Bergami Lorenzo - 01 / 12 / 2024

Smart Waste Disposal System – Report

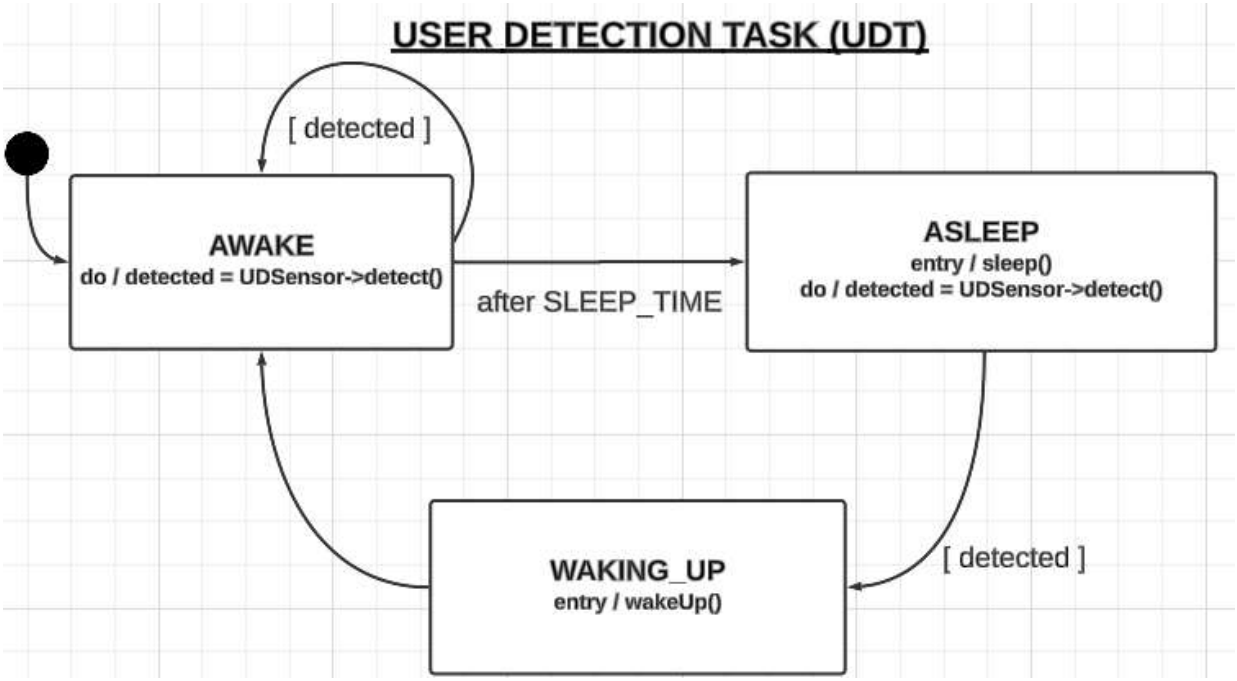
Descrizione Generale

Il sistema di gestione dei rifiuti è stato sviluppato su Arduino Uno, utilizzando un'architettura sincrona task-based per garantire modularità e scalabilità. La gestione dell'interfaccia utente è stata implementata tramite una GUI Java, progettata per fornire un monitoraggio dettagliato dello stato del sistema e un logger per le operazioni di debug.

