TU WIEN Informatics

Fiware-based Architecture for Smart Local Energy Communities

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Computer Engineering

by

Max Albin Thoma
Registration Number 11809931

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Ass. DDipl.-Ing. Dr.techn. Gernot Steindl
Assistance: Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner

Vienna, 8th February, 2023

Max Albin Thoma	Gernot Steindl

Erklärung zur Verfassung der Arbeit

Max Albin Thoma

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. Februar 2023

Max Albin Thoma

Acknowledgements

I want to thank my thesis advisor Univ.Ass. DDipl.-Ing. Dr.techn. Gernot Steindl for his support and many constructive feedback rounds. His mentoring helped me during all phases of this thesis.

Abstract

The goal of this thesis is to propose a suitable architecture for smart Local Energy Communities based on FIWARE which utilizes smart data models for optimized energy flow control and support virtual commissioning. First, an overview of existing data modelling frameworks is given and their differences in terminology is discussed. Then, a system design using the Industrial Internet Consortium's three-tier architecture pattern is developed. The FIWARE platform plays a key role in managing the entities of the Local Energy Community. With the aid of the co-simulation tool mosaik a virtual environment is created that allows simulating a Local Energy Community and its entities. It connects different simulators and interacts with the FIWARE platform to access the data models of the entities. For the evaluation three scenarios are simulated. The first scenario is a simple simulation set-up that gets augmented by a new virtual device, namely a smart energy meter. Afterwards, an optimization algorithm for energy flow control is deployed and the capability of the system for seamlessly integrating new devices is evaluated. It is shown that the proposed system design is capable of supplying the Optimizer with vital context information. Moreover, it can act as virtual commissioning environment for Local Energy Communities.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The need and demand for sustainable and green energy has never been more apparent. The *Paris Climate Agreement* in 2015 was an important step from the international community to tackle the issue of rapid climate change [1]. Four years later, in 2019 the European Union published the *Clean Energy For All Europeans* package, which has as one of its objective to comply with the aforementioned agreement [2]. However, this is not the sole goal of the legislation. It also aims to provide means for '*clean and fair energy [. . .] at all levels of the economy [. . .] all the way to people houses*' [2]. Improving energy efficiency and the extension of renewable energy production will play a key role in the implementation. Energy communities are an essential tool in this legislation and will enable the modernization of the energy market. What is more, they will empower consumers which is also an intent of European Union. The decentralization and stabilization of the grid is a future goal for them, too.

Local Energy Communities (LECs) have already been a proven concept for achieving community energy goals, which would not have been viable in a conventional setting. They can vary in size and structure, but all complete the goal of empowering the (local) community and provide means to sustainable energy sources. They were first officially recognized by the aforementioned legislation. In general, there is a distinction been made between so-called *Citizen Energy Communities* (CEC) and *Renewable Energy Communities* (REC). The main difference is that Renewable Energy Communities are communities that are confined by a smaller geographic region and their members should be living close together [3]. The overall goal of providing and producing renewable and sustainable energy sources to domestic consumers in the community still remains the same. In this thesis, the emphasis lies more on REC.

There exist numerous established LECs around Europe that have shown the feasibility and opportunities of the concept. Now follow three examples of early LECs that shall highlight their strength. The first example is the community pool in Dingen, Germany. A solar installation on public roofing with a (peak) capacity of 245 kWp and an annual production of 200 000 kWh allows the pool to be opened more months of the year, even in the off-season [4]. Notably, even small project can achieve a big impact. In Lohtaja, Finland a small scale biomass heating plant was built and is managed by a small cooperation with 40 members. The wood chips that are consumed by the boiler come from the local forest. Even though it is only a small scale heating plant it has supplanted the need for 100 000 litres of heating-oil [5]. What is more, also LECs that have focused on improving the energy efficiency of the whole community have been proven to be a success. In Erris, Ireland, various improvements through the whole community were carried out. This includes upgrades to energy efficient lighting, the installation of intelligent storage heater and the insulation of buildings [6]. The estimated energy savings for the whole community in 2016 alone were 373 470 kWh.

In Austria, the legislation that enabled Energy Communities as described in the *Clean Energy for all Europeans Packag* was passed and became active in July 2021 [7]. It is called '*Erneuerbaren-Ausbau-Gesetzespaket*' and it defines two types of Energy Communities very similar to RECs and CECs mentioned above. At the time of writing, there are around 90 communities registered online at [7], with trends going up.

1.2 Problem Statement

The exemplary LECs from the previous section were an illustration of how a community effort can achieve energy goals sustainably. They showed various ways and methods on how the implementation of LECs might look like. Although, all three projects have been successful in their own ways and benefit the local population significantly, they are not especially smart or intelligent solutions. In Austria, the legislation demands that participants of Energy Communities are equipped with Smart Energy Meters [8]. These devices are required to measure the power consumption at minimum every 15 minutes. This data acquisition might be frequent enough for some form of load balancing, however the network operator is only required to provide this data the following day. This makes it virtually impossible to use the data in a real-time control scenario. LECs can act solely as 'metering structure' where energy consumption and production is measured regularly, but not necessarily in real time. Members of the LEC who produce energy can sell their surplus to the grid.

If the goal is to use the resources of LECs to their full potential, more intelligent solutions are necessary. To enable concepts like agile load balancing, reliable real-time data is required, which the current structures cannot supply. Therefore, a more versatile solution is needed that can handle real-time data from production sites and consumers as well as provide semantic data models to facilitate interoperability.

Data is only useful if it is known or explicitly stated what it actually describes. This is not always the case: different application and standards may use the same name for different concepts or conversely use other nomenclature for the same topic. The context of any given data point is often only known inside an application or device, since vendors might use proprietary or simply different data representations. This is not an issue if only one ecosystem is deployed, yet it facilitates vendor lock-in. When connecting multiple independent systems that shall interact with each other, it almost becomes a necessity to document the context information of the used ontology. Using a framework like FIWARE's NGSI (Next Generation Service Interface) Context Broker, makes it very easy to explicitly state all context information and make them also available in a machine-processable form [9]. It becomes apparent that an open platform like FIWARE can help to boost interoperability and reusability of the data models.

We define a Smart Local Energy Community as a community that shares its resources in an intelligent fashion beyond a simple metering structure. This means that energy consumption and production data is captured in real-time and can be processed by an optimizing algorithm in order to balance the electrical loads of the community. We call this application the *Optimizer*.

A suitably smart data model can support us in achieving this more advanced and complex goal. It allows explicitly stating context information. What is more, the names of the attributes that are used in the data model are unique and well-defined. Plus, assumptions about the system and meta information will not be hidden away as implementation detail. When collecting data points from various devices and sources their context and all meta-data describing them will to be preserved. This facilitates historic trends analysis and forecasting functionality (e.g. of production or consumption) and makes an implementation of an *Optimizer* more transparent and reusable.

Smarter solutions like the smart LEC with its optimizing algorithm can become very complex. This implies longer and more involved development and testing cycles. If the implementation of an optimizing algorithm can only start when all hardware choices have been made, a longer than desirable implementation delay could occur. Also, if the program is dependent on the used hardware (and maybe even data-model) the reusability is severely harmed. Therefore, the need for a certain abstraction layer arises. It should be possible to develop an optimizing controller without having implicit assumptions on hardware choices. Thus, solutions should not be specific to certain Energy Communities. All the information about the system must be explicitly stated within the data models where they are visible and documented. A virtual test-bed with a structural identical architecture to the production system is needed to facilitate the development process. While most research uses simulation to analyse specific control strategy and algorithms, they rarely focus on setting up simulation as a virtual commissioning platform.

Our research aim is to help LEC operators to transform common LECs into smart LECs by providing open source solutions which can be easily deployed and extended. Ready to go co-simulation set-ups should make it simple to perform a virtual commissioning of the LEC before it is deployed to the real world. The solution should consist of a reference architecture and tooling for smart data models. Operators should be able to adapt the solution for their needs and develop new applications that integrate with the architecture. The openness would enhance the reusability of code and interoperability of different systems. The structure of this architecture and its tooling should be same whether the deployment is virtual or in the real world.

The objective of this Bachelor thesis is to explore the possibilities and opportunities of FIWARE's context enabled data models and assess their suitability to be used as base architecture for a load balancing algorithm in a smart LEC. Moreover, the open source co-simulation framework mosaik [10] will be used to support virtual commissioning. Its benefits for the development cycle and architecture will be analysed. The aim is to provide an architecture where an energy flow optimizing algorithm can be developed independent of the actual entities of the LEC. In other words, the Optimizer should be independent of hardware choices and size and structure of the community. There shall be no information hidden in the implementation of the Optimizer. It should gather all information that it needs to perform the load balancing from the underlying data models. Thus, our research question can be boiled down to: what is a suitable architecture for smart LECs based on FIWARE which utilizes smart data models for optimized energy flow control and support virtual commissioning?

CHAPTER 2

State of the Art

2.1 Terminology: Data Model vs. Information Model vs. Ontology

The terminology concerning information models, data models and ontologies is not always agreed upon. There does not seem to be a definitive answer in the literature to what the exact difference between these three terms are. The Eclipse Vorto [11] framework augments the vocabulary with the term device description and uses it in a similar fashion to information model [12]. There is, however, a differentiation made between ontologies and knowledge bases. In the 2002 guide *Ontology Development 101: A Guide to Creating Your First Ontology*, the authors Noy and McGuinness explain that a knowledge base can be derived from an ontology by creating instances of the defined classes [13]. It should be noted that the authors use an object-oriented approach to develop a '*taxonomical hierarchy*'. The *knowledge* comes from the class and instance variables. This means that an ontology can be seen as a form of blueprint for a knowledge base. Nevertheless, the authors state that in real-world applications the distinguishing between the two concepts can be very difficult. In Microsoft's Azure Digtal Twin (ADT) [14] framework, the term ontology describes the combination of models and the relationship between entities. They implement the so-called *'digital twin graph'* [15]. In this sense, it is used in a very similar way as in the aforementioned paper, if the term 'model' is replaced with 'class', while an 'entity' becomes a 'instance of a class'. On the other hand, FIWARE's documentation uses exclusively the term 'data model'. However, the data models can link to and reference (external) ontologies. This property is called the 'context information'. A special attribute (namely '@context') is used for these references.

The differentiation between information and data models seems to be more clearly defined. In 2003, there was a memo sent to the Internet Engineering Task Force (DMTF) which explains the difference between these two terms, when they are used to describe '*managed objects in network management*' [16]. When reading the memo it becomes apparent while the exact context might not be the same, it can still be applied in our application. The authors state that data models include '*implementation- and protocol-specific details*' [16]. This seems to be true for ADT as well as FIWARE. They both use the term 'model' and 'data model' respectively. Their data models are based on the JSON-LD (JavaScript Object Notation-Linked Data) format and communication with their APIs (Application Programming Interface) uses HTTP (Hypertext Transfer Protocol) requests. Thus, the definition of the memo seems to be in line with how these two frameworks use the terminology. In the same document ([16]), the authors also describe two important features of information models being that they are '*protocol neutral*' and allow defining '*relationships between objects*'. The authors denote, that an information model can be specified in a '*natural language*'. Two more terms for describing an information model are given: '*conceptual model*' and '*abstract model*'. Interestingly, the specification for NGSI-LD (Next Generation Service Interface-Linked Data) [17] (not FIWARES's documentation to be precise) uses the term *'information model'*. The following statement can be found in the specification: '*The NGSI-LD Information Model prescribes the structure of context information that shall be supported by an NGSI-LD system. It specifies the data representation mechanisms that shall be used by the NGSI-LD API itself. In addition, it specifies the structure of the Context Information Management vocabularies to be used in conjunction with the API.*' [17]. This seems to be more or less in line with the DMTF's definition.

Even though the name *information model* is not being actively used by all the aforementioned frameworks, this does not mean it is not used by other modelling applications. Two prominent examples are the Common Information Model [18] (CIM) developed by the CIM Forum which is part of the Distributed Management Task Force and the OPC Unified Architecture (UA) [19] which is maintained by the Open Platform Communications (OPC) Foundation.

When looking at the Wikipedia entry for information models, one can find the following statement: '*An information model provides formalism to the description of a problem domain without constraining how that description is mapped to an actual implementation in software. There may be many mappings of the information model. Such mappings are called data models, irrespective of whether they are object models (e.g. using UML), entity relationship models or XML schemas.*' [20] Unfortunately, there seems to be no direct source for this statement in the article. This definition seems to be compatible with the DMTF's memo. Nevertheless, the data modelling frameworks described in this thesis do not always seem to adhere to these exact definitions. They often use their own terminology. Especially the definition of information models seems to be very strict. For instance, the NGSI-LD specification has a UML representation of the NGSI-LD information model [17]. Following the definition, this *mapping* would again be a data model. It is not quite clear, how the '*formalism*' to describe an information could be anything other than written in natural language. It could be argued that the frameworks themselves represent a '*formalism to description of a problem domain*'. However, they do not meet the second criteria of this definition, since they are implementation-depending encodings. Except for the Vorto language, which claims to offer a '*technology-agnostic*' way to describe the information model [11]. Nevertheless, the quoted statement might be valid, if the terms are used in a more abstract way, i.e. when describing modelling in general, but it seems to be fuzzy when discussing specific frameworks.

To conclude, two frameworks cannot be compared by looking at the used terminology. Especially, when it comes to the fine details, lines become more blurred. In this thesis, FIWARE's terminology will be used, i.e. *(smart) data model*, for describing the (data) models that define the entities, and *context information* when describing meta information about a data model. The *meta information* further defines the name space and may link to ontologies.

2.2 Overview of existing data modelling frameworks

There exist a multitude of frameworks and development kits for managing Internet of Things (IoT) information models. In this section, there will be an overview and comparison of which frameworks exist. Specifically, the capabilities of including meta and context information will be evaluated.

2.2.1 FIWARE

FIWARE is the chosen platform and framework of this thesis. The FIWARE foundation describes itself as '*A curated framework of Open Source Platform components to accelerate the development of Smart Solutions*' [9]. The tool chain has a focus on what they call smart data models. They have a strong emphasis on preserving and processing context and meta information about entities. The heart piece of the framework is the so-called Context Broker. It implements the *Next Generation Service Interface* (short NGSI) which is an API that can be used by other software components. The Context Broker is so essential in the FIWARE-ecosystem that any platform which incorporates it may call itself '*Powered by FIWARE*' [21]. There exist multiple implementations of the Context Broker, notably FIWARES own, *Orion*[1]. There currently exist two versions of the NGSI standard: NGSI v2 which is designed by FIWARE [22] and NGSI-LD (NGSI-Linked Data) which is published and standardized by the European Telecommunications Standards Institute (ETSI) [17]. Both interfaces are open standards with freely available specification documentation. The latter was chosen for this paper, since it is the newer standard and seems to have more momentum than NGSI v2. Orion-LD [23] was used as the Context Broker: it implements both the NGSI-LD specification version 1.3.1[2] [24] as well as NGSI v2. NGSI-LD is based on NGSI v2, but has some interesting enhancement.

