[neOCampus] IoT architecture

Dr Thiebolt François / IRIT [SEPIA, SMAC]

UNIVERSITÉ TOULOUSE III PAUL SABATIER

Université de Toulouse

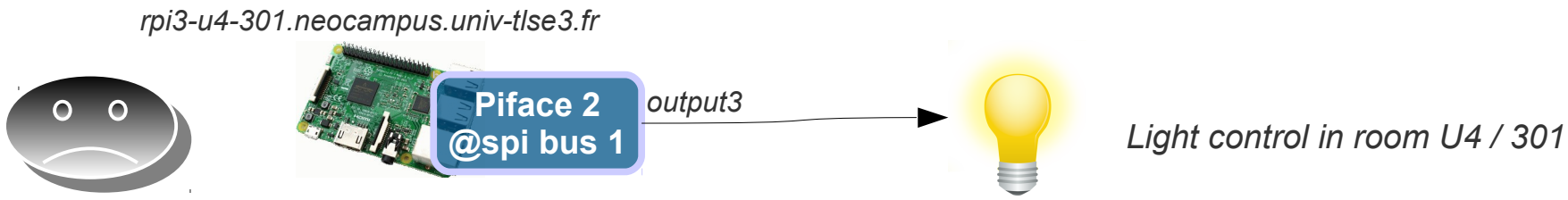
Université Paul Sabatier
Opération neOCampus

