ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE a.s. 2020/2021

Indirizzo: ITEC - ELETTRONICA ED ELETTROTECNICA ARTICOLAZIONE ELETTRONICA

Elaborato: ELETTROTECNICA ED ELETTRONICA E SISTEMI AUTOMATICI

Candidato: MACCIONE Giuseppe classe 5 A ELETTRONICA

Il candidato deve progettare un sistema automatico che sia capace di condurre una partita a scacchi con un essere umano, tramite rilevazione delle mosse dell'avversario, elaborazione e spostamento dei pezzi sulla scacchiera.

In particolare il candidato, fatte le ipotesi aggiuntive che ritiene opportune, deve:

1. definire le specifiche del progetto;
2. fornire uno schema a blocchi del sistema di controllo utilizzando un microcontrollore o altro sistema programmabile di sua conoscenza;
3. dimensionare le interfacce necessarie al condizionamento dei segnali provenienti dai sensori, descrivere una possibile modalità di visualizzazione dei dati acquisiti e progettare le interfacce di segnalazione e di potenza necessarie;
4. fornire uno schema elettrico e/o di cablaggio;
5. sviluppare un algoritmo di gestione delle acquisizioni, della visualizzazione e segnalazione dei dati, dei blocchi di potenza;
6. fornire il codice del programma;
7. eventualmente fornire foto della simulazione e/o realizzazione del prototipo.

Scacchiera Automatica

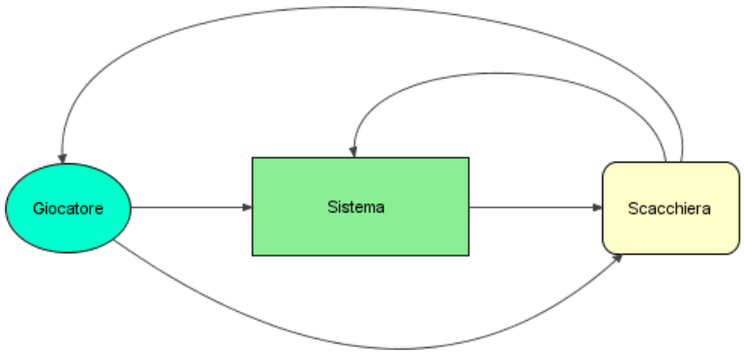
Specifiche:

È un sistema automatico il cui obiettivo è quello di avere una scacchiera automatica interattiva, capace di condurre delle partite di scacchi con un essere umano. Permette di giocare contro un computer capace di rilevare le mosse compiute dal giocatore e compiere una contromossa per mezzo di un attuatore meccanico. Inoltre è possibile decidere alcune modalità di gioco e visualizzare importanti informazioni legate alla partita.

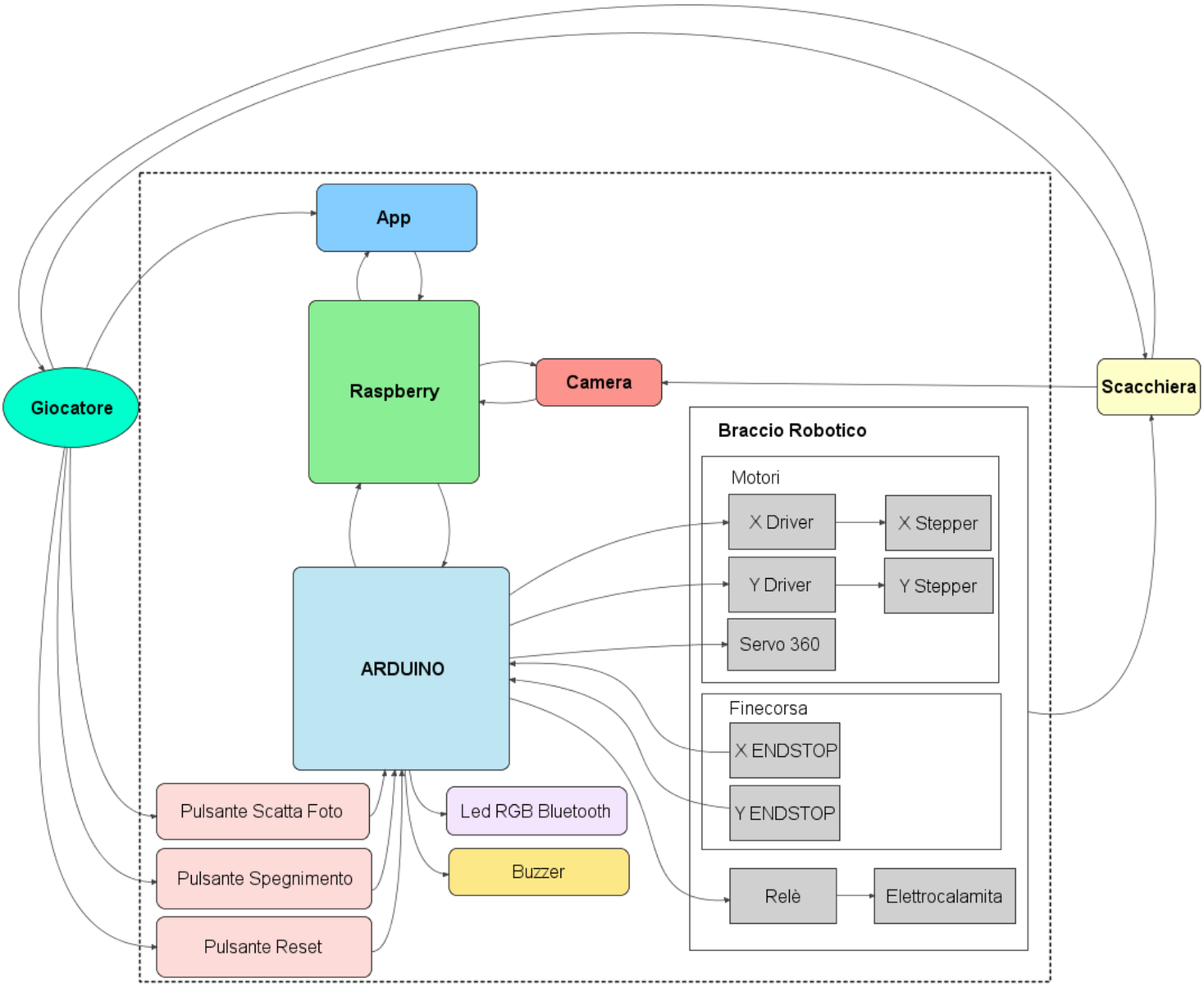
Questa sistema è composto da un'unità di elaborazione centrale, rappresentata da Raspberry, alla quale sono connessi:

- Un dispositivo di input ottico utilizzato per la visione della scacchiera;
- Un Applicazione che permette al giocatore di poter scegliere le modalità di gioco tra cui il colore dei pezzi e la difficoltà del computer avversario, ed inoltre visualizzare lo stato corrente della scacchiera e il vantaggio;
- Un microcontrollore che permette d'interfacciare l'elaboratore con l'impianto di potenza ed alcuni dispositivi di comando e di segnalazione, come pulsanti, diodi led e speaker;

Schema a Blocchi ridotto:



Schema a Blocchi esteso:

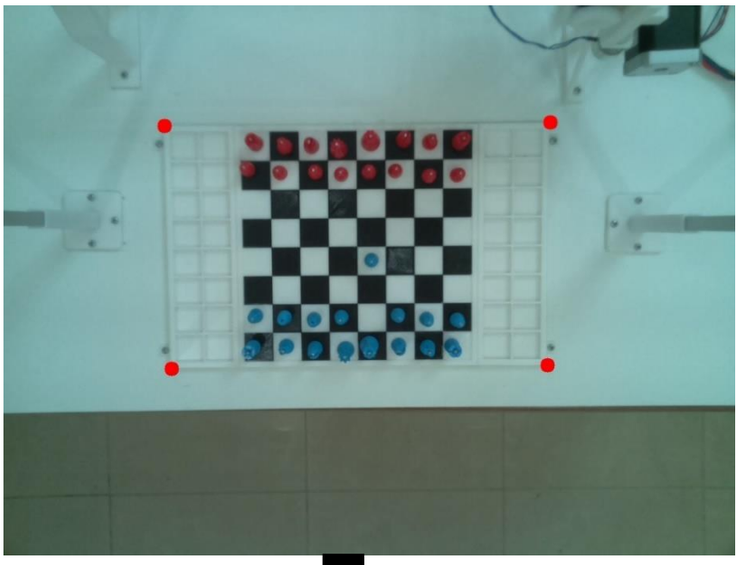


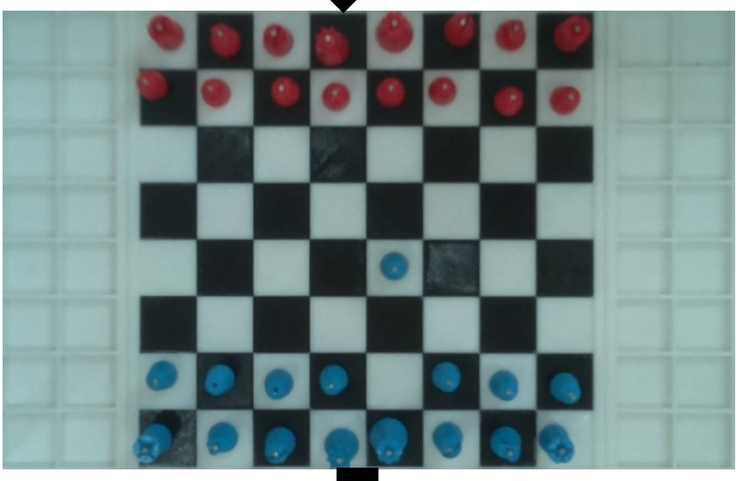
Come avviene la visione delle Mosse?

Passo 1:

Il primo passo per poter visualizzare la mossa ed analizzarla, consiste nell'effettuare una fotografia "grezza" della scacchiera che verrà successivamente ritagliata, ridimensionata e convertita in una matrice i cui elementi saranno stabiliti in base alla presenza o meno di pedine:

- **2** -> *Pedina Rossa*
- **1** -> *Pedina Blu*
- **0** -> *Casella Vuota*





LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

Le pedine **blu** corrispondono ai **bianchi**.

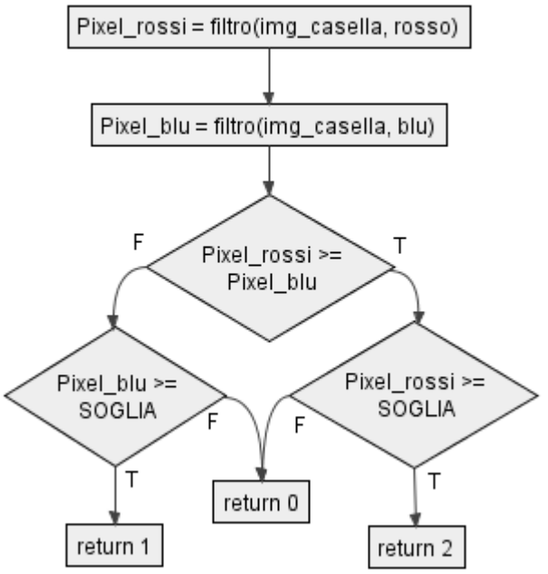
Le pedine **rosse** corrispondo ai **neri**.

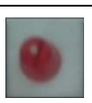
Sono stati scelti questi colori per ottenere maggiore contrasto con la scacchiera e rendere così la visione più accurata.

La fase del ritaglio e del ridimensionamento avviene tramite l'utilizzo di 4 punti preimpostati (ricavati manualmente dall'immagine) indicanti i bordi dell'immagine da ritagliare

Dopo il **ritaglio** e il **ridimensionamento**, l'immagine viene ritagliata ulteriormente in tante immagini quante sono le caselle ed analizzando singolarmente ogni immagine corrispondente a ogni casella, è possibile capirne il contenuto.

Ciò avviene mediante l'utilizzo di un **algoritmo di riconoscimento dei colori** mediante un filtro per il colore ROSSO e uno per il colore BLU. Questo restituisce il numero di pixel ROSSI e BLU presenti nell'immagine. Queste due quantità di pixel vengono confrontate e viene considerato solamente il colore con il numero di pixel maggiore. Successivamente se il numero di pixel considerato è in grado di superare una SOGLIA preimpostata di 500 pixel, allora vuol dire che probabilmente è presente una pedina di quel colore. In caso contrario vuol dire che la casella è vuota.



INPUT	Filtro Blu	Filtro Rosso	N° Pixel Blu	N° Pixel Rossi	OUTPUT
			**0**	**1200**	**2**
			**1200**	**0**	**1**
			**0**	**0**	**0**

Passo 2:

Nel secondo passo bisogna confrontare la nuova matrice della scacchiera ottenuta tramite la visione, alla matrice della posizione precedente che corrisponde alla scacchiera del programma.

Il programma possiede un sua matrice indicante la scacchiera i cui elementi sono rappresentati sotto forma di stringhe: “p”,”n”,”b”,”r”,”q”,”k” per i neri, “P”,”N”,”B”,”R”,”Q”,”K” per i bianchi, “ ” per la casella vuota.

A inizio partita la matrice del programma ha un valore preimpostato che corrisponde alla posizione di partenza nota per regolamento, e a ogni mossa giocata viene aggiornata in maniera tale da permette il confronto sempre con la posizione precedente alla mossa del giocatore.

Prima del confronto bisogna adattare la matrice del programma (con le stringhe) alla matrice ottenuta dalla visione (con i numeri), generando un ulteriore matrice convertendo le stringhe in numeri come segue:

- dove è presente una lettera maiuscola, impostare 1
- dove è presente una lettera minuscola, impostare 2
- dove è presente una stringa vuota, impostare 0

matrice del programma in forma di stringhe

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“P”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”

CONVERSIONE IN FORMA NUMERICA

matrice del programma convertita in forma numerica

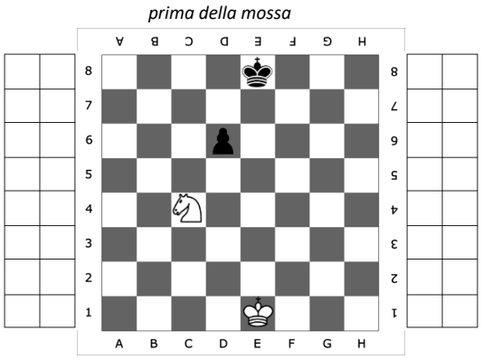
LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

Il confronto consiste nel verificare se la matrice del programma, dopo la conversione, coincide con la matrice ottenuta dalla visione, e in caso contrario bisogna definire il tipo di cambiamento avvenuto.

