Università di Pisa
Embedded Computing Systems

Design of Embedded Systems

Stefano Fiori Michele Baldassini

Relazione

Sistema dedicato alla gestione di un parcheggio

Pisa, 24/02/2017

1 Introduzione

Questo documento contiene la descrizione della progettazione e dello sviluppo di un sistema di gestione di un parcheggio, intendendo con quest'ultimo un parco auto al chiuso (struttura dedicata con unico scopo di sostare le auto).

2 Requisiti Utente

Il sistema deve occuparsi della gestione di un parcheggio.

L'interazione con il sistema, come verrà analizzato successivamente, avviene con due tipi di utente diversi.

In generale le funzioni che esso deve svolgere sono:

- Tenere conto dei parcheggi totali liberi e occupati
- Tenere conto del tempo per il quale ogni parcheggio è rimasto occupato(al fine di associarne un prezzo)
- Quando un parcheggio viene occupato il sistema deve notificare tale evento al gestore del parcheggio il quale potrà successivamente accettare la "richiesta di sosta" e far partire il conteggio del tempo.
- Quando un parcheggio viene liberato il sistema deve notificare tale evento al gestore del parcheggio il quale, una volta ricevuto il pagamento da parte del cliente, renderà disponibile il parcheggio ad altri clienti.
- Quando il parcheggio viene occupato il sistema notifica tale evento al gestore del parcheggio il quale sarà libero o meno di fornire il servizio di sosta attraverso un'interazione con il sistema.
- Se un cliente libera il parcheggio prima che il gestore abbia accettato il suo arrivo, il parcheggio diventa di nuovo disponibile
- Nel caso in cui un cliente occupi un parcheggio che si è appena liberato e del quale non è stato ancora accettato il "pagamento": non appena il gestore avrà accettato la partenza del cliente uscente dovrà accettare la richiesta del cliente entrante.

2.1 Tipi Utente

Gli utenti che interessano il sistema sono due:

1. Il gestore del parcheggio, colui che offre il servizio di sosta.
2. Autisti, che intendono sostare e, quindi, usufruire del servizio.

Il parcheggio sarà dedicato al solo parcheggio di auto. Per questa categoria si intendono auto che si utilizzano tipicamente in città (auto familiari, sportive, Suv, etc…).

2.2 Dimensioni Utente

Il parcheggio sarà alto 2.00m.

Dato che le auto saranno quelle tipicamente quelle utilizzate in città, le loro dimensioni saranno circa:

	Min	Media	Max	
Altezza	1	1,5	2	metri
Larghezza	1,2	1,5	1,8	metri
Lunghezza	7		5	metri
Peso	700	1200	2000	kilogrammi

3 Specifiche Funzionali

Per il rilevamento della presenza delle auto sono stati scelti i sensori a ultrasuoni, uno per ogni parcheggio.

Mentre per l'interazione con il gestore del parcheggio è stato scelto uno schermo LCD-Touch; su quest'ultimo avviene anche la visualizzazione dello stato in cui si trova il parcheggio.

3.1 Requisiti funzionali

- La presenza dell'auto viene rilevata da un sensore a ultrasuoni
  - Il rilevamento avviene calcolando la distanza del pianale sotto-scocca[1] dal suolo: quando quest'ultima è sotto il massimo il parcheggio viene occupato.
  - Quando la distanza rilevata è sotto il minimo ammissibile si deve segnalare tale situazione al gestore del parcheggio
- L'interazione con il gestore del parcheggio avviene attraverso uno schermo LCD/Touch
  - Quando un parcheggio viene occupato il gestore deve toccare l'immagine del parcheggio interessato al fine di accettare la richiesta di servizio.
  - Quando un parcheggio viene liberato il gestore deve toccare l'immagine del parcheggio interessato al fine di accettare il rilascio del servizio.

