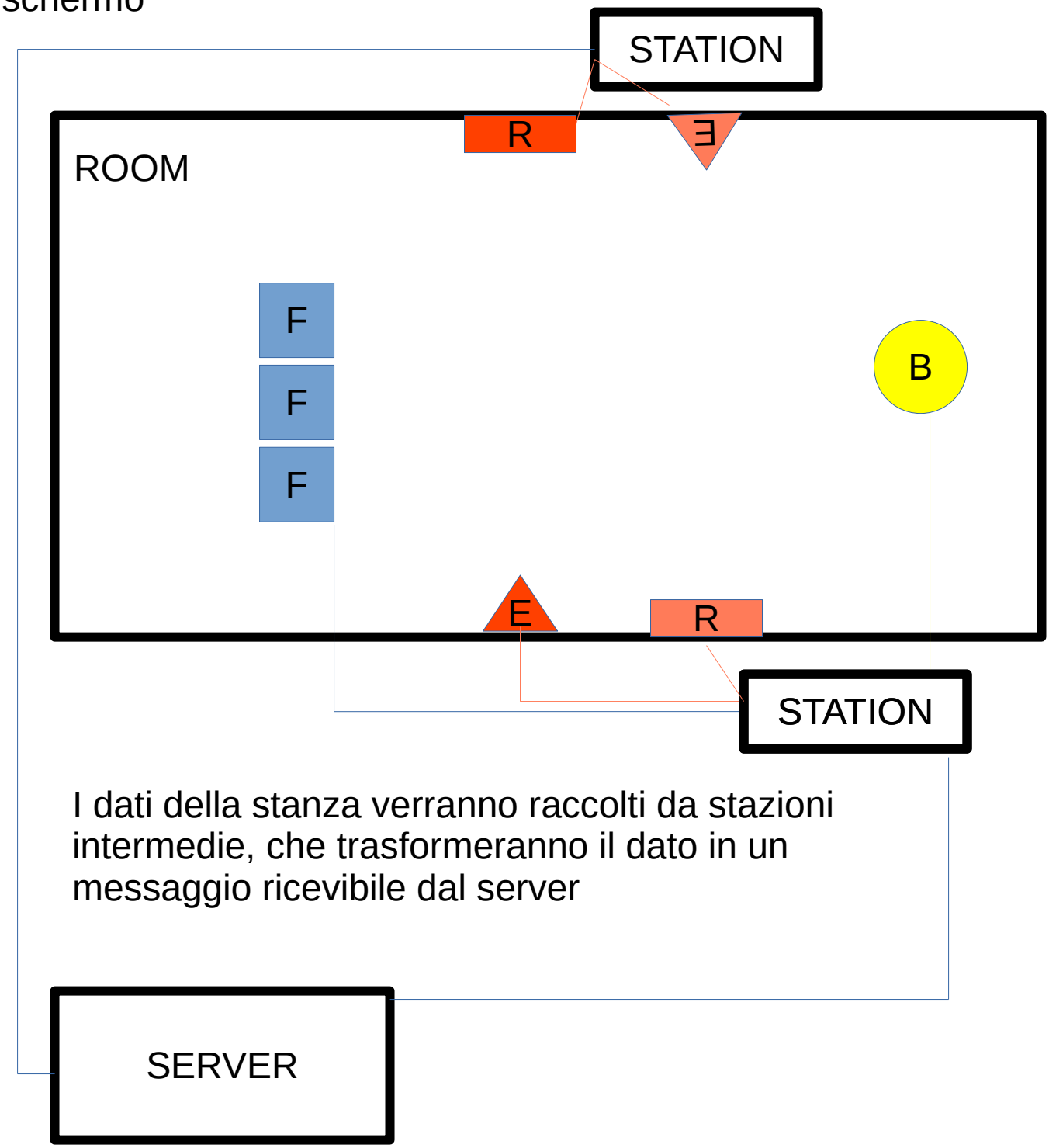
