Json 명령을 이용한 제어

Json이란?

- JSON (JavaScript Object Notation)은 데이터 전송을 위한 경량 형식으로, 주로 다양한 시스템에서 데이터 전송과 저장에 사용
- JavaScript에서 유래되었지만, 이제는 독립적인 데이터 형식으로 다른 프로그래밍 언어에서도 사용된다

ROARM-M2-S에서의 Json 명령

예시 :

```
{"T":1041,"x":235,"y":0,"z":234,"t":3.14}
```

- "T"는 명령 유형을 나타내며, 이는 RoArm-M2-S의 슬레이브 데모의 헤더 파일 "json_cmd.h"에서 정의되어 있다
- 여기서 "1041"은 이 명령이 `CMD_XYZT_DIRECT_CTRL` (로봇 팔을 지정된 위치로 직접 이동시키는 명령이며, 움직임 중에 멈추지 않음)를 의미한다
- X, Y, Z, T는 각각 EoAT(End of Arm Tooling, 로봇 팔 끝단 도구)의 3D 좌표와 "Clamp/Wrist"의 각도를 나타낸다

JSON 명령을 사용하여 통신하는 이유

로봇 팔에 대한 기본 웹 기반 튜토리얼을 주요 튜토리얼 페이지에서 소개했지만, 다양한 JSON 형식의 명령 인터페이스를 설계하여 사용자가 다른 장치나 프로그램을 통해 로봇 팔의 움직임을 쉽게 제어할 수 있도록 했습니다. 사실, 웹 기반 제어의 하위 인터페이스도 통신을 위해 JSON 명령을 사용합니다. JSON 형식의 명령을 사용하여 로봇을 제어하는 장점은 다음과 같습니다:

1. **더 나은 가독성:**
   JSON은 경량의 텍스트 데이터 형식으로, 사람이 읽고 작성하기 쉽습니다. Key-Value 쌍 형식(파이썬의 Dict) 으로 되어 있어 이해하고 디버깅하기 용이하며, 특히 개발 및 테스트 단계에서 유용합니다.
2. **분석 용이:**
   많은 프로그래밍 환경에서 JSON 분석기를 채택하고 있어, JSON을 분석하기 쉽습니다. 이로 인해 명령을 실행 가능한 작업으로 변환하는 과정이 간편해집니다.

3. **크로스 플랫폼 호환성:**
   JSON은 거의 모든 프로그래밍 언어와 플랫폼에서 사용할 수 있는 범용 형식입니다. 즉, 다양한 프로그래밍 언어를 사용하여 JSON 명령을 주고받을 수 있습니다.
4. **구조적 데이터:**
   JSON은 객체와 배열을 포함한 중첩된 데이터 구조를 지원합니다. 이를 통해 명령을 명확하게 구성할 수 있으며, 파라미터, 옵션, 하위 명령 등을 잘 정리할 수 있습니다.
5. **확장성:**
   JSON 명령에 새로운 세그먼트나 파라미터를 쉽게 추가하여 더 많은 기능과 옵션을 지원할 수 있습니다. 이 과정에서 명령의 전체 구조를 변경할 필요가 없습니다.
6. **통합 용이:**
   JSON은 많은 앱과 웹 서비스의 표준 입력 및 출력 형식이므로, 로봇이 다른 시스템 및 서비스와 원활하게 호환됩니다. 예를 들어, REST API를 통한 통신이 가능합니다.
7. **표준:**
   JSON은 널리 사용되는 표준 데이터 형식으로, 다양한 라이브러리와 도구를 통해 JSON 데이터를 처리할 수 있습니다.
8. **다양한 언어 지원:**
   JSON은 다양한 프로그래밍 언어에서 사용될 수 있기 때문에, 다른 언어로 프로그래밍된 로봇 제어 시스템에서도 명령 분석기를 재프로그래밍할 필요가 없습니다.

간단히 말해, JSON 형식의 명령은 로봇을 제어하는 데 있어 간단하고 유연하며, 가독성이 뛰어나고 쉽게 파싱할 수 있는 방법을 제공합니다. 이를 통해 로봇 제어 시스템을 더 견고하고 유지보수가 용이하게 만들 수 있습니다.

통신 모드

- 유선 통신은 RX/TX를 통한 직렬 통신 (UART) 또는 USB-C 타입의 직렬 통신이 가능하다
  - RX (Receive) : 데이터를 받는 핀
  - TX (Transmit) : 데이터를 보내는 핀
- 무선 통신은 HTTP(웹), ESP-NOW를 통해 설정할 수 있다
- 예시에 사용된 무선 통신은 일반적으로 WIFI 모듈을 기반으로 구현된다

🌈UART 통신이란?

