Alma Mater Studiorum · University of Bologna

SCHOOL OF SCIENCE
Master's Degree Course in Computer Science

Smart pot project: technical report

Prof.:
Luciano Bononi
Marco Di Felice

Student:
Damiano Bellucci

Subject: Internet of Things

Contents

List of Figures

Chapter 1

Introduction

The goal of this project was to create an IT support to obtain information on the state of condition of pot plants or, more generally, of any land used for cultivation. To do this, heterogeneous technologies were used: hardware such as microcontroller and sensors, software to act as a bridge between hardware and the storing of the acquired data and software to predict the future conditions of some parameters and software for the frontend to show results.

The software can be used for a single smart pot and provides information on different parameters: air temperature and humidity, soil humidity and additional information such as the quality of the connection of the smart pot with the wifi, id and position of the smart pot. In addition to this, a further parameter is calculated which is the result of the interpolation of soil humidity and air temperature called SHI (health index) which gives information on the general state of the smart pot.

A 10 seconds forecasting support is provided for the soil temperature and humidity parameters.

Once all the parts of the project have been configured, a dashboard will be available where the information is summarized in the form of graphs.

The user can set some setting parameters such as data acquisition frequency, soil moisture and temperature range that will be used to check and notify with a blink led on the microcontroller if the sensor data are out of range.

The project is available at this link: https://github.com/damianobellucci/iot-project

Chapter 2

Project's Architecture

As shown in figure 2.1, the architecture of the project consists of three parts: the hardware part, the bridge and the frontend.

The hardware part takes care of data acquisition through sensors that will be published via the mqtt protocol on the broker, the bridge takes care of acquiring the data and saving them in the database. This data will then be shown in the frontend part, specifically in the dashboard.

The frontend includes a part (http client) in which it is possible to set the setting parameters of the microcontroller that will be requested (if the microcontroller is starting up) or sent to the microcontroller to be dynamically changed.

The forecast part in the bridge works independently by taking the data already saved in the database and writing the forecast data to the database as output.

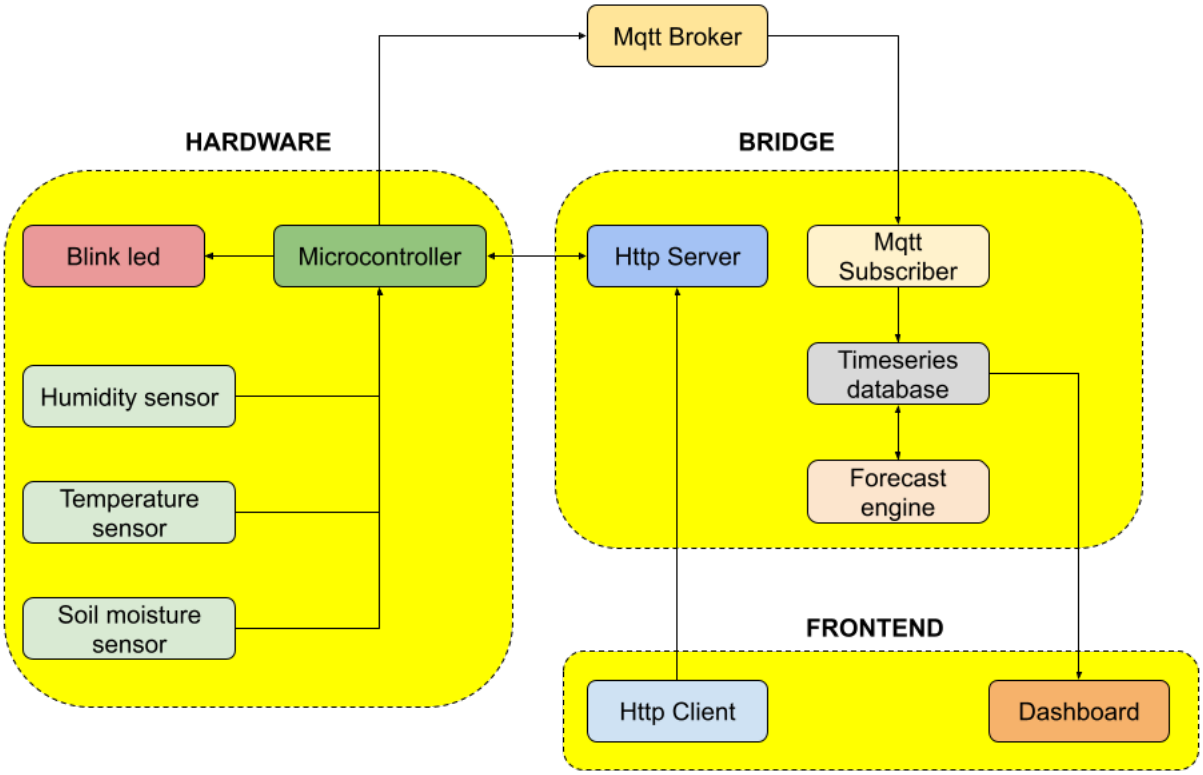

Figure 2.1: Project's architecture

Chapter 3

Project's Implementation

In this chapter we will talk about how the various parts of the project were implemented and why: protocols used for the communication of the various parts, software and libraries used, components and forecasting techniques adopted.

