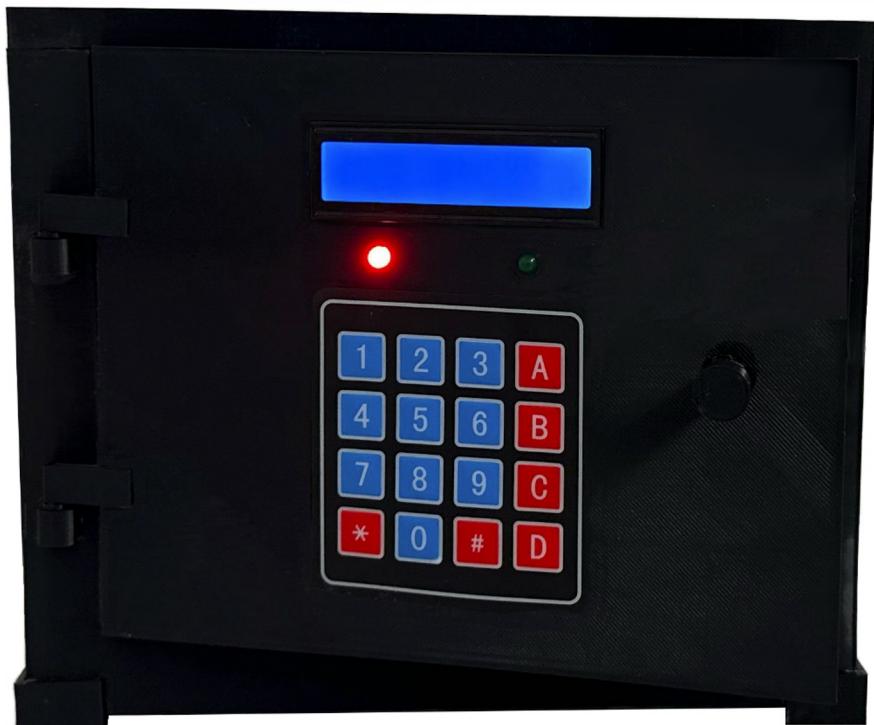


RELAZIONE

Nicolò Emilii



COGNOME PROGETTISTA: Emilii NOME PROGETTISTA: Nicolò

SCHEMA ELETTRICO

(1)

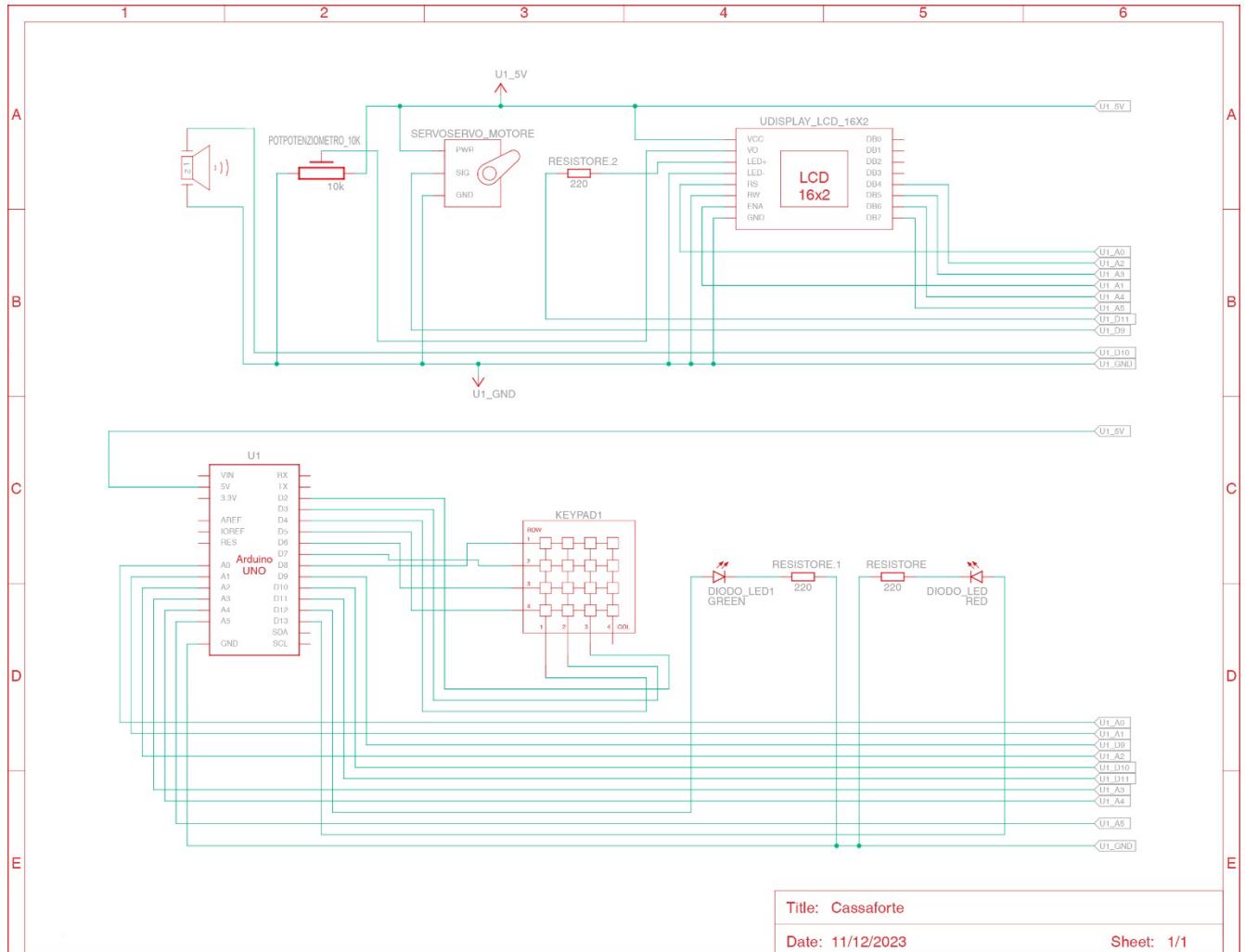


TABELLA COLLEGAMENTI

(2)

PIN	COMPONENTE	DESCRIZIONE
2	KeyPad	Colonna 3
3	KeyPad	Colonna 2
4	KeyPad	Colonna 1
5	KeyPad	Riga 4
6	KeyPad	Riga 3
7	KeyPad	Riga 2
8	KeyPad	Riga 1
9	Servo motore	Segnale
10	Buzzer	Segnale
11	Display	Gestione retroilluminazione
12	Led verde	Alimentazione Led verde
13	Led rosso	Alimentazione Led rosso
14/A0	Display	Selezione registro
15/A1	Display	Abilitazione
16/A2	Display	DB4
17/A3	Display	DB5
18/A4	Display	DB6
19/A5	Display	DB7

ELENCO COMPONENTI UTILIZZATI

(3)

- N.1 Potenziometro 10kΩ
- N.3 Resistori 220Ω
- N.1 KeyPad 4x4
- N.1 Servo motore
- N.1 Diodo LED Rosso
- N.1 Diodo LED Verde
- N.1 Display LCD 16x2
- N.1 Buzzer passivo
- N.1 Arduino UNO R3

PER LA REALIZZAZIONE INOLTRE:

- N.1 Batteria 9 V da 565 mAh
- N.1 Cavo adattatore batteria Arduino DC 9V
- N.1 Bobina 1Kg PLA nero SUNLU 1,75 precisione dimensionale +/- 0.02mm

