Integrating External Data Sources into Internet of Things Architectures for Weather and Environmental Monitoring in Former Mining Areas

Jonah Windolph[1] Robert Wehlitz[2] Theo Zschörnig[3] Bogdan Franczyk[4]

Abstract: The management of former mining areas requires knowledge about local weather and environmental conditions in order to ensure the compliance with governmental regulations. Since mining areas are typically of large geographic scale, equipping them with sensors is a challenge at high cost. One approach to keep these low is the enrichment and in some cases substitution of sensor data utilizing external sources, such as weather services. Against this background, this paper describes requirements for a solution that allows the use of external data sources to enrich or substitute sensor data for historic and real-time analysis. An architecture that supports this approach is presented. The architecture is implemented as a proof of concept for the purpose of demonstrating its abilities by the use case of land use regulations.

Keywords: Internet of Things; System Architecture; Microservices; System Integration; External Data Sources

1 Introduction

Even with more and more countries deciding to phase out of coal-fired power generation in order to meet climate goals (cf. [En20]), the management of former mining areas remains an important task. It requires detailed knowledge of the environmental conditions at these sites in order to ensure the compliance with governmental regulations that are in place regularly due to the geographic conditions in these areas. Regulations often depend on the current environmental situation. For example, using an unpaved road with heavy equipment might be forbidden if the road is wet after it has rained. Internet of Things (IoT) approaches are able to support the management of former mining sites by providing real-time access to and analysis of sensor data. Past scientific research has mainly been focused on enabling access to devices or providing analytics or automation capabilities. Little attention has been

[1] Institute for Applied Informatics (InfAI), Competence Center Smart Energy Systems, Goerdelerring 9, 04109 Leipzig, Germany, windolph@infai.org
[2] Institute for Applied Informatics (InfAI), Competence Center Smart Energy Systems, Goerdelerring 9, 04109 Leipzig, Germany, wehlitz@infai.org
[3] Institute for Applied Informatics (InfAI), Competence Center Smart Energy Systems, Goerdelerring 9, 04109 Leipzig, Germany, zschoernig@infai.org
[4] Leipzig University, Informations Systems Institute, Grimmaische Str. 12, 04109 Leipzig, Germany, franczyk@wifa.uni-leipzig.de; Wrocław University of Economics, Business Informatics Institute, ul. Komandorska 118-120, 53-345 Wrocław, Poland

given to integrating external data sources, such as public web services. The enrichment or substitution (if possible) of sensor data with data from external sources simplifies environmental monitoring of large areas immensely, since equipping them with sensors can be expensive (e. g. a single air quality sensor considered to be low cost still costs a few hundred euros (cf. [Al17])).

Therefore, the approach of integrating external data sources into IoT architectures is investigated in this paper. It is structured as follows: In the second section we describe related work. Requirements for a solution that enables the integration of external data sources are defined in the third section. In the fourth section we describe our proposed solution and demonstrate its abilities in section five. In the last section we summarize our findings and outline further research opportunities.

2 Related Work

In this section, preliminary work related to the topic of integrating web services into IoT architectures is presented.

Senožetnik et al. [Se18] describe an architecture they use to retrieve groundwater data from different sources for real-time queries and analysis. Their architecture consists of the components *retriever*, *watchdog*, *collector* and *API management*. *Retrievers* collect the data from their corresponding source and transform them into a unified format. The *watchdog* provides monitoring capabilities and ensures the restart of retrievers in case of errors. The *collector* collects the data from various retrievers and applies rule-based pre-processing before storing the data and making them available for querying. Finally, the *API management* allows for user and access management.

Kwon et al. [KOJ19] propose an IoT framework that is capable of handling sensor and externally available data. Measurements are collected by devices themselves or through data source-specific *external data collectors* that use the HTTP protocol to publish the data to a message broker. A *data collector manager* ensures error recovery by restarting failed data collectors. The collected data is saved in an OpenTSDB instance or further analyzed by *enrichers*, depending on a routing table in the message broker. All components run inside a Kubernetes cluster.

Heideker et al. [He20] introduce an IoT solution based on FIWARE to increase crop yield and decrease cost by optimizing water usage of agriculture in arid regions. In order to improve their computations they also include data from weather forecast services and historic records of agriculture yield by implementing custom mappers for the NGSI API, FIWARE uses. Terroso-Saenz et al. [Te19] follow a similar approach for energy consumption prediction of heating, ventilation and air conditioning systems based on sensor and weather data.

