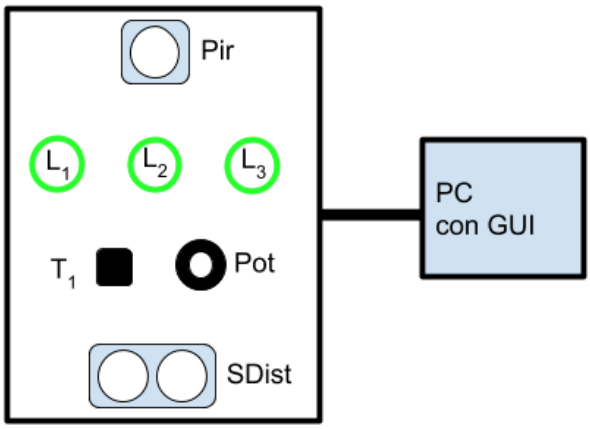
Progetto 02
Smart Coffee Machine

Paolo Baldini Andrea Giulianini

18 novembre 2018

1 Specifiche

Si vuole realizzare un sistema embedded che implementa una macchina del caffè smart.



1.1 Componenti

- 3 led (L1, L2, L3) verdi
- 1 pulsante tattile T1 e 1 potenziometro Pot
- 1 sensore di movimento PIR e un sonar SDist
- Collegamento seriale con PC

1.2 Comportamento del Sistema

La macchina parte in una modalità STAND BY, in cui si presuppone avere un consumo ridotto di energia elettrica. Quando viene rilevata la presenza di qualcuno nelle vicinanze allora esce dalla modalità di risparmio, entrando in una modalità ON.

Se viene rilevato qualcuno ad una distanza inferiore a DIST1 cm per un certo numero DT1 secondi, viene visualizzato un messaggio "Welcome!" sulla GUI lato PC e la macchina entra in una modalità READY, in cui può accettare comandi. Se la macchina non rileva più nessuno entro quella distanza per DT2a secondi, la macchina torna nella modalità ON. Dalla modalità ON la macchina torna in modalità STAND BY se non viene rilevata la presenza di nessuno per DT2b secondi.

In modalità READY, l'utente può regolare lo zucchero con la manopola POT. Ogni volta che cambia il livello di zucchero, lato PC viene visualizzato opportunamente il livello aggiornato di zucchero (mediante componente grafico o

messaggio testuale).

L'utente può fare un caffè premendo il pulsante T1. Quando preme il pulsante, parte il processo rappresentato dall'accensione progressiva dei 3 led, della durata di DT3 secondi. Lato PC, viene visualizzato il messaggio "Making a coffee" e poi "The coffee is ready" quando il caffè è pronto.

L'utente ha tempo DT4 secondi per prendere il caffè - in questo caso si simula con distanza che va sotto i DIST2 cm, ovvero: rilevo distanza sotto DIST2 cm, significa che l'utente ha preso il caffè. Se non si prende il caffè entro DT4 secondi, in ogni caso la macchina torna nella modalità READY (e viene rimosso il messaggio lato PC).

Infine, la coffee machine può esaurire il materiale per il caffè. Si suppone che possa essere caricata per fare NMAX caffè. Dopo aver fatto NMAX caffè, lato viene prodotto il messaggio "No more coffee. Waiting for recharge" ed entra in modalità MAINTENANCE. In questa modalità, il sistema aspetta il ricaricamento che può avvenire mediante un'azione lato PC (es: pulsante di ricarica), corrispondente ad una certa quantità di dosi NC. Avvenuta la ricarica, lato PC viene visualizzato il messaggio "Coffee refilled: "+NC e la macchina torna in modalità STAND BY.

Realizzare il sistema su Arduino + PC collegati via seriale, implementando il programma su Arduino in C++ e il programma su PC in Java. Utilizzare un approccio a task, con modello di comportamento basato su macchine a stati finiti sincrone.

1.3 Valori di Funzionamento

- DIST1 (Engagement distance) = 0.3 m
- DIST2 (Take coffee) = 0.1 m
- DT1 (Min engagement time) = 1 s
- DT2a (Max time with no engagement) = 5 s
- DT2b (Max time with no presence) = 5 s
- DT3 (Coffee making process duration) = 3 s
- DT4 (Max time to remove coffee) = 5 s

Per tutti gli aspetti non specificati, fare le scelte che si credono più opportune.