CARATTERISTICHE STRUMENTI UTILIZZATI

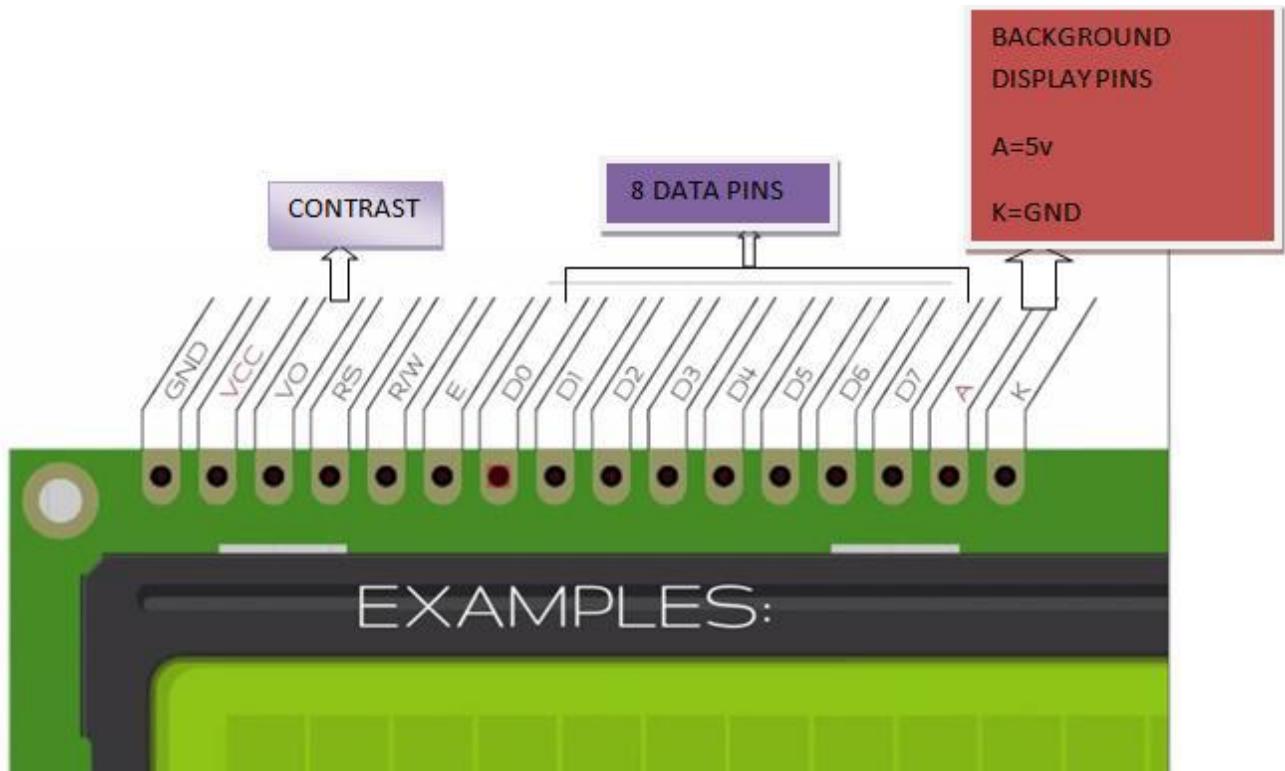
(4)

MARCA	MODELLO	ALTRO
///	///	Breadboard
Lexman	PT1000	Multimetro
Creality	ENDER 3v2	Stampante 3D

DATASHEET

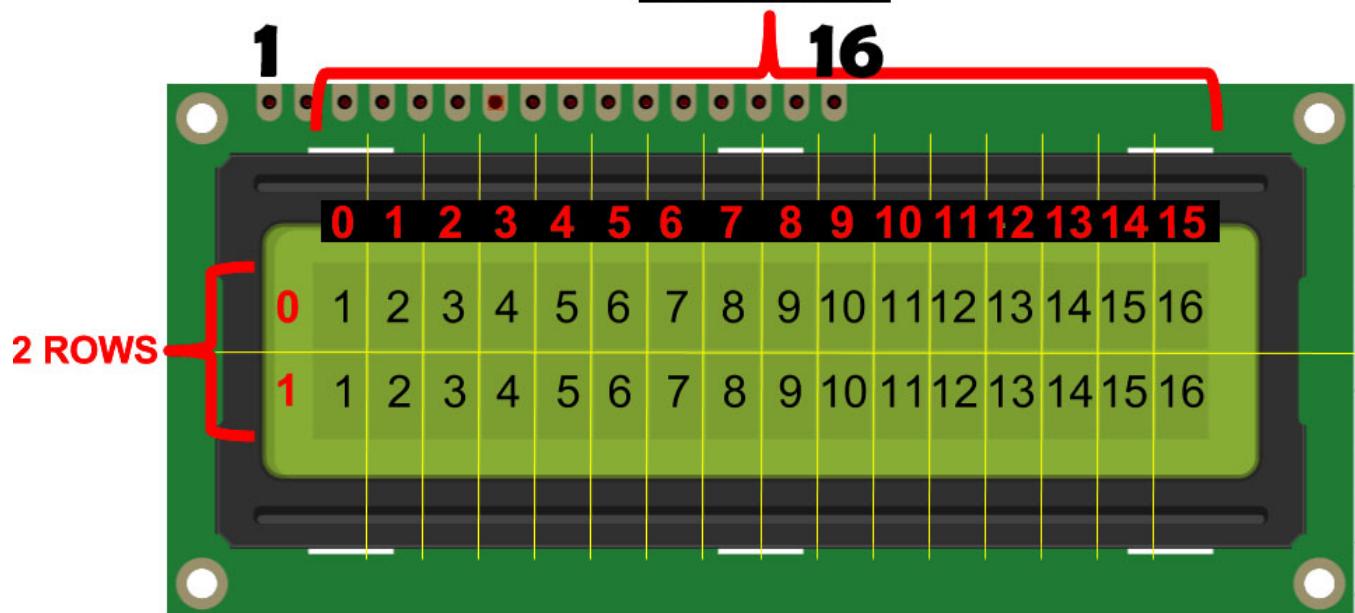
DISPLAY LCD 16x2:

(5)



(6)

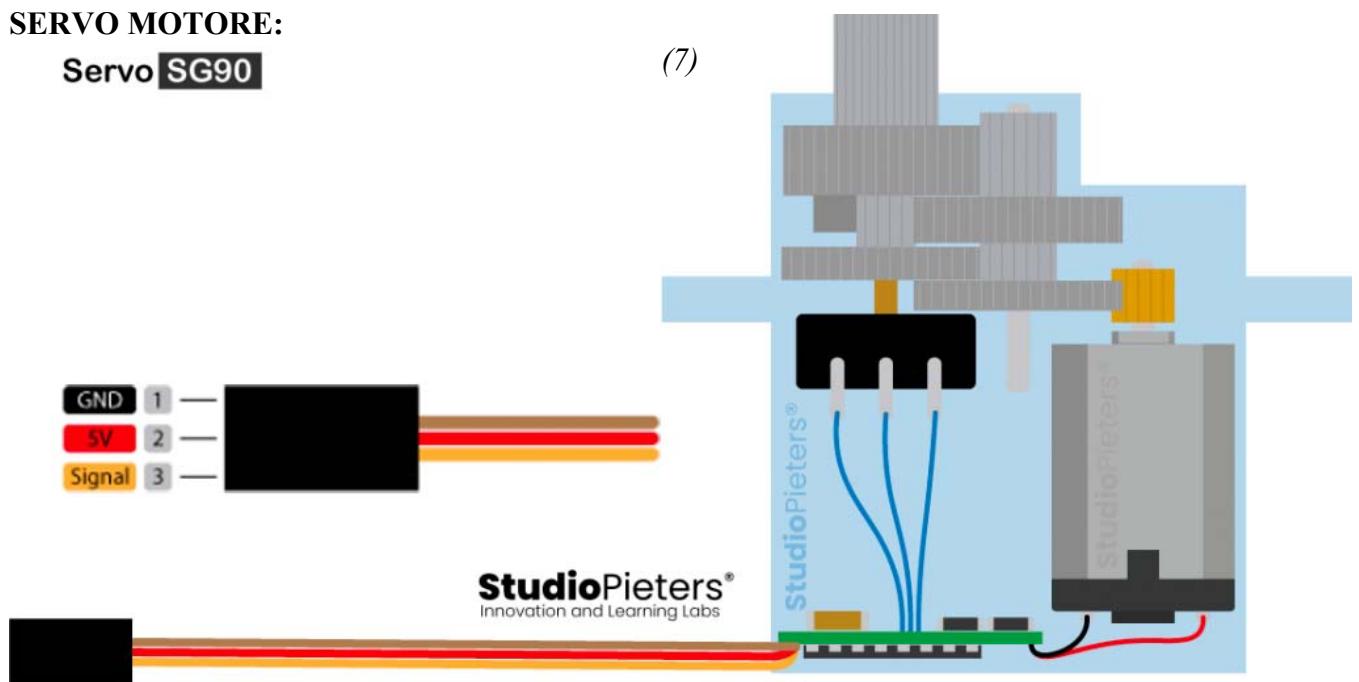
16 COLUMNS



SERVO MOTORE:

