University of Pisa

School of Engineering

Master of Science in Artificial Intelligence and Data Engineering

Project documentation

IoT Based Power Monitoring System for Diesel Generator

Work group:
Francesco Hudema
Jacopo Cecchetti

Academic year 2021-2022

Contents

Figure 1: The RF52840 USB Dongle by Nordic Semiconductors

1 Introduction

A Diesel generator is a fossil-fuel powered generator employed to supply electrical power. Diesel generating sets are used in places without connection to a power grid, or as emergency power-supply if the grid fails. Electrical power is the most important source for electronic devices. In industrial systems, electrical power is needed to keep control systems running for 24 hours. Stable power source should also be taken care to prevent devices from being damaged.

The objective of this project is to design and implement an IoT telemetry and control system for diesel generators used in a data center. The system is composed of nodes that keep track of the energy generated by the generator, its fuel level and its temperature with their sensors. Sampled data are sent by the monitors, either using MQTT or CoAP, to a collector, which saves them in a database for data collection and analysis. The collector can also detect anomalies in the measurements and handle them.

2 Design

2.1 Overview

The system is composed of by a network of diesel generator monitors, installed in a data center. Each monitor is a IoT device (nRF52840 USB Dongle by Nordic Semiconductors) equipped with the Contiki-NG operating system and exchanging data with a collector, which is executed on a Ubuntu virtual machine.

The network is a LLN exploiting the 802.15.4 and IPv6 protocols, where the multihop communication is made possible by RPL and the traffic with the collector is enabled by a border router.

The monitors are equipped with three different sensors (temperature, fuel level and energy generated) and an alarm system. The latter is able to turn on a set of LEDs when activated. The

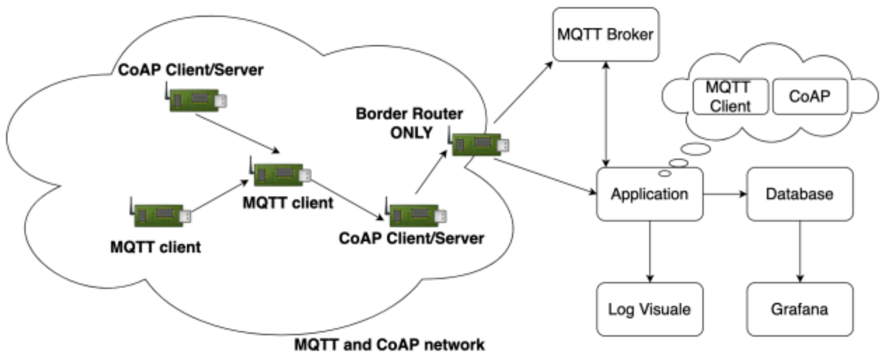


Figure 2: The system scheme

devices support MQTT and CoAP as application protocols to exchange measurements and actuator commands. The MQTT broker is deployed on the Ubuntu virtual machine.

The collector, written in Java, is responsible for receiving data from the dongles, saving them in a database and detecting anomalies in the temperature and fuel level measurements.
A monitor operates in the following way:

1. once turned on, it computes its ID using the link layer address
2. it waits until it is connected to the network
3. once connected, CoAP nodes send a registration message to the collector, while MQTT nodes register with the MQTT broker
4. once registered, the monitor is fully operational. During this state it periodically acquires data on the assigned diesel generator using its sensors. Once a new sample is ready, the monitor sends it to the collector
5. at the same time the collector analyses the samples. If the measurements are above/below a certain threshold the collector sends an alarm message the node
6. depending on the type of alarm the node turns on a certain led. During the fully operational state, if the temperature value is under a predefined threshold, the LD2 led is always on and its color is blue. If the temperature exceeds the threshold the led color changes to red and the energy generated is reduced until the temperature gets below the defined threshold. Instead, if the fuel level gets below the minimum threshold, the node turns on the LD1 led until a data center operator manually tops up the generator tank with diesel. The operator then presses the button of the device, which triggers the shutdown of the alarm

At each step the exchanged data is encoded in plain JSON using a proprietary format. This choice allows for a more lightweight communication, faster processing times and a simpler representation of information.

