

Micro-Ros 사용 메뉴얼

목차

0. 들어가기 전 TEENSY4.1 세팅.....	2
1. MICRO-ROS 란?.....	3
2. ROS의 기본 용어.....	4
3. MICRO-ROS 설치.....	5
4. MICRO-ROS 작업 환경 세팅 : PLATFORMIO + VSCODE.....	9
5. 하드웨어 구성 : ARDUINO IDE 테스트	13
6. MICRO-ROS 실행하기	15

0. 들어가기 전 Teensy4.1 세팅

Teensy 4.1을 사용하기 위해서는 환경 설정이 필요함.

* Teensy 4.1 Loader 설치

https://www.pjrc.com/teensy/loader_win10.html

* Arduino IDE 에 Boards Manager teensy 설치

https://www.pjrc.com/teensy/td_download.html

*테스트 코드는 아래 깃허브를 참고할 것

Github : https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/tree/main

1. Micro-Ros 란?

1) ROS란 무엇인가?

ROS(Robot Operating System, 로봇 운영체제)는 로봇 소프트웨어 개발을 위한 **프레임워크**임. 로봇에는 센서, 모터, 카메라 등 다양한 하드웨어가 존재하며, 이를 제어하기 위해 데이터 통신, 제어 알고리즘, 장치 간 메시지 교환 등 복잡한 작업이 필요함.

ROS는 이러한 기능을 표준화된 형태로 제공하여, 개발자가 하드웨어 제어나 통신을 직접 구현하지 않고도 로봇 애플리케이션을 효율 적으로 개발할 수 있도록 도와줌.

즉, **로봇 소프트웨어 개발을 위한 운영체제와 같은 역할을 수행**한다고 이해할 수 있음.

2) ROS의 한계

ROS는 본래 성능이 좋은 PC나 서버 환경에서 사용하기 적합하게 설계됨. 그러나 실제 로봇 시스템에는 **성능이 제한된 임베디드 보드**가 많이 사용됨. 이러한 장치들은 메모리, CPU, 전력 등이 제한적이기 때문에 ROS를 그대로 적용하기 어렵다는 한계가 있음.

3) Micro-ROS의 등장

이러한 한계를 해결하기 위해 Micro-ROS가 개발됨.

Micro-ROS는 **ROS 2를 기반으로 경량화된 버전**으로, 리소스가 제한된 임베디드 디바이스에서도 ROS 2의 기능을 사용할 수 있도록 설계됨.

이를 통해 작은 센서 보드나 모터 제어 보드도 ROS 네트워크에 포함되어,

임베디드 장치 <-> 메인 PC <-> 클라우드

까지 하나의 통합된 ROS 환경을 구축할 수 있음.

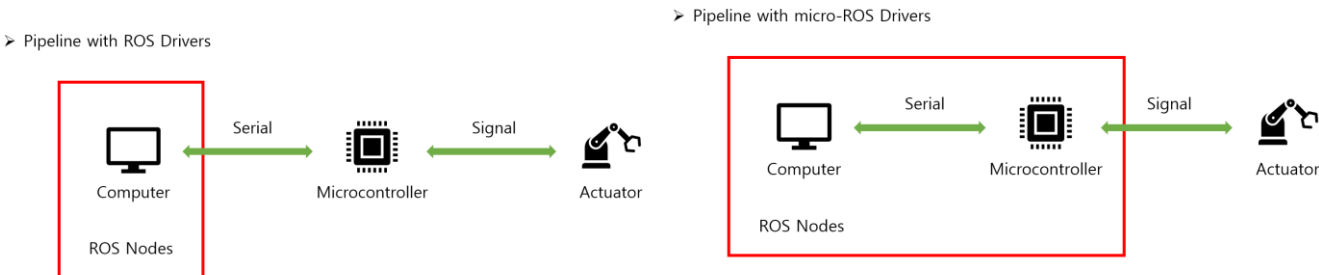


그림 1 micro-ROS pipeline

그림 1 ROS pipeline (micro-ros x)

기존에는 PC에서만 ROS 통신이 가능했지만 micro-ros 를 통해 MCU 에서도 ROS 를 사용하게 하므로써 하나의 노드로 데이터를 주고 받을 수 있어 따로 통신 설정을 안해도 됨.

