DOI: https://doi.org/10.46354/i3m.2019.mas.017

INTERNET OF THINGS AND INDUSTRIAL AUTOMATION APPROACH APPLICATION FOR BEEKEEPING PROCESSES AUTOMATION

Anatolijs Zabasta, Nadezda Kunicina, Kaspars Kondratjevs, Leonids Ribickis

The Institute of Electrical Engineering and Electronics, EEF
The Riga Technical University, Riga, Latvia

Anatolijs.Zabasta@rtu.lv, Nadezda.Kunicina@rtu.lv, Kaspars.Kondratjevs@rtu.lv,
Leonids.Ribickis@rtu.lv

ABSTRACT
In this paper we discuss the Internet of Things (IoT) and industrial systems automation approach application for development a prototype of an autonomous beekeeping system. In this research we focus on one of the Arrowhead Framework core systems named the Event Handler system. MQTT service broker being a part of autonomous beekeeping system demonstrates the services provided by the Event Handler System. The proposed services broker applies a visual data flow programming approach. A Node-RED is used to prove viability and advantages of offered architecture implemented in a local automation cloud of autonomous beekeeping System of Systems. The autonomous beekeeping system' prototype provides useful data on beehives and bee apiary status (internal and ambient temperature, humidity, weight of the hives, etc.) to the system users, so they can evaluate the hive status and take further action.

Keywords: Internet of Things, Arrowhead, UML, Wireless Sensor Networks, autonomous beekeeping.

1. INTRODUCTION

Beekeepers, who to engage intensive honey production, have to reallocate bee apiaries several times pursuing better conditions for harvesting of honey products by the end of a summer season. Therefore, they need such monitoring system that ensure monitoring of the hives conditions remotely, e.g. whether the weight of a beehive approaches to maximum, thus the harvesting must be arrange, otherwise, honey production will be interrupted. In winter time, a beekeeper wants to know, whether the inside temperature is critical, if the family is missing feed, to detect, and prevent dangerous deviations in time. In our research we offer a solution that performs bee apiary control without interfering with its processes, which helps to optimize frequency of the apiary inspection. The prototype helps to analyze monitoring data in correlation with video, meteodata, weight changes in time as well as interpretation of nest temperature, humidity and linking to local ambient conditions.

This research is implemented in a frame of the project "Autonomous Beekeeping", which is started at the beginning of 2018. The project is funded by the European Agricultural Fund for Rural Development Program 2014-2020 Cooperation: Support new products, methods, processes and technologies.

The paper is structured as follows. The Chapter 2 provides a review of the literature devoted to application of SOA for industrial systems automation and innovative approaches for automation of beekeeping maintenance processes. The model and composition of a prototype of the autonomous beekeeping system is described in the Chapter 3. Discussion about main discoveries and critical issues is done in the Chapter 4. The review of main contribution of the research and possible directions for future works are discussed in the Chapter 5.

2. REVIEW OF THE LITERATURE

Several researches have been devoted to monitoring of bee colonies. One of them is a project "The Application of Information Technologies in Precision Apiculture" (ITAPIC 2016), which proposed implementation of precision agriculture technologies and methods in the beekeeping. Zacepins, Stalidzans, and Meitalovs (2012) defined the notion of precision beekeeping as an apiary management strategy based on monitoring of individual bee colonies to minimize the resource consumption and maximize the productivity of bees. Altun (2012), Zacepins et al. (2015) depicted a few examples of approaches and solutions: temperature and humidity controlling system, which is powered by solar energy. IoT system prototype for honey bee colony temperature monitoring is described by Zacepins, Kviesis, Pecka, and Osadcuks (2017).

Striving to apply IoT and industrial automation approaches to beekeeping monitoring processes automation we developed a system, which is based on Service Oriented Architecture (SOA) (Alessandrelli, Petraccay, and Pagano 2013; Karnouskos S. et al. 2010), in order to provide highly compatibility with existing and emerging solutions. Gross (2005) developed a model with the purpose of flexible composition and reuse of software artifacts. The method uses UML (Norris 2004) as primary model-based notation for analysis and design

activities. The model applies a notation System or a System-of-Systems (Maier M. W. 1998) as a component that can interact with others through interfaces and can be decomposed in other Systems or components.

Blomstedtet et al (2014) presented the first steps of realizing the Arrowhead vision for interoperable services, systems and systems-of-systems aiming to support the documentation of their structural services. Each service, system and system-of-systems within the Arrowhead Framework must be documented and described in such way that it can be implemented, tested and deployed in an interoperable way. The research (Varga 2016, Delsing 2017) goes beyond Blomstedtet et al (2014), therefore, interoperability in-between almost any service provided by heterogeneous systems are here addressed by the core services that are necessary to meet the requirements and enable a collaborative automation cloud. Both papers present an overview of the framework together with its core elements and provides guidelines for the design and deployment of interoperable, Arrowhead-compliant cooperative systems.

The local cloud concept developed by the Arrowhead Framework (Delsing ed. 2016) addresses many challenges related to IoT-based automation, and is unique in its support for integration of applications between secure localized clouds. The viability of Arrowhead Framework has been demonstrated by implementing of this approach in diverging fields of industrial automation. The case study of small enterprise (Lindström 2018) concerns a multi-usable cloud service platform for big data collection and analytics.

A case study of the service broker, implemented for control of utilities systems in urban environment, is presented in (Zabasta et al. 2018). Arrowhead core Event Handler service s created and tested as a MQTT enabled service broker, for wiring together divergent hardware devices and nodes, and APIs for online services.

A macro-programming model capable to oversee the network of distributed sensors as a whole rather than to program individual pieces is suggested by Newton, Arvind, and Welsh (2005). Giang (2017) and Blackstock, M. and Lea, R. (2014) offered an advanced version of dataflow model to express application logic of IoT devices suitable for large scale IoT, called as "Adaptive Distributed Dataflow".

