



UNIVERSITÀ DEGLI STUDI DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Robotica e
dell'Automazione

INFORMATICA E SISTEMI IN TEMPO REALE


Report di Progetto

T A N K S S i m u l a t o r

E-mail:

livio.bisogni@gmail.com

ORCID iD:

 <https://orcid.org/0000-0002-7471-2022>

Indice

1	INTRODUZIONE.....	3
2	MODELLO FISICO.....	3
2.1	TEMPO CONTINUO	3
2.2	TEMPO DISCRETO.....	4
2.3	SENSORI E CONTROLLORI	4
3	TASK E RISORSE.....	6
3.1	TASK.....	6
3.1.1	<i>Task Serbatoio</i>	6
3.1.2	<i>Task Sensore</i>	6
3.1.3	<i>Task Utente</i>	6
3.1.4	<i>Task Grafico</i>	6
3.1.5	<i>Task per l'attivazione sincrona</i>	7
3.2	RISORSE CONDIVISE	7
3.3	DIAGRAMMA TASK-RISORSE	8
3.4	PARAMETRI DEI TASK	9
4	INTERFACCIA GRAFICA UTENTE	10
4.1	PANNELLO DEI SERBATOI	11
4.2	PANNELLO DEI COMANDI	11
4.3	PANNELLO DEI PARAMETRI.....	12
4.4	PANNELLO DELLE DEADLINE MISS E DEI TEMPI DI RISPOSTA	12
5	WCET	14

1 Introduzione

Scopo del progetto¹ è realizzare un sistema di N serbatoi automatizzati. Ogni serbatoio contiene del liquido la cui altezza viene stimata da un sensore di prossimità posto alla sua cima; inoltre, ogni serbatoio ha una valvola di uscita (la cui percentuale di apertura viene decisa dall'Utente), da cui fuoriesce il liquido, e una valvola di ingresso, da cui invece entra. L'Utente decide l'altezza desiderata a cui vorrebbe che fosse il liquido, mentre un controllore si occuperà di modificare opportunamente l'apertura della valvola di ingresso nel tempo, fino al raggiungimento (e mantenimento) di tale valore.

Il numero di serbatoi N , $N \in \mathbb{N}$, è impostabile dall'Utente², tuttavia è fortemente consigliato simularne 5. Infatti, 5 è al contempo un numero che è stato reputato adeguato a simulare molteplici livelli desiderati e flussi di uscita contemporaneamente su più serbatoi, senza tuttavia risultare un numero troppo dispersivo per l'Utente. Infine, l'interfaccia grafica è stata accuratamente progettata per ottimizzare al meglio la visione di 5 serbatoi, opzione preferita rispetto al mostrarne un numero maggiore ma più in piccolo.

2 Modello fisico

Il sistema è costituito da N serbatoi: ognuno di essi ha un proprio trasduttore, un proprio controllore e un proprio attuatore. L'intero sistema è stato ovviamente realizzato a tempo discreto. Per illustrarne il modello usato conviene partire dalle equazioni a tempo continuo, dopodiché discretizzarle, e infine modificarne le variabili di interesse tramite l'utilizzo di un controllore.

2.1 TEMPO CONTINUO

I serbatoi sono stati modellati come cilindri di raggio r e altezza h_{max} , per cui la superficie della sezione trasversale di ciascun serbatoio è pari ad $A = \pi r^2$.

Si consideri d'ora in avanti un generico i -imo serbatoio, $i = 1, \dots, N$. Sia $h(t)$ l'altezza a cui si trova il liquido contenuto al suo interno, e sia invece $h_d(t)$ l'altezza desiderata³, presa come riferimento. L'equazione di bilancio della materia impone che, ad ogni generico istante t , valga:

$$A \cdot \frac{dh(t)}{dt} = q_{in}(t) - q_{out}(t) \quad (1)$$

dove $q_{in}(t)$ e $q_{out}(t)$ rappresentano rispettivamente la portata volumetrica in ingresso e in uscita. Le portate si suppongono proporzionali al livello $h(t)$. La massima portata ottenibile in ingresso, che riesce a garantire un sufficiente riempimento del serbatoio anche in condizioni di valvola di scarico totalmente aperta, è:

$$q_{in,max} = k_{in} \cdot h_{max} \quad (2)$$

¹ Di seguito si riporta la richiesta originale: "Simulate N tanks that have to maintain the liquid at a desired level. Each tank has an output tap at the bottom to get the liquid and an input tap at the top to add new liquid. The liquid level is acquired by a proximity sensor located on the top. Each input tap is automatically controlled by a periodic task, while output taps are controlled by the user through the mouse."

² Modificando l'header file `init.h`.

³ La lettera d sta per *desired*.

Si ha quindi che la portata di ingresso è:

$$q_{in}(t) = \Phi_{in}(t) \cdot q_{in,max} \quad (3)$$

dove Φ_{in} è l'apertura della valvola di ingresso, mentre la portata di uscita è:

$$q_{out}(t) = \Phi_{out}(t) \cdot k_{out} \cdot h(t) \quad (4)$$

dove Φ_{out} e k_{out} sono rispettivamente l'apertura e la costante della valvola di scarico. Si noti che Φ_{out} è modificabile dall'operatore addetto all'impianto, mentre Φ_{in} viene automaticamente variato dal controllore, descritto nel resto della sezione.

2.2 TEMPO DISCRETO

D'ora in avanti, al fine di non appesantire la scrittura, data una generica funzione dipendente dal tempo x , $x : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$, si userà la notazione $x(t_k)$ per indicare $x(k \cdot T_s)$, dove k , $k \in \mathbb{N}$, denota il k -imo istante di tempo (discreto) e T_s , $T_s \in \mathbb{R}_{>0}$, è il periodo di campionamento scelto⁴.

