ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20220517173151

Preliminary communication
Received: 2022-05-17, Accepted: 2022-05-19

Condition Monitoring of Rotary Machinery Using Industrial IOT Framework: Step to Smart Maintenance

Davor Kolar*, Dragutin Lisjak, Martin Curman, Michał Pająk

Abstract: Modern maintenance strategies, such as predictive and prescriptive maintenance, which derived from the concept of Industry and Maintenance 4.0, involve the application of the Industrial Internet of Things (IIoT) to connect maintenance objects enabling data collection and analysis that can help make better decisions on maintenance activities. Data collection is the initial step and the foundation of any modern Predictive or Prescriptive maintenance strategy because it collects data that can then be analysed to provide useful information about the state of maintenance objects. Condition monitoring of rotary equipment is one of the most popular maintenance methods because it can distinguish machine state between multiple fault types. The topic of this paper is the presentation of an automated system for data collection, processing and interpretation of rotary equipment state that is based on IIoT framework consisting of an IIoT accelerometer, edge and fog devices, web API and database. Additionally, ISO 10816-1 guidance has been followed to develop module for evaluation of vibration severity. The collected data is also visualized in a dashboard in a near-real time and shown to maintenance engineering, which is crucial for pattern monitoring. The developed system was launched in laboratory conditions using rotating equipment failure simulator to test the logic of data collection and processing. A proposed system has shown that it is capable of automated periodic data collection and processing from remote places which is achieved using Node RED programming environment and MQTT communication protocol that enables reliable, lightweight, and secure data transmission.

Keywords: accelerometer; automated data collection; Industrial Internet of Things (IIoT); MQTT; Node RED

1 INTRODUCTION

In today's era of modern digital based technology there is a huge amount of data being generated every second. Companies are trying to adopt the new trends by using all sorts of data to gain competitive and financial advantage by optimising production lines, increasing productivity and efficiency of processes and utilizing modern information technologies. In many cases, companies try to increase production and machine usage to bring their product to market with maximum output and capacity but fail to do so because of inadequate maintenance strategies [1]. Using modern data based maintenance strategies, such as Predictive maintenance, has shown that there are more benefits in terms of operation and maintenance costs. To implement such strategy, several modern Industry 4.0 technologies are required which can appear as high cost to companies which want to implement them [2]. One of the required technology is an Industrial Internet of Things (IIoT) framework which provides environment for data acquisition and transmission from data source to the desired destination. IIoT can integrate various manufacturing devices to sense, identify, process, communicate, operate and network [3]. IIoT sensors are small, flexible, versatile and cost efficient solution for modern data acquisition requirements. The use of sensors in Predictive maintenance is associated with monitoring of a rotary machinery, which includes motors, pumps, fans, turbines, gearboxes and similar equipment. Vibration analysis is one of the main techniques used to monitor rotary machinery equipment state because vibration forms unique patterns that are associated with specific rotary equipment faults [3]. To acquire such data, the most common sensor for such measurement are accelerometers. A key advantage of accelerometers are their versatility and in the embedded domain, microelectromechanical systems (MEMS) accelerometers are widely used due to their small size, low cost and low power consumption [4]. Using an IIoT MEMS accelerometer has even more advantages because of its ability to be connected to Internet and transmit data over it. Because of this characteristic, data can be acquired from variety of locations where some sort of Internet connectivity is available.

The main purpose of this paper is to showcase the development of an automated periodic data collection IIoT system that can collect, process, visualize and transmit data from an IIoT sensors in a place where conventional network is not available. The system can also automatically evaluate machine state using ISO 10816-1 reference. The rest of the paper is organized as follows: Section 2 overviews the related work in the field of IIoT vibration condition monitoring and data acquisition, Section 3 contains system architecture and components description after which data collection and transformation into knowledge process is described in Section 4. Results of system evaluation are discussed in Section 5 and finally, conclusion is drawn in Section 6.

2 RELATED WORK

As the field of application of IIOT in the field of maintenance is emerging, there are few papers that are linked to this research which cover similar topics. In this section, the papers are briefly described. Paper [5] gives an overview of current state of predictive maintenance and intelligent sensors in smart factories. Authors concluded that the importance of predictive maintenance is growing over time in relation to Industry 4.0 technologies which gives an advantage to intelligent sensor that can connect to the Internet. As already mentioned, IIoT MEMS accelerometer is the main solution for intelligent data acquisition which is demonstrated in paper [2]. This paper demonstrated the use of MEMS accelerometer for condition monitoring of induction motors and an expander machine in two different industrial settings. Each sensor was connected to the data acquisitioning device (NI CDAQ9191) and every 3 seconds vibration signal was stored with a use of a generic web application on a web server. The research concluded that the faults on two different machines can be detected using a low-cost sensor. Authors in paper [6] developed a versatile wireless sensor unit called "Mesimo" which is operated from web browser making it remotely accessible from a broad range of devices. Such device was used to measure translatory movement of linear actuator and the vibrations of large rotary machinery. Similarly in [7] the authors used Zigbee enabled IIoT MEMS accelerometer which uses low power so it can be used in a situation where there is no power source available. The IIoT sensor was used to collect data from robot joints to evaluate the system performance. In [8] the authors developed and implemented an IIoT system designed to monitor electric motors in real time. This setup also used low-cost MEMS accelerometers, temperature sensor and open source IoT software. The setup used 1 kHz sampling frequency with data transmitted over wireless access point as an input to scripts written in Phyton. The paper also touched on the topic of edge computing where in one case the data was processed right after sampling before being sent to the server. In [9] condition monitoring on four CNC machines was conducted using a remote vibration monitoring system which used MEMS accelerometer. The Raspberry Pi was used to send data to the database in real time after which the Phyton script analysed the data in terms of acceleration in the time domain and applied FFT to obtain the acceleration in the frequency domain. It is important to point out that in the papers above, software components weren't highlighted enough. The authors in [10] used Node RED software designed to handle IoT to implement MQTT based air quality monitoring system. In this paper we followed similar strategy to handle and visualize data. Similar approach was used in [11] to capture temperature and humidity data using a Raspberry Pi and Node RED, however in this paper the visualization has not been showcased. Another great example of an IIoT system that uses 4G is in [12] where authors used low-cost MEMS accelerometer with Raspberry Pi and cloud technologies to monitor vibration impact on the surrounding environment during construction activities to gain better insight into structure safety, human comfort and equipment functionality. This paper has also many connecting points with our previous research [13]. The papers above highlighted the importance of remote data collection ability which is where the industry is oriented in this present moment. This paper focuses on the remote data collection, processing and interpretation of data using modern IIoT technologies and protocols such as Node RED and MQTT to achieve described functionality.

