



Integrated Demand REsponse SOlution Towards Energy POsitive NeighbourhooDs

WP2 USE CASE DEPLOYMENT AND FOLLOW-UP

T2.5 DEMAND RESPONSE PLATFORM DEPLOYMENT

D2.5 DEPLOYMENT OF RESPOND CLOUD-BASED PLATFORM

The RESPOND Consortium 2019



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 768619

PROJECT ACRONYM	RESPOND
DOCUMENT	D2.5 Deployment of RESPOND cloud-based platform
TYPE (DISTRIBUTION LEVEL)	<input checked="" type="checkbox"/> Public <input type="checkbox"/> Confidential <input type="checkbox"/> Restricted
DELIVERY DUE DATE	30.09.2019.
DATE OF DELIVERY	30.09.2019.
STATUS AND VERSION	Final, version 1.1
DELIVERABLE RESPONSIBLE	IMP
CONTRIBUTORS	DEV, FEN, AURA, ALBOA, NUIG, ARAN, ENE, DEXMA
AUTHOR (S)	Lazar Berbakov, Dea Pujić, Marko Jelić, Dejan Paunović, Lisbet Stryhn Rasmussen, Avril Sharkey, Iker Esnaola, Francisco Javier Díez, Rodrigo López, Federico Seri, Andreu Pagès
OFFICIAL REVIEWER(S)	TEK

DOCUMENT HISTORY

	ISSUE DATE	CONTENT AND CHANGES
0.1	05.07.2019.	Table of contents
0.2	15.07.2019.	Pupin's contribution
0.3	01.08.2019.	Pupin's contribution
0.4	23.08.2019.	Aura's contribution
0.5	26.08.2019.	Aran Island's contribution
0.6	28.08.2019.	Contribution harmonization
0.7	04.09.2019.	Pupin's contribution
0.8	04.09.2019.	Tekniker's contribution
0.9	05.09.2019.	Pupin's contribution
0.10	05.09.2019.	Content formatting
0.11	06.09.2019.	Fenie's contributions
0.12	09.09.2019.	Editing and formatting
0.13	09.09.2019.	Tekniker's contribution
0.14	29.09.2019.	NUIG and Dexma contribution
0.15	25.09.2019.	Internal review by Tekniker
1.0	26.09.2019.	Final editing
1.1	30.09.2019.	Final version

EXECUTIVE SUMMARY

In this document on ‘Deployment of RESPOND cloud-based platform’, we summarize the outcomes from the Task 2.5 from Month 13 to Month 24 of the RESPOND project. The goal of this task was to undertake key activities towards deployment of RESPOND cloud-based platform and its core services at pilot sites. The activities have been carried out during second year of the project along with integration activities of Demand Response (DR) enabling components in WP3, WP4, and WP5. The goal was to deploy each of the underlying concepts, such as DR optimization, energy demand forecast, user interface, etc, which have been developed as core services of the RESPOND platform and their integration with hardware and software systems at pilot sites. All the DR enabling components will be deployed matching the specifications of reference system architecture and deployment plan which has been created in previous tasks of the WP2.

First, we briefly revisit the RESPOND cloud-based platform architecture with its main building blocks. Then, we move to the deployment of different analytic services. Next, we provide an overview of the visualisation layer which comprise web dashboard and mobile application. Finally, deployment of field level devices at the pilot sites is described.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	4
TABLE OF CONTENTS	5
LIST OF FIGURES	7
LIST OF TABLES	9
ABBREVIATIONS AND ACRONYMS	10
1. INTRODUCTION	11
1.1 RELATION TO OTHER RESPOND ACTIVITIES	11
1.2 REPORT STRUCTURE	11
2. RESPOND CLOUD PLATFORM ARCHITECTURE	12
2.1 OPENWHISK MICROSERVICE ARCHITECTURE	13
3. RESPOND CLOUD-BASED PLATFORM DEPLOYMENT	15
3.1 RESPOND PLATFORM CONTROL LOOP	15
3.1.1 DEPLOYMENT OF DR OPTIMIZATION SERVICE	16
3.1.2 DEPLOYMENT OF ENERGY CONSUMPTION FORECAST SERVICE	19
3.1.3 DEPLOYMENT OF ENERGY PRODUCTION FORECAST SERVICE	20
3.1.4 DEPLOYMENT OF OPTIMIZED CONTROL SERVICE	22
3.1.5 DEPLOYMENT OF USER BEHAVIOUR ADAPTER	23
3.1.6 DEPLOYMENT OF PREDICTIVE MAINTENANCE SERVICE	24
3.2 RESPOND PLATFORM VISUALIZATION	25
3.2.1 DEPLOYMENT WEB VISUALIZATION DASHBOARD	26
3.2.2 DEPLOYMENT OF MOBILE APPLICATION	28
4. DEPLOYMENT AT PILOT SITES	30
4.1 DEPLOYMENT AT MADRID PILOT SITE	30
4.1.1 CHALLENGE WITH THE DEVICES AND OTHER REASONS	34
4.1.2 THERMOSOLAR'S OGEMA ENERGY GATEWAY	38
4.1.3 WATER MONITORING SERVICE (FEN)	41
4.2 DEPLOYMENT AT AARHUS PILOT SITE	41
4.2.1 CHALLENGE WITH THE DEVICES AND OTHER REASONS	46
4.2.2 THE THERMOSTAT CHALLENGE	46
4.3 DEPLOYMENT AT ARAN ISLAND PILOT SITE	48

5. CONCLUSIONS	52
REFERENCES	53

LIST OF FIGURES

FIGURE 1: RESPOND PLATFORM ARCHITECTURE	12
FIGURE 2: APACHE OPENWHISK HIGH LEVEL ARCHITECTURE (SOURCE: OREILLY.COM)	14
FIGURE 3: RESPOND'S MEASURE-FORECAST-OPTIMIZE-CONTROL LOOP	15
FIGURE 4: ILLUSTRATION OF A DR EVENT LOAD DEVIATION	16
FIGURE 5: OPENWHISK KUBERNETES CLUSTER (SOURCE: IBM.COM)	18
FIGURE 6: DOCKER CONTAINERS ENVIRONMENT DEPLOYMENT	19
FIGURE 7: ENERGY CONSUMPTION FORECASTING SERVICE ENVIRONMENT DEPLOYMENT	20
FIGURE 8: RESPOND SERVICE ARCHITECTURE	21
FIGURE 9: THE BUILDING REDUCED ORDER MODEL	22
FIGURE 10: USER BEHAVIOUR ADAPTER ENVIRONMENT DEPLOYMENT	24
FIGURE 11: PREDICTIVE MAINTENANCE SERVICE ENVIRONMENT DEPLOYMENT	25
FIGURE 12: GATEWAY CONFIGURATION	26
FIGURE 13: LOCATIONS AND SUBLOCATIONS METADATA CONFIGURATION	27
FIGURE 14: HIERARCHY TREE	27
FIGURE 15: REST SERVICE API ENVIRONMENT DEPLOYMENT	28
FIGURE 16: MADRID PILOT SITE DEVICES SCHEMA	30
FIGURE 17: MADRID PILOT SITE DWELLING BLUEPRINT	31
FIGURE 18: ENERGOMONITOR'S POWER CONSUMPTION SENSOR INSTALLED	32
FIGURE 19: ENERGOMONITOR'S GATEWAY INSTALLED	32
FIGURE 20: ENERGOMONITOR'S DISPLAY INSTALLED	32
FIGURE 21: ENERGOMONITOR'S CO2 SENSOR INSTALLED	32
FIGURE 22: ENERGOMONITOR'S ELECTRICITY METERS SENSORS INSTALLED	33
FIGURE 23: SOLAR COLLECTORS INSTALLED	33
FIGURE 24: THERMOSOLAR SYSTEM DIAGRAM	34
FIGURE 25: TICKSCRIPT EXAMPLE (MEAN)	35
FIGURE 26: CHRONOGRAF ALERT RULE BUILDER (TYPE AND CONDITIONS)	36
FIGURE 27: CHRONOGRAF ALERT RULE BUILDER (HANDLERS AND MESSAGE)	37

FIGURE 28: RECEIVED ALERT EMAIL _____	37
FIGURE 29: ALERT AS TICKSCRIPT _____	38
FIGURE 30: SIEMENS RMS705B AND ZENNIO KNX-USB BRIDGE _____	39
FIGURE 31: GATEWAY PC CONNECTED TO THE KNX SWITCHBOX _____	39
FIGURE 32: RMS705B CONFIGURATION IN ETS _____	40
FIGURE 33: WATER MEASUREMENTS MESSAGE EXAMPLE _____	41
FIGURE 34: THE ORIGINAL PLAN FOR EQUIPMENT AVAILABLE IN ALL THE AARHUS HOUSES _____	42
FIGURE 35: PHOTO OF THE APARTMENT BLUEPRINTS WITH ALL 3 FLOORS _____	43
FIGURE 36: PHOTOS OF INSTALLATION OF DEVELCO SMART CABLE FOR THE DISHWASHER ON THE GROUND FLOOR _____	44
FIGURE 37: PHOTOS OF INSTALLATION OF DEVELCO SMART CABLE FOR THE WASHING MACHINE IN THE BASEMENT _____	44
FIGURE 38: PHOTO OF INSTALLATION OF THE DEVELCO SQUID.LINK GATEWAY IN THE LIVINGROOM ON THE GROUND FLOOR _____	44
FIGURE 39: PHOTO OF INSTALLATION OF THE KAMSTRUP MC403 WITH WMBUS (HEADMETER) IN THE BASEMENT _____	45
FIGURE 40: PHOTOS OF THE INSTALLATION OF THE DEVELCO SMART PLUG AS A WIRELESS EXTENDER IN THE BASEMENT _____	45
FIGURE 41: PHOTOS OF THE INSTALLATION OF DEVELCO TEMPERATUR/HUMIDITY SENSOR IN THE BATHROOM ON 1. FLOOR _____	45
FIGURE 42: PHOTOS OF THE INSTALLATION OF DEVELCO TEMPERATUR/HUMIDITY SENSOR (1,8 METER ABOVE THE FLOOR) IN THE BEDROOM ON 1. FLOOR _____	46
FIGURE 43: PHOTOS FROM THE INSTALLATION AND TEST OF THE SPIRIT THERMOSTAT _____	47
FIGURE 44: PHOTO OF THE DANFOSS ECO2 BLUETOOTH THERMOSTAT _____	47
FIGURE 45: SOME OF THE DEVELCO DEVICES THAT COULD BE INSTALLED AT THE PILOT. ABOVE – TEMPERATURE AND HUMIDITY SENSORS. BELOW – SMART PLUG ON PORTABLE HEATER. _____	49
FIGURE 46: THE DEVICES THAT ARRIVED IN THE FIRST ENERGOMONITOR SHIPMENT TO THE ARAN PILOT _____	50
FIGURE 47: ABOVE – THE SPECIALLY DESIGNED VERSION OF THE PLUGSENSE DEVICE. BELOW -THE AIRSENSE CO2 METER _____	51

LIST OF TABLES

TABLE 1: MADRID PILOT SITE DEVICES DESCRIPTION	31
TABLE 2: MADRID PILOT SITE COMMON AREAS DEVICES DESCRIPTION	33
TABLE 3: SOLAR SENSORS CONFIGURATION	40
TABLE 4: LIST OF DEVICES INSTALLED IN AARHUS PILOT SITE DWELLINGS	42

ABBREVIATIONS AND ACRONYMS

API	Application Program Interface
CDM	Canonical Data Model
DB	Database
DoW	Description of Work
DR	Demand Response
ETS	Engineering Tool Software
IP	Internet Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
MQTT	Message Queuing Telemetry Transport
PV	Photo-Voltaic
RES	Renewable Energy Sources
REST	Representational State Transfer
ROM	Reduced Order Model
STC	Solar Thermal Collector
TICK	Telegraf InfluxDB Chronograf Kapacitor
WP	Work Package

1. INTRODUCTION

In WP2, the goal is to define the system reference architecture of RESPOND system based on the requirements specified in WP1 and to conduct early deployment activities. In particular, WP2 will deliver architectural principles and reference architecture which will facilitate the deployment of Demand Response (DR) coordinated actions and the adoption by all the involved stakeholders. This document presents the description of the main outcomes of Task 2.5, which are provided in the form of software artefacts. As it has been described in the Description of Work (DoW), the activities within Task 2.5 aim to perform the deployment of RESPOND cloud-based platform and its core services at pilot sites. The goal of these activities is to ensure the proper assembly of the RESPOND solution's underlying concepts and their deployment in order to conduct optimized control actions under cooperative demand strategy.

Activities within this task have been carried out for all the pilots in parallel, having in mind particularities of each of them. This work has been performed in an iterative manner in order to ensure synchronization of activities of related WPs. In order to stay aligned with the operation scenarios defined for each pilot, this task has been performed in accordance with the overall DR requirements and technical characterization made in WP1. Finally, all the components that enable DR will be deployed matching the specifications of reference system architecture and deployment plan designed in Task 2.1 and Task 2.3, respectively.

1.1 RELATION TO OTHER RESPOND ACTIVITIES

Task 2.5, which has lasted from Month 13 to Month 24 of the project, takes the outcomes of all previous tasks within this Work Package (WP). In Task 2.1, a design of system reference architecture has been performed, that resulted in D2.1 [1]. Moreover, Task 2.2 dealt with seamless integration of core services of RESPOND platform and suggested their interaction and relations which are necessary in order to conduct cooperative demand management and optimal control strategy. In addition, the design of initial deployment plan has been performed within Task 2.3, whereas activities related to early deployment at the pilot sites have been performed in Task 2.4. Finally, RESPOND platform will be validated within WP6.

1.2 REPORT STRUCTURE

In this document, which reports the results of Task 2.5, we provide more details regarding the development and deployment of RESPOND cloud-based platform and its core services at pilot sites. In Section 2, a summary of RESPOND cloud platform is presented focussing on integration of RESPOND core services. Then, in Section 3, we present the activities related to cloud platform deployment where we provide more details related to RESPOND control loop and visualisation applications. Next, in Section 4, a brief summary of deployment activities at pilot sites is presented. Finally, in Section 5, we conclude the document with the main outcomes of Task 2.5.

2. RESPOND CLOUD PLATFORM ARCHITECTURE

In this section, a brief summary of the RESPOND platform and middleware architecture (described in D5.3 on RESPOND middleware deployment [2]) will be presented. As we can see in Figure 1, the components of the RESPOND system can be grouped into following groups:

- **Field level devices:** which comprise energy assets, sensors, smart meters and actuators deployed in the buildings with the corresponding gateways (Energomonitor, Develco and OGEMA)
- **Middleware:** consisting of Message Queuing Telemetry Transport (MQTT) broker and Data collection components (TICK stack)
- **Analytic services:** which provide the core value of the RESPOND platform by performing analysis of the collected data in order to provide results to be shown in visualisation applications
- **Visualisation layer:** which consists of a web dashboard for energy managers and the mobile app for household occupants.

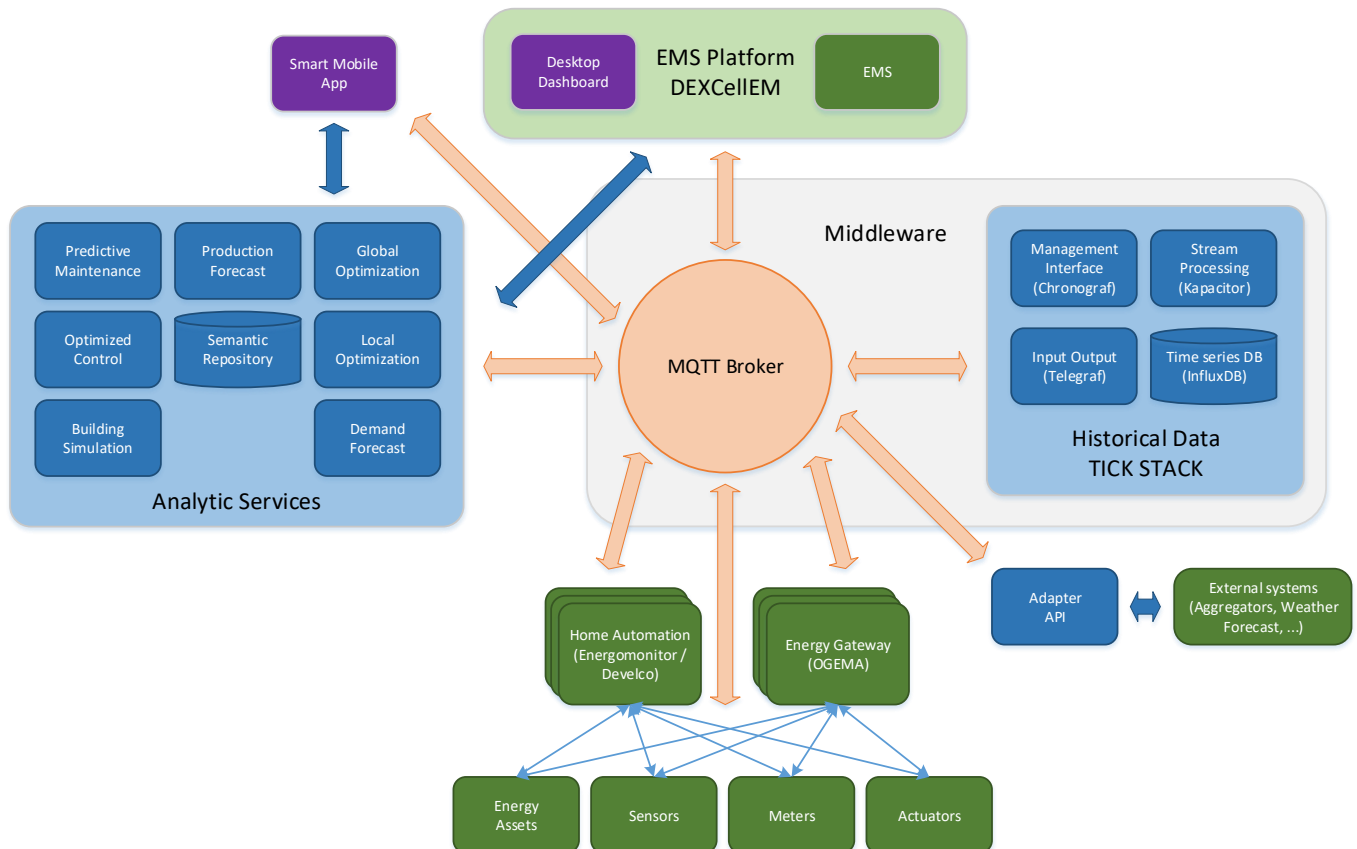


FIGURE 1: RESPOND PLATFORM ARCHITECTURE

In this document, we focus on and provide more details on the deployment of analytic services.