neOCampus

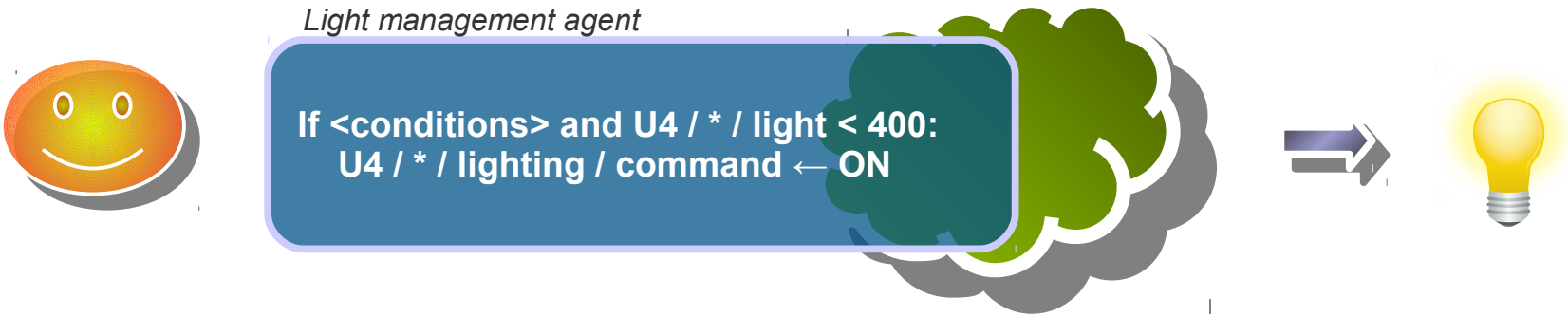
Goal

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

✖ High level of hardware details



✔ Useful data

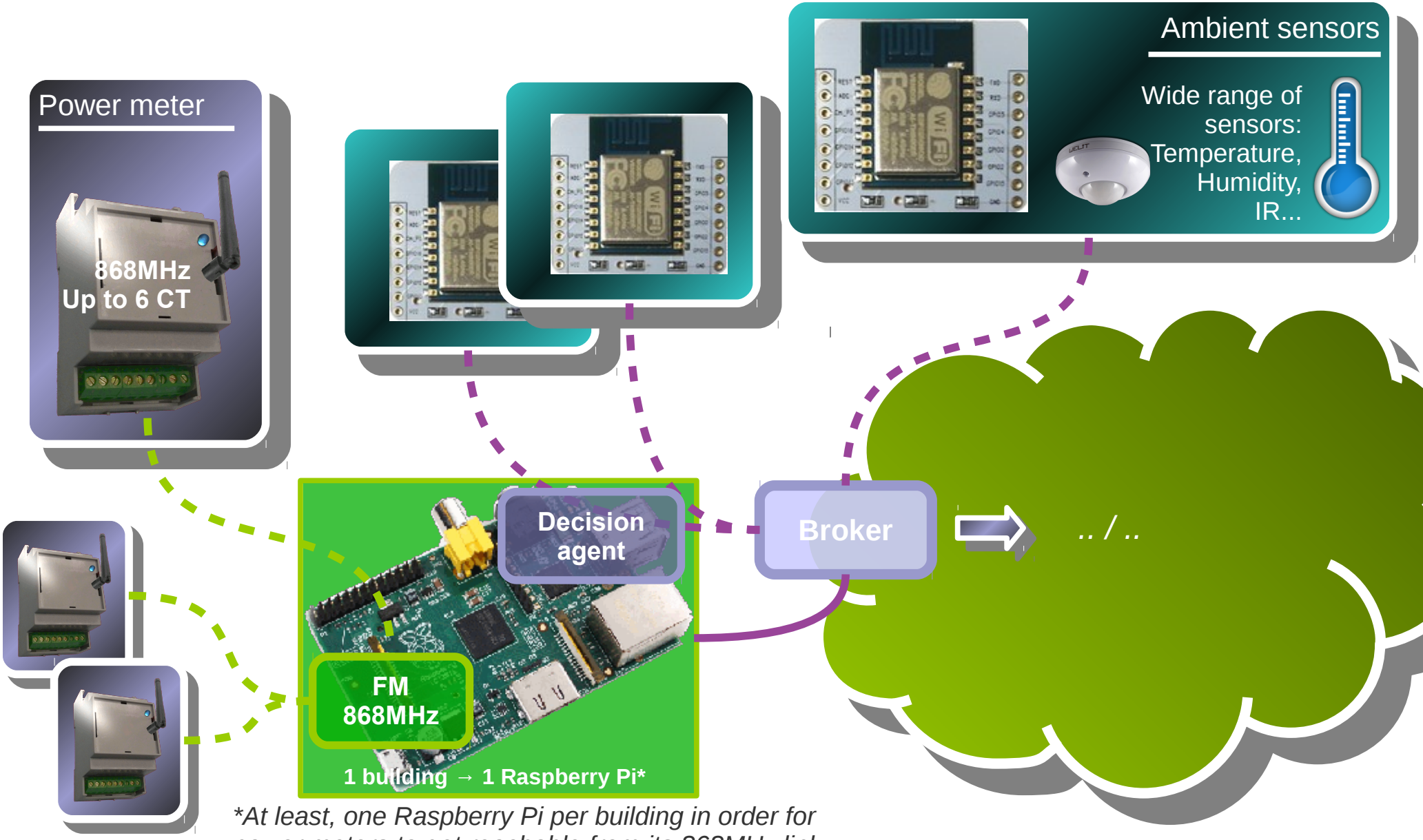


Plan

General

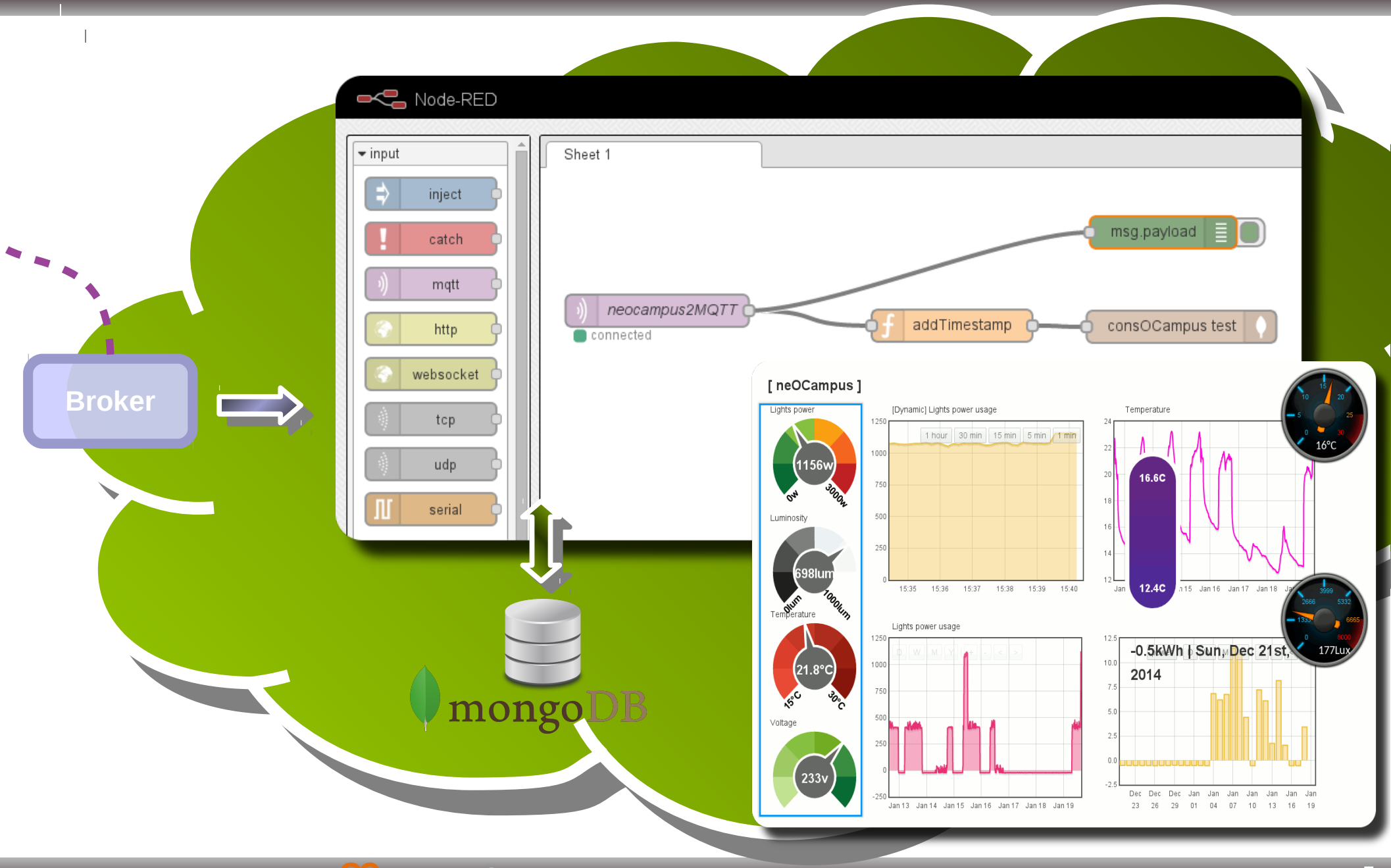
- Infrastructure overview,
- MQTT topics | communication abstraction,
- *Devices* registration | sensOCampus web. app.,
- Sensors / actuators | the *modules* way,
- *Backend* | I/O abstraction for modules,
- The affluencesOCampus use case.

Infrastructure overview



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

Infrastructure overview



Infrastructure overview

- **neOCampus** 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.

neOCampusLabs (U4 300,301,302)

IRIT

BU

neOCampus / CampusFab (U4 rdc)

```
rw-test: (g=0): rw=randrw, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=64
fio-2.2.8
Starting 1 process
Jobs: 1 (f=1): [m(1)] [100.0% done] [37676KB/12148KB/0KB /s] [9419/3037/0 iops] [eta 00m:00s]
rw-test: (groupid=0, jobs=1): err= 0: pid=2296: Thu Nov 26 11:40:28 2015
 read : io=3071.7MB, bw=25901KB/s, iops=6475, runt=121441msec
 write: io=1024.4MB, bw=8637.3KB/s, iops=2159, runt=121441msec
 cpu          : usr=3.20%, sys=27.39%, ctx=691717, majf=0, minf=1896
 IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
    submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
    complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
    issued    : total=r=786347/w=262229/d=0, short=r=0/w=0/d=0, drop=r=0/w=0/d=0
    latency   : target=0, window=0, percentile=100.00%, depth=64

Run status group 0 (all jobs):
   READ: io=3071.7MB, aggrb=25900KB/s, minb=25900KB/s, maxb=25900KB/s, mint=121441msec,
maxt=121441msec
   WRITE: io=1024.4MB, aggrb=8637KB/s, minb=8637KB/s, maxb=8637KB/s, mint=121441msec,
maxt=121441msec
```

OS & app. deployment

- Network automated install and configuration of our Raspberry Pi(s) :)

- ✔ FAT32 formatted SDCard
- ✔ 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)

DHCP

neocampus.univ-tlse3.fr

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

Summary

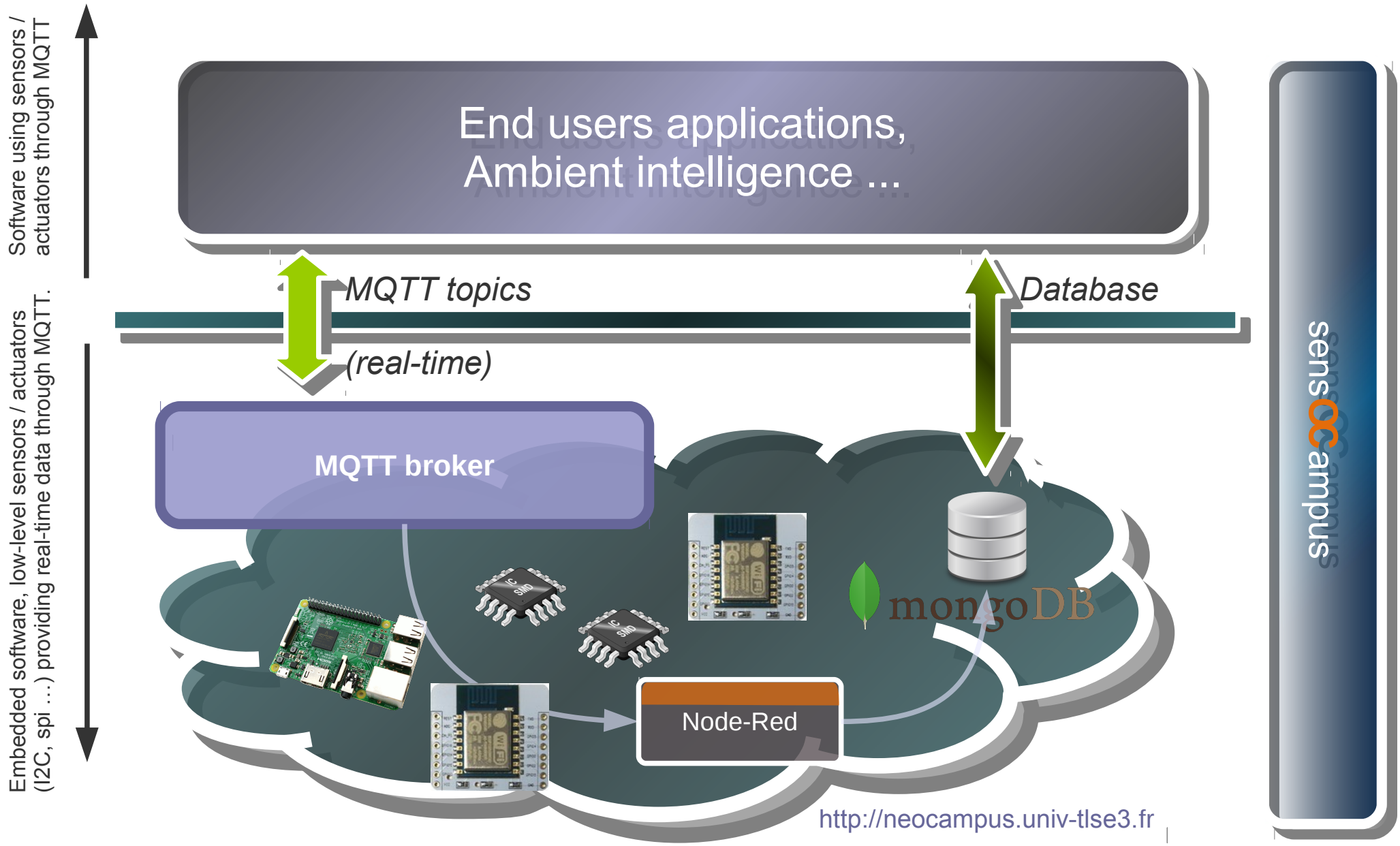
- Only registered *devices* will obtain a valid IP address,
- Wired / wireless neOCampus network is dedicated to IoT devices,
- Raspberry Pi can be fully re-installed directly through network (PXE-like),
- Raspbian-neOCampus (OS) latest images http://neocampus.univ-tlse3.fr/images
- Per-device Raspbian 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)

