VOL. 18, NO. 3-2, 2019, 7-12
www.elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Automotive Real-Time Data Acquisition Using Wi-Fi Connected Embedded System

Ahmad Faiz Ab Rahman[1], **Hazlina Selamat**[2*], **Ahmad Jais Alimin**[3], **Mohd Taufiq Muslim**[1], **Muhammad Mazizan Msduki**[2] and **Nurulaqilla Khamis**[2]

[1]Apt Touch Sdn. Bhd., UTM-MTDC Technology Centre, UTM Technovation Park, 81300 Skudai, Johor, Malaysia.
[2]Centre for Artificial Intelligence & Robotics, Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, Malaysia.
[3]Faculty of Mechanical & Manufacturing Eng., Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia.

[*]Corresponding author: hazlina@fke.utm.my, Tel: 607-5535324, Fax: 607-5566272

Abstract: The advancement in embedded systems, which includes the mass deployment of internet-connected electronics, allows the concept of Internet of Things (IoT), to become a reality. This paper discusses one example of how an internet-connected embedded system is utilized in an automotive system. An Electronic Control Unit (ECU), which functions as a control unit in a fuel injection system, are equipped with Wi-Fi capability and installed on 110cc motorcycle. The ECU is connected to multiple sensors that is used by the ECU as part of control system, as well as giving raw data in real time to the server by using Wi-Fi as the communication medium. The server will accumulate data transmitted from ECU by using MQTT protocol, chosen due to its minimal data profile. The data can be visualized through web portal, or opened by any other web-enabled devices. The data collected may also be used later for any other purposes, such as On-Board Diagnostics (OBD) system, etc.

Keywords: Internet of Things (IoT), ESP8266, Message Queue Telemetry Transport (MQTT), Node-RED, 802.11.

Article History: received 4 September 2019; accepted 1 December 2019; published 24 December 2019.

1. INTRODUCTION

The advancement in new and emerging technologies, such as wired/wireless internet embedded systems, cloud systems as well as faster internet, gives the new field of technologies, such as the concept known as Internet of Things (IoT). This concept revolves on the idea that low power consuming devices, equipped with wireless internet communication capability as well as multiple sensors, will be able to collect and send data to the network of high-performance computer servers known as cloud farms, where all heavy computations are done here, so that the embedded systems do not need to do all the heavy tasks. The output data will then be presented to the interested parties through a Graphical User Interface (GUI) which also connected to the cloud farm through the wired/wireless internet.

In this paper, the concept described above is implemented on an automotive system, where an 110cc motorcycle, which has been retrofitted with an Electronic Fuel Injection (EFI) system before. EFI is a system where the main controller, usually called Electronic Control Unit (ECU), is tasked to mix air and fuel in an Internal Combustion Engine (ICE) with efficient and precise manner. The injected fuel amount and the injection timing are dictated from multiple inputs from sensors connected to the ECU.

Since ECU already possess all the sensor readings during its operation, the same data will also be fed to an ESP8266, a 32-bit Wi-Fi System on Chip (SoC), which is tasked to further transmit it to the server by utilizing Message Queuing Telemetry Transport (MQTT) protocol, a lightweight publish-subscribe based messaging protocol across the internet. While the SoC acts as a MQTT client, the MQTT server software, sometimes called a broker, will listen and then pass the data received to a Node-RED flow. Node-RED is a NodeJS based IoT development platform which used flow-based programming for wiring up inputs and outputs such as MQTT client, API or even online services. The flow will then execute relevant data processing, before the output being presented in a dynamic web application service, which can be opened by devices such as another PC or smartphones.

2. LITERATURE REVIEW

It is important to remember that for this project, the overall setup will be divided into client and server. Both are constructed separately and only connected together through the internet connection. The client part obviously will be attached close to the sensors, which is an EFI system equipped motorcycle. Meanwhile, the server side that collects and process data will not be physically connected but wirelessly received data transmitted by the client.

For the client side, sensors have to be attached on the motorcycle and connected to the client-side transmitter so that data readings from the sensors will be sent through transmitter. However, since the motorcycle in question is known to be equipped with an EFI system, it is such a

waste to not utilizing the sensors that is built as part of the EFI system, such as what is shown in [1] and [2]. In addition, [1] and [2] mentions that the main controller of the system, ECU is actually an embedded system centred around a microcontroller, which means direct communication between ECU and client-side transmitter is possible. This means any sensor readings sent from sensor to the microcontroller could also be sent to the transmitter through a communication line, such as UART. There is also another literature [3] that clearly states that communication between ECU being used in the research is connected with the PC through serial port.

