PROJECT - IoT

By Stefano Agresti, Gianluca Guarro

PERSONAL INFO

***Name**: Stefano Agresti*

***Person code**: 10685203*

***Matricola**: 913079*

-

***Name**: Gianluca Guarro*

***Person code**: 10658177*

***Matricola**: 918696*

-

***Link to simulation:** https://www.youtube.com/watch?v=Cu4VAJeTB1Q*

***Github repository**: https://github.com/steflyx/IoT-Project*

ASSIGNMENT

You are requested to design and implement a software prototype for a social distancing application using TinyOS and Node-Red and test it with Cooja. The application is meant to understand and to alert you when two people (motes) are close to each other.

DEVELOPMENT

A working simulation of our project can be found on Youtube at the link provided above.

1. **Tiny-OS**

There's not much to say about this part. Most of the code is similar to the one used in the previous activities, the only main difference being the logic of the *Receive* function.
We decided to create an array which stores up to ten distinct motes and then just deletes the oldest entries when space for new ones is needed. We felt this was the best solution for the project, because it won't fill your phone with notification just to tell you you're standing next to the same person (as it might happen if no mote was stored), but it will alert you if you're too close to the same person in different moments of the day (which wouldn't happen if the number of stored motes was unlimited or very big).

> *Example 1*: Elia and Gloria live together, so they can't practice social distancing between each other. The app will alert them that they're too close only from time to time (that is, only after they've met at least ten other people).

Example 2: Andrea and Sabrina work in the same building and they sometimes take the elevator together. Since they don't meet very often, their motes will have enough time to get in touch with ten new motes between each of their meeting, thus giving the app the opportunity to alert them every time they take the elevator together.

Of course this is just our interpretation of the problem, not necessarily the best one.
In a real system the maximum number of motes to be stored should be evaluated empirically and maybe a mote that is encountered really often shouldn't cause alert anymore, but these are just ideas to improve the system beyond the scope of the project.

2. Node-Red

The Node-Red part wasn't particularly different from the previous activities. Its structure can be seen in *Figure 1* and includes the following nodes:

- *TCP* node: to coonect to the motes in the simulation.
- *Function* node #1: cleans the message coming from Cooja (to eliminate bad characters).
- *Function* node #2: prepares a request to be sent to *IFTTT*, containing the id of the mote that was encountered.
- *HTTP request* node: sends a *POST* request to *IFTTT* which in turn triggers a notification to be sent to the user.

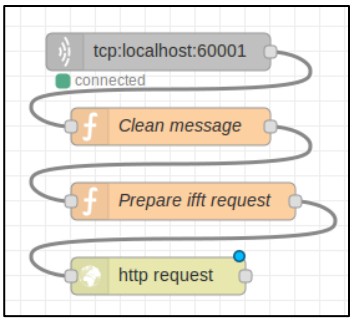

Figure 1: Node-Red structure

To send a request to *IFTTT* (once a service is set up), it's enough to add an *event* field to the *HTTP* request, a *key* field containing the *API-KEY* associated to the applet and send the message payload in format *{value1:x, value2:y…}*.

3. Cooja simulation

Again, nothing new. After compiling the *Tiny-OS* motes, just add as many sky motes as desired and then start the simulation. In the debug page, the messages sent to *Node-Red* will be visible.
To connect to *Node-Red*, open a *Serial Socket (SERVER)* on the mote you want to test and specify the same port in the *TCP* node in *Node-Red*. Examples are shown in *Figure 2* and *3*.