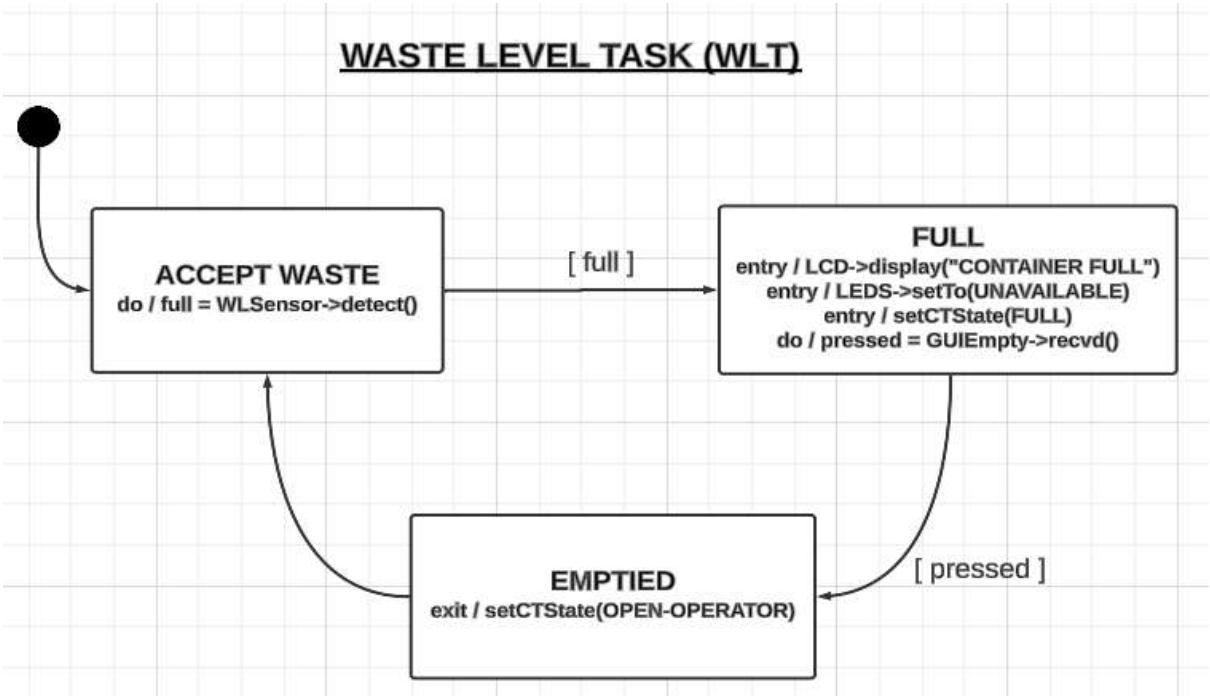
La comunicazione tra Arduino e la GUI avviene tramite una connessione seriale.

Architettura Funzionale

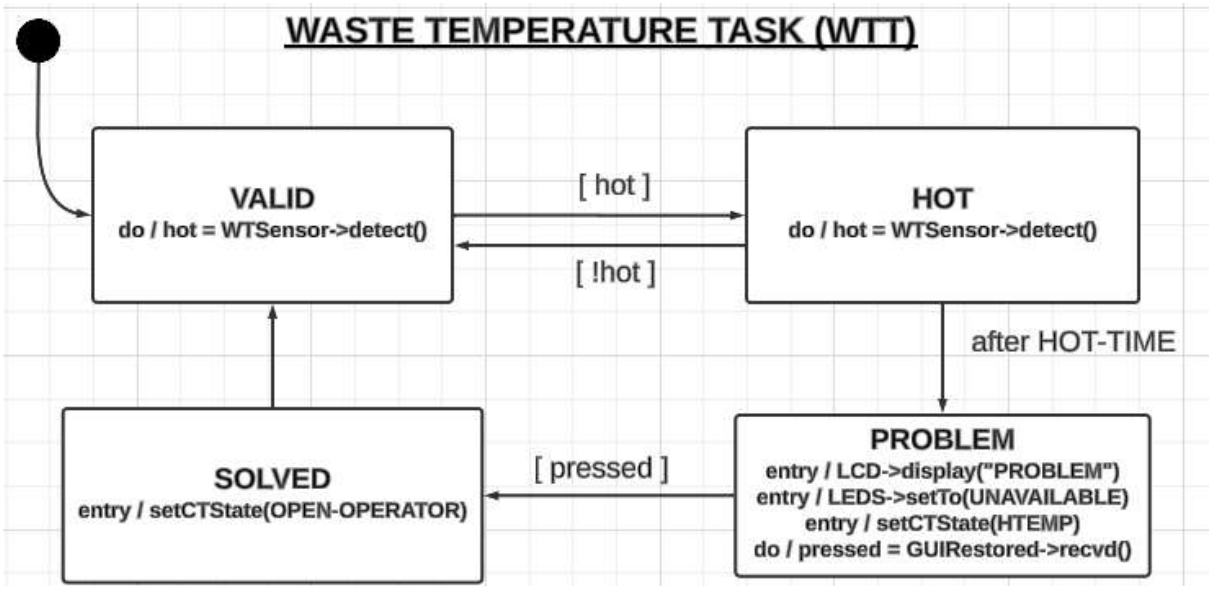
1. **FSM e Task**:
    - **User Detection Task (UDT)**:
      L'UDT rileva la presenza di un utente utilizzando il sensore PIR e distingue due modalità operative: **awake**, in cui tutti i task vengono schedulati normalmente, e **asleep**, in cui lo scheduler sospende i task non essenziali per ridurre il consumo di risorse. In modalità **asleep**, il rilevamento di un utente avvia la transizione allo stato **waking up**, durante il quale i task sospesi vengono riattivati, riportando il sistema allo stato operativo.



