THE SPIRIT OF SOCIETY JOURNAL
International Journal of Society Development and Engagement

ISSN : 2594-4777 (Online) – ISSN : 2597-4742 (Print)

Development of SCADA Dynamic Application Design

Joesianto Eko Poetro, M. Basuki Rahmat*, Agus Khumaidi, Hananta A, Bhakti
Department of Marine Electrical Engineering, Shipbuilding Institute of Polytechnic Surabaya, Indonesia
mbasuki.rahmat@ppns.ac.id

Abstract: The COVID-19 pandemic has changed major habits in learning patterns. Before the pandemic, almost all learning activities took place face-to-face. The pandemic situation brings problem especially to a vocational education where competence or expertise is an absolute achievement that must be achieved. Learning patterns where practical learning reaches 60% of students must obtain practical learning directly. During the pandemic, direct learning activities cannot be carried out in full. This will have an impact on student competence. Solution is needed to solve it. One of them is how students can practice virtually. This article does not discuss the extent of competence achieved by a student through online practice. The focus of this article is to discuss how practical devices can be controlled remotely by designing a virtual system on the platform and running it in real time or known as dynamic application. And technicians or PLP will still supervise the operation of equipment in the laboratory. The device will be built using the MODBUS communication protocol.

Keywords: Dynamic Applications, Modbus, Communication, Competence

INTRODUCTION

Distance Learning is becoming a new format as the COVID-19 pandemic escalates. The Community Activity Restriction Policy (PPKM) which requires limited activities greatly affects learning patterns. In Polytechnic learning or Vocational education where the composition of practical learning is more than the composition of theoretical learning, it becomes very disturbed. By not carrying out practical learning activities, the competencies that are the main characteristics of vocational education are lost or reduced. Although there is some practical learning done using assistive software to simulate. However, many practical learning activities cannot be replaced by simulations.

Along with the development of information technology and automation, in the industrial era 4.0, most industrial equipment can be monitored and controlled remotely. This can also be applied to the world of education and practical learning from a distance. Practical equipment in the laboratory can be connected to devices that can be monitored and controlled remotely. Students can do the setting process remotely. One of the communication protocols used to communicate between devices is the MODBUS protocol.

Modbus is the first data exchange communication protocol published by Modicon in 1979 (Anonym, 2006). Modbus is an application located at the top layer (7th layer) of the Open Systems Interconnection (OSI) communication architecture model. For data exchange between computers acting as masters and sensors acting as slaves, Modbus requires a physical hardware interface. One of those physical devices is a network card called Ethernet (Anonym, 2012). In order for data exchange with modbus to be carried out via an ethernet card, modbus requires a network protocol. One such network protocol is TCP/IP. Figure 1 shows the MODBUS Conceptual Architecture. The command request representation format from the master is broken down into groups of frames as shown in Table 1 (Guarese, et al., 2014).

ISSN : 2594-4777 (Online) – ISSN : 2597-4742 (Print)

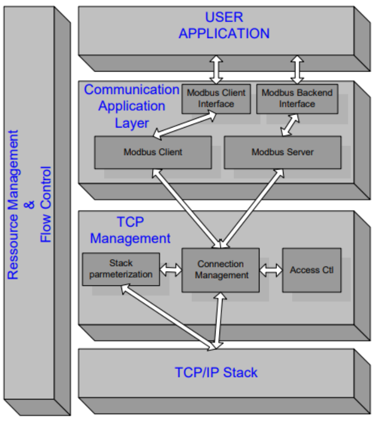


Figure 1. MODBUS Messaging Service Conceptual Architecture

Format the master command shown in Table 1. which consists of 5 is 8 bytes long. The first frame contains the slave address code. The slave address code is unique, for example each sensor connected as a slave has a different address. Code size the address is 1 byte, so the number of sensors that can be connected to the protocol.

Table 1. Format Command Request from Master

Kode Alamat Slave	Kode Fungsi	Register	Panjang Data	CRC_16
1 byte	1 byte	2 byte	2 byte	2 byte

SCADA Dynamic Application Design

Advances in information technology have been widely implemented in the industrial world. The use of automation control systems in industry aims to help and improve the effectiveness of human work. Today we are in the middle of the Industry 4.0 period where the number 4.0 indicates the Industrial Revolution which has entered the 4th wave. The characteristics of the model from Industry 4.0 are a combination of several of the latest technological developments which include information and communication technology, communication networks, big data and cloud computing as well as equipment developed to facilitate human interaction with computers (Anonym, 2006). All aspects of industrial computing can communicate with each other by utilizing communication protocols, one of which is Modbus.

