RASPBERRY INTRUDER ALARM

Sistema di allarme su Raspberry Pi programmato tramite linguaggio Forth

UNIVERSITÀ DEGLI STUDI DI PALERMO
EMBEDDED SYSTEMS
Domenico Giosuè | Prof. Daniele Peri

Sommario

1. Introduzione

Il progetto in questione rappresenta un prototipo di Sistema di Allarme controllato tramite il Single Board Computer Raspberry Pi 3 Model A+

Il sistema proposto è dotato di un sensore HC-SR501 in grado di rilevare il movimento consentendo, di conseguenza, l'attivazione dell'allarme.
Il resto delle componenti, che verranno descritte in seguito, consentirà il controllo del Sistema e il funzionamento dell'Allarme.

Il sistema può essere suddiviso in cinque differenti stati principali:

- Start State
    - Il Sistema è stato appena avviato e rimane in attesa dell'inserimento di una password da parte dell'utente.
      Questo stato può essere riconosciuto dalla luce Viola accesa.

- Idle State
    - Il Sistema è in funzione, ma l'allarme non è impostata.
      Tramite la pressione del pulsante bianco, per un periodo di 3 secondi, è possibile determinare l'attivazione dell'Alert State o la reimpostazione della Password tramite Start State.
      In particolare, le suddette opzioni verranno selezionate tramite la pressione di due differenti tasti del tastierino numerico presente nel dispositivo.

- Alert State
    - Il Sistema è in funzione e l'allarme è impostata.
      In questo stato, il sensore HC-SR501 invierà un segnale nel caso in cui dovesse rilevare un movimento, consentendo al Sistema di passare allo stato di allarme.
      Questo stato può essere riconosciuto dal LED Verde acceso.
      È possibile passare all'Idle State tramite la pressione del pulsante bianco.

- Alarm State
    - Il Sistema ha rilevato un movimento e ha avviato l'allarme.
      In questo stato un Buzzer Attivo emetterà dei suoni intermittenti e, contemporaneamente, un modulo KY-016 alternerà i colori Rosso e Blu del LED.
      Per disattivare l'allarme è possibile tenere premuto il pulsante bianco fin quando il Sistema non si troverà in Security State

- Security State
    - Il Sistema è prossimo alla disattivazione dell'allarme ma richiede l'inserimento della Password registrata dall'utente durante lo Start State.
      Il Buzzer Attivo emetterà un suono continuo e un LED Rosso rimarrà acceso fino all'inserimento della Password corretta che porterà all'Idle State.

2. Hardware Utilizzato

Di seguito viene fornita una lista che elenca tutti i componenti hardware utilizzati per la realizzazione del prototipo.
Si tenga presente che alcuni dei suddetti verranno dettagliati al fine di migliorare la comprensione del codice utilizzato durante la fase di programmazione del sistema.

- N.1 Raspberry Pi 3 Model A+
- N.1 Breadboard
- N.1 Adattatore USB-UART
- N.1 Buzzer Attivo TMB12A05
- N.1 Modulo KY-016 LED RGB
- N.1 LED Giallo
- N.1 Push Button
- N.1 Resistenza 1kΩ
- N.1 Resistenza 220Ω
- N.1 Sensore di Movimento HC-SR501 PIR
- N.1 Push Button Matrix
- N.25 Cavi M-F
- N.5 Cavi F-F
- N.3 Cavi M-M

2.1. Dettagli Sensore di Movimento HC-SR501 PIR

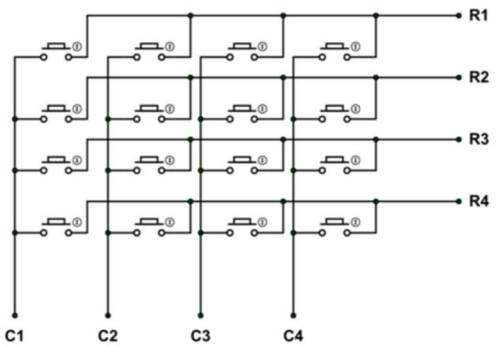
Il modulo HC-SR501 utilizza un Sensore a Infrarossi Passivo LHI778 e un Controllore BISS0001, fornendo in output una tensione di 3.3V qualora questo rilevasse un movimento.

