

# WIRELESS SIGNAL STRENGTH METER

Stefano Cicero

Corso di Sistemi Concorrenti e Distribuiti – modulo Sistemi Real Time  
Corso di Laurea Magistrale in Ingegneria Informatica  
Università di Pisa  
A.A. 2011-2012

## Sommario

<i>1.INTRODUZIONE</i>	3
<i>2.REQUISITI HARDWARE E SOFTWARE</i>	3
<i>3.ACQUISIZIONE POTENZA SEGNALE WIRELESS</i>	5
<i>4.MAPPATURA POTENZA SEGNALE WIRELESS SU ARRAY DI LEDS</i>	5
<i>5.FUNZIONAMENTO GLOBALE</i>	6
<i>6.LOG</i>	7
<i>7.ALLEGATI</i>	7

## 1. INTRODUZIONE

L'obiettivo del progetto consiste nella realizzazione di un dispositivo che, mediante un array di leds, indichi la potenza del segnale wireless proveniente da un altro dispositivo dello stesso tipo. La potenza del segnale wireless varierà in base alla distanza dei due dispositivi, e/o in base alla presenza o meno di ostacoli fisici tra i due.

## 2. REQUISITI HARDWARE E SOFTWARE

L'hardware utilizzato, mostrato in figura 1, consiste in:

- 2x Evidence FLEX Mini kit
- Array di led collegato a una delle due Mini Board
- Convertitore seriale – USB
- Programmatore/debugger MPLAB ICD 2
- Cavo USB A-B
- Cavo USB A-mini USB A
- Cavo RJ-11 a 6 pin
- 2x alimentatore da 12 V
- 2x batteria da 9 V



Figura 1

Ciascun FLEX Mini kit, mostrato nelle figure 2 e 3, è composto da:

- PIC24FJ64JA004 (16 Mhz) microcontroller
- PIC battery monitor
- Real-time clock
- 1 KB Serial EEPROM
- ZigBee
- Pins for PICkit programming
- 3-axis accelerometer (selectable range from 1.5g to 6g)

- Buzzer
- Thermal sensor
- Light sensor
- 2x DIP switches
- 2x LEDs
- 9 V battery connector + DC in jack

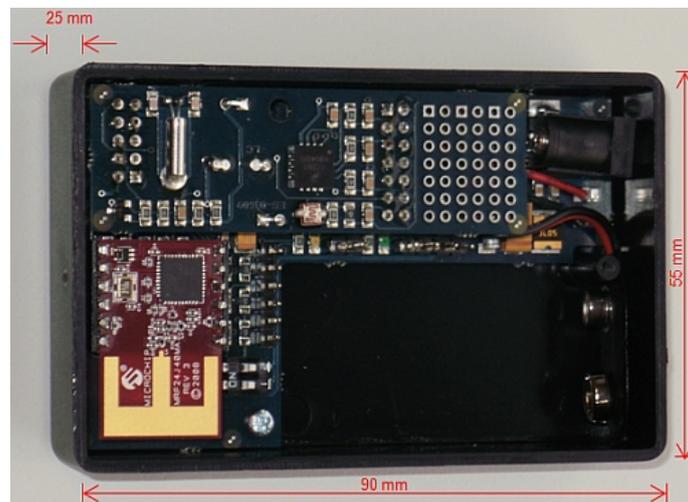


Figura 2

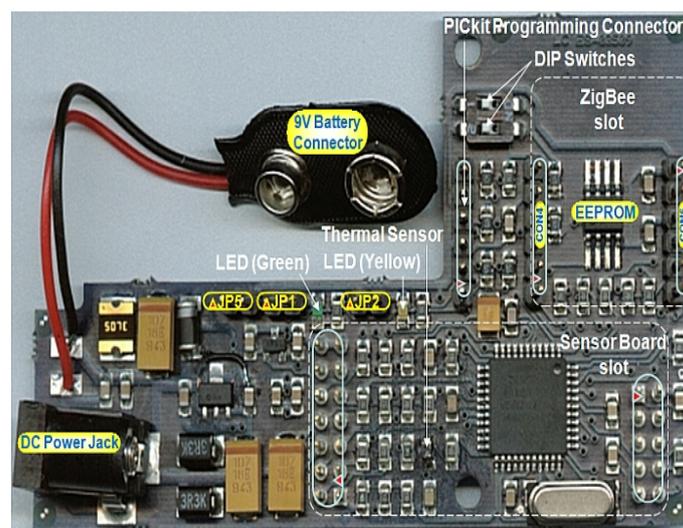


Figura 3

Il software utilizzato su sistema operativo Windows Xp Professional SP3 consiste in:

- MPLAB IDE
- Eclipse
- RT-Druid plugin
- Erika Enterprise Kernel
- HTerm

### 3. ACQUISIZIONE POTENZA SEGNALE WIRELESS

Le due MiniFlex, per comunicare, utilizzano il protocollo wireless MiWi P2P, che è una variazione del IEEE 802.15.4 . Il protocollo fornisce una comunicazione wireless diretta e affidabile, tramite un'interfaccia di programmazione di facile utilizzo. Le principali caratteristiche di questo protocollo sono:

- Fornisce 16 canali nello spettro dei 2.4 GHz
- Opera su piattaforme Microchip PIC18, PIC24, dsPIC33 e PIC32
- Supporta i compilatori Microchip C18, C30 e C32
- Funziona come una macchina a stati (non dipendente da RTOS)
- Supporta un dispositivo dormiente (sleeping) al termine della comunicazione
- Consente di effettuare una scansione energetica (energy detect) per operare sul canale con meno disturbi
- Supporta la scansione attiva per rilevare connessioni esistenti
- Supporta tutte le modalità di sicurezza definite nel IEEE 802.15.4
- Supporta il channel hopping

Utilizzando tale protocollo, le MiniFlex creano una connessione su un canale prefissato e periodicamente iniziano a scambiarsi pacchetti che contengono un messaggio che chiameremo "messaggio di hello". Il protocollo fornisce delle strutture dati nelle quali vengono memorizzate ulteriori informazioni sui pacchetti ricevuti oltre al "messaggio di hello". Tra queste informazioni aggiuntive troviamo la potenza del segnale, denominata RSSI (Received Signal Strength Indication). Ne consegue che avremo un nuovo RSSI (e quindi una nuova configurazione dei leds) contestualmente alla ricezione di un nuovo pacchetto.

### 4. MAPPATURA POTENZA SEGNALE WIRELESS SU ARRAY DI LEDS

Recuperato l'RSSI (ogni volta che viene ricevuto un nuovo pacchetto) occorre che tale valore venga rappresentato correttamente nell'array di 10 leds. L'RSSI sta su 8 bit, il che significa che può assumere valori da 0 a 255 in base 10. Per semplicità, ho optato per una mappatura linearmente proporzionale. Per limitarmi all'utilizzo di interi, ho approssimato il valore massimo di 255 a 250, ottenendo così delle soglie di accensione/spegnimento dei led, multiple di 25. In figura 4 è riportata una rappresentazione grafica di quanto appena detto.

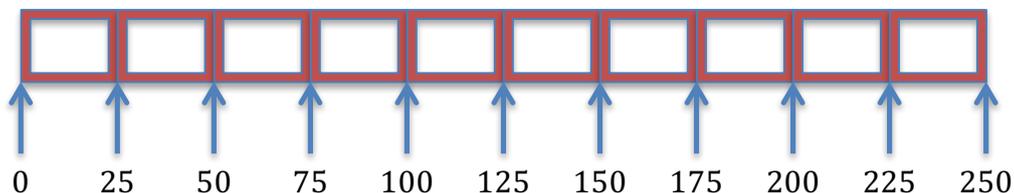


Figura 4

I leds, purtroppo, non sono collegati a un solo registro ma bensì a due registri; il che complica leggermente le cose. Più precisamente, i primi 8 leds sono collegati alla parte meno significativa del registro LATB, mentre i restanti 2 leds sono collegati ai bit 8 e 9 del registro LATC. Quindi, per accendere/spegnere i leds desiderati occorrerà scrivere dei valori opportuni su questi registri. In sostanza quello che l'algoritmo fa è leggere l'RSSI del

