IoT Integration for Combined Energy Systems at the CoSES Laboratory

Matthias Mayer
Department of Electrical and Computer Engineering
Technical University of Munich

Anurag Mohapatra
Munich School of Engineering
Technical University of Munich

Vedran S. Perić
Munich School of Engineering
Technical University of Munich

Abstract—**Combined Smart Energy Systems (CoSES) microgrid lab at TU Munich, has a large number of measured and calculated data points which need a robust tool to monitor and visualize. Furthermore, the tool should be simple to use among microgrid researchers from different fields of study. Based on these requirements, an IoT dashboard using open-source tools for experiment monitoring and visualization is developed for the CoSES microgrid lab. The solution uses InfluxDB 2.0 as a time-series database and Grafana for the visualizations. The dashboard can be easily customized and is both locally and remotely accessible from any modern browser. This enables the researcher to analyze the experiment data without detailed knowledge of the Hardware-in-the-loop setup. A demonstrative experiment is conducted using one CoSES prosumer to verify the toolchain and the results are shown in the paper.**

I. Introduction

The Center for Combined Smart Energy Systems (CoSES) at TU Munich was established to research sector coupled, low-inertia, active distribution grids required for the future sustainable energy systems. The lab is realized as a small microgrid, with electrical and heat subsystems, with close integration of information and communication technology (ICT). It is contains six fully controllable electrical and heat prosumers, based on Power-Hardware-in-the-loop (PHIL) philosophy, which can emulate a real world multi energy grid and thus eliminating the need to model them using real-time simulators. Detailed hardware specifications for the lab can be found in our previous publication [1].

One aspect of ICT usage in multi energy systems, such as CoSES, is the acquisition, logging and visualization of the grid data in real time. To this end, the presence of a variety of sensors, embedded systems, smart devices and protocols, over the different energy systems adds to the complexity of the problem. Finally, the ease of access to the data and enhanced user experience in handling the time-series of measurements are also of significance. The research within CoSES is vastly inter-disciplinary and thus attracts practitioners from fields outside the traditional power systems community. To facilitate collaboration, it is crucial to be able to abstract the lab and the results to the relevant level of detail based on the researcher's need. Therefore, a tool is needed to present the lab and its experiments at a glance and to be able to monitor results for the systems of your interest, from your own desk either within the lab building or elsewhere.

IoT paradigms offer an elegant solution to these needs where we can consider our prosumers or devices as intelligent nodes. This can extend widely used connectivity and seamless integration concepts from the web towards the distributed embedded controllers for heat and electrical grid in the CoSES smart grid [2].

This paper presents an open-source IoT dashboard developed for CoSES. This tool can be used to monitor experiments in real-time and trend the previous results for analysis. It extends the sensor measurements and calculated data points, from models in the PHIL setup, to a time-series database. This database backup can be visualized quickly by the user for analyses, not limited to, comparing different runs of the same experiment, trend high-level alarms and events or observe physical phenomenons over a wide-range of timescales.

The remainder of the paper is structured as follows. In Section II we explain the general control philosophy of CoSES. Then in Section III, we introduce the IoT toolchain to connect to this control network. A demonstration experiment is presented in Section IV to show how the dashboard visualizes the lab and the concluding statements are provided in Section V.

II. CoSES Control Philosophy

The control architecture of CoSES is based on the following three key features,

- *Modelling freedom* - Researchers working on PHIL test benches in CoSES, must have the option to work with the modelling and simulation tool of their choice to design operation and control strategies.
- *Time scales synchronization* - Control models in CoSES have widely different time scales based on energy system and model objective. They need to be synchronized to maintain real time causality through data exchange between models.
- *Commercial device protocols* - PHIL test benches should be open to commercially available smart grid components, and therefore industry standard communication protocols, to ensure the research is easily implemented in a real world scenario.

CoSES uses National Instruments (NI) hardware and software to realize the real-time control system within the lab. A schematic of the implementation is shown in Fig 1. Six PXIe-8880 embedded controllers are used as distributed real-time control agents for the electrical grid and six IC-3171

