How to add a sensor / actuator to neOCampus ?

sensOCampus API for end-devices

End-devices API
2017, October
Dr Thiebolt François, thiebolt@irit.fr

Abstract

This guide explains how to publish / subscribe data through the neOCampus **MQTT broker** and its associated device management application named **sensOCampus**.

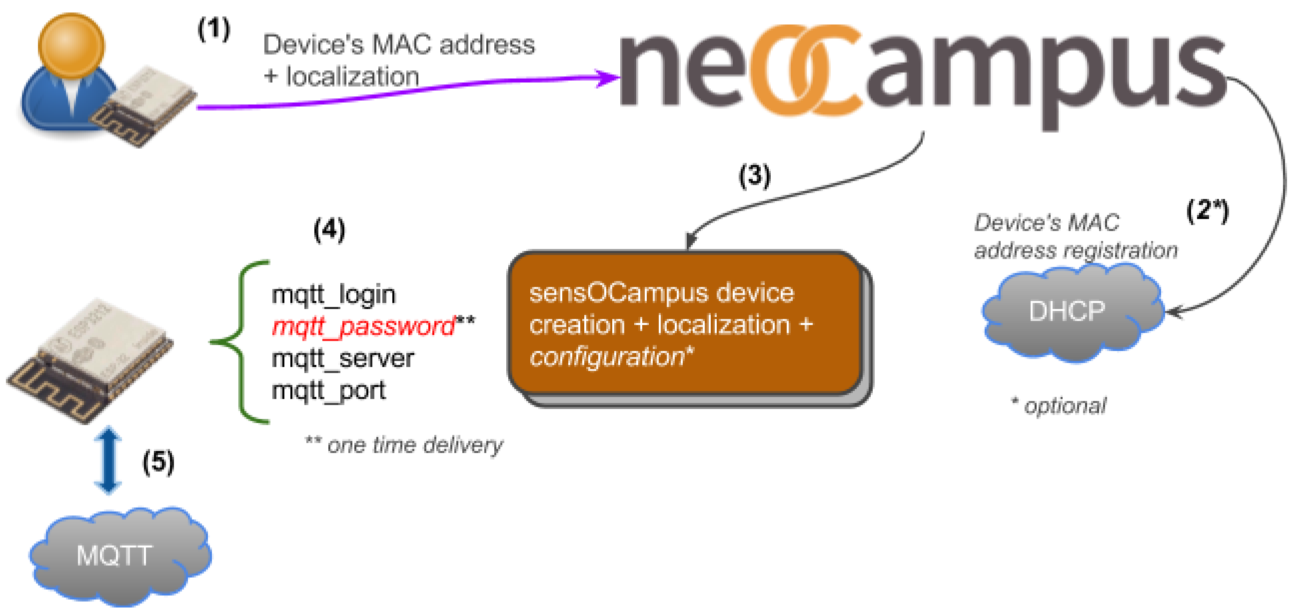
Table of contents

End-devices

We consider as a **device (or end-device)** a low-level piece of **hardware connected to a network**. Such a device may encompasses one to several sensors / actuators. It is devices' firmware responsibility to publish sensors values to the proper topic and to subscribe to relevant topics.

In upper example, device is a Raspberry Pi that could be connected to either a wired / wireless network. Each kind of sensors / actuators map to a topic class. However, to be able to publish / subscribe to the MQTT broker, device's client needs credentials. To obtain these credentials, you first need to:

1. declare device's MAC address to the neOCampus technical staff,
2. interact with sensOCampus application that will give you your credentials.



sensOCampus is the main end-devices management application developed for neOCampus. It takes care of managing device's own specific setup (MAC, configuration, status, topics etc).

sensOCampus credentials API

We below describe the various involved steps that end-devices need to undertake with the sensOCampus application to retrieve their credentials.

