Report Project NETRF

IoT using MQTT

Achten Brecht, Fierens Tom

**Introduction**

This project involves the creation of an Internet of Things application using the MQTT protocol for communication. A thermistor and photoresistor will be used as sensors to measure the temperature and lighting at a specific location. This data will be transferred using the ESP32 to a Raspberry Pi 4. The MQTT protocol will be used for this communication through Wifi. The Raspberry Pi will process this data and store it in an SQL database, which can be accessed with a server. This application can be used to check from a distance if the environment for plants is ideal. An LED close to the sensors can be controlled from the server to alter the lighting. This project can be expanded by adding more sensors (for example for the pressure or humidity) and/or by adding more controlling instruments (for example to regulate the temperature at the sensor location).

**The setup**

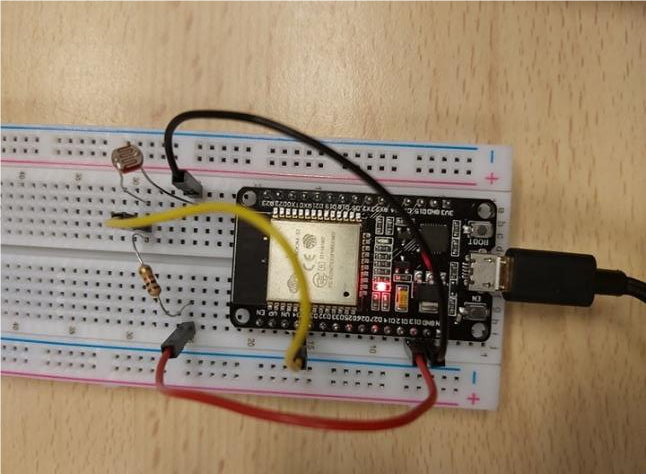
1. **Raspberry Pi 4**

The main component that will be used is the Raspberry Pi. The Raspbian GNU/Linux 10 operating system was installed.



1. **ESP32 module**

The second important component is an ESP32 module. This will be connected to the sensors to gather the data and transmit it to the Raspberry Pi through Wifi.



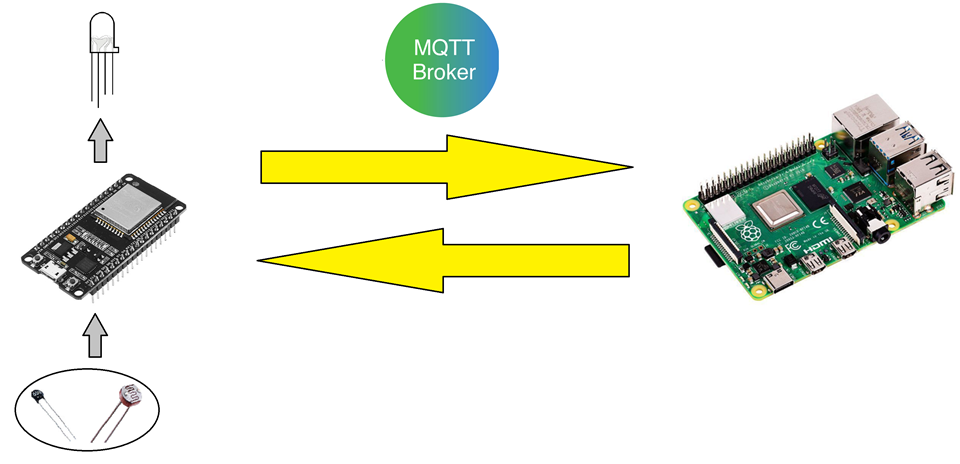
1. **The sensors**

We are using a thermistor and photoresistor for this project, but any type of sensor can be used or added for extra functionality.



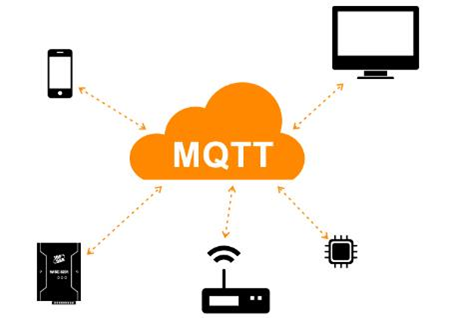
**Structure and components**

The ESP32 will be mounted onto a breadboard and will be connected with the sensors. The readings of the sensors will be transmitted to the Raspberry Pi using a Wifi connection. As previously mentioned, the MQTT protocol will be used for the exchange of information. The Raspberry Pi has a variety of functionalities. It will store the data in an SQL database, which can be accessed through a server created using the Spring framework. As an example, the user can control an LED from the server to alter the light values, but many expansions can be made.



