Jarno Mattila

EMBEDDED SYSTEMS DEVELOPMENT PROJECT – IOT THERMOMETER

EXCERCISE REPORT

Jarno Mattila

Project Report

Autumn 2017

Degree Programme in Information Technology

Oulu University Of Applied Sciences

Table of contents

About the project

The objective of the project is to build a temperature measurement device that communicates with a remote system by IoT protocol MQTT or CoAP. Main components of the device are Raspberry Pi 3 singe board computer and Dallas DS18B20 temperature sensor.

Preparing the project

In the beginning of the projcet I subscribed on Trello and created a board for the project.

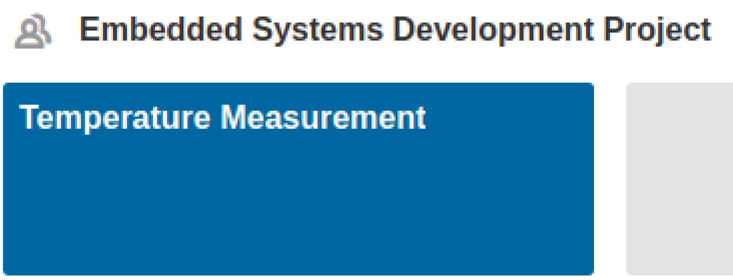

Figure 1: Project board on Trello

Trello is new to me, so I'd had to spend few hour to get start with it. Finally, I managed to create Backlog and other cards required in this project.

I also created a new Github repository to handle version control and share the files of the project. The location of the project repository is:

https://github.com/t5maja03/Embedded1

After I had both management platforms ready for the project, I invited teacher to my Trello board.

Assembly

The objective of this project is to build a temperature measurement device that communicates with a remote system by IoT protocol MQTT or CoAP. Main components of the device are Raspberry Pi 3 singe board computer and Dallas DS18B20 temperature sensor.
The assembly contains Raspberry Pi 3 board, Dallas DS18B20 sensor, 4,7kΩ resistor, breadboard and connection wires.
The assemply uses 3.3V voltage and GPIO-pin to read sensor value. In this assembly, Raspberry Pi is connectet to local network by ethernet interface and it is controlled remotely by secure shell connection (ssh).

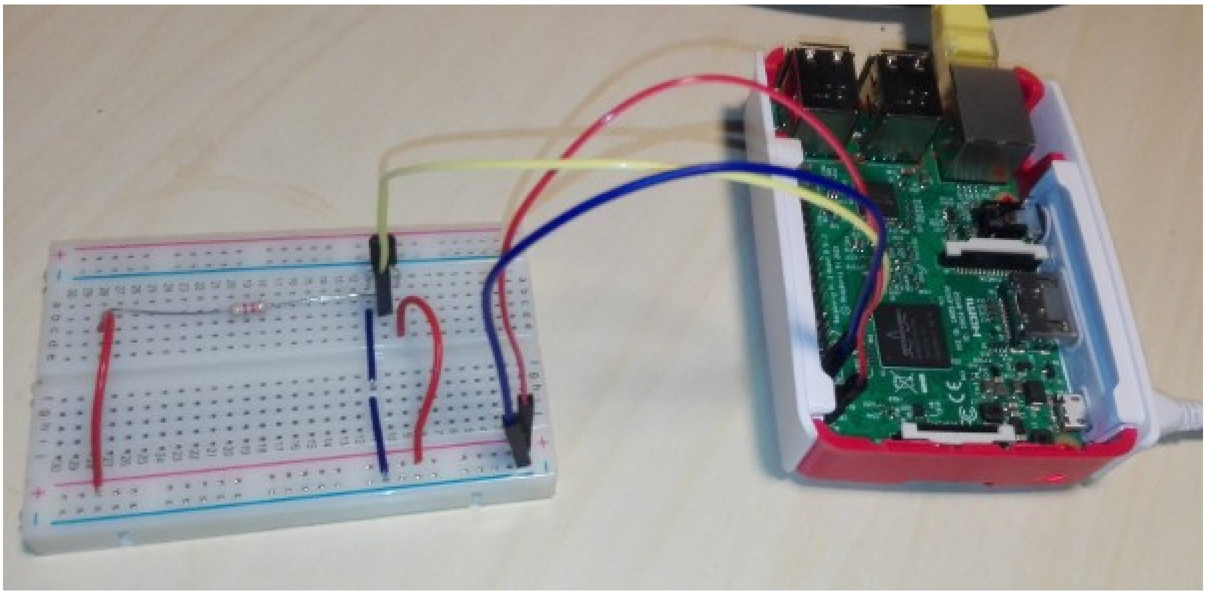

Figure 2: Raspberry Pi 3, Dallas DS18B20 connected using GPIO pin, 3.3V voltage and 4.7 kΩ resistor

Preparing Raspberry PI

Raspberry Pi 3 uses a mini-SD card as a bootable hard drive which in this case wasn't bootable at all. So, the first task was to initialize the disk with reliable operating system and drivers.