Il modulo consente di selezionare il range di rilevamento in valori compresi tra 3 e 7 metri. Similmente, è possibile impostare per quanto tempo il valore di output rimane a 3.3V, in seguito alla rilevazione del movimento, in un range che va dai 5 secondi ai 5 minuti.

È necessario tenere in considerazione il tempo di inizializzazione del dispositivo, che si aggira attorno al minuto, prima di utilizzare i segnali di output da esso prodotti.

2.2. Dettagli Push Button Matrix

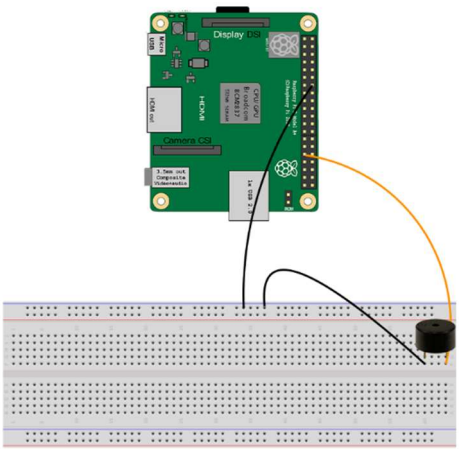
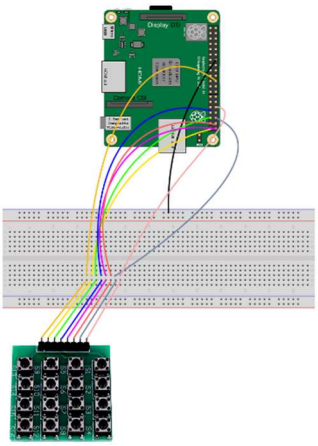
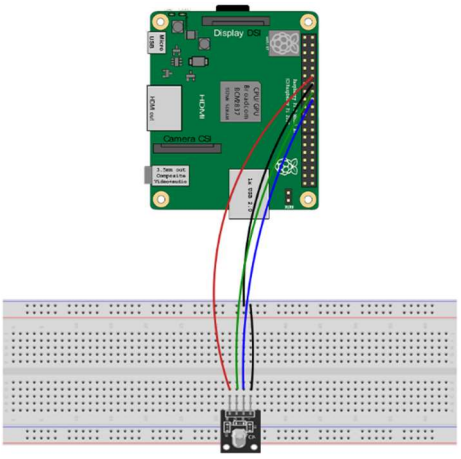
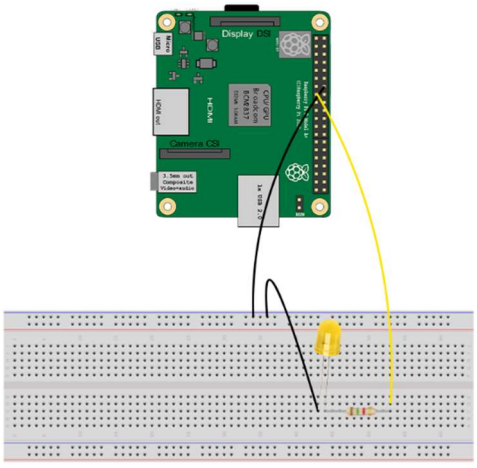
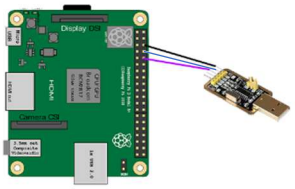
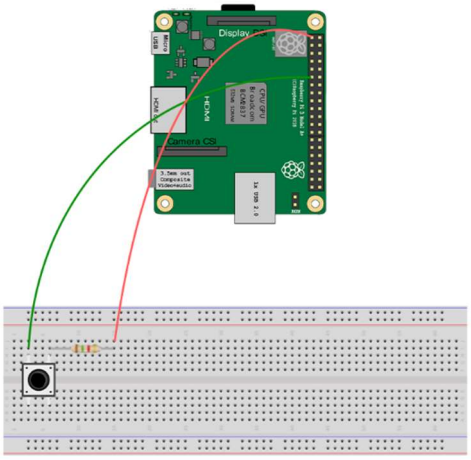
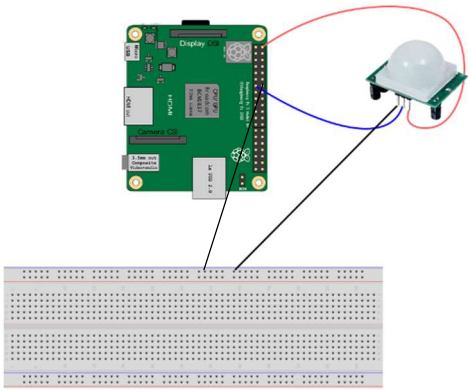
Questo modulo non è altro che una matrice di pulsanti collegati tra di loro in righe e colonne, come rappresentato in figura.