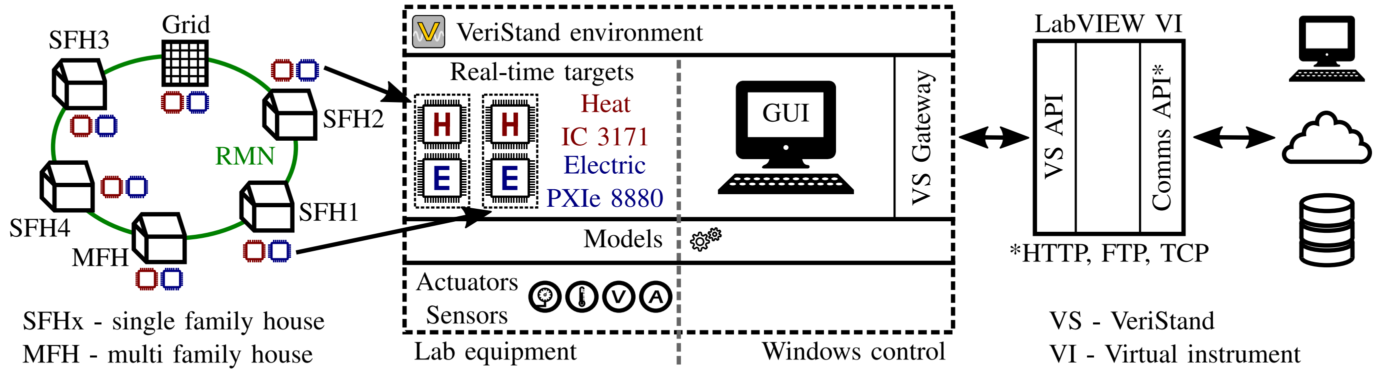


Fig. 1. Control schematic for CoSES Microgrid Lab.

are similarly used for the heat grid. An optic fiber ring called Reflective Memory Network (RMN) is used to make data recorded or calculated at any distributed agent available everywhere within the network within 1 ms. This can be seen in the left part of the Fig. 1.

NI VeriStand is a software environment for real-time PHIL applications and can be deployed over general purpose computers and embedded controllers [3]. In the central part of Fig. 1, an example CoSES project with two prosumers is shown and the role of Veristand is emphasised. Measurements for current, voltage, temperature, pressure, flows and counters are acquired within Veristand through associated NI input-output (IO) cards. The rate of signal acquisition is different for different energy systems. PHIL operation setpoints are generated in prosumer control and grid emulator models and relayed back to the hardware from Veristand through the same IO cards. These PHIL models are compiled using a variety of tools, not limited to, MATLAB/Simulink, LabVIEW, SimulationX and Python. Veristand is able to synchronously deploy these models to the real-time embedded controllers in the lab and Windows PCs in the control room. The environment is capable of handling the different toolchains used for the models, their different execution rates and mapping IOs in-between models and hardware. On the Windows PC in the control room, a rudimentary operator screen with configurable user interface (UI) is also provided. This can be used to interact with the experiment in real-time and for low level monitoring of the results.

The right side of Fig. 1, shows how the Veristand environment can be opened to the general data processing and communication protocols. LabVIEW is a graphical programming language by NI which is used as a backend unit for many other NI software [4]. VeriStand offers an Application Programming Interface (API), in a LabVIEW toolbox, which connects to a gateway on the control room Windows PC. This API can be used to read from and write into channels for both models and hardware, within a deployed real-time project. A LabVIEW virtual instrument (VI) can combine the Veristand API with any other toolboxes for standard communication protocols. This creates a powerful tool to interact with the PHIL test bench from virtually any platform or over any protocol.

III. CoSES as an IoT Network

As mentioned in Section II, Veristand offers rudimentary support for operator screens, graphical viewing of data and control panel widgets to interact with the hardware. However it is not meant for long term visualisation and data handling in a manner consistent with state-of-the-art ideas on data management. IoT based services and applications offer the ease of usage and robust handling of data which can be useful for a lab like CoSES.

A. Network Architecture

The lab consists of actuators and sensors connected to controllers on the same switch. Therefore, as explained in Section I, the setup can be re-imagined as an IoT network with each house being an intelligent node. Through the VeriStand API, it is also possible to couple the PHIL infrastructure with an open-source analytics and monitoring platform. This can be supported by a time-series database and also enable remote viewing of the experiments.

A schematic of the CoSES IoT setup is shown in Fig. 2 and is divided into three sections - lab equipment, local network and cloud network. In the lab, the CoSES prosumers send measurements through sensors and receive setpoints from the embedded controllers. Within a single Veristand project, the embedded controllers for each prosumer, run synchronously with their respective PHIL models. In the local network, a VeriStand instance running on the control room Windows PC is also part of the same project. In CoSES, there are approximately 1000 different channels which are either measured or calculated in Veristand. As already explained in Section II and shown in Fig. 1, a LabVIEW VI for datalogging can connect to the PHIL project using an API and then broadcast the channels to a time-series database using standard protocols. The data is stored into a time-series database and read by a local observation tool with a dashboard. A short term copy of the data is created, within the local network, in a real-time metrics database provided by a cloud observation tool. In the cloud network, this temporary copy is read by a cloud counterpart and associated with a remotely hosted dashboard. Thus a solution is proposed for both local and remote monitoring of the lab over any modern browser.