3 SYSTEM REQUIREMENTS AND COMPONENTS

Based on the conducted research and internal requirements, the system requirements can be defined as the following:

1) Automated periodic data acquisition – the system must be able to periodically collect data from sensors, process it and transmit it to the web API on the server.
2) Ability to acquire and process data from remote locations – the system must be able to collect data from remote locations where conventional WiFi and Ethernet networks are not available. The data must also be processed and aggregated accordingly.
3) Data acquisition from different types of sensors – the system must be able to collect data from multiple different types of sensors such as an accelerometer, temperature, air pressure sensor and other.
4) Data processing and visualisation - the system must be able to process the data and interpret it based on an ISO 10816-1 reference. Also, data must be visualized in a convenient way i.e. in a dashboard.
5) Secure data transmission – data must be securely transmitted over the Internet to assure data integrity and security.

The system must be designed in such way to satisfy the requirements above. This can be achieved by using modern IoT software and hardware components the system architecture is shown in Fig. 1.

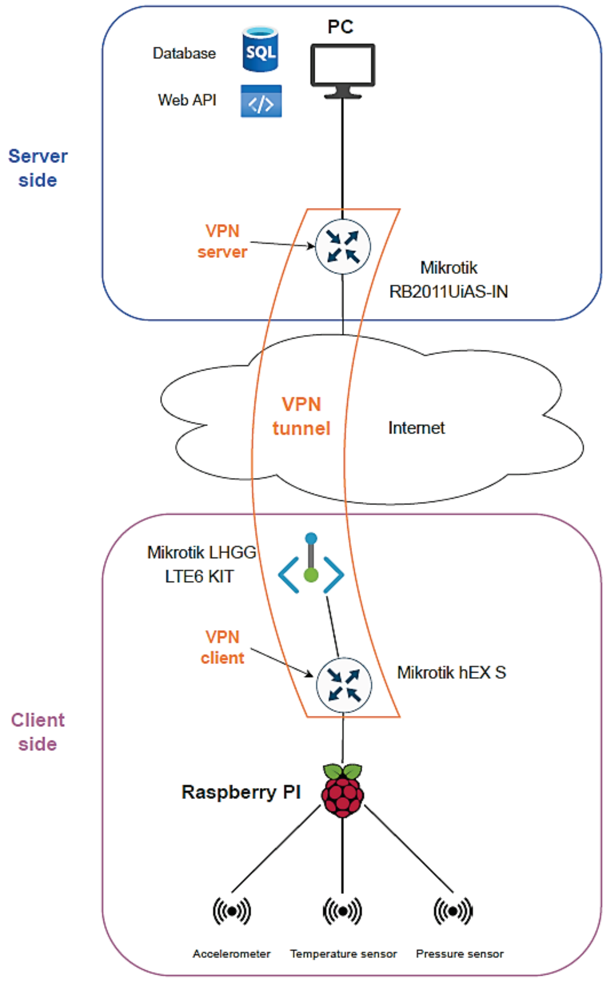


Figure 1 System design

The architecture is divided into two parts: the server side and the client side. The server side consists of the following components:

(1) MikroTik RB2011UiAS-IN router - configured with static IP address and DHCP server, which enables it to have more than one client connected. This router also runs a VPN server.

(2) Application/database server - hosts an IIS web server that runs the Web API. There is also an MSSQL server with a database on this server. The client side consists of the following components:

1) MikroTik LHGG LTE6 KIT - LTE router with antenna designed for Internet access in remote locations with low network coverage.
2) MikroTik hEX-S router - designed for places where wireless connection is not required, contains five Gigabit Ethernet ports as well as PoE output on the rear port.
3) Raspberry PI 4 - a small computer suitable for various types of tasks. A 4 GB RAM model was used.
4) HAT module - HAT extension that plugs into Raspberry's GPIO port. Sensors are connected to the HAT module.
5) IOT sensors - for acceleration, temperature, and air pressure:

- Accelerometer - KX122 MEMS sensor
- Temperature sensor - STS3x sensor
- Air pressure sensor - LPS22HB sensor.

Communication takes place via a site-to-site VPN network. This method is also known as VPN between routers. In this method, a router that supports a VPN client always establishes a VPN tunnel with a VPN server that allows private networks located behind the router to communicate with each other. OpenVPN was chosen for the VPN system due to its availability on MikroTik routers. Using a virtual private network, it is possible to connect remote locations in a secure and reliable way. What is important to emphasize is that one always strives to achieve communication over a virtual private network. It is not recommended to expose the server and communication directly to the Internet because then the level of security is reduced, and it is necessary to provide additional security settings. The client side uses a 4G router and antenna which provides connectivity in a remote locations or other places where WiFi or Ethernet is not available. On the client side, the Raspberry Pi serves as the edge node where the sensors connect to, and which also processes the collected data. Raspberry Pi runs Node RED programming environment that is designed for IoT environments. Node RED is an open-source rapid embedded environment design for easier integration of IoT devices and related software. Node RED provides visual browser based editor that is easy to understand and use [11]. Since the system is based on IOT technology, for communication between the sensor and the Raspberry Pi the MQTT protocol will be used. MQTT is a fast and lightweight protocol that allows messaging between devices located on unstable networks and ensures secure, reliable and two-way messaging [14]. The protocol uses the principle of publication/subscription on the topic. Fig. 2 shows an example of such system.

In this case, the MQTT broker and client will be on the same device, the Raspberry Pi. If there were more than one edge nodes, only clients connecting to the broker would be installed on the others. The IIoT sensors are connected to the Raspberry Pi using the HAT module. The HAT module is a Raspberry Pi HAT with a standard Raspberry Pi HAT shape factor. The module follows the HAT specification and will work with Raspbian automatically and without any changes. Up to 8 sensors can be connected to the HAT module. HAT module is replacing stack used in [13] because of its simplicity and Raspberry Pi compatibility. The appearance of the Raspberry Pi with HAT can be seen in Fig. 3.

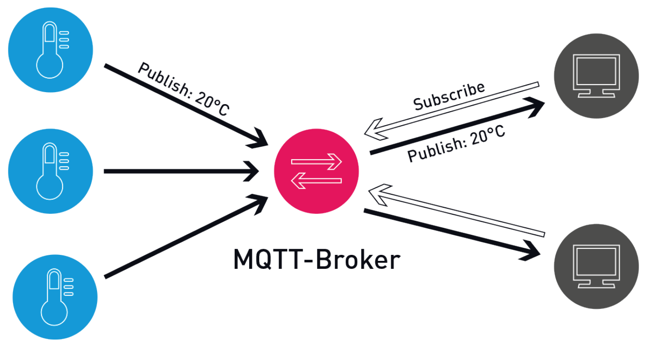


Figure 2 MQTT system [15]

Figure 3 Raspberry Pi with HAT module

For this research, our setup used three different types of sensors: an accelerometer, temperature sensor and pressure sensor. All sensors were produced by same manufacturer which simplifies integration process. MEMS accelerometer can sample data from three axis (x, y, z) with data rate up to 25,6 kHz [16]. Accelerometer is shown in Fig. 4.