3.2 Dimensioni Funzionali

Dato che si è scelto di rilevare la presenza di un autovettura attraverso l'utilizzo di un sensore a ultrasuoni posto al livello del suolo si vede necessaria l'introduzione di altre misure, e cioè della distanza dal suolo del pianale sotto-scocca.

	Min	Media	Max	
Distanza del pianale sotto-scocca dal suolo	10	30	50	cm

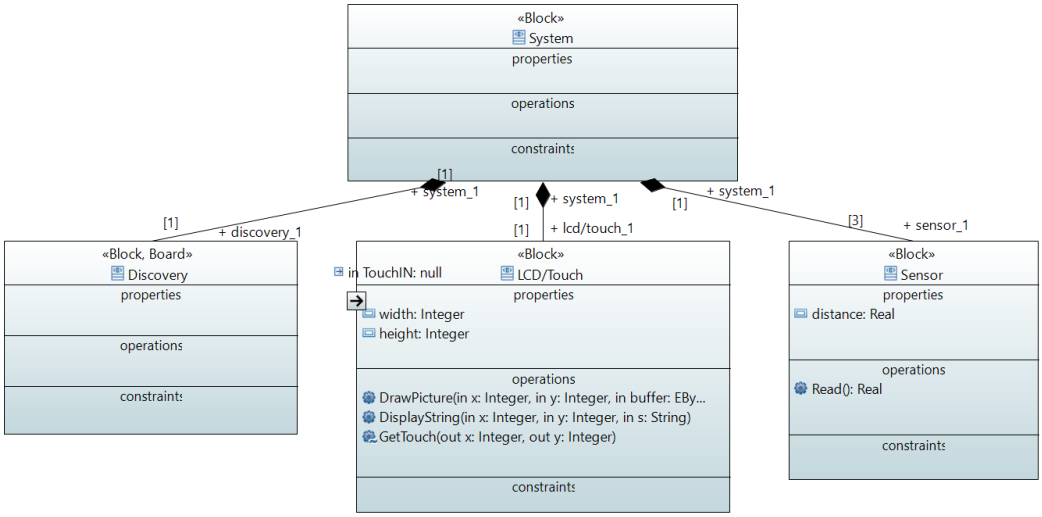
[1] E' la parte inferiore dell'auto, quella più vicina a suolo dopo le ruote.

4 Progettazione

4.1 Hardware

In questa sezione vengono analizzati quali sono i componenti fisici dell'intero sistema necessari a soddisfare i requisiti utente.

Di seguito il Block Definition Diagram che da un idea generale dei componenti coinvolti nella realizzazione:



4.1.1 Board

La scheda sulla quale vengono eseguiti i processi che si occupano di soddisfare i requisiti è la stm32f4-Discovery

4.1.2 Sensori

Dal punto di vista hardware i sensori a ultrasuoni scelti sono gli HYSR05.

Tali sensori possiedono 5 Piedini:

1. VCC — Alimentazione 5v
2. TRIGGER — Piedino di Input
3. ECHO — Piedino di Output
4. OUT — non utilizzato
5. GND — Ground

4.1.2.1Funzionamento

Per il corretto funzionamento dei sensori si assume che essi siano alimentati regolarmente e che il trigger sia, in condizioni di riposo, a livello logico basso.

Verificate tali assunzioni per una lettura si procede nel seguente modo:

1. Mandando un impulso di almeno 10us sul piedino Trigger, questo risulta in un comando di lettura per il sensore.
2. Calcolando successivamente la durata per il quale il piedino Echo rimane a uno; tale durata sta ad indicare quanto tempo ha impiegato un suono ad andare e tornare da sensore.
3. Perché il suono torni alla fonte e necessario che rimbalzi su qualche superficie; nel caso in cui non sia presente una superficie abbastanza vicina la lettura deve tornare un valore massimo.

4.1.3 LCD-Touch

Lo schermo LCD-Touch scelto è lo schermo standard in dotazione con la Discovery.

