Microphase ANTSDR E310 PYNQ: 종합 기술 가이드 및 튜토리얼

섹션 **1:** 플랫폼 아키텍처 및 기능

이 섹션에서는 Microphase ANTSDR E310 하드웨어 플랫폼과 PYNQ(Python on Zynq) 소프트웨어 프레임워크에 대한 심층적인 분석을 제공합니다. 각 구성 요소의 개별적인 강점과 두 기술이 결합되었을 때 발휘되는 강력한 시너지 효과를 탐구하여, 사용자가 이 플랫폼의 잠재력을 최대한 활용할 수 있는 기반을 마련합니다.

1.1. ANTSDR E310 하드웨어 심층 분석

ANTSDR E310은 단순한 SDR(Software Defined Radio) 보드를 넘어, 고성능 컴퓨팅과 유연한 RF(Radio Frequency) 기능을 결합한 독립형 임베디드 플랫폼입니다. 이 장치의 아키텍처는 처리 코어, RF 프런트엔드, 메모리, 그리고 다양한 연결 옵션이 유기적으로 통합된 구조를 가집니다.

처리 코어**: Zynq-7000 SoC**

플랫폼의 핵심은 Xilinx Zynq-7000 SoC(System-on-Chip), 구체적으로 XC7Z020CLG400 모델입니다.[1] Zynq 아키텍처의 가장 큰 특징은 단일 칩 내에 두 가지 이질적인 컴퓨팅 유닛을 통합했다는 점입니다.

- 프로세싱 시스템 **(PS, Processing System)**: 766 MHz에서 866 MHz 사이에서 작동하는 듀얼 코어 ARM Cortex-A9 프로세서가 탑재되어 있습니다.[2] 이 PS는 운영체제(Linux) 구동, 네트워크 스택 관리, 사용자 애플리케이션 실행 등 고수준의 소프트웨어 작업을 담당합니다.
- 프로그래머블 로직 **(PL, Programmable Logic)**: 7 시리즈 FPGA(Field-Programmable

Gate Array) 패브릭이 통합되어 있습니다. 85,000개의 로직 셀, 630 KB의 블록 RAM, 그리고 220개의 DSP 슬라이스를 포함하여 병렬 처리에 최적화된 하드웨어 가속을 가능하게 합니다.[1]

이러한 하이브리드 구조는 PYNQ 프레임워크가 작동하는 근간이 됩니다. 즉, Python과 같은 고수준 언어로 작성된 코드는 ARM 프로세서에서 실행되면서, 실시간 신호 처리나 고속 연산과 같이 지연 시간에 민감한 작업은 FPGA 패브릭에 구현된 맞춤형 하드웨어 회로(오버레이)에 오프로드하여 처리 효율을 극대화합니다.

RF 프런트엔드**: Analog Devices** 트랜시버

ANTSDR E310은 Analog Devices의 고성능 RF 트랜시버를 채택하여 넓은 주파수 범위와 대역폭을 지원합니다. 사용자는 두 가지 옵션 중 하나를 선택할 수 있습니다 [1]:

- **AD9361**: 70 MHz부터 6 GHz까지의 광범위한 주파수 범위를 커버하며, 최대 56 MHz의 순간 대역폭을 제공합니다.
- **AD9363**: 325 MHz부터 3.8 GHz까지의 주파수 범위를 지원하며, 최대 20 MHz의 대역폭을 가집니다.

두 트랜시버 모두 2개의 송신 채널과 2개의 수신 채널(2x2 MIMO)을 지원하여 다중 안테나 시스템을 구현할 수 있습니다.[1] 또한, 이 플랫폼은 송신 및 수신 경로에 RF 필터 뱅크를 내장하고 있습니다. 이 필터들은 선택도를 향상시켜 대역 외 신호의 간섭을 줄이고, 송신 시 고조파(harmonics)를 억제하여 신호 품질을 개선하는 중요한 역할을 합니다.[3]

메모리 및 스토리지

ANTSDR E310은 유사 플랫폼 대비 월등한 메모리 용량을 자랑합니다. 512MB DDR3 칩 두 개로 구성된 총 1 GByte의 RAM을 탑재하여, ARM 프로세서에서 실행되는 복잡한 애플리케이션이나 대용량 데이터 버퍼링을 원활하게 지원합니다.[1] 부팅 펌웨어나 핵심 설정 저장을 위해 32MB의 QSPI 플래시 메모리도 내장되어 있습니다.[2] 주 운영체제와 사용자 데이터는 외부 microSD 카드를 통해 저장 및 로드됩니다.

연결성 및 주변 장치

이 플랫폼은 현장 배치 및 독립형 운영을 염두에 두고 설계되었으며, 풍부한 I/O 포트를 제공합니다 [1]:

- 기가비트 이더넷: 고속 데이터 스트리밍 및 원격 제어를 위한 필수적인 인터페이스입니다.
- **USB 2.0 OTG**: 키보드, 마우스, 저장 장치 등 다양한 USB 주변 장치를 연결할 수 있습니다.
- **USB-UART**: 시리얼 콘솔 접근을 통해 부팅 과정 모니터링 및 시스템 디버깅을 수행합니다.
- 통합 **USB-JTAG**: 별도의 JTAG 프로그래머 없이 FPGA 개발 및 디버깅이 가능합니다.
- 확장 포트: GPS 수신기를 연결하여 정밀한 시간 동기화(PPS) 및 위치 정보를 활용할 수 있습니다.