2 Soluzione

2.1 Introduzione all'Approccio

La soluzione del progetto viene fornita, come richiesto, con l'approccio a task. Ogni task svolge un compito, per quanto semplice esso sia. Anche se alcuni elementi non avrebbero veramente avuto bisogno di un task singolo, ma sarebbero potuti essere inseriti direttamente nelle classi che li utilizzavano (vedi pir), è stato comunque scelto di utilizzare una filosofia di "scorporamento" abbastanza aggressiva per massimizzare la modularità del sistema. Ne risulta un sistema con più entità in gioco di quelle necessarie, ma il funzionamento non risulta affatto più complesso di un sistema con minor modulazione.
I moduli, i task e le entità del progetto dialogano attraverso variabili comuni. Non è stato volontariamente utilizzato il termine "globali" perchè effettivamente globali queste non sono. Queste variabili comuni sono infatti incluse dentro una struttura dati, e non istanziate singolarmente tramite la keyword "extern". Non c'è un vero motivo per il quale abbiamo preferito questo approccio ne abbiamo trovato un motivo per il quale questo possa essere svantaggioso, è stata quindi una pura scelta implementativa.

2.2 Analisi dei Componenti

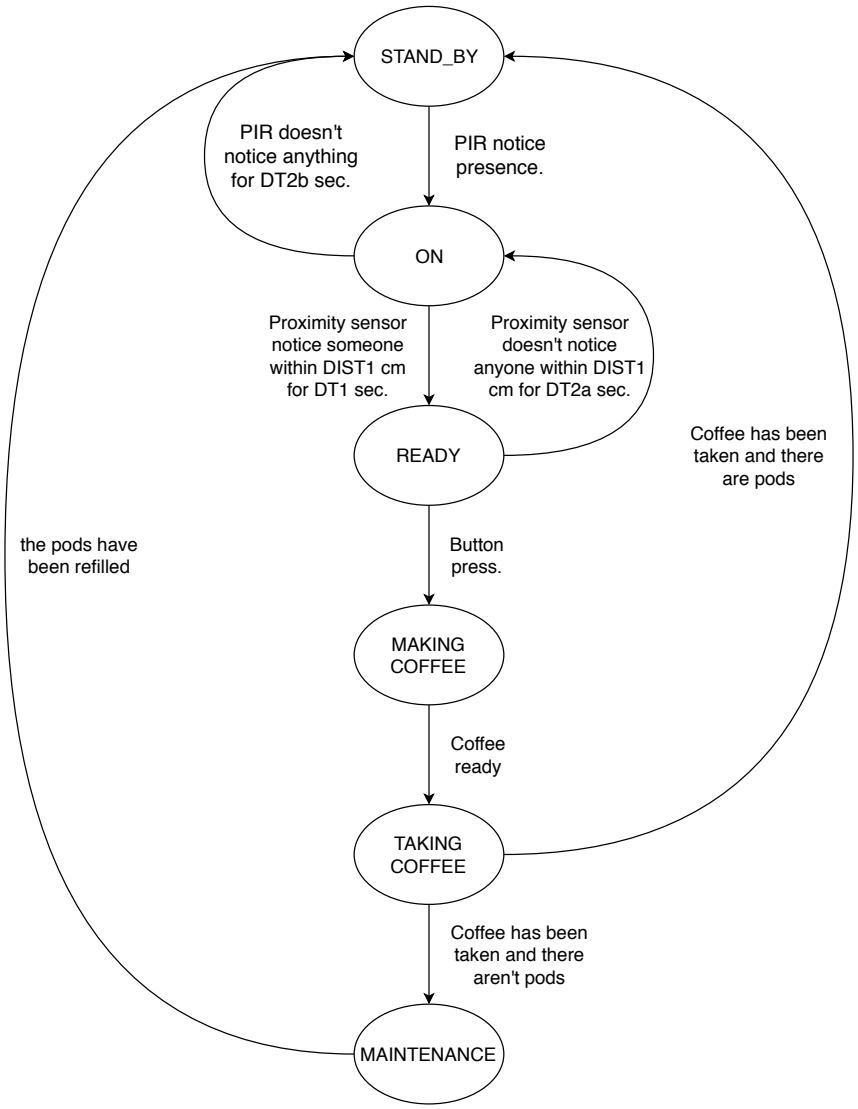
In questa sezione verrà analizzato l'utilizzo dei singoli task e la logica che vi è dietro. Le FSM presentate sono relative alle componenti più importanti del programma. Sono quindi escluse dalla trattazione FSM estremamente semplici e con transizioni banali (come può ad esempio risultare quella del sensore di presenza).

