A Toolchain Architecture for Condition Monitoring Using the Eclipse Arrowhead Framework

A Toolchain Architecture for Condition Monitoring Using the Eclipse Arrowhead Framework

Federico Montori*†, Ivan Zyrianoff*†, Lorenzo Gigli*†, Riccardo Venanzi*, Simone Sindaco†, Cristiano Aguzzi†, Federica Zonzini†‡, Matteo Zauli†‡, Nicola Testoni†‡, Enrico Alessi§, Marco Di Felice*†, Luciano Bononi*, Paolo Bellavista*, Luca De Marchi†‡, Tullio Salmon Cinotti†

* Department of Computer Science and Engineering, University of Bologna, Italy
‡ Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi", University of Bologna, Italy
† Advanced Research Center on Electronic Systems "Ercole De Castro", University of Bologna, Italy
§ Advanced Research and System Platforms RND, Analog, MEMS and Sensors Group, STMicroelectronics, Catania, Italy
Corresponding author's email: federico.montori2@unibo.it

***Abstract*—Condition Monitoring is one of the most critical applications of the Internet of Things (IoT) within the context of Industry 4.0. Current deployments typically present interoperability and management issues, requiring human intervention along the engineering process of the systems; in addition, the fragmentation of the IoT landscape, and the adoption of poor architectural solutions often make it difficult to integrate third-party devices in a seamless way. In this paper, we tackle these issues by proposing a tool-driven architecture that supports heterogeneous sensor management through well-established interoperability solutions for the IoT domain, *i.e.* the Eclipse Arrowhead framework and the recent Web of Things (WoT) standard released by the W3C working group. We deploy the architecture in a real Structural Health Monitoring (SHM) scenario, which validates each developed tool and demonstrates the increased automation derived from their combined usage.**

***Index Terms*—IoT, WoT, Condition Monitoring, SHM, Arrowhead, Toolchain**

I. Introduction

The Internet of Things (IoT) has evolved rapidly over the last years, and it is nowadays used in a variety of application domains [1]. Among others, Condition Monitoring (CM) has gained considerable interest for industrial deployments of the IoT. In few words, CM refers to the monitoring of relevant condition parameters, such as vibration in machinery to identify faults or abnormal values, as well as forming a basis for predictive maintenance [2]. Most of these parameters are acquired by sensors that are deployed in the field. Although the concept of CM is general and could be applied to different use–cases, from railway tracks to wind turbines [3], its implementation is often problematic and may introduce significant engineering costs. On the one hand, the lack of automation in the various engineering phases, especially operation and maintenance, may imply a significant usage of manual work. This is an overhead in IoT scenarios where many operations should be managed remotely and automatically [4]. On the other hand, severe interoperability issues nowadays affect industrial IoT ecosystem, to the point that manually integrating legacy appliances constitutes an unbearable cost [5].

In the context of interoperability, we acknowledge the recent standardization effort on the W3C Web of Things (WoT), which aims to abstract simple sensors and actuators in an IoT environment into well-described Web resources [6]. However, there are still significant issues in automatically integrating legacy resources and meeting the requirements of a fully-fledged service-oriented architecture (SOA).

In this paper, we propose a toolchain architecture for the CM of industrial environments characterized by the presence of multiple sensors, heterogeneous in terms of capabilities and specifications. More in detail, we discuss a use–case related to Structural Health Monitoring (SHM), which is also a demonstrator for the EU project Arrowhead Tools[1], also available in a video[2]. The current baseline is constituted by an SHM sensor network, provided with a W3C WoT interface that monitors the vibrations over a frame building. Several requirements emerged during the operational phase, such as integrating third-party sensors and accessing external SOA resources, not yet mapped to the W3C WoT. For this reason, we enhance the baseline by considering a modular architecture composed of separate tools which integrate heterogeneous data services into a single fruition channel and automate the system configuration. As a result, the proposed CM is capable of: (i) supporting the integration of sensors with external IoT frameworks, such as the Eclipse Arrowhead framework, to be part of wider ecosystems and be easily managed by Orchestration facilities, (ii) integrating third-party sensors via a seamless and straightforward on-boarding procedure, and (iii) potentially adjusting the parameters of such sensors without on-site intervention. We validate the efficacy of the toolchain through selected experiments and report the observed system behavior utilizing the MODRON [7] dashboard.

