MQTT Based Air Quality Monitoring System using NodeMCU and Node-RED

Somphop Chanthakit
College of Industrial Technology
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
s5903053816027@email.kmutnb.ac.th

Choopan Rattanapoka
College of Industrial Technology
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
choopanr@kmutnb.ac.th

***Abstract*—This paper present an implementation of MQTT based air quality monitoring system. The air quality measurement device is a hardware using ESP8266 NodeMCU that connects to sensors for measuring temperature, humidity, concentration of carbon monoxide (CO), ozone gas (O_3), and PM2.5. The firmware of device makes the device act as a publisher that reads the sensor data and sends them to MQTT Broker. Node-RED is used to be a subscriber that subscribes to receive data from MQTT Broker. With Node-RED, we can easily make a flow to manage and handle received data. Then, Node-RED will send data to the air quality monitoring dashboard which is a responsive web application to display data in gauge, text and chart user interface. In addition, when the value of some data exceed the configured range, Node-RED will send an alarm message via LINE Notify to notify users.**

Keywords—Internet of things, MQTT, Air quality, Sensors, Node-RED, ESP8266 NodeMCU

I. INTRODUCTION

Indoor air pollution and poor urban air quality are listed as two of the world's worst toxic pollution problems in the 2008 Blacksmith Institute World's Worst Polluted Places report [1]. From the 2014 World Health Organization report, air pollution in 2012 caused the deaths of around 7 million people worldwide [2]. Thailand's Air Quality Index (AQI) is an index for reporting daily air quality. The AQI focuses on health effects that people may experience within a few hours or days after breathing polluted air. The Pollution Control Department (PCD) calculates the AQI from 5 major air pollutants regulated by the National Ambient Air Quality Standards (NAAQS): ground-level ozone (O_3), carbon monoxide (CO), sulfur dioxide (SO_2), nitrogen dioxide (NO_2) and airborne particulate matter 10 micrometers or less in diameter (PM10). Ground-level ozone and airborne particles are the two pollutants that pose the greatest to human health in Thailand according to the PCD. In January 2018, Bangkok air quality was extremely unhealthy with the level of particles sized at equal or smaller than 2.5 microns (PM2.5) reach over 200 $\mu g/m^3$ while the Thai standards safe level of PM2.5 is 50 $\mu g/m^3$. However, The PCD had failed to warn the public because the lack of proper air-quality warning system and the PM2.5 is not included in the AQI calculation as the nationwide installation of PM2.5 monitoring devices was not yet complete.

Nowadays Internet of Things (IoT) [3] gained a great attention from researchers, since it becomes an important technology that allowing a communications between objects, machines and every things together with peoples. The IoT represents a system, which consists things in the real world, and sensors attached to or combined to these things, connected to the Internet via wired and wireless network structure. The connected devices need a protocol using which they could communicate only when it is required. Devices with constrained resources should be able to communicate with various other devices. The Message Queue Telemetry Transport (MQTT) protocol [4] is one of the most popular protocol that play as an important role in IoT communication. Because MQTT is so lightweight and can be used by some of the smallest measuring and monitoring devices, and it is designed to overcome the challenges of connecting the rapidly expanding physical world of sensors, actuators, mobile phones, and tablets with established software processing technologies. These principles also turn out to make this protocol ideal for the emerging Machine-to-Machine (M2M) or IoT world of connected devices where bandwidth and battery power are at a premium.

Thus, this paper presents an implementation of a low-cost air quality monitoring system by using a ESP8266 NodeMCU connects to a temperature and humidity sensor (DHT22), a carbon monoxide sensor (MQ-7), an ozone gas sensor (MQ-131), and PM2.5 dust sensor (PPD42NJ). The ESP8266 collects the data from sensors and publishes them to a MQTT Broker and also we implement an air quality monitoring dashboard using Node-RED [5] that subscribes to the MQTT Broker and retrieves sensor data to display them in the form of gauge, text and chart user interface. Moreover, when the value of some data exceed the standard range, the system will notify people via LINE Notify [6].

II. BACKGROUND

A. MQTT Protocol

Message Queuing Telemetry Transport (MQTT) is an ISO standard (ISO/IEC PRF 20922) lightweight publish/ subscribe messaging protocol. It is useful the devices with limited resources.

The MQTT protocol is based on the principle of publishing messages and subscribing to topics, or "pub/sub". Multiple clients connect to a broker and subscribe to topics that they are interested in. Clients also connect to the broker and publish messages to topics. Many clients may subscribe to the same topics and do with the information as they please. The broker and MQTT act as a simple, common interface for everything to connect to. Messages in MQTT are published on topics. There is no need to configure a topic, publishing on it is enough.

