Ubiquitous System Integration as a Service in Smart Factories

Mirco Soderi, Vignesh Kamath, Jeff Morgan, John G. Breslin
Data Science Institute, National University of Ireland Galway, Ireland {firstname.lastname}@nuigalway.ie

***Abstract*—The state-of-the-art in manufacturing technology identifies a data rich, horizontally and vertically integrated environment, with collaborative control and intelligence systems, to enable what is called a *smart factory*. Similar to *digital factories*, smart factories signify a greater digital convergence between manufacturing Operational Technology (OT), and wider enterprise Integrated Technology (IT). With these new capabilities comes new challenges in scale, complexity, and the skills needed to enable ubiquitous system integration. While in the past, engineers have utilized a variety of proprietary graphical configuration and programming tools to intuitively create simple to complex manufacturing systems, presently, the standard technology ecosystem which engineers design, control, and maintain, is observably expanding, and/or evolving to incorporate more open-source technologies, network services, and distributed devices. Therefore, this paper will examine the emerging next generation technologies and tools which will support engineers in designing and interacting with the new digital manufacturing ecosystem in smart factories. To achieve this: a state-of-the-art survey of commercial Industrial Internet of Things (IIoT) technology is presented; three key ubiquitous technology trends are examined; and a use-case is presented with a universal framework. Uniquely, this framework demonstrates the collective capabilities of the technologies to overcome ubiquitous system integration problems.**

***Index Terms*—Digital Factory, Smart Factory, Internet of Things (IoT), Software Containers, Graphical Programming Interfaces (GPI), Enterprise Integration, Micro-services, Node-RED, Cyber Devices.**

I. Introduction

It has been envisaged that the next generation of manufacturing factories will integrate their business functions with manufacturing systems, creating new levels of connectivity and autonomy, representative of a *digital factory* [1]. Presently, this is being achieved through the merging of (business) Integrated Technology (IT) and (production) Operational Technology (OT), with enabling Information and Communication Technologies (ICT), such as the Internet of Things (IoT) [2], Digital Platforms [3], and the Cloud [4]. With new connectivity comes new opportunities to align manufacturing execution and control systems with analytical and artificially intelligent systems, to collectively produce Cyber-Physical Systems (CPS) [5], and enable *smarter* manufacturing, also called *smart factories* [6]. In both the Smart and Digital factory paradigm, internal systems redefine their constructs as either subscribing to services, and/or presenting their functions as services, in a true Service-Oriented Architecture (SOA) design [5]. Observably, this creates a new digital frontier, a distributed digital manufacturing ecosystem [7], as production systems become distributed digital assets in wide-area networks [8].

To date, considerable efforts have been made to understand the new digital ecosystem, via state-of-the-art reviewers and technical articles relating to Advanced Sensor Technologies [9], Digital Platforms [3], Cloud [4], Production Monitoring and Visualization [10], Preventative Maintenance [11], Security [12], Industrial Communication [13] and Robotic Control [14]. From these references, the composition of the new digital ecosystem can be imagined as consisting of a range of IoT devices, layered software environments (Edge-Fog-Cloud), commercial and industrial communication standards and servers, data models, services/micro-services, and both commercial and open-source technology. However, throughout the extended IoT literature, the digital ecosystem presents several challenges, such as: (i) the diversity and scaling complexity of industrial IoT devices available which impact the cost of deployment and integration [7]; (ii) the complexity of the heterogeneous networks and connected devices (greenfield/brownfield IT/OT systems) which require significant work to integrate and unify information models [15]; (iii) a lack of technical designing interfaces for the integration of Cyber and Physical Systems and exploring the interaction dynamics of the resulting CPS [16]; (iv) limited interfaces between enterprise corporate enterprise levels and the manufacturing shop floor level [17]; and (v) a requirement for a diverse technical and analytical skillset for the new *digital worker* [18] [19] [20], a *worker 4.0* [21], or commonly referred to as a *digital engineer*. Presently, there is an opportunity to investigate solutions to these challenges, which from here on in is referred to as the *Ubiquitous System Integration* challenge.

In summary, this paper will examine the emerging next generation technologies and tools which will support engineers in designing and interacting with the new digital manufacturing ecosystem in smart factories. To achieve this, a state-of-the-art survey of Industrial Internet of Things (IIoT) technology has been carried out, and this has allowed the identification of some key technology trends which offer potential solutions to the *Ubiquitous System Integration* challenge, that are analysed and discussed in Section II. Furthermore, such ubiquitous technologies are formulated into a universal framework representative of the new digital manufacturing ecosystem. This universal framework is presented

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number SFI/16/RC/3918 (Confirm).

in Section III, with a use case operating within the *Industrial Computation Laboratory*. A specialization of the universal framework, namely the secure-by-design *Ubiquitous System Integration as a Service* architecture is presented in Section IV alongside the implementation details of its key components. It is important to note that all artifacts are available to download as open source, in order to promote research and development in the community as well as industry adoption of these foundational digital and smart factory capabilities. The results are summarized in Section V. Future directions are announced in Section VI. Conclusions are drawn in Section VII.