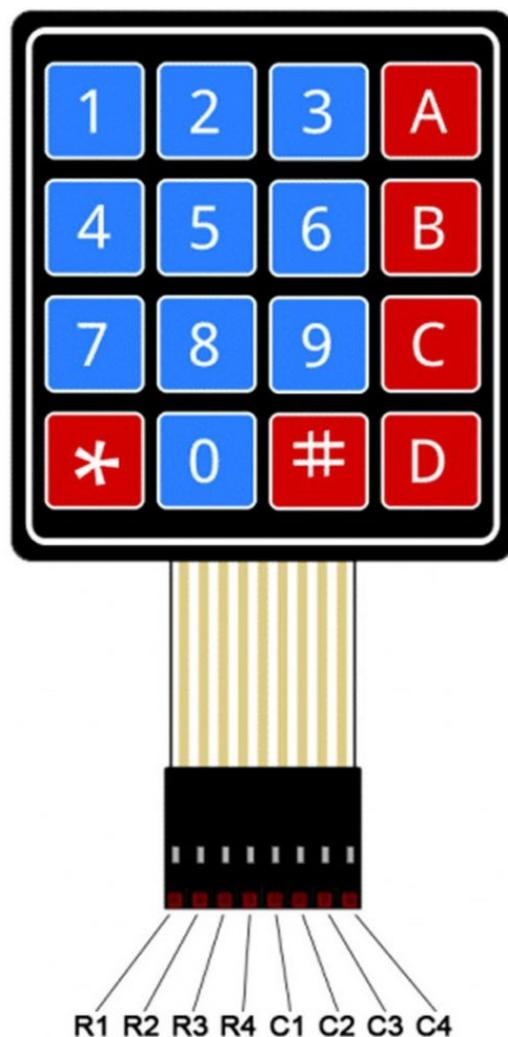
Servo SG90

(7)



KEYPAD 4X4:

(8)



PROGRAMMA

(9)

```
#include <Servo.h>
#include <Keypad.h>
#include <EEPROM.h>
#include <LiquidCrystal.h>

// Combinazione per entrare nella modalità di cambio codice
const String cambiocodice = "****";
// Combinazione per entrare nella modalità di recupero
const String recupero = "####";
// Codice di sblocco di default
String codice = "1234";
// Codice di recupero (si suppone sia univoco per ogni cassaforte)
String cod_rec = "123456789";
// Dichiaro la stringa che conterrà il codice e la lascio vuota
String cod_ins = "";

char ins[4] = {'1','2','3','4'};
char chiusura = ' ';
int err = 0;
int secondi_blocco = 30;
int sup = 0;
bool giaAvviato = EEPROM.read(4); //Controllo se già avviato (nel primo avvio è false
(con condizione EEPROM = 0/false))

// Errori dopo il quale la cassaforte entra nel primo "livello" di blocco
const int err_max = 4;
// Numero colonne KeyPad
const int colonne = 3;
// Numero righe KeyPad
const int righe = 4;
// PIN servo motore
const int servo = 9;
// PIN buzzer
const int buzzer = 10;
// PIN retroilluminazione
const int illum = 11;
// PIN LED Verde
const int verde = 12;
// PIN LED Rosso
const int rosso = 13;

//----- Inizializzazione tastiera

// Creo la matrice per assegnare ad ogni pulsante sul KeyPad un carattere
char keys[righe][colonne] = {
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
```

```

        {'*', '0', '#'},
};

byte pinrighe[righe] = {8, 7, 6, 5};
byte pincolonne[colonne] = {4, 3, 2};
Keypad tastiera = Keypad(makeKeymap(keys), pinrighe, pincolonne, righe, colonne);
//----- Assegnazione pin display
const int rs = 14, en = 15, d4 = 16, d5 = 17, d6 = 18, d7 = 19;
LiquidCrystal lcd (rs, en, d4, d5, d6, d7);
//----- Inizializzazione servo
Servo myservo;
// 80° aperto
// 0° chiuso
// Creo la grafica del lucchetto chiuso
byte lucchettoChiuso[8] = {
    0b01110,
    0b10001,
    0b10001,
    0b10001,
    0b11111,
    0b11011,
    0b11011,
    0b11111
};
// Creo la grafica del lucchetto aperto
byte lucchettoAperto[8] = {
    0b01110,
    0b10000,
    0b10000,
    0b10000,
    0b11111,
    0b11011,
    0b11011,
    0b11111
};

//----- SETUP
void setup() {
    // Apro comunicazione seriale alla velocita di 9600 baud
    Serial.begin(9600);

    // Inizializzo il Display LCD dicendo che ha 16 colonne e 2 righe

    lcd.begin(16, 2);
    lcd.setCursor(0, 0);
    lcd.clear();

    myservo.attach(servo);
    // Chiudo la porta
    myservo.write(0);
}

```

```

// Imposto i pin in uscita
pinMode(verde, OUTPUT);
pinMode(rosso, OUTPUT);
pinMode(illum, OUTPUT);

lcd.createChar(0, lucchettoChiuso);
lcd.createChar(1, lucchettoAperto);
if(giaAvviato){
    // Scrive solo al primo avvio il PIN dalla RAM, in quelli successivi lo legge
    // dalla EEPROM

    eeprom_pin(0);
}else{
    eeprom_pin(1);
    giaAvviato = true;
    EEPROM.write(4, giaAvviato);
}
}

void loop() {
    INIZIO:

    // Svuoto la variabile che contiene il codice inserito, pulisco il Display LCD
    // e stampo il lucchetto
    cod_ins = "";
    lcd.clear();
    lucchetto();
    lcd.setCursor(0,0);

    // Inserimento PIN
    lcd.print("Inserisci PIN:");
    Serial.println("Inserisci PIN:");
    lcd.setCursor(0,1);
    // Richiamo la funzione per il risparmio energetico
    attesa();
    inserimento_cod();

    if(cod_ins == codice){
        // Codice esatto
        lcd.setCursor(0,1);
        lcd.print("Codice esatto");
        Serial.println("Codice esatto");
        delay(500);
        lcd.clear();
        lucchetto();

        // Gestisco porta e azzero il conteggio degli errori
        err = 0;
        lcd.setCursor(0,0);
    }
}

```

```

lcd.print("Premi #");
lcd.setCursor(0,1);
lcd.print("per chiudere");
Serial.println("Premi # per chiudere");
// Apro porta
myservo.write(80);
lucchetto();

RIMANI:
chiusura = ' ';
// Richiamo la funzione per il risparmio energetico
attesa();
// Attendo che venga premuto un pulsante
chiusura = tastiera.waitForKey();
// Accendo la retroilluminazione del Display LCD
digitalWrite(illum, HIGH);
// Emetto un suono con il buzzer
tone(buzzer,800,70);
delay(70);

if(chiusura == '#'){
    // Se il pulsante che è stato premuto è il "#" la porta si chiude
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("Chiusura...");
    Serial.println("Chiusura");
    lucchetto();

    // Chiudo porta
    myservo.write(0);
    delay(1000);
} else{
    // Se il pulsante premuto non è il "#" torno all'etichetta
    "RIMANI"
    goto RIMANI;
}

cod_ins = "";
else if(cod_ins == cambiocodice){
    //----- Cambio codice
    // Se il codice inserito è ****
    lcd.clear();
    lucchetto();

    lcd.setCursor(0,0);
    lcd.print("Cambio codice");
    Serial.println("Cambio codice");
    delay(1000);
    lcd.clear();
}