Communication Protocol is a language to communicate with each other between hardware, and software. Modbus is a two-serial communication protocol method, where communication can occur with the modbus master and modbus slave systems (Anonym, 2012). Modbus can be transmitted via serial communication (Modbus RT), via Ethernet communication (Modbus TCP/IP), via ASCII (Modbus ASCII) and modbus plus. Modbus RTU is also equipped with a cyclic redundary error (CRC) mechanism that functions to reduce errors and ensure data reliability. In practice, modbus supports many modems and gateways to work (Capocci, et al., 2017). Figure 2, shown MODBUS RTU Communication Protocol.

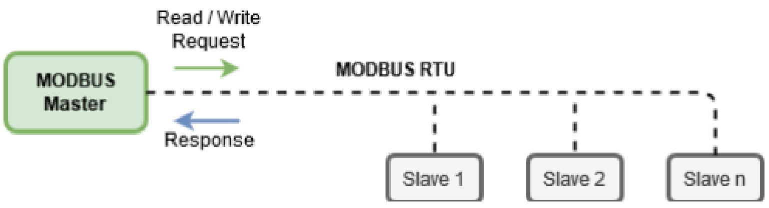


Figure 2. MODBUS RTU Communication Protocol

ISSN : 2594-4777 (Online) – ISSN : 2597-4742 (Print)

Figure 2, shows the communication of the MODBUS RTUI Protocol, which is used in industry. Remote MODBUS.

Terminal Unit (RTU) and ASCII are different types of protocols used for serial communication to create Master-Slave or Server-Client communication between smart devices (Guarese, et al., 2014). It is a widely accepted protocol due to its ease of use and reliability. Remote terminal unit (RTU) is connected to sensors and actuators to interact with physical systems.

The industrial control system consists of a master terminal unit (MTU) connected to a remote terminal unit (RTU) through communication channels (Morris, et al., 2012). MODBUS RTU protocols commonly used for Building Management Systems (BMS) and Industrial Automation Systems (IAS). The wide acceptance of this application is largely due to Ease of use MODBUS RTU.

MODBUS remote terminal unit message is simple 16-bit Cyclic-Redundant Checksum (CRC). 16-bit base The MODBUS RTU register structure can be used to package floating point, tables, ASCII text, queues and more unrelated the data (Nurpatmi, 2010). Modbus RTU communication protocol associated with half-duplex communication mode. Main frame sends a command signal to the terminal device according to the different slave addresses. After appropriate operation, terminal equipment sends an answering signal to mainframes. In Remote Terminal Unit mode, the button messages are transmitted in a continuous stream format. Every 8-bit (1 byte) framed by 1 Start bit, 8 Data bits, 0 or 1 Parity bits, 1 or 2 Stop bits. RS-485 is known as the most common serial cable communication protocols in industrial automation. RS-485 interface capable of supporting for a maximum distance of 1200 meters with multiple devices (up to 32 devices) on the same bus (Chandra, et al., 2016). Table 2 describes common wired communications protocols used in industry.

Table 2. Common wired Communication Protocols

Characteristics	**Protocol**		
	RS-232	**RS-485**	**GB-Ethernet**
Comms mode	Full-Duplex	Full/Half Duplex	Full/Half Duplex
Max. Distance	15 – 20 m	1200 m	100 m
Max. Transmission	20 Kb.s^{-1}	20 Mb.s^{-1}	1 Gb.s^{-1}
Typical Logic Level	± 5 to ± 15 V	± 1.5 to ± 5 V	± 0.5 to ± 2 V

METHODOLOGY

The method of the development of this system consisted of two part. The first is the hardware section and the second is the software section. The hardware will connect the devices in the laboratory.

Research Flow

The initial system design is divided into two part. The first part is software design. The second part is hardware design. The software and hardware design can be shown in Figure 2.