In our research we studied Data Flow Systems, which provide a data flow-programming paradigm for IoT applications. The WoTKit Processor (Blackstock, and Lea 2012) provides the basic requirements for lightweight toolkits: integration, visualization and processing components and a RESTful API. NoFlo (NoFlo 2019) is a flow based programming environment for JavaScript. It is targeted to ease the development of web applications. Like Node-RED, NoFlo provides a visual editor for a data flow runtime leveraging Node.js. Unlike FRED, NoFlo is not a cloud service; developers install and deploy NoFlo themselves, on its own, or embedded in an application. Glue.Things Composer (Kleinfeld, R. et al. 2014) is designed to connect to other components of the GlueThings platform: Device Manager, and Deployment Manager. A mashup editor is built on Node-RED. Blockly (Blockly 2019) is a client-side JavaScript library for creating visual block programming languages and editors. It is a project of Google and is open-source under the Apache 2.0 License.

Since the IoT has been recognized as a promising solution since in very divergent areas such as public utility systems automation (Kunickis, Dandens, Bariss 2015), efficient transportation systems (Alps I. 2016) centralized healthcare (Sultanovs, Skorobogatjko, and Romanovs 2016), efficient waste management, smart grids, digital tourism, etc., we tested it applicability in an agriculture area, namely in the beekeeping.

In this research we focus on one of the Arrowhead Framework core systems named the Event Handler system. MQTT service broker being a part of autonomous beekeeping system demonstrates the services provided by the Event Handler System.

The proposed smart services broker applies a visual data flow programming approach (Blackstock, and Lea, 2014). A Node-RED (Lewis 2016) was selected among other tools to prove viability and advantages of offered architecture implemented in a local automation cloud of autonomous beekeeping System of Systems.

3. DEVELOPMENT OF THE MODEL OF THE SYSTEM

3.1. Arrowhead Framework Approach for the Autonomous beekeeping

For creation of Autonomous beekeeping system (AB system), we applied the Arrowhead Framework approach that supports the development of generic SOA systems. The Arrowhead Framework (AF) acts as an enabler for systems from different areas: industrial automation, energy production, home automation, smart grids, etc. to facilitate their interaction with each other and exchange information. This multi-area approach can enable considerable savings in terms of efficiency, interoperability and maintenance cost. AF approach promotes the different application systems in an easy and flexible way being able to collaborate successfully due to support provided by the common core services.

The Arrowhead Framework (AF) includes a set of Core Services, which support the interaction between Application Services (e.g. sensor readings, controlling of actuating devices, energy consumption, temperature measurement services, etc.). The Core Services handle the support functionality within the AF to enable application services to exchange information (see Fig.1). The AF is built upon the local cloud concept, where local automation tasks should be encapsulated and protected from outside interference.

Each local cloud must contain at least the three mandatory core systems: Discovery, Authorization and Orchestration, which allow to establish connections between Arrowhead application services (Delsing ed. 2016).

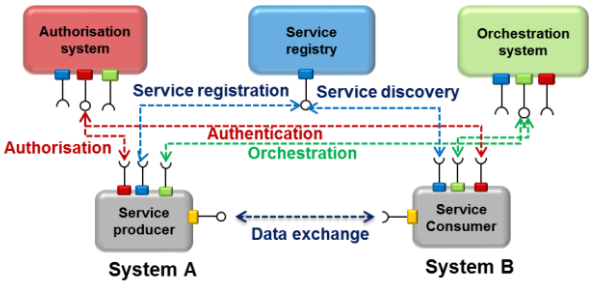

Figure 1: Data exchange between a service producer system and a service consumer system (Delsing 2015).

3.2. Event Handler System and its services

The core system, named as Event Handler System (Albano, Ferreira, and Sousa 2016), belongs to the automation supporting core systems. It provides functionality for the handling of events that occur in an Arrowhead network. The Event Handler System (EHS) receives events from Event Producers and dispatches them to registered Event Consumer. A notation "event" could imply an application service, for example, data provided by water flow sensor, or another event, such as a log of the service failure in case, when EHS is used as a part of the quality of service evaluation system.

There are scenarios, which require a stronger level of decoupling, in terms of space, time and synchronization, when Arrowhead Core Services are used to ensure exchange of services between the systems.

Space decoupling means that a publisher and a subscriber do not need to be aware of each other's location or identities. Time decoupling means that a publisher and a subscriber do not need to be online and actively collaborating in the interaction at the same time. Synchronization decoupling allows asynchronous notification of subscribers by using event services callbacks.

For such scenarios Event Handler System acts as an intermediary between the event producer and the event consumer, providing asynchronous and one-to-many or many-to-many communication model (Eugster et al. 2003). Filtering rules to incoming events could be applied, based on the predefined criteria, for example, on the subscription to the particular services provided by the EHS. In such case, only events, which are interested for Event Consumer, will be sent.

The Event Handler System provides three services – the Registry Service, Publish Service and Notify Service. The SOA approach is implementing due to operations performed in the context of the of the Publish/Subscribe paradigm (see Fig. 2).

The EHS Registry Service is provided in order to store and keep track of the service consumers and service producers in the System of System. If a consumer desires to receive services, or a producer wants to publish services, both need to be registered through this service. At the registration time the producer should advertise the kind of services it produces. The consumer has to specify the filtering rules regarding incoming events by defining a set of conditions to be applied to all incoming events, to be routed to the subscriber.

The EHS Publish Service and Notify Service are used to deliver data regarding the events. A producer system accesses the Publish service of EHS to post the events it produces. The EHS defines, which consumers must receive the event, and then launches the Notify service to provide the incoming event to a particular subscriber of the service.

