ID-PHYCS - Interactive Disseminated PHYsical Complex Sensors

General presentation

We will design a system able to collect and supervise a set of sensors (IP and non-IP) scattered in the U3 building of the UPS.

The sensors are grouped by **aggregates** (the whole sensors of a room or an office) and a **slot** is composed of a set of rooms and/or offices. There are therefore 4+1 **slots** in the U3 (UPSSITECH administration is a separate **slot**). In this project, we will specifically focus on the administration network (see Figure 1.

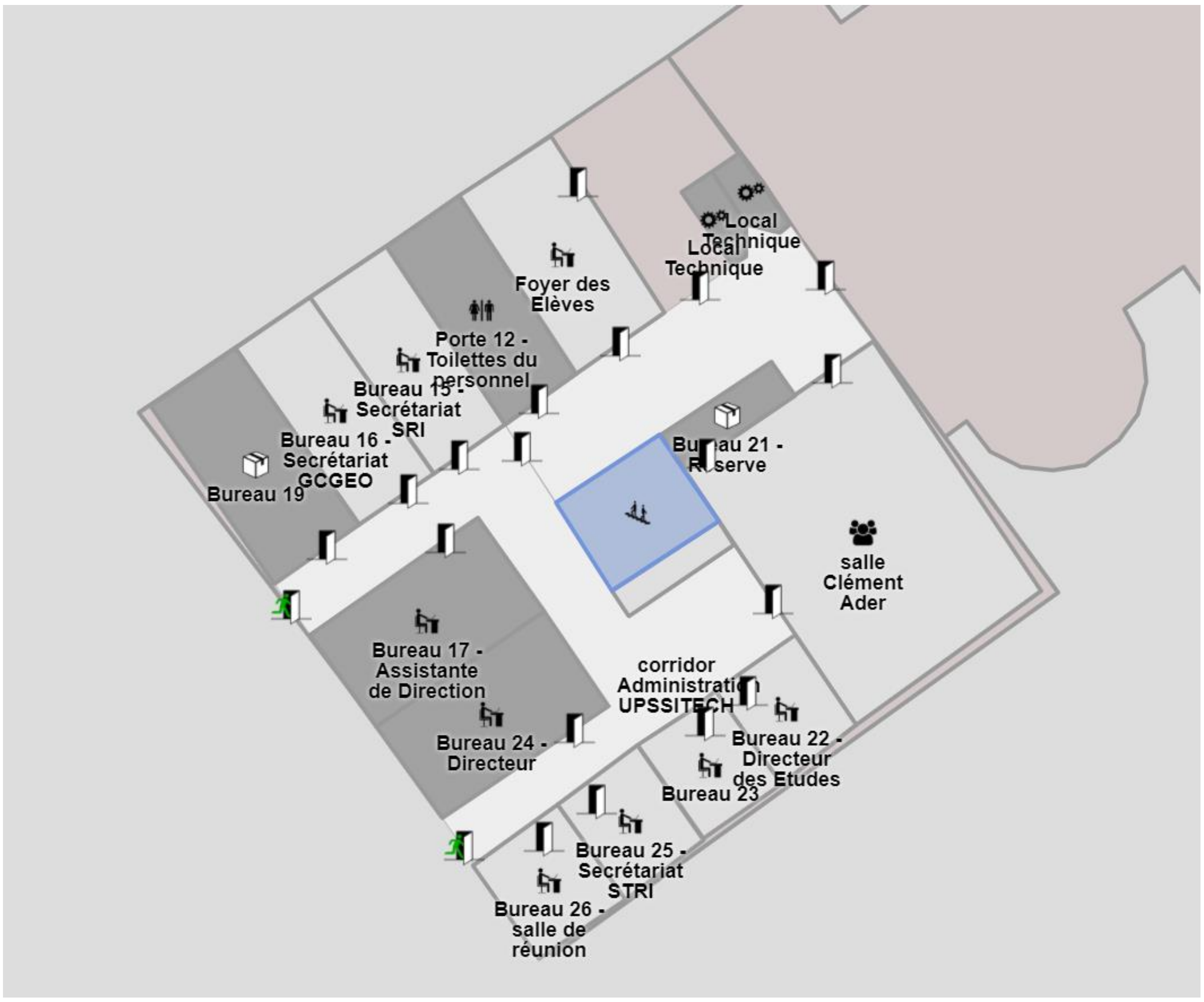
Local Technique
Local Technique
Foyer des Elèves
Porte 12 - Toilettes du personnel
Bureau 15 - Secrétariat SRI
Bureau 16 - Secrétariat GCGEO
Bureau 19
Bureau 21 - Réserve
salle Clément Ader
Bureau 17 - Assistante de Direction
corridor Administration UPSSITECH
Bureau 24 - Directeur
Bureau 22 - Directeur des Etudes
Bureau 23
Bureau 25 - Secrétariat STRI
Bureau 26 - salle de réunion