Plan

General

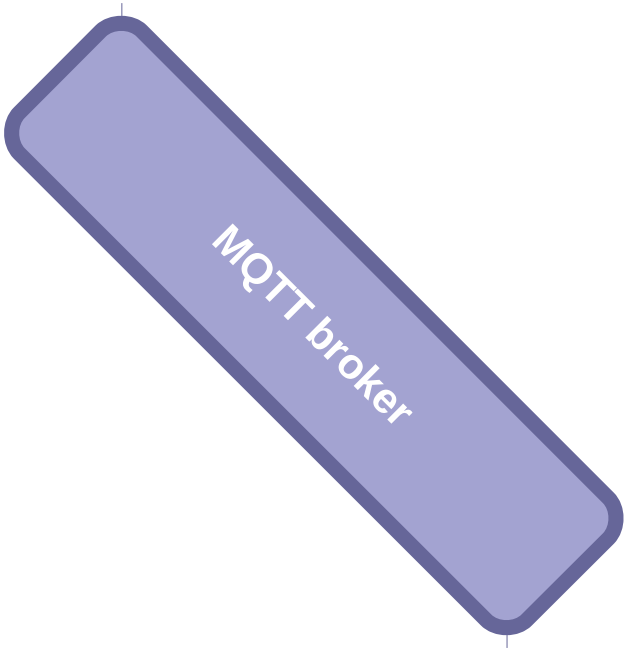
- Infrastructure overview,
- MQTT topics | communication abstraction,
- *Devices* registration | sensOCampus web. app.,
- Sensors / actuators | the *modules* way,
- *Backend* | I/O abstraction for modules,
- The affluencesOCampus use case.

Overview


Software using sensors / actuators through MQTT
Embedded software, low-level sensors / actuators (I2C, spi ...) providing real-time data through MQTT.
End users applications, Ambient intelligence ...
MQTT topics
(real-time)
Database
MQTT broker
mongoDB
Node-Red
http://neocampus.univ-tlse3.fr
sensOCampus

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)

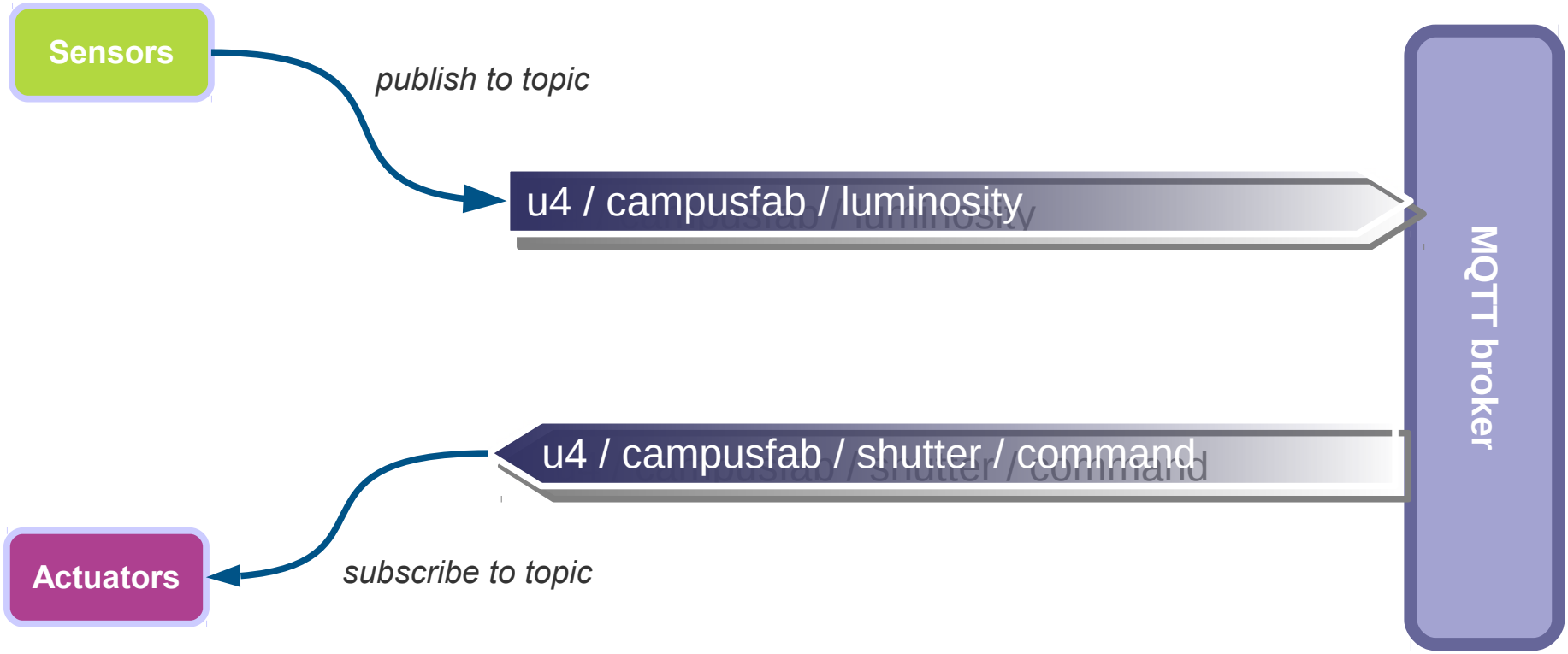
- Single topic publishing

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

u4 / campusfab / shutter / command

Base | type | [optional] command

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 topics

- MQTT payloads are json frames

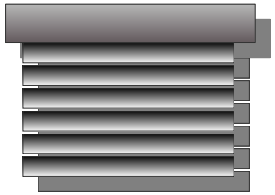
✔ Sending order to a shutter (with proper mqtt login / passwd)

```
Order: "UP"
Dest: "all"
```

Json frame as mqtt payload

u4 / campusfab / shutter / command

Shutter



✔ … then shutter publish its status back

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

Json frame as mqtt payload

u4 / campusfab / shutter

One caveats is that you can't send an order to a single module (shutter), hence the `dest` *field.*

MQTT json payload

- Shutter's MQTT status code snippet (json payload)

```
def _mqttStatus(self):
    ''' send a json frame reflecting shutter's status '''

    jsonFrame = { }
    jsonFrame['unitID'] = str(self.unitID)
    jsonFrame['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))
```

MQTT code snippet

- Sample MQTT publish / subscribe code snippet

```
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);

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

main()

publish()

```
# Acquire sensors and publish
def publishSensors():
    # get CPU temperature (string)
    CPU_temp = getCPUtemperature()
    # add some randomisation to the temperature (float)
    _fcputemp = float(CPU_temp) + random.uniform(-10,10)
    # reconvert to string with quantization
    CPU_temp = "{:.2f}".format(_fcputemp)
    print(time.strftime("%H:%M:%S") + " RPi temperature = " + CPU_te
    # generate json payload
    jsonFrame = { }
    jsonFrame['temperature'] = json.loads(CPU_temp)
    jsonFrame['unit'] = 'celsius'
    # ... and publish it!
    client.publish(MQTT_PUB, json.dumps(jsonFrame), MQTT_QOS)
```

Subscribe's callback
on msg recv()

