Realizzazione di un sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su arduino

Titolo del progetto:	Realizzazione di un sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su arduino
Alunno/a:	Bryan Beffa, Filippo Finke, Matteo Ghilardini
Classe:	I3AA/I3AC
Anno scolastico:	2018/2019
Docente responsabile:	Adriano Barchi

Sommario

1 Introduzione

1.1 Informazioni sul progetto

Allievi coinvolti nel progetto: Bryan Beffa, Filippo Finke, Matteo Ghilardini
Classe: Informatica 3AA/3AC presso la sede Scuola Arti e Mestieri Trevano
Docenti responsabili: Adriano Barchi
Data inizio: 13.02.2019
Data consegna: 22.05.2019

1.2 Abstract

The school needs a system that it improves and tests the reaction skills of a person. There is a previous project "Batak" that the students of the last year did, we had to complete it and add more programs and modality to improve it. They must be program for senior and junior, juniors do not use buttons 6, 7, 8, 9. Senior use all buttons of the system. The programs are different in duration, speed, reaction time. This system is used to train reaction times and improve them. The different times and point are showed in two seven segments displays and they are updated in real time. The system use also a LCD display to inform the user about the program that's currently running and some instructions. There is also a procedure created to check if the system is working fine without hardware problems. The system is built on top of two arduino, one that manage the programs and the LCD display and one that manage the representation of the points and the time. The two arduino are connected using SDA and SCL.

1.3 Scopo

Lo scopo di questo progetto è di realizzare, sulla base del progetto "Badak" realizzato dagli alunni di terza dell'anno scorso, un apparecchio che permetta di migliorare i tempi di reazione e di coordinamento ottico – motorio premendo i pulsanti che si illuminano in modo casuale. Come detto in precedenza si lavora sulla base del progetto "Badak" utilizzandone quindi il telaio e l'hardware generale già preparato l'anno precedente. Sono disponibili diversi programmi di allenamento che variano per durata, complessità, velocità di reazione. I programmi vengono gestiti da un sistema composto da Arduini.

Analisi

1.4 Analisi del dominio

Ci è stato richiesto da parte del committente, il docente Adriano Barchi, di realizzare un sistema che permetta agli utilizzatori di misurare/migliorare le loro performance negli ambiti di reazione ed ottico – motorio. Questo sistema deve essere sviluppato partendo dal risultato del progetto "Badak" realizzato dagli allievi del terzo anno dello scorso anno. Il sistema iniziale presenta alcuni limiti come i display utilizzati per il punteggio attuale e il tempo che non sono funzionanti.
Deve esserci la possibilità di riprodurre diverse modalità di allenamento che variano in difficoltà, durata e tempo di reazione.
Vi devono essere programmi per due tipi di utente, ossia senior (tutti i numeri sono utilizzati) oppure junior (pulsanti 6, 7, 8 ed 9 esclusi).
Mentre un programma viene svolto sui display posti in alto al telaio devono apparire i punti aggiornati e il timer.

1.5 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Cumulativo 60 secondi, senior
Priorità	1
Versione	1.0
Note	La velocità del gioco è definita dalla velocità con la quale il giocatore preme i pulsanti
Sotto requisiti	
001	Il programma dura 60 secondi
002	Vengono mostrati punteggio e tempo rimanente
003	Accensione casuale di tutti i pulsanti
004	Accensione sequenziale
005	Spegnimento quando premuti
006	Conteggio dei pulsanti premuti correttamente

ID: REQ-02	
Nome	Maratona (5 minuti) senior
Priorità	1
Versione	1.0
Note	La velocità del gioco è definita dalla velocità con la quale il giocatore preme i pulsanti
Sotto requisiti	
001	Il programma dura 300 secondi
002	Vengono mostrati punteggio e tempo rimanente
003	Accensione casuale di tutti i pulsanti
004	Accensione sequenziale
005	Spegnimento quando premuti
006	Conteggio dei pulsanti premuti correttamente

ID: REQ-03	
Nome	Corsa 50 pulsanti, senior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 100 secondi o 50 pulsanti
002	Vengono mostrati punteggio e tempo rimanente (al decimo di secondo)
003	Accensione casuale di tutti i pulsanti
004	Accensione sequenziale
005	Spegnimento quando premuti
006	Conteggio dei pulsanti premuti correttamente

ID: REQ-04	
Nome	Stretching angolare, 100 pulsanti, senior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 100 pulsanti
002	Vengono mostrati punteggio (numero di pulsanti rimanenti) e tempo trascorso
003	Accensione casuale dei pulsanti agli estremi del telaio per 1 secondo
004	Accensione sequenziale
005	Quando un pulsante viene premuto erroneamente la sequenza accelera (diminuisce il tempo per cui rimane acceso il pulsante)

ID: REQ-05 (non implementato per limitazione hardware)	
Nome	Tiro a vuoto – pulsanti 1 secondo, senior
Priorità	2
Versione	2.0
Note	
Sotto requisiti	
001	Il programma dura 100 pulsanti
002	Vengono mostrati punteggio (numero di pulsanti rimanenti) e tempo trascorso
003	Accensione casuale di tutti i pulsanti per 1 secondo
004	Accensione sequenziale
005	Alcuni pulsanti possono lampeggiare per simulare una finta, quindi se premuto, conta come errore
006	Il colpire una finta comporta: • Accelerazione del sistema (vedi sotto requisito 007); • Avviso verbale “Vuoto”; • Perdita di 5 punti;
007	Quando un pulsante viene premuto erroneamente la sequenza accelera (diminuisce il tempo per cui rimane acceso il pulsante)
008	Se i 3 pulsanti centrali si illuminano contemporaneamente l’utente deve fare un salto indietro e interrompere la barriera luminosa o premere una pedano con i piedi.

ID: REQ-06	
Nome	Test di Léger o Beep test, senior
Priorità	1
Versione	1.0
Note	Basato sul Test di Léger o Beep test o Bleep test (https://www.scienzadellosport.it/Test-Leger.php)
Sotto requisiti	
001	10 livelli, 30 pulsanti per livello
002	Frequenza di 1,4 [s] al livello 1 accelerando progressivamente fino al livello 10 (0,5 [s]).
003	Il livello viene mostrato sul display.
004	Se il giocatore sbaglia 3 colpi consecutivi la sequenza si interrompe: • Al primo colpo viene emesso un “Beep”; • Al secondo, ne vengono emessi 2 (“Beep Beep”); • Al terzo, 3 (“Beep Beep Beep”);
005	Se dopo un colpo errato, ne segue uno corretto, il contatore degli errori viene resettato.

ID: REQ-07	
Nome	50 pulsanti temporizzati, 1 secondo per pulsante, senior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	50 pulsanti in sequenza casuale.
002	Un display mostra i pulsanti rimanenti. L'altro, il numero di colpi corretti.
003	Il tempo per premere ogni pulsante è di 1 secondo.
004	Se il giocatore preme un pulsante errato (o in ritardo), la sequenza accelera.

ID: REQ-08	
Nome	Staffetta 4 giocatori (tempo totale 120 secondi), senior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	4 giocatori (30 [s] a testa) che giocano a staffetta
002	Un display mostra il tempo. L'altro i colpi corretti (punteggio).

ID: REQ-09	
Nome	Reazione, somma aritmetica
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma inizia premendo "@".
002	Premere un numero da 2 a 9 per selezionare il tempo per rispondere alla somma.
003	Vengono poste 8 semplici addizioni
004	Vengono mostrati i 2 addendi sui 2 display separatamente.
005	Il giocatore deve rispondere alla somma proposta entro il tempo impostato all'inizio.
006	Alla fine il punteggio viene mostrato sui display (Es: 6/8…)
007	Se il punteggio è di 8/8, viene emesso un riconoscimento vocale e un applauso.

ID: REQ-10	
Nome	Tabelline, test di velocità
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Premere il pulsante relativo alla tabellina che si vuole esercitare
002	Il sistema propone verbalmente delle tabelline
003	L'utente deve premere il pulsante relativo alla risposta. Se la risposta è di 2 cifre, premere prima le decine e poi le unità (Es: 18, premere prima 1 e poi 8).
004	Il test propone 12 tabelline
005	Il tempo è illimitato e mostrato sul display.

ID: REQ-11	
Nome	Stretching angolare, 25 pulsanti
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 25 pulsanti
002	Vengono mostrati punteggio (numero di pulsanti rimanenti) e conteggio colpi corretti.
003	Accensione casuale dei pulsanti agli estremi del telaio per 1 secondo
004	Accensione sequenziale
005	Quando un pulsante viene premuto erroneamente la sequenza accelera (diminuisce il tempo per cui rimane acceso il pulsante)

ID: REQ-12	
Nome	Stretching angolare, 50 pulsanti
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 50 pulsanti
002	Vengono mostrati punteggio (numero di pulsanti rimanenti) e conteggio colpi corretti.
003	Accensione casuale dei pulsanti agli estremi del telaio per 1 secondo
004	Accensione sequenziale
005	Quando un pulsante viene premuto erroneamente la sequenza accelera (diminuisce il tempo per cui rimane acceso il pulsante)

ID: REQ-13	
Nome	25 pulsanti temporizzati, 1 secondo per pulsante, junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	25 pulsanti in sequenza casuale (solo quelli <= 5, ossia sotto la metà del telaio).
002	Un display mostra i pulsanti rimanenti. L'altro, il numero di colpi corretti.
003	Il tempo per premere ogni pulsante è di 1 secondo.

ID: REQ-14	
Nome	50 pulsanti temporizzati, 1 secondo per pulsante, junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	50 pulsanti in sequenza casuale (solo quelli <= 5, ossia sotto la metà del telaio).
002	Un display mostra i pulsanti rimanenti. L'altro, il numero di colpi corretti.
003	Il tempo per premere ogni pulsante è di 1 secondo.

ID: REQ-15	
Nome	Cumulativo 30 secondi, junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 30 secondi
002	Vengono mostrati punteggio e tempo rimanente
003	Accensione casuale dei pulsanti (solo quelli <= 5, ossia sotto la metà del telaio)
004	Accensione sequenziale
005	Spegnimento quando premuti

ID: REQ-16	
Nome	Cumulativo 60 secondi, junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 60 secondi
002	Vengono mostrati punteggio e tempo rimanente
003	Accensione casuale dei pulsanti (solo quelli <= 5, ossia sotto la metà del telaio)
004	Accensione sequenziale
005	Spegnimento quando premuti

ID: REQ-17	
Nome	Corsa 25 pulsanti, junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 50 secondi o 25 pulsanti
002	Vengono mostrati punteggio e tempo rimanente (al decimo di secondo)
003	Accensione casuale dei pulsanti (solo quelli <= 5, ossia sotto la metà del telaio)
004	Accensione sequenziale
005	Spegnimento quando premuti
006	Conteggio dei pulsanti premuti correttamente

ID: REQ-18	
Nome	Corsa 50 pulsanti, junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 100 secondi o 50 pulsanti
002	Vengono mostrati punteggio e tempo rimanente (al decimo di secondo)
003	Accensione casuale dei pulsanti (solo quelli <= 5, ossia sotto la metà del telaio)
004	Accensione sequenziale
005	Spegnimento quando premuti
006	Conteggio dei pulsanti premuti correttamente

ID: REQ-19	
Nome	Maratona (3 minuti) junior
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Il programma dura 180 secondi
002	Vengono mostrati punteggio e tempo rimanente
003	Accensione casuale dei pulsanti (solo quelli <= 5, ossia sotto la metà del telaio)
004	Accensione sequenziale
005	Spegnimento quando premuti

ID: REQ-20	
Nome	Semplice gioco Simon: 20 pulsanti, 17 livelli
Priorità	1
Versione	2.0
Note	Basato sul gioco Simon Says (https://en.wikipedia.org/wiki/Simon_Says)
Sotto requisiti	
001	Premere i pulsanti riproducendo la sequenza mostrata
002	Al livello 1 si accendono 4 pulsanti
003	Ad ogni livello si accende un pulsante in più
004	Al termine di ogni sequenza (o livello) vengono emessi 2 “Beep”
005	Il gioco termina quando l’utente commette un errore o completa tutti e 17 o livelli
006	Vengono stampati il livello e il punteggio accumulato sui display

ID: REQ-21	
Nome	Flash test, 5 schemi
Priorità	1
Versione	2.0
Note	
Sotto requisiti	
001	Il sistema accende fino a 6 pulsanti, per 5 schemi (5 volte)
002	L'utente sceglie la durata dell'accensione dei pulsanti selezionando il tasto '@' più i secondi (da 1 a 8)
003	Quando il tempo termina il sistema emette 2 "Beep"
004	L'utente deve premere in un ordine casuale gli stessi pulsanti accesi in precedenza dal sistema
005	Ogni schema completato in modo corretto incrementa il punteggio
006	Vengono stampati il punteggio raggiunto e il punteggio massimo possibile

