite(apertura, LOW);
97      delay(1000);
98    }
99    else if(msgLast == "statoporta") {
100     send("-1001934292459", "Stato Porta:");
101     //TODO
102     if(digitalRead(Pindato1)==HIGH && digitalRead(Pindato2)==LOW)
103     {
104       send("-1001934292459", "Porta aperta");
105     }else if (digitalRead(Pindato1)==LOW && digitalRead(Pindato2)==HIGH)
106     {
107       send("-1001934292459", "Porta chiusa");
108     }else if (digitalRead(Pindato1)==HIGH && digitalRead(Pindato2)==HIGH)
109     {
110      send("-1001934292459", "allarme");
111     }else
112     {
```

```
      send("-1001934292459", "errore rivelamento");
    }
    delay(1000);
  }
  else if(msgLast == "storico") {
    send("-1001934292459", "Storico: ");
    //TODO
    send("-1001934292459", " 0=vuoto, 1=aperta, 2=chiusa, 3=allarme, 4=errore rilevamento: ");
    send("-1001934292459", stringastorico);
  }else if(msgLast == "toktok"){
    send("-1001934292459", "FBI open UP");
  }
  digitalWrite(chiusura, LOW);
  digitalWrite(apertura, LOW);

}


// ----------------------------------------------------------------------------------------
String tempPayload="";
String recive()
{
  String url = "https://api.telegram.org/bot" + String(botToken) + "/getUpdates?offset=-1"; // of
  HTTPClient http;
  http.begin(url);
  int httpCode = http.GET();
  if (httpCode > 0) {
    String payload = http.getString();
  //  Serial.println(payload);
    http.end();
    delay(100);
    if(tempPayload == "")
    {
      tempPayload = payload;
    }
    if(tempPayload != payload)
    {
      tempPayload = payload;
      return search(payload);
    }
    return "";
  }

