Building a Cellular Network using OpenBTS

Project Students

February 2025

BUILDING A CELLULAR NETWORK USING OpenBTS

BY

Izu Ogaga Jeremiah	**ENG1905340**
Opeyemi Babafemi	**ENG1905373**
Umukoro Gregory Jeffery	**ENG1905401**
Udosen Emmanuel	**ENG1905396**

A PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRICAL/ELECTRONIC ENGINEERING. FACULTY OF ENGINEERING UNIVERSITY OF BENIN, BENIN CITY

February 2025

CERTIFICATION

This is to accurately certify that this project work was carried out by **Izu Ogaga Jeremiah**, **Opeyemi Babafemi**, **Umukoro Gregory Jeffery**, and **Udosen Emmanuel**, in partial fulfillment of the requirement for the award of Bachelor of Electrical/Electronic Engineering, Department of Electrical/Electronic Engineering, Faculty of Engineering, University of Benin, Benin City, Edo State.

Engr. Dr. N. Bello **(Supervisor)**	**Date**
Engr. Prof. K.O. Ogbeide **(HoD)**	**Date**

CERTIFICATION OF THESIS ON PLAGIARISM

We, the undersigned, attest and declare that this project report of **Izu Ogaga Jeremiah**, **Opeyemi Babafemi**, **Umukoro Gregory Jeffery**, and **Udosen Emmanuel** on **Building a Cellular Network using OpenBTS**, has successfully passed the anti-plagiarism test and does not violate any copyright regulations.

Engr. Dr. N. Bello (Supervisor)	**Date**
Engr. Prof. K.O. Ogbeide (HoD)	**Date**

DEDICATION

This work is unreservedly dedicated to **almighty God**; the **Izu's family**; the **Opeyemi's family**; the **Umukoro's family**; the **Udosen's family**, for their unending support and love exhausted during our study at the University of Benin.

ACKNOWLEDGMENT

We unreservedly, without measure, channel thanks to **almighty God** for His love, grace, mercies, and protection upon us in the period of our studying at the University of Benin. We also thank the families of **Izu**, **Opeyemi**, **Gregory**, **Udosen**, and our friends, who supported us financially, morally, and spiritually, throughout our journey in the University of Benin. Lastly, we stret to thank **Engr. Dr. N. Bello**, our project supervisor, for his immense sacrifice, time and effort during (and in reviewing) this project - instilling in us the discipline to always aim for the best. Forgetting not our esteemed, **Engr. Prof. K.O. Ogbeide**, HoD for his support directed towards the entire Department of Electrical and Electronic Engineering, **Uniben**, we thank him, too.

ABSTRACT

The relevance of communication in our world today cannot be overemphasized. This project is thus aimed at designing and implementing a GSM Network using OpenBTS software paired with an SDR (Software-defined Radio), imitating the GSM Network as we know it. This setup can be used in small-scale operations and can solve the major issue of connectivity in rural areas such as villages and small towns.

This arrangement involved a **USRP B210** and OpenBTS software running on the **Ubuntu 24.04** the latest version of the operating system - as of the time of writing.

At the end, the network was launched, subscribers were registered on the test network, and they were able to send messages, which is a supported feature on the GSM Network.

Contents

Certification ii

Certification of Thesis on Plagiarism iii

Dedication iv

Acknowledgments v

Abstract vi

List of Figures xii

List of Tables xiii

Abbreviation xiv

1 INTRODUCTION 1
- 1.1 Background of Study . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Aim . . . 3
- 1.4 Objectives . . . 3
- 1.5 Methodology . . . 3
- 1.6 Outline . . . 4

2 LITERATURE REVIEW 5
- 2.1 Introduction . . . 5
- 2.2 Cellular Networks . . . 5
- 2.3 History of Cellular Networks . . . 6
  - 2.3.1 Early Years . . . 6