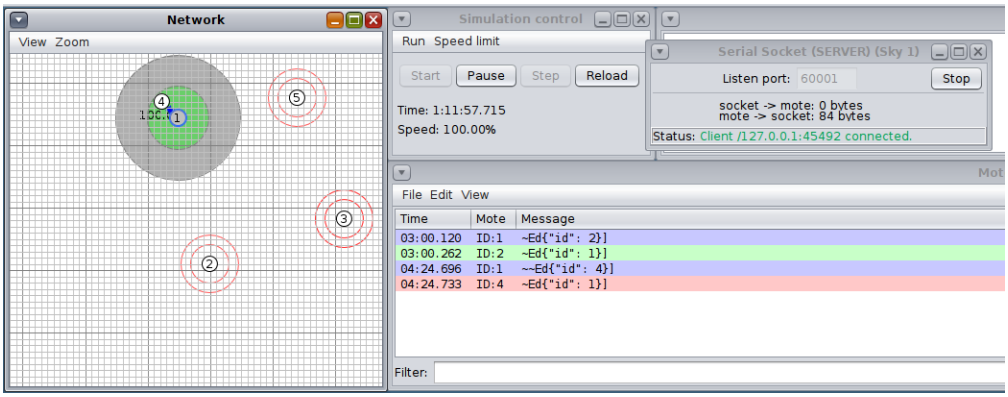

Figure 2: Mote #1 is first moved next to mote #2 and then next to mote #4

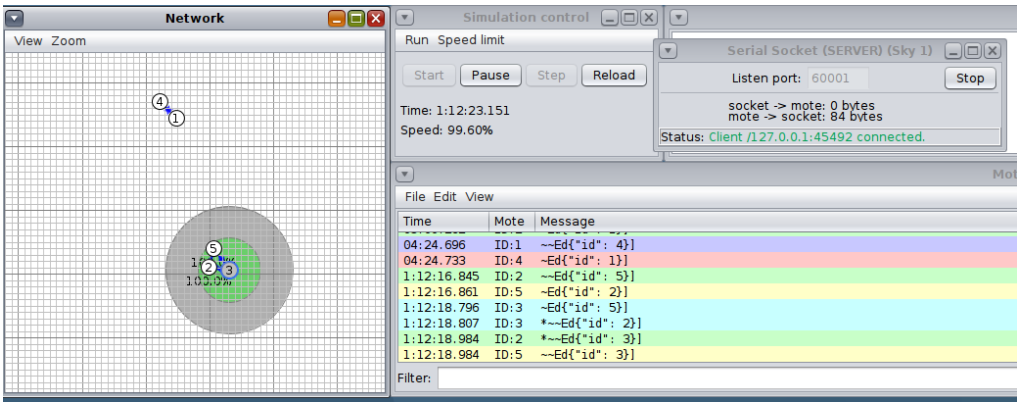

Figure 3: Motes #2, #3 and #5 are all placed next to each other

4. IFTTT

The last part to complete is the *If this than that* section. The website is quite intuitive and allows to create an applet without great effort. In *Figure 4* is the page of the applet we built for this project. The applet is triggered when a "*mote_encountered*" event happens (to notify that is *Node-Red*), sending an email to the specified email address.

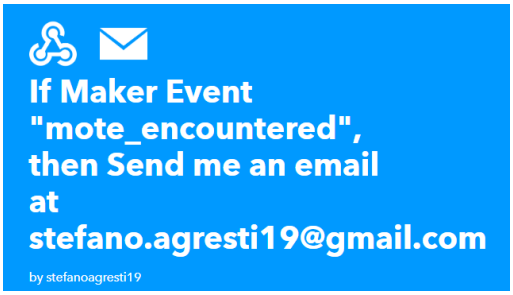

Figure 4: Specification of the applet

The email can be customized and can contain information about the event when these are sent by the source. Examples from our simulation can be seen in *Figure 5* and *6*.

