PROGETTO ARDUINO

CONTROLLO ACCESSI COVID 19

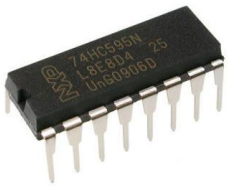
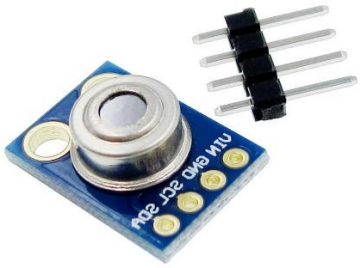
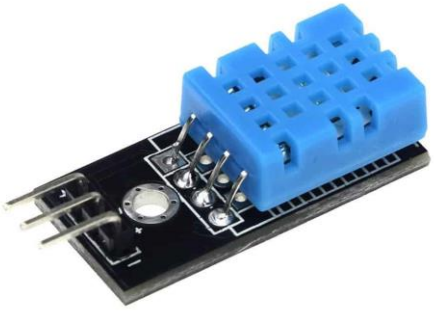
L'idea progettuale consiste in un sistema gestito tramite scheda ARDUINO, che, al verificarsi di determinate condizioni consente o meno l'accesso ad un determinato luogo.

Il funzionamento del sistema è determinato in modo da consentire l'accesso per mezzo di una scheda o badge RFID abilitato, oppure tramite l'inserimento di un PIN d'accesso. Qualora si fosse in possesso sia del PIN di accesso che di una scheda RFID che ancora non è stata abilitata, sarà possibile renderla abilitata seguendo la procedura che viene descritta tramite l'interfaccia LCD con l'utente.

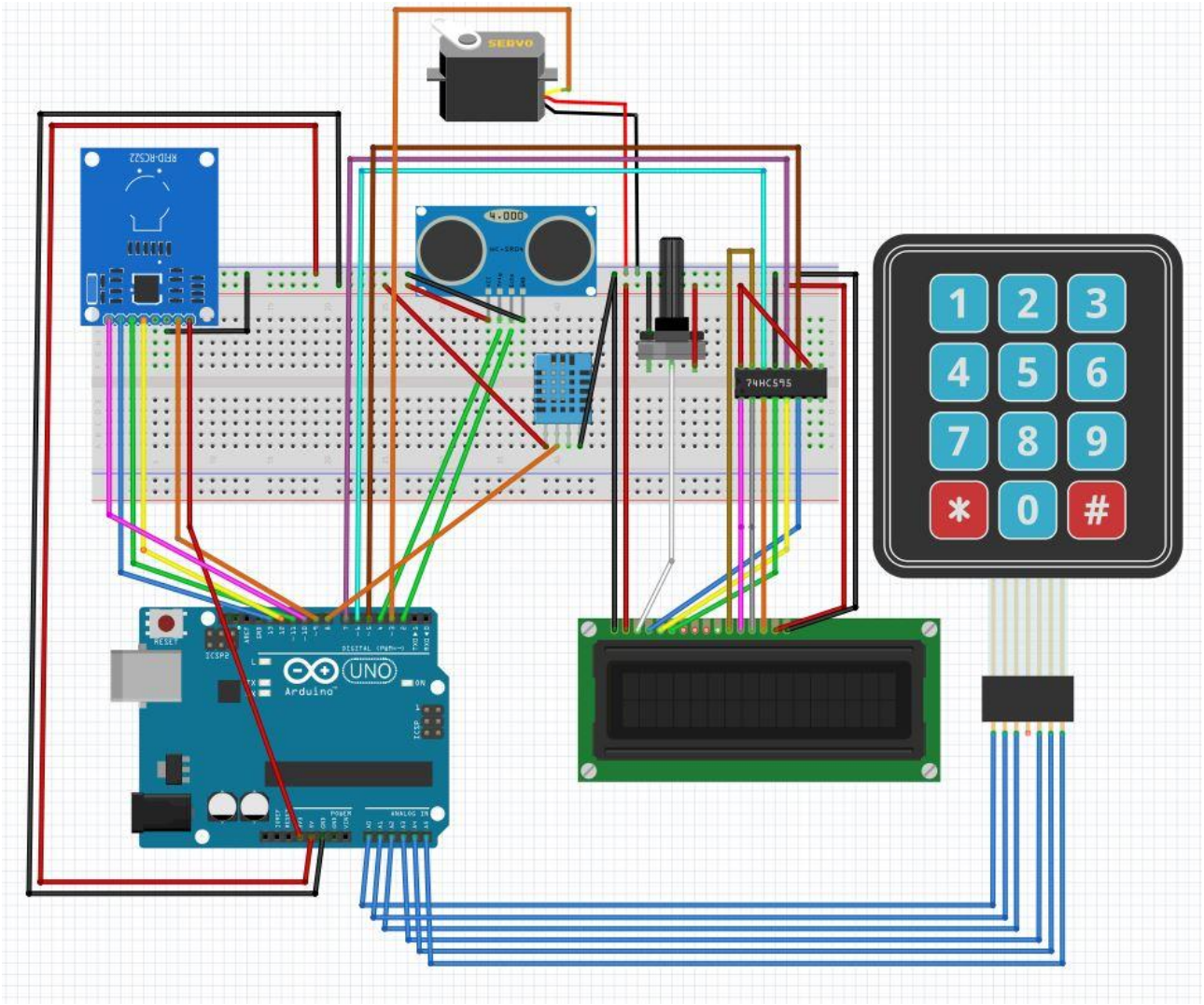
Successivamente alla verifica delle condizioni sopra elencate, vi sarà un successivo controllo gestito tramite sensore ultrasonico (per aiutare l'utente alla corretta vicinanza con il sensore) ed un sensore di rilevamento della temperatura. A tal proposito il sensore consigliato è il MLX90614 (foto in basso), ma data la difficoltà nell'ottenerlo in questo periodo, si è deciso di emularne il comportamento tramite un semplice sensore di temperatura.

