Modbus Data Provider for Automation Researcher Using C#

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Computer Science.
Type of the Paper: Experimental Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.8162680
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Chakraborty, S., & Aithal, P. S. (2023). Modbus Data Provider for Automation Researcher Using C#. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *7*(3), 1-7. DOI: https://doi.org/10.5281/zenodo.8162680

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJCSBE.2581.6942.0285

Paper Submission: 28/06/2023
Paper Publication: 19/07/2023

Modbus Data Provider for Automation Researcher Using C#

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *Modbus is a popular protocol for data exchange between devices in industrial automation. It has several advantages over other protocols, like noise immunity, long-distance coverage, and easy integration with the microcontroller's serial module. Sometimes we need a device that provides data for our research work. Here we demonstrate a procedure so researchers can create a virtual Modbus client to get Modbus data for their research work. We created a Modbus client in C# language In Visual Studio. We added a couple of modules, Like the Modbus client, serial module, and message display. We added a couple of graphical user interface elements to control the application. The project code is available on GitHub. The researcher can get and customize according to their needs.*

Design/Methodology/Approach: *We created an application in C#. The application has several modules. The main module is the Modbus client. The external device is connected through USB to RS485 converter. When our application starts, the COM object is created. One timer is also started. Its interval is one millisecond. It checks whether the data is reached or not. Once the serial object receives the data, the packet is passed to that Modbus client. The Modbus client starts parsing the received packet. If the packet is OK, then, Extract the command. According to the received command, it created a response packet and added the data of the requested register. Calculate the CRC and add it at the end of the packet. After preparing the packet, send a response back to the master.*

Findings/Result: *Sometimes, the automation researcher does not have the device to provide the research data. Through this research work, we provide a procedure so the researcher can create a virtual Modbus client to provide the data for their research work. So it can be helpful to the researcher to get the data using their working system. We tested it several times with the baud rate of 9600. It is working perfectly without any issues. Researchers can use it for their research work as a software tool.*

Originality/Value: *Several software programs are available to provide the data over the Modbus. Sometimes we need to customize the software according to our requirements. Most of the software is not open source, so here we provide an application so that researchers can optimize and customize the code for their research work so it can provide them with some valuable resources.*

Paper Type: *Experimental-based Research.*

Keywords: Modbus Client, Virtual Modbus client, virtual Data Provider.

1. INTRODUCTION :

Nowadays, research in automation is the most common scenario. We use several products that are produced by automation systems. Now, most of the industry has transformed from traditional to that automation. So industries need automation researchers. In automation, generally, we use Modbus communication to communicate between inter-devices for long distances. It has several advantages. It covers a long distance. It has noise immunity power due to its differential signal in nature. It is easy to interface with the microcontroller UART module. Only one RS485 driver IC is needed. The differential

pair signal quickly converts from Rs TTL level to RS485, conveying the signal around 4000 feet. In Modbus, two types of devices are connected: master and client. A maximum of 32 client devices can be connected at a particular time—only one master available, which initiates the communication. The master initiates all communication. Only the master sends the request to do that to a specific device. All devices in active listening mode respond only to that device that matches the device ID. If the master wants to transmit the command to all devices, send a broadcast message. No client responds to the broadcast message.

Sometimes researcher needs devices that can provide data for their testing purpose. Generally, we use physical Modbus client devices, but the problem is that Modbus client devices cannot send the data according to our requirements without firmware changes. So we need one client device to send the data to the master according to our needs. There are several software available on the web to provide Modbus clients, but most of that is closed source and unable to customize according to our researcher's needs.

Here we developed an open-source application that researchers can integrate into their projects quickly. We created an application in C#. To test this application, we need USB to RS485 modules. First, we have to connect the module with the working system. From the device manager, find the PORT number and add it to the application PORT text box. Press the connect button so it will try to connect with the device's PORT. If the connection is successful, it can send or receive the data. On another end, in Modbus master, send the request, and this application provides the data according to the requested register address. We added some extra features to the Graphical interface. For the code, we kept them for the researcher to implement.

