Progetto #2 - Smart Coffee Machine

Lorenzo Casini, Simone Del Gatto, Sophia Fantoni

November 17, 2018

Contents

1 Introduzione

Le funzionalità del sistema sono state ripartite in tre task distinti, ognuno dei quali è descritto da una macchina a stati sincrona. I tre task comunicano tra di loro tramite l'uso di variabili globali. In una prospettiva orientata agli oggetti ogni task è una classe che implementa l'interfaccia Task. Il coordinamento tra i vari task è gestito dallo Scheduler, anch'esso pensato come un oggetto, che si occupa di far eseguire i task in ordine. Nonostante la presenza di uno Scheduler il multi-tasking è gestito in maniera cooperativa tra i vari task. In questo caso quindi non si dovranno gestire problematiche di corse critiche o di concorrenza perchè il comportamento del sistema, nonostante la divisione dei lavori, è comunque di tipo sequenziale. Ogni task è eseguito con un determinato periodo e lo Scheduler lavora con un periodo che è il massimo comune divisore tra i task che deve gestire.

Ogni task gestisce anche un insieme di componenti hardware, resi in forma digitale tramite il paradigma ad oggetti. Si troveranno all'interno del progetto la classe Pir, Sonar, Button, Led e Potentiometer che non solo hanno la classica funzione di emulazione della realtà tipica della programmazione ad oggetti, ma sono pensati per incapsulare tutte quelle funzionalità che il framework Wiring mette a disposizione astraendo il più possibile dall'hardware sottostante.

2 Variabili Globali

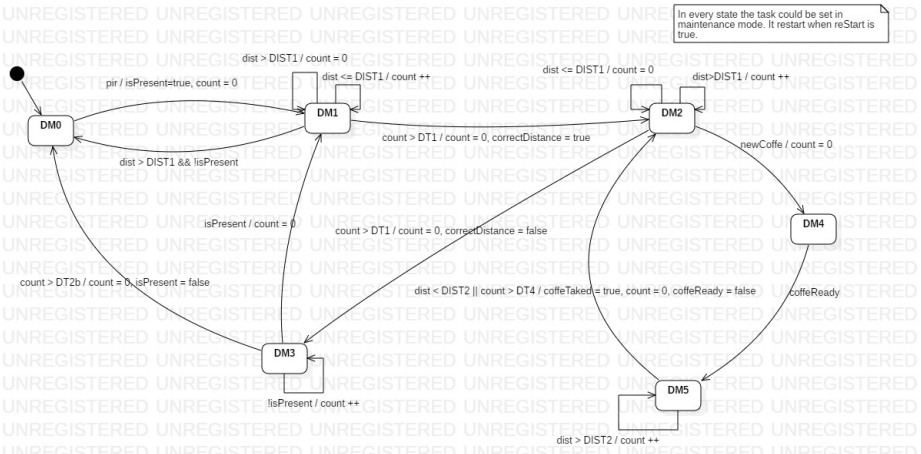
- **bool isPresent:** la variabile indica se è stato rilevato un movimento nei pressi della macchina.
- **bool correctDistance:** indica se l'utente si trova alla distanza di engagement per un tempo maggiore di DT1.
- **bool newCoffe:** indica che l'utente ha richiesto un caffè.
- **bool coffeReady:** indica che il caffè è pronto.
- **bool coffeTaked:** indica che l'utente ha preso il caffè.
- **int numCoffe:** numero di caffè ancora disponibili.
- **bool maintenaceActive:** indica se è attiva la modalità di manutenzione. Disabilita tutti i task tranne UserHandler.
- **bool reStart:** variabile utilizzata per ripristinare il sistema dopo la fase di manutenzione.

3 Detect Motion

Questo task si occupa della rilevazione dell'utente avvalendosi dell'uso del pir e del sonar e, in base alle rilevazioni effettuate, setta le variabili booleane globali utili per la comunicazione con gli altri elementi del sistema. In particolare il task si occupa di:

- Rilevare il movimento di un individuo
- Rilevare e notificare la sua permanenza di fronte alla macchina
- Una volta che è stato ordinato un caffè deve rilevare e notificare se questo è stato preso
- Gestire l'eventuale comportamento impredicibile dell'utente che potrebbe avvicinarsi e allontanarsi dalla macchina

3.1 FSM



3.2 Comportamento

Il comportamento della macchina si articola in sei stati:

- **DM0:** il task quando si trova in questo stato controlla il valore del pir. Nel caso in cui rilevi la presenza di un utente, setta la variabile isPresent a true e cambia lo stato passando in DM1.
- **DM1:** quando si trova in questo stato la macchina del caffè deve valutare se l'utente per un tempo DT1 rimane a una distanza inferiore o uguale a DIST1. Se ciò accade il task passa allo stato DM2 settando la variabile correctDistance a true. Altrimenti controlla il pir, se questo registra una movimento la macchina rimane in questo stato. Nel caso in cui non venga rilevato più movimento il task torna nello stato precedente e setta la variabile isPresent a false.