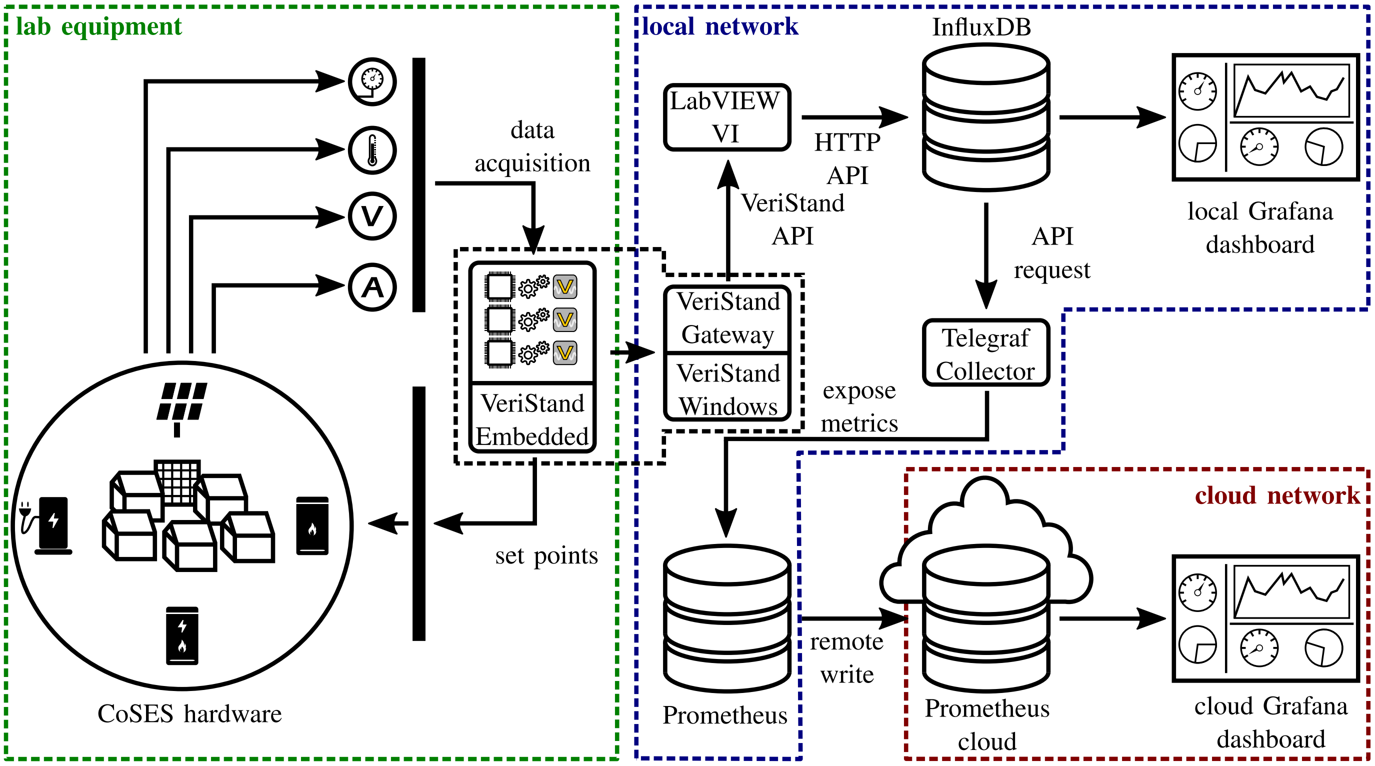
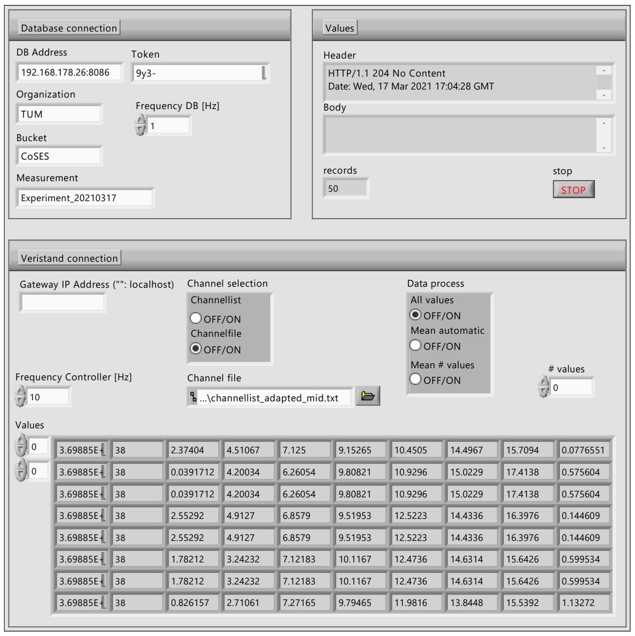


Fig. 2. CoSES laboratory control schematic with the IoT toolchain and dashboard.


