ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20210728122543

Preliminary communication

Automated and Controlled Data Collection Using Industrial IoT System for Smart Maintenance

Martin Curman, Davor Kolar*, Dragutin Lisjak, Tihomir Opetuk

Abstract: Maintenance 4.0 is a concept that involves the use of IIoT (Industrial Internet of Things) technology to connect maintenance objects, which enables remote data collection, information exchange, analysis and potential improvement in productivity and efficiency, as well as planning maintenance activities. The purpose of this paper is to present the development of the Industrial Internet of Things data collection system, which relies on Tinkerforge IoT modules, that enables automated data collection alongside control of sensor and data collection parameters. To evaluate the ability of the system, an experiment was conducted where two equipment states were simulated using a rotational equipment failure simulator. The experiment determined that the presented IIoT system had successfully gathered information and that there is a clear distinction in acceleration patterns when simulating two different equipment states.

Keywords: accelerometer; automated data collection; Industrial Internet of Things; Tinkerforge

1 INTRODUCTION

The Industry 4.0 paradigm involves the interconnection of physical objects such as sensors, devices, and enterprise assets to the Internet. Industry 4.0 requires the evolution of ordinary machines into self-aware and self-taught machines to improve their performance, enable sustainable production, and reduce the need for unplanned maintenance.[1] Recently, Industry 4.0 has been considered the future, and today it is widely accepted as a reality that is changing the way companies do business and affecting almost every industry around the world. In the context of the definition of Industry 4.0, access to sustainable production can be linked to production systems developed to be transparent, intelligent, efficient, flexible, mobile and accountable. In this context, various initiatives and approaches have been established to help companies adopt the principles of the fourth industrial revolution in terms of sustainability. Within such initiatives, one of the predominant themes of smart and sustainable production uses a modern approach to maintenance such as Maintenance 4.0 (also called Smart Maintenance). [2]

Maintenance 4.0 is a subset of a smart manufacturing system that uses modern technologies and techniques that can predict when a failure will occur, suggest the best solution, and analyse possible decisions. Predictive maintenance involves the application of sensors to collect data which is then analysed using supervised or unsupervised machine learning methods to obtain a classification of the condition and to predict when a failure will occur. The key technologies involved in predictive maintenance are the Internet of Things (IoT), cloud computing, predictive analytics (machine learning, fuzzy logic, neural networks, evolutionary algorithms, etc.) as well as modern machine repair and parts replacement technologies. In addition to predictive maintenance, the term prescriptive maintenance also appears in Maintenance 4.0. Prescriptive maintenance uses advanced data analysis techniques that allow not only to predict failures and downtime but also to recommend actions to be taken to avoid failures and to optimize maintenance schedules as well as the resources required. Prescriptive maintenance uses a combination of advanced analytical methods, a collection of standardized maintenance procedures, historical maintenance data, and current data collected to predict downtime and propose a solution. [2]

Data collection is achieved by using sensors that translate physical phenomena emitted by the machine into digital signals that represent parameters such as temperature and vibration. However, simple collection of machine data is not enough. Maintenance 4.0 requires an Internet of Things (more precisely an Industrial Internet of Things (IIoT) [3]) infrastructure that reliably connects maintenance facilities to data centres and allows distributed sensor data collection. [4]

The main purpose of this paper is to present the development of an Industrial Internet of Things data collection system that enables automated data collection alongside control of sensor and data collection parameters. For this research, vibration data of rotary machinery fault simulator was collected as an example for demonstrating data collection capability. Vibration is the most popular and widely used parameter in predictive maintenance because the machine vibration response is sensitive to change, and it contains patterns about machine state which can, when analysed, lead to fault diagnosis. [5] This paper will examine the capability of high-frequency data sampling on two machine states (one with normal state and one with combined bearing fault) which will determine if the system is gathering data successfully.

The rest of the paper is organized as follows: **Section 2** overviews the related work in the field of IoT data collection, **Section 3** provides requirements and architecture of the developed system which is showcased in **Section 4** which contains experiment setup and methods. Results are presented and discussed in **Section 5** after which the conclusion is drawn in **Section 6**.

2 RELATED WORK

In this section related work in the field of IoT data collection has been reviewed. The previous research of authors [6] highlights the importance of the Internet of Things and imperfections of traditional data sampling such as human error, lack of data from some machines due to a

limited number of costly data collection devices, etc. To solve these problems, they have proposed to make a rotary machinery IoT enabled by adding adequate hardware to them which can stream continuous data to a web server. They have developed a web application that acts as a dashboard and displays the data. In this research, authors used hardware that supports data stream with up to 2.5 kHz sampling rate with the range of ± 3 g. In addition, the specified settings cannot be changed through the GUI. Finally, continuous data streaming can quickly build up data which poses a problem if there is limited storage space available.

In [7] authors propose an automatic data collection solution for an industrial metal stamping unit, which has the ability of monitoring, predicting and optimising based on collected data from the machine. This system uses several various observed parameters such as acceleration and environmental parameters (temperature, humidity, and pressure). Data is presented through web application that acts as a dashboard for visualization and monitoring of observed parameters. The system presented in this paper provides automatic condition-based monitoring as well as visualization but lacks any control functions which enables the management of the collection process. It should also be mentioned that the system uses Message Queuing Telemetry Transport (MQTT) technology for transmitting data from IoT devices to the server.

