Keysight SG/SA 및 **PyVISA**를 활용한 **ANTSDR RF** 자동화 테스트 튜토리얼 가이드

제**1**장**.** 자동화 테스트 환경 개요 및 설정 **(Introduction to the Automated Test Environment)**

1.1. ANTSDR 기반 **RF** 자동화 테스트의 필요성 및 목표

ANTSDR(AntSDR E200/E310 등) 플랫폼은 Analog Devices의 AD936x 칩셋과 Xilinx ZYNQ를 기반으로 설계된 유연한 소프트웨어 정의 라디오(SDR) 장치입니다.[1] 이 플랫폼은 무선 통신 프로토타이핑, 연구 개발 및 교육 분야에서 폭넓게 사용됩니다. 이러한 유연한 장치에서 생성되는 RF 신호의 정확한 특성화는 설계 검증의 핵심 단계입니다. 높은 정밀도와 신뢰성을 보장하기 위해, 상업용 기성품(COTS) 고성능 계측기인 Keysight Technologies의 신호 발생기(SG) 및 스펙트럼 분석기(SA)를 활용한 정밀 측정이 필수적입니다.[3]

본 보고서의 주된 목표는 Python 프로그래밍 언어를 마스터 제어 플랫폼으로 사용하여, Keysight의 정밀한 측정 능력(SA의 스펙트럼 순도 분석, SG의 깨끗한 신호 생성)과 ANTSDR의 유연한 송수신 능력(DUT, Device Under Test)을 결합하여 반복 가능하고 확장 가능한 RF 자동화 테스트 프레임워크를 구축하는 방법을 제시하는 것입니다. Keysight 장비는 표준 SCPI(Standard Commands for Programmable Instruments) 프로토콜을 사용하며 PyVISA 라이브러리를 통해 제어됩니다.[5] 반면, ANTSDR과 같은 SDR 장치는 ADI(Analog Devices)의 IIO(Industrial I/O) 하위 시스템을 기반으로 하며, Python 환경에서는 pyadi-iio라는 전용 라이브러리를 통해 제어됩니다.[7] 성공적인 자동화는 이처럼 이질적인 두 통신 프로토콜을 Python 마스터 스크립트 내에서 동기화하고 조율하는 능력에 달려 있습니다.

1.2. 시스템 구성 요소**: Keysight SG/SA, ANTSDR (DUT), Python/PyVISA/pyadi-iio**

자동화 테스트 시스템은 크게 세 가지 주요 구성 요소로 나뉩니다.

1.2.1. Keysight SG 및 SA

Keysight SG (예: PSG, MXG, EXG 시리즈)는 ANTSDR 수신(Rx) 테스트 시 기준 입력 신호를 제공하거나, 송신(Tx) 테스트 시 교정 및 측정 기준을 제공할 수 있습니다.[8] Keysight SA (예: PSA, X-Series SA)는 ANTSDR이 송신하는 신호의 스펙트럼 마스크, 전력 레벨, 인접 채널 누설비(ACLR)와 같은 중요 RF 성능 지표를 정밀하게 분석합니다.[10] 이들 장비는 PyVISA를 통해 이더넷(LXI) 또는 USBTMC 인터페이스로 원격 제어됩니다.[12]

1.2.2. ANTSDR (DUT)

ANTSDR E200 또는 E310 모델은 테스트 대상 장치로 사용됩니다.[1] 이 장치는 PlutoSDR 펌웨어와 호환되며 [1], 이더넷 연결을 통해 IIO 드라이버를 기반으로 하는 pyadi-iio 라이브러리의 원격 호출로 제어됩니다. 장치의 기본 IP 주소는 통상 192.168.2.1로 설정되는 경우가 많습니다.[14] 이더넷 연결을 통한 원격 제어는 Keysight 장비의 VISA 통신과 독립적으로 운영됩니다.

1.2.3. 소프트웨어 스택

자동화 스택은 Python 언어 위에 구축됩니다. PyVISA는 Keysight 계측기와의 통신을 위한 상위 추상화 계층 역할을 하며 [15], 이 통신을 위해서는 Keysight IO Libraries Suite와 같은 VISA 백엔드(Shared Library)가 필수적으로 설치되어야 합니다.[16] ANTSDR 제어를 위해서는 ADI에서 제공하는 pyadi-iio 라이브러리가 설치되어야 하며, 이는 ANTSDR/PlutoSDR 내부의 IIO 드라이버와 통신하여 SDR의 RF 속성(주파수, 이득, 샘플 속도 등)을 설정하고 IQ 데이터 버퍼를 관리합니다.[7]

1.3. 권장 하드웨어 연결 및 안전 수칙 (RF Attenuation 필수 강조)

정밀한 RF 자동화 테스트를 수행하기 위해서는 DUT(ANTSDR)와 계측기(Keysight SA) 간의 물리적 연결이 정확하고 안전하게 이루어져야 합니다.

1.3.1. 물리적 연결 설정

ANTSDR의 송신(Tx) 포트는 RF 케이블을 통해 Keysight SA의 RF 입력 포트에 연결됩니다. 이 연결은 ANTSDR이 생성하는 신호를 SA가 측정하는 기본 경로입니다. 만약 ANTSDR의 수신(Rx) 성능을 테스트한다면, Keysight SG의 RF 출력 포트가 ANTSDR의 Rx 포트에 연결됩니다.

1.3.2. 안전 필수 사항**: RF** 감쇠 **(Attenuation)**

Keysight SA의 RF 입력단은 최대 입력 전력 레벨이 엄격하게 제한되어 있습니다 (일반적으로 +30 dBm 이하).[18] ANTSDR과 같은 SDR 장치는 사용자가 Tx Gain을 높게 설정하거나 예기치 않은 동작 모드에서 최대 피크 전력을 발생시킬 수 있습니다. 따라서 SA 입력단을 보호하기 위해 충분한 감쇠량 확보는 절대적으로 중요합니다.

