MODBUS TCP/IP Communication Implementation

İlker Keser, Yahya Ekin
E492 Graduation Project
Advisor: Ergün Gözek
Department Of Electrical And Electronics Engineering

Abstract

Importance of communication protocols increases day by day due to expansion of machinery and automation in the factories. The adaptation of control systems to new machines and control interfaces requires a common language between all the controllers. This common language is MODBUS protocol. MODBUS is the most widespread communication protocol between various devices such as programmable logic controllers (PLC), human machine interface (HMI) panels, etc. In this project, the descendent of the MODBUS protocol, MODBUS TCP/IP communication protocol implemented to generate complete communication between both master device and slave device. To represent the master device, a graphical user interface is designed in python programming language. To simulate the slave device, atmega368 based microcontroller and STM32 based microcontroller used. In both devices, the first 6 function of MODBUS TCP/IP protocol is implemented without using any external libraries. These functions are: Read Coils, Read Discrete Inputs, Read Holding Registers, Read Input Registers, Write Single Coil and Write Single Register. After the implementation, both devices are tested, and the communication is verified. Master program is tested with Schneider Electric PLC and the communication is established.

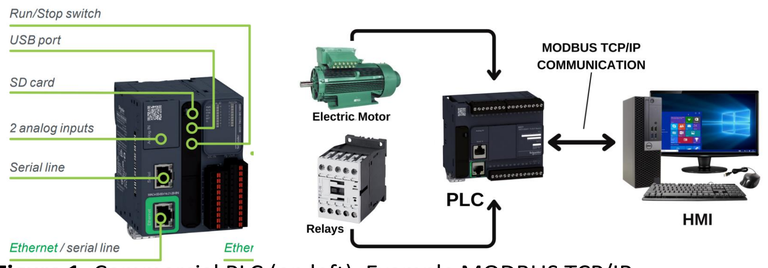

Figure 1. Commercial PLC (on left), Example MODBUS TCP/IP Communication Scheme (on right).

Introduction

In industrial automation and control systems, the Modbus TCP/IP protocol is a commonly used communication standard. It was first created in the late 1970s for use in programmable logic controllers (PLCs) by Modicon, which is now part of Schneider Electric. An extension of the Modbus protocol designed for TCP/IP networks; Modbus TCP/IP enables smooth device-to-device communication via Ethernet.

What is PLC ?

PLCs, or programmable logic controllers, are industrial computers that are frequently employed in control and automation systems. They are intended to execute specialized functions and govern a variety of industrial operations. PLCs consist of CPU, input/output modules which might be transistor or relay based, and programming device.

Why MODBUS protocol developed ?

After the development of various kind of PLCs, the need for communication between them occurred due to application reasons. As a simple example, a factory which had X brand PLC should had to buy the same brand to establish communication between them. Since there are numerous brands in the industry, to establish communication among different brands MODBUS protocol is developed. By this development, the PLC industry is grooved, and PLCs became more advanced than before. MODBUS TCP/IP frame is shown in figure X.

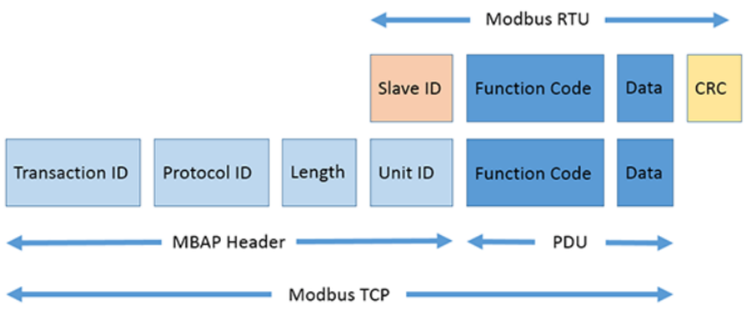

Figure 2. MODBUS TCP/IP Frame Structure.

Problem Definition

In the world of industrial automation, where precision and efficiency are critical, there's a growing demand for advanced monitoring and control systems. These systems must not only monitor parameters in real-time but also control analog and digital inputs, and generate digital and analog outputs. They need to ensure seamless communication and adaptability to the fast paced industrial environment. As an example factory, the need of reading sensor data and controlling the input/output of the production devices from the computer using graphical user interface is occurred. The expected communication system consists of 1 master device which has graphical user interface to perform the first 6 function of MODBUS TCP/IP protocol and 1 slave device which has capability of performing the first 6 function of MODBUS TCP/IP protocol needs to be designed. The design constraint is using MODBUS libraries from anywhere for both system is forbidden.