Luckner et al. [Lu20] propose an IoT architecture for a smart city IoT application. In order to integrate external data sources they implement custom data sources for Apache Flume. For less frequently updated data such as public transport timetables they use custom components that pull data from these sources regularly. Following the lambda architecture, their solution provides real-time analysis capabilities using a speed layer and analysis of historic data using a batch layer. Contrary to the lambda architecture they explicitly do not use a serving layer to make data available for downstream applications or users, but instead define a new layer they call *exposition layer* that provides application specific interfaces.

Reviewing the related literature shows that approaches for integrating external data sources into IoT architectures exist. However, existing approaches either require all sensors to send data via one specific protocol or rely on IT ecosystems like FIWARE that are very complex (cf. [We17]). Regarding analytics capabilities, previous approaches are based on the lambda architecture that requires two separate processing logics for historic and real-time analytics, resulting in higher complexity compared to the kappa architecture that only uses a single layer for analytics (cf. [St16]). Therefore, we present our own solution that eases integration of sensors (that may provide data via diverse protocols) and external data sources, does not rely on complex ecosystems and uses the kappa architecture for data analytics.

3 Solution Requirements

In this section, requirements for integrating external data sources into IoT architectures are defined as guidance for the implementation. These requirements arose in the ongoing research project Smart Regional Development Infrastructure (SARDINE), in which a novel data platform for renaturation projects in former mining areas is developed.[5] Part of the project is the automated detection of weather conditions that affect governmental regulations. Since no weather sensors are deployed in the field at the time, integrating data from external sources is required. Our requirements are defined as follows:

(R1): *Scalability*: With a lot of countries increasing their efforts to make data publicly available (cf. [Wo18]), the solution needs to be able to scale with both the amount of external data that could be integrated and an increasing number of potential users (i. e. when the data is meant to be publicly accessible).

(R2): *Expandability*: Solutions should provide a way to integrate new data sources. This is necessary because external data sources are typically focused on a specific kind of data (e. g. weather data), but the management of former mining areas requires knowledge of diverse conditions. In order to be able to add support for new data sources, developers require assistance (e. g. in the form of a software development kit (SDK)).

[5] `https://infai.org/sardine/`, accessed 12.04.2021

(R3): *Error recovery*: Solutions should provide an ability to resume importing data after a failure. External data sources like weather services often provide huge data sets (e. g. historic records of weather data for whole countries). If the integration of external data sources fails (e. g. due to a hardware fault or an unforeseen programming error) reloading all data unnecessarily stresses the external data sources and own computation resources. Thus, a way to recover from errors is required.

(R4): *Configurability*: The inclusion of external data sources should be reconfigurable in order to meet user requirements. Integrating external data sources in their entirety might not be required for the management of former mining areas. Therefore, it should be possible to configure integration processes in order to only integrate relevant data.

(R5): *User friendly management*: Users without deep technical knowledge should be able to control integration processes and use configuration capabilities as described in (R4).

(R6): *Sharing capabilities & access management*: After providing a way to integrate an external data source, users should be able to allow other users to use their implementation in order to avoid multiple implementations for the same data source. On the other hand, users might want to specifically use an integration for themselves (e. g. when private data is involved). Thus, fine-grained access management is required.

(R7): *Interoperability*: Integrated data from external data sources should be usable in the same ways as device data. Both data types need to be used jointly. We conceive this to be a key requirement, because it allows to flexibly vary the level of substitution of sensors with data from external sources (e. g. an external data source needs to be replaced with a sensor for more detailed monitoring or a sensor needs to be substituted in order to decrease cost).