messaggio ricevuto, dividerlo per 25 ottenendo così il numero di led da accendere e conseguentemente il valore da inserire nei registri. L'accensione di un led aggiuntivo si ottiene assegnando al valore di partenza "val",  $val=(val*2)+1$  e scrivendo tale valore nel registro opportuno. Dunque, ottenendo il numero di leds da accendere, è sufficiente reiterare il precedente assegnamento per tale numero di volte, ottenendo così il valore da scrivere nel registro, avendo cura di inizializzare a zero "val" ogni volta che si presenta un nuovo RSSI. Avendo a che fare con due registri occorre fare attenzione ai casi in cui è richiesta l'accensione degli ultimi due leds (RSSI maggiore di 225): in questi casi, se viene richiesta l'accensione di un ulteriore led e i primi 8 sono già accesi, viene calcolato un secondo valore da inserire nel registro apposito che sarà pari a 0 (leds spenti), a 1 (primo led acceso, secondo led spento) o a 3 (leds accesi).

Con un'implementazione di questo tipo si ottiene un buon risultato ma comunque migliorabile. Infatti, ad esempio, il primo led viene acceso soltanto se l'RSSI è maggiore o uguale a 25. In realtà sarebbe più corretto che un led venisse acceso anche per RSSI inferiori a 25 e approssimabili a quest'ultimo. Più precisamente vorremmo che il primo led si accendesse per RSSI maggiori di 12, dato che il valore 13 è più vicino, e quindi approssimabile, a 25. Lo stesso discorso può essere applicato a tutti i multipli di 25. Per realizzare ciò è sufficiente comandare l'accensione di un led in più nel caso in cui il resto del rapporto tra RSSI e il valore 25 sia maggiore di 12.

## 5. FUNZIONAMENTO GLOBALE

Il software di controllo eseguito sulle due MiniFlex, a seguito delle dovute inizializzazioni, è costituito sostanzialmente da 3 tasks:

- TaskSend (frequenza 2 Hz): compone e invia periodicamente un "messaggio di hello" all'altro dispositivo.
- TaskMiWiOp (frequenza 20 Hz): nel dispositivo dotato di array di leds, verifica la presenza di nuovi messaggi, in caso positivo attiva il task "TaskMiWiOp", altrimenti incrementa un contatore che, raggiunto un certo valore (timeout) spegne gradualmente tutti i leds. Nel dispositivo senza array di leds se è presente un nuovo messaggio attiva il task "TaskMiWiOp".
- TaskMiWiOp: nel dispositivo dotato di array di leds, legge l'RSSI del messaggio appena ricevuto, calcola il numero di leds da accendere e i corrispondenti valori da inserire nei registri LATB e LATC. Nel dispositivo senza array di leds, scrive sulla porta UART i dati del pacchetto ricevuto (log).

In figura 5 è mostrato il dispositivo con i leds in funzione.



Figura 5

## 6. LOG

E' possibile visualizzare il log dei pacchetti ricevuti dalla MiniFlex senza array di leds, collegandola via USB al computer e utilizzando l'applicazione HTerm. Il valore Baud va impostato a 57600.

```
Broadcast with RSSI 96 from 1122334455667701
TxNum: 114
RxNum: 111
Received Packet:
Hello from Miniflex signal strenght indicator
Developed by Stefano Cicero
```

Figura 6

In figura 6 è riportato un esempio di log, dove troviamo: l'RSSI, l'id del dispositivo dal quale il pacchetto ricevuto è stato inviato, il numero di pacchetti trasmessi e ricevuti, e il messaggio contenuto nel pacchetto.

## 7. ALLEGATI

I sorgenti allegati alla relazione, per quanto riguarda i due dispositivi, sono dati dai seguenti files:

- **code.c:** file principale di progetto, contiene le operazioni di inizializzazione e definizione dei tasks

- **conf.oil:** file di configurazione dell'applicazione e del sistema operativo parte dello standard OSEK/VDX
- **ConfigApp.h:** file di configurazione dell'hardware
- **HardwareProfile.h:** definisce le funzioni utilizzate dall'hardware
- **SystemProfile.h:** definisce i moduli usati dal sistema