MQTT 및 JSON을 Modbus 게이트웨이로 변환하는 고급 튜토리얼

임 베 디 드　장 치　네 트 워 킹　솔 루 션

호환 제품 : Z series 제품 군

.ZFEC232485
.ZMEC232485
.ZMEC485DI
.ZWEC232485
.ZDEC485DI
.ZQEC485DI

(주)이지넷유비쿼터스

1. 제품 특징

1. MQTT 기반 프로토콜을 사용하여 서버와 연결을 설정하고, 구독 및 게시 방식으로 데이터 통신을 수행합니다.
2. Modbus RTU 레지스터의 독립적인 설계 및 자동 수집을 지원합니다.
3. 특정 Modbus 레지스터 내용을 JSON 형식으로 변환하여 정기적으로 전송하는 기능을 지원합니다.
4. 장치 ID, 시간 및 임의의 문자열을 JSON 형식에 추가할 수 있습니다.
5. JSON 형식에서 중첩 쓰기 방식을 지원합니다.
6. NTP 프로토콜을 지원하여 시간을 자동으로 가져옵니다.
7. 부호 없는 데이터와 부호 있는 데이터를 모두 지원하며, 소수점 표현 및 4바이트 길이의 데이터를 지원합니다.
8. 모든 설정은 인터페이스에서 구성 가능하며, 별도의 사용자 정의가 필요하지 않습니다.
9. MQTT 외에도 HTTP POST 및 GET 프로토콜을 지원합니다.

2. JSON 예시

2.1 Modbus RTU를 JSON으로 변환

Modbus RTU to JSON은 Modbus RTU 테이블을 자동으로 수집하여 JSON 형식으로 클라우드 서버로 자동 전송할 수 있습니다.

2.2 Modbus table

함수 코드 3, 주소 1을 가진 Modbus 테이블이 있고, 다음과 같은 레지스터 주소와 파라미터 이름이 있다고 가정해 보겠습니다. 바이트 길이가 4라는 것은 두 개의 레지스터를 연속해서 읽어야 함을 나타냅니다.

Register address	Parameter name	Byte	Remarks
0	Current total active power	4	Unsigned, 2 decimal place reserved。
97	A-phase voltage	2	Unsigned, 1 decimal place reserved
98	B-phase voltage	2	
99	C-phase voltage	2	
100	A-phase current	2	Unsigned, 2 decimal place reserved
101	B-phase current	2	
102	C-phase current	2	
119	frequency	2	
358	B phase active power	4	
360	C phase active power	4	
362	Total active power	4	

부호 있는 정수란 2바이트 또는 4바이트 데이터의 최상위 비트가 부호 비트임을 의미합니다. 예를 들어, 0xFFFF는 -1로 간주됩니다. 소수점 이하 두 자리 유지란 데이터를 정수로 변환한 후 소수점을 오른쪽 끝에서 왼쪽으로 두 자리 이동시키는 것을 의미합니다.

2.3 장치 구성

장치를 클라이언트로 구성합니다.

SocketDlgTest(http://www.zlmcu.com/document/tcp_debug_tools.html)를 사용하여 로컬 컴퓨터의 1883번 포트에서 실행되는 TCP 서버를 모니터링합니다.

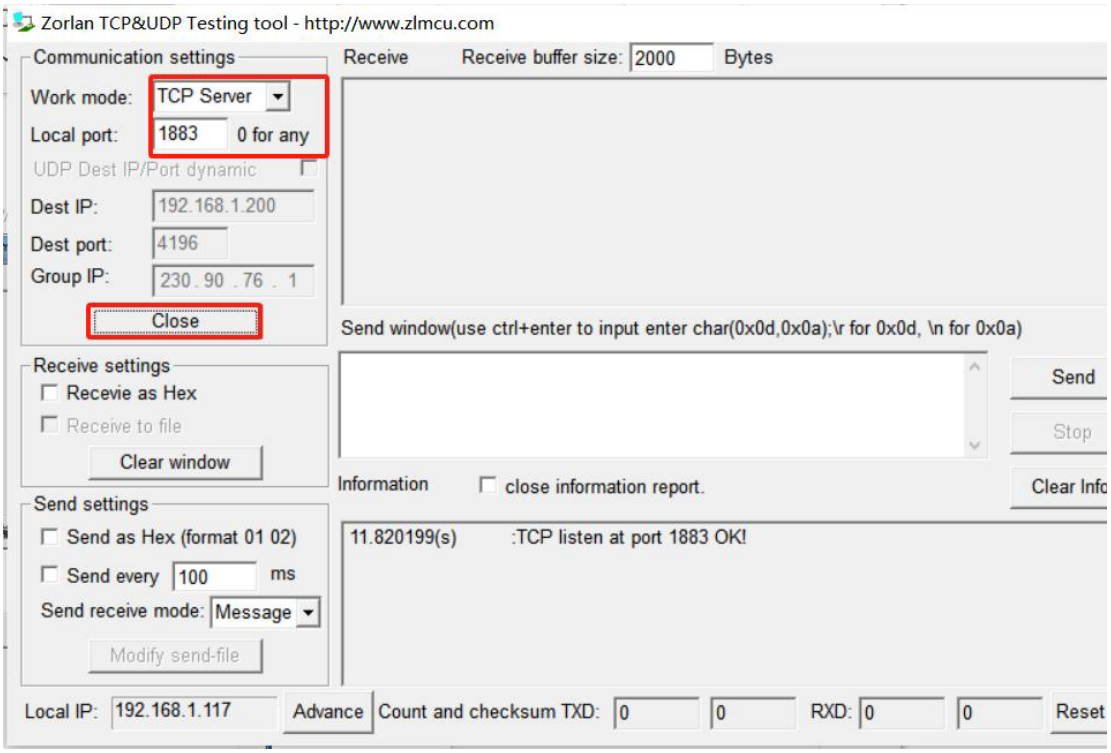


그림 1. 데이터를 수신하는 소켓 시뮬레이션 서버

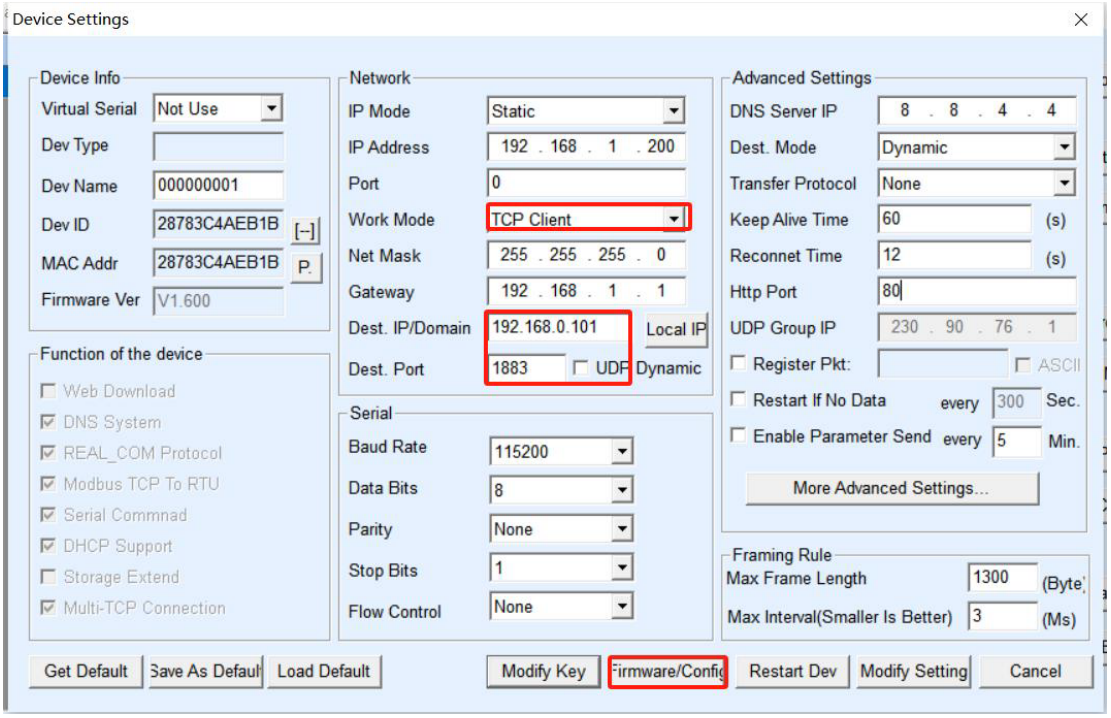


그림 2. 장치 구성

[구성 수정]을 클릭하여 장치를 SocketDlgTest 도구에 연결합니다. [장치 편집] 대화 상자를 다시 엽니다. [펌웨어 및 구성] 버튼을 클릭합니다.

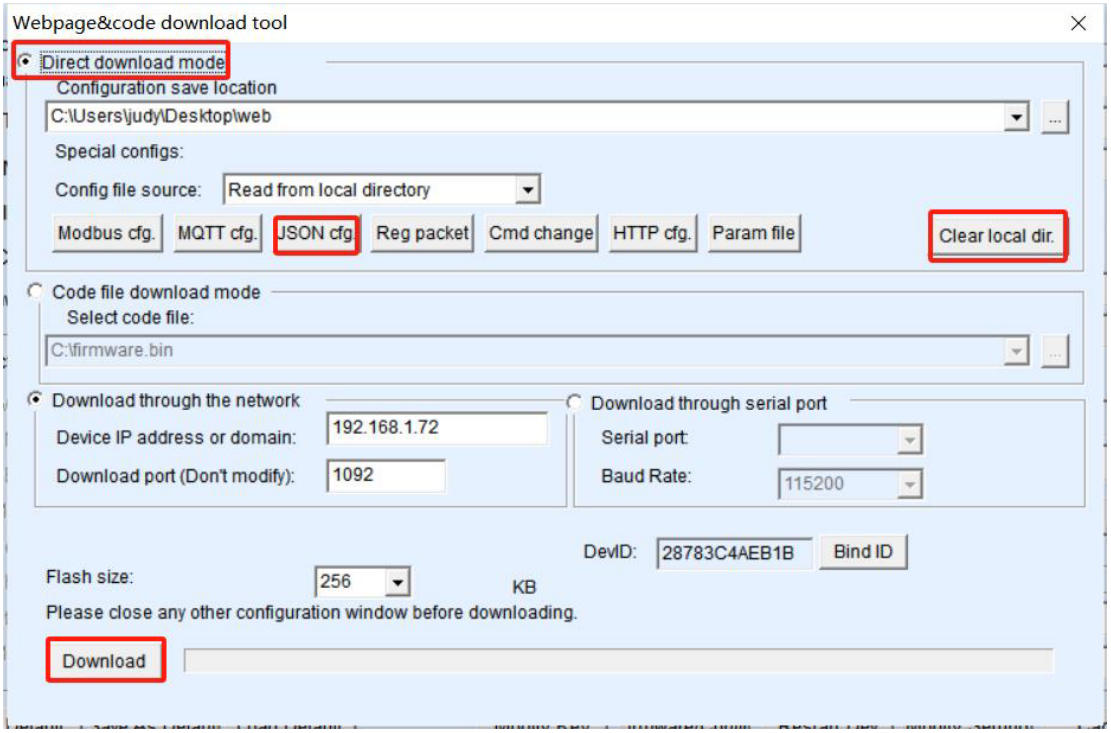


그림 3. 다운로드 인터페이스

먼저 "웹 디렉터리 다운로드"를 클릭하여 구성 다운로드 모드로 들어갑니다. 그런 다음 MQTTHTTPD 디렉터리와 같은 새 빈 디렉터리를 선택합니다. 이전 디자인 내용이 남지 않도록 하려면 먼저 "모두 지우기" 버튼을 클릭하여 이전 디자인 내용을 삭제하십시오. 디자인 파일은 이 디렉터리에 저장되며, 나중에 "다운로드" 버튼을 클릭하여 장치로 다운로드할 수 있습니다. 그다음 "JSON 구성" 버튼을 클릭합니다.

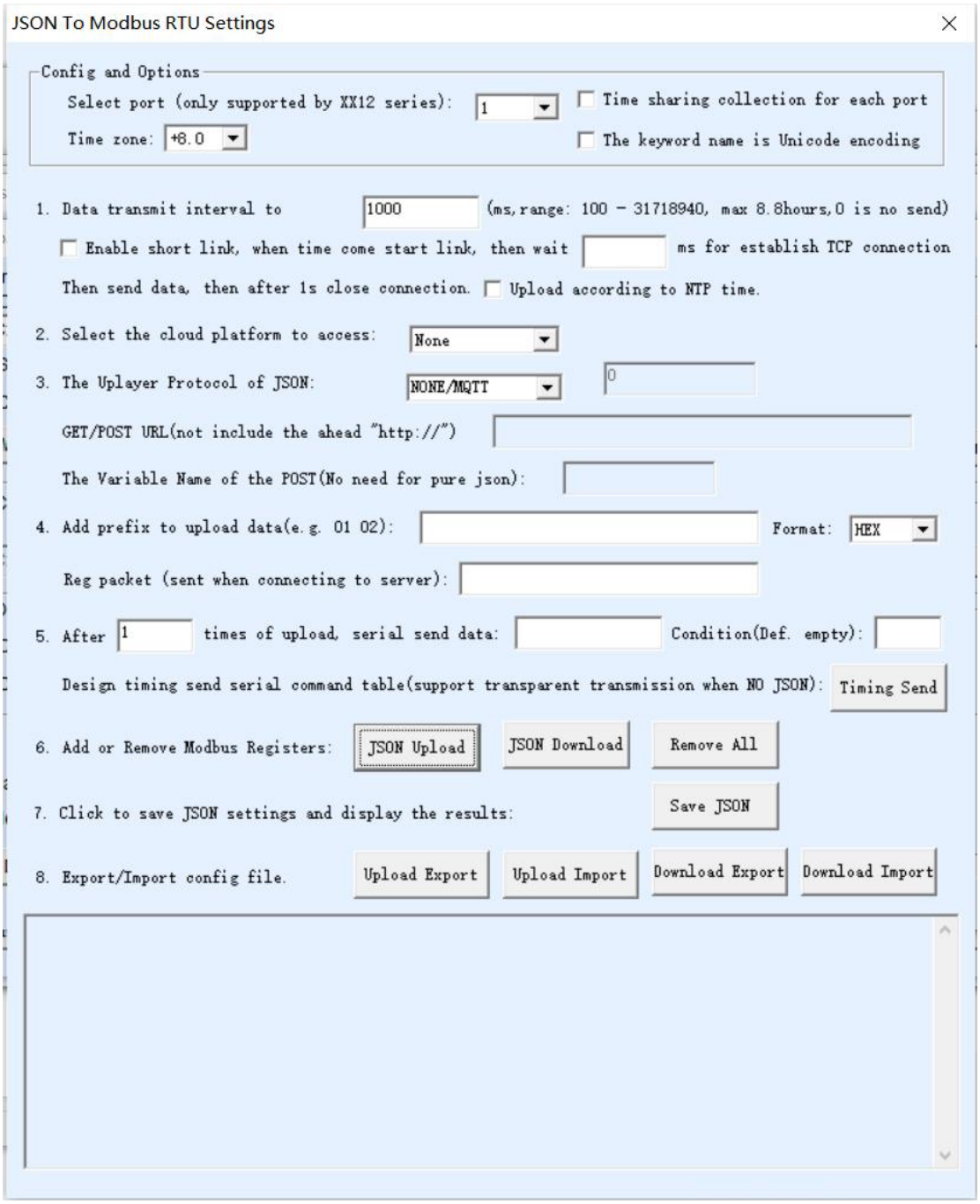


그림 4. JSON 구성 메인 인터페이스

각 매개변수는 다음과 같습니다.

1. 서버 전송 시간: 기본 JSON 데이터는 매회 서버로 전송되며, 서버 전송 시간은 장치 구성 인터페이스에서 대상 IP 주소로 설정되며 단위는 밀리초입니다.
2. 추가/보기: 이 버튼을 클릭하면 JSON 노드를 하나씩 추가할 수 있으며, 이미 추가된 내용을 볼 수도 있습니다.
3. 모두 삭제: "추가/보기" 버튼으로 추가한 모든 Modbus 레지스터를 삭제하여 처음부터 다시 설계할 수 있습니다.
4. JSON 설정 저장: 설계가 완료되면 이 버튼을 클릭하여 방금 지정한 다운로드 디렉터리에 데이터를 저장하고 장치로 다운로드할 수 있습니다.

이제 "추가/보기" 버튼을 클릭하세요. 이전 Modbus 테이블의 첫 번째 행의 경우:

Register address	Parameter name	Byte length	remark
0	Current total active power	4	Unsigned, 2 decimal places reserved.

해당 구성은 다음과 같습니다.

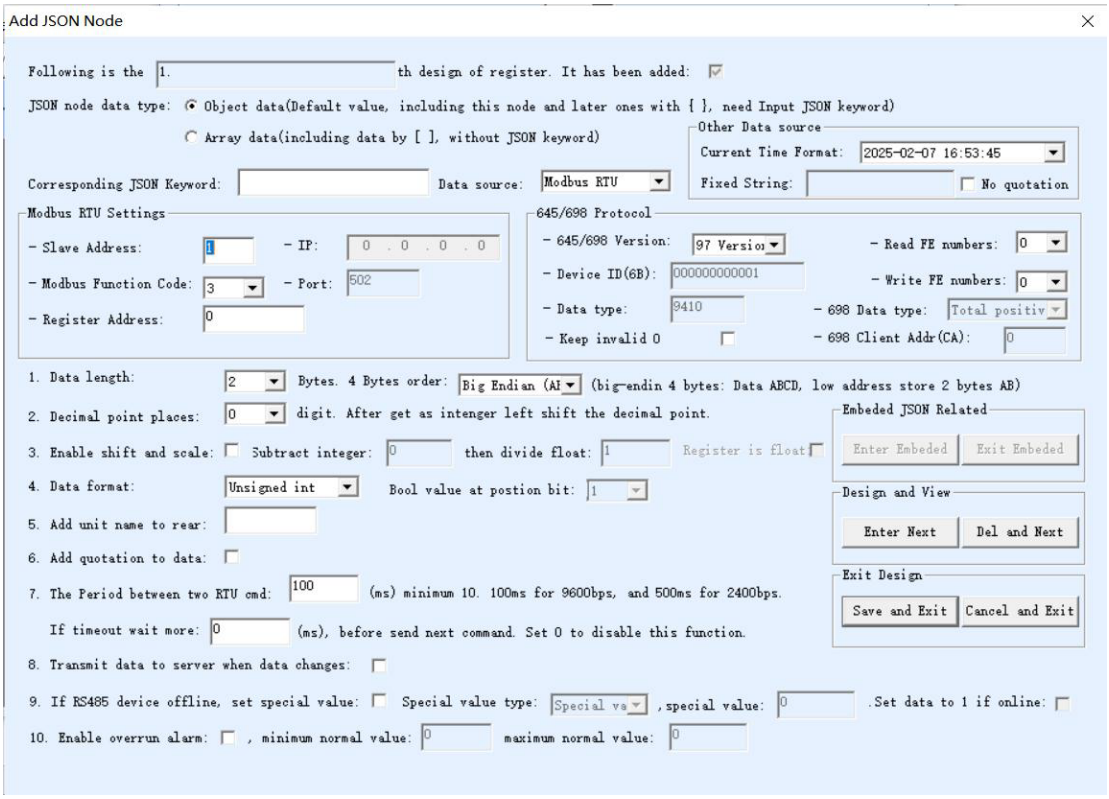


그림 5. 레지스터 설정

매개변수는 다음과 같이 설명됩니다.

① 아래 그림은 첫 번째 JSON 키워드입니다. 여기서 "1."은 디자인 인터페이스의 현재 JSON 키워드를 나타내고, 두 번째가 "2."이면 중첩된 노드 아래의 두 번째 JSON 키워드를 나타내는 "2.1" 등으로 표시됩니다.

② 추가됨: 체크 표시가 있으면 추가된 것으로 간주하며, 구성 정보를 볼 때 체크 표시가 나타나 편집 상태임을 나타냅니다. 체크 표시가 없으면 추가된 상태입니다.

③ 해당 JSON 키워드: JSON 노드의 이름입니다.

④ 데이터 소스: JSON 데이터의 소스를 선택합니다.

- 1. Modbus RTU: 예를 들어 CurrentW:123.45는 직렬 포트를 통해 Modbus RTU 테이블에서 데이터를 수집함을 나타냅니다. 그림의 왼쪽 절반은 Modbus RTU 디자인과 관련된 매개변수를 보여줍니다.
- 2. 고정 문자열: 예를 들어 DevName: "MyDev" 형식에서 오른쪽 고정 문자열에 MyDev를 입력하면 JSON 이름이 DevName이 되어 고정 문자열을 가진 JSON 노드가 생성됩니다.
- 3. 장치 ID: JSON 노드 이름이 DevID인 경우 전송되는 노드 문자열은 DevID: "285301020304"입니다. 여기서 285301020304는 장치의 MAC 주소 또는 고유 번호입니다.
- 4. 현재 시간: JSON 노드 이름이 ColletTime인 경우 전송되는 문자열은 ColletTime: "2019-05-13 22:23:31"입니다. 시스템은 NTP 프로토콜을 통해 시간을 가져옵니다.
- 5.중첩 JSON: 노드 이름이 Alarm인 경우 Alarm:{temp1: "25.1",temp2: "26.2"} 형식으로 전송됩니다. 즉, Alarm의 내용은 여전히 JSON 컬렉션입니다.

⑤ Modbus 관련 설정

- 1. 슬레이브 주소: Modbus 테이블의 주소입니다.
- 2. Modbus 기능 코드: 현재 기능 코드 03과 04가 지원됩니다.
- 3. 레지스터 주소: 여기에 해당하는 0을 입력합니다.
- 4. 데이터 길이: 4바이트입니다.
- 5. 데이터 형식: 부호 없는 정수입니다.
- 6. 소수점 유지: 소수점 이하 두 자리까지 유지합니다.
- 7. 데이터 뒤에 단위 추가: 예를 들어, "CurrentW" :25.6W인 경우, 25.6 뒤의 "W"는 큰 단위를 나타냅니다. 이 칸에 "W"를 입력합니다.
- 8. 데이터에 따옴표 추가: 이 매개변수를 선택하면 "CurrentW" :25.6W를 "CurrentW" :25.6W로 변경합니다.
- 9. 시리얼 포트 폴링 시간: 이 매개변수를 100ms로 설정합니다. 이는 이 레지스터와 다음 레지스터 사이의 폴링 간격을 나타내며, 이 명령어의 폴링 간격이 아닙니다.

⑥ 고정 문자열: 소스가 고정 문자열로 설정된 경우 문자열 내용을 입력할 수 있습니다.

⑦ 버튼

- 1. 중첩 JSON: 현재 노드 소스가 "중첩 JSON" 유형으로 선택된 경우, 이 버튼을 클릭하여 중첩 JSON 디자인으로 들어갑니다. 현재 노드가 "2"인 경우 "2.1" 노드 디자인으로 들어갑니다.
- 2. 이전 레벨로 돌아가기: 현재 노드가 N 레벨에 중첩된 경우, 이 버튼을 클릭하면 N-1 레벨의 노드 디자인으로 돌아가고 N-1 레벨의 새 노드 디자인을 유지합니다.
- 3. 다음 노드 디자인: 이 버튼을 클릭하여 다음 로컬 JSON 노드로 들어갑니다. 다음 노드가 이전 디자인에 없는 경우 "이미 추가됨" 확인란이 해제되어 새 노드임을 나타냅니다.
- 4. 디자인 저장: 디자인을 완료하려면 마지막 디자인 노드 인터페이스에서 "디자인 저장"을 클릭합니다. 그런 다음 메인 화면으로 돌아가서 "JSON 구성 저장"을 클릭합니다.
- 5. 디자인 취소: 현재 디자인을 모두 취소합니다. 디자인 내용을 보려면 이 버튼을 클릭하여 종료할 수 있습니다.

Modbus 테이블의 다른 레지스터 디자인을 계속하려면 여기에서 "다음 디자인" 버튼을 클릭하십시오. 테이블의 모든 레지스터 디자인을 완료한 후 "디자인 완료"를 클릭하고 "JSON 구성 저장"을 클릭하여 종료합니다. 그런 다음 "웹 다운로드" 페이지에서 "다운로드" 버튼을 클릭하십시오.

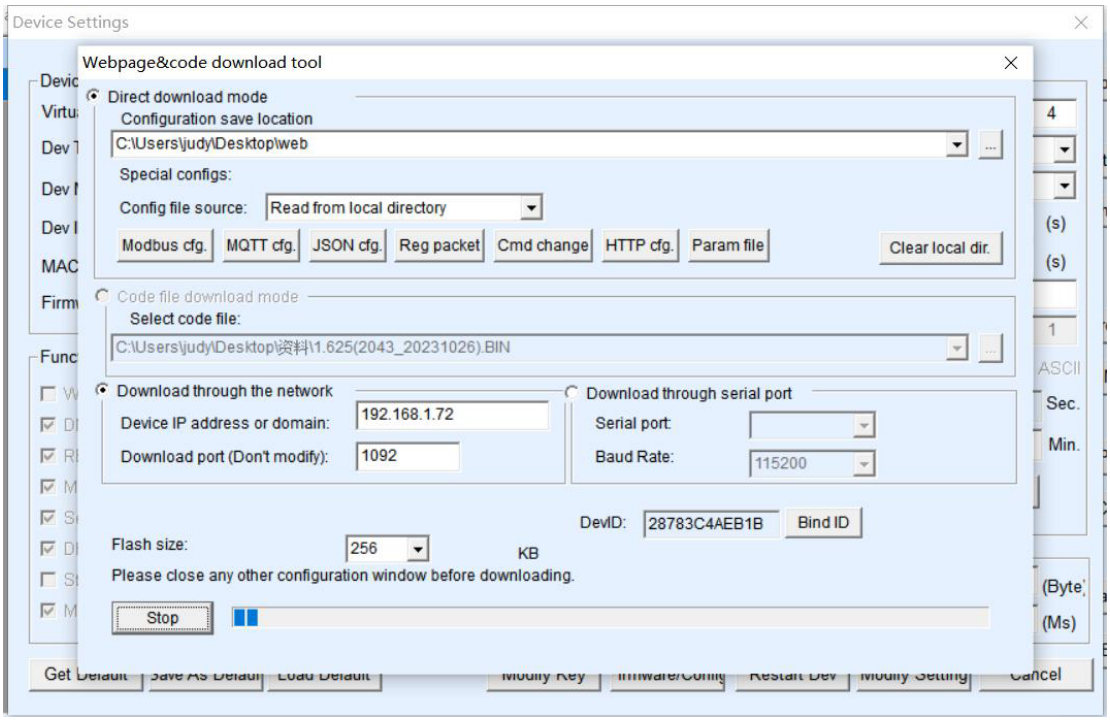


그림 6. 다운로드

그런 다음 "확인"을 클릭하면 장치가 자동으로 다시 시작됩니다. 그렇지 않으면 수동으로 다시 시작하십시오.

2.4 Modbus 아날로그 테이블을 생성

여기서 Modbus Slave는 테이블을 시뮬레이션하는 데 사용됩니다.

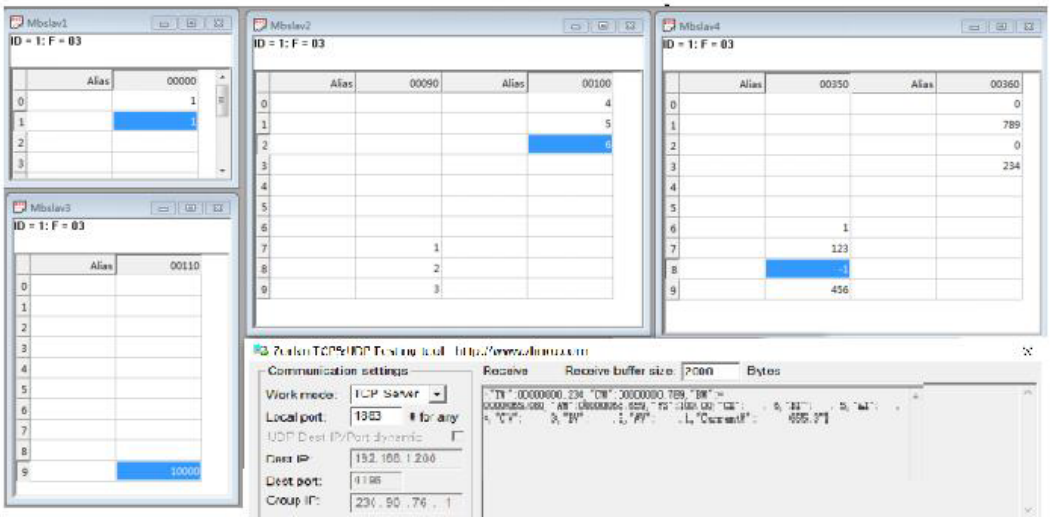


그림 7. 시험 결과

테스트 결과에 따르면 Modbus 슬레이브 툴로 시뮬레이션된 계측기의 데이터를 게이트웨이를 통해 수집할 수 있으며, 동시에 해당 데이터를 JSON 형식으로 SocketDlgTest 시뮬레이션 서버 소프트웨어에 주기적으로 전송할 수 있습니다.

3. JSON Complex 예시

3.1 NTP 활성화

시간을 포함한 JSON을 사용하려면 기기의 NTP 기능을 활성화해야 합니다. NTP 기능은 네트워크를 통해 현재 시간을 가져올 수 있습니다.

httpd.txt 파일이 있는 웹 다운로드 디렉터리에 다음 내용으로 빈 텍스트 파일을 생성하세요.

```
[NTP]
NTP_SERVER1=a1.a2.a3.a4
NTP_SERVER2=b1.b2.b3.b4
NTP_SERVER3=c1.c2.c3.c4
RE_ARUIRE_TIME=0
```

NTP_SERVER1, NTP_SERVER2, NTP_SERVER3은 NTP 시간 서버의 IP 주소입니다. 실제 상황에 따라 이 매개변수를 설정하십시오. 최대 3개의 서버를 설정할 수 있지만, 서버가 하나만 있는 경우 NTP_SERVER1에만 쓰고, 두 개만 있는 경우 NTP_SERVER1과 NTP_SERVER2 모두에 써야 합니다.

저장 후 ntp.txt 파일이 다른 파일들과 함께 기기에 다운로드됩니다.

3.2. 중첩 JSON 설계

JSON을 다음과 같이 디자인해야 한다고 가정해 보겠습니다.

```
{"header":{"DEVID":"285301020304",
                    "time":" 2019-05-13 22:23:31"},
 "data":        {"id":"MyData123456",
                 "alarm":{"alarm1":123.4C
                          "alarm2":567.8C
                         }
                },
 "value":2345
}
```

디자인 단계는 다음과 같습니다.

① 1단계에서 키워드는 "헤더"이고, 소스에서 "JSON 중첩"을 선택한 다음 "중첩 JSON 디자인" 버튼을 클릭합니다.

② 1.1단계로 이동하여 "DEVID"를 "285301020304"로 디자인합니다. 키워드는 DEVID로 입력하고, 소스에서 "장치 ID"를 선택한 다음 "다음 디자인"을 클릭합니다.

③ 1.2단계로 이동하여 "시간"을 "2019-05-13 22:23:31"로 디자인합니다. 키워드는 시간으로 입력하고, 소스에서 "현재 시간"을 선택합니다. 여기서 중요한 점은 1단계 디자인이 완료되었더라도 "다음 디자인"을 클릭한 다음 1.3단계로 이동해야 한다는 것입니다. "이전 단계로 돌아가기"를 클릭하면 1.3단계는 자동으로 건너뛰어집니다.

④ 2단계로 이동합니다. 여기에 키워드 데이터를 입력하고 소스에서 "JSON 중첩"을 선택한 다음 "중첩 JSON 디자인" 버튼을 클릭합니다.
⑤ 2.1단계로 이동하여 "id" : "MyData123456"을 디자인합니다. 키워드 id를 입력하고 소스를 고정 문자열로 선택한 다음 고정 문자열 상자에 "MyData123456"을 입력하고 "다음 디자인"을 클릭합니다.
⑥ 2.2단계로 이동하여 키워드 alarm을 입력하고 소스에서 "JSON 중첩"을 선택한 다음 "중첩 JSON 디자인" 버튼을 클릭합니다.
⑦ 2.2.1단계로 이동하여 "alarm1" :123.4C을 디자인합니다. 이는 단위가 있는 Modbus 데이터이며 기능 코드는 3이고 레지스터는 0입니다. 그러면 디자인은 다음과 같이 표시됩니다.

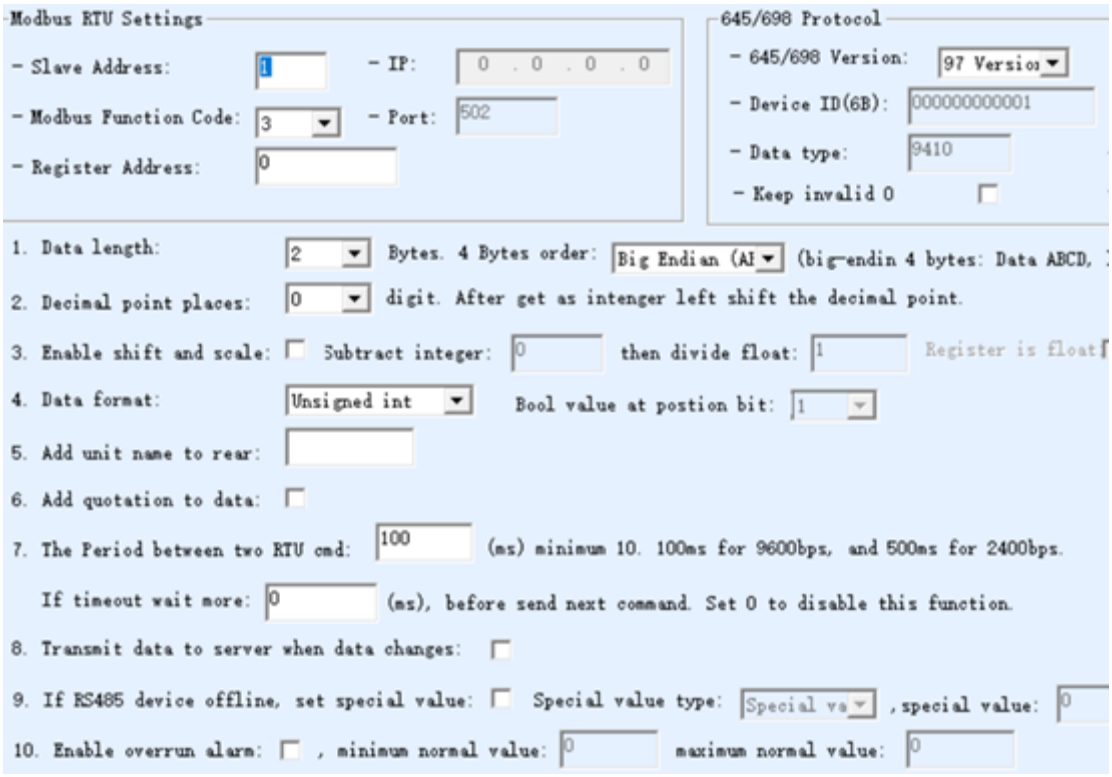


그림 8. 레지스터 디자인

그런 다음 "다음 디자인"을 클릭합니다.
⑧ 2.2.2단계로 이동하여 "alarm2"를 디자인합니다. 방법은 alarm1과 유사하며 레지스터 주소를 1로 설정합니다. 마찬가지로 먼저 "다음 디자인"을 클릭한 다음 "2.2.3" 단계에서 "이전 단계로 돌아가기"를 클릭합니다.
⑨ 2.3으로 들어갑니다. 2.3은 디자인이 필요하지 않으므로 "이전 단계로 돌아가기"를 클릭합니다. 아직 디자인되지 않은 "value" :2345가 있으므로, 그렇지 않으면 바로 "디자인 저장"을 클릭할 수 있습니다.
⑩ 세 번째 단계로 들어갑니다. 여기에는 키워드 값을 사용하는 Modbus 데이터가 디자인되어 있습니다. 이제 "디자인 저장"을 클릭합니다. 참고: 만약 "값" :2345가 여기에 없다면, 이전 단계에서 직접 저장 버튼을 클릭해야 합니다. 만약 실수로 "이전 단계로 돌아가기"를 클릭하여 세 번째 단계로 이동했는데 세 번째 단계가 존재하지 않는다면, NXT_Vircom의 새 버전에서 "삭제하고 다음으로 이동" 기능을 사용하여 이 불필요한 노드를 삭제할 수 있습니다.
⑪ JSON to Modbus RTU 설정 화면으로 돌아가서 "JSON 설정 저장"을 클릭합니다. 그런 다음 장치에 다운로드하여 사용합니다.

TCP 연결이 설정되면 다음 데이터가 수신됩니다.
{"header":{"DEVID":"2850002F0EEC","time":"2019-05-13 23:41:26"},"data":{"id":"MyData-123456","alarm":{"alarm1": 123.4C," alarm2":" 567.8C"}},"value": 2345}
참고로, 현재 JSON 디자인을 편집하는 경우 웹 다운로드 인터페이스에서 올바른 디렉토리를 먼저 선택하고 디자인 단계에 따라 버튼을 클릭하여 모든 노드를 완전히 탐색해야 합니다.

3.3. 바이트 레지스터의 비트를 읽기

Modbus에서 03/04 기능 코드를 사용하여 읽는 데이터는 특정 의미를 나타내는 비트를 사용하는 경우가 있습니다. 예를 들어, 03 기능 코드로 읽는 00 주소 레지스터는 0x8183이며, 여기서 비트16, 비트9, 비트8, 비트2, 비트1은 모두 1입니다. 이 비트들은 각각 다른 의미를 가지며, 각 비트가 1일 때는 다른 경보를 나타냅니다. 따라서 각 비트에 맞는 JSON 키워드를 사용하여 데이터를 업로드해야 합니다. 이 기능은 2003 펌웨어 1.579 이상 버전에서 사용 가능하며, zlvircom 버전 5.13 이상을 사용하여 설계되었습니다. 방법은 다음과 같습니다.

