ebalanceplus

Prototypes implemented

Deliverable D5.5

Date : 31/03/2022
Author(s) : LUIS ALONSO MUÑOZ, JOSE ANTONIO RUIZ, JAIME CHEN, PIOTROWSKI KRZYSZTOF

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grand agreement N°864283

Technical References

Project Acronym	ebalance-plus
Project Title	Energy balancing and resilience solutions to unlock the flexibility and increase market options for distribution grid
Project Coordinator	CEMOSA
Project Duration	42 months

Deliverable No.	D5.5
Dissemination level [1]	PU
Work Package	WP5 Communication Platform and system integration
Task	T5.5 Implementation of platform prototypes
Lead beneficiary	SOF
Contributing beneficiaries	IHP, SOF, CEM, AMP, UMA, REE, TPS, EMT, MGC, UNC, YNC, DTU, ENF
Due date of deliverable	30 September 2022
Actual submission date	21 December 2022
Reviewed	31 March 2023

[1] PU = Public
PP = Restricted to other programme participants (including the Commission Services)
RE = Restricted to a group specified by the consortium (including the Commission Services)
CO = Confidential, only for members of the consortium (including the Commission Services)

Document history

V	Date	Beneficiary	Author
0	20/09/2022	SOF, IHP	Jaime Chen, Luis Alonso, Jose Antonio Ruiz, Piotrowski Krzysztof
1	15/03/2022	SOF	Javier Barbarán

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Summary

Summary of Deliverable

The document describes the implementation of the prototypes that will be used in the demosites planned to test the ebalance plus platform. The document focuses on the hardware and software that is used in each of the prototype management units (MUs) deployed for each of the demosites and on the devices that have been integrated in the ebalance plus platform. Each management unit is an autonomous device (either physical or virtualized) that runs the ebalance plus platform composed of the data exchange middleware, several adapters (used to integrate external devices within the ebalance plus system) and a set of algorithms. The document explains each of the parts previously mentioned focusing on the technical details that make the ebalance plus system capable of exchanging and collecting information from different parts of the system, with a strong emphasis on the demosites.

Disclaimer

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Table of Contents

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Table of tables

Table of figures

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

List of Abbreviations

Abbreviations	Definitions
CMU	Customer Management Unit
DER	Distributed Energy Resources
DERMU	Distributed Energy Resources Management Unit
DMU	Device Management Unit
LVGMU	Low Voltage Grid Management Unit
MU	Management Unit
MVGMU	Medium Voltage Grid Management Unit
SG	Smart Grid
TLGMU	Top Level Grid Management Unit

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

1 Introduction

In task 5.5, the prototypes of the communication platform have been developed. The prototypes use different techniques such as in-lab deployment, virtualization, or containers and will be used in the final deployment in the demonstrators. Fog and cloud technologies have been chosen to be used in selected MUs to test these novel technologies to reduce latency and communication. In the context of this task and this deliverable the term prototype refers both to a development-phase device used to test the system before its deployment and to the final device that will be deployed at each demo site.

The ebalance-plus project proposes a hierarchical structure of MUs that coordinate among themselves to improve the performance of the smart grid. This means that the ebalance plus system does not follow the typical centralized approach. On the contrary, it relies on autonomous devices called MUs that operate together. In simple terms, each MU runs an instance of the data exchange middleware and several adapters (described in [1]) and possibly one or more algorithms (described in WP4).

The aim of this deliverable is to provide the technical details about the different MUs that will be finally deployed in the demo sites. Section 2 presents an overview of the ebalance-plus architecture emphasizing the relationship between the different MUs. Section 3 presents the technical features of the different models of MUs that are being developed as part of this project. Section 4 focuses on the ebalance plus platform and how it is installed in each prototype. Section 5 describes the devices integrated for each demo site. Section 6 closes the document presenting the conclusions.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

2 The ebalance-plus architecture

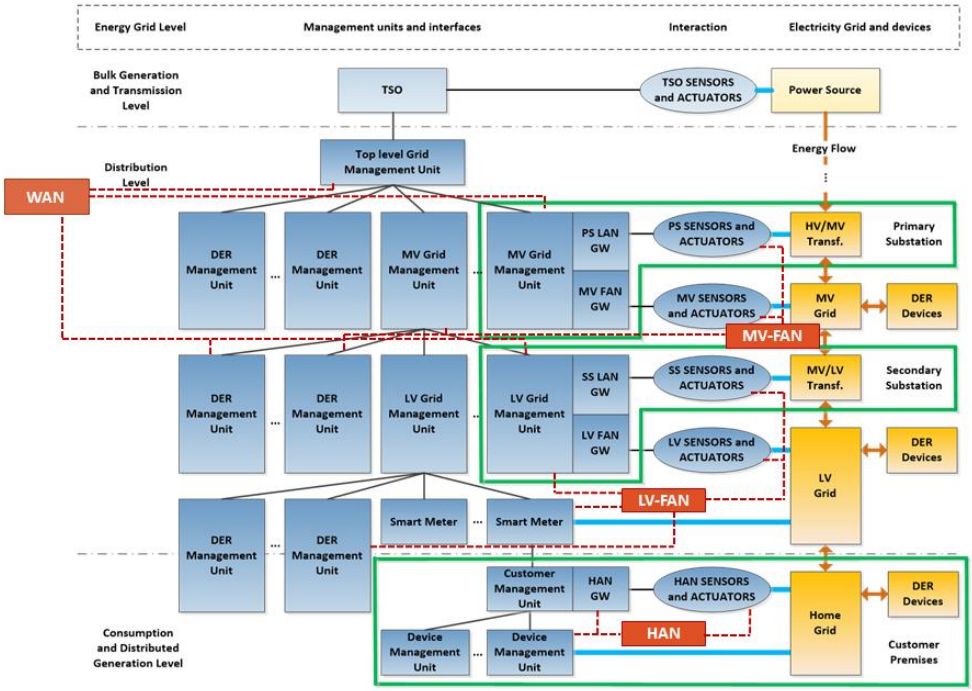

Figure 2.1 Example of the ebalance-plus architecture

An example ebalance-plus architecture is presented in Figure 2.1. The architecture is divided in several layers which in the example are located at a primary substation, secondary substation, and customer premises. Each layer contains at least one management unit that utilizes an instance of the middleware server – our distributed dataspace software component. These layers correspond to different levels of a tree structure that allows MUs to organize themselves hierarchically to exchange information. Ebalanceplus is therefore a distributed system which can store information at each level and MU independently. This distributed approach and the software necessary to support it, called the data exchange middleware, are some of the novel features of the proposed system which has already been described in [1]. The number of layers and MUs in a deployment can be tailored to the specific needs of the demo site.

Three main types of information exchange take place in the ebalance plus system. The first one is the communication between different MUs. This communication is performed by the data exchange middleware, a software layer that takes care of communication between MUs and information storage. The next interaction takes place between external devices (such as REST APIs, Modbus devices, etc.) and a single instance of a middleware located in a specific MU. This interaction is handled by a given adapter as described also in [1] and depicted as device management unit in the picture (DMU). Finally, there are algorithms that use the information stored in the middleware instances to achieve a given goal, such as using the available flexibility in the grid to optimize the performance. A general overview of how this communication works has been already presented in D5.3 [1]. This document focuses on the MU hardware and software necessary to facilitate such communication.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

3 Management units

A management unit is the basic building block in the ebalance plus architecture. It collects and stores information, it provides means to communicate with other MUs and it offers a platform where algorithms can run. Smart grids are complex system with different requirements so it is essential that the ebalance plus code can run in a high variety of devices. In general, Java is the main programming language used to code the ebalance plus system. This allows us to run the code in a high number of devices, including embedded devices, virtualized environments (such us the ebalance plus fog server deployed at University of Málaga which is one of the demo sites) or powerful servers. In fact, the decision about which MU is going to be used is given by each demo site requirement. Also, different approaches have been chosen to be tested in the demo sites to validate the ebalance plus architecture and evaluate the performance of different MUs.
This section describes the technical details of the MUs that have been developed as prototypes for the ebalance plus project. Section 3.1 describes the hardware of the different MUs that will be deployed. Section 3.2 presents the operating system and software that run on the MUs. Finally, Section 3.3 describes additional tools and scripts that have been developed to simplify the task of managing the MUs.

3.1 Hardware and virtualization

The proposed ebalance-plus architecture is designed, as seen in Section 2, in layers following a tree structure that allows MUs to exchange information in a hierarchically and organized way. Consequently, it is likely that different layers of the structure have different requirements, depending on how much information the device must manage. This section shows and details all the devices tested, the selection procedure, and why some tested devices were rejected.

3.1.1 Physical units

For each physical device chosen to implement the management units, a search, test and dismiss procedure was performed. As explained in previous sections, each position in the hierarchy of the ebalance-plus system will possibly have different power, flexibility and costs requirements that influences the final decision.
Table 3-1 lists a general view of the final devices that were selected to implement the architecture of the demonstrators.

Table 3-1. Devices general information.

MU TYPE	SYSTEM TYPE	MODEL	MEMORY	DISK SPACE	OS
CMU (Softcrits)	Mini PC	Quieter 2q	8GB	~100GB	Ubuntu 20.04.4 LTS
CMU (Reengen)	Embedded	Raspberry Pi 4 Model B	2GB	~20GB	Raspbian 10 x86
LVGMU (Emtech)	Embedded	Beaglebone Black	512MB	~64GB	Debian 10 x86
DERMU (Emtech)	Embedded	Beaglebone Black	512MB	~64GB	Debian 10 x86
MVGMU (Emtech)	Embedded	Beaglebone Black	512MB	~64GB	Debian 10 x86

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

TLGMU (Virtualized)	UMA Fog server (container)	-	-	-	Linux

3.1.1.1 Softcrits CMU

This type of management unit is going to be deployed in the last layer in the hierarchical structure of the demo site scenarios. It will launch several modules (adapters) in parallel and will save all data generated by the modules' readings. Therefore, two main characteristics are required: maximize RAM memory for devices with multiple java executions running at the same time, and a large disk space available to save the data. The lack of RAM memory is one of the main reasons why embedded devices were discarded. In this sense, Nvidia Jetson platforms includes sufficient RAM memory, and even a GPU is included to improve the execution of AI dependent applications, what may be useful for ebalance-plus algorithm needs. The main disadvantage of these devices is the availability in the market, making impossible to obtain after GPU stock restrictions. At last, Mini PC devices was selected mainly because of their flexibility, limited power consumption, great computing capabilities and great prices and availability in the market.
This specific model has a great disk space, good communications options with Ethernet and Wi-Fi, and does not require great heat dissipation needs. These units will be used as CMU for the University of Málaga demo site.

Figure 3.1 Quieter2Q device.

3.1.1.2 Reengen CMU

For this case use, Raspberry Pi was selected as hardware device. This type of embedded devices is a great choice for scenarios where low computing needs are required, a big amount of RAM memory is not needed, and very reduced price is the target. In addition, Raspberry offers a multitude of libraries, sensors and expansion cards that can be useful if readings from physical sensors must be done by the management unit. The partner Reengen provides a Raspberry Pi-based MU that they already sell as a commercial product. These units will be used as CMU for all the demo sites except for the one at University of Málaga (the one that requires more external devices to be integrated).

