Heikki Saha*, TKE Oy

An improved IoT gateway and configuration approach for mobile mining machine applications

Abstract: Typical IoT systems have been based on dedicated sensor networks. Gateways (GW) have been designed to route to server the entire communication message by message, without any processing. Further, automatic discovery of sensors has often been used. Data buffering has been used to cope with temporary communication failures only. Modern mining equipment contain electronic control systems, which provide various kinds of data for monitoring. In each site there may be numerous kinds of equipment, each providing different data as different sets. Especially in underground mines, there are areas without any communication network available due to a harsh conditions. This paper presents a new GW concept, which has intentionally been designed to provide a flexible connectivity to various equipment, networks and servers. Intentionally selected data will be extracted into a source independent format. Data buffered by the GW may be sent to the remote server, when a connection exists. Configurability requires well defined management processes to result reliable operation. This paper presents process examples for three major alternative connections, J1939 fleet management, CAN-based superstructure network and a CANopen sensor network. Field tests proved, that the presented GW concept reached the defined targets. The presented configuration processes improved getting the GWs configured without errors.

Keywords: IoT, IioT, gateway, configuration, mining

***Corresponding Author: Heikki Saha:** Dr., E-mail: heikki.saha@tke.fi

1 Background

A set of physical objects – ‘‘things’’ – that send data and communicate with a network was called first time as a term of the Internet of Things (IoT) in 1999 by the MIT [1]. Industrial IoT (IIoT) may be defined integration of internal and external data, which means typically collecting process data from the actual control system and supplemental extra sensors and combing the data in the server [2] [3]. Different requirements apply to IoT and IIoT by means of environmental protection and interfacing [2]. Modern distributed control systems use embedded networks, “fieldbuses” for sharing the control system internal data inside the systems [4]. Similar networks are also in use in mobile assets, but may be called differently, e.g. “automotive networks” [5]. Typical use cases for IIoT data include production process, asset usage and service monitoring and optimization as well as invoicing [2] [4] [6].

Mining operations consist of multiple tasks – drilling, charging, exploding, secondary breaking, scaling, loading, hauling and reinforcing following each other and dedicated equipment are required in each task [7]. Typically such means the use of different kinds of equipment from multiple vendors, which leads into further challenges regarding fleet wide production process and condition monitoring systems.

Modern equipment contain distributed electronic onboard control system, e.g. trucks [5], with a standardized network interface for fleet management [8], which is a typical interface to obtain various kinds of remote monitoring data. In addition to the mandatory FMS (fleet management system) data, equipment vendors may provide optional data [9] as vendor specific sets. Some truck vendors provide similar communication also by so called BBM (body builder module) interface, which is intended for superstructure integration. Additionally, there may be one or more control networks dedicated for superstructure, which may need to be connected. Reliable installation of supplemental sensors has been found to be challenging [2].

Depending on mining method, production areas may vary and be in use for short periods. Thus it does not make any sense to build communication infrastructure covering the entire production areas [7]. There is typically an excellent infrastructure in the fixed areas of the mine, in which the data may be sent to the network. The networks available in the mines are company specific, from which a managed Internet

connection is available through firewall. The networks contain business critical data, why communication is strictly constrained due to security and privacy reasons. Thus, the companies prefer to keep the server's master database strictly under their own control. As a result, each company may use different server framework and further use different application programming interface (API) for interfacing.

In underground mines, cellular networks are typically provided in very limited areas such as service areas, offices and lounges. Main underground communication medium is a WiFi. Cellular networks exist underground, but are more widely used for IIoT in surface mines. However, there may exist random blind spots.

Direct internet connection of an individual sensor is difficult to implemented in a proper way, especially with constrained resources [10]. A gateway (GW) is a device integrating sensors with heterogenous interfaces to a cloud platform over an available network connection [11] [4] [12]. In addition to the data routing, a GW may filter, preprocess and buffer the data [13] [12]. Due to the central role of the GW and its configuration in the IioT systems, GW behavior and related configuration management are in the main scope of this paper.

This paper has been organized as follows. Section 2 contains a review of the major existing GW concepts. An analysis of the adaptability of the existing concepts into mining IIoT GW is shown and improvement objectives set in section 3. Section 4 describes the details of a new concept as three asset connection scenarios. Test experience is summarized in section 5. Discussion on the results is in section 6 and conclusions set in section 7.