Proposed Solution

The solution consists of developing master device on python language. To generate the user interface, tkinter module of the python is used. The GUI performed the first 6 functions of MODBUS protocol. To develop a slave device, Arduino Nano is selected. Due to simulation of the connected devices to slave device, 2 LED is used to represent 2 connected coil. 2 push-button used to simulate 2 discrete inputs. Also, 2 potentiometer is used as analog inputs to slave device.

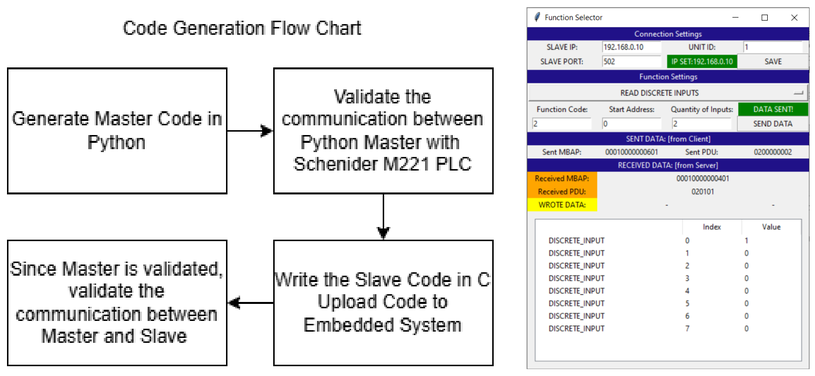

Figure 3. Code generation flowchart (on left), Master Python GUI (on right).

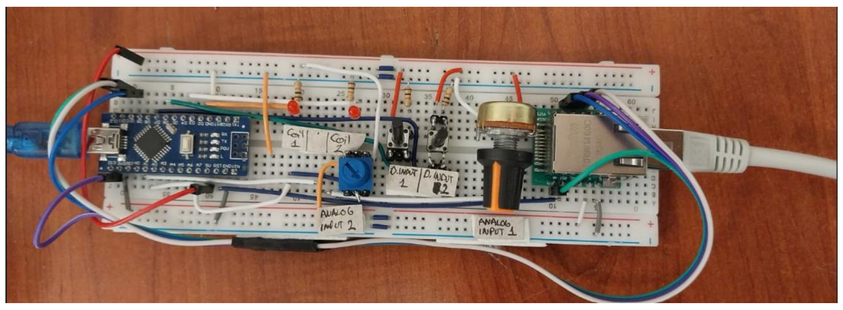

Figure 4. Our Arduino Nano based test setup.

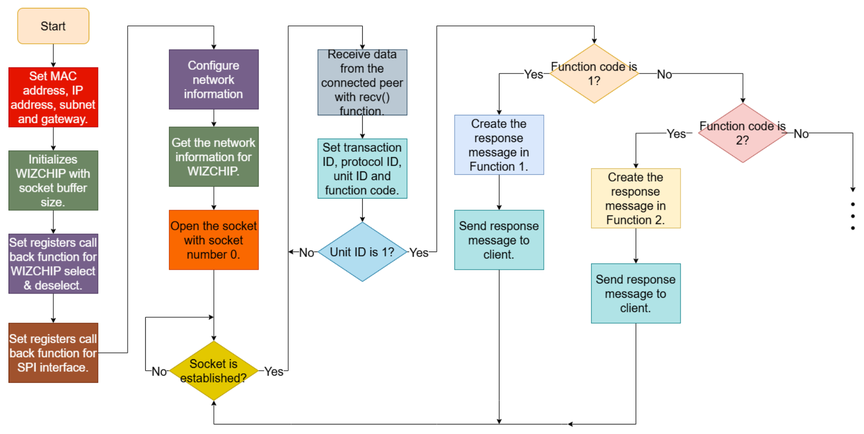

Figure 5. Flowchart of slave program.

Table 1. First 6 MODBUS functions and their purposes.

Function	Purpose
Read Coils	Reads the status of connected coils in remote device.
Read Discrete Inputs	Reads the status of connected digital inputs in remote device.
Red Holding Registers	Reads the memory (holding) registers of remote device.
Read Input Registers	Reads the input registers (analog inputs) of the remote device.
Write Single Coil	Drives the connected coil on remote device.
Write Single Register	Writes data to memory(holding) registers of remote device.

