Internet of Things

Smart Bracelet

Edoardo Longo
841677
edoardo.longo@mail.polimi.it

Simone Montalto
841359
simone.montalto@mail.polimi.it

July 25th, 2016

Politecnico di Milano

Project1 - Smart Bracelet

Project Report

Introduzione

Il progetto preso in esame è *Smart Bracelet*, che consiste nella progettazione di una coppia di braccialetti per genitori/figli utili a monitorare la posizione dei figli oppure segnalare l'allontanamento del bambino o la sua caduta.

Per realizzare il progetto abbiamo utilizzato *TinyOS* come sistema operativo, *TOSSIM* come simulatore e *TOSSIM-Live* per la trasmissione del messaggio di errore sulla porta seriale con output leggibile da terminale.

Descrizione del programma

Ogni braccialetto possiede una chiave, che lo identifica univocamente, e può accoppiarsi con un solo braccialetto. Per questo motivo, ciascun dispositivo deve avere in memoria sia la sua chiave che la chiave dell'unico braccialetto con il quale deve accoppiarsi. A tale scopo in ogni dispositivo vengono precaricate due chiavi:

- `myKey[RANDOMKEYLENGHT]`: la chiave personale;
- `matchKey[RANDOMKEYLENGHT]` : la chiave del braccialetto con il quale si deve interfacciare.

La costante `RANDOMKEYLENGHT` va a definire la lunghezza della chiave.

All'accensione, ogni nodo invia ogni 5 secondi un messaggio in **BROADCAST** chiedendo l'accoppiamento con il terminale del quale conosce la chiave. Quando un braccialetto riceve il messaggio **BROADCAST** verifica se la chiave del terminale che ha inviato il messaggio corrisponde con la chiave, memorizzata in locale, del terminale con il quale accoppiarsi. In caso affermativo invia un messaggio in **UNICAST** confermando che è stato ricevuto il messaggio **BROADCAST** e che i due braccialetti possono accoppiarsi.

In fase di progettazione abbiamo deciso di realizzare tre `struct` diverse da utilizzare per l'invio dei messaggi tra i due terminali.

- `coupling_msg` da utilizzare per i messaggi da inviare in BROADCAST in fase di accoppiamento;
- `confirm_msg` per i messaggi da inviare in UNICAST per la conferma dell'accoppiamento;

- `info_msg` da utilizzare per i messaggi che il braccialetto del bambino invia periodicamente al braccialetto del genitore dopo che i due terminali sono stati accoppiati.

Al termine dell'accoppiamento, due Timer vengono settati:

- `Timer1`: impostato sul braccialetto del figlio che scatta ogni 10 secondi, invia con regolarità un messaggio informativo dal braccialetto del figlio al braccialetto del genitore, contenente informazioni sulla posizione e sulla *cinematica* del braccialetto;

- `Timer2`: impostato sul braccialetto del genitore con durata di 60 secondi, che avvisa il genitore nel caso in cui non vengono ricevute informazioni dal figlio. Ogni volta che il braccialetto del genitore riceve un messaggio informativo dal braccialetto del figlio quest'ultimo timer viene stoppato, resettato e fatto ripartire.

Quando il braccialetto del padre riceve un nuovo messaggio informativo da parte del braccialetto del figlio, le coordinate presenti nel messaggio vengono memorizzate in locale, di modo che, nel caso in cui il braccialetto figlio non fosse in grado di inviare un messaggio entro 60 secondi, sul braccialetto del genitore possono essere mostrate le ultime coordinate disponibili.

Ogni volta che il braccialetto del padre riceve un messaggio del figlio con stato `FALLING` oppure non riceve informazioni per 60 secondi dall'ultimo messaggio, un messaggio seriale viene inviato dal braccialetto ed una stringa viene stampata sul Terminale.

Per l'invio del messaggio seriale tramite *Tossim-LIVE* è stata realizzata una struct chiamata `test_serial_msg_t`.

Composizione del messaggio di info del figlio

Il messaggio informativo che il figlio invia al genitore ogni 60 secondi è composto nel seguente modo:

- `type`: il tipo del messaggio, in questo caso contiene il numero 3 che indica INFO;

- `pos_x`: un numero casuale per la coordinata x del figlio;

- `pos_y`: un numero casuale per la coordinata y del figlio;

- `state`: un terzo numero casuale compreso tra 1 e 10 viene generato per calcolare lo stato del braccialetto figlio. Per gestire le probabilità, è stato utilizzato un costrutto SWITCH-CASE:

- un numero compreso tra 1 e 3 genera lo stato `STANDING`, contrassegnato dal numero 11;
- un numero compreso tra 4 e 6 genera lo stato `WALKING`, contrassegnato dal numero 12;
- un numero compreso tra 7 e 9 genera lo stato `RUNNING`, contrassegnato dal numero 13;
- numero 10 genera lo stato `FALLING`, contrassegnato dal numero 14.

In questo modo si mantengono le percentuali di probabilità per ogni stato cinematico.

Funzionamento del programma

Per testare il funzionamento di *Smart Bracialet*, tramite il file di configurazione `run.py` vengono generati 4 nodi che al tempo 0 iniziano ad inviare messaggi di accoppiamento.

La configurazione della rete è la seguente:

- Nodo 1 (genitore) si accoppia con Nodo 2 (figlio);
- Nodo 3 (genitore) si accoppia con Nodo 4 (figlio).

L'intera durata della simulazione è nel `range (0, 5000)`. Al tempo 3000 il nodo 4 viene spento per simulare che uno dei nodi esca dal range utile per comunicare con il braccialetto con cui è accoppiato.

Contenuto dell'archivio

Nell'archivio sono presenti i file `smartBracelet.h`, `smartBraceletAppC.nc` e `smartBraceletC.nc` che compongono i programma *TinyOS* descritto precedentemente. Inoltre, per garantirne il corretto funzionamento sono stati creati i seguenti file:

- `README.txt` con indicazioni sui file modificati ai fini del funzionamento del progetto;
- Lo script *python* per lanciare la simulazione, `run.py`;
- `simulation.txt`: il file di LOG contenente l'andamento temporale dell'intera esecuzione;
- Le classi *Java* per il funzionamento dei messaggi seriali: `TestSerial.java` e `TestSerialMsg.java`;
- I file della topologia della rete e del noise, rispettivamente `topology.txt` e `meyer-heavy.txt`.