AUTHORS: CARINI DAVIDE 10568649 ZANELLI RICCARDO 10572141

- **STRUCTURE OF THE MESSAGES**

To approach the problem, we have used 2 types of messages: pairing_msg and position_msg.

The **PAIRING_MSG** has the following structure:

msg_type	pairing_key[KEY_LENGTH]	node_id_source
This field represents the type of the message that can be 1 (request pairing) or 2 (response to the pairing). The request of the pairing is a BROADCAST message while the response of the message is a UNICAST message that it is sent when the node that receives the broadcast message has the same key.	The pairing key is an array of 20 chars that is in common between 2 devices. Every device has a pre-loaded key memorized locally.	It is the source node that sends the message.

The **POSITION_MSG** is used by the CHILD device to send its position to the PARENT device every 10 seconds and it has the following structure:

X	Y	Status[12]
It is the X coordinate of the CHILD.	It is the Y coordinate of the CHILD.	It is an array of char that represents the status of the CHILD device. As described in the project rules the status can be: STANDING, WALKING, RUNNING, FALLING. Each of these status is generated with a certain probability.

These 2 payloads are in the file SmartBracelet.h and they are well commented also there.

- **SmartBraceletAPPC.nc**

The ”SmartBraceletAppC.nc” file contains all the components used for the project and the wiring. The file contains the standard MainC and SmartBraceletC renamed as App, the components for the communications (AMSenderC, AMReceiverC, ActiveMessageC), the 3 TimerMillicC and the RandomC component to generate random values .The wiring is done as usual. We wired PacketAcknowledgements renamed as Ack to the ActiveMessageC.

The 3 millitimers are MilliTimerBroadcastPairing, MilliTimer10sec and MilliTimer60sec. The MilliTimerBroadcastPairing is started by every device after the power on. Every 400 ms each device sends in BROADCAST the pairing message: this mechanism is used to prevent the loss of

packets in the network. The MilliTimer10sec is used by the CHILD device to send the position every 10 seconds while the MilliTimer60sec is used by the PARENT device to raise a MISSING alarm when the CHILD device doesn't send any message within a minute.

- **GENERATION OF KEYS**

The generation of key works as follows: every time a device is booted, it generates a key in which every value of the array is (TOS_NODE_ID-1)/2. Thus, the first and the second node generate the key 0000000000000000000, the third and the fourth generate the key 11111111111111111, etc. We have adopted this method because there are only 2 couples of devices. Thus, if we had many other devices, we had to change this approach and implement it in another way.

- **EXECUTION FLOW**

When a node is switched on, it automatically starts the pairing phase with periodic sending of the request every 400 milliseconds. This is to prevent the paired node from not receiving the request if it is switched on later.

After 400 milliseconds, the node broadcasts the pairing request and the recipient checks that the key contained in the message is the same as its own key. If not, it ignores the request. If so, it sends a reply that functions as an acknowledgement with its ID inside. Now the node that had requested the pairing saves the ID of the paired device upon receiving the acknowledgement, changes phase (from pairing phase to operational phase) and according to its role starts a timer:

- Role: parent

  Starts a 60-second timer which, when triggered, signals that we have received no more messages from the child and the last position received.

- Role: child

  Starts a 10-second timer that sends your position and status to your parent when it is triggered.

Odd nodes are parent and even node are child.

The parent is notified, with the child's position, when the status received is fall alarm.

Response messages to a pairing and position request are implemented with ack enabled to retransmit in case they are lost.

- **Files in the Directory**

In the "simulation.py" file we have 4 nodes (2 couples of CHILD-PARENT devices) and we write the simulation log in the file called "simulation.txt". All the devices are power on at time 0 and at a certain point the CHILD of the first couple is switched off.

We added in the directory also meyer-heavy.txt, topology.txt and the Makefile. We didn't modify the meyer-heavy.txt while we have inserted the 4 devices in the topology.txt and all the links between all the pairs.

- **Optional Part**

We failed to import the library for the serial in python. The error was “No module named Serial” and we have tried in every way to install the pyserial in the directory where is installed Python failing miserably.