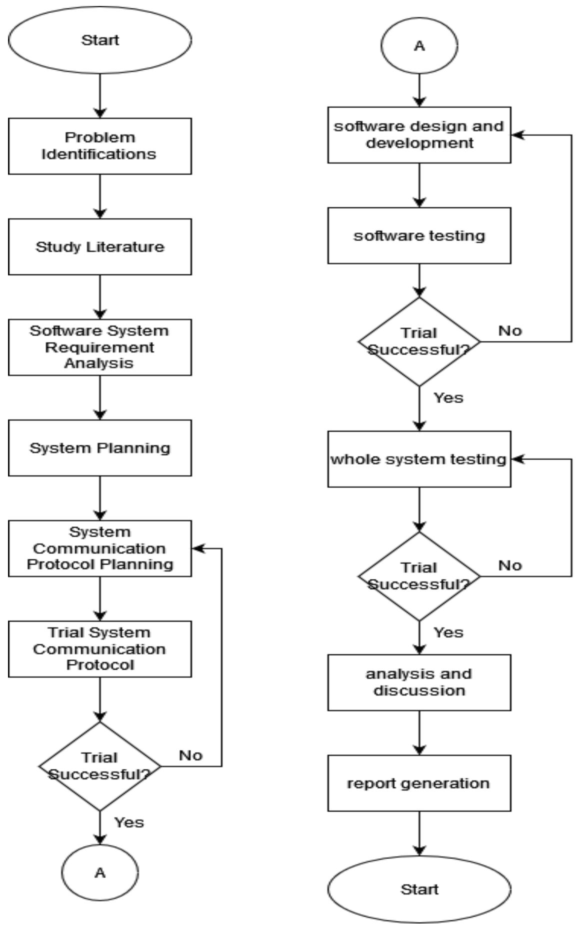


Figure 3. Initial system design.

Hardware Design

In the system concept, it can be explained that there is a block concept to design and create the system. In the system block concept, it is explained that there are inputs, processes and outputs. For this system, the input entered is in the form of sensor reading data. Then the data is processed on a microcontroller equipped with a framework. The result of the process is the output of the system in the form of data sent to the server or MQTT broker. Based on the concept of this system, a block system is described which can be seen in Figure 4.

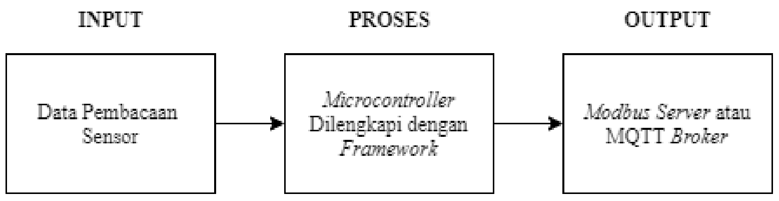

INPUT
PROSES
OUTPUT
Data Pembacaan Sensor
Microcontroller Dilengkapi dengan Framework
Modbus Server atau MQTT Broker

Figure 4. System Concept

Figure 3 describes the system diagram designed in accordance with the research flow. The system is designed using sensor reading data as system input, then the input data is sent to the microcontroller and the data will be processed and then the data will be sent to the MQTT server or broker. The working system of this tool starts with initializing all systems, namely turning on the system and preparing system requirements such as connecting sensors as system inputs using an RS485 serial cable to the screw terminal connected to the RS485 to TTL Serial module, connecting sensors as system inputs with the SN65HVD230 CAN bus transceiver module and Connect the

Ethernet ENC module to a PC/laptop using an RJ45 cable and connect the built in ESP32 Wi-Fi with an available access point as an internet connection. After system initialization, the sensor will send data to a microcontroller via the Modbus RTU protocol and the CAN bus protocol as input data.

The input data will be processed by a microcontroller equipped with a framework that has been prepared. Before the data is processed by the microcontroller, the microcontroller has been programmed according to the data processing needs and the communication protocol needs to be used. The process of preparing the program is made easier by the framework created in this research. The processed data will be sent to a server using the Modbus TCP/IP protocol and sent to the MQTT broker using the MQTT protocol. If the data transmission process is not successful, then return to the data processing stage by the microcontroller. If the data transmission is successful, the output of this tool is in the form of data that has been successfully sent to the server or MQTT broker so that online data monitoring can be carried out. The Hardware concept shown in figure 5.