One of the functions implemented by the EHS is the storage of information regarding events (services) for future access. The storage of information could be implemented locally at the data base of the Event Handler System, or through a Historian Service, which is one of the Arrowhead cores services used to store data (Pereira et al. 2014). To perform this function, EHS proceeds events and routs them on a storage area together with information about a subscriber, which received this event and its meta data. The EHS Get Historical Data service applies filtering rules to permanently stored events (in a database, log file or through the Historian) and retrieves data regarding stored events.

When talking about Quality of Service (QoS) there are always two different parts of delivering an event: a publisher to EHS and a subscriber. It is worth to look at them separately, since there are subtle differences. The QoS level for a publisher to EHS is depending on the QoS level the publisher sets for the particular message. When the EHS transfers an event to a subscriber, it uses the QoS of the subscription made by the subscriber earlier. That means, the QoS guarantees can get downgraded for a particular client, if it subscribed with a lower QoS.

In work (HiveMQ 2015), three Quality of Service levels in MQTT are depicted. Since in this research we developed a MQTT service broker, it looks reasonable to interpret QoS notation defined in (HiveMQ 2015) aiming to depict QoS of the Event Handler System.

QoS 0 – at most once. The minimal level is zero and it guarantees a best effort delivery. A message won't be acknowledged by the receiver or stored and redelivered by the sender. This is often called "fire and forget" and provides the same guarantee as the underlying TCP protocol.

QoS 1 – at least once. When using QoS level 1, it is guaranteed that a message will be delivered at least once to the receiver. But the message can also be delivered more than once. The sender will store the message until it gets an acknowledgement from the receiver.

QoS 2. The highest QoS is 2, it guarantees that each message is received only once by the counterpart. It is the safest and also the slowest quality of service level. The guarantee is provided by two flows there and back between sender and receiver.

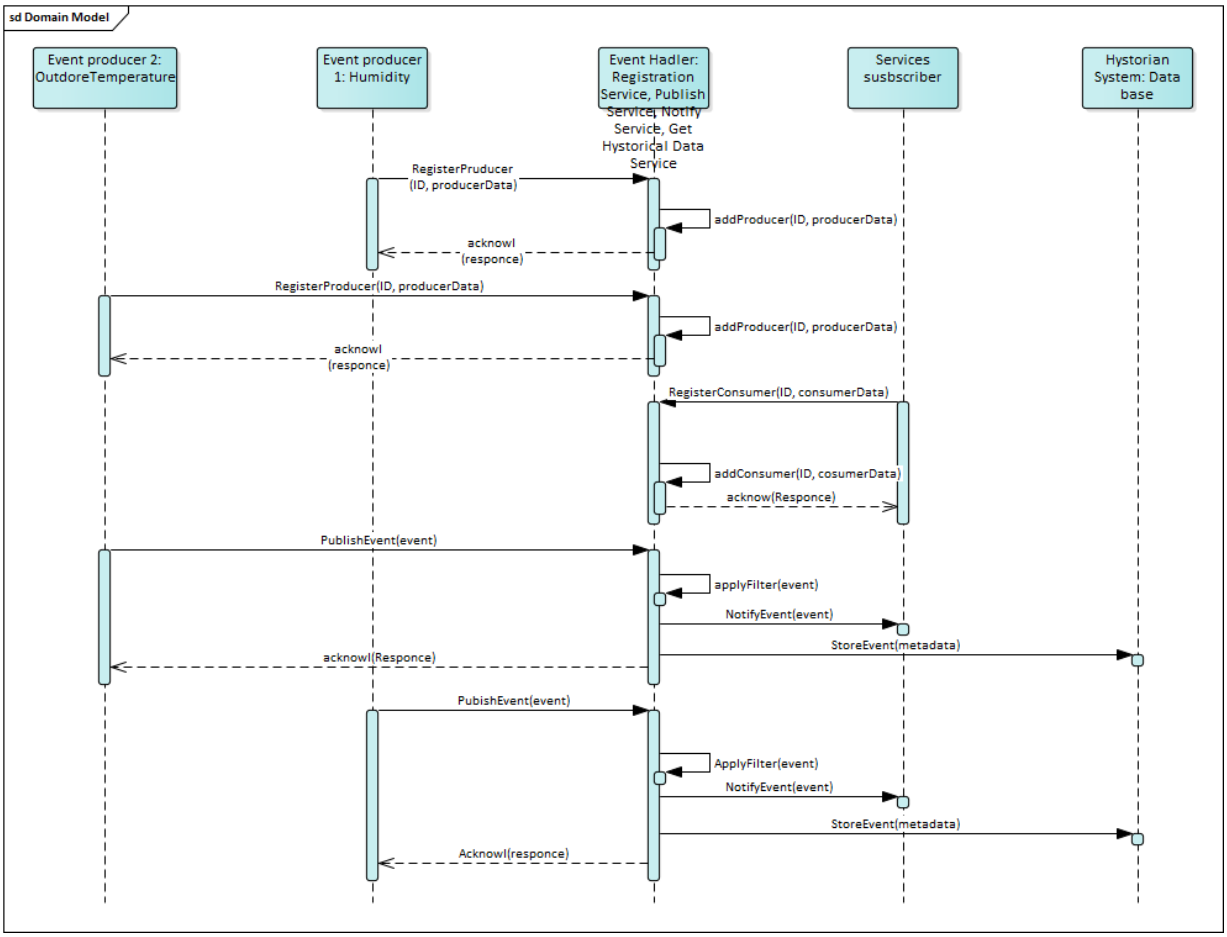

Figure 2: Event Handler System interaction with the systems.