Per rilevare il tasto premuto è necessario alternare le righe e le colonne come sorgenti di input e di output, al fine di rilevare in un primo momento la colonna a cui appartiene il tasto pressato e, infine, la sua riga di appartenenza.

3. Circuito

Questa sezione mostra lo schema di collegamento dei singoli dispositivi al Raspberry Pi.
Si consideri che lo schema proposto è relativo allo schema di collegamento del prototipo che si appoggia a una breadboard ed è privo di saldature.

3.1. Tabella dei Collegamenti

DISPOSITIVO	GPIO	PIN
Buzzer Attivo TMB12A05	06	GND
Push Button Matrix Rows	12-16-20-21	-
Push Button Matrix Columns	13-19-25-26	-
Modulo KY-016 LED RGB	18-23-24	GND
LED Giallo	22	GND
Adattatore USB-UART	14-15	GND
Push Button	17	3V3
Sensore PIR HC-SR501	27	5V-GND

4. Ambiente di Lavoro

Questa sezione descriverà l'ambiente di lavoro utilizzato per la realizzazione del progetto, dettagliando tutte le fasi relative all'impostazione di quest'ultimo.

4.1. Macchina Target

In primo luogo, è necessario predisporre la nostra macchina target per consentire l'esecuzione dell'ambiente interattivo Forth.
In particolare, il suddetto ambiente consentirà di effettuare il test del codice in modo interattivo.

Nel nostro caso utilizzeremo un Raspberry Pi 3 Model A+ che monta un SoC Broadcom BCM2837B0.

4.1.1. File Necessari

Per eseguire l'ambiente interattivo Forth è necessario preimpostare una scheda micro-SD da inserire nel nostro Raspberry Pi.
Questa scheda, formattata in formato FAT32, dovrà contenere i seguenti file al fine di consentire l'operazione di boot con conseguente esecuzione dell'ambiente Forth.

- bootcode.bin
    - Bootloader che, leggendo il contenuto del file di configurazione, imposta una serie di parametri

- fixup.dat
    - Contenente codice di inizializzazione della fase di boot

- start.elf
    - Second-stage bootloader che si occuperà di cercare e avviare il file immagine contenente il codice eseguibile per la CPU ARM

- Kernel7.img
    - File immagine del sistema interattivo Forth
    - Nel nostro caso utilizzeremo una versione modificata del file 'pijFORTHos' fornitaci dal Prof. D. Peri e rinominata come 'kernel7.img'

- config.txt
    - File di configurazione contenente differenti impostazioni
    - Nel nostro caso, per consentire la comunicazione tra la macchina target e il nostro terminale, sarà necessario abilitare la comunicazione UART eliminando il simbolo di commento '#' nella riga 'enable_uart=1'

4.2. Macchina Esterna

Per quanto riguarda la macchina che utilizzeremo per interfacciarci con la macchina target, questa dovrà essere predisposta alla comunicazione effettuata tramite l'adattatore USB-UART che consentirà di sfruttare una porta USB (sicuramente più comune nelle macchine General Purpose) per avviare una comunicazione Seriale UART.

Riguardo il Sistema Operativo, nel nostro caso, utilizzeremo Windows 11 con il sottosistema Linux WSL, le cui procedure di configurazione coincidono con quelle di Windows 10.

4.2.1. Impostazioni

Per sfruttare il sottosistema WSL è necessario, innanzitutto, procedere con l'installazione di una distribuzione Linux e con la configurazione del dispositivo di comunicazione USB-UART.

In primo luogo, sarà necessario impostare, dalle funzionalità di Windows, la voce "Piattaforma Macchina Virtuale" e la voce "Piattaforma Windows Hypervisor".
Successivamente, tramite Windows PowerShell, sarà necessario eseguire il comando '*wsl --install <nome_distribuzione>'* per installare la distribuzione Linux desiderata (nel nostro Debian).

