Relazione assignment 2

Smart coffee machine

Corso di "Embedded systems and IoT"

Filippo Pilutti

11 maggio 2022

Lo scopo di questo assignment era quello di creare un sistema embedded che emulasse il funzionamento di una macchina del caffé smart. Nello sviluppo del progetto ho utilizzato un'architettura basata su task, modellati attraverso macchine a stati finiti sincrone, come da specifiche. Questo approccio semplifica il design e l'implementazione di questo sistema permettendone una decomposizione e modularizzazione.

Il sistema hardware é costituito da diversi componenti: un LCD collegato ad Arduino attraverso I2C, un sensore di movimento PIR, un sensore di distanza sonar, un piccolo servo motore usato per simulare la preparazione di una bevanda, un potenziometro, tre bottoni per selezionare e confermare le scelte, un sensore di temperatura TMP36 e ovviamente la scheda Arduino.

Per quanto riguarda la parte software, oltre al progetto Arduino caricato e mandato in esecuzione sulla scheda, vi é anche un semplice programma sviluppato in Java che, attraverso la linea seriale, permette di scambiare informazioni con il sistema.

Task

Analizzando il sistema da progettare, ho pensato di scomporre il comportamento generale in sei task.

- **MachineStateTask** é il task "principale" del sistema, in quanto rappresenta tutti i possibili stati in cui la macchina puó trovarsi durante la sua esecuzione e quindi influisce sul comportamento degli altri task. Gli stati possibili sono: *boot, ready, operating, sleep, assistance, testing.*

- **ProductsTask** modella il comportamento relativo alla selezione delle diverse tipologie di bevande e alla simulazione della loro preparazione. I suoi stati sono: *idle, select-coffee, select-tea, select-choc, making-coffee, making-tea, making-choc, product-ready.*

- **CheckDistanceTask** si occupa unicamente di rivelare la distanza dell'utente dalla macchina attraverso il sensore di distanza sonar, e i suoi unici due stati sono: *idle, checking.*

- **CheckPresenceTask** similmente al task precedente si occupa di rivelare periodicamente la presenza di un utente, e i suoi due task sono sempre *idle* e *checking.*

- **SelfTestTask** il compito di questo task é quello di eseguire un test di controllo della macchina ogni periodo di tempo, verificando il corretto

funzionamento del servo motore e l'adeguata temperatura. Gli stati sono: *idle, testmotor, testtemp.*

- **SerialCommunicationTask** si occupa di comunicare con un programma in esecuzione su un computer scambiando informazione attraverso la linea seriale. Gli stati sono: *show-state, assistance.*

Variabili globali

L'interazione tra i diversi task é gestita attraverso l'utilizzo di variabili condivise che indicano lo stato globale del sistema (*readyState, sleepState, operatingState, testState, assistanceState*), rappresentano determinate situazioni (*productReady, productTaken, userPresent, fixed, tempOutOfBounds*) oppure tengono traccia di informazioni che cambiano nel tempo (*nMaxCoffee, nMaxChoc, nMaxTea, sugarLevel, nSelfTests*).

