Industry 4.0 Synoptics Controlled by IoT Applications in Node-RED

Claudio Badii, Pierfrancesco Bellini, Daniele Cenni, Nicola Mitolo, Paolo Nesi, Gianni Pantaleo, Mirco Soderi
Department of Information Engineering, DISIT Lab, University of Florence, Florence, Italy
<name>.<surname>@unifi.it, https://www.disit.org, https://www.snap4city.org

***Abstract*— The push towards Industry 4.0 is constraining the industries to work in integrated supply chains. This implies to be open to the integration their production plants with other plants, and to provide access to their data and processes. In most cases, this also means to give access at data and flows to their customers and to perform some synchronizations and permit supervision, and may to create integrated control rooms, synoptics and dashboards. This activity is also facilitated by the introduction of IoT solutions with IoT Devices, IoT Brokers, etc., which have a completely different approach with respect to DCS and SCADA solutions usually adopted in the industry for controlling their local productions. In this paper, Snap4Industry with its IoT development environment and framework for implementing the Control and Supervision of Multiple Supply Chains in the view of Industry 4.0 as a service are presented. In particular, the paper describes the motivations/requirements and the actions performed to extend IoT Snap4City 100% open source platform to comply with Industry 4.0 requirements. The main additions for creating Snap4Industry solution have been on: (i) industry protocols, (ii) custom widgets and synoptics Dashboards, (iii) new MicroServices for Node-RED for enabling the usage of synoptics as event driven devices, (iv) automatized process for producing synoptics templates according to GDPR. The research has been founded into SODA R&D RT project.**

Keywords*— *IoT, Industry 4.0, data driven, synoptics

I. Introduction

Most of the Industry 4.0 solutions present control systems as DCS (Distributed Control Systems), SCADA (Supervisory control and data acquisition), PLC (programmable Logic controller), etc. PLC are typically directly connected to the production machines and work with a high rate regular periodic loop by: reading the status of all variables/inputs, computing control outputs, and finally producing outputs to the actuators (the new setting) [Langmann, et al., 2018]. The loop is controlled by precise period since in most cases the computation of the control function depends on the value of the period (T2-T1), such as in the case of PID control (proportional, integrative, and derivative control models), etc. The period corresponds to the sampling period and to the acting period. Above the PLC level, DCS/SCADA solutions are typically applied that may put together more than one PLC solutions with supervision activities. In most cases, the DCS/SCADA solutions provide some HMI (human machine interface) for the control room operation. In some cases, the PLC are replaced by RTU (remote terminal units) more general-purpose computer-based solutions for low-level control resulting more programmable, while PLC are typically programmable by using the so-called Ladder Diagrams. PLC programming are based simple logic equations (substantially a set of clauses in the form: *If<Condition>then<Action>*) visually formalized with the above semantic: reading all data for computing Conditions, computing Conditions and Actions, and finally applying Actions by writing outputs. There are streams of discussions in which DCS is considered more process oriented while SCADA more event driven; but in reality, the providers of those solutions are improving them thus making top level DCS and SCADA products very similar. In fact, in most cases, they are used as interchanging terms when a network of control systems is coordinated to work together, and data collected are stored into some database for historical data observation mainly. In addition, DCS/SCADA also provide suitable control graphic interface which represent the plant as synoptics. The synoptics may represent the status of the plant in real-time with icons, animated icons, numbers, graphs, etc., and in some cases, they are also interactive. This means that the same synoptic elements can be used to send commands to the control network for: changing parameters, turning on/off some switch, controlling valves, etc.

In large industry plants, multiple DCS/SCADA controllers are present and they need of hierarchical solutions for: (i) collecting data with different rates, both inputs and values of the control parameters also acted by the users in the control room, (ii) creating the history of the data values and decision taken, (iii) monitoring the status of the system and eventually generating some higher level alarms and actions, (iv) representing the higher level status of the system with some graphical representation and synoptics. At low level, the DCS/SCADA typically communicate with PLC and devices with MODBUS, Profibus, CanBUS, FieldBus, HART, AS-I, etc., and others low-level protocols, while at higher level, they may communicate with protocols such as OPC/OPC-UA, DDS of OMG, MTConnect, etc.

In the context of Industry 4.0 applications, IoT solutions are going to be integrated in the factory, and among different plants and factories, playing a sort of glue among the different processes, by offering data/status and actions (see **Figure** 1). Among them, the devices such as IoT Edge and IoT Gateways allow to collect sensor data from the plant and close the loop with some actuator. In addition, similarly to RTU they can send data towards the DCS/SCADA or on cloud for telemonitoring/control [Karnouskos et al., 2011], [Lu, 2017]. This approach sees the usage of Node-RED / node.js solutions applied in many case [Taba et al., 2018], [Andrei et al., 2018], to implement more flexible solutions and to take into account in the plant also of other external parameters in the connection of the different plant machines and equipment. In fact, Node-RED can support protocols such as ModBus, OPC/OPC-UA, MQTT, AMQP, NGSI, I2C, TCP IP, RS232, etc., and can be used on hardware based on ARM, i86, etc., and on many operating systems (Windows, Linux, Raspberry Pi, etc.). Thus, a mixt of IoT and Industry 4.0 and also many home-automation protocols (ALEXA, Philips HUE, TP Link, KNX, ENOCEAN, ZigBee, …) are supported on Node-RED. Thus, Node-RED being based on node.js engine in JavaScript, is probably one of the best solutions for system integration. On one hand, Node-RED may be not very effective in terms of performance since the round trip in: reading, computing and acting presents some limitations in terms of performance with respect to RTU developed in native code as C++/C, Assembly, and Python, and even wrt PLC. On the other hand, Node-RED approach can be a suitable solution for implementing IoT Gateways mixing up different protocols and data sources and thus for rapid prototyping and when hight performance are note requested. In fact, there are a number of industrial solutions that use Node-RED as RTU gateway with control logic such as SmartTech [https://netsmarttech.com/page/st-one], for running H24 processes in production lines of relevant customers. [https://iotobox.com/]

