Internet of Things
2023-2024

Final Project

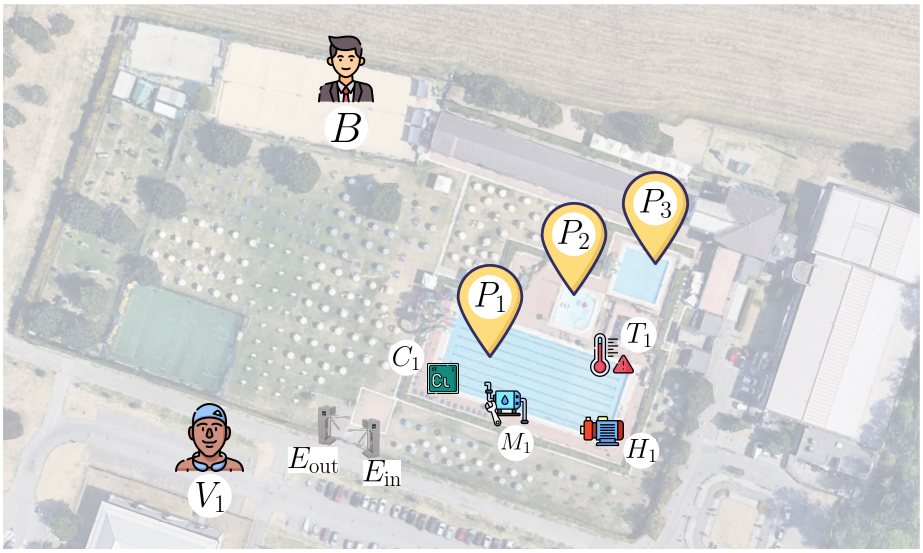
May 29, 2024

After completing the "Hands on `CoAP`" tutorial on `Java` and Californium framework, you are able to build both `CoAP` servers and clients, all running on a single laptop, as well as running on different machines, addressable through IP addresses on a common IPv4/IPv6 network. The final project is the following and is related to the generation, transmission, and management of IoT data related to the management of a generic *smart wellness village*. The project will rely on modelling parts of the system at application layer (relying on the CoAP protocol and the Californium framework) in a simulated environment. A final optional extension will encompass the transmission of selected information from a virtual node using a real LoRaWAN dongle connected to your laptop.

1 Smart Wellness Village

1.1 CoAP-based System Modeling

Consider a generic smart wellness village in which customers can relax taking advantage of the presence of $N_{\text{pool}} = 3$ swimming pools (denoted as P_i, $i \in \{1, \ldots, N_{\text{pool}}\}$). The first one is an Olympic swimming pool, the second one



is the relaxing lagoon, and finally a swimming pool specifically reserved for children. Then, each swimming pool P_i is provided with the following sensing elements, both monitoring the internal conditions of the swimming pool:

1. an immersion temperature sensor, denoted as T_i;
2. a chlorine concentration sensor, denoted as C_i.

Moreover, on the actuating side, each swimming pool P_i features the presence of the following components:

1. an heating pump, denoted as H_i, to be used for heating the water in the case the temperature drops below a certain threshold $T_{i-\text{th}}$;
2. a chlorine mixer, denoted as M_i, to be activated if the concentration of chlorine in the swimming pool P_i drops below a safety threshold $C_{i-\text{th}}$.

At this point, given the need to map these wellness village's elements in an IoT-like way, each swimming pool P_i can be modelled as a CoAP server hosting both the temperature sensor T_i and the chlorine concentration sensor C_i, while both the heating pump H_i and the chlorine mixer M_i can be modelled as CoAP clients. As a consequence, the behaviour of these latter components can be the following:

- H_i should `OBS`erve the behaviour of T_i; then, when the temperature drops below $T_{i-\text{th}}$, the heating pump must heat the water `POST`ing some consecutive messages on T_i (e.g., containing incremental temperature values), until the swimming pool's temperature reaches an optimal condition;
- M_i should `OBS`erve the behaviour of C_i; then, if the chlorine contained in the water drops below $C_{i-\text{th}}$, the chlorine mixer must sanitize the water `POST`ing some consecutive messages on C_i (e.g., containing incremental chlorine concentration values), until the hygienic conditions inside the swimming pool P_i reach an acceptable situation.

But the real question is the following: *how could a smart wellness village be interesting without holidaymakers?* So let's model them, too.

In order to monitor how many people enter into/exit from the wellness village, a smart entrance turnstile and a smart exit turnstile (denoted as E_{in} and E_{out}, respectively) have been installed. Both the turnstiles are candidate to be mapped as CoAP servers, and they should both support the following operations:

- accept `POST` requests containing the member identifier (denoted as V_{id}) of the customer entering/exiting the wellness village, and returning an error in the case a visitor wont to provide his/her member information;
- accept `GET` requests and return the total amount of people entered into/exited from the wellness village;
- support external CoAP clients interested in `OBS`erving the total amount of people entered into/exited from the wellness village.

As a consequence, it would be easy to think to a customer as a CoAP client required to interact with both the entrance and the exit turnstiles in the proper way, having to provide the required information.