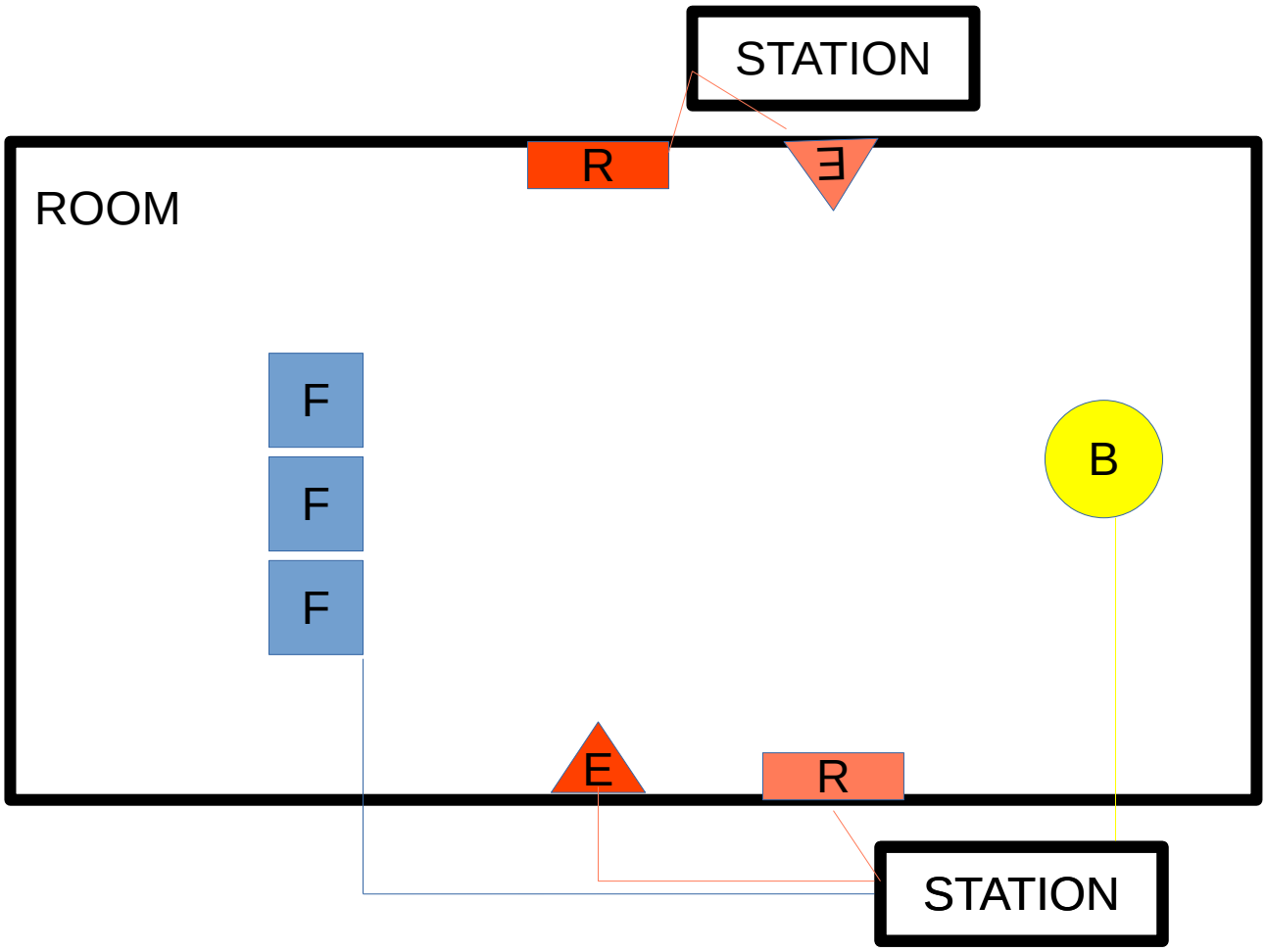
Schema generico