Keysight SA의 내부 감쇠기(예: 최대 30 dB)만으로는 불충분할 수 있으므로, SA 입력단 앞에 외부 고정 감쇠기(Fixed Attenuator)를 추가하는 것이 강력하게 권장됩니다.[19] 예를 들어, ANTSDR의 최대 예상 출력이 +10 dBm이고 SA의 최대 안전 입력이 +30 dBm이라고 가정할 때, SA의 내부 감쇠기를 30 dB로 설정하더라도 40 dB의 총 감쇠량(안전 마진 포함)이 요구될 수 있습니다. 20 dB 또는 30 dB의 고정 감쇠기를 직렬로 사용하면 안전 마진을 충분히 확보할 수 있습니다.

테스트 자동화의 효율성을 극대화하고 동적 범위 조절이 필요할 경우, Keysight 11713D 드라이버에 의해 제어되는 8496G와 같은 프로그래머블 감쇠기(Programmable Attenuator)를 PyVISA를 통해 SG나 SA와 함께 자동화하여 사용할 수 있습니다.[18] 이 경우, 감쇠량 설정 역시 PyVISA/SCPI 명령으로 제어되어 전체 테스트 시퀀스에 통합됩니다.

다음 표는 Keysight SG/SA와 ANTSDR 간의 RF 연결 및 안전 대책을 요약한 것입니다.

Table 1: RF 연결 구성 및 안전 대책

연결 경로	소스 장비	목표 장비	권장 중간 장치	**SCPI/Control Interface**	안전 고려사항

ANTSDR Tx 특성 분석 (DUT 출력)	ANTSDR Tx Port	Keysight SA RF Input	고정 RF 감쇠기 (최소 20 dB), RF 케이블	SA: PyVISA/SCPI, ANTSDR: pyadi-iio	SA 입력단 영구 손상 방지를 위한 충분한 감쇠량 확보.
ANTSDR Rx 특성 분석 (DUT 입력)	Keysight SG RF Output	ANTSDR Rx Port	고정 RF 감쇠기 (5~10 dB), RF 케이블	SG: PyVISA/SCPI, ANTSDR: pyadi-iio	ANTSDR LNA/믹서 보호. 입력 신호 레벨 제한 준수.
장비 제어	호스트 PC	Keysight SG/SA	LAN (LXI) 또는 USBTMC	PyVISA (Keysight IO Suite)	Keysight Connection Expert를 통한 VISA 주소 확인.
ANTSDR 제어	호스트 PC	ANTSDR	이더넷 (IP: 192.168.2.1)	pyadi-iio (libiio)	Keysight VISA 환경과 독립적으로 IIO 드라이버 및 IP 주소 확인.

제**2**장**.** 소프트웨어 환경 구축 및 통신 설정 **(Software Setup and Communication Establishment)**

2.1. Python 및 핵심 라이브러리 설치

자동화 테스트를 시작하기 위해, 호스트 PC에는 Python 3.6 이상의 환경이 구축되어야 합니다.

필요한 핵심 라이브러리 패키지는 pip를 사용하여 설치합니다.

1. **PyVISA** 설치**:** Keysight SG/SA와 통신하기 위한 VISA 래퍼 라이브러리입니다.[16]
   Bash
   pip install pyvisa

2. **NumPy** 및 **Matplotlib** 설치**:** 신호 처리를 위해 복소 파형(IQ data)을 생성하고, 측정 결과를 시각화하는 데 필수적입니다.
   Bash
   pip install numpy matplotlib

3. **pyadi-iio** 설치**:** ANTSDR(AD9361 칩셋 기반)을 제어하고 IQ 데이터를 스트리밍하기 위한 ADI의 Python 인터페이스입니다.[7]
   Bash
   pip install pyadi-iio

PyVISA는 순수 Python으로 작성된 프론트엔드에 불과하며 [6], 실제 하드웨어 통신을 위해서는 시스템에 벤더가 제공하는 VISA 공유 라이브러리(백엔드)가 설치되어 있어야 합니다.

2.2. Keysight IO Libraries Suite 설치 및 VISA 백엔드 구성

Keysight IO Libraries Suite는 Keysight 장비와의 통신을 위한 필수 드라이버 및 VISA 구현체입니다. 이 소프트웨어를 설치하면 LXI(LAN), USBTMC, GPIB와 같은 다양한 인터페이스를 통해 계측기를 제어할 수 있게 됩니다.[23] PyVISA는 이 Keysight IO Libraries Suite를 백엔드로 사용하여 장비와 통신합니다.[16]

Keysight IO Libraries Suite 설치 후, PyVISA의 ResourceManager 객체를 생성하여 통신 세션을 관리합니다. PyVISA는 시스템에 설치된 VISA 라이브러리를 자동으로 감지하려고 시도하지만 [25], 경우에 따라 명시적으로 Keysight VISA 백엔드 경로를 지정할 수 있습니다.[26]

Python

```
import pyvisa

# Keysight VISA 백엔드 경로를 명시적으로 지정할 경우 (Windows 예시)
# rm = pyvisa.ResourceManager('C:\\Program Files (x86)\\IVI
```

```
Foundation\\VISA\\WinNT\\agvisa\\agbin\\visa32.dll')

# 또는 PyVISA가 설치된 백엔드를 자동으로 찾도록 설정
rm = pyvisa.ResourceManager()
```

시스템이 PyVISA를 통해 올바른 VISA 백엔드를 사용하는지 확인하는 것은 안정적인 통신을 보장하는 첫 단계입니다.

