Journal of Engineering and Technology

ISSN 2180-3811 eISSN 2289-814X https://jet.utem.edu.my/index

DEVELOPMENT OF AN IOT SCADA SYSTEM FOR A CHEMICAL MANUFACTURING PLANT

C. H. Liew[1] and T. F. Chay*[2]
[12]Faculty of Engineering and Technology,
Tunku Abdul Rahman University of Management and Technology
Kuala Lumpur, Malaysia.
corresponding: chaytf@tarc.edu.my

Article history:
Received Date:
24 October 2022
Revised Date:
26 June 2023
Accepted Date:
21 July 2023

Keywords:
SCADA, IoT,
OPC UA,
Industrial
Automation

Abstract— A traditional chemical plant in operation requires human workers to manually switch chemical hoses from different chemical tanks to a mixer tank every work shift following the production schedule. Due to manual works, the production system is exposed to human error where workers sometimes connecting wrong hoses to the mixer tank hence resulting incorrect chemical composition. It causes tremendous losses to the company. Therefore, an IoT SCADA (Supervisory Control and Data Acquisition) system is proposed and developed to solve this problem. The proposed architecture of the system consists of three layers which are device layer, control and monitoring layer and cloud layer. At the device layer, a sensor network is built using IO-Link Master with RFID tags for data collection from the production line. The IO-Link Master acts as

an OPC UA server to publish sensor data to its clients. At the control and monitoring layer, an open-source SCADA software is developed using C# on Microsoft .Net platform. The SCADA is configured as an OPC UA client to receive sensor data from the OPC UA server for process supervisory and control. At the cloud layer, the production data can be uploaded to the Azure MySQL database server via the SCADA software. All the functions of the proposed IoT SCADA system have been successfully tested. The proposed system is able to reduce manual involvement in the chemical plant which the processes can be controlled automatically and monitored remotely thus human errors can be reduced to the minimal level.

I. Introduction

A chemical factory located in Klang, Selangor, Malaysia is running their operations using traditional methods. In their mixing process plant, they require human workers to manually switch chemical hoses from different chemical tanks to a mixer tank every work shift according to their production schedule. The production system is highly exposed to human errors where the workers would sometimes connect wrong hoses to the mixer tank hence resulting incorrect chemical composition. The human errors cause tremendous losses to the company. To embark their journey of Industry 4.0 digital transformation, an IoT SCADA (Supervisory Control and Data Acquisition) is proposed and developed to solve their problem in the mixing process plant. This paper starts with a review of previous studies on the system architecture of open-source SCADA and IoT published in

recent five years. This follows by the hardware and software development of the proposed IoT SCADA system for the chemical plant. Lastly, it discusses the outcome of the system and the conclusion remarks of this study.

II. Literature Review

SCADA is a software and hardware integrated system that allows local and remote monitoring and control. The system offers real-time process control, data collection, data logging and processing, system security as well as process recipes and alarm management. SCADA system began in 1970s, the first generation was called Monolithic SCADA. Then, followed by the second generation, Distributed SCADA; and the third generation, Network SCADA. With rapid development in technology, it comes to the fourth generation which is known as the Internet of Things (IoT) SCADA [1, 2].

IoT is an advanced IT technology that provides communication for a variety of devices including sensors, actuators, PLCs, electronic devices, as well as the systems network configuration and connectivity that allows data exchange between these devices. While SCADA provides a platform for human-to-machine interaction in process or machine control, IoT is commonly used for machine-to-machine data exchange rather than presenting information directly to human [3].

Although SCADA and IoT involve sensors and data acquisition in different ways but they share the common goal which is for optimization use and greater control over some devices or a process. The integration of SCADA and IoT is accelerated by the idea of smart grid in energy industry. Their integration allows direct interaction between physical field devices and computer-based monitoring system. This would increase the efficiency and scalability of a control system while reducing human errors and costs. Furthermore, the IoT-based SCADA system that encompasses web or cloud services is able to provide a more robust remote monitoring and control system [4]. The dynamics of IoT-based SCADA cloud computing system

would alter the landscape of industrial automation and moving the industries towards Industry 4.0.

This paper reviewed several past studies on architecture of open-source SCADA and IoT systems published in recent five years.

