



UNIVERSITÀ DI PISA

Ingegneria Robotica e dell'Automazione

PROGETTO DEL CORSO:

”REAL TIME SYSTEMS”

TRAINS

ANNO ACCADEMICO 2022/2023

Professore:

Giorgio Buttazzo

Studenti:

**Paolo Marinelli
Caterina Pieri**

1 Introduzione

Il progetto tratta la simulazione di una stazione ferroviaria con otto binari monodirezionali sui quali vengono smistati i treni. I treni arrivano dai corrispettivi binari centrali e vengono smistati sulla stazione adatta al loro livello di priorità. L'obiettivo del progetto è quello di gestire gli scambi dei binari, i semafori e di scegliere il percorso di ogni treno in modo coerente con la sua priorità per minimizzare la permanenza in stazione dei treni a priorità maggiore.

I treni possono essere generati in modo casuale dall'utente, oppure è possibile definirne direzione e priorità.

Un'ulteriore possibilità consiste nella generazione automatica di treni casuali, in cui è comunque possibile definire una direzione.

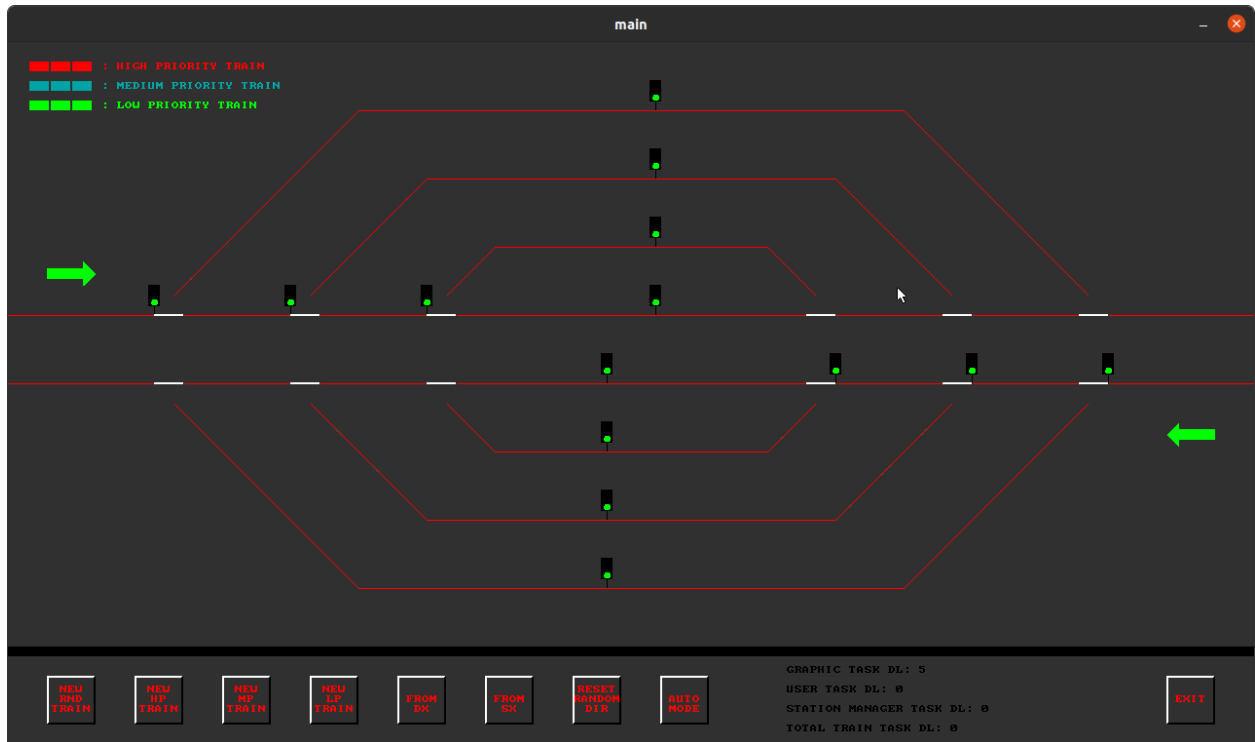


Figura 1: Anteprima dell'applicazione

2 Overview: funzionamento e interfaccia grafica

2.1 Stazione

Il sistema costruito è una versione semplificata di una stazione reale e si compone di otto binari, di cui due che la attraversano longitudinalmente e si occupano dell'ingresso e dell'uscita dei treni.