2.2.1 Funzionamento generale

Questa FSM rappresenta il funzionamento generale della macchina, evidenziando i vari stati e le condizioni di transizione a un livello astratto e senza considerare quindi le variabili in gioco nel programma. Come si può evincere dallo schema, agli stati definiti nella consegna sono stati aggiunti due ulteriori stati:

- MAKING COFFEE
- TAKING COFFEE

Ci siamo presi la libertà di fare ciò perchè abbiamo considerato l'inserimento di questi due stati più esplicativo del funzionamento della macchina e perchè non abbiamo ritenuto che questo rendesse più facile o snaturasse l'intento del compito, ovvero l'apprendimento di utilizzo di un modello a task in un sistema integrato.
Fatta questa premessa, il funzionamento si basa sul passaggio fra vari stati, ognuno dei quali svolge compiti ben definiti in funzione dello stato attuale della macchina.



2.2.2 Scheduler

Questo non è un task, ma è l'entità che permette all'approccio a task di avere applicazione. Ogni tot millisecondi esce da un costante stato di sleep per lanciare in esecuzione i vari task. L'uscita dallo stato di sleep è determinata dall'interrupt generato dal watchdog timer (settato come interrupt e non come reset).

2.2.3 ITask

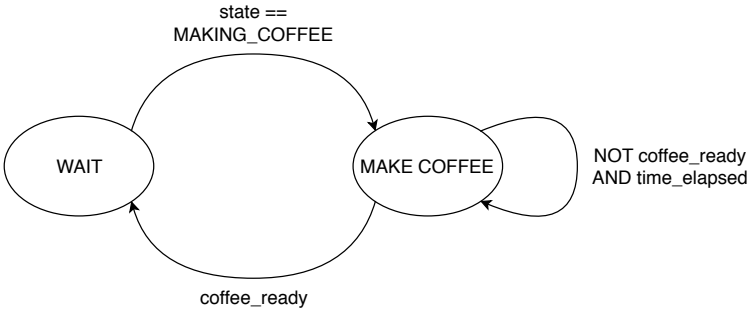
Non si tratta di un vero e proprio task ma di un interfaccia alla quale questi devono aderire per essere considerati tali. Definisce un metodo virtuale puro

Exec() che verrà richiamato dallo scheduler ogni volta che il task dovrà essere eseguito.

2.2.4 Distance

Il compito di questo task consiste nel valutare se la distanza acquisita dal sensore di prossimità rientra nel range valutato in uno specifico stato. Se ad esempio ci trovassimo nello stato ON, allora sapremmo che la distanza richiesta per passare allo stato READY deve essere inferiore a DIST1. Con questi dati possiamo informare gli altri task (tramite variabili condivise) se un individuo si trova nel "range" corretto per il passaggio al successivo stato.

2.2.5 MakeCoffee



Se la macchina si trova nello stato MAKING_COFFEE, il task si occuperà di "preparare il caffe" e di visualizzare l'avanzamento della preparazione tramite i led. La presenza di una freccia rientrante nello stato di "MAKE COFFEE" nella FSM indica che a ogni scadere del timer, se il caffè non risulta pronto, si accenderà un nuovo led.
Sarebbe anche stato possibile scomporre la gestione dei led in un ulteriore task (per permetterne la gestione in ulteriori possibili circostanze non specificate nel testo), ma per mancanza di tempo e per la relativa utilità di questo nel contesto attuale si è deciso di integrare le due parti.