The Event Handler System can handle a variety of communication protocols applied by IoT embedded devices, such as MQTT (Message Queuing Telemetry Transport) (OASIS Standard 2014; ISO 2016; Shelby, Hartke, and Bormann 2014), Constrained Application Protocol (CoAP) and REST (Representational state transfer), and can decode commonly used semantics, e.g. SenML (Sensor Markup Language), XML, JSON (JavaScript Object Notation), and plain text. EHS should be able to convert between protocols in a message exchange between different users. For example, a device can push data to the Event Handler System CoAP or MQTT, while clients can either use MQTT, or poll data using HTTP.

In our research we developed Event Handler System as a service broker, which enables SOA based services and data flow between divergent type of embedded devices and nodes, for example: humidity, wind strength, weight, outdoors and indoor temperature sensors, etc. The implementation of EHS core system as a MQTT service broker is depicted in the Chapter 3.5.

3.3. Autonomous beekeeping System functionality and composition

The first prototype of the autonomous beekeeping (AB) system was tested at five of bee hives allocated at the Riga Botanic Garden during the summer 2018. The weight of bee hives, humidity as well as the temperature sensors, which measure the temperature inside of the beehives and outside temperature, were tested (see Fig.3).

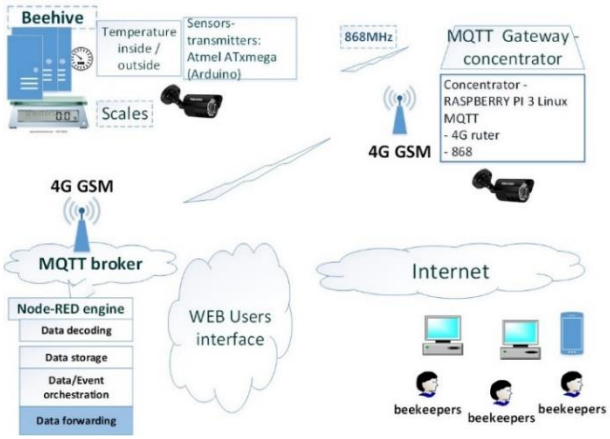

Figure 3: A bock-scheme of autonomous beekeeping system.

A communication system of AB System-of-Systems comprises sensors-transmitters and gateway-concentrators as the elements of wireless sensor network (WSN). The sensors-transmitter consists of a microcontroller, readout interface, power supply and ISM band radio module operating at 868MHz.

The data from the sensor-transmitter are transmitted to the concentrator from sensors using ISM range signal 868 MHz (for example, weight sensor 1C3003AE0030 transmitted data via radio gateway-concentrator).
The transmission range of sensors-transmitters is up to several hundred meters depending on the installation environment. The measurement data are encapsulated in telegrams to be transmitted using Manchester coding and GFSK modulation to one or to multiple gateways that provide an area of coverage.
The gateway node consists or a radio module, GSM GPRS module. Additional or 802.11b/g/n Wi-Fi module that integrates a microcontroller (ESP8266) is available. The Wi-Fi module supports both station: Wi-Fi client mode and access point modes.
A 4G reliable and secure LTE router with I/O, GNSS and RS232/RS485 has been used for communication between gateway-concentrator and MQTT broker at the back-end. Router delivers mission-critical cellular communication and GPS location capabilities.

3.4. The system UML model

In order to create a model of the autonomous beekeeping system we use a methodology represented by OMG's (Object Management Group) solution for system abstraction, modelling, development, and reuse—Model Driven Architecture (MDA) (OMG 2005). The key component of a system modelling, which underlies the principles of MDA, is the Unified Modelling Language (UML) a widely accepted standard for modelling and designing different types of systems. In order to create the UML model of AB system we apply Enterprise Architect Professional modelling tool.

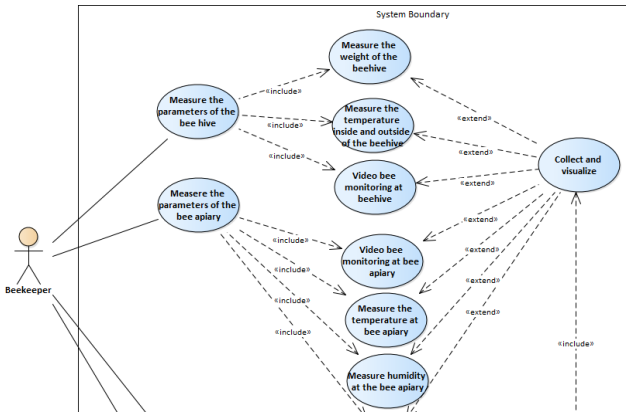

Figure 4: Use case diagram of AB system

Fifteen primary Use cases have been defined aiming to describe AB model (see Fig.4). Among them are Collect and visualize parameters, Measure beehive weight, Measure beehive's internal temperature, Measure the beehive's internal moisture, Monitor energy consumption, etc. We use the actor symbol to represent the agent that activates the use case, probably beekeeper or the other AB system' client, which is willing to analyze the behavior of a bee colony.