Figure 4 KX122 accelerometer sensor

Accelerometer specifications are listed in Tab. 1.

This system was configured to sample data at 3200 Hz and ±8g range with 16bit resolution. For acquiring temperature information, a temperature sensor was used. Temperature sensor can measure ambient temperature with

0.2 °C accuracy and a temperature range from –40 °C to 125 °C with an output of 0.01 °C step. Temperature sensor specifications are listed in Tab. 2.

Table 1 Accelerometer specification [16]

Property	Value
Output data rate	0.781 Hz - 25.6 kHz
Full-scale range	± 8g
Sensitivity	4096 - 16384 counts/g
Offset	± 20 mg
Non-Linearity	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

Table 2 Temperature sensor specification [16]

Property	Value
Sensor	STS3x
Current consumption	28 mW (5.6 mA at 5V)
Ambient Temperature	–40 °C to 125 °C in 0.01 °C steps
Accuracy	typical 0.2 °C in the range of 0 °C to 65 °C*
Dimensions ($W \times D \times H$)	25 × 15 × 5 mm
Weight	2 g

Air Pressure sensor with integrated air pressure, temperature and altitude measurement was also used. The sensor has the capability to measure air pressure in range of 260 to 1260 hPa with a resolution of 0.0075 hPa. Detailed specifications are listed in Tab. 3.

Table 3 Air Pressure sensor specification [16]

Property	Value
Sensor	LPS22HB
Current consumption	30 mW (6 mA at 5V)
Pressure Range	260 to 1260 hPa
Resolution	0.0075 hPa / 6.25 cm
Accuracy (0-65 °C)	±1.1 hPa uncalibrated, ±0.2 hPa calibrated*
Dimensions ($W \times D \times H$)	25 × 15 × 5 mm
Weight	1.6g

Temperature and air pressure sensor are shown in Fig. 5.

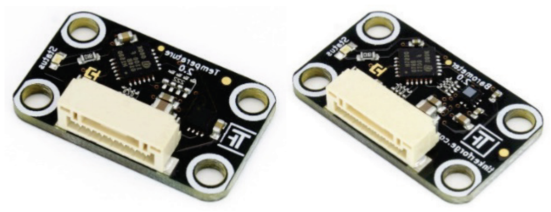


Figure 5 Temperature and air pressure sensor

4 DATA COLLECTION AND PROCESSING

The data collection work process consists of multiple flows. A flow can be viewed as a standalone program (script) which contains functions and methods to achieve specific tasks. Node RED supports the creation of multiple flows that can interchange data and have its own sub flows. General data collection workflow can be seen in Figure 6. For the purpose of this paper, the system ability was evaluated using a rotational equipment fault simulator referenced in [13] using a accelerometer, temperature and air pressure sensor.

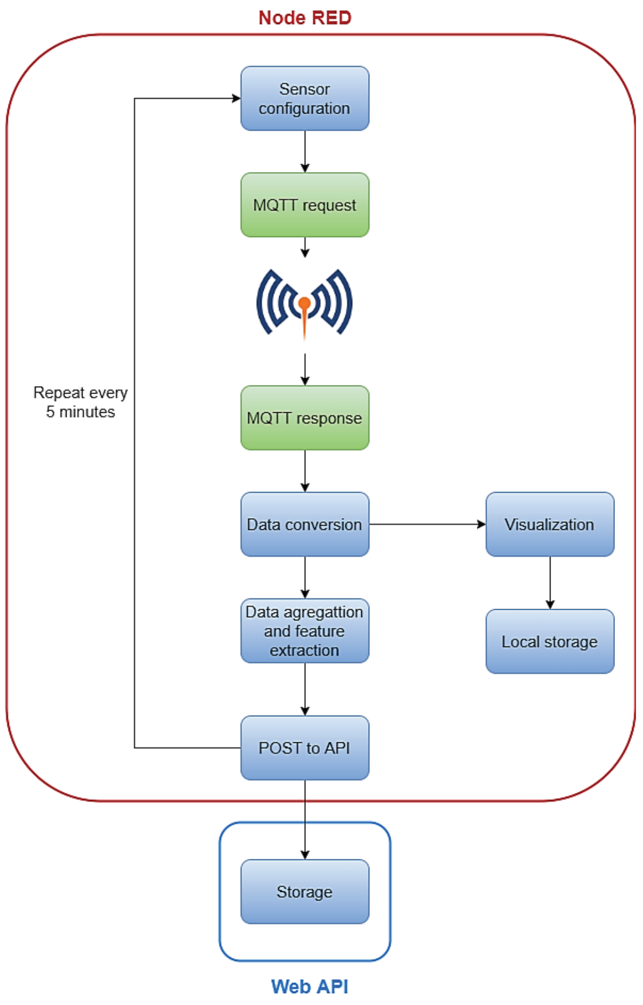


Figure 6 Data collection workflow

Data collection process starts with setting the sensor configuration. The configuration setting is sent over the MQTT broker to the MQTT client which governs the sensors. Data from sensors is then sent back over to the main process where data is converted, aggregated and in a case of acceleration, features are extracted and interpreted. The data is then simultaneously visualized in a dashboard, stored locally and sent to the server where it is stored in a database. The whole process repeats every 5 minutes. Detailed explanation is provided below.

The flow begins with a request that is sent over the MQTT broker to the MQTT client that is responsible for data acquisition process. Since the accelerometer supports a wide frequency range of sampling (from 0.718 Hz to 10 kHz) and a range of values (from ± 2g to ± 8g) it is necessary to define these parameters before the process starts. In this experimental setup, the sampling rate is 3200 Hz and the range is ± 8g. Also, in addition to the frequency and range, it is necessary to define the collection resolution as well as the number of axes from which the data is collected. There is an option to select 8-bit or 16-bit resolution and any number of axes. A higher resolution is taken, i.e., 16-bit and all three axes are enabled. This process of configuring and enabling data collection from the accelerometer is performed only once, 30 seconds after startup. The temperature and air pressure configuration are set by the default so that only the values must be called.