Fatto ciò, avremo il nostro sottosistema Linux e non ci resterà che consentire a quest'ultimo di utilizzare il dispositivo USB-UART per la comunicazione con la macchina target.
Per fare questo, ci basterà installare il servizio **usbipd**, sul nostro terminale, tramite il comando 'winget install --interactive --exact dorssel.usbipd-win'.
Sarà dunque necessario installare il servizio **usbip** sul sottosistema Linux tramite il comando da terminale '*sudo apt-get intall usbip*'.

Installati i suddetti servizi, dalla Windows PowerShell, sarà necessario eseguire l'attach del dispositivo USB-UART.
Per fare ciò, dopo aver collegato il dispositivo USB-UART dal nostro terminale alla macchina target, basterà digitare il comando '*usbipd wsl list*', verificare il valore del **busid** del dispositivo d'interesse e digitare il comando '*usbipd wsl attach --busid <id_bus>*'.
A questo punto, digitando nuovamente il comando '*usbipd wsl list´* potremo confermare che il dispositivo è stato collegato correttamente al sottosistema Linux.

Terminate le operazioni di impostazione, potremo utilizzare il nostro sottosistema Linux per comunicare con la macchina target.
Per fare ciò utilizzeremo il terminale **picocom**, basato su **minicom**, che installeremo digitando i due comandi seguenti…

'*sudo apt install minicom'*
'sudo apt install picocom'

A questo punto non ci resterà che cominciare a utilizzare il suddetto terminale per interfacciarci con la nostra macchina target.

4.2.2. Terminale Picocom

Per avviare il terminale picocom non dovremo fare altro che eseguire il comando '*sudo picocom*' seguito da una serie di opzioni che consentiranno di impostare alcuni parametri…
Le opzioni che utilizzeremo sono le seguenti:

- -b 115200
    - Parametro che imposta il Baudrate a 115200

- /dev/ttyUSB00
    - Parametro che indica il dispositivo USB con cui avviare la comunicazione (Di default questo risulta essere sempre 'ttyUSB00'

- --send "ascii-xfr -sv -l100 -c10"
    - Stabilisce i parametri di trasmissione che, in ordine, saranno:
        - *ascii-xfr*, che consente la trasmissione di caratteri ASCII
        - *-sv*, che rende dettagliata la modalità di invio e silenzia la trasmissione disabilitando la visualizzazione dei caratteri trasmessi
        - *-l100*, che impone un ritardo di 100ms dopo l'invio di ogni riga
        - *-c10*, che impone un ritardo di 10ms dopo ogni carattere inviato

- --imap delbs
    - Mappa l'operazione di cancellazione nel tasto 'backspace' consentendo di cancellare caratteri attraverso la sua pressione

Avvieremo, quindi, il terminale picocom attraverso il comando
'*sudo picocom -b 115200 /dev/ttyUSB00 --send "ascii-xfr -sv -l100 -c10" --imap delbs*'.

Fatto ciò, basterà alimentare il nostro Raspberry Pi per poter interagire con l'ambiente Forth avviato da quest'ultimo.
Inoltre, è possibile premere la combinazione di tasti *CTRL + A + H* per visualizzare un messaggio contenente tutte le combinazioni di tasti utilizzabili all'interno del terminale.
Le più importanti saranno quella per importare dei file '*CTRL + A + S*' e quella per chiudere il terminale '*CTRL + A + X*'.

5. Software

Il software, scritto interamente in linguaggio Forth, viene suddiviso in differenti file al fine di rendere il progetto modulare, facilitando eventuali modifiche e astraendo le operazioni di sistema dalle operazioni di basso livello.

Di seguito illustreremo il contenuto di ogni file, descrivendo tutte le WORD Forth definite.
L'ordine con cui saranno descritti i file corrisponde all'ordine con cui essi dovranno essere caricati nella macchina target per garantire il corretto funzionamento del sistema.

5.1. setup.f

File contenente tutte le WORD e le costanti relative alle funzionalità di base, alla scrittura e alla lettura di registri.
Il codice contenuto in questo file ha lo scopo di semplificare la scrittura del codice successivo al fine di mantenere un certo livello di astrazione tra la reale configurazione dei dispositivi e il loro utilizzo.