Per l'utilizzo di tale dispositivo si fa perciò utilizzo delle funzioni di libreria fornite dalla casa produttrice stessa.

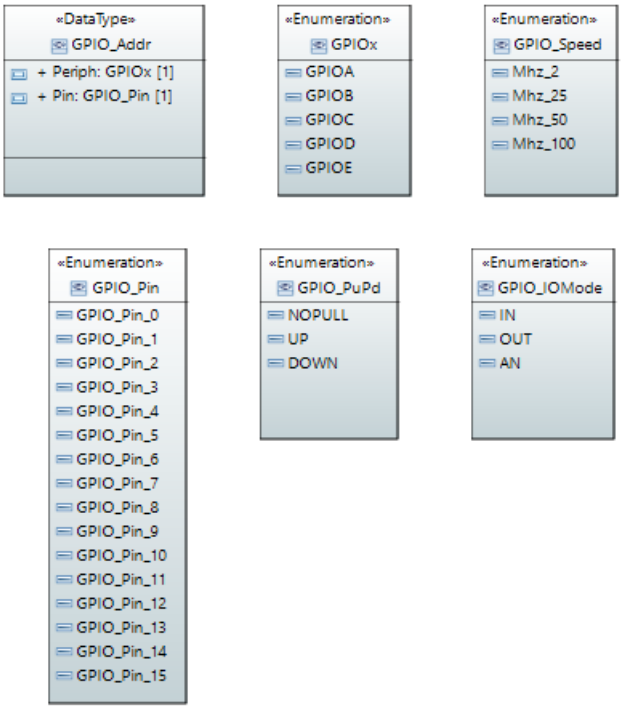
4.1.4 Interconnessione

I sensori e lo schermo sono connessi alla board tramite i piedini di GPIO(General Purpose Input/Output). Tali piedini possono essere configurati a livello software al fine di gestire una gran parte di dispositivi esterni e creare protocolli di comunicazione più complessi.

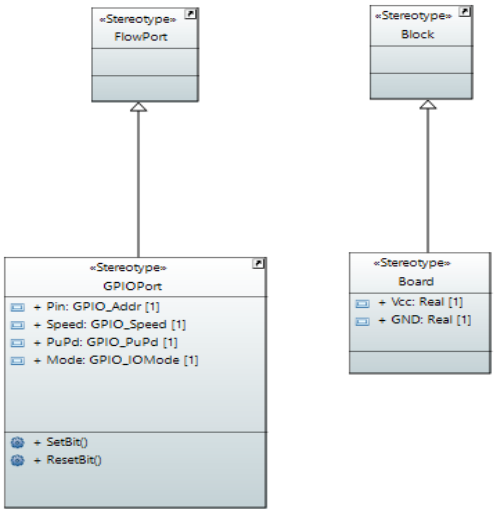
La board è dotata di 80 GPIO, 40 dei quali però sono già utilizzati dallo schermo. I restanti 40 piedini possono essere liberamente utilizzati per collegare i sensori.

Per la configurazione e le operazioni di lettura e scrittura si può fare utilizzo delle funzioni di libreria della casa produttrice della board.