The paper is structured as follows: Section II introduces the Eclipse Arrowhead framework. Section III presents the architecture of the whole System-of-Systems, while Section IV details the implemented tools and components, Section V describes the experiments and the validation results and, finally, Section VI concludes the paper and discusses some future works.

[1] https://arrowhead.eu/arrowheadtools
[2] https://youtu.be/f8R5vz6kKN4

II. The Eclipse Arrowhead framework

The latest years have been characterized by a fast-paced industrial revolution, especially in IoT-based ecosystems. In particular, the concept of Industry 4.0 caused a shift from the legacy SCADA/DCS systems to more flexible SOA architectures, where single systems are consuming or providing services, and the interactions are loosely coupled to ensure portability to different scenarios. The Eclipse Arrowhead Framework is a platform developed within the Arrowhead Project founded on the concept of Local Clouds: controlled environments that implement the base concepts of SOA – loose coupling, late binding, and discovery – and managed by a single instance of a set of Core Services [8]. Each Local Cloud hosts a so-called System-of-Systems in which single elements are generally addressed to as Arrowhead Application Systems or Arrowhead Tools. They are either service providers, thus servers that are exposed through endpoints, service consumers, thus clients that query other services, or both. The Core Services supervise all these interactions; the most important ones – the minimum set that must be deployed in a Local Cloud – are: the Service Registry, the Authorization, and the Orchestration. The Service Registry records the basic information of each of the services in the Local Cloud. Each provider registers itself, enabling discoverability and loose coupling. The Authorization detains a set of rules that establish which consumer is authorized to interact with which provider, also providing a token-based authentication facility. Finally, the Orchestration enables late binding by allowing the cloud manager to bind specific consumers to providers at run-time.

III. Proposed Toolchain Architecture

The architecture presented in this Section is related to a pilot use case of the Arrowhead Tools Project. In particular, this scenario includes a set of inertial sensors for Structural Health Monitoring (SHM) deployed onto a real building. Sensors are abstracted into Web Things (WTs) in order to be accessible via the Web, following the W3C WoT standard [6]. A set of engineering tools were developed in order to meet the following project objectives:

1) Integrate the sensor network with established interoperability frameworks (Eclipse Arrowhead[3]) to enhance the monitoring functions provided to the final user. Additionally, to provide sensor data persistence via the MODRON platform [7];
2) Integrate third-parties legacy sensor tools – in our case, a Gas Sensor – to increase the quality and quantity of information gathered from the scenario;
3) Automate the configuration of the sensors according to internal and external factors.

The overall architecture of the use case is shown in Figure 1: the remainder of the Section is dedicated to outlining the complex interactions between the components, which are then explored individually in Section IV.

The first project objective is achieved through tool called WoT-Arrowhead Enabler (WAE), which was presented in its early version in [9]. The WAE is the main hub for the translation between the WoT ecosystem and the Arrowhead System-of-Systems (SoS) in both directions. Since all WTs in the scenario are registered into a Thing Directory, the WAE periodically checks for newly registered WTs. Subsequently, it creates a server that acts as a proxy for the WT for each of them. Such proxies are called Arrowhead Thing Mirrors (ATM) and are deployed on the same machine where the ATM runs. This way, a WoT ecosystem offering a number of services can be queried and discovered in full using Eclipse Arrowhead. Therefore, clients do not have to implement the complete W3C WoT stack. Instead, they can virtually query WoT sensors via a REST API.

