Smart Bracelet Report

Francesco Puoti : 10595640
Daniele Casciani : 10638165

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

ASSUMPTIONS

We considered all the specifications written in the projectDescription file. The random-keys are pre-loaded at the start from the SmartBracelet.h. Two bracelet can be paired if keyP and keyC are the same for both. Since the division in NescC rounds to the smaller integer number, we used the division by 2 in order to choose the right key for each couple (parent, child): the nodes with even id represent the parents, the nodes with odd id represent the children. Just two pre-loaded keys were added just to spawn two couples of motes, but adding more keys it is possible to add more couple of motes.

DATA STRUCTURES (DATAGRAMS)

We used two data structures:

- **sb_msg**:
  it is the data structure used to define the structure of a message sent by motes (as req/resp).
  Noteworthy are the fields data, which carries the pairing-key, and msg_type.
  The msg_type will be :
  - 0 for a pairing message
  - 1 for a paring-confirmation message
  - 2 for an INFO message
    Then the fields status, X and Y are filled with the value returned by a read of the fake sensor.
- **sensorStatus**:
  used to return the FakeSensor read, if completed. The status field can be (STANDING | WALKING | RUNNING | FALLING)

ALARMS

By requirements, we have to types of alarms:

- "ALARM: MISSING", thrown when the TimerMissing fires since no message has been exchanged among a couple of motes. The timer has an expiring time of MAX_MISSING_TIME = 60000[ms]
- "ALARM: FALLING!", sent by the child bracelet to the parent one when the fake sensor returns that state. The fake sensor is sensed every SENSING_TIME = 10000[ms]

Notice: The constants MAX_MISSING_TIME and SENSING_TIME are defined in the file SmartBracelet.h.

SIMULATION: TOSSIM LOGS NODE-RED

The TOSSIM simulation runs for 12000 steps (see the RunSimulation.py script), writing all the logs on the simulation.txt file. At the end of the simulation, using the pyserial library, the logs containing "ALARM" are sent to node-red. The messages, when received, are then accessible in the debug session of the web gui.

BRIEF DESCRIPTION OF MAIN EVENT AND PHASES: THE RECEIVE EVENT

This receive event manages the workflow of the motes interaction.
Starting from phase 0 (pairing phase), when a *broadcast* message containing the same pairing-key is received, the pairing phase leaves room to the confirmation one (phase 1). The motes involved in the pairing will send a confirmation message. When it is ack-ed, the coupling phase is ended, and the motes can start the required timers (TimerMissing for both the motes of a couple, and TimerSensing for the child mote. See the event `AMSend.sendDone`) and move to phase 2 where the child bracelets will start to periodically pull from the fake sensor and send info message to their parents bracelets.

INSTRUCTIONS

In order to properly run the simulation, execute the following steps in order:

- if you want to recompile our code, run `make micaz sim`
- check if the program `socat` is installed
- check if the library `pyserial` is installed in your python environment
- deploy the node-red export, using `sudo`, in order to set up the connections. It is important to deploy first the node-red server before of running the simulation.
- you can finally run `sudo python RunSimulation.py` in the src folder.

Notice that `sudo` is needed in order to work with serial sockets. It may happen to have some errors during socket configuration, just run the script a couple of times.