I formatted the disk by SD Card Formatter:

https://www.sdcard.org/downloads/formatter_4/

The OS installation was made by Noobs

https://www.raspberrypi.org/downloads/noobs/

The Operating system's Debian based Raspian Linux

http://www.raspbian.org/

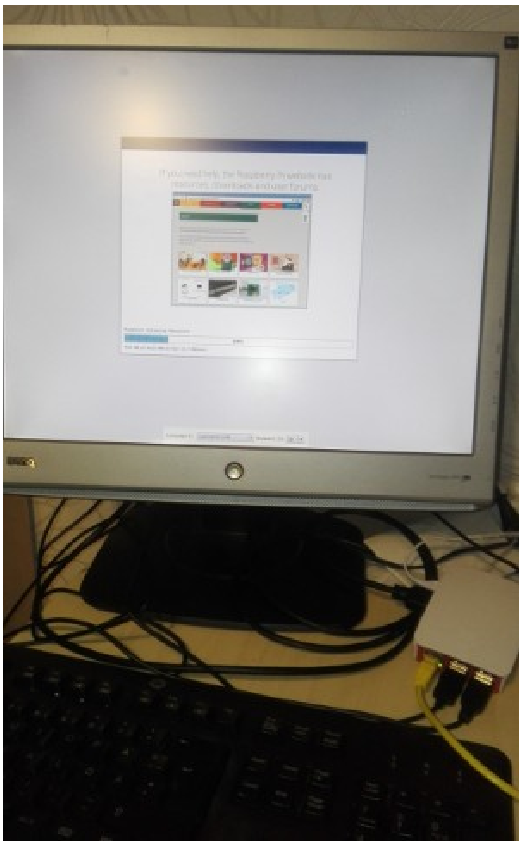

Figure 3: Installing Raspian OS by using Noobs

Testing components and assembly

I tested components and assembly by using simple python script found on this site:

https://www.modmypi.com/blog/ds18b20-one-wire-digital-temperature-sensor-and-the-raspberry-pi

The test script simply reads and parses a file in sensor device folder. Then it converts the data to Celsius and Farenheits and prints the result out. To be able to do that, it is necessary to import OS interface and system time, as well as the drivers for GPIO and 1-wire devices.

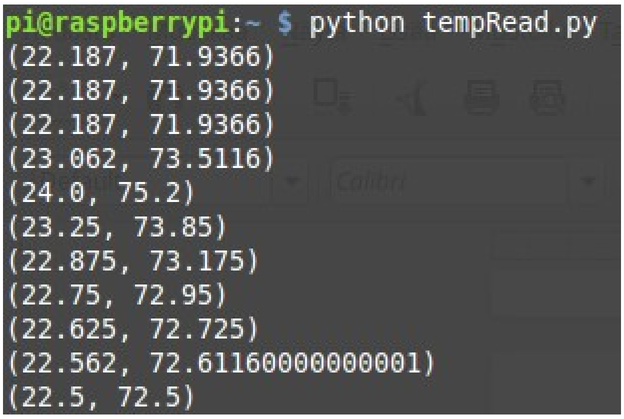

Figure 4: Testing assembly in bash console. The output is temperature in Celsius and Farenheit degrees

I wrote the script and run it in bash console. I also warmed the sensor by holding it in my hand to see that it responds correctly for changing temperature.

The test result shows that the assembly works properly and it is ready for further development as an IoT thermometer.

Implementing MQTT on Raspberry Pi

Implementinin MQTT on Raspberry PI Raspian, or almost any other Linux distribution, is straightforward. Installing mosquitto MQTT package and mosquitto_clients package. This can be done by package manager (as root).

```
#apt-get install mosquitto mosquitto_clients
```

Mosquitto has very well commented example conf file and it runs smoothly with default values as well. So, I could test MQTT immediately after installing packages. However, first I had to enable and start the service:

```
#systemctl enable mosquitto.service
#systemctl start mosquitto.service.
```

Enabling service means that it is starts automatically during reboot.