- **DM2:** rimane in questo stato finché l'utente risulta essere ad una distanza inferiore a DIST1, ma non svolge altre azioni. Se però il sonar rileva che l'utente si è allontanato viene settata a false la variabile correctDistance e il task passa nello stato DM3. Nel caso in cui l'utente chieda un caffè invece la macchina passa nello stato DM4.
- **DM4:** in questo stato il task aspetta l'erogazione del caffè, quindi non effettua rilevazioni e cambierà stato solo quando il caffè sarà pronto.
- **DM5:** il task adesso deve rilevare quando l'utente prende il caffè. Se il sonar rileva che l'utente si avvicina a una distanza inferiore o uguale a DIST2 o è trascorso un tempo DT4, allora cambia stato e torna a DM2 settando la variabile coffeTaked a true e coffeReady a false.
- **DM3:** in questo stato viene controllato il pir, se per un tempo maggiore di DT2b non viene rilevato nessuno allora passo allo stato DM0 altrimenti se viene rilevato qualcuno passo allo stato DM1.

Una volta erogato il caffè, numCoffe potrebbe diventare uguale a 0 e il sistema deve entrare in modalità maintenace. Il task non dovrà quindi fare più alcuna operazione fino a che non gli viene detto di ripristinarsi. Tutte le volte che il metodo tick() del task viene chiamato vengono quindi controllati i valori di maintenaceActive e reStart.

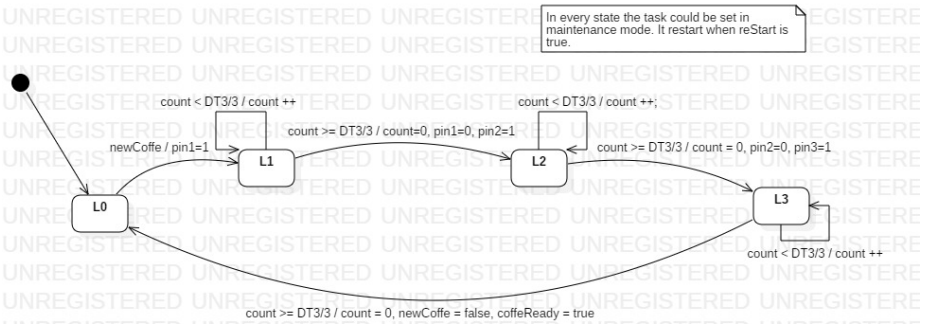
3.3 Considerazioni

Nella fase di testing di questo task sono stati riscontrati alcuni problemi con il pir e il sonar. Il pir è risultato uno strumento poco preciso, in certi casi il sistema sembra essere poco reattivo, ma tale problema è dato dalla sensibilità dello strumento.
Il sonar a volte esegue misurazioni sbagliate con uno scarto molto importante rispetto ai valori precedenti compromettendo talvolta il comportamento del sistema. Per risolvere il problema non potendo considerare sbagliate a priori misurazioni troppo diverse dai valori precedentemente rilevati abbiamo optato per effettuare una media ponderata tra la media dei valori precedentemente misurati (70%) e l'ultimo valore misurato (30%). Questa soluzione risulta essere efficiente nell'offuscare i valori errati, ma allo stesso tempo potrebbe causare un ritardo nella notifica delle variazioni effettive. Impostando in modo adeguato il periodo con cui lavora la macchina dovrebbe essere garantita una risposta rapida dello strumento agli stimoli esterni. La scelta del periodo è stata fatta anche per ridurre al minimo i problemi di rilevazione del pir utilizzato.

4 Make Coffe

Questo task simula l'erogazione del caffè, gestendo l'accensione progressiva dei tre led.