3.1 Communication between the parties

The main communication flows between the parties are two: the first is that between microcontroller and bridge and between client and bridge to set or update the setting parameters of the microcontroller, while the second is that which concerns the communication of the data of the microcontroller to the bridge.

3.1.1 Communication for setting parameters

The microntroller at the first start requests the setting parameters from the http server in the bridge, which exposes an endpoint through which it is possible to request the parameters through a GET.

When the request for the start parameters is successfully completed, the microcontroller is ready to start its work but the setting parameters can be changed at any time through a POST operation to the http server in the bridge, which in turn it will do a POST on an http server in the microcontroller which exposes a POST endpoint which will be used to change the setting parameters dynamically.

The HTTP protocol was chosen as being based on TCP it gives us guarantees on the successful reception of messages, which is fundamental in this case as the setting parameters strongly change the execution flow of the entire system, so it's not possible not to know whether the messages are received or not, at the cost of a greater waste of processing resources and transmitted bits, also justified by the fact that it is a one-off operation and therefore it is not a fixed cost.

3.1.2 Communication for raw data transfer

The mqtt protocol was used to transfer the raw data coming from the microcontroller sensors and characteristics of the microcontroller itself (gps coordinates, id) to the bridge. This is because the aforementioned data is a continuous flow with a frequency in the order of seconds, therefore mqtt guarantees an open TCP channel between publisher and broker (to which the subscriber will subscribe in the bridge) which allows to amortize the overhead cost of the individual communications. It also allows a communication 1 (publisher, the microcontroller) to n (the subscribers), so the microcontroller has to worry only about the communication with the broker avoiding to manage any n subscribers at its expense. It also provides mechanisms to ensure that messages have arrived at the broker with the quality of service setting.

3.2 Hardware

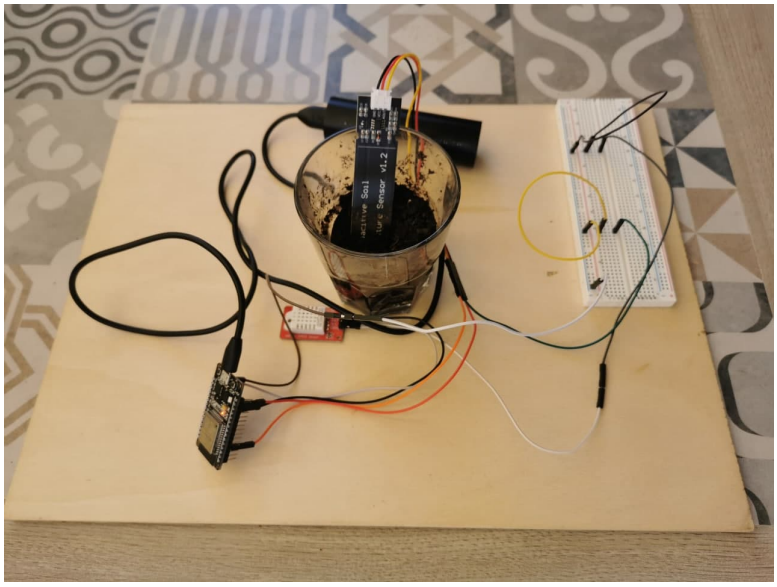

Figure 3.1: Hardware

3.2.1 Microcontroller

The device used as a microcontroller is the esp32 which adopts a wifi module with which it will communicate with the bridge. It also has an on-board LED that will be used to notify by flashing when the sensor values deviate from the range. In addition, this device has numerous libraries available on the Arduino platform through which it was possible

to use the mqtt, http, wifi protocols, as well as libraries for managing the sensors used in the project.

The microcontroller has been assigned hardcoded GPS coordinates and id, it has also been set to have a static IP to ensure that the bridge's http server knows its address at each restart.

The data are acquired according to the sample frequency in the setting parameters and are sent merged into a string, this to avoid having a communication for each sample rather than multiple communications for each single dal for each sample. When the data of the temperature and soil moisture sensors deviate from the range in addition to sending the data to the broker, the on-board led lights up.

3.2.2 Temperature and humidity sensor

The DHT 22 was used as a temperature and humidity sensor, which through libraries on the Arduino platform allows to easily obtain the data.

3.2.3 Soil moisture sensor

A capacitive sensor was used as a soil moisture sensor which gives information on soil moisture. For this sensor there were no libraries, so the information was obtained directly from the analog output signal from the sensor and mapped on a percentage range.