2.2 MQTT node

An MQTT diesel generator node connects as a client to an MQTT broker, publishing and subscribing to dedicated topics to communicate with the collector. Each node is characterised by an ID, which is derived from its link layer address. The nodes publish the data to the following topics: *energy_generated*, *fuel_level* and *temperature*, while the monitor publishes on *alarm_ID* topic of each nodes to notify the alarm message.

- `topic_name` holding the last value sampled by all the sensors in the format specified above. For example, the resource temperature will make available the latest measurement as {"n": "fuel_level", "v": "746", "u": "L", "id": "2"}.

To receive the alerts from the collector the sensor subscribes to the following topic:

- `alarm_NODE_ID` which is used to receive alarm messages from the collector. The messages send by the collector are the following: `NO_ERROR`, `FUEL_LEVEL_ERROR` and `TEMPERATURE_ERROR`.

2.3 CoAP node

A CoAP diesel generator monitor acts both as a client and server. It registers to the collector issuing a GET request to the server. In addition to this it exposes a sets of resources: *energy_generated*, *fuel_level* and *temperature* in order to notify the collector about state changes and to receive actuator commands. The telemetry related resources are observable.
The monitor exposes the following endpoints:

- `/topic_name`, holding the last value sampled by the corresponding sensor in the format specified above. For example, at the endpoint `temperature` the monitor will make available the latest measurement as `{"n": "temperature", "v": "58", "u": "C", "id": "3"}`. This endpoint only accepts GET requests;

- `/alert`, which is used to receive alarm messages from the collector. It accepts POST/PUT requests with a plain text payload containing the type of anomaly detected. For example `value=2` means that the collector has detected a temperature measurement above the predefined threshold. The POST/PUT handler will then take care of changing the LED colors and notifying the relative processes. If `value=0` is received from the collector it means that the temperature alarm can be deactivated because the measurements have gone back to normal. `value=1` means that the collector has detected a low fuel level. This issue can only be resolved manually.

2.4 Collector

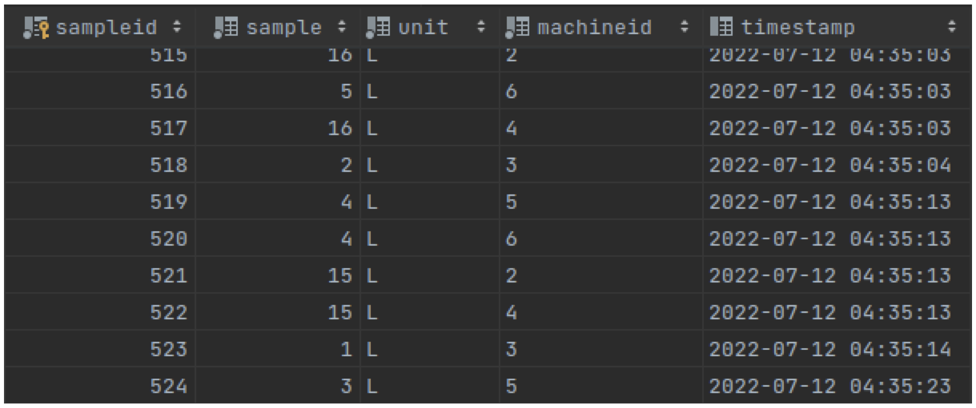
The collector is responsible for interacting with bot MQTT and CoAP monitors, saving the data in a MySQL database and detecting anomalies in the measurements.

To receive information from the MQTT monitors, the collector subscribes to the topics:

- `/resoruce_name` in order to receive telemetry data sent by all the monitors.