2.3.2 First Generation Networks (1G) . . . 7
2.3.3 Second Generation Networks (2G) . . . 7
2.3.4 Third Generation Networks (3G) . . . 7
2.3.5 Fourth Generation Networks (4G) . . . 7
2.3.6 Fifth Generation Networks (5G) . . . 7
2.4 Why GSM . . . 8
2.5 The GSM Network . . . 8
2.6 The GSM Architecture . . . 9
2.7 THE MOBILE STATION (MS) . . . 10
2.7.1 Mobile Equipment (ME) . . . 11
2.7.2 Subscriber Identity Module (SIM) . . . 11
2.8 THE BASE STATION SUBSYSTEM (BSS) . . . 12
2.8.1 Base Transceiver Station (BTS) . . . 12
2.8.2 Base Station Controller (BSC) . . . 13
2.9 NETWORK AND SWITCHING SUBSYSTEM (NSS) . . . 13
2.9.1 Mobile Switching Center (MSC) . . . 14
2.9.2 Gateway Mobile Switching Center (GMSC) . . . 14
2.9.3 Home Location Register (HLR) . . . 15
2.9.4 Visitor Location Register (VLR) . . . 15
2.9.5 Equipment Identity Register (EIR) . . . 15
2.9.6 Authentication Center (AuC) . . . 15
2.10 OPERATIONS AND SUPPORT SUBSYSTEMS (OSS) . . . 16
2.11 SOFTWARE DEFINED RADIO . . . 16
2.12 BRIEF HISTORY OF SDR . . . 17
2.12.1 SDR Operation . . . 18
2.13 RF Hardware . . . 19
2.14 ADC/DAC . . . 19
2.15 Tiers of Software Defined Radio . . . 20
2.15.1 Hardware-based Radios (Tier 0) . . . 20
2.15.2 Software Controlled Radios (Tier 1) . . . 20
2.15.3 Reconfigurable Software Defined Radio (Tier 2) . . . 21
2.15.4 Ideal Software Radio (Tier 3) . . . 21
2.15.5 Tier 4 (Ultimate Software Radio (USR)) . . . 21
2.16 Applications of SDR . . . 21
2.17 Universal Software Radio Peripheral (USRP) . . . 22
2.18 UHD (USRP Hardware Driver) . . . 22
2.19 ANTENNAS . . . 22
2.20 OPEN BASE TRANSCIEVER STATION (OPENBTS) . . . 23

2.21 Brief History of OpenBTS 24
2.22 OpenBTS Components 24
2.22.1 OpenBTS 25
2.22.2 Asterisk 25
2.22.3 SMQueue 25
2.22.4 SIPAuthServe 25
2.23 OpenBTS vs Traditional GSM 25
2.24 Previous Works on GSM Projects 26

3 INSTALLATION AND IMPLEMENTATION 30
3.1 INTRODUCTION 30
3.2 Hardware Components 30
3.2.1 Computer 31
3.2.2 Software Define Radio 31
Table of USRP 32
3.2.3 Antennas 32
3.3 Software Components 33
3.3.1 OpenBTS Software 33
3.3.2 Operating System 34
3.3.3 Git 34
3.3.4 UHD 34
3.3.5 Asterisk 35
3.4 Project Setup Instructions 35
3.4.1 Initial Setup 35
3.4.2 Downloading the source code 36
3.4.3 Building and Installing UHD 37
3.4.4 Testing Built UHD 42
3.4.5 Building and Installing OpenBTS Components 43
3.4.6 Running OpenBTS Processes 50
3.4.7 Connecting a Mobile Device to the Network 52
3.4.8 Checking Device Registration 52
3.4.9 Creating a Subscriber 54

4 RESULTS AND ANALYSIS 58
4.1 Introduction 58
4.2 Results 58
4.3 Analysis 62
4.3.1 Infrastructure 62

5 CONCLUSION AND RECOMMENDATIONS 64
5.1 Problems Encountered/Limitations 64
5.2 Recommendations/Solutions 64
5.3 Conclusion 65

List of Figures

2.1 Evolution of Cellular Network 6
2.2 GSM Architecture 10
2.3 Mobile Station 11
2.4 Base Station Subsystem 12
2.5 NSS 14
2.6 SDR Receiver Block Diagram 18
2.7 SDR Transmitter Block Diagram 18
2.8 OpenBTS Architecture 24
2.9 Comparison Between OpenBTS and Traditional GSM Network 26

3.1 Terminal view 35
3.2 Updating Libraries 36
3.3 Cloning Git 36
3.4 Installing necessary components to build UHD 38
3.5 Versions of UHD available 38
3.6 Checking out to a compatible UHD version for B210 39
3.7 Creating a build folder 39
3.8 Response of cmake 40
3.9 Response from successfully building UHD 40
3.10 Installing UHD 41
3.11 Updating LD LIBRARY PATH 41
3.12 Save changes 42
3.13 Confirm UHD installation with uhd_find_devices 42
3.14 Installing library from Git 43
3.15 Library Completed downloading 43
3.16 Library downloaded 44
3.17 Running preinstall script 44
3.18 Asterisk installation 45
3.19 Bootstrapping software 45

