

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	3
1.1	Informazioni sul progetto	3
1.2	Abstract	3
1.3	Scopo	3
Analisi		3
1.4	Analisi del dominio	3
1.5	Analisi e specifica dei requisiti	4
1.6	Pianificazione	15
1.6.1	Preventivo	15
1.7	Analisi dei mezzi	17
1.7.1	Software	17
1.7.2	Hardware	17
2	Progettazione	18
2.1	Design dell'architettura del sistema	18
2.2	Design procedurale	19
2.2.1	Programmi	19
2.2.2	Procedura di check	20
3	Implementazione	21
3.1	Cumulativo 60 secondi, senior	23
3.2	Maratona (5 minuti) senior	24
3.3	Corsa 50 pulsanti, senior	25
3.4	Stretching angolare, 100 pulsanti, senior	27
3.5	(tiro a vuoto, - pulsanti 1 secondo, senior)	29
3.6	Test di Lèger o Beep test, senior	29
3.7	50 Pulsanti temporizzati, 1 secondo per pulsante, senior	32
3.8	Staffetta 4 giocatori (tempo totale 120 secondi), senior	34
3.9	Reazione, somma matematica	35
3.10	Tabelline, test di velocità	38
3.11	Stretching angolare, 25 pulsanti	41
3.12	Stretching angolare, 50 pulsanti	41
3.13	25 Pulsanti temporizzati, 1 secondo per pulsante, junior	41
3.14	50 Pulsanti temporizzati, 1 secondo per pulsante, junior	41
3.15	Cumulativo 30 secondi, junior	41
3.16	Cumulativo 60 secondi, junior	41
3.17	Corsa 25 pulsanti, junior	42
3.18	Corsa 50 pulsanti, junior	42
3.19	Maratona (3 minuti) junior	42
3.20	Semplice gioco Simon: 20 pulsanti, 17 livelli	43
3.21	Flash test, 5 schemi	45
3.22	Anti - flash test	50
3.23	Reazione veloce, 10 schemi	51
3.24	Errori e problemi da gestire	54
4	Test	55
4.1	Protocollo di test	55
4.2	Risultati test	67
4.3	Mancanze/limitazioni conosciute	68
5	Consuntivo	69
6	Conclusioni	72
6.1	Sviluppi futuri	72
6.2	Considerazioni personali	72
7	Fonti	73
7.1	Sitografia	73
8	Allegati	73

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: <ul style="list-style-type: none"> • 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 pedana 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: <ul style="list-style-type: none"> • 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

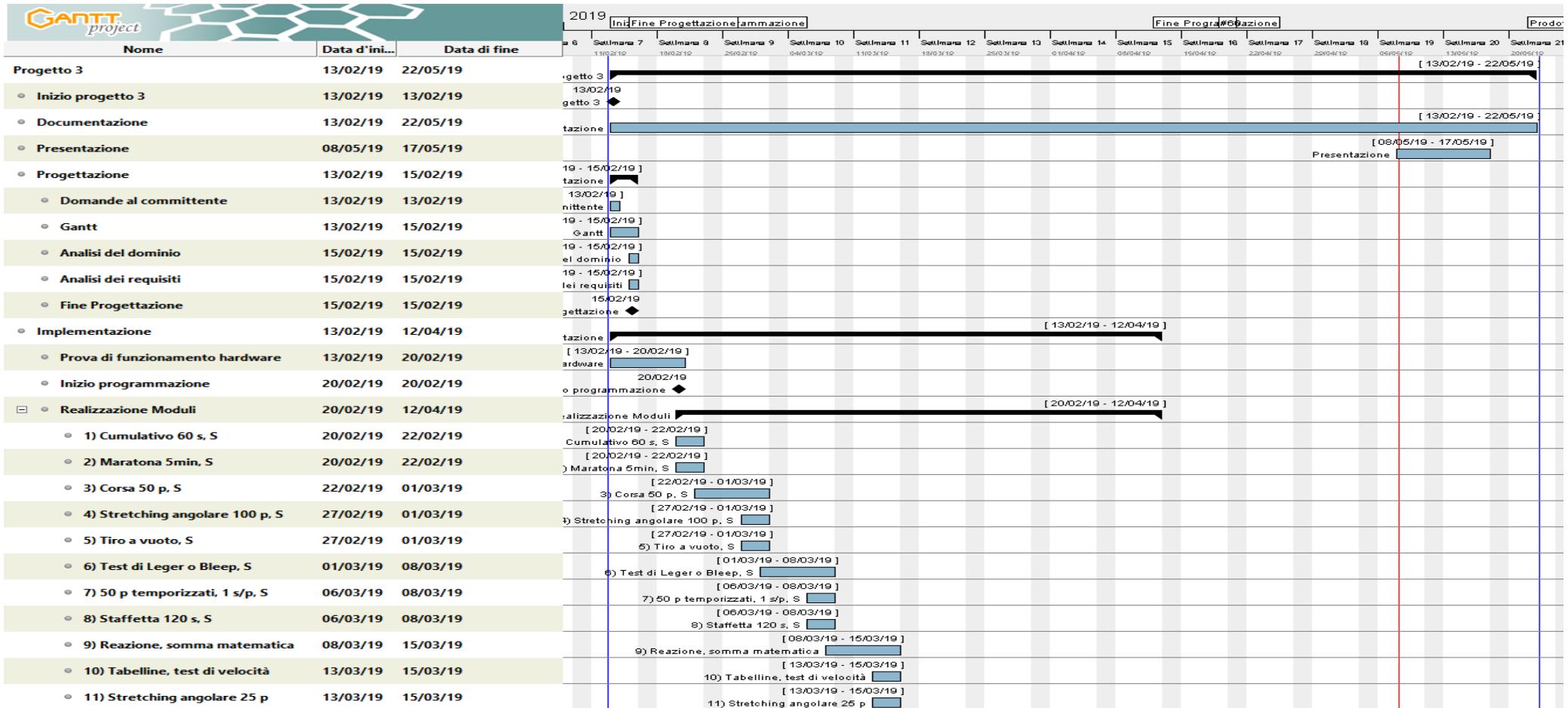
ID: REQ-22	
Nome	Anti - flash test
Priorità	1
Versione	2.0
Note	
Sotto requisiti	
001	Il sistema accende fino a 6 pulsanti, per 5 schemi (5 volte)
002	L'utente sceglie la durata dell'accensione dei pulsanti selezionando il tasto '@' più i secondi (da 1 a 8)
003	Quando il tempo termina il sistema emette 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