II. Technology Trends

The purpose of the IIoT technology survey carried out in the context of this work is to understand the state-of-the-art for engineers working within the digital ecosystem. For context, present speculation into the impact of the Industrial Internet of Things (IIoT) technology has placed focus on the optimization of supply chains with new efficiency improvements [22]. It is forecasted that the enterprise and automotive market will grow to 5.8 billion IIoT endpoints [23], with the total IIoT market expected to reach 263 billion dollars in 2027 [24]. The composite IIoT technologies which represent this speculation have been analytically reviewed in Boyes et al. [25] both conceptually and characteristically. Key reference is made to the Purdue reference model for manufacturing enterprise architecture, and an IoT architecture model consisting of OT (sensors, actuators, gateways, data acquisition devices), and IT (Edge processing, data centre, and the Cloud).

In total 33 technologies were surveyed, including 5 IoT devices, 5 Application-Specific Integrated Circuit (ASIC) devices, 8 IIoT and industrial compute devices, 10 industrial control devices, and 5 interconnected middleware (Fog) technologies. For context, the technologies reviewed can enable high-speed machine control and supervisory control, small-to-large sensing and data acquisition capabilities, low-to-high Edge computing performance, fixed-to-configurable-to-customizable hardware and software capabilities, and low-to-high level programming. They all have been analysed as for their functional capability, scaling cost, strategic technology partnership, analytical strategy (Edge/Fog/Cloud dependencies), technology familiarity, time to market, configuration vs customization requirements, solution ownership/support/maintenance, and data/communication standards.

The following key trends and collective insights have emerged: (i) *Integration of Information Communication Technology*, specifically the universal communication standards, such as OPC UA and MQTT/SparkPlug, and the embedded communication mediums, such as Wi-Fi and dual ethernet for WAN and LAN capabilities. All of them enable standard data distribution paths from Edge devices to Fog systems, and new paths toward Cloud connectivity. (ii) *Utilization of open hardware and software platforms*, examples of which can be seen in (a) the low-cost Raspberry Pi platform for wide spread sensing, (b) the increasing usage of Linux operating systems for real-time control, higher level programming capabilities, and Cloud connectivity, and (c) the virtual software Containers that enable ubiquitous software and micro-service deployment across multiple platforms. (iii) *Diversity of compute technology*, from Application Specific Integrated Circuits (ASICs) such as *on-chip* programming via Field Programmable Gate Arrays (FPGA), Graphical Programming Units (GPUs) for Edge machine learning, and dedicated Visual Processing Units (VPUs) for pattern recognition, to embedded and modular controllers, devices, industrial computers, and network gateways. (iv) *Expansion of the Cloud to Edge services*, intended as the deployment of Cloud services to both Edge and Fog environments, offering Software as a Services (SaaS) onto dedicated computing hardware which is closer to its intended application of use (examples of this can be seen in vision machine learning services, and IoT hubs for data acquisition and transportation to the Cloud via OPC UA and MQTT communication). (v) *Increase of graphical programming and configuration interfaces*, for example devices/controllers with "out of the box" graphical and Web browser compatible user interfaces to configure and sometimes even program the device, and the use of graphical programming languages like Node-RED. (vi) *Singular to modular designs*, separation between the low cost centralized IoT devices, and distributed/decentralized modular IoT devices offering extensible re configurable modular sensing, and clustered compute modules. Uniquely, the use of virtual software Containers is enabling modularity in software solutions deployable across different device types, offering modular M2M communication, graphical programming, data processing and data sharing (vii) *Collective vendor offerings*, where IoT technology typically comes in a bundle with a vendor-specific Fog/Cloud solution, thereby encouraging the building of digital ecosystems that are exclusive to a single brand (although in most cases connectivity to the main Cloud providers is granted anyway). (viii) *Dual operating systems*, which enable higher programming and wider communication capabilities: some industrial controllers rely on a dual operating system to balance real-time control and complex task execution, and some providers of automation control devices rely on Linux-based operating systems to enable the usage of solutions and services that are freely available, and maintained by wider software communities.

III. Ubiquitous System Integration Framework

Based on the technology trends outlined in Section II, a Ubiquitous System Integration Universal Framework is proposed and represented in Fig. 1. The framework encompasses: (i) field-level industrial communication with OPC UA and MQTT; (ii) Graphical Programming Interfaces (GPI) for Cyber-Physical System integration and data visualization; (iii) virtual Containers to deploy software applications and services in a ubiquitous scalable *Fog cluster*; (iv) a set of *Dock functionalities*, that typically consist of customizable signal/data processing services, such as signal processing, complex fuzzy logic, or machine learning. As such, this

framework is representative of a digital ecosystem which is decentralized yet interconnected, and collaborative. The purpose of this framework is to connect industrial controllers (manufacturing machines) and IIoT devices (process monitoring) to factory services (Manufacturing Execution Systems), and to collect, analyse and display data (Manufacturing Intelligence).

