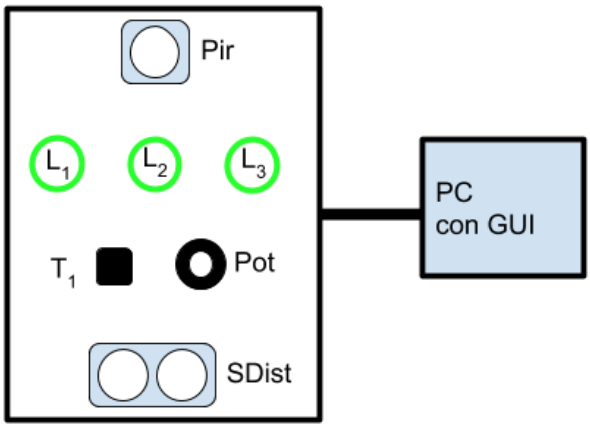
Progetto 02
Smart Coffee Machine

Paolo Baldini Andrea Giulianini

November 18, 2018

1 Specifics

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



1.1 Components

- 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 System Behaviour

La macchina parte in una modalit 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 modalit di risparmio, entrando in una

modalit 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 modalit READY, in cui pu accettare comandi. Se la macchina non rileva pi nessuno entro quella distanza per DT2a secondi, la macchina torna nella modalit ON. Dalla modalit ON la macchina torna in modalit STAND BY se non viene rilevata la presenza di nessuno per DT2b secondi.

In modalit READY, lutente pu 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).

Lutente pu fare un caff premendo il pulsante T1. Quando preme il pulsante, parte il processo rappresentato dallaccensione 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 caff pronto.

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

Infine, la coffee machine pu esaurire il materiale per il caff. Si suppone che possa essere caricata per fare NMAX caff. Dopo aver fatto NMAX caff, lato viene prodotto il messaggio No more coffee. Waiting for recharge ed entra in modalit MAINTENANCE. In questa modalit, il sistema aspetta il ricaricamento che pu avvenire mediante unazione lato PC (es: pulsante di ricarica), corrispondente ad una certa quantit di dosi NC. Avvenuta la ricarica, lato PC viene visualizzato il messaggio Coffee refilled: +NC e la macchina torna in modalit 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 Default Value

- 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 pi opportune.

2 Project Solution

2.1 Introduction to the Approach

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), stato comunque scelto di utilizzare una filosofia di "scorporamento" abbastanza aggressiva per massimizzare la modularit del sistema. Ne risulta un sistema con pi entit in gioco di quelle necessarie, ma il funzionamento non risulta affatto pi complesso di un sistema con minor modulazione.
I moduli, i task e le entit del progetto dialogano attraverso variabili comuni. Non stato volontariamente utilizzato il termine "globali" perch 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' un vero motivo per il quale abbiamo preferito questo approccio ne abbiamo trovato un motivo per il quale questo possa essere svantaggioso, stata quindi una pura scelta implementativa.

2.2 Task Analize

In questa sezione verr analizzato l'utilizzo dei singoli task e la logica che vi dietro. Le FSM verranno mostrate e analizzate in una sezione a se stante posta successivamente a questa.

2.2.1 Scheduler

Questo non un task, ma l'entit 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 determinata dall'interrupt generato dal watchdog timer (settato come interrupt e non reset).

2.2.2 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 verr richiamato dallo scheduler ogni volta che il task dovr essere eseguito.

2.2.3 Distance

Il compito di questo task consiste nel valutare se la distanza acquisit dal sensore di prossimit rientra nel range valutato in uno specifico stato. Se ad esempio ci troviamo nello stato ON, allora sappiamo 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.

2.2.4 MakeCoffee

Se la macchina si trova nello stato MAKING_COFFEE, questo si occuper di "preparare il caffe" e di visualizzare l'avanzamento della preparazione tramite i 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 utilit di questo nel contesto attuale si deciso di integrare le due parti.

2.2.5 Postman

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

2.2.6 Potentiometer

Questo task acquisisce il valore del potenziometro e lo mappa nelle varie possibili quantit 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 StateSwitcher

E' il task che si occupa del passaggio di stato della macchina. La transizione definita dalle variabili settate dagli altri task (e.g.: pir_present).