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Figure 3.2 Raspberry PI 4B board.

3.1.1.3 EMTECH LVGMU/DERMU/MVGMU

At last, LVGMU, DERMU and MVGMU use the same device to be the core of the computing hardware: Beaglebone Black. It is part of the single board computers family, that stands out among other options because of its reliability. Even when Raspberry or other devices are more powerful, have more memory, if the scenario does not require higher speeds or memory requisites, this type of board is known for its reliability in terms of failures or shutdowns. In the scenarios that will be used it's critical to minimize failures on the system, because they will be implement higher positions of the ebalance-plus demo sites hierarchy. The device is provided by EMTECH which has developed a Beaglebone-black commercial product shown in Figure 3.3. This device uses the expansion pins, in a custom developed motherboard that includes all the electronics to read from external sensors if needed. These units will be used as LVGMU, DERMU and MVGMU in all the demo sites. For more detail regarding each of the device refer to [3].

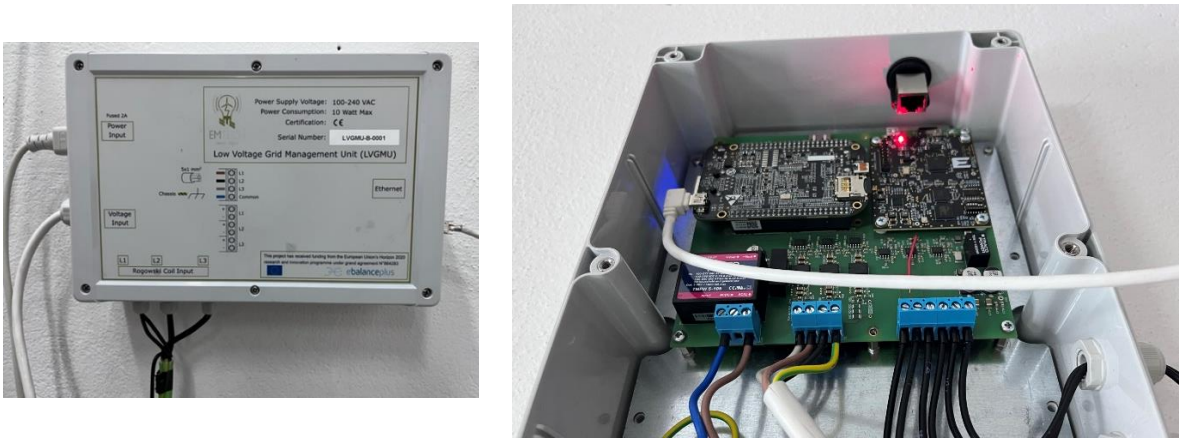

Figure 3.3 Beaglebone Black board.

3.1.2 Virtualized units

The Fog computing infrastructure at UMA provides a HW/SW environment for the development of applications with low latency requirements. Among the HW features, its 364 cores provided by Intel Xeon SP G6230R processors (26 cores, 2.1GHz), 448GB of graphics memory provided by NVIDIA V100 cards (5120 Cuda cores, 32GB RAM), 384GB of RAM and a total of 12.8TB of storage stand out. To sum up, the HW infrastructure comprise:

- **7 Fog nodes with 104 cores, 2 Nvidia NVIDIA V100 cards (5120 Cuda cores, 32GB RAM), and 384GB of RAM.**
- **2 Edge nodes with an Intel Xeon D2187NT processor (16 cores, 2GHz), an Nvidia T4 graphics card and its support for Wi-Fi and LTE/4G wireless communications.**
- **Total of 364 CPU cores, 448GB of GPU, 2688GB of RAM and 12.8TB of storage.**

The virtualized infrastructure provides a production environment for the development of critical applications on top of Docker containers. The orchestration tool to deploy and manage containers and their resources is Kubernetes (K8s), an open-source orchestration platform for containers-based applications with worldwide adoption. The developed applications can make use of GPU acceleration for processing-intensive techniques such as machine learning. This infrastructure has been chosen because the adoption of containers is a lightweight method to create virtual environments. They can be easily deployed, and don't comprise an entire operative system, only the relevant application and its dependencies are bundled into a

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

package, comprising a good solution for the portability of applications. Moreover, the capabilities of the infrastructure allow the deployment of the whole ebalance plus software stack and further applications which make intensive processing like flexibility algorithms.

3.2 Software

The setup of the device for the ebalance-plus functions is simple, but it must be done carefully, essentially because different versions of packages or operating systems may not work properly with the dependencies and executions of ebalance-plus.
This section summarizes the main tasks needed to achieve before proceeding with the installation described in Section 3.3.

3.2.1 Operating System Installation

Different MUs need different specific procedures, but in general terms the procedure to install the operating system is similar:

1. **Download the operating system image file.**
2. **Write the image into a SD card (Raspberry, Beaglebone, etc) or into an USB stick (Mini PC) using a OS flasher application, for example BalenaEtcher or Win32DiskImager**
3. **Insert the memory SD or stick in the device, power on, and follow instructions to finish the installation.**

Since the ebalanceplus system have been mainly implemented in Java and supports Python the main operating systems are supported. However, to reuse most of the procedures and associated code, Linux has been the main choice for all the MUs. Regarding the Linux version, there are no specific requirements.

3.2.2 Network

The device needs a stable internet connection as well as permission to connect through several ports. Therefore, the installer of the device must ensure with network administrator that the following requirements are met:

1. **Stable internet connection**
2. **Static Ip assignment**
3. **Open main ports (it may differ from config chosen in each module):**
    a. **Port 22: SSH connections**
    b. **Ports 20100 to 20105 (default): Middleware (the middleware uses 6 different ports for different purposes such as inter-MU connection, adminGUI server, polling server, etc.**

3.2.3 Date and time

Management units must ensure date and time info is precisely set since data read from devices and written in database requires an internal timestamp. To maintain a synchronization, NTP (network time protocol) is used.

3.2.4 Dependencies

Execution of the scripts and programs in the management units requires a sort of external dependencies that must be installed before the launch of the services.
Two types of dependencies must be listed in this section: those the dependencies which must be installed by the administrator manually; and the dependencies that the installer script will setup automatically.
The manual dependencies are:

1. **Python3**
2. **Tar files application. It is usually included in Linux distributions.**
3. **Apt application. It is usually included in Linux distributions.**
4. **Bash terminal. It is usually included in Linux distributions.**
5. **[Optional] OpenSSH or any other server-side SSH implementation.**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

6. **[Optional] Any remote desktop application if needed.**

The auto installed dependencies are:

1. **Python3-pip: Python3 packet manager.**
2. **ConfigParser: Python3 package needed to easily read config files.**
3. **Ppp: OS Package that enables VPN interfaces creation. OpenFortiVPN dependency.**
4. **OpenfortiVPN: Package that install OpenFort application, a VPN connection manager used for the JUNIA demo site.**
5. **OpenSSH(Server side): OS Package needed to transfer files. It should be installed manually if needed to remotely transfer the files. If it's not installed, it will be downloaded and setup.**
6. **Nano: Text editor used for installer to modify text in multi-instance config files.**

3.3 Tools and management

To ease the interaction between user and management units, various utilities were developed. In this section, how they work, and how to use them is described in detail.

3.3.1 SSH Access

SSH connections allow the user to control and command the machine and transferring files. It opens a terminal connection, so the user can execute commands remotely. OpenSSH Server must be installed in the target machine by the installer of the device if remote access is needed. Once the device has installed and launched SSH Server service, it must be checked that port 22 is allowed to be used in the network connected to the management unit.
Lastly, the user should connect the MU using a software that manages SSH connections. For example, Putty is a free-to-use software that handles the connection and lets the user save connection credentials.

3.3.2 Watchdog

Watchdog service is an external Python program developed to monitor and check the state of middleware, modules, and system processes. If any module or middleware suddenly fails, stops, or blocks, the watchdog will stop and reload it. In addition, it reset a specific timer included in operating system so if by any circumstances, the own Python program fails and the timer countdown reaches zero, the system will reboot automatically.

3.3.3 Installer script

The target of the project in terms of the installation procedure was to have a script that can handle every type of device from the hardware structure, automatically download and install dependencies, and helps the user to easily install middleware and adapter modules.
The developed script collects the needed information from the user interactively, as shown in Figure 3.4 and detect which type of device is being installed.
The steps to complete the procedure are:

1. **Insert MU installation ID (the name of the device). It should be unique for every device, since it will be used as to uniquely identify each unit.**
2. **Select from list which adapter modules need to be installed.**
   a. **If no module is selected, only middleware will be installed.**
   b. **If more than one instance of any module must be installed, it should be selected in the prompt.**
3. **[OPTIONAL] If multi-instance is selected, a second list with modules chosen in step 2 will appear. User should select which adapter modules will be configured as multi-instance, and how many instances are needed.**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

a. **Text editor will be launched for config files that must be edited for each instance of each module.**

4. **User is asked if middleware should be installed as a service launched at bootup.**
5. **User is asked if watchdog should be installed as a service launched at bootup**
6. **User is asked if adapter modules should be installed as a service launched at bootup**

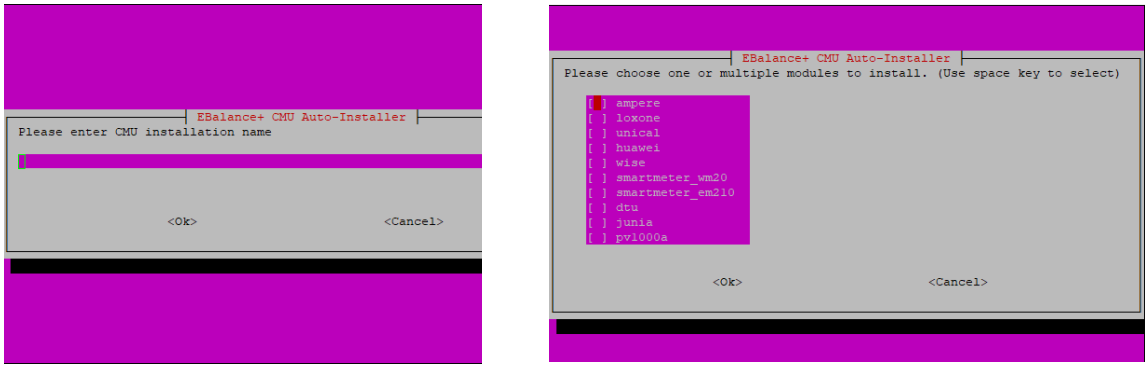

Figure 3.4 Installer screenshots

3.3.4 Stop and uninstaller scripts

In addition, in installer folders, two simple scripts are included. The first stops the middleware, adapter modules, and closes watchdog functionality, avoiding system resets when stopped. It is usually launched when reinstalling, upgrading, or fixing a module.
On the other side, the uninstaller script removes from the device every service, file and folders that create ebalanceplus implementation code.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

4 Ebalanceplus software

Section 4 describes the operating system and basic libraries that each MU uses. On top of this common hardware and software configuration the ebalance plus system needs to be installed in these units. There are two main entities that comprise the system: the data exchange middleware and the adapters. A detailed description of these two entities can be found in [1].

