Università degli studi di Palermo 2022-2023

DEFUSE THE BOMB

GIOCO DI SIMULAZIONE

Vito Di Grigoli | Sistemi Embedded | Prof. Daniele Peri

1. Introduzione

Lo scopo di questo progetto è quello di creare un gioco che simuli il disinnesco di una bomba da parte di un artificiere.

Una volta avviato il programma, il Raspberry inizializzerà tutti i vari componenti e rimarrà in attesa che lo switch sia in posizione corretta per avviare il gioco.

Una volta avviato, verrà visualizzata una sequenza di accensione e spegnimento dei led. L'artificiere dovrà replicare correttamente la sequenza per aprire la bomba e disinnescarla.

In realtà, la bomba è dotata di un meccanismo di protezione al suo interno che si attiverà all'apertura della bomba, la quale avvierà un timer che segnerà un countdown di 9 secondi prima dell'esplosione.

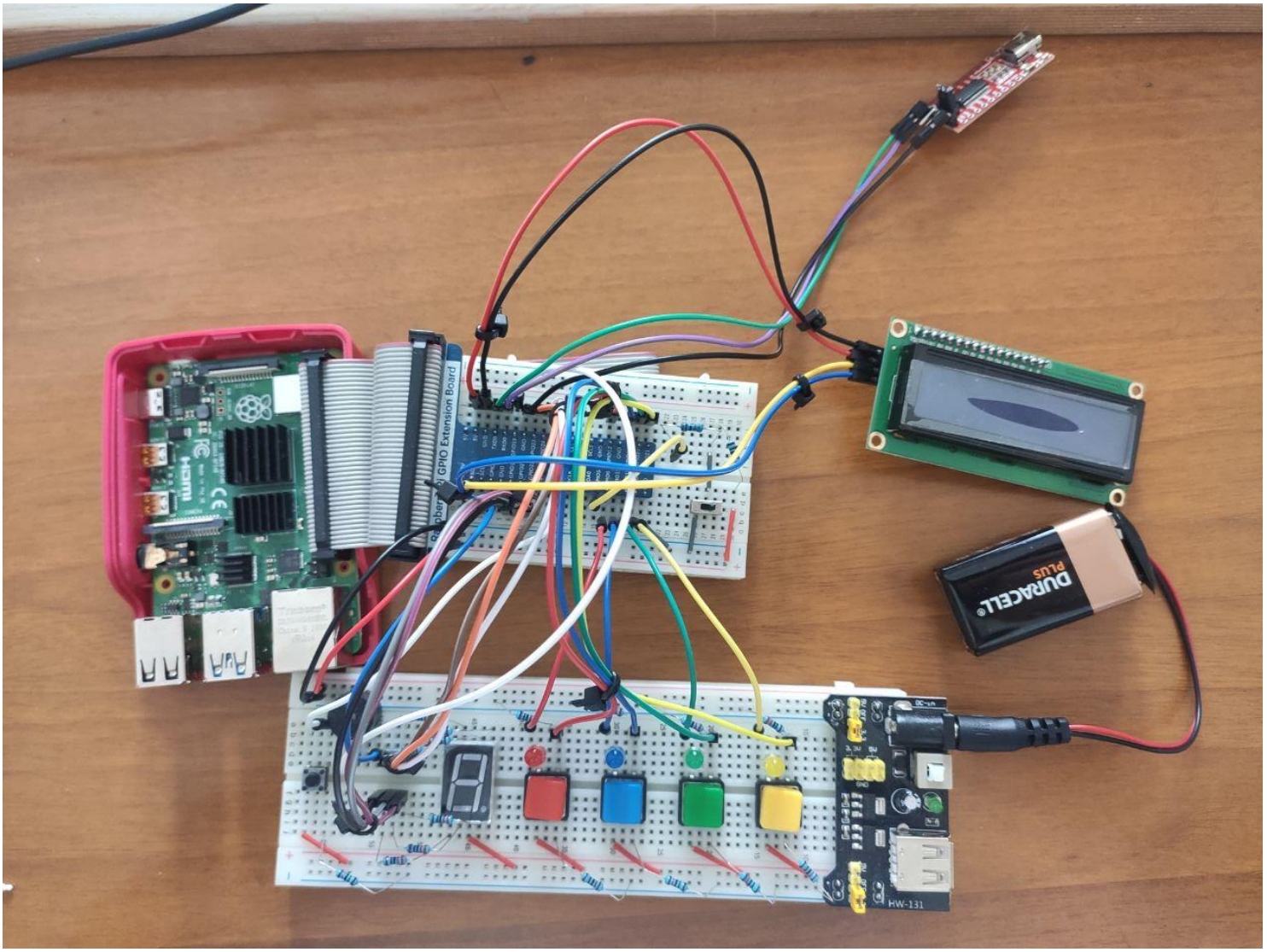
Il nostro artificiere dovrà cercare il bottone che si trova al suo interno che permetterà di disinnescare la bomba e salvare delle innocenti vite, il tutto correlato da messaggi sul display e da segnali acustici per rendere ancora più immersiva l'esperienza utente.

