In our instance, we want to monitor and analyse UnivAQ’s buildings.

Our architecture is based on a PubSub pattern, implemented using the Kafka framework.

The data concerning the system’s organization is stored in a relational database while the sensed data is stored in a NoSQL Mongodb database.

This approach, along with features provided by Kafka, allow the system to have a great level of dependability such as: if a producer loses the connection to the cluster it will wait for the connection to come back online without crashing or losing messages. If a consuming client loses the connection it will simply wait for the cluster to come back online and it will resume the consumption from where it left.

The fault-tolerance of the cluster can be increased by adding replicas: a leader will be elected among the participant and when a component goes offline the others will proceed its work with all the up to date data.

It also provides good performance (as exposed on the previous pages we are able to sense-and-store 240000 values in 15 seconds when the requirements were referring to 40000/hour), with the clients running on a common laptop (we expect greater performance on dedicated hardware).

All the clients can run on different machines and on different geographic areas (as demonstrated on the video-demo).

We can have an automatic load balance by simply running more instances of a client. For instance, if we run the Storing client 3 times, as each of the instances belong to the same group, the cluster will automatically assign a portion of the messages to each client, balancing the load (as each instance can run on a different machine).

This approach also provides a great level of decoupling as all the components are independent from each other.

For instance, the Storing client is not related in any way to the Visualization client or the Analysis Engine or the Actuators client.

We are also able to add new sensors and new areas simply by installing the physical device, let him publish on its topic and register the topic on the relational database: the clients will periodically check for changes and will subscribe automatically to the new topics without any kind of manual action, plus each client rceives its configuration by a configuration file, this avoid to change the code if, for instance, the cluster address changes.