Another use case of IoT system is in [8] where authors proposed a smart solution based on IoT and cloud computing technologies to automatically monitor and control machines. However, this paper lacks information about the acquisition parameters. and as was the case in previous paper, the system uses MQTT as a transmission protocol.

Furthermore, in [9] the presented system is using the MQTT protocol for collecting temperature and humidity data. As well as in [6] and [7] the system also uses web application for online monitoring. MQTT protocol uses a publish-subscribe method opposite to the traditional server-client which allows it to have a smaller overhead than traditional HyperText Transfer Protocol (HTTP) protocol. This is useful because overhead size affects power consumption. However, if the number of devices increases, alongside the length of the topics, the overhead size grows linearly with respect to the length of the topics whilst the HTTP header remains constant. [10] On the other hand, the HTTP protocol has higher latency in comparison with the MQTT. This is crucial in cases with high data rates because otherwise there would be delays and missing data. Compromise between these two technologies is the WebSocket protocol. WebSocket protocol enables full-duplex constant data exchange between client and server and is more appropriate for applications that require high data rates. [11] Based on this premise as well as the fact that there are yet no use cases of using WebSocket protocol for vibration data acquisition which we could find, WebSocket is chosen as transmission protocol in the presented system.

The proposed IoT system in [12] is used to monitor large rotor vibrations. This system is closest to what we are presenting because it uses user-defined control modules such as sampling rate, sampling time and axis information. However, this system uses data rate up to 4000 Hz alongside UDP protocol which suffers from reliability issue (unlike WebSocket which is based on TCP protocol). [13] Furthermore, unlike in other papers, the web application that the system uses lacks data visualization modules that can be used for monitoring, and the feedback about the current sampling cycle and sampling status. The last thing that needs to be pointed out is that the sampled data is being saved to an ASCII-file which is locally bound to a single server (PC). A better solution is to store the data into the database from which the data can then be accessed from anywhere as was the case in the previously mentioned papers.

All compared articles have only one accelerometer as a source of data as opposed to presented IoT system which contains multiple accelerometers placed on different parts of the machine.

3 SYSTEM REQUIREMENTS AND ARCHITECTURE

After discussing related works in the previous section, it is possible to define IIoT system requirements and architecture. System requirements are as follows:

1) Data collection parameters input - the user must have a choice to choose from: location of the IIoT device and sensor from which the data will be collected, type of data collection (individual or collective) the frequency and range of accelerometer, duration of sampling and number of samples as well as the time between samples.
2) Data collection – the system needs to collect data based on user-defined setting, store it in a database and record the state of collection (whether it was successful or whether it failed). Also, it needs to inform the user about the progress of data collecting (current number of collecting cycles).
3) Data visualisation – finally, the system needs to visualise data after collecting it.

For better understanding, the system requirements are decomposed and shown in Fig. 1.

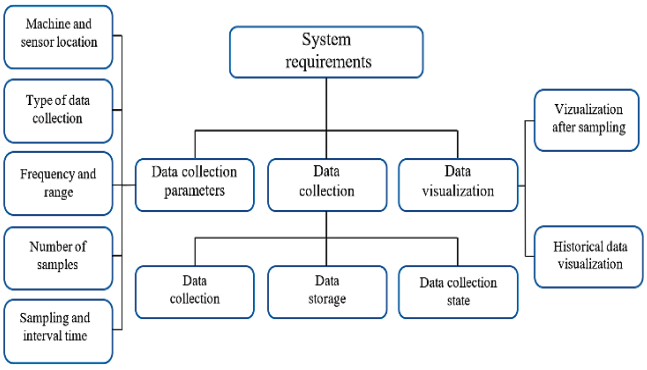

Figure 1 System requirements

To achieve the defined requirements, it is necessary to determine the system architecture. The system architecture is a conceptual model that defines the structure, behaviour, and overview of a system. The system architecture can be seen as a description of a system and its components organized to describe the functioning of the system.[14] The presented system architecture is shown in Fig. 2.

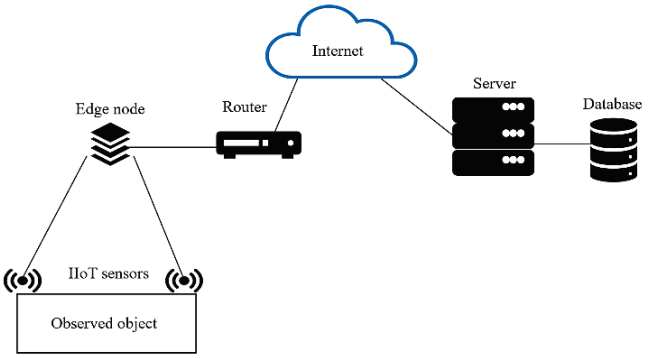

Figure 2 System architecture

The system architecture consists of IIoT sensors (accelerometers), edge node, edge gateway, server, and database. The architecture also has a software part which is a web application. The system components are described below.

