[Modbus 프로토콜]

홍지우

1. 개요

[Modbus란 무엇인가?]

산업용으로 만들어진 통신 프로토콜. 주로 산업용 설비 장비와 PLC간에 통신하고 데이터를 주고 받기 위해 사용된다. 현업에서 PLC와 통신하기 위해서는 다양한 프로토콜 방식들이 제시 되고 있지만, 대부분은 Modbus 프로토콜을 사용하고 있다.
개인적인 생각으로 PLC나 IOT 연동장비라면, 필수적인 요소로 라고 생각한다.
조사하였던 PLC나 IOT 장비들의 공통점은 다양한 프로토콜을 지원하지만, 꼭 필수적으로 지원되는 프로토콜이 Modbus 였다.(그것이 RTU가 됐건, TCP가 됐건....) 그 말은 산업 장비 연동 프로토콜로 가장 대중적으로 사용되고 있다는 것이다.

[Modbus 종류]

- **Modbus RTU**

RS485, RS232, RS422 물리적 규격 회선을 사용하는 시리얼 통신 방식이며,
대부분 RS485를 사용한다고 보면 된다. 이유는 간단하다. RS232 규격의 회선은 1:1 통신만 지원된다. RS422는 전이중 방식이지만 산업현장에서 비용적으로 효율이 낮기 때문에 사용하지 않는다. 그래서 저렴하고 가장 무난한 반이중 방식의 RS485를 채택하여 사용하고 있다. 시리얼 통신을 할 때 전송속도 9600bps , 데이터 단위 8비트 , 패리티 검출 N, 정지비트 1비트로 셋팅을 대부분 많이 한다고 한다.

- **Modbus ASCII**

RTU 방식과 물리적인 규격이나 통신 방식이 시리얼로 똑같다.
회선도 시리얼통신으로 구현이 되며, RS232, RS485, RS422 규격을 사용한다. 그러나 차이점은 아스키코드의 데이터형식을 사용하여 데이터를 주고 받는다.
그것외에는 차이가 없다. 비교적으로 이 방식은 사용하지 않는다.

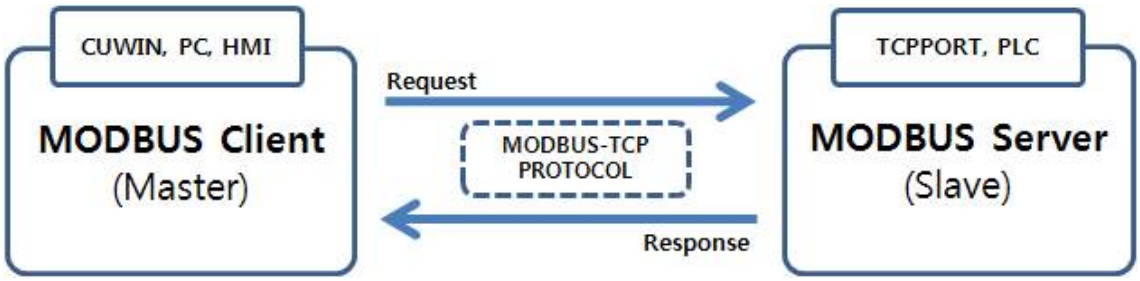
- **Modbus TCP**

TCP/IP로 데이터를 주고 받는 방식이다. 그 말은, 유선이던 무선이던 이더넷 기반의 통신을 사용 한다면 데이터를 주고 받을 수 있다는 것이다. 프로토콜 메시지 메모리 맵에서 용어나 방식에 차이가 있긴 있다. 예를들면, RTU 나 ASCII 방식에서는 마스터와 슬레이브 라고 구분되지만, TCP에서는 서버와 클라이언트로 구분된다. (**기본 TCP 포트 : 502**)

