# **pyadi-iio** 라이브러리를 이용한 **ANTSDR E200**의 **Python** 기반 테스트 **5**단계 전문가 가이드

## 서론**: Python**으로 **ANTSDR**의 잠재력 활용하기

Microphase사의 ANTSDR E200은 AMD/Xilinx Zynq-7020 SoC(System-on-Chip)와 Analog Devices의 AD936x RFIC(Radio-Frequency Integrated Circuit) 제품군을 기반으로 제작된 고성능, 다목적 소프트웨어 정의 라디오(SDR) 플랫폼이다.[1] ANTSDR E200은 원격 작동을 위한 기가비트 이더넷 인터페이스, 더욱 강력한 FPGA, 그리고 향상된 오실레이터 정밀도와 같은 주요 이점을 통해 동급의 다른 플랫폼들과 차별화된다.[2]

본 보고서는 "pySDR"을 테스트하기 위한 가이드를 제공한다. 여기서 "pySDR"이라는 용어는 여러 의미로 해석될 수 있으나, 본 문서에서는 Python을 사용한 SDR 입문용 온라인 교재인 pysdr.org에서 제시하는 방법론을 적용하는 것으로 정의한다.[5] 이 접근법의 핵심은 ANTSDR이 ADALM-PLUTO SDR과 펌웨어 수준에서 높은 호환성을 갖는다는 점에 있다.[2] 이러한 호환성 덕분에 Analog Devices의 산업용 I/O(Industrial I/O, IIO) 드라이버 스택을 활용할 수 있다. 따라서 본 가이드 전반에 걸쳐 사용될 Python 인터페이스는 ADI의 IIO 장치를 위한 공식 지원 Python 래퍼(wrapper)이자

pysdr.org의 PlutoSDR 튜토리얼에서 비중 있게 다루어지는 pyadi-iio 라이브러리가 될 것이다.[9]

이러한 접근 방식은 ANTSDR의 전략적 장점을 극대화한다. ANTSDR은 IIO와 UHD라는 두 가지 주요 API를 모두 지원함으로써, PlutoSDR로 대표되는 취미 개발자 생태계와 USRP로 대표되는 전문 연구 생태계 사이의 교량 역할을 수행한다.[2] 본 가이드는 이 중 IIO/Pluto 경로를 의도적으로 선택하였는데, 이는 Python 기반 애플리케이션 개발에 가장 직접적이고 잘 지원되는 경로를 제공하여 사용자의 요구사항에 정확히 부합하기 때문이다.

본 보고서는 하드웨어 준비, 호스트 환경 및 드라이버 설치, Python 환경 설정, 초기 데이터 수신 스크립트 작성, 그리고 실시간 스펙트럼 분석기 애플리케이션을 통한 종합 검증에 이르는 다섯 가지 논리적 단계로 구성된다. 각 단계는 사용자가 ANTSDR E200을 성공적으로 Python 환경과 연동하고 그 기능을 검증할 수 있도록 명확하고 상세한 지침을 제공할 것이다.

# **1**단계**:** 하드웨어 준비 및 네트워크 검증

첫 단계는 ANTSDR 하드웨어를 물리적으로 설정하고 호스트 컴퓨터와의 기본적인 네트워크 통신을 확인하는 것이다. 이 단계의 성공적인 완료는 이후의 모든 소프트웨어 설정 과정의 기반이 된다.

## 개봉 및 물리적 연결

ANTSDR E200 패키지에는 일반적으로 본체, 모노폴 안테나 2개, 이더넷 케이블, 전원 공급용 USB-C 케이블 등이 포함된다.[2] 연결 절차는 다음과 같다.

1. 제공된 두 개의 모노폴 안테나를 MIMO(Multiple-Input Multiple-Output) 작동을 위해 장치 전면의 SMA 포트에 각각 연결한다.
2. 기가비트 이더넷 케이블을 ANTSDR의 RJ45 포트와 호스트 컴퓨터 또는 네트워크 스위치에 연결한다.
3. USB-C 케이블을 ANTSDR의 전원 포트와 적절한 USB 전원 공급 장치에 연결하여 전원을 공급한다.

여기서 중요한 점은 ANTSDR E200의 USB 포트는 주로 전원 공급 및 콘솔 접근용으로 사용되며, ADALM-PLUTO와 달리 고속 데이터 전송에는 사용되지 않는다는 것이다.[14] 데이터 통신은 전적으로 이더넷 인터페이스를 통해 이루어진다.

## **PlutoSDR** 호환성을 위한 부팅 모드 설정

이 단계는 ANTSDR을 pyadi-iio 라이브러리와 함께 사용하기 위한 가장 핵심적인 하드웨어 설정이다. ANTSDR은 두 가지 펌웨어 모드를 지원하며, 부팅 모드 선택은 물리적인 DIP 스위치를 통해 이루어진다.