2.2.9 Timer

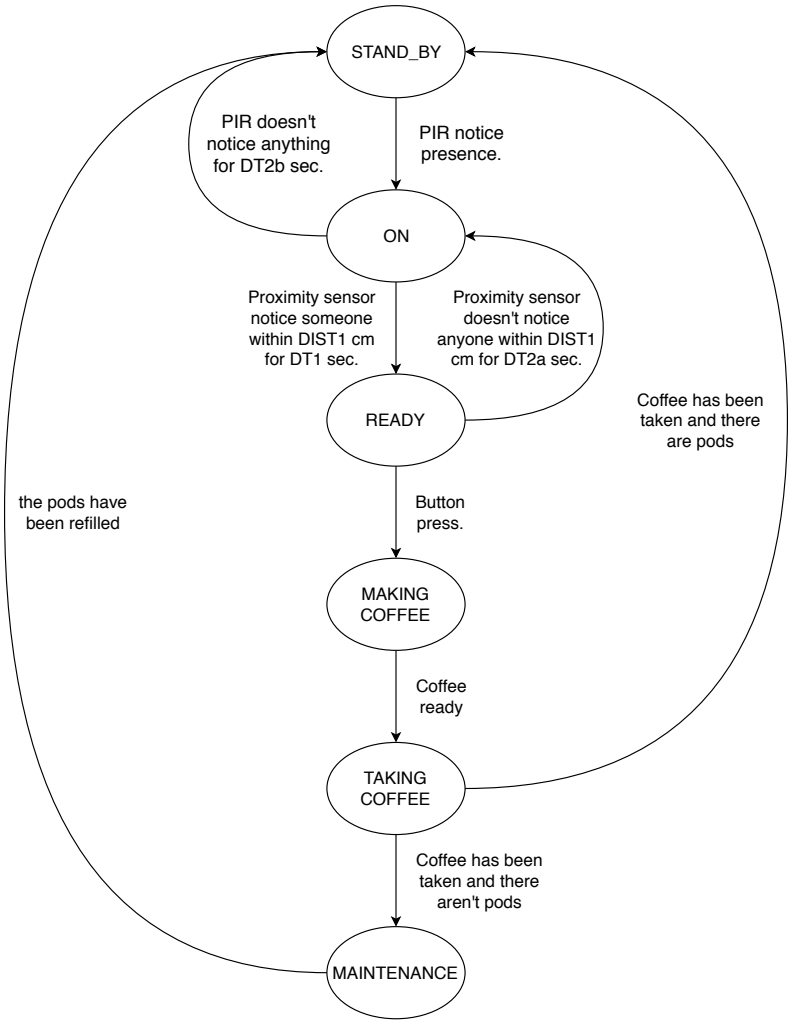
L'idea dietro al task timer che a seconda dello stato generale della machina questo dovr partire (o aspettare in alcuni casi: vedasi stato "READY in range") con l'intento di raggiungere differenti tempistiche. Allo scadere del timer questo notificher l'evento agli altri stati tramite il solo utilizzo di variabili globali. Questa separazione del compito evita che all'inserimento di un nuovo task si debba andare a inserire all'interno del timer un riferimento a quest'ultimo ed

evitando quindi che il timer debba andare a notificare direttamente a questo l'esito.

3 Finite State Machines - Schemes

Sono di seguito presentate le macchine a stati finiti relative alle componentistiche pi importanti e meno banali del programma. Sono quindi escluse dalla trattazione FSM presentanti solo due stati e con condizioni banali (come pu ad esempio risultare quella del sensore di presenza).