이러한 하드웨어 사양은 ANTSDR E310이 단순한 SDR 주변기기가 아니라, 그 자체로 완결된 고성능 임베디드 신호 처리 시스템임을 명확히 보여줍니다.

이 플랫폼의 설계 철학은 널리 사용되는 ADALM-PLUTO SDR을 계승하면서 그 한계를 극복하는 데 있습니다. 여러 자료에서 ANTSDR E310은 PLUTO의 "영감을 받은" 또는 PLUTO 사용자가 "더 적합하게" 사용할 수 있는 장비로 소개됩니다.[1] 이는 단순한 모방이 아닌, PLUTO 사용자들이 겪었던 성능 병목 현상을 해결하기 위한 전략적 업그레이드임을 시사합니다. 예를 들어, 더 큰 Zynq-7020 FPGA(PLUTO의 7010 대비)는 더 복잡한 하드웨어 가속 로직을 구현할 수 있게 하고 [1], 두 배로 늘어난 1GB RAM은 더 큰 온보드 애플리케이션을 지원하며 [1], PLUTO에는 없는 기가비트 이더넷 포트는 고대역폭 데이터 스트리밍에 필수적입니다.[2] 따라서 이 장비의 핵심 사용자는 PLUTO로 SDR에 입문했지만,

openwifi나 맞춤형 MIMO 시스템과 같은 고급 애플리케이션을 위해 더 많은 연산 능력, 메모리, I/O 대역폭이 필요해진 개발자라고 볼 수 있습니다.

항목	사양	참고 자료
데이터 처리 장치		
SoC	Xilinx Zynq-7000 XC7Z020CLG400	[1]
프로세서	듀얼 코어 ARM Cortex-A9 (766 MHz)	[1]
FPGA	85k 로직 셀, 220 DSP 슬라이스	[1]
RF 사양		

RF IC (선택)	Analog Devices AD9361 / AD9363	[1]
주파수 범위	70 MHz – 6 GHz (AD9361) / 325 MHz – 3.8 GHz (AD9363)	[1]
신호 대역폭	200 kHz – 56 MHz (AD9361) / 200 kHz – 20 MHz (AD9363)	[1]
RF 채널	2 송신, 2 수신 (2x2 MIMO)	[1]
메모리 및 스토리지		
RAM	1 GByte DDR3 (2 x 512MB)	[1]
플래시	32 MByte QSPI	[2]
인터페이스		
이더넷	기가비트 이더넷 (10/100/1000 Mbps)	[1]
USB	USB 2.0 OTG, USB-UART, 통합 USB-JTAG	[1]
스토리지	MicroSD 카드 슬롯	[2]
확장	GPS 클럭 동기화 입력 지원	[2]
기타		
클럭	0.5ppm TCXO (고정밀 VCTCXO 옵션)	[1]
폼 팩터	103 mm x 66 mm	[1]

전원	USB 전원	[1]

1.2. PYNQ 프레임워크: Python과 프로그래머블 로직의 만남

PYNQ는 AMD/Xilinx에서 주도하는 오픈소스 프로젝트로, Zynq SoC의 강력한 성능을 소프트웨어 개발자들이 보다 쉽게 활용할 수 있도록 설계되었습니다.[7] 전통적인 FPGA 개발 방식이 Verilog나 VHDL과 같은 하드웨어 기술 언어(HDL)에 대한 깊은 이해를 요구했던 것과 달리, PYNQ는 Python이라는 고수준 언어를 통해 FPGA의 프로그래머블 로직에 접근하는 혁신적인 방법을 제시합니다.

핵심 아키텍처

PYNQ의 아키텍처는 Zynq SoC의 ARM 프로세서에서 실행되는 완전한 Linux 배포판을 기반으로 합니다. 이 Linux 환경 위에서 Jupyter Notebook 웹 서버가 구동됩니다.[4] 사용자는 웹 브라우저를 통해 보드의 IP 주소로 접속하기만 하면, 대화형 Python 코딩 환경인 Jupyter Notebook을 통해 보드를 제어하고 프로그래밍할 수 있습니다. 모든 연산과 제어는 보드 자체에서 이루어지며, 사용자의 PC는 단지 웹 인터페이스를 표시하는 '씬 클라이언트(thin client)' 역할만 수행합니다.

"오버레이(Overlay)" 패러다임

PYNQ의 가장 핵심적인 개념은 '오버레이'입니다.[9] 오버레이는 특정 기능을 수행하도록 미리 컴파일된 FPGA 디자인(비트스트림)으로, 재사용 가능한 '하드웨어 라이브러리'에 비유할 수 있습니다. 소프트웨어 개발자가 특정 기능을 위해 소프트웨어 라이브러리를

import하여 사용하듯, PYNQ 환경에서는 필요한 오버레이를 런타임에 FPGA로 로드하여 사용할 수 있습니다.