The main differences between the specifications are explained in the FIWARE documentation [25] and are summarized as follows:

[1]Note: the so-called Context Broker, Orion **should not** be confused with *Eclipse Orion*, which is a different program, that is **not** used in this thesis.

[2]Note: As of writing the latest version of the NGSI-LD API is version 1.6.1. The older version that Orion-LD implements can be found here [24].

- A NGSI-LD payload is a subset of JSON-LD and uses many of its feature, notably the *‘@context’* attribute.
- The IDs of NGSI-LD entities are Universal Resource Identifiers (URIs).
- In NGSI v2 a dictionary is used to describe context information. In NGSI-LD, meta data is encoded as nested properties (of properties).
- In NGSI-LD, there is a richer standard name space for describing meta data.
- NGSI-LD allows for attributes to describe relationships and link to other entities.
- NGSI-LD has a special property (called *GeoProperty*) to encode location data.

In particular the first point, the integration of JSON-LD is an important enhancement over the NGSI v2.

As stated above, the NGSI-LD API uses a subset of JSON-LD as its payload. JSON-LD is defined and standardized by W3C at [26]. In the standard, it is described as a ‘*lightweight syntax to serialize Linked Data in JSON*’. JSON-LD can be used with almost all existing JSON infrastructure since it is designed to be ‘*100% compatible with JSON*’. An important feature of this standard is the *‘@context’* attribute that allows to embed and link to resources that describe the context of any given object. It is also used to translate between name spaces that might overlap or use different identifiers for properties. This plays a key role in the FIWARE platform as can be seen in their documentation concerning *‘@context’* [25, 27, 28, 29].

2.2.2 Eclipse Vorto

This framework is part of the Eclipse project [11] and has released its stable version in December 2020 [30]. Eclipse describes the Vorto Project as a ‘*Language for Digital Twins*’ [11]. It aims to be a device description language that can be used for creating IoT information models for various components. Apart from the main Vorto Information Model, there also exist a metamodel that shares similarities with FIWARE’s context information. The metamodel is used to describe the relationships and connection between the components [31]. However, the metamodel is more limited than FIWARE’s: It allows specifying data types and the structures, but it cannot convey data semantics like measurement units [32]. One of the main focus points of the Vorto Project is providing an open ecosystem that supports the interoperability of various IoT devices and vendors [33]. This is done by creating a domain specific device description language, namely vortolang [34], and by having uniform interfaces for communication [33]. The Vorto Projects also aims to be as technology-agnostic as possible. To help with a fast integration there exists a repository of device descriptions that are ready to use. What is more, there are code generators that can be used to accelerate the integration by generating platform specific artefacts. One major plus point of the Vorto device description language is, that it is very human-readable. Listing 2.1 is an example of a light dimmer. The code comes from the official Wiki [35]. The device description is intended to be self-explanatory and should be understandable by a non-programmer [33].

Listing 2.1: Example of a dimmer switch written in vortolang.

```
namespace vorto.examples.infomodels
version 1.0.0
using vorto.examples.functionblocks.Dimmer; 1.0.0
using vorto.examples.functionblocks.RGBColorPicker; 1.0.0
using vorto.examples.functionblocks.Switch; 1.0.0

infomodel Color_Lamp {
    displayname "Color_Lamp"
    description "Information model for a Color Lamp"
    category example

    functionblocks {
        dimmer as Dimmer
        colorpicker as RGBColorPicker
        switch as Switch
    }
}
```

Eclipse Vorto has its main focus on the IoT domain. It provides the tooling for creating, modifying and sharing device descriptions. This means that a lot of the documentation and design focus lies on IoT devices like sensors, single board computers and so forth. It is not really a universal modelling framework. Describing something that is not strictly in the IoT domain might be possible but is not actively encouraged. Compared to FIWARE the ability to add context information to the information model is more restricted. Although it is possible to export Vorto device descriptions to JSON-LD the other way around is in general not possible [34]. This is a major advantage for the FIWARE framework that natively supports this format. JSON-LD has the benefit of being a more universal modelling standard that is not bound to a specific domain.

In conclusion, the Eclipse Vorto framework could probably be used for the modelling of a Local Energy Community. Though the inclusion of very specific context information might not be possible, which would be less than ideal.

2.2.3 Microsoft Azure Digital Twins

This Microsoft developed framework Azure Digital Twins (ADT) is probably the most similar to FIWARE. It is a more general modelling platform that is not restricted to IoT alone [36]. The developer's documentation describes it as '*a digital representation of real-world things, places, business processes, and people*' [14]. The language that Azure Digital Twins uses is called Digital Twins Definition Language (DTDL) and is based on JSON-LD [37]. Therefore, ADT and NGSI-LD models are very comparable to another. One modelling feature that does not exist in NGSI-LD but is present in ADT are the differentiation between properties and telemetry. In ADT, the property type is used for data points that are non-volatile and should be stored, while the telemetry type is used for events and volatile data acquisition [38]. The same effect is achieved less explicitly in NGSI-LD by creating subscriptions.

In ADT there is also a big emphasis on data visualization. There are first-party tools for drawing the property graphs as well as a 3D visualizer that can represent real-world objects and entities [36]. This is partially possible to achieve with FIWARE, however there seems to be no out-of-the-box solution for representing data or entities in 3D.

The biggest difference between Azure Digital Twins and the FIWARE platform is the *openness* of the ecosystems. While every component in FIWARE's tool chain is open-source and self-hostable, ADT is explicitly a Platform as a Service (PaaS) and therefore tightly integrated in proprietary Microsoft Cloud Infrastructure. This might be beneficial if a comprehensive service contract is wanted, but prohibits the ability to control the software stack independently and will result in a more centralized and locked-in solution.

2.2.4 OpenEMS

OpenEMS stands for Open Source Energy Management System [39]. The main application of OpenEMS is a management platform for energy sources, especially of the renewable kind. It has a modular architecture and consists of three main components:

1. *OpenEMS Edge* handles the connection to the real-world devices via various communication protocols and has control and data acquisition duties.
2. *OpenEMS UI* is a web bases UI.
3. *OpenEMS Backend* is the connecting point of all *OpenEMS Edge* deployments.

Unlike FIWARE, OpenEMS it is not a general purpose modelling platform, however they do share a few similarities. Entities in OpenEMS are called *Components*, and they have a unique identifier [40]. *Components* are essentially digital twins of real-world IoT energy devices. What is more, *Components* can also be strictly virtual and be used for simulations. Simple data points are called channels and are as well uniquely addressable. The name comes from the fact, that these data points are directly used for internal communication. Some simple form of context information is possible: the data-type as well as a measurement unit can be specified. This is more restrictive compared to FIWARE's data model, since the context of the whole application is also more confined. If a new component needs to be added to the system, it has to implement the *Openems Component Interface* [40].

Even though OpenEMS is not a modelling framework per se, it does have similarities with other modelling procedures. Nevertheless, its primary function is the management and control of energy application. The possible usage of OpenEMS in Local Energy

Communities has been discussed in [41], where the author suggests its suitability in this environment.

2.3 Virtual Commissioning

Virtual commissioning is a very active topic in research as well as in industry applications. On the IEEE Xplore Platform alone, there are 175 papers[3] that have the keyword *virtual commissioning*. Many of these articles where published this year. Exact definitions as of what counts as virtual commissioning may vary depending on the domain. In the context of process automation and system validation, the author in [42] sums it up as virtualizing the code of a programmable logic controller (PLC) to control a simulated environment, that is as close as possible to the one, that the PLC will be deployed at. With the aid of this simulation the performance and conformity of the controller can be assessed.

This definition is not one to one compatible with the use case of a LEC, since no PLCs will be deployed. Moreover, the context is simply not the same. Nevertheless, many of the concepts of virtual commissioning can be used for the development of a LEC. The usage of virtual commissioning in other areas is something the author discusses in the paper mentioned [42]. Smart Grids are highlighted, however, it is explored from a manufacturing and production point of view.

One key feature, namely a simulated environment can be integrated in the LEC software stack. Instead of simulating a production plant and its machinery, this thesis will experiment with using a co-simulation framework to virtualize devices in the LEC. The concept of using simulation to assess LECs is not new. For instance in [43], a MATLAB model of a LEC is being developed to test various strategies concerning peer-to-peer energy trading inside the community. However, most research uses simulation as a mean to analyse control strategy and algorithms, like the paper above. They rarely focus on setting up simulation as a virtual commissioning platform.

2.3.1 Co-simulation in Smart Grids

In the broader context of smart grids, simulation, especially co-simulation, is a key component for validating their functionality. In [44], the authors explain, that due to the high system complexity of smart grids new methods for testing their compliance and reliability are needed. They conclude that co-simulation is a possible technology to tackle the challenge. Co-simulation is defined as a form of distributed simulation where multiple different simulation models are computed in parallel. The idea behind it is, that there exist a multitude of domain specific models that use various ways of representing simulation results. Hence, they are often incompatible or have limited usability in other simulation environments. A co-simulation framework or interface can provide a way to exchange data between models in a standardized fashion. This allows for using the *best*

[3]These numbers are as of the 7th Oktober 2022 and where obtained from `https://ieeexplore.ieee.org/`

model for the task without being constraint by the simulation framework and enable cross domain simulation approaches. The authors of paper [45] explore the possibilities of the Functional Mock-Up Interface [46] (FMI) for co-simulation in cyber-physical systems such as buildings and community energy systems. The following example is given that illustrates when co-simulation can shine and provide solutions for unsolved problems, '*[...] performance optimization studies for community energy systems need a system-level view that integrates domain-specific tools beyond the scope of typical building simulations.*'

One use-case of co-simulation that is arguably the most similar to virtual commissioning is the so-called *Hardware-In-the-Loop* or short HIL. This is a test procedure that uses co-simulation to calculate (real) input for a physical device. The behaviour of the devices is recorded and later analysed. With the help of co-simulation the hardware can be validated for scenarios that would either be too expensive or impossible to recreate within a standard test setting [44]. HIL uses the fact that the co-simulation framework controls the speed in which the time steps are calculated. Thus, real-time processing, that means a simulation that runs at wall-clock time, is possible.

2.3.2 Co-simulation with mosaik

For simulating smart grid application, like Local Energy Communities, the mosaik[4] framework seems to be a solid choice: mosaik is a co-simulator, that has been used extensively in research projects in this field since 2011 [47]. The program provides an interface to connect various (external) simulators. These simulators can each use different time-steps as long as they are discrete. mosaik will handle the schedule and provides various means to deal with algebraic loops. It is written in Python and supports other high-level languages such as Java or MATLAB, but can also be used with low-level languages. In this case however the manual integration is more involved. Even for smaller or simple simulation models it can be beneficial to have a clear and modular architecture. The framework itself does not come with any simulators. Nevertheless, the authors of the framework have developed some simulators that can be used out of the box.

The mosaik framework itself is not specifically designed to be used a virtual commissioning tool. For example, it does not have built-in support for IoT communication protocols. Nevertheless, the feasibility to extend the framwork with a basic MQTT implementation was shown in the bachelor thesis [48] written by Thomas Teodorowicz. This is important since many IoT devices use MQTT for communication and will the technology of choice for this bachelor thesis.

While formally the concept of virtual commissioning might not exist for LECs, this thesis will explore a more generalized definition.

[4]Note: mosaik should **not** be confused with *Eclipse Mosaic*, which is also a co-simulation framework, but was not used in this implementation.

CHAPTER 3

System Design

As discussed in the introduction, Local Energy Communities can be very heterogeneous. There is no one definition of what entities a LEC has to be comprised of. Nevertheless, there seems to be some common consensus of frequently used components. The following section will explain the use-case definition of the LEC utilised in this thesis. Then a suitable data model and system architecture is derived.

3.1 LEC Data Models

The habitants of the LEC are classified into three categories, with different properties concerning their power consumption characteristics.

- **Passive consumer** can only consume energy, their load profile is not adjustable and they do not produce any power.
- **Active consumer** can consume energy and produce none, but they have *smart* appliances that allow for adjusting their consumption on-demand.
- **Prosumer** is an active consumer that can produce energy.

In the following use-case, the energy production is accomplished through photovoltaic panels while load balancing is archived by controlling electrical warm water tanks, that act like heat buffer. This results into the following entity list:

- Person: Active/Passive Consumer
- Person: Prosumer
- Building

- Photovoltaic Panel and Measurement
- Warm Water Tank and Measurement
- (Smart) Power Meter

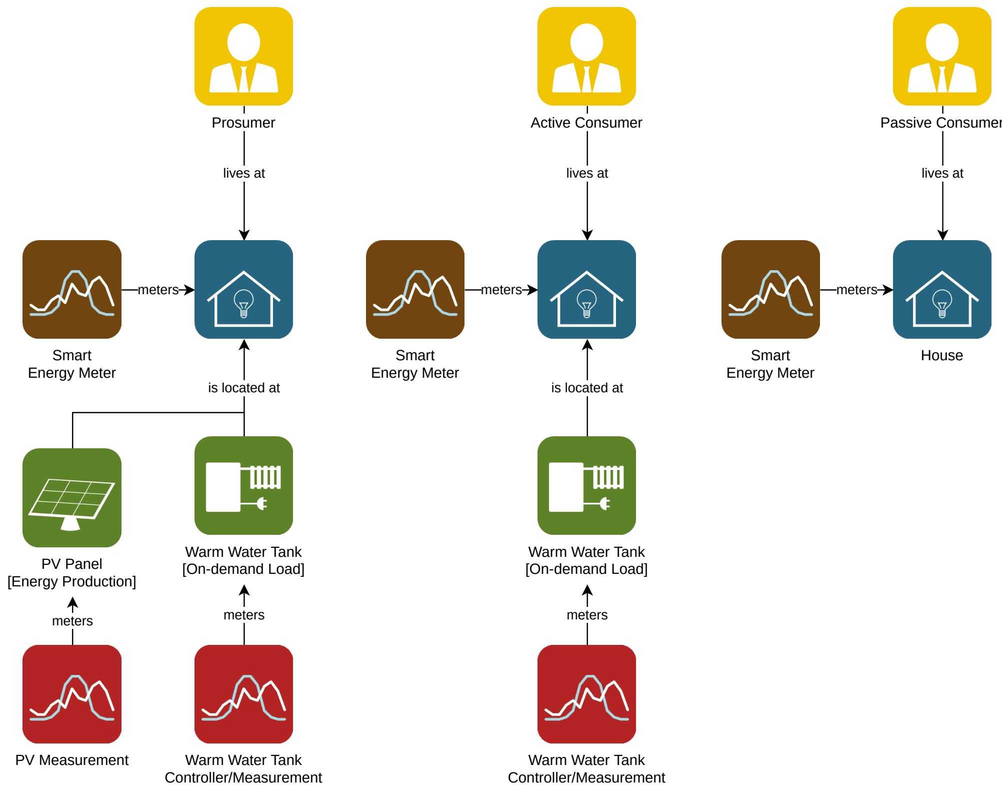


Figure 3.1: Overview of the entities in the LEC and their relationship.