In this paper, **Snap4Industry** with its IoT development environment and framework for implementing the Control and Supervision of Multiple Supply Chains in the view of Industry 4.0 as a service, are presented. In particular, the paper describes the motivations/requirements and the actions performed to extend IoT Snap4City platform to comply with Industry 4.0 requirements. Snap4City is 100% open source and it is available at [Https://www.snap4city.org], [Badii et al., 2018], any interested can download and install on local and on cloud without fee. The main additions for creating Snap4Industry solution have been: (i) the addition of a number of industry protocols, (ii) the possibility of producing custom widgets and synoptics and connecting them respecting security into a Dashboard system, (iii) the addition of a number of new MicroServices for Node-RED for enabling the usage of synoptics as event driven devices in/out, (iv) the usage of WebSocket secure for the communication with custom Synoptics and Widgets for dashboards, (v) the automatized process for producing synoptics templates and instances according to GDPR (General Data Protection Regulation European guidelines). Please note that the additions have been incorporated in the Snap4City platform to enlarge the coverage of the solution.

This paper is structured as follows. In Section II, the requirements of a solution for control and supervision of multiple supply chains from cloud, in the view of Industry 4.0 as a service as presented. Section III describes the former Snap4City architecture, and in Section III.A the changes and additions performed to cover the industry derived requirements. In Section IV, the development environment for creating synoptics based on SVG graphics is presented. The aim was to enable the creation of custom widgets and synoptics into dashboards without the needs of manual intervention on the server side, neither on programming, doing all from by setting info on web pages. Section V presents the integration of custom synoptics with the visual programming in Node-RED and thus the MicroServices developed based on WebSockets and their performance. Conclusions are drawn in Section VI.

II. Requirements for Industry 4.0 Applications

As described above, the aim of the research reported in this paper is to design and implement a solution for covering the business logic and the human machine interaction of Industry 4.0 high level applications in the context of high level controls even among different factories working on the same or in any way connected supply chain. Therefore, the main needs of the solution that can cover **Figure 1** can be formalized as follow. *Please note that the context is the usage of the solution for enabling the control and supervision of multiple supply chains based on Industry 4.0, by using IoT, cloud, machine learning and big data technologies*. Thus, implementing in some sense Industry 4.0 as a Service, **I4aaS**.

The solution has to support:

1. functionalities which are present at level of RTU, SCADA and Cloud for telecontrol, from Control Room and remote devices. The support may consist in the integration of the functionalities of these elements and/or the interoperability with already present solutions.

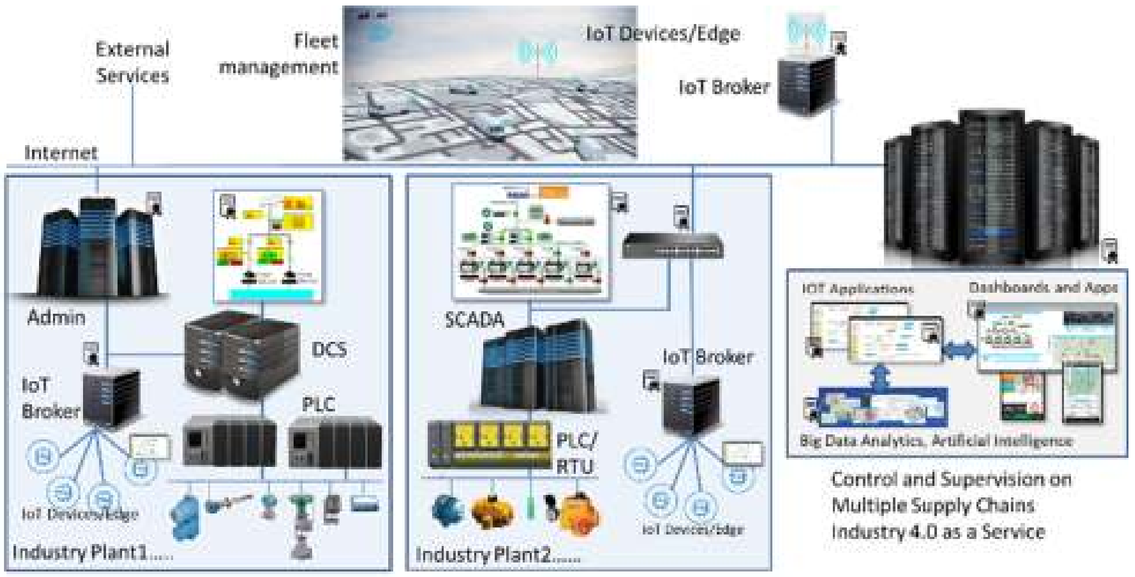

Figure 1: The integration of Industry 4.0 with the support of IoT, Big Data and cloud technologies

