Mateusz Krawczak 241318

Karol Jaskółka 241306

Grupa: Pon P 17:00

Data wykonania ćwiczenia: 13.11.2019

Urządzenia Peryferyjne

Ćwiczenie 18 – Analizator parametrów sieci

1. Wstęp

Na drugich zajęciach laboratoryjnych otrzymaliśmy zadanie polegające na napisaniu programu pozwalającego na odczytanie różnych parametrów m.in. napięcia prądu dzięki analizatorowi parametrów sieci EMA-90N firmy Contrel. Zadania do wykonania:

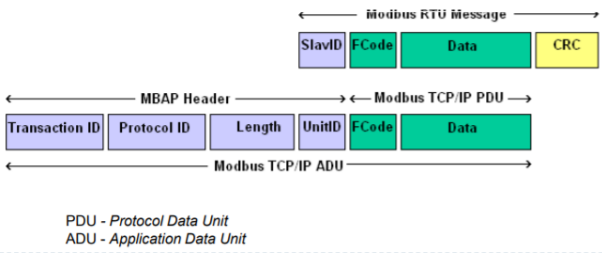
- Połączyć urządzenie EMA-90N z komputerem za pomocą komunikacji Ethernet.
- Uruchomić aplikację demonstracyjną i połączyć się z urządzaniem odczytując napięcie i prąd na L1.
- Napisać aplikację w C#, która połączy się z urządzeniem i umożliwi odczytanie napięcia i prądu L1 z użyciem protokołu modbus.

2. Zagadnienia

- **Protokół Modbus:**
  - protokół komunikacyjny opracowany przez firmę Modicon
  - wykorzystuję regułę wymiany danych typu master-slave (nadrzędny-podrzędny)
  - wykorzystywany do znakowej wymiany informacji pomiędzy urządzeniami

- **Modbus TCP/IP:**
  - to protokół Modbus RTU z interfejsem TCP/IP
  - wykorzystuję sieć Ethernet do transportu danych
  - dla aplikacji wykorzystuje port systemowy 502

- **Modbus TCP/IP ramka:**
  - z ramki Modbus RTS usuwa się pole adresu (indentyfikator slave'a) oraz pole sumy kontrolnej CRC, natomiast dalej wykorzystuję się pole kodu funkcji i danych
  - dodatkowo ramka TCP/IP rozrasta się o 7 bajtowy nagłówek MBAP (Modbus Application Header)



PDU - *Protocol Data Unit*
ADU - *Application Data Unit*

- **Nagłówek MBAP :**
  - identyfikator transakcji (2 bajty): do identyfikacji kolejnego zapytania w ramach jednego połączenia TCP
  - identyfikator protokołu (2 bajty): obecnie niewykorzystywane (ustawione na zero 0000)
  - długość pola: liczba bajtów wiadomości (2 bajty)
  - identyfikator jednostki (1 bajt) : identyfikuje podłączonego klienta

- **Wybrane kody funkcji:**
  - 0x01 odczyt wyjść bitowych
  - 0x02 odczyt wejść bitowych
  - 0x03 odczyt n rejestrów wyjściowych
  - 0x04 odczyt n rejestrów wejściowych

- **Numery rejestrów:**
  - Discrete Output Coils: 1-9999
  - Discrete Input Contacts: 10001-19999
  - Analog Input Registers: 30001-39999
  - Analog Output Holding Registers: 40001-49999

- **Przykładowa analiza ramek (według przykładu na simplymodbus.ca):**
  Request :

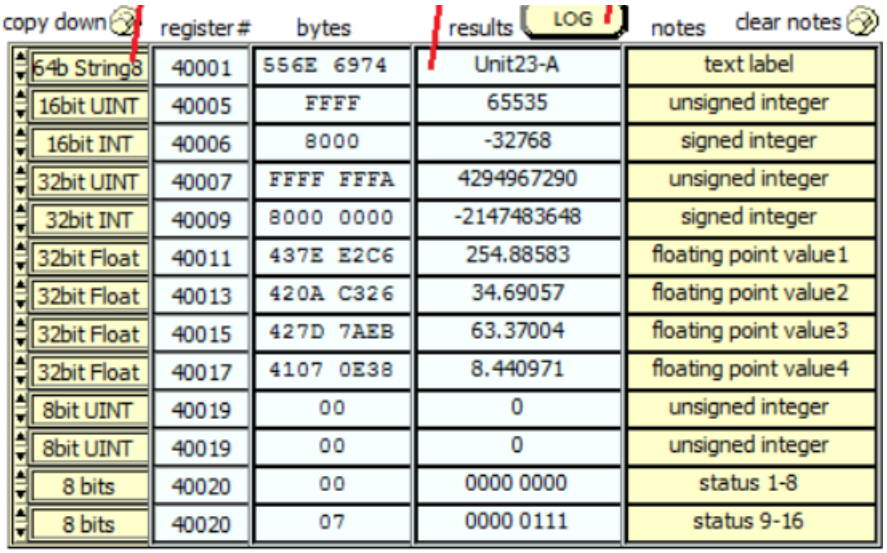
```
2015/10/02 13:58:30  >>> 00 06 00 00 00 06 01 03 00 00 00 14
```