The number of entities could easily be augmented. Notably an electrical energy storage system (e.g. a battery) is missing. The theoretical energy community was reduced on purpose to its essentials, since the focus lies on providing a suitable data model for an optimization algorithm. The presented concepts could be adopted for a more varied scenario.

The data models are created using NGSI-LD and are inspired by the Smart Data Models Program [49]. In particular the *PhotovoltaicDevice* [50] and *PhotovoltaicMeasurement* [51] were used as a base to model for the photovoltaic energy production. The *PhotovoltaicDevice* was extended by an *azimuth* and *surface tilt* degree. For modelling buildings and people, the standard FIWARE data models where used. A list of data models that are developed by FIWARE can be found here [52]. The model for the Energy Meter originated from a definition of the Indian Urban Data Exchange (IUDX) and can be found here [53]. Unfortunately there exist no Smart Data Model as reference for warm water tanks, thus this model was created from scratch. Including the one for the warm water tank measurements.

Smart Data Models is a joint effort between multiple projects (including FIWARE and IUDX) to provide high quality data models for various domains. The data models from Smart Data Models tend to be very detailed and allow encoding very specific context information. However, it is very easy to extend or reduce the properties of any given data model such that it meets the needs of one's application. Nevertheless, some properties are mandatory. This includes the *id* as well as *type* of the entity. The *id* has to be unique in system. Both the *id* and the property are interpreted as URI (Universal Resource Identifier). While it is not obligatory (from a technical point of view) to use a URI for the *id*, it is highly recommended. The URI of the type can resolve to a resource that further explains and documents the entity's type and its properties. However, it might also be a URN (Uniform Resource Name) in which case this is not guaranteed to happen.

Another key property that was discussed in previous the chapters is the *@context* property, which can link to a (remote) resource that documents the context information. For the proof-of-concept, the data models have been reduced to its minimum and non-essential entries have been removed.

Listing 3.1: Datamodel of an examplatory photovoltaic panel.

```
{
    "id": "urn:ngsi-ld:PhotovoltaicDevice:001",
    "type": "PhotovoltaicDevice",
    "name": "Photovoltaic Pannel",
    "description": "A residential photovoltaic panel",
    "MaximumSystemVoltage": {
        "type": "Property",
        "value": 400,
        "unitCode": "VLT"
    },
    "NominalPower": {
        "type": "Property",
        "value": 8000,
        "unitCode": "WTT"
    },
    "location": {
        "type": "Point",
        "coordinates": [
        16.470517,
        47.856821
        ]
    },
    "surfaceTilt": {
        "type": "Property",
        "value": 30,
        "unitCode": "DD"
    },
    "surfaceAzimuth": {
        "type": "Property",
        "value": 180,
        "unitCode": "DD"
    }
}
```

The Listing 3.1 shows the used data model for the photovoltaic panels. The *id* of the model follows closely the convention of URNs used by FIWARE. One can quickly grasp the most important property for an entity: its *type*. The *location* property sticks out from the other properties as it has *Point* as its type. It is a so-called *GeoProperty* and follows the *GeoJSON* format which standardizes location formats. The *unite codes* of the other properties follow the UN/CEFACT standard of common code, specifying measurement in a simple alphanumerical encoding. For instance, *DD* stands for degree, *VLT* for Volt, *WTT* for Watt and *KWT* for kilo Watt. An informative document listing many of these code can be found in [54].

The data models for static objects and measurement objects are separated. For example, the Listing 3.1 describes the static real-world properties of a photovoltaic panel, while Listing 3.2 shows how measurement of the photovoltaic panels are modelled. It is assumed, that the static properties rarely change. Measurements however, may update frequently. The reference data models from Smart Data Models often use two linked models to emphasis this fact. Therefore, two separate, but linked models were chosen as well.

Listing 3.2: Part of the data model for the photovoltaic measurement.

```
{
    "id": "urn:ngsi-ld:PhotovoltaicMeasurement:001",
    "type": "PhotovoltaicMeasurement",
    "dateObservedFrom": {
        "type": "Property",
        "value": ""
    },
    "dateObservedTo": {
        "type": "Property",
        "value": ""
    },
    "refPhotovoltaicDevice": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:PhotovoltaicDevice:001"
    },
    "activePower": {
        "type": "Property",
        "value": 5,
        "unitCode": "KWT"
    }
}
```

In Listing 3.2, part of the data model that was used for the measurements of a photovoltaic panel is displayed. Two properties are notable. First, the *refPhotovoltaicDevice* creates the link between measurement and the actual device. The type of this property is *Relationship* as it signals a connection between to data models. The link can also be observed in Figure 3.1. The value of the property is again a URN. At this point it becomes very clear, why the URN scheme used by FIWARE is useful. It allows to double-check that only a valid entity is linked. Second, the *dateObservedFrom* and *dateObservedTo* properties store an ISO date-time stamp, when the measurements took place.

It should be noted that the previous two Listings (Listing 3.1 and 3.2) were not in fact the full underlying data models but instances of data models. The very important *@context* tag instance was omitted for better readability. The data models that store all the explicit context information about the name space is stored separately as JSON-LD file. A snippet of this can be seen in Listing 3.3, where some core FIWARE terms are defined. The full file can be found here [55].

Listing 3.3: Part of the NGSI core context (name space).

```
{
    "@context": {
        "ngsi-ld": "https://uri.etsi.org/ngsi-ld/",
        "id": "@id",
        "type": "@type",
        "value": "https://uri.etsi.org/ngsi-ld/hasValue",
        "object": {
            "@id": "https://uri.etsi.org/ngsi-ld/hasObject",
            "@type":"@id"
        },
        "Property": "https://uri.etsi.org/ngsi-ld/Property",
        "Relationship": "https://uri.etsi.org/ngsi-ld/Relationship",
        "DateTime": "https://uri.etsi.org/ngsi-ld/DateTime",
        "Date": "https://uri.etsi.org/ngsi-ld/Date",
        "Time": "https://uri.etsi.org/ngsi-ld/Time",
        "observedAt": {
            "@id": "https://uri.etsi.org/ngsi-ld/observedAt",
            "@type": "DateTime"
        }
}
```

This for example explains how the Context Broker knows and expects that the *observedAt* property is a *DateTime* object. This default name space is implicitly given and cannot be redefined [56]. The *@context* file for the described data models can be found in the Appendix. Various on-line ontology services were used as reference. This includes DBPedia [57] which creates a general purpose knowledge graph out of *Wikimedia* resources. For some terms, however, there were no fitting definitions available. For instance, the term *capacity* links to a DBPedia's entry about volume and *thermalLoss* references its definition of thermal transfer. Also, *maxTemperatureDifference* links to `schema.org`'s *maxValue* definition. These name space references are a bit too general for our use case. In a real-world deployment, it would be better to self-host them.

Even though all the mentioned data models above are using the NGSI-LD specification, in order to configure the IoT Agent NGSI-v2 models are needed. The IoT Agent is used as management instance for the IoT devices and its purpose will be described in detail the next section. The Listing 3.4 shows the data model for the Smart Energy Meter. Two things should be noted about the entry. First, the name of the device (*device_name*) uses the NGSI-LD identifiers. This is done deliberately as the *device_name* will become the *id* once the entity and its data model get converted to an NGSI-LD instance by the Context Broker. The *entitiy_type* will be mapped to the as NGSI-LD *type*. *device_id* on the other hand is only used by the IoT agent and will not be transferred into NGSI-LD. The NGSI-LD representation of the Smart Energy Meter can be found in Listing 3.5. Second, since there exist no *@context* tag in NGSI-v2 there need to be a data model available to the Context Broker that defines the name space and meaning of the used attributes. This could be done in form of a separate JSON-LD file, like in Listing 3.3.

Listing 3.4: Part of the NGSI-v2 compliant data model of the Smart Energy Meter device.

```
{
    "device_id": "em001",
    "entity_name": "urn:ngsi-ld:EnergyMeter:001",
    "entity_type": "EnergyMeter",
    "timezone": "Europe/Berlin",
    "static_attributes": [
    {
        "name": "name",
        "type": "Property",
        "value": "(S)MS 1"
    },
    {
        "name": "description",
        "type": "Property",
        "value": "Smart Energy Meter data"
    }
    ]
}
```

Listing 3.5: The converted Model of the Smart Energy Meter.

```
{
    "id": "urn:ngsi-ld:EnergyMeter:003",
    "type": "EnergyMeter",
    "name": {
        "value": "(S)MS 3",
        "type": "Property",
        "observedAt": "2022-09-15T23:30:00.000Z"
    },
    "description": {
        "value": "Smart Energy Meter data",
        "type": "Property",
        "observedAt": "2022-09-15T23:30:00.000Z"
    },
    "supportedProtocol": {
        "value": "ul20",
        "type": "Property",
        "observedAt": "2022-09-15T23:30:00.000Z"
    },
    "dateObservedFrom": {
        "value": "2022-09-15T23:30:00",
        "type": "Property",
        "observedAt": "2022-09-15T23:30:00.000Z"
    },
    "dateObservedTo": {
        "value": "2022-09-15T23:35:00",
        "type": "Property",
        "observedAt": "2022-09-15T23:30:00.000Z"
    },
    "energyConsumed": {
        "value": 0.015076,
        "type": "Property",
        "unitCode": "KWH",
        "observedAt": "2022-09-15T23:30:00.000Z"
    }
}
```

The data models that were used in the proof-of-concept part of this thesis can be found in the Appendix. It should be noted that the building and people models where not used in for the proof-of-concept and therefore where omitted.

3.2 Smart LEC Architecture

The architecture of the theoretical energy community follows closely the *three-tier architecture pattern* of The Industrial Internet of Things Reference Architecture published by the Industrial Internet Consortium [58]. As the name suggests, it is composed of three different tiers:

1. **Edge Tier** acquires all relevant data points.

2. **Platform Tier** collects and processes the data edge nodes. It communicates with the Edge Tier through gateways.

3. **Enterprise Tier** domain-specific applications and end-user interfaces.

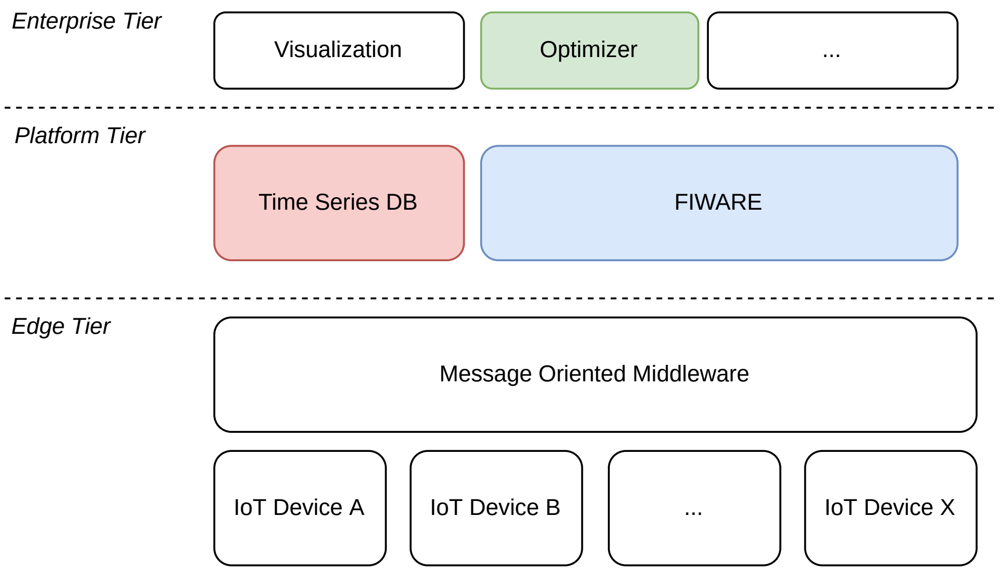


Figure 3.2: Overview of the proposed architecture of a prototypical smart LEC.

All three tiers have different duties and provide special services. The separation into these three tiers is also very suitable for LECs. The Edge Tier connects all edge devices. In a LEC for intance, these are smart measurement devices like Smart Energy Meters or the EMS of a photovoltaic system. The *Message Oriented Middleware* (MOM) is a suitable solution to connect many of these low-power IoT devices to a central point (hub). They usually use lightweight protocols and use principles like *Publish and Subscribe* or simple *Message Passing*. The MOM approach also comes with security benefits as only one communication channel to device is open, which is initiated by the device. Some also allow for more advanced authorization methods that further ensure the network security. Figure 3.2 depicts the architectural overview of a prototypical LEC using FIWARE's framework within the main *Platform Tier*. A special kind of database is used as main storage for historic data, called *time series-database*. They are specifically designed and optimized for data entries with time stamps. This includes data points like energy measurements. Therefore, they are a good choice for our application. What is more, they usually show very good performance for time range queries. For that reason, the data visualization directly communicates with the time series database. In the *Enterprise Tier* a variety of different applications can be found. They are often very domain specific or centre around data visualization or monitoring. In our case, we have two applications: One for graphing historic measurements and trends and the other is for energy flow optimization.