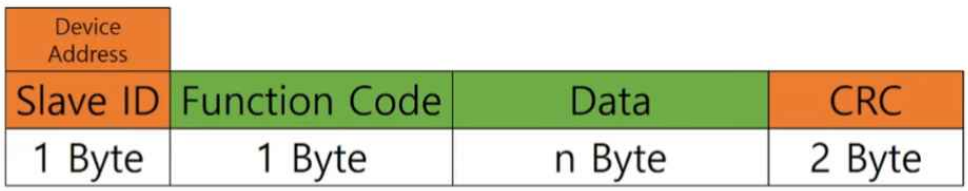
Modbus RTU와 TCP는 서로 다른 프로토콜이므로, 통신하기 위해서는 중간에 프로토콜을 변역해주는 장비가 필요하다. 현업에 종사하는 개발자 말에 따르면 이 경우가 꾀 있긴 하다고 하다. 장비 추천도 덤.(KG30X - 13만원)

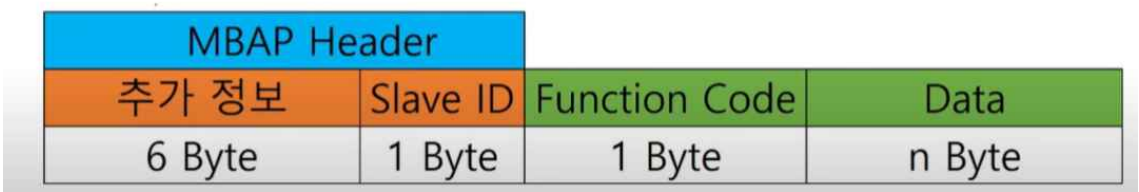
[Modbus 마스터 - 슬레이브]

Modbus는 마스터 슬레이브 개념으로 구성이 된다. 마스터 쪽에서 요청을 해야만 슬레이브에서 응답을 할 수 있는 구조다. Modbus TCP에서는 Server와 Client로 구분된다. TCP 에서는 Master를 Client, Slave를 server라고 부른다. (보통 반대로 생각할 수도 있지만 **요청을 하는 쪽이 마스터 이므로 클라이언트, 요청에 응답하는 쪽이 슬레이브 이므로 서버**의 개념이 맞다고 볼 수 있다.)



마스터에서 요청을 보낼 때, RTU 방식이건 TCP 방식이건, 필수로 같이 보내는 것이 어떤 슬레이브에게 요청을 할지에 대한 슬레이브 ID (S-ID, 국번 등 다양하게 부름)이다.
RTU 방식 프로토콜에서는 어디로 요청할지 슬레이브 ID로 구분하고, TCP방식 프로토콜에서는 IP주소로 어디로 요청을 할지 구분 한다.
또한 RTU 방식은 에러검출을 하는 패킷헤더가 있는 반면, TCP 방식은 TCP 자체가 에러검출을 수행하므로 에러검출 패킷헤더가 없다.





2. 메모리 맵 (Memory Map Table)

프로토콜이 있는 이유는 사전에 정해 놓은 규칙대로 데이터를 주고받기 위해서 이다.
Modbus에서 메모리 맵이란, 어떤 주소로 어떤 값을 주면, 어떻게 응답할지 사전에 정의 해놓은 테이블이다. 당연하게도 장비마다 기능과 속성이 다르므로, 이 메모리 맵도 다 다르다.
일종의 해당 장비의 Modbus 프로토콜을 위한 설명서라고 볼 수 있다.
만약 Modbus를 지원하는 장비라면 기본적으로 정의되어 있는 메모리 맵이 제공된다.
제공이 되어야 해당 장비의 주소에 어떤 값을 주고 받을지 알기 때문에, 꼭 제공되야한다.
장비는 상태가 바뀌면 메모리 맵에 미리 지정된 주소의 값을 변경한다.(ex. 센서 값을 저장)

[메모리 맵 테이블의 예시]

Register	Address	Name	Format
40001	0	Slot #1 DI/DO	F001
40002	1	Slot #2 DI/DO	F001
40003	2	Slot #3 DI/DO	F001
40004	3	Slot #4 DI/DO	F001
40005	4	Slot #5 DI/DO	F001
40006	5	SW #1	F002
40007	6	SW #2	F002
40008	7	SW #3	F002
40009	8	SW #4	F002
40010	9	PPA Voltage	F010
40012	11	PPB Voltage	F010
40014	13	PPC Voltage	F010

- slot#1 값이 필요하다면 40001번 레지스터 값을 읽으면 된다.
- 스위치 SW #1이 켜진 것을 알고 싶다면, 400006번 레지스터를 읽고,
  그것을 Off 하고 싶다면 같은 400006번 레지스터에 제어 값을 쓰기 하면 된다.

