Laboratory Data Networks

The goal of the laboratory is to build a small IoT architecture allowing to monitor the plugs and control them from a simulated cloud interface. This laboratory will span multiple sessions.

Your job is made up of two main two parts:

1. develop a program hosted by a first laptop, to interface your smart plug and the cloud, and
2. develop another program hosted by a second laptop, to to visualize your data and control all the plugs.

In this laboratory, each group dispose of a smart plug with an interruptible load connected behind it. The smart plug is connected to a WiFi network through a wireless router that can be accessed with the following parameters:

Wireless network: labo-EnInf WiFi password: 217665e13ebab7e870add29c75

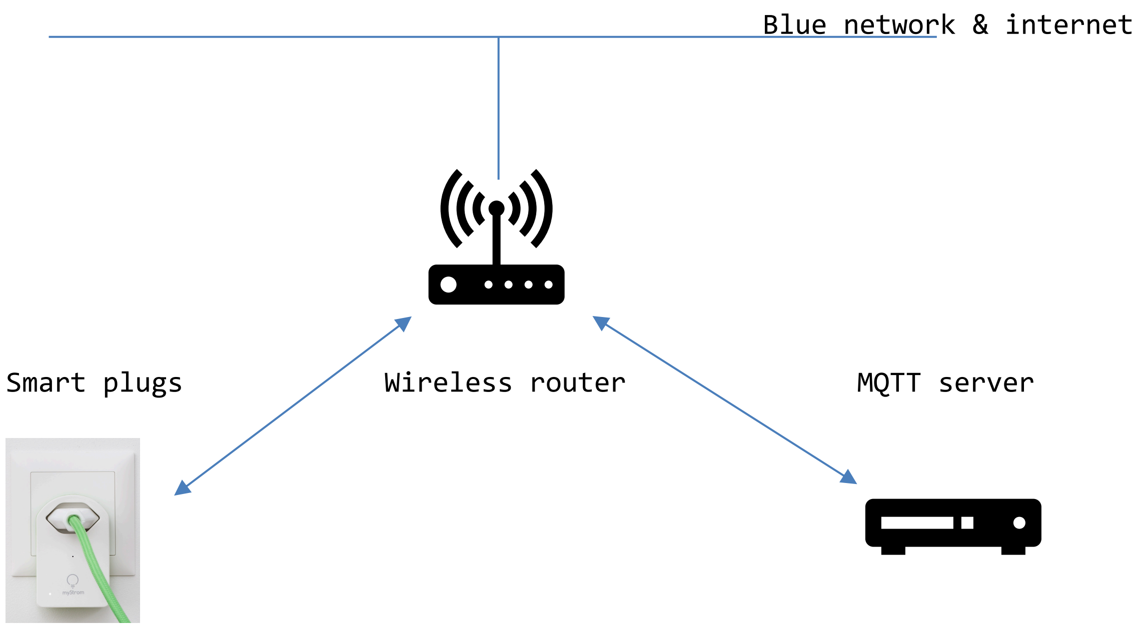

Figure 1 Overview of the lab's installation

Part 1 - Install and test your "myStrom Switch" smart plug (session 1)

A "myStrom WiFi Switch" smart plug (https://mystrom.ch/fr/wifi-switch/):

- measures the consumed electrical power drawn from the plug,
- turns on/off a relay to enable/disable power consumption when requested, and
- provides an estimated value of the ambient temperature.

A myStrom plug is connected on the WiFi access point "labo-EnInf". It has a static IP address distributed by the router:

Plug's MAC address	IP Address
08:3a:f2:72:c8:bc	192.168.0.150
08:3a:f2:a6:6f:84	192.168.0.151
f0:08:d1:c6:7b:68	192.168.0.152
84:0d:8e:3d:1a:14	192.168.0.153

You can find the plug's MAC address on the back of the plug.

The "REST API" mode" of the myStrom WiFi Switch provides a so-called "REST API", concretely an integrated HTTP (web) server bound to local port 80 that allows fetching the current power, ambient temperature and relay state, and also to set the relay state.