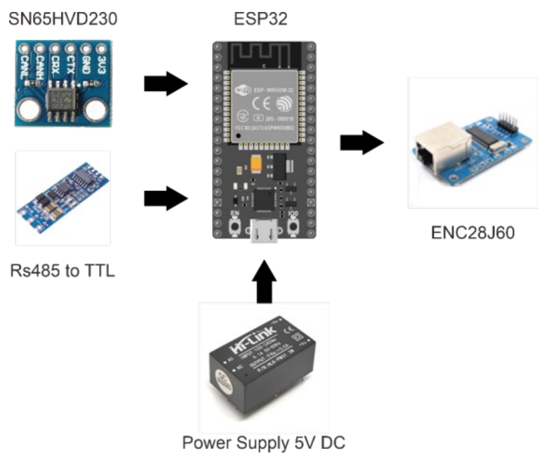


Figure 5. Hardware Concept

There is a 5V Hi-Link Power Supply module which functions as a conversion of a 220V AC voltage source into a 5V DC voltage which is used to supply voltage for this device. Then, in the first process using a Serial RS485 to TTL module which is useful for serial data conversion from sensors that will be sent to a microcontroller using the Modbus RTU protocol and there is also an SN65HVD230 CAN bus transceiver module which is useful for data transactions using the CAN interface. The main working system of this tool is on an ESP32 microcontroller because the ESP32 input data will be processed and controlled with a programming algorithm that is equipped with a framework. Then there is an Ethernet ENC module that functions as a Local Area Network (LAN) port for data transmission media to the server using the Modbus TCP/IP protocol. The electronics hardware design is designed by reading the RS485 to TTL module and the SN65HVD230 CAN bus transceiver as input data, as well as the use of a microcontroller and ENC28J60 Ethernet module as an output data controller. Figure 6 shows the hardware design.

ISSN : 2594-4777 (Online) – ISSN : 2597-4742 (Print)

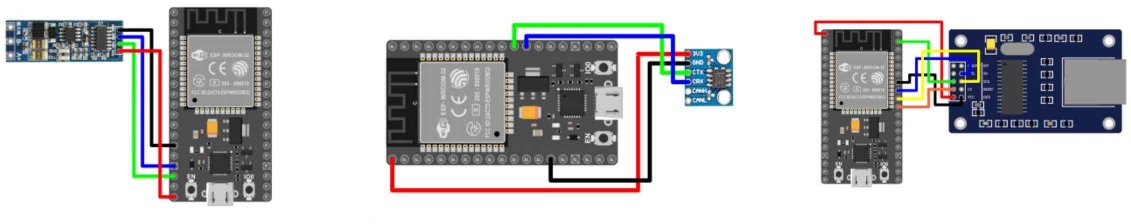

(a) (b) (c)

Figure 6. Hardware Design (a).RS485 to TTL. (b) Modul SN65HVD230 CAN Bus Transceiver. (c). Modul Ethernet ENC28J60

Software Design

System requirements analysis is a step to identify and understand the system requirements to be used. The system is divided into three stages, namely the input or input stage, the stage of sorting data or parsing data, and the last one presenting the data intuitively using charts, gauges and so on and sending data back to control the data.

The first stage is that data from the Modbus Gateway module will be sent using the Ethernet port using the Modbus TCP/IP communication protocol where the application will identify and connect the Module with the application using the specified IP address. After connecting, then the input data or data packets will be read and parsed according to the type of packet where each data packet consists of address, function code, data, and error check. Data from data packets will be stored and displayed for further id tagging to be associated with id tag gauges, charts, and so on. In addition to reading data (read coil and register), dynamic applications will also send data back to the Modbus module (write coil and register) using the Modbus TCP protocol, making it possible to control data through dynamic applications as well as through the MQTT protocol using a Broker so that users can wirelessly to connect and control data. The data will then be sent and stored in a MySQL or Firebase database in Real-Time. system requirements analysis is shown in Figure 7.