After sending the configuration and the request to the broker comes the phase of receiving values from the broker. The acceleration values arrive in a series of 30 data points which allows the collection of values at higher frequencies (> 800 Hz). The standard "Get" invocation of values works by sending a request to the broker every *n* millisecond. Due to the very nature of the MQTT protocol, it is not possible to have a reliable flow of information at low values of the collection interval (< 3 ms). Therefore, the accelerometer allows continuous invocation of values which accelerometer then sends in series, allowing a higher frequency of collection. If the object received from the broker is empty, an error is logged and sent to the API. The data coming from the sensor takes the shape of an array with 30 values. The acceleration data inside an array is grouped by three (x, y, z, x_1, y_1, z_1…) and it needs to be separated so that further manipulation can take place. After dividing the data into groups of three, it is necessary to convert the data into the correct form because the data arrives in "raw" form, and it needs to be converted into g values. This is done by multiplying the acceleration values by a range-dependent constant. The equation for calculating the acceleration at 8g range is:

$$a_x = value * \frac{2500}{1024} \tag{1}$$

$$a_y = value * \frac{2500}{1024} \tag{2}$$

$$a_z = value * \frac{2500}{1024} - 1 \tag{3}$$

The value of the *Z* axis must be subtracted with 1 to compensate the influence of gravity. After conversion of values, an object is created that contains only *x* values, only *y* values and only *z* values. The data is now properly grouped but needs to be separated into three separate streams: one for *x* values, another for y and a third for *z.* After separation, each individual stream has only its own specific values for each axis. Since the collection frequency is 3200 Hz, and the values arrive in a series of 10 values, it is necessary to accumulate the values until the specific number of data values is reached. The system is configured to "wait" for the arrival of 320 messages (ten data points 320 times) so that all acceleration values are grouped together. When 3200 values arrive, the flow continues to the point where RMS (root mean square value) for each axis is calculated. To calculate RMS values, the data needs to be converted to velocity; the following equation is used [17]:

$$v = \frac{9{,}81a}{2\pi f} \tag{4}$$

where: a – acceleration value, f – frequency.

Finally, the equation for *RMS* is:

$$RMS = \sqrt{\frac{1}{n}\sum\nolimits_i x_i^2} \tag{5}$$

where: n – number of values (3200), x – acceleration values.

After calculating the *RMS* values for each axis, the values are consolidated into a single object. The final object contains the following parameters:

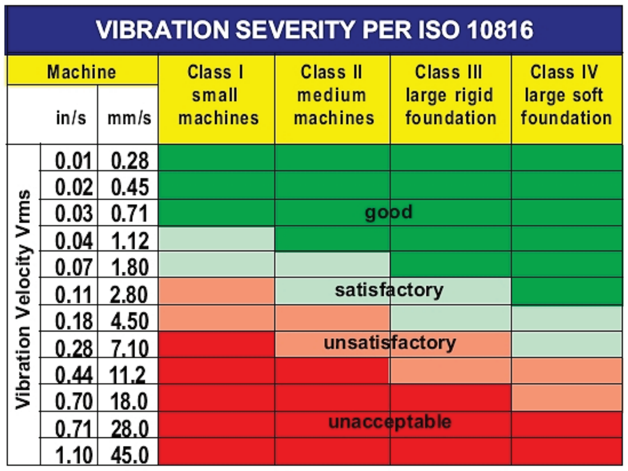
- RMS *x* - RMS value of the *x* axis.
- RMS *y* - RMS values of the *y* axis.
- RMS *z* - RMS values of the *z* axis.
- UID - unique sensor identifier.
- Location - the location of the sensor.
- Timestamp - timestamp.

The object thus created is further sent to the Web API and to the dashboard. Unlike acceleration workflow which has data transformation and feature extraction, temperature and air pressure flow is much simpler. After receiving data from the broker, data is converted to JavaScript object for simpler manipulation. The temperature values need to be divided by 1000 to get values in °C after which a temperature and air pressure variables are created. The data is then visualized in a dashboard and sent to the Web API. Once the data from the sensor is collected, it needs to be stored somewhere. A database is a digital way of storing data based on a relational model. Relational databases use a database management system that allows you to define, create, maintain, and control a database. The Microsoft SQL Server database management system will be used for data storage and management on the server side. To enable different clients to store and read data from the database, a Web API was created. The Web API is a server-side programming interface that operates based on a request-response system, usually expressed in JSON or XML format, and is available via the Internet, most commonly via HTTP web servers. Currently, there are various programming environments that can be used to create a Web API. The ASP.NET Core desktop environment was chosen for its simplicity, speed, and reliability. Web API also have the functionality of error handling which is useful in a case when an accelerometer, temperature sensor or air pressure sensor stop sending values. Alongside traditional relational database, which is located on the server, a separate local database was created. Local database was created with Influx DB, an open-source time series database (TSDB) optimised for storage of time series data. Local database is used as a backup in a case of network problems, so that the values are not lost because of connectivity problems.

Finally, the observed object state is assessed based on values from the sensors. By calculating the *RMS* (5) and referencing the values by ISO 10816-1, machine "health" and state can automatically be assessed, and the user can be notified. This is one of the most important features of this IIoT system because it provides users with direct and upfront information about system state without the need for further data analysis. To achieve described functionality, an ability to visualize and interpret data is necessary. Data visualisation is achieved using a Node RED dashboard module. Dashboard module provides a set of nodes to quickly create a live data visualization based on incoming data. Dashboard contains nodes such as gauge, line chart, form input, data picker, and other common items. Dashboard is configured to contain several items:

1) ISO 10816-1 reference picture - a reference picture containing an ISO 10816-1 evaluation of machine vibration using measurements made on rotating parts.
2) Air pressure and temperature gauge - gauge displaying air pressure and temperature values.
3) Vibration parameters gauge - gauge displaying RMS values of vibration.
4) Vibration state information.
5) History of measurements - time series plot of history trends for vibration, temperature, and air pressure.

Interpretation of the data is possible due to developed logic that follows ISO 10816-1 guidance. Fig. 7 shows ISO 10816-1 machine state classification categories.



VIBRATION SEVERITY PER ISO 10816

	Machine		Class I small machines	Class II medium machines	Class III large rigid foundation	Class IV large soft foundation
	in/s	mm/s				
Vibration Velocity Vrms	0.01	0.28	🟩	🟩	🟩	🟩
	0.02	0.45	🟩	🟩	🟩	🟩
	0.03	0.71	🟩	🟩 good	🟩	🟩
	0.04	1.12	▨	🟩	🟩	🟩
	0.07	1.80	▨	▨	🟩	🟩
	0.11	2.80	🟧	▨ satisfactory	▨	🟩
	0.18	4.50	🟧	🟧	▨	▨
	0.28	7.10	🟥	🟧 unsatisfactory	🟧	▨
	0.44	11.2	🟥	🟥	🟧	🟧
	0.70	18.0	🟥	🟥	🟥	🟧
	0.71	28.0	🟥	🟥 unacceptable	🟥	🟥
	1.10	45.0	🟥	🟥	🟥	🟥

Figure 7 ISO 10816 classification [18]

Fig. 8 shows acceleration data transformation, processing and interpretation flow, which was described above.