이와 같이 장비의 기능에 따라서 메모리 맵을 구성 한다. Modbus 프로토콜은 단지 장비의 메모리 맵에서 특정 주소의 값을 읽거나, 쓰기를 한다.
주소에 따라 용도가 다르다. 예를들면 1~9999에 해당되는 주소들은 읽기쓰기가 전부 가능한 메모리 주소이다.

[메모리 영역]

Coil : 읽기/쓰기가 가능한 메모리 1비트 단위의 boolean 데이터가 들어간다.
Input Coil : 인풋 스테이터스 라고도 하며, 읽기만 가능한 1비트 단위의 Boolean데이터이다.
Hoding Reg : 홀딩 레지스터. 읽기/쓰기가 가능한 2byte(1word)단위의 정수 형태의 데이터
Input Reg : 인풋 레지스터. 읽기만 가능한 레지스터 메모리 영역이다.

타입	블록	번호	접근	함수	용도
Bit	Coils	1~9999	Read/Write	1	Read Coil
				5	Write Single Coil
				15	Write Multiple Coils
	Discrete Input	10001~19999	Read Only	2	Read Discrete Inputs
Word (16Bit)	Input Registers	30001-39999	Read Only	4	Read Input Register
	Holding Registers	40001-49999	Read/Write	3	Read Holding Register
				6	Write Single Register
				16	Write Multiple Register

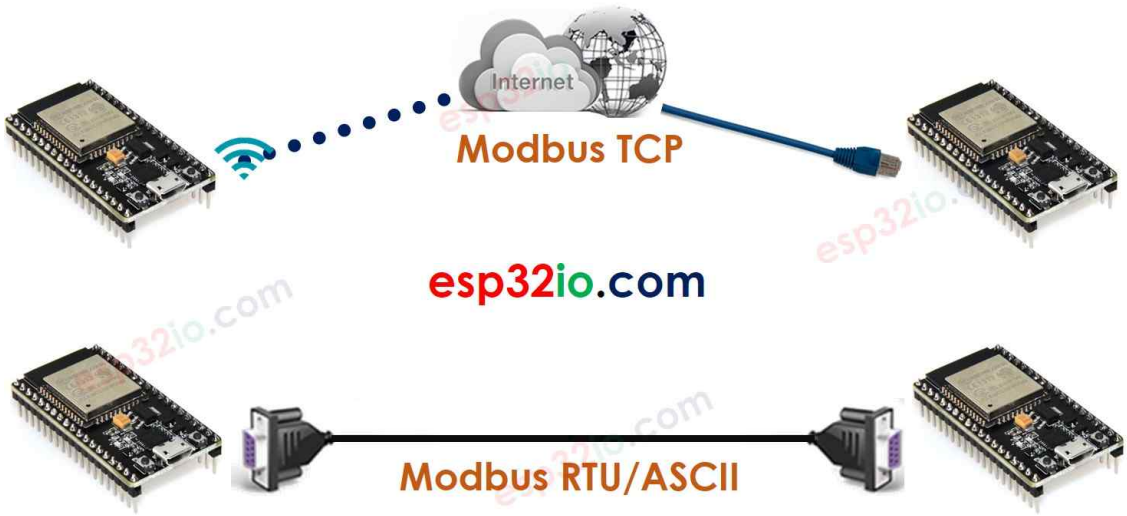
슬레이브에서는 이 메모리 영역에 따라서 메모리 맵을 구성한다.
마스터는 기능함수를 사용하여 그 메모리 영역에 접근하게 되고, 읽기와 쓰기를 할 수 있다.

[기능함수(Function Code)]
마스터에서 사용하는 기능함수는 다음과 같다.
Coil이나 holding reg 메모리 영역은 읽기와 쓰기가 모두 가능한 영역이기 때문에,
read write 기능을 제공하고
Input Coil이나 Input Reg은 읽기만 가능한 메모리이기 때문에 마스터 쪽에서는 read만 할 수 있는 기능함수를 사용한다.
그리고 얼마나, 몇 개의 코일이나 레지스터에 쓸지에 따라 싱글과 멀티로 구분이 될 수 있다.
즉, 하나의 코일이나 레지스터에 기능함수를 사용할때는 Single Coil/reg Write
여러개의 코일이나 레지스터에 기능함수를 사용할때는 Multiple Coil/reg Write
처럼 사용한다.