2 State-of-the-art

In IoT domain it is typical to have many systems operating in the same area but owned and operated by different parties [6] [14] [15], which also applies to the IIoT domain, including mining. The systems may provide concurrent application using same sensor and actuator data up to some extent [15]. The systems may serve the same master process but each system may provide business value primarily to a single company [6]. Such scenario requires managed data sharing among the systems [15] [6] [14]. In many application domains the sharing is not only a technical problem – data is business critical and shall be available for the owner only or the trusted parties by means of e-commerce or comparable mechanisms [16]. Privacy may also be an issue [6], because e.g. several subcontracting companies competing with each other may operate in the same areas. There is also variation across the applications in data timeliness, which has effect on, how to share the data [14] [11]. Requesting data by multiple consumers [14] is less efficient. Therefore, sharing is typically used instead. Together with the asset operator company silos, data sharing problems apply to the infrastructure after a GW [17]. However, GWs of different parties may need to use the same networks without conflicts.

Performance of GW is one of the critical criteria together with asset specific investment cost [2]. Processing power and data flow analyses are documented approaches to identify application specific performance requirements [13]. Due to a heterogenous selection of source interfaces [13] [10], also the used interfaces, protocols and data structures have effect on the performance demand. In larger systems the performance requirements have been managed by hierarchy, limiting the number of data flows through a single GW [18] [19]. Literature has also identified the significance of communication cost and server data storage and processing cost [20], which may be reduced by moving processing closer to the field and thus reducing the data to be sent to server domain for storage and processing.

Same GW components may be used by different companies and with different assets and servers in order to reduce the GW unit cost. Such kind of sharing leads into the use different server APIs and communication networks [2] [12]. There are server framework specific interface libraries in the market [2]. Such will shorten development time but may degrade re-usability by dependency to the server framework.

GWs are categorized in the literature according their configuration approach.

Basic	A basic GW just forwards all messages sent by the IoT devices upwards to the remote server [12].
Smart	A smart gateway handles data efficiently by preprocessing, filtering, analyzing the data, and delivering only the related or necessary data to the cloud platform [12].
passive	Passive GW needs to configured manually [12]. IoT devices need to

	be configured manually in order to get them registered to the network and communicating with a GW [21] [10].
semi-automated	Like passive one but auto registration of devices to the network and configuring devices through an interface [12] [21] [10].
automated	Fully automated device discovery, registering, configuring and release [12]. still under research [19] [21]. due to lack of relevant processes [18].

Variable protocols and data structuring are well known topics that increase the complexity of GWs [13] [10]. Networks may be wired [2] or wireless, in which there is a limited lifetime due to the use of batteries [2] [21]. Furthermore, sensor configuration management is challenging, especially assignment of unique identifiers for the sensors [22]. An idea of a generic device description exists [23]. It is dedicated to IoT-oriented sensors and is not compatible with existing field buses. Dedicated sensor networks serve also retrofitting of old assets by IoT connectivity [2]. Extracting sensor data into a generic format in the GW has been found to solve the problems of heterogenous sensor interfaces and protocols [11].

Field buses are used as a sensor networks in industrial applications [4]. The most commonly used field buses in machinery applications, such as CANopen, typically support standardized system design and configuration management process improving the efficiency of configuration data management [24]. A comparable system design process for e.g. J1939 looks different [25].

3 Required improvements to the GW concept

Objectives for the concept are set in this section based on the identified requirements. When an existing in-vehicle control network is used as a data source, the GW concepts presented in the literature do not fully apply. Most essential is, that control over the source network design and maintenance is managed by the asset vendor. Nodes, messages and signals may be added, modified and removed based on the vendor's decision. A concept of an automated, smart GW would route the entire in-vehicle communication without any control to a remote server and is thus totally out of sense. Automatic integration of nodes will introduce a significant attack surface, which cannot be accepted in the industrial applications. Communication network to the server has always constrained bandwidth and such automation may overload the entire network infrastructure in the worst case. Thus, instead of the fully automated behavior, a proper process for configuration management shall be used.

O1: Smart and passive kind behavior

In-vehicle communication is heterogenous from communication details point of view, too. E.g. in J1939, there may be different source addresses used by different vendors for same messages. All signals of a message may not be supported by the all vendors. Practically same information may be transmitted as different signals packed into different messages among the vendors. Transmission rate of the messages is designed for in-vehicle controls and is thus too high to be forwarded as is to the server, without downsampling.