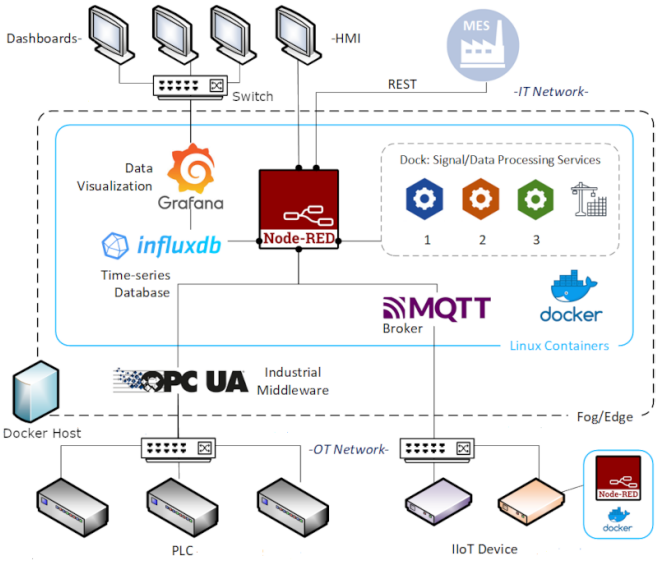

Fig. 1. Ubiquitous System Integration Universal Framework

Objectively, this framework provides an *entry-level* capability for digital and smart factories. This capability is however evidently scalable and universal for Cyber-Physical System integration, which will enable it to be applied to a wide range of manufacturing production systems. For example, multiple Fog clusters can be realised and seamlessly connected. As such, the framework can systematically scale in complexity, with new IIoT connectivity, computation, user interfaces, system integration. In doing so, the framework can be developed and expanded over time to include new nodes, new Containers, and new clusters to support advanced digital and smart factory capabilities (see also Section IV).

A. *Proof of concept*

A proof of concept of the framework has been put in place in the *Industrial Computation Laboratory*. A picture of this framework is provided in Fig. 2.

In our proof of concept, the hardware of the digital ecosystem consists of three industrial PLCs and two IIoT devices, which are connected to two network switches representing an industrial network and an IoT network. The PLCs are programmed with a *virtual state machine*, with five operating steps and three global states. The PLCs are connected to the industrial network and communicate with an industrial middleware server via a proprietary Ethernet protocol. The industrial middleware server has read/write access to the PLC data and presents this data to other systems via OPC UA protocols. The IIoT devices acquire data from three sensors and are connected to the IoT network. The IIoT devices

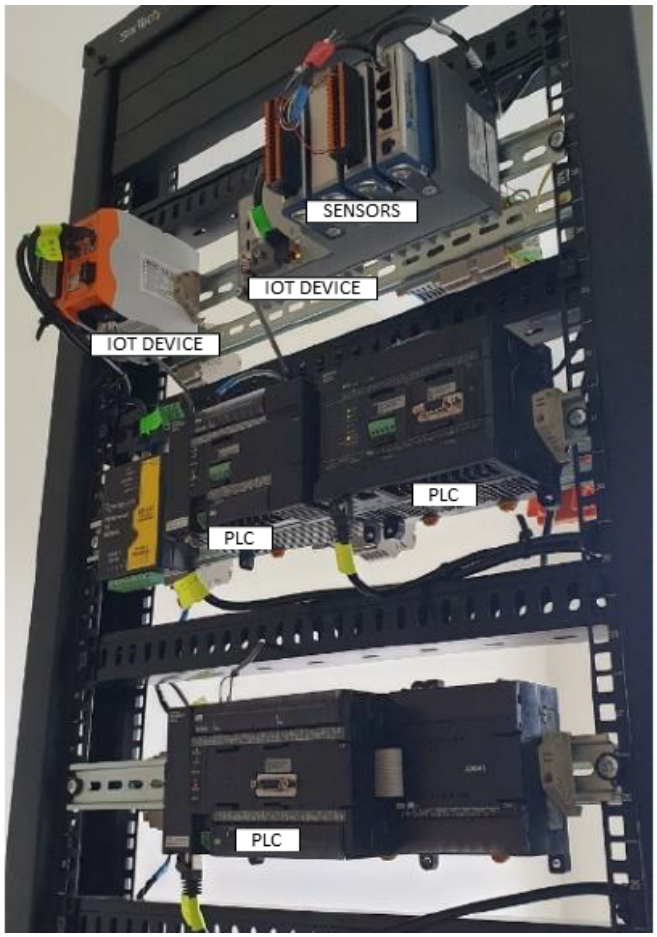

Fig. 2. Proof of concept of the Universal Framework

communicate sensor data to a MQTT broker via the MQTT protocol.