A more detailed view of the system design can be gained from Figure 3.3. The *MQTT Broker* acts as a gateway between the *Edge Tier* and the *Platform Tier*. The *IoT Agent* might be considered to be part of the *Edge Tier*, since it can also be used as a gateway to the *IoT devices*. In this implementation, it does not communicate with the devices directly

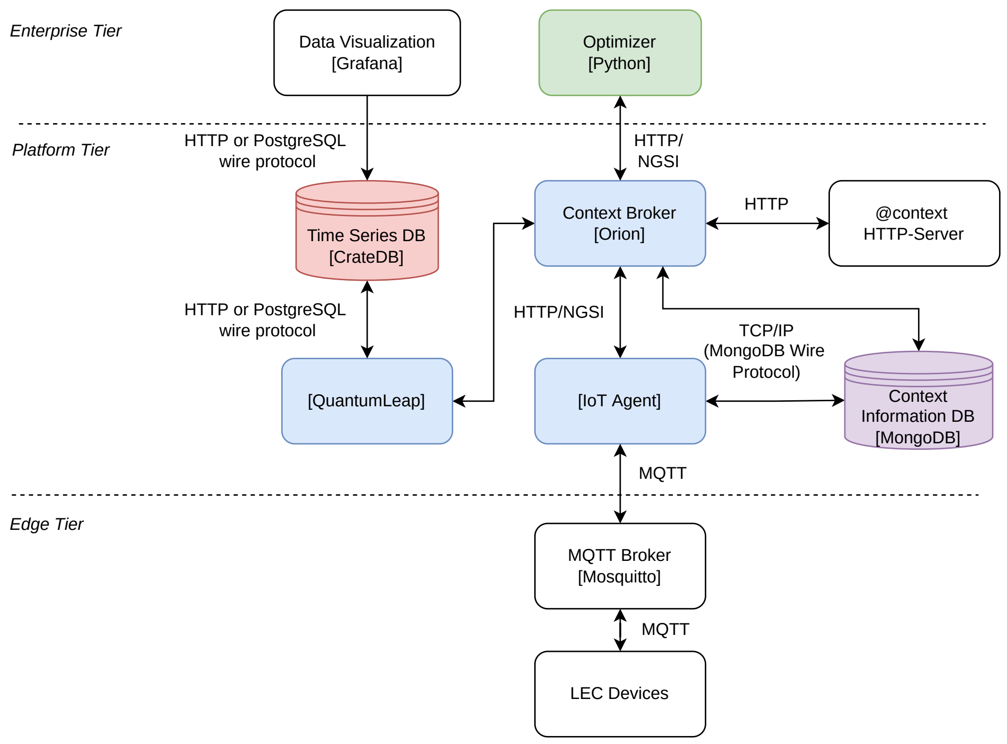


Figure 3.3: Detailed view of the production Architecture of a prototypical LEC.

and shares resources with the *Context Broker*. The *IoT Agent* is also used to configure and commission service groups, a task that is typical for this tier. It is theoretically also possible to communicate directly with the *IoT Agent* from the *Enterprise Tier*. Therefore, the component was considered to be part of the *Platform Tier*. The *Context Broker* is the main interface for the domain-specific applications. It can provide the current state and meta information of all registered entities. What is more, it connects external context providers and can relay commands from the *Enterprise Tier* to the *Edge Tier*. This is typical service for the *Platform Tier*. Yet the *Context Broker* does not store any historic data about the entities. For this functionality, it relies on the *QuantumLeap* component which enables the storage of time-series data in an appropriate database. It also provides a simple interface to query historic information and could also be used by the *Enterprise Tier*. The developers form *QuantumLeap* recommend *CrateDB* [59] for storage of the time-series data. It can be used as a standalone component and may be queried from the *Enterprise Tier*. However, a more opaque approach, where queries are only addressed to *QuantumLeap* is possible as well. It might even be useful to use both independently. The data visualization tool *Grafana* [60] for instance has an out-of-the-box integration for *CrateDB* thus it makes more sense to use this built-in support than writing a custom one. On the other hand, if the *Optimizer* wants to use historic data for its computations it might be more intuitively to use the *QuantumLeap* API as it is very similar to the *Context Broker*. The *HTTP-Server* as well as the *MongoDB* [61] database are auxiliary components for the *Context Broker*. The *HTTP-Server* hosts the *@context* files and *MongoDB* holds the current state and context information. While it is theoretically possible to query *MongoDB* manually, it defeats the purpose of the *Context Broker*. For the same reason, new database entries should only be added through the appropriate platform components (in blue).

In the Figure 3.3, the connecting arrows reveal the various protocols used for information exchange. Note, NGSI is not a protocol per se, but an interface that is provided by the Context Broker to exchange linked data models, in particular in the NGSI-LD format or as plain JSON.

3.3 Enabling Virtual Commissioning

Typically, in the design and prototyping phase not all details of the system have been finalized. For example not all IoT components might be deployed or even purchased. Some might not even be decided to be included. Using simulation and virtual commissioning can bring benefits to this project planning phase. Development of the Enterprise Tier applications can begin, before any device is deployed. This also includes the development of control strategies. There does not exist a universal strategy that always yields the best result. Especially in Smart Grid application, there is a lot of research going on, to improve current schemes. Having a digital twin of a LEC can help to test and derive novel solutions. What is more, it facilitates testing the scalability and flexibility of the optimizing algorithm. Making changes to the size and structure of the LEC in a virtual environment is easier and faster than in the real-world. From Figure 3.4, it becomes apparent that the structure of the architecture has not been changed drastically. Notably the Enterprise and Platform Tier have been completely untouched. This enables the aforementioned benefit, where significant parts of critical infrastructure can be directly deployed to production. Especially the *Optimizer* can be developed independently of the other components, as long as suitably detailed data models are chosen. The specific nomenclature does not need to be decided yet, since a simple translation between terms is always possible. Moreover, the *Optimizer* can rely on the *Context Broker* to provide adequate context information.

The individual simulators are managed by mosaik. They communicate using sockets. The simulators represent the IoT devices (*LEC devices*) as seen in Figure 3.3. The *MQTT-Connector* component is a custom mosaik simulator that enables the virtual LEC devices to communicate to the *IoT Agent* via MQTT. It relays simulation results to the *IoT Agent* and requests from the *Optimizer* through the *IoT Agent* to the *Simulators*.

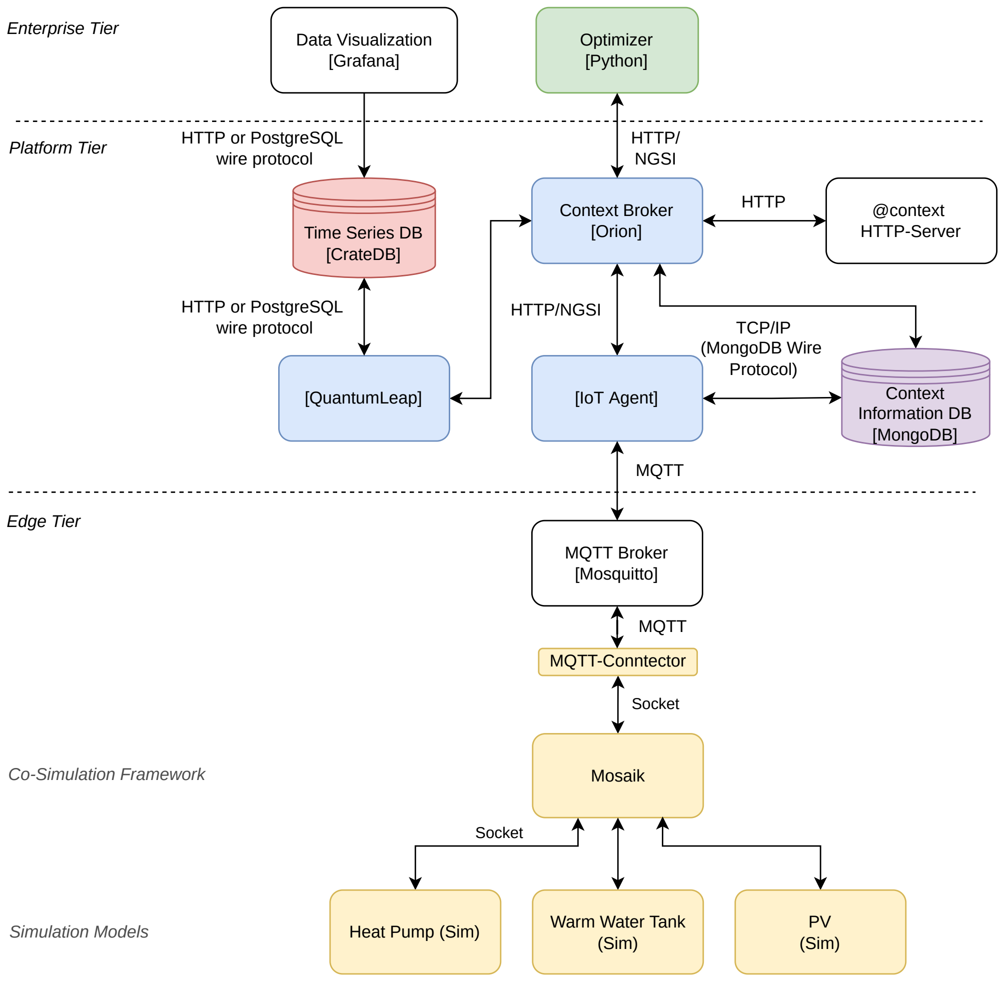


Figure 3.4: View of the system architecture when using mosaik as simulation host.

CHAPTER 4

Proof-of-concept Implementation

This chapter shows the viability of two concepts: *concept I*, with the help of mosaik as co-simulation framework a sort of virtual commissioning can take place which allows using a structural identical architecture to the real-world system. *Concept II*, an Optimizer can use the context information of data models provided by FIWARE's Context Broker to optimize the energy flow of the LEC.

4.1 Scenario Definition

To show that the concepts are possible, the following two scenarios will be deployed. An established LEC in Austria consisting of three *active consumers* and two photovoltaic power installations (*PV Panel*) will be simulated for a period of 24 hours. In Figure 4.1, the scenario with all its entities is depicted. All *households* are equipped with *warm water tanks* that can monitor their energy consumption. It should be noted that the Smart Energy Meters (SEM) meter the energy consumption of the households without the warm water tanks. The energy consumption of the warm water tanks is measured separately. This is important as the first scenario only considers the energy consumption of the warm water tanks. As simulation date mid-September was chosen, because at this time of year photovoltaic energy production is still viable while warm water usage might start to increase. At first the simulation will be computed with the Optimizer being disabled. Next, the amount of imported energy will be computed, that is all the energy that was not being used from the photovoltaic production. Then, without changing the architecture of the system a new entity will be added. SEMs will be introduced into the *Edge Tier* of the LEC. Contrary to real SEMs, they will report the electrical energy consumption of the three households periodically in real-time. This addition shall show that virtual commissioning environment can easily be augmented and made more realistic.

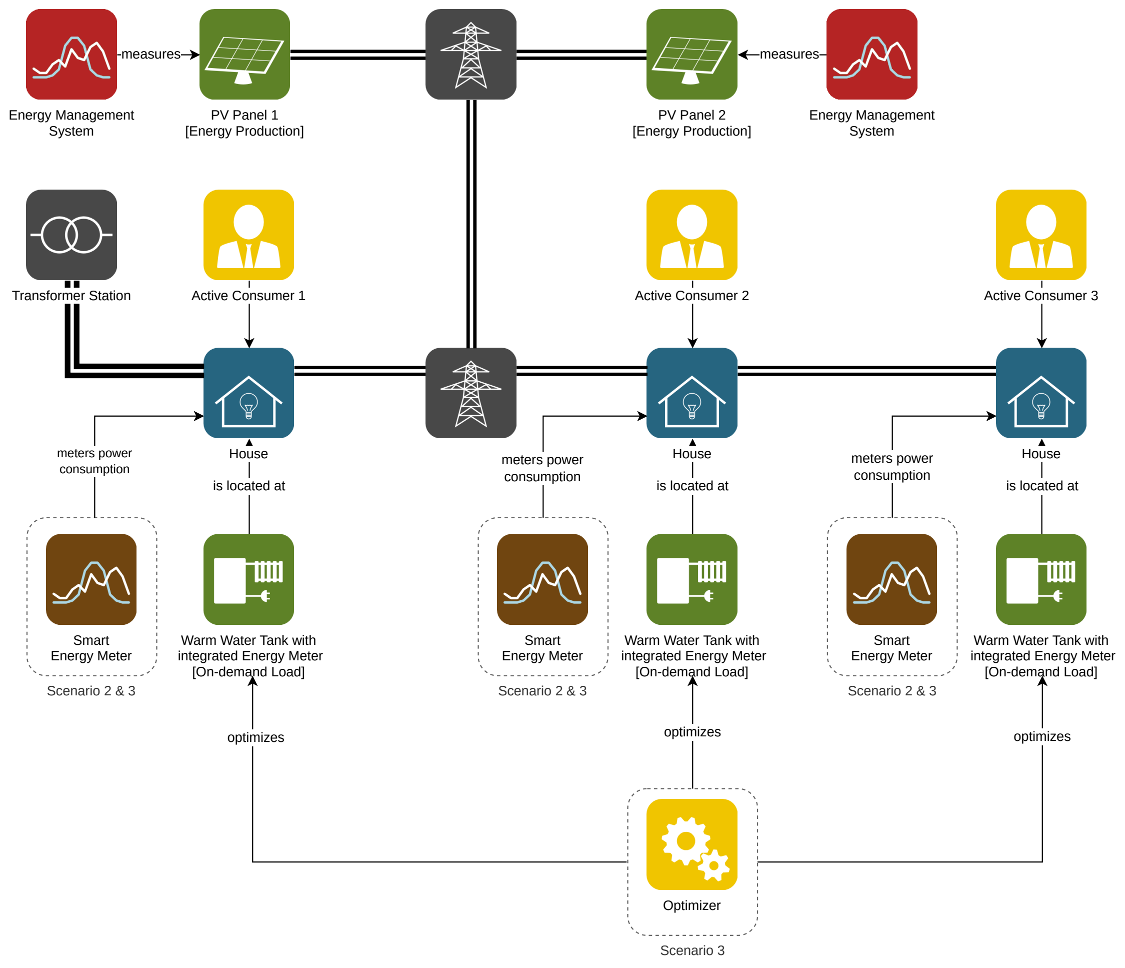


Figure 4.1: Graphical overview of the described scenarios.

Finally, in order to show *concept II*, the same scenario with the Optimizer enabled will be repeated. The results shall show that the implementation of an Optimizer is feasible within the proposed system design and can lead to less energy imported from the grid.

To further show the benefits of the proposed system design, the Enterprise Tier will be augmented by a data visualization application. Again, this addition will cause no change in the overall architecture. As data visualization tool Grafana was chosen, as it integrates well into the system and offers a multitude of configuration options. It could be used as monitoring tool or as dashboard for the members of the LEC.

4.2 Deployment Steps

Even though, the software stack is quite complex, the deployment can be done very systematically. With the help of a *docker-compose* [62] file, all third-party applications can be managed with simple commands. For easier management, it is feasible to have the Optimizer algorithm also running in a container. Importing the data models into the FIWARE framework is a bit more tedious, since there are multiple ways of tackling the problem. This, however, only needs to be done once.

1. Starting the containers: There are several applications that are run in separate containers. Their relationship can be seen in Figure 3.3.
    a) Enterprise Tier: Grafana, Optimizer
    b) Platform Tier: CrateDB, QuantumLeap, Orion, IoT Agent, MongoDB and HTTP-Server
    c) Edge Tier: Eclipse Mosquitto (MQTT Broker)
2. Commissioning the data-models: with the base structure started and running, the following steps are required to commission a device.
    a) Create the entities at the Context Broker
    b) Create a service group at the IoT-Agent
    c) Provision a sensor/device
    d) Create subscriptions for the Optimizer and QuantumLeap (the time series database)
3. Co-Simulation: Now that all data models are fed into the system and are known by the Context Broker as well as the IoT Agent, the simulation can be started. moasik, the co-simulation tool could be run as container or locally.

When creating a service group at the IoT-Agent, a new API-key needs to be chosen. The service group also allows specifying certain attributes that are common upon all devices that are part of the service group. What is more, it determines the communication protocol (for instance Ultralight 2.0 [63]) and sets the origin of the Context Broker.