Aghenta and Iqbal [5] designed an IoT based SCADA system using ESP32 microcontroller to control a Solar Photovoltaic system. The data acquired from the sensor would be transferred from the ESP32 to the Thingsboard IoT server. MQTT is used as the communication protocol to transfer sensor data via a local Wi-Fi network where the ESP32 acts as the MQTT Client while the ThingsBoard server node as the MQTT Broker. For data integrity and security, the ThingsBoard IoT server is locally deployed with a PostgreSQL database on a Raspberry Pi single-board computer and hosted locally on MUN Network. The dashboard is created on the ThingsBoard for data trending and history visualization.

Kao et al. [6] developed an IoT-based SCADA for remote monitoring and control a variable frequency drive (VFD). The system has four different layers namely monitor layer, server layer, cloud layer, and client layer. The monitor layer has a sensor, a VFD and a Wi-Fi router. A computer is at the server layer while MySQL database is connected to the cloud layer. Remote devices are connected to the system at the client layer. At the monitor layer, voltage and frequency data is sent from the sensor to the VFD then published to the database through a Wireless Sensor Network (WSN). An open-source SCADA system is created using Asynchronous JavaScript and XML (AJAX) and Responsive Web Design (RWD) technologies.

In their paper, Li et al. [7] proposed a cloud-based SCADA system by integrating JustIoT framework with SCADA. The system was used in smart house monitoring. The Modbus TCP and OPC client were used to connect the JustIoT framework to the traditional SCADA for cloud capabilities. The controllers used are Raspberry Pi3 and Arduino Uno microcontroller. Data such as humidity, temperature, and

brightness would be transferred from the controllers to Firebase cloud database via MQTT. PC and mobile devices are able to monitor the real-time data and stored data in the Firebase.

Merchán et al. [8] developed an open-source SCADA system written in Python. Their system is divided into three layers namely supervision layer, control layer and device layer. Field devices such as sensors and actuators at the device layer are connected to two units of Programmable Logic Controllers (PLCs) at the control layer. An OPC UA server and MySQL database is hosted by a Raspberry Pi3 at the supervision layer. The SCADA and PLC are connected to the database via OPC UA. The system provides fault-tolerance features with the implementation of Active Fault-Tolerance Control (AFTC). However, this open-source SCADA was found less reliable due to too many components connected to system and improvement is needed.

Osaretin et al. [9] presented an open-source IoT-based SCADA system based on Node-RED for remote monitoring and control of motors and sensors in oil and gas facilities. This system consists of a host computer with Node-RED development platform. The host computer is connected to a unit of Arduino Uno and a unit of Arduino Mega. The Arduino Uno gathers and transmits data from various sensors to the host computer through via serial connection. A web-based graphical user interface (GUI) is developed with Node-RED for sensor data visualisation in the form of gauges and trending charts. Nginx is used to perform basic access authentication to control client access from the Internet.

Zare & Iqbal [10] presented a home automation system with SCADA integrating with ESP32 microcontroller. MQTT is used as the communication protocol in a local Wi-Fi network and open for access by mobile devices where a Raspberry Pi SBC acting as the MQTT broker. As a MQTT client, the ESP32 publishes the house conditions to the MQTT broker. The data of the house condition is then stored in the SQLite Server in the SBC. A MQTT client web-based dashboard is created to

visualize the data using Node-Red by subscribing to the data published by the SBC.

In summary, all the IoT SCADA in the reviewed works have several common features. Their system architectures are designed in multi layers where device layer contains field devices such as sensors and actuator; control layer contains controllers such as computers, microcontrollers and PLCs; supervision layer which the SCADA monitors the automation system with a database for data logging; and a cloud layer for remote clients to access to the data. These features are summarised and implemented in the proposed IoT SCADA solution to solve the problem of the chemical plant discussed in this paper.

III. System Design

A. System Architecture

Figure 1 shows the system architecture of the IoT SCADA proposed to the chemical plant. It has three layers namely the device layer, control and monitoring layer, and the cloud layer. The device layer has an IO-Link Master connecting to RFID tags and can be used to connect to other field sensors and actuators should the system is to be further expanded. Each RFID tag will be attached to the hose connecting to a chemical tank. The RFID provides feedback to the SCADA via the IO-Link Master in a local Wi-Fi network to ensure that the hoses are connected to the correct chemical tanks according to the recipe setting in the SCADA following production schedule of the chemical plant. If wrong hose is connected, then the process cannot be started. This can greatly prevent human error in production system. PortVision DX is used to configure the IO-Link Master and the RFID Read/Write heads. The address of the IO-Link Master and its role as the OPC UA server are set in the PortVision DX.

