

Project report - Internet of things

Daniele Polidori
University of Bologna
daniele.polidori2@studio.unibo.it

I. INTRODUCTION

I implemented an IoT system that monitors the temperature of an house, to prevent useless HVAC consumption, detecting it by rapid temperature changes. Reducing energy usage has good environmental reasons and also saves money on the bills. In this report I show the components and the structure of the system I've realized. Then I show the experimental setup and the consequent results I've obtained; at the end, I present some thoughts on them.

II. PROJECT'S ARCHITECTURE

The system is composed by an ESP-WROOM-32 board linked to two DHT22 sensors (one placed inside the house and one outside) and to a LED.

The board periodically collects indoor and outdoor temperature values. They are constantly analysed: if the indoor temperature values rapidly change (going towards the values of outdoor ones) the LED is temporarily turned on, thus showing an alarm signal to the user. In this way, if the temperature change is caused by, e.g., an open window, you can save unnecessary HVAC waste. At the beginning, based on all past data, the system predicts some future temperature values.

The temperature data collected are continuously sent to a gateway, that stores them, together with the alarm triggers and the forecasted temperature values, on a local time-series database. All data are interactively visualized by a local web application, that shows them by means of charts.

III. PROJECT'S IMPLEMENTATION

A. Data acquisition

I made the `data_acquisition.ino` file to program the ESP32 board. I use the `Thing.CoAP` library to make the board act as a CoAP server and the `PubSubClient` library to make it act as a MQTT subscriber.

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

Through the MQTT protocol, the board can receive some commands: to start or stop the sensors reading (at the beginning they are off), to change the interval between consecutive sensors readings — in both cases you can decide for just one of the two sensors — and to turn on or off the LED. My laptop acts as a MQTT broker, through Mosquitto.

B. Data proxy

I made the `data_proxy.py` file to create a Python application. I use the `aiocoap` library to make the script act as a CoAP client, the `paho-mqtt` library to make it act as a MQTT publisher and the `influxdb-client` library to store data.

Through the MQTT protocol, initially, the application gives commands to the ESP32 to start the sensors reading and to sets their sampling rate.

Through the CoAP protocol, it periodically requests, to the board, the last indoor and the last outdoor temperature value collected. The script continuously stores these temperature values on a local InfluxDB instance.

The network latency, between the temperature value request (to the ESP32) and its reception, is continuously monitored; after a while, the application evaluates the mean latency of this process.

C. Data analytics

I made the `data_analytics.py` file to create a Python application. I use the `influxdb-client` library to get and store data on the database, the `prophet` library to forecast some future temperature values and the `paho-mqtt` library to make it act as a MQTT publisher.

... [and, when needed, it turns on (or off) the LED]

IV. RESULTS

...