2.1 OPENWHISK MICROSERVICE ARCHITECTURE

Despite their benefit, microservice-based solutions often remain difficult to build using mainstream cloud technologies, often requiring control of a complex toolchain, and separate build and operations pipelines. Small and agile teams, spending too much time dealing with infrastructural and operational complexities (fault-tolerance, load balancing, auto-scaling, and logging), especially want a way to develop streamlined, value-adding code with programming languages they already know and that are best suited to solve particular problems.

Serverless computing encourages and simplifies developing microservice architecture solutions in order to decompose complex applications into small and independent modules that can be implemented, deployed, scaled, tested and monitored independently. It is essentially about reducing maintenance efforts to allow developers to quickly focus on developing value-adding code. Serverless computing refers to a model where the existence of servers is simply hidden from developers. I.e. that even though servers still exist developers are relieved from the need to care about their operation. They are relieved from the need to worry about low-level infrastructural and operational details such as scalability, high-availability, infrastructure-security, and so forth.

One of the initial leaders in open-source serverless computing was IBM, with OpenWhisk, which is now an Apache project. Apache OpenWhisk [5] is an open-source serverless platform, composed of microservices written in Scala. It is built on proven, solid open source foundations. The platform uses numerous external open-source projects, including NGINX, CouchDB, Kafka, Zookeeper, Redis, and Docker. OpenWhisk is distributed, flexible, extensible, and integrates with a variety of internal and external event sources. Custom runtime code is bundled and managed as platform services using Docker.

The modular and inherently scalable nature of OpenWhisk makes it ideal for implementing granular pieces of logic in actions. OpenWhisk actions are independent of each other and can be implemented using variety of different languages supported by OpenWhisk and access various backend systems. Each action can be independently deployed, managed and scaled. Interconnectivity between actions is provided by OpenWhisk in the form of rules, sequences and naming conventions. This bodes well for microservices based applications.

Apache OpenWhisk, as shown in Figure 2, works by executing functions (called actions) in response to events. Events can originate from multiple sources, including timers, databases, message queues, or websites like Slack or GitHub.

The OpenWhisk platform supports Action code written for any of its ever-growing, built-in language runtimes. The following is a list of currently supported languages: .Net, Go, Java, NodeJS, PHP, Python, Ruby, Swift, Ballerina, Rust. If there is a need for languages or libraries the current "out-of-the-box" runtimes do not support, there is an option to create and customize executables that run "black box" Docker Actions using the Docker SDK which are run on the Docker Runtime.

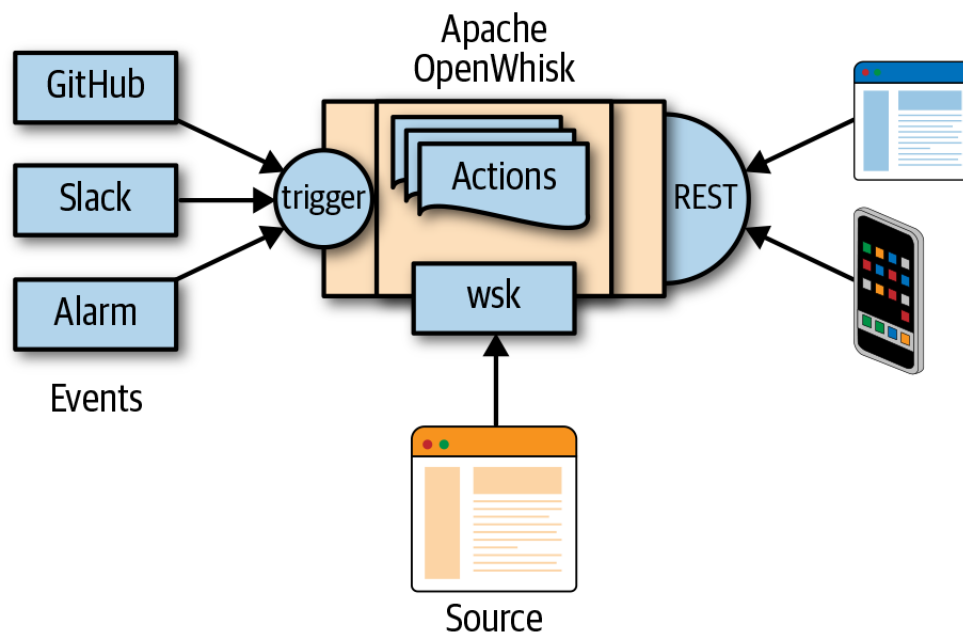


FIGURE 2: APACHE OPENWHISK HIGH LEVEL ARCHITECTURE (SOURCE: OREILLY.COM)

3. RESPOND CLOUD-BASED PLATFORM DEPLOYMENT

3.1 RESPOND PLATFORM CONTROL LOOP

One of the aims of the RESPOND platform is to enable the application of advanced energy services in order to provide users about more insights in relation to their energy consumption, help them to reduce the costs and engage them through both automatic and manual control of their most important energy assets. The aim of this section is to provide more details about the deployment of RESPOND analytical services by briefly overviewing their main functioning principles. Before going into details regarding each particular analytic service, we will present the overall control loop composed of measure-forecast-optimize-control main blocks, as can be seen in Figure 3.

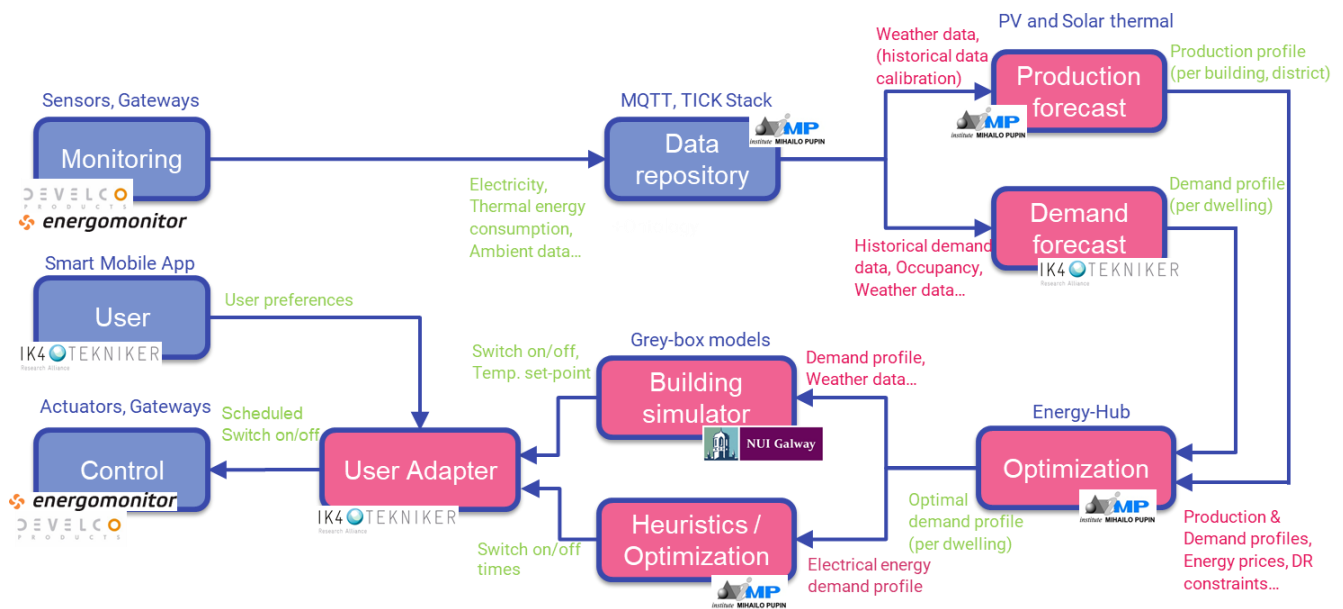


FIGURE 3: RESPOND'S MEASURE-FORECAST-OPTIMIZE-CONTROL LOOP

First, the monitoring layer collects measurements from various sensors and smart meters deployed in people home which are connected through a dedicated gateway. The collected data are stored in time-series database InfluxDB (part of the central Data repository), and made available to other parts of the RESPOND platform. Next, in the forecast stage, production and demand forecasters, which employ historical measurements in addition to weather data, provide production and demand profiles for different types (electricity, heating and cooling).

Based on the forecast results, the optimization seeks to find the optimal control and scheduling of available energy assets, while also considering other contextual information such as energy prices, import/export regulation, etc. As a result, the optimization stage provides optimal energy dispatching in a multi-carrier energy environment. Next, these results are translated into applicable control actions, which can be applied to existing energy assets. This translation is performed by employing both building simulation (for cooling and heating) and Heuristics/Optimization (for other energy assets). The control actions must be approved by the user. This confirmation is done by user preferences (stored in User Adapter) of explicit confirmation

thanks to the smart mobile assistant. User Adapter component can also provide stored user preferences to other parts of RESPOND platform, such as constraints on temperature setpoint, device operation intervals, etc. Finally, the control actions are sent to the installed actuators by complementing both manual and semi-automatic control approaches. In the sequel, deployment of all aforementioned services will be described in more details.

3.1.1 DEPLOYMENT OF DR OPTIMIZATION SERVICE

The DR optimization service developed for the RESPOND project represents a holistic tool for energy management of aggregated load profiles on a neighbourhood level. In other words, the loads of multiple individual consumers (several apartments, a couple of houses, an entire building, etc.) are joined into a single profile which should be governed in a DR environment where changes in energy source prices and explicitly defined DR events are supposed to reshape the load to a certain extent to provide a more stable system and facilitate energy and monetary savings. The optimizer relies on the Energy Hub concept model that describes energy flow through a system in several stages (power import, input storage, input transformation, conversion, export, output transformation, output storage, and load). For the RESPOND project, the model is extended with advancements in load manipulation features and the resulting model is solved using mixed-integer linear programming engine, or IBM ILOG Optimization Studio's CPLEX to be more precise. Two main options are provided for load manipulation and DR event enforcement: price-based DR and explicit DR. In the case of price-based DR, the load naturally inversely follows the price profile, i.e. when the prices are low, the load should increase to minimize costs for end users, and vice-versa. Therefore, implicit DR events can be obtained by creating time frames with extreme prices. Alternatively, explicit DR events can be defined in which the difference between the requested load profile and the optimized one are penalized accordingly, as illustrated in Figure 4. An integral constraint maintains the total amount of load for a given period (day/night, daily, weekly, etc.) while load tolerance bounds ($\pm 10\%$, $\pm 20\%$) simulate real-world load elasticity that can be obtained by residential users.

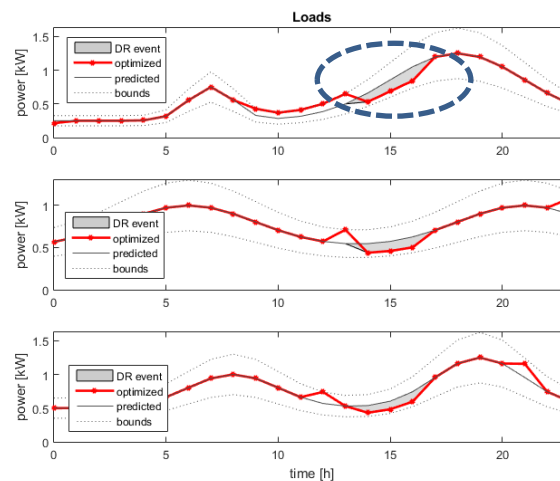


FIGURE 4: ILLUSTRATION OF A DR EVENT LOAD DEVIATION

The optimization service receives inputs in the form of the predicted load profile (from the energy consumption forecasting service) and predicted renewable production (from the energy production forecasting service), and assumes a predefined topology that depicts a certain pilot neighborhood and

performs the necessary optimization. After the optimized load profile is obtained, the differences between the predicted, optimized and the real (measured) profile from end users can be employed to provide user-tailored requests for load increase or decreases and can even be correlated with specific appliance measurements in the project's InfluxDB to obtain appliance-tailored requests for activations or deactivations.

3.1.1.1 OPENWHISK DEPLOYMENT OF DR OPTIMIZATION SERVICE

Apache OpenWhisk is a serverless, open source cloud platform that executes functions in response to events at any scale. As a distributed software, OpenWhisk can be deployed as a container cluster where each of the components can be run in its own container. This allows users to deploy OpenWhisk on any platform, which supports container management and orchestration.

Kubernetes is a well-known container orchestration tool, which can be used to deploy container-native applications. Deploying OpenWhisk over Kubernetes can leverage the capabilities provided by Kubernetes to better control and manage OpenWhisk containers, which can result in a stable OpenWhisk runtime [6]. One of more Kubernetes-native way, to deploy OpenWhisk over Kubernetes is using Helm package manager.

Helm is a tool for managing Kubernetes charts, while charts are packages of pre-configured Kubernetes resources. In other words, users can write charts, which are in template format, to define a set of Kubernetes resources (each resource stands for a component of the application), and use Helm to deploy the charts over a Kubernetes cluster.

To deploy OpenWhisk over Kubernetes, charts need to be designed for each of the necessary components of the OpenWhisk runtime:

- **Kafka Node:** Kafka provides the messaging service for the OpenWhisk controller and invoker. In a Kubernetes cluster, a StatefulSet can be deployed to run Kafka queue and this StatefulSet will be bound with a corresponding Service in Kubernetes to provide a public service to the controller node and invoker node. Considering running Kafka requires the Zookeeper runtime, a Deployment for holding Zookeeper and bind a Service to this Zookeeper Deployment is required to provide the public service for Kafka.
- **Couchdb Node:** Couchdb provides a database service for the OpenWhisk controller and invoker. In the Kubernetes cluster, a Deployment is needed to run Couchdb and bind the corresponding Service to provide a public service endpoint for the controller node and invoker node.
- **Controller Node:** this is a standard OpenWhisk component that depends on a Kafka service and a Couchdb Service. To deploy the controller, a StatefulSet should be created (so that multi controllers can have an index to identify themselves) in a Kubernetes cluster.
- **Invoker Node:** this is a standard OpenWhisk component that depends on a Kafka service and a Couchdb Service. To deploy the invoker, a Deployment is created (so that users can auto-scale invokers in the future) in a Kubernetes cluster. All of the action containers are created and controlled by the invoker on a Kubernetes node and these containers are not under the control of Kubernetes.
- **Nginx Node:** this is the web server which forwards the outside client request to the controller node. To deploy an Nginx node, a Deployment in Kubernetes is created to run the Nginx server and create a Kubernetes Service of type NodePort to provide public IP/Port so that an outside client can visit the OpenWhisk cluster through this Nginx server.

Based on this design, Helm charts are created for the above components. Each of the above components can be created as a single chart. Once all the necessary charts are collected together, a single Helm command can be used to deploy OpenWhisk.

Based on the above Helm charts, a minimum deployment of OpenWhisk over Kubernetes is shown in the Figure 5.