3.1 Edge Node and IIoT Sensor

Edge node enables intelligent, automated access and collection, communication and exchange of information with an industrial environment. [15] In this paper, Tinkerforge modules are used as an edge node. Tinkerforge is a system of component blocks for professional use, blocks are widely used to implement projects quickly and efficiently. Blocks can be stacked on top of each other and do not depend on the order of stacking. The integration of blocks with the software is achieved through the use of API which is available for multiple programming languages such as C/C++, C#, Delphi/Lazarus, Java, JavaScript, LabVIEW, Mathematica, MATLAB/Octave, Perl, PHP, Python, Ruby, Shell and VB.NET. [16] Edge node consist of three Tinkerforge modules:

- Master module – is used to control other modules. It routes messages between the other modules on the set and the control device.
- Master extensions - expand the communication capabilities of the module. In this paper, an Ethernet module is used to connect the edge node to the router.
- RED module - can control other modules and execute programs. Programs that control modules can be placed directly on a RED module which is running the Debian Linux operating system located on the Micro-SD card on the bottom of the RED module. It is possible to use several programs at the same time, set the time of their execution as well as their monitoring.

The Edge node is stacked into three tiers, each tier containing one Tinkerforge module. The stack is configured in the following order:

- Tier 1: RED module.
- Tier 2: Master module.
- Tier 3: Ethernet master extension.

Fig. 3 shows the tiers with the modules and their parts.

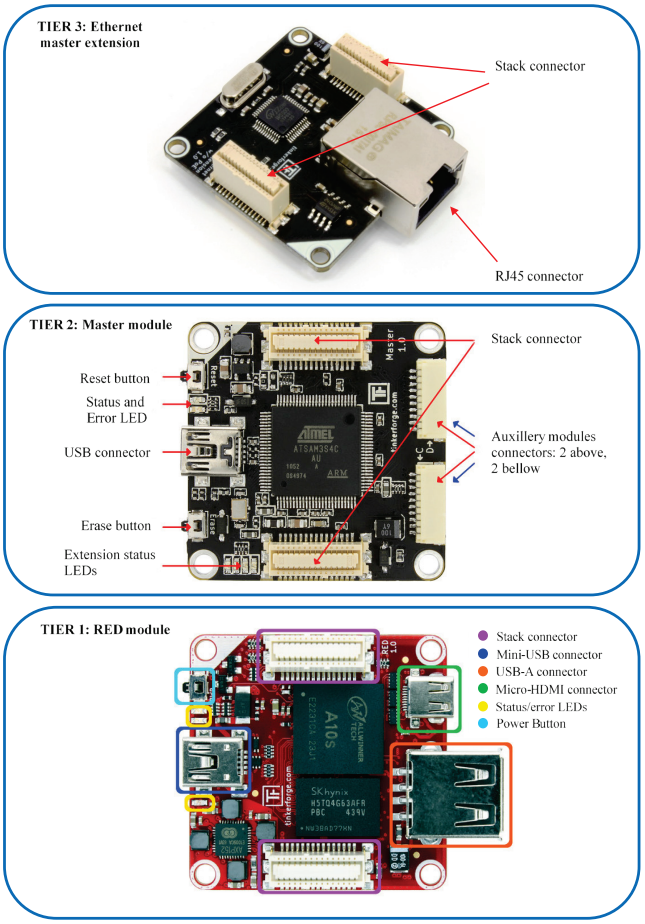

Figure 3 Edge node modules stack

Power to the stack is supplied through a 5 V USB plug which is connected to the Master module. The RED module provides computing power, the main module allows communication and connection of IIoT sensors while the Ethernet extension allows communication over the Internet. These three modules are the basis of the IIoT data collection system, and they must always be present (only the Ethernet extension can be replaced with a Wi-Fi extension). It is possible to add additional modules that allow some other functions (connection of more than 4 sensors, additional power supply, etc.) to the stack.

Tinkerforge also offers auxiliary modules. Auxiliary modules are used to expand the functionality of the master module. Their application can be in the form of measurement (sensors) or control. Each auxiliary module is connected to the other modules via a wired connection. As an IIoT sensor auxiliary module Accelerometer 2.0 was used. It is a three-axis accelerometer designed as an add-on for the main modules and allows the measurement of acceleration in x, y and z direction with the sampling frequency up to 25.6 kHz, which is suitable for use in predictive maintenance. [17] The accelerometer is Kionix KX122-1037 and it has a ± 2·g, ± 4·g, or ± 8·g collection range. The sensing element of acceleration is based on the principle of differential capacitance resulting from the movement of the sensing element caused by acceleration. The sensing element then uses a noise reduction process to reduce errors due to process

variation, temperatures, and other atmospheric influences. The sensing element is hermetically sealed, and a separate ASIC device allows signal conditioning and the application of intelligent control that can be programmed by the user. The accelerometer comes with a built-in analogue-to-digital converter, which reduces the need to purchase a special device that serves as a converter. [18] Fig. 4 shows the Tinkerforge Accelerometer 2.0.

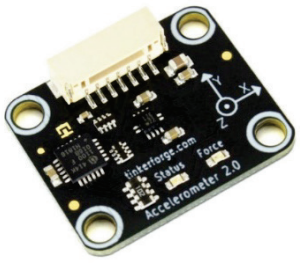

Figure 4 Tinkerforge Accelerometer 2.0 [17]

For proper use of the accelerometer, it is recommended to mount it in a housing so that the measured values are more accurate, and that the accelerometer is additionally protected from external influences. The technical specifications of the accelerometer are given in Tab. 1.