4 Solution Proposal

In this section, we describe an IoT architecture that supports the integration of external data sources. The architecture is shown in Fig. 1. It is based on the architecture previously described in Zschörnig et al. [Zs18] that has been evaluated in Zschörnig et al. [Zs20a] and Zschörnig et al. [Zs20b]. *Sensors* are connected to an *inbound data broker* that receives sensor measurements. If sensors are unable to connect to the inbound data broker themselves, a *device connector* is used to ensure connectivity of the device. This is typically required for sensors in wireless networks that are not directly connected to the internet, such as Zigbee or Z-Wave. *Platform connectors* use the stored metadata from the *device management* to consume the raw sensor data, transform them into a platform-wide unified format and produce the transformed data in a *streaming platform* that distributes the data to further system components, e.g. analytics or storage services. Besides holding metadata of

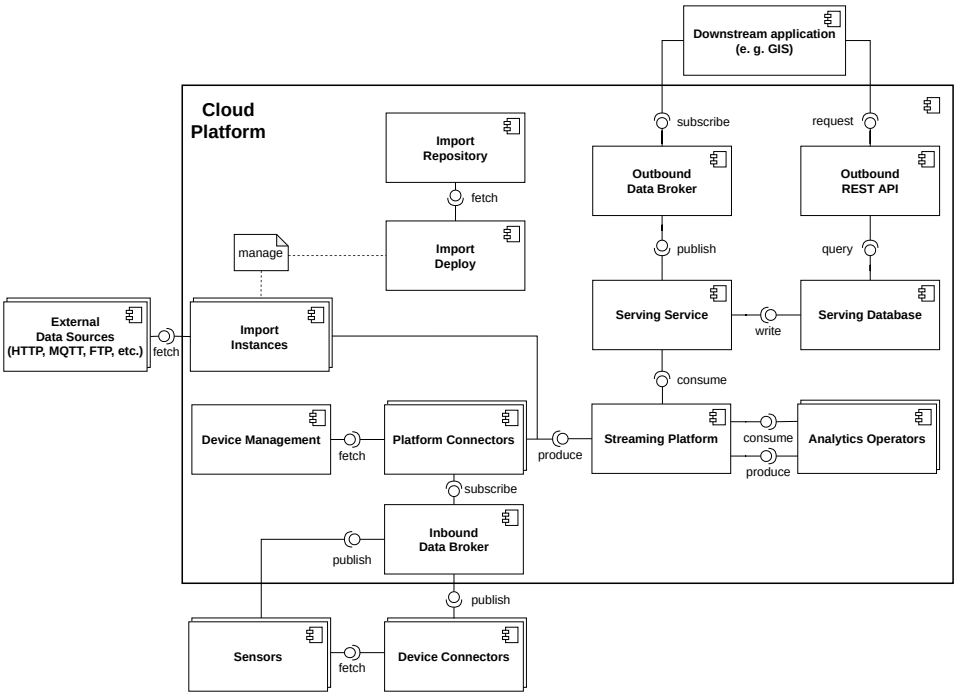

Fig. 1: Proposed architecture

different device types, the device management also provides information of which devices are currently connected to the platform and offers sharing capabilities among users (i. e. allowing other users to access sensor data). *Analytics operators* connect to the streaming platform as well in order to receive sensor data and perform analytics tasks. Analytics operators can be composed into complex analytics pipelines and are deployed automatically on user request. The results are then pushed back to the streaming platform again. Since the architecture is based on the kappa architecture, no additional analytics layer for batch processing is required. If necessary, historical data can be played back from the streaming platform. The *serving service* consumes messages from the streaming platform in order to provide real-time access via the *outbound message broker* or to store the data in a *serving database*. The *outbound REST API* is used to access historic data from the serving database. *Downstream applications* (e. g. a geographic information system (GIS)) can consume data from the outbound data broker or the outbound REST API. The individual components offer REST endpoints that can be used comfortably via an overarching *web user interface*.

External data sources may provide the data in various different formats and make them available via a variety of different protocols. Therefore, integrating these data sources requires data source-specific implementations that we refer to as *imports*.

The *import repository* is responsible for storing metadata of implemented imports. It holds information on how an import can be instantiated, in which ways it can be configured and which data it provides. It also grants sharing capabilities among users with fine-grained access management mechanisms (R6). The *import deploy* component is responsible for managing instances of imports stored in the import repository. It provides an interface with operations to create, read, update and delete (CRUD) import instances.

Imports can be packaged using container technology in order to simplify deployment (R5) and scalability (R1). Container management software usually allows for automatic restarts of failed containers, thus fulfilling the requirement of error recovery (R3). Following the microservice principle, the new components are maintained as loosely-coupled applications. We opted for this principle because it eases scalability (cf. [Dr17]) and provides great flexibility in terms of development and deployment (cf. [Ch18]). Both, the import repository and import deploy component provide interfaces that can be integrated in the overarching web user interface for user-friendly import management (R5).