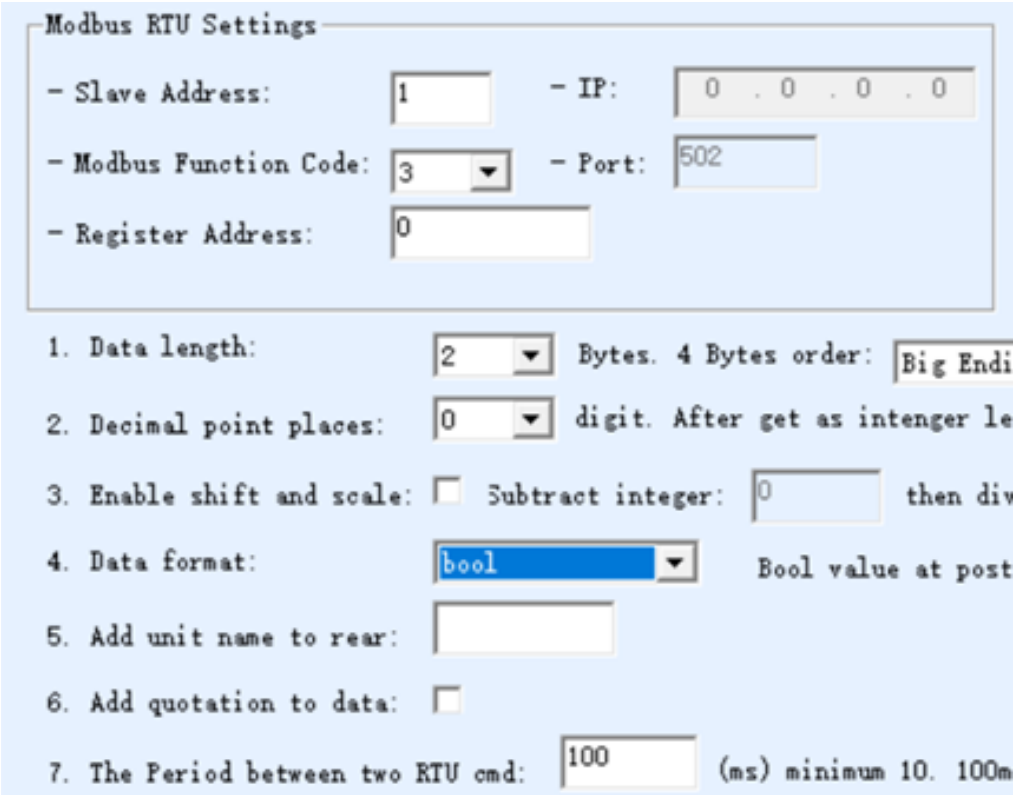


그림 9. 바이트 레지스터

설계 방법은 기본적으로 이전 방법과 동일하며, 유일한 차이점은 데이터 형식을 "Boolean"으로 선택하고, Boolean 값이 위치하는 곳에서 JSON 변수 bit2의 비트 위치(03 함수 코드)와 레지스터 00의 비트 위치를 선택한다는 것입니다. 레지스터 값이 00 20이면, 여기서 1은 2번째 위치에 해당합니다.

동일한 Modbus 스테이션 주소, 기능 코드 및 레지스터를 서로 다른 JSON 키워드에 대해 설정할 수 있으며, 부울 값의 위치만 다르면 서로 다른 변수 내용을 얻을 수 있다는 점에 유의하십시오. 예를 들어, bit1, bit2, bit8, bit9, bit16을 설계하고 레지스터 내용이 0x8183일 때 다음과 같은 JSON 데이터를 반환합니다.
{"bit1":1,"bit2":1,"bit8":1,"bit9":1,"bit16":1}

3.4 01 및 02 기능 코드

01/02 기능 코드 비트 레지스터에 대한 JSON 노드를 설정할 수 있지만, 각 JSON 노드는 레지스터 주소를 한 번만 설정해야 하므로 매번 읽는 비트 수는 1비트입니다. 바이트 레지스터의 비트는 03/04 기능 코드에서 읽는 것과 다릅니다. 03/04 기능 코드는 바이트 레지스터의 비트를 읽지만, 두 바이트 중 특정 비트의 값만 가져옵니다. 01/02 기능 코드 자체는 한 번에 1비트만 읽기 때문에 부울 값의 위치는 일반적으로 1로 기록됩니다. 1이면 '1'로 표시되고, 그렇지 않으면 '0'으로 표시됩니다.

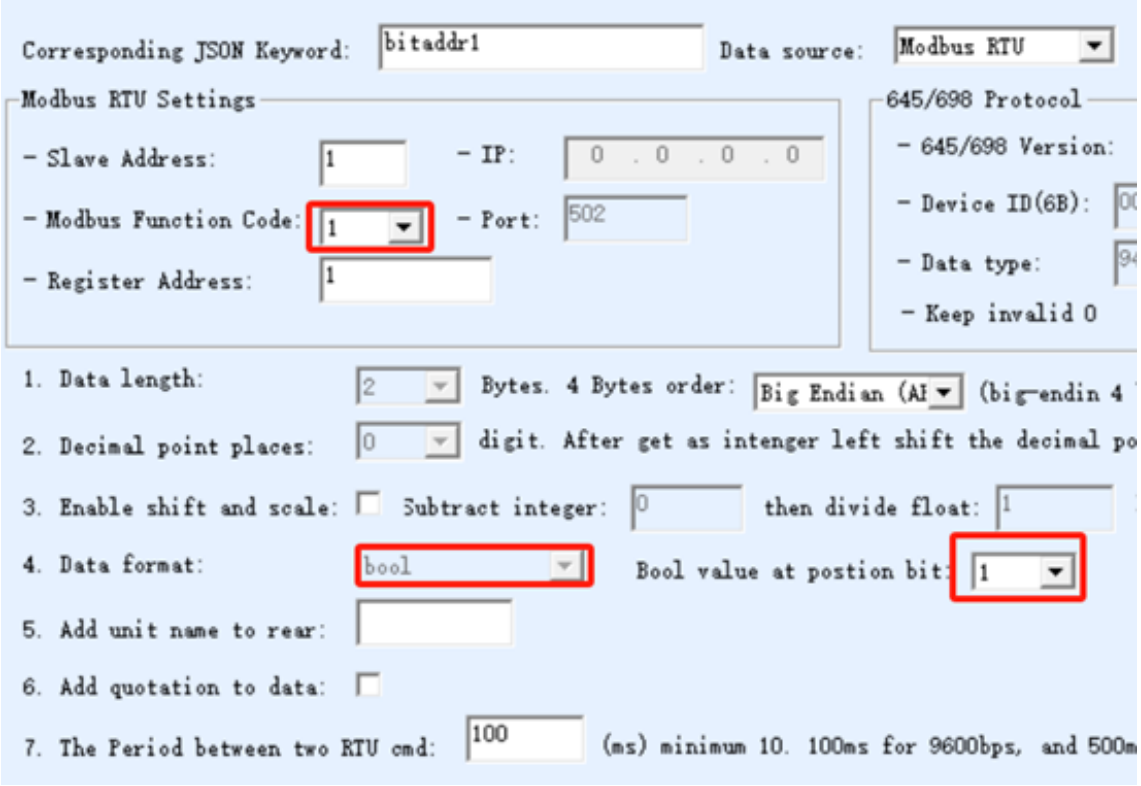


그림 10. 바이트 레지스터

Modbus 기능 코드를 01/02로 선택하면 데이터 길이, 데이터 형식 및 예약된 소수점 자리는 선택 사항이 아닙니다.

3.5 디스플레이 디자인 결과

디자인을 완료한 후 "JSON 디자인 저장"을 클릭하면 완성된 JSON 형식의 내용이 표시 상자에 나타나 디자인을 한눈에 확인할 수 있습니다. 또한 "추가/보기"를 클릭하면 비교할 수 있는 인덱스도 제공됩니다.

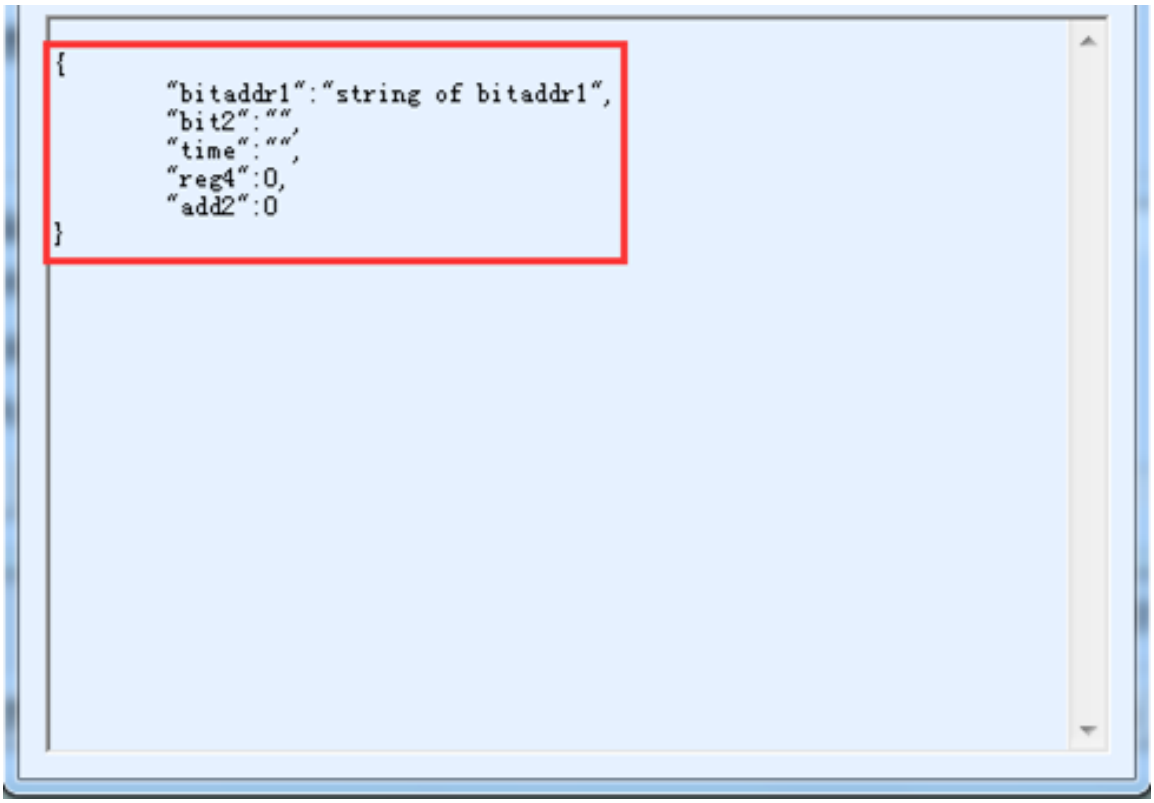


그림 11. 결과 표시

3.6 엑셀 모드에서 편집

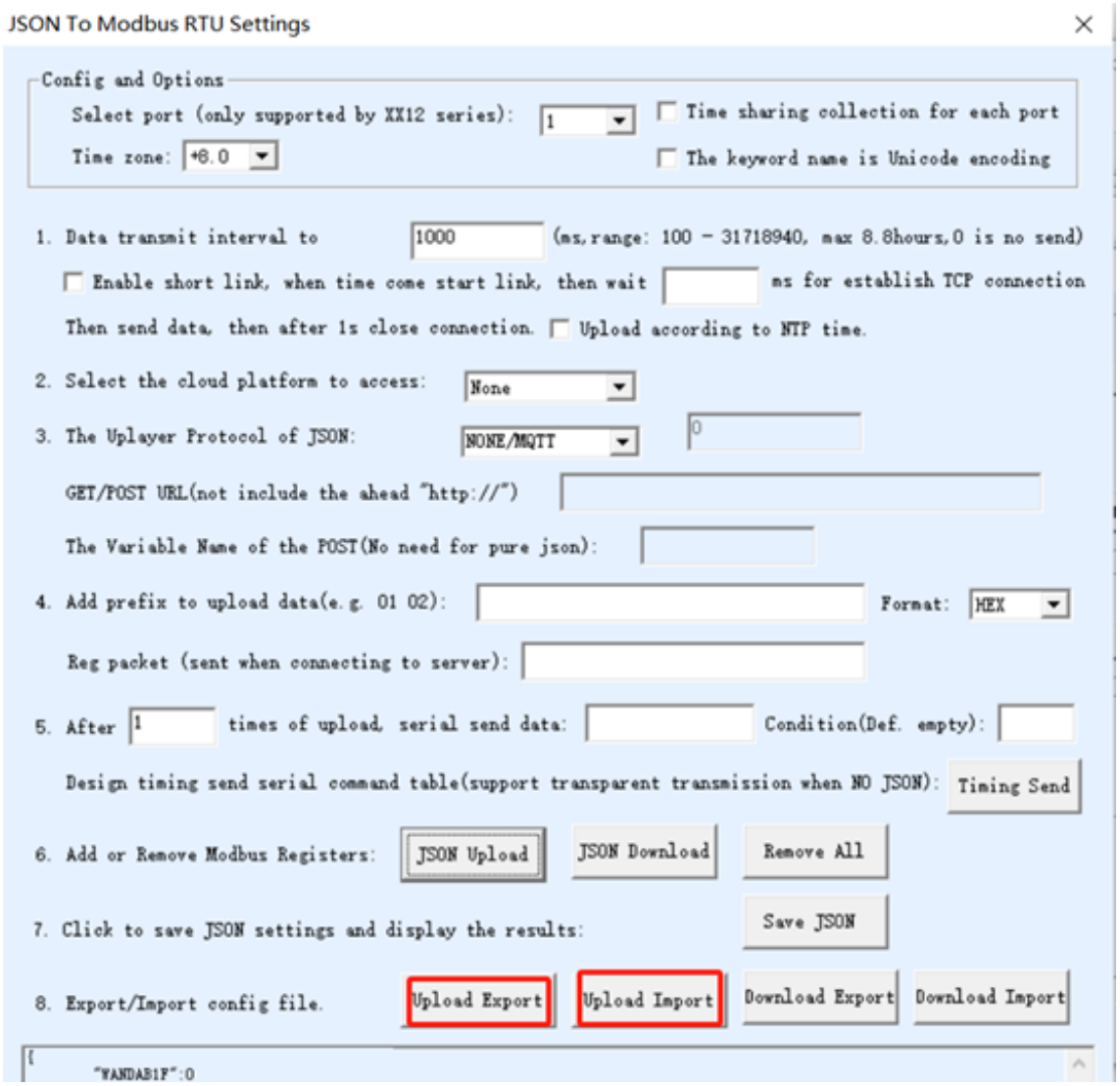


그림 12. CSV 형식 가져오기 및 내보내기

편집을 용이하게 하기 위해 디자인 내용을 CSV 형식으로 내보낸 다음, 엑셀에서 편집하고 다시 CSV 파일로 저장한 후 가져올 수 있습니다.

하지만 CSV 파일에는 형식 문제가 있습니다.

序号	JSON关键词	数据来源	固定字符串	Modbus从站	Modbus功能	Modbus寄存	645设备ID	645数据类	数据长度
1	1234	嵌套JSON		1	3	0	2.02E+11	04FF0402	1
1.1.	1235	设备ID		1	3	1	2.02E+11	04FF0402	1
1.2.	1236	当前时间		1	3	2	2.02E+11	04FF0402	1
1.3.	1237	嵌套JSON		1	3	3	2.02E+11	04FF0402	1
1.3.1.	1238	645协议		1	3	4	2.02E+11	2010100	2

그림 13. 데이터 형식 오류

이 경우 CSV 파일을 XLS 형식으로 저장하여 편집할 수 있습니다. 편집 후 다시 CSV 형식으로 저장하고 가져오세요.

3.7 대괄호와 배열

NXT_Vircom의 새 버전은 JSON 배열 디자인을 지원합니다. 다음은 간단한 예입니다.

```
{
        "reqBody":
        [
                0,1
        ]
}
```

이 예제에서는 reqBody라는 JSON 객체 하나만 있으며, 그 내용은 첫 번째 요소가 data 0이고 두 번째 요소가 data 1인 배열입니다. 여기서는 대괄호로 묶인 배열을 중첩된 JSON처럼 취급하지만, 중첩된 JSON은 키워드 이름과 콜론을 필요로 하지 않습니다. 설계 단계는 다음과 같습니다.

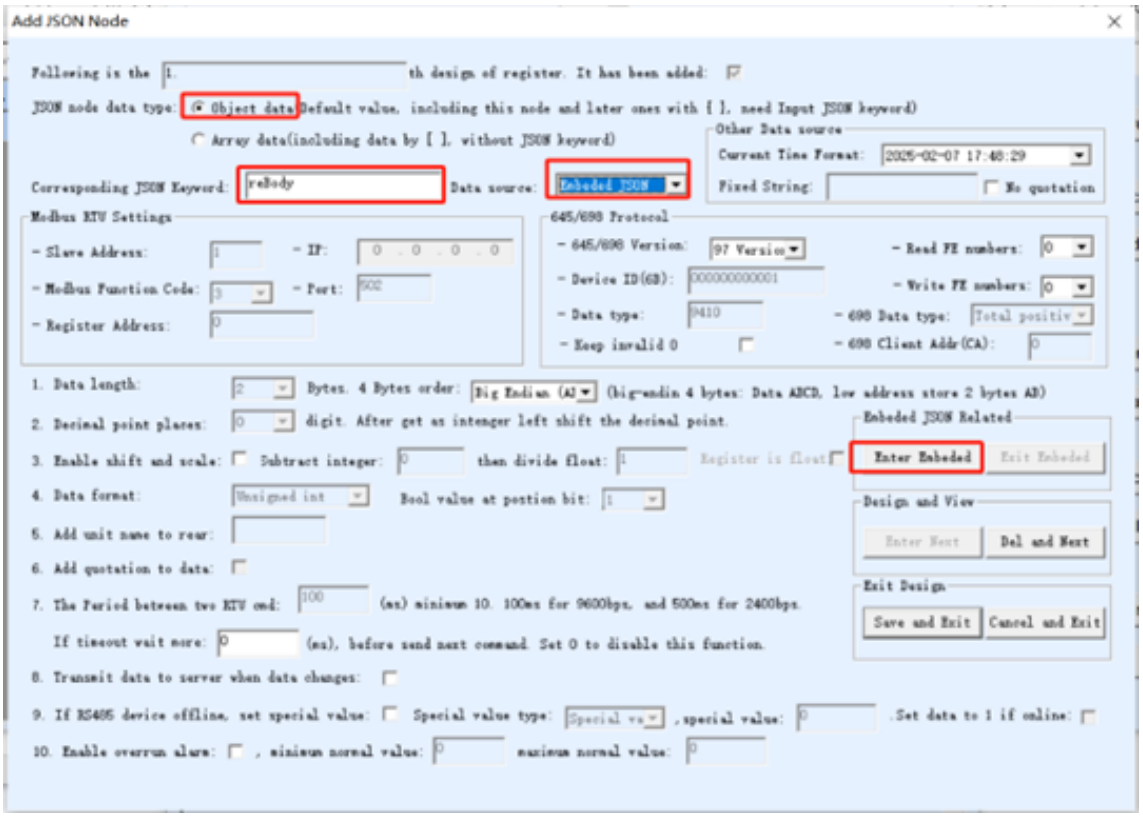


그림 14. 배열 객체

reqBody 자체가 배열의 단위가 아닌 객체이므로 노드 유형은 배열 데이터가 아닌 객체 데이터용으로 선택됩니다. 내용이 배열이므로 이 reqBody의 데이터 소스는 대괄호를 중괄호로 간주하여 "중첩 JSON"으로 설정합니다. 그런 다음 "중첩 JSON 디자인"을 클릭합니다. 대괄호가 있는 모든 곳에서 데이터 소스를 "중첩 JSON"으로 선택해야 하며, 노드 유형이 "객체 데이터"인지 "배열 데이터"인지는 해당 요소가 배열의 단위(배열 요소)인지에 따라 결정됩니다.

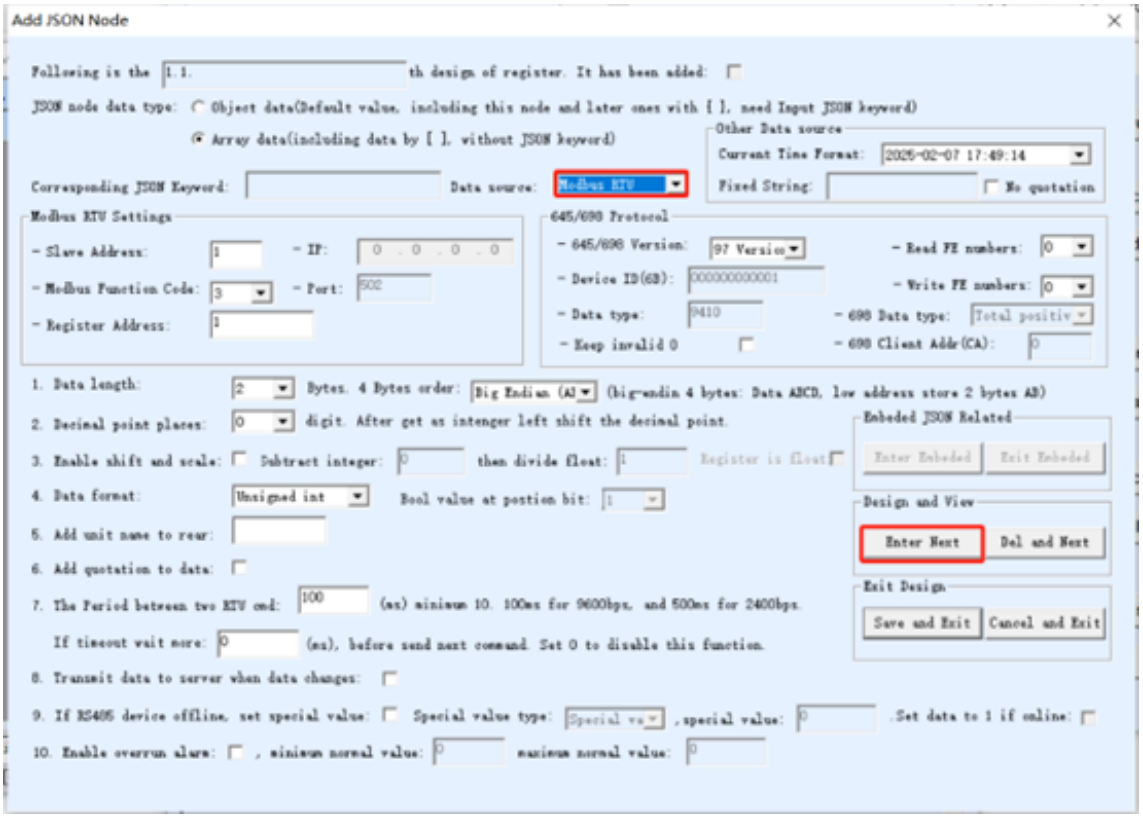


그림 15. 배열 내용

배열의 첫 번째 요소부터 디자인해 보겠습니다. 배열의 내용이기 때문입니다. 따라서 노드 유형은 배열 데이터입니다. 데이터 소스는 Modbus RTU 컬렉션이며, Modbus 관련 매개변수를 입력합니다. 그런 다음 "다음으로 이동"을 클릭하여 같은 방식으로 배열의 두 번째 요소를 디자인합니다. 두 번째 요소 디자인이 완료되면 더 이상 디자인할 것이 없으므로 "모두 저장하고 종료"를 클릭합니다. 이전 화면으로 돌아가서 "JSON 설정 저장"을 클릭한 다음 다운로드합니다. 전송된 데이터는 다음과 같습니다.

{"reqBody":[2,3]}.

이제 좀 더 복잡한 예제를 살펴보겠습니다.

```
{
        "reqBody":
        [
                {
                        "workshop_id:"1008",
                        "machine_code":"XS114"
                },
                {
                        "workshop_id:"1008",
                        "machine_code":"XS116"
                }
        ]
}
```

이 경우 배열의 내용은 단순히 데이터 값이 아니라 JSON 그 자체입니다.

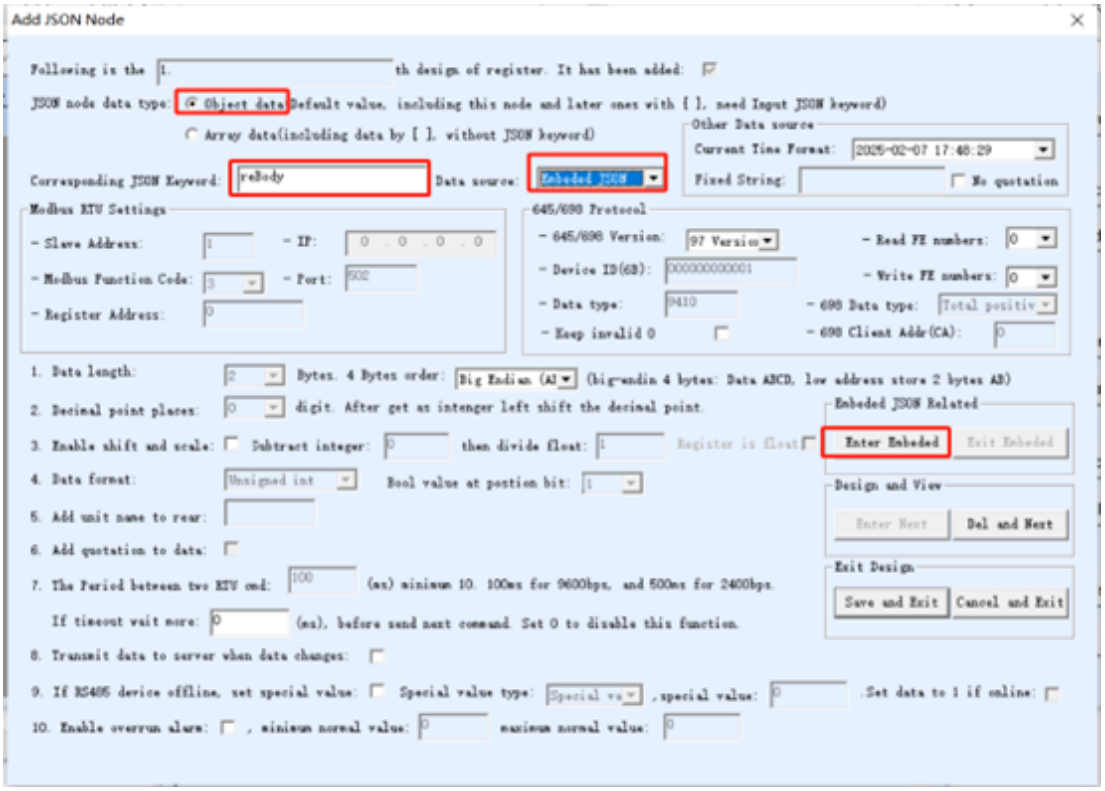


그림 16. 1단계

여기 1.1에서 이전 예제 1은 노드 유형을 "배열 데이터"로 선택한 다음 Modbus 형식으로 데이터를 직접 디자인하는 것이었지만, 여기서는 데이터 내용이 JSON 객체이므로 데이터 소스를 "중첩 JSON"으로 선택한 다음 "중첩 JSON 디자인"을 클릭해야 합니다.

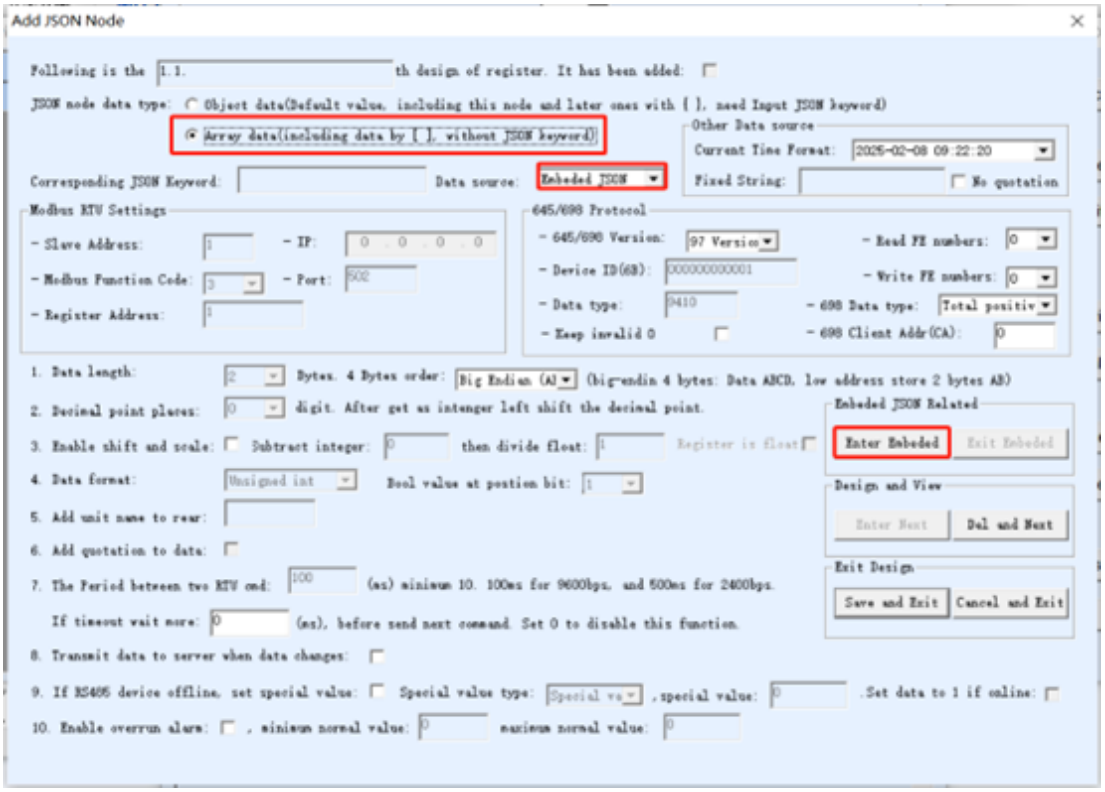


그림 17. 1-1단계

그런 다음 1.1.1단계와 1.1.2단계에서 Modbus 소스 데이터 workshop_id와 고정 문자열 소스 데이터 machine_code라는 두 가지 객체 유형을 설계합니다.

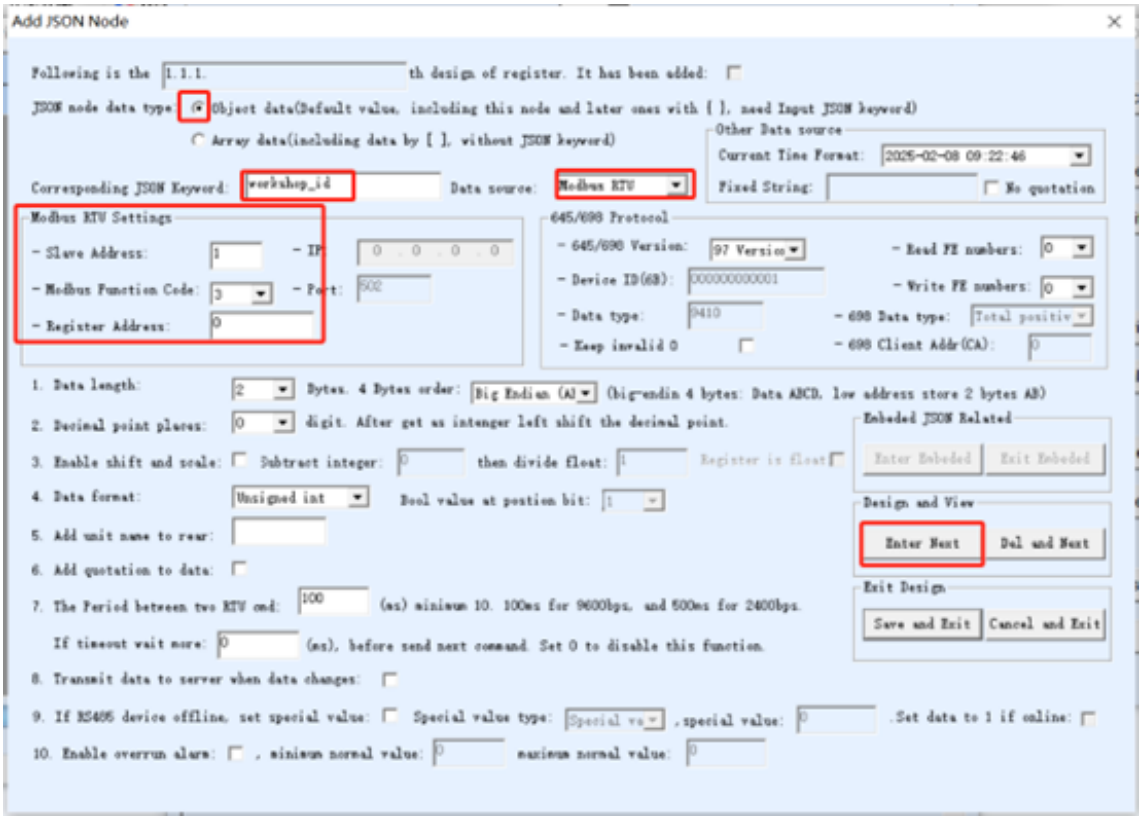


그림 18. 1-1-1단계

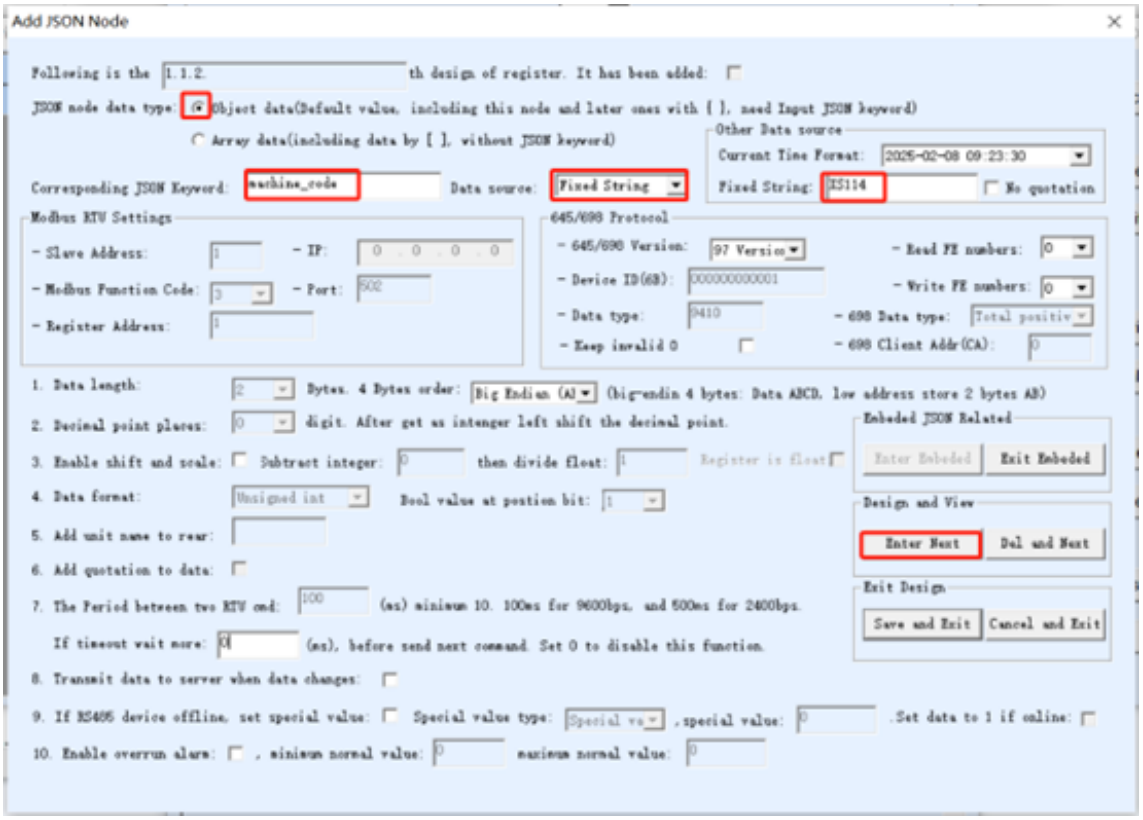


그림 19. 1-1-2단계

여기서 "다음으로 이동"을 클릭해야 합니다. 이 빈 콘텐츠 노드(즉, 1.1.3)에서 "이전 레벨로 돌아가기"를 클릭하여 "1.2"로 이동합니다. 여기서 1.1.3은 실제로 존재하지 않는 노드입니다.

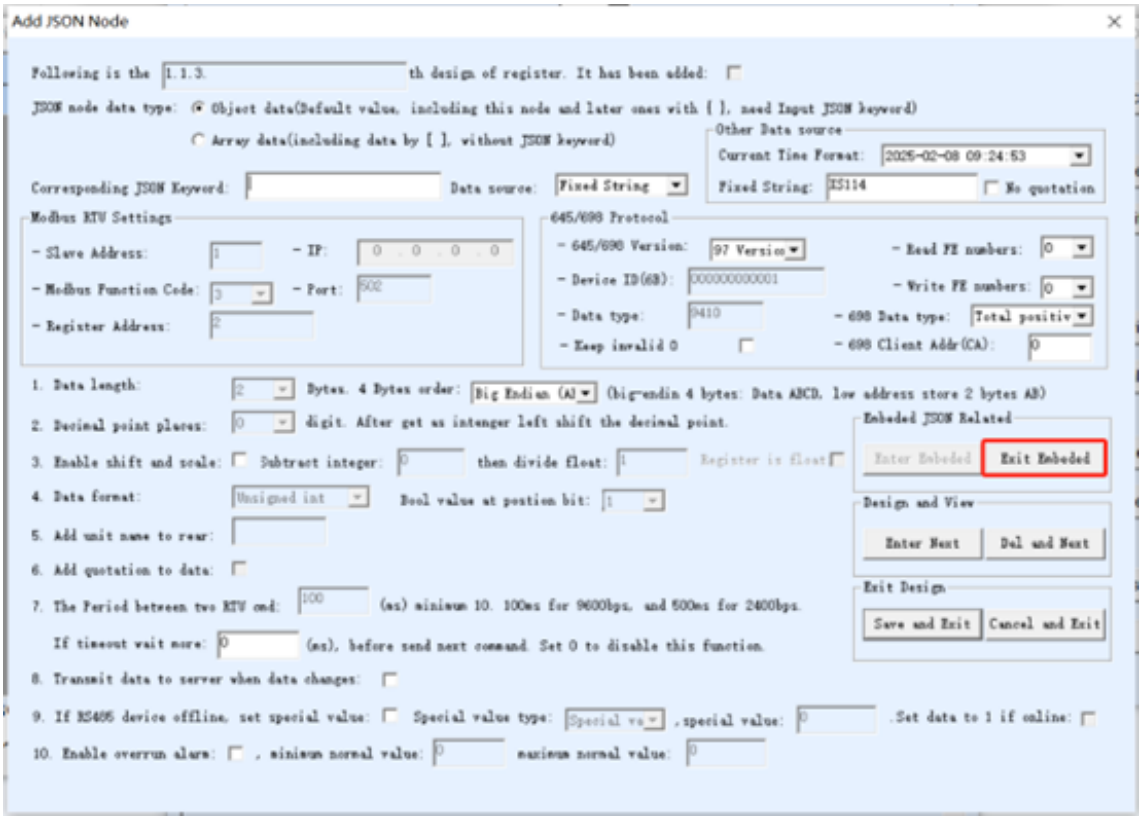


그림20. 1-1-3단계

"1.2"도 배열 유형이며, 데이터 소스는 중첩 JSON입니다. 1.1 단계를 참조하여 1. 배열 데이터, 2. 중첩 JSON을 선택하는 데 주의하십시오. "1.2.2"까지 이동하여 "모두 저장하고 종료"를 클릭합니다.

마지막으로 업로드된 데이터 형식은 다음과 같습니다. 여기서 workshop_id는 Modbus 레지스터에서 읽어온 값입니다.