Macchine a stati finiti

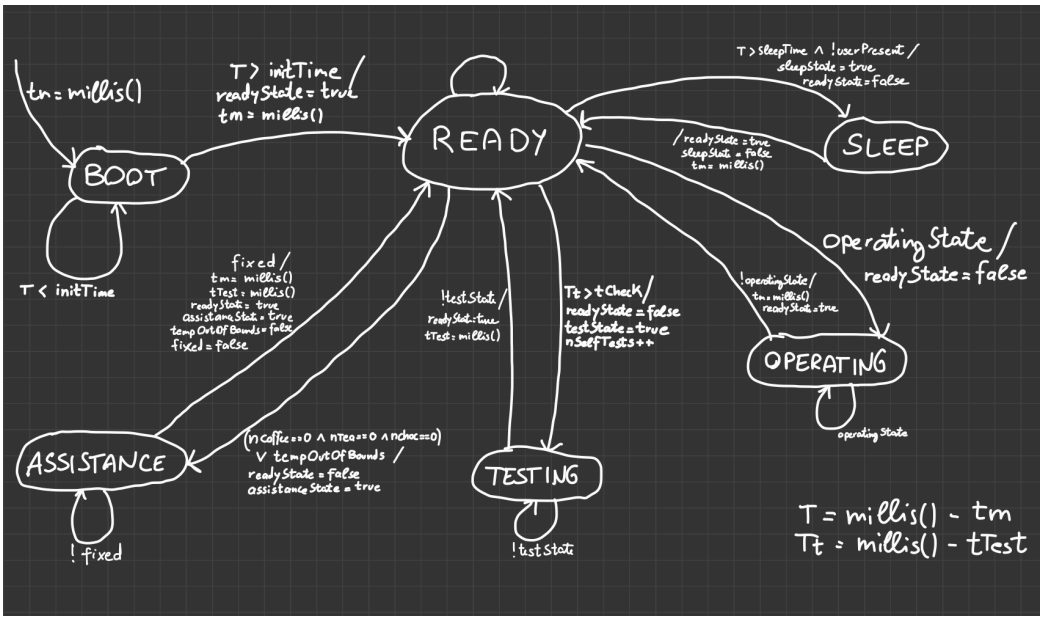

Figura 1: Macchina a stati finiti per MachineStateTask.