O2: Fully configurable data reception for each supported protocol

O3: Interface and protocol agnostic internal data model

The level of heterogenity increases when superstuctures or work hydraulics are considered. Depending on the asset type, there may be one or more 3rd party control networks using arbitrary protocols to be connected to the GW. From GW point of view, dedicated sensor networks and retrofit deployments are the simplest ones, because in such cases the entire network designs are in own control and dedicated to the IoT needs only.

O4: Configuration data management for each supported protocol

Storage capacity for buffering the data in a GW needs to be highly configurable, because part of the assets' typical working areas are outside the range of communication networks. Out-of-area times are relatively short during normal operation, but may be much longer in process exceptions or for certain process phases.

O5: Flexible local data buffering

Different organizations may use different cloud services or other kind of server frameworks. Thus, the use of a same GW product by various organizations

expects a flexible support of various data formats and mechanisms for communication with the server environment. Over-the-air (OTA) update of a GW from server has been found to be challenging [20] and a support of multiple server frameworks increase the challenge. Therefore, a support OTA updates by means of modularity and configurability will be considered in this paper.

O6: Server- and network agnostic, plugin-based uplink

Test assets include trucks, wheel loaders and a secondary breaking machine. Interfacing for each network type shall be solved separately, because of the process differences between the networks. Therefore, configuration management mechanism for J1939, CANopen and raw CAN networks are in the scope of this paper and thus covered by the example scenarios.

4 A new GW concept

A modular, plugin-based structure illustrated in Figure 1 has been designed to cope with arbitrary number and kind of data sources, source networks and communication protocols [11]. Due to heterogenous assets, a highly configurable implementation is needed – connectivity may be expanded by adding supplemental plugins, without any effect on the existing ones.

Modular structure of SW and configuration also helps bandwidth optimization of updates. Each update may cover all or selected items, depending on a need.

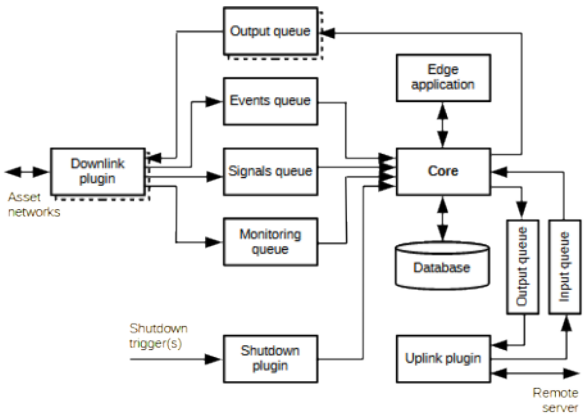

Figure 1: Modular structure of the GW

Instead of manual writing, configurations are intended to be generated from source network descriptions.

Typical source network has been designed by asset vendor. Such means that there may be partial documentation available, in the best case a standard description file, e.g. DBC-file for J1939 network. Supplemental sensor networks are intentionally designed for the IoT implementation. The self designed networks are easier, because the entire network projects with the GW included are available, e.g. standardized nodelist and DCF files in a case of CANopen.

Workflow introduced in this paper expects the use of a standard CANopen and J1939 design processes. Thus, format of the results is well known and may be used as a source for automated workflow. When the 3rd party designs are used, there is messaging of the entire network included. When only a written documentation is available, a formal communication description needs to be manually created as a DBC file.

When the communication description is in a standard DBC format, such may be validated by testing and fixed until correct. In addition to the CAN design tools, most CAN analyzer applications use DBC format for exchanging communication decoding rules, which makes the use of such files very practical. After validation with a real asset one can ensure, that the description is correct and all required signals are included.

Conversion from a validated source projects to GW specific JSON module is performed for each interface. Then the all modules are joined into a single downlink configuration file. Port assignment shall be checked while joining the modules to avoid port mismatches or overlaps, because such cannot be validated with CAN analyzer applications.