1. `get_credentials` → will give your device its mqtt related credentials,
2. `get_config` → your device will be given a **MQTT base topic** along with optional configuration registered for each peculiar device at the sensOCampus level.

get_credentials

```
https://sensocampus.univ-tlse3.fr/device/credentials?mac=<device_mac_addr>
```

… response will be in JSON format

```
{
   "login" : "<mqtt_login>",
   "password" : "<mqtt_password>",
   "server" : "neocampus.univ-tlse3.fr",
   "port" : 1883
}
```

Please pay attention to the facts that:

- **password** field is a one-time delivery parameter → if you loose it, you need to apply for new credentials at the neOCampus technical staff,
- "**server**" and "**port**" fields are **optionals** → you ought to have these default values in your code if sensOCampus does not deliver them to your device.

It's device's own responsibility to save these credentials in some non volatile hardware. If you apply for a credentials renewal operation, both login and password will change.

get_config

```
prepare HTTPS request with previously delivered credentials (https_auth) …
get https://sensocampus.univ-tlse3.fr/device/config
```

… response will be in JSON format

```
{
   'zones': [],
   'topics': ['bu/hall']
}
```

In this simple example, we have no specific configuration (empty zones) and we must take into account the **topics field as BASE_TOPIC.**

Note: in case 'topics' contains multiple fields, just select the first one

summary

You now have the following:

login	"<*mqtt_login*>"
password	"<***mqtt_password***>"
server	neocampus.univ-tlse3.fr
port	1883 or 8883 (tls)
BASE_TOPIC	bu/hall

Later, this BASE_TOPIC means that you've been granted the following topics rules:

bu/hall/+	publish & subscribe (i.e write & read)
bu/hall/+/command	subscribe (i.e read)

In next section, we'll start to talk about the MQTT conventions that apply to neOCampus.

MQTT topics conventions

The following describes various rules about topics conventions that apply to the neOCampus IoT.

- Each **end-device** get specified a **BASE_TOPIC** through its `get_config` procedure,
- A **BASE_TOPIC** is named accordingly to `<building>/<room>`, eg. **u4/302**
- Each sensor / actuator belongs to a **class** (e.g temperature, co2, shutter …) that is **appended** to the device's **BASE_TOPIC** (e.g bu/hall/temperature), named a **class topic**,
- Each sensor / actuator subscribe to a **command topic** with a **command** token appended to the class topic (e.g bu/hall/temperature/command)
- The **end-device** itself publish to a **class topic** (e.g bu/hall/device) and subscribe to a **command topic** (e.g bu/hall/device/command)
- Each **end-device** is identified by its **MAC_ADDRESS**,
- A **sensor** is either identified by an ID specified at sensOCampus server or automatically discovered at startup (e.g i2c scan). Sensors automatically discovered have an ID prefixed with auto (e.g 'auto_C32F' with last 2 digits being end of device's MAC_ADDR),
- An **actuator** is identified by an ID specified at sensOCampus server (e.g u4/302/shutter with 3 shutters identified as "front", "center" and "back").
- JSON frames' keys are mostly lower case ;)

Hence, for each device, each sensor and each actuator, there's:
- a **class topic** to publish to → BASE_TOPIC / CLASS
- a **command topic** to subscribe to → BASE_TOPIC / CLASS / **command**

msg to a command topic

Whenever a message is sent to a **command topic** (e.g bu/hall/shutter/command), the JSON frame **OUGHT** to contain a **'dest'** field.

- 'dest' : "**all**" → message is for all of those that subscribed to this command topic,
- 'dest' : "<*ID*>" → message is only for those whose ID matches

Example, if you wish to send an order to a specific **device**, 'dest' will contains its **MAC_ADDRESS**. if you wish to send an order to a shutter, 'dest' will contains its ID specified at the sensOCampus interface ("front" for example).

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

In example above, all shutters from u4 / campusfab will receive the "**up**" order thus opening all of them.

msg to a class topic

Whenever a device, a sensor or an actuator send a message to a **class topic** (e.g bu/hall/noise or u4/cfab/device), associated JSON frame **OUGHT** to contains a '**unitID**' field whose value reflect sender's identity:

- 'unitID' : "<*ID*>"

'unitID' : "00:08:a2:1f:cb:3f"	'unitID' : 'auto_CB3F'	'unitID' : "front"
sender is a device. *Note* that for compatibility, a 'unit' key with same value may be added.	sender is an auto-detected sensor usually associated with 'subID' : '<*i2c_addr*>' *Note* the last 2 digits are from device's MAC_ADDR	sender may be a sensor or actuator declared at sensOCampus level. *Note* a 'subID' will be added if it has been declared.

