Development and Application of a FIWARE-based ICT-Platform for Multi-Energy Systems on Building and District Level

Lidia Westphal*, Marcel Schröder*, Daniele Carta*, André Xhonneux*, Andrea Benigni*,†,‡ and Dirk Müller*,†

*Forschungszentrum Jülich GmbH, Energy Systems Engineering (ICE-1), Jülich 52425, Germany
†RWTH Aachen University, Aachen 52056, Germany
‡JARA-Energy, Jülich 52425, Germany

***Abstract*—The integration of volatile, renewable energies into building and district energy systems requires novel algorithms that are capable of controlling the augmented complexity and the interactions of subsystems efficiently. Local energy communities will consist of a multitude of different small and large scale components using a variety of communication protocols. A reliable communication among the heterogeneous components is of utmost importance. In this regard, information and communication technology (ICT) platforms are implemented to collect data from the field devices and coordinate the various components. At Forschungszentrum Jülich (FZJ), in the framework of the Living Lab Energy Campus (LLEC) project, a FIWARE-based ICT cloud platform has been developed to gather field data from the energy demonstrators, store them and make them available for control solutions and other applications. In this paper, the architecture and the components of the LLEC ICT platform are described. The implemented solutions are motivated by the needs underlined by the demonstrators in the LLEC project, but easily transferable to other similar setups. Finally, exemplary applications that make use of the proposed ICT platform are presented, to showcase the flexibility of the platform.**

***Index Terms*—Building automation, Building management systems, Cloud computing, FIWARE, Information and communication technology, Open source software**

I. Introduction

Despite ongoing major efforts to slow down global warming, annual CO2 emissions continue to increase steadily [1]. Cities account for over 70 % of global use and, 40 to 50 % of greenhouse gas emissions worldwide [2]. Consequently, there exists an urgent imperative for implementing energy-saving strategies. Optimizing the operation of building and district energy systems presents considerable potential for achieving energy savings. At the same time, there is a need to integrate volatile, decentralized renewable energy and waste heat sources into the existing energy systems, which leads to a coupling of the thermal and electrical sectors, for example, which significantly increases complexity. Advanced control algorithms, like for instance model predictive control (MPC), form a promising solution in addressing this complexity and raise efficiency. In literature, savings of about 20 % to 30 % are reported for the building sector [3]. Regularly, these advanced control algorithms require comparatively high computing power which is often not available on-site. External cloud systems form a promising solution since they can provide the necessary computing power and storage capacities for execution, offering scalable and flexible solutions to manage the complexities of energy systems. By shifting computational tasks to external cloud systems, automation systems can leverage advanced algorithms, without overburdening local hardware resources. This approach not only enhances the efficiency and effectiveness of control strategies but also facilitates seamless integration, e.g., of data analytic tools, and allows sharing of data with users, fostering greater transparency and user engagement in energy management. Through intuitive user interfaces and personalized dashboards, building occupants can access (near) real-time and historical data about their energy usage, indoor climate conditions, and system performance. Currently, a diverse range of commercial Internet of Things (IoT) platforms are available, provided by various companies and cloud service providers. Notable examples include AWS (Amazon Web Services) IoT, Microsoft® Azure IoT, and numerous others. Nevertheless, open-source Information and Communication Technology (ICT) platforms offer a flexible, transparent and cost-effective alternative, making them an attractive option for a wide range of IoT deployments and applications. At the beginning of LLEC, a prototypical ICT setup was successfully implemented and operated without the use of a dedicated middleware. For example, a model-based room controller [4] and dashboard suite [5] were deployed over a longer period of time using this setup. Nevertheless, during operation it became once more clear, that without a dedicated middleware layer, managing and orchestrating interactions between devices becomes complex and cumbersome, leading to increased development time and effort [6]. FIWARE [7] is a notable example of an open-source middleware solution. Developed as part of the FIWARE project, it provides a comprehensive set of tools and components for developing and managing IoT solutions. Notably, FIWARE stands out for its agnostic approach, offering standardized APIs, data models, and protocols that facilitate interoperability across diverse

devices, sensors, and applications. In recent years, multiple FIWARE-based setups have been proposed, e.g. [8], [9]. In a few cases, a quick setup is made available[1]. Nevertheless, in most cases, such a deployment kit is not publicly available and considerable time must be invested again for the configuration of the components needed. In this work, we introduce a state-of-the-art customized ICT-platform built on FIWARE technology, tailored specifically for applications within the building and district energy system domain which. Even tough it was initially developped within the context of the LLEC at FZJ, the setup applies for similar setups. To facilitate seamless deployment, we have developed and released an open-source deployment kit for the proposed stack.

