# FORTH SECURITY SYSTEM

Embedded Systems


**Componenti del gruppo:**

Alessandro Giambanco

Daniele Franco

Rosario Bongiorno

**Professore:**

Daniele Peri

Università degli Studi di Palermo

# Sommario

# 1. Descrizione del progetto

Il progetto consiste nella realizzazione di un sistema di allarme domestico, il quale scopo è la rilevazione di intrusioni all'interno di abitazioni private.

È dotato di due diversi sensori che permettono di individuare movimento all'interno delle stanze ed ingressi dalle aperture quali porte e finestre, inoltre attraverso dei led e un buzzer segnala l'effettiva intrusione.

Inoltre, è dotato di un display lcd che permette di notificare all'utente in che stato si trova il sistema e di un ricevitore infrarossi che permette mediante un telecomando di inserire il pin che disattiva l'allarme.

Il sistema presenta tre fasi:

- Allarme disattivo: in questa fase l'allarme non è attivo, è acceso un led verde e mediante un pulsante l'utente attiva l'allarme, questo avviene soltanto dopo un countdown di dieci secondi che permette all'utente di uscire dalla stanza.
- Allarme attivo: in questa fase l'allarme è attivo, è acceso un led rosso, dopo aver rilevato un movimento ed un'intrusione il sistema avvia l'allarme o premendo un pulsante l'utente può inserire la password per disattivare l'allarme
- Allarme in corso: in questa fase l'allarme è in corso, è acceso un buzzer che emette un suono e il led rosso lampeggia, l'utente può premere il pulsante che gli permetterà di inserire la password e disattivare l'allarme.

Il sistema è stato realizzato a scopo didattico mediante l'uso del target Raspberry Pi 3A+ e di diversi sensori e attuatori, il linguaggio di programmazione utilizzato è il Forth e l'interprete è pijFORTHos rielaborato dal Professore Daniele Peri.

## 1.1. Sviluppi futuri

Come primo sviluppo è stata simulata l'intrusione in una stanza che dispone di un solo ingresso, per un'applicazione reale il sistema è stato pensato per prevedere più stanze con più ingressi in ognuna di esse.

# 2. Componenti del progetto

## 2.1. Componenti Hardware

I Componenti elettronici utilizzati per la realizzazione sono:

- Raspberry Pi 3A+ 512MB RAM
- Sensore di distanza ad ultrasuoni HC-SR04
- Sensore di movimento HC-SR501
- Display LCD 1602A
- Ricevitore infrarosso
- Telecomando infrarosso
- Interfaccia seriale UART FT232RL
- Buzzer acustico
- Led rosso
- Led verde
- Cavi jumper
- 4 resistori da 200 ohm

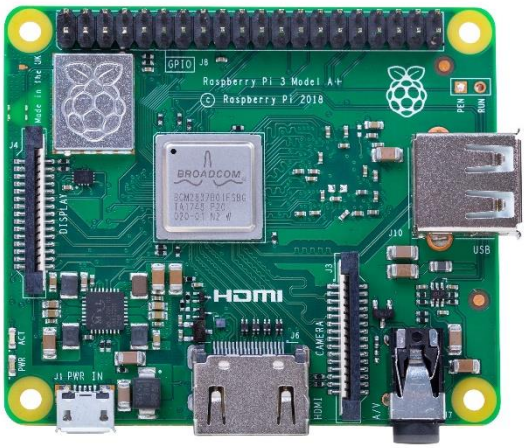
## 2.2. Componenti Software

Per la programmazione del dispositivo target è stato usato il sistema operativo con interprete **pijFORTHos** che permette l'interazione a basso livello con l'hardware del dispositivo mediante il linguaggio di programmazione Forth.

Per l'interazione con il dispositivo e il caricamento del codice è stata instaurata una comunicazione seriale mediante il software **ZOC8 Terminal** da pc Windows.

# 3. Descrizione componenti

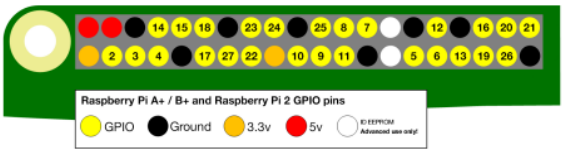
## 3.1. Raspberry Pi 3 Model A+




Single board computer, progettato per essere accessibile, versatile e adatto a una vasta gamma di progetti elettronici e di programmazione. Nato come una versione più economica e compatta del Raspberry Pi 3 Model B+, questo dispositivo mantiene molte delle sue funzionalità principali, offrendo al contempo un formato ridotto e un prezzo più contenuto.

È dotato di un processore quad-core ARM Cortex-A53 da 1.4 GHz, la memoria RAM di 512 MB lo rende adatto per applicazioni che non richiedono molta memoria, come progetti di automazione domestica o piccoli server.

Per la realizzazione del progetto abbiamo utilizzato l'interfaccia GPIO che dispone di 40 pin, grazie ai quali è possibile collegare una vasta gamma di componenti esterni.




Dispone di diversi tipi di pin che si dividono in **pin di alimentazione** quali pin a 3.3V, 5V e GND, pin **GPIO generici** che possono essere configurati sia come input che, come output e pin, **GPIO specializzati** che permettono funzionalità specifiche come ad esempio **I2C, SPI** ed **UART**

## 3.2. Sensore di distanza ad ultrasuoni HC-SR04


È un dispositivo economico e ampiamente utilizzato per misurare distanze, basato sul principio dell'eco, funziona emettendo un breve impulso ultrasonico e misurando il tempo che impiega l'eco a tornare al sensore dopo aver rimbalzato su un oggetto.

Misura distanze da circa 2 cm fino a 400 cm (4 metri) con una buona precisione, utilizza due trasduttori uno emette l'onda sonora (trasmettitore), e l'altro riceve l'eco (ricevitore).

Ha quattro pin (VCC, Trig, Echo, GND) che consentono di connetterlo facilmente a microcontrollori come Arduino o Raspberry Pi.

Il HC-SR04 è apprezzato per la sua semplicità, affidabilità, e costo contenuto, rendendolo ideale per progetti didattici e per prototipazione rapida.

## 3.3. Sensore di movimento HC-SR501

È un sensore PIR (Passive Infrared) utilizzato per rilevare la presenza o il movimento di oggetti emettenti calore, come persone o animali, all'interno di un'area specifica. Questo tipo di sensore è ampiamente utilizzato in applicazioni di sicurezza, illuminazione automatica e automazione domestica grazie alla sua affidabilità e semplicità d'uso.

Il sensore rileva la radiazione infrarossa emessa dagli oggetti caldi nel suo campo di rilevamento. Quando un oggetto caldo, come una persona, entra nel campo di rilevamento del sensore, il sensore rileva un cambiamento nei livelli di infrarossi, attivando così un segnale di uscita.

Ha un campo di rilevamento che varia da circa **3 a 7 metri** a seconda delle condizioni ambientali e dell'impostazione della sensibilità.

## 3.4. Display LCD 1602A

L'LCD 1602A è un modulo display a cristalli liquidi alfanumerico, comunemente utilizzato in progetti di elettronica per visualizzare testo. Ha un display di 16 colonne per 2 righe, il che significa che può mostrare fino a 16 caratteri per riga e dispone di due righe di testo. Questo tipo di LCD è ampiamente utilizzato con microcontrollori come Arduino, Raspberry Pi e altri sistemi embedded perché è facile da interfacciare.

## 3.5. Ricevitore infrarosso e telecomando

Ricevitore infrarosso (IR) è un componente elettronico utilizzato per rilevare segnali di luce infrarossa, emessi da un telecomando. È ampiamente utilizzato in dispositivi elettronici come televisori, lettori DVD, e sistemi di automazione domestica per ricevere comandi a distanza. Il ricevitore IR è progettato per captare segnali di luce infrarossa modulata, che vengono emessi da un LED IR di un telecomando. Questa luce è invisibile all'occhio umano, include filtri ottici per bloccare la luce visibile e un circuito elettronico che demodula il segnale ricevuto, rimuovendo la modulazione di alta frequenza e lasciando solo il segnale digitale che rappresenta il comando inviato.

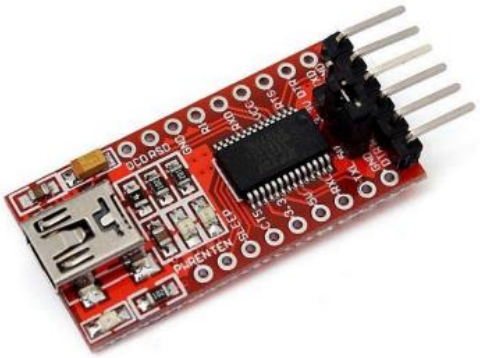

Ha tre pin:

- **VCC**: Alimentazione (5V).
- **GND**: Collegamento a terra.
- **OUT**: Uscita del segnale.

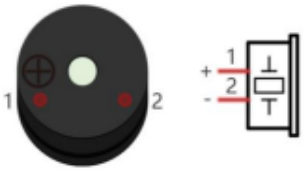
Il segnale IR ricevuto viene convertito in impulsi digitali che rappresentano codici specifici per ogni pulsante del telecomando.

## 3.6. Interfaccia seriale UART FT232RL

La USB to Serial Adapter with FT232RL permette di collegare i PC con qualsiasi sistema a microcontrollore attraverso la porta USB. Oltre ai segnali TX e RX, sono presenti anche le linee CTS, RTS e le altre linee di Handshaking. Collegando la scheda alla porta USB, il PC la riconoscerà come una porta COM attraverso la quale possiamo stabilire la connessione con il target, senza bisogno di alcuna modifica. Per il collegamento è sufficiente collegare la scheda al GND e collegare le linee segnale TX ed RX alle rispettive RX e TX del target



## 3.7. Buzzer Acustico attivo




Un **buzzer attivo** è un dispositivo elettromeccanico che emette un suono quando viene alimentato. Contiene un circuito oscillatore interno che produce la frequenza del suono. Questo significa che per far suonare un buzzer attivo, è sufficiente fornire una tensione continua (DC) al dispositivo.