Temperature and air pressure flow, as already mentioned, have much simpler flow, which does not include automated decision making. The idea behind including temperature and air pressure sensor is to use this information to observe abnormal behaviours. It can be assumed that temperature and air pressure values correlate i.e., when temperature increases the air pressure also increases. If the air pressure is constant, and the temperature of the system is increasing, we can assume that there is something going on in the system.

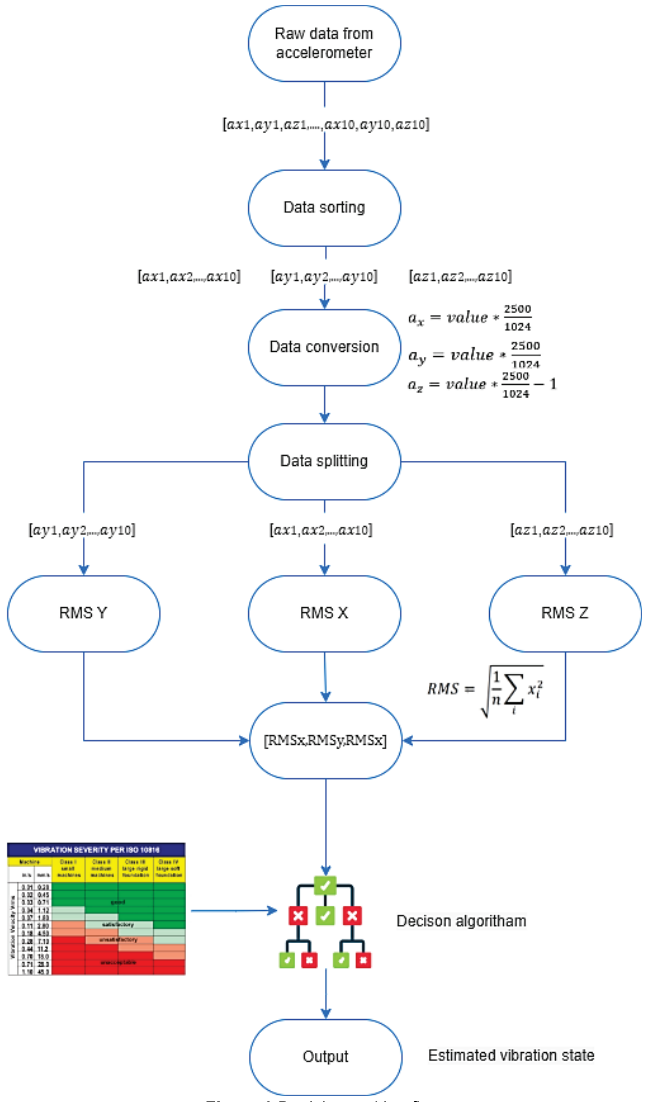

Raw data from accelerometer
[ax1,ay1,az1,...,ax10,ay10,az10]
Data sorting
[ax1,ax2,...,ax10]
[ay1,ay2,...,ay10]
[az1,az2,...,az10]
Data conversion
$a_x = value * \frac{2500}{1024}$
$a_y = value * \frac{2500}{1024}$
$a_z = value * \frac{2500}{1024} - 1$
Data splitting
[ay1,ay2,...,ay10]
[ax1,ax2,...,ax10]
[az1,az2,...,az10]
RMS Y
RMS X
RMS Z
$RMS = \sqrt{\frac{1}{n}\sum_i x_i^2}$
[RMSx,RMSy,RMSx]
VIBRATION SEVERITY PER ISO 10816
Decison algoritham
Output
Estimated vibration state

Figure 8 Decision making flow

5 SYSTEM EVALUATION AND RESULTS

To test system performance, a series of experiments have been concluded. The system performance was tested using a rotating equipment failure simulator in a laboratory condition with following states:

1) State 0 - machine down.
2) State 1 - machine up and running at 1500 rpm
3) State 2 - machine up and running at 1500 rpm with fault (eccentric rotor fault).
4) State 3 - machine running at 1500 rpm with enhanced fault (eccentric and cocked rotor fault).

Below are figures taken from dashboard for each state, followed by description. General dashboard layout can be seen on Fig. 9. This screenshot of a dashboard was taken in State 0 where the machine was down and not operating. It can be observed that vibration gauges are displaying low values, which correspond to natural environment vibration.