La seguente immagine chiarisce i parametri che caratterizzano un GPIO

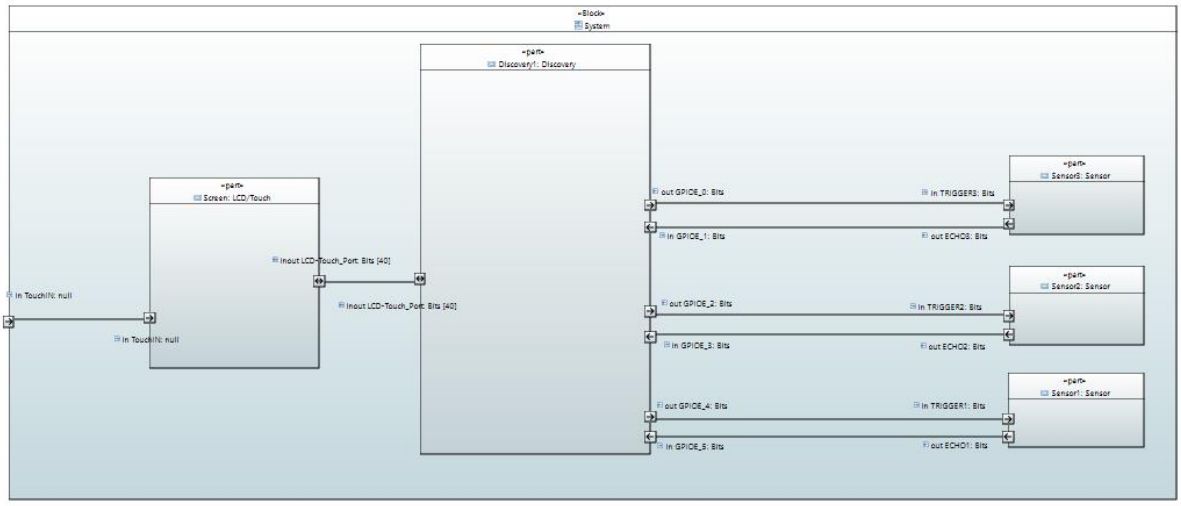


In generale quindi i vari componenti comunicano attraverso GPIO, e al fine di modellare questo concetto è stato creato un profilo contenente i precedenti “types” e un estensione alle classiche “FlowPort” del Sysml che rappresentano le GPIOPort.

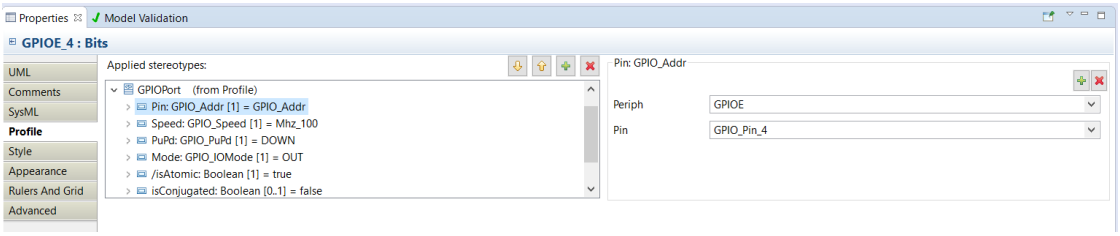


E stato incluso nel profilo anche il concetto di Board.

Una volta definite le GPIO Port si può passare ad analizzare come vengono interconnessi i vari componenti: di seguito l’Internal Block Diagram con 3 sensori come esempio



E come esempio la specifica per la GPIO Port connessa al TRIGGER del sensore 1:



4.2 Software

Questa sezione tratta la progettazione delle operazioni che vengono eseguite dal processore, al fine di soddisfare i requisiti utente utilizzando tutto ciò che è stato definito al capitolo Harware.

Le operazioni principali sono suddivise in:

1. Lettura da ogni sensore della distanza
2. Lettura del Touch
3. Aggiornamento dello schermo

4.2.1 Lettura di sensori

Il processo che si occupa della lettura dei sensori sarà quello a periodo più breve e priorità più alta, questo perché l'attesa che il piedino ECHO vada a livello logico basso e un attesa attiva. L'interruzione di tale attesa, da parte di eventuali processi a maggiore priorità, implicherebbe la lettura di un dato errato;

Il processo una volta letta la distanza:

- se essa è tale da assicurare la presenza di un auto setta l'evento per il parcheggio interessato.
- se essa è tale da assicurare l'assenza di un auto resetta l'evento per il parcheggio interessato.

4.2.2 Lettura Touch

Il Processo che si occupa della lettura del Touch ha il secondo periodo più breve e quindi il secondo per priorità.