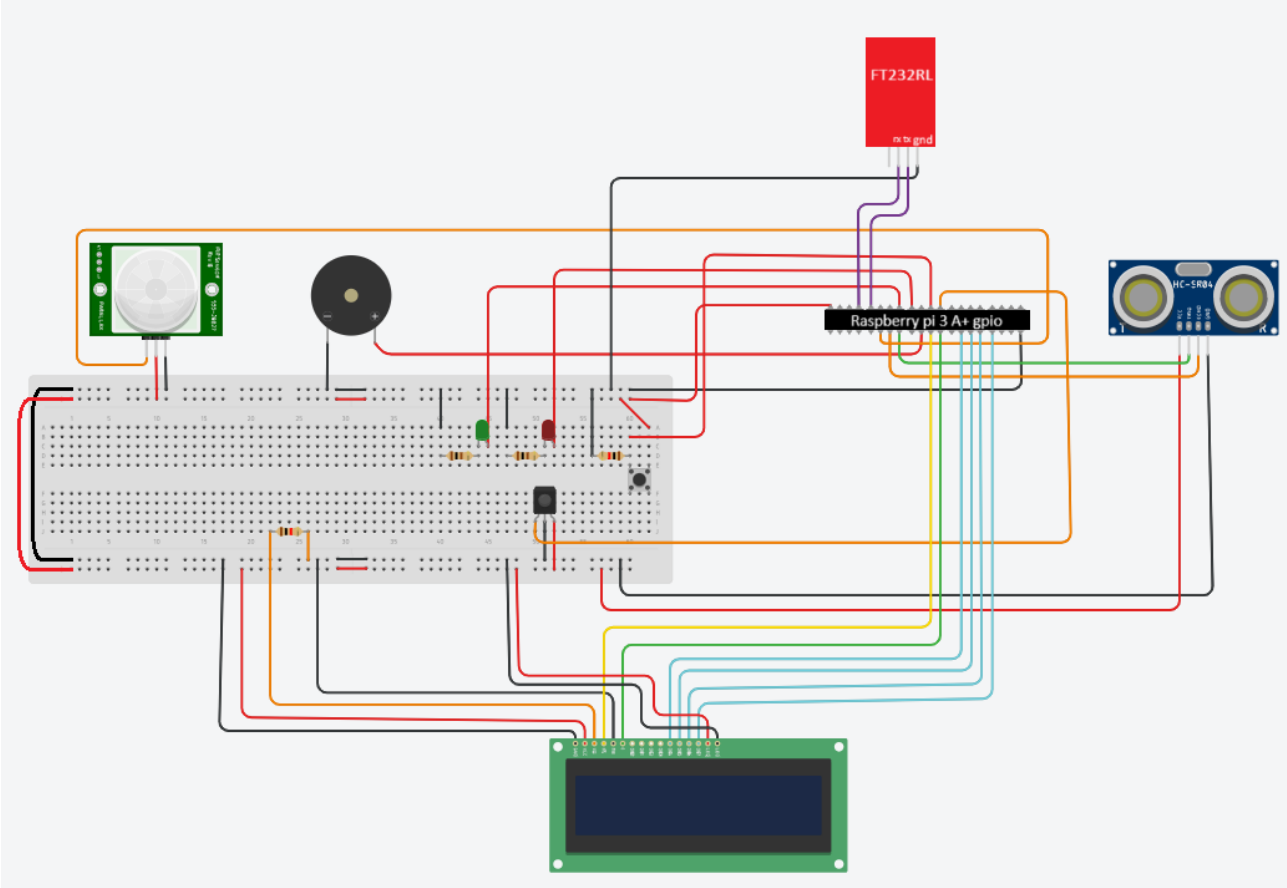
Non richiede segnali di controllo esterni o impulsi specifici, può essere facilmente collegato direttamente a una fonte di alimentazione o controllato semplicemente con un interruttore o un pin digitale. Funziona a tensioni di alimentazione che variano da 3V a 12V.

## 3.8. Led

Sono presenti, infine, due led uno rosso e uno verde ai quali sono collegati due resistori da 200 ohm.

# 4. Schema del progetto

Lo schema circuitale risulta come segue:



Collegamenti GPIO:

| CONN. | HEADER 1 | PIN | PIN | HEADER 2 | CONN. 2 |
|---|---|---|---|---|---|
| | 3V3 Power | 1 | 2 | 5V Power | VCC-BRED |
| | GPIO 2 (SDA) | 3 | 4 | 5V Power | |
| | GPIO 3 (SCL) | 5 | 6 | Ground | GND-BRED |
| | GPIO 4 (GPCLK0) | 7 | 8 | GPIO 14 (TXD) | FTDI - RX |
| | Ground | 9 | 10 | GPIO15 (RXD) | FTDI - TX |
| HC-SR501 MOV | GPIO 17 | 11 | 12 | GPIO 18 (PCM_CLK) | |
| HC-SR04 ECHO | GPIO 27 | 13 | 14 | Ground | |
| HC-SR04 TRIG | GPIO 22 | 15 | 16 | GPIO 23 | GREEN LED+ |
| | 3V3 Power | 17 | 18 | GPIO 24 | RED LED+ |
| BUZZER - VCC | GPIO10(MOSI) | 19 | 20 | GROUND | |
| LCD-RS | GPIO9(MISO) | 21 | 22 | GPIO25 | BUTTON |
| LCD-E | GPIO11(SCLK) | 23 | 24 | GPIO8(CE0) | IR |
| | GROUND | 25 | 26 | GPIO7(CE1) | |
| LCD-DB4 | GPIO 0(ID_SD) | 27 | 28 | GPIO1(ID_SC) | |
| LCD-DB5 | GPIO5 | 29 | 30 | GROUND | |
| LCD-DB6 | GPIO6 | 31 | 32 | GPIO12(PWM0) | |
| LCD-DB7 | GPIO13(PWM1) | 33 | 34 | GROUND | |
| | GPIO19(PCM_FS) | 35 | 36 | GPIO16 | |
| | GPIO 26 | 37 | 38 | GPIO 20 (PCM_DIN) | |
| | Ground | 39 | 40 | GPIO 21 (PCM_DOUT) | |

# 5. Procedura di Installazione

Prima di poter effettivamente utilizzare il Sistema sono necessari una serie di passaggi per la preparazione dell'ambiente di sviluppo sia del Target che del PC che utilizzeremo per effettuare la programmazione Interattiva utilizzando il protocollo di comunicazione UART.

## 5.1. Dispositivo Target

Avendo utilizzato un Raspberry pi 3 A+, è necessario formattare la micro SD che conterrà il sistema operativo pijFORTH. Per fare ciò è necessario usare il software Raspberry pi Imager ed installare Raspberry OS Lite.

Una volta terminata la procedura di formattazione, accedendo alla directory della micro SD, bisognerà eseguire i seguenti passaggi:

1. Eliminare tutti i file denominati come "kernelX.img" e copiare il file "kernel7.img", ridenominazione di pijForth OS.
2. Modificare il file "config.txt" aggiungendo in coda la stringa "enable uart=1" e salvare le modifiche

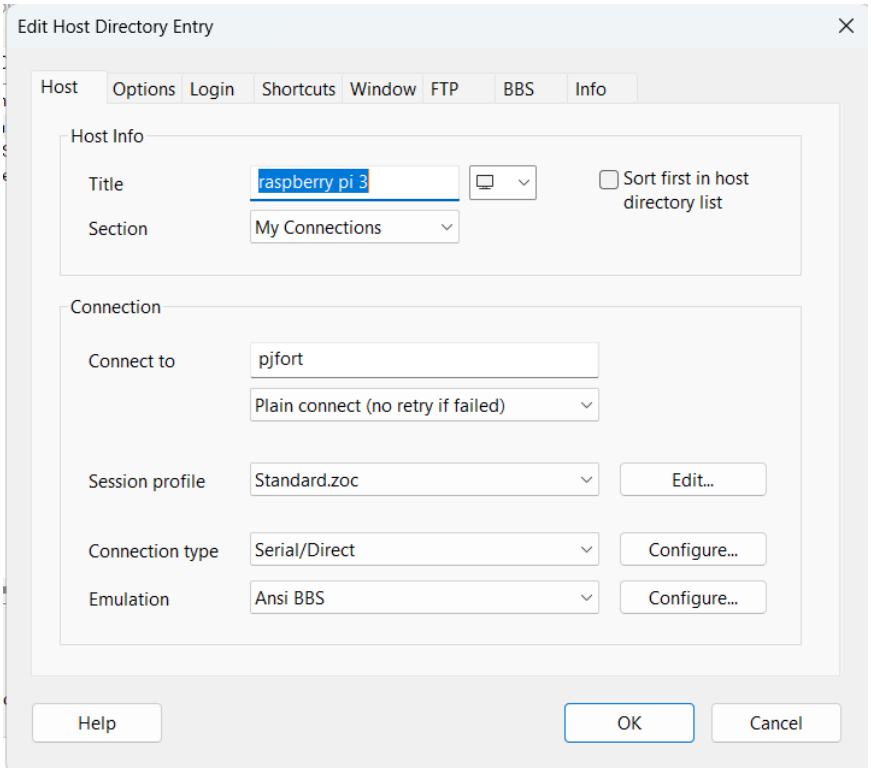
A questo punto il dispositivo di target è pronto per comunicare attraverso il protocollo UART.

## 5.2. Preparazione del PC

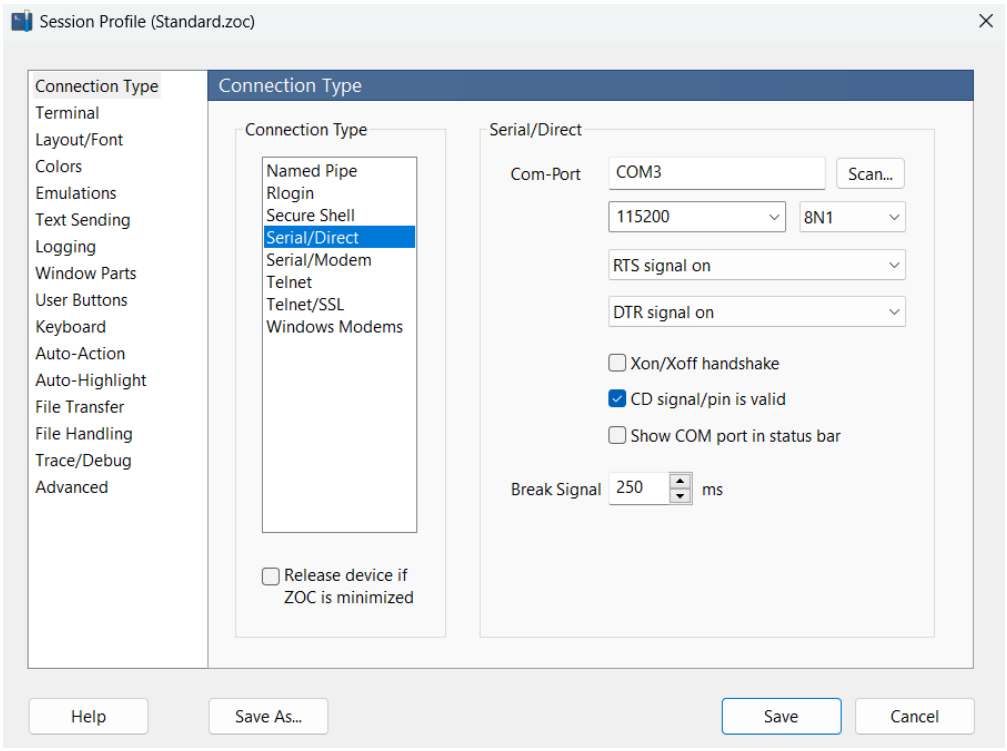
Per fare in modo che Target e PC possano comunicare, è necessario installare un software che permette la comunicazione seriale tramite protocollo UART.