예를 들어, FFT(고속 푸리에 변환) 가속 오버레이, 머신러닝 추론 오버레이, 또는 비디오 처리 오버레이 등을 동적으로 FPGA에 로드할 수 있습니다. 일단 로드된 오버레이의 기능들은 Python API를 통해 마치 일반적인 Python 객체처럼 접근하고 제어할 수 있습니다. 이 패러다임은 하드웨어 설계 전문가가 한 번 잘 만들어 놓은 고성능 하드웨어 블록을, 수많은 소프트웨어 개발자들이 전문 지식 없이도 쉽게 재사용할 수 있게 해줍니다. 이는 "한 번 만들고,

여러 번 재사용한다(build once, re-use many times)"는 소프트웨어 공학의 모범 사례를 하드웨어 개발에 적용한 것입니다.[9]

PYNQ 이미지를 사용하기로 한 결정은 ANTSDR E310의 운영 모델을 근본적으로 변화시킵니다. 이 장비는 더 이상 호스트 PC에 데이터를 공급하는 단순한 '데이터 파이프'가 아니라, 그 자체로 완결된 Python 프로그래밍이 가능한 신호 처리 장비가 됩니다. 기존의 다른 펌웨어들, 예를 들어 PLUTO 호환 펌웨어나 UHD 펌웨어는 주로 장비를 호스트 컴퓨터가 제어하는 네트워크 또는 USB 주변 장치로 설정합니다.[11] 반면 PYNQ는 주된 제어 및 처리 환경을 장비 내부의 ARM 코어에 둡니다.[9] 이러한 아키텍처의 전환은 고속 신호 처리가 FPGA와 ARM 코어 상에서 직접 수행될 수 있음을 의미하며, 이는 호스트로의 대역폭 높은 데이터 전송 필요성을 극적으로 줄이고 전체 시스템의 지연 시간을 낮춥니다. 이는 "현장 배치" 및 "독립형" 애플리케이션에 매우 중요한 이점입니다.[3]

1.3. ANTSDR E310 PYNQ 이미지: 개발자 중심의 환경

ANTSDR E310을 위해 제공되는 PYNQ 이미지는 강력한 개발 환경을 제공하지만, 사용자의 초기 설정이 일부 필요한 '개발자 키트'의 성격을 띱니다.

이미지 소스 및 특징

이 PYNQ 이미지는 공식 기업 포털이 아닌, 커뮤니티가 관리하는 GitHub 저장소(MicroPhase/antsdr-pynq)를 통해 배포됩니다.[14] 이는 활발한 개발과 빠른 업데이트가 가능하다는 장점이 있지만, 동시에 문서화나 지원이 공식 제품만큼 체계적이지 않을 수 있음을 시사합니다.

필수적인 초기 설정

제공된 문서에 따르면, 사용자는 PYNQ 이미지를 부팅한 후 반드시 장치를 인터넷에 연결하고 몇 가지 핵심 라이브러리를 소스 코드로부터 직접 빌드 및 설치해야 합니다.[14]

- **libiio**: Analog Devices의 RF 칩과 같은 Linux IIO(Industrial I/O) 장치와 통신하기 위한 핵심 C 라이브러리입니다.
- **pyadi-iio**: libiio의 기능을 Python 환경에서 사용할 수 있도록 해주는 Python 바인딩(래퍼)입니다.

이 과정은 표준 Linux 패키지 관리(sudo apt update)와 소스 코드 컴파일(git clone, cmake, make)을 포함하며, 이는 이 이미지가 Debian/Ubuntu 기반의 파일 시스템을 사용하고 있음을 나타냅니다.[14]

이 PYNQ 이미지는 상자에서 꺼내자마자 모든 기능이 완벽하게 작동하는 완제품이 아닙니다. 오히려 사용자가 직접 핵심 시스템 구성 요소를 설치해야 하는 "조립이 일부 필요한(Some Assembly Required)" 키트에 가깝습니다. 만약 이미지가 완전히 완성된 형태였다면, libiio와 pyadi-iio 같은 필수 라이브러리들이 사전에 설치되어 있었을 것입니다. 공식 설명서에서 사용자가 직접 git 저장소를 복제하고 C 코드를 컴파일하도록 요구하는 것은 이 이미지가 개발자 중심의 환경임을 명확히 보여주는 증거입니다.[14] 특히, 초기 설정 과정에서 인터넷 연결이 필수적이라는 점은 이 장비가 "독립형" 및 "현장 배치"용으로 홍보되는 것과 다소 상충될 수 있습니다.[3] 인터넷이 없는 현장에서 처음 장비를 켜는 사용자는 RF 트랜시버를 사용할 수 없게 될 것입니다. 따라서, 이어지는 튜토리얼에서는 이 초기 설정 단계를 선택 사항이 아닌, 장비의 핵심 기능을 활성화하기 위한 필수적인 절차로 상세히 다룰 것입니다.