II. Considered framework

In 2018, at Forschungszentrum Jülich (FZJ), the Living Lab Energy Campus (LLEC) project was initiated, with the aim of transforming the campus of the research center into a living lab[2] [10]. Under this framework, several energy demonstrators for the generation, conversion and storage of energy in various forms, an extensive number of sensors and actuators, and an ICT infrastructure are being integrated to create the ideal test-bed for future multi-modal energy systems [10]. In this section, the various software components integrated in this framework are introduced.

A. Living Lab Energy Campus (LLEC)

LLEC builds on top of the already existing infrastructure of the FZJ campus, which covers an area of 1.7 square kilometres, with more than 150 buildings, where various activities are conducted, as well as test facilities and laboratories, like the Jülich Supercomputing Centre (JSC). By adding various large-scale energy demonstrators for the generation, conversion and storage of renewable energies, as well as a dedicated measurement infrastructure for research purposes, the resulting setup forms a test-bed for testing and validating novel hardware and software solutions under real-world conditions [10]. Figure 1 shows a schematic overview of LLEC's energy demonstrators.

Fig. 1. Overview of energy demonstrators, which are part of the LLEC.

[1]https://github.com/RWTH-EBC/FIWARE-STACK
[2]in German, also known as "Reallabor"

The waste heat of one water-cooled supercomputer will soon be used to supply heat to surrounding buildings via a low-temperature district heating network (LTDH), by means of heat pumps controlled by a cloud-based MPC algorithm. More than 15 buildings have been equipped with various actuators and sensors for both monitoring and control purposes [11]. They exchange data to a cloud-based room controller to provide comfortable offices with minimal energy demand. The controller considers the occupant's preferences and anticipated presence in the office, which is entered by the users via the web-based user interface JuControl. In addition to the installation of a large photovoltaic (PV) field, with a peak power of $1.1\,\mathrm{MW}$, around the campus, many small PV systems have been installed on the roofs of the buildings, or as facade-integrated photovoltaics. To provide storage for intermittent energy generated, two battery energy storage (BES) systems have been also installed on campus. The first is a high-energy Megapack BES from Tesla ($0.5\,\mathrm{MW}$/ $2.5\,\mathrm{MWh}$). The second is a high-power BES from Riello ($1.5\,\mathrm{MW}$ / $0.5\,\mathrm{MWh}$) which is also used as an uninterruptible power supply (UPS) to ensure continuous supply to a dedicated building. Close to this BES two bi-directional charging stations from Nex2, with nominal power of 250 and $250\,\mathrm{kW}$ respectively, have been installed to study vehicle-to-grid applications. Furthermore, the LLEC setup foresees a hydrogen infrastructure, featuring a test facility for electrolyzer stacks, a liquid organic hydrogen carrier (LOHC) storage and the possibility of co-firing of hydrogen in a combined heat and power plant.

B. FIWARE

As mentioned above, in the LLEC project we process data from various sources which communicate via various protocols, e.g., Modbus [12], OPC UA (Open Platform Communications Unified Architecture) [13], MQTT (Message Queuing Telemetry Transport) [14], and more. Collected data are stored as time series, with related metadata, in a homogeneous, secure and reliable way. These homogenized data must be accessible via well-defined interfaces for further processing or visualisation. Thus, an IoT (Internet-of-Things) middleware is required to manage data storage and communication between devices and applications. Furthermore, for scalability and reliability reasons this IoT middleware has to be cloud-based.

Nowadays there is a variety of commercial IoT platforms offered by different companies and cloud providers such as AWS (Amazon Web Services) IoT, Microsoft® Azure IoT and others. However, due to internal project requirements, an on-premises solution which could be deployed on our private cloud was needed. For these reasons, we chose FIWARE [7] which is an open-source framework providing components and standard architectures for smart solutions in different domains. FIWARE offers a catalogue of open-source components which are based on REST (representational state transfer), interoperable and can be combined according to the projects' needs. These components are built on top of mature and widely used databases (e.g., MongoDB [15], TimescaleDB [16], and the like), and support various standard IoT protocols, such

as MQTT, OPC-UA, AMQP (Advanced Message Queuing Protocol) [17] and many others.