```
# 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("Received message '" + json.dumps(payload) + "' on topic '" + msg.topic + "' with QoS " + str(msg.qos))
    print("Temperature is %s deg. %s" % (payload['temperature'],payload['unit']))
```

MQTT Topics

- CampusFab / neOCampus' showroom use case

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,
- ……

Plan

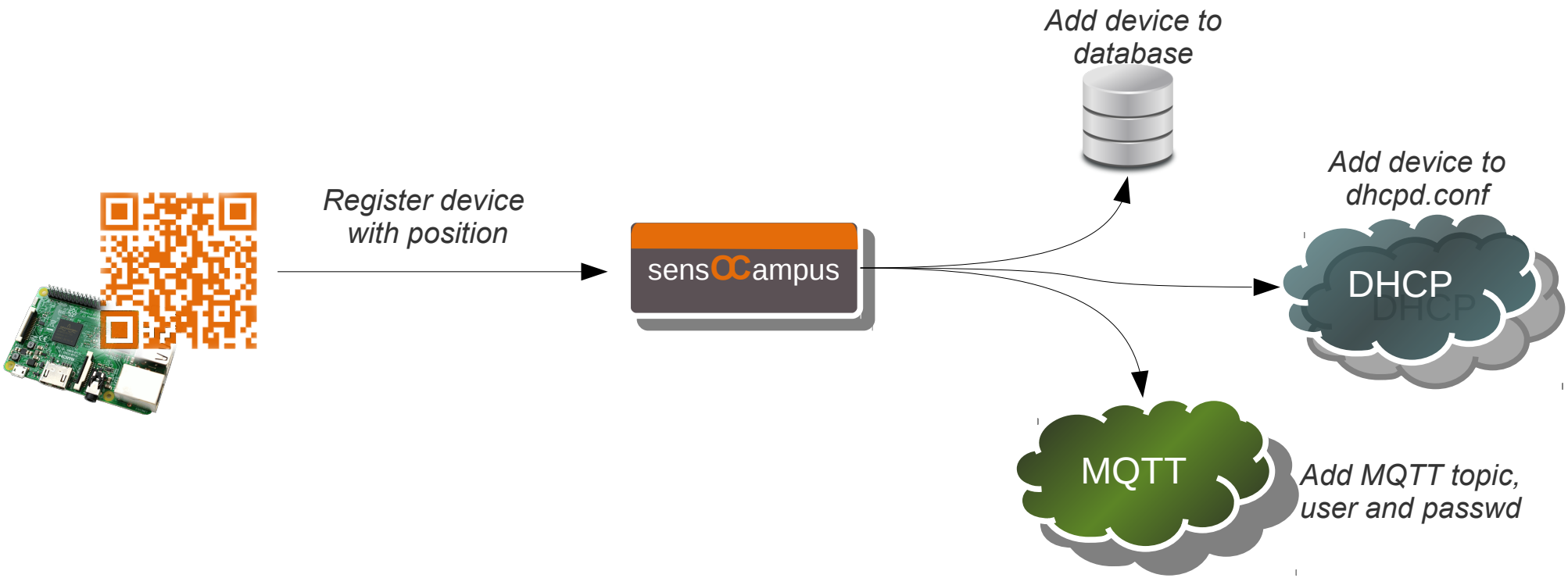
General

- 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.

Device registration

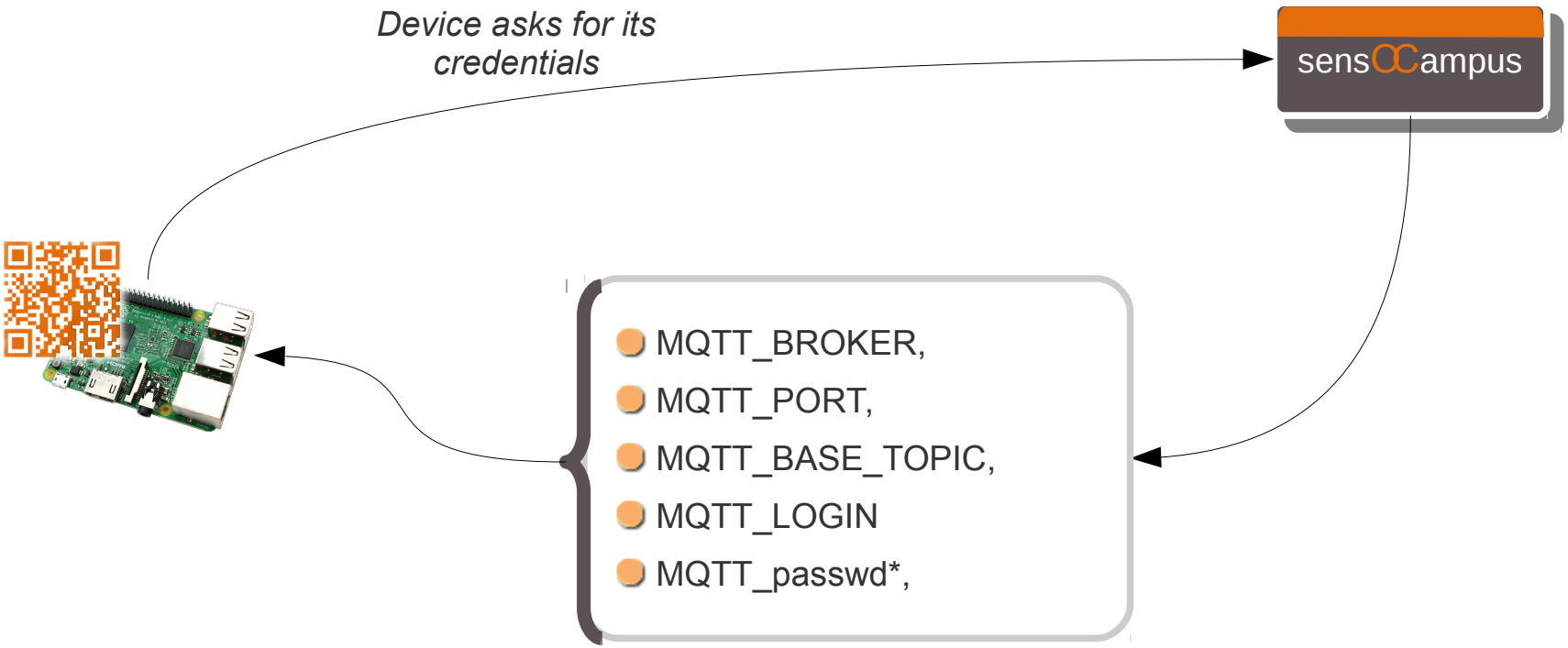
- Device registration | sensOCampus web. App. (Django)

A *device* is a physical embedded system connected to a network (eg. Raspberry Pi, ESP8266 ...).



Device registration

- … then registered device fetches its configuration from sensOCampus

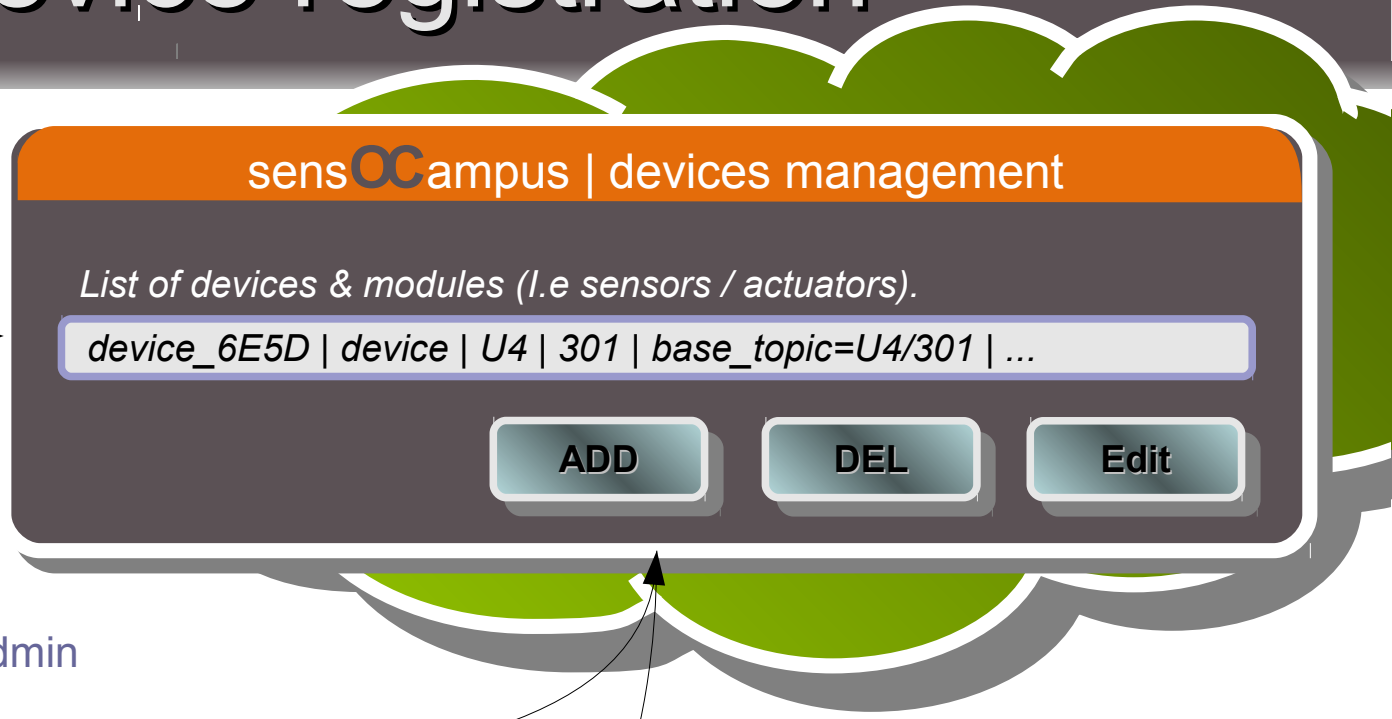


Minimum configuration sent from sensOCampus to a device
*MQTT passwd is only sent on first call (admin action required to create a new one otherwise)

Device registration

Register device with position

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



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

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

Device's topics

e.g. PiFace, Arbox etc

publish to

u4 / 301 / device

✔ device publish its status

