Esiot-assignment2 report

Smart Car-Washing System

Penazzi Riccardo
0001021734 riccardo.penazzi@studio.unibo.it

Pracucci Filippo
0001020606 filippo.pracucci@studio.unibo.it

Anno accademico 2023/2024

Indice

Capitolo 1

Architettura

Abbiamo realizzato un'architettura basata sul concetto di macchine a stati finiti sincrone (synchronous FSM), dove le valutazioni avvengono periodicamente da parte dello scheduler. Dunque abbiamo realizzato uno scheduler che periodicamente controlla la lista di task (ovvero le FSM), di conseguenza manda in esecuzione il primo task che incontra il cui stato è attivo e il periodo è soddisfatto dal timer dello scheduler. Nel nostro caso specifico alcuni task non richiedevano un periodo specifico, ma semplicemente di essere eseguiti non appena fossero attivi; di conseguenza gli è stato assegnato un periodo simbolico di 1 ms. Il periodo assegnato allo scheduler equivale al massimo comune divisore tra i periodi dei singoli task. La nostra scelta sui task da realizzare si è fondata sulle diverse fasi che compongono il processo di lavaggio. Quindi i task realizzati sono:

- BlinkTask
- PirDetectionTask
- WelcomeTask
- ProceedTask
- PreWashingTask
- WashingTask
- ExitTask

Ogni task si impegna a realizzare esclusivamente i compiti richiesti dalla specifica fase di lavoro, senza interagire direttamente con altre fasi, ma comunicando lo stato complessivo tramite variabili globali. Difatti è compito dello scheduler dare il comando al task opportuno in base allo stato del lavoro, cioè valutando quale task è in quel momento attivo. La maggior parte dei task sono composti da diversi stati, in modo da suddividere il lavoro in base alla fase di lavorazione richiesta; facendo ciò siamo in grado di gestire i periodi e quindi assegnare, se necessario, più volte consecutivamente il controllo allo stesso task che dopo aver svolto una parte del suo lavoro cambia il proprio stato e restituisce il comando allo scheduler. Sarà quindi compito dello scheduler restituire il comando allo stesso task, il quale è rimasto attivo, ma trovandosi in uno stato diverso.