In the following sections, we describe our ICT Platform employing FIWARE components for data storage and data exchange, the underlying hardware infrastructure as well as frameworks and software required for robust, secure and scalable cloud-based middleware for multi-energy building and district energy systems.

III. ICT Platform

A. Infrastructure

The reliable and secure operation of LLEC's FIWARE-based ICT platform and additional services requires a stable setup of hardware and software. IT-Services of Forschungszentrum Jülich (FZJ) operates an OpenStack cluster [18] as a virtualization infrastructure which provides the ability to run Virtual Machines (VMs). While the ICT platform is, in general, agnostic to the hardware or virtualization infrastructure it is running on, using OpenStack simplifies the administrative overhead for the users.

The services provided by the ICT platform are containerized, therefore a container orchestration system is required. We decided to use Docker Swarm [19] for this because of its ease of set-up and maintenance. This allows us to create a cluster of 3 or more VMs running the ICT-Platform, with Docker Swarm providing failover mechanisms to automatically reassign services from failed nodes to healthy ones within the cluster.

To properly run a Docker Swarm cluster additional requirements have to be fulfilled. To ensure the availability of the ICT platform in case of failure of a VM it is important to provide a unique external IP, or Domain Name System (DNS) name, for all VMs of the cluster. For this purpose, we utilise the load balancer capability natively provided by OpenStack. Nevertheless, alternative solutions such as Keepalived [20] or round-robin DNS can be used instead. In general, it is advisable to provide a mechanism to map meaningful DNS names derived from the cluster name to the ports where services are actually running, e.g., redirecting *my-http-service.cluster.domain* to *cluster.domain:80*. This way one does not need to publish the ports of each service and gains flexibility in case the ports are changed. This can be achieved by using an appropriately configured reverse proxy.

Some services of the ICT platform (like MongoDB) require storage to save their data. Docker Swarm does not have any capability to provide a single logical storage volume and make it available to the whole cluster. Thus, if a service containing data or requiring configuration fails on one VM, its data will be unavailable on the other VMs. Therefore, the service will not work properly until the original VM resumes operation, if the data are not restored otherwise. To avoid these issues, and to reduce the need for external storage, GlusterFS [21] is used. GlusterFS enables the creation and management of a virtual volume available to all VMs of the Cluster, while the actual data saved on the virtual volume is replicated in the base storage of each VM. This mechanism is used to provide configurations, docker-compose files, and the like, within the whole cluster. Since the synchronization process with the default configuration of GlusterFS comes with performance losses, we decided to limit the distribution of MongoDB services to certain VMs with each of them storing their data internally.

Finally, to ensure that encrypted communication to the cluster is possible, we use Certbot [22]. Certbot is a free, open-source, widely used software for creating and renewing Let's Encrypt TLS (Transport Layer Security) certificates [23] automatically.

All of these individual components enable us to provide a stable and reliable foundation for the ICT platform built on top of it.

B. Platform Components

The proposed ICT platform consists of multiple individual containerized services. We use the Eclipse Mosquitto MQTT broker [24] and a set of FIWARE components to store and exchange device data which are explained in more detail below. Although FIWARE also offers components for storing time-series data, e.g., QuantumLeap [25], we decided to rely on our existing InfluxDB-based solution, for which we implemented a custom API to receive data from FIWARE. Not being a part of the ICT platform, InfluxDB is beyond the scope of this paper. Figure 2 shows a schematic overview of the components.

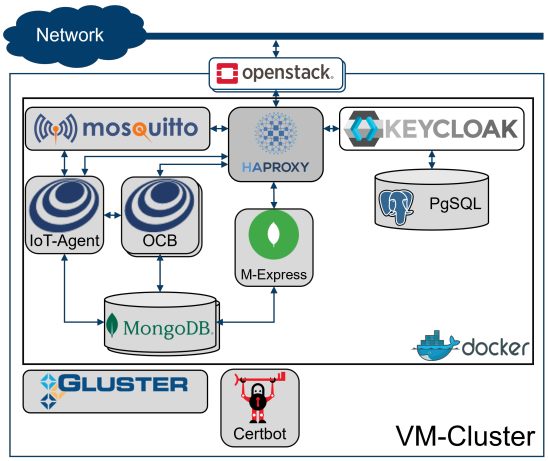

Fig. 2. Overview of the ICT platform setup.