To send informations to the MQTT monitors, the collector publishes to the topics:

- `/alarm_NODE_ID` in order to notify the nodes when it detects an anomaly in the measurements.



sampleid	sample	unit	machineid	timestamp
515	16	L	2	2022-07-12 04:35:03
516	5	L	6	2022-07-12 04:35:03
517	16	L	4	2022-07-12 04:35:03
518	2	L	3	2022-07-12 04:35:04
519	4	L	5	2022-07-12 04:35:13
520	4	L	6	2022-07-12 04:35:13
521	15	L	2	2022-07-12 04:35:13
522	15	L	4	2022-07-12 04:35:13
523	1	L	3	2022-07-12 04:35:14
524	3	L	5	2022-07-12 04:35:23

Figure 3: The database table

The CoAP collector is deployed on the virtual machine outside the LLN, so the nodes need to register to the collector which is assigned a well known ip, in our case localhost. After a node registers, the collector sets up the observe relations to receive updates about the measurements made by the sensors belonging to that node. An observe relation is created by instantiating a CoapCollector class for each possible sensor. Each time the Collector receives a new sample, it checks if the values are below/above a certain threshold. If an anomaly is detected the collector sends a POST request to the `/alarm` endpoint specifying the type of error in the payload of the request.

2.5 Database

Our database consists of three tables, one for each type of sensor (temperature, fuel level and energy generated). In each table we save the measurements taken by the sensors along with an auto incremented id, the sample unit, the node id and a timestamp generated at insertion time.

2.6 Data encoding

Regarding the data encoding, we decided to proceed as follows:

- the exchange of measurements between both CoAP and MQTT nodes and the collector is implemented with JSON messages. This choice allows for a more lightweight communication, faster processing times and a simpler representation of information

```
    {"n": "TOPIC",
     "v": "SAMPLE_VALUE",
     "u": "SAMPLE_UNIT",
     "id": "NODE_ID"
     }
```

- the POST request sent by the collector to an actuator in the CoAP network has a plain text payload due to the fact that the C CoAP library comes with predefined functions that allow the parsing of data inside the payload of the request

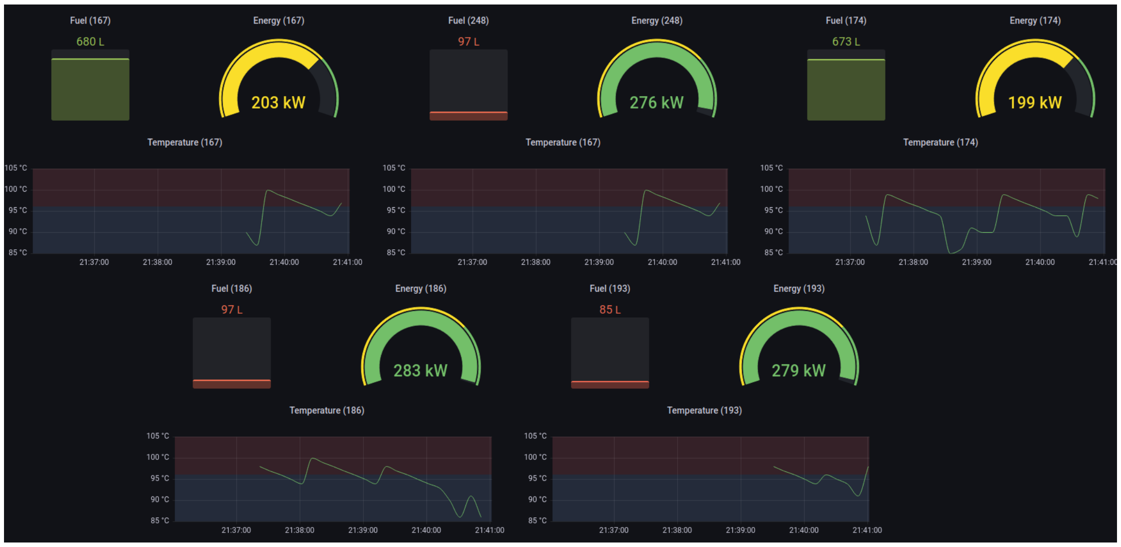


Figure 4: The Grafana dashboard

- the alert sent by the collector to an actuator in the MQTT network has a plain text message containing the error type