The second objective is achieved through the integration of third-party sensors within the whole ecosystem. In our use–case, we utilized an experimental Gas Sensor, as it provides valuable information for SHM scenarios, *e.g.* the possibility to perform degradation analyses based on data coming from multiple domains. The Gas Sensor has been integrated with Eclipse Arrowhead via an Arrowhead adapter, as it is explained in detail in Section IV-B, which makes it a fully-fledged Arrowhead service provider. Furthermore, the WAE has been modified to meet an additional requirement, *i.e.* to make such third-party service available also within the WoT ecosystem as a WT. This is implemented similarly to the point above, *i.e.* the WAE periodically discovers Arrowhead services that are suitable to be translated; we defined such suitability as a particular format of Open API Specifications (OAS)[4]. Upon finding the service endpoint, the WAE generates a new WT, which acts as a proxy for the third-party sensor and registers it into the Thing Directory of MODRON. This process is logically similar to the opposite presented above; however, it features a much more complex translation phase (WT Description has to be generated from the OAS).

Finally, the third objective is achieved by the Configurator tool, an Arrowhead service consumer used by a maintenance operator, able to change the configuration of the sensors at run-time. To do so, it discovers the WTs from the Arrowhead Orchestrator, and it issues one or more property changes, such as the sampling frequency. The whole sensor network has been redesigned in order to be able to receive commands from the WTs, enabling two-way communication. The Configurator tool comes with a Web interface, and it is described in detail in Section IV-C.

IV. Tools and Components

In the Arrowhead Tools project, it is important to outline the concept of "tool". An Arrowhead tool is a software component (or hardware with dedicated software on board) that improves an established baseline by bringing in additional automation over the course of one or more phases of the

[3] https://projects.eclipse.org/projects/iot.arrowhead

[4] https://swagger.io/specification/

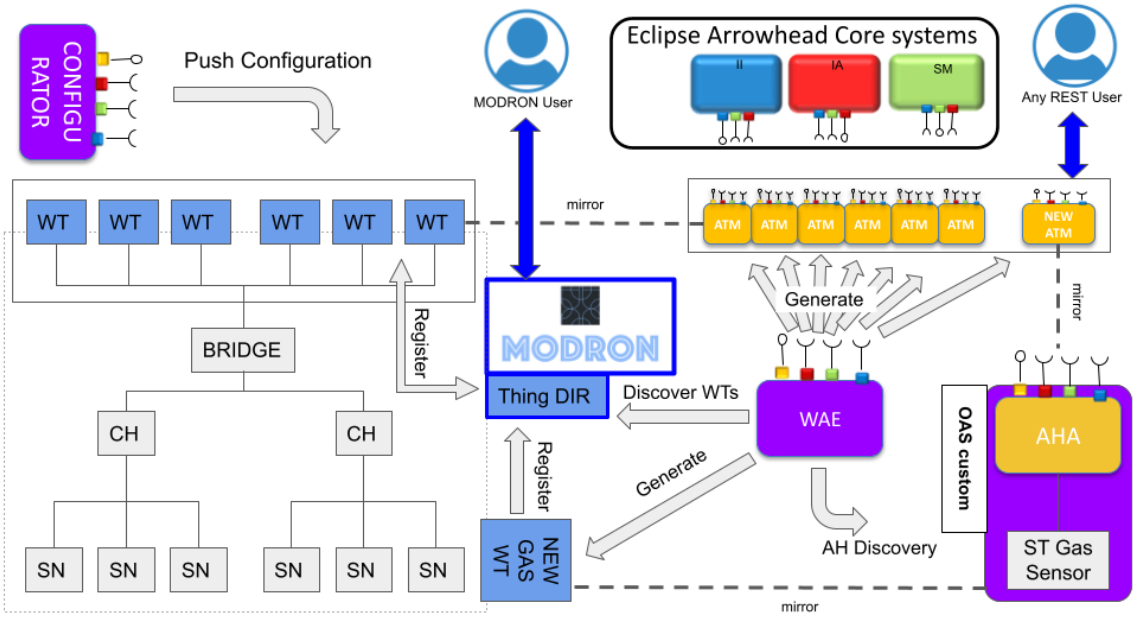

Fig. 1. Toolchain architecture of the whole System-of-Systems. Main abbreviations are: WT (Web Thing), SN (Sensor node), CH (Cluster Head), WAE (WoT Arrowhead Adapter), ATM (Arrowhead Thing Mirror), OAS (OpenAPI Specification), AHA (Arrowhead Adapter).

