Analysing Design Approaches for the Power Consumption in Cyber-Physical Systems

Patrizia Sailer[1], Igor Ivkic[1, 2], Markus Tauber[1], Andreas Mauthe[3], Antonios Gouglidis[2]

[1]*University of Applied Sciences Burgenland* - Eisenstadt, Austria
[2]*Lancaster University* - Lancaster, United Kingdom
[3]*University of Koblenz-Landau* - Koblenz, Germany

***Abstract*—The importance of Cyber Physical Systems (CPS) and Internet of Things (IoT) applications is constantly increasing, especially in the context of Industry 4.0. Architectural decisions are crucial not just for performance, security and resilience reasons but also regarding costs and resource usage. In this paper we analyse two of the fundamental approaches to design control loops (i.e. time-driven and event-driven), show how they can be realised and evaluate their power requirements. Through this the design criteria can be extended also considering the optimization of energy related aspects.**

***Keywords*—Cyber-Physical Systems, Internet of Things, Power Consumption, Design Approaches**

I. Introduction

In recent years, advances in computing technologies have had a profound influence in the development of the fourth industrial revolution [1]. The so-called Industry 4.0 is driven by Cyber-Physical Systems (CPS) and the Internet of Things (IoT), which are tightly integrated in and interacting with the physical world [2] [3]. A CPS is formed by components that are capable of communicating with each other, measuring their environment, analysing those measurements and set actions to change environmental conditions. The IoT could be considered as the backbone of a CPS, which is responsible for connecting IoT devices, sensors and actuators [4]. The combination of components, sensors and actuators connected over the IoT enables applications to be created to handle different tasks.

A CPS can have different functionalities, where several metrics (e.g. utilization of hardware components) can be used to measure, resulting in changing the physical environment. An example could be measuring the distance between two objects. A proximity sensor might measure the distance between a person and an automated teller machine (ATM), based on which, if it is less than 0.5 meters, the withdrawal is allowed. Since such a proximity sensor is placed outdoors, it cannot be connected to the power grid in all cases, thus a power source must be found elsewhere. This can be achieved by using an accumulator or battery. As illustrated in this example, CPS may include devices that are constrained by limited energy resources. Hence, it can be pointed out that a CPS must be functional on the one hand to fulfil a certain task, while on the other hand it has to perform well to ensure the longest possible lifetime. To enable a CPS to complete a task, various approaches exist on how to design the control loops required. Each of those design approaches is specialized in certain functionalities and comes along with advantages and disadvantages. However, it is important to note that design approaches of control loops are only one responsible part of power consumption, as other factors such as hardware selection, security or communication protocols used also contribute. Therefore, we have addressed the issue of how power consumption can be reduced and subsequently optimized by choosing the most suitable design approach for control loops.

In order to answer the question regarding the relationship between control loop design approaches and energy consumption, a concept is preparing. To the best of our knowledge, we found related work (see Section II) where design approaches are compared, but none in context with power consumption associated with a comparison of those approaches related to control loops. In this paper we present an experimental setup where we measure the power consumption of relevant hardware components within a CPS, implementing two different design approaches (time-driven and event-driven). This CPS consists of a sensor, an actuator and an IoT framework. The purpose of the IoT framework is to control the way of how new components enter the CPS. Furthermore, it manages the interaction between already participating components to measure and change the physical environment. In this case the Arrowhead Framework is used as representational IoT framework, as it is an open source project written in Java, which can be extended with new functionalities to support more use cases [5]. After the implementation of the prototype, measurements were taken to see if a difference in the power consumption exists between the different approaches for control loops. The motivation is to determine whether power consumption savings can be achieved by specifically implementing design approaches of control loops in CPS.

This paper builds on the work of Sailer [6] where the trade-off between security and power consumption for a CPS was investigated. Therefore, in this paper the experimental setup is extended to enable the comparison of two design approaches, while using a representative IoT framework. The aim of this paper is to measure the power consumption of different design approaches for implementing control loops in a simple use case. The main focus is on the design approaches, therefore other factors, such as the selection of protocols or security are

978-3-903176-32-4

not taken into account in these initial measurements, but are planned for future work. In summary, the contribution of this paper can be structured as follows:

- identification and analyses of two representative design approaches
- development of a prototype using an IoT framework
- preliminary measurement of power consumption based on identified design approaches and developed prototypes

The remainder of this paper is organized as follows: Section II summarizes the related work in the field and points out the background of this paper. Next, in Section III we present an overview and a comparison of two approaches, based on an initial literature review. Furthermore, it explains the implementation of a prototype, followed by the presentation of preliminary measurements and the discussion of those results. Finally, Section IV gives an outline of future work in the field and concludes this paper.