2. RELATED WORKS :

Herath et al. (2020) presented a study on developing a data acquisition and monitoring system using the MODBUS RTU communication protocol [1]. Tofani et al. (2020) discussed implementing a SCADA system for small-system electricity. The authors emphasize SCADA systems' significance in managing and controlling electricity systems [2]. Bajer (2014) explored dataflow in modern industrial automation systems, focusing on both theoretical aspects and practical implementations [3]. Ungurean et al. (2016) proposed a middleware-based architecture for the Industrial Internet of Things (IIoT). The authors highlighted the need for an efficient and scalable architecture to enable seamless communication and coordination among IIoT devices [4]. Phuyal et al. (2020) focused on a study on designing a cost-efficient SCADA system for industrial automation [5]. Fovino et al. (2009) presented a secure MODBUS protocol [6]. Del Carmen Curras-Francos et al. (2014) presented the cooperative development of an Arduino-compatible automation system [7]. Stamatescu et al. (2017) presented the Advanced System for Process Control, a power engineering specialization [8]. Ioana and Korodi (2019) focussed on a VSOMEIP-OPC UA gateway solution used in the automotive industry. The authors highlighted the importance of efficient communication between different automotive systems [9]. He and Jinyong (2018) conducted research on a photovoltaic inverter monitoring system based on cloud service. The authors focused on monitoring photovoltaic inverters using cloud-based technologies [10]. Overall, the literature review covers various topics related to data acquisition, monitoring systems, SCADA, industrial automation, communication protocols, secure protocols, building automation, intelligent systems, and cloud-based monitoring.

3. OBJECTIVES :

The research work aims to provide the researcher with reference information to create and customize the application for research work using the modus. When we work with Modbus, we need data provider tools to test the hardware and our algorithm. The physical Modbus client is not always readily available. So this virtual Modbus client can be helpful to the researcher for their research needs.

4. APPROACH AND METHODOLOGY :

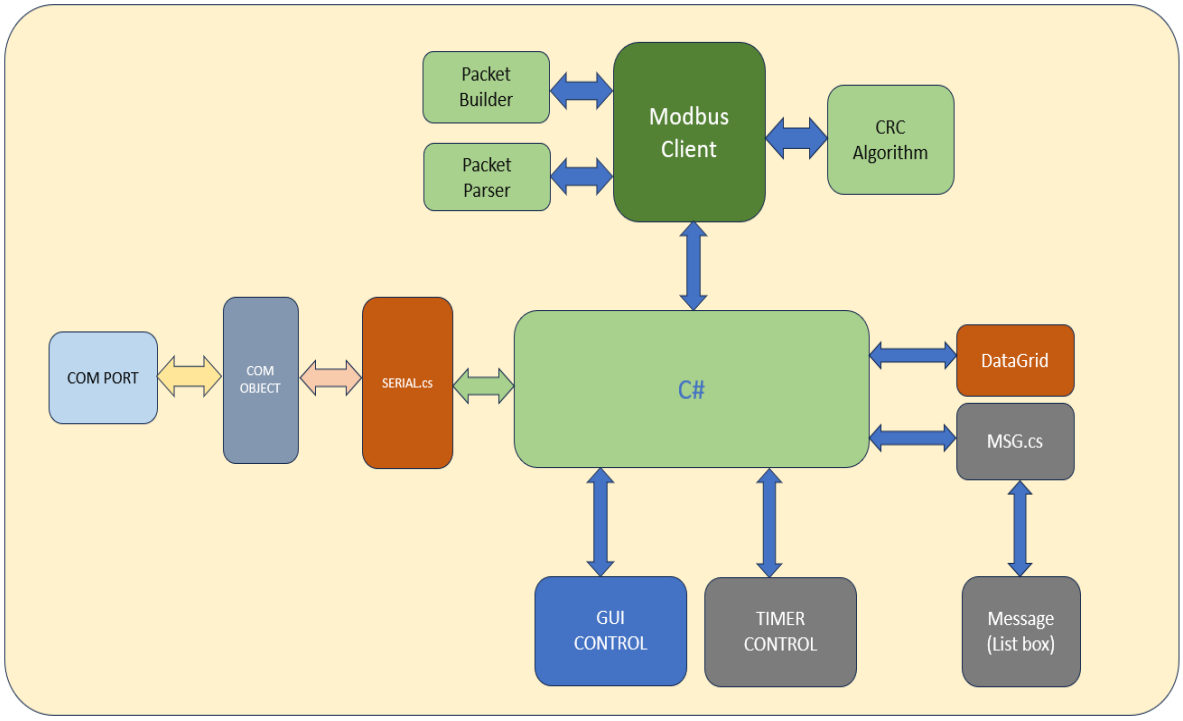

Fig. 1: Block diagram of the project [Source: Authors]

In Figure 4.1, we depicted the complete project through the block diagram. The heart of this block diagram is the C# main thread. It coordinates all of these modules. Now, we go through the module one by one.