2. programming the business and control logics with a uniform visual program, which could be used at low level as well as in the integration on cloud. DCS and SCADA present their own programming languages for implementing business processes, while the remote control or the supply chain integration has to be performed with web-oriented programming. Most IoT development environments are mainly based on developing applications for IoT Devices, or for IoT Edges with native programming languages as happen in the RTU. The adoption of a uniform programming model (at least on Edge and Cloud) provides more flexibility in moving the processes and reusing the code.
3. a large number of different protocols covering classical DCS/SCADA but also IoT, Industry 4.0, home automation and smart city. This means that the language has to be capable of performing requests using many different available protocols [Dizdarević et al., 2019], such as pull protocols to obtain data (e.g., Rest Call, Web services, FTP, HTTP/HTTPS, etc.), and push protocols to receive data via data-driven subscriptions (WS, MQTT, NGSI, COAP, AMQP, etc.).
4. direct communications among business logic processes which can be executed at level of IoT Edge, as well as on Cloud/servers. The possibility of having processes on cloud would guarantee the possibility of exploiting high performance computational capabilities for example for predictions, advanced planning, and integration along the supply chains.
5. registration of IoT Devices which are provided via some IoT Broker, regardless of their protocols. This feature is strongly relevant for the registration of IoT devices integrating the control in the industrial plants. In fact, the data coming from the former generation of production machine are typically insufficient for controlling the production process and distributed supply chain among different plats. Thus, additional IoT devices are added and have to be integrated in the higher-level control and remote monitoring. This would allow the partners of the supply chain to monitor the previous phases of production and thus they can be known in real-time when their activity will be needed. For example, a DCS would present to the higher-level control a subset of data, etc. The data collected from those devices has to be kept strictly private and in accordance with the GDPR guidelines [Badii et al., 2017], [Valtolina et al., 2019].
6. exploitation of big data storage to save and retrieve historical data related to the whole supply chain but managing the access to them according to the GDPR [Badii et al., 2020]. This implies that each factory has to control the access right to its data and provide the access to the other industrial partners of the same supply chain in a controlled manner. The whole data can be used for creating a knowledge base expert system for computing KPIs (Key Performance Indicator) and deductions.
7. applications activated by event driven protocols as well supporting pull protocols. The development model for applications has to provide support for integrating paradigm of data driven with that of functional data processing to become capable to exploit computational resources and data on local/premise (e.g., IoT Edge) as well as on cloud. This means that from IoT Edge must be possible execute computation on Cloud or locally.
8. execution of data analytics as business intelligence and machine learning solutions on powerful processors and thus exploiting historical data, big data store on cloud and on premise if any. For example, for computing: predictions, long terms setting for production plan, detection of anomalies, identification of maintenance actions, etc.
9. exploitation of External Services, for example for collecting: costs of energy/gas, environmental data, contextual marketing data, fluctuation of contextual data, etc.
10. creation of local dashboards on IoT Edge and Gateways, which have to be capable to report and interact with the IoT Applications on local basis.
11. production and exploitation of synoptics for graphic rendering of data and plant status and for collecting actions according to the event driven paradigm. They have to be integrated and controllable directly from the IoT Applications.
12. creation of powerful dashboards (e.g., user interface) for remote control (telecontrol, telemonitoring, control supply chain, etc.) with respect to those that can be produced on Edge computer. Dashboards for remote control have to be capable to represent data and to collect actions from the users, as new events for settings/controls the plant. Communication with Dashboard has to be event driven, secure and according to GDPR. Data visualization should be expressive enough, including values in real time data, time trends, histograms, pie charts, Kiviat, barlines, multitrends, etc., but also actuations as buttons, switch, dimmers, selections, etc.

In addition, the solution has to satisfy a number of **non-functional requirements,** such as: robustness (in terms of availability and fault tolerance), scalability (to be capable of serving from small to very large businesses with corresponding volumes of processes per second), security (authentication, authorization, secure connection, etc.), full privacy respect (compliance with data privacy, according to the GDPR), and openness in terms of being an open source and having the possibility of adding new modules and functionalities, etc.

In this paper, we are focusing the attention on requirements 11 and 12 related to the accessible functionality of providing integrated synoptics also connected with IoT Apps. Therefore, in this paper, the evidence of the changes performed on addressing SVG with WebSocket into the infrastructure have been reported. The possibility of addressing Synoptics has led to create a number of new MicroServices that where not present in the Node-RED library, and thus a huge improvement for the community. Most of the other requirements mentioned above are verified by the former Snap4City solution. This fact is also remarked in the next section and in [Bellini et al., 2018], [Badii et al., 2020], [Badii et al., 2019].

III. SNAP4CITY VS SNAP4INDUSTRY ARCHITECTURE

In this section, the general architecture of Snap4City is presented in **Figure** 2, and the specific evolution and solution