```

```

lcd.print("Ins. PIN prec:");
Serial.println("Ins. PIN prec:");
lucchetto();
lcd.setCursor(0,1);
cod_ins = "";

// Inserimento PIN precedente
inserimento_cod();

if(cod_ins == codice){
    // Se il codice inserito è giusto permetto il cambio del codice
    lcd.setCursor(0,0);
    lcd.print("Codice esatto");
    Serial.println("Codice esatto");
    delay(500);
    cod_ins = "";
}else{
    // Se il codice inserito non è quello giusto non viene permesso
    // il cambio codice
    lcd.clear();
    lucchetto();

    // Segnalo scrivendo sul Display LCD e sul monitor seriale
    "Codice errato"
    lcd.setCursor(0,1);
    lcd.print("Codice errato");
    Serial.println("Codice errato");
    suon_err();
    err = err + 1;

    // Verifico se è stato superato il limite di errori per il
    // quale la cassaforte si deve bloccare
    if(err >= err_max){
        blocco();
    }

    delay(500);
    goto INIZIO;
}

lcd.clear();
lucchetto();

// Dato che il controllo precedente ha avuto esiti positivi si può
// cambiare il PIN di sblocco
lcd.setCursor(0,0);
lcd.print("Ins. nuovo PIN:");
Serial.println("Ins. nuovo PIN:");
lcd.setCursor(0,1);

// Inserisco il nuovo PIN

```

```

inserimento_cod();

if(cod_ins == cambiocodice || cod_ins == recupero){

    // Se il nuovo PIN inserito è uguale alla combinazione di
    // recupero PIN o al cambio codice, il PIN non viene cambiato
    lcd.clear();
    lucchetto();
    lcd.setCursor(0,0);
    lcd.print("PIN non");
    lcd.setCursor(0,1);
    lcd.print("inseribile");
    Serial.println("Codice non inseribile (il codice non può essere
    come la password di cambio password)");
    suon_err();
    delay(1000);
}else{

    // Il nuovo PIN viene scritto nella EEPROM e la variabile
    // "codice" viene sovrascritta
    codice = cod_ins;
    //----- Scrittura PIN in EEPROM
    eeprom_pin(1);
    lcd.setCursor(0,0);
    lcd.clear();
    lucchetto();

    lcd.print("Password cambiata");
    Serial.println("Password cambiata");
    delay(1000);
}
//----- Recupero codice
else if(cod_ins == recupero){

    // Se il codice inserito è la combinazione per il recupero del PIN
    // viene avviata la procedura di recupero richiamando la funzione "rec"
    rec();
}else{

    // Altrimenti il codice inserito non è valido e viene segnalato dal
    // Display LCD, dal monitor seriale, dal Diodo LED rosso e dal buzzer
    lcd.clear();
    lucchetto();
    lcd.setCursor(0,1);
    Serial.println("Codice errato");
    lcd.print("Codice errato");
    suon_err();
    delay(500);
    err = err + 1;
    cod_ins = "";
    delay(1000);
}

```

```

        // Verifico se è stato superato il limite di errori per il quale la
        // cassaforte si deve bloccare
        if(err >= err_max){
            blocco();
        }

        // Torno all'etichetta "INIZIO"
        goto INIZIO;
    }

void lucchetto(){
    // Stampo sul Display LCD 16x2 in basso a destra un lucchetto apero o chiuso in
    // base alla posizione del servo motore
    // In base alla posizione del servo motore decido anche che diodo LED accendere
    // Diodo LED Rosso acceso --> Cassaforte bloccata
    // Diodo LED Verde acceso --> Cassaforte sbloccata
    if(myservo.read() == 0){
        lcd.setCursor(15,1);
        lcd.write(byte(0));
        digitalWrite(verde, LOW);
        digitalWrite(rosso, HIGH);
    }else if(myservo.read() == 80){
        lcd.setCursor(15,1);
        lcd.write(byte(1));
        digitalWrite(rosso, LOW);
        digitalWrite(verde, HIGH);
    }
    return;
}

void blocco(){
    // Gestisco il numero di errori e il tempo di blocco della cassaforte
    if(err == 4){
        sup = 0;
    }else if(err < 6){
        sup = 30;
    }else if(err < 8){
        sup = 60;
    }else if(err < 10){
        sup = 90;
    }else if(err > 9){
        sup = 270;
    }
    int tem = secondi_blocco + sup;
    for(int i = tem ; i > 0; i-- ){

        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print(err);
    }
}

```

```

    lcd.setCursor(2,0);
    lcd.print(" Errori");
    lcd.setCursor(0,1);
    lcd.print("Riprova tra:");
    lcd.print(i);
    Serial.println(i);
    lucchetto();
    delay(1000);

}

return;
}

void suon_err(){
    // Riproduco un suono di errore con il buzzer
    // Faccio lampeggiare il diodo LED Rosso
    delay(200);
    tone(buzzer,100,1000);
    digitalWrite(verde, LOW);
    for(int i = 3; i > 0; i--){
        digitalWrite(rosso, LOW);
        delay(180);
        digitalWrite(rosso, HIGH);
        delay(120);
    }
}

void inserimento_cod(){
    // Inserisco il codice e lo salvo nella variabile cod_ins
    for(int i=0; i<4; i++){
        ins[i] = tastiera.waitForKey();
        digitalWrite(illum, HIGH);
        tone(buzzer,800,70);
        delay(70);
        lcd.print("*");

        cod_ins = cod_ins + ins[i];
    }
}

void attesa(){
    // Gestisco il risparmio energetico (spengo la retroilluminazione del Display LCD)
    digitalWrite(illum, LOW);
}

void rec(){
    // Gestisco il recupero PIN facendo inserire il PIN di recupero

    lcd.clear();
    lucchetto();
}

```

```

lcd.setCursor(0,0);
lcd.print("Recupero PIN");
Serial.println("Recupero PIN");
delay(1000);

lcd.clear();
lucchetto();
lcd.setCursor(0,0);
lcd.print("Ins. PIN di rec.");
Serial.println("Inserisci PIN di recupero");
lcd.setCursor(0,1);
cod_ins = "";

// Inserimento PIN a 9 cifre
for(int i=9; i>0; i--){
    char ins = tastiera.waitForKey();
    digitalWrite(illum, HIGH);
    tone(buzzer,800,70);
    delay(70);
    lcd.print("*");

    cod_ins = cod_ins + ins;
}

if(cod_ins == cod_rec){
    lcd.clear();
    lucchetto();
    lcd.setCursor(0,0);
    lcd.print("Nuovo PIN:");
    lcd.setCursor(0,1);
    lcd.print("1234");
    Serial.println("Nuovo PIN --> 1234");
    codice = "1234";
    ins[0] = '1';
    ins[1] = '2';
    ins[2] = '3';
    ins[3] = '4';
    // Salvo il "nuovo" PIN nella EEPROM
    eeprom_pin(1);
}else{
    // Se il PIN a 9 cifre è errato lo segnalo
    lcd.clear();
    lucchetto();
    lcd.setCursor(0,0);
    lcd.print("PIN errato");
    Serial.println("PIN di recupero errato");
}
delay(5000);
}

```

```

void eeprom_pin(int write){
    /*ASSEGNAZIONE CELLE EEPROM
    0 - PIN_CHAR_TO_INT_0
    1 - PIN_CHAR_TO_INT_1
    2 - PIN_CHAR_TO_INT_2
    3 - PIN_CHAR_TO_INT_3
    4 - FLAG_FIRST_START*/
    String tempPin = "";
    switch(write){
        case 0:{    //leggo il PIN salvato nella EEPROM
            for(int i=0; i<4; i++){
                int intPin = EEPROM.read(i);
                tempPin += char(intPin);
            }
            codice = tempPin;
            break;
        }
        case 1:{    //scrivo il PIN nella EEPROM
            for(int i=0; i<4; i++){
                EEPROM.write(i, ins[i]);
            }
            break;
        }
        default:{   //se il parametro "write" manca non fa nulla
            break;
        }
    }
}

```

DATI RACCOLTI

(10)

Quando		Corrente misurata (mAh)
1	Retroilluminazione Display LCD spenta, LED Verde acceso, porta aperta	54,4
2	Retroilluminazione Display LCD spenta, LED Rosso acceso, porta chiusa	60,0
3	Retroilluminazione Display LCD accesa, LED Rosso acceso, porta chiusa	66,0
4	Retroilluminazione Display LCD accesa, LED Rosso acceso, porta chiusa, suono errore buzzer	71,7
5	Retroilluminazione Display LCD accesa, LED Rosso acceso, porta chiusa, suono buzzer pressione pulsante	67,0
6	Retroilluminazione Display LCD accesa, LED Verde acceso, rotazione servo motore	200,0