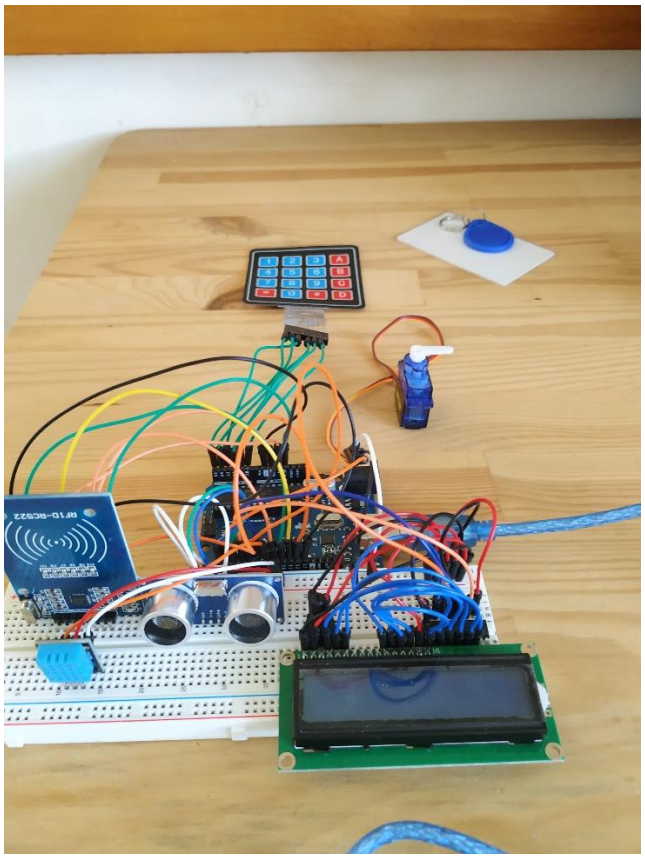
Terminate tutte le verifiche del caso, tramite un attuatore, un servo motore, simuleremo l'apertura di una serratura con un movimento di 90°.

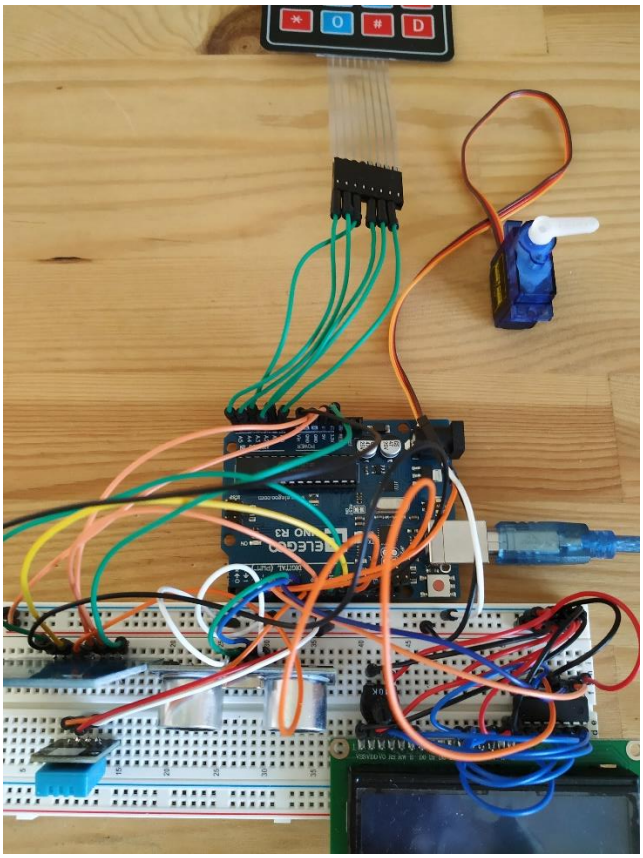
In questo progetto è stato utilizzato il seguente HARDWARE:

SCHEDA ARDUINO UNO 	LCD 16X2 
DISPOSITIVO RFID 	SENSORE ULTRASONICO 
MICRO SERVOMOTORE 	SHIFT REGISTRO 74HC595 
SENSORE TEMPERATURA MLX90614 → 	← SENSORE TEMPERATURA DHT11 

MODELLO IN FRITZING:





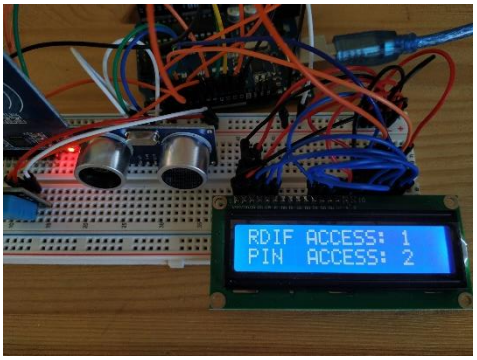


DESCRIZIONE PROGETTO:

Il sistema si interfaccia con l'utente attraverso il display LCD, pilotato tramite 3 connettori per mezzo dello shift register 74HC595, in modo da ridurre il numero di pin necessari al funzionamento.

LIQUID CRYSTAL DISPLAY:

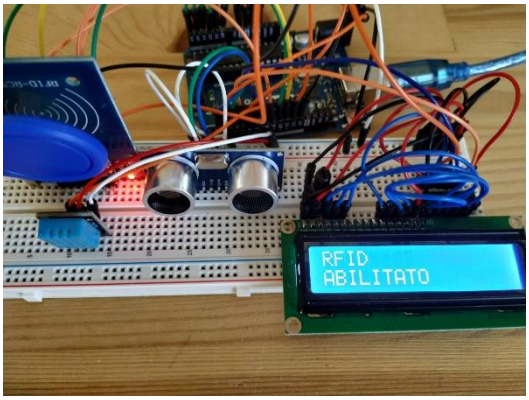
```
//INCLUDO LIBRERIA PER LA GESTIONE LCD TRAMITE REGISTRO
#include <Wire.h>
//Inizializzo le costanti per la gestione dello shift register per utilizzare l'LCD
const int PIN_LCD_STROBE =  7; // Out: LCD IC4094 shift-register strobe LACH PIN  - STCP
const int PIN_LCD_DATA =  6; // Out: LCD IC4094 shift-register data     DATA PIN  - DS
const int PIN_LCD_CLOCK =  5; // Out: LCD IC4094 shift-register clock   CLOCK PIN - SHCP
// srdata / srclock / strobe / bl pin on SR / blpol (positive\negative)
LiquidCrystal_SR3W lcd(PIN_LCD_DATA, PIN_LCD_CLOCK, PIN_LCD_STROBE, 7, POSITIVE); // 7 e POSITIVE definiscono lo stato associato all'output 7
```

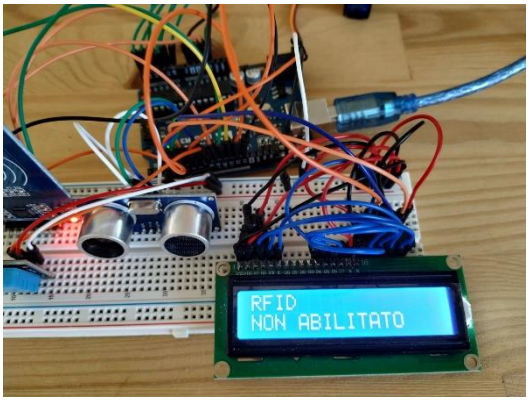


Lcd ci guiderà nella procedura di accesso attraverso la scelta tra scheda/badge RFID abilitato oppure tramite l'inserimento tramite keypad di un PIN d'accesso.

