Programming Assignment4

19학기 2분반
21600301
박지현

#Abstract

Modbus TCP 프로토콜을 이용하여 다음 네 가지의 통신을 수행하는 modTCPClient 프로그램을 만든다.

1. Read coils
2. Write Multiple Coils
3. Read Holding Registers
4. Write multiple(holding) Registers

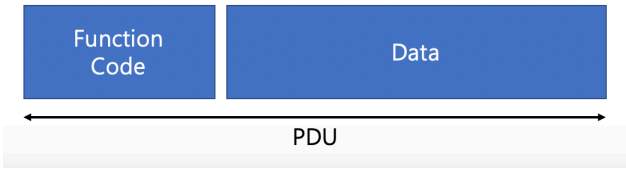
#Process

modTCPClient의 소스 파일 명은 modTCPClient.c으로, 이 C파일이 컴파일 한 프로그램을 실행함으로 위 기능을 수행할 수 있다. 실행 코드는 다음과 같은 명령어로 실행되어야 한다.

```
→ $ modTCPClient <Modbus-TCP_server_IP_address>
```

#Result

1. Modbus-TCP의 서버의 포트 번호는 502번이다.
2. Modbus-TCP의 PDU 포맷은 다음과 같이 생겼다.



Function code는 1Byte를 차지하고 Data는 FunctionCode에 따라 길이가 가변적이다.

3. "Read coils" 기능에 대한 request와 response format의 형태는 다음과 같다.

< Request>

PDU의 크기 : 5bytes이다.

Function Code (0x01)	Starting Address	Quantity of Outputs (1~n)

Function Code(1bytes) : Read coils의 함수 코드는 0x01이다.
Starting Address(2bytes) : 첫번째로 읽을 코일의 주소이다.

Quantity of Outputs(2bytes) : 읽어 들일 코일의 개수이다.

<Response>

크기 : 최소 3bytes이다.

ByteCount가 1씩 증가함에 따라 Output Status를 표현하기 위한 바이트를 하나씩 추가한다. **즉, 3bytes이상의 크기를 가진다.**

Function Code (0x01)	ByteCount	Output Status

Function Code(1bytes) : Read coils의 함수 코드는 0x01이다.

Byte Count(1bytes) : 코일의 수를 표현하기 위해 필요한 바이트 수이다. Byte Count는 (읽어들일 코일의 개수+7) / 8로 계산된다.

Output Status(1bytes~) : 가장 왼쪽 output status byte 부터 오른쪽으로 코일의 상태에 대한 정보를 기록한다. 한 바이트 내부에는 8개의 코일에 대한 정보가 담겨 있다. 여기서 바이트 내부에 각 코일에 대한 상태를 기록할 때는 LSB부터 기록하여 MSB방향으로 기록하도록 한다.

4. "Write multiple coils" 기능에 대한 request와 response format의 형태는 다음과 같다.

<request>

크기 : 최소 7bytes이상.

ByteCount가 1씩 증가함에 따라 Output Status를 표현하기 위한 바이트를 하나씩 추가한다. 즉, **7bytes이상의 크기를 가진다.**

Function Code (0x0F)	Starting Address	Quantity of Outputs (1~n)	ByteCount	Output value

Function Code(1bytes) : Read coils의 함수 코드는 0x0F이다.

Starting Address(2bytes) : 첫 번째로 쓸 코일의 주소이다.

Quantity of Outputs(2bytes) : 쓰고자 하는 코일의 개수이다.

Byte Count(1bytes) : 코일의 수를 표현하기 위해 필요한 바이트 수이다. Byte Count는 (읽어 들일 코일의 개수+7) / 8로 계산된다

Output Value(1byte~) : 가장 왼쪽 output status byte 부터 오른쪽으로 코일에 쓸 정보를 기록한다. 한 바이트 내부에는 8개의 각 코일에 쓰고자 하는 정보가 담겨 있다. 여기서 바이트 내부에 각 코일에 대한 상태를 기록할 때는 LSB부터 기록하여 MSB방향으로 기록하도록 한다.