 *Risparmio energetico non attivo*

 *Risparmio energetico attivo*

CALCOLI

(11)



Capacità batteria utilizzata = 565 mAh

- **Durata batteria nella situazione descritta al punto 1 della tabella 10**

$$\text{Durata batteria standby LED Verde acceso} = \frac{565 \text{ mAh}}{54.4 \text{ mA}} = 10.39h$$

- **Durata batteria nella situazione descritta al punto 2 della tabella 10**

$$\text{Durata batteria standby LED Rosso acceso} = \frac{565 \text{ mAh}}{60 \text{ mA}} = 9.42h$$

- **Durata batteria nella situazione descritta al punto 3 della tabella 10**

$$\text{Durata batteria con retroilluminazione display accesa} = \frac{565 \text{ mAh}}{66 \text{ mA}} = 8.56h$$

- **Durata batteria nella situazione descritta al punto 4 della tabella 10**

$$\text{Durata batteria con retroilluminazione display accesa e suono errore buzzer} = \frac{565 \text{ mAh}}{71.7 \text{ mA}} = 8.28h$$

- **Durata batteria nella situazione descritta al punto 5 della tabella 10**

$$\text{Durata batteria con retroilluminazione display accesa e suono KeyPad} = \frac{565 \text{ mAh}}{67 \text{ mA}} = 8.43h$$

- **Durata batteria nella situazione descritta al punto 6 della tabella 10**

$$\text{Durata batteria con retroilluminazione display accesa e rotazione servo motore} = \frac{565 \text{ mAh}}{200 \text{ mA}} = 3.23h$$

RELAZIONE

Il seguente progetto ha lo scopo di realizzare una cassaforte che abbia le seguenti caratteristiche:

- Display per la visualizzazione di eventuali istruzioni, per restituire lo stato della cassaforte e per fornire un feedback alla pressione del tasto durante l'inserimento del codice.
- Sul Display ad ogni pressione di un tasto deve apparire “*”.
- Deve esserci un buzzer che segnali la pressione di un tasto o eventuali segnali di errore in caso di codice errato.
- Un LED **Rosso** che segnali lo stato di porta chiusa e che lampeggi in caso di codice errato.
- Un LED **Verde** che segnali lo stato di porta aperta.
- Il codice “****” deve permettere il cambio del PIN.
- Il codice “#####” deve permettere il ripristino del PIN alle impostazioni di fabbrica “1234”.
- Deve bloccare la cassaforte in caso di codice errato consecutivamente per un numero definito di volte.
- Deve ricordare, in caso la cassaforte perda alimentazione, il PIN, nell'eventualità fosse stato cambiato.
- Deve avere una funzione di risparmio energetico in modo tale da preservare la batteria.



LIBRERIE

SERVO.H:

La libreria < Servo.h > è una libreria software per Arduino che consente di controllare i servomotori. I servomotori sono motori con un angolo di rotazione preciso che possono essere utilizzati per vari scopi.

La libreria < Servo.h > fornisce una serie di funzioni per controllare i servomotori, tra cui:

- attach(): Associa un servomotore a un pin Arduino.
- write(): Imposta l'angolo di rotazione di un servomotore.
- read(): Legge l'angolo di rotazione corrente di un servomotore.
- detach(): Disassocia un servomotore da un pin Arduino.



KEYPAD.H:

La libreria < Keypad.h > è una libreria software per Arduino che consente di utilizzare tastierini matriciali. I tastierini matriciali sono dispositivi che consentono di inserire dati tramite una matrice di tasti.

La libreria < Keypad.h > fornisce una serie di funzioni per utilizzare i tastierini matriciali, tra cui:

- begin(): Inizializza il tastierino matriciale.
- getKey(): Legge il valore del tasto premuto.
- isKeyPressed(): Controlla se è stato premuto un tasto.
- waitForKey(): Attende che venga premuto un tasto.



EEPROM.H:

La libreria < EEPROM.h > è una libreria software per Arduino che consente di accedere alla memoria EEPROM dell'Arduino. La memoria EEPROM è una memoria non volatile, il che significa che i dati memorizzati in essa vengono mantenuti anche quando l'Arduino è spento.

La libreria < EEPROM.h > fornisce una serie di funzioni per accedere alla memoria EEPROM, tra cui:

- write(): Scrive un valore nella memoria EEPROM.
- read(): Legge un valore dalla memoria EEPROM.

- `clear()`: Cancella l'intera memoria EEPROM.
- `begin()`: Inizializza la memoria EEPROM.



LIQUIDCRYSTAL.H:

La libreria <LiquidCrystal.h> è una libreria software per Arduino che consente di controllare i display LCD basati sul chipset Hitachi HD44780 (o compatibile).

La libreria <LiquidCrystal.h> fornisce una serie di funzioni per controllare i display LCD, tra cui:

- `begin()`: Inizializza il display LCD.
- `clear()`: Cancella il display LCD.
- `home()`: Riporta il cursore all'inizio del display LCD (0,0).
- `print()`: Visualizza del testo sul display LCD.
- `write()`: Visualizza un carattere sul display LCD.

Il funzionamento della cassaforte è il seguente:

- All'accensione viene visualizzato sul Display LCD "Inserisci PIN".
- Il programma verifica se è la prima accensione. Se è la prima, carica nella EEPROM il PIN di fabbrica, se non è la prima richama il PIN salvato nella EEPROM così da poterlo sostituire al PIN di fabbrica nella verifica del PIN corretto.
- Dopo aver inserito il PIN il programma lo confronta con il PIN corretto.
- Nel caso in cui il PIN inserito sia uguale al PIN corretto, la cassaforte si apre e segnala lo sblocco accendendo il Diodo LED Verde. Dopo aver aperto la porta attende che l'utente prema il tasto "#" così da poter chiudere la porta.
- Nel caso in cui il PIN inserito non sia uguale al PIN corretto, ma sia uguale al codice di cambio PIN, il programma entra nella modalità cambio PIN.
- Viene fatto inserire prima il PIN corretto e dopo averne valutato la correttezza permette all'utente di inserire il nuovo PIN, che non può essere uguale al codice di cambio PIN o al codice di recupero. Se il PIN è conforme viene salvato nella EEPROM.
- Nell'eventualità in cui il PIN inserito non sia uguale né al PIN corretto né al codice di cambio PIN viene confrontato con il codice di recupero.
- Se il PIN inserito è uguale al codice di recupero si entra nella modalità di recupero PIN.
- Viene richiesto all'utente di inserire il codice univoco di 9 cifre fornito con la cassaforte.
- Se l'utente inserisce il codice univoco corretto, il PIN viene resettato a quello di fabbrica cioè "1234", se invece è errato si torna alla modalità classica in cui viene richiesto di inserire il PIN corretto.
- Se il PIN inserito non è né uguale al PIN corretto, né al codice di cambio PIN, né al codice di recupero, vuol dire che il PIN inserito è errato.
- Se il PIN è errato viene segnalato dalla scritta sul Display, dal buzzer che riproduce un segnale di errore e dal Diodo LED Rosso che lampeggerà.

Per rendere il programma più sintetico e leggibile, sono state create delle funzioni per delle azioni che vengono ripetute in più punti del codice.

Le funzioni che sono state create sono:

- `lucchetto` (Richiamata 16 volte) (La funzione occupa 18 righe).
- `blocco` (Richiamata 2 volte) (La funzione occupa 31 righe).
- `suon_err` (suono di errore) (Richiamata 3 volte) (La funzione occupa 13 righe).
- `inserimento_cod` (inserimento codice) (Richiamata 3 volte) (La funzione occupa 12 righe).
- `attesa` (Richiamata 2 volte) (La funzione occupa 4 righe).

- rec (recupero) (Richiamata 1 volta) (La funzione occupa 56 righe).
- eeprom_pin (Richiamata 4 volte) (La funzione occupa 27 righe).

Il programma completo senza la parte dichiarativa delle funzioni è lungo circa 300 righe.

