Studente: Raffaele Pilloni **Matricola:** 65151

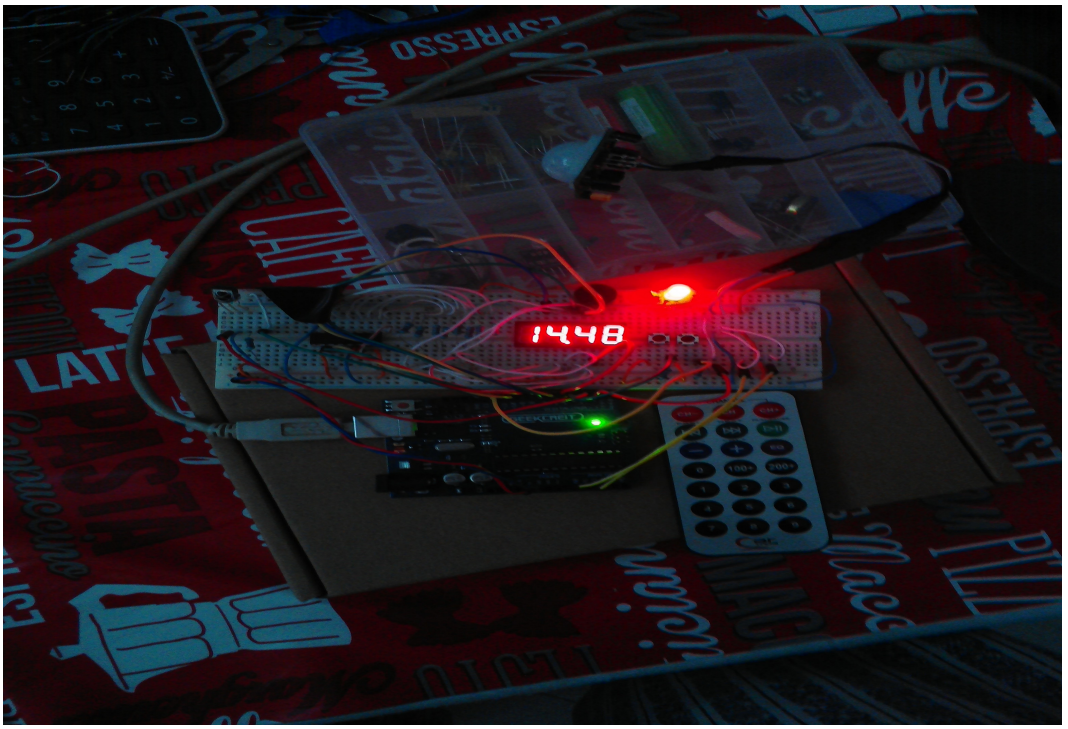
Architettura Degli Elaboratori 1 : Progetto Arduino

Centralina Allarme

Obbiettivi:

Il progetto punta alla realizzazione di una **centralina allarme**, dal quale è possibile effettuare il controllo di N stanze di un edificio tramite l utilizzo di **N sensori PIR** (per un massimo di 6 stanze rispettivamente le 6 entrate analogiche di Arduino). La centralina dispone di un display 4 digit 7 segment, al rilevamento di un oggetto in movimento in una delle 6 stanze La centralina emette un suono acuto tramite un Buzzer attivo, e il display mostra le stanze(**S. 01-S06** rispettivamente entrate **A0-A5**) in cui è stato rilevato il movimento. Tramite l utilizzo di appositi bottoni presenti nella centralina si potranno visualizzare data e ora odierna, oppure la data,l'ora e le stanze degli ultimi 10 rilevamenti salvati nella **EEPROM di Arduino**. Non avendo a disposizione un **RTC** (real time clock) la data e l'ora devono essere settate all'avvio (momento in cui la centralina tiene conto dei segnali che arrivano dai sensori), in tal caso la centrale riprenderà a controllare i sensori settando l'ora alle **00:00** e la data alle **01/01/2015**, ciò permette alla centratale di riprendere il suo funzionamento senza bloccarsi al settaggio della data, in caso di distacchi della corrente o sbalzi di tensione. La centralina potrà essere, attivata o disattivata, dopo l'inserimento di una password da un telecomando e la pressione del Tasto **CH+**, la password puo essere modificata dopo l'inserimento della vecchia password, la pressione del tasto **CH**, e il successivo inserimento della nuova.

avendo a disposizione un solo sensore PIR, per simulare utilizzo di più sensori, ho collegato l' uscita del sensore in parallelo alle uscite analogiche utilizzate. Per evitare false letture le uscite analogiche utilizzate devono essere per forza collegate a un sensore, o a massa(GND), oppure si può avvertire Arduino del range di porte utilizzate tramite la manipolazione delle costanti **ALLARMINPUT1 ALLARMINPUTN.**