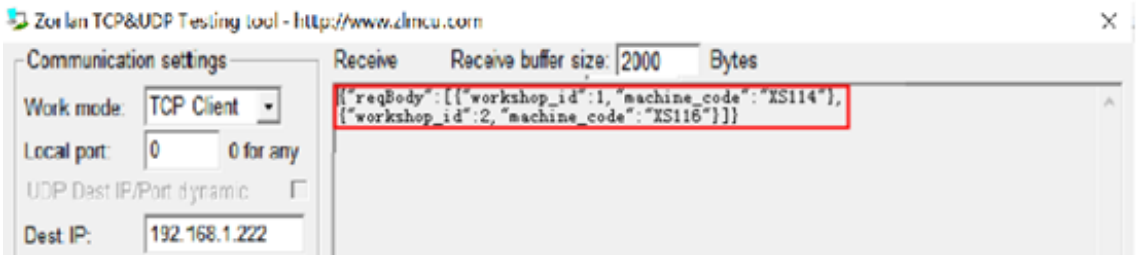


그림21. 1-1-3단계

3.8 Little endian

이 글의 나머지 부분을 참조하십시오.

3.9 읽기 실패로 인해 삭제되지 않았습니다.

레지스터를 읽을 수 없는 경우, 데이터 값 0을 사용하여 데이터가 읽히지 않았음을 나타낼 수 있습니다.

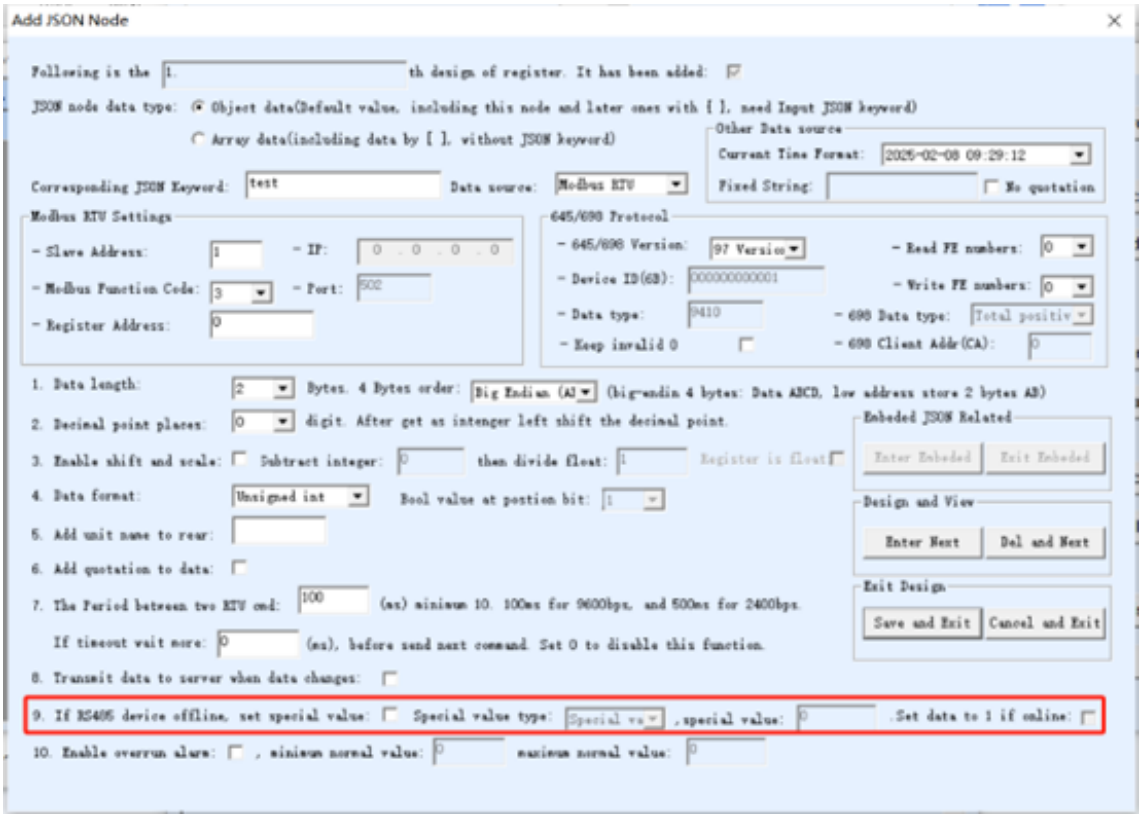


그림22. 데이터 삭제

여기서는 RS485 장치의 오프라인 데이터 삭제 기능을 확인하는 테스트 JSON을 설계합니다. 레지스터를 읽을 수 있는 경우, 전송되는 데이터는 {"test":123}이며, 여기서 123은 실제 레지스터 내용입니다. 장치가 오프라인 상태이거나 데이터를 읽을 수 없는 경우, {"test":0}이 전송됩니다. 이렇게 하면 장치가 오프라인 상태인데도 데이터가 남아 있다는 오해를 방지할 수 있습니다.

3.10 기기가 오프라인 상태입니다.

읽은 데이터가 없으면 데이터는 0입니다. 경우에 따라 장치가 오프라인인지 감지할 수 없는 경우가 있습니다. 예를 들어 데이터 내용 자체가 0인 경우 장치가 오프라인인지 또는 데이터가 0인지 판단할 수 없습니다. 또한 장치에 읽어야 할 레지스터가 여러 개 있는 경우, 장치가 온라인 상태임을 나타내기 위해 {online1:0}과 같은 별도의 JSON 키워드를 0 또는 1로 설정해야 합니다. 이를 위해 online1을 다음과 같이 설계할 수 있습니다. 먼저 모든 레지스터를 설계한 다음 JSON 노드를 추가합니다.

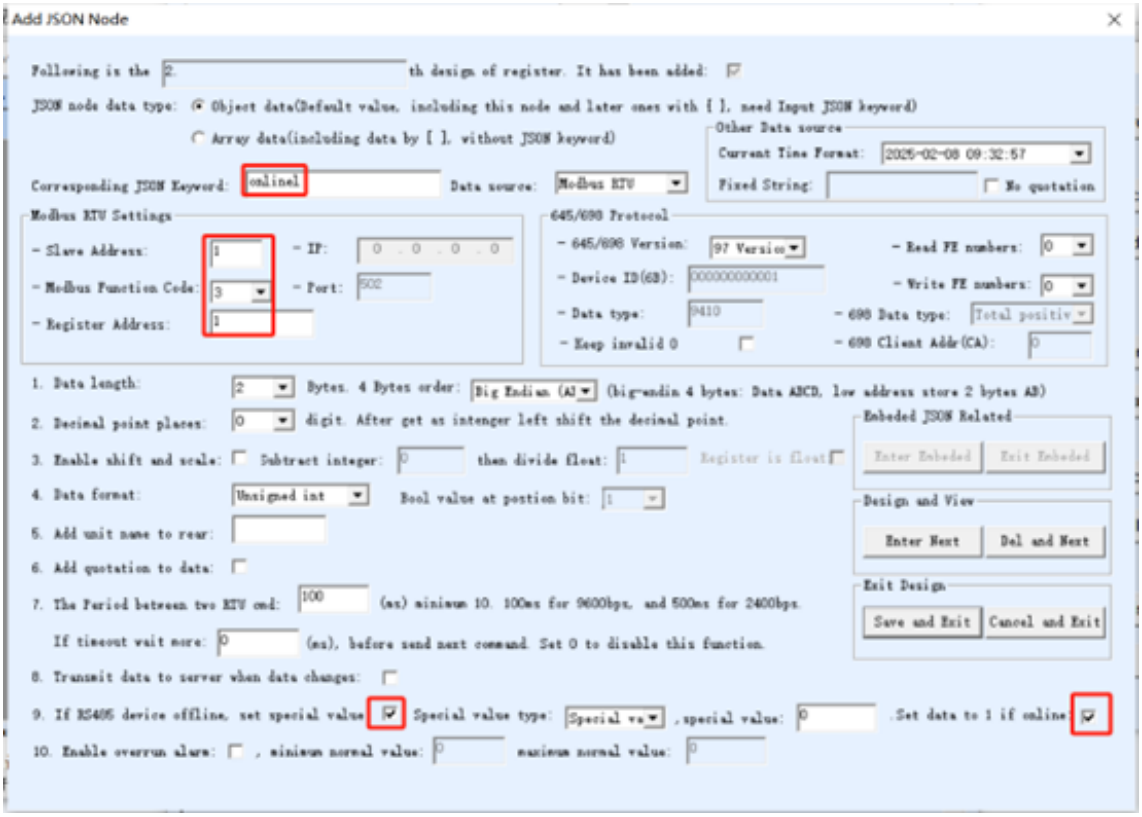


그림 23. 온라인 기기

이 경우 이름은 online1이지만 원하는 이름을 선택할 수 있습니다. 등록 번호는 기존 등록 번호 중 아무거나 선택해도 됩니다. 가장 중요한 것은 RS485 장치의 오프라인 데이터가 0인지 확인하고, 장치가 온라인 상태이면 등록 내용과 관계없이 강제로 1로 설정하는 것입니다. 이렇게 하면 읽은 데이터가 0이더라도 1로만 인식됩니다. 즉, online1에는 0과 1의 데이터만 저장됩니다. 1은 온라인, 0은 오프라인을 나타냅니다.

3.11 시프트 및 스케일(비트 이동 및 배율 조정)

변환 및 스케일링은 Modbus 레지스터에서 읽은 데이터 내용에서 2바이트 부호 있는 정수를 뺀 다음, 단정밀도 부동 소수점 숫자로 나누어 단정밀도 부동 소수점 숫자를 얻는 과정입니다(아래 참조).

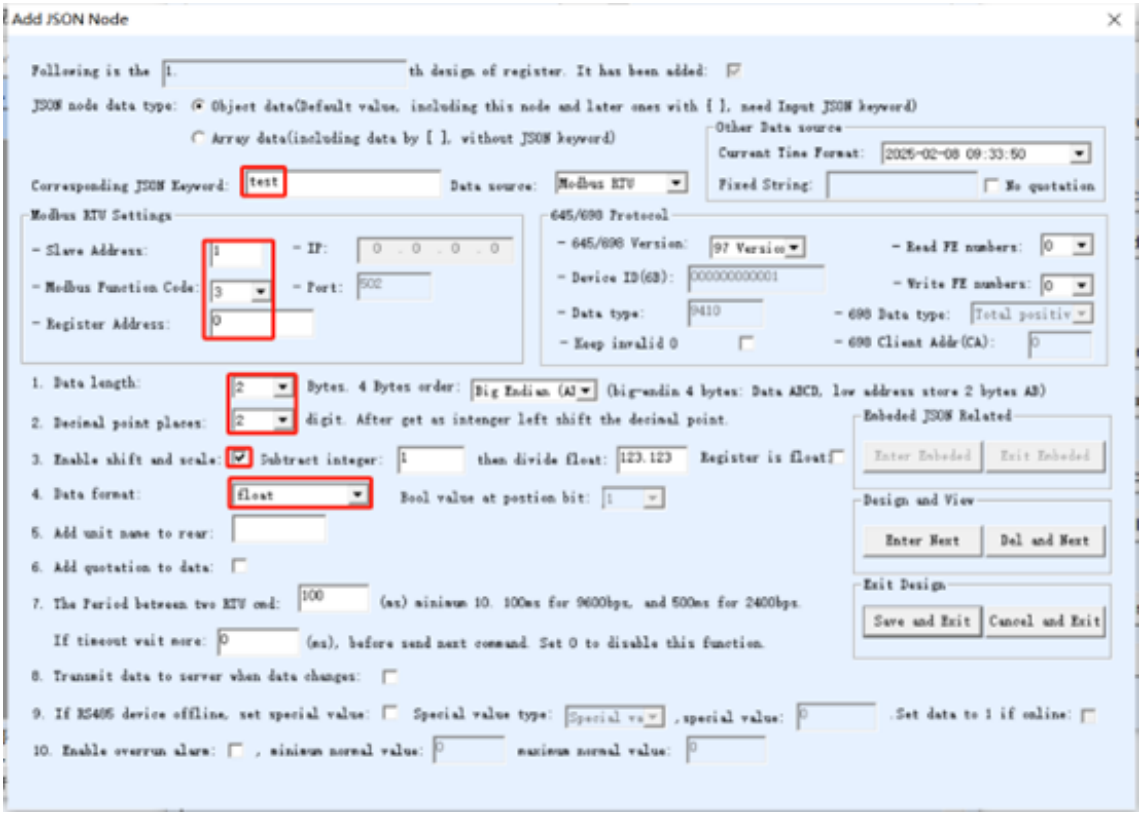


그림 24. 패닝 및 확대/축소

그림과 같이 JSON 키워드 테스트를 설계합니다. 스테이션 주소 1, 레지스터 1의 레지스터 0(2바이트) 값을 읽은 다음 1을 빼고 123.123으로 나눕니다. 이는 레지스터에서 읽은 데이터를 2칸 아래로 시프트한 다음 123.123만큼 축소하는 것과 같습니다. 레지스터 내용이 124일 때 전송되는 데이터는 {"test":0.99 "}이며, (124-1)/123.123=0.99입니다.

Modbus RTU 데이터 소스, 2/4바이트 데이터 길이, 03 또는 04 기능 코드, 부동 소수점 출력만 지원됩니다. 소수점 자릿수는 0에서 4자리까지 가능합니다. 가장 중요한 것은 "패닝 및 확대/축소 활성화" 옵션을 확인하는 것입니다. "뺄셈 정수" 필드에는 -32768에서 32767 사이의 부호 있는 정수를 입력하고, "나누기 부동 소수점" 필드에는 부동 소수점 숫자를 입력합니다.

설계에서 하나의 데이터라도 변환 및 스케일링이 필요한 경우, 다른 모든 데이터도 변환 및 스케일링이 필요하지만, 스케일링 비율과 변환 비율은 자유롭게 설정할 수 있습니다. 스케일링이 필요 없는 데이터는 0을 뺀 다음 1.00001로 스케일링하도록 설정할 수 있습니다. 1.00001이 아닌 1.00001로 설정하는 이유는 다음과 같습니다. 값이 1.0으로 설정되면, 스케일링 없이 2바이트 데이터에 대해 부동 소수점 형식을 선택하는 것이 불가능하다고 자동으로 가정합니다. 123과 같은 정수 값의 경우, 스케일링을 적용하면 변환은 0이 되고 나누는 값은 0.99999가 됩니다. 따라서 결과값은 소수점 이하 자릿수가 0이므로 정확도에는 영향을 미치지 않고 여전히 123이 됩니다. 그러므로 변환 스케일링을 증가시켜도 정수에는 아무런 영향을 미치지 않습니다.

다양한 데이터 형식 지원:
① 2바이트 셰이핑 데이터의 변환 및 스케일링을 지원합니다.
② 4바이트 셰이핑 데이터의 변환 및 스케일링을 지원합니다. ABCD, CDAB, 두 개의 Little Endian 및 Little Endian 형식을 지원합니다. 이전 버전에서는 바이트 스왑을 지원하지 않습니다. 최신 버전에서는 "변환 스케일링 매개변수 혼합"을 참조하십시오.
③ 데이터 소스를 부동 소수점으로 설정하면 부동 소수점 형식인 ABCD와 CDAB를 지원합니다.
④ 배정밀도 부동 소수점 변환 스케일링은 지원하지 않습니다.

3.12 팬 스케일링 매개변수 혼합

XX07 모델은 펌웨어 1.511 이상, XX06 모델은 펌웨어 1.480 이상, XX12 모델은 1.480 이상 버전을 사용해야 하며, NXT_Vircom 6.71 이상 버전에서 변환 및 확대/축소 매개변수를 혼합하여 사용할 수 있습니다.
이전의 패닝 및 확대/축소 기능에는 다음과 같은 제약 사항이 있었습니다.
1. 한 노드에서 패닝 및 확대/축소를 선택하면 모든 노드에서 Endian/ Endian Swap를 선택해야 했습니다.
2. 변환 및 확대/축소 매개변수는 2/4바이트 선택, 크기 끝/크기 끝 교환 4가지 조합 선택, 소스 데이터가 정수 또는 부동 소수점인지 여부 선택이 가능했습니다. 이전에는 서로 다른 매개변수가 있는 경우 첫 번째 노드의 매개변수만 선택되었고, 여러 노드의 독립적인 매개변수를 구현할 수 없었습니다.

혼합 사용 매개변수 버전:
1. 호환성을 유지하려면 한 노드에서 팬 줌 기능을 활성화해야 하며, 모든 노드에서 팬 줌 기능을 활성화해야 합니다. 팬을 0으로, 줌을 0.99999로 설정하면 실제 팬 줌 기능이 비활성화됩니다.
2. 변환 및 크기 조정 매개변수는 독립적으로 설정할 수 있습니다.
NXT_Vircom 6.71 이상 버전은 호환성 처리를 수행합니다. 변환 및 크기 조정 매개변수가 동일한 경우 이전 구성 파일 형식이 계속 사용되므로 이전 펌웨어에서도 새 NXT_Vircom을 구성할 수 있습니다. 팬 및 크기 조정 매개변수가 일치하지 않는 경우 새 형식이 적용되어 이전 장치는 제대로 작동하지 않으며, 새 펌웨어 장치가 모든 클래스 매개변수를 가져옵니다.

설정 파일에 특별한 위치는 없으며, 각 노드는 다음과 같은 다양한 매개변수를 선택할 수 있습니다.

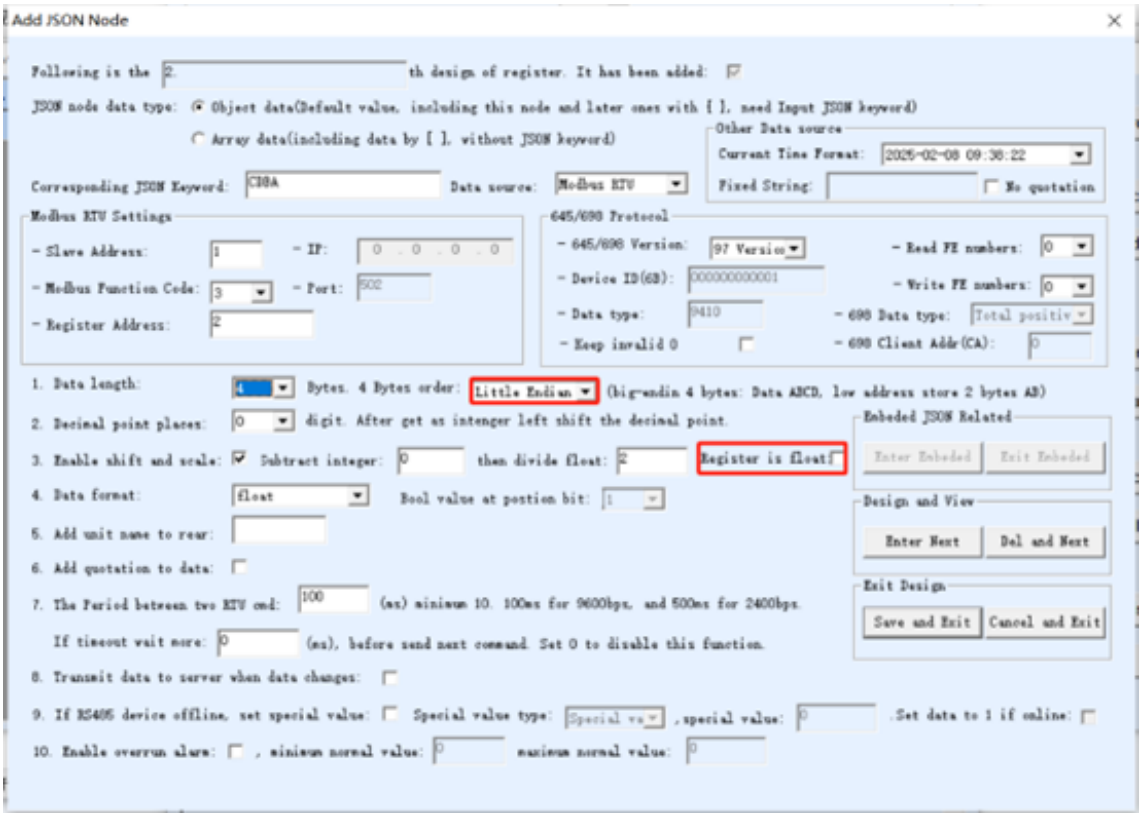


그림 25. 스몰 엔디안 형식의 LONG 정수 구성

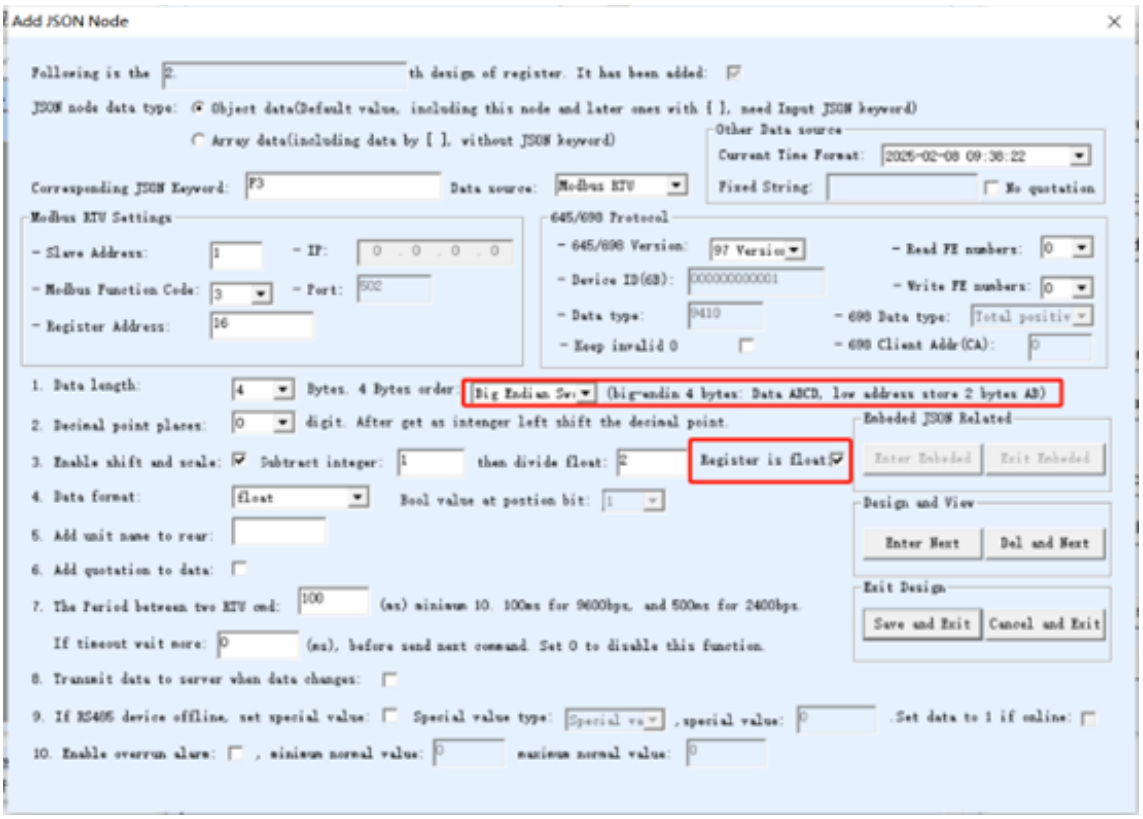


그림 26. Big End 스위치 FLOAT 구성

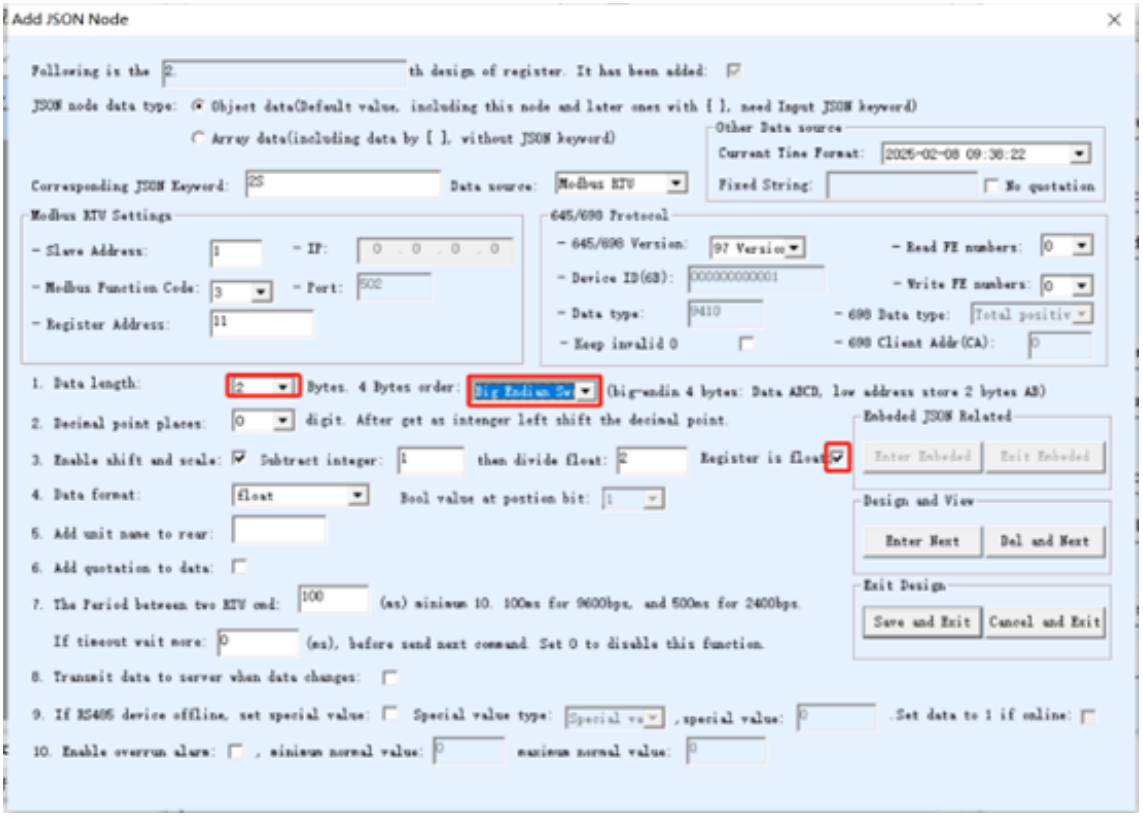


그림 27. 바이트 교환을 위한 2바이트 정수 구성

앞 그림의 세 가지 설정, 즉 바이트 길이, 4바이트 순서 및 부동 소수점 데이터 소스는 서로 다르며 독립적으로 설정할 수 있습니다. 모든 변환 스케일링은 원본 데이터에서 1을 뺀 후 2로 나눕니다. 원본 데이터가 123인 경우 결과는 61이 됩니다.

설정이 완료되면 설정을 다운로드하십시오. 테스트 결과는 다음과 같습니다.

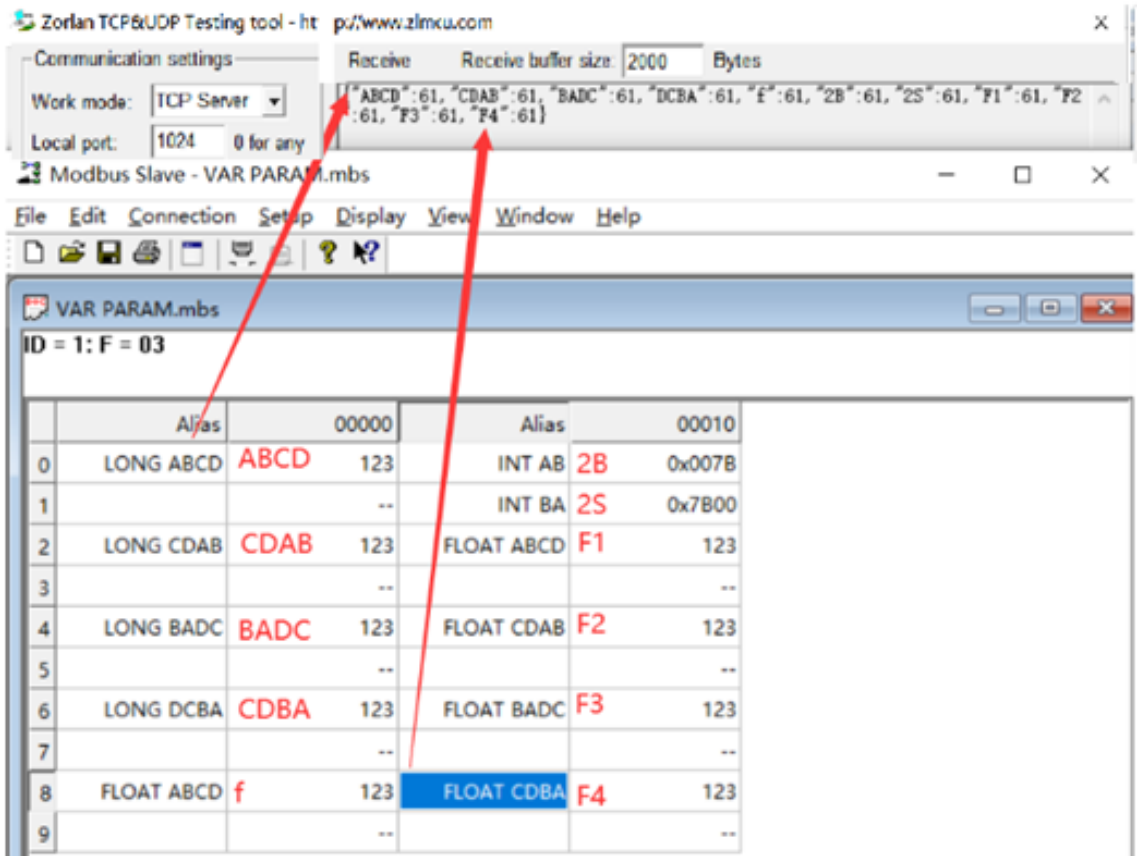


그림 28. 바이트 교환을 위한 2바이트 정수 구성

그래프의 윗부분은 네트워크에서 수신한 데이터입니다. {"ABCD":61,"CDAB":61,"BADC":61,"D-CBA":61,"f":61,"2B":61,"2S":61,"F1":61,"F2":61,"F3":61,"F4":61}, 레지스터는 총 15개입니다.

Json Node name	Number of bytes	Floating point type	Big and small end	Upload results
ABCD	4	N	ABCD	accord
CDAB	4	N	CDAB	accord
BADC	4	N	BADC	accord
CDBA	4	N	CDBA	accord
f	4	N	ABCD	accord
2B	2	N	AB	accord
2S	2	Y	BA	accord
F1	4	Y	ABCD	accord
F2	4	Y	CDAB	accord
F3	4	Y	BADC	accord
F4	4	Y	CDBA	accord

Modbus 슬레이브 구성 결과는 페이지 하단에 표시됩니다. 다양한 구성을 위해 Modbus 슬레이브의 형식과 데이터 유형을 선택하십시오. 테스트 결과, 서로 다른 파라미터 조합으로 구성된 15개의 레지스터 모두 각각 올바른 값을 가져올 수 있음을 보여줍니다.
전송할 데이터 소스는 부동 소수점 및 바이트 교환 방식일 수 있습니다. 이를 지원하기 위해 새로운 vircom과 펌웨어도 필요합니다.

3.13 데이터 변경 보고

경우에 따라 데이터 트래픽을 줄이기 위해 데이터를 자주 업로드할 필요 없이 수집된 데이터에 변경 사항이 있을 때만 업로드하는 것이 좋습니다. 현재 NXT_Vircom 5.30 이상 버전과 1.589 펌웨어를 함께 사용하면 이 기능을 구현할 수 있습니다. 실제로 업로드 간격은 최대 8.8시간까지 설정할 수 있습니다. 트래픽 측면에서 보면 이 시간 동안에는 데이터를 업로드하지 않는 것과 같습니다. 하지만 데이터 변경 시 업로드를 선택하면 노드의 데이터에 변경 사항이 있을 때마다 모든 데이터가 업로드됩니다.
여기서 데이터 변경 기준은 각 JSON 노드별로 설정할 수 있으며, 장치가 오프라인 상태일 때(데이터 내용이 변경될 때도 업로드 가능) 업로드를 트리거하는 데 사용할 수 있습니다.
예를 들어, DI 상태를 수집하고 변경될 때 업로드하는 경우를 생각해 볼 수 있습니다.

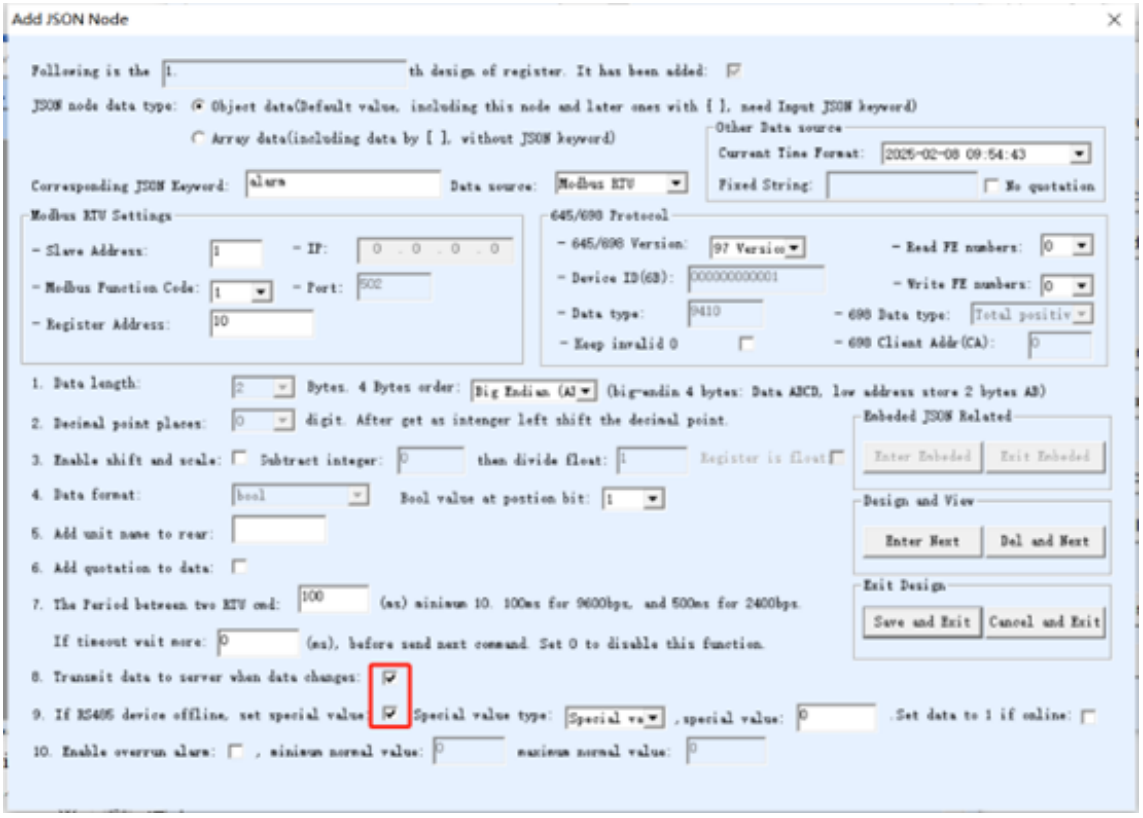


그림 29. 데이터 변경 업로드

RS485 장치의 오프라인 상태를 해제하고 데이터를 업로드하려면 '데이터 변경'을 선택해야 합니다. 이 매개변수를 선택하면 장치가 오프라인 상태가 될 때 보고됩니다. 단, 이 오프라인 보고는 원래 데이터 값이 1인 경우에만 해당됩니다. 모든 오프라인 데이터를 보고하려면 별도의 JSON 노드를 추가해야 합니다. 자세한 내용은 "장치 오프라인" 섹션을 참조하십시오.

IP 주소가 11인 alarm2 노드를 하나 더 추가하되, '데이터 변경 업로드'는 선택하지 마십시오.

일반 데이터는 10초마다 업로드됩니다. 주소 10에서 비트 변경(알람)이 발생하면 보고가 즉시 트리거됩니다(실제로는 데이터 변경의 회전에 수백 밀리초가 소요되므로 업로드가 약간 지연될 수 있습니다). 그러나 주소 11의 비트가 변경되는 경우 변경 보고 옵션은 선택되지 않습니다.

알람이 1인 경우에도 장치가 오프라인 상태이면 즉시 보고됩니다.

3.14 JSON 문제

■ 사용법 소개

JSON 전송의 목표는 단일 JSON 키워드와 해당 Modubs 명령어 출력을 일치시키는 것입니다. 전송된 문자열이 정확히 일치할 필요는 없으며, 전송된 데이터에 해당 JSON 키워드와 관련 데이터가 존재하기만 하면 됩니다.

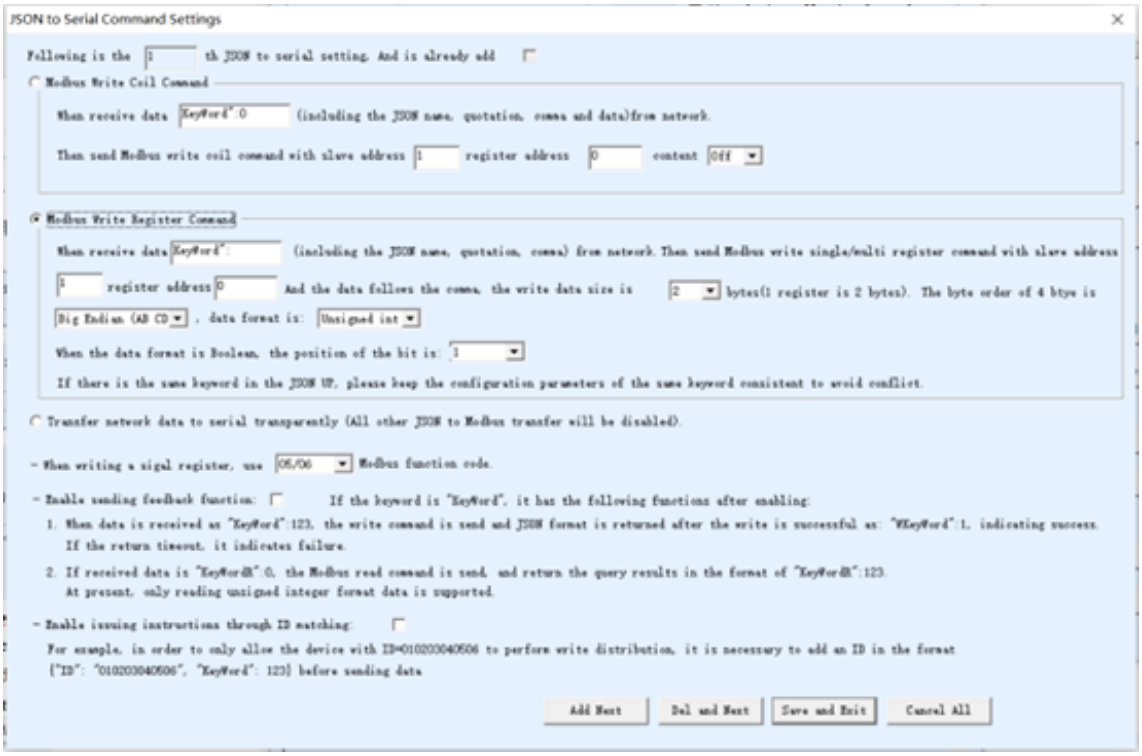


그림 30. JSON을 Modbus로 전송

"JSON to Modbus RTU 설정"의 "JSON 릴리스" 버튼을 클릭하면 위의 대화 상자가 열립니다. 이 대화 상자는 Modbus 코일 설정 명령, Modbus 레지스터 쓰기 명령, 투명 전송 명령의 세 가지 범주로 나뉩니다.