5.1.1. Costanti

- GPFSEL0 = 3F20000
- GPFSEL1 = 3F20004
- GPFSEL2 = 3F20008
    - Costanti che memorizzano l'indirizzo dei registri utili a definire le funzioni dei pin GPIO
- GPSET0 = 3F20001C
- GPCLR0 = 3F200028
    - Costanti che memorizzano l'indirizzo dei registri utili a impostare determinati pin GPIO, compresi nell'intervallo [0-31], sugli stati High e Low
- GPLEV0 = 3F200034
    - Costante che memorizza l'indirizzo del registro da consultare per ricevere il valore del livello di tutti i GPIO nell'intervallo [0-31]
- SYSCLO = 3F003004
- SYSCHI = 3F003008
    - Costanti che memorizzano l'indirizzo dei registri relativi al System Timer Counter. In particolare, SYSCLO memorizza i 32 LSB del timer, SYSCHI memorizza i 32 MSB

5.1.2. KEYROWOUT (--)

WORD per impostare in Output Mode tutti i GPIO utilizzati per le righe della Button Matrix

5.1.3. KEYROWIN (--)

WORD per impostare in Input Mode tutti i GPIO utilizzati per le righe della Button Matrix

5.1.4. KEYCOLOUT (--)

WORD per impostare in Output Mode tutti i GPIO utilizzati per le colonne della Button Matrix

5.1.5. KEYCOLIN (--)

WORD per impostare in Input Mode tutti i GPIO utilizzati per le colonne della Button Matrix

5.1.6. LEDR (-- 00040000 gpset0 gpclr0)

5.1.7. LEDV (-- 00800000 gpset0 gpclr0)

5.1.8. LEDB (-- 01000000 gpset0 gpclr0)

5.1.9. LEDG (-- 00400000 gpset0 gpclr0)

5.1.10. LEDACT (-- 20000000 gpset0 gpclr0)

5.1.11. BUZZ (-- 00000040 gpset0 gpclr0)

WORD che inseriscono nello stack il valore '1' nella posizione dell'n-esimo bit, che rappresenta l'n-esimo GPIO, seguito dagli indirizzi dei registri GPSET0 e GPCLR0

5.1.12. MATRIXROWS (-- 00311000 gpset0 gpclr0)

5.1.13. MATRIXCOLS (-- 06082000 gpset0 gpclr0)

WORD che inseriscono nello stack il valore '1', in tutte le posizioni degli i-esimi bit che rappresentano gli i-esimi GPIO, seguito dagli indirizzi dei registri GPSET0 e GPCLR0

5.1.14. ON (ngpio gpset0 gpclr0 --)

WORD che, data la rappresentazione dei GPIO a cui cambiare stato e i registri GPSET0 e GPCLR0, imposta lo stato High tramite il registro GPCLR0

5.1.15. OFF (ngpio gpset0 gpclr0 --)

WORD che, data la rappresentazione dei GPIO a cui cambiare stato e i registri GPSET0 e GPCLR0, imposta lo stato Low tramite il registro GPCLR0

5.1.16. ALLOFF (--)

WORD che facilita l'impostazione dello stato di Low per i dispositivi di output utilizzati nel sistema

5.1.17. ?PUSHED (-- f)

WORD che, consultando il registro GPLEV0, verifica lo stato del GPIO associato al Push Button bianco, restituendo un flag non-zero nel caso in cui questo sia High e un flag zero altrimenti

5.1.18. ?DETECTED (-- f)

WORD che, consultando il registro GPLEV0, verifica se lo stato del GPIO associato al Sensore di Movimento PIR, restituendo un flag non-zero nel caso in cui questo sia High e un flag zero altrimenti

5.1.19. C1 … C4 (-- n)

Insieme di WORDS che, consultando il registro GPLEV0, qualora rilevassero la pressione di un tasto della Push Button Matrix restituirebbero il valore della colonna a cui esso appartiene.
Viceversa, non rilevando la suddetta pressione, restituirebbero il valore '0'.

5.1.20. R1 … R4 (-- n)

Insieme di WORDS che, consultando il registro GPLEV0, qualora rilevassero la pressione di un tasto della Push Button Matrix restituirebbero il valore della riga a cui esso appartiene.
Viceversa, non rilevando la suddetta pressione, restituirebbero il valore '0'.

5.1.21. TIME (-- time)

WORD che effettua il fetch del valore contenuto nel registro SYSCLO e lo inserisce nello Stack

5.1.22. DELAY (usec --)

WORD che, dato un valore di tempo espresso in microsecondi, genera un busy loop della durata del valore prelevato dallo stack.