4.3 Optimizer

Optimizing the energy flow of a grid is a non-trivial control task. There are many approaches one may take. For the purpose of this thesis, a simple rule-based strategy was chosen. This yields a compact program, but may not produce the best results. Since the focus of this work does not lie on finding the best control strategy, it is sufficient to show that an optimizing algorithm can reduce the overall amount of imported energy. Its rules are very simple and they use the following data points:

- Current (average) temperature of the Warm Water Tank (WWT)
- Target temperature of the WWT

- Maximum allowed difference between the target and actual temperature
- Available output power of the PV Panels.
- Current measured power consumption from the SEMs

First the algorithm checks whether there is any additional power available. Then it calculates a ratio which WWT could absorb the most energy without violating any temperature limits. Finally, it sends out a recommendation to the controller of the WWTs to do additional heating. The controller can decide whether or not to take the recommendation. It is assumed the controller of WWT uses a simple hysteresis heating strategy and will accept optimization suggestion as long as they do not violate the device's capability (i.e. using a higher input power than the devices supports).

The communication between the Optimizer (Enterprise Tier) and the other Tiers is straight forward. It can be seen in Figure 4.2.

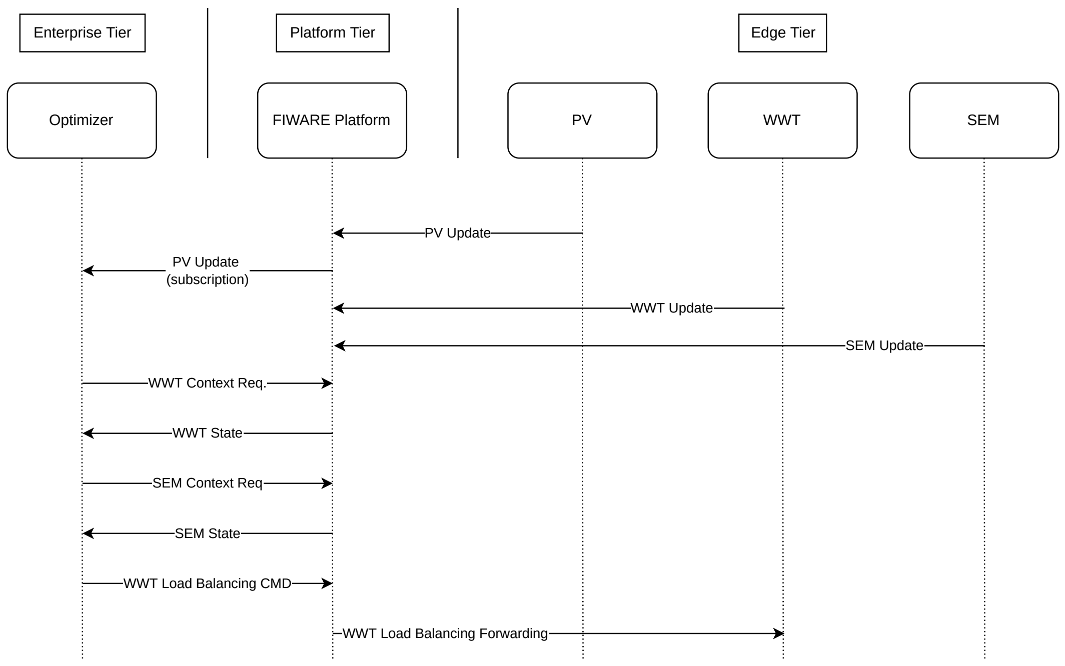


Figure 4.2: Flow Diagram Optimizer.

On the Edge Tier, there are three types of devices that regularly send their updates: *Photovoltaic Panels (PV)*, the *Warm Water Tanks (WWT)* and the *Smart Energy Meters (SEM)*. The co-simulation framework uses time intervals of five minutes. This means that the Photovoltaic Devices publishes its cell temperature and power production twelve times per hour over MQTT. The IoT-Agent listens to specific device topic and informs the Context Broker about any state changes. Due to an active subscription, if any context information about the PV panels updates, the Context Broker sends a request to the

Nominal Power	8 kWp
Surface Tilt	30°
Surface Azimuth	180° (South)
Location	Burgenland, near Eisenstadt
Date	15.9.2022

Table 4.1: Configuration of Photovoltaic Panel.

Optimizer. Then, the Optimizer checks the current state of the Warm Water Tanks by sending a request to the Context Broker. With all the needed information on hand, it can send a Load Balancing Command/Recommendation which gets relayed to the Controller of the Warm Water Tanks. In order for this communication to work reliably, the time intervals of status updates should be reasonably small. Otherwise the Load Balancing Command/Recommendation might become invalid until the controller considers it.

4.4 Simulation Models

Simulating Photovoltaic Panels can be very difficult since the output is dependent on many variables. Therefore, an existing (Python) library was chosen in order to get more realistic results. pvlib [64] seemed to be a good choice since it is easy to use and configure. Like mosaik, it is written in Python thus the integration into the co-simulation framework was trivial. It is used for all photovoltaic performance estimations. At the start of the simulation, when the photovoltaic simulator module is initialized, it requests the context information about the photovoltaic panels from the Context Broker. As the simulator initially does not know anything about the devices it requests all devices of type '*PhotovoltaicDevice*'. The key information needed for the simulation is:

- Location of the panel (longitude and latitude)
- Surface tilt and azimuth
- The nominal power rating of the panel
- Date

The last point does not come from the Context Broker, but is configured through the co-simulator. The output performance curve for the photovoltaic panel with the configuration like shown in Listing 3.1 can be seen in Figure 4.3. The properties of the photovoltaic panel are summarized in Table 4.1.

The simulation of the Warm Water Tanks uses a very simple model. While there exist more accurate and realistic models, for simplicity's sake, a less complex was created. It simply tracks the mean temperature of the Warm Water Tank and assumes a constant

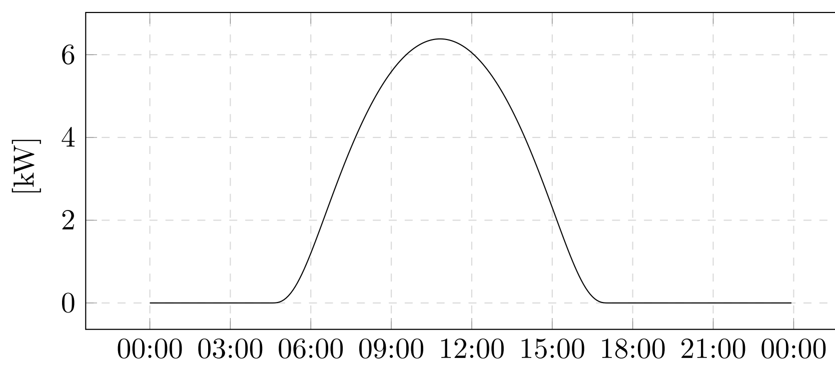


Figure 4.3: Output Power of the PV Panel.

linear thermal loss. The simulator has two inputs: the water outlet volume and the heat input in Watts.

The following properties can be set as initial values:

- Total water volume
- Target temperature
- Maximum temperature difference between the actual and the target temperature.
- Water inlet temperature

All properties apart from the last point are configured through the data model that is stored in the Context Broker. Equivalently to the Photovoltaic Simulator, the co-simulator requests all the context information about the warm water tanks and creates the models, respectively.

The component that connects between the IoT-Agent and mosaik, is written as two (Python) simulators. One simulator allows receiving MQTT topics (‘*mqtt-to-mosaik*’) from the IoT-Agent while the other publishes specific MQTT topics (‘*mosaik-to-mqtt*’). Both components need the API key that was configured, when setting up the service group at the IoT Agent. The ‘*mqtt-to-mosaik*’ component initially also receives a dictionary of device IDs and attributes. Then it listens continuously to the specified MQTT topics and collects the commands for theses virtual devices. At each simulation step it passes the collected data onto the connected device. This shows once more the power of co-simulation: As each simulator is a stand-alone program, it is very easy to integrate existing libraries into the framework. There are also very few limitations on additional communication channels that a simulator might use.

4.5 House Hold Simulation

In order to get more realistic load profiles for the warm water usage and electrical energy consumption, the LoadProfileGenerator (LPG) [65] tool was used. It was chosen as it is a mature tool and has many configuration options. It was initially created by Noah Daniel Pflugradt for his PhD thesis [66]. For our scenario, three distinct households from the pre-sets of the program where chosen. The households should represent a variety of living constellations that could occur in a LEC. The identifier in parentheses is the code used in LPG.

- *(CHR16)* Couple (age 75 and 80)
- *(CHR27)* Two Parents with two Children (age 9 and 13). Both Parents are working
- *(CHR45)* Two Parents with one Teenage Child (age 16). One Parent is working

The results of LPG are available to CSV (Comma Separated Values) files. The warm water usage of the households is depicted in Figure 4.5.

4.6 Simulation Results

As stated in the beginning of the chapter, three distinct scenarios were simulated using the mosaik framework. A structural overview of the simulation scenarios can be seen on Figure 4.1. It should be noted that in first scenario only the energy consumption from the WWT is simulated. The overall power consumption from the households is simulated in scenario two and three.

- **Scenario 1**: no SEM and no optimization algorithm
- **Scenario 2:** with SEM and no optimization algorithm
- **Scenario 3**: with SEM and with opimization algorithm

For all three scenarios, the results of the photovoltaic simulation are the same. The configuration of the panels is summarized in Table 4.2. The results of the simulation are depicted in Figure 4.4. We can see that the power production starts around 06:00, peaks at 12:00 and finally stops at 16:00. The electrical energy production for the smaller panel is around 45 kWh and around 80 kWh kWh for the larger one. In total, an energy production of around 125 kWh is simulated. It should be kept in mind, that the simulation assumed ideal conditions.

	PV Panel 1	PV Panel 2
Nominal Power	8 kWp	14 kWp
Surface Tilt	30°	44°
Surface Azimuth	180° (South)	
Location	Burgenland, near Eisenstadt	
Weather	sunny and no clouds	

Table 4.2: Configuration of the PV Panels.

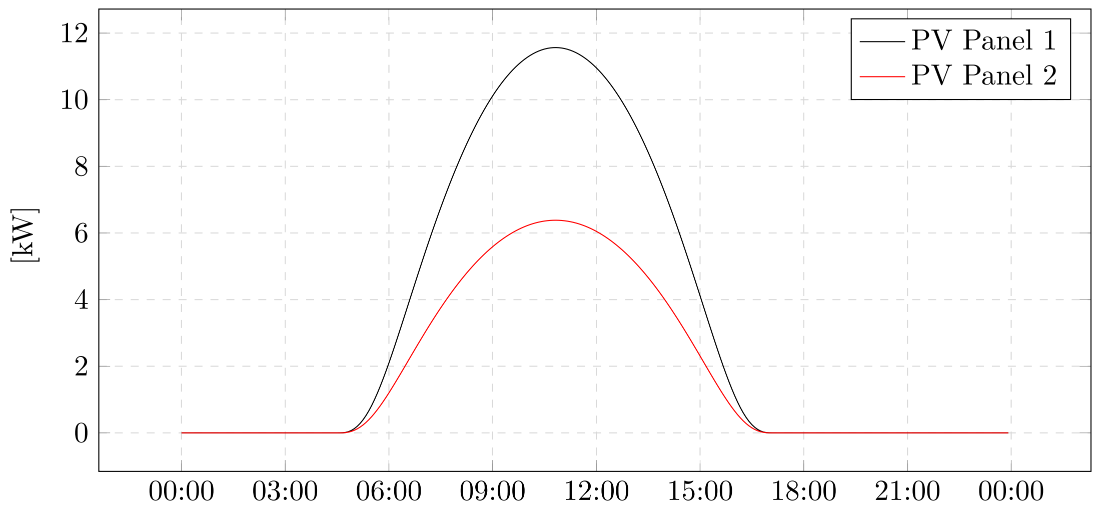
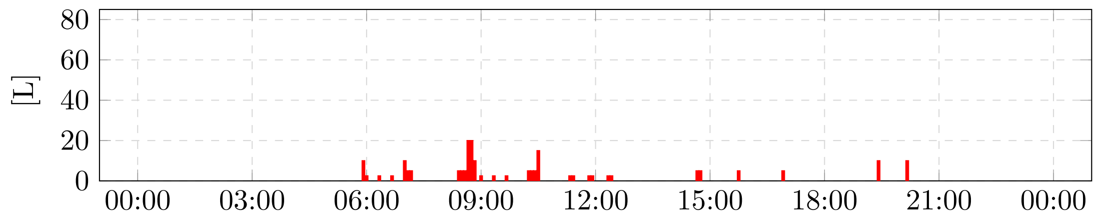
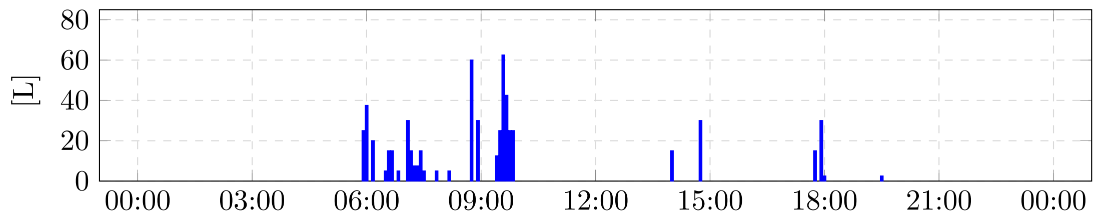


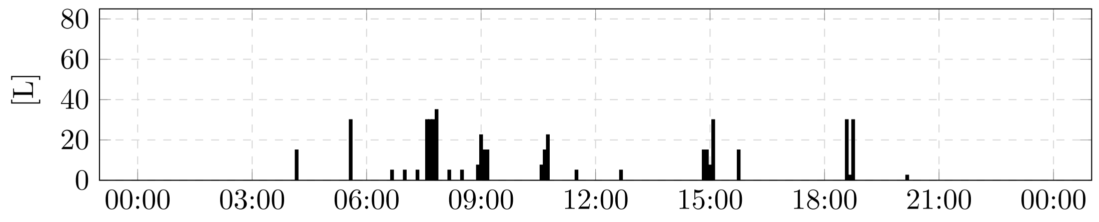
Figure 4.4: Output Power of the PV Panels.



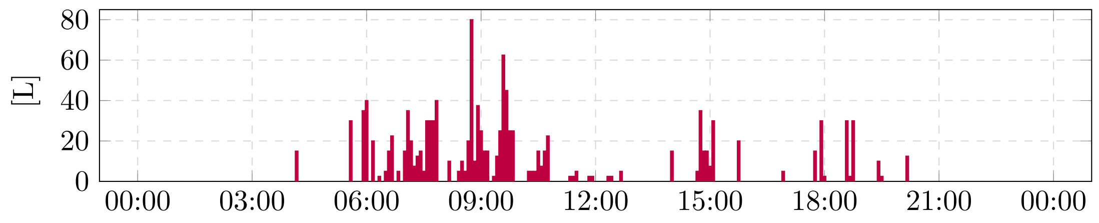
(a) Household 1 (CHR16).