4.1 Data exchange middleware

The ebalance-plus system uses a data-centric middleware framework that allows the participants to communicate, exchange and store information. The framework stores data in tuple space structures which allows to implement a variety of logical structures that can contain all the information necessary to identify a value, its description, source, and time of creation. The tuple space is accessed by creating variables that can be written, read, or removed. A variable is the entity that holds historical data for related values. For example, a variable named weather can store historical information about temperature and humidity.
Variable management is the basic form of communication between a program and a local or remote instance of the middleware. In the context of ebalance plus a single instance of the middleware is installed in each MU. This means that each MU can locally store data. Programs accessing those units can either be installed within the MU itself or in a remote server as it is the case for programs running in the fog server, as explained in Section 3.1.2.
The middleware also provides a handy administration GUI, called adminGUI, that can be used to query the middleware information and for simple management tasks. Figure 4.1 shows a screenshot of the adminGUI.

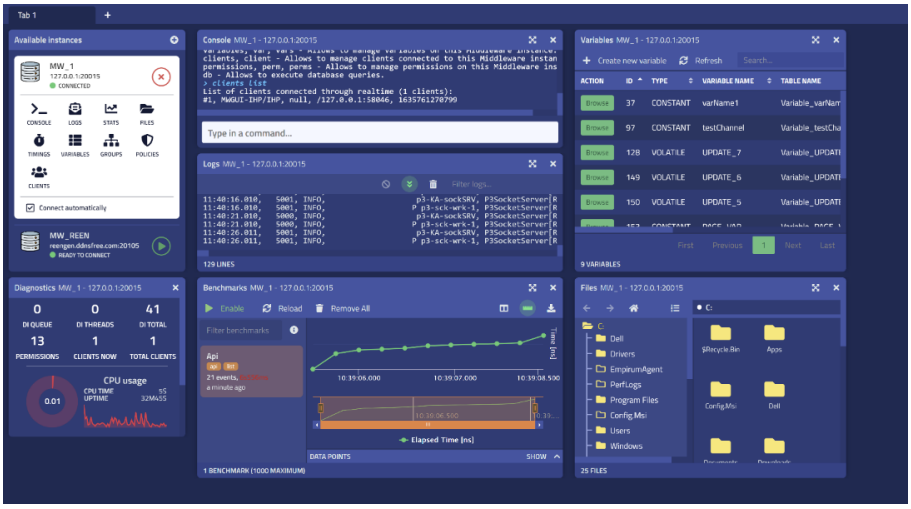

Figure 4.1 Admin GUI

The middleware is installed as a systemd service, so the process is launched, controlled, and supervised by the system (Linux and service manager), offering a wide variety of tools to manage, diagnose, and logging the middleware execution. In addition, the middleware is automatically launched by systemd after the network is available. This kind of installations makes the process more robust against possible energy shutdowns, or any circumstances that restart the operating system.
The middleware installation is done for every management unit using an installer script that inspect the device, download architecture-specific packages, and transfers and create all the files and folders that middleware execution is going to need. All the requirements for execution are detected and installed automatically by the installer script. The procedure in detail is explained in detail in 3.3 Tools and management.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

4.2 Adapter modules

There is a high number of devices whose data is interesting for the algorithms that run on the ebalance plus system. The adapter is an additional layer of abstraction used to convert information from all these external communication protocols to data that the middleware can understand and process. In ebalance plus, this layer of abstraction is provided by adapter modules. The adapter module communicates with a specific device and handles the integration with the data exchange middleware. The main goal of the adapter is to monitor an external device and get the relevant information to store it in the middleware. Also, when command requests are detected in the middleware, the adapter transforms the requests into commands that the device external can understand. As already presented in [1] a set of generic adapters was developed to ease the implementation of specific adapters for each demo site. Table 4-1 summarizes the generic adapters that were implemented.

Table 4-1 Generic adapters

Adapter	Communication protocol	Comments
Modbus adapter	Modbus TCP/IP	It maps a middleware variable to a set of Modbus registers (each column of a variable is a Modbus register)
JDBC adapter	Generic relational database using JDBC	It maps a middleware variable to a database table (each column of a variable is a column of a table)
REST API adapter	HTTP-based REST API	Each REST API endpoint is mapped to a middleware variable
Websocket adapter	Websocket	Handles the websocket connection and queries information but the mapping is left to the corresponding implementation (classes extending this generic class)

These generic adapters have been used to implement a set of adapters for each of the demo site planned in the project (See [2]).
Adapter modules are executed in the management units in a similar way as middleware. All the functions and classes that implements the module is built in a .jar file. As a result, transferring, installation and execution is simpler to setup and automatize. Jar file is executed and monitored using a Linux service managed by systemd, that offers some advantages explained in 4.1. Specifically, after the system boots up, middleware is first launched. After an initial delay, each module is sequentially launched, giving time to the module to start, acquire its first data, communicate, and initialize with middleware before the next module is launched. This kind of launch installation is done by the same installer script that installs middleware. To setup the installation, the user must select which modules should be installed from the available module list shown in the installer. If the MU needs to deploy multiple instances of a module, the option must be selected and modify each instance config file to avoid collisions in variables or instances names that should be unique. Each instance will be installed and launched as independent services, and their data saved as independent variables. The installation process is explained in detail in section 3.3

The following sections details the protocol and features of each of the adapters grouped by demo site focusing on the description of the information they collect and the setpoints that are available.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

4.2.1 UMA

4.2.1.1 LOXONE

The Loxone system connects all components in a smart building. All devices run to the central controller, the Miniserver, so they can communicate and become automated together. At Ada Byron building at University of Málaga the system has two miniservers, one in each building module connected to a router. And we have information and control over two devices:

- Air Handling Unit: Oversees managing the proper ventilation of the interior with outside air
- Weather Station: In this device we have information about brightness, wind speed, temperature, rain, wind warning and sunshine

The total number of components installed at the terrace at the Ada Byron building are:

- Miniserver (2 units)
- Wattmeters (2 units)
- Air Handling Unit, AHU (2 units)
- Heating pump (1 unit)
- Weather station (1 unit)

The Loxone system offers a WebSocket communication protocol that is connection-oriented and best suited for monitoring web applications. In fact, the manufacturer monitoring app uses this API to show the information. In our case, to avoid having a persistent connection to the system, the adapter periodically connects, it retrieves the information and it disconnect.
In the Ada Byron building the system has the following functions:

- Conditioning system energy
- Consumption monitoring
- Weather condition monitoring
- Control the speed fans of AHU
- Switching ON/OFF the Heating pumps
- Management of filtration and control of quality of the air
- Control of the air temperature that regulates the air conditioning system
- Relative humidity monitoring

The information is periodically queried, and historical data is sent to the middleware database. The current polling period, for the variables shown in Table 4-2 and Table 4-3 is once per hour.

Table 4-2 UMA: Loxone weather station variables

VARIABLE	DESCRIPTION	UNIT
read_timestamp	Unix timestamp	String date UTC
weatherType	numeric weather type enumeration value (see weatherTypeTexts)	Enum
windDirection	Wind direction	º
solarRadiation	Solar radiation	
relativeHumidity	Relative humidity	%
temperature	Temperature	ºC
perceivedTemperature	Perceived temperature	ºC
dewPoint	Drew point	º
precipitation	Precipitation	mm
windSpeed	Wind sped	km/h
barometicPressure	Barometric pressure	hPa

Table 4-3 UMA: Loxone building status variables

VARIABLE	DESCRIPTION	UNIT
uma_loxone_clima_mode	Climate mode 0 -> auto 1 -> cold 2-> heat	0.0

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

uma_loxone_heat_pump_switch	Heat pump status 1.0 -> on 0.0 -> off	1.0
uma_loxone_uta_switch_a	UTA status 1.0 -> on 0.0 -> off	1.0
uma_loxone_uta_switch_b	UTA status 1.0 -> on 0.0 -> off	1.0
uma_loxone_heating_threshold	Heating threshold (degrees celsius)	17 ºC
uma_loxone_max_water_temperature	Max. Summer temperature (degrees Celsius)	20 ºC
uma_loxone_min_water_temperature	Min. Winter temperature (degrees Celsius)	31 ºC
uma_loxone_speed_fans_mod_b	Fan speed module A (long) [0-10]	8
uma_loxone_speed_fans_mod_a	Fan speed module B (long) [0-10]	8

The setpoints shown in Table 4-4 allows information in the Loxone system to be changed by the algorithms.

Table 4-4 UMA: Loxone setpoints

Setpoints	Description	Values	Example
SETPOINT_B	Setpoint module B – objective temperature	Temperature value	18
SETPOINT_A	Setpoint module A – objective temperature	Temperature value	18
MAX_TEMP_SUMMER	Max. summer temperature	Temperature value	20
MIN_TEMP_WINTER	Min. winter temperature	Temperature value	31
FAN_SPEED_A	Fan speed module a	[0-10]	8
FAN_SPEED_B	Fan speed module b	[0-10]	8
UMBRAL_CALF	Heating threshold (degree Celsius)	Temperature value	17
FRIO_CALOR_MAN	Climate mode	0 -> auto 1 -> cold 2 -> heat	0
HEAT_PUMP	Turn on/off the heat pump	true/false	true

4.2.1.2 EM210

The EM210 by Carlos Gavazzi is an intelligent three-phase energy meter. The meter is connected to the general electrical Switchboard. A MODBUS RTU – TCP/IP gateway is employed to allow the access by MODBUS TCP/IP, and it is connected to the ebalance plus subnet. The meter is installed in the room S2 at the parking of Ada Byron building.
The system is used to:

- Reading and monitoring the Ada Byron building energy consumption
- Collection of historical data

The current polling period is once per five minutes. Historical data, depicted in Table 4-5, is stored in the middleware.