Table 1 Accelerometer specification [18]

Property	Value
Output data rate	0.781 Hz - 25.6 kHz
Full-scale range	± 8 g
Sensitivity	4096 - 16384 counts/g
Offset	± 20 mg
Non-Linearety	0.6 %
Resolution	0.0001 g, 16-bit
Input voltage	1.71 – 3.6 V
Current consumption	145 mA
Output voltage	1.368 - 28.8 V

The highest throughput (sampling data rate) depends on the selected resolution. Tab. 2 shows the dependence of the number of axes and the sampling data rate

Table 2 Dependence of the number of axes on the throughput of the accelerometer [17]

Number of axis	Throughput (8-bit) (Hz)	Throughput (16-bit) (Hz)
1	25 600	25 600
2	25 600	15 000
3	20 000	10 000

The number of used axes can be selected via the Tinkerforge accelerometer API. The IIoT data collection system uses three axes and a resolution of 16 bits which means that the highest possible data rate is 10 000 Hz.

3.2 Router, Web Server and Database

The MikroTik RB951-2nd hAP router is used as the gateway of the system, which is suitable for smaller systems that do not require a lot of users and traffic. Due to its small size, it is suitable for use in almost any location. Microsoft Internet Information Services is used as a web server because of the native support within the Microsoft Visual Studio development environment. The system also uses the Microsoft SQL Server database management system for creating, editing, and controlling the database.

3.3 Web Application

Tinkerforge modules can be controlled via the Application Programming Interface (API). An application programming interface is a way in which two computer applications communicate with each other over a network (Internet) using a common language that they both understand. [19] As mentioned before, Tinkerforge API supports a range of programming languages. The IIoT data collection system uses the JavaScript programming language API. Because JavaScript is a programming language that runs on the frontend, the data collection process will take place on the user's front-end while the data storage function will take place on the back-end. The structure of the web application for automated data collection is shown in Fig. 5. To connect the client to the edge node, a ZeroTier VPN connection was used which enables security and privacy.

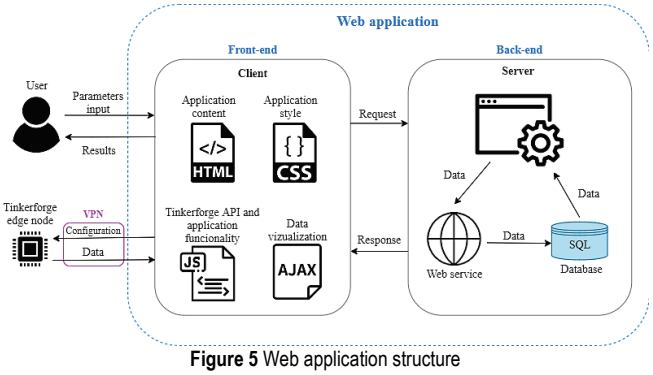

Figure 5 Web application structure

The front (client) side consists of HTML, CSS, JavaScript, and AJAX programming languages, and each of them serves a specific function. The function of HTML is to organize and display the content of the application. CSS defines element sizes, colours, text style, and all graphical user interface settings. For the web application to be used on different device sizes (computer, tablet, mobile phone ...), the Bootstrap CSS library is used, which contains a set of commands for easier manipulation of the size of the elements. JavaScript ensures the functionality of the application by communicating with the Tinkerforge edge node through which data is collected. Also, it serves to check parameter input, manipulate HTML text and objects, perform arithmetic operations, and visualize data. For data visualization, a Plotly JavaScript graphic library was used, which allows plotting data in the form of a graph. Using AJAX (Asynchronous JavaScript and XML) web application can asynchronously send and retrieve data from the server without interfering with the display and behaviour of the web page. The back-end (server) of the web application is used to retrieve data from the database, forward data to the web service, as well as monitor the state of data collection (number of collected samples). Web service is a special part of the application that is used exclusively to store

acceleration values in a database. The back-end of the web application uses the C# programming language. The interface of the application can be seen in Fig. 6.

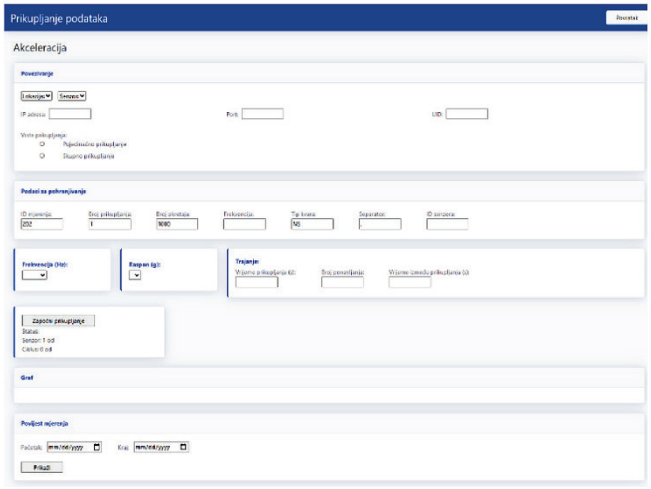

Figure 6 Web application interface

4 EXPERIMENTAL SETUP AND METHODS

The system performance is evaluated by using a rotational equipment failure simulator. The rotational equipment failure simulator is used to investigate and understand the various vibrational patterns, which simulate work at a normal (correct) condition of the equipment and in different states corresponding to the failures of individual parts of the machine. The data collected by the failure simulator is used for further analysis and research. The failure simulator uses a three-phase asynchronous electromotor whose speed is regulated using the frequency inverter containing the speed regulator. By changing the output frequency of the motor, the speed control is enabled, thus reaching the variable frequency of rotation ranging from 0 to 6000 rpm. The engine power is transmitted to the shaft over the claw coupling. The shaft is embedded with two ball bearings mounted with concentric safety rings and has a mass weight of 5 kg which increases the base load of the bearings. By changing the bearing type it is possible to simulate the following malfunctions [20]:

- Damage to rolling elements (BBF),
- Damage to the outer roll path (ORBF),
- Damage to the internal roll track (IRBF),
- Combined damage (CBF).