(b) Household 2 (CHR27).



(c) Household 3 (CHR45).



(d) All Households.

Figure 4.5: Warm Water Consumption.

In the first simulation round, only the WWTs and the PV panels were simulated. The power consumption of the LEC can be seen in Figure 4.7. The power consumption is dependent on the warm water usage, thus the peak in power consumption at around 10:00 (Figure 4.7) can be explained by the significant warm water usage in the morning (Figure 4.5). Remarkably, the main power consumption coincides with the main power production phase and only around 11 kWh of the 64 kWh need to be imported from the grid (Figure 4.6 and 4.11).

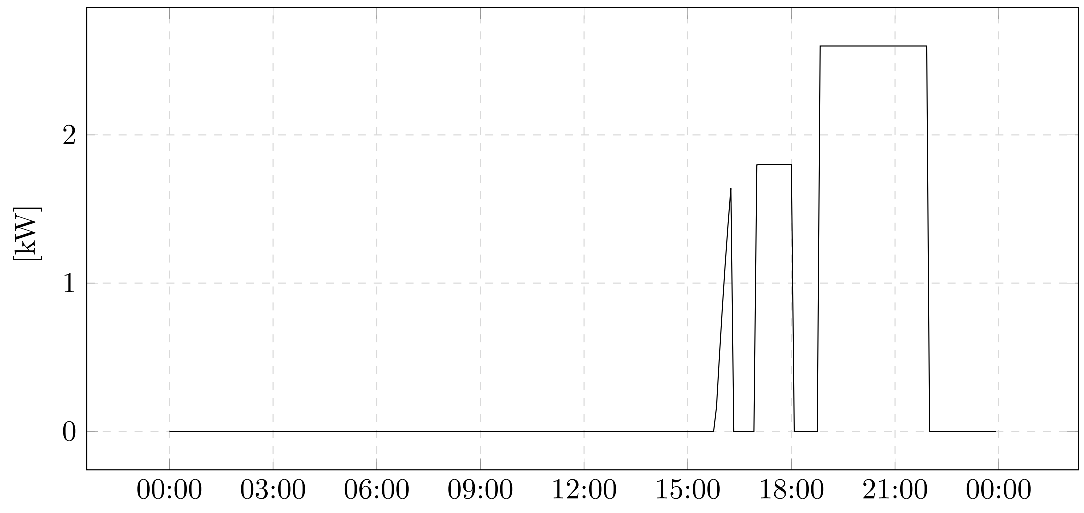


Figure 4.6: Power Import from Grid Scenario 1.

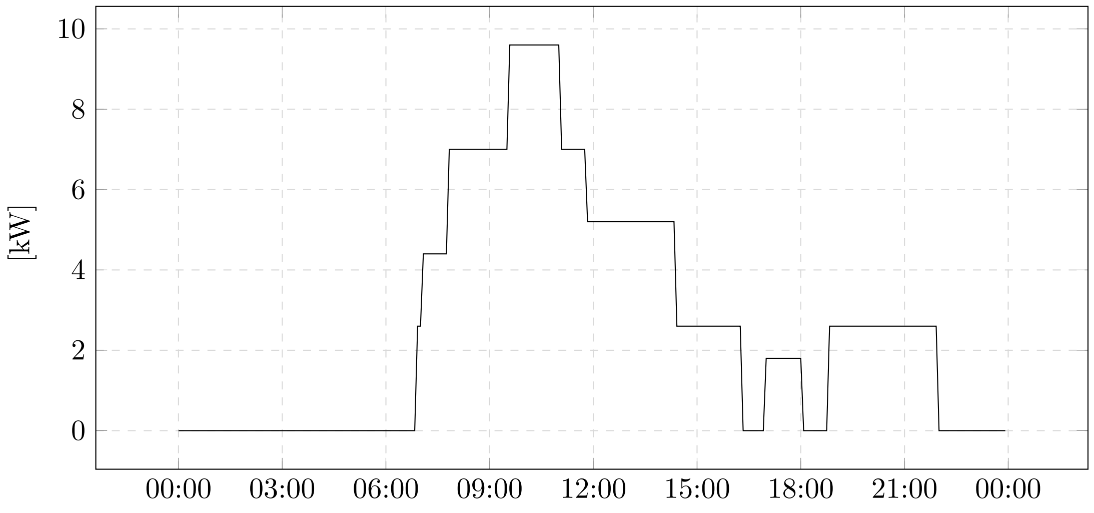


Figure 4.7: Power Consumption Scenario 1.

The second scenario added the SEMs. This was simply done by creating a new service group at the *IoT Agent* and provisioning the devices. In mosaik, a new simulator was added that reads the results form the LPG tool. With the SEMs enabled, it becomes apparent, why neglecting the general electrical energy consumption of households could lead to inaccurate results. With the new inclusion, the overall power consumption almost doubled form 64 kWh to 110 kWh. The discrepancy can be seen in Figure 4.9. Especially the increased power usage starting in the afternoon was not correctly represented in the first simulation. This can be seen in the cumulative power consumption in Figure 4.11. What is more, the amount of imported energy from the grid tripled from 11 kWh to 33 kWh compared to the first scenario. The elevated evening consumption plays a big factor in this significant growth as can be observed in Figure 4.9. Once again, during the main phase of the photovoltaic energy production only a small amount of power is imported from the grid (Figure 4.8).

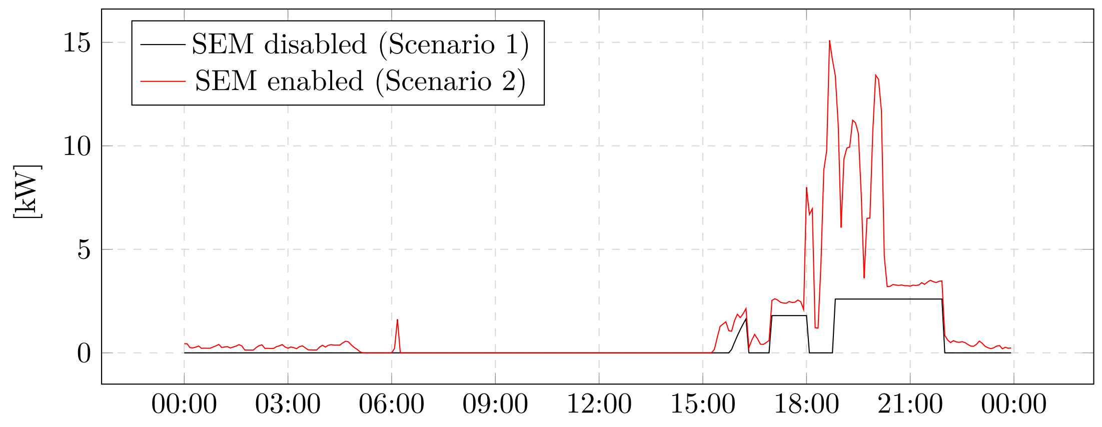


Figure 4.8: Power Import from Grid Scenario 1 vs. Scenario 2.

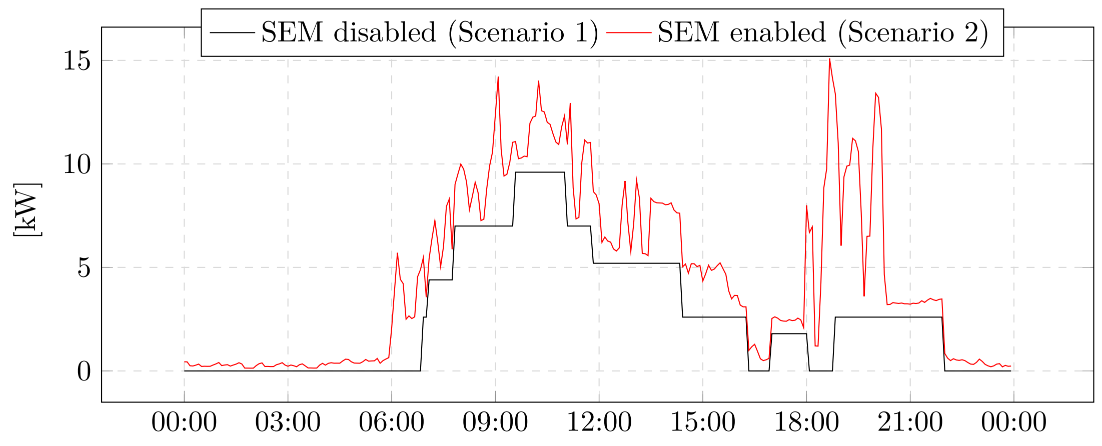


Figure 4.9: LEC Total Power Consumption Scenario 1 vs. Scenario 2.

In the third and last simulation experiment, the optimization algorithm was enabled. Recall, the idea of this simple Optimizer was to store the surplus energy from the PV in the WWTs of the three households. With the desired effect being needing less energy in the evening during the second phase of high warm water usage as the WWTs have a higher average water temperature. The Optimizer periodically (every five miutes) sends a message to the individual WWTs that tells them how much additional energy they are allowed to use. The surplus power from the photovoltaic panels peaks around midday, when the overall energy production is the highest. In this period, the Optimizer sends the majority of its recommendation requests to the controllers. In Figure 4.10, the sum of the individual recommendation requests is depicted. Figure 4.13 shows that during the same time, the overall energy consumption of the optimized LEC is higher than the unoptimized. This indicates, that the extra power is used and buffered into the WWTs. The effect becomes visible in Figure 4.12. While from 00:00 to around 20:00 the amount of imported energy is very similar, the optimized LEC performs significantly better after 20:00. Figure 4.11 concludes the results of the scenarios. Even though the total energy consumption is very similar (Optimized LEC 109 kWh versus 110 kWh unoptimized LEC) the imported power is measurably less with the Optimizer enabled. The unoptimized LEC imports 33 kWh and the optimized only 26 kWh. This is reduction of around 20 percent.

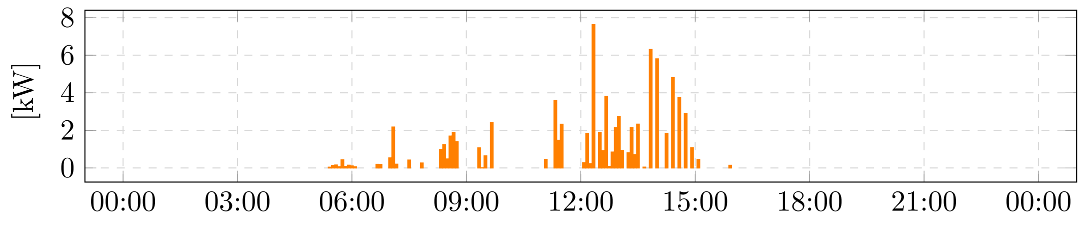


Figure 4.10: Optimization Recommendation from the Optimizer (Surplus Power from the Photovoltaic Panels).

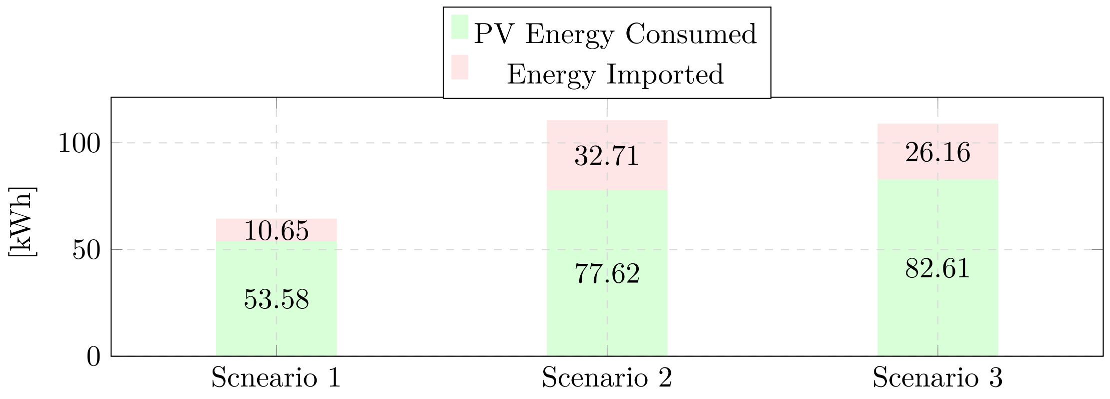


Figure 4.11: Power Statistics Comparison.

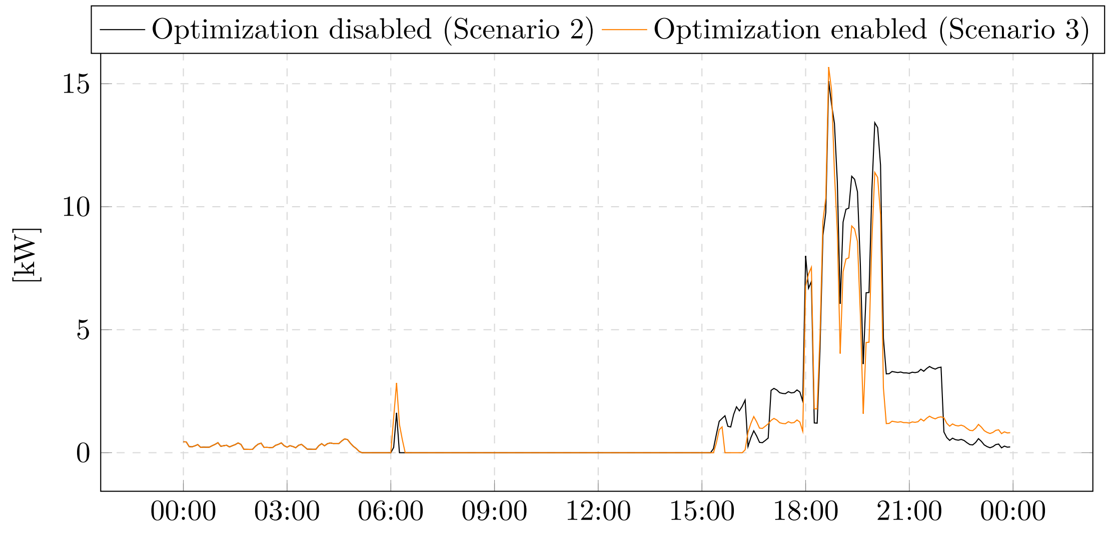


Figure 4.12: Power Import from Grid Scenario 2 vs. Scenario 3.