2. ROS의 기본 용어

1) Node (노드)

- 의미 : ROS 에서 실행되는 하나의 프로그램 단위
- 비유 : “사람의 기관” 같은 것, 눈(카메라 노드), 귀(센서 노드), 다리(모터 제어 노드)
- 로봇은 여러 개의 노드가 **동시에 실행(Launch)**되면서 서로 협력해 움직임

2) Topic (토픽)

- 의미 : 노드 간에 메시지를 주고받는 통로(채널)
- 비유 : “단톡방”같은 개념
- 예시 : /camera/image 라는 토픽 -> 카메라 노드가 사진 데이터를 이 방에 올림

3) Publisher (퍼블리셔)

- 의미 : 특정 토픽에 메시지를 보내는 노드
- 비유 : 단톡방에 “글 올리는 사람”
- 예시 : 카메라 노드가 /camera/image 토픽에 사진 데이터 퍼블리시

4) Subscriber (서브스크라이버)

- 의미 : 특정 토픽의 메시지를 받는 노드
- 비유 : 단톡방에서 “글 읽는 사람”
- 예시 : AI 노드가 /camera/image 토픽을 subscribe 해서 이미지 받아서 인식처리

* 더 많은 용어가 있지만 본 매뉴얼에서는 이정도만 알아도 됨.

3. Micro-ROS 설치

사용 환경 : Ubuntu 24.04 / ROS2 Jazzy

사용된 MCU : Teensy4.1

Manual : https://github.com/hippo5329/micro_ros_arduino_examples_platformio/wiki

1) 필수 패키지 설치

```
sudo apt update  
sudo apt upgrade -y  
sudo apt remove -y brltty  
sudo apt install -y python3-venv build-essential cmake git curl software-properties-common
```

2) PlatformIO & udev 규칙 설치

```
curl -fsSL -o get-platformio.py https://raw.githubusercontent.com/platformio/platformio-core-installer/master/get-platformio.py  
python3 get-platformio.py  
echo "PATH=W"WP$PATH:$HOME/.platformio/penv/binW"" >> $HOME/.bashrc  
source ~/.bashrc  
  
curl -fsSL https://raw.githubusercontent.com/platformio/platformio-core/develop/platformio/assets/system/99-platformio-udev.rules | sudo tee /etc/udev/rules.d/99-platformio-udev.rules  
sudo service udev restart  
sudo usermod -a -G dialout $USER  
sudo usermod -a -G plugdev $USER
```

* \$USER는 Ubuntu의 **사용자 이름으로 대체**하여 쓸 것.

예시) 터미널에 machine@gnu 의 형식이 보일 것임.

machine -> 사용자 이름(USER), gnu -> 컴퓨터 이름(Hostname)

3) ROS2 Jazzy 설치 (설치 되어 있다면 건너뛰기)

```
export ROS_DISTRO=jazzy

sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo
$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

sudo apt update

sudo apt install -y ros-$ROS_DISTRO-desktop ros-dev-tools python3-colcon-common-extensions
python3-pip

sudo rosdep init

rosdep update

echo "source /opt/ros/$ROS_DISTRO/setup.bash" >> ~/.bashrc

source ~/.bashrc
```

4) Micro-ROS Agent 빌드

```
mkdir -p ~/uros_ws/src

cd ~/uros_ws/src

git clone -b $ROS_DISTRO https://github.com/micro-ROS/micro-ROS-Agent.git
git clone -b $ROS_DISTRO https://github.com/micro-ROS/micro-ros_msgs.git

cd ..

rosdep install --from-paths src --ignore-src -r -y

colcon build --symlink-install

cd ~

echo "source $HOME/uros_ws/install/setup.bash" >> ~/.bashrc

source ~/.bashrc
```

*Micro-ROS Agent 설치

- Micro-ROS는 임베디드 보드 <-> PC(ROS2) 간 통신을 위해 **Agent가 필요**
- Agent는 ROS 2 네트워크와 임베디드 보드를 연결하는 **브리지 역할**

5) 예제 파일 다운 받기

```
git clone https://github.com/hippo5329/micro_ros_arduino_examples_platformio.git
```

6) 예제 코드 실행

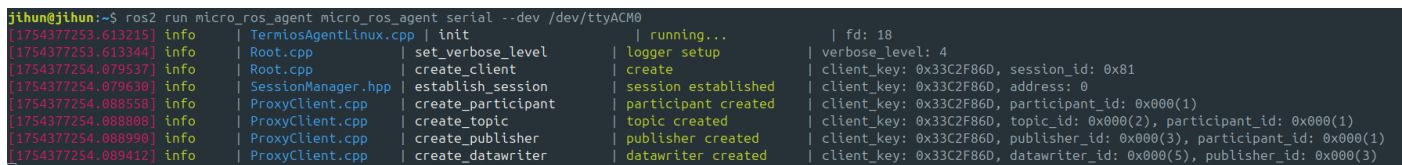
```
cd micro_ros_arduino_examples_platformio/micro-ros_reconnection_example  
pio run -e teensy41 -t upload  
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0
```

이 예제는 teensy4.1 에 micro-ros 를 통해 1 초마다 데이터를 publish 하고 LED 를 켜고 끄는 예제임.