In the control and monitoring layer, the open-source SCADA software is written in C# using Visual Studio on Microsoft .Net platform. The SCADA has a local SQL database. Written as an OPC UA client, the SCADA obtain the RFID data from the IO-Link Master in a local Wi-Fi network. In the cloud layer, Azure MySQL database is employed to the solution. The data received by the

SCADA is store in the local SQL database for the company internal use and at the same, the data can be uploaded to the cloud, which is the Azure MySQL Database. MySQL Workbench is used to visually design, model, generate, and manage databases.

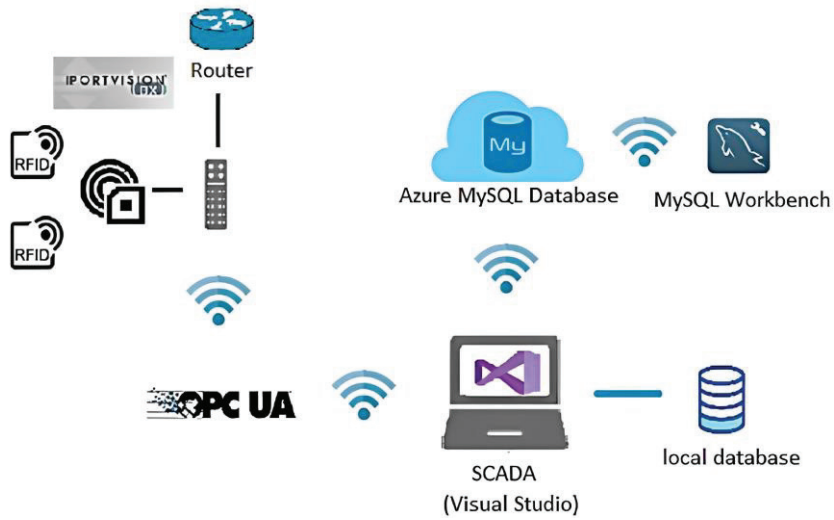


Figure 1: System Architecture of the proposed IoT SCADA system

B. IO-Link Master

Figure 2 shows the IO-Link Master from Pepperl+Fuchs which is powered by a 24V 5A DC power supply and a sample of RFID tag connected to one of its ports. The IO-Link Master is connected to a Wi-Fi router and its configuration as the OPC UA server is done in a computer running the PortVision DX software.

C. IO-Link Master

Figure 2 shows the IO-Link Master from Pepperl+Fuchs which is powered by a 24V 5A DC power supply and a sample of RFID tag connected to one of its ports. The IO-Link Master is connected to a Wi-Fi router and its configuration as the OPC UA server is done in a computer running the PortVision DX software.

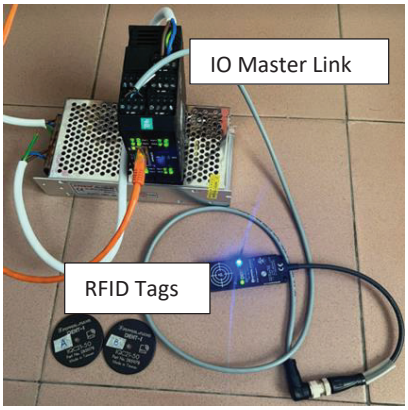


Figure 2: Demonstration of the IO Master Link and RFID tags

D. Software Design

The Graphic User Interface (GUI) of the SCADA main page designed in C# WinForms is shown in Figure 3. The GUI is designed according to the chemical plant's mixer process setup where there are four chemical tanks for four types of chemicals which identified as A, B, C and D connecting to a mixer tank. It has an operation start-stop button and an emergency stop button. A menu strip with seven items and a few labels labelling the components shown in the form.