L'Equazione (1) discretizzata diventa:

$$A \cdot \frac{h(t_k) - h(t_{k-1})}{T_s} = q_{in}(t_k) - q_{out}(t_k) \quad (5)$$

Esplicitando l'altezza e sostituendo le Equazioni (3) e (4), si ottiene:

$$h(t_k) = \frac{\Phi_{in}(t_k) \cdot q_{in,max} \cdot T_s + A \cdot h(t_{k-1})}{A + \Phi_{out}(t_k) \cdot k_{out} \cdot T_s} \quad (6)$$

2.3 SENSORI E CONTROLLORI

Il livello di un generico i -imo serbatoio viene acquisito da un sensore di prossimità situato in cima al serbatoio. Sia $h_m(t_k)$ l'altezza del liquido all'istante t_k misurata⁵ tramite il trasduttore. L'errore che si vuole minimizzare⁶ è la discrepanza tra il riferimento e il valore misurato:

$$e(t_k) = h_d(t_k) - h_m(t_k) \quad (7)$$

La portata d'uscita q_{out} viene modificata da una persona fisica che agisce sulla valvola di scarico Φ_{out} . Poiché si vuole portare (e mantenere) il liquido al livello desiderato, è necessario variare opportunamente la portata d'ingresso q_{in} . Per regolare il sistema descritto in Equazione (6) è possibile usare un controllore PID che agisca sulla valvola di ingresso Φ_{in} basandosi su 3 azioni in parallelo, qui espresse in forma incrementale:

- una azione proporzionale

$$a_p(t_k) = K_p \cdot \frac{[e(t_k) - e(t_{k-1})]}{h_{max}} \quad (8)$$

- una azione integrale

$$a_i(t_k) = K_i \cdot T_s \cdot \frac{e(t_k)}{h_{max}} \quad (9)$$

⁴ Il periodo di campionamento è stato scelto come multiplo del periodo dei task che si occupano della modellizzazione dei serbatoi (i *tank_task*, illustrati successivamente), e risulterà essere pari a 20 millisecondi.

⁵ La lettera m sta per *measured*.

⁶ Si osservi che – per rendere più realistico il simulatore – nel calcolo dell'errore si considera il livello di liquido rilevato dal sensore (i.e., $h_m(t_k)$), non il livello di liquido realmente presente nel serbatoio (i.e., $h(t_k)$).

- una azione derivativa

$$a_d(t_k) = \frac{K_d}{T_s} \cdot \frac{[e(t_k) - 2 \cdot e(t_{k-1}) + e(t_{k-2})]}{h_{max}} \quad (10)$$

dove K_p, K_i e K_d sono rispettivamente il guadagno proporzionale, integrale e derivativo. Complessivamente, il contributo delle tre azioni del controllore PID è:

$$\Delta u(t_k) = a_p(t_k) + a_i(t_k) + a_d(t_k) \quad (11)$$

Si noti che l'algoritmo è espresso in forma incrementale (anziché posizionale/assoluta), ossia il valore in uscita dal controllore PID ($\Delta u(t_k)$) rappresenta l'incremento al precedente valore di controllo ($u(t_{k-1})$); pertanto la variabile di controllo è:

$$u(t_k) = u(t_{k-1}) + \Delta u(t_k) \quad (12)$$

Se si hanno inoltre delle conoscenze sulla posizione e sulla costante della valvola di scarico all'istante t_k è possibile introdurre anche una azione di feedforward:

$$a_f(t_k) = \frac{\Phi_{out}(t_k) \cdot k_{out} \cdot h_d(t_k)}{q_{in,max}} \quad (13)$$

Considerando tutti i contributi pesati, il controllore complessivo diventa:

$$u(t_k) = (1 - f) \cdot [u(t_{k-1}) + \Delta u(t_k)] + f \cdot a_f(t_k), \quad f \in [0,1] \quad (14)$$

ossia è la combinazione convessa del PID e del feedforward al variare del coefficiente di feedforward f , dove $f = 0$ equivale a usare solo il PID mentre $f = 1$ equivale a usare solo la parte di feedforward.

Al fine di ridurre le variazioni più piccole di attuazione sono state inserite 3 condizioni sulla variabile di controllo e sull'errore:

1. $|u(t_k) - \Phi_{in}(t_{k-1})| < \delta_1 \wedge |e(t_k) - e(t_{k-1})| < \delta_2;$
2. $|u(t_k) - \Phi_{in}(t_{k-1})| < \delta_3;$
3. $|e(t_k) - e(t_{k-1})| < \delta_4;$

dove $\delta_1, \delta_2, \delta_3, \delta_4 \in \mathbb{R}_{\geq 0}$ sono delle tolleranze (tipicamente molto piccole). Qualora all'istante t_k anche una sola condizione sia verificata, l'azione di controllo rimane invariata rispetto all'istante precedente: $u(t_k) = \Phi_{in}(t_{k-1})$.

Il segnale in uscita dal controllore viene quindi filtrato da un filtro a media mobile con finestra pari a W campioni⁷ e pesi uniformi (pari a 1). Sia u il segnale di ingresso al filtro e u_W il segnale in uscita dal filtro. Siano quindi $u(t_{k-W}), \dots, u(t_{k-1})$ i W valori passati del segnale di controllo mentre $u(t_k)$ il valore attualmente proposto. Allora l'uscita del filtro all'istante k -imo è:

$$u_W(t_k) = \frac{1}{W+1} \cdot \sum_{j=-W}^0 u(t_{k+j}), \quad W \in \mathbb{N} \quad (15)$$

Questo è un semplice filtro passa-basso che ha la funzione di "addolcire" il segnale di controllo: facendo la media con i precedenti W campioni, riesce a smussare le variazioni più veloci. È inoltre facilmente verificabile che, con una finestra nulla ($W = 0$), il filtro non modifica il segnale ($u_W(t_k) = u(t_k)$).