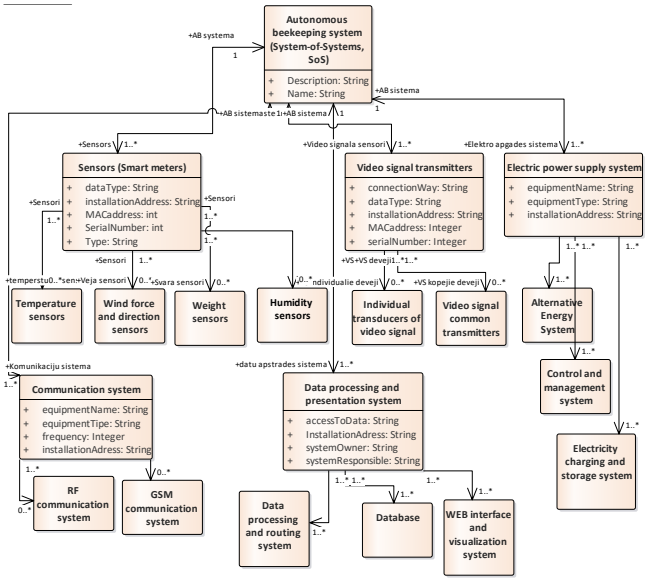

Figure 5: Domain Objects diagram of autonomous beekeeping system

A Domain Objects diagram at Fig. 5 shows nineteen classes, where four of them represent different kinds of sensors with its services components, four classes represent power supply and energy storage, four classes data storage processing and visualization services, two classes data communication services, and etc. The parameters of attributes and operations in classes have been omitted in the interest of figure clarity.
Data processing and presentation system will be explored and described in more details as a MQTT service broker in the chapter 3.5.

3.5. Event Handler System implementation

We implemented Event Handler System as a MQTT service broker that applies a software integration platform Node-RED to interconnect heterogeneous systems in IoT way. The service broker applies Message Queuing Telemetry Transport (MQTT) protocol, which is a Client Server subscribe messaging transport protocol. It is lightweight, open, simple, and designed so as to be easy to implement. These characteristics make it suitable for constrained environments such as for communication in Machine-to-Machine (M2M) and Internet of Things (IoT) contexts.
The beehive monitoring system deployed a Node-RED (Lewis 2016) as a gateway concentrator for wiring together hardware devices (temperature, weight, humidity sensors, etc.), APIs and online services. At the heart of Node-RED is a visual editor allowing complex data flows to be wired together with a little coding skills. Node-RED main functionality is to decode and to route MQTT smart metering data to further service orchestration or use them in external services as monitoring system or external clients (see Fig. 3 and 6).

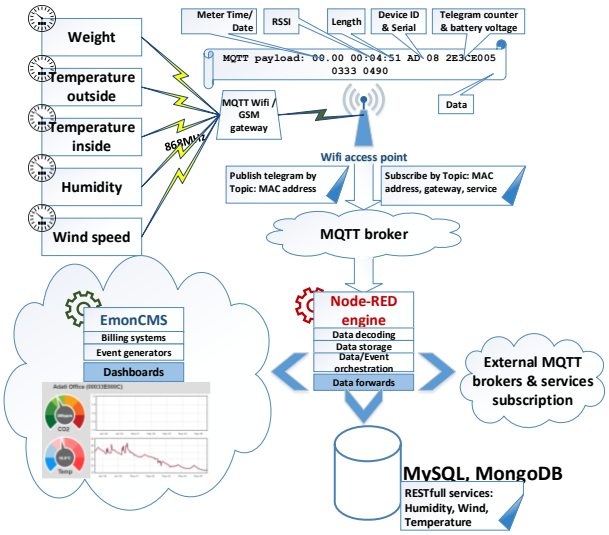

Figure 6: The Scheme of MQTT Service Broker

The services broker depicted in this paper ensures QoS level "0" that is sufficient for the nature of services consumed by the clients of Autonomous Beekeeping system.

Fifure: 7. Outdoor weight scale and sensors installation.

The temperature sensor-transmitters transfer data to the server-broker every 15 minutes, the weight sensor transmits data to the server every 5 minutes. The data collected by the MQTT broker are displayed in graphical form. The users interpret data in graphics or as "dashboard", based on their own experience and understanding of ongoing processes in the hive.

For monitoring of the temperature two sensors are allocated inside of a beehive: one in the center and the second one in the side part of a beehive The third sensor is allocated at the outside wall of the beehive to get measurements of ambient air temperature. The weight of each hive is measured by a specially designed weight platform (Fig. 7). Measurement data are sent by a sensor-transmitter to the gateway-concentrator without additional processing.

Beekeepers and the other project partners are interested to compare behavior of bee colonies and particularly, at the transitions of the seasons, when bees start awaking before to leave hives after the winter time. A multi-chart view provides an opportunity of comparison of bee colonies (see Fig. 8).

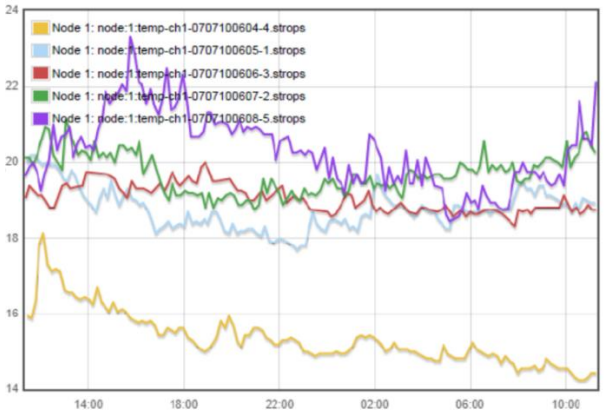

Figure: 8. Multi-chart view of temperature measurement inside of 5 beehives

Node-RED ensure decoding and to routing MQTT smart metering data (Fig. 9) to further service orchestration or/ and for use in external services as customer billing or monitoring systems.

