

End-to-End OAI 5G SA Testbed

Objective: This document outlines the step-by-step process for setting up an end-to-end OpenAirInterface (OAI) 5G Standalone (SA) testbed using OAI Core Network (CN5G), OAI gNB, and a Commercial Off-The-Shelf (COTS) User Equipment (UE). The goal is to establish a functional 5G SA network utilizing USRP B210 SDRs and test its connectivity with real UEs.

The setup includes:

- **Building UHD from source** to support USRP B210 hardware.
- **Deploying OAI Core Network (CN5G) using Docker** to manage essential 5G control and data plane functions (AMF, SMF, UPF, etc.).
- **Compiling and configuring OAI gNB** to communicate with CN5G and handle radio access.
- **Programming and configuring SIM cards** to enable authentication and network registration.
- **Testing connectivity with UEs**, ensuring successful network registration and data transmission.
 - A **COTS UE** and
 - **Simulated UE** with a second USRP B210 and a laptop

Tables of Contents:

1. Equipment
2. Requirements
 - A. UHD
 - B. OAI Core - CN5G
 - C. OAI gNB
 - D. Running CN5G + gNB
 - E. COTS UE
 - F. Special Configurations for CN5G
 - G. Special Configurations for gNB
 - H. Connecting Phone to CN5G + gNB
 - I. Notes on phone selections and settings

Equipment:

Equipment Name	Hardware/Software	Version (if applicable)	Number	Notes
Desktop	Hardware	OS 22.04	2	CN5G, gNB, nrUE
Laptop	Hardware	OS 20.04	1	ChipWhisperer
ChipWhisperer Husky	Hardware		1	For side channel analysis
H Probe	Hardware		1	For side channel analysis
USRP B210	Hardware - Radios		2	For gNB and nrUE
COTS UE	Hardware	Google Pixel 5 OS Android 11	1	UE
SIM Card	Hardware		4	For COTS UE
OAI	Software	(latest version)		Installs CN5G, gNB, and nrUE
ChipWhisperer	Software	(latest version)		Can use CW Jupyter-Notebook interface

Requirements:

- One Ubuntu 22.04 Desktop to install the OAI CN5G and gNB
- If creating a simulated UE, another Ubuntu 22.04 Desktop for OAI nrUE implementation. This machine must be on the same subnet as the CN5G and gNB desktop.
- One Ubuntu 20.04 Laptop/Desktop to install ChipWhisperer
- If using a COTS UE, you need OAI-compatible phones. A list can be found [here](#). We used a Google Pixel 5 with OS Android 11 (we had to downgrade the OS version manually).
- SIM card for the COTS UE matters. We found that the OYEITIMES SIM Card was compatible with OAI.

A. UHD

Build from Source:

```
# https://files.ettus.com/manual/page_build_guide.html
sudo apt install -y autoconf automake build-essential ccache cmake
cpufrequtils doxygen ethtool g++ git inetutils-tools libboost-all-dev
libncurses-dev libusb-1.0-0 libusb-1.0-0-dev libusb-dev python3-dev
python3-mako python3-numpy python3-requests python3-scipy
python3-setuptools python3-ruamel.yaml

git clone https://github.com/EttusResearch/uhd.git ~/uhd
cd ~/uhd
git checkout v4.7.0.0
cd host
mkdir build
cd build
cmake ../
make -j $(nproc)
make test # This step is optional
sudo make install
sudo ldconfig
sudo uhd_images_downloader
```

B. OAI Core - CN5G

- Runs with docker

Pre-requisites:

```
sudo apt install -y git net-tools putty

# https://docs.docker.com/engine/install/ubuntu/
sudo apt update
sudo apt install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo
"$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null
```

```
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin

# Add your username to the docker group, otherwise you will have to run in
sudo mode.
sudo usermod -a -G docker $(whoami)
reboot
```

Download Configuration Files:

```
wget -O ~/oai-cn5g.zip
https://gitlab.eurecom.fr/oai/openairinterface5g/-/archive/develop/openairi
nterface5g-develop.zip?path=doc/tutorial_resources/oai-cn5g
unzip ~/oai-cn5g.zip
mv
~/openairinterface5g-develop-doc-tutorial_resources-oai-cn5g/doc/tutorial_r
esources/oai-cn5g ~/oai-cn5g
rm -r ~/openairinterface5g-develop-doc-tutorial_resources-oai-cn5g
~/oai-cn5g.zip
```

Pull Docker Image:

```
cd ~/oai-cn5g
docker compose pull
```

Ready to Run!