Json frame as mqtt payload

In example above, a shutter identified by its 'unitID' sent back its status through the class topic.

REMEMBER: when declaring several devices in a same room (e.g u4/campusfab), it is users' responsibility to manage identity uniqueness of sensors or actuators declared at the sensOCampus level.

unitID and subID

Whenever a sensor is <u>automatically</u> detected at startup (e.g i2c scan), it gets automatically attributed a 'unitID' (identity) and a 'subID' (informative only field ---e.g i2c addr)

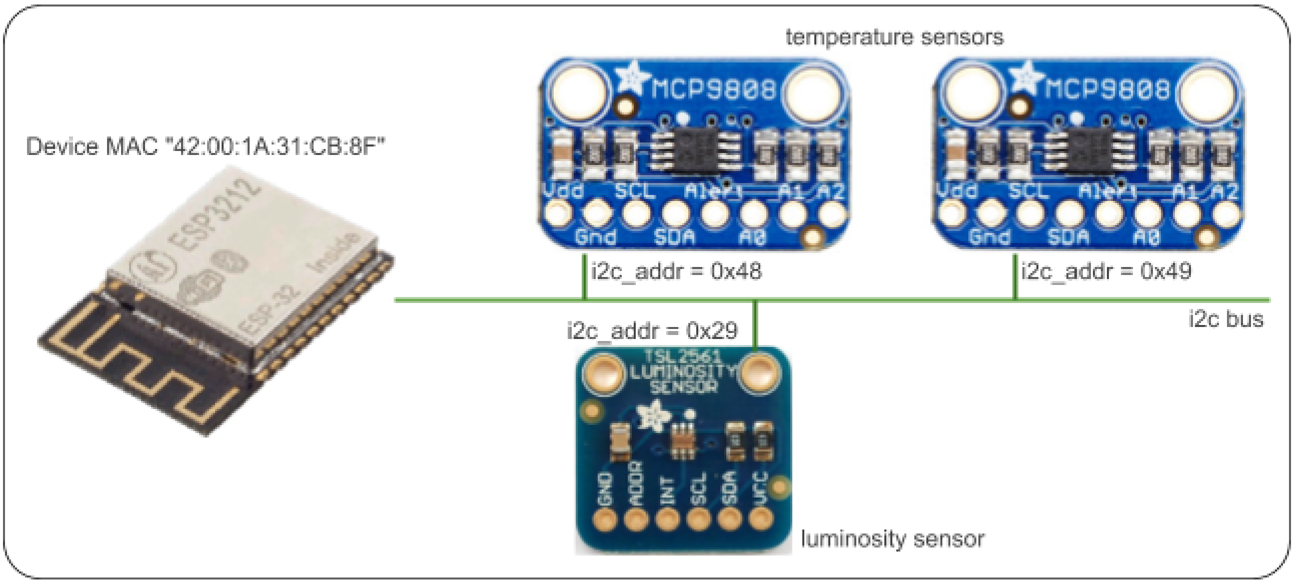
Example:

unitID	subID	Note
'unitID' : 'auto_CB8F'	'subID' : '32'	At least one auto-detected sensor with i2c addr 0x20. Device's MAC_ADDR end with CB8F
'unitID' : 'inside'	'subID' : 'ilot1'	A sensor or actuator declared at sensOCampus

Note: the nature of the sensor will be revealed according to it class topic publishing

scenario example

Considering the following device featuring 3 x i2c sensors.