2.2.6 Potentiometer

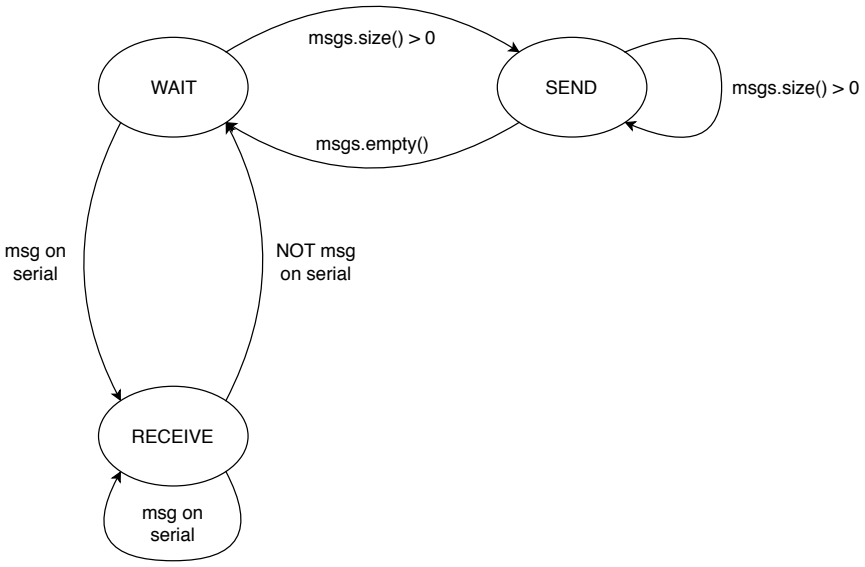
Questo task acquisisce il valore del potenziometro e lo mappa nelle varie possibili quantità di zucchero selezionabili. Se il valore acquisito differisce da quello precedente, allora richiede l'invio di un messaggio attraverso la seriale che notifichi il cambiamento.

2.2.7 Presence

E' il task che si occupa di catturare la presenza o assenza di un individuo tramite il sensore a infrarossi. Non sarebbe stato necessario creare un task solo

per questo sensore, ma data la sua importanza a livello logico e di gestione nel programma, abbiamo ritenuto che inserirlo come task avrebbe suscitato in un lettore (del codice) un idea lampante di quanto questo elemento sia fondamentale nel funzionamento della macchina.