- 스위치 위치: 부팅 모드 DIP 스위치는 일반적으로 이더넷 네트워크 포트 아래에서 찾을 수 있다.[8] 케이스가 장착된 경우, 'BOOT QSPI SD'라는 문구를 통해 위치를 쉽게 식별할 수 있다.
- 설정: 스위치를 **QSPI** 위치로 설정해야 한다. 이 설정은 공장 출하 시 온보드 QSPI 플래시 메모리에 내장된 PlutoSDR 호환 펌웨어를 로드한다. 반대편의 "SD" 모드는 UHD(USRP

Hardware Driver) 펌웨어를 SD 카드로부터 부팅할 때 사용된다.[8]

이 설정을 올바르게 하지 않으면, 장치는 IIO 드라이버와 호환되지 않는 모드로 부팅되어 2단계 이후의 모든 과정이 실패하게 된다.

## 네트워크 구성 및 연결 테스트

ANTSDR이 Pluto 펌웨어로 부팅되면 네트워크 어댑터로 동작하며 IP 주소를 할당받는다. 기본적으로 이 주소는 192.168.2.1로 설정되어 있는 경우가 많다.[8] 호스트 컴퓨터에서 ANTSDR에 접근하기 위해서는 동일한 네트워크 서브넷에 있도록 호스트의 네트워크 설정을 변경해야 한다.

1. 호스트 **IP** 설정: 호스트 컴퓨터의 이더넷 어댑터에 고정 IP 주소를 설정한다. 예를 들어, IP 주소를 192.168.2.10으로, 서브넷 마스크를 255.255.255.0으로 설정한다.
2. 연결 확인: 터미널 또는 명령 프롬프트를 열고 다음 명령을 실행하여 ANTSDR과의 네트워크 연결을 확인한다.
   Bash
   ```
   ping 192.168.2.1
   ```

성공적인 응답(reply)은 하드웨어 전원이 켜져 있고, 펌웨어가 정상적으로 로드되었으며, 물리적 네트워크 링크가 활성화되었음을 의미한다. 만약 ping이 실패한다면, 케이블 연결, 호스트 IP 설정, 그리고 방화벽 설정을 다시 확인해야 한다. ANTSDR의 이더넷 의존성은 원격 및 분산 애플리케이션과 같은 강력한 전문 기능을 가능하게 하지만, PlutoSDR의 간단한 USB 연결 모델에 비해 네트워크 구성 및 문제 해결(방화벽, 서브넷 불일치 등)이라는 추가적인 복잡성을 야기한다. 따라서 ping 실패는 하드웨어 결함보다는 사용자 측의 네트워크 설정 문제일 가능성이 높다.

여러 대의 ANTSDR을 동일 네트워크에서 사용해야 할 경우, 각 장치의 IP 주소를 변경해야 한다. 이는 SSH(사용자명: root, 비밀번호: analog)를 통해 장치에 접속한 후 fw_setenv 명령어를 사용하여 수행할 수 있다.[8]

본 가이드의 목적에 맞게 ANTSDR의 성능을 명확히 이해하기 위해, 널리 알려진 ADALM-PLUTO와의 주요 사양을 비교한 표는 다음과 같다. 이 표는 사용자가 ANTSDR 하드웨어의 성능적 이점을 명확히 인지하는 데 도움을 준다.

표 **1: ANTSDR E200** 대 **ADALM-PLUTO** 사양 비교

| 사양 | ANTSDR E200 (AD9361 버전) | ADALM-PLUTO |
|---|---|---|

| **RFIC** | Analog Devices AD9361 | Analog Devices AD9363 |
|---|---|---|
| 주파수 범위 | 70 MHz – 6 GHz | 325 MHz – 3.8 GHz |
| **RF** 대역폭 | 최대 56 MHz | 최대 20 MHz |
| 샘플링 속도 | 최대 61.44 MSPS | 최대 61.44 MSPS |
| **ADC** 해상도 | 12 bits | 12 bits |
| **TX/RX** 채널 | 2x2 MIMO (Full-Duplex) | 2x2 MIMO (Full-Duplex) |
| 연결 인터페이스 | 기가비트 이더넷 | USB 2.0 |
| **FPGA** | Xilinx Zynq 7020 (85k 로직 셀) | Xilinx Zynq 7010 (28k 로직 셀) |
| 오실레이터 정밀도 | ±2 ppm (초기) | ±20 ppm (초기) |

데이터 출처: [1]

이 표는 ANTSDR이 더 넓은 주파수 범위, 더 큰 RF 대역폭, 더 강력한 FPGA, 그리고 월등히 우수한 클럭 안정성을 제공함을 명확히 보여준다. 특히 기가비트 이더넷 인터페이스는 USB 2.0의 대역폭 한계를 극복하고 원격 운용의 유연성을 제공하는 핵심적인 차별점이다.

## **2**단계**:** 호스트 환경 및 코어 **libiio** 드라이버 설치

하드웨어 준비가 완료되면, 다음 단계는 호스트 컴퓨터에 ANTSDR과 통신하는 데 필요한 핵심 소프트웨어 드라이버를 설치하는 것이다. 이 과정의 중심에는 Analog Devices의 libiio 라이브러리가 있다.