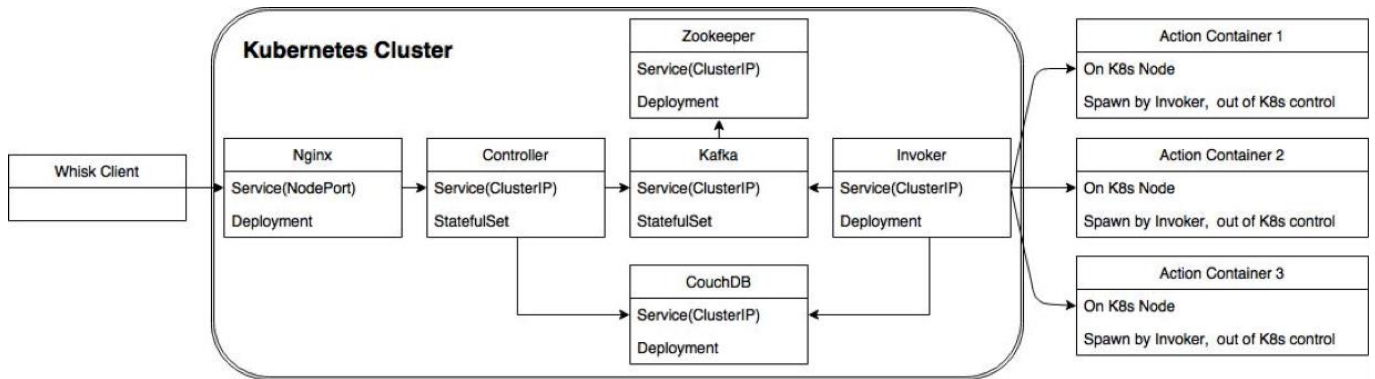


FIGURE 5: OPENWHISK KUBERNETES CLUSTER (SOURCE: IBM.COM)

Once the OpenWhisk platform has been installed, the advanced energy services can be deployed. The services are composed of multiple Python files, so a zip file containing all the files needs to be created. Python libraries can be imported into the Python runtime in Apache OpenWhisk by including a virtualenv folder in the deployment archive, but this approach does not work when the deployment archive would be larger than the action size limit (48MB). Instead a custom Docker runtime image is needed which extends the project's Python runtime image and runs pip install during the container build process. The libraries are then pre-installed into the runtime and can be excluded from the deployment archive. Following is the example of Dockerfile which is used to create the custom runtime.

```
FROM openwhisk/python3action:latest
# Set the working directory to /app
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt
```

This Dockerfile is used to create a Docker image:

```
docker build -t respond_dr_optimization_runtime
```

Newly created docker image should be tagged with the Docker Hub username and pushed to Docker Hub:

```
docker tag respond_dr_optimization_runtime dpaun/respond_dr_optimization_runtime
docker push dpaun/respond_dr_optimization_runtime
```

Finally, the services are deployed as OpenWhisk Actions, executing wsk commands, such as:

```
wsk -i action create RespondDROptimization --docker
dpaun/respond_dr_optimization_runtime
/home/respondservice/DROptimization/droptimization.zip
```

3.1.2 DEPLOYMENT OF ENERGY CONSUMPTION FORECAST SERVICE

The energy consumption forecasting service has been deployed in an Apache Tomcat Server docker container. At the same time, the docker container has been deployed in a virtual machine.

On the one hand, a virtual machine is a computer architectures-based software that simulates a computer system providing the same functionality as a physical computer. The main advantages of the virtual machines are the possibility to simulate any kind of operating system with custom requirements in the same machine from Linux to Windows depending on the user needs and the possibility to simulate multiple and different operating systems on the same physical machine at once. The operating system used is an Ubuntu 16.04.3.

On the other hand, a docker container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Available for Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. As said before, a docker container is used to deploy an Apache Tomcat Server on the Ubuntu 16.04.3 virtual machine.

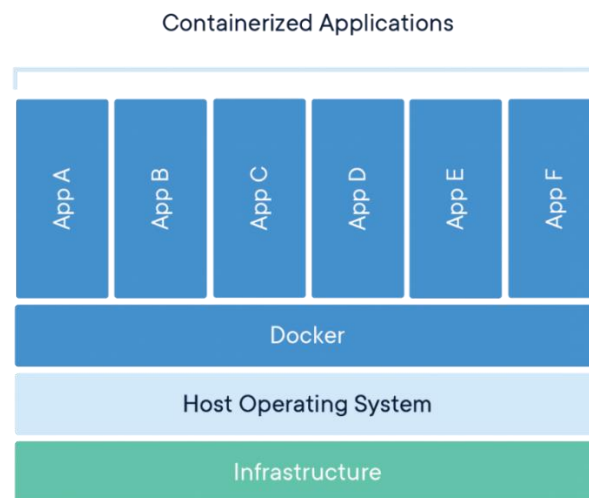


FIGURE 6: DOCKER CONTAINERS ENVIRONMENT DEPLOYMENT

Finally, an Apache Tomcat Server 8.5.33 is used to deploy the energy consumption forecasting service. Apache Tomcat is an open source software implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies that allows to deploy Java based applications.

The energy consumption forecasting service has the Virtuoso Server and an InfluxDB timeseries database as inputs and publishes its results to the MQTT Mosquitto message broker. Figure 7 depicts the energy consumption forecasting service environment deployment.

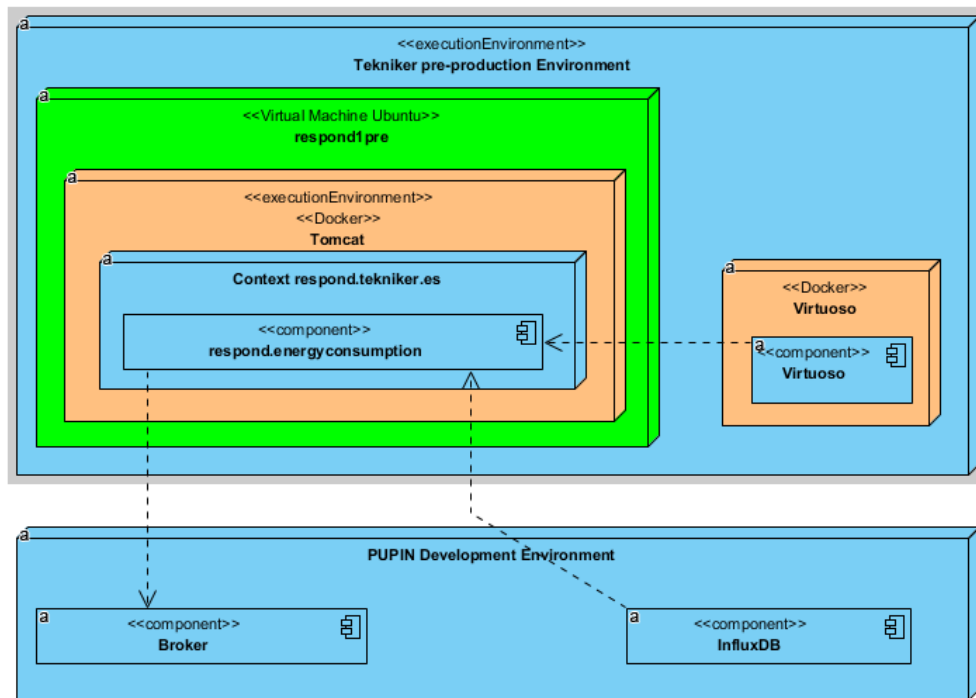


FIGURE 7: ENERGY CONSUMPTION FORECASTING SERVICE ENVIRONMENT DEPLOYMENT

3.1.3 DEPLOYMENT OF ENERGY PRODUCTION FORECAST SERVICE

In order to help ecological interest, it is necessary to increase share of the production of the renewable energy sources (RES) in the electricity market. However, all RES technologies highly depend on the weather conditions, and therefore influence grid stability, which is the main problem of using RES technologies.

Therefore, the optimizer, necessary for adapting the user demand with the stochastic nature of the RES, has been developed as a part of Task 4.2 and 4.3. However, the optimization can be carried out only if the forecasted production and demand are given as the input values. Accordingly, as a part of Task 4.4, energy production and demand forecasters are going to be developed and deployed in the RESPOND platform. Due to the fact that the optimization horizon is one day, and that it considers hourly resolution, the outputs of the forecasting services are expected to be arrays with 24 elements, each corresponding to one hour of the day.

For the energy production forecaster, the use of machine learning techniques has been considered, as it was shown in literature that they achieve the highest performance when necessary data for training process is given, which is the case in this situation. For the purpose of Aran Island and Aarhus pilot sites with Photo-Voltaic (PV) panels, neural networks were used, while for Solar Thermal Collector (STC) production forecasting, random forest algorithm has been chosen. Nonetheless, all of them were implemented in Python using TensorFlow and Scikit-Learn libraries, in order to provide easy integration.

The trained models have then further been used as a part of forecasting service as the core component. Namely, the master, shown in Figure 8, is supposed to run the forecaster at every point of time needed. Then, the necessary inputs are collected by the service (forecasted weather parameters collected from the

Weatherbit service and stored in MySQL Database (DB) and additionally previous production in case of STC) and the production estimation in form of 24 element array is given and stored further in MySQL data base, from where it can be taken either for visualization or optimization purposed. The final service deployment on server has been carried out using Open Whisk.

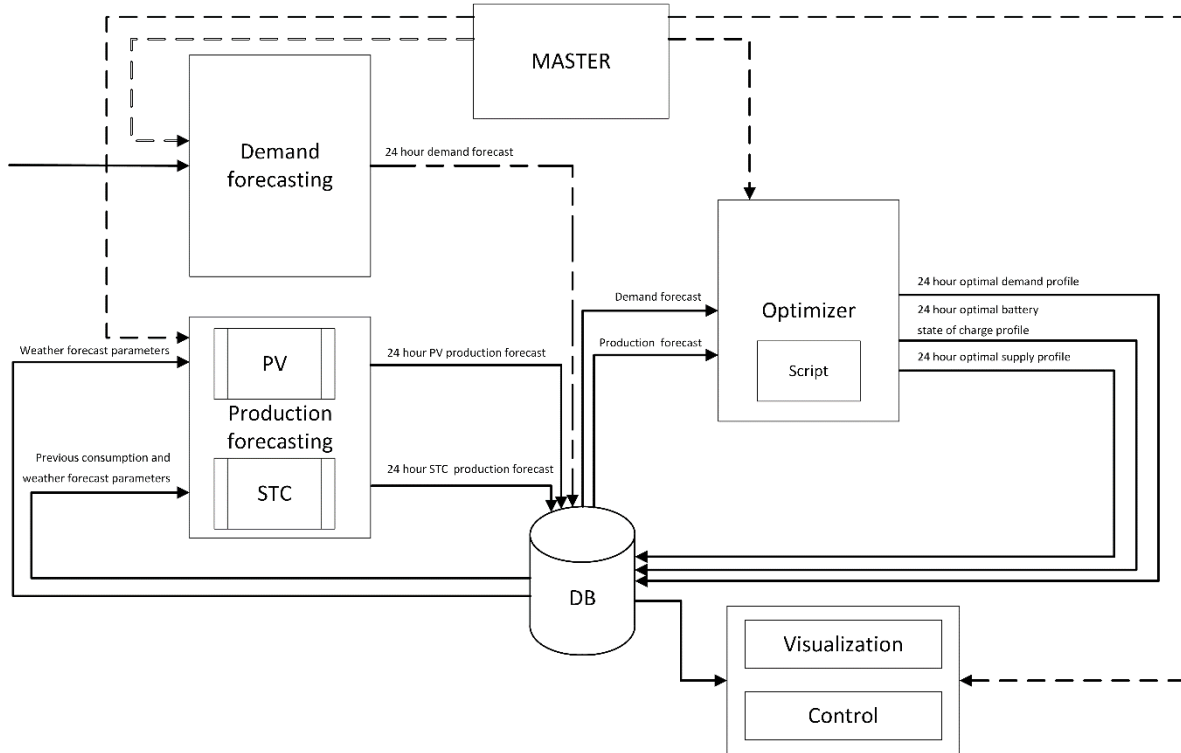


FIGURE 8: RESPOND SERVICE ARCHITECTURE

Energy production forecast service is deployed in a similar way as DR optimization service. With the Apache OpenWhisk platform deployed, Python files are packed into a zip file, a custom Docker runtime image is built, extending the Python runtime image with necessary external libraries, and finally OpenWhisk action is created.

Build a Docker image:

```
docker build -t respond_energy_production_forecast_runtime
```

Tag the image with the Docker Hub username and push it to Docker Hub:

```
docker tag respond_energy_production_forecast_runtime  
dpaun/respond_energy_production_forecast_runtime  
docker push dpaun/respond_energy_production_forecast_runtime
```

Create OpenWhisk action, executing wsk command:

```
wsk -i action create RespondEnergyProductionForecast --docker  
dpaun/respond_energy_production_forecast_runtime  
/home/respondservice/EnergyProductionForecast/energyproductionforecast.zip
```

3.1.4 DEPLOYMENT OF OPTIMIZED CONTROL SERVICE

Optimized control services on RESPOND platform is referred to the translation of results from the optimization service into optimal control actions related to the building heating/cooling systems.

In particular, the developed service is based on a Reduced Order Model (ROM), which is a simplified model of a building and its energy systems able to reproduce the effect of heating/cooling control actions on indoor thermal comfort (mean indoor air temperature value).

The Building Reduced Order Model (ROM) is developed using the Modelica language. The Modelica Standard library and IBPSA Project 1 Library (2018) have been used to develop the elements of the model.

The ROM is composed of the following four main components:

- 1) Internal Gains
- 2) Heating and Cooling
- 3) Building
- 4) Weather

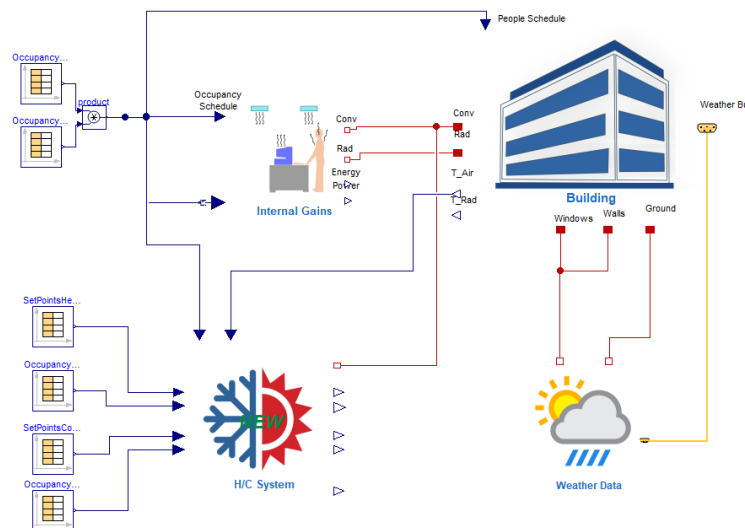


FIGURE 9: THE BUILDING REDUCED ORDER MODEL

The methodology to develop the RESPOND service is composed by three main steps:

1. The Tool for the ROM parameters estimation which receives building data as input;
2. A .fmu file of the Modelica model's described in the section above;
3. The ROMFit Python script for the model simulation and the automatic calibration.

A Tool has been created to estimate the parameters needed for the model. The information needed by the Tool can be simply obtained during the tendering phase through data collection and during on-site site surveys. In case of missing data, standardized packages and associated parameter values have been provided to support good model parameters estimation.

Since the model considers the main physical dependencies among the variables, the calibration phase is much simpler compared to a Whole Building Energy model. A dedicated Python script (ROMFit) has been developed to simplify the parameter insertion and increase the speed and accuracy of the calibration

procedure utilising an FMU file (Functional Mock-up Interface, 2018) generated from JModelica.org (2018). By using ROMFit the model is simulated and then, after the first simulation, it adjusts the uncertain parameters and simplifies the model calibration and its verification using real data collected by the RESPOND platform. The model is considered calibrated only if the model forecasting performances are within verified statistical values from comparison with real measured data.

Once the model is calibrated a python function is generated and uploaded to the RESPOND platform.

The model will be used as RESPOND service using the following inputs:

1. Optimal thermal energy demand profile per dwelling for the next day (RESPOND Optimization service);
2. Forecasted thermal energy demand profile per dwelling for the next day (RESPOND Demand forecast service);
3. Weather data forecast profile per location for the next day (RESPOND production forecast service)
4. Thermal comfort settings per hour/day/week used by the dwelling owner as hard constraints for the service (focus group interview);

The model will use the inputs to generate several scenarios in terms of heating/cooling control actions (i.e. shifting the heating profile from 6 am to 8 am, instead of 7 am to 9 am) to reproduce the optimized thermal energy profile for the next day, provided by the optimization service.

In case the service is able to find a proper scenario able to be compliant with both, the hard constraints defined by the dwelling owner and the optimized thermal energy profile, a message, using the RESPOND app, will be sent to the dwelling owner to perform (manually or automatically) the DR scenario during the next day. In case the service is not able to find a successful solution, the DR scenario will not be performed on the next day.