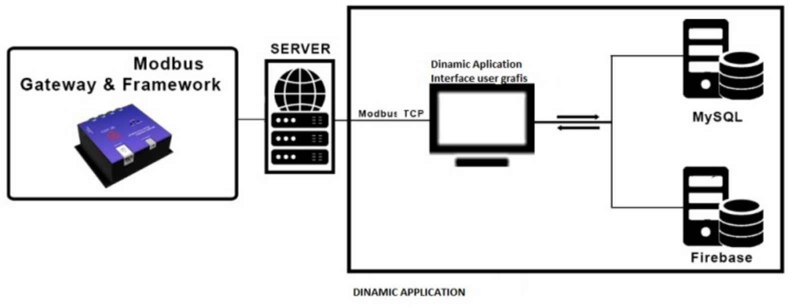
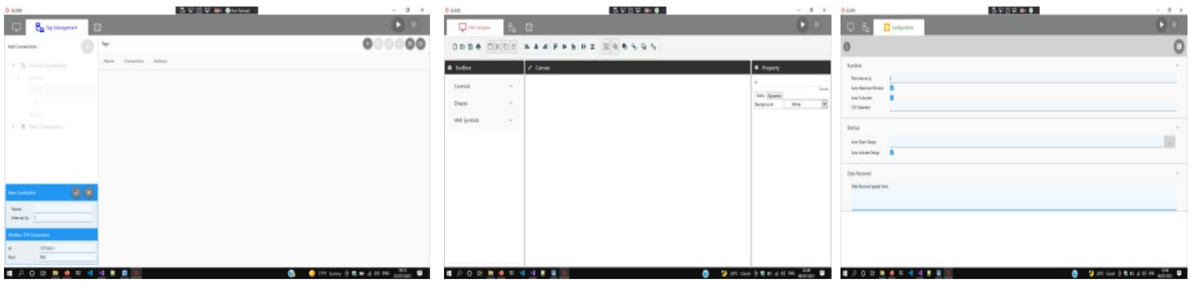


Figure 7. System requirements

The equipment is equipped with a SCADA framework and Dynamic Applications for connectivity configuration and visualization of input data. The application is equipped with a Graphical User Interface which has the option to select the type and protocol of communication between the device and the software. The Graphical User Interface application supports several communications such as Modbus, MQTT and data connections such as SQL and Firebase. In addition, the application also supports dynamic graphics devices using live gauges, moving images and bar indicators, making it easier for users to carry out surveillance, data control and data acquisition in the industry. Figure 8 describes the application's graphical interface.

The application has three workspaces, including the Tag Management tab, Graphic Designer and Configurations. On the Tag Management tab there are several menu boxes consisting of:

1. Device Connection
   On the Device Connection tab, the user can select the communication that will be used to connect the module with the application, including the Modbus TCP/IP protocol, Modbus RTU (serial) or MQTT.
2. Data Connection
   On the Data Connection menu, users can choose a storage method for server data, including using MySQL, or Firebase for Real-Time Databases. On the Graphic Designer tab there are several menus for designing the HMI (Human Machine Interface) display, including:
   i. Menu bar
      On the Menubar there are several icons that function to create new files, set layouts, grouping, copying and pasting objects to commands to print work pages.
   ii. Toolbox
      On the Toolbox tab there are also several menus consisting of menu controls, shapes, to HMI Symbols which serves to add image tags, such as gauges, charts, graphs, or pipe or actuator logos.
   iii. Canvas
      The Canvas menu serves as a work page for the design of the interface that you want to create. In this canvas menu, users can drag and drop and arrange the components they want to create in order to facilitate controlling and monitoring data.
   iv. Properties
      In the property menu there are several forms and configurations to set the properties of the data, such as id tag, image position and size. The last tab is the Configurations tab where we can set the application configuration. The Configuration tab consists of several menus, including:
      a. Runtime
         On the Runtime tab, users can provide Runtime settings such as auto maximize windows to make the application automatically adjust to the maximum screen size, auto full screen to make the application fully open the first time it is run until the plot interval timer to set the time interval for incoming data.
      b. Startups
         Startup has a setting to automatically open the project file that we created earlier in the application.



(a) (b) (c)

Figure 8. Describes the application's graphical interface.

(a). Application Conection Tab (b).Grafis Design Tab (c). Configuration Tab

Test Result

Hardware testing is done to find out the results of testing each hardware component and analyze it.

A. Hardware

1. RS485 to TTL Module Test

RS485 to TTL module serves to convert TTL serial data to RS485 serial. In this test, the RS485 to TTL module is connected to a USB to RS45 converter connected to a computer. Then serial data will be sent through the module to the computer. The following is a test of the RS485 to TTL module. Table 2 shows the test results.