Il codice è lungo 468 righe. Avendo utilizzato delle funzioni, il codice è stato abbreviato. Se non fossero state utilizzate, le righe sarebbero state circa 897. Grazie quindi alle funzioni il codice è stato abbreviato circa del 48%.

È stata introdotta una modalità di risparmio energetico così da preservare la batteria. La modalità di risparmio energetico viene attivata quando si attende l'inserimento di un carattere mediante il KeyPad.

Questa modalità ci permette di “recuperare” notevole tempo, rispetto alla modalità senza risparmio energetico. I calcoli sulla durata **teorica** della batteria, sono visibili visionando l'allegato (11).

Visionando l'allegato (11) (*punto 2 e 3*) si può notare come spegnendo la retroilluminazione del Display si riesca ad aumentare la durata della batteria di 46 minuti, ovvero circa 1'8,7%.

Si può notare grazie all'allegato (10) (*punto 4 e 5*) quanto consumi il buzzer mentre riproduce il segnale di errore e mentre riproduce il suono ad ogni pressione di un pulsante sul KeyPad.

Durante il segnale di errore consuma circa 4,7 mAh e durante il suono alla pressione di un tasto, circa 1 mAh. Si può notare che la riproduzione di suoni con il buzzer non causa una grande perdita sotto il punto di vista dell'autonomia.

Il componente che comporta il consumo maggiore di corrente è il servo motore (*punto 6 dell'allegato(10)*). Il suo consumo si aggira attorno ai 134 mAh. Sebbene questo dispositivo assorba molta corrente, non è un grande problema sotto il punto di vista dell'autonomia, dato che il servo motore verrà azionato solo in caso di apertura della cassaforte, che si ipotizza non avvenga in maniera ripetitiva.

PROBLEMI RISCONTRATI E SOLUZIONI

1. Insufficienza di pin digitali su Arduino UNO R3.

- *Arduino UNO R3 dispone di 20 pin digitali, 6 dei quali possono essere utilizzati anche come pin analogici. Per risolvere il problema dell'insufficienza di pin è bastato richiamare i pin 14-20 invece che i pin A0-A5.*

2. Mancanza del KeyPad 4x3 sia sul programma di simulazione Tinkercad, sia nel cablaggio.

- *Per ovviare a questo problema è stato utilizzato un KeyPad 4x4, lasciando scollegata l'ultima colonna visibile nell'allegato (8) contrassegnata con la lettera C4.*

3. La libreria Keypad.h non è disponibile di default in Arduino.

- *Per renderla disponibile basta caricarla facendo: “Sketch” > “Includi Libreria” > “Aggiungi libreria da file .ZIP...” e inserendo il file .ZIP che può essere scaricato andando su www.arduinolibraries.info/libraries/keypad.*

4. Il programma è ciclico, quindi se non si utilizzano interrupt la pressione del tasto sul KeyPad potrebbe non essere rilevata.

- *Per risolvere questo problema si può utilizzare la funzione waitForKey() della libreria Keypad.h che permette di bloccare l'esecuzione del programma finché non viene premuto un tasto del KeyPad.*

5. Il servo motore assorbe troppa corrente.

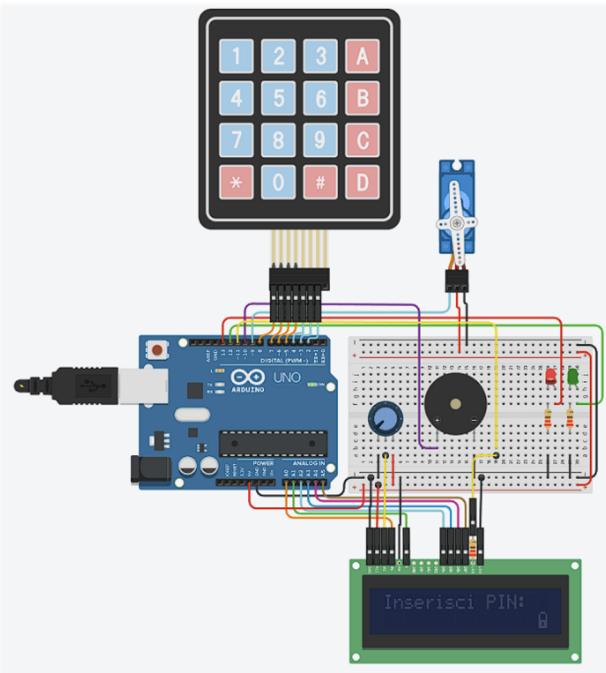
- Fino ad un massimo di 2 servo motori l'alimentazione può esser data direttamente da Arduino, se si supera però questa quantità, ci sarà bisogno di aggiungere un'alimentazione esterna.

6. Se viene cambiato il PIN e la cassaforte perde alimentazione, il codice che era stato cambiato, alla nuova accensione, sarà resettato a quello di fabbrica.

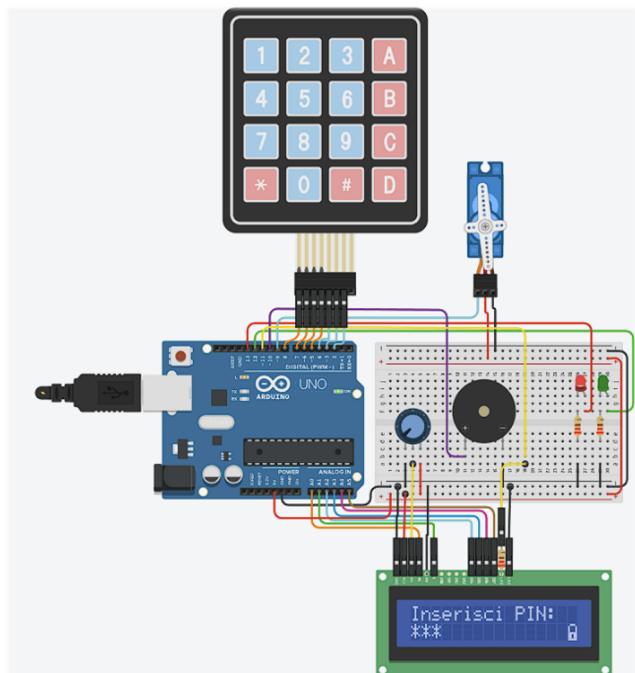
- Per mantenere in memoria il PIN cambiato si può utilizzare la libreria EEPROM.h precedentemente spiegata.

IMMAGINI

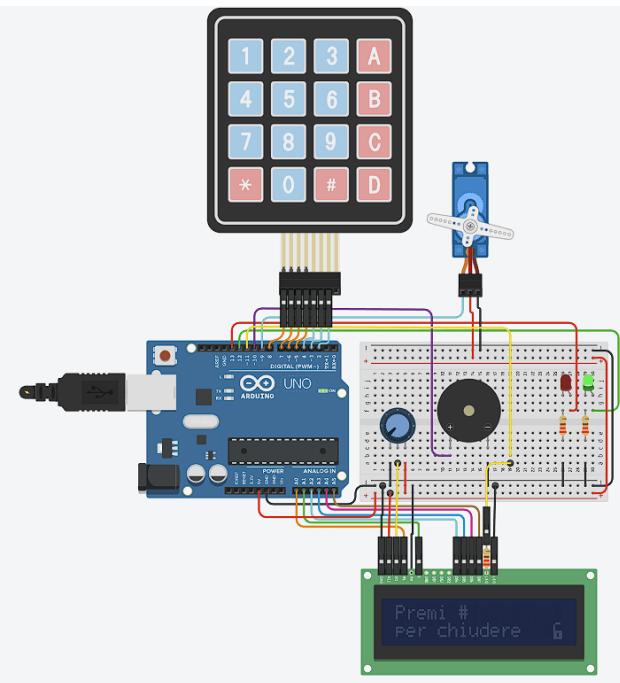
(12)