Topics are treated as a hierarchy, using a slash (/) as a separator. Clients can receive messages by creating subscriptions. A subscription may be to an explicit topic, in which case only messages to that topic will be received, or it may include wildcards. Two wildcards are available, + or #.

978-1-5386-7805-3/18/$31.00 ©2018 IEEE

B. Node-RED

Node-RED is a flow-based development tool developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a browser-based flow editor, which can be used to create JavaScript functions. The elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14 MQTT nodes can make properly configured TLS connections. In 2016, IBM contributed Node-RED as an open source JS Foundation project.

C. LINE Notify

LINE Notify is a notification service that allows users to link to a web service and receive notification from the official account "LINE Notify" that is provided by LINE [7]. LINE Notify can be used free of charge. LINE Notify supports the event notification from Mackerel: a server monitoring platform for engineers, GitHub: a shared web service for software development projects, IFTTT: a web-based service which allows users to link together other web services, and also some other services that can connect to LINE Notify API.

III. System Design

The air quality monitoring system consists of three main components shown in Fig. 1.

The first component is an air quality measurement device. We connected sensors to ESP8266 NodeMCU that acts as a publisher. The ESP8266 NodeMCU will collect, parse and transform data from sensors, then publish them to MQTT Broker.

The second component is MQTT Broker. We use Eclipse Mosquitto [8], which is an open source message broker that implements the MQTT protocol version 3.1 and 3.1.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full server.

The third component is an air quality monitoring dashboard and Line Notify Service. We use Node-RED that acts as a subscriber to receive data from MQTT broker. With Node-RED, we design flows to manage and handle the received data to display them in the form of gauge, text and chart on the dashboard. In addition, we set some constraints in Node-RED that if the value of data exceed the configure range, Node-RED will contact LINE Notify service to send a LINE message to warn users.

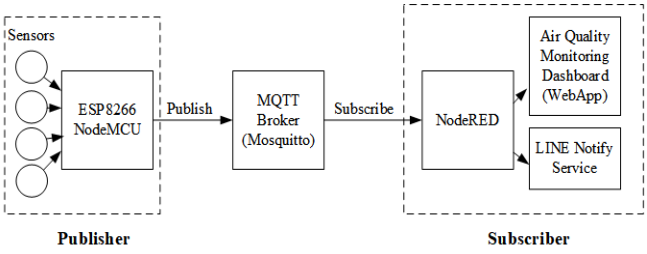

Fig. 1. Air Quality Monitoring System Architecture.

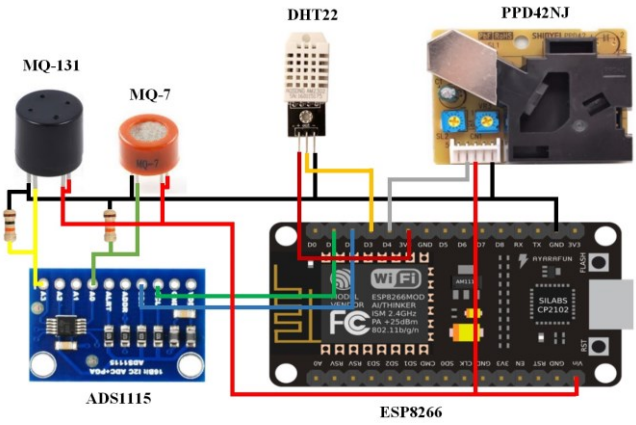

Fig. 2. The Components of Device Hardware.

A. Air Quality Measurement Device (Publisher)

The implementation of air quality measurement device can be divided into two parts: hardware and software.

1) Hardware

The device hardware's design shown in Fig. 2. The main hardware component of the device is a ESP8266 NodeMCU board that connects to sensors. In this device, there are four sensors.

- Temperature and humidity sensor (DHT22), a low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and output a digital signal on the data pin.
- Carbon monoxide sensor (MQ-7), a simple to use CO sensor. It can detect CO-gas concentrations from 20 to 2000 ppm. The output of sensor is an analog resistance.
- Low concentration ozone gas sensor (MQ-131), a gas sensor that use SnO_2 as sensitive material. It can detect ozone-gas concentrations from 10 ppb to 2 ppm. The output of sensor is an analog resistance.
- Grove – particle sensor (PPD42NJ), a sensor that create digital output to particulate matters (PM) whose size is around 1 micro meter or larger.