Also, with the simulation of the bearing failure, it is possible to simulate rotor failure such as:

- Rotor imbalance (IMRF),
- Rotor slot (CRF),
- Rotor eccentricity (ERF).

The IIoT sensors were mounted to the front and rear bearing and then connected to the edge node. The edge node was connected by a UTP cable to a router that connects to the Internet. The layout of the fault simulator, the IIoT sensor, the edge node and the router are shown in Figure 6. The data is collected from the rear bearing as the change is made on it. A laptop serves as the access to the web application that establishes a connection, assigns the sampling parameters, and stores data. The laptop is in another location to demonstrate remote data collection functionality.

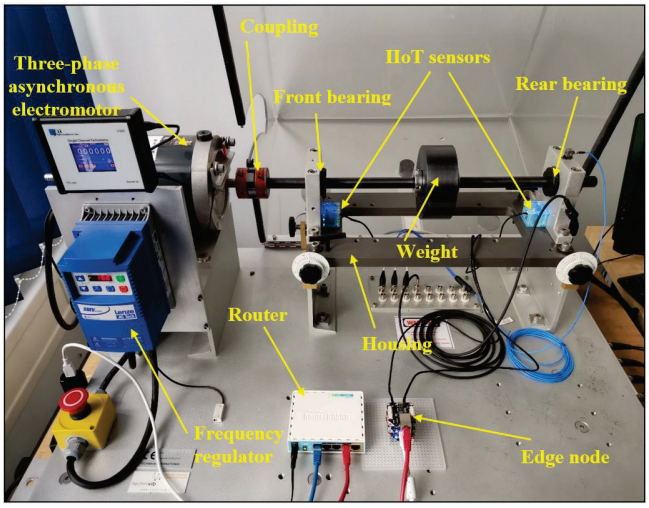

Figure 7 Experimental setup

For this paper, to showcase the operation of the system, a bearing fault experiment was concluded with combined damage to the individual elements (CBF), whose vibration patterns are compared to the normal state of the equipment (NS). The experiment parameters are in Tab. 3.

Table 3 Experiment parameters

Property	Value
Rotation speed	600 rpm
Sampling rate	6400 Hz
Accelerometer range	± 8·g
Equipment state	NS, CBF
Number of sampling per bearing	30
Sampling duration	1 s
Time between sampling	1 s

4.1 Data Collection Workflow

The Data collection process starts within the web application where the machine location and the sensor from which the data is collected are chosen. Then, the data rate of 6400 Hz and the range of 8·g is chosen after which the number of samples, sampling duration and time between sampling is entered. If all parameters were entered, by clicking the button, connection is established with the sensor and the desired data rate and range parameters are transmitted to the edge node. After the connection and transfer of parameters, the transmission of data from the sensor begins. Following the end of completion of sampling, the data is stored in the database and the system is waiting for the next sampling for 1 second after which the cycle starts again and so on another 29 times until all cycles are finished. At the end of the process, the data is visualized, and the user is notified that the collection is completed. The user has a choice to collect data again, review the history of collected data or exit

the application. Detailed system workflow is shown in the activity diagram on Fig. 8.

Additional function of this IIoT system is recording of process state. After each sampling cycle, if the connection was successfully established, the connection state is stored into a database as successful. Else, if there was an error in establishing a connection, the connection status is stored in a database as an error and the user is notified. The same procedure applies to collecting and storing process which enables the system to use stored states for later analysis. The results of the experiment are discussed in the next section.