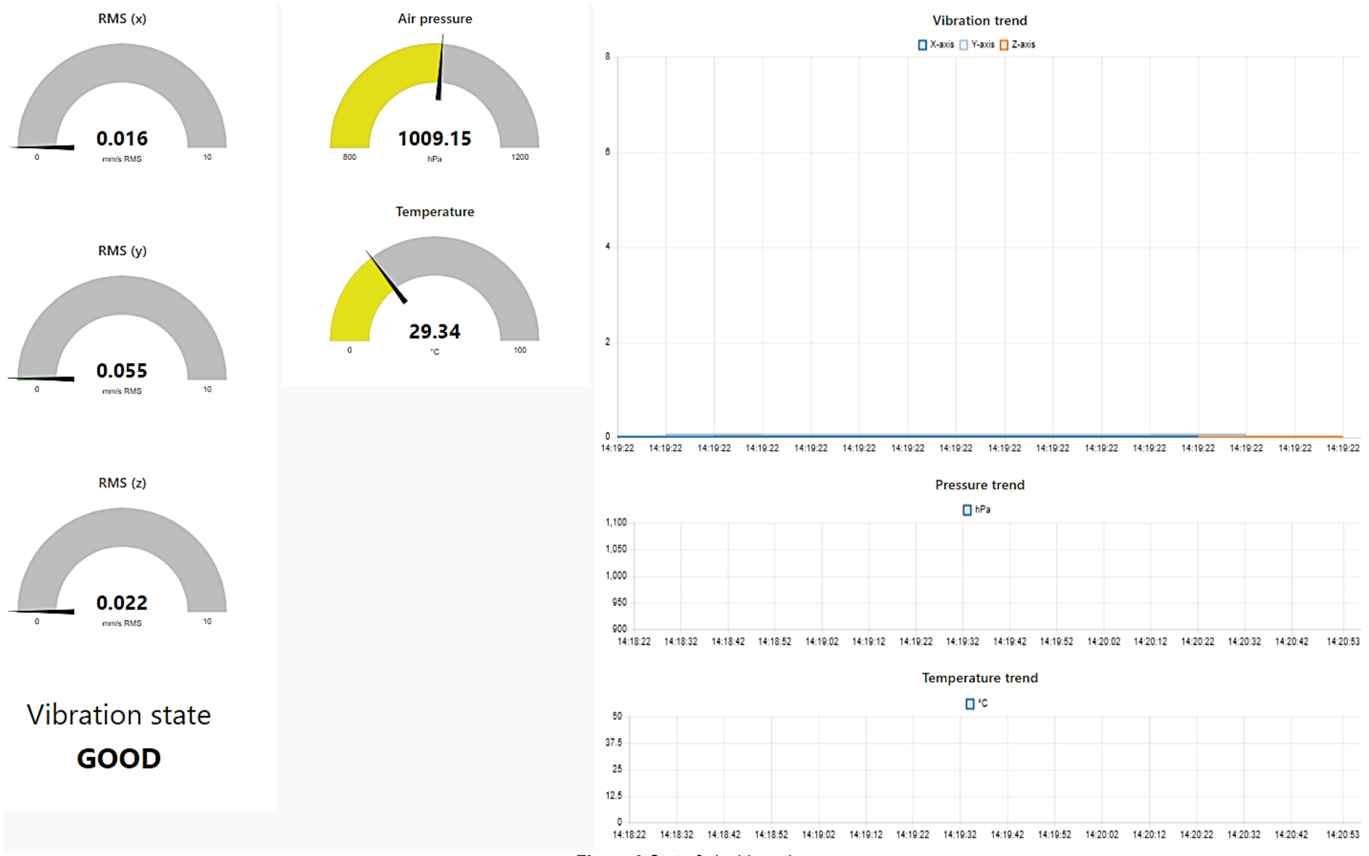


Figure 9 State 0 dashboard

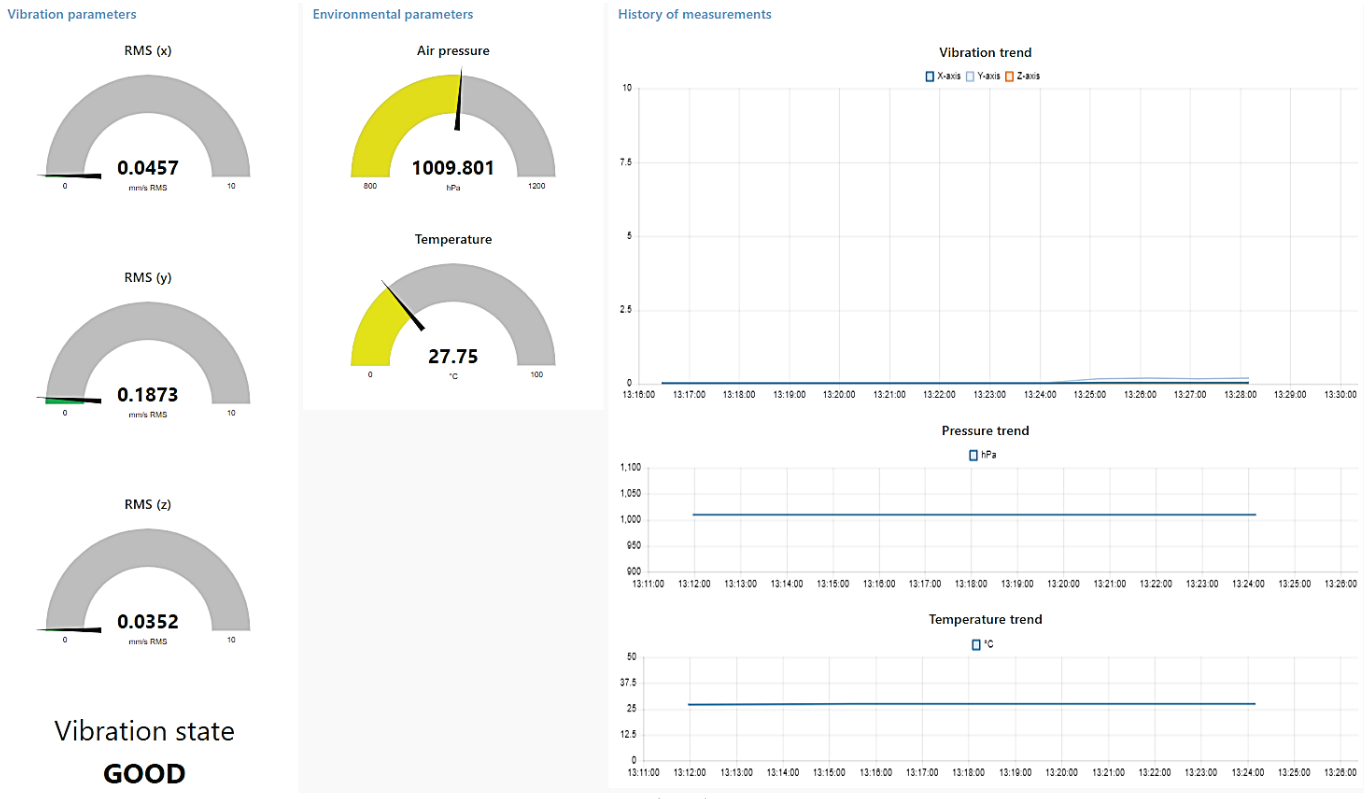


Figure 10 State 1 dashboard

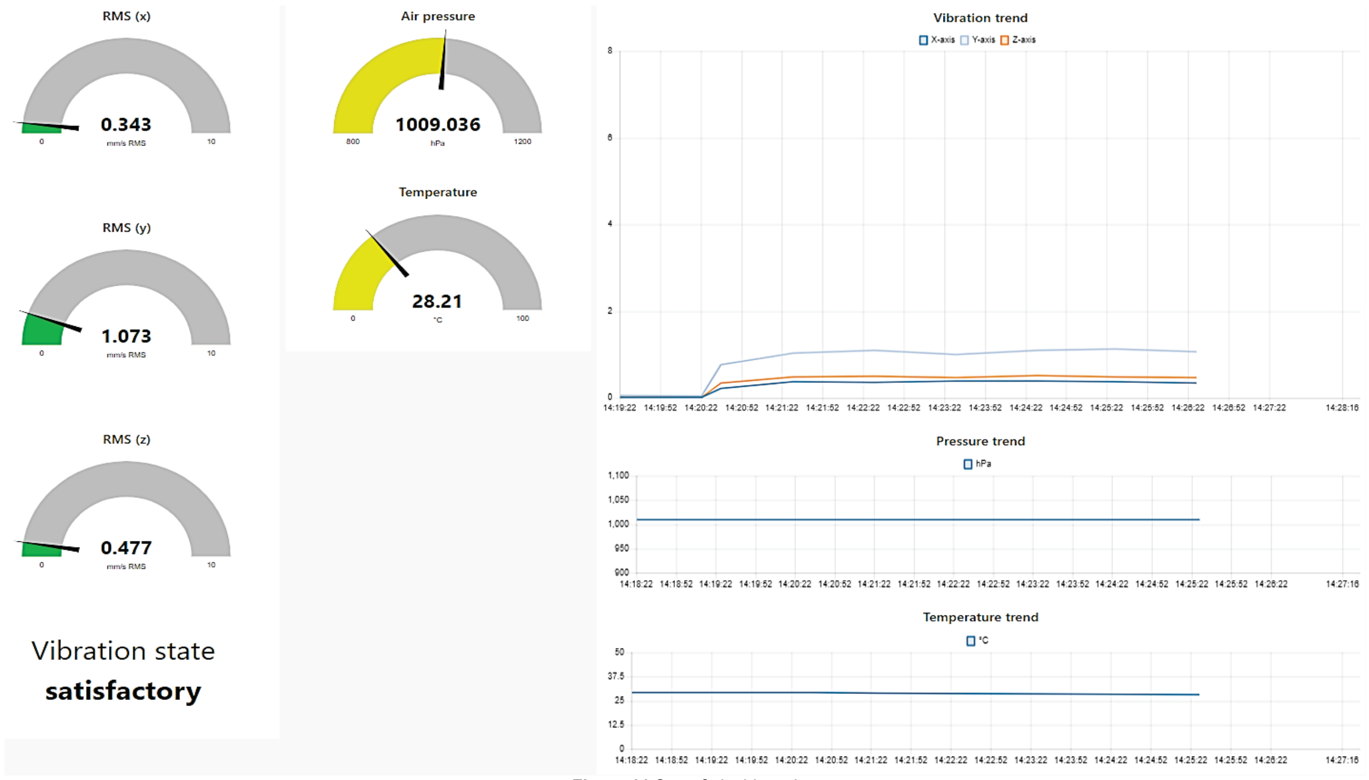


Figure 11 State 2 dashboard

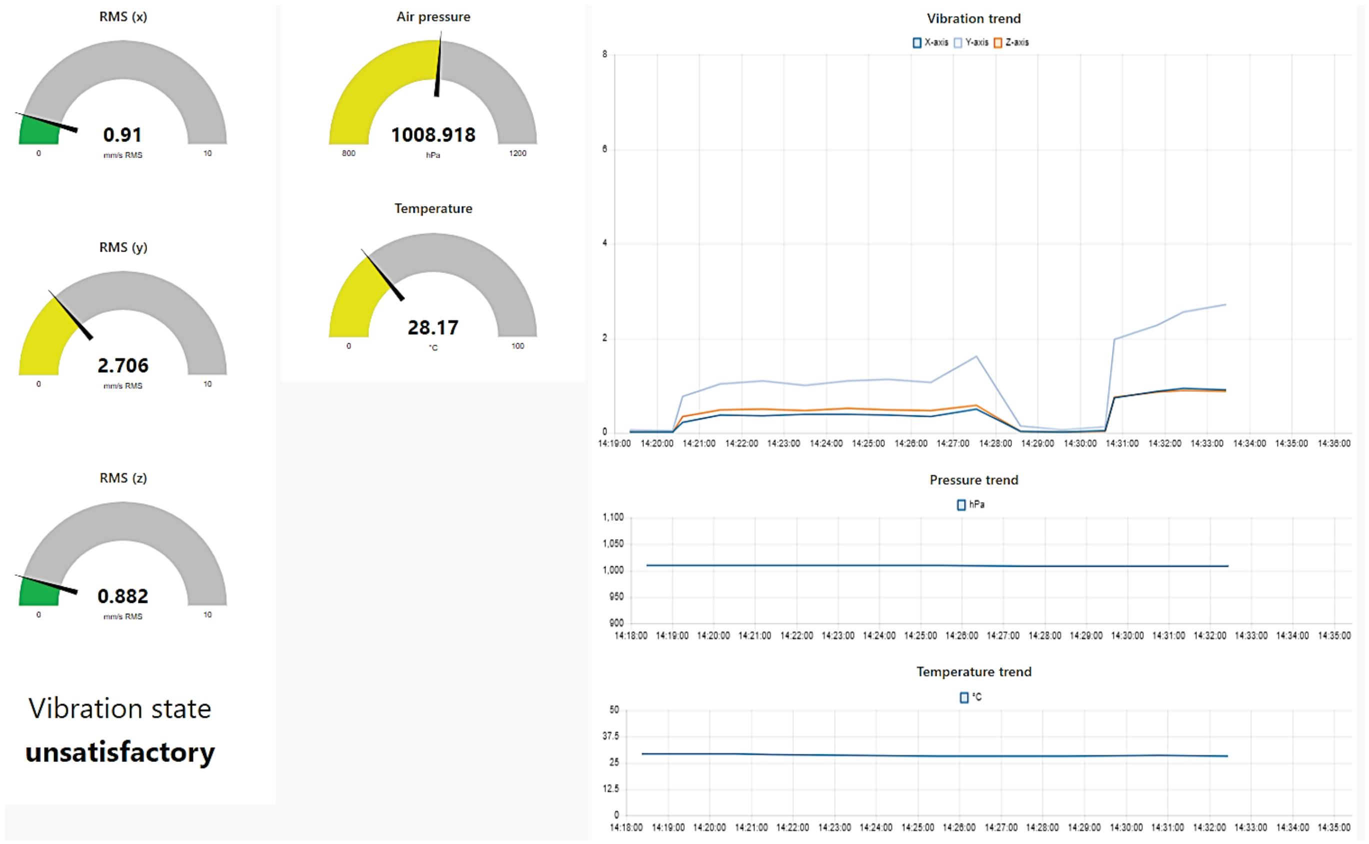


Figure 12 State 3 dashboard

Temperature and air pressure values are corresponding to the natural ambient in which the machine is located. Because the machine is not in operation, the system classifies the condition as good. In addition, on the vibration plots, nothing significant is showing up. Switching the machine on and configuring the rotational speed of 1500 rpm, the system now acts differently. Fig. 10 shows system in its natural running state - State 1.

The slight change on vibration gauges can be observed. The machine state remains good. To observe system ability to classify multiple states, major eccentric rotor fault was simulated in State 2. Fig. 11 shows the system parameters in this state.

The system classifies this state as satisfactory, which corresponds to ISO 10816-1 reference. Vibration plot is also showing significant jump from initial values. Finally, the State 3 was simulated with two simultaneous failures (major eccentric rotor fault + cocked rotor). Fig. 12 show the system dashboard in that state.

It can be observed that *RMS x*, *y* and *z* values have significantly jumped (vibration plot also increases in value) and the system classifies vibration state as unsatisfactory.

The simulated machine states gave us an important insight into system ability to successfully classify system vibration state based on ISO 10816-1 reference. As it was shown, the system successfully classified between "Good", "Satisfactory" and "Unsatisfactory" vibration states based on near real time vibration data. The system is therefore ready to be tested outside laboratory environments.