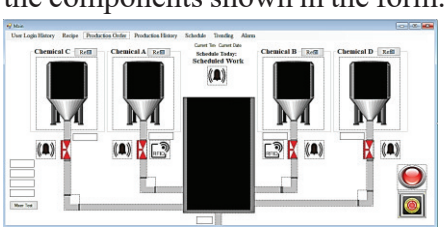


Figure 3: SCADA GUI design

As mentioned, the IO-Link Master acts as the OPC UA server and the SCADA system acts as the OPC UA client. Once the client is ready, it connects to the server via the configured IP Address of the server to start the communication. Figure 4 shows the excerpt of the SCADA programme in C# running as an OPC UA client connecting to the OPC UA server.

```
var client = new OpcClient("opc.tcp://192.168.0.250");
client.Connect();
var RTag = client.ReadNode(readtag);
Boolean Readnodevalue = RTag.As<Boolean>();
if (Readnodevalue == true)
{
    var databyte = client.ReadNode("ns=1;s=IOLM/Port 2/Attached Device/PDI Data Byte Array");
    byte[] byteValue = databyte.As<byte[]>();
    //var byteCount = client.ReadNode(tagbyteCount);
    //int intValue = byteCount.As<int>();
    byte[] temp = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
    int b = 13;
    for (int i = 0; i < 10; i++)
    {
        temp[i] = byteValue[b];
        b--;
    }
    label1.Text = System.Text.Encoding.ASCII.GetString(temp);
```

Figure 4: An excerpt of the SCADA programme in C# as an OPC UA client connecting to the OPC UA server

At the device layer, once data is received from the RFID tags, the SCADA system will know which tank is connected. Input tag of volume in-flow from each chemical tank is also created to monitor how much volume of the chemical is discharged. The data will be loaded into the local database, and it will be used for trending display. Figure 5 shows the excerpt of the C# code of the SCADA loading data into local database.

```
private void updateVolume()
{
    Con.Open();
    String query1 = "Update Volume set TankA=@x, TankB=@y, MixerTank=@z, TankC=@c, TankD=@d";
    SqlCommand cmd = new SqlCommand();
    cmd.CommandText = query1;
    cmd.Connection = Con;
    cmd.Parameters.AddWithValue("@x", x);
    cmd.Parameters.AddWithValue("@y", y);
    cmd.Parameters.AddWithValue("@z", z);
    cmd.Parameters.AddWithValue("@c", c);
    cmd.Parameters.AddWithValue("@d", d);
    cmd.ExecuteNonQuery();
    Con.Close();
}
```

Figure 5: An excerpt of SCADA programme in C# loading chemical tank data to local database

The SCADA system will also upload the production data to the cloud, Azure MySQL database. Figure 6 shows the excerpt of the C# code of the SCADA to upload production data to Azure MySQL database.

```
private void updatecloudV()
{
    try
    {
        string Query = "update scada.`dbo.volume` set `TankA`='" + x + "',`TankB`='" + y + "'," +
            "`MixerTank`='" + z + "',`TankC`='" + c + "', `TankD`='" + d + "'";
        MySqlCommand MyCommand2 = new MySqlCommand(Query, Conn);
        MySqlDataReader MyReader2;
        Conn.Open();
        MyReader2 = MyCommand2.ExecuteReader();
        while (MyReader2.Read())
        {
            Console.WriteLine("uploaded");
        }
        Conn.Close();//Connection closed here
    }
    catch (Exception ex)
    {
        Console.WriteLine("error volume");
    }
}
```

Figure 6: An excerpt of SCADA programme in C# loading data to Azure MySQL database

The SCADA is also written to equip with the following features namely multi-level user login, recipe, scheduling, production control, production history, and alarm.

IV. Results and Discussion

A. Results and Discussion

Figure 7 shows the main page of the SCADA system when in operation.

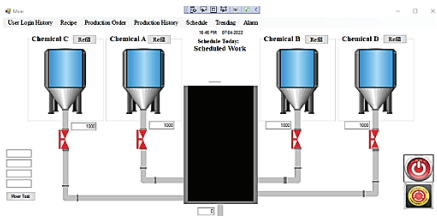
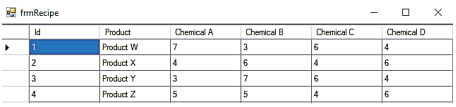


Figure 7: SCADA system main page in operation

There are two levels of user login namely executive and operator. Executive has a higher accessibility permission to access all the features developed in this system; while operator can only access to production start-stop control, production status monitoring, data trending, production schedule monitoring and alarm history.

