



# Deployment of 5G Non-Standalone networks based on OpenAirInterface

**Pablo Picazo Martínez**

**Tutor: Jose Francisco Monserrat del Río**

**Cotutor: Sergio Pastor Tur**

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2020-21

Valencia, 10-Mar-21



## Resumen

Las dudas sobre la integridad y la seguridad de las comunicaciones móviles han impulsado el desarrollo de soluciones de código abierto para el despliegue de redes móviles. Una de estas alternativas se conoce como la alianza Open Air Interface, de la que es miembro la UPV. En este Trabajo Final de Grado se despliega una solución de red 5G Non-Standalone. Se utilizará equipamiento de última generación y se realizarán los cambios pertinentes al código abierto de Open Air Interface para optimizar su funcionamiento. Esto incluye la integración dentro de un USRP de propósito general del código necesario para el correcto funcionamiento de la red privada 5G. Una vez esta red esté desplegada se realizarán medidas de calidad de cobertura utilizando software especializado y un terminal móvil comercial.

## Resum

Els dubtes sobre la integritat i la seguretat de les comunicacions mòbils han impulsat el desenvolupament de solucions de codi obert per al desplegament de xarxes mòbils. Una d'aquestes alternatives es coneix com l'aliança Open Air Interface, de la qual és membre la UPV. En aquest Treball Final de Grau es desplega una solució de xarxa 5G Non-Standalone. S'utilitzarà equipament d'última generació i es realitzaran els canvis pertinents al codi obert d'Open Air Interface per optimitzar el seu funcionament. Això inclou la integració dins d'un USRP de propòsit general del codi necessari pel correcte funcionament de la xarxa privada 5G. Un cop aquesta xarxa estigui desplegada es realitzaran mesures de qualitat de cobertura utilitzant programari especialitzat i un terminal mòbil comercial.

## Abstract

Doubts about the integrity and security of mobile communications have prompted the development of open source solutions for the deployment of mobile networks. One of these alternatives is known as the Open Air Interface alliance, of which the UPV is a member. In this Final Degree Project, it is deployed a 5G Non-Standalone network solution. Last generation equipment will be used, and the pertinent changes will be made to the Open Air Interface open source to optimize its operation. This includes the integration into a general-purpose USRP of the code necessary to ensure the proper operation of a private 5G network. Once this network is deployed, coverage quality measurements will be carried out using specialized software and a commercial mobile terminal.



## Index

Chapter 1.	Introduction .....	6
1.1	Context .....	6
1.2	Motivation .....	8
1.3	Objectives.....	9
1.4	Outline.....	9
Chapter 2.	Technical foundations .....	10
2.1	Traditional RAN vs Open-RAN.....	10
2.2	Core Network Evolution .....	13
2.3	Deployment Scenarios of 5G Networks.....	15
2.4	Software Radio Systems SRS .....	16
2.5	Open Air Interface (OAI).....	17
2.6	Amarisoft.....	19
2.7	System setup.....	21
2.7.1	Minimum requirements .....	21
2.7.2	Working environment specifications.....	21
2.7.3	Previous installations.....	21
2.8	Universal Software Radio Peripheral (USRP) .....	22
2.8.1	USRP N321 Content .....	22
2.8.2	USRP N321 Connection.....	23
Chapter 3.	Open Air Interface 5G NSA: Deployment .....	26
3.1	OAI Build.....	26
3.2	Configuration Files.....	27
3.2.1	OAI EPC LTE .....	27
3.2.2	OAI EPC NR .....	28
3.2.3	FOKUS EPC LTE .....	29
3.2.4	FOKUS EPC NR.....	31
3.3	gNB and eNB execution.....	32
3.3.1	OAI EPC .....	32
3.3.2	FOKUS EPC .....	33
3.3.3	eNB execution .....	34
3.3.4	gNB execution.....	34
3.3.5	5G Interworking with LTE.....	35
3.4	UE attach demonstration .....	38



Chapter 4.	Open Air Interface 5G NSA: Performance Testing .....	39
4.1	OAI NR spectrum used .....	39
4.2	COTS UE log tracking .....	40
4.3	COTS UE coverage analysis .....	41
4.3.1	LTE RSRP .....	43
4.3.2	LTE RSRQ .....	44
4.3.3	LTE SINR .....	45
4.3.4	NR RSRP .....	46
4.3.5	NR RSRQ .....	47
4.3.6	NR SINR .....	48
Chapter 5.	Open Air Interface USRP Upgrade .....	49
5.1	Theoretical throughput .....	49
5.2	UHD-FPGA of Ettus USRPs .....	50
5.3	UHD upgrade .....	51
5.3.1	USRP UHD 3.15 .....	52
5.3.2	Linux Host UHD 3.15 .....	54
5.4	Sample rate and master clock changes .....	56
Chapter 6.	Conclusions, Limitations and Future Work .....	60
References	.....	62



## Acronym list

3G 3<sup>rd</sup> Generation Networks

3GPP 3rd Generation Partnership Project

4G 4<sup>th</sup> Generation Networks

ADSL Asymmetric Digital Subscriber Line

AI Artificial Intelligence

AMF Access and Mobility Management Function

ARM Acorn RISC Machine (Processor architecture)

AUSF Authentication Server Function

BTSs Base Transceiver Stations

COTS Commercial Off-The-shelf

CP Control Plane

CS Circuit-Switched

DDR4 Double Data Rate type four Synchronous Dynamic Random-Access Memory (4<sup>th</sup> Gen)

DHCP Dynamic Host Configuration Protocol

DU Data Unit

DW Down Link

eNB Enhanced Node B

EPC Evolved Packet Core

FAPI Functional Application Platform Interface

FDD Frequency Division Duplex

FPGA Field Programmable Gate Array

FR Frequency Range

gNB Greater Node B

GPIO General Purpose Input/Output

GSM Global System for Mobile

HSS Home Subscriber Service

IoT Internet of Things

IP Internet Protocol

IQ Quadrature Sample

IMSI International Mobile Subscriber Identify

iTEAM Instituto de Telecomunicaciones y Aplicaciones Multimedia

LBT Listen Before Talk

LTE Long Term Evolution

MCC Mobile Country Code

MCG Mobile Communications Group



MIMO Multiple in Multiple Out  
MME Mobility Management entity  
MNC Mobile Network Code  
MSISDN Mobile Station Integrated Services Digital Network  
NRF Network Repository Function  
NSA Non Stand Alone  
OAI Open Air Interface  
O-DU ORAN Distributed Unit  
OFDM Orthogonal Frequency Division Multiplexing  
O-RU ORAN Radio Unit  
PDU Protocol Data Unit  
P-GW Packed Data Network Gateway  
PTP GM Precision Time Protocol Grand Master  
RAN radio access network  
RB Resource Block  
RE Resource Elements  
RF Radio Frequency  
RIC Radio Intelligent Controller  
RNC Radio Network Controller  
RRC Radio Resource Control  
RRM Radio Resource Management  
RS Reference Signal  
RSRP Reference Signal Receive Power  
RSRQ Reference Signal Received Quality  
RSSI Received Signal Strength Indicator  
RU Radio Unit  
SA Stand Alone  
SCS Sub Carrier Spacing  
SDN Software Defined Network  
SDR Software Defined Radio  
SCG Secondary Cell Group  
SFP Small Form-factor Pluggable transceptor  
S-GW Service Gateway  
SINR Signal to Interference plus Noise Ratio  
SMO Service management & orchestration  
SMS Short Message Services  
SSD Solid State Drive



SSH Secure Shell

TAC Tracking Area Code

TDD Time Division Duplex

TIP Telecom Infra Project

UDM Unified Data Management Function

UHD USRP Hardware Driver

UL Up Link

UP User Plane

UPF User Plane Function

USRP Universal Software Radio Peripheral

VHDL Very High-Speed Integrated Circuit Hardware Description Language

## Chapter 1. Introduction

### 1.1 Context

Technological advances have always been important for the well-being of society and have made life easier. As communications is an inherent characteristic of human, and sometimes face-to-face communication is not possible, telecommunications join a very important role. In addition, with the entrance of COVID in our lives, this role has become even more important. It would not be possible to talk to our parents, assist to class, work, or even buy some clothes at shops without the help of telecommunications.

Since the 80s we have cell phones in the market, that allowed us to be connected to every single part of the world at any time. However, that phone has evolved a lot from the first terminal until 2021, and now, people is not satisfied with simply being able to send our voice to other part of the world. 1G only allowed voice transmissions and there was a need for other services to be satisfied. Those were covered with rise of 2G, in the 90s, which introduced SMS (Short Message Services) using GSM (Global System for Mobile) and was able to send text between two mobile phones.

The appearance of internet in our lives, and its extension to ordinary people was in the end of the century. Of course, this was a challenge to mobile networks as both data and voice service had to be in the same network. 3GPP was the name of the organization that standardized all the coming 3G technologies, which are still used in many applications nowadays. 3G was able to send data and voice in the same network.

However, the most important leap in cellular networks was the introduction of 4G technology. With 4G, fixed and mobile broadband networks are integrated for the first time and the 3G system was simplified with technologies as LTE (Long Term Evolution). 4G has an end-to-end Internet Protocol (IP) -based network system, which causes the reduction of costs in the deployment, the increase of the quality of service and reaches better rates of data transmission. 4G networks can operate with a throughput of 20 Mbps up to 50 Mbps, with maximums of 150 Mbps. This means it is faster than ADSL (Asymmetric Digital Subscriber Line) and approaches optic fiber levels [1]

Notwithstanding 50 Mbps can seem enough for some people, the appearance of new applications such as self-driving vehicles, high speed gaming, massive IoT or industry 4.0, require a more powerful network, with higher throughput and lower latency. The rise of 5G will join an important role to solve this problem. **Figure 1** shows mobile network evolution since the end of 80s, with the introduction of 1G.

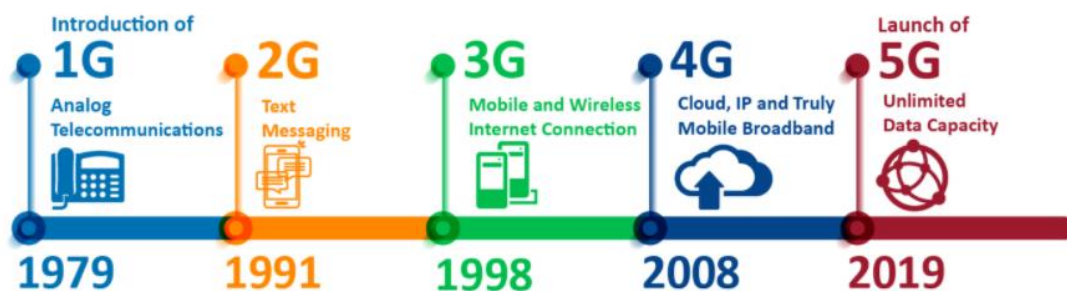


Figure 1: Mobile network evolution [2]

Mobile communication networks rely on a wired network, between the core network and the radio transceivers. This network is called radio access network (RAN) and provides functions to coordinate radio access among multiple radio stations, each of service multiple mobile terminals.

A new architecture has come with 5G technology. OpenRAN is based on a distributed service model with IP network routing as the transport fabric underlying it [3]. The goal of OpenRAN is



to reach an open, flexible, distributed and scalable radio access network architecture. It would be open, because interfaces that in the past were closed and proprietary, now are standardized. It would be flexible because many implementations are admitted, depending on the resources. It would be distributed, because there is no need to have all the network elements in one physical place, indeed, some of them could be in the cloud. It would be scalable, because the architecture defines interfaces that allows the core to be designed independently, making the RAN grow without having core related issues. The architecture would not require changes in radio link protocols and radio link IP based protocol would not be necessary.

By deploying an OpenRAN based network, public operators could get independent access networks, giving the core network more freedom. They also could leverage their core, service-based network and mobility support through a variety of technologies, making their market for their services larger.

OpenRAN aims to reach cost-effective use of their spectrum assets by selection the radio protocol that is most appropriate in each situation. Duplication wire-line infrastructure for different radio technologies is not necessary. Other cost-efficiency way in which OpenRAN joins an important role is separation of control and bearer plane functions. Control plane could be implemented on all-purpose server platforms while the real time bearer plane could use highly specialized hardware.

OpenRAN also increases reliability by removing points of failure. Old RAN functions that were clustered into monolithic nodes, now are distributed through the OpenRAN. This improves the potential for redundancy, because the cost of multiple installations for each RAN element is reduced. This provides scalability, as the bearer plane and transport plane can scale independently, raising deployment flexibility.

The admission of flexible deployment scenarios makes OpenRAN a good choice for operators to select a deployment that matches the available backhaul network resources. Past RAN architectures based in a star topology were optimized for cases where not the best backhaul network resources were available. Operators now could deploy either a star or a mesh topology, where richer bandwidth resources are available, since OpenRAN is compatible with both. The ability to handle multiple radio link protocols with a single radio network will facilitate interoperability between them. Some functions as mobile management, quality of service or security can move into the IP transport layer.

Finally, OpenRAN could also allow to develop more flexible business models that evolve around provision of wireless Internet access. The separation between core and RAN allows business that provide core and RAN to be independent. For example, a public operator could separate their core unit in order to make business from other access vendors [3]. **Figure 2** shows a schematic of what is called an OpenRAN distribution.



Figure 2: OpenRAN distribution [4]

## 1.2 Motivation

If the need of 5G was not clear to society before COVID, it is undeniable nowadays that mobile communications need to reach the next step. 5G has amazing applications, as remote surgery or self-driving cars, between lots of other useful uses. Other uses are cloud computation, which would bring gaming to the next level, as 5G can offer an excellent gaming experience without the need of a powerful device, as all the computational part will be cloud based. The main impact, as explained, will not just be having a faster network for the users to surf the internet. The goal is to reach new fields and to improve the current ones.

Commercialization potential is huge, since enterprises as Huawei are selling their own devices to deploy a network for private uses [5]. Other 5G use case is the European project 5GCroCo, with a total budget of 17 million Euros, partially funded by the European Commission, whose goal is to validate 5G technologies in the Metz-Merzig-Luxembourg cross-border corridor. 5GCroCo validation is focusing on tele-operated driving, map generation and distribution for automated vehicles at high speed, and anticipated collision avoidance, to make cars safe in the face of unforeseen [6].

The demand of private networks, customized and flexible, is raising. As a result, the concept of OpenRAN is necessary to cover this need. Different service providers will be able to use open source software and standardized hardware in order to offer their service. Cloud based interfaces will also join an important role, since RAN high layers could run in the cloud.

The OpenRAN architecture will accomplish the expectations for 5G technology. It will also allow new service providers to enter the market. This will benefit the flexibility and security of the network as well as the competitiveness of the market. Evolution will also be more constant since the service providers will release new standards and solutions in order to be competitive and stay updated.

ORAN alliance was created in order to develop OpenRAN projects. Its effort is making this new concept of RAN gets more importance. ORAN's contribution is speeding up this process in order to stablish OpenRAN solutions. Events as Mobile World Congress Barcelona, in April 2020, made ORAN show their potential. Many vendors as Intel, Nokia or Lenovo participated in some ORAN demos [7].

OpenAirInterface, OAI, raised by Eureka, is a non-profit consortium that research open source RAN. OAI is working on an open source EPC (Evolved packet core, 4G core) and 5G Core Network. It is also developing a software defined network (SDN) access point for 4G/5G (eNB and gNB respectively). Their code resides in Gitlab, where a user can download it to test it with its own equipment. Users can also make contributions in order to improve the code or test it in other circumstances [8]

There are also private solutions as the ones provided by Amarisoft, a telecommunication company that brings 4G/5G ready to use equipment. They have products to deploy 5G and LTE networks with an all in one structure. Both core network and RAN are in the same physical device. This is useful to test the network performance and study its characteristics.

All these organizations are in a growth phase. This is possible due to the implication of 3GPP, Telecom Infra Project (TIP) or the biggest providers as Cisco or Ericsson, among others. Each of these enterprises are helping projects like ORAN or OAI to gain force and to generate interest among the telecommunications industry.



### 1.3 Objectives

This final degree project aims at testing Open Air Interface code, upgrading its current state. It also shows the current state of OAI communication with ORAN. This is done inside the project Valencia Campus 5G, at the Institute of Telecommunications and Multimedia Applications (iTEAM), at the Universitat Politècnica de València (UPV).

In order to develop it, the specific objectives are determined below:

- To study of the architecture of 4G and 5G networks, the evolution from 4G to 5G and major changes.
- To configure and deploy a OAI 5G Non Standalone (NSA) system, using Universal Software Radio Peripheral (USRP) as radiant element with software-defined radio.
- To perform measurements of the OAI based 5G NSA network using ROMES, a software that allows tracking 4G/5G networks.
- To improve the current state of OAI by including the last generation USRP N321 in the open source code and testing its connectivity.

### 1.4 Outline

This section pretends to be a guideline to the reader, exposing the contents of the memory and its organization, in order to facilitate the search of information through it. The chapters of this dissertation are:

Chapter 2. Technical foundations. This chapter will first explain the evolution of the RAN network, introducing Open RAN architecture. Next, mobile network core evolution, from 4G to 5G core evolution. Once both RAN and core are understood, deployment scenarios of 5G networks will be shown. Then, private and open source 5G solutions in the market at the moment will be studied. Last, all the specifications needed to replicate the project will be explained.

Chapter 3. Open Air Interface 4G/5G NSA Deployment. This chapter includes the steps that need to be follow in order to replicate the 5G NSA network deployed in the project. This consists of OAI build, configuration files used, core commands and eNB+gNB commands. It also features OAI secondary node addition procedure and an attach from a smartphone to the network.

Chapter 4. Open Air Interface 5G NSA: use case. In this chapter, a use case of the network will be explained. This includes checking the spectrum used and designing a measurement route in the MCG lab. Parameters of the network will be tracked using ROMES software and a Samsung A90 in order to probe the quality of the network.

Chapter 5. Open Air Interface USRP upgrade. This chapter features the upgrade performed to the USRP (Universal Software Radio Peripheral) used by Open Air Interface. This includes an UHD (USRP Hardware Driver) upgrade of the new USRP N321 and the adaptations to the Linux machine drivers and OAI code in order to run OAI gNB in the N321.

## Chapter 2. Technical foundations

### 2.1 Traditional RAN vs Open-RAN

3G traditional networks had a star-based topology with a centralized architecture. The radio network controller (RNC) is connected with point-to-point links to each radio base transceiver stations (BTSs). Those stations handle the connectivity for the radio network for a particular region or cell. RNCs are also interconnected themselves to allow mobile nodes roaming between geographical areas. The core network does not need to participate there since RNCs are interconnected. As a mobile node roams between cells that are controlled by an interconnected RNC, the RAN arranges for voice or data sessions to be handed off between those cells. **Figure 3** acts as a schematic of the 3G RAN Architecture [4].

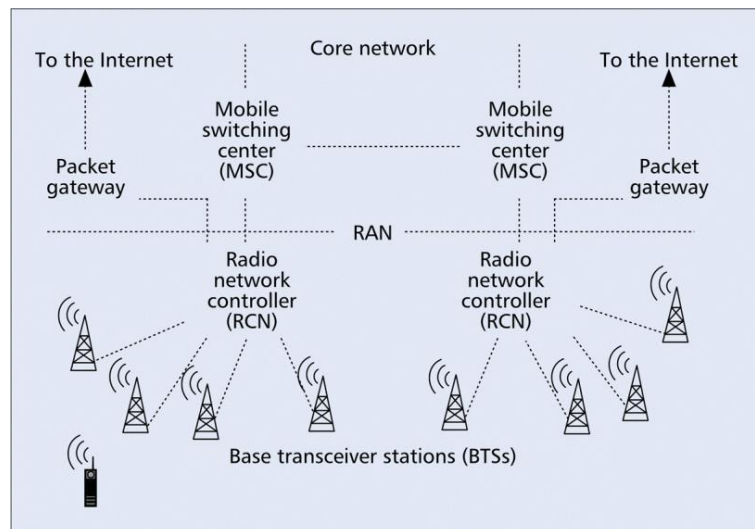


Figure 3: 3rd Generation RAN Architecture [4]

In the 3<sup>rd</sup> generation systems (3G), the RNC is connected to a CS (circuit-switched) core for voice and data. It is also connected to an access gateway packet-switched based, that allowed direct access to the Internet. When a mobile node roamed beyond the interconnected RNCs a handoff was necessary. Although the performance of the 3G architecture has served well, there were potential problems:

- If one RNC fails, the entire geographical region that handled many cells will leave them without service.
- Upgrading RAN to handle more terminals was not possible sometimes. If the number of cells extends the expansion limit of the RNC, a new RNC needs to be deployed, at great expense, even only an upgrade in bearer capacity is required.
- Each radio link protocol has its own radio layer protocol, that has been implemented by the RNC to get control of the radio link. Some functions as mobility management could be handled by protocols as routing. The management of the spectrum is disturbed by the lack of common functions.