```
unitID: "<macAddr>"
IO: "[ I / O list ]"
……
```

subscribe to

u4 / 301 / device / command

✔ Command to device

```
dest: "<macAddr>"
order: "load_module'
module: "shutter"
shutter_ID: "<ID>"
shutter_IO: [ 2, 3 ]
```

MQTT Topics

- CampusFab / neOCampus' showroom use case

Device's topics

✔ Publish status

I/O list,

Modules list,

Backends list,

Internal stuffs,

……

<mqtt_base> / device

<mqtt_base> / device / command

✔ Subscribe to orders

Load'n setup modules,

Delete module,

Init backends list,

Add specific low-level backend,

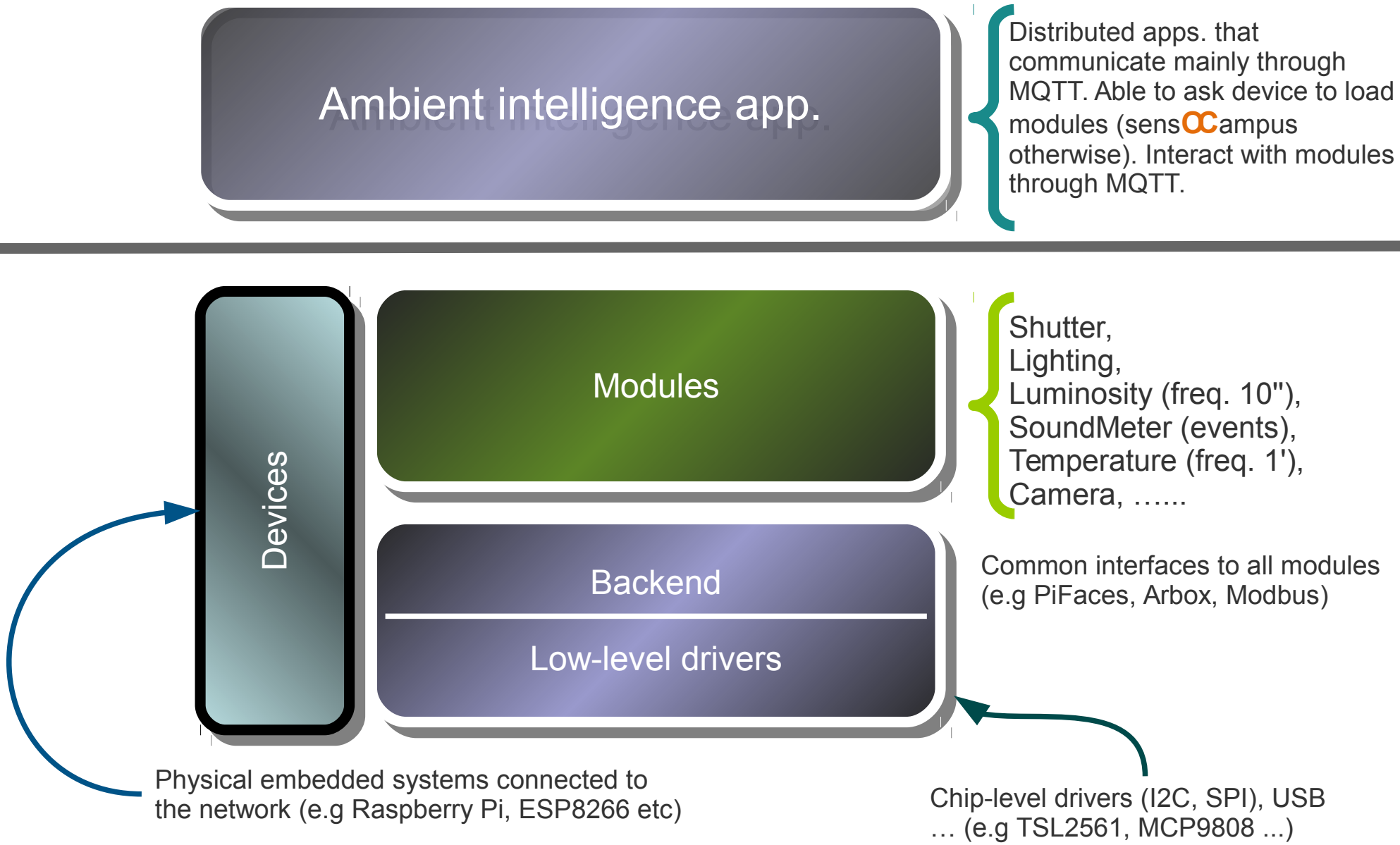
Delete one specific backend,

Update code from git,

……

A *device* API (python lib.) will be provided to the sensOCampus web. app.

Software architecture



Summary

- There's no direct communication between ambient intelligent application and *devices*,
- Several *devices* in a same room will share the same topic,
- Both sensOCampus web. app. and device framework are a work in progress,
- A python API will be provided to ease orders generation / status processing,
- ……

Plan

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.

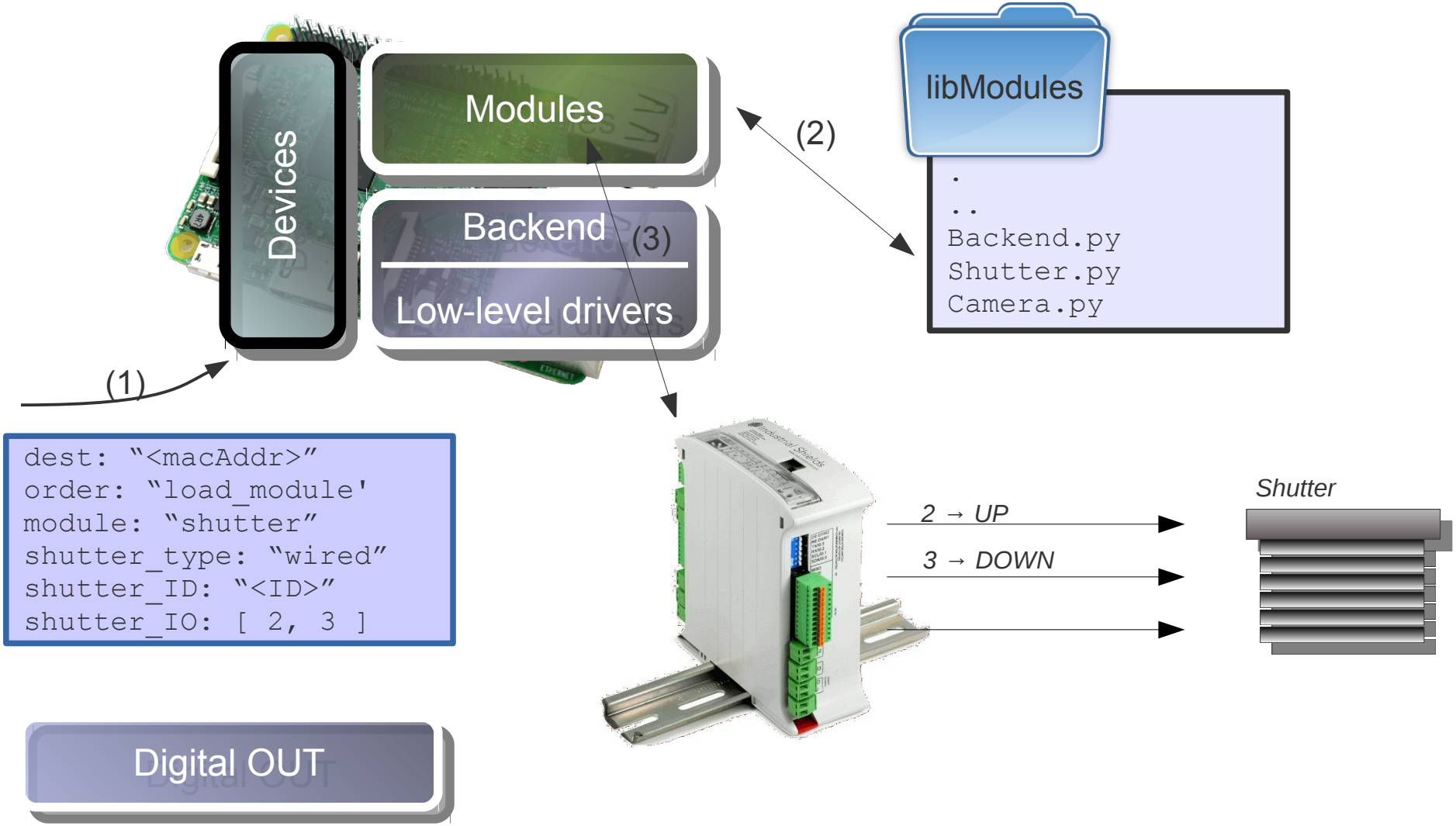
Modules

A *module* is an abstraction of a sensor / actuator that hides underlying hardware complexity.

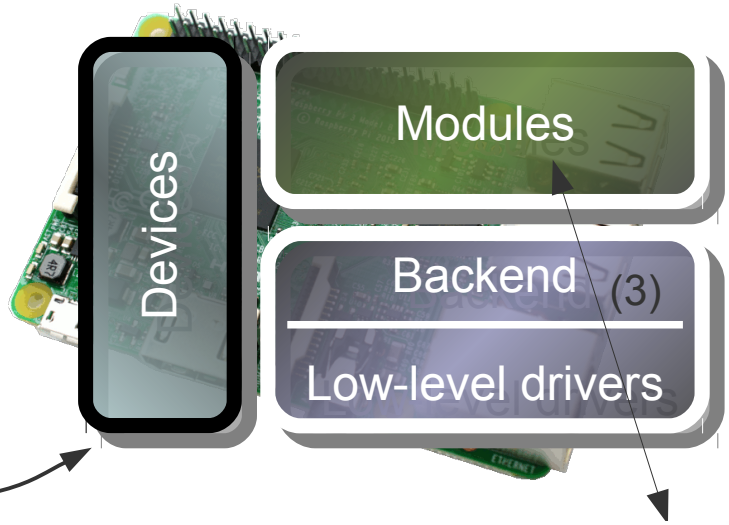
MQTT Type	Sub-type	Orders	Parameters
shutter	Wired or Wireless	UP, DOWN, STOP	CourseTime, outputs(Up,Down,Stop), ID ...
luminosity	numeric analog		Acquisition delay (default=10"), analog_range, ...
temperature	numeric analog		Acquisition delay (default=1'), analog_range, ...
camera	Motion-detect or RTSP		ID, rtsp_dest, ...
lighting	Telerupter, direct, DALI	TOGGLE, ON, OFF,xx% (DALI)	ID, output

Modules

- Sensors / actuators as pluggables *modules* (i.e dynamically loaded on-demand)



Modules



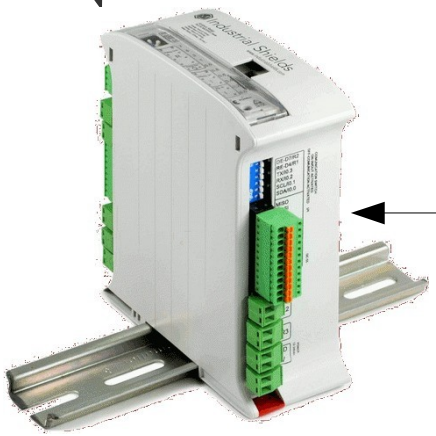
(2)

libModules