3.3 Bridge

The bridge is interposed between the frontend and the hardware and is composed of several independent parts which are the database, the subscribe mqtt part for receiving data from the broker, the server that deals with interacting with the microcontroller for the parameters of setting and finally the forecast engine. These parts will be analyzed individually.

3.3.1 Http server

The http server, as already mentioned previously, exposes two endpoints concerning the configuration of the setting parameters:

- GET: endpoints that allows the microcontroller to set the initial setting parameters
- POST: endpoint that will be used to dynamically set the setting parameters. You can choose to set one or more parameters.

The server is implemented in node.js and must be launched before the microcontroller to start acquiring data. The server must be set on a static ip as it must always be reachable by the microcontroller.

3.3.2 Mqtt subscriber

The mqtt subscriber takes care of acquiring data from the broker, parsing, carrying out some preliminary checks on the data and entering them in the database with appropriate gps coordinate tags and microcontroller id.

The parsing consists in converting the merged data in the form of a string from the broker into separate data according to the type so as to enter them in the database. Before entering them in the database they will be checked if there are invalid values (null, etc ...) and in case they will be discarded. The script was developed in node.js.

3.3.3 Timeseries database

The database is a timeseries database that favors the work on data that is identified by a time stamp, in particular influxdb was used.

The timestamp to the data is assigned at the time of data entry by the mqtt subscriber.

Downsample od data with Task

A task has been created within the database that aggregates the data every 10 seconds by averaging them and entering them in another location in the database, thus keeping the original data with the original time stamp. This is done to minimize the data provided to the forecast engine and minimize fluctuations in the data due to sensors. The resulting data will then have a frequency of 10 seconds.

3.3.4 Forecast engine

The forecast engine is responsible for processing data for forecasts at 10 seconds of temperature and soil moisture.

To procude the forecasteed values was used Prophet, a library that allows to predict values of timeseries dataset in an automatic way with a certaint frequency (for this project was 10 seconds).

Both temperature and soil moisture prediction modules were developed in python.

Validation

The approach that was used for the validation was that of the walktrough: starting from a dataset split between training dataset and test dataset, a forecast is made for

the next value starting from the training dataset and every time it's done the prediction the training dataset expands to the right by one value, and so on until a number of predictions equal to the number of values in the test set are made. Finally, the RMSE is obtained by relating forecast values and test set values.

The validation dataset includes 300 values (samples frequency 10 seconds), of which 66% of training set and 34% of test set.

the models created with Prophet were validated and resulted with a good RMSE scores (0.092 for temperature and 0.466 soil moisture). This can be seen graphically in figure 3.2 for the temperature and in figure 3.3 for the soil moisture.

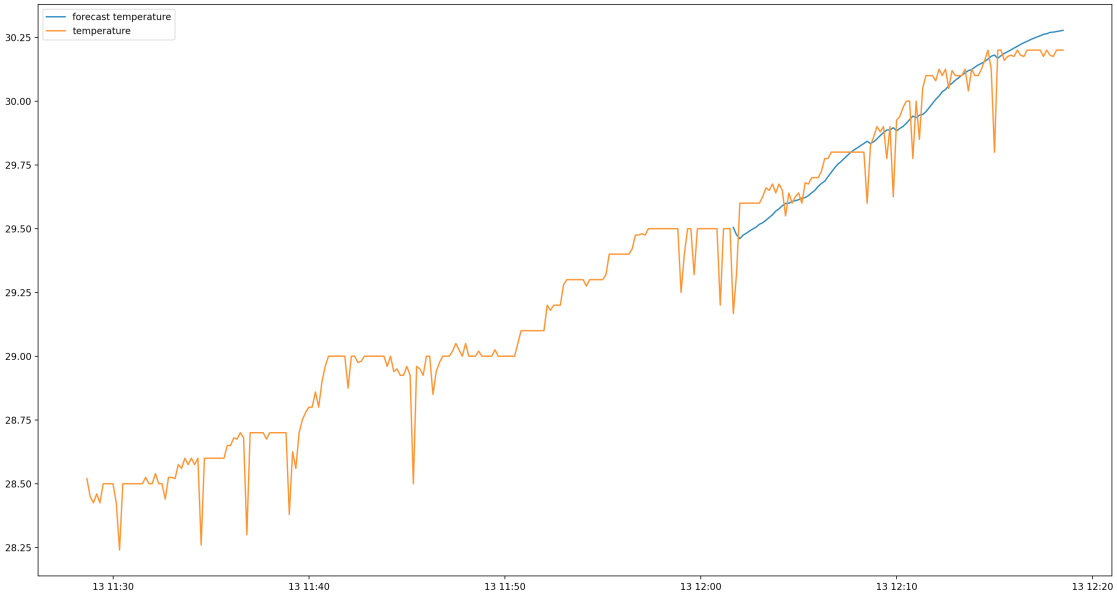

Figure 3.2: Forecast temperature

3.4 Frontend

The frontend part can be seen in two parts: the http client which is used to dynamically set the setting parameters and the dashboard for displaying real and forecast data.