3.1.5 DEPLOYMENT OF USER BEHAVIOUR ADAPTER

The user behaviour adapter has been deployed in the same environment as the energy consumption forecasting service. That is, it has been deployed in an Apache Tomcat Server 8.5.33 docker container in an Ubuntu 16.04.3 virtual machine.

Furthermore, its inputs are retrieved from the Virtuoso Server and an InfluxDB timeseries database, and its results are sent to the MQTT Mosquito message broker. User preferences are registered in a PostgreSQL database and retrieved when they are needed by the user personal assistant or other service.

Figure 10 depicts the user behaviour adapter environment deployment.

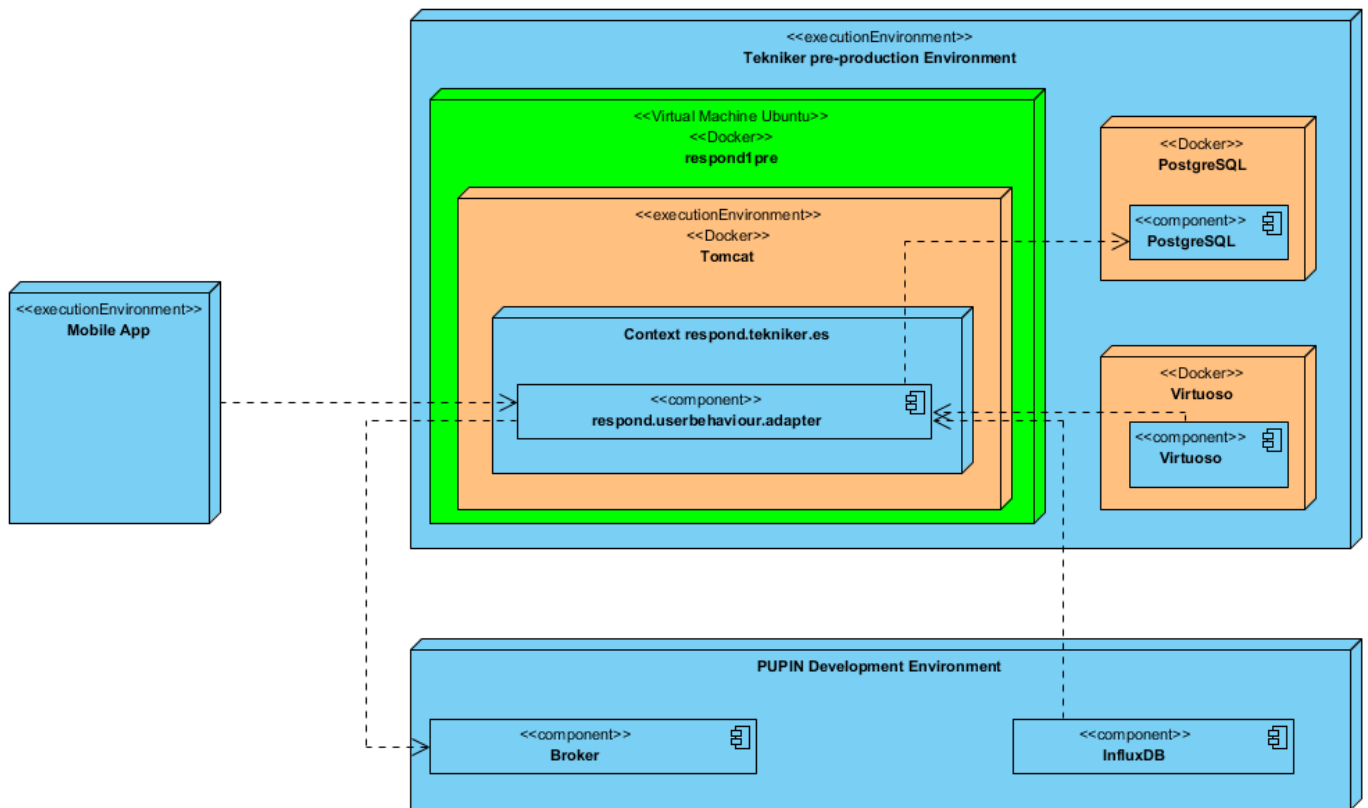


FIGURE 10: USER BEHAVIOUR ADAPTER ENVIRONMENT DEPLOYMENT

3.1.6 DEPLOYMENT OF PREDICTIVE MANTENAINCE SERVICE

The predictive maintenance service has been deployed in the same environment as the energy consumption forecasting service and the user behaviour adapter. It has been deployed in an Apache Tomcat Server 8.5.33 docker container in an Ubuntu 16.04.3 virtual machine.

Furthermore, its inputs are retrieved from the Virtuoso Server and an InfluxDB timeseries database, and its results are sent to the MQTT mosquito message broker.

Figure 11 depicts the predictive maintenance service environment deployment.

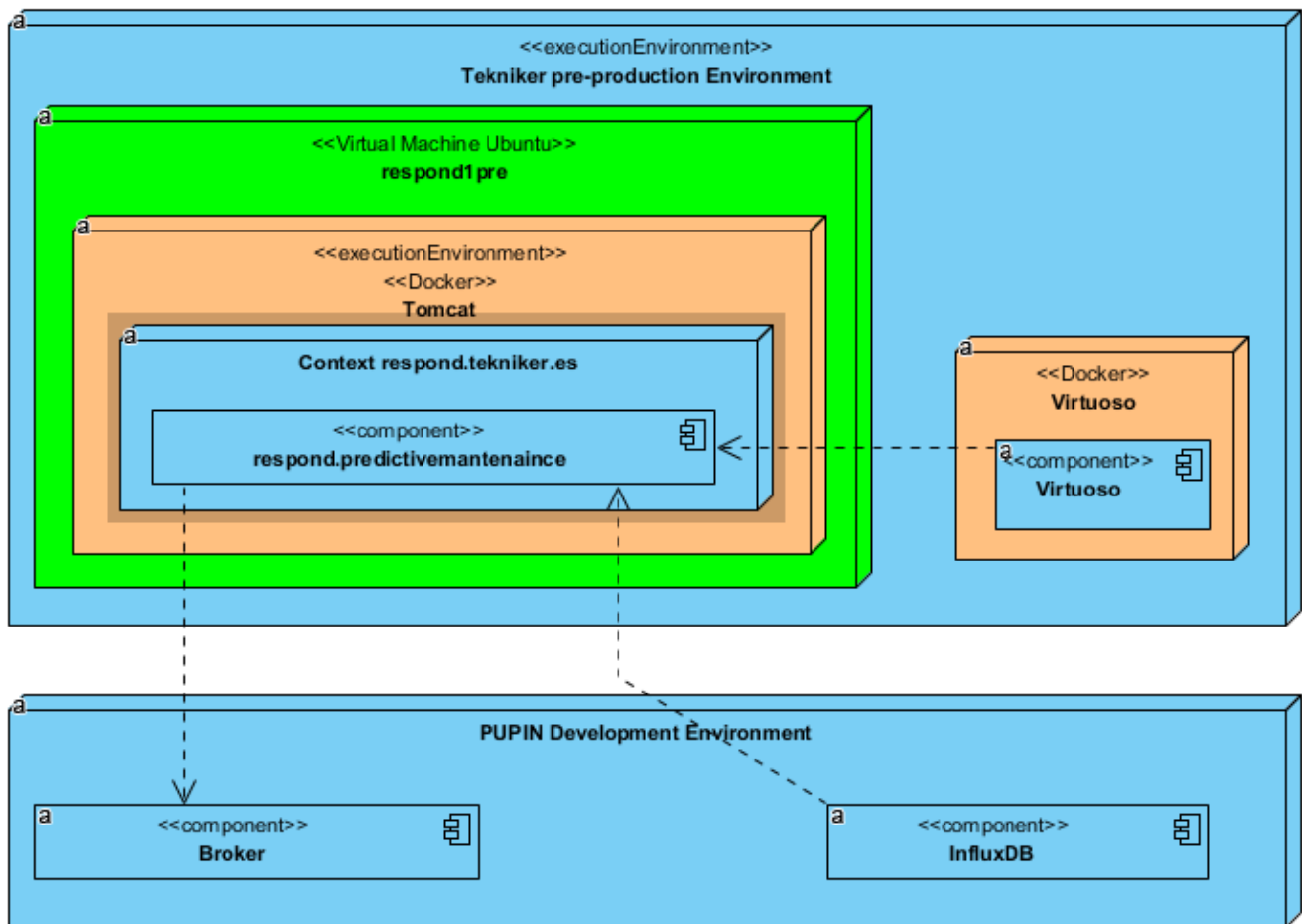


FIGURE 11: PREDICTIVE MAINTENANCE SERVICE ENVIRONMENT DEPLOYMENT

3.2 RESPOND PLATFORM VISUALIZATION

The core value of RESPOND project lies in the aforementioned analytics services. Nevertheless, in order to provide a meaningful interface so that end-users can use RESPOND platform services during daily activities, meaningful visualisation components have to be deployed. In RESPOND, depending on the type of end users, two visualisation components have been envisioned:

- Web visualisation dashboard, aimed at energy/building manager end users that show the overall building performance KPIs and more detailed reports.
- Mobile application, aimed primarily and building occupants, which will enable two-way interaction between end-user and the RESPOND platform in an intuitive and engaging way.

3.2.1 DEPLOYMENT WEB VISUALIZATION DASHBOARD

RESPOND project in comparison with most of the projects in Dexma includes a high-resolution range of metadata due to the granularity of the information and the number of dwellings. For this reason, a big effort has been made from all the partners to join efforts and gather all the data in a standardized format.

With this data Dexma operations team was able to configure the project account efficiently.

The steps followed to deploy the platform were:

- Gateway configuration
- Location configuration
- In-bulk files configuration
- Device validation and extra metadata insertion
- Accounts configuration

Gateway configuration:

The first step is to subscribe to the different devices' topic in the MQTT broker. To do so the gateway was configured on the platform. In Figure 12, the gateway configuration screen can be observed.

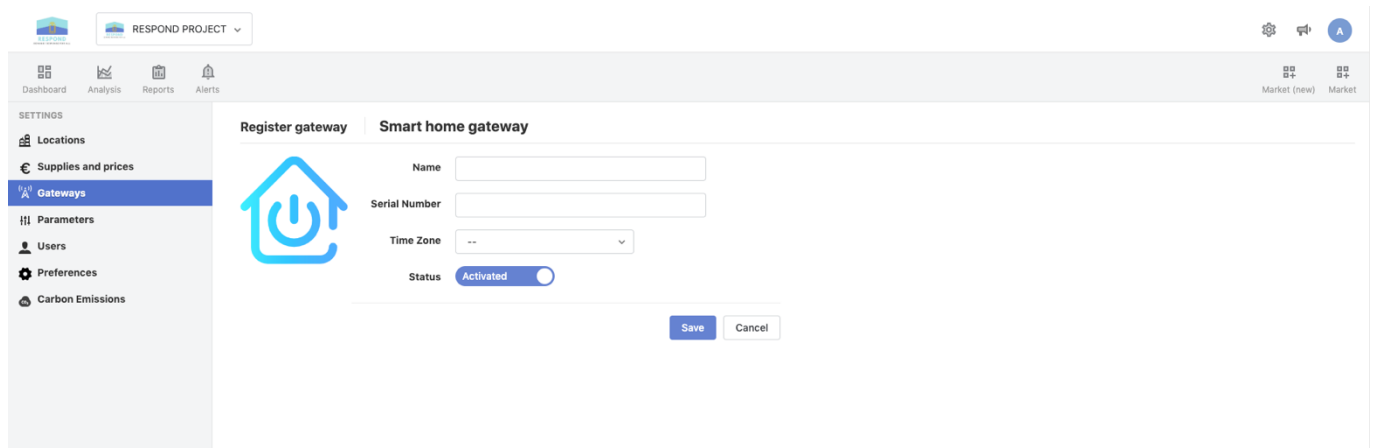


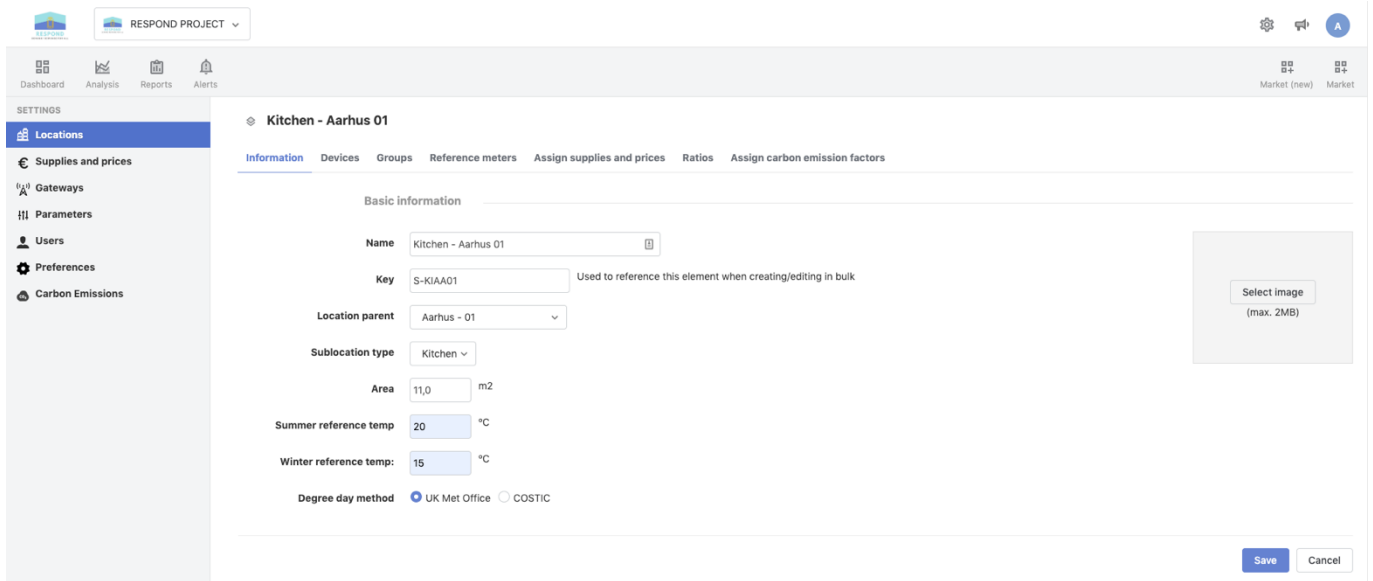
FIGURE 12: GATEWAY CONFIGURATION

Location configuration:

Once the gateways were configured, the information from the pilot sites started to get stored in DEXCell data base. Then the creation of locations was the next task. As it has been said, the project granularity follows the following pattern:

Pilot (zone) → Building (zone) → Dwelling (location) → Room (sublocation)

For each of the locations and sublocations metadata was configured, as it is presented in Figure 13. Metadata includes: i) Locations name, ii) Location type, iii) Surface, iv) summer reference temperature, v) winter reference temperature, and vi) Location address.



Kitchen - Aarhus 01

Information Devices Groups Reference meters Assign supplies and prices Ratios Assign carbon emission factors

Basic information

Name: Kitchen - Aarhus 01

Key: S-KJAA01 (Used to reference this element when creating/editing in bulk)

Location parent: Aarhus - 01

Sublocation type: Kitchen

Area: 11,0 m2

Summer reference temp: 20 °C

Winter reference temp: 15 °C

Degree day method: ☒ UK Met Office ☐ COSTIC

Save Cancel

FIGURE 13: LOCATIONS AND SUBLOCATIONS METADATA CONFIGURATION

The final result can be observed in the hierarchy tree in Figure 14.

