Industry 4.0 – LabVIEW Based Industrial IoT Condition Monitoring System

1st Harvey William Picot
Department of Electrical and Electronic Engineering
Faculty of Engineering and Technology, Liverpool John Moores University
Liverpool, United Kingdom
harveypicot@yahoo.com

2nd Dr Muhammad Ateeq
Department of Electronics & Electrical Engineering
Faculty of Engineering and Technology, Liverpool John Moores University
Liverpool, United Kingdom
m.ateeq@ljmu.ac.uk

3rd Dr Badr Abdullah
Department of Built Environment
Faculty of Engineering and Technology, Liverpool John Moores University
Liverpool, United Kingdom
b.m.abdullah@ljmu.ac.uk

4th Dr Jeff Cullen
Department of Built Environment
Faculty of Engineering and Technology, Liverpool John Moores University
Liverpool, United Kingdom
j.d.cullen@ljmu.ac.uk

Abstract*— As a result of a substantial shift in focus towards a more digital industry, multiple sectors of industry are now realising the potential of Industry 4.0 and Internet of Things (IoT) technology. The manufacturing industry in particular is subject to unexpected machine downtime from component wear over an extended period. With Industrial IoT (IIoT) technology implemented, there is the potential for gathering large quantities of data, which can be used for preventative maintenance. This research article addresses some of the technological requirements for developing an IoT industrial condition monitoring network, whose composition makes use of wireless devices along with conventional wired methods to enable a series of data capture and control operations in amongst a network of nodes. To provide a platform to host these operations, the industry standard fieldbus protocol Modbus TCP was used in conjunction with the LabVIEW development environment, where a bespoke graphical user interface was developed to provide control and a visual representation of the data collected. In addition, one of the nodes acted as the output for hardware displays, which in turn correlated the alarm status of the user interface. By using industry standard communication protocols, it was also possible to enable connectivity between real industry hardware, further extending the capabilities of the system.

Keywords—IIoT, Industry 4.0, Machine down-time, Modbus TCP, LabVIEW

I. Introduction

Industry as a whole and the manufacturing industry in particular are now seeking a solution to overcome some of the modern-day problems and improvement requirements commonly required within the manufacturing industry. To stay competitive within their respective markets these companies must ensure they continuously improve in relation to three key areas i) manufacturing processes (through technology) ii) process/production efficiency and iii) cost. Without introducing different methods and technology advancements/components, a window may open for potential competitors to overtake them in their respective market, thus highlighting the importance of innovation, where innovation can be classified as one of the primary methods for increasing/maintaining market share as stated in [1]. Nevertheless, coming up with solutions in the modern age can be difficult, and in some situations expensive. So, the real question is how can they do it? According to Egham, U.K. in [2], 20.4 billion IoT devices will be connected globally in 2020 alone with up to half of the organisations investing in IIoT technology have already implemented a specific strategy [2]. It is not a case of when will organisations realise the potential, but how have they already started to implement the solutions.

II. Related Work

As discussed in [3] it is possible to gain an understanding of previous attempts to produce an industrial condition monitoring system. As described in [3], LabVIEW was used as the graphical user interface to manipulate and analyse the raw accelerometer readings. The project designed and constructed a system using a combination of hardware components (for both the sensing element and for the exchange of the vibration data) to a host output system (laptop). The system subsequently running a LabVIEW application collected and analysed the data. The specific sensor used for detecting vibration was a MEMS Accelerometer or more specifically the MMA7455L accelerometer. It was interfaced with the Arduino microcontroller using the I2C communication protocol, where data was formatted and transmitted using the RF module which was governed by the ZigBee wireless communication protocol as described by the IEEE 802.12.4 standard [3]. The extent of this project does not go far as developing a true IoT system, rather a wireless sensor network (WSN) for machine-to-machine communication.

Later discussed, the system could be improved by interfacing the sensor node with a Wi-Fi router, and communicating with a cloud server using some form of web-based communication protocol such as the hyper-text-transfer-protocol (HTTP), or as described in [4] the MQ telemetry transport (MQTT) protocol.

978-1-7281-3021-7/19/$31.00 ©2019 IEEE

Salai et al. [5] published a paper similar to what is proposed within the current research project. Instead of monitoring machine condition parameters, the project was more concerned with environmental monitoring. The overall topology was similar, only it utilised a GSM module to communicate the LabVIEW application with a mobile phone. As a result, the user could gather data from the system by simply reviewing the alert file sent from the user interface in the form of an SMS text message.