### **Linux** 호스트 시스템 준비

안정성과 광범위한 커뮤니티 지원을 고려하여 **Ubuntu 22.04 LTS**를 호스트 운영 체제로 사용할 것을 권장한다. Windows 사용자의 경우, **WSL2(Windows Subsystem for Linux 2)**를 통해 Ubuntu 22.04 배포판을 설치하면 본 가이드의 모든 작업을 거의 동일하게 수행할 수 있으며, 뛰어난 호환성을 보여준다.[14]

드라이버와 Python 라이브러리를 소스 코드로부터 빌드하기 위해 필요한 사전 패키지들을 설치해야 한다. 다음 명령어를 터미널에서 실행하여 필수 개발 도구들을 한 번에 설치한다. 이 명령어는 여러 정보 소스에서 필요한 패키지들을 종합한 것이다.[16]

Bash

```
sudo apt-get update
sudo apt-get install -y git cmake build-essential pkg-config libusb-1.0-0-dev libboost-all-dev
```

## **libiio** 클로닝 및 컴파일

libiio는 하드웨어의 IIO 드라이버와 상호작용하기 위한 저수준(low-level) 인터페이스를 제공하는 핵심 C 라이브러리다.[9] 최신 하드웨어와의 완벽한 호환성을 보장하기 위해, 패키지 관리자를 통해 구버전을 설치하는 대신 공식 소스 코드 저장소에서 직접 빌드하는 것이 바람직하다. SDR 하드웨어와 펌웨어는 지속적으로 개발되므로 [22], 소스 코드에서 직접 컴파일하면 최신 기능과 버그 수정 사항을 즉시 적용할 수 있다. 이는 전문가 수준의 개발 환경에서 발생할 수 있는 진단하기 어려운 호환성 문제를 예방하는 최선의 방법이다.

다음은 libiio를 설치하는 정확한 명령어 순서이다.

1. 소스 코드 클론: 공식 GitHub 저장소에서 최신 소스 코드를 내려받는다.
   Bash
   ```
   git clone https://github.com/analogdevicesinc/libiio.git
   cd libiio
   ```

2. 빌드 디렉토리 생성: 소스 트리 외부에서 빌드를 진행하는 것이 좋다.
   Bash
   ```
   mkdir build
   cd build
   ```

3. **CMake**를 이용한 빌드 구성: cmake를 사용하여 빌드 환경을 구성한다. 여기서 가장 중요한 옵션은 -DPYTHON_BINDINGS=ON이다. 이 플래그는 3단계에서 필요한 Python 바인딩을

함께 빌드하도록 지시한다. 이 단계를 생략하면 Python에서 libiio를 사용할 수 없다.[19]
Bash
cmake -DPYTHON_BINDINGS=ON..

4. 컴파일 및 설치: make 명령어로 소스 코드를 컴파일하고, sudo make install로 시스템에 설치한다.
Bash
make -j$(nproc)
sudo make install

5. 공유 라이브러리 캐시 업데이트: 시스템이 새로 설치된 공유 라이브러리(libiio.so)를 찾을 수 있도록 라이브러리 캐시를 업데이트한다.
Bash
sudo ldconfig

## 드라이버 레벨 통신 검증

libiio가 성공적으로 설치되면, iio_info라는 유용한 명령줄 유틸리티가 함께 설치된다. 이 도구는 호스트 머신이 드라이버 수준에서 ANTSDR과 정상적으로 통신할 수 있는지 확인하는 가장 확실한 방법이다.

터미널에서 다음 명령을 실행한다. -n 플래그는 네트워크를 통해 특정 IP 주소의 장치를 찾도록 지시한다.

Bash

iio_info -n 192.168.2.1

명령이 성공적으로 실행되면, ANTSDR에서 발견된 IIO 장치들의 목록이 출력된다. 예상 출력에는 ad9361-phy와 같은 핵심 장치가 포함되어야 한다.[24] 이 출력을 확인하는 것은 하드웨어, 네트워크, 그리고 저수준 드라이버가 모두 올바르게 작동하고 있음을 증명하는 중요한 이정표이다. 이 검증을 통과해야만 다음 단계인 Python 환경 설정으로 넘어갈 수 있다.

# 3단계: Python 환경 및 pyadi-iio 라이브러리 설정

저수준 드라이버 설치가 완료되었으므로, 이제 Python 환경을 구성하고 ANTSDR을 제어할 고수준(high-level) 라이브러리를 설치할 차례이다. 이 단계에서는 격리된 개발 환경을 구축하고 pyadi-iio 라이브러리를 설치한다.

## 격리된 Python 환경 구축

프로젝트별 의존성을 관리하고 시스템의 기본 Python 패키지와의 충돌을 방지하기 위해 Python 가상 환경을 사용하는 것이 현대적인 개발 방식의 표준이다.[20] Python에 내장된

venv 모듈을 사용하여 가상 환경을 생성하고 활성화하는 절차는 다음과 같다.

1. 가상 환경 생성: 프로젝트 디렉토리에서 다음 명령을 실행하여 antsdr_env라는 이름의 가상 환경을 생성한다.
   Bash
   ```
   python3 -m venv antsdr_env
   ```

2. 가상 환경 활성화: 생성된 환경을 현재 터미널 세션에서 활성화한다.
   Bash
   ```
   source antsdr_env/bin/activate
   ```

   활성화되면 터미널 프롬프트 앞에 (antsdr_env)와 같은 표시가 나타나, 현재 가상 환경 내에서 작업 중임을 알려준다.

## 필수 Python 패키지 설치

가상 환경이 활성화된 상태에서 pip를 사용하여 필요한 라이브러리들을 설치한다.

- **pyadi-iio**: 이 패키지는 2단계에서 설치한 libiio 라이브러리를 위한 고수준 Python 래퍼이다. 복잡한 libiio API를 사용하기 쉬운 Python 객체로 추상화하여 제공한다.[11]
- **numpy**: IQ 데이터와 같은 수치 데이터를 효율적으로 처리하기 위한 필수 라이브러리다.[11]
- **matplotlib**: 5단계에서 스펙트럼 분석 결과를 시각화하는 데 사용될 플로팅 라이브러리다.[6]

다음 명령어를 사용하여 이 패키지들을 한 번에 설치한다.

Bash

```
pip install pyadi-iio numpy matplotlib
```

pip install pyadi-iio 명령은 의존성 패키지인 pylibiio를 자동으로 함께 설치한다. pylibiio는 실제 Python 코드와 C로 작성된 libiio 라이브러리를 연결하는 바인딩 역할을 수행한다.[20]

이러한 소프트웨어 스택은 계층적 구조를 가진다: 하드웨어 -> IIO 커널 드라이버 -> libiio (C 라이브러리) -> pylibiio (바인딩) -> pyadi-iio (Python 추상화 계층). 이 단계에서 문제가 발생한다면, 대부분은 하위 계층의 문제(예: libiio가 올바르게 설치되지 않았거나 sudo ldconfig가 실행되지 않은 경우)에서 기인한다. 이 의존성 구조를 이해하는 것은 문제 발생 시 체계적인 디버깅을 가능하게 하는 핵심이다.

## **Python** 설치 검증

패키지 설치가 완료된 후, pyadi-iio 라이브러리가 올바르게 임포트되고 2단계에서 설치한 공유 라이브러리(libiio.so)와 성공적으로 연결되는지 확인해야 한다. Python 인터프리터를 실행하거나 간단한 스크립트를 작성하여 다음 코드를 실행한다.

Python

```
import adi
import iio
import sys

try:
    print(f"pyadi-iio version: {adi.__version__}")
    print(f"libiio version: {iio.version}")
    print(f"Python version: {sys.version}")
except Exception as e:
    print(f"An error occurred: {e}")
```

이 코드가 ImportError나 공유 라이브러리를 찾을 수 없다는 오류 없이 각 라이브러리의 버전

정보를 성공적으로 출력하면, Python 환경 설정이 올바르게 완료된 것이다. 이로써 실제 하드웨어와 통신하는 코드를 작성할 준비가 모두 끝났다.

# 4단계: 초기 Python 스크립팅: 연결 및 IQ 데이터 수신

이제 모든 설정이 완료되었으므로, Python 스크립트를 통해 ANTSDR에 실제로 연결하고 무선 신호 데이터를 수신하는 과정을 진행한다. 이 단계는 전체 시스템이 유기적으로 동작하는지를 처음으로 확인하는 과정이다.

## 첫 SDR 스크립트 작성

먼저, test_rx.py와 같은 이름으로 새로운 Python 파일을 생성한다. 스크립트의 시작 부분에는 pyadi-iio 라이브러리와 numpy를 임포트한다.

Python

```
import adi
import numpy as np
```

## ANTSDR에 연결하기

pyadi-iio 라이브러리는 URI(Universal Resource Indicator)를 사용하여 통신할 대상 장치를 지정한다.[24] ANTSDR은 이더넷을 통해 연결되므로, IP 주소를 포함하는 URI를 사용한다. ANTSDR이 PlutoSDR 호환 펌웨어로 실행되고 있으므로,

adi.Pluto 클래스를 사용하여 장치 객체를 생성할 수 있다.

Python

```
# ANTSDR의 IP 주소를 사용하여 URI 정의
sdr_ip = "ip:192.168.2.1"

# Pluto 클래스를 사용하여 SDR 장치 객체 생성
try:
    sdr = adi.Pluto(uri=sdr_ip)
    print("Successfully connected to ANTSDR.")
except Exception as e:
    print(f"Error connecting to SDR: {e}")
    exit()
```

adi.ad9361(uri=sdr_ip) 클래스를 사용하는 것도 유효한 대안이다. 이는 장치의 핵심 RFIC를 직접 지정하는 방식으로, 두 방법 모두 동일하게 작동한다.[29]

## 핵심 수신 파라미터 설정

SDR 객체가 성공적으로 생성되면, 신호를 수신하기 전에 몇 가지 핵심 파라미터를 설정해야 한다. pyadi-iio는 복잡한 하드웨어 레지스터나 드라이버 속성을 간단한 Python 객체의 속성(property)으로 추상화하여 제공한다. 이 설계 덕분에 코드가 매우 직관적이고 가독성이 높으며, 다른 ADI 장치 간에도 코드 이식성이 좋다. 이는 복잡한 RF 하드웨어 제어에 대한 진입 장벽을 크게 낮추는 강력한 기능이다.

다음은 주요 수신 파라미터를 설정하는 예시 코드이다. 각 라인에는 해당 파라미터의 역할에 대한 설명이 포함되어 있다.[9]

Python

```
# 샘플링 속도 설정 (초당 샘플 수)
sdr.sample_rate = int(2.5e6)  # 2.5 MSPS

# 수신 중심 주파수(LO 주파수) 설정 (Hz 단위)
sdr.rx_lo = int(915e6)  # 915 MHz (ISM 대역)

# 수신 RF 대역폭 설정 (Hz 단위)
sdr.rx_rf_bandwidth = int(2e6)  # 2 MHz
```

```
# rx() 호출 시 수신할 샘플의 개수 (버퍼 크기)
sdr.rx_buffer_size = 1024

# 수신 이득(gain) 제어 모드를 'manual'로 설정
sdr.gain_control_mode_chan0 = 'manual'

# 수동 수신 이득 설정 (dB 단위, 최대값은 장치마다 다름)
sdr.rx_hardwaregain_chan0 = 70.0

print(f"Sample Rate: {sdr.sample_rate / 1e6} MSPS")
print(f"RX LO Frequency: {sdr.rx_lo / 1e6} MHz")
print(f"RX RF Bandwidth: {sdr.rx_rf_bandwidth / 1e6} MHz")
print(f"RX Buffer Size: {sdr.rx_buffer_size} samples")
print(f"RX Gain: {sdr.rx_hardwaregain_chan0} dB")
```

## IQ 데이터 수신 및 검사

파라미터 설정이 완료되면, sdr.rx() 메소드를 호출하여 IQ 데이터를 수신할 수 있다. 이 메소드는 rx_buffer_size에 설정된 개수만큼의 샘플이 수집될 때까지 실행을 멈추고(blocking), 수집된 데이터를 NumPy 배열 형태로 반환한다.[9]

Python

```
# IQ 샘플 수신
samples = sdr.rx()

# 수신된 데이터 검사
print(f"\nReceived {len(samples)} samples.")
print(f"Data type of the array: {samples.dtype}")
print(f"First 5 samples: {samples[:5]}")
```

성공적으로 실행되면, 수신된 샘플의 개수와 데이터 타입이 출력된다. 데이터 타입은 complex64 또는 complex128로, 각 샘플이 실수부(I, In-phase)와 허수부(Q, Quadrature)로 구성된 복소수임을 나타낸다. 이 출력을 통해 ANTSDR이 성공적으로 신호를 디지털 IQ 샘플로 변환하여 네트워크를 통해 Python 스크립트로 전달했음을 최종적으로 확인할 수 있다.

# **5**단계**:** 종합 검증**:** 실시간 스펙트럼 분석기 구축

마지막 5단계는 지금까지의 모든 설정을 종합하여 실질적인 애플리케이션을 구축하고, 이를 통해 시스템 전체가 올바르게 작동하는지 시각적으로 검증하는 과정이다. 무선 신호를 주파수 영역에서 분석하는 스펙트럼 분석기는 모든 RF 엔지니어에게 가장 기본적인 도구이며, 이를 구현하는 것은 시스템 검증을 위한 이상적인 방법이다. 이 접근 방식은 pysdr.org의 교육적 방법론과도 일치한다.[9]

이 최종 단계는 단순한 테스트를 넘어, 향후 사용자가 개발할 모든 수신 기반 SDR 애플리케이션의 기초가 되는 재사용 가능한 코드 베이스를 제공한다. '수집 -> 처리 -> 시각화'라는 핵심적인 순환 구조는 복조기, 디코더, 신호 분류기 등 다양한 애플리케이션의 기본 패턴이 되기 때문에, 이 단계의 완성은 사용자가 원시 샘플을 수신하는 것에서 한 걸음 더 나아가 의미 있는 신호 분석으로 나아갈 수 있도록 돕는다.

