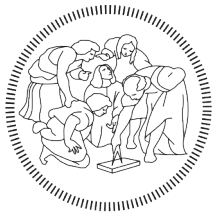


POLITECNICO
MILANO 1863

IoT Project

Lightweight publish-subscribe application protocol

Giovanni Paolino 10696774
Ilaria Paratici 10707097

Academic Year
2022/2023

Contents

1 Introduction

Among the 3 projects proposed during the IoT course of this year, we decided to implement the first one.

The project requires to implement an MQTT like application with a PAN-Coordinator and 8 nodes.

In this application, nodes can connect to the Coordinator; when connected, they can subscribe to topics among the 3 existing ones: temperature (topic 0), humidity (topic 1), luminosity (topic 2).

Nodes can also publish values on topics.
When the PAN Coordinator receives a publish, it sends it to a ThingSpeak public channel through Node-Red. On the ThingSpeak public channel, the charts of the 3 topics are plotted.

2 TinyOS Implementation with TOSSIM

2.1 Topology

Nodes are numerated from 0 to 8.
Node 0 is the PAN Coordinator; nodes 1-8 are the 8 nodes connected to the Coordinator in a star topology.

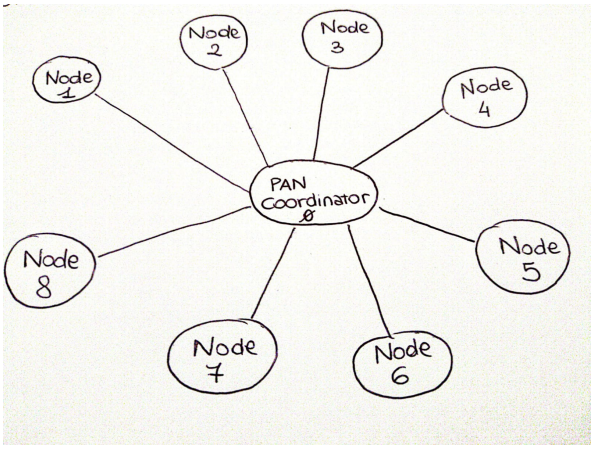
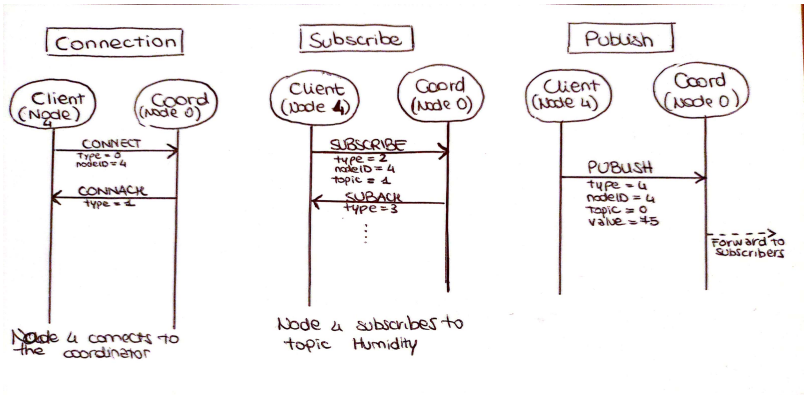


Figure 1: Topology

2.2 Message Format

The messages exchanged by the nodes have 4 fields: type, nodeID, topic, value. There are 6 types of messages in our application:

- **Type 0**: Connect message
- **Type 1**: Connack
- **Type 2**: Subscribe
- **Type 3**: Suback
- **Type 4**: Publish
- **Type 5**: endSub, it's sent by the Coordinator to the nodes and it means that first phases related to the connections and the subscriptions are ended; when this message is sent, nodes can start the Publish phase



The nodeID field contains the ID of the sender of the message.
The topic field, set and relevant only in subscribe and publish messages, indicates the topic the node wants to subscribe to / publish on.
The value field, set and relevant only in publish messages, indicates the value the node wants to publish on the specified topic.