The work carried out by Laxmi Prasanna et al [6] again utilised the LabVIEW software to monitor the condition of PV solar cells. An array of four PV solar panels interfaced with a bespoke LabVIEW application, which communicated to a web server through a raspberry Pi, acting as a gateway. The general parameters this system monitored included maximum power, voltage, current, efficiency, irradiance, temperature and so forth. Like the alerting mechanism in the current project, if any of the above parameters exceeded the allowable tolerances alert was activated.

Alphonsa and Ravi [7] developed an earthquake early warning system that utilised accelerometers and ZigBee modules to wirelessly transfer vibration data to a host LabVIEW application. From this, the operator was able to monitor seismic activity using sensors and provided with both numerical and graphical representations of the data collected.

III. METHODOLOGY

A. *Description*

As shown in Fig 1, the entire network can be summarised by a combination of inputs and outputs, wired and wireless sensing nodes, along with the necessary hardware to route all of the incoming and outgoing traffic. It also provides distributed monitoring and control through the host PC LabVIEW application. To organise the nodes so that they can be individually accessed, their specific IP addresses (static IP address) are fixed as opposed to assigned by the DHCP.

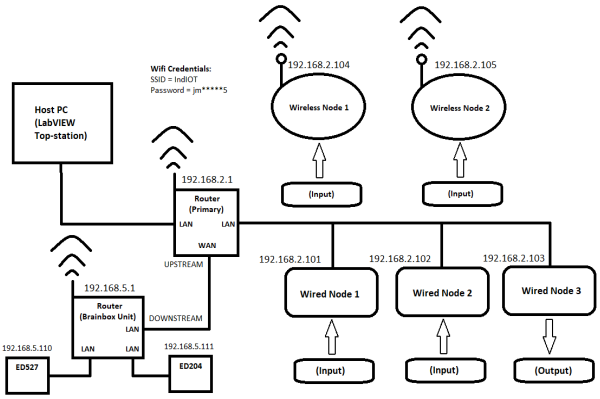

Figure 1 Network Topology

B. *Hardware*

A combination of sensors processors and network devices were used to construct all the elements of the system. In addition, the brainbox 527 and 204 I/O devices were used to further demonstrate the capabilities of the network by enabling communication to real industrial equipment. Listed in Table 1 below are the components.

Table 1 Component Itemisation and Application

Component	**Application**
X3 Arduino UNO	To host the sensor interfacing requirements, along with network connectivity in conjunction with the W5100 network shield.
X3 Wiznet Ethernet Shield W5100	Used alongside the Arduino to enable connectivity using Modbus TCP.
X1 Grove 6-axis accelerometer and compass (LSM303D)	To provide measurement information about the general inclination, vibration and heading.
X1 Grove High Temperature Thermal Couple	To allow temperature data to be collected by the LabVIEW top-station and utilised to enable a fall-back network in controlling the fan (installed on the brainbox unit),
X3 Adafruit 8x8 Bi-Colour LED Matrix	To display various alarm graphics corresponding to each type of the sensor data.
X1 Grove 16x2 LCD RGB	To display the raw sensor data, and to distinguish the most critical alarm out of each sensor through changing the colour of the RGB backlight.
X2 WIO Node Wifi Module	To enable wireless communication for collection of the proximity data from the sharp infrared (IR) distance sensors.
X2 Sharp IR Distance Sensor GP2Y0A60SZLF	As stated, these sensors are used in conjunction with the wireless modules to detect proximity data and to transmit data to the LabVIEW.

X2 Serial-to-USB converters	To enable the Arduino IDE to write to the flash memory of the Wio node WiFi modules.

C. *Software*

a) Arduino

The Arduino IDE (version 1.8.8) was used to write the software, in addition to the following libraries used as listed in Table 2. For reference purposes, Fig 2 shows the Arduino IDE.