Traditional RAN architecture evolved through generations, introducing in 4G/5G functional splits, dynamic and software defined RAN, allowing deployment flexibility. Some 5G requirements, as ultra-low latency and high throughput require a flexible topology, that was not implemented in 3G networks. This will be reached with the split of RAN functions, separating user plane (UP) and control plane (CP) in higher layers of the network.

Radio Resource Management (RRM) has also been improved to reach connectivity across all access network technologies, antenna points and sites. This is achieved by applying carrier aggregation, dual connectivity, beamforming, or MIMO.

The capability to reconfigure and rescale the network, that was a problem in 3G as explained before, has been improved. This has been possible thanks to software defined RAN, separating out logical nodes which are suitable for virtualization. Functions that require specialized hardware are dynamically reconfigurable.

The deployment flexibility enables an operator to configure the RAN with spectrum efficiency and service performance, regardless the topology of the site or the transport network and spectrum scenario.

To achieve this, a correct split into logical nodes is needed. It can be done by deploying each node type in the most appropriate site given the service requirements and the physical topology. **Figure 4** shows 4G split into a radio unit (RU) and a data unit (DU) [9]

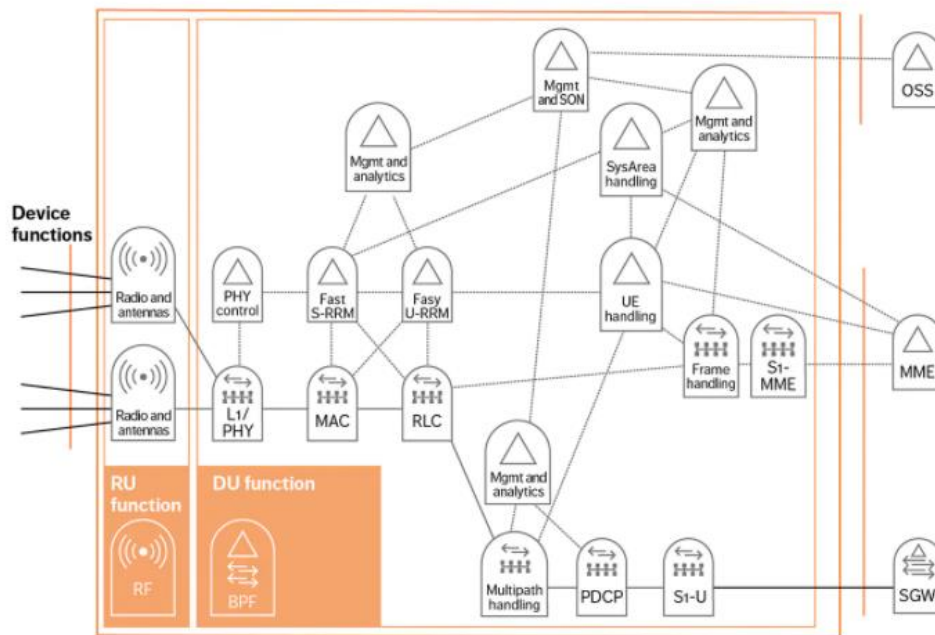


Figure 4: 5G RU/DU RAN Split [9]

This figure illustrates the logical RAN architecture. In the downlink, PDUs enter the RAN over S1-U and are delivered to devices over the radio interface. A single UE can receive and send data through different radio channels, for example LTE and NR ones. The MAC function is the anchor for carrier aggregation and handles multi-beam transmissions. In the uplink L1/PHY function performs soft combining, MAC aggregates data in carrier aggregation and multipath handling aggregates data from dual connectivity uplink data streams.

Open-RAN goes one step further. Traditionally the RAN systems are one provider property, what means Ericsson, Huawei or Nokia own the entire RAN network and implement the whole solution. A network was not able to handle two providers of different technologies due to a big drop of performance of itself. Even it was possible, all the problems related made operators deploy one provider network. For example, Telefonica decided to use Huawei equipment in order to deploy its 5G network.

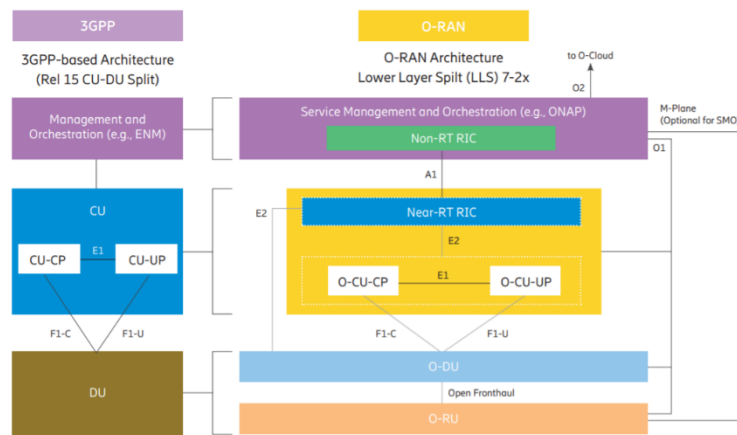
Open-RAN is born with the goal of building networks using different vendors' equipment. It takes profit of the split and virtualization that has been performed in last generation RAN networks to establish standardized interfaces. Those interfaces should be able to interconnect different solutions in a flexible, smart, and energetically efficient way.

The alliance ORAN is compromised with an evolution for the radio access network, guiding networks evolution on a smart and open way. Characteristics as backend AI modules rocket automatic learning systems, whose empower this network capacitance [10].



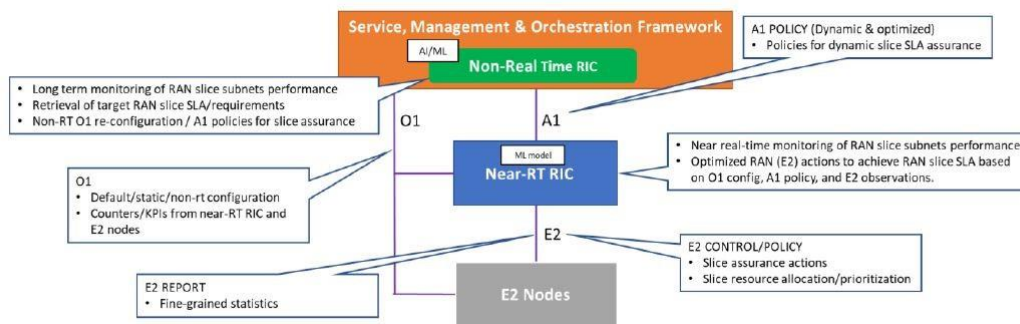
Virtualized elements with open standardized interfaces are the key aspects in the designs of ORAN, inside the Open-RAN principles. Open source technology and smart elements are some of the features included in the next generation networks. To build an agile and profitable network, accessible models are required. Open interfaces are essential in order to allow providers and operators to introduce their own services. Customization of the network for the final mobile operator is a feature that could be possible thanks to Open-RAN characteristics.

Networks need to be autonomous, with 5G arrival, complexity and exigence makes traditional RANs hard to expand, deploy or optimize for a single provider. ORAN includes AI in all its architectural layers. It is applied not only to components but also to the network, allowing dynamic assignation of resources. Taking profit of this automatization of operative functions in order to reduce costs is one of the main characteristics. **Figure 5** shows a comparison between 3GPP Traditional RAN (Release 15) and ORAN Specifications.



**Figure 5: Traditional RAN vs Open-RAN [10]**

3GPP focuses on global specifications, interface freedom among RAN elements, functions or core network is essential in order to break down the network and facilitate access to additional providers. ORAN includes interfaces as A1, E2, O1 and O2 so as Open Fronthaul in order to split and intercommunicate functions, as shown in **Figure 6**. It also adds SMO (Service management & orchestration) and RIC (Radio Intelligent Controller) functions [10].



**Figure 6: ORAN Slicing [11]**

The SMO is responsible of collecting data from DU and CU units of inferior layers through O1 interface. The real time RIC monitors the RAN performance in order to optimize it. RIC predicts network saturation and transport this data through A1 interface. xAPPs in Near-RT RIC are in charge of processing this to apply a solution to the DU and CU, using E2 interface [11].

ORAN architecture is expanding rapidly among the implemented RAN solutions even though its incorporation is progressive. Thanks to the support of both service providers and operators, Open-

RAN networks serve 21.8% of mobile phone subscribers all around the world. It is expected that in 2024 nearly half of the subscribers use Open-RAN based networks.

## 2.2 Core Network Evolution

To understand the transition from a 4G network to a 5G we need to push out into their core architecture. The core of a 4G network is composed by four parts, which are specified in 3GPP. Without these components the basic functionality of an LTE network could not be performed. The communication interfaces between each part are split into signaling bearers (S6a, S11 and S1-c) and user traffic (SGi, S5, S1u). **Figure 7** shows the 4G core schematic, with its components and interfaces.

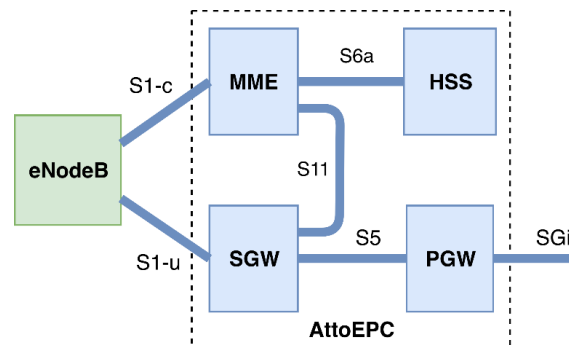


Figure 7: 4G Core Schematic [12]

The MME, Mobility Management Entity, is the element that centralizes LTE signalling of the control plane, related with the user sessions and subscriptions. The MME implements security procedures through the negotiation of ciphering algorithms and identity protection to authenticate the user. It also manages the session between the user (UE) and the network, being the responsible of the environment of the data packets. In addition, it regulates parameters as the quality of service (QoS). The interfaces used by the MME are S1-c to connect to each eNB (the radio part of the network) and S6a to reach the HSS. It also has an interface called S11 to the SGW, being the MME in charge of choosing the Gateways associated to the user.

The HSS, Home Subscriber Service, is a database that is stored in a single node to give service to the control entities of the core network that manage the data traffic of users. The server stores the data of each subscribed in the network. Main parameters are IMSI (International Mobile Subscriber Identify) and MSISDN (Mobile Station Integrated Services Digital Network). It also storages information related to the hired services of each subscriber and to the QoS offered. It is in charge of the authentication of the user and generates the security information to control the access to the network. The interface used to connect the MME with the HSS is the S6a.

The S-GW, Service Gateway, is one of the nodes that manages the user's data traffic. It is the point of interconnection between the eNB radio access and the packet commutation plane, using the interface S1-U to do the link. The S-GW also provides the control of UE mobility between the eNB, being responsible of the handover, which occurs when a user is moving and gets better connection from other radio node. In 4G the user has a list of active eNBs and can transmit information through all of them but when the UE moves, maybe other eNB offers better coverage, then the system procedures to handover, what means disconnect the UE from the worse eNB and reconnect it to a better one. This improves QoS and allows the user to move freely while keeping his 4G connection.

The P-GW, Packed Data Network Gateway, is as the S-GW, an interconnection point with external IP. It manages packet forwarding from the core network to other networks, as could be Internet. The interface used as exit to internet is the SGi. The functions of the P-GW also cover IP assigning and resources management for the users. The interface used to reach the S-GW is

S5, if the radio network is from the same operator or S8 if the user is connected to another operator, also known as roaming [13].

The next generation core, known as 5GC, has some similarities and changes regarding the 4G one. The user data packet processing and the RAN integration is similar to the last generation. However, the signalling network has been defined completely different to the 4G core. The basic structure of the 5GC includes [14]:

The AMF, Access and Mobility Management Function, that interacts with the RAN and the UE through encrypted signalling across the interfaces N1 and N2. Most part of data flux comes across the AFM. It allows devices to register, authenticate and move among the network jails. To perform these functions, it queries other network functions and forwards signalling data to the appropriate entity. Other function that the AMF implements is to activate idle devices when necessary.

The SMF, Session Management Function, that is in charge of the administration of the user's sessions. This includes to stablish, modify and free sessions, as so as IP assignments to each device. It uses the AMF to indirectly communicate with the users, forwarding processed signalling messages through the interface N11. In addition, the SMF interacts with other network functions, as user plane functions (UPF), using interface N4.

The UPF, User Plane Function, that has as main role to process and forward user plane traffic and it is under SMF control. It uses interface N6 to connect the core with foreign networks that have external IP. In addition, the UPF performs data processing for forwarded data. On one hand, it generates reports of traffic use for the SMF. On the other hand, it can analyse the data packet content and make that constitute other element in the policy decision. Traffic redirection or throughput imposition as well as device activation are other policies that UPF covers.

The UDM, Unified Data Management Function, which represents the updated database of the mobile subscribers. It generates authentication credentials used to connect to the network. It also authorizes the access to specific users according to the available information in the database. This allows, for example, applying different access rules to roaming users. The UDM also executes functions requested by the AMF, using the interface N8 to stablish communication. The SMF can also communicate with the UDM through the interface N10.

The AUSF, Authentication Server Function, which has a limited but important content. It is the service provider of device authentication, using the UDM credentials that reach the AUSF through the N13 interface.

The NRF, Network Repository Function, the core element in charge of registering the service providing network functions. It also allows communication between the service providing functions and the service consuming functions. **Figure 8** shows the schematic of a 5G core explained.

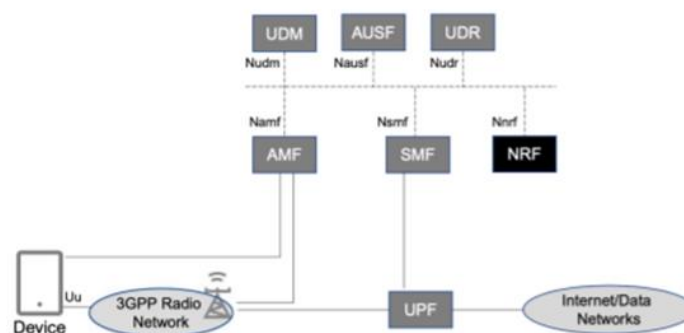


Figure 8: 5G core schematic [14]



### 2.3 Deployment Scenarios of 5G Networks

The radio part of the network has to carry the signal from the base station to the UE. 4G name for the node is eNB (evolved node B). 5G name is gNB (greater node B). There are two main ways to deploy a 5G based network.

The cheapest one would be to use the 4G core and just add the gNB to the 4G setup. This will increase the throughput of the network since data plane will flow through the gNB. However, latency will not improve, as the control plane will not be improved. Latency in a 4G network is about 15-20ms. The fact that 4G networks are not amortized makes companies deploy those NSA 5G networks. Just adding one gNB in the 4G base station will turn that station into a 5G NSA node.

The 5G full potential will be unleashed with the appearance of SA networks. This requires bigger investment, but the reward is an extremely low latency (less than 1ms) network with up to 1Gbps throughput. Low latencies are needed to perform remote surgeries or to track self-driving cars. The importance of a fast response of the network raises since a delay could be a risk for a human life. **Figure 9** shows the two main configurations indicated above:

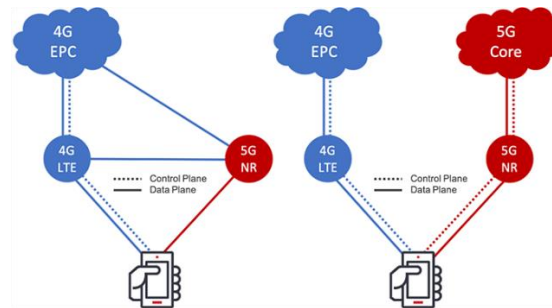


Figure 9: NSA vs SA configurations [15]

As a result, the main difference from the 4G network to the 5G NSA network is that the radio part will have both gNB and eNB. They can communicate each other using the X2 interface. As explained, this will provide very high speed to the user with relatively low investment, just by adding a new radio node with different radio features. **Figure 10** shows basic interfaces used for both NSA and SA modes:

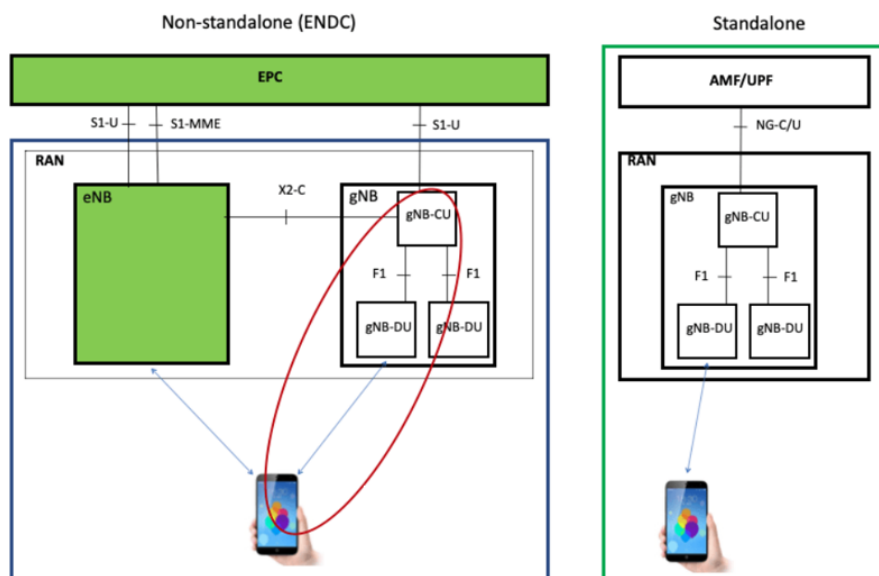


Figure 10: NSA vs SA Architecture [16]

Apart from mobile known operators, there are entities that are trying to develop an open-source code for 5G networks. Open Air Interface (OAI) has developed a 4G network core and radio access network. It is on its way to develop a fully functional 5G NSA network using USRP (a kind of SDR, software defined radio) as their eNB/gNB radio access points. Their code resides in a git server and everybody can download it to test the network and contribute to the code.

This would allow a user to have his own 5G network and to configure the parameters desired to deploy a private service of 5G. There are also other solutions (non-open source) as Amarisoft to test 5G networks. Amarisoft provides an all-in-one solution to support functional performance testing of 4G eNB and 5G gNB as well as the 4G/5G core networks. It can simulate up to 1000 users and perform MIMO (Multiple-input Multiple-output), what means that more than one antenna to transmit or receive is used [17].

## 2.4 Software Radio Systems SRS

Software Radio Systems is an Irish company that provides LTE open solutions, with customized system design and development [18]. A software radio has its components like modulators/demodulators, filters or detectors implemented in a software upon general-purpose platform. Historically those elements were implemented in hardware, but SRS provides high performance software solutions to replace that hardware with software. This allows a range of wireless technologies as LTE, to be low cost and easy modifiable/updatable. In addition, performance can be upgraded by improving the software solution with new technologies.

SRS provides L1, L2, L3 protocol stacks for UE and eNB. Their library, which is hosted in github, supports a high range of software defined radio platforms. They also include solutions for network testing and equipment diagnostic in addition to LTE wide-band scanners. The code is adaptable to applications and environments like machine-to-machine communication, airborne or high-speed deployments.

In the area of machine-to-machine communications, IoT, SRS develops customized low cost, low energy waveforms and network protocols. The flexibility offered by the software-defined radio supports a diversity of applications in this field, allowing using a wide range of frequency bands [20].

SRSLTE is a high-performance library for SDR applications. It supports both eNB and UE, with minimal inter-module or external dependencies. It is implemented in C and available under commercial and open-source licences. The features included are:

- LTE Release 15
- FDD and TDD
- Bandwidths 1.4,3,5,10,20 MHz
- Transmission modes 1-4
- Cell search and sync procedure for the UE
- All DL channels (UE+eNB): PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH
- All UE channels (UE): PRACH, PUSCH, PUCCH, SRS
- Frequency-based ZF and MMSE equalizers
- Turbo decoder in Intel and C
- Matlab and Octave Mex library generation
- UE receiver tested against commercial networks as Telefonica or Eircom (Ireland)
- eNB tested up to 150 Mbps DL in MIMO with commercial UEs (Nexus 5 and Moto G4)
- eNB standard S1AP and GTP-U interfaces to the Core Network

Their SRSENB currently supports Ettus Research USRP, Epiq Solutions Sidekiq and Nuand bladeRF frontends. **Figure 11** shows all SRS LTE characteristics, including Core, physical channels, UE processes and example applications.