2.7 Grafana

In order to visualize the status of each diesel generator we created a Grafana dashboard. We defined a section for each node. Inside each section we have three panels:

- the fuel panel shows the most recent measurement made by the fuel sensor. The bar turns red when the fuel level goes below a predefined threshold
- the energy panel shows the most recent measurement made by the energy sensor
- the temperature panel shows the temperature trend. Whenever the temperature goes above the threshold we can see that the energy generated is reduced by an actuator until the temperature returns to normal

3 Testing

The dongles employed are IoT devices without any sensing capability, therefore we need to simulate the sensors using the C language. We do that by defining and starting a process for each type of sensor. During the operating phase a sensor performs measurements by generating a random value inside a predefined range. For example the temperature sensor range is between 85 and 100.

In addition to this also the alert system is simulated:

- when the temperature measurement gets above the temperature threshold the alert is simulated by changing the LD2 led color to red. Moreover we simulate actuator commands by reducing the energy generated measurements and by altering the way the temperature samples are generated. In fact, at each measurement the temperature is reduced by 1 degree Celsius in order to simulate the diesel generator returning to a normal temperature
- when the fuel level measurement gets below the fuel level threshold the alert is simulated by turning on the LD1 led. The fuel level returns to normal when the device button is pressed in order to simulate a data center operator performing diesel refueling

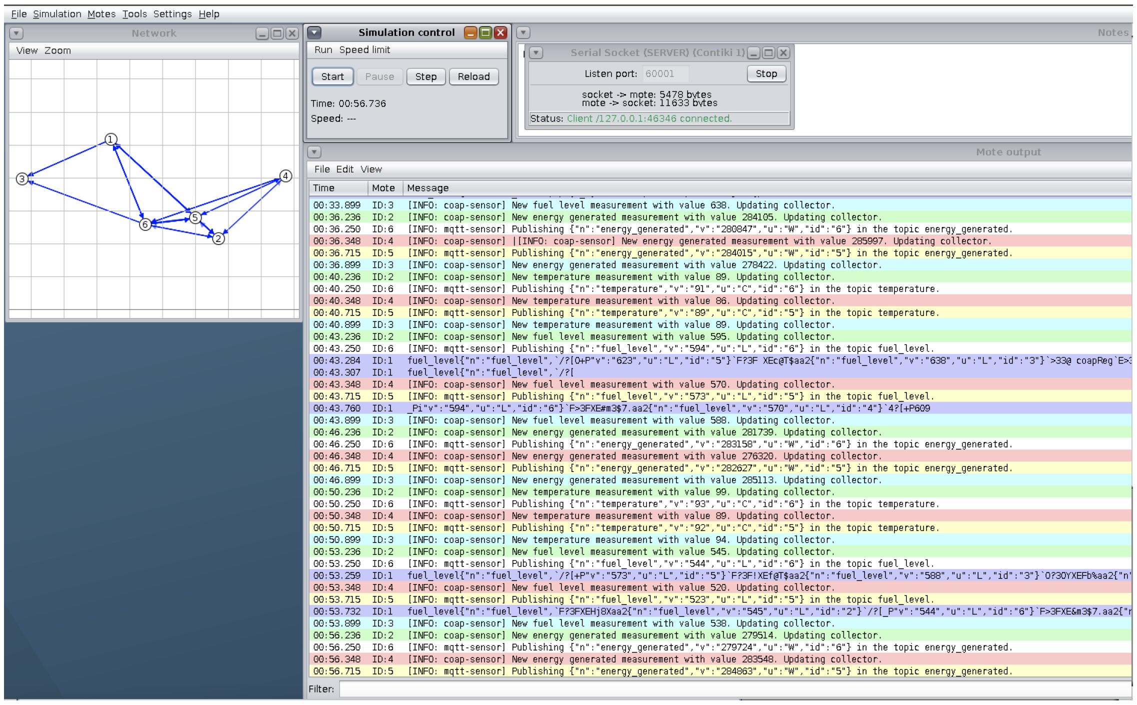
The collector is implemented in Java and contains both MQTT and CoAP collectors.