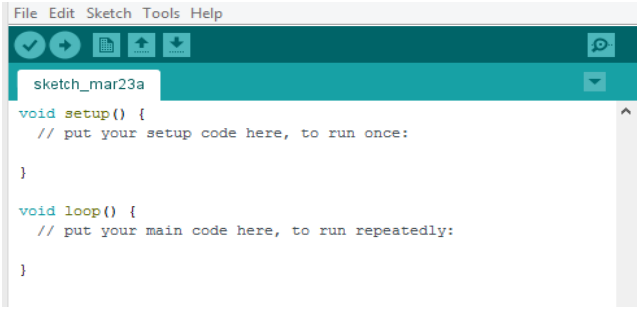

Figure 2 Arduino IDE

Table 2 Arduino Library Itemisation

Library	**Application**
Wire.h	The wire library is used to enable the I2C protocol.
SPI.h	Although SPI was not directly used for any of the development of the code, it was necessary to enable communication between the Arduino and the W5100 shields
Ethernet.h Modbus.h ModbusIP.h	In addition to the ModbusIP and Ethernet libraries, all the necessary functions could be accessed programmatically through the Arduino IDE.
Adafruit_GFX.h Adafruit_LEDBackpack.h	The two libraries were used to access the necessary functions to control the LED displays.
High_Temp.h	This library came with the Grove high temperature sensor. The library incorporates the I2C protocol and provided the higher level functions to request the data.
LSM303D.h	The LSM303D library again provided functions to bypass the lower level I2C commands and enabled data collection to be a simpler process.
ESP8266WiFi.h ModbusIP_ESP8266.h	The final two libraries were primarily used to enable wireless nodes and the ability to use the Modbus TCP protocol.

b) LabVIEW Development Envirnoment

The LabVIEW 2018 version was used to develop the application. In addition to the LabVIEW development environment, the distributed supervisory control (DSC) package was used to include an API to enable the Modbus TCP protocol. Other than this addition, most functions came with this version of LabVIEW and do not require listing.

IV. RESULTS

A. *LabVIEW Front Panel*

By utilising core attributes of the LabVIEW software, it was possible to develop a full interface to provide control over the connected/selected input/output nodes. The front panel was segregated into normal operation and advanced operation as to ensure all parties that may use the software need not to ask for additional assistance. It was designed to be as user friendly as possible and require as little manual intervention. Once the software was appropriately configured for the particular type of monitoring, i.e. the allowable parameters for each type of sensor, all that would be left is to start the application. The system was designed to establish a connection between both the wired and wireless nodes distributed around measuring various aspects of any industrial system. If at any point the connection to one or more of the nodes fails, the software was able to recognise and advise on the appropriate course of action to rectify the issue(s). Fig 3 below illustrates the main panel where most operators will assume control over the network. When the software application is started, it immediately attempts to connect to the sensor nodes, where afterwards the data capture procedure commences.

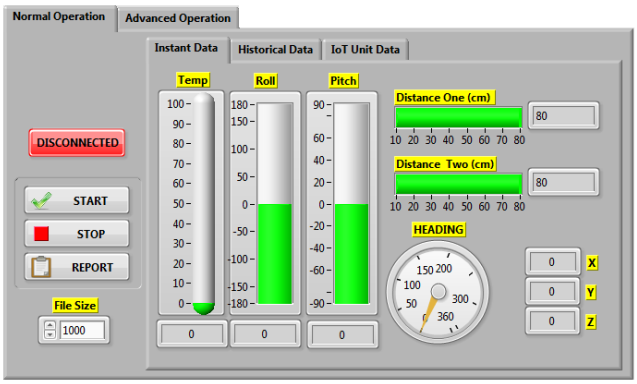

Figure 3 User Interface (Normal Operation)

The normal operation tab includes additional tabs for the different representations of the incoming sensor data. The instantaneous tab houses indicators to gain a clear and prompt understanding of any potential issues with the sensors. For a more detailed view, the operator can then refer to the historical data tab as shown by Fig 4.

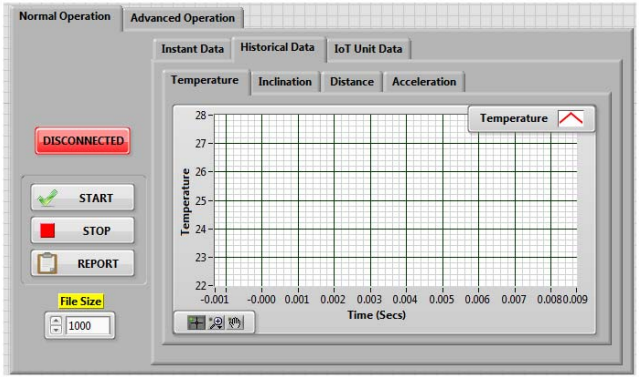

Figure 4 User Interface (Historical Data)

Since the Modbus TCP protocol was used, it was also possible to interface with the brainboxes ED series modules. Brainbox ED module is a replica of an industrial system simulating measurements of an industrial process through various sensors and actuators along with output devices such as proximity sensor, alarm system and production counter, the fan and the seven-segment display.