```
.
..
Backend.py
Temperature.py
Camera.py
```

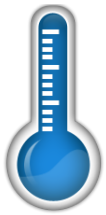
(1)

```
dest: "<macAddr>"
order: "load_module'
module: "temperature"
thermo_type: "analog"
thermo_ID: "<ID>"
thermo_range: [ -30, +125]
thermo_IO: [ 5 ]
```



5 → temperature

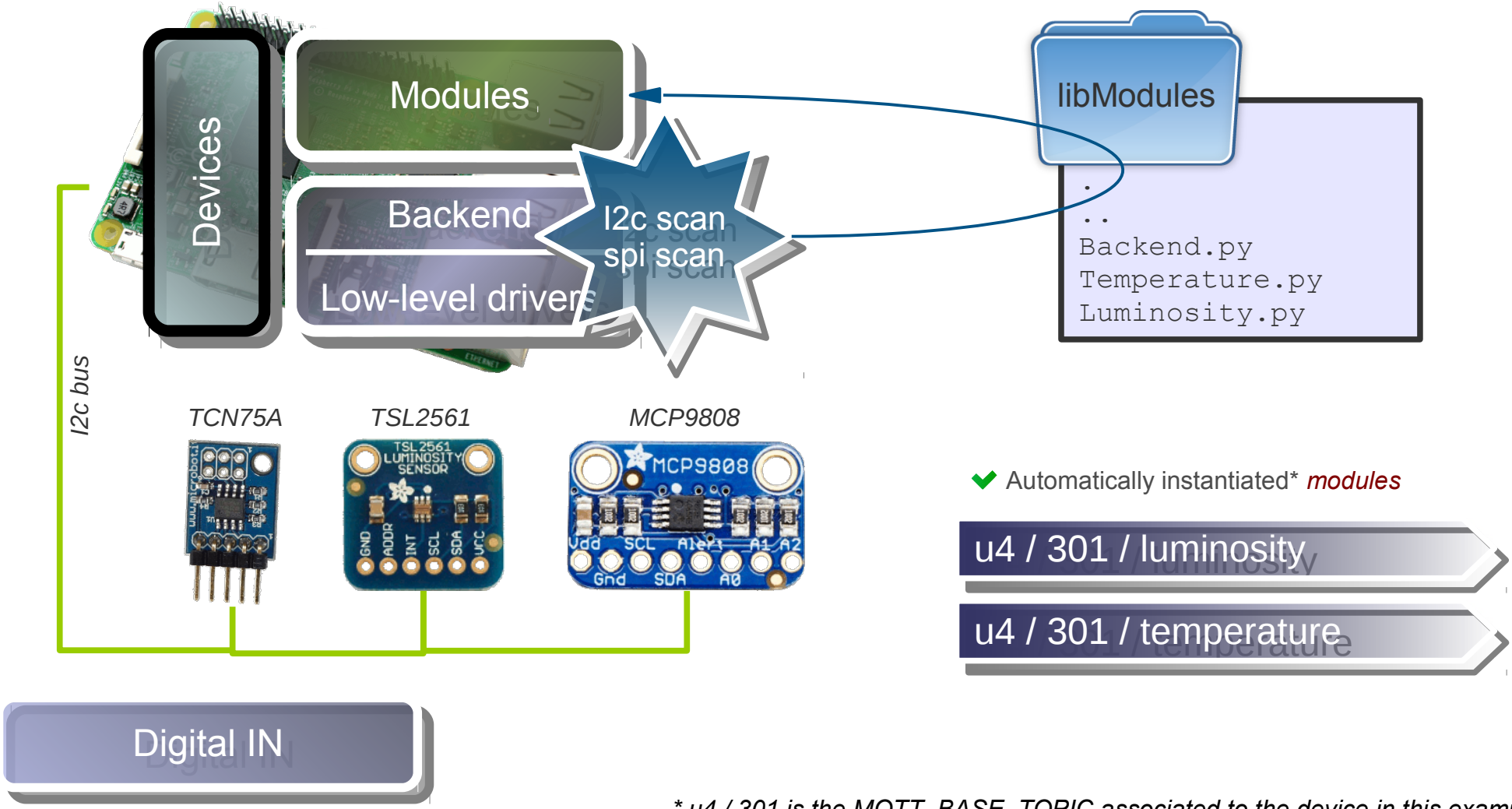
Thermometer (0-10v)



Analog IN

Modules auto-detection

- Some kinds of *modules* can be automagically detected / instantiated



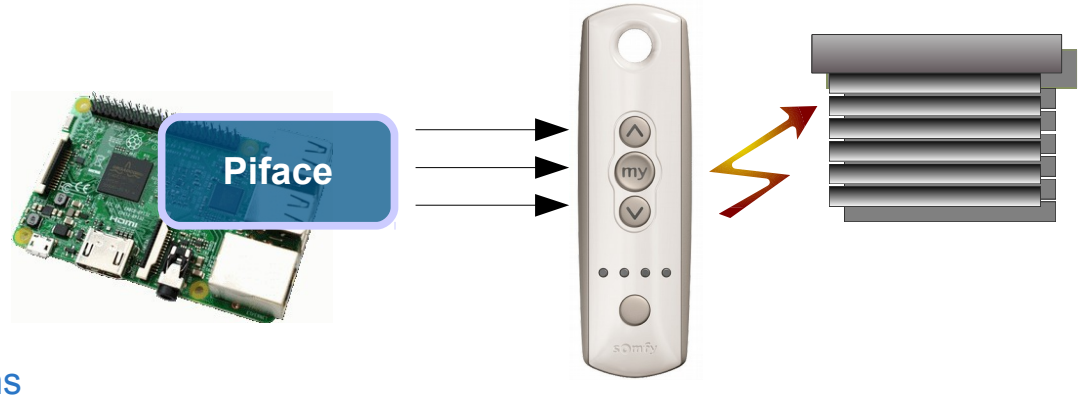
** u4 / 301 is the MQTT_BASE_TOPIC associated to the device in this example.*

Modules use case

- CampusFab / neOCampus' showroom | shutter *module* use case

✔ CampusFab shutter *module*

✔ Hacked SOMFY remote controller



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

Live demo

Work in progress

Summary

- 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,

Plan

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.

Backends

Work in progress

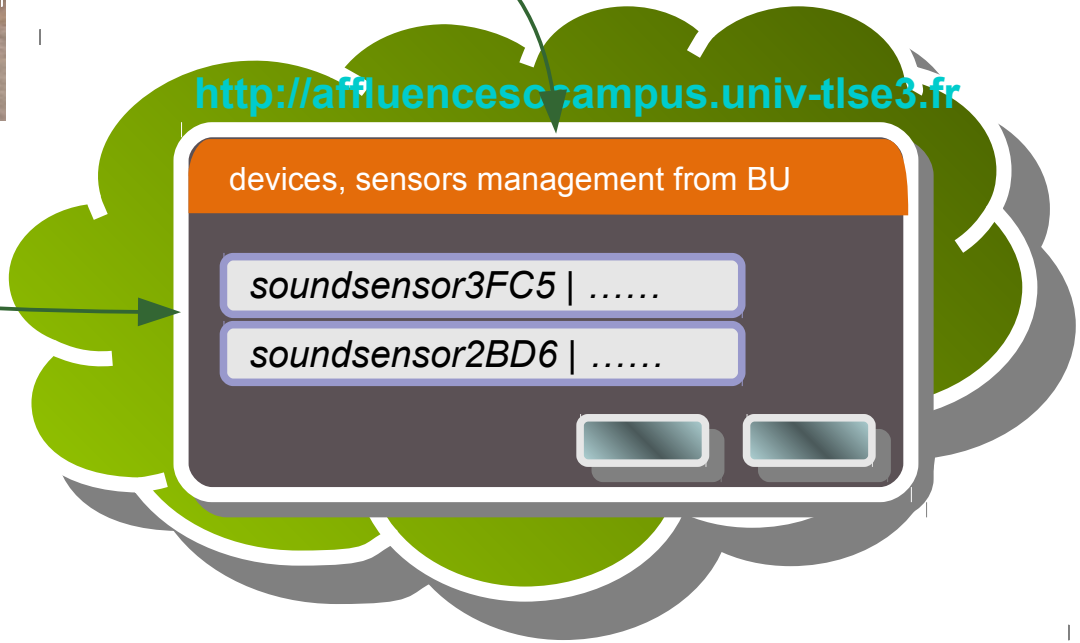
Plan