## 스펙트럼 시각화를 통한 검증 목표

이 단계의 목표는 ANTSDR로부터 지속적으로 IQ 데이터를 수신하고, 이를 주파수 스펙트럼으로 변환하여 실시간으로 화면에 표시하는 것이다. 이를 통해 특정 주파수 대역에 존재하는 신호를 시각적으로 확인할 수 있으며, 이는 전체 신호 경로(하드웨어, 네트워크, 드라이버, Python 코드)가 완벽하게 연동되고 있음을 최종적으로 증명한다.

## 처리 및 플로팅 로직 추가

4단계에서 작성한 스크립트를 확장하여 matplotlib.pyplot을 시각화에 사용하고, while True: 루프를 통해 지속적인 데이터 캡처와 처리가 가능하도록 구조를 변경한다.

핵심 로직은 다음과 같은 단계로 구성된다.

1. 데이터 캡처: 루프 내에서 samples = sdr.rx()를 호출하여 새로운 IQ 데이터 블록을 수신한다.
2. **FFT** 계산: 수신된 시간 영역의 샘플을 주파수 영역으로 변환하기 위해 고속 푸리에 변환(FFT)을 적용한다. numpy.fft.fft 함수를 사용하고, numpy.fft.fftshift를 통해 스펙트럼의 중심이 0 Hz가 되도록 재정렬한다.
3. 전력 스펙트럼 밀도**(PSD)** 계산: FFT 결과인 복소수 값을 데시벨(dB) 단위의 전력 값으로

변환한다. 일반적으로 PSDdB=20log10(|FFTresult|) 공식을 사용한다.

4. 주파수 축 생성: 플롯의 x축을 올바르게 표시하기 위해 numpy.fft.fftfreq 함수를 사용하여 각 FFT 결과값에 해당하는 주파수 값을 계산한다.
5. 플로팅: matplotlib을 사용하여 계산된 전력 스펙트럼을 주파수 축에 대해 플로팅한다. plt.cla(), plt.plot(), plt.pause(0.01) 등의 함수를 사용하여 루프 내에서 동적으로 플롯을 갱신한다.

## 완성된 실시간 스펙트럼 분석기 코드

다음은 위 로직을 모두 포함한 완성된 Python 스크립트이다. 각 코드 블록에는 상세한 주석이 포함되어 있다.

Python

```
import adi
import numpy as np
import matplotlib.pyplot as plt

# --- 1. SDR 설정 ---
sdr_ip = "ip:192.168.2.1"
try:
    sdr = adi.Pluto(uri=sdr_ip)
except Exception as e:
    print(f"Error connecting to SDR: {e}")
    exit()

sdr.sample_rate = int(2.5e6)
sdr.rx_lo = int(915e6)  # 915 MHz
sdr.rx_rf_bandwidth = int(2e6)
sdr.rx_buffer_size = 4096
sdr.gain_control_mode_chan0 = 'manual'
sdr.rx_hardwaregain_chan0 = 70.0

# --- 2. Matplotlib 설정 ---
plt.ion()  # 대화형 모드 활성화
fig, ax = plt.subplots()
line, = ax.plot(,)
```

```
ax.set_xlabel("Frequency (MHz)")
ax.set_ylabel("Power (dB)")
ax.set_ylim(-80, 0) # Y축 범위 설정

# --- 3. 실시간 처리 루프 ---
try:
    while True:
        # IQ 샘플 수신
        samples = sdr.rx()

        # FFT 계산
        fft_size = len(samples)
        fft_result = np.fft.fftshift(np.fft.fft(samples))

        # 전력 스펙트럼 밀도(PSD) 계산 (dB 단위)
        # 작은 값 추가(1e-15)로 log(0) 오류 방지
        psd = 20 * np.log10(np.abs(fft_result) / fft_size + 1e-15)

        # 주파수 축 생성
        freq_axis = np.fft.fftshift(np.fft.fftfreq(fft_size, 1/sdr.sample_rate))
        freq_axis_mhz = (freq_axis + sdr.rx_lo) / 1e6 # 중심 주파수 반영 및 MHz 단위 변환

        # 플롯 데이터 업데이트
        line.set_xdata(freq_axis_mhz)
        line.set_ydata(psd)

        # 플롯 축 범위 자동 조정
        ax.set_xlim(freq_axis_mhz, freq_axis_mhz[-1])
        ax.set_title(f"Live Spectrum at {sdr.rx_lo / 1e6:.2f} MHz")
        ax.grid(True)

        # 플롯 다시 그리기
        fig.canvas.draw()
        fig.canvas.flush_events()

except KeyboardInterrupt:
    print("\nStreaming stopped by user.")
finally:
    plt.ioff() # 대화형 모드 비활성화
    plt.show() # 최종 플롯 보여주기
```

### 테스트 실행 및 결과 확인

위 스크립트를 spectrum_analyzer.py로 저장하고 python spectrum_analyzer.py 명령으로 실행한다. 실행하면 실시간으로 업데이트되는 스펙트럼 플롯 창이 나타난다. sdr.rx_lo 값을 주변의 알려진 신호 소스(예: 지역 FM 라디오 방송국 주파수인 88-108 MHz 대역)로 변경하고 스크립트를 다시 실행하면, 해당 주파수에서 뚜렷한 피크(peak)를 관찰할 수 있다. 이 시각적인 확인은 ANTSDR 시스템이 성공적으로 설정되었음을 보여주는 가장 확실한 증거이다.