Arrowhead engineering process [10], [11]. In our use–case, we can identify as tools:

- The **WAE**, which operates in the Deployment phase of the engineering process.
- The **Gas Sensor** together with its adapter and service description, which improves the Operation & Management phase of the engineering process.
- The **Configurator**, which operates in the Maintenance phase of the engineering process.

This Section outlines in detail both tools and other components of the use–case.

A. SHM Sensor Network

A Smart Sensor Network (SSN) composed of low–power, light–weight and small–footprint accelerometer sensors was installed on a high–rise (5 m height) five–story building for the purpose of vibration analysis [12]. The physical layer of the monitoring network is composed of three different types of devices: (*i*) multiple peripheral sensor nodes (SNs) permanently attached to the structure, (*ii*) a few cluster head nodes (CHs) orchestrating a small subset of SNs and (*iii*) one aggregating unit (Bridge) in charge of data management and outsourcing. The sensing element consists of a MEMS–based iNemo inertial measurement unit released by STMicroelectronics, capable to capture dynamic measurements in quite a broad frequency range, while the core element of the sensor node is given by an STM32F303 microcontroller unit featuring a floating point unit for optimized digital signal processing functionalities; for a detailed hardware description of the SN, readers are referred to [13]. The processing flow is organized as follows [7]: once signals are sensed by the peripheral node, information is then transmitted to the bridge controller through the companion gateway network interface acting as a CH. Next, each of the sensors is abstracted into a WT, which acts as a virtual proxy and exposes data in the form of *properties*. Some properties are mutable (*e.g.* the sampling frequency) and, when changed by an external mashup application, they propagate the change over the sensor network to the actual SN.

B. Gas Sensor and Arrowhead Adapter

Integration is the key enabler for factory digitalization and to achieve interoperability in Industry 4.0 scenarios. The Arrowhead Framework provides components' integration through the development of a specific component, an Arrowhead Adapter (AHA). AHA has the pivotal role of overcoming components heterogeneity by making them Arrowhead compliant. It typically targets two integration scenarios, respectively the legacy component integration, and the heterogeneous component integration. In this Section, we showcase the capabilities of AHA through the integration of a real Gas Sensor with the Arrowhead Framework. Figure 2 depicts the components' architecture taking part into the Gas Sensor integration. It is composed of three main components, the Gas Sensor Node, the LoraWAN Network, and the AHA.

1) The Gas Sensor Node: The Gas Sensor Node is a sensor node powered by a B-L072Z-LRWAN1 evaluation board, which has an STM32L0, by STMicroelectronics, as main core and an SX1276 device by Semtec as LoRa transceiver. Thanks to its peripherals (I2C, SPI, etc.), it offers the possibility to manage different kinds of sensors, to acquire data and send them through a LoRaWAN network. The sensor managed by