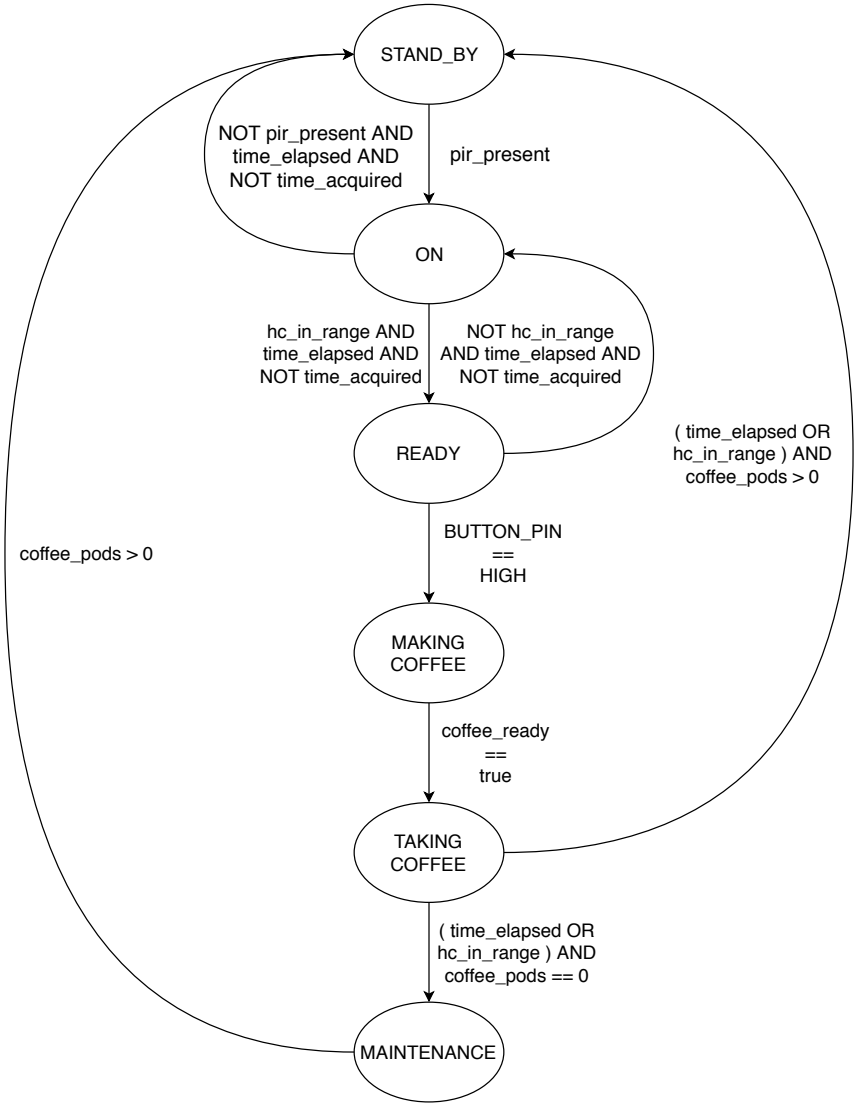
2.2.8 Postman



Quest'entità si occupa di inviare i messaggi accodati da altri task e di riceverne tramite la seriale. La logica è semplice: finchè sono presenti messaggi da inviare li trasmette, mentre finchè sono presenti messaggi in arrivo li riceve dalla seriale.

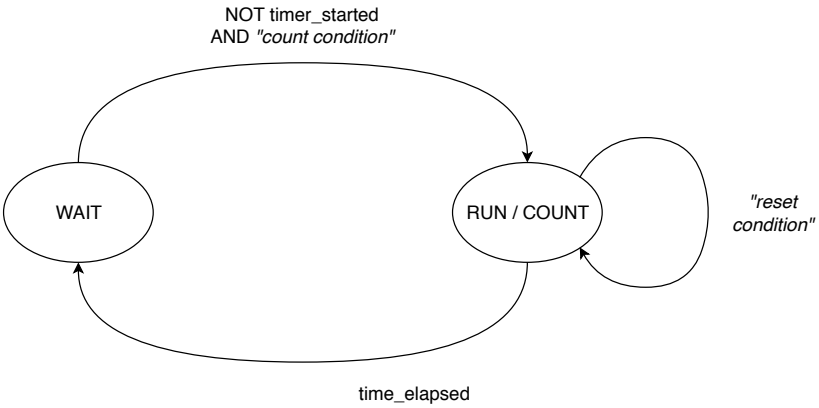
2.2.9 StateSwitcher

E' il task che si occupa del passaggio di stato della macchina. La transizione è definita dalle variabili settate dagli altri task (e.g.: pir_present).
E' di seguito inserito lo schema della FSM relativa. E' molto simile a quello del funzionamento generale della macchina ma, a differenza di quello, mostra le condizioni di transizione da uno stato all'altro tramite le variabili presenti nel programma.



2.2.10 Timer

L'idea dietro al task timer è che a seconda dell'operazione in atto nella machina, questo dovrà dire agli altri task se il tempo atteso (che varia da operazione a operazione) è passato o meno. Allo scadere, il timer notificherà l'evento agli altri task tramite il solo utilizzo di variabili globali.
La FSM del task Timer è molto semplice ma le condizioni di transizione tra i due stati sono varie. Per non appesantire troppo lo schema è stato scelto di non mostrare tutte le condizioni direttamente su questo, ma di inserire al loro posto un "riferimento" ad esse ed esporle quindi sotto.