1) Eclipse Mosquitto: for the communication between the edge layer and FIWARE, we mainly use the MQTT protocol. Thus, we installed Eclipse Mosquitto MQTT broker which is an open-source MQTT implementation recommended by the FIWARE foundation. Eclipse Mosquitto MQTT broker is easy to install and configure and it supports a variety of MQTT communication options:

- Plain MQTT
- MQTT over TLS
- MQTT over TLS (with client certificate)
- MQTT over WebSockets
- MQTT over WebSockets with TLS

2) FIWARE NGSI: FIWARE NGSI (Next Generation Service Interfaces) is an information model and API specification used for the interaction between FIWARE components within the FIWARE ecosystem. FIWARE NGSI is also used by applications to update, or process, context information. There are two NGSI specifications currently used in the FIWARE community:

- NGSI v2, offering JSON based interoperability
- NGSI-LD [26], which has been developed on the basis of JSON-LD [27]

3) Orion Context Broker: FIWARE Context Broker is the core and mandatory component of any FIWARE platform. FIWARE offers several Context Broker implementations which differ in the FIWARE NGSI version they support (NGSI v2 or NGSI-LD), communication protocols they use (HTTP, MQTT) and integrations they bring with them.

We use FIWARE Orion Context Broker which implements NGSI v2 specification and supports HTTP communication protocol. It offers:

- a data model for context information, which is based on entities where each represents a physical or logical object (e.g. a sensor, a room, etc.), their attributes (e.g. temperature, location, etc.) and relationships between them;
- an interface for accessing entities and their attributes through queries, subscriptions, and update operations.

The OCB stores context information and metadata of the actual field devices while managing relationships between them and other logical entities, e.g., a relationship between a humidity sensor and the room where it is installed.

4) IoT-Agent JSON: a FIWARE IoT Agent is a bridge component between various, widely-spread device protocols, e.g., OPC UA, Ultralight 2.0, LoRaWAN (Long Range Wide Area Network) [28], etc. and the NGSI interface of a FIWARE Context Broker. For the communication and data exchange over MQTT we use the FIWARE IoT Agent for a JSON-based protocol. In particular, the FIWARE JSON IoT-Agent subscribes to the Mosquitto MQTT broker and translates messages received via MQTT to the FIWARE custom data format and interface language, NGSI v2 (Next Generation Service Interface) [29], and vice-versa.

5) MongoDB: both FIWARE components, the OCB and the JSON IoT agent, make use of the MongoDB database for data persistence. MongoDB is a source-available, NoSQL document-oriented database with a free-to-use Community version. To reduce the probability of data loss, to increase the data availability and to ensure automatic failover we set up a three-node replica set. A replica set is a group of MongoDB instances maintaining the same data, having one primary node which receives all write operations, and secondary nodes which replicate the primary node and can serve read operations.

6) Mongo-Express: a visualization tool for MongoDB [30]. It provides a web interface with the capability to browse through all databases, tables and entries stored within MongoDB. In our setup, Mongo-Express is being used to easily manage the MongoDB.

7) Keycloak: to enforce user permissions and to establish correct data access policies the open-source identity and access management software Keycloak [31] is used. Keycloak provides such functionality as user management, authentication and fine-grained authorization. Within Keycloak it is possible to define users, groups and roles/scopes and to allow access to web applications and RESTful APIs based on these definitions. It is also possible to integrate Keycloak with existing LDAP (Lightweight Directory Access Protocol) and Active Directory servers.

To request access for a specific API, a request is sent towards the Keycloak API containing the predefined user credentials. After validating the login data, Keycloak returns a JWT (JSON Web Token) [32], all roles and scopes defined for that specific user, the token's expiration time (the time for how long the token is valid, which is 5 min by default) and more. With that token, it is possible to send a proper request towards the target API over HAProxy. HAProxy can then authorise requests based on the token. We configured Keycloak to store its data in PostgreSQL [33] which is a widely-used and reliable database.

8) HAProxy: for obvious reasons, it is not recommended to provide unencrypted websites, APIs and so on on a possibly public network. One way to implement the TLS encryption of communication is to hide the actual services in a private network behind a reverse proxy and to configure the reverse proxy to encrypt the network traffic between it and clients via a TLS certificate and its private key.