2.3. Keysight Connection Expert를 통한 장비 식별 및 VISA Resource String 확보

Keysight Connection Expert는 설치된 Keysight IO Libraries Suite에 포함된 GUI 도구로, 호스트 PC에 연결된 모든 Keysight 장비를 감지하고 관리하며, 각 장비의 고유한 VISA 주소 문자열(Resource String)을 제공합니다.[23] 이 주소는 PyVISA 스크립트에서 특정 장비에 대한 통신 세션을 열 때 사용되는 핵심 식별자입니다.

VISA Resource String의 형식:
VISA 주소 문자열은 버스 유형(GPIB, USB, TCPIP, ASRL), 인터페이스 번호, 그리고 장비의 고유 식별자(IP 주소, 시리얼 번호, 제조사 ID 등)로 구성됩니다.27

- LAN (LXI) 연결 예: TCPIP0::192.168.1.100::inst0::INSTR (Keysight SG 또는 SA)
- USBTMC 연결 예: USB::0x0957::0xXXXX::MYXXXXXX::INSTR

스크립트 실행 전, ResourceManager의 list_resources() 메서드를 사용하여 연결된 장비 목록과 해당 VISA 주소를 확인하는 것이 좋습니다.[12]

Python

```
resources = rm.list_resources()
print("Available VISA Resources:", resources)
# 예: SA_VISA_ADDRESS = resources
```

이 주소를 사용하여 rm.open_resource(SA_VISA_ADDRESS)를 호출함으로써 SA 또는 SG와의 통신 세션을 확립합니다. 이 과정을 통해 PC와 Keysight 장비 간의 안정적인 원격 제어 경로가 확보됩니다.

제3장. Keysight 장비의 SCPI 기반 제어 (SCPI Control of Keysight Instruments)

3.1. SCPI 통신 기본 원리 및 PyVISA를 활용한 I/O 동작

SCPI는 Keysight 장비를 포함한 대부분의 프로그래밍 가능한 계측기를 제어하기 위한 표준화된 명령어 세트입니다.[5] SCPI 명령은 계층적 트리 구조를 가지며, 계측기의 전면 패널에서 수행할 수 있는 모든 기능을 원격으로 제어할 수 있습니다.

PyVISA는 Python에서 이러한 SCPI 명령을 장비로 전송하고 응답을 받는 통신 레이어 역할을 수행합니다.[12] 핵심적인 I/O 함수는 다음과 같습니다:

- **inst.write("COMMAND"):** 장비에 설정을 변경하거나 동작을 지시하는 명령(예: 주파수 설정)을 전송합니다.
- **inst.query("COMMAND?"):** 설정 값을 쿼리하거나 측정을 요청하는 명령을 전송하고, 즉시 장비의 응답을 읽어옵니다. 이는 write와 read를 결합한 기능입니다.[25]

3.1.1. 장비 초기화 및 동기화

모든 자동화 테스트 시작 시, *RST (Reset) 및 *CLS (Clear Status) 명령을 사용하여 장비를 알려진 초기 상태로 설정하고 에러 큐를 지우는 것이 표준 관행입니다.[29] 또한, 장비의 모델 및 펌웨어 정보를 확인하는 *IDN? 쿼리는 연결 확인의 기본 단계입니다.[12]

Python

```
inst.write("*RST; *CLS")
print(f"IDN Query Result: {inst.query('*IDN?').strip()}")
```

SCPI 명령 처리에서 가장 중요한 측면 중 하나는 동기화입니다. Keysight 장비의 복잡한 설정 변경(예: SA에서 스팬 또는 RBW 변경)이나 측정 수행은 시간이 소요되는 비동기 작업입니다. 따라서 설정 명령을 보낸 후, 계측기가 해당 작업을 완전히 완료하기 전에 다음 명령(특히 측정

데이터 쿼리)을 보내면 예상치 못한 결과가 반환되거나 장비 에러가 발생할 수 있습니다. 이를 방지하기 위해, 시간이 오래 걸리는 명령 뒤에는 반드시 *OPC? (Operation Complete Query) 명령을 사용하여 장비가 명령 처리를 완료했는지 확인해야 합니다.[26]

3.2. Keysight SA (Spectrum Analyzer) 설정 및 측정 SCPI 명령 상세

ANTSDR Tx 성능을 특성화하기 위해 Keysight SA를 구성하는 SCPI 명령 세트는 다음과 같습니다.

3.2.1. SA 기본 설정

SA와의 통신 세션이 열리면, ANTSDR이 송신할 주파수에 맞게 SA의 설정을 조정합니다.

- 중앙 주파수 설정 **(Center Frequency):** SA가 스캔할 주파수 범위의 중심을 설정합니다.
  코드 스니펫
  `:SENSe:FREQuency:CENTer 2.4E9`

- 스팬 설정 **(Frequency Span):** 측정할 주파수 폭을 설정합니다.[11] SDR 신호의 대역폭을 포함하도록 충분히 넓게 설정해야 합니다.
  코드 스니펫
  `:SENSe:FREQuency:SPAN 50E6`

- 분해능 대역폭 **(RBW)** 설정**:** 스펙트럼의 해상도와 노이즈 플로어를 결정합니다. 측정 속도와 노이즈 사이의 중요한 균형점입니다.[11]
  코드 스니펫
  `:SENSe:SA:BANDwidth:RESolution 10E3`

- 탐지기 및 평균 설정**:** 전력 정확도를 높이기 위해 RMS 탐지기를 사용하고 비디오 대역폭(VBW)을 RBW의 3배 정도로 설정하는 것이 일반적입니다.
  코드 스니펫
  ```
  :SENSe:SA:DETector:FUNCtion RMS
  :SENSe:SA:BANDwidth:VIDeo:RATio 3
  ```

3.2.2. 채널 전력 측정 및 데이터 획득

ANTSDR의 출력을 측정하는 가장 효율적인 방법 중 하나는 SA의 내장된 Channel Power (CHPower) 측정 기능을 활용하는 것입니다.[33]

1. 측정 활성화 및 설정: SA 모델에 따라 명령어가 다를 수 있으나, 일반적으로 측정 유형을 지정하고 해당 측정의 대역폭을 설정합니다.
   코드 스니펫
   # SA 모드에서 채널 전력 측정 활성화
   :CALCulate:MEASure:CHPower:BWIDth 1E6 # 1 MHz 채널 대역폭 설정

2. 단일 스윕 실행 및 대기: 측정 프로세스를 시작하고 완료될 때까지 대기합니다.
   코드 스니펫
   INITiate:IMMediate
   *OPC?