The production of the chemical plant will be running automatically according to the schedule and recipe set by the executive. The executive has the access to set the recipe of chemical composition for different product variance and to plan the schedule of the production line. When the system is running, it will check the production order and the schedule entered by the executive. For different products to be produced, the system will follow the recipe set by the executive. Figure 8

shows the screenshot of the recipe menu.



	Id	Product	Chemical A	Chemical B	Chemical C	Chemical D
▸	1	Product W	7	3	6	4
	2	Product X	4	6	4	6
	3	Product Y	3	7	6	4
	4	Product Z	5	5	4	6

Figure 8: Screenshot of the recipe menu

Control of actuators such as solenoid valve will come in handy with the use of IO-Link master. The solenoid valve of the chemical tank will be turned on by the SCADA after the system has confirmed the right chemical tank is connected via the status of RFID tags. Figure 9 shows an example of SCADA acting as the OPC UA client receiving data from a RFID tag via the IO-Link Master. The label in red box is the converted RFID tag data into text.

The correct composition of the chemical will flow from the chemical tanks to a mixer tank. After that, mixing of the chemical product commenced. Figure 10 shows the screenshot of the SCADA displaying the production status.

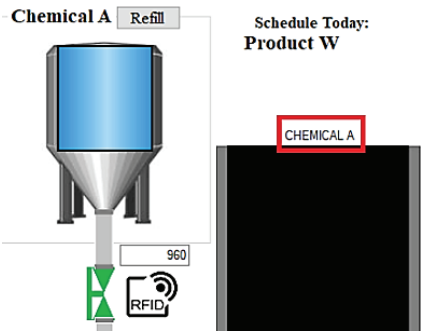


Figure 9: SCADA as OPC UA client responding to the OPC UA server

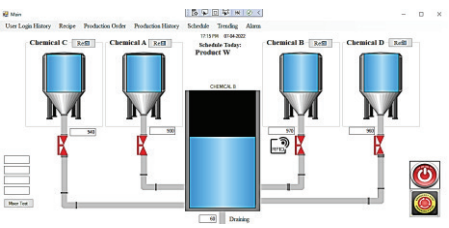


Figure 10: Product mixing

In the SCADA, there are charts developed to visualize real-time trending data of chemical volume for each tank. Figure 11 shows an example of data trending of the SCADA system.

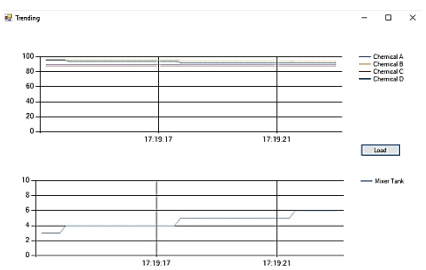
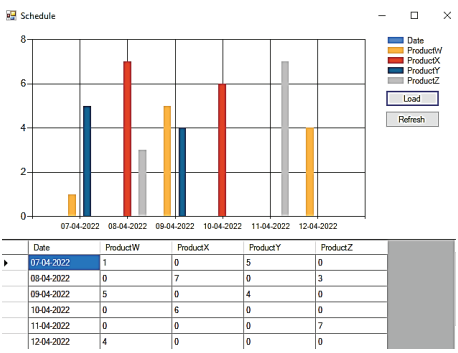


Figure 11: Data trending in the SCADA

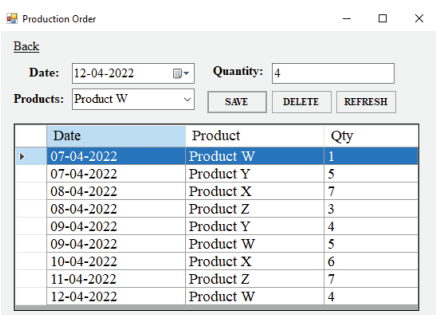
The production order is designed in First-in First-Out

(FIFO) manner. The SCADA system will run the production based on the schedule planned by the executive. Figure 12 shows the production schedule with recipe of the SCADA while Figure 13 shows the details of the schedule with date and quantity required to produce on the day. When one particular production batch is done, the system will check the production order of the day again, and its required quantity of the next batch without human involvement.