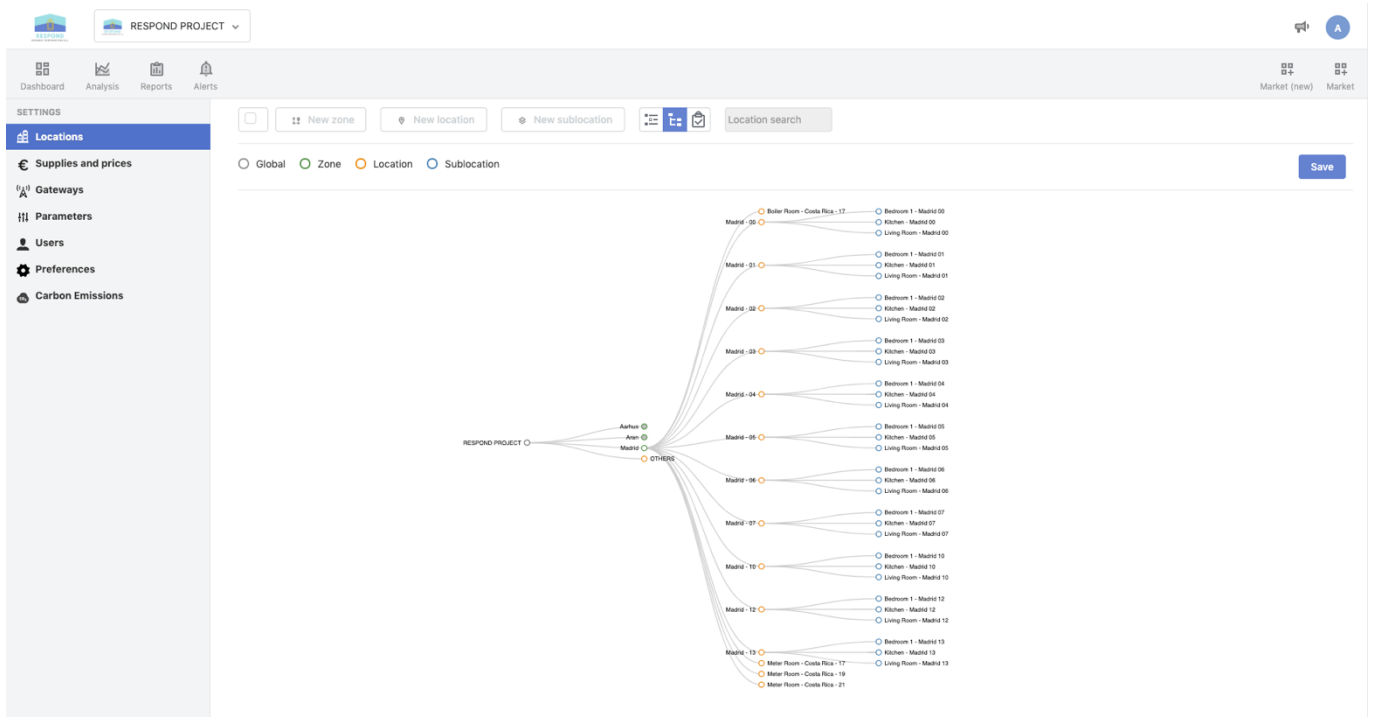


FIGURE 14: HIERARCHY TREE

In-bulk files configuration:

Once the locations and sublocation were configured, an excel file is download from DEXCell in which the operation was able to match the account devices with their location and sublocations. This file is sent to dev-ops team, in charge of uploading the new configuration.

Device validation and extra metadata insertion:

In order to validate if all the registered devices are in the platform, help was asked to the pilot coordinators. So Dexma operations team work jointly with them validating each of the locations and updating the data point files which summarizes the device status.

Also, extra metadata such energy cost was configured during this stage.

Accounts configuration:

The account configuration task was divided in two stages. The first step included the project partners accounts and was done at the beginning of the project. This way they could help with the configuration and also test functionalities related with the Application Program Interface (API).

The second stage will be carrying out before the platform deployment in the pilots, in which the Dexma operation team will configure the accounts for each of the inhabitants for the project pilot dwellings, along with the access permissions. That will be needed for data security reasons.

3.2.2 DEPLOYMENT OF MOBILE APPLICATION

The Mobile App Interacts with a REST Service API to retrieve user's data about their apartments.

The REST Service API has been deployed in an Apache Tomcat Server docker container that is also deployed in an Ubuntu virtual machine. Figure 15 depicts the REST Service API environment deployment.

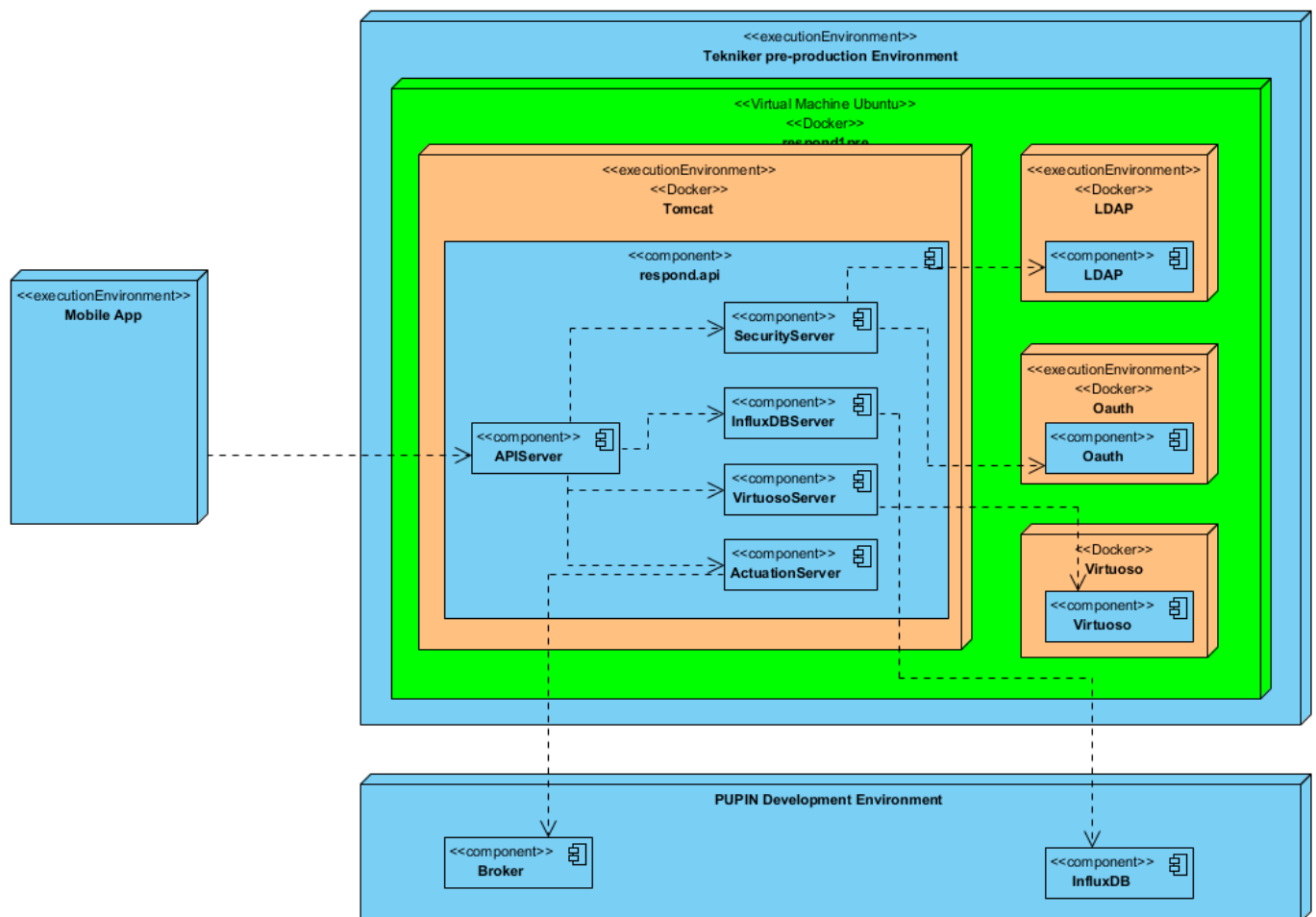


FIGURE 15: REST SERVICE API ENVIRONMENT DEPLOYMENT

The REST Service API works as a Proxy Server whose main objective is to control which data is accessed and by who. That way, only authorized users will be able to get authorized data.

At the same time, the REST Service API is divided in multiple servers with different objectives. Firstly, the API Server is in charge of the session management. Secondly, the Security Server is responsible for controlling that only authorized users can access their data by interacting with the Lightweight Directory Access Protocol (LDAP) and OAuth Server that is deployed in the same machine by using docker containers. In that way, the Security Server providing tokens allows the Mobile App not to use user's credentials every time a request takes place (only at the beginning). Thirdly, the InfluxDBServer and the VirtuosoServer are in charge of providing data from InfluxDB database and Virtuoso Server respectively once the APIServer has verified that the user is authorized for it. Finally, the ActuationServer allows the Mobile App to send commands to the MQTT mosquitto message broker for executing actions on devices.

4. DEPLOYMENT AT PILOT SITES

4.1 DEPLOYMENT AT MADRID PILOT SITE

In the Spanish pilot site, eleven dwellings have been provided with the necessary devices to both, monitor consumptions and indoor climate variables, and also actuate over some devices when necessary within the DR framework.

The schema below shows the devices deployed per dwelling and the variable measured or the appliances controlled. An Energomonitor gateway has been installed to centralize communications with the rest of sensors/actuators and to provide internet connection in order to allow external access to and from RESPOND server. Furthermore, regarding the comfort variables, an air quality sensor for measuring temperature, humidity and CO₂ has been deployed, as well as a digital display to measure temperature and humidity, and a thermometer. The whole house electricity consumption is monitored by a sensor located in the electricity supplier company meter. In addition to the total house consumption, there is also a clamp to measure AC consumption and two smart plugs for washing machine and dishwasher. These last two can not only measure power but also actuate like an on/off button remotely activated when necessary.

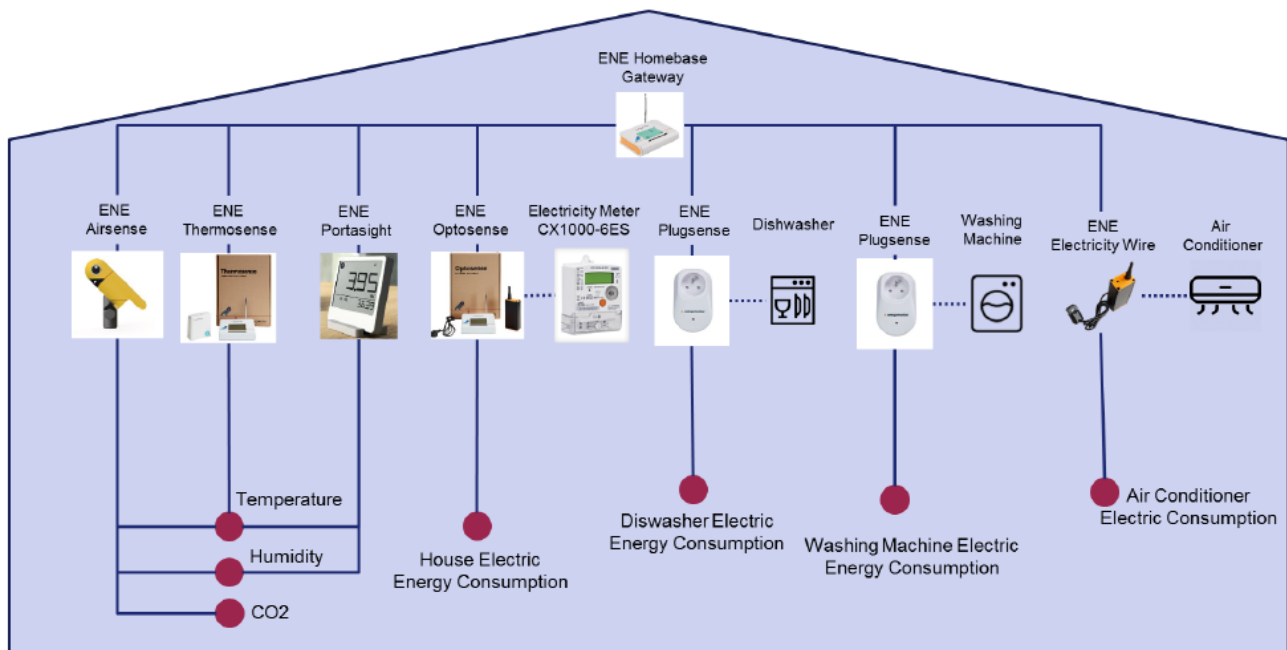


FIGURE 16: MADRID PILOT SITE DEVICES SCHEMA

The following table details the function of each element, alongside with the location within the dwellings which is also showed in the blueprint of one of Madrid pilot houses.

TABLE 1: MADRID PILOT SITE DEVICES DESCRIPTION

Device installed	Function	Location
Energomonitor Homebase	Gateway	Living room
Energomonitor Optosense	Monitoring of electric energy consumed	Electrical meters room in the building basement
Energomonitor AirSense	Monitoring of the humidity, temperature and CO2 of the room in which it is installed	Living room
Energomonitor Thermosense	Monitoring of the temperature of the room in which it is installed	Bedroom
Energomonitor Portasight	Monitoring of the humidity and temperature of the room in which it is installed	Kitchen
Energomonitor Plugsense 1	Monitoring of the dishwasher's power consumption, and controlling the dishwasher's activation according to DR actions	Kitchen
Energomonitor Plugsense 2	Monitoring of the washing machine's power consumption, and controlling the washing machine's activation according to DR actions	Kitchen
Energomonitor Powersense	Monitoring of the air conditioner's power consumption	Electrical panel

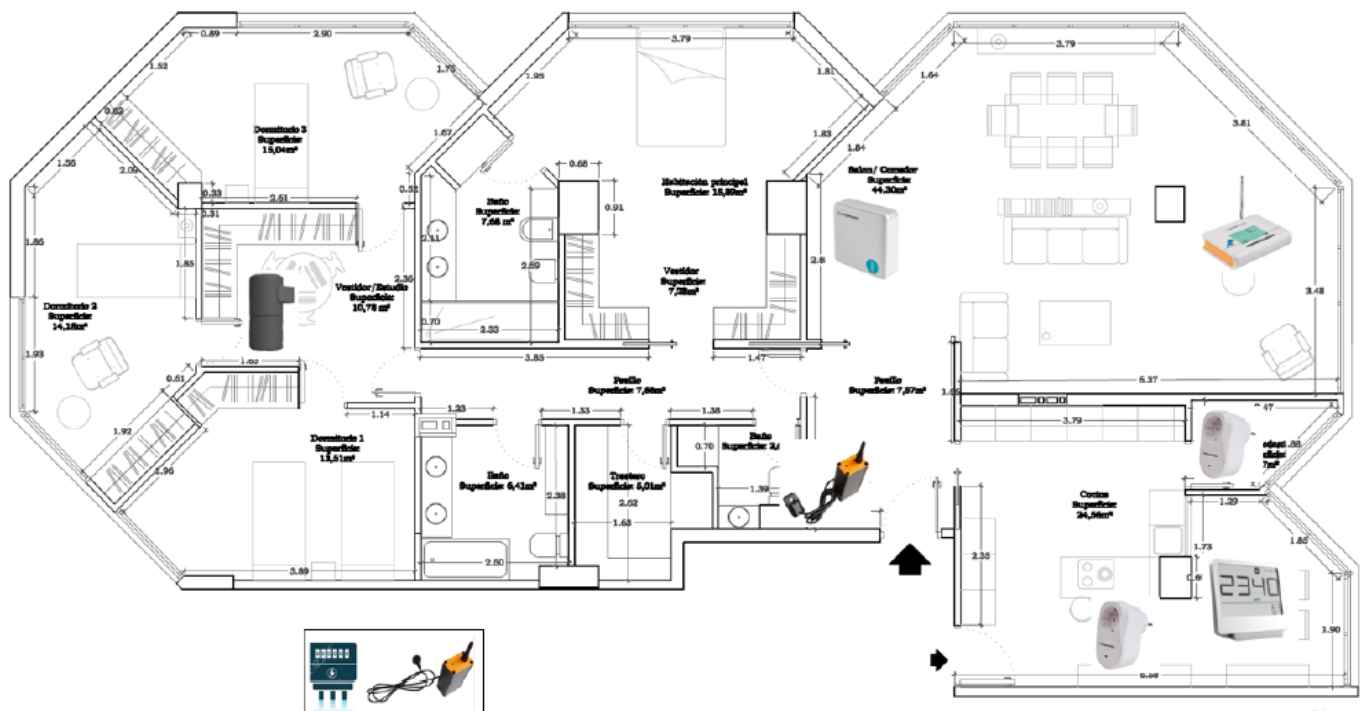


FIGURE 17: MADRID PILOT SITE DWELLING BLUEPRINT

In figures below, we present some photo examples of the devices deployed within the dwellings of the Madrid pilot site.

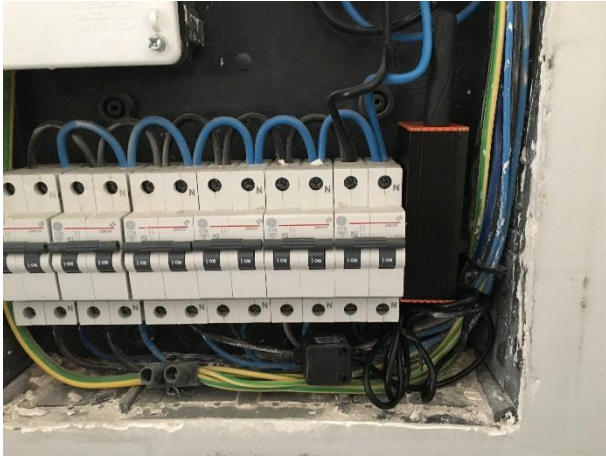


FIGURE 18: ENERGOMONITOR'S POWER CONSUMPTION SENSOR INSTALLED



FIGURE 19: ENERGOMONITOR'S GATEWAY INSTALLED



FIGURE 20: ENERGOMONITOR'S DISPLAY INSTALLED



FIGURE 21: ENERGOMONITOR'S CO2 SENSOR INSTALLED

With regards to common areas of the three building of the pilot, the following devices have been installed: an Energomonitor gateway in each of the electricity meters rooms to provide connectivity to the Energomonitor Optosense sensors deployed in these rooms that collect house total consumptions. Furthermore, a thermosolar system has been installed with the purpose of studying the inhabitant's habit changes when receiving information about real local RES generation. These devices are encompassed in the Table 2 below.