II. Related Work

Many studies about designing a CPS have been conducted. Gacrilescu et al. [7] carried out research on a framework representing an event-driven simulator to configure Programmable System on Chip (PSoC) within a CPS. A classification of a CPS in design (architecture, modelling, tools, simulator, verification), aspects and issues (security, resiliency, reliability, Quality of Services, real-time requirements), applications (such as Vehicular systems, Medical systems, Smart homes, Scheduling, Social network and gaming, Power Grid etc) and challenges has been presented by Khaitan and McCalley [8]. The research of Mo et al. [9] focus on mobile actuators by introducing a new event-driven method, which provides a required level of control accuracy, simultaneously reducing the energy consumption of the actuators, with restricted action delay is guaranteed. However, up to this time research has mainly been conducted in the field of stationary actuators, for example with focus to the actuator control problem, in which the requirements of control accuracy should be fulfilled [10]. If this is taken into account, the control quality can be further increased by reducing the action delays [11] or the packet loss rate [12]. Furthermore, inaccurate system parameters should be identified to prevent them [13].

Further there are studies comparing event-driven and time (scheduled)-driven approaches for CPS. While Albert et al. [14] compare these and propose hybrid systems as solution, Dai et al. [15] concentrate on time-stamped event execution within industrial CPS for the IEC 61499 standard.

Regarding using an IoT framework a huge number of studies exists. Commercially available ones are summarized by Derhamy et al. [16], including Arrowhead [5], AllJoyn [17] or IoTivity [18]. Ammar et al. [19] analysed among other the IoT frameworks SmartThings Samsung [20], AWS IoT Amazon [21] or Azure IoT Microsoft [22] focusing on different properties like architecture components, hardware and software dependencies, access control or communication.

Additionally, there are many studies looking at power measurement in a CPS considering aspects like security, use of Wireless Local Area Network (WLAN) vs. Local Area Network (LAN) or different protocols. Carrara et al. [23] focus on implementing an IoT-based management program to collect temperature and humidity data, while Tauber et al. [24] investigate energy efficiency and performance in a WLAN to identify upper and lower bounds of energy efficiency due to different data flow characteristics. The influence of different security settings on the power consumption was investigated by Sailer et al. [6] by showing the impact of strength of security in regard to the power needed. An application model with the possibility to describe the optimal schedule using mathematical formulas to achieve maximum energy savings was presented by Jiang et al. [25]. Shih et al. [26] present a strategy for achieving power savings in battery-powered devices and compared it with other strategy approaches. Further research on energy consumption were carried out by Tauber et al. [27], [28] and Kansal et al. [29]. Furthermore, there are many papers dealing with the issue related to event vs. time-driven energy performance, but their focus is mainly on hardware issues, monitoring frameworks [30], [31] or are looking at a specific scenario or application areas (e.g. smart grids) [32]. In our extensive research we did not come across research that is directly related to the proposed scheme. Therefore, with our concept we would like to extend the existing studies of comparisons of design approaches in control loops through establishing a connection to the topic of power consumption. For this purpose, the setup from previous work [6] will be extended by control loops, which will be implemented using existing design approaches. The result should be a ranking of the power consumption of the different approaches of control loops in the context of different use cases.

III. Methodology approach

In this section we present the experimental setup for the power consumption measurements within a CPS. To test our concept, a CPS is built with the minimum requirements, consisting of a single sensor, actuator and controller (IoT framework). This means that one participant in this CPS measures the physical environment, another changes it, while the third is responsible for controlling the entire interaction between the sensor and actuator. This setup represents the smallest setup of interacting components that are needed to form a CPS, measure and change the physical world. After careful consideration, it was decided that this CPS was sufficient to show whether a difference is made by selecting a different design approach. It is also possible to extend the CPS with additional components to gain complexity, ensuring scalability. A measurement would be required for each additional component, but the measurement method would not change, regardless of the number of components within a CPS. In summary, we have chosen this simple CPS to proof the feasibility of the concept in the smallest possible CPS.