Modbus 코일 설정 명령: 05번 명령으로, 슬레이브 스테이션 주소, 레지스터 주소를 지정하고 켜짐/꺼짐으로 설정할 수 있습니다. 데이터가 "alarm" : "1"과 같은 JSON 형식인 경우, 따옴표와 콜론을 포함하여 "alarm" : "1" 전체를 입력해야 합니다.

Modbus 레지스터 쓰기 명령: 06번 명령으로, 현재 2~8바이트(즉, 1, 2, 4바이트 레지스터 쓰기)를 지원합니다. 데이터는 소수점 없이 형식이어야 합니다. 여기에서 전송 스테이션 주소, 레지스터 주소 및 바이트 수를 설정할 수 있습니다. 4바이트 또는 8바이트인 경우, Large endian 또는 Little endian 형식을 선택할 수 있습니다. 해당 방식이 temp: 1234인 경우, 입력란에 temp:를 입력합니다. 따옴표를 포함하여 1234 이전의 문자는 모두 입력해야 합니다.

네 가지 small 및 small end 형식을 포함하여 8바이트의 배정밀도 JSON 전송을 지원합니다.

투명 전송 모드에서는 전송된 명령이 파싱 없이 시리얼 포트로 투명하게 전송됩니다. 투명 전송 모드를 선택하면 모든 JSON 전송 파싱 기능이 비활성화됩니다.

■ 멀티바이트 전송

JSON을 Modbus RTU로 변환하는 기능은 05/06/16 명령어를 지원합니다. 15 명령어를 사용하여 두 개 이상의 코일을 설정해야 하는 경우, 05 명령어를 여러 번 사용하십시오.
시스템은 바이트 수에 따라 06 또는 16 명령어를 자동으로 선택하여 전송합니다. 다음은 코일 설정 및 레지스터 설정 예시입니다.
{alert: "on"}에서 JSON 데이터를 수신하는 경우, 스테이션 주소 02의 코일을 레지스터 03부터 설정하려면 05 명령어를 사용해야 합니다. JSON을 Modbus로 변환 화면에서 "JSON 전송"을 클릭하십시오.

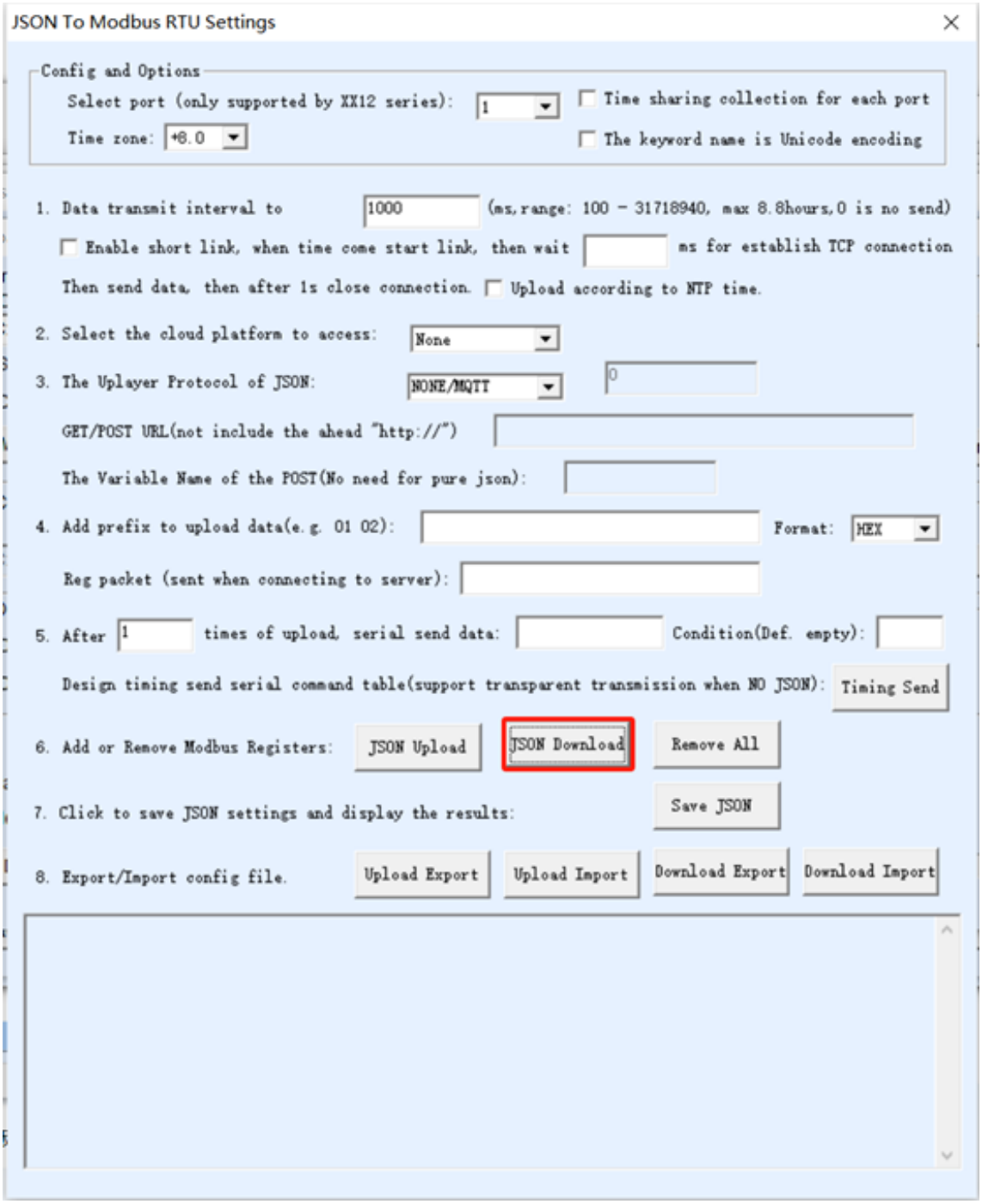


그림 31. JSON 전송 진입

설정 화면은 다음과 같습니다. "켜짐" 설정 시 필요한 부분을 입력해야 한다는 알림에 유의하십시오.

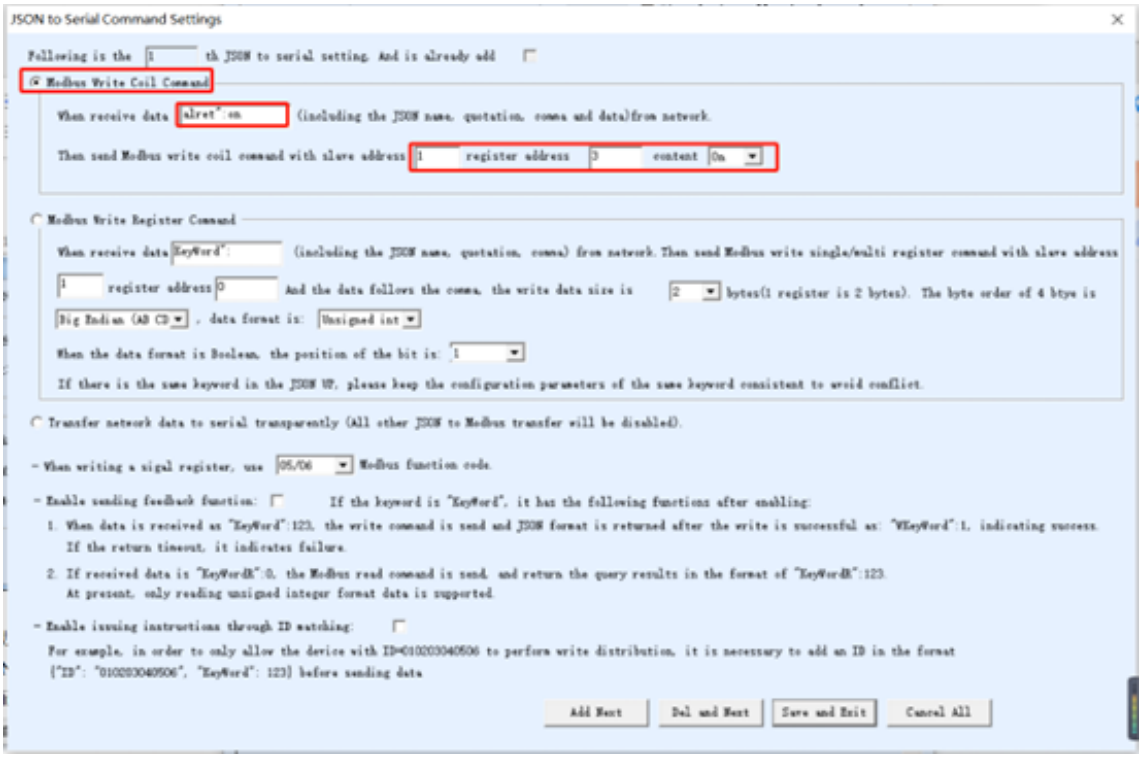


그림 32. 코일 구성

다른 전송 변환을 추가하려면 "다음"을 클릭하고, 그렇지 않으면 "모두 저장하고 종료"를 클릭합니다. 메인 화면으로 돌아가서 "JSON 설정 저장"을 클릭한 다음 "돌아가기"를 클릭합니다. 그런 다음 다운로드 화면에서 "다운로드"를 클릭합니다. 이렇게 하면 구성이 완료됩니다.

{power: "12345"}를 전송하는 경우, power 값 12345를 스테이션 주소 2, 등록 3에 설정해야 합니다. 설정은 다음과 같습니다.

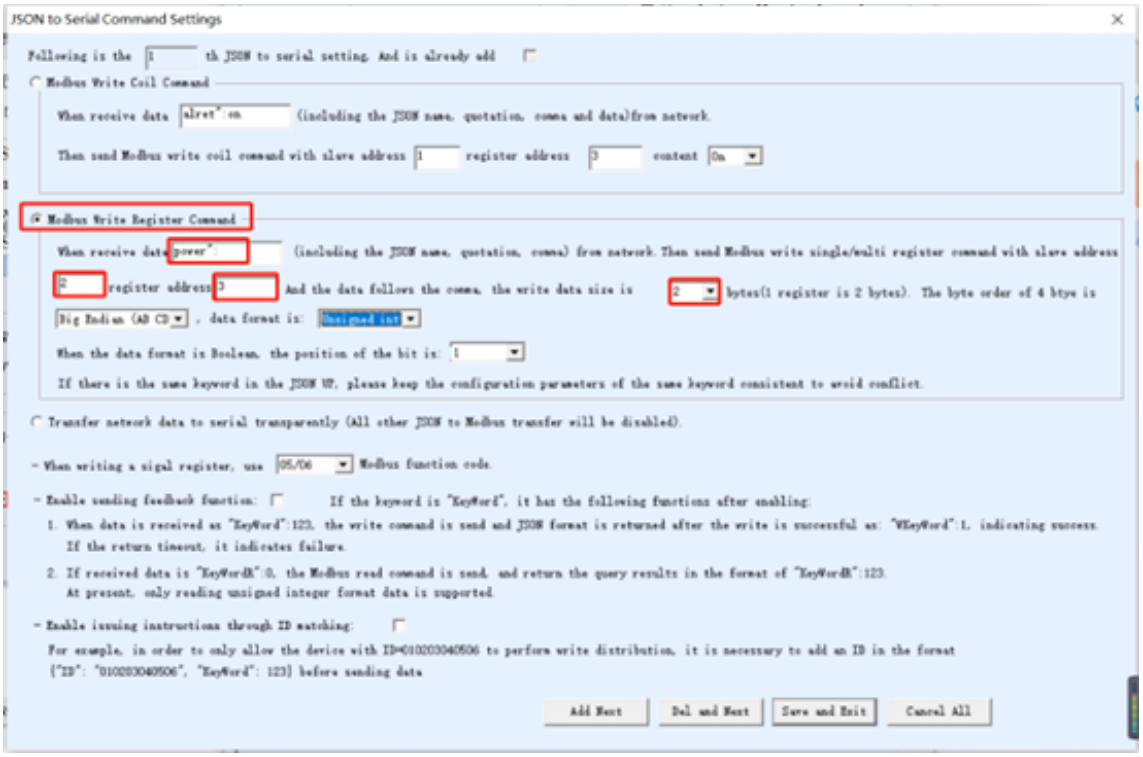


그림 33. JSON 설정 레지스터

여기서 키워드는 "power: "만 입력하면 되고, 뒤따르는 12345는 입력할 필요가 없습니다. 이 값은 변경되기 때문입니다. 하지만 전달된 데이터의 따옴표도 따옴표로 묶어야 하는 경우에는 콜론(:)을 입력해야 합니다.

■ 15/16 명령어 사용

경우에 따라 기기가 05/06 기능 코드를 지원하지 않으므로 단일 바이트에도 15/16 명령어 기능 코드를 사용해야 합니다. 이 경우 기본 전송 규칙을 수정해야 합니다. 기본 전송 규칙은 단일 코일/레지스터 설정에는 05/06 기능 코드를 사용하고, 그렇지 않으면 15/16 기능 코드를 사용하는 것입니다.

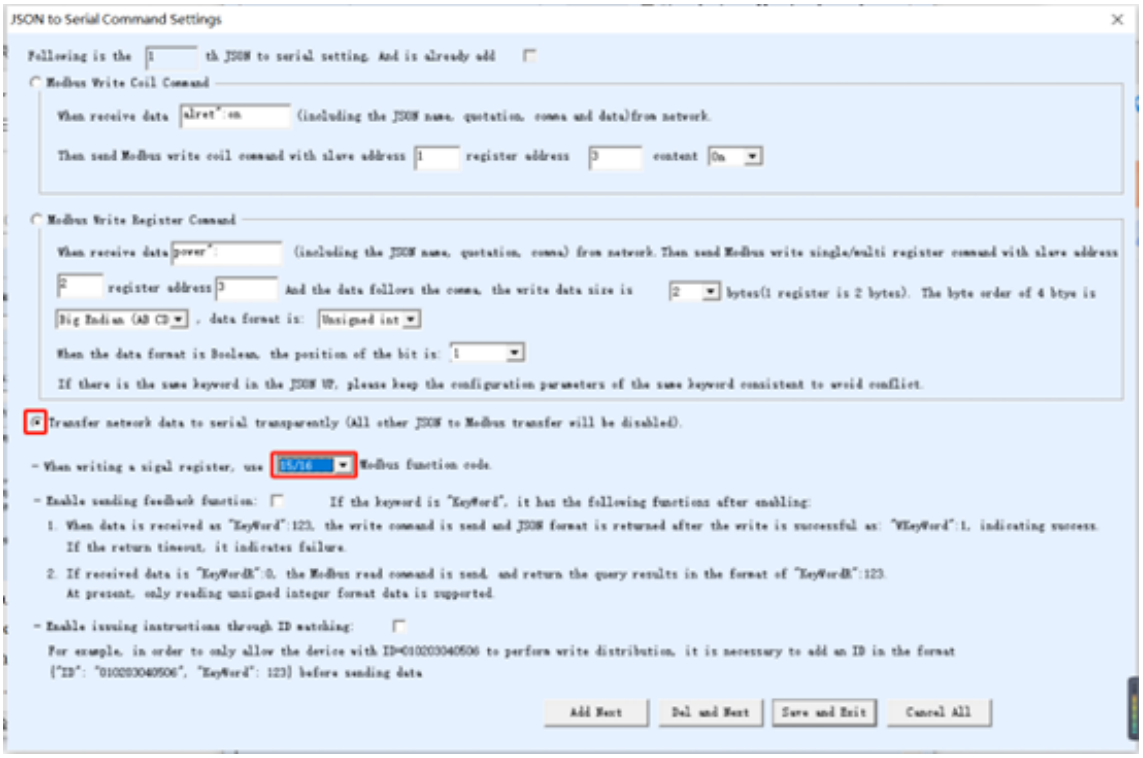


그림 34. 15/16 지시 사용

NXT_Vircom 6.73 이상 버전에서는 위 그림과 같이 15/16 기능 코드를 수동으로 선택할 수 있습니다. 이 옵션은 모든 노드가 아닌 특정 노드 하나에만 적용됩니다. 코일과 레지스터에 모두 사용할 수 있습니다. 15/16 기능 코드를 선택하면, 해당 설정이 단일 코일/레지스터인 경우에도 15/16 기능 코드가 사용됩니다.

■ 송신 및 송신 매치

JSON은 설정할 매개변수가 더 많지만, 전송되는 매개변수는 더 적습니다. 때때로 전송된 JSON 키워드와 전달된 JSON 키워드가 동일한 경우가 있습니다. 이 경우, 다음과 같은 상황에서는 전송된 매개변수 중 일부가 전달에 사용될 수 있습니다.

1. 데이터 소스가 Mdobus TCP인 경우: IP와 포트는 위에서 전송된 매개변수를 사용합니다.
2. 변환 및 스케일링이 있는 경우: 변환 값, 스케일링 값, 데이터 길이, 최종 크기, 소스 데이터가 부동 소수점인 경우, 위에서 전송된 매개변수가 사용됩니다. 단, 레지스터 위치는 여전히 정확하게 설정해야 하며, 전송되는 데이터 유형은 정수 또는 부동 소수점일 수 있습니다.
3. 03 기능 코드 bool 유형: 기능 코드는 03으로 설정되고, 데이터 유형은 bool로 설정되며, bool 값의 위치는 JSON 파일에 설정된 매개변수입니다.
4. 부동 소수점 데이터(변환 및 스케일링 제외): 데이터 유형은 부동 소수점이고, 데이터 길이는 JSON 업로드 값으로 설정되며, 최종 크기도 JSON 업로드 값으로 설정됩니다.

3.15 함수 코드 BOOL 유형 전달

예를 들어, 03 기능 코드의 BOOL 키워드가 {"a":1}과 같이 전송되는 경우, "a"라는 이름이 동일하게 설정되면 BOOL 비트의 위치를 식별할 수 있습니다. 상위 전송이 없고 별도의 전송이 있는 경우, 이전 NXT_Vircom에는 BOOL 비트의 위치를 나타내는 매개변수가 없습니다.

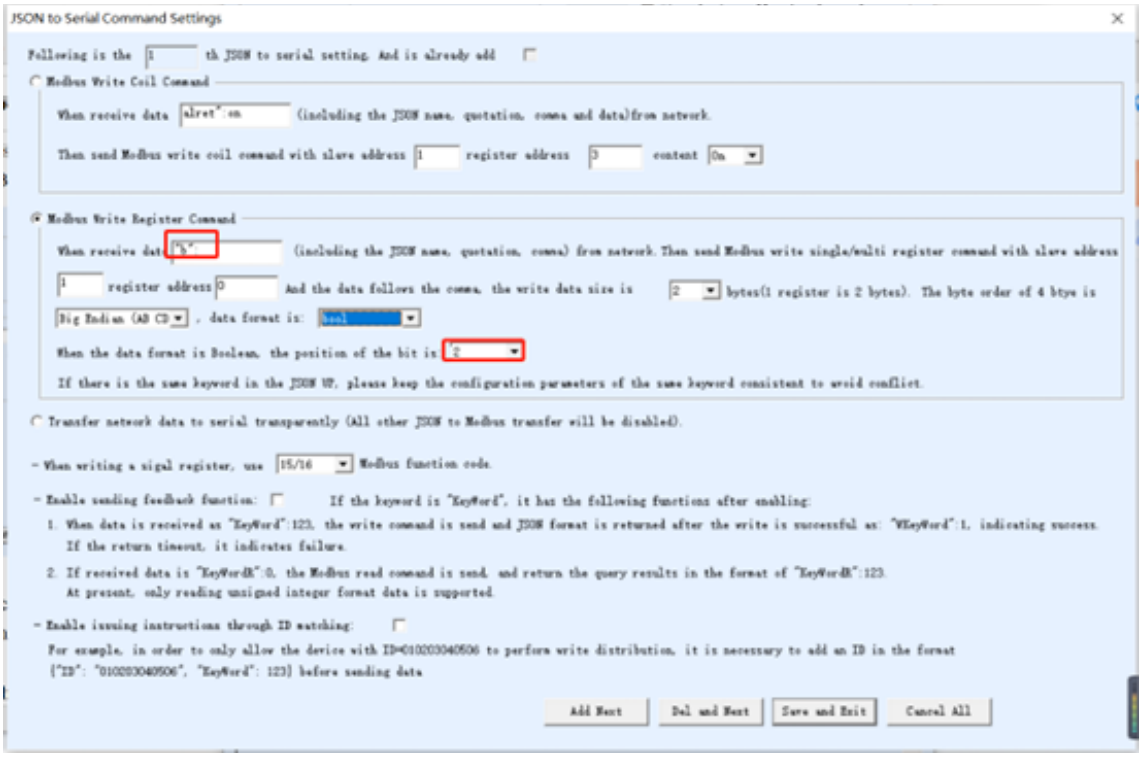


그림 35. BOOL 비트를 전달하기 위한 위치 설정

NXT_Vircom 6.73 이상 버전에서는 이슈에 존재하지 않는 키워드일지라도 별도의 비트 위치 세트를 지정할 수 있습니다. 단, 부울 값이 저장될 비트를 선택하려면 데이터 형식을 부울로 설정해야 합니다.
설정 후 "b" :1을 전송하면 두 번째 비트가 변경됩니다. 즉, 레지스터 값이 이전의 비부울 타입인 0x0001 대신 0x0002로 변경됩니다.

3.16 JSON 일괄 전송

JSON 형식으로 설정 레지스터를 전송할 때, {"a1":1,"a2":2,"a3":3}과 같이 여러 변수가 동시에 전송될 수 있습니다. 이전 버전에서도 이러한 전송 방식을 지원했지만, a1, a2, a3의 순서는 설정 순서와 일치해야 하며, 변수 사이에 공백이 없어야 했습니다. 하지만 실제 전송되는 변수의 개수가 약간 적거나 순서가 다를 수 있습니다. 2023년 11월 12일 이후 버전으로 업그레이드하면 이 문제가 해결됩니다. 이후에는 설정 가능한 모든 JSON 변수 조합을 한 번에 전송할 수 있습니다.

3.17 645 프로토콜 타이밍

645 프로토콜의 타이밍은 68 99 99 99 99 99 99 99 99 99 68 08 06 SS MM HH DD MM YY CS 16이며, 여기서 SS MM HH DD MM YY는 각각 초, 분, 시, 일, 월, 년을 나타냅니다.
NXT_Vircom 5.32 및 4.99(ZLSN7044)/5.92(ZLSN2043) 펌웨어를 사용하면 타이밍 설정을 구현할 수 있습니다. 다음 문자열 "68 99 99 99 99 99 99 99 99 99 68 08 06 TIME[25...30] CRC[3] 16"을 시리얼 포트 출력 명령에 동시에 복사합니다. "TIME[0]"을 출력 조건에 복사합니다.
타이밍 전송을 위해서는 최소 하나의 업로드 JSON 데이터가 설정되어 있어야 합니다. 또한, 시간은 네트워크를 통해 전송되므로 TCP 연결이 설정되어야만 시간을 전송할 수 있습니다. TCP 연결이 설정되지 않으면 시간이 전송되지 않고, 따라서 시간도 전송되지 않습니다.

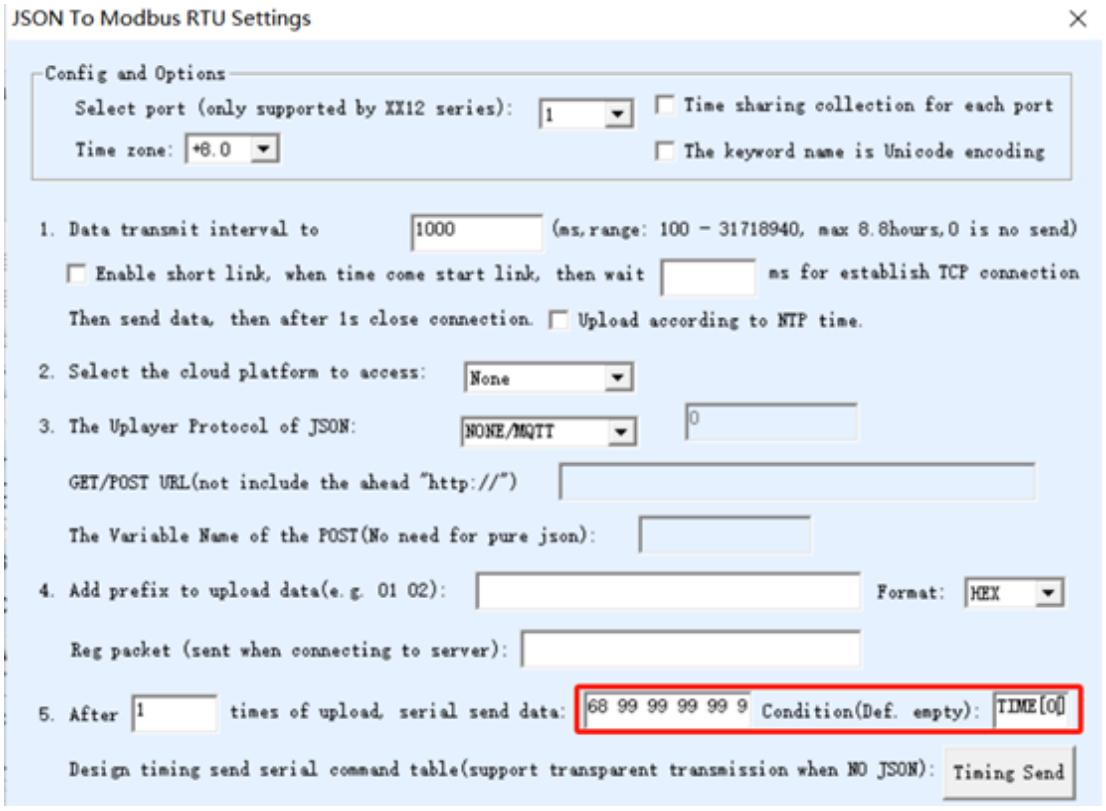


그림 36. 645 타이밍 구성

여기서 "TIME[0]"은 선택 사항입니다. 입력하면 유효한 시간 후에 장치가 전송됩니다. 입력하지 않으면 설정된 시간에 데이터가 전송되며, 시간이 올바르지 않을 경우 오류가 발생할 수 있습니다."

3.18. JSON 응답 전송

3.19. UDP 모드에서 JSON 형식으로 데이터 전송

3.20 한계 초과 경보

범위 초과 경보 기능은 Modbus로 수집된 데이터가 특정 범위를 초과할 경우 즉시 데이터를 보고하는 기능을 제공합니다. 데이터 계산 및 처리는 지역화되어 있습니다. 수집된 데이터를 원하는 값으로 변환한 후 c라고 할 때, c의 최소 정상값을 a, 최대 정상값을 b라고 하면, a<=c<=b일 때 정상이고, 그렇지 않으면 비정상입니다.
장치는 수집된 데이터가 정상 상태에서 비정상 상태로 변경되거나 비정상 상태에서 정상 상태로 복구되는 것을 감지하면 즉시 서버로 JSON 패킷을 전송합니다.
시스템이 정상 상태 또는 비정상 상태를 감지하는 속도는 폴링 시간과 관련이 있으며, 이는 Modbus로 전송되는 JSON의 "직렬 포트 폴링 간격"을 통해 설정할 수 있고, 설정된 Modbus 수집 지점 수에도 영향을 받습니다.
일반적으로 최대 시간은 수집 지점 수에 폴링 시간을 곱한 값 + 1초를 초과하지 않습니다.

현재, 상한/하한값이 일반적으로 부동 소수점이고 수집된 아날로그 값은 일반적으로 정수이기 때문에, 비교를 위해 수집된 데이터를 부동 소수점으로 변환해야 하므로, 패닝 및 스케일링 기능은 팬터그래프 기능과 함께 사용해야 합니다. 패닝 및 스케일링은 2바이트, 4바이트, 사이드 엔드 형식의 데이터 소스뿐만 아니라 단정밀도 부동 소수점 데이터 소스도 지원합니다.
패닝 및 스케일링은 모든 수집 데이터에 대해 활성화해야 하므로, 상한 경보를 사용할 때는 모든 데이터에 대해 패닝 및 스케일링을 적용해야 합니다. 패닝 및 스케일링이 필요하지 않은 데이터의 경우, 패닝 및 스케일링을 활성화하더라도 패닝 및 스케일링 값을 0 또는 기본값인 0.99999로 유지하면 데이터에 영향을 주지 않습니다.
다음은 구체적인 사용 예입니다.

4~20mA 계측기를 가정해 보겠습니다. 4mA일 때 유량은 0이고, 20mA일 때 유량은 100입니다. 전류를 I, 유량을 F라고 하면 전류를 유량으로 변환하는 공식은 F=(I-4)/(20-4)*(100-0)+0= 6.25* i-25입니다.
ZLAN6002A는 아날로그 값을 수집하는 데 사용됩니다. 레지스터에서 읽은 값이 R이면 실제 전류 값은 I=R/1024*25(mA) 공식으로 계산됩니다. 위 공식을 대입하면 다음과 같습니다.
F I - 25 = = 6.25 * 6.25 * (R) / 1024 x 25-25 = (R - 163.84) / 6.5536
변환은 정수만 가능하므로 164를 사용하고, 소수점 이하 6자리까지만 유지하므로 6.5536이 됩니다. 즉,
F = (R-164)/6.5536
만약 12.1에서 80.123 사이의 정상 범위에 있다면, 소수점을 제거한 후 해당 방법의 값이 65535보다 클 수 없기 때문에(80.123은 65535보다 큼), 80.123을 80.12로 변경해야 합니다. 또한 음수 상한 및 하한은 지원되지 않습니다. ZLAN6002A의 첫 번째 채널 아날로그 신호를 수집한다고 가정하면, 기능 코드는 4이고 레지스터 주소는 0이며 설정은 다음과 같습니다.

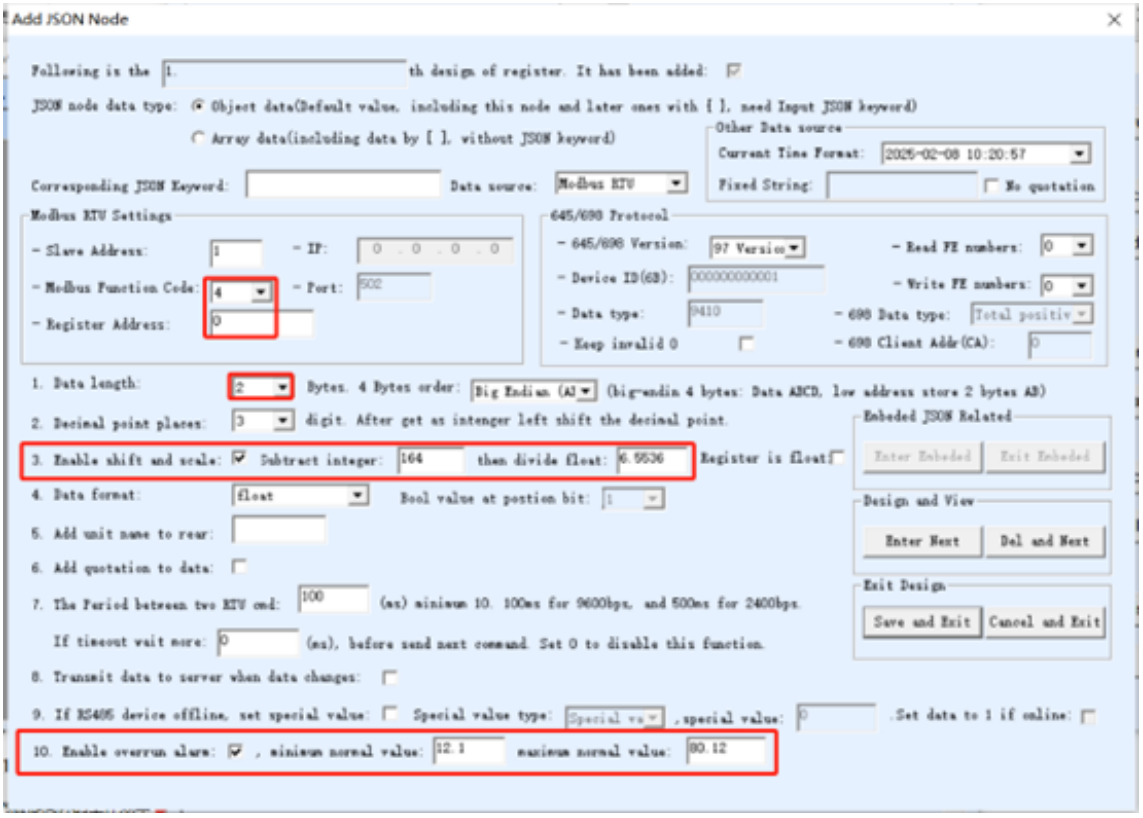


그림 37. 한계 초과 경보

역추론에 따르면 레지스터 R=243 정도가 하한이고, R=689 정도가 상한입니다. 레지스터 R=243과 R=244가 변경되면 즉시 {"Flow":11.902} 또는 {"Flow":12.207}이 전송되어 하한 및 복구 상태가 보고됩니다. 레지스터 R=690과 R=689가 변경되면 즉시 {"Flow":80.261} 또는 {"Flow":80.109}이 전송되어 상한 및 복구 상태가 보고됩니다.

여러 레지스터에 대해 서로 다른 변환 스케일링과 상한 및 하한을 설정할 수 있습니다.

3.20 TCP 단거리 연결에 대한 데이터 보고

이 기능은 데이터 패킷을 전송해야 할 때 TCP 서버 모드에서 TCP 클라이언트 모드로 자동으로 전환합니다. 목적지 IP 주소와 포트는 매개변수에 미리 설정되어 있습니다. T밀리초 동안 대기하여 연결이 설정되면 전송을 시작하고, 전송이 완료되면 1초 후에 다시 TCP 서버 모드로 돌아갑니다. 여기서 T는 설정 가능하며 기본값은 1000입니다.

다음 그림과 같이 짧은 연결을 위해서는 장치 매개변수를 미리 수정해야 합니다. 먼저 TCP 클라이언트 모드로 전환하여 목적지 IP 주소(192.168.0.101)와 포트(1024)를 설정한 다음, TCP 서버 모드로 전환하고 설정을 저장합니다.

3.21 TCP 단거리 연결에 대한 데이터 보고

이 기능은 데이터 패킷을 전송해야 할 때 TCP 서버 모드에서 TCP 클라이언트 모드로 자동으로 전환합니다. 목적지 IP 주소와 포트는 매개변수에 미리 설정되어 있습니다. T밀리초 동안 대기하여 연결이 설정되면 전송을 시작하고, 전송이 완료되면 1초 후에 다시 TCP 서버 모드로 돌아갑니다. 여기서 T는 설정 가능하며 기본값은 1000입니다.

다음 그림과 같이 짧은 연결을 위해서는 장치 매개변수를 미리 수정해야 합니다. 먼저 TCP 클라이언트 모드로 전환하여 목적지 IP 주소(192.168.0.101)와 포트(1024)를 설정한 다음, TCP 서버 모드로 전환하고 설정을 저장합니다.

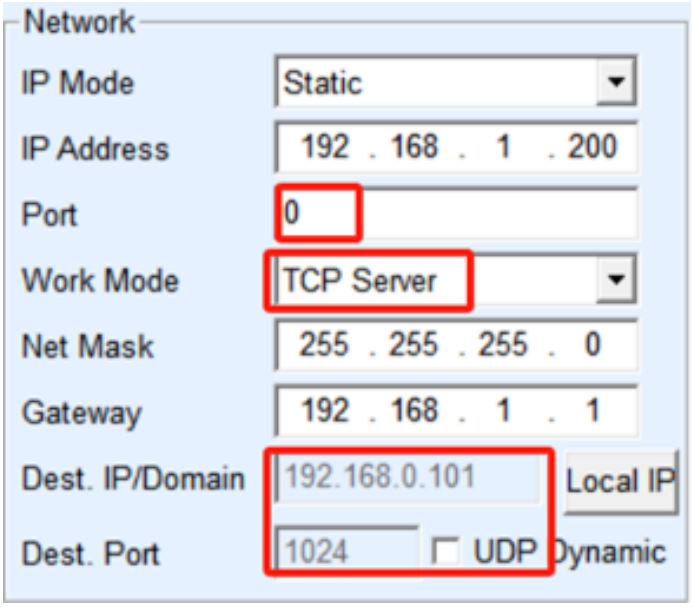


그림 38. 짧은 연결에 대한 매개변수 설정

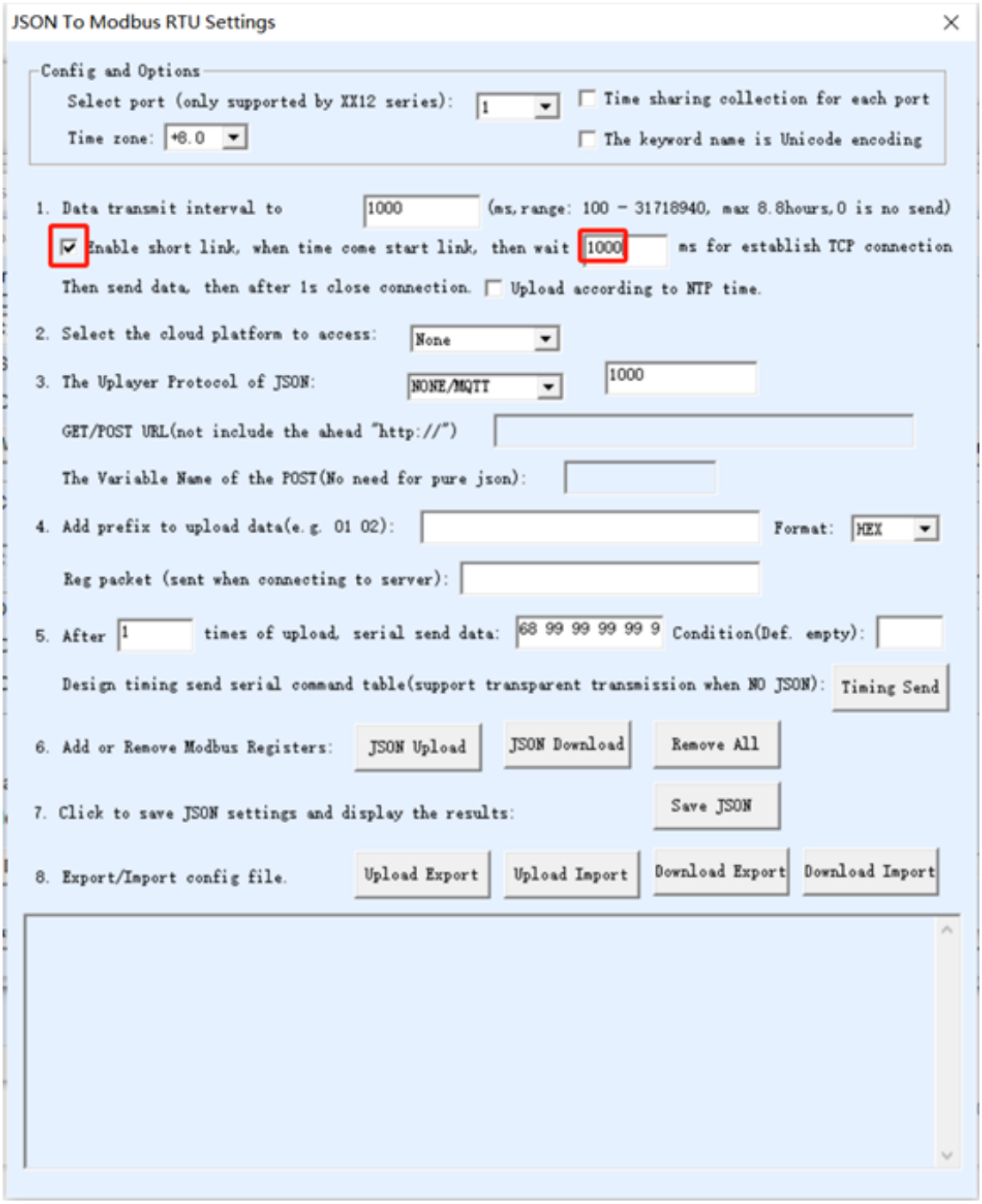


그림 39. 짧은 연결 활성화

JSON-Modbus RTU 설정 대화 상자로 이동하여 '단축 연결' 모드를 선택하고 서버 연결 대기 시간을 설정합니다(일반적으로 1000ms). 1000ms 이내에 연결이 설정되지 않으면 전송된 데이터가 손실됩니다. 따라서 연결이 설정되도록 충분한 시간을 설정해야 합니다. 단, 설정 시간이 너무 길면 데이터 보고 시간이 지연될 수 있습니다.

