POLITECNICO
MILANO 1863

Internet of Things - Final Project (Project 1)

Politecnico di Milano AA 2019/2020
Computer Science and Engineering

Matteo Falconi
Codice Persona: 10568259
Matricola: 945222

Davide Fiozzi
Codice Persona: 10535048
Matricola: 945107

GitHub Repository:
https://github.com/TheFalco/IoT_Project

TinyOS:

We implemented a TinyOS application in which each (Sky) mote broadcasts its presence every 500ms with a message containing its ID number to the other motes. If a message is received, two motes are near, and a printf with the corresponding ID of the two motes must be sent through an open server port in order to be received by Node-Red. The server ports must be opened in the Cooja simulation as seen in class (Mote 1: port 60001, Mote 2: port 60002…).

Below are provided the main details about our TinyOS implementation of the project.

File Project1C.nc:
We introduced an array to store the last 50 (*Mem_Length*) received IDs of the other motes:

```
uint8_t received_id[Mem_Length];
```

If the motes receives more than 50 messages, the array is left-shifted by one position, in order to maintain in memory the last 50 IDs. In order to not flood the array with the same ID, an ID is added to the array only if it is different from its predecessor in the array.

File Project1AppC.nc:
As we have seen in class, we added a `PrintfC` component to implement the communication with node-red. The other components are needed in order to implement basic functionalities.

File Project1.h:
A new struct is needed to implement a message that contains its ID:

```
typedef nx_struct radio_msg {
     nx_uint16_t id;
} radio_msg_t;
```

Makefile:
The Makefile shall contain the reference to the printf library

```
COMPONENT=Project1AppC
CFLAGS += -I$(TOSDIR)/lib/printf
include $(MAKERULES)
```

See Source code comments for a detailed description.

Node-Red:

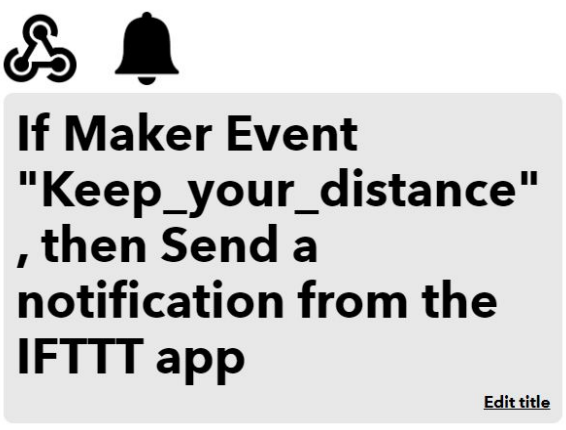
Message flow in Node-Red:
tcp-in → setPort() → clearInput() → setTime → checkLastID() → setEvent() → web request (IFTTT)
Our Node-Red implementation supports up to 5 different motes, but the flow its highly scalable.

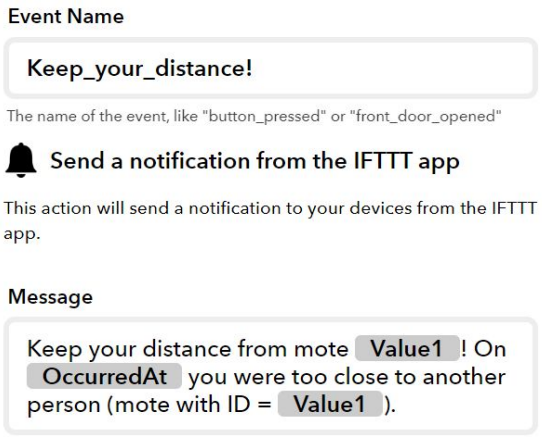
TCP-in:
Receives message from mote X through port 6000X, set in the Cooja simulation
setPort():
Sets port and user of the received message
clearInput():
Clears the input, deleting useless characters introduced by printf function of TinyOS and discharges malformed messages.
setTime:
Sets msg.date to the current time. This information is useful in order to implement a timer for every notification: a new notification between two given motes will be sent only if at least 10 minutes have passed.
checkLastID():
Implements the logic described above. If the current user has already met the X user in the last 10 minutes, a notification should not be sent to its *IFTTT* application.
setEvent():
Sets the event that has to be notified to the *IFTTT* application, including the value of the mote/user that is too near.
Web Request (IFTTT):
Sends the web request to *IFTTT* server, in order to implement push notifications on the user's phone. In this module, there must be specified the API Key of the user that has to be notified.

Note: motes 3, 4 and 5 don't have a valid URL in the Node-Red flow (blank URL) because we have done a simulation using only two real devices with *IFTTT* installed (and the corresponding API Key). The 3 users can be easily activated using 3 new different *IFTTT* accounts.

IFTTT Details:

Every user, in order to receive a notification on its phone, must install *IFTTT* and configure a new applet. We followed the guide at: https://wiki.instar.com/Advanced_User/Node-RED_and_IFTTT





LOG Details:

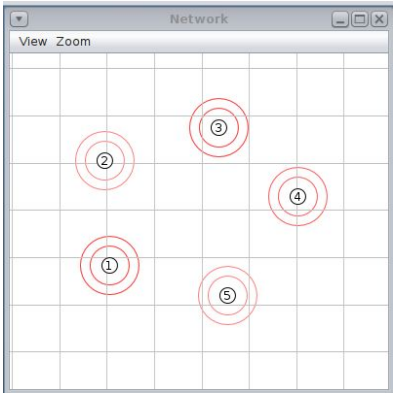
TinyOS:
We used dbg function in order to populate a log. The debug will be visible only compiling and using a TOSSIM simulation.

Node-Red:
During a Cooja simulation a log.txt file will be populated with all the actions executed on Node-Red.

If this message "*msgID: fc7eca4e.228938, message received on port 60002 for user 2."* is not followed by "*msgID: fc7eca4e.228938, message for user 2 cleared: user is too near from user* X" it means that the received message is invalid or malformed.

If this message "*msgID: fc7eca4e.228938, sending notification to user* X*..."* is not followed by "*Congratulations! You've fired the Keep_your_distance%21 event (msgID: fc7eca4e.228938, notification sent to the user* X*)*" it means that for the user *X* wasn't specified a valid *IFTTT* URL inside the web request node.

In order to provide a sample log.txt, we have run a simple Cooja simulation: the initial configuration of the motes is showed in the following picture.



We have simulated the mote 1 moving near all the other motes, following this order: 2 -> 3 -> 4 -> 5 -> 2. The resulting log.txt of this test is provided in the .zip file of the project.

Result of the sample:

The following picture provides an example of the notification that a user receives through *IFTTT* when he is too close to another user.