A. Use Case

In principle, a CPS consists of several components interacting with each other, measuring and changing the environment

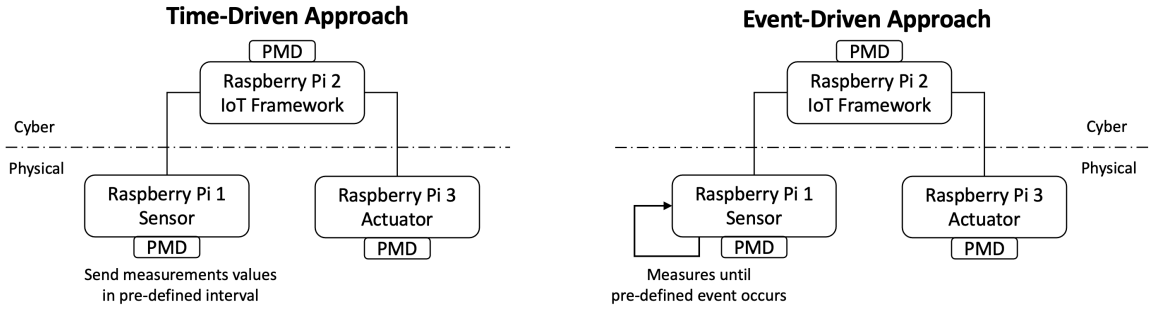

Fig. 1. Prototype architecture consists of Raspberry Pis and power measurement devices (PMD) to compare the time-driven and event-driven design approach.

conditions (if necessary). The minimum possible implementation of this definition would be to build a CPS with one component including a sensor, one component including an actuator and a IoT framework. The IoT framework manages the communication and interaction of the sensor and the actuator. For this purpose, authorisation rules are specified to avoid connections between invalid components. To show any difference in the power consumption between design approaches of control loops, we have implemented a use case with minimal requirements using two different design approaches. Furthermore, Table I provides an overview of possible actions to be performed in a CPS. For this purpose, a sensor and an actuator were identified, followed by the functionality they should perform.

These use cases are scenarios of real-world situations. Since it is difficult to obtain perfect measurement conditions in the real world, these use cases are simulated. This means the sensor sends pre-defined data to the actuator, which simulates an action, if required. Therefore, it is possible to repeat and adjust the measurements to make the results comparable.

However, if sensors and actuators were actually used for the measurements to compare the results, a laboratory experiment would have to be carried out. A laboratory experiment offers the possibility to specify perfect conditions for measurements and to change them if necessary. To explain this with an example, Use Case 1 from Table I is taken. In a real environment, it would not be possible to influence the measurement values and the actions of the actuator to keep exactly the same schedule twice. Measuring the temperature in a room for one hour, it is probable to have different temperature values in the following hour. Even if the measurements are taken at the same time, the conditions are not identical because the experimental setups cannot be set up in the same location. It is highly possible that an experiment set up is placed closer to the air-conditioning system. This was only one example of the difficulty of comparing measurements in the real world. The purpose of this paper is to show if there is an impact on power consumption even in a minimal CPS by comparing only two design approaches. These results can later be extended by a range of factors, such as WLAN instead of LAN, other protocols or different security settings, to generate an insight into how these could be used in more complex environments.

TABLE I
OVERVIEW OF REPRESENTATIVE USE CASES

	Actuator	**Sensor**	**Action**
Use Case 1	Air Condition	Temperature Sensor Inside	Switch the air conditioning system on/off by reaching certain temperatures
Use Case 2	Heating System	Temperature Sensor Outside	Switch the heating system on/off by reaching certain temperatures outside
Use Case 3	Fan/ Humidifier	Humiditiy Sensor	Switch the fan/humidifier on/off by reaching certain limits
Use Case 4	Camera	Motion Sensor	Video surveillance of e.g. pets at home

B. Design Approaches for Control Loop

In an initial literature review we identified two different design approaches, time-driven and event-driven, which offer different procedures since tasks are processed using various strategies. Managing a CPS requires the use of design approaches for control loops to ensure the CPS is doing exactly what it is supposed to do. This means the process for handling tasks is predefined in terms of when a component should do what. In order to manage such a system, appropriate rules for these interactions need to be designed and established. The challenge in designing control loops of CPS is to specify the expected behaviour of the computing components and their impact on the physical environment. Therefore, the programming language has to be considered, since it has to provide an integration of time-driven computation and event-driven computation to enable an asynchronous dynamic [27].

