

HVAC Waste Detection - Project Presentation

Daniele Polidori

Course of Internet of things
University of Bologna

Academic year 2023-24

Introduction

I realized an IoT system that monitors the temperature of an house, to prevent useless HVAC consumption, detecting it by rapid temperature changes.

Composed by an
ESP-WROOM-32 board
linked to:

- ▶ 1 **indoor DHT22** sensor,
- ▶ 1 **outdoor DHT22** sensor,
- ▶ 1 **LED**.



Data acquisition - ESP32

`Thing.CoAP` : To act as a CoAP server.

`PubSubClient` : To act as a MQTT subscriber.

Through the CoAP protocol, the ESP32 is able to send the latest collected indoor and outdoor temperature value, when asked.

Through the MQTT protocol, the board can receive some commands:

- ▶ to start or stop the sensors reading,
- ▶ to change the interval between consecutive sensors readings,
- ▶ to turn on or off the LED.

My laptop acts as a MQTT broker, through Mosquitto.

Data proxy - 1st Python script

`paho-mqtt` : To act as a MQTT publisher.

`aiocoap` : To act as a CoAP client.

`influxdb-client` : To store data.

Initially, through the MQTT protocol, the application gives commands, to the ESP32, to start the sensors reading and to set their sampling rate.

Periodically, through the CoAP protocol, the script requests, to the board, the latest collected indoor and outdoor temperature value. It continuously stores these values on a local InfluxDB instance.

The network latency, between the temperature value request and its reception, is continuously monitored; after a while, the script evaluates the mean latency of this process.

Data analytics - 2nd Python script (1/2)

`influxdb-client` : To get and store data on the database.

`prophet` : To forecast future temperature values.

`paho-mqtt` : To act as a MQTT publisher.

At the beginning, the script gets all past temperature values from the database to make a prediction of some indoor and outdoor future values. As the forecasted temperature times are reached, the application stores the predicted values on the database.

Ciclically, the application retrieves some of the latest temperature values from the database and analyses them to check a possible HVAC waste.

When the alarm goes off, the script stores the alarm event on the database and, through the MQTT protocol, gives the command, to the ESP32, to turn on the LED; when the risk has passed, the script gives the command to turn it off.

Data analytics - 2nd Python script (2/2)

The alarm goes off if:

- ▶ the indoor temperature is changing rapidly (1) and
- ▶ the indoor temperature is approaching the outdoor one (2).

Mathematically speaking:

$$var(i_1, i_2, \dots, i_n) > threshold \quad (1)$$

$$min(i_n, o_n) < mean(i_1, i_2, \dots, i_n) < max(i_n, o_n) \quad (2)$$

where i is the indoor temperature, o is the outdoor temperature and t_1, t_2, \dots, t_n are the n latest temperature values retrieved: t_1 is the newest and t_n is the farthest.

Data visualization

Local Grafana instance that shows:

- ▶ the collected temperature values,
- ▶ the forecasted ones,
- ▶ the counting of the alarm events.