Table 4-5 UMA: EM210 variables

VARIABLE	DESCRIPTION
read_timestamp	Timestamp
v_l1_n	V L1-N Phase Voltage, between phase 1- neutral
v_l2_n	V L2-N Phase Voltage, between phase 2- neutral
v_l3_n	V L3-N Phase Voltage, between phase 3- neutral
v_l1_l2_n	V L1-L2 Voltage line to line (between phase 1- phase 2)
v_l2_l3_n	V L2-L3
v_l3_l1_n	V L3-L1
a_l1	A L1 current phase 1
a_l2	A L3
a_l3	A L3
w_l1	W L1 power in phase 1
w_l2	W L2
w_l3	W L3
va_l1	VA L1 Apparent power phase 1
va_l2	VA L2
va_l3	VA L3

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

var_l1	VAR L1 Reactive power phase 1
var_l2	VAR L2
var_l3	VAR L3
v_l_n_sum	V L-N SUM Phase Voltage
v_l_l_sum	V L-L SUM Voltage line to line
w_sum	W SUM Active power
va_sum	VA SUM Apparent power
var_sum	VAR SUM Reactive power
pf_l1	Power Factor L1
pf_l2	Power Factor L2
pf_l3	Power Factor L3
pf_sum	Power factor
phase_sequence	Phase sequence
hz	Frequency of electric power transmission
kwh_consumed_tot	kWh (+) TOT (Active energy consumed)
kvarh_sonsumed_tot	kvarh (+) TOT (Reactive energy consumed)
kwh_generated_tot	kWh (-) TOT (Active energy generated)

4.2.1.3 WISE

This system, developed in a previous project, is used for management of educational centres in aspects such as sustainability, safety, hygiene, control, intelligence and traceability of people and goods. All this, through various actions enabled by the platform such as optimising the closing/opening of windows, guaranteeing the quality of the air above or preventing the risk of contagion by facilitating, for example, contact tracing. This already existing system has been used at UMA demo site to obtain weather station and air quality variables.

The sensors are installed in the room A.0.11 at Ada Byron building at University of Málaga. The sensor devices collect values and send them to the WISE server. WISE system offers a REST API with an endpoint for querying the data gathered. The values stored are presented in Table 4-6. The current polling period is one time per five minutes.

Table 4-6 UMA: WISE variables

VARIABLE	DESCRIPTION	UNIT
read_timestamp	Timestamp	String date UTC
temperature	Temperature in the room	ºC
co2	Level of co2 in the room	%
noise	Noise in the room	ppm
barometric_ pressure	Barometric pressure in the room	hPa
humidity	Humidity in the room	%

4.2.1.4 AMPERE

Ampere is a smart storage system of the energy produced by solar PV systems. The system wil be used in UMA, UNC demo sites. In the UMA demo site, it is installed in the room sb5 in the parking at Ada Byron building at University of Málaga.
The components installed at the Ada Byron building are:

- 120kWH battery. In two racks of Ketter. The Ketter Power Charge 60 kWh is a battery for a high-power lithium-ion phosphate system with intelligent control technology.
- 100 kW inverter (Ingeteam 100TL). A three-phase battery inverter without transformer

The system has the following functions:

- Energy management
- Power control
- Performance optimization
- Data monitoring and representation
- Allows to make charging schedules for the batteries

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Also, the Smart Energy Management device integrated in the storage system provides information about the installation that is sent to the AMPERE Cloud. The information is provided in the cloud via a REST API that it has been used by our adapter to collect the parameters shown in Table 4-7.

Table 4-7 UMA: AMPERE Smart storage parameters

VARIABLE	DESCRIPTION	UNIT
timestamp	Unix timestamp	String date UTC
batterySoc	Battery charge level	%
batteryPower	Baterry power	kW (average)
batteryTemperature	Battery temperature	Cº (instant)
inverterActivePowerPh1	Active battery power on phase 1	kW (average)
inverterActivePowerPh2	Active battery power on phase 2	kW (average)
inverterActivePowerPh3	Active battery power on phase 3	kW (average)
meterActivePowerPh1	Active power of the meter on phase 1	kW (average)
meterActivePowerPh2	Active power of the meter on phase 2	kW (average)
meterActivePowerPh3	Active power of the meter on phase 3	kW (average)

The current polling period is once per fifteen minutes.
In addition, the adapter allows the smart battery schedule to be changed from the middleware.

4.2.1.5 OPENWEATHER

OpenWeather provides hyperlocal minute forecast, historical data, current state, and from short-term to annual and forecasted weather data. All data is available via a REST API. In our scenario, the information is checked once per minute and collected in the middleware. The information is used at UMA for the training of the different prediction models. After a first evaluation, it has been possible to observe the importance of the climate for the production of energy by solar panels or for forecasting future consumption values of the buildings. Moreover, these OpenWeather records will be studied to observe their correlation with the corresponding prediction models of other components of the system not yet developed. The weather values stored in the middleware are shown in Table 4-8.

Table 4-8 UMA: Openweather parameters

VARIABLE	DESCRIPTION	UNIT
datetime	Timestamp	String date UTC
weather	Group of weather parameters (Rain, Snow, Extreme etc.)	String
temp_min	Minimum temperature	Cº
temp_max	Maximum temperature	Cº
pressure	Atmospheric pressure (on the sea level if there is no sea_level or grnd_level data)	hPa
humidity	Humidity	%
wind	Wind speed	meter/sec

The current polling period is once per minute.

4.2.1.6 MCG EV CHARGER

MAGNUM CAP, one of the partners of the consortium, has developed EV chargers that where the vehicles can operate like huge storehouses of renewable energy. Also, it is possible that the energy produced by the solar panels is used to charge the vehicle anytime.
Five chargers will be installed in the parking of Ada Byron building. Each charger can load and unload two vehicles simultaneously and has bidirectional charging, using the energy provided by the PV panels that will be installed near to the chargers.
The information and the communication with the chargers is provided via the Modbus TCP/IP protocol. The EV charger provides parameters about the charger and the current transaction and allows commands to be sent by writing specific Modbus registers. The adapter that has

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

been implemented in ebalance plus periodically polls and reads read-only registers and allows commands to be sent from the middleware to write-only registers. The read/write parameters provided by the EV charger are shown in Table 4-9.

Table 4-9 UMA: Magnum cap EV charger parameters

VARIABLE	DESCRIPTION
read_timestamp	Timestamp
status_code	Gives information about charger's operation mode
present_dc_voltage	Output DC voltage
present_dc_power	Output DC power
dc_charged_energy	Session DC charged energy
dc_discharged_energy	Session DC discharged energy
time_to_end	Remaining transactions time
present_soc	Actual state of charge
start_button	Indicates if start button is active (1 inactive 0 active)
ac_energy_charged	Total charged energy, AC side
ac_discharded_energy	Total discharged energy, AC side
ac_power	Present AC power
ac_power_timestamp	Timestamp of the power reading
Max_charge_current	EVSE max charge current
Max_discharge_current	EVSE max discharge current
max_charge_voltage	EVSE max voltage
max_charge_power	EVSE max charge power
chademo_version	EVSE CHAdeMO version
v2h_version	EVSE V2H version
epo_button	Informs if emergency button is pressed (0 not pressed 1 pressed)
vehicle_present	Informs if there is a vehicle connected to the charger (0 no vehicle connected 1 vehicle connected)
ev_min_charge_current	Vehicle minimum charge current
ev_max_charge_current	Vehicle maximum charge current
ev_max_battert_voltage	Vehicle maximum battery voltage
ev_target_battery_voltage	Vehicle target battery voltage
ev_min_discharge_voltage	Vehicle minimum discharge voltage
ev_min_soc	Vehicle minimum SOC
ev_max_soc	Vehicle maximum SOC
vehicle_chademo_version	Vehicle Chademo version
vehicle_v2h_version	Vehicle V2H version

4.2.1.7 TPS INVERTER

TPS provides a MIMO Inverter that connects the AC grid, the PV canopies, battery racks and EV chargers to the common DC link. The MIMO inverter consists of a 125kVA GTI (inverter), a 70kW unidirectional DC/DC converter for PV canopies, a 70kW bidirectional DC/DC converter for Li-ion batteries and an AC and DC switchgear. The MIMO uses modern SiC (Silicon Carbide) semiconductor devices that have very low losses at high switching frequencies.
The system has the following functions:

- Bidirectional power conversion between the grid and the DC-Link.
- Reactive power compensation.
- Phase imbalance improvement.

The information and the communication with TPS Inverter system is provided via Modbus TCP/IP communication protocol. More specifically, in terms of communication, The TPS system is composed of three different Modbus slaves which need to be synchronized: GTI, DC/DC converter and the BMS. To synchronize these systems a controller, called the TPS controller, has been developed using Java. The TPS controller is a process that runs indefinitely and synchronizes readings between the different subsystems:

1. Periodically send heartbeat messages to each subsystem

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

2. Periodically send information about the BMS to the DC/DC converter
3. Handle connection/disconnection messages and errors

The TPS Inverters will be installed in the parking of Ada Byron building. In addition to the TPS controller, the usual adapter is deployed, whose function is to periodically read information (see Table 4-10, Table 4-11 and Table 4-12) from the TPS system and store it in the middleware.

Table 4-10 UMA: TPS inverter variables

VARIABLE	DESCRIPTION	UNIT
Time	Timestamp	String date UTC
GTI L1, L2, L3 Current	PV DC/DC - PV Pannels Current	Amps
GTI L1, L2, L3 Voltage	PV DC/DC - PV Panels Voltage	Volts
GTI L1, L2, L3 Real Power	PV DC/DC - Real Power	kW
GTI L1, L2, L3 Reactive Power	PV DC/DC - Reactive Power	kVAr
GTI L1, L2, Reactive Setpoint		-
GTI Max P Export Limit	GTI Max P Export Limit	kW
GTI Max Q Export Limit	GTI Max Q Export Limit	kVAr
GTI Max P Import Limit	PV DC/DC - Max P Import Limit Setpoint Report Back (curtailment)	kW
GTI Max Q Import Limit	PV DC/DC - Max Q Import Limit Setpoint Report Back (curtailment)	kVAr
Status register		-
Error register 1, 2		-
Analogue Spare 1, 2, 3, 4, 5		-

Table 4-11 UMA: TPS BESS variables

VARIABLE	DESCRIPTION	UNIT
Fully charged status	If 1 BESS is fully charged	-
Fully discharged status	If 1 BESS is fully discharged	-
BESS status	1: Idle, 2:Run, 4: error	-
BESS mode	Depending on Application	-
BESS error	Error Code	-
BESS warning	Warning code	-
BESS watchdog	Heartbeat counter, increments every sec, it resets at , 65.535	-
V sum of Cells (H) & (L)	Total cell Voltage in rack. Average if more than one	Volts
Total I DC (H) & (L)	Total BESS DC current	Amps
Cell T min (H) & (L)	Temperature of cell with lowest temperature	Celsius
Cell T max (H) & (L)	Temperature of cell with Highest temperature	Celsius
Charge available (H) & (L)	Maximum allowed charging power	kW
Discharge available (H) & (L)	Maximum allowed discharging power	kW
BESS SOC total (H) & (L)	State of Charge of BESS	%
Enable system	1: start 0: stop	-
Reset error	1: reset active error	-
EMS watchdog heartbeat	Heartbeat counter. If not changed for more than 60 seconds while BESS active system goes to idle.	-
Acknowledge fault	Clears reg: BESS error, BESS warning	-