In Fig. 1 two design approaches for control loops are compared. The left one is the time-driven approach and the right one is the event-driven approach. In the following, we will explain the two selected approaches in more detail.

The time-driven approach focuses on a time frame and the execution of different tasks at certain predefined times, which are repeating. If this approach is considered with respect to a real-time system, it can be claimed such a system has a deterministic behaviour. In other words, it can be determined in advance how this system has to behave. As a result, it is possible to plan how the system or subsystem will be executed, since this takes place periodically. When planning, it is important the engineer is aware of that tasks that share a

resource cannot be executed simultaneously. Another important issue is to distinguish whether a task is time critical (hard time condition) or not (soft time condition). Fig. 1 shows the process of time-driven approach, in which the actuator requests the values of the sensor in a predefined time interval to detect if an action is needed. This implies each measured value is sent to the actuator via the IoT framework. To make such a system work, the following four requirements must be fulfilled:

- **Timeliness** - correctness of the system is assumed by implementing the logic for hard time conditions, since result and time of the result creation are important
- **Reliability** - It is essential to ensure there is an emergency solution in case of failure to prevent loss of life, irreparable damage or material damage.
- **Availability** - It is expected the system will be operational at the specified time.
- **Predictability** - All actions to be performed by time interval must be known beforehand. This is necessary to ensure trouble-free action scheduling by taking all necessary precautions.

In contrast, with the event-driven approach, no predefined interval exists for sending the measurement data, as this is only transmitted from the sensor to the actuator when an event occurs to activate the system. One characteristic of this approach is the possibility that multiple events can occur at the same time. In such a scenario the next upcoming event is chosen for execution. Often event queues are used for this purpose, which, if configured incorrectly, represent a high risk. A further problem of event-driven systems can be the time delay when several events are processed, which is shown in Fig. 2. As a result, such systems can exhibit a jitter, which means that a desired signal is delayed in time causing the process to be delayed.

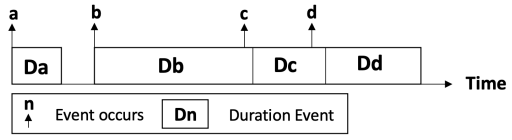

Fig. 2. Timeline indicator when processing various events, showing the time of occurrence of an event, the duration and the end time.

This comparison shows how important it is to identify the purpose of a CPS before creating it. Not every approach is suitable for each use case. Since both design approaches for control loops can be possible solutions for the implementation of use cases, it should be investigated which of them consumes less power. Especially in such a situation, power consumption could be a criterion for deciding which approach to choose.

C. Experimental Setup

In this section we present the experimental setup for the power consumption measurements. Although choice of hardware can already have an influence on power consumption, we selected same devices, equipped with identical operating system, software and services. Therefore, to test our concept, we chose three Raspberry Pis 3 Model B+, which are used to deploy a CPS consisting of a sensor, an actuator and the IoT framework Arrowhead. As operation system we chose Rasbian Stretch Lite to reduce unnecessary factors like, e.g. resources used for running Graphical User Interface (GUI). To avoid network latency (or any unnecessary external influences) the devices were connected via a switch in a LAN environment. Each Raspberry Pi is equipped with a power measurement device (PMD) to determine the energy consumption. We decided to use the PiLogger One [33] as PMD, which can be attached directly to a Raspberry Pi and will be connected via Inter-Integrated Circuit (I^2C) bus. An advantage of the PiLogger One is the possibility to use time precise intervals for the measurements, as it offers its own time base. The measured values are stored and can be evaluated afterwards. Based on Fig. 1 it is visible which Raspberry Pi assumes which functionality in the experimental setup.

As shown in Fig. 1 the sensor and the actuator are connected over an IoT framework. The Arrowhead Framework [5] is used as a representative IoT framework to facilitate the creation of local automation clouds, which includes devices, various application-specific systems and services to perform the automation tasks and provide a boundary to the open internet and the outside activities. This way Arrowhead is enabling local (on-site or private), real time performance and security, paired with simple and cheap engineering, while enabling scalability through multi-cloud interactions. Further it provides an architecture composed of three mandatory core systems: Service Registry System, Orchestration System and Authorisation System.