섹션 2: 5단계로 따라하는 첫 신호 수신 가이드

이 섹션은 비어있는 microSD 카드부터 시작하여 실시간 RF 스펙트럼을 시각화하기까지의 전 과정을 명령어 단위로 상세하게 안내하는 튜토리얼입니다. 일반적인 PYNQ 보드 설정 가이드와 ANTSDR E310 고유의 지침을 통합하여, 누구나 쉽게 따라 할 수 있도록 구성되었습니다.[14]

1단계: 환경 준비 및 초기 부팅

이 단계에서는 PYNQ 이미지를 다운로드하여 SD 카드에 설치하고, 하드웨어를 올바르게 연결하여 첫 부팅을 준비합니다.

상세 절차

1. 이미지 다운로드: antsdr-pynq GitHub 저장소에서 제공하는 링크를 통해 PYNQ 이미지를 다운로드합니다. 해외 사용자는 pCloud 링크를 사용하는 것이 편리합니다.[14]
   - pCloud 다운로드 링크: https://e.pcloud.link/publink/show?code=kZOI7TZsFHJaEPgb1jBTQRKHnclPjjPIHyX
2. SD 카드 플래싱: 다운로드한 .img 압축 파일을 해제하고, BalenaEtcher 또는 Raspberry Pi

Imager와 같은 도구를 사용하여 microSD 카드(최소 16GB 권장)에 이미지를 기록합니다.[1] 이 과정은 SD 카드의 기존 내용을 모두 삭제하므로 주의해야 합니다.

3. 하드웨어 연결: 다음 순서에 따라 하드웨어를 정확하게 연결합니다.
    - 플래싱이 완료된 microSD 카드를 ANTSDR E310 보드 하단의 슬롯에 삽입합니다.[15]
    - USB-UART 케이블을 보드의 해당 포트와 PC에 연결합니다. 이는 시리얼 콘솔 접근에 사용됩니다.[2]
    - 기가비트 이더넷 포트를 인터넷 접속이 가능한 유무선 공유기(라우터)에 연결합니다. 이 연결은 3단계에서 라이브러리를 설치하는 데 필수적입니다.[14]
    - 제공된 안테나 중 하나를 RX 포트(예: RX1)에 연결합니다.[16]
    - USB 케이블을 사용하여 보드에 전원을 공급합니다.[1]
4. 부팅 모드 설정: 보드의 이더넷 포트 아래에 위치한 부팅 모드 DIP 스위치를 찾아 'SD' 모드로 설정합니다. 이를 통해 보드가 QSPI 플래시가 아닌 microSD 카드로 부팅됩니다.[12]
5. 전원 인가 및 **LED** 관찰: 전원을 켜고 보드의 LED 상태를 관찰합니다. 부팅이 진행됨에 따라 여러 LED가 켜지고 깜박이며, 시스템이 완전히 준비되면 특정 LED 패턴(예: 파란색 LED 켜짐)이 나타납니다. 이 상태는 시스템에 접속할 준비가 되었음을 의미합니다.[15]

2단계**:** 네트워크 및 터미널 연결 설정

부팅이 완료된 보드에 접속하여 제어할 수 있는 통신 채널을 설정합니다. 시리얼 콘솔과 네트워크(SSH, Jupyter) 두 가지 방법을 모두 설정하여 안정적인 작업 환경을 구축합니다.

상세 절차

1. 시리얼 콘솔 연결: PC에서 장치 관리자(Windows)나 ls /dev/tty* 명령어(Linux)를 통해 USB-UART 브릿지에 할당된 포트 번호(예: COM5) 또는 장치 파일(예: /dev/ttyUSB0)을 확인합니다.[16] PuTTY, MobaXterm, 또는
   screen과 같은 터미널 에뮬레이터를 사용하여 다음 설정으로 연결합니다 [15]:
    - 전송 속도 **(Baud Rate)**: 115200
    - 데이터 비트: 8
    - 패리티: 없음 (None)
    - 정지 비트: 1
      연결에 성공하면 부팅 메시지와 함께 로그인 프롬프트가 나타납니다.
2. 네트워크 연결 **(DHCP)**: 보드를 공유기에 연결했기 때문에, DHCP 서버로부터 자동으로 IP 주소를 할당받습니다. 이 IP 주소는 공유기의 관리 페이지나, 시리얼 콘솔에 로그인하여 ifconfig 명령어를 실행하여 확인할 수 있습니다.[16]
3. **SSH** 접속: 할당된 IP 주소를 확인한 후, PC의 터미널에서 SSH를 통해 보드에 원격으로

접속합니다. Pluto 기반 시스템의 기본 로그인 정보는 보통 root 계정과 analog 비밀번호입니다.[16]

```
Bash
ssh root@<보드의_IP_주소>
```

4. **Jupyter Notebook** 접속: 마지막으로, PC의 웹 브라우저를 열고 주소창에 http://<보드의_IP_주소>:9090을 입력합니다. PYNQ 환경의 기본 로그인 정보는 사용자 이름 xilinx와 비밀번호 xilinx입니다.[15] Jupyter Notebook의 대시보드 화면이 나타나면 이 단계가 성공적으로 완료된 것입니다.

3단계**:** 시스템 초기화 및 핵심 라이브러리 설치

이 단계는 ANTSDR E310의 RF 기능을 PYNQ 환경에서 사용하기 위한 가장 중요한 과정입니다. 기본 PYNQ 이미지에는 AD9361 RF 칩을 제어하는 데 필요한 특정 라이브러리가 포함되어 있지 않으므로, 이를 수동으로 설치해야 합니다.