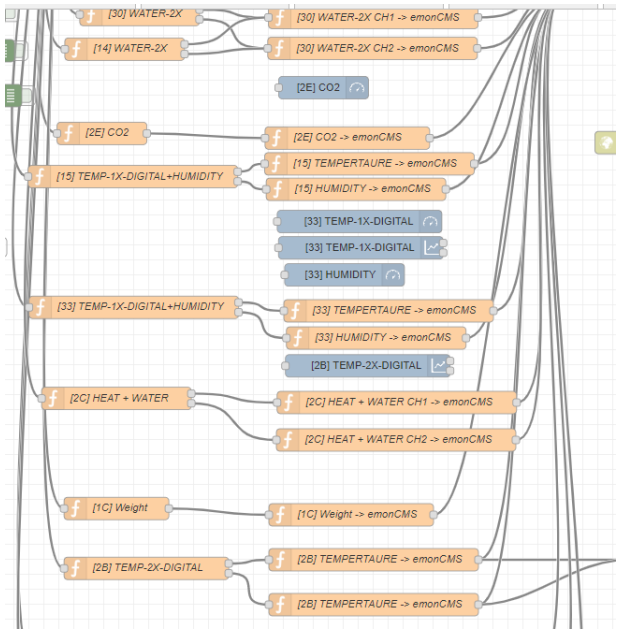

Figure: 9. The nodeRED workflow for wireing of different devices and data processing.

An Inject node (is not shown due to the lack of space) is a testing element to generate a simulated payload for debugging purposes. Below is an example of a sample payload for a sensor with weight data:

MQTT payload: `00:04:40 42 0B 1C3003AE0030 0337 002580`, where 1C3003AE0030 is a sensor ID, RSSI LENSN03, 37 (battery level) and 002580 is data read out.

The task is to decode the payload and forward it to a data storage and visualization service using the IoT approach. The first step is to define a MQTT source, which is a gateway.

Gateways post received metering data to topic based on their MAC address that also serves as a configuration service by subscribing to MAC configuration. The node "SERIAL+VALUE" creates an array of elements from the initial payload (telegram); each array element is processed separately depending on the needed of output or post-processing. A new object is created by dividing name/value pairs, where the crucial elements are: serial number – that identifies the data type and coding format and the unique device id, which makes the metering devices distinct. The second value is the data block needed for decoding procedures. Double output creates two new MQTT messages containing separated data from each sensor, with measurement type topic and devices unique serial number.

For further processing (e.g. monitoring, for serving of external clients) emoncms (Open Energy Monitor) code base is used (node Wight->emonCMS). From the example above a type+device_id object name is generated. After data delivery to emoncms, these data objects are discovered as inputs. Inputs are generic variable that can be processed using emoncms processing engine.

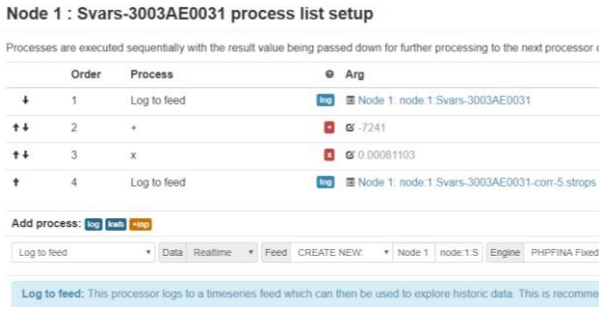

Figure: 9. An algorithm of the block for Query Emoncms API

Before saving information for analysis and representation, pre-calculation of sensors measurement offset values should be done in order to calculate the actual sensor readout value. At Fig.9 an example of "log to feed" value pre-calculation for a *weight* (*Svars* in Latvian) sensor is depicted. Users can customize data transformation, delta offsets, calibration, scaling etc. Typically, the broker does not save MQTT data. This can be done by defining a flow to a data storage, like MySQL DB. In this application, data storage is defined in emoncms with the "Log to feed" processor.

RESTful services can be implemented on demand from the MySQL DB via EmonCMS API by encapsulating the API request into a restful call or by direct definition of a request in form of a RESTful HTTP request. An example of returned value from a request for the latest data of the weight measurement sensor. A request from 03.05.2019 to 05.05.2019 at an interval of 5 minutes from the feed ID=798. GET: http://broker.ventspils.lv:9990/emoncms/feed/data.json?id=798&start=1556443800000&end=1557049500000&interval=900&skipmissing=1&limitinterval=1

```
1556443800000,44938,[1556444700000,44914.666666667],[1556445600000,44987.666666667,….
```

4. DISCUSSION OF THE RESULTS

In our research we applied the Arrowhead Framework approach to develop and document a small SoS using industrial automation and IoT methods. This real SoS operates in a local automation cloud, which comprises core systems and application systems services (temperature, humidity etc.). A core system, namely the Event Handler System, provides several core services: Registry, Publish, Notify and Get Historical Data services.

The use case of autonomous beekeeping system demonstrates, how a MQTT service broker implements the functions and services of EHS. Among the others it provides opportunity to different stakeholders to "subscribe" to the services taking into account a define QsS.

In the future research we plan to add a new SoS to the existing automation cloud, therefore a new automated beekeeping system, which belongs to a different stakeholder, to be incorporated.

We are going to apply advantages of the AF in order to enlarge the automation cloud without significant reconfiguration, which would require much time and efforts. Reconfiguration process should facilitate a correct evolution of the SoSs by updating documentation, systems interfaces, while minimizing the required changes. This could be done thank to:

- Formal description of the SoS structure in UML – SysML.
- Possibility to map the elements of the documentation to engineering procedures.
- Feasibility and benefits of AF approach, which are demonstrated at real-life use cases (see chapter 2).

A concern about security issue should be investigated in the future research, since several stakeholder systems will operate in a single automation cloud.

The research provided in AB project is aware of achievements of the FP7 Framework project ITAPIC, which was devoted to "precision beekeeping" topic. Unlike ITAPIC, the AB project:

- Applies new, more efficient wireless technology and IoT solutions:
- It focusses on development of an integrated / multifunctional (mass, temperature, humidity, meteorological data measurement, video data) system, aiming at application in today's practical beekeeping.
- The system allows remote monitoring not only at individual bee colonies, but also at apiary level.
- We test the AB system at real production conditions, under the supervision of highly qualified and professional beekeepers.