Table 3. Test results RS485 to TTL Module Test

No	Data Sent	Data Received	Status
1	1	1	Sent
2	2	2	Sent
3	3	3	Sent
4	4	4	Sent
5	5	5	Sent

Table 3 shows the results of testing the RS485 to TTL module. This module is capable of sending TTL serial data and converting it to RS485 serial data. From this test, it can be concluded that the RS485 to TTL module can be used to convert TTL serial data to RS485 serial and can be used for data transactions properly.

2. SN65HVD230 CAN Bus Transceiver Module

The SN65HVD230 CAN Bus Transceiver module is used for serial data conversion to the CAN bus. This test uses 2 SN65HVD230 CAN Bus Transceiver modules that are connected to the microcontroller and connect the H pin on module 1 to the H pin of module 2, the L pin of module 1 to the L pin of module 2. The test results of Module, shown in Table 4

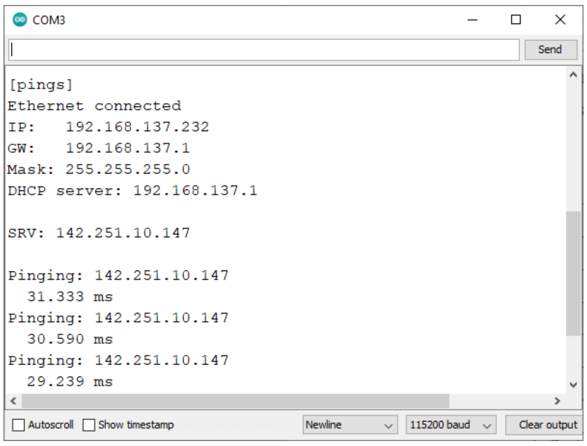
Table 4. SN65HVD230 Module Testing

No	Data Sent	Data Received	Status
1	1	1	Sent
2	2	2	Sent
3	3	3	Sent
4	4	4	Sent
5	5	5	Sent
6	6	6	Sent
7	7	7	Sent

Table 3 shows the test results for the SN65HVD230 CAN Bus Transceiver module. This module is capable of converting serial data to the CAN bus and transmitting data. From this test it can be concluded that the SN65HVD230 CAN Bus Transceiver sensor can be used for serial data conversion to the CAN bus and can also be used for data transmission.

3. ENC28J60 Ethernet Module

The ENC28J60 Ethernet module is used to connect the ESP32 to a local area network (LAN) via an Ethernet cable. Testing the ENC28J60 Ethernet module is done by connecting the module to a PC/laptop or router. The test is carried out using a laptop as a gateway to the local area network. The test was carried out with the ESP32 pinging "www.google.com" to measure latency as well as testing the ENC28J60 Ethernet module connection. The test results shown in figure 9.



```
[pings]
Ethernet connected
IP:   192.168.137.232
GW:   192.168.137.1
Mask: 255.255.255.0
DHCP server: 192.168.137.1

SRV: 142.251.10.147

Pinging: 142.251.10.147
  31.333 ms
Pinging: 142.251.10.147
  30.590 ms
Pinging: 142.251.10.147
  29.239 ms
```

Figure 9. ENC28J60 Ethernet Module Testing On Serial Monitor

In Figure 9, shows a serial monitor image that shows an ethernet connection has been connected, the ESP32 IP address is 192.168.137.232. In the serial monitor image, it can be seen that the connection test by pinging the latency has been successful.

B. Software

Software design aims to implement all communication protocols with HMI Designer so that all aspects of communication can be well integrated with each other. The system is designed using the Visual Studio 2019 application with the XAML programming language as the front-end and the C# language as the back-end. There are several interfaces in the dynamic software developed including:

1. HMI Designer interface

The HMI Designer interface is the first screen that will be encountered when the application is used. Users can freely design the HMI according to the available plans using the symbols and gauges found in the toolbox. All components and symbols used will be placed on the designer's dashboard canvas. In the properties column, users can set the metadata of each object on the canvas such as position, size, font, color and tags for data binding.

2. Interface Tag Manager

In the tag manager, there is a list of data communication protocols and database protocols that can be selected according to the available communication plan. On the right side there is a table that contains the name of the data variable, the name of the communication, and the data value. To facilitate table management, there is a toolbox that contains copy id, paste id, delete id edit id, and import and export tags

3. Interface Settings

On the settings tab, contains several tools to adjust the refresh rate and layout of the application when it is first run. By default, some of the variables on the settings tab are set the first time it is run. However, the user can change the value or status of the available interface settings

ISSN : 2594-4777 (Online) – ISSN : 2597-4742 (Print)

4. Dependency Libraries

To support and facilitate dynamic application design, several library packages are used, all of which are managed by NuGet Package dependencies. All available libraries depend on the version of the .NET Framework used in the project files. The design of dependency libraries is very important considering that the more libraries that are inserted in the project file, the greater the computational resources required which will have an impact on lagging or the application becomes very heavy to run.