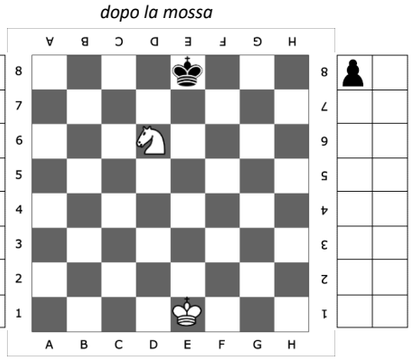
- “+” -> indica che è stata aggiunta una pedina
- “-“ -> indica che è stata rimossa una pedina
- “/” -> indica che è presente una pedina ma è avvenuto un cambio di colore

Coordinata Posizione Precedente	**Coordinata Nuova Posizione**	**Cambiamento**
0	1 o 2	“+”
1 o 2	0	“-“
1 o 2	1 o 2	“/”

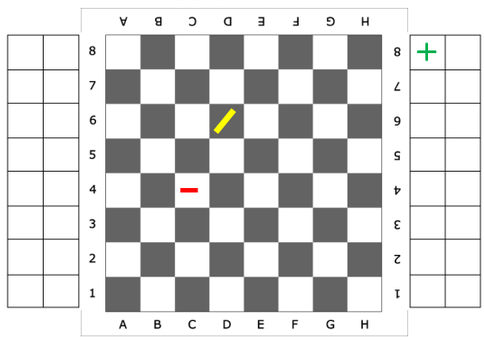
prima della mossa



dopo la mossa



cambiamenti avvenuti



<u>Al tipo di cambiamento è associata la coordinata in cui è avvenuto.</u>

Analizzando il tipo, la quantità e le coordinate in cui sono avvenuti i cambiamenti è possibile dedurre la mossa giocata dal giocatore. Dopo aver validato la mossa ottenuta, si procede con la modifica alla matrice della scacchiera del programma.

La validazione delle mosse è divisa in due fasi: nella prima fase si valuta se corrisponde a una delle tre categorie di mosse possibili, nella seconda fase si valuta con l’ausilio di un motore scacchistico se dal punto di vista del regolamento è una mossa valida. Nel caso in cui una mossa viene ritenuta non valida il programma procede senza far avvenire nessuna modifica ed emetterà un suono associato a questo tipo di situazione.

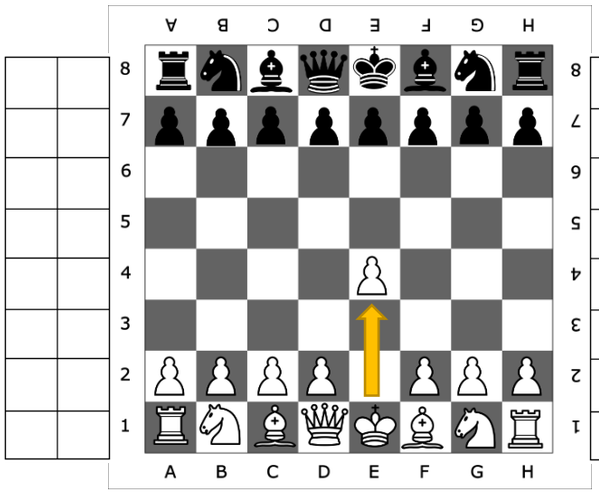
Le mosse vengono divise in 3 categorie:

- ***Spostamento*** -> sono presenti un “+” e un “-“;
- ***Presa*** -> sono presenti un “+”, un “-“ e un “/”;
- ***Arrocco*** -> sono presenti due “+” e due “-“;

Seguono alcuni esempi dei tre tipi di mosse

Spostamento:

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“P”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”



CONVERSIONE IN FORMA NUMERICA

VISIONE DELLA SCACCHIERA

LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

CONFRONTO

LEFT		GRID								RIGHT	
						+					
						-					

e2e4

MODIFICA DELLA SCACCHIERA

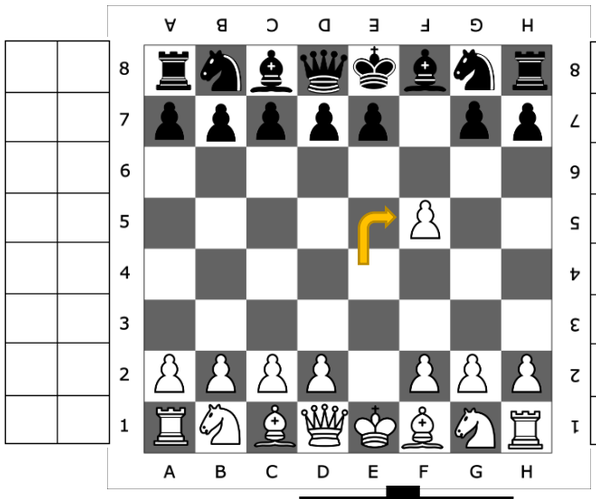
LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”

In questo esempio di mossa i cambiamenti ottenuti sono: un “+” e un “-“ quindi si tratta di uno spostamento da e2 (cordinata da cui è stato rimosso il pezzo) a e4 (coordinata in cui è stato aggiunto il pezzo). Dopo aver validato la mossa si procede con la modifica della scacchiera del programma impostando “ “ in corrispondenza del “-“ e, il pezzo che inizialmente era in “-“, in “+”.

<u>Nel caso in cui sia possibile la promozione il pezzo impostato sarà “q” o “Q” (regina).</u>

Presa:

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“p”	“ ”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“p”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”



CONVERSIONE IN FORMA NUMERICA

VISIONE DELLA SCACCHIERA

LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	0	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

LEFT		GRID								RIGHT	
0	1	2	2	2	2	2	2	2	2	2	0
0	0	2	2	2	2	2	0	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

CONFRONTO

LEFT		GRID								RIGHT	
										+	
							/				
						-					

e4f5

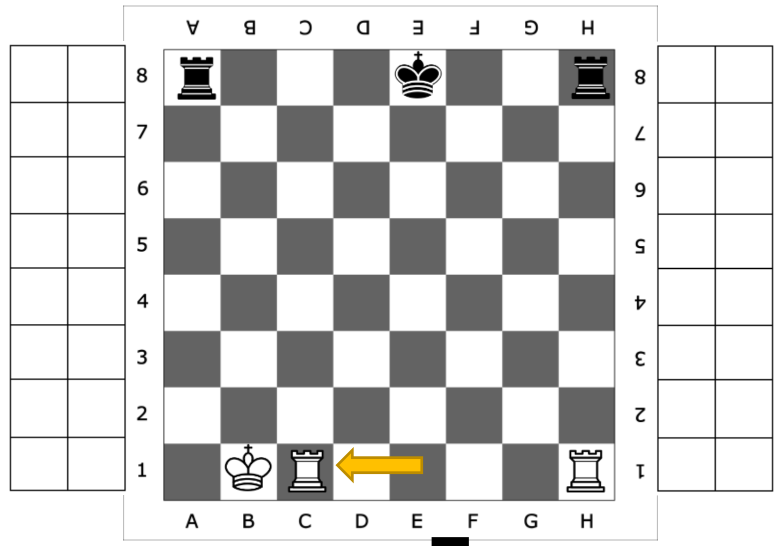
MODIFICA DELLA SCACCHIERA

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“p”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“p”	“ ”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”

In questo esempio di mossa i cambiamenti ottenuti sono: un “+”, un “-“ e un “/” quindi è il caso di una cattura da e4 (coordinata da cui è stato rimosso il pezzo) a f5 (coordinata in cui è avvenuto il cambio di colore). Dopo aver validato la mossa si procede con la modifica della scacchiera del programma impostando “ “ in corrispondenza del “-“ e, il pezzo che inizialmente era in “-“ (attaccante), in “/”, e il pezzo che era inizialmente in “/” (pedina che viene presa), in “+”.

Arrocco:

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“ ”	“ ”	“ ”	“k”	“ ”	“ ”	“r”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“R”	“ ”	“ ”	“ ”	“K”	“ ”	“ ”	“R”	“ ”	“ ”



CONVERSIONE IN FORMA NUMERICA

VISIONE DELLA SCACCHIERA

LEFT		GRID								RIGHT	
0	0	2	0	0	0	2	0	0	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	1	0	0

CONFRONTO

LEFT		GRID								RIGHT	
0	0	2	0	0	0	2	0	0	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	1	0	0

LEFT		GRID								RIGHT	
		-	+	+		-					

e1c1

MODIFICA DELLA SCACCHIERA

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“ ”	“ ”	“ ”	“k”	“ ”	“ ”	“r”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“K”	“R”	“ ”	“”	“ ”	“ ”	“R”	“ ”	“ ”

In questo esempio di mossa i cambiamenti ottenuti sono: due “+” e due “-“ quindi vuol dire che si tratta di un arrocco. L’arrocco può essere di 4 quattro tipi: arrocco corto bianco e nero, arrocco lungo bianco e nero. Per capire che tipo di arrocco è avvenuto si confrontano le coordinate in cui sono avvenuti i cambiamente con le coordinate dei vari tipi di arrocchi, note per regolamento. Ogni arrocco ha una mossa identificativa: arrocco corto bianco-> “e1g1”, arrocco lungo bianco-> “e1c1”, arrocco corto nero-> “e8g8”, arrocco lungo nero-> “e8c8”. In questo caso ci si trova davanti ad un arrocco lungo bianco e quindi la scacchiera viene modificata di conseguenza.

Come avviene la contromossa?

La mossa avversaria è generata dal motore scacchistico Stockfish il quale, ricevendo una posizione scacchistica, restituisce, in base al suo livello, la mossa migliore a cui poi seguirà la modifica della scacchiera del programma e della scacchiera reale.

La posizione da impostargli segue la notazione **FEN**, che permette di indentificare una posizione scacchistica tramite una stringa. È quindi necessario convertitre la scacchiera del programma secondo la notazione FEN, considerando solamente la griglia ed escludendo il lato destro e sinistro.

GRID							
“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”
“p”	“p”	“p”	“p”	“p”	“p”	“p”	“p”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”
“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”

CONVERSIONE IN FEN

“rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR”

La modifica della scacchiera fisica avviene medianete l’utilizzo di un Braccio Robotico gestito da Arduino Nano. Il programma Raspberry converte la mossa ottenuta da Stockfish in una o più stringhe che indicano li spostamenti da far effetture al Braccio Robotico per ottenere la modifica della scacchiera voluta, e le invia tramite Seriale ad Arduino che dopo averle interpretate adeguatamente esegue la modifica.

“M-syx-syx-p”

Keyword | Start Cord | End Cord | Piece

La Stringa è composta da 4 sezioni:

- **Keyword**-> “M” è una semplice *parola chiave* che ha lo scopo di far comprendere ad Arduino che ciò che la segue indica uno spostamento da effettuare
- **Start Cord** -> è la coordinata a cui deve muoversi il braccio e *prendere* la pedina
- **End Cord** -> è la coordinata a cui deve muoversi il braccio e *rilasciare* la pedina
- **Piece** -> indica la *pedina* che deve essere mossa

Una coordinata è rappresentata da tre numeri:

- **S** -> indica la *sezione* di scacchiera interessata
- **Y** -> indica la *y* della coordianta
- **X** -> indica la *x* della coordinata

Ulteriori utilizzi di Stockfish

Il motore scacchistico, oltre che generare una contromossa e validare le mosse del giocatore, permette anche di regolare il suo livello di difficoltà in un intervallo da 1 a 20, e ottenere una valutazione della posizione scacchistica corrispondente a un numero relativo al vantaggio associato al bianco e al nero.

Segue un esempio di contromossa

GRID							
“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”
“p”	“p”	“p”	“p”	“p”	“p”	“p”	“p”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”
“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”

CONVERSIONE IN FEN

“rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR”

INVIO A STOCKFISH

“M-114-134-P”

d7d5

CALCOLO SPOSTAMENTI

Mossa migliore

MODIFICA DELLA SCACCHIERA DEL PROGRAMMA

INVIO STRINGHE AD ARDUINO

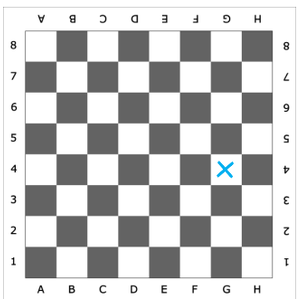
MODIFICA DELLA SCACCHIERA REALE

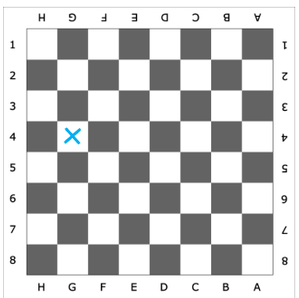
LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“ ”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“p”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”

Gioco con i Neri

Nel caso in cui si scelga di giocare con i neri sarà il computer a compiere la prima mossa.

Per quanto concerne al gioco con i bianchi la visione delle mosse e la contromossa avviene come spiegato precedentemente, al contrario del gioco con i neri in cui sono necessarie alcune modifiche.
Come primo cambiamento, il giocatore dovrà posizionare le pedine sulla scacchiera secondo la posizione di partenza corrispondente a una partita giocata con i neri. In questa posizione la scacchiera del nero risulta ruotata di 180 gradi rispetto alla scacchiera del bianco.
Diventa quindi necessario, a seguito alla visione della scacchiera, una rotazione di 180 gradi della matrice ottenuta, in maniera tale da poter eseguire i processi precedentemente spiegati.
Successivamente quando bisognerà attuare la modifica della scacchiera reale sarà necessario, dopo aver generato le stringhe indicanti li spostamenti, modificarne le coordinate in relazione alla matrice ruotata nuovamente di 180 gradi.