Adesso il segnale di controllo è finalmente pronto per essere attuato. Per farlo, bisogna considerare che la valvola di ingresso può assumere valori solo in un determinato intervallo: quello compreso tra le posizioni limite di

⁷ La W sta per *window size*. Tale filtro è stato implementato usando un buffer circolare.

funzionamento 0 (0%, valvola totalmente chiusa) e 1 (100%, valvola totalmente aperta); pertanto per rendere più realistica la simulazione è necessario saturare il valore da attuare:

$$\Phi_{in}(t_k) = \begin{cases} 1, & \text{se } u_W(t_k) > 1 \\ u_W(t_k), & \text{se } 0 \leq u_W(t_k) \leq 1 \\ 0, & \text{se } u_W(t_k) < 0 \end{cases} \quad (16)$$

Si noti infine che, al fine di gestire situazioni di overflow (i.e., $h(t_k) > h_{max}$) e di underflow (i.e., $h(t_k) < 0$), anche il livello reale del liquido all'istante t_k dato dall'Equazione (6) viene saturato:

$$h(t_k) = \begin{cases} h_{max}, & \text{se } h(t_k) > h_{max} \\ h(t_k), & \text{se } 0 \leq h(t_k) \leq h_{max} \\ 0, & \text{se } h(t_k) < 0 \end{cases} \quad (17)$$

3 Task e risorse

L'insieme dei task (task set) Γ è costituito da n task periodici e 1 task aperiodico a esecuzione singola (single job): $\Gamma = \{\tau_1, \dots, \tau_n, \tau_{n+1}\}$, dove il numero di task periodici è pari a $n = 2 \cdot (N + 1)$.

In particolare, l'applicazione genera:

- N task serbatoio periodici (*tank_task*): τ_1, \dots, τ_N ;
- N task sensore periodici (*sensor_task*): $\tau_{N+1}, \dots, \tau_{2N}$;
- 1 task utente periodico (*user_task*): τ_{2N+1} ;
- 1 task grafico periodico (*graphics_task*): τ_{2N+2} ;
- 1 task a single job per l'attivazione sincrona (*activate_all_tasks*): τ_{n+1} .

3.1 TASK

Adesso verrà brevemente illustrato lo scopo di ognuno.

3.1.1 Task Serbatoio

Gli N task serbatoio si occupano sia di simulare la fisica del sistema (Equazioni (5), (6) e (17)) che di controllarlo (Equazioni (7), (8), (9), (10), (11), (12), (13), (14), (15) e (16)), sfruttando i livelli misurati dai task sensore.

3.1.2 Task Sensore

Ciascun task sensore si occupa di misurare l'altezza del liquido contenuto all'interno del rispettivo serbatoio. Si è scelto di simulare un sensore di prossimità rilevando la posizione del primo pixel di colore blu: la distanza dal sensore (posto nell'estremità sinistra del serbatoio) a tale pixel viene quindi convertita in unità espresse nel sistema MKS, ottenendo così $h_m(t_k)$. Il valore misurato viene infine usato dal controllore del corrispondente task serbatoio.

3.1.3 Task Utente

Il task utente gestisce l'interazione con l'Utente, il quale può interagire sia attraverso la tastiera che attraverso il mouse. Per ulteriori approfondimenti sui comandi gestiti si rimanda alla Sezione 4.

3.1.4 Task Grafico

Il task grafico si occupa di generare la grafica per l'intera applicazione, che viene aggiornata ad ogni frame rate. Al fine di velocizzarne la computazione è stato implementato un meccanismo di double-buffering e,

inoltre, gli elementi visivi statici (ossia quelli non varianti nel tempo) vengono calcolati una volta soltanto (all'apertura dell'applicazione).

3.1.5 Task per l'attivazione sincrona

Affinché l'insieme Γ parta in contemporanea – idealmente, allo stesso istante – viene generato un task fittizio (con priorità superiore a quella del *main task*) il cui scopo è soltanto quello di avviare i task periodici all'unisono; dopodiché il task termina.

3.2 RISORSE CONDIVISE

Al fine di consentire la comunicazione tra i vari task vengono create delle risorse condivise. Queste si possono suddividere in due macrocategorie:

- le variabili globali che incidono sul comportamento del simulatore, ossia le strutture dati per l'interazione inter-task;
- i mutex per proteggere le operazioni di lettura e scrittura delle suddette risorse.

Viene creato un mutex per ogni struttura dati condivisa, e ciascuno di essi adotta il Priority Inheritance Protocol. Nel programma non sono presenti sezioni critiche annidate, per cui si evita la possibilità di avere deadlock (ossia bloccaggi reciproci), che è uno dei potenziali svantaggi del Priority Inheritance Protocol.

In Tabella 1 vengono riportate tutte le risorse condivise usate. Il flag *end*, modificato solo una volta dal *main task* alla fine della simulazione, non è protetto. Si noti che le dimensioni dei serbatoi, i parametri dei controllori e dei filtri a media mobile, e le costanti delle valvole di uscita, sono sì modificabili, ma univoci per tutti i serbatoi. Infatti, rendere tali parametri diversi per ogni serbatoio (semplicemente, ponendoli come campi dello stato di tale serbatoio) avrebbe solo complicato la *user experience* senza aggiungere alcun vantaggio pratico.

Tabella 1. Risorse condivise tra i task.

