Politecnico di Milano

Dipartimento Elettronica, Informazione e Bioingegneria

Final IoT Project

Keep Your Distance

Author:
Francesco Monti
Matr: 919755

Supervisor:
Dr. Edoardo Longo
Dr. Matteo Cesana

May 29, 2020

POLITECNICO
MILANO 1863

Abstract

This document contains the documentation for the final project of the course "Internet of Things", Academic Year 2019/2020.
This document has been also uploaded on the following GitHub repository: `https://github.com/Framonti/IoT_Projects`

0.1 TinyOS App

The logic of the TinyOS app is indeed very simple: it opens the radio and pings its presence (broadcasting its Mote ID) every half second. If it receives a message, it internally saves the ID of the sender by logging it and then it sends an alarm through the collaboration of Cooja, Node-RED and ITTT.

In order to saves the messages, we declared a part of the flash memory of the device as reserved for logging. It would be possible to easily extend our application to read the logs (e.g. after a timeout, or if the memory is full), preprocess them (e.g. counting how many times the same device was logged) and finally send the data to a gateway. We remember that local preprocessing in IoT is usually recommended for reducing the size of messages.

0.2 Simulation

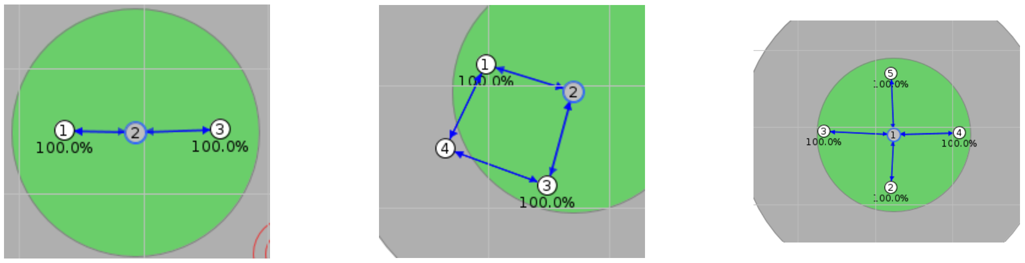


Figure 2: Topology examples: From left to right, linear, ring, star topology

We simulate our application using both TOSSIM and Cooja. We tried six different topologies using up to six motes (e.g. 2). The simulation results and the topologies can be found in the folder *"Sim_Results"*. We point out a couple of important details:

- We can build the app for both telosb and micaz devices (*make telosb* and *make micaz*), but TOSSIM doesn't work with the Loggers nor the printf, because it's not able to simulate the flash memory nor the printf function. In order to obtain the simulations, we commented away the related code. A dbg statement replace the printf functionality, so it's still possible to see and validate the message exchanges.

- Another problem arose while simulating (*python2 RunSimulationScript.py*), probably due to the small RAM available to the VM. We had to simulate the topologies one by one (by modifying line 52 of the Python script).

- We split the simulations in different files, depending on the topology used. A *"readme.txt"* file in the *"Topologies Used"* folder describes the topologies.

- The Cooja simulation contains only the printf statements; due to some compatibility issues the messages contains some strange characters, which we removed in Node-RED before firing ITTT requests.

0.3 Node-RED and ITTT

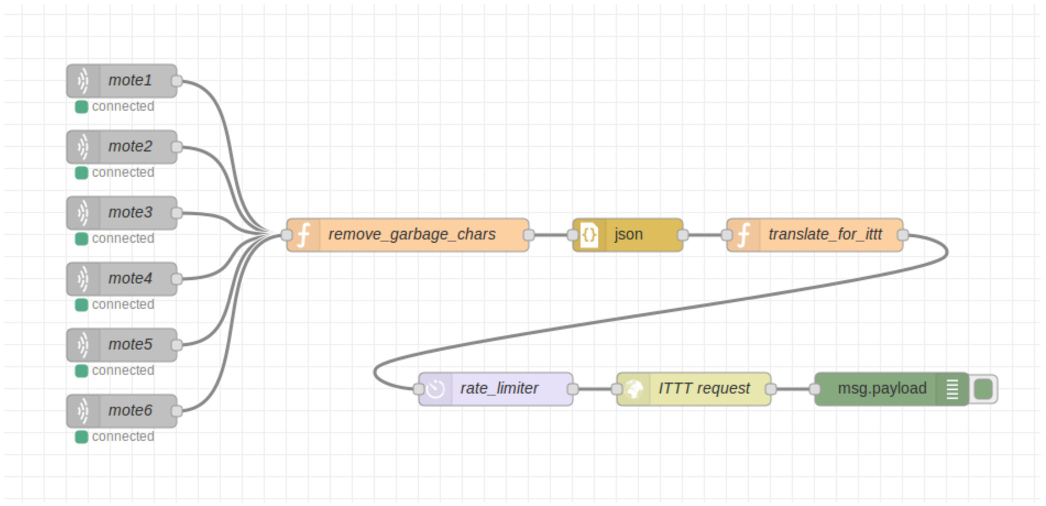
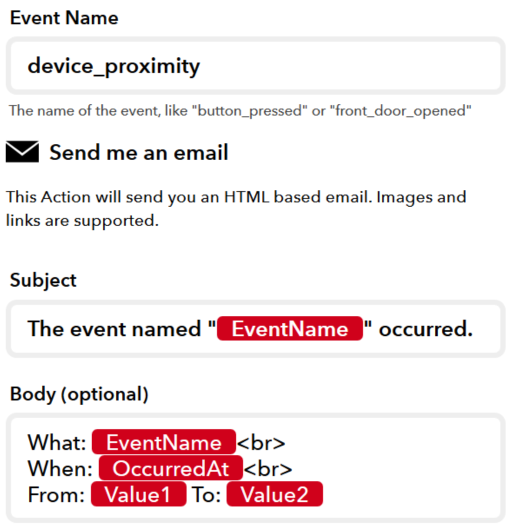


Figure 3: Node-RED Design

We finally connects the messages sent by the motes to Node-RED through Cooja. Figure 3 shows the design. We can see that each mote is connected thanks to a TCP socket, then all the messages are preprocessed, transformed into a JSON object and sent to ITTT using a HTTP POST request.

In ITTT we defined a simple rule, that fires an e-mail in order to simulate the alarm of a possible real application (4).



Event Name

device_proximity

The name of the event, like "button_pressed" or "front_door_opened"

Send me an email

This Action will send you an HTML based email. Images and links are supported.

Subject

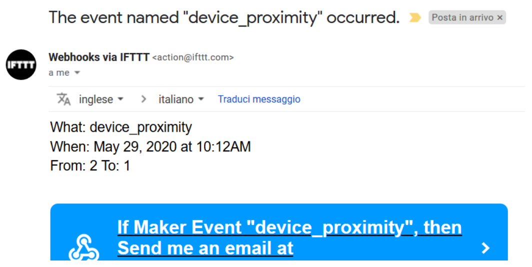
The event named " EventName " occurred.

Body (optional)

What: EventName
When: OccurredAt
From: Value1 To: Value2

Figure 4: ITTT Event

By simulating our application in Cooja, we received some e-mails (e.g. 5). We can see that ITTT has created an e-mail for the event "device_proximity", including the Timestamp of the interaction, the Sender and the Receiver (identified with their Mote ID).



The event named "device_proximity" occurred.

Posta in arrivo ×
Webhooks via IFTTT <action@ifttt.com>
a me
inglese > italiano Traduci messaggio

What: device_proximity
When: May 29, 2020 at 10:12AM
From: 2 To: 1

If Maker Event "device_proximity", then Send me an email at

Figure 5: Mail Received by ITTT