Table 4-12 UMA: TPS DC/DC converter variables

VARIABLE	DESCRIPTION	UNIT

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

PV DC/DC converter		
PV DC/DC - PV Panels Current	DC Current on the PV panels input of DC/DC converter.	Amps
PV DC/DC - PV Panels Voltage	DC Voltage on the PV panels input of DC/DC converter	Volts
PV DC/DC - Real Power	Real Power taken from the PV Panels	kW
PV DC/DC - Max P Import Limit Setpoint Report Back (curtailment)	Report back for Maximum real power that can be imported (load) on DC/DC converter	kW
MPPT set point	MPPT set point	Volts
Status register	1 - Initialization 2 - Stand-by 3 - Start-up 4 - Run 5 - Fault 6 - Locked out	-
Error register 1	Details meaning of individual bits will be defined later.	-
Error register 2	Details meaning of individual bits will be defined later.	-
PV DC/DC converter heartbeat out	Heartbeat counter: increments every sec, resets at 65,535	-
PV DC/DC - Max P Import Limit Setpoint (curtailment)		kW
DC/DC converter heartbeat in	Heartbeat counter. If not changed for more than 60 seconds while DC/DC active system goes to shutdown	-
Enable PV DC/DC	Latch On = Enabled Latch Off = Disabled	-
PV DC/DC Power limit update (curtailment)	Control to update Setpoints Send 1 to trigger update	-
BESS DC/DC converter		
BESS DC/DC - Battery Current (converter measurement)	DC Current on the Batteries input of DC/DC converter. Positive current = Import from the battery Negative current = Export to the battery	Amps
BESS DC/DC - Battery Voltage (converter measurement)	DC Voltage on the Batteries input of DC/DC converter	Volts
BESS DC/DC - Real Power (converter measurement)	Positive current = import from the battery Negative current = Export to the battery	kW
BESS DC/DC - Power reference setpoint Report Back	Positive current = import from the battery Negative current = Export to the battery	kW
Status register	1 - Initialization 2 - Stand-by 3 - Start-up 4 - Run 5 - Fault 6 - Locked out	-
Error register 1	Details meaning of individual bits will be defined later.	-
Error register 2	Details meaning of individual bits will be defined later.	-
Reset error	1: reset active error for BESS	-
BESS DC/DC converter heartbeat out	Heartbeat counter: increments every sec, resets at 65,535	-
Fully charged status	If 1 BESS is fully charged	-
Fully discharged status	If 1 BESS is fully discharged	-
BESS status	1: Idle, 2:Run, 4: error	-
BESS mode	Depending on Application	-
BESS error	Error Code	-
BESS warning	Warning code	-
BESS watchdog	Heartbeat counter, increments every sec, it resets at , 65.535	-
V sum of Cells	Total cell Voltage in rack. Average if more than one	V
Total I DC	Total BESS DC current	A
Cell T min	Temperature of cell with lowest temperature	C
Cell T max	Temperature of cell with highest temperature	C
Charge available	Maximum allowed charging power	kW
Discharge available	Maximum allowed discharging power	kW
BESS SOC total	State of Charge of BESS	%
BESS DC/DC - Power Setpoint	Request power transfer from/to BESS	kW

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Enable BESS DC/DC	Latch On = Enabled Latch Off = Disabled	-
BESS DC/DC Update Setpoint	Control to update Setpoints Send 1 to trigger update	-

4.2.1.8 Arrigo BMS (Regin)

Arrigo BMS is the building management system from Regin that controls and monitors the mechanical and electrical systems that will be installed in the Psychology Faculty building. It is the case of the PV inverters or the power meters to monitoring the energy consumption.
Arrigo BMS provides the data through a REST API. Currently, the system is not available, but it is expected to be deployed and ready to be used soon.

4.2.1.9 EM24

It is used to control and monitor the energy consumption in the same way as the one described for 4.2.1.2 EM210. Two energy meters will be installed in Module 2 and Module 3, respectively, in the Computing Sciences Faculty building at University of Málaga.
The communication protocol is MODBUS TCP/IP. The adapter periodically connects and retrieves instantaneous data which is stored in the middleware. The polling period is once per five minutes and the variables that are collected are the ones shown in Table 4-5.

4.2.1.10 SACE EMAX2

The SACE EMAX2 is an energy meter. The system is used for:

- Reading and monitoring the building energy consumption
- Collection of historical data

Currently, the system is not available, but it is expected to be deployed soon. The system will be installed in Module 1 of the Computing Sciences Faculty building at University of Málaga, and it is used to control and monitor the energy consumption for this module in the building. The communication protocol is the same as other energy meters: MODBUS TCP/IP. The adapter periodically connects and retrieves data (see Table 4-5) that is stored in the middleware. The polling period is one time per five minutes.

4.2.1.11 WM30

The power meter operates in a similar way than the rest of energy meters, for example the one described in Section 4.2.1.2 EM210. Currently, this device is not installed but it is expected to be installed and used soon. The WM30 power meter will be installed in Sport Center at UMA. The communication protocol is MODBUS TCP/IP and the polling period is once per five minutes, (see Table 4-5).

4.2.1.12 WM20

The WM20 energy meter, uses Modbus TCP/IP and is almost identical to the WM30 (see 4.2.1.12). This system is installed in Module 4 of the Computing Sciences Faculty building at University of Málaga. The polling period is once per five minutes. And the communication protocol is Modbus TCP/IP. The variables are the same as the ones shown in Table 4-5.

4.2.1.13 PV1000A

The PV1000A is a smart logger that monitors and manages the photovoltaic power system deployed in the Computing Science Faculty at University of Málaga. It transparently converts protocols, collect, and save data, monitor, and manages all the devices of the photovoltaic power system.
At University of Málaga, the grid of PV panels consists of 4 inverters connected to four different set of photovoltaic panels. Each of this inverter is connected to one power meter and in turn with one SmartLogger. The SmartLogger is the one in charge of controlling and managing the power meter and the inverter. Four different SmartLoggers and four different adapters make it possible to collect information (see Table 4-13) and store it in the middleware. The information

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

and the communication with Smart Loggers are provided via the Modbus TCP/IP protocol. The polling period is once per five minutes

Table 4-13 UMA: PV1000A variables

VARIABLE	DESCRIPTION	UNIT
Date&time	Epoch sec UTC	N/A
The local time	Epoch seconds, local time of theSmartLogger	N/A
Trasfer trip	0:Run; 1:Fautl outage. The device shuts down when it stop due it faults and doesn't respond to the starip request	N/A
DC current	Equals the total input DC current of all inverters. If the value exceeds the rage specified by U32, use reg 40554	A
Input power	Equals the total input power of all inverters	kW
CO2 reduction	Equals the total CO2 reduction of all inverters. If the value exceeds the rage specified by U32, use reg 40550	kg
Active power	Equals the total active output power of all inverters	kW
PF	Equals the total power factor of all inverters	N/A
Reactive power	Equals the total reactive output power of all inverters	kVar
DC current 2	Represent a larga value compared with register 40524	A
E-Total	Equals energy yield generated by all inverters	kWh
E-Daily	Equals daily energy yield generated by all inverters	kWh
Duration of daily power generation		h
Plant status	0: Idle; 1: On-grid; 2: On-grid: self-derating; 3: On-grid: Power limit; 4: Planned outage; 5: Power limit outage; 6: Fault outage; 7: Communication interrupt	N/A
Phase A current	Equals the sum of phase A currents of all inverters	A
PhaseB current	Equals the sum of phase B currents of all inverters	A
Phase C current	Equals the sum of phase C currents of all inverters	A
Uab	Line voltage between phases A and B	V
Ubc	Line voltage between phases B and C	V
Uca	Line voltage between phases C and A	V
Max. reactive adjustment	The real-time range for reactive power adjustment. Equals the total max. power of all inverters connected in parallel multiplied by 60%	kVar
Min.reactive adjustment	The real-time range for reactive power adjustment. Equals the total max. power of all inverters connected in parallel multiplied by 60%	kVar
Max.active adjustment	The real-time range for active power adjustment. Equals the total max. power of all inverters connected in parallel multiplied by 60% x (-1)	kW
Locked	0: Locked; 1: Unlocked; If more than one inverter is on-grid and feeding power to the grid, the status is Unlocked.	N/A
CO2 emission reduction coeficient	[0-10]	kg/kWh

The implemented adapter also allows the power of the inverter to be limited to a given percentage.

4.2.2 UNC

The Unical demo site located at University of Calabria in Italy follows a different approach in terms of communication. The data from the demonstrator is collected in a centralized MySQL database provided by Amazon. The ebalance plus system and more specifically each adapter can query the database to obtain the needed information. As explained in Section 1 ebalance plus encourages the use of a distributed architecture. This does not mean that it cannot be adapted to a centralized scenario like the one at UNC. However, the centralized scenario is

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

more likely to be found in already existing deployments that need to be adapted to ebalance plus. For new deployments the distributed approach is preferred.

4.2.2.1 SMARTMETER

The smart meters are configured to send the measures every 5 seconds to the application server which saves the records on the DB MySQL. For each monitored structure/building more than one smart meter is installed so the data of the smart meters are collected and averaged every 15 minutes and stored in a separate table. Overall, the following aggregated values (see Table 4-14) are provided:

- **1 smart meter in Cubo18B inside Office Building**
- **1 smart meter in Cubo44B inside Office Building**
- **1 smart meter in office Cubo31B inside Office Building**
- **1 smart meter in office Cubo41B inside Office Building**
- **1 smart meter in Monachi121 inside Residential Building**
- **1 smart meter in Monachi122 inside Residential Building**
- **1 smart meter in Monachi123 inside Residential Building**
- **1 smart meter in Monachi124 inside Residential Building**
- **1 smart meter in Monachi223 inside Residential Building**
- **1 smart meter in Chiodo2 inside Experimental Building**

Table 4-14 UNC: smart meter parameters

VARIABLE	DESCRIPTION	UNIT
v1, 2, 3	Voltage value measured by the first, second and third phase	V
p1	Active power adsorbed by loads connected on the first phase	W
p2	Active power adsorbed by loads connected on the second phase	W
p3	Active power adsorbed by loads connected on the third phase	W
q1	Reactive power adsorbed by loads connected on the first phase	VAr
q2	Reactive power adsorbed by loads connected on the second phase	VAr
q3	Reactive power adsorbed by loads connected on the third phase	VAr
q4	Reactive power adsorbed by loads connected on the fourth phase	VAr
q5	Reactive power adsorbed by loads connected on the fifth phase	VAr
q6	Reactive power adsorbed by loads connected on the sixth phase	VAr
created_at	Timestamp relating to the creation of each row	
measured_at	Timestamp relating to the last measure time	

Our adapter periodically perform database queries and retrieves instantaneous data which is stored in the middleware. The polling period is once per fifteen minutes.

4.2.2.2 PV INVERTER

The functionality of PV Inverters is similar that the smart meters, they are configured to send the measures every 5 seconds to the application server which saves the records on the DB MySQL. For each monitored structure/building more than one smart meter is installed so the data of the PV Inverters are collected and averaged every 15 minutes and stored in a separate table. Overall, the following aggregated values (see Table 4-15) are provided:

- **1 PV Inverter in Cubo18B inside Office Building**
- **1 PV Inverter in Cubo44B inside Office Building**
- **1 PV Inverter in office Cubo31B inside Office Building**
- **1 PV Inverter in office Cubo41B inside Office Building**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

- **1 PV Inverter will be installed in Monachi inside Residential Building**

Table 4-15 UNC: Office and residential PV inverter parameters

VARIABLE	DESCRIPTION	UNIT
p4	Active power produced by PV plant connected on the first phase	W
p5	Active power produced by PV plant connected on the second phase	W
p6	Active power produced by PV plant connected on the third phase	W
created_at	Timestamp relating to the creation of each row	
measured_at	Timestamp relating to the last measure time	

The adapter periodically perform database queries and retrieves instantaneous data which is stored in the middleware. The polling period is once per fifteen minutes.