Studente: Raffaele Pilloni **Matricola:** 65151

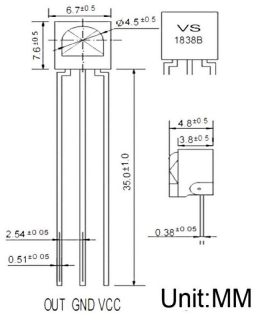
Architettura Degli Elaboratori 1 : Progetto Arduino
Centralina Allarme

Componenti utilizzati:

- **Scheda elettronica geekcreit(*clone Arduino uno*) con microcontrollore ATMEL atmega 328p:**

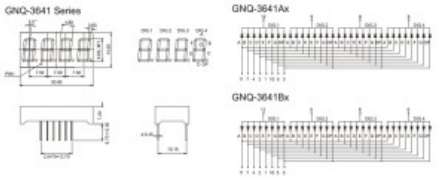
  Arduino è una piattaforma hardware **Open Source**, che permette l'interfacciamento con il microcontrollore **ATMEL atmega 328p,** e ne facilita la programmazione tramite un bootloader precaricato nella **EEPROM** e l'utilizzo di un attacco seriale collegato ai **Pin TX RX** del microcontrollore. La scheda è d'appoggio anche al successivo utilizzo e collegamento con moduli e componenti elettrici, e attraverso vari input per l'alimentazione.
  Caratteristiche:
  Tensione operativa: 5V
  Input Voltage: (consigliata) 7-12V
  Input Voltage: (limiti) 6-20V
  Pin di I/O Digitali: 14 (6 dei quali forniscono in uscita segnali PWM
  Pin di Input Analogici: 6
  DC Current per I/O Pin: 40 mA
  DC Current per Pin alimentati a 3.3V: 50mA
  Flash Memory: 32KB (di cui 0.5KB utilizzati dal bootloader)
  SRAM: 2KB
  EEPROM: 1KB
  Frequenza di Clock: 16MHz

- **Ricevitore IR e telecomando:**



  Tale componente permette la ricezione di segnali IR, che tramite l'utilizzo della libreria **IRremote** possono essere decodificati, i segnali vengono trasformati in parole di bit, e quindi permette la comunicazione con Arduino tramite un telecomando IR. I segnali cambiano da telecomando a telecomando, ho predisposto delle costanti che manipolate, permettono un veloce adattamento del codice con i diversi telecomandi, le costanti da **T0** a**T9** sono i numeri da **0** a **9**, le costanti **CHPSS** e **CHST** sono i tasti utilizzati per il cambio password e cambio stato, **RES** è la costante utilizzata per il reset password. La costante **DECODE** serve per isolare il segnale da altri telecomandi IR se non si conosce il tipo di codifica impostare a ***UNCKNOWN*.**

- ***Display 4 digit 7 segment 3641AS(catodo comune):***



  Il 4 digit è un componente elettronico che permette la visualizzazione di un numero di 4 cifre, tramite l' accensione di un tot di segmenti **LED** e l'attivazione di un digit impostando a **LOW** il Pin relativo al digit,se il display è a **catodo comune,** o ad **HIGH** se esso è ad **anodo comune.** Per poter visualizzare una cifra diversa per ogni digit bisogna utilizzare la tecnica del **multiplxing**, cioè attivare un digit alla volta con i relativi segment attivati e muoversi ad una velocità superiore dell'occhio umano tra un digit e l'altro ,modificando i valori dei segment(permette di utilizzare solo **12 Pin**).

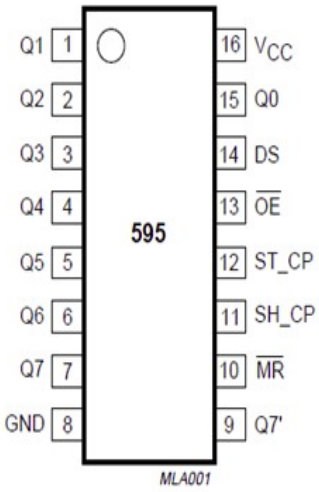
Studente: Raffaele Pilloni **Matricola:** 65151

Architettura Degli Elaboratori 1 : Progetto Arduino

Centralina Allarme

- **Shift Register 8 bit SN74HC595(*SIPO*):**

  Shift Register(registro a scorrimento), è un microchip composto da un numero di **flip-flop SR**(componenti elettronici capaci di memorizzare 1 **Bit** attraverso il settaggio di due input **set** e **reset**)collegati in serie, in questo caso 8, permette quindi la memorizzazione di un dato di **8 bit** inviato in seriale, e il successivo invio parallelo nei Pin di output (**Q0-Q7**).il microchip dispone di 3 Pin di input, uno per attivare e disattivare la comunicazione seriale(**Latch**), uno per il **Clock** per la scansione del messaggio seriale e uno per il dato da inviare. In questo progetto viene adoperato per diminuire i Pin da utilizzare per il Display 4 digit, e anche per semplificare la comunicazione con esso. Infatti andando a collegare le 8 uscite dello Shift Register alle otto entrate dei segmenti del display, i Pin utilizzati non saranno più 12 ma 7, e in più l'attivazione di uno o più segment corrisponderà a un numero di 8 bit, che viene spedito allo Shift Register tramite la funzione **ShiftOut**.