3.20 Configuration 46
3.21 Asterisk is running 46
3.22 OpenBTS Installation 47
3.23 Generating Makefiles for all executables 47
3.24 OpenBTS configuration 48
3.25 File configuration 48
3.26 Make Installation 49
3.27 sudo installation 49
3.28 USRP Probe 50
3.29 Smqueue process 50
3.30 OpenBTS initialization 51
3.31 Client initialization 51
3.32 Help Center 52
3.33 Network Scan and Renaming 53
3.34 Accessing NodeManager for Device Configuration 54
3.35 Assigning phone number 55
3.36 Assigning phone number b 55
3.37 Fixing the python 2 error 56
3.38 UFT-8 Character Support 56
3.39 Subscriber Creation Command 57

4.1 Welcome SMS 59
4.2 Test Echo 60
4.3 Test to Unauthenticated Numbers 61

List of Tables

2.1 GSM Assets in Nigeria 8
2.2 Antennas 23

3.1 Recommended USRP 32

Abbreviation

1G First Generation
2G Second Generation
3G Third Generation
4G Fourth Generation
ADC Analog to Digital Converter
ARFCH Absolute Radio Frequency Channel
AuC Authentication Center
BSC Base Station Controller
BSS Base Station Subsystem
BTS Base Transceiver Station
CDMA Code Division Multiple Access
CEPT European Conference of Postal and Telecommunications Administrations
DAC Digital to Analog Converter
EIR Equipment Identity Register
FCC Federal Communications Commission
FCH Frequency Correction Channel
GMSC Gateway Mobile Switching Center
GSM Global System for Mobile Communication
HLR Home Location Register
ICT Information and Communication Technology
IEEE Institute of Electrical and Electronics Engineers
IMEI International Mobile Equipment Identity
IMSI International Mobile Subscriber Identity
ISDN Integrated Services Digital Network
LTE Long Term Evolution
ME Mobile Equipment
MIMO Multiple Input Multiple Output
MS Mobile Station
MSC Mobile Switching Center
MSISDN Mobile Station International Subscriber Directory Number
NCC National Communication Commission
NSS Network and Switching Subsystem
NTT Nippon Telegraph and Telephone
OSS Operation and Support Subsystem
PSTN Public Switched Telephone Network

RAM Random Access Memory
RF Radio Frequency
SCH Synchronization Channel
SDR Software Defined Radio
SIM Subscriber Identity Module
SMS Short Message Service
SNR Signal to Noise Ratio
UHD Universal Hardware Driver
USB Universal Serial Bus
USRP Universal Software Radio Peripheral
VLR Visitor Location Register
VoIP Voice over Internet Protocol

Chapter 1

INTRODUCTION

1.1 Background of Study

Humans have always been inherently social creatures, driven by the need to exchange information. This fundamental desire for communication has evolved from ancient methods, cave paintings and rudimentary signaling, to the sophisticated digital mobile networks of today. Among these, the Global System for Mobile Communications (GSM) still stands out, being a useful technology.

Developed by the European Telecommunications Standards Institute (ETSI), GSM is a digital cellular communication standard that defines the protocols for second-generation (2G) mobile networks. Initially deployed in Finland in December 1991, GSM has since become a cornerstone of global mobile communication infrastructure. As of 2023, Nigeria boasts approximately 224 million mobile cellular subscriptions, reflecting a significant increase from the previous year.

In Nigeria, the 2G network remains dominant, accounting for over 90

For sustainable development in rural areas, it is imperative to extend cellular network coverage. This initiative not only fosters economic growth but also facilitates innovation and enhances the quality of life by creating opportunities for local communities. In regions where connectivity is scarce, improved mobile services can stimulate productivity and bring about transformative changes. Chikanta and Mweetwa (2007) highlight that rural areas often face significant infrastructure challenges, including poor access to electricity, transportation, and telecommunications, which hinder overall devel-

opment. These disparities in resource availability perpetuate unequal access to opportunities that could enhance the daily lives of rural populations.