I binari della parte superiore vengono attraversati dai treni che si spostano da sinistra a destra, mentre quelli della parte inferiore vengono attraversati dai treni con direzione opposta.

Successivamente i treni vengono smistati nei vari binari, grazie a dei binari mobili, in modo coerente con la politica di gestione della loro priorità che verrà analizzata in seguito.

Quando arrivano in stazione effettuano una fermata e, sempre in base alla loro priorità, il sistema decide quale sarà il primo treno a lasciarla e sposta i binari mobili nella posizione corretta.

A seconda del traffico possono formarsi delle code, sia ai semafori delle stazioni che a quelli dei binari mobili, il sistema si occupa anche della loro corretta gestione in modo da evitare sovrapposizioni di treni.

2.2 Binari mobili

I binari mobili presenti possono muovere in modo fluido in due configurazioni in modo da collegarsi a due diversi binari. Ci sono due gruppi di binari mobili che vengono gestiti in modo leggermente diverso:

- *Binari mobili precedenti alla stazione:* Questi binari sono gestiti in modo dinamico, cercando di non interrompere il flusso dei treni. Quando un treno si trova in prossimità del binario, quest'ultimo legge la posizione richiesta dal treno e, in caso non si trovasse in quella posizione, inizia a spostarsi e il semaforo collegato diventa rosso fino al completamento dell'animazione. Il treno leggendo lo status del semaforo inizia a rallentare e continua a farlo fino alla sua fermata o finché il semaforo diventa verde, a quel punto accelera e torna alla sua velocità massima.
- *Binari mobili successivi alla stazione:* Questi binari iniziano il loro movimento nell'istante in cui viene deciso il treno che dovrà lasciare la stazione e si posizionano nella configurazione richiesta. Visto che l'animazione viene sempre terminata prima del movimento del treno, non hanno bisogno di un semaforo per indicare al treno che si deve fermare.

2.3 Treni

Ogni nuovo treno è completamente gestito da un relativo task e, durante il suo movimento, riceve e invia comandi con la stazione, in modo da attraversarla correttamente.

Quando viene creato, se non diversamente specificato, ha un priorità scelta in modo casuale tra tre valori (alta, media o bassa) e una direzione anch'essa casuale.

Per individuare visivamente i treni con differenti priorità gli sono stati assegnati bitmap di colori diversi, come si può vedere in figura 2.



Figura 2: Legenda dei diversi treni

Dopo aver deciso priorità e direzione, ad ogni treno viene assegnato un binario dalla stazione in base alla sua priorità e al traffico attuale. Dopo aver eseguito quest'operazione, il treno attraversa il sistema cambiando il proprio moto durante il percorso, in base alla situazione. Per gestire le differenti tipologie di movimento è stata implementata una macchina a stati:

GO FAST Il treno si muove a velocità massima e costante, questa situazione si verifica quando il treno si trova ancora lontano da stazione o semafori.

SLOW DOWN Il treno comincia a rallentare con decelerazione massima e costante, questa situazione si verifica prima di un semaforo.

QUEUE Il treno si trova in coda a un semaforo e deve traslare di posizione quando la coda viene fatta scorrere.

STOP Il treno è fermo ed è il primo della coda ad un semaforo, attende che quest'ultimo diventi verde per muoversi.

SPEED UP Il treno si muove con accelerazione massima e costante fino al raggiungimento della velocità massima.

In figura 3 è possibile visualizzare il diagramma logico relativo alla macchina a stati appena descritta.