To showcase the potential of connecting the two systems (user interface with Brainboxes ED unit), the metal proximity sensor was monitored, and the count recorded. In addition, the built-in fan on the unit was activated through comparing the temperature with pre-defined limits that may correspond to an industrial process. Fig 5 illustrates the indicators included for this particular element of the system.

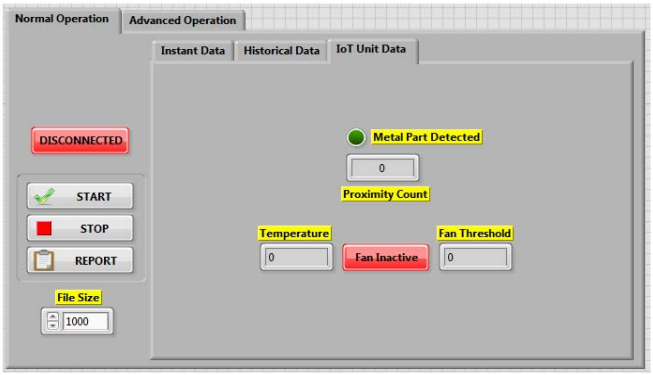

Figure 5 User Interface (Brainboxes Interface Indicators)

B. LabVIEW Block Diagram

The LabVIEW block diagram is where all necessary coding procedures took place. The specific architecture used to format the code was a master slave style. When the application is started, the master interprets all incoming commands by the user, and provides signals to the slave state machine as to which state it should transition too. Fig 6 is the master loop which includes an event-based structure contained within a while loop. To communicate the two loops, queues were utilised.

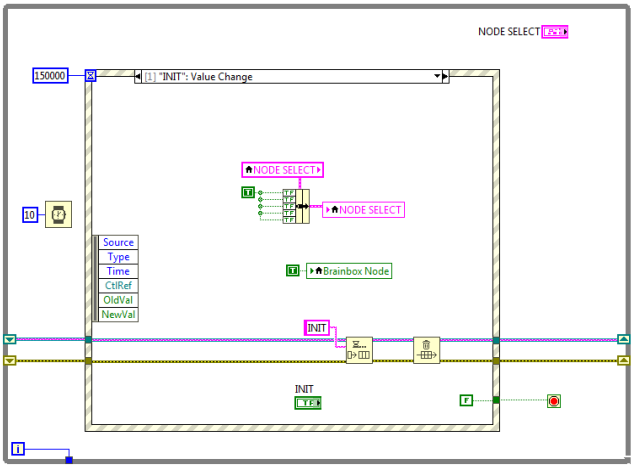

Figure 6 LabVIEW Block Diagram (Master Loop)

After running the application via the start button, the master "enqueues" the initialisation state where the slave will "dequeue" and perform the requested state. This is shown in Fig 7.

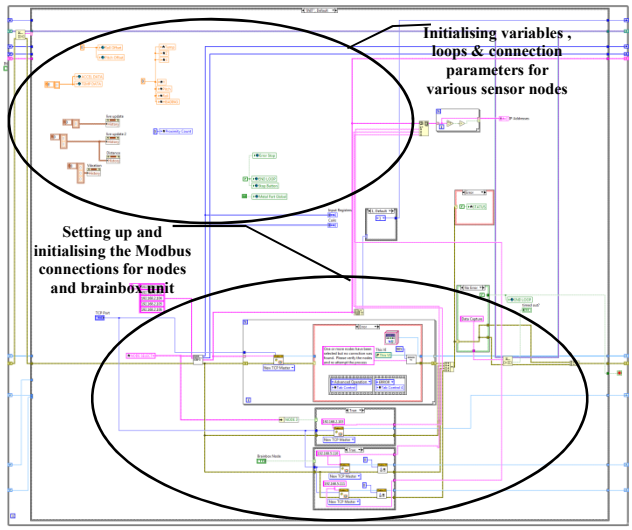

Figure 7 Block Diagram Initialise State

After the correct Modbus instances are generated for each of the connected nodes (identified by IP addresses), the information is passed onto the data capture state. For

the majority, all key features of the application design are performed in this state. It includes a number of parallel while loops with each performing a different task. For example, the main loop simply monitors the incoming data, and updates the displays accordingly. The remaining loops either deal with the interface to the brainboxes unit, or send back data from the application down to the hardware display Arduino node. This will then correlate the alarm status of the graphical user interface on physical displays. Fig 8 is the block diagram for the data capture state.