Database connection

DB Address: 192.168.178.26:8086
Token: 9y3-
Organization: TUM
Frequency DB [Hz]: 1
Bucket: CoSES
Measurement: Experiment_20210317

Values

Header:
```
HTTP/1.1 204 No Content
Date: Wed, 17 Mar 2021 17:04:28 GMT
```
Body:
records: 50
stop: STOP

Veristand connection

Gateway IP Address ("": localhost):
Channel selection: Channellist OFF/ON; Channelfile [✓ OFF/ON]
Data process: All values [✓ OFF/ON]; Mean automatic OFF/ON; Mean # values OFF/ON
# values: 0
Frequency Controller [Hz]: 10
Channel file: ...\channellist_adapted_mid.txt

Values: 0, 0

3.69885E+	38	2.37404	4.51067	7.125	9.15265	10.4505	14.4967	15.7094	0.0776551
3.69885E+	38	0.0391712	4.20034	6.26054	9.80821	10.9296	15.0229	17.4138	0.575604
3.69885E+	38	0.0391712	4.20034	6.26054	9.80821	10.9296	15.0229	17.4138	0.575604
3.69885E+	38	2.55292	4.9127	6.8579	9.51953	12.5223	14.4336	16.3976	0.144609
3.69885E+	38	2.55292	4.9127	6.8579	9.51953	12.5223	14.4336	16.3976	0.144609
3.69885E+	38	1.78212	3.24232	7.12183	10.1167	12.4736	14.6314	15.6426	0.599534
3.69885E+	38	1.78212	3.24232	7.12183	10.1167	12.4736	14.6314	15.6426	0.599534
3.69885E+	38	0.826157	2.71061	7.27165	9.79465	11.9816	13.8448	15.5392	1.13272

Fig. 3. User interface of LabVIEW VI for datalogging.

B. *LabVIEW VI for Datalogging*

A LabVIEW VI, as seen in Fig. 2, is the intermediary between the measurements and the time-series database in the local network. It is also an alternative to the Veristand UI to configure execution parameters for the PHIL project. LabVIEW also offers an HTTP API toolbox which can execute standard methods such as POST and GET. These are used to transfer the data from the VeriStand channels to an InfluxDB 2.0 database [5]. InfluxDB 2.0, is a time-series database designed to handle the volume of timestamped data generated within projects like CoSES. It is also compatible with popular observation platforms as an additional benefit.

However, before the data can be transferred to a database, it has to be first logged from the channels with certain flexibility in choosing them and their frequency of logging. The mandatory inputs and the VI user interface is shown in Fig. 3. The bottom part of Fig. 3 shows some channel values received from VeriStand and the top half shows the response from the HTTP API. The Algorithm 1 shows the data logging idea in brief. The VI takes the location of the InfluxDB time-series database, the IP and port address and token credentials to write into the database as inputs. These do not need to be changed in subsequent runs unless the database itself is changed.

The VI must be provided with a channel list, which is obtained from the VeriStand signal mappings and can be exported directly as a text file. The next input is the frequency f_c, which determines the rate at which the VeriStand API requests new data from the channels. If this rate is too fast, the API would not have finished processing the previous batch before the new read/write request is initialized leading to duplicate values. Finally, the frequency f_{db} must be provided, which determines the rate at which the buffered data is written into the database. All the data can be transferred to the database as it is or can be optionally averaged using two modes

Algorithm 1: LabVIEW VI for datalogging pseudo code.

```
Controller ← $IP\,address,\ channels,\ f_c$;
Database ← $IP\,address,\ port,\ token,\ f_{db}$;
$r \leftarrow \frac{f_c}{f_{db}}$;
while $!stop$ do
    while $i < r$ do
        read channel values;
        store data in array;
    end
    send values to InfluxDB;
end
```

made available in the VI. The automatic mode averages over all the buffered data and in the manual mode the user can set averaging range as an integer between 1 and the frequency ratio, r. For example, if $f_c = 10\,\mathrm{Hz}$, $f_{db} = 1\,\mathrm{Hz}$ and the manual averaging is set to 5, it will send two entries per second per channel into the database whereas the automatic averaging will send one entry per second per channel. This is helpful for very long experiments, lasting multiple days, where even a 5min precision of data is acceptable.