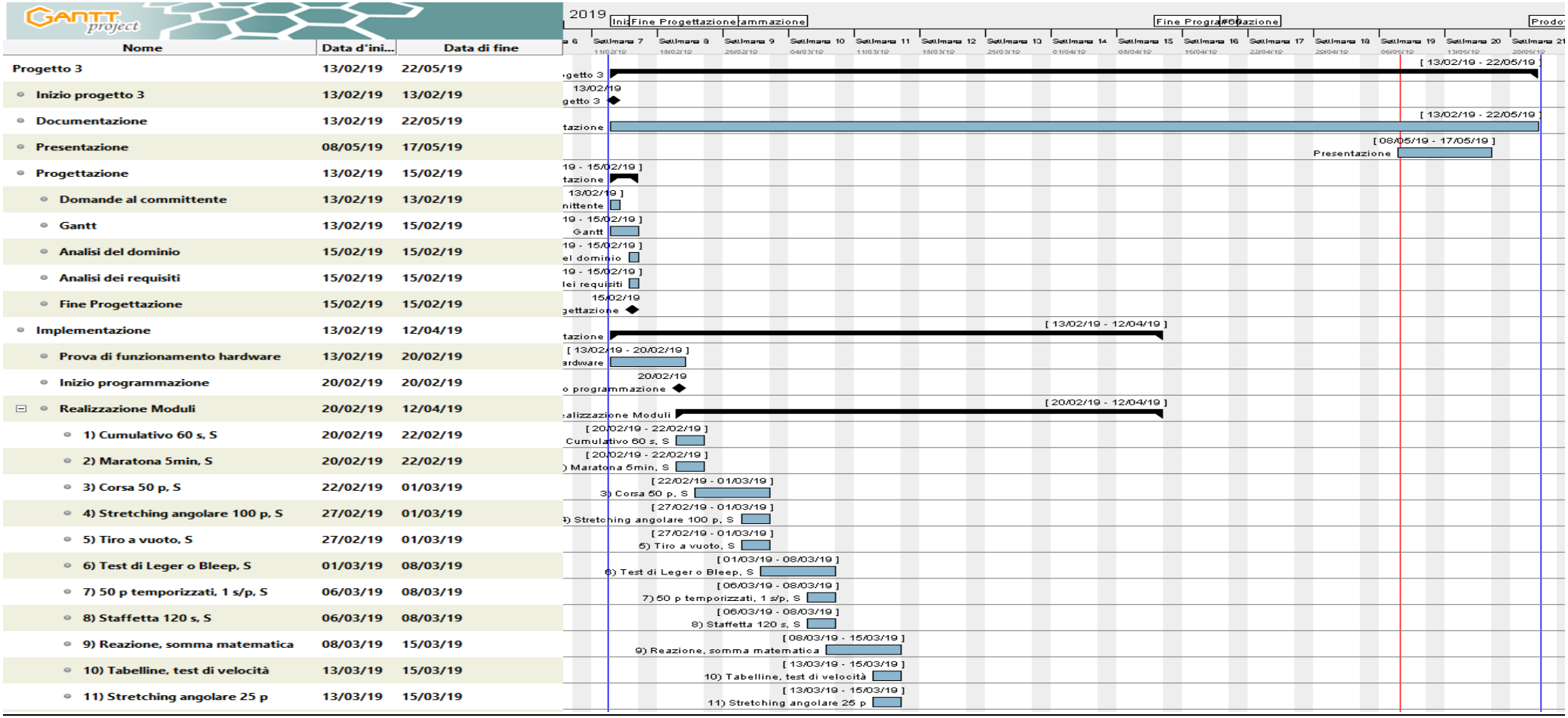
ID: REQ-22	
Nome	Anti - flash test
Priorità	1
Versione	2.0
Note	
Sotto requisiti	
001	Il sistema accende fino a 6 pulsanti, per 5 schemi (5 volte)
002	L'utente sceglie la durata dell'accensione dei pulsanti selezionando il tasto '@' più i secondi (da 1 a 8)
003	Quando il tempo termina il sistema emette 2 "Beep"
004	L'utente deve premere in un ordine casuale i pulsanti che non sono stati accesi in precedenza dal sistema
005	Ogni schema completato in modo corretto incrementa il punteggio
006	Vengono stampati il punteggio raggiunto e il punteggio massimo possibile

ID: REQ-23	
Nome	Reazione veloce, 10 schemi
Priorità	1
Versione	2.0
Note	
Sotto requisiti	
001	Il programma dura 10 schemi
002	L'utente seleziona i pulsanti, dopo aver cliccato il pulsante "#", che vuole utilizzare (massimo 11)
003	I pulsanti accesi in modo casuale sequenzialmente rimangono accesi fino a quando l'utente li preme
004	Il giocatore determina la velocità di gioco
005	Vengono stampati sul display tempo e pulsanti premuti in modo corretto

ID: REQ-24	
Nome	Errori e problemi da gestire
Priorità	1
Versione	2.0
Note	
Sotto requisiti	
001	Gestione problemi dal punto di vista elettrico, pulsanti incastrati
002	Gestione problemi led interni ai pulsanti (contatti problematici)
003	Led che rimangono costantemente accesi
004	Prevedere una procedura di check (all'accensione della macchina o su richiesta dell'operatore)
005	Individuare errori e segnalarli

1.6 Pianificazione

1.6.1 Preventivo



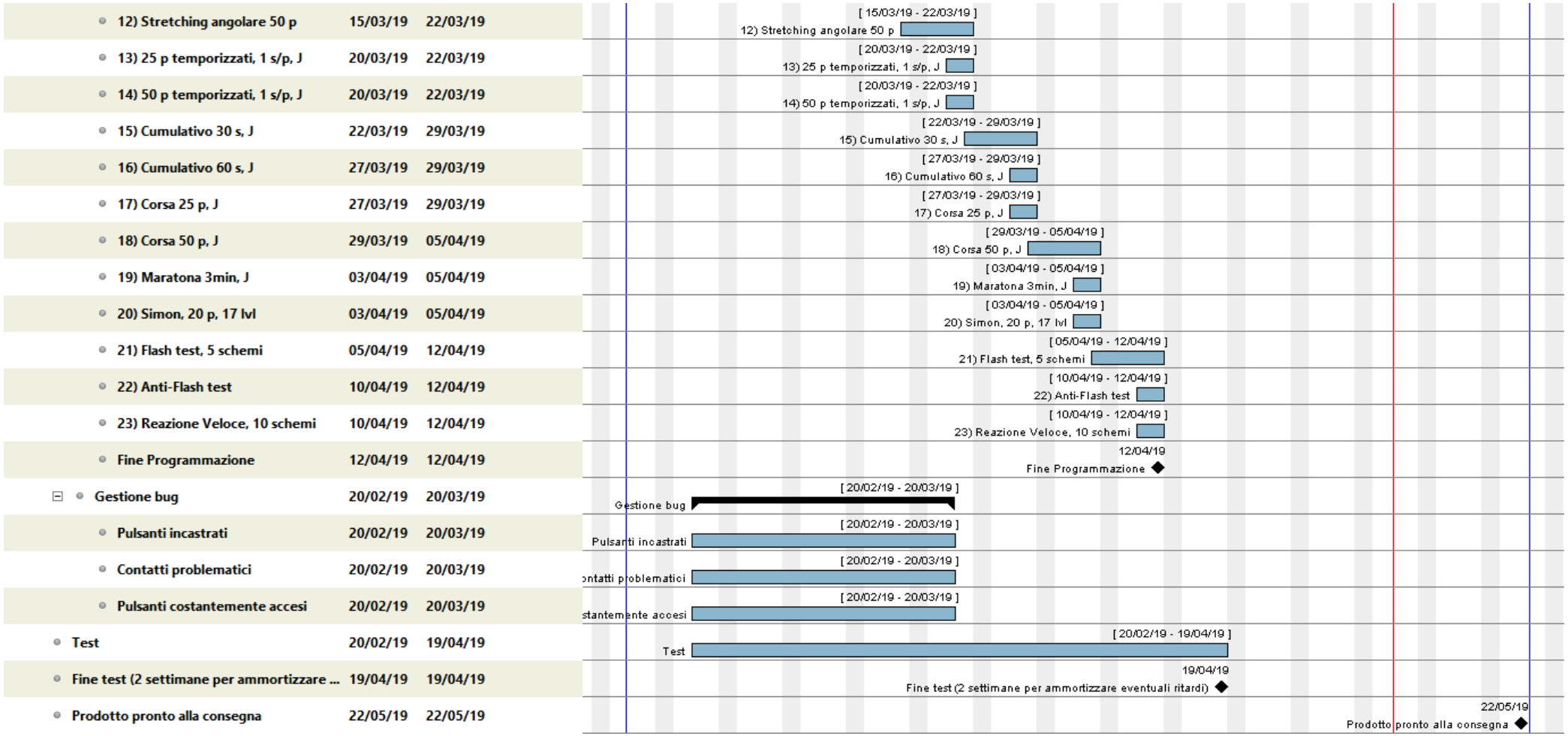


Figura 1 Gantt preventivo

Il progetto ci è stato commissionato il 13 febbraio 2019 e programmiamo di completarlo entro il 17 maggio 2019.

La prima settimana la dedicheremo alla progettazione. Per prima cosa realizzeremo l'intervista al committente in modo da poter chiarire qualunque eventuale dubbio in merito ai requisiti proposti. Sulla base del QdC e degli eventuali chiarimenti, realizzeremo il Gantt nel corso di 2 lezioni in modo da poterlo ancora adattare in caso di bisogno dopo aver realizzato l'analisi del dominio e dei requisiti.

In parallelo a questo, effettueremo vari test riguardo al funzionamento dell'hardware e apportare eventuali modifiche se necessario. Questa fase si estenderà per 3 lezioni circa.

A questo punto iniziamo la fase di implementazione: ogni modulo viene accompagnato in parallelo dai test relativi al funzionamento dello stesso e alla gestione di eventuali bug. Ogni modulo dovrebbe essere distribuito su circa 2 lezioni (8 ore scolastiche), ma (salvo inconvenienti) lavoreremo su 3 moduli contemporaneamente dividendoci il lavoro. I moduli da realizzare sono 23.

La gestione (e risoluzione) di eventuali bug dovrebbe terminare entro la prima metà del progetto o comunque entro quella data, non si dovrebbero più presentare problemi.

A questo punto abbiamo 2 settimane (16 ore lezione) libere in modo da poter recuperare eventuali ritardi. Se non dovessero essercene, cominceremo già con la realizzazione della presentazione che è altrimenti fissata per le ultime 2 settimane di lavoro.

1.7 Analisi dei mezzi

1.7.1 Software

I software che sono stati utilizzati per la realizzazione di progetto sono:

- Arduino 1.8.7
- Atom 1.34.0
- GanttProject 2.8.9
- Google Chrome 72.0
- Microsoft Word 2016
- Microsoft PowerPoint 2016
- StarUML
- Microsoft VS Code 1.33.1

Librerie utilizzate:

- Wire (Già presente in arduino)
- LiquidCrystal_I2C (https://github.com/arduino-libraries/LiquidCrystal)
- Adafruit_GFX (https://github.com/adafruit/Adafruit-GFX-Library)
- Adafruit_LEDBackpack (https://github.com/adafruit/Adafruit_LED_Backpack)
- SevSeg (https://github.com/DeanIsMe/SevSeg)

1.7.2 Hardware

Questo progetto richiede i seguenti hardware:

- Portatili senza particolari specifiche tecniche
  - Sistemi operativi: macOS Mojave, Windows 10
- Sistema fisico ricavato dal progetto preesistente "Batak"

2 Progettazione

In questo capitolo è rappresentato attraverso diagrammi e schemi come abbiamo deciso di progettare il nostro sistema, in modo da renderlo efficace e coerente con i requisiti richiesti.

2.1 Design dell'architettura del sistema

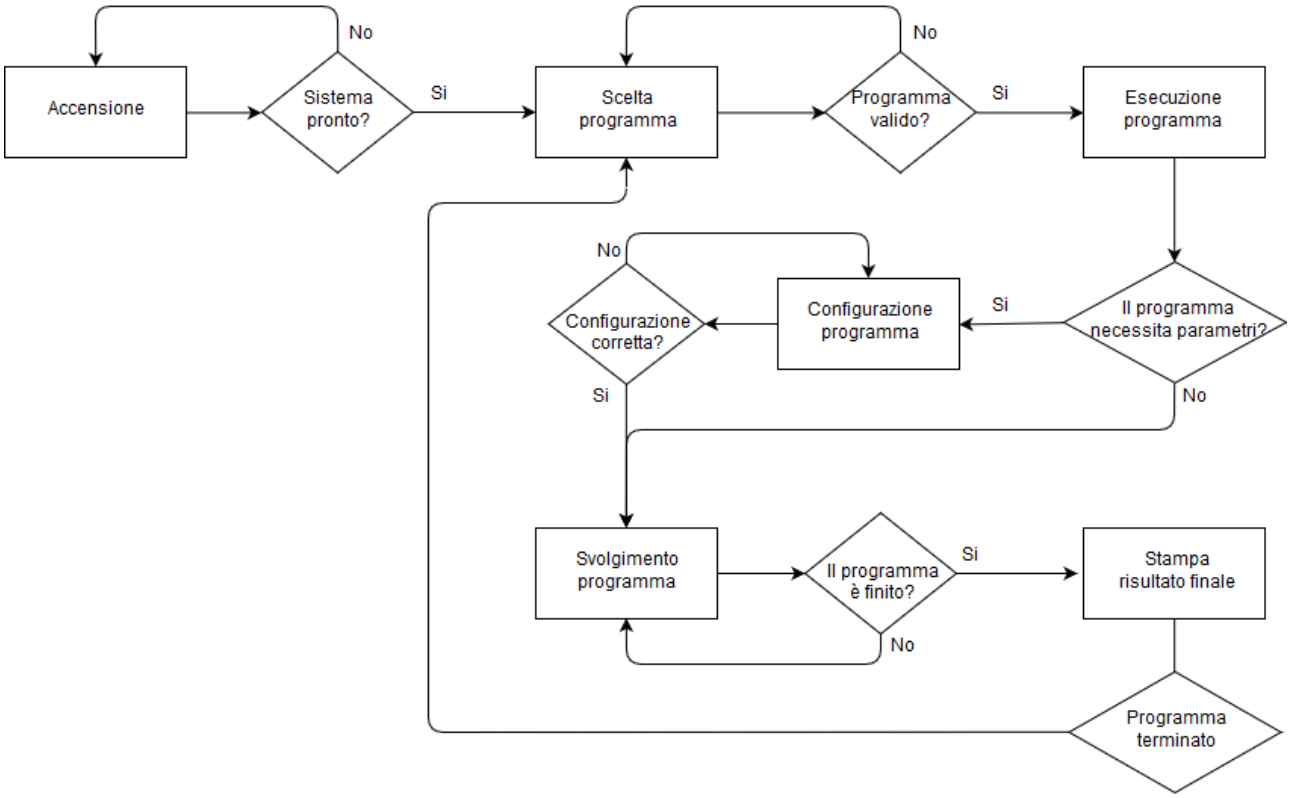


Figura 2 Design dell'architettura del sistema

2.2 Design procedurale

2.2.1 Programmi

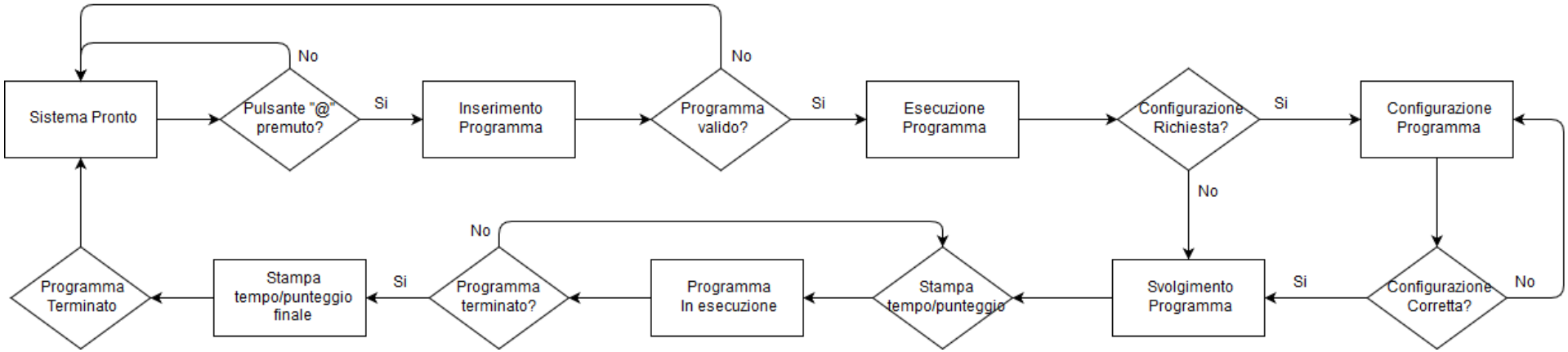


Figura 3 Diagramma procedurale "Programmi"