See Section D on how to run.

C. OAI gNB

****Note: This runs a monolithic gNB (not O-RAN)**

```
# Get openairinterface5g source code
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
~/openairinterface5g
cd ~/openairinterface5g
git checkout develop

# Install OAI dependencies
cd ~/openairinterface5g/cmake_targets
./build_oai -I

# Build OAI gNB
```

```
cd ~/openairinterface5g/cmake_targets
./build_oai -w USRP --ninja --gNB -C
```

Ready to Run!

See section D on how to run.

D. Running CN5G + GnB:

→ **CN5G**

```
cd ~/oai-cn5g
docker compose up -d
```

Or to see the core logs

```
docker compose up
```

→ **GnB (specific to our B210 config file)**

```
cd ~/openairinterface5g/cmake_targets/ran_build/build
sudo ./nr-softmodem -O
../../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/gnb.sa.band78.fr1.106PRB.usrp
b210.conf -E --continuous-tx
```

gNB implementation with O-RAN needs a CU/DU split gNB. To achieve this, follow the steps below. Once the monolithic gNB is set up, only a few tweaks are required to split the CU and DU.

The cu_gnb.conf and the du_gnb.conf are the configuration files that need to be reviewed to set up the gNB using ORAN. The local port number in the du_gnb.conf must be altered as per your setup and must match the cu_gnb.conf. Furthermore, the cu_gnb.conf must be altered to contain the amf IP address listed in the docker-compose config file and the gnb IP address listed in the gNB config file.

E. COTS UE

Devices we have tested:

- *SIM Card:*
 - Sysmocom SJA5
 - OYEITIMES Test Sim
- *Phones:*
 - One Plus 8T
 - Motorola
 - Galaxy A5
 - Google Pixel 5

- *Combinations that worked:*
 - Google Pixel 5 + OYEITIMES Test Sim

Programming your SIM Card:

- For a SJA5 - use pysim.
- For OYEITIMES Test SIM, use downloaded software and edit under the GSM/WCDM/LTE tab. Values to edit:
 - GSM Parameters:
 - IMSI15
 - KI
 - PLMN
 - SPN - changed to oai
 - FPLMN, HPLMN delete those values
 - Everything else stays the same
 - LTE/WCDMA Parameters:
 - IMSI15
 - KI
 - OPC
 - SPN - changed to oai
 - Everything else is deleted

We used default values as seen below.

[image]

Phone Settings:

- Configuring APN: [image of our APN settings]
- Force 5G
 - Dial `*##4636##`
 - This will take you to Phone Information
 - Set to NR Only.

F. Special Configurations for CN5G:

oai_db.sql:

- This database needs to be configured specific to the UE you are using. In our case, we will use a COTS solution, a Google Pixel phone.
- Under Authentication Management, we will either change or add an entry that specific our phone's SIM information.
- This information includes IMSI, OPc, Ki, and AMF value.
- This must match the SIM Information.

Our Database Entry:

[image]

docker_compose.yaml:

- This file defines all the containers with the core settings such as the amf, udm, upf etc.

- All IP addresses for each must be changed to reflect the new router IP. In our case the first three numbers were 192.168.0._ The last number of the IP must be different for each container.

[image]

G. Special Configurations for GnB:

Our setup works with a router to create a local network, rather than use WiFi/Ethernet. Because of this, we need to change IP addresses in the GnB config file,

`gnb.sa.band78.fr1.106PRB.usrpb210.conf`.

[image of gnb search list IPs]

- PLMN must be changed to test PLMN and match the OAI_DB.SQL and the SIM card. Ours is 00101. *NOTE: PLMN is a combination of MCC and MNC. MCC=001; MNC=01

[image of our PLMN settings]

H. Connecting Phone to CN5G+GnB

- Run CN5G first.
- Run GnB
- Make sure GnB is connected to the core.
- On phone, put it in airplane mode first.
- Then connect to Mobile Network. It should automatically select. The top should read OAI 5G.