The software of the digital ecosystem is wrapped in a Fog cluster that consists of a few Docker Containers, and it includes: (i) a Mosquitto MQTT broker for internal and external data streaming; (ii) an InfluxDB Timeseries database instance for historic data storage; (iii) Grafana for data visualization; (iv) a Node-RED instance for the implementation of the cyber devices, i.e. the Cyber-Physical Systems.

Fundamental to the Fog cluster is the integration of local and global Cyber Physical Systems (CPS). This is achieved by virtually/remotely wiring Node-RED *nodes*, wrapped in *flows*, through the Node-RED graphical programming interface that is accessible through the Web. A flow example follows: (i) a node is placed/configured to acquire data, such as an MQTT/TCP/IP input node; (ii) the node is connected to another node which processes the data, for example extracting specific values from a message represented as a JSON object; and (iii) the data processing node is connected to the end of the flow, which is still a node that stores the data in a relational database. Other flows have been developed in the context of this proof of concept that are aimed at: (i) controlling machine start and stop; (ii) producing machine and MES records; (iii) detecting and quantifying the machine

downtime. Node-RED nodes can be assigned unique names and disposed on the screen in a convenient way to ease software reading and maintenance. Node-RED also provides useful tools that assist the developer in system design, control, and error trapping. As such, an engineer can intuitively manage and scale digital connections and transactions.

Data are displayed through interactive dashboards. This is achieved in several ways. Firstly, Node-RED enables the displaying of data, the monitoring of the internal status of the system, and the performing of actions onto the system, through the Web. Also, Grafana enables the interactive displaying of historical data, which is achieved through queries to the time-series database. Since data is exposed through the Web, it is available to multiple users, such as manufacturing operators, engineers, and managers. All users can utilize this data to understand what is happening and get supported in decision making.

IV. Ubiquitous System Integration as a Service

The Ubiquitous System Integration as a Service (USIaaS) architecture is for building a network of cyber devices, each of which, named *Service Nodes (SN)*, is characterized by the following: (i) it asynchronously communicates with the other nodes and physical devices by means of MQTT brokers; (ii) it can have an arbitrary number of inputs and outputs; (iii) it can perform any of the data transformations, readings, and writings that are available in an extensible external library, including reading from and writing to OPC UA servers; (iv) it is created, deleted, copied, moved, fully configured, in a matter of milliseconds through secure calls to APIs that are exposed by the Service Node itself; (v) it can be moved, stopped, restarted, its execution environment can be updated, without losing any of its configurations; (vi) it comes with a graphical Web interface through which it is possible to monitor and possibly even edit its implementation on the fly (although discouraged).

In Fig. 3, the internal structure of a Service Node is represented, along with its interconnections to the other actors in the digital ecosystem of our proposed reference architecture and implementation of the Ubiquitous System Integration as a Service. A Service Node is a dockerized Node-RED application. Creating a new Service Node is running a new Docker container that bears a Node-RED instance with a pre-installed implementation of the Service Node Node-RED application. The Node-RED implementation of the Service Node is available as open source[1]. The application is made up of two flows: (i) the *Configuration* flow, where the APIs that allow the configuration of the Node are implemented; (ii) the *Business Logic* flow, where Node-RED MQTT input and output nodes are located, along with the *Transformation Instance* which is an instance of the subflow that implements the actual job the Service Node performs.

More specifically, the APIs implemented in the Configuration flow allow us to: (i) create and delete input and output MQTT nodes; (ii) configure the endpoint, topic, Quality of Service, secure connection, and credentials, for the created input and output MQTT nodes; (iii) set and update the data transformation of the Service Node, selected from an expandable set of subflows that are implemented in a separated Node-RED application running in a separated Docker Container, named the *Transformations Library (TL)*; (iv) create and remove the wires that connect the input and output MQTT nodes to the input and output ports of the Transformation Instance; (v) add and remove Node-RED modules to allow the execution of subflows that require specific modules that are not pre-installed in the Node-RED server instance that comes with the official Docker Image of Node-RED; (vi) rearrange the graphical elements in the Business Logic subflow; (vii) pause and restore the Service Node, disabling and enabling the Business Logic flow; and (viii) read the current configuration through a number of specific APIs, one for each configurable aspect of the nodes that are in the Business Logic subflow.

The typical execution flow of a Configuration API that is aimed at *setting* a configuration parameter is as follows: (i) The ACL API is called to verify if the requester is authorized to set that specific configuration. The ACL API is implemented in a separated Node-RED application that executes in a separated Docker Container. Although in our reference implementation the ACL API is implemented in Node-RED, it can be implemented in any language, and it can execute the most diverse organization-specific business logics to decide if the requester is authorized or not, possibly including LDAP access, Keycloak API calls, and much more. A sample implementation of the ACL API, arguably not suitable for production use just yet, is available as open source[2]. (ii) The current implementation of the Business Logic flow, or of the entire Node-RED application, is retrieved through appropriate calls to the Node-RED admin APIs. (iii) The object of interest is identified, and in it, the value of interest is modified. (iv) The new implementation of the Business Logic flow or of the entire Node-RED application is persisted through appropriate calls to the Node-RED admin APIs. (v) Finally, the response is returned to the requester.