2.2.2 Procedura di check

Questo diagramma di flusso descrive il ciclo di vita della procedura di check che può essere selezionata dall'operatore digitando, nel menu per la scelta dei programmi" la seguente combinazione: "@" + "9" + "9"

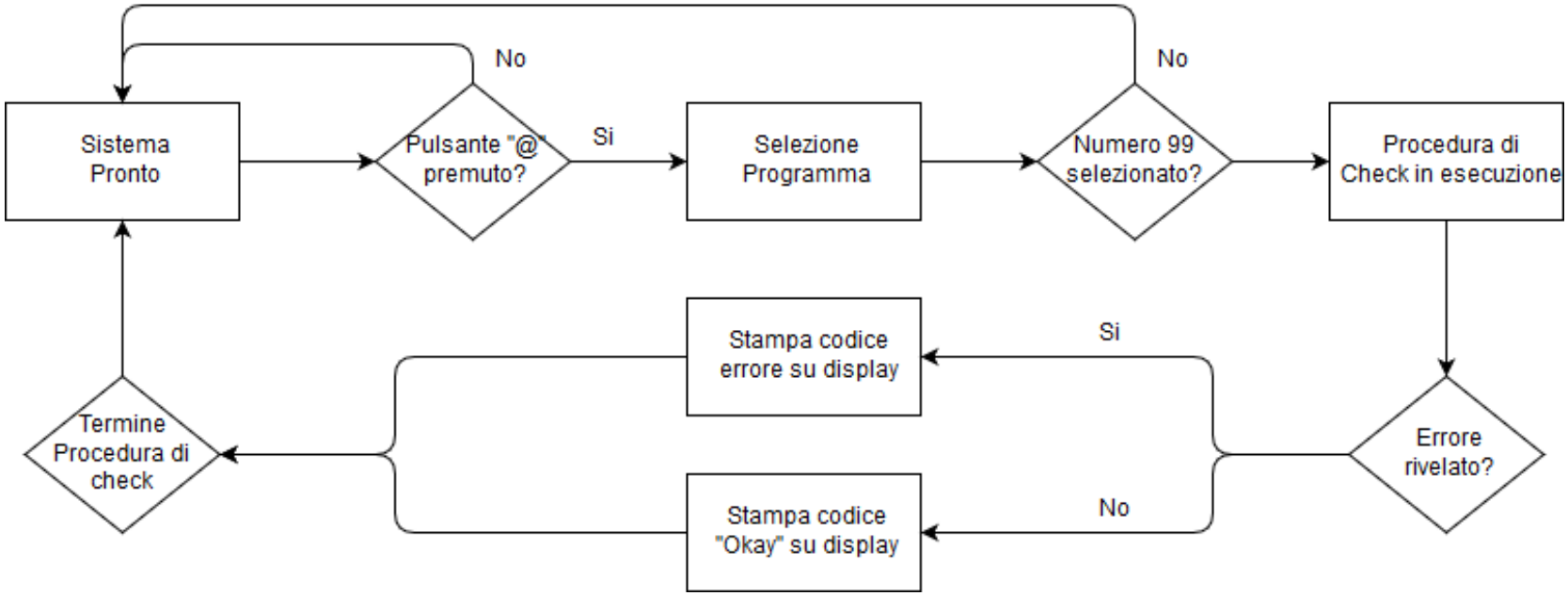


Figura 4 Diagramma procedurale "Procedura di check"

3 Implementazione

Il codice è strutturato in due progetti, uno riguarda l'Arduino UNO mentre l'altro riguarda l'Arduino Mega.
La struttura del codice che riguarda l'Arduino UNO è composto da un unico file chiamato "ArduinoUNO.ino".
Mentre la struttura del progetto per l'Arduino Mega è la seguente:

```
├── Global.ino
├── a_config.ino
├── b_functions.ino
├── c_songs.ino
├── d_programs.ino
└── z_main.ino
```

Questa struttura è stata creata in quanto Arduino compila i file che contiene un progetto in modo alfabetico.
Il file "Global.ino" è un file che rappresenta solamente un intestazione del progetto ed è il file di entrata per il compilatore. Il file "a_config.ino" è la parte di codice che si occupa di impostare i pin all'avvio di Arduino e di dichiarare variabili globali. All'interno del file "b_functions.ino" sono salvate tutte le funzioni utilizzate nei file successivi. Il codice del file "c_songs.ino" viene utilizzato per eseguire delle canzoncine personalizzate attraverso il buzzer. A seguire il file "d_programs.ino" che contiene il codice di tutti i programmi veri e propri. Come ultimo "z_main.ino" si occupa di avviare i programmi, gestire le interazioni dell'utente e molto altro.

All'interno del file "b_function.ino" troviamo anche la parte di codice che si occupa dell'interruzione dei vari programmi, in sostanza, se viene rilevato che il pulsante "@" è premuto, parte un timer, e se tale bottone rimane premuto per più di 2,5 secondi, il programma in corso viene terminato:

```
104 void setLastState(int pin, bool state)
105 {
106   int index = getLabel(pin);
107   if (state == HIGH && pin == 44)
108   {
109     if (resetTime != 0)
110     {
111       long elapsed = millis() - resetTime;
112       Serial.println(elapsed);
113       if (elapsed > 2500)
114       {
115           programRunning = false;
116           resetTime = 0;
117       }
118     }
119     else
120     {
121       resetTime = millis();
123     }
124
125
126
```

```
127     }
128     else if (state == LOW && pin == 44)
129     {
130       resetTime = 0;
131     }
132     buttonStatus[index] = state;
133   }
134 }
```

Un' altra funzione molto utilizzata all'interno dei programmi nel file delle funzioni generali è la seguente. Permette di ricavare un pin di un bottone randomico nel range 22 e 45, il controllo supplementale è dovuto al fatto che vi possono essere delle interferenze.

```
21  int getRandom(int pins[], int size) {
22    int pin = pins[random(0, size + 1)];
23    if (pin >= 22 && pin <= 45)
24    {
25      return pin;
26    }
27    return getRandom(pins, size);
28  }
```

Altro metodo importantissimo è il seguente, che permette di inviare un valore di tipo long attraverso le connessione SDA e SCL. Viene richiesto inoltre un parametro "type" per definire su quale 7 segmenti stampare il dato.

```
66  void sendData(byte type, long value)
67  {
68    Wire.beginTransmission(8);
69    byte times[5];
70    times[0] = type;
71    times[1] = (value >> 24) & 0xFF;
72    times[2] = (value >> 16) & 0xFF;
73    times[3] = (value >> 8) & 0xFF;
74    times[4] = value & 0xFF;
75    Wire.write(times, 5);
76    Wire.endTransmission();
77  }
```

Oltre a questi metodi considerati principali sono presenti altri metodi di supporto come per esempio il reset dello stato dei pulsanti, il reset dei sette segmenti, il reset del display lcd e la verifica per controllare se un pulsante è premuto oppure no.

3.1 Cumulativo 60 secondi, senior

Il programma viene eseguito dal metodo cumulative che richiede 2 parametri, uno dei quali rappresenta la durata del programma definito come long (valori positivi), il secondo invece definisce se il programma è senior o junior tramite un parametro booleano.

```
425 void cumulative(long duration, boolean senior) {
```

Il programma, come prima cosa, rileva l'istante in cui il programma è stato avviato, poi setta a 0 la variabile che definisce il tempo trascorso dall'avvio del programma. Di seguito inizializza anche la variabile che conterrà l'indice del pulsante da premere ("elapsed").

```
429 long start = millis();
430 long elapsed = 0;
431 int currentPin;
```

In seguito viene eseguito un controllo per capire se il programma è per i senior o per i junior. Se il programma è per senior, viene richiamato il metodo per estrarre casualmente il pulsante da premere fra tutti quelli disponibili; se junior, la lista di pulsanti da estrarre casualmente si limita a quelli che si trovano nella metà inferiore del telaio. In entrambi i casi, il pin del bottone estratto, viene salvato nella variabile "currentPin". Dunque, viene acceso il led relativo al bottone estratto nella variabile "currentPin".

```
432 if (senior)
433 {
434    currentPin = getRandom(buttonPins, SIZE);
435 }
436 else
437 {
438    currentPin = getRandom(juniorPins, JUNIOR_SIZE);
439 }
440 int pressedButtons = 0;
441 digitalWrite(currentPin + 1, HIGH);
```

A questo punto viene avviato un ciclo che ha una durata uguale a quella ricevuta da parametro.

Nel ciclo, la prima cosa che viene fatta è la rilevazione dell'istante corrente, che viene sottratto con l'istante di partenza del programma per rilevare il tempo trascorso, appunto dall'avvio del programma.

Ora il programma verifica se lo stato dell'ultimo bottone premuto è ad HIGH (o 1, che corrisponde a "premuto") e che lo stato dell'ultimo bottone premuto è diverso dallo stato corrente dello stesso bottone. Nel caso in cui entrambi i vincoli siano rispettati, il programma procede correttamente in caso contrario termina.

Il programma prosegue incrementando il contatore dei pulsanti premuti fino a questo punto e invia tale dato all'Arduino che si occupa di gestire i display 7 segmenti che rappresentano, appunto il punteggio (che corrisponde al numero di pulsanti premuti) e il tempo rimanente.

```
443 while (elapsed <= duration)
444 {
```

```
445     elapsed = (millis() - start);
446     sendData(0, elapsed / 100);
447     bool presslastState = getLastState(currentPin);
448     bool presscurrentState = isPressed(currentPin);
449     if (presslastState == HIGH && presslastState != presscurrentState)
450     {
451       pressedButtons++;
452       sendData(1, pressedButtons);
453       Serial.print("Premuto: ");
454       Serial.println(currentPin);
455       digitalWrite(currentPin + 1, LOW);
456       if (senior)
457       {
458         currentPin = getRandom(buttonPins, SIZE);
459       }
460       else
461       {
462         currentPin = getRandom(juniorPins, JUNIOR_SIZE);
463       }
464       Serial.print("Accendo: ");
465       Serial.println(currentPin);
466       digitalWrite(currentPin + 1, HIGH);
467     }
468 }
```

Esempio di utilizzo:

```
XXX  cumulative(60, true);
```

3.2 Maratona (5 minuti) senior

(Vedi capitolo 3.1)

Esempio di utilizzo:

```
XXX  cumulative(300, true);
```

3.3 Corsa 50 pulsanti, senior

Il programma viene eseguito dal metodo *rush* che richiede 3 parametri, uno dei quali rappresenta la durata del programma definito come long (valori positivi), il secondo indica il numero massimo di pulsanti da premere sotto forma di intero, mentre il terzo definisce se il programma è senior o junior tramite un parametro booleano.

```
606  void rush(long duration, int maxbuttons, boolean senior) {
```

Il programma, come prima cosa, rileva l'istante in cui il programma è stato avviato, poi setta a 0 la variabile che definisce il tempo trascorso dall'avvio del programma ("*elapsed*"). Di seguito inizializza anche la variabile che definisce quanti pulsanti sono stati premuti.

```
607  long start = millis();
608  long elapsed = 0;
609  int pressedButton=0;
611  int currentPin;
```

In seguito viene eseguito un controllo per capire se il programma è per i senior o per i junior. Se il programma è per senior, viene richiamato il metodo per estrarre casualmente il pulsante da premere fra tutti quelli disponibili; se junior, la lista di pulsanti da estrarre casualmente si limita a quelli che si trovano nella metà inferiore del telaio. In entrambi i casi, il pin del bottone estratto, viene salvato nella variabile "*currentPin*". Dunque, viene acceso il led relativo al bottone estratto nella variabile "*currentPin*".

```
612  if (senior)
613  {
614     currentPin = getRandom(buttonPins, SIZE);
615  }
616  else
617  {
618     currentPin = getRandom(juniorPins, JUNIOR_SIZE);
619  }
620  digitalWrite(currentPin + 1, HIGH);
```

A questo punto viene avviato un ciclo che si interrompe o quando si esaurisce il tempo ricevuto da parametro, o se il programma viene interrotto ("programRunning=false"), o ancora se il giocatore preme tutti i pulsanti in tempo.

Nel ciclo, la prima cosa che viene fatta è la rilevazione dell'istante corrente, che viene sottratto con l'istante di partenza del programma per rilevare il tempo trascorso, appunto dall'avvio del programma.

Ora il programma verifica se lo stato dell'ultimo bottone premuto è ad HIGH (o 1, che corrisponde a "premuto") e che lo stato dell'ultimo bottone premuto è diverso dallo stato corrente dello stesso bottone. Nel caso in cui entrambi i vincoli siano rispettati, il programma procede correttamente, in caso contrario termina.

Il programma prosegue incrementando il contatore dei pulsanti premuti fino a questo punto e invia tale dato all'Arduino che si occupa di gestire i display 7 segmenti che rappresentano, appunto il punteggio (che corrisponde al numero di pulsanti premuti) e il tempo rimanente.