Now everything is ready for testing. I have a remote computer where I have installed mosquitto_clients as well. On this computer I say:

```
#mosquito_sub  -t testi -h 192.168.1.110
```

This means I subscript a topic named ”testi” on Raspberry Pi.

After that, on Raspberry Pi I say:

```
#mosquitto_pub -t testi -m ”Testing...”
```

which means I publish a message ”Testing...” to topic ”testi”.

Result screenshots are shown below:

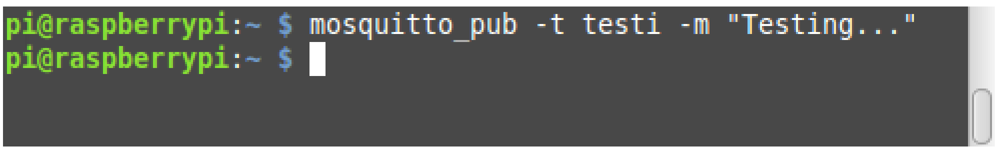

Figure 5: Publishing MQTT message from Raspberry Pi

Figure 6: Receiving MQTT message on remote computer

Creating a MQTT driver for testing

After implementing MQTT I updated my python script as a MQTT driver, so that I could read the sensor value and publish it immediately after reading. I replaced the print function with call function, so that I can run mosquitto_pub inside the python script:

```
    # print out
    # values rounded to 2 decimals and converted to string
    subprocess.call(["/usr/bin/mosquitto_pub",
                     "-t", "testi",
                     "-h", "mqtt.datakolmio.net",
                     "-p", "1883",
                     "-u", "t5maja03",
                     "-P", "tvt15smo",
                     "-m", str(round(mqttstring[0],2))+"  C "+
str(round(mqttstring[1],2))+"F"])
```

I also decided to create a real IoT enviroment including remote server in public internet. I installed mosquitto on server (CentOS7) and added some security by username and password.

The server is behind a firewall so I had to add MQTT in ACL and create port forwarding in NAT.

When everything was ready, I run python script on Raspberry Pi and read the messages from subscriber computer. Screenshots below