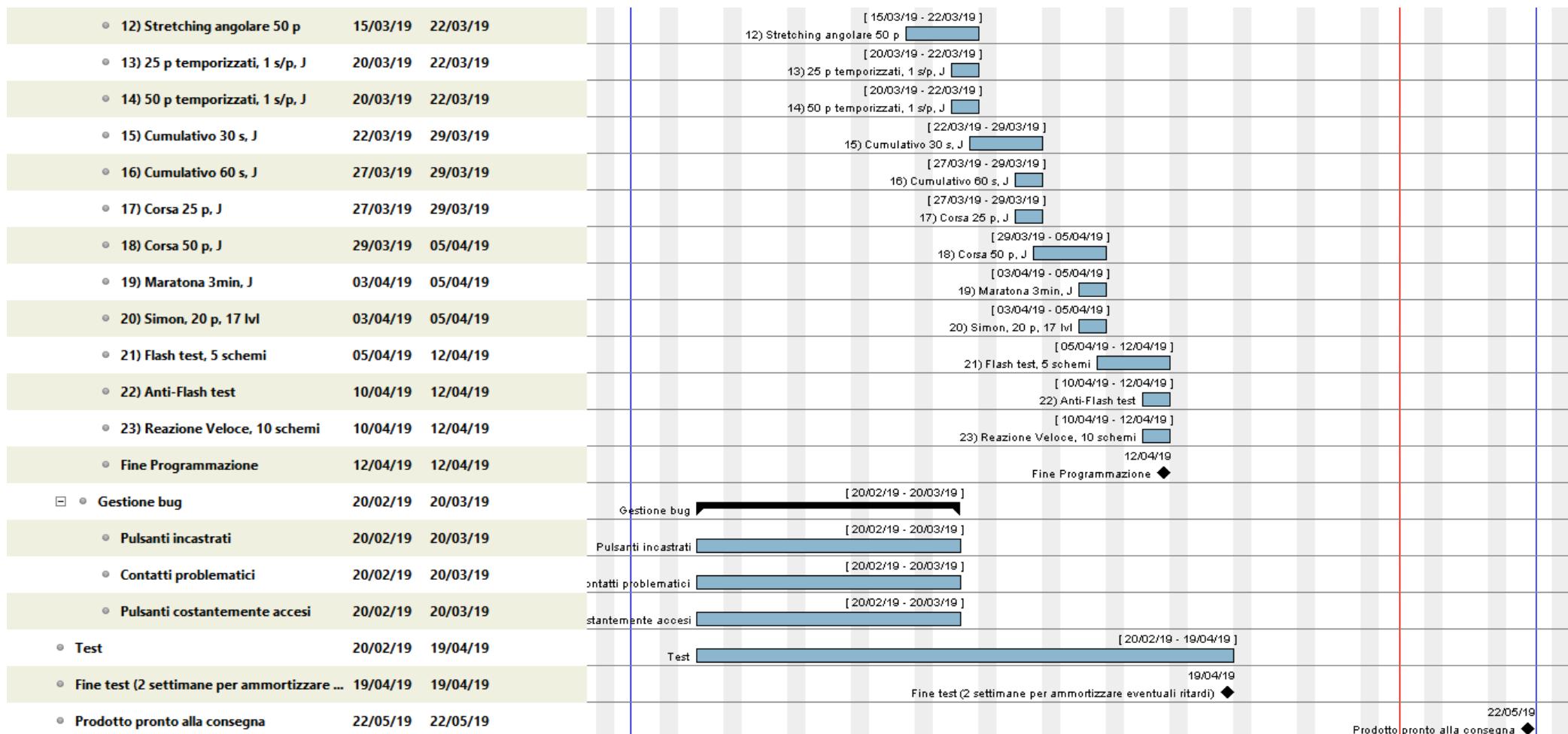


Figura 1 Gantt preventivo

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

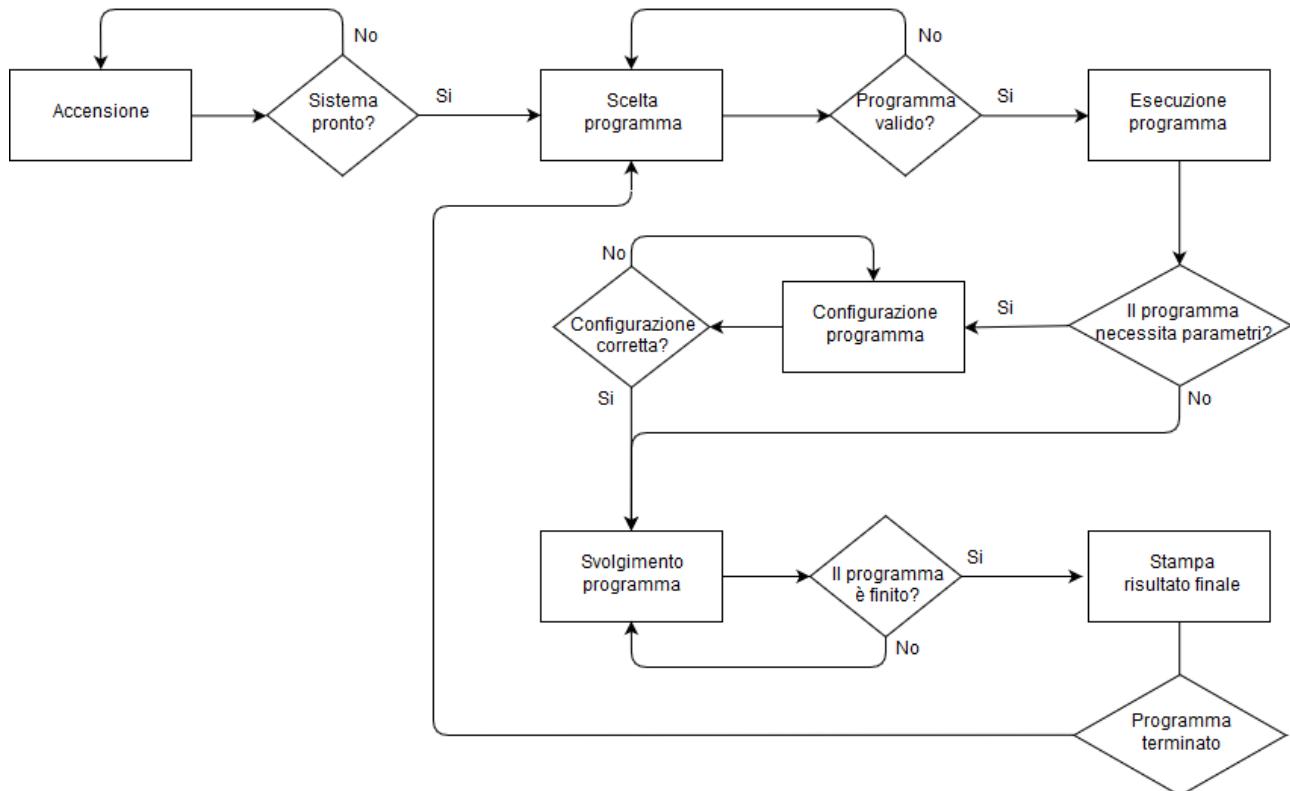


Figura 2 Design dell'architettura del sistema

2.2 Design procedurale

2.2.1 Programmi

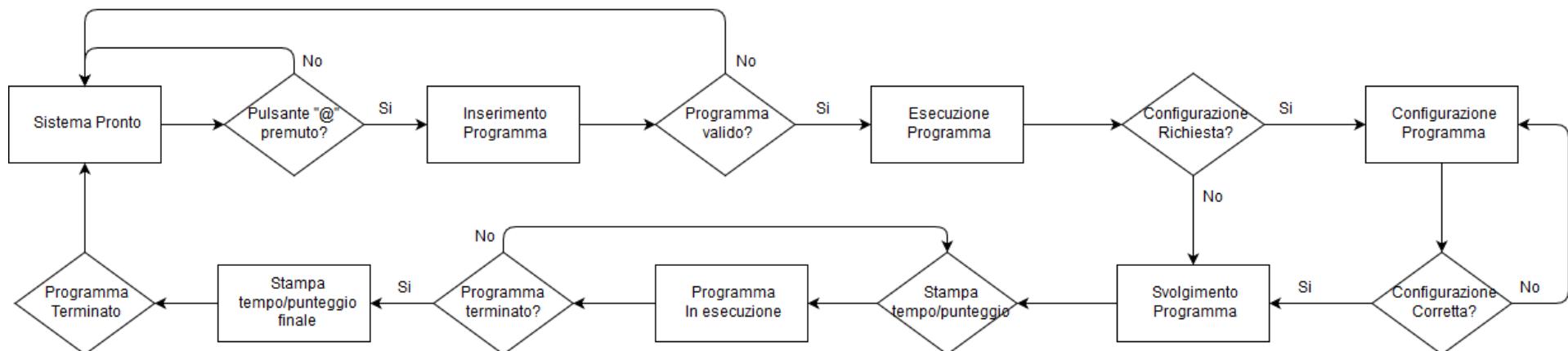


Figura 3 Diagramma procedurale "Programmi"

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"