For the transmitter side, a 32-bit Wi-Fi System on Chip (SoC) named ESP8266 is chosen. Manufactured by Espressif Systems, it is based on Xtensa L106 RISC architecture, operated on 80MHz [4][10]. There are 16 General Purpose Input Output (GPIO) pins, an Inter-Integrated Circuit (I^2C) bus port, a Serial Peripheral Interface (SPI) bus port, two Inter-IC Sound (I^2S) bus ports, three PWM pins and an ADC input pin [4][10]. Due to its extensive amount of documentations [4], libraries and example codes [5-7], it is not only sought by the makers and hobbyist, but also chosen by the researchers in many published papers [8-17].

While there are researchers utilizing ESP8266 to connect to server through numerous methods such as to HTTP/MySQL protocol to HTTP/MySQL/PHP server [8], or HTTP API to cloud services like ThingSpeak [14] or Microsoft Azure [17], others utilize a lightweight protocol called Message Queue Telemetry Transport (MQTT) [9-13][15][16]. MQTT works by clients, which is divided to publisher, the one that sends data, while the other one is called subscriber, which receives data [10]. All connections must through the broker, also known as MQTT server. There are numerous examples of MQTT used by researchers, such as ESP8266 used as MQTT publisher to local MQTT broker such as Mosquitto [9] or Mosca [12], or same setup but using public broker [10][11][15]. The popularity of MQTT is due to its lightweight characteristics, as well as abundance of libraries and code related to it.

While there are different methods utilized by researchers to for receiving data (subscribe) from MQTT broker, such as using readily available MQTT client on PC and Android devices [9-11], in this project a software called Node-RED is used. Node-RED is a tool written in Node.JS which connects IoT devices and software together. Like ESP8266, Node-RED is also popular among researchers. Originally developed by IBM as an open source project in 2013 [18][19], it is now extensively used different IoT related projects, due to its simple to use flow-based approach to connect devices and codes altogether.

Due to its simple learning curve, it is also recommended by some researchers as an education tool for learning basic IoT technology [19]. However, its vast options for different types of communication protocols and mediums also attract researchers to consider Node-RED to be part of the possible replacement for industrial control systems [21][22], even though it is also excellent to be part of wireless sensor networks when installed at compatible devices such as Raspberry Pi [20][23]. However, due to its built-in nodes that allows MQTT protocol connectivity [21], allows us to connect Node-RED with ESP8266, which is the setup that will be done on this project.

3. METHODOLOGY

In order to fully understand the overall setup of this research project, a diagram was prepared and shown in Figure 1. There is hardware on both client and server sides. Clients are connected to the server through the Wi-Fi protocol. Both client and server are located inside the same local network, which is provided by Wi-Fi router modem.

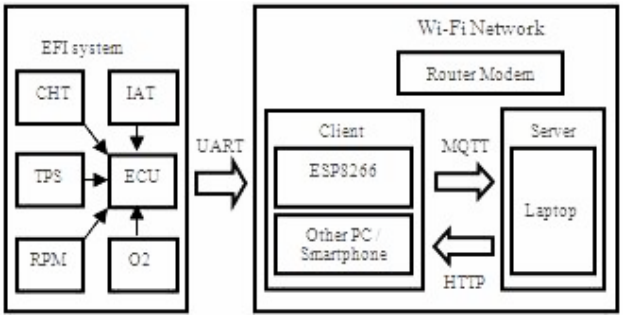

Figure 1. Diagram showing the hardware used in data acquisition system

Client side, as it is named, is an embedded system tasked to collect data from sensors through the ECU, which is part of the EFI system. Communication is done through Universal Asynchronous Receiver/Transmitter (UART) bus in serial port that is available on ECU cables. After all required data are received by the client; it will be converted into a simple string and transmitted through Wi-Fi to the server side, utilizing MQTT protocol. Since the hardware on the client side is quite minimal, to data processing is further done here. This will ensure that the data fetching process can be done as fast as possible, thus reducing data latency.