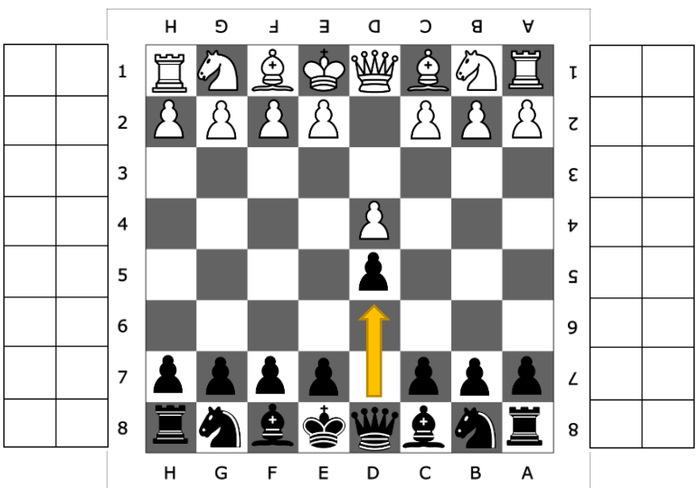




Con queste due immagini è possibile vedere che posizione occupa nella matrice la coordinata g4 prima e dopo la rotazione di 180 gradi.
Il calcolo che viene eseguito è:
x_ruotata = 7 - x
y_ruotata = 7 - y

Segue un esempio di gioco con il nero

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”



CONVERSIONE IN FORMA NUMERICA

VISIONE DELLA SCACCHIERA

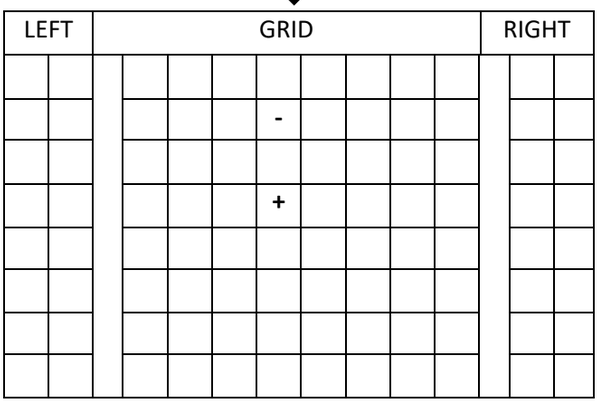
LEFT		GRID								RIGHT	
0	0	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	0	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	2	2	2	0	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0

ROTAZIONE DELLA SCACCHIERA DI 180 GRADI

LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0

CONFRONTO

LEFT		GRID								RIGHT	
0	0	2	2	2	2	2	2	2	2	0	0
0	0	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0



d7d5

LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“ ”	“p”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“p”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”

CONVERSIONE IN FEN

“rnbqkbnr/ppp1pppp/8/3p4/3P4/8/PPP1PPPP/RNBQKBN

INVIO A STOCKFISH

“M-114-134-P”

CALCOLO SPOSTAMENTI

Mossa migliore

e7e5

MODIFICA DELLA SCACCHIERA DEL PROGRAMMA

MODIFICA DELLE COORDINATE

“M-163-143-P”

INVIO STRINGHE AD ARDUINO

MODIFICA DELLA SCACCHIERA REALE

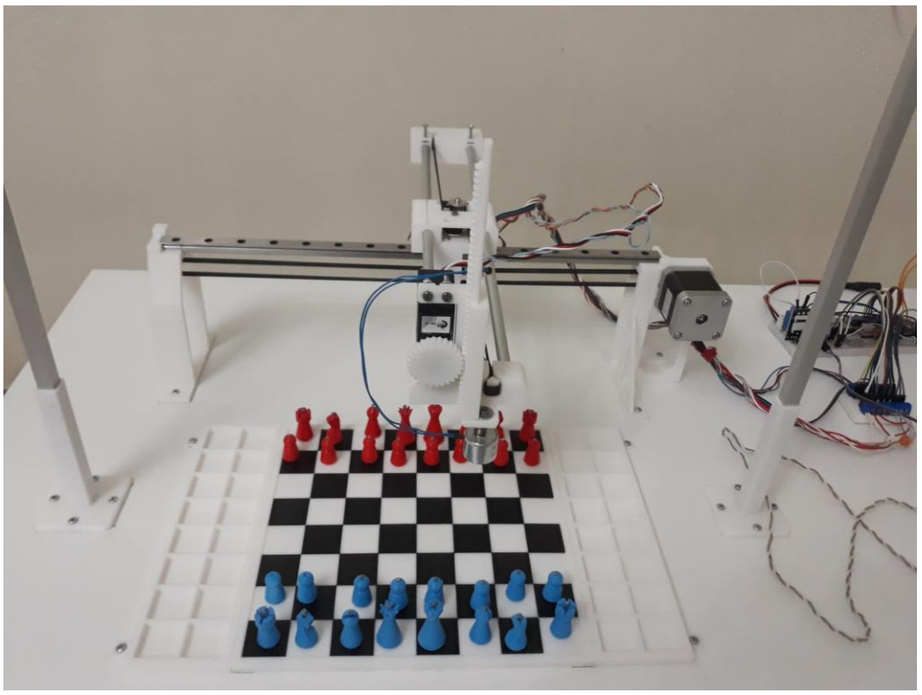
LEFT		GRID								RIGHT	
“ ”	“ ”	“r”	“n”	“b”	“q”	“k”	“b”	“n”	“r”	“ ”	“ ”
“ ”	“ ”	“p”	“p”	“p”	“ ”	“ ”	“p”	“p”	“p”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“p”	“p”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“P”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
“ ”	“ ”	“P”	“P”	“P”	“P”	“ ”	“P”	“P”	“P”	“ ”	“ ”
“ ”	“ ”	“R”	“N”	“B”	“Q”	“K”	“B”	“N”	“R”	“ ”	“ ”

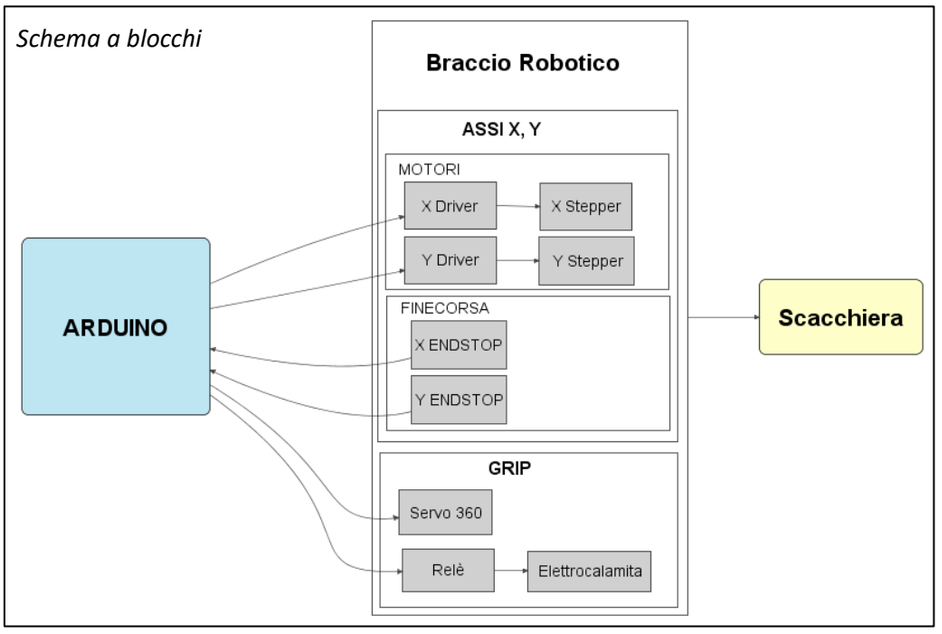
Come avviene lo spostamento delle Pedine?

Lo spostamento delle pedine avviene con l'utilizzo di un attuatore meccanico rappresentato da un Braccio Robotico gestito da Arduino Nano, il quale è connesso all'elaboratore tramite collegamento seriale che rende possibile lo scambio reciproco di informazioni.

Per quanto riguarda lo spostamento delle pedine, come descritto precedentemente, Arduino riceve le stringhe indicanti gli spostamenti che, dopo essere state interpretate adeguatamente, danno il via allo spostamento delle pedine tramite il Braccio Robotico.

Braccio Robotico:



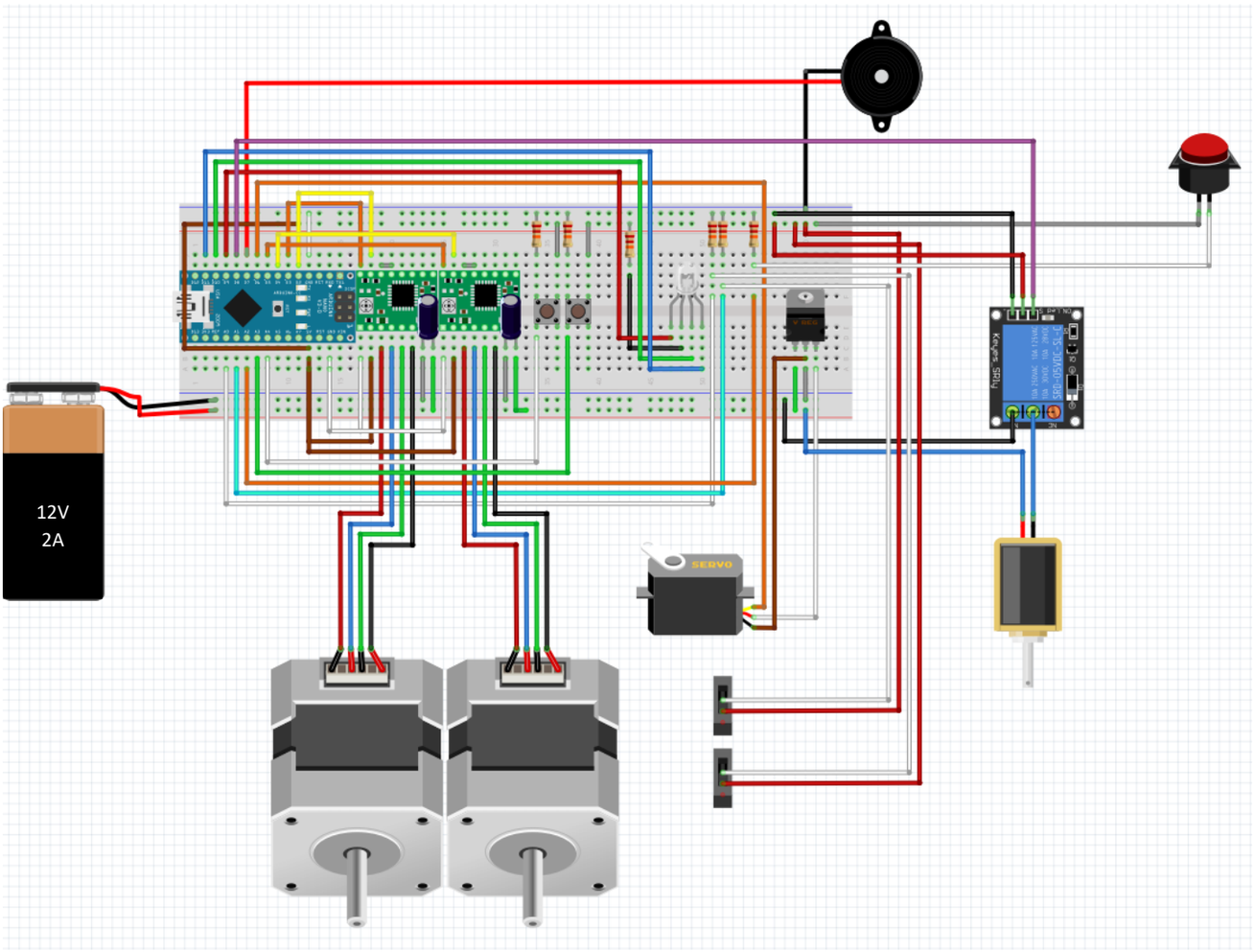


Il Braccio robotico è una macchina CNC capace di raggiungere ogni casella presente sulla scacchiera per mezzo di due Stepper Motor che gli consentono il movimento negli assi X ed Y. È inoltre in grado di prendere e rilasciare le pedine per mezzo di un attuatore lineare composto da un Servomotore a rotazione continua, che gli consente il movimento nell'asse Z, ed un'elettrocalamita che gli consente l'aggangio e lo sgancio delle pedine.

Per gli assi X ed Y sono presenti due finecorsa che permettono al sistema di comprendere quando i due assi hanno raggiunto la loro massima escursione di movimento.

Nel codice Arduino è associato un valore di step per ogni coordinata della scacchiera

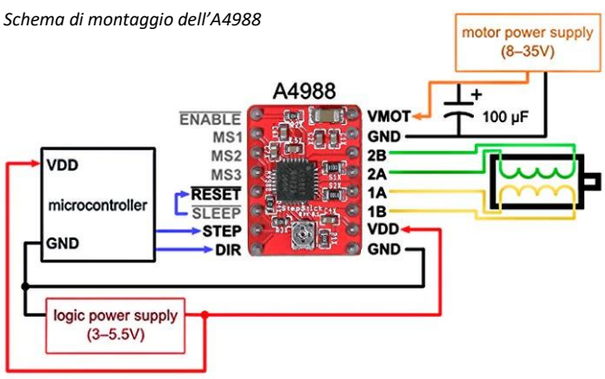
Schema di montaggio



Pilotaggio degli Stepper:

Il pilotaggio degli Stepper Motor avviene tramite i driver A4988.

Schema di montaggio dell'A4988



Il driver presenta due alimentazioni diverse:

- 12V -> tensione utilizzata per alimentare gli avvolgimenti del motore
- 5V -> tensione utilizzata per alimentare il circuito del driver

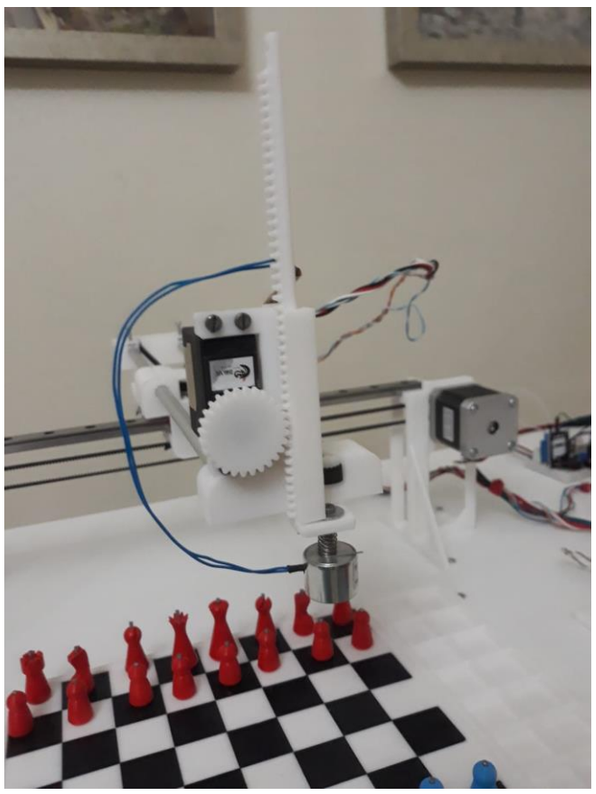
Possiede 4 pin di OUTPUT a cui sono collegati gli avvolgimenti del motore.