DHT22 and PPD42NJ sensors give digital output, so we can connect them directly to ESP8266 NodeMCU digital GPIO pin. However, ESP8266 NodeMCU has only one analog pin, thus we use ADS1115 that is a 16-bit analog to digital converter to take the analog outputs from MQ-7 and MQ-131 sensors and convert them to digital outputs. Then, we connect ADS1115's SCL and SDA pins to ESP8266 NodeMCU's digtal GPIO pins so that ESP8266 NodeMCU can read data from these two sensors.

2) Software

The software or firmware that we implemented on the device will collect the data from four sensors and transform them to the standard unit of measurement. Then, we use a MQTT client library [9] to publish data to MQTT Broker.

For the temperature and humidity, the data from DHT22 sensors already give the value of temperature in degree Celsius (°C) and the value of humidity in relative humidity

(%RH). Thus, we write a program to collect these two data from sensors every 2 seconds. Then, the program will combine two data into one string in the format of *temperature,humidity* and publish this data to MQTT Broker in the topic named *"sensor/TEMPAndHUMID"*.

For the data of CO-gas concentration from MQ-7 sensor and the Ozone gas concentration from MQ-131 sensor, the output that program can collect from these sensors are voltage. Thus, we need to calculate the resistance of sensor (R_S) using the equation in (1).

$$R_S = ((V_C / V_{RL}) - 1) \times R_L \quad (1)$$

Where

- V_C is the circuit voltage (5V).
- V_{RL} is the output voltage from sensor.
- R_L is the load resistance (we use 10kΩ).

After we obtain the value of R_S, we can find the CO-gas and Ozone gas concentration in ppm (parts per million) by using the sensitivity characteristic curve in datasheet of each sensor. The program will collect these two data from sensors every 1 second, Then, the program combines two data into one string in the format of *co-gas-value,ozone-gas-value* and publish this data to MQTT Broker in the topic named *"sensor/COAndO3"*.

For the data of PM2.5 from PPD42NJ sensor, the program needs to sample the total duration of "logic low" in 30 seconds (Ratio: *R*) because this duration illustrates the dust density of environment. Then, we can calculate the PM2.5 concentration (*C*) in particles per 1/100 of a cubic foot (pcs/0.01cf) unit by using the equation in (2).

$$C = (1.1 \times R^3) + (3.8 \times R^2) + (520 \times R) + 0.62 \quad (2)$$

Because the air quality standard unit for PM2.5 is in µg/m^3, our program will transform the value *C* in pcs/0.01cf unit to C_S in µg/m^3 unit using the equation in (3).

$$C_S = C \times 3531.5 \times (1.65 \times 10^{12}) \times ((4/3)\pi(0.44 \times 10^{-6})^3 \quad (3)$$

Noted that:

- All particles are spherical, with a density of 1.65E12 µg/m^3.
- The radius of a particle in the PM2.5 channel is 0.44 µm.
- 0.01 ft^3 can be converted to m^3 by multiplying by 3531.5.

After we obtain the value of the PM2.5 concentration in µg/m^3 unit, the program will publish this value to MQTT Broker in the topic named *"sensor/PM2.5"* every 1 minute.

B. Node-RED (Subscriber)

Node-RED is a MQTT client that subscribes to MQTT Broker on the topic that the device publishes. We design the flow to manage and handle data from device's four sensors via 3 topics (*sensor/TEMPAndHUMID*, *sensor/COAndO3*, and *sensor/PM2.5*) in Node-RED.

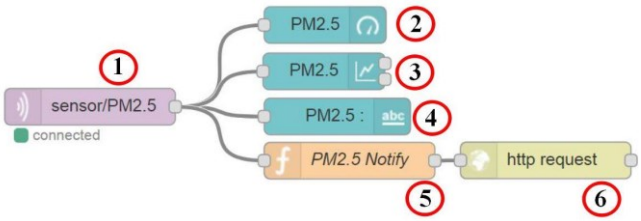

Fig. 3. The Flow Management for PM2.5 Data.

A flow design to handle the PM2.5 data shown in Fig. 3. First, a subscribe node (number 1) subscribes the topic *sensor/PM2.5*. This node will obtain the PM2.5 data from MQTT Broker whenever the device publishes the PM2.5 concentration data. Then, the PM2.5 data will pass to gauge node, chart node and text node in number 2, 3, and 4 respectively. These three nodes will be displayed as user interface on the air quality monitoring dashboard that is a responsive web application. In addition, the subscribe node passes the PM2.5 data to the *PM2.5 Notify* function node (number 5).