3.1 Overview



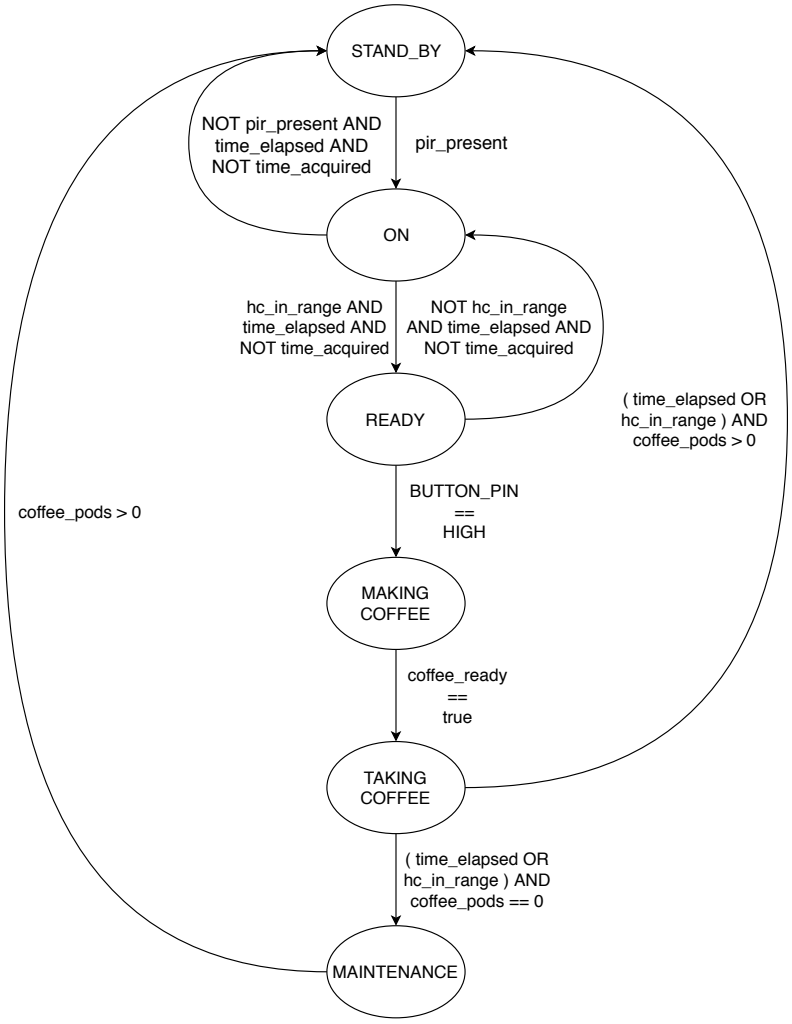
Questa FSM rappresenta il funzionamento generale della macchina, evidenziando i vari stati e le condizioni di transizione a un livello astratto e senza considerare quindi le variabili in gioco nel programma. Come si pu evincere

dallo schema, agli stati definiti nella consegna, sono stati aggiunti due ulteriori stati:

- MAKING COFFEE
- TAKING COFFEE

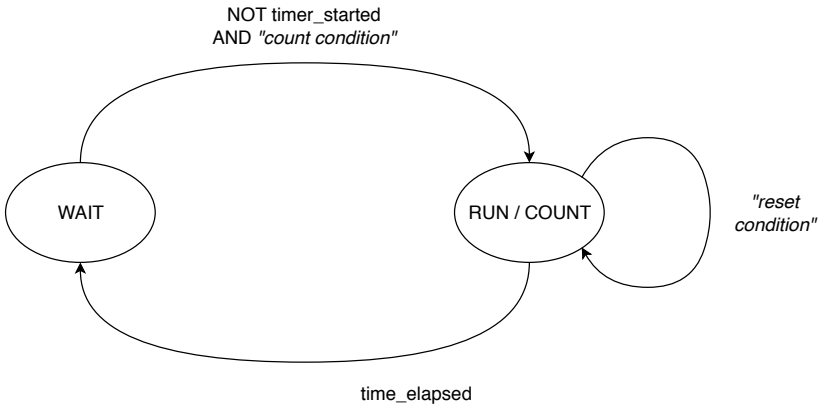
Ci siamo presi la libert di fare ci perch abbiamo considerato l'inserimento di questi due stati pi esplicativo del funzionamento della macchina e perch non abbiamo ritenuto che questo rendesse pi facile o snaturasse l'intento del compito, ovvero l'apprendimento di utilizzo di un modello a task in un sistema integrato.

3.2 State Switcher Task



E' di seguito inserito lo schema della FSM relativa al vero e proprio task "State Switcher". La descrizione di questa non si discosta dalla precedente, ma invece mostra le condizioni di transizione da uno stato all'altro tramite le variabili presenti nel programma.