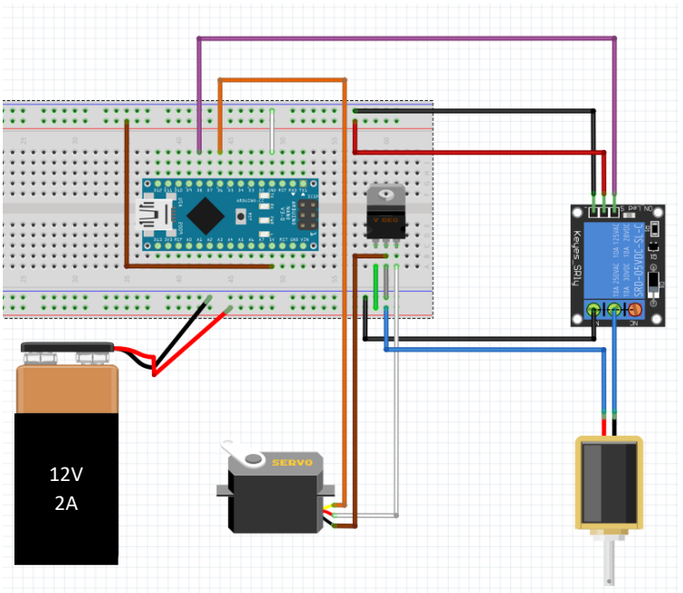
Possiede 2 pin di INPUT digitali collegati ad Arduino, in particolare:

- DIR -> in base a se alimentato o meno, definisce il verso di rotazione del motore
- STEP -> se alimentato fa eseguire uno step al motore nella direzione impostata.

Per far eseguire un determinato numero di passi angolari allo stepper bisogna applicare al pin STEP un segnale PWM con duty cycle 50%, il cui numero di impulsi corrisponde al numero di steps.

Grip:





Pilotaggio del Servomotore a rotazione continua:

Il Servomotore a rotazione continua, diversamente dal Servomotore a 180 gradi, non è in grado di determinare la posizione del rotore. Per pilotare questo tipo di motori bisogna regolare il senso di rotazione ed il tempo in cui sono in azione.

Necessita di un alimentazione di 5V, quindi è stato utilizzato un regolatore di tensione L7805 per abbasare la tensione dalla linea di alimentazione da 12V a 5V.

Aggancio e Sgancio delle pedine:

L'aggancio e lo sgancio avviene tramite un elettrocalamita alimentata a 12V, la quale viene attivata o disattivata per mezzo di un relè comandato da Arduino.

È possibile ottenere l'aggancio delle pedine tramite l'elettromagnete, grazie ad un materiale metallico presente sull'estremità delle pedine.

Come sono stati ricavati i passi angolari per ogni coordinata?

Punto 1:

Il primo punto per ottenere il numero di passi angolari per ogni coordinata, consiste nell'ottenere il numero passi angolari corrispondenti alla massima escursione di movimento per gli assi X ed Y. Per raggiungere lo scopo, è stata effettuata in fase di realizzazione, la calibratura del Braccio Robotico.

```
void calibration(){
  x_total_steps=0;
  y_total_steps=0;
  digitalWrite(X_DIR,false);
  digitalWrite(Y_DIR,false);

  while(digitalRead(X_ENDSTOP)){
    digitalWrite(X_STEP,HIGH);
    delayMicroseconds(STEP_DELAY);
    digitalWrite(X_STEP,LOW);
    delayMicroseconds(STEP_DELAY);
    x_total_steps++;
  }
  while(digitalRead(Y_ENDSTOP)){
    digitalWrite(Y_STEP,HIGH);
    delayMicroseconds(STEP_DELAY);
    digitalWrite(Y_STEP,LOW);
    delayMicroseconds(STEP_DELAY);
    y_total_steps++;
  }
  Serial.print("x_total_steps: ");
  Serial.println(x_total_steps);
  Serial.print("y_total_steps: ");
  Serial.println(y_total_steps);
}
```

Esegue uno step per l'asse X finchè non arriva al finecorsa corrispondente, e ad ogni ciclo incrementa di uno *x_total_steps*

Esegue uno step per l'asse Y finchè non arriva al finecorsa corrispondente, e ad ogni ciclo incrementa di uno *y_total_steps*

Visualizza sul monitor seriale

Monitor seriale

```
x_total_steps: 1646
y_total_steps: 1400
```

Punto 2:

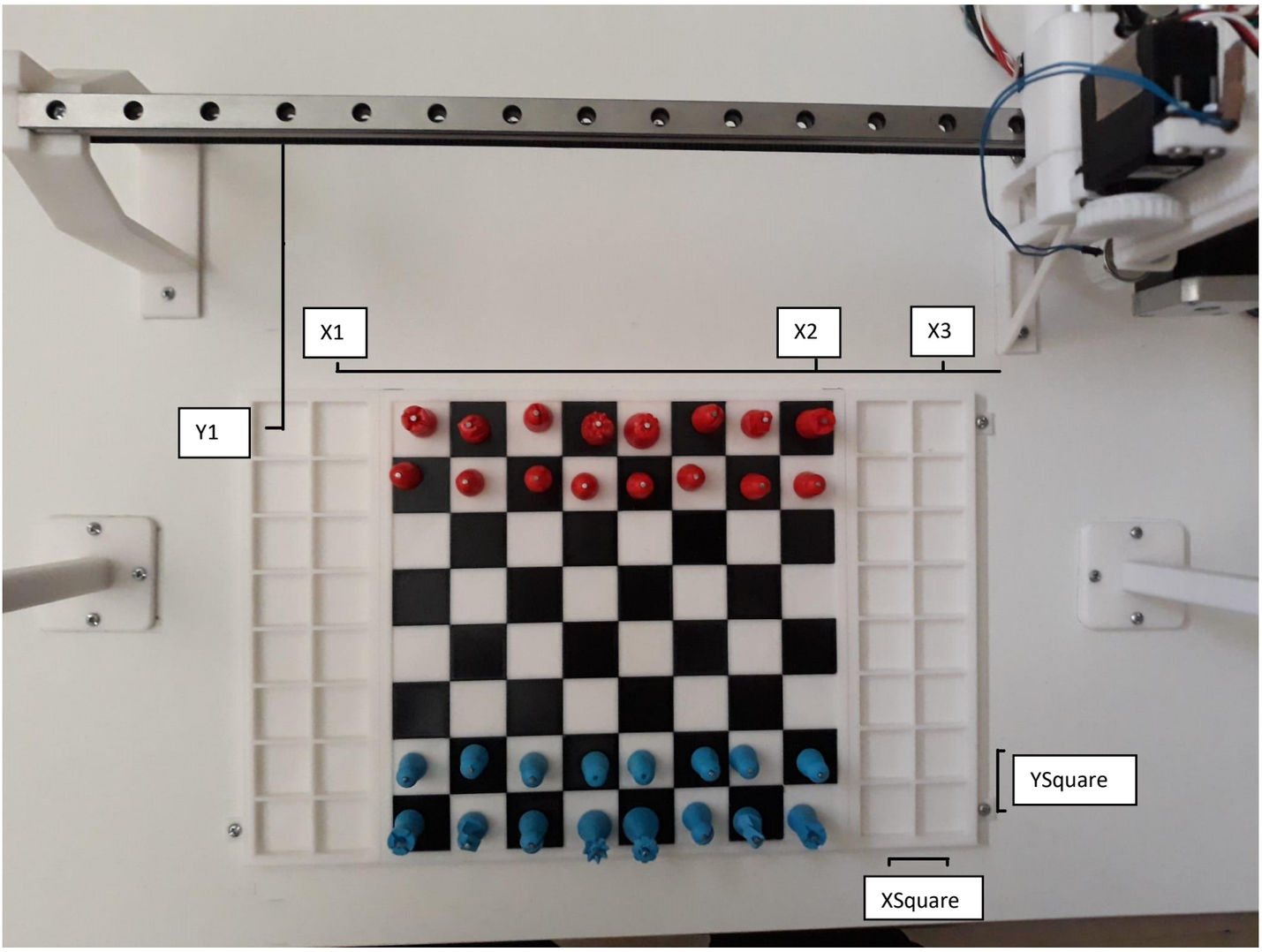
Una volta ottenuti i valori, conoscendo le dimensioni del braccio e della scacchiera è possibile, tramite delle proporzioni, calcolare dei valori di step che corrispondono ad alcune sezioni della scacchiera, con i quali poi si potrà comporre la matrice contenente i valori di step per ogni casella.

Larghezza = 33.3cm
Altezza = 27.5cm
Proporzione per l'asse X -> 33.3 : 1646 = lunghezza : step
Proporzione per l'asse Y -> 27.5 : 1400 = lunghezza : step

	LUNGHEZZE	STEPS
X1	32.6cm	1615
X2	26.3cm	1300
X3	5cm	250
Y1	3.14cm	160
X square	2.5cm	123
Y square	2.5cm	127

Matrice contenente i valori di step per ogni coordinata

```
steps[3][8][8][2] = {
 /*----LEFT----*/
 {
  {{X1-(Xsquare*0), Y1+(Ysquare*0)},   {X1-(Xsquare*1), Y1+(Ysquare*0)}},
  . . .
  {{X1-(Xsquare*0), Y1+(Ysquare*7)},   {X1-(Xsquare*1), Y1+(Ysquare*7)}}
 },
 /*----GRID----*/
 {
  { {X2-(Xsquare*0), Y1+(Ysquare*0)}, . . . {X2-(Xsquare*7), Y1+(Ysquare*0)},  },
  . . .
  { {X2-(Xsquare*0), Y1+(Ysquare*7)}, . . . {X2-(Xsquare*7), Y1+(Ysquare*7)},  }
 },
 /*----RIGHT----*/
 {
  {{X3-(Xsquare*0), Y1+(Ysquare*0)},   {X3-(Xsquare*1), Y1+(Ysquare*0)}},
  . . .
  {{X3-(Xsquare*0), Y1+(Ysquare*7)},   {X3-(Xsquare*1), Y1+(Ysquare*7)}}
 }
}
```



Aquisizione della stringa:

```
if(start_byte == 'M'){
   get_and_move();
}
```

```
/*ottiene la mossa disponibile dal buffer seriale e la fa compiere al braccio*/
void get_and_move(){
  delay(50);
  Serial.read();//skip the '-'

  int start_section = Serial.read() - '0';
  int start_row = Serial.read() - '0';
  int start_column = Serial.read() - '0';

  Serial.read();//skip the '-'

  int end_section = Serial.read() - '0';
  int end_row = Serial.read() - '0';
  int end_column = Serial.read() - '0';

  Serial.read();//skip the '-'

  char piece = Serial.read();

  move(start_section,start_row,start_column, end_section,end_row,end_column, piece);
}
```

Start Cord

End Cord

Piece

Dopo aver letto il primo valore dal buffer seriale, nel caso in cui sia uguale a “M”, si procede con l’ottenimento dei successivi valori della stringa, che corrispondono alle coordinate e al tipo pedina. Successivamente questi valori vengono usati come parametri nel richiamo della funzione *move* che serve ad eseguire lo spostamento della pedina.

Movimento:

La funzione *move* serve a far avvenire lo spostamento di una pedina da una coordinata all'altra, facendo posizionare inizialmente l'estremità del braccio alla prima coordinata dove avverrà la presa della pedina, successivamente trasportare la pedina alla seconda coordinata dove ne avverrà il rilascio.

```
void move(int start_section, int start_row, int start_column, int end_section, int end_row, int end_column, char piece){
  int start_steps_x = steps[start_section][start_row][start_column][0];
  int start_steps_y = steps[start_section][start_row][start_column][1];
  int end_steps_x = start_steps_x - steps[end_section][end_row][end_column][0];
  int end_steps_y = start_steps_y - steps[end_section][end_row][end_column][1];

  step(false, false, start_steps_x, start_steps_y);
  delay(500);

  grip(true,piece,start_section);
  delay(500);

  if((end_steps_x<0) && (end_steps_y<0)){
    step(false, false, end_steps_x*-1, end_steps_y*-1);
  }else if(end_steps_x<0){
    step(false, true, end_steps_x*-1, end_steps_y);
  }else if(end_steps_y<0){
    step(true, false, end_steps_x, end_steps_y*-1);
  }else {
    step(true, true, end_steps_x, end_steps_y);
  }
  delay(500);

  grip(false,piece,end_section);
  delay(500);

  step(true, true, end_total_steps_x, end_total_steps_y);
  delay(1000);
}
```

Valori di step per X ed Y necessari per raggiungere la coordinata di inizio e fine dello spostamento

Raggiungimento coordinata di inizio

Presa della pedina

Raggiungimento coordinata di fine.

In base al segno del valore di step si decide in che direzione muoversi

Rilascio della pedina

Ritorno alla posizione iniziale

Nella prima sezione i valori di step per x e y della prima coordinata sono stati ottenuti dalla matrice di *steps* inserendone i dati come indici.

Per i valori della seconda coordinata si esegue la differenza tra i valori di step della prima coordinata e i valori di step ricavati da *steps* inserendo come indici i dati della seconda coordinata. La differenza è necessaria perchè il braccio dovrà muoversi dalla sua posizione corrente, che corrisponde alla prima coordinata, alla seconda coordinata.

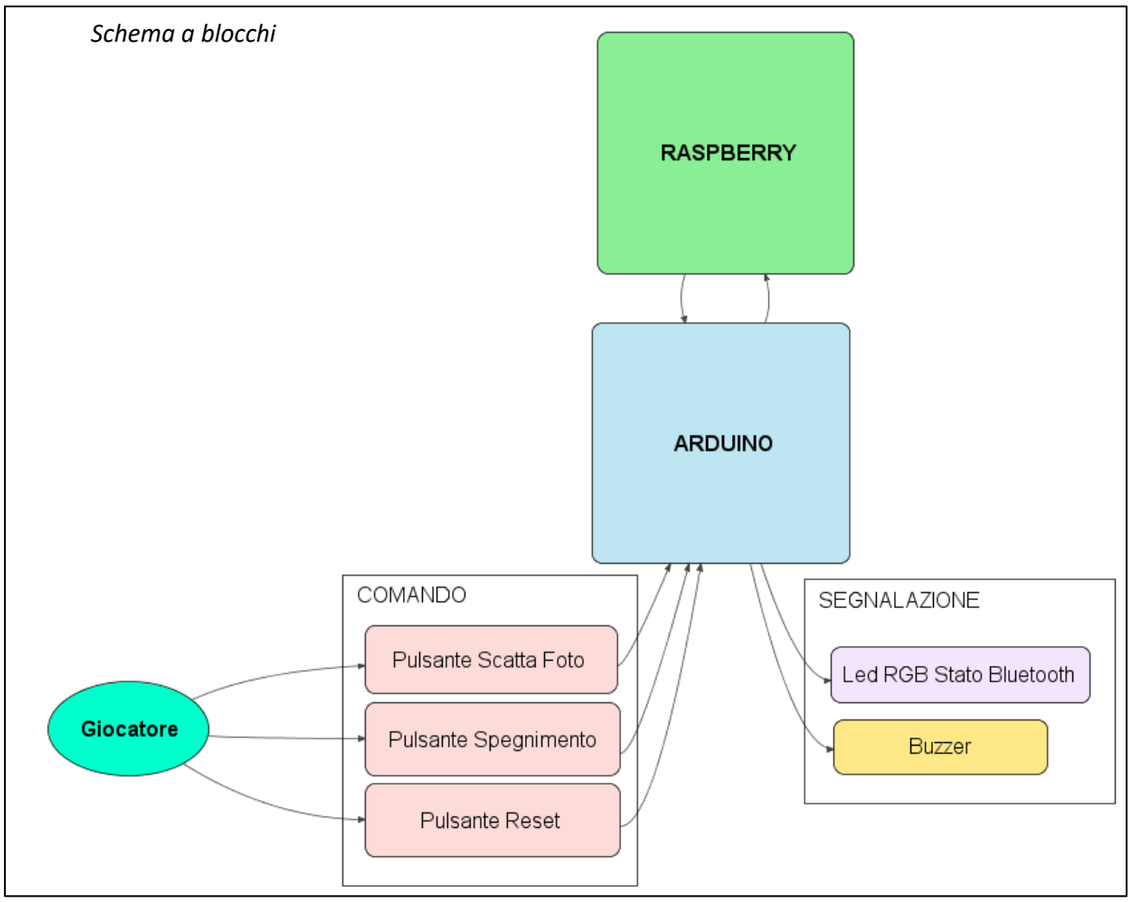
La differenza ottenuta potrebbe assumere valori negativi, in tal caso è necessario renderla positiva e invertire il senso di rotazione del motore interessato.