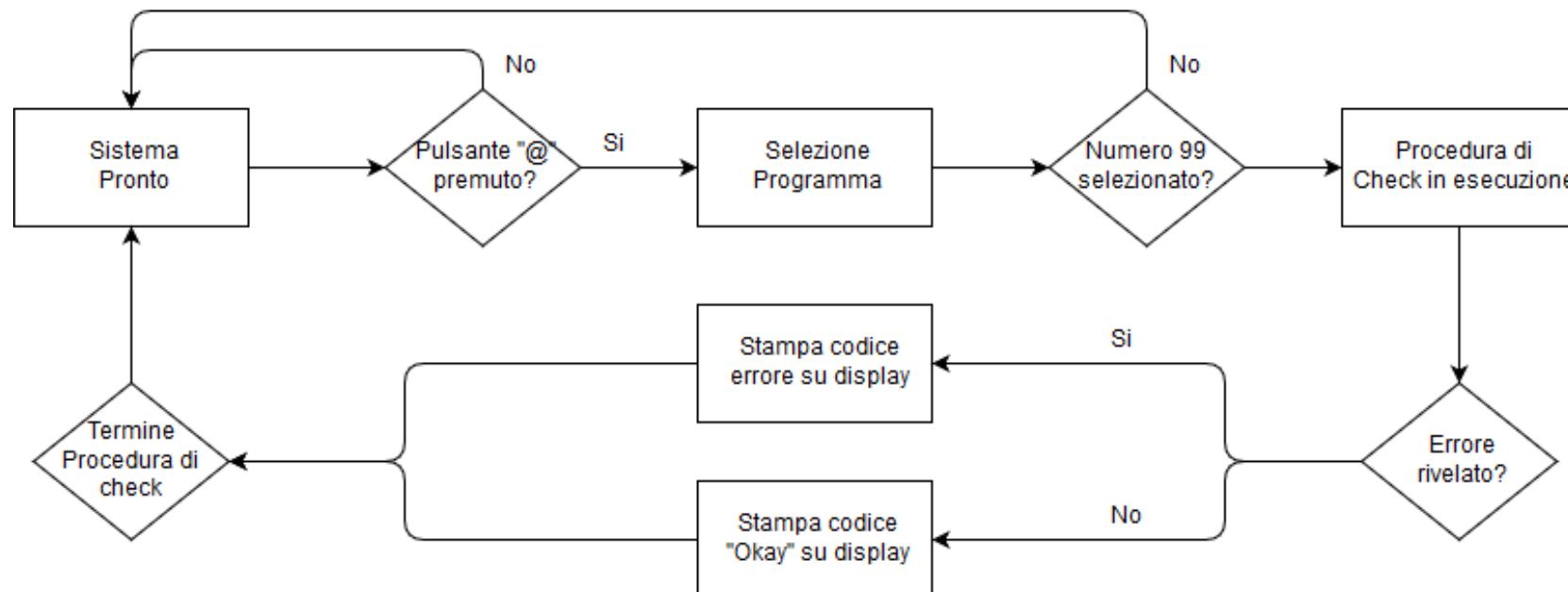


Figura 4 Diagramma procedurale "Procedura di check"

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     }
120 }
121 else
122 {
123     resetTime = millis();
124 }
125 }
```

```
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 connessioni 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;
610 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 &&
624 programRunning)
625 {
626     isPressed(44);
627     elapsed = (millis() - start);
628     sendData(0, elapsed / 100);
629     bool presslastState = getLastState(currentPin);
630     bool presscurrentState = isPressed(currentPin);
631     if (presslastState == HIGH && presslastState != presscurrentState)
632     {
633         pressedButtons++;
634         sendData(1, pressedButtons);
635         digitalWrite(currentPin + 1, LOW);
636         if (senior)
637         {
638             currentPin = getRandom(buttonPins, SIZE);
639         }
640         else
641         {
642             currentPin = getRandom(juniorPins, JUNIOR_SIZE);
643         }
644         digitalWrite(currentPin + 1, HIGH);
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'array *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);
584     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 pulsanti 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 }
```

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         }  
769     }  
770 }
```

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
509             2.0");
510         }
511     }
```

Esempio di utilizzo:

XXX	<code>temporized(25, true);</code>
-----	------------------------------------

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

Esempio di utilizzo:

XXX	<code>cumulative(30000, true);</code>
-----	---------------------------------------

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.

```
810 setLcdText("Hai selezionato: " + String(millisec / 1000), "Inizio in 3  
811 secondi", "", "BATTAK 2.0");  
812 delay(3000);  
813 resetButtonsState();  
814 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.

```
815 for (int i = 0; i < 8; i++)  
816 {  
817     int sum = 10;  
818     int one = 0;  
819     int two = 0;  
820     while (sum >= 10)  
821     {  
822         one = random(0, 9);  
823         two = random(0, 9);  
824         sum = one + two;  
825     }  
826     setLcdText(String(one) + " + " + String(two) + " = ?", "Punti: " +  
827     String(points) + "/8", "", "BATTAK 2.0");  
828     sendData(0, one);  
829     sendData(1, two);  
830  
831     startTime = millis();  
832     waiting = true;  
833     while (millis() - startTime <= millisec && waiting && programRunning)  
834     {  
835         for (int x = 0; x < SIZE; x++)  
836         {  
837             int bpin = buttonPins[x];  
838             bool lastState = getLastState(bpin);  
839             bool currentState = isPressed(bpin);  
840             if (currentState == HIGH && currentState != lastState &&  
841                 getLabel(bpin) == sum)  
842             {  
843                 points++;  
844                 setLcdText(String(one) + " + " + String(two) + " = ?", "Punti: " +  
845                 String(points) + "/8", "", "BATTAK 2.0");  
846                 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 pulsanti, 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'istante in cui parte il gioco.

```
79 setLCDText("Hai selezionato la", "tabellina del " + String(boards), "",
80 "BATTAK 2.0");
81 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 sono il 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 *ò* 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 creata 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 effettuato 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**), iniziate 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	}

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", "", "",  
253 "BATTAK 2.0");  
254 resetLeds();  
255 int points = 0;  
256 bool playing = true;  
257 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         {
317             bool edited = false;
318             for (int x = 0; x < 6; x++) {
319                 if (sequence[x] == pin) {
320                     edited = true;
321                     if (onButtons)
322                     {
323                         digitalWrite(pin + 1, HIGH);
324                     }
325                     else
326                     {
327                         digitalWrite(pin + 1, LOW);
328                     }
329                     pressed += 1;
330                     points += 1;
331                     sendData(0, points);
332                     if (pressed == 6)
333                     {
334                         insideLevel = false;
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 effettuati 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.

Innanzitutto 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         digitalWrite(pin + 1, HIGH);
40         setLcdText("Il pulsante dovrebbe", "essere acceso", "", "BATTAK 2.0");
41         delay(1000);
42     }
}
```

Esempio di utilizzo:

```
XXX systemCheck();
```

4 Test

4.1 Protocollo di test

Test Case:	TC-001	Nome:	Cumulativo 60 secondi, senior
Riferimento:	REQ-01		
Descrizione:	Programma che svolge il programma cumulativo (senior)		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	<ol style="list-style-type: none">1. Accendere il sistema2. Cliccare la combinazione di buttoni '@', '1' e '#'. 3. Premere i buttoni 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 buttoni cliccati.		

Test Case:	TC-002	Nome:	Maratona (5 minuti) senior
Riferimento:	REQ-02		
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:	<ol style="list-style-type: none">1. Accendere il sistema2. Cliccare la combinazione di buttoni '@', '2' e '#'. 3. Premere i buttoni 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 buttoni 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:	<ol style="list-style-type: none"> 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 pulsanti cliccati correttamente o il tempo con cui si ha cliccato i 50 pulsanti.		

Test Case:	TC-004	Nome:	Stretching angolare, 100 pulsanti, senior
Riferimento:	REQ-04		
Descrizione:	Si accendono 100 pulsanti in modo casuale, se l'utente sbaglia a cliccare il bottone la velocità di accensione dei pulsanti aumenta. Si accendono solo i pulsanti agli estremi del telaio. Inizialmente i pulsanti rimangono accesi per 1 secondo.		
Prerequisiti:	Sistema hardware “Batak” funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '4' e '#'. 3. Premere i pulsanti 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 pulsanti premuti correttamente. Se l'utente preme il pulsante sbagliato, la velocità di accensione aumenta. Dopo 100 pulsanti cliccati il programma termina. Si accendono solo i pulsanti agli estremi del telaio.		

Test Case:	TC-005	Nome:	Tiro a vuoto – pulsanti 1 secondo, senior
Riferimento:	REQ-05		
Descrizione:	Si accendono 100 pulsanti in modo casuale, se l'utente sbaglia a cliccare il bottone la velocità di accensione dei pulsanti aumenta (alcuni pulsanti simulano l'accensione). Inizialmente i pulsanti 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '5' e '#'. 3. Premere i pulsanti 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 pulsanti premuti correttamente. Se l'utente preme il pulsante sbagliato (alcuni pulsanti simulano l'accensione) la velocità di accensione aumenta e vengono sottratti 5 punti. Dopo 100 pulsanti 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 consecutive 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '6' e '#'. 3. Premere i pulsanti che si illuminano fino a quando il sistema si ferma. 		
Risultati attesi:	Il programma funziona in modo corretto, ci sono 10 livelli e crescente 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	Nome:	50 pulsanti temporizzati, 1 secondo per pulsante, senior		
Riferimento:	REQ-07				
Descrizione:	Il sistema accende in successione casualmente 50 pulsanti. 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '7' e '#'. 3. Premere i pulsanti 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 pulsanti 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	Nome:	Staffetta 4 giocatori 120 secondi (30 secondi a testa)		
Riferimento:	REQ-08				
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 pulsanti in modo corretto.				
Prerequisiti:	Sistema hardware "Batak" funzionante				
Procedura:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '8' e '#'. 3. Premere i pulsanti 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:	TC-009	Nome:	Reazione somma aritmetica
Riferimento:	REQ-09		
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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '9' e '#'. 3. Premere i buttoni 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:	TC-010	Nome:	Tabelline di velocità
Riferimento:	REQ-10		
Descrizione:			
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '1' e "0". 3. Premere i buttoni 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni ‘@’, ‘1’ e “1.” 3. Premere i buttoni che si illuminano fino a quando il sistema si ferma. 		
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il numero di buttoni rimanenti e il numero di colpi corretti. Si accendono solo i buttoni agli estremi del telaio, se si preme il bottone sbagliato la sequenza aumenta di velocità. Dopo l'accensione di 25 buttoni 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni ‘@’, ‘1’ e “2”. 3. Premere i buttoni che si illuminano fino a quando il sistema si ferma. 		
Risultati attesi:	Il programma funziona in modo corretto, viene stampato il numero di buttoni rimanenti e il numero di colpi corretti. Si accendono solo i buttoni agli estremi del telaio, se si preme il bottone sbagliato la sequenza aumenta di velocità. Dopo l'accensione di 25 buttoni il programma termina.		

Test Case:	TC-013	Nome:	25 pulsanti temporizzati, 1 secondo per pulsante, junior		
Riferimento:	REQ-13				
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:	<ol style="list-style-type: none">1. Accendere il sistema2. 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 stampati i pulsanti premuti correttamente e i pulsanti rimanenti.				

Test Case:	TC-014	Nome:	50 pulsanti temporizzati, 1 secondo per pulsante, junior		
Riferimento:	REQ-14				
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:	<ol style="list-style-type: none">1. Accendere il sistema2. 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 stampati 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '1' e "5". 3. Premere i buttoni che si illuminano fino a quando il sistema si ferma. 		
Risultati attesi:	Il programma funziona in modo corretto, vengono accesi solo i buttoni 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '1' e "6". 3. Premere i buttoni che si illuminano fino a quando il sistema si ferma. 		
Risultati attesi:	Il programma funziona in modo corretto, vengono accesi solo i buttoni 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '1' e "7". 3. Premere i buttoni 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '1' e "8". 3. Premere i buttoni 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti ‘@’, ‘1’ e “9”. 3. Premere i pulsanti 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 pulsanti 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti ‘@’, ‘2’ e “0”. 3. Premere i pulsanti 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '2' e "1". 3. Premere i pulsanti 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di pulsanti '@', '2' e "2". 3. Premere i pulsanti 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:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '2' e "3". 3. Premere i buttoni 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 incastri, contatti problematici, li rimangono costantemente accesi). Vi è una procedura di check (all'accensione del sistema oppure su richiesta). Il sistema di gestione errori li individua e li segnala.		
Prerequisiti:	Sistema hardware "Batak" funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Accendere il sistema 2. Cliccare la combinazione di buttoni '@', '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).

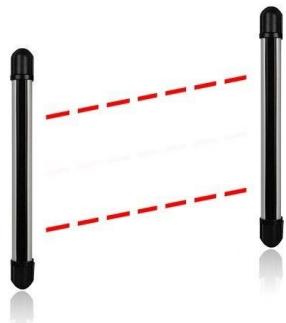


Figura 5 Esempio di barriera luminosa



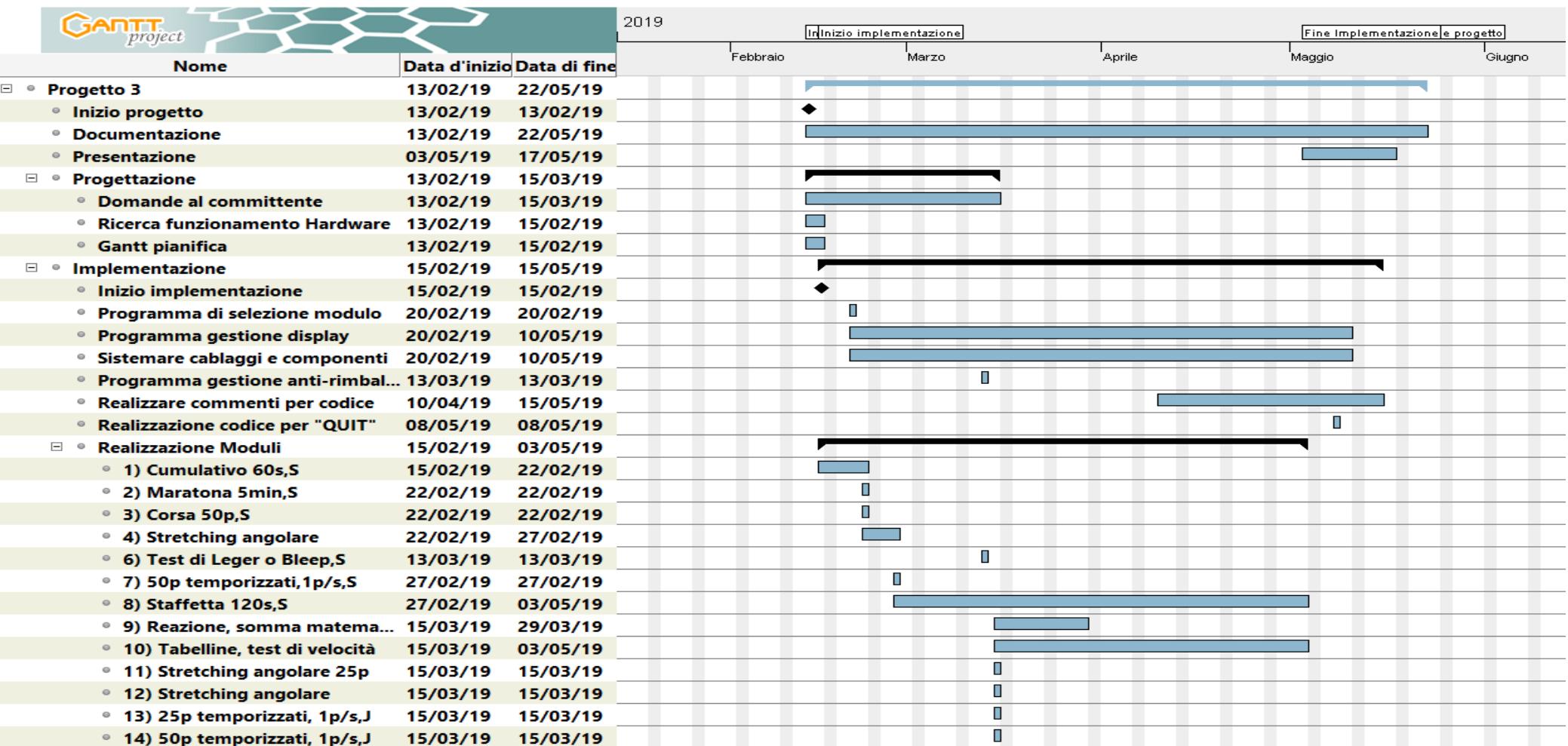
Figura 6 Esempio di pedana

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



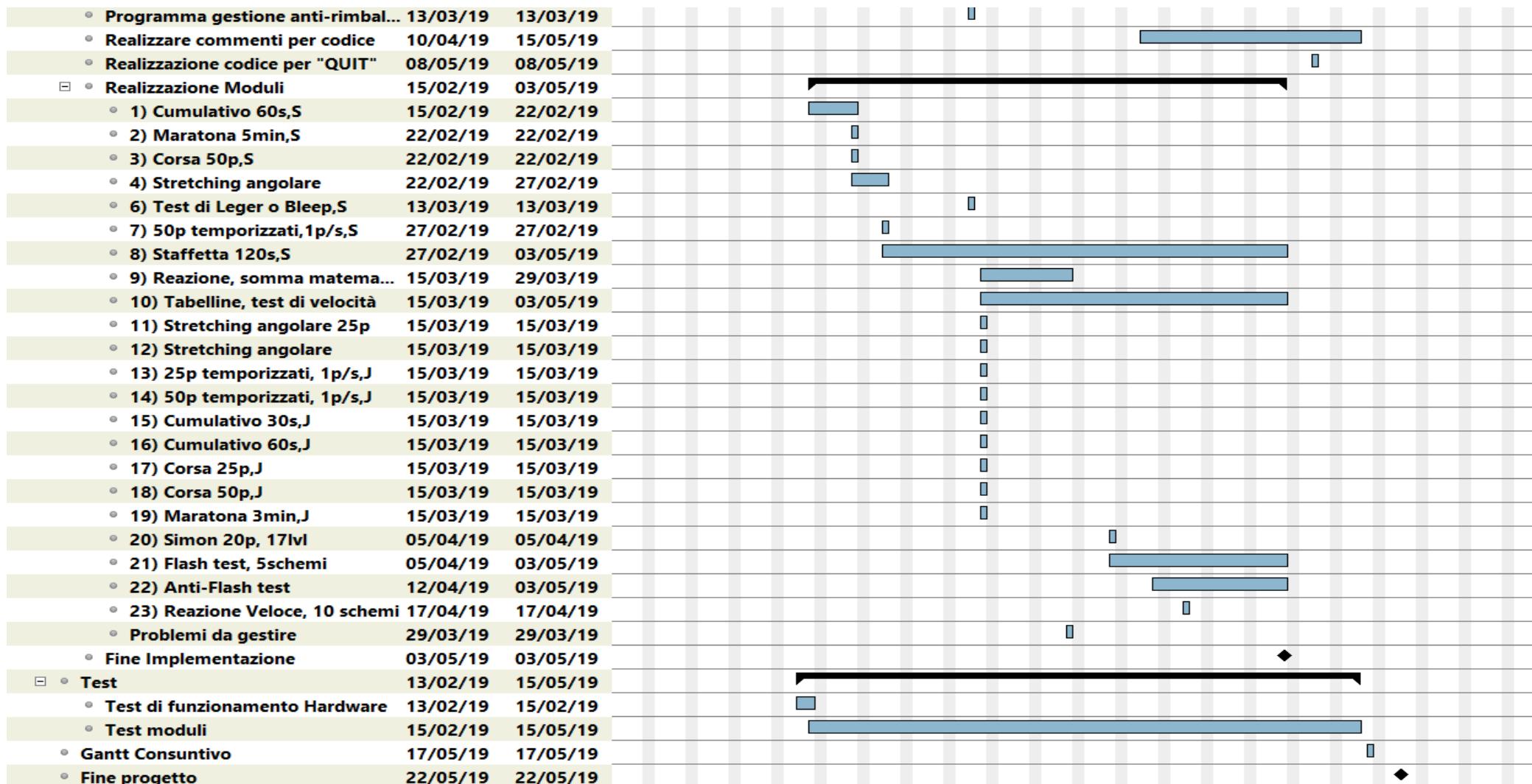


Figura 7 Diagramma di Gantt consuntivo

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 pagine 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

Procedura di check

Procedura di check

Per controllare che il sistema sia funzionante è stata progettata una procedura di check. È una procedura utilizzata per verificare che il sistema non rilevi errori.

Per avviare la funzione di check bisogna eseguire i seguenti passi:

1. Accendere il sistema e aspettare che sia pronto
2. Premere il pulsante “@”
3. Successivamente selezionare “99”
4. Attendere che il sistema finisca il check del sistema

Diario di lavoro

Luogo	Trevano
Data	2019-02-13

Lavori svolti

- Abbiamo ricevuto il QDC;
- Abbiamo realizzato le domande per il committente;
- Iniziato a raccogliere le informazioni per il progetto (documentazione dell'hardware, ...);
 - Sulla base di questa abbiamo cominciato ad eseguire dei test sul funzionamento del circuito;
 - Testato il funzionamento led e bottoni del telaio;
 - Mappato i pin dei bottoni con il relativo nome;
- Abbiamo realizzato una prima bozza della pianificazione;
- Cominciato la realizzazione della documentazione impostando i requisiti;

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

(Non ancora svolta definitivamente, ma per quanto riguarda la bozza: In linea)

Programma di massima per la prossima giornata di lavoro

Aggiunta dei requisiti alla documentazione

Completare il capitolo di analisi

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-02-15

Lavori svolti

- Finke ha dovuto svolgere la presentazione del progetto 2;
- Abbiamo:
 - Completato il diagramma Gantt;
 - Proseguito con la parte di analisi dei requisiti;
 - Completato l'analisi del dominio;
 - Completato lo scopo;
 - Realizzato un codice per verificare il funzionamento del display sette-segmenti;
 - Iniziato a realizzare il codice per il primo modulo;

Problemi riscontrati e soluzioni adottate

- Il sette-segmenti che dovrebbe indicare il tempo non funzionava;
 - Ci siamo accorti che il problema era nei contatti del display, quindi la soluzione che abbiamo adottato è stata quella di raddrizzarli e inserirli in modo corretto nei vari pin.
- La luminosità dei vari pulsanti era troppo bassa;
 - Avevamo dimenticato di impostare il pinmode;

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo

Programma di massima per la prossima giornata di lavoro

Completare la fase per il check del funzionamento dell'hardware

Proseguire il modulo 1 (Cumulativo 60 [s], Senior) e iniziare il successivo

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-02-20

Lavori svolti

- Abbiamo:
 - Proseguito con la parte di analisi dei requisiti;
 - Completato il codice per il primo requisito;
 - Iniziato a realizzare un menù per la selezione dei vari programmi (selezionare l'indice corrispondente all'esercizio desiderato);
 - Iniziato a realizzato il codice per gestire contatore punteggio e timer (o comunque i 2 display);
 - Realizzato interfaccia per far comunicare 2 Arduino;

Problemi riscontrati e soluzioni adottate

- Alcune parti del QdC non erano ben definite per via di alcuni copia-incolla;
 - Chiesto chiarimenti riguardo le specifiche in dubbio.
- Dopo vari test ci siamo accorti che uno dei 2 display non funziona;
 - Ne abbiamo chiesto un altro che però arriverà solo venerdì (22/2/2019).
- Il cavo di rete che avremmo utilizzato per collegare i 2 Arduino (quello principale con quello dei 2 display) è troppo corto;
 - Ne abbiamo chiesto uno più lungo e lo abbiamo preparato in modo da poterlo utilizzare come avevamo bisogno.

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo (di circa 2 ore di lezione)

Programma di massima per la prossima giornata di lavoro

Rendere completamente operativo l'hardware

Eseguire i test relativi al primo requisito.

Completare il secondo requisito e iniziare il 3°

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

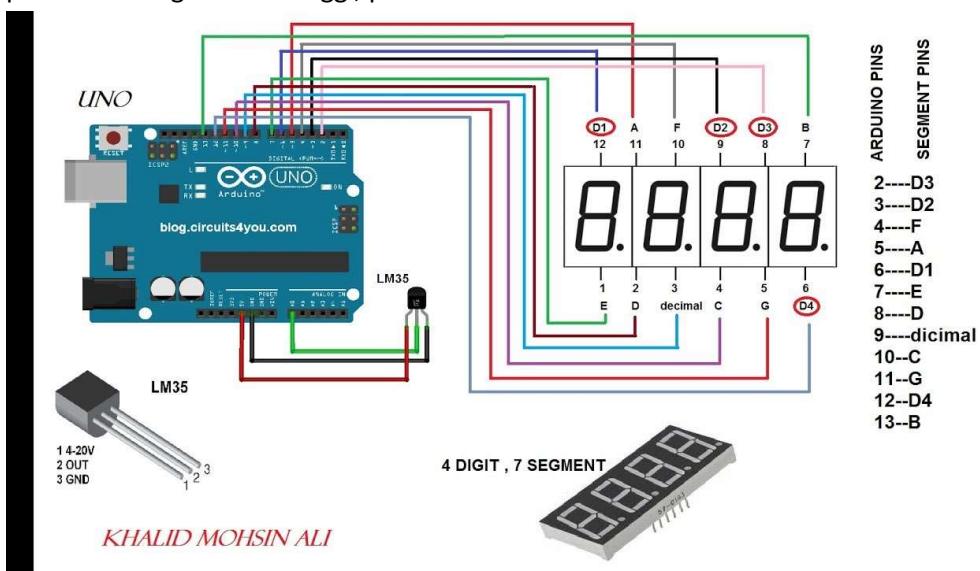
Luogo	Trevano
Data	2019-02-22

Lavori svolti

- Abbiamo:
 - Sistemato parzialmente la questione del display 7 segmenti che non funzionava correttamente.
Per fare questo, oltre a ricercare il problema, e agire di conseguenza, abbiamo dovuto dissaldare e poi saldare nuovamente un pin del display;
 - I 2 display sono stati collegati fra loro;
 - Risolto problema di conversione da stringa ad intero utilizzando la funzione atol()
<https://forum.arduino.cc/index.php?topic=475803.0>;
 - Risolto problema parametri di tipo long passando da: cumulative(60 * 1000, false), a cumulative (60000, false);
 - Modificato il tipo di ritorno della funzione principale per la seleziona delle modalità di gioco da stringa ad intero;
 - Implementato definitivamente programmi 1, 2 e 3;
 - Iniziata l'implementazione del programma 4;

Problemi riscontrati e soluzioni adottate

- Il secondo display (quello fissato direttamente su una veroboard e che nella lezione precedente abbiamo dato per rotto e quindi chiesto una sostituzione) non accende tutti i segmenti che dovrebbe, ma rispetto alla lezione precedente, ci siamo resi conto che probabilmente il problema è legato ai cablaggi, per controllare abbiamo utilizzato:



https://i.ytimg.com/vi/utsQWM_FVQg/maxresdefault.jpg ;

- Ci siamo accorti che 1 pin del display non era saldato correttamente, quindi l'abbiamo risaldato;

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

- Per semplificare la stesura del codice abbiamo deciso di utilizzare una libreria (<https://github.com/sparkfun/SevSeg>) .
- Altri link che abbiamo utilizzato per raggiungere la soluzione:
 - <https://create.arduino.cc/projecthub/essoelectronic/sevseg-library-on-4-digit-anode-segment-ae7eed>;
 - <https://www.hackster.io/SAnwandter1/programming-4-digit-7-segment-led-display-2d33f8>;
 - <https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>;
- Alcuni valori del display vengono ancora rappresentati in maniera errata;

Punto della situazione rispetto alla pianificazione

In anticipo

Programma di massima per la prossima giornata di lavoro

Risolvere problema del display.

Proseguire l'implementazione del programma 4

Iniziare implementazione programma 5

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-02-27

Lavori svolti

- Ci è stato consegnato il display 7 segmenti che avevamo chiesto qualche settimana fa
- Abbiamo:
 - Implementati programmi 4, 7 e 8 (5 e 6 posticipati a causa della necessità del cicalino);
 - Realizzato i primi 8 test case;

Problemi riscontrati e soluzioni adottate

- Avendo ricevuto il nuovo 7 segmenti, abbiamo risolto tutti i problemi legati ad esso, ci rimane solo da installarlo sul telaio.
- Non c'è l'anti-rimbalzo sui pulsanti, dovremo implementarlo al più presto.

Punto della situazione rispetto alla pianificazione

In anticipo

Programma di massima per la prossima giornata di lavoro

Completare programmi 5 e 6

Installare il display 7 segmenti sul telaio

Diario di lavoro

Luogo	Trevano
Data	2019-03-01

Lavori svolti

Dal momento che entrambi i miei compagni di gruppo (Beffa e Finke), oggi erano assenti, ho approfittato di queste ore per effettuare una revisione generale del lavoro svolto nel corso delle lezioni svolte e sistemare:

- Capitolo 1.6 della documentazione realizzando una versione più compatta e indicativa del Gantt per poterlo rendere più grande e quindi più leggibile;
- Relativo al capitolo 1.6 ho anche realizzato il commento relativo al suddetto Gantt;
- Aggiunto altri requisiti;

Problemi riscontrati e soluzioni adottate

- Non c'è l'anti-rimbalzo sui pulsanti, dovremo implementarlo al più presto.

Punto della situazione rispetto alla pianificazione

In anticipo di circa 1 settimana

Programma di massima per la prossima giornata di lavoro

Completare programmi 5 e 6

Installare il display 7 segmenti sul telaio

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-03-13

Lavori svolti

Abbiamo:

- Preparato il cavo per collegare l'Arduino principale con quello che gestisce i 2 display;
- Risolto il problema relativo all'anti-rimbalzo;
- Rilevati i pin da utilizzare per i Buzzer;
- Implementati programma 6;
- Risolti vari problemi in altri programmi;

Problemi riscontrati e soluzioni adottate

- Codice che gestisce l'anti-rimbalzo:

```
bool getLastState(int pin)
{
    int index = getLabel(pin);
    return buttonStatus[index];
}

void setLastState(int pin, bool state)
{
    int index = getLabel(pin);
    buttonStatus[index] = state;
}
```

Punto della situazione rispetto alla pianificazione

In ritardo (non avevamo preso in considerazione la pausa dovuta alle vacanze scolastiche).

Programma di massima per la prossima giornata di lavoro

Completare programmi fino all'11 e cominciare il 12

Fissare il display 7 segmenti sul telaio

Riflettere sull'implementazione del display LCD

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-03-15

Lavori svolti

Oggi abbiamo:

- Aggiunto i programmi dal 9 al 19;
- Revisionato alcuni di quelli realizzati in precedenza per aggiungere dinamicità alla stesura del codice;
- Discusso dei programmi “Tabelline, test di velocità” e “Tiro a vuoto – pulsanti 1 secondo, senior”, il problema consiste nel fatto di dover fare un suono audio e non di frequenza e che di conseguenza con il buzzer non possiamo farlo. Ci è stato detto di lasciarli da parte per il momento;
- Proseguito i protocolli di test dal 12 al 20;
- Completato la stesura dei requisiti (realizzati 20, 21, 22, 23, 24);
- Revisionato e corretto la doc;

Problemi riscontrati e soluzioni adottate

- Con il buzzer possiamo solo esprimere delle frequenze e non del “testo”
 - Lasciare da parte per il momento;

Punto della situazione rispetto alla pianificazione

Recuperato il ritardo della lezione scorsa arrivando ad essere in anticipo sulla pianificazione

Programma di massima per la prossima giornata di lavoro

Test di Léger o Beep test, senior

Mancanza: Implementare il displaylcd

Reazione, somma aritmetica

Mancanza: Displaylcd

Stretching angolare, 25 pulsanti

Fix display 7 segmenti (la virgola)

Stretching angolare, 50 pulsanti

Fix display 7 segmenti (la virgola)

Sistema di gioco e simulazione per l’allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-03-20

Lavori svolti

Oggi abbiamo utilizzato tutto il tempo a nostra disposizione per tentare di collegare i 2 Arduino per far sì che possano comunicare fra loro. Abbiamo dovuto rifare tutto il lavoro relativo al cavo di collegamento perché risultava impossibile mantenere collegati gli Arduino in maniera semplice, quindi abbiamo deciso di saldare ogni apice di ogni filo di rame ad un terminale di metallo in maniera che il collegamento sia molto più affidabile. Per quanto l'analisi con il Voltmetro sembri ritornare risultati corretti, il programma non risponde correttamente.

D'altro abbiamo terminato la redazione dei test case e dei requisiti sulla doc.

Problemi riscontrati e soluzioni adottate

- Cavo per comunicazione fra i 2 Arduino non funziona correttamente:
 - Test col voltmetro;
 - Riconrollo del codice e delle porte di ogni Arduino.

Punto della situazione rispetto alla pianificazione

In anticipo

Programma di massima per la prossima giornata di lavoro

Sistemare problema di comunicazione fra Arduino

Test di Léger o Beep test, senior

Mancanza: Implementare il displaylcd

Reazione, somma aritmetica

Mancanza: Displaylcd

Stretching angolare, 25 pulsanti

Fix display 7 segmenti (la virgola)

Stretching angolare, 50 pulsanti

Fix display 7 segmenti (la virgola)

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-03-22

Lavori svolti

Come prima cosa abbiamo sostituito il supporto per il pulsante 1 che ci era stato dato già rotto (seppur funzionante).

Abbiamo poi utilizzato il tempo fino alla pausa per fare diversi test sulla trasmissione dei dati fra gli Arduino per arrivare alla conclusione che probabilmente il segnale viene disperso a causa della lunghezza (spropositata) del cavo. Per questo dopo la pausa abbiamo preparato dei nuovi cavi tagliati su misura per poterli provare immediatamente la prossima lezione.

In parallelo a questo abbiamo implementato il display LCD in tutti i moduli dove era richiesto.

Abbiamo anche realizzato il capitolo del design dell'architettura nella documentazione e creato un documento a parte per illustrare la procedura di check.

Durante l'ultima ora di lezione abbiamo dovuto assistere alla presentazione di una possibile scuola da affrontare dopo la SAM.

Problemi riscontrati e soluzioni adottate

- Cavo per comunicazione fra i 2 Arduino non funziona correttamente:
 - Test col voltmetro;
 - Ricontrollo del codice e delle porte di ogni Arduino;
 - Tagliare dei cavi su misura per escludere la perdita di segnale,

Punto della situazione rispetto alla pianificazione

Stiamo lavorando in maniera più separata di quanto avessimo pronosticato pensando che l'hardware fosse già pronto quando invece, necessitava di modifiche e/o migliorie.

Tutto sommato siamo in linea con la pianificazione per quanto riguarda il rapporto tempo/lavoro rimanente.

Programma di massima per la prossima giornata di lavoro

Sistemare problema di comunicazione fra Arduino

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-03-27

Lavori svolti

Appena entrati in classe abbiamo fatto il solio controllo dell'hardware e ci siamo accorti che il tasto "#" è incastrato e quindi risulta sempre premuto.

Abbiamo:

- Proseguito la stesura della doc realizzando (utilizzando <https://www.draw.io/>) una base di:
 - Diagramma del design dell'architettura;
 - Diagramma procedurale;
- Sistemato la struttura;
 - Collegati i 2 Arduino con i nuovi cavi e testata la comunicazione;
 - Fissati sia i 2 display 7 segmenti che l'Arduino che li comanda al telaio;
 - Fissato il display LCD al telaio;
- Eseguito un programma a scelta fra quelli completati fino ad ora come beta per i ragazzi delle porte aperte;
- Testati i programmi completamente funzionanti:
1, 2, 3, 4, 5 (non da implementare quindi lo consideriamo come completato), 7, 13, 14, 15, 16, 17, 18 e 19;

Problemi riscontrati e soluzioni adottate

- Tasto "#" incastrato:
 - Smontato il bottone e relativo supporto;
 - Disincastato il bottone;
 - Sistemati i contatti del led e del bottone stesso.

Punto della situazione rispetto alla pianificazione

I programmi completati ma ancora da testare sono:

6, 8 e 9;

Programmi in corso:

- Stretching angolare, 25 pulsanti: dobbiamo sistemare i display 7 segmenti;
- Stretching angolare, 50 pulsanti: dobbiamo sistemare i display 7 segmenti;

Programmi da revisionare:

- Tabelline, test di velocità;
- Tiro a vuoto – pulsanti 1 secondo, senior;

Programmi ancora da fare:

- Semplice gioco Simon: 20 pulsanti, 17 livelli;
- Flash test, 5 schemi;
- Anti-flash test;
- Reazione veloce, 10 schemi;

Globalmente siamo in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

Proseguire con i programmi mancanti

Diario di lavoro

Luogo	Trevano
Data	2019-03-29

Lavori svolti

Appena entrati in classe abbiamo fatto il solio controllo dell'hardware e ci siamo accorti che i display 7 segmenti in alto rispetto alla struttura, non mostravano ciò che avrebbero dovuto: abbiamo quindi controllato tutti i contatti (questo è bastato a risolvere il problema).

Abbiamo:

- Sistemato e testato il programma 9 (somma matematica);
- Iniziato l'implementazione del requisito "Errori e problemi da gestire";
- Completati i capitoli 4 e 6 della documentazione;
- Sistemato alcuni dettagli vari nella doc;

Durante la pausa un tester ha fatto incastrare il pulsante #, quindi lo abbiamo dovuto smontare e sistemare.

Problemi riscontrati e soluzioni adottate

- Display 7 segmenti non funzionanti:
 - Controllato i contatti fra i 2 Arduino;
 - Resettabili entrambi gli Arduino (prima quello principale e poi quello che controlla solo i 2 display);
- Pulsante # incastrato:
 - Smontato il bottone e relativo supporto;
 - Disincastrato il bottone;
 - Sistemati i contatti del led e del bottone stesso.

Punto della situazione rispetto alla pianificazione

Programmi in corso:

- Stretching angolare, 25 (e 50) pulsanti: dobbiamo sistemare i display 7 segmenti;

Programmi da revisionare:

- Tabelline, test di velocità;
- Tiro a vuoto – pulsanti 1 secondo, senior;

Programmi ancora da fare:

- Semplice gioco Simon: 20 pulsanti, 17 livelli;
- Flash test, 5 schemi;
- Anti-flash test;
- Reazione veloce, 10 schemi;

Globalmente siamo in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

Proseguire con i programmi mancanti

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-04-03

Lavori svolti

Abbiamo deciso di utilizzare la lezione di oggi per eseguire un resoconto del lavoro svolto fino ad oggi. In sostanza abbiamo revisionato ciò che abbiamo scritto fino ad ora nella documentazione ed eseguito i test di ogni programma svolto fino ad oggi.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programmi in corso:

- Stretching angolare, 25 (e 50) pulsanti: dobbiamo sistemare i display 7 segmenti;

Programmi da revisionare:

- Tabelline, test di velocità;
- Tiro a vuoto – pulsanti 1 secondo, senior;

Programmi ancora da fare:

- Semplice gioco Simon: 20 pulsanti, 17 livelli;
- Flash test, 5 schemi;
- Anti-flash test;
- Reazione veloce, 10 schemi;

Globalmente in linea.

Programma di massima per la prossima giornata di lavoro

Proseguire con i programmi mancanti

Diario di lavoro

Luogo	Trevano
Data	2019-04-05

Lavori svolti

Abbiamo completato i programmi:

- Semplice gioco Simon: 20 pulsanti, 17 livelli;
- Flash test, 5 schemi;

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programmi in corso:

- Stretching angolare, 25 (e 50) pulsanti: dobbiamo sistemare i display 7 segmenti;

Programmi da revisionare:

- Tabelline, test di velocità;
- Tiro a vuoto – pulsanti 1 secondo, senior;

Programmi ancora da fare:

- Anti-flash test;
- Reazione veloce, 10 schemi;

Globalmente in linea.

Programma di massima per la prossima giornata di lavoro

Proseguire con i programmi mancanti.

Diario di lavoro

Luogo	Trevano
Data	2019-04-10

Lavori svolti

Abbiamo:

- Realizzato tutti i diari di lavoro quotidiani in formato md;
- Avviato la stesura del capitolo “Implementazione”;
- Aggiunto commenti al codice;
- Revisionato la doc modificando la struttura;

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programmi in corso:

- Stretching angolare, 25 (e 50) pulsanti: dobbiamo sistemare i display 7 segmenti;

Programmi da revisionare:

- Tabelline, test di velocità;
- Tiro a vuoto – pulsanti 1 secondo, senior;

Programmi ancora da fare:

- Anti-flash test;
- Reazione veloce, 10 schemi;

Nelle tempistiche ma non rispettando il programma riportato nel Gantt.

Programma di massima per la prossima giornata di lavoro

Proseguire con i programmi mancanti.

Sistema di gioco e simulazione per l’allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-04-12

Lavori svolti

Abbiamo:

- Realizzato il programma 22 (Anti-Flash test);
- Proseguito con la realizzazione della doc realizzando il primo sotto capitolo del capitolo Implementazione;
- Sistemato nuovamente piccoli dettagli nella doc;
- Iniziato l'inserimento dei commenti nel codice;

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programmi in corso:

- Stretching angolare, 25 (e 50) pulsanti: dobbiamo sistemare i display 7 segmenti;

Programmi da revisionare:

- Tabelline, test di velocità;
- Tiro a vuoto – pulsanti 1 secondo, senior;

Programmi ancora da fare:

- Reazione veloce, 10 schemi;

Nelle tempistiche ma non rispettando il programma riportato nel Gantt.

Programma di massima per la prossima giornata di lavoro

Proseguire con i programmi mancanti.

Proseguire la stesura del capitolo "Implementazione".

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-04-17

Lavori svolti

Abbiamo:

- Riempito la tabella dei test nella documentazione, notando che alcuni programmi presentavano qualche imperfezione da sistemare;
- Implementato il programma 23 e sistemati altri;
- Proseguito il capitolo “Implementazione” della doc;
- Proseguito l’inserimento dei commenti nel codice;
- Inserito il requisito 5 nella doc perché lo avevamo lasciato indietro a causa dell’impossibile realizzazione di esso;

Per realizzare il programma 10, ci serve un amplificatore per poter “dire” le tabelline che poi il giocatore dovrà risolvere. Dobbiamo procurarcene uno il prima possibile. Abbiamo trovato questo:

<https://www.sparkfun.com/products/12660>

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Nelle tempistiche ma non rispettando il programma riportato nel Gantt.

Programma di massima per la prossima giornata di lavoro

Sistemare i programmi mancanti.

Proseguire la stesura del capitolo “Implementazione”.

Sistema di gioco e simulazione per l’allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-05-03

Lavori svolti

Abbiamo:

- Dovuto spostare tutto il telaio in un'altra aula perché quella in cui eravamo fino alla lezione scorsa, a partire da lunedì sarà occupata per gli esami di 4°;
- Iniziato la realizzazione della presentazione del progetto;
- Risolto il problema del programma 8;
- Risolto il problema legato al “random” nei programmi 21, 22, 23;
- Realizzato il programma 10;

Problemi riscontrati e soluzioni adottate

- Problema programma 8: non resettava lo stato dei led.
 - Aggiunto l'utilizzo dei metodi resetButtonsState() e resetLeds();
- Problema programmi 21, 22, 23: non veniva estratto un valore random correttamente.
 - Sistemato codice come segue:

```
bool random = true;
int rndpin = 0;
while (random)
{
  rndpin = getRandom(buttonPins, SIZE);
  bool contain = false;
  for (int j = 0; j < 6; j++)
  {
    if (buttonseq[j] == rndpin) {
      contain = true;
    }
  }
  if (!contain)
  {
    break;
  }
}
```

Punto della situazione rispetto alla pianificazione

Essendoci dimenticati di considerare le vacanze scolastiche nel Gantt, risultiamo in ritardo. Ma in tale Gantt, avevamo un margine di 2 settimane.

Riusciamo a mantenere la tempistica per la consegna del progetto senza problemi.

Programma di massima per la prossima giornata di lavoro

Controllare tutti i programmi

Proseguire la stesura del capitolo “Implementazione” e la realizzazione della presentazione.

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-05-08

Lavori svolti

Abbiamo:

- Fatto il punto della situazione dal momento che mancano esattamente 2 settimane alla consegna;
- Proseguito la realizzazione della Presentazione (quasi finito);
- Proseguito la doc:
 - Rimosso capitolo “Design delle interfacce” perché non rilevante per questo progetto;
 - Aggiornato la lista di software utilizzati;
 - Aggiunto i risultati mancanti dei test;
 - Sistemato il capitolo del Gantt preventivo;
 - Realizzato i diagrammi delle classi di “z_main.ino”, “b_function.ino” e “d_programs.ino”.
- Realizzato il codice per il “quit” dai programmi;