In this use case the components sensor and actuator are forming a CPS by using the Arrowhead Framework to become part of it and controlling the interactions. Once the on-boarding procedure has been completed successfully and the two IoT-devices are part of the CPS their main purpose is to measure the temperature of a physical room and to regulate its temperature (if necessary). The experimental setup emulates a network in which data is collected with a sensor and is sent via an Ethernet switch to the Arrowhead local cloud by using the the Hypertext Transfer Protocol (HTTP) protocol. The components are explained in detail below:

- Actuator simulates an air-conditioning system to cool down a physical room if necessary.
- Sensor measures temperature of a physical environment.
- Service Registry is responsible to register new services.
- Authorisation System is responsible for authorisation rules, which regulate, which components are allowed to interact with each other within the CPS. Every time a component sends a request to the Arrowhead Framework, it verifies if this component is allowed to do so.
- Orchestration System is responsible for handling any request by any component. Every time a component needs information about the CPS, the orchestration handles the request and make sure that the right information is passed to the requesting component.

It is necessary to deploy the Arrowhead Core Systems in a local cloud environment and register the sensor and actuator components before this CPS can work. After these steps have been successfully performed, it is possible to execute the CPS using each of the presented design approaches. The time-driven approach uses a sensor to measure the temperature at defined intervals, e.g. every ten minutes and send this value to the Arrowhead Framework. Within the framework, the value is first passed to the Orchestration System, which requests the Service Registry to verify whether there is a suitable actuator. After successful response, Authorisation System is used to check whether the sensor is allowed to communicate with the matching actuator. If this is confirmed, the value is forwarded to the actuator, in this case the air condition, who subsequently decides whether an action has to be set or not. This means that this process is carried out continuously, irrespective of the measured temperature value. On the contrary to this, the event-driven approach offers the possibility of a prior check whether the measured temperature value has been reached or exceeded a predefined limit. If the value is below this limit, no action is required and the measurement is repeated. This action is continued until the value is above the limit, which means the actuator has to be informed about performing an appropriate action. In this case, the value is sent to the Arrowhead Framework and the same procedure is performed as in the time-driven approach.

To reinforce our presented concept, we have made preliminary measurements including the two presented approaches. Fig. 1 shows the experimental setup for both implementations including the comparison of the approaches. A total of 22 test runs with 100 measurements each were carried out, whereas the measured values were predefined in order to be able to reproduce each test run under the same conditions. In each of the 100 measurements per test run, a predefined value was read from an array containing 50 values below and 50 above a predefined limit, which was set at the value 25. Therefore, measurements 1 to 50 do not activate the actuator, in contrast to measurements 51 to 100. The first test run was a warm-up and was not taken into account for the exploitation. Furthermore, all unnecessary services on the Raspberry Pis were turned off to avoid unnecessary power consumption.

D. Evaluation

The results of the measurements are summarized in Fig. 3, whereby the time-driven approach is shown in blue and the event-driven approach in green. The results show a significant difference between the design approaches for the devices representing the sensor and actuator. This is because the sensor in the time-driven approach establishes a connection to the IoT framework for each measurement in order to search for the actuator. In the event-driven approach this connection was only established when the specified value achieved a defined limit. The test values were chosen regularly, whereby the first half was below the limit and the second half was above it. Further observation of the test runs showed that a test run in the time-driven approach needed an average of 125 seconds to take 100 measurements, while the event-driven approach needed only 114 seconds, showing a time savings in addition to the power savings.

In contrast, the measurements of the device with the Arrowhead Framework gave a similar result, observing the time-driven approach needs more power. However, it also shows that in the event driven approach the framework has a high dispersion, which implies that parts of the energy consumption of both approaches overlap. At this stage, we have no clear indication of this behaviour. Moreover, it was noticed in all cases that the initial test runs consumed more power, which can be interpreted as a result of the warm-up phase. Therefore, a longer warm-up phase is planned for further measurements. Summarising, these first measurements show that choosing a specific design approach has direct impact on the resulting power consumption. Our results show that the event-driven approach consumed approximately 7% less power, compared to the time-driven approach.