Dispositivi di Comando e Segnalazione

Il sistema presenta diverse modalità di comando e segnalazione, gestite attraverso due dispositivi i quali instaurano una connessione con l'elaboratore permettendo uno scambio reciproco di informazioni. Questi dispositivi sono Arduino e l'Applicazione.

Arduino:

Schema a blocchi



La comunicazione instaurata tra l'elaboratore e Arduino avviene tramite collegamento USB che permette una connesione di tipo Seriale.

Arduino aquisisce informazioni dai suoi dispositivi di input, rappresentati da dei pulsanti, ed invia a Raspberry delle stringhe associate al tipo di input ricevuto.

La segnalazione avviene in seguito alla ricezione di alcune stringhe, che indicano il tipo di segnalazione voluta, tramite l'utilizzo di un diodo led RGB ed un Buzzer.

Tabella relativa allo scambio di stringhe

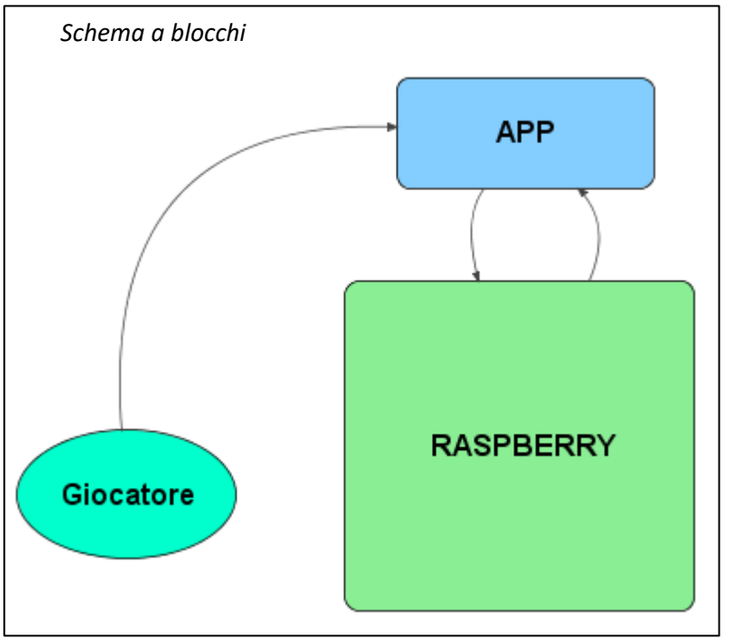
ARDUINO <- RASPBERRY	
"M-syx-syx-p"	Esegue uno spostamento
"B"	Emette un suono acuto
"BB"	Emette due suoni acuti
"Z"	Emette un suono indicante un'errore
"T"	Imposta il colore del led RBG a verde
"F"	Imposta il colore del led RBG a rosso

ARDUINO -> RASPBERRY	
"SHOOT"	Comanda lo scatto di una foto
"SHUTDOWN"	Comanda lo spegnimento del sistema

Il pulsante di reset serve a reinizializzare la posizione del braccio robotico

Applicazione:

Schema a blocchi



La comunicazione tra l'elaboratore e Arduino avviene in maniera wireless tramite una connesione di tipo Bluetooth.

L'Applicazione soddisfa la necessità di avere un interfaccia grafica per il controllo e la visualizzazione delle informazioni di gioco. Svolge le funzioni di:

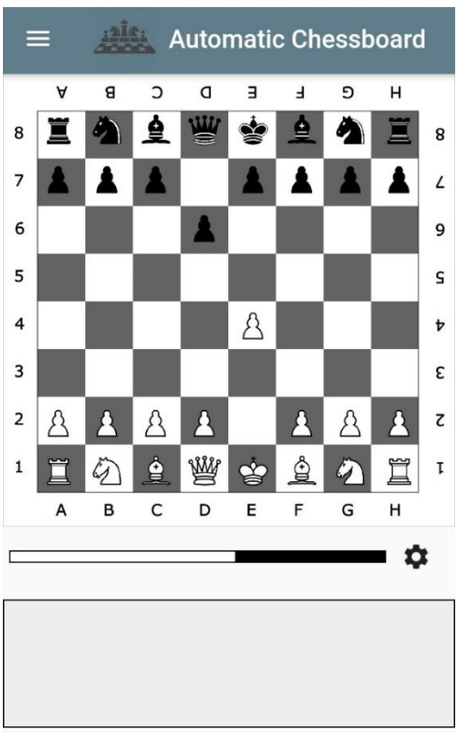
- Visualizzazione in tempo reale dello stato della scacchiera
- Visualizzazione del vantaggio
- Comandare l'Inizio di una nuova partita con i Bianchi o con i Neri
- Comandare la conclusione di una partita già in corso
- Impostare il livello di difficoltà del Computer

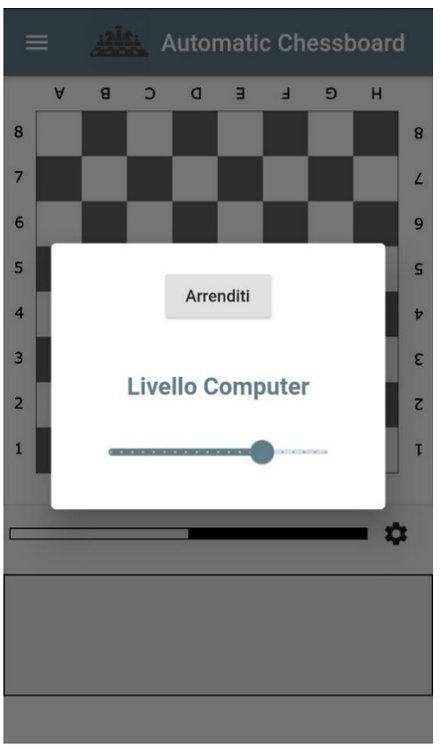
Tabella relativa allo scambio di stringhe

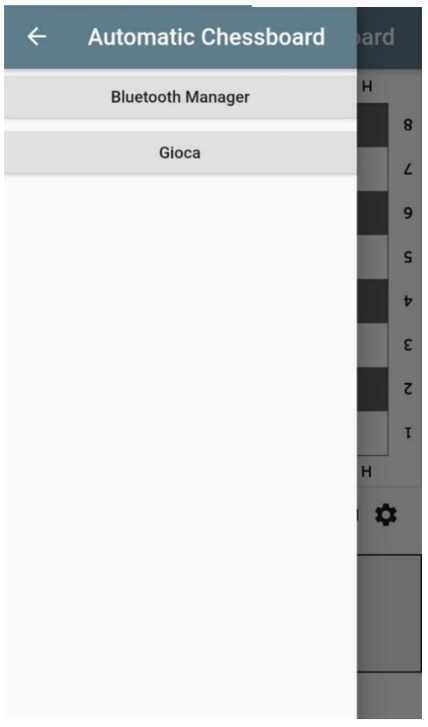
APP <- RASPBERRY	
"VANTAGE-[X]"	Visualizza nella barra del vantaggio il valore X ricevuto
"CB-LEFT-[00p- ... -71p]-GRID-[00p- ... -77p]-WHITE-[00p- ... -71p]"	Questa stringa corrisponde alla scacchiera del programma che dopo essere stata analizzata verrà visualizzata graficamente

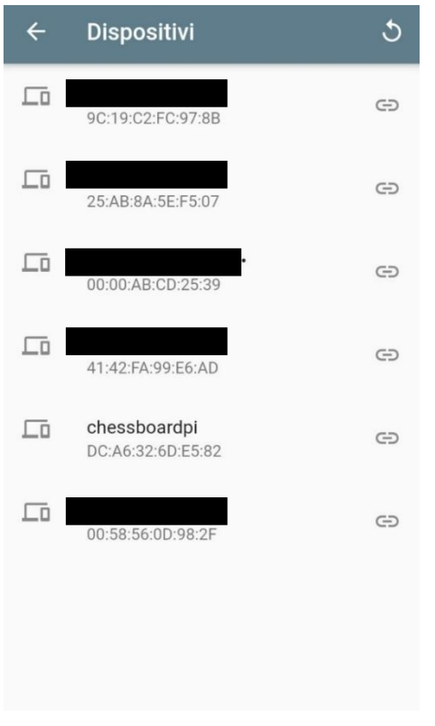
APP -> RASPBERRY	
"NEWGAME-[WHITE/BLACK]"	Avvia una nuova partita con i binachi o i neri
"SURRENDER"	Conclude una partita in corso
"DIFFICULTY-[N]"	Imposta il livello di diffocoltà del computer

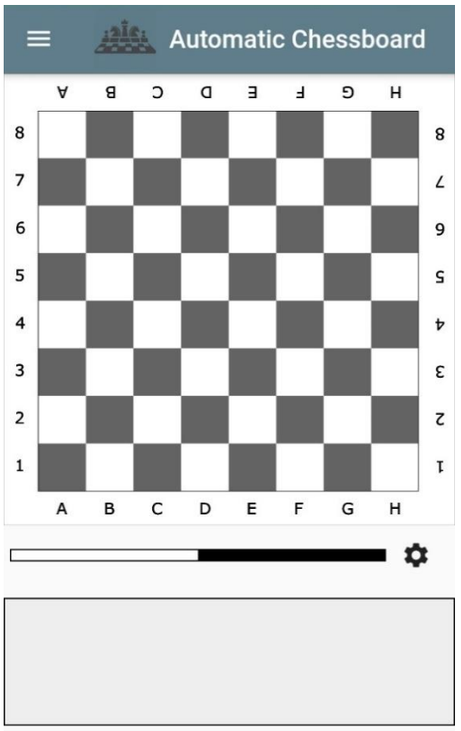
Seguono gli screen dell'Applicazione

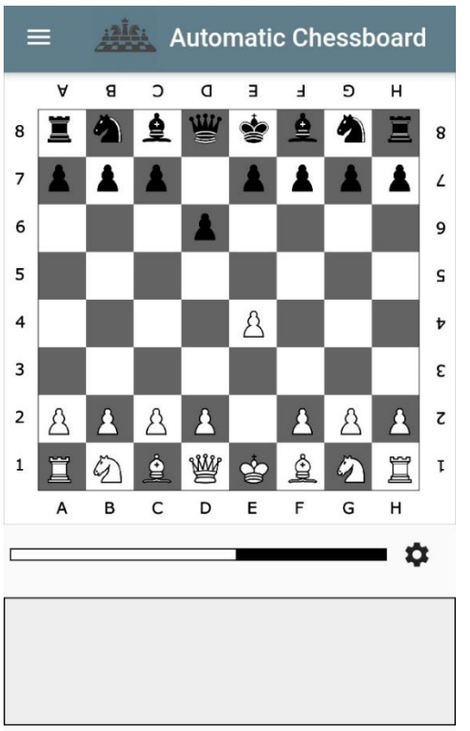












Schema di Montaggio

Componenti:

- x2 Stepper Motor Nema 17
- x1 Arduino Nano
- x1 Raspberry pi 4
- x1 Servo a rotazione continua
- x1 Buzzer piezoelettrico
- x2 push button (4 pin)
- x1 push button (2 pin)
- x2 A4988 Stepper Driver
- x2 Condensatori 47uF
- x5 Resistori 20Kohm
- x1 Resistore 1kOhm
- x1 Diodo Led RGB
- x1 Raspberry pi Camera
- x1 regolatore di tensione l7805
- x1 elettrocalamita
- x2 finecorsa
- x1 Alimentatore 12V 2A
- x1 Alimentatore 5V 3A
- x1 Relè