The algorithm in PM2.5 Notify function node indicated in the flowchart shown in Fig. 4. At the first time, a static variable *State* is set to Normal.

If the PM2.5 data is more than 50 µg/m^3 and *State* is Normal, then *State* is set to Alarm and a variable *Msg* is set to a warning message "Unhealthy, PM2.5 is more than 50 µg/m^3". Then, this function node sends the warning message to the *http request* node (number 6).

If the PM2.5 data is less than 35 µg/m^3 and *State* is Alarm, then *State* is set to Normal and a variable *Msg* is set to a notify message "Good, PM2.5 is less than 35 µg/m^3". Then, this function node sends the notify message to the *http request* node (number 6).

Otherwise, no message passes to the *http request* node in order to prevent the users to receive multiple warning or notify messages.

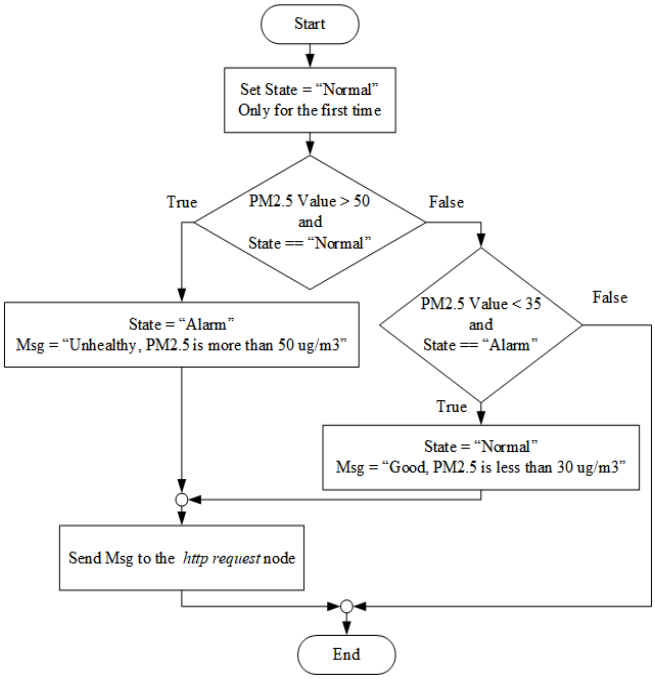

Fig. 4. The Algorithm of the PM2.5 Notify Function Node.