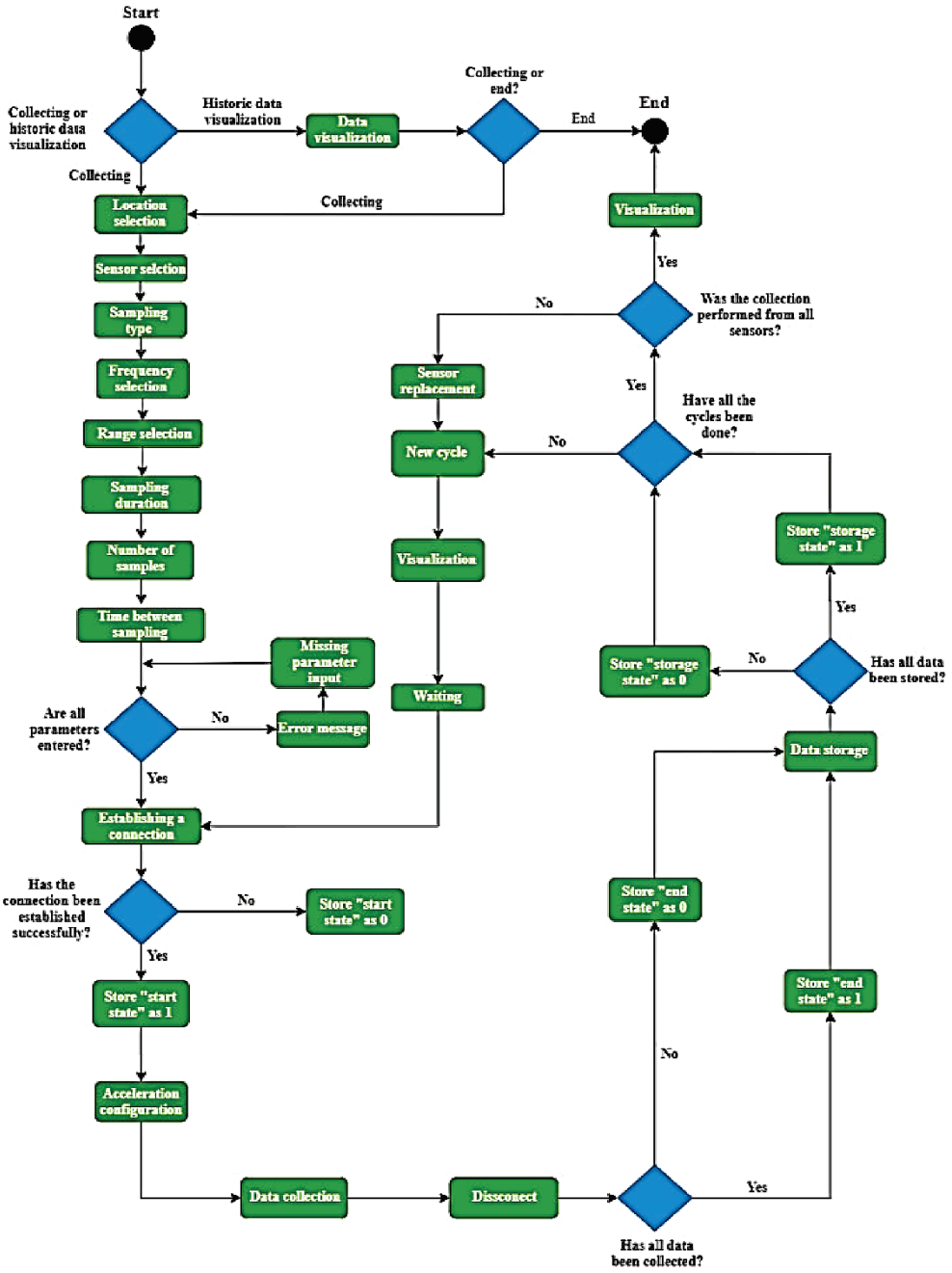

Figure 8 Activity diagram of system workflow

5 RESULTS

During the experiment, 384 000 data points were collected. 192 000 were in the normal state (NS) while another 192 000 were collected in the combined bearing failure (CBF) state. To visualize such a large amount of data, the MATLAB program was used. One collection cycle is set aside from the rest. Fig. 9 shows a graphical representation of one normal collection cycle.

The Fig. 9 shows that in the normal state there are no significant deviations of the acceleration value, as well as patterns that would indicate the presence of a fault. For comparison, Fig. 10 shows a graphical representation of the data collected when simulating a combined bearing failure.

The Fig. 10 clearly shows that there are significant deviations of the acceleration values when simulating the combined bearing fault state. Acceleration values assume patterns that are characteristic of such a simulated state. The same acceleration plot can be seen in web application during the data collection process. Fig. 11 shows the acceleration plot captured in web application.