Input Module: The left side describes the input converter module connected with the working system to convert RS485 to USB. The most left part is the "COM" PORT, a physical device. When connected with the working system, the operating creates a COM object. The assigned COM port number is available under the device manager. We have a SERIAL module inside the application responsible for communicating with the COM object. When the application starts and presses the connect button inputting the proper COM PORT, it will try to communicate with the hardware module. The application can send or receive the data when a connection is established. We can see the "device connected" message in the message box.

Modbus Client: A Modbus client module is on the top of the figure. Once the data is received, it starts to check CRC by the CRC algorithm. The CRC function is used for two purposes. One is for incoming packet validation, and the other is to add CRC inside the response packet.

GUI Element: Bottom side of that figure is the graphical user element and Timer module. To interact with the user, we added several graphical elements like buttons, textbox, list box, etc. The timer control is used to fetch the data from the serial module from the main thread. The timer interval is one millisecond. It polls within this timeframe. If the data is present, then it starts to process.

Datagrid: in the figure, the right side is a Datagrid. The data grid is used to hold the data of the holding register, and it is also used to modify the data. When we modify the data into the data grid, it modifies into the grid and saves into that holding register, which is the main array to hold the Modbus register. When we change or modify that data grid, it also reflects into that holding register.

Message Display: We display various messages for our understanding. We added one module called "MSG.cs." It is tied with a list box. When we send a string to display, it adds to the first row of the list box control.

5. EXPERIMENT :

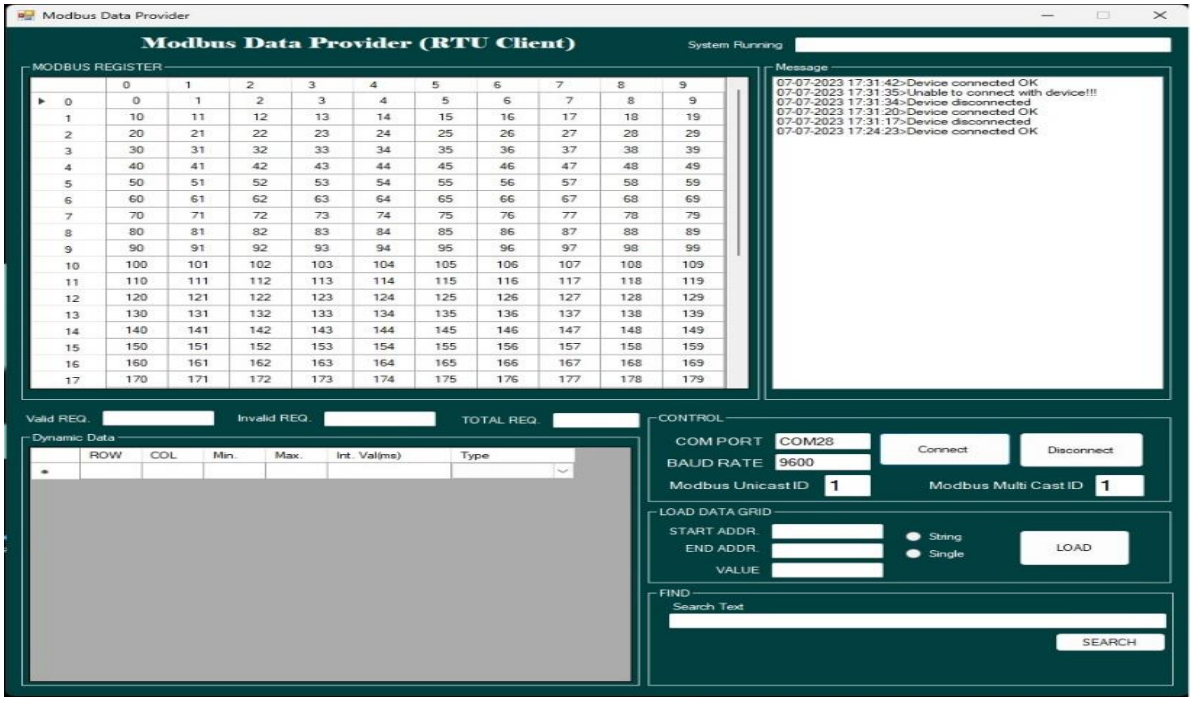

Fig. 2: GUI of the project [Source: Authors]

It is time to experiment to understand the work's flow better. Below are the steps to follow.