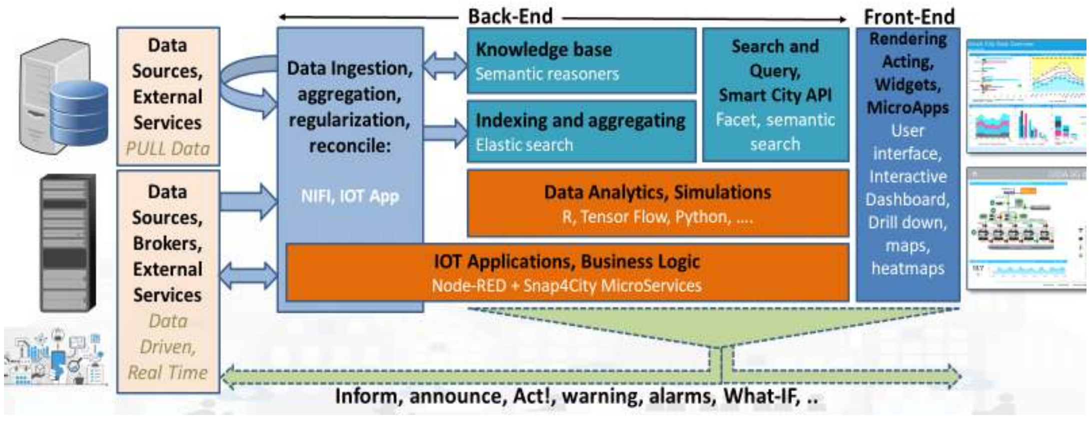
Back-End
Front-End
Data Sources, External Services
PULL Data
Data Sources, Brokers, External Services
Data Driven, Real Time
Data Ingestion, aggregation, regularization, reconcile:
NIFI, IOT App
Knowledge base
Semantic reasoners
Indexing and aggregating
Elastic search
Search and Query, Smart City API
Facet, semantic search
Data Analytics, Simulations
R, Tensor Flow, Python, ...
IOT Applications, Business Logic
Node-RED + Snap4City MicroServices
Rendering Acting, Widgets, MicroApps
User interface, Interactive Dashboard, Drill down, maps, heatmaps
Inform, announce, Act!, warning, alarms, What-IF, ..

Figure 2. Snap4City general architecture.

for addressing the above-mentioned industry 4.0 aspects (and thus Snap4Industry) is presented in **Figure** 3.

Snap4City has been developed in response to the research challenge satisfying all the Select4Cities PCP requirements [https://www.select4cities.eu/]. It allows the creation and management of user communities to collaboratively, (i) may create IoT Solutions, (ii) exploits open and private data with IoT/IoE, respecting GDPR, and (iii) creates/uses processes and IoT Applications that could run on Edge, Mobile and cloud, with the capability of interacting with one another and with users via messages, Dashboards and Applications. In Snap4City, a special attention has been devoted to the creation of an integrated development environment for IoT Apps (cloud and edge), based on Node-RED, with dashboards, and supporting data analytics and resource sharing, thus minimizing any required technical expertise from programmers. To this end, a large number of visual elements developed as MicroServices for the IoT are present. The Snap4City developed tools provide assistance during the development life cycle, from data gathering to dashboard production, as well as in IoT App development, in the sharing of results with other developers and in the community, as self-assessment, together with security support, as described in [Badii et al., 2020]. In Snap4City, data provided from External Services or operators is collected in pull via REST Calls using scheduled processes, as IoT App. IoT Apps are Node-RED Applications which can be executed on cloud as well as on IoT Edge. The collected data are regularized/reconciled, according to the Km4City Knowledge Base [Badii et al., 2018], and then pushed into the Big Data Cluster storage. The reconciliation process allows for the reconnection of data to entities already in place, for example, by connecting new instances with the already in place elements of the knowledge base.

The data arriving in push (data-driven) are managed by several IoT Brokers, at which the Apache NIFI distributed cluster is subscribed. The NIFI process is in charge of performing a regularization/reconciliation task on the data, according to the Km4City Knowledge Base, and then it pushes the resulting data into the Big Data Cluster storage, while indexing and thus creating the Data Shadow. The above-described parallel solution based on NIFI [https://nifi.apache.org/] is normalized in a single approach when the volume of data. The Big Data cluster storage includes an RDF (Resource Description Framework) store for semantic reasoning, and an ElasticSearch Index. The whole solution presents textual indexes on all fields, semantic indexes, and elastic aggregations to perform advanced "faceted" filtering queries [Badii et al., 2018].

The IoT Apps can be executed on-cloud or on-Edge. When IoT Applications are executed on IoT Edge, which is located on the field, it may directly communicate with the IoT Apps or Dashboards on cloud or by means of IoT Brokers (to which all of the others can be subscribed). On such grounds, without the risk of losing generality, in Snap4Industry, the IoT Applications can be obtained as follows:

IoT App = Node-RED + Snap4City MicroServices.

The IoT App exploits the basic nodes of Node-RED Node.JS plus Snap4City MicroServices, which have been extended from the smart city to Industry 4.0. The Snap4City solution has implemented a large set of MicroServices which provide an easy and formalized access to all the Smart City services [Badii et al., 2019]. The Snap4City MicroServices are distributed into two official libraries of Node-RED nodes by the JS Foundation portal. The two libraries are dedicated to final users (basic) and to developers (advanced). The version for Developers (to be installed on top of the basic version for final users) presents several nodes/blocks that can accept complex JSON message inputs to create strongly dynamic IoT Applications. Both Libraries of Snap4City Nodes can be installed in any Node-RED tool on any operating system: Linux, Raspberry pi, Windows, etc.

In the following subsections and sections, the main improvements from former Snap4City and the present solution Snap4Industry are described. The main changes have been: (i) the possibility of producing custom widgets and synoptics and connecting them respecting security, (ii) a number of new MicroServices for enabling the usage of synoptics as event driven devices in/out, (iii) the usage of WebSocket secure for the communication with Synoptics, (iv) the automatized process for producing synoptics templates and instances also supporting GDPR. Please note that the additions have been incorporated in the Snap4City platform to enlarge the coverage of the solution.