Developer support is supplied by a library that eases the implementation of new imports, simplifying expandability (R2). The library handles the publication of the data on the streaming platform in the same platform-wide unified format that is already in use for sensor data. It also parses user supplied configuration (R4) and improves error recovery capabilities (R3) by allowing access to already imported data, hence preventing possible duplication. After the data is published on the streaming platform it can be handled in the same way as data from sensors since it uses the same format. Thus, it is possible to use external data sources in analytics operators or to make it available for downstream applications (R7).

5 Demonstration

In former mining areas, land use is typically regulated due to the geographical circumstances in these areas. For example, driving on an unpaved road with heavy equipment (such as excavators) might only be permitted, if it has not rained in the last 24 hours. Since installing and maintaining weather sensors might be expensive, using data from a public weather service is a more practicable approach.

In order to check precipitation in the area of question, we implemented the necessary system components (cf. Fig. 1) as a proof of concept.[6] Both, the import repository and import deployment component are implemented in Go[7] and use a MongoDB[8] instance as persistent data storage. The import deployment component provides the corresponding Docker image

[6] All source code is available at `https://github.com/senergy-platform`
[7] `https://golang.org/`, accessed 26.03.2021
[8] `https://www.mongodb.com/`, accessed 26.03.2021

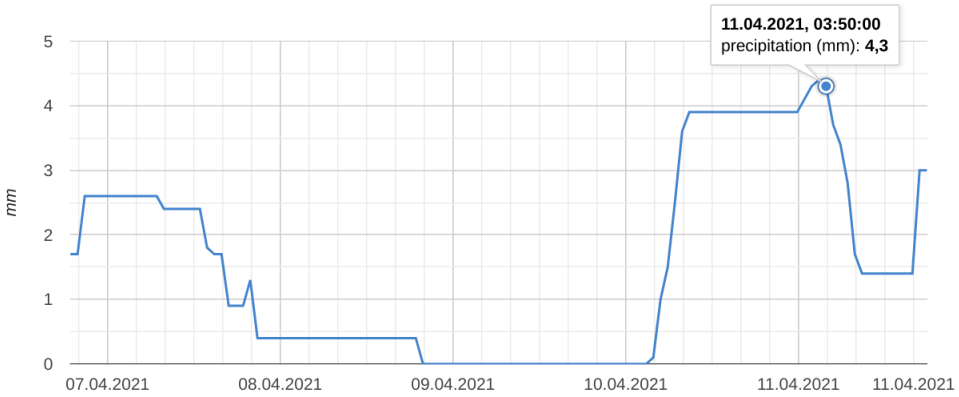

Fig. 2: Integrated precipitation data (24 hour sum) visualized in the web user interface

of an import either directly to a Docker[9] host or to a Rancher[10]/Kubernetes[11] cluster. Using the import library (developed in python), an import for precipitation data from the German weather service was implemented. The German weather service provides precipitation data with a high geographic resolution of 1 km² by combining ombrometers with radar data (cf. [DW20]). Their service offers aggregated precipitation amounts over different time intervals. For this specific use case 24 hour precipitation sums were chosen that are updated hourly. The implemented import can be configured by supplying areas of interest. The import is registered at the import repository and deployed with the import deploy component using the area in question. Using the serving service, access to the imported data via the outbound REST API is enabled, the web user interface is used to visualize the imported data. As shown in Fig. 2, using the unpaved road with heavy equipment in a former mining area south of Leipzig (Germany) on 11th April 2021 is not permitted, since it rained less than 24 hours earlier.

Furthermore, the integrated data can also be used by other downstream applications. Fig. 3 shows how a GIS uses the precipitation data as an additional data layer. In this example, weather-dependent land use regulations are highlighted in green while the current precipitation levels are shown in shades of blue, hence allowing for real-time evaluation of regulations.

This demonstration shows that our solution is capable of providing current environmental data in order to evaluate the status of governmental regulations. Since the solution allows for expandability (R2), it could be used in other scenarios that require environmental data. The example of land use regulations based on weather conditions illustrates the relevance of the subject matter.