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.

affluencesOCampus

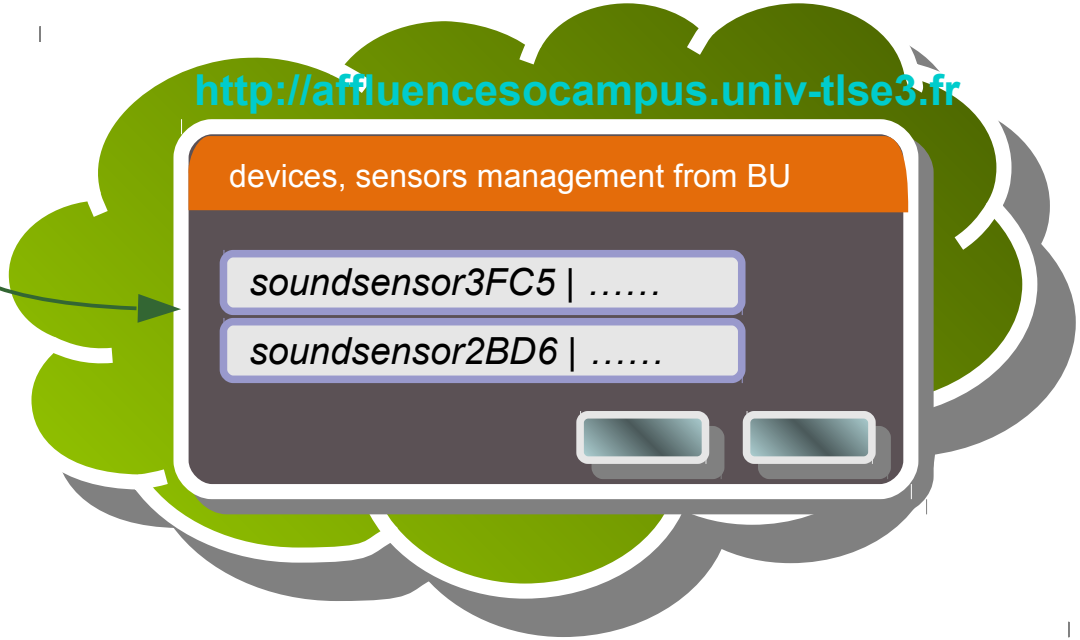
- [neOCampus] Attendance & noise measurement @ BU



Sensors and actuators benefit from neOCampus infrastructure being integrated within!

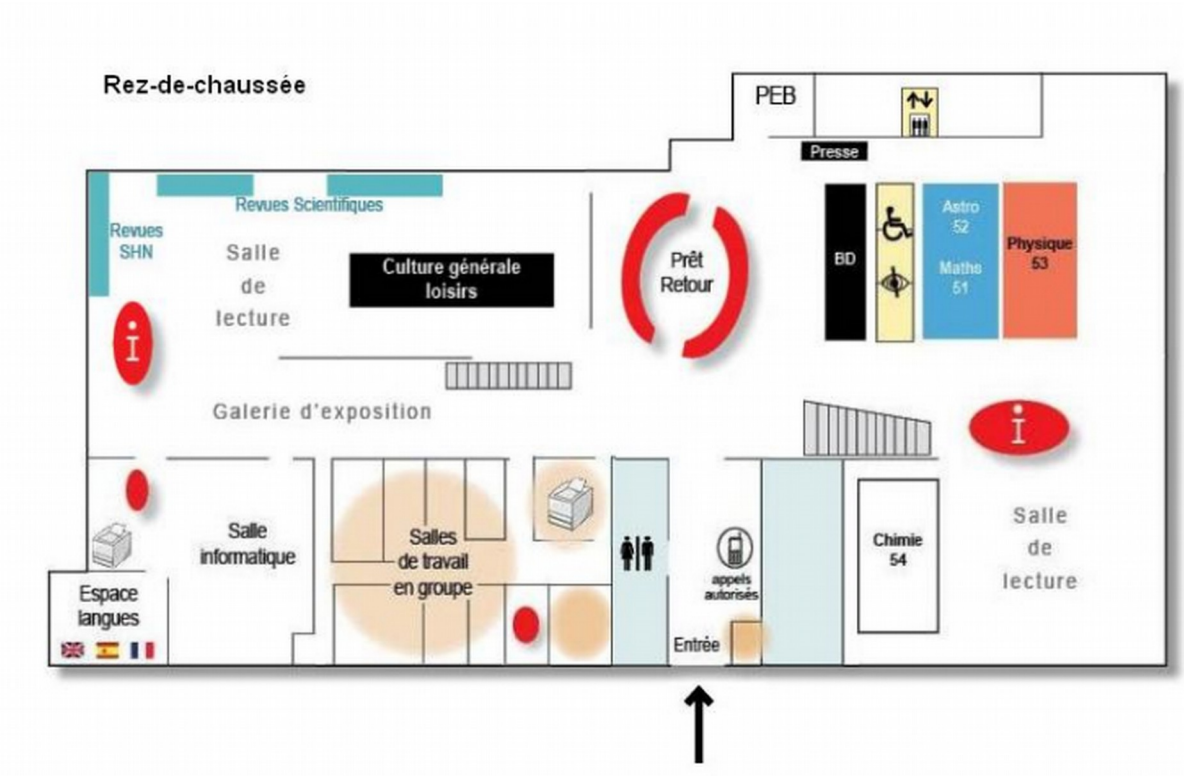
- noiseMeter registration @ sensOCampus

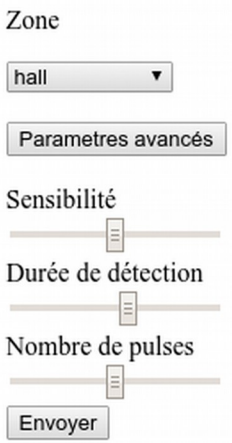
Register noiseMeter



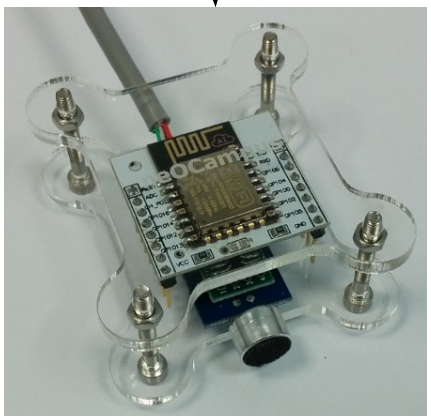
affluencesOCampus

- noiseMeter sensitivity parameters directly from browser!



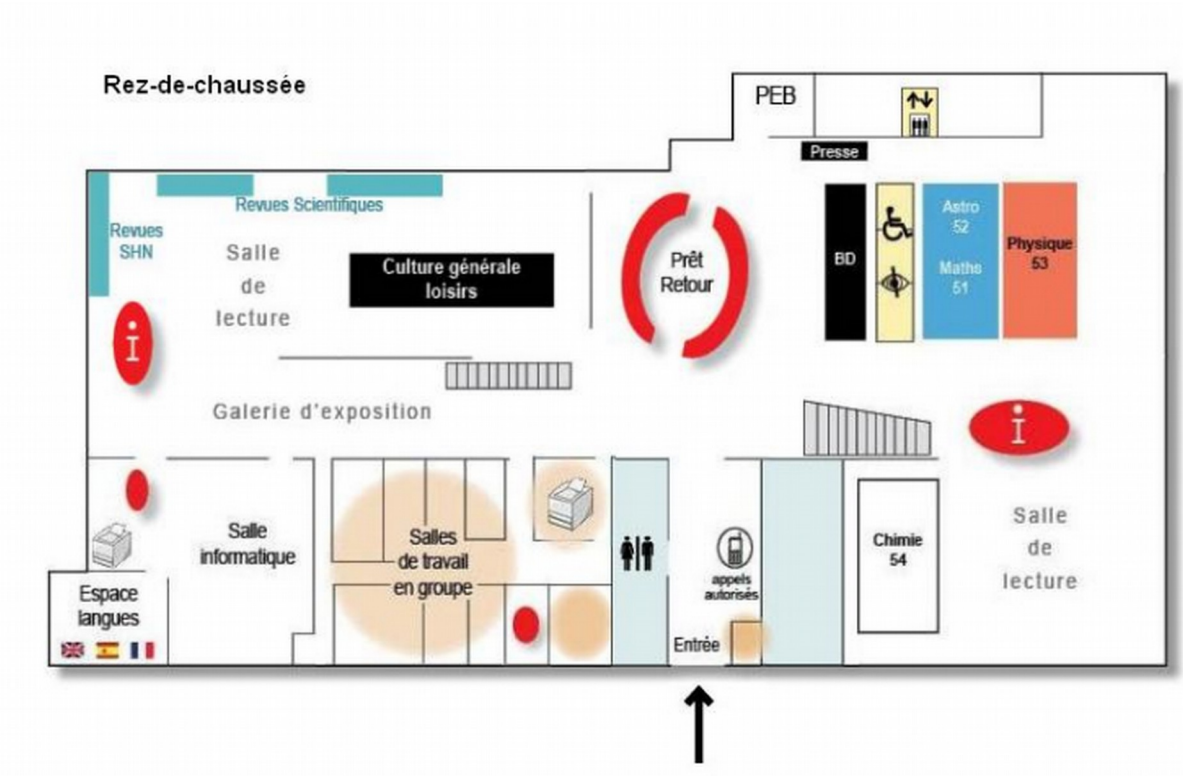


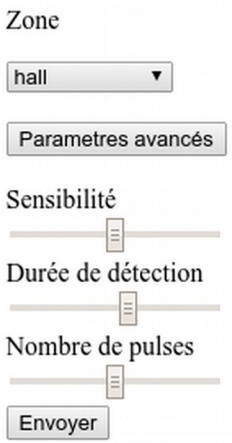
MQTT



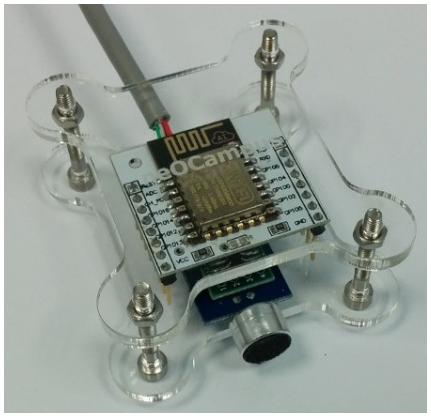
affluencesOCampus

- NoiseMeters live from browser !





MQTT



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

Links

Links

- Wiki | http://neocampus.univ-tlse3.fr
- 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 neOCampus 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 ?!?!

END

[2016, May 16th] new Raspberry Pi zero v1.3 with camera support … **$5** !!