Per le piattaforme Windows e MacOS è possibile utilizzare ZOC, mentre per quelle Linux è possibile utilizzare PicoCom-3.1. In questa documentazione verrà illustrata la procedura su piattaforma Windows.

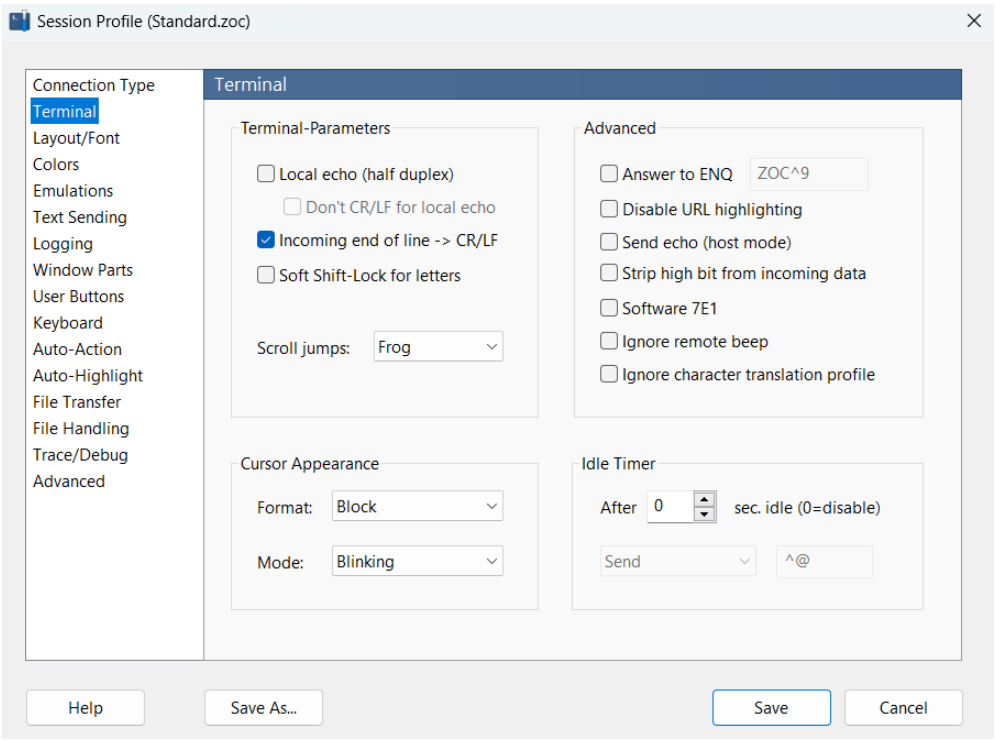
Per configurare l'ambiente ZOC creare una nuova connessione e configurarla come da immagine:




Successivamente cliccare su Edit e configurare la connessione seriale



Infine settare i parametri del terminale




Adesso collegare il dispositivo target al pc tramite la porta micro USB, avviare la connessione e alimentarlo.

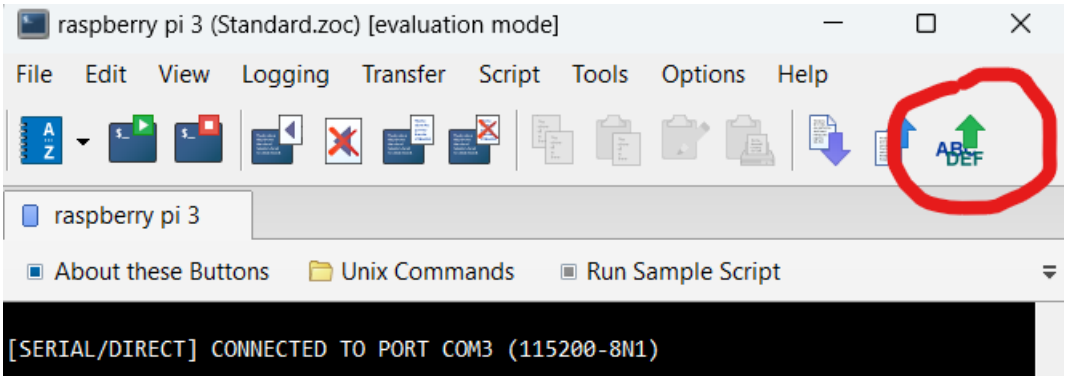
Dopo pochi secondi comparirà sul terminale la console dell'interprete

```
[SERIAL/DIRECT] CONNECTED TO PORT COM3 (115200-8N1)


PURE324ES
Palermo University 32 bit Research Environment for Embedded Systems
(2022) - Daniele Peri
v. 202201182136
PERI BASE 3F000000
TIME      00000000002BD34B
CPUID     410FD034
CPSR      600001DA
BOARDREV  009020E0
Board model       0x00000000
Board revision    0x009020E0
Board MAC address 0x0000B827EB0431F8
Board serial      0x00000000D40431F8
ARM memory base address 0x00000000, size 0x08000000
 VC memory base address 0x08000000, size 0x08000000
Clock rates (Hz):
UART:    48000000
CPU:     600000000
CPU MAX: 1400000000
CORE:    250000000
ISP:     250000000
Initializing frame buffer
framebuffer at 0x0E8FA000 width 00000400 height 00000300 depth 00000020 pitch 00
001000

Changing clock rate
clock max rate: 1400000000, clock current rate: 1400000000
Clock rates (Hz):
UART:    48000000
CPU:     1400000000
CPU MAX: 1400000000
CORE:    250000000
ISP:     300000000
pijFORTHos 0.1.8-se-20220113 sp=0x08000000
```

Dalla barra dei pulsanti selezioniamo il caricamento di un file sorgente




Caricare i file “ans.f” e “final.f”.

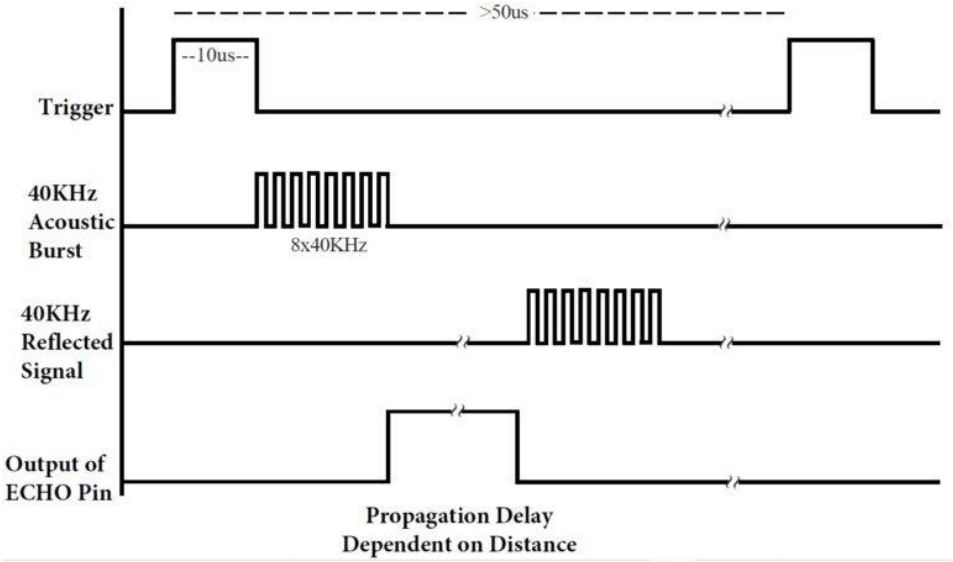
Terminato il caricamento viene lanciata automaticamente la word START che avvia il sistema.

# 6. Funzionamento del sistema

## 6.1. Sensore distanza a ultrasuoni

Il dispositivo HC-SR04 funziona nel seguente modo:

1. Si attiva il Trigger con un segnale di almeno 10 µs
2. Il sensore manda una serie di impulsi a ultrasuoni: 40 kHz
3. Successivamente alla trasmissione, il segnale di Echo viene impostato ad alto
4. Non appena il segnale ritorna al sensore, il segnale di Echo torna basso




Per utilizzare il dispositivo, viene misurato il tempo in microsecondi per il quale il segnale Echo rimane alto.

Poiché il sensore andrà posizionato in prossimità degli ingressi, affinché possa rilevare un'intrusione, viene impostata una distanza fissa, con la quale viene confrontato il valore rilevato, con un margine di errore di 32 microsecondi, che corrispondono a mezzo centimetro, secondo la formula:

$$s = \frac{v \cdot t}{2}$$

Dove $v$ è la velocità del suono, 341m/s, e $t$ è il tempo.

## 6.2. LCD

Il segnale RS (Register Select) serve a distinguere le istruzioni dai dati da stampare sullo schermo. Se è 0 lo schermo interpreta i dati in input come un'istruzione, altrimenti come un carattere.

Nella seguente tabella sono indicate le istruzioni utilizzate per il funzionamento dello schermo LCD

| INSTRUCTION | RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Clear Display | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Return Home | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x |
| Display ON/OFF | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B |
| Function Set | 0 | 0 | 0 | 0 | 1 | D/L | N | F | x | x |
| Cursor or Display Shift | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | x | x |

- **Clear Display**: serve a cancellare tutti i dati dal display e posiziona il cursore nella posizione in alto a sinistra
- **Return Home**: serve ad impostare il cursore nella posizione in alto a sinistra
- **Display ON/OFF**: se D è alto, il display è acceso, altrimenti spento; se C è alto viene mostrato il cursore, altrimenti non viene mostrato; se B è alto il cursore lampeggia, altrimenti no.
- **Function Set**: se D/L è alto, viene impostata la modalità ad 8 bit, 4 altrimenti; se N è basso viene attivata solo la prima linea del display, entrambe altrimenti; F regola il formato, ma se N è impostato a 1 non viene considerato
- **Cursor or Display Shift**: se S/C è basso viene spostato solo il cursore, altrimenti il display; se R/L è alto viene effettuato uno spostamento a destra, a sinistra altrimenti

Per utilizzare il dispositivo è stata impostata la modalità a 2 linee e a 4 bit, ed è stato disattivato il cursore.

