Relazione Assignment n.2

Autori: Acampora Andrea, Accursi Giacomo

Presentazione progetto e obbiettivi

Il progetto prevede l'implementazione di un sistema con Arduino UNO per realizzare esperimenti di fisica tracciando la posizione di un oggetto e calcolandone relative velocità e accelerazioni istantanee.
A tal fine sono stati utilizzati i seguenti sensori e attuatori:

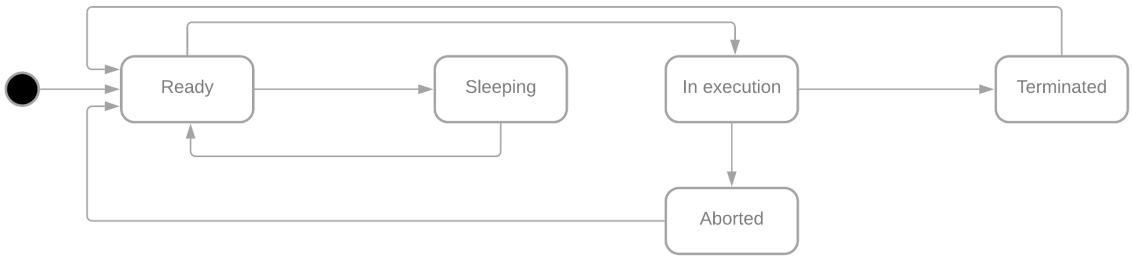
- 2 led per segnalare gli stati del sistema;
- 2 bottoni per gestire l'inizio e la fine dell'esperimento;
- 1 potenziometro per regolare la frequenza di campionamento dell'oggetto da tracciare;
- 1 pir per risvegliare il sistema dal risparmio energetico a fronte di un movimento rilevato;
- 1 sonar per calcolare la distanza dell'oggetto;
- 1 sensore di temperatura, necessario per calibrare il sonar;
- 1 servo motore, utilizzato come tachimetro per mostrare la velocità dell'oggetto in tempo reale;

I dati rilevati saranno inviati alla seriale e catturati da un applicativo Java che si occuperà di visualizzarli graficamente.

Analisi del dominio

In fase di analisi sono stati identificati 5 diversi stati in cui si può trovare il sistema:

- Ready: pronto per l'inizio dell'esperimento.
- Sleeping: in risparmio energetico.
- In execution: durante l'esecuzione dell'esperimento.
- Terminated: quando l'esperimento è stato concluso con successo.
- Aborted: nel caso in cui l'inizializzazione dell'esperimento fallisca.



Progettazione dell'architettura

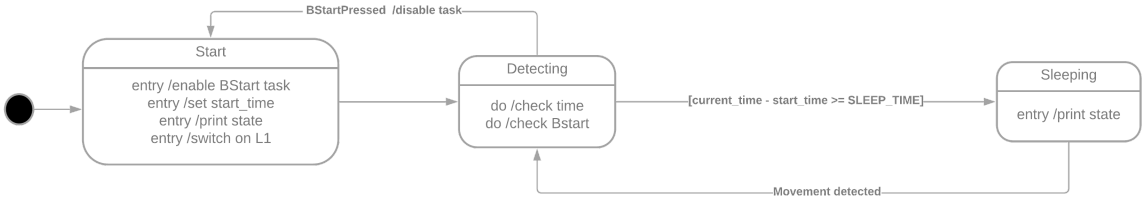
Il sistema è stato realizzato utilizzando una struttura a task.
Ogni task è composto da un piccolo set di istruzioni e l'insieme dei task in esecuzione determina il comportamento del sistema.
Tutti i task creati vengono aggiunti alla coda di uno scheduler non preemptive che li eseguirà sequenzialmente, la posizione nella codà determinerà l'ordine di esecuzione.
Ad ogni task viene attribuito un periodo che determina ogni quanto deve essere eseguito mentre allo scheduler verrà attribuito un periodo uguale al MCD calcolato sui periodi di tutti i task.
Il modello seguito è quello di una macchina a stati finiti sincrona.