상세 절차 **(SSH** 또는 시리얼 터미널에서 실행**)**

1. 패키지 목록 업데이트: apt 패키지 관리자가 최신 소프트웨어 정보를 참조할 수 있도록 목록을 동기화합니다.

```
Bash
sudo apt update
```

2. 빌드 의존성 설치: libiio와 관련 라이브러리를 소스 코드로부터 컴파일하는 데 필요한 도구와 라이브러리들을 설치합니다.[14]

```
Bash
sudo apt install libxml2 libxml2-dev bison flex cmake git libaio-dev libboost-all-dev
libusb-1.0-0-dev libavahi-common-dev libavahi-client-dev
```

3. **libiio** 클론 및 빌드: Analog Devices IIO 장치와 통신하기 위한 핵심 C 라이브러리인 libiio를 GitHub에서 복제하여 빌드하고 시스템에 설치합니다. Python 바인딩 옵션을 활성화하는 것이 중요합니다.[14]

```
Bash
git clone -b v0.24 https://github.com/analogdevicesinc/libiio.git
cd libiio
mkdir build && cd build
cmake.. -DPYTHON_BINDINGS=ON
```

```
make
sudo make install
cd../..
```

4. **libad9361-iio** 클론 및 빌드: AD9361 칩에 특화된 헬퍼 라이브러리를 동일한 방식으로 빌드하고 설치합니다.[14]
   ```
   Bash
   git clone https://github.com/analogdevicesinc/libad9361-iio.git
   cd libad9361-iio
   mkdir build && cd build
   cmake..
   make
   sudo make install
   cd../..
   ```

5. **Python** 바인딩 설치: 마지막으로, 컴파일된 C 라이브러리들의 기능을 Python/Jupyter 환경에서 사용할 수 있도록 pip를 통해 Python 패키지를 설치합니다.[14]
   ```
   Bash
   sudo pip3 install pyadi-iio
   ```

   이 모든 과정이 오류 없이 완료되면, 시스템은 RF 하드웨어를 제어할 준비를 마칩니다.

4단계: SDR을 위한 "Hello, World!" - RF 프런트엔드 검증

이제 Jupyter Notebook 환경으로 이동하여, 3단계에서 설치한 라이브러리가 정상적으로 작동하고 하드웨어와 통신할 수 있는지 확인합니다.

상세 절차 (Jupyter Notebook 코드)

1. Jupyter 대시보드에서 'New' -> 'Python 3'를 선택하여 새 노트북을 생성합니다.
2. 첫 번째 셀에 다음 Python 코드를 입력하고 실행(Shift+Enter)합니다.
   ```
   Python
   import adi

   # 로컬 장치를 네트워크 컨텍스트로 지정하여 SDR 객체 생성
   sdr = adi.ad9361(uri="ip:127.0.0.1")
   ```

```
# 일부 장치 속성을 출력하여 통신 확인
print(f"수신 LO 주파수: {sdr.rx_lo} Hz")
print(f"샘플링 속도: {sdr.sample_rate} SPS")
print(f"수신 이득 (채널 0, 'manual'): {sdr.rx_gain_chan0} dB")

# 파라미터 변경 테스트
sdr.rx_lo = 915000000 # 수신 LO 주파수를 915 MHz로 설정
print(f"새로운 수신 LO 주파수: {sdr.rx_lo} Hz")
```

3. 코드 설명:
   - import adi: pyadi-iio 라이브러리를 가져옵니다.
   - adi.ad9361(uri="ip:127.0.0.1"): 로컬호스트(자기 자신)의 IIO 데몬에 접속하여 AD9361 장치를 제어하는 객체(sdr)를 생성합니다.
   - sdr.rx_lo, sdr.sample_rate 등: 이 객체의 속성에 접근하는 것만으로 하드웨어의 레지스터 값을 읽거나 쓸 수 있습니다.

이 코드가 오류 없이 실행되고 현재 설정된 하드웨어 값들이 출력되면, 소프트웨어와 하드웨어 간의 통신이 성공적으로 이루어진 것입니다.

5단계**:** 스펙트럼 캡처 및 시각화

튜토리얼의 마지막 단계로, 실제 RF 신호를 수신하여 그 스펙트럼을 그래프로 그려보는 과정입니다. 이를 통해 사용자는 자신의 SDR 플랫폼이 작동하는 것을 시각적으로 확인할 수 있습니다.

상세 절차 **(Jupyter Notebook** 코드**)**

1. 새로운 셀에 수신기 설정을 구성하고 데이터를 캡처하는 코드를 입력합니다. 여기서는 주변에서 쉽게 찾을 수 있는 FM 라디오 방송 대역을 예시로 사용합니다.

