POLLO SHIBA – SPECIFICHE SOFTWARE

Estratto e rimaneggiato dal documento abstract.odg:

Il server si presenta come una applicazione Linux, con un piccolo logo ed una interfaccia grafica (UI) per il game-master (user). L'applicazione permetterà di configurare una logica di gioco (attivazione dei laser a comando, seguendo pattern temporali o casuali, temporizzazione dei livelli), l'applicazione avrà anche un Monitor ausiliario in cui visualizzare alcuni aspetti di gioco

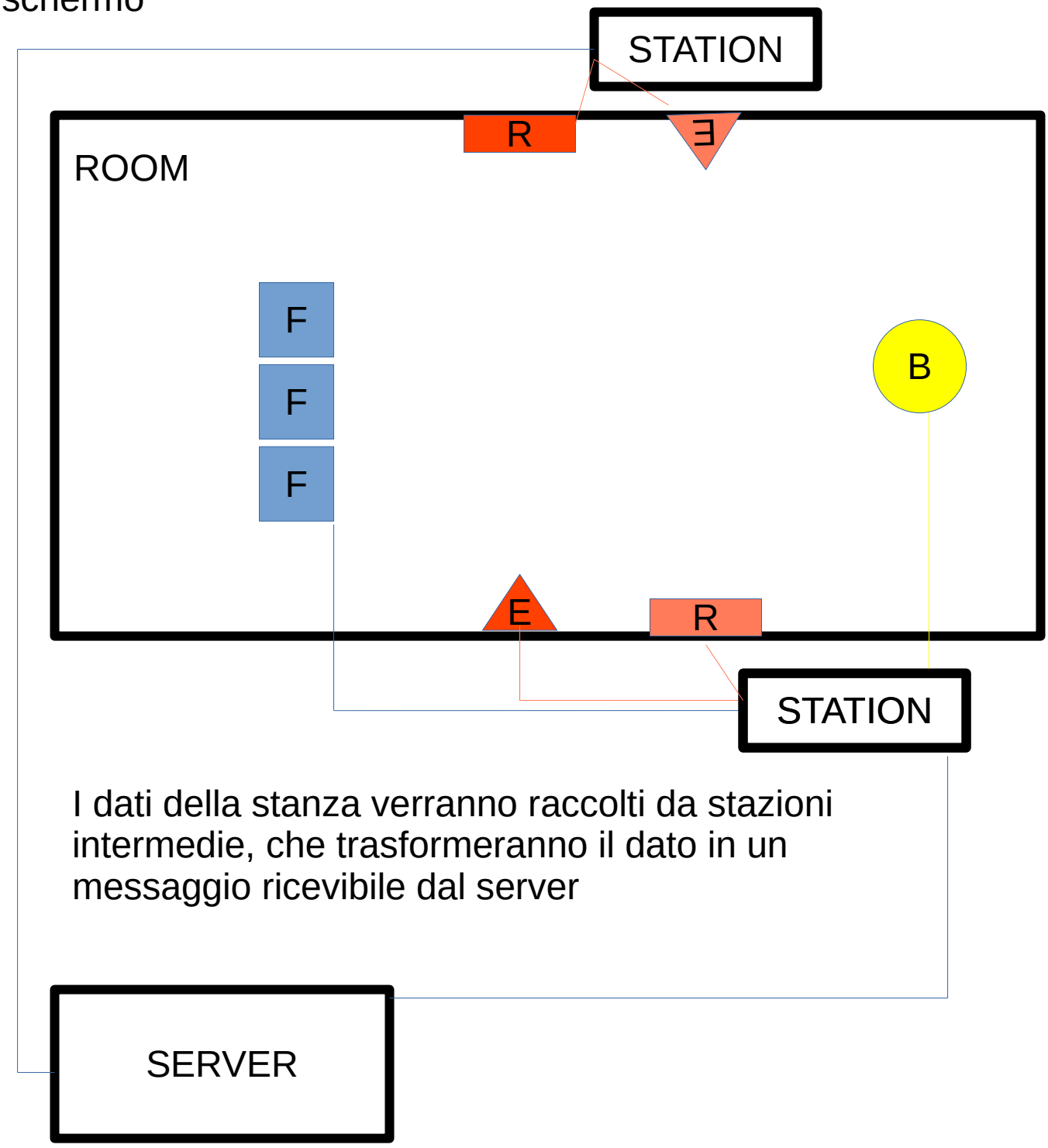
Lo stato della stanza sarà controllato in real-time e visualizzato in una mappa con una semplice interfaccia grafica stile videogame. [1]

