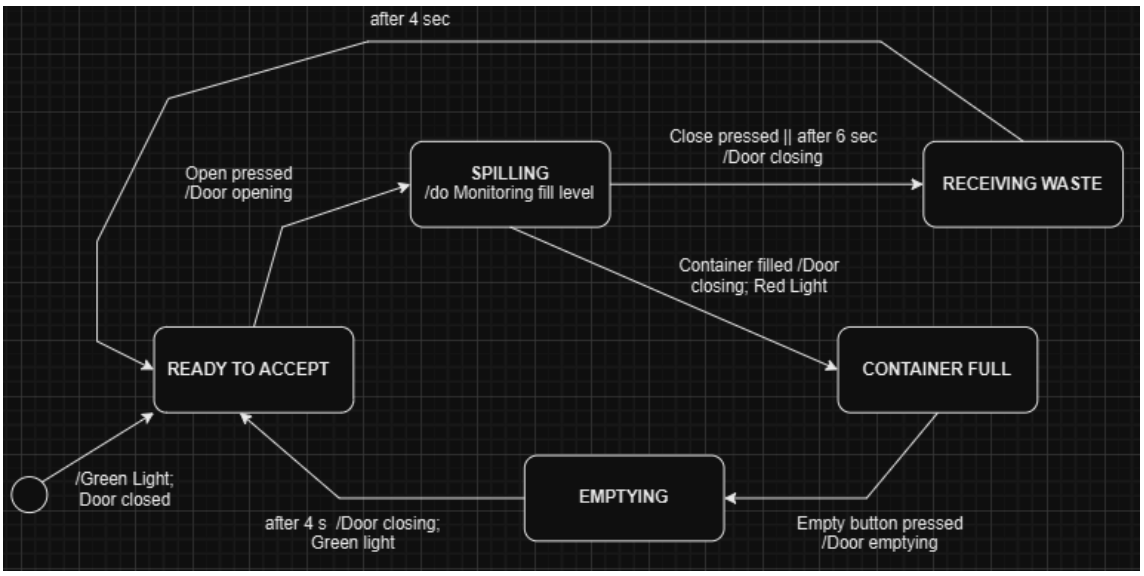
Relazione Assignment #2 - Sistemi Embedded e IOT 24-25

Michele Farneti
Samuele Casadei

Le valutazioni effettuate sul dominio del sistema, ci hanno portato ad optare per un'implementazione del sistema che prevedesse la presenza di uno scheduler sincrono, il cui periodo nello specifico viene seguito sfruttando il Timer1 dell'Arduino. Ciclicamente, ogni 50 ms, lo scheduler verifica quali task siano in attesa di eseguire il proprio tick e di conseguenze ne comanda l'esecuzione del codice. La durata dell'intervallo tra i tick dello scheduler, è stata scelto seguendo il metodo round-robin, ed è dunque il minimo comun divisore tra il periodo delle diverse task. Nello specifico, abbiamo deciso di scomporre le funzionalità messe a disposizione dal sistema del container in 4 task differenti, la cui esecuzione viene effettuata parallelamente e i cui tick sono intervallati dallo scheduler. Le task, sono in grado di interagire tra l'oro, eventualmente alterando i propri comportamenti, per mezzo della modifica e/o valutazione di variabili condivise. Tali variabili sono contenute in un oggetto container, il quale è condiviso tra le task e al quale ciascuna può accedere mediante una propria personale interfaccia. L'oggetto container ha anche il compito di incapsulare la gestione di ogni device presente nel sistema, sollevando le task dalla responsabilità di interagire direttamente con tali componenti.
Di seguito viene descritto il comportamento delle task che compongono il sistema.