	Nome variabile	Tipo variabile	Scopo variabile
Variabili globali del simulatore	<i>end</i>	int	Flag per la terminazione del simulatore
	<i>tank</i>	tank_attr	È l'array di stati di N serbatoi; ciascuno di essi contiene, tra l'altro: h , h_m , h_d , Φ_{in} e Φ_{out}
	<i>pid</i>	pid_controller	Guadagni PID e percentuale di feedforward: K_p , K_i , K_d e f
	<i>dim</i>	dimension	Dimensione dei serbatoi: r e h_{max}
	<i>k</i>	double	Costante delle valvole di scarico: k_{out}
	<i>w</i>	int	Numero di campioni della finestra del filtro a media mobile: W

	<i>util_scale</i>	double	Fattore di scala per rimpicciolire (zoom out) / ingrandire (zoom in) la visualizzazione delle barre di utilizzo
Mutexes	<i>mux</i>	pthread_mutex_t	Proteggono gli <i>N tank</i>
	<i>mux_pid</i>	pthread_mutex_t	Protegge <i>pid</i>
	<i>mux_dim</i>	pthread_mutex_t	Protegge <i>dim</i>
	<i>mux_k</i>	pthread_mutex_t	Protegge <i>k</i>
	<i>mux_w</i>	pthread_mutex_t	Protegge <i>w</i>
	<i>mux_util</i>	pthread_mutex_t	Protegge <i>util_scale</i>
	<i>mux_print</i>	pthread_mutex_t	Protegge le stampe su terminale

3.3 DIAGRAMMA TASK-RISORSE

Per facilitare la gestione dei task è stato costruito un diagramma che schematizza le interazioni tra i task e le risorse globali. Al fine di compattare al meglio lo schema, per le risorse condivise si è usata la notazione introdotta nella Sezione 2 (e impiegata anche nella Sezione 3, Tabella 1), mentre per quanto riguarda i task si è adottata la seguente notazione:

- $\tau_{T,i}$, $i = 1, \dots, N$, è l' i -imo task serbatoio (*tank_task*);
- $\tau_{S,i}$, $i = 1, \dots, N$, è l' i -imo task sensore (*sensor_task*);
- τ_U è il task utente (*user_task*);
- τ_G è il task grafico (*graphics_task*).

In Figura 1 è riportato tale diagramma. Per avere un ingrandimento superiore dell'immagine è possibile visionare il file "*Bisogni_tanks_task-resource_diagram.pdf*".

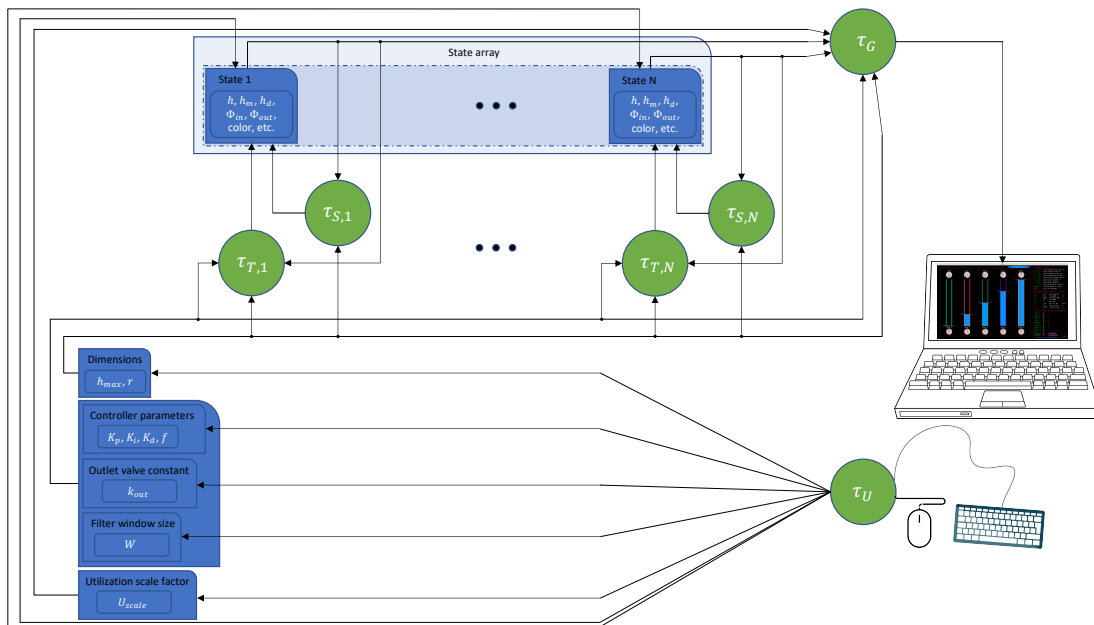


Figura 1. Diagramma task-risorse.

3.4 PARAMETRI DEI TASK

Per facilitare la gestione dei task e del tempo è stata sviluppata anche una libreria; per non dilungarsi troppo si rimanda il Lettore al suo codice, che è stato largamente commentato. È comunque opportuno esplicitare che, per la generazione dei task (tramite la funzione *task_create*) viene richiamata la funzione *pthread_create* in cui viene usato – per tutti quanti i task – lo scheduler *SCHED_RR*, che adotta uno scheduling a priorità fissa (con 99 livelli di priorità⁸: 1, ..., 99) e che, in caso di task con priorità identica, li gestisce immettendoli in una coda a politica Round-Robin.

Per ciascun task periodico τ_j , $j = 1, \dots, n$, i parametri modificabili sono:

1. il periodo T_j , $T_j \in \mathbb{N}^+$ (espresso in millisecondi);
2. la deadline relativa D_j , $D_j \in \mathbb{N}^+$ (espressa in millisecondi);
3. la priorità P_j , $P_j \in \{1, \dots, 99\}$.