This will result in the following identity of sensors:

'unitID'	'subID'	class & command topics
auto_CB8F	41	BASE_TOPIC/luminosity BASE_TOPIC/luminosity/command
auto_CB8F	72	BASE_TOPIC/temperature BASE_TOPIC/temperature/command
auto_CB8F	73	BASE_TOPIC/temperature BASE_TOPIC/temperature/command

sensors auto detection and messages publishing

Each sensor value is sent as a separate message. It means for example that if you feature 8 temperature sensor on a same device, you'll have 8 different message when it comes to push the data.

sensors and actuators uniqueness

It is IoT manager responsibility to ensure unitID uniqueness at the room-level.
Of course, if you add to the same room two devices whose MAC_ADDR last 2 digits are the same … use another device ;)

neOCampus MQTT sandbox

To ease testing of your sensor / actuator, you may give a try to the neOCampus MQTT sand box:

login	test
passwd	<*ask for it!*>
server	neocampus.univ-tlse3.fr
port	1883
BASE_TOPIC	TestTopic/#

Hence, you won't need the sensOCampus credentials and you are free to create / read / write in any topic you want considering it is BASE_TOPIC biased.

class topics and command topics

Below is a description of the currently existing classes:

Class	Publish	Subscribe
device	BASE_TOPIC/device	BASE_TOPIC/device/command
temperature	BASE_TOPIC/temperature	BASE_TOPIC/temperature/command
luminosity	BASE_TOPIC/luminosity	BASE_TOPIC/luminosity/command
humidity	BASE_TOPIC/humidity	BASE_TOPIC/humidity/command
co2	BASE_TOPIC/co2	BASE_TOPIC/co2/command
energy	BASE_TOPIC/energy	BASE_TOPIC/energy/command
camera	BASE_TOPIC/camera	BASE_TOPIC/camera/command
digital	BASE_TOPIC/digital	BASE_TOPIC/digital/command
noise	BASE_TOPIC/noise	BASE_TOPIC/noise/command
weight	BASE_TOPIC/weight	BASE_TOPIC/weight/command
uv	BASE_TOPIC/uv	BASE_TOPIC/uv/command
lighting	BASE_TOPIC/lighting	BASE_TOPIC/lighting/command
dali	BASE_TOPIC/lighting	BASE_TOPIC/lighting/command
shutter	BASE_TOPIC/shutter	BASE_TOPIC/shutter/command
display	BASE_TOPIC/display	BASE_TOPIC/display/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

device

Basis of all sensors / actuators, end-devices are connected to a network and are identified via their MAC address.
Each device ought to be able to:

- 'publish' some information (e.g status)
- 'subscribe' to a command topic

publish

BASE_TOPIC/device	JSON frame
status	*is **automatically** published every 30mn (default)* { 'unitID' : <*MAC_ADDR*>, 'status': "OK", <optional fields> }

Note: there's no 'values' because a device is not supposed to deliver such items.

The **'status'** key:

OK	normal operation
FAIL	an error occurred

Note: since this is only a user informative message, you can send any string you want!

subscribe

BASE_TOPIC/device/**command**	JSON frame
order	{ 'dest' : <*MAC_ADDR*>, 'order' : "*action*", <optional fields> }
frequency change order	{ 'dest' : <*MAC_ADDR*>, 'order' : "frequency", 'value' : <*integer seconds*> }

Note: 'frequency' is about 'status' delivery, not 'values' (whose message does not exists).

'order' command possible *actions*:

reset	reset application configuration and restart app.
restart	restart application
reboot	reboot the whole board

update	update application configuration
upgrade	update application and restart
reinstall	[Raspberry Pi] start whole SDCard reinstallation
status	force immediate delivery of a status report to its class topic
frequency	change frequency of **status report** delivery (min. 10mn, max 6h)

Note that status report is automatically published for each device while it is only published on explicit request for the sensors and actuators.