A. Snap4Industry Architecture

In **Figure 3,** an instance of the functional architecture for **Snap4Industry** is presented. The example reported is mainly focused on creating an Industry 4.0 supervisors for control room and remote control from cloud (as in **Figure** 1) with the capability of computing **production plan and predictions**. Both, production plans and predictions may be performed several times per day. The data analytic computations need 15 minutes to be performed and produce resulting data for the next 48 hours in advance. The predictions and plans are computed on the basis of the data coming from the:

- DCS/SCADA which includes the status of the industrial plant (different lines of productions) including the status of the physical storages of the products and production stocks silos.
- Administrative data which are the orders and the business targets of the next days.
- External Services which are energy costs, prime matter cost, marketing context, real time data from the commercial chain, data from the delivering fleets, etc.
- Control Parameters are a large set of parameters which are useful for the planning and prediction algorithms. For example, the quality of production, some target key performance indicators, a number of flags which may put offline a segment of the supply chain for example for preventive mantenance, etc.

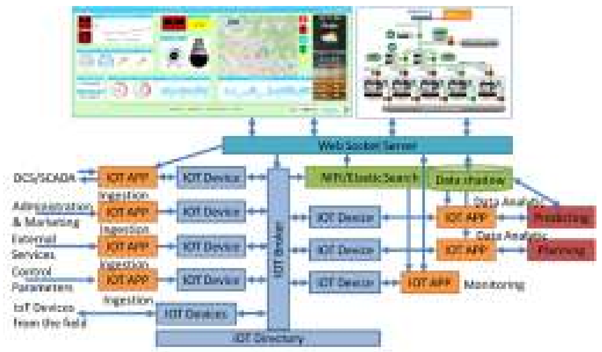

Figure 3. Snap4Industry: example of Functional Architecture. On the user interface: Dashboard with custom widgets and Synoptic panels.

In **Snap4Industry**, IoT Apps are realized as Node-RED processes as Dockers containers on cloud or an IoT Edge (in RTU as Raspberry Pi or Linux). Data is received in push and/or pull, and almost every data message with several attributes is considered an IoT Device instance. To this end, the IoT Devices must be registered, and their data structure/model formalized. In Snap4City, the IoT Device formal registration is performed by means of the IoT Directory [Valtolina et al., 2019], [Badii et al., 2020].

DCS/SCADA data messages are received in push and converted into an IoT Device instance via an IoT App which is posting them on IoT Orion Broker Fi-Ware and from that to the Data Shadow Storage implemented by using NIFI/Elastic Search. **Administrative and Marketing** data coming from the administration of the factory are collected by accessing to the databases -- e.g., Oracle, DB2. This information may include the new target production for the factory, that are the orders, and it is converted into an IoT Device instance by using and IoT App. Every time, a value instance of sensors of an IoT Device is created, it is marked with a time stamp. This allows to recover the acquisition time instant and to create into the Data Shadow the history of the acquisitions and execution of Data Analytics. **External Services** are acquired by means of an IoT App and converted in IoT Devices instance and saved in the storage. A similar operation is performed for the **Control Parameters**. Once the above described data inputs are collected (with different rate of update), the production plan and predictive Data Analytic can be computed. The execution of Data Analytics algorithm is performed by calling them from IoT App and executing Data Analytics in Dockers containers allocated on cloud. The resulting prediction and production plans can be saved into the storage or saved as IoT Devices instance as well, and accessible from Dashboards.

All the processes (IoT App, IoT Brokers, Web Sockets, etc.) are data driven processes activated/fired by the arrival of new messages/events. On the other hand, the data messages in input are differently temporized. For example:

- **DCS/SCADA** may produce a new data set every second. The arrival of the new data on broker leads to send the message to NIFI, which in turn activates the IoT App of monitoring to produce the KPIs to be shown on synoptic via web sockets. With the aim to perform high-level control, to save all messages may be not necessary; thus, a message every 30-15 second could be enough. This may also depend on the velocity of the production process phase.
- **Administrative** data sources produce asynchronous data regarding the commercial and thus production orders. Typically, the arrival of new orders provokes several recomputing for the planning and prediction a few numbers of times per day.
- **External Services** are typically sampled according to their rate of change, for example many times per day.
- **Control Parameters** need to be produced every time a new computation of production plan and/or prediction are performed.
- **IoT Devices from the field** may be sensors and actuators providing data and accepting to perform actions completely asynchronously with respect to the DCS/SCADA processes. They are typically event driven and may adopt IoT protocols towards IoT Brokers registered on IoT Directory.
- **Dashboards** can be used to: (i) just monitor the production or the plant status and in this case they can be given in access to the all partners of the supply chain, (ii) produce events/settings for the general integrated system (e.g., the pressing of a button, the setting of a value), and in this case the access has to be strongly regulated. Thus, the pressing of the button could be performed several times per minute, while the effects on production plan and prediction is relative. The new value is produced as a Web Socket message [Silvia et al., 2018] for a corresponding device that reaches the broker, which in turn, it is sending the message to all subscribed processes and IoT Apps. On the other hand, the actions on the

dashboards can produce some action on DCS/SCADA as well.

In the context of the above example, it is particularly relevant to save the data adopted for the computation of the predictions and planning and the corresponding results. In fact, they may need to be executed again and again with small differences on parameters, with the aim of improving the planning efficiency and the predictive algorithms based on machine learning (also activated on Docker Containers by IoT App). In addition, a number of dashboards are produced for comparing KPI as computed by: (i) planned results and setting of control, (ii) actual production, and (iii) predicted values and controls.