[9] `https://www.docker.com/`
[10] `https://rancher.com/`, accessed 26.03.2021
[11] `https://kubernetes.io/`, accessed 06.04.2021

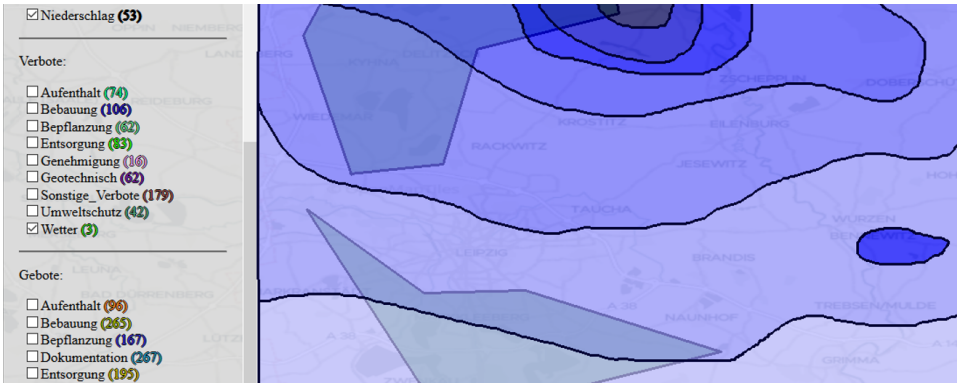

Fig. 3: A GIS displaying weather based regulations and current precipitation levels

6 Conclusion

In this paper, we described how integrating external data sources into IoT architectures supports weather and environmental monitoring in former mining areas. We defined a list of requirements a solution should be capable of in this context and proposed an architecture that tackles this goal. We implemented the necessary system components as open source software following the microservice approach and demonstrated the abilities of our solution in a real-world use case. Our solution is scalable, can easily be expanded to support new external data sources, supports error recovery, allows instance level configurability, provides end-users without deep technical knowledge with a web user interface for comfortable usage and offers sharing mechanisms among users.

While we were able to demonstrate the integration of external data sources into our architecture, some open questions remain. For example, unit conversions are not covered yet by our solution. Downstream applications depend on the unit external data sources provide. Support for unit conversion could be added using an analytics pipeline as described in section 4. Furthermore, data from external sources is usually not as geographically precise as data from own sensors. Enhancing geographic precision in the most relevant areas while maintaining a less precise overview of surrounding areas might be possible by combined analytics of both data sources. It is unclear however, how many sensors would be needed for an actual improvement and (depending on the geographic circumstances) where exactly sensors would need to be deployed. Future research might address these issues and develop generalized rules for effective sensor placement. Additionally, it would be interesting to integrate sensors, data from external sources and geographical models of former mining areas in order to create a digital twin. This model could be used to simulate scenarios (cf. [GV17]) such as flooding in order to gain new insights, e. g. where extend efforts in disaster prevention. Moreover, administration processes (e. g. requests for information on flooding risks based on current weather conditions) could be enhanced by combined analytics of sensor data and data from external sources in order to (partly) automate such processes.

Acknowledgements

The work presented in this paper is funded by the Free State of Saxony (SAB 100400223).

References

[Al17] Alvarez-Campana, M.; López, G.; Vázquez, E.; Villagrá, V.; Berrocal, J.: Smart CEI Moncloa: An IoT-Based Platform for People Flow and Environmental Monitoring on a Smart University Campus. Sensors 17/12, p. 2856, Dec. 8, 2017, ISSN: 1424-8220.

[Ch18] Chen, L.: Microservices: Architecting for Continuous Delivery and DevOps. In: 2018 IEEE International Conference on Software Architecture (ICSA). 2018 IEEE International Conference on Software Architecture (ICSA). IEEE, Seattle, WA, pp. 39–397, Apr. 2018, ISBN: 978-1-5386-6398-1.

[Dr17] Dragoni, N.; Lanese, I.; Larsen, S. T.; Mazzara, M.; Mustafin, R.; Safina, L.: Microservices: How To Make Your Application Scale, Feb. 23, 2017, arXiv: 1702.07149 [cs], URL: http://arxiv.org/abs/1702.07149, visited on: 04/08/2021.

[DW20] DWD: Radargestützte Analysen Stündlicher Niederschlagshöhen Im Echtzeitbetrieb Für Deutschland (RADOLAN) Und Mitteleuropa (RADOLAN-ME), Oct. 27, 2020, URL: https://www.dwd.de/DE/leistungen/radolan/radarniederschlagsprodukte/radolankurzbeschreibung_pdf.pdf?__blob=publicationFile&v=7, visited on: 03/29/2021.