RFID RC522:

```
//INCLUDO LIBRERIA RFID
#include <MFRC522.h>
//RFID
MFRC522 rfid(10, 9); //COMUNICAZIONE CON RFID TRAMITE PIN 9 E 10
```





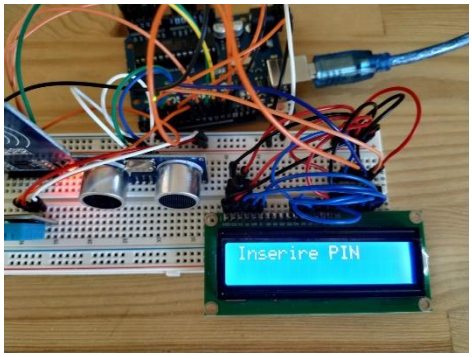
Scegliendo l'opzione 1 possiamo avvicinare l'RFID al sensore, e verificare se il nostro RFID è abilitato o meno.

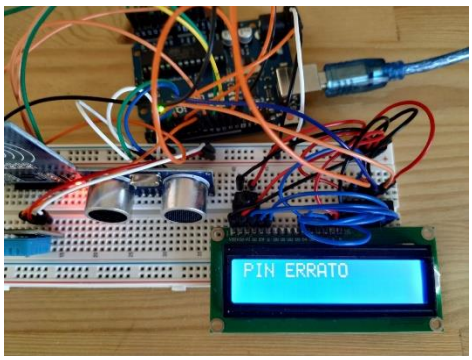
FUNZIONE PER LETTURA ID DA RFID:

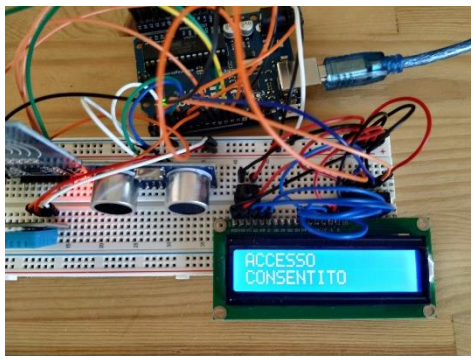
```
//FUNZIONE PER LA LETTURA DEL CODICE ID DELL'RFID
String getUID(){
  String uid = "";
  for(int i = 0; i < rfid.uid.size; i++){
    uid += rfid.uid.uidByte[i] < 0x9 ? "0" : "";  //Nel sistema HEX, lo zero iniziale è presente perch é i numeri devono iniziare con un carattere
numerico, e la 'x' significa esadecimale
    uid += String(rfid.uid.uidByte[i], HEX);
  }
  rfid.PICC_HaltA(); //Halt PICC
  return uid;
}
```

KEYPAD:

```
//INCLUDO LIBRERIA KEYPAD
#include <Keypad.h>
//KEYPAD
const byte ROWS = 3; //3 righe (ho scollegato la quarta riga per risparmiare pin)
const byte COLS = 3; //3 colonne (ho scollegato la quarta colonna)
char hexaKeys[ROWS][COLS] = { //definisco la matrice bidimensionale dei pulsanti del keypad
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
   };
byte rowPins[ROWS] = {19, 18, 17}; //utilizzo gli ingressi A3 A4 A5 come ingressi digitali che hanno numerazione 17 18 e 19 per gli ingressi righe
byte colPins[COLS] = {16, 15, 14}; //utilizzo gli ingressi A0 A1 A2 come ingressi digitali che hanno numerazione 14 15 e 16 per gli ingressi colonne
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);  //Inizializza un'istanza della classe Keypad
const char PIN[] ={'1','2','3','4'}; //definisco il PIN d' accesso 1234
```





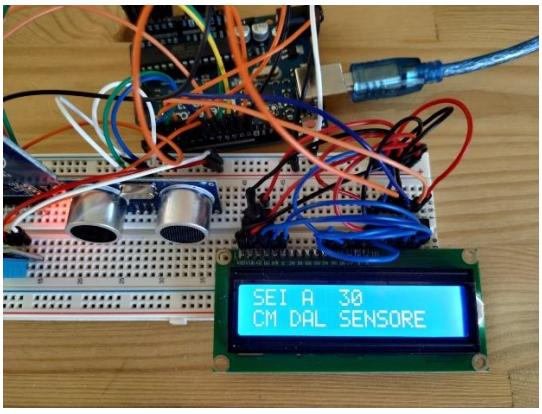


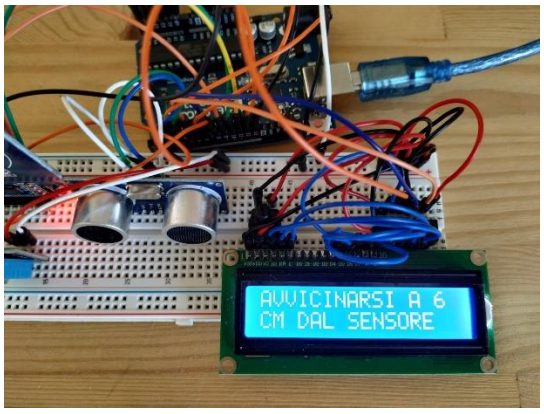
Scegliendo l'opzione 2, possiamo inserire il PIN attraverso il Keypad. Se il PIN è sbagliato ci comparirà la scritta PIN ERRATO, altrimenti, se corretto, ci consentirà l'accesso.