```

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();
122         }
123     }
124 }
125 else if (state == LOW && pin == 44)
126 {
127     resetTime = 0;
128 }
129 buttonStatus[index] = state;
130 }
```

- Installato il display LCD definitivo che però non mostra il messaggio;
- Realizzato i commenti di tutti i file sorgente;

Problemi riscontrati e soluzioni adottate

- Il display LCD definitivo non mostra il messaggio (non funziona, probabilmente è dovuto al fatto che questo display ha 4 righe invece quello che avevamo prima (con il quale il messaggio veniva mostrato) ne aveva solamente 2):
 - *Questo problema probabilmente è risolvibile cambiando semplicemente la mappatura delle celle del display o la libreria che lo comanda, ma avendo riscontrato questa problematica alla fine della lezione, non siamo riusciti ancora a risolverla.*

Punto della situazione rispetto alla pianificazione

Pensiamo di riuscire a mantenere la tempistica per la consegna del progetto senza problemi.

Programma di massima per la prossima giornata di lavoro

Controllare definitivamente tutti i programmi e tutte le funzionalità del Batak

Proseguire (e se possibile completare) la stesura del capitolo “Implementazione” e la realizzazione della presentazione.

Sistema di gioco e simulazione per l’allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-05-10

Lavori svolti

Abbiamo:

- Implementato il display LCD a 4 righe;
- Tolti i messaggi di testing;
- Modificato i messaggi per l'utente di ogni programma (sfruttando le 2 righe in più);
- Sistemato la funzione di quit;
- Quasi finito la presentazione (manca gantt consuntivo);
- Sistemato il capitolo del gantt preventivo della doc;
- Iniziato il capitolo 3.7 della doc;
- Aggiunto la limitazione conosciuta dell'assistenza vocale;
- Proseguito il capitolo Implementazione;

Problemi riscontrati e soluzioni adottate

- Il display LCD definitivo non mostra il messaggio (non funziona, probabilmente è dovuto al fatto che questo display ha 4 righe invece quello che avevamo prima (con il quale il messaggio veniva mostrato) ne aveva solamente 2):
 - Modificato l'istanza del display;
 - Modificato il parametro *begin*;
 - Aggiunte le 2 righe in più rispetto al vecchio display;
- Se si usa la funzione *quit* nel menù, il programma crasha:
 - Aggiunto funzione all'interno del controllo.

Punto della situazione rispetto alla pianificazione

Pensiamo di riuscire a mantenere la tempistica per la consegna del progetto senza problemi.

Programma di massima per la prossima giornata di lavoro

Proseguire (e se possibile completare) la stesura del capitolo “Implementazione” e realizzare il gantt consuntivo in modo da inserirlo e completare così la presentazione.

Sistema di gioco e simulazione per l’allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-05-15

Lavori svolti

Oggi abbiamo proseguito il capitolo “Implementazione” (i sotto-capitoli fino al 20 e il 22 e il 24 sono completati) e realizzato nuovi commenti per il codice sistemando anche nuovi piccoli errori sorti dall’analisi più approfondita del codice.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Non escludiamo la possibilità di terminare il lavoro in anticipo di qualche ora.

Programma di massima per la prossima giornata di lavoro

Terminare la stesura del capitolo “Implementazione” e realizzare il gantt consuntivo in modo da inserirlo e completare così la presentazione.

Diario di lavoro

Luogo	Trevano
Data	2019-05-17

Lavori svolti

Oggi abbiamo:

- Realizzato il Gantt consuntivo;
- Completato i sotto-capitoli dell'implementazione 20 e 22;
- Terminato la presentazione;
- Aggiunto il consuntivo alla doc (manca il commento);
- Sistemato il layout della maggior parte dei codici del capitolo implementazione;
- Aggiunta la sitografia alla doc;

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Nelle tempistiche

Programma di massima per la prossima giornata di lavoro

Completare il capitolo 21 e il 23 dell'implementazione

Revisione globale del lavoro prima di consegnarlo

Sistema di gioco e simulazione per l'allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.

Diario di lavoro

Luogo	Trevano
Data	2019-05-22

Lavori svolti

Oggi abbiamo:

- Completato la documentazione;
- Sistemato il pulsante “#” che si incastrava molto spesso;
- Fissato al telaio il pulsante “1” che dall’inizio del progetto era fissato solo con del nastro adesivo;
- Unito i diari per la stampa;
- Stampato sia i diari, sia la documentazione;
- Rilegato il fascicolo;
- Consegnato il progetto;

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Progetto completato entro la data designata

Programma di massima per la prossima giornata di lavoro

Sistema di gioco e simulazione per l’allenamento dei tempi di reazione ed il miglioramento della coordinazione per sportivi basato su Arduino.