Date	ProductW	ProductX	ProductY	ProductZ
07-04-2022	1	0	5	0
08-04-2022	0	7	0	3
09-04-2022	5	0	4	0
10-04-2022	0	6	0	0
11-04-2022	0	0	0	7
12-04-2022	4	0	0	0

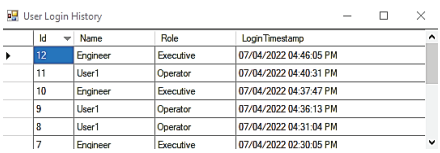
Figure 12: Production Schedule



Date	Product	Qty
07-04-2022	Product W	1
07-04-2022	Product Y	5
08-04-2022	Product X	7
08-04-2022	Product Z	3
09-04-2022	Product Y	4
09-04-2022	Product W	5
10-04-2022	Product X	6
11-04-2022	Product Z	7
12-04-2022	Product W	4

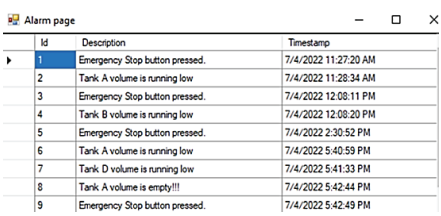
Figure 13: Details of the production Order

The SCADA system will log user login history, alarm history and the production history as shown in Figure 14 to 16 in the local database. Operator can only check the alarm history while executive can view the history of all activities in the system. The data history features help the company to trace the progress or the current status of the production line.



Id	Name	Role	LoginTimestamp
12	Engineer	Executive	07/04/2022 04:46:05 PM
11	User1	Operator	07/04/2022 04:40:31 PM
10	Engineer	Executive	07/04/2022 04:37:47 PM
9	User1	Operator	07/04/2022 04:36:13 PM
8	User1	Operator	07/04/2022 04:31:04 PM
7	Engineer	Executive	07/04/2022 02:30:05 PM

Figure 14: Screenshot of user login history



Id	Description	Timestamp
1	Emergency Stop button pressed.	7/4/2022 11:27:20 AM
2	Tank A volume is running low	7/4/2022 11:28:34 AM
3	Emergency Stop button pressed.	7/4/2022 12:08:11 PM
4	Tank B volume is running low	7/4/2022 12:08:20 PM
5	Emergency Stop button pressed.	7/4/2022 2:30:52 PM
6	Tank A volume is running low	7/4/2022 5:40:59 PM
7	Tank D volume is running low	7/4/2022 5:41:33 PM
8	Tank A volume is empty!!!	7/4/2022 5:42:44 PM
9	Emergency Stop button pressed.	7/4/2022 5:42:49 PM

Figure 15: Screenshot of alarm history

MySQL Workbench is used to connect to the Azure MySQL database server. It is also used to visualise the data stored in the cloud. Figure 17 shows the screenshot of MySQL Workbench and the SCADA interface which running in the same computer, at

the same time. With MySQL Workbench, the data in the Azure MySQL can be monitored anytime everywhere.

Production History

	Id	Product	Timestamp
▶	1	Product W	7/4/2022 11:31:12 AM
	2	Product X	7/4/2022 11:32:54 AM
	3	Product W	7/4/2022 12:20:38 PM
	4	Product X	7/4/2022 2:18:39 PM
	5	Product W	7/4/2022 4:59:53 PM
	6	Product W	7/4/2022 5:15:37 PM
	7	Product Y	7/4/2022 5:20:06 PM
	8	Product Y	7/4/2022 5:21:37 PM
	9	Product Y	7/4/2022 5:22:21 PM
	10	Product Y	7/4/2022 5:23:01 PM