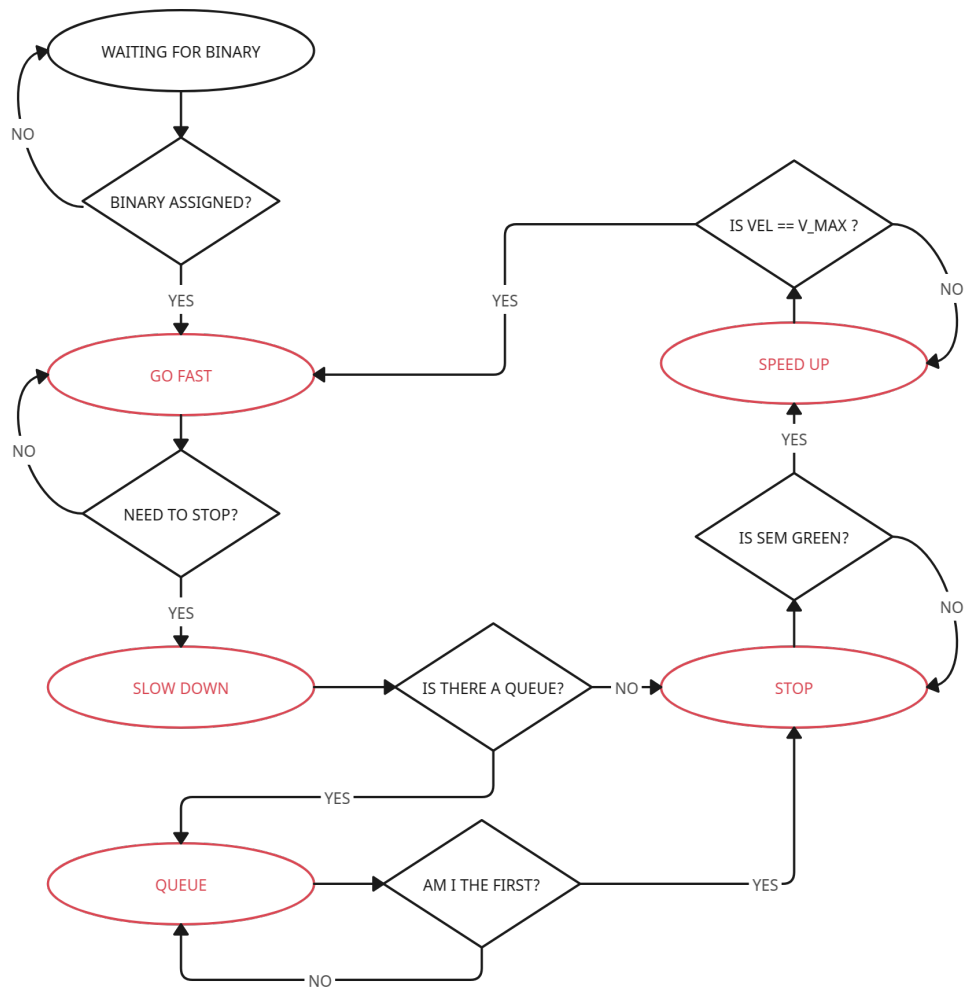


Figura 3: Diagramma logico della macchina a stati

2.4 Interfaccia utente

L'utente può interagire con il programma tramite tastiera o con dei pulsanti che si trovano nella parte inferiore della grafica che eseguono le seguenti funzioni:

NEW RND TRAIN Genera un treno con priorità random;

NEW HP TRAIN Genera un treno con alta priorità;

NEW MP TRAIN Genera un treno con media priorità;

NEW LP TRAIN Genera un treno con bassa priorità;

FROM DX Imposta la direzione di comparsa dei treni su FROM DX. Finché rimane attiva, tutti i treni generati compariranno da destra;

FROM SX Imposta la direzione di comparsa dei treni su FROM SX. Finché rimane attiva, tutti i treni generati compariranno da sinistra;

RESET RANDOM DIRECTION Resetta la decisione presa nei due pulsanti precedenti. Ovvero i nuovi treni verranno generati con direzione casuale;

AUTO MODE Genera automaticamente un nuovo treno ogni 2 secondi. La priorità sarà sempre casuale mentre è possibile definire una preferenza sulla direzione utilizzando i 3 pulsanti precedenti. Inoltre in questa modalità è comunque possibile generare dei treni in modo manuale, il sistema se ne accorge e genera un nuovo treno solo dopo 2 secondi dall'ultimo treno generato;

EXIT Chiude tutti i task e esce dal programma;



Figura 4: Pulsanti

Attraverso la tastiera invece le opzioni sono più limitate per evitare di complicare inutilmente l'interazione:

SPACE Genera un nuovo treno con priorità random e direzione in base alla modalità attiva al momento;

ESC Chiude tutti i task e esce dal programma;

2.5 Politica di assegnazione dei binari

Il task *station manager* si occupa dell'assegnamento dei binari dei nuovi task *treno* al momento della loro generazione. La decisione viene presa considerando sia la priorità del treno che il traffico attuale presente in stazione.