The typical execution flow of a Configuration API that is aimed at *reading* the current value of a configuration parameter follows: (i) the ACL API is called to verify if the requester is authorized to read that specific configuration; (ii) the current implementation of the Business Logic flow, or of the entire Node-RED application, is retrieved through appropriate calls to the Node-RED admin APIs; and (iii) the configuration of interest is located therein, and it is returned to the requester.

The Configuration API that allows setting the transformation (Transformation Library subflow) that the Service Node must execute is also implemented in the Configuration subflow of the Service Node itself. It is exposed at the path */node/transformation* and it proceeds through the following steps: (i) the validity of the payload of the request, available in the Transformation Library, is verified. No spaces

[1]https://drive.google.com/file/d/1jCxldMvs6XUx33E__CKdjBRoFgbeQqNQ [Note: access to Google Drive may be slow]

[2]https://drive.google.com/file/d/1EliH_pqZS_FQMSW13dWl5GC8XQz1-8Wi

are allowed, and *node* cannot be used as the name of a transformation subflow since it is reserved for Service Node level configurations. (ii) The ACL API is called to verify if the requester is authorized to set the transformation to be executed. (iii) The Node-RED admin API is called on the Transformation Library to retrieve the set of the available transformations. (iv) The subflow of interest is identified, and its contained nodes also are identified, along with Node-RED configuration nodes that are used in them. (v) The current implementation of the Service Node is retrieved through calls to the Node-RED admin APIs exposed by the Node-RED instance that is running in the Service Node. (vi) The metadata and content of the subflow that implements the transformation that the Service Node performs are updated, and set equal to the metadata and content of the new subflow retrieved by name from the Transformation Library. (vii) A check is performed to see if all required Node-RED modules are available in the Service Node. (viii) Also a check is carried out such that the IDs of objects copied from the Transformation Library do not conflict with the IDs of existing nodes in the Service Node. (ix) Appropriate calls are made to the Node-RED admin APIs of the Node-RED instance that is running in the Service Node to persist all changes. (x) Lastly, a response is returned to the requester.

Our current implementation of the Transformation Library encompasses several ready-to-use subflows, that can be categorized in: (i) I/O adapters; (ii) filters; (iii) data transformers; (iv) timed/flow controllers. The available subflows that belong to the category of the *I/O adapters* are self-explanatory: (i) dbtail; (ii) dbread; (iii) dbwrite; (iv) filetail; (v) filewrite; and (vi) the OPC UA set. The available subflows that belong to the category of the *filters* are also self-explanatory: (i) greaterthan; (ii) lowerthan; (iii) between; (iv) outside; (v) enum. The available subflows that belong to the category of the *data transformers* are also self-explanatory: (i) expr; (ii) template; (iii) scale; (iv) switch. The available subflows that belong to the category of the *timed/flow controllers* are: (i) clock; (ii) sampler; (iii) quarantine; (iv) watchdog; (v) fork. All subflows include input and output ports, and a status port, through which the status of the transformation subflow (and so the status of the Service Node) is made available. In this way, the status comes to be visible in the Web interface that displays (and possibly allows editing of) the Business Logic flow, where it is represented by means of a coloured icon and short textual description. Typically, a Transformation Library subflow implements both the data transformation and the configuration APIs that are specific to that transformation. The current implementation of the Transformation Library is available as open source[3].

The proposed architecture and its implementation have been designed having security in mind since the outset. All Node-RED instances are secured through the configuration of usernames and passwords in their respective configuration files, and all communications take place over HTTPS. In the reference architecture, the open-source EMQ broker is used. The architecture and its implementation are in any case agnostic with respect to the specific broker in use, if it implements the MQTT protocol. The EMQ broker used in the reference architecture has been configured to use an external ACL API, that in the reference architecture has been implemented as a Node-RED application. It is at present mostly suitable for development and testing purposes, and its sample implementation is available as open source[4]. In the reference architecture, all Docker Containers are connected to the same Docker user-defined bridge/network. For production use, if not all Containers can be connected to the same Docker user-defined bridge/network, for example because they are running on different Docker hosts, the *add-host* option can be used to add DNS entries and allow Containers to be in sight of each other. A number of environment variables are also expected to be set for Docker Containers that run Service Nodes[5].