Capitolo 2

Tasks

2.1 BlinkTask

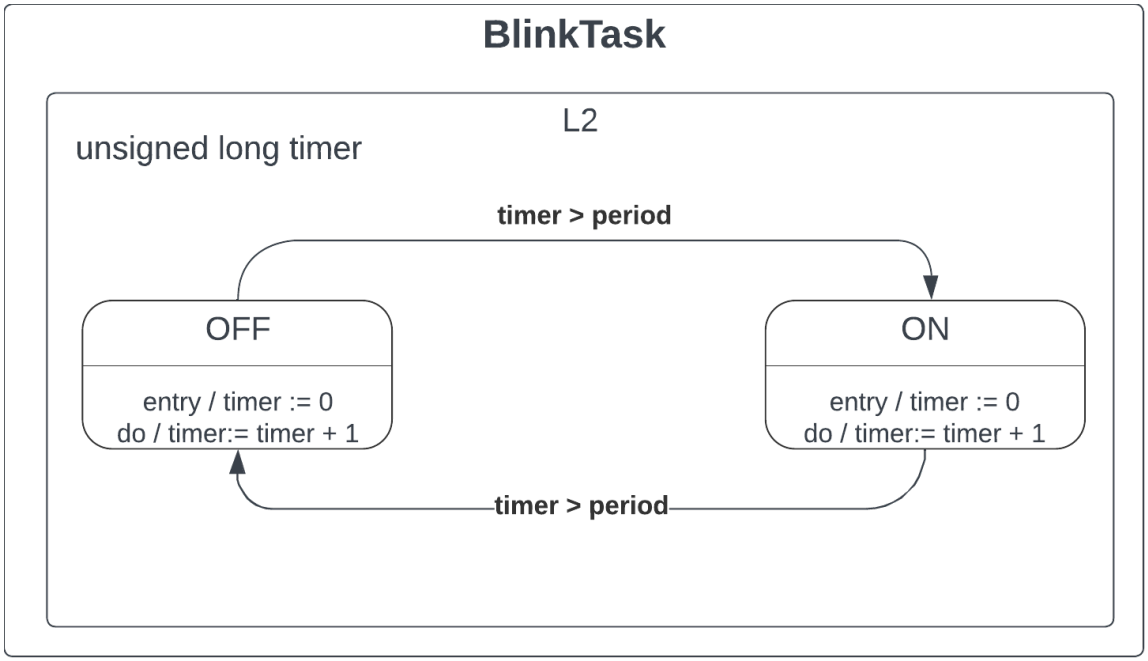

Figura 2.1: BlinkTask FSM

BlinkTask è costituito da due stati: *on* e *off*. Il primo si occupa di spegnere il led nel caso in cui il tempo trascorso, misurato tramite un timer, è maggiore del periodo. Lo stato *off* è speculare rispetto al primo.

2.2 PirDetectionTask

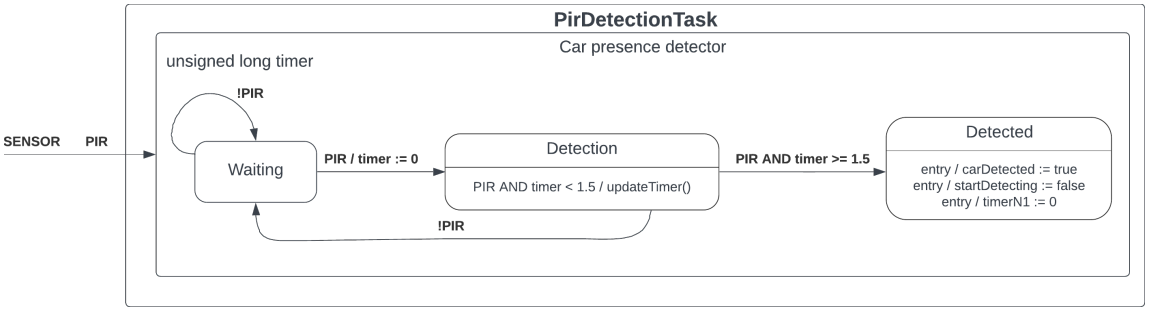

Figura 2.2: PirDetectionTask FSM

PirDetectionTask si compone di due stati: *detecting* e *end*. Il primo si occupa di azionare l'ascolto da parte del sensore *PIR*, cioè di attesa di movimento nel suo raggio di azione; questo simula l'ingresso dell'auto nella zona di check-in. Nel caso in cui non venga rilevato nulla il sistema viene posto in *deep sleep* fino a quando non viene rilevata un'auto; quindi modifica il proprio stato. Lo stato *end* ha il compito di modificare opportunamente le variabili globali e resettare il proprio stato, infine si disattiva e restituisce il controllo allo scheduler.

2.3 WelcomeTask

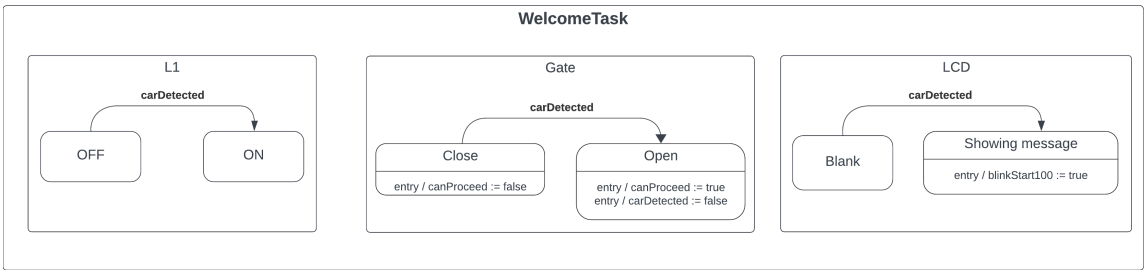

Figura 2.3: WelcomeTask FSM

WelcomeTask è composto da tre stati: *start*, *waiting* e *end*. Il primo si occupa di accendere il led *(L1)*, mostrare un messaggio sullo schermo *lcd* ed attivare la variabile relativa al blinking. Il secondo verifica che sia trascorso un predeterminato intervallo di tempo. Infine l'ultimo stato apre il cancello e modifica le opportune variabili globali e si disattiva.