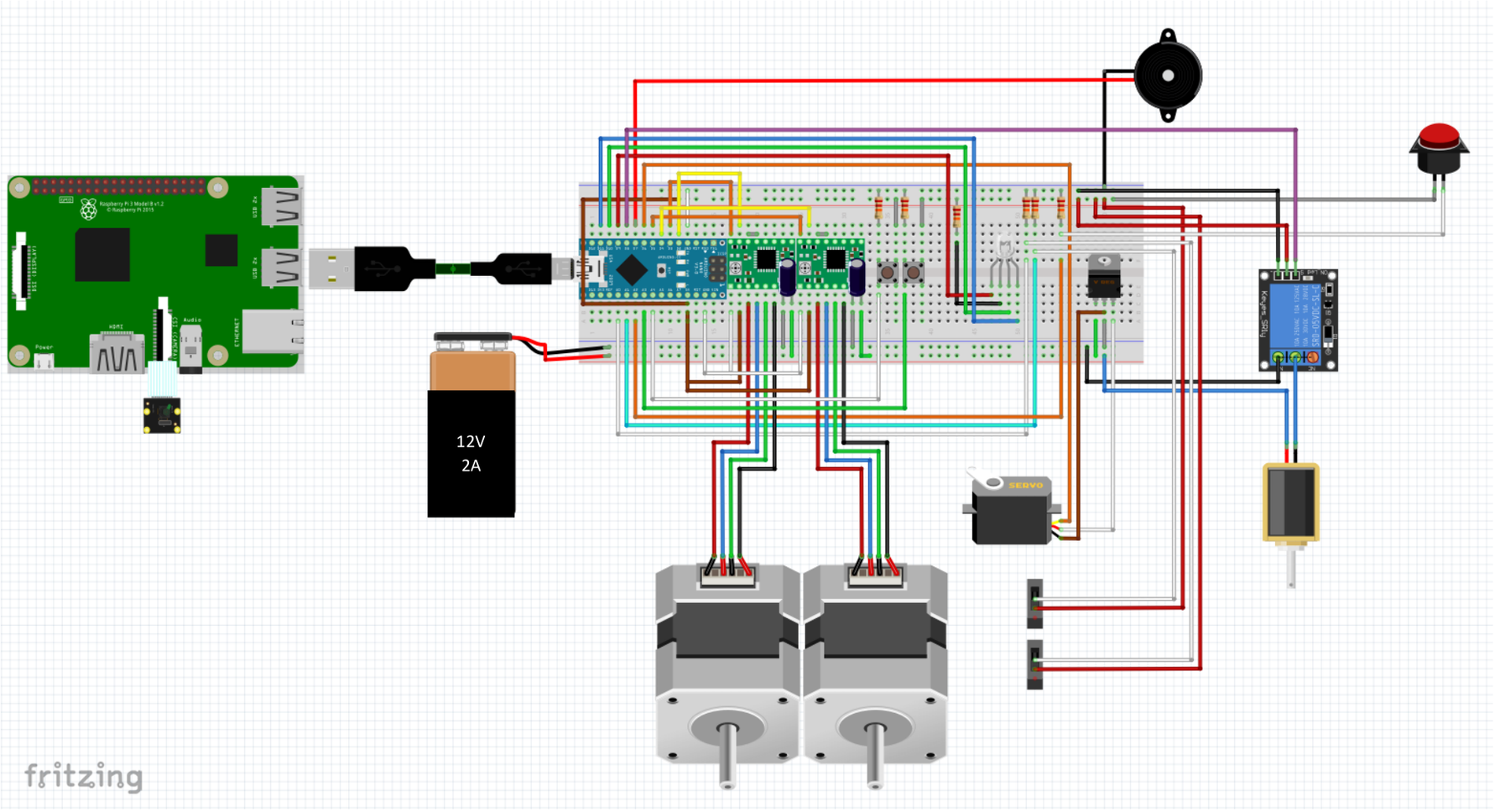
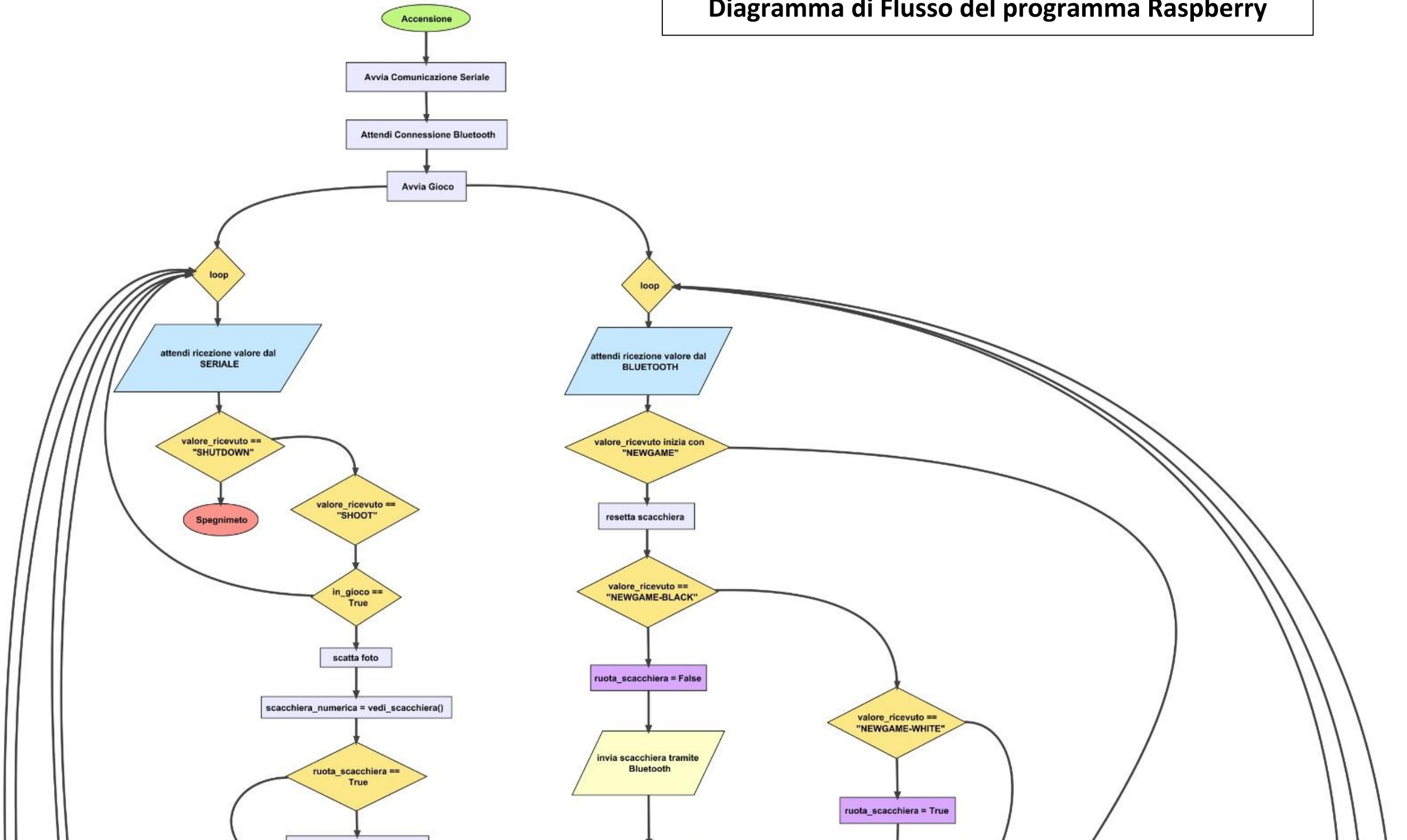
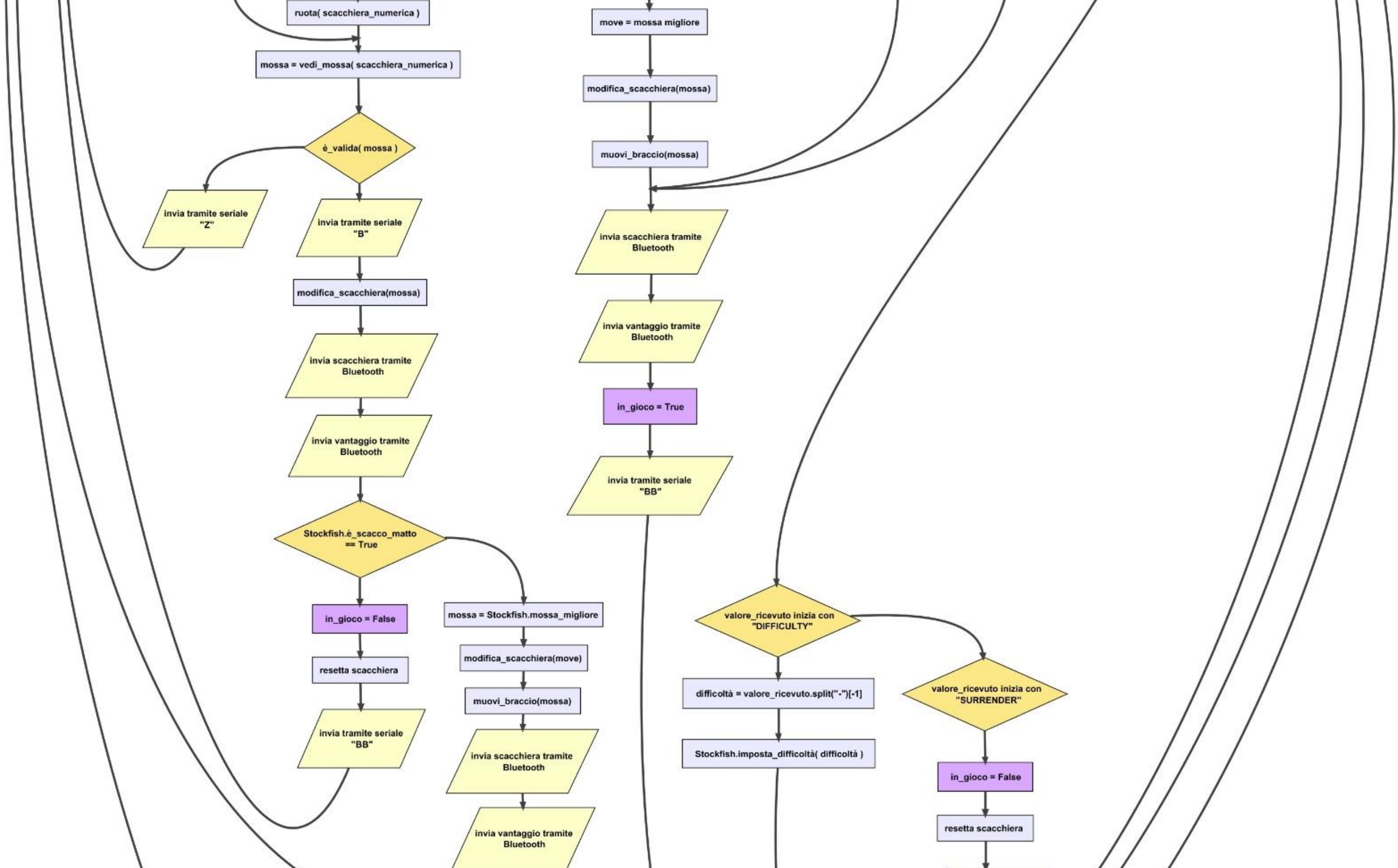


Diagramma di Flusso del programma Raspberry




ruota(scacchiera_numerica)
mossa = vedi_mossa(scacchiera_numerica)
è_valida(mossa)
invia tramite seriale "Z"
invia tramite seriale "B"
modifica_scacchiera(mossa)
invia scacchiera tramite Bluetooth
invia vantaggio tramite Bluetooth
Stockfish.è_scacco_matto == True
in_gioco = False
resetta scacchiera
invia tramite seriale "BB"
mossa = Stockfish.mossa_migliore
modifica_scacchiera(move)
muovi_braccio(mossa)
invia scacchiera tramite Bluetooth
invia vantaggio tramite Bluetooth
move = mossa migliore
modifica_scacchiera(mossa)
muovi_braccio(mossa)
invia scacchiera tramite Bluetooth
invia vantaggio tramite Bluetooth
in_gioco = True
invia tramite seriale "BB"
valore_ricevuto inizia con "DIFFICULTY"
difficoltà = valore_ricevuto.split("-")[-1]
Stockfish.imposta_difficoltà(difficoltà)
valore_ricevuto inizia con "SURRENDER"
in_gioco = False
resetta scacchiera