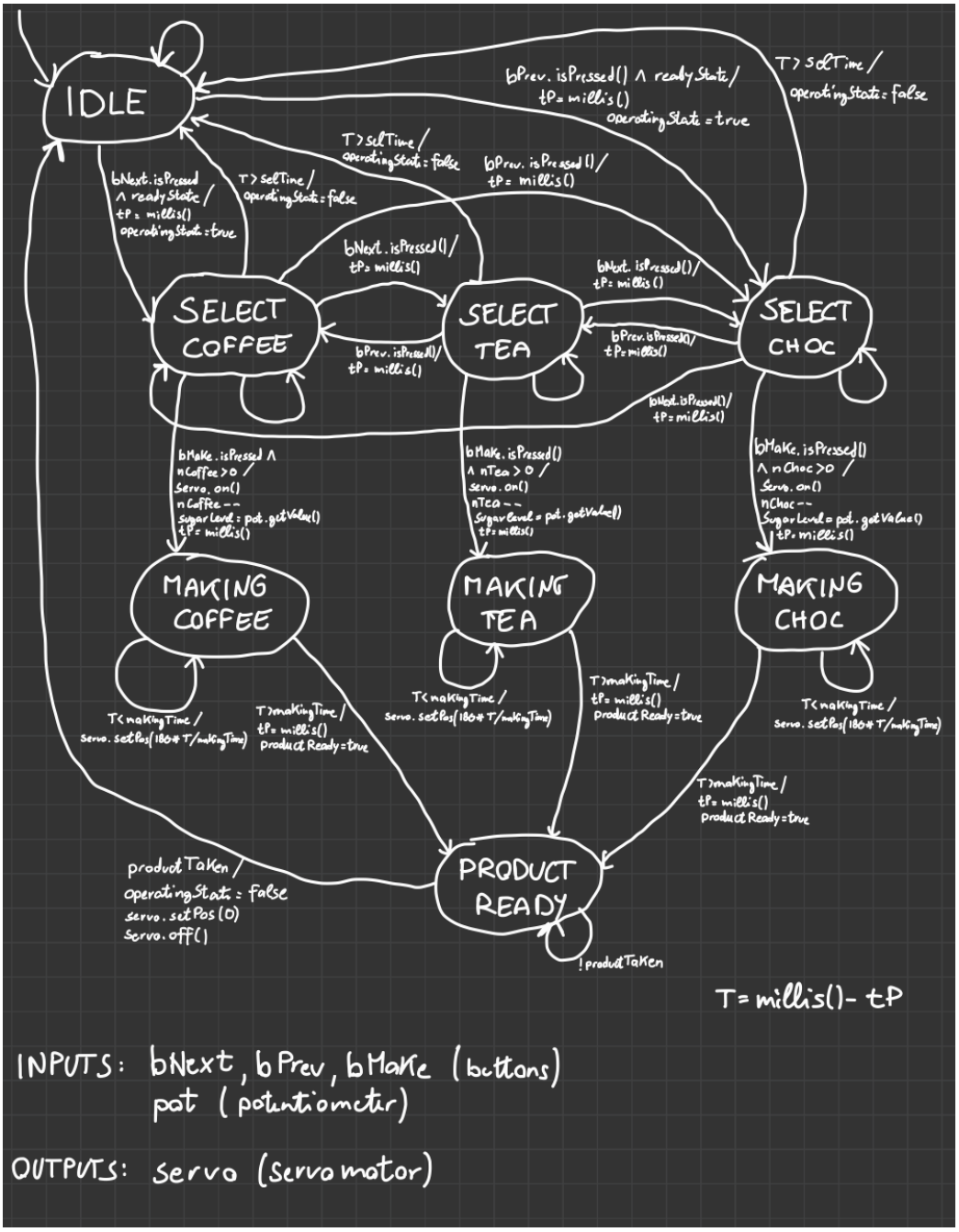

Figura 2: Macchina a stati finiti per ProductsTask.

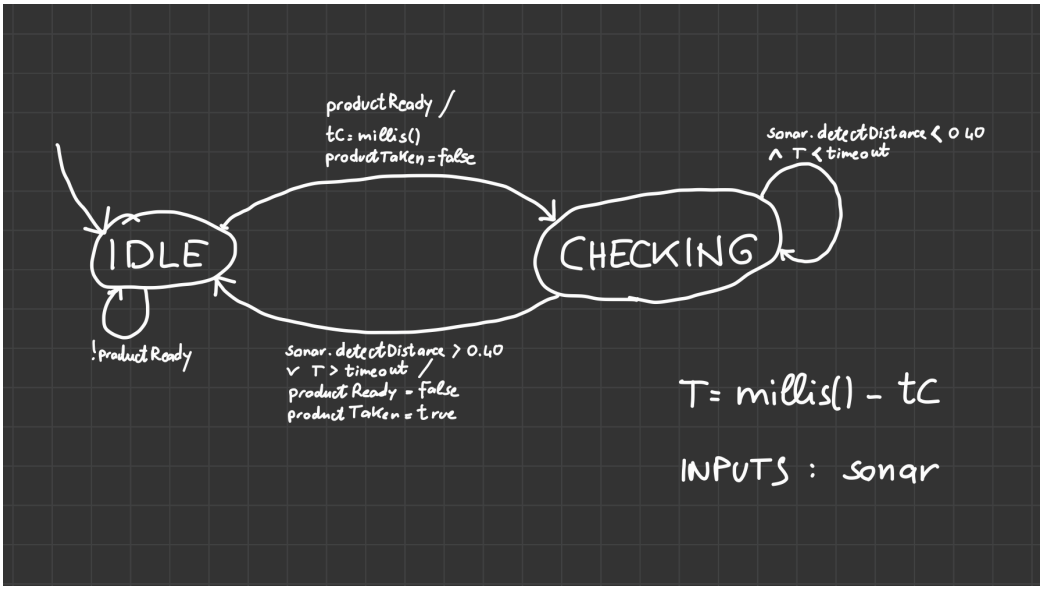

Figura 3: Macchina a stati finiti per CheckDistanceTask.