우리는 1 초마다 데이터가 publish 되는것만 ros 를 통해 확인해볼 것임

* 예제 코드 실행 과정은 알아 둘 것

- 1) 실행 시킬 예제 코드 상위 파일로 이동
- 2) pio run -e teensy41 -t upload 명령으로 코드를 teensy4.1에 업로드 (Arduino IDE에서 코드를 보드에 업로드 시키는 과정과 같음. 터미널에서 명령을 할 뿐)
- 3) ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0 로 Micro-ROS Agent 실행



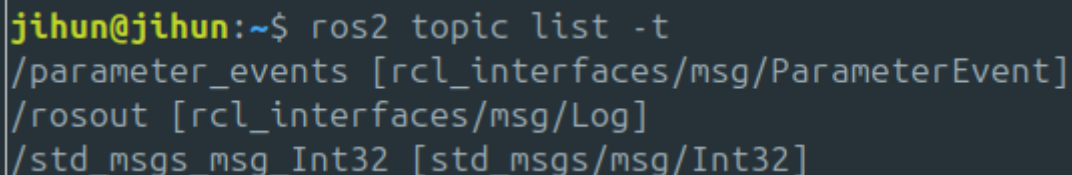
```
jihun@jihun:~$ ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0  
[1754377253.613215] info | TermiosAgentLinux.cpp | init | running... | fd: 18  
[1754377253.613344] info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4  
[1754377254.079537] info | Root.cpp | create_client | create | client_key: 0x33C2F86D, session_id: 0x81  
[1754377254.079630] info | SessionManager.hpp | establish_session | session established | client_key: 0x33C2F86D, address: 0  
[1754377254.088558] info | ProxyClient.cpp | create_participant | participant created | client_key: 0x33C2F86D, participant_id: 0x000(1)  
[1754377254.088808] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x33C2F86D, topic_id: 0x000(2), participant_id: 0x000(1)  
[1754377254.088998] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x33C2F86D, publisher_id: 0x000(3), participant_id: 0x000(1)  
[1754377254.089412] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x33C2F86D, datawriter_id: 0x000(5), publisher_id: 0x000(3)
```

그림 3 Micro-ROS Agent를 실행하면 위 사진처럼 로딩됨

7) 실행 확인

* Micro-ROS Agent를 실행시킨 터미널이 아닌, 새로운 터미널을 열어서 실행함.

```
ros2 topic list
```



```
jihun@jihun:~$ ros2 topic list -t  
/parameter_events [rcl_interfaces/msg/ParameterEvent]  
/rosout [rcl_interfaces/msg/Log]  
/std_msgs_msg_Int32 [std_msgs/msg/Int32]
```

그림 4 실행되고 있는 topic 확인

/parameter_events : 동적 파라미터 관리를 가능하게 해주는 토픽 (기본적으로 있음)

/rosout : ROS 에서 분산된 노드들의 로그를 모니터링 하기위한 표준 통로 (기본적으로 있음)

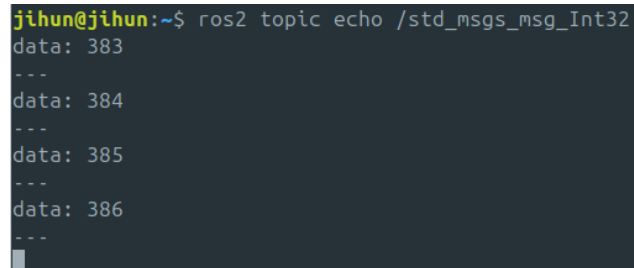
/std_msgs_msg_Int32 : 방금 실행 시킨 예제 코드에서 실행된 토픽 (→ 정상적으로 업로드 됨)

8) 데이터 확인

topic에서 발행하는 데이터 값을 아래 명령으로 확인할 수 있음.

```
ros2 topic echo /std_msgs_msg_Int32
```

* ros2 topic ehco <topic_name>



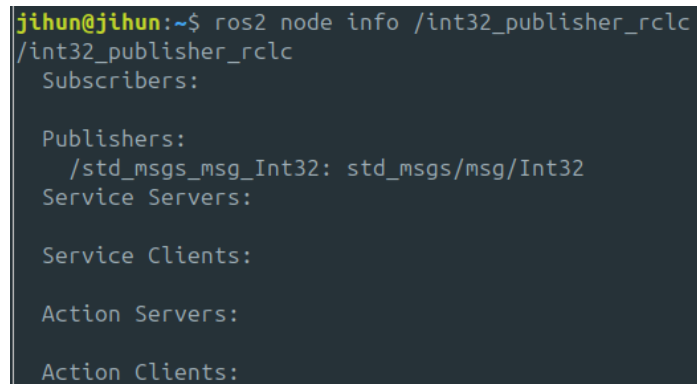
```
jihun@jihun:~$ ros2 topic echo /std_msgs_msg_Int32
data: 383
---
data: 384
---
data: 385
---
data: 386
---
```