3.1 Cooja simulation

The test is performed by deploying all the components (MQTT broker, Java collector, database) on the same Ubuntu virtual machine. The IoT system is simulated by creating a single LLN in which a border router (node 1), three CoAP nodes (nodes 2, 3 and 4) and two MQTT nodes (nodes 5 and 6) are istantiated as Cooja motes.

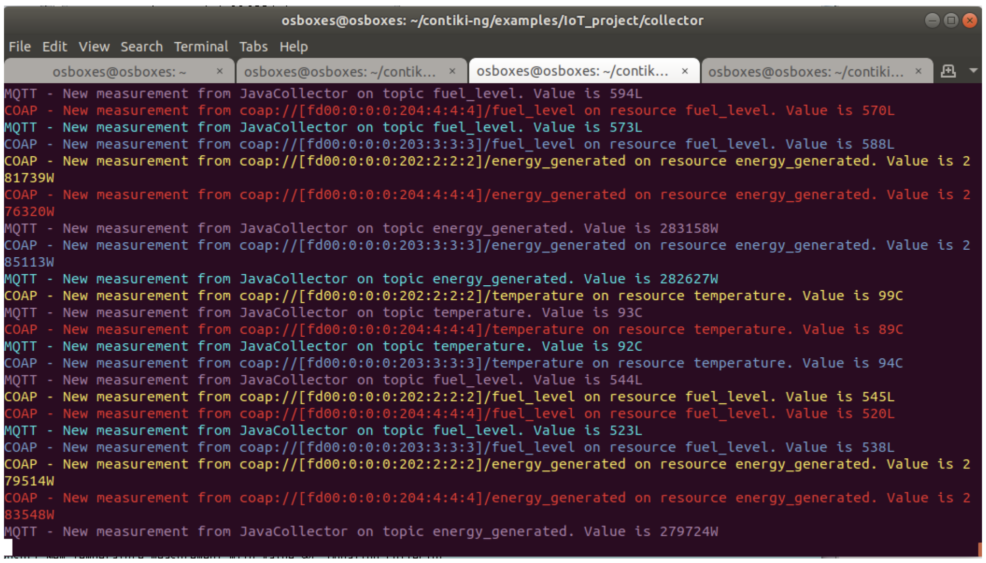
The default configuration is used:

- the broker listens on port 1883
- the CoAP collector is reachable at [fd00::1]:5683
- the database is reachable at localhost:3306
- new samples are generated every 10 seconds