5. Application of installer file

Dynamic applications are also equipped with a single installer file or single installer with an executable or .exe format to make it easier to install software on a computer. To design the installer file, the project file must first be published or deployed to convert from a programming language into an executable file that can later be executed by commands on the computer. In designing the installer file, three different files are needed according to their respective functions. To determine the success of the system being developed, several software tests are needed, including:

1. Modbus TCP Communication test
2. Modbus RTU Communication Test
3. MQTT Communication
4. MySQL Database
5. Firebase Database
6. Local CSV data

In reality, the practitioner will assemble a series of specified plants. To provide a real picture of how a plant is assembled and run and controlled and monitored remotely. Then set up the HMI system to make it easier to design a series of practices. An example of an HMI that can be used in this dynamic application system is shown in the figure 10.

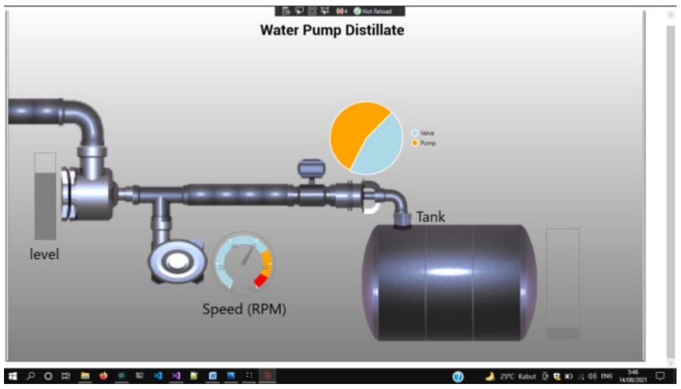


Figure 10. HMI display when full screen.

CONCLUSION

Based on the description above, it is concluded that:

1. Development of Modbus RTU Communication Protocol which can be used as a system capable of controlling and monitoring plant performance
2. Applications in control and monitoring can be applied to practical equipment factories in laboratories or in workshops
3. The appearance of the HMI design can be adjusted to the needs of the equipment in the laboratory or workshop

REFERENCES

Anonym. (2006). *Modbus Messaging on TCP/IP Implementation Guide V1.0b.* October 24**.** Modbus Organization,Inc.,http://www.modbus.org/docs/odbus_Messaging_implementation_Guide_V1_0b.pdf

Anonym. (2012). *Modbus Application Protocol Specification V1.1b3*. April 26. Modbus Organization,Inc., http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

G. B. M. Guarese, F. G. Sieben, T. Webber, M. R. Dillenburg, and C. Marcon. (2012). Exploiting Modbus Protocol in Wired and Wireless Multilevel Communication Architecture, *2012 Brazilian Symposium on Computing System Engineering*.

K. Wang, D. Peng, L. Song, and H. Zhang, (2014). Implementation of Modbus communication protocol based on ARM Coretx-M0," *Conf. Proc. - 2014 Int. Conf. Syst. Sci. Eng. ICSSE 2014*, pp. 69–73, 2014, doi: 10.1109/ICSSE.2014.6887907.

Nurpatmi, (2010). Studi Tentang Modbus Protokol Sistem control, *Forum Teknologi*, Vol.01, No. 2010.

R. Capocci, G. Dooly, E. Omerdić, J. Coleman, T. Newe, and D. Toal, Inspection-Class Remotely Operated Vehicles - A Review*, Journal of Marine Science and Engineering*, vol. 5, no. 1, p. 13, 2017.

S. D. Chandra, H. Kusuma, and Suwito, (2016). *Desain Dan Implementasi Protokol Modbus Untuk Sistem Antrian Terintegrasi Pada Pelayanan Surat Izin Mengemudi (Sim) Di Kepolisian Resort*.

T. Morris, R. Vaughn, and Y. Dandass. (2012). A Retrofit Network Intrusion Detection System for MODBUS RTU and ASCII Industrial Control Systems, *45th Hawaii International Conference on System Sciences*