- Recommendations for beekeepers how to interpret equipment data measurements will be elaborated.

The restrictions of the research derive from requirements of the European Agricultural Fund for Rural Development Program. According to the Program, the cost of the prototype must be affordable for potential clients in Latvia, which are represented by farmers, who maintain usually 50-350 beehives. The AB system prototype is developed as a compromise between cost and quality of the system.

5. CONCLUSIONS AND NEXT STEPS

In our research we developed Event Handler System (EHS) as a service broker, which enables SOA based services and data flow between divergent type of embedded devices and nodes, such as outdoor and indoor temperature, humidity, weigh monitoring and humidity sensors. We applied the previous experience, when the EHS provided services have been used for monitoring and control of Smart City systems and utility networks (water supply, district heating, etc.) (Zabasta 2018). In our research, the service broker demonstrates how Arrowhead Event Handler system can be implemented for processes automation at rather different field such as intensive beekeeping.

By now the Event Handler system' working prototype is located and maintained at the "Ventspils Digital Center" (VDC) servers cloud, which belongs to Ventspils City Council as an institution responsible for development and maintenance of ICT infrastructure in the Ventspils city, Latvia. It is planned to migrate ICT infrastructure of AB system from VDC to one of public hosting providers in order to ensure sustainability of AB system beyond the project.

Development of autonomous beekeeping system' prototype is still in progress. The research by the end of the project should be focused on the opportunity to apply alternative energy sources, such as wind and photovoltaic elements, as well on security issues.

ACKNOWLEDGEMENT

The article is based upon the project "Autonomous Beekeeping" funded by the European Agricultural Fund for Rural Development Program, 2014-2020, Cooperation: Support new products, methods, processes and technologies.

REFERENCES

Albano M., Ferreira L. L., and Sousa J., 2016. Extending publish/subscribe mechanisms to SOA applications. 2016 IEEE World Conference on Factory Communication Systems (WFCS), P. 1-4.

Alps I., Gorobecs M., Beinarovica A., Levcenkovs A., 2016. Immune Algorithm and Intelligent Devices for Schedule Overlap Prevention in Electric Transport. No: 2016 57th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Latvia, Riga, 13.-14. October, 2016. Riga: IEEE, 2016, pp. 1-7.

Altun A. A. 2012. Remote Control of the Temperature-Humidity and Climate in the Beehives with Solar Powered Thermoelectric System, J. Control Eng. Appl. Informatics, vol. 14, no. 1, pp. 93–99, 2012.

Alessandrelli D., Petraccay M., and Pagano P., 2013. T-Res: Enabling reconfigurable in-network processing in IoT-based WSNs. In Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCoSS 2013, pages 337-344.

Blackstock M., and Lea R., 2012. WoTKit: a lightweight toolkit for the web of things, in Proceedings of the Third International Workshop on the Web of Things. ACM, 2012, p. 3.

Blackstock, M. and Lea, R. 2014. Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED). Proceedings of the 5th International Workshop on Web of Things (New York, NY, USA, 2014), pp. 34–39.

Blockly, 2019. Google Blockly Homepage: https://developers.google.com/blockly/. [Accessed: March 2019].

Blomstedt, F., Ferreira, L., Klisics, M., Chrysoulas, C., de Soria, I., Zabašta, A., Moris, B., Eliasson, E. Johansson M., Varga P., 2014. The Arrowhead Approach for SOA Application Development and Documentation. In: Proceeding 40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014), United States of America, Dallas, 29 Oct-1 Nov., 2014. Dallas: The Institute of Electrical and Electronics Engineers (IEEE), 2014, pp.2637-2637.

Delsing J., ed., 2016. IoT based Automation - made possible by Arrowhead Framework. CRC Press, Taylor & Francis Group.

Delsing J., 2017. Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions 2017 In: IEEE Industrial Electronics Magazine, Vol. 11, no 4, p. 8-21 Article in journal (Refereed).

Eugster P. T., et al, 2003. Felber P., Guerraoui R., Kermarrec A-M., 2003. The Many Faces of Publish/Subscribe, ACM Computing Surveys 35(2), 2003, pp. 114-131

Giang N., Lea R., Blackstock M., Leung V.C. M., 2016. On Building Smart City IoT Applications: a Coordination-based Perspective, in Proceeding SmartCities '16, December 12-16, 2016, Trento, Italy pages 1-6.

Gross H-G., 2005. Component-Based Software Testing with UML, Springer-Verlag Berlin Heidelberg, ISBN 3-540-26733-6.

HiveMQ, 2015. MQTT Essentials Part 6: Quality of Service 0, 1 & 2, Written by The HiveMQ Team, published: February 16, 2015, [accessed in April 2019].

ISO 2016. MQTT 3.1.1 Specifications. International Organization for Standardization, ISO/IEC 20922. https://www.iso.org/standard/69466.html. [Accessed: April 2019].

ITAPIC, 2016. The Application of Information Technologies in Precision Apiculture (ITApic) ERA-NET ICT-Agri Project, http://www.itapic.eu/index.php [accessed in April 2019].

Karnouskos S. et al., 2010. Karnouskos S., Colombo A.W., Jammes F., Delsing J., Bangemann T., 2010 Towards an architecture for service-oriented process monitoring and control, in: 36th Annual Conference of the IEEE Industrial Electronics Society (IECON-2010), Phoenix.

Kleinfeld, R. et al. 2014. Glue.Things: A Mashup Platform for Wiring the Internet of Things with the Internet of Services. Proceedings of the 5th International Workshop on Web of Things New York, NY, USA, 2014, pp.16–21.

