IoT Project: Keep your distance 2020/21

Prof. Cesana Matteo

Shalby Hazem Hesham Yousef (Personal Code: 10596243)

POLITECNICO
MILANO 1863

1 Structure description

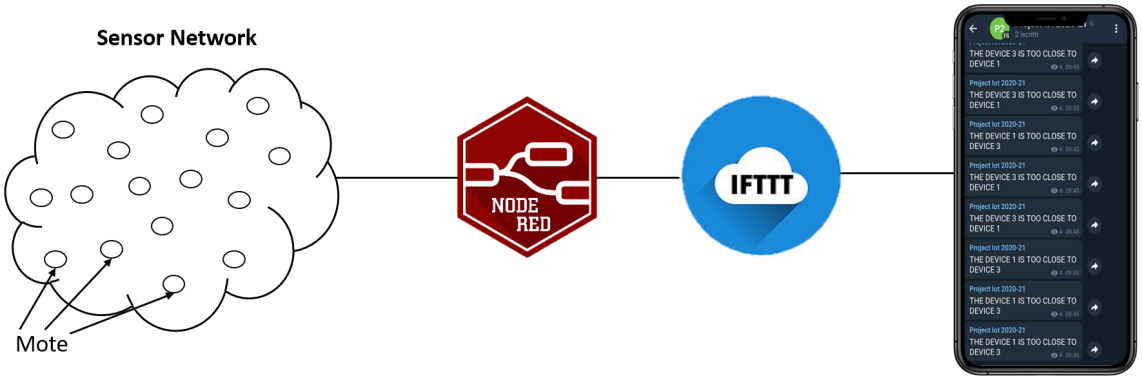


Figure 1: General project schema

The software purpose is to detect and alert the user when two people (motes) are close to each other.[1]. The software structure, as in the Figure 1, could be split in three main parts:

- The sensor network
- Node red
- IFTTT[2]

In the sensor network each mote broadcasts its presence every 500ms with a message containing its ID and a counter, which is incremented every time a message is sent. If a mote receives 10 consecutive messages from the same mote it triggers an alarm.
The alarm triggered by a mote is captured by node red via socket and then a web req to the IFTTT service is done.
The last component of the software (IFTTT) simply sends a message to the user using a telegram channel.

1.1 THE SENSOR NETWORK

Per se the motes code is very simple except for the part that triggers the alarm when 10 consecutive messages are received. To recognize/sense the event described before the developed code uses two arrays:

- `last_msg_count`
- `cons_msg_count`

The first array (`last_msg_count`) saves the last message count received from the other motes and it's updated each time a new message is received, while the second one (`cons_msg_count`) takes trace of the number of consecutive meshttps://www.overleaf.com/project/60d2d38e78167f6bd7f3d3fcsage received and it's managed using the following rule.

[1] two people/motes are close to each other when they remain close for at least 5 seconds (10 message)
[2] IFTTT is a service to create conditional statements, and it's an easy way to send notifications to your smartphone

$$\begin{cases} \texttt{cons\_msg\_count[rcm->id] = 1} & \texttt{if last\_msg\_count[rcm->id] +1 < rcm->counter} \\ \texttt{cons\_msg\_count[rcm->id]+= 1} & \texttt{if last\_msg\_count[rcm->id] +1 == rcm->counter} \\ \texttt{cons\_msg\_count[rcm->id] = 0} & \texttt{if cons\_msg\_count[rcm->id]==10} \end{cases}$$

Note that:

- In the previews rule `rcm` is the received message.
- If the last rule is rule is verified an alarm is triggered and the counter of the consecutive messages of the corresponding motes is reset.
- Both the array keep the information about each mote in the position corresponding to the mote ID (for example: in the mote with ID.2 arrays, information about mote 3 are saved in the pos 3 of the two array)[3]

Another fact that is noteworthy is how the mote trigger an alarm. it's done by simply printing a JSON string containing the information about the two motes that produced the alarm. The following is an example of an alarm triggered because mote 3 and mote 4 were too close to each other.

```
{
  "ALARM": {
    "id_first_mote": 3,
    "id_second_mote": 4
  }
}
```

1.2 THE NODE RED

The node red part simply sends an http request, once it captures an alarm.
The node-red part could be divided into three parts:

- Reception of messages: done via socket, one foreach mote.
- Parsing of messages: since motes send JSON strings as described before, this part of the code simply parse the received strings.
- Sending results: sending an HTTP request to the IFTTT service.

Noteworthy is the fact that, if two motes are too close to each other, two alarms are triggered but since the user in the simulation is only one, node red drops one of the two alarm (the one triggered by the mote with smallest id). The last rule is added only to enable a better understanding of the simulation part.

1.3 IFTTT

Once an HTTP request is received simply send a message to a public telegram channel [4]

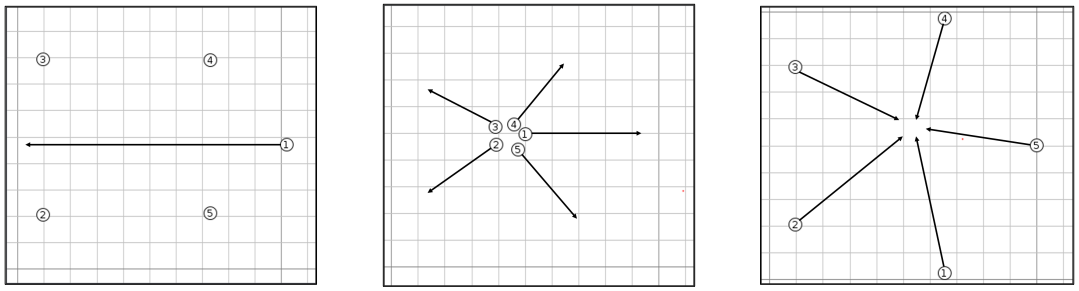
[3] The algorithm is described with arrays with index that starts from 1 but in the implementation the arrays starts from 0

[4] the link to join the telegram channel is the following: Telegram Channel

2 SIMULATION

The simulation is done using 5 motes but could be extended to N mote, where N is a parameter in the code running on the node (by default is set to 10).
The simulation was used to test the behavior of the system under a lot of circumstances. The simplest simulation is done by approaching and moving away two motes from each other to see their behavior[5].
Then a lot of simulation could be done extending the approach from two mote to multiple motes, and in particular four main simulation are noteworthy:

- A mote crossing an aisle created by the other motes
- Motes that are all collapsed in one point moving away
- Motes not close to each other that are going to the same point
- A mote going from one side to the other of a room, crossing other nodes in its path (this similar to the first one but the other motes are sparsed)



(a) Crossing aisle (b) Moving away from center (c) collapsing in one point

Figure 2: Simulations representation.

3 Warnings

Since IFTTT impose a limit on the triggerable events per minute[6], in the node red part a rate limit is necessary. to better appreciate the simulation results a speed limit in the Cooja simulator is recommended

[5] A self-explanatory log file showing this configuration is provided and is the only one useful to show that the project works

[6] IFTTT limit: 120 events/minute