그림 5 Micro-ROS를 통해 teensy4.1에 전송되는 데이터

아래 명령을 통해 노드 구조를 확인할 수 있음

```
ros2 node info /int32_publisher_rclc
```

* ros2 node info <node_name>



```
jihun@jihun:~$ ros2 node info /int32_publisher_rclc
/int32_publisher_rclc
Subscribers:

Publishers:
  /std_msgs_msg_Int32: std_msgs/msg/Int32
Service Servers:

Service Clients:

Action Servers:

Action Clients:
```

그림 6 node의 세부사항 - publisher만 설정함

4. Micro-ROS 작업 환경 세팅 : PlatformIO + Vscode

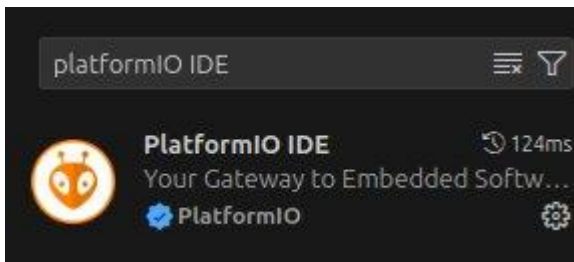
이전 목차에서 Micro-ROS를 설치하고 예제 코드 실행 과정에 대해 알아보았었음. 이때 작업하는 환경은 Ubuntu Terminal 이라는 텍스트 기반 명령어(Command Lin Interface, CLI)에서 운영체제와 상호작용할 수 있는 창에서 실행하였음.

하지만 이러한 환경에서 자신의 프로젝트 코드를 작성하고 실행시키기에는 상당히 번거로움.

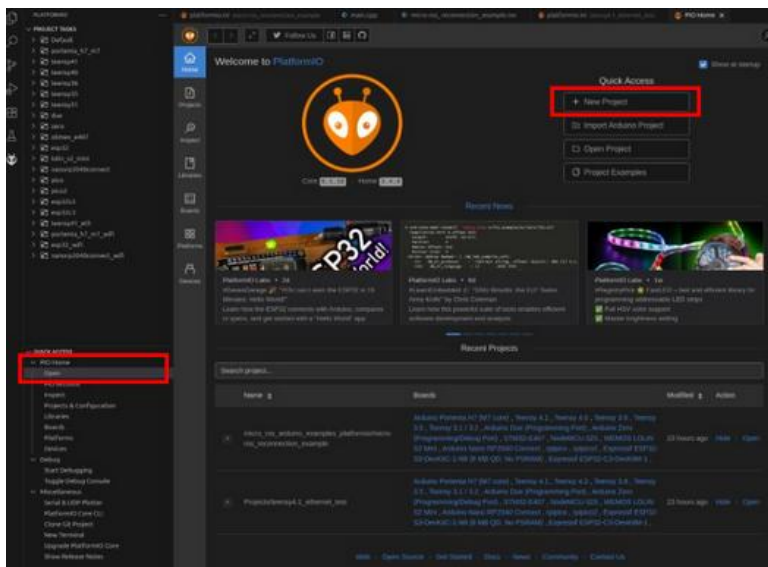
그래서 많은 사람들이 코드를 사용하는 Vscode를 통해 작업할 수 있는 환경을 구성해볼 것임.

* PlatformIO + Vscode 환경 세팅

1. Vscode 확장자에서 PlatformIO IDE를 설치한다.

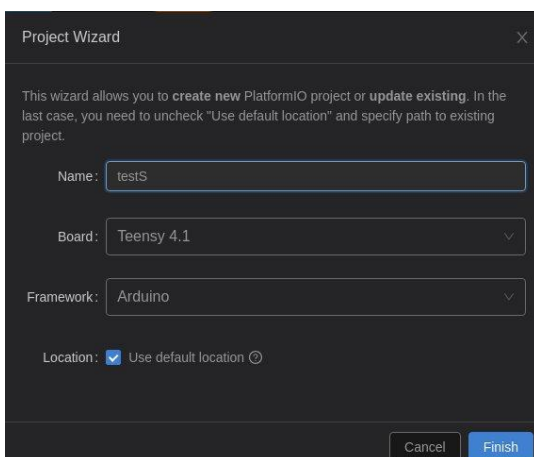


2. 왼쪽 사이드 바에서 PlatformIO IDE(개미 모양)을 클릭한다.



3. 왼쪽 메뉴 창에서 PIO home > open 을 클릭한다. (기본적으로 home 화면이 뜨지만 없다면 실행)

4. New project를 클릭한다.



5. 프로젝트 이름을 정하고, Board에 teensy4.1을 설정해 준다.

