Stabile Pietro 10688876 a.y. 2021/2022

Internet of Things – Smart bracelets project

The aim of the project is to develop a software for a smart bracelet. The bracelet is worn by a child and the parent to send info on the position and status of the child. The bracelet can also sense if any dangers occur to child, such as falling, or if he/she goes too far.
The project has been developed working on Tinyos and simulating on TOSSIM, so the included files are the python script, the log file and the Tinyos codes (header, configuration, modules, and fake sensor code).

In ***SmartBra.h***, I define some constants values (the phases: pairing, operation an alert; the status: running, walking, standing and failing; the role: child or parent) and I indicate the structs for the three types of messages:

- *my_key_t* is the message sent at first in broadcast: it contains the sender and its key.
- *pair_msg_t* is sent back in unicast by a device that completes the pairing phase (it receives a broadcast message with a key equal to the personal one). It is sent to the paired mote. The message contains only the sender, so the receiving device gets the message and knows its coupled node.
- *info_msg_t* is the message periodically sent by a child to its parent. It contains an ID and the sender, to uniquely identify the message, X position, Y position and the state of the child, read from a fake sensor every 10 sec.

In ***SmartBraApp.nc*** I put all the components that I need: 4 timers, the fake sensor to get values of status and place, and ones for communication. Then, I wire the components and the interfaces.

In ***SmartBraC.nc*** I wrote the code that implements the operation of the project.
I declare some variables that will be mentioned and described and I put the keys into an array *vect*. The array *pairings* is to keep track of the pairing for each node. At first, it has 0 values, then for each element (element with index 0 refers to mote 1, index 1 to node 2, index 2 to node 3, index 3 to node 4) the 0 is replaced with TOS_NODE_ID of coupled bracelet.

I made three procedures. Each of them sends one type of 3 messages above described. The implementation is similar, but *sendKey()* is the one with Non-confirmable messages because when I turn on modes, I activate *TimerOn* and with a periodicity of 1 sec, every mode sends broadcast messages. If the coupled device does not receive a key message from its own, I wait 1 sec and the mode sends again the message to all. When a device finds its coupled one, *TimerOn* turns off (row 204) and its broadcast messages are over.
If a device, that does not share the same key does not receive a broadcast message, it is not an issue because it is assumed that for every parent there is only a child and vice versa.
Conversely, *sendInfo()* and *sendPair()* send Confirmable messages. If a message is not received, the procedure is called again (row 351 and 357).

When nodes are activated, I give them their role: child or parent and I turn on *Timer10s* with a periodicity of 10 sec and *TimerOn*, as described above.

Important is *event Receive.receive*. I distinguish the 3 cases, when a mote receives a key, pair or info message.
In the first case, I check if the key of the sender is the same of the receiver. If yes, the two devices are paired and I update *pairings*. I print a massage that informs two devices are paired, then I change for this node the phase in *OPERATION_MODE.* I also stop *TimerOn* and call *sendPair().*

In the second case, a mote receives a message with the information of a successful pairing.

In the last case, a parent receives a message containing all the informations about the child.
I save the last position in x and y, in case that the child goes too far. When an info message arrives, I stop the timer that counts 60 sec (respectively *Timer260s* and *Timer460s* for node 2 and 4) and I turn it on again. If it is the first time that a info message is sent, I only activate the timer, by the use of variable *TIMERGO*, initially equal to 0 and then to 1 after the first message. The timer allows to track if a parent does not receive an info message after 60 second the previous one. If a parent does not receive a message after 60 sec, phase is changed to *ALERT_MODE* and a message indicating the danger is printed. If the child transmits again, phase is back to *OPERATION_MODE.*
To simulate that a child goes to far, I had some troubles using *mote.turnoff()* in python so I used a variable *counter* for motes 3-4. When it reaches 10 starting from 0, the child 3 stop sending messages and parent 4 prints an alert message with the last position, after 60 sec, as reported in log file.
In the end, when a parent receives a message, it prints the position X,Y and the status. If the child is falling, an alert message is sent.

The position and status values are obtained from a fake sensor. As indicated in ***FakeSensorP.nc***, X and Y are randomly chosen in a range 0-100 and the status randomly chosen in an array of 10 elements containing 3 values corresponding to STANDING, 3 for RUNNING, 3 for WALKING and 1 for FALLING, with probabilities of 0.3, 0.3, 0.3 and 0.1.
These values are used by *sendInfo()* to send info messages.

Last part is dedicated to get to know if a message pair or info is received. To distinguish two cases, I used ID of info messages. When ID is 0, none of info messages have been sent so I am still sending pair messages. If bool value *ACK* is false, the message is not received, so I call again the procedures, in both cases.

RunSimulationScript.py contains the indication for the TOSSIM simulation. I create the communication channels, I create the 4 nodes all of them instantly at time 0 and I include the noise (-60 dB for every 12 cases). The range of events is 0-7900, corresponding to 3:15 minutes of simulation.

The log file represents all the events above described. After the pairing phase, the node 1 communicates to 2 and 3 to 4. Node 2 and 4 receive info messages every 10 seconds.

After 60 seconds the last message from node 3 at time 1:27, it appears the message *BE CAREFUL Your Child is gone* with the last position of 60 seconds before: 54 and 99.