3.4.1 Http client

From the client it is possible to set all the setting parameters through an http POST request. The possible values must respect some constraints inherent to the operation of

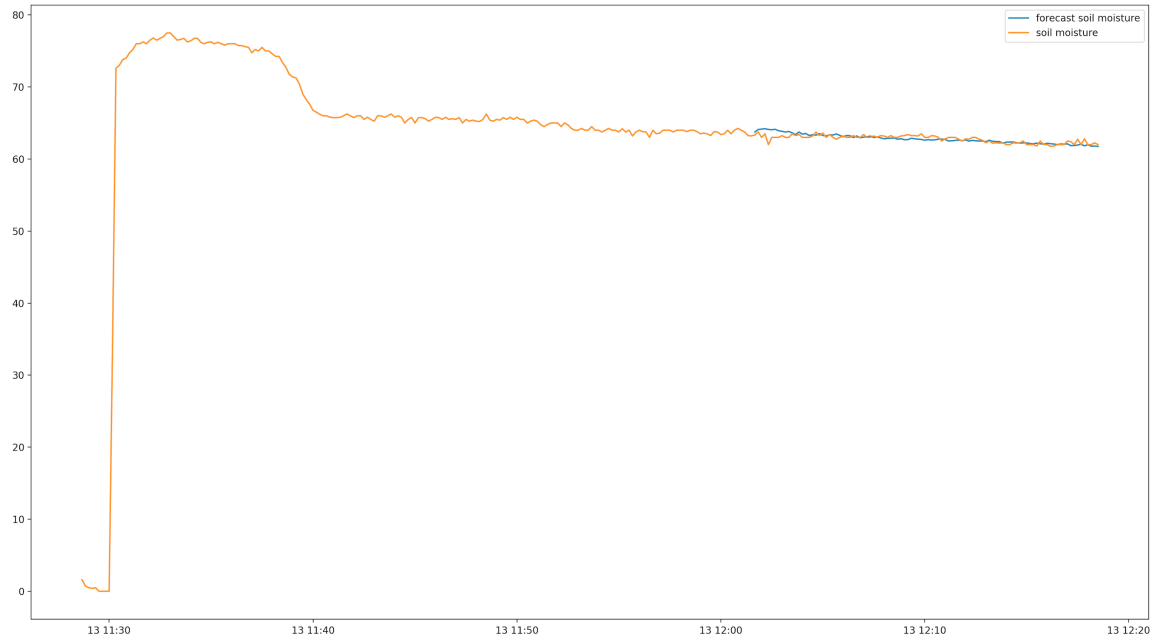

Figure 3.3: Forecast soil moisture

the project / syntactic / logical and if they are not respected, an error will be received in response and the parameters will not be updated. For example, for the frequency of the samples that cannot be greater than 5 seconds and not less than 1 second, as above 5 seconds there would be no aggregation of the data regarding the task in the influxdb database (aggregates the data at 10 seconds) and less than a second you would not notice the blink of the led if it were to turn on. In addition, beyond 5 seconds there would be a risk that the measurement would be too affected by the oscillations of the sensors.

3.4.2 Dashboard

The dashboard was created using the grafana software, which allows you to view the data from the influxdb database on graphs set at will. The dashboard includes a graph for each parameter and also in the temperature and soil moisture graphs there will also be displayed the data relating to the forecast at 10 seconds from the real data (example in figure 3.4).

Figure 3.4: Dashboard

Chapter 4

Results

The results obtained in this project were consistent with expectations and follow the original objective: to prototype an IT support for cultivation. The forecast of temperature and soil moisture responds well to 10 seconds, larger values would result in a divergence between actual and expected measurements, especially in the case of non-linear events such as sudden changes in the environmental state of the smart pot or random oscillations of the sensors due to external events. Furthermore, the 10-second temperature and soil moisture forecasting models have been validated obtaining a good RMSE score, which gives us an indication of how the model is able to make a good forecast.

The sampling frequency of the microcontroller is deliberately kept high (more than 1 second and less than 5 seconds) because the data would otherwise be affected too much by the oscillations of the sensors (especially the soil moisture sensor, which does not have a dedicated management library and therefore the value is obtained from a mapping on a percentage of the analog current values between a minimum and a maximum determined empirically (100 % sensor in the water, while 0 % sensor out of the water).

Possible future developments of the project can be to determine the successful irrigation, design a frontend from which to manage the setting parameters taking into account the user experience and also the possibility of a single user to manage multiple smart pots.