다른 설정은 장기 연결에 사용되는 설정과 동일합니다. 정확한 시간 보고에 NTP 시간을 사용하려면 'NTP 시간으로 보고'를 선택합니다. 데이터 변경 보고를 동시에 활성화할 수 있으며, 한계 초과 경보 기능도 동시에 사용할 수 있습니다.

JSON TCP 단축 연결 보고 설명:

1. TCP 단축 연결은 MQTT 프로토콜과 동시에 사용할 수 있으며, 이 기능은 장치가 MQTT 서버에서 장시간 연결이 끊어지는 경우를 해결하는 데 사용할 수 있습니다.

2. 전송 서버 시간: 연결이 끊어진 시점부터 다시 연결이 설정된 시점까지의 간격을 나타냅니다. 연결 시간은 장치 홈 화면에 설정된 연결 해제 및 재연결 시간과 무관합니다.

3. 위 그림의 "데이터 전송 전 연결 설정 대기 시간(밀리초)": MQTT의 경우 연결에는 일반적으로 3초가 소요되며, 3000으로 설정할 수 있습니다.

4. JSON 데이터의 경우 콘텐츠 변경 사항이 보고되며, 짧은 연결을 함께 사용할 수 있습니다. 데이터가 변경되면 장치는 연결을 설정하고 데이터를 전송하려고 시도합니다. 이 경우 데이터가 주기적으로 전송되는 데이터와 함께 사용될 수 있습니다.

5. Ping 시간(Keep Alive)은 반드시 데이터 수집 주기보다 길게 설정해 주세요.

3.22 Modbus TCP를 JSON으로 변환

192.168.0.102의 502번 포트와 192.168.0.103의 502번 포트에서 Modbus TCP 슬레이브 장치의 데이터를 수집하고, 동시에 Modbus RTU 장치의 데이터도 수집해야 한다고 가정해 보겠습니다. 수집된 데이터는 192.168.0.102의 1024번 포트로 전송됩니다.

설정은 다음과 같습니다. 변환 프로토콜을 '없음'으로 설정합니다.

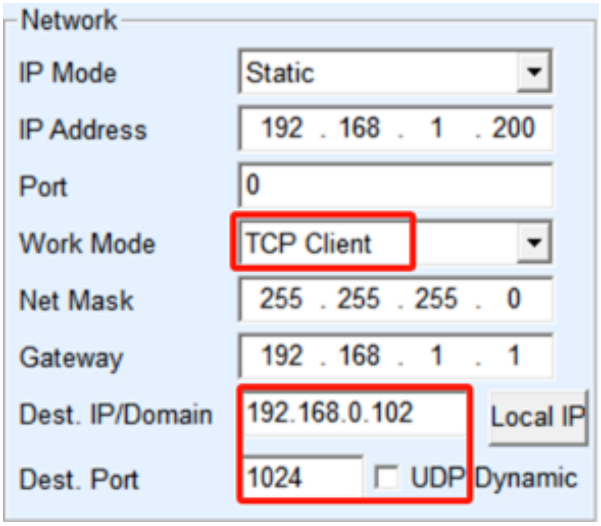


그림 40. 기본 소켓 연결 설정

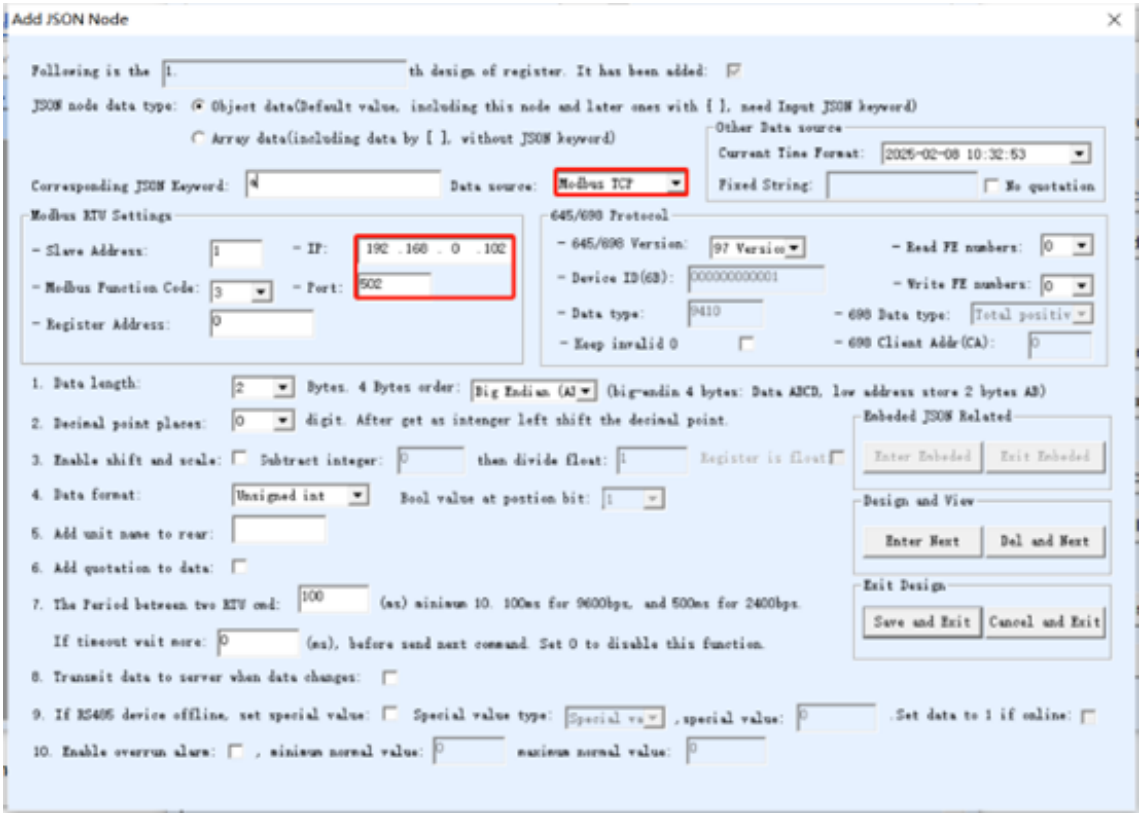


그림 41. 기본 소켓 연결 설정

Modbus TCP를 사용하려면 데이터 소스로 Modbus TCP를 선택하고 IP 주소와 포트 번호를 입력하십시오.
최대 6개의 IP 주소와 포트를 설정할 수 있습니다.
Modbus RTU 장비와 함께 사용할 수 있습니다.
다른 용도는 Modbus RTU와 호환됩니다.

■ Modbus TCP 간략 연결

JSON에서 Modbus RTU로의 연결 모드를 '단거리'로 선택한 경우, Modbus TCP를 사용하면 데이터 수집이 실패할 수 있습니다.

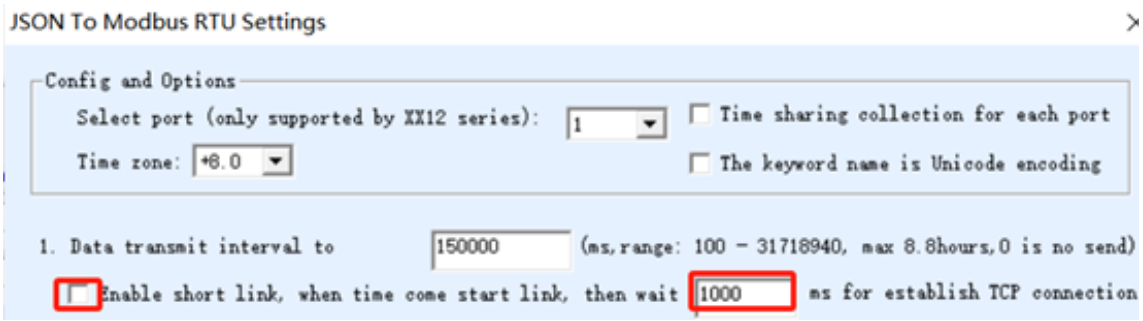


그림 42. 짧은 연결 시작

Modbus TCP 데이터는 장치가 TCP 클라이언트 모드일 때만 수집할 수 있기 때문입니다. 하지만 짧은 연결의 경우 장치는 대부분 TCP 서버 모드에 있고 TCP 클라이언트 모드는 1초 남짓만 유지됩니다. 따라서 연결 해제 시간을 0으로 변경해야 합니다. 그렇지 않으면 데이터를 수집할 수 없습니다.

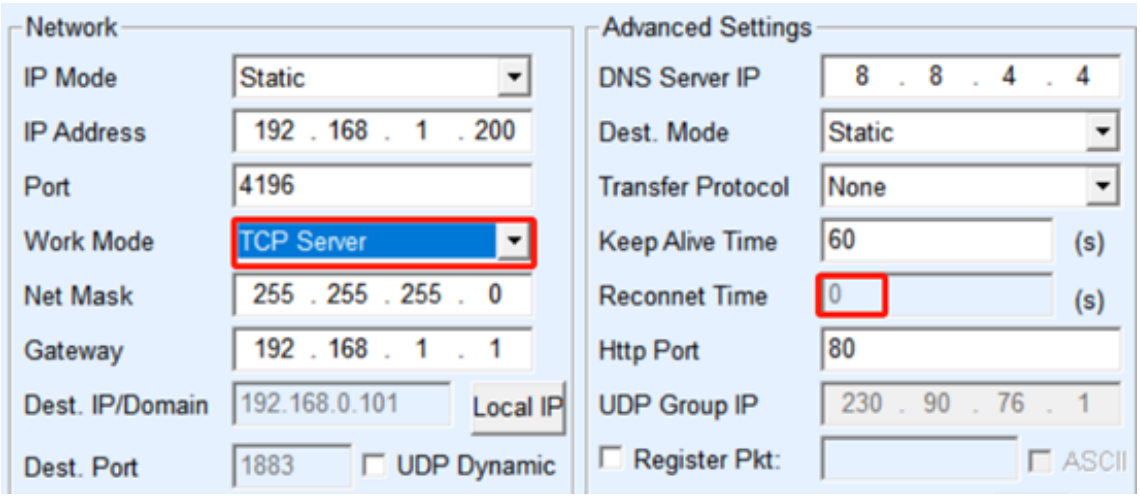


그림 43. 단선된 선이 0에 다시 연결되었습니다.

단, "단축 연결" + "Modbus TCP"는 "한계 초과 경보"와 함께 사용할 수 없습니다.

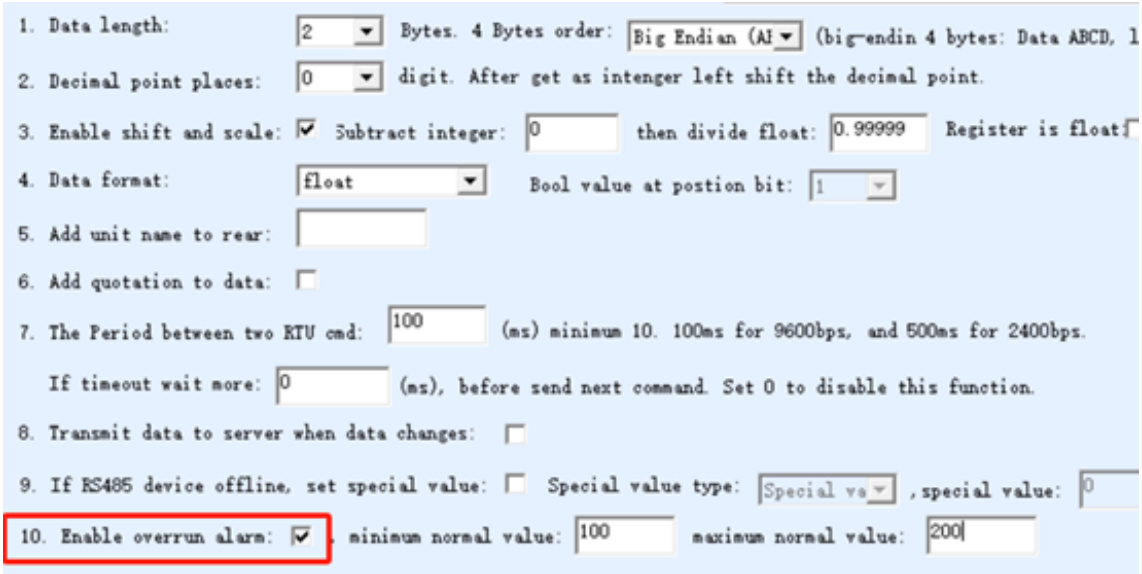


그림 44. 한계 초과 경보

초과 경보 기능은 장치가 Modbus TCP 레지스터 데이터를 지속적으로 수집해야 하므로, 범위를 초과하는 경우 JSON 데이터를 전송해야 합니다. 그러나 장치가 대부분 Modbus TCP 모드로 작동하기 때문에 데이터 수집이 불가능하여 데이터 변경 여부를 알 수 없습니다. TCP 클라이언트 모드로 전환되어 데이터를 수집하는 시점이 JSON 데이터 전송 시점과 겹치므로, 사전에 JSON 경보 데이터를 전송하는 것은 의미가 없습니다.

3.23 JSON을 Modbus TCP로 변환

■ 코일 설정

Modbus TCP를 JSON으로 전송하는 설정이 다음과 같다고 가정해 보겠습니다.

keyword	IP	port	Slave address	Function code	address
a	192.168.0.100	502	1	1	0
b	192.168.0.100	503	1	1	1

JSON에서 Modbus RTU로 데이터를 전송할 경우, 다음 그림과 같이 aoff, aon, boff, bon 등의 전송 키워드를 설정하여 전송 제어를 지정할 수 있습니다. 슬레이브 스테이션 주소, 레지스터 주소, 그리고 켜짐/꺼짐 키워드의 내용만으로도 명령 정보를 충분히 표현할 수 있습니다.

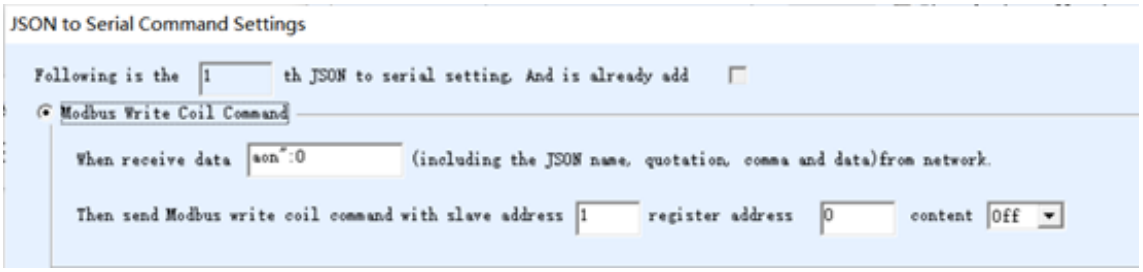


그림 45. 일반 JSON-Modbus RTU의 코일 제어

하지만 JSON을 Modbus TCP로 변환할 때는 IP 주소와 포트 정보도 필요합니다. 이 경우, 이전과 동일한 이름을 사용하여 일치하는 항목을 찾아 IP 주소와 포트를 확인해야 합니다.

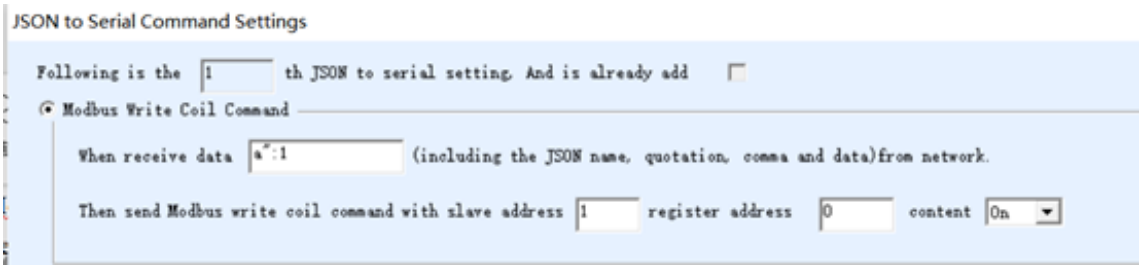


그림 46. JSON을 Modbus TCP로 제어하는 코일 온

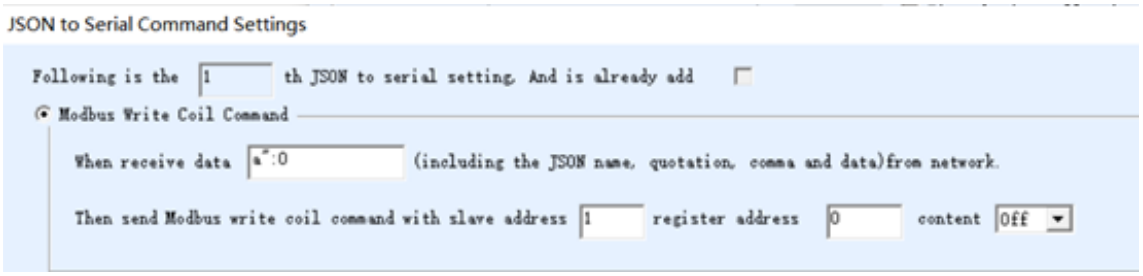


그림 47. JSON에서 Modbus TCP로의 코일 오프 제어

그림에서 ":1"과 ":0"은 전달된 데이터의 유효한 부분을 나타내는 데 사용됩니다. 실제 전달된 데이터는 {"a":1}일 수 있습니다. 키워드 이름은 앞의 텍스트와 동일하게 "a"여야 합니다. 또한 위의 대화 상자에서 레지스터 위치와 켜짐/꺼짐 여부도 올바르게 설정해야 합니다.

{"a":0}, {"a":1}, {"b":0}, {"b":1}로 발행된 JSON 서버는 Modbus TCP의 해당 코일을 설정할 수 있습니다.

■ 레지스터 설정

Modbus TCP를 JSON으로 전송하는 설정이 다음과 같다고 가정해 보겠습니다.

keyword	IP	port	Slave address	Function code	address
aReg	192.168.0.100	502	1	3	0
bReg	192.168.0.100	503	1	3	1

불리언 데이터 형식을 선택하지 말고 일반적으로 부호 없는 정수를 선택하십시오.
코일(Coil)설정과 마찬가지로 설정 레지스터에도 JSON 키워드 이름을 입력해야 하며, 위에서 언급한 내용과 일치해야 합니다. 여기서는 bReg ":로 설정합니다. 단, 뒤의 0과 1은 입력할 필요가 없으며, 장치가 실제로 수신한 셰이핑 데이터에 따라 값을 설정합니다.

그림 48. JSON을 Modbus TCP에 등록하는 설정

레지스터 주소는 위에 보낸 주소와 동일해야 하므로, 여기서는 1로 설정하십시오. 만약 주소가 다르다면, 여기에 설정된 주소가 기본 주소가 됩니다. 업센딩과 송신에 사용되는 레지스터는 서로 다릅니다.

■ 부동 소수점 데이터

Modbus 부동 소수점 형식을 사용할 경우: 데이터 형식은 부동 소수점으로 설정하고, 데이터 길이는 4바이트로 설정하며, 소수점 이하 자리 수는 실제 요구 사항에 따라 설정합니다.
이 경우 전송되는 바이트의 길이도 4바이트로 설정해야 합니다. 예를 들어, {"bReg":123.123}과 같이 부동 소수점 레지스터를 설정할 수 있습니다.

■ 반환

JSON 형식은 Modbus TCP로 전송됩니다. 현재 JSON 형식으로 레지스터 내용을 읽고 JSON 형식으로 반환하는 기능은 지원되지 않습니다.

3.24 645 프로토콜 심볼

645 프로토콜에서 반환되는 데이터의 기호는 다음 세 가지 방식으로 처리됩니다.
1. 부호 없는 정수를 선택한 경우: 모든 데이터는 부호 없는 것으로 간주됩니다. 예를 들어 8012는 0x8012로 처리됩니다.
2. 부호 있는 형식을 선택한 경우: 최상위 비트가 1인 경우 음수 기호로 간주됩니다. 예를 들어 8012는 -12로 처리됩니다.
3. 부동 소수점 형식을 선택한 경우: 최상위 비트가 1인 경우 음수 기호로 간주되지만 기호는 표시되지 않습니다. 예를 들어 8012는 -12로 간주되지만, 출력은 절댓값만 표시되고 기호는 표시되지 않습니다.

3.25 16진수 데이터 전송

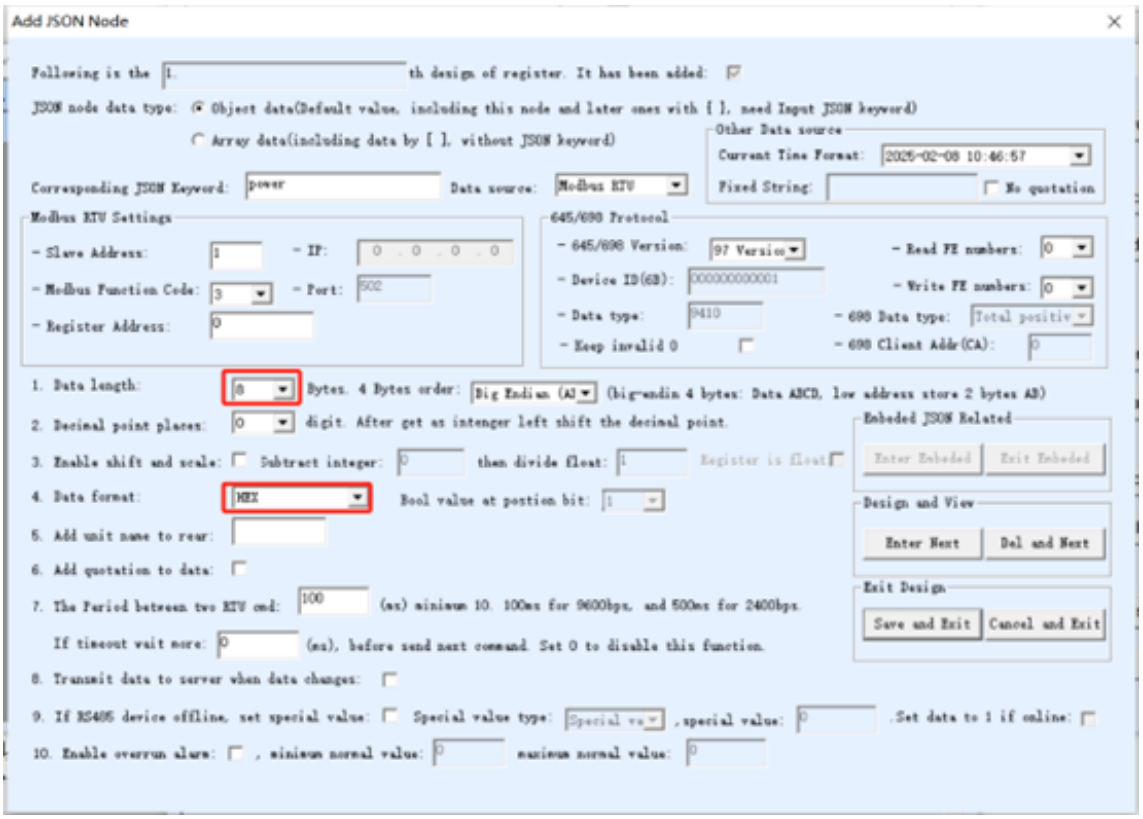


그림 49. 16진수 보고 설정

데이터 형식이 16진수로 설정된 경우, 데이터 형식은 16진수 문자열로 보고될 수 있습니다. 이 경우 데이터 길이는 2에서 8 사이의 값이 될 수 있습니다.

예를 들어, 레지스터 0의 내용은 0x1234, 레지스터 1은 0x5678, 레지스터 2는 0x90AB, 레지스터 3은 0xCDEF입니다.

다음과 같은 JSON 데이터가 전송됩니다: {power=0x1234567890ABCDEF}

3.26 시간대 설정

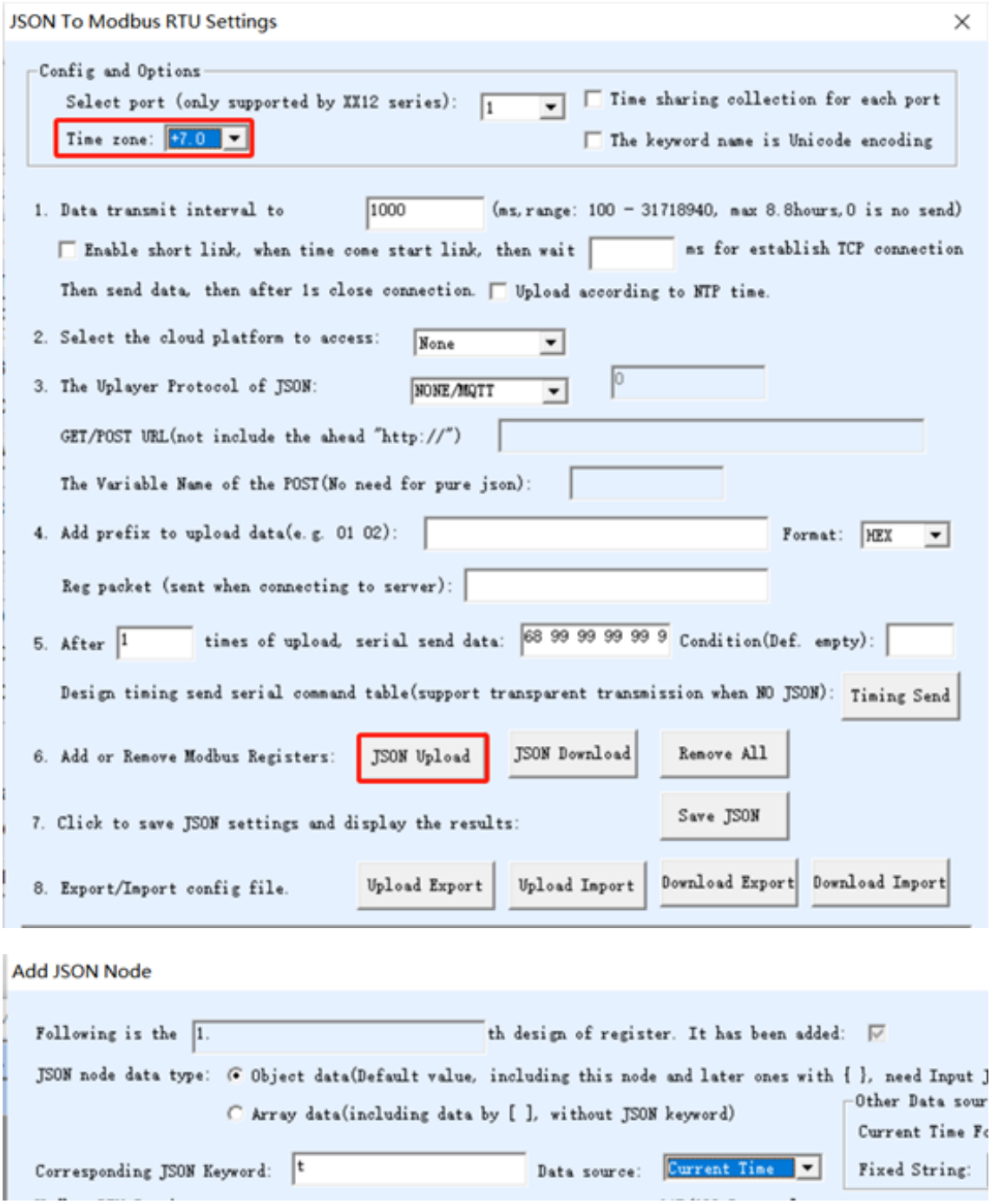


그림 50. 16진수 보고 설정

JSON에서 전송 시간 t를 설정한 다음 7번째 시간대를 선택하면 전송 시간은 베이징 시간에서 1시간을 뺀 시간이 됩니다. 총 48개의 시간대가 있으며 각 시간대는 0.5시간 간격입니다. 베이징은 동8구에 해당합니다.

3.27 DTL-645가 수신한 FE 수

DTL-645에서 수신한 FE의 개수가 확실한 경우, 아래 그림과 같이 선택할 수 있습니다. 확실하지 않은 경우 모든 FE를 선택할 수 있습니다. 그러나 "모두"를 선택하면 비교 속도가 느려지므로, 정확한 값을 선택하는 것이 좋습니다.

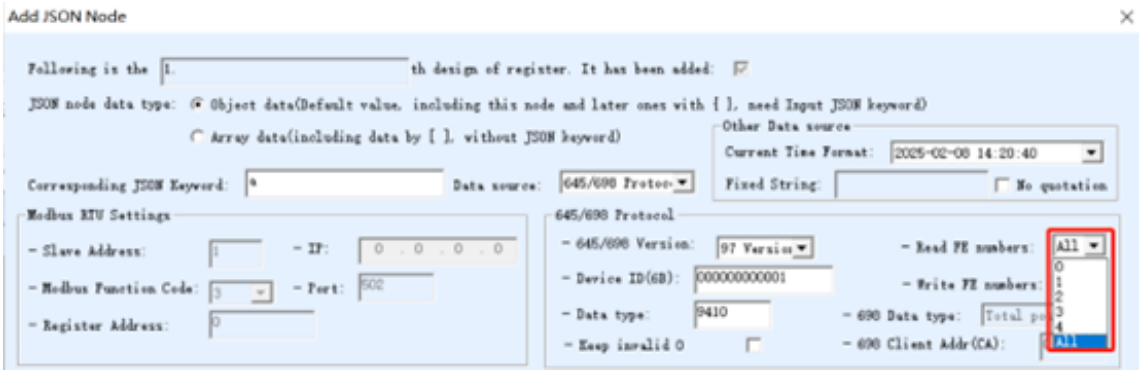


그림 51. 16진수 보고 설정

3.28 주기적 명령 전달

자세한 내용은 타이밍 명령 전달을 참조하십시오.

3.29 JSON 및 등록 패키지

그림 52. JSON 대화 상자에서 등록 패키지 설정하기

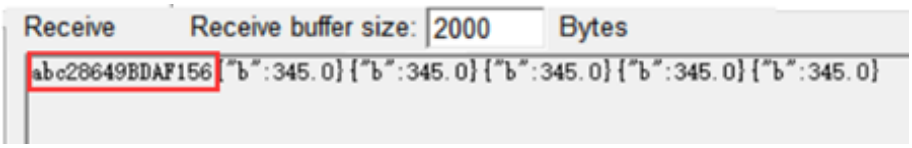


그림 53. 서버는 등록 패키지를 수신합니다.

JSON 전송 외에도 TCP 연결 설정 및 등록 패킷 전송을 지원합니다. 기본 등록 패킷 형식은 공백을 포함한 61, 62, 63과 같은 16진수입니다. MAC 주소는 MAC[6...17] 형식으로 추가할 수 있습니다. 문자를 입력할 경우 오른쪽에서 ASCII 모드를 선택하십시오.

3.30 698 프로토콜을 JSON으로 변환

"JSON 형식으로 645미터를 전송하는 방법"을 참조하세요.

3.31 장치 오프라인 시 NULL 데이터 전송 설정

Mobus를 JSON으로 변환할 때, 데이터 비트 유형이 정수 또는 부동 소수점인 경우, 장치가 오프라인 상태일 때 0 대신 NULL을 전송할 수 있습니다. 실제 데이터에 0이 포함될 수 있기 때문에 0 데이터와 장치 오프라인 상태를 구분하기 어렵기 때문입니다. 현재 부울 유형, DLT-645 및 DLT-698은 이 기능을 지원하지 않습니다. 이 기능을 사용하려면 NXT_Vircom 6.38 이상 버전이 필요합니다. 새로운 펌웨어 지원이 필요합니다. 예를 들어, 5144J는 펌웨어 1.526(7044).bin 이상 버전을 사용합니다.

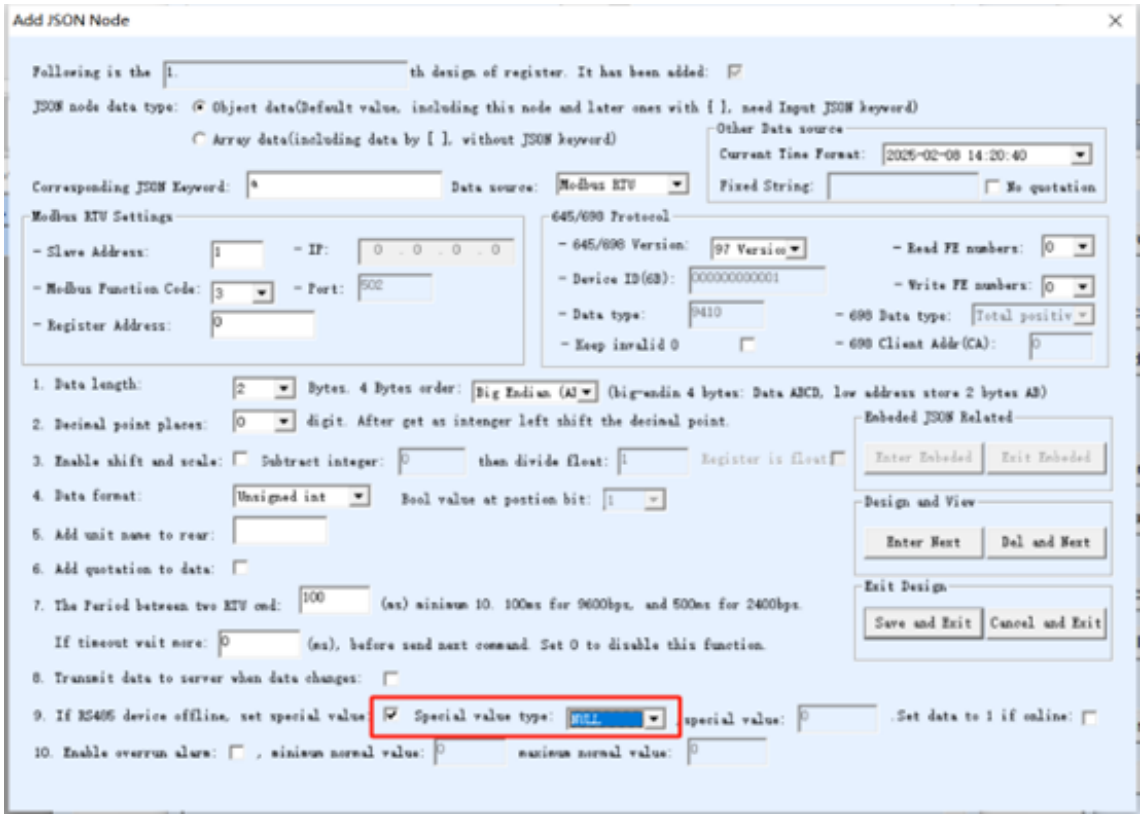


그림 54. NULL 값을 오프라인으로 전송하기 위한 설정

그림에서 보는 바와 같이 오프라인에서 NULL 값을 전송해야 하는 경우, 먼저 "RS458 장치 오프라인 특수값 설정" 기능을 확인한 후 특수값 목록에서 NULL을 선택하십시오. 기본값은 0입니다.

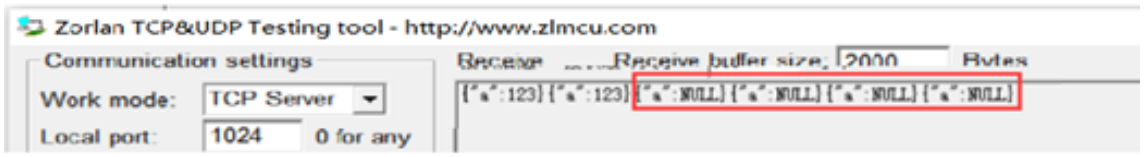


그림 55. NULL을 오프라인으로 전송했을 때의 효과

위 그림은 장치가 오프라인에서 NULL을 전송할 때의 효과를 보여줍니다.
내보낸 CSV 파일에서 "기능 옵션" 값의 다섯 번째 비트가 1이면 오프라인에서 NULL을 전송하는 기능이 포함되어 있음을 의미합니다. 아래 그림을 참조하십시오.

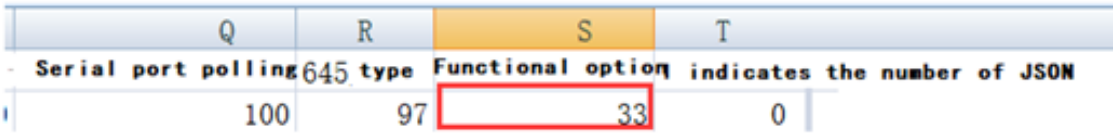


그림 56. 설정 파일에서 오프라인으로 전송될 값을 NULL로 설정합니다.

이 중 33개는 5번째 비트가 1입니다.