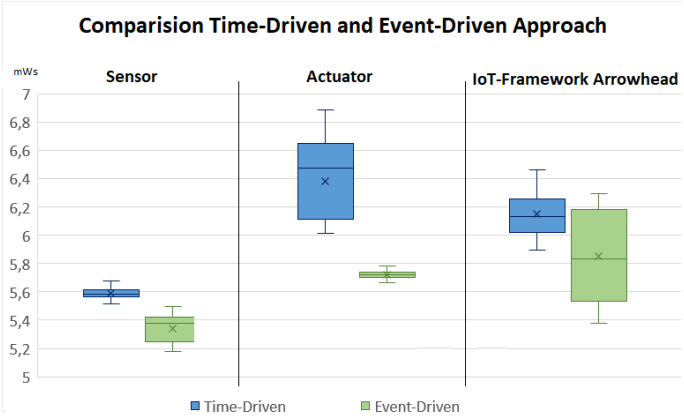

Fig. 3. Overview of the power consumption in Milliwattseconds (mWs) of the testruns performed. To compare the time-driven and event-driven approach the components Sensor, Actuator and Arrowhead Framework were measured.

IV. Conclusion and Future Work

In this paper, we show the impact of the selection of the design approach of control loops in a CPS on power consumption. Therefore, we identified two design approaches in an initial literature review, which were explained in detail. In addition to that, we describe the minimum requirements to build and measure the power consumption of design approaches of control loops of a CPS. Hence, we performed power consumption measurements with the developed prototype using Raspberry Pi 3 Model B+ as devices, each equipped with a PiLogger One as PMD. Furthermore, we presented initial measurements of the power consumption by using the time-driven and event-driven approach to design control loops. Finally, our experimental study indicated that the event-driven approach consumes 7% less power to process the same task comparing with the time-driven approach.

The main contribution of this paper is the initial measurements of the impact of power consumption on the selection of design approaches for control loops. A difference in power

consumption by considering different design approaches has been shown. This will be enhanced in future work by addressing further existing design approaches and implementing other use cases to make additional measurements. Since the results of the initial measurements of the IoT framework were partly overlapping, an aspect of swinging in control loops will be taken into account for future measurements, as well as the extension of the warm-up phase. Furthermore, other IoT devices, such as Arduino, can be used in order to examine the connection to power consumption from the hardware side as well. In addition, a more detailed investigation by extending the protocols, security and algorithms will be carried out in continuing work. Furthermore, a model for power measurements should be developed in order to be generalized for further research and to be used in industry.

In addition, we are considering how to conceptually use the different design approaches of control loops to create a dynamic and autonomic system. Such a system will be able to identify the current situation regarding power consumption and switch to the most suitable design approach for control loops. For this purpose, all impacts on power consumption must first be identified to enable the control loops to be adjusted accordingly. To create a self-adaptable system, tools such as the interactive framework Monitor-Analysis-Plan-Execute over a shared Knowledge (MAPE-K) [34] can be used.

ACKNOWLEDGMENT

The research has been carried out in the context of the EFRE project MIT 4.0 (FE02), funded by IWB-EFRE 2014-2020 coordinated by Forschung Burgenland GmbH.

REFERENCES

[1] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial cyberphysical systems: A backbone of the fourth industrial revolution," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 6–16, 2017.

[2] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *System Sciences (HICSS), 2016 49th Hawaii International Conference*, pp. 3928–3937, 2016.

[3] F. Almada-Lobo, "The industry 4.0 revolution and the future of manufacturing execution systems (mes)," in *Innovation Management*, vol. 3(4), pp. 16–21, 2016.

[4] L. Esterle and R. Grosu, "Cyber-physical systems: challenge of the 21st century," in *Elektrotechnik und Informationstechnik*, vol. 133(7), pp. 299–303, 2016.

[5] J. Delsing, *Iot automation: Arrowhead framework*. CRC Press, 2017.

[6] P. Sailer, C. Schmittner, and M. Tauber, "Managing the trade-off between security and power consumption for smart cps-iot networks," vol. 119, pp. 23–24, ERCIM News, 2019.

[7] M. Gavrilescu, G. Magureanu, D. Pescaru, and A. Doboli, "A simulation framework for psoc based cyber physical systems," in *2010 International Joint Conference on Computational Cybernetics and Technical Informatics*, pp. 137–142, 2010.