For the TLS encryption of the communication between client applications and the ICT platform, as well as for the integration of Keycloak, we employ HAProxy [34]. HAProxy is an open-source software providing, among other things, reverse proxy and load balancer functionality for web and TCP-based applications. Thus, HAProxy uses the TLS certificate created by Certbot to encrypt the communication. It is easy to set up, fast, efficient and reliable. Additionally, HAProxy can be customised using individual scripts based on the Lua language. Besides, it can be managed by multiple configuration files to easily extend the base configuration.

For proper redirection of requests, HAProxy can use subdomain names and the internal Docker DNS which allows service name resolution for all services within a network. This way, HAProxy can redirect incoming requests to the desired service based on the URL without publishing the port used internally, e.g., if the ICT platform is running on *example.com* it is possible to map *orion.example.com* to the OCB service.

C. Deployment

Navigating the intricacies of the FIWARE and other components and configuring them to ensure stable operation can require considerable effort in familiarization and fine-tuning. This involves comprehensive exploration and understanding of each component's functionalities, interfaces, and dependencies. It requires close attention to detail and a systematic

approach to configuring parameters, orchestrating interactions, and resolving potential conflicts or compatibility issues. Moreover, it often entails extensive testing and validation to verify the effectiveness and reliability of the configured components under various scenarios and workloads. For easy deployment of the proposed setup in this paper on Docker Swarm, we made a deployment kit available as open source [35].

IV. Exemplary Applications

The initial monitoring and control applications developed within LLEC were successfully deployed without relying on a dedicated middleware solution by the prototypical setup proposed by Redder et al. [6]. Following the deployment of the proposed FIWARE-based setup, these and additional applications are now being migrated. The current setup is shown in Fig. 3. For the ingestion of data, dedicated adapters were developed, which are for the sake of brevity not described further here.

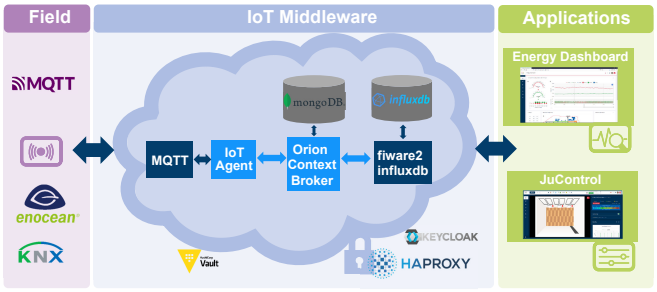

Fig. 3. FIWARE-based ICT-Platform with first applications.

In the following, two use cases are introduced and discussed briefly.

A. Ingestion of metering data

The first application deployed making use of the new setup is the transmission and ingestion of metering data of the regular metering infrastructure at FZJ, which provides for the measurement of e.g. electricity, heating and cooling requirements at building level. This data is used for various purposes, including the display of energy data for FZJ's staff members via the LLEC Energy Dashboard (see Fig. 4). In the upper part, the green curve shows the overall electricity demand and the red curve the overall heating demand of FZJ's campus.

In this case, the adapter automatically provisions a new metering data point as soon as a new data point occurs that fulfils the requirements of the scheme. We found that the automatic registration and logging of about 500 metering data points worked flawlessly.

B. Ingestion of sensor data on room level

Within the aforementioned web application JuControl, users can visualize measurement data of the sensors installed in their office (see Fig. 5). The figure shows the user's view of an exemplary office. The current weather data is shown

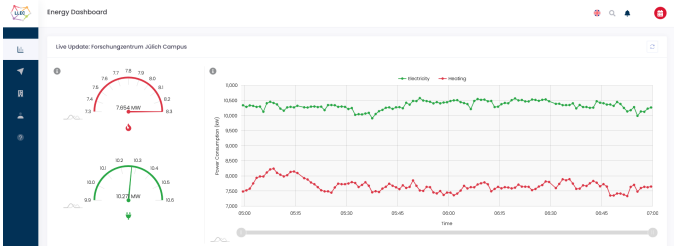

Fig. 4. LLEC Energy Dashboard - Energy demands of FZJ campus and buildings (green curve = electricity demand, red curve = heating demand) [5].

in the upper section. In the upper left part, the current indoor air (quality) measurements of the room are displayed (CO2 concentrations, relative humidiy level and air temperature).