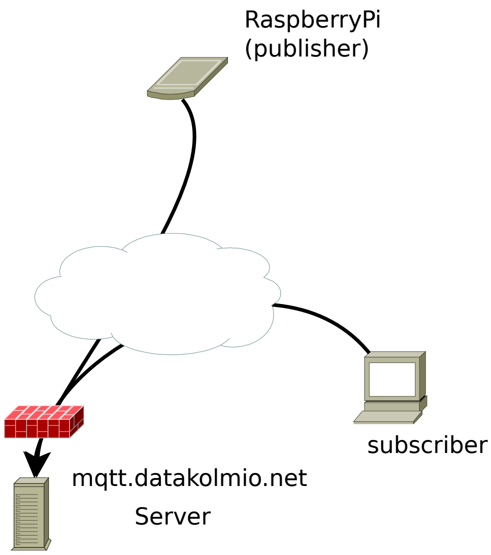

Figure 7: Network topology

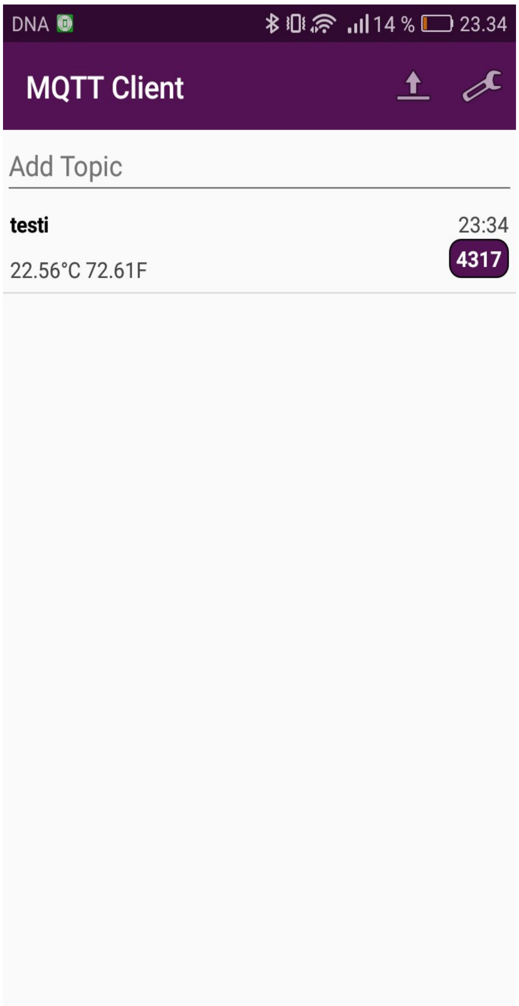

Figure 8: Reading temperature from Raspberry Pi by phone

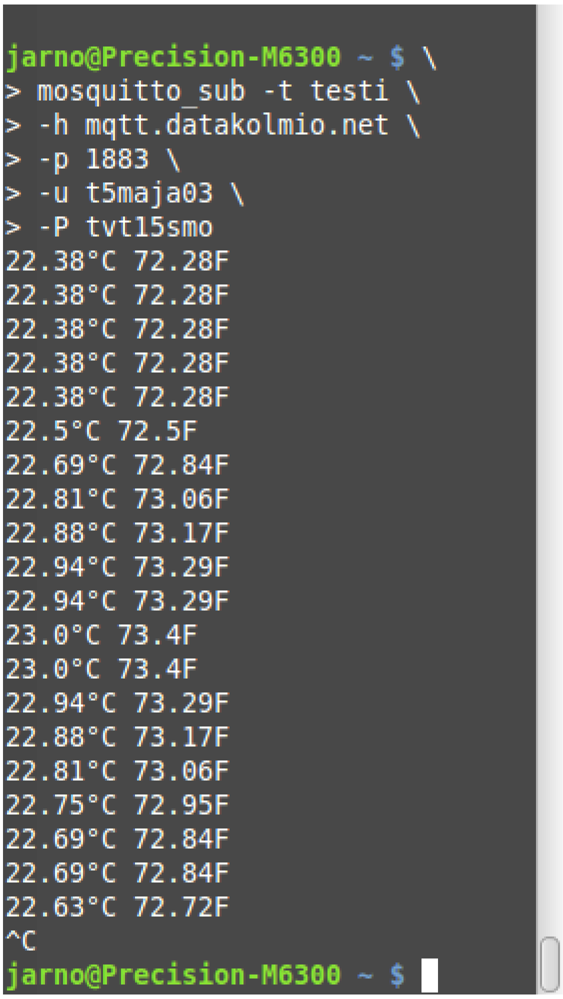

Figure 9: Reading temperature from Raspberry Pi by remote computer

The driver test shows that I can read temperature value from remote computer. There is 1 sec. polling time between the values and the sensor value changes if I warm the sensor by my hand.

Server works with other MQTT devices as well. Image 8 is a screenshot from my Android phone where I run basic MQTT client, while image 9 is a screenshot from my PC.

Web UI with graphical temperature diagram

After successful driver test I was ready to build up a web UI. My goal was to create graphical UI where I show the current temperature as a decimal value with two digits. Additionally, I want to show temperature history as a graphical diagram. The current temperature value and the diagram should update itself every time the new temperature value is received from publisher via websocket.

In this project, I don't want to save temperature data permanently, for instance a database or a file, but keep the values save as long as UI is active. So, closing web browser will wipe out all the data. I also want to control the amout of data that is saved during session, so that if user leaves the UI active for hours or even days, the UI can wipe the oldest data out of history automatically.