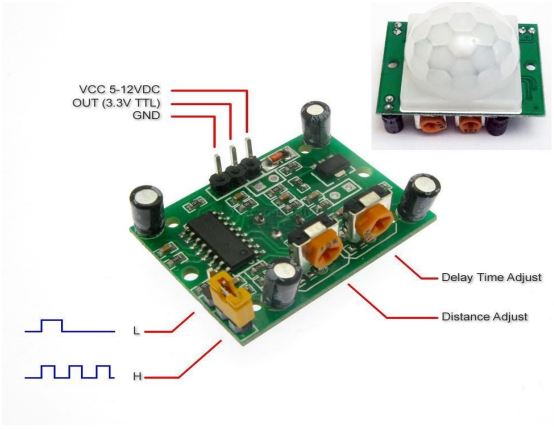


PINNING

PIN	SYMBOL	DESCRIPTION
1	Q1	parallel data output
2	Q2	parallel data output
3	Q3	parallel data output
4	Q4	parallel data output
5	Q5	parallel data output
6	Q6	parallel data output
7	Q7	parallel data output
8	GND	ground (0 V)
9	Q7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_CP	shift register clock input
12	ST_CP	storage register clock input
13	$\overline{OE}$	output enable (active LOW)
14	DS	serial data input
15	Q0	parallel data output
16	V_{CC}	positive supply voltage

- **Modulo sensore PIR**:

  il modulo sensore PIR ci permette di rilevare i movimenti, in pratica al rilevamento di un movimento davanti al suo raggio, aziona l'uscita da 3,3 volt che noi andremo ad intercettare con i Pin analogici di Arduino. Il modulo dispone di due potenziometri uno per regolare la distanza del raggio,e uno per regolare il timer di distacco dopo ogni rilevamento, dispone di due modalità, una dove il timer di distacco si avvia al primo rilevamento, e una dove il timer non si avvia sinché l'oggetto in movimento non si sposta da raggio del sensore.