Figure 16: Screenshot of production history

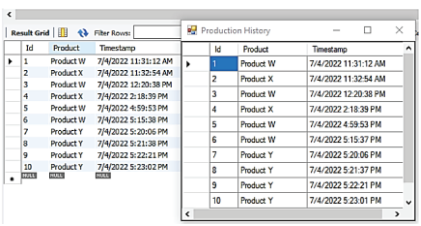


Figure 17: Interface of production history in MySQL Workbench and local database

B. Comparison with Previous Work

The proposed IoT SCADA system in this paper which is written in Microsoft .Net platform is compared with the system developed by Merchán et al. [8], "Open-Source SCADA System for Advanced Monitoring of Industrial Processes" which their open-source SCADA system was written in Python. Although the two open-source SCADA systems having some similarities in terms of SCADA features such as user level access and management, data logging and trending as well as alarm system, the work proposed by this paper include recipe function as well for the manufacturer to configure their chemical composition for different product variance. The recipe feature is integrated with the production schedule which provides the convenience of automatic control and remote monitoring of their production activities. In comparison of database system, their work is based on MySQL database while the work proposed in this paper has further to put the SQL database to the Microsoft Azure server which has advantages on automatic scheduling, better performance and widely accessible.

V. Conclusion

In conclusion, an IoT SCADA system is successfully developed. The architecture of the system

consists of three layers which are device layer, control and monitoring layer and cloud layer. At the device layer, a sensor network is established using IO-Link Master with RFID tags to collect data from the production line. The IO-Link Master is configured as an OPC UA server and publishing sensor data to its client. At the control and monitoring layer, the SCADA is designed with the following features: graphic user interface (for process monitor and control), data logging, scheduling, recipe, trending, and alarms. The SCADA system also acts as an OPC UA client to receive data from the OPC UA server. At the cloud layer, the data in the SCADA system can be uploaded to the Azure MySQL database server. This IoT SCADA system is proposed to the management of the chemical plant. A pilot project will be carried out at one of their production lines as part of their Industry 4.0 transformation. With this system, the chemical plant processes can be carried out automatically and remotely without the intervention of workers. Therefore, human errors can be reduced to the minimal level.

VI. References

[1] L. O. Aghenta and M. T. Igbal, "Low-Cost, Open Source IoT-Based SCADA System Design Using Thinger.IO and ESP32 Thing" Electronics, vol. 8, no. 8, p. 822, 2019.

[2] S. Phuyal, D. Bista, J. Izykowski and R. Bista, "Design and Implementation of Cost Efficient SCADA System for Industrial Automation" International Journal of Engineering and Manufacturing, vol. 10, no. 2, pp. 15-28, 2020.

[3] A. Shazad, Y. Kim and A. Elgamoudi, "Secure IoT Platform for Industrial Control Systems" in International Conference on Platform Technology and Service (PlatCon), 2017, IEEE. DOI: 10.1109/PlatCon.2017.7883726

[4] L. O. Aghenta and M. T. Iqbal, "Development of an IoT Based Open Source SCADA System for PV System Monitoring" in IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), 2019, IEEE. DOI: 10.1109/CCECE.2019.8861827

[5] L. O. Aghenta and M. T. Igbal, "Design and implementation of a low-cost, open source IoT-based SCADA system using ESP32 with OLED, Thingsboard and MQTT protocol" in AIMS Electronics and Electrical Engineering, vol. 4, no. 1, pp. 57-86, 2019.

[6] K. C. Kao, W. H. Chieng and S. L. Jeng, "Design and development of an IoT-based web application for an intelligent remote SCADA system" in IOP Conference Series: Materials Science and Engineering, 2018, IOP Publishing. DOI: 10.1088/1757-899X/323/1/012025

[7] W. J. Li, J. C. Wang, Y. S. Lin and S. C. Tung, "Cloud supervisory control system based on JustIoT" in 2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE), 2019, IEEE. DOI: 10.1109/SMILE.2018.8353974

[8] D. F. Merchán, J. A. Peralta, A. Vazquez-Rodas, L. I. Minchala and D. Astudillo-Salinas, "Open source SCADA system for advanced monitoring of industrial processes" in 2017 International Conference on Information Systems and Computer Science (INCISCOS), 2017, IEEE. DOI: 10.1109/INCISCOS.2017.9

[9] C. A. Osaretin, M. Zamanlou, M. T. Iqbal and S. Butt, "Open source IoT-based SCADA system for remote oil facilities using Node-RED and Arduino microcontrollers" in 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2020, IEEE. DOI: 10.1109/IEMCON51383.2020.9284826

[10] A. Zare and M. T. Iqbal, "Low-cost ESP32, Raspberry Pi, Node-RED, and MQTT protocol based SCADA system" in 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020, IEEE.DOI:10.1109/IEMTRONICS51293.2020.921641