In seguito viene eseguito una nuova estrazione per il prossimo pulsante estraendolo come sopra (eseguendo anche la distinzione fra senior e junior).

```
623 while (elapsed <= duration && pressedButtons < maxbuttons &&
    programRunning)
624 {
625   isPressed(44);
626   elapsed = (millis() - start);
627   sendData(0, elapsed / 100);
628   bool presslastState = getLastState(currentPin);
629   bool presscurrentState = isPressed(currentPin);
630   if (presslastState == HIGH && presslastState != presscurrentState)
631   {
632     pressedButtons++;
633     sendData(1, pressedButtons);
634     digitalWrite(currentPin + 1, LOW);
635     if (senior)
636     {
637       currentPin = getRandom(buttonPins, SIZE);
638     }
639     else
640     {
641       currentPin = getRandom(juniorPins, JUNIOR_SIZE);
642     }
643     digitalWrite(currentPin + 1, HIGH);
644   }
645 }
```

Esempio di utilizzo:

```
XXX cumulative(100, 50, true);
```

3.4 Stretching angolare, 100 pulsanti, senior

Il programma viene eseguito dal metodo *angularStretching* che richiede 2 parametri, uno dei quali indica il numero massimo di pulsanti da premere sotto forma di intero, mentre il secondo definisce quale programma deve essere avviato.

```
534 void angularStretching(int maxbuttons, int program) {
```

Il programma, come prima cosa cambia il valore della costante *ANGULAR_STRETCHING_SIZE* in 7, poi salva una variabile intera che definisce il timeout fra 1 pulsante e l'altro, a questo punto genera l'arra< *angularPins* di dimensione *ANGULAR_STRETCHING_SIZE* inserendovi anche i valori corrispondenti ad ogni pulsante esterno del telaio.

In seguito rileva l'istante in cui il programma è stato avviato, poi setta a 0 la variabile che definisce il tempo trascorso dall'avvio del programma ("*elapsed*"). A seguire inizializza anche la variabile intera *leftButtons* che definisce quanti pulsanti rimangono da premere (inizialmente è uguale al numero massimo di pulsanti da premere ottenuto come parametro del metodo.

A questo punto viene istanziata la variabile *currentPin* che contiene l'indice del pin corrente, inizialmente viene riempita con un valore casuale fra quelli dell'array sopracitato e il led di tale pulsante viene acceso.

```
535 #define ANGULAR_STRETCHING_SIZE 7
536 int timeout = 1000;
537 int angularPins[ANGULAR_STRETCHING_SIZE] = {22, 42, 24, 30, 32, 38, 40};
538 long start = millis();
539 long startButton = millis();
540 long elapsed = 0;
541 int leftButtons = maxbuttons;
542
543 int currentPin = getRandom(angularPins, ANGULAR_STRETCHING_SIZE);
544 digitalWrite(currentPin + 1, HIGH);
```

Ora comincia il ciclo effettivo di gioco, il quale non si interrompe fino a che non vengono premuti tutti quanti i pulsanti, oppure se il programma viene interrotto ("programRunning=false").

```
546 while (leftButtons > 0 && programRunning){
```

All'interno del ciclo, vengono fatti vari controlli:

1. Se il programma richiesto è il 4, vengono inviati i dati necessari all'Arduino secondario (display superiori);
2. Se il tempo trascorso fra la pressione di un pulsante e l'altro è maggiore al timeout, il pulsante precedente viene spento e ne viene acceso uno nuovo casuale;
3. Se è stato premuto il pulsante corretto, viene decrementato il numero di pulsanti rimanenti e tale valore viene inviato all'Arduino secondario, in seguito viene spento quello vecchio e acceso uno nuovo casuale;
4. A questo punto c'è un altro ciclo che al suo interno controlla che il pulsante premuto sia quello corretto e se non è così decrementa il tempo a disposizione del giocatore fra un pulsante e l'altro di 100 millisecondi fino ad un minimo di 200 millisecondi. Lo stato di partenza è 1000 millisecondi (= 1 secondo).

```
548 elapsed = (millis() - start);
549 if (program == 4)
550 {
551    sendData(0, elapsed / 100);
552 }
553 if (millis() - startButton > timeout)
554 {
555    digitalWrite(currentPin + 1, LOW);
556    currentPin = getRandom(angularPins, ANGULAR_STRETCHING_SIZE);
557    startButton = millis();
558    digitalWrite(currentPin + 1, HIGH);
559 }
560 bool presslastState = getLastState(currentPin);
561 bool presscurrentState = isPressed(currentPin);
562 if (presslastState == HIGH && presslastState != presscurrentState)
563 {
564    leftButtons--;
565    sendData(1, leftButtons);
566    if (program != 4)
567    {
568       sendData(0, maxbuttons - leftButtons);
569    }
570    digitalWrite(currentPin + 1, LOW);
571    currentPin = getRandom(angularPins, ANGULAR_STRETCHING_SIZE);
572    startButton = millis();
573    digitalWrite(currentPin + 1, HIGH);
574 }
575 for (int i = 0; i < SIZE; i++)
576 {
577    int pin = buttonPins[i];
578    if (pin == currentPin)
579    {
580       continue;
581    }
582    bool lastState = getLastState(pin);
583    bool currentState = isPressed(pin);
284    if (currentState == HIGH && currentState != lastState)
585    {
586       timeout -= 100;
587       if (timeout < 200)
588       {
589          timeout = 200;
590       }
591       setLcdText("Hai sbagliato,", "diminuisco il delay", "", "BATTAK
592       2.0");
593    }
594 }
```

Esempio di utilizzo:

```
xxx angularStretching(100, 4);
```

3.5 (tiro a vuoto, - pulsanti 1 secondo, senior)

(Programma non implementato)

3.6 Test di Lèger o Beep test, senior

Il programma viene eseguito dal metodo *beepTest*.

```
704 void beepTest() {
```

Il programma, come prima cosa imposta delle variabili che si possono definire di configurazione. In ordine: il numero di livelli, il numero di bottoni per ogni livello e il timeout per poter premere ogni livello.

In seguito rileva l'istante in cui il programma è stato avviato, poi setta a 0 la variabile che definisce il tempo trascorso dall'avvio del programma ("*elapsed*"). A seguire inizializza anche le variabili corrispondenti al numero di pulsanti premuti e al numero di errori commessi.
L'ultima variabile contiene il valore estratto a random corrispondente al pulsante da premere e poi accende il led relativo a tale pulsante.

```
705 int levels = 10;
706 int buttons = 10;
707 int timeout = 1400;
708
709 long elapsed = 0;
710 long start = millis();
711 long startButton = millis();
712 int pressedButtons = 0;
713 int errors = 0;
714 int currentPin = getRandom(buttonPins, SIZE);
715 digitalWrite(currentPin + 1, HIGH);
```

Ora troviamo un ciclo che prosegue fino a quando non sono stati completati tutti i livelli, o se vengono commessi più di 2 errori oppure ancora se non viene registrato il comando di interruzione.

```
716 for (int l = 0; l < levels && errors <= 2 && programRunning; l++){
```

A questo punto gestiamo i livelli.
La prima cosa che viene fatta è scrivere sul display LCD un messaggio che indichi all'utente il livello in cui si trova.

```
718 setLcdText("Livello:", String(l + 1), "", "BATTAK 2.0");
```

Un nuovo ciclo gestisce ogni livello. Questo si interrompe solo quando tutti i pulsanti sono stati premuti, oppure nuovamente, se sono stati registrati più di 2 errori o se è stato ricevuto il comando di interruzione.

```
719 while (pressedButtons <= buttons * l && errors <= 2 && programRunning){
```

L'interno di questo ciclo si sviluppa con 2 condizioni e un'altro ciclo dopo di esse.
La prima cosa che viene fatta è registrare il tempo trascorso dall'inizio del programma e inviare tale dato all'Arduino secondario.

```
721 elapsed = (millis() - start);
722 sendData(0, elapsed / 100);
```

La prima condizione verifica se il tempo trascorso dall'ultimo pulsante premuto sia maggiore al timeout a disposizione. Se tale condizione risulta vera, il pulsante corrente viene spento, ne viene estratto uno nuovo e viene aggiunto 1 errore.
A seconda del numero di errori viene emesso un suono intermittente che corrisponde appunto al numero di errori.
Dunque viene registrato l'istante in cui viene acceso il pulsante successivo.

```
723 if (millis() - startButton > timeout)
724 {
725   digitalWrite(currentPin + 1, LOW);
726   currentPin = getRandom(buttonPins, SIZE);
727   errors++;
728   for (int i = 0; i < errors; i++)
729   {
730     tone(buzzerPin, 2000);
731     delay(100);
732     noTone(buzzerPin);
733     delay(50);
734   }
735   startButton = millis();
736   digitalWrite(currentPin + 1, HIGH);
737 }
```

Ora viene verificato se è stato premuto il pulsante corretto.
Se è così, viene azzerato il contatore di errori e incrementato il numero di pulsanti premuti (tale valore viene inviato al secondo Arduino.
Quindi viene spento il pulsante corrente e ne viene estratto uno nuovo (memorizzando tale istante) e questo viene acceso.

```
740 if (presslastState == HIGH && presslastState != presscurrentState)
741 {
742   errors = 0;
743   pressedButtons++;
744   sendData(1, pressedButtons);
745   digitalWrite(currentPin + 1, LOW);
746   currentPin = getRandom(buttonPins, SIZE);
747   startButton = millis();
748   digitalWrite(currentPin + 1, HIGH);
}
```

In questo ciclo, vengono controllati tutti I pulsanti del telaio per verificare se uno di questi è premuto. Se non ne viene rilevato nemmeno 1, il ciclo si interrompe subito.
Ora viene controllato se il pin premuto non è quello corretto. Se questa condizione è soddisfatta, viene incrementato il numero di errori e vengono prodotti i suoni che indicano a quale errore ci troviamo.

```
750 for (int i = 0; i < SIZE; i++)
751 {
752   int pin = buttonPins[i];
753   if (pin == currentPin)
754   {
755     continue;
756   }
757   bool lastState = getLastState(pin);
758   bool currentState = isPressed(pin);
759   if (currentState == HIGH && currentState != lastState)
760   {
761     errors++;
762     for (int i = 0; i < errors; i++)
763     {
764       tone(buzzerPin, 2000);
765       delay(100);
766       noTone(buzzerPin);
767       delay(50);
768     }
   }
}
```

Esempio di utilizzo:

```
XXX beepTest();
```

3.7 50 Pulsanti temporizzati, 1 secondo per pulsante, senior

Il programma viene eseguito dal metodo temporized che richiede 2 parametri, il primo parametro ***maxbuttons*** richiede un numero intero positivo che indica il numero massimo di pulsanti che l'utente deve premere, il secondo invece definisce se il programma è senior o junior tramite un parametro booleano.

```
449 void temporized(int maxbuttons, boolean senior) {
```

Il programma, come prima cosa, rileva l'istante in cui il programma è stato avviato, poi setta a 0 la variabile che definisce il tempo trascorso dall'avvio del programma. Di seguito inizializza anche la variabile che conterrà l'indice del pulsante da premere ***elapsed***. La variabile ***pressedButtons*** serve conta il numero di pulsanti premuti correttamente da parte dell'utente. Mentre la variabile ***currentPin*** rappresenta il pin corrente da premere.

```
452 long start = millis();
453 long startButton = millis();
454 long elapsed = 0;
455 int pressedButtons = 0;
456 int currentPin;
```

In seguito viene eseguito un controllo per capire se il programma è per i senior o per i junior. Se il programma è per senior, viene richiamato il metodo per estrarre casualmente il pulsante da premere fra tutti quelli disponibili; se junior, la lista di pulsanti da estrarre casualmente si limita a quelli che si trovano nella metà inferiore del telaio. In entrambi i casi, il pin del bottone estratto, viene salvato nella variabile ***currentPin***. Dunque, viene acceso il led relativo al bottone estratto nella variabile ***currentPin***.

```
457 if (senior){
458   currentPin = getRandom(buttonPins, SIZE);
459 }
460 else{
461   currentPin = getRandom(juniorPins, JUNIOR_SIZE);
462 }
463 digitalWrite(currentPin + 1, HIGH);
```

Il ciclo while mostrato qua sotto è la parte del programma che esegue la modalità di gioco. Il programma continua a funzionare fino a quando il numero di pulsanti premuti in modo corretto corrisponde al numero di pulsanti massimo raggiungibile, oppure fino a quando viene attivata la procedura di uscita forzata che in questo caso è rappresentato dalla variabile ***programRunning***.

Nel ciclo, la prima cosa che viene fatta è la rilevazione dell'istante corrente, che viene sottratto con l'istante di partenza del programma per rilevare il tempo trascorso, appunto dall'avvio del programma.

Viene incrementato il contatore dei bottoni premuti correttamente ogni volta che l'utente esegue un colpo corretto.

Viene inoltre controllato se viene premuto un pulsante sbagliato. Se questo avviene il tempo che l'utente ha a disposizione per premere il pulsante corretto diminuisce fino ad un massimo di 0.1 secondi.

```
467 while (pressedButtons < maxbuttons && programRunning){
468   elapsed = millis() - startButton;
```

```
469   sendData(0, (timeout - elapsed) / 100);
470   if (elapsed > timeout){
471     digitalWrite(currentPin + 1, LOW);
472     if (senior){
473       currentPin = getRandom(buttonPins, SIZE);
474     }
475     else{
476       currentPin = getRandom(juniorPins, JUNIOR_SIZE);
477     }
478     startButton = millis();
479     digitalWrite(currentPin + 1, HIGH);
480   }
481   bool presslastState = getLastState(currentPin);
482   bool presscurrentState = isPressed(currentPin);
483   if (presslastState == HIGH && presslastState != presscurrentState){
484     pressedButtons++;
485     sendData(1, pressedButtons);
486     digitalWrite(currentPin + 1, LOW);
487     if (senior){
488       currentPin = getRandom(buttonPins, SIZE);
489     }
490     else{
491       currentPin = getRandom(juniorPins, JUNIOR_SIZE);
492     }
493     startButton = millis();
494     digitalWrite(currentPin + 1, HIGH);
495   }
496   for (int i = 0; i < SIZE; i++){
497     int pin = buttonPins[i];
498     if (pin == currentPin){
499       continue;
500     }
501     bool lastState = getLastState(pin);
502     bool currentState = isPressed(pin);
503     if (currentState == HIGH && currentState != lastState){
504       timeout -= 100;
505       if (timeout < 200){
506         timeout = 200;
507       }
508       setLcdText("Hai sbagliato,", "diminuisco il delay", "", "BATTAK
    2.0");
509     }
510   }
511 }
```

Esempio di utilizzo:

XXX	`temporized(25, true);`

3.8 Staffetta 4 giocatori (tempo totale 120 secondi), senior

Il programma staffetta 4 giocatori di 120 secondi utilizza la funzione **cumulative(long duration, boolean senior)** (vedi capitolo 3.1) 4 volte di fila di durata 30 secondi. Tra un giocatore e l'altro si hanno 5 secondi di pausa per cambiare giocatore.

```
126 //eseguo il programma staffetta 4 giocatori senior
127 for (int i = 0; i < 4 && programRunning; i++){
128   cumulative(30000, true);
129   long start = millis();
130   while (millis() - start <= 5000 && programRunning){
131     long elapsed = millis() - start;
132     sendData(0, elapsed / 100);
133   }
134 }
```