IV. Synoptics Development Environment

In this section, the introduction of synoptics and their integration into the Snap4City solution is presented. This new functionality has constrained to change the architecture as described in the following. A synoptic panel is a graphic representation of the plant, in which graphic elements may present data, animations and can be interactive. For example, to turn on/off a switch, change a valve status, move slider, insert a value, etc. In **Figure 4**, an example is reported. Typically, these graphic layouts are designed in SVG (Scalable Vector Graphics) or other graphics approaches. The advantage of SVG is its integration on Web browsers. The resulting graphics factory layout (synoptic) is typically addressed by proprietary solutions which allow to connect them to DCS and SCADA with proprietary tools. In Snap4Industry all the solution is based on Open Source components and the new developments have been also released on Affero GPL on GITHUB/DISIT, Snap4City.

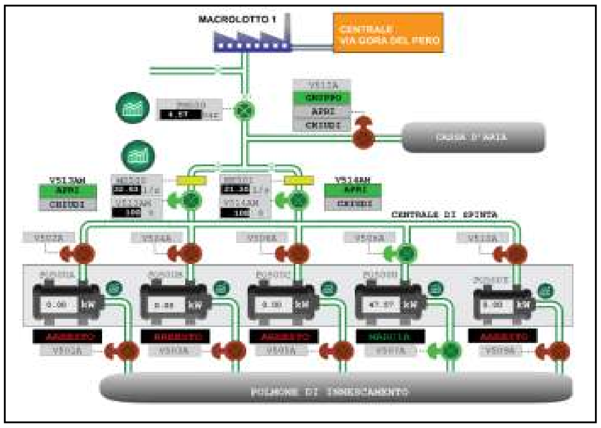

Figure 4. An example of synoptic, GIDA Plant in Prato. Demo also accessible on https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MjE4Mg==

In the context of IoT solutions, the possibility of connecting the Synoptics with IoT Applications has been identified very interesting to cover some of the above-mentioned requirements. In the DCS/SCADA, the usage of the SVG is mainly confined to create synoptics. On the other hand, the power of SVG could be also exploited for creating single interactive/animated graphics Widgets in Dashboards: a single switch, a lamp, an alarm blinking alarm, etc., (as reported in **Figure** 3 on the left). Thus, our approach started with the possibility of using the design of SVG graphics as a generic tool for creating Custom Widget Templates that can be instantiated to create both single widget and complex Synoptics for Dashboards. At the instantiation time the Template is used to create an actual Widget defining the connections with specific WebSocket streams according to **Figure 3**, to play the role of event driven graphic interface.

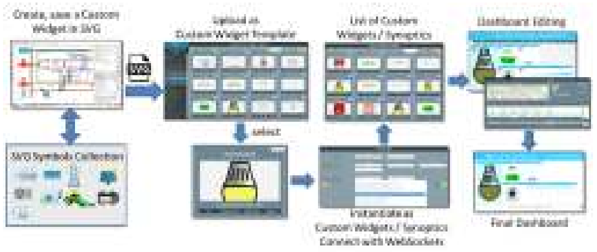
Create, save a Custom Widget in SVG
Upload as Custom Widget Template
List of Custom Widgets / Synoptics
Dashboard Editing
select
SVG Symbols Collection
Instantiate as Custom Widgets / Synoptics Connect with WebSockets
Final Dashboard

Figure 5. The process for producing Custom Widgets into Dashboards for graphic nitration of IoT Control.

In **Figure** 5, the procedure needed to create an SVG based Custom Widget into a Dashboard for IoT Control, is reported. The production starts with the creation of the SVG graphic design by using the Inkscape open source SVG Editor (https://inkscape.org/news/2020/05/04/introducing-inkscape-10/). The production of SVG with the editor has to follow a few of guidelines to be sure that the modality in which variables are defined into the SVG are compatible with the successive steps and Snap4Industry tools. The guidelines are accessible on: https://www.snap4city.org/595. The SVG may: contain animations, visualize effects according to the value of some messages received, collect data from the users, be freely zoomed, provide transparent background, etc. The user may access to the SVG of the Custom Widget Templates to change them, and re-upload as modified / improved versions as well. In addition, they can be created by exploiting a large part of the market and open libraries of SVG symbols. Once the SVG of the Template is created, it can be uploaded into the Snap4City service which parses the file to identify the SVG variables in Writing and Reading. They can be integer, float, status, structured/JSON, and strings. Thus, a **Custom Widget Template** is created as private of the user, according to GDPR, and can be shared with other users or fully published for everybody by the owner.

A Custom Widget Template can be instantiated for creating specific **Custom Widgets/Synoptics**. To this end, the identified variables in Reading and Writing have to be associated with some real variable in the system or directly connected with and IoT App. The Reading value can be also set to constant values. The processes and the different kinds of connections are discussed in the next section.

Please note that, Custom Widget/Synoptics are considered in the Snap4City environment a HighLevelType. So that, it is possible to select them as internal elements in the composition of a Widget for the Dashboards. Then the dashboard widget can be resized, coloured, etc.