3. 결과 쿼리: 측정된 채널 전력 값(dBm)을 쿼리하여 Python 변수에 저장합니다.
   코드 스니펫
   :CALCulate:MEASure:CHPower?

측정 속도 최적화를 위해서는 Keysight SA의 디스플레이 업데이트를 비활성화(:DISP:ENABLE OFF)하고, 장비 내부의 불필요한 계산이나 포스트 프로세싱을 끄는 것이 좋습니다 (:CALC:MATH:STATE OFF).[30] 이러한 최적화는 장비 프로세서가 측정 및 데이터 전송에 전념하게 하여 자동화 테스트 시간을 단축시킵니다.

3.3. Keysight SG (Signal Generator) 설정 SCPI 명령 상세

SG는 ANTSDR Rx 테스트를 위한 깨끗한 입력 신호가 필요할 때 사용됩니다.

- 출력 주파수 설정:
  코드 스니펫
  :SOURce:FREQuency 2.4E9

- 출력 전력 레벨 설정: SG의 출력 전력 레벨을 설정합니다.[9] 이 레벨은 ANTSDR Rx 경로의 최대 입력 레벨을 초과하지 않도록 주의해야 합니다.
  코드 스니펫
  :SOURce:POWer -10.0

- RF 출력 활성화:
  코드 스니펫

:OUTPut:STATe ON

제4장. ANTSDR (DUT)의 Python 기반 제어 (pyadi-iio 활용)

4.1. ANTSDR 이더넷 설정 및 pyadi-iio 연결

ANTSDR은 Analog Devices의 AD936x RFIC를 기반으로 하며, PlutoSDR 펌웨어 환경에서 작동합니다.[1] 따라서 ANTSDR과의 통신은 ADI가 제공하는 Python IIO 인터페이스인 pyadi-iio 라이브러리를 통해 이루어집니다.[7] IIO는 SDR의 내부 하드웨어 속성(Attributes)을 추상화하여 Python에서 객체 속성 설정을 통해 RF 파라미터를 제어할 수 있게 합니다.[17]

ANTSDR은 일반적으로 이더넷(LAN)을 통해 호스트 PC와 통신하며, 기본 IP 주소는 192.168.2.1입니다.[14]

Python

```
import adi

ANTSDR_URI = 'ip:192.168.2.1'
try:
    sdr = adi.ad9361(uri=ANTSDR_URI)
    # Tx 채널 활성화 및 Rx 채널 비활성화 (Tx 테스트 시)
    sdr.rx_enabled_channels =
    sdr.tx_enabled_channels =
except Exception as e:
    print(f"ANTSDR connection failed. Ensure IP address and IIO driver stability: {e}")
```

4.2. ANTSDR 송신 (Tx) 경로 구성 및 신호 발생

pyadi-iio를 사용하여 ANTSDR의 송신 경로를 설정하고, Keysight SA가 측정할 CW 톤을 생성하고 스트리밍합니다. ANTSDR 제어의 특징은 Keysight SG처럼 주파수 설정 명령을 보내는 것이 아니라, 호스트 PC에서 생성된 복소 파형(IQ Data) 버퍼를 SDR 하드웨어로 실시간 스트리밍해야 한다는 점입니다.[34]

4.2.1. Tx 파라미터 설정

LO 주파수, 샘플 속도, RF 대역폭 등의 핵심 RF 파라미터는 SDR 객체의 속성으로 직접 설정됩니다.[35]

Python

```
sdr.tx_lo = 2.4e9              # LO 주파수 2.4 GHz 설정
sdr.sample_rate = 30e6         # 샘플 속도 30 MHz 설정
sdr.tx_rf_bandwidth = 20e6     # Tx 필터 대역폭 20 MHz 설정
```

4.2.2. CW 톤 IQ 데이터 생성 및 스트리밍

ANTSDR이 CW 톤을 송신하게 하려면, Python NumPy 라이브러리를 사용하여 복소 사인파 형태의 IQ 데이터를 생성해야 합니다. 이 데이터는 DAC의 동적 범위(일반적으로 ± 1.0) 내에서 정규화되어야 합니다.