Then, *last but not least*, our smart wellness village should be supervised by an expert manager (denoted as B, "*like a boss*"), in charge of ensuring that the customers' well-being would be as high as possible. To this end, the manager B, which can be mapped (similarly to a customer) as a CoAP client, will have to manage the following operations:

- `OBS`erve both the turnstiles: in the case of need, if a customer cannot enter or exit the wellness village (i.e., due to a malfunction of a turnstile), B should be able to open the turnstile with the required member identifier information;
- `OBS`erve the status of the swimming pools P_1, P_2, and P_3, with regards to their corresponding temperature and chlorine sensors: in the case the hygienic conditions inside a swimming pool tend to get too worse because of an actuating device not intervening properly (e.g., heating pump or chlorine mixer not sending their proper commands, as detailed before), then the manager B should `POST` a "drastic" value (e.g., temperature or chlorine) toward the corresponding system, in order to re-establish the well-being for the customers.

1.2 LoRaWAN Integration (OPTIONAL)

Given the fact that each swimming pool is actively monitored by different sensing elements, it could be useful to give the smart wellness village the possibility to send alert messages to the remote administrator of the village itself, in the case one or more swimming pools have problems. Among the possible communication technologies that could be used, LoRaWAN seems to fit perfectly with the proposed scenario. Let us then improve the aforementioned scenario with an enhanced functionality!

In detail, suppose that the wellness village is equipped (on its overall) with a physical LoRaWAN node able to communicate with this network technology. Then, in the case of a warning situation, the village should prepare a LoRaWAN uplink packet L_{up}, to be then sent toward the LoRaWAN Network Server that, in turn, will pass the sent information to its LoRaWAN Application Server (you can use TTN (`https://www.thethingsnetwork.org/`) as LoRaWAN infrastructure). So as, each LoRaWAN packet L_{up} should contain, as payload, the following information (encoded in JSON format): (i) swimming pool's identifier, (ii) information class to be reported (e.g., temperature or chlorine), and (iii) sensed value to be reported.

On the implementation level, since the CoAP-based modeling detailed in Section 1.1 is only a virtual deployment, while the LoRaWAN-based communication should exploit a real LoRaWAN node connected to your laptop, the interaction between the wellness village and the LoRaWAN node should require the integration (in your Java project) of a Java library allowing you to open a serial connection on the COM port assigned by your operating system to the LoRaWAN dongle itself. Finally, assuming that that LoRaWAN node has been previous registered on the LoRaWAN Application Server (e.g., with its

LoRaWAN parameters and activation mode), after the successful serial connection, you will need to use textual commands on this serial connections in order to perform the following operations:

1. at the project's boot time, let the LoRaWAN node joining the LoRaWAN network (only the first time);
2. when needed, prepare the payload to be sent through the LoRaWAN connection (as detailed above);
3. send the payload through LoRaWAN.

Finally, connecting to the Web interface of the LoRaWAN Application Server, it will be possible to verify if these uplink transmissions succeed or fail.

2 Repository Entity

In the considered scenario, in case there is the need to share some information among the different entities mentioned in the previous section or as will be described in the following, it is possible to adopt the following strategy. Create a "fake" repository (logically similar to a database), represented by a `Java` class, in which it is possible to maintain some static objects, as follows:

```
public class Repository {
    private static Repository instance = null;
    public static Repository instance() {
        if (instance == null) {
            instance = new Repository();
        }
        return instance;
    }

    public YourObjectClass var = new YourObjectClass();
}
```

In this way, the required information is available everywhere, and one can refer to this object with `Repository.instance().var.<method>()`.

Note #1: the `Repository` class has to be used ONLY if it is strictly necessary; otherwise, it is recommended to exchange data ONLY exploiting the `CoAP` protocol.

Note #2: the content of the `Repository` class will exist only at run time, i.e., it is not physically stored on disk. Please, be aware of this aspect.

In general, it is suggested to follow the Plain Old Java Object (POJO) paradigm: keep internal variables related to the information proper of your classes as private variables; implement `get/set` methods for each of these private variables.

Finally, it is up to you to define the exchange *schema* to be adopted for representing the information to be sent through `CoAP POST` requests. As mentioned before, one possibility is represented by the adoption of the `JSON` format.

3 Final delivery of the project

You are requested to prepare a 10 minute presentation (PowerPoint, PDF, Keynote, whatever you prefer) in order to *face-to-face* detail and explain your work. One week before the day you would like to present your work please send the complete developed code in a ZIP archive.

- The ZIP archive should be named `surname_name_studentid.zip`, where `studentid` is your *matricola* number.
- The ZIP archive should contain a folder named `surname_name_studentid`, in turn containing the project files. These files should allow the commission to run the delivered project and to evaluate it.

The ZIP archive MUST be sent through a **unique** e-mail to:

- Prof. Gianluigi Ferrari (**gianluigi.ferrari@unipr.it**)
- Dr. Luca Davoli (**luca.davoli@unipr.it**)
- Dr. Laura Belli (**laura.belli@unipr.it**)

Please be sure to send the ZIP archive to all these addresses together