6. Finish를 누르면 나만의 프로젝트가 생성된다.

```
jihun@jihun:~/micro_ros_arduino_examples_platformio/micro-ros_reconnection_example$ tree
.
├── micro-ros_reconnection_example.ino
├── platformio.ini
└── src
    └── main.cpp
2 directories, 3 files
```

그림 8 micro-ros 예제 파일 구조

```

testS
├── .pio
├── .vscode
├── include
├── lib
├── src
│   ├── main.cpp
│   └── test
├── .gitignore
└── platformio.ini

```

그림 1 vscode에서 생성한 PlatformIO 파일 구조

예제 파일과 vscode에서 생성한 프로젝트 파일 구조를 보면 vscode의 일부 설정 파일들을 제외하고는 구조가 같은 것을 볼 수 있음. 다만 아두이노 코드(.ino) 와 C++ 코드(main.cpp) 이 두 개로 나누어 지는데, 메인 코드는 둘 중에 아무곳에서나 작성해서 쓰면 됨.

이 메뉴얼에서는 main.cpp을 메인 코드로 하여 작성할 예정.

* PlatformIO + Vscode 초기 세팅

vscode에서 환경을 만들고 난 후, 추가적인 설정이 더 필요함.

vscode에서 환경을 만들게 되면 PlatformIO 툴만 생성이 된 것이고 Board에 대한 통신 설정이 되어 있지 않기 때문에 처음에 다운 받았던 예제 파일에서 옮겨주는 과정이 필요함.

1. Page. 7에 5번에서 설치했던 micro_ros_arduino_examples_platformio 예제 폴더로 이동한다.
2. micro_ros_arduino_examples_platformio/platformio 폴더를 복사해서 /home/\$User/Documents/PlatformIO/Projects 경로에 그대로 붙여준다.
(vscode에서 platformIO 프로젝트 경로가 Documents/PlatformIO에 생성됨)
3. 진행할 프로젝트 코드에서, platformio.ini는 다음과 같이 작성을 한다.

```
[platformio]
extra_configs =
    ../platformio/platformio.ini
```

```
teensy4.1_ethernet_test > 🐛 platformio.ini
1  [platformio]
2  extra_configs =
3  |    ../platformio/platformio.ini
4  |
```

4. 작성할 main.cpp에서는 다음 코드를 추가한다.

```
#include "../platformio/src/setup.cpp"
```

```
teensy4.1_ethernet_test > src > 🐛 main.cpp > ...
1  #include "../platformio/src/setup.cpp"
2
3  #include <Arduino.h>
4
5  #include <rcl/rcl.h>
6  #include <rcl/rcl.h>
```




5. 하드웨어 구성 : Arduino IDE 테스트

Micro-ros를 설치하고 예제 코드를 실행해보았으니, Stepper motor와 Encoder 센서를 활용해서 직접 데이터 값을 주고 받는 것을 해볼 것 임. 본론에 들어가기전에 회로도를 구성하고 Arduino IDE에서 각 부품들을 실행해서 이상없는지부터 확인해볼 것임.

사용 부품 : Rotary encoder(E30s4-1024-6-L-5), Stepper motor(OT-42HS40-005B / NEMA 17 계열)

사용된 MCU : Teensy4.1

사용 프로그램 : Arduino IDE

Teensy4.1	Rotary Encoder	Stepper motor
		

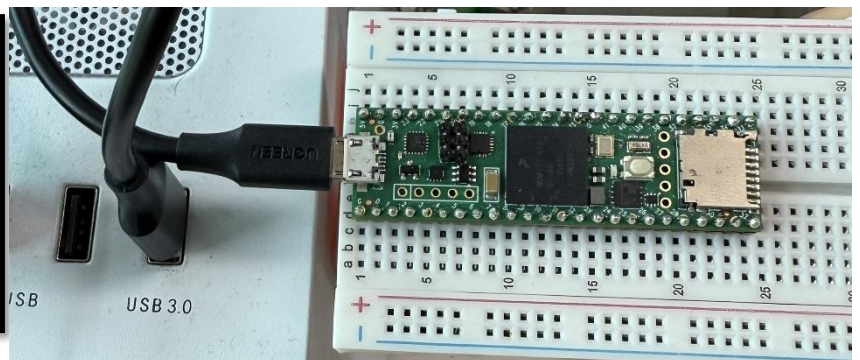
* 각 부품의 회로도 및 테스트 코드 *

1) Teensy 4.1

* USB Type-B Micro (Micro-USB) 케이블로 teensy4.1와 PC에 연결



그림 7 USB Type-B Micro (Micro-USB)