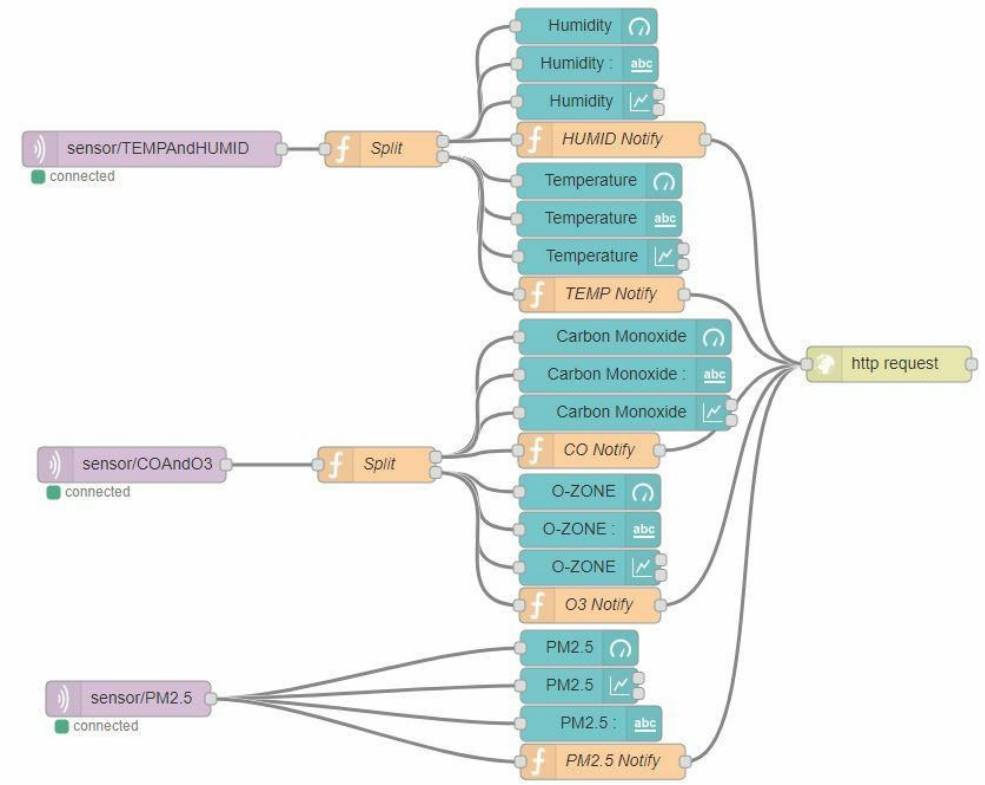

Fig. 5. The Overall Flow Management Design in Node-RED.

The *http request* node (number 6) is a node that connects to LINE Notify service via Web API. It will send a received message to LINE Notify service to notify user on LINE messenger application.

Fig. 5 shows the overall flow designed in Node-RED to handle data from three topics. All of these three flows are mostly the same. The flow of *sensor/TEMPAndHUMID* and *sensor/COAndO3* have one more addition node, a split node.

Because, the data of the topic *sensor/TEMPAndHUMID* contains temperature and humidity value, separated by comma. Also, the data of topic *sensor/COAndO3* contains CO-gas and Ozone-gas concentration, separated by comma. Thus, the split node is set to separate two data and feed them to the appropriated flow.

IV. Results

A. Air Quality Measurement Device

The real device is implemented as we designed shown in Fig. 6. The size of device is quite small as we compare to a Thai Baht coin. In addition, the total cost of the device is quite cheap. It is only around 2,000 Baht (62 USD).

B. Air Quality Monitoring Dashboard

The dashboard is a responsive web application provided by Node-RED. It displays three kinds of user interface from each sensor data (gauge, text, and chart). Moreover, because the dashboard is a responsive web application, so it can display perfectly on both mobile phone as shown in Fig. 7. and web browser as shown in Fig. 8.

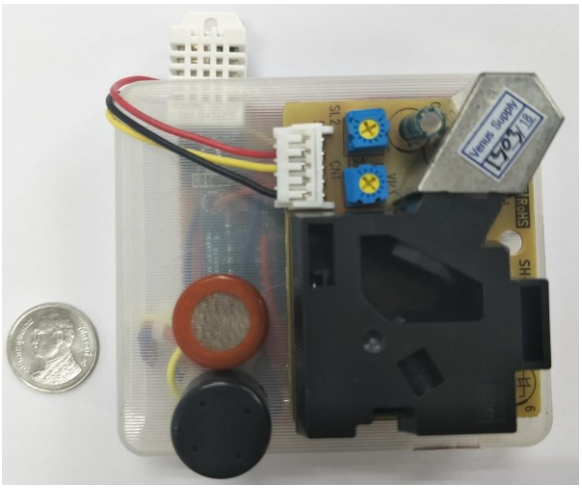

Fig. 6. The Complete Air Quality Measurement Device.

Fig. 7. The Air Quality Monitoring Dashboard on Mobile Phone.

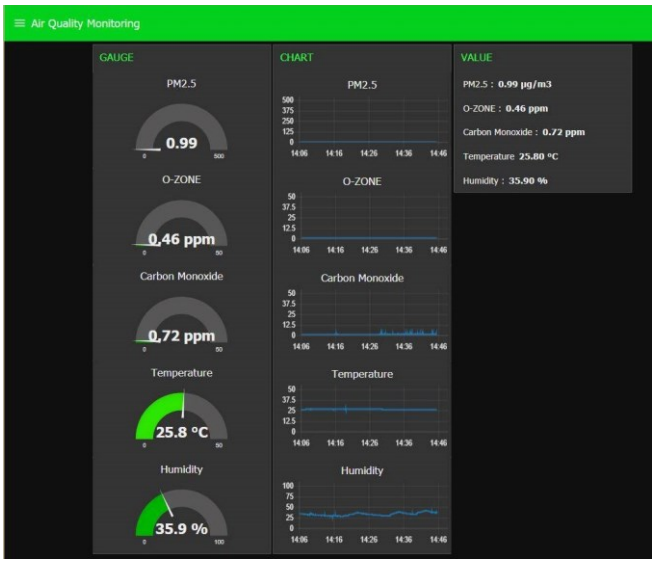

Fig. 8. The Air Quality Monitoring Dashboard on Web Browser.

