



**POLITECNICO**  
MILANO 1863

INTERNET OF THINGS

PROJECT REPORT

# Data collection with Thingspeak

*Authors:*

MANASJYOTI BHUYAN (893800)

BAKTI ARIANI MELINDA PERTIWI (854745)

*Project advisor:*

ASSISTANT PROFESSOR A.REDONDI

*Course instructor:*

PROFESSOR M.CESANA

JUNE 28, 2019

# INDEX

1. Introduction	1
2. Implementation of the MQTT model with TinyOS	1
i. Booting phase	1
ii. Connection of the clients to the MQTT Broker	1
iii. Subscription of the clients to the topics	1
iv. Publishing a message	1
v. Screenshots	1
3. Implementation of the HTTP model with Contiki	2
i. HTTP GET (Pull Request)	2
ii. HTTP POST (Push Request)	2
iii. Screenshots	2
4. Some screenshots	3
i. Node-Red flow for extracting data from TOSSIM and Pushing it to Thingspeak.	3
ii. Node-Red flow for getting data from Thingspeak and for sending 'Alert' on Email upon exceed a predefined threshold	3
iii. Email alert on exceeding of predefined threshold of Average of Temperature	
5. Conclusion	3

## 1. Introduction

The goal of the project is to mimic two different WSN- one with MQTT (Fig.1) and the other with HTTP (Fig.2) that are connect to a 'temperature' and a 'humidity' sensor. Here we are required to collect the data from this sensors and upload in Thingspeak. Another requirement we had fulfilled was to collect the data from Thingspeak through Node-Red and send alert through email. We used TinyOS-TOSSIM for MQTT, while we used Contiki-Cooja for HTTP , for the simulation purpose.

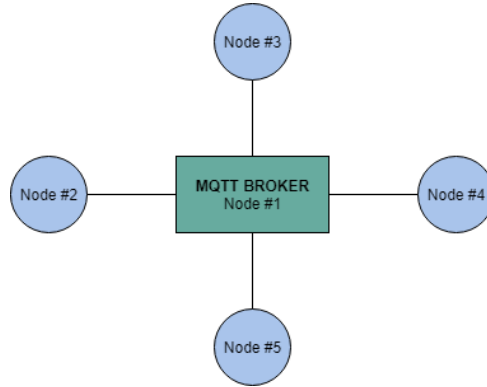


Fig 1: Topology of the MQTT model

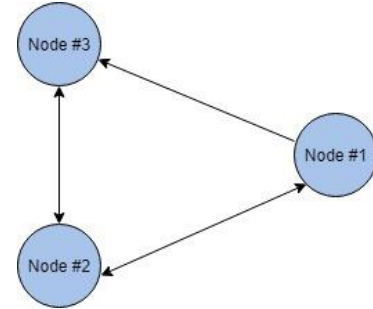


Fig 2: Topology of the HTTP model

## 2. Implementation of the MQTT model with TinyOS

### i. Booting phase

This is the first phase which is performed by all the nodes to set up their radio. For the MQTT broker, it also initializes variables that are useful to manage the network flow such connected nodes, subscriptions, etc., while for the clients it triggers the connection phase.

### ii. Connection of the clients to the MQTT Broker

After booting the clients need to get connected to the MQTT broker. To do so each node transmits a CONNECT message to the MQTT Broker which contains only one field which is the type of the message. Defined message types as: CONNECT=1, SUBSCRIBE=2, PUBLISH=3, FORWARD=4

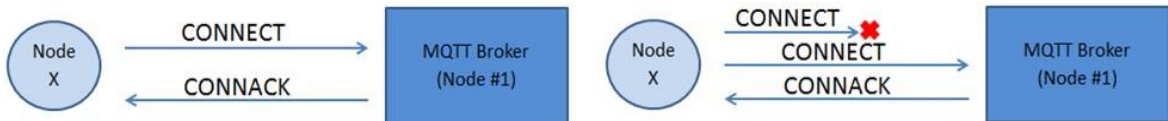


Fig 3: Acknowledgment policy for the Connect phase

### iii. Subscription of the clients to the topics

A node has to subscribe to some topics with a given QoS (Low QoS = 0, High QoS = 1). Here we have defined 2 topics: TEMPERATURE=1, HUMIDITY=2. The SUBSCRIBE message is composed of: Message type, Topic, QoS

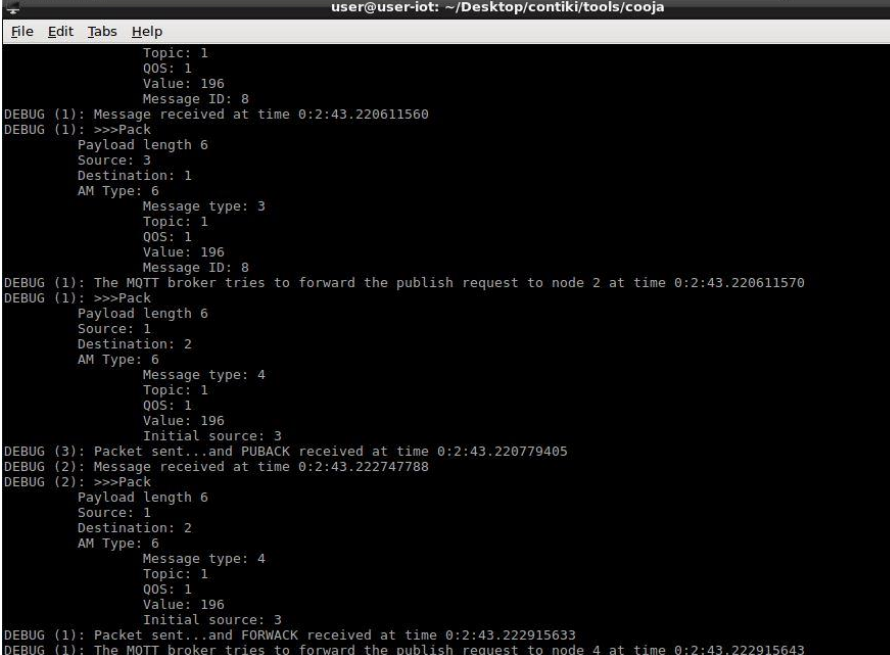


Fig 4: Acknowledgment policy for the subscribe phase

### iv. Publishing a message

The publish phase, was split into 2 subphases: publish and forward. In our implementation the publish subphase corresponds to the transmission from the client to the MQTT Broker while the forward phase corresponds to the distribution of that message from the Broker to the subscribed nodes.

## v. Screenshots



```

user@user-iot: ~/Desktop/contiki/tools/cooja
File Edit Tabs Help
Topic: 1
QoS: 1
Value: 196
Message ID: 8
DEBUG (1): Message received at time 0:2:43.220611560
DEBUG (1): >>>Pack
Payload length 6
Source: 3
Destination: 1
AM Type: 6
Message type: 3
Topic: 1
QoS: 1
Value: 196
Message ID: 8
DEBUG (1): The MQTT broker tries to forward the publish request to node 2 at time 0:2:43.220611570
DEBUG (1): >>>Pack
Payload length 6
Source: 1
Destination: 2
AM Type: 6
Message type: 4
Topic: 1
QoS: 1
Value: 196
Initial source: 3
DEBUG (3): Packet sent...and PUBACK received at time 0:2:43.220779405
DEBUG (2): Message received at time 0:2:43.222747788
DEBUG (2): >>>Pack
Payload length 6
Source: 1
Destination: 2
AM Type: 6
Message type: 4
Topic: 1
QoS: 1
Value: 196
Initial source: 3
DEBUG (1): Packet sent...and FORWACK received at time 0:2:43.222915633
DEBUG (1): The MQTT broker tries to forward the publish request to node 4 at time 0:2:43.222915643

```

Fig. 5: WSN via MQTT in TinyOS

## 3. Implementation of the HTTP model with Contiki

### i. HTTP GET (Pull Request)

In this GET method means the client request for a piece of resources from the server/sink and identified by the Request-URI. The Request-URI refers to a data-producing process. The produced data which shall be returned as the output of the process.

### ii. HTTP POST (Push Request)

POST request method is used to "post" additional data up to the server/sink.

### iii. Screenshots

This opens the COOJA terminal which runs sink in one node having IP address aaaa::212:7401:1:101 and client in the another node having IP address aaaa::212:7402:2:202. HTTP uses 60001 port number as default. The client in the example periodically accesses the resources of the sink and prints the payload which can be seen on the mote output.

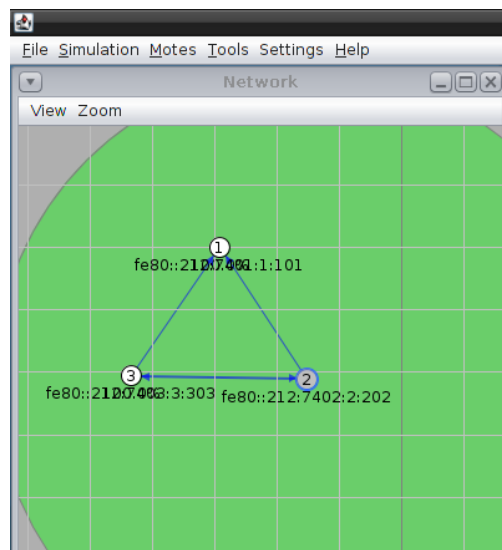
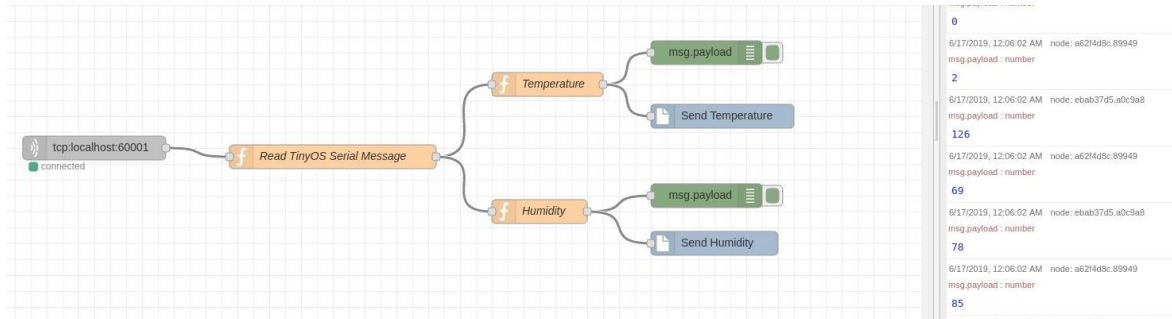


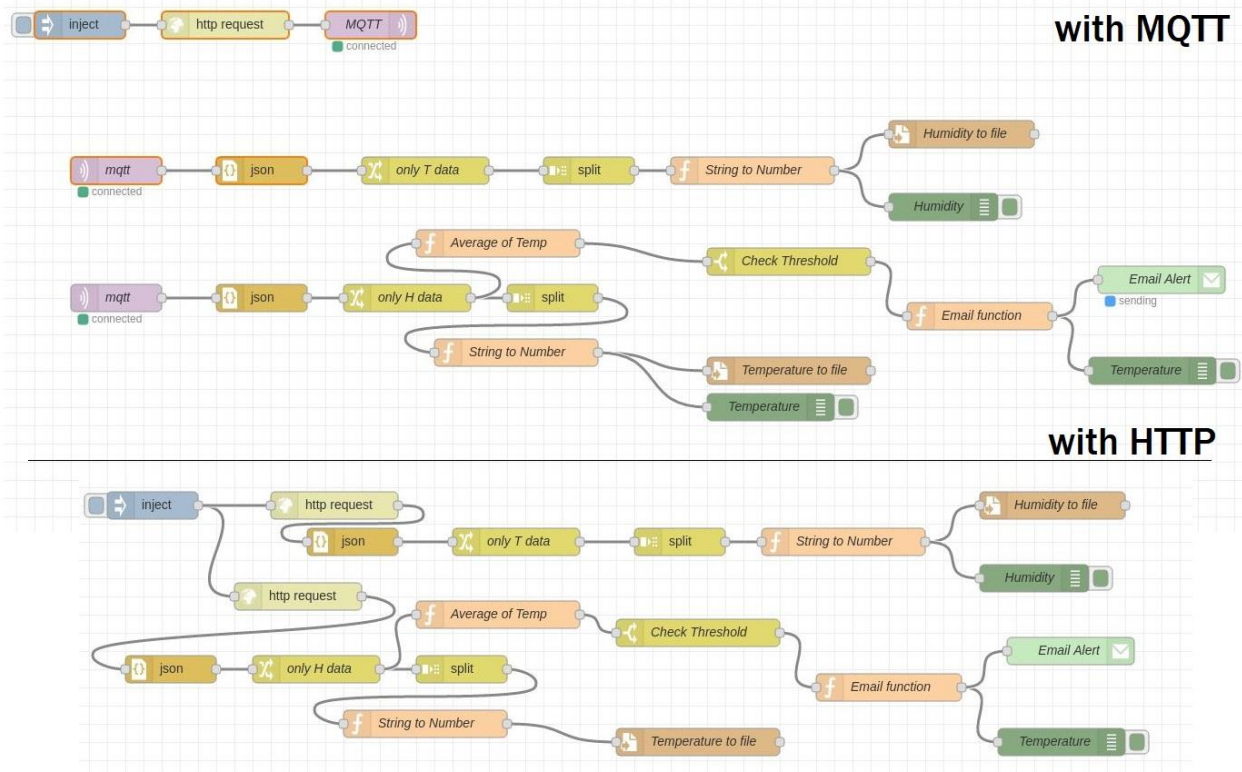
Fig. 6: WSN via HTTP in Cooja

#### 4. Some screenshots-

- i. Node-Red flow for extracting data from TOSSIM and Pushing it to Thingspeak.



- ii. Node-Red flow for getting data from Thingspeak and for sending 'Alert' on Email upon exceed a predefined threshold



- iii. Email alert on exceeding of predefined threshold of Average of Temperature

Temperature Rising Alert ➤



bakti.ariani@gmail.com  
to me

12:57 AM (2 min)

Emergency Alert! There is an high alert of increase in Temperature. Run!!Thu Jun 27 2019 17:57:14 GMT+0200 (CEST)

#### 5. Conclusion

This project was an optimal chance to handle the practical execution of an IOT application and get to know TinyOS, Contiki, Node-Red and Thingspeak better. The most interesting parts were undoubtedly the design choices for the QoS part in MQTT and the consideration of the various problematic situations in which the application could be plunged.

**Thingspeak:** <https://thingspeak.com/channels/802840> , <https://thingspeak.com/channels/803134>