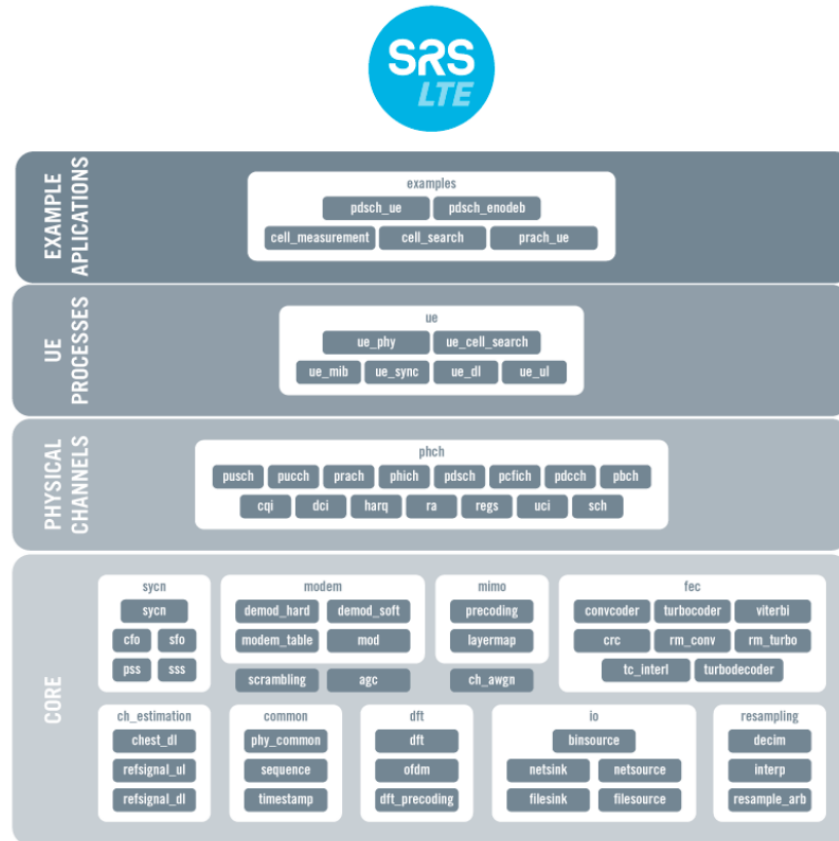


Figure 11: SRS LTE Structure [18]

## 2.5 Open Air Interface (OAI)

Open Air Interface is an open source platform that has been developed by the French research institute Eurecomm. OAI's goal is to create tools to implement an open source solution for both LTE and 5G services. It will have the defined protocols by 3GPP, and its software can be executed in general processors as X86, and its mainly programmed using C, in a Linux environment. OAI has software-based network functionalities, reducing the implementation cost and increasing the flexibility of the deployment, allowing lab scenarios to test its solutions.

The code is hosted in gitlab. It has a full functional 4G network, including EPC and eNB. 5G network is in development phase and resides in the develop git branch. The 4G physical layer includes the following characteristics [19]:

- FDD and TDD, with bandwidth of 5,10 and 20 MHz
- SISO and MIMO
- Downlink channels: PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH, PMCH
- Uplink channels: PRACH, PUSCH, PUCCH, SRS, DRS
- HARQ for both UL and DL
- Turbo decoding
- 64 QAM on DL, 16 QAM on UL

5G branch includes the following features [20]:

- Polar codes
- Release 15 implementation
- Downlink channels: NR-PSS, NR-SSS, NR-PBCH, NR-PDCCH, NR-PDSCH
- Uplink channels: NR-PSS, NR-SSS, NR-PBCH, NR-PDCCH, NR-PDSCH

- X2-AP interface (interconnection of eNB and gNB for NSA)

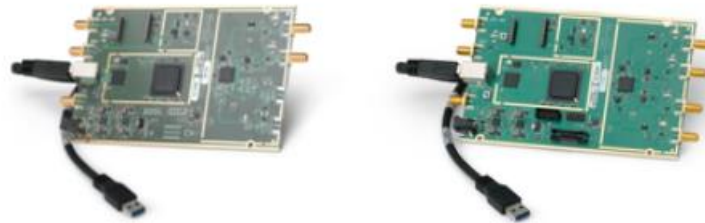
To create the eNB/gNB, OAI makes use of Universal Software Radio Peripheral (USRP). The SDR (Software Defined Radio) allows processes (that traditionally were performed by hardware) to be software based. The USRP has SDR cards with FPGAs (Field Programmable Gate Array) that are coded using VHDL (Very High-Speed Integrated Circuit Hardware Description Language).

OAI makes use of Ettus USRP models B200/B210 that are connected to the Linux machine using USB3.0. The main features of the USRPs used is summarized [21]:

B200: 56 MHz bandwidth, full duplex, SISO, USB3.0

B210: 56 MHz bandwidth, full duplex, MIMO 2X2, USB3.0

**Figure 12** shows B series, **Figure 13** shows X/N310 and **Figure 14** shows last generation USRP N321.



**Figure 12: Ettus USRP B200/B210 [21]**

However, the USRPs used in 5G OAI network need to have bigger bandwidths in order to accomplish 5G requirements. 5G OAI can work with B210 and X310 but Ettus USRP “N” series is the one that OAI highly recommends for NR-5G testing, using N310. They have the following features [22], [23]:

X310: 160 MHz bandwidth, MIMO 2X2, SFP+ 1/10Gb

N310: 100 MHz bandwidth, MIMO 4X4, SFP+ 1/10Gb



**Figure 13: Ettus USRP X310/N310 [22], [23]**

Nevertheless “N” series has upgraded versions that could perform better with OAI, those are N320 and N321 [23]:

N320/321: 200 MHz bandwidth, MIMO 4X4, SFP+ and QSFP+ (Aurora)



Figure 14: Ettus USRP N321 [23]

OAI has not implemented this in its code yet, but this USRP could increase the throughput since it has some features upgraded.

## 2.6 Amarisoft

Amarisoft is a company founded in 2012, which is dedicated to telecoms industry. They bring 4G/5G all in one solutions. This means both eNB/gNB and core are in the same device, which is Linux based [24]. Since its creation, they have implemented different services showed below in **Figure 15**:

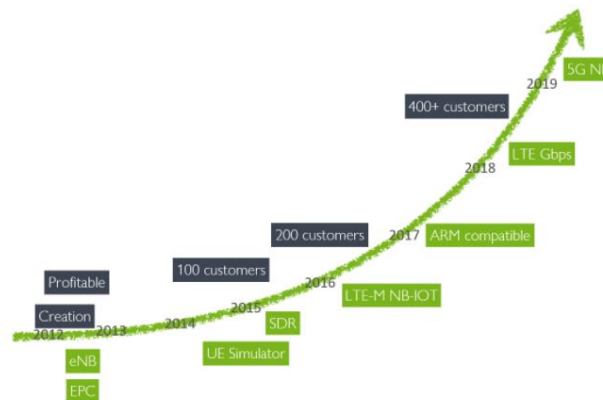


Figure 15: Amarisoft services evolution [24]

Amarisoft has over 580 customers, being a reference in LTE solutions markets, from the lab to the field. They integrate a ready-to deploy LTE and NR macro base stations and small sells. This technology offers a flexible solution and makes it easy to implement and customize the network. For private networks, their compact all in one machine have interesting tools as html logs that are helpful to debug or track the network. Its remote API is a good option for automation.

They have partners as Ettus Research, the leading supplier of SDR platforms and fabricant of the USRPs used by OAI. Other interesting partners are Valid8, who tests their services and offer validation for the telecom products and software applications. Enterprises as Paralink Networks or GleeFlex act as resellers to distribute Amarisoft products to China or Russia markets.

Regarding all in one solution, they have a wide variety to satisfy customer requirements, summarized below and shown in **Figure 16**:

AMARI Callbox mini (LTE): Up to 500 UEs with 200 Mbps on DL and 75 Mbps on UL (MIMO 2X2).

AMARI Callbox Classic (LTE + NSA NR): Up to 1000 UEs with 600 Mbps on DL and 150 Mbps on UL (MIMO 2X2).

AMARI Callbox PRO (LTE + NSA/SA NR): Up to 1000 UEs with 1200 Mbps on DL and 150 Mbps on UL (MIMO 4X4 + Handover).

AMARI Callbox Ultimate (LTE + NSA/SA NR): Up to 1000 UEs with 5000 Mbps on DL and 500 Mbps on UL (MIMO 4X4 + Handover).



Figure 16: From left to right: AMARI mini, Classic and PRO/Ultimate [24]

The Amarisoft eNodeB supports:

- LTE-A, MIMO 4X4 with 5 cells in DL and 3 in UL.
- IoT, Category 0, 1, M1, NB1, NB2.
- All FDD and TDD bands + custom frequencies.
- Multi cell support of Intra eNB, S1 and X2 handover.

The Amarisoft gNodeB supports:

- FR1 and FR2 frequency ranges.
- TDD and FDD mode.
- Up to 8x8 MIMO cell in DL
- NSA and SA
- All data subcarrier spacing combinations

The Amarisoft EPC includes:

- MME, S-GW, P-GW, HSS
- NB-IoT RAT
- All LTE QCI, dedicated bearers
- Linux stack integrated
- Access point model

The Amarisoft 5GC includes:

- AMF, AUSF, SMF, UPF
- NSA/SA support
- Multi PDU sessions and built in QoS flow setup
- Linux stack integrated
- Access point model

If no mobile device is available to test Amarisoft, they also include a UE simulator. It can simulate hundreds of 5G SA/NSA, LTE or IoT users. It supports multi-cell as well as intra-frequency, inter-frequency and inter-rat Handovers. It has an IP simulation option that allows UDP or TCP packets and VoIP testing within third party apps [24]. **Figure 17** schematizes those components below:

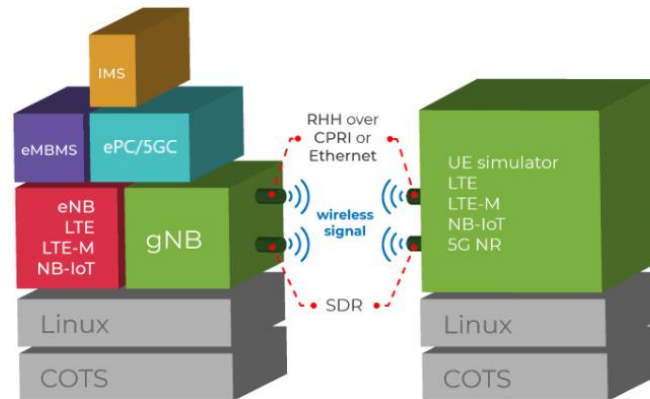


Figure 17: Amarisoft Components [24]

## 2.7 System setup

This section makes an overall description of the solution implemented. Minimum requirements to deploy an EPC and RAN are described. Working environment specifications are explained below.

### 2.7.1 Minimum requirements

Ubuntu 16.04 or 18.04 with low latency kernel option. USRP B210 or higher.

Recommended hardware requirements:

Intel Core i7 6900K (8 Cores) or higher, at least 16 GB DDR4 RAM and 256 SSD

Phyton 2.7 and above. Required for building some UHD utilities written in Phyton.

Mako 0.5 or above. CMake 2.8 or above. LibUSB 1.0 or above. Numpy 1.19.

### 2.7.2 Working environment specifications

Superserver 1029P-WTRT (Supermicro)

Ubuntu 18.04 low latency kernel.

Intel Xeon Silver 4216 32 cores @2.10 GHz

960 GB SSD

2x10 Gigabit Ethernet + 2x1 Gigabit Ethernet

6 USB 3.0

USRP B210 and N321.

### 2.7.3 Previous installations

Low latency kernel installation:

```
sudo apt-get install linux-image-lowlatency linux-headers-lowlatency
```

Install the git using the command:



```
sudo apt-get update  
sudo apt-get install subversion git
```

Clone the OAI git repository:

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
```

Checkout develop branch (OAI has a lot of git branch, develop will be used):

```
git checkout develop
```

Since Ettus USRP N321 is the one that this final project will optimize, content and how to connect it to the host computer will be explained in this chapter:

## 2.8 Universal Software Radio Peripheral (USRP)

OAI makes use of USRPs as radiant element in its RAN solution. Since the latest USRP used by OAI is N310, this subchapter aims to introduce the new USRP N321 to the community.

### 2.8.1 USRP N321 Content

- USRP N321
- DC 12V 7A adapter
- RJ45 to SFP cable + adapter
- Ethernet Cat 5, 5m cable
- USB-A to micro USB-B, 1m cable
- User guide
- Ettus Research sticker



Figure 18: N321 box content [23]

The available interfaces to connect the USRP are:

- Serie console: Low level interface used to debug.
- 1GB RJ45: Interface that connects with the ARM CPU onboard. Can be used in order to connect to the ARM via SSH.





- SPF+ Dual: supports multiple configurations for high quality streaming and low latency, depending on FPGA.
- QSPF: Supports two 10 Gb lines for high quality streaming and low latency (Ethernet extension that allows sub-ns synchronization through 1 Gb Ethernet).

## 2.8.2 USRP N321 Connection

There are three ways to establish connection with the USRP using a Linux machine.

### 2.8.2.1 Serial Console

It is possible to access the shell using a terminal through this port. Most Unix based operating systems have a tool called screen that can be used for this purpose. To install it run:

```
$ sudo apt install screen
```

Default baud rate for console is 115200. The node depends on the system used. Linux offers alternatives to test device nodes. The system needs to have a *symlinks* directory in */dev/serial/by-id*:

```
$ ls /dev/serial/by-id
usb-Digilent_Digilent_USB_Device_25163511FE00-if00-port0
usb-Digilent_Digilent_USB_Device_25163511FE00-if01-port0
usb-Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6CB5-if00-
port0
usb-Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6CB5-if01-
port0
```

Exact names will depend on the system and version. N321 will appear as 4 different devices. The ones with “*USB\_to\_UART\_Bridge\_Controller*” name offer serial requests. First one (*if00*) connects to the ARM CPU and second (*if01*) connects to the microcontroller STM32.

To connect to the ARM CPU, we use the Linux command:

```
$ sudo screen /dev/serial/by-id/usb-
Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6CB5-if00-port0
115200
```

Once the process has finished, something similar will appear:

```
OpenEmbedded test ni-n3xx-313ABDA ttyPS0
ni-n3xx-313ABDA login:
```

This is the login process. The user is root and password is blank. When password is asked, just press enter. There is access to the motherboard:

```
root@ni-n3xx-<motherboard serial #>:~#
```

Using the default configuration, the console will show kernel logs (not available if SSH used) and will allow access to the boot loader. This will facilitate kernel debugging.



To connect to the microcontroller, use the other port (*if01*) with the same command:

```
$ sudo screen /dev/serial/by-id/usb-  
Silicon_Labs_CP2105_Dual_USB_to_UART_Bridge_Controller_007F6CB5-if01-port0  
115200
```

STM32 interface is simple. Help command shows the available commands. This is a direct connection to the microcontroller and can be used to perform a hard reset without physical access to the device (emulates reset or power button).

### 2.8.2.2 SSH connection RJ45

Default connection RJ45 1GB uses DHCP in order to configurate an IP. You can either change that IP using the router configuration or create a DHCP server with Linux utility *dnsmasq*:

```
$ sudo dnsmasq -i <ETHERNET_ADAPTER_NAME> --dhcp-  
range=192.168.1.151,192.168.1.254 --except-interface=lo --bind-dynamic --no-daemon
```

Modify the value of *<ETHERNET\_ADAPTER\_NAME>* with the desired interface to create the DHCP server, then introduce the IP range. Once the IP is assigned use the *ssh* command:

```
$ ssh root@<IP>
```

Where *<IP>* is the assigned IP. This way a SSH tunnel will be created to the USRP.

```
root@ni-n3xx-<motherboard serial #>:~#
```

### 2.8.2.3 SFP connection

This interface will allow high quality streaming with low latency. To connect it, use the SFP ports in the USRP. Port 0 is 1GB and port 1 is 10GB. The steps to follow to connect with SFP0 are:

1. Configure the host ethernet adapter.

```
IP Address: 192.168.10.1  
Subnet Mask: 255.255.255.0  
Gateway: 0.0.0.0  
MTU: 1500
```

It is important to assign MTU a value of 1500 (not automatic).

2. Insert the RJ45-SFP cable with the 3 adapter to the port SFP0
3. Connect the adapter to the host terminal with Ethernet A SFP0 cable, the green led “SFP Port 0” will switch on.
4. To test the connection, use the ping command:



```
$ ping 192.168.10.2  
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.  
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=1.06 ms  
^C  
--- 192.168.10.2 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 1.065/1.065/1.065/0.000 ms
```

If 10 GB SFP1 port wants to be used, follow the same steps but note that MTU need to be 8000 instead of 1500 and IP is 192.168.20.2. Green led SF1 will switch on.

## Chapter 3. Open Air Interface 5G NSA: Deployment

### 3.1 OAI Build

This chapter will explain how to install and run OAI in order to deploy 4G and 5G network using open source tools. Please make sure to read previous installations in order to be able to follow this chapter.

Once the code is cloned from the git server, a folder named `openairinterface5g` will be created. The git will be cloned in the Linux machine described in the following path:

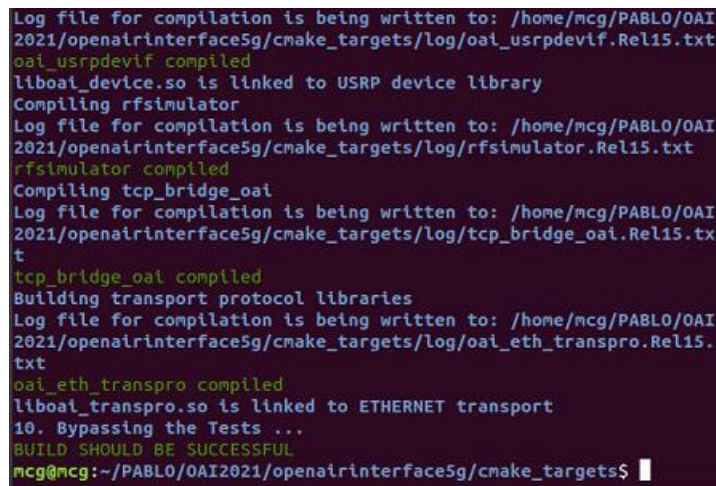
```
/home/mcg/PABLO/OAI2021/openairinterface5g/
```

Once the git is cloned, the next step is to make the build from OAI code. This will work for the USRP B210 (or higher, but not N320/N321). If N32X wants to be used, read Chapter 5 (Open Air Interface USRP upgrade), since it is not yet officially implemented by OAI developers. To build the code follow:

```
cd /home/mcg/PABLO/OAI2021/openairinterface5g/  
source oaienv  
cd cmake_targets/  
sudo ./build_oai -I -w USRP --eNB --gNB --UE
```

- -I installs all the required software and libraries for OAI.
- -w USRP installs all the radio software requirements
- --eNB and --gNB builds all the libraries for LTE and NR. Executables `lte-softmodem` and `nr-softmodem` will be created.
- --UE builds the `lte-uesoftmodem` executable

If the build has been done correctly, logs will show similar to **Figure 19**:



```
Log file for compilation is being written to: /home/mcg/PABLO/OAI  
2021/openairinterface5g/cmake_targets/log/oai_usrpdevif.Rel15.txt  
oai_usrpdevif compiled  
liboai_device.so is linked to USRP device library  
Compiling rfsimulator  
Log file for compilation is being written to: /home/mcg/PABLO/OAI  
2021/openairinterface5g/cmake_targets/log/rfsimulator.Rel15.txt  
rfsimulator compiled  
Compiling tcp_bridge_oai  
Log file for compilation is being written to: /home/mcg/PABLO/OAI  
2021/openairinterface5g/cmake_targets/log/tcp_bridge_oai.Rel15.tx  
t  
tcp_bridge_oai compiled  
Building transport protocol libraries  
Log file for compilation is being written to: /home/mcg/PABLO/OAI  
2021/openairinterface5g/cmake_targets/log/oai_eth_transpro.Rel15.  
txt  
oai_eth_transpro compiled  
liboai_transpro.so is linked to ETHERNET transport  
10. Bypassing the Tests ...  
BUILD SHOULD BE SUCCESSFUL  
mcg@mcg:~/PABLO/OAI2021/openairinterface5g/cmake_targets$
```

Figure 19: Successful Build Message

OAI 5G is still in a development phase, as a result, SA mode is not available yet, in spite that Open5GCore exists, the roadmap indicates that it will be functional in the end of 2021. NSA mode will be the one tested, it is also in development phase but in a more advanced state.

## 3.2 Configuration Files

In order to make OAI work, configuration files need to be adapted to the EPC used. Most important parameters are explained and shown below for both FOKUS and OAI EPC. There is one file for LTE softmodem and other for NR softmodem.

Firstly, parameters configured are MCC (Mobile Country Code), MNC (Mobile Network Code) and TAC (Tracking Area Code).

Secondly, parameters as band frequencies, number of antennas, gains, resource blocks and transmission mode (TDD or FDD), are configured.

Thirdly, MME directions are included in order to establish connection with the EPC.

Finally, X2 is enabled in order to establish gNB and eNB connection in the NSA mode and the network interfaces used are defined.

Those configuration files are stored in the mcg server, with the path: `/home/mcg/oai2021/config_files`

Next subsections just collect all configuration parameters in order to be able to reproduce the same experiments as performed in this project.

### 3.2.1 OAI EPC LTE

#### 3.2.1.1 Area Parameters