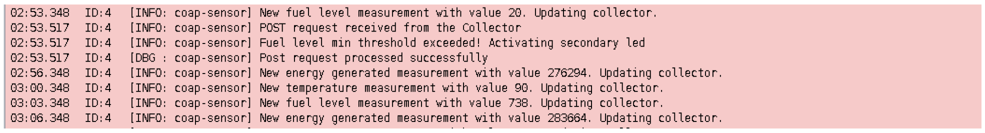
```
Time: 00:56.736
Speed: ---

Listen port: 60001
socket -> mote: 5478 bytes
mote -> socket: 11633 bytes
Status: Client /127.0.0.1:46346 connected.

Time       Mote  Message
00:33.899  ID:3  [INFO: coap-sensor] New fuel level measurement with value 638. Updating collector.
00:36.236  ID:2  [INFO: coap-sensor] New energy generated measurement with value 284105. Updating collector.
00:36.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"energy_generated","v":"280847","u":"W","id":"6"} in the topic energy_generated.
00:36.348  ID:4  [INFO: coap-sensor] |[INFO: coap-sensor] New energy generated measurement with value 285997. Updating collector.
00:36.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"energy_generated","v":"284015","u":"W","id":"5"} in the topic energy_generated.
00:36.899  ID:3  [INFO: coap-sensor] New energy generated measurement with value 278422. Updating collector.
00:40.236  ID:2  [INFO: coap-sensor] New temperature measurement with value 89. Updating collector.
00:40.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"temperature","v":"91","u":"C","id":"6"} in the topic temperature.
00:40.348  ID:4  [INFO: coap-sensor] New temperature measurement with value 86. Updating collector.
00:40.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"temperature","v":"89","u":"C","id":"5"} in the topic temperature.
00:40.899  ID:3  [INFO: coap-sensor] New temperature measurement with value 89. Updating collector.
00:43.236  ID:2  [INFO: coap-sensor] New fuel level measurement with value 595. Updating collector.
00:43.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"fuel_level","v":"594","u":"L","id":"6"} in the topic fuel_level.
00:43.284  ID:1  fuel_level{"n":"fuel_level",`/?[O+P"v":"623","u":"L","id":"5"}`F?3F XEc@T$aa2{"n":"fuel_level","v":"638","u":"L","id":"3"}`>33@ coapReg`E>
00:43.307  ID:1  fuel_level{"n":"fuel_level",`/?[
00:43.348  ID:4  [INFO: coap-sensor] New fuel level measurement with value 570. Updating collector.
00:43.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"fuel_level","v":"573","u":"L","id":"5"} in the topic fuel_level.
00:43.760  ID:1  _Pi"v":"594","u":"L","id":"6"}`F>3FXE#m3$7.aa2{"n":"fuel_level","v":"570","u":"L","id":"4"}`4?[+P609
00:43.899  ID:3  [INFO: coap-sensor] New fuel level measurement with value 588. Updating collector.
00:46.236  ID:2  [INFO: coap-sensor] New energy generated measurement with value 281739. Updating collector.
00:46.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"energy_generated","v":"283158","u":"W","id":"6"} in the topic energy_generated.
00:46.348  ID:4  [INFO: coap-sensor] New energy generated measurement with value 276320. Updating collector.
00:46.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"energy_generated","v":"282627","u":"W","id":"5"} in the topic energy_generated.
00:46.899  ID:3  [INFO: coap-sensor] New energy generated measurement with value 285113. Updating collector.
00:50.236  ID:2  [INFO: coap-sensor] New temperature measurement with value 99. Updating collector.
00:50.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"temperature","v":"93","u":"C","id":"6"} in the topic temperature.
00:50.348  ID:4  [INFO: coap-sensor] New temperature measurement with value 89. Updating collector.
00:50.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"temperature","v":"92","u":"C","id":"5"} in the topic temperature.
00:50.899  ID:3  [INFO: coap-sensor] New temperature measurement with value 94. Updating collector.
00:53.236  ID:2  [INFO: coap-sensor] New fuel level measurement with value 545. Updating collector.
00:53.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"fuel_level","v":"544","u":"L","id":"6"} in the topic fuel_level.
00:53.259  ID:1  fuel_level{"n":"fuel_level",`/?[+P"v":"573","u":"L","id":"5"}`F?3F!XEf@T$aa2{"n":"fuel_level","v":"588","u":"L","id":"3"}`O?3OYXEFb%aa2{"n"
00:53.348  ID:4  [INFO: coap-sensor] New fuel level measurement with value 520. Updating collector.
00:53.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"fuel_level","v":"523","u":"L","id":"5"} in the topic fuel_level.
00:53.732  ID:1  fuel_level{"n":"fuel_level",`F?3FXEHj8Xaa2{"n":"fuel_level","v":"545","u":"L","id":"2"}`/?[_P"v":"544","u":"L","id":"6"}`F>3FXE&m3$7.aa2{"r
00:53.899  ID:3  [INFO: coap-sensor] New fuel level measurement with value 538. Updating collector.
00:56.236  ID:2  [INFO: coap-sensor] New energy generated measurement with value 279514. Updating collector.
00:56.250  ID:6  [INFO: mqtt-sensor] Publishing {"n":"energy_generated","v":"279724","u":"W","id":"6"} in the topic energy_generated.
00:56.348  ID:4  [INFO: coap-sensor] New energy generated measurement with value 283548. Updating collector.
00:56.715  ID:5  [INFO: mqtt-sensor] Publishing {"n":"energy_generated","v":"284863","u":"W","id":"5"} in the topic energy_generated.
```

Figure 5: The output of the Cooja simulation


osboxes@osboxes: ~/contiki-ng/examples/IoT_project/collector
File Edit View Search Terminal Tabs Help
osboxes@osboxes: ~
osboxes@osboxes: ~/contik...
osboxes@osboxes: ~/contik...
osboxes@osboxes: ~/contiki...