Kunickis M., Dandens A., Bariss U., 2015. Justification of the Utility of Introducing Smart Meters in Latvia. Latvian Journal of Physics and Technical Sciences. Volume 52, Issue 6, 1 December 2015, Pages 13-21.

Lewis K., 2016. Node-RED visual programming for the Internet of Things (IoT) is now a JS Foundation Project. https://www.ibm.com/blogs/internet-of-things/open-source-iot/ [Accessed in March 2019].

Lindström J., Hermanson A., Blomstedt F., and Kyösti P., 2018. A Multi-Usable Cloud Service Platform: A Case Study on Improved Development Pace and Efficiency. In: Applied Sciences, 2018, 8(2), 316, Vol. 8, no 2, https://doi.org/10.3390/app8020316.

Norris D., 2004. Communicating Complex Architectures with UML and the Rational ADS, In Proceedings of the IBM Rational Software Development User Conference.

Maier M. W., 1998. Architecting Principles for Systems-of-Systems, In Systems Engineering, volume 1, issue 4: pp. 267-284.

Newton R., Arvind, and Welsh M., 2005. Building up to macroprogramming: An intermediate language for sensor networks. In: 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005, volume 2005, pages 37-44.

NoFlo, 2019. Flow-Based Programming for JavaScript NoFlo: http://noflojs.org/. [Accessed: March 2019].

OASIS Standard, 2014. MQTT Version 3.1.1, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf. [Accessed: March 2019].

OMG, 2005. OMG Model Driven Architecture, [Online]. Available from: http://www.omg.org/mda [Accessed: February, 2019].

Open Energy Monitor, Open source monitoring for understanding energy, https://openenergymonitor.org/ [Accessed in March 2018].

Pereira et al, 2014. Pereira, Punal P., Jens Eliasson J., and Delsing J. An authentication and access control framework for CoAP-based Internet of Things. In Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE, pp. 5293-5299. IEEE, 2014.

Shelby Z, Hartke K., and Bormann C., 2014. The constrained application protocol (CoAP), RFC 7252 (Proposed Standard), Jun. 2014.

Sultanovs E., Skorobogatjko A., and Romanovs A., 2016. Centralized Healthcare Cyber-Physical System's Architecture Development. In: Proceedings of the 2016 57th International Scientific Conference on Power and Electrical Engineering of Riga Technical University, Latvia, Riga, 13-14 October, 2016. Riga: RTU Press, 2016, pp.153-158.

Varga P., Blomstedt F., Ferreira L. L., Eliasson J., Johansson M., Delsing J, and de Soria I. M., 2016. Making system of systems interoperable— The core components of the Arrowhead Framework, J. Network Computer Appl., vol. 81, pp. 85–95, Mar. 2016.

Zabasta A., Kuņicina N., Kondratjevs K., Patlins A., Ribickis L., Delsing J., 2018. MQTT Service Broker for Enabling the Interoperability of Smart City Systems". International Conference on Energy and Sustainability in Small Developing Economies - ES2DE18. Funchal, Spain, 9 - 11 July 2018, pp.1-6.

Zacepins A., V. Brusbardis V., Meitalovs J., and Stalidzans E., 2015. Challenges in the development of Precision Beekeeping, Biosyst. Eng., vol. 130, pp. 60–71, Feb. 2015.

Zacepins A., Kviesis A., Pecka A., and Osadcuks V., 2017. Development of Internet of Things concept for Precision Beekeeping, Conference: 2017 18th International Carpathian Control Conference (ICCC), P.1-5.

Zacepins A., Stalidzans E., and Meitalovs J., 2012. Application of information technologies in precision apiculture, in Proceedings of the 13th International Conference on Precision Agriculture (ICPA 2012), p.1-6.

AUTHORS BIOGRAPHY

Anatolijs Zabasta, Dr. Sc. Ing., Senior researcher and Projects manager has been working in RTU since 2010. The fields of scientific interests are electrical engineering, embedded systems, critical infrastructure. He was a researcher and project manager in EU projects: ERASMUS+ CBHE, TEMPUS, FP7 ARTEMIS, INTERREG, COST Actions and Latvian State Research Program's projects. A. Zabasta is an author of 75 scientific publications. Contacts: +371-29232872, anatolijs.zabasta@rtu.lv.

Nadezhda Kunicina, Dr. Sc. Ing. Professor, has been working in RTU from 2005. The fields of scientific interests are electrical engineering, embedded systems,

sustainable transport systems, and energy effectiveness in industrial electronics and electric transport. She was a senior researcher and a scientific project manager of FP7 ARTEMIS, INTERREG, COST Actions, ERASMUS+ CBHE, TEMPUS and Latvian State Research Program's projects. N. Kunicina is an author of 139 scientific publications. Contacts: +371-67089052, nadezda.kunicina@rtu.lv.

Kaspars Kondratjevs, PhD student, Researcher has been working in RTU since 2013. The fields of scientific interests are electrical engineering, embedded systems, critical infrastructure, wireless communication systems. He was a researcher in EU projects: FP7 ARTEMIS, INTERREG, COST Actions and in Latvian State Research Program's projects. K. Kondratjevs is an author of 26 scientific publications. Contacts: +371-26123500, Kaspars.kondratjevs@gmail.com.

Leonids Ribickis, Dr. Habil. Sc. Ing., prof., Director of Institute of Industrial Electronics and Electrical Engineering, Rector of RTU, Academician of the Latvian Academy of Sciences. His academic expertise - training of the specialists in the field of electro physics, energy effective lighting, power electronics, equipment design, industrial as well as micro- and nanoelectronics, usage of semiconductors and energy saving. Member of the Board of the World Energy Council Latvian National Committee, Head of the Latvian sub-department of the IEEE. Contacts: +371-67089300, leonids.ribickis@rtu.lv