TABLE 2: MADRID PILOT SITE COMMON AREAS DEVICES DESCRIPTION

Device installed	Function	Location
Energomonitor Homebase	Gateway	Electrical meters room in the building basement
Energomonitor Optosense	Monitoring of electric energy consumed	Electrical meters room in the building basement
Thermosolar system	Generate DHW to test habits changes in inhabitants related to solar production	Rooftop and central boiler room

In the following figures, we present some photos that illustrate the work carried out:



FIGURE 22: ENERGOMONITOR'S ELECTRICITY METERS SENSORS INSTALLED



FIGURE 23: SOLAR COLLECTORS INSTALLED

In addition, regarding the thermosolar system, in Figure 24 we provide the installation diagram and its interaction with the already existing DHW system elements.

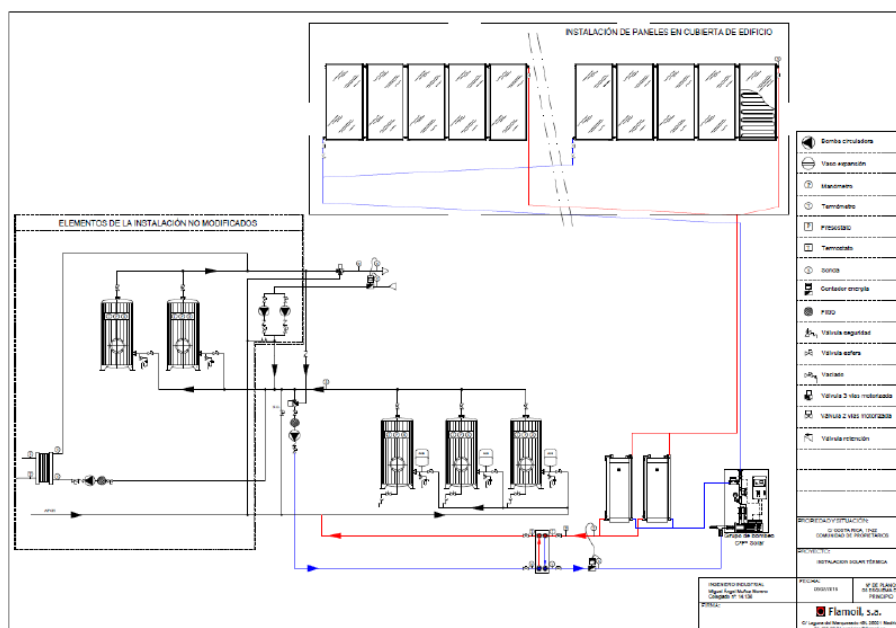


FIGURE 24: THERMOSOLAR SYSTEM DIAGRAM

4.1.1 CHALLENGE WITH THE DEVICES AND OTHER REASONS

Although the deployment of devices was thoroughly planned in the early stages of the project, a set of challenges and problems arose when the real deployment happened. One of the most important issues was related to the loss of data gathered by the devices deployed in the Madrid pilot site.

The problem was spotted by TEK when a report of available data was made in May 2019. More specifically, the encountered problem was related with malfunctioning of three gateways that were not sending data gathered by the deployed devices. The first two gateways that were not sending data were located in the meter rooms of two different building, thus the electric consumption of 7 houses and 4 common areas was lost for around 6 months. Likewise, the information gathered by the devices deployed in another house of the Madrid pilot site was lost due to the malfunctioning of the gateway. Once the problem was detected, Madrid pilot managers analyzed the problem and saw that, it was a matter of connectivity with the telephone grid due to their location in the basement of buildings. The problem could fortunately be solved by resetting the SIM cards.

The malfunctioning of these gateways couldn't have been spotted unless partners with access to the gathered data verified that the data was sent correctly. A task that cannot be manually performed on a daily basis, due to the big amount of data being collected. Therefore, in order to avoid such undesirable situations in the future, an alert service has been developed. Namely, this alert service has been implemented in Chronograf by means of rules. These rules are in charge of automatically verifying that data is being sent and generating alerts (via e-mail) when data is not being sent during a certain period of time.

Kapacitor is a mechanism that allows processing incoming data before inserting it in the database. It allows different ETL (Extract, Transform and Load) jobs, defining alerts and detecting abnormalities. These are defined in a scripting language called TICKscript.

The scripts can be defined as continuous(streams) or cyclic (batch) jobs. They follow a pipe-based approach, where each consecutive instruction is used as input for the next ones. The example in Figure 25 shows a script that stores the mean of the “demand” variables each 5 minutes.

```
1 batch
2 |query('SELECT mean(value) demand FROM "respondpre"."autogen".demand')
3   .period(5m)
4   .every(5m)
5   .groupBy('deviceId')
6 |influxDBOut()
7   .database('respondpre')
8   .retentionPolicy('autogen')
9   .measurement('RESPOND')
10
```

FIGURE 25: TICKSCRIPT EXAMPLE (MEAN)

Chronograf includes a specific Alert Rule Builder in order to ease the definition of alerts. Three alert types can be defined, each one with different conditions:

- **Threshold:** Generate an alert if a value crosses a defined limit or it's outside a defined range
- **Relative:** Generate an alert if a value changes relative to its previous ones, within a time window
- **Deadman:** Generate an alert if a defined time windows has passed without receiving data

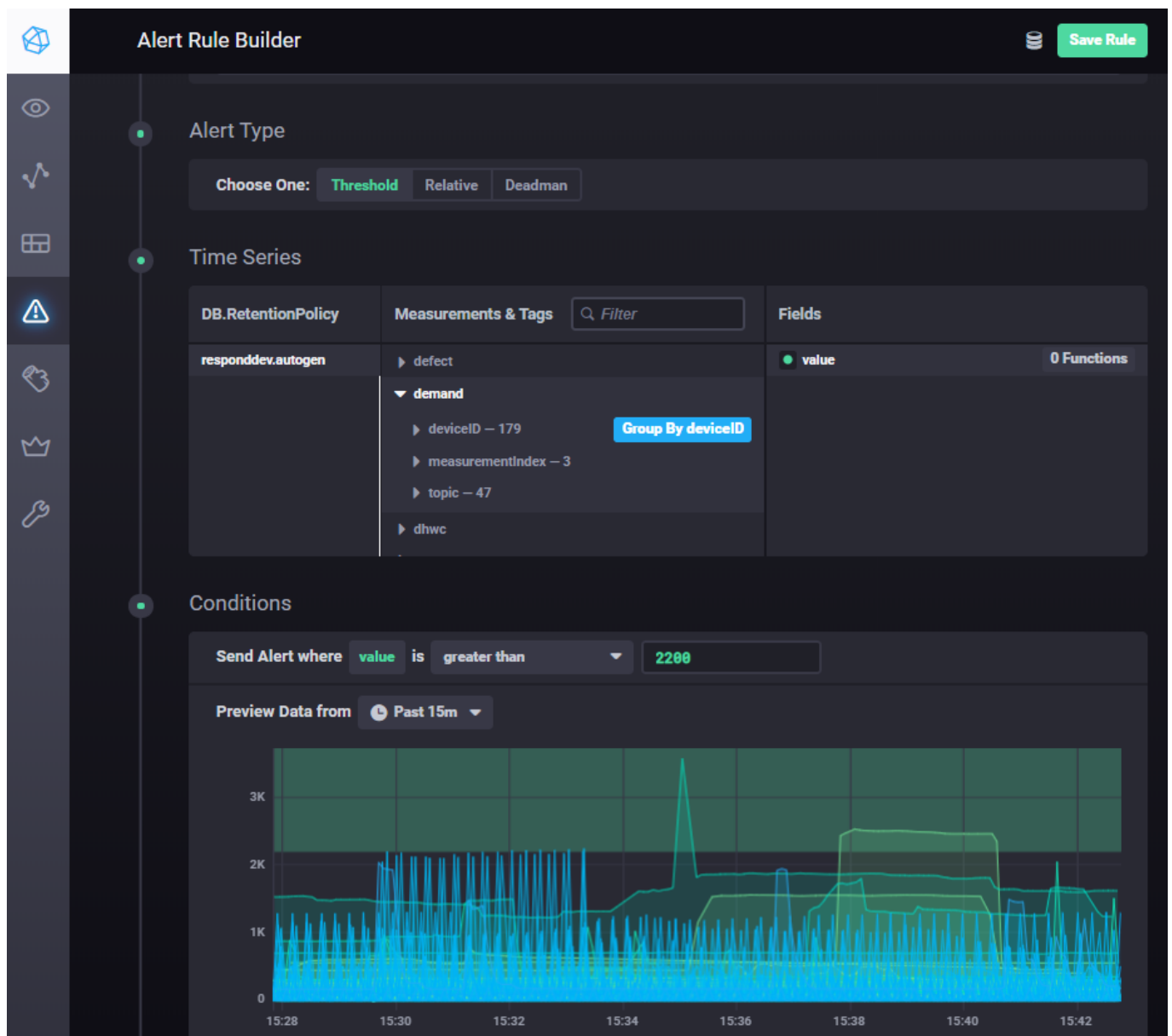


FIGURE 26: CHRONOGRAF ALERT RULE BUILDER (TYPE AND CONDITIONS)

Deadman is the alert used to detect that data are not received in the last 24 hours.

The alert handlers describe the action to take when the alert rises. In RESPOND two handlers have been used, log and email. Log just writes the alerts to a specific log, while email will send an email to a list of users using a configured SMTP server.

The message defines what part of the alert will be showed: the name, the criticality, the value of certain fields... In the case of the email handlers this message will be sent as the subject, as shown in Figure 28

Alerts can also be defined outside the Rule builder, by defining the specific TICKscript [Figure 29]. This allows more complex conditions and rules, but requires a more profound knowledge of TICKscript grammar and functionalities [<https://docs.influxdata.com/kapacitor/v1.5/tick/>]

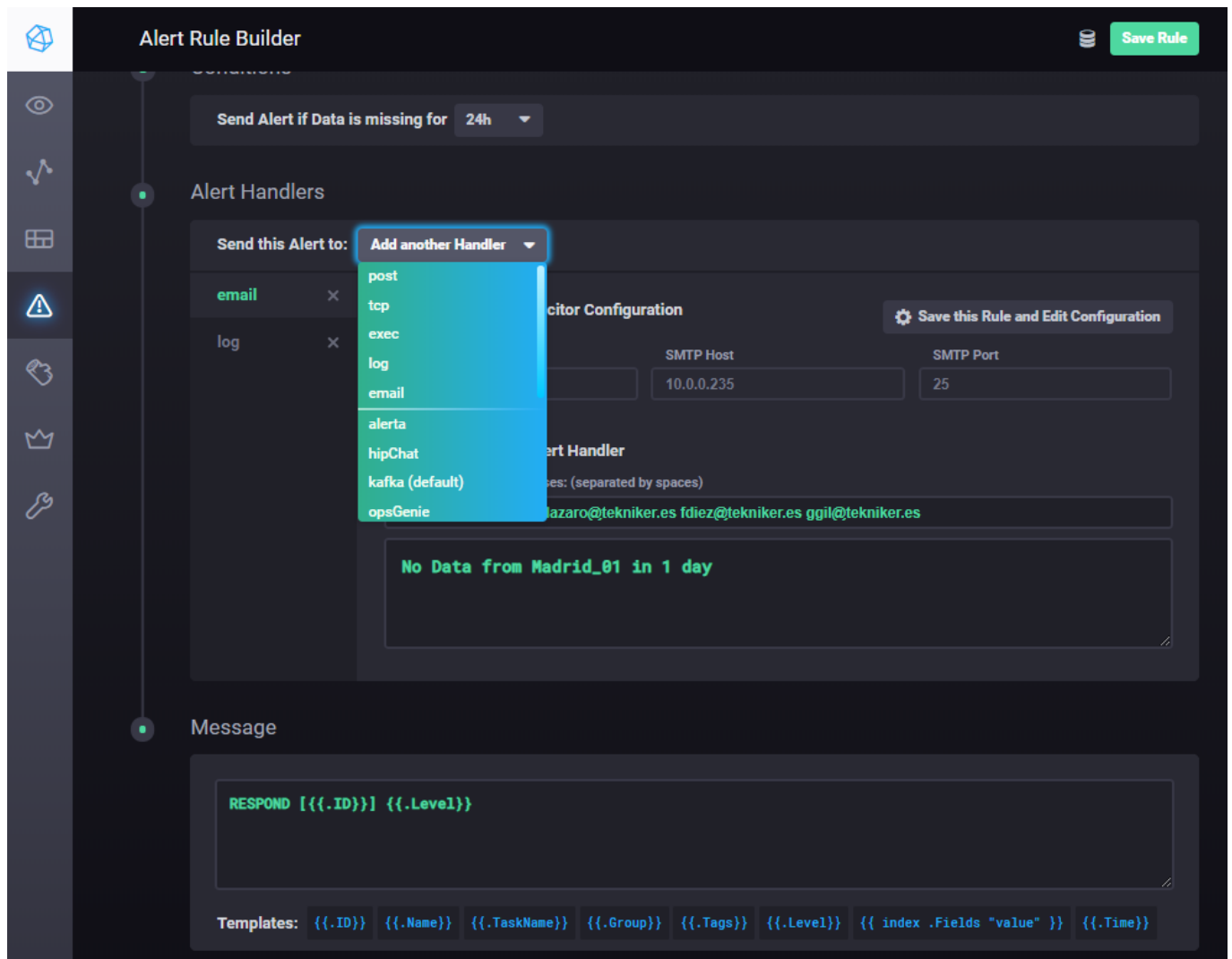


FIGURE 27: CHRONOGRAF ALERT RULE BUILDER (HANDLERS AND MESSAGE)

Álvaro García	RESPOND [NoDataIn1Hour(Tek_Gateway):deviceId=ENE-0C000C24] CRITICAL	ma. 24/09/20...	42 KB
Álvaro García	RESPOND [NoDataIn1Hour(Tek_Gateway):deviceId=ENE-0C000C24] CRITICAL	ma. 24/09/20...	42 KB
Álvaro García	RESPOND [NoDataIn1Hour(Tek_Gateway):deviceId=ENE-0C000C1B] OK	ma. 24/09/20...	42 KB

RESPOND [NoDataIn1Hour(Tek_Gateway):deviceId=ENE-0C000C24] CRITICAL



Álvaro García

Para Álvaro García; Ignacio Lázaro; Francisco Javier Díez; Gonzalo Gil

Responder Responder a todos Reenviar ...

ma. 24/09/2019 15:54

No Data from Tekniker Gateway in 1 HOUR

FIGURE 28: RECEIVED ALERT EMAIL

```

1
2 var data = stream
3   |from()
4     .database('respond')
5     .retentionPolicy('autogen')
6     .measurement('demand')
7     .groupBy([])
8     .where(lambda: ("topic" == 'ENE-ACD381946C6E4ED6/data'))
9
10 var trigger = data
11   |deadman(0.0, 24h)
12     .stateChangesOnly()
13     .message('RESPOND [{{.ID}}] {{.Level}}')
14     .id('NoDataIn1Day_Madrid01')
15     .idTag('alertID')
16     .levelTag('level')
17     .messageField('message')
18     .durationField('duration')
19     .details('No Data from Madrid_01 in 1 day')
20     .email()
21     .to('agarcia@tekniker.es')
22     .to('ilazaro@tekniker.es')
23     .to('fdiez@tekniker.es')
24     .to('ggil@tekniker.es')
25
26 trigger
27   |eval(lambda: "emitted")
28     .as('value')
29     .keep('value', 'message', 'duration')
30   |eval(lambda: float("value"))
31     .as('value')
32     .keep()
33   |influxDBOut()
34     .create()
35     .database('chronograf')
36     .retentionPolicy('autogen')
37     .measurement('alerts')
38     .tag('alertName', 'NoDataIn1Day_Madrid01')
39     .tag('triggerType', 'deadman')
40
41 trigger
42   |httpOut('output')
43

```

FIGURE 29: ALERT AS TICKSCRIPT

Alerts are received by e-mail and the pilot owner could supervise the state of the corresponding gateway and find a solution to prevent more data loss.