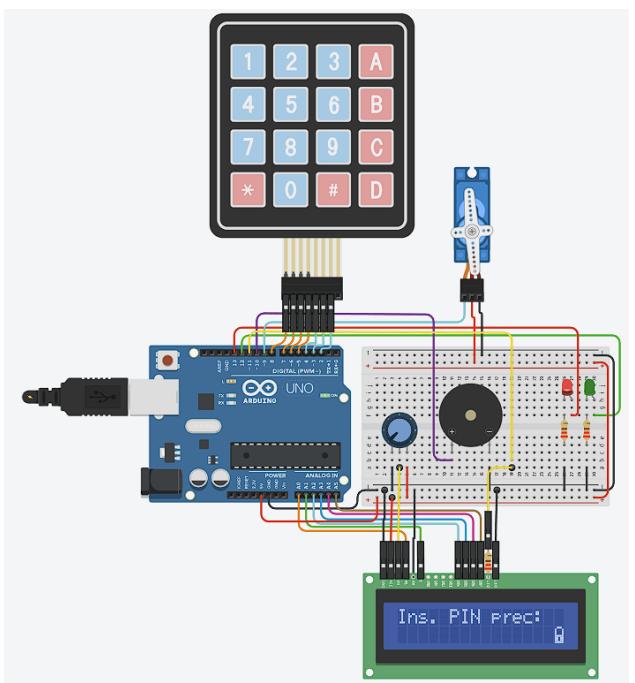
(13)



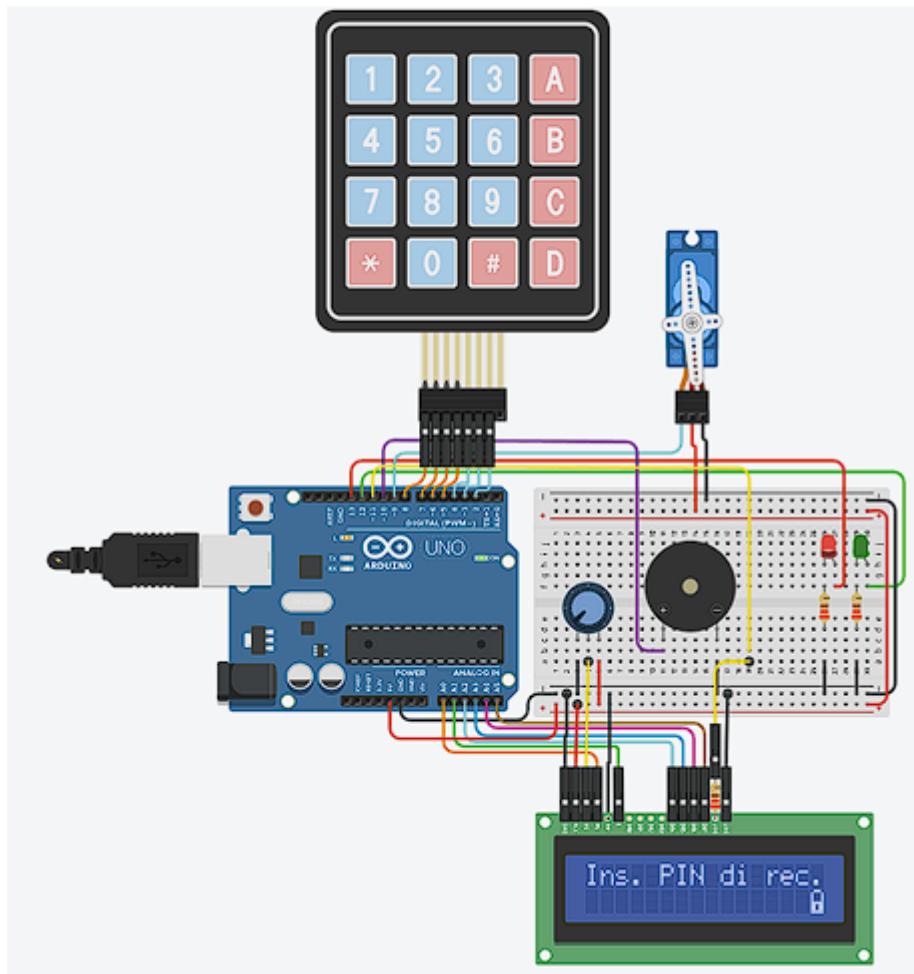
(14)



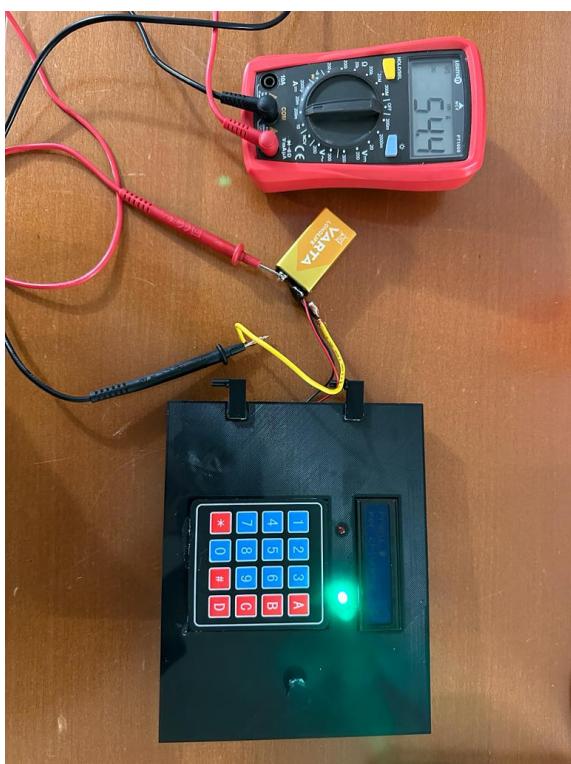
(15)



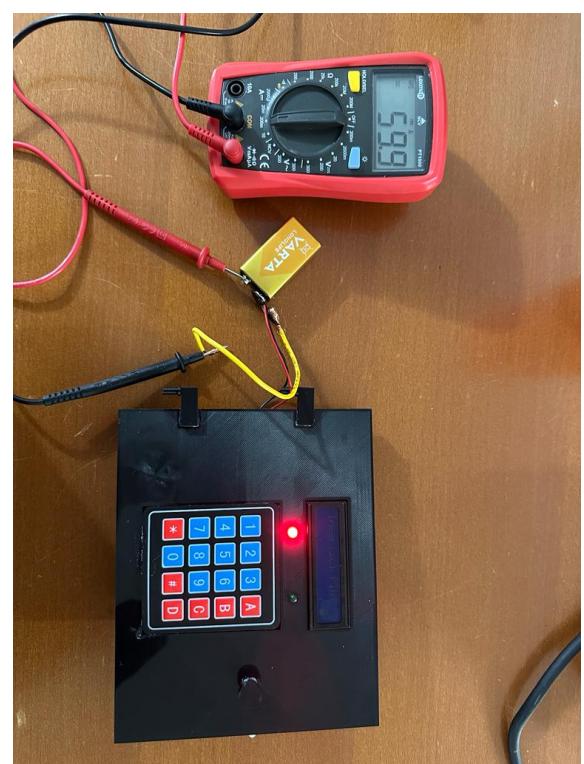
(16)



(17)



(18)



(19)



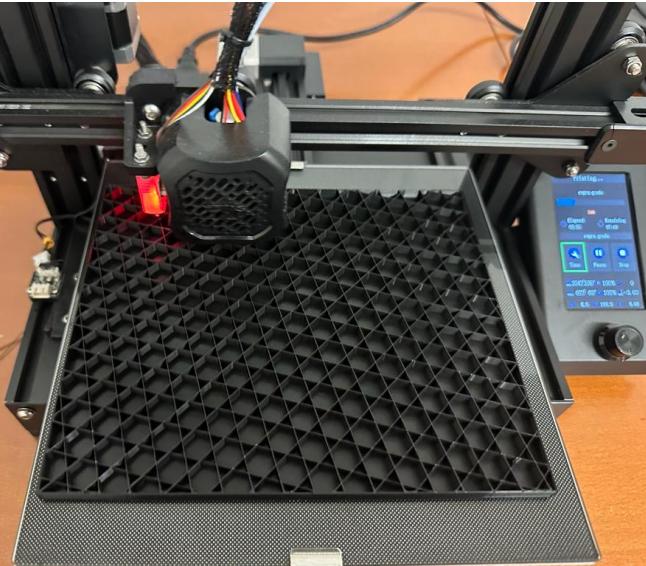
(20)