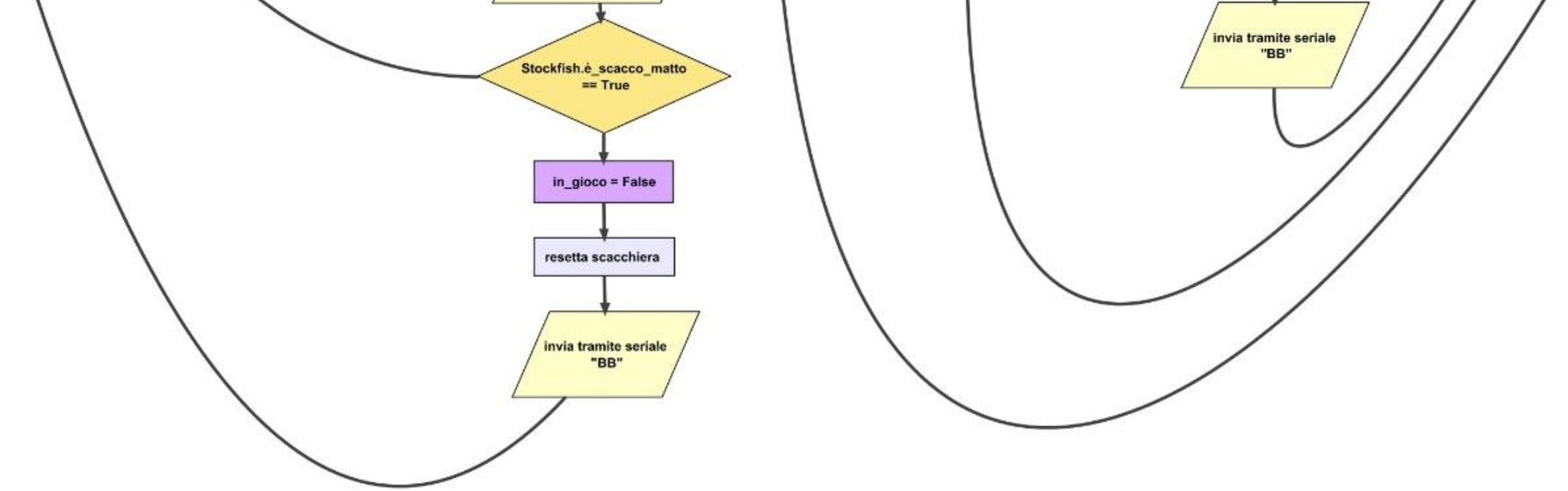
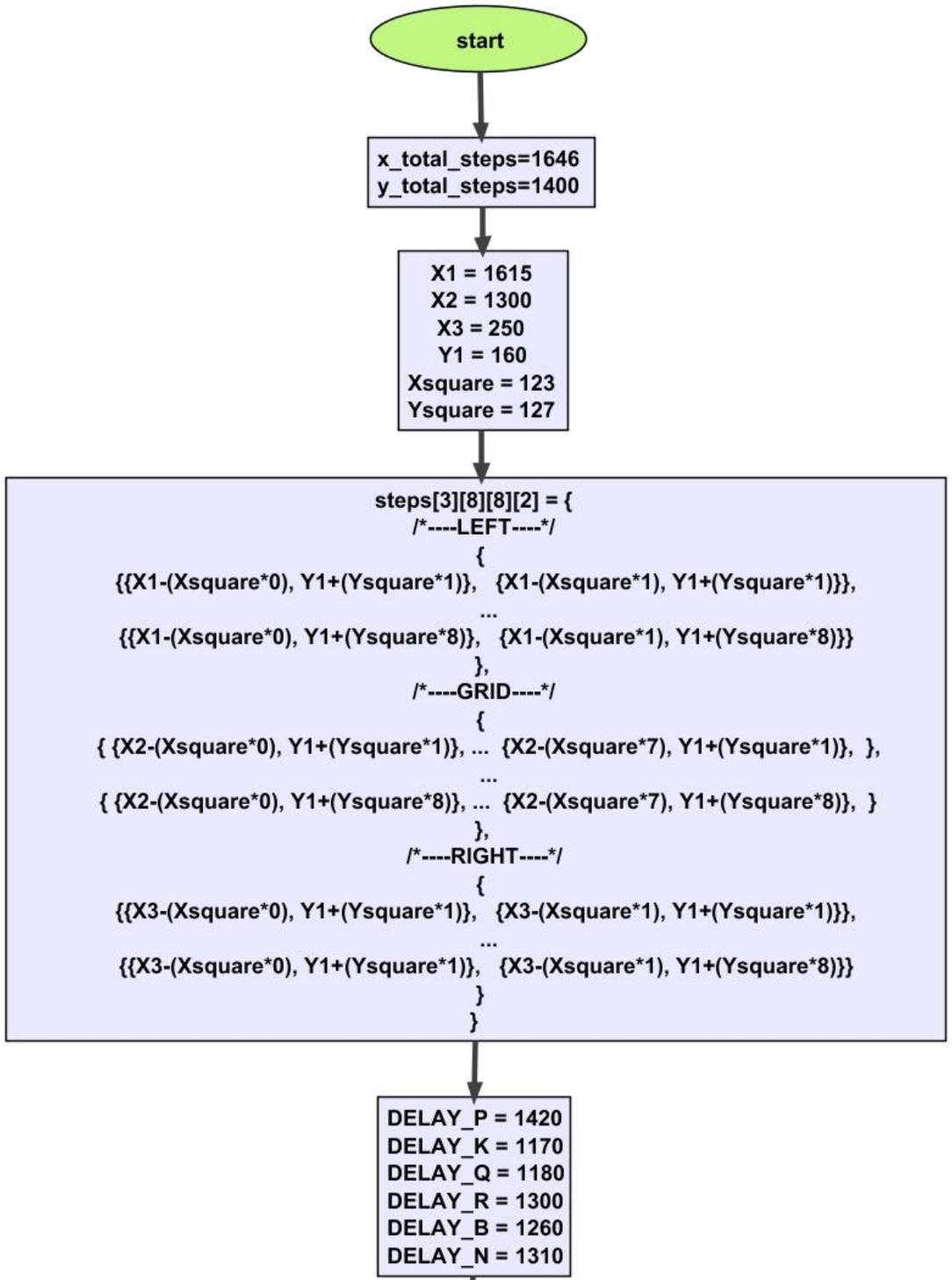
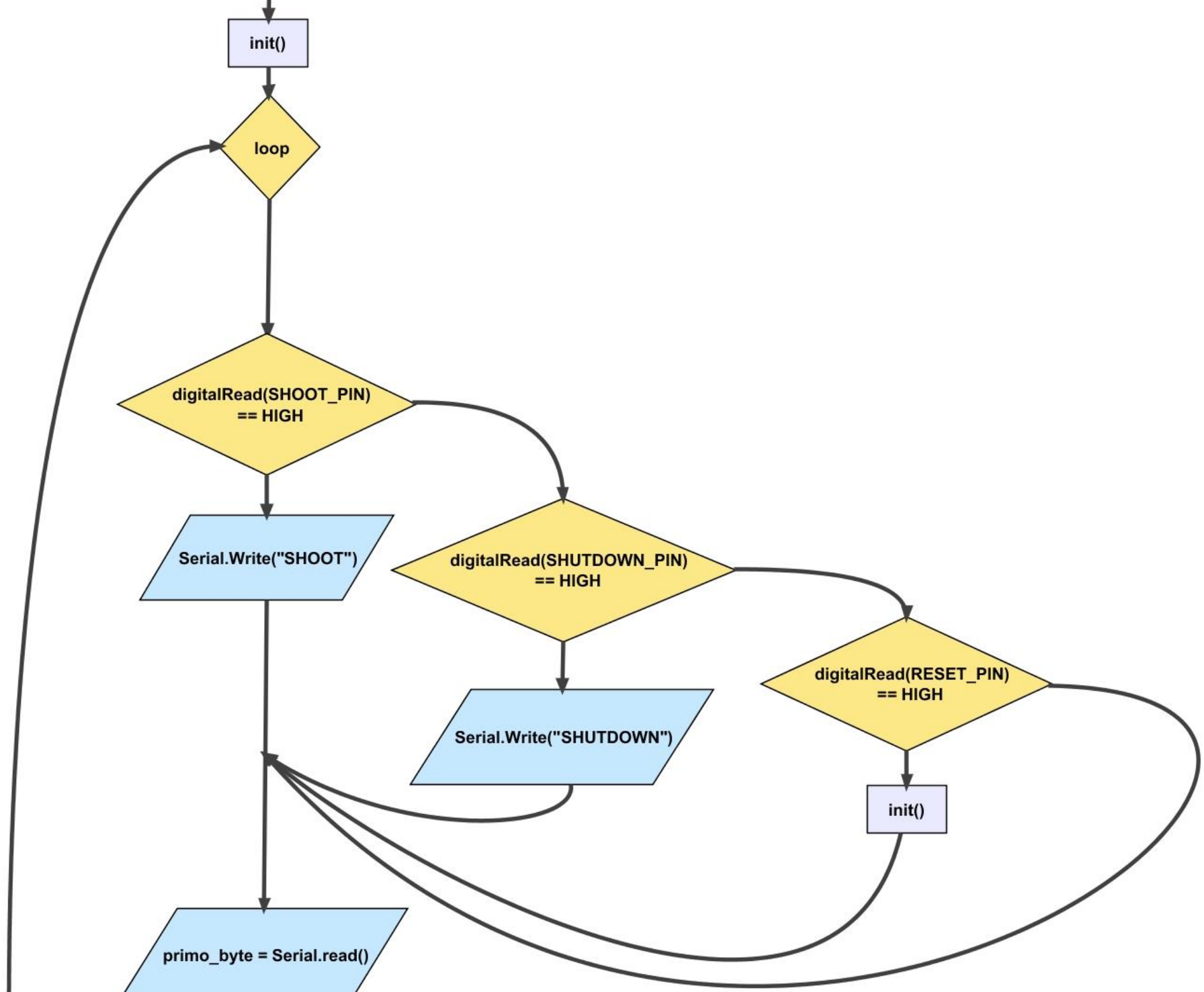
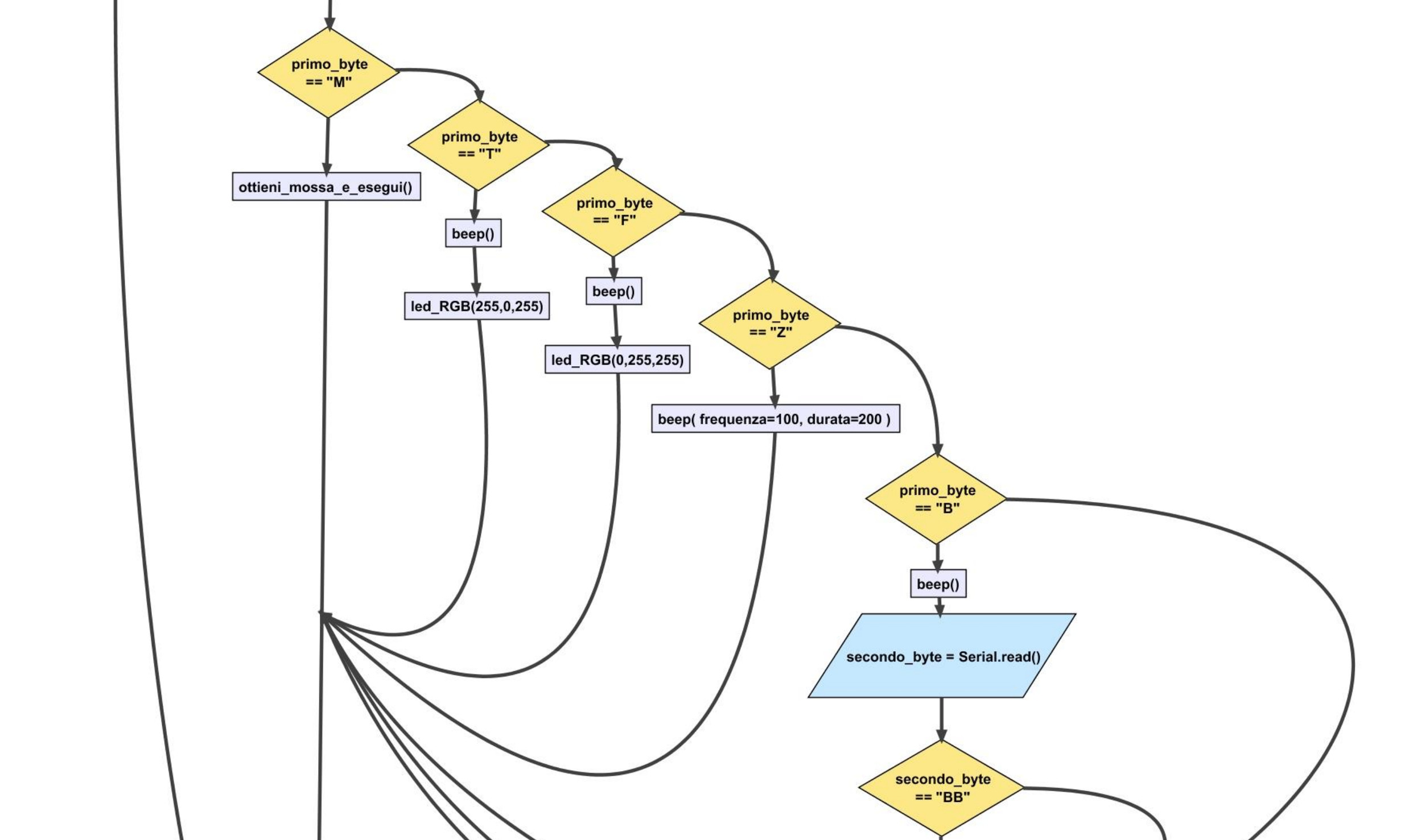
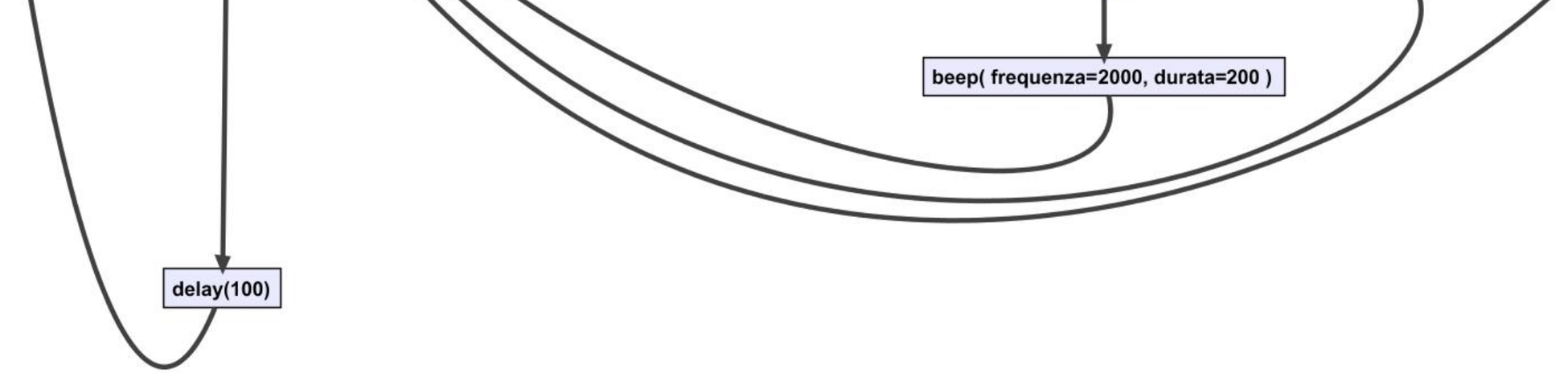

Stockfish.è_scacco_matto
== True
in_gioco = False
resetta scacchiera
invia tramite seriale
"BB"
invia tramite seriale
"BB"

Diagramma di Flusso dello Sketch Arduino




init()
loop
digitalRead(SHOOT_PIN)
== HIGH
Serial.Write("SHOOT")
digitalRead(SHUTDOWN_PIN)
== HIGH
Serial.Write("SHUTDOWN")
digitalRead(RESET_PIN)
== HIGH
init()
primo_byte = Serial.read()


primo_byte == "M"
ottieni_mossa_e_esegui()
primo_byte == "T"
beep()
led_RGB(255,0,255)
primo_byte == "F"
beep()
led_RGB(0,255,255)
primo_byte == "Z"
beep(frequenza=100, durata=200)
primo_byte == "B"
beep()
secondo_byte = Serial.read()
secondo_byte == "BB"


beep(frequenza=2000, durata=200)
delay(100)

Links ai codici de progetto:

- https://github.com/giuseppemacc/Automatic-Chessboard
- https://github.com/giuseppemacc/ChessApp
- https://github.com/giuseppemacc/Automatic-Chessboard-Arduino