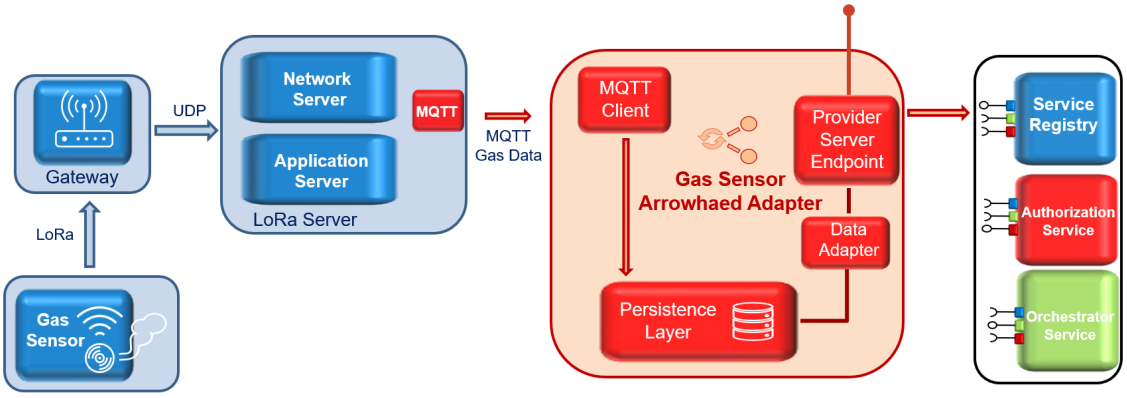

Fig. 2. Gas Sensor Arrowhead Adapter and Network

the board is an experimental Gas Sensor by STMicroelectronics [14]. More in detail, it is a Metal Oxide Semiconductor (MOX) sensor, based on MEMS technology which can provide information about the concentration of gas and different kinds of volatile organic compounds (*i.e.* Ammonia (NH_3), Nitrous Oxide (N_2O) and Methane (CH_4)) in the air. The low-power profile of the MCU, along with long range communication offered by the LoRa protocol, contribute to the implementation of a totally autonomous and *plug & play* sensor-node for SHM.

2) The LoRaWAN Network: The communication with the sensor node has been implemented using the infrastructure provided by The Things Network [5], that hosts the Application Server and the Network Server for the LoRaWAN Network. A proprietary gateway is used in order to forward data from the sensor-node to the LoRaWAN Server. The topology of the network is depicted in Figure 2. The Things Network equips LoRa Server with an MQTT module that enables the access to sensor data through a publish-subscribe mechanism.

3) AHA: Arrowhead Adapter: The AHA is the key unit that bridges any industrial component with the Arrowhead Framework (red box in Figure 2). It is mainly composed by two sub-components: an HTTP/REST interface for communicating with the Arrowhead Framework and for providing the service, and a component-side interface that communicates with the integrating system through its specific protocol. In this case, the Gas Sensor communicates through MQTT. At start-up time, the Gas Sensor AHA subscribes its MQTT Client to the Application Server's MQTT broker and remains in a wait state for new Gas Sensor data update. At the same time, the AHA creates a Gas Sensor Service to act as Arrowhead Provider and to make the Gas Sensor data queryable. AHA connects to the Arrowhead Core Services and registers the Provider Server Endpoint to Service Registry. When new Gas Sensor data are available, the MQTT client receives the update and stores it into a Persistence Layer. The Persistence Layer is an abstraction of any type of storage component, spanning from in-memory storage to any database (DAO pattern is used) passing through file, JSON, or XML. At this point, the Data Adapter is responsible for converting data from the stored format to a format easily manageable from HTTP/REST (JSON). Finally, the Provider Server Endpoint listens for direct requests from Arrowhead consumers and returns data.

[5]https://www.thethingsnetwork.org

C. Configurator

The WoT–Arrowhead Configurator[6] is an open–source application that provides a friendly browser–based user interface for managing WTs within the Arrowhead ecosystem. The application was developed in Typescript using the Angular framework. It lists all the WTs registered in the Service Registry, allowing the user to easily check their attributes, properties, actions, and events in two different modes: (i) In a raw JSON – adequate for advanced users; (ii) In an interactive tabular matter, divided in categories (*i.e*, general attributes, properties, actions and events) – as depicted in Figure 3. Furthermore, the Configurator allows the user to change properties or invoke actions using the graphic interface without needing to have specific knowledge of the WT interface. The Configurator reads the information of each Thing Description of available WTs, and only allows to update writable properties (in our scenario, the sampling frequency); if present, it displays an array of accepted values for a given property as a drop-down list. Given that the list of WTs registered in the Service Registry is maintained and updated by the WAE, the Configurator guarantees that the web interface is in synchrony with the rest of the system.

D. MODRON

MODRON is a software framework for sensor–to–cloud data gathering and management in SHM scenarios [7]. The system leverages the W3C WoT standard to support multi–source data acquisition from heterogeneous sensors, which may use different protocols and sensing principles. The MODRON architecture envisions a layered approach, with an edge layer on which WTs are exposed and a cloud layer that includes a suite of services for data storage, aggregation, and