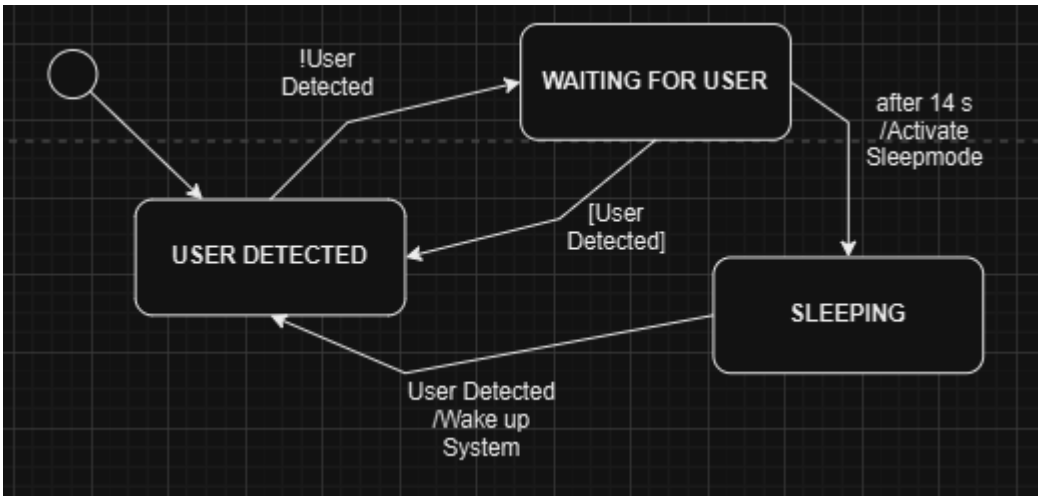
Waste Disposal Task

Questa task gestisce le interazioni basilari dell'utente con il container, partendo da uno stato di **READY_TO_ACCEPT,** in cui il sistema attende l'avvenuta pressione del pulsante open da parte dell'utente. Una volta premuto tale tasto, il container entra nella modalità di accettazione rifiuti (**SPILLING**), in questo stato, il livello di riempimento del container viene continuamente monitorato per verificarne l'eventuale superamento di una soglia massima prefissata. Qualora ciò non accada, allo scadere di un timeout o alla pressione del tasto close, dopo il passaggio per uno stato intermedio di accettazione rifiuti (**WASTE RECEIVING**), il container tornerà allo stato iniziale. Qualora invece il container raggiungesse la soglia di riempimento, la task verrebbe bloccata sullo stato di **CONTAINER FULL,** dal quale sarà possibile uscire solo qualora al container venga notificata una richiesta di svuotamento da parte della dashboard, informazione poi letta dalla task. A seguito del via libera allo svuotamento, la task passerà per uno stato intermedio di svuotamente, in cui resterà pochi secondi prima di ritornare allo stato iniziale. L'esecuzione di ogni funzionalità di questa task, viene effettuata solamente nel caso di sistema non in stato di allarme o in sleep mode.



Sleep Mode Manager Task

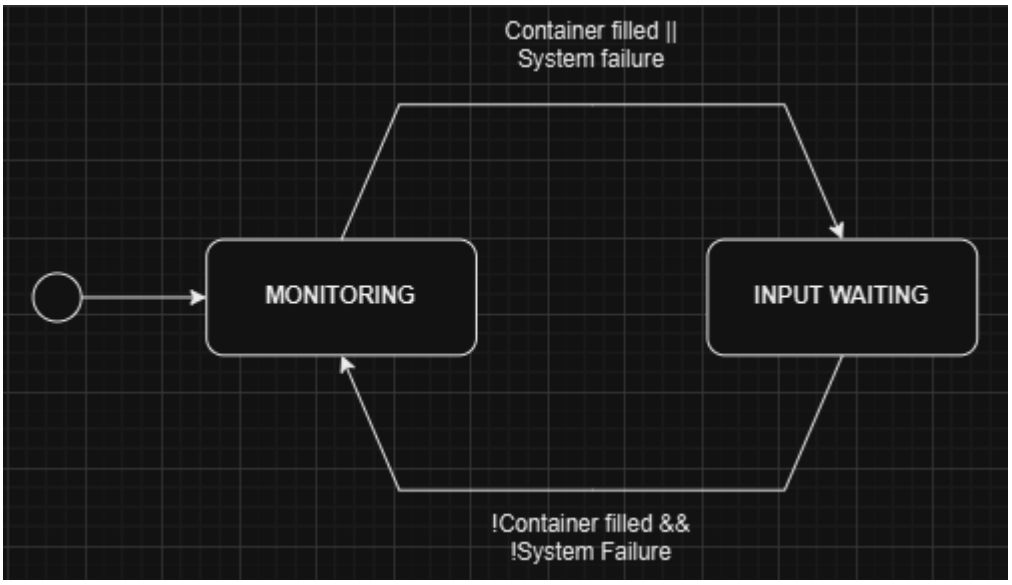
La task per la gestione della sleep mode, prevede nel suo stato iniziale, la costante rilevazione della presenza dell'utente mediante pir detector, anche in questo caso incapsulato dal container. Quando viene effettuata una rilevazione negativa, la task entra nello stato di **WAITING_FOR_USER,** al cui ingresso viene inizializzato un conteggio di tempo. Qualora l'utente dovesse esser rilevato nuovamente entro una soglia di tempo prestabilita, il sistema tornerà nello stato di **USER_DETECTED**, altrimenti entrerà nello stato **SLEEPING,** il quale avrà l'effetto di vanificare l'esecuzione del tick di tutte le altre task.