Advances in engineering have led to a reduction in the cost of electronic components, allowing for the production of more affordable mobile phones. However, other factors such as inadequate infrastructure, geographic limitations, and the lack of awareness of available opportunities continue to impede mobile service expansion in rural locales. The World Bank reports that mobile network operators often find it economically unfeasible to deploy services in such areas.

In Nigeria, many rural areas remain disconnected from the national electricity grid, predominantly consisting of sparsely populated regions with insufficient road networks. These challenges contribute to the slow pace of development and limited access to Information and Communication Technology (ICT) services. Deploying traditional GSM networks in these areas proves inefficient due to the large distances between villages and the sparsely populated nature of these regions. According to a 2011 National Geographic report, the poorest billion people globally own 22 mobile phones per 100 individuals, while only 1.2 computers are available per 100 people. Those living on less than USD 1,000 annually, most of whom reside in Africa, are part of this demographic, highlighting the reliance on mobile phones as their primary means of communication.

Traditional GSM infrastructure, such as Base Transceiver Stations (BTS), which are essential for network operation, are costly to establish and maintain. The installation of these massive infrastructures, which are also vulnerable to natural disasters such as earthquakes and storms, can result in prolonged periods of connectivity loss. In Nigeria, the cost of setting up a single BTS is between 40 million to 50 million Naira, excluding ongoing maintenance costs.

1.2 Problem Statement

Natural disasters, such as earthquakes, can lead to the destruction of vital communication infrastructure, such as base stations, leaving affected regions without means of communication. Furthermore, some remote villages and towns remain underserved due to the low income of the residents, which makes it financially unviable for network providers to extend services to these areas. This creates a technological divide, leaving rural populations in a

state of isolation, often referred to as the "Third World" state. Similarly, communication challenges also exist in isolated environments such as oil rigs, where communication between staff members is critical. These scenarios highlight the urgent need for viable communication solutions in underserved regions.

1.3 Aim

This project aims to design and implement a GSM network capable of making and receiving calls and SMS messages, carefully utilizing OpenBTS, Software Defined Radio (SDR), and a Unix-based operating system - Ubuntu 24.04.

1.4 Objectives

- To acquire an SDR (B210) and set up a Unix-based operating system.
- To install and configure OpenBTS, Asterisk, and SMSQueue on the Unix-based platform.
- To establish a fully functional GSM network enabling calls and SMS communication.
- To perform necessary tests to ensure network functionality.

1.5 Methodology

- Procurement of the B210 SDR and installation of the required software, beginning with the Unix-based operating system, specifically Ubuntu 24.04.
- Downloading, compiling, and configuring OpenBTS and Asterisk on the selected OS.
- Setting up the SDR in duplex mode and configuring the GSM network for communication.
- Final testing, including the successful establishment of calls and SMS functionality over the network.

1.6 Outline

This report consists of five chapters. Chapter one, as discussed above, simply introduces the study, outlining the problem statement, objectives, and the aim of the project. Chapter two presents a review of related literature and previous work on OpenBTS. Chapter three details the methodology adopted, including the process of software installation, system configuration, and data collection. Chapter four provides a comprehensive analysis of the implementation process, including the tests conducted and the corresponding results. Finally, chapter five concludes the report with key findings and recommendations for future work.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

The GSM Network, with subscribers worldwide of approximately ten billion[10], has become an essential component of mobile telecommunications infrastructure. It provides a solid and dependable platform for voice and data services.[12]

In this chapter, we will explore the underlying technologies and protocols that the GSM Network operates on. We will design, implement, and analyze the network, focusing on its architecture, components, and operational characteristics.

2.2 Cellular Networks

A cellular network, in simple terms, refers to a wireless communication network providing mobile phone coverage to a broad geographical area. It can also be described as a complex framework using a combination of cell towers, base stations (BSs), and user equipment (UE), collectively providing voice and data services to connected subscribers. These networks are divided into basic units called cells. These cells are either square- or hexagonal-shaped, and their sizes depend on the number of users in the area. Cell sizes tend to be larger in densely populated areas than in sparsely populated regions. Each cell is assigned a specific frequency band different from its neighboring cells, eliminating interference and ensuring proper service quality.[9]