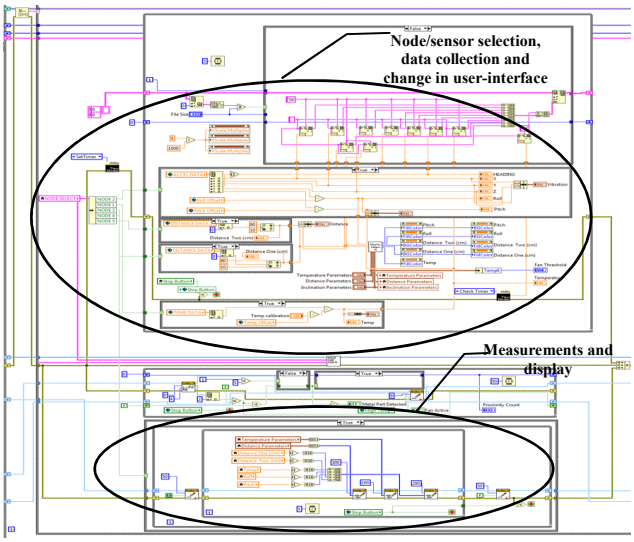

Figure 8 Block Diagram Data Capture State

displays. These displays would be situated closer to the piece of equipment under monitoring. It comprises of three 8x8 LED matrices for each type of measurement, along with a 16x2 LCD with backlight to which provides actual data for each sensor, along with a changing colour depending on the most critical alarm. The following two figures (Fig 9 and Fig 10) show the internal and external view of the display panel.

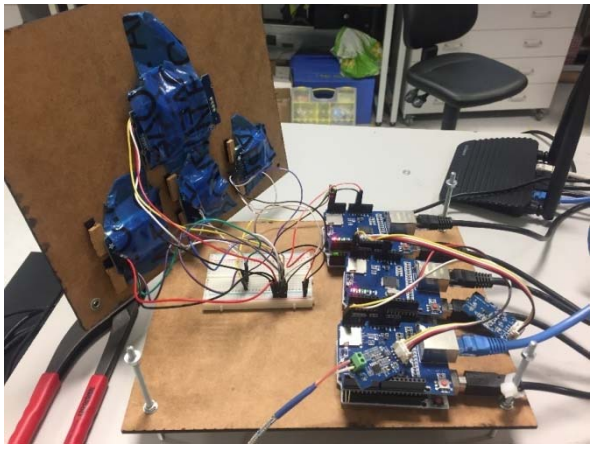

Figure 9 Hardware Assembly (Internals)

As shown by Fig 9, the panel houses the Arduino/w5100 devices along with the daisy chained I2C connections for each of the displays.

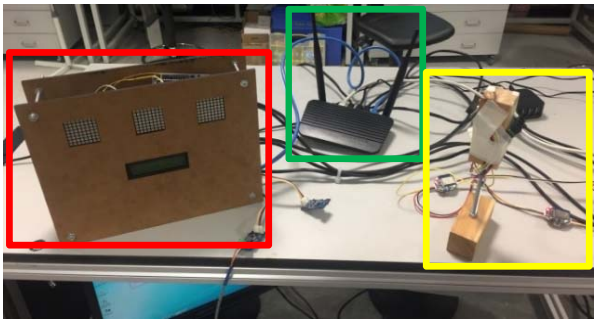

Figure 10 Hardware Assembly (Externals)

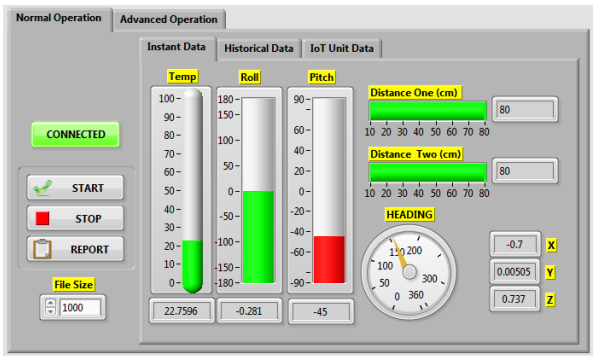

Figure 11 Graphical User Interface GUI (Pitch at -45)

C. GUI and Panel in Operation

As shown by Fig 11, the respective indicators on the instant tab change according to variations in the sensor data. For better visualisation of the alarm status, the colour scheme follows that of a traffic light; green for no concern, yellow for a warning and to indicate some action may be necessary, and finally red indicating the level is critical and requires immediate action by the operator. Fig 12 shows the display panel in operation. As can be seen, the LED matrix has changed to the most critical state, with the LCD screen displaying the actual value of the pitch angle. The colour of the backlight has also changed accordingly.