4.1.2 THERMOSOLAR'S OGEMA ENERGY GATEWAY

The Madrid pilot site included a solar thermal installation that used the solar panels to pre-heat boiler water, reducing the energy costs. This installation is monitored by different sensors and counters connected to a

KNX network through a Siemens RMS705B digital control unit. It also included an Internet router to enable remote monitoring and support to the KNX service providers.



FIGURE 30: SIEMENS RMS705B AND ZENNIO KNX-USB BRIDGE

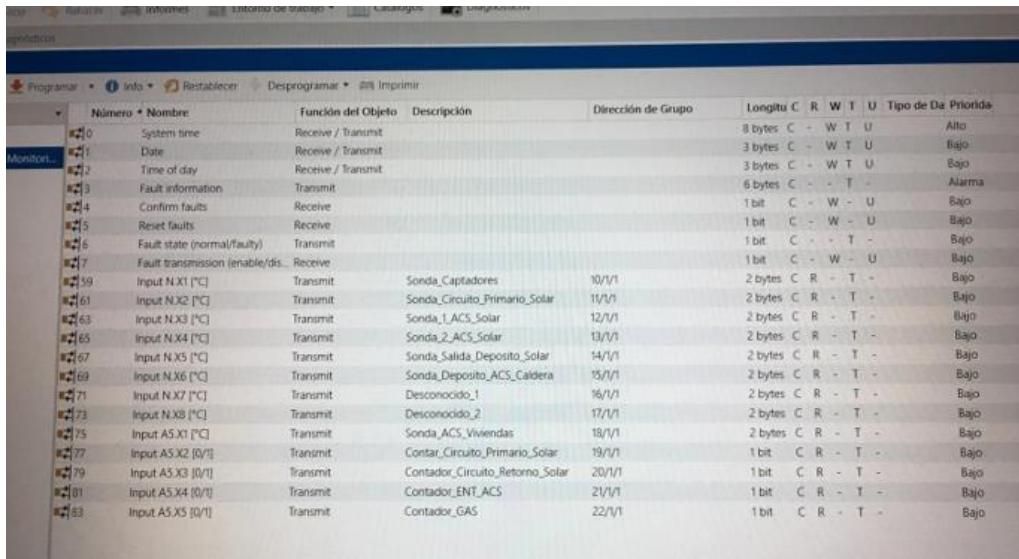
Two new devices were added to the existing infrastructure: A Zennio KNX-USB bridge and a PC with the developed Energy Gateway installed. The PC used a Linux Ubuntu 18.04 operating system. It was connected to the KNX network through the KNX-USB bridge and to Internet through the router via Ethernet.



FIGURE 31: GATEWAY PC CONNECTED TO THE KNX SWITCHBOX

During the deployment of the gateway there were some problems due to the difference in the KNX configuration used by the Siemens Synco devices and the KNX driver from the gateway. Synco devices can be configured in two modes: LTE (easy mode) and S-Mode (system mode). LTE mode is used to quickly connect and use Synco devices to a KNX network and is the one that they use by default, while S-Mode is a more advanced configuration mode used when connecting to other KNX entities outside the Synco network, such as the gateway's KNX driver.

KNX devices were configured in Madrid with LTE, and during the deployment none of the presents had the expertise to change it to S-Mode. Other functionalities were tested such as the configuration of communication (ensuring the messages were correctly published in the main broker) and other resilience tests, such as disconnecting the Wi-Fi communications, sudden shutdowns due to power loss...



Número	Nombre	Función del Objeto	Descripción	Dirección de Grupo	Longitud	C	R	W	T	U	Tipo de Da	Prioridad
0	System time	Receive / Transmit			8 bytes	C	-	W	T	U		Alto
1	Date	Receive / Transmit			3 bytes	C	-	W	T	U		Bajo
2	Time of day	Receive / Transmit			3 bytes	C	-	W	T	U		Bajo
3	Fault information	Transmit			6 bytes	C	-	-	T	-		Alarma
4	Confirm faults	Receive			1 bit	C	-	W	-	U		Bajo
5	Reset faults	Receive			1 bit	C	-	W	-	U		Bajo
6	Fault state (normal/faulty)	Transmit			1 bit	C	-	-	T	-		Bajo
7	Fault transmission (enable/dis)	Receive			1 bit	C	-	W	-	U		Bajo
59	Input N.X1 [°C]	Transmit	Sonda_Capitadores	10/1/1	2 bytes	C	R	-	T	-		Bajo
61	Input N.X2 [°C]	Transmit	Sonda_Circuito_Primary_Solar	11/1/1	2 bytes	C	R	-	T	-		Bajo
63	Input N.X3 [°C]	Transmit	Sonda_1_ACS_Solar	12/1/1	2 bytes	C	R	-	T	-		Bajo
65	Input N.X4 [°C]	Transmit	Sonda_2_ACS_Solar	13/1/1	2 bytes	C	R	-	T	-		Bajo
67	Input N.X5 [°C]	Transmit	Sonda_Salida_Deposito_Solar	14/1/1	2 bytes	C	R	-	T	-		Bajo
69	Input N.X6 [°C]	Transmit	Sonda_Deposito_ACS_Caldera	15/1/1	2 bytes	C	R	-	T	-		Bajo
71	Input N.X7 [°C]	Transmit	Desconocido_1	16/1/1	2 bytes	C	R	-	T	-		Bajo
73	Input N.X8 [°C]	Transmit	Desconocido_2	17/1/1	2 bytes	C	R	-	T	-		Bajo
75	Input A5.X1 [°C]	Transmit	Sonda_ACS_Viviendas	18/1/1	2 bytes	C	R	-	T	-		Bajo
77	Input A5.X2 [0/1]	Transmit	Contar_Circuito_Primary_Solar	19/1/1	1 bit	C	R	-	T	-		Bajo
79	Input A5.X3 [0/1]	Transmit	Contador_Circuito_Returno_Solar	20/1/1	1 bit	C	R	-	T	-		Bajo
81	Input A5.X4 [0/1]	Transmit	Contador_ENT_ACS	21/1/1	1 bit	C	R	-	T	-		Bajo
83	Input A5.X5 [0/1]	Transmit	Contador_GAS	22/1/1	1 bit	C	R	-	T	-		Bajo

FIGURE 32: RMS705B CONFIGURATION IN ETS

Later on, the RMS705B was properly configured with ETS (Engineering Tool Software) to be used in S-Mode. This enabled the openMUC gateway to detect data changes, format them in Canonical Data Model (CDM) and send it to PUPIN's broker. Temperature probes send data in a decimal format, depicting the temperature in °C. Counters send binary data, sending a message whenever they were activated. The next table shows the full list of the monitored elements:

TABLE 3: SOLAR SENSORS CONFIGURATION

DEVICE	RMS705B	KNX -DTP	CDM Topic	CDM DeviceID	CDM MeasurementID
Probe - Panels	N.X1	10/1/1 (9.001)	TEK-001	TEK-000001-001	inlet_flow_temperature
Probe – Primary Solar Circuit	N.X2	12/1/1 (9.001)	TEK-001	TEK-000001-002	inlet_flow_temperature
Probe - Solar ACS 1	N.X3	12/1/1 (9.001)	TEK-001	TEK-000001-003	inlet_flow_temperature
Probe - Solar ACS 2	N.X4	13/1/1 (9.001)	TEK-001	TEK-000001-004	inlet_flow_temperature
Probe - Solar Tank output	N.X5	14/1/1 (9.001)	TEK-001	TEK-000001-005	inlet_flow_temperature
Probe - Boiler ACS Tank	N.X6	15/1/1 (9.001)	TEK-001	TEK-000001-006	inlet_flow_temperature
Probe – Household ACS	A.X1	18/1/1 (9.001)	TEK-001	TEK-000001-007	inlet_flow_temperature
Counter – Primary Solar Circuit	A.X2	19/1/1 (1.001)	TEK-001	TEK-000001-008	heat_energy
Counter – Return Solar Circuit	A.X3	20/1/1 (1.001)	TEK-001	TEK-000001-009	heat_energy
Counter - ENT ACS	A.X4	21/1/1 (1.001)	TEK-001	TEK-000001-010	dhwc
Counter - GAS	A.X5	22/1/1 (1.001)	TEK-001	TEK-000001-011	gas_consumption

The Energy Gateway included a MQTT client, which was configured to publish to the PUPIN broker, with predefined user and password as security measure. Additionally, TeamViewer was installed in the PC in order to enable remote support. Both Energy Gateway and TeamViewer were configured to be launched on start-up, making it able to recover in case of an energy shutdown.

Additional alert was configured in TICK STACK to detect the gateway disconnection and take the corresponding actions as explained in the previous section.

4.1.3 WATER MONITORING SERVICE (FEN)

In addition to the initial devices/software originally planned to be deployed in Madrid pilot site, it was necessary to come out with a workaround to be able to monitor water consumptions during the trials. Originally it was intended to use Energomonitor water meters for that purpose but, due to the characteristic of the legacy water meters property of the water supplier company, this approach was dismissed due to technical problems. The adopted solution has been developed internally in FEN. Namely, the developed solution is a software system that reads water consumptions from water supplier company's web page, and processes, cleans and transforms them before forwarding them to RESPOND's MQTT server.

Daily water consumptions, both for hot and cold water, are published in the web on a weekly basis. Therefore, it is necessary to use an hourly profiled customized based on previous experience to obtain information on an hourly basis.

This service is running in a cloud server own by FEN and it has been developed using Microsoft .NET 4.7.1 framework in Visual Studio 2017 IDE by FEN's IT developers. For this purpose, it was necessary to use Selenium WebDriver, Newtonsoft Json and M2MQTT tools. Further information on the design and workflow of this service can be found in deliverable D5.1 [3] section 4.4. In Figure 33, we present an example of the message following CDM in JSON format sent to the RESPOND MQTT broker.

```
[{"timestamp":1553040000000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-244481269","measurementID":"dcwc"}, {"timestamp":1553043600000,"value":0.0,"measuor","value":5.8,"measurementIndex":1,"deviceId":"FEN-244481269","measurementID":"dcwc"}, {"timestamp":1553076000000,"value":4.2,"measurementIndex":1,"deviceIdI":ntIndex":1,"deviceId":"FEN-244481269","measurementID":"dcwc"}, {"timestamp":1553108400000,"value":5.7,"measurementIndex":1,"deviceId":"FEN-244481269","measureN-244480743","measurementID":"dcwc"}, {"timestamp":1553054400000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-244480743","measurementID":"dcwc"}, {"timestampI":1553119200000,"value":0.9,"measurementIndex":1,"deviceId":"FEN-244480743","measurementID":"dcwc"}, {"timestamp":1553094000000,"value":1.4,"measurementIndex":1,"deviceId":"FEN-244480743","measurementID":"dcwc"}, {"timestamp":1553128000000,"value":0.4,"measurementIndex":1,"measurementIndex":1,"deviceId":"FEN-241565276","measurementID":"dcwc"}, {"timestamp":1553068800000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-241565276","measurementID":"dcwc"}, {"timestamp":1553101200000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-241565276","measurementID":"dcwc"}, {"timestamp":1553072000000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-241565108","measurementID":"dcwc"}, {"timestamp":1553082000000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-241565108","measurementID":"dcwc"}, {"timestamp":1553093200000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-241565108","measurementID":"dcwc"}, {"timestamp":1553061600000,"value":0.5,"measurementIndex":1,"deviceId":"FEN-241565290","measurementID":"dcwc"}, {"timestamp":1553094000000,"value":4.3,"measurementIndex":1,"deviceId":"FEN-241565290","measurementID":"dcwc"}, {"timestamp":1553040000000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-244480750","measurementID":"dcwc"}, {"timestamp":1553043600000,"va":mp":1553072400000,"value":26.5,"measurementIndex":1,"deviceId":"FEN-244480750","measurementID":"dcwc"}, {"timestamp":1553076000000,"value":18.9,"measureme":value":22.7,"measurementIndex":1,"deviceId":"FEN-244480750","measurementID":"dcwc"}, {"timestamp":1553108400000,"value":25.8,"measurementIndex":1,"deviceIdI":ntIndex":1,"deviceId":"FEN-244481580","measurementID":"dcwc"}, {"timestamp":1553054400000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-244481580","measureE-FEN-244481580","measurementID":"dcwc"}, {"timestamp":1553086800000,"value":3.5,"measurementIndex":1,"deviceId":"FEN-244481580","measurementID":"dcwc"}, {"timestamp":1553119200000,"value":3.5,"measurementIndex":1,"deviceId":"FEN-244481580","measurementID":"dcwc"}, {"timestamp":1553128000000,"va":mp":1553065200000,"value":0.5,"measurementIndex":1,"deviceId":"FEN-244481450","measurementID":"dcwc"}, {"timestamp":1553068800000,"value":1.4,"measurementIndex":1.4,"measurementIndex":1,"deviceId":"FEN-244481450","measurementID":"dcwc"}, {"timestamp":1553101200000,"value":2.4,"measurementIndex":1,"deviceId":"FEN-2444",deviceId":"FEN-244480736","measurementID":"dcwc"}, {"timestamp":1553072400000,"value":0.0,"measurementIndex":1,"deviceId":"FEN-244480736","measurementID":"dcwc"}, {"timestamp":1553079600000,"value":4.3,"measurementIndex":1,"deviceId":"FEN-244480736","measurementID":"dcwc"}, {"timestamp":15531200000,"value":15.5,"measurementIndex":1,"deviceId":"FEN-244480736","measurementID":"dcwc"}, {"timestamp":1553115600000,"value":7.7,
```

FIGURE 33: WATER MEASUREMENTS MESSAGE EXAMPLE

4.2 DEPLOYMENT AT AARHUS PILOT SITE

The families that live in Aarhus pilot site were informed about the installation by mail. However, not all the families were responding the email inquiries. In particular, in relation to the installation of the Kampstrup Calorimeter (Heatmeter), we have to consider that the installation had to be done by an external supplier (Plumbing firm). There was also a challenge in relation to the possibility to close the hot water supply. In practice it affected other tenants and not only the pilot families. It was necessary to have the possibility to shut down the water main line for the whole block and not only by the RESPOND family.

The installation got slightly delayed and it was necessary to make changes and adjust certain aspects of the installation. These have made the work more complex than originally assumed (see also deliverable D2.4 [4]).

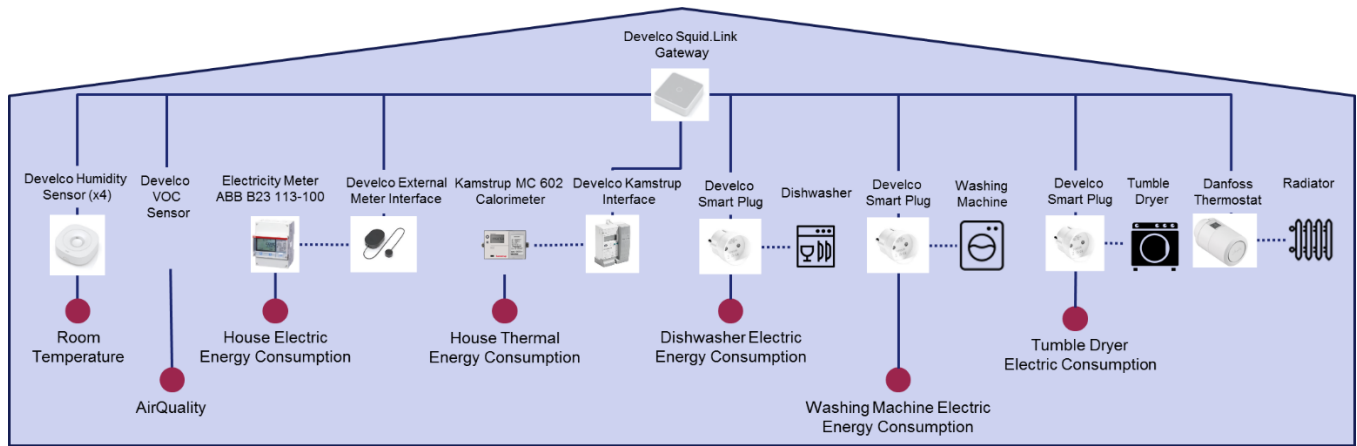


FIGURE 34: THE ORIGINAL PLAN FOR EQUIPMENT AVAILABLE IN ALL THE AARHUS HOUSES

Originally, all products should have been installed in one session and before the start of the heat season in September 2018. However, we couldn't install the Develco VOC Sensor and new Danfoss Thermostats.

The planned thermostat model expired and it was not clear which manufacturer and thermostat model should be used instead (see the Thermostats problem in section 4.2.2). As for the Air Quality (VOC) sensor, they were still in production at the time, and it was expected to be installed by autumn 2018. For sake of having enough data for the baseline period, it was decided to install all the other products, except for the Develco VOC sensors and the new Danfoss Thermostats.