형 태	메모리	이 름	Function code	디바이스
비트 (Bit)	Coil	Read Discrete Inputs	02 (0x02)	INPUT, OUTPUT
		Read Coil	01 (0x01)	
		Write Single Coil	05 (0x05)	
		Write Multiple Coils	15 (0x0F)	
워드 (16비트)	Register	Read Input Register	04 (0x04)	ADC, PWM
		Read Holding Registers	03 (0x03)	
		Write Single Register	06 (0x06)	
		Write Multiple Registers	16 (0x10)	

3. Arduino 구현

가장 기본적인 동작:
Master로부터 요청을 받으면 Slave는 해당 작업을 수행, 온습도 메모리를 읽을 수 있다.

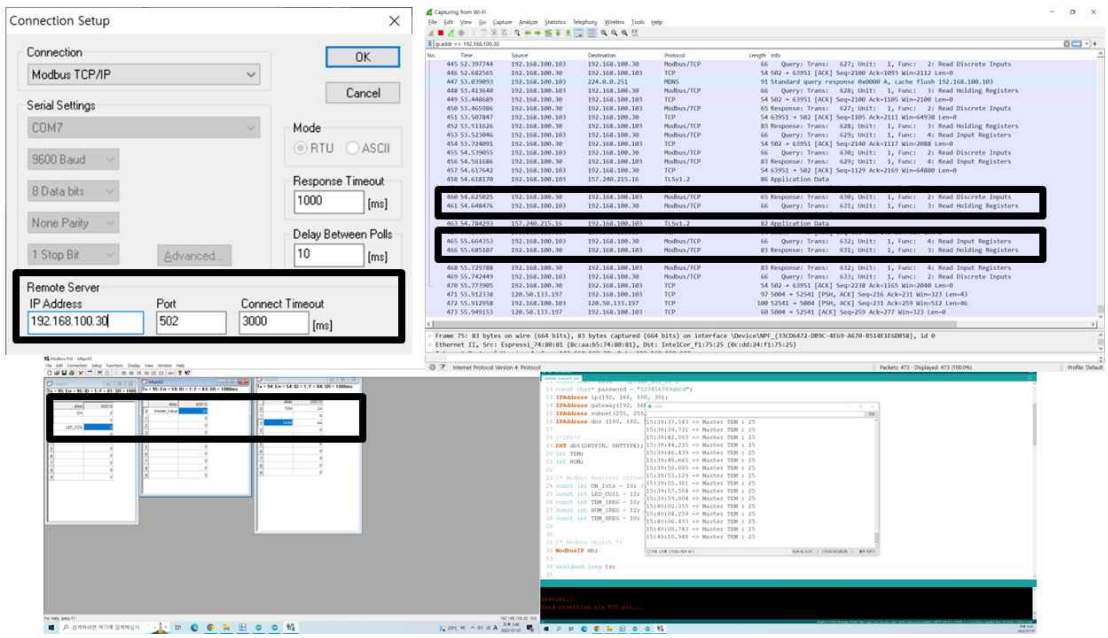


프로토콜은 Modbus TCP를 사용하였다. Modbus RTU는 물리적인 RS485 시리얼 통신규격이 필요하기 때문에 추가적인 하드웨어가 필요하다.