Esempio di utilizzo:

XXX	`cumulative(30000, true);`

3.9 Reazione, somma matematica

Il programma somma matematica viene eseguito dal metodo **mathSum** che non richiede parametri.

```
781 void mathsum() {
```

Nella prima parte di codice vengono inizializzate le variabili ***waiting***, ***millisec*** e ***points***. Viene inoltre stampata a display la stringa di codice “Seleziona un tempo entro il quale rispondere (2 – 9 s)”.

```
783 bool waiting = true;
784 int millisec = 0;
785 int points = 0;
786 setLcdText("Seleziona un tempo", "entro il quale", "rispondere (2 - 9 s)",
    "BATTAK 2.0");
```

Successivamente il sistema aspetta che venga inserito un valore da 2 a 9 secondi, che rappresenterà il tempo massimo che l’utente ha a disposizione per rispondere a calcolo proposto sul display. Se l’utente inserisce un valore non consentito il sistema richiede il valore affinché sia valido.

```
787 while (waiting)
788 {
789   for (int i = 0; i < SIZE; i++)
790   {
791     int bpin = buttonPins[i];
792     bool lastState = getLastState(bpin);
793     bool currentState = isPressed(bpin);
794     if (currentState == HIGH && currentState != lastState)
795     {
796       digitalWrite(bpin + 1, HIGH);
797       int label = getLabel(bpin);
798       if (label >= 2 && label <= 9)
799       {
800         millisec = label * 1000;
801         waiting = false;
802       }
803       else
804       {
805         return;
806       }
807     }
808   }
809 }
```

Successivamente viene effettuato un reset di tutti i pulsanti e a display appare il numero di secondi a disposizione e dopo 3 secondi il sistema inizia a stampare i calcoli.

```
setLcdText("Hai selezionato: " + String(millisec / 1000), "Inizio in 3 secondi", "", "BATTAK 2.0");
delay(3000);
resetButtonsState();
resetLeds();
```

Viene eseguito per 8 volte un ciclo in cui viene proposto un calcolo casuale. Il primo while server ad ottenere una somma che sia inferiore a 10, questo perché l'utente può premere solo un pulsante per rispondere al calcolo. Una volta che il sistema ha creato un calcolo valido lo propone all'utente e da quel momento parte il timer scelto in precedenza dall'utente per rispondere alla domanda.

Quando l'utente preme un pulsante viene controllato che la somma sia corretta. Se è corretta viene incrementato il punteggio altrimenti si passa direttamente al calcolo successivo.

```
for (int i = 0; i < 8; i++)
{
  int sum = 10;
  int one = 0;
  int two = 0;
  while (sum >= 10)
  {
    one = random(0, 9);
    two = random(0, 9);
    sum = one + two;
  }
  setLcdText(String(one) + " + " + String(two) + " = ?", "Punti: " + String(points) + "/8", "", "BATTAK 2.0");
  sendData(0, one);
  sendData(1, two);

  startTime = millis();
  waiting = true;
  while (millis() - startTime <= millisec && waiting && programRunning)
  {
    for (int x = 0; x < SIZE; x++)
    {
      int bpin = buttonPins[x];
      bool lastState = getLastState(bpin);
      bool currentState = isPressed(bpin);
      if (currentState == HIGH && currentState != lastState && getLabel(bpin) == sum)
      {
        points++;
        setLcdText(String(one) + " + " + String(two) + " = ?", "Punti: " + String(points) + "/8", "", "BATTAK 2.0");
        waiting = false;
```

```
844         }
845       }
846     }
847 }
```

Quando sono stati proposti tutti e 8 i calcoli viene stampato il punteggio totale. Se l'utente ha risposto correttamente a tutti i calcoli il sistema riproduce una canzoncina di vittoria.

```
848 if (points == 8)
849 {
850   setLcdText("Punteggio massimo!", "", "", "BATTAK 2.0");
851   winSong();
852 }
853 setLcdText("Hai finito!", "Punti: " + String(points) + "/8", "", "BATTAK
    2.0");
```

Esempio di utilizzo:

```
XXX mathsum();
```

3.10 Tabelline, test di velocità

Il programma viene eseguito dal metodo *boards*.

```
54  void boards() {
```

Le prime cose che troviamo sono i reset dei led e degli stati dei bottoni, seguiti dal metodo che mostra un messaggio tramite il display LCD.
Vengono anche dichiarate 2 variabili, 1 per definire lo stato del programma (se *true* indica che il programma sta aspettando che gli venga passata la tabellina da calcolare), mentre la seconda definisce la tabellina da calcolare.

```
55  resetLeds();
56  resetButtonsState();
57  setLcdText("Seleziona una", "tabellina da 2 a 9", "", "BATTAK 2.0");
58  bool waiting = true;
59  int boards = 0;
```

Ora troviamo il ciclo che attende che venga definita la tabellina, infatti il ciclo non si interrompe fino a quando non viene scelta una tabellina valida o se viene registrato il comando di interruzione.

```
60  while (waiting && programRunning){
```

All'interno di questo ciclo troviamo un altro ciclo che scorre tutti i pulsanti del telaio. Per ognuno di questi verifica se è premuto e nuovamente, se uno è premuto, verifica che il suo indice corrisponda ad un valore valido (indica maggiore o uguale a 2 e minore o uguale a 9). Se anche questa condizione è valida viene salvata la tabellina scelta e invertito lo stato del gioco (ossia comincia il gioco vero e proprio).

```
55  for (int i = 1; i <= SIZE; i++)
56  {
57    int bpin = buttonPins[i];
58    bool lastState = getLastState(bpin);
59    bool currentState = isPressed(bpin);
60    if (currentState == HIGH && currentState != lastState)
61    {
62      digitalWrite(bpin + 1, HIGH);
63      int label = getLabel(bpin);
64      if (label >= 2 && label <= 9)
65      {
66        boards = label;
67        waiting = false;
68      }
69    }
70  }
```

A questo punto viene scritto un messaggio sul display LCD e memorizzato l'instante in cui parte il gioco.

```
79  setLcdText("Hai selezionato la", "tabellina del " + String(boards), "",
    "BATTAK 2.0");
80  long startTime = millis();
```

A seguire troviamo il ciclo principale dove vengono calcolate le tabelline.
Vengono proposte 11 tabelline prima che il programma venga interrotto. Il ciclo può anche venire interrotto dal comando apposito.

```
81  for (int i = 0; i < 12 && programRunning; i++){
```

Le prime cose che troviamo in questo ciclo son oil reset dello stato di tutti I led, seguito dalla dichiarazione dei 3 parametri della moltiplicazione: la base (valore scelto in precedenza), il moltiplicatore (estratto ogni volta casualmente all'interno di un intervallo che va da 0 a 9 inclusi) e in fine il risultato.
Viene dunque riportato un messaggio sul display LCD e rilevato l'istante di avvio del gioco. Viene nuovamente impostata a *true* la variabile che determina che lo stato del gioco corrisponde ad un ô stato di attesa.

```
83  resetLeds();
84  int one = boards;
85  int two = random(0, 9);
86  int sum = one * two;
87
88  setLcdText("Tabellina del " + String(boards), String(one) + " * " +
89  String(two) + " = ?", "", "BATTAK 2.0");
90
91  startTime = millis();
92  waiting = true;
```

Questo nuovo ciclo di attesa è identico al primo, cambia solo per le azioni che vengono svolte al suo interno: il ciclo non si interrompe fino a quando non viene scelta una tabellina valida o se viene registrato il comando di interruzione.

```
92  while (waiting && programRunning){
```

All'interno del ciclo troviamo un nuovo ciclo che scorre tutti i pulsanti del sistema, ma prima di entrare in questo ciclo, viene inviato al secondo Arduino il tempo a disposizione del giocatore per risolvere la tabellina.

```
94  sendData(1, (millis() - startTime) / 100);
95
96  for (int x = 0; x < SIZE; x++){
```

Come detto sopra, questo ciclo scorre tutti i pulsanti del telaio. Per ognuno di questi verifica se è premuto e se quello premuto corrisponde al risultato della tabellina, questo pulsante viene acceso e viene invertito lo stato del programma. Viene anche interrotto il ciclo corrente.
Se invece il valore ottenuto non corrisponde al risultato corretto, viene verificato se il valore corrisponde al risultato diviso 10. In questo caso, viene acceso il pulsante indicato e la somma viene aggiornata con sé stessa meno il valore dell'utente moltiplicato per 10. Questa procedura serve a poter avere anche tabelline con decine e unità (ossia 2 cifre).

```
96  for (int x = 0; x < SIZE; x++)
97  {
98    int bpin = buttonPins[x];
99    bool lastState = getLastState(bpin);
100   bool currentState = isPressed(bpin);
101   if (currentState == HIGH && currentState != lastState)
102   {
```

```
103       if (getLabel(bpin) == sum)
104       {
105         digitalWrite(bpin + 1, HIGH);
106         waiting = false;
107         break;
108       }
109       else
110       {
111         if (getLabel(bpin) == sum / 10)
112         {
113           digitalWrite(bpin + 1, HIGH);
114           sum = sum - getLabel(bpin) * 10;
115         }
116       }
117     }
118   }
```

Esempio di utilizzo:

```
XXX   boards();
```

3.11 Stretching angolare, 25 pulsanti

(Vedi capitolo 3.4)

Esempio di utilizzo:

```
XXX angularStretching(25, 11);
```

3.12 Stretching angolare, 50 pulsanti

(Vedi capitolo 3.11)

Esempio di utilizzo:

```
XXX angularStretching(50, 12);
```

3.13 25 Pulsanti temporizzati, 1 secondo per pulsante, junior

(Vedi capitolo 3.7)

Esempio di utilizzo:

```
XXX temporized(25, false);
```

3.14 50 Pulsanti temporizzati, 1 secondo per pulsante, junior

(Vedi capitolo 3.7)

Esempio di utilizzo:

```
XXX temporized(50, false);
```

3.15 Cumulativo 30 secondi, junior

(Vedi capitolo 3.1)

Esempio di utilizzo:

```
XXX cumulative(30000, false);
```

3.16 Cumulativo 60 secondi, junior

(Vedi capitolo 3.1)

Esempio di utilizzo:

```
XXX cumulative(60000, false);
```

3.17 Corsa 25 pulsanti, junior

(Vedi capitolo 3.3)

Esempio di utilizzo:

XXX	`rush(50000, 25, false);`

3.18 Corsa 50 pulsanti, junior

(Vedi capitolo 3.3)

Esempio di utilizzo:

XXX	`rush(100000, 50, false);`

3.19 Maratona (3 minuti) junior

(Vedi capitolo 3.1)

Esempio di utilizzo:

XXX	`cumulative(180000, false);`

3.20 Semplice gioco Simon: 20 pulsanti, 17 livelli

Il programma somma matematica viene eseguito dal metodo **simon** che non richiede nessun parametro.

```
367 void simon() {
```

Dopo ciò vengono inizializzate le variabili ***buttons*** e ***sequence[ ]***.

```
368 int buttons = 20;
369 int sequence[buttons];
```

Successivamente viene create la sequenza iniziale, che verrà ripetuta in ogni livello, aggiungendo ogni volta un pulsante.

```
370 for (int i = 0; i < buttons; i++)
371 {
372   int rndpin = getRandom(buttonPins, SIZE);
373   sequence[i] = rndpin;
373 }
```

Il programma viene eseguito per un massimo di 17 livelli (volte). Se l'utente sbaglia a ripetere una sequenza la modalità termina.

```
381 for (int i = 0; i < 17 && playing && programRunning; i++)
```

Prima di tutto il gioco propone la sequenza che il sistema ha creato in precedenza, accendendo un led alla volta per il tempo di 0.1 secondi.

```
385 for (int x = 0; x < toShow; x++)
386 {
387   for (int y = 0; y < 3; y++)
388   {
389     digitalWrite(sequence[x] + 1, HIGH);
390     delay(100);
391     digitalWrite(sequence[x] + 1, LOW);
392     delay(100);
393   }
394   delay(500);
395 }
```

Quando il sistema ha mostrato tutta la sequenza viene emesso un segnale vocale ("beep"), che segnala a l'utente che la sequenza è terminata.

```
396 tone(buzzerPin, 2000);
397 delay(250);
398 noTone(buzzerPin);
399 delay(100);
```

```
400 tone(buzzerPin, 2000);
401 delay(250);
402 noTone(buzzerPin);
403 setLcdText("Ripeti la sequenza", "", "", "BATTAK 2.0");
404 resetButtonsState();
405 pressed = 0;
406 toShow += 1;
407 insideLevel = true;
```

Successivamente viene effettutato un ciclo while che termina quando l'utente ha completato con successo il livello. Se preme un pulsante sbagliato il ciclo viene interrotto e la modalità termina.

```
408 while (insideLevel && playing && programRunning)
```

Infine viene controllato che ogni bottone che l'utente preme appartenga effettivamente alla sequenza prodotta dal sistema. Se il pulsante cliccato non corrisponde a quello che dovrebbe venire premuto alla variabile booleana presente nel while precedente ***playing*** viene assegnato il valore **false** e il ciclo viene interrotto.

```
410 for (int b = 0; b < SIZE && playing && insideLevel && programRunning; b++)
411 {
412   int pin = buttonPins[b];
413   bool presslastState = getLastState(pin);
414   bool presscurrentState = isPressed(pin);
415   if (presslastState == HIGH && presslastState != presscurrentState)
416   {
417     if (pin == sequence[pressed])
418     {
419       digitalWrite(pin + 1, HIGH);
420       pressed += 1;
421       points += 1;
422       sendData(0, points);
423       delay(150);
424       if (pressed == toShow - 1)
425       {
426         insideLevel = false;
427         }
428       }
429       else
430       {
431         playing = false;
432         setLcdText("Hai sbagliato a", "ripetere la sequenza", "", "BATTAK
433 2.0");
434         delay(1000);
435       }
436     }
437 }
```

Esempio di utilizzo:

XXX	`simon();`

3.21 Flash test, 5 schemi

Il programma somma matematica viene eseguito dal metodo **flashTest** che richiede come unico parametro se il flash test deve essere eseguito sui pulsanti che si accendono ("**false**") o su quelli spenti ("**true**").

```
228 void flashtest(bool onButtons) {
```

Nella prima parte di codice vengono resettati tutti i led (spenti se ***onButtons*** è **false** oppure accesi se è **true**), inizializzate le variabili ***waiting***, ***selectedTime***. Viene inoltre stampata a display la stringa "Scegli un tempo per spegnere tutti i pulsanti (1-8)".

```
229 resetLeds();
230 setLcdText("Scegli un tempo per", "spegnere tutti i", "pulsanti (1-8)",
    "BATTAK 2.0");
231 bool waiting = true;
232 int selectedtime = 0;
```