Unlike client-side setup, server-side hardware is usually powerful, which might be part of the cloud server farm, or at least a decent computer, which are several magnitudes faster at computation than a single microcontroller at client side. This is where all the resource intensive computation is done. In our case, a personal laptop with average hardware specifications is used as the only hardware for the server-side software to reside in. After all computations are done, the data will be represented on the laptop itself in a Web-based GUI or can be sent through HTTP REST API protocol to another device, such as smartphone, which will also show the same data through an app, if needed.

3.1 EFI System Equipped Motorcycle as Data Provider

Since this project is done for data collection on an automotive system, a fully working vehicle needs to be prepared for this setup. A SYM E-Bonus 110cc motorcycle is prepared and equipped with an Electronic Fuel Injection (EFI) system; where it's Electronic Control Unit (ECU) have capability to communicate to other devices through DB-9 serial port. The aforementioned EFI system is equipped with six different sensors, which are Throttle Position Sensor (TPS), Manifold Air Pressure (MAP), engine revolutions per minute (RPM), Intake Air Temperature (IAT), Cylinder Head Temperature (CHT) and lastly, oxygen sensor (O2). The picture of the motorcycle used for this setup is shown in Figure 2.

Figure 2. SYM E-Bonus 110cc motorcycle used for the project

3.2 Embedded ESP8266 as Data Collection Device

A simple, minimal circuit is developed by using ESP8266 Wi-Fi SoC as the main controller for the client-side embedded system, tasked for data collection and transmission to the server. ESP8266 is a small, 32-bit microcontroller produced by Espressif Systems. Due to its low price and simple to use, it is a popular choice for many electronic design projects which requires Wi-Fi connections. The microcontroller is equipped with multiple input/output peripherals for different needs. However, there are different types of modules, which consist of Printed Circuit Board (PCB) with ESP8266 available in the market, each with different sizes and number of pins available to the user.

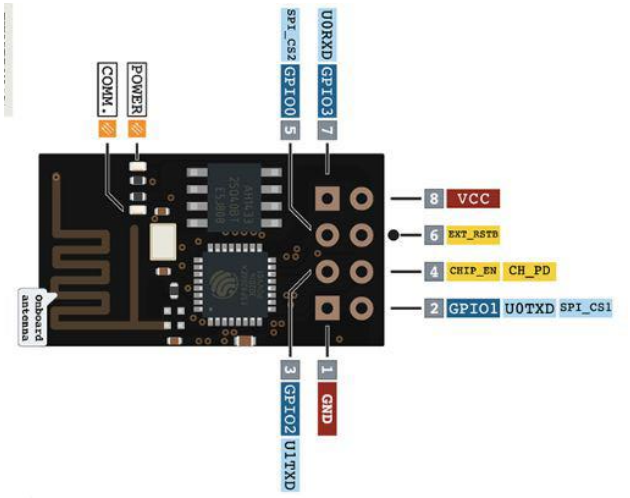

Figure 3. Pinout diagram for ESP-01 module

However, in this setup, the ESP-01 module, which reveals only 8 pins, is more than enough, since the only connection that is needed is the power rails and UART bus, as well as minimal circuitry for other important features like reset and flashing mode trigger. Other circuits such as crystal oscillator are already done on PCB, so this will ease the job. The pinout diagram for ESP-01 module is shown in Figure 3.

In addition of official RTOS-based and non-OS based Software Development Kit (SDK) freely provided by the manufacturer themselves, there are also implementations on Arduino IDE, where user can use Arduino IDE ecosystems alongside ESP8266, thus making programming this microcontroller simpler. Therefore, a short code was written which is tasked to fetch the sensor reading from ECU UART bus.

Basically, the program flow is pretty straightforward, which begins with initialization phase first, before entering an infinite loop phase. The whole code flow is illustrated in Figure 4. The initialization phase contains codes for authentication and connection to local Wi-Fi router as well as to the server-side, which enters through MQTT protocol at port 1883. The infinite loop phase begins with initializing the input string with all zero values of sensor readings, followed by waiting for input string from UART bus. If there is no input from UART detected, the initial values will be sent instead. A check whether connection to the server has been established or not will be done first, before publishing the string containing sensor readings to the server through MQTT protocol.