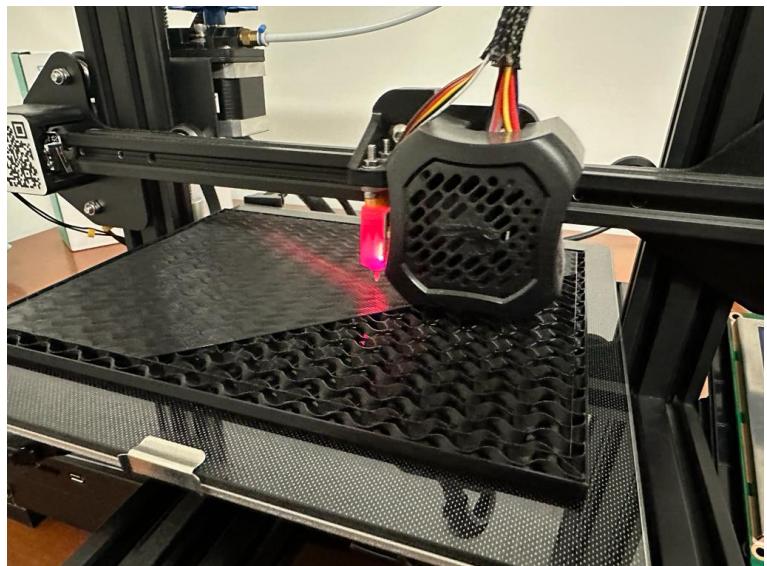
PROTOTIPO

REALIZZAZIONE:

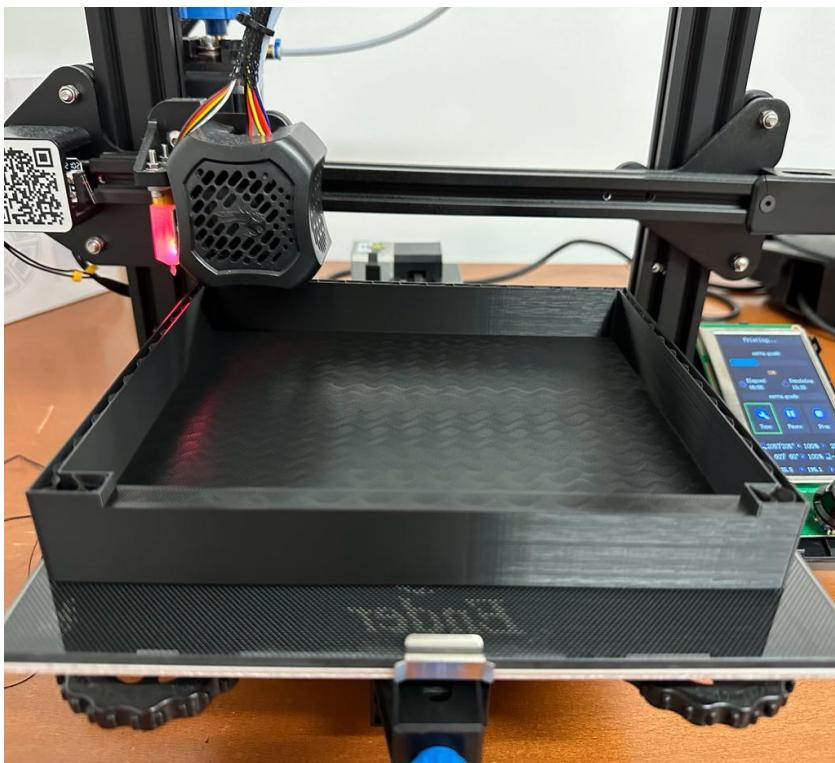
(21)



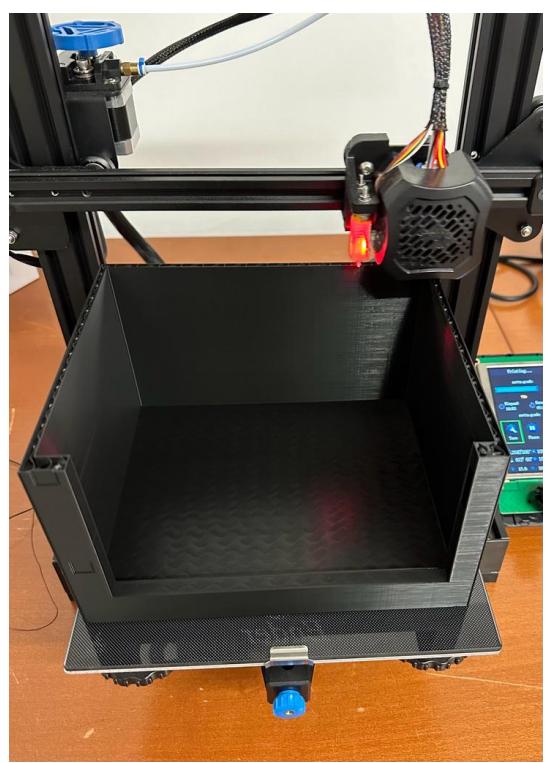
(22)



(23)



(24)



(25)



(26)



(27)





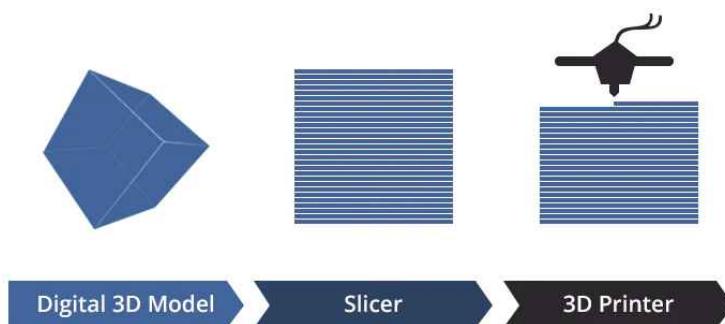
STIMA TEMPI E MATERIALI STAMPA:

Per realizzare la struttura della cassaforte è stato prima fatto il disegno su Fusion 360 che è un software CAD sviluppato da Autodesk dove è possibile progettare modelli 3D, circuiti stampati e disegni 2D e 3D.

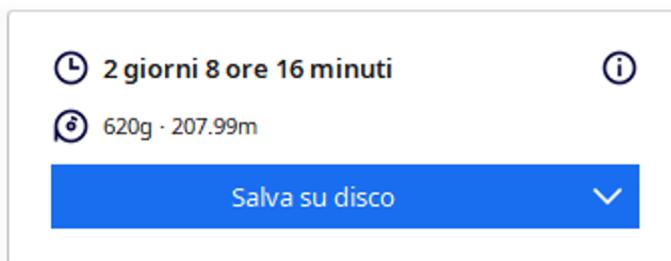
Dopo aver realizzato il disegno ed averlo esportato con l'estensione “ .3mf ”, viene elaborato dal software Ultimaker Cura, che è un software open-source per la stampa 3D e permette lo slicing del componente da stampare.

Lo slicing è il processo che trasforma un modello 3D in una serie di istruzioni che la stampante 3D può seguire per creare l'oggetto reale. In altre parole, lo slicer "taglia" il modello 3D in strati sottili, e per ognuno di questi strati genera istruzioni che specificano la posizione, la dimensione e la direzione del filamento che deve essere depositato.

Lo slicing è una fase fondamentale del processo di stampa 3D, in quanto determina la qualità e le caratteristiche dell'oggetto stampato. I parametri di slicing possono essere regolati per ottimizzare l'aspetto, la resistenza e la velocità di stampa dell'oggetto.



Ultimaker Cura fornisce anche un calcolo approssimativo del materiale che si andrà ad utilizzare, del tempo che impiegherà la stampante per stampare il pezzo e della lunghezza del filamento che verrà utilizzato.



In questo caso il calcolo di Ultimaker Cura è abbastanza affidabile dato che non si discosta di molto dal tempo effettivo che la stampante ha impiegato.

Un calcolo approssimativo sul costo del materiale in questo caso si può realizzare nel seguente modo:

$$\text{Costo filamento} = 22.99 \text{ €/Kg}$$

$$\text{Costo materiale utilizzato} = \frac{22.99}{1000} * 620 = 14.25 \text{ €}$$

$$\text{Stima filamento utilizzato} = 620 \text{ g}$$