A cellular network consists of a mobile station or user equipment capa-

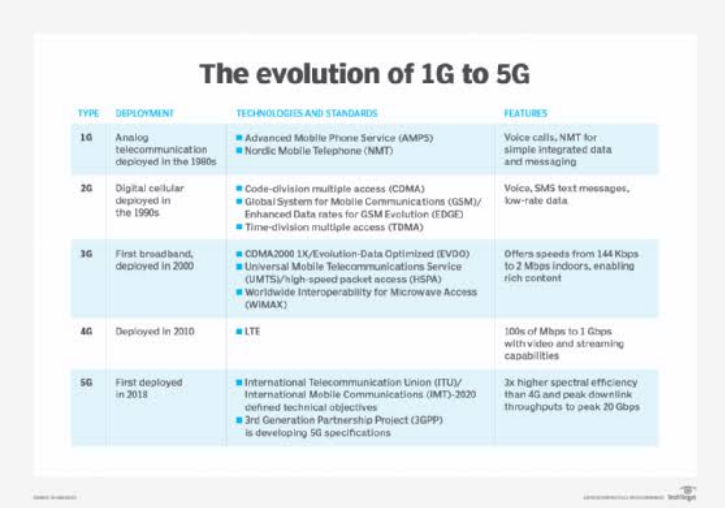


Figure 2.1: Evolution of Cellular Network

ble of transmitting and receiving signals to and from a fixed base station, which then relays the signal to a core network. Cellular networks offer several desirable features, including high capacity, extensive coverage, built-in security, mobility support, good quality of service (QoS), and effective network management.

2.3 History of Cellular Networks

Cellular networks have revolutionized everyday communication, providing mobile phone coverage to billions of people worldwide. The history of cellular networks is a fascinating story of innovation, technological advancements, and collaboration among industry players. figure 2.1.

2.3.1 Early Years

The concept of cellular networks dates back to the 1940s, when engineers at AT&T's Bell Labs began testing the idea of using radio waves for mobile phone service. [13]The first prototype of a cellular network emerged in the 1960s, using a system called the Advanced Mobile Phone System (AMPS), which marked the beginning of the first-generation (1G) network.[11]

2.3.2 First Generation Networks (1G)

The first commercial cellular network was launched in Japan in 1979, using 1G AMPS technology. This network provided analog voice services, which distinguished it from the later 2G digital networks. However, 1G networks suffered from capacity and coverage limitations. In the 1980s, 1G networks were deployed in Europe and the United States, marking the beginning of the cellular era.[15]

2.3.3 Second Generation Networks (2G)

The 1990s saw the introduction of 2G networks, which provided digital voice services with limited data capabilities. The most widely adopted 2G technology was GSM (Global System for Mobile Communications), developed in Europe to unify mobile communication standards across the continent. GSM became the de facto global standard for 2G networks. The first GSM network was launched in Finland in 1991.[8]

2.3.4 Third Generation Networks (3G)

With the limitations of 2G, the early 2000s saw the introduction of 3G networks, known as the era of mobile broadband. These networks provided faster data speeds and support for multimedia services. [14]The most prominent 3G technology was UMTS (Universal Mobile Telecommunications System), developed in Europe.

2.3.5 Fourth Generation Networks (4G)

The 2010s brought the development of 4G networks, offering significantly higher data speeds and support for high-definition video streaming. The most widely used 4G technology was LTE (Long-Term Evolution), developed by the 3GPP consortium. [4]

2.3.6 Fifth Generation Networks (5G)

5G networks emerged in response to increasing connectivity demands. Currently in their early stages, 5G networks aim to provide extremely fast data speeds and support a wide range of applications, including IoT, mission-critical communications, and enhanced mobile broadband.

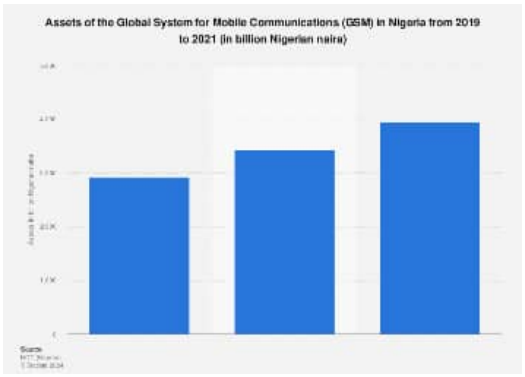


Table 2.1: GSM Assets in Nigeria

2.4 Why GSM

The GSM protocol is a global standard with coverage in over 200 territories and countries.This wide coverage makes the network widely available, with many MNOs providing GSM services.However its major selling point is its cost effectiveness, since GSM services are relatively low cost, making it affordable for carrying out many projects.Its scalability also makes it suitable for large and small projects alike.[7]Figure 2.1.