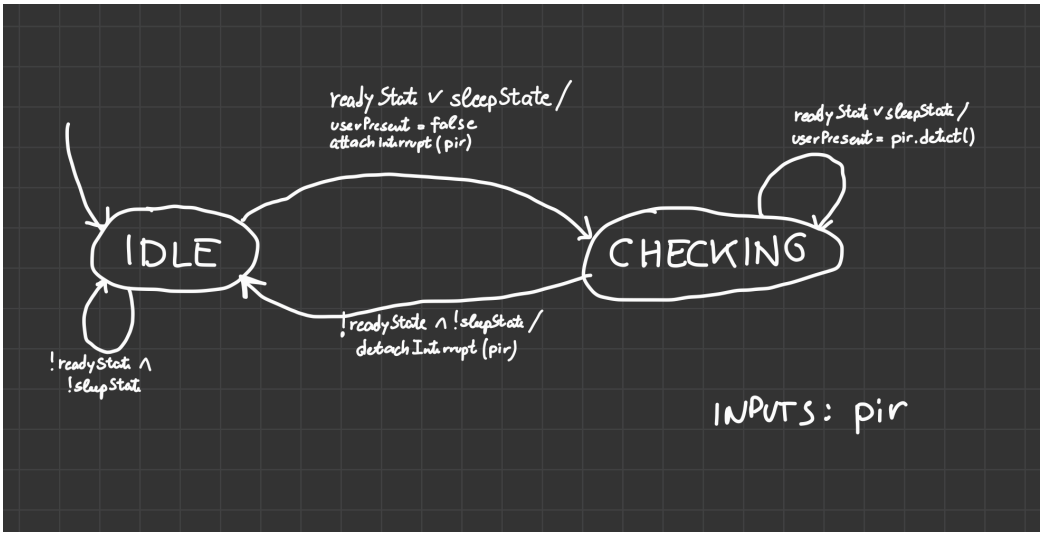

Figura 4: Macchina a stati finiti per CheckPresenceTask.

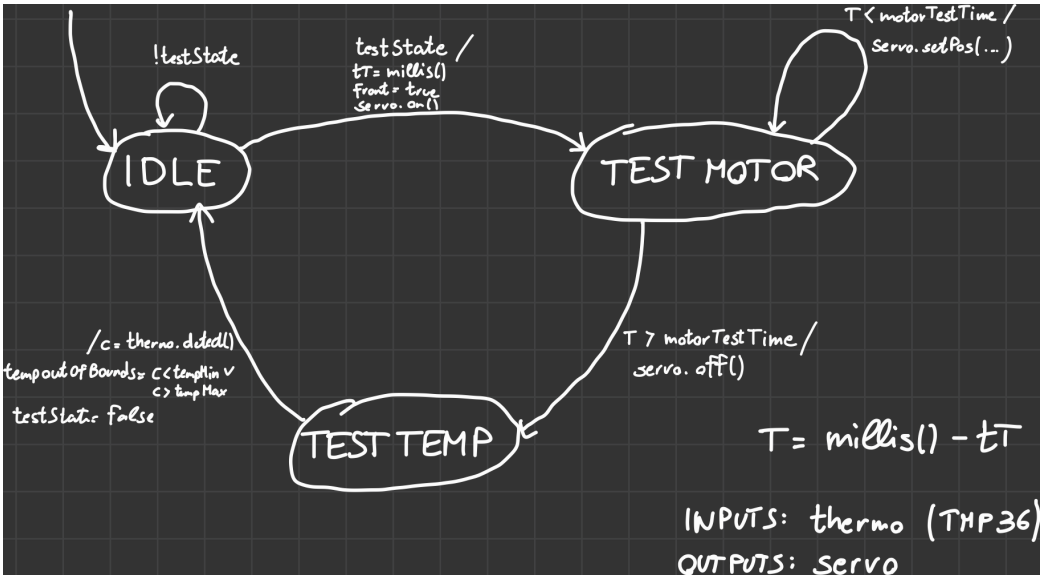

Figura 5: Macchina a stati finiti per SelfTestTask.