Presentazione dettagliata dei task

Partendo dagli stati del sistema sono stati individuati 4 macrotask i quali hanno anche il compito di attivare o disattivare alcuni microtask a seconda della necessità.
E' stata creata una entità TaskManager (anch'esso implementa la classe astratta Task) che conosce l'ordine nel quale i macrotask vengono eseguiti. Questo è sempre attivo e ad ogni tick controlla se il macrotask corrente è ancora attivo, in caso contrario provvederà ad attivare il task successivo.
Il task manager è l'unico task a conoscere la presenza dei macrotask, mentre ognuno di essi è a conoscenza solo dei task di cui necessita.

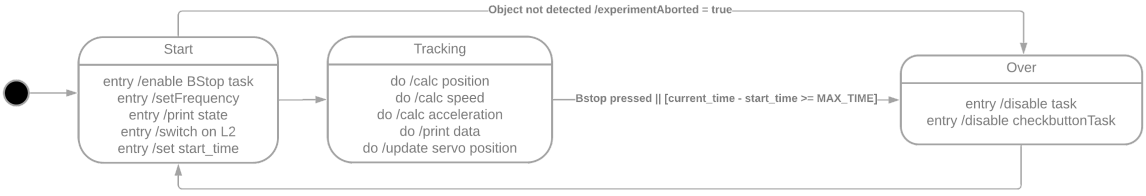
I quattro macrotask individuati sono:

ReadyTask



E' il primo macrotask ad essere mandato in esecuzione; in questa fase il sistema è in attesa che inizi l'esperimento. Al momento dell'attivazione il task entra nello stato START nel quale viene inizializzato il timer, manda lo stato "READY" sulla seriale, accende il Led L1 e attiva il task checkButtonStartTask il quale si occuperà di rilevare la pressione del bottone BStart, dopodichè setterà il proprio stato a DETECTING.
Quando il task si trova nello stato DETECTING ha il compito di controllare se il pulsante è stato premuto(non lo controlla direttamente ma controlla se il task checkButtonStartTask è ancora attivo oppure no) o se il tempo trascorso sia maggiore di SLEEP_TIME. nel caso quest'ultima condizione si avverasse il task passerà allo stato SLEEPING nel quale sono presenti le istruzioni per attivare il risparmio energetico. Solo a fronte di un movimento rilvato dal Pir il sistema si risveglierà e tornerà nello stato START per reinizializzare le variabili.
Nel caso invece di pressione del pulsante, il task checkButtonStartTask provvederà a disattivarsi, di conseguenza il task readyTask si accorgerà dell'evento, si disattiverà e cambierà il suo stato in START per la prossima esecuzione.
Abbiamo deciso di non creare un task per il risparmio energetico ma di considerarlo come uno stato di ReadyTask perchè il sistema in risparmio energetico non svolge alcuna operazione.

InExecutionTask



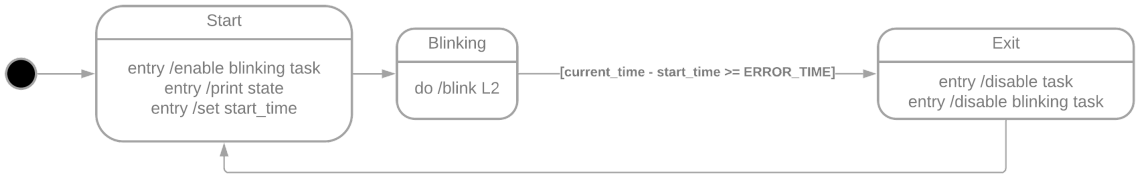
Viene attivato quando l'utente sceglie di avviare l'esperimento e si occupa di tutte le operazioni che concernono il tracciamento dell'oggetto. Al momento dell'attivazione si trova in stato START. Il suo primo compito è verificare che sia presente un oggetto da tracciare, in caso di esito negativo il task passerà in stato OVER e verrà attivato un flag globale che indica che l'esperimento è stato