## 6.3. Ricevitore a infrarossi

Il trasmettitore a infrarossi, per decodificare il tasto premuto, invia un segnale a 32 bit, e il ricevitore lo restituisce modificando nel tempo lo stato del pin DATA.

Per interpretare l'output, abbiamo campionato il segnale con una frequenza di 20 kHz, e misurato per quanto tempo il pin rimanesse alto. Così facendo risultano due stati: in uno stato il pin rimane alto per un range compreso tra 250 e 750 microsecondi, nel secondo per un range compreso tra 1100 e 1800 microsecondi.

Abbiamo assegnato al primo stato 0 e all'altro 1.

È risultata la seguente codifica:

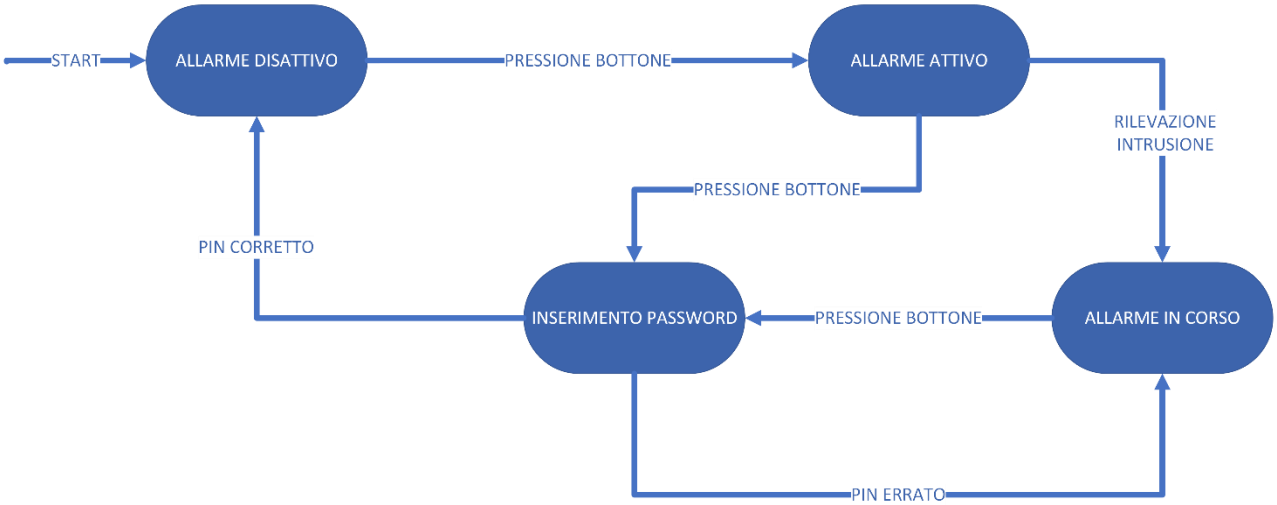
| | |
|---|---|
| 0 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 1 1 |
| 1 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 |
| 2 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 |
| 3 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1 |
| 4 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 1 0 1 1 1 1 |
| 5 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1 1 |
| 6 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0 1 |
| 7 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 0 1 |
| 8 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 |
| 9 | 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 |

Tutti i tasti hanno in comune i 16 bit più significativi, dei quali la prima metà 0 e la seconda a 1. Così abbiamo utilizzato la maschera 0x00FF0000 per discriminare la pressione di un tasto.

Poiché i bit arrivano in forma seriale, per memorizzarli viene shiftato a sinistra di una posizione il valore attuale del tasto e aggiunto il nuovo bit finché non viene corrisposta la maschera sopracitata.