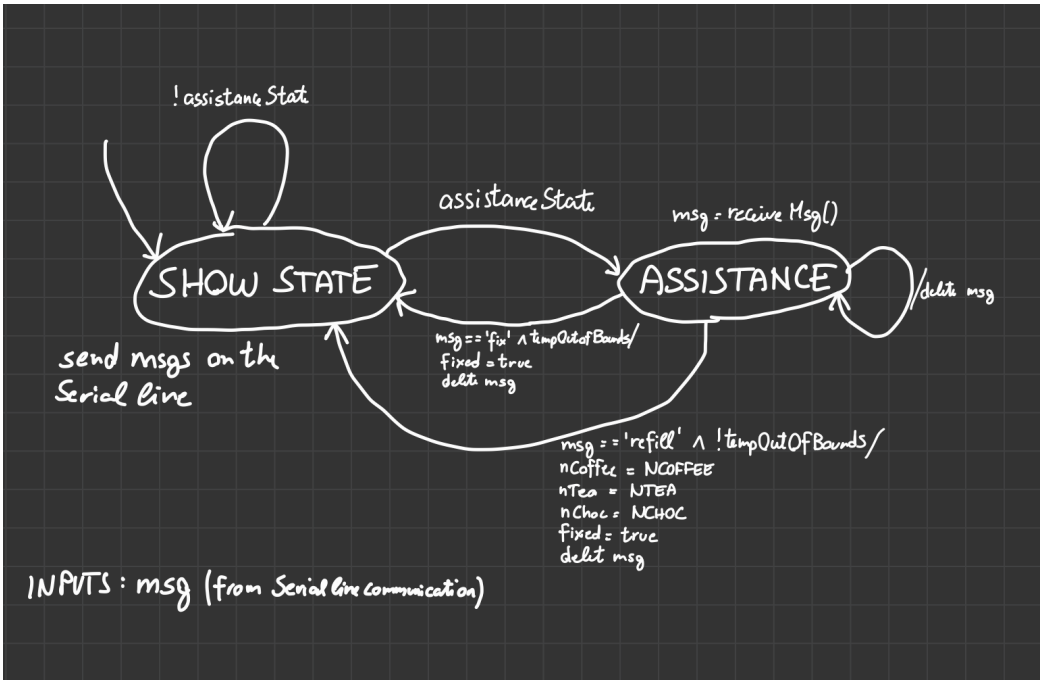

Figura 6: Macchina a stati finiti per SerialCommunicationTask.

Sistema hardware

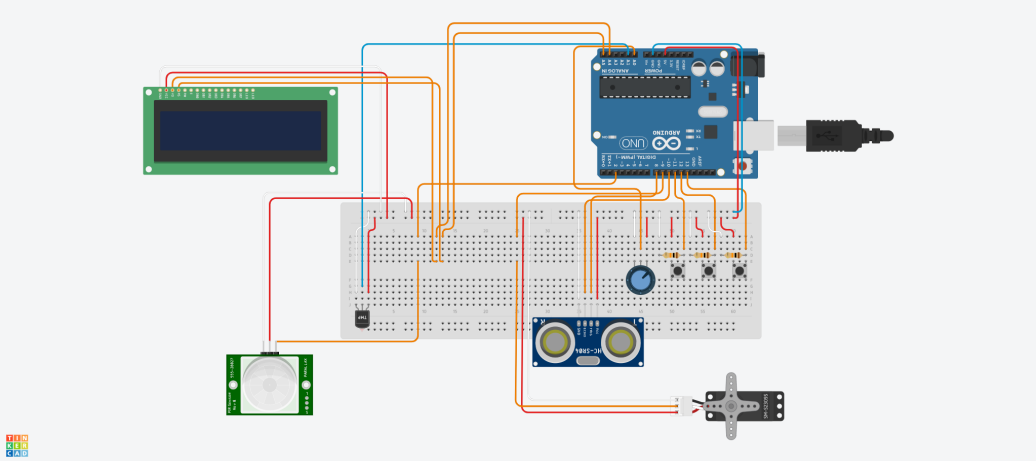

Figura 7: Schema collegamenti dei dispositivi

Applicazione java

Figura 8: Vista della gui del programma coffee machine manager