During the early deployment period, all 20 dwellings in Aarhus have been equipped with the following devices:

TABLE 4: LIST OF DEVICES INSTALLED IN AARHUS PILOT SITE DWELLINGS

Device installed	Function	Location
Develco Squid.Link	Gateway	Living room
Kamstrup MC403 with WMBus	Monitoring of hot water consumed	Basement
Develco External Meter	Monitoring of Electricity consumed	Electrical meter in the Basement
Develco Smart Cable – Washing Machine	Monitoring of the electricity consumption of the washing machine	Laundry Room
Develco Smart Cable - Dryer	Monitoring of the electricity consumption of the Dryer	Basement
Develco SmartPlug	Wireless extender	Basement and 1. floor

Develco Smart Cable - Dishwasher	Monitoring of the electricity consumption of the Dishwasher	Basement
Develco Temp./Humidity Sensor	Monitoring of the humidity and temperature of the room in which it is installed	Bathroom
Develco Temp./Humidity Sensor	Monitoring of the humidity and temperature of the room in which it is installed	Bedroom
Develco Temp./Humidity Sensor	Monitoring of the humidity and temperature of the room in which it is installed	Kitchen
Develco Temp./Humidity Sensor	Monitoring of the humidity and temperature of the room in which it is installed	Living Room

Once the products were installed, a mobile operator subscription for data connectivity for each apartment was made.

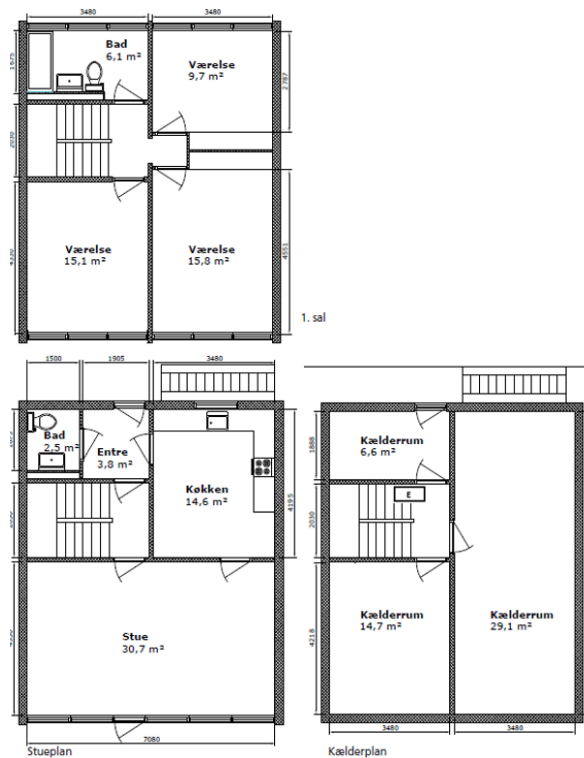


FIGURE 35: PHOTO OF THE APARTMENT BLUEPRINTS WITH ALL 3 FLOORS

Normally, two photos of the installation for each products were taken, as can be seen in Figure 36 - Figure 42.



FIGURE 36: PHOTOS OF INSTALLATION OF DEVELCO SMART CABLE FOR THE DISHWASHER ON THE GROUND FLOOR



FIGURE 37: PHOTOS OF INSTALLATION OF DEVELCO SMART CABLE FOR THE WASHING MACHINE IN THE BASEMENT



FIGURE 38: PHOTO OF INSTALLATION OF THE DEVELCO SQUID.LINK GATEWAY IN THE LIVINGROOM ON THE GROUND FLOOR



FIGURE 39: PHOTO OF INSTALLATION OF THE KAMSTRUP MC403 WITH WMBUS (HEADMETER) IN THE BASEMENT

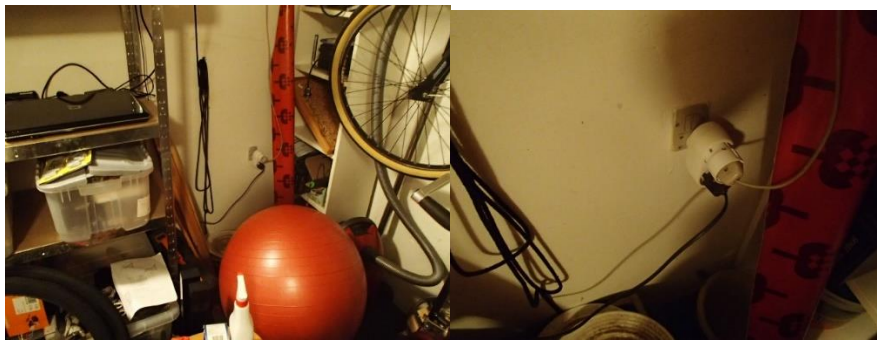


FIGURE 40: PHOTOS OF THE INSTALLATION OF THE DEVELCO SMART PLUG AS A WIRELESS EXTENDER IN THE BASEMENT



FIGURE 41: PHOTOS OF THE INSTALLATION OF DEVELCO TEMPERATUR/HUMIDITY SENSOR IN THE BATHROOM ON 1. FLOOR



FIGURE 42: PHOTOS OF THE INSTALLATION OF DEVELCO TEMPERATUR/HUMIDITY SENSOR (1,8 METER ABOVE THE FLOOR) IN THE BEDROOM ON 1. FLOOR

4.2.1 CHALLENGE WITH THE DEVICES AND OTHER REASONS

In the period following the installation of gateways, plugs and sensors, unexpected challenges emerged for various reasons.

One of the challenges was related to a new subscription of the gateway SIM card. The SIM cards for the gateways were purchased at ALBOA, as they could obtain them cheaper than Develco Products through an ALBOA agreement with the telecommunications company, but in connection with that new contact at ALBOA got new work on October 1, 2018, ALBOA's administration was not aware that these SIM cards was associated with gateways with families in the RESPOND project.

Alboa had switched to new subscription with Telia. The Telia telecommunications company informed us that the gateway needed to be restarted in order for Develco Products to receive the data again. Accordingly, an email was sent to the families with a guideline description on how to restart the gateways. It meant turning off the gateway for a minimum of 6 hours to ensure that the gateways battery backup was emptied. Consequently, approximately half of the gateways performed the reset correctly, but unfortunately this procedure was not carried out correctly for the remaining gateways. Some gateways fluctuated between being on/off and some families tried to turn on and off several times. RESPOND team visited several families and it turned out that there were no technical problems with the equipment for that moment, but e.g. one had turned off the Smart plug and not the gateway; one had not waited for 6 hours; one was registered under a wrong serial number and had been online all the time. Data was transferred directly to RESPOND and not being stored locally at Develco Products.

4.2.2 THE THERMOSTAT CHALLENGE

The thermostat model which was initially planned to be installed was not available anymore. Consequently, several factors had to be considered in order to find a new solution. One of the requirements was that new thermostat model should be compatible with the Develco Products gateway. Mechanically, it was necessary to find out if there was a need for the replacement of the radiator valves, since the current valves are legacy without pre-set (flow regulation). Another possibility would be to use an adapter since otherwise these radiator valves would not be suitable for the type of thermostat planned to be used in the project at the moment. If it is necessary to make a replacement of radiator valves for installation of thermostats, it requires

further visits/installation at the homes as well as the coordination between the parties involved and budget for replacement of radiator valves, which must be performed by an authorized plumbing installer.

It has been decided to perform several tests with two models of solutions: The Spirit ZigBee Thermostats (new product from Autumn 2018) and Danfoss ECO2 Bluetooth.



FIGURE 43: PHOTOS FROM THE INSTALLATION AND TEST OF THE SPIRIT THERMOSTAT

Develco Products made a test for over one month at their offices, and the thermostat worked very well. It was decided to make one more test with the equipment installed at a RESPOND pilot household. All 11 thermostats in the house were changed, and the old canister was also replaced. Nevertheless, the old valves were not replaced, because it has to be done by a plumbing installer and it was outside of the RESPOND budget. The test worked very well for around one week, then 9 thermostats stopped working. The tests showed that 9 out of 11 thermostats stopped working. The reason for the malfunctioning is still unknown. It could be caused by the old valves or because the weather changed very quickly from warm to cold in Denmark the night the 9 thermostats stopped working.

The other type of thermostat, the Danfoss ECO2 Bluetooth thermostat, is available from the summer 2017 and it is a well-known product in Denmark.



FIGURE 44: PHOTO OF THE DANFOSS ECO2 BLUETOOTH THERMOSTAT

Nevertheless, Danfoss thermostats have a well-known problem with signal range in houses built of concrete. The aim of the test was to find out whether the signal range to the gateway would be good enough to have good connection from each radiator. The tests were performed for one day. The Danfoss thermostats didn't have good connection, but they fit very well with the mechanical valves.

The problem was the Bluetooth signal range, since the gateway was placed on ground floor and the signal cannot penetrate the concrete wall up to the 1st floor or down into the basement. Consequently, the gateway had to be close to the radiator, which means that there must be at least 3 gateways (one on each floor). Besides, if the doors to the individual rooms are closed, more gateways are likely to be used.

Danfoss company stated that the thermostats work together with the old valves if the stuffing box is replaced. Next test was decided to be at apartment Aarhus _09. We made an installation of two more

gateways only for getting signal to the Thermostat; one gateway at 1st floor and one gateway in the basement. It has been tested in July and August 2019 and they are working well. The test has been performed during the Summer period.

It has to be emphasized that all the tests which have already been made, took a lot of time and resources and there are still some questions remaining: How many houses are going to have thermostats solution? What does other families said to get 3 gateways? In the project proposal budget, two visits to the families were planned: one for installation of equipment and another visit by the end of the project in order to take equipment down again. As consequence, AURA spent more hours on WP2 than budgeted. Nevertheless, the above technical challenges are likely to be solved, as Danfoss is in the test phase with a new ECO thermostat that uses ZigBee. This means that there will be enough having only 1 gateway in the home. It will therefore be relevant to test the new Danfoss ECO thermostats with ZigBee.

4.3 DEPLOYMENT AT ARAN ISLAND PILOT SITE

To recruit RESPOND participants to the project, the pilot site manager opted to produce and display posters in the locality inviting expressions of interest in the project. At first only 5 householders were recruited through this process, who immediately took part in a survey detailing occupancy etc. While the pilot site manager worked on finding more participants for the project, it was agreed among all the partners that work would begin on installing the Develco products in the first five homes.

In all, 11 participants were recruited to the project by the time the Develco equipment arrived. Unfortunately, some of the devices turned out not to be suited for the Irish electrical system or the intended devices they were needed to monitor. A more detailed explanation of these issues has been given in D2.4 [4].

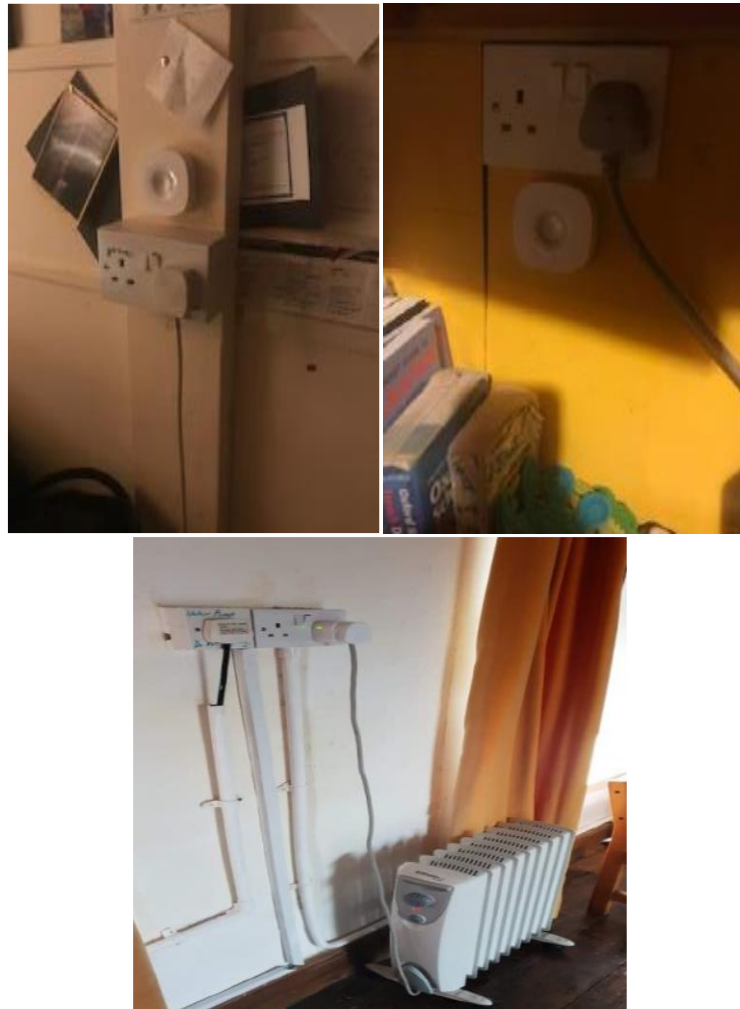


FIGURE 45: SOME OF THE DEVELCO DEVICES THAT COULD BE INSTALLED AT THE PILOT. ABOVE – TEMPERATURE AND HUMIDITY SENSORS. BELOW – SMART PLUG ON PORTABLE HEATER.

After liaising with Develco and the project coordinators, the pilot manager agreed with all partners that Energomonitor equipment would be installed instead of Develco equipment at the Aran pilot site. Before ordering a full shipment of Energomonitor equipment for the pilot site, the coordinators sent the pilot manager a sample shipment. These devices were deployed successfully and without any complications in one dwelling by the pilot site manager with assistance from NUIG.

Having proven that the Energomonitor equipment was ideal for the pilot, an order for the 11 recruited dwellings was placed with Energomonitor. This order was placed at the end of January, and the first half of the shipment arrived in March. This shipment contained Powersense devices, homebase receivers, and thermosense monitors and portasight displays. The deployment of these devices began, at first in the original 5 dwellings, and afterwards into the rest of the homes.



FIGURE 46: THE DEVICES THAT ARRIVED IN THE FIRST ENERGOMONITOR SHIPMENT TO THE ARAN PILOT

The deployment of these devices in the remaining 4 of the first 5 dwellings began in early April. Two of these were carried out without difficulty. Installation went very smoothly, and householders were very pleased to see progress in the project. One of the remaining dwellings were unable to facilitate access to the property for some time due to personal circumstances. The last, house number 2, was an extremely complicated case which has yet to be completed although a solution has now been found. The building dates back to the 1800s, and so the electrical system is very complex. There was also an issue with connectivity in the household as it is stone built, the devices were not able to transmit back to the homebase easily. Wifi was also installed in the building as part of this installation. This also had to be done for two other dwellings.

Installations were slowed down considerably during the summer months on the Aran Islands, this is because most residents are involved in tourism in one form or another and their schedules were very busy. By early August most had returned from family holidays etc and the installation process began again. The second shipment of energomonitor equipment had arrived during the summer months. This shipment contained airsense CO² meters, a specially designed version of the plugsense device. An appointed installer began to roll out all devices in all 11 participant homes. Unfortunately, one participant, who is quite elderly, seemed to have a change of heart when the installer visited and so he was given more time to consider his participation in the project.



FIGURE 47: ABOVE – THE SPECIALLY DESIGNED VERSION OF THE PLUGSENSE DEVICE. BELOW -THE AIRSENSE CO2 METER

The installer is currently completing the total installation in all households as the busy season on the islands winds down, this is expected to be complete by the end of September. The pilot site manager continues to recruit more participants to the project. There are many more houses across all three Aran Islands with PV panels, Heatpumps or both who are interested in participating. The pilot site manager is currently reviewing these and intends to place an order with Energomonitor for enough equipment for 12 more dwellings, this would complete the 24 houses aimed for the Aran Pilot.

5. CONCLUSIONS

In this document, we have presented the outcomes of the work performed within Task 2.5. The goal was to perform key activities towards deployment of RESPOND cloud-based platform and its core services at the pilot sites. All the activities have been carried out during the second year of the project, in parallel with the integration activities of DR components in WP3, WP4 and WP5. All the DR components were deployed matching the specifications of the reference architecture and the deployment plan which were created in tasks preceding Task 2.5 within WP2.

Firstly, we briefly revisited the RESPOND platform architecture and present the OpenWhisk microservice architecture which is used to deploy each individual analytic service. Then, we present the RESPOND platform control loop which connects different services from the measurement to the control actions. Next, a brief description of different analytic services is provided along with the mobile and web applications which are used for interactions with end users. Finally, we provide a description of the tasks performed in relation to deployment of RESPOND platform components at pilot sites. Furthermore, the challenges and problems faced during the deployment of such components is described.

REFERENCES

RESPOND DOCUMENTS

- [1]D2.1 RESPOND system reference architecture
- [2]D5.3 RESPOND Middleware deployment
- [3]D5.1 Energy gateway for home automation interoperability
- [4]D2.4 Early deployment activities report

EXTERNAL DOCUMENTS

- [5] <https://openwhisk.apache.org/>. (Accessed on 01/09/2019).
- [6] <https://developer.ibm.com/> (Accessed on 01/09/2019).