2.4 ProceedTask

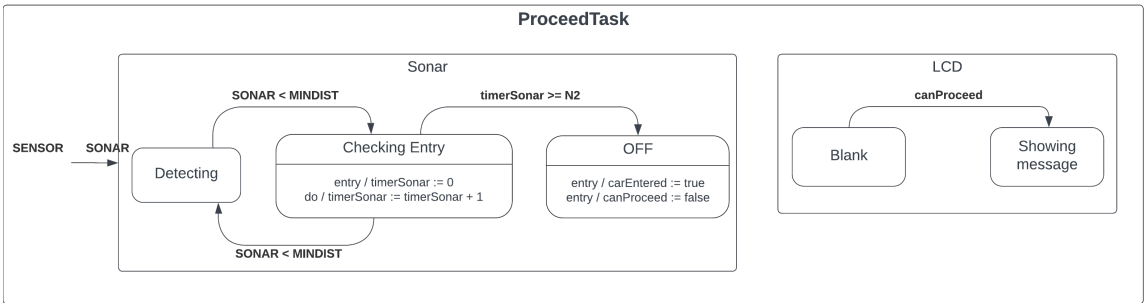

Figura 2.4: ProceedTask FSM

ProceedTask è formato da due stati interni: *monitoring* e *end*. Il primo svolge il ruolo principale del task, ovvero stampare nello schermo *lcd* un

messaggio esplicativo e soprattutto controllare tramite il sensore *sonar* la distanza dell'auto; in particolare se è minore di una soglia limite si verifica se questo è avvenuto continuativamente per il tempo richiesto, grazie all'ausilio di un timer.

2.5 PreWashingTask

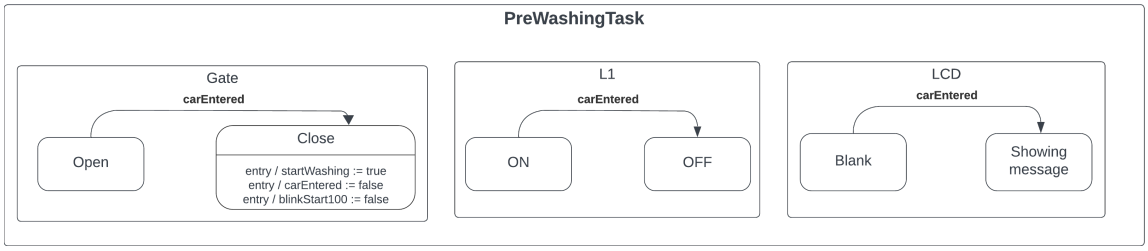

Figura 2.5: PreWashingTask FSM

PreWashingTask innanzitutto deve chiudere il cancello, successivamente si occupa di spegnere il led verde *(L1)* ed accendere quello rosso, infine mostra il messaggio *"Ready to wash"*. Il task termina come i precedenti e non viene suddiviso in diversi stati in quanto si tratta semplicemente di un task di transizione.