Una volta effettuate le procedure di accesso, si proseguirà attraverso l'avvicinamento al sensore di temperatura. L'avvicinamento dell'utente è agevolato attraverso l'utilizzo di un sensore ultrasonico, e successivamente viene utilizzato un sensore di temperatura per verificare che l'utente non presenti febbre.

SENSORE ULTRASONICO:

```
//INCLUDO LIBRERIA SENSORE ULTRASONICO
#include "SR04.h"
//definisco i pin ai quali il sensore  è  collegato
#define TRIG_PIN 4
#define ECHO_PIN 2
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN); // creo un'istanza per il sensore
```





FUNZIONE RILEVAMENTO DISTANZA DA RFID:

```
//FUNZIONE VICINANZA
boolean getClose(){
      while (sr04.Distance()>6){
        lcd.clear();
        lcd.home ();
        lcd.setCursor (0, 0); //XY
        lcd.print(F("AVVICINARSI A 6"));
        lcd.setCursor (0, 1); //XY
        lcd.print(F("CM DAL SENSORE"));
        delay(1000);
        lcd.clear();
        lcd.home ();
        lcd.setCursor (0, 0); //XY
        lcd.print(F("SEI A "));
        lcd.setCursor (7, 0); //XY
```

```
        lcd.print(sr04.Distance());
        lcd.setCursor (0, 1); //XY
        lcd.print(F("CM DAL SENSORE"));
        delay(1000);
      }
        lcd.clear();
        lcd.home ();
        lcd.setCursor (0, 0); //XY
        lcd.print(F("CONTROLLO"));
        lcd.setCursor (0, 1); //XY
        lcd.print(F("IN CORSO"));
        delay(3000);
        return true;
}
```

SENSORE TEMPERATURA:

```
#include <dht_nonblocking.h> //Includo la libreria di un sensore DHT11
#define DHT_SENSOR_TYPE DHT_TYPE_11 //definisco il tipo di sensore
static const int DHT_SENSOR_PIN = 8; //definisco come pin d' ingresso per il sensore il pin 8
DHT_nonblocking dht_sensor( DHT_SENSOR_PIN, DHT_SENSOR_TYPE ); //inizializzo il sensore
```

FUNZIONE LETTURA TEMPERATURA E CONTROLLO:

```
static bool getemperature (){
   float temperature;
   float humidity;

   while(dht_sensor.measure(&temperature,&humidity)==false){}
   Serial.print( temperature );
   lcd.clear();
   lcd.home ();
   lcd.setCursor (0, 0); //XY
   lcd.print(F("TEMPERATURA"));
   lcd.setCursor (0, 1); //XY
   lcd.print(temperature);/*stampiamo sul monitor la temperatura*/
   delay(1000);/*ritardo di un secondo*/
  if (temperature<37.5)
    return true;
  else
    return false;
}
```

Una volta che tutte le verifiche sono state effettuate, per consensire l'accesso si utilizzerà un servo motore che attraverso una rotazione di 90 gradi, simulerà l'apertura di una serratura.

ATTUATORE SERVO MOTORE MICRO:

```
//INCLUDO LIBRERIA SERVO MOTORE
#include <Servo.h>
//SERVOMOTORE
Servo myservo; //Creo un'istanza della classe Servo
```

FUNZIONE ROTAZIONE SERVO MOTORE MICRO:

```
void servomotor(){
  myservo.write(90);
  delay(5000);
  myservo.write(0);
}
```

FUNZIONI EXTRA:

Interessante in questo elaborato è la possibilità di programmare la memoria EEPROM del dispositivo Arduino in modo da memorizzare gli ID RFID, per rendere più celere l'accesso in futuro.

La memoria EEPROM è una memoria che non viene resettata in assenza di alimentazione, e per questo motivo si rende interessante il suo utilizzo per questo progetto.

Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of wl
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Dalle caratteristiche della scheda Arduino, in questo caso la UNO, possiamo osservare che la memoria è così strutturata:

- 512 byte che contengono il bootloader, ossia il sistema base che consente l'utilizzo di Arduino tramite l'esecuzione degli sketch;
- 2 KB di SRAM;
- 1KB di EEPROM ossia 1024 byte in cui salvare dati, che si possono ritrovare anche successivamente all'interruzione di corrente.