[En20] Enerdata: Coal Phase Out - What Are the Major Global Coal Trends?, Mar. 20, 2020, URL: https://d1owejb4br3l12.cloudfront.net/publications/executive-briefing/global-coal-phase-out.pdf, visited on: 04/07/2021.

[GV17] Grieves, M.; Vickers, J.: Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In (Kahlen, F.-J.; Flumerfelt, S.; Alves, A., eds.): Transdisciplinary Perspectives on Complex Systems. Springer International Publishing, Cham, pp. 85–113, 2017, ISBN: 978-3-319-38754-3 978-3-319-38756-7.

[He20] Heideker, A.; Ottolini, D.; Zyrianoff, I.; Neto, A. T.; Salmon Cinotti, T.; Kamienski, C.: IoT-Based Measurement for Smart Agriculture. In: 2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor). 2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor). IEEE, Trento, Italy, pp. 68–72, Nov. 4, 2020, ISBN: 978-1-72818-783-9.

[KOJ19] Kwon, D.; Ok, K.; Ji, Y.: IBFRAME: IoT Data Processing Framework for Intelligent Building Management. In: 2019 IEEE International Conference on Big Data (Big Data). 2019 IEEE International Conference on Big Data (Big Data). IEEE, Los Angeles, CA, USA, pp. 5233–5238, Dec. 2019, ISBN: 978-1-72810-858-2.

[Lu20] Luckner, M.; Grzenda, M.; Kunicki, R.; Legierski, J.: IoT Architecture for Urban Data-Centric Services and Applications. ACM Transactions on Internet Technology 20/3, pp. 1–30, Oct. 2020, ISSN: 1533-5399, 1557-6051.

[Se18] Senožetnik, M.; Herga, Z.; Šubic, T.; Bradeško, L.; Kenda, K.; Klemen, K.; Pergar, P.; Mladenić, D.: IoT Middleware for Water Management. Proceedings 2/11, p. 696, July 31, 2018, ISSN: 2504-3900.

[St16] Stolpe, M.: The Internet of Things: Opportunities and Challenges for Distributed Data Analysis. ACM SIGKDD Explorations Newsletter 18/1, pp. 15–34, Aug. 2016, ISSN: 1931-0145, 1931-0153.

[Te19] Terroso-Saenz, F.; González-Vidal, A.; Ramallo-González, A. P.; Skarmeta, A. F.: An Open IoT Platform for the Management and Analysis of Energy Data. Future Generation Computer Systems 92/, pp. 1066–1079, Mar. 2019, ISSN: 0167739X.

[We17] Wehlitz, R.; Häberlein, D.; Zschörnig, T.; Franczyk, B.: A Smart Energy Platform for the Internet of Things – Motivation, Challenges, and Solution Proposal. In (Abramowicz, W., ed.): Business Information Systems. Vol. 288, Lecture Notes in Business Information Processing, Springer International Publishing, Cham, pp. 271–282, 2017, ISBN: 978-3-319-59335-7 978-3-319-59336-4.

[Wo18] World Wide Web Foundation: OpenData Barometer, Sept. 20, 2018, URL: https://opendatabarometer.org/doc/leadersEdition/ODB-leadersEdition-Report.pdf, visited on: 04/12/2021.

[Zs18] Zschörnig, T.; Wehlitz, R.; Rößner, I.; Franczyk, B.: SEPL: An IoT Platform for Value-Added Services in the Energy Domain - Architectural Concept and Software Prototype: In: Proceedings of the 20th International Conference on Enterprise Information Systems. 20th International Conference on Enterprise Information Systems. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, pp. 593–600, 2018, ISBN: 978-989-758-298-1.

[Zs20a] Zschörnig, T.; Windolph, J.; Wehlitz, R.; Franczyk, B.: A Cloud-Based Analytics Architecture for the Application of Online Machine Learning Algorithms on Data Streams in Consumer-Centric Internet of Things Domains. In: IoTBDS. Pp. 189–196, 2020.

[Zs20b] Zschörnig, T.; Windolph, J.; Wehlitz, R.; Franczyk, B.: A Cloud-Based Analytics-Platform for User-Centric Internet of Things Domains – Prototype and Performance Evaluation. In. Hawaii International Conference on System Sciences. 2020.