4.2.2.3 AMPERE

It is the same device as the one explained in section 4.2.1.4 AMPERE. Currently, this device is not installed but it is expected to be installed and used soon. The plans include the following installations:

- **Cubo Office Building:** 2 batteries.
- **Monachi Residential Building:** 5 batteries.

4.2.2.4 IOT DEVICES

Currently, these devices have not been installed but they are currently being tested and are expected to be installed and used soon. All IOT devices will be installed in Residential Monachi Building. The difference IOT devices will be:

- **Shelly H&T sensor. Smart humidity and temperature sensors: gather humidity and temperature data.**
- **Shelly Plus 1 and Shelly Plus 1PM. Smart Plugs (with optional power metering capabilities): Allows selected devices to be turn on/off and to measure power consumption.**
- **Xiaomi Light sensor. Light sensors: Collect luminosity readings**

4.2.2.5 SMART APPLICANCES

Currently, these devices are not installed but it is expected to be installed and used soon. All Smart appliances will be used in Residential Monachi Building. And The different Smart appliances will be:

- **Air Conditioning System:** REST API to control and monitor the system.
- **Tekka dishwasher:** REST API to control and monitor the device
- **Fridge (optional):** REST API to control and monitor the device

4.2.2.6 SMART METER CHIODO2

This smart meter is installed inside Experimental Chiodo2 Building and it is configured to send the measures every 5 seconds to the application server which saves the records on the DB MySQL. The smart meter is collected and averaged every 15 minutes and stored in a separate table. Overall, the following aggregated values (see Table 4-16) are provided:

Table 4-16 UNC: Chiodo2 smart meter parameters

VARIABLE	DESCRIPTION	UNIT
p1	Active power produced by PV plant connected on the first phase	W
p2	Active power produced by PV plant connected on the second phase	W
p3	Active power produced by PV plant connected on the third phase	W

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

created_at	Timestamp relating to the creation of each row	
measured_at	Timestamp relating to the last measure time	

The adapter periodically perform database queries and retrieves instantaneous data which is stored in the middleware. The polling period is once per fifteen minutes.

4.2.2.7 BATTERY NANO_GRID_2

This battery is installed inside Experimental Chiodo 2 Building, and it is monitored through the nano_grid_2 table. See Table 4-17.

Table 4-17 UNC: Battery nano grid values

Name	Description nano_grid_2	Unit
p_ups	Power exchanged with the UPS [W] – Not monitored (now)	W
p_batt	Power exchanged with the battery [W]	W
v_bus_dc	Nanogrid DC bus voltage [V]	V
v_rms_ups	RMS UPS voltage [V]	V
v_batt	Battery Voltage [V]	V
i_rms_ups	RMS supplied ups current [W] – Not monitored (now)	W
i_batt	Exchanged battery current [A]	A
fault_ups	0/1: 1 if a fault do to the ups occurs	0/1
stato_batt	0/1 : Battery connected/disconnected	0/1
stato_ng	It indicates the state of the Nanogrid	-
on_off	0/1: off/on state	0/1
soglia	[0 to 5] nanogrid operative voltage threshold	0-5
temp_bms	Battery Temperature [°C]	ºC
soc_bms	Battery state of charge [%]	%
i_charge_bms	Maximum battery charge current [A]	A
time_save	Unix timestamp	

The adapter periodically perform database queries and retrieves instantaneous data which is stored in the middleware. The polling period is once per fifteen minutes.

4.2.2.8 NANO_GRID_1

The nano grid (1 and 2) at UNC is monitored through nano_grid tables. The adapter collects the data in Table 4-18 and Table 4-19.

Table 4-18 UNC: Nano grid 1 parameters

Name	Description nano_grid_1	Unit
p_rete	Power exchanged with the grid [W]	W
p_pv	Power exchanged with the PV system [W]	W
v_bus_dc	Nanogrid DC bus voltage [V]	V
v_rms_rete	RMS grid voltage [V]	V
v_pv	PV voltage [V]	V
i_rms_rete	RMS exchanged grid current [A]	A
i_pv	Supplied PV current [A]	A

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

fault_rete	0/1: 1 if a fault do to the grid occurs	0/1
stato_rete	0/1 : Grid connected/disconnected	0/1
stato_pv	0/1 : PV connected/disconnected	0/1
stato_ng	It indicates the state of the Nanogrid	-
on_off	0/1: not operative/operative state	0/1
soglia	[0 to 5] nanogrid operative voltage threshold	0-5
time_save	Unix timestamp	

Table 4-19 UNC: Nano grid 2 parameters

Name	Description nano_grid_2	Unit
p_ups	Power exchanged with the UPS [W] – Not monitored (now)	W
p_batt	Power exchanged with the battery [W]	W
v_bus_dc	Nanogrid DC bus voltage [V]	V
v_rms_ups	RMS UPS voltage [V]	V
v_batt	Battery Voltage [V]	V
i_rms_ups	RMS supplied ups current [W] – Not monitored (now)	W
i_batt	Exchanged battery current [A]	A
fault_ups	0/1: 1 if a fault do to the ups occurs	0/1
stato_batt	0/1 : Battery connected/disconnected	0/1
stato_ng	It indicates the state of the Nanogrid	-
on_off	0/1: off/on state	0/1
soglia	[0 to 5] nanogrid operative voltage threshold	0-5
temp_bms	Battery Temperature [°C]	ºC
soc_bms	Battery state of charge [%]	%
i_charge_bms	Maximum battery charge current [A]	A
time_save	Unix timestamp	

4.2.2.9 INVERTER FRONIUS/PV PLANT CHIODO2

It is monitored with the smart meter named SMEBPLUSCHIODO2. It is installed in the Experimental Building and provides the parameters shown in Table 4-20.

Table 4-20 UNC: Inverter Fronius/PV plant parameters

VARIABLE	DESCRIPTION	UNIT
p4	Active power produced by PV plant connected on the first phase	W
p5	Active power produced by PV plant connected on the second phase	W
p6	Active power produced by PV plant connected on the third phase	W
created_at	Timestamp relating to the creation of each row	
measured_at	Timestamp relating to the last measure time	

The adapter periodically perform database queries and retrieves instantaneous data which is stored in the middleware. The polling period is once per fifteen minutes.

4.2.3 JUNIA

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

The JUNIA demo site takes place in three different buildings: HEI (Hautes Etudes d'Ingénieur) Building (5RNS building and 13RT building), HA (Hotel Académique) Building and Rizomm Building as shown in Figure 4.2.

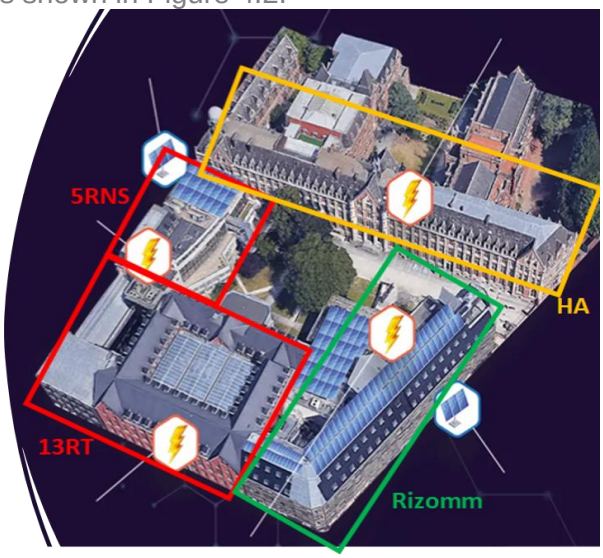

Figure 4.2 JUNIA demo site buildings

All the information in this demo site is provided by means of a centralized REST API which gives information about the following subsystems:

- **Smart meters (Consumption)**
- **Inverters**
- **Hot water storage**
- **HVAC (Heating, Ventilation, and Air Conditioning)**
- **Aurion (classroom reservation system)**
- **Hyperplanning**
- **Weather Station in 'common' location, accessible by all locations**

4.2.3.1 SMART METER

Smart meters provide information of consumption of each building. There are smart meters available for all buildings: HEI (13RT, 5RNS), Rizomm, HA.
The information is provided in the cloud via the REST API that has been used by our adapter to collect the parameters shown in Table 4-21.

Table 4-21 JUNIA: smart meter readings

VARIABLE	DESCRIPTION
measurement_name	The name of the building
Value	Consumption of the building
Unit	W
last_updated_at	Datetime when generate value of consumption

4.2.3.2 INVERTER

The inverters get information about PV production. There is PV Inverter data available from HEI and Rizomm buildings. The information is provided in the cloud via a REST API that it has been used by our adapter to collect the parameters shown in Table 4-22.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Table 4-22 JUNIA: PV inverter variables

VARIABLE	DESCRIPTION
measurement_name	The name of the building
value	Production of the building
unit	W
last_updated_at	Datetime when generate value of production

The REST API offer an endpoint to modify the max amount of power of a given inverter. The value must be specified in Watts (W).

4.2.3.3 HOT WATER STORAGE

The hot water storage endpoint gives hot water consumption information. This is available for HEI, Rizomm and HA buildings. The information is provided in the cloud via a REST API that it has been used by our adapter to collect the parameters shown in Table 4-23. The REST API also provides an endpoint for changing the status of the hot water storage system for each building (On/Off).

Table 4-23 JUNIA: Hot water consumption variable

VARIABLE	DESCRIPTION
measurement_name	The name of the building
value	Consumption of hot water storage system of the building
unit	kW
last_updated_at	Datetime when generate value of production

4.2.3.4 HVAC

The HVAC system provides a high number of parameters that can be obtained and monitored coming for a different number of sensors deployed in the rooms. E.g.: co2, temperature, occupancy, heat valve status, etc. Given the high number of variables available a current analysis of the algorithm requirements is being carried out to limit data collection to the endpoints needed (selected rooms). The HVAC system is available from the API for the HEI building and its development is currently in progress for the Rizomm building.
The values that the HVAC provides are:

- **Co2**. The unit for this value is ppm
- **Damper position**. The unit for this value is %
- **Ambient temperature**. The unit for this value is ºC
- **Occupancy mode**. Integer number describing the occupancy
- **Heat valve position**. The unit for this value is %

The REST API offer an endpoint for changing setpoints of the HEI HVAC System. The setpoints are positive integers and can either be for Temperature or CO2 ppm.

4.2.3.5 AURION (PLANNING SOFTWARE)

It is a software for classroom reservations. The information is provided in the cloud via a REST API that it has been used by our adapter to collect the parameters shown in Table 4-24. With the REST API the reservations for HEI building for today and tomorrow are provided.