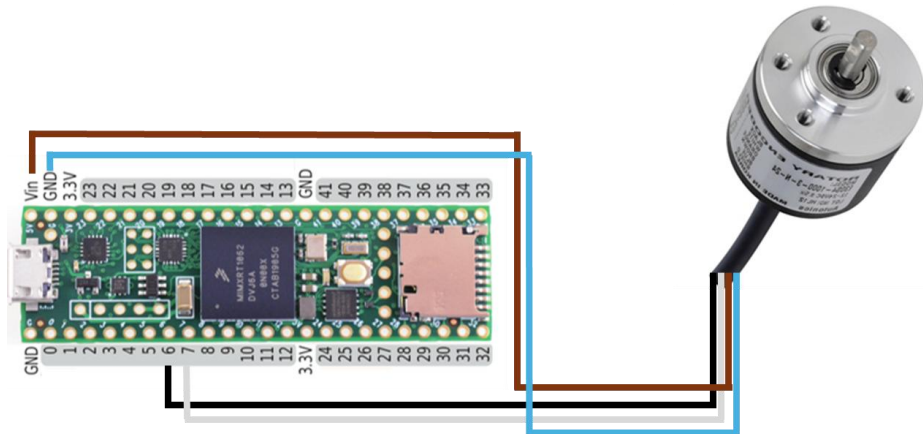
테스트 코드 :

https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/teensy4.1_test/teensy4.1_test.t.ino

2) Rotary Encoder

-회로도

Wire	Pin
A+ (Black)	D6
B+ (White)	D7
+V (Brown)	Vin
0V (Blue)	GND

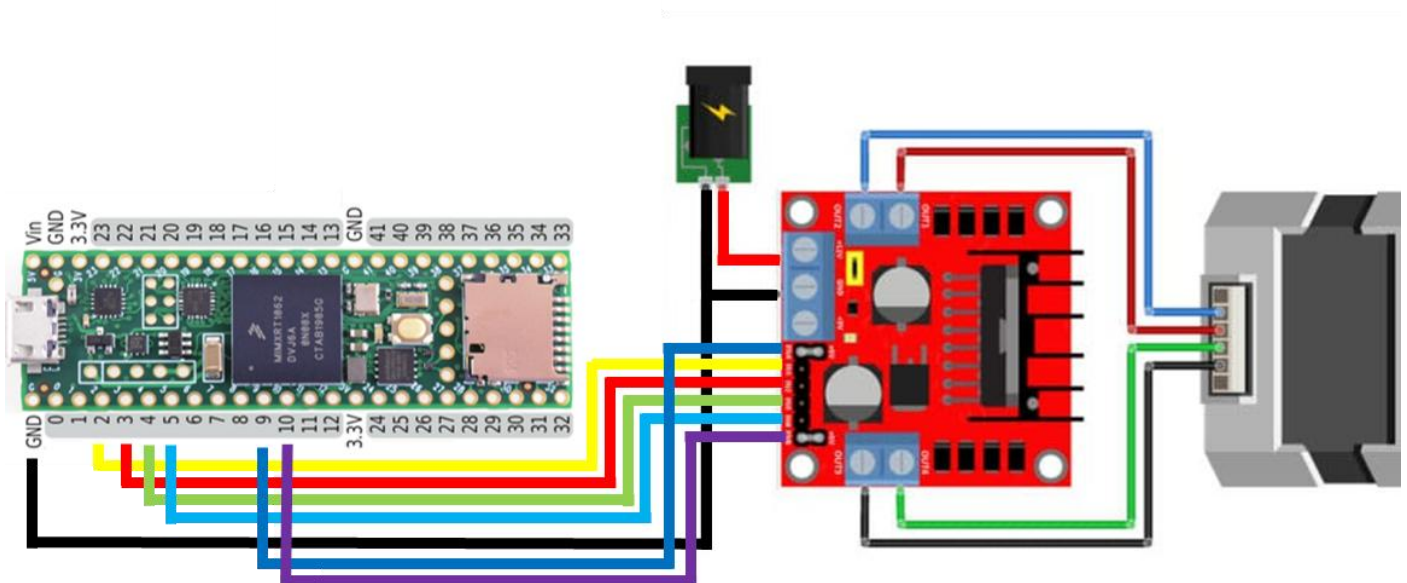


테스트 코드 :

[https://github.com/jihun-](https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/encoder_test/encoder_test.ino)

[2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/encoder_test/encoder_test.ino](https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/encoder_test/encoder_test.ino)

3) Stepper motor (모터드라이브 : L298N)



* L298N에 12V의 외부 전압이 필요함 (power supply : 12 V/ 0.5A 로 설정)

테스트 코드 :

[https://github.com/jihun-](https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/stepper_motor_test/stepper_motor_t)

[2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/stepper_motor_test/stepper_motor_t](https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/stepper_motor_test/stepper_motor_test.ino)
[est.ino](https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/blob/main/Arduino_test/teensy4.1_test/stepper_motor_test/stepper_motor_test.ino)