3.3 Timer Task



Count condition:

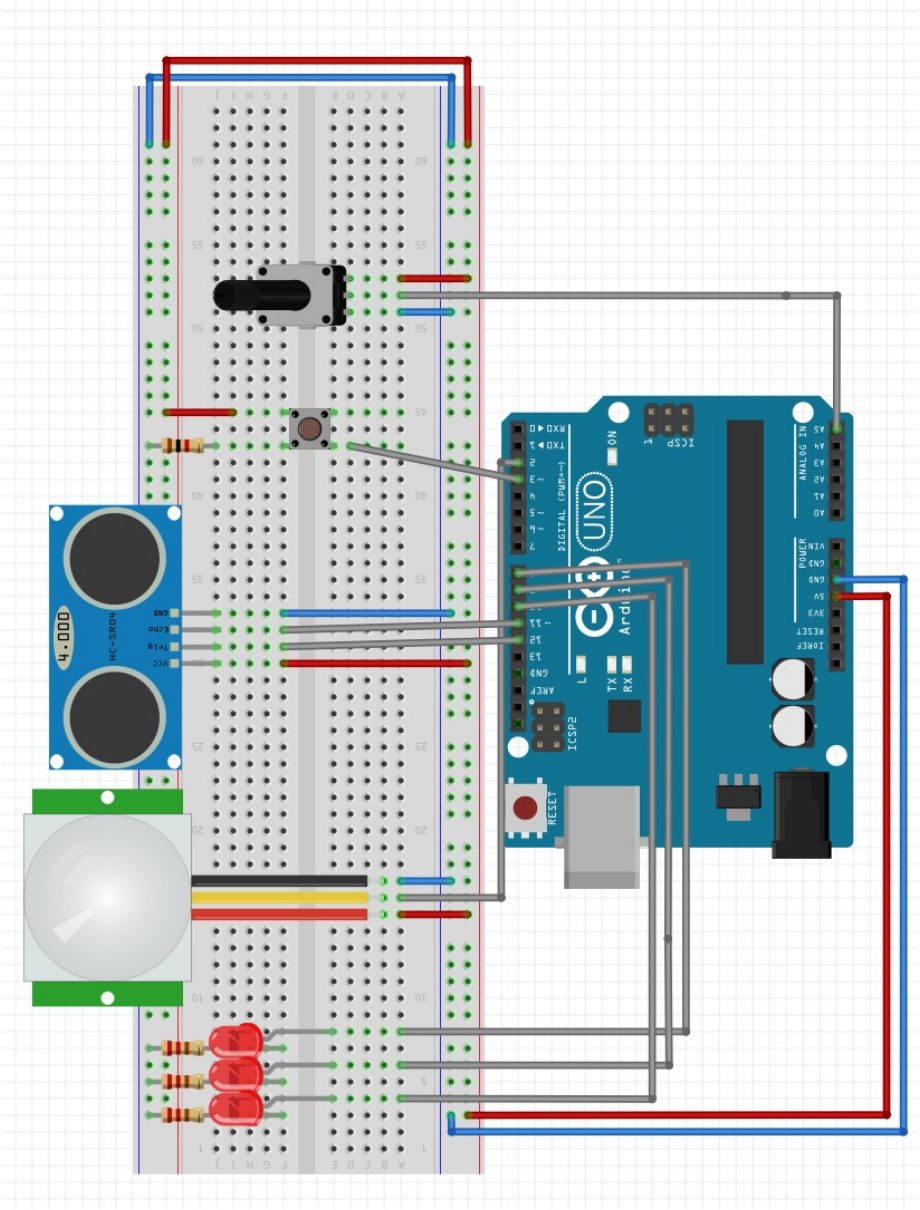
(state == ON AND NOT pir_present)
OR (state == ON AND hc_in_range)
OR (state == READY AND NOT hc_in_range)
OR (state == MAKING_COFFEE)
OR (state == MAKING_COFFEE AND coffee_ready)

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)

La FSM del task Timer molto semplice ma le condizioni di transizione tra i due stati sono varie. Per non appesantire troppo lo schema stato scelto di non mostrare tutte le condizioni direttamente su questo, ma di inserire al loro posto un "riferimento" ad esse.

4 Fritzing


HC-SR04
UNO
ANALOG IN
POWER
DIGITAL (PWM~)
ICSP
ICSP2
RESET
ON
TX
RX