L'intero progetto è stato scritto nel linguaggio FORTH ed è eseguito in un ambiente installato sulla memoria SD del Raspberry PI 4B.

2. Componenti hardware utilizzati

- N.1 Raspberry PI 4B
- N.2 Breadboard
- N.1 GPIO Extension Board
- N.1 40 pin GPIO Cable
- N.1 LCD 16x02
- N.1 Modulo I2C con chip PCF8574AT
- N.1 Display a 7 segmenti
- N.1 Buzzer attivo
- N.4 LED (ROSSO, BLUE, VERDE, GIALLO)
- N.1 Pulsante piccolo
- N.4 Pulsanti grandi (ROSSO, BLUE, VERDE, GIALLO)
- N.1 Switch a scorrimento
- N.12 Resistenze da 220Ω
- N.5 Resistenze da 10KΩ
- N.1 Resistenze da 1KΩ
- N.1 FT232RL FTDI USB (Uart Serial)
- N.1 Modulo di alimentazione per la Breadboard
- N.1 Batteria da 9V con cavo di collegamento
- Jumper e ponticelli per i collegamenti

3. Circuito



4. Descrizione driver

4.1. NOTA IMPORTANTE

L'ordine con cui sono elencati i file in questo paragrafo corrisponde all'ordine in cui devono essere caricati i file sul Raspberry Pi. Tutti i valori sono in esadecimale.

4.2. JONESFORTH.F

Questo file appartiene al repository originale da cui è stato ricavato il kernel utilizzato per il Raspberry Pi e che è possibile su GitHub in questo repository https://github.com/organix/pijFORTHos.

4.3. HAL.F

File contenente le configurazioni dell'hardware collegato al Raspberry. Da questo file è possibile cambiare velocemente l'assegnazione di un dispositivo a un determinato GPIO senza bisogno di andare a riconfigurare il tutto.

Questo rende il codice molto flessibile ai cambiamenti di dispositivi tra i vari pin.

Inoltre, dato che è abbastanza comune mettere in relazione la pressione di un tasto con l'accensione di un led, è stato ideato un meccanismo che permette di associare questi dispositivi utilizzando il colore in comune.

4.3.1.Costanti

- **RPI4:** indirizzo base del Raspberry Pi 4

4.3.2.Words

Oltre alla configurazione dei pin con l'hardware, sono presenti anche alcune funzioni di utility.

- **BIT_SET**: setta il bit i-esimo a 1 di una posizione specifica.
- **BIT_CLEAR**: setta il bit i-esimo a 0 di una posizione specifica.
- **3BIT_SET**: setta i 3 bit i-esimi a 1 di una posizione specifica.
- **3BIT_CLEAR**: setta i 3 bit i-esimi a 0 di una posizione specifica.
- **4BIT_SET**: setta i 4 bit i-esimi a 1 di una posizione specifica.
- **4BIT_CLEAR**: setta i 4 bit i-esimi a 0 di una posizione specifica.
- **16BIT_SET**: setta i 16 bit i-esimi a 1 di una posizione specifica.
- **16BIT_CLEAR**: setta i 16 bit i-esimi a 0 di una posizione specifica.
- **STACK_CLEAR**: azzera il contenuto dello stack.