temperature / luminosity / co2 / humidity / weight / uv

These sensors send back ambient parameters. They are able to change their acquisition frequency and they transmit both 'value' of the sensor along with its physical unit (e.g 'value_units' : 'celsius')

publish

BASE_TOPIC / CLASS	JSON frame
status	*is published on request* { 'unitID' : <*ID*>, 'frequency': <*acquisition frequency seconds*>, <optional fields> } *optional fields may contain* *- 'sensors' → declared sensors* *- 'i2c_sensors' → automatically discovered sensors*
value	*is **automatically** published every <freq> seconds* { 'unitID' : <*ID*>, 'value': <*value*>, 'value_units' : "<*string*>" <optional fields> } *optional fields may contain* *- 'subID' → either i2c addr of sensor or explicit value set at sensOCampus level*

subscribe

BASE_TOPIC / CLASS /**command**	JSON frame
order	{ 'dest' : <*ID*>, 'order' : "status" }
frequency change order	{ 'dest' : <*ID*>, 'order' : "frequency", 'value' : <*integer seconds*> }
immediate acquisition order	{ 'dest' : <*ID*>, 'order' : "acquire" }

'order' command possible *actions*:

status	force immediate delivery of a status report to its class topic
frequency	change frequency of status report delivery (min. 10mn, max 6h)
acquire	force immediate delivery of sensor value(s).

energy

Power and energy consumption is usually gathered from Modbus energy meters. This kind of sensor can't be automatically detected, hence requiring a sensOCampus definition.

Moreover, each sensor gives a bunch a data (power, freq, energy, power_factor, intensity, voltage …) all packed as a list in the 'value' field along with their corresponding units in 'value_units':

- 'value' : ['158426.00', '158420.00', '235.22', '0.14', '20.00', '20.00', '30.00', '0.70']
- 'value_units' : ['Wh', 'Ea+', 'V', 'A', 'W', 'VAR', 'VA', 'cosPhi']

publish

BASE_TOPIC / CLASS	JSON frame
status	*is published on request* { 'unitID' : <*ID*>, 'frequency' : <*acquisition frequency seconds*>, 'backend' : <*backend type*>, <optional fields> } *optional fields may contain* *- 'link' : "/dev/usb0" (for example),* *- 'link_speed' : 9600,* *- 'nodes' : [(<modbus addr>,<meter_name>), …]*
value	*is* ***automatically*** *published every <freq> seconds* { 'unitID' : <*ID*>, 'value': <*value*>, 'value_units' : "<*string*>" } *Note: see above for a description of 'value' and 'value_units'*

<*backend type*> possible values:

modbus	either RS-485 or TCP modbus energy meter
rf868	868MHz energy meter (from consOCampus project)
unknown	as you guess ;)

subscribe

BASE_TOPIC/energy/**command**	JSON frame
order	{ 'dest' : <*ID*>, 'order' : "status" }
frequency change order	{ 'dest' : <*ID*>, 'order' : "frequency", 'value' : <*integer seconds*> }
immediate acquisition order	{ 'dest' : <*ID*>, 'order' : "acquire" }

'order' command possible *actions*:

status	force immediate delivery of a status report to its class topic
frequency	change frequency of status report delivery (min. 10mn, max 6h)
acquire	force immediate delivery of sensor value(s).

camera

digital

This class of sensor is related to everything that is relevant to digital inputs (e.g open window detector, motion sensor etc). This kind of sensors ought to get declared at the sensOCampus level.

For each event on a digital input (i.e rising_edge and falling_edge), a message will be sent immediately (i.e no timer involved but direct hardware events management).

publish

BASE_TOPIC / CLASS	JSON frame
status	*is published on request* { 'unitID' : <*ID*>, 'sensors': [[101, "button", "ilot1"], [102, "presence", "ilot1"]] } *'sensors' : [(input, type, subID), …] these values are coming from sensOCampus definitions*
value	*is **automatically** published for each event on input(s)* { 'unitID' : <*ID*>, 'value': 1, 'input' : 102, 'type' : "presence", 'subID' : "ilot1" }

subscribe

BASE_TOPIC/digital/**command**	JSON frame
order	{ 'dest' : <*ID*>, 'order' : "status" }

'order' command possible *actions*:

status	force immediate delivery of a status report to its class topic

noise

This sensor amplifies sound from a microphone and set a threshold on a comparator delivering pulses when sound intensity goes beyond. Pulses count are recorded over a sliding window giving their total number for an elapsed time (default 5s). If this total number of pluses is higher than a user defined threshold, then a noise message is sent.
Thus, this sensor is driven by two threshold:

- **'sensitivity'** → 0 to 100%. Set DAC output to the comparator,
- '**threshold**' → noise limit (pulses count).

publish

BASE_TOPIC / noise	JSON frame
status	*is published on request* { 'unitID' : <*ID*>, 'sensitivity' : <*0 to 100%*>, 'threshold' : <*integer*>, <optional fields> } *optional fields may contain* *- '_scan_window' → size of sliding window*
value	*is* ***automatically*** *published upon noise limit reached* { 'unitID' : <*ID*>, 'value': <*sum pulses count*>, 'value_units' : "pulses" <optional fields> } *optional fields may contain* *- 'input' → pin used to count pulses* *- 'subID' → DAC i2c addr*

subscribe

BASE_TOPIC/noise/**command**	JSON frame
order	{ 'dest' : <*ID*>, 'order' : "status" }
order to change sensitivity	{ 'dest' : <*ID*>, 'order' : "sensitivity", 'value' : <*integer 0 to 100%*> }
order to change noise limit	{

threshold (i.e pulse count limit)	'dest' : <*ID*>, 'order' : "threshold", 'value' : <*integer*> }
immediate acquisition order	{ 'dest' : <*ID*>, 'order' : "acquire" }

'order' command possible *actions*:

status	force immediate delivery of a status report to its class topic
sensitivity	set new value to DAC output used as input to the comparator
threshold	set noise limit through a maximum number of pulses count across the whole sliding windows
acquire	force immediate delivery of sensor value(s).

Note: there's no 'frequency' order because values delivery is not dependant of a timer.

lighting

Like all actuators, its setup is defined at the sensOCampus level.
This actuator drives various lighting command systems like **telerupteur** or **directly connected** lights sources.

publish

BASE_TOPIC / CLASS	JSON frame
status	*is published on request* ***and*** *upon light event (on, off)* { 'unitID' : <*ID*>, 'status': '[ON | OFF | unknown]' }

subscribe

BASE_TOPIC/device/**command**	JSON frame
order	{ 'dest' : <*ID*>, 'order' : '[ON | OFF] or *status*' }

'order' command possible *actions*:

status	force immediate delivery of a status report to its class topic
ON or OFF	set output to ON or OFF. *Note: if teleruptor type, any order will just toggle the output (i.e we don't know whether it is on or off)*

dali

shutter

Like all actuators, its setup is defined at the sensOCampus level.
This actuator is able to drive two kinds of shutters (blinds): **wired** and **wireless blinds** (difference is the way outputs are activated ---i.e short pulses for wireless).

publish

BASE_TOPIC / CLASS	JSON frame
status	*is published on request* ***and*** *upon shutter event* { 'unitID' : <*ID*>, 'status' : '[CLOSED | OPENED | UNKNOWN]', 'order' : '[UP | DOWN | STOP | UNKNOWN]' } *'status' field reflect current state of the shutter* *'order' field is the action currently undertaken*

subscribe

BASE_TOPIC/device/**command**	JSON frame
order	{ 'dest' : <*ID*>, 'order' : '[UP | DOWN | STOP] or *status*' }

'order' command possible *actions*:

status	force immediate delivery of a status report to its class topic
UP **DOWN** **STOP**	Set action to open, close or stop shutter in its current position