<response>

크기 : 5bytes

Function Code (0x0F)	Starting Address	Quantity of Outputs (1~n)

Function Code(1bytes) : Write multiple coils의 함수 코드는 0x0F이다.

Starting Address(2bytes) : 첫번째로 읽을 코일의 주소이다.

Quantity of Outputs(2bytes) : 읽어 들일 코일의 개수이다.

5. "Read Holding Registers Function" 기능에 대한 request와 response format의 형태는 다음과 같다.

<request>

크기 : 5bytes

Function Code (0x03)	Starting Address	Quantity of Outputs (1~n)

Function Code(1bytes) : Read Holding Registers의 함수 코드는 0x03이다.

Starting Address(2bytes) : 첫번째로 읽을 레지스터의 주소이다.

Quantity of Outputs(2bytes) : 읽어 들일 레지스터의 개수이다.

<Response>

크기 : 최소 4bytes이다. ByteCount가 1씩 증가함에 따라 Output Status를 표현하기 위한 바이트를 두 개씩 추가한다.(레지스터는 16bits이기에) **즉, 4bytes이상의 크기를 가진다**.

Function Code (0x03)	ByteCount	Register Value

Function Code(1bytes) : Read Multiple Holding Regsiters의 함수 코드는 0x03이다.

Byte Count(1bytes) : 레지스터의 수를 표현하기 위해 필요한 바이트 수이다. Byte Count는 레지스터의 개수X2로 계산된다.

Register Value(2bytes~) : 가장 왼쪽 Register Value부터 오른쪽으로 2byte씩 시작 주소부터 마지막 레지스터의 값에 대한 정보를 기록한다. 2바이트 내부에는 각 레지스터에 대한 에 대한 정보가 담겨 있다. 여기서 2바이트 내부에 각 레지스터의 값을 기록할 때는 LSB부터 기록하여 MSB방향으로 기록하도록 한다.

6. "Write multiple (holding) registers"기능에 대한 request와 response format의 형태는 다음과 같다.

<request>

크기 : 최소 8bytes이상.

ByteCount가 1씩 증가함에 따라 Register Value를 표현하기 위한 바이트를 두 개씩 추가한다. **즉, 8bytes이상의 크기를 가진다.**

Function Code (0x10)	Starting Address	Quantity of Registers (1~n)	ByteCount	Register Value

Function Code(1bytes) : Write multiple registers의 함수 코드는 0x10이다.
Starting Address(2bytes) : 첫 번째로 쓸 레지스터의 주소이다.
Quantity of Outputs(2bytes) : 쓰고자 하는 레지스터의 개수이다.
Byte Count(1bytes) : 레지스터의 수를 표현하기 위해 필요한 바이트 수이다. Byte Count는 레지스터의 수X 2 로 계산된다
Register Value(2byte~) : 가장 왼쪽 Register Value부터 오른쪽으로 코일에 쓸 정보를 2byte씩 기록한다. 2바이트 내부에는 Register에 쓰고자 하는 value 정보가 담겨 있다. 여기서 2바이트 내부에 각 코일에 대한 상태를 기록할 때는 LSB부터 기록하여 MSB방향으로 기록하도록 한다.

<Response>
크기 : 5bytes

Function Code (0x10)	Starting Address	Quantity of Registers (1~n)

Function Code(1bytes) : Write Multiple Registers의 함수 코드는 0x10이다.
Starting Address(2bytes) : 첫번째로 읽을 레지스터의 주소이다.
Quantity of Outputs(2bytes) : 읽어 들일 레지스터의 개수이다.