## 결론 및 향후 방향

본 보고서는 ANTSDR E200 SDR 플랫폼을 Python 환경에서 테스트하기 위한 5단계의 포괄적인 가이드를 제시했다. 사용자는 하드웨어 준비 및 네트워크 확인부터 시작하여, 호스트 시스템에 libiio 드라이버를 설치하고, 격리된 Python 환경을 구성했으며, pyadi-iio 라이브러리를 사용하여 IQ 데이터를 수신하는 초기 스크립트를 작성했다. 마지막으로, 실시간 스펙트럼 분석기 애플리케이션을 성공적으로 구축함으로써 하드웨어, 네트워크, 드라이버, 그리고 Python 코드로 이어지는 전체 신호 처리 체인이 완벽하게 작동함을 검증했다.

이 과정을 통해 사용자는 ANTSDR E200과 pyadi-iio를 활용한 기본적인 수신 애플리케이션 개발의 토대를 마련했다. 여기서부터 다음과 같은 다양한 방향으로 탐구를 확장할 수 있다.

- 신호 송신 **(tx)**: 본 가이드는 수신에 초점을 맞췄지만, ANTSDR은 송신 기능도 완벽하게 지원한다. sdr.tx() 메소드를 사용하여 디지털 방식으로 생성된 신호를 무선으로 송출할 수 있다. pyadi-iio 공식 문서와 예제 코드에서 관련 사용법을 찾을 수 있다.[29]
- **MIMO(Multiple-Input Multiple-Output)** 운용: ANTSDR E200은 2개의 송신 채널과 2개의 수신 채널을 갖춘 2x2 MIMO 장치이다. pyadi-iio는 두 채널을 동시에 활성화하고 제어하는 기능을 지원하므로, 빔포밍, 다이버시티, 공간 다중화와 같은 고급 통신 기술을 실험할 수 있다.[2]
- 고급 프레임워크와의 통합: 더 높은 성능이나 대규모 프로젝트와의 통합이 필요한 전문가 사용자를 위해, ANTSDR은 UHD(USRP Hardware Driver) 펌웨어 모드를 지원한다. DIP 스위치를 'SD' 모드로 변경하고 관련 펌웨어를 SD 카드에 준비하면, GNU Radio와 같은 전문 SDR 프레임워크나 srsRAN, Open5G-PHY와 같은 오픈 소스 셀룰러 스택과 완벽하게 호환된다.[2] 이는 ANTSDR을 단순한 실험용 장비를 넘어, 전문가 수준의 무선 시스템 연구 및 개발 플랫폼으로 활용할 수 있는 길을 열어준다.

결론적으로, ANTSDR E200은 PlutoSDR의 사용 편의성과 USRP의 전문적인 성능을 결합한 강력하고 유연한 SDR 플랫폼이다. 본 가이드에서 제시한 pyadi-iio 기반의 접근법은 Python을 활용하여 신속하게 프로토타입을 개발하고 무선 신호를 분석하고자 하는 엔지니어와 연구자들에게 견고한 출발점을 제공할 것이다.