Python

```
fs = sdr.sample_rate      # 30 MHz
N = 2**14                 # 버퍼 크기 (16384 samples)
tone_offset = 1e6         # 중심 주파수로부터 1 MHz 오프셋 톤
```

```
ts = np.arange(N) / fs    # 시간 축
amplitude = 0.5           # DAC 포화를 피하기 위한 진폭
# 복소 사인파 생성: I + jQ = A * exp(j * 2 * pi * f_offset * t)
iq_data = amplitude * np.exp(1j * 2 * np.pi * tone_offset * ts)

# 데이터 송신 및 순환 버퍼 설정
sdr.tx_cyclic_buffer = True # 버퍼의 내용을 반복하여 송신
sdr.tx(iq_data)           # 송신 시작
```

이 과정을 통해 ANTSDR은 설정된 LO 주파수(2.4 GHz)에 1 MHz 오프셋을 가진 톤 신호를 출력하기 시작합니다. 이때 송신 전력은 sdr.tx_hardwaregain_chan0 속성을 통해 제어됩니다.[14]

제5장. 통합 자동화 테스트 시나리오 구현 (Integrated Automated Test Scenario Implementation)

5.1. 사례 연구: ANTSDR Tx 출력 전력 스윕 자동 측정

본 통합 자동화 시나리오는 ANTSDR의 핵심 RF 성능 중 하나인 송신 전력 특성을 분석하는 것을 목표로 합니다. Keysight SA를 사용하여 ANTSDR의 Tx LO 주파수를 고정하고, pyadi-iio를 통해 ANTSDR의 Tx Hardware Gain(tx_hardwaregain_chan0)을 단계적으로 스윕하면서 각 이득 레벨에서의 실제 RF 출력 전력을 Keysight SA가 측정하고 기록합니다.

5.2. 테스트 시퀀스 로직 설계 (Flowchart Logic)

성공적인 통합 테스트는 두 장비의 독립적인 제어 메커니즘을 마스터 Python 스크립트 내에서 효과적으로 동기화하는 데 달려 있습니다.

1. 초기 환경 설정: PyVISA ResourceManager를 초기화하고, Keysight SA (SCPI) 및 ANTSDR (IIO)에 대한 통신 세션을 동시에 엽니다.
2. **ANTSDR Tx** 설정: pyadi-iio를 사용하여 LO 주파수 및 샘플 속도를 설정하고, 측정에 사용될 IQ 톤 신호를 생성하여 순환 송신 버퍼에 로드합니다. 이때, 초기 Tx Gain을 가장 낮은 값으로 설정하여 RF 출력을 시작합니다.
3. **Keysight SA** 설정: PyVISA/SCPI를 사용하여 SA의 중앙 주파수와 스팬을 ANTSDR 송신

주파수에 맞추고, RBW, VBW, Channel Power 측정 모드를 구성합니다.

4. **Sweep** 루프 실행**:**
   - **ANTSDR Gain** 업데이트**:** sdr.tx_hardwaregain_chan0 속성을 다음 스텝으로 업데이트합니다.
   - 안정화 대기**:** SDR 하드웨어 설정이 완전히 적용되고 RF 출력이 안정화될 시간을 확보하기 위해 짧은 지연 시간(Time Delay)을 삽입합니다. 이는 비동기적 하드웨어 설정에서 발생하는 오류를 방지하는 중요한 단계입니다.
   - **SA** 측정 동기화**:** Keysight SA에 INITiate:IMMediate 명령을 보내 단일 스윕을 시작하도록 지시하고, 곧바로 *OPC? 쿼리를 실행하여 스윕 완료를 기다립니다.
   - 결과 수집**:** SA로부터 Channel Power 측정 결과(:CALCulate:MEASure:CHPower?)를 쿼리하여 전력 값(dBm)을 추출합니다.
   - 데이터 기록**:** 현재 Tx Gain 값과 측정된 Power 값을 배열에 저장합니다.
5. 종료 및 정리**:** Sweep 완료 후, ANTSDR Tx Gain을 최소값으로 설정하고 (-89.75 dBm 등) Keysight SA 및 ANTSDR 연결을 안전하게 해제합니다.

5.3. Python 자동화 마스터 스크립트 (Keysight/ANTSDR 통합)

마스터 스크립트는 Keysight 장비의 SCPI 기반 명령을 캡슐화하는 함수와 ANTSDR의 객체 속성 설정 및 버퍼링을 담당하는 함수로 구조화되어야 합니다. 다음은 이 통합 구조를 구현하는 주요 코드 블록입니다.

Table 2: 핵심 SCPI 및 pyadi-iio 제어 명령 요약

장비	기능	**SCPI / pyadi-iio** 명령 **(Python Syntax)**	설명
Keysight SA	장비 초기화	SA.write("*RST; *CLS")	장비를 알려진 초기 상태로 설정
Keysight SA	측정 완료 대기	SA.query("*OPC?")	이전 명령이 완료될 때까지 호스트 대기 [29]
Keysight SA	채널 전력 측정 쿼리	power = SA.query(":CALCulate:MEASure:CHPow	SA에서 채널 전력 측정 값 쿼리 [33]

		er?").split(',')	
ANTSDR (Tx)	객체 초기화	sdr = adi.ad9361(uri='ip:192.168.2.1')	AD9361 (ANTSDR) IIO 객체 연결 [14]
ANTSDR (Tx)	LO 주파수 설정	sdr.tx_lo = <Hz>	ANTSDR 송신 LO 주파수 설정 [35]
ANTSDR (Tx)	Tx Gain 설정 (Sweep 변수)	sdr.tx_hardwaregain_chan0 = <dB>	아날로그/하드웨어 Tx 이득 설정
ANTSDR (Tx)	IQ 데이터 송신 시작	sdr.tx(iq_data)	호스트에서 생성된 복소 파형 데이터 버퍼 송신 [17]

Python

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

# ------------------------------------------------------------------
# 1. 환경 설정 및 장비 주소 (사용 환경에 맞게 수정 필요)
# ------------------------------------------------------------------
SA_ADDRESS = 'TCPIP0::192.168.1.100::inst0::INSTR'
ANTSDR_URI = 'ip:192.168.2.1'

def initialize_instruments(sa_addr, sdr_uri):
    """PyVISA ResourceManager 및 Keysight SA/ANTSDR 연결 초기화"""
    rm = pyvisa.ResourceManager()