In particolare se il treno ha una priorità alta vengono controllati, a partire dal binario centrale, i binari in modo da scegliere quello con meno treni presenti prima della stazione. In caso di parità di treni presenti viene scelto quello più vicino al centro.

Se il treno ha una priorità media, la politica di scelta rimane la stessa però si evita di controllare il binario centrale.

Se il treno ha priorità bassa invece, vengono saltati i due binari più vicini al centro.

In questo modo il binario centrale viene riservato ai treni ad alta priorità, quello successivo è riservato a treni a priorità alta o media e nei due più esterni possono muoversi tutti i treni.

Con questo tipo di politica di assegnazione dei binari è possibile, seppur poco frequente, che un treno a priorità maggiore si trovi in coda ad una stazione con davanti un treno a priorità minore della sua. Questo inconveniente è stato considerato accettabile visto che viene parzialmente risolto dalla politica di abbandono della stazione.

(NB: La procedura descritta considera il caso generico di assegnamento su una delle due direzioni, quindi i binari tra i quali scegliere sono quattro e non otto).

2.6 Politica di abbandono della stazione

Durante l'esecuzione del programma è normale che si verifichino situazioni in cui più treni si trovano fermi in attesa in diverse stazioni, con un'eventuale coda.

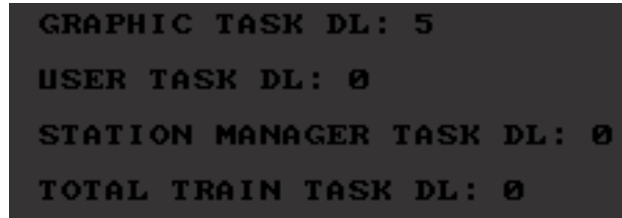
Il task *station manager* si occupa anche di individuare qual'è il primo treno a dover lasciare la stazione, per ciascuna delle due direzioni.

Data una direzione, vengono controllati tutti i treni in testa alla coda di una stazione con semaforo rosso e viene fatto partire per primo quello con priorità maggiore, in caso di parità si prediligono i treni sui binari più vicini al centro.

Visto che in questo modo poteva succedere che i treni a priorità bassa non riuscissero mai a verificare le condizioni descritte sopra, quando un qualsiasi treno si trova fermo in stazione per più di 8 secondi, la sua priorità viene automaticamente impostata al massimo.

2.7 Status monitor

Nella zona dedicata all'interfaccia utente è presente un sintetico status monitor che mostra il numero totale di deadline miss dei vari task. Per quanto riguarda i task relativi ai treni viene conteggiato il totale delle deadline miss di ciascuno di essi dall'inizio dell'esecuzione del programma.



```
GRAPHIC TASK DL: 5
USER TASK DL: 0
STATION MANAGER TASK DL: 0
TOTAL TRAIN TASK DL: 0
```

Figura 5: Status monitor

3 Implementazione

3.1 Linguaggi e librerie

L'applicazione è interamente sviluppata in C, utilizzando le librerie Allegro 4.4 e pthread. La directory principale del progetto *Trains* ha due sottocartelle: in *SRC* si trovano tutti i file sorgente usati nell'implementazione del progetto e in *img* ci sono i bitmaps utilizzati dall'applicazione. Due cartelle aggiuntive, *bin* e *build*, vengono create al momento della compilazione e contengono rispettivamente i file binary e gli oggetti.

3.2 Strutture dati

Gli stati delle entità utilizzate nell'applicazione sono rappresentati tramite diverse strutture dati globali, tutte definite nel file *init.h*, di seguito vengono elencate le più importanti:

- *train parameters* sono strutture dati associate ad ogni task treno e contengono tutte le informazioni sullo stato corrente del treno. Gli altri task possono interagire leggendo/modificando i dati presenti in questa struttura.
- *station struct* sono strutture analoghe associate ad ogni stazione o semaforo. Contengono informazioni come lo status del semaforo, gli *id* dei treni in coda o la posizione del binario mobile.
- *buttons struct* sono strutture associate ad ogni pulsante, contengono info sulla sua posizione nella finestra grafica e lo stato del pulsante.
- *task par* (dichiarate in *ptask.h*) sono strutture dati associate ad ogni thread in real-time e contengono tutte le info necessarie al corretto funzionamento di questi ultimi.