A. GDPR Functionalites and Aspects

Please note that: an SVG Template, the corresponding instances as Custom Widget/Synoptic, the dashboards which exploit them, and the related variables connected to the Synoptic: once created are private of the user which has created them. On the other hand, according to GDPR, the user can make them public or can give them in access to other users. When a Dashboard containing a Custom Widget using some MyKPI is made publicly accessible, all its elements have to be public as well otherwise the authentication will be requested to access. In Snap4City/Industry platform, the grant authorization can be provided to Organizations (all users belonging to), Group of Users, and to single users. The accesses can be audited any time, and the grant authorization can be revoked at any time, making the distributed control solution developed on the platform secure and respectfully of the privacy of each industry in sharing their data only to their specific partners. In addition, all the connections are performed via a WebSocket Secure, into HTTPS, thus implementing end-2-end secure communications even on Synoptics.

V. Controlling Synoptics from IoT Apps

In the connection and interaction from Synoptic (read as SVG Custom Widgets) and the rest of the infrastructure, including IoT Apps and storage, there are 3 main Cases as depicted in **Figure** 6.

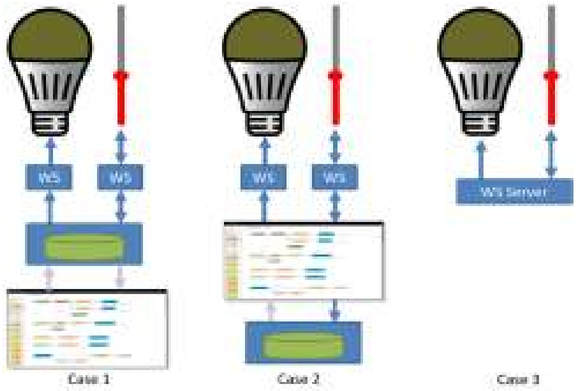

Figure 6. Connections with SVG Custom widget cases.

In **Case 1**, the data can be collected from the Synoptic via WS and saved (connected in writing) or read (connected in reading) directly into/from the storage (namely MyKPI or Sensors data for Snap4City terminology). This means that for changing the light intensity the IoT App writes into the storage and it is the storage that provokes an event in the WS for changing the lamp status. It implies that the velocity for closing the loop, *round trip (moving the slider and observing the change on the lamp)* depends on the velocity of writing/reading on/from the storage. Please note that, when the slider is moved a new value is saved into the database of MyKPI and an event is created into the IoT App and to the WS of lamp connected. This approach increases a bit the velocity of the round trip. The IoT App uses in this case specific storage Nodes of Snap4City to interfaces with WS and Storage. For this specific case, the presence of the IoT App is optional since the WS of the Slider can be connected to same MyKPI variable of the Lamp, thus closing the loop on storage cloud. With the aim of assessing performance, a series of messages have been saved into the MyKPI and thus they are automatically posted on WS for both slider and lamp (in fact the slider is producing a data but also reading the value if changes and move the slider accordingly). Thus, it has been possible to sustain without loss of messages up to 5 changes per second. Which means 10 WS messages per second (change the lamp and change the slider). Please note that, the round loop is closed passing from the web page to the WS Server via internet and on IoT App on cloud, which generates the message. This means that the measure has been performed in real conditions and at the same time is influenced by the capabilities of the internet connection.

In **Case 2**, the Synoptic is connected with the WS Server which can be read/written directly from specific Nodes of Snap4City for communicating with Synoptics. This modality has been added to speed up the process and the limit is imposed by the capabilities of WS server and Node-RED to manage event driven processes. In this case, it is possible to save the value in the storage but is not needed to read it again to close the loop that is closed into the Node-RED IoT App on cloud. With the aim of assessing the performance, the IoT App has been subscribed to the WS of the Slider and once the value it sends the value to Lamp and again on the slider with some random changes, so that to create an infinite loop. From the measures performed the round trip is performed at maximum of 20 changes per second without missing data. Which means 40 WS messages per second (change the lamp and change the slider). Please note that, also in this case, the round loop is closed passing from the web page to the WS Server via internet and on IoT App on cloud, which generates a random value on message at each round.

In **Case 3**, the Synoptics are connected with the WS Server that directly shortcut the reading with the writing and thus the loop is close in nearly real time. Please note that the round loop is closed passing from the web page to the WS Server via internet without the IoT App. The maximum number of messages per second without loss is about 2500 messages per second (3 sender and 1 receiver). On the other hand, this modality is only used for user interaction and may be used for closing the loop from two WS belonging to different parts of the supply chain, for example, an action in the plant and a visualization toward a different partners remotely connected.

Therefore, for connection synoptics with the IoT App, the MicroServices (of the Snap4City library on official Node-RED Palette) reported in **Table I** have been implemented. In addition, the Synoptics variables can be changed by connecting their corresponding WS to MyKPI or Sensors values of the Snap4City Snap4Industry platform directly in configuration/instantiation phase, see **Figure 5**.

Table I: MicroServices of IoT App for Synoptics

event driven my kpi	Send a message into the IoT App once a MyKPI is written.
synoptic subscribe	Send a message into the IoT App once a Synoptic variable is written.
synoptic write	Send a message from the IoT App to a Synoptic variable.
synoptic read	Read the last message of a Synoptic variable.

In the development of solutions, the synoptics may use a mixt of the above-mentioned cases according to the needs in terms of velocity and of saving data changes on storage. In extreme IoT Edge configurations, the IoT App could receive IoT messages from IoT Brokers in push and the data can be directly sent on Synoptic without saving them on storage, or just saving a part of them by subsampling values. Thus, saving storage and at the same time making possible the data collection for machine learning and predictions is possible.