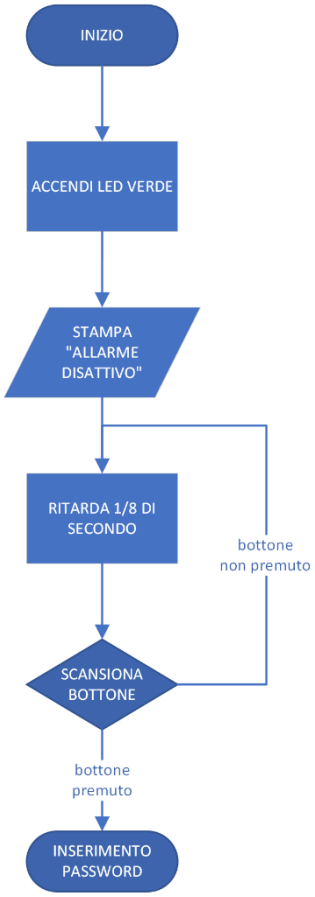
# 7. Diagrammi di funzionamento

Diagramma a stati finiti

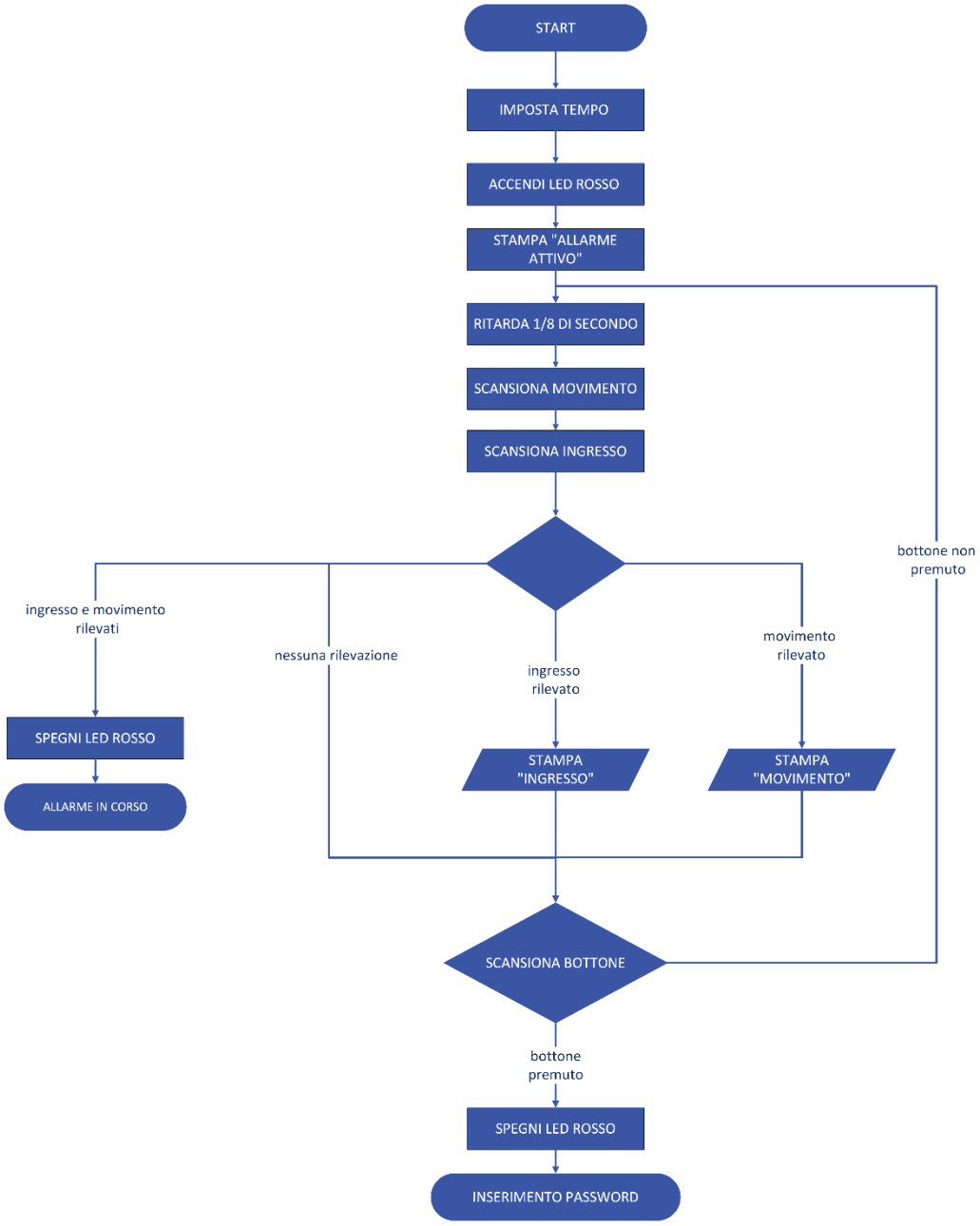



## 7.1. Diagrammi di flusso

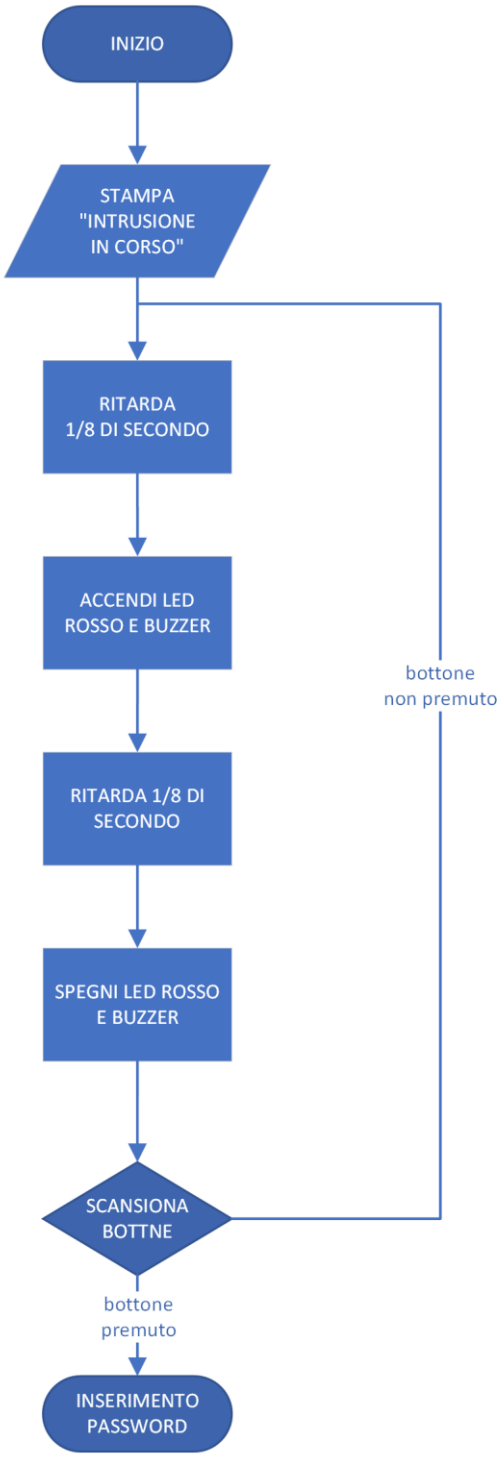
Allarme disattivo



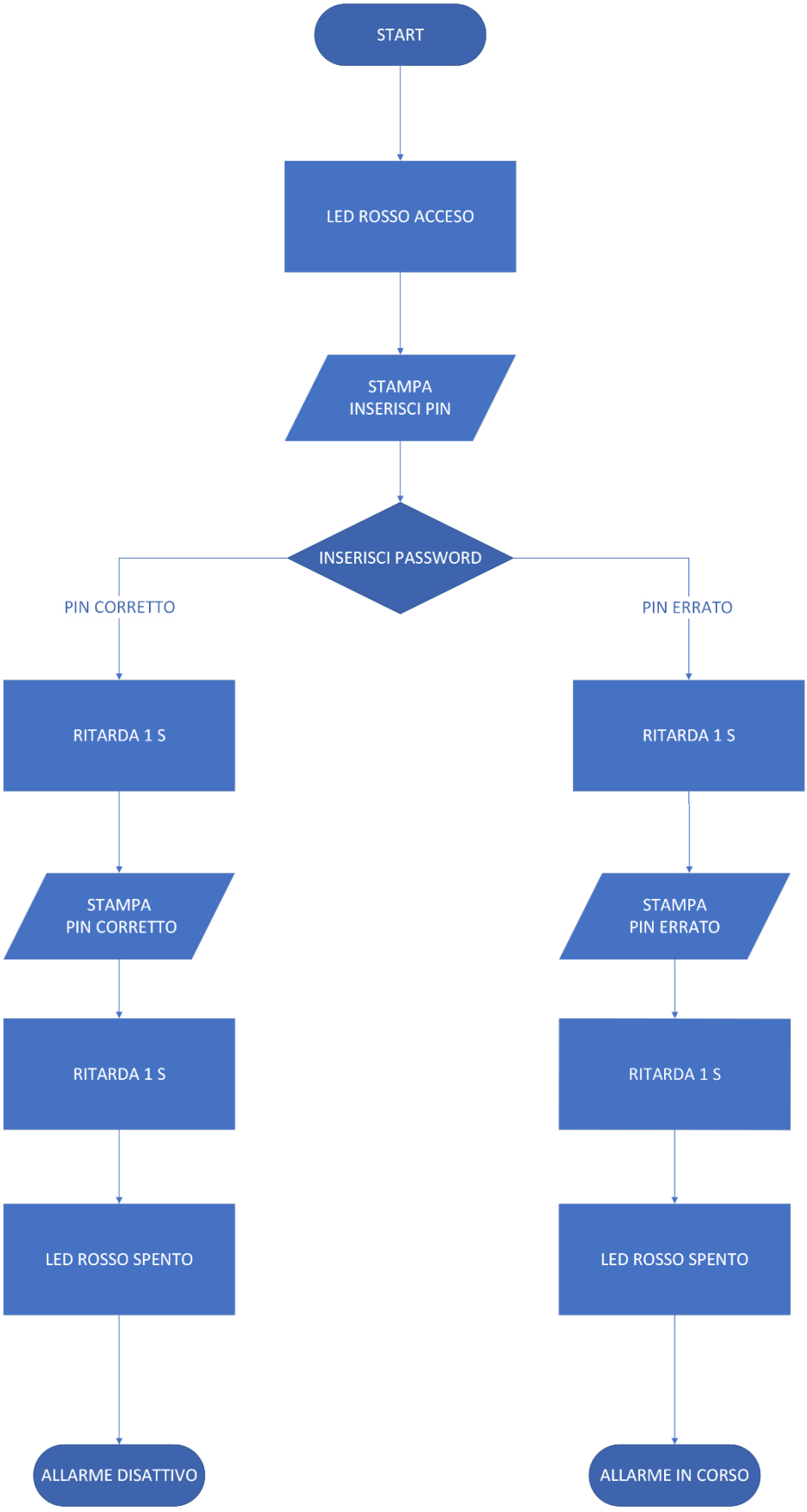
Allarme attivo



Allarme in corso



Inserimento password



# 8. Documentazione dizionario

## Indirizzi di base

- **BASE**: costante che memorizza l'indirizzo di base del primo registro, GPFSEL0. Gli indirizzi degli altri registri sono ricavati aggiungendo un off-set di 4 all'indirizzo base.
- **TIMER_BASE**: costante che memorizza l'indirizzo base dei registri utilizzati per la gestione del tempo.

## Word di utilità

- **OUTPUT**: ( -- cod) costante che memorizza la codifica della funzione output per i registri GPSEL
- **ABS**: (num -- abs_val) restituisce il valore assoluto del numero in input
- **BTST**: (num, mask_bit -- bool) verifica se il bit indicato nella maschera è impostato a 1 nel numero in input, ritornandone il booleano corrispondente.
- **ON**: (mask -- ) imposta a 1 il bit indicato dalla maschera nel registro GPSET0
- **OFF**: (mask --) imposta a 1 il bit indicato dalla maschera nel registro GPCLR0
- **READ_TIME**: ( -- ) restituisce il numero di microsecondi trascorsi dall'accensione del sistema
- **DELAY**: (micro_sec -- ) mette in attesa il sistema finché non trascorre un numero di microsecondi maggiore a quello in input
- **1/8_SEC**: ( -- ) mette in attesa il sistema per un ottavo di secondo
- **1/2_SEC**: ( -- ) mette in attesa il sistema per mezzo secondo
- **1_SEC**: ( -- ) mette in attesa il sistema per un secondo

## Impostazione GPIO in modalità output

- **MASK**: (num, cod -- mask ) genera una maschera shiftando di 3 posti a sinistra il codice ricevuto in input per un numero di volte pari al modulo 10 del numero in input
- **OUT**: ( num -- mask ) genera la maschera della funzionalità di output per il numero di gpio indicato in input
- **SELECT**: ( mask, reg -- ) applica la maschera generata data in input all'indirizzo del registro dato in input
- **GPIOn_OUT**: ( -- ) imposta l'n-esimo GPIO in modalità output

## Setting componenti di output

- **LCD_SET** imposta i pin riferiti ai GPIO 0, 5, 6, 9, 11, 13 come output per il controllo del display LCD 1602A

- **BUZZER_SET** configura il pin riferito al GPIO 10 come output per il buzzer
- **TRIG_SET** configura il pin riferito al GPIO 22 come output per il trigger del sensore di distanza HC-SR04
- **LED_SET** configura i pin riferiti ai GPIO 23 e 24 come output per i due LED

## Costanti maschere output

- **GREEN** è la maschera che permette di valutare il pin, corrispondente al LED verde, confrontandolo con il contenuto di GPSET0/GPCLR0
- **RED** è la maschera che permette di valutare il pin, corrispondente al LED rosso, confrontandolo con il contenuto di GPSET0/GPCLR0
- **BUZZER** è la maschera che permette di valutare il pin, corrispondente al Buzzer, confrontandolo con il contenuto di GPSET0/GPCLR0
- **TRIGGER** è la maschera che permette di valutare il pin, corrispondente al trigger del sensore di distanza HC-SR04, confrontandolo con il contenuto di GPSET0/GPCLR0
- **EN** è la maschera che permette di valutare il pin, corrispondente al segnale di Enable del display LCD 1602, confrontandolo con il contenuto di GPSET0/GPCLR0
- **RS** è la maschera che permette di valutare il pin corrispondente al segnale RS confrontandolo con il contenuto di GPSET0/GPCLR0
- **DB4** è la maschera che permette di valutare il pin corrispondente al segnale DB4 confrontandolo con il contenuto di GPSET0/GPCLR0
- **DB5** è la maschera che permette di valutare il pin corrispondente al segnale DB5 confrontandolo con il contenuto di GPSET0/GPCLR0
- **DB6** è la maschera che permette di valutare il pin corrispondente al segnale DB6 confrontandolo con il contenuto di GPSET0/GPCLR0
- **DB7** è la maschera che permette di valutare il pin corrispondente al segnale DB7 confrontandolo con il contenuto di GPSET0/GPCLR0

## Costanti maschere input

- **MOV_MASK** è la maschera che permette di valutare il pin, corrispondente al GPIO17 il quale è connesso al sensore di movimento HC-SR501, confrontandolo con il contenuto di GPLEV0
- **BUTTON_MASK** è la maschera che permette di valutare il pin, corrispondente al GPIO25 il quale è connesso al bottone, confrontandolo con il contenuto di GPLEV0
- **ECHO_MASK** è la maschera che permette di valutare il pin, corrispondente al GPIO27 il quale è connesso al sensore di distanza HC-SR504, confrontandolo con il contenuto di GPLEV0

- **IR_MASK** è la maschera che permette di valutare il pin, corrispondente al GPIO8 il quale è connesso al ricevitore a infrarossi, confrontandolo con il contenuto di GPLEV0
- **MASK_TASTO** è la maschera che permette di valutare se il segnale inviato dal telecomando al ricevitore infrarossi è un tasto

## Lettura dei pin di input

- **READ_PIN**: (mask -- bool) verifica lo stato di un pin guardando dentro il registro GPLEV0 e confrontandolo con la maschera corrispondente in input

### Setting sensore distanza

- **TIME** è la variabile di riferimento per memorizzare il tempo
- **TRIG**: ( -- ) permette di inviare un impulso di 10 microsecondi al pin corrispondente al trigger del sensore di distanza
- **WAIT_UP**: ( -- ) attende finché il pin corrispondente al segnale di Echo del sensore di distanza non è attivo
- **ECHO_TIME**: ( -- time) attende che il segnale di Echo sia alto e successivamente calcola il tempo, in microsecondi, fino a quando non diventa basso
- **SET_TIME**: ( -- ) imposta la variabile TIME con il valore restituito da ECHO_TIME