Per poter correttamente utilizzare queste strutture dati globali vengono utilizzati dei *mutex* che vengono bloccati ogni volta che una funzione legge/modifica i dati presenti nelle strutture.

3.3 Task

Durante l'esecuzione del programma vengono eseguiti molteplici threads periodici in real-time:

graphics gestisce tutta la parte grafica, disegnando sullo schermo treni, binari e semafori nella loro posizione attuale.

user è il task associato alle interazioni dell'utente con il programma tramite mouse o tastiera. In particolare questo task si occupa di creare i task treno.

station manager è il task associato alla stazione e si occupa di assegnare i binari ai nuovi treni, decidere lo status dei semafori, spostare i binari mobili nella direzione corretta e gestire l'abbandono della stazione.

train sono i task associati ad ogni nuovo treno. Si occupano di gestire il moto del treno in modo coerente da quanto viene ordinato da parte dello station manager. Segue il percorso del binario assegnatogli e legge lo status dei semafori che si trova davanti per decidere se fermarsi o se continuare la sua corsa

La configurazione standard dei parametri di scheduling utilizzati è mostrata in tabella. Questi valori sono stati decisi sia da analisi del sistema che da test empirici. Per esempio il task grafico viene aggiornato ogni 20ms per permettere una simulazione fluida dell'applicazione. Il task utente è meno importante e può avere un periodo di riattivazione più lungo, impostato a 50ms. I task dei treni e il task station manager sono entrambi impostati a 20ms per evitare conflitti.

I valori delle deadline sono tutti impostati a 10ms per permettere di conteggiare qualche deadline miss visto che il programma, relativamente leggero, altrimenti non si sarebbe mai trovato in quella situazione.

Task	Periodo	Deadline	Priorità
Graphics	20	10	255
Station manager	20	10	255
User	50	10	255
Train	20	10	255

In figura 6 è infine possibile vedere un diagramma schematico del funzionamento dei task. La funzione *initialize()* genera i task *graphics*, *station manager* e *user*. Quest'ultimo a sua volta genera i nuovi task treno.

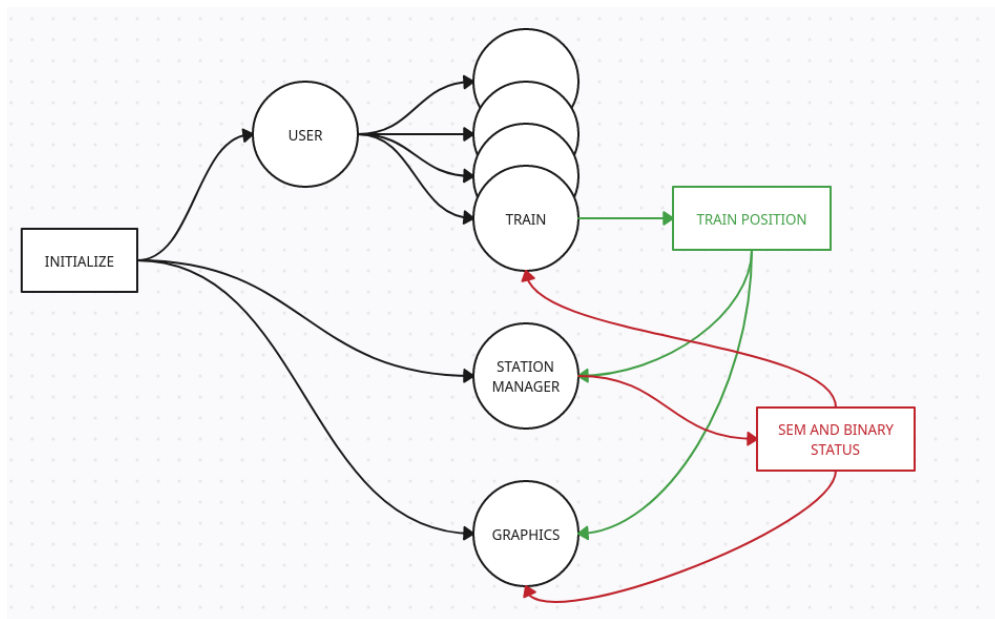


Figura 6: Diagramma dipendenze dei task