Sarà possibile caricare diverse configurazioni di gioco ed operare con intervento umano. [2]

Nel software sono introdotti concetti di base di game design come best score, best time, leaderboard e varie altre da implementare [3]

Schema generico, estratto dal documento abstract.odg

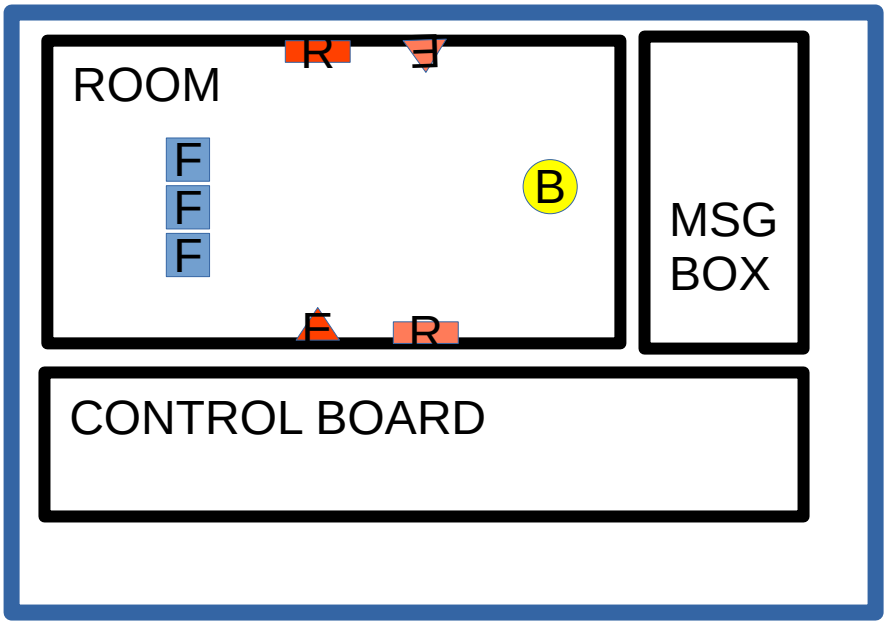
Un laser maze in cui gli elementi della stanza (pagina 2, F-R-E-B) vegono configurati e pilotati da un software di gestione (server, gestione livelli, enigmi, temporizzazione) e visualizzazione su schermo