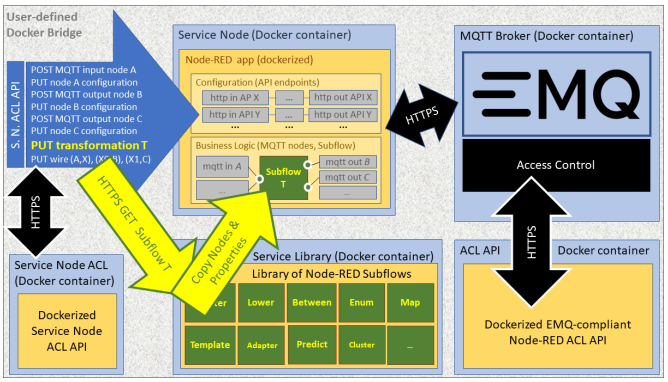

Fig. 3. Ubiquitous System Integration as a Service

V. Results

The proof of concept for the *Ubiquitous System Integration Framework* presented in Section III encompasses Node-RED dashboards through which the user is able to monitor the hardware apparatus, and even to interact with it in real-time. More specifically, through the Node-RED dashboards, the user can: (i) display the current values of humidity, vibration, and temperature, in speedometer/gauge charts; (ii) display the current status of the PLCs through a separate LED for each state and PLC, coloured in red or green; (iii) switch on and off the PLCs through graphical switches drawn on the Node-RED dashboard. A sample Node-RED dashboard is represented in Fig. 4. Also, the proof of concept for the *Ubiquitous System Integration Framework* encompasses Grafana dashboards through which the user can: (i) display the current values of temperature and humidity in gauge charts that also carry a qualitative description of the current value by means of a coloured edge; (ii) display the history

[3]https://drive.google.com/file/d/1xLL_sGaIG9PDwXAxbOhUbrznKieNKXEK

[4]https://drive.google.com/file/d/1qa4Dy53Bj4fUAuw8-GFO2GmDxWFzEUrt

[5]NRADM_USER, NRADM_PASS, CFG_ACL_ENDP (URL of the ACL API used by Configuration APIs), CFG_ACL_CACE (local path and file-name of the CA certificate to be used to validate the self-signed certificate provided by the Node-RED server instance where the ACL API runs, in case a self-signed certificate is used), NRLIB_ENDP (URL of the Node-RED server instance where the Transformation Library runs), NRLIB_USER, NRLIB_PASS, NRLIB_CACE

of the detected values of temperature and humidity both by means of time series at different resolutions and by means of bar charts that carry the average values by day for the last week along with the overall maximum, minimum, and average value; and (iii) display the history of the PLCs statuses by means of horizontal bars and pie charts. Sample Grafana dashboards are represented in Fig. 5.

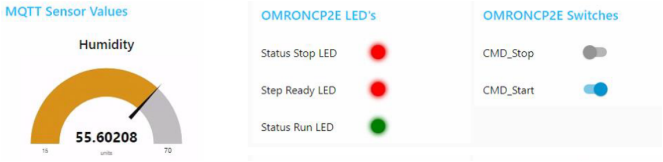

Fig. 4. Node-RED dashboards

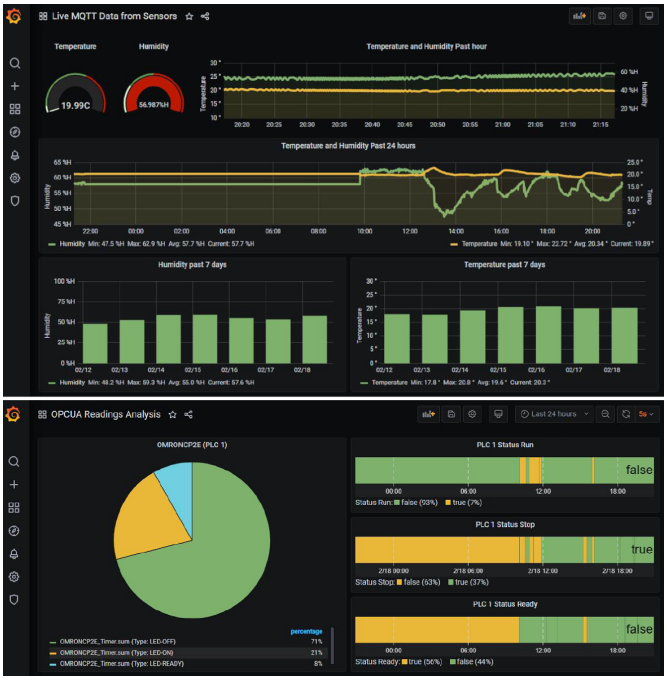

Fig. 5. Grafana dashboards

As for the Ubiquitous System Integration as a Service presented in Section IV, the following scenario has been considered for a proof of concept. An operator, or an automated system, populates a *source* table in a relational database. We want to periodically check for new entries, extract a value of interest, compare it with a configurable threshold, and write a new row in a different database table, namely the *alert* table, when the newly added value in the *source* table is greater than the configured threshold. The goal is achieved through the network of Service Nodes represented in Fig. 6. Each block is a Docker container. All containers are connected to the same user-defined Docker network/bridge. The (host) name of the container is in bold in each block. Docker volumes are used for data storage, and backups of these are available for download[6]. When starting Service Nodes, some environment variables must be set[7]. All communications are over HTTPS and authenticated. The backup of the volume to be used for Service Nodes is not pre-configured , so some rework is needed before it can be restored to Service Node volumes. Service Nodes must be configured[8] after creation and volume restore. The APIs can be called when the internal port 1880 of Service Nodes has been mapped to port numbers from 1993 onward, clockwise from the ExampleDbtail container. The RDB schema is also available for download[9].