Sono state effettuate le seguenti scelte:

1. I periodi sono stati scelti tutti pari a 40 ms. Infatti:
 - a. per motivi di visualizzazione grafica delle utilizzazioni stimate – abbondantemente trattate nella Sezione 4.4 – è risultato pratico avere tutti i task con lo stesso periodo;
 - b. i task sensore a ogni esecuzione cercano nel buffer video i pixel di un dato colore lungo una certa direzione: è quindi sensato che i task sensore e il task grafico abbiano lo stesso periodo;
 - c. i task serbatoio leggono il dato più recente calcolato dai (rispettivi) task sensore: è ragionevole che i task serbatoio e i task sensore lavorino allo stesso periodo;
 - d. il task utente è un task sporadico che rileva e gestisce gli interrupt generati dal mouse e dalla tastiera: viene trattato come un task periodico in cui il periodo è posto pari al minimo tempo di interarrivo. La scelta $T_{2N+1} = MIT_{2N+1} = 40$ ms appare come un valore ragionevolmente basso;
 - e. tali periodi consentono di rispettare i $WCET_j$ stimati⁹;
 - f. curiosità: $1/(0.040 \text{ s}) = 25$ fps (frames per second) è lo standard usato nei sistemi di codifica PAL e SÉCAM, in passato utilizzati dalle televisioni analogiche europee.

Si osservi peraltro come, con tale scelta, Γ risulti essere un insieme di task armonico: $\forall j, l = 1, \dots, n \exists k \in \mathbb{N}^+ : T_j = k \cdot T_l \vee T_l = k \cdot T_j$.

2. Le deadline relative sono state poste pari ai rispettivi periodi, ossia: $D_j = T_j \quad \forall j = 1, \dots, n$. Sia $a_{j,k} = \phi_j + (k - 1) \cdot T_j$ il tempo di attivazione del k -imo job del j -imo task, dove la fase ϕ_j si può supporre sempre nulla, i.e. si suppongono attivazioni sincrone ($\phi_j = 0$)¹⁰. Allora la deadline

⁸ Qualora si voglia rendere l'applicazione compatibile con lo standard *POSIX* il livello massimo consentito è 32 (anziché 99).

⁹ Sono stati calcolati i $WOET_j$ ottenuti dopo $M = 10^5$ esecuzioni per ogni tipologia di task (di volta in volta diversa) eseguente singolarmente sul sistema. Per condurre questa analisi si è posto $N = 1$, per cui i $WCET_j$ dei task serbatoio e dei task sensore sono poi stati moltiplicati per un fattore 5. I risultati sono riportati nella Sezione 5.

¹⁰ Assunzione realistica in quanto, come scritto nella Sezione 3, tutti gli n task periodici vengono generati da un task a single job a priorità più alta di tutti (e, pertanto, più alta anche del *main task*).

assoluta del k -imo job del j -imo task è data da: $d_{j,k} = a_{j,k} + D_j \quad \forall j = 1, \dots, n \quad \forall k \in \mathbb{N}^+$.

3. Per quanto riguarda le priorità:

- tutti i task sensore hanno priorità massima tra quelle assegnate: $P_j = 5 \quad \forall j = N + 1, \dots, 2N$;
- tutti i task serbatoio hanno priorità inferiore a quella dei sensori, in modo da poter utilizzare i dati più recenti provenienti da essi: $P_j = 4 \quad \forall j = 1, \dots, N$;
- il task utente, non strettamente essenziale ma al contempo non energivoro, ha priorità più bassa ma superiore a quella del task grafico: $P_{2N+1} = 3$;
- il task grafico, essendo il più esigente dal punto di vista delle richieste di calcolo, ha la priorità minima tra tutte quelle assegnate: $P_{2N+2} = 2$;
- infine, il task aperiodico a esecuzione singola ha priorità massima: $P_{n+1} = 99$.

Per quanto queste possano apparire come scelte conservative, tali valori sono in grado di garantire una piacevole esperienza utente.

4 Interfaccia grafica utente

L'interfaccia grafica dell'applicazione, mostrata in Figura 2, è suddivisa in 4 macro-zone:

1. il pannello dei serbatoi simulati (a sinistra, pannello più grande);
2. il pannello dei comandi (a destra, in alto);
3. il pannello dei parametri (a destra, al centro);
4. il pannello delle deadline miss e dei tempi di risposta (a destra, in basso).

Nel resto della sezione verranno brevemente illustrati.

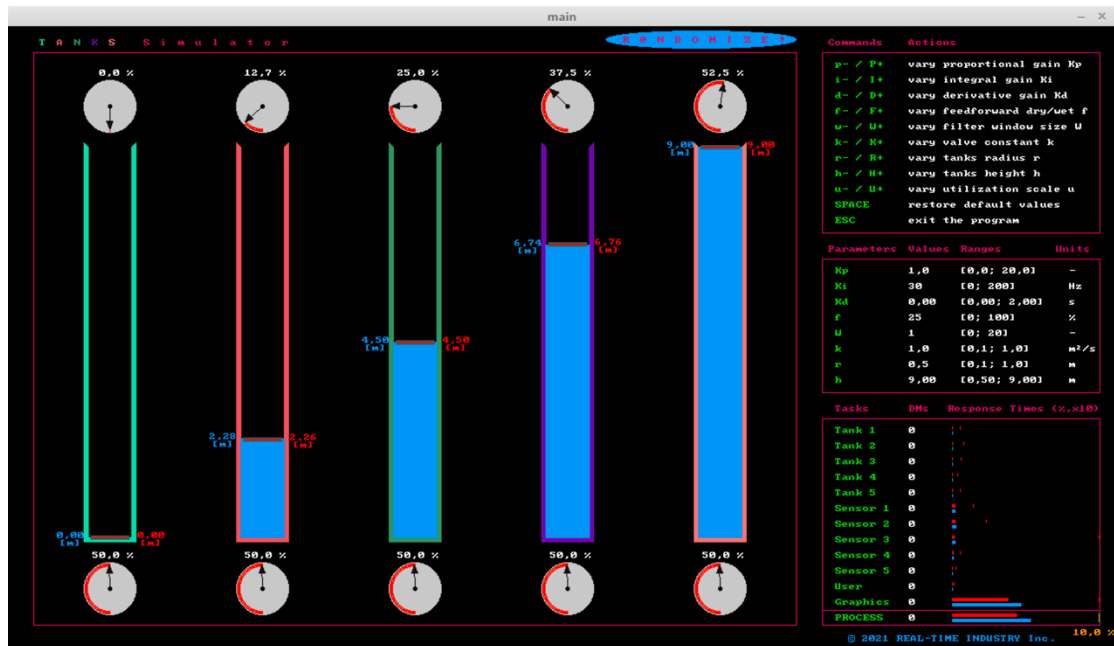


Figura 2. GUI del Tanks Simulator.