Table 4-24 JUNIA: Aurion class reservation software variables

VARIABLE	DESCRIPTION	UNIT
classcode	The name of classroom	String

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

numplaces	Number of places of the classroom	Integer
eventstart	Timestamp to start the reservation	Datetime
eventend	Timestamp to end the reservation	Datetime

4.2.3.6 WEATHER STATION

The Junia demonstrator provides weather data coming from a weather station that is used by all three buildings. The following information is provided:

- **Humidity.** The unit for this value is '%'.
- **Temperature**. The unit for this value is 'ºC'.
- **Solar irradiation**. The unit for this value is 'W/m2'.
- **Wind speed**. The unit for this value is 'm/s'.
- **Wind direction**. The unit for this value is 'º'.
- **Precipitation**. Get value of rain. The unit for this value is 'mm'.

4.2.3.7 BATTERY

Batteries are common for all three buildings and provide state of charge and current power information. They also provide a setpoint for specifying the charge/discharge power. Now, this endpoint is available through the API, but some issues have been detected with the equipment and is currently taken care of by the manufacturer. The information provided by this system is described in Table 4-25.

Table 4-25 JUNIA: battery variables

VARIABLE	DESCRIPTION	UNIT
sto_power	[INPUT/OUTPUT] Power	W
sto_soc	State of charge	none (%)
sto_pcharge_max	Maximum power for charging	W
sto_pdischarge_max	Discharge maximum power	W
sto_capacity	Capacity	Wh
sto_energy_discharge	Available energy for charge	Wh
sto_energy_charge	Available energy for discharge	Wh

4.2.3.8 EV CHARGERS

At the moment, no EV chargers are available through the API, but its setup and integration are expected to start soon.

4.2.3.9 HYPERPLANNING

Hyperplanning is a software system like the one explained in Section 4.2.3.5. This software provides information about class reservation for the Rizomm building. This information is not available now, but it is expected to be integrated in the API.

4.2.4 DTU

The DTU demo site consists of 30 Summer houses with a swimming pool heated by a heat pump (see Figure 4.3). A REST API makes it possible to monitor the status of the swimming pool and to actuate on the valve that controls the flow of hot water. Using the REST API it is possible to obtain the water temperature forward and water return temperature of the swimming pool and the state of the valve (open/close) as shown in Table 4-26. Finally, the heat pump valve can be controlled through this REST API.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

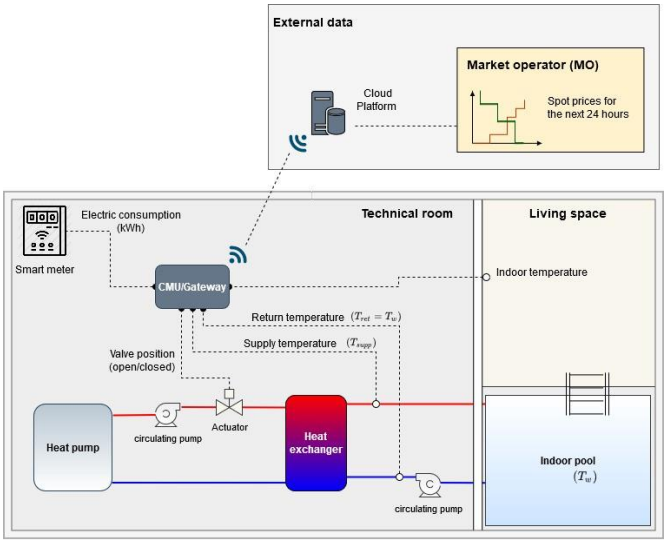

Figure 4.3 DTU heat pump diagram

Table 4-26 DTU monitoring variable

VARIABLE	DESCRIPTION	UNIT
read_timestamp	Unix timestamp	
water_forward_temperature	Water forward temperature	ºC
water_return_temperature	Water return temperature	ºC

The adapter periodically perform database queries and retrieves instantaneous data which is stored in the middleware. The polling period is once per minute for each of the monitored houses.

4.2.5 In-lab demo

The in-lab demo gives great flexibility when it comes to testing different scenarios. In this sense any type of information can be simulated. The information that will be stored and used in this demo site is not completed yet and the design of the corresponding experiments is still in progress.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

5 Demo site prototype overview

Section 3 has presented the term MU in the context of the ebalance plus project. Then, in Section 4 the different integrations done to connect an instance of the middleware and an external device have been shown. This section depicts the planned global architecture for each demo site showing how many MUs are expected for each demo site, and in which instances the data coming from adapters is collected. Note, that these diagrams are a work in progress and might change in the final deployment based on a change in requirements.

5.1 UMA

The University of Málaga demo site is composed of four different buildings all of them connected to a single MVGMU (UMA_MV_1) which is on the top of the MU hierarchy.

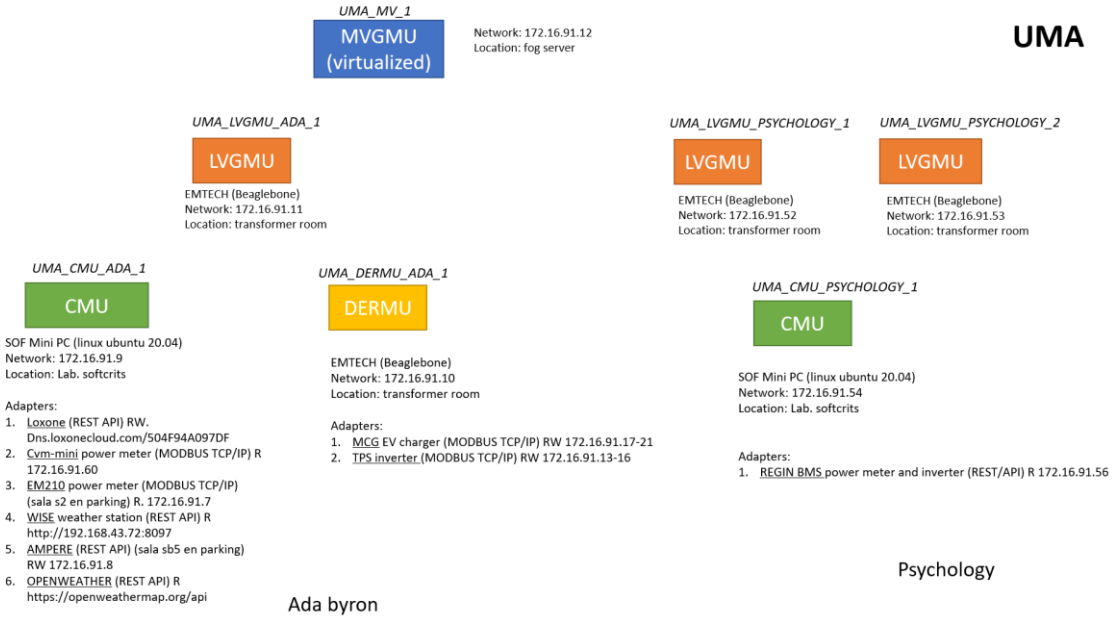

Figure 5.1 UMA: Ada Byron and Psychology faculty connectivity scheme

Figure 5.2 UMA: Sport centre and Computing science faculty connectivity scheme

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

The connectivity scheme for the Ada Byron building is depicted in the left side of Figure 5.1. One CMU (UMA_CMU_ADA_1) and one DERMU (UMA_DERMU_ADA_1) are in charge of collecting information from the devices explained in previous sections. These two units are connected to a single LVGMU (UMA_LVGMU_ADA_1). Finally, the Ada Byron demo site and the rest of buildings in the UMA demonstrator are monitored by a single MVGMU (UMA_MV_1) which is expected to be installed in the fog server described in Section 3.1.2.
The information collected from the Psychology faculty (right side of Figure 5.2) comes from the Arrigo BMS by the manufacturer Regin. This information is handled by a CMU (UMA_CMU_PSYCHOLOGY_1) which connects to a LVGMU (UMA_LVGMU_PSYCHOLOGY_1). An additional LVGMU (UMA_LVGMU_PSYCHOLOGY_2) is deployed. These 2 LVGMU are connected to UMA_MV_1. The Sport centre at University of Málaga (see Figure 5.2) will have one CMU (UMA_CMU_SPORT_1) and two LVGMUs (UMA_LVGMU_SPORT_1 and UMA_LVGMU_SPORT_2). Finally, the computing science faculty (Figure 5.2) contains one CMU (UMA_CMU_COMPUTING_1) and a DERMU (UMA_DERMU_COMPUTING_1) on the lowest level of MUs and two LVGMU at the middle layer.

5.2 UNC

The UNC demo site (see Figure 5.3) located at University of Calabria in Italy is a complex demo site that spans different locations: Cubo office buildings, Monaci residential buildings, Mega centrale (primary MV/MV substation) and Chiodo2 building.

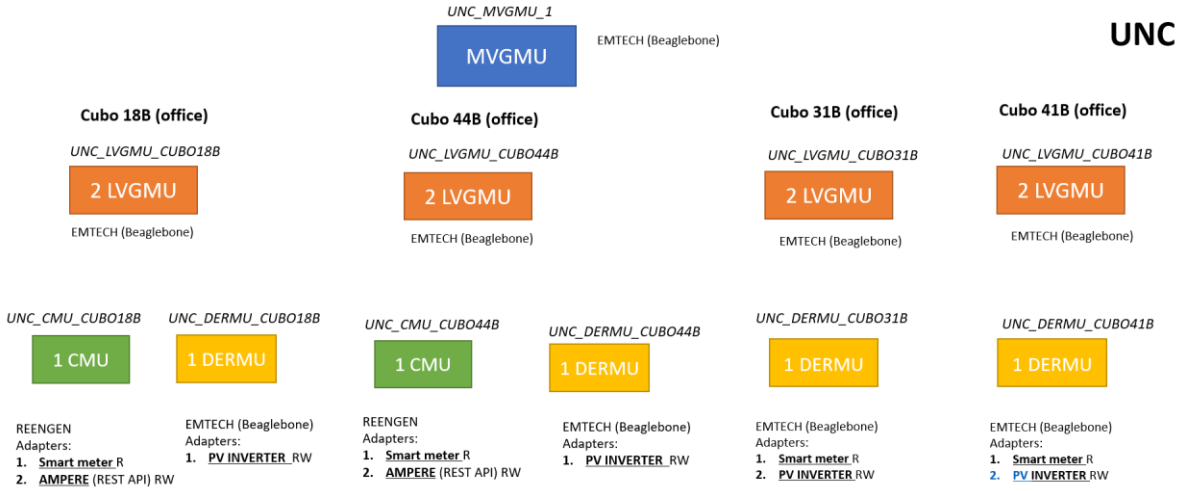

Figure 5.3 UNC: Connectivity scheme of the Cubo residential buildings

Cubo residential buildings span 4 different buildings (18B, 44B, 31B and 41B). For buildings 18B and 44B, 1 CMU, 1 DERMU and 2 LVGMU for each are deployed. For buildings 31B and 41B only 1 DERMU is deployed and 2 LVGMU. In the same way as for the UMA demo site only 1 MVGMU (UNC_MVGMU_1) is deployed for the whole demonstrator.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