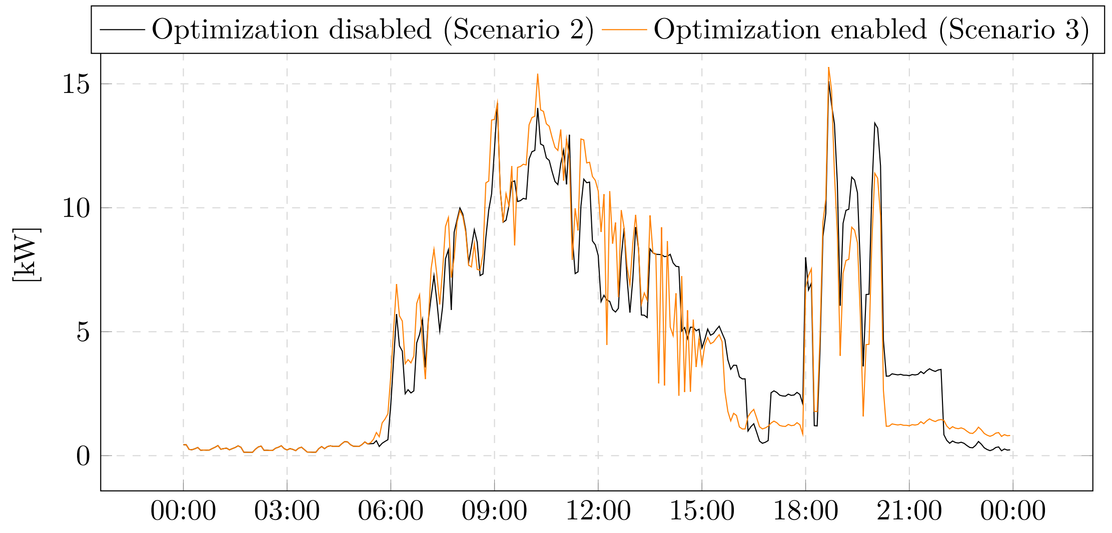


Figure 4.13: LEC Total Power Consumption Scenario 2 vs. Scenario 3.

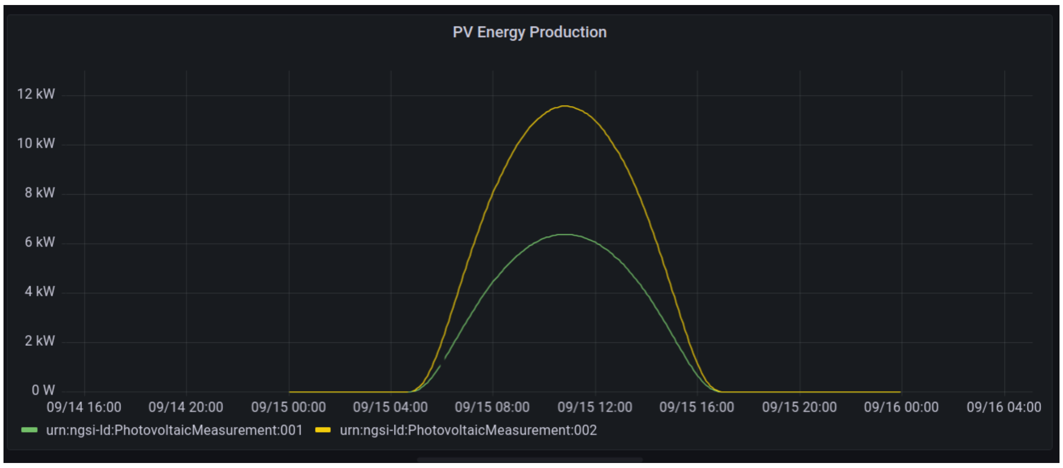


Figure 4.14: A dashboard in Grafan, showing information about the PV-Panels

4.7 Data Visualization: Grafana

As mentioned in the beginning of the chapter, adding a data visualization tool like Grafan brings many benefits. It allows monitoring the LEC and can also be used by less technically skilled people. There are many configuration options, nevertheless, the initial set-up does not take a long time. Fortunately, it is easy to integrate a tool like Grafana into our system design. Like the Optimizer, Grafana lives in the Enterprise Tier. For simplicity's sake, it communicates directly with the time-series database CrateDB. As CrateDB uses standard SQL, there exist an integration for Grafana out-of-the-box. Figure 4.14 shows what a simple dashboard might look like. It is a plot of the power production of the two photovoltaic panels. Listing 4.1 shows the corresponding SQL-Request.

Listing 4.1: Grafana SQL Request

```
SELECT
    time_index AS "time",
    entity_id AS metric,
    activepower
FROM doc.etphotovoltaicmeasurement
WHERE
    $__timeFilter(time_index)
ORDER BY 1,2
```

CHAPTER 5

Discussion and Future Work

The goal of this Bachelor thesis was to find a suitable architecture for smart LECs based on FIWARE which utilizes smart data models for optimizing energy flow control and support virtual commissioning. At first, the terminology surroundings ontologies was explored. It was concluded that there may exist standardized definition of what the difference between an information model and a data model is; however, different modelling frameworks use different definitions for the same terms and often expand them with new ones. Thus, it is not possible to compare two modelling frameworks by terminology that they use.

In the proof-of-concept part of this thesis, it was shown that the proposed architecture does indeed work for implementing an algorithm that optimizes the energy flow. In particular, smart data models supplied the optimization algorithm with all the required context information. However, there are some limitations to the presented results. First and for most, the conditions of the scenario were heavily idealized. In September, a reasonable high energy production for photovoltaic panels can be expected, but only if there is sunny weather and no overcast. Moreover, the simulation did not regard the cleanness of the individual panels. The International Energy Agency for Photovoltaic Powers Systems Program estimates that dirty panels can lead to energy production losses of up to 5 percent [67]. Besides that, the peak performance of the photovoltaic installation was chosen to be large enough, such that there is surplus energy for the Optimizer. Likewise, the efficiency of the Optimizer is highly debatable. Its strategy is very likely to be too simple for real-world use-cases and might even lead to worse results. This is due to its dependency on surplus energy in system. Nevertheless, the proposed architecture could be used to implement different more involved optimization strategies. So far the optimization strategy could only *suggest* to the warm water tank controller to put more energy into to tank. However, the opposite could also be done. The heating of the warm water tank could be temporarily disabled if a peak energy usage is detected or expected. Additionally, historic data and trends could be incorporated into the design of the optimization algorithm in order to predict future energy demand.

The second shortcoming of the simulation results is the accuracy of the individual simulators. In particular, the warm water tank simulator used arguably a too naive implementation. There exist better approaches and libraries to simulate the thermodynamic behaviour of insulated tanks. The current implementation neglects and oversimplifies many physicals characteristics.

The third and last shortcoming are the data models. They could be augmented by more details and context information. For instance, there are many more unused attributes in the reference data model for the photovoltaic devices. However, the larger problem is that the current *@context* model is suboptimal. For instance, it links to third party definitions that are too vague and general. This is the case for the *maxTemperatureDifference* attribute that has a very specific meaning for the warm water tank controller but is not a common term. The same is true for *thermalLoss* which describes the constant heat loss of the warm water tanks and is not a standard property. There are two possible strategies to fix this issue, either by submitting the missing definitions to a schema provider, or by self-hosting the resources. It might even be reasonable to crate a URN name space specifically for LECs (in Austria).

Finally, it should be kept in mind that the presented solution is very much a proof-of-concept implementation. The self-developed tools are not ready for a market-ready solution. While the third party tools from FIWARE, can be used in the real-world, the current configuration is not suitable. More care has to be given to security and proper authorization. This is supported by the FIWARE platform as well as the MQTT message broker. The configuration should also be updated to allow for better scaling. Currently, there is no *nice* or proper way to on-board a new entity/member of the LEC. These points should not be ignored in a real-world deployment. However, configuring for better security, scalability and a proper on-boarding process would not change the system design. For this reason they were deemed to be less important for this work.

To conclude, there exist quite a few points that could be improved and augmented in the system's implementation. The system design and architecture however, showed their potential. They would not need any significant changes in order to incorporate the mentioned improvements.

APPENDIX A

Data Models

A.1 Warm Water Tank

```
{
    "id": "urn:ngsi-ld:WarmWaterTank:001",
    "type": "WarmWaterTank",
    "name": "Smart Warm Water Tank 1",
    "description": "A smart warm water tank, that supports variable
       input power levels",
    "capacity": {
        "type": "Property",
        "value": 300.0,
        "unitCode": "LTR"
    },
    "NominalPower": {
        "type": "Property",
        "value": 3.6,
        "unitCode": "KWT"
    },
    "thermalLoss": {
        "type": "Property",
        "value": 50,
        "unitCode": "WTT"
    },
    "targetTemperature": {
        "type": "Property",
        "value": 60,
        "unitCode": "CEL"
    },
    "maxTemperatureDifference": {
        "type": "Property",
        "value": 5,
            "unitCode": "CEL"
        }
},
{
    "id": "urn:ngsi-ld:WarmWaterTank:002",
    "type": "WarmWaterTank",
    "name": "Smart Warm Water Tank 2",
    "description": "A smart warm water tank, that supports variable input power levels",
    "capacity": {
        "type": "Property",
        "value": 1200.0,
        "unitCode": "LTR"
    },
    "NominalPower": {
        "type": "Property",
        "value": 5.2,
        "unitCode": "KWT"
    },
    "thermalLoss": {
        "type": "Property",
        "value": 35,
        "unitCode": "WTT"
    },
    "targetTemperature": {
        "type": "Property",
        "value": 60,
        "unitCode": "CEL"
    },
    "maxTemperatureDifference": {
        "type": "Property",
        "value": 5,
        "unitCode": "CEL"
    }
},
{
    "id": "urn:ngsi-ld:WarmWaterTank:003",
    "type": "WarmWaterTank",
    "name": "Smart Warm Water Tank 3",
    "description": "A smart warm water tank, that supports variable input power levels",
    "capacity": {
        "type": "Property",
        "value": 1200.0,
        "unitCode": "LTR"
    },
    "NominalPower": {
        "type": "Property",
        "value": 5.2,
            "unitCode": "KWT"
        },
        "thermalLoss": {
            "type": "Property",
            "value": 35,
            "unitCode": "WTT"
        },
        "targetTemperature": {
            "type": "Property",
            "value": 60,
            "unitCode": "CEL"
        },
        "maxTemperatureDifference": {
            "type": "Property",
            "value": 5,
            "unitCode": "CEL"
        }
}
```

A.2 Warm Water Tank Measurement

```
{
    "devices": [
    {
        "device_id": "wwt001",
        "entity_name": "urn:ngsi-ld:WarmWaterTankMeasurement:001",
        "entity_type": "WarmWaterTankMeasurement",
        "protocol": "PDI-IoTA-MQTT-UltraLight",
        "transport": "MQTT",
        "commands": [
        {
            "name": "p",
            "type": "command"
        }
        ],
        "static_attributes": [
        {
            "name": "refWarmWaterTank",
            "type": "Relationship",
            "value": "urn:ngsi-ld:WarmWaterTank:001"
        },
        {
            "name": "name",
            "type": "Property",
            "value": "WWT Measurement"
        },
        {
            "name": "description",
                "type": "Property",
                "value": "Measurement data from the Warm Water Tank"
            }
            ]
        },
        {
            "device_id": "wwt002",
            "entity_name": "urn:ngsi-ld:WarmWaterTankMeasurement:002",
            "entity_type": "WarmWaterTankMeasurement",
            "protocol": "PDI-IoTA-MQTT-UltraLight",
            "transport": "MQTT",
            "commands": [
            {
                "name": "p",
                "type": "command"
            }
            ],
            "static_attributes": [
            {
                "name": "refWarmWaterTank",
                "type": "Relationship",
                "value": "urn:ngsi-ld:WarmWaterTank:002"
            },
            {
                "name": "name",
                "type": "Property",
                "value": "WWT Measurement"
            },
            {
                "name": "description",
                "type": "Property",
                "value": "Measurement data from the Warm Water Tank"
            }
            ]
        },
        {
            "device_id": "wwt003",
            "entity_name": "urn:ngsi-ld:WarmWaterTankMeasurement:003",
            "entity_type": "WarmWaterTankMeasurement",
            "protocol": "PDI-IoTA-MQTT-UltraLight",
            "transport": "MQTT",
            "commands": [
            {
                "name": "p",
                "type": "command"
            }
            ],
            "static_attributes": [
            {
```

```
            "name": "refWarmWaterTank",
            "type": "Relationship",
            "value": "urn:ngsi-ld:WarmWaterTank:003"
        },
        {
            "name": "name",
            "type": "Property",
            "value": "WWT Measurement"
        },
        {
            "name": "description",
            "type": "Property",
            "value": "Measurement data from the Warm Water Tank"
        }
        ]
    }
    ]
}
```

A.3 Photovoltaic Panel

```
{
    "id": "urn:ngsi-ld:PhotovoltaicDevice:001",
    "type": "PhotovoltaicDevice",
    "name": "Photovoltaic Pannel",
    "description": "A residential photovoltaic panel",
    "MaximumSystemVoltage": {
        "type": "Property",
        "value": 400,
        "unitCode": "VLT"
    },
    "NominalPower": {
        "type": "Property",
        "value": 8000,
        "unitCode": "WTT"
    },
    "location": {
        "type": "Point",
        "coordinates": [
        16.470517,
        47.856821
        ]
    },
    "surfaceTilt": {
        "type": "Property",
        "value": 30,
        "unitCode": "DD"
    },
    "surfaceAzimuth": {
        "type": "Property",
        "value": 180,
        "unitCode": "DD"
    }
},
{
    "id": "urn:ngsi-ld:PhotovoltaicDevice:002",
    "type": "PhotovoltaicDevice",
    "name": "Photovoltaic Pannel",
    "description": "A residential photovoltaic panel",
    "MaximumSystemVoltage": {
        "type": "Property",
        "value": 400,
        "unitCode": "VLT"
    },
    "NominalPower": {
        "type": "Property",
        "value": 14,
        "unitCode": "KWT"
    },
    "location": {
        "type": "Point",
        "coordinates": [
        16.480517,
        47.846821
        ]
    },
    "surfaceTilt": {
        "type": "Property",
        "value": 44,
        "unitCode": "DD"
    },
    "surfaceAzimuth": {
        "type": "Property",
        "value": 180,
        "unitCode": "DD"
    }
}
```

A.4 Photovoltaic Measurement

```
{
    "devices": [
    {
        "device_id": "ems001",
        "entity_name": "urn:ngsi-ld:PhotovoltaicMeasurement:001",
        "entity_type": "PhotovoltaicMeasurement",
        "timezone": "Europe/Berlin",
        "static_attributes": [
        {
            "name": "refPhotovoltaicDevice",
            "type": "Relationship",
            "value": "urn:ngsi-ld:PhotovoltaicDevice:001"
        },
        {
            "name": "name",
            "type": "Property",
            "value": "Photovoltaic station (EMS)"
        },
        {
            "name": "description",
            "type": "Property",
            "value": "Photovoltaic data"
        }
        ]
    },
    {
        "device_id": "ems002",
        "entity_name": "urn:ngsi-ld:PhotovoltaicMeasurement:002",
        "entity_type": "PhotovoltaicMeasurement",
        "timezone": "Europe/Berlin",
        "static_attributes": [
        {
            "name": "refPhotovoltaicDevice",
            "type": "Relationship",
            "value": "urn:ngsi-ld:PhotovoltaicDevice:001"
        },
        {
            "name": "name",
            "type": "Property",
            "value": "Photovoltaic station (EMS)"
        },
        {
            "name": "description",
            "type": "Property",
            "value": "Photovoltaic data"
        }
        ]
    }
    ]
}
```