```
// Tracking area code, 0x0000 and 0xfffe are reserved values
tracking_area_code = 1;
plmn_list = (
  { mcc = 208; mnc = 93; mnc_length = 2; }
);

tr_s_preference = "local_mac"
```

#### 3.2.1.2 Physical Parameters

```
component_carriers = (
  {
    node_function           = "eNodeB_3GPP";
    node_timing             = "synch_to_ext_device";
    node_synch_ref         = 0;
    nb_antenna_ports       = 1;
    ue_TransmissionMode    = 1;
    frame_type              = "FDD";
    tdd_config              = 3;
    tdd_config_s            = 0;
    prefix_type             = "NORMAL";
    extra_band              = 7;
    downlink_frequency     = 2680000000L;
    uplink_frequency_offset = -120000000;

    Nid_cell                = 0;
    N_RB_DL                 = 25;
    Nid_cell_mbsfn          = 0;
    nb_antennas_tx          = 1;
    nb_antennas_rx          = 1;
    prach_root              = 0;
    tx_gain                  = 90;
    rx_gain                  = 115;
    pbch_repetition         = "FALSE";
```

### 3.2.1.3 MME parameters

```
////////// MME parameters:
mme_ip_address = ( { ipv4      = "192.168.16.3";
                    ipv6      = "192:168:30::17";
                    port      = 36412 ;
                    active    = "yes";
                    preference = "ipv4";
                    }
);
```

### 3.2.1.4 X2 and network interfaces

```
///  
enable_x2 = "yes";  
t_reloc_prep      = 1000;      /* unit: millisecond */  
tx2_reloc_overall = 2000;      /* unit: millisecond */  
t_dc_prep         = 1000;      /* unit: millisecond */  
t_dc_overall      = 2000;      /* unit: millisecond */  
  
NETWORK_INTERFACES :  
{  
    ENB_INTERFACE_NAME_FOR_S1_MME      = "enp216s0f1";  
    ENB_IPV4_ADDRESS_FOR_S1_MME        = "192.168.10.2";  
  
    ENB_INTERFACE_NAME_FOR_S1U         = "enp216s0f1";  
    ENB_IPV4_ADDRESS_FOR_S1U           = "192.168.10.2";  
    ENB_PORT_FOR_S1U                   = 2152; # Spec 2152  
  
    ENB_IPV4_ADDRESS_FOR_X2C           = "192.168.10.2";  
    ENB_PORT_FOR_X2C                   = 36422; # Spec 36422  
};
```

## 3.2.2 OAI EPC NR

### 3.2.2.1 Area Parameters

```
// Tracking area code, 0x0000 and 0xfffe are reserved values  
tracking_area_code = 1;  
plmn_list = ({mcc = 208; mnc = 93; mnc_length = 2;});  
  
tr_s_preference = "local_mac"
```

### 3.2.2.2 Physical Parameters

```
# downlinkConfigCommon  
#frequencyInfoDL  
# this is 3600 MHz + 84 PRBs@30kHz SCS (same as initial BWP)  
absoluteFrequencySSB = 641272; //641032;  
+ 82 RBs 641032=center of SSB at center of cell  
dl_frequencyBand = 78;  
# this is 3600 MHz  
dl_absoluteFrequencyPointA = 640000;  
  
#uplinkConfigCommon  
#frequencyInfoUL  
ul_frequencyBand = 78;  
#scs-SpecificCarrierList  
ul_offstToCarrier = 0;
```

### 3.2.2.3 MME parameters

```
////////// MME parameters:
mme_ip_address      = ( { ipv4      = "192.168.16.3";
                          ipv6      = "192:168:30::17";
                          port      = 36412 ;
                          active     = "yes";
                          preference = "ipv4";
                        }
);
```

### 3.2.2.4 X2 and network interfaces

```
///X2
enable_x2 = "yes";
t_reloc_prep      = 1000;      /* unit: millisecond */
tx2_reloc_overall = 2000;      /* unit: millisecond */
t_dc_prep         = 1000;      /* unit: millisecond */
t_dc_overall      = 2000;      /* unit: millisecond */
target_enb_x2_ip_address = (
  { ipv4      = "192.168.10.2";
    ipv6      = "192:168:30::17";
    preference = "ipv4";
  }
);

NETWORK_INTERFACES :
{
  GNB_INTERFACE_NAME_FOR_S1_MME      = "enp216s0f0";
  GNB_IPV4_ADDRESS_FOR_S1_MME        = "192.168.20.3";
  GNB_INTERFACE_NAME_FOR_S1U         = "enp216s0f0";
  GNB_IPV4_ADDRESS_FOR_S1U           = "192.168.20.3";
  GNB_PORT_FOR_S1U                   = 2152; # Spec 2152
  GNB_IPV4_ADDRESS_FOR_X2C           = "192.168.20.3";
  GNB_PORT_FOR_X2C                   = 36422; # Spec 36422
};
```

## 3.2.3 FOKUS EPC LTE

### 3.2.3.1 Area Parameters

```
// Tracking area code, 0x0000 and 0xffffe are reserved values
tracking_area_code = 1;
plmn_list = (
  { mcc = 001; mnc = 01; mnc_length = 2; }
);

tr_s_preference      = "local_mac"
```



### 3.2.3.2 Physical Parameters

```

component_carriers = (
{
  node_function           = "eNodeB_3GPP";
  node_timing            = "synch_to_ext_device";
  node_synch_ref         = 0;
  nb_antenna_ports       = 1;
  ue_TransmissionMode    = 1;
  frame_type             = "FDD";
  tdd_config             = 3;
  tdd_config_s           = 0;
  prefix_type            = "NORMAL";
  eutra_band             = 7;
  downlink_frequency     = 2680000000L;
  uplink_frequency_offset = -120000000;

  Nid_cell               = 0;
  N_RB_DL                = 25;
  Nid_cell_mbsfn         = 0;
  nb_antennas_tx         = 1;
  nb_antennas_rx         = 1;
  prach_root             = 0;
  tx_gain                 = 90;
  rx_gain                 = 115;
  pbch_repetition        = "FALSE";
}

```

### 3.2.3.3 MME parameters

```

////////// MME parameters:
mme_ip_address = ( { ipv4      = "192.168.4.80";
                    ipv6      = "192:168:30::17";
                    port      = 36412 ;
                    active    = "yes";
                    preference = "ipv4";
                    }
);

```

### 3.2.3.4 X2 and network interfaces

```

///X2
enable_x2 = "yes";
t_reloc_prep = 1000; /* unit: millisecond */
tx2_reloc_overall = 2000; /* unit: millisecond */
t_dc_prep = 1000; /* unit: millisecond */
t_dc_overall = 2000; /* unit: millisecond */

NETWORK_INTERFACES :
{
  ENB_INTERFACE_NAME_FOR_S1_MME = "enp216s0f1";
  ENB_IPV4_ADDRESS_FOR_S1_MME = "192.168.4.60";

  ENB_INTERFACE_NAME_FOR_S1U = "enp216s0f1:up";
  ENB_IPV4_ADDRESS_FOR_S1U = "192.168.4.100";
  ENB_PORT_FOR_S1U = 2152; # Spec 2152

  ENB_IPV4_ADDRESS_FOR_X2C = "192.168.4.60";
  ENB_PORT_FOR_X2C = 36422; # Spec 36422
};

```





### 3.2.4 FOKUS EPC NR

#### 3.2.4.1 Area Parameters

```
// Tracking area code, 0x0000 and 0xffffe are reserved values
tracking_area_code = 1;
plmn_list = ({mcc = 001; mnc = 01; mnc_length = 2;});

tr_s_preference     = "local_mac"
```

#### 3.2.4.2 Physical Parameters

```
# downlinkConfigCommon
#frequencyInfoDL
# this is 3600 MHz + 84 PRBs@30kHz SCS (same as initial BWP)
absoluteFrequencySSB = 641272; //641032;
+ 82 RBs 641032=center of SSB at center of cell
dl_frequencyBand = 78;
# this is 3600 MHz
dl_absoluteFrequencyPointA = 640000;
#scs-SpecificCarrierList
dl_offstToCarrier = 0;
# subcarrierSpacing
# 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120
dl_subcarrierSpacing = 1;
dl_carrierBandwidth = 106;
#initialDownlinkBWP

#uplinkConfigCommon
#frequencyInfoUL
ul_frequencyBand = 78;
#scs-SpecificCarrierList
ul_offstToCarrier = 0;
# subcarrierSpacing
# 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120
ul_subcarrierSpacing = 1;
ul_carrierBandwidth = 106;
pMax = 20;
```

#### 3.2.4.3 MME parameters

```
////////// MME parameters:
mme_ip_address = ( { ipv4 = "192.168.4.80";
                    ipv6 = "192:168:30::17";
                    port = 36412 ;
                    active = "yes";
                    preference = "ipv4";
                    }
);
```

### 3.2.4.4 X2 and network interfaces

```
///  
enable_x2 = "yes";  
t_reloc_prep      = 1000;      /* unit: millisecond */  
tx2_reloc_overall = 2000;      /* unit: millisecond */  
t_dc_prep         = 1000;      /* unit: millisecond */  
t_dc_overall      = 2000;      /* unit: millisecond */  
target_enb_x2_ip_address = (  
  { ipv4      = "192.168.4.60";  
    ipv6      = "192:168:30::17";  
    preference = "ipv4";  
  }  
);  
  
NETWORK_INTERFACES :  
{  
  // GNB_INTERFACE_NAME_FOR_S1_MME      = "enp216s0f1:n321";  
  // GNB_IPV4_ADDRESS_FOR_S1_MME        = "192.168.4.61/24";  
  // GNB_INTERFACE_NAME_FOR_S1U         = "enp216s0f1:n321";  
  // GNB_IPV4_ADDRESS_FOR_S1U           = "192.168.4.61/24";  
  // GNB_PORT_FOR_S1U                    = 2152; # Spec 2152  
  GNB_IPV4_ADDRESS_FOR_X2C              = "192.168.4.61/24";  
  GNB_PORT_FOR_X2C                      = 36422; # Spec 36422  
};  
};  
);
```

## 3.3 gNB and eNB execution

Next step is to run *lte-softmodem* and *nr-softmodem*, the executables that OAI has to launch eNB and gNB, respectively. Before doing this, EPC needs to be operative. Core was not deployed in this project; however, the basic commands will be indicated in order to be able to follow this guide. OAI EPC and FOKUS Core (Fraunhofer) will be used. The core deployment has been performed in [25].

### 3.3.1 OAI EPC

OAI EPC uses dockers, it is recommended to open four windows on the *cmd*, each will contain one component of the EPC. First step is to start dockers in each window. Cassandra is used in the database so it will be executed in the HSS window, just before executing HSS docker.

```
sudo docker start prod-cassandra  
sudo docker start prod-oai-hss  
sudo docker exec -it prod-oai-hss /bin/bash  
./bin/oai_hss -j ./etc/hss_rel14.json --reloadkey true
```

Next, run the MME:

```
sudo docker start prod-oai-mme  
sudo docker exec -it prod-oai-mme /bin/bash  
./bin/oai_mme -c ./etc/mme.conf
```

Last, SPGW, that is split in control and user:

```
sudo docker start prod-oai-spgwc
sudo docker exec -it prod-oai-spgwc /bin/bash
./bin/oai_spgwc -o -c ./etc/spgw_c.conf
```

```
sudo docker start prod-oai-spgwu-tiny
sudo docker exec -it prod-oai-spgwu-tiny /bin/bash
./bin/oai_spgwu -o -c ./etc/spgw_u.conf
```

This will make EPC operative. If the machine needs to reboot or shut down, it is important that the dockers are all closed, so that they work properly after the reboot.

```
sudo docker stop prod-cassandra
sudo docker stop prod-oai-hss
sudo docker stop prod-oai-mme
sudo docker stop prod-oai-spgwc
sudo docker stop prod-oai-spgwu-tiny
```

### 3.3.2 FOKUS EPC

The execution of the core components is centralized in the script *ph\_init.sh*. This is located in:

```
cd /opt/phoenix/tools/ph_init/
./ph_init.sh
./ph_init.sh
```

It needs to be executed twice, since first execution creates the network space names and second allows entering to that session and interacting with the EPC.

Shortcuts are allowed using Ctrl+B plus:

- N, to go forward and P, to go backward (on each EPC window)
- 0-9 to go to the desired window
- Up or down arrow to activate the top or bottom terminal instance of a window
- D to exit the session, leaving it in the background

To stop the execution of the core, just run the same command followed by down.

```
./ph_init.sh down
```

### 3.3.3 eNB execution

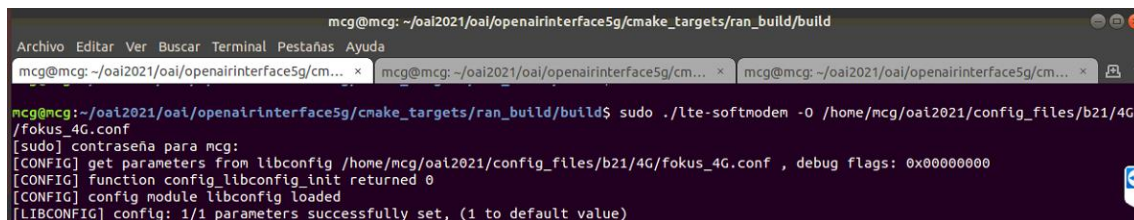
Once the build is completed, the configuration files are ready and the core is deployed, next step is to run OAI eNB. USRP needs to be connected via USB (if B series) or via SFP (if N series). It is important that just one USRP is connected when eNB is executed, since otherwise OAI code will not be able to know where to load eNB. To run OAI eNB go to build folder, then execute the softmodem.

```
cd oai2021/oai/openairinterface5g/cmake_targets/ran_build/build
sudo ./lte-softmodem -O *YOUR_CONFIGURATION_FILE_PATH*
```

In this guide, the configuration files used will be the ones shown in the head 5.2. For example, to run B210 LTE-FOKUS configuration file, the command would be:

```
sudo ./lte-softmodem -O /home/mcg/oai2021/config_files/b21/4G/focus_4G.conf
```

**Figure 20** shows eNB first lines of execution (truncated for readability).



```
mcg@mcg: ~/oai2021/oai/openairinterface5g/cmake_targets/ran_build/build
[sudo] contraseña para mcg:
[CONFIG] get parameters from libconfig /home/mcg/oai2021/config_files/b21/4G/fokus_4G.conf , debug flags: 0x00000000
[CONFIG] function config_libconfig_init returned 0
[CONFIG] config module libconfig loaded
[LIBCONFIG] config: 1/1 parameters successfully set, (1 to default value)
```

**Figure 20: eNB execution example**

To check that is working properly, last lines of execution should be similar to:

```
[0m [0m[PHY] prach_I0 = 0.0 dB
[0m [0m[PHY] max_I0 16 (rb 13), min_I0 11 (rb 0), avg I0 13
```

### 3.3.4 gNB execution

Once the eNB is running, gNB is ready to be executed. USRP needs to be connected via USB (if B series) or via SFP (if N series). It is mandatory to start gNB after eNB is fully loaded. To load OAI gNB, go to the build folder, then execute the softmodem. Note that -E is used only if using USRP B210, since it needs to use  $\frac{3}{4}$  sampling rate (because of hardware limitations).

```
cd oai2021/oai/openairinterface5g/cmake_targets/ran_build/build
sudo ./nr-softmodem -O *YOUR_CONFIGURATION_FILE_PATH* -E
```

Like with eNB, this guide will load B210 NR-FOKUS configuration file, with the following command:

```
sudo ./nr-softmodem -O /home/mcg/oai2021/config_files/b21/5G/focus_5G.conf -E
```

To save a log file of the execution, use the command `|tee *YOUR_LOG_FILE*|` after the configuration file path, substituting your log file desired path for those logs.

**Figure 21** shows gNB first lines of execution (truncated for readability).

```

mcg@mcg: ~/oai2021/oai/openairinterface5g/cmake_targets/ran_build/build
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
mcg@mcg: ~/oai2021/oai/openairinterface5g/cm... x  mcg@mcg: ~/oai2021/oai/openairinterface5g/cm... x  mcg@mcg: ~/oai2021/oai/openairinterface5g/cm... x
mcg@mcg:~/oai2021/oai/openairinterface5g/cmake_targets/ran_build/build$ sudo ./nr-softmodem -O /home/mcg/oai2021/config_files/b21/5G/
okus_5g.conf -E
CONFIG] get parameters from libconfig /home/mcg/oai2021/config_files/b21/5G/fokus_5G.conf , debug flags: 0x00000000
CONFIG] function config_libconfig_init returned 0
CONFIG] config module libconfig loaded
LIBCONFIG] config: 1/1 parameters successfully set, (1 to default value)
  
```

Figure 21: gNB execution example

Next, the procedure of the gNB addition to the LTE network will be explained and a UE attach will be shown from the OAI log file point of view. Once this procedure finishes, the network should be ready for a UE attach.

### 3.3.5 5G Interworking with LTE

OAI uses one of the NR deployment options where LTE works as master and NR becomes a secondary cell. In this configuration, the UE first gets LTE connected to LTE and then redirected to the NR cell via RRC Connection Reconfiguration process. This section will show the procedure checking OAI logs. Schematic is shown in the **Figure 22** below:

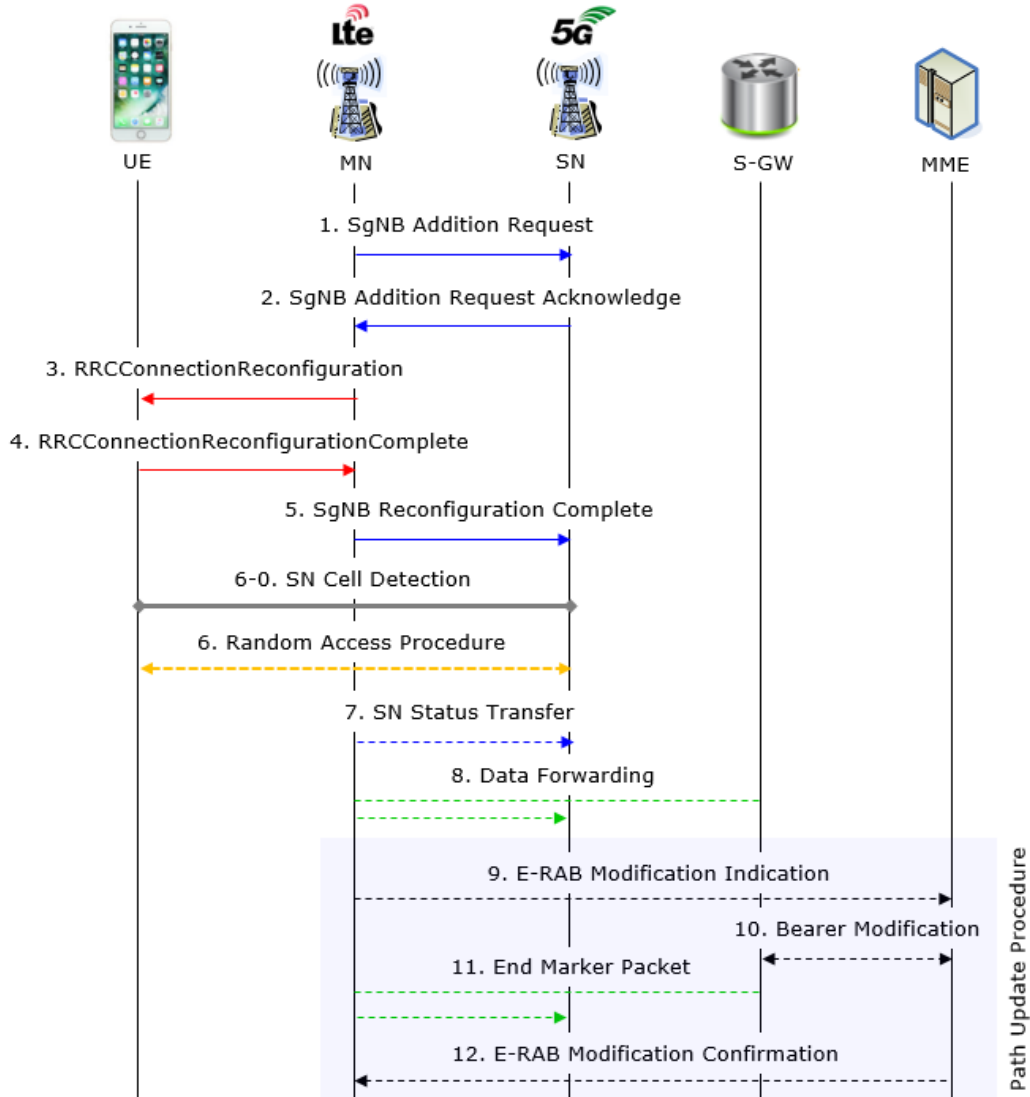


Figure 22: Secondary Node Addition procedure [19]

OAI can monitor several important steps to assess that the test was successful.

Step 1: eNB receives UE capabilities, that include NR capabilities, and triggers SgNB Addition Request.