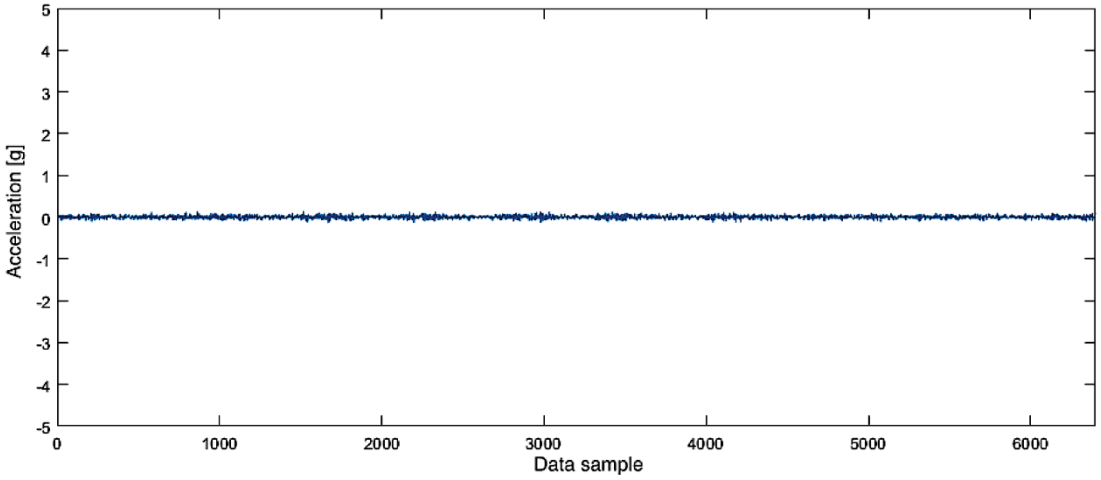

Figure 9 Graphical representation of collected data in normal state

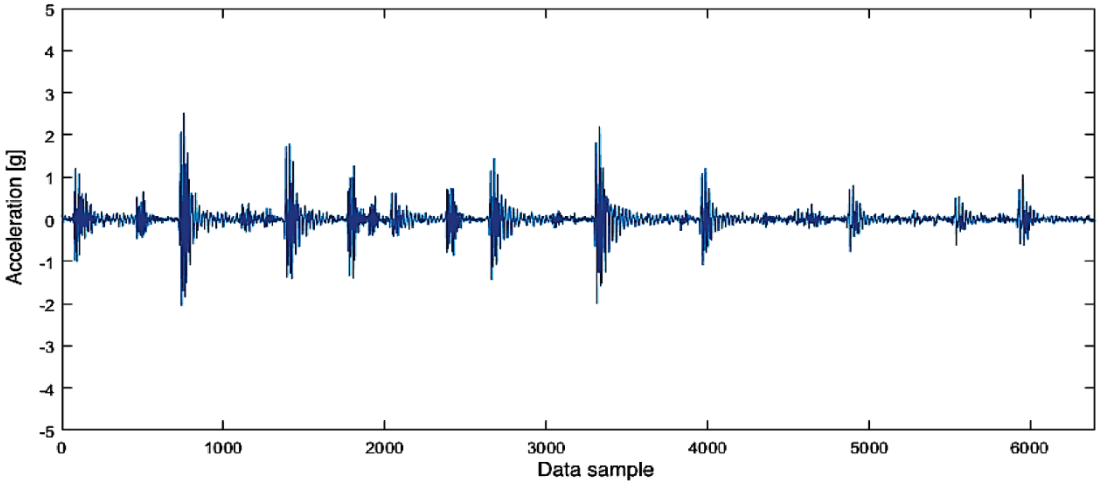

Figure 10 Graphical representation of collected data in combined bearing fault state

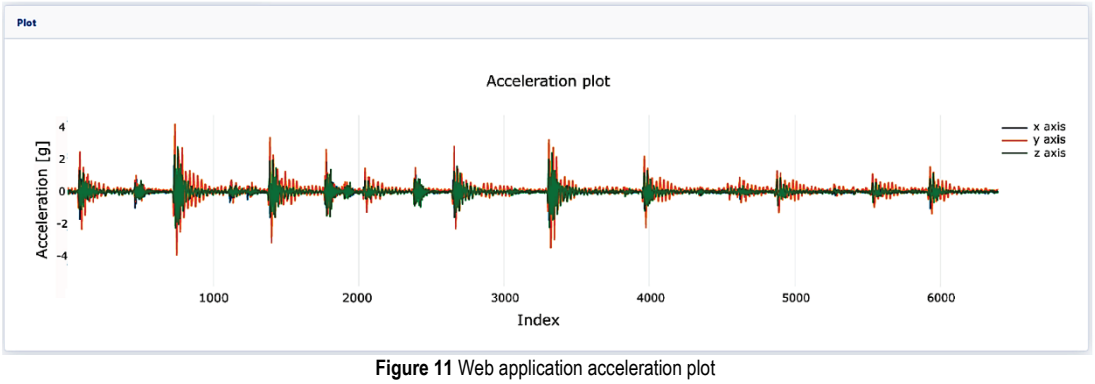

Figure 11 Web application acceleration plot

One of the main limitations of this new system is the impossibility of simultaneous collection from multiple sensors due to the very nature of data transfer which depends on establishing an IP connection between two devices. That means that connection cannot be established simultaneously with multiple devices and for this reason, the system collects data sequentially from each sensor. However, this system limitation is not a decisive factor when used in maintenance. It is expected that faults on certain equipment will be able to be detected by a certain sensor from the moment of

occurrence to the moment of elimination. It is difficult to expect that the system will return to initial state on its own. Therefore, synchronization in this case is not crucial and it is possible to allow a deviation, but this should be kept in mind when displaying and processing data.

This experiment was conducted to show that the IIoT system is successfully gathering information and that there is a clear distinction in acceleration values when simulating two different equipment states. The results indicate that there is a clear distinction in acceleration values which correlate with vibration patterns of normal and combined bearing fault state. In summary, these results indicate that the system is successfully collecting vibration data which can be used for further analysis.

6 CONCLUSION AND FUTURE WORK

As mentioned in the review of previous and related work, many studies use IoT as a framework for automated data collection. One initial objective of this study was to determine if high frequency (>5000 Hz) data sampling could be achieved by the means of IoT technology. The second objective was to enable automated and controlled data collection which allows for an easier data collection process compared to usual methods as previously mentioned.

By conducting an experiment with two equipment states it was shown that the system is successfully collecting high-frequency vibration data that are distinct to each state. This means that this data can be used for further analysis such as machine learning or time series analyses. Also, by enabling automation and control, the foundations for smart maintenance have been set. Future research will aim to improve the current system and explore the possibility of applying a fast Fourier transform to the data and send such data to the server which will speed up the analysis process.

The IIoT data collection system naturally has its limitations, but it is an initial step in the development of a larger system that will have the ability to analyse data from a variety of sources, predict failures, propose maintenance activities and decision support, which is the main feature of current maintenance trends such as prescriptive maintenance whose goal is to reduce downtime costs and increase productivity.