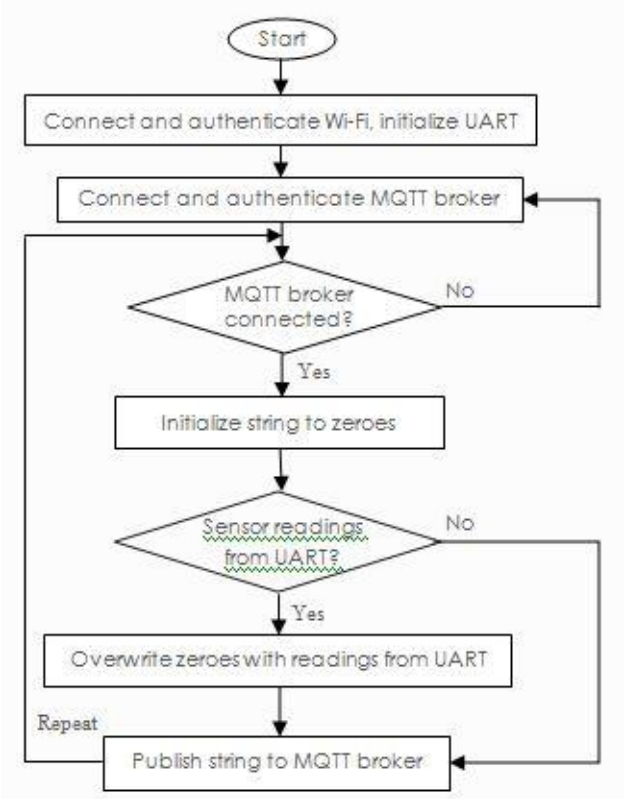

Figure 4. Flow chart describing ESP8266 tasks

3.3 MQTT Server and Node-Red for Data Processing and Visualization

For the server, a laptop is used as hardware for hosting server-side software. The laptop was installed with Node.JS, which is required for installing and running Node-RED. Node-RED is basically simple, GUI flow-based tool for developing Internet-of-Things (IoT) program. The Node-RED GUI was served as a web application, so it can be opened by any decent web browser. A snapshot of Node-RED interface is shown in Figure 5.

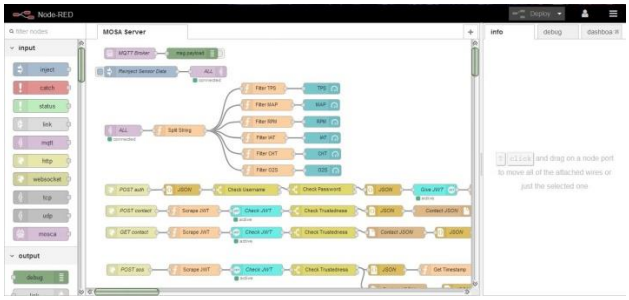

Figure 5. Snapshot of Node-RED interface

In Node-RED, all work is done by connecting and configuring flowcharts. There are a lot of different types of nodes that is available by default, while there are also optional nodes that can be easily installed inside the interface itself. While there are already MQTT publish and subscribe node already available by default, there is no MQTT broker. A MQTT broker node named Mosca is installed later, so there will be no need for external broker.

Figure 6. Snapshot of web-based UI produced by Dashboard UI node in Node-RED

On top of having Node-RED to install a built-in MQTT broker node, another node to be installed are UI nodes. These nodes are used for rendering different types of web-based user interface, like gauges and switches. In our case, meters are used to show the value of different types of sensors. Since there are six main types of sensor used by the EFI system, there will be six gauges rendered in the UI web pages. The snapshot of the web-based UI can be seen at Figure 6, where web-based UI here can be opened by any web browser on any decent PC, tablet or smartphone alike.

4. RESULT AND DISCUSSIONS

There are several different ways of how the result data could be presented for the end user. In this section, three different types of data visualization and representation schemes are developed, each with different purposes and methods. These visualizations are by using debug node, the dashboard UI node, as well storing inside a CSV file. The overall Node-RED flow chart is shown in Figure 7.

As what can be seen in Figure 7, the flow starts with one Mosca MQTT broker node, which acts independently for providing MQTT broker to both ESP8266 and other MQTT publish or subscribe nodes. This then followed by an inject node, used just for resetting the dashboard UI if needed. The MQTT subscribe node, which listens on topic named ‘all’, will produce output data received from both ESP8266 and inject node, which the data is in the form of comma separated string. This string will be presented straight to the debug node, where the string from ESP8266 can be seen in the debug tab. The same data is also split in accordance to different variable types before fed into dashboard UI nodes for rendering gauges. The last part is the file generation node, where a CSV file is generated, and can be opened later in Microsoft Excel. This data can later be used for producing graphs in Microsoft Excel itself.