  else
    return "Error recive: " + String(httpCode);
}


```

```
161 void send(String chatID, String message)
162 {
163   String url = "https://api.telegram.org/bot" + String(botToken) + "/sendMessage?chat_id=" + chat
164   HTTPClient http;
165   http.begin(url);
166   int httpCode = http.GET();
167 
168   if(httpCode > 0)
169     Serial.println("CODE: " + httpCode);
170 
171   Serial.println("Message sent");
172   http.end();
173   delay(100);
174 }
175 
176 
177 String search(String text)
178 {
179   String tmp = "";
180   for(int i=0; text[i]!='\0'; i++)
181   {
182     if(text[i] == '/')
183     {
184       i++;
185       for(int j=i; text[j]!='@' ;j++)
186         tmp += text[j];
187     }
188   }
189   return tmp;
190 }
191 
192 // ------------------------------------------------------------------------------------------
```

3.2 Descrizione del funzionamento del codice

In questa sezione verranno descritte le parti più importanti del codice e il loro funzionamento.

3.2.1 Descrizione del MASTER:

Il master serve per il controllo dell' accesso all'utilizzo di un tastierino e un modulo radio NRF24L01. Le funzioni principali del codice sono le seguenti:

- setup(): Questa funzione viene eseguita una volta all'avvio del programma. Inizializza la comunicazione seriale, imposta la modalità di uscita per un pin LED,

inizializza il display LCD, inizializza il modulo radio NRF24L01 impostando la potenza di trasmissione, la velocità di trasmissione, il canale radio e l'indirizzo del mittente.

- loop(): Questa funzione viene eseguita in modo continuo dopo l'esecuzione della funzione setup(). Gestisce la logica principale del programma. Legge l'input dal tastierino, visualizza i caratteri inseriti sul display LCD, controlla se la password inserita corrisponde a quella prevista, gestisce le azioni in base alla corrispondenza della password e invia un messaggio tramite il modulo radio.
- customKeypad.getKey(): Questa funzione viene utilizzata per leggere il tasto premuto sul tastierino matriciale. Restituisce il carattere corrispondente al tasto premuto o restituisce NULL se nessun tasto è premuto.
- lcd.setCursor(col, row): Questa funzione imposta la posizione del cursore sul display LCD in base alla colonna (col) e alla riga (row) specificate.
- lcd.print(str): Questa funzione stampa una stringa (str) sul display LCD alla posizione del cursore corrente.
- strcmp(str1, str2): Questa funzione confronta due stringhe (str1 e str2) e restituisce

3.2.2 Descrizione dello SLAVE:

Lo slave serve il controllo della serratura, dell'allarme, dell'LCD e per interfacciarsi con l'ESP32. Le funzioni principali del codice sono le seguenti:

- Funzione `setup()`:
  - Imposta la modalità dei pin (input o output).
  - Inizializza il display LCD.
  - Inizializza la comunicazione seriale a 9600 bps.
  - Delay iniziale di 2 secondi.
  - Inizializza il modulo RF24 con le impostazioni specificate (livello di potenza, velocità di trasmissione, canale, indirizzo del destinatario) e abilita la ricezione dei messaggi.
- Funzione `getdist()`:
  - Utilizza un sensore di distanza per misurare la distanza da un oggetto.
  - Invia un impulso di trigger al sensore elettronico ad ultrasuoni.
  - Calcola la durata del segnale di echo.
  - Calcola la distanza in base alla durata del segnale.
  - Restituisce la distanza misurata.
- Funzione `apri()`:
  - Pulisce il display LCD.

  - Imposta il pin `m1p` a `HIGH` (avanti) e il pin `m1n` a `LOW` (indietro) per aprire la porta.
  - Visualizza un messaggio sul display LCD.
  - Ritarda per 500 millisecondi (tempo per girare la chiave).
  - Imposta lo stato della porta a 1 (porta aperta).
  - Accende il LED verde (`ledblu1`) e spegne il LED rosso (`ledblu2`).

- Funzione `chiudi()`:
  - Pulisce il display LCD.
  - Imposta il pin `m1p` a `LOW` (avanti) e il pin `m1n` a `HIGH` (indietro) per chiudere la porta.
  - Visualizza un messaggio sul display LCD.
  - Ritarda per 500 millisecondi (tempo per girare la chiave).
  - Imposta lo stato della porta a 0 (porta chiusa).
  - Spegne il LED verde (`ledblu1`) e accende il LED rosso (`ledblu2`).

- Funzione `errore()`:
  - Pulisce il display LCD.
  - Ferma il motore impostando entrambi i pin `m1p` e `m1n` a `LOW`.
  - Visualizza un messaggio di errore sul display LCD.
  - Legge i messaggi in arrivo tramite il modulo RF24 finché il valore di `messaggio` è 0.
  - Attiva il buzzer alternando l'accensione e lo spegnimento per tre volte.
  - Accende sia il LED verde (`ledblu1`) che il LED rosso (`ledblu2`).

- Funzione `fermo()`:
  - Pulisce il display LCD.
  - Ferma il motore impostando entrambi i pin `m1p` e `m1n` a `LOW`.
  - Visualizza una sequenza di punti animati sul display LCD.

- Funzione `loop()`:
  - Controlla se ci sono messaggi disponibili tramite il modulo RF24.
  - Legge il messaggio ricevuto.
  - Ottiene la distanza dall'oggetto tramite il sensore di distanza.
  - Legge i valori di `apertura` e `chiusura` dai pin analogici `ledverde` e `ledrosso`.
  - Esegue una serie di controlli condizionali per determinare l'azione da intraprendere.