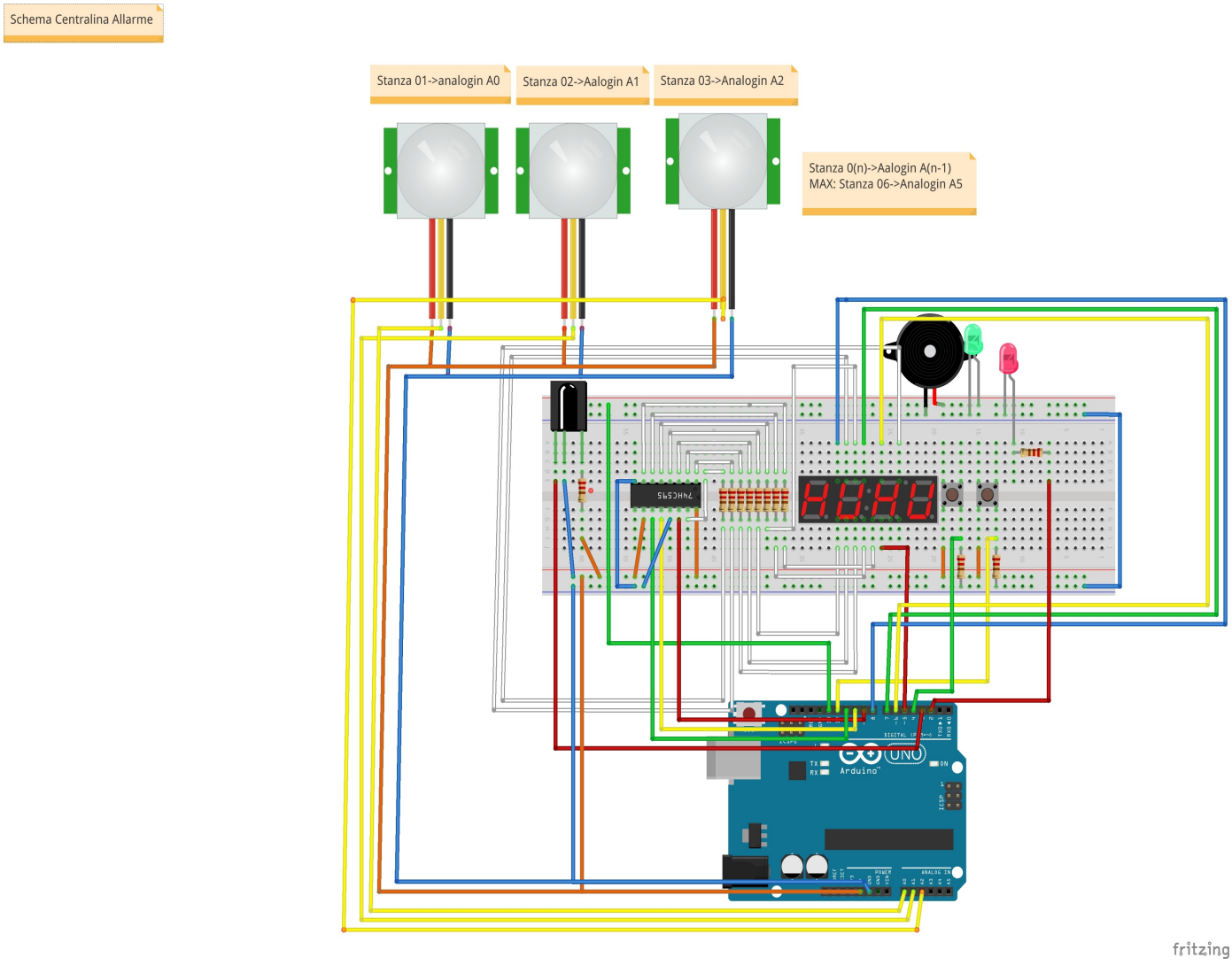


- **10 resistenze da 220 Ohm:**
  Utilizzate per i led del display per il led rosso, e per il ricevitore IR.
- **2 resistenze da 1 Kohm:**
  resistenze di Pull-down utilizzate per i due bottoni
- **2 Bottoni.**
- **1 LED rosso.**
- **1 LED verde.**
- **1 Buzzer Attivo.**