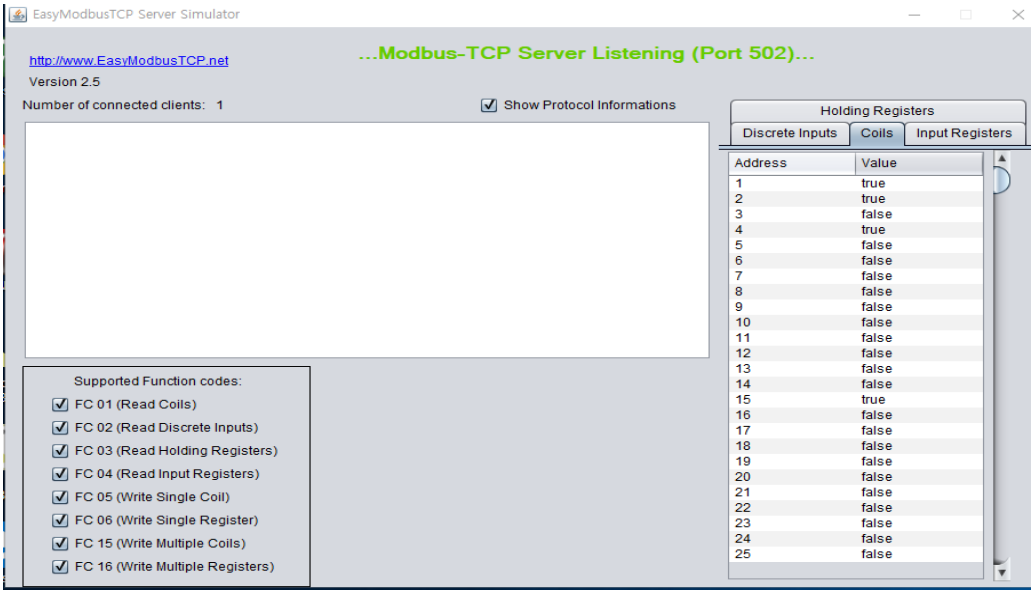
<다음 실행환경은
SERVER 413실습실 server : 203.252.118.204
CLIENT : 203.252.106.83 에서 이루어진 것이다. >

7. "Read coils"기능에 대한 예시로 modTCPClient프로그램에서 기능을 수행할 때의 동작은 다음과 같다.

```
============================
What function do you want?
(1) : Read Coils
(2) : Write multiple Coils
(3) : Read Holding Registers
(4) : Write multiple (Holding) Registers
(0) : Quit
============================
Select a function [1, 2, 3, 4, 0] :1
Enter the Start Address :0
Enter the number of Coils to Read :4

-----<RESPONSE>-----
::Recved length is 10 BYTES
::Function - Read Coils
::3 coil(s) - 'True' state from server!
--------------------
```

<client쪽에서 보낸 request와 받은 response>



<server쪽 코일 상태>

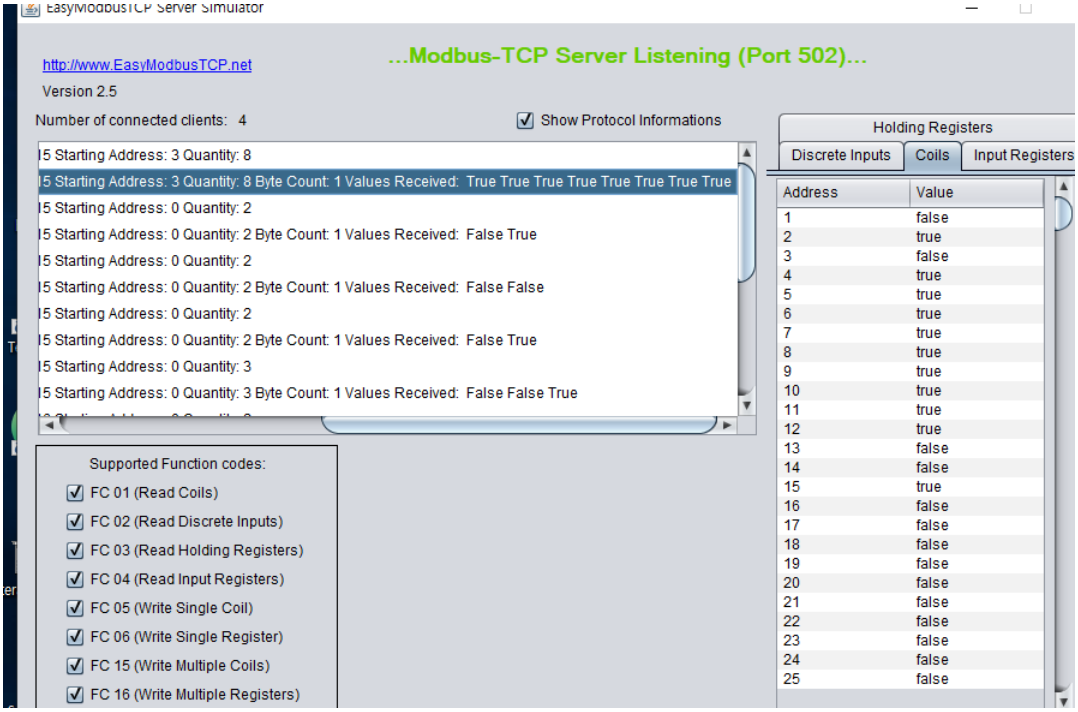
Client : 1번 Read coils를 하고자 하였다. 시작 주소를 0,(서버 상에서는 1이다)로 하여 4개의 코일을 읽는 request를 보냈다.
Server : coil을 보면, 1,2,4번 코일이 true이다.
Client : 서버에서 시작 주소를 0으로 하는 보낸 코일 4개중 받은 코일 중 3개가 true이고 나머지 하나는 false라는 응답을 받는다.