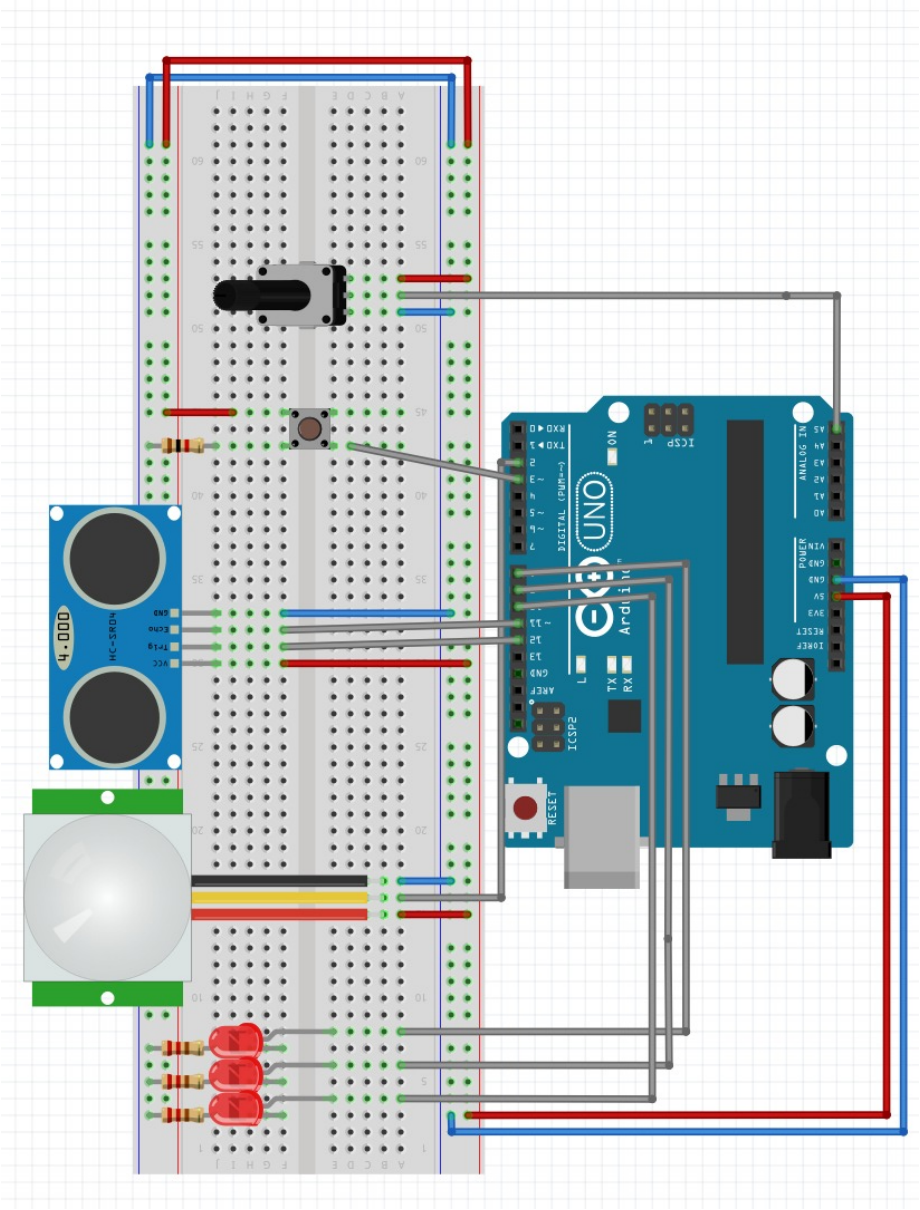
Count condition:

(state == ON AND NOT pir_present)
OR (state == ON AND hc_in_range)
OR (state == READY AND NOT hc_in_range)
OR (state == MAKING_COFFEE)
OR (state == MAKING_COFFEE AND coffee_ready)

Reset condition:

time_acquired
OR (state == ON AND next_state == STAND_BY AND pir_present AND NOT time_acquired)
OR (state == ON AND next_state == READY AND NOT hc_in_range AND NOT time_acquired)
OR (state == READY AND next_state == ON AND hc_in_range AND NOT time_acquired)

3 Fritzing



4 Note

Durante lo sviluppo del progetto si è notato che la "libreria" di comunicazione sulla seriale fornita dal docente non si integrava bene nel progetto, non funzionando. Analizzando il problema ci siamo accorti che seppure la seriale rilevasse un passaggio di informazioni (notato tramite il led montato sulla scheda), alla chiamata della funzione isMsgAvailable() di MsgService, non si riscontrava esisto positivo. Si è cercato di risolvere il problema, ma a causa dello svolgimento di tirocinio e lavoro dei membri del gruppo, non è stato possibile trovare sufficiente tempo da dedicare alla risoluzione, rimanendo quindi in uno stato di non-soluzione. A causa di ciò non è stato quindi possibile testare l'operazione di "refill", che però confidiamo (data l'estrema semplicità della stessa) non avrebbe avuto nessun problema a funzionare.