## Scansione dei sensori

- **SCAN_DIST**: ( -- bool) verifica se il sensore di distanza abbia rilevato un'intrusione, valutando se la differenza tra il tempo di ritorno del segnale Echo e il valore della variabile time è maggiore di 32 microsecondi restituendo 1 se vero altrimenti 0
- **SCAN_MOV**: ( -- bool) verifica se il sensore di movimento abbia rilevato un movimento leggendo il pin corrispondente ad esso
- **SCAN_BUTTON**: ( -- bool) verifica se è stato premuto il bottone leggendo il pin corrispondente ad esso

## Setting LCD

- **MOD_INSTR**: ( -- ) permette di impostare il pin RS (Register Select) a 0 quindi il display interpreta i dati che riceve come istruzioni di controllo

- **MOD_DATA**: ( -- ) permette di impostare il pin RS a 1 quindi il display interpreta i dati ricevuti come dati da visualizzare sullo schermo
- **CLEAR**: ( -- ) disattiva tutti i 4 pin di dati del display
- **SWITCH**: (mask, bool -- ) cambia lo stato di un pin rispetto il booleano ad input
- **SET_DB**: (data -- ) configura i pin di dati (DB4, DB5, DB6, DB7) impostandoli al valore corrispondente all'input
- **WRITE**: (data -- ) permette di scrivere un byte di dati sul display, inviando prima i 4 bit più significativi e successivamente quelli meno significativi abilitando e disabilitando il segnale di Enable tra un gruppo di 4 bit e l'altro
- **SETUP_LCD**: ( -- ) inizializza il display inviando due istruzioni, la prima attiva il cursore nel display mentre la seconda imposta la modalità a 4 bit per il display ed abilita la seconda riga del display, in fine imposta la modalità dati

## Words LCD

- **'char'**: ( -- ) tutte le parole con gli apici ad inizio e fine di esse permettono di inviare l'ASCII corrispondente al char ,al display e visualizzarlo
- **PAROLA**: ( -- ) permette di scrivere la parola mostrandola nel display
- _ : ( -- ) permette di inserire un carattere vuoto nel display
- **LEFT**: ( -- ) permette di spostare il cursore di una posizione a sinistra
- **CLEAR_DISPLAY**: ( -- ) permette di cancellare il display
- **NL**: (len_string -- ) permette di andare a capo valutando quanti spazi stampare a schermo in base alla lunghezza ad input, considerando che una riga del display può contenere massimo 40 caratteri
- **CANC**: ( -- ) permette di cancellare un carattere dal display spostando il cursore indietro di una posizione e poi ponendo un carattere vuoto al suo posto
- **CL**: (len_string -- ) permette di cancellare una stringa di una certa lunghezza data ad input dal display
- **COUNTDOWN**: ( -- ) permette di avviare il countdown dopo aver premuto il bottone di attivazione d'allarme, facendo lampeggiare il led verde, suonare il buzzer e stampare a schermo i valori da 9 a 0 prima della effettiva attivazione dell'allarme

## Setting IR

- **TASTO**: variabile che memorizza la codifica esadecimale del tasto premuto
- **INIT_TASTO**: ( -- ) inizializza la variabile TASTO a 0
- **MEASURE_TIME**: ( -- time) misura per quanti microsecondi il segnale mandato dal ricevitore a infrarossi rimane basso

- **RANGE**: (time -- state) confronta il tempo in input con due range e, se è racchiuso nel primo ritorna 0, se è racchiuso nel secondo ritorna 1, mentre se non è racchiuso in nessuno dei due ritorna il tempo originale
- **SET_TASTO**: (bit -- ) accoda il bit ricevuto in input al valore presente nella variabile TASTO e memorizza il nuovo valore
- **FIND_INPUT**: (state -- bool) se lo stato in input è 1 o 0, aggiorna la variabile TASTO e verifica se sia stato premuto un tasto valido confrontandone il valore con la maschera MASK_TASTO, ritornandone il booleano corrispondente. Se in input viene ricevuto un qualsiasi altro valore, allora ritorna falso.
- **TASTO_IR**: ( -- ): richiama la word INIT_TASTO e aggiorna continuamente la variabile TASTO finché FIND_INPUT non ritorna vero, quindi finché non viene premuto un tasto.
- **TASTO_0 ... TASTO_9**: (hex -- ) codifica in esadecimale del tasto corrispondente ai numeri da 0 a 9 e nel caso in cui lo riconosca, lo stampa nello schermo LCD

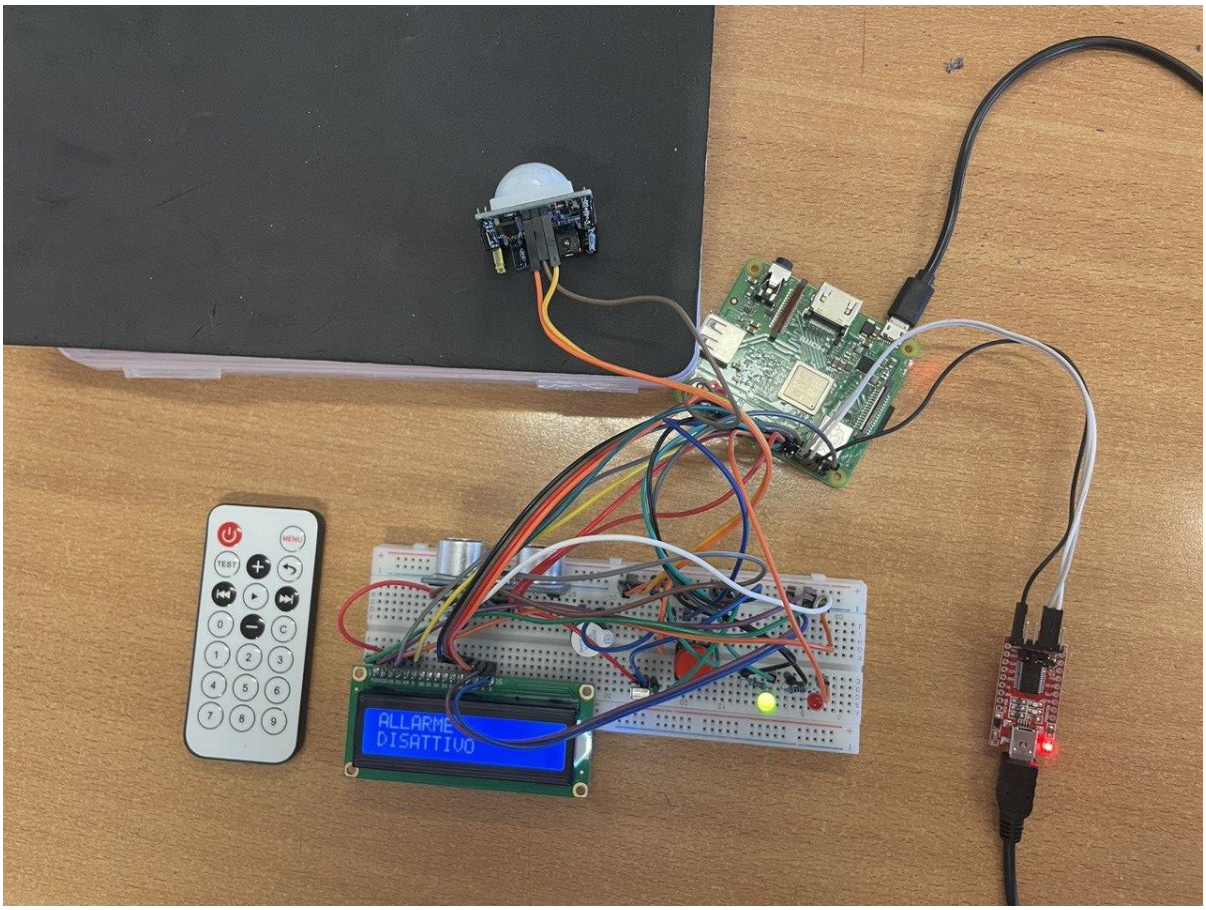
## Word stampe

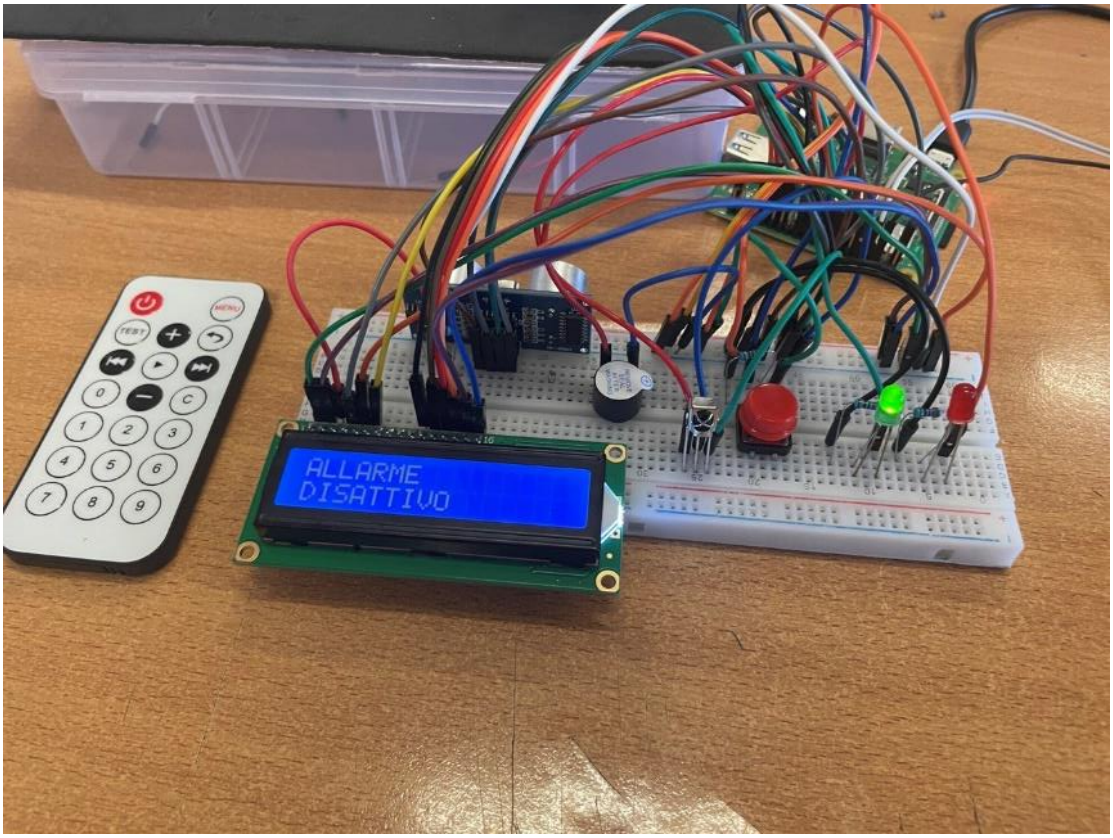
- **FLAG_MOV**: variabile che memorizza lo stato della flag relativa al rilevamento del movimento
- **FLAG_INGR**: variabile che memorizza lo stato della flag relativa al rilevamento dell'ingresso
- **PRINT_INGR**: ( -- ) se è stato rilevato un ingresso stampa nello schermo LCD la parola "INGRESSO" e la cancella quando non è più rilevato
- **PRINT_INGR**: ( -- ) se è stato rilevato un movimento stampa nello schermo LCD la parola "MOVIMENTO" e la cancella quando non è più rilevato
- **PRINT_PRESS**: ( -- ) stampa nello schermo LCD quale tasto è stato premuto, qualora sia un numero