5.1.23. MSEC (usec -- msec)

WORD utile per convertire il valore del tempo da microsecondi a millisecondi

5.1.24. SEC (usec -- sec)

WORD utile per convertire il valore del tempo da microsecondi a secondi

5.2. components.f

File contenente le WORD utili a definire il controllo dei dispositivi hardware al fine di fornire le funzionalità di base del sistema.
Questo file utilizzerà le WORD definite nel file *setup.f* al fine di ottenere un maggiore livello di astrazione.

5.2.1. ALARMBLINK (--)

WORD che esegue un ciclo completo di allarme, attivando e disattivando ripetutamente il Buzzer e alternando i LED Rosso e Blu

5.2.2. IDLESETUP (--)

WORD che facilita l'operazione preliminare all'ingresso del Sistema in Idle State

5.2.3. ALERTSETUP (--)

WORD che facilita l'operazione preliminare all'ingresso del Sistema in Alert State

5.2.4. ALARMSETUP (--)

WORD che facilita l'operazione preliminare all'ingresso del Sistema in Alarm State

5.2.5. ?3SECPUSH (-- f)

WORD che, avvertita la pressione del Push Button bianco, verifica se questo risulta essere ancora premuto dopo un intervallo temporale pari a 3 secondi

5.2.6. WAITPRESSCOL (-- col)

WORD che genera un loop indefinito che termina quando uno dei pulsanti della Push Button Matrix viene premuto e che restituisce il valore della colonna a cui il suddetto pulsante appartiene

5.2.7. WAITPRESSROW (-- row)

WORD che genera un loop indefinito che termina quando uno dei pulsanti della Push Button Matrix viene premuto e che restituisce il valore della riga a cui il suddetto pulsante appartiene

5.2.8. READCOL (-- col)

WORD che esegue la procedura corretta per consentire l'utilizzo della WORD WAITPRESSCOL, inserendo nello Stack il valore di colonna a cui appartiene il pulsante premuto

5.2.9. READROW (-- row)

WORD che esegue la procedura corretta per consentire l'utilizzo della WORD WAITPRESSROW, inserendo nello Stack il valore di riga a cui appartiene il pulsante premuto

5.2.10. B1 … B16 (-- n)

Insieme di WORD che restituiscono valori da associare ai pulsanti della Push Button Matrix

5.2.11. ?B1 … ?B16 (col row -- b)

Insieme di WORD che, dati i valori di colonna e riga di un pulsante premuto, restituiscono il valore del pulsante che incrocia la colonna e la riga.
Qualora, per una WORD del suddetto insieme, i valori di colonna e di riga non dovessero rappresentare il pulsante in questione, la WORD non restituirà alcun valore

5.2.12. BTNVALUE (-- b)

WORD che esegue tutte le WORD dell'insieme precedentemente descritto restituendo il valore corrispondente al pulsante della Push Button Matrix premuto

5.2.13. READBUTTON (-- b)

WORD che esegue le WORD READCOL, READROW e BTNVALUE per portare il Sistema in attesa della pressione di un pulsante della Push Button Matrix e per restituirne il valore associato

5.3. login.f

File contenente le WORD e le Variabili utili per la gestione della password di Sistema e per lo sblocco di quest'ultimo dal Security State.
Per l'utilizzo di queste funzionalità si sfrutteranno principalmente tutte le WORD relative all'utilizzo della Push Button Matrix definite nel file *components.f.*

5.3.1. Variabili

- PSW → Array per memorizzare il valore della Password di Sistema impostata durante lo Start State
- TMP → Array per memorizzare il valore della Password inserita dall'utente

5.3.2. CELLS (n -- v)

Ridefinizione della WORD CELLS per garantire la corretta dimensione di allocazione per le celle degli Array

5.3.3. CELLS+ (n -- v)

WORD che, dato un numero che indica una quantità di celle, restituisce l'offset totale per localizzare l'n-esima cella

5.3.4. SETSOUND (--)

WORD utilizzata per riprodurre un breve effetto sonoro da utilizzare per confermare la corretta impostazione della Password in fase di Start State