A nested while loop structure is at the core of the VI. The inner loop collects the specified channel values from the controller at f_c rate and buffers them in two arrays - a two dimensional data array and a single dimensional timestamp array. Individual columns in the data array are used for each individual channel which fills up in every execution. Normally, a time-series database such as InfluxDB 2.0 would automatically generate its timestamp when entries are updated. However to synchronize the entries with the lab, the system time from the controllers is read as a further channel from VeriStand and used to timestamp the signals. When the inner loop index, i, reaches the frequency ratio, r, the arrays are combined to a string and passed through the HTTP POST method to InfluxDB 2.0 as a new entry. Once the data insertion is finished, the inner loop restarts and the process is repeated until the experiment is stopped or an unexpected error occurs.

C. *Visualization of the data*

A powerful visualization tool is needed to exploit a time-series database for analysis and observation. Since the use case is in research, an open-source tool is always preferable due to the community around it and access to the source code if necessary. A variety of free open-source software exist for this purpose. Grafana was the most suitable for CoSES as it accepts around 30 different data sources, has many visualization plugins, is well documented and has a robust growing community [6]. Grafana can run locally on the PC to generate dashboards on any modern browser on a localhost address. The Grafana server runs as an executable on the local PC and is configured with a .ini file and the browswer user interface (UI). The InfluxDB 2.0 database is arranged as sets of measurements associated with an experiment label. Grafana can perform queries on this database and visualize the stored metrics on its dashboard UI.

The Grafana dashboard is developed using panel plugins available in the tool. The dashboard allows the operator to choose between the tracked experiment and subsequently, the available data sets for each experiment is populated in a drop down list with check boxes. Thus previous experiments can be analysed with the full dataset and experiments currently underway update the plots as new data arrives. The minimum rate of data refresh is set in the Grafana .ini file and the actual rate can be set by the operator on the dashboard based on need. The default value is at $5\,\mathrm{s}$ and this can be lowered depending on how fast the queries can get executed. Ultimately the ideal refresh rate and the maximum viewing window, depends on the amount of data being visualized at once and the computational resources of the PC used by the operator.

D. *Remote access to the dashboard*

The localhost dashboard limits the observation within the CoSES building network. For internet based viewing we run Grafana Cloud which uses the Prometheus time-series database [7]. Automatic transfer of measurements from the local InfluxDB 2.0 database is enabled through the metrics collector service, Telegraf [8] and a local Prometheus database instance as shown in Fig. 2. In this setup, Telegraf sends an API request to the local InfluxDB 2.0 and receives the new measurements back. It then converts the data to a format compatible with Prometheus and exposes them on a localhost port. The local instance of Prometheus keeps listening on this port and records the new data. Later, a periodically executed remote write command extends this data to a Prometheus cloud instance which is linked to the Grafana Cloud services and its associated dashboards. Although user accounts are needed to access the cloud services, a snapshot of the dashboards can be hosted on a separate webserver and be refreshed periodically. Thus a truly public platform can be created to observe the CoSES experiments as they take place from anywhere in world. This allows researchers to share live results with collaborators and enhances the possibilities through an online presence.

IV. Experiment and results

A simple experiment, shown in Fig 4, was setup to demonstrate the aforementioned IoT tools and the dashboard. On the electrical grid, two room heaters, a cooking plate and an electric drill were added as single phase loads. For the heat grid, a condensing boiler was used as the heating source combined with a domestic hot water storage facility. Although CoSES contains six electrical and heat prosumers, for simplicity reasons, only one was used for the experiment in this paper. This does not diminish the results as the dashboard and the associated tools can be extended to the other prosumers with no extra work provided every controller has its respective PHIL models.

Three phase current and voltage measurements were acquired for the electrical grid. Water flow and temperatures