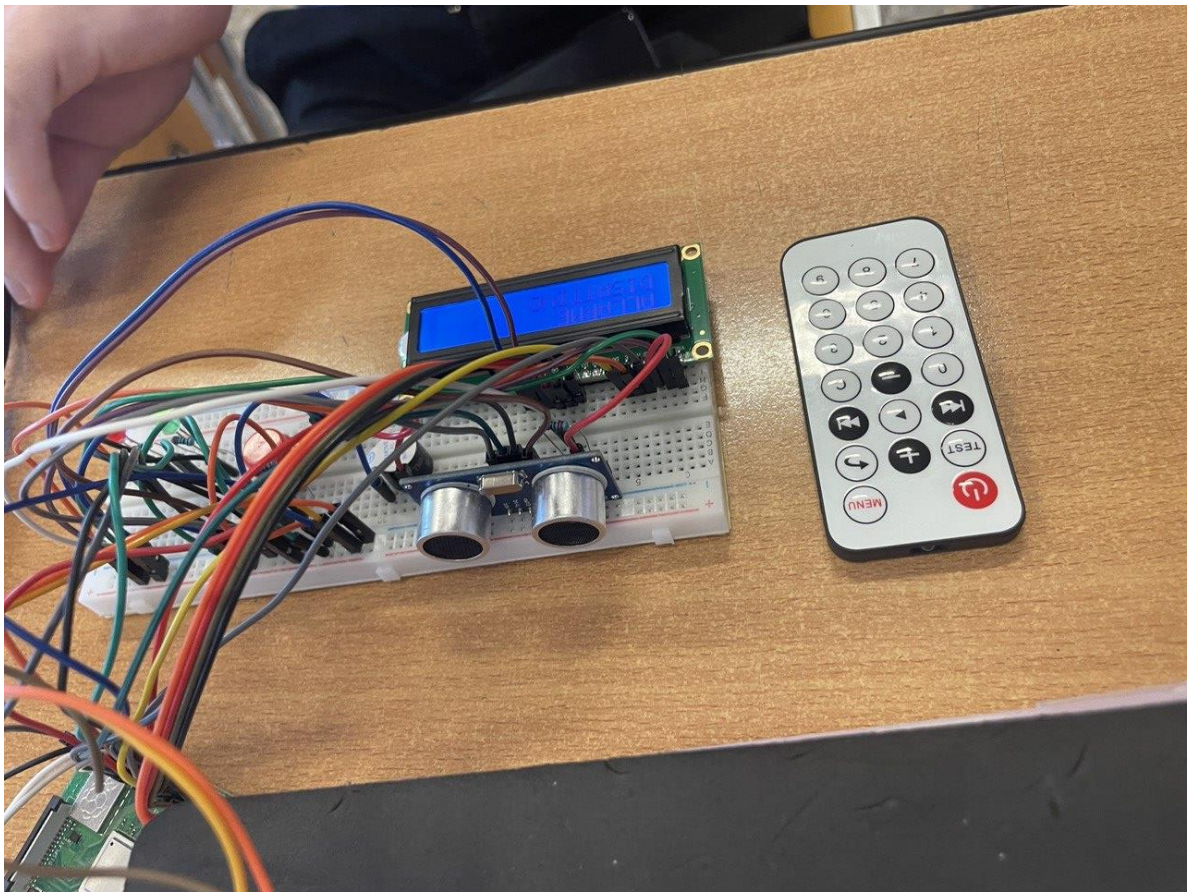
## Controllo password

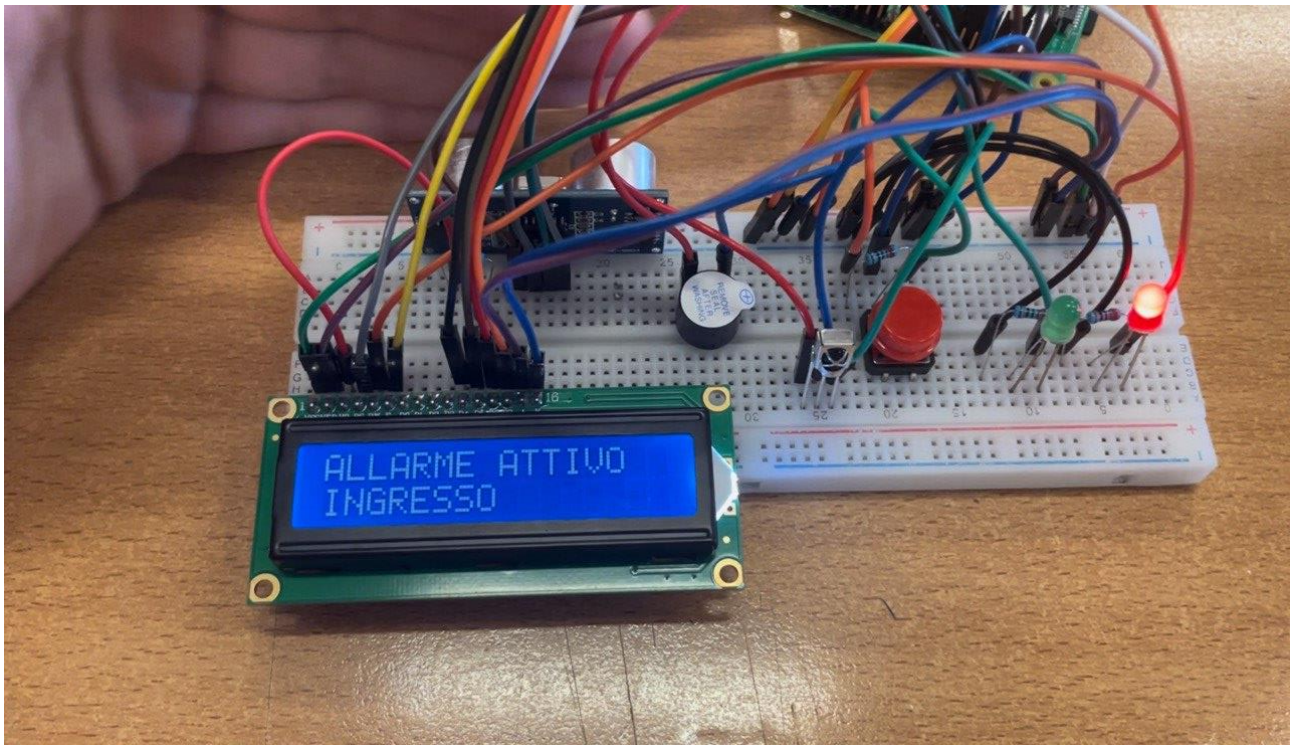
- **FLAG_PASS**: ( -- ) variabile che memorizza lo stato della flag relativa al non rilevamento della password
- **PASSWORD**: array che memorizza la sequenza di numeri corretta
- **SET_FLAG_PASS**: (bool -- ) imposta la flag al valore dato in input
- **INSERISCI_PASS**: ( -- ) inizializza la variabile FLAG_PASS a true e rileva 5 tasti. Qualora dovesse essere rilevato un tasto che non fa parte della password, la FLAG_PASS viene impostata a false.