Figure 6. W5500 Module.

MASTER DEVICE PYTHON GUI

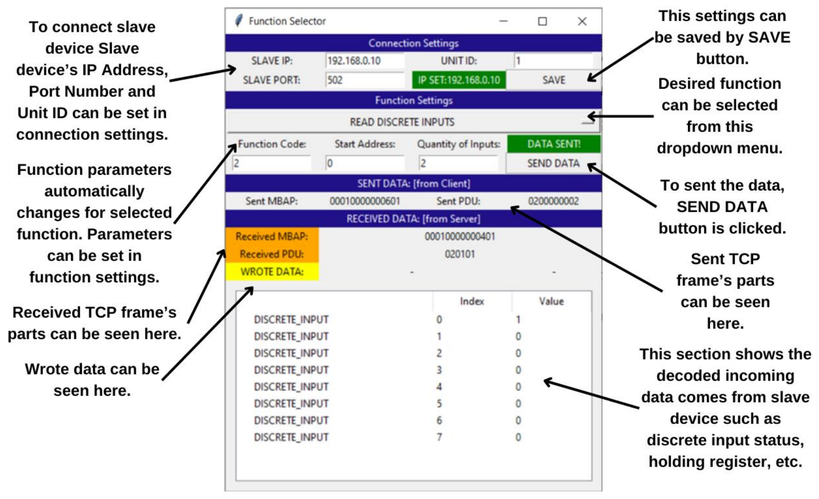

Figure 7. Master GUI explanation.

Results & Discussion

Result of communication is shown in the proposed solution section of the report for every implemented MODBUS function. This section briefly explain the communication results in three different cases.

- Python Master – Schneider M221 PLC Slave
- Python Master – Arduino Nano Slave
- Python Master – STM32F407 Discovery Board Slave

As a summary, in all cases although there are minor faults, majority of MODBUS TCP/IP communication functions between Master program and Slave programs is implemented successfully. Our Master communicated with commercial PLC and our embedded systems.

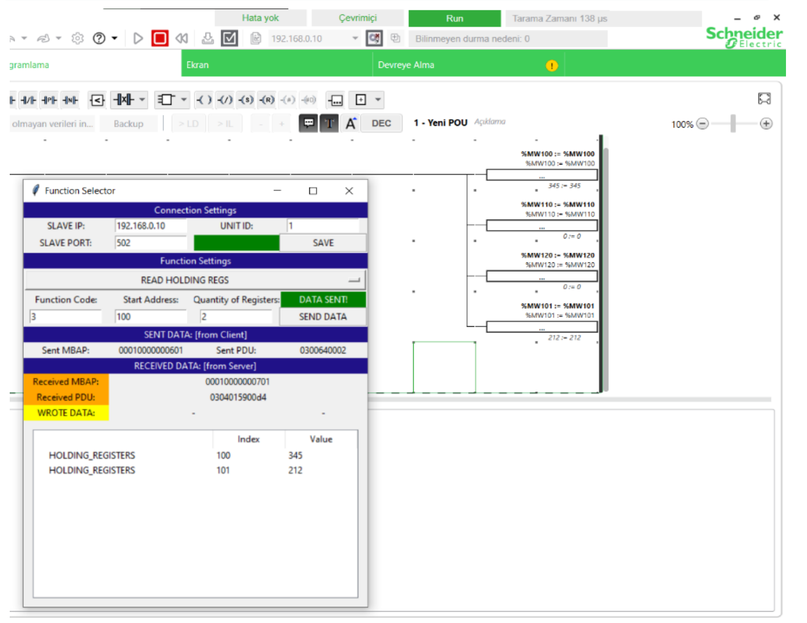

Figure 8. Python Master Schneider M221 PLC Slave test.

Figure 9. Python Master STM32F4 Discovery Slave test.

References

1. modbus.org. (n.d.). Modbus. Retrieved from https://www.modbus.org/
2. Modbus-IDA. (n.d.). Modbus Application Protocol V1.1b. Retrieved from https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf
3. IPC2U. (n.d.). Detailed Description of the Modbus TCP Protocol with Command Examples. Retrieved from https://ipc2u.com/articles/knowledge-base/detailed-description-of-the-modbus-tcp-protocol-with-command-examples/
4. Modbus-IDA. (n.d.). Modbus Messaging Implementation Guide V1.0b. Retrieved from https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf