HVAC Waste Detection - Project Presentation

Daniele Polidori

Course of Internet of things
University of Bologna

Academic year 2023-24

IoT system

It 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**,
- 1 **outdoor DHT22**,
- 1 **LED**.

Data acquisition - ESP32

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

`PubSubClient` : To act as a MQTT subscriber.

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

Through MQTT, the board can receive some commands:

- to start or stop the sensors reading,
- to change their sampling rate,
- 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 MQTT, the application gives commands, to the ESP32, to start the sensors reading and to set their sampling rate.

Periodically, through CoAP, 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 and, after a while from the beginning, the mean value is calculated.

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

`influxdb-client` : To get and store data.

`prophet` : To forecast future temperature values.

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

At the beginning, the script gets all past temperature values to forecast some indoor and outdoor values.
As the times are reached, the application stores the predicted values on the database.

Ciclically, the application retrieves some of the latest temperature values and analyses them to check a possible HVAC waste.
When the alarm goes off, the script stores the event on the database and, through MQTT, 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:

$$var(i_1, i_2, \ldots, i_n) > threshold \tag{1}$$

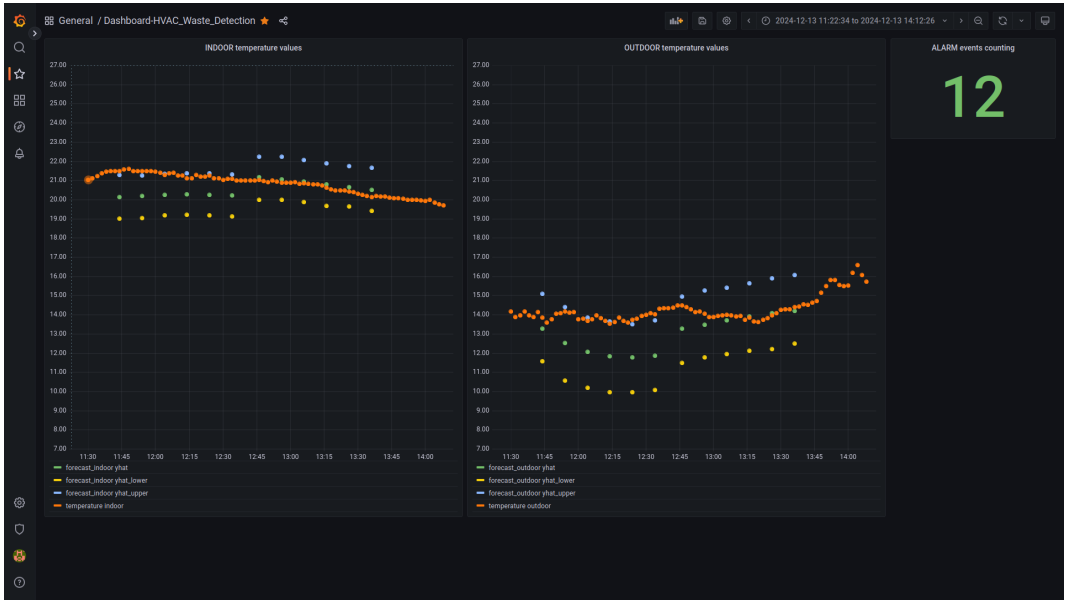
$$min(i_n, o_n) < mean(i_1, i_2, \ldots, i_n) < max(i_n, o_n) \tag{2}$$

where i is the indoor temperature, o is the outdoor temperature and $t_1, t_2, \ldots, t_n$ are the n latest temperature values retreived: 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.



Setup (data proxy)

ESP32)

Indoor DHT sampling rate : 3 sec.

Outdoor DHT sampling rate : 20 sec.

Data acquisition process)

Latest temperatures request : every 5 sec.

Mean network latency evaluation : after 1 hour.

Setup (data analytics)

Forecast)

Data obtained	:	on start.
Temperatures collection	:	past month (unevenly).
aggregateWindow	:	every 20 sec (mean function).
Num. of values retreived	:	6500 indoor, 6500 outdoor.
Forecast	:	every 10 min, for 6 times.

Alarm)

Data obtained	:	every 30 sec.
Temperatures collection	:	latest 2 min.
Alarm threshold (1)	:	0.03.