참고:
1. 데이터가 정수이고 소수점 유지 옵션이 0이 아닌 경우, 전송되는 데이터는 NULL입니다. 소수점이 뒤에 추가됩니다. 그 외의 경우 및 부동 소수점 모드에서는 상위 데이터가 직접 NULL로 전송됩니다.
2. 다양한 형식을 지원합니다.
3. 연속 레지스터의 결합 조회를 지원합니다.

3.31 변경 시에만 업로드하세요.

데이터 변경 시 업로드'를 선택하면 데이터 변경 시 업로드가 활성화됩니다. 단, 주기적 보고는 취소해야 합니다. 주기적 보고 시간에서 0을 선택하십시오(NXT_Vircom.53 이상 버전 필요). 이렇게 하면 데이터가 변경될 때만 업로드가 활성화됩니다. 제공된 설정이 성공적으로 적용되었는지 여부를 나타내는 데 사용할 수 있습니다.

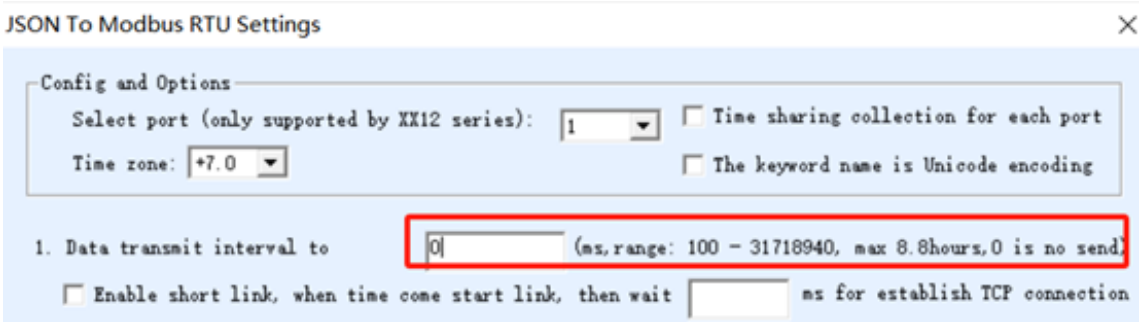


그림 57. 정기 보고를 취소합니다

4. 다양한 데이터 형식

현재 지원되는 데이터 형식은 부호 없는 정수, 부호 있는 정수, 부동 소수점, 부울, 16진수 및 기타입니다. 자세한 내용은 다음과 같습니다.
값이 4바이트 이상인 경우, 시스템은 ABCD, CDAB, BACD, CDBA 형식을 지원합니다. 여기서 A, B, C, D는 모두 0xE1 및 0x2F와 같은 8비트(1바이트) 16진수입니다.

4.1 부호 없는 정수

부호 없는 정수는 0보다 크거나 같은 값을 갖는 숫자입니다.
단순 부호 없는 정수는 다음과 같이 설정됩니다.

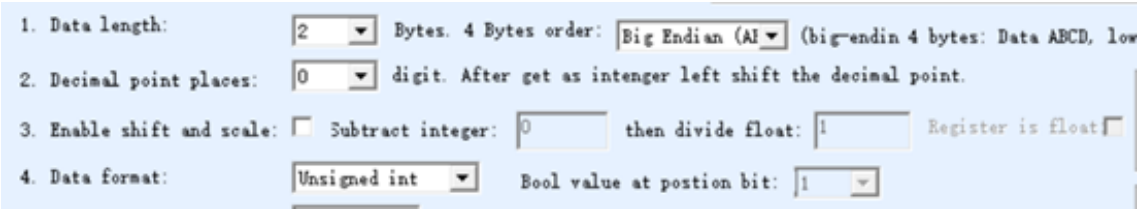


그림 58. 소수점이 없는 부호 없는 정수 2바이트

두 레지스터를 읽으면 레지스터 내용은 123과 456입니다. 이전 데이터는 {"uint1":123,"uint2":456}입니다. 데이터를 시프트해야 하는 경우 시프트할 비트 수는 1에서 5까지입니다. 위 그림에서 "소수점 유지"를 설정하십시오. 이 값이 4로 설정되면 업로드된 데이터는 {"uint1":0.0123,"uint2":0.0456}입니다. 옵션 3이 {"uint1":0.123,"uint2":0.456}인 경우, 옵션 2가 {"uint1":1.23, "uint2":4.56}입니다.
소수점 끝단: 2바이트 교환 형식, 즉 BA 형식입니다. 이 경우 Big Endian Swap은 선택할 수 있지만 Little Endian Swap은 선택할 수 없습니다.

이제 데이터 길이를 4바이트로 변경합니다. reg1은 여전히 123이고 reg2는 여전히 456입니다. 두 번째 레지스터의 시작 주소를 1씩 증가시키는 대신 2씩 증가시키도록 변경할 수 있습니다. 위의 데이터는 다음과 같습니다. {"uint1":123,"uint2":456}
실제 데이터 상황에 따라 빅 엔디안, 스몰 엔디안, 빅 엔디안, 스몰 엔디안의 네 가지 형식을 선택할 수 있습니다. 네 가지 형식을 혼합하여 사용하는 것도 지원합니다. 데이터가 2바이트 형식(Big Endian)이 아닌 경우, 4바이트 이상의 데이터만 Little Endian 형식으로 선택할 수 있습니다. BA 형식의 경우 빅 엔디안 형식을 선택할 수 있지만 스몰 엔디안 형식은 선택할 수 없습니다.

8바이트의 부호 없는 정수는 지원되지 않으며, 8바이트는 부동 소수점 숫자에만 사용할 수 있습니다.
장치가 오프라인이거나 레지스터 값이 반환되지 않을 경우 0 또는 NULL로 설정하는 것을 지원합니다. NULL로 설정하면 실제 값 0과 오프라인 값 0을 구분할 수 있습니다. 각 노드는 오프라인이 아닌 경우의 설정값(원래 값 유지), 오프라인 설정값 0, 오프라인 업로드 값 NULL을 개별적으로 설정할 수 있습니다.

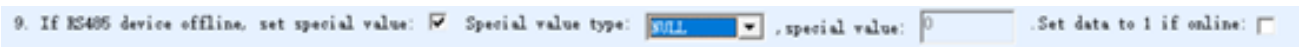


그림 59. 오프라인에서 0 또는 NULL로 값을 초기화하는 기능을 지원합니다.

지원하는 최대 노드 수: 2바이트 및 4바이트 모두 최소 1000개의 JSON 노드를 지원합니다. 이 수는 레지스터 연속 노드와 부분 연속 노드를 모두 지원합니다.

4.2 부호 있는 정수

부호 있는 정수와 부호 없는 정수의 사용법은 거의 동일합니다. 다만, 읽은 데이터의 최상위 자릿수가 1이면 음수를 나타냅니다.

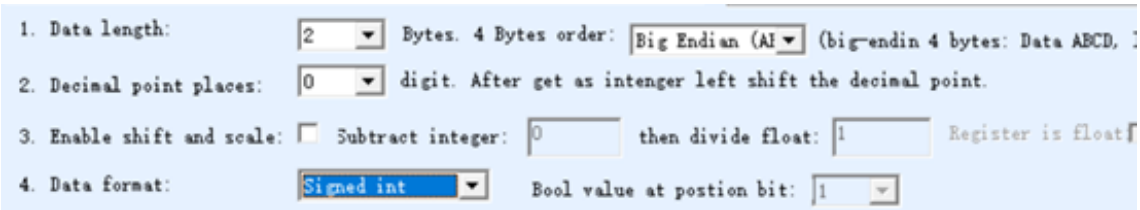


그림 60. 소수점 없는 2바이트 부호 있는 정수

반환된 데이터는 다음과 유사합니다: {" int1 ": - 123," int2 ": 0.456," int3 ": 0.00789}
지원하는 최대 노드 수: 2바이트 및 4바이트 모두 최소 1000개의 JSON 노드를 지원합니다. 이 수는 레지스터 연속 노드와 부분 연속 노드를 모두 지원합니다.

4.3 부동 소수점 유형

부동 소수점이란 읽은 값이 4바이트의 부동 소수점 데이터 또는 8바이트의 배정밀도 데이터임을 의미합니다. 간단한 부동 소수점 유형 설정은 다음과 같습니다.

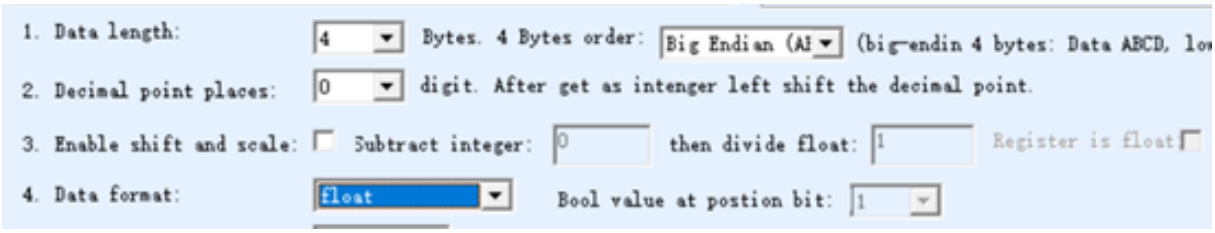


그림 61. 소수점 없는 2바이트 부호 없는 정수

세 개의 레지스터를 읽으면 레지스터 내용은 1.234567, 2.345678, 3.456789가 됩니다. 소수점 이하 자릿수가 각각 0, 3, 5인 경우, 앞의 데이터는 {"float1":1,"float2":2.346,"float3":-3.4568}입니다. 첫 번째 레지스터의 원래 값은 1.234567이고 소수점 이하 자릿수는 0이므로 1입니다. 두 번째 숫자는 소수점 이하 3자리를 유지하므로 2.346입니다(여기서 3번째 소수점 자리는 반올림됨). 세 번째 숫자는 5로 설정되지만, 여기서는 최대 4자리까지만 가능하므로 4자리만 유지됩니다. 부동 소수점 최대값은 4294967296.0입니다.
실제 데이터 상황에 따라 큰 소수점 교환, 작은 소수점 교환, 큰 소수점 교환, 작은 소수점 교환의 네 가지 형식을 선택할 수 있습니다. 또한 네 가지 형식의 혼합 사용을 지원합니다.

기기가 오프라인 상태이거나 레지스터 값이 반환되지 않을 경우 0 또는 NULL로 설정하는 기능을 지원합니다. NULL로 설정하면 실제 값 0과 오프라인 값 0을 구분할 수 있습니다. 각 노드는 오프라인이 아닌 경우 설정값(원래 값 유지), 오프라인 설정값 0, 오프라인 업로드 값 NULL을 개별적으로 설정할 수 있습니다.

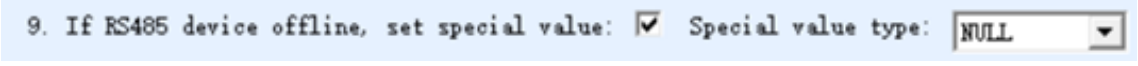


그림 62. 오프라인에서 0 또는 NULL로 값을 초기화하는 기능을 지원합니다.

8바이트의 배정밀도 부동소수점 형식을 지원합니다. 데이터가 8바이트일 경우, 업로드되는 데이터의 소수점은 최대 4자리까지 가능합니다.

지원하는 최대 노드 수: 단정밀도 및 배정밀도 모두 최소 1000개의 JSON 노드를 지원합니다. 이 수는 레지스터 연속 노드와 부분 연속 노드를 모두 지원합니다.

4.4 16진수 형식입니다.

16진수를 선택하면 Modbus에서 반환되는 데이터에 따라 16진수로 데이터가 업로드되며, 부동 소수점 및 정수 형식에 대한 분석은 수행되지 않으므로 일부 비표준 형식을 업로드하는 데 적합합니다.

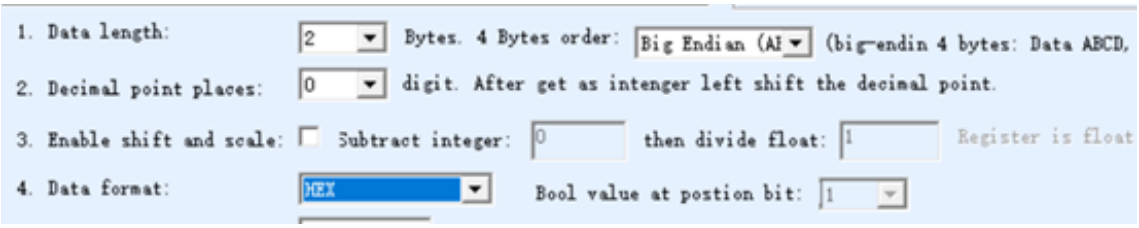


그림 63. 16진수 2바이트

레지스터 내용이 3FF3일 경우, {"hex":0x3FF3}이 업로드됩니다. 2바이트, 4바이트, 8바이트 데이터를 지원합니다. 4바이트의 경우 형식은 {"hex":0x11112222}이며, 여기서 1111은 하위 레지스터의 내용을 나타냅니다. 2222는 하위 레지스터 내용에 +1을 더한 값입니다. 8바이트의 경우에도 마찬가지입니다.

4.5 불리언 타입

불리언 타입: 03 및 04 기능 코드로 읽은 2바이트 데이터의 비트가 1인지 여부를 나타냅니다. 비트가 1이면 1이 업로드되고, 그렇지 않으면 0이 업로드됩니다. 01 또는 02 기능 코드를 사용하는 경우 데이터 타입은 자동으로 불리언이 됩니다.

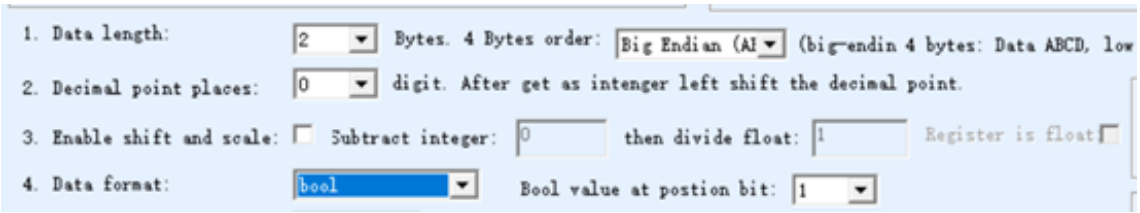


그림 64. 부울 타입 2바이트

4.6 JSON 노드 수

새 버전(2023년 10월 27일 이후)은 8308의 경우 1600개 노드, 8305의 경우 2700개 노드를 포함합니다.

5. 데이터 형식 제공

기본적으로 전송 데이터의 JSON 이름이 전송된 데이터와 동일하면 전송된 매개변수가 사용되고 전달된 매개변수는 유효하지 않게 됩니다. 소수점 이동, 변환 배율, 03 함수 코드 BOOL 유형 등 더 많은 매개변수를 설정할 수 있습니다.
현재 전달 시에는 데이터 길이, 형식 및 유형만 구성할 수 있습니다.

5.1 Modbus는 코일을 설정합니다.

꺼짐으로 설정하는 방법:

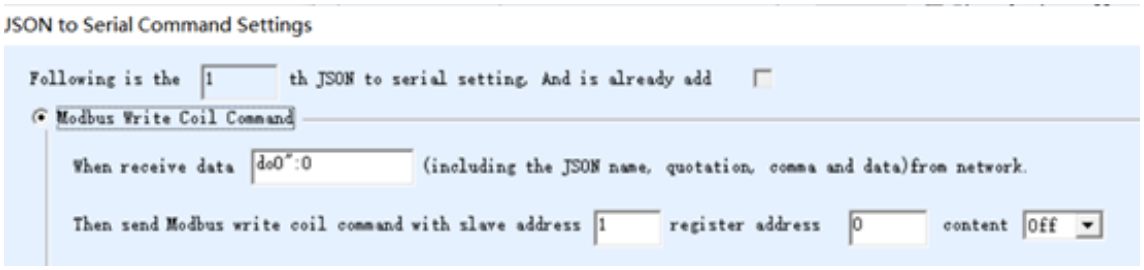


그림 65. 관습 설정

켜짐으로 설정하는 방법:

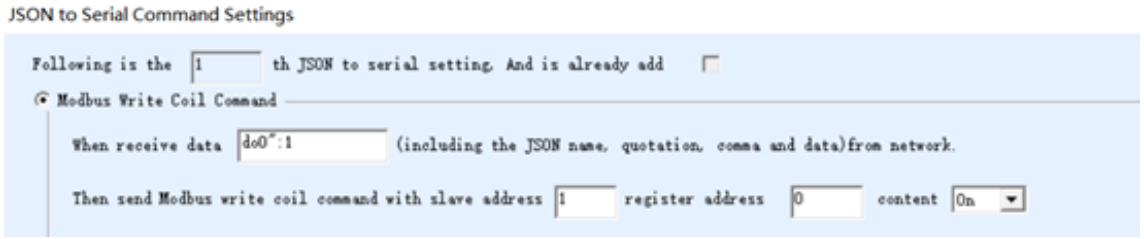


그림 66. 설정

5.2 Modbus 레지스터 설정

레지스터 설정의 기본 모드는 아래와 같습니다.

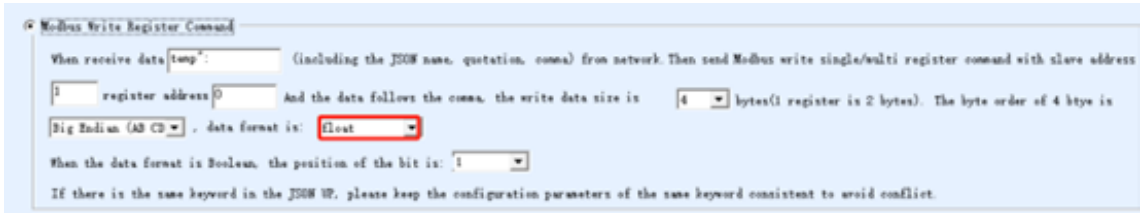


그림 67. 레지스터 설정

지원되는 기타 모드 목록은 다음과 같습니다.

Number of bytes	Big and small end	Data format	Instructions
2	/	unsigned integer	Send-1 is also supported, and the actual value is 65535
2	/	signed integer	No different than an unsigned integer
4	AB CD	unsigned integer	No different than an unsigned integer
4	CD AB	unsigned integer	No different than an unsigned integer
4	BA DC	unsigned integer	No different than an unsigned integer
4	DC BA	unsigned integer	No different than an unsigned integer
4	AB CD	Floating point type	
4	CD AB	Floating point type	
4	BA DC	Floating point type	
4	DC BA	Floating point type	
8	AB CD	Floating point type	
8	CD AB	Floating point type	
8	BA DC	Floating point type	
8	DC BA	Floating point type	
/	/	HEX	The HEX type is invalid. It is actually a signed

6. MQTT

MQTT는 단독으로 사용하거나 JSON 기능과 함께 사용할 수 있습니다. 단독으로 사용할 경우, MQTT 기능은 시리얼 데이터를 MQTT 서버로 투명하게 전송합니다. 즉, 시리얼 포트로 수신된 데이터가 MQTT 페이로드로 사용됩니다. 동시에 MQTT 로드는 시리얼 포트 출력을 통해 투명하게 전송됩니다. 시리얼 포트에서 MQTT로의 전송을 구현합니다.

6.1. 장치 구성

먼저 기기를 검색한 다음 '기기 편집'을 클릭하세요.

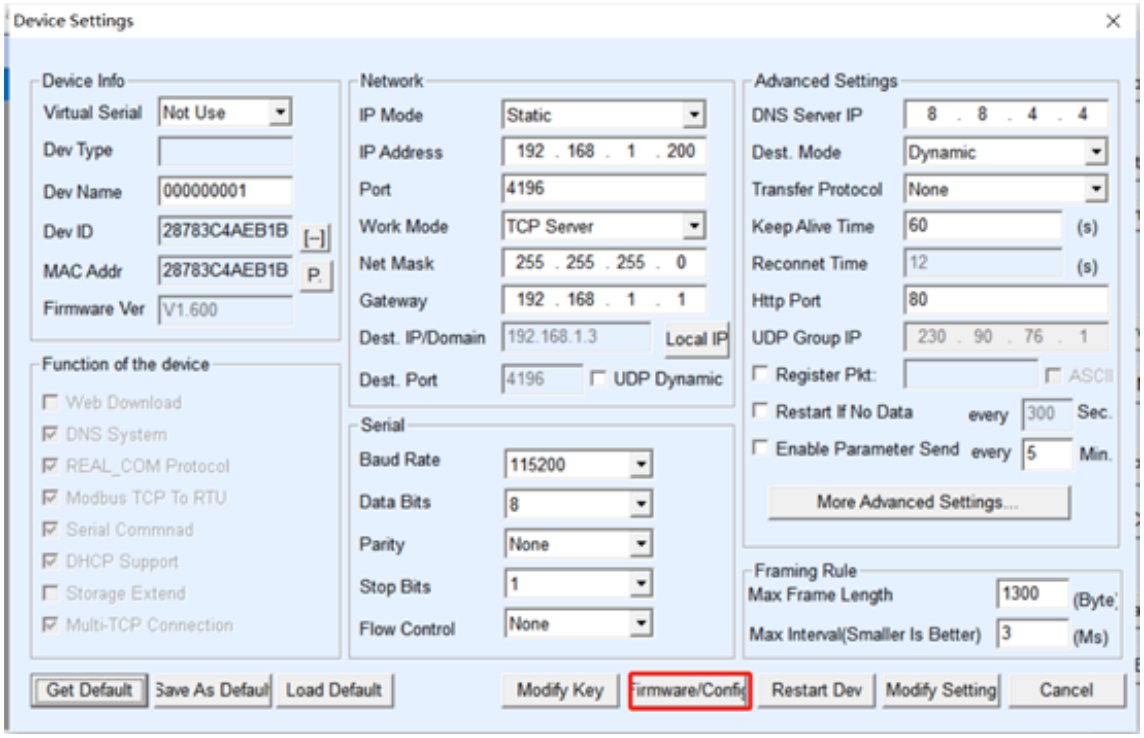


그림 68. MQTT 구성 1

"펌웨어 및 구성"을 클릭하면 구성 다운로드 및 디자인 대화 상자가 나타납니다.

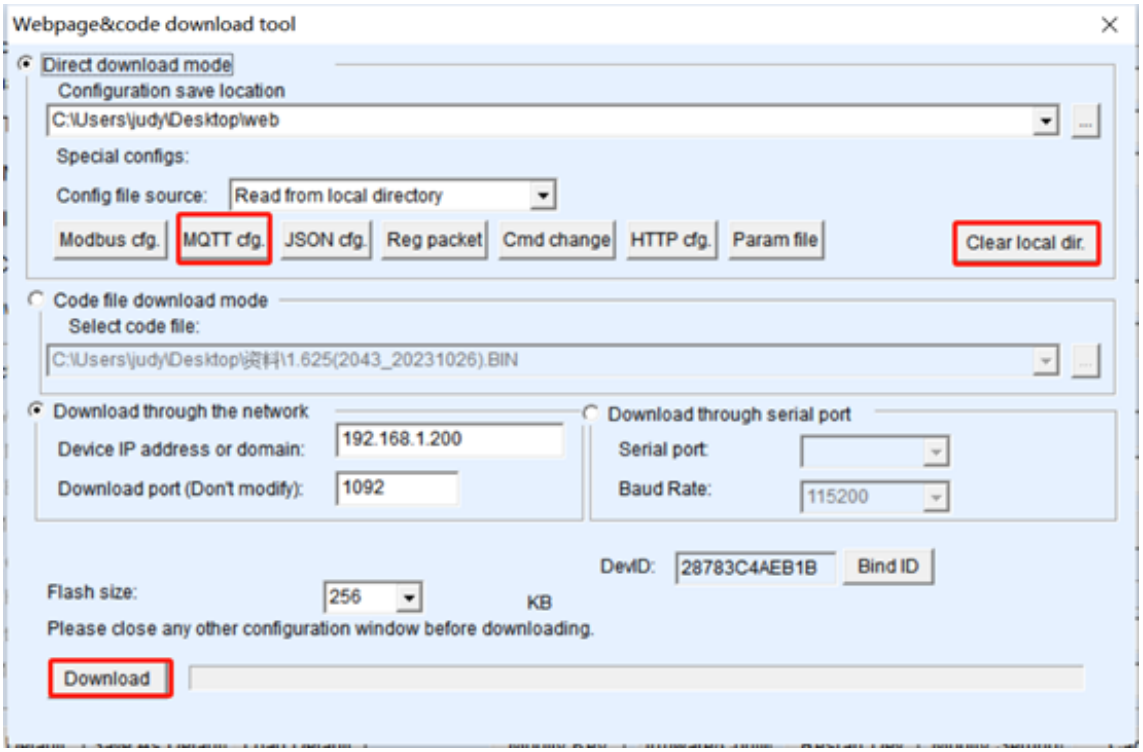


그림 69. MQTT 구성 2

여기서 "웹 디렉터리 다운로드"를 선택한 다음 MQTTHTTPD 디렉터리와 같은 빈 디렉터리를 선택하고 "모두 지우기"를 클릭하여 이전 설정을 삭제합니다(이전 설정이 JSON 형식으로 되어 있는 경우 "모두 지우기"를 클릭하지 마십시오. 그렇지 않으면 이전 JSON 설정이 삭제됩니다). 그런 다음 "MQTT 구성"을 클릭합니다.

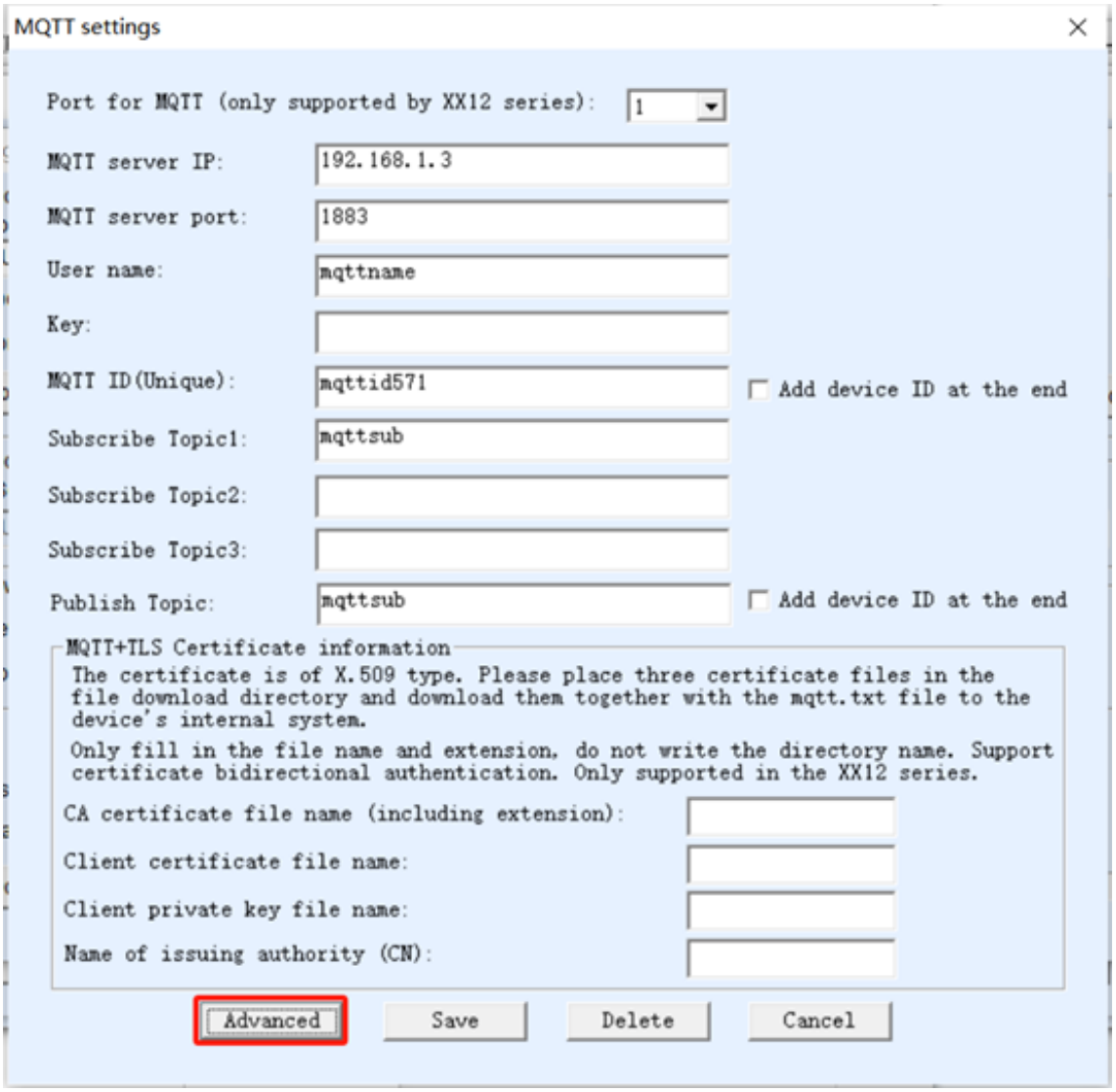


그림 70. MQTT 구성 3

설정 방법은 다음과 같습니다.

1. 서버 도메인 이름 또는 IP 주소: MQTT 서버의 IP 주소를 입력하십시오. 최대 길이는 30자입니다.
2. 사용자 이름: MQTT 서버의 사용자 이름을 입력하십시오.
3. 비밀번호: 사용자의 로그인 비밀번호를 입력하십시오.
4. 클라이언트 ID: MQTT의 클라이언트 ID를 입력하십시오.
5. 구독 테마: 이 장치가 구독하는 테마입니다. 다른 장치가 이 테마를 게시하면 서버는 이 장치로 전송합니다. 게시만 하는 경우 이 필드를 입력할 필요가 없습니다.
6. 게시 토픽: 장치의 시리얼 포트가 MQTT로 변환될 때 서버로 전송되는 데이터의 토픽입니다.
7. MQTT 고급 매개변수: 고급 매개변수를 구성하는 데 사용됩니다.
8. MQTT 설정 저장: 이 버튼을 클릭하여 설정을 저장한 다음 웹 다운로드 디렉토리에서 "다운로드" 버튼을 클릭하여 다운로드하십시오.

먼저 "MQTT 고급 매개변수"를 클릭하세요(일반적으로 고급 매개변수는 설정할 필요가 없습니다).

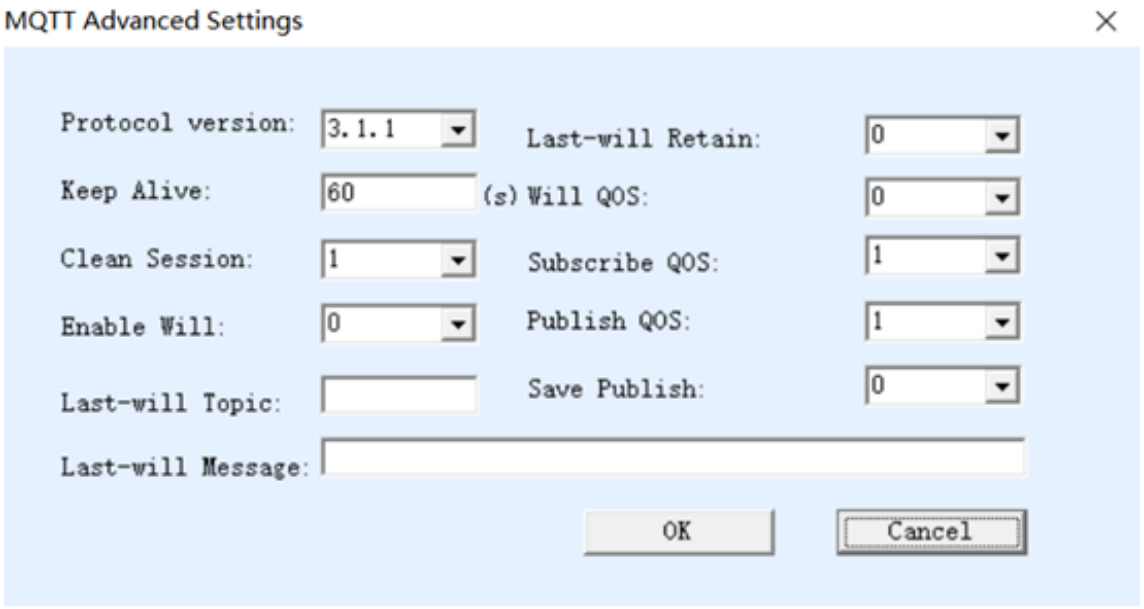


그림 71. MQTT 고급 파라미터 구성

각 항목에 대한 설명은 다음과 같습니다.

1. 프로토콜 버전: 현재 주류 버전은 3.1.1입니다. 3.1 버전을 선택해야 하는 경우 여기에서 선택하십시오.
2. Keepalive 시간: MQTT의 하트비트 시간입니다. 최소 10초이며 기본값은 60초입니다.
3. 서버 구독 정보 삭제: 클라이언트 연결이 끊어진 후 서버에서 구독 정보를 삭제할지 여부를 지정합니다.
4. 마지막 소원 메시지 활성화 여부: 마지막 소원 메시지를 표시할지 여부를 지정합니다.
5. 마지막 소원 메시지 테마: 마지막 소원 메시지 테마입니다.
6. 마지막 소원 메시지 정보: 마지막 소원 메시지에 대한 정보입니다.
7. 소원 메시지 저장: 클라이언트가 오프라인 상태일 때 서버에서 클라이언트에게 소원 메시지를 보낼지 여부를 지정합니다.
8. 소원 메시지 품질: 서버에서 전송하는 마지막 소원 메시지의 전달 품질 수준입니다.
9. 구독 품질: 구독의 전달 품질 수준입니다. 재전송으로 인한 연결 끊김을 방지하기 위해 경우에 따라 값을 0으로 설정해야 합니다.
10. 게시 품질: 클라이언트가 게시한 메시지의 전달 품질 수준입니다. 경우에 따라 재전송으로 인한 연결 끊김을 방지하기 위해 값을 0으로 설정해야 합니다.
11 게시 저장 여부: 서버가 마지막 메시지(새 클라이언트 구독이 있는 경우 클라이언트로 전송된 메시지)를 보관할지 여부.

여기서는 고급 매개변수를 수정하지 않습니다. "MQTT 설정 저장"을 바로 클릭한 다음 "다운로드"를 클릭하세요.

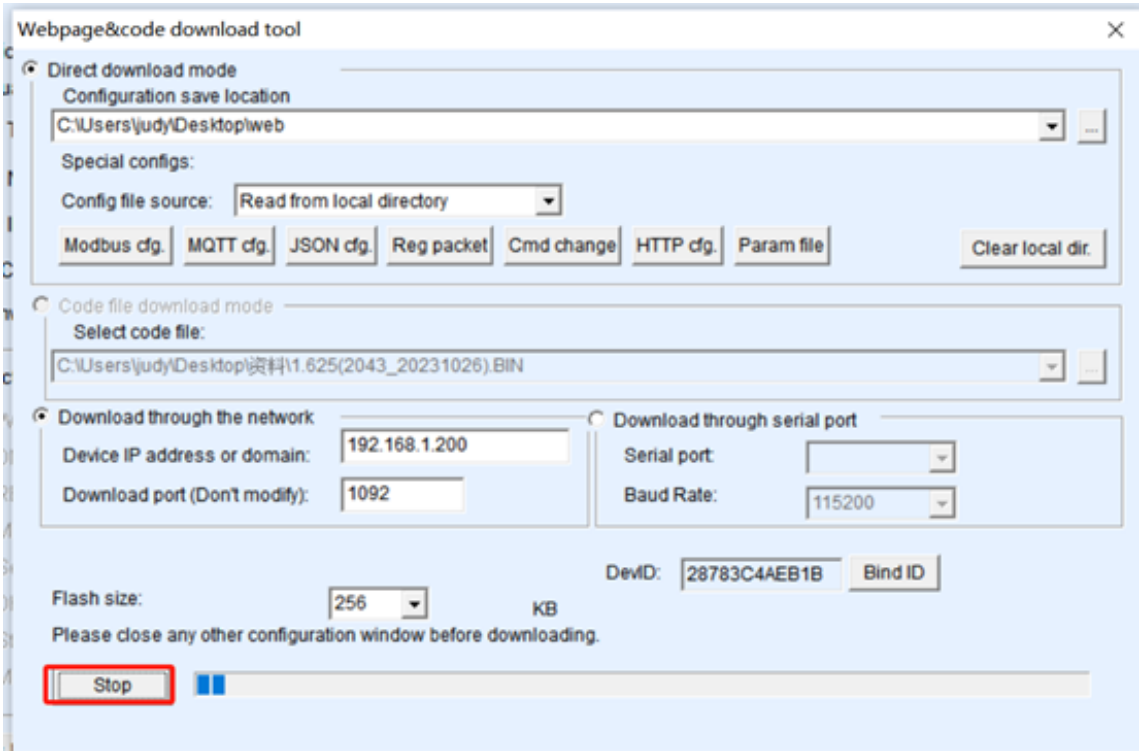


그림 72. 다운로드

다운로드가 완료되면 확인을 클릭하십시오. 그러면 장치 관리 대화 상자로 돌아가 장치의 대상 IP, 작동 모드 및 대상 포트가 MQTT 설정으로 자동으로 변경된 것을 확인할 수 있습니다.

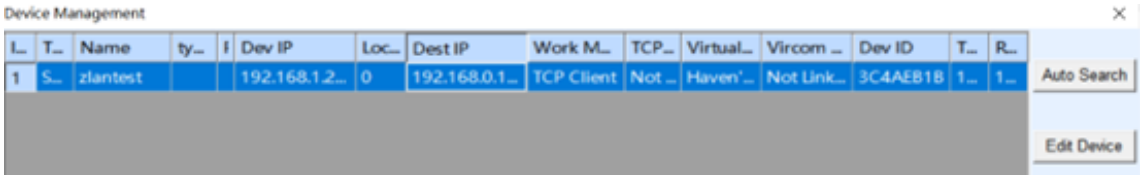


그림 73. 자동 수정

아니오인 경우, 장치 편집 대화 상자에서 대상 IP 주소, 작동 모드 및 대상 포트를 설정해야 합니다. 그런 다음 "설정 수정"을 클릭하십시오.