Per utilizzare questa memoria ci serviamo della libreria EEPROM.h già presente nell'IDE di Arduino. Questa libreria mette a disposizione due metodi:

1. read(address): con questo metodo possiamo leggere il valore memorizzato nella cella indicata dall'indirizzo di memoria;
2. write(address, value): con questo metodo possiamo scrivere in una delle 1024 celle (address) il valore (value) da memorizzare.

In questo progetto si è scelto di utilizzare l'indirizzo 0 della memoria EEPROM per salvare un contatore del numero di ID salvati, in modo da poter ottimizzare la ricerca in memoria, e successivamente dall'indirizzo di memoria 1 salvare in uno spazio di 9 byte tutti gli ID.

EEPROM:

```
//INCLUDO LIBRERIA EEPROM
#include <EEPROM.h>
//EEPROM
//Constanti per la gestione della memoria EEPROM, minimo e massimo degli indirizzi EEPROM
const int EEPROM_MIN_ADDR = 0;
const int EEPROM_MAX_ADDR = 511; //Possiamo utilizzarli fino all' indirizzo 1023
```

1. SAVEUIDRFID: verifica la presenza di un dispositivo RFID in prossimità del sensore, altrimenti ritorna *false,* se viene rilevato il dispositivo RFID, attraverso il metodo *getUID* leggerà l'ID del dispositivo come stringa. A questo punto tramite il for si controlla che l'ID non sia presente in memoria, e se tutto è corretto converte la stringa dell'ID in un array char in modo da usarlo come parametro per la scrittura nella EEPROM insieme all'indirizzo di memoria alla quale vogliamo scrivere l'ID.

```
boolean SaveRFIDuid(){
  const int BUFSIZE = 9; //9 è la dimensione necessaria per salvere l' uid dell' RFID
  char buf[BUFSIZE];
  String myString;
  char myStringChar[BUFSIZE];

  String uid;
  //codice rfid
  if(rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()){
     uid = getUID();
     for(int a=1; a<EEPROM.read(0)*9+1;a=a+9){ //la EEPROM pu ò salvare al massimo 1023 byte, quindi essendo 9byte necessari per ogni UID,
pu ò salvare 113 Codici RFID
       eeprom_read_string(a, buf, BUFSIZE);
       if (uid==buf){
         lcd.clear();
         lcd.home ();
         lcd.setCursor (0, 0); //XY
         lcd.print(F("RFID GIA'"));
         lcd.setCursor (0, 1); //XY
         lcd.print(F("ABILITATO"));
         delay(3000);
         return true;
       }
     }
     myString=uid;
     myString.toCharArray(myStringChar, BUFSIZE); //converto string in char array
     strcpy(buf, myStringChar);
     eeprom_write_string(EEPROM.read(0)*9+1, buf);
     EEPROM.write(0, EEPROM.read(0)+1); //il primo byte della eeprom è dedicato ad un contatore, che tiene conto del numero di uid salvati ogni
volta che ne viene aggiunto uno
     return true;
     }
     else
       return false;
}
```

2. CHECKUIDRFID: semplicemente verifica se un ID è presente o meno nella EEPROM.

```
boolean CheckRFIDuid(){
  const int BUFSIZE = 9;
  char buf[BUFSIZE];
```

```
  String myString;
  char myStringChar[BUFSIZE];

  String uid;
  //codice rfid
  if(rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()){
     uid = getUID();

     for(int a=1; a<=EEPROM.read(0)*9+1;a=a+9){ //la EEPROM pu ò  salvare al massimo 1023 byte, quindi essendo 9byte necessari per ogni UID,
pu ò  salvare 113 Codici RFID
         eeprom_read_string(a, buf, BUFSIZE);
       if (uid==buf)
           return true;
     }
  }
  else {
   lcd.clear();
   lcd.home ();
   lcd.setCursor (0, 0); //XY
   lcd.print(F("ERRORE DI LETTURA"));
   delay(2000);
   return false; //return false se fallisce la lettura dell'RFID
   }
   return false; //return false se l'ID non  è  presente in memoria
}
```

3. EEPROM_IS_ADDR_OK: verifico se l'indirizzo della EEPROM è valido

```
boolean eeprom_is_addr_ok(int addr) {
  return ((addr >= EEPROM_MIN_ADDR) && (addr <= EEPROM_MAX_ADDR));
}
```

4. EEPROM_WRITE_BYTES

```
boolean eeprom_write_bytes(int startAddr, const byte* array, int numBytes) {
  int i; //contatore
  if (!eeprom_is_addr_ok(startAddr) || !eeprom_is_addr_ok(startAddr + numBytes)) { //primo e ultimo byte devono ricadere nel range corretto
    return false;
   }
  for (i = 0; i < numBytes; i++) {
    EEPROM.write(startAddr + i, array[i]); //scrivo nell'  indirizzo specificato il valore contenuto in array[i]
  }
  return true;
}
```