5.3.5. SETPSW (--)

WORD che consente al Sistema di rimanere in attesa di digitazione della password di tre cifre da parte dell'utente durante lo Start State.
Le cifre digitate saranno memorizzate nell'array *PSW*.
In particolare, le letture dei pulsanti premuti sono separate da intervalli temporali pari a 500ms

5.3.6. WRITEPSW (--)

WORD che consente al Sistema di rimanere in attesa di digitazione della password di tre cifre da parte dell'utente durante il Security State.
Le cifre digitate saranno memorizzate nell'array *TMP*.
In particolare, le letture dei pulsanti premuti sono separate da intervalli temporali pari a 500ms

5.3.7. ?CHECKPSW (-- f)

WORD che confronta, cella per cella, i valori contenuti negli array PSW e TMP verificando la corrispondenza tra la Password di Sistema registrata nell'array PSW e la Password digitata dall'utente in Security State registrata nell'array TMP.
Il confronto tra i due array restituirà un flag non-zero in caso di corrispondenza, un flag zero altrimenti

5.4. main.f

Questo file contiene il corpo del programma e la definizione delle WORD che implementano il funzionamento del Sistema nei suoi differenti stati.
Il file, infine, termina con l'esecuzione di una WORD che consentirà l'effettivo avvio del Sistema.

5.4.1. IDLELOOP (--)

WORD che definisce le operazioni possibili durante l'Idle State

5.4.2. SENSORLOOP (--)

WORD che definisce le operazioni possibili durante l'Alert State

5.4.3. ALARMLOOP (--)

WORD che definisce le operazioni possibili durante l'Alarm State

5.4.4. SECURITYLOOP (--)

WORD che definisce le operazioni possibili durante il Security Loop

5.4.5. START (--)

WORD che avvia il Sistema in Start State e, dopo l'impostazione della Password di Sistema, entra in un loop infinito che avvierà il funzionamento del Sistema consentendo il passaggio di questo tra i suoi differenti stati

6. Il Sistema

Una volta effettuati tutti i collegamenti necessari, dopo aver impostato l'ambiente di lavoro, sarà necessario caricare i suddetti file sulla nostra macchina target tramite il terminale picocom.

In particolare, i file andranno caricati nell'ordine con cui essi sono stati descritti, così da garantire la definizione delle WORD di base prima che se ne faccia uso nelle WORD di più alto livello.
L'ordine di caricamento, quindi, sarà il seguente:

- *setup.f*
- *components.f*
- *login.f*
- *main.f*

Una volta caricato l'ultimo file (*main.f*), il Sistema entrerà in funzione e si troverà in Start State.

6.1. Rappresentazione degli Stati

Questa sezione ha lo scopo di illustrare il comportamento di tutte le componenti del Sistema in base allo stato in cui quest'ultimo si trova.

6.1.1. Start State

In questo stato, tutte le componenti risultano spente eccetto che i LED Rosso e Blu che, combinando i due colori, formeranno una luce Viola volta a indicare l'attesa di inserimento della Password di Sistema tramite tastierino numerico.

6.1.2. Idle State

Stato in cui il Sistema è in attesa di un'azione da parte dell'utente e mantiene il solo LED Giallo attivo.
In particolare, tramite la pressione prolungata del Push Button Bianco, l'utente potrà entrare nel loop di selezione che consentirà di scegliere se procedere con l'Alert State o se passare nuovamente allo Start State per reimpostare la password di Sistema.

In questa fase il solo LED Rosso rimarrà acceso fino al selezionamento di una delle due opzioni tramite i pulsanti 16 e 13 del tastierino numerico.
Nel caso in cui l'utente decidesse di reimpostare la password, il LED Rosso si spegnerebbe e, dopo un breve effetto ottico/sonoro prodotto dal Buzzer e dal LED Giallo, il LED Verde si accenderebbe fino a quando l'utente non avrà verificato la propria identità, inserendo la password precedentemente registrata e consentendo il passaggio allo Start State.

La pressione di un qualunque tasto differente dai sopracitati comporterà il passaggio del Sistema verso l'Idle State.

6.1.3. Alert State

In questo stato, il LED Verde rimarrà acceso fin quando non verrà rilevato un movimento o fin quando non verrà premuto il pulsante Bianco per il passaggio all'Idle State.