참고 자료


1. Tagged: antsdr - RTL-SDR.com, 9월 20, 2025에 액세스, https://www.rtl-sdr.com/tag/antsdr/
2. AntSDR E200 - Crowd Supply, 9월 20, 2025에 액세스, https://www.crowdsupply.com/microphase-technology/antsdr-e200
3. Chaochen Wei's AntSDR E200 Aims to Combine the Best of the PlutoSDR and the USRP B200 - Hackster.io, 9월 20, 2025에 액세스, https://www.hackster.io/news/chaochen-wei-s-antsdr-e200-aims-to-combine-the-best-of-the-plutosdr-and-the-usrp-b200-f60f55736cec
4. ANTSDR E200 set to begin Crowdfunding on CrowdSupply soon - RTL-SDR.com, 9월 20, 2025에 액세스, https://www.rtl-sdr.com/antsdr-e200-set-to-begin-crowdfunding-on-crowdsupply-soon/
5. PySDR: A Guide to SDR and DSP using Python, 9월 20, 2025에 액세스, https://pysdr.org/
6. Introduction | PySDR: A Guide to SDR and DSP using Python, 9월 20, 2025에 액세스, https://pysdr.org/content/intro.html
7. MicroPhase ANTSDR E200-9363 325M~3.8GHz Software Defined Radio SDR (AD9363) pe6 | eBay, 9월 20, 2025에 액세스, https://www.ebay.com/itm/356394846201
8. Getting Started with AntSDR - Crowd Supply, 9월 20, 2025에 액세스, https://www.crowdsupply.com/microphase-technology/antsdr-e200/updates/getting-started-with-antsdr
9. PlutoSDR in Python | PySDR: A Guide to SDR and DSP using Python, 9월 20, 2025에 액세스, https://pysdr.org/content/pluto.html
10. Pyadi Iio - conda install - Anaconda.org, 9월 20, 2025에 액세스, https://anaconda.org/conda-forge/pyadi-iio
11. pyadi-iio - PyPI, 9월 20, 2025에 액세스, https://pypi.org/project/pyadi-iio/
12. Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 20, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/
13. Software radio E310-9363 ZYNQ7020 ADI Pluto Experimental Platform Antsdr, 9월 20, 2025에 액세스, https://www.sdrstore.eu/software-defined-radio/instruments/plutosdr/software-radio-e310-9363-zynq7020-adi-pluto-experimental-platform-antsdr-en/
14. Using the MicroPhase AntSDR e200 on Ubuntu 23.10 - RadioReference.com Forums, 9월 20, 2025에 액세스, https://forums.radioreference.com/threads/using-the-microphase-antsdr-e200-on-ubuntu-23-10.465766/page-2
15. Using the ANTSDR E200 on Windows with WSL Ubuntu 22.04 - YouTube, 9월 20, 2025에 액세스, https://www.youtube.com/watch?v=sYnqp-skjEo
16. Using the ANTSDR E200 on Windows with WSL Ubuntu 22.04 - Google Groups, 9월 20, 2025에 액세스, https://groups.google.com/g/sdrglut-users/c/bsM0Q4GptDI
17. MicroPhase/antsdr-fw: ANTSDR Firmware - GitHub, 9월 20, 2025에 액세스,

https://github.com/MicroPhase/antsdr-fw
18. Using The ANTSDR E200 on Ubuntu 24.04 - YouTube, 9월 20, 2025에 액세스, https://www.youtube.com/watch?v=nyC2BJI9kSA
19. MicroPhase/antsdr-pynq - GitHub, 9월 20, 2025에 액세스, https://github.com/MicroPhase/antsdr-pynq
20. Quick Start — Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 20, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/guides/quick.html
21. AD9361 Setup and Interfacing - Ian P. Roberts, 9월 20, 2025에 액세스, https://ianproberts.com/notes/ad9361setup.html
22. MicroPhase/antsdr-fw-patch: Repository of antsdr firmware make - GitHub, 9월 20, 2025에 액세스, https://github.com/MicroPhase/antsdr-fw-patch
23. Activity · MicroPhase/antsdr_uhd - GitHub, 9월 20, 2025에 액세스, https://github.com/MicroPhase/antsdr_uhd/activity
24. gr-iio: Nuances, Hidden Features, and New Stuff - GNU Radio, 9월 20, 2025에 액세스, https://www.gnuradio.org/grcon/grcon19/presentations/gr-iio_Nuances_Hidden_Features_and_New_Stuff/Travis%20Collins%20-%20gr_iio.pdf
25. ADM-Pluto-File-Transfer & FM Receiver PlutoSDR - Jay Patel, 9월 20, 2025에 액세스, https://patel999jay.github.io/ADALM-Pluto-File-Transfer/
26. libiio Direct Access — Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 20, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/libiio.html
27. aa2il/pySDR: Software Receiver written in Python - GitHub, 9월 20, 2025에 액세스, https://github.com/aa2il/pySDR
28. analogdevicesinc/pyadi-iio: Python interfaces for ADI hardware with IIO drivers (aka peyote), 9월 20, 2025에 액세스, https://github.com/analogdevicesinc/pyadi-iio
29. Examples — Analog Devices Hardware Python Interfaces 0.0.3 documentation, 9월 20, 2025에 액세스, https://pyadi-iio.readthedocs.io/en/latest/guides/examples.html
30. Connectivity — Analog Devices Hardware Python Interfaces 0.0.3 documentation, 9월 20, 2025에 액세스, https://pyadi-iio.readthedocs.io/en/latest/guides/connectivity.html
31. Examples — Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 20, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/guides/examples.html
32. Using Python To Control The Pluto Radio And Plot Data - EngineerZone Spotlight - EZ Blogs, 9월 20, 2025에 액세스, https://ez.analog.com/ez-blogs/b/engineerzone-spotlight/posts/using-python-to-control-the-pluto-radio-and-plot-data
33. Buffers — Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 20, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/buffers/index.html
34. MicroPhase ANTSDR U220 70MHz-6GHz SDR AD9361 AD9363 - SDRstore.eu, 9월 20, 2025에 액세스,

https://www.sdrstore.eu/software-defined-radio/instruments/plutosdr/microphase-antsdr-u220-sdr-demo-board-70mhz-6ghz-sdr-usb3-0-adi-ad9361-ad9363-mimo-srsran/
35. This repo contains both the uhd host driver and firmware for microphase antsdr devices. - GitHub, 9월 20, 2025에 액세스, https://github.com/MicroPhase/antsdr_uhd