Figure 1 – 'UPSSITECH **slot** (`https://openlevelup.net/?l=0#18/43.56172/1.46952`)

Sensors are either **field sensors** that transmit regularly (frequency specific to each sensor) to an **aggregate**, or **"virtual" sensors** (accessible via some REST APIs) that are connected directly to a **slot**. Field sensors can be connected via different hardware and/or software protocols (serial link, USB, RF 433 radio, Zigbee, BLE, etc.). **Aggregates** store data "*locally*".

The objective of this project is to propose the *most detailed management* of this kind of network and to propose a dashboard allowing to visualize, manage the data and foresee potential problems.

Step 1: a general architecture

In this step, you have to choose the best architecture of your application based on **one** or **more** technologies seen in Tutorials (or other ones if you wish) to support links between sensors, **aggregates** and **slot**.

For this project, an **aggregate** consists of at least 2 sensors (**temperature, presence, power consumption**, ...) disseminated and located in the building which periodically send back their data. **The aggregates** store the configuration (in a JSON file) of the room (number of sensors, type, location, frequency, ...) and process data received (calculation of the average of sensor values for example).

Each **aggregate** sends its data back to a **slot** exposed on the internet via a REST API every 10 minutes (6 times per hour). Any client can retrieve this data from the slot at any time through a web request (the type of request is to be defined).

Propose a network architecture that supports this specification.

Step 2: a proof of concept

This step implements the initial version of the network. That's consist of a **slot** and an **aggregate** composed of at least two sensors. You will need to use at least one **"real"** sensor agent (on arduino, a nodeMCU, a RPi Zero or 3), etc.) The other sensors can be simulated.

Note: hardware can be given (micro-controller or Raspberry + sensors) on request for the development during this project.
Develop a first version of the project.

Step 3: A focus on aggregates

In this step, the **slot** has to be able to retrieve the current configuration of the different **aggregates** (its state over time, how many sensors are connected to the **aggregate**, their types, location in the room, etc.). The **aggregates** will have to register at their launch to the **slot**.
Add a second aggregate to your system. Use a JSON file between the slot and aggregates to allow the configuration retrieval.

Step 4: An operational service

In this last step, we want to **propose a dashboard** that can retrieve and display easily all data stored in the **slot** from the **aggregates.** You can use a map to allow the localization of the sensors (you can use for example the OpenStreetMap API with the OpenLevelUP layer! layer) and data of each sensor or their **aggregate**.

Propose for the dashboard a detection of any abnormal value in the different sensors, code the corresponding alerts and propose a prediction on the evolution of the values. (The dashboard can be coded on the web or via a "classic" graphic interface).

The project is done by two. At the end of the project, you will have to hand in:

- A **report** containing a list of your design choices illustrated by diagrams, screenshots or explanatory video. Make an assessment of the advantages/disadvantages/limitations of your solution. Try to be as objective as possible!
- The **source code** of your developments (git repository or zip archive) as well as the possible executable.

All documents and links must be sent by email (archive attachment or link) to Philippe Truillet (Philippe.Truillet@univ-tlse3.fr) before **Sunday 16 January 2022 23:55 UTC.**

A **0.25 pt** penalty per day of delay will be applied after this deadline.

You will also be able to propose an appointment to present your work orally if you wish.