Studente: Raffaele Pilloni **Matricola:** 65151

Architettura Degli Elaboratori 1 : Progetto Arduino
Centralina Allarme

- **Schema circuito (immagine creata con Fritzing):**



Sketch Principale: *ARE1.Pilloni.Raffaele.65151._cetralinaAllarmeArduino_.ino*

```
/** LIBRERIE **/
#include <Time.h> //Libreria utilizzata per la scasione del tempo
#include <EEPROM.h> //Libreria utilizzata per la lettura e scrittura EEPROM Arduino
#include <IRremote.h> //Libreria utilizata per la ricezione dei segnali IR
/** COSTANTI D'APPOGGIO PER LA GESTIONE DEI DIGIT DEL DISPLAY **/
#define DIGIT1 8 //Pin per attivare il primo digit
#define DIGIT2 7 //Pin per attivare il secondo digit
#define DIGIT3 6 //Pin per attivare il terzo digit
#define DIGIT4 5 //Pin per attivare il quarto digit
#define DIGITON LOW //Attiva digit
#define DIGITOFF HIGH //Disattiva digit
/** COSTANTI D'APPOGGIO PER LA GESTIONE DEI SEGMENTI DEL DISPLAY TRAMITE SHIFT
REGISTER **/
#define CLK 11 //Pin utilizzato come clock per Shift Register
#define LATCH 10 //Pin utilizzato come LATCH(accendi o spegni comunicazione)Shift
Register
#define DATA 9 //Pin di trasmissione dati Shift Register
#define N1 0x6 //Valore numero 1 display
#define N2 0x5B //Valore numero 2 display
#define N3 0x4F //Valore numero 3 display
#define N4 0x66 //Valore numero 4 display
#define N5 0x6D //Valore numero 5 display
#define N6 0x7D //Valore numero 6 display
#define N7 0x7 //Valore numero 7 display
#define N8 0x7F //Valore numero 8 display
#define N9 0x6F //Valore numero 9 display
#define N0 0x3F //Valore numero 0 display
#define PSSINSH 0x40 //Utilizzate per mostrare l avanzamento pasword
#define PSSINSL 0x8
#define ADD_DP 0x80 //Utilizzata per aggiunge il punto
#define SENDON LOW //Attiva comunicazione Shift Register
#define SENDOFF HIGH //Disattiva comunicazione Shift Register
#define BUTTON1 4 //Pin bottoni utilizzati per la visualizzazione del display
#define BUTTON2 12
/** COSTANTI UTILIZZATE PER LA GESTIONE DEI SEGNALI IR **/
#define RCV 3 //Pin utilizzato per la ricezione segnali IR
#define T1 0xFF30CF //Valore tasto 1 telecomando
#define T2 0xFF18E7 //Valore tasto 2 telecomando
#define T3 0xFF7A85 //Valore tasto 3 telecomando
#define T4 0xFF10EF //Valore tasto 4 telecomando
#define T5 0xFF38C7 //Valore tasto 5 telecomando
#define T6 0xFF5AA5 //Valore tasto 6 telecomando
#define T7 0xFF42BD //Valore tasto 7 telecomando
#define T8 0xFF4AB5 //Valore tasto 8 telecomando
#define T9 0xFF52AD //Valore tasto 9 telecomando
#define T0 0xFF6897 //Valore tasto 0 telecomando
#define RES 0xFF906F //Valore tasto EQ telecomando utilizzato per resettare la
password inserita
#define CHPSS 0xFF629D //Valore tasto CH+ telecomando utilizzato per cambiare
password
#define CHST 0xFFE21D //Valore tasto CH telecomando utilizzato per attivare o
disattivare la centralina
#define DECODE NEC
/** COSTANTI PER LA GESTIONE DELL'ALLARME **/
#define LPSS 4 //Lunghezza password
#define ADRNUMBERALLARMSAVE 5 //Indirizzo della EEPROM in cui ?presente il numero di
stanze rilevate salvate nella EEPROM
#define MAXALLARMSAVE 10 //Massimo numero stanze salvate
#define ADRSTATE 0 //Indirizzo dello stato della centralina
#define ALLARMINPUT1 14 //Pin primo sensore stanza 01
#define ALLARMINPUTN 16 //Pin ultimo sensore collegato massimo 19
#define LED_ONOFF 2 //Pin led di attivazione centralina
#define ALLARMON 13 //Pin led e Buzzer di rilevamento
#define MILLISECALLARMCTRL 3000 //Millisecondi di cambio da rilevamento a rilevamento
/** STRUTTURE E VARIABILI GLOBALI **/
```

```
typedef struct Password{ //struttura password
 int flag; //flag di controllo password
 int pssw[LPSS]; // password
 int guess[LPSS]; //password iserita dall'utente
 int ContPssw=0; //contatore pasword
}Password;
IRrecv irrecv(RCV); //inizializzazione pin di ricezione segnali IR
decode_results results; //risultati segnali IR
int Allarm=ALLARMINPUT1; //contatore utilizzato per il controllo dei sensori
long int Old_time=0; //utilizata per salvare il valore di millis e scansionare il
tempo trascorso dal rilevamento
int flagFirst=0; //primo rilevamento di ogni stanza
Password ATpssw; //variabile password
/** SETUP INIZIALE **/
void setup() {
 irrecv.enableIRIn();//inizializzazione ricezione IR
 /* inizializzazione pin */
 pinMode(LED_ONOFF,OUTPUT);
 pinMode(ALLARMON,OUTPUT);
 pinMode(BUTTON1,OUTPUT);
 pinMode(BUTTON2,OUTPUT);
 pinMode(CLK,OUTPUT);
 pinMode(LATCH,OUTPUT);
 pinMode(DATA,OUTPUT);
 for(int i=DIGIT1;i>=DIGIT4;i--){
 pinMode(i,OUTPUT);
 digitalWrite(i,DIGITOFF);
 }
 for(int i=ALLARMINPUT1;i<=ALLARMINPUTN;i++) pinMode(i,INPUT);

 ATpssw.ContPssw=0; //inizializzazione contatore password
 ATpssw.flag=0; //inizializzazione flag password
 //EEPROM.write(ADRNUMBERALLARMSAVE,0);
 caricaPssw(); //carica password da EEPROM
 if(!EEPROM.read(ADRSTATE))setTemp(); //se stato 0 l utente puo settare la data
 else setTime(00,00,00,01,01,2015); //altrimenti la data sara settata alle 00:OO del
01-01-2015
}
/** LOOP PRINCIPALE **/
void loop() {
 MenuIR();
 DisplayPresentation();

if(EEPROM.read(ADRSTATE) && digitalRead(Allarm)){ // se lo stato e attivo e viene
rilevato un movimento
 digitalWrite(LED_ONOFF,LOW); //attiva led stato attivo
 digitalWrite(ALLARMON,HIGH);//attiva buzzer e led allarme
 if(flagFirst==0){ // se e il primo giro attivo per la stanza
 if(EEPROM.read((EEPROM.read(ADRNUMBERALLARMSAVE)*7)+ADRNUMBERALLARMSAVE-6) !=
Allarm-ALLARMINPUT1+1) // se il rilevamento non proviene dalla stessa stanza gia
salvata
 SaveDateAllarm(Allarm-ALLARMINPUT1+1,hour(),minute(),day(),month(),year()); //salva
stanza e ora
 flagFirst=1; //disattiva primo giro
 Old_time=millis(); //conserva il tempo
 }
}
else if(EEPROM.read(ADRSTATE)){ //se lo stato e attivo ma non vengono rilevati
movimenti
 digitalWrite(LED_ONOFF,HIGH);//attiva led stato attivo
 digitalWrite(ALLARMON,LOW);//disattiva led e buzer allarm
}
else{//stato e spento
 digitalWrite(LED_ONOFF,LOW);//disattiva led stato
 digitalWrite(ALLARMON,LOW);//disattiva led e buzer allarm
}

if(millis()-Old_time>=MILLISECALLARMCTRL){ //se son passati piu millisec di
MILLISECALLARMCTRL dall ultimo rilevamento della stanza
```

```
 Allarm++; //controlla prossima stanza
 flagFirst=0;//attiva primo giro
 Old_time=0;
 }
if(Allarm>ALLARMINPUTN) Allarm=ALLARMINPUT1; //torna a controllare la stanza 1 se
superata stanza N
}
```