A.5 Smart Energy Meters

```
{
    "devices": [
    {
        "device_id": "em001",
        "entity_name": "urn:ngsi-ld:EnergyMeter:001",
        "entity_type": "EnergyMeter",
        "timezone": "Europe/Berlin",
        "static_attributes": [
        {
            "name": "name",
            "type": "Property",
            "value": "(S)MS 1"
        },
        {
            "name": "description",
            "type": "Property",
            "value": "Smart Energy Meter data"
        }
        ]
    },
    {
        "device_id": "em002",
        "entity_name": "urn:ngsi-ld:EnergyMeter:002",
        "entity_type": "EnergyMeter",
        "timezone": "Europe/Berlin",
        "static_attributes": [
        {
            "name": "name",
            "type": "Property",
            "value": "(S)MS 2"
        },
        {
            "name": "description",
            "type": "Property",
            "value": "Smart Energy Meter data"
        }
        ]
    },
    {
        "device_id": "em003",
        "entity_name": "urn:ngsi-ld:EnergyMeter:003",
        "entity_type": "EnergyMeter",
        "timezone": "Europe/Berlin",
        "static_attributes": [
        {
            "name": "name",
            "type": "Property",
            "value": "(S)MS 3"
        },
        {
            "name": "description",
            "type": "Property",
            "value": "Smart Energy Meter data"
        }
        ]
    }
    ]
}
```

A.6 @context LEC

```
{
    "@context": {
        "fiware": "https://uri.fiware.org/ns/data-models#",
        "NominalPower": "https://smartdatamodels.org/dataModel.GreenEnergy/NominalPower",
        "activePower": "https://smartdatamodels.org/dataModel.GreenEnergy/activePower",
        "MaximumSystemVoltage": "https://smartdatamodels.org/dataModel.GreenEnergy/MaximumSystemVoltage",
        "temperature": "https://smartdatamodels.org/dataModel.GreenEnergy/temperature",
        "energyConsumed": "https://voc.iudx.org.in/energyConsumed",
        "owner": "https://smartdatamodels.org/dataModel.GreenEnergy/owner",
        "surfaceTilt": "https://brickschema.org/ontology/1.2/entity_properties/tilt",
        "surfaceAzimuth": "https://brickschema.org/ontology/1.2/entity_properties/azimuth",
        "capacity": "https://dbpedia.org/page/Volume",
        "thermalLoss": "https://dbpedia.org/page/Heat_transfer",
        "targetTemperature": "https://brickschema.org/ontology/1.2/classes/Water_Temperature_Setpoint",
        "maxTemperatureDifference": "https://schema.org/maxValue",
        "Person": "fiware:Person",
        "Building": "fiware:Building",
        "PhotovoltaicMeasurement": "https://github.com/smart-data-models/dataModel.GreenEnergy/tree/master/PhotovoltaicMeasurement",
        "PhotovoltaicDevice": "https://github.com/smart-data-models/
           dataModel.GreenEnergy/tree/master/PhotovoltaicDevice",
        "EnergyMeter": "https://voc.iudx.org.in/EnergyMeter",
        "WarmWaterTank": "https://brickschema.org/ontology/1.2/
           classes/Electric_Boiler/"
    }
}
```

List of Figures

All diagrams where drawn by the author using the free and open-source tool ‘*drawio-desktop*’. For more information see: `https://www.diagrams.net/`

Certain icons where taken from the mosaik project. The repository of these open source icons can be found here [68]. They are licensed under the GNU LGPL 2.1

List of Tables

Bibliography

[1] UNFCCC, “The Paris Agreement.” [Online]. Available: https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement

[2] Publications Office of the European Union, “Clean energy for all Europeans Energy,” 7 2019.

[3] S. O. M. Boulanger, M. Massari, D. Longo, B. Turillazzi, and C. A. Nucci, “Designing collaborative energy communities: A european overview,” *Energies*, vol. 14, 12 2021.

[4] O.-P.-S. O. Dingden and R. Kajimura, “Germany - Case Study: Outdoor-Pool-Society Of Dingden.” [Online]. Available: https://localenergycommunities.net/wp-content/uploads/2019/05/GERMANY-CASE-STUDY-1.OUTDOOR-POOL-SOCIETY-OF-DINGDEN.pdf

[5] Centria University of Applied Sciences and Einar Nystedt, “Finland - Case Study: Lohtaja Cooperative.” [Online]. Available: https://localenergycommunities.net/wp-content/uploads/2019/05/FINLAND-CASE-STUDY-2.pdf

[6] Western Development Commission and Orla Nic Dr Suibhne, “Ireland - Case Study: The Erris Community: Becoming An Sustainable Energy Community.” [Online]. Available: https://localenergycommunities.net/wp-content/uploads/2019/05/IRELAND-CASE-STUDY-1.pdf

[7] “Energiegemeinschaften AT.” [Online]. Available: https://energiegemeinschaften.gv.at/energiegemeinschaften-in-oesterreich/

[8] RIS, “Elektrizitätswirtschafts- und -organisationsgesetz.” [Online]. Available: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007045

[9] FIWARE, “About.” [Online]. Available: https://www.fiware.org/about-us/

[10] J. S. Schwarz, R. J. Eilers, and A. Ofenloch, “Mosaik 3.0 – Open-Source Smart Grid Co-simulation Framework,” 9 2021. [Online]. Available: https://zenodo.org/record/5547934

[11] Eclipse Vorto, “Homepage.” [Online]. Available: https://www.eclipse.org/vorto/

[12] ——, “Overview.” [Online]. Available: https://wiki.eclipse.org/Vorto_/_Project_Overview

[13] N. F. Noy and D. L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology,” 2001. [Online]. Available: www.unspsc.org

[14] Azure Digital Twins, “Modeling and Simulations.” [Online]. Available: https://azure.microsoft.com/en-us/services/digital-twins/#overview

[15] ——, “What is an ontology?” [Online]. Available: https://learn.microsoft.com/en-us/azure/digital-twins/concepts-ontologies

[16] A. Pras and J. Schoenwaelder, “RFC 3444 – On the Difference between Information Models and Data Models,” 1 2003. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3444

[17] European Telecommunications Standards Institute, “GS CIM 009 - V1.6.1 - cross-cutting Context Information Management (CIM),” 2022. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[18] DMTF, “Common Information Model Standard.” [Online]. Available: https://www.dmtf.org/standards/cim/

[19] OPC Foundation, “Unified architecture.” [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

[20] Wikipedia, “Information model.” [Online]. Available: https://en.wikipedia.org/wiki/Information_model

[21] FIWARE, “Developers Catalogue.” [Online]. Available: https://www.fiware.org/catalogue/

[22] ——, “NGSI v2 Specification.” [Online]. Available: https://fiware-ges.github.io/orion/archive/api/v2/

[23] ——, “Orion-LD.” [Online]. Available: https://github.com/Fiware/context.Orion-LD

[24] European Telecommunications Standards Institute, “GS CIM 009 - V1.3.1 - Context Information Management (CIM),” 2020. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[25] FIWARE, “NGSI-LD FAQ.” [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_faq/index.html

[26] W3, “JSON-LD 1.1.” [Online]. Available: https://www.w3.org/TR/json-ld/

[27] FIWARE, “Understanding @context.” [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/understanding-%40context.html

[28] ——, “Working with @context.” [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/working-with-%40context.html

[29] ——, “Linked Data.” [Online]. Available: https://fiware-tutorials.readthedocs.io/en/latest/linked-data.html

[30] Eclipse Vorto, “Release Notes.” [Online]. Available: https://github.com/eclipse/vorto/blob/master/docs/release-notes.md

[31] ——, “Getting Started.” [Online]. Available: https://github.com/eclipse/vorto/blob/master/docs/gettingstarted.md

[32] ——, “Meta Information Model Discussion.” [Online]. Available: https://wiki.eclipse.org/Vorto_/_Meta_Information_Model_/_Discussion

[33] ——, “The next step in IoT device integration.” [Online]. Available: https://blog.bosch-si.com/developer/eclipse-vorto-the-next-step-in-iot-device-integration/

[34] ——, “vortolang.” [Online]. Available: https://github.com/eclipse/vorto/blob/development/docs/vortolang-1.0.md

[35] ——, “Describing Devices/Information Models.” [Online]. Available: https://wiki.eclipse.org/Vorto_/_Describing_Devices_/_Information_Models

[36] Azure Digital Twins, “What is Azure Digital Twins?” [Online]. Available: https://docs.microsoft.com/en-us/azure/digital-twins/overview

[37] ——, “DTDL Standard.” [Online]. Available: https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDL/v2/dtdlv2.md

[38] ——, “DTDL models.” [Online]. Available: https://docs.microsoft.com/en-us/azure/digital-twins/concepts-models

[39] S. Feilmeier, wgerbl, ebakir, fabianfnc, huseyinsaht, pooran c, M. Obermeier, sebastianasen, Sagar, Kyle, luzpaz, lukasrgr, L. Kaindl, C. Lehne, ahummelsberger, lukas-bender rodriguez, M. Grill, S. Fey, greemo, AlexanderHollandFenecon, L. Verhovskij, M. Grüning, D. Jasselette, DerWahreKlinki, M. Lang, A. Fischer, Anita4Schmid, and A. Braovic, “OpenEMS/openems: 2022.8.0,” 8 2022. [Online]. Available: https://zenodo.org/record/6959465

[40] “Open Energy Management System – Core concepts and terminology.” [Online]. Available: https://openems.github.io/openems.io/openems/latest/coreconcepts.html

[41] G. Franzl, “Technical Framework on Local Energy Communities (TF-LEC) Vol.1 Version 0.5 - First trial release,” vol. 1, 2021. [Online]. Available: https://repositum.tuwien.at/handle/20.500.12708/16728

[42] T. Lechler, E. Fischer, M. Metzner, A. Mayr, and J. Franke, "Virtual Commissioning – Scientific review and exploratory use cases in advanced production systems," *Procedia CIRP*, vol. 81, pp. 1125–1130, 1 2019.

[43] W. K. Yew and D. Flynn, "Smart energy management for prosumers in local energy communities," vol. 2021-October. IEEE Computer Society, 10 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9589209

[44] C. Steinbrink, S. Lehnhoff, S. Rohjans, T. I. Strasser, E. Widl, C. Moyo, G. Lauss, F. Lehfuss, M. Faschang, P. Palensky, A. A. van der Meer, K. Heussen, O. Gehrke, E. Guillo-Sansano, M. H. Syed, A. Emhemed, R. Brandl, V. H. Nguyen, A. Khavari, Q. T. Tran, P. Kotsampopoulos, N. Hatziargyriou, A. Akroud, E. Rikos, and M. Z. Degefa, "Simulation-based Validation of Smart Grids - Status Quo and Future Research Trends," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10444 LNAI, pp. 171–185, 10 2017. [Online]. Available: http://arxiv.org/abs/1710.02315http://dx.doi.org/10.1007/978-3-319-64635-0_13

[45] E. Widl, B. Delinchant, S. Kübler, D. Li, W. Müller, V. Norrefeldt, T. S. Nouidui, S. Stratbücker, M. Wetter, F. Wurtz, and W. Zuo, "Novel simulation concepts for buildings and community energy systems based on the Functional Mock-up Interface specification," *2014 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES 2014 - Held as Part of CPS Week, Proceedings*, 2014.

[46] FMI, "Functional Mock-up Interface." [Online]. Available: https://fmi-standard.org/

[47] OFFIS, "mosaik – Publications." [Online]. Available: https://mosaik.offis.de/publications/

[48] T. Teodorowicz and A. Remke, "Comparison of SCADA protocols and implementation of IEC 104 and MQTT in MOSAIK," 2017.

[49] Smart Data Models, "About." [Online]. Available: https://smartdatamodels.org/

[50] ——, "PhotovoltaicDevice." [Online]. Available: https://github.com/smart-data-models/dataModel.GreenEnergy/tree/master/PhotovoltaicDevice

[51] ——, "PhotovoltaicMeasurement." [Online]. Available: https://github.com/smart-data-models/dataModel.GreenEnergy/tree/master/PhotovoltaicMeasurement

[52] FIWARE, "Data Models." [Online]. Available: https://uri.fiware.org/ns/dataModels

[53] IUDX, "Energy Meter." [Online]. Available: https://voc.iudx.org.in/EnergyMeter

[54] UNECE, "Annex III (Informative) Units of Measure: Code elements listed by common code." [Online]. Available: https://unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_rev3_Annex3e.pdf

[55] “NGSI-LD-Core-Context.” [Online]. Available: https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld

[56] FIWARE, “NGSI-LD How to.” [Online]. Available: https://fiware-datamodels.readthedocs.io/en/stable/ngsi-ld_howto/index.html

[57] DBpedia, “About.” [Online]. Available: https://www.dbpedia.org/about/

[58] Industrial Internet Consortium, “The Industrial Internet of Things Volume G1: Reference Architecture,” 2019. [Online]. Available: https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf

[59] CrateDB, “Distributed SQL Database Enabling Data Insights at Scale.” [Online]. Available: https://crate.io/

[60] Grafana, “The open observability platform.” [Online]. Available: https://grafana.com/

[61] MongoDB, “The Developer Data Platform.” [Online]. Available: https://www.mongodb.com/

[62] Docker, “Overview Compose.” [Online]. Available: https://docs.docker.com/compose/

[63] Telefónica I+D, “IoT Agent for a UltraLight 2.0 based protocol (with HTTP, MQTT and AMQP transports).” [Online]. Available: https://github.com/telefonicaid/iotagent-ul

[64] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, “pvlib python: a python package for modeling solar energy systems,” *Journal of Open Source Software*, vol. 3, p. 884, 9 2018.

[65] N. D. Pflugradt, “LoadProfileGenerator.” [Online]. Available: https://www.loadprofilegenerator.de/

[66] ——, “Modellierung von Wasser und Energieverbräuchen in Haushalten,” 8 2016.

[67] “Soiling Losses – Impact on the Performance of Photovoltaic Power Plants,” *Report IEA-PVPS Task 13*, 2022. [Online]. Available: https://iea-pvps.org/research-tasks/performance-operation-

[68] mosaik, “energy-icons.” [Online]. Available: https://gitlab.com/mosaik/tools/energy-icons