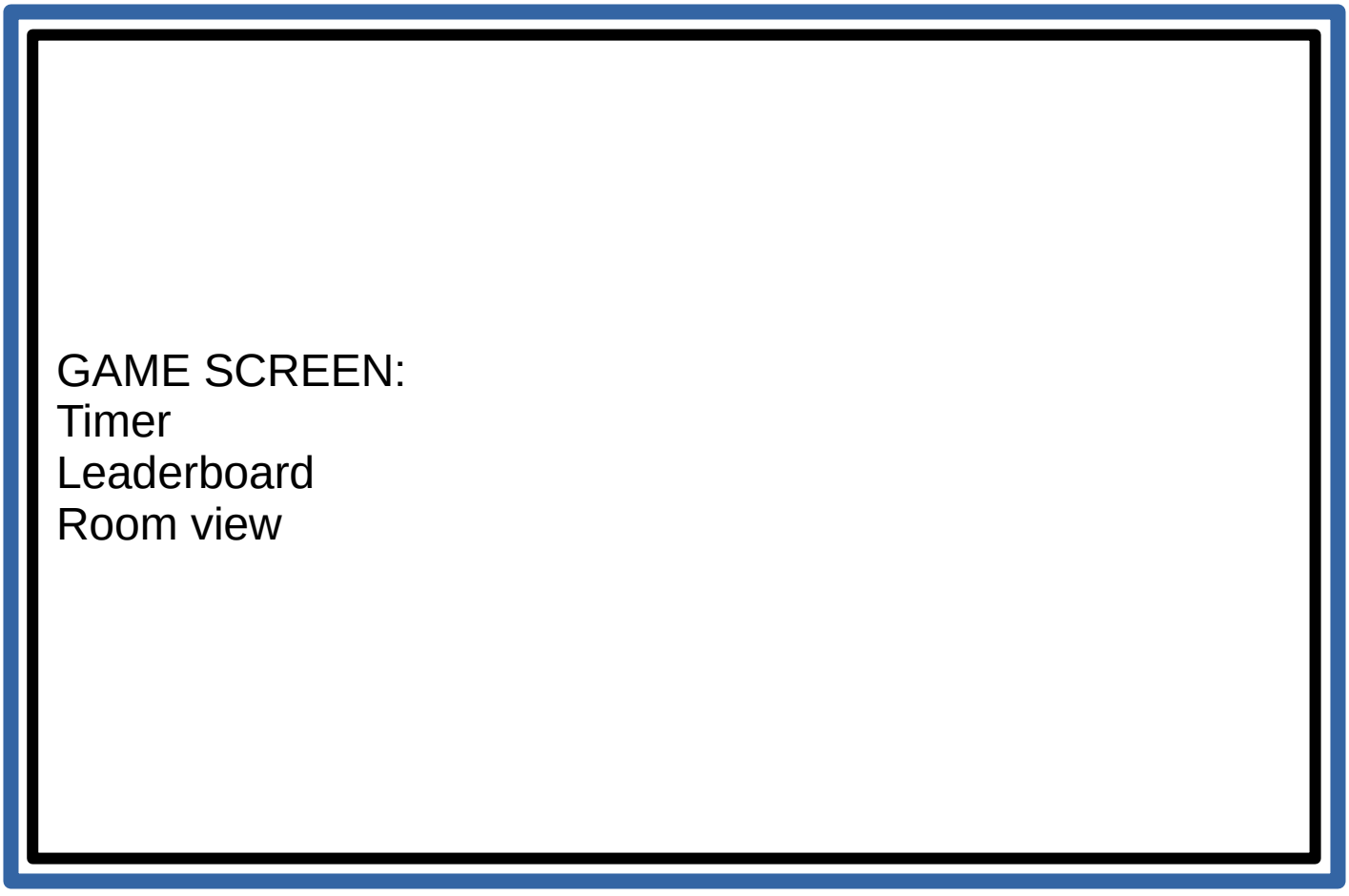
I dati della stanza verranno raccolti da stazioni intermedie, che trasformeranno il dato in un messaggio ricevibile dal server

Viewport Server side User interface (UI) [1]

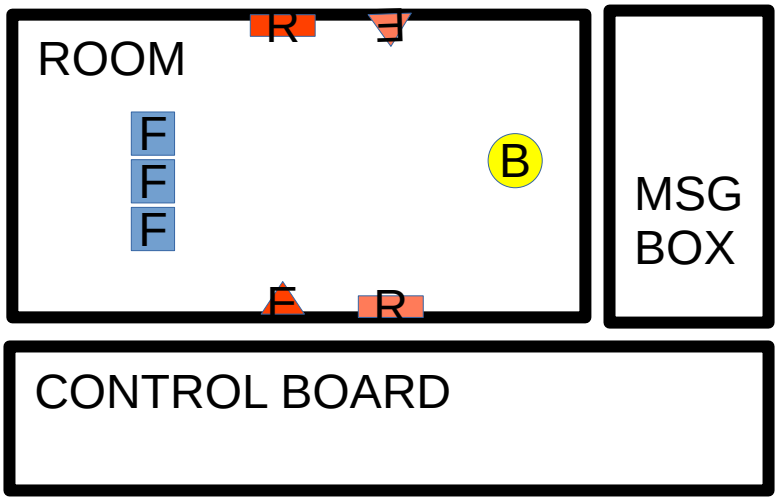
SCHERMO 1



SCHERMO 2



Server side User interface (UI) [1]

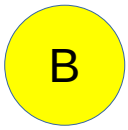


ROOM: fornisce una vista grafica dello stato degli agenti in campo. [1]

CONTROL BOARD: permette di pilotare gli agenti in campo in tempo reale e di eseguire azioni scriptate (laser temporizzati, pattern, timer) [2]

MSG BOX: terminale di controllo delle interazioni con gli agenti sul campo e low-level debugger

Descrizione Hardware ed elementi di gioco [2]:



Pulsante (B): se è attivato lancia una notifica quando viene premuto
Se è disattivato lancia una notifica ogni volta che cambia stato

Mattonella (F): sia se attiva che disattiva
emette una notifica se calpestata

Emitter (E): dispositivo di output, viene acceso ogni volta che riceve un messaggio di attivazione e spento
ogni volta che riceve una disattivazione

Receiver (R): dispositivo di input, viene acceso ogni volta
Che riceve un segnale da un emitter.