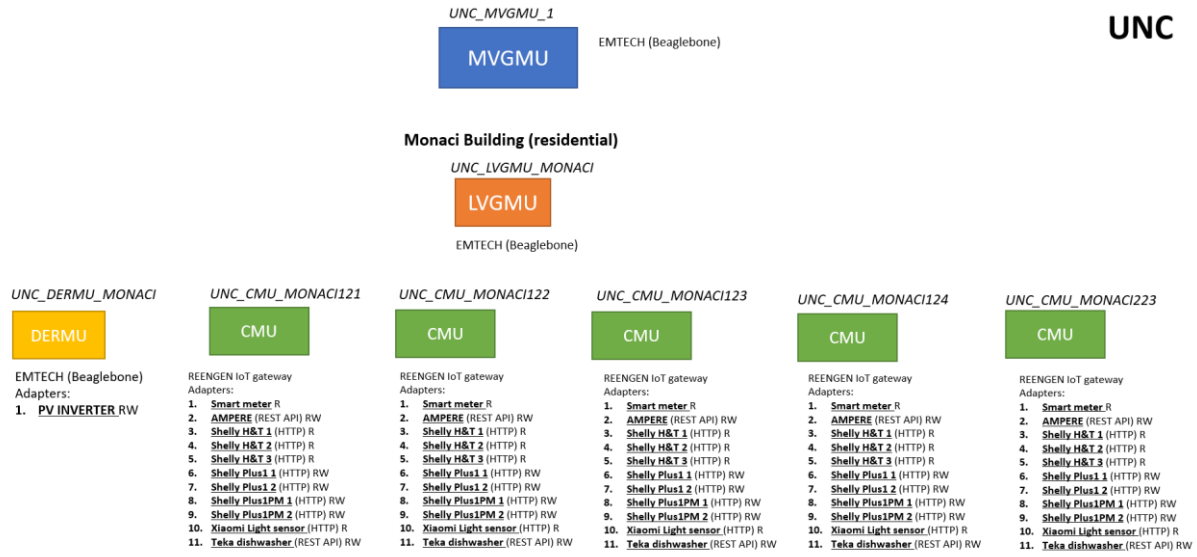

Figure 5.4 UNC: Connectivity scheme of Monaci residential buildings

For the Monaci residential buildings (shown in Figure 5.4) a single DERMU is installed to monitor and control the PV inverters. Then one CMU per building (121, 122, 123, 124 and 223) is used to collect information about IoT devices, smart appliances, batteries, and smart meters. All the CMUs and the DERMU are connected to a single instance of a LVGMU (UNC_LVGMU_MONACI).

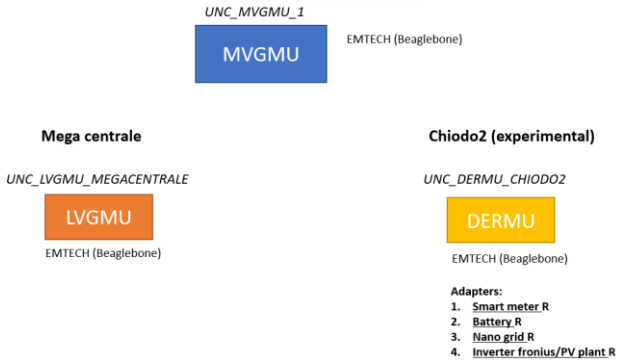

Figure 5.5 UNC: Connectivity scheme of Mega centrale and Chiodo2 building

Finally, the Mega centrale primary substation is monitored by an LVGMU (UNC_LVGMU_MEGACENTRALE) and the Chiodo2 experimental building by a DERMU (UNC_DERMU_CHIODO2).

5.3 JUNIA

The JUNIA demo site (see Figure 5.6) takes place in three different buildings, presented in Figure 5.6. Each building is monitored by a CMU (JUNIA_CMU_HEI_1, JUNIA_CMU_RIZOMM_1, JUNIA_CMU_HA_1). Each CMU oversees the information collection of the devices deployed within the building. An LVGMU (JUNIA_LVGMU_1) which is going to virtualized, will connect to each CMU.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

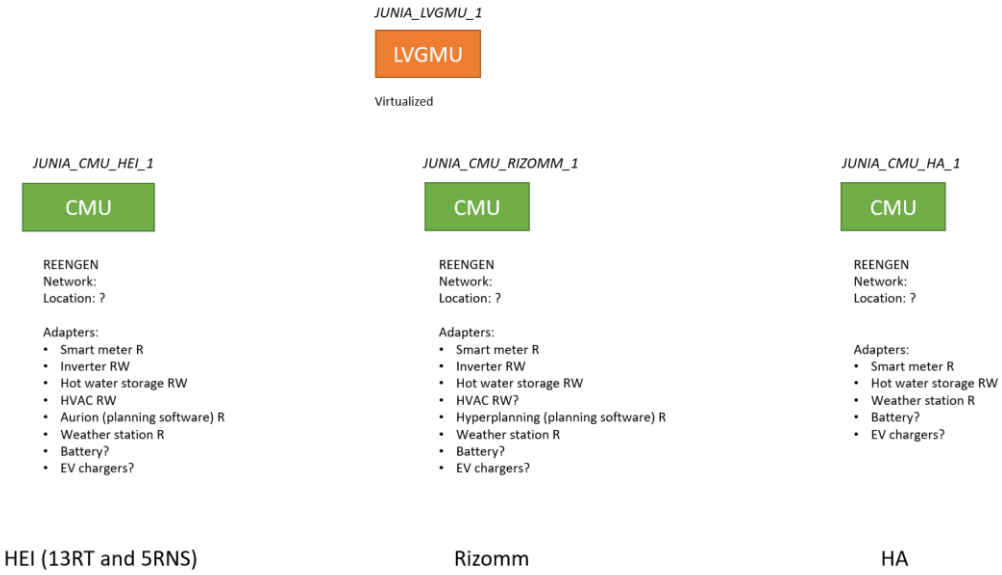

Figure 5.6 JUNIA: connectivity scheme

5.4 DTU

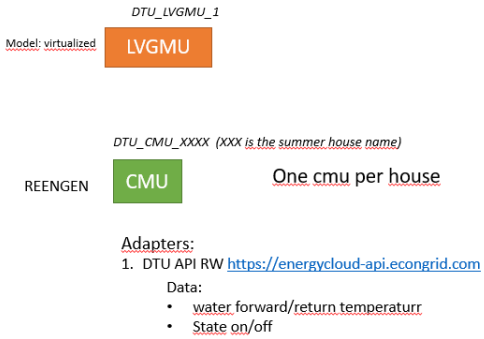

Figure 5.7 DTU: connectivity scheme

The DTU demo site (see Figure 5.7) is composed of a set of summer houses, each of them monitored by a CMU. All CMUs are connected to a single instance of an LVGMU (DTU_LVGMU_1).

5.5 In-lab demo

The In-Lab demonstrator (see Figure 5.8) consists of 24 prosumer blocks, each containing a management unit (MU) that can be representing either a CMU or a DERMU, depending on the configuration of the block. The demonstrator consists also of eight (8) secondary substation (SS) blocks, each containing a LVGMU and one primary substation (PS) block with a MVGMU. Finally, there is the TLGMU. All these management units are connected to one local network, but any other partitioning in sub-networks is also possible. The MUs can talk to each other directly, but it is also possible to involve the proxy server for that, to check the caused communication overhead.
On each MU, there are respective adapters. All MUs involve the smart meter adapter to interact with the measurement equipment and actuators (including the transmission line blocks on LVGMUs and MVGMUs). Additionally, CMUs and DERMUs involve the behavior simulator adapter, to interact with the simulation generating the energy profile.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

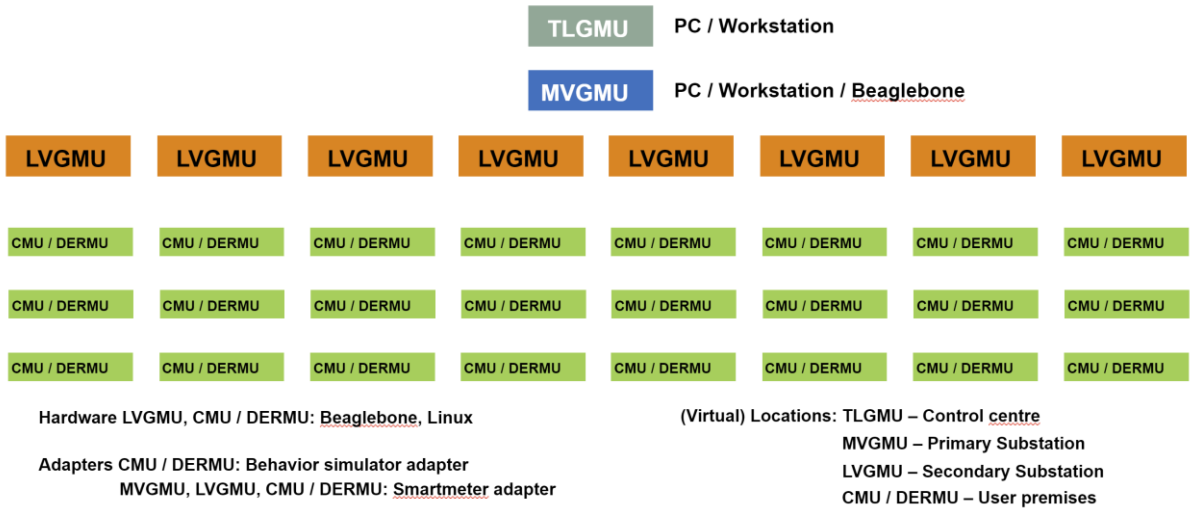

Figure 5.8 In-Lab: connectivity scheme

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

6 Conclusions

A set of prototypes MU have been developed to be tested in several demo sites. An MU is a physical or virtualized autonomous entity that contains the software necessary to:

- Communicate among themselves using the data exchange middleware
- Gather information from external devices or APIs using the adapters
- To run general purpose algorithms that made use of the information collected by the local or remote MUs

The developed MUs all run Linux operating system with preconfigured services that automatically run when the device boots up. A single instance of the middleware with its embedded database is installed in each MU. Also, based on the location of the MU specific adapters are installed as separate java processes (which also run as Linux services) to collect information and store in the local database. Four different demo sites and an additional in-lab demo have been presented with an estimated number of deployed MUs higher than thirty. Some of the most used communication protocol supported by the ebalance plus system are REST API, Modbus TCP/IP, web socket and SQL database.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

References

[1] Ebalanceplus project, "D5.3 Data exchange middleware specification," 2022.

[2] Ebalanceplus project, "D6.1 Evaluation methodology," 2021.

[3] Ebalanceplus project, "D3.4 - Specification and implementation of grid automation and control devices and interfaces"

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283