display

Annexe-A

A - 1 ESP8266 credentials sample code

```
bool _http_get( const char *url, char *buf, size_t bufsize, const char *login, const char
*passwd ) {

  HTTPClient http;

  http.begin(url);
  log_debug(F("\n[HTTP] GET url : ")); log_debug(url); log_flush();

  // authentication ?
  if( login!=NULL and passwd!=NULL ) {
    http.setAuthorization( login, passwd);
  }

  // perform GET
  int httpCode = http.GET();

  // connexion failed to server ?
  if( httpCode < 0 ) {
    log_error(F("\n[HTTP] connexion error code : ")); log_debug(httpCode,DEC); log_flush();
    return false;
  }

  // check for code 200
  if( httpCode == HTTP_CODE_OK ) {
    String payload = http.getString();
    snprintf( buf, bufsize, "%s", payload.c_str() );
  }
  else {
    log_error(F("\n[HTTP] GET retcode : ")); log_debug(httpCode,DEC); log_flush();
  }

  // close connexion established with server
  http.end();

  yield();

  return ( httpCode == HTTP_CODE_OK );
}
```

```
// HTTP get
bool http_get( const char *url, char buf[], size_t bufsize ) {
    return _http_get( url, buf, bufsize, NULL, NULL );
}

// HTTP get with credentials
bool http_get( const char *url, char *buf, size_t bufsize, const char *login, const char
*passwd ) {
    return _http_get( url, buf, bufsize, login, passwd );
}
```