5 CONCLUSION AND FUTURE WORK

The result of this paper provides a possibility to use such IIoT system for remote data collection and automated near real time condition monitoring in a real production environment. Using low cost IIoT sensors, such as MEMS accelerometer, temperature sensor and air pressures sensor, enables a low-cost functional ability for the system to gather data about environmental parameters. Data processing and transformation was possible using developed Node RED flows, which contained functions for data transformation, feature extraction and decision-making logic. Specifically, acceleration data was used to calculate *RMS* values for each axis, which was then, compared to ISO 10816-1 reference that provides users with system state classification. The Node RED range of nodes can be expanded with additional functionalities making this program truly comprehensive in the IIOT area. By storing data locally, the system is more redundant in a case of network connectivity problems. On the server side, the Web API enabled easy, fast and reliable entry of incoming data which is then stored into a relational database.

Based on all the above, it can be concluded that the system successfully collects and interprets data in a laboratory environment. The next step in the further development of the experimental system is to test the system at a remote location to gain insight into the reliability of the system. To achieve more precise control, monitoring, and configuration of the data collection process, it is necessary to develop a web application that will serve as a control panel for the entire process which was described and configured in our previous research. Web application integration also implies necessary changes within the program logic of Node-RED and Web API, but this is not a problem since both components of the system are extremely flexible. Also, it is important to mention that the collected data can be used for multiple predictive machine learning algorithms which is a right step towards smart maintenance. The goal of the future works will be to configure the edge node to use machine learning algorithms such as Convolutional Neural Networks (CNN) directly on the edge which will enable automated prediction of faults without the need to send potentially large amount of data to the server for processing. Implementation of the machine learning model can be used to detect anomaly or classify the type of anomaly. CNN model can be trained on labeled raw vibration signal data to detect and diagnose type of anomaly and then implemented directly on edge nodes in a way that smart sensor reports only detected and diagnosed anomalies to central supervisory system. Such workflow reduces the amount of data that is interchanged between edge node and central supervisory system, consequently affects the lower energy demand and has a positive effect on network throughput. The final goal is to use this information for prescriptive maintenance decision making.

Acknowledgement

The Article Processing Charges (APCs) is funded by the European Regional Development Fund, grant number KK.01.1.1.07.0031.

Notice

The paper was presented at MOTSP 2022 – 13th International Conference Management of Technology – Step to Sustainable Production, which took place in Primošten/Dalmatia (Croatia) on June 8–10, 2022. The paper will not be published anywhere else.

6 REFERENCES

[1] Sowmya, K. & Chetan, N. (2016). A Review on Effective Utilization of Resources Using Overall Equipment Effectiveness by Reducing Six Big Losses. *IJSRSET, 2*.

[2] Ompusunggu, A. P., Eryılmaz, K., & Janssen, K. (2021). Condition monitoring of critical industrial assets using high performing low-cost MEMS accelerometers. *Procedia CIRP, 104*, 1389-1394. https://doi.org/10.1016/j.procir.2021.11.234

[3] 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

[4] Koene, I., Viitala, R., & Kuosmanen, P. (2019). Internet of Things Based Monitoring of Large Rotor Vibration with a Microelectromechanical Systems Accelerometer. IEEE Access, 7, 92210.92219. https://doi.org/10.1109/ACCESS.2019.2927793

[5] Pech, M., Vrchota, J., & Bednář, J. (2021). *Predictive Maintenance and Intelligent Sensors in Smart Factory: Review*, p. 40.

[6] Koene, I. (2020). *IoT connected device for vibration analysis and measurement*, p. 15,

[7] Jaber, A. A. & Bicker, R. (2016). Design of a Wireless Sensor Node for Vibration Monitoring of Industrial Machinery. *International Journal of Electrical and Computer Engineering 6*(2), 639-653

[8] Magadán, L., Suárez, F. J., Granda, J. C., & García, D. F. (2020). Low-cost real-time monitoring of electric motors for the Industry 4.0. *Procedia Manufacturing, 42*, 393-398. https://doi.org/10.1016/j.promfg.2020.02.057

[9] Al-Naggar, Y. M., Jamil, N., Hassan, M. F., & Yusoff, A. R. (2021). Condition monitoring based on IoT for predictive maintenance of CNC machines. *Procedia CIRP, 102*, 314-318. https://doi.org/10.1016/j.procir.2021.09.054

[10] Chanthakit, S. & Rattanapoka, C. (2018). MQTT Based Air Quality Monitoring System using Node MCU and Node-RED. *Seventh ICT International Student Project Conference (ICT-ISPC)*, Nakhonpathom, 1-5. https://doi.org/10.1109/ICT-ISPC.2018.8523891

[11] Lekic, M. & Gardasevic, G. (2018). IoT sensor integration to Node-RED platform. *The 17th International Symposium (INFOTEH)*, Jahorina, East Sarajevo, 1-5. https://doi.org/10.1109/INFOTEH.2018.8345544

[12] Meng, Q. & Zhu, S. (2020). Developing IoT Sensing System for Construction-Induced Vibration Monitoring and Impact Assessment. *Sensors, 20*(21), p. 6120. https://doi.org/10.3390/s20216120

[13] Curman, M., Kolar, D., Lisjak, D., & Opetuk, T. (2021). Automated and Controlled Data Collection Using Industrial IoT System for Smart Maintenance. *Tehnički glasnik, 15*(3), 401-409. https://doi.org/10.31803/tg-20210728122543

[14] Atmoko, R. A., Riantini, R., & Hasin, M. K. (2017). IoT real time data acquisition using MQTT protocol. *J. Phys. Conf. Ser., 853*, p. 012003. https://doi.org/10.1088/1742-6596/853/1/012003

[15] What is MQTT? Definition and Details. https://www.paessler.com/it-explained/mqtt (accessed Apr. 12, 2022).

[16] Doc | Tinkerforge. https://www.tinkerforge.com/en/doc/Hardware/Bricks/HAT_Brick.html (accessed Apr. 04, 2022).

[17] Looney, M. (n/a) MEMS Vibration Sensing: Velocity to Acceleration, p. 2. https://www.mouser.com/pdfdocs/MEMS-Vibrating-Sensing-Velocity-to-Acceleration.pdf

[18] ISO 10816 Vibration Severity Standards. https://www.reliabilitydirectstore.com/articles.asp?id=122 (accessed Apr. 13, 2022).

Authors' contacts:

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

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

Martin Curman, MEng
Klimaoprema d.d., Gradna 78A, 10430 Samobor, Croatia
martin.curman@outlook.com

Michał Pająk, Assoc. Prof. PhD
Department of Thermal Technology,
University of Technology and Humanities in Radom,
Stasieckiego Street 54, 26600, Radom, Poland
m.pajak@uthrad.pl