[8] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, 2014.

[9] L. Mo, P. You, X. Cao, Y. Song, and A. Kritikakou, "Event-driven joint mobile actuators scheduling and control in cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 11, pp. 5877–5891, 2019.

[10] L.-W. Yeh, C.-Y. Lu, C.-W. Kou, Y.-C. Tseng, and C.-W. Yi, "Autonomous light control by wireless sensor and actuator networks," *IEEE Sensors Journal*, vol. 10, no. 6, pp. 1029–1041, 2010.

[11] L. Mo, X. Cao, Y. Song, and A. Kritikakou, "Distributed node coordination for real-time energy-constrained control in wireless sensor and actuator networks," *IEEE internet of things journal*, vol. 5, no. 5, pp. 4151–4163, 2018.

[12] X. Cao, J. Chen, Y. Xiao, and Y. Sun, "Building-environment control with wireless sensor and actuator networks: Centralized versus distributed," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3596–3605, 2009.

[13] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, "Distributed collaborative control for industrial automation with wireless sensor and actuator networks," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 12, pp. 4219–4230, 2010.

[14] A. Albert *et al.*, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," *Embedded world*, vol. 2004, pp. 235–252, 2004.

[15] W. Dai, V. Vyatkin, C. Pang, and J. H. Christensen, "Time-stamped event based execution semantics for industrial cyber-physical systems," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 1263–1268, IEEE, 2015.

[16] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the internet of things," in *In IEEE International Conference on Emerging Technologies and Factory Automation: 08/09/2015-11/09/2015*, IEEE Communications Society, 2015.

[17] A. Alliance, "Alljoyn framework. linux foundation collaborative projects," 2016.

[18] L. F. C. Projects, "Iotivity website," 2019.

[19] M. Ammar, G. Russello, and B. Crispo, "Internet of things: A survey on the security of iot frameworks," vol. 38, pp. 8–27, Journal of Information Security and Applications, 2018.

[20] SmartThings, "Smartthings classic developer documentation," 2019.

[21] Amazon, "Aws iot framework," 2019.

[22] Microsoft, "Azure iot solution accelerators," 2019.

[23] M. Carrara, P. Catania, G. L. Re, M. Ortolani, and M. Vallone, "An innovative system for vineyard management in sicily," in *Journal of Agricultural Engineering*, vol. 41(1), pp. 13–18, 2010.

[24] M. Tauber, S. N. Bhatti, and Y. Yu, "Application level energy and performance measurements in a wireless lan," IEEE/Green Computing and Communications (GreenCom), 2011.

[25] W. Jiang, G. Xiong, and X. Ding, "Energy-saving service scheduling for low-end cyber-physical systems," in *2008 The 9th International Conference for Young Computer Scientists*, pp. 1064–1069, IEEE, 2008.

[26] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Proceedings of the 8th annual international conference on Mobile computing and networking*, pp. 160–171, 2002.

[27] M. Tauber, S. N. Bhatti, and Y. Yu, "Towards energy-awareness in managing wireless lan applications," pp. 453–461, IEEE/Network Operations and Management Symposium (NOMS), 2012.

[28] M. Tauber and S. N. Bhatti, "The effect of the 802.11 power save mechanism (psm) on energy efficiency and performance during system activity," pp. 573–580, IEEE/Green Computing and Communications (GreenCom), 2012.

[29] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," pp. 39–50, Proceedings of the 1sCM symposium on Cloud computing, 2010.

[30] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang, "Web-of-things framework for cyber–physical systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 9, pp. 905–923, 2011.

[31] F. Makedon, Z. Le, H. Huang, E. Becker, and D. Kosmopoulos, "An event driven framework for assistive cps environments," *SIGBED Rev.*, vol. 6, July 2009.

[32] X. Yu, C. Cecati, T. Dillon, and M. G. Simões, "The new frontier of smart grids," *IEEE Industrial Electronics Magazine*, vol. 5, no. 3, pp. 49–63, 2011.

[33] G. Weiß-Engel, "Manual pilogger one," 2018.

[34] T. A. Nguyen, M. Aiello, T. Yonezawa, and K. Tei, "A self-healing framework for online sensor data," in *2015 IEEE International Conference on Autonomic Computing*, pp. 295–300, IEEE, 2015.