- **Waste Level Task (WLT)**:
  Il WLT monitora il livello dei rifiuti tramite un sensore a ultrasuoni, confrontando i dati raccolti con una soglia predefinita. Al superamento del limite, il task attiva lo stato di blocco del sistema. Il ripristino richiede l'intervento di un operatore tramite la GUI.

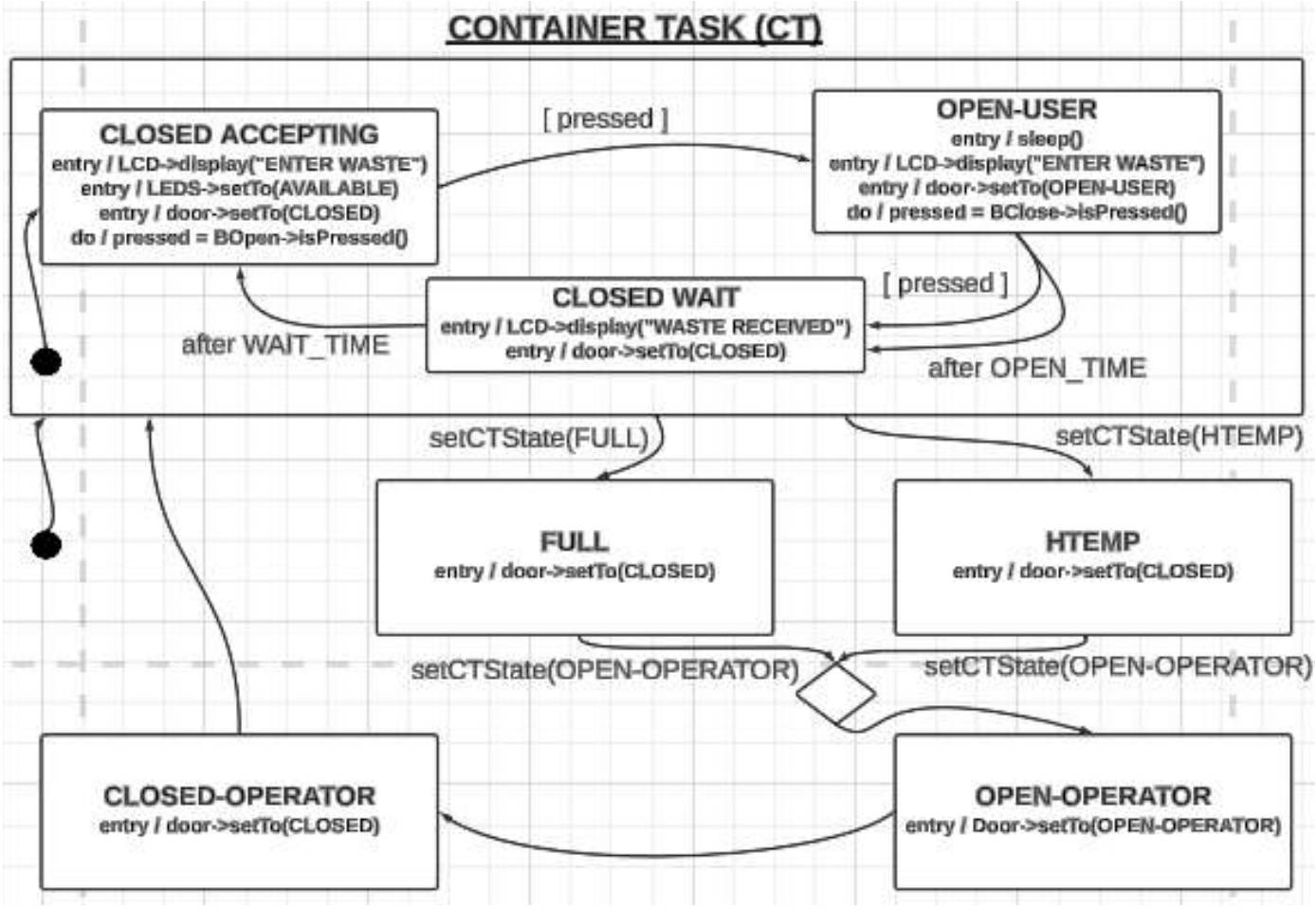


- **Waste Temperature Task (WTT)**:
  Il WTT monitora la temperatura interna del contenitore tramite il sensore TMP36, confrontando i valori con una soglia critica predefinita. Se la temperatura supera il limite per un intervallo configurabile, il task attiva lo stato di blocco del sistema. Il ripristino avviene tramite intervento dell'operatore attraverso la GUI.



- **Container Task**:
  Il Container Task gestisce l'apertura e la chiusura del contenitore tramite un servomotore, con tre posizioni operative: 0° (apertura operatore), 90° (contenitore chiuso) e 180° (apertura utente). Il suo stato dipende dallo stato del sistema, determinato dai task WLT e WTT.

  Quando il sistema entra in uno stato di errore (**"full"** segnalato dal WLT o **"problem"** segnalato dal WTT), il Container Task viene bloccato, impedendo qualsiasi interazione con il contenitore. Sebbene nello schema sia rappresentato come un evento asincrono per semplificarne la visualizzazione, a livello implementativo il Container Task, ad ogni ciclo di esecuzione, interroga il modello del sistema per verificare il proprio stato.