C. *Line Notify*

We simulate the notification by heating the device. In our setup the temperature that exceeds 40 degree Celsius, we mark it as an unhealthy condition. For the temperature that is lower than 35 degree Celsius, we mark it as a good condition.

The Fig. 9. is a chart on the dashboard that is showing when we heat the device. It detects that the temperature is over 40 degree Celsius at 15:10. And the Fig. 10. shows that the temperature went down to below 35 degree Celsius at 15:12.

Thus, users who register to our LINE Notify service will receive the warning message at 15:10 and the notification message at 15:12 as shown in Fig. 11.

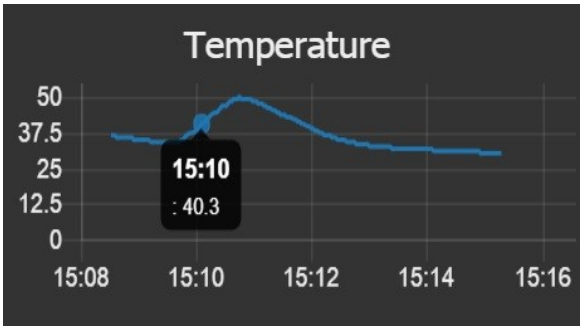

Fig. 9. The Chart when the temperature is over 40 °C at 15:10.

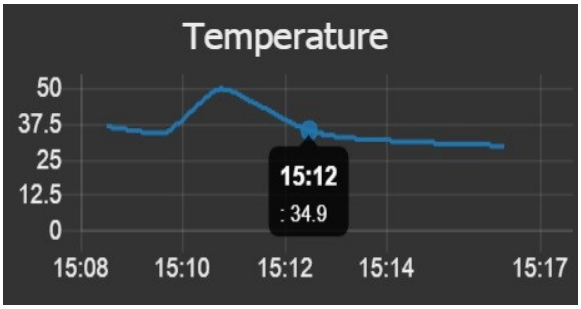

Fig. 10. The Chart when the Temperature is below 35 °C at 15:12.

Fig. 11. The Notification Message on LINE Messenger Application.

V. Conclusion

We presents a low-cost real-time air quality monitoring system. The air quality measurement device is built on ESP8266 NodeMCU board connects to 4 sensors: DHT22, MQ-7, MQ-131 and PPD42NJ to detect the temperature, humidity, carbon monoxide (CO) concentration, ozone-gas (O_3) concentration, and PM2.5 concentration. The air quality monitoring dashboard is implemented on Node-RED to receive and display air quality data. With Node-RED, we can design the flow to manage and handle air quality data, thus we can set the condition to notify users about the unhealthy environment through LINE Notify service. The device and Node-RED communicate via MQTT Broker in the publish and subscribe model.

However, the Air Quality Index (AQI) does not take and calculate the value from real-time data. It need to use the average value of each kind of gas concentration. Hence, the future works will be store the data into the database and calculate the unhealthy condition to notify users from it to make a proper air quality monitoring system.

References

[1] Worstpolluted.org, “The World’s Worst Pollution Problems: The Top Ten of The Toxic Twenty”, 2008. [Online]. Available: http://www.worstpolluted.org/reports/file/World's%20Worst%20Pollution%20Problems%202008.pdf. [Accessed: 19 May 2018].

[2] World Health Organization, “7 million premature deaths annually linked to air pollution”, 2014. [Online]. Available: http://www.who.int/mediacentre/news/releases/2014/air-pollution/en/. [Accessed: 19 May 2018].

[3] K. Ashton, “That “Internet of Things” Thing: In the Real World Things Matter More than Ideas”, 2019. RFID Journal. [Online]. Available: http://www.rfidjournal.com/articles/view?4986. [Accessed: 19 May 2018].

[4] A. Banks, R. Gupta, “MQTT version 3.1.1 Plus Errata 01, OASIS Standard Incorporating Approved Errata 01”, 2015, [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. [Accessed: 19 May 2018].

[5] Node-RED, [Online]. Available: http://www.nodered.org. [Accessed: 19 May 2018].

[6] LINE Notify, [Online]. Available: https://notify-bot.line.me/en/. [Accessed: 19 May 2018].

[7] LINE, [Online]. Available: https://line.me/en/. [Accessed: 19 May 2018].

[8] Eclipse Mosquitto, [Online]. Available: https://mosquitto.org/. [Accessed: 19 May 2018].

[9] Arduino Client for MQTT, [Online]. Available: https://pubsubclient.knolleary.net/. [Accessed: 19 May 2018]