2.3 Phases

The project runs in 4 subsequent phases in the following order:

- **Booting Phase**: Starting of the application
- **Connection Phase**: Each node connects to the PAN Coordinator
- **Subscribe Phase**: Nodes subscribe to topics
- **Publish Phase**: Nodes publish values on topics

At the end of the Subscription phase, an endSub message, sent by the Coordinator to the nodes, informs the nodes that they can begin the Publish phase, because all the subscriptions have been already done. From that moment on, the messages sent will be only publish messages.

2.3.1 Booting Phase

When booting, the Coordinator lets immediately start Timer0 by calling the function startOneShot(0).
Then, each node starts Timer0 in a different moment (node 1 starts the timer after 1s, node 2 starts it after 2s etc.)

2.3.2 Connection Phase

When Timer0 fires, the nodes send the CONNECT message to the Coordinator, which answers them with a CONNACK message.
At the reception of the CONNACK message, the nodes start the subscription phase.

2.3.3 Subscription Phase

When nodes receive the CONNACK by the Coordinator, they can start to send the SUBSCRIBE messages.
The value of the publish messages is always chosen at random between 0 and 100.
In order to test the implementation, we have chosen to assign a priori the topics to which nodes 1, 3 and 6 have to subscribe.

Node 1 subscribes to topics 0 and 1 (Temperature and Humidity).
Node 3 subscribes to topics 1 and 2 (Humidity and Luminosity).
Node 6 subscribes to topics 0,1 and 2 (Temperature, Humidity and Luminosity).

They subscribe to the first topic after the connack of the Coordinator and then to the second, and eventually the third topic, when they receive from the Coordinator the suback of the previous subscription.

The other nodes subscribe to a topic chosen at random with a value chosen randomly between 0 and 100.

The Coordinator, at the reception of a SUBSCRIBE message, checks the topic of the subscription and updates to 1 the element of the array SubToTopic# corresponding to the node which has sent the subscription (eg. if node 1 subscribes to topic 2, the Coordinator will update SubToTopic2[0] = 1; where 0 = #node –1 because nodes go from 1 to 8 while the elements of the arrays of subscriptions go from 0 to 7).

Then the Coordinator sends the SUBACK message to the node that has sent the subscription and finally checks if the subscription phase is ended (i.e. when all the nodes have subscribed at at least one topic and nodes 1, 3 and 6 are subscribed to the topics decided a priori).
If the subscription phase is ended, the Coordinator sends in broadcast to all the nodes a endSub message (message with type = 5).

When a node receives the endSub message from the Coordinator, it starts the publish phase.

2.3.4 Publish Phase

When a node receives the endSub message from the Coordinator, it calls the function Timer1.startPeriodicAt(—, 9000) to let Timer1 start periodically with a period of 9s; each node lets start the timer for the first time at a different time instant (Node 1 calls Timer1.startPeriodicAt(1000, 9000), Node2 calls Timer1.startPeriodicAt(2000, 9000), and so on).

Every time Timer1 fires, the node sends to the Coordinator a PUBLISH message on a topic chosen at random with a random value between 0 and 100.

The Coordinator, at the reception of a PUBLISH message, forwards the PUBLISH message to all the nodes subscribed to the topic on which the publish has been done (it can check the subscribers looking at the array corresponding to the topic: subToTopic#).

After that, the Coordinator connects to Node-Red by opening the socket and sends the publish message to it.

Node-Red will process the message and will perform the publish on the ThingSpeak public channel we created for this purpose.