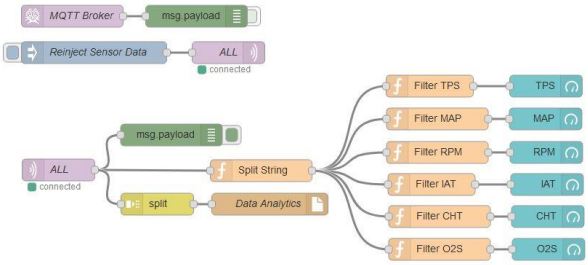

Figure 7. Node-RED flow chart for this project, which includes Mosca MQTT broker, MQTT subscribe node, debug nodes, dashboard UI nodes as well as file output node.

The easiest way would be by utilizing debug capability inside Node-RED. There are debug node available by default inside Node-RED, which simply prints whatever input data it accepts at the debug tab on the right side of the Node-RED user interface. The example of how the debug data output data can be seen on snapshot at Figure 8.

Figure 8. Example of debug node output showing comma separated strings, written in red.

However, the clearest way to present the data is by utilizing dashboard UI node. Dashboard UI is also the best method to present data on mobile devices, such as smartphones and tablets. Figure 9 shows an example of six different gauges, each rendered and controlled by six separate dashboard UI nodes shown earlier in Figure 7 flow chart. Since each node is fed input data of different types of sensors, the gauges in Figure 9 will point values that correspond to the pre-assigned variables.

Figure 9. Dashboard UI nodes rendered web-based GUI shows 6 gauges represents readings of different kinds of sensors

The last method or representing data produced by this platform is by generating a Comma Separated Values (CSV) file. By storing the data inside a CSV file, it can be opened and processed later by other data processing software, such as Microsoft Excel, which can do all sorts of tasks, such as producing graphs. Figure 10 shows how a CSV file can be used inside Microsoft Excel.

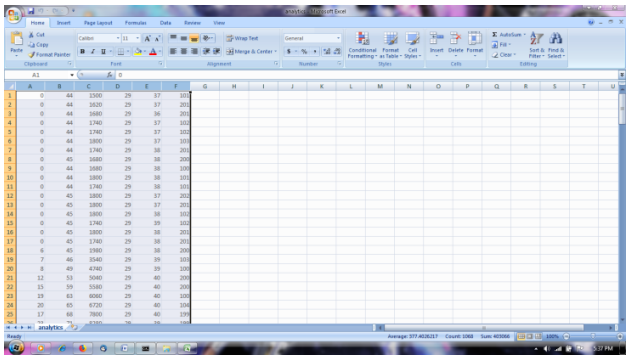

Figure 10: Example of CSV files which stores variables based on types of sensors opened in Microsoft Excel

5. CONCLUSIONS

The emergence of new and emerging technologies such as low-power, Wi-Fi capable System on Chip (SoC) and cloud computing allows rapid deployments of sensor based monitoring system which is easy to build, simple learning curve as well as affordable at lower cost. Implementation on automotive system such as motorcycles will allow us to introduce a way in which a vehicle can be closely monitored not in a controlled environment such as in a laboratory, instead being operated in field test. The collected data, of course could be used for multitude of usages, such as usage for analysis on EFI performance whether based on power or emission levels, or the data can be input to an On-Board Diagnostics (OBD) system for troubleshooting, etc.

REFERENCES

[1] Muslim, M. T., Selamat, H., Alimin, A. J., and Hushim, M. F, "Electronic Control Unit Design for a Retrofit Fuel Injection System of a 4-stroke 1-cylinder Small Engine," *Applied Mechanics and Materials*, 2012. 229: 968-972.

[2] Muslim, M. T., Selamat, H., Alimin, A. J., Rohi, N. M., and Hushim, M. F, "A Review on Retrofit Fuel Injection Technology for Small Carburetted Motorcycle Engines towards Lower Fuel Consumption and Cleaner Exhaust Emission," *Renewable and Sustainable Energy Reviews*, 2014. 35: 279-284.

[3] Muslim M. T., Selamat H., Alimin A. J., Haniff M. F., "Manifold absolute pressure estimation using neural network with hybrid training algorithm," PLoS ONE 12(11): e0188553. https://doi.org/10.1371/journal.pone.0188553.