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

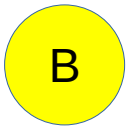


I dati della stanza verranno raccolti da stazioni intermedie, che trasformeranno il dato in un messaggio ricevibile dal server

Pianta della stanza



Descrizione Hardware ed elementi di gioco:



Pulsante: descrizione

Mattonella: descrizione

Emitter: descrizione

Receiver: descrizione

Questi oggetti vanno collegati alle station (raspberry o arduino tipicamente) che comunicheranno al server lo stato degli elementi.

Schema sessione giocatore (game loop):

Initialization:
Viene visualizzata la leaderboard
1) operatore inserisce da terminale
- player name
- vite (2 o 3 vite)

GAME LOOP:

il software esegue il loop di gioco:
0) operatore controlla che nessuno sia nella stanza
1) operatore inserisce la password di inizio sessione
2) il software entra nello stato ready

3) il giocatore puo entrare nella stanza
esecuzione level loop:
il loop esce in uno di questi tre stati:
WIN, DEAD, TIMEOUT
4) il giocatore esce dalla stanza
repeat

FINALIZATION:
Nome e punteggio del giocatore vengono aggiunti alla leaderboard
Le statistiche vengono aggiornate

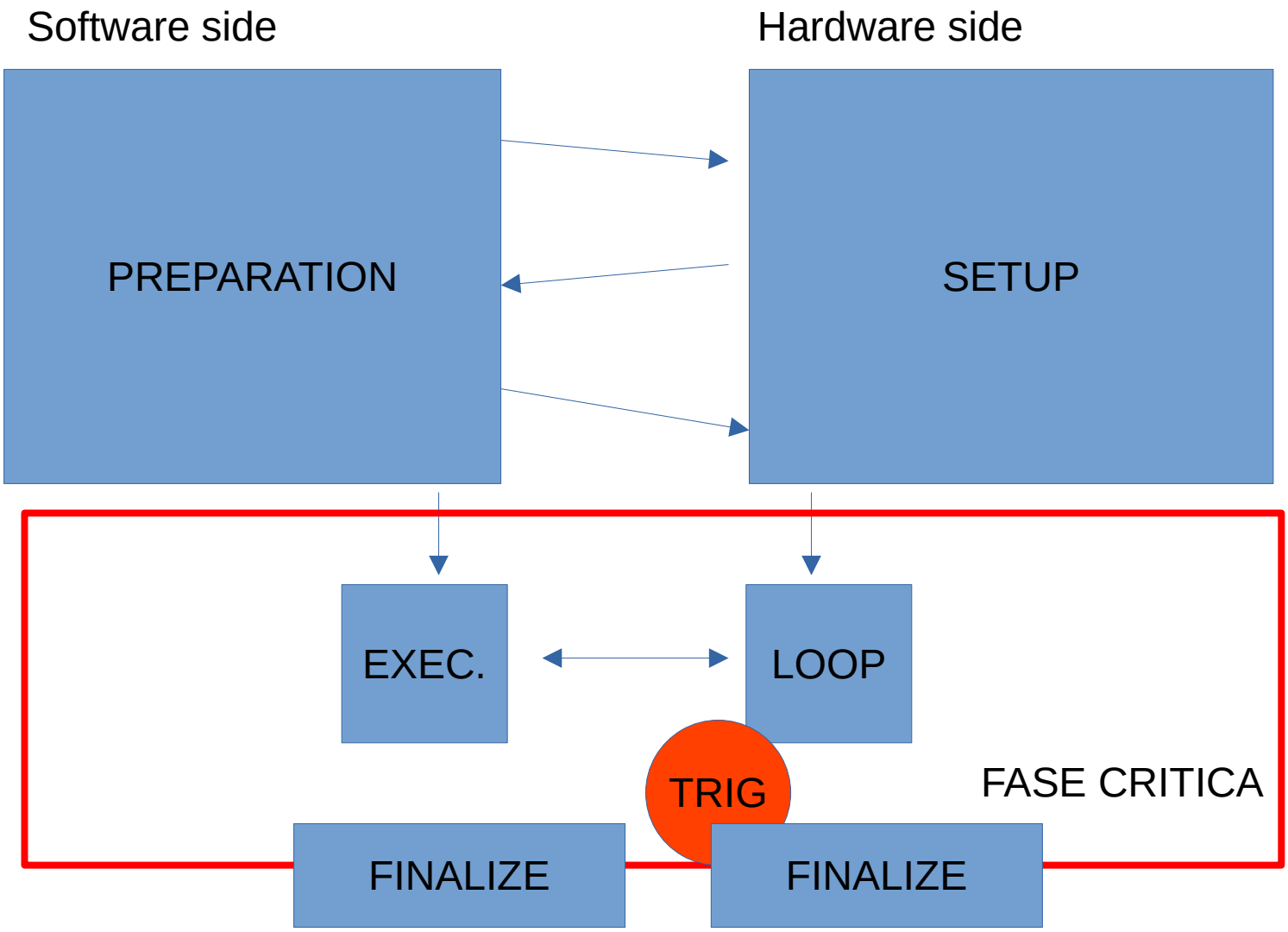
Schema sessione di gioco (level loop):