2.5 The GSM Network

The GSM project was initiated in 1987 by the European Conference of Postal and Telecommunications Administrations (CEPT) to create a unified, pan-European mobile communication standard. At the time, mobile communication systems were fragmented, with different countries using different standards, making cross-border roaming difficult. GSM addressed this issue by establishing a common standard for mobile communication across Europe.

The first GSM architecture was developed in the early 1990s, with the first GSM call made in 1991. The initial GSM architecture consisted of several key components:

- **Mobile Station (MS)**: The mobile device used by subscribers to access the GSM network.
- **Base Transceiver Station (BTS)**: The radio transmission and reception equipment that communicates with the MS.

- **Base Station Controller (BSC)**: Manages multiple BTSs and controls the allocation of channels and time slots.
- **Mobile Switching Center (MSC)**: The central component that manages multiple BSCs and connects the network to the Public Switched Telephone Network (PSTN).

In the late 1990s and early 2000s, the GSM architecture evolved with the introduction of General Packet Radio Service (GPRS), which provided packet-switched data services, enabling mobile internet access and multimedia messaging. This marked a transition from purely circuit-switched networks to packet-switched networks.

By the mid-2000s, 3G (UMTS) technology improved data speeds and overall performance. The GSM architecture continued evolving as many networks transitioned to 3G and later to 4G (LTE), which further enhanced data speeds, performance, and network capacity.

Today, GSM still plays an essential role in mobile communications, especially in regions where newer technologies like 5G are not yet available. However, GSM is no longer the dominant standard, as newer and more efficient technologies such as CDMA and LTE have taken precedence.

2.6 The GSM Architecture

The GSM network architecture is a complex system that provides mobile phone coverage to a wide geographic area. It is a digital cellular network that uses a mix of FDMA (Frequency Division Multiple Access) and TDMA (Time Division Multiple Access) technologies to provide multiple access to the network - figure 2.2.

Several key components make up the network architecture, including the Mobile Station (MS), Base Transceiver Station (BTS), Base Station Controller (BSC), Mobile Switching Center (MSC), Home Location Register (HLR), and Visitor Location Register (VLR).

When a mobile device (MS) is turned on, it registers with the nearest base transceiver station (BTS), which communicates with the mobile device. The BTS then authenticates the mobile device using the subscriber identity module (SIM) card and allocates a channel and time slot for the device.

Once the mobile device is authenticated and allocated a channel and time slot, it can then communicate with the network. The mobile switching center

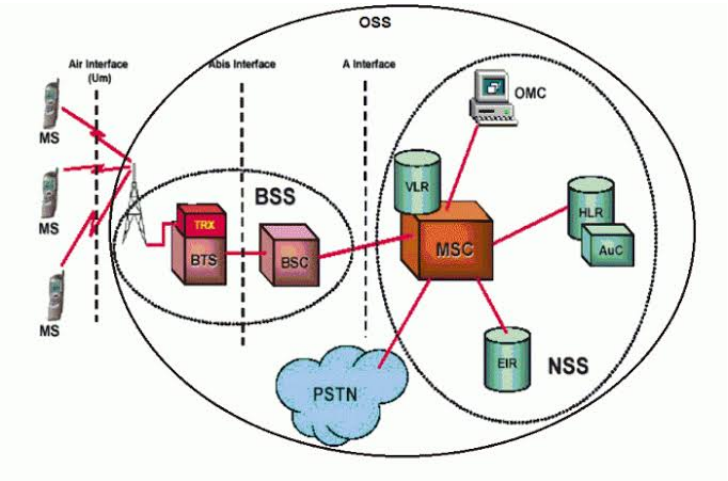


Figure 2.2: GSM Architecture

(MSC) is the central component of the GSM network that manages multiple BTSs and connects the network to the Public Switched Telephone Network (PSTN).

When a call is made, the BTS allocates a channel and a time slot to the subscriber. The signal is then transmitted via the BTS, through the MSC, and onto the PSTN to reach the subscriber being called. Once the subscriber terminates the call, the BTS deallocates the time slot and channel, and the MS returns to its idle state.