[6]https://github.com/UniBO-PRISMLab/wot-configurator

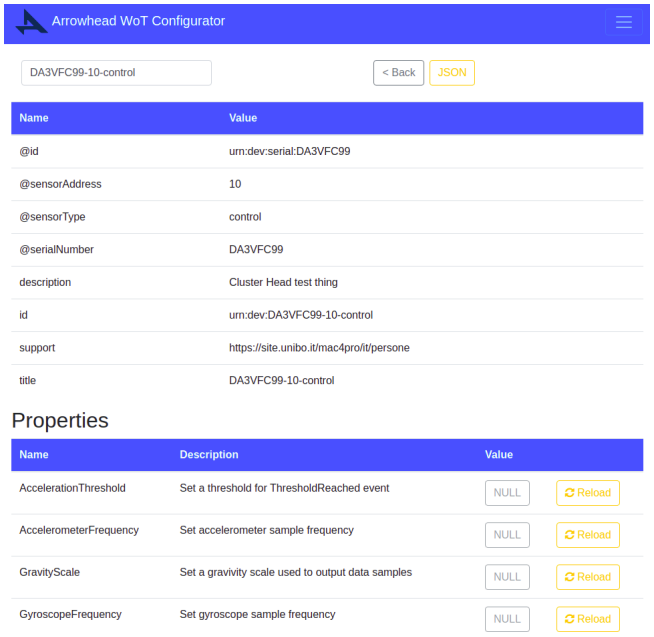

Fig. 3. A screenshot of the Configurator tool.

visualization. On the cloud side, we can find six specific components: the Thing Discovery Service (TDS), the Thing Visualizer Service (TVS), the Persister and Analytics mash–up applications, the Data Aggregator and Plotter. The TDS is basically a Thing Directory (highlighted in Figure 1) that talks directly to the edge layer and is responsible for automatically registering and notifying each new WT that is deployed. The TVS is a playground where users can visualize and interact with their WTs thanks to a dynamically generated graphic interface starting from the TDs. The Persister is in charge of retrieving data from the WTs by querying them at fixed intervals or subscribing to their "data publishing" events. Data is then saved within a distributed database, ensuring their scalability and availability. The Analytics application extracts the raw data from the database and performs signal processing techniques for structural integrity evaluation. Finally, the Data Aggregator service allows setting aggregation and combination operations on the data, and the Data Plotter service for the visualization and export of the data exposed by the Aggregator, which serves as a user dashboard within our demo.

V. Results and Validation

In this Section we present the validation of the whole ecosystem by showcasing two results: (i) the efficacy of the Configurator, which is able to change the properties of the leaf SN at runtime, and (ii) the integration of external, non–WoT, Arrowhead services (in this case, the ST Arrowhead-adapted Gas Sensor) within the same monitoring facility. These two achievement significantly enhance the baseline, in which said operations were done by hand.

Controlling the data rate at which information is gathered is of the uttermost importance in energy–efficient and responsive monitoring scenarios. As such, being capable to change this parameter from remote could be even more crucial to better track potential changes in the vibration signature of the monitored structure. To cope with this requirement, the functionality to reconfigure at runtime the sampling frequency at which each specific SN is working has been enabled by acting on the corresponding property of the WTs by means of the Configurator. In the specific case of the frame building, tests have been performed in which the output data rate of one sensor node has been decreased from $813\,\mathrm{Hz}$ to $416\,\mathrm{Hz}$, which are both compatible values with the spectral signature of the structure since the most energetic components are located below $200\,\mathrm{Hz}$. In Figure 4, a number of snapshots taken from an experimental test are displayed while configuring the system to acquire 2000 samples–long time series. In the initial configuration (step 1), the sampling frequency corresponds to $416\,\mathrm{Hz}$, implying a total observation window of $4.807\,\mathrm{s}$; then, after the Configurator is called, the data rate is set at $833\,\mathrm{Hz}$, thus leading to a new burst duration of $2.401\,\mathrm{s}$. To further corroborate the obtained results, data tips are included in which the acquisition time for two consequent data points are shown: as can be observed, this time step halves from $2.4\,\mathrm{ms}$ (*i.e.* $1/416$) to $1.2\,\mathrm{ms}$ (*i.e.* $1/833$). All results are shown by means of the MODRON dashboard. Figure 5 shows the experimental Gas Sensor on the top, together with its monitored values from the MODRON dashboard. For displaying purposes, we show the Gas Sensor in its state of stillness on the left end side, while on the right-hand side we stimulate it by spraying gas from a lighter, causing the monitored resistance value (shown in red) to decrease sharply.