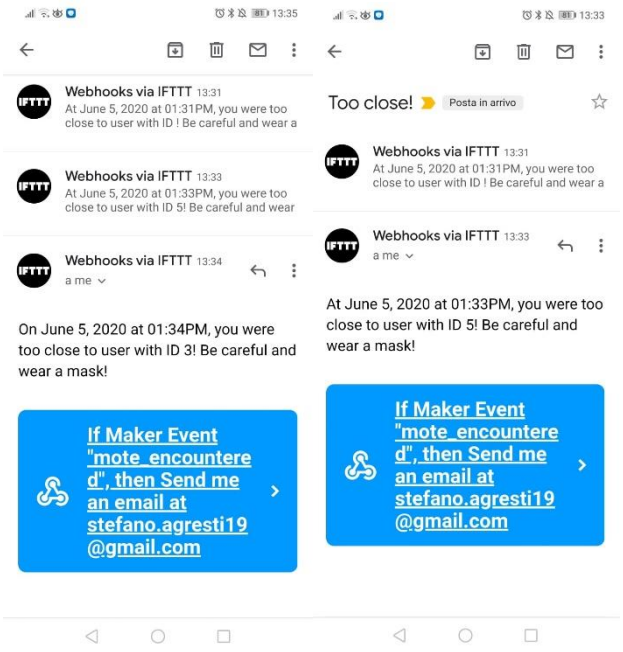

Figure 5 and 6: Examples from the simulation of the applet

5. Small update

After finishing the project, we realized that we might have misinterpreted the assignment, as we assumed that the "you" in "alerts you when two motes are too close" was referred to the user whose mote was too close to another one.

We realized that a more straightforward interpretation would be that that “you” is a third user who’s controlling all the motes, so we slighlty adjusted the code to adapt to this second interpretation:

- In *TinyOS* we print both *sender id* and *node id*, but only when the latter is bigger (to avoid double messages).
- In *Node-Red* we added four *TCP* nodes, all connected to the same *function* node, but different port, and we now send two values instead of one.
- In *Cooja* we opened a socket for each mote, with increasing numbers (from 60001 to 60005).
- In *IFTT* we slightly changed the message.

The results can be seen in *Figure 7* to *9* and we included these files as well in our delivery.

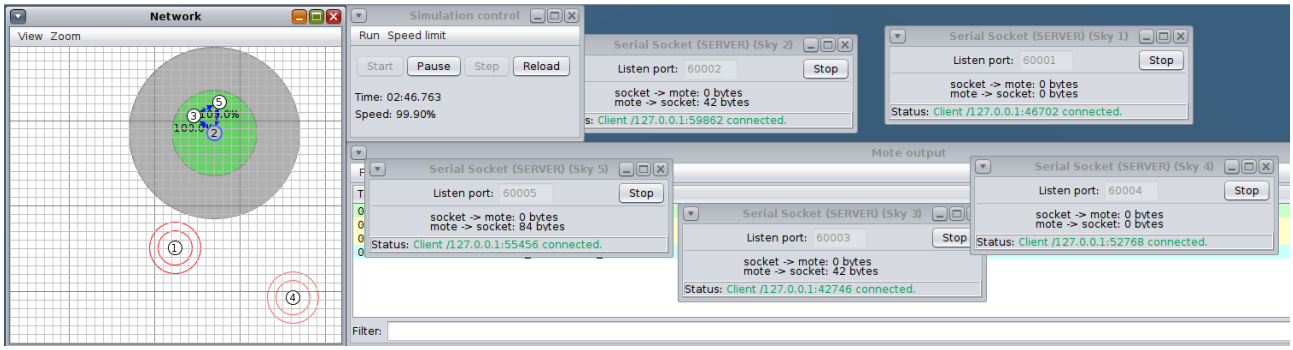

Figure 6: The new Cooja simulation, with 5 open sockets

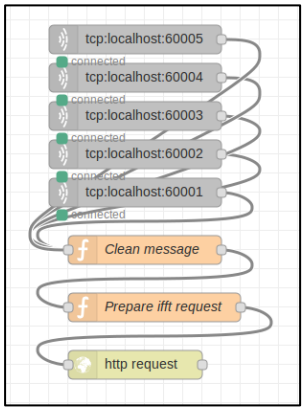

Figure 7: The new Node-Red structure

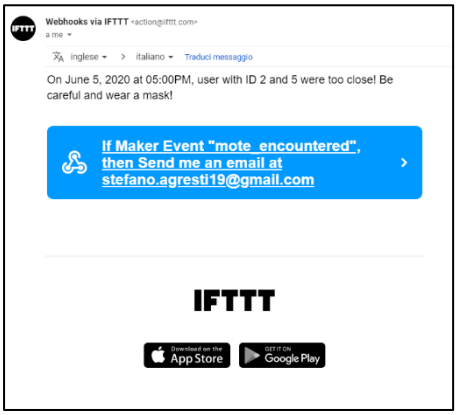

Figure 9: The new message sent by IFTTT