4.1 PANNELLO DEI SERBATOI

Nel Pannello dei serbatoi (Figura 3) vengono mostrati gli N serbatoi. Per ognuno di essi è possibile notare che:

- in cima si trova una manopola, la cui apertura (i.e., $\Phi_{in}(t_k)$) viene automaticamente determinata dal controllore, che quindi influenza la portata volumetrica in ingresso;
- è contenuto del liquido bluastro: alla sua sinistra compare un valore in blu, che denota il livello misurato dal sensore del liquido (i.e., $h_m(t_k)$; in generale diverso da $h(t_k)$);
- è presente un fader (rettangolo rosso-nero): alla sua destra è scritto un valore in rosso, che denota il livello di liquido correntemente desiderato (i.e., $h_d(t_k)$) e che viene impostato scorrendo il fader con il mouse;
- in basso si trova la valvola di scarico: l'Utente può variarne l'apertura (i.e., $\Phi_{out}(t_k)$) sempre tramite mouse.

L'Utente può inoltre variare tutti questi parametri contemporaneamente in maniera pseudo-casuale premendo il bottone "r@ndomize!" (*Randomizer button*). Ciò assegnerà anche un nuovo colore ai serbatoi, anch'esso generato pseudo-casualmente.

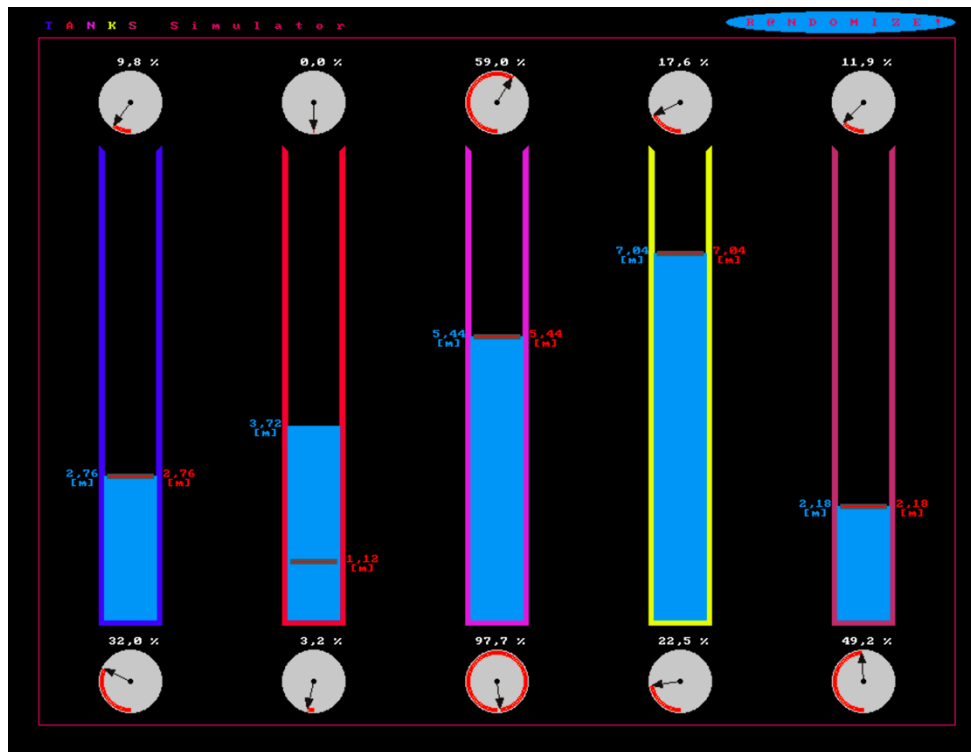


Figura 3. Pannello dei serbatoi.

4.2 PANNELLO DEI COMANDI

Il Pannello dei comandi (Figura 4) dà informazioni all'Utente su come variare alcuni parametri del simulatore tramite tastiera: nella colonna di sinistra è riportato il tasto da premere, mentre in quella di destra è riportata l'azione corrispondente.

Commands	Actions
p- / P+	vary proportional gain Kp
i- / I+	vary integral gain Ki
d- / D+	vary derivative gain Kd
f- / F+	vary feedforward dry/wet f
w- / W+	vary filter window size W
k- / K+	vary valve constant k
r- / R+	vary tanks radius r
h- / H+	vary tanks height h
u- / U+	vary utilization scale u
SPACE	restore default values
ESC	exit the program

Figura 4. Pannello dei comandi.

4.3 PANNELLO DEI PARAMETRI

Nel Pannello dei parametri (Figura 5) è invece possibile vedere il valore corrente di tali parametri (seconda colonna), l'intervallo in cui ciascuno di essi può variare (terza colonna), e la rispettiva unità di misura (quarta colonna).

Parameters	Values	Ranges	Units
Kp	0,8	[0,0; 20,0]	-
Ki	28	[0; 200]	Hz
Kd	0,06	[0,00; 2,00]	s
f	35	[0; 100]	%
W	2	[0; 20]	-
k	1,0	[0,1; 1,0]	m ² /s
r	0,7	[0,1; 1,0]	m
h	8,94	[0,50; 9,00]	m

Figura 5. Pannello dei parametri.

