네, 알겠습니다. PYNQ 3.1을 Ubuntu 22.04 LTS 시스템에 설치하여 원격 타겟 보드를 제어하거나 Alveo 카드와 같은 로컬 AMD 가속기를 관리하는 호스트(Host)로 구성하는 방법을 단계별로 안내해 드리겠습니다.

PYNQ 프레임워크는 타겟 보드에 설치되는 임베디드 리눅스 이미지 외에도, 일반적인 x86 머신(예: 데스크톱, 서버)에 Python 패키지 형태로 설치하여 사용할 수 있도록 설계되었습니다.[1] 이 호스트 설치는

PYNQ.remote 기능을 통해 원격 보드를 제어하거나, 데이터센터의 Alveo 카드를 활용하는 데 필수적입니다.[3]

다음은 Ubuntu 22.04 LTS 시스템을 PYNQ 3.1 호스트로 설정하는 절차입니다.

1단계**:** 시스템 사전 요구사항 확인

설치를 시작하기 전에 호스트 시스템이 다음 요구사항을 충족하는지 확인합니다.

1. 운영체제**:** Ubuntu 22.04 LTS (Jammy Jellyfish)가 공식적으로 지원됩니다.[4]
2. **Python** 버전**:** PYNQ 3.1은 Python 3.10과 함께 릴리스되었습니다.[2] Ubuntu 22.04는 기본적으로 Python 3.10을 제공하므로 별도의 Python 설치는 필요하지 않습니다. 프로젝트 간의 의존성 충돌을 방지하기 위해 Python 가상 환경(venv)을 사용하는 것을 강력히 권장합니다.

먼저, python3-venv 패키지를 설치하고 가상 환경을 생성합니다.

Bash

```
# venv 패키지 설치
sudo apt update
sudo apt install python3.10-venv

# 프로젝트를 위한 디렉토리 생성 및 이동
mkdir pynq_host_project
cd pynq_host_project

# 'pynq_env'라는 이름의 가상 환경 생성
python3 -m venv pynq_env
```

```
# 가상 환경 활성화
source pynq_env/bin/activate
```

가상 환경이 활성화되면 터미널 프롬프트 앞에 (pynq_env)가 표시됩니다. 이제부터 설치되는 모든 Python 패키지는 이 격리된 환경에 설치됩니다.

2단계: PyPI를 통해 PYNQ 라이브러리 설치

PYNQ 라이브러리는 Python Package Index(PyPI)에 공식적으로 등록되어 있어 pip를 통해 간단하게 설치할 수 있습니다.[2]

활성화된 가상 환경에서 다음 명령어를 실행합니다.

```
Bash
```

```
pip install pynq
```

이 명령어는 PYNQ 호스트 기능을 수행하는 데 필요한 모든 Python 모듈을 다운로드하고 설치합니다. 여기에는 원격 장치에 연결하고 제어하는 데 사용되는 pynq.remote 클래스가 포함됩니다.

3단계: AMD 런타임(XRT) 설치 (권장)

호스트 머신에서 Alveo 카드를 사용하거나, PYNQ.cpp로 개발된 애플리케이션과 상호작용하는 등 일부 고급 기능을 활용하려면 AMD 런타임(XRT)을 설치해야 합니다.[3]

PYNQ.remote를 통한 단순 원격 제어만 사용할 경우에도 XRT는 PYNQ 생태계의 핵심 구성 요소이므로 설치하는 것이 좋습니다.

PYNQ 3.1은 XRT **2024.1** 버전에 맞춰져 있습니다.[5]

1. **XRT** 다운로드**:** AMD 다운로드 페이지에서 Ubuntu 22.04용 XRT 2024.1 설치 패키지(.deb 파일)를 다운로드합니다.
2. **XRT** 설치**:** 다운로드한 .deb 파일을 사용하여 XRT를 설치합니다. 일반적으로 다음

명령어를 사용합니다.

```Bash
# 다운로드한 파일명으로 대체해야 합니다.
sudo apt install./xrt_202410.2.17.0_22.04-amd64-xrt.deb
```

4단계: 설치 확인

모든 설치가 완료되면 간단한 Python 스크립트를 실행하여 PYNQ 라이브러리가 올바르게 설치되었는지 확인할 수 있습니다.

1. Python 인터프리터를 시작합니다.

   ```Bash
   python3
   ```

2. 인터프리터 내에서 다음 코드를 실행합니다.

   ```Python
   import pynq

   # 설치된 PYNQ 버전 출력
   print(pynq.__version__)
   ```

   오류 없이 PYNQ 버전 번호(예: 3.1.0)가 출력되면 호스트 시스템에 PYNQ 라이브러리가 성공적으로 설치된 것입니다.

3. 연결 테스트 (원격 보드가 준비된 경우): 이전 보고서에서 생성한 원격 PYNQ 타겟 보드가 네트워크에 연결되어 부팅된 상태라면, 다음 스크립트를 사용하여 최종 연결 테스트를 수행할 수 있습니다. test_remote.py 파일을 생성하고 아래 내용을 작성합니다.

   ```Python
   from pynq.remote import RemoteDevice

   # 보드의 호스트명 또는 IP 주소로 변경하세요.
   BOARD_HOSTNAME = "pynq-z2"

   try:
       # 기본 사용자 이름과 비밀번호는 'xilinx' 입니다.
       device = RemoteDevice(BOARD_HOSTNAME, "xilinx", "xilinx")

       print(f"Successfully connected to '{device.name}'")
       print(f"PYNQ Version: {device.pynq_version}")
       print(f"OS: {device.operating_system}")
       print(f"Target Temperature: {device.sensors['temp0']['value']} C")
   ```

```
except Exception as e:
    print(f"Failed to connect to the board: {e}")
```

스크립트를 실행합니다.

```
Bash
python3 test_remote.py
```

성공적으로 연결되면 보드의 이름, PYNQ 버전, OS 정보 등이 출력됩니다.

이상의 단계를 완료하면 Ubuntu 22.04 LTS 시스템이 PYNQ 3.1 호스트로 기능할 모든 준비를 마친 것입니다. 이제 이 호스트를 사용하여 원격 PYNQ 보드를 제어하고 고성능 하이브리드 애플리케이션을 개발할 수 있습니다.