Qui scriviamo il funzionamento e la logica dei vari elementi:

Definizione Messaggi alle station

Tutti I messaggi inviati da pollo shiba iniziano con

$pollo$

Tutti I messaggi inviati dalle station iniziano con la lettera s seguita dall’identificativo della station (2 cifre), ad esempio

$s01$

I messaggi di pollo saranno indirizzati a un agent nel secondo campo del messaggio, il contenuto del messaggio è contenuto nel terzo campo, il quarto campo è un sequence number del messaggio:

$pollo$R1$an_information_payload$01337babe

I messaggi delle station sono sempre indirizzati a pollo, la restante struttura del messaggio è la stessa

$s02$pollo$an_information_payload$01337babe

Descrizione Livelli di gioco [3]

Scriviamo qui le logiche di gioco: marvin ha accennato a livello facile-difficile-estremo.

Descrizione task di gioco.

Generazione leaderboard dei giocatori.

Schema generico di una sessione di gioco e cose cosi

Descrizione Game Screen e Grafiche [3]:

Qui ci mettiamo la leaderboard, una eventuale grafica,
Il nome del gioco, il timer, insomma è uno schermo per il pubblico

Si puo usare un monitor o un proiettore, si puo mettere in una lobby
O waiting room

Roadmap astratta

Fase 1: toymodel (circa 10 ore)

In questa fase generiamo:
-La documentazione del progetto citato e del codice associato:
-un prototipo di interfaccia grafica in python3, con modulo pygame.
-Uno schema di descrizione del campo attraverso un config file
-Un sistema di messaging semplificato da visualizzare in msgbox

Fase 2: fattorizzazione (circa 25 ore)

Questa fase è quasi completamente di coding, in cui il codice viene rifattorizzato in una versione software stabile, viene implementata la logica di gioco, vengono limati gli angoli, viene eseguito il bugfixing e viene presentata la versione del codice definitiva.

Fase 3: interfaccia (tra 5 e 15 ore)
In questa fase si sviluppa e testa lo stato dell'interfaccia con stations e altri impianti hardware. Queste ore sono dedicate al tempo di Sviluppo e interfaccia con i sistemi hardware e sono correlate allo stato di
Avanzamento del lavoro di Edo

Fase 4: azione sul campo e manutenzione (? ore)
Fase di test, pronto intervento, manutenzione, da specificare bene Insieme

Cronotappe – Totale da 26 maggio a 15 giugno

Fase 1: toymodel (circa 10 ore)
Aperta e chiusa tra maggio 26 e maggio 27
26 Maggio:
- Generazione documentazione
- Generazione toy hardware arduino e codice toy hardware
- Codice comunicazione polloshiba-arduino

27 Maggio:
- prototipo funzionante del toymodel shiba con un toy hardware
- spiegazione funzionamento del toy e chiusura milestone

Fase 2: fattorizzazione (circa 25 ore)
Timeframe 31 Maggio – 10 Giugno:
25 ore da dividere in circa
4 ore refactoring codice toymodel,
10 ore development delle feature descritte nel documento
10 ore sviluppo e perfezionamento grafico
2 ore documentazione

31 maggio refactoring
1-4 giugno implementazione feature
5-9 giugno perfezionamento
10 giugno appuntamento con riunione e chiusura milestone

Fase 3: interfaccia (tra 5 e 15 ore)
Time frame 10 giugno- 15 giugno
Una o due giornate intere per perfezionare l'interfaccia
con gli agenti Hardware sistemati in ambiente di laboratorio

Fase 4: azione sul campo e manutenzione (? ore, ??? date)
Fase di test, pronto intervento, manutenzione, da specificare bene Insieme