4.4 PANNELLO DELLE DEADLINE MISS E DEI TEMPI DI RISPOSTA

Nel Pannello delle deadline miss e dei tempi di risposta (Figura 6) per ogni task τ_j (prime n righe della prima colonna) è possibile visualizzare:

- il numero di deadline miss $dm_{j,k}$ (seconda colonna) al tempo corrente t_k ;
- una stima delle utilizzazioni (terza colonna), effettuata calcolando il rapporto tra il tempo di risposta misurato per la k -ima esecuzione $RT_{j,k}$ del j -imo task e il periodo T_j di tale task. Più in dettaglio:

- il rettangolo rosso è una stima dell'utilizzazione istantanea all'istante k -imo

$$U_{j,k} = \frac{RT_{j,k}}{T_j}, \quad j = 1, \dots, n \quad (18)$$

- la barra rossa è una stima dell'utilizzazione istantanea massima all'istante k -imo

$$\hat{U}_{j,k} = \max_{l \in \{1, \dots, k\}} U_{j,l}, \quad j = 1, \dots, n \quad (19)$$

Si noti che $\hat{U}_k \geq \hat{U}_{k-1} \quad \forall k \in \mathbb{N}^+ \setminus \{1\}$.

- il rettangolo blu è una stima dell'utilizzazione media all'istante k -imo

$$\bar{U}_{j,k} = \frac{1}{k} \cdot \sum_{l=1}^k U_{j,l}, \quad j = 1, \dots, n \quad (20)$$

Infine, nell'ultima riga viene riportata la situazione per l'intero processo:

- il numero di deadline mancate in totale $dm_k = \sum_{j=1}^n dm_{j,k}$ (seconda colonna) al tempo corrente t_k ;
- una stima delle utilizzazioni per l'intero processo (terza colonna).

In particolare:

- il rettangolo rosso è la somma delle stime delle utilizzazioni istantanee degli n task all'istante k -imo

$$U_k = \sum_{j=1}^n U_{j,k} \quad (21)$$

- la barra rossa è una stima dell'utilizzazione istantanea massima degli n task all'istante k -imo

$$\hat{U}_k = \max_{l \in \{1, \dots, k\}} U_l \quad (22)$$

- la barra verde è la somma delle stime delle utilizzazioni istantanee massime degli n task all'istante k -imo

$$\hat{\hat{U}}_k = \sum_{j=1}^n \hat{U}_{j,k} \quad (23)$$

Si noti che $\hat{\hat{U}}_k \geq \hat{U}_k \quad \forall k \in \mathbb{N}^+$.

- il rettangolo blu è una stima dell'utilizzazione media degli n task all'istante k -imo

$$\bar{U}_k = \sum_{j=1}^n \bar{U}_{j,k} \quad (24)$$

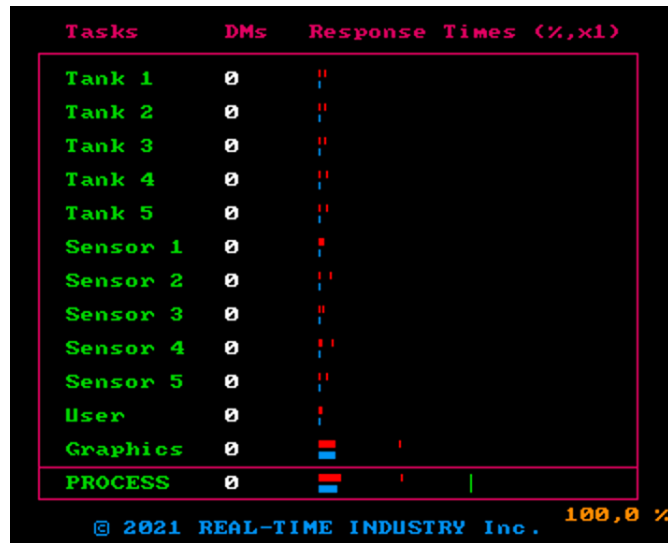


Figura 6. Pannello delle deadline miss e dei tempi di risposta.

5 WCET

Al fine di individuare una plausibile stima del tempo di esecuzione nel caso peggiore di un generico task τ_j , i.e. $WCET_j$ (*Worst-Case Execution Time*), è stata condotta un'analisi sui tempi di calcolo di $M = 10^5$ esecuzioni, ottenendo così – per ogni tipologia di task – i tempi di risposta $RT_{j,k}$, dove $k, k = 1, \dots, M$, è l'indice di esecuzione.

Prima di procedere con l'analisi è utile introdurre alcune definizioni¹¹:

- il miglior tempo di esecuzione osservato (*Best Observed Execution Time*)

$$BOET_j = \min_{k \in \{1, \dots, M\}} RT_{j,k}, \quad j = 1, \dots, n \quad (25)$$

- il peggior tempo di esecuzione osservato (*Worst Observed Execution Time*)

$$WOET_j = \max_{k \in \{1, \dots, M\}} RT_{j,k}, \quad j = 1, \dots, n \quad (26)$$

- il tempo di esecuzione medio (*Average Observed Execution Time*)

$$AOET_j = \frac{1}{M} \cdot \sum_{k=1}^M RT_{j,k}, \quad j = 1, \dots, n \quad (27)$$

- la deviazione standard

$$\sigma_j = \sqrt{\frac{1}{M-1} \cdot \sum_{k=1}^M (RT_{j,k} - AOET_j)^2}, \quad j = 1, \dots, n \quad (28)$$

Si noti che σ_j è stata calcolata usando la correzione di Bessel, ossia dividendo lo scarto quadratico per $M - 1$ anziché per M .