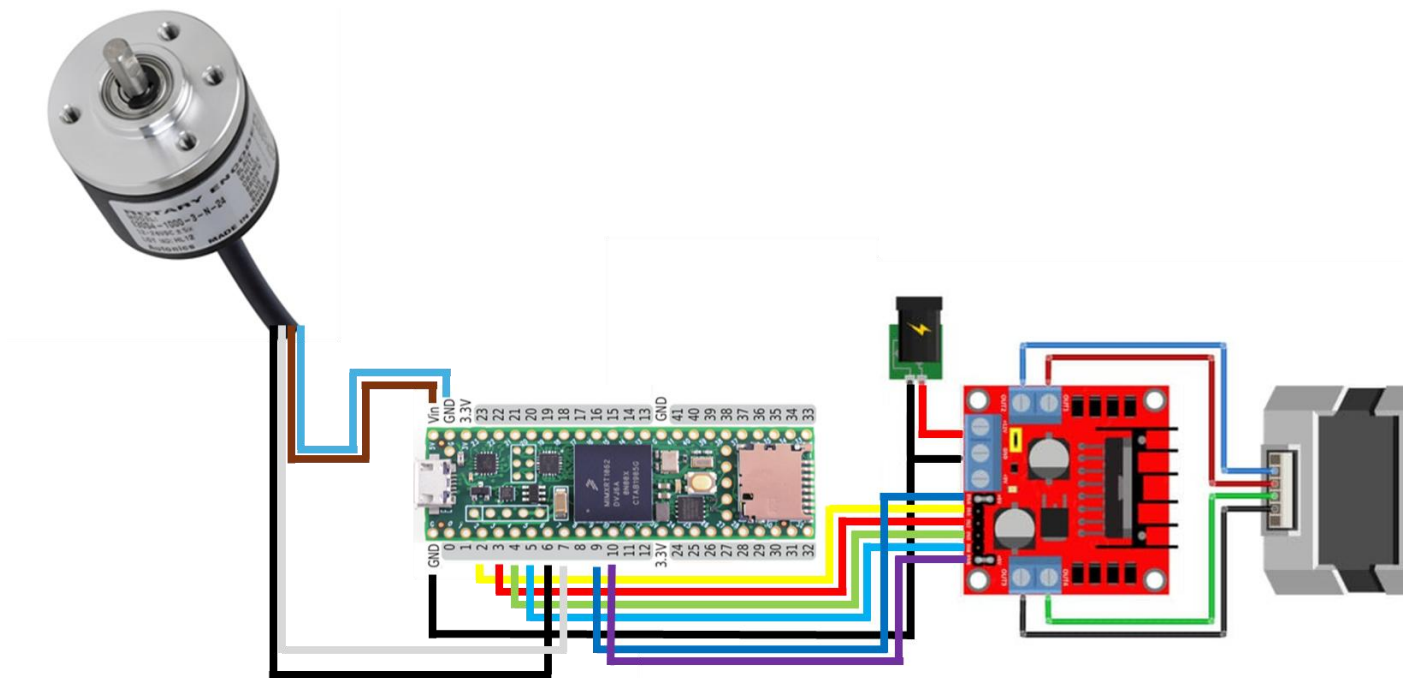
6. Micro-ROS 실행하기

Stepper motor와 엔코더를 활용해서 micro-ros를 통해 publish하고 subscribe하는 코드를 실행시켜볼 것임. 통신설정은 ethernet 연결로 할 예정이니 아래 메뉴얼 참고.

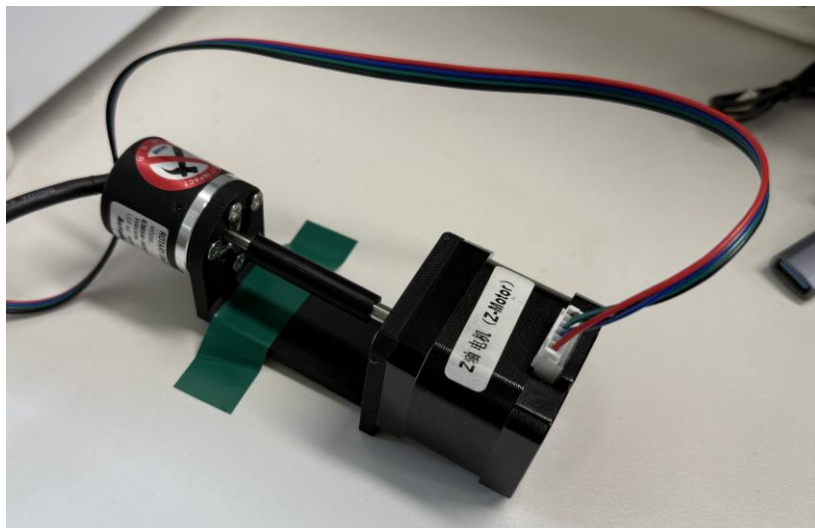
* teensy4.1에 ethernet 연결하기

https://www.pjrc.com/store/ethernet_kit.html

* 전체 회로도



* 하드웨어 구성



* 코드

https://github.com/jihun-2sh/Micro_ros_tutorial_GNU/tree/main/Documents

1. vscode에서 PlatformIO 새로운 프로젝트를 생성한다.
2. github에서 코드를 복사해서 main.cpp에 붙여넣는다.
3. 터미널창을 열고 프로젝트 폴더에서 코드를 업로드 해준다.

```
pio run -e teensy41_eth -t upload
```

```
Opening Teensy Loader...
===== [SUCCESS] Took 10.95 seconds =====
```