4.1 FSM



4.2 Comportamento

La macchina presenta quattro stati:

- **L0:** è lo stato di iniziale, in cui il sistema rimane finché non viene richiesto un nuovo caffè. In tal caso accende il primo led ed effettua il cambio di stato a **L1**.
- **L1:** il primo led rimane acceso per DT3.
- **L2:** viene spento il primo led e acceso il secondo.
- **L3:** è l'ultimo stato, in cui viene acceso il terzo led e spento il secondo. Dopo DT3 anche il terzo led viene spento e la macchina torna nello stato iniziale **L0**. La variabile newCoffe viene settata a false, mentre viene comunicato al sistema che il caffè è pronto tramite il settaggio a true della variabile coffeReady.

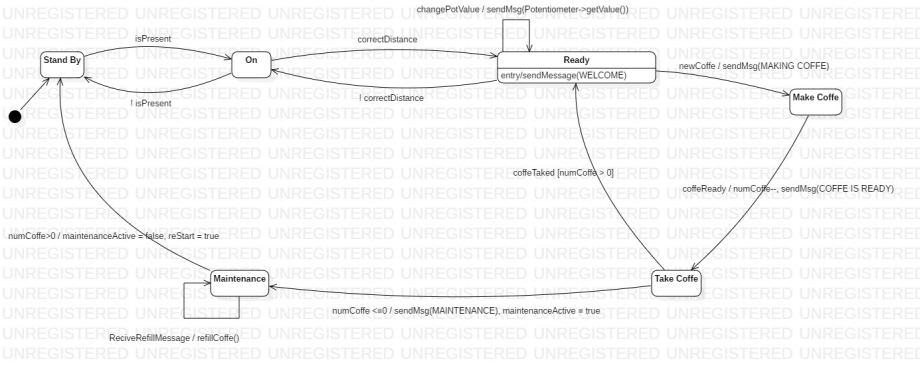
5 User Handler

Questo task è responsabile dell'interazione con l'utente, sia per scambiare messaggi con esso attraverso l'interfaccia grafica, che per accettare e gestire richieste dell'utente. In particolare si occupa di:

- Gestire la richiesta di un caffè comunicata dall'utente tramite la pressione di un pulsante tattile.
- Stampare sull'interfaccia grafica messaggi per la comunicazione con l'utente.
- Gestire il caricamento del caffè da parte di un agente esterno.

Grazie a questo task inoltre è possibile intuire più facilmente il comportamento generale della Smart Coffe.

5.1 FSM



5.2 Comportamento

Il suo comportamento si esprime attraverso sei stati esplicitati nell'enumerazione MainState:

- **STANDBY:** rappresenta lo stato di partenza, finché la macchina rimane in questo stato, è attiva la modalità di risparmio energetico. Eseguirà il passaggio di stato verso **ON** solo nel momento in cui verrà rilevato l'utente.
- **ON:** il sistema rimarrà in questo stato fino a che non ci si è accertati dell'effettiva presenza di un utente, cioè finchè le variabili correctDistance = false e isPresent = true. Nel momento in cui correctDistance assume valore true, si effettuerà il passaggio di stato verso **READY**, inviando un messaggio tramite la seriale verso l'interfaccia grafica. La GUI una volta ricevuto il messaggio mostrerà la scritta "WELCOME". Invece, nel caso in cui l'utente si allontani si effettuerà un controllo sulla reale presenza di qualcuno nei pressi della macchina. In caso negativo si ritorna allo stato **STANDBY**.
- **READY:** verificato che l'utente si trova a una distanza corretta, gli viene concesso di ordinare un caffè tramite la pressione del pulsante tattile.

Nel momento in cui avviene la richiesta, viene settata a true la variabile newCoffe e si passa allo stato **MAKECOFFE** inviando un messaggio sulla seriale, la GUI visualizzerà il messaggio "MAKING A COFFE". Quando la macchina è in questo stato viene data la possibilità all'utente di regolare il livello dello zucchero tramite il potenziometro. Alla GUI verranno notificati eventuali cambiamenti tramite seriale in modo da renderli visibili all'utente.

- **MAKECOFFE:** in questo stato la macchina sta erogando il caffè. In fase di uscita viene inviato un messaggio tramite seriale alla GUI, la quale stampa la stringa "COFFE READY". Viene inoltre decrementato numCoffe.
- **TAKECOFFE:** in questo stato la macchina controlla il valore della variabile coffeTaked, nel caso in cui essa sia true e numCoffe > 0, si tornerà in stato di **READY**. Nel caso in cui sia più possibile erogare il caffè la macchina passa nello stato **MAINTENANCE**, sulla seriale viene inviato il messaggio "NO MORE COFFE WAITING FOR RECHARGE" alla GUI.
- **MAINTENANCE:** la macchina non è in grado di erogare ulteriori caffè. L'utente può effetturare la ricarica premendo il bottone sulla GUI, in questo caso verrà inviato un messaggio sulla seriale, che incrementerà numCoffe e la macchina tornerà allo stato di **STANDBY**. Fino a quando la macchina resta in questo stato la variabile maintenanceActive rimane settata a true. Quando effettua il cambio di stato viene settata a true la variabile di reStart per riabilitare opportunamente il sistema.