URLs and their roles are specified in the myStrom documentation available at:
https://mystrom.ch/wp-content/uploads/REST_API_WSE-11.txt.

Assignment (to finish before session 2):

1. Use a web browser to read the current consumed power, ambient temperature and relay state.
2. Use a web browser to change the relay state.
3. Use the `curl` or `wget` command to perform the same operations.

Part 2 - Access your myStrom smart plug in Python (session 1)

In this task, you will develop an initial version of the `MyStromSwitch` python program on your laptop. The program will implement the access to your smart plug using HTTP. As a base for your code, you can use the example file *RelayControl.py* provided on Cyberlearn. **Take care! all plugs are not necessary the same generation/firmware and could not answer exactly the same way as in the documentation.**

Assignment (to finish before session 2):

1. Install the `requests` package to handle http requests
2. You are free to fix the arguments and return type of the methods. Choose wisely.
3. Develop and test the `read_status()` method.
4. Develop and test the `read_temperature()` method.
5. Develop and test the `set_switch()` method.
6. Create a small program that:
    a. reads the status and print them
    b. toggles the switch
    c. wait 10 seconds
    d. returns to a.

Part 3 - Interface the Python program to the cloud

Your program will serve as gateway to the cloud, i.e.:

- all measured values are pushed to the cloud right after measurement, and
- the cloud may request opening / closing the relay; such an order must be executed immediately.

For this purpose, you will make use of the MQTT based platform of the lab. You will have to propose 3 topics and their body as summarized in Table 1. Choose a topic structure that will allow multiple plugs to be identified without ambiguity. Choose message body to be as most universal as possible.

Message topic	**Direction**	**Description**	**Sample message body**
`Your proposition`	laptop to cloud	Instantaneous power consumed by appliances behind the smart plug	`Your proposition`
`Your proposition`	laptop to cloud	Instantaneous temperature measured by the smart plug	`Your proposition`
`Your proposition`	Cloud to laptop	Relay command (“`open`”, “`close`”)	`Your proposition`

Table 1 MQTT topics

The coordinates to access the MQTT broker are:

Broker	`192.168.0.199`, port 1883 (default port for MQTT), the broker is completely open, no Username & Password.

Table 2 Coordinates of the MQTT broker

Assignment:

1. Make a proposition for the 3 different topics and message structure
2. Install the `paho.mqtt.client` package for MQTT programming.
3. Write a small python program to connect to the broker. Don't forget to call the method `loop_forever()` at the end, to initiate the consumption of MQTT messages.
4. Read periodically every second the power and the temperature on the smart plug and publish the corresponding messages (using `qos` 0).
5. Subscribe to the relay's topic and register a call-back method (for example `on_relay_set()`). Develop this method and link it to the smart plug's relay.
6. To validate your MQTT programming, install the MQTT Explorer program (http://mqtt-explorer.com). Use the same credentials to login on the broker. Subscribe to the smart plug's topic and display the received messages. Publish a message to change the state of the relay.
7. You can use the example file *MqttConnectTest.py* provided on Cyberlearn.

Part 4 - Readout the values and control the plug on another laptop

At this step of the project, your program on your laptop is running a permanent loop that:

- reads the status of the plug every 10 seconds and sends them to the cloud (MQTT)
- waits for a "Relay command" message and immediately opening/closing the relay.

All the visualization and control can be done manually through the MQTT explorer. If nobody accesses the MQTT, nothing happens. Now the aim of this step is to automate reading and control on another computer.

Assignment:

1. If not done on the other laptop, also install the `paho.mqtt.client` package for MQTT programming.
2. Instantiate a `Client` object and pass the appropriate parameters to its constructor to connect to the broker. Don't forget to call the method `loop_forever()` at the end of the constructor, to initiate the consumption of MQTT messages.
3. Develop and test the `on_status()` method callback.
4. Develop and test the `on_temperature()` method callback.
5. Develop and test the `set_switch()` method.
6. Create a small program that:
    a. reads the status and print them
    b. toggles the switch
    c. wait 10 seconds
    d. returns to a.