8. "Write multiple coils" 기능에 대한 예시로 modTCPClient프로그램에서 기능을 수행할 때의 동작은 다음과 같다.

```
============================
What function do you want?
(1) : Read Coils
(2) : Write multiple Coils
(3) : Read Holding Registers
(4) : Write multiple (Holding) Registers
(0) : Quit
============================
Select a function [1, 2, 3, 4, 0] :2
Enter the Start Address :3

Enter the number of Coils in you want to write on(N of coils at most 8) :8
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:3
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:4
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:5
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:6
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:7
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:8
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:9
Enter the Coil address #3 to #10 you want switch ON, otherwise left will be OFF
:10
255
-----<RESPONSE>-----
::Recved length is 12 BYTES
::Function - Write Multiple Coils
::Starting Address is 3!
::The number of controlled Coils in server is 8
--------------------
```

<client쪽에서 보낸 request와 response>



<server 쪽에서 바뀐 코일의 상태>

Client : 2번 Write multiple coils를 하고자 하였다. 시작 주소를 3,(서버 상에서는 4이다)로 하여 8개의 코일을 write request를 보냈다.

Server : coil을 보면, 4,5,6,7,8,9,10,11번 코일이 true로 변했다.

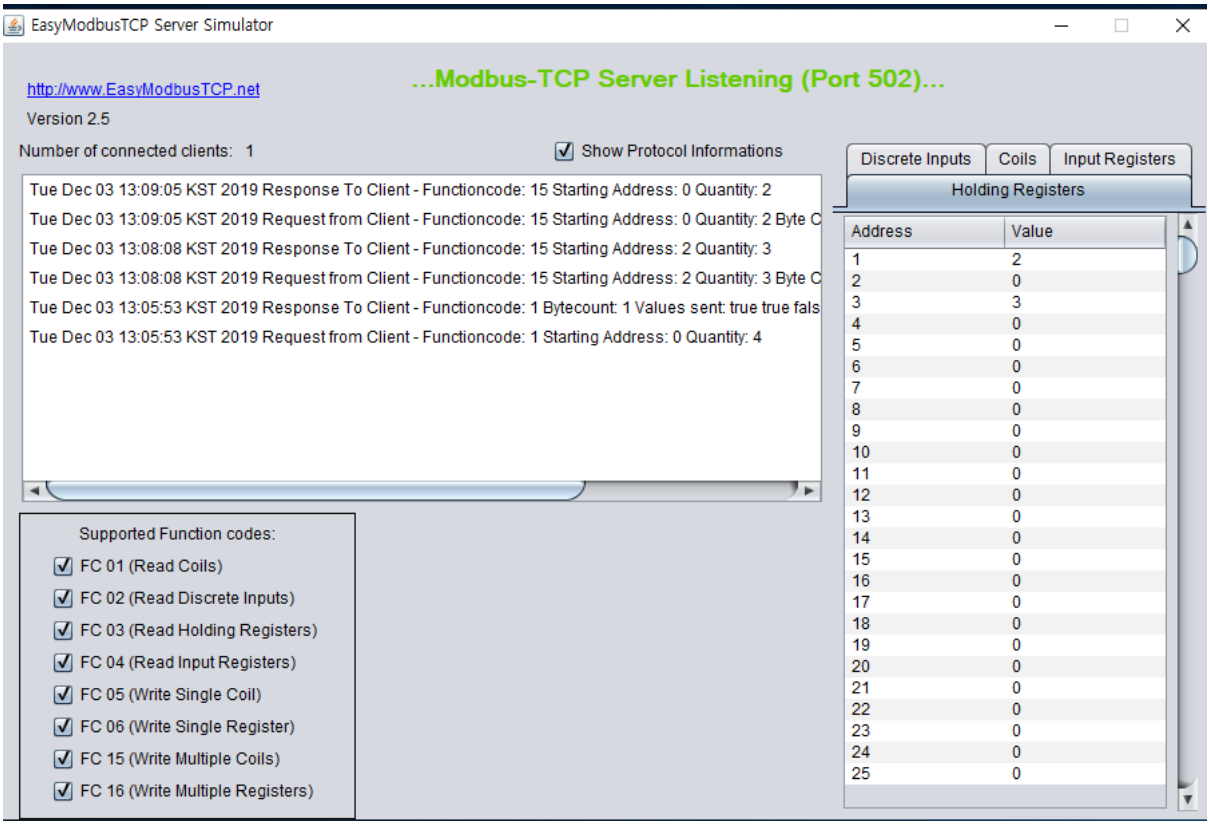
Client : 서버에서 시작 주소를 3으로 하는 8개의 코일을 true로 바꾼 응답을 받았다.

9. "Read Holding Register Function" 기능에 대한 예시로 modTCPClient프로그램에서 기능을 수행할 때의 동작은 다음과 같다

```
============================
What function do you want?
(1) : Read Coils
(2) : Write multiple Coils
(3) : Read Holding Registers
(4) : Write multiple (Holding) Registers
(0) : Quit
============================
[Select a function [1, 2, 3, 4, 0] :3
[Enter the Start Address :0
[Enter the number of Registers to Read (AWARE!available range from 1 to 8) :4

-----<RESPONSE>-----
::Recved length is 17 BYTES
::Function - Read holding Registers
::The # of written on registers is 4 in server
::Value of Registers got from server are following
Value of 0(th) register from starting address = 2
Value of 1(th) register from starting address = 0
Value of 2(th) register from starting address = 3
Value of 3(th) register from starting address = 0
```

<client에서 보낸 request 와 받은 response>



<server의 register 상태>

Client : 3번 read multiple holding registers를 하고자 하였다. 시작 주소를 0,(서버 상에서는 1이다)로 하여 4개의 코일을 read request를 보냈다.
Server : 주소1(클라이언트에서 0)부터 시작하는 레지스터 4개의 값을 보낸다.
Client : 서버에서 시작 주소를 0으로 하는 4개의 레지스터 값을 읽고 순서대로 출력한다.

