IoT FINAL PROJECT

Keep your distance

LUCA COLOMBO - 10530695

General overview.

The Keep your distance project is composed of three main parts: TinyOS that is used for managing the motes and that contains the logic in order to trigger the alarms, Cooja that is used for the simulation of the scenario and in order to forward the messages via socket and Node-Red that receives the messages and through the IFTTT service sends notifications to the mobile phone.
For this project I supposed that 5 motes are present.

TinyOS.

I started from the TinyOS part where first of all I defined the structure of the broadcast messages sent by all the motes. Here, I decided to also include a message counter in addition to the sender ID in order to manage the check on the consecutiveness of messages received. Then, in the configuration file I have identified the interfaces that I had to use in order to accomplish the task. They are principally the ones in order to send, receive and modify messages and the timer for the motes in order to broadcast their presence.
At this point I wrote the logic of the program in the fooC.nc file. I defined the principal global variables:

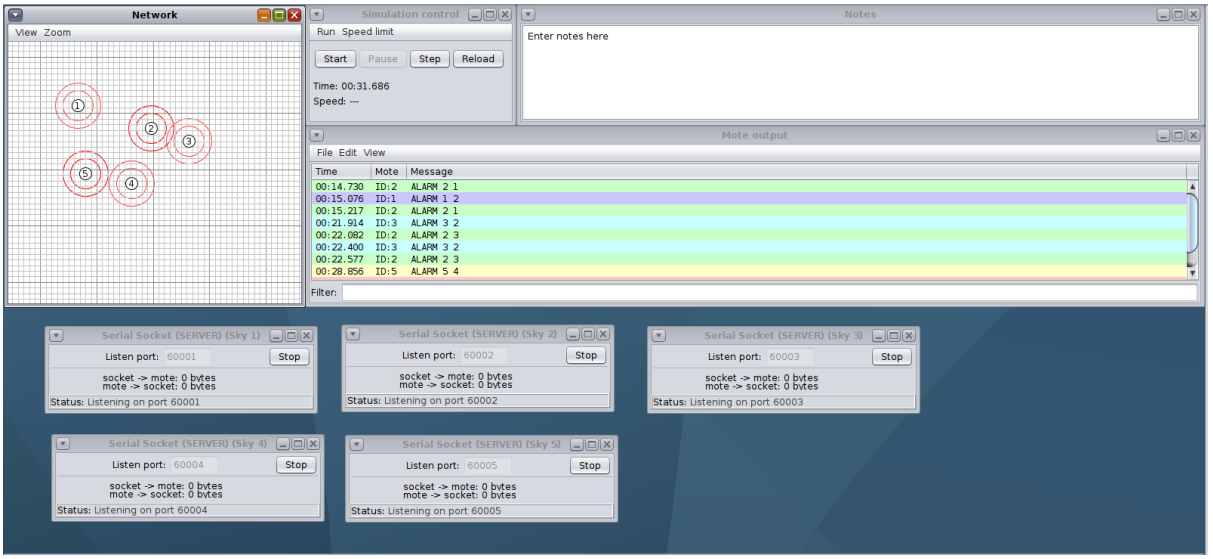
1. A message counter for each mote that is incremented every time a new message is sent.
2. An array of counters for each mote where in the i-th position of the array there is the counter of consecutive messages received from the i-th mote.
3. An array of the last message counters received for each one of the motes where in the i-th position of the array is stored the message counter of the last received message from the i-th mote.

Once I defined the control variables, I wrote the configuration of the packet that each mote needs to send when the timer fires. In the message each mote inserts its ID and its message counter and sends in a broadcast way the packet.

For what concerns the receiving of messages, each mote checks if the message counter inside the packet is consecutive to the one previously received from the same mote (that is stored in the lastMessageCounters array) or if it is the first message received by that mote. If it is so, it increments the counter related to that specific sender mote and if the value is >= 10, it raises an alarm containing the IDs of the 2 motes.
In case the message counter is not consecutive to the one previously received from the same mote, it means that the sender mote has moved away for a period of time and then returned to the range of the current mote and so the counter and the lastMessageCounter are reset to 0.

Cooja.

Inside Cooja I created the 5 sky motes and started the simulation. In order to create the log file that explains what happens during the simulation I added some additional comments that I have removed later. In the final version of the project, only the alarm messages are printed with the number of the 2 neighboring motes.
Finally, I added a server socket for each mote in order to forward the alarm messages to Node-Red.



Node-Red.

Finally inside Node-Red I created the flow for receiving the forwarded messages through the TCP input node, for splitting the message through a regexp in order to retrieve the number of the 2 neighboring motes and for triggering the IFTTT service in order to send a telegram notification containing the IDs of the 2 motes to the mobile phone.