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

# FM 라디오 방송 대역으로 수신기 설정
sdr.rx_lo = 98500000 # 98.5 MHz
sdr.sample_rate = 2400000 # 2.4 MSPS
sdr.rx_rf_bandwidth = 200000 # 200 kHz
sdr.rx_gain_chan0 = 60 # 수신 이득 설정
```

```
sdr.rx_buffer_size = 16384 # 캡처할 샘플 수

# 첫 번째 채널에서 IQ 샘플 캡처
samples = sdr.rx()

print(f"{len(samples)}개의 샘플을 캡처했습니다.")
```

2. 다음 셀에 캡처한 데이터를 처리하고 스펙트럼을 그리는 코드를 입력합니다.
   Python
   ```
   # 전력 스펙트럼 밀도(PSD) 계산 및 플로팅
   plt.figure(figsize=(10, 6))
   plt.psd(samples, NFFT=1024, Fs=sdr.sample_rate/1e6, Fc=sdr.rx_lo/1e6)
   plt.xlabel("주파수 (MHz)")
   plt.ylabel("상대 전력 (dB)")
   plt.title("ANTSDR E310에서 수신한 실시간 RF 스펙트럼")
   plt.grid(True)
   plt.show()
   ```

3. 코드 설명:
   - sdr.rx(): 이 함수를 호출하면 설정된 버퍼 크기만큼의 IQ 샘플 데이터가 하드웨어로부터 수집되어 numpy 배열 형태로 반환됩니다.
   - plt.psd(): matplotlib 라이브러리의 함수로, 입력된 샘플 데이터에 대해 FFT를 수행하고 그 결과를 주파수 대역별 전력 그래프로 시각화해줍니다. Fs는 샘플링 속도, Fc는 중심 주파수를 의미합니다.

이 코드를 실행하면 노트북 출력 창에 그래프가 나타납니다. 만약 주변에 강력한 FM 라디오 방송국이 있다면, 설정한 중심 주파수(98.5 MHz) 근처에서 뚜렷한 신호 피크를 관찰할 수 있을 것입니다. 이것으로 ANTSDR E310과 PYNQ 환경을 이용한 첫 신호 수신 및 분석이 성공적으로 완료되었습니다.

섹션 **3:** 플랫폼 활용도 향상 방안

초기 설정을 성공적으로 마친 사용자가 플랫폼을 더욱 심도 있게 활용할 수 있도록, 일반적인 문제 해결 방안과 고급 개발을 위한 경로를 제시합니다.

3.1. 문제 해결 가이드

개발자 중심의 플랫폼에서는 다양한 문제가 발생할 수 있습니다. 이 섹션에서는 포럼이나 GitHub 이슈에서 자주 보고되는 문제들과 그 해결책을 정리하여, 사용자가 겪을 수 있는 어려움을 사전에 대비하고 신속하게 해결할 수 있도록 돕습니다.[5]

증상 / 오류 메시지	예상 원인	해결 방안
Jupyter에 접속 불가 (http://...:9090)	네트워크 설정 오류, 보드 부팅 미완료	1. 이더넷 케이블 연결 상태 확인. 2. PC 터미널에서 보드 IP로 ping 테스트. 3. 보드의 부팅 완료 LED 상태 확인. 4. 시리얼 콘솔로 접속하여 ifconfig 명령어로 IP 주소 재확인.[24]
git clone 실패 (Permission denied (publickey))	GitHub 계정의 SSH 키 문제	GitHub 계정에 PC의 SSH 공개키를 등록하거나, https 프로토콜을 사용하도록 저장소 주소를 변경하여 복제.
libiio 빌드 중 make 실패	의존성 패키지 누락 또는 툴체인 문제	1. 튜토리얼 3단계의 모든 apt install 명령어가 성공했는지 재확인. 2. sudo apt update && sudo apt upgrade로 시스템 패키지 업데이트. 3. SD 카드의 남은 용량 확인.
import adi 실패 (ModuleNotFoundError)	Python 바인딩 설치 오류	1. sudo pip3 install pyadi-iio 명령어를 다시 실행. 2. 시스템의 기본 Python 3 인터프리터가 사용되고 있는지 확인.
adi.ad9361() 실행 시 시간 초과 또는 실패	IIO 서비스 미작동 또는 하드웨어 인식 문제	1. 보드를 재부팅. 2. 시리얼 콘솔에서 dmesg 명령어로 AD9361 드라이버 관련 오류 메시지 확인. 3. uri가 로컬 접속의 경우 "ip:127.0.0.1"로

		정확히 입력되었는지 확인.
수신 스펙트럼이 매우 약하거나 노이즈가 심함	낮은 이득, 필터 설정 오류, 안테나 문제	1. sdr.rx_gain_chan0 값을 더 높게 설정 (예: 70). 2. 안테나가 올바른 RX 포트에 단단히 연결되었는지 확인. 3. E310은 PLUTO와 RF 프런트엔드 구성이 다르므로, 최적의 성능을 위해 맞춤형 이득 테이블이 필요할 수 있음에 유의.[5]
PYNQ 오버레이 로드 시 보드가 멈춤	전원 부족 또는 비트스트림/하드웨어 충돌	1. USB 전원 사용 시, 외부 전원 어댑터 사용을 고려. 2. 시리얼 콘솔을 연결한 상태에서 멈춤 현상 발생 시 커널 패닉 메시지가 출력되는지 확인.[23] 3. 가장 단순한 오버레이부터 시작하여 점진적으로 복잡도를 높여가며 테스트.

3.2. 추가 개발을 위한 경로

기본적인 신호 수신을 넘어 더 복잡하고 흥미로운 프로젝트를 진행할 수 있는 몇 가지 경로를 소개합니다.

PYNQ 오버레이 탐색 및 활용

PYNQ의 진정한 힘은 하드웨어 가속에 있습니다. 비록 ANTSDR PYNQ 이미지에 사전 빌드된 예제가 많지 않을 수 있지만 [14], 일반적인 PYNQ 커뮤니티 예제를 통해 오버레이 활용법을 배울 수 있습니다.[26] 기본적인 워크플로우는 다음과 같습니다.

1. FPGA 디자인(비트스트림, .bit 파일)과 하드웨어 핸드오프 파일(.hwh)을 보드로 복사합니다.
2. Jupyter Notebook에서 Overlay 클래스를 사용하여 비트스트림을 FPGA에 로드합니다: ol = Overlay('design.bit').