UART (Universal Asynchronous Receiver/Transmitter) 통신은 **직렬 통신** 방식의 하나로, 두 장치 간에 데이터를 송수신하는 방법을 정의하는 하드웨어 및 프로토콜입니다. 이 통신 방식은 주로 컴퓨터나 마이크로컨트롤러와 같은 장치 간의 데이터 전송에 사용됩니다.

UART 통신의 구성 요소:

- **TX (Transmit)**: 데이터를 보내는 핀
- **RX (Receive)**: 데이터를 받는 핀
- **보드레이트 (Baud Rate)**: 데이터 전송 속도, 보통 초당 전송할 비트 수를 의미합니다. 예를 들어, 보드레이트가 115200이라면 초당 115,200개의 비트를 전송하는 속도입니다.

- **데이터 비트 (Data Bits)**: 송수신되는 데이터의 크기를 나타냅니다. 보통 8비트 데이터가 사용됩니다.
- **패리티 비트 (Parity Bit)**: 데이터 오류를 검출하기 위한 추가적인 비트입니다. (보통 사용하지 않음)
- **스톱 비트 (Stop Bits)**: 데이터 전송이 끝났음을 나타내는 비트로, 1비트 또는 2비트로 설정됩니다.

UART (Universal Asynchronous Receiver/Transmitter) 통신은 **직렬 통신** 방식의 하나로, 두 장치 간에 데이터를 송수신하는 방법을 정의하는 하드웨어 및 프로토콜입니다. 이 통신 방식은 주로 컴퓨터나 마이크로컨트롤러와 같은 장치 간의 데이터 전송에 사용됩니다.

주요 특징:

1. **비동기 방식 (Asynchronous)**:
   - **비동기**란 데이터가 송수신되는 동안 송신자와 수신자가 동일한 클럭 신호를 공유하지 않는다는 의미입니다. 즉, 별도의 클럭 신호 없이 데이터를 주고받습니다.
   - 데이터가 일정한 속도로 전송되며, 송수신 양쪽 모두 같은 보드레이트(baud rate)를 설정해야 합니다.
2. **직렬 통신 (Serial Communication)**:
   - **직렬** 통신은 데이터를 한 비트씩 순차적으로 전송하는 방식입니다. 반대로, **병렬 통신**은 여러 비트를 동시에 전송합니다.
   - UART는 직렬 통신 방식이므로, 전송할 데이터가 한 비트씩 한 줄로 나열되어 전송됩니다.

UART 통신의 구성 요소:

- **TX (Transmit)**: 데이터를 보내는 핀
- **RX (Receive)**: 데이터를 받는 핀
- **보드레이트 (Baud Rate)**: 데이터 전송 속도, 보통 초당 전송할 비트 수를 의미합니다. 예를 들어, 보드레이트가 115200이라면 초당 115,200개의 비트를 전송하는 속도입니다.
- **데이터 비트 (Data Bits)**: 송수신되는 데이터의 크기를 나타냅니다. 보통 8비트 데이터가 사용됩니다.
- **패리티 비트 (Parity Bit)**: 데이터 오류를 검출하기 위한 추가적인 비트입니다. (보통 사용하지 않음)
- **스톱 비트 (Stop Bits)**: 데이터 전송이 끝났음을 나타내는 비트로, 1비트 또는 2비트로 설정됩니다.

UART 통신의 장점:

1. **간단한 하드웨어**: UART 통신은 두 개의 핀 (TX, RX)만 있으면 되기 때문에 하드웨어가 간단합니다.
2. **낮은 비용**: 대부분의 마이크로컨트롤러, 컴퓨터, 임베디드 시스템에 기본적으로 UART 포트를 내장하고 있습니다.
3. **짧은 거리 통신에 유용**: UART는 비교적 짧은 거리에서 안정적인 통신을 제공합니다.

UART 통신의 사용 예:

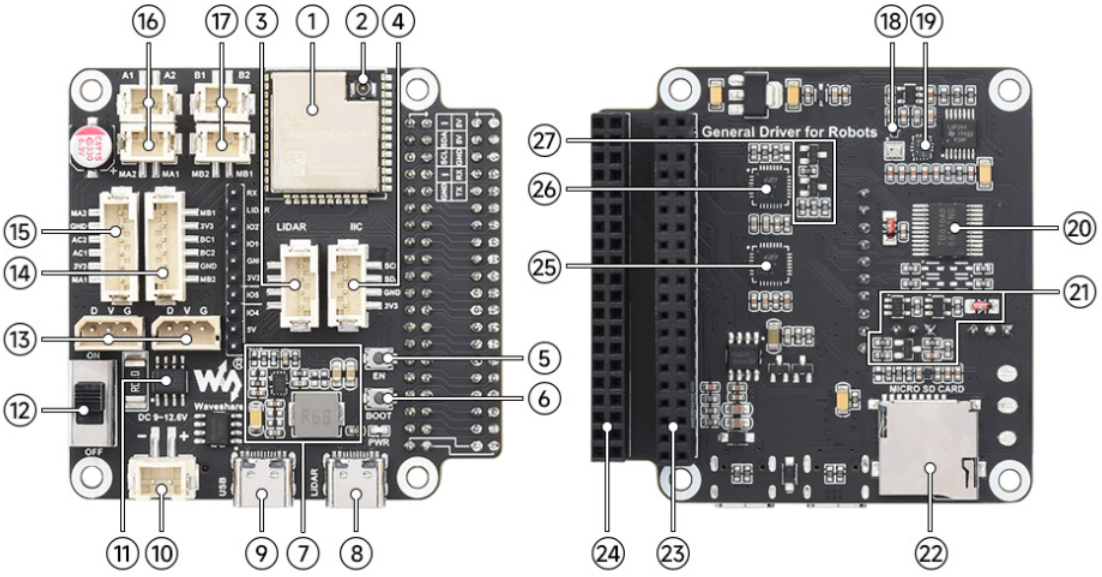
- **컴퓨터와 마이크로컨트롤러 간의 데이터 전송**: 예를 들어, 아두이노와 같은 마이크로컨트롤러와 컴퓨터 간의 시리얼 통신.
- **임베디드 시스템 간의 데이터 전송**: 다양한 센서나 모듈 간의 통신.
- **디버깅**: 개발 중에 UART를 통해 실시간으로 데이터를 확인할 수 있어 디버깅 용도로 자주 사용됩니다.

UART와 다른 통신 방식의 차이점:

- **I2C, SPI**와 같은 다른 직렬 통신 방식에 비해 UART는 **두 장치 간의 1:1 통신**을 지원하며, **복잡하지 않지만 속도나 다중 장치 연결에 있어 제약이 있을 수 있습니다**.

따라서, UART 통신은 단순한 데이터 전송에 적합하며, 간단한 연결과 빠른 설정이 필요한 경우에 많이 사용됩니다.

ESP32기반 드라이버 보드



번호	리소스 이름	설명
①	ESP32-WROOM-32 컨트롤러 모듈	Arduino IDE를 사용하여 개발 가능
②	IPEX1 와이파이 커넥터	무선 통신 거리 확장 WIFI 안테나 연결
③	레이더 인터페이스	통합 레이더 어댑터 기능
④	2C 주변 확장 인터페이스	OLED 스크린 또는 다른 I2C 센서로 연결
⑤	재설정 버튼	ESP32을 재부팅
⑥	다운로드 버튼	전원을 켠 후 ESP32 다운로드 모드 실행
⑦	5V 전압 조정기 회로 DC-DC	라즈베리 파이 또는 제트 슨 나노와

번호	리소스 이름	설명
⑮	모터 인터페이스 PH2.0 6P (그룹 A)	인코더가 있는 모터에 대한 그룹 A 인터페이스
⑯	모터 인터페이스 PH2.0 2P (그룹 A)	인코더가 없는 모터의 그룹 A 인터페이스
⑰	모터 인터페이스 PH2.0 2P (그룹 B)	인코더가 없는 모터의 그룹 B 인터페이스
⑱	AK09918C	3축 전자 나침반
⑲	QMI8658C	6축 모션 센서
⑳	TB6612FNG	모터 제어 칩
㉑	직렬 버스 서보 제어 회로	여러 ST3215 직렬 버스 서보를 제어하고 서보 피드백을 얻기 위한 회로

		같은 호스트 컴퓨터 용 전원 공급 장치
⑧	Type-C 포트 (LADAR)	레이더 데이터 전송
⑨	Type-C 포트 (USB)	ESP32 프로그램 업로드 통신 인터페이스
⑩	XH2.54 전원 포트	DC 7 ~ 13V 입력 지원, 직렬 버스 서보 및 모터에 직접 전원 공급 가능
⑪	INA219	전압/전류 모니터링 칩
⑫	전원 켜기/끄기	외부 전원 공급 장치 ON/OFF
⑬	ST3215 직렬 버스 서보 인터페이스	ST3215 직렬 버스 서보 연결
⑭	모터 인터페이스 PH2.0 6P	인코더가 있는 모터의 그룹 B 인터페이스
㉒	TF 카드 슬롯	로그 또는 WIFI 구성을 저장하는 데 사용할 수 있음
㉓	40PIN GPIO 헤더	라즈베리 파이 또는 다른 호스트 보드와 연결
㉔	40PIN 확장 헤더	라즈베리 파이 또는 다른 호스트 보드의 GPIO 핀을 사용
㉕	CP-2102	UART to USB, 레이더 데이터 전송 용
㉖	CP-2102	ESP32 통신용 UART to USB
㉗	자동 다운로드 회로	EN 및 부팅 버튼을 누르지 않고 ESP32에 프로그램 업로드 가능