GW uses a uniform internal data model based on source agnostic signal samples and events as shown in Figure 2 and Figure 3. Signals and events use separate databases in order to prevent one override another. All signals and events are timestamped by the GW, enabling variable and configurable data buffering.

```
{
  'ts': '<timestamp>'   # Receive timestamp
  'sname': '<name>'     # Signal name
  'sval': 0.1,          # Value
  'sunit': 'm',         # Unit
  'sstat': 0            # Signal status
}
```

Figure 2: Internal signal sample data structure

Existing systems have already defined names.

Therefore, such names are re-used to keep the overall namespace consistent. As a result, the use of the same, application specific namespace everywhere results intrinsic linking of signal data to other data without any supplemental concepts.

Signal unit is included to keep decoding straightforward. Configuration may vary from time to time and possible changes in scaling are easy to handle, when signal value is supplemented by unit information.

Signal status is a way to identify signal sample timeliness. A signal value in process image may be valid but updated last time too long time ago and thus be outdated.

Events are also timestamped as signal samples. Each event is identified as a numeric event code to provide use-case and language agnostic identification. A signal sample value may be attached to each event in order to provide supporting data for consumers.

```
{
  'ts': '<timestamp>'  # Receive timestamp
  'ecode': <code>      # Event code
  'esval': 0.0,        # Signal value
  'enrof': 1,          # Burst size
  'sstat': 0           # Signal status
}
```

Figure 3: Internal event data structure

Events may be generated by the GW itself, its edge application or read by a downlink plugin directly from another device. Events generated locally by the GW itself are typically single ones. Events generated by edge application or read from another device may be combined so, that a number of occurrences since the last read is given in “enrof” attribute. Such enables efficient transfer of event bursts with a reduced processing power and communication overhead. Threre is also a smaller risk of overriding less often occurred events by event bursts.

Event status may be used for e.g. to indicate whether an exception has been asserted or negated.

From processing point of view a GW may generate many kinds of events, but the constraints are set by the typical configuration processes. The typical available metadata consists of minimum and maximum values only, enabling only out-of-range events. More details are provided in the case scenarios explained further.

4.1. Scenario 1, connection to an existing J1939 network

Typically trucks support FMS or BBM providing subset of J1939 signals and messages. Other kinds of machines have J1939 at least for engine and transmission integration.

J1939 allows the use of a “real” J1939 GW interface, which is easy to configure based on source address, priority, suspected parameter and parameter group numbers instead of raw CAN-IDs. The supported signals and messages may be imported from a standard J1939 communication database (DBC) in order to avoid manual typing and human mistakes [25].

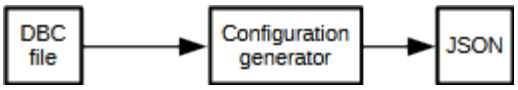

Figure 4: J1939 configuration generated from DBC-file

The asset specific DBC may then be tested by analyzer and fixed if required before generating a GW interface configuration from it according to Figure 4. Such testing is essential, because the messages and signals may exist in the network but marked as “not supported” by a dedicated signal value.

The DBC may contain both standard and custom attributes for data, but typically the standard attributes are commonly used across the tools. CAN-ID, name and cycle time are commonly supported for messages and name, data type, scaling, unit, minimum and maximum values for signals. There does not exist any generic method to control a use of a signal or message for certain purpose, e.g. IoT-connection. Thus, a conversion covers all included messages and signals.

4.2. Scenario 2: supplemental CANopen sensor network

Supplemental sensor networks are typically full, self designed CANopen networks with a GW intentionally included as an integral part of the network. GW acts as an NMT-master, taking care of the network start-up. There are also heartbeat and receiving process data monitoring services available providing status information for each signal. Status may indicate signal invalidity due to e.g. configuration problem or a damaged cable or sensor.

Thus, a complete CANopen project with nodelist and device configuration files (DCF) is available as a result of the standard design process [24]. Figure 5 illustrates, that the interface configuration for a GW may be generated from the CANopen project. CANopen system tools create a DBC file from the project, but it does not contain as much information as the entire project.