4.3.3. Esempi di utilizzo

- **0 3 BIT_SET**: Prende il valore (0 nel nostro caso) e setta a 1 il bit numero 3, senza andare ad intaccare i valori dei bit delle altre posizioni e lasciando sullo stack il nuovo valore (8 nel nostro caso).
- **:BUZZER 5**: definisce il collegamento del buzzer nel pin.
- **:RED 6 C**: definisce il collegamento del led rosso nel pin 6 e del bottone rosso nel pin 12.

- **:POWER 0 7**: definisce il collegamento del bottone power nel pin 7, senza un led di stato ad esso associato. Qualora poi si volesse aggiungere in futuro, basterà semplicemente sostituire il valore 0 con il pin annesso.

4.4. TIMER.F

File contenente le word per poter utilizzare le funzionalità del timer di sistema.

N.B. Il timer ha 4 comparatori C0, C1, C2 e C3 ma le applicazioni possono utilizzarne solo 2, C1 e C3, poiché C0 e C2 vengono utilizzati dalla GPU.

4.4.1.Costanti

- **TIMER:** Registro base del timer.
- **TIMER_CONTROL_STATUS:** Registro di controllo / stato del System Timer.
- **TIMER_COUNTER_LOW:** Registro contenente i 32 bit inferiori del System Timer Counter.
- **TIMER_COUNTER_HIGH**: Registro contenente i 32 bit superiori del System Timer Counter.
- **TIMER_COMPARE_0:** Indirizzo del registro del System Timer Compare 0.
- **TIMER_COMPARE_1:** Indirizzo del registro del System Timer Compare 1.
- **TIMER_COMPARE_2**: Indirizzo del registro del System Timer Compare 2.
- **TIMER_COMPARE_3:** Indirizzo del registro del System Timer Compare 3.

4.4.2.Words

- **NOW**: legge il valore attuale dei 32 bit inferiori del System Timer Counter corrispondenti al valore in microsecondi dall'avvio del sistema.
- **DELAY**: delay di un tempo pari ad n microsecondi utilizzando un busy wait.
- **USEC**: unità di misura per i microsecondi.
- **MSEC**: unità di misura per i millisecondi.
- **SEC**: unità di misura per i secondi.

4.4.3. Esempi di utilizzo

- **2 SEC DELAY**: Aggiunge un delay di 2 secondi

4.5. GPIO.F

File contenente le word per poter utilizzare le funzionalità del GPIO.

4.5.1.Costanti

- **GPIO:** indirizzo base per i GPIO.
- **GPFSEL0:** registro di selezione della funzione dei pin 0-9.
- **GPFSEL1:** registro di selezione della funzione dei pin 10-19.
- **GPFSEL2:** registro di selezione della funzione dei pin 20-29.
- **GPFSEL3:** registro di selezione della funzione dei pin 30-39.
- **GPFSEL4:** registro di selezione della funzione dei pin 40-49.
- **GPFSEL5:** registro di selezione della funzione dei pin 50-57.
- **GPSET0**: registro per settare l'output a 1 dei pin 0-31.
- **GPSET1**: registro per settare l'output a 1 dei pin 32-50.
- **GPCLR0**: registro per settare l'output a 0 dei pin 0-31.
- **GPCLR1**: registro per settare l'output a 0 dei pin 32-50.
- **GPLEV0**: registro che ritorna il valore dei pin 0-31.
- **GPLEV1**: registro che ritorna il valore dei pin 32-50.
- **GPEDS0**: registro che memorizza gli eventi dei pin 0-31.
- **GPEDS1**: registro che memorizza gli eventi dei pin 32-50.
- **GPREN0**: registro di abilitazione del rilevamento del fronte di salita dei pin 0-31.
- **GPREN1**: registro di abilitazione del rilevamento del fronte di salita dei pin 32-50.
- **GPFEN0**: registro di abilitazione del rilevamento del fronte di discesa dei pin 0-31.
- **GPFEN1**: registro di abilitazione del rilevamento del fronte di discesa dei pin 32-50.
- **GPHEN0**: registro di abilitazione del rilevamento del livello alto dei pin 0-31.
- **GPHEN1**: registro di abilitazione del rilevamento del livello alto dei pin 32-50.
- **GPLEN0**: registro di abilitazione del rilevamento del livello basso dei pin 0-31.
- **GPLEN1**: registro di abilitazione del rilevamento del livello basso dei pin 32-50.
- **GPAREN0**: registro di abilitazione del rilevamento del fronte di salita asincrono dei pin 0-31.
- **GPAREN1**: registro di abilitazione del rilevamento del fronte di salita asincrono dei pin 32-50.
- **GPAFEN0**: registro di abilitazione del rilevamento del fronte di discesa asincrono dei pin 0-31.
- **GPAFEN1**: registro di abilitazione del rilevamento del fronte di discesa asincrono dei pin 32-50.