ROARM-M2-S의 데모 코드

UART 통신을 위한 데모

```
import serial  # 시리얼 통신을 위한 pySerial 라이브러리 가져오기
import argparse  # 명령행 인수 파싱을 위한 argparse 라이브러리 가져오기
import threading  # 멀티스레딩을 위한 threading 라이브러리 가져오기

# 시리얼 포트에서 데이터를 읽어오는 함수
def read_serial():
    while True:
        data = ser.readline().decode('utf-8')  # 시리얼 포트에서 한 줄을 읽고, UTF
        if data:  # 읽은 데이터가 있으면
            print(f"Received: {data}", end='')  # 받은 데이터를 출력 (끝에 줄바꿈 

# 메인 함수
def main():
    global ser  # 전역 변수로 ser 선언 (시리얼 포트 객체)
    parser = argparse.ArgumentParser(description='Serial JSON Communication')
    parser.add_argument('port', type=str, help='Serial port name (e.g., COM1
    # port라는 인수를 필수로 받고 문자열로 받겠다는 뜻
    # help 옵션을 사용했을때, 포트 인수가 무엇을 뜻하는지 설명하는 문구를 출력함
```

```
    args = parser.parse_args()  # 명령행 인수를 파싱하여 args 변수에 저장, 사용자가 프

    # 시리얼 포트 열기 (포트 이름, 보드레이트, DSR/DTR 관련 설정)
    ser = serial.Serial(args.port, baudrate=115200, dsrdtr=None)
    ser.setRTS(False)  # RTS(Ready To Send) 신호를 비활성화
    ser.setDTR(False)  # DTR(Data Terminal Ready) 신호를 비활성화

    # 시리얼 포트에서 데이터를 읽는 스레드를 생성하여 시작
    serial_recv_thread = threading.Thread(target=read_serial)
    serial_recv_thread.daemon = True  # 데몬 스레드로 설정 (메인 프로그램 종료 시 스레
    serial_recv_thread.start()  # 스레드 시작

    try:
        while True:
            command = input("")  # 사용자로부터 명령 입력 받기
            ser.write(command.encode() + b'\n')  # 입력 받은 명령을 시리얼 포트로 전
    except KeyboardInterrupt:  # 키보드 인터럽트 (Ctrl+C) 발생 시
        pass  # 예외 처리 없이 종료
    finally:
        ser.close()  # 시리얼 포트 닫기 (프로그램 종료 시)

# 프로그램 실행을 위한 진입점
if __name__ == "__main__":
    main()  # main 함수 실행
```

HTTP 통신을 통한 데모

```
import requests  # HTTP 요청을 보내기 위한 requests 라이브러리 가져오기
import argparse  # 명령행 인수를 파싱하기 위한 argparse 라이브러리 가져오기


def main():
    # 명령행 인수 파서를 생성하고 프로그램 설명 추가
    parser = argparse.ArgumentParser(description='Http JSON Communication')

    # 'ip'라는 이름의 인수를 추가, IP 주소를 입력 받기 위한 파라미터 정의
    parser.add_argument('ip', type=str, help='IP address: 192.168.10.104')

    # 명령행 인수를 파싱하여 'args' 객체에 저장
    args = parser.parse_args()

    # 입력 받은 IP 주소를 변수에 저장
    ip_addr = args.ip

    try:
        # 무한 루프를 시작하여 사용자가 명령을 입력할 때마다 반복
```

```
        while True:
            # 사용자로부터 JSON 명령을 입력 받음
            command = input("input your json cmd: ")

            # 입력받은 JSON 명령을 URL로 변환 (http://ip/js?json=명령)
            url = "http://" + ip_addr + "/js?json=" + command

            # 해당 URL로 GET 요청을 보내고 응답을 받음
            response = requests.get(url)

            # 응답의 본문 내용 (텍스트)을 변수에 저장
            content = response.text

            # 응답 내용을 출력
            print(content)
    except KeyboardInterrupt:
        # 프로그램 실행 중 사용자가 Ctrl+C를 눌렀을 때, 예외 처리
        pass


# 프로그램의 진입점 (메인 함수 실행)
if __name__ == "__main__":
    main()  # main 함수 실행
```

- 위의 데모코드를 이용하여 로봇암에 JSON 명령을 보내면 피드백 정보를 받을 수 있다
- https://www.waveshare.com/wiki/RoArm-M2-S_Python_UART_Communication 에서 데모 다운로드와 사용법을 볼 수 있다

🤩 차후에 라즈베리 파이를 이용해 ROS2 시스템을 구축하여 메인 서버로 만들고 ESP-NOW나 UART 통신을 이용하여
라즈베리파이와 직접연결하여 동작을 제어할 수 있을 것이라 생각된다