Codice Arduino

```
#include <Servo.h>


/*----Servo object----*/
Servo servo;

/*----PIN----*/
const int X_DIR = 5;
const int X_STEP = 4;

const int Y_DIR = 3;
const int Y_STEP = 2;

const int RELE_PIN = 8;
const int BUZZER_PIN = 7;

const int SERVO_PIN = 6;

const int LED_R_PIN = 9;
const int LED_B_PIN = 10;
const int LED_G_PIN = 11;

const int Y_ENDSTOP = A1;
const int X_ENDSTOP = A0;

const int SHOOT_PIN = A2;
const int SHUTDOWN_PIN = A3;
const int RESET_PIN = A4;

/*----DELAY STPPER PULSE----*/
const int STEP_DELAY = 600;//560;

/*----step totali per una massima escursione di movimento----*/
int x_total_steps=1646;
int y_total_steps=1400;

/*----Servo DELAY for each piece----*/
const int DELAY_P = 1420;
const int DELAY_K = 1170;
const int DELAY_Q = 1180;
const int DELAY_R = 1300;
const int DELAY_B = 1260;
const int DELAY_N = 1310;


/*=======================-STEPS per ogni casella-=======================*/

/*----Step margin X axis from left to right----*/
int X1 = 1615;
int X2 = 1300;
int X3 = 250;

/*----Step margin Y axis at the top----*/
int Y1 = 160;

/*----X and Y squares Steps for one square----*/
int Xsquare = 123;
```

```
int Ysquare = 127;

/*----STEPS for each square----*/
int steps[3][8][8][2] = {
 /*----LEFT----*/
 {
  {{X1-(Xsquare*0), Y1+(Ysquare*1)},  {X1-(Xsquare*1), Y1+(Ysquare*1)}},



 {{X1-(Xsquare*0), Y1+(Ysquare*2)},  {X1-(Xsquare*1), Y1+(Ysquare*2)}},
  {{X1-(Xsquare*0), Y1+(Ysquare*3)},  {X1-(Xsquare*1), Y1+(Ysquare*3)}},
  {{X1-(Xsquare*0), Y1+(Ysquare*4)},  {X1-(Xsquare*1), Y1+(Ysquare*4)}},
  {{X1-(Xsquare*0), Y1+(Ysquare*5)},  {X1-(Xsquare*1), Y1+(Ysquare*5)}},
  {{X1-(Xsquare*0), Y1+(Ysquare*6)},  {X1-(Xsquare*1), Y1+(Ysquare*6)}},
  {{X1-(Xsquare*0), Y1+(Ysquare*7)},  {X1-(Xsquare*1), Y1+(Ysquare*7)}},
  {{X1-(Xsquare*0), Y1+(Ysquare*8)},  {X1-(Xsquare*1), Y1+(Ysquare*8)}}
 },
 /*----GRID----*/
 {
  { {X2-(Xsquare*0), Y1+(Ysquare*1)}, {X2-(Xsquare*1), Y1+(Ysquare*1)},  {X2-(Xsquare*2), Y1+(Ysquare*1)},
{X2-(Xsquare*3), Y1+(Ysquare*1)},  {X2-(Xsquare*4), Y1+(Ysquare*1)},  {X2-(Xsquare*5), Y1+(Ysquare*1)},  {X2-
(Xsquare*6), Y1+(Ysquare*1)},  {X2-(Xsquare*7), Y1+(Ysquare*1)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*2)}, {X2-(Xsquare*1), Y1+(Ysquare*2)},  {X2-(Xsquare*2), Y1+(Ysquare*2)},
{X2-(Xsquare*3), Y1+(Ysquare*2)},  {X2-(Xsquare*4), Y1+(Ysquare*2)},  {X2-(Xsquare*5), Y1+(Ysquare*2)},  {X2-
(Xsquare*6), Y1+(Ysquare*2)},  {X2-(Xsquare*7), Y1+(Ysquare*2)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*3)}, {X2-(Xsquare*1), Y1+(Ysquare*3)},  {X2-(Xsquare*2), Y1+(Ysquare*3)},
{X2-(Xsquare*3), Y1+(Ysquare*3)},  {X2-(Xsquare*4), Y1+(Ysquare*3)},  {X2-(Xsquare*5), Y1+(Ysquare*3)},  {X2-
(Xsquare*6), Y1+(Ysquare*3)},  {X2-(Xsquare*7), Y1+(Ysquare*3)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*4)}, {X2-(Xsquare*1), Y1+(Ysquare*4)},  {X2-(Xsquare*2), Y1+(Ysquare*4)},
{X2-(Xsquare*3), Y1+(Ysquare*4)},  {X2-(Xsquare*4), Y1+(Ysquare*4)},  {X2-(Xsquare*5), Y1+(Ysquare*4)},  {X2-
(Xsquare*6), Y1+(Ysquare*4)},  {X2-(Xsquare*7), Y1+(Ysquare*4)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*5)}, {X2-(Xsquare*1), Y1+(Ysquare*5)},  {X2-(Xsquare*2), Y1+(Ysquare*5)},
{X2-(Xsquare*3), Y1+(Ysquare*5)},  {X2-(Xsquare*4), Y1+(Ysquare*5)},  {X2-(Xsquare*5), Y1+(Ysquare*5)},  {X2-
(Xsquare*6), Y1+(Ysquare*5)},  {X2-(Xsquare*7), Y1+(Ysquare*5)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*6)}, {X2-(Xsquare*1), Y1+(Ysquare*6)},  {X2-(Xsquare*2), Y1+(Ysquare*6)},
{X2-(Xsquare*3), Y1+(Ysquare*6)},  {X2-(Xsquare*4), Y1+(Ysquare*6)},  {X2-(Xsquare*5), Y1+(Ysquare*6)},  {X2-
(Xsquare*6), Y1+(Ysquare*6)},  {X2-(Xsquare*7), Y1+(Ysquare*6)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*7)}, {X2-(Xsquare*1), Y1+(Ysquare*7)},  {X2-(Xsquare*2), Y1+(Ysquare*7)},
{X2-(Xsquare*3), Y1+(Ysquare*7)},  {X2-(Xsquare*4), Y1+(Ysquare*7)},  {X2-(Xsquare*5), Y1+(Ysquare*7)},  {X2-
(Xsquare*6), Y1+(Ysquare*7)},  {X2-(Xsquare*7), Y1+(Ysquare*7)},  },
  { {X2-(Xsquare*0), Y1+(Ysquare*8)}, {X2-(Xsquare*1), Y1+(Ysquare*8)},  {X2-(Xsquare*2), Y1+(Ysquare*8)},
{X2-(Xsquare*3), Y1+(Ysquare*8)},  {X2-(Xsquare*4), Y1+(Ysquare*8)},  {X2-(Xsquare*5), Y1+(Ysquare*8)},  {X2-
(Xsquare*6), Y1+(Ysquare*8)},  {X2-(Xsquare*7), Y1+(Ysquare*8)},  }
 },
 /*----RIGHT----*/
 {
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*1)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*2)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*3)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*4)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*5)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*6)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*7)}},
  {{X3-(Xsquare*0), Y1+(Ysquare*1)},  {X3-(Xsquare*1), Y1+(Ysquare*8)}}
 }
};
/*====================================================================*/

/*----fa muovere il braccio in x ed y contemporaneamente, con una determinata direzione e numero di step per
asse----*/
void step(bool x_dir, bool y_dir, int x_step, int y_step){
 /*
 Direction:
  False: destra, avanti //verso il finecorsa
  True: sinistra, dietro //lontano dal finecorsa
 */
 digitalWrite(X_DIR,x_dir);
 digitalWrite(Y_DIR,y_dir);
 delay(STEP_DELAY);
```

```
  bool is_x_step_max=false;
  if(x_step>y_step)
    is_x_step_max=true;

  if(is_x_step_max){
    for(int i=0; i<x_step; i++){
      if(i<y_step){
        digitalWrite(X_STEP,HIGH);
        delayMicroseconds(STEP_DELAY/2);

        digitalWrite(Y_STEP,HIGH);
        delayMicroseconds(STEP_DELAY/2);

        digitalWrite(X_STEP,LOW);
        delayMicroseconds(STEP_DELAY/2);

        digitalWrite(Y_STEP,LOW);
        delayMicroseconds(STEP_DELAY/2);
      }else{
        digitalWrite(X_STEP,HIGH);
        delayMicroseconds(STEP_DELAY);
        digitalWrite(X_STEP,LOW);
        delayMicroseconds(STEP_DELAY);
      }
    }
  }else{
    for(int i=0; i<y_step; i++){
      if(i<x_step){
        digitalWrite(X_STEP,HIGH);
        delayMicroseconds(STEP_DELAY/2);

        digitalWrite(Y_STEP,HIGH);
        delayMicroseconds(STEP_DELAY/2);

        digitalWrite(X_STEP,LOW);
        delayMicroseconds(STEP_DELAY/2);

        digitalWrite(Y_STEP,LOW);
        delayMicroseconds(STEP_DELAY/2);
      }else{
        digitalWrite(Y_STEP,HIGH);
        delayMicroseconds(STEP_DELAY);
        digitalWrite(Y_STEP,LOW);
        delayMicroseconds(STEP_DELAY);
      }
    }
  }
}

/*----fa abbassare e salire il grip e prendere o rilasicare una pedina----*/
void grip(bool pick, char piece, int type){
  //0 scendi, 1 sali
  int delay_piece;
  if(piece == 'P')
    delay_piece = DELAY_P;
  else if(piece == 'K')
    delay_piece = DELAY_K;
  else if(piece == 'Q')
    delay_piece = DELAY_Q;
  else if(piece == 'R')
    delay_piece = DELAY_R;
  else if(piece == 'B')
    delay_piece = DELAY_B;
  else if(piece == 'N')
    delay_piece = DELAY_N;

  if(type!=1){
    delay_piece +=50;
  }
```

```
  int errore_scendi_1 = 0;
  int errore_sali_1 = 50;//20;

  int errore_scendi_2 = -45;//-15;
  int errore_sali_2 = 60;//30;

  servo.write(0);
  if(pick)
    delay(delay_piece+errore_scendi_1);
  else
    delay(delay_piece+errore_scendi_2);
  servo.write(90);
  delay(200);
  digitalWrite(8,pick);
  digitalWrite(8,pick);
  digitalWrite(8,pick);
  digitalWrite(8,pick);
  delay(2000);

  servo.write(180);
  if(pick)
    delay(delay_piece+errore_sali_1);
  else
    delay(delay_piece+errore_sali_2);
  servo.write(90);
  delay(2000);

}

/*----fa compiere una mossa completa al braccio----*/
void move(int start_type, int start_row, int start_column, int end_type, int end_row, int end_column, char
piece){
  int start_stps_x = steps[start_type][start_row][start_column][0];
  int start_stps_y = steps[start_type][start_row][start_column][1];

  int end_total_stps_x = steps[end_type][end_row][end_column][0];
  int end_total_stps_y = steps[end_type][end_row][end_column][1];

  int end_stps_x = start_stps_x - end_total_stps_x;
  int end_stps_y = start_stps_y - end_total_stps_y;

  step(false, false, start_stps_x, start_stps_y);

  delay(500);

  grip(true,piece,start_type);

  delay(500);

  if((end_stps_x<0) && (end_stps_y<0)){
    step(false, false, end_stps_x*-1, end_stps_y*-1);
  }else if(end_stps_x<0){
    step(false, true, end_stps_x*-1, end_stps_y);
  }else if(end_stps_y<0){
    step(true, false, end_stps_x, end_stps_y*-1);
  }else {
    step(true, true, end_stps_x, end_stps_y);
  }

  delay(500);

  grip(false,piece,end_type);

  delay(500);

  step(true, true, end_total_stps_x, end_total_stps_y);

  delay(1000);
```

```
}

/*----controlla la mossa disponibile sul buffer seriale e la fa compiere al braccio----*/
void check_and_move(){
  delay(50);
  Serial.read();//skip the '-'

  int start_type = Serial.read() - '0';
  int start_row = Serial.read() - '0';
  int start_column = Serial.read() - '0';

  Serial.read();//skip the '-'

  int end_type = Serial.read() - '0';
  int end_row = Serial.read() - '0';
  int end_column = Serial.read() - '0';

  Serial.read();//skip the '-'

  char piece = Serial.read();

  move(start_type,start_row,start_column, end_type,end_row,end_column, piece);

}

/*----Inizializza il braccio alla sua posizione iniziale----*/
void init_(){
  digitalWrite(X_DIR,false);
  digitalWrite(Y_DIR,false);

  while(digitalRead(X_ENDSTOP)){
    digitalWrite(X_STEP,HIGH);
    delayMicroseconds(STEP_DELAY);
    digitalWrite(X_STEP,LOW);
    delayMicroseconds(STEP_DELAY);
  }
  while(digitalRead(Y_ENDSTOP)){
    digitalWrite(Y_STEP,HIGH);
    delayMicroseconds(STEP_DELAY);
    digitalWrite(Y_STEP,LOW);
    delayMicroseconds(STEP_DELAY);
  }
  delay(800);
  step(true,true,x_total_steps,y_total_steps);

}

/*----ricava i valori di step totali per X e Y e li visualizza sul monitor seriale----*/
void calibrate(){
  /*Questa funzione è stata utilizzata inizialmente per potersi ricavare i valori di step necessari per una massima
escursione del braccio nei suoi due assi*/
  x_total_steps=0;
  y_total_steps=0;
  digitalWrite(X_DIR,false);
  digitalWrite(Y_DIR,false);

  while(digitalRead(X_ENDSTOP)){
    digitalWrite(X_STEP,HIGH);
    delayMicroseconds(STEP_DELAY);
    digitalWrite(X_STEP,LOW);
    delayMicroseconds(STEP_DELAY);
    x_total_steps++;
  }
  while(digitalRead(Y_ENDSTOP)){
    digitalWrite(Y_STEP,HIGH);
    delayMicroseconds(STEP_DELAY);
    digitalWrite(Y_STEP,LOW);
    delayMicroseconds(STEP_DELAY);
    y_total_steps++;
  }
```

```
  Serial.print("x_total_steps: ");
  Serial.println(x_total_steps);
  Serial.print("y_total_steps: ");
  Serial.println(y_total_steps);
}

/*----fa suonare il buzzer----*/
void beep(int frequency=2000, int duration=100){
  tone(BUZZER_PIN,frequency,duration);
  delay(100);
}

/*----gestisce i task da eseguire alla pressione di dei pulsanti----*/
void button_controller(){
  if(digitalRead(SHOOT_PIN)==HIGH){
    beep();
    Serial.write("SHOOT");
    delay(1000);
  }
  if(digitalRead(SHUTDOWN_PIN)==HIGH){
    beep();
    Serial.write("SHUTDOWN");
    delay(1000);
  }
  if(digitalRead(RESET_PIN)==HIGH){
    beep();
    delay(1000);
    init_();
    delay(1000);
  }
  if(digitalRead(X_ENDSTOP)==LOW){
    beep();
  }
  if(digitalRead(Y_ENDSTOP)==LOW){
    beep();
  }
}

/*----imposta il colore del led RGB e emette un beep----*/
void led_RGB(int r, int g, int b){
  digitalWrite(LED_R_PIN, r);
  digitalWrite(LED_G_PIN, g);
  digitalWrite(LED_B_PIN, b);
  beep();
}

/*----gesttisce i task da eseguire in base al primo valore ricevuto dal buffer Seriale----*/
void serial_controller(int start_byte){
  if(start_byte == 'M'){
    check_and_move();
  }else if(start_byte == 'B'){
    beep();
    int next_byte = Serial.read();
    if(next_byte == 'B'){
      beep(2000,200);
    }
  }else if(start_byte == 'Z'){
    beep(100,200);
  }else if(start_byte == 'T'){
    led_RGB(255,0,255);
  }else if(start_byte == 'F'){
    led_RGB(0,255,255);
  }
}

/*----SETUP----*/
void setup() {
  /*----OUTPUT----*/
  pinMode(X_DIR,OUTPUT); pinMode(X_STEP,OUTPUT);
  pinMode(Y_DIR,OUTPUT); pinMode(Y_STEP,OUTPUT);
```

```
  pinMode(RELE_PIN,OUTPUT);
  pinMode(BUZZER_PIN,OUTPUT);
  pinMode(SERVO_PIN,OUTPUT);

  pinMode(LED_R_PIN,OUTPUT);pinMode(LED_G_PIN,OUTPUT);pinMode(LED_B_PIN,OUTPUT);

  /*----INPUT----*/
  pinMode(X_ENDSTOP,INPUT); pinMode(Y_ENDSTOP,INPUT);
  pinMode(SHOOT_PIN,INPUT);
  pinMode(SHUTDOWN_PIN,INPUT);
  pinMode(RESET_PIN,INPUT);

  /*----SERVO INIT----*/
  servo.attach(6);
  /*----SERIAL INIT----*/
  Serial.begin(9600);
  /*----LED RGB RED DEFAULT VALUE----*/
  led_RGB(0,255,255);

  //calibrate();
  delay(4000);
  /*----STEPPER INIT----*/
  init_();
}

/*----LOOP----*/
void loop() {
  button_controller();
  int start_byte = Serial.read();
  serial_controller(start_byte);
}
```