Sketch Funzioni: *FunzioniCentralinaAllarme.ino*

```
/** FUNZIONI DI SETUP INIZIALI **/
void caricaPssw(){ //funzione carica password dall EEPROM
 for(int i=0;i<LPSS;i++) ATpssw.pssw[i]=EEPROM.read(i+1);
}
void setTemp(){ //funzione set data e ora
 //variabili bottone 1
 int valb1=0;
 int old_valb1=0;
 int stateb1=0;
 //variabili bottone 2
 int valb2=0;
 int old_valb2=0;
 int stateb2=0;
 //variabili data
 int hour=0;
 int minut=0;
 int day=0;
 int month=0;
 int year=0;
 //varibili d appoggio anno selezionatori cifra
 int old_valT=0;
 int old_valT2=0;
 int SelectCfr=1000;
 do{
 valb1=digitalRead(BUTTON1);
 valb2=digitalRead(BUTTON2);
 if(valb2==HIGH && old_valb2==LOW) { //pressione bottone 2
 stateb2++; //cambia info
 stateb1=0;//azzera counter
 if(stateb2>4) SelectCfr=SelectCfr/10; //se anno passa a cotrollare altra cifra
 }
 old_valb2=valb2;
 valb2=0;

 if( stateb1==0 && stateb2>=4 || valb1==HIGH && old_valb1==LOW){ //pressione bottone
1 o nel caso setaggio anno se 0 entra anche senza pressione
 if( valb1==HIGH && old_valb1==LOW) stateb1=(stateb1+1)%10; //pressione bottone 1
aumenta stato cifra
 switch(stateb2){
 case 0: hour=(hour+1)%24;break;
 case 1: minut=(minut+1)%60; break;
 case 2:day=(day+1)%32;break;
 case 3: month=(month+1)%13;break;
 case 4:
 year=stateb1*SelectCfr;
 old_valT=year;
 break;
 case 5:
 year=old_valT+(stateb1*SelectCfr);
 old_valT2=year;
 break;
 case 6:
 year=old_valT2+(stateb1*SelectCfr);
 old_valT=year;
 break;
 case 7:year=old_valT+stateb1;
 break;
```

```
 }
 }
 old_valb1=valb1;
 valb1=0;

 /*visualizazione display nel settaggio orario*/
 if(stateb2 >=0 && stateb2<2) DisplayTimeDate(hour,minut);
 else if(stateb2 >=2 && stateb2<4) DisplayTimeDate(day,month);
 else if(stateb2>=4) DisplayYrs(year);
 }while(stateb2<8);
 setTime(hour,minut,00,day,month,year);//setta tempo
 delay(1000);
}
/** FUZIONI INTERAZIONE IR E AUTENTICAZIONE **/
void MenuIR(){ //funzione menu decisionale cambio stato o cambio password
 insPssw();//inserimento password
 if (irrecv.decode(&results) && results.decode_type==DECODE && ATpssw.flag){ //se
arrivano segnali IR e la password e corretta
 if(results.value==CHST){ //cambia stato
 changState();
 ATpssw.flag=0;
 }
 else if(results.value==CHPSS){//cambia password
 changePssw();
 ATpssw.flag=0;
 }
 irrecv.resume();
 }
}
void changState(){//funzione cambia stato
EEPROM.write(ADRSTATE,(EEPROM.read(ADRSTATE)+1)%2);
}
void insPssw(){ //funzione inserisci password
 int i=0;
 if (irrecv.decode(&results) && results.decode_type==DECODE && !ATpssw.flag){//se
password non esatta allora inserisci password
 if(results.value!=RES) ATpssw.guess[ATpssw.ContPssw]=psswTranslate(results.value);
 else ATpssw.ContPssw=0;
 irrecv.resume();
 if(ATpssw.ContPssw==LPSS){//controlla password inserita
 ATpssw.flag=1;
 for(i=0;i<LPSS;i++)
 if(ATpssw.guess[i]!=ATpssw.pssw[i]) ATpssw.flag=0;
 ATpssw.ContPssw=0;
 }
 }
}
void changePssw(){ //funzione cambio password
do{
 DisplayPssw();
 if (irrecv.decode(&results) && results.decode_type==DECODE){ //inserimento nuova
password
 if(results.value!=RES) ATpssw.guess[ATpssw.ContPssw]=psswTranslate(results.value);
 else{
 ATpssw.ContPssw=0;
 return;
 }
 irrecv.resume();
 }
}while(ATpssw.ContPssw<LPSS);
for(int i=0;i<LPSS;i++){