The GSM Network is classified into four major parts:

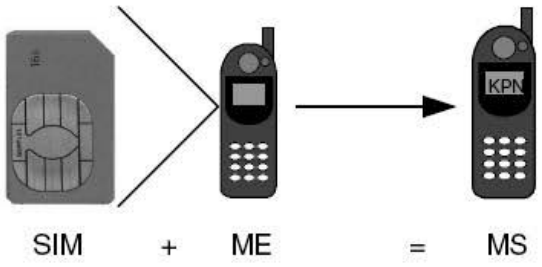
1. Mobile Station (MS)
2. Base-Station Subsystem (BSS)
3. Network and Switching Subsystem (NSS)
4. Operations and Support Subsystem (OSS)

2.7 THE MOBILE STATION (MS)

Mobile Station (MS) is a crucial component of a GSM network. It is the mobile device used by a subscriber to access the network and utilize its

various features. It is the section of a GSM mobile communications network that the user sees and operates

A Mobile Station consists of two main components: the Mobile Equipment (ME) and the Subscriber Identity Module (SIM). Figure 2.3



Components of the mobile station

Figure 2.3: Mobile Station

2.7.1 Mobile Equipment (ME)

The ME, or the user equipment (UE), is the physical device that contains the radio transceiver, antenna, screen, battery, case, and other necessary components. It is solely responsible for transmitting and receiving radio signals to and from the Base Transceiver Station (BTS). It also possesses a unique number, called the International Mobile Equipment Identity (IMEI), which is installed in the phone when produced and cannot be altered. The unique feature of the IMEI helps the network check if the device has been identified as stolen.

2.7.2 Subscriber Identity Module (SIM)

The SIM is a smart card that stores the subscriber's profile, including their phone number, account information, and security keys. It also contains a number unique to the SIM, called the International Mobile Subscriber Identity (IMSI). It is used to authenticate the subscriber and provide access to the GSM network.

2.8 THE BASE STATION SUBSYSTEM (BSS)

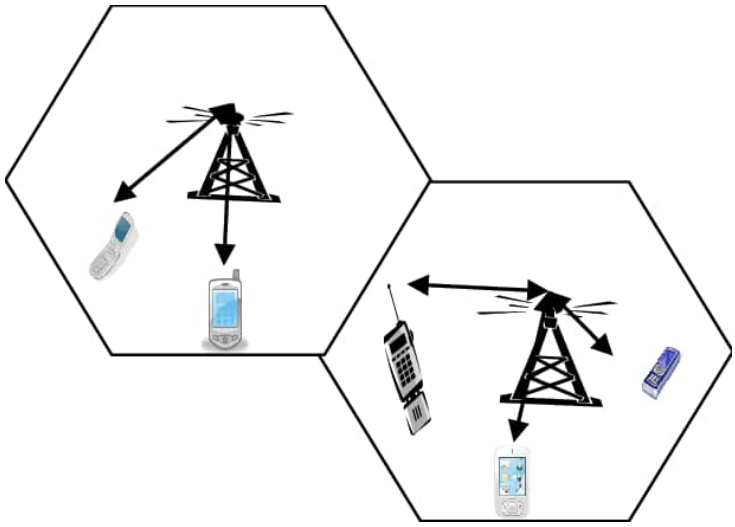


Figure 2.4: Base Station Subsystem

The Base Station Subsystem (BSS) is a critical component of a GSM (Global System for Mobile Communications) network. It provides the radio interface between the Mobile Station (MS) and the network, enabling communication between the MS and the network. The BSS is responsible for managing the radio resources, establishing communication channels, and handling the handover process between Base Stations. Figure 2.4

The **BSS** simply consists of two main components:

- **Base Transceiver Station (BTS)**
- **Base Station Controller (BSC)**

2.8.1 Base Transceiver Station (BTS)

The Base Transceiver Station (BTS) is the hardware component that communicates directly with the Mobile Station (MS). It contains the radio transceivers, antennas, and equipment necessary for radio transmission and reception. The BTS is responsible for establishing the radio connection with the MS, encoding and decoding voice or data signals, and relaying the signals to and from

the Base Station Controller (BSC). Additionally, the BTS handles the frequency planning, power control, and other critical tasks related to the radio interface.

2.8.2 Base Station Controller (BSC)

The Base Station Controller (BSC) is responsible for controlling the operation of multiple BTS units. It manages the allocation of radio resources for the BTS, handles call setup, mobility management, and handover between Base Stations. The BSC is connected to the Mobile Switching Center (MSC) and plays a key role in managing the handoff process as the Mobile Station (MS) moves from one BTS coverage area to another. The BSC helps to ensure the smooth operation of the BSS by optimizing network resources and ensuring efficient communication between MS and the network.