Successivamente il sistema aspetta che venga inserito un valore da 1 a 8 secondi, che rappresenterà il tempo massimo che l'utente ha a disposizione per spegnere i pulsanti che sono accesi. Se l'utente inserisce un valore non consentito il sistema richiede il valore affinché sia valido.

```
233 while (waiting && programRunning)
234 {
235   for (int i = 1; i <= SIZE; i++)
236   {
237     int bpin = buttonPins[i];
238     bool lastState = getLastState(bpin);
239     bool currentState = isPressed(bpin);
240     if (currentState == HIGH && currentState != lastState)
241     {
242       digitalWrite(bpin + 1, HIGH);
243       int label = getLabel(bpin);
244       if (label >= 1 && label <= 8)
245       {
246         selectedtime = label * 1000;
247         waiting = false;
248       }
249     }
250   }
251 }
```

Successivamente vengono inizializzate le variabili ***point*** e ***playing***, oltre a resettare lo stato di tutti i pulsanti e stampare sul display la stringa contenente il numero di secondi che l'utente ha selezionato in precedenza.

```
252 setLcdText("Hai scelto: " + String(selectedtime / 1000) + " s", "", "",
    "BATTAK 2.0");
253 resetLeds();
254 int points = 0;
255 bool playing = true;
256 delay(200);
```

Il primo ciclo for serve a generare gli schemi e controlla che un pulsante non venga ripetuto.

```
257 for (int i = 0; i < 5 && playing && programRunning; i++)
258   {
259     resetLeds();
260     sendData(1, i + 1);
261     setLcdText("Schema: " + String(i + 1) + "/5", "", "", "BATTAK 2.0");
262     int sequence[6];
263     if (onButtons)
264     {
265       for (int i = 0; i < SIZE; i++)
266       {
267         digitalWrite(buttonPins[i] + 1, HIGH);
268       }
269     }
270     for (int i = 0; i < 6; i++)
271     {
272       bool random = true;
273       int rndpin = 0;
274       while (random)
275       {
276         rndpin = getRandom(buttonPins, SIZE);
277         bool contain = false;
278         for (int j = 0; j < 6; j++)
279         {
280           if (sequence[j] == rndpin) {
281             contain = true;
282           }
283         }
284         if (!contain)
285         {
286           break;
287         }
288       }
289       sequence[i] = rndpin;
290       if (onButtons)
291       {
292         digitalWrite(rndpin + 1, LOW);
293       }
294       else
295       {
```

```
296           digitalWrite(rndpin + 1, HIGH);
297         }
298       }
299
302   while (insideLevel && programRunning)
303   {
304     if (millis() - startTime >= selectedtime)
305     {
306     playing = false;
307       break;
308     }
309     for (int b = 0; b < SIZE && insideLevel && playing && programRunning;
310   b++)
311     {
312       int pin = buttonPins[b];
313       bool presslastState = getLastState(pin);
314       bool presscurrentState = isPressed(pin);
315       if (presslastState == HIGH && presslastState != presscurrentState)
          {
316         bool edited = false;
317         for (int x = 0; x < 6; x++) {
            if (sequence[x] == pin) {
318           edited = true;
319           if (onButtons)
320           {
321             digitalWrite(pin + 1, HIGH);
322           }
323           else
324           {
325             digitalWrite(pin + 1, LOW);
326           }
327           pressed += 1;
328           points += 1;
329           sendData(0, points);
330           if (pressed == 6)
331           {
332             insideLevel = false;
333           }
334           sequence[x] = -1;
335         }
336       }
337       if (!edited)
338       {
339         playing = false;
340         insideLevel = false;
341         setLcdText("Hai sbagliato a", "premere un pulsante!", "", "BATTAK
342   2.0");
343         delay(1000);
```

```
344         }
345         delay(150);
346       }
347     }
348   }
```

La seconda parte del ciclo for è composta da un ciclo while in cui vengono effettuti tutti i controlli per verificare che l'utente abbia cliccato effettivamente i pulsanti corretti.

```
while (insideLevel && programRunning)
{
  if (millis() - startTime >= selectedtime)
  {
  playing = false;
    break;
  }
  for(int b = 0; b < SIZE && insideLevel && playing && programRunning;b++)
  {
    int pin = buttonPins[b];
    bool presslastState = getLastState(pin);
    bool presscurrentState = isPressed(pin);
    if (presslastState == HIGH && presslastState != presscurrentState)
    {
      bool edited = false;
      for (int x = 0; x < 6; x++) {
        if (sequence[x] == pin) {
          edited = true;
          if (onButtons)
          {
            digitalWrite(pin + 1, HIGH);
          }
          else
          {
            digitalWrite(pin + 1, LOW);
          }
          pressed += 1;
          points += 1;
          sendData(0, points);
          if (pressed == 6)
          {
            insideLevel = false;
          }
          sequence[x] = -1;
        }
      }
      if (!edited)
      {
        playing = false;
        insideLevel = false;
        setLcdText("Hai sbagliato a", "premere un pulsante!", "", "BATTAK
2.0");
        delay(1000);
      }
      delay(150);
    }
```

```
    }
}
```

Esempio di utilizzo:

```
XXX  flashtest(false);
```

3.22 Anti - flash test

(Vedi capitolo 3.21)

Esempio di utilizzo:

```
XXX  flashtest(true);
```

3.23 Reazione veloce, 10 schemi

Il programma viene eseguito dal metodo *fastreaction*.

```
131 void fastreaction() {
```

La prima variabile che troviamo è *schemes*, si tratta di una variabile di configurazione che definisce da quanti schemi sarà composto il gioco.
In seguito troviamo 2 metodi che rispettivamente, uno spegne tutti i led, l'altro riporta sull'LCD il testo indicato.
La variabile booleana definisce lo stato del gioco (in attesa o in esecuzione), mentre la variabile *button* definisce il numero di pulsanti già premuti.

```
132 int schemes = 10;
133 resetLeds();
134 setLcdText("Seleziona un numero", "di pulsanti per", "schema (Da 1 a 11)",
    "BATTAK 2.0");
135 bool waiting = true;
136 int buttons = 0;
```

Ora troviamo un ciclo che viene interrotto solo se viene lo stato del gioco non è in "*waiting"* oppure se riceve il messaggio di interruzione.

```
137 while (waiting && programRunning)
```

Al suo interno troviamo un nuovo ciclo che verifica quale pulsante è stato premuto e se è valido (intervallo compreso fra 1 e 11) esce dal ciclo esterno impostando il valore di *waiting* a false.

```
139 for (int i = 1; i <= SIZE; i++)
140 {
141   int bpin = buttonPins[i];
142   bool lastState = getLastState(bpin);
143   bool currentState = isPressed(bpin);
144   if (currentState == HIGH && currentState != lastState)
145   {
146     digitalWrite(bpin + 1, HIGH);
147     int label = getLabel(bpin);
148     if (label >= 1 && label <= 11)
149     {
150       buttons = label;
151       waiting = false;
152     }
153   }
154 }
```

Vengono istanziate 3 nuove variabili: una che definisce l'array di pulsanti che dovranno venir premuti; una che salva l'istante in cui viene eseguito il programma; e l'ultima che serve a salvare il punteggio

```
156 int buttonseq[buttons];
157 long startTime = millis();
158 int points = 0;
```

Ora entriamo in un ciclo che percorre tutti gli schemi del gioco eccezion fatta se registra il comando di *"quit"*.

```
159 for (int i = 0; i < schemes && programRunning; i++)
```

A questo punto vengono resettati nuovamente tutti i LED e scritto un messaggio per l'utente sull'LCD.
Entrando in questo ciclo, viene popolato l'array di pulsanti citato prima tramite un'estrazione randomica.

```
161 resetLeds();
162 setLcdText("Schema: " + String(i + 1) + "/" + String(schemes), "Pulsanti:
    " + String(buttons), "", "BATTAK 2.0");
163 for (int a = 0; a < buttons; a++)
164 {
165   bool random = true;
166   int rndpin = 0;
167   while (random)
168   {
169     rndpin = getRandom(buttonPins, SIZE);
170     bool contain = false;
171     for (int j = 0; j < 6; j++)
172     {
173       if (buttonseq[j] == rndpin) {
174         contain = true;
175       }
176     }
177     if (!contain)
178     {
179       break;
180     }
181   }
182   buttonseq[a] = rndpin;
183   digitalWrite(rndpin + 1, HIGH);
184 }
```

Ancora una volta resettiamo lo stato dei pulsanti, impostiamo a 0 la variabile che definisce quanti pulsati sono stati premuti e dichiariamo una nuova variabile che definisce lo stato del livello (dentro o fuori da esso).

```
185 resetButtonsState();
186 int pressed = 0;
187 bool insideLevel = true;
```

Ora entriamo nel ciclo del livello.
Innanzi tutto viene inviato al display 7 segmenti il timer entro il quale devono essere premuti tutti i pulsanti, di seguito entriamo nel ciclo che verifica quali pulsanti sono stati premuti e se corretti, incrementa il punteggio.
Entrambi i cicli possono venire interrotti, prima del loro completamento, dal comando di *"quit"*.

```
188 while (insideLevel && programRunning)
189 {
190   sendData(0, (millis() - startTime) / 100);
191   for (int b = 0; b < SIZE && insideLevel && programRunning; b++)
192   {
193     int pin = buttonPins[b];
```

```
194     bool presslastState = getLastState(pin);
195     bool presscurrentState = isPressed(pin);
196     if (presslastState == HIGH && presslastState != presscurrentState)
197     {
198       for (int x = 0; x < buttons; x++) {
199         if (buttonseq[x] == pin) {
200           digitalWrite(pin + 1, LOW);
201           pressed += 1;
202           points += 1;
203           sendData(1, points);
204           if (pressed == buttons)
205           {
206             insideLevel = false;
207           }
208           buttonseq[x] = -1;
209         }
210       }
211     }
212   }
213 }
```

Esempio di utilizzo:

```
XXX   fastreaction();
```

3.24 Errori e problemi da gestire

La procedura di check è molto semplice. Prima di tutto bisogna premere il programma “99” nel menu del sistema. Prima di tutto viene emesso un beep per 1 secondo, dopodiché scorre tutti i pulsanti del sistema accendendone uno alla volta, quando si accende un bottone viene stampato il pulsante da premere e quando viene cliccato dall’utente appare a display la stringa “Il pulsante dovrebbe essere acceso”.

```
24  for (int i = 0; i < SIZE && programRunning; i++)
25  {
26    int pin = buttonPins[i];
27    String text = String(getLabel(pin));
28    if (text == "10")
29    {
30      text = "#";
31    }
32    else if (text == "11")
33    {
34      text = "@";
35    }
36    setLcdText("Premi il pulsante", text, "", "BATTAK 2.0");
37    while (!isPressed(pin) && programRunning) {
38      isPressed(44);
39    }
40    digitalWrite(pin + 1, HIGH);
41    setLcdText("Il pulsante dovrebbe", "essere accesso", "", "BATTAK 2.0");
42    delay(1000);
}
```

Esempio di utilizzo:

```
XXX  systemCheck();
```

4 Test

4.1 Protocollo di test

Test Case: Riferimento:	TC-001 REQ-01	Nome:	Cumulativo 60 secondi, senior
Descrizione:	Programma che svolge il programma cumulativo (senior)		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘1’ e ‘#’. 3. Premere i bottoni che si illuminano fino a quando il timer si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, quando il timer si ferma (dopo 60 secondi) viene stampato il numero di bottoni cliccati.		

Test Case: Riferimento:	TC-002 REQ-02	Nome:	Maratona (5 minuti) senior
Descrizione:	Per 300 secondi si accendono pulsanti in modo casuale, quando viene premuto il pulsante acceso, si spegne e viene incrementato il punteggio.		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘2’ e ‘#’. 3. Premere i bottoni che si illuminano fino a quando il timer si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, quando il timer si ferma (dopo 300 secondi) viene stampato il numero di bottoni cliccati		

Test Case:	TC-003	Nome:	Corsa 50 pulsanti, senior
Riferimento:	REQ-03		
Descrizione:	Per 100 secondi si accendono pulsanti in modo casuale, quando viene premuto il pulsante acceso, si spegne e viene incrementato il punteggio. Se l'utente preme correttamente 50 pulsanti il programma termina.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '3' e '#'. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, quando il timer si ferma (dopo 100 secondi o 50 pulsanti) viene stampato il numero di bottoni cliccati correttamente o il tempo con cui si ha cliccato i 50 bottoni.		

Test Case:	TC-004	Nome:	Stretching angolare, 100 pulsanti, senior
Riferimento:	REQ-04		
Descrizione:	Si accendono 100 bottoni in modo casuale, se l'utente sbaglia a cliccare il bottone la velocità di accensione dei bottoni aumenta. Si accendono solo i bottoni agli estremi del telaio. Inizialmente i bottoni rimangono accesi per 1 secondo.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '4' e '#'. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il tempo trascorso e il numero di bottoni premuti correttamente. Se l'utente preme il bottone sbagliato, la velocità di accensione aumenta. Dopo 100 bottoni cliccati il programma termina. Si accendono solo i bottoni agli estremi del telaio.		

Test Case:	TC-005	Nome:	Tiro a vuoto – pulsanti 1 secondo, senior
Riferimento:	REQ-05		
Descrizione:	Si accendono 100 bottoni in modo casuale, se l'utente sbaglia a cliccare il bottone la velocità di accensione dei bottoni aumenta (alcuni bottoni simulano l'accensione). Inizialmente i bottoni rimangono accesi per 1 secondo. Se si illuminano i tre pulsanti centrali l'utente deve interrompere una barriera luminosa o premere una pedana.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '5' e '#'. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il tempo trascorso e il numero di bottoni premuti correttamente. Se l'utente preme il bottone sbagliato (alcuni pulsanti simulano l'accensione) la velocità di accensione aumenta e vengono sottratti 5 punti. Dopo 100 bottoni cliccati il programma termina.		

Test Case:	TC-006	Nome:	Test di Léger o Beep test, senior
Riferimento:	REQ-06		
Descrizione:	Il programma possiede 10 schemi (livelli) con frequenza variabile da 1.4 secondi per il primo livello a 0.5 secondi per il decimo livello. Vengono mostrati i livelli su display. Se l'utente sbaglia consecutivamente 3 volte la sequenza viene interrotta. Al primo errore viene emesso un "beep", al secondo e al terzo tre. Se dopo uno o due errori l'utente effettua un colpo corretto il conteggio degli errori viene azzerato.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '6' e '#'. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, ci sono 10 livelli e crescendo sia ha meno tempo per premere i pulsanti. Se l'utente sbaglia 3 volte consecutive il programma termina. Quando si sbaglia viene emesso un "beep" che corrisponde al numero di errori consecutivi effettuati.		