2. **Sistema**:
   Il Sistema è l'entità centrale che gestisce la comunicazione tra i Task e fornisce le interfacce per l'accesso ai sensori. Inoltre, il Sistema è responsabile del controllo degli attuatori e dei dispositivi di comunicazione. La logica di cambio di stato è delegata ai Task, che valutano autonomamente i criteri operativi basandosi sui dati raccolti. Il Sistema, invece, esegue le azioni associate agli stati dei Task, come l'attivazione degli attuatori o l'invio di messaggi alla GUI, garantendo una separazione chiara tra logica decisionale e gestione delle operazioni hardware.

GUI dell’Operatore

La GUI Java è composta da:

1. **Logger**:
    - Utile durante la fase di avvio e testing.
2. **Finestra di Monitor Stato**:
    - Consultata attivamente dall’operatore per visualizzare lo stato del sistema, contiene:
        - Stato del **Container Task** e degli altri task.
        - **Livello di riempimento** (%).
        - **Temperatura rilevata**.
        - Stato dello **User Detection Task**.
        - Bottoni per comunicare al sistema il ripristino del contenitore (Empty Button e Restore Button).
        - Start Button (per avviare il sistema).

Conclusioni

Il sistema combina FSM, un’architettura sincrona task-based e una separazione netta tra hardware e logica applicativa, garantendo modularità, manutenibilità e scalabilità. La GUI permette monitoraggio attivo e gestione efficace del sistema da parte dell’operatore.