2.9 NETWORK AND SWITCHING SUBSYSTEM (NSS)

The Network and Switching Subsystem (NSS) is a critical component of a GSM network. It is responsible for managing the connections between the Mobile Stations (MS) and the Public Switched Telephone Network (PSTN), as well as between different mobile networks. The NSS is central to the operation of the GSM network, enabling seamless communication both within the mobile network and with external networks.figure 2.5

The **NSS** basically consists of several key components:

- **Mobile Switching Center (MSC)**
- **Gateway Mobile Switching Center (GMSC)**
- **Home Location Register (HLR)**
- **Visitor Location Register (VLR)**
- **Equipment Identity Register (EIR)**
- **Authentication Center (AuC)**

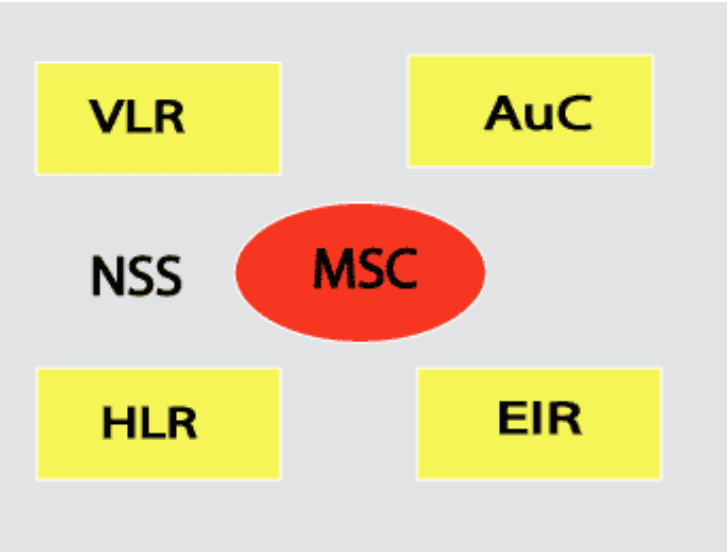


Figure 2.5: NSS

2.9.1 Mobile Switching Center (MSC)

The Mobile Switching Center (MSC) is a critical component of the Network and Switching Subsystem (NSS) in a GSM network. It is responsible for managing the switching of calls between the Mobile Stations (MS) and the Public Switched Telephone Network (PSTN). The MSC performs several key functions, including call setup, call routing, call switching, and many other tasks essential to maintaining communication.

2.9.2 Gateway Mobile Switching Center (GMSC)

The Gateway Mobile Switching Center (GMSC) is a specialized MSC that acts as a gateway between the GSM network and the PSTN. It is primarily responsible for routing calls between the GSM network and the PSTN. The GMSC also performs several key functions, such as call routing, call switching, and signaling, ensuring smooth communication between the two networks.

2.9.3 Home Location Register (HLR)

The Home Location Register (HLR) is a database that stores information about subscribers, including phone numbers, profiles, and current location. The HLR is responsible for managing the subscription information of mobile users. Functions such as subscriber authentication, subscription management, and location management are performed by the HLR, ensuring that subscribers can access network services regardless of their location.

2.9.4 Visitor Location Register (VLR)

The Visitor Location Register (VLR) is a database that stores information about subscribers who are currently roaming in a particular area. The VLR is responsible for managing the location information of roaming subscribers. Other functions, such as location management, subscriber authentication, and subscription management, are carried out by the VLR to enable roaming subscribers to access services.

2.9.5 Equipment Identity Register (EIR)

The Equipment Identity Register (EIR) is a database that stores information about mobile devices, including their International Mobile Equipment Identity (IMEI) numbers. It helps to identify and block stolen or unauthorized mobile devices from accessing the GSM network. The EIR contributes to network security by preventing theft and unauthorized use of mobile devices, thereby reducing fraud and improving the integrity of the GSM network.

2.9.6 Authentication Center (AuC)

The Authentication Center (AuC) is a database that stores security information, including encryption keys and authentication algorithms. The AuC authenticates mobile devices and ensures secure communication between the device and the GSM network. By providing secure authentication and encryption, the AuC protects mobile devices and the GSM network from unauthorized access and