3 Node-Red Implementation

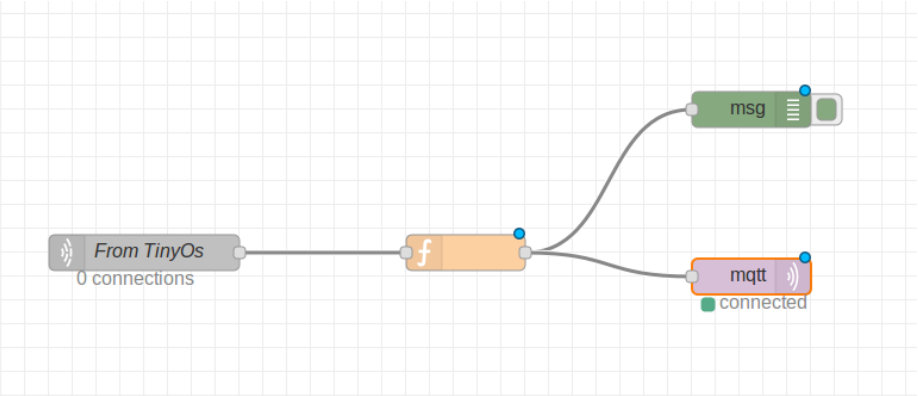


Figure 2: Node-Red Flow

In Node-Red we can see the following components:

- From TinyOS, that is a TCPin module, used to listen to the PAN Coordinator when it publishes the values
- Function that parses the messages from TinyOS, pulls over the topic, the value, and then creates an MQTT Publish message to be sent to the channel ID specified (the one of the ThingSpeak public channel)
- MQTT out is used to perform the publish on ThingSpeak public channel

4 ThingSpeak Charts

The public channel where the value for each topic is published is here
The Field1 Chart represents the variation of the temperature topic
The Field2 Chart represents the variation of the humidity topic
The Field3 Chart represents the variation of the luminosity topic

The values that a node can publish are limited to 100 as a maximum value

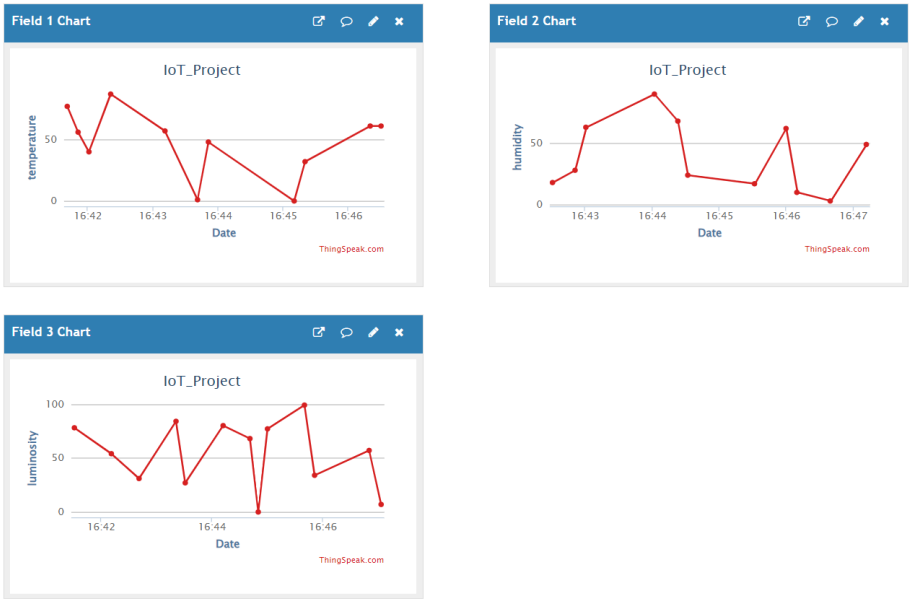


Figure 3: Public Charts

5 Conclusion

This project has been a wonderful opportunity to deal with some IoT tools, as the simulation tool TOSSIM, Node-Red and ThingSpeak.

The most interesting and tricky thing was to connect all of them such that they are able to interact with each other in a proper way, especially connecting the TOSSIM code with Node-Red.