2.6 WashingTask

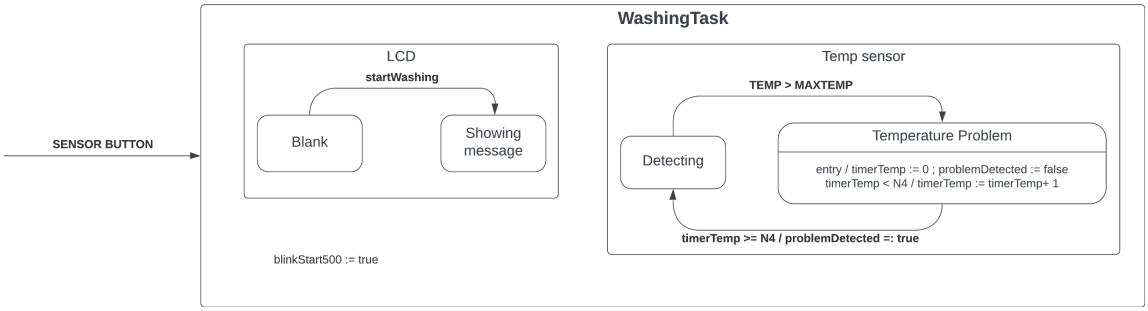

Figura 2.6: WashingTask FSM

Washing task è costituito da diversi stati: *waiting_button*, *washing*, *error* e *end*. All'interno del primo si rimane semplicemente in attesa della pressione del pulsante per poter passare allo stato successivo. Quando il pulsante viene premuto si attiva la variabile globale relativa al *BlinkTask*. Lo stato *washing* ha il compito di gestire la fase di lavaggio dell'auto, la quale prevede la comunicazione all'utente del progresso (tramite schermo *lcd*) e il controllo della temperatura tramite un apposito sensore. Nel caso in cui la temperatura abbia superato la soglia limite continuativamente per un determinato intervallo di tempo si passa allo stato *error*. Nel suddetto stato si mostra nella dashboard grafica e nello schermo *lcd* un messaggio di errore e si rimane in attesa di riparazione; ovvero l'operatore deve comunicare la risoluzione dell'errore tramite la dashboard. Una volta risolto il problema il lavaggio ricomincia, quindi si passa nuovamente allo stato *washing*. Lo stato *end* entra in azione quando il lavaggio termina mostrando nello schermo *lcd* un messaggio e reimpostando le variabili necessarie, oltre a disattivare il task.

2.7 ExitTask

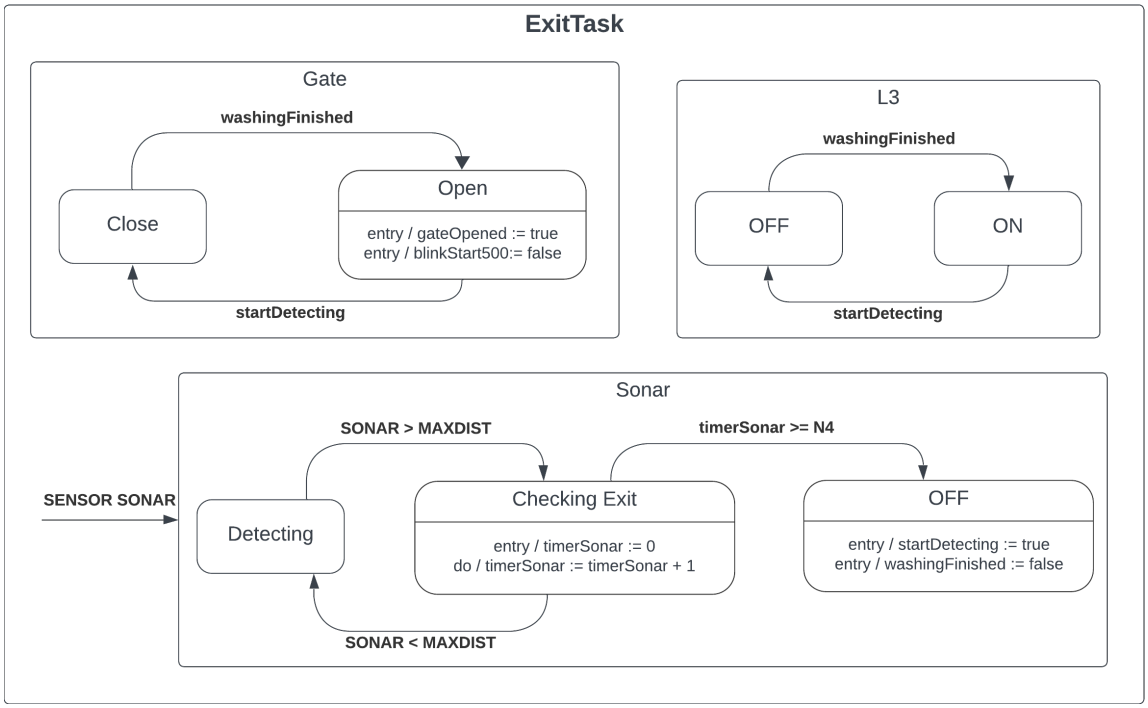

Figura 2.7: ExitTask FSM

ExitTask è formato da tre stati interni: *open*, *detecting* e *close*. Il primo deve spegnere il led rosso e accendere quello verde *(L3)*, inoltre si occupa di aprire il cancello. Il secondo stato deve gestire l'uscita dell'auto, quindi tramite il sensore *sonar* controlla che la distanza sia maggiore della soglia; nel caso in cui questo avvenga continuativamente per l'intervallo di tempo prefissato allora si passa allo stato successivo. Lo stato *close* ha il compito di chiudere il cancello, spegnere il led verde *(L3)* e aggiornare il numero di lavaggi completati che verrà comunicato all'utente tramite la dashboard. Infine modifica opportunamente le variabili, comprese quelle globali, lo stato del task e si disattiva.