On the other hand, in most of the cases for supply chain integration, it may need to have together: fast events, machines and productions (for example those for manufacturing), and IoT devices just aside to glued. Slow processes for example may be those for: moving material among plants, heating processes, chemical production, electrolysis, milling, etc. In classical DCS/SCADA slower signals are saved in the storage more often than what is really needed, thus limiting the collection of historical data which is mandatory for big data algorithms and predictions. On the other hand, the presented event driven approaches can allow to perform the integration of the different rates with the aim of high-level supervision and control feasible. When the needed rate is not very high as in the DCS solutions closing the loop aside the numerical control.

VI. Conclusions

The strong push towards Industry 4.0 has constrained many industries to work in integrated and interoperable supply chains. In most cases, it implied to open at the integration of their production processes with those of other industries in the supply chain, and thus with their customers and suppliers. This fact led to be capable to give access at data and flows and to create integrated dashboards for their customers and to perform some synchronization and supervision, and may create integrated control rooms, synoptics and dashboards, and allow to collect data for big data analytic, business intelligence, machine learning on production processes. This activity has been in most cases facilitated by the introduction of IoT solutions with IoT Devices, IoT Brokers, etc., which provide a completely different approach with respect to DCS and SCADA solutions usually adopted in the industry for controlling their local productions. In this paper, **Snap4Industry** with its IoT development environment and framework for implementing the Control and Supervision of Multiple Supply Chains in the aim of Industry 4.0 as a service have been presented. In particular, the paper described the motivations, requirements and the actions performed to extend IoT Snap4City 100% open source platform to comply with Industry 4.0 requirements. The main additions for creating Snap4Industry solution have been on: (i) a number of industry protocols, (ii) tools for producing custom widgets and synoptics and connecting them respecting security into a Dashboard system, (iii) a number of new MicroServices for Node-RED to enable the usage of synoptics as event driven devices in/out, (iv) the usage of WebSocket secure for the communication with custom Synoptics and Widget for dashboards, (v) the automation of process for producing synoptics templates and instances according to GDPR. The research has been founded into SODA R&D RT project and tested on chemical plant of ALTAIR, one of the main chemical industry in Italy, and into 5G national project in GIDA plant for water depuration. The specific case of ALTAIR has not been reported for security reason and since it is covered by industry secret.

Acknowledgment

The authors would like to thank SODA project of ALTAIR, and 5G partners as Wind-3, ESTRA, GIDA, Prato, etc., for which we have developed a part of the solutions described in this paper. Snap4City is 100% open source and has been founded by European Commission Select4Cities PCP project under the grant agreement No. 688196. Snap4Industry, Snap4City and Km4City are 100% open source technologies and the platform of DISIT Lab can be accessed at https://www.snap4city.org.

References

- Badii C., P. Bellini, A. Difino, P. Nesi, G. Pantaleo, M. Paolucci, MicroServices Suite for Smart City Applications, Sensors, MDPI, 2019. https://doi.org/10.3390/s19214798
- Badii C., P. Bellini, A. Difino, P. Nesi, Smart City IoT Platform Respecting GDPR Privacy and Security Aspects, IEEE Access, 2020. 10.1109/ACCESS.2020.2968741
- Badii, C.; Belay, E.G.; Bellini, P.; Cenni, D.; Marazzini, M.; Mesiti, M.; Nesi, P.; Pantaleo, G.; Paolucci, M.; Valtolina, S.; et al. Snap4City: A Scalable IoT/IoE Platform for Developing Smart City Applications. In Proceedings of the 2018 IEEE Smart World, Guangzhou, China, 8–12 October 2018.
- Badii, C.; Bellini, P.; Cenni, D.; DiFino, A.; Nesi, P.; Paolucci, M. Analysis and assessment of a knowledge based smart city architecture providing service APIs. Future Gener. Comput. Syst. 2017, 75, 14–29, doi:10.1016/j.future.2017.05.001.
- Bellini P., et al. Smart City architecture for data ingestion and analytics: processes and solutions. 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, 2018.
- Langmann, Reinhard, and Leandro F. Rojas-Peña. "A PLC as an Industry 4.0 component." 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV). IEEE, 2016.
- Silva, Diego RC, et al. "Latency evaluation for MQTT and WebSocket Protocols: an Industry 4.0 perspective." 2018 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2018.
- Dizdarević, J.; Carpio, F.; Jukan, A.; Masip-Bruin, X. A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. ACM Comput. Surv. 2019, 51, 116
- Karnouskos S., and A. W. Colombo, "Architecting the next generation of service-based SCADA/DCS system of systems," *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, Melbourne, VIC, 2011, pp. 359-364, doi: 10.1109/IECON.2011.6119279.
- Lu, Yang. "Cyber physical system (CPS)-based industry 4.0: A survey." Journal of Industrial Integration and Management 2.03 (2017): 1750014.
- Nicolae, Andrei, and Adrian Korodi. "Node-Red and OPC UA Based Lightweight and Low-Cost Historian with Application in the Water Industry." 2018 IEEE 16th International Conference on Industrial Informatics (INDIN). IEEE, 2018.
- Tabaa, Mohamed, et al. "Industrial communication based on modbus and node-RED." Procedia computer science 130 (2018): 583-588.
- Valtolina, S.; Hachem, F.; Barricelli, B.R.; Belay, E.G.; Bonfitto, S.; Mesiti, M. Facilitating the Development of IoT Applications in Smart City Platforms. In International Symposium on End User Development; Springer: Cham, Switzerland, 2019; pp. 83–99.