**The MQTT protocol**

In the MQTT protocol, both sender and receiver communicate with an MQTT broker. This broker acts as a central mediator, the communicating devices are not directly connected. The device that wants to send data, it uses a *publish* type message. When another device wants to receive data of a certain topic, it lets the broker know with a *subscribe* command. The advantage of this protocol is that it is very lightweight and ideal for simple sensor communication. In this project, the Eclipse Mosquitto MQTT broker will be used [1].



The ESP32 first connects to the WiFi network. This is done using the library WiFi.h. The program waits until it is connected to the WiFi. If the connection is lost during the program than it will try to reconnect and wait until it is connected again. This is all done in the function connect\_WiFi().

After the ESP32 is connected to the WiFi it will try to connnect to the MQTT broker this is done using the PubSubClient.h and the program will wait until it is connected to the broker.

After that the program is connected to the broker it will check if it already has done a successful setup. This is done by reading values that are stored in the non-volatile memory using the EEPROM.h library. It will store 2 values, a 16 bit number that is the thing\_id and a boolean value that is true if the setup has been completed.

If the device has not yet been setup successfully than it will publish on the topic with the name: “iot/setup/{thingId}” where {thingId} is a random 16 bit number. Then it will wait for an answer from the server. If the id is not yet being used than it will receive an “OK” message and it will use this id from now on. If it receives a “NOK” message then it will go through the setup again and generate a new random number.

Once the thing is set up, the program will start sending the values. This is done in the function sendValues(), first it checks if it is still connected to the WiFi and MQTT broker and than it will read the sensor values and send them to their respective topics. Every sensor gets it own topic with the name: “iot/data/{thingId)/{sensor}”, where {thingId} is the thingId generated during the setup process and {sensor} is an id for the sensor that can be the same for multiple things.

**MySQL database**

MySQL will be used to establish an SQL database. The data that will be stored consists of 5 columns: ID, Thing\_id, Value, Time and sensor\_id. The meaning of each column will be explained below:

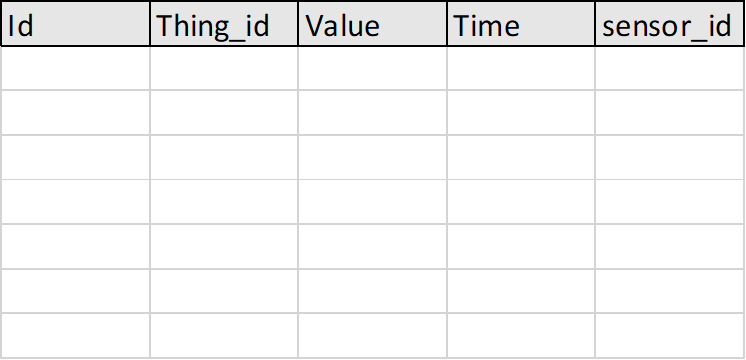
Id: Indicates the id of the transmitted data

Thing\_id: The id of the device

Value: The measured value

Time: Indicates the time of measurement

Sensor\_id: Indicates which sensor transmitted the data



**Server using the Spring framework**

To create the REST-API, the Spring framework will be used. Developing a server in Spring can be done in Java. The Spring framework makes it easier to communicate with a database, so it is ideal for this application [2].

The server uses “Spring Boot Web Starter” and “Spring Data JPA”. The “Spring Boot Web Starter” is used for the easy creation of REST API’s and the “Spring Data JPA” allows for easy access to the database. We have one REST controller named IOTResource and two repositories named ThingRepository and ValueRepository.

**Receiving and storing the data on the Raspberry Pi**

Hier wat uitleg en een foto van de code in Python waar de data ingelezen wordt op de raspberry en ook naar de database wordt weggeschreven.

**References**

[1] “MQTT voor IoT: betrouwbaar protocol voor data-uitwisseling - c’t.” [Online]. Available: https://www.ct.nl/achtergrond/iot-protocol-mqtt-betrouwbaar-data/. [Accessed: 11-Dec-2019].

[2] “Spring.” [Online]. Available: https://spring.io/. [Accessed: 11-Dec-2019].