Test Case: TC-007 Riferimento: REQ-07	Nome:	50 pulsanti temporizzati, 1 secondo per pulsante, senior
Descrizione:	Il sistema accende in successione casualmente 50 bottoni. Sul display viene stampato il conto alla rovescia dei 50 pulsanti. L'utente ha a disposizione 1 secondo per premere il pulsante attivato. La velocità aumenta se viene cliccato il bottone sbagliato.	
Prerequisiti:	Sistema hardware "Batak" funzionante	
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '7' e '#'. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.	
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il numero di pulsanti rimanenti e il numero di bottoni cliccati in modo corretto. Se l'utente preme il bottone sbagliato (alcuni pulsanti simulano l'accensione) la velocità di accensione aumenta. Il programma termina dopo l'accensione di 50 pulsanti.	

Test Case: TC-008 Riferimento: REQ-08	Nome:	Staffetta 4 giocatori 120 secondi (30 secondi a testa)
Descrizione:	4 giocatori si sfidano in una staffetta, ogni giocatore ha 30 secondi a disposizione per premere il maggior numero di pulsanti. Vince chi preme il maggior numero di bottoni in modo corretto.	
Prerequisiti:	Sistema hardware "Batak" funzionante	
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '8' e '#'. 3. Premere i bottoni che si illuminano fino a quando scade il tempo.	
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il tempo e il punteggio di ogni giocatore.	

Test Case: Riferimento:	TC-009 REQ-09	Nome:	Reazione somma aritmetica
Descrizione:	Questo programma presenta all'utente 8 semplici addizioni, al quale l'utente deve rispondere indicando il risultato. Prima che il sistema propone le addizione chiede all'utente il numero di secondi (da 2 a 9) che vuole avere a disposizione per rispondere.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '9' e '#'. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, l'utente può scegliere l'intervallo di tempo in secondi (da 2 a 9), vengono poste 8 addizioni. Alla fine del programma viene stampato il numero di addizioni corrette.		

Test Case: Riferimento:	TC-010 REQ-10	Nome:	Tabelline di velocità
Descrizione:			
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '1' e "0". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:			

Test Case:	TC-011	Nome:	Stretching angolare, 25 pulsanti
Riferimento:	REQ-11		
Descrizione:	Si accendono in ordine sequenziale 25 pulsanti situati agli estremi del telaio, ogni pulsante rimane acceso per 1 secondo. Vengono mostrati i pulsanti rimanenti e i colpi corretti effettuati. Se viene premuto il pulsante sbagliato la sequenza aumenta la velocità.		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘1’ e “1.” 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il numero di bottoni rimanenti e il numero di colpi corretti. Si accendono solo i bottoni agli estremi del telaio, se si preme il bottone sbagliato la sequenza aumenta di velocità. Dopo l’accensione di 25 bottoni il programma termina.		

Test Case:	TC-012	Nome:	Stretching angolare, 50 pulsanti
Riferimento:	REQ-12		
Descrizione:	Si accendono in ordine sequenziale 50 pulsanti situati agli estremi del telaio, ogni pulsante rimane acceso per 1 secondo. Vengono mostrati i pulsanti rimanenti e i colpi corretti effettuati. Se viene premuto il pulsante sbagliato la sequenza aumenta la velocità.		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘1’ e “2”. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il numero di bottoni rimanenti e il numero di colpi corretti. Si accendono solo i bottoni agli estremi del telaio, se si preme il bottone sbagliato la sequenza aumenta di velocità. Dopo l’accensione di 25 bottoni il programma termina.		

Test Case: TC-013 Riferimento: REQ-13	Nome:	25 pulsanti temporizzati, 1 secondo per pulsante, junior
Descrizione:	Si accendono 25 pulsanti casualmente, si accendono solo i bottoni inferiori al numero 5, ossia i numeri sotto la metà del telaio. L'utente ha a disposizione 1 secondo per premere il pulsante.	
Prerequisiti:	Sistema hardware "Batak" funzionante	
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '1' e "3". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.	
Risultati attesi:	Il programma funziona in modo corretto, si accendono 25 pulsanti, solo quelli sotto la metà del telaio, in ordine casuale. Sul display vengono stampanti i pulsanti premuti correttamente e i pulsanti rimanenti.	

Test Case: TC-014 Riferimento: REQ-14	Nome:	50 pulsanti temporizzati, 1 secondo per pulsante, junior
Descrizione:	Si accendono 50 pulsanti casualmente, si accendono solo i bottoni inferiori al numero 5, ossia i numeri sotto la metà del telaio. L'utente ha a disposizione 1 secondo per premere il pulsante.	
Prerequisiti:	Sistema hardware "Batak" funzionante	
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '1' e "4". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.	
Risultati attesi:	Il programma funziona in modo corretto, si accendono 50 pulsanti, solo quelli sotto la metà del telaio, in ordine casuale. Sul display vengono stampanti i pulsanti premuti correttamente e i pulsanti rimanenti.	

Test Case: TC-015	Nome:	Cumulativo 30 secondi, junior
Riferimento: REQ-15		
Descrizione:	Il programma dura 30 secondi, vengono accesi in ordine casuale i pulsanti inferiori alla metà del telaio. Avviene un accensione sequenziale. Viene stampato il punteggio e il tempo rimasto	
Prerequisiti:	Sistema hardware “Batak” funzionante	
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘1’ e “5”. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.	
Risultati attesi:	Il programma funziona in modo corretto, vengono accesi solo i bottoni compresi nella metà inferiore del telaio. Viene stampato il tempo rimanente e il punteggio.	

Test Case: TC-016	Nome:	Cumulativo 60 secondi, junior
Riferimento: REQ-16		
Descrizione:	Il programma dura 60 secondi, vengono accesi in ordine casuale i pulsanti inferiori alla metà del telaio. Avviene un accensione sequenziale. Viene stampato il punteggio e il tempo rimasto	
Prerequisiti:	Sistema hardware “Batak” funzionante	
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘1’ e “6”. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.	
Risultati attesi:	Il programma funziona in modo corretto, vengono accesi solo i bottoni compresi nella metà inferiore del telaio. Viene stampato il tempo rimanente e il punteggio.	

Test Case:	TC-017	Nome:	Corsa 25 pulsanti, junior
Riferimento:	REQ-17		
Descrizione:	Il programma dura 25 pulsanti oppure 50 secondi. Avviene un'accensione casuale dei pulsanti nella metà inferiore del telaio, viene stampato il numero di pulsanti premuti in modo corretto e il tempo rimanente.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '1' e "7". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona correttamente, accendendo solo i pulsanti compresi nella metà del telaio inferiore. Viene stampato il numero di pulsanti premuti in modo corretto e il tempo rimanente. Il programma termina dopo 50 secondi o 25 pulsanti premuti correttamente.		

Test Case:	TC-018	Nome:	Corsa 50 pulsanti, junior
Riferimento:	REQ-18		
Descrizione:	Il programma dura 50 pulsanti oppure 100 secondi. Avviene un'accensione casuale dei pulsanti nella metà inferiore del telaio, viene stampato il numero di pulsanti premuti in modo corretto e il tempo rimanente.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '1' e "8". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona correttamente, accendendo solo i pulsanti compresi nella metà del telaio inferiore. Viene stampato il numero di pulsanti premuti in modo corretto e il tempo rimanente. Il programma termina dopo 100 secondi o 50 pulsanti premuti correttamente.		

Test Case:	TC-019	Nome:	Maratona (3 minuti) junior
Riferimento:	REQ-19		
Descrizione:	Il programma dura per 180 secondi ed a display viene stampato il punteggio e il tempo rimanente. I pulsanti inferiori alla metà del telaio, si accendono in ordine casuale.		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘1’ e “9”. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, per 180 secondi vengono accesi in modo casuale i pulsanti compresi nella parte inferiore del telaio. Viene stampato a display il numero di colpi corretti e il tempo rimanente.		

Test Case:	TC-020	Nome:	Semplice gioco Simon: 20 pulsanti, 17 livelli
Riferimento:	REQ-20		
Descrizione:	Il programma accende inizialmente, in ordine casuale, 4 bottoni che successivamente l’utente dovrà riprodurre. Ogni volta che l’utente riproduce in modo corretto la sequenza, il livello di difficoltà aumenta aggiungendo al livello successivo un pulsante. Il programma termina se l’utente commette un errore o completa tutti i 17 livelli.		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni ‘@’, ‘2’ e “0”. 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona correttamente, ogni volta che l’utente riproduce in modo corretto la sequenza, il numero dei pulsanti aumenta. Il programma termina quando l’utente sbaglia o completa tutti i livelli.		

Test Case:	TC-021	Nome:	Flash test, 5 schemi
Riferimento:	REQ-21		
Descrizione:	Il sistema accende fino a 6 pulsanti, per 5 schemi. L'utente inizialmente sceglie la durata dell'accensione dei pulsanti, grazie al carattere "@" più i secondi (da 1 a 8). L'utente deve premere in ordine casuale gli stessi pulsanti accesi precedentemente dal sistema. Quando il tempo termina il sistema emette 2 "Beep". Ogni schema completato incrementa il punteggio, sui display vengono stampati il punteggio raggiunto e il punteggio massimo possibile.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '2' e "1". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona correttamente, l'utente può scegliere il tempo a disposizione per riprodurre, in ordine casuale, lo schema proposto dal sistema. Viene emesso un doppio "Beep" quando il programma termina. Viene stampato il punteggio ottenuto e il punteggio massimo possibile.		

Test Case:	TC-022	Nome:	Anti - flash test
Riferimento:	REQ-22		
Descrizione:	Il sistema accende fino a 6 pulsanti, per 5 schemi. L'utente inizialmente sceglie la durata dell'accensione dei pulsanti, grazie al carattere "@" più i secondi (da 1 a 8). L'utente deve premere in ordine casuale i pulsanti che non sono stati accesi precedentemente dal sistema. Quando il tempo termina il sistema emette 2 "Beep". Ogni schema completato incrementa il punteggio, sui display vengono stampati il punteggio raggiunto e il punteggio massimo possibile.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '2' e "2". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona correttamente, l'utente può scegliere il tempo a disposizione per riprodurre, in ordine casuale, lo schema dei pulsanti che non sono stati accesi. Viene emesso un doppio "Beep" quando il programma termina. Viene stampato il punteggio ottenuto e il punteggio massimo possibile.		

Test Case:	TC-023	Nome:	Reazione veloce, 10 schemi
Riferimento:	REQ-23		
Descrizione:	Il programma dura per 10 schemi, l'utente seleziona i pulsanti che vuole utilizzare dopo aver premuto il bottone "#". I pulsanti vengono accesi in ordine casuale in modo sequenziale. La velocità del gioco è determinata dal giocatore. Vengono stampati sui display il tempo e il numero dei pulsanti premuti in modo corretto.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '2' e "3". 3. Premere i bottoni che si illuminano fino a quando il sistema si ferma.		
Risultati attesi:	Il programma funziona in modo corretto, l'utente decide quali pulsanti utilizzare. I pulsanti vengono accesi casualmente in modo sequenziale, il giocatore determina la velocità di gioco. Sui display vengono stampati il tempo e i pulsanti premuto in modo corretto.		

Test Case:	TC-024	Nome:	Errori e problemi da gestire
Riferimento:	REQ-24		
Descrizione:	È necessario un sistema di gestione errori (pulsanti incastrati, contatti problematici, l rimangono costantemente accesi). Vi è una procedura di check (all'accensione del s oppure su richiesta). Il sistema di gestione errori li individua e li segnala.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	1. Accendere il sistema 2. Cliccare la combinazione di bottoni '@', '9' e "9". 3. Aspettare che il sistema termini la procedura di check		
Risultati attesi:	Il sistema effettua la procedura di check, vengono accesi i led e controllato corretto funzionamento.		

4.2 Risultati test

In seguito è mostrata la tabella contenente il codice del protocollo di test, il risultato del test e, se vi sono, delle note per documentare il risultato del test.

Codice Test	Risultato	Note
TC – 001	Passato	Nessuna
TC – 002	Passato	Nessuna
TC – 003	Passato	Nessuna
TC – 004	Passato	Nessuna
TC – 005	-	Limitato (vedi capitolo 4.3 Limitazioni e mancanze)
TC – 006	Passato	Nessuna
TC – 007	Passato	Nessuna
TC – 008	Passato	Nessuna
TC – 009	Passato	Nessuna
TC – 010	Passato	Nessuna
TC – 011	Passato	Nessuna
TC – 012	Passato	Nessuna
TC – 013	Passato	Nessuna
TC – 014	Passato	Nessuna
TC – 015	Passato	Nessuna
TC – 016	Passato	Nessuna
TC – 017	Passato	Nessuna
TC – 018	Passato	Nessuna
TC – 019	Passato	Nessuna
TC – 020	Passato	Nessuna
TC – 021	Passato	Nessuna
TC – 022	Passato	Nessuna
TC – 023	Passato	Nessuna
TC – 024	Passato	Nessuna

4.3 Mancanze/limitazioni conosciute

Il nostro progetto presenta la mancanza del programma descritto nel requisito numero 5 (REQ-05). Non abbiamo potuto implementare questo programma a causa del mancato sistema di barriera luminosa da interrompere (vedi Figura 6) o pedana (vedi Figura 7).