3. 로드된 오버레이 객체(ol)를 통해 내부에 포함된 IP 블록(예: DMA, GPIO)에 Python 속성처럼 접근하여 제어합니다.

GNU Radio와의 연동

PYNQ가 온보드 처리에 강점이 있지만, 여전히 ANTSDR E310을 호스트 PC 기반의 툴과 연동하여 사용할 수 있습니다. GNU Radio는 강력한 SDR 프레임워크이며, gr-iio 블록셋을 사용하면 네트워크를 통해 ANTSDR E310에 접속하여 신호 소스 또는 싱크로 활용할 수 있습니다. 이는 기존에 GNU Radio에 익숙한 사용자들이 PYNQ 환경을 거치지 않고도 E310의 하드웨어를 활용할 수 있는 좋은 방법입니다. MicroPhase는 ANTSDR을 위한 GNU Radio 데모도 제공하고 있습니다.[28]

대체 펌웨어 탐색

PYNQ 외에도 ANTSDR E310은 다른 펌웨어들을 지원합니다. 사용자의 경험과 프로젝트 요구사항에 따라 다른 환경을 선택할 수 있습니다.

- **PLUTO** 펌웨어: ADALM-PLUTO와 동일한 방식으로 장치를 사용하고자 할 때 유용합니다. libiio와 pyadi-iio를 통해 제어되며, 수많은 PLUTO용 튜토리얼과 소프트웨어를 활용할 수 있습니다.
- **UHD (USRP Hardware Driver)** 펌웨어: Ettus Research의 USRP 제품군과 호환되는 환경을 제공합니다. UHD 드라이버를 사용하는 애플리케이션(예: srsRAN, OpenAirInterface)을 구동하고자 할 때 필요합니다.

이러한 대체 펌웨어들은 MicroPhase/antsdr-fw 및 MicroPhase/antsdr_uhd GitHub 저장소에서 찾을 수 있습니다.[1]

커뮤니티 및 학습 자료

지속적인 학습과 문제 해결을 위해 커뮤니티의 도움을 받는 것이 중요합니다. 다음 리소스들을 적극적으로 활용하는 것을 권장합니다.

- **PYNQ** 지원 포럼: PYNQ 프레임워크 전반에 대한 질문과 답변을 찾을 수 있는 공식 커뮤니티입니다.[29]
- **MicroPhase GitHub** 저장소: 펌웨어, 드라이버, 예제 코드 등 ANTSDR 관련 최신 정보를

얻을 수 있는 가장 중요한 소스입니다.[13]
- 온라인 튜토리얼: DragonOS와 같은 SDR 특화 운영체제에서 ANTSDR을 활용하는 다양한 비디오 튜토리얼이 존재하며, 실제 애플리케이션 예제를 통해 많은 영감을 얻을 수 있습니다.[30]

이러한 경로들을 통해 사용자는 단순한 신호 수신을 넘어, ANTSDR E310과 PYNQ가 제공하는 강력한 온보드 처리 능력을 활용하여 자신만의 복잡한 무선 통신 시스템을 설계하고 구현하는 단계로 나아갈 수 있을 것입니다.

참고 자료

1. MicroPhase ANTSDR E310 AD9363 SDR Development Board for ADI Pluto Communication- | eBay, 9월 23, 2025에 액세스, https://www.ebay.com/itm/126254547389
2. MicroPhase ANTSDR E310 AD9363 SDR Development Board for ADI Pluto Communications | eBay, 9월 23, 2025에 액세스, https://www.ebay.com/itm/355222519305
3. USRP E310 Datasheet - Ettus Research, 9월 23, 2025에 액세스, https://www.ettus.com/wp-content/uploads/2019/01/USRP_E310_Datasheet.pdf
4. PYNQ-Z1 Reference Manual - Digilent, 9월 23, 2025에 액세스, https://digilent.com/reference/programmable-logic/pynq-z1/reference-manual
5. ANTSDR E310 Features - Satsagen - Groups.io, 9월 23, 2025에 액세스, https://groups.io/g/satsagen/topic/antsdr_e310_features/96870466
6. MicroPhase ANTSDR E310 Software Defined Radio Demo Board transceiver ZYNQ 7000 SoC ADI AD9361 SDR transmitter and receiver - AliExpress, 9월 23, 2025에 액세스, https://www.aliexpress.com/item/1005003181244737.html
7. PYNQ | Python Productivity for AMD Adaptive Computing platforms, 9월 23, 2025에 액세스, http://www.pynq.io/
8. Xilinx/PYNQ: Python Productivity for ZYNQ - GitHub, 9월 23, 2025에 액세스, https://github.com/Xilinx/PYNQ
9. PYNQ Introduction — Python productivity for Zynq (Pynq), 9월 23, 2025에 액세스, https://pynq.readthedocs.io/
10. Python productivity for Zynq (Pynq) Documentation - Read the Docs, 9월 23, 2025에 액세스, https://buildmedia.readthedocs.org/media/pdf/pynq/v2.4/pynq.pdf
11. sunleechina/antsdr-fw - Gitee, 9월 23, 2025에 액세스, https://gitee.com/sunleechina/antsdr-fw