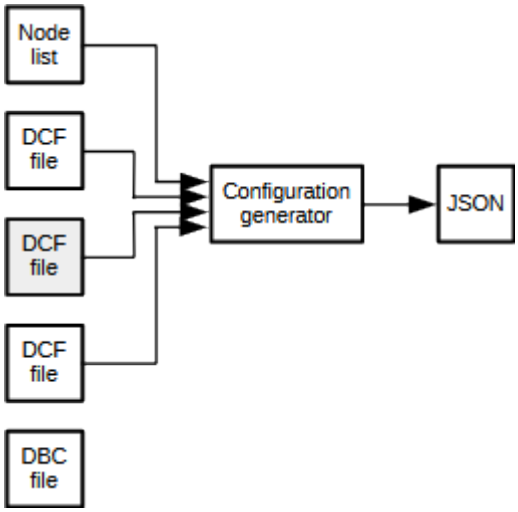

Figure 5: CANopen configuration generated from a CANopen project

Commonly supported message and signal attributes are as constrained as with DBC-files. DCF-files enable the use of custom attributes, but the tools in the market typically support the standard attributes only. A GW is intentionally designed in this scenario, which enables managed exports by relying on the common design practices.

4.3. Scenario 3: connection to an existing CANopen network

When machine control or truck superstructure control networks need to be connected, a 3rd party GW cannot be included into the subsystem network project. Instead, it shall be invisible to the subsystem and just consume specified data from the network. It is assumed that CANopen is used in this scenario.

When a GW cannot actively produce and consume CANopen services, it may only consume the data. Thus, CANopen management services are not fully accessible because of the constrained configuration data available in a DBC-file format. According to Figure 6 the DBC-file is in this scenario the main source for GW configuration. Machine or superstructure vendors do not typically provide such data and the DBC-file shall be reverse engineered.

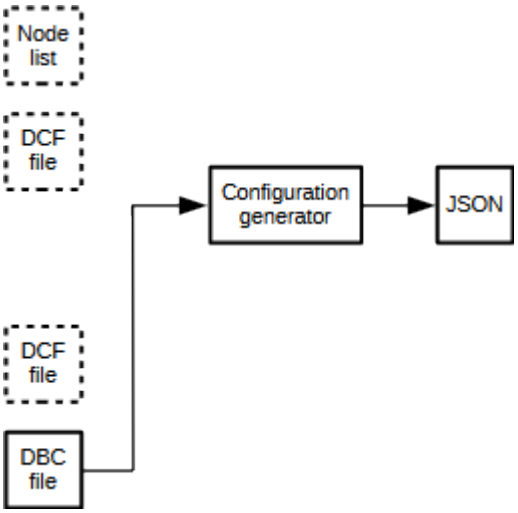

Figure 6: CAN configuration generated from a DBC-file

As in a case of J1939, also in this case a GW configuration DBC may be exported from the covering the entire network. Then, the exported file may be validated by an analyzer application and finally exported to a GW specific JSON.

This scenario is applicable to any CAN protocol, only the source of a system-wide DBC-file may vary from protocol to protocol.

5 Test experience

A GW with smart and passive kind behavior (O1) has been deployed. The major advantage is, that such enabled abstraction of the asset differences by providing harmonized data sets to the server, independent on the asset specific deviations. Uniform data sets improved e.g. efficient re-use of the business logic.

Fully configurable data reception for each supported protocol (O2) was found a major tool to manage the constrained bandwidth of network between GW and server. Sample rates of even with a small subset of the available signals had to be adjusted to avoid overloading the networks.

Interface and protocol agnostic internal data model (O3) enables flexible adaptations and supplementing missing data by dedicated sensing networks but still keeping uplink constant. The use of a high-level, signal oriented internal data model enabled the use of

practically any network type and protocol inside the downlink plugins.

Configuration data management follows the same scheme for each supported type of interface (O4), enabling easy but interface specific configurations. Standardized and commonly used configuration workflows were used. The basic setups were working well, but some standard improvements would help adoption of additional edge processing features.

Lossless data forwarding (O5) has been achieved by flexible local data buffering. Due to a finite buffer capacity, the oldest data may be overwritten, when the designed capacity will be exceeded. The maximum configurable buffer capacity depends on the available capacity of a target HW.

Server- and network agnostic, plugin-based uplink (O6) supports modern way of working. On one hand such enables easy testing of various cloud platforms and in the another hand such keeps it simple to change the cloud platform, if needed. It also provides a freedom to optimize the data structure to optimize data storage and processing in the cloud. Core application functionality is independent of the server connection.

