POLITECNICO MILANO 1863

IOT

SMART BRACELET

ANTLAB PROJECT

Massimiliano Paci - 852720
Leonardo Turchi - 853738

Description of the software for Smart Bracelet, based on TinyOS.

ASSUMPTIONS

We considered all the specifications written in the projectDescription file, except for the key-pairing process.

In particolar the key generation relies on a tiny algorithm that creates n pairs of random keys as follow:

Each bracelet has a couple of 20-char random key (keyP and keyC) .
- In a **child** bracelet keyP represents his parent's key and keyC represents his own key.
- In a **parent** bracelet keyP represents his own key and keyC represents his parent's key.

Two bracelet can be paired if keyP and keyC are the same for both.
By modifing the C_MAX value, this algotithm can create as many keys as required (in order to have n couples of bracelets working).

We assigned at **odd** nodes are parents, the **even** ones are children.
e.g. with C_MAX=3 we have the couples 1-2, 3-4, 5-6.

DATA STRUCTURES (DATAGRAMS)

We assume 3 types of datagram with a relative code:
BROADCAST (0), UNICAST (1), INFO (2).

The INFO datagram sent by the child, can have 4 different status:
STANDING (10), WALKING (20), RUNNING (30), FALLING (40).

The datagram for pairing (sent in broadcast) has the following structure:

type	key	address	identifier

The acknoledgment datagram (sent after pairing datagram reception) has the following structure:

type	acknowledgment

The information datagram has the following structure:

type	posX	posY	status	identifier

CONSTANTS

We designed our code making all the following constants to be scalable

- C_MAX : max number of bracelet supported couples *[set to 4 couples]*
- K_LEN : key length *[set to 20 byte]*
- T_1: pairing interval *[set to 15000 ms]*
- T_2: info transmission interval *[set to 10000 ms]*
- T_3: alert interval *[set to 60000 ms]*

ALARMS

Every alarm message has a format as follow:

!MISSING ALARM! received from child | Address: 2 | PosX: 43847 / PosY: 49971
!FALLING ALARM! received from child | Address: 4 | PosX: 42989 / PosY: 58694

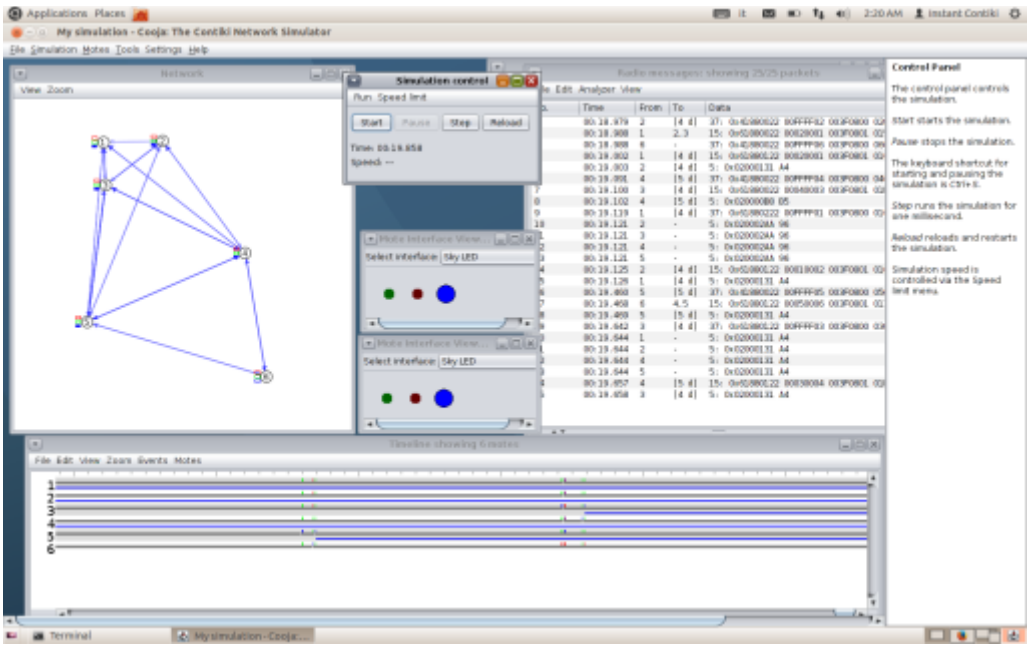
We implemented a LED notification mechanism that informs the owner of a bracelet on the system situation.

[led interface] 0 => RED : FALLING MESSAGE RECEIVED
[led interface] 1 => GREEN : INFO MESSAGE RECEIVED
[led interface] 2 => BLUE : BRACELET IS PAIRED WITH ITS ASSOCIATE

[led interface] 0 1 2 => R G B : MISSING ALARM RECEIVED
This could happen only in a parent's bracelet, to notify the effective missing status of a child.

PAIRING

Here we can see the pairing process in a *Cooja* simulation; in this phase all the messages are in broadcast, and every couple of bracelets is waiting for a correct partner message.
In the center we can see the LED status (BLUE + BLUE stand for *'PAIRED'*):



POLITECNICO
MILANO 1863

SIMULATION

By running the *python* file, the user can choose if

1. run and **display** the simulation on the terminal
2. run and **save** the simulation in a log file
3. run the simulation with a **serial** forwarder for *Node-Red*

LOG (from TOSSIM)

Every simulation with TOSSIM creates a file named **simulation.txt**

COOJA

After a compilation for the use with Cooja, it is possible to follow the evolution of the simulation from the PAIRING, to the MISSING alarm.

NODE RED

If is used the serial port forwarder, every message is sent via socket to Node Red. Node Red is configured to filter and show messages only of a specific type (alarms). The message, when received, is shown on the debug section of node-red web gui.

SOURCE CODE

All the file generated for the project can be found at the following github repository:

https://github.com/leontur/IoT-SmartBracelet

Are attached

- The source code
- The logs
- Some screenshots
- A simulation video