VI. Conclusion

In this paper, we tackled the interoperability and management issue in IoT–based Condition Monitoring applications. We provided a concrete experience on how an established baseline has been improved through the development of engineering tools. In practice, the usage of such tools guarantees an increased automation in the engineering process of the whole use case, thus reducing operational costs, as well as facilitating the on-boarding of third-party legacy devices. Both the Eclipse Arrowhead framework and the WoT paradigm have been used extensively in pair. As a future work, we expect to enhance the automation further by: (i) automatically issue the Configurator with optimal parameters based on environmental conditions and (ii) shift the data storage and governance to the cloud, for a wider fruition by different stakeholders.

Acknowledgment

This research is funded by ECSEL, the *Electronic Components and Systems for European Leadership Joint Undertaking* under grant agreement No 826452 (Arrowhead Tools), supported by the European Union Horizon 2020 research and innovation programme and by the member states, and by INAIL within the BRIC/2018, ID=11 framework, project MAC4PRO ("Smart maintenance of industrial plants and civil structures via innovative monitoring technologies and prognostic approaches").

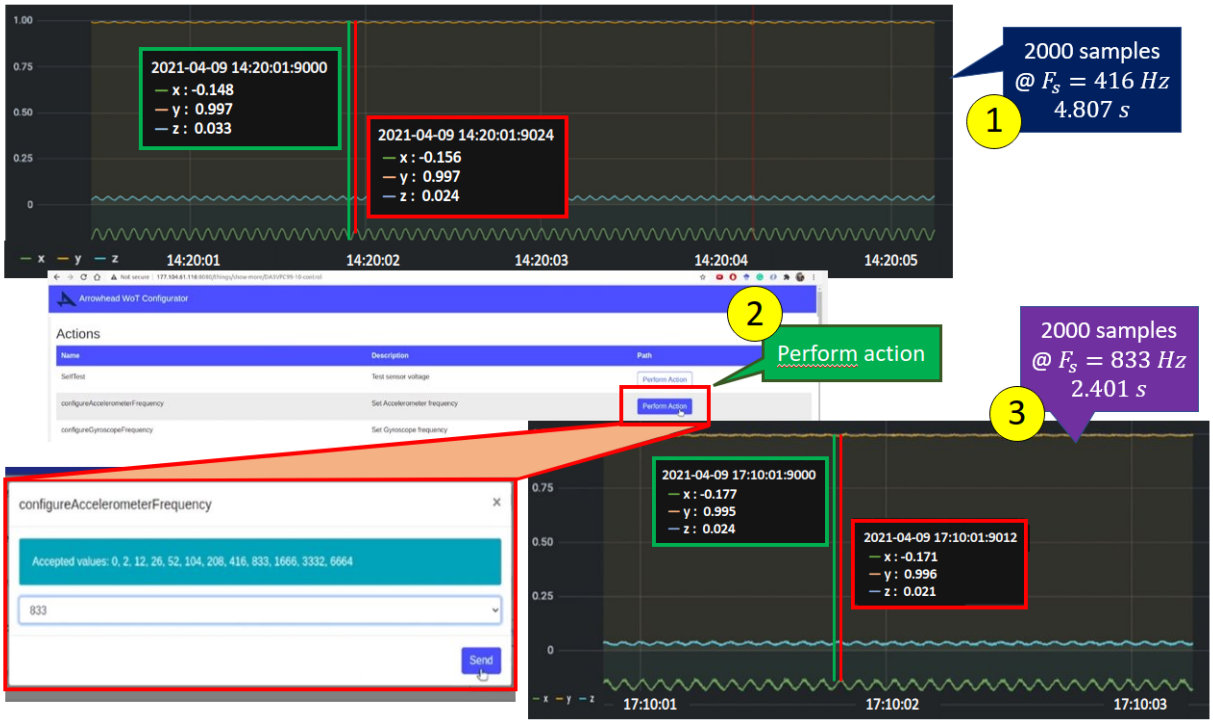

Fig. 4. Screenshots showing the temporal effects of the sampling frequency change made through the Arrowhead Configurator tool for 2000 samples length data series: starting from an initial data rate of 416 Hz, which corresponds to 4.807 s (step 1), the acquisition time is halved (*i.e.* 2.401 s) (step 3) after setting the sample rate to 833 Hz (step 2).