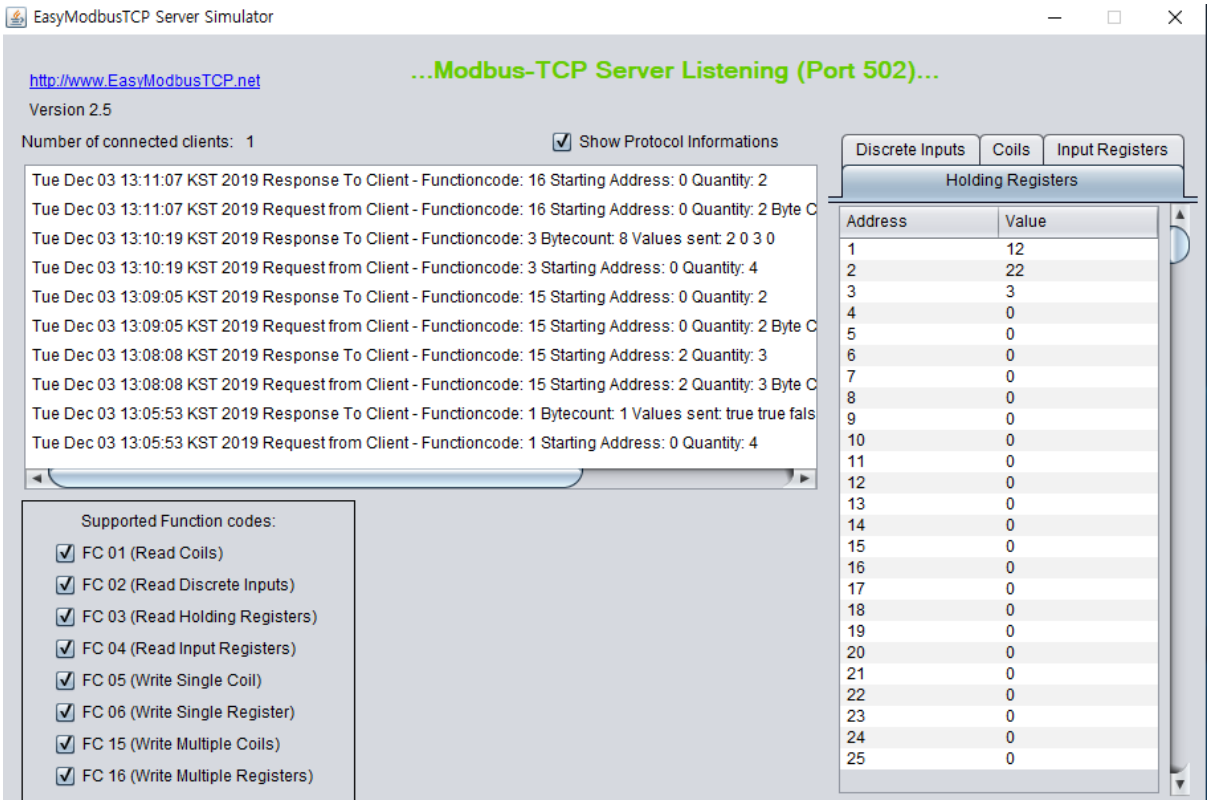
10. "Write multiple (holding) registers"기능에 대한 예시로 modTCPClient프로그램에서 기능을 수행할 때의 동작은 다음과 같다

```
============================
What function do you want?
(1) : Read Coils
(2) : Write multiple Coils
(3) : Read Holding Registers
(4) : Write multiple (Holding) Registers
(0) : Quit
============================
Select a function [1, 2, 3, 4, 0] :4
Enter the Start Address :0

Enter the number of Registers in you want to write on:2
Write the 0(th) register value :12
Write the 1(th) register value :22

-----<RESPONSE>-----
::Recved length is 12 BYTES
::Function - Write Multiple Registers
::Starting Address is 0!
::The number of controlled Registers in server is 2
--------------------
```

<client에서 보낸request와 받은 response>



<server에서 바뀐 register의 값들>

Client : 4번 Write multiple register를 하고자 하였다. 시작 주소를 0,(서버 상에서는 1이다)로 하여 2개의 레지스터를 write 하는 request를 보냈다. 시작주소 부터 시작하여, 값을 주었다. 즉, 시작 주소는 12, 그 다음 레지스터는 22라는 값을 쓰고자 하였다.

Server : 레지스터를 보면 1,2번 레지스터의 값이 클라이언트에서 보낸 요청대로 변했다.

Client : 서버에서 시작 주소를 0으로 하는 2개의 레지스터의 값을 바꿨다는 응답을 받았다.