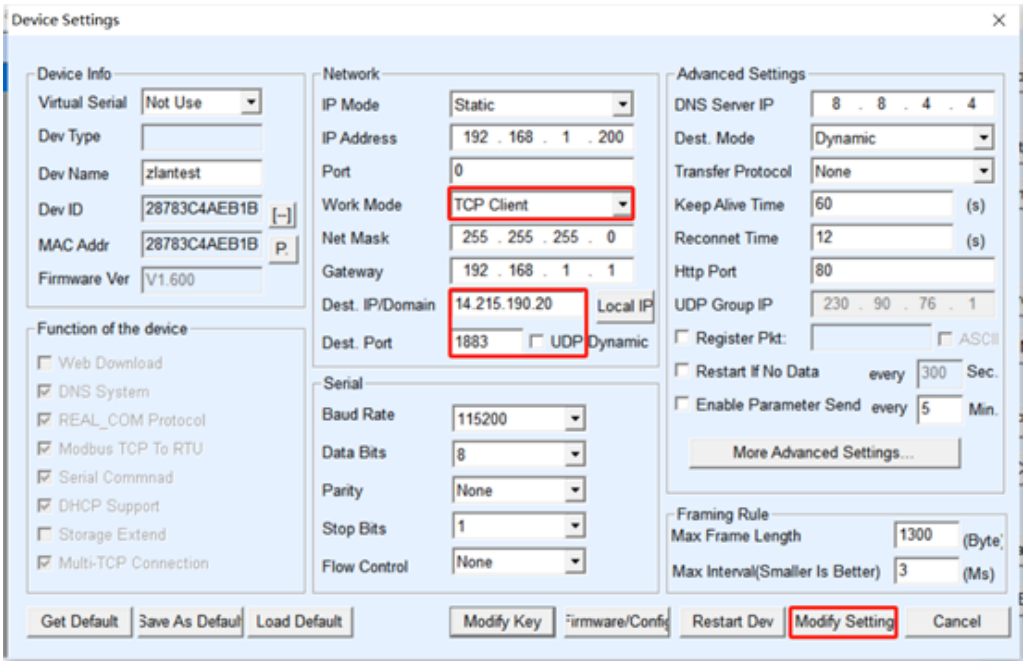


그림 74. IP 구성

이로써 설정이 완료되었습니다.

6.2 데이터 테스트

연결이 완료되면 장치의 LINK 표시등(일반적으로 가운데 파란색 표시등)이 켜집니다. 장치가 MQTT 서버에 연결되었습니다. 이제 시리얼 툴을 엽니다.

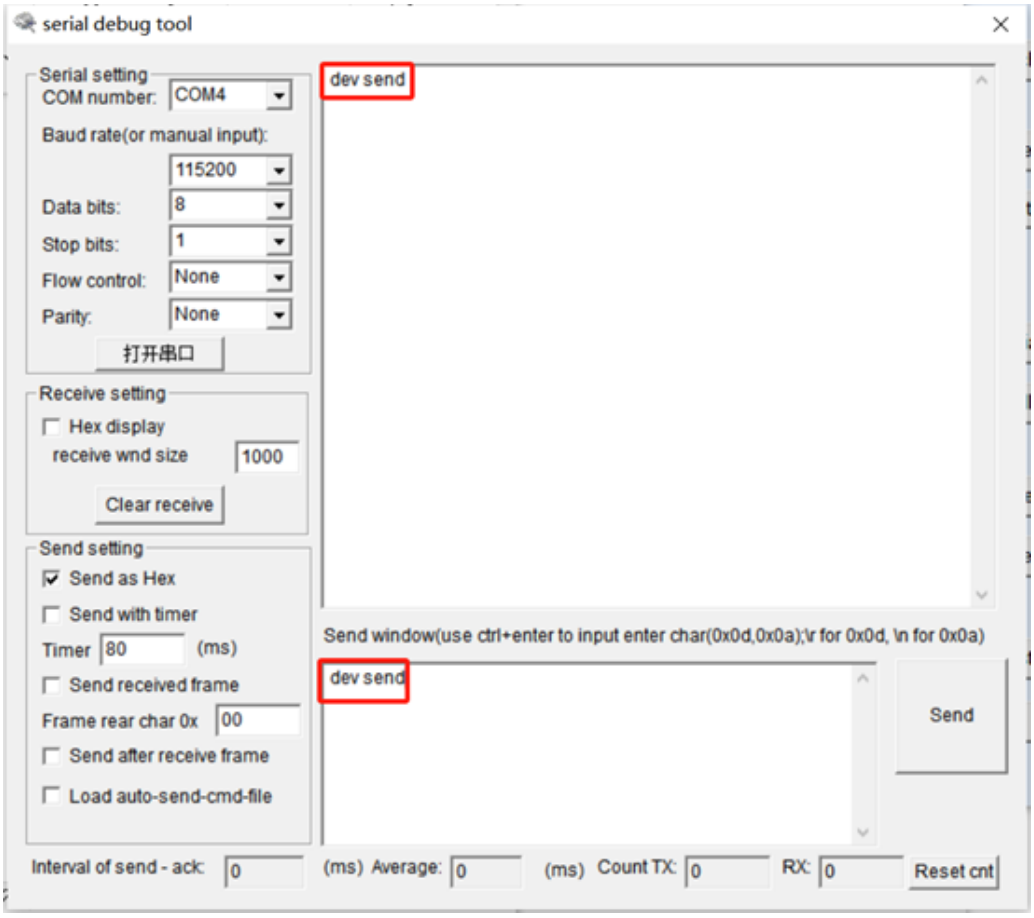


그림 75. 시리얼 포트 송수신

기기와 동일한 전송 속도로 시리얼 포트를 열고 `dev send`를 입력합니다. 수신 창에 `dev send`가 표시되는 이유는 기기가 MQTT 서버에 `zlansub`라는 주제로 정보를 전송했기 때문입니다. 우리 기기도 `zlansub` 주제를 구독하고 있었기 때문에 서버는 즉시 우리에게 구독 메시지를 전송했고, 이 구독 메시지의 내용이 바로 `dev send`였습니다. 이 정보는 MQTT 페이로드로 다운로드되어 투명 전송 방식을 통해 시리얼 포트로 출력됩니다. 다른 기기에서 메시지를 전송하면 이 기기도 데이터를 수신할 수 있습니다.
일반적으로 사용자는 시리얼 포트 명령(예: Modbus RTU)을 MQTT 서버로 직접 전송할 수 있습니다. 또한 JSON 기능을 사용하여 Modbus RTU 형식을 자동으로 수집하고 JSON 형식을 지정할 수도 있습니다. 이 외에도 상하이 ZLAN에서 제공하는 일부 비표준 기기 및 호스트 컴퓨터 프로토콜 형식에 대한 사용자 정의 옵션을 활용할 수 있습니다.

7. MQTT+JSON을 Modbus RTU로 변환

위의 JSON과 MQTT를 조합하면 다음과 같은 기능을 구현할 수 있습니다.

1. MQTT 기반 프로토콜을 사용하여 서버와 연결하고, 구독 및 게시 방식으로 데이터 통신을 수행합니다.
2. Modbus RTU 레지스터를 독립적으로 설계하고 자동으로 수집할 수 있습니다.
3. 특정 Modbus 레지스터 내용을 JSON 형식으로 변환하여 정기적으로 전송할 수 있습니다.
4. 클라우드에서 장치를 쉽게 식별할 수 있도록 JSON 형식에 장치 ID를 추가할 수 있습니다.

MQTT+JSON을 이용한 Modbus RTU 변환 기능이 필요한 경우, MQTT와 JSON을 순서에 상관없이 별도로 설계할 수 있습니다. 디자인을 하나 완료한 후에는 "디자인 지우기" 버튼을 클릭하지 마시고, 두 번째 디자인이 완료되면 "다운로드" 버튼을 동시에 클릭하여 기기 콘텐츠를 다운로드하세요.
일반적으로 기기를 수동으로 재시작하고 설정을 불러올 수 있습니다.

8. HTTP POST/GET + JSON

호스트 프로토콜은 MQTT 외에도 HTTP 프로토콜을 선택할 수 있으며, POST 및 GET 명령어를 통해 데이터를 업로드할 수 있습니다. 다음은 POST 명령어를 예로 든 예입니다.
POST/GET+JSON 기능을 지원해야 하는 경우 NXT_Vircom 구성 도구에서 5.17 이상 버전을 선택하십시오. 또한 POST 명령어를 지원해야 하는 경우 2003 펌웨어는 5.81 이상이어야 합니다(GET만 지원하는 경우 JSON을 지원하는 일반 2003 펌웨어를 사용하십시오).

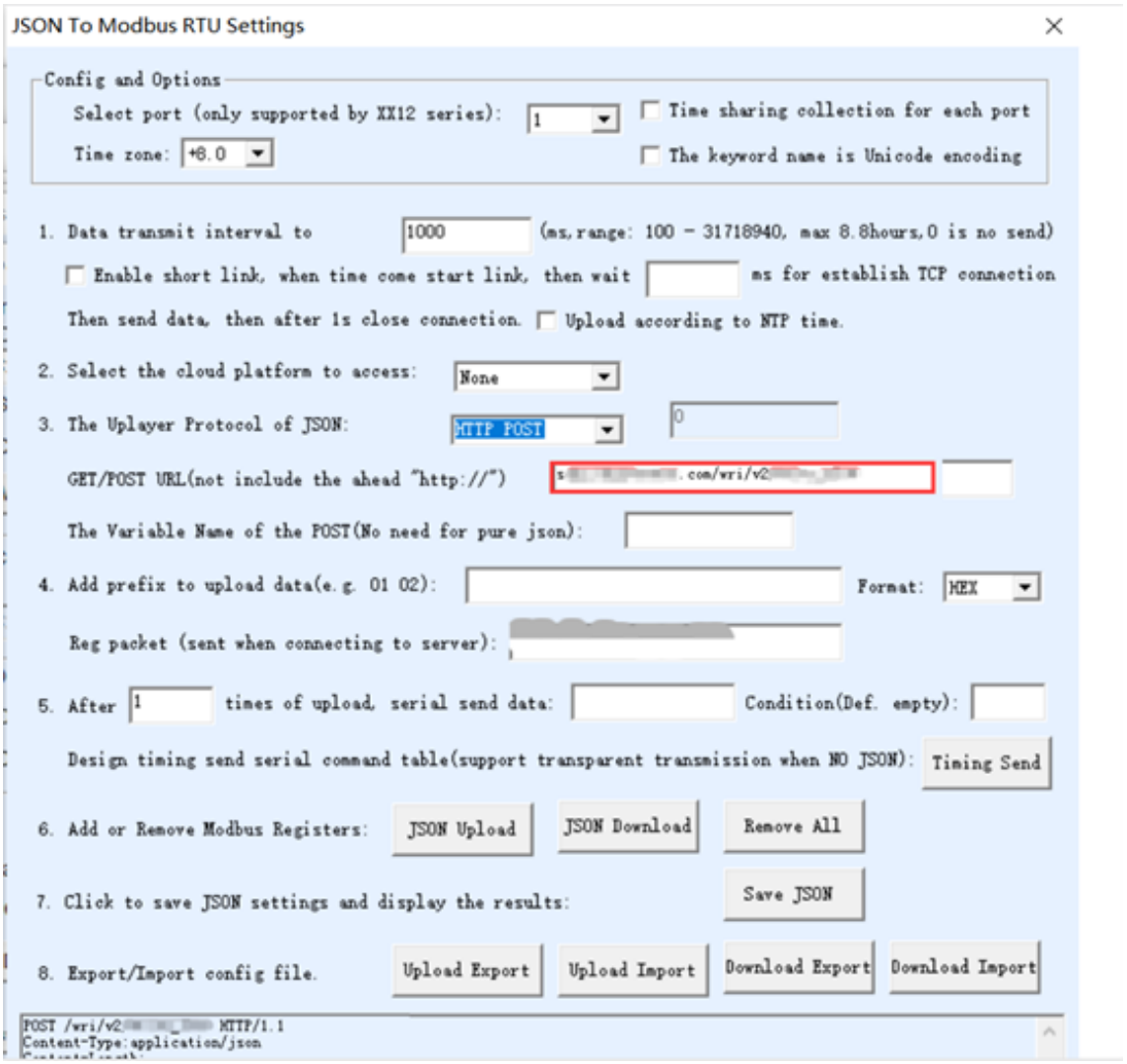


그림 76. POST+JSON

NXT_Vircom 5.17 버전에서는 Modbus RTU 설정에 JSON 관련 옵션이 두 가지 추가되었습니다(그림 참조).

1. JSON 상위 계층 프로토콜: 프로토콜을 사용하지 않거나 MQTT 프로토콜을 사용하는 경우 첫 번째 옵션인 "NONE/MQTT"를 선택합니다. HTTP POST를 사용하려면 두 번째 항목 "HTTP POST"를, HTTP GET을 사용하려면 세 번째 항목 "HTTP GET"을 선택합니다.
2. POST/GET URL: POST 또는 GET을 선택할 때 URL을 입력해야 합니다. 예를 들어 URL이 http://s.a.com/wri/v2인 경우 http://를 삭제하고 s.a.com/wri/v2를 직접 입력합니다.

다른 JSON 구조 설계 방법은 이전에 설명한 방법과 동일하며, "JSON 설정 저장" 버튼을 클릭하면 POST/GET을 선택한 경우 HTTP 전송 프로토콜을 지원하기 위해 JSON 데이터 앞에 HTTP 헤더 형식 정보가 추가됩니다.

이 POST/GET 설계 방식은 간단하고 실용적이며, HTTP POST/GET+JSON 방식을 통해 Modbus RTU와 같은 계측기 데이터를 서버로 간단하고 빠르게 전송할 수 있습니다.

9. 접두사에 +BASE64 인코딩을 추가합니다.

이제 일반 JSON 형식 앞에 비밀 필드와 타임스탬프 필드를 추가해야 합니다. 여기서 비밀 필드는 고정된 비밀번호이고, 타임스탬프는 현재 시간을 나타내는 10자리 타임스탬프입니다. 업로드된 모든 데이터는 Base64로 인코딩됩니다. 즉, 최종 업로드 형식은 base64Encode(secret + "" + timestamp + "" + json)입니다.

여기서는 NXT_Vircom 버전 6.73과 함께 5812LD 펌웨어 버전 1.481(2012L).bin을 소개합니다.

먼저 빈 로컬 폴더 디렉터리를 생성한 다음, NXT_Vircom에서 해당 디렉터리를 선택하여 구성 정보를 저장합니다.

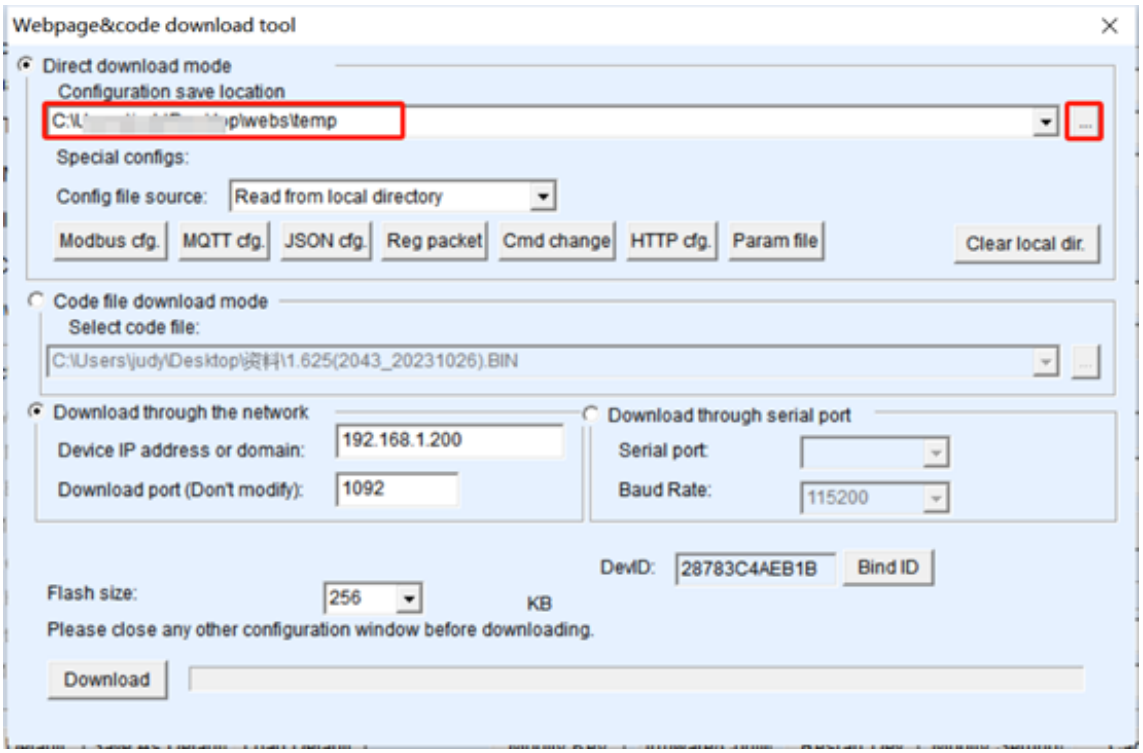


그림 77. 구성 디렉터리 선택

9.1 접두사 추가

위 그림에서 "JSON 구성"을 클릭하고 "데이터 업로드 프레임 헤더 추가" 필드에 다음 데이터를 입력하십시오.

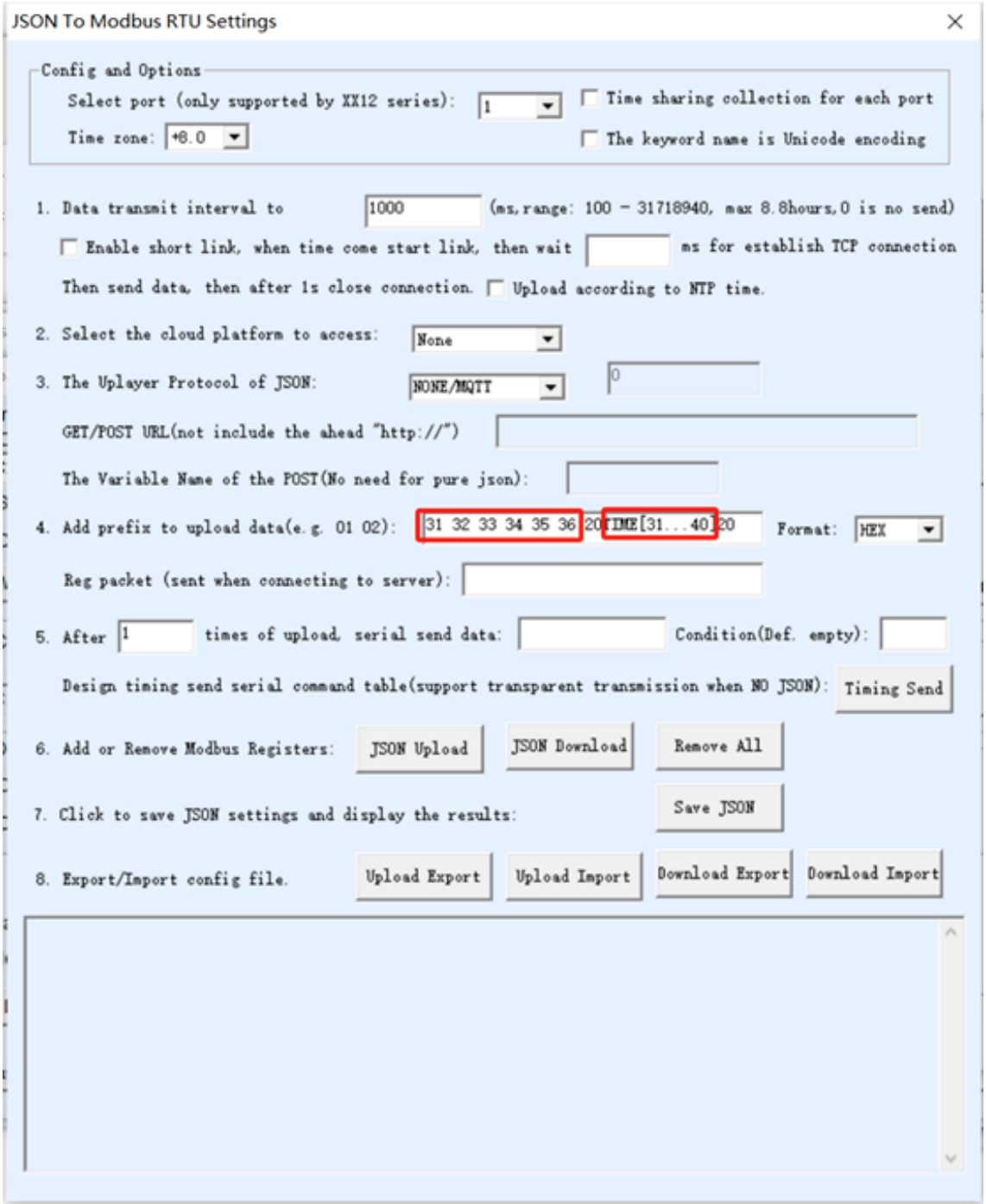


그림 78. 입력 프레임 헤더

여기서 31 32 33 34 35 36 20 TIME[31...40] 20은 다음과 같이 설명됩니다. 노란색 부분 31 32 33 34 35 36은 비밀 문자열의 16진수 표현입니다. 문자열을 직접 입력하려면 오른쪽에 있는 "format" HEX를 먼저 ASCII로 변경한 다음, 문자열 형식으로 비밀을 입력한 후 다시 HEX로 변경하면 비트 HEX 형식이 자동으로 변환됩니다. 녹색 20은 공백입니다. 파란색 TIME[31...40]은 10비트 타임스탬프입니다(NXT_Vircom 6.73 이상 버전이 필요합니다). 모든 필드는 공백으로 구분됩니다.
그런 다음 "JSON으로 전송"을 클릭하여 Modbus RTU를 JSON으로 구성합니다. 이 부분은 여기서 다시 설명하지 않겠습니다. 설정을 완료한 후 위의 인터페이스로 돌아가서 "JSON 설정 저장"을 클릭하면 됩니다.

9.2 BASE64 인코딩

BASE64 인코딩을 지원하려면 위의 내용을 바탕으로 위의 "웹 페이지/프로그램 다운로드 도구 구성" 대화 상자에서 "매개변수 파일" 버튼을 클릭하고, 팝업 대화 상자에 BASE64_EN을 입력한 다음 오른쪽에 1을 입력하십시오.

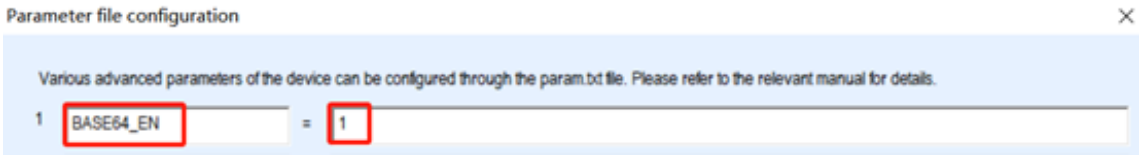


그림 79. BASE64 인코딩

그런 다음 "저장"을 클릭합니다. 이 설정은 param.txt 파일로 로컬 컴퓨터 디렉터리에 저장되고, base64 인코딩을 활성화하는 JSON 설정과 함께 장치로 다운로드됩니다.
BASE64_EN=1로 설정하면 MQTT 업로드 및 투명 업로드를 포함하여 장치의 업링크 데이터가 base64로 인코딩됩니다.
param.txt 파일이 없거나, BASE64_EN 설정이 없거나, BASE64_EN=0인 경우에는 이 기능이 활성화되지 않습니다.

9.3 MQTT 구성

MQTT 프로토콜을 지원하려면 위 내용을 바탕으로 위의 "설정 페이지/프로그램 다운로드 도구" 대화 상자에서 "MQTT 설정" 버튼을 클릭하고, 팝업 대화 상자에 사용자 고유의 MQTT 설정을 입력하십시오. 다음은 예시입니다.

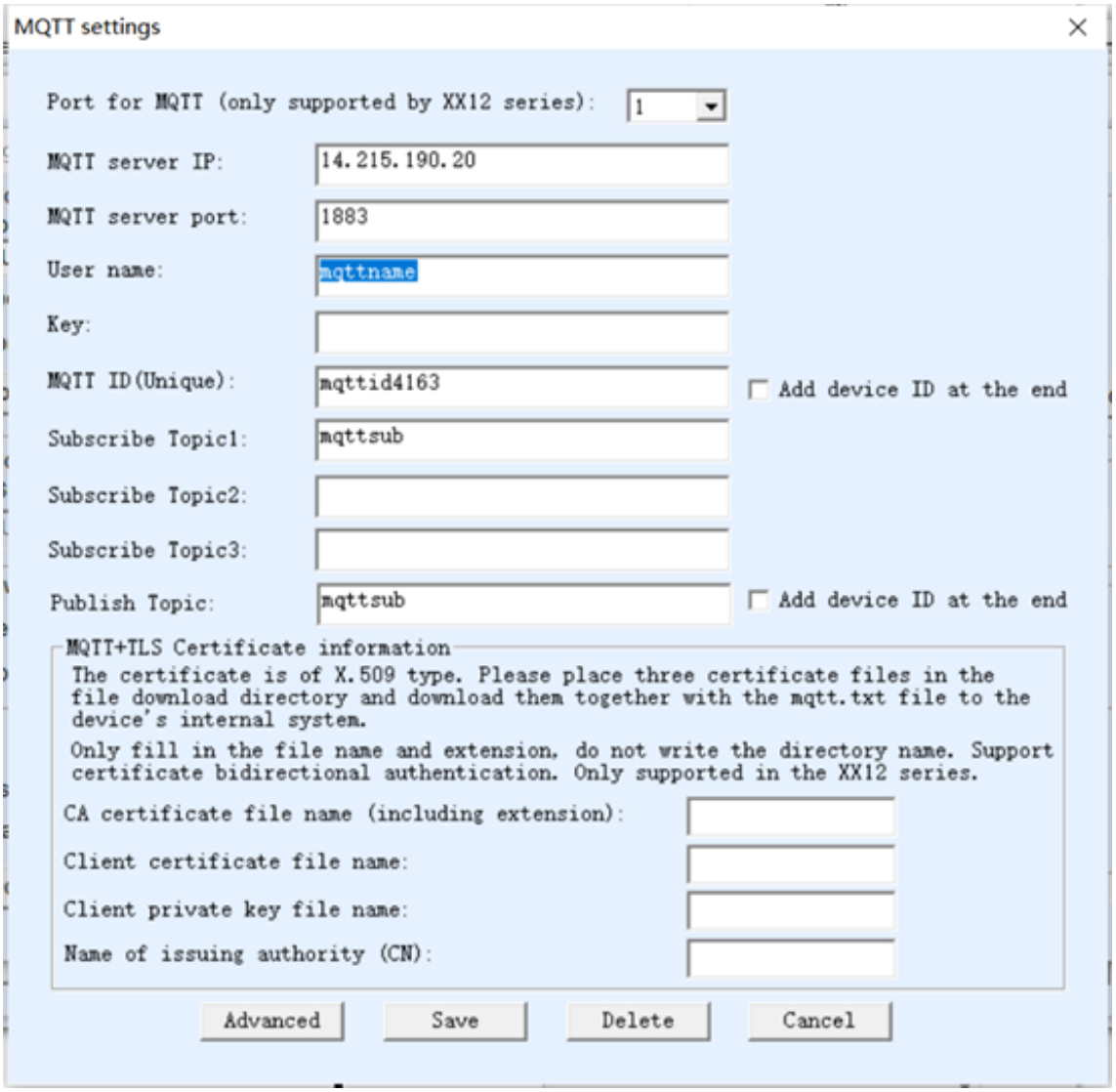


그림 80. MQTT 구성

그런 다음 "MQTT 설정 저장"을 클릭합니다.

9.4 테스트

"웹/프로그램 다운로드 도구 구성" 대화 상자로 돌아가서 "다운로드" 버튼을 클릭하여 이전 JSON 구성, Base64 구성, MQTT 구성을 장치에 다운로드하십시오.

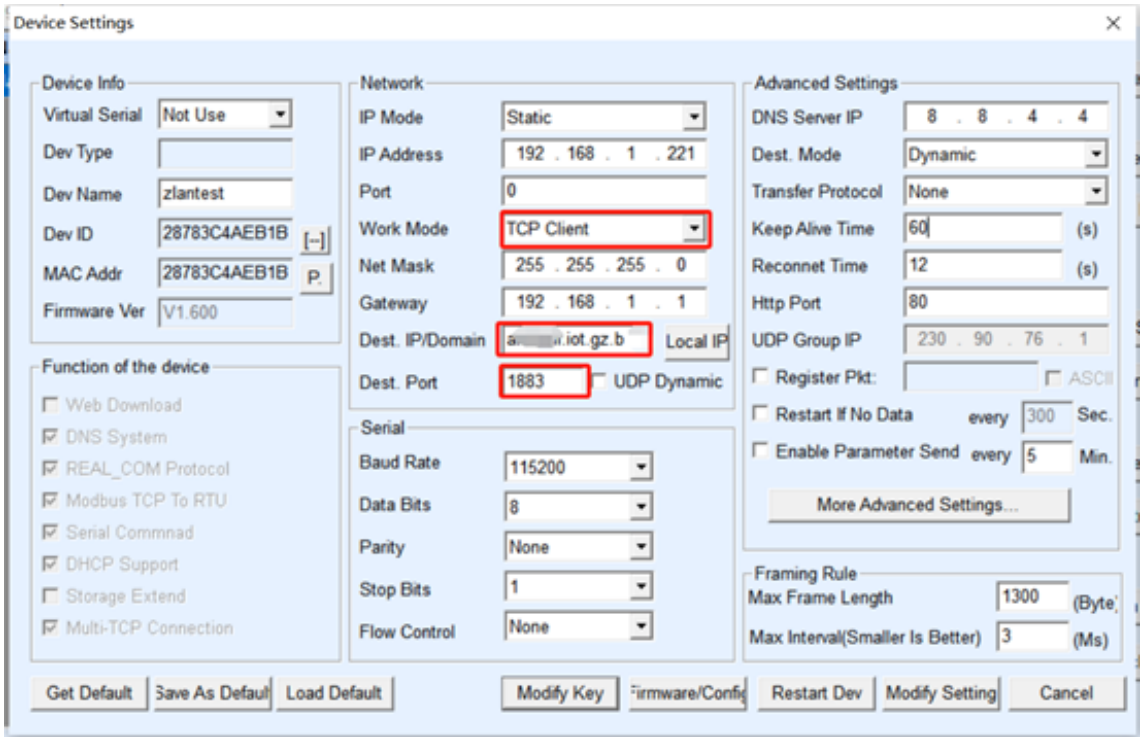


그림 81. 매개변수 확인

장치가 TCP 클라이언트 모드로 작동하는지, 대상 도메인 이름이 MQTT 서버를 가리키는지, 대상 포트가 올바른지 확인하십시오. 잘못된 경우 수정하십시오.

MQTT fx 소프트웨어를 사용하여 이 토픽을 구독하면 base64 인코딩 형식으로 구독 정보를 수신할 수 있습니다.

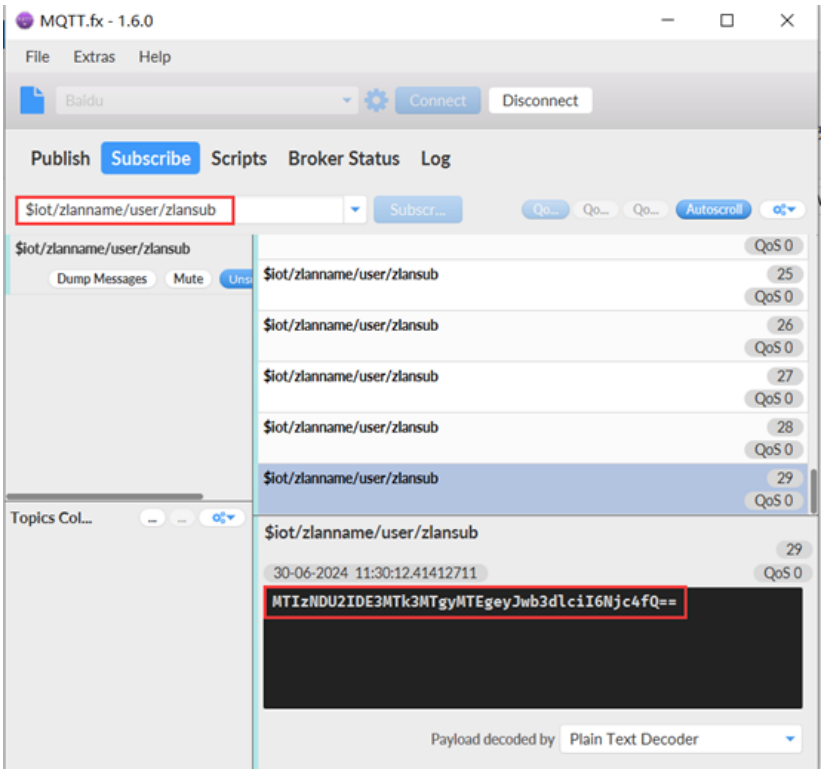


그림 82. 구독 정보

base64 정보를 복사한 후, https://www.iamwawa.cn/base64.html 에서 온라인 도구를 열어 base64를 디코딩하세요.

그림 83. base64 디코딩

필요한 비밀 키 + "" +타임스탬프 +" "+JSON 원시 데이터를 가져옵니다.

10. 다양한 MQTT+JSON 설정 방법

이 기능은 ZLANXX12 시리즈 제품에서만 지원되며 펌웨어 버전 1.445 이상이 필요합니다. NXT_Vircom 6.45 이상 버전에서 설정할 수 있습니다. 설정을 위해 NXT_Vircom에서 ZLANXX12 시리즈의 포트를 선택하십시오.

그림 84. 포트 선택

10.1 MQTT 처리

"장치 편집"을 클릭하고, "펌웨어 및 구성"을 클릭하고, "웹 디렉터리 다운로드 구성"을 클릭한 다음, "MQTT 설정"을 클릭합니다.

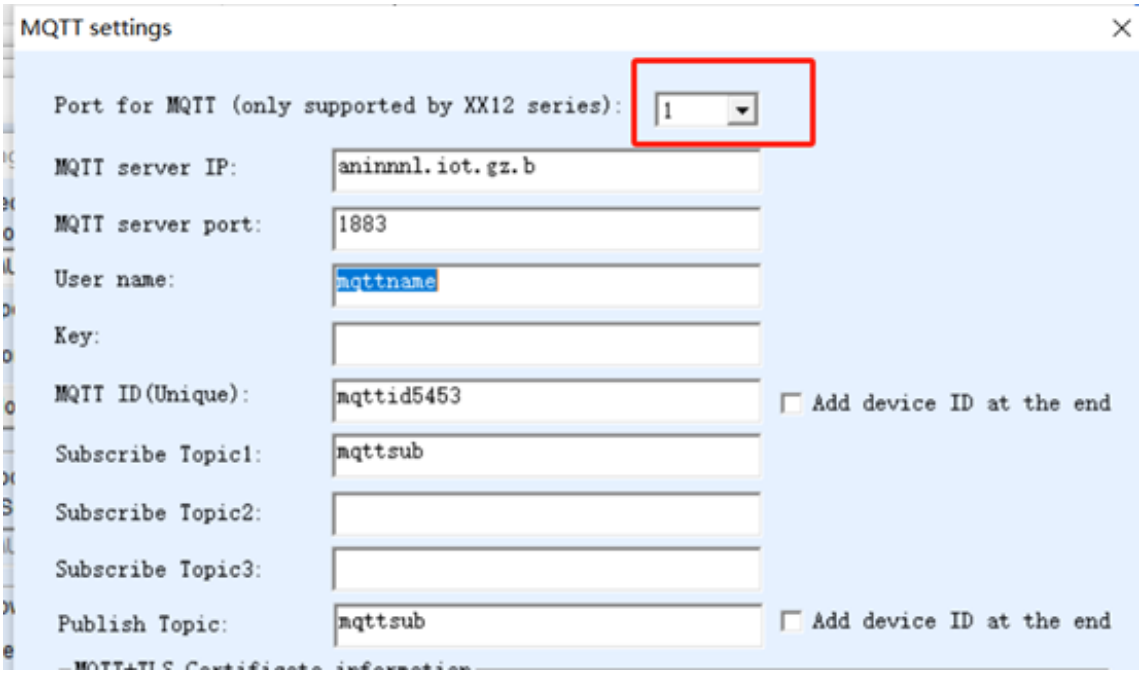


그림 85. MQTT 설정

일반적인 MQTT 설정과 달리, PORT를 선택하여 이 MQTT가 바인딩될 포트를 지정할 수 있습니다. "MQTT 설정 저장"을 클릭하면 디스크에 MQTT.txt, mqtt2.txt~mqtt8.txt(포트에 따라 다름)와 같은 파일이 저장되고, 이러한 파일은 장치 내부 저장소로 다운로드됩니다. MQTT 파일이 여러 개인 경우, MQTT 연결도 여러 개 생성해야 합니다.

각 포트는 서로 다른 시리얼 포트(시리얼 포트 1~8)에 대응하므로, IP 또는 TCP 포트도 서로 다릅니다. 따라서 각 MQTT 파일은 서로 다른 시리얼 포트와 TCP 연결에 연결됩니다.

"MQTT 설정 저장"을 클릭하면 현재 PORT의 MQTT 설정만 저장되고 다른 포트의 설정은 저장되지 않습니다. 다른 포트의 MQTT 설정을 저장하려면 대화 상자에서 PORT를 변경하여 다시 저장해야 합니다.

또한 각 PORT에 대한 클라이언트 ID는 서로 다르게 유지해야 합니다.

여기에는 mqttsub1부터 mqttsub3까지에 해당하는 테마를 가진 세 개의 MQTTS가 설정되어 있습니다.

10.2 JSON 처리 방식

마찬가지로, 구성 페이지/프로그램 다운로드 대화 상자에서 "JSON 구성"을 클릭합니다. 값은 Modbus 03 slave 01 addr 01 in JSON으로 설정됩니다. 전송되는 JSON 키워드는 reg1입니다. 그런 다음 PORT를 1로 선택하고 "JSON 설정 저장"을 클릭합니다.