Definite queste quantità e implementate nella libreria *MyPtask*, si è proceduto nel seguente ordine:

1. si è ridotto il numero di serbatoi – e di conseguenza anche di sensori – a 1, ossia si è posto $N = 1$: si ha così 1 task serbatoio (τ_1), 1 task sensore (τ_2), 1 task utente (τ_3), e 1 task grafico (τ_4);
2. a ciascuna delle $n = 4$ esecuzioni del simulatore, ogni task esegue almeno una volta (all'inizio; utile ai fini di inizializzazione¹²). Tuttavia, 1 solo task (di volta in volta diverso) esegue M volte, ossia: $\tau_j, j \in \{1, \dots, n\}$, esegue per M volte, mentre $\{\tau_l : l \in \{1, \dots, n\} \wedge l \neq j\}$ eseguono solo 1 volta (all'apertura dell'applicazione);
3. il punto 2 viene ripetuto – cambiando il task τ_j di volta in volta – fino a che tutti e 4 i task non hanno eseguito M volte continuativamente.

Grazie alle scelte effettuate nei punti 1 e 2 si riesce a garantire che – per il task in esecuzione (τ_j) – vengono eliminati:

- le interferenze di task a priorità maggiore che fanno preemption;
- i bloccaggi di task a priorità inferiore che utilizzano le stesse risorse condivise.

¹¹ Supponendo tempi di risposta coincidenti con i tempi di esecuzione.

¹² Ad esempio, il task sensore richiede che il liquido del corrispondente serbatoio sia memorizzato sul buffer video, al fine di poter rilevare il primo pixel di colore del liquido.

In Tabella 2 vengono riportati i valori ottenuti. Sono stati calcolati, rispettivamente, per un (singolo) task serbatoio, un (singolo) task sensore, un task utente, e un task grafico. Infine, nelle ultime 20 righe della tabella, vengono riportati i 20 tempi di esecuzione maggiori (ossia i 20 peggiori), ordinati in ordine decrescente. I $WOET_j$ così ottenuti sono una ragionevole stima (per difetto) dei $WCET_j$.

Tabella 2. Analisi di tempi di calcolo per $M = 10^5$ esecuzioni.

	Task serbatoio	Task sensore	Task utente	Task grafico
$WOET_j$ [ms]	0.208178	1.358188	0.729771	19.264059
$AOET_j$ [ms]	0.000710	0.022216	0.000132	1.397963
$BOET_j$ [ms]	0.000430	0.019434	0.000031	0.995599
σ_j [ms]	0.001198	0.012666	0.002478	0.290638
I 20 tempi di esecuzione di τ_j peggiori [ms]	0.208178	1.358188	0.729771	19.264059
	0.112895	0.839436	0.273364	13.017983
	0.098171	0.757392	0.039746	12.324868
	0.091970	0.725529	0.031039	11.281874
	0.088783	0.599602	0.020305	11.202346
	0.078125	0.455512	0.019477	10.594229
	0.073934	0.445934	0.018015	9.411926
	0.066733	0.418360	0.017727	8.496800
	0.064216	0.402923	0.017518	8.367287
	0.058696	0.392693	0.016364	8.194718
	0.057665	0.379021	0.014898	8.045096
	0.049298	0.378137	0.013237	7.929624
	0.036092	0.363051	0.009394	7.874197
	0.034076	0.362972	0.009139	7.576037
	0.031473	0.357383	0.007930	7.519869
	0.031109	0.354394	0.006902	7.104203
	0.029056	0.341079	0.006682	7.015666
	0.028735	0.334806	0.004768	6.949978
	0.026765	0.332548	0.004431	6.867835
	0.026102	0.332137	0.004403	6.684919

In generale, tali risultati vengono stampati su terminale – per ogni task τ_j – al termine di ogni simulazione; un esempio è riportato in Figura 7. Tuttavia, i tempi di risposta potrebbero essere alquanto distanti dai tempi di calcolo effettivi, per via di task a priorità superiore che fanno preemption, di task a priorità inferiore che fanno bloccaggi su risorse condivise, e di maggiore overhead dovuto ai costi di preemption (e.g., tempi di commutazione di contesto non nulli).

```
v@v-Parallels-Virtual-Platform ~/Scrivania/C/TanksProject-master - + x
File Edit View Search Terminal Help

*****
***** WCET ESTIMATION: *****
***** Graphics Task *****
*****

Loop length: 82671 cycles
Deadline misses: 2
NOTE: Times are expressed in milliseconds.
Response time analysis:
  Average (AORT):      1,820858
  Maximum (WORT):      34,920520
  Minimum (BORT):      1,273668
  Total:               150532,141663
  Standard deviation:  0,464532
  Utilization factor:  4,552145 %

20 worst response times:
  1) rt[82616]: 34,920520
  2) rt[79819]: 24,135712
  3) rt[57319]: 22,416414
  4) rt[25819]: 12,704050
  5) rt[63808]: 9,537036
  6) rt[80760]: 9,517341
  7) rt[82587]: 9,319828
  8) rt[58991]: 8,379423
  9) rt[70820]: 8,252726
 10) rt[20291]: 8,187747
 11) rt[63319]: 8,085694
 12) rt[42803]: 8,038885
 13) rt[46521]: 7,970052
 14) rt[69319]: 7,946617
 15) rt[68261]: 7,890043
 16) rt[54126]: 7,748464
 17) rt[50348]: 7,742843
 18) rt[50717]: 7,734707
 19) rt[78320]: 7,603784
 20) rt[61391]: 7,584347

*****
***** TANK SIMULATOR: *****
***** © 2021 REAL-TIME INDUSTRY Inc. *****
*****

Total elapsed time:      3306,919036 s
Utilization factor:      5,17 %

Deadline misses:
  Task 1 (Tank[1]):      0
  Task 2 (Tank[2]):      0
  Task 3 (Tank[3]):      0
  Task 4 (Tank[4]):      0
  Task 5 (Tank[5]):      0
  Task 6 (Sensor[1]):    0
  Task 7 (Sensor[2]):    1
  Task 8 (Sensor[3]):    0
  Task 9 (Sensor[4]):    0
  Task 10 (Sensor[5]):   0
  Task 11 (User):        1
  Task 12 (Graphics):    2
```

Figura 7. Stampa su terminale di alcuni tempi di risposta.