 ATpssw.pssw[i]=ATpssw.guess[i];
 EEPROM.write(i+1,ATpssw.pssw[i]);//salva nella EEPROM
}
ATpssw.ContPssw=0;
}
int psswTranslate(int val){ //funzione di traduzione tasti numerici telecomando
 switch(val){
 case T1: ATpssw.ContPssw++;return 1;
```

```
 case T2: ATpssw.ContPssw++;return 2;
 case T3: ATpssw.ContPssw++;return 3;
 case T4: ATpssw.ContPssw++;return 4;
 case T5: ATpssw.ContPssw++;return 5;
 case T6: ATpssw.ContPssw++;return 6;
 case T7: ATpssw.ContPssw++;return 7;
 case T8: ATpssw.ContPssw++;return 8;
 case T9: ATpssw.ContPssw++;return 9;
 case T0: ATpssw.ContPssw++;return 0;
 }
}
/** FUNZIONI DI VISUALIZAZZIOE DISPLAY 7 SEGMENT 4 DIGIT UTILIZZANDO TECNICA
MULTIPLEXER ATTRAVERSO SHIFT REGISTER **/
void DisplayPresentation(){ //funzione menu decisionale scelta di visualizazzione
 int fine=(EEPROM.read(ADRNUMBERALLARMSAVE)*7)+ADRNUMBERALLARMSAVE;//indirizzo EEPROM
fine rilevamenti salvati
 static int numberAdress=ADRNUMBERALLARMSAVE+1;//indirizzo EEPROM inizio rilevamenti
salvati
 //variabili bottone 1
 static int valb1=0;
 static int old_valb1=0;
 static int stateb1=0;
 //variabili bottone 2
 static int valb2=0;
 static int old_valb2=0;
 static int stateb2=0;
 valb1=digitalRead(BUTTON1);
 valb2=digitalRead(BUTTON2);
 if(valb2==HIGH && old_valb2==LOW) {//se pressione bottone 2
 stateb2=(stateb2+1)%2; //cambia stato bottone 2
 stateb1=0; //azzera bottone 1
 numberAdress=ADRNUMBERALLARMSAVE+1;//torna a inizio lista rilevamenti salvati
 }
 old_valb2=valb2;
 valb2=0;

 if(valb1==HIGH && old_valb1==LOW && stateb2==0) //stato bottone 2=0 e pressione
bottone1
 stateb1=(stateb1+1)%3;// cambia stato bottone 1
 else if(valb1==HIGH && old_valb1==LOW && stateb2==1 && fine>5){//stato bottone 2=1 e
pressione bottone1
 stateb1=(stateb1+1)%4;// cambia stato bottone1
 if(stateb1==0) numberAdress++; // elemento successivo
 else numberAdress+=2;
 if(numberAdress>fine){// se arriva a fine lista torna a capo
 numberAdress=ADRNUMBERALLARMSAVE+1;
 stateb1=0;
 }
 } else if(fine==5) stateb2=0; //se non e presente nessun elemento torna alla data
 old_valb1=valb1;
 valb1=0;
 if(ATpssw.ContPssw>0) DisplayPssw(); // se niserimetno password visualizza andamento
 else if(EEPROM.read(ADRSTATE) && digitalRead(Allarm)) DisplayRoom(Allarm-
ALLARMINPUT1+1); //altrimenti se scatta allarme visualizza stanza
 else if(stateb2==1) DisplayUltimateAllarmOn(stateb1,numberAdress); //altrimenti se
in stato bottone2=1 visualizza rilevamenti salvati
 else if(stateb2==0)DisplayTimeDataYears(stateb1); // altrimenti se stato=0
visualizza data
}
void DisplayRoom(int Room){ //funzione visualizza stanza
 SendToRegister(N5+ADD_DP); //visualizza S.
 digitalWrite(DIGIT1,DIGITON);
 delay(2);
 digitalWrite(DIGIT1,DIGITOFF);
 SendToRegister(0);
 digitalWrite(DIGIT2,DIGITON);
 delay(2);
 digitalWrite(DIGIT2,DIGITOFF);
 SendToRegister(DisplayTranslate(0));
```

```
 digitalWrite(DIGIT3,DIGITON);
 delay(2);
 digitalWrite(DIGIT3,DIGITOFF);
 SendToRegister(DisplayTranslate(Room)); //visualizza numero stanza
 digitalWrite(DIGIT4,DIGITON);
 delay(2);
 digitalWrite(DIGIT4,DIGITOFF);

}
void DisplayPssw(){ //funzione visualizza andamento password
 for(int i=DIGIT1;i>DIGIT1-ATpssw.ContPssw;i--){ // segmento G per numero password
inserito
 SendToRegister(PSSINSH);