Environment	Status	Duration
teensy41_eth	SUCCESS	00:00:10.946

```
===== 1 succeeded in 00:00:10.946 =====
```

4. Micro-ROS agent를 실행시킨다.

```
ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888 -v6
```

```
0000: 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 92 53 CC 42
[1756270404.494153] debug    | UDPv4AgentLinux.cpp | send_message | [** <<UDP>> **] | client_ke
y: 0x3F3411DE, len: 13, data:
0000: 81 00 00 00 0A 01 05 00 41 01 00 00 80
[1756270404.513963] debug    | UDPv4AgentLinux.cpp | recv_message | [==>> UDP <<==] | client_ke
y: 0x3F3411DE, len: 36, data:
0000: 81 80 41 01 07 01 1C 00 01 4B 00 05 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 00
0020: D0 5D CC 42
[1756270404.514282] debug    | DataWriter.cpp      | write       | [** <<DDS>> **] | client_key
: 0x00000000, len: 24, data:
0000: 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 D0 5D CC 42
```

* 연결이 성공되면 터미널 창에 로그가 계속 올라옴.

5. 새로운 터미널 창을 열어 토픽이 제대로 발행되었는지 확인해본다.

```
ros2 topic list
```

```
jihun@jihun:~$ ros2 topic list
/encoder_data
/parameter_events
/rosout
/stepper_cmd
```

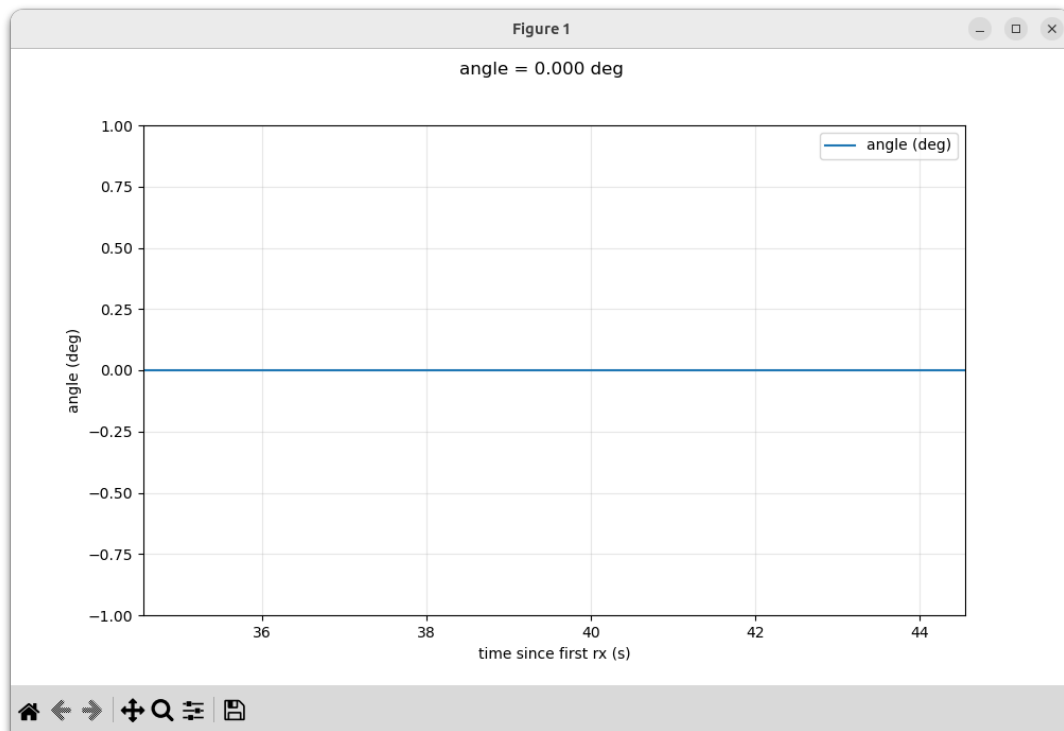
* main.cpp 코드에서 /encoder_data 와 /stepper_cmd 토픽을 만들어 주었었음.

/encoder_data : teensy4.1에서 엔코더 데이터를 받아 PC로 publish

/stepper_cmd : PC에서 모터 명령(각도 데이터)를 받아 MCU로 publish

6. 엔코더 데이터 값을 받아오는 코드를 실행한다.

```
python3 encoder_degree_plot.py
```



7. stepper motor에 각도 명령을 넣어준다.

```
ros2 topic pub -1 /stepper_cmd std_msgs/msg/Float32 "{data: 90.0}"
```

* `ros2 topic pub -1 /stepper_cmd std_msgs/msg/Float32 "{data: <움직일 각도>}"`