Notice

The paper was presented at MOTSP 2021 – 12th International Conference Management of Technology – Step to Sustainable Production, which took place in Poreč/Porenzo, Istria (Croatia), on September 8–10, 2021. The paper will not be published anywhere else.

7 REFERENCES

[1] Vaidya, S., Ambad, P., & Bhosle, S. (2018). Industry 4.0 – A Glimpse. *Procedia Manuf., 20*, 233-238. https://doi.org/10.1016/j.promfg.2018.02.034

[2] Jasiulewicz-Kaczmarek, M., Legutko, S., & Kluk, P. (2020). Maintenance 4.0 Technologies – New Opportunities for Sustainability Driven Maintenance. *Management and Production Engineering Review, 11*(2), 74-87. https://doi.org/10.24425/MPER.2020.133730

[3] Sisinni, E., Saifullah, A., Han, S., Jennehag, U., & Gidlund, M. (2018). Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Inform., 14*(11), 4724-4734. https://doi.org/10.1109/TII.2018.2852491

[4] Jasiulewicz-Kaczmarek, M. & Gola, A. (2019). Maintenance 4.0 Technologies for Sustainable Manufacturing - an Overview. *IFAC-Pap., 52*(10), 91-96. https://doi.org/10.1016/j.ifacol.2019.10.005

[5] Sinha, J. K. & Elbhbah, K. (2013). A future possibility of vibration based condition monitoring of rotating machines. *Mechanical Systems and Signal Processing (MSSP), 34*(1-2), 231-240. https://doi.org/10.1016/j.ymssp.2012.07.001

[6] Khademi, A., Raji, F., & Sadeghi, M. (2019). IoT Enabled Vibration Monitoring Toward Smart Maintenance. *The 3rd International Conference on Internet of Things and Applications (IoT)*, Isfahan, Iran, 1-6. https://doi.org/10.1109/IICITA.2019.8808837

[7] Cachada, A., et al. (2019). Using Internet of Things Technologies for an Efficient Data Collection in Maintenance 4.0. *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, Taipei, Taiwan, 113-118. https://doi.org/10.1109/ICPHYS.2019.8780217

[8] Ayad, S., Terrissa, L. S., & Zerhouni, N. (2018). An IoT approach for a smart maintenance. *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, Hammamet, 210-214. https://doi.org/10.1109/ASET.2018.8379861

[9] Atmoko, R. A., Riantini, R., & Hasin, M. K. (2017). IoT real time data acquisition using MQTT protocol. *Journal of Physics: Conference Series, 853,* p. 012003. https://doi.org/10.1088/1742-6596/853/1/012003

[10] Yokotani, T. & Sasaki, Y. (2016). Comparison with HTTP and MQTT on required network resources for IoT. *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, Bandung, Indonesia, 1-6. https://doi.org/10.1109/ICCEREC.2016.7814989

[11] Oliveira, G. M. B., et al. (2018). Comparison between MQTT and WebSocket Protocols for IoT Applications Using ESP8266. *2018 Workshop on Metrology for Industry 4.0 and IoT*, Brescia, 236-241. https://doi.org/10.1109/METROI4.2018.8428348

[12] Koene, I., Viitala, R., & Kuosmanen, P. (2019). Internet of Things Based Monitoring of Large Rotor Vibration with a Microelectromechanical Systems Accelerometer. *IEEE Access, vol. 7*, 92210-92219. https://doi.org/10.1109/ACCESS.2019.2927793

[13] Masirap, M., Amaran, M. H., Yussoff, Y. M., Rahman, R. A., & Hashim, H. (2016). Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT). *2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, Penang, Malaysia, 200-205. https://doi.org/10.1109/ISCAIE.2016.7575063

[14] Jaakkola, H. & Thalheim, B. (2011). Architecture-Driven Modelling Methodologies, p. 21.

[15] Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. *Comput. Ind., 101*, 1-12. https://doi.org/10.1016/j.compind.2018.04.015

[16] Tinkerforge Documentation. (2021). https://www.tinkerforge.com/en/doc/ (accessed Mar. 05, 2021)

[17] Accelerometer Bricklet 2.0. (2021). Tinkerforge GmbH, https://www.tinkerforge.com/en/doc/Hardware/Bricklets/Accelerometer_V2.html (accessed Mar. 05, 2021)

[18] Kionix. (2018). KX122-1037: Akcelerometar. Kionix.

[19] Jacobson, D., Brail, G., & Woods, D. (2012). *APIs: A Strategy Guide*. Sebastopol: O'Reilly.
[20] Kolar, D. (2019). Deep learning-based early fault diagnosis model for rotary machinery. *Doctoral thesis*, University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture, Zagreb.

Authors' contacts:

Martin Curman, mag. ing. mech.
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10002 Zagreb, Croatia
martin.curman@fsb.hr

Davor Kolar, PhD
(Corresponding author)
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10002 Zagreb, Croatia
davor.kolar@fsb.hr

Dragutin Lisjak, PhD, Assoc. Prof.
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10002 Zagreb, Croatia
dragutin.lisjak@fsb.hr

Tihomir Opetuk, PhD, Assist. Prof.
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10002 Zagreb, Croatia
tihomir.opetuk@fsb.hr