5. EEPROM_WRITE_STRING

```
//Scrive una stringa iniziando da un indirizzo specifico.
boolean eeprom_write_string(int addr, const char* string) {
  int numBytes; // numero di byte da scrivere
  numBytes = strlen(string) + 1;  //scrive il contenuto della stringa e il carattere terminatore (0x00)
  return eeprom_write_bytes(addr, (const byte*)string, numBytes); //ritorna true se la stringa  è  stata scritta correttamente, altrimenti se qualche bit
ricade fuori range ritorna false
}
```

6. EEPROM_READ_STRING

```
//Legge una stringa a partire da uno specifico indirizzo, ritorna true se almeno un byte  è  letto, ritorna false se l'  indirizzo ricade al di fuori di quelli
//consentiti. La lettura puo'  fermarsi se: non vi  è  spazio nel buffer, l'  ultimo indirizzo eeprom viene raggiunto, o incontra la stringa terminatore 0x00
boolean eeprom_read_string(int addr, char* buffer, int bufSize) {
```

```
  byte ch; // byte che andro' a leggere dalla eeprom
  int bytesRead; // numero di byte letti
  if (!eeprom_is_addr_ok(addr)) {       //controllo se l' indirizzo è valido
  return false;
  }
 if (bufSize == 0) {    //esco se il buffer è vuoto
  return false;
 }
 if (bufSize == 1) {   // se c' è solo il carattere terminatore, esco
  buffer[0] = 0;
  return true;
 }
 bytesRead = 0;  // inizializzo il contatore per i bytes
 ch = EEPROM.read(addr + bytesRead);   // leggo il byte successivo dalla eeprom
 buffer[bytesRead] = ch;    // lo memorizzo nel buffer utente
 bytesRead++;    // incremento il contatore byte
 // Il while si ferma quando: Incontro il carattere 0x00, oppure ho riempito il buffer oppure ho raggiunto l' ultimo indirizzo eeprom
 while ( (ch != 0x00) && (bytesRead < bufSize) && ((addr + bytesRead) <= EEPROM_MAX_ADDR) ) {
  // finche' sono all' interno del while, leggo il byte successivo dalla eeprom
  ch = EEPROM.read(addr + bytesRead);
  buffer[bytesRead] = ch; // lo memorizzo nel buffer
  bytesRead++; // incremento il contatore
 }
 // mi assicuro che il buffer ha la stringa terminatore come ultimo byte, (0x00)
 if ((ch != 0x00) && (bytesRead >= 1)) {
  buffer[bytesRead - 1] = 0;
 }
 return true;
}
```

SETUP DELLO SKATCH:

```
 //INIZIALIZZAZIONE MOTORE SERVO

 myservo.attach(3);//connect pin 9 with the control line(the middle line of Servo)
 myservo.write(0);// lo pongo in posizione zero

 lcd.begin(16, 2);  // Inizializzo l'LCD per 16 char su 2 linee

 //Gli ingressi dal 15 al 19 vanno utilizzati come input
 pinMode(14,INPUT);
 pinMode(15,INPUT);
 pinMode(16,INPUT);
 pinMode(17,INPUT);
 pinMode(18,INPUT);
 pinMode(19,INPUT);

 //Inizializzazione RFID
 SPI.begin();          // Init SPI bus
 rfid.PCD_Init();

 //Monitor Seriale
 Serial.begin(9600);
}
```