```
[RRC] [FRAME 00000][eNB][MOD 00][RNTI 7cd3] received ueCapabilityInformation
on UL-DCCH 1 from UE

...

[eNB 0] frame 0 subframe 0: UE rnti 7cd3 switching to NSA mode

...

<X2AP-PDU>
  <initiatingMessage>
    <procedureCode>27</procedureCode>
    <criticality><reject/></criticality>
    <value>
      <SgNBAdditionRequest>
        <protocollEs>
          <SgNBAdditionRequest-IEs>
```

Step 2: gNB receives the *SgNB AdditionRequest* message and triggers ACK, carrying NR RRC Reconfiguration message. The Radio Resource Control (RRC) protocol is used in 5G to share information between UE and Base station on the Air interface. It is a layer 3 protocol specified by 3GPP in 38.331 for 5G New Radio.

```
<X2AP-PDU>
  <successfulOutcome>
    <procedureCode>27</procedureCode>
    <criticality><reject/></criticality>
    <value>
      <SgNBAdditionRequestAcknowledge>
        <protocollEs>
          <SgNBAdditionRequestAcknowledge-IEs>
            <id>111</id>
            <criticality><reject/></criticality>
            <value>
              <UE-X2AP-ID>0</UE-X2AP-ID>
            </value>
          </SgNBAdditionRequestAcknowledge-IEs>
```

Steps 3 and 4: When SgNB Addition Request ACK is received in eNB, it sends a new RRC Connection Reconfiguration with NR Reconfiguration. UE replies with the Reconfiguration Complete message (message may vary with UE used):

```
[RRC] [FRAME 00000][eNB][MOD 00][RNTI 43eb] UE State =  
RRC_RECONFIGURED (default DRB, xid 1)
```

Step 5: eNB sends SgNB Reconfiguration Complete to gNB through X2AP interface:

```
<initiatingMessage>  
  <procedureCode>28</procedureCode>  
  <criticality><ignore/></criticality>  
  <value>  
    <SgNBReconfigurationComplete>  
      <protocolIEs>  
        <SgNBReconfigurationComplete-IEs>
```

Steps 6,7,8: The Random Access procedure of the UE to the gNB occurs, data is forwarded to gNB.

```
[MAC] [gNB 0] Adding UE with rnti 1ad2 (num_UEs 0)  
[MAC] [gNB 0] Add NR UE_id 0 : rnti 1ad2  
[MAC] UCI: CSI_bit len : ssbri 0, rsrp: 7, diff_rsrp: 4  
[MAC] [gNB 0][RAPROC] PUSCH with TC_RNTI 1ad2 received correctly, adding UE  
MAC Context UE_id 0/RNTI 1ad2  
[MAC] reset RA state information for RA-RNTI 1ad2/index 0  
[MAC] handle_nr_uci_pucch_0_1(): unknown RNTI 0000 in PUCCH UCI  
[PHY] LI_thread isn't ready in 68.9, aborting RX processing  
[PHY] could not wakeup gNB rtx process for subframe 9  
[MAC] 69. 6 UL retransmission RNTI 1ad2 sched 69. 8 HARQ PID 0 round 1 NDI 1  
[MAC] 70. 6 UL retransmission RNTI 1ad2 sched 70. 8 HARQ PID 0 round 2 NDI 1  
[MAC] 71. 6 UL retransmission RNTI 1ad2 sched 71. 8 HARQ PID 0 round 3 NDI 1  
[MAC] [gNB 0] Frame 72 : MIB->BCH CC_id 0, Received 3 bytes
```

Step 9: eNB triggers the path switch procedure towards the MME, to route the traffic from the SGW towards the gNB on the S1-U plane.

```
[RRC] sent RRC_DCCH_DATA_REQ to TASK_PDCP_ENB  
[RRC] E-RAB modification indication: nb nb_of_e_rabs 1 status 4  
[RRC] SIAP_E_RAB_MODIFICATION_IND: sending the message:  
nb_of_erabstobemodified 1, total e_rabs 1, index 1
```



Steps 10, 11 and 12 are not visible in OAI logs, but MME should send E-RAB Modification Confirmation in order to reach step 12.

### 3.4 UE attach demonstration

This subchapter's goal is to show from the commercial phone point of view, the attach to the OAI network deployed. Several Commercial Off-The-Shelf (COTS) UE have been tested. However, the best performing one is Samsung A90 5G since OAI code is optimized for Qualcomm Snapdragon 855 Chipset.

In order to show the attach, the following steps will be validated from the COTS UE standpoint.

- 1) Make sure eNB and gNB are running properly, as shown in 5.3.
- 2) Switch the COTS UE to airplane mode (**Figure 23**):



Figure 23: A90 Airplane mode

- 3) Check the phone attach to eNB (**Figure 24**):



Figure 24: A90 4G connection

- 4) Check the handover to gNB cell (**Figure 25**):



Figure 25: A90 5G connection

5) Perform a speed test. The results are shown at **Figure 26**, throughput is low because OAI has just implemented this feature and needs to optimize the network in order to increase it. Still this attach test has shown that the network is fully operative from the signalling point of view and the project objective is reached.

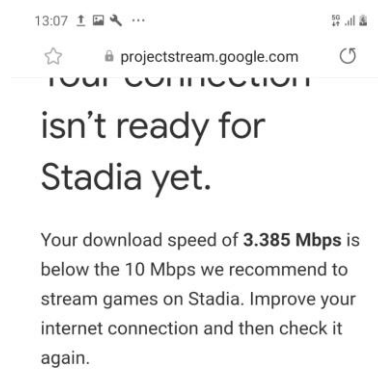


Figure 26: A90 5G speed test

## Chapter 4. Open Air Interface 5G NSA: Performance Testing

This chapter aims to test the deployed network in Chapter 3. The purpose of the network is to give 5G coverage to the lab. In order to measure the coverage and analyse the network quality, ROMES software will be used. ROMES is a powerful tool with features as spectrum analysis, indoor navigation, LTE, NR or IoT scanner, among others.

### 4.1 OAI NR spectrum used

A resource block (RB) is defined as 12 number of consecutive subcarriers in frequency domain. The sub carrier spacing (SCS) is the space between each subcarrier in frequency domain. Maximum bandwidth of USRP B210 is 56MHz. OAI makes use of  $\frac{3}{4}$  of the sample rate to fit 106 RB with 30 kHz of SCS. As a result, the transmission bandwidth is calculated in the **Equation 1**:

$$1 \text{ RB} = 30 \text{ kHz} \cdot 12 = 360 \text{ kHz};$$

$$106 \text{ RB} = 360 \text{ kHz} \cdot 106 = 38,16 \text{ MHz} \quad (1)$$

Each RB has 12 subcarriers with 30 kHz SCS each, as a result it has 360 kHz bandwidth. Since OAI NR uses 106 RB, the total bandwidth used is 38,16 MHz.

The configuration file used assigns the absolute radio-frequency channel number (ARFCN) 641272, which corresponds to a central frequency of 3619.08 MHz, located in 3GPP NR band 78. The spectrum can be visualized in the **Figure 27** below:



Figure 27: OAI NR b78 spectrum

ROMES software also lists some radio parameters as Reference Signal Received Power (RSRP) Reference Signal Received Quality (RSRQ) or Signal to Interference plus Noise Ratio (SINR). This parameters are shown in **Figure 28** for OAI network:

#	PCI	S...	RSSI	SS-RSRP	SS-SI...	SS-RS...	SS-ReP...	NR-ARF...	SS-Ref	GF-l...	GF-l...
1	1	0	-41.35	-68.16	8.47	-10.85	-67.58	641272	3619.08		
1	1	131	0	-80.52	-96.75	24.62	-10.26	-96.73	645020	3675.30	17.4
2	2	131	5	-80.52	-114.16	6.29	-11.21	-113.24	645020	3675.30	
3	3	131	1	-80.52	-115.41	6.47	-11.19	-114.53	645020	3675.30	
1	1	408	1	-62.76	-84.80	18.44	-10.30	-84.74	642048	3630.72	14.3
2	2	408	2	-62.76	-90.26	16.25	-10.34	-90.16	642048	3630.72	
3	3	408	3	-62.76	-91.12	17.58	-10.31	-91.05	642048	3630.72	

Figure 28: ROMES OAI Quality measurements

As the image shows, OAI signal is in ARFCN 641272, and has an RSRQ of -10.85 dB, an RSRP of -68.16 and a SINR of 8.47 dB. Those values have been measured next to the USRP with ROMES scanner.

## 4.2 COTS UE log tracking

Before measuring the coverage offered by OAI network, Samsung logs will be checked in order to match them with the analysed OAI eNB and gNB logs. As the **Figure 29** shows, a few seconds after the gNB is launched, the phone detects the node and starts the attach procedure explained in 3.3.5. Once the reconfiguration is completed, the phone will be attached to both eNB and gNB, with a 5G connection.

```

00:00:14                                     rrcConnectionRequest(Up)
00:00:14    rrcConnectionSetup(Down)
00:00:14                                     rrcConnectionSetupComplete(Up)
00:00:14    dlInformationTransfer(Down)
00:00:14    EmmMsgAuthenticationRequest(Down)
00:00:14                                     EmmMsgAuthenticationResponse(Up)
00:00:14                                     ulInformationTransfer(Up)
00:00:14    dlInformationTransfer(Down)
00:00:14    EmmMsgSecurityModeCommand(Down)
00:00:14                                     EmmMsgSecurityModeComplete(Up)
00:00:14                                     ulInformationTransfer(Up)
00:00:14    securityModeCommand(Down)
00:00:14                                     securityModeComplete(Up)
00:00:14    ueCapabilityEnquiry(Down)
00:00:14                                     ueCapabilityInformation(Up)
00:00:14    rrcConnectionReconfiguration(Down)
    - measConfig
    - dedicatedInfoNASList
    - radioResourceConfigDedicated
    - Dual Connectivity Radio Bearers,RB AddMod...
    - Dual Connectivity Radio Bearers,RB AddMod...
00:00:14                                     rrcConnectionReconfigurationComplete(Up)

```

Figure 29: rrc Connection Reconfiguration

At second 37 of the measurement, UE is detached from the network after some MSG3 (Scheduled PUSCH transmission, with measurement reports) messages failure. As eNB cannot receive a MSG3, a timer expires and eNB sends a SCG (Secondary Cell Group) failure and releases *rrcConnection*. As a result, UE lost 5G connection as shown in **Figure 30**.

```

00:00:37    5G NR RACH: Failure MSG3 (1)(Up)
00:00:37    5G NR RACH: Failure MSG3 (2)(Up)
00:00:37    5G NR RACH: Failure MSG3 (3)(Up)
00:00:37    5G NR RACH: Failure MSG3 (4)(Up)
00:00:37    5G NR RACH: Failure MSG3 (5)(Up)
00:00:37    measurementReport(Up)
00:00:37    5G NR RACH: Failure MSG3 (6)(Up)
00:00:37    5G NR RACH: Failure MSG3 (7)(Up)
00:00:37    measurementReport(Up)
00:00:37    measurementReport(Up)
00:00:37    5G NR RACH: Failure MSG3 (8)(Up)
00:00:37    5G NR RACH: Failure MSG3 (9)(Up)
00:00:37    c2(Up)
    - Dual Connectivity Mobility,SCG Failure,NR SC...
00:00:37    rrcConnectionRelease(Down)
00:00:37    5G NR RACH: Failure MSG3 (10)(Up)
00:00:37    5G NR RACH: Aborted (11)(Up)

```

Figure 30: Dual Connectivity lost

After 3 seconds, at second 40, the phone requests again the *rrcConnection*. After the *rrc Connection Reconfiguration* is completed, the phone has a 5G connection again. This is shown in **Figure 31**:

```

00:00:40      rrcConnectionRequest(Up)
00:00:40      rrcConnectionSetup(Down)
00:00:40      rrcConnectionSetupComplete(Up)
00:00:40      securityModeCommand(Down)
00:00:40      securityModeComplete(Up)
00:00:40      ueCapabilityEnquiry(Down)
00:00:40      ueCapabilityInformation(Up)
00:00:40      rrcConnectionReconfiguration(Down)
- measConfig
- radioResourceConfigDedicated
- Dual Connectivity Radio Bearers, RB AddMod...
- Dual Connectivity Radio Bearers, RB AddMod...
00:00:40      rrcConnectionReconfigurationComplete(Up)
  
```

Figure 31: rrc Connection Reconfiguration after failure

ROMES also summarises the connections available in the COTS UE during the measurements. Since the network deployed is on NSA mode, both LTE and NR are important in order to establish a 5G connection. A summary of the type of coverage obtained along the route through the lab is shown on the serving cell view, in the **Figure 32**:

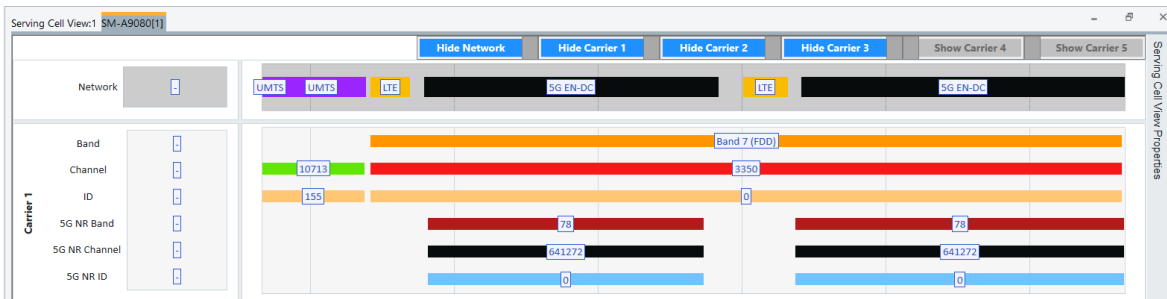


Figure 32: Serving cell view

As it shows, in the beginning the phone is only connected to LTE until gNB is launched and *rrc* is configured. Then it is connected to 5G until coverage is lost, at second 37, as explained before. Seconds after, signal from gNB is recovered and 5G connection is available until the end of the route through the lab. Carrier 1 summary shows LTE band 7 channel 3350 and NR band 78 channel 641272, the parameters assigned from OAI config files.

### 4.3 COTS UE coverage analysis

This subchapter will study the coverage offered by the 5G NSA network deployed with OAI. In order to do it, Samsung A90 5G will be attached to the network and ROMES will track its coverage levels. To determine the quality of the signal the following parameters will be recorded: RSRP (dBm), RSRQ (dB) and SINR (dB).

RSRP (dBm) is the average power of the pilot signals received. In a mobile phone, this is commonly known as the coverage bars that the phone shows. RSRQ (dB) features the quality of the pilot signals received. SINR (dB) is the relation between the signal and the interference plus noise, the bigger this value is, the better the quality is.

A map of the iTEAM lab, shown in **Figure 33**, will be designed and loaded to ROMES software. Some waypoints will be established in order to make the measurements.

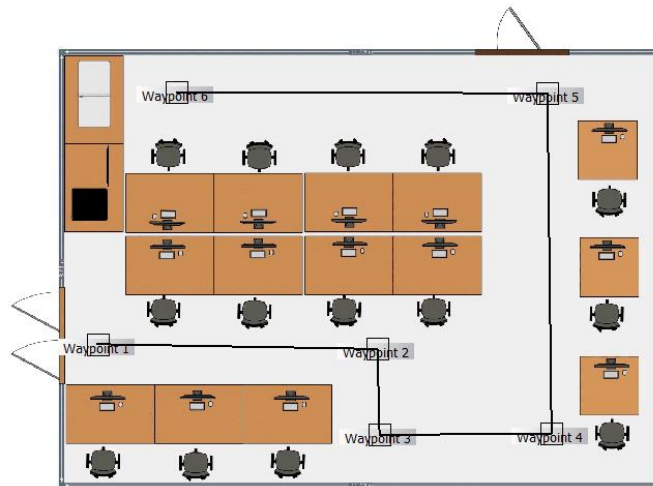


Figure 33: iTEAM lab map

A90 needs to be connected by USB 3.0 to the computer where ROMES will run. To check that all the permissions are given by the smartphone use the code `*#0808#` in the phone app (works for Samsung). When it is connected, airplane mode is switched on and OAI eNB+gNB are launched. Once OAI is running, airplane mode is switched off. Then, the attach procedure will begin, and the coverage analysis will start.

OAI antennas are located next to the waypoint 1, hanging from the door as **Figure 34** shows. OAI eNB is launched at second 6 of the measurement record and gNB at second 9. The measuring procedure will consist of moving the phone around the lab following the route designed. Once a waypoint is reached, waypoint arrival key will be pressed and ROMES will draw in the map the parameter desired, following a colour scale previously defined.



Figure 34: Waypoint 1, antennas location

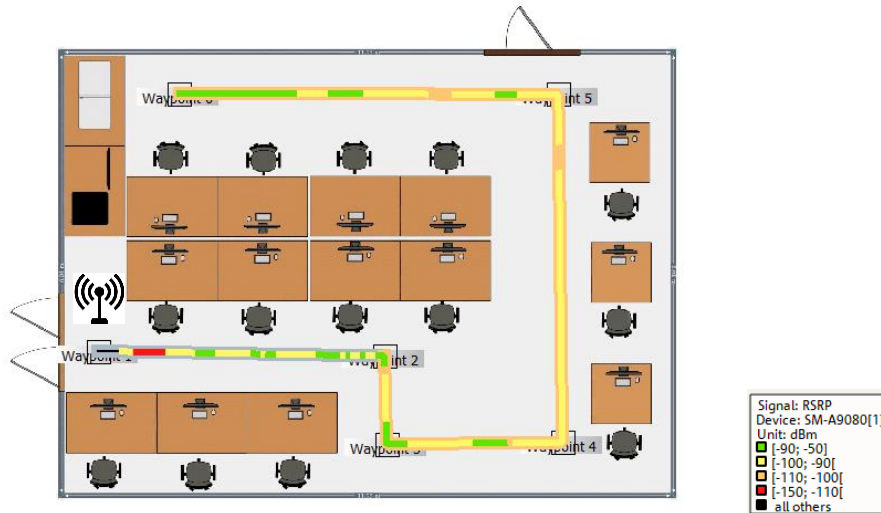
During the path, Samsung A90 will send values of the parameters of the signal received. As it was explained before, the interesting ones are RSRP (dBm), RSRQ (dB) and SINR (dB). The scale used to determine the quality of the parameters is adapted for the setup as **Figure 34** shows:

	RSRP (dBm)	RSRQ (dB)	SINR (dB)
<b>Excellent</b>	$RSRP > -90$	$RSRQ > -5$	$SINR > 15$
<b>Good</b>	$-90 > RSRP > -100$	$-5 > RSRQ > -10$	$15 > SINR > 10$
<b>Regular</b>	$-100 > RSRP > -110$	$-10 > RSRQ > -15$	$10 > SINR > 5$
<b>Bad</b>	$RSRP < -110$	$RSRQ < -15$	$SINR < 5$

Figure 35: Quality threshold defined

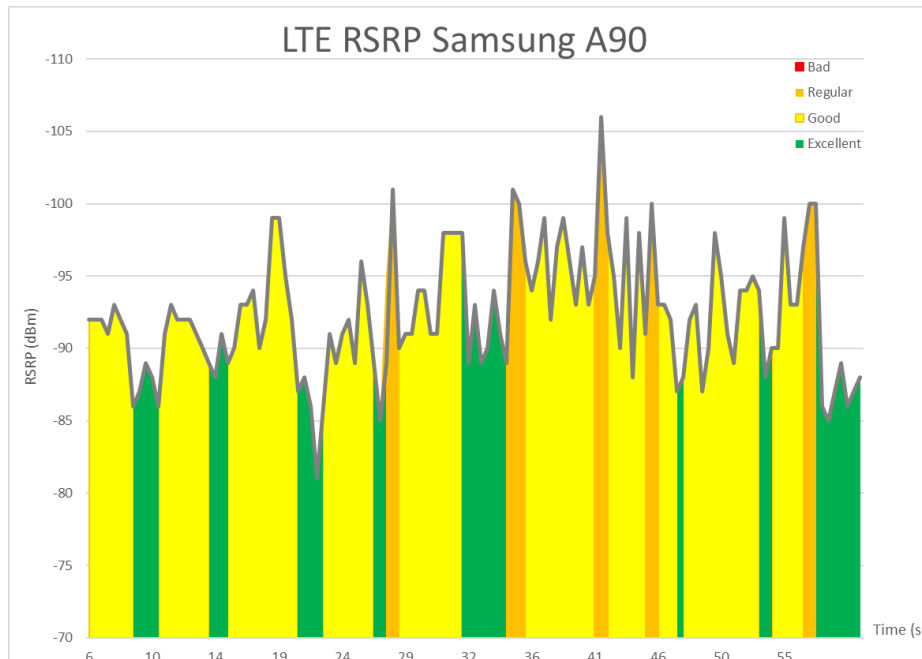
### 4.3.1 LTE RSRP

**Figure 36** shows LTE RSRP measurement. In the beginning of the route, the phone attaches to the network and has average levels of coverage until waypoint 4. From there, it begins to lose power until waypoint 5, after this one, the levels are worse since there are more obstacles and distance from the antenna to the phone. Since the antennas used are for testing purposes and not for commercial use, once the COTS UE is far from the OAI nodes, its coverage is affected. However, all the RSRP levels obtained could handle data traffic without problems.



**Figure 36: LTE RSRP (lab map)**

**Figure 37** is a graph that contains the RSRP levels from the LTE network over time. Waypoint 2 is reached around second 23, and the RSRP level until there is good to excellent. From there on, level is good but turns regular in some far spots, around waypoint 5 (second 41). Once waypoint 5 is passed, RSRP levels improve until the end.

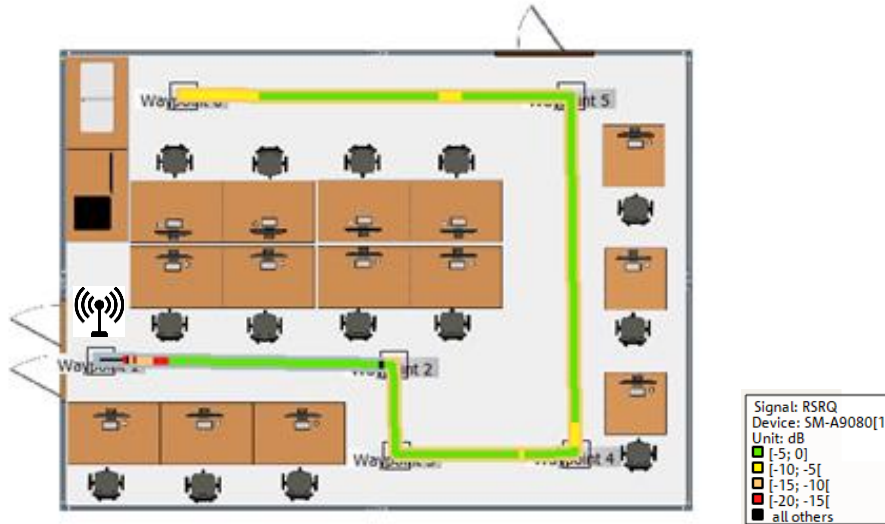


**Figure 37: LTE RSRP (graph)**



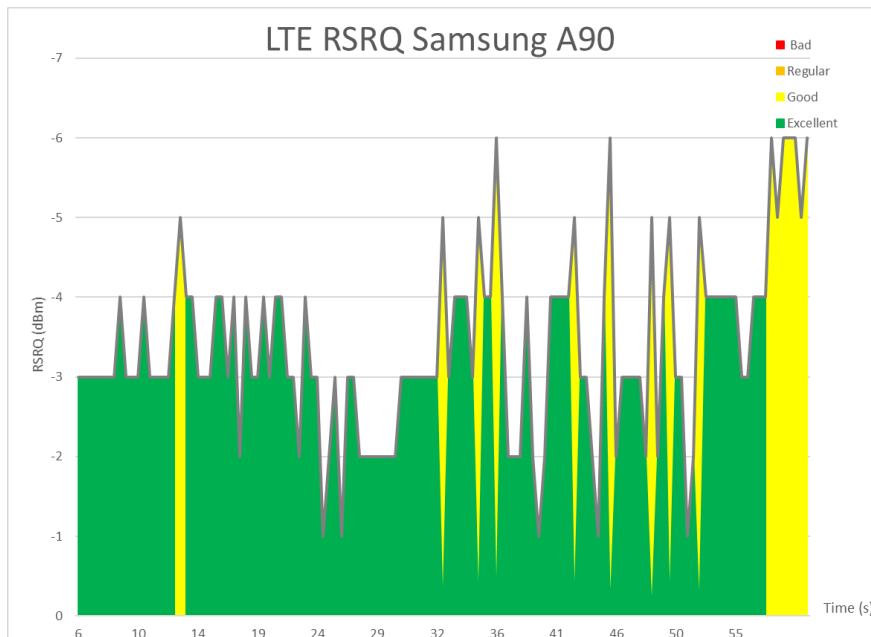
### 4.3.2 LTE RSRQ

**Figure 38** shows LTE RSRQ measures. Once the connection is established, RSRQ has an excellent level the whole walkthrough with some exception spots where the RSRQ level is only good.



**Figure 38: LTE RSRQ (lab map)**

**Figure 39** features its corresponding graph, showing that the RSRQ level is excellent with some small exceptions, where it is just good.

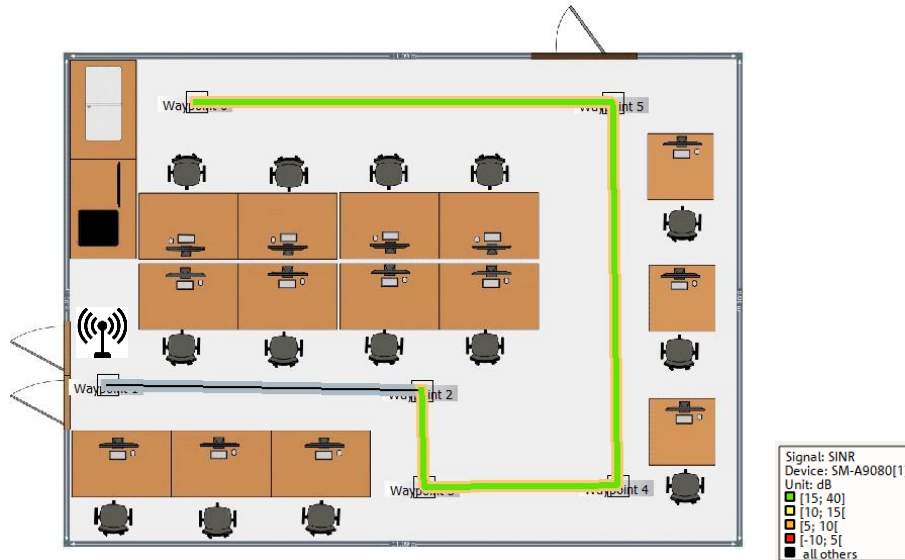


**Figure 39: LTE RSRQ (graph)**



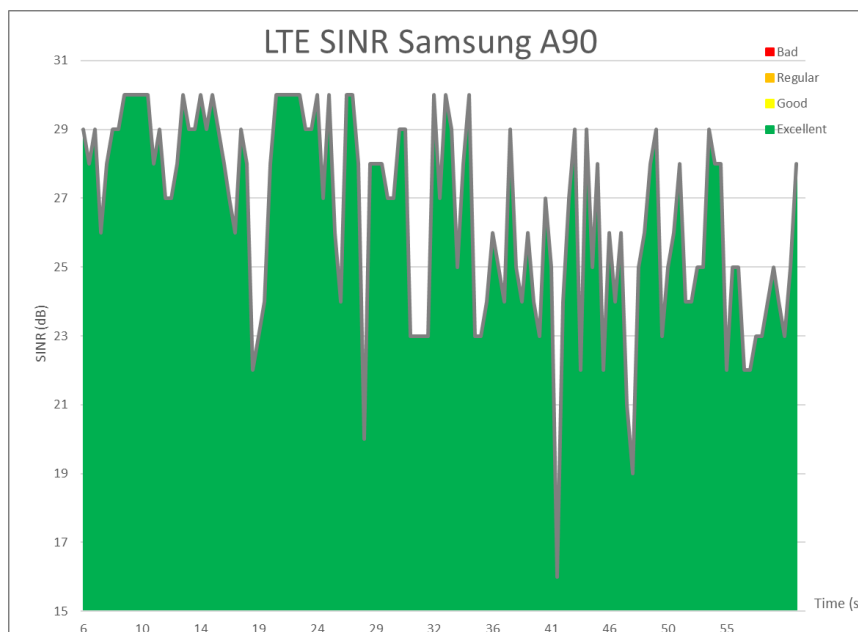
### 4.3.3 LTE SINR

**Figure 40** shows LTE SINR measures. The phone starts sending those measurements two seconds after the connection is established. The levels obtained when direct vision to the antenna is available are good. However, once the signal has to pass through some lab equipment and people (from the waypoint 4 on), SINR decreases. Nonetheless, the levels are excellent in all the lab.



**Figure 40: LTE SINR (lab map)**

**Figure 41** is the graph from the map above.



**Figure 41: LTE SINR (graph)**

In conclusion, after measuring all these parameters, the eNB has good coverage and quality levels. As control plane goes through eNB, gNB needs good parameters from eNB in order to work properly in this space.

#### 4.3.4 NR RSRP

In this section, the 5G coverage levels will be analysed through RSRP parameter. The measurements start when the *rrcConnectionReconfigurationComplete* message is sent from the phone to the eNB. The average RSRP level is -105.6 dBm. In general terms, UE reports better RSRP levels when the antenna has close frontal vision. On other hand, the coverage is degraded in the outer points of the route, due to the pointing of the antenna. The equipment used as gNB is not designed to be used for commercial uses and big distances, which is why when distances higher than 15 m are reached, connection is not very stable. **Figure 42** shows NR RSRP measurements.

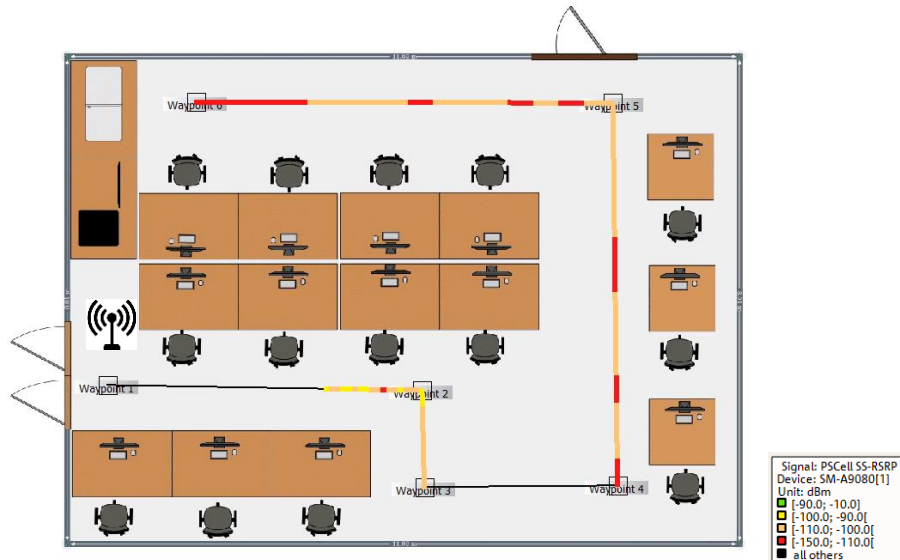


Figure 42: NR RSRP (lab map)

**Figure 43** shows the corresponding graph, which gives a more accurate view.

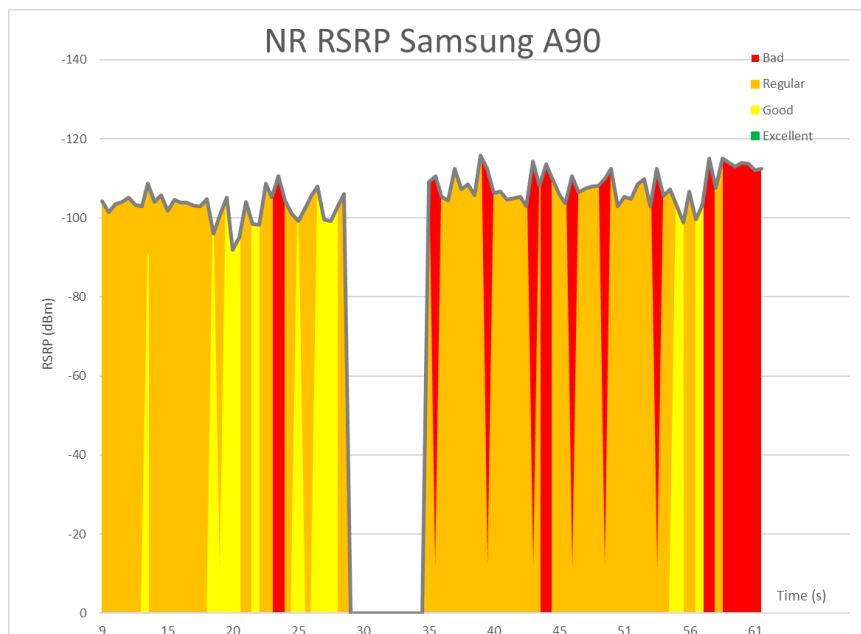
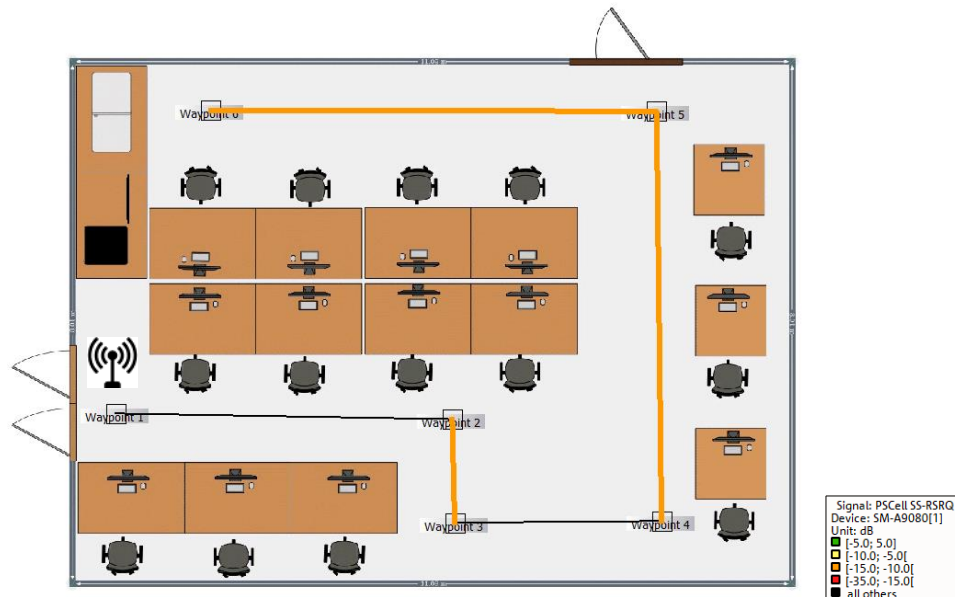


Figure 43: NR RSRP (graph)

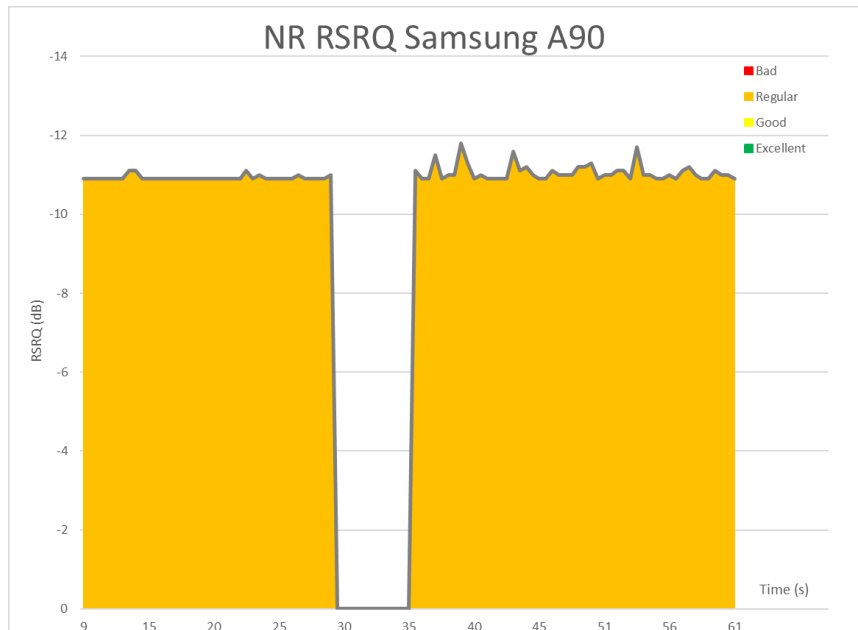
### 4.3.5 NR RSRQ

**Figure 44** shows NR RSRQ levels. Those are acceptable whenever the 5G connection is available.



**Figure 44: NR RSRQ (lab map)**

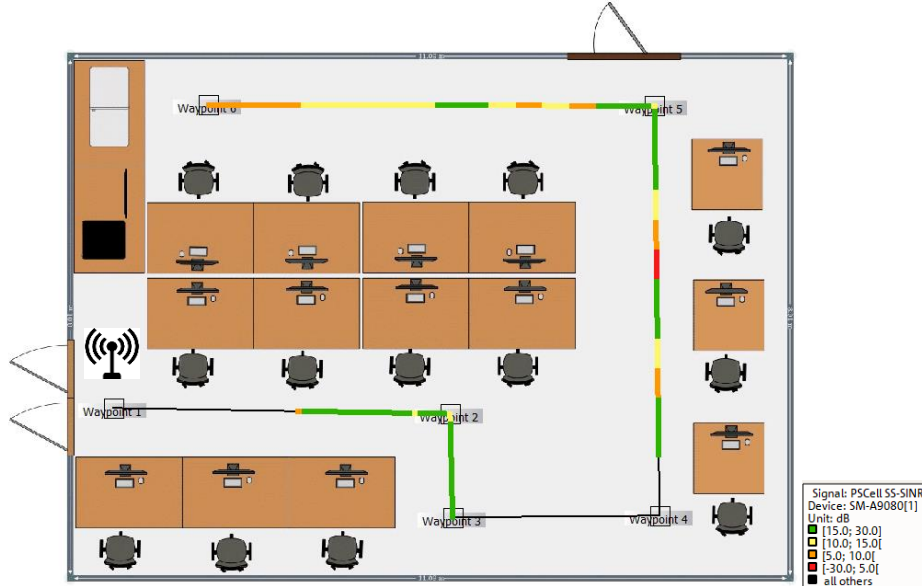
As **Figure 45** displays, those levels are very stable, and are close to reach the good threshold.



**Figure 45: NR RSRQ (graph)**

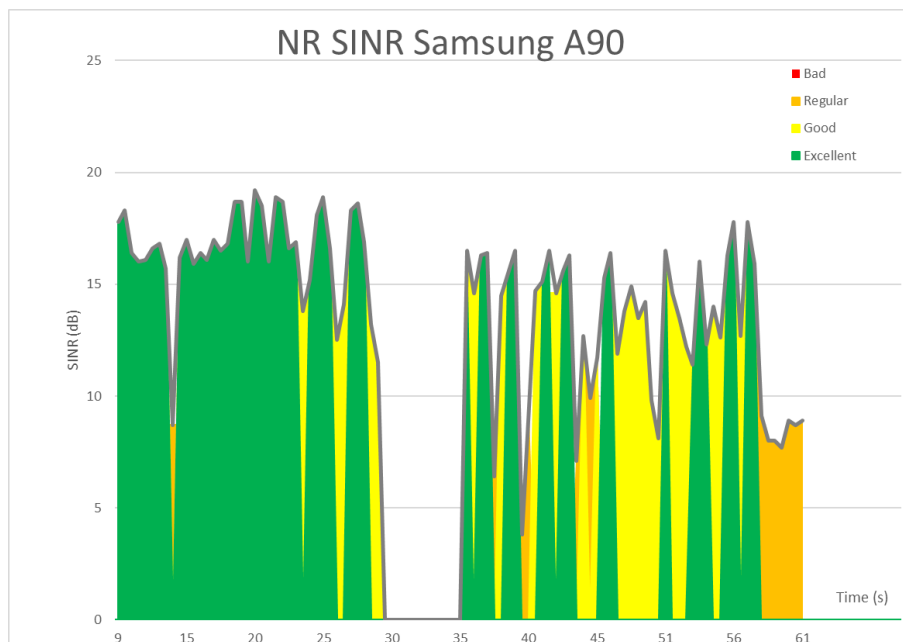
### 4.3.6 NR SINR

**Figure 46** shows SINR levels of 5G are good nearly all the route. When 5G connection is lost it takes a little to the phone to start sending again those levels. Some spots between waypoint 4 and 5 have worse SINR, probably due to a punctual noise or interference.



**Figure 46: NR SINR (lab map)**

The corresponding graph is represented by **Figure 47**, offering a more accurate view of NR SINR levels.



**Figure 47: NR SINR (graph)**

In conclusion, the levels obtained are enough to handle data traffic but far spots may experience lower throughput because of the distance to the gNB. However, as explained before, this equipment is not designed to offer coverage, but for research purposes.

## Chapter 5. Open Air Interface USRP Upgrade

OAI currently makes use of USRP B210 or N310 in order to deploy the gNB. The highest bandwidth reached with N310 is 100 MHz. iTEAM has available the next generation of USRP, the N321. This USRP samples at higher rates, allowing up to 200 MHz of instantaneous bandwidth per channel, sampling at 245.76 MHz. This would increase the throughput because more resource blocks will fit in the spectrum and higher values of SCS are allowed. As a result, following 3GPP equations, up to 273 RB with 30kHz of SCS could be handled by OAI.

### 5.1 Theoretical throughput

In 5G, data speed is calculated regarding 3GPP technical specification 38.306, following the Equation 2.

$$\text{Throughput (Mbps)} = 10^{-6} \cdot \sum_{j=1}^J \left( v_{Layers}^{(j)} \cdot Q_m^{(j)} \cdot f^{(j)} \cdot R_{max} \cdot \frac{N_{PRB}^{BW(j) \cdot \mu} \cdot 12}{T_s^\mu} (1 - OH^{(j)}) \right) \quad (2)$$

Where  $j$  is the number of aggregated carriers in a band and  $R_{max}$  is a constant with the value of 948/1024. For each carrier:

- $v^{(j)}$  is the maximum number of layers admitted
- $Q^{(j)}$  is the modulation order, can be 2 for QPSK, 4 for 16QAM, 6 for 64QAM or 8 for 256QAM
- $f^{(j)}$  is the scaling factor and can take the values of 1, 0.8, 0.75 y 0.4
- $\mu$  is the numerology, that indicates the subcarrier spacing. Can take the value of:
  - 0 for FR1 with 15 kHz
  - 1 for FR1 with 30 kHz
  - 2 for FR1 with 60 kHz
  - 3 for FR2 with 60 kHz
  - 4 for FR2 with 120 kHz
  - \*(FR1 goes from 4.1 GHz to 7.125 GHz and FR2 24.25 GHz to 52.6 GHz)
- $T_s^\mu$  is the average OFDM symbol duration for each  $\mu$ . Use to take the value of  $\frac{10^{-3}}{14 \cdot 2^\mu}$
- $N_{PRB}^{BW(j) \cdot \mu}$  is the max. number of resource blocks per bandwidth part:
  - 270 for FR1 with 15 kHz
  - 273 for FR1 with 30 kHz
  - 135 for FR1 with 60 kHz
  - 264 for FR2 with 60 kHz
  - 264 for FR2 with 120 kHz
- $OH^{(j)}$  is the overhead and takes the value of:
  - 0.14, for FR1 in DL
  - 0.18, for FR2 in DL
  - 0.08, for FR1 in UL
  - 0.10, for FR2 in UL

The throughput using N321 for a single layer using 16QAM, in DL and 30 kHz SCS should be:

$$\text{Throughput (Mbps)} = 10^{-6} \cdot 1 \cdot 1 \cdot 4 \cdot 1 \cdot \left(\frac{948}{1024}\right) \cdot 273 \cdot 12 \cdot \frac{(14 \cdot 2^0)}{10^{-3}} \cdot (1 - 0.14) = 146.06 \text{ Mbps} \quad (2)$$

In order to use USRP N321, adaptations to the UHD and to OAI code need to be done. First, this guide will explain the adaptations to the UHD of the USRP (this is equivalent to a firmware). Then, changes to OAI code will be shown, and finally, a benchmark of the communication between Linux machine and USRP will be performed.

## 5.2 UHD-FPGA of Ettus USRPs

The USRP, as every electronic device, like a graphics card or an ethernet board, needs hardware drivers in order to work properly. Those drivers are called UHD, and its function is to control the USRP FPGA motherboards and daughterboards. It communicates with and controls all the USRP device family. In order to communicate with the Linux machine both USRP and Linux machine need to have the same UHD version. **Figure 48** shows the USRP motherboards and FPGAs:

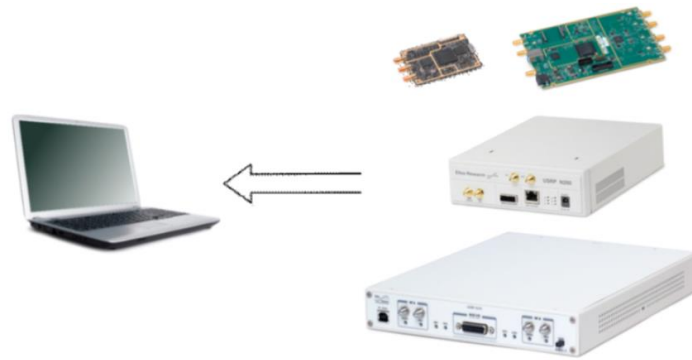


Figure 48: UHD Function [26]

USRP can transmit and receive RF signals simultaneously (full duplex mode). UHD provides the control used to transport user waveform samples to and from the USRP as well as control parameters like sampling rate, centre frequency or gains of the radio peripheral. Host computer driver and firmware is written mainly in C/C++ and the FPGA code is written in Verilog. **Figure 49** shows the UHD components of both USRP and host computer [26].

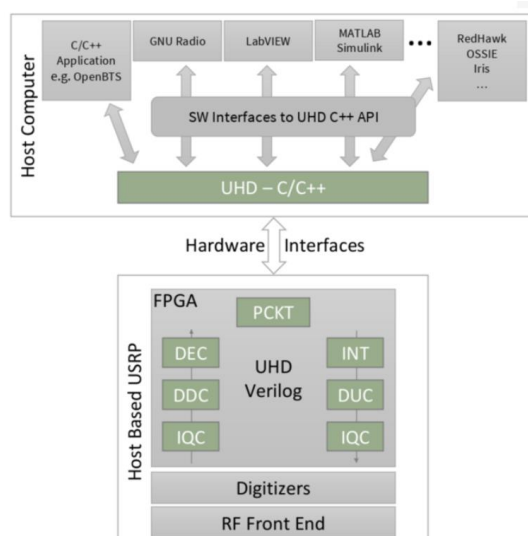


Figure 49: UHD Components [26]

UHD code is open source, available under the open-source GNU Public License version 3. Ettus Research also offer a less-restrictive license.

Different SDR configurations, waveforms and applications require different sampling rates. Both host PC and USRP need to be capable of analysing the applied bandwidth. If either PC or USRP are not able to handle the set sampling rate, the system loses samples.

General Purpose Input Output pins from the FPGA can be controlled through UHD to trigger when events as TX or RX occur. Open Air Interface makes use of it in order to communicate the host computer with the USRP. OAI code makes sure during the build procedure that the UHD in the Linux machine is compatible with the USRP tested. Currently, since they are working with B200/B210 for LTE or X310/N310 for NR, they have configured and checked only those USRPs in their code. As explained before, the bandwidth of those USRP is smaller than the N321. In order to boost the throughput USRP N321 could use 200 MHz bandwidth channel, increasing the performance of OAI. **Figure 50** shows N32X FPGA (it is the same for N320/N321, the only difference between those USRPs is that N321 is designed to perform massive MIMO).

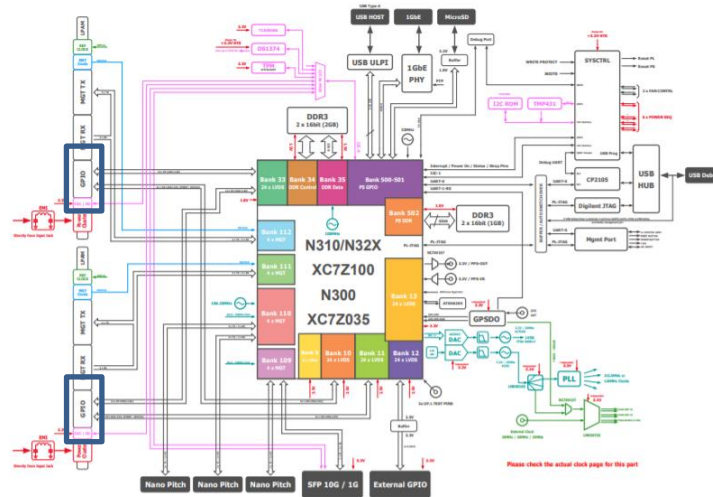


Figure 50: N32X FPGA [27]

As shown in **Figure 50**, there are many banks with lots of pins each, which can be used for multiple functions. General Purpose Input/Output (GPIO) pins are the ones used by the UHD to establish communication with the Linux host machine.

Eurecomm have not tested this USRP yet since it is a new model. However, iTEAM MCG has this USRP and from now on, adaptations to both USRP and OAI code will be explained. The goal is to make the USRP N321 run OAI code.

### 5.3 UHD upgrade

OAI installs UHD 3.14 with its code during the build procedure. This UHD version is not controlling USRP N321 properly since it does not provide access to the FPGA GPIO pins of the USRP N321. As a result, while trying to run OAI code the following error pops out, shown in **Figure 51**:

```

TYPE <CTRL-C> TO TERMINATE
Entering ITTI signals handler
terminate called after throwing an instance of 'uhd::runtime_error'
what(): RuntimeError: The hardware has no GPIO bank 'FP0'
Linux signal Aborted...
/home/mcg/5goai/openairinterface5g-develop/executables/softnoden-common.c:187 signal_handler() Exiting OAI softnoden: softnoden starting exit procedure

mcg@mcg:~/5goai/openairinterface5g-develop/cmake_targets/ran_build/build$ sudo nando /home/mcg/5goai/openairinterface5g-develop/executables/nr-softnoden
sudo: nando: orden no encontrada
mcg@mcg:~/5goai/openairinterface5g-develop/cmake_targets/ran_build/build$

```

Figure 51: OAI Runtime Error UHD 3.14



As the image shows, OAI log shows that the hardware has no GPIO bank 'FP0'. But if the motherboard of the FPGA is checked, that GPIO bank does exist. The problem is that the UHD 3.14 does not provide the correct access to that pin, then the FPGA is not able to find it and use it to start full duplex communication with the Linux machine.

An upgrade needs to be done in order to make UHD recognise GPIO of the N321 motherboard. UHD 3.15 does provide access to those pins at N321, and will be installed in both the USRP and the Linux machine in order to solve that error.

### 5.3.1 USRP UHD 3.15

Before doing anything with OAI code, UHD 3.15 needs to be installed in the USRP N321. Two options are available to flash to USRP the new UHD version. The first one requires physical access to the USRP with an SD card. The second one does not require physical access, using the tools provided in Chapter 2 we can connect to the USRP. The tool used to update the USRP UHD is Mender. Next steps are followed to update the USRP UHD [28]:

The first step consist of the installation of Mender, reboot of the system and mender commit in order the boot loader boot into the new Mender partition.

```
$ mender install /path/to/latest.mender  
$ reboot  
$ mender commit
```

To obtain the file system Mender image (files with a *.mender*) run the following command of Linux:

```
$ sudo uhd_images_downloader -t mender -t n3xx --yes
```

This should automatically download in */usr/local/share/uhd/images* UHDv3.15 or higher. Next step is to copy this file in the USRP, this can be done by using scp utility:

```
$ scp /usr/local/share/uhd/images/usrp_n3xx_fs.mender  
root@<USRP_IP>:~/.
```

Note that the path should be where the image is, */usr/local* is the default one and substitute *<USRP\_IP>* with the IP of your USRP. After the file is copied use SSH to connect to the USRP to update the system UDH:

```
$ ssh root@<USRP_IP>  
root@ni-n3xx-serial:~# mender -rootfs  
/home/root/usrp_n3xx_fs.mender
```

Output should be similar to the next one if the installation has performed properly. This procedure may take a few time:

```
root@ni-n3xx-serial:~# mender -rootfs /home/root/usrp_n3xx_fs.mender
INFO[0000] Configuration file does not exist: /var/lib/mender/mender.conf module=config
INFO[0000] Loaded configuration file: /etc/mender/mender.conf module=config
INFO[0000] Mender running on partition: /dev/mmcbk0p2 module=main
INFO[0000] Start updating from local image file: [/home/root/usrp_n3xx_fs.mender]
module=rootfs

Installing update from the artifact of size 460989952
INFO[0000] no public key was provided for authenticating the artifact module=installer
INFO[0000] opening device /dev/mmcbk0p3 for writing module=block_device
INFO[0000] partition /dev/mmcbk0p3 size: 7851737088 module=block_device
..... 0% 1024 KiB
..... 0% 2048 KiB
..... 0% 3072 KiB
[truncated for readability]
..... 99% 448512 KiB
..... 99% 449536 KiB
..... 100% 450185 KiB

INFO[3004] wrote 7851737088/7851737088 bytes of update to device /dev/mmcbk0p3
module=device

INFO[3009] Enabling partition with new image installed to be a boot candidate: 3
module=device
```

After updating, reboot the system and commit the changes again, then perform an UHD probe to check that the installation is working properly and UHD has access to all FPGA features.

```
$ reboot
$ mender -commit
$ uhd_usrp_probe
```

Output from the `uhd_usrp_probe` should be similar to the following image, check if all clocks initialize and lock properly, this means UHD is working perfectly in the N320/N321 USRP and all functionalities are correct as **Figure 52** shows:

```
root@ni-n3xx-31BF638:~# uhd_usrp_probe
[INFO] [UHD] linux; GNU C++ version 8.2.0; Boost_106800; UHD_3.15.0.0-0-gaea0e2de
[INFO] [MPMD] Initializing 1 device(s) in parallel with args: mgmt_addr=127.0.0.1,type=n3xx,product=n320,serial=31BF638,claimed=False
[INFO] [MPM.PeriphManager] init() called with device args `mgmt_addr=127.0.0.1,time_source=internal,product=n320,clock_source=internal'.
[INFO] [MPM.Rhodium-0] init() called with args `mgmt_addr=127.0.0.1,time_source=internal,product=n320,clock_source=internal'.
[INFO] [MPM.Rhodium-1] init() called with args `mgmt_addr=127.0.0.1,time_source=internal,product=n320,clock_source=internal'.
[INFO] [0/Replay_0] Initializing block control (NOC ID: 0x4E91A00000000004)
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000000320)
[INFO] [0/Radio_1] Initializing block control (NOC ID: 0x12AD100000000320)
[INFO] [0/DDC_0] Initializing block control (NOC ID: 0xDDC000000000001)
[INFO] [0/DDC_1] Initializing block control (NOC ID: 0xDDC000000000001)
[INFO] [0/DUC_0] Initializing block control (NOC ID: 0xD0C000000000000)
[INFO] [0/DUC_1] Initializing block control (NOC ID: 0xD0C000000000000)
[INFO] [0/FIFO_0] Initializing block control (NOC ID: 0xF1F000000000000)
[INFO] [0/FIFO_1] Initializing block control (NOC ID: 0xF1F000000000000)
```

Figure 52: UHD 3.15 USRP Probe

As the image shows, all is initialized properly and UHD 3.15.0 is installed on our USRP device. Also, check that the motherboard is detected properly and UHD 3.15 is functional, shown in **Figure 53**:

```
Device: N300-Series Device
-----
Mboard: ni-n3xx-31BF638
eeprom_version: 3
mpm_version: 3.15.0.0-gaea0e2de
pid: 16962
product: n320
rev: 7
rpc_connection: local
serial: 31BF638
type: n3xx
MPM Version: 1.2
FPGA Version: 5.3
FPGA git hash: fde2a94.clean
RFNoC capable: Yes

Time sources: internal, external, gpsdo, sfp0
Clock sources: external, internal, gpsdo
Sensors: gps_locked, fan, ref_locked, gps_ppgga, gps_sky, gps_tpv, gps_time, temp
```

Figure 53: USRP Probe information

### 5.3.2 Linux Host UHD 3.15

If Linux machine has not the same UHD version as the USRP that is communicating with, an error similar to this will pop out during OAI execution:

```
[INFO] [0/DmaFIFO_0] Initializing block control (NOC ID: 0xF1F0D00000000000)
[ERROR] [0/DmaFIFO_0] Major compat number mismatch for noc_shell: Expecting 4, got 5.

Error: RuntimeError: FPGA component `noc_shell' is revision 5 and UHD supports revision 4. Please either upgrade UHD (recommended) or downgrade the FPGA image
```

The main problem found here is that OAI installs automatically UHD 3.14 in the Linux machine when building USRP libraries with the command:

```
./build_oai -I -w USRP --eNB --gNB
```

The solution to this issue is to change OAI code in order to install UHD 3.15 instead. The function that install UHD 3.15 is located in:

```
cd ./openairinterface5g/cmake_targets/build_oai
```

Depending on the branch of the git, there will be a function in the `build_oai` script that installs UHD. In the develop branch, used for NR, this function is `install_usrp_uhd_driver` and needs to be replaced by `install_usrp_uhd_driver_from_source`. **Figure 54** shows the exact line to be changed:

```
if [ "$INSTALL_EXTERNAL" = "1" ]; then
  echo_info "Installing packages"
  check_install_oai_software
  if [ "$HW" == "OAI_USRP" ]; then
    echo_info "installing packages for USRP support"
    #check_install_usrp_uhd_driver
    if [ ! "$DISABLE_HARDWARE_DEPENDENCY" == "True" ]; then
      install_usrp_uhd_driver_from_source $UHD_IMAGES_DIR_
    fi
  fi
```

Figure 54: Modifications to build\_oai

This function is called from `build_helper`, which is located in the path:

```
cmake_targets/tools/build_helper
```

OAI allows you to install a UHD driver from source using a function described in *build\_helper*. The goal now is to make this function install manually UHD 3.15. Depending on the git version used it can have UHD 3.12 or 3.13 in the function. To make it install 3.15 replace the version for 3.15.0.0, as **Figure 55** shows, which will be compatible with USRP N321. This function is around line 450.

```
install_usrp_uhd_driver_from_source(){
    uhd_install_log=${OPENAIR_DIR}/cmake_targets/log/uhd_install_log.txt
    echo_info "\nInstalling UHD driver from sources. The log file for UHD driver installation is here: $uhd_install_log "
    (
        cd /tmp
        echo "Downloading UHD driver"
        rm -rf /tmp/uhd
        git clone https://github.com/EttusResearch/uhd.git
        cd uhd
        git checkout tags/v3.15.0.0
        mkdir -p host/build
        cd host/build
        $CMAKE ../
        echo "Compiling UHD"
        make -j `nproc`
        make test
        $SUDO make install
        $SUDO ldconfig
    ) >& $uhd_install_log
}
```

**Figure 55: UHD 3.15 installation from source**

This function now will install UHD 3.15 from the Ettus Research git when *build\_oai* is performed. To check if it is successfully installed run the build command at *cmake\_targets* path:

```
./build_oai -I -w USRP --eNB --gNB
```

When doing the build procedure, it will install uhd by source instead, this will take longer than a usual build but will install UHD 3.15 from Ettus Research git. Once the build is completed, check the UHD version using the same procedure as done at USRP N321.