Questo task legge periodicamente il Touch e controlla se è avvenuto un tocco.

Se è avvenuto un tocco controlla a quale parcheggio appartiene e, per il parcheggio interessato, setta l'evento "Tocco".

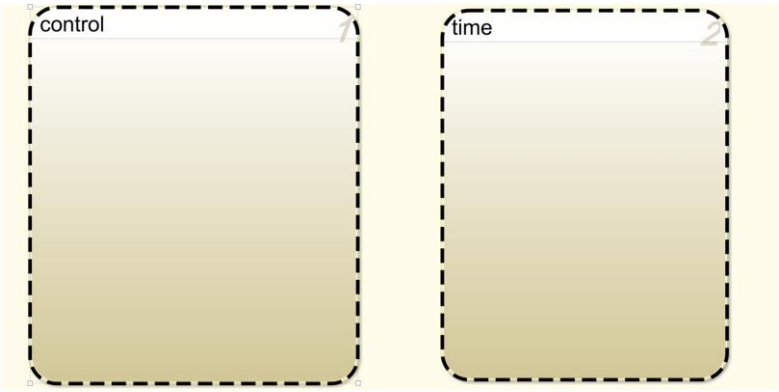
4.2.3 Aggiornamento Schermo

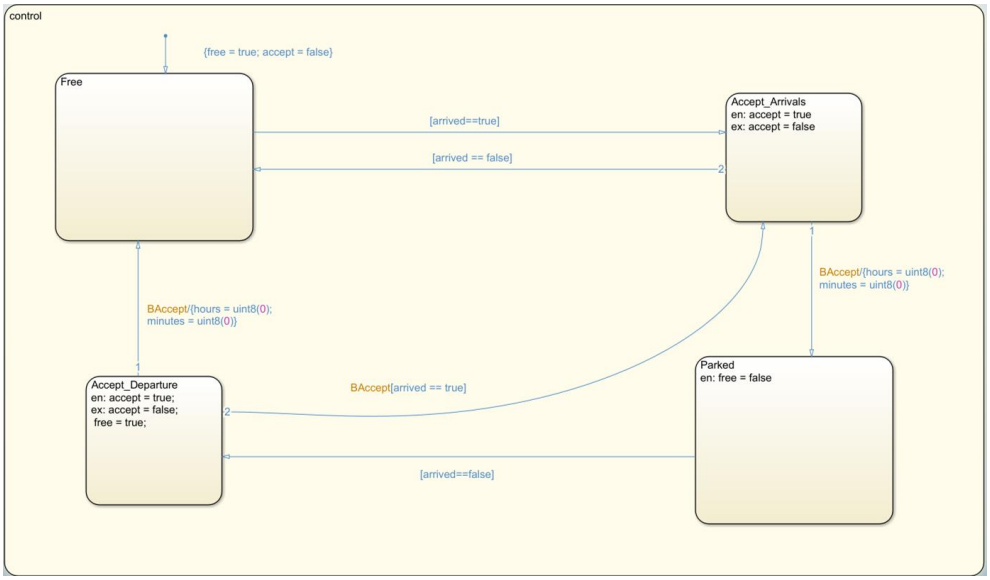
Questo task aggiorna le informazioni di ogni parcheggio visualizzate sullo schermo.

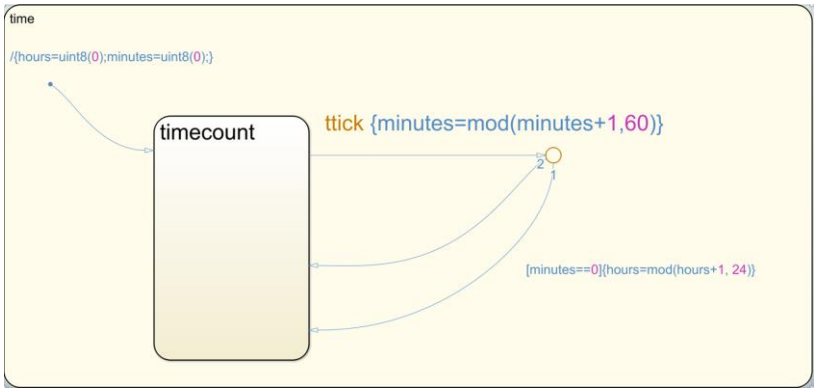
Al fine di mantenere tali informazioni è stato scelto di gestire ogni parcheggio con una macchina a stati i quali input e output sono:

- Input
  - tick evento utile a fornire il concetto di tempo.
  - BAccept l'area dello schermo relativa al parcheggio è stata toccata.
  - arrived
    - se settato indica che è presente un auto sopra al sensore.
    - se resettato indica che non è presente un auto sopra al sensore.
- Output
  - free se settato indica che il parcheggio è libero.
  - accept se settato indica che quel parcheggio sta attendendo l'accettazione.
  - hours da quante ore è occupato il parcheggio.
  - minutes da quanti minuti è occupato il parcheggio.

Immagine della macchina a stati:







L’immagine del parcheggio sullo schermo viene quindi aggiornata dipendentemente dagli output della macchina a stati dello stesso.

5 Unit Test

La fase di test è stata eseguita con CUunit e interessa i moduli che non fanno uno stretto utilizzo delle librerie fornite dalla casa produttrice della Board.

Pertanto la parte testata è la read del sensore, la quale implementa un filtro passa basso.

5.1 Sensor Test

Sul sensore sono stati eseguiti due test, simulando la lettura dal GPIO di un tempo per il quale il piedino ECHO rimane attivo alto.

I dati coinvolti in questo test sono:

- float rawdata: distanza restituita da read grezza(non filtrata) dal sensore(simulato)
- float olddist: vecchia distanza letta con una read.
- float distance: ultima distanza letta con una read.

In ogni caso la distanza dovrà comunque essere maggiore di zero e minore della distanza massima impostata:

- CU_ASSERT(distance < = HYSR05_DMAX);
- CU_ASSERT(distance >= 0);

5.1.1 Rising Test

In questo test viene simulato un oggetto che si allontana dal sensore(distanza crescente) incrementando il time. Ci si aspetta che la read torni un valore che è minore alla distanza letta dal sensore ma maggiore della vecchia distanza letta.

- CU_ASSERT(distance < raw_data);
- CU_ASSERT(distance > old_dis);

5.1.2 Falling Test

In questo test viene simulato un oggetto che si avvicina al sensore(distanza decrescente) decremntando il time. Ci si aspetta che la read torni un valore che è maggiore alla distanza letta dal sensore e minore della vecchia distanza letta.

- CU_ASSERT(distance > raw_data);
- CU_ASSERT(distance < old_dis);

5.1.3 Risultati dei Test

```
michele@michele-VirtualBox:~/CUnit/filter$ make
gcc -c progs//test_filter.c -Iinc/ -o obj/test_filter.o
gcc -c testedCode//filter.c -Iinc/ -o obj/filter.o
gcc -c testedCode//globals.c -Iinc/ -o obj/globals.o
gcc --static  obj/test_filter.o obj/filter.o obj/globals.o -L /usr/local/lib/ -lcunit -o test_filter
michele@michele-VirtualBox:~/CUnit/filter$ ./test_filter


     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/


Initializing suite1

Suite: Suite_1
  Test: Filter_Init(Filter *, 0) ...passed
  Test: Filter_Init(Filter *, HYSR05_DMAX) ...passed
  Test: Filter_Read_Rise(Filter *) ...passed
  Test: Filter_Read_Fall(Filter *) ...passed
Completing suite1


Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      4      4      4      0        0
             asserts  58002  58002  58002      0      n/a

Elapsed time =    0.000 seconds
```