# Machine Learning for IoT

Homework 3

*****DUE DATE: 27 Jan (h23:59)*****

---

**Submission Instructions:**

Each group will send an e-mail to andrea.calimera@polito.it and valentino.peluso@polito.it (in cc) with subject *ML4IOT23 TeamN* (replace *N* with the team ID). Attached with the e-mail, a single ZIP archive (*.zip*) named *HW3_TeamN.zip* (replace *N* with the team ID) containing the following files:

1. The deliverables specified in the text of the exercises.
2. One-page pdf report, titled *TeamN_Homework3.pdf*, organized in different sections (one for each exercise). Each section should motivate the main adopted design choices.

Late messages, or messages not compliant with the above specs, will be automatically discarded.

## Exercise 1: Data Collection, Communication, and Storage (3 points)

### 1.1 Data Collection and Communication with the MQTT protocol (1pt)

In VS Code, develop an MQTT client that publishes information about the battery status (battery level and power plugged) of your PC every 1 second. The message must be sent in JSON format and must contain the parameters reported in the following table:

| Parameter | Description |
|---|---|
| mac_address | string, the MAC address of the PC network card. |
| timestamp | integer, the timestamp in milliseconds. |
| battery_level | integer, the battery level in percentage. |
| power_plugged | integer, a flag indicating whether the power supply is plugged (1) or not (0). |

**Example:**

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamp: 1664630309530,
  battery_level: 90,
  power_plugged: true,
}
```

Use the student ID of a member of your team as message topic (e.g., *s001122*).
Use the message broker provided by eclipse (mqtt.eclipseprojects.io at port 1883).

Run the script from the PCs of different team members (always use the same topic) to emulate a fleet of connected devices.

### 1.2 Data Storage (1pt)

In Deepnote, create a new Python notebook to develop an MQTT subscriber that receives the battery status of your PCs and store the received data in Redis. For data storage, define two Redis TimeSeries for each PC, called *mac_address:*battery and *mac_address:*power, respectively (e.g., *0xf0b61e0bfe09:battery* and *0xf0b61e0bfe09:power*), where *mac_address* is the MAC address of the corresponding PC (use *hex(uuid.getnode())*).

Run the notebook while collecting data with the script of 1.1.

### 1.3 Reporting (1pt)

In the report, explain why MQTT is a better choice than REST as the communication protocol for this application.

**Deliverables**

- A Python script named *publisher.py* that contains the code of 1.1. The code is intended to be run on a laptop and must use only the packages that get installed with *requirements.txt* provided during the labs. Moreover, the script should contain all the methods needed for its correct execution.
- A Python notebook named *subscriber.ipynb* that contains the code of 1.2. The code is intended to be run on Deepnote, must use only the packages that get installed with *requirements.txt* uploaded on the workspace and must run without any additional dependency.

## Exercise 2: Data Management & Visualization (3pts)

### 2.1 REST Server (1pt)

In Deepnote, create a new Python notebook to develop a REST Server that enables users to retrieve and manage the collected battery data. The API must be compliant with the following specifications:

**RESOURCES**

**Resource: BatteryStatus**

| Parameter | Description |
|---|---|
| mac_address | string, the MAC address of the PC network card |
| timestamps | list of integers, each integer is the timestamp in milliseconds |
| battery_levels | list of integers, each integer is the battery level in percentage |
| power_plugged | list of integers, each integer indicates whether the power supply is plugged (1) or not (0). |

Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamps: [1664630309530, 1664630310567, 1664630311542],
  battery_levels: [90, 90, 89],
  power_plugged: [1, 0, 0],
}
```

**ENDPOINTS**

**Endpoint /devices**

- ____ **/devices** (fill the form with the most appropriate HTTP method)
  Description: Retrieve the list of MAC addresses of the monitored devices.

  Path Parameters: N/A

  Query Parameters: N/A

  Response Status Code:
  - 200 – OK: Everything worked as expected.

Response Schema:

| Parameter | Description |
|---|---|
| mac_addresses | list of strings, each string is a MAC address. |

Response Example:

```
{
  mac_addresses: ["0xf0b61e0bfe09", "0xf0b41e2abe15"]
}
```

**Endpoint /device/{mac_address}**

- ____ **/device/{mac_address}** (fill the form with the most appropriate HTTP method)
  Description: Retrieve battery status information of the device with the specified MAC address in the specified time range.

  Path Parameters:

| Parameter | Description |
|---|---|
| mac_address | string (required), the MAC address of the device to retrieve. |

  Query Parameters:

| Parameter | Description |
|---|---|
| from | int (required), the start of the time range (timestamp in ms) in which to retrieve the battery status information. |
| to | int (required), the end of the time range (timestamp in ms) in which to retrieve the battery status information. |

  Response Status Code:
  - 200 – OK: Everything worked as expected.
  - 400 – Bad Request: missing MAC address.
  - 400 – Bad Request: missing start time.
  - 400 – Bad Request: missing end time.
  - 404 – Not Found: invalid MAC address.

  Response Schema: BatteryStatus

  Response Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamps: [1664630309530, 1664630310567, 1664630311542],
  battery_levels: [90, 90, 89],
  power_plugged: [1, 0, 0],
}
```

- **____ /device/{mac_address}** (fill the form with the most appropriate HTTP method)
  Description: Delete the timeseries associated to the specified MAC address.

  Path Parameters:

| Parameter | Description |
|---|---|
| mac_address | string (required), the MAC address of the device to delete. |

  Query Parameters: N/A

Response Status Code:

- 200 – OK: Everything worked as expected.
- 400 – Bad Request: missing MAC address.
- 404 – Not Found: invalid MAC address.

Response Schema: N/A

### 2.2 REST Client for Data Visualization (1pt)

In Deepnote, create a Python notebook to develop a REST client that sequentially executes the following actions:

a) Retrieve and print the list of monitored devices.
b) For each device, retrieve the battery status from a given time range. Plot the battery level over time on a line chart.
c) Delete the data of the last device in the device list.

### 2.3 Reporting (1pt)

In the report, fill the following table selecting the most suitable HTTP method (GET, POST, PUT, or DELETE) for each row and motivate your answer.

| Method | Endpoint | Description |
|---|---|---|
| | **/devices** | Retrieve the list of MAC addresses of the monitored devices. |
| | **/device/{mac_address}** | Retrieve battery status information of the device with the specified MAC address in the specified time range. |
| | **/device/{mac_address}** | Delete the timeseries associated to the specified MAC address. |

**Deliverables**

- A Python notebook named *rest_server.ipynb* that contains the code of 2.1. The code is intended to be run on Deepnote, must use only the packages that get installed with *requirements.txt* uploaded on the workspace and must run without any additional dependency.
- A Python notebook named *rest_client.ipynb* that contains the code of 2.2. The code is intended to be run on Deepnote, must use only the packages that get installed with *requirements.txt* uploaded on the workspace and must run without any additional dependency.