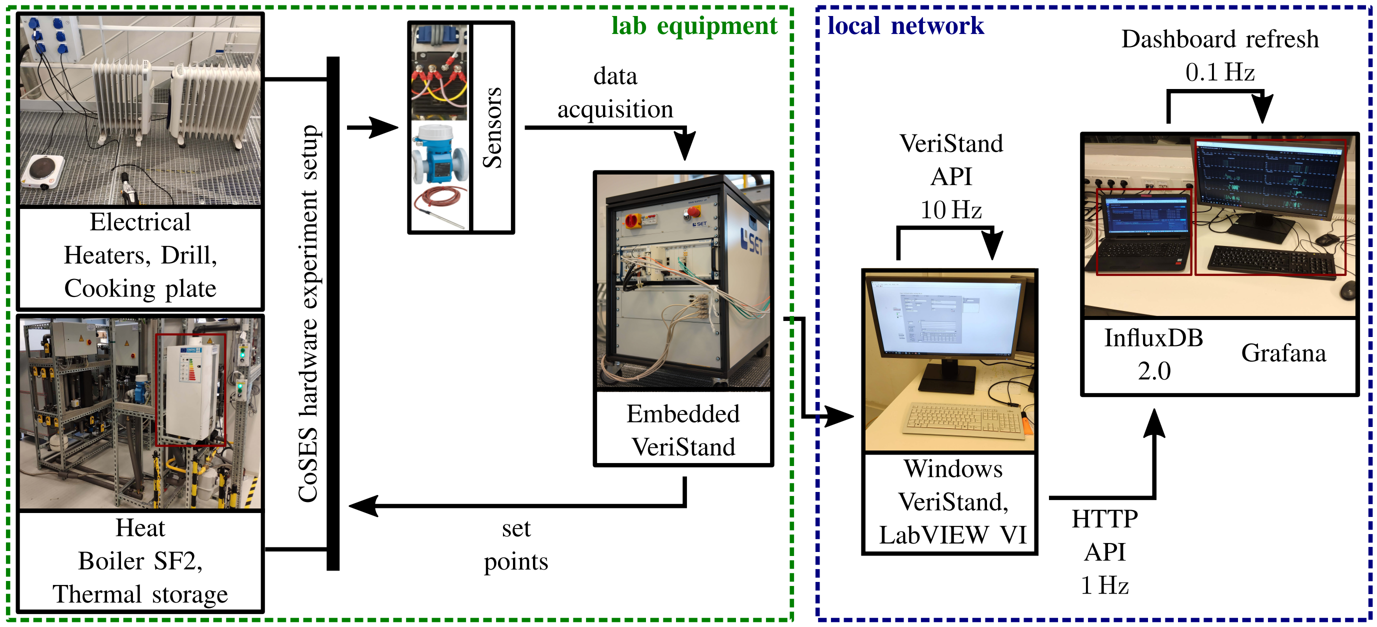


Fig. 4. Experiment setup for IoT dashboard demonstration in the CoSES laboratory.

TABLE I
DETAILED EXPERIMENT PARAMETERS.

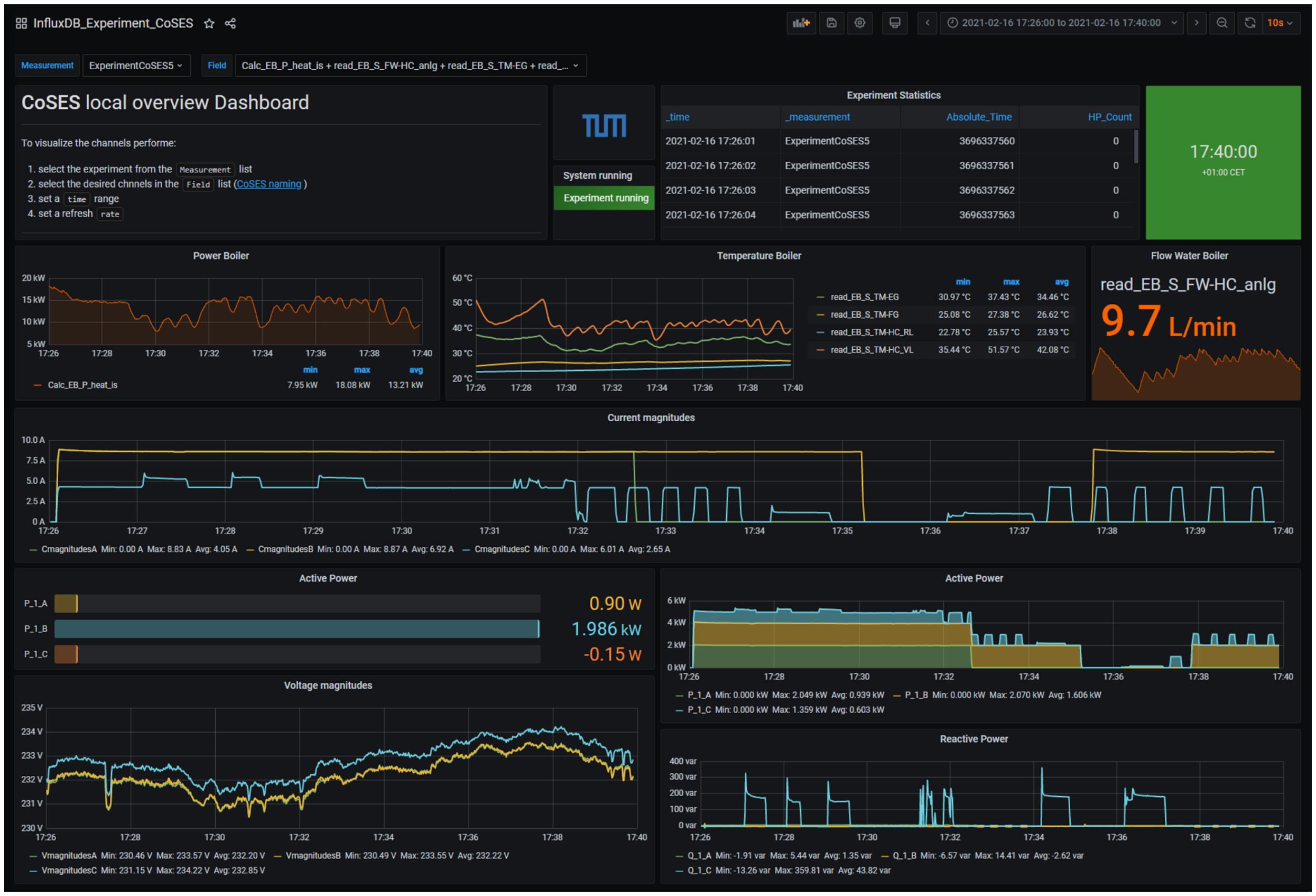
	Electrical grid
	Heat grid
Equipment	2 x 2 kW room heater, 1 x 1 kW cooking plate, 1 x Electric drill
	1 x 20 kW$_{th}$ Condensing boiler 800L Thermal storage
Sensor	4 x LEM LF 210-S/SP3 Current transducers 4 x LEM CV 3-1000 Voltage transducers
	10 x PT-100 4 wire, temperature sensors 2 x Promag E 100, flow sensors
Data acquisition card	NI PXIe 4303, AI card
	NI REM-11100, AI card (Flow) NI 9216 RTD, cRIO AI card (Temperature)
Embedded Controller	PXIe-8880, Pharlap RTOS Target rate - 10 kHz
	IC-3171, Linux 64-bit Target rate - 100 Hz