Table 1: Tested target operating systems and processor architectures

Operating system	**x86**	**ARM**
Debian Linux		X
macOS	X	
Raspberry Pi OS (Linux)		X
Ubuntu Linux	X	X
Windows 7	X	
Windows 8	X	
Windows 10	X	
Windows 11	X	
Yocto Linux		X
Ångström Linux		X

Operating systems (OS) and processor architectures, in which the GW with the presented scenarios has been tested, are summarized in Table 1.

HW agnostic SW is easy to scale by just changing the HW, which is often essential due to a tight HW budget [2]. It was found typical to start with an entry-level HW, which had to replaced with a higher capacity one during testing. Adaptations in in-asset GWs are natural due to increasing knowledge during the deployment projects. The use of a flexible GW enabled making the changes locally, without changes in the cloud and business intelligence (BI).

Truck vendors offer warranty only when FMS and BBM interfaces are used and not directly power train or body networks. FMS/BBM data may need to be activated by a vendor's service tool. Description of available data was well documented. There were not documents available for superstructure control networks and control networks of other machines. Thus, thee presented configuration processes with support for reverse engineering and verification by measurement were essential.

Reliable mechanical installation of supplemental sensors was found difficult. In almost each asset there were missing usable process attachments, cable routing paths and free space for the supplemental installations.

6 Discussion

CAN or J1939 configuration needs to be defined as a dedicated DBC-file containing exactly the required messages and signals to enable validation by an analyzer. DBC format would allow custom attributes to control e.g. usage by a GW, but e.g. CAN analyzers do not support data visibility control based on such signal attributes. This may be a point to be generally improved in the future by the tool vendors.

A generic configuration data model covers, but is not limited to CAN, J1939 and CANopen. The data model may be expanded in the future to cover at least DDS, Modbus, NMEA0183 and general-purpose I/O.

Support of supplemental sensor network also applies to retrofitting of older assets. Together with the generic internal data model a retrofitted asset does not necessarily differ from a modern asset from server point of view.

There exist generalized data set specifications for BI layer in the industry, e.g. [26], why it is a good idea to provide data sets supporting such already by the in-asset GWs. Such enables the use of harmonized data streams and APIs in the IT-domain and deployment freedom in the assets – in the OT-domain.

Flexible configurability enables easy adoption of various assets. However, the dominating constraint is uplink bandwidth. Thus, specification of any

deployment shall be started from the throughput constraints between a GW and a server. Then it makes sense to prioritize the selection of signals to be sampled and their sample rates.

Configurability enables scaling the concept usage vertically to other application areas. It is not mandatory to use buffering capability. The GW operates as well between the continuously available networks. It also works with much larger buffering capacity, supporting operation out of uplink up to several days or weeks, if required.

The challenge becomes more significant, when assets operate partially outside the network coverage. Then the question is, how fast the buffered data may be sent to the server? Or do we have sufficient online time to get the data sent and avoid buffer overflow? The answers will vary from application to application and will be found by data flow based analysis.

7 Concluding remarks

Based on tests, the presented modular and HW-agnostic GW implementation solves most of the current challenges in IIoT deployments, especially in mining applications. Loose coupling to the HW, OS and communication network technologies enables efficient updates according to the changes in the related technologies and application requirements.

It was found a significant improvement to export configurations from existing, machine understandable sources instead of writing such manually. Both efficiency and quality were improved.

The identified main area for future development will be data availability from the assets. Standardized FMS and BBM interfaces are commonly available in trucks. Similar kind of open, standardized interfaces will be needed also in other kinds of machines.

8 Acknowledgment

This work has been done in Mechatronics Research Group of Tampere University as a part of the Dig_It project.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 869529.

9 Bibliography

[1] Ding J., Nemati M., Ranaweera C., Choi J., IoT Connectivity Technologies and Applications: A Survey, IEEE Access VOLUME 8, IEEE, 2020

[2] Strauß P., Schmitz M., Wöstmann R., Deuse J., Enabling of Predictive Maintenance in the Brownfield through Low-Cost Sensors, an IIoT-Architecture and Machine Learning, IEEE International Conference on Big Data (Big Data), IEEE, 2018

[3] Poulter A. J., Cox S. J., Enabling Secure Guest Access for Command-and-Control of Internet of Things Devices, IoT Journal, Volume 2, Issue 2, MDPI, 2021

[4] Tsiknas K., Taketzis D., Demertzis K., Skianis C., Cyber Threats to Industrial IoT: A Survey on Attacks and Countermeasures, IoT Journal, Volume 2, Issue 1, MDPI, 2021