The majority of sensors are wireless sensors making use of the EnOcean protocol but also wired KNX sensors and actuators are part of the setup. Some of these sensors report a measured value on a time-controlled basis, while others report in the event of a status change. The interface between the field level and cloud is formed by edge devices, where the Machine-to-machine (M2M) protocol ADS is used for communication between the edge devices and cloud [6]. Data models and the so-called pyADS-Adapters were developed in order to establish communication between the FIWARE and the edge devices in the field. The edge devices need to be parametrized for the specific EnOcean devices and group addresses as they are programmed as a part of the KNX setup. Adding new EnOcean devices or KNX group address triggers the provisioning of the device in FIWARE.

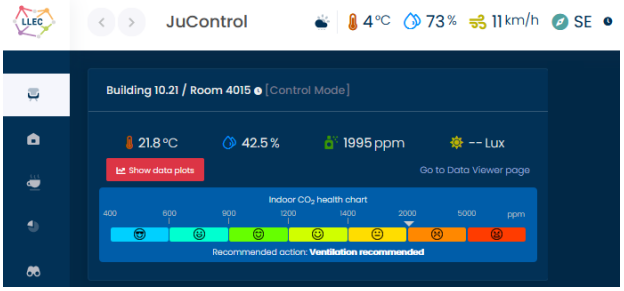

Fig. 5. JuControl - Excerpt of the web-based interface of an exemplary office - showing real-time measurement data regarding indoor air (quality) and ambient conditions. [5].

Some of the sensors (e.g. door and window state sensors) report in case of change of value (COV). Other sensors (e.g. the wireless thermostats), which depending on the actual operating mode, measure and report either the flow temperature at the radiator or the room temperature in the vicinity of the radiator based in a parameterizable time interval (e.g. 15 minutes). Some other sensors (e.g. operating panels next to the door) report both a parameterizable time interval and COV. We found that the automatic provisioning and logging of about 4,000 EnOcean devices and 20,000 KNX group addresses worked flawlessly and data transfer ran smoothly.

V. Transferability

While the ICT platform has initially been developed for the LLEC setup at FZJ, its architecture and functionality are designed to fit to the needs of similar setups elsewhere. This inherent adaptability and scalability make the ICT platform highly transferable to other setups, e.g. city districts or local energy communities facing comparable challenges and requirements. The architecture ensures that it can be seamlessly integrated and customized to suit varying environments. For an easy deployment, a deployment package has been prepared to provide a production ready setup for the FIWARE Platform. The repository contains all information to setup FIWARE on a a Docker Swarm cluster. Further details can be found in the description of the corresponding Gitlab repository.

VI. Conclusions

In this work, a setup for a FIWARE-based Information and Communication Technologies (ICT) platform for local energy communities is proposed. Core components of the FIWARE setup have been adopted, e.g. the Orion Context Broker (OCB). The setup diverges at specific points as required, accommodating unique needs and circumstances, e.g., with respect to the time-series database. For a first service, it could be demonstrated that the proposed setup works well for first use-cases at Forschungszentrum Jülich, such as the ingestion of metering data of the regular metering infrastructure of the research center and data of wireless sensors at room level. For easy deployment for similar use-cases, a deployment kit has been developed and made available as open source.

While the ICT platform and all its components are already sturdy and reliable, additional adjustments will be made in future. An upcoming add-on is the addition of logging and monitoring through an *ElasticStack* (Elasticsearch, Kibana and Filebeat/Metricbeat). It is also planned to move from a Docker swarm setup to a Kubernetes setup for a more future-proof setup.

Funding

Parts of this research were funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK), grant numbers 84703ET1551A, 03ET1551A, and the German Federal Ministry of Education and Research (BMBF), grant number 03EK3047.

Acknowledgments

The authors gratefully acknowledge the constructive collaboration with the internal and external project partners. In particular, we would like to thank the Jülich Supercomputing Center (JSC), and IT-Services (ITS).

References

[1] F. et al., "Global carbon budget 2022," *Earth System Science Data*, vol. 14, no. 11, pp. 4811–4900, 2022. [Online]. Available: https://essd.copernicus.org/articles/14/4811/2022/

[2] International Energy Agency, "CO2 Emissions in 2022," International Energy Agency, Paris, France, Tech. Rep., 2023. [Online]. Available: https://www.iea.org/reports/co2-emissions-in-2022

[3] J. Drgoňa, J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Ollé, J. Oravec, M. Wetter, D. L. Vrabie, and L. Helsen, "All you need to know about model predictive control for buildings," *Annual Reviews in Control*, vol. 50, pp. 190–232, 2020.