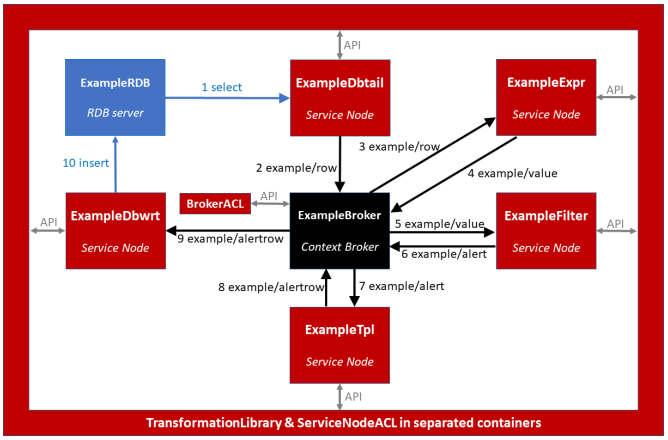

Fig. 6. A proof of concept for the Ubiquitous System Integration as a Service

VI. Future Directions

In the literature, reference is made to the application of automation and Artificial Intelligence (AI) techniques for *smart* system integration [26]. These future IoT systems would include self-configuration, self-optimization, self-protection, and/or self-healing capabilities, that would potentially enable the automated system integration. Presently, the state-of-the-art identifies trends toward enabling this capability, such as the OPC UA Service Oriented Architecture (SOA) to enable interoperability between internal and external factory systems, and MQTT Sparkplug with autonomous data structure sharing. As such, the framework presented in this paper can be expanded upon and improved, yet presently it offers a pragmatic open-source solution to enable ubiquitous systems integration for the new digital workforce to enable smart factories.

Our own next steps will then consist of the following. Firstly, a Scala application will be developed that will act as a server application, and that will be characterized by the following: (i) it will rely on Spark for distributing the computation; (ii) it will rely on some of the most commonly used cloud-storage technologies (HDFS, HBase, Solr, Kudu) as for the distribution of data; (iii) it will allow the building of networks of atomic advanced analytics/ML tasks, fully configurable through API calls; and (iv) it will allow data storage to the cloud, algorithms/networks executions, and status and result reading, by means of API calls. The Scala

[6]https://drive.google.com/file/d/1_vANjmXCVcxhq29JolcSb02bUnpKqKf0

[7]https://drive.google.com/file/d/1D_5bCJGIQOatVx5DxWbVABOsE3jDWz8z

[8]https://documenter.getpostman.com/view/16531967/Tzm5Gbqy

[9]https://drive.google.com/file/d/1RhQ5kWffvroG5f-5lRqbx-DWjpJfB-Rk

server application will be then integrated with Node-RED to make it straightforward to access its functionalities through the creation and configuration of Node-RED nodes. Remarkably, Service Nodes are Node-RED applications. Therefore, we envisage a day when the Edge/Fog layers will come to encompass "cyber devices that compute advanced analytics" with the highest possible flexibility.

VII. Conclusions

In conclusion, in this paper, a number of key technology trends to solve ubiquitous system integration challenges have been identified and described, based on a survey of the state-of-the-art in Industrial Internet of Things (IIoT) technologies. The trends can be summarized as follows: (i) standardized communication with OPC UA and MQTT; (ii) Graphical Programming Interfaces (GPI) with Node-RED flow programming; and (iii) Virtual Containers with microservice deployment in the Docker engine. These enabling ubiquitous technology trends were also formulated into a scalable universal framework which provides foundational digital and smart factory capabilities. Collectively, this framework and its technologies offer potential solutions to ubiquitous system integration challenges, and in doing so provide a means to: (i) standardize the complexity in Cyber-Physical System (CPS) integration; (ii) deploy scalable solutions to meet present and future application requirements; and (iii) systematize the performing of scalable factory-wide digital/smart transformations. Therefore, objectively the key technology trends identified and demonstrated in this paper, represent ubiquitous building blocks and digital tools for next generation engineers to construct, develop, and expand the new digital frontier, namely the digital manufacturing ecosystem. As such, the proposed architecture and implementation for a Ubiquitous System Integration that is fully configurable through APIs can be reviewed as a valid starting point for the building of basic to complex digital ecosystems for intelligent manufacturing.

References

[1] G. Chryssolouris, D. Mavrikios, N. Papakostas, D. Mourtzis, G. Michalos, and K. Georgoulias, "Digital manufacturing: History, perspectives, and outlook," in Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2009, doi: 10.1243/09544054JEM1241.