1) Download and install Microsoft Visual Studio from the respective website.
2) Download the project from the Github repository. The link is as below: https://github.com/sudipchakraborty/Modbus-Data-Provider-For-Automation-Researcher-Using-C_Sharp.git
3) open "**Modbus_Data_Provider.sln**" solution file. Build and run the project. The interface looks like Figure 2.
4) connect with USB to the RS485 module. From the device manager, note the assigned PORT number, add it to the **COM PORT** textbox, then press connect button.
5) The message box will display a "Device Connected OK" if our application is connected.
6) Inside the application is a textbox labeled "Modbus Unicast ID." Enter desired client ID.
7) Turn on the Modbus master device—Configure Device ID as an application's Unicast ID.
8) Now send the command from the master device and observe the data fetching from the client application. The current fetched location turns green, indicating the address is read by the master.
9) If we do not have a Modbus Master device to test the application, there is another way to test our application.
10) In this scenario, we need two USB to RS485 modules, and we have to connect back to back. Every module is A, and every module's B needs to short each other, i.e., A-A and B-B.
11) Download and open the "**QModMaster"** application depicted in the figure 3. It is a popular application for testing Modbus client devices.
12) Open the test application. Under the "Options," some configurations, like COM port and baud rate, Need to be set. The assigned COM port is available from the device manager.
13) Press connect button. The icon turns into connected if PORT is successfully connected.
14) Now set "**Slave Addr**" which was given into our application.
15) Select Function "Read Holding Registers(0x03)" from the combo box.
16) Enter Start Address from which register we want to read.
17) Fill "Number of Registers" field. It is the number of 16-bit registers to read from our client application.

18) Now click on the "**Read/Write**" button for one time read or click on the "**Scan**" button for continuous reading.
19) Then we observe that the data is coming from our application,
20) we can verify that data is coming from our application only. If we change the data grid value, it will also reflect in the test application. The data is changing, so we are ensuring that our data comes from our application only.

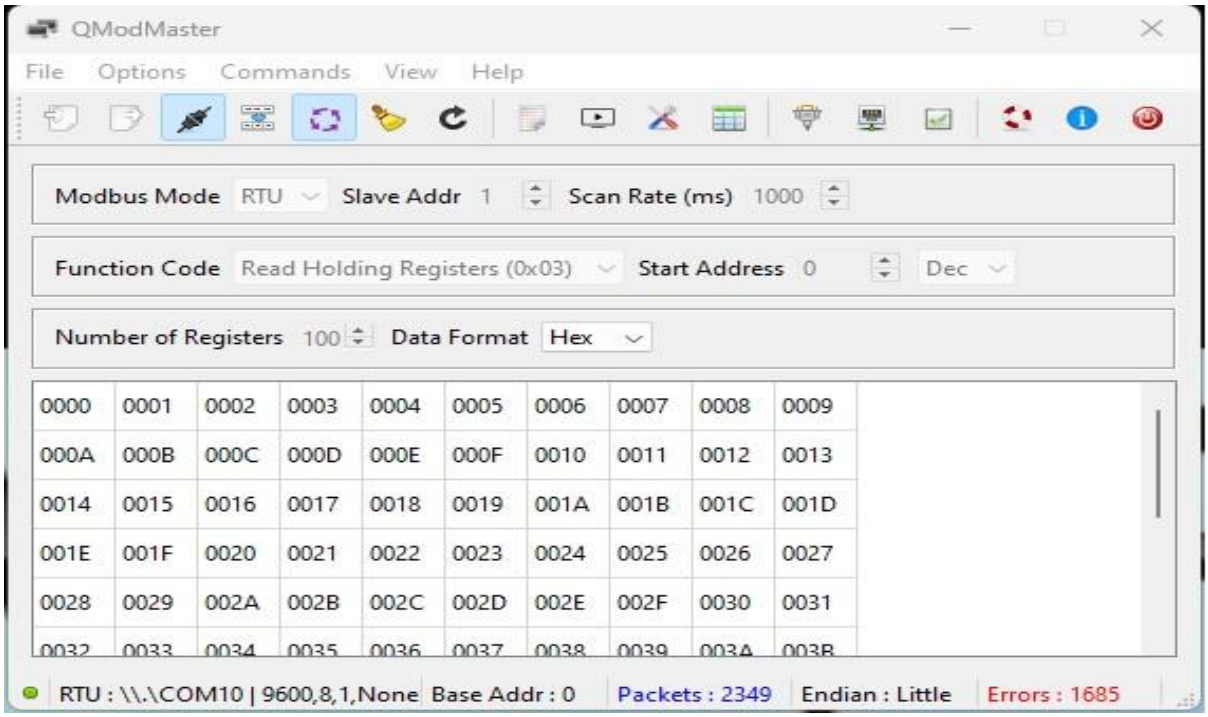

Fig. 3: QModMaster Interface- A popular Modbus Client test program [Source: Authors]