12. Getting Started with AntSDR - Crowd Supply, 9월 23, 2025에 액세스, https://www.crowdsupply.com/microphase-technology/antsdr-e200/updates/getting-started-with-antsdr
13. This repo contains both the uhd host driver and firmware for microphase antsdr devices. - GitHub, 9월 23, 2025에 액세스, https://github.com/MicroPhase/antsdr_uhd
14. MicroPhase/antsdr-pynq: pynq framework for antsdr - GitHub, 9월 23, 2025에 액세스, https://github.com/MicroPhase/antsdr-pynq

15. PYNQ-Z2 Setup Guide — Python productivity for Zynq (Pynq), 9월 23, 2025에 액세스, https://pynq.readthedocs.io/en/v2.7.0/getting_started/pynq_z2_setup.html
16. ANTSDR Unpacking and Test Rev. 1.1, 9월 23, 2025에 액세스, https://down.hgeek.com/download/93159/93159-E310-AD9361-Rev-1-1-English-Manual.pdf
17. Software radio E310-9363 ZYNQ7020 ADI Pluto Experimental Platform Antsdr, 9월 23, 2025에 액세스, https://www.sdrstore.eu/software-defined-radio/instruments/plutosdr/software-radio-e310-9363-zynq7020-adi-pluto-experimental-platform-antsdr-en/
18. Getting Started Guide USRP™ E300 Series Devices - Ettus Research, 9월 23, 2025에 액세스, https://files.ettus.com/e3xx_resources/E310%20Getting%20Started%20Guide.pdf
19. PYNQ Edition! Building PYNQ Images - Hackster.io, 9월 23, 2025에 액세스, https://www.hackster.io/news/pynq-edition-building-pynq-images-4f0ec30e172f
20. antsdr : Problem with SRSRAN4g and the timestamp solution · Issue #1500 · srsran/srsRAN_4G - GitHub, 9월 23, 2025에 액세스, https://github.com/srsran/srsran_4g/issues/1500
21. Sd image build error Pynq 3.0.1 - Support, 9월 23, 2025에 액세스, https://discuss.pynq.io/t/sd-image-build-error-pynq-3-0-1/6533
22. Low accuracy on FPGA - Support - PYNQ, 9월 23, 2025에 액세스, https://discuss.pynq.io/t/low-accuracy-on-fpga/4021
23. PYNQ board connection stops as soon as I load the bitstream through the overlay on Jupyter : r/FPGA - Reddit, 9월 23, 2025에 액세스, https://www.reddit.com/r/FPGA/comments/1jowngp/pynq_board_connection_stops_as_soon_as_i_load_the/
24. Pynq Z2 ethernet connection problem : r/FPGA - Reddit, 9월 23, 2025에 액세스, https://www.reddit.com/r/FPGA/comments/1ct2hxx/pynq_z2_ethernet_connection_problem/
25. Pynq-z2 board crashing/freezing - Support, 9월 23, 2025에 액세스, https://discuss.pynq.io/t/pynq-z2-board-crashing-freezing/6383
26. PYNQ community, 9월 23, 2025에 액세스, http://www.pynq.io/community.html
27. Tutorial: Creating a hardware design for PYNQ - Learn, 9월 23, 2025에 액세스, https://discuss.pynq.io/t/tutorial-creating-a-hardware-design-for-pynq/145
28. some gnu radio demo for antsdr - GitHub, 9월 23, 2025에 액세스, https://github.com/MicroPhase/gnu-radio-demo
29. Latest Learn topics - PYNQ, 9월 23, 2025에 액세스, https://discuss.pynq.io/c/tutorials-workshops/15?page=2
30. DragonOS Focal w/ ANTSDR E310 + DragonSDR Firmware (SigDigger, SDRAngel, SDR++, QRadioLink, GQRX) - YouTube, 9월 23, 2025에 액세스, https://www.youtube.com/watch?v=L3Rc640FMzw
31. ANTSDR - YouTube, 9월 23, 2025에 액세스, https://www.youtube.com/playlist?list=PLXjRrYsXOd9fcqqFYkB1Sq4eNIj2XWGtO