were measured in the heat grid. The boiler loading and control models were deployed through VeriStand on the IC-3171 embedded controller. The models to resolve raw current and voltage into fundamental components and calculate active and reactive power was deployed through VeriStand on a PXIe-8880 controller. Remaining details on the experimental setup is provided in Tab. I. In the control room, VeriStand on Windows, LabVIEW API, InfluxDB 2.0 and Grafana dashboard on Google Chrome are all running on local network computers.

A fifteen minutes measurement duration is shown in the resulting local dashboard in Fig. 5. On the top the user can provide the *Measurement* name and the desired channels with the *Field* selection. This drop down list keeps itself updated with the time-series database. The screen refresh rate can be set on the top right corner of the dashboard. This can go as low as specified in the .ini file but high refresh rate can lead to spotty graphics as the local PC resources could get maxed out. In the example shown in this paper, we chose a 10s refresh cycle as the optimum.

In the first row of the dashboard shown in Fig. 5, user instructions and current status of experiment is provided. It also shows if Veristand execution is happening in real-time through the HP count. When this count is non-zero, it signifies that the target rate is too fast and the execution loops are taking longer than specified in reality. In the second row, three charts show the metrics corresponding to the heat grid. The boiler heating power, which follows a constantly changing set point, is shown on the left side. Temperature measurements at different levels in the hot water storage tank is shown in the center graph and the water flow rate out of the boiler is displayed on the right. All electricity data generated during the experiment is observed with the remaining charts. The two heaters are connected to phase A and B. Their thermostat regulation is shown in the current magnitudes plot. Phase C is connected to the cooking plate and the drill, which can be seen as discontinuous patterns on the blue curve in the current magnitudes plot. The third row shows active power metrics of the experiment. The left shows an instantaneous comparison of the three phases. The right is a stacked plot of the single phase active powers. The left plot on the fourth row shows the grid voltage during the experiment. The reactive power is shown in the bottom right plot which is being substantially contributed by only the electric drill.

The remote viewing dashboard was not presented in the results as the output window looks identical to the local dashboard. The dashboard in Fig. 5, is one of the many possible permutations an operator can make using the vi-


InfluxDB_Experiment_CoSES

2021-02-16 17:26:00 to 2021-02-16 17:40:00 · 10s

Measurement: ExperimentCoSES5 · Field: Calc_EB_P_heat_is + read_EB_S_FW-HC_anlg + read_EB_S_TM-EG + read_...