abortito.
In caso contrario stampa "IN_EXECUTION" sulla seriale, attiva il task checkButtonStopTask per il controllo della pressione del pulsante di stop, accende il Led L2, legge frequenza dal potenziometro e aggiorna il proprio periodo, accende il servo motore, inizializza il timer e infine cambia il proprio stato in TRACKING.
A questo punto inizia il vero e proprio esperimento: ad ogni tick viene calcolata la posizione dell'oggetto, aggiornate la velocità e l'accelerazione istantanea. Questi dati vengono inviati sulla seriale e al servo motore verrà attribuito un nuovo angolo in base alla velocità rilevata. Se durante l'esperimento viene premuto il bottone BStop, il task checkButtonStopTask si disattiva, l'esperimento termina e viene settato lo stato OVER. Ad ogni tick in stato TRACKING viene controllato che il tempo trascorso non sia maggiore di MAX_TIME, anche in questo caso lo stato diventerà OVER.
Lo stato OVER è lo stato finale del task e si occupa di spegnere in servo motore, settare lo stato a START e disattivare il task.

TerminatedTask

Viene attivato al termine dell'esperimento, nel suo stato iniziale START stampa sulla seriale "TERMINATED", attiva blinkingTask il quale si occupa di far lampeggiare il led L2, e cambia il proprio stato in WAITING_FOR_RESTART.
A questo punto il task aspetterà l'OK dell'utente (il quale arriverà sulla seriale) per ricominciare l'esperimento e in caso affermativo cambierà il proprio stato in EXIT nel quale disattiverà il blinkingTask e se stesso, tornando poi allo stato START.
Quando il task manager si accorgerà che questo task si è disattivato provvederà a riattivare il task Ready in modo che l'utente possa effettuare un nuovo esperimento.

Aborted task



Se questo task viene attivato significa che l'esperimento non è stato eseguito per mancata rilevazione di oggetti. Nel suo stato iniziale START inizializza un timer, attiva blinkingTask e passa allo stato BLINKING.
A questo punto il led L2 lampeggia fino a che non è trascorso un tempo maggiore o uguale di ERROR_TIME. Al verificarsi di ciò il task passerà allo stato EXIT nel quale disattiverà blinkingTask e se stesso, tornando poi allo stato START.

Riassumendo i task utilizzati sono:

- TaskManager

- ReadyTask
- InExecutionTask
- AbortedTask
- TerminatedTask
- BlinkingTask
- CheckButtonTask

Note implementative

In fase di progettazione abbiamo notato che le operazioni per il tracciamento dell'oggetto potevano essere scomposte e delegate a più tasks: ad esempio poteva essere creato un task per lettura della posizione ed elaborazione dei dati, un altro per aggiornare l'angolo del servo e uno per scrivere i dati sulla seriale.
Dividere i compiti in questo modo significava utilizzare variabili globali condivise fra tutti i task. Nel nostro sistema invece, abbiamo scelto di trattare tutti i task come unità autonome e indipendenti e mantenere la logica di tracciamento tutta all'interno dello stesso task.

Per quanto riguarda la gestione dei timer all'interno dei tasks non abbiamo utilizzato interrupts ma abbiamo scelto di controllare se il tempo è scaduto ad ogni tick, gestendo quindi le scadenze dei timer in modo sincrono.

Dopo svariate prove empiriche abbiamo notato che a volte il sonar fatica a rilevare la corretta posizione dell'oggetto restituendo valori sbagliati che alterano i dati dell'esperimento. In particolare venivano restituite distanze superiori ad un metro quando l'oggetto era a pochi centimetri di distanza.
Per ovviare a ciò abbiamo deciso di escludere tutti le rilevazioni che restituivano una distanza maggiore di una costante MAX_OBJECT_DISTANCE che nel nostro caso è uguale a un metro. Durante l'esperimento, nel caso in cui l'oggetto non venisse rilevato correttamente o fosse del tutto assente, verrà stampato sulla seriale il messaggio "DATA ERROR".

Le istruzioni del progetto prevedevano una frequenza di campionamento da 1Hz a 50Hz. Nel caso di frequenza massima il periodo del task InExecutionTask veniva settato a 20ms. Dopo aver misurato la durata dell'esecuzione di tutta la coda dei task abbiamo rilevato un durata media di 30/35ms. Di conseguenza per evitare eccezioni di overrun siamo stati costretti a portare il periodo dello scheduler a 40ms e a ridurre la frequenza massima a 49Hz, in modo da avere un periodo minimo del task InExecutionTask di 40 ms.
Tutti gli altri task hanno un periodo che è multiplo di quello dello scheduler.