  - 0006 : identyfikator transakcji
  - 0000 : identyfikator protokołu
  - 0006 : długość wiadomości (6 kolejnych bajtów)
  - 01: identyfikator jednostki 0x01
  - 03: kod funkcji (0x03 read analog output holding registers)
  - 0000: adres pierwszego odpytywanego rejestru (40001)
  - 0014: ilość odpytywanych rejestrów heksadecymalnie 0x14

  Response:

```
2015/10/02 13:58:30  < 00 06 00 00 00 3B 01 03 28 55 6E 69 74 32 33 2D 41 FF FF 80 00 FF FF FF FA 80 00
00 00 43 7E E2 C6 42 0A C3 26 42 7D 7A EB 41 07 0E 38 00 00 00 07
```

  - 0006 : identyfikator transakcji
  - 0000 : identyfikator protokołu
  - 003B : długość wiadomości heksadecymalnie 0x3B
  - 01: identyfikator jednostki 0x01
  - 03: kod funkcji (0x03 read analog output holding registers)
  - pozostałe bajty to zawartości rejestrów jak na zdjęciu poniżej



copy down	register #	bytes	results	LOG	notes	clear notes
64b String	40001	556E 6974	Unit23-A		text label	
16bit UINT	40005	FFFF	65535		unsigned integer	
16bit INT	40006	8000	-32768		signed integer	
32bit UINT	40007	FFFF FFFA	4294967290		unsigned integer	
32bit INT	40009	8000 0000	-2147483648		signed integer	
32bit Float	40011	437E E2C6	254.88583		floating point value1	
32bit Float	40013	420A C326	34.69057		floating point value2	
32bit Float	40015	427D 7AEB	63.37004		floating point value3	
32bit Float	40017	4107 0E38	8.440971		floating point value4	
8bit UINT	40019	00	0		unsigned integer	
8bit UINT	40019	00	0		unsigned integer	
8 bits	40020	00	0000 0000		status 1-8	
8 bits	40020	07	0000 0111		status 9-16	

Rysunek 1 Przykład ze strony http://www.simplymodbus.ca/

3. Przebieg zajęć i kod programu

- Na początku zajęć udało nam się połączyć urządzenie EMA-90N z komputerem za pomocą komunikacji Ethernet i uruchomić aplikację demonstracyjną
- Następnie przystąpiliśmy do napisania aplikacji w C# wykorzystując bibliotekę EasyModbus
- Zgodnie z zaleceniem ze strony z zadaniem pole UnitIdentifier (identyfikator jednostki) zostało ustawione na wartość 0x01
- Poniższy kod powstał w oparciu o dokumentację biblioteki EasyModbus

Cały program znajduję się pod tym linkiem
https://github.com/matson19/UP/tree/master/Lab%202%20-%20analizator_sieci

```
public partial class Form1 : Form
    {
       //obiekt klasy modbus potrzebny do odczytania parametrow
        ModbusClient modbusClient;

        public Form1()
        {
            InitializeComponent();
        }

        private void buttonConnect_Click(object sender, EventArgs e)
        {
            try
            {
             //inicjalizacja ip podanego przez użytkownika oran numer portu
                modbusClient = new ModbusClient(textBoxIP.Text, 502);
                modbusClient.UnitIdentifier = 0x01;
                modbusClient.ConnectionTimeout = 350;
                modbusClient.Connect();
                labelStatus.Text = "Connected";
```

```
                timer.Start();
            } catch(Exception ex)
            {
                labelStatus.Text = ex.ToString();
            }

        }

        private void buttonDisconnect_Click(object sender, EventArgs e)
        {
            modbusClient.Disconnect();
            labelStatus.Text = "Offline";
        }

        private void timer_Tick(object sender, EventArgs e)
        {

            modbusClient.WriteMultipleCoils(4, new bool[] { true, true, true, true,
true, true, true, true, true, true });    //Write Coils starting with Address 5
            bool[] readCoils = modbusClient.ReadCoils(9, 10);
//Read 10 Coils from Server, starting with address 10
            int[] readHoldingRegisters = modbusClient.ReadHoldingRegisters(0, 10);
//Read 10 Holding Registers from Server, starting with Address 1

            for (int i = 0; i < readCoils.Length; i++)
            {
                textBox1.AppendText("Value of Coil " + (9 + i + 1) + " " +
readCoils[i].ToString());
            }

            for (int i = 0; i < readHoldingRegisters.Length; i++)
            {
                textBox2.AppendText("Value of HoldingRegister " + (i + 1) + " " +
readHoldingRegisters[i].ToString());
                modbusClient.Disconnect();
            }

            timer.Stop();
        }
```

Próby uruchomienia kody zakończyły się niepowodzeniem oraz uzyskaniem błędu EasyModbus.Exception.FunctionCodeNotSupportedException: ‘Function code not supported by master’

4. Podsumowanie

Program nie działał całkowicie poprawnie, ponieważ pomimo faktu, że pogram wykrywał urządzenie to przy próbie odczytania zawartości rejestrów otrzymywaliśmy wyżej wymieniony błąd. W związku z czym nie byliśmy w stanie odczytać parametrów prądu. Kod programu został wykonany zgodnie z dokumentacją EasyModbus, a mimo tu to podczas kompilacji wystąpił błąd. Był on prawdopodobnie spowodowany problemem programu z określeniem które urządzenie funkcjonuję jako master, a które jako slave. Działanie protokołu modbus TCP/IP oraz analiza ramek została dokładnie przestawiona w punkcie numer 2 (Zagadnienia).