Internet of Things
Project Work:
Automatization of Irrigation system

Francesco Bruno

Dipartimento di Ingegneria dell'Informazione

Università di Pisa

Master degree in Computer Engineering, course of Internet of Things

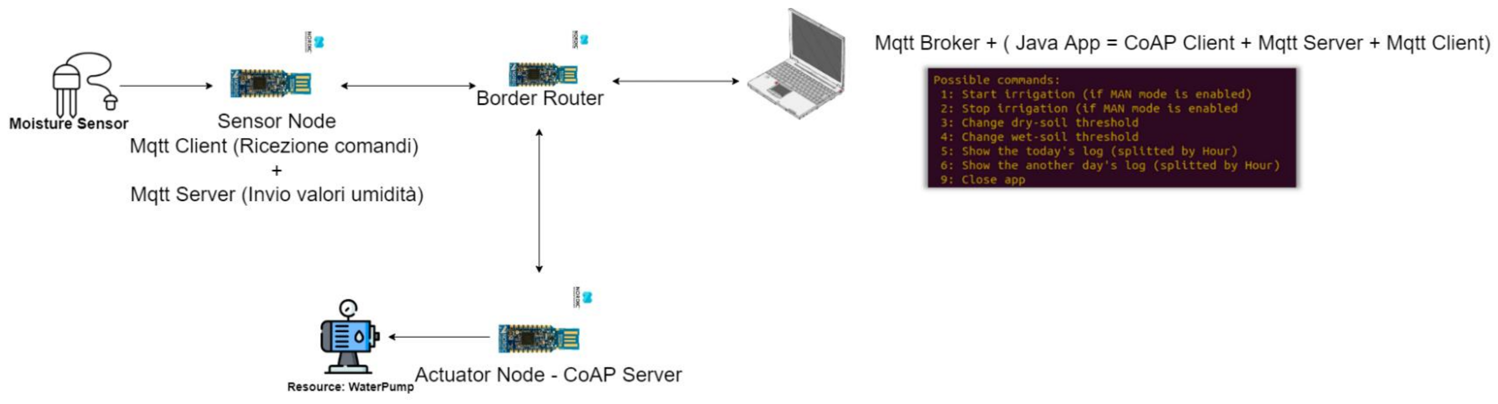
Index

Abstract

The project implements a telemetry system for keep tabs soil moisture, in which it is controlled by a water pump and a moisture-sensor.

The irrigation system’s interface is *IrrigApp*, a CLI application.
It shows a menu that allow to the user to change the irrigation mode (Auto or Manual), show the log and change the moisture threshold (see after for more details). Moreover, the app in background keeps track soil moisture, saves various events in a local log file and saves moisture values in a remote database.

The entire System shown in a picture


Moisture Sensor
Sensor Node
Mqtt Client (Ricezione comandi)
+
Mqtt Server (Invio valori umidità)
Border Router
Mqtt Broker + (Java App = CoAP Client + Mqtt Server + Mqtt Client)
Resource: WaterPump
Actuator Node - CoAP Server

```
Possible commands:
 1: Start irrigation (if MAN mode is enabled)
 2: Stop irrigation (if MAN mode is enabled
 3: Change dry-soil threshold
 4: Change wet-soil threshold
 5: Show the today's log (splitted by Hour)
 6: Show the another day's log (splitted by Hour)
 9: Close app
```

The telemetry system is comped by:

- Moisture Sensor Node.
  This network node acts as Mqtt Server when it takes care of sense moisture values and publish it on topic "viaNapoli/data".
  Moreover, this node behaves as Mqtt Client when it subscribes itself on topic "viaNapoli/commands". In this topic will be published all commands for this node, like the newThreshold-Command and pumpState-Command.
  So, in the future if I will add a new command, I only need to send a new Json message in this topic.
- Actuator Node – CoAP Server
  This node provides a CoAP observable resource which is the water pump.
  The other nodes in the network can change the pump state by CoAP message.
  I chosen to declare the pump-resource as observable. In this way, a node that want to know when the pump state change, it needs to register to CoAP Server.
  Once registered, if the java app changes the pump state, the node will receive automatically a (CoAP) notification of pump state change.
- Border Router (Sink Node)
  This node is the Gateway of LLN and takes care of connect the LLN with the MQTT-Broker, that is emulated by VM. So, it allow to Moisture-Sensor-Node to exchange mqtt-messages with mqtt-broker and allow to Actuator-Node to exchange coap-messages with java app.
- Irrig-App
  This is the CLI user interface and allow to the user to interact with the telemetry system.
  See the index to more details about IrrigApp.

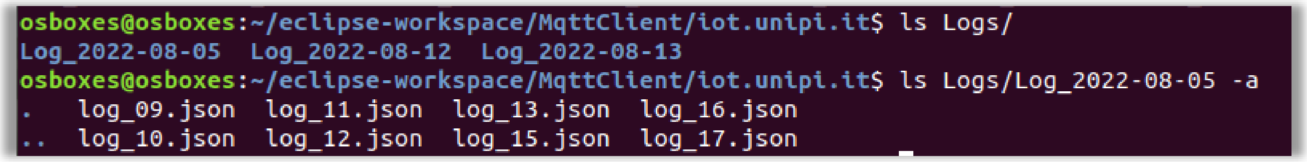
At startup, the moistureNode and the actuatorNode will search for RPL border router. You can see the yellow led blinking until the border router won't be found.
When the rpl will be found, the led stop blinking and the nodes can start the "operative mode".

Programming Language, Data Encoding Language choice

The application has been developed using JAVA programming language with eclipse IDE.

For data log encoding, I've chosen JSON because of its low redundant about objects respect to XML.

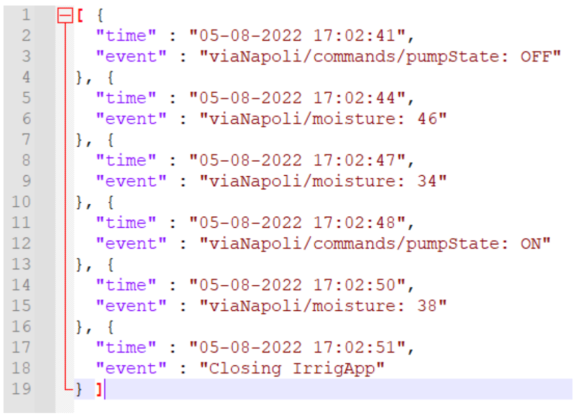
In order to have a staggered log, the application for every day will create a folder named ex. "LOG_2022-08-13" in which there will be at most 24 log files, one for each hour.



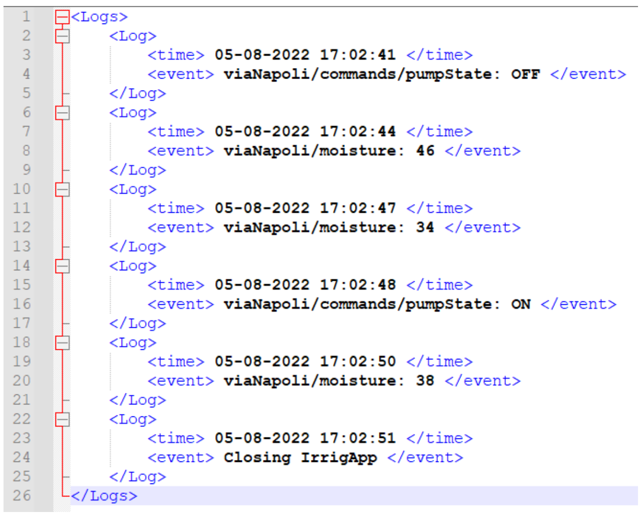
```
osboxes@osboxes:~/eclipse-workspace/MqttClient/iot.unipi.it$ ls Logs/
Log_2022-08-05  Log_2022-08-12  Log_2022-08-13
osboxes@osboxes:~/eclipse-workspace/MqttClient/iot.unipi.it$ ls Logs/Log_2022-08-05 -a
.   log_09.json  log_11.json  log_13.json  log_16.json
..  log_10.json  log_12.json  log_15.json  log_17.json
```

This is useful to keep track of all actions/events which are performed by network nodes, so you can find a specific instant of an occurred error.
I've chosen to capture these events:

- The starting of java app
- A new moisture value published by sensor node
- The water pump's state changing
- A new soil's wet/dry threshold
- The irrigation mode changing
- The closing of java app



```
[ {
  "time" : "05-08-2022 17:02:41",
  "event" : "viaNapoli/commands/pumpState: OFF"
}, {
  "time" : "05-08-2022 17:02:44",
  "event" : "viaNapoli/moisture: 46"
}, {
  "time" : "05-08-2022 17:02:47",
  "event" : "viaNapoli/moisture: 34"
}, {
  "time" : "05-08-2022 17:02:48",
  "event" : "viaNapoli/commands/pumpState: ON"
}, {
  "time" : "05-08-2022 17:02:50",
  "event" : "viaNapoli/moisture: 38"
}, {
  "time" : "05-08-2022 17:02:51",
  "event" : "Closing IrrigApp"
} ]
```

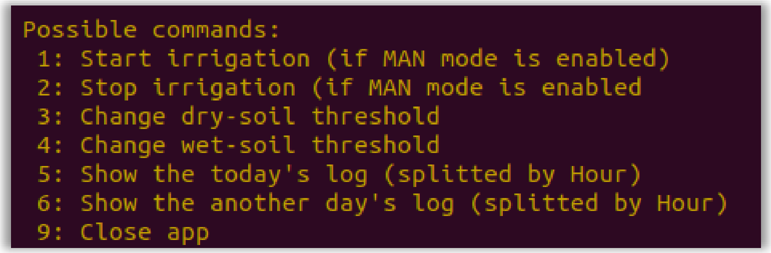


```
<Logs>
    <Log>
        <time> 05-08-2022 17:02:41 </time>
        <event> viaNapoli/commands/pumpState: OFF </event>
    </Log>
    <Log>
        <time> 05-08-2022 17:02:44 </time>
        <event> viaNapoli/moisture: 46 </event>
    </Log>
    <Log>
        <time> 05-08-2022 17:02:47 </time>
        <event> viaNapoli/moisture: 34 </event>
    </Log>
    <Log>
        <time> 05-08-2022 17:02:48 </time>
        <event> viaNapoli/commands/pumpState: ON </event>
    </Log>
    <Log>
        <time> 05-08-2022 17:02:50 </time>
        <event> viaNapoli/moisture: 38 </event>
    </Log>
    <Log>
        <time> 05-08-2022 17:02:51 </time>
        <event> Closing IrrigApp </event>
    </Log>
</Logs>
```

In the above picture is shown on left an example of my json file and on the right the equivalent in XML. The xml file is more redundant because of the opening and closing tags.
For this reason, I've chosen JSON as encoding language, and because I can better see the representation of my JAVA object. In addition, the application will save on Database the moisture value and info related to water pump's state changing.

How to use IrrigApp

Irrig-App is very simple interface. The bottom picture shown the interface thanks to it the user can interact with the irrigation system.



```
Possible commands:
 1: Start irrigation (if MAN mode is enabled)
 2: Stop irrigation (if MAN mode is enabled
 3: Change dry-soil threshold
 4: Change wet-soil threshold
 5: Show the today's log (splitted by Hour)
 6: Show the another day's log (splitted by Hour)
 9: Close app
```

On the starting, the app will show this simple menu, so the user has to choose a specific number to perform a specific operation.

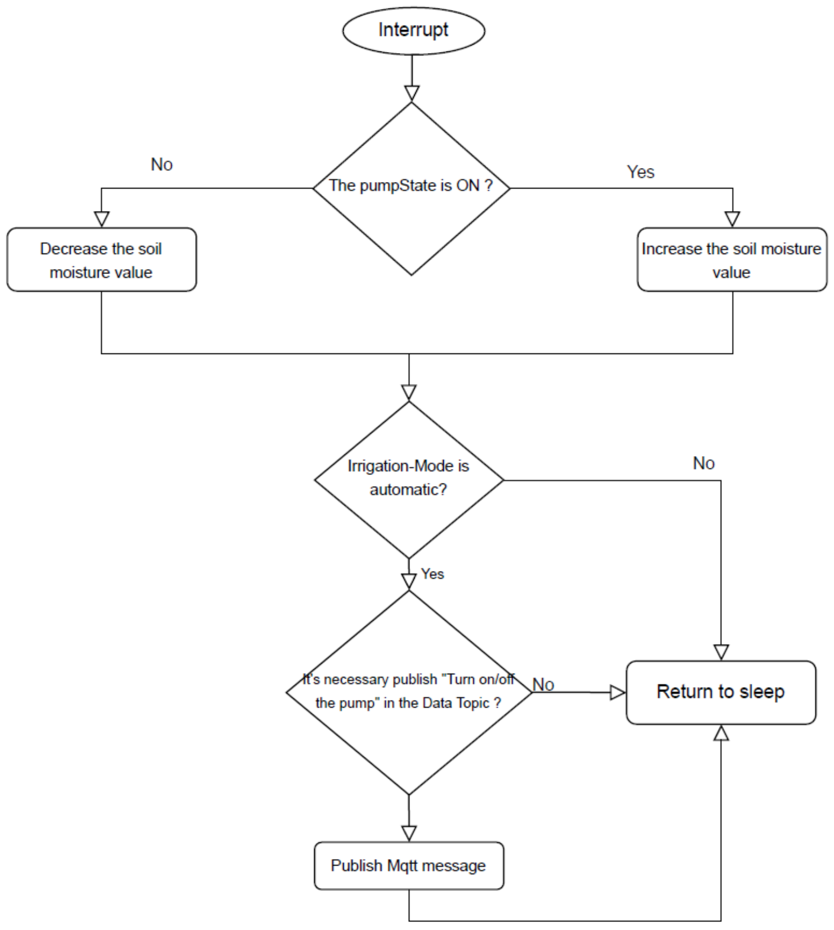
The application starts with the AUTO-Mode default, so the irrigation is automatized by network node.
If you want to change the irrigation-mode, you can simply press the button on Moisture-Node. You will receive a response on the display in order to have feedback.

If you want to change the dry/wet-soil threshold, you can choice the relative value on menu and the app will ask you the new value. If it is a correct value, this command will be sent to MoistureNode, in order to effectively change the threshold.

As I mentioned before, the app will construct a local log in a json file. If you wish, you can see all the event happened “today” (by pressing 5) or you can see the log of another day (by pressing 6). Later, the app will ask you to enter the wanted date.

How to work the automatic irrigation logic?

Remember that in the project the moisture sensor is emulated by MoistureNode. In details, the sensor is emulated by *fakeMoistureSensor* Class. Just below there is the simplified flow-chart of soil moisture change


Interrupt
The pumpState is ON ?
No
Yes
Decrease the soil moisture value
Increase the soil moisture value
Irrigation-Mode is automatic?
No
Yes
It's necessary publish "Turn on/off the pump" in the Data Topic ?
No
Return to sleep
Publish Mqtt message

emulation. Basically, in the Contiki process every 500ms a specified timer (*sensingTimer* attribute of above-mentioned class) is set and the relative interrupt is handled by *changeSensigvalue* function.

To change the irrigation mode, the user must press the button on the MoistureNode.
In order to notify the user of success irrigation mode change, when the *MAN* mode is enabled, a red light is turned-on on dongle. While, if the user presses the button when the MAN mode is already enabled, the red light will be turned off (*AUTO* mode is on now).

Meanwhile the *AUTO* mode is enabled, when the *DrySoil* threshold is reached by sensor(fake), the MoisturNode will publish a Mqtt-Message in which there will be the “Pump-ON” command.
In order to notify the successfully reception, a green light is turned-on on CoapServer dongle to give you a feedback and to simulate the start of water pump.
On the other hand, if the water pump is turned off, the green light will be turned-off.