[5] Recommended Practice for Control and Communications Network for On-Highway Equipment, J1939-01, SAE, 2000

[6] Brous P., Janssen M., Herder P., The dual effects of the Internet of Things (IoT): A systematic review of the benefits and risks of IoT adoption by organizations, International Journal of Information Management 51 (2020) 101952, Elsevier, 2022

[7] Matikainen R., Bäckström C., Heiskanen R., Koski V., Sundquist P., Valtakari U., Kaivos- ja louhintatekniikan käsikirja, Vuorimiesyhdistys Ry., 1982, 802 p. (in finnish)

[8] FMS-Standard description, Version 04, ACEA Task Force HDEI/BCEI, 13.10.2017

[9] Vehicle Application Layer, J1939-71, SAE, 2006

[10] Kang B., Choo H., An experimental study of a reliable IoT gateway, ICT Express 4 (2018) 130–133, Science Direct, 2017

[11] Yacchirema D. C., Palau C., Smart IoT Gateway For Heterogeneous Devices Interoperability, IEEE LATIN AMERICA TRANSACTIONS, VOL. 14, NO. 8, IEEE. 2016

[12] Beniwal G., Singhrova A., A systematic literature review on IoT gateways, Journal of King Saud University – Computer and Information Sciences 34 (2022) 9541–9563, Science Direct, 2021

[13] Zyrianoff I., Heideker A., Silva D., Kleinschmidt J.,

Soininen J., Salmon Cinotti T. and Kamienski C., Architecting and Deploying IoT Smart Applications: A Performance–Oriented Approach, Sensors 2020, 20, 84, MDPI, 2020

[14] Banaie F., Misic J., Misic V.B, Moghaddam M. H. Y., Seno S. A. H., Performance Analysis of Multithreaded IoT Gateway, IEEE INTERNET OF THINGS JOURNAL, VOL. 6, NO. 2, IEEE, 2019

[15] Pradeep P. Kant K., Conflict Detection and Resolution in IoT Systems: A Survey, IoT Journal,Volume 3, Issue 1, MDPI, 2022

[16] Bröring A., Schmid S., Schindhelm C., Khelil A., Käbisch S., Kramer D., Le Phuoc D., Mitic J., Anicic D., Teniente E., Enabling IoT Ecosystems through Platform Interoperability, IEEE SOFTWARE 0740-7459/17/$33.00, 2017, IEEE

[17] Aziz A., Schelén O., Bodin U., A Study on Industrial IoT for the Mining Industry: Synthesized Architecture and Open Research Directions, IoT Journal, Volume 1, Issue 2, MDPI, 2020

[18] Sifakis J., System Design in the Era of IoT — Meeting the Autonomy Challenge, Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, June 2018

[19] Gelenbe E., Nakip M., Czachórski T., Improving Massive Access to IoT Gateways, Jounal of Performance Evaluation 157–158 (2022) 102308, Elsevier, 2022

[20] Papcun P., Kajati E., Cupkova D., Mocnej J., Miskuf M., Zolotova I., Edge-enabled IoT gateway criteria selection and evaluation,Concurrency and Computation: Practice and Experience 2020;32:e5219, Wiley, 2019

[21] Kang B., Kim D., Hyunseung Choo H., Internet of Everything: A Large-Scale Autonomic IoT Gateway, IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS, VOL. 3, NO. 3, IEEE, 2017

[22] Abboud K., Li Y., Bermudez S., eSNAP: Enabling Sensor Network Automatic Positioning in IoT Lighting Systems, IEEE INTERNET OF THINGS JOURNAL, VOL. 7, NO. 10, IEEE, OCTOBER 2020

[23] Khaled A. E., Helal A., Lindquist W., Lee C., IoT-DDL—Device Description Language for the ‘‘T’’ in IoT, IEEE Access, VOLUME 6, IEEE, 2018

[24] Saha H., Automated workflow for generation of CANopen system monitoring GUI, Proceedings of the 17:th international CAN-Conference, CAN in Automation, 2020

[25] Fellmeth P., Löffler T., Networking Heavy-Duty Vehicles Based on SAE J1939, Technical Article, Vector Informatik GmbH, 12p

[26] International specification for in-service data feedback, ASD F5000S, Issue No. 1.0, ASD, 2016, 694p