CoSES local overview Dashboard

To visualize the channels performe:

1. select the experiment from the `Measurement` list
2. select the desired chnnels in the `Field` list (CoSES naming)
3. set a `time` range
4. set a refresh `rate`

System running

Experiment running

Experiment Statistics

_time	_measurement	Absolute_Time	HP_Count
2021-02-16 17:26:01	ExperimentCoSES5	3696337560	0
2021-02-16 17:26:02	ExperimentCoSES5	3696337561	0
2021-02-16 17:26:03	ExperimentCoSES5	3696337562	0
2021-02-16 17:26:04	ExperimentCoSES5	3696337563	0

17:40:00
+01:00 CET

Power Boiler

	min	max	avg
Calc_EB_P_heat_is	7.95 kW	18.08 kW	13.21 kW

Temperature Boiler

	min	max	avg
read_EB_S_TM-EG	30.97 °C	37.43 °C	34.46 °C
read_EB_S_TM-FG	25.08 °C	27.38 °C	26.62 °C
read_EB_S_TM-HC_RL	22.78 °C	25.57 °C	23.93 °C
read_EB_S_TM-HC_VL	35.44 °C	51.57 °C	42.08 °C

Flow Water Boiler

read_EB_S_FW-HC_anlg 9.7 L/min

Current magnitudes

CmagnitudesA Min: 0.00 A Max: 8.83 A Avg: 4.05 A — CmagnitudesB Min: 0.00 A Max: 8.87 A Avg: 6.92 A — CmagnitudesC Min: 0.00 A Max: 6.01 A Avg: 2.65 A

Active Power

P_1_A 0.90 W
P_1_B 1.986 kW
P_1_C -0.15 W

Active Power

P_1_A Min: 0.000 kW Max: 2.049 kW Avg: 0.939 kW — P_1_B Min: 0.000 kW Max: 2.070 kW Avg: 1.606 kW
P_1_C Min: 0.000 kW Max: 1.359 kW Avg: 0.603 kW

Voltage magnitudes

VmagnitudesA Min: 230.46 V Max: 233.57 V Avg: 232.20 V — VmagnitudesB Min: 230.49 V Max: 233.55 V Avg: 232.22 V
VmagnitudesC Min: 231.15 V Max: 234.22 V Avg: 232.85 V

Reactive Power

Q_1_A Min: -1.91 var Max: 5.44 var Avg: 1.35 var — Q_1_B Min: -6.57 var Max: 14.41 var Avg: -2.62 var
Q_1_C Min: -13.26 var Max: 359.81 var Avg: 43.82 var

Fig. 5. Local Grafana Dashboard, showing the metrics of the experiment in Fig. 4.

sualization plugins available in Grafana. we have used the standard table, line graph, bar graph, single statistics and clock plugins. Ideally, multiple dashboard would be setup in a single experiment by the operator and observed as separate tabs in the browser.

V. Conclusion

The CoSES lab at TU Munich with its PHIL and ICT components runs on a distributed control and data acquisition platform which neeed a powerful observation and monitoring software. This paper presented an open-source visulalization tool for CoSES based on IoT paradigms. This tool works alongside the distributed control philosophy of the lab and thus allows the lab to function with the same freedom as before. A live dashboard, viewed from the browser tab of any local PC, is presented which encapsulates the key measurements per experiment for the operator to monitor or analyse. A remote dashboard is also made available with limited history, to access from anywhere over the internet. This tool removes the necessity to intricately understand the control architecture and the NI toolchain to visualize an experiment. This could promote exchange of ideas and accessibility of the lab in the interdisciplinary microgrid research community.

Acknowledgment

The work shown in this paper was supported by the Deutsche Forschungsgemeinschaft (DFG) under the project number 350746631. Vedran S. Perić was additionally supported by the DFG project number 450821044.

References

[1] V. S. Perić et al., *CoSES Laboratory for Combined Energy Systems At TU Munich*, 2020 IEEE Power Energy Society General Meeting (PESGM), Montreal, QC, Canada, 2020, pp. 1-5, doi: 10.1109/PESGM41954.2020.9281442.
[2] N. Bui, A. P. Castellani, P. Casari and M. Zorzi, "The internet of energy: a web-enabled smart grid system," in IEEE Network, vol. 26, no. 4, pp. 39-45, July-August 2012, doi: 10.1109/MNET.2012.6246751.
[3] *VeriStand 2018*, National Instruments, 2018.
[4] *LabVIEW*. National Instruments, 2018
[5] *InfluxDB 2.0*. InfluxData Inc., v2.0.3, 2020. Available: https://docs.influxdata.com/influxdb/v2.0/
[6] *Grafana: The open observability platform*, Grafana Labs, 7.3.7, 2021. Available: https://grafana.com/
[7] *Prometheus Monitoring system & Time series database* , v2.24.1, 2021. Available: https://prometheus.io/
[8] *Telegraf Open Source Server Agent*, InfluxData, 1.17.2, 2021 Available: https://www.influxdata.com/time-series-platform/telegraf/