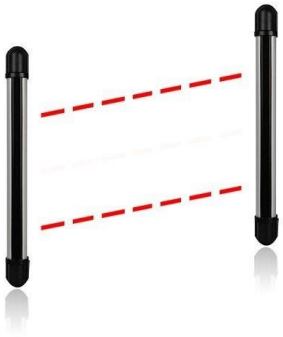


Figura 5 Esempio di barriera luminosa

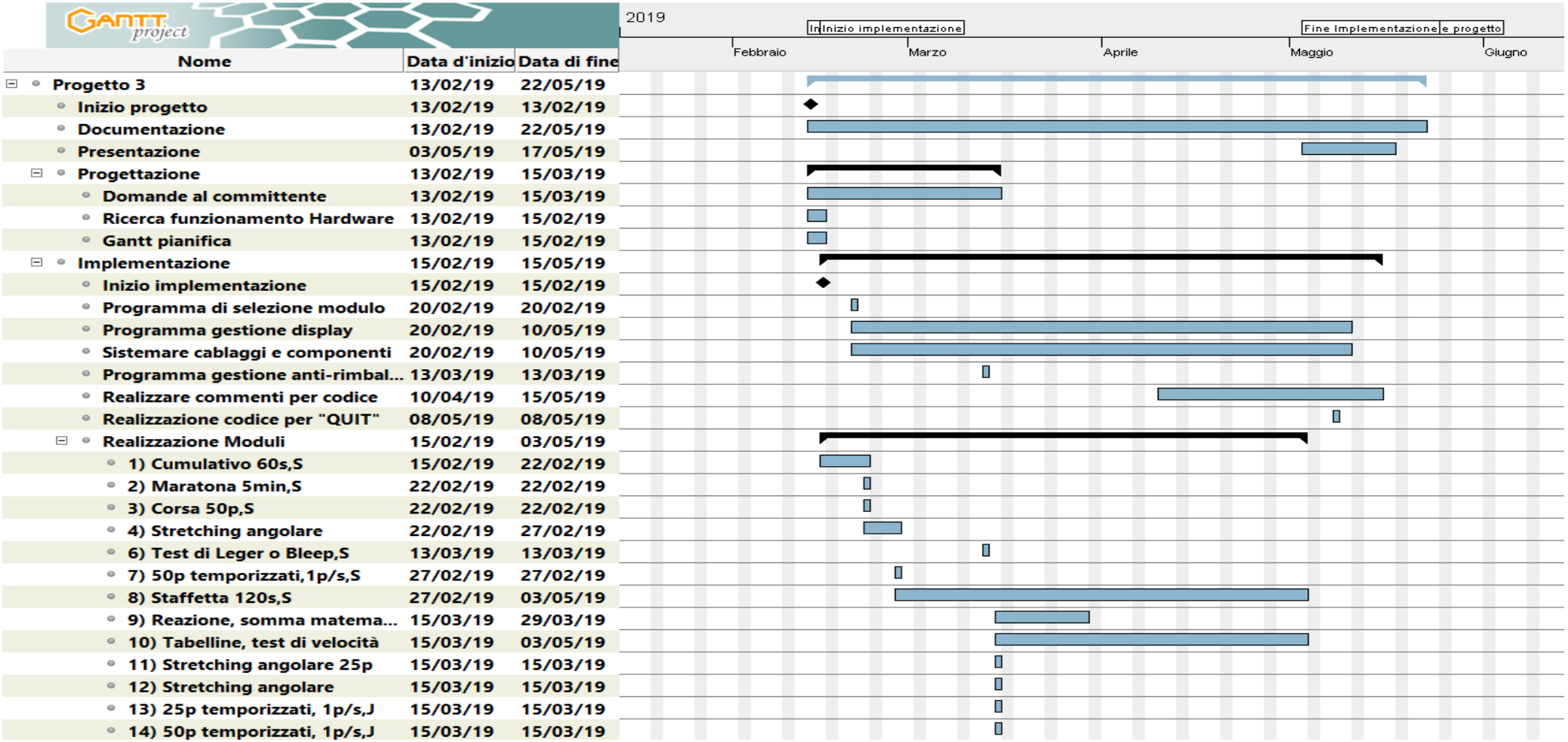
Figura 6 Esempio di pedana

Dopo aver richiesto al docente responsabile A. Barchi, ci è stato riferito di non implementare il programma all'interno del sistema.

Una seconda limitazione conosciuta è la scarsa illuminazione dei led all'interno dei pulsanti posti sul telaio che in presenza di forte illuminazione esterna si fatica a vedere i pulsanti accesi.

La terza limitazione che il nostro sistema possiede è la mancanza dell'assistente vocale che è presente in alcuni programmi (ev. RQ – 05, RQ – 10). Abbiamo parzialmente ovviato a questa limitazione rappresentando ciò che avrebbe dovuto venir detto dall'assistente vocale sul display LCD.

5 Consuntivo



Nome	Data d'inizio	Data di fine
Progetto 3	13/02/19	22/05/19
Inizio progetto	13/02/19	13/02/19
Documentazione	13/02/19	22/05/19
Presentazione	03/05/19	17/05/19
Progettazione	13/02/19	15/03/19
Domande al committente	13/02/19	15/03/19
Ricerca funzionamento Hardware	13/02/19	15/02/19
Gantt pianifica	13/02/19	15/02/19
Implementazione	15/02/19	15/05/19
Inizio implementazione	15/02/19	15/02/19
Programma di selezione modulo	20/02/19	20/02/19
Programma gestione display	20/02/19	10/05/19
Sistemare cablaggi e componenti	20/02/19	10/05/19
Programma gestione anti-rimbal...	13/03/19	13/03/19
Realizzare commenti per codice	10/04/19	15/05/19
Realizzazione codice per "QUIT"	08/05/19	08/05/19
Realizzazione Moduli	15/02/19	03/05/19
1) Cumulativo 60s,S	15/02/19	22/02/19
2) Maratona 5min,S	22/02/19	22/02/19
3) Corsa 50p,S	22/02/19	22/02/19
4) Stretching angolare	22/02/19	27/02/19
6) Test di Leger o Bleep,S	13/03/19	13/03/19
7) 50p temporizzati,1p/s,S	27/02/19	27/02/19
8) Staffetta 120s,S	27/02/19	03/05/19
9) Reazione, somma matema...	15/03/19	29/03/19
10) Tabelline, test di velocità	15/03/19	03/05/19
11) Stretching angolare 25p	15/03/19	15/03/19
12) Stretching angolare	15/03/19	15/03/19
13) 25p temporizzati, 1p/s,J	15/03/19	15/03/19
14) 50p temporizzati, 1p/s,J	15/03/19	15/03/19

Programma gestione anti-rimbal...	13/03/19	13/03/19
Realizzare commenti per codice	10/04/19	15/05/19
Realizzazione codice per "QUIT"	08/05/19	08/05/19
Realizzazione Moduli	15/02/19	03/05/19
1) Cumulativo 60s,S	15/02/19	22/02/19
2) Maratona 5min,S	22/02/19	22/02/19
3) Corsa 50p,S	22/02/19	22/02/19
4) Stretching angolare	22/02/19	27/02/19
6) Test di Leger o Bleep,S	13/03/19	13/03/19
7) 50p temporizzati,1p/s,S	27/02/19	27/02/19
8) Staffetta 120s,S	27/02/19	03/05/19
9) Reazione, somma matema...	15/03/19	29/03/19
10) Tabelline, test di velocità	15/03/19	03/05/19
11) Stretching angolare 25p	15/03/19	15/03/19
12) Stretching angolare	15/03/19	15/03/19
13) 25p temporizzati, 1p/s,J	15/03/19	15/03/19
14) 50p temporizzati, 1p/s,J	15/03/19	15/03/19
15) Cumulativo 30s,J	15/03/19	15/03/19
16) Cumulativo 60s,J	15/03/19	15/03/19
17) Corsa 25p,J	15/03/19	15/03/19
18) Corsa 50p,J	15/03/19	15/03/19
19) Maratona 3min,J	15/03/19	15/03/19
20) Simon 20p, 17lvl	05/04/19	05/04/19
21) Flash test, 5schemi	05/04/19	03/05/19
22) Anti-Flash test	12/04/19	03/05/19
23) Reazione Veloce, 10 schemi	17/04/19	17/04/19
Problemi da gestire	29/03/19	29/03/19
Fine Implementazione	03/05/19	03/05/19
Test	13/02/19	15/05/19
Test di funzionamento Hardware	13/02/19	15/02/19
Test moduli	15/02/19	15/05/19
Gantt Consuntivo	17/05/19	17/05/19
Fine progetto	22/05/19	22/05/19

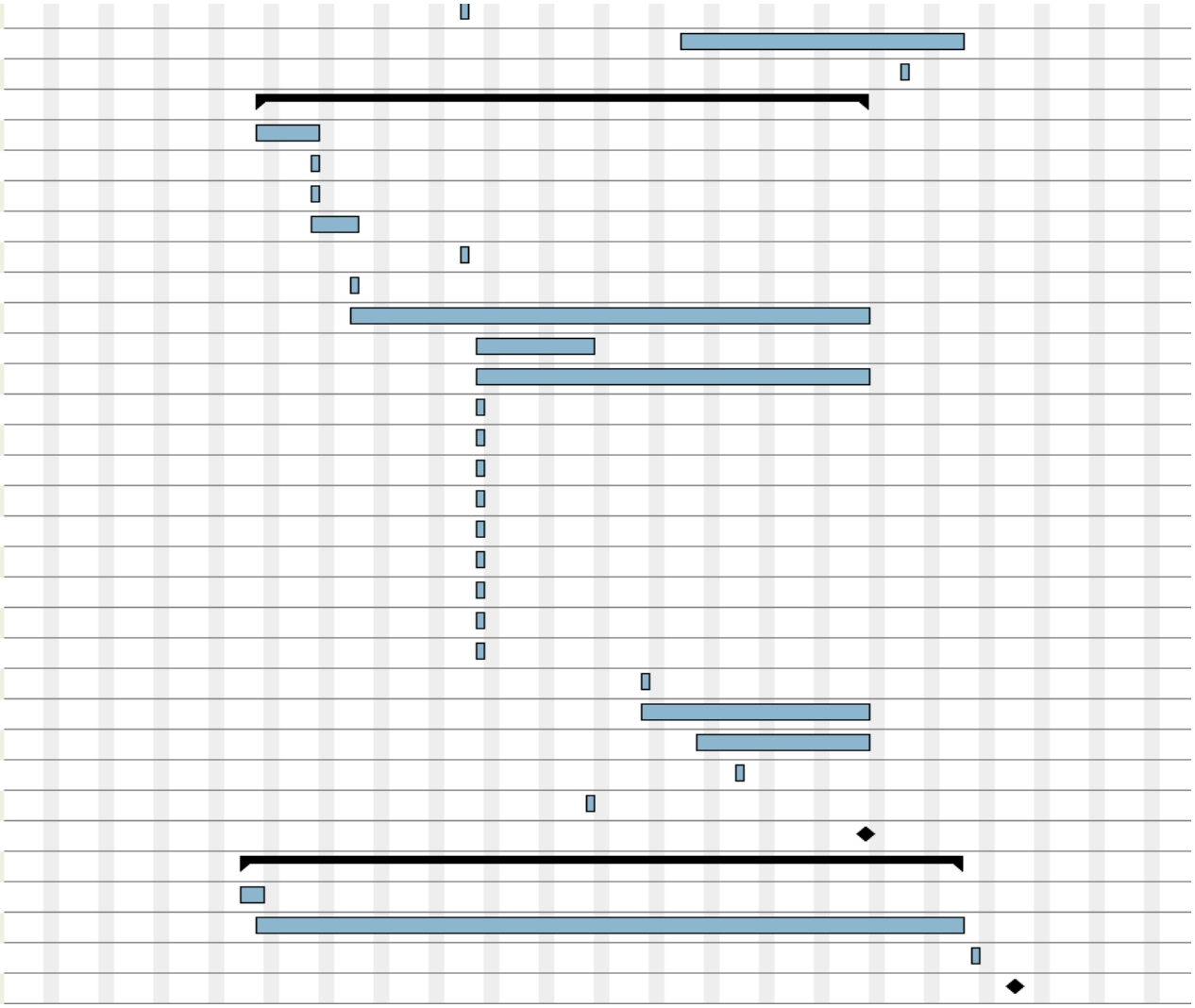


Figura 7 Diagramma di Gantt consuntivo

Rispetto alla pianifica, la data di consegna è stata posticipata di 1 lezione.

Siamo riusciti ad anticipare di 1 lezione l'inizio della realizzazione della presentazione, in questo modo abbiamo potuto curarla maggiormente terminandola comunque secondo la data prevista.

A causa di alcuni difetti del QdC, la fase di domande al committente si è protratta oltre la prima lezione proprio perché mano a mano che procedevamo nella realizzazione dei requisiti, ci rendevamo conto di alcune ambiguità o limitazioni che non potevamo risolvere.

Per quanto riguarda l'implementazione, abbiamo aggiunto alcune fasi che nella pianifica non consideravamo necessarie perché eravamo erroneamente convinti che l'hardware fosse già pienamente operativo.
La maggior parte dei programmi sono stati realizzati in modo che potessero servire diversi requisiti semplicemente modificando un parametro nella chiamata del metodo. Per questo motivo, alcuni programmi hanno richiesto molto tempo, mentre altri solo il tempo di testarli.
Il requisito numero 5 non abbiamo potuto implementarlo dal momento che era richiesta una piattaforma di cui non disponevamo.

I test si sono estesi su tutta la durata dell'implementazione come previsto.

La realizzazione del Gantt consuntivo è stata l'ultima cosa che abbiamo svolto (escluso il lavoro relativo alla doc).

6 Conclusioni

Troviamo questo progetto molto utile per chiunque voglia migliorare i propri riflessi e tempo di coordinamento ottico – motorio.

Questo sistema può essere utilizzato da chiunque, grazie ai vari programmi senior e junior e unisce in divertimento con attività fisica e visiva.

6.1 Sviluppi futuri

Come sviluppo futuro abbiamo pensato a l'implementazione di un sistema di archiviazione per i punteggi effettuati dai vari utenti, in particolare tramite la rete scolastica andare a scrivere in un file i punteggi dei giocatori ai relativi programmi.

Abbiamo inoltre pensato di aggiungere eventualmente alcune modalità per rendere il sistema ancora più completo.

Si potrebbero effettuare delle modifiche all'hardware del progetto, in particolare utilizzare dei LED più potenti in modo da poter utilizzare il sistema in un contesto anche molto illuminato (vedi capitolo Mancanze/limitazioni conosciute).

6.2 Considerazioni personali

Abbiamo trovato il progetto interessante sotto diversi punti di vista. Prima di tutto abbiamo apprezzato l'idea di continuare un progetto ideato dalla classe di terza dello scorso anno.

7 Fonti

7.1 Sitografia

Questi sono i link delle pagini principali dei siti (le pagine più dettagliate sono situate all'interno dei diari di lavoro) che abbiamo visitato durante il progetto:

- https://forum.arduino.cc/index.php, Forum Arduino, 02-22-2019
- https://www.draw.io/, Draw.io, 03.27.2019
- https://www.sparkfun.com/products/12660, SparkFun Electronics, 04-17-2019
- https://www.draw.io/, Draw.io, 03.05.2019

8 Allegati

- Diari di lavoro
- Codici sorgente
- Manuali procedura di check
- Qdc
- Prodotto