6. RECOMMENDATIONS :

- This research aims to provide valuable information on integrating the virtual Modbus client into the researcher's project.
- We added some GUI elements to the application but did not add the functionality. We left it to the researcher.
- The experienced researcher can add more functionality to this application to make it more helpful.
- The application demonstrates the basic functionality. We kept all functions as simple as possible to understand better.
- Researchers need to handle exceptions and be more functional to make it production-grade.

7. CONCLUSION :

Through this research work, we provided the procedure for creating a data provider application for Modbus research work. Sometimes we need the device to provide the data for our research work. Virtual devices are safer than physical devices, especially at debugging or research time. Here we build an application in C# language under the .NET framework. With the main UI thread, we integrate a couple of modules. Using the application, we can get testing data. We can also get dynamic data adding some functionality to the application. The source code is available on GitHub. The researcher can download and customize it according to the project's needs.

REFERENCES :

[1] Herath, H. M., Ariyathunge, S. V.., & Priyankara, H. D. (2020). Development of a data acquisition and monitoring system based on MODBUS RTU communication protocol. Development, 5(6), 433-440. Google Scholar↗

[2] Tofani, K. M., Permana, P. A., Harsono, B. A., Jintaka, D. R., & Mangunnkusumo, K. H. (2020, October). SCADA system implementation for small system electricity. In 2020 2nd International Conference on Industrial Electrical and Electronics (ICIEE) (pp. 57-61). IEEE. Google Scholar↗

[3] Bajer, M. (2014). Dataflow in modern industrial automation systems. Theory and practice. Int. J. Appl. Control Electr. Electron. Eng, 2(4), 01-11. Google Scholar↗

[4] Ungurean, I., Gaitan, N. C., & Gaitan, V. G. (2016). A Middleware-Based Architecture for the Industrial Internet of Things. KSII Transactions on Internet & Information Systems, 10(7), 2874-2891. Google Scholar↗

[5] Phuyal, S., Bista, D., Izykowski, J., & Bista, R. (2020). Design and implementation of a cost-efficient SCADA system for industrial automation. International Journal of Engineering and Manufacturing, 10(2), 15-28. Google Scholar↗

[6] Fovino, I. N., Carcano, A., Masera, M., & Trombetta, A. (2009). Design and implementation of a secure Modbus protocol. In Critical Infrastructure Protection III: Third Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Hanover, New Hampshire, USA, March 23-25, 2009, Revised Selected Papers 3 (pp. 83-96). Springer Berlin Heidelberg. Google Scholar↗

[7] Del Carmen Curras-Francos, M., Diz-Bugarín, J., García-Vila, J. R., & Orte-Caballero, A. (2014). Cooperative development of an Arduino-compatible building automation system for practical electronics teaching. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, *9*(3), 91-97. Google Scholar↗

[8] Stamatescu, I., Stamatescu, G., Fagarasan, I., Arghira, N., Calofir, V., & Iliescu, S. S. (2017, September). ASID: Advanced system for process control towards intelligent specialization in the power engineering field. In 2017 9th, i.e., the international conference on intelligent data acquisition and advanced computing systems: Technology and Applications (IDAACS) (Vol. 1, pp. 475-480). IEEE. Google Scholar↗

[9] Ioana, A., & Korodi, A. (2019, June). VSOMEIP-OPC UA gateway solution for the automotive industry. In 2019 *IEEE international conference on Engineering, technology, and Innovation* (ICE/ITMC) (pp. 1-6). IEEE. Google Scholar↗

[10] He, Y. A. N. G., & Jinyong, W. A. N. G. (2018, November). Research on Photovoltaic Inverter Monitoring System Based on Cloud Service. In 2018 Chinese Automation Congress (CAC) (pp. 3829-3832). IEEE. Google Scholar↗

****