A - 2 sensOCampus configuration U4/302


```
[
        {
                "topic": "u4/302",
                "modules":
                [
                        {
                                "module": "Shutter",
                                "unit": "front",
                                "params":
                                [
                                        {
                                                "param": "shutterType",
                                                "value": "wired"
                                        },
                                        {
                                                "param": "courseTime",
                                                "value": 20
                                        },
                                        {
                                                "param": "upOutput",
                                                "value": 100
                                        },
                                        {
                                                "param": "downOutput",
                                                "value": 101
                                        }
                                ]
                        },
                        {
                                "module": "Shutter",
                                "unit": "center",
                                "params":
                                [
                                        {
                                                "param": "shutterType",
                                                "value": "wired"
                                        },
                                        {
                                                "param": "courseTime",
                                                "value": 20
                                        },
                                        {
                                                "param": "upOutput",
                                                "value": 102
                                        },
                                        {
                                                "param": "downOutput",
                                                "value": 103
                                        }
                                ]
                        },
                        {
                                "module": "Shutter",
                                "unit": "back",
                                "params":
                                [
                                        {
                                                "param": "shutterType",
                                                "value": "wired"
                                        },
                                        {
                                                "param": "courseTime",
                                                "value": 20
                                        },
                                        {
                                                "param": "upOutput",
                                                "value": 104
```

```
                    },
                    {
                            "param": "downOutput",
                            "value": 105
                    }
            ]
    },
    {
            "module": "Digital",
            "unit": "inside",
            "params":
            [
                    {
                            "param": "frequency",
                            "value": 0
                    },
                    {
                            "param": "subIDs",
                            "value": [
                                                    "ilot1",
                                                    "ilot2",
                                                    "ilot3",
                                                    "window"
                                            ]
                    },
                    {
                            "param": "inputs",
                            "value": [
                                                    101,
                                                    106,
                                                    111,
                                                    115
                                            ]
                    },
                    {
                            "param": "types",
                            "value": [
                                                    "presence",
                                                    "presence",
                                                    "presence",
                                                    "open_close"
                                            ]
                    }
            ]
    },
    {
            "module": "Luminosity",
            "unit": "inside",
            "params":
            [
                    {
                            "param": "frequency",
                            "value": 0
                    },
                    {
                            "param": "subIDs",
                            "value": [
                                                    "ilot1",
                                                    "ilot2",
                                                    "ilot3"
                                            ]
                    },
                    {
                            "param": "inputs",
                            "value": [
                                                    100,
                                                    105,
                                                    110
                                            ]
                    },
                    {
                            "param": "min",
                            "value": [
```

```
                                                0,
                                                0,
                                                0
                                        ]
                        },
                        {
                                "param": "max",
                                "value": [
                                                1000,
                                                1000,
                                                1000
                                        ]
                        },
                        {
                                "param": "units",
                                "value": [
                                                "",
                                                "",
                                                ""
                                        ]
                        }
                ]
        },
        {
                "module": "Luminosity",
                "unit": "outside",
                "params":
                [
                        {
                                "param": "frequency",
                                "value": 60
                        },
                        {
                                "param": "subIDs",
                                "value": [
                                                "ouest"
                                        ]
                        },
                        {
                                "param": "inputs",
                                "value": [
                                                116
                                        ]
                        },
                        {
                                "param": "min",
                                "value": [
                                                0
                                        ]
                        },
                        {
                                "param": "max",
                                "value": [
                                                1400
                                        ]
                        },
                        {
                                "param": "units",
                                "value": [
                                                "w/m2"
                                        ]
                        }
                ]
        },
        {
                "module": "Temperature",
                "unit": "inside",
                "params":
                [
                        {
                                "param": "frequency",
                                "value": 0
                        },
```

```
                        {
                                "param": "subIDs",
                                "value": [
                                                "ilot1",
                                                "ilot2",
                                                "ilot3"
                                        ]
                        },
                        {
                                "param": "inputs",
                                "value": [
                                                103,
                                                108,
                                                113
                                        ]
                        },
                        {
                                "param": "min",
                                "value": [
                                                5,
                                                5,
                                                5
                                        ]
                        },
                        {
                                "param": "max",
                                "value": [
                                                40,
                                                40,
                                                40
                                        ]
                        },
                        {
                                "param": "units",
                                "value": [
                                                "",
                                                "",
                                                ""
                                        ]
                        }
                ]
        },
        {
                "module": "Temperature",
                "unit": "outside",
                "params":
                [
                        {
                                "param": "frequency",
                                "value": 0
                        },
                        {
                                "param": "subIDs",
                                "value": [
                                                "ouest"
                                        ]
                        },
                        {
                                "param": "inputs",
                                "value": [
                                                117
                                        ]
                        },
                        {
                                "param": "min",
                                "value": [
                                                0
                                        ]
                        },
                        {
                                "param": "max",
                                "value": [
                                                50
```

```
                                        ]
                        },
                        {
                                "param": "units",
                                "value": [
                                                ""
                                        ]
                        }
                ]
        },
        {
                "module": "Humidity",
                "unit": "inside",
                "params":
                [
                        {
                                "param": "frequency",
                                "value": 0
                        },
                        {
                                "param": "subIDs",
                                "value": [
                                                "ilot1",
                                                "ilot2",
                                                "ilot3"
                                        ]
                        },
                        {
                                "param": "inputs",
                                "value": [
                                                104,
                                                109,
                                                114
                                        ]
                        },
                        {
                                "param": "min",
                                "value": [
                                                30,
                                                30,
                                                30
                                        ]
                        },
                        {
                                "param": "max",
                                "value": [
                                                80,
                                                80,
                                                80
                                        ]
                        },
                        {
                                "param": "units",
                                "value": [
                                                "",
                                                "",
                                                ""
                                        ]
                        }
                ]
        },
        {
                "module": "Humidity",
                "unit": "outside",
                "params":
                [
                        {
                                "param": "frequency",
                                "value": 0
                        },
                        {
                                "param": "subIDs",
                                "value": [
```

```
                                                "ouest"
                                        ]
                },
                {
                        "param": "inputs",
                        "value": [
                                                118
                                        ]
                },
                {
                        "param": "min",
                        "value": [
                                                0
                                        ]
                },
                {
                        "param": "max",
                        "value": [
                                                100
                                        ]
                },
                {
                        "param": "units",
                        "value": [
                                                ""
                                        ]
                }
        ]
},
{
        "module": "CO2",
        "unit": "inside",
        "params":
        [
                {
                        "param": "frequency",
                        "value": 0
                },
                {
                        "param": "subIDs",
                        "value": [
                                                "ilot1",
                                                "ilot2",
                                                "ilot3"
                                        ]
                },
                {
                        "param": "inputs",
                        "value": [
                                                102,
                                                107,
                                                112
                                        ]
                },
                {
                        "param": "min",
                        "value": [
                                                0,
                                                0,
                                                0
                                        ]
                },
                {
                        "param": "max",
                        "value": [
                                                2000,
                                                2000,
                                                2000
                                        ]
                },
                {
                        "param": "units",
                        "value": [
```

```
                                                                                                  "",
                                                                                                  "",
                                                                                                  ""
                                                                                         ]
                                                              }
                                                    ]
                                          }
                                ]
                      }
]
```