[neOCampus] IoT architecture

Dr Thiebolt François / IRIT [SEPIA, SMAC]
Goal

- to give users / applications access to **useful data** without hassle about networks, sensors technology or underlying embedded systems.

- High level of hardware details

  ```
  rpi3-u4-301.neocampus.univ-tlse3.fr
  ```

- Useful data

  ```
  If <conditions> and U4 / * / light < 400:
  U4 / * / lighting / command ← ON
  ```
General

- Infrastructure overview,
- MQTT topics | communication abstraction,
- Devices registration | sensO Campus web. app.,
- Sensors / actuators | the modules way,
- Backend | I/O abstraction for modules,
- The affluencesO Campus use case.
Infrastructure overview

- **Power meter**
  - 868MHz
  - Up to 6 CT

- **Ambient sensors**
  - Wide range of sensors: Temperature, Humidity, IR...

- **Decision agent**

- **Broker**

*At least, one Raspberry Pi per building in order for power meters to get reachable from its 868MHz link.*
neO Campus

Infrastructure overview

neO Campus network infrastructure & server

12 x SAS 300GB 15ktpm
2 x 6 Xeon E5-2609v3 @ 1.9GHz
64 GB DDR4 RAM

neocampus.univ-tlse3.fr

(4 x Ethernet Gigabit aggregated links)

Users storage space: 2.7TB RAID6
1.6GB/s raw read, 6500 iops read & 2200 iops write.

neO Campus Labs (U4 300,301,302)

IRIT

BU

neO Campus / CampusFab (U4 rdc)

Infrastructure overview

neO Campus network infrastructure & server

12 x SAS 300GB 15ktpm
2 x 6 Xeon E5-2609v3 @ 1.9GHz
64 GB DDR4 RAM

neocampus.univ-tlse3.fr

(4 x Ethernet Gigabit aggregated links)

Users storage space: 2.7TB RAID6
1.6GB/s raw read, 6500 iops read & 2200 iops write.

neO Campus Labs (U4 300,301,302)

IRIT

BU

neO Campus / CampusFab (U4 rdc)
Network automated install and configuration of our Raspberry Pi(s) :) 

- FAT32 formatted SD Card
- Unzip http://neocampus.univ-tlse3.fr/images/noobs.zip
  (custom NOOBS tailored to suit our needs)

- Download & install latest neOCampus-Raspbian
- Git retrieval of fully automated setup of the OS & application to deploy. 
  + works at home (depends on your internet provider)

Network automated installation of our customized Raspbian has been successfully tested on RPi, RPi2 and RPi3 :)

neocampus.univ-tlse3.fr
Only registered *devices* will obtain a valid IP address,

Wired / wireless neOCampus network is dedicated to IoT devices,

Raspberry Pi can be fully re-installed directly through network (PXE-like),

Raspbian-neOCampus (OS) latest images [http://neocampus.univ-tlse3.fr/images](http://neocampus.univ-tlse3.fr/images)

Per-device Raspbian customisation (specific hardware setup, application deployment …),

Only devices belonging to the same vlanID can communicate directly (i.e without MQTT),

SSH tunneled / VPN devices from abroad will gain access to the broker,

Near 40 RPi spreaded across our campus, tenths of ESP8266 (2015 - 2017)
General

- Infrastructure overview,
- MQTT topics | communication abstraction,
- *Devices* registration | sens*OC*ampus web. app.,
- Sensors / actuators | the *modules* way,
- *Backend* | I/O abstraction for modules,
- The affluences*OC*ampus use case.
Overview

End users applications, Ambient intelligence ...

MQTT broker

Node-Red

MongoDB

Database

MQTT topics

(real-time)

Software using sensors / actuators through MQTT.

Embedded software, low-level sensors / actuators (I2C, spi, …) providing real-time data through MQTT.

- Publish / subscribe paradigm,
- Complete abstraction of device's own location through topics,
- Topics can be seen as network unix pipes,
- Per-user security read, write or read'n write to topics,
- Real-time data exchange with various QoS,
- Callbacks managed by a thread loop,
- Bindings in Python (paho), C, Java, Ruby …
- Using Mosquitto v1.4.9 + auth_plugin → MQTT v3.1.1
  note: AMQP emulates MQTT protocol but without user security :(
- All topics are lower case (neOCampus convention),
- [Feb.16] Added support to websockets.
MQTT topics

- Multi-topics subscribing

  - u4 / # / temperature
    - multi-level subscribing (e.g. u4 / campusfab / temperature, u4 / hall / box1 / temperature)
  - u4 / + / temperature
    - single-level subscribing (e.g. u4 / campusfab / temperature, u4 / 301 / temperature)

- Single topic publishing

  MQTT specifications does not allow to publish to topics containing wildcards.