    # Keysight SA 연결 (PyVISA)
    SA = rm.open_resource(sa_addr)
    SA.timeout = 10000
    SA.write("*RST; *CLS")
    print(f"Keysight SA Connected: {SA.query('*IDN?').strip()}")
```

```
    # ANTSDR 연결 (pyadi-iio)
    sdr = adi.ad9361(uri=sdr_uri)
    sdr.rx_enabled_channels =
    sdr.tx_enabled_channels =
    print("ANTSDR (AD9361) Connected.")
    return SA, sdr

def config_sa_measurement(SA, center_freq, span, rbw):
    """SCPI를 사용하여 Keysight SA 측정 환경 설정"""
    # Keysight SA의 디스플레이를 끄고 불필요한 계산을 비활성화하여 속도 최적화
    SA.write(":DISP:ENABLE OFF")
    SA.write(":CALC:MATH:STATE OFF")

    SA.write(f":SENS:FREQ:CENT {center_freq}")
    SA.write(f":SENS:FREQ:SPAN {span}")
    SA.write(f":SENS:SA:BAND:RES {rbw}")
    SA.write(":SENS:SA:DET:FUNC RMS")  # RMS 탐지기 사용

    # Channel Power 측정 설정 (예시)
    SA.write(":CALC:CUST:DEF 'CHPWR', 'Channel Power', 'SA1:Spectrum Analyzer'")
    SA.write(":CALC:MEAS:CHPower:BWIDth 1E6") # 1 MHz Channel BW

    # 설정 완료 대기
    SA.query("*OPC?")

def generate_and_transmit_cw(sdr, center_freq, sample_rate, tone_offset=1e6):
    """ANTSDR TX 설정을 완료하고 CW 톤 송신 시작"""
    sdr.sample_rate = int(sample_rate)
    sdr.tx_lo = int(center_freq)
    sdr.tx_rf_bandwidth = int(sample_rate)

    # IQ Data 생성 (DAC 포화를 피하기 위해 0.5 진폭 사용)
    N = 2**14
    ts = np.arange(N) / sample_rate
    iq_data = 0.5 * np.exp(1j * 2 * np.pi * tone_offset * ts)

    sdr.tx_cyclic_buffer = True
    sdr.tx(iq_data)

def measure_channel_power(SA):
    """SA에서 Channel Power 측정 값을 읽어옴 (SCPI 동기화 포함)"""
    # 단일 스윕 시작 및 완료 대기 (*OPC?를 사용하여 동기화)
```

```
    SA.write("INITiate:IMMediate")
    SA.query("*OPC?")

    # 측정 결과 쿼리 (쉼표로 구분된 문자열 반환)
    result = SA.query(":CALCulate:MEASure:CHPower?")

    try:
        # PyVISA는 SCPI 문자열 결과를 반환하며, 이를 파싱하여 실수 값만 추출
        power_dbm = float(result.split(','))
        return power_dbm
    except ValueError as e:
        # 파싱 오류 처리
        print(f"SCPI parsing error: {result}")
        return float('nan')

def main_test_sequence():

    # 1. 초기화
    SA, sdr = initialize_instruments(SA_ADDRESS, ANTSDR_URI)

    TEST_LO_FREQ = 2.401e9  # 2.401 GHz (ANTSDR Tx LO)
    TEST_SAMPLERATE = 30e6

    # 2. Keysight SA 설정
    config_sa_measurement(SA, TEST_LO_FREQ + 1e6, span=5e6, rbw=10e3) # SA 중심 주파수를
톤 주파수에 맞춤

    # 3. ANTSDR 송신 시작
    generate_and_transmit_cw(sdr, TEST_LO_FREQ, TEST_SAMPLERATE, tone_offset=1e6)

    # 4. Sweep 루프
    # Tx Hardware Gain -20 dBm 부터 +5 dBm까지 5 dB 스텝
    gain_levels = np.arange(-20.0, 5.0, 5.0)
    measured_powers =

    print("\n--- Starting ANTSDR Tx Power Sweep (SA Measurement) ---")
    for gain in gain_levels:
        # ANTSDR Gain 설정 (IIO API)
        sdr.tx_hardwaregain_chan0 = gain
        time.sleep(0.5) # 하드웨어 설정 안정화 대기

        # Keysight SA 측정 및 결과 획득 (PyVISA/SCPI API)
        power = measure_channel_power(SA)
```

```
        measured_powers.append(power)

        # 에러 체크 (선택 사항: SA.query(":SYSTem:ERRor?"))

        print(f"Tx Gain Setting: {gain:.1f} dB | Measured RF Power: {power:.2f} dBm")

    # 5. 정리
    sdr.tx_hardwaregain_chan0 = -89.75 # RF 출력 최소화
    sdr.close()
    SA.write(":DISP:ENABLE ON") # 디스플레이 복원
    SA.close()

    # 6. 결과 시각화
    plt.figure()
    plt.plot(gain_levels, measured_powers, 'o-')
    plt.title('ANTSDR Tx Gain vs. Measured RF Output Power')
    plt.xlabel('ANTSDR Tx Hardware Gain Setting (dB)')
    plt.ylabel('Measured Channel Power (dBm)')
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    try:
        main_test_sequence()
    except Exception as e:
        print(f"Test Execution Failed. Review connections, VISA/IIO drivers, and SCPI addresses: {e}")
```

제6장. 결과 분석, 시각화 및 결론 (Data Analysis, Visualization, and Conclusion)

6.1. 테스트 결과 데이터 파싱 및 저장

자동화 루프를 통해 수집된 데이터(Tx Gain 설정값과 SA 측정 전력값)는 NumPy 배열 또는 Python 리스트 형태로 메모리에 저장됩니다. Keysight SA에서 측정 결과(예: :CALCulate:MEASure:CHPower?)를 쿼리할 때 반환되는 데이터는 일반적으로 쉼표로 구분된

ASCII 문자열입니다. 스크립트는 이 문자열을 파싱하여(예: result.split(',')) 실수(float) 값으로 변환해야 합니다.

측정된 데이터를 영구적으로 기록하기 위해서는, 테스트 종료 후 이 데이터를 CSV 또는 기타 표준 형식으로 저장합니다. 이는 반복 가능한 테스트 기록을 보존하고 후속 분석을 위한 기반을 마련합니다.

6.2. 측정 데이터 시각화

Matplotlib를 사용하여 저장된 데이터를 시각화하는 것은 ANTSDR의 성능 특성을 이해하는 데 필수적입니다. 위의 스윕 테스트에서 얻은 Tx Gain 대 RF 출력 전력 곡선(Power Transfer Curve)은 장치의 선형 작동 영역을 명확하게 보여줍니다. 이 곡선이 선형적으로 증가하는 구간은 증폭기가 신호를 왜곡 없이 처리하는 영역이며, 곡선이 평탄해지기 시작하는 지점은 장치가 포화(Saturation)되기 시작하여 성능이 저하됨을 나타냅니다. 플롯의 축 레이블에 정확한 RF 단위(dB, dBm, Hz)를 명시하는 것은 엔지니어링 보고의 정확성을 높이는 데 기여합니다.

6.3. 자동화 테스트 환경의 확장 가능성 및 성능 최적화 방안

6.3.1. 확장성 및 복잡한 측정 통합

본 통합 프레임워크는 Keysight SA의 다양한 측정 기능으로 쉽게 확장 가능합니다. Keysight X-Series SA와 같은 장비는 인접 채널 누설비(ACLR), 변조 품질(EVM), 스퓨리어스 분석 등을 위한 전문 측정 애플리케이션을 지원합니다.[36] 이러한 측정을 자동화하려면, 해당 애플리케이션을 활성화하고 측정 파라미터를 설정하는 SCPI 명령(예: ACLR 측정 관련 명령)을 통합 스크립트에 추가하고, pyadi-iio를 통해 ANTSDR이 측정에 필요한 변조 파형(예: 5G NR 또는 LTE)을 송신하도록 구성하면 됩니다.

6.3.2. 성능 최적화 전략

자동화 테스트의 속도를 높이는 것은 대규모 테스트 캠페인에 필수적입니다. 성능 최적화를

위한 두 가지 주요 방법이 확인됩니다:

1. 데이터 전송 최적화: Keysight SA에서 전체 스펙트럼 트레이스 데이터(TRACe:DATA?)와 같이 대용량 데이터를 쿼리해야 할 때, ASCII 형식 대신 Binary Block Transfer 형식을 사용하면 데이터 전송 속도를 크게 향상시킬 수 있습니다. Keysight 장비와 PyVISA는 이진 블록 데이터 전송을 지원하며, 이는 특히 대용량 트레이스를 획득할 때 I/O 오버헤드를 줄여줍니다.[29]
2. 장비 내부 프로세스 최소화: 계측기가 불필요한 작업을 수행하지 않도록 제어하는 것이 중요합니다. 예를 들어, SA의 전면 패널 디스플레이 업데이트는 상당한 처리 시간을 소모할 수 있으므로, :DISP:ENABLE OFF 명령을 사용하여 이를 비활성화합니다. 또한, 장비 내부에서 수행되는 통계 계산이나 한계선 검사와 같은 포스트 프로세싱 작업도 :CALC:MATH:STATE OFF와 같은 명령으로 비활성화하여 프로세서의 자원을 측정 및 통신에 집중시키면 측정 속도가 빨라집니다.[30]

결론 및 요약

본 보고서는 Keysight SG/SA 계측기와 ANTSDR SDR 플랫폼의 RF 자동화 테스트를 위한 통합 튜토리얼을 제공하였습니다. 핵심은 PyVISA/SCPI를 사용하여 Keysight 장비의 정밀한 측정 기능을 제어하고, pyadi-iio를 사용하여 ANTSDR의 유연한 RF 속성과 IQ 버퍼 스트리밍을 제어하는 Python 마스터 스크립트를 성공적으로 통합하는 것입니다.

이 프레임워크는 RF 안전을 위한 외부 감쇠기 사용의 중요성을 강조하고, SCPI의 *OPC? 명령을 통한 Keysight 장비의 동기화와 pyadi-iio의 객체 속성 설정을 통한 ANTSDR 제어를 명확히 구분하여 설명했습니다. 제시된 ANTSDR Tx Power Sweep 시나리오는 실제 RF 특성 분석의 기초를 제공하며, Binary Data Transfer 및 장비 내부 프로세스 최적화와 같은 고급 기술을 적용하여 테스트 속도와 효율성을 극대화할 수 있습니다. 이 통합 자동화 환경은 RF 시스템 통합 엔지니어 및 연구원들이 ANTSDR과 같은 SDR 기반 DUT의 성능을 Keysight의 상용 계측기로 빠르고 정확하게 특성화할 수 있는 강력한 기반을 제공합니다.

참고 자료

1. Getting Started with AntSDR - Crowd Supply, 10월 19, 2025에 액세스, https://www.crowdsupply.com/microphase-technology/antsdr-e200/updates/getting-started-with-antsdr
2. Using AntSDR for Cellular Networking - Crowd Supply, 10월 19, 2025에 액세스, https://www.crowdsupply.com/microphase-technology/antsdr-e200/updates/using-antsdr-for-cellular-networking
3. RF Test Solutions | Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/lib/resources/solution-briefs/rf-test-solutions-2185428.html
4. Keysight Solutions for RF Wireless Coexistence testing according to ANSI C63.27,

10월 19, 2025에 액세스, https://www.youtube.com/watch?v=tEb0mVEKTxA
5. Controlling instruments over GPIB with SCPI - H.Gens, 10월 19, 2025에 액세스, https://h-gens.github.io/controlling-instruments-over-gpib-with-scpi.html
6. pyvisa/pyvisa: A Python package with bindings to the "Virtual Instrument Software Architecture" VISA library, in order to control measurement devices and test equipment via GPIB, RS232, or USB. - GitHub, 10월 19, 2025에 액세스, https://github.com/pyvisa/pyvisa
7. analogdevicesinc/pyadi-iio: Python interfaces for ADI hardware with IIO drivers (aka peyote), 10월 19, 2025에 액세스, https://github.com/analogdevicesinc/pyadi-iio
8. E8257D, E8267D, E8663D PSG Signal Generators SCPI Command Reference - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/assets/9018-04965/programming-guides/9018-04965.pdf
9. Agilent Technologies PSG Signal Generators - SCPI Command Reference - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/ae/en/assets/9018-40793/programming-guides/9018-40793.pdf
10. SCPI Command Reference - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/assets/9018-07931/reference-guides/9018-07931.pdf
11. Setup Commands - Spectrum Analyzer Measurement Class - Keysight, 10월 19, 2025에 액세스, https://helpfiles.keysight.com/csg/pxivna/Programming/CF_Setup_Commands_-_SA.htm
12. PyVISA: Control your instruments with Python — PyVISA 1.15.1.dev27+g908d86744 documentation, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/
13. ANTSDR Unpacking and Test Rev. 1.1, 10월 19, 2025에 액세스, https://down.hgeek.com/download/93159/93159-E310-AD9361-Rev-1-1-English-Manual.pdf
14. Using Python To Control The Pluto Radio And Plot Data - EngineerZone Spotlight - EZ Blogs, 10월 19, 2025에 액세스, https://ez.analog.com/ez-blogs/b/engineerzone-spotlight/posts/using-python-to-control-the-pluto-radio-and-plot-data
15. Control your instruments with Python — PyVISA 1.15.0 documentation, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/en/stable/
16. Installation — PyVISA 1.15.0 documentation - Read the Docs, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/en/stable/introduction/getting.html
17. Examples — Analog Devices Hardware Python Interfaces 0.0.3 documentation, 10월 19, 2025에 액세스, https://pyadi-iio.readthedocs.io/en/latest/guides/examples.html
18. 8496G Programmable Attenuator, 4 GHz, 110dB, 10 dB steps - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/dk/en/product/8496G/programmable-attenuator-4-gh

z-110db-10-db-steps.html
19. Attenuators | Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/products/accessories/attenuators.html
20. 8494B Manual Attenuator, 18 GHz, 11 dB, 1-dB Steps - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/dk/en/product/8494B/manual-attenuator-18-ghz-11-db-1-db-steps.html
21. 84905M Programmable Step Attenuator, 50 GHz, 60 dB, 10 dB steps | Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/sg/en/product/84905M/programmable-step-attenuator-50-ghz-60-db-10-db-steps.html
22. Python Installation Guide - Keysight, 10월 19, 2025에 액세스, https://helpfiles.keysight.com/csg/pxivna/Programming/Learning_about_GPIB/Python_Installation_Guide.htm
23. Communicating with your instrument — PyVISA 1.15.1.dev27+g908d86744 documentation, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/en/latest/introduction/communication.html
24. IO Libraries Example Programs - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/lib/software-detail/programming-examples/io-libraries-example-programs-2798637.html
25. Tutorial — PyVISA 1.8 documentation - Read the Docs, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/en/1.8/tutorial.html
26. Python Example - Keysight, 10월 19, 2025에 액세스, https://helpfiles.keysight.com/csg/FFProgrammingHelpWebHelp/Examples/Python_Example.htm
27. VISA resource names — PyVISA 1.8 documentation - Read the Docs, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/en/1.8/names.html
28. Use PyVISA to Program Test Tools with Python - Workbench Wednesdays - YouTube, 10월 19, 2025에 액세스, https://www.youtube.com/watch?v=1HQxnz3P9P4
29. A more complex example — PyVISA 1.8 documentation - Read the Docs, 10월 19, 2025에 액세스, https://pyvisa.readthedocs.io/en/1.8/example.html
30. Making fast measurements with a frequency counter - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/lib/software-detail/programming-examples/making-fast-measurements-with-a-frequency-counter-524044.html
31. SCPI Basic Commands - Keysight, 10월 19, 2025에 액세스, https://helpfiles.keysight.com/csg/n5106a/scpi_basic_commands.htm
32. Create a Spectrum Analyzer Measurement - Keysight, 10월 19, 2025에 액세스, https://helpfiles.keysight.com/csg/e5055a/Programming/GPIB_Example_Programs/Spectrum_Analyzer.htm
33. A SA Mode Commands - Keysight, 10월 19, 2025에 액세스, https://helpfiles.keysight.com/csg/FFProgrammingHelpWebHelp/A_SA_Mode_Commands.htm
34. Buffers — Analog Devices Hardware Python Interfaces 0.0.19 documentation,

10월 19, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/buffers/index.html
35. PlutoSDR in Python | PySDR: A Guide to SDR and DSP using Python, 10월 19, 2025에 액세스, https://pysdr.org/content/pluto.html
36. C8702000A RF Automation Test Software - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/mx/en/product/C8702000A/rf-automation-test-software.html
37. S8702A RF Automation Toolset - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/se/en/product/S8702A/s8702a-rf-automation-toolset.html
38. Tips for Querying CW and Average Power - Keysight, 10월 19, 2025에 액세스, https://www.keysight.com/us/en/assets/7018-03995/application-notes/5991-2641.pdf