 digitalWrite(i,DIGITON);
 delay(2);
 digitalWrite(i,DIGITOFF);
 }
 for(int i=DIGIT1-ATpssw.ContPssw;i>=DIGIT4;i--){//segmento D per numeri password
mancanti
 SendToRegister(PSSINSL);
 digitalWrite(i,DIGITON);
 delay(2);
 digitalWrite(i,DIGITOFF);
 }
}
void DisplayUltimateAllarmOn(int stateb1,int numberAdress){//funzione scelta
visualizazzione allarmi salvati
 switch(stateb1){
 case 0: DisplayRoom(EEPROM.read(numberAdress));break;
 case 1:DisplayTimeDate(EEPROM.read(numberAdress-1),EEPROM.read(numberAdress));break;
 case 2:DisplayTimeDate(EEPROM.read(numberAdress-1),EEPROM.read(numberAdress));break;
 case 3:DisplayYrs((EEPROM.read(numberAdress-
1)*256)+EEPROM.read(numberAdress));break;
 }
}
void DisplayTimeDataYears(int stateb1){//funzione scelta visualizza data
 switch(stateb1){
 case 0:DisplayTimeDate(hour(),minute());break;
 case 1:DisplayTimeDate(day(),month());break;
 case 2:DisplayYrs(year());break;
 }
}
void DisplayTimeDate(int DH,int MM){//funzione visualizza ore minuti o giorno mese
 SendToRegister(DisplayTranslate(DH/10));
 digitalWrite(DIGIT1,DIGITON);
 delay(2);
 digitalWrite(DIGIT1,DIGITOFF);
 SendToRegister(DisplayTranslate(DH%10)+ADD_DP);
 digitalWrite(DIGIT2,DIGITON);
 delay(2);
 digitalWrite(DIGIT2,DIGITOFF);
 SendToRegister(DisplayTranslate(MM/10));
 digitalWrite(DIGIT3,DIGITON);
 delay(2);
 digitalWrite(DIGIT3,DIGITOFF);
 SendToRegister(DisplayTranslate(MM%10));
 digitalWrite(DIGIT4,DIGITON);
 delay(2);
 digitalWrite(DIGIT4,DIGITOFF);
}
void DisplayYrs(int Y){//fuzione visualizza anno
 SendToRegister(DisplayTranslate(Y/1000));
 digitalWrite(DIGIT1,DIGITON);
 delay(2);
 digitalWrite(DIGIT1,DIGITOFF);
 Y=Y%1000;
 SendToRegister(DisplayTranslate(Y/100));
 digitalWrite(DIGIT2,DIGITON);
 delay(2);
```

```
 digitalWrite(DIGIT2,DIGITOFF);
 Y=Y%100;
 SendToRegister(DisplayTranslate(Y/10));
 digitalWrite(DIGIT3,DIGITON);
 delay(2);
 digitalWrite(DIGIT3,DIGITOFF);
 Y=Y%10;
 SendToRegister(DisplayTranslate(Y));
 digitalWrite(DIGIT4,DIGITON);
 delay(2);
 digitalWrite(DIGIT4,DIGITOFF);
}
int DisplayTranslate(int N){ //fuzione di traduzione per display tramite Shift
Register
 switch(N){
 case 1:return N1;
 case 2:return N2;
 case 3:return N3;
 case 4:return N4;
 case 5:return N5;
 case 6:return N6;
 case 7:return N7;
 case 8:return N8;
 case 9:return N9;
 case 0:return N0;
 }
}
void SendToRegister(int val){// funzione per inviare un dato allo Shift Register
utilizato per accendere led numerici
 digitalWrite(LATCH,SENDON);
 shiftOut(DATA, CLK, MSBFIRST,val);
 digitalWrite(LATCH,SENDOFF);
}
/** SALVATAGGIO RILEVAMENTI **/
void SaveDateAllarm(int r,int h,int m,int D,int M,int Y){//funzione salva rilevamento
 int numberAlarmSave=EEPROM.read(ADRNUMBERALLARMSAVE);
 int numberAdress=(numberAlarmSave*7)+ADRNUMBERALLARMSAVE;
 numberAdress++;
 EEPROM.write(numberAdress,r);
 numberAdress++;
 EEPROM.write(numberAdress,h);
 numberAdress++;
 EEPROM.write(numberAdress,m);
 numberAdress++;
 EEPROM.write(numberAdress,D);
 numberAdress++;
 EEPROM.write(numberAdress,M);
 numberAdress++;
 EEPROM.write(numberAdress,highByte(Y));
 numberAdress++;
 EEPROM.write(numberAdress++,lowByte(Y));
 numberAlarmSave=(numberAlarmSave+1)%MAXALLARMSAVE;
 EEPROM.write(ADRNUMBERALLARMSAVE,numberAlarmSave);
}
```