  The multiple method enables you to publish multiple data to multiple topics in a one-shot way.
MQTT topics

Real-time data exchange through TOPICs: the publish / subscribe paradigm

Since actuators initiate a TCP connection to the broker, they can be sent data back from the broker even when they are located behind a firewall (e.g Internet box).
### MQTT topics

#### Topics segmentation in neOCampus

<table>
<thead>
<tr>
<th>Base</th>
<th>type</th>
<th>[optional] command</th>
</tr>
</thead>
<tbody>
<tr>
<td>u4 / campusfab / shutter / command</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Base**: defined at *device* registration time according to location
  - e.g. u4 / 300 or bu / hall ...

- **Type**: kind of sensor / actuator (*module*) defined by sensOCampus or automagically detected
  - e.g. shutter, luminosity, temperature, sound, lighting ...

- **Command**: to send orders to a sensor / actuator (*module*)
  - e.g. orders to shutter like UP, STOP, DOWN
MQTT payloads are JSON frames

- Sending order to a shutter (with proper MQTT login / passwd)

  Order: "UP"
  Dest: "all"

  JSON frame as MQTT payload

- Then shutter publish its status back

  Order: "idle"
  unitID: "<shutter ID>"
  Status: "OPEN"

  JSON frame as MQTT payload

One caveat is that you can't send an order to a single module (shutter), hence the `dest` field.
def _mqttStatus(self):
    ''' send a json frame reflecting shutter's status '''

    jsonFrame = {
    'unitID' : str(self.unitID),
    'order' : '',
    if self._curCmd == __class__.SHUTTER_ACTION_CLOSE:
        jsonFrame['order'] = 'DOWN'
    elif self._curCmd == __class__.SHUTTER_ACTION_OPEN:
        jsonFrame['order'] = 'OPEN'
    elif self._curCmd == __class__.SHUTTER_ACTION_STOP:
        jsonFrame['order'] = 'STOP'
    elif self._curCmd == __class__.SHUTTER_ACTION_IDLE:
        jsonFrame['order'] = 'STOP'
    elif self._curCmd == __class__.SHUTTER_ACTION_UNKNOWN:
        jsonFrame['order'] = 'UNKNOWN'

    jsonFrame['status'] = 'CLOSED' if self.status==__class__.SHUTTER_POS_CLOSED else 'OPENED' if self.status==__class__.SHUTTER_POS_OPEN else 'UNKNOWN' if self.status==__class__.SHUTTER_POS_UNKNOWN else ''

    # send frame
    self._client.publish(self.MQTT_TOPIC, json.dumps(jsonFrame))
Sample MQTT publish / subscribe code snippet

```python
def main():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_publish = on_publish
    client.on_subscribe = on_subscribe
    client.username_pw_set(MQTT_USER, MQTT_PASSWD)

    # Start MQTT operations
    client.connect(MQTT_SERVER, MQTT_PORT, 60)
    client.loop_start()

    # Subscribe to topic
    client.subscribe(MQTT_SUB);

    # Acquire sensors and publish
    def publishSensors():
        # get CPU temperature (string)
        CPU_temp = getCPUtemperature()
        _fcputemp = float(CPU_temp) + random.uniform(-10, 10)
        CPU_temp = '{:.2f}'.format(_fcputemp)

        # Launch Acquisition & publish sensors till shutdown
        do_every(measure_interleave, publishSensors);

    main()
```

---

```python
# Acquire sensors and publish
def publishSensors():
    # get CPU temperature (string)
    CPU_temp = getCPUtemperature()
    _fcputemp = float(CPU_temp) + random.uniform(-10, 10)
    CPU_temp = '{:.2f}'.format(_fcputemp)

    # Launch Acquisition & publish sensors till shutdown
    do_every(measure_interleave, publishSensors);

    # The callback for when a PUBLISH message is received from the server.
    def on_message(client, userdata, msg):
        # code in this function ought to be threaded otherwise your app will get stuck until it is finished.
        payload = json.loads(msg.payload.decode('utf-8'))
        print("Temperature is \d.\d\s deg. %s" % (payload['temperature'],payload['unit']))

    on_message(
```
CampusFab / neOCampus' showroom use case