6.1.4. Alarm State

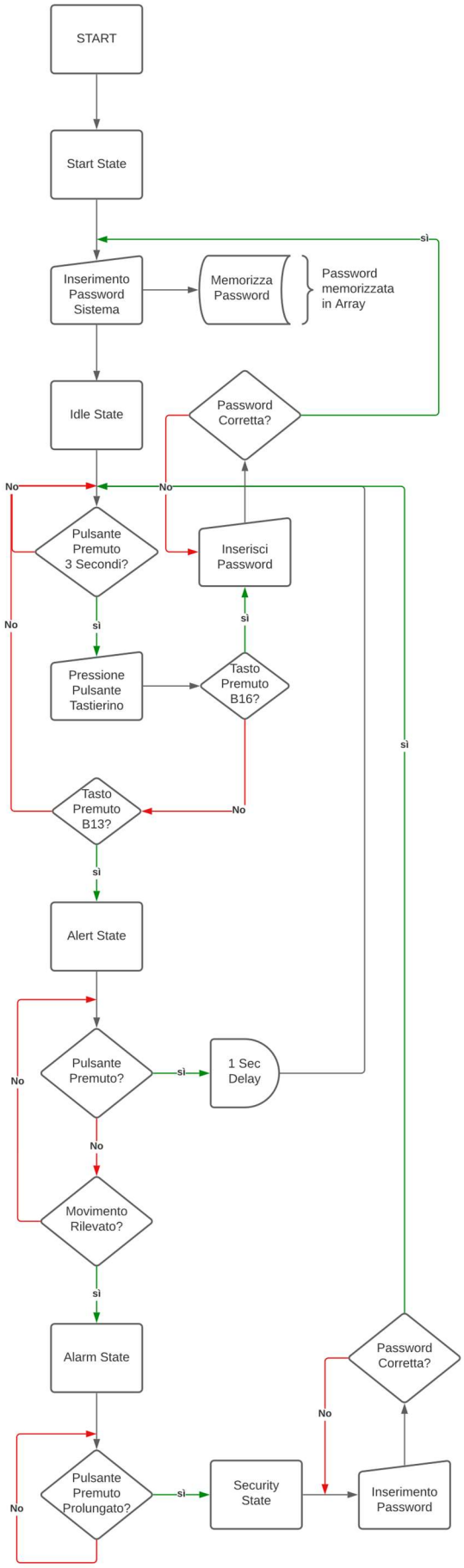
Stato in cui il Sistema passa in seguito alla rilevazione di un movimento durante l'Alert State.
In questo stato vi è un effetto ottico/sonoro intermittente che vedrà l'alternarsi dei colori prodotti dai LED Rosso e Blu.
Il solo modo per uscire da questo stato prevede la pressione prolungata del pulsante Bianco fino al passaggio verso il Security State.

6.1.5. Security State

Stato in cui il Sistema passa in seguito alla pressione prolungata del pulsante Bianco durante l'Alarm State.
In particolare, questo stato manterrà acceso il LED Rosso e imposterà il Buzzer per produrre un suono continuo di durata indefinita.

L'unico modo per terminare questo stato, tornare all'Idle State, è quello di inserire correttamente la password di Sistema tramite il tastierino numerico.

6.2. Diagramma di Flusso



Questo diagramma illustra sinteticamente le operazioni che il Sistema svolge prima di procedere con un cambiamento di stato, senza però descrivere nel dettaglio il comportamento specifico dei sensori e degli attuatori.

Si tratta di una rappresentazione grafica della sequenza di operazioni che vengono svolte dalla WORD START contenuta nel file *main.f* che servirà ad avviare il sistema.

È ovviamente possibile effettuare delle modifiche al codice al fine di cambiarne il flusso di esecuzione, modificando così l'effettivo comportamento del Sistema.

7. Conclusioni

Il Sistema proposto è un prototipo di allarme che, con le dovute accortezze, potrebbe essere implementato e utilizzato in scenari reali.
La disposizione dei file che compongono il software, inoltre, consente l'implementazione del Sistema su macchine target differenti con la sola necessità di modifica delle WORD di basso livello presenti nel file *setup.f*, lasciando inalterata la logica di alto livello definita negli altri files.

Inoltre, è possibile migliorare il Sistema aggiungendo ulteriori dispositivi o sostituendo quelli esistenti.
In tal caso, ovviamente, sarebbe necessario modificare parti di codice e implementare nuove WORD per l'esecuzione delle nuove funzionalità.

Domenico Giosuè