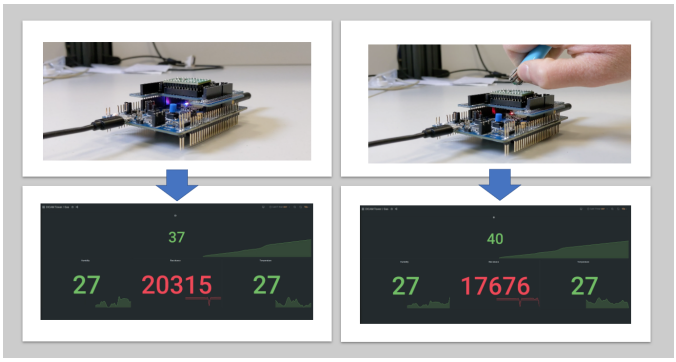

Fig. 5. A snapshot of the Gas Sensor values on the MODRON Dashboard.

References

[1] L. Atzori, A. Iera, and G. Morabito, "Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017.

[2] R. B. Randall, *Vibration-based condition monitoring: industrial, automotive and aerospace applications*. John Wiley & Sons, 2021.

[3] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review," *Renewable energy*, vol. 133, pp. 620–635, 2019.

[4] G. Wang, M. Nixon, and M. Boudreaux, "Toward cloud-assisted industrial iot platform for large-scale continuous condition monitoring," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1193–1205, 2019.

[5] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, "Interoperability for industrial cyber-physical systems: An approach for legacy systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3370–3378, 2017.

[6] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Toumura, and K. Kajimoto, "Web of things (wot) architecture," W3C recommendation, Apr. 2020. https://www.w3.org/TR/wot-architecture/.

[7] C. Aguzzi, L. Gigli, L. Sciullo, A. Trotta, F. Zonzini, L. De Marchi, M. Di Felice, A. Marzani, and T. S. Cinotti, "Modron: A scalable and interoperable web of things platform for structural health monitoring," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–7, IEEE, 2021.

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

[9] L. Sciullo, F. Montori, A. Trotta, M. Di Felice, and T. S. Cinotti, "Discovering web things as services within the arrowhead framework," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1, pp. 571–576, IEEE, 2020.

[10] G. Urgese, P. Azzoni, J. van Deventer, J. Delsing, and E. Macii, "An engineering process model for managing a digitalised life-cycle of products in the industry 4.0," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, IEEE, 2020.

[11] G. Kulcsár, M. S. Tatara, and F. Montori, "Toolchain modeling: Comprehensive engineering plans for industry 4.0," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pp. 4541–4546, IEEE, 2020.

[12] F. Zonzini, C. Aguzzi, L. Gigli, L. Sciullo, N. Testoni, L. De Marchi, M. Di Felice, T. S. Cinotti, C. Mennuti, and A. Marzani, "Structural health monitoring and prognostic of industrial plants and civil structures: A sensor to cloud architecture," *IEEE Instrumentation & Measurement Magazine*, vol. 23, no. 9, pp. 21–27, 2020.

[13] N. Testoni, C. Aguzzi, V. Arditi, F. Zonzini, L. De Marchi, A. Marzani, and T. S. Cinotti, "A sensor network with embedded data processing and data-to-cloud capabilities for vibration-based real-time shm," *Journal of Sensors*, vol. 2018, 2018.

[14] C. Bruno, A. Licciardello, G. A. M. Nastasi, F. Passaniti, C. Brigante, F. Sudano, A. Faulisi, and E. Alessi, "Embedded artificial intelligence approach for gas recognition in smart agriculture applications using low cost mox gas sensors," in *2021 Smart Systems Integration (SSI)*, pp. 1–5, IEEE, 2021.