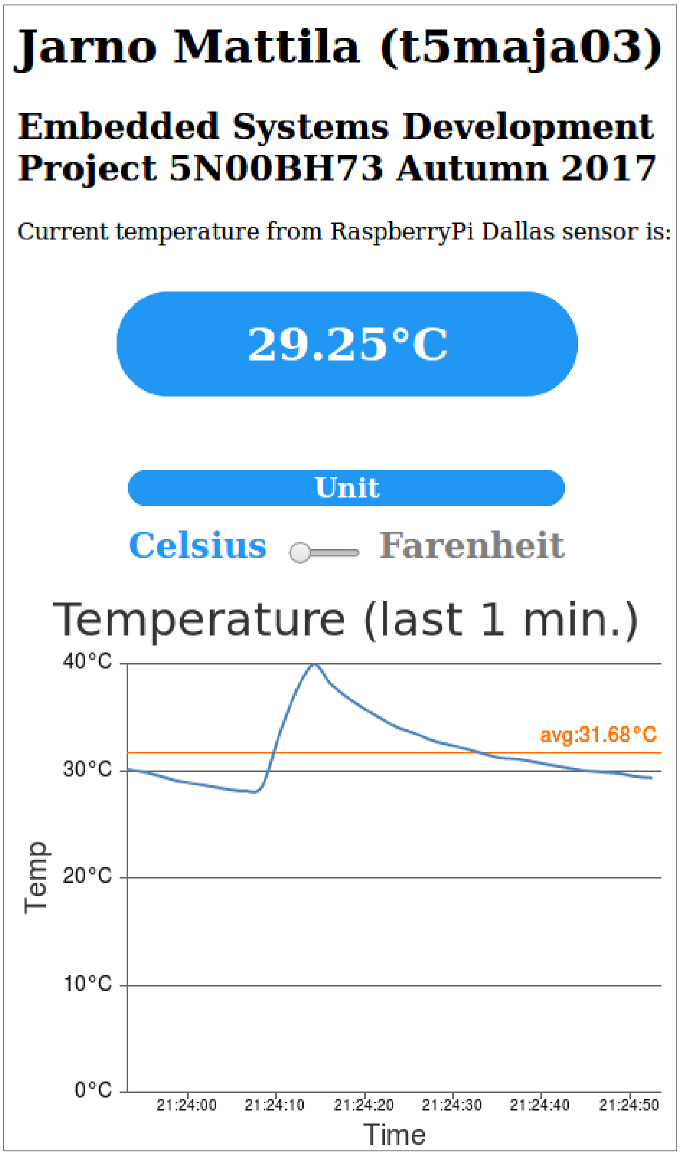

Figure 10: Web User interface located on http://www1.datakolmio.net/mqtt/

WEBSOCKET

As Mosquitto library includes websocket listener interface, I just needed to activate it as an extra listener on the MQTT server. This requires a change in Mosquitto configuration. By default, Mosquitto listens all network interfaces so I don't need to bind any IP for the listener, just port is enough:

```
listener 9001
```

Naturally, the websocet port must be forwarded through NAT and firewall as well.

On the client side of websocket I used Eclipse Paho JavaScript Client library which includes MQTT websocket support, and a nice sample script made by Jan-Piet Mens. Chart was made by CanvasJS Chart Library.

Urls of these sources are:

- Eclipse Paho JavaScript Client
  https://www.eclipse.org/paho/clients/js/
- Jan-Piet Mens' site
  http://jpmens.net/2014/07/03/the-mosquitto-mqtt-broker-gets-websockets-support/
- CanvasJS Chart Library.
  https://canvasjs.com/

IMPLEMENTATION

The UI is simple one view web page, written by HTML5. The layout styles are in single CSS-file. All functionality is made by Javascript in the same html file, except config.js which can be in different location for better security. I made the UI responsive by viewport meta declaration and keeping css relative.

The UI includes three functionality:

- Current Temperature with 1 sec updating period

- Last 1 min temperature history and agerave temperature as a graphical chart
- Unit switch for switching between Celsius and Farenheit.

Here is the address of the UI: http://www1.datakolmio.net/mqtt/

Conclusion

The goal of this project was to create IoT thermometer with web UI by using Raspberry Pi3, Dallas DS18B20 1-wire temperature sensor and MQTT protocol. The hardware part of the project was very simple, and so was the sensor reading by python. I expected some difficulties with MQTT because it's new to me. However, I found very good MQTT implementation for Linux named Mosquitto. So, the hardest part of the project actually was very simple too.

I'm rather familiar with Linux servers, especially Apache web servers, so I had a platform for my web site in no time. I spend most of my time with javascript to get all the functionality I wanted for my UI.

For me, advangate of this project was to learn MQTT basics and learn to implement it on Linux platform. I also learnt how to use new libraries and technics to create IoT implementation based on MQTT.

The biggest challenge was a project management with Trello.com and Kanban, which were new technics to me. The task planning and time tracking were rather clumsy while the project working with the hardware and software went on quite smoothly. Anyway, I think I got some kind of clue about project management by agile Kanban method and Trello.com project management tool as well.