```
$ uhd_usrp_probe
```

Output should probe that UHD 3.15 is installed properly in the Linux machine as shown in **Figure 56**:

```
root@openocp:~# uhd_usrp_probe
[INFO] [UHD] Linux; GNU C++ version 7.5.0; Boost 106581; UHD 3.15.0-HEAD-0-gaea0e2de
[INFO] [RPD] Initializing 1 device(s) in parallel with args: mgmt_addr=192.168.20.2,type=n3xx,product=n320,serial=31BF638,claimed=False,addr=192.168.20.2
[WARNING] [MPP_RPCServer] A timeout event occurred!
[INFO] [MPP_Parameters] init() called with device args 'mgmt_addr=192.168.20.2,time_source=internal,product=n320,clock_source=internal'
[INFO] [MPP_Rhodium_0] init() called with args 'mgmt_addr=192.168.20.2,time_source=internal,product=n320,clock_source=internal'
[INFO] [MPP_Rhodium_1] init() called with args 'mgmt_addr=192.168.20.2,time_source=internal,product=n320,clock_source=internal'
[INFO] [0/Replay_0] Initializing block control (NOC ID: 0x4E91A00000000004)
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000000320)
[INFO] [0/Radio_1] Initializing block control (NOC ID: 0x12AD100000000320)
[INFO] [0/DUC_0] Initializing block control (NOC ID: 0xD0C0000000000001)
[INFO] [0/DUC_1] Initializing block control (NOC ID: 0xD0C0000000000001)
[INFO] [0/DUC_2] Initializing block control (NOC ID: 0xD0C0000000000001)
[INFO] [0/DUC_3] Initializing block control (NOC ID: 0xD0C0000000000000)
[INFO] [0/DUC_4] Initializing block control (NOC ID: 0xD0C0000000000000)
[INFO] [0/FIFO_0] Initializing block control (NOC ID: 0xF1F0000000000000)
[INFO] [0/FIFO_1] Initializing block control (NOC ID: 0xF1F0000000000000)

-----
Device: N300-Series Device
-----
Mboard: ni-n3xx-31BF638
eprom_version: 3
mgmt_version: 3.15.0.0-gaea0e2de
pid: 16992
product: n320
rev: 7
rpc_connection: remote
serial: 31BF638
type: n3xx
MPP Version: 1.2
FPGA Version: 5.3
FPGA git hash: fde2a94.clean
RFNOC capable: Yes

Time sources: internal, external, gpsdo, sfp0
Clock sources: external, internal, gpsdo
Sensors: gps_locked, fan, ref_locked, gps_ppgga, gps_sky, gps_tpv, gps_time, temp

-----
RX Dboard: A
ID: Unknown (0x0152)
Serial: 3191E7A
```

**Figure 56:UHD 3.15 in Linux Machine**

Once this step is reached, the N321 and the Linux machine have the same UHD, and it is compatible with the new USRP, since it gives access to the GPIO pins used by OAI.

#### 5.4 Sample rate and master clock changes

Other field that needs to be updated in OAI code is the sample rate and the master clock, since it is only defined for B series and N310. As the USRP used will be N321, OAI code does not recognise its sample rates, which are summarized in the table below in **Figure 57**:

Master Clock Rate	Decimation / Interpolation Rate Host Sample Rate [Msps]									
	1	2	4	6	8	10	12	14	16	18
200e6	100e6	50e6	33.33e6	25e6	20e6	16.66e6	14.2857e6	12.5e6	11.11e6	
245.76e6	122.88e6	61.44e6	30.72e6	20.48e6	15.36e6	12.288e6	10.24e6	8.7771e6	7.68e6	
250e6	125e6	62.5e6	31.25e6	20.833e6	15.625e6	12.5e6	10.417e6	8.9286e6	7.8125e6	

Master Clock Rate	Decimation / Interpolation Rate Host Sample Rate [Msps]									
	20	30	32	64	100	128	200	256	512	1024
10e6	6.667e6	6.25e6	3.125e6	2e6	1.5625e6	1e6	781.25e3	390.625e3	195.3125e3	
6.8267e6	6.144e6	4.096e6	3.84e6	1.92e6	1.2288e6	960e3	614.4e3	480e3	240e3	
6.9444e6	6.25e6	4.1667e6	3.90625e6	1.953125e6	1.25e6	976.5625e3	625e3	488.28125e3	244.14e3	

**Figure 57: N321 Sample rate for each master clock**

As the **Figure 57** shows, there are three master clocks recognised by USRP N312, 200 MHz, 245.76 MHz and 250 MHz. For each, sample rate can be chosen by the user, as it will be shown below, OAI code chooses it to depend on some parameters. Sampling rate can be changed in the *script nr-ru.c*, that is located in the path:

```
openairinterface5g/executables/nr-ru.c
```

```

if (mu == NR_MU_0) { //or if LTE
    if(N_RB == 80) {
        cfg->sample_rate=1e6;
        cfg->samples_per_frame = 10000;
        cfg->tx_bw = 1e5;
        cfg->rx_bw = 1e5;
    }
    if(N_RB == 100) {
        if (fp->threequarter_fs) {
            cfg->sample_rate=23.04e6;
            cfg->samples_per_frame = 230400;
            cfg->tx_bw = 10e6;
            cfg->rx_bw = 10e6;
        } else {
            cfg->sample_rate=30.72e6;
            cfg->samples_per_frame = 307200;
            cfg->tx_bw = 10e6;
            cfg->rx_bw = 10e6;
        }
    }
    else if(N_RB == 50) {
        cfg->sample_rate=15.36e6;
        cfg->samples_per_frame = 153600;
        cfg->tx_bw = 5e6;
        cfg->rx_bw = 5e6;
    }
    else if (N_RB == 25) {
        cfg->sample_rate=7.68e6;
        cfg->samples_per_frame = 76800;
        cfg->tx_bw = 2.5e6;
        cfg->rx_bw = 2.5e6;
    }
    else if (N_RB == 6) {
        cfg->sample_rate=1.92e6;
        cfg->samples_per_frame = 19200;
        cfg->tx_bw = 1.5e6;
    }
} else if (mu == NR_MU_1) {
    if(N_RB == 273) {
        if (fp->threequarter_fs) {
            AssertFatal(0 == 1, "three quarter sampling not supported for N_RB 273\n");
        } else {
            cfg->sample_rate=122.88e6;
            cfg->samples_per_frame = 1228800;
            cfg->tx_bw = 100e6;
            cfg->rx_bw = 100e6;
        }
    }
    else if(N_RB == 217) {
        if (fp->threequarter_fs) {
            cfg->sample_rate=92.16e6;
            cfg->samples_per_frame = 921600;
            cfg->tx_bw = 80e6;
            cfg->rx_bw = 80e6;
        } else {
            cfg->sample_rate=122.88e6;
            cfg->samples_per_frame = 1228800;
            cfg->tx_bw = 80e6;
            cfg->rx_bw = 80e6;
        }
    }
    else if(N_RB == 106) {
        if (fp->threequarter_fs) {
            cfg->sample_rate= 200.00e6;
            cfg->samples_per_frame = 2000000;
            cfg->tx_bw = 120e6;
            cfg->rx_bw = 120e6;
        }
        else {
            cfg->sample_rate= 122.88e6;
            cfg->samples_per_frame = 1228800;
            cfg->tx_bw = 40e6;
            cfg->rx_bw = 40e6;
        }
    }
    else {
        AssertFatal(0==1, "N_RB %d not yet supported for numerology %d\n",N_RB,mu);
    }
}

```

Figure 58: OAI Sample rate definition

This code defines the sample rate depending on the technology used (LTE or NR) and the resource blocks assigned in the configuration file. Before changing sampling rate, make sure that your setup handles that rate. In order to probe that, UHD has a tool to test the communication between the UHD and the Linux machine. The script needs to be executed in the UHD examples folder, usually located in:

```

/usr/local/lib/uhd/examples

```

The command to perform a sample transmission to test a certain sample rate is:

```

./benchmark_rate --rx_rate <SR>--tx_rate <SR>

```

Where *SR* is the desired sample rate. Since the USRP that this project is optimizing is N321, its sample rate will be tested. Master clock defined from 3GPP and the one that will be used with OAI code is 245.76 MHz, therefore it could allow a bandwidth of 200 MHz (N310 max bandwidth was 100 MHz). Channel bandwidth and sampling rates are shown in **Figure 59**:

Channel Bandwidth (MHz)	Nominal FFT size	Sampling Rate (MHz)
50	1024	61.44
100	2048	122.88
200	4096	245.76

Figure 59: Channel Bandwidth and sapling rate

Before testing it, master clock for N series needs also to be changed in OAI code, since it is defined for N310, which had 122.88 MHz, 125.0 MHz or 153.6 MHz as available master clocks. OAI defined 122.88 MHz in its code, that value should be substituted by 245.76 MHz. It is located in the script *usrp\_libb.c*, which is always located in the path:



```
openairinterface5g/targets/ARCH/USRP/USERSPACE/LIB/usrp_libb.c
```

Around line 1000, depending on OAI branch and commit, the master clock definition for each device type (B, N or X series) is set. The one that needs to be modified, since N321 belongs to N series is “n3xx” type. The image below shows the exact line that needs to be changed. Set the value 247.76 MHz since it is the one established as explained before as shown in **Figure 60**.

```
if (device_adds[0].get("type") == "b200") {
    device->type = USRP_B200_DEV;
    usrp_master_clock = 30.72e6;
    args += boost::str(boost::format(",master_clock_rate=%f") % usrp_master_clock);
    args += ",num_send_frames=256,num_rcv_frames=256, send_frame_size=7680, rcv_frame_size=7680" ;
}

if (device_adds[0].get("type") == "n3xx") {
    printf("Found USRP n300\n");
    device->type=USRP_N300_DEV;
    usrp_master_clock = 245.76e6;
    args += boost::str(boost::format(",master_clock_rate=%f") % usrp_master_clock);
    //args += ", send_buff_size=33554432";
}

if (device_adds[0].get("type") == "x300") {
    printf("Found USRP x300\n");
    device->type=USRP_X300_DEV;
    usrp_master_clock = 184.32e6;
    args += boost::str(boost::format(",master_clock_rate=%f") % usrp_master_clock);

    // USRP recommended: https://files.ettus.com/manual/page_usrp_x3x0_config.html
    if ( 0 != system("sysctl -w net.core.rmem_max=33554432 net.core.wmem_max=33554432") )
        LOG_W(HW,"Can't set kernel parameters for X3xx\n");
}
}
```

Figure 60: N321 Master clock modified

Once this is changed, USRP code needs to be compiled again. To accelerate the process, instead of compiling the whole OAI code or USRP files using the build command in *cmake\_targets*, run this code to just compile the USRP device:

```
$ sudo make oai_usrpdevif
```

If the build is performed properly, the output should be like **Figure 61**:

```
mcg@mcg:~/oai/4G/openairinterface5g-develop/cmake_targets/ran_build/build$ sudo make oai_usrpdevif
[ 0%] Built target generate_T
[100%] Built target oai_usrpdevif
mcg@mcg:~/oai/4G/openairinterface5g-develop/cmake_targets/ran_build/build$ _
```

Figure 61: USRP build successful

Everything now is ready to test the new configuration set for USRP N321. 128.88 MHz will be value tested, since 245.76 MHz had some data loss (probably due to the physical setup). However, note that this may not work in every Linux machine since it depends on the setup. Having a higher frequency CPU would help to perform 245.76 MHz sampling rates without data loss. To test the rate, go to the folder *uhd/examples* and run:

```
./benchmark_rate --rx_rate 122.88e6 tx_rate 122.88e6
```

The UHD will start sending and receiving and the output should be something similar to (**Figure 62**):







## Chapter 6. Conclusions, Limitations and Future Work

This final degree project has deployed, tested, and improved a fully operation open-source 5G network in a non-stand-alone mode. This has been in the framework of Valencia Campus 5G project, led by the iTEAM research institute of the Universitat Politècnica de València.

This study has consisted, firstly, in a thorough review of the technical foundations of 4G and 5G open source networks, in addition to the main entities surrounding it.

The second step has been to decide the equipment that is going to be used during the project and to learn how it works. This means to get acquainted with the working environment, the available tools and the specific devices used to deploy the network.

Once all the equipment was ready, the deployment of the network had a lot of troubleshooting work, since it was deployed while OAI was developing it, and some modifications were done in order to make the open source code work. This includes logs depuration, minor changes in OAI scripts, changes in the phone configuration and choosing the best terminal. Six COTS UE were tested, among which is Samsung A90 5G, the best performing one in the analysis. This device was added to the OAI supported devices in their website after this study.

Open Air Interface radiance element, the USRP, was also improved in this final degree project. To make the new generation USRP N321 work with OAI code was a real challenge, because the iTEAM was the first one to make it work, and it has been possible thanks to contributions with other OAI users. Indeed, when iTEAM acquired the USRP, the firmware released was not able to control yet all the motherboard pins that OAI used. N321 usage will make OAI code support higher bandwidths, which would allow higher throughput in upcoming tests. Some guides regarding all those aspects have been shared with OAI community during the realization of the project. Moreover, some support regarding this has been offered to other research groups, so they could test the changes made in this final degree project.

After solving all the found errors while executing the code, the network deployed was tested. In the beginning, it was not very stable as the phone detached from the network after some seconds. An optimization work was done improving the network stability. The best scenario was using different servers to run eNB and gNB, since the CPU usage of each process was high and sometimes made the node crash. The higher the CPU frequency was, the easier it was to handle the node. USRP N321 stability is lower than B210 because the information sent by the USRP N321 to the host computer was bigger and, as a result, harder to manage by the Linux machine. Some EPC were also tested and the best performing one was FOKUS EPC.

When the network was stable enough to test, the use case was performed. A coverage study in the lab proved that the actual state of the project is enough to give a 5G connection to the Mobile Communications Group lab.

The project had to be adapted to all the limitations experienced. It started in March 2020, at UPV. As it is known, in the middle of March, the pandemic stopped the activity at university. As a result, the project had to be developed at home, but not all the equipment available could be moved and the setup was adapted the first months.

The network was not functional for Eurecomm until the end of 2020, and during that process, the research consisted of helping OAI community with troubleshooting and improves to the code. This was possible thanks to OAI mailing list, sharing information with the rest of the researchers. Once the network was functional, the limitations experienced were related to the hardware used, since the CPU frequency was not higher enough to handle all the setup, and nodes crashed after some time. The solution given was to split in two Linux machines the setup. In addition, not every COTS UE was functional with OAI because each phone processor sends messages in a particular way, and some of them were not recognised by OAI code. Samsung A90 was the best performing phone in the test, but in the beginning it was not recognised as compatible with OAI, and this



project made OAI realise that it also works. The last limitation experienced was the lack of power of USRP to give coverage during the test. As the device is not designed for commercial uses, the QoS was not very good during the tests.

In spite of the limitations, thanks to all the community, OAI state is improving day by day, and goals as having a standalone 5G stable network are not that far. During the development of the project, more research groups, institutes, and private entities joined part in the OAI community. This is a very good new, since it has been proved that open source 5G solutions are functional and this will offer operators a new tool, transparent and adaptable to each situation. Industry will also benefit from this improvements since low size networks could be deployable for private uses, making use of unused spectrum.

Regarding ORAN, it has more support, investors, and resources each day. Every day this organization has more support in order to improve OpenRAN architecture. An integration ORAN-OAI project started at Eurecomm on Summer 2020 and it is still in an early phase. ORAN fronthaul library can be integrated in the OAI gNB in order to boost it with ORAN characteristics and it is supposed to be testable at the end of 2021, according to OAI roadmap. The branch that is developing this process is called *oai-oran-integration* and it is available in the OAI git repository. However, the code is still in stage of development and all the values are hard-coded for now. It is possible only to read the hard coded parameters and to read IQ samples from OAI. The integration of ORAN with OAI will be a very important step as it will build a new concept of RAN, open, smart and malleable.

As future lines of work, the next points are considered:

- To design more use cases of the network deployed, as controlling self-driving vehicles or machines.
- To improve the stability of the network and the throughput by using USRP N321. This could be possible using better CPUs and manipulating OAI code.
- To test the integration of OAI with ORAN, in order to open a new investigation line in a technology that is going to be very important in the future of mobile networks.
- To test OAI SA mode and optimize that network, testing the latency, throughput and coverage offered with each USRP.

## References

- [1] D. Mejía, G. B. Naranjo, D. Paúl, C. Buenaño and Iván Tenecora, Evolución de la tecnología móvil. Camino a 5G” Revista Contribuciones a las Ciencias Sociales, 2016.
- [2] R. Griffins, “The news unlimited: 5G is an Evolution of 4G Mobile Radio,” [Online]. Available: <https://www.thenewsunlimited.com/5g-is-an-evolution-of-4g-mobile-radio-technologies/>. [Accessed 3 12 2020].
- [3] J. Kempf and P. Yegani, “OpenRAN: a new architecture for mobile wireless Internet radio access networks,” *IEEE Communications Magazine*, vol. 40, no. 5, pp. 118-123, 2002.
- [4] J. Kempf, “OpenRAN Architecture in 3rd Generation Mobile Systems,” MWIF WG4, 2001.
- [5] Huawei, “Wireless private network, test of potential.,” 2020. [Online]. Available: <https://e.huawei.com/es/solutions/business-needs/wireless-private-network/broadband-access>. [Accessed 31 12 2020].
- [6] D. Hetzer, M. Muehleisen, A. Kousaridas and J. Alonso-Zarate, “5G Connected and Automated Driving: Use Cases and Technologies in Cross-border Environments,” *European Conference on Networks and Communications (EuCNC)*, pp. 78-82, 2019.
- [7] ORAN, “O-RAN Virtual Exhibition,” 2020. [Online]. Available: <https://www.virtualexhibition.o-ran.org/classic/generation/2020>. [Accessed 3 1 2021].
- [8] OAI, “OAI Wiki,” 2020. [Online]. Available: <https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/home>. [Accessed 3 1 2021].
- [9] E.Westerberg, “4G/5G RAN architecture: how a split can make the difference,” *Ericsson Magazine*, 2016.
- [10] ORAN, “O-RAN: Towards an Open and Smart RAN,” *O.-R Alliance*, 2018.
- [11] D. Yadav, “Network Slicing Architecture in ORAN,” July 2020. [Online]. Available: <https://www.linkedin.com/pulse/network-slicing-architecture-oran-dheeraj-kumar/>. [Accessed 2021 1 13].
- [12] “Attocore,” 2020. [Online]. Available: <https://www.attocore.com/products/attoepc/>. [Accessed 2020 12 5].
- [13] “LTE Encyclopedia,” LTE Network Infrastructure and Elements, [Online]. Available: <https://sites.google.com/site/lteencyclopedia/>. [Accessed 12 1 2021].
- [14] S. Rommer, *5G Core Networks: Powering Digitalization.*, Academic Press ISBN: 9780081030097., 2020.
- [15] T.Kassis, “NSA vs SA Configurations,” [Online]. Available: <https://es.gearbest.com/blog/how-to/nsa-or-sa-who-is-the-real-5g-network-mode-for-smartphones-and-whats-the-difference-8585>. [Accessed 2020 12 4].
- [16] OAI, “OAI NSA and SA architecture,” [Online]. Available: <https://openairinterface.org/5g-modem/> Visited 18/01/2021. [Accessed 18 1 2021].



- [17] Alepo Company, “5G SA vs 5G NSA What are the differences.,” 2020. [Online]. Available: <https://www.alepo.com/5g-sa-vs-5g-nsa-what-are-the-differences/>. [Accessed 5 12 2020].
- [18] Software Radio Systems, “SRS,” [Online]. Available: <https://www.softwareradiosystems.com/>. [Accessed 10 1 2021].
- [19] Open Air Interface, “OAI,” 2020. [Online]. Available: <https://www.openairinterface.org/>. [Accessed 3 1 2021].
- [20] R. Knopp and F. Kaltenberger, “OpenAirInterface 5G: Overview, Installation, Usage,” p. 75, 2019.
- [21] Ettus Research, “USRP B200/B210,” 2020. [Online]. Available: <https://kb.ettus.com/B200/B210/B200mini/B205mini>. [Accessed 13 1 2021].
- [22] Ettus Research, “USRP X310/X310,” 2020. [Online]. Available: <https://kb.ettus.com/X300/X310>. [Accessed 13 1 2021].
- [23] Ettus Research, “USRP N300/N310/N320/N321,” 2020. [Online]. Available: [https://kb.ettus.com/USRP\\_N300/N310/N320/N321\\_Getting\\_Started\\_Guide](https://kb.ettus.com/USRP_N300/N310/N320/N321_Getting_Started_Guide). [Accessed 13 1 2021].
- [24] Amarisoft, “Amarisoft About Us,” [Online]. Available: <https://www.amarisoft.com/about-us/>. [Accessed 14 1 2021].
- [25] A. M. Handzel., Implementación del Núcleo de Red LTE/5G Virtualizado, Valencia: TFG UPV, 2020.
- [26] Ettus Research, “Ettus UHD,” [Online]. Available: <https://kb.ettus.com/UHD>. [Accessed 17 1 2021].
- [27] E. Research, “Ettus N32X FPGA,” [Online]. Available: [https://kb.ettus.com/images/f/f4/USRP\\_N3XX\\_MB\\_Schematic.pdf](https://kb.ettus.com/images/f/f4/USRP_N3XX_MB_Schematic.pdf). [Accessed 17 1 2021].
- [28] Ettus Research, “Mender UHD Update Guide,” [Online]. Available: [https://kb.ettus.com/USRP\\_N300/N310/N320/N321\\_Getting\\_Started\\_Guide#Mender\\_Update\\_Process](https://kb.ettus.com/USRP_N300/N310/N320/N321_Getting_Started_Guide#Mender_Update_Process). [Accessed 20 1 2021].