[4] M. Mork, F. Redder, A. Xhonneux, and D. Müller, "Real-world implementation and evaluation of a model predictive control framework in an office space," *Journal of Building Engineering*, vol. 78, p. 107619, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352710223017990

[5] E. Ubachukwu, J. Pick, L. Riebesel, P. Lieberenz, P. Althaus, D. Müller, and A. Xhonneux, "LLEC energy dashboard suite: User engagement for energy-efficient behavior using dashboards and gamification." ECOS2023 - 36th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems, Las Palmas de Gran Canaria, Spain, 2023, doi: 10.34734/FZJ-2024-00035.

[6] F. Redder, P. Althaus, E. Ubachukwu, M. Mork, S. Johnen, C. Küpper, P. Lieberenz, M. Oden, L. Westphal, T. Storek, , A. Xhonneux, and D. Müller, "nformation and communication technologies (ict) for holistic building energy system operation in living labs: Conceptualization, implementation, evaluation," *Pre-print available at 10.2139/ssrn.4743282*.

[7] FIWARE - open APIs for open minds. [Online]. Available: https://www.fiware.org/

[8] T. Storek, J. Lohmüller, A. Kümpel, M. Baranski, and D. Müller, "Application of the open-source cloud platform fiware for future building energy management systems," *Journal of Physics Conference Series Journal of Physics Conference Series 1343(1)*, 2019.

[9] G. Vaglica, F. Bono, and G. Renaldi, *A JRC FIWARE testbed for SMART building and infrastructures: Implementation of the FIWARE platform for performance testing and heterogeneous sensor nodes*, ser. EUR. Luxembourg: Publications Office of the European Union, 2020, vol. 30038.

[10] A. Benigni, A. Xhonneux, D. Carta, T. Pesch, and D. Müller, "On the development of control solutions for local energy communities: An incremental prototyping approach and related infrastructure," *at - Automatisierungstechnik*, vol. 70, pp. 1095–1115, 2022.

[11] P. Althaus, F. Redder, E. Ubachukwu, M. Mork, A. Xhonneux, and D. Müller, "Enhancing building monitoring and control for district energy systems: Technology selection and installation within the living lab energy campus," *Applied Sciences*, vol. 12, no. 7, p. 3305, 3 2022.

[12] Modbus. [Online]. Available: https://modbus.org

[13] OPC UA. [Online]. Available: https://opcfoundation.org

[14] MQTT. [Online]. Available: https://mqtt.org

[15] MongoDB. [Online]. Available: https://www.mongodb.com

[16] TimescaleDB. [Online]. Available: https://www.timescale.com

[17] AMQP. [Online]. Available: https://www.amqp.org

[18] OpenStack. [Online]. Available: https://www.openstack.org

[19] Docker. [Online]. Available: https://www.docker.com

[20] Keepalived. [Online]. Available: https://www.keepalived.org

[21] GlusterFS. [Online]. Available: https://www.gluster.org

[22] Certbot. [Online]. Available: https://certbot.eff.org

[23] Let's Encrypt. [Online]. Available: https://letsencrypt.org

[24] Eclipse Mosquitto™. [Online]. Available: https://mosquitto.org

[25] QuantumLeap. [Online]. Available: https://github.com/orchestracities/ngsi-timeseries-api

[26] NGSI LD. [Online]. Available: https://forge.etsi.org/swagger/ui/?url=https://forge.etsi.org/rep/NGSI-LD/NGSI-LD/-/raw/1.6.1/ngsi-ld-api.yaml

[27] JSON LD. [Online]. Available: https://json-ld.org

[28] LoRaWAN. [Online]. Available: https://lora-alliance.org

[29] NGSI v2. [Online]. Available: https://swagger.lab.fiware.org/?uri=https://raw.githubusercontent.com/Fiware/specifications/master/OpenAPI/ngsiv2/ngsiv2-openapi.json

[30] Mongo-Express. [Online]. Available: https://www.timescale.com

[31] Keycloak. [Online]. Available: https://www.keycloak.org

[32] JSON Web Token. [Online]. Available: https://jwt.io

[33] PostgreSQL. [Online]. Available: https://www.postgresql.org

[34] HAProxy. [Online]. Available: https://www.haproxy.org

[35] ICT platform · GitLab. [Online]. Available: https://jugit.fz-juelich.de/iek-10/public/ict-platform