[Slave Memory Map]

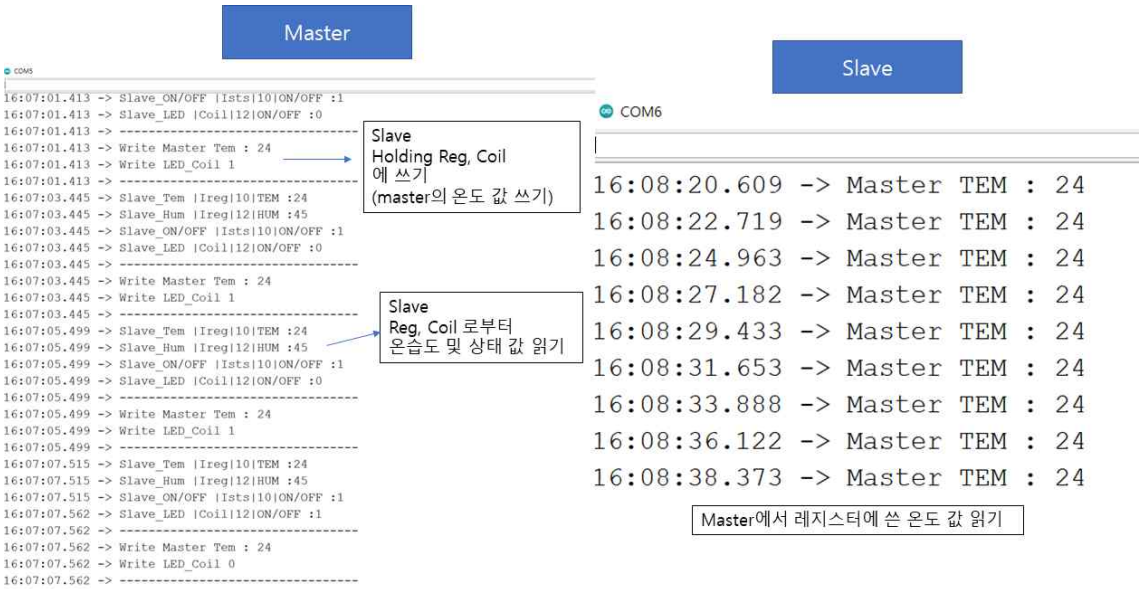
내용	메모리 영역	설정 주소	메모리 주소
온도	Input Reg	10	30010
습도	Input Reg	12	30012
마스터 온도	Holding Reg	10	40010
작동상태 LED	Input Coil	10	10010
LED	Coil	12	12

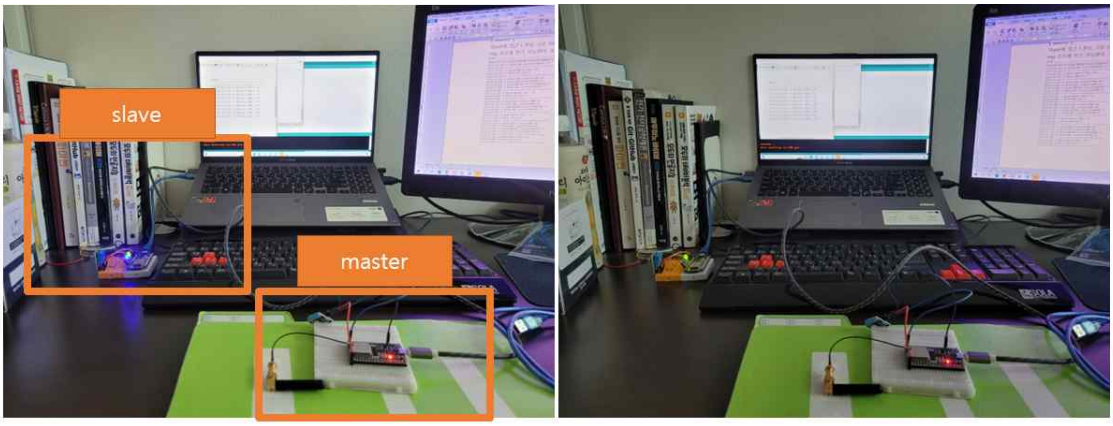
[Slave 통신 검증 : 모드버스 검증 툴 및 패킷 분석 프로그램 사용]



[Master]

Slave에 접근 (IP로 구분)하여 온도 및 습도 값을 읽고 현재 마스터의 온도 값을 holding reg 주소에 쓰기 가능하다. 또한 Slave의 LED를 제어 할 수 있다.




slave
master


master
Master: Write LED Coil ， read 온도, 습도, 상태 값 레지스터
Slave: 온도습도 상태 LED 제공, LED Coil 값 읽고 LED 기능 수행
slave