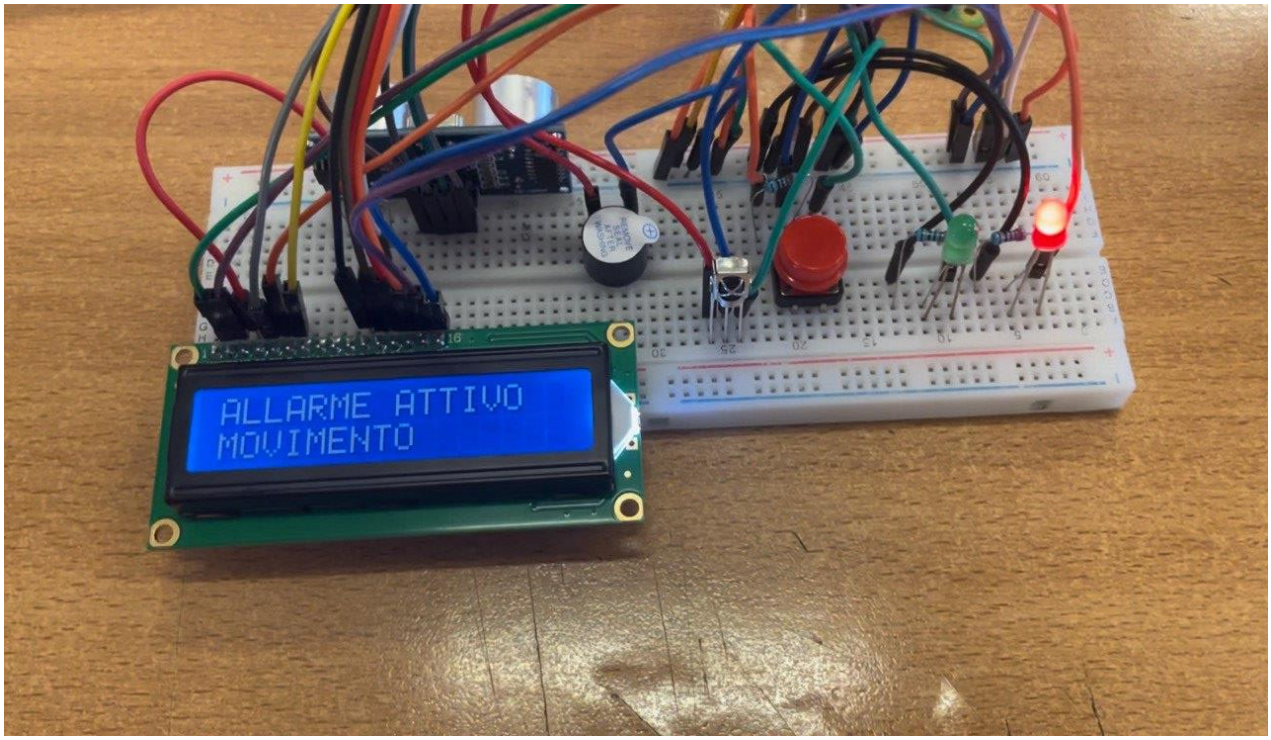
## 9. Foto

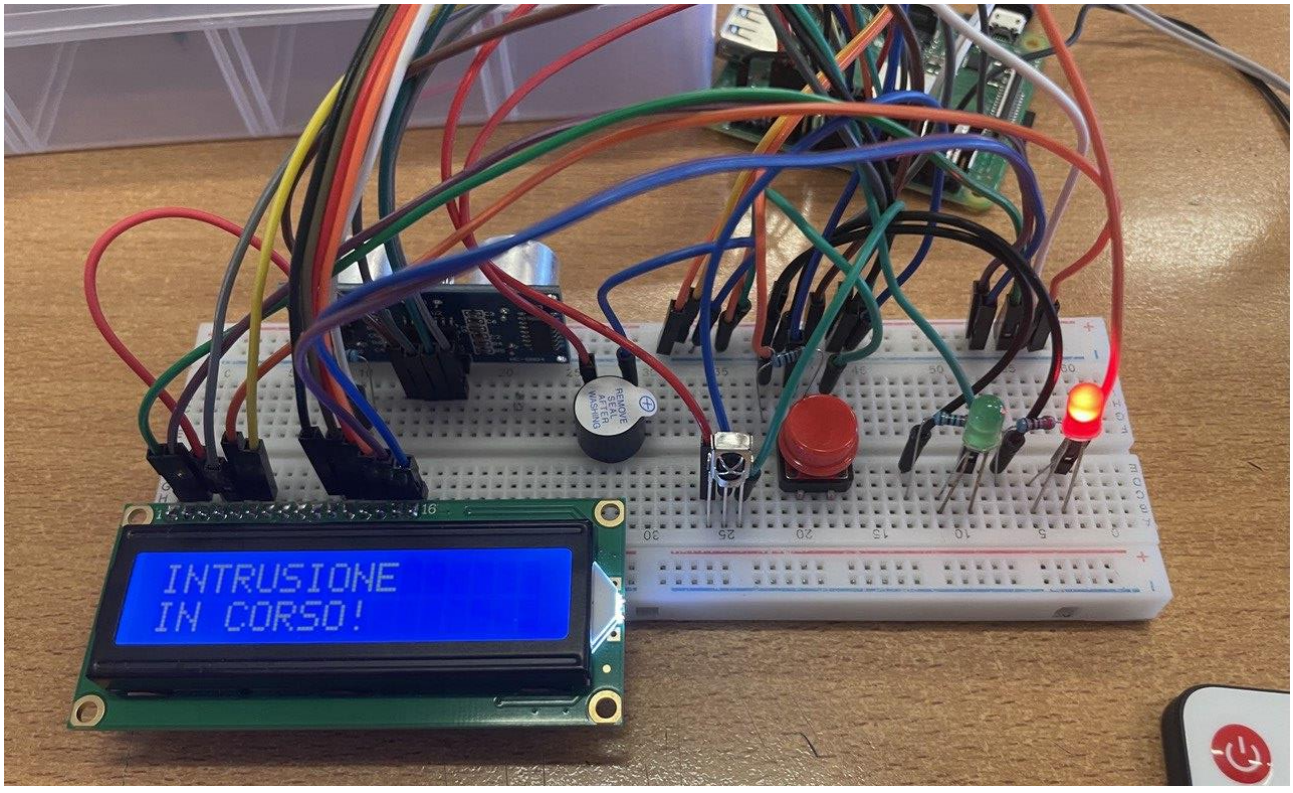


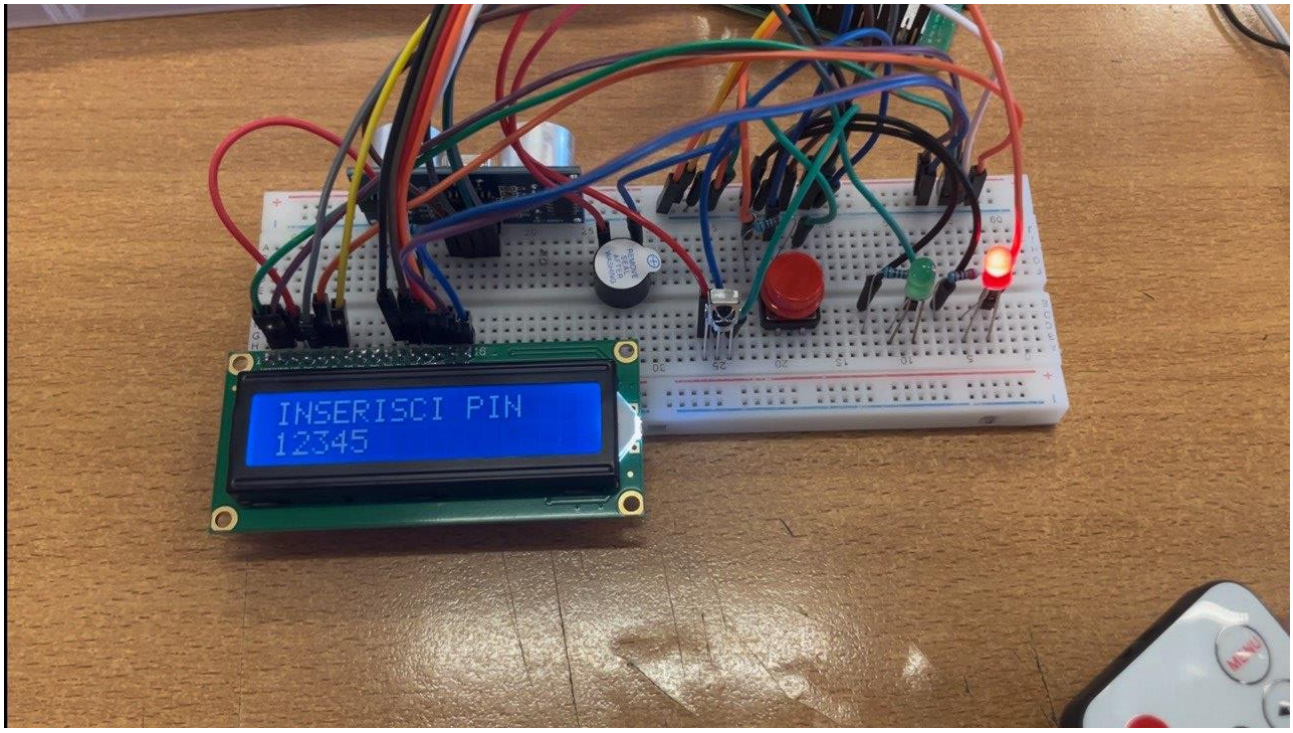



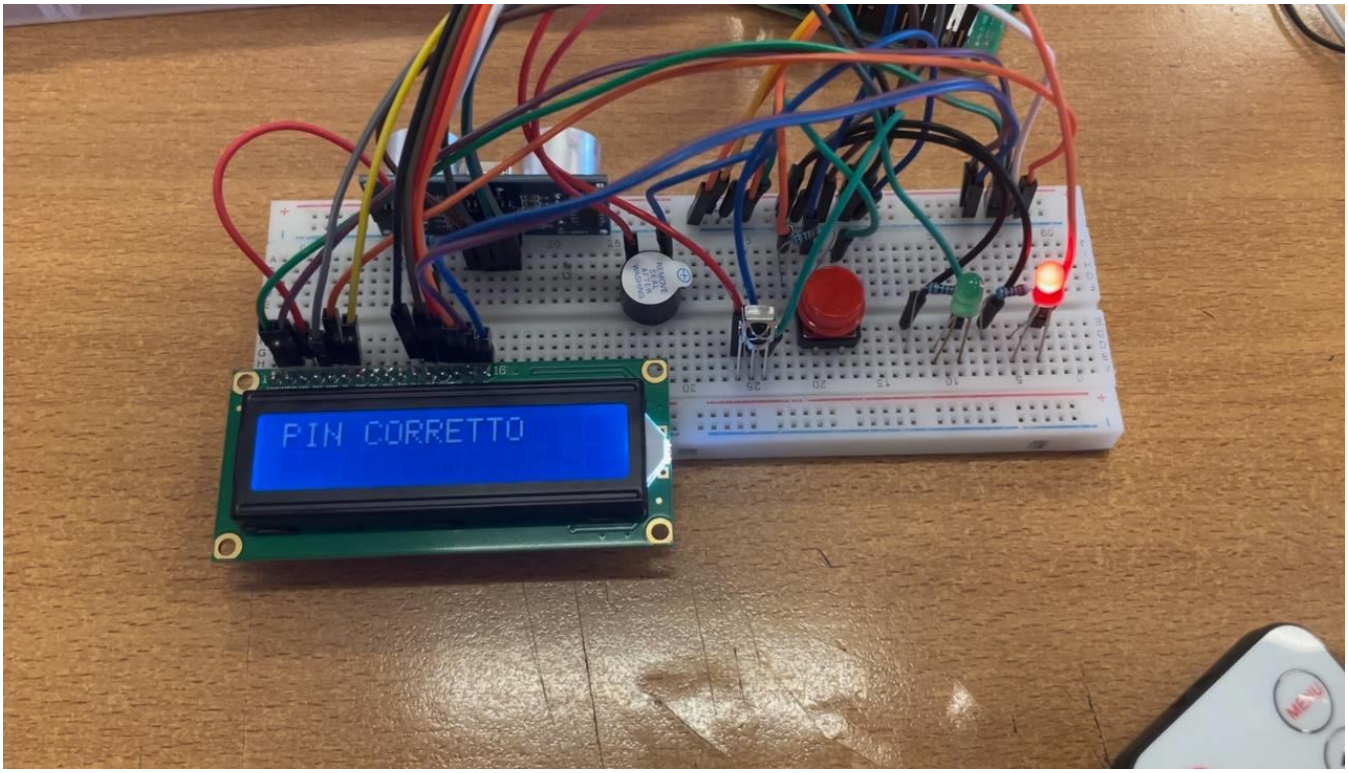

TEST
MENU


ALLARME ATTIVO
INGRESSO


ALLARME ATTIVO
MOVIMENTO


INTRUSIONE
IN CORSO!


INSERISCI PIN
12345


PIN CORRETTO