MQTT Topics

- u4 / campusfab / luminosity
- u4 / campusfab / temperature
- u4 / campusfab / shutter
- U4 / campusfab / shutter / command

http://neocampus.univ-tlse3.fr/domoticz/#/Floorplans
Summary

- Publish / subscribe paradigm to leverage our needs for all M2M communications,
- MQTT clients can even work behind a firewall,
- [HA] Cluster of MQTT brokers can behave as a single virtual broker,
- Almost unlimited size of messages (max. 256MB –defined @ compile-time),
- Thousands of thousands of messages per second (users'n app. wanted!),
- Per-user fine-grained security setup,
- Websocket support enabled,
- [TODO] let's encrypt support,
- ……
General

- Infrastructure overview,
- MQTT topics | communication abstraction,
- *Devices* registration | sensOCampus web. app.,
- Sensors / actuators | the *modules* way,
- *Backend* | I/O abstraction for modules,
- The affluencesOCampus use case.
A *device* is a physical embedded system connected to a network (eg. Raspberry Pi, ESP8266 ...).

**Device registration**

- Device registration | sensOCampus web. App. (Django)

**Diagram:**

1. Register device with position
2. sensOCampus
3. Add device to database
4. Add device to dhcpd.conf
5. DHCP
6. Add MQTT topic, user and passwd
7. MQTT
Device registration

... then registered device fetches its configuration from sensOCCampus

Device asks for its credentials

- MQTT_BROKER,
- MQTT_PORT,
- MQTT_BASE_TOPIC,
- MQTT_LOGIN
- MQTT_passwd*

Minimum configuration sent from sensOCCampus to a device
*MQTT passwd is only sent on first call (admin action required to create a new one otherwise)
Device registration

QR-code is a just a http link to a Django application managing sensors / actuators. GPS data will be read from browser.

Register device with position

http://sensocampus.univ-tlse3.fr/admin
?add_device=<macAddr>

http://sensocampus.univ-tlse3.fr/admin
?get_config=<macAddr>
Device's topics

- **I/O**
  - e.g. PiFace, Arbox etc

- **Device's topics**
  - **publish to**
    - **u4 / 301 / device**
  - **subscribe to**
    - **u4 / 301 / device / command**

- **Device publish its status**
  - `unitID: "<macAddr>"
  - `IO: "[ I / O list ]"
  - ...

- **Command to device**
  - `dest: "<macAddr>"
  - `order: "load_module'
  - `module: "shutter"
  - `shutter_ID: "<ID>"
  - `shutter_IO: [ 2, 3 ]`
CampusFab / neOCampus' showroom use case
Device's topics

✔ Publish status
I/O list,
Modules list,
Backends list,
Internal stuffs,

🔍 <mqtt_base> / device

✔ Subscribe to orders
Load'n setup modules,
Delete module,
Init backends list,
Add specific low-level backend,
Delete one specific backend,
Update code from git,

🔍 <mqtt_base> / device / command

A *device* API (python lib.) will be provided to the sens**OCampus** web. app.
Software architecture

Ambient intelligence app.

Distributed apps. that communicate mainly through MQTT. Able to ask device to load modules (sense OCampus otherwise). Interact with modules through MQTT.

Modules

- Shutter,
- Lighting,
- Luminosity (freq. 10''),
- SoundMeter (events),
- Temperature (freq. 1'),
- Camera, ……

Backend

Low-level drivers

- Common interfaces to all modules (e.g PiFaces, Arbox, Modbus)

Devices

Physical embedded systems connected to the network (e.g Raspberry Pi, ESP8266 etc)

Chip-level drivers (I2C, SPI), USB … (e.g TSL2561, MCP9808 ...)
There's no direct communication between ambient intelligent application and *devices*,

Several *devices* in a same room will share the same topic,

Both sensO Campus web. app. and device framework are a work in progress,

A python API will be provided to ease orders generation / status processing,

......
## General

- Infrastructure overview,
- MQTT topics | communication abstraction,
- *Devices* registration | sens**O**C**a**mpus web. app.,
- Sensors / actuators | the *modules* way,
- *Backend* | I/O abstraction for modules,
- The affluences**O**C**a**mpus use case.
A *module* is an abstraction of a sensor / actuator that hides underlying hardware complexity.