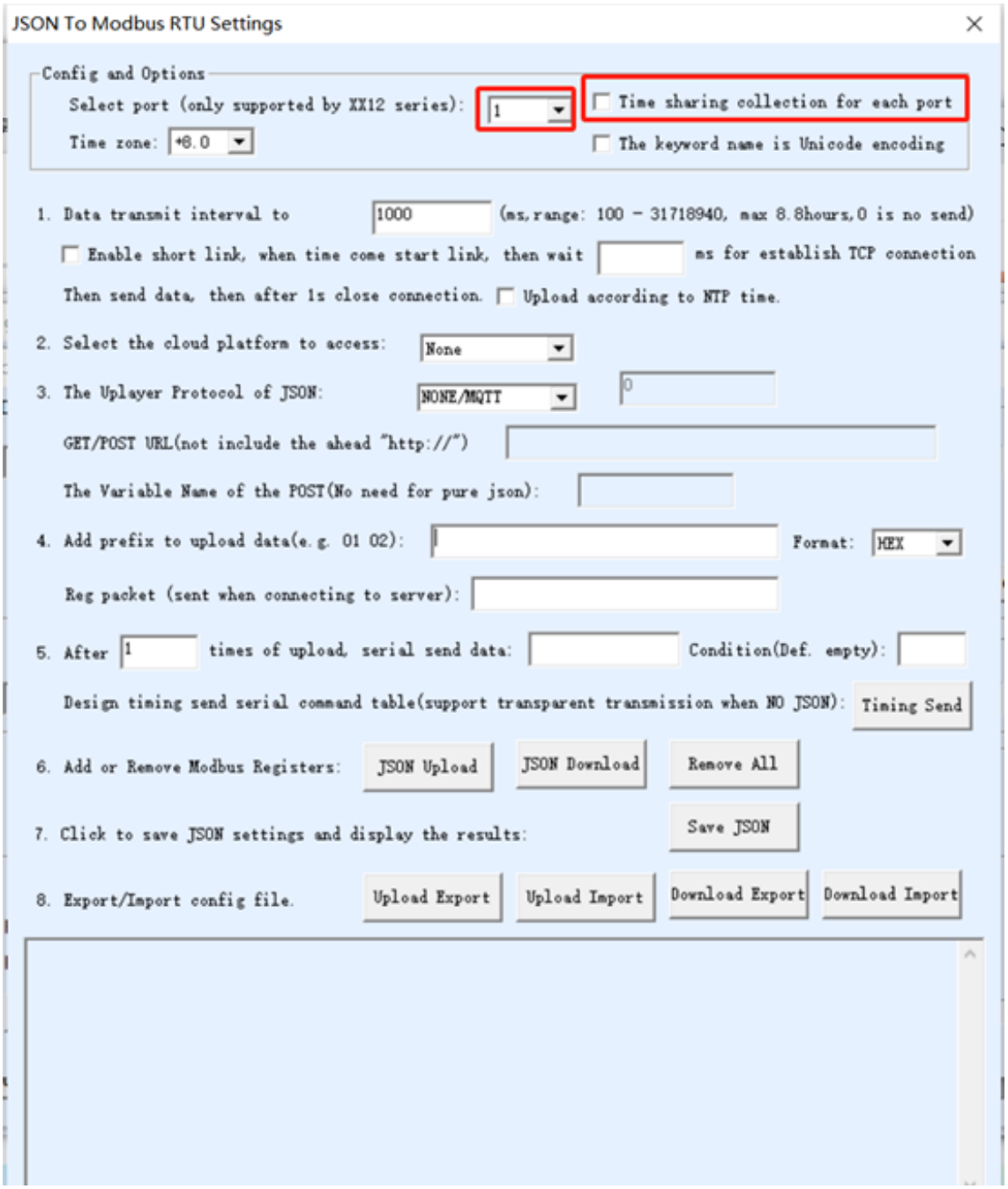


그림 86. 멀티 JSON 설정

MQTT 설정과 유사하게, PORT 번호를 선택하면 JSON 파일이 httpd.txt, httpd2.txt~httpd8.txt와 같이 여러 개로 저장되며, 이는 데이터 수집에 해당하는 시리얼 포트와 TCP 연결을 나타냅니다.

"각 포트별 시간 분할 수집" 옵션은 각 포트의 수집 시간을 조정하는 옵션으로, 자세한 내용은 뒤에서 설명합니다.

PORT를 변경해야 하는 경우, "JSON 설정 저장"을 다시 클릭하십시오. 그러면 디렉터리에 여러 개의 httpd.txt 파일이 생성됩니다.

이제 이러한 mqtt 파일과 httd 파일을 장치에 다운로드하십시오. 그리고 장치를 클라이언트로 구성하여 MQTT 서버에 연결하십시오.

10.3 각 항구별 시간별 수집

위에서 언급한 "포트별 시간 분할 수집" 옵션은 각 포트의 수집 시간을 조정하기 위한 것입니다. 여러 서버가 동일한 Modbus RTU 계측기에서 데이터를 수집해야 하는 경우, 여러 포트의 RS485 직렬 포트를 병합하여 계측기에 연결하고 위에서 설명한 방법으로 구성할 수 있습니다. 그러나 여러 포트에서 동시에 데이터를 전송하면 수집 과정에서 충돌 및 코드 오류가 발생할 수 있습니다.
이러한 경우 JSON을 구성해야 할 때 "각 포트 시간 분할 수집"을 선택하십시오. 이 옵션을 선택하면 httpd.txt 파일이 HTTpc.txt 파일로 변경되고, 여러 httpc 파일이 조정되어 직렬 포트에서 명령이 동시에 출력되지 않고 특정 순서대로 출력되도록 합니다. 이를 통해 하나의 RS485 버스에서 여러 포트를 결합하여 데이터를 수집할 수 있습니다.
시간 분할 수집의 효과를 확인하려면 JSON의 직렬 포트 폴링 시간을 1000ms로 늘리는 것이 좋습니다.

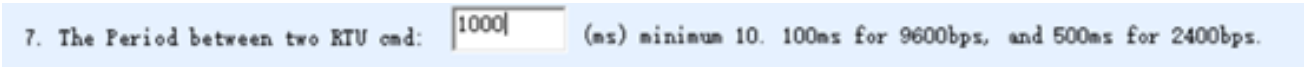


그림 87. 투표 시간 수정

10.4 MQTT 다중 구독 토픽

최대 3개의 토픽을 동시에 구독할 수 있습니다. 현재 zlsn2012 모델이 지원되며, 추후 다른 모델들도 지원될 예정입니다. 자세한 내용은 JSON 테이프 전송 ID 식별을 참조하세요.

10.5 MQTT 토픽에는 ID가 포함되어 있습니다.

지원 메시지에 포함된 MQTT 클라이언트 ID에는 장치 ID가 포함되어 있으며, 게시된 테마에도 장치 ID가 포함되어 있습니다. 따라서 mqtt.txt 파일은 각 장치에서 동일하지만 MQTT 클라이언트 ID와 게시 토픽은 서로 다르게 설정할 수 있습니다. 현재는 zlsn2012 모델이 지원되며, 향후 다른 모델들도 지원될 예정입니다.
자세한 내용은 JSON 테이프 전송 ID 식별을 참조하십시오.

10.6 JSON 테이프 배송 ID 식별

JSON 전송 ID 식별 기능은 모든 장치가 동시에 토픽을 구독하고 전송되는 데이터가 {"ID":"010203040506", "Keyword":123}인 경우, 해당 ID를 가진 장치만 키워드에 해당하는 레지스터를 업데이트하도록 합니다. 메시지가 일치하지 않는 다른 장치는 이 메시지를 무시합니다. 이 기능을 통해 동일한 토픽을 구독하는 여러 장치가 동일한 명령을 모두 실행하는 대신 개별적으로 명령을 실행할 수 있습니다. 이 기능은 zlvircom을 업데이트하면 모든 모델에서 지원됩니다.

MQTT 관련: ZLSN2812 모듈은 이제 시리얼 포트당 최대 3개의 서로 다른 구독 토픽을 지원합니다. 또한 MQTT 클라이언트 설정 단계를 간소화하기 위해 MQTT 클라이언트 ID에 장치 ID를 추가하고 토픽 접미사를 게시하는 기능이 업데이트되었습니다.
JSON: 2812는 이제 장치 ID 접두사 기능을 추가하여 장치 ID를 기반으로 후속 JSON 전송 명령 실행 여부를 결정할 수 있도록 업데이트되었습니다.
펌웨어 버전: 1.447(2012년) 이상.

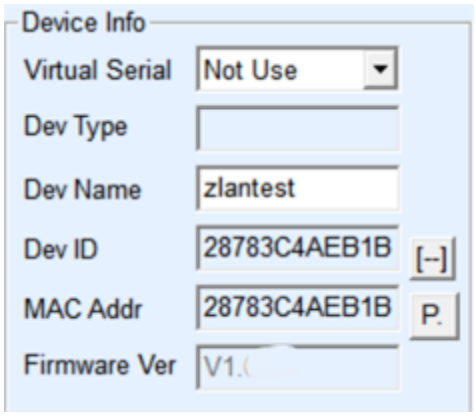


그림 88. 펌웨어 버전

VIRCOM 소프트웨어 버전: 6.46 이상.

그림 89. VIRCOM 버전

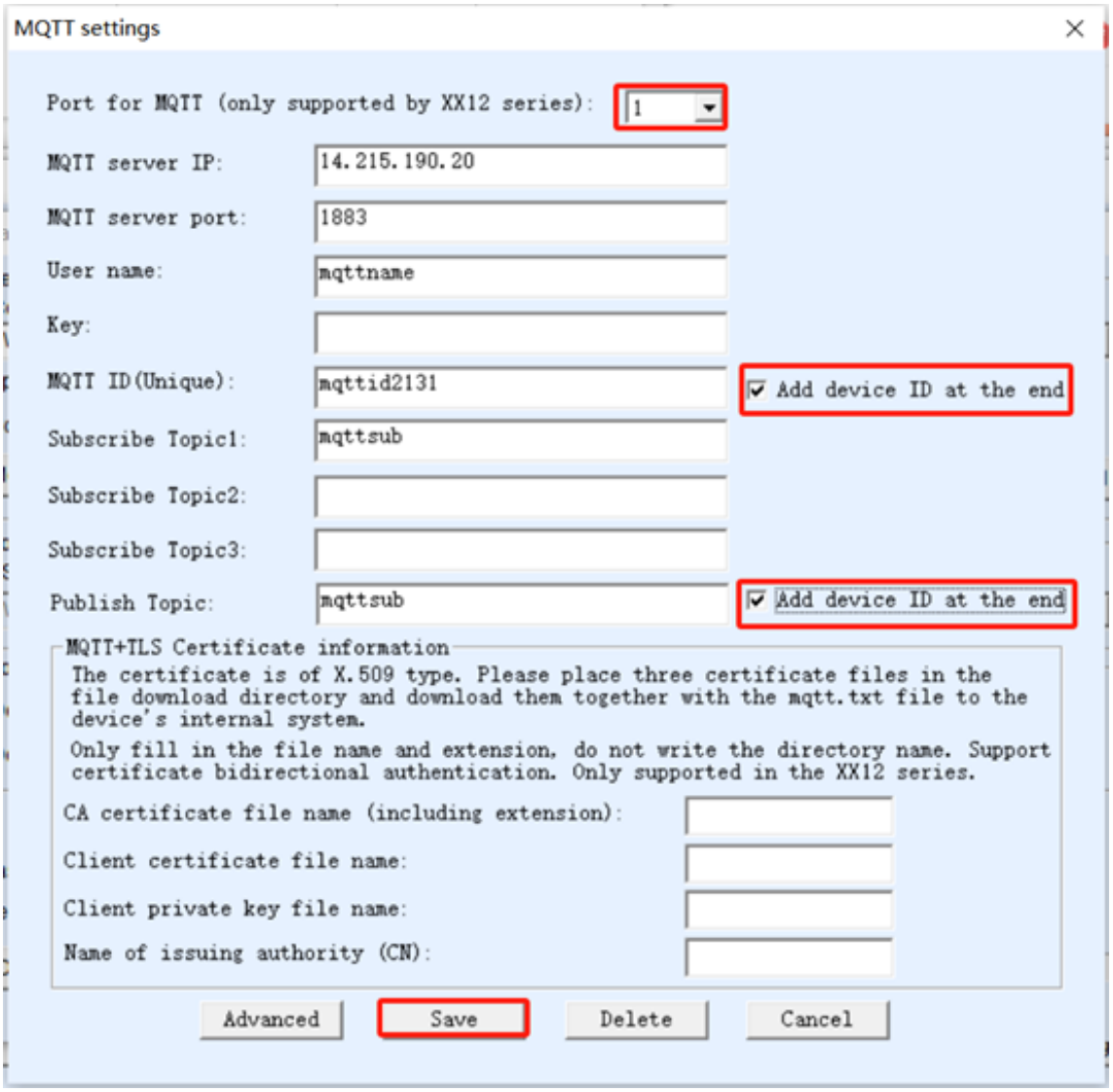


그림 90. MQTT1 설정

오른쪽 상단에서 두 개의 PORT2를 선택하고, 나머지 매개변수는 MQTT1 설정에 따라 조정합니다. MQTT2 토픽 구독은 mqttsub2, mqttsub3, mqttsub4로 각각 설정합니다. 여기서 mqttsub3과 mqttsub4는 PORT1과 PORT2를 동시에 구독하는 것과 같습니다.

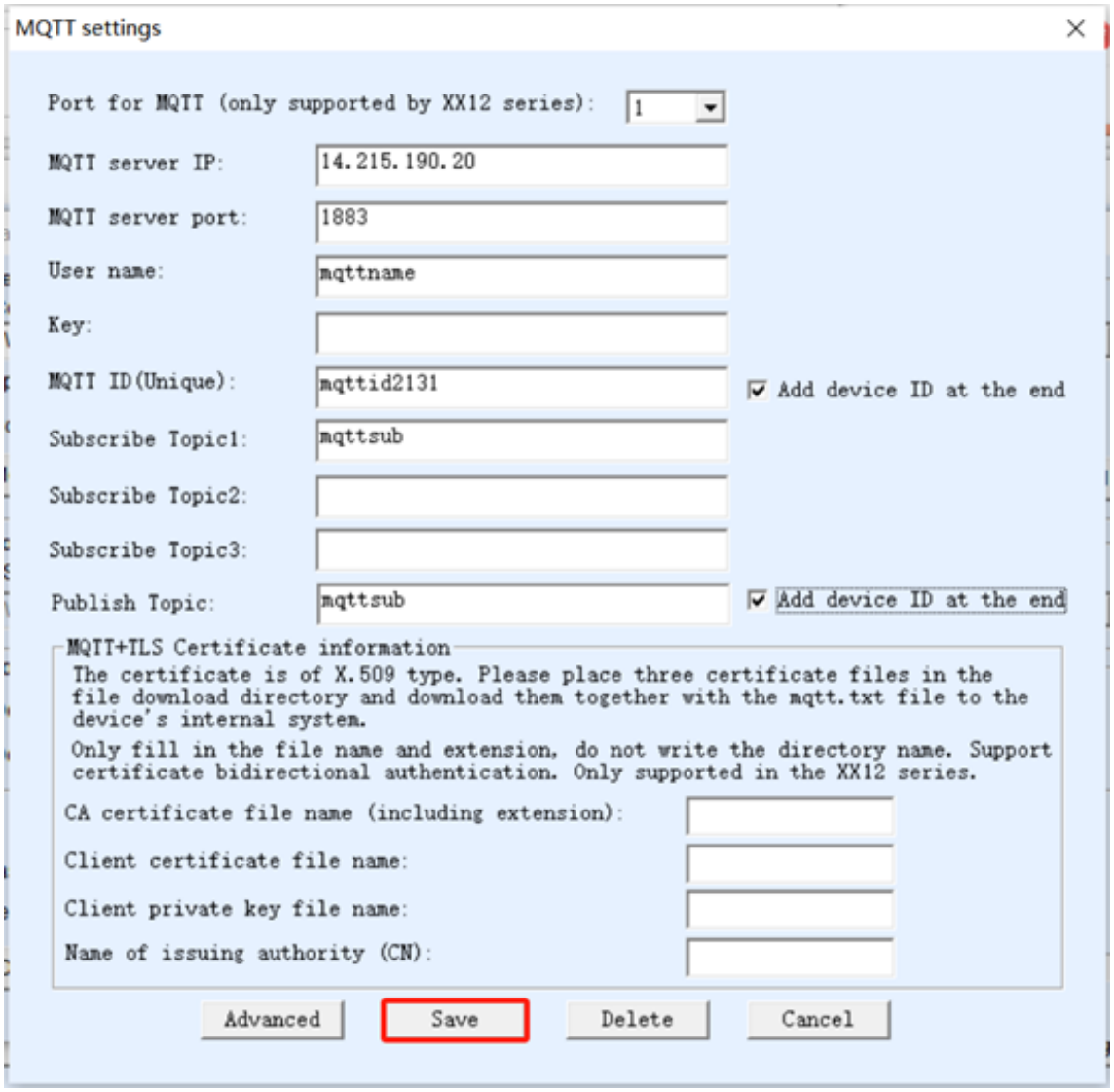


그림 91. MQTT2 설정

JSON 설정

수집 포트 선택 1, JSON 전송을 클릭, 키워드 TEMP1, 스테이션 주소 1, 기능 코드 3, 등록 주소 0을 추가합니다.

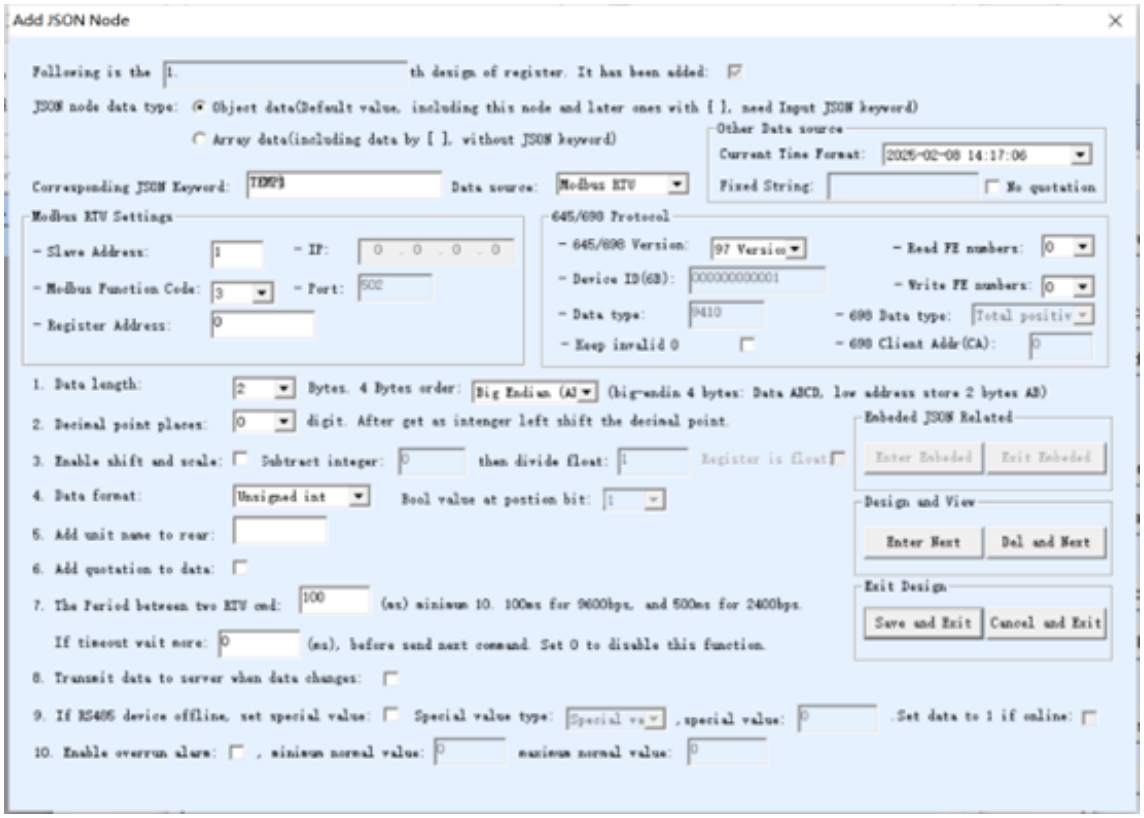


그림 92. JSON1 업로드 설정

JSON1은 ID 일치를 기반으로 명령을 전달할 수 있도록 합니다. 키워드 FS_VALUE:

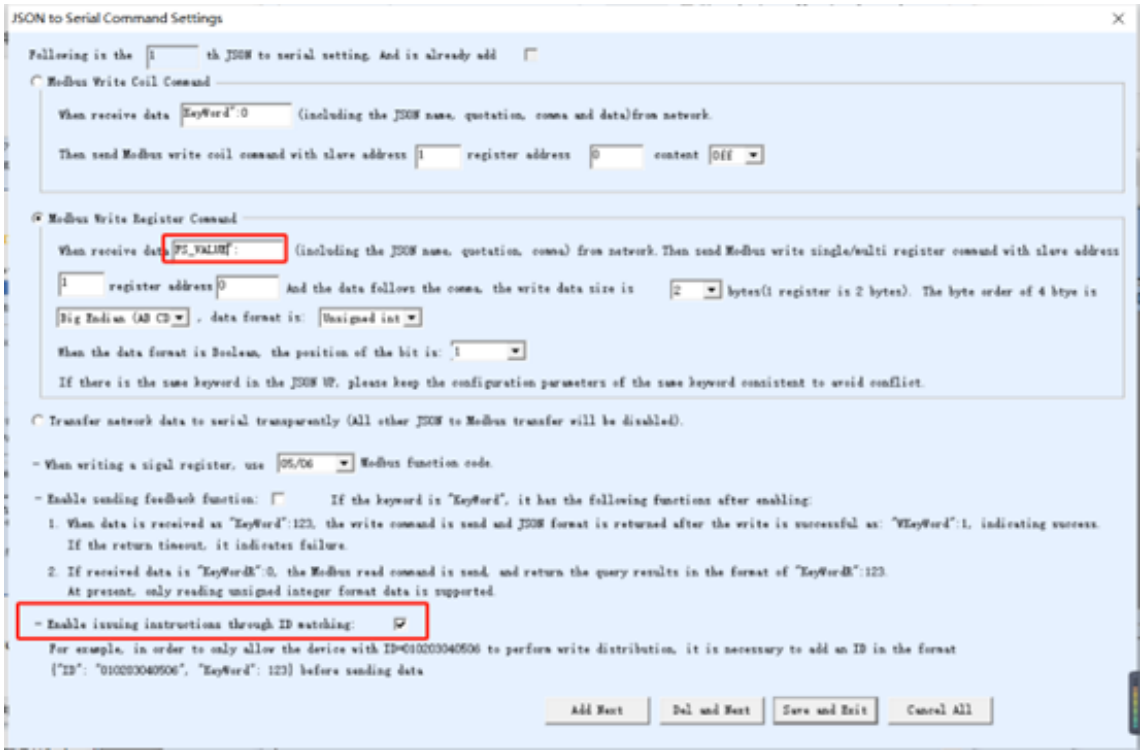


그림 93. JSON1 전달 설정

설정을 완료한 후 "JSON 설정 저장"을 클릭하세요.

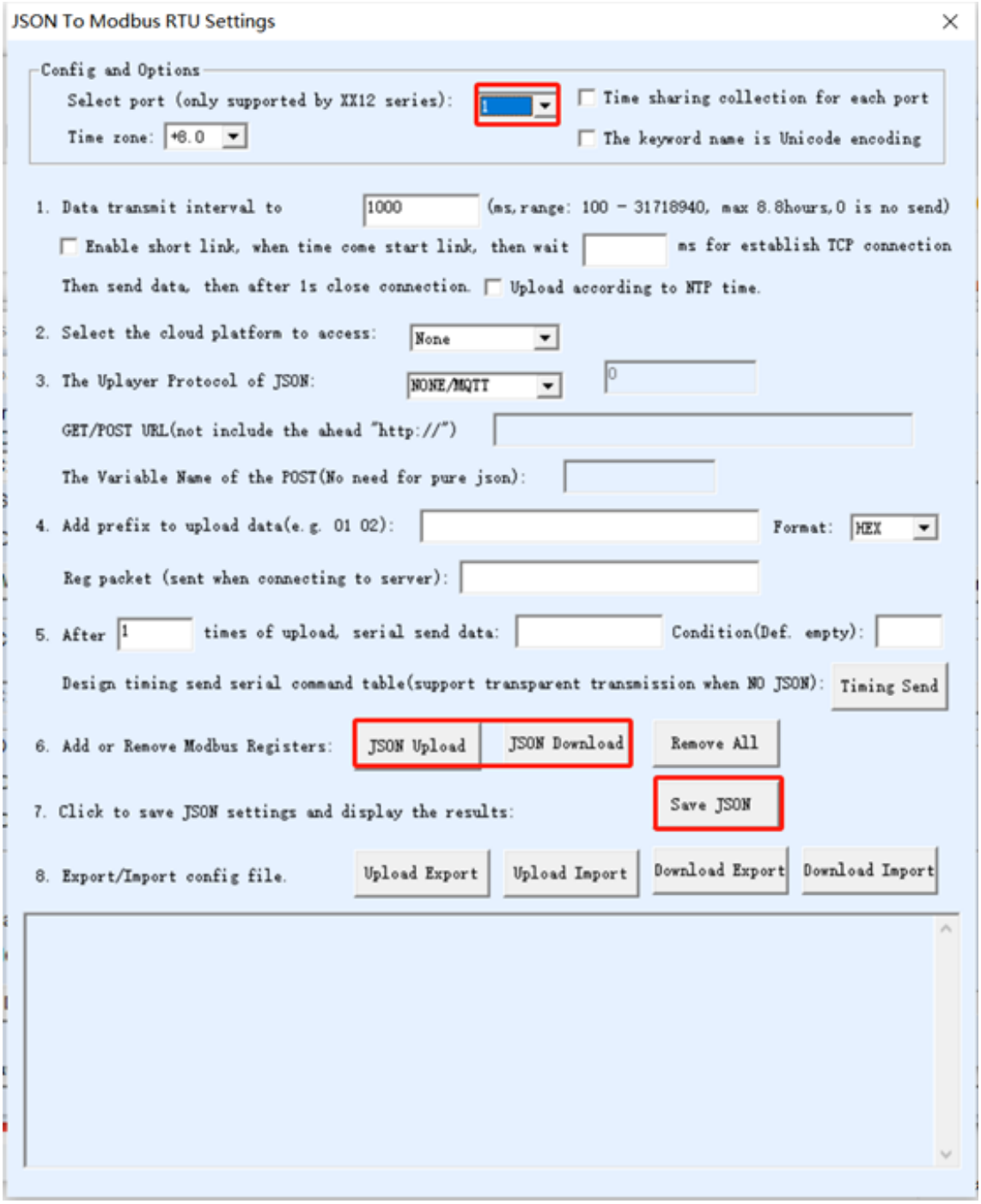


그림 94. JSON1 설정

JSON2 설정에서는 포트 선택 2를 수집한 다음 JSON의 키워드를 TEMP2로 변경해야 합니다. 나머지는 JSON1 설정과 동일합니다.

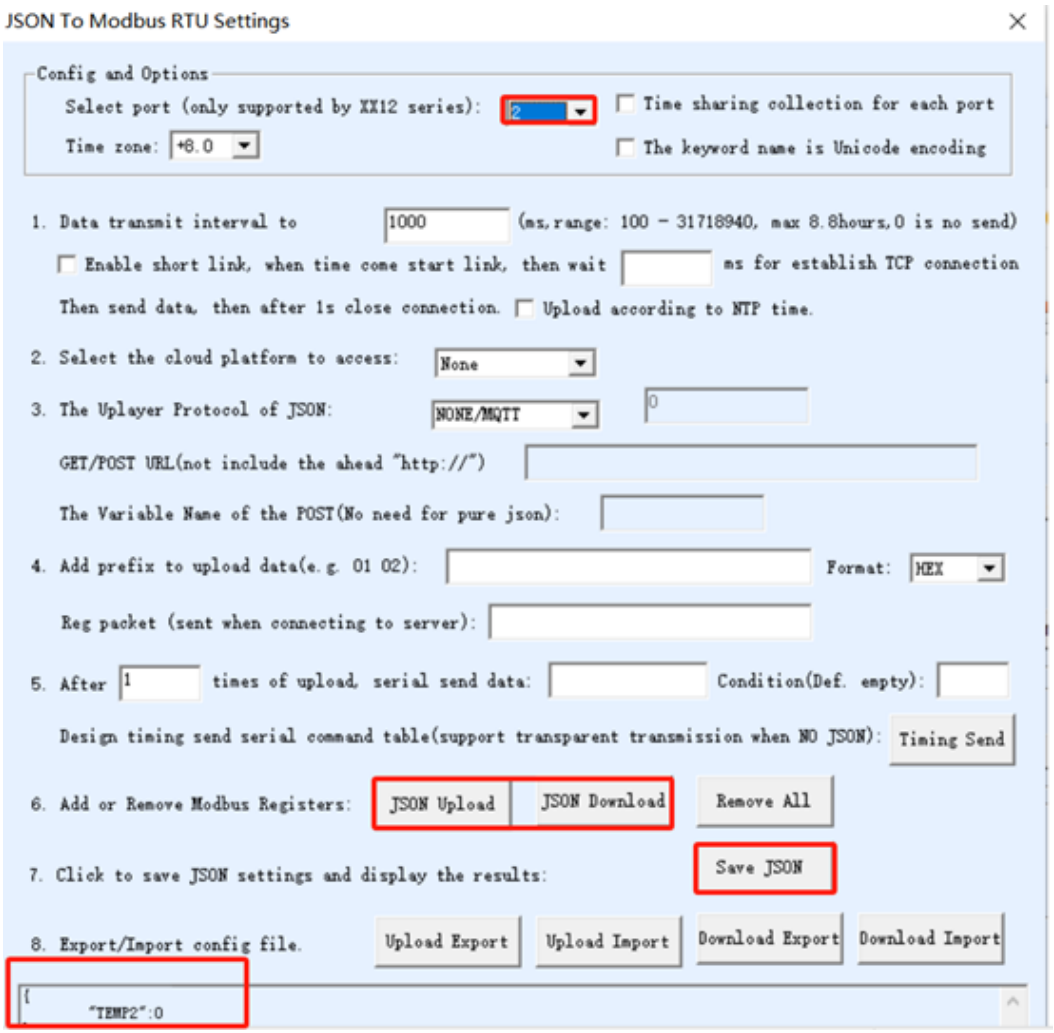


그림 95. JSON2 설정

마지막으로, 수정된 웹 파일을 기기에 다운로드하세요. MQTT와 JSON은 수정 및 저장된 포트에서만 작동하며, 나머지 포트에서는 작동하지 않습니다.

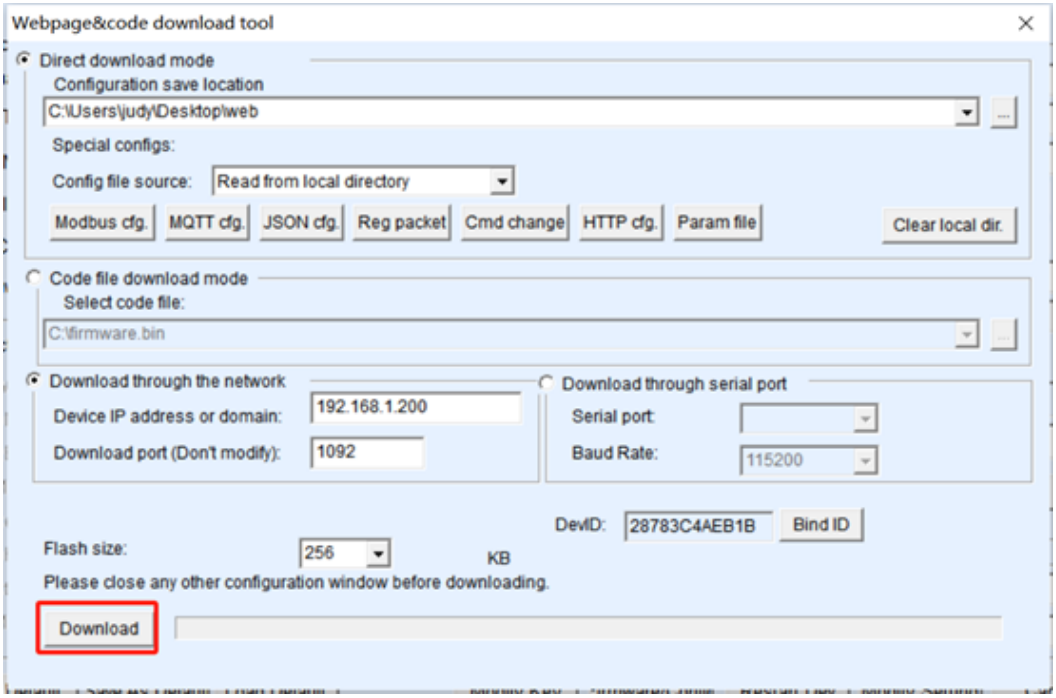


그림 96. 다운로드

■ MQTT/JSON 테스트

MQTT 백그라운드에서 방금 설정한 port1과 port2가 MQTT에 연결되었음을 확인할 수 있으며, 장치 ID는 아래 그림과 같이 MQTT 매개변수에서 사용자가 직접 편집한 mqttid+장치 ID입니다(mqttid28649C1CCA1C, mqttid28649C1CCA1D).

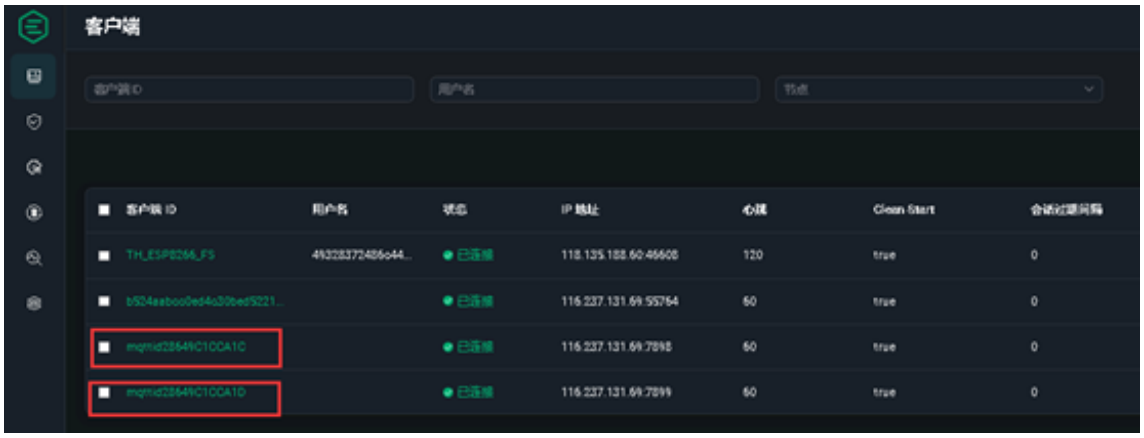


그림 97. 클라이언트 ID

아래 그림에서 볼 수 있듯이 MQTT 서버 백그라운드 클라이언트는 해당 장치가 실제로 세 가지 토픽을 구독하고 있음을 확인할 수 있습니다.

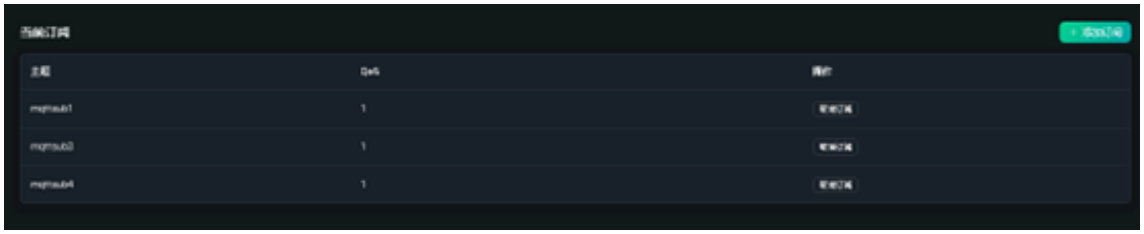


그림 98. 클라이언트 구독 주제

JSON을 전송할 때 MQTT를 통해 MQTT1의 mqttsub28649C1CCA1C 토픽과 MQTT2의 mqttsub-28649C1CCA1D 토픽을 구독하면 장치가 정상적으로 전송하는 것을 확인할 수 있습니다.

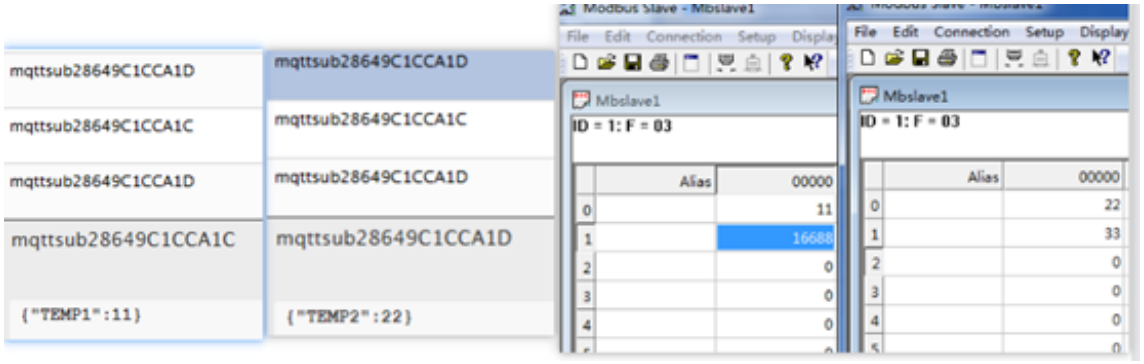
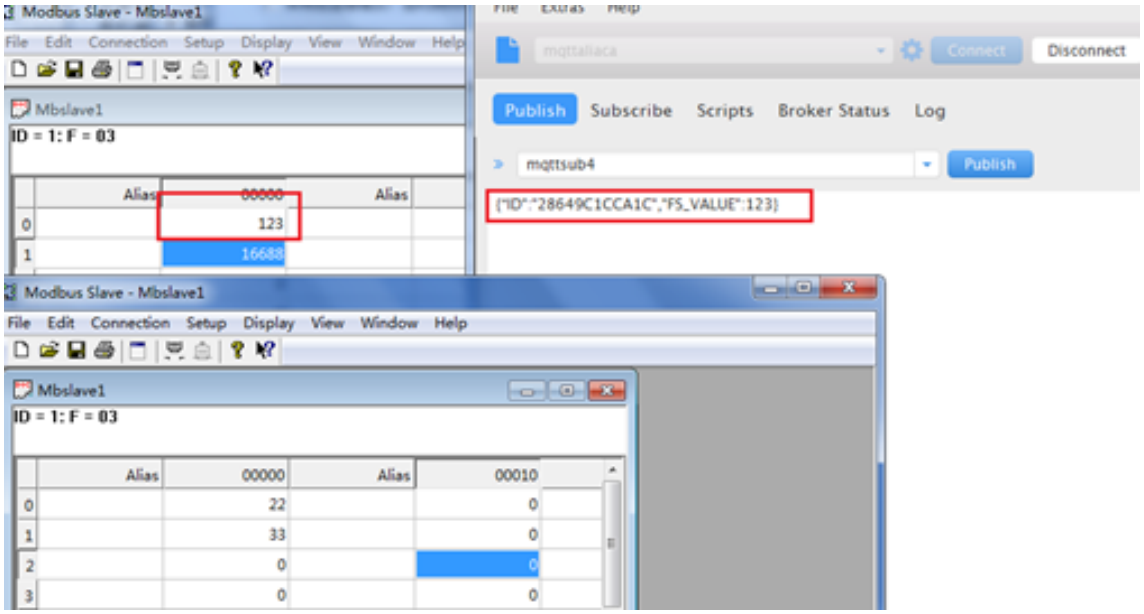


그림 99. 게시할 주제

JSON이 전달될 때, 다음 그림과 같이 테마가 mqttsub4로 설정된 경우를 보여줍니다. 이 경우 MQTT1과 MQTT2 모두에 구독됩니다. {" ID":"28649C1CCA1C","FS_VALUE":123}이 전송되면 해당 ID를 가진 MQTT 장치에서만 MODBUS 데이터가 전송됩니다.



11. 최고 속도

여러 명령을 동시에 전송할 경우, XXX7 모델의 속도는 XXX3 모델보다 약 3배 빠릅니다. XXX4 모델은 XXX3 모델과 동일하지만 포인트 수가 증가했습니다. XX12 모델은 속도가 더 빠르므로 필요에 따라 다른 모델을 선택하십시오.