4.5.2.Words

- **?GPFSEL**: restituisce il registro di selezione corretto in base al numero del pin.
- **?GPSETCLR**: restituisce i due registri di settaggio dell'output corretti in base al numero del pin.
- **?GPEDS**: restituisce il registro di memorizzazione degli eventi corretto in base al numero del pin.

- **?GPLEV**: restituisce registro del livello dei pin corretto in base al numero del pin.
- **?GPAFEN**: restituisce il registro di abilitazione del rilevamento del fronte di discesa asincrono corretto in base al numero del pin.
- **GPIO_MODE:** permette di selezionare la funzionalità del pin.
- **INPUT**: setta il pin in modalità input.
- **OUTPUT**: setta il pin in modalità output.
- **ALT_0**: setta il pin con la sua funzione alternativa 0.
- **ALT_1**: setta il pin con la sua funzione alternativa 1.
- **ALT_2**: setta il pin con la sua funzione alternativa 2.
- **ALT_3**: setta il pin con la sua funzione alternativa 3.
- **ALT_4**: setta il pin con la sua funzione alternativa 4.
- **ALT_5**: setta il pin con la sua funzione alternativa 5.
- **LED**: seleziona il pin relativo al led.
- **BUTTON**: seleziona il pin relativo al bottone.
- **ON**: imposta il livello del pin su alto.
- **OFF**: imposta il livello del pin su basso.
- **BLINK**: imposta il livello del pin su alto per 500ms.
- **TIC**: imposta il livello del pin su alto per 100ms.
- **PRESSED?**: controlla se il bottone è stato premuto.
- **LEVEL?**: controlla il livello di output del pin.
- **RESET!**: resetta il rilevamento della pressione dello specifico bottone.
- **RESET_ALL!:** resetta il rilevamento della pressione su tutti i bottoni.
- **ACTIVATE:** attiva il rilevamento di pressione per uno specifico bottone.
- **DEACTIVATE:** disattiva il rilevamento di pressione per uno specifico bottone.

4.5.3. Esempi di utilizzo

- **RED LED OUTPUT:** setta il pin a cui è collegato il led rosso in output.
- **RED BUTTON INPUT:** setta il pin a cui è collegato il bottone rosso in input.
- **RED LED ON:** imposta il valore del pin a cui è collegato il led rosso su alto.
- **RED BUTTON ACTIVATE:** attiva il rilevamento della pressione del pin a cui è collegato il bottone rosso.
- **RED BUTTON PRESSED?:** controlla se il bottone rosso è stato premuto.
- **RED BUTTON RESET!**: resetta lo stato di pressione del bottone rosso.

4.6. I2C.F

File contenente le word per poter utilizzare il bus I2C.

4.6.1.Costanti

- **BSC1:** indirizzo base per il controller BSC 1.

- **I2C_CONTROL:** registro utilizzato per abilitare gli interrupt, cancellare il FIFO, definire un'operazione di lettura o scrittura e avviare un trasferimento.
- **I2C_STATUS:** utilizzato per registrare lo stato delle attività, gli errori e le richieste di interruzione.
- **I2C_DATA_LENGTH:** registro che definisce il numero di byte di dati da trasmettere o ricevere nel trasferimento I2C.
- **I2C_SLAVE_ADDRESS:** registro che specifica l'indirizzo dello slave e il tipo di ciclo.
- **I2C_DATA_FIFO:** registro utilizzato per accedere alla FIFO.
- **I2C_CLOCK_DIVIDER:** viene utilizzato per definire la velocità di clock della periferica BSC.
- **I2C_DATA_DELAY**: registro che fornisce un controllo preciso sul punto di campionamento/invio dei dati.
- **I2C_CLOCK_STRETCH_TIMEOUT**: registro che fornisce un timeout su quanto tempo il master attende che lo slave allunghi il clock prima di decidere che lo slave sia sospeso.

4.6.2.Words

- **RESET_FIFO**: resetta la FIFO.
- **RESET_STATUS**: resetta lo stato di attività, errori e interruzioni.
- **SET_SLAVE**: setta l'indirizzo dello slave.
- **I2C_SETUP**: inizializza la trasmissione I2C.
- **I2C_STORE**: copia i dati nella FIFO.
- **I2C_SEND:** avvia un nuovo trasferimento.
- **I2C_DELAY**: aggiunge un delay alla trasmissione.
- **>I2C**: combina le word precedenti per scrivere correttamente sul bus I2C.
- **SEND**: combina due scritture I2C. Utile per inviare una coppia di comandi allo slave.

4.6.3. Esempi di utilizzo

- **08 >I2C:** invia 00001000 sul bus I2c
- **08 05 SEND:** invia 00001000 00000101 sul bus I2C.

4.7. LCD.F

File contenente le word per poter utilizzare LCD 1608 con modulo I2C e chip PCF8574AT.

4.7.1.Words

- **SEND_4BIT_MORE**: invia due volte i 4 bit più significativi di un comando concatenandoli con i 4 bit di configurazione dell'LCD (B, E, RW, RS).
    - Prima trasmissione: B = 1, E = 1, RW = 0, RS = 1.
    - Seconda trasmissione: B = 1, E = 0, RW = 0, RS = 1.
- **SEND_4BIT_LESS**: invia due volte i 4 bit meno significativi di un comando concatenandoli con i 4 bit di configurazione dell'LCD (B, E, RW, RS).
    - Prima trasmissione: B = 1, E = 1, RW = 0, RS = 1.
    - Seconda trasmissione: B = 1, E = 0, RW = 0, RS = 1.
- **>LCD**: invia un comando all'LCD.
- **SMILE**: sequenza di caratteri per disegnare uno smile.
- **HEART_EMPY**: sequenza di caratteri per disegnare un cuore chiaro.
- **HEART**: sequenza di caratteri per disegnare un cuore scuro.
- **BOMB**: sequenza di caratteri per disegnare una bomba.
- **SKULL**: sequenza di caratteri per disegnare un teschio.
- **LOCK**: sequenza di caratteri per disegnare un lucchetto.
- **BELL**: sequenza di caratteri per disegnare una campana.
- **CHECK**: sequenza di caratteri per disegnare una spunta.
- **4BIT-CONFIG**: inizializza LCD con una configurazione a 4 bit.
- **LCD**: comando base per le operazioni sull'LCD.
- **CLEAR**: cancella lo schermo.
- **SWITCH_ON**: accende lo schermo.
- **SWITCH_OFF**: spegne lo schermo disattivando anche la retroilluminazione.
- **SHIFT_LEFT**: abilita l'incremento del contatore verso sinistra.
- **SHIFT_RIGHT**: abilita l'incremento del contatore verso destro.
- **CURSOR**: comando base per le operazioni con il cursore.
- **UP**: sposta il cursore nella prima riga.
- **DOWN**: sposta il cursore nella seconda riga.
- **TURN_ON**: attiva la visualizzazione del cursore.
- **TURN_OFF**: disattiva la visualizzazione del cursore.
- **BLINKS**: attiva il lampeggio del cursore.
- **!BLINKS**: disattiva il lampeggio del cursore.
- **RETURN**: fa ritornare il cursore nella sua posizione iniziale.
- **MOVE**: comando base per le operazioni di movimento.
- **CURSOR_LEFT**: sposta il cursore verso sinistra.
- **CURSOR_RIGHT**: sposta il cursore verso destra.
- **DISPLAY_LEFT**: sposta la visualizzazione dello schermo verso sinistra.
- **DISPLAY_RIGHT**: sposta la visualizzazione dello schermo verso destra.
- **LCD_RESET**: resetta LCD allo stato iniziale.
- **DEL**: cancella un carattere verso sinistra.
- **SAVE-0**: setta la posizione 0 nel Character Generator Ram Address.
- **SAVE-1**: setta la posizione 1 nel Character Generator Ram Address.
- **SAVE-2**: setta la posizione 2 nel Character Generator Ram Address.
- **SAVE-3**: setta la posizione 3 nel Character Generator Ram Address.
- **SAVE-4**: setta la posizione 4 nel Character Generator Ram Address.
- **SAVE-5**: setta la posizione 5 nel Character Generator Ram Address.

- **SAVE-6**: setta la posizione 6 nel Character Generator Ram Address.
- **SAVE-7**: setta la posizione 7 nel Character Generator Ram Address.
- **MYCHAR**: permette di salvare un carattere personalizzato nella posizione preferita.
- **'SIMBOLO**: permette di stampare a schermo il carattere digitato. Sono stati definiti tutti i simboli della tabella ascii.

4.7.2. Esempi di utilizzo

- **'H 'E 'L 'L 'O 'SPACE 'W 'O 'R 'L 'D '!:** stampa HELLO WORLD! sullo schermo
- **LCD CLEAR**: pulisce lo schermo.
- **0 MYCHAR BOMB**: salva il carattere personalizzato nella posizione 0.
- **0 >LCD**: stampa il carattere personalizzato salvato nella posizione 0.
- **CURSOR BLINKS**: attiva il lampeggio del cursore.

4.8. DISPLAY.F

File contenente le word per poter utilizzare il display a 7 segmenti.

4.8.1.Words

- **>OFF**: spegne tutti e 7 i segmenti.
- **>ON**: accende soltanto il segmento centrale e spegne gli i restanti.
- **0>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 0.
- **1>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 1.
- **2>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 2.
- **3>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 3.
- **4>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 4.
- **5>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 5.
- **6>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 6.
- **7>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 7.
- **8>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 8.
- **9>DISP**: combina l'accensione e spegnimento dei segmenti per formare il numero 9.
- **>DISP**: permettere di visualizzare un numero da 0 a 9 sul display.
- **COUNTDOWN**: avvia un countdown sul display.

4.8.2. Esempi di utilizzo

- **7 >DISP:** visualizza il numero 7 sul display
- **10 >DISP:** restituisce un messaggio di errore perché è possibile visualizzare i numeri da 0 a 9.
- **5 >COUNTDOWN**: avvia una sequenza di countdown da 5 secondi.

4.9. MAIN.F

File contenente le word per il programma principale.

4.9.1.Words

- **SETUP_CHAR**: salva tutti i caratteri personalizzati creati sull'LCD
- **INIT_GPIO**: inizializza l'hardware collegato ai GPIO.
- **INIT_LCD**: inizializza LCD, attivando il bus I2C e salvando i caratteri personalizzati.
- **INIT_DISPLAY**: inizializza i GPIO collegati al display a 7 segmenti.
- **MSG0-5**: contengono i messaggi da far visualizzare sull'LCD durante l'esecuzione del programma.
- **BOOT**: avvia la sequenza di inizializzazione dei vari dispositivi descritti sopra.
- **CHECK_PRESSED**: controlla quale bottone è stato premuto e lascia il valore nello stack.
- **CHECK_VALUE**: controlla se il valore lasciato nello stack corrisponde a quello corretto della sequenza.
- **STATE_0**: stato iniziale in cui viene fatta lampeggiare la sequenza che l'artificiere deve replicare.
- **STATE_1**: l'artificiere preme il tasto e viene controllato se il valore è corretto. Se errato, si passa allo stato 2. Se corretto, si ritorna allo stato 1 fino a quando non viene completata l'intera sequenza.
- **STATE_2**: si pulisce lo stack, suona il buzzer per segnalare l'errore e si ritorna allo stato 0.
- **STATE_3**: l'artificiere è riuscito a completare correttamente la sequenza, ma il dispositivo era dotato di un sistema di protezione che attiva il timer per la detonazione della bomba. Il nostro eroe avrà 9 secondi per trovare il bottone che sarà in grado di fermare il timer e salvare delle vite. In caso di successo, viene fermato il timer e visualizzato un messaggio di complimenti. In caso di insuccesso verrà simulata l'esplosione della bomba dal suono del nostro buzzer.
- **MAIN**: contiene gli stati finiti del nostro programma. Una volta avviato, sarà possibile iniziare o fermare in qualunque momento il gioco grazie allo switch a scorrimento presente sulla breadboard.