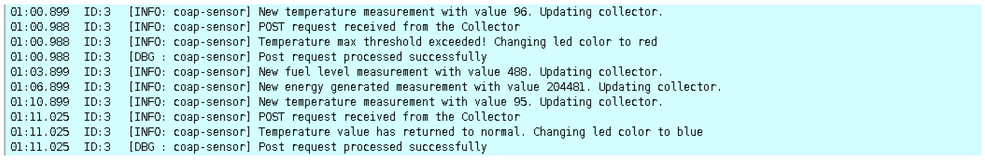
```
MQTT - New measurement from JavaCollector on topic fuel_level. Value is 594L
COAP - New measurement from coap://[fd00:0:0:0:204:4:4:4]/fuel_level on resource fuel_level. Value is 570L
MQTT - New measurement from JavaCollector on topic fuel_level. Value is 573L
COAP - New measurement from coap://[fd00:0:0:0:203:3:3:3]/fuel_level on resource fuel_level. Value is 588L
COAP - New measurement from coap://[fd00:0:0:0:202:2:2:2]/energy_generated on resource energy_generated. Value is 2
81739W
COAP - New measurement from coap://[fd00:0:0:0:204:4:4:4]/energy_generated on resource energy_generated. Value is 2
76320W
MQTT - New measurement from JavaCollector on topic energy_generated. Value is 283158W
COAP - New measurement from coap://[fd00:0:0:0:203:3:3:3]/energy_generated on resource energy_generated. Value is 2
85113W
MQTT - New measurement from JavaCollector on topic energy_generated. Value is 282627W
COAP - New measurement from coap://[fd00:0:0:0:202:2:2:2]/temperature on resource temperature. Value is 99C
MQTT - New measurement from JavaCollector on topic temperature. Value is 93C
COAP - New measurement from coap://[fd00:0:0:0:204:4:4:4]/temperature on resource temperature. Value is 89C
MQTT - New measurement from JavaCollector on topic temperature. Value is 92C
COAP - New measurement from coap://[fd00:0:0:0:203:3:3:3]/temperature on resource temperature. Value is 94C
MQTT - New measurement from JavaCollector on topic fuel_level. Value is 544L
COAP - New measurement from coap://[fd00:0:0:0:202:2:2:2]/fuel_level on resource fuel_level. Value is 545L
COAP - New measurement from coap://[fd00:0:0:0:204:4:4:4]/fuel_level on resource fuel_level. Value is 520L
MQTT - New measurement from JavaCollector on topic fuel_level. Value is 523L
COAP - New measurement from coap://[fd00:0:0:0:203:3:3:3]/fuel_level on resource fuel_level. Value is 538L
COAP - New measurement from coap://[fd00:0:0:0:202:2:2:2]/energy_generated on resource energy_generated. Value is 2
79514W
COAP - New measurement from coap://[fd00:0:0:0:204:4:4:4]/energy_generated on resource energy_generated. Value is 2
83548W
MQTT - New measurement from JavaCollector on topic energy_generated. Value is 279724W
```

Figure 6: The output of the collector



```
02:53.348  ID:4  [INFO: coap-sensor] New fuel level measurement with value 20. Updating collector.
02:53.517  ID:4  [INFO: coap-sensor] POST request received from the Collector
02:53.517  ID:4  [INFO: coap-sensor] Fuel level min threshold exceeded! Activating secondary led
02:53.517  ID:4  [DBG : coap-sensor] Post request processed successfully
02:56.348  ID:4  [INFO: coap-sensor] New energy generated measurement with value 276294. Updating collector.
03:00.348  ID:4  [INFO: coap-sensor] New temperature measurement with value 90. Updating collector.
03:03.348  ID:4  [INFO: coap-sensor] New fuel level measurement with value 738. Updating collector.
03:06.348  ID:4  [INFO: coap-sensor] New energy generated measurement with value 283664. Updating collector.
```

Figure 7: Fuel level minimum threshold exceeded



```
01:00.899  ID:3  [INFO: coap-sensor] New temperature measurement with value 96. Updating collector.
01:00.988  ID:3  [INFO: coap-sensor] POST request received from the Collector
01:00.988  ID:3  [INFO: coap-sensor] Temperature max threshold exceeded! Changing led color to red
01:00.988  ID:3  [DBG : coap-sensor] Post request processed successfully
01:03.899  ID:3  [INFO: coap-sensor] New fuel level measurement with value 488. Updating collector.
01:06.899  ID:3  [INFO: coap-sensor] New energy generated measurement with value 204481. Updating collector.
01:10.899  ID:3  [INFO: coap-sensor] New temperature measurement with value 95. Updating collector.
01:11.025  ID:3  [INFO: coap-sensor] POST request received from the Collector
01:11.025  ID:3  [INFO: coap-sensor] Temperature value has returned to normal. Changing led color to blue
01:11.025  ID:3  [DBG : coap-sensor] Post request processed successfully
```

Figure 8: Temperature maximum threshold exceeded

3.2 Deployment and execution

The code can be found inside this GitHub repository.

3.2.1 Database

In order for the system to work it is necessary either to create a new database and the tables or to create a new database and import the MySQL dump.

```
CREATE DATABASE smartgenerator;

CREATE TABLE smartgenerator.energy_generated (
sampleid  int auto_increment primary key,
sample    float not null,
unit      varchar(45) not null,
machineid varchar(45) not null,
timestamp datetime default CURRENT_TIMESTAMP null);

CREATE TABLE smartgenerator.fuel_level (
sampleid  int auto_increment primary key,
sample    float not null,
unit      varchar(45) not null,
machineid varchar(45) not null,
timestamp datetime default CURRENT_TIMESTAMP null);

CREATE TABLE smartgenerator.energy_generated (
sampleid  int auto_increment primary key,
sample    float not null,
unit      varchar(45) not null,
machineid varchar(45) not null,
timestamp datetime default CURRENT_TIMESTAMP null);
```

3.2.2 Collector

Before deploying the collector check that the Mosquitto MQTT broker is running. To deploy the collector move into the `IoT_Project/collector` folder and run the following commands:

```
mvn clean install
mvn package
java -jar target/collector.iot.unipi.it-0.0.1-SNAPSHOT.jar
```

3.2.3 Simulation

Open the Cooja simulator and import `SmartGenerator-simulation.csc`. To deploy the border router:

- add the socket on the border router (Tools -> Serial Socket (SERVER) -> Contiki 1)
- start the serial socket
- use Tunslip6

  ```
  make TARGET=cooja connect-router-cooja
  ```

  Start the simulation.

3.2.4 Deployement on the nRF52840-Dongle

To deploy the system on the dongles flash the code on all the nodes:

- border router:

  ```
  make TARGET=nrf52840 BOARD=dongle border-router.dfu-upload PORT=/dev/ttyACM0
  ```

- MQTT nodes:

  ```
  make TARGET=nrf52840 BOARD=dongle mqtt-sensor.dfu-upload PORT=/dev/ttyACM0
  ```

- CoAP nodes:

  ```
  make TARGET=nrf52840 BOARD=dongle coap-sensor.dfu-upload PORT=/dev/ttyACM0
  ```

  After that power on all the nodes and use Tunslip6 to connect with the border router:

```
make TARGET=nrf52840 BOARD=dongle connect-router PORT=/dev/ttyACM0
```