GUI Update Task

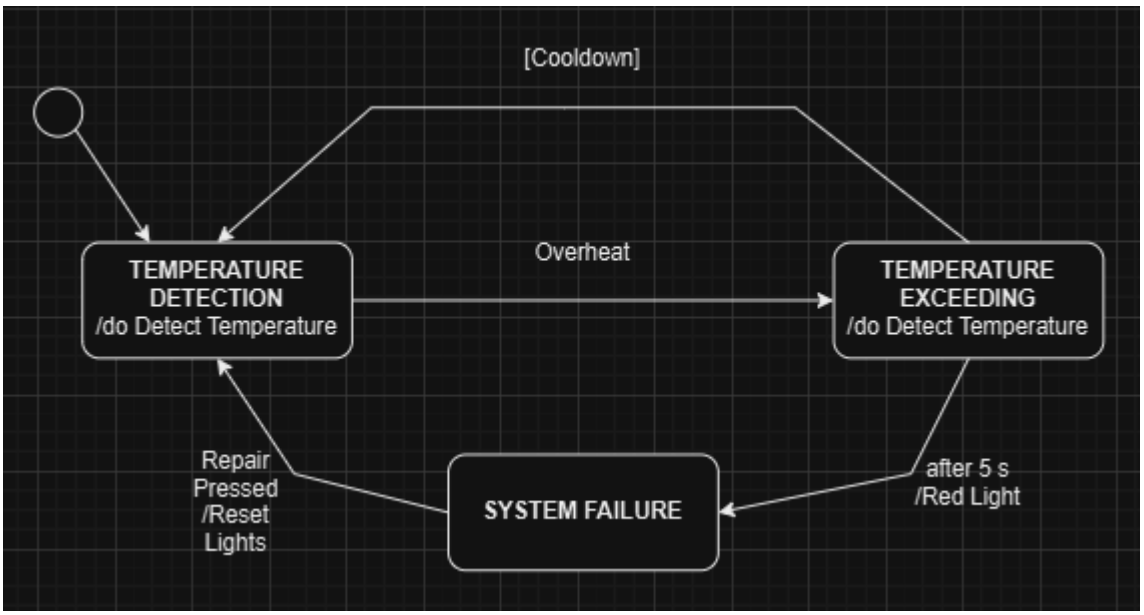
Il task per mantenere aggiornata la Dashboard si basa su due stati principali: **MONITORING** e **INPUT_WAITING**; lo stato di monitoring è lo stato in cui il task entra se il container non è né pieno, né in stato di allarme, in questo stato le informazioni sul sistema vengono inviate alla dashboard.
Lo stato di input waiting, invece, viene acceduto solo in caso in cui il container si trovi in stato di allarme, oppure sia pieno; in questo stato, oltre ad inviare le informazioni sullo stato

di sistema alla dashboard, si rimane in attesa di un input proveniente dalla dashboard sulla linea seriale .



Temperature Detection Task

Il task di rilevazione della temperatura si basa su tre stati in particolare:
TEMPERATURE_DETECTION, **TEMPERATURE_EXCEEDING** e **SYSTEM_FAILURE**;
lo stato di temperature detection è lo stato base del sistema, in cui si rileva la temperatura all'interno del container. Si entra nello stato di temperature exceeding nel momento in cui la temperatura rilevata supera una data soglia (nel nostro caso 21 °C), in questo stato oltre a tenere traccia della temperatura, si tiene traccia del tempo trascorso dal momento di ingresso nello stato.
Se, entro cinque secondi, la temperatura scende naturalmente sotto la soglia massima, si rientra nello stato di temperature detection; se, invece, la temperatura supera la soglia per cinque secondi si entra nello stato di system failure.
Si può uscire dallo stato di system failure solo con la pressione del pulsante "Repair" dalla dashboard, per poi ritornare nello stato iniziale.



Dashboard

La dashboard è stata sviluppata in java seguendo il pattern MVC, il model utilizza la libreria jssc per comunicare sulla linea seriale.
Una view rappresenta lo stato del container (livello di riempimento, temperatura e stato di allarme), fornendosi di plot su grafici (JFreeChart) per tenere traccia della history (ultimi 30 valori rilevati).
Un controller implementa la comunicazione tra view e model mantenendo alto il livello di astrazione sui dati.
Un controller implementa lo scambio di informazioni tra model e view
Le informazioni inviate lato Arduino => Dashboard sono strutturate come una tripletta di valori “**livello di riempimento ; temperatura ; flag stato di allarme**”, mentre lato Dashboard => Arduino viene inviata un singolo carattere: “**L**” per lo svuotamento del container, “**T**” per la riparazione in caso di surriscaldamento.