Una sessione di gioco è divisa in tre fasi:
1) preparation: in questa fase viene preparata una sessione di gioco.
Lato software: nome giocatore, new game si/no, livello da caricare

Segue una sessione di comunicazione con l'hardware:
Pollo comunica alle station quali agenti avviare (a seconda del livello)
Le station accendono gli agenti e comunicano a pollo l'esito della proc.

Una volta finita la sessione di preparazione, viene eseguita la
Sessione di esecuzione

Per ragioni di efficienza tutto il grosso della sessione è effettuato nella
Fase di preparazione, l'esecuzione deve essere sicura e performante



Descrizione Station

Le station si dividono in due categorie:
- Station
- Spotlight

Le station sono le stazioni critiche, richiedono comunicazione Veloce e risposta sicura. Una stazione controlla lo stato degli emitter, dei button e dei receiver (e in futuro delle trap), genericamente nominati “Agents”

Le spotlight sono stazioni non critiche, come luci di scena, fumo, sirene. Non è necessaria una elevata precisione nella gestione delle spotlight,

Descrizione routine setup:

Descrizione routine loop:

Descrizione Server – canovaccio

Il server si presenta come una applicazione Linux, con un piccolo logo ed una interfaccia grafica 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)

Sarà possibile caricare diverse configurazioni ed operare con intervento umano.

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

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

Descrizione Scene server side:

Preparation:

A seconda del livello pollo invia una stringa di inizializzazione:

(schema semplice, con config solo lato server)
Livello1:
#pollo$initE01E02E03@

Livello2:
#pollo$initE01E02E03@

Livello3:
#pollo$initE01E02E03E04E05@

Schema con config lato hardware:

Livello<N>:
#pollo$initN

E la configurazione del livello è caricata in un config file hardware

[to do: scegliere la strada da prendere]

Quando una stazione si avvia invia un messaggio di avvenuta init (a seconda del caso)
#S01$initOK@
#S01$initFAIL@

Quando pollo riceve l'ok da tutte le stazioni:
1) disegna la mappa, I controlli e gli agenti in campo

2) avverte il giocatore che
Puo entrare nella stanza e si entra nella fase di execution:

Descrizione Scene server side:

Execution: (prima versione)

Per ragioni di sicurezza, il numero di operazioni in questa fase è
Ridotta al minimo possibile:

Polloshiba interrompe ogni comunicazione con le station e si mette
In listen, la grafica non è aggiornata

Alla ricezione di un trigger (eseguito dalla lettura sottosoglia del
Receiver) da parte di una station, polloshiba ordina a tutte
Le station di finalizzare tutti gli agenti

Finalization:
Spegne tutti gli emitter (hardware side)