5.3 Considerazioni: Sleep Mode

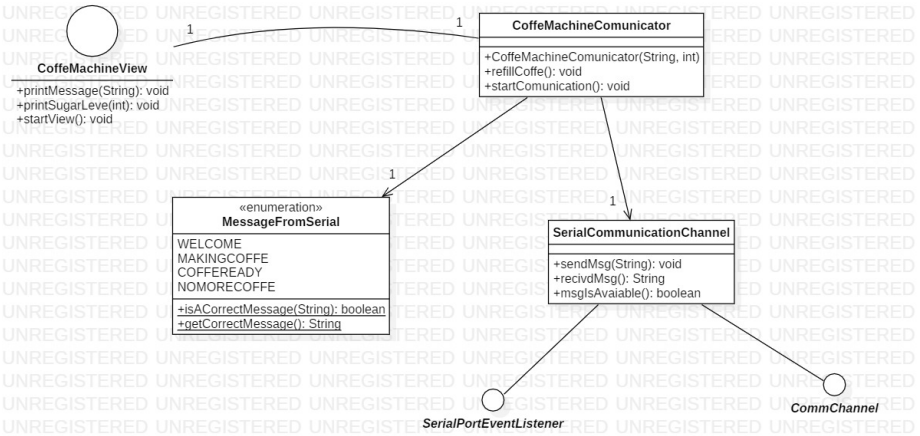
Strettamente correlata a questo task è la discussione legata alla richiesta di mettere il sistema in risparmio energetico. In particolare sono state elaborate tre possibili strategie implementative:

- La prima è quella di creare uno scheduler intelligente che, dopo aver eseguito tutti i task, si mette in sleep mode. Tale soluzione però non risulta essere ottimale in quanto lo sleep mode non viene attivato direttamente nello stato **STANDBY**, quello in cui è effettivamente richiesta.
- La seconda è quella di mettere in sleep mode il sistema durante lo stato **STANDBY**. La soluzione sembra apparentemente più intuitiva, ma lascia qualche perplessità. La modalità di scheduling è cooperativa ma, nel momento in cui un singolo task manda l'intero sistema in sleep mode, questa sembra essere in conflitto con l'idea di scheduler come regista del sistema. Inoltre, a svegliare il task non è una componente logica come un metodo, ma è il tic del timer. Questo elemento è completamente incapsulato nel funzionamento dello scheduler, poichè il suo tick genera un'interruzione, non è possibile isolarlo mascherando il suo comportamento dal punto di vista hardware. E' evidente che c'è una forte dipendenza tra l'implementazione dello scheduler e il task che non dovrebbe sussistere.

– L'ultima possibile soluzione è quella di utilizzare il Pir come sorgente di interruzioni. In questo modo il Timer non interagisce più con il task e rimane incapsulato nello Scheduler, tuttavia anche in questo caso il task si trova a dipendere dal comportamento di un dispositivo che non gli è propriamente associato.

La seconda soluzione è più attinente al testo dell'esercizio quindi si è optato per quest'ultima.

6 Comunicazione seriale e GUI



La comunicazione seriale è gestita tramite scambio di caratteri tra Arduino e un programma Java su PC. Tramite la classe CoffeMachineComunicator si attiva un thread che legge i messaggi inviati da Arduino e li elabora opportunamente. Per fare ciò utilizza l'enumerazione MessageFromSerial che rappresenta tutti i possibili messaggi che le due parti si possono scambiare. Alcune delle informazioni vengono utilizzate per notificare all'utente lo stato della macchina tramite una GUI apposita, il cui comportamento è descritto nell'interfaccia CoffeMachineView. Altre invece fanno in modo che vengano rese disponibili alcune operazioni come il caricamento del caffè tramite bottone grafico.

La GUI lato PC è stata implementata utilizzando javaFX. Per la comunicazione seriale si è deciso di utilizzare la libreria JSSC vista la sua caratteristica di non dover essere configurata in base allo specifico sistema su cui si trova il software.

7 Schema del circuito