D. Results CSV Reports

For gathering the data over the period of system operation, periodic CSV reports, as in Table 3, were generated where the number of samples was dictated by the control present in the normal operation tab. This functionality can be activated by simply clicking the report button. The corresponding columns represent the data collected by each sensor, with the accelerometer data split into each of the

Figure 12 Hardware Display (Pitch at -45)

three axis of motion, the timestamp, temperature and distance measurements, roll and pitch angle in degrees.

Table 3 Results CSV Report

Time Base	Accel_X	Accel_Y	Accel_Z	Roll	Pitch	Heading	Temperat	Distance1	Distance2
10.47	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.47	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.48	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.48	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.49	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.49	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.50	0.026668	-0.00244	1.039586	0.002983	-0.01659	164	22.76411	80	80
10.50	0.026644	-0.00247	1.039505	0.004862	-0.01781	162	22.76411	80	80

The number of files generated can be tracked by viewing the report tab under normal operation, as shown by Fig 13.

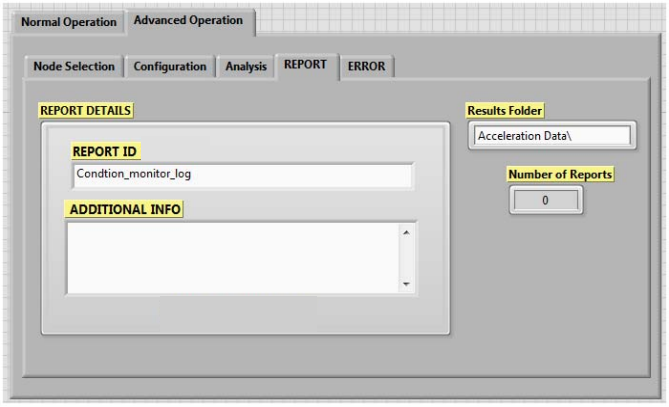

Figure 13 Report Controls

In addition to viewing the number of files generated, the operator can also change the name of the files, along with including any additional information, which may be useful to the individuals analysing the files. For convenience, the file location is also specifiable where a folder is generated if no such folder exists prior.

V. Conclusion and Recomendations

Condition monitoring is an increasingly important process to undergo with regards to reducing the occurrence of unexpected machine down-time. This article attempted to address the issue of catching machine failure before occurrence, and provided a multitude of different methods for assessing the condition of the equipment (i.e. through different sensor devices), and in addition provided clear representation (visuals) of when a particular sensor(s) had detected an anomaly in the data. The future work will enable a full interface with web-based devices through simple extensions to the existing design. It would be a simple process to enable this functional element. Further work would expand the system to not only monitor the process conditions but also will add the hardware and sensors to extend the applications to carry out environmental conditions monitoring. The analysis aspect could also be improved through utilising some of the signal processing libraries already available on the development environment.

Acknowledgment

The project would like to acknowledge the funding provided by Faculty Pump Prime that enabled the purchase of electronic components and sensors.

References

[1] Egham, U.K (2017) *Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, UP 31 Percent From 2016* [online].

[2] IIoT. (2017) *Information Technology Talking to operational Technology* [online].

[3] Upadhye, M. Y. Borole, P. B. and Sharma, A. K. (2015) Real-time wireless vibration monitoring system using LabVIEW *International Conference on Industrial Instrumentation and Control (ICIC)* [online] Pune India, 28-30 May pp. 925-928.

[4] Kanawaday, A. and Sane, A. (2017) Machine learning for predictive maintenance of industrial machines using IoT sensor data *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS),* Beijing China, 24-26 November pp. 87-90.

[5] Salai Thillai Thilagam, J. Sarath Babu, T. Siva Reddy, B. (2017) Weather monitoring system application using LabVIEW *2018 2nd IEEE International Conference on I-SMAC*, Palladam, India, 30-31 August pp. 52-55.

[6] Laxmi Prasanna J. Lavanya, D. Anil Kumar, T. (2017) Condition Monitoring of a Virtual Solar System using IoT *2017 2nd International Conference on Communication and Electronics Systems (ICCES),* Coimbatore, India, 24-26 November pp. 286-290.

[7] Alphonsa, A. Ravi, G. (2016) Earthquake early warning system by IOT using Wireless sensor networks *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India, 23-25 November pp. 1201-1205.