Capitolo 3

Schema elettrico

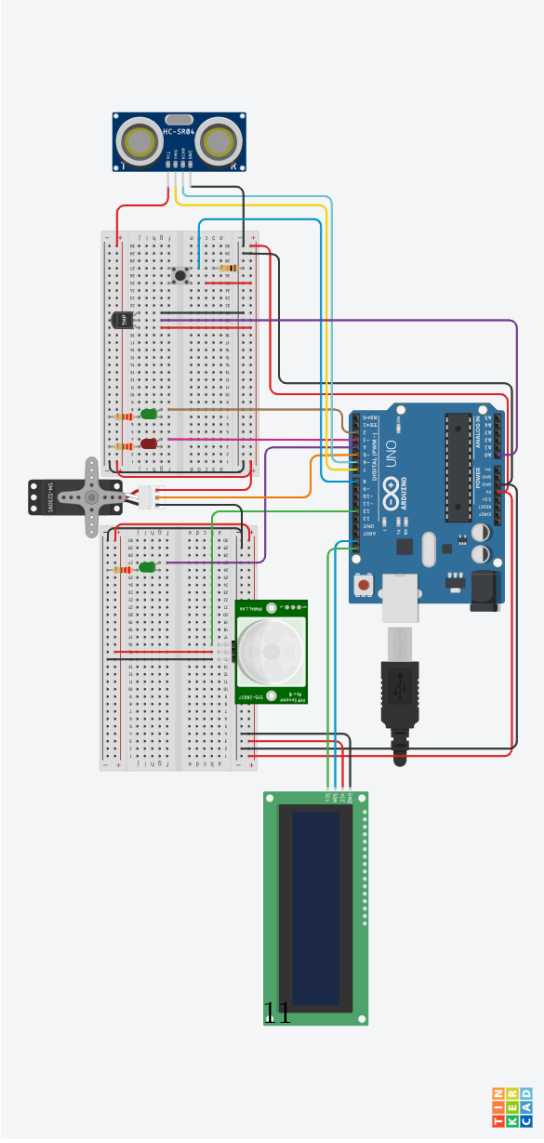

Figura 3.1: Schema elettrico

Capitolo 4

Comunicazione dashboard

Per quanto rigurada la comunicazione tra la dashboard e Arduino abbiamo scelto di utilizzare la libreria *JSSC* e un *EventListener* associato alla SerialPort lato Java, mentre lato Arduino abbiamo usato la libreria standard Serial. Il programma invia dati, sotto forma di stringhe, secondo il seguente schema: se è necessario comunicare il numero totale di lavaggi allora viene inviato solamente un numero, se vi è un problema legato alla temperatura allora viene inviata la stringa *"warning"*, se vuole comunicare la temperatura allora invia una stringa contenente la parola *"temp"* seguita dal valore della temperatura e la stessa cosa vale per comunicare lo stato della sleep, infine quando deve indicare lo stato in cui si trova il sistema, quindi il task in esecuzione, invia una stringa che lo identifica. A questo punto la decodifica lato Java è piuttosto semplice: se si riceve solo un numero allora questo indica sicuramente il numero di lavaggi eseguiti fino a quel momento, in caso contrario invece è sufficiente controllare se la stringa ricevuta contiene una delle parole che identifica un messaggio specifico ovvero: warning, sleep o temp, nel caso in cui nessuna di queste condizioni sia vere allora la stringa ricevuta indica certamente il task in esecuzione. L'unica stringa che Arduino riceve è *"fix"* per indicare che il problema legato alla temperatura è stato risolto, per fare questo una volta inviata la stringa *"warning"* il programma resta in attesa di ricevere qualcosa sulla seriale, una volta ricevuto controlla che contenga effettivamente *"fix"* e in caso positivo riprende l'esecuzione, in caso negativo rimane ancora in attesa.