[2] Y. Bin Zikria, S. W. Kim, O. Hahm, M. K. Afzal, and M. Y. Aalsalem, "Internet of things (IoT) operating systems management: Opportunities, challenges, and solution," Sensors (Switzerland). 2019, doi: 10.3390/s19081793.

[3] J. K. Gerrikagoitia, G. Unamuno, E. Urkia, and A. Serna, "Digital Manufacturing Platforms in the Industry 4.0 from Private and Public Perspectives," Appl. Sci., 2019, doi: 10.3390/app9142934.

[4] S. Yangui, "A panorama of cloud platforms for iot applications across industries," Sensors (Switzerland), 2020, doi: 10.3390/s20092701.

[5] M. Moghaddam, M. N. Cadavid, C. R. Kenley, and A. V. Deshmukh, "Reference architectures for smart manufacturing: A critical review," J. Manuf. Syst., 2018, doi: 10.1016/j.jmsy.2018.10.006.

[6] L. Monostori et al., "Cyber-physical systems in manufacturing," CIRP Ann., vol. 65, no. 2, pp. 621–641, 2016, doi: https://doi.org/10.1016/j.cirp.2016.06.005.

[7] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," IEEE Trans. Ind. Informatics, vol. 14, no. 11, pp. 4724–4734, Nov. 2018, doi: 10.1109/TII.2018.2852491.

[8] I. Grángel-Gonzalez et al., "The industry 4.0 standards landscape from a semantic integration perspective," in IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2017, doi: 10.1109/ETFA.2017.8247584.

[9] T. Kalsoom, N. Ramzan, S. Ahmed, and M. Ur-Rehman, "Advances in sensor technologies in the era of smart factory and industry 4.0," Sensors (Switzerland). 2020, doi: 10.3390/s20236783.

[10] P. Moens et al., "Scalable fleet monitoring and visualization for smart machine maintenance and industrial iot applications," Sensors (Switzerland), 2020, doi: 10.3390/s20154308.

[11] S. Cavalieri and M. G. Salafia, "A model for predictive maintenance based on asset administration shell," Sensors (Switzerland), 2020, doi: 10.3390/s20216028.

[12] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities," IEEE Commun. Surv. Tutorials, 2020, doi: 10.1109/COMST.2020.3011208.

[13] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," IEEE Ind. Electron. Mag., 2017, doi: 10.1109/MIE.2017.2649104.

[14] M. Salhaoui, A. Guerrero-González, M. Arioua, F. J. Ortiz, A. El Oualkadi, and C. L. Torregrosa, "Smart industrial iot monitoring and control system based on UAV and cloud computing applied to a concrete plant," Sensors (Switzerland), 2019, doi: 10.3390/s19153316.

[15] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," Inf. Syst. Front., 2015, doi: 10.1007/s10796-014-9492-7.

[16] Y. Cohen, M. Faccio, F. Pilati, and X. Yao, "Design and management of digital manufacturing and assembly systems in the Industry 4.0 era," International Journal of Advanced Manufacturing Technology. 2019, doi: 10.1007/s00170-019-04595-0.

[17] H. Panetto, B. Iung, D. Ivanov, G. Weichhart, and X. Wang, "Challenges for the cyber-physical manufacturing enterprises of the future," Annual Reviews in Control. 2019, doi: 10.1016/j.arcontrol.2019.02.002.

[18] S. S. Fernández-Miranda, M. Marcos, M. E. Peralta, and F. Aguayo, "The challenge of integrating Industry 4.0 in the degree of Mechanical Engineering," Procedia Manuf., 2017, doi: 10.1016/j.promfg.2017.09.039.

[19] U. Mohammad et al., "Smart factory reference model for training on Industry 4.0," J. Mech. Eng., 2019.

[20] D. Calvetti, P. Mêda, M. C. Gonçalves, and H. Sousa, "Worker 4.0: The future of sensored construction sites," Buildings, 2020, doi: 10.3390/BUILDINGS10100169.

[21] FESTO, "Industry 4.0 User's Guide: Educator Edition," 2020. [Online]. Available: https://www.festo.com/net/SupportPortal/Files/551591/Festo_Inudstry 4.0 User's Guide_Educator Edition_White Paper.PDF. [Accessed: 09-Feb-2021].

[22] Deloitte, "From Interpretation to Prediction Unleashing the Value of the Industrial Internet of Things," 2017.

[23] Gartner, "Gartner.com," 2019. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io. [Accessed: 01-Feb-2021].

[24] Meticulous Market Research, "Industrial Iot Market," 2020. [Online]. Available: https://www.meticulousresearch.com/product/industrial-iot-market-5102.

[25] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," Comput. 524 Ind., 2018, doi: 10.1016/j.compind.2018.04.015.

[26] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," IEEE Trans. Ind. Informatics, vol. 10, no. 4, pp. 2233–591 2243, Nov. 2014, doi: 10.1109/TII.2014.2300753.