LOOP DELLO SKATCH:

```
void loop() {

  lcd.clear();
  lcd.home ();
  lcd.setCursor (0, 0); //XY
  lcd.print(F("RDIF ACCESS: 1"));
  lcd.setCursor (0, 1);
  lcd.print(F("PIN  ACCESS: 2"));

  char customKey;

//ATTENDO L'INSERIMENTO DA KEYPAD
  while(true){
     customKey = customKeypad.getKey();
       if (customKey=='1' || customKey=='2')
         break;
   }

  switch(customKey){
   case '1':
          lcd.clear();
          lcd.home ();
          lcd.setCursor (0, 0); //XY
          lcd.print(F("Avvicinare RFID"));
          delay(5000);
          if(CheckRFIDuid()){
            lcd.clear();
            lcd.home ();
            lcd.setCursor (0, 0); //XY
            lcd.print(F("RFID"));
            lcd.setCursor (0, 1); //XY
            lcd.print(F("ABILITATO"));
            delay(3000);
          }
          else{
            lcd.clear();
            lcd.home ();
            lcd.setCursor (0, 0); //XY
            lcd.print(F("RFID"));
            lcd.setCursor (0, 1); //XY
            lcd.print(F("NON ABILITATO"));
            delay(3000);
            break;
          }
          getClose();
          if(getemperature()){
            lcd.clear();
            lcd.home ();
            lcd.setCursor (0, 0); //XY
            lcd.print(F("PREGO, PUO'"));
            lcd.setCursor (0, 1); //XY
            lcd.print(F("ACCEDERE"));
            delay(3000);
            servomotor();
          }
          else{
            lcd.clear();
            lcd.home ();
```

```
          lcd.setCursor (0, 0); //XY
          lcd.print(F("RILEVATA FREBBRE"));
          lcd.setCursor (0, 1); //XY
          lcd.print(F("VIETATO ENTRARE"));
          delay(3000);
        }
        break;

   case '2':
    lcd.clear();
    lcd.home ();
    lcd.setCursor (0, 0); //XY
    lcd.print(F("Inserire PIN"));  //IMPOSTIAMO PASSWORD "1234"

    //inserimento pin
    char pin_key[4];
    for(int j=0;;){
     customKey = customKeypad.getKey();
     if (customKey){
      pin_key[j]=customKey;
      lcd.setCursor (j, 1); //XY
      lcd.print(F("*"));
      j++;
      if (j==4)
       break;
     }
    }
    delay(1000);

    //SE IL CODICE CORRISPONDE AL PC CONSENTI L'ACCESSO
    if(pin_key[0]==PIN[0]&&pin_key[1]==PIN[1]&&pin_key[2]==PIN[2]&&pin_key[3]==PIN[3]){ //CONFRONTO OGNI SINGOLO
CARATTERE
     lcd.clear();
     lcd.home ();
     lcd.setCursor (0, 0); //XY
     lcd.print(F("ACCESSO"));
     lcd.setCursor (0, 1); //XY
     lcd.print(F("CONSENTITO"));
     delay(3000);
    }
    else {
     lcd.clear();
     lcd.home ();
     lcd.setCursor (0, 0); //XY
     lcd.print(F("PIN ERRATO"));
     delay(3000);
     break;
    }

    lcd.clear();
    lcd.home ();
    lcd.setCursor (0, 0); //XY
    lcd.print(F("VUOI REGISTRARE"));
    lcd.setCursor (0, 1);
    lcd.print(F("RFID? 1 SI 2 NO"));

    //ATTENDO L'INSERIMENTO DA KEYPAD
    while(true){
    customKey = customKeypad.getKey();
```

```
 if (customKey=='1' || customKey=='2')
   break;
}

switch (customKey){
  case '1':
    lcd.clear();
    lcd.home ();
    lcd.setCursor (0, 0); //XY
    lcd.print(F("Avvicinare RFID"));
    delay(3000);

    if(SaveRFIDuid()){
      lcd.clear();
      lcd.home ();
      lcd.setCursor (0, 0); //XY
      lcd.print(F("OPERAZIONE"));
      lcd.setCursor (0, 1); //XY
      lcd.print(F("COMPLETATA"));
    }
    else{
      lcd.clear();
      lcd.home ();
      lcd.setCursor (0, 0); //XY
      lcd.print(F("ERRORE DURANTE"));
      lcd.setCursor (0, 1); //XY
      lcd.print(F("LA MEMORIZZAZIONE"));
    }
    delay(3000);
    getClose();

   if(getemperature( )){
     lcd.clear();
     lcd.home ();
     lcd.setCursor (0, 0); //XY
     lcd.print(F("PREGO, PUO'"));
     lcd.setCursor (0, 1); //XY
     lcd.print(F("ACCEDERE"));
     delay(3000);
     servomotor();
    }
   else{
     lcd.clear();
     lcd.home ();
     lcd.setCursor (0, 0); //XY
     lcd.print(F("RILEVATA FREBBRE"));
     lcd.setCursor (0, 1); //XY
     lcd.print(F("VIETATO ENTRARE"));
     delay(3000);
   }
    break;

  case '2':
    getClose();
     if(getemperature()){
       lcd.clear();
       lcd.home ();
       lcd.setCursor (0, 0); //XY
       lcd.print(F("PREGO, PUO'"));
```

```
            lcd.setCursor (0, 1); //XY
            lcd.print(F("ACCEDERE"));
            delay(3000);
            servomotor();
          }
        else{
          lcd.clear();
          lcd.home ();
          lcd.setCursor (0, 0); //XY
          lcd.print(F("RILEVATA FREBBRE"));
          lcd.setCursor (0, 1); //XY
          lcd.print(F("VIETATO ENTRARE"));
          delay(3000);
        }
          break;

        }
      default:
       break;
  }
}
```