<table>
<thead>
<tr>
<th>MQTT Type</th>
<th>Sub-type</th>
<th>Orders</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>shutter</td>
<td>Wired or Wireless</td>
<td>UP, DOWN, STOP</td>
<td>CourseTime, outputs(Up, Down, Stop), ID ...</td>
</tr>
<tr>
<td>luminosity</td>
<td>numeric analog</td>
<td></td>
<td>Acquisition delay (default=10''), analog_range, ...</td>
</tr>
<tr>
<td>temperature</td>
<td>numeric analog</td>
<td></td>
<td>Acquisition delay (default=1'), analog_range, ...</td>
</tr>
<tr>
<td>camera</td>
<td>Motion-detect or RTSP</td>
<td></td>
<td>ID, rtsp_dest, ...</td>
</tr>
<tr>
<td>lighting</td>
<td>Telerupter, direct, DALI</td>
<td>TOGGLE, ON, OFF, xx% (DALI)</td>
<td>ID, output</td>
</tr>
</tbody>
</table>
Sensors / actuators as pluggables *modules* (i.e. dynamically loaded on-demand)

- **Dest:** `<macAddr>`
- **Order:** "load_module'
- **Module:** "shutter"
- **Shutter type:** "wired"
- **Shutter ID:** `<ID>`
- **Shutter IO:** [2, 3]
This is a slide from a presentation about modules in a system. The slide includes a diagram with flow arrows indicating the relationship between different components. The components include:

- **Dest**: `<macAddr>`
- **Order**: `load_module`
- **Module**: `temperature`
- **Thermo_type**: `analog`
- **Thermo_ID**: `<ID>`
- **Thermo_range**: [-30, +125]
- **Thermo_IO**: [5]

The components are connected with lines and arrows showing the direction of data flow and control flow. The components are labeled as:

- **Devices**
- **Modules**
- **Backend**
- **Low-level drivers**
- **libModules**

The **libModules** folder contains files such as `Backend.py`, `Temperature.py`, and `Camera.py`. The diagram also includes a note for a **Thermometer (0-10v)** with an arrow labeled `5 → temperature` pointing to an analog input (Analog IN).
Some kinds of *modules* can be automagically detected / instantiated.

- **TCN75A**
- **TSL2561**
- **MCP9808**

![Image of devices and modules]

- **Backend**
- **Low-level drivers**

**libModules**

- `Backend.py`
- `Temperature.py`
- `Luminosity.py`

- `I2c scan`
- `spi scan` 

**Automatically instantiated** *modules*:

- **u4 / 301 / luminosity**
- **u4 / 301 / temperature**

* `u4 / 301` is the MQTT_BASE_TOPIC associated to the device in this example.
CampusFab / neOCampus' showroom | shutter *module* use case

- CampusFab shutter *module*
- Hacked SOMFY remote controller

http://neocampus.univ-tlse3.fr/domoticz/#/Floorplans
Modules framework is kept simple (one file ← → one module) to ease auto-loading on purpose ==> easy to extend :)  
Several modules can share the same MQTT topic,  
Modules rely on backend abstraction of I/O,  
A module can be controlled from everywhere as long as you get access to the MQTT broker and your credentials are sufficient,  
Modules orders like START, STOP, FREQ, ENABLE, DISABLE … have not been shown for clarity,  
A python API will be provided to ease orders generation / status processing,
General

- Infrastructure overview,
- MQTT topics | communication abstraction,
- Devices registration | sensOCampus web. app.,
- Sensors / actuators | the modules way,
- Backend | I/O abstraction for modules,
- The affluencesOCampus use case.
General

- Infrastructure overview,
- MQTT topics | communication abstraction,
- Devices registration | sensOCampus web. app.,
- Sensors / actuators | the modules way,
- Backend | I/O abstraction for modules,
- The affluencesOCampus use case.
[neO Campus] Attendance & noise measurement @ BU

Sensors and actuators benefit from neO Campus infrastructure being integrated within!
noiseMeter registration @ sensOCampus

- Register noiseMeter


devices, sensors management from BU

- soundsensor3FC5 | …..
- soundsensor2BD6 | …..
noiseMeter sensitivity parameters directly from browser!
NoiseMeters live from browser!

Thanks to javascript and MQTT websockets → dynamic display of data from noiseMeters!
Links

- MQTT broker | neocampus.univ-tlse3.fr:1883
- Data flows | http://neocampus.univ-tlse3.fr/nodered
- UI | http://neocampus.univ-tlse3.fr/domoticz
FAQ

- Which git repository should I use?
- Docker? What is docker and why should I use it?
- How do I create virtual sensors? How to have them integrated within neoC campus arch.?
- How to gain access to the MQTT broker? when abroad?
- Is there a MQTT sandbox somewhere?
- I'm a dev. and I'd like to know where to launch my app.?
- How to gain read access to all data?
- Is there an archived version of all sensors somewhere?
- Where may I retrieve these slides → http://neocampus.univ-tlse3.fr
- Hey, I've been told the presentation ought to last half an hour and I'm still there?!?!
[2016, May 16^{th}] new Raspberry Pi zero v1.3 with camera support … $5 !!