3.2.3 Descrizione del codice per l'ESP32:

Tale codice serve per la gestione dell'ESP e per interfacciarlo con il bot di telegram per il controllo dello slave. Le funzioni principali del codice sono le seguenti:

- `setup()`: Funzione di inizializzazione eseguita una sola volta all'avvio del dispositivo.
- `loop()`: Funzione principale eseguita ciclicamente dopo la fase di inizializzazione.
- `recive()`: Funzione per ottenere gli aggiornamenti dei messaggi dal bot Telegram.
- `search(String text)`: Funzione per cercare un comando specifico nel testo del messaggio ricevuto.
- `send(String chatID, String message)`: Funzione per inviare un messaggio di testo al chatId specificato utilizzando l'API di Telegram.
- Le librerie utilizzate nel codice sono:
  - `WiFi.h`: Libreria per gestire la connessione Wi-Fi.
  - `HTTPClient.h`: Libreria per effettuare richieste HTTP e gestire le risposte.
  - `ArduinoJson.h`: Libreria per analizzare e generare dati JSON.

Il codice controlla lo stato di una porta utilizzando due pin di input (`Pindato1` e `Pindato2`) e due pin di output (`apertura` e `chiusura`).
La variabile `vet` è un array utilizzato per tenere traccia dello stato della porta negli ultimi 10 cicli di loop. Il valore di `vet` viene convertito in una stringa e inviato come storico tramite il comando "storico" del bot Telegram.

Capitolo 4

CONCLUSIONI

4.1 Descrizione delle prove effettuate

Di seguito le prove effettuate per testare i singoli componenti e il loro funzionamento:

1. Abbiamo verificato il corretto collegamento dei dispositivi: abbiamo controllato che i dispositivi Arduino fossero collegati correttamente tra di loro e che il secondo dispositivo fosse in grado di collegarsi con l'ESP32 e con il Bot di Telegram. In caso di problemi di connessione, abbiamo controllato i cablaggi e/o i codici utilizzati per la comunicazione wireless.

2. Abbiamo verificato la tastiera numerica: abbiamo controllato che la tastiera numerica del primo dispositivo fosse in grado di ricevere e leggere i dati inseriti dall'utente in modo corretto. Abbiamo quindi verificato che i tasti funzionassero correttamente e che la password inserita venisse verificata dal codice correttamente.

3. Abbiamo verificato il sensore di prossimità: abbiamo controllato che il sensore di prossimità del secondo dispositivo fosse in grado di rilevare la presenza di oggetti o persone vicino alla porta (entro un determinato range) e di attivare l'apertura o la chiusura della stessa in modo corretto. Abbiamo quindi verificato che il sensore funzionasse correttamente e che la porta si aprisse e si chiudesse senza problemi.

4. Abbiamo verificato il display e il Buzzer: abbiamo controllato che il display del secondo dispositivo mostrasse in modo corretto lo stato della porta (aperta o chiusa) e che il Buzzer fosse in grado di attivarsi in caso di allarme (cioè che si attivasse al terzo errore consecutivo). Abbiamo quindi verificato che il codice fosse stato scritto correttamente e che i componenti funzionassero correttamente.

5. Abbiamo verificato il controllo tramite Telegram: abbiamo controllato che l'ESP32 del secondo dispositivo fosse in grado di interfacciarsi correttamente con Telegram e di ricevere i comandi dall'utente tramite lo smartphone. Abbiamo quindi verificato che il Bot fosse stato configurato correttamente e che l'interfacciamento con Telegram funzionasse in modo corretto.

4.2 Limitazioni del progetto

Di seguito riportate alcune limitazioni del progetto:

1. Problemi di alimentazione: il progetto necessita l'alimentazione di vari componenti che essendo alimentati a batteria, richiedono una ricarica o sostituzione di essa.

2. Sicurezza della password: la sicurezza della password inserita potrebbe rappresentare una possibile limitazione. Se la password è facilmente indovinabile o se venisse trasmessa in modo non sicuro, il sistema potrebbe essere facilmente violato.

3. Limitazioni di Telegram: l'utilizzo di Telegram come interfaccia di controllo potrebbe comportare alcune limitazioni, come ad esempio la necessità di una connessione internet per inviare i comandi alla porta. Inoltre, potrebbe essere necessario un ulteriore grado di sicurezza per evitare che utenti non autorizzati possano controllare la porta.

4.3 Possibili sviluppi futuri

Un possibile sviluppo futuro/miglioramento potrebbe essere l'aggiunta di un criptaggio alla connessione wireless tra i due dispositivi Arduino, l'introduzione di una password anche per l'accesso tramite Telegram, e la creazione, mediante stampa 3D, di un contenitore personalizzato per il dispositivo master e lo slavo che si adatti ad una qualsiasi serratura (chiave) utilizzata.

Questa soluzione potrebbe risultare molto utile per l'installazione e l'utilizzo all'interno di una casa, soprattutto per il controllo di porte interne come, ad esempio, le porte delle stanze o del garage. Infatti, l'utilizzo di un sistema di controllo wireless potrebbe semplificare e rendere più sicuro l'accesso a queste aree della casa.