[4] Espressif Systems, "ESP8266EX Datasheet", Dec. 2015, [Revised Feb. 2018].

[5] M. Schwartz, *Internet of Things with ESP8266*. Packt Publishing Ltd, 2016.

[6] M. Schwartz, *ESP8266 Internet of Things Cookbook*. Packt Publishing Ltd, 2017.

[7] N. Kolban, *Kolban's Book on ESP8266.* 2016.

[8] T. Thaker, "ESP8266 based implementation of wireless sensor network with Linux based web-server," *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, Indore, 2016, pp. 1-5. doi: 10.1109/CDAN.2016.7570919.

[9] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, Agra, 2016, pp. 1-5. doi: 10.1109/R10-HTC.2016.7906845.

[10] R. K. Kodali and K. S. Mahesh, "A low cost implementation of MQTT using ESP8266," *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, Noida, 2016, pp. 404-408. doi: 10.1109/IC3I.2016.7917998.

[11] N. M. Sonawala, B. Tank and H. Patel, "IoT protocol based environmental data monitoring," *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, 2017, pp.1041-1045. doi: 10.1109/ICCMC.2017.8282629.

[12] G. M. B. Oliveira *et al.*, "Comparison Between MQTT and WebSocket Protocols for IoT Applications Using ESP8266," *2018 Workshop on Metrology for Industry 4.0 and IoT*, Brescia, 2018, pp. 236-241. doi: 10.1109/METROI4.2018.8428348.

[13] S. R. Akbar, K. Amron, H. Mulya and S. Hanifah, "Message queue telemetry transport protocols implementation for wireless sensor networks communication - A performance review," *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, 2017, pp. 107-112. doi: 10.1109/SIET.2017.8304118.

[14] Endy Silveira, Samir Bonho, "Temperature Monitoring Through Wireless Sensor Network Using an 802.15.4/802.11 Gateway," *IFAC-PapersOnLine*, 2016. 49: 120-125.

[15] Monika Kashyap, Vidushi Sharma, Neeti Gupta,"Taking MQTT and NodeMcu to IOT: Communication in Internet of Things," *Procedia Computer Science*, 2018. 132: 1611-1618.

[16] Y. Nait Malek, A. Kharbouch, H. El Khoukhi, M. Bakhouya, V. De Florio, D. El Ouadghiri, S. Latre, C. Blondia, "On the use of IoT and Big Data Technologies for Real-time Monitoring and Data

Processing," *Procedia Computer Science*, 2017. 113: 429-434.

[17] Carmelo Ardito, Paolo Buono, Giuseppe Desolda, Maristella Matera, "From smart objects to smart experiences: An end-user development approach," *International Journal of Human-Computer Studies*, 2018. 114: 51-68.

[18] Node-RED, "Documentation", Oct. 2013. [Online]. Available: https://nodered.org/docs. [Accessed July 10, 2018].

[19] Z. Chaczko and R. Braun, "Learning data engineering: Creating IoT apps using the node-RED and the RPI technologies," *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)*, Ohrid, 2017, pp. 1-8.

[20] M. Lekić and G. Gardašević, "IoT sensor integration to Node-RED platform," *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, East Sarajevo, 2018, pp. 1-5. doi: 10.1109/INFOTEH.2018.8345544.

[21] Antonin Gavlas, Jan Zwierzyna, Jiri Koziorek, "Possibilities of transfer process data from PLC to Cloud platforms based on IoT," *IFAC Conferences on Programmable Devices and Embedded Systems (PDeS)*, Ostrava, 2018, pp. 156-161. doi: 10.1016/j.ifacol.2018.07.146.

[22] Mohamed Tabaa, Brahim Chouri, Safa Saadaoui, Karim Alami, "Industrial Communication based on Modbus and Node-RED," *Procedia Computer Science*, 2018. 130: 583-588. doi: 10.1016/j.procs.2018.04.107.

[23] Jiri Skovranek, Martin Pies, Radovan Hajovsky, "Use of the IQRF and Node-RED technology for control and visualization in an IQMESH network," *IFAC Conferences on Programmable Devices and Embedded Systems (PDeS)*, Ostrava, 2018, pp. 295-300. doi: 10.1016/j.ifacol.2018.07.169.