I. Notes on phone selections and settings

Selecting a phone:

- Must be compatible with OAI
- Test SIM card for the phone must be compatible with OAI

From our experience:

- We tested multiple phones for our UE (Motorola, Nokia, OnePlus, Samsung). We found that the OS version is critical. OAI states up to Android 13 is compatible, however, we were unable to connect our Android 12 and 13 phones to our testbed.
- This is why a Google Pixel worked the best. Google software is online and available for an easy OS downgrade to Android 11. We used a Pixel 5, but we assume any Pixel could be compatible as long as it has Android 11.
- However, the Pixel purchased MUST be carrier unlocked and NEW. We tested on a revamped Pixel 5 and the OEM option in developer settings (crucial to enable when downgrading the OS version) was grayed out. This indicates that the phone was not truly unlocked.
- **Downgrading the OS Version:**

- OEM Unlock in Developer Settings must be enabled. This is the first step in unlocking the bootloader.
- Using fastboot, unlock the bootloader.
- Using Android SDK, download the OS version and then flash it to the phone.

Programming the SIM Card:

- Need a card reader!
- We tried using Sysmocom SIM cards. According to the OAI website, only SJA2 cards are compatible with OAI. However, they aren't selling those anymore and replaced them with SJA5. We tested with SJA5 and were not able to get it to work.
- For the SJA5, Pysim was required to program the SIM card.
- **Using Pysim:**
- We used OYEITIMES test SIM cards with our Google Pixel. OYEITIMES came with its own software to program.

SIM Personalize tools(Copyright: OYEITIMES Co.Ltd Ver 4.2.11)

Reader(PC/SC): Refresh Read Card Write Card Save Data Load Data Exit

Batch Write Card
Data File: Select File Go First Prev Next Last Find Continue Template

Common Parameter
ATR: 3B9F94801FC78031E073FE2113578681098698621880 Type: LTE(VH02):LTE+GSM Language: English ... ADN
ICCID: 98081730640100007024 ☐ Inc (DEC20) PIN1: 1234 PUK1: 88888888 PIN2: 1234 PUK2: 88888888 (ASC8) ADM: 3838383838383838 (HEX16/8)

GSM/wCDMA/LTE CDMA/EVDO/CSIM VoLTE/ISIM Java PKCS#15/AC

GSM Parameter
☐ IMSI18: 909001010000000001 ☒ IMSI15: 0010100000000001 ☐ Inc (DEC18/15)
ACC: 0002 ☐ Input (DEC4) AD: 00000002 ...
☒ Inc KI: 00112233445566778899AABCCDEF00 (HEX32)
PLMN: 00101 ... Auto
EHPLMN: ...
FPLMN: ...
HPLMN: 05 (HEX2) GID1: GID2: (HEX)
SMSP: +6807790000 MSISDN: ☐ Inc (ASC)
SPN: oai (ASC)
ECC: ...
Algorithm: ☒ Comp128-1 ☐ Comp128-2 ☐ Comp128-3 ☐ Milenage

Other files Same with LTE

LTE/wCDMA Parameter
☐ IMSI18: 909001010000000001 ☒ IMSI15: 0010100000000001 ☐ Inc (DEC18/15)
ACC: 0002 ☐ Input (DEC4) AD: 00000002 ...
☒ Inc KI: 00112233445566778899AABCCDEF00 (HEX32)
☒ OPC: 63BFA50EE6523365FF14C1F45F88737D (HEX32)
☐ OP: (HEX32)
PLMNwAct: ... Auto
OPLMNwAct: ...
HPLMNwAct: ...
EHPLMN: ...
FPLMN: 311480; 310260; 310410; 313100; 313340 ...
HPPLMN: 05 (HEX2) GID1: GID2: (HEX)
SMSP: +6807790000 (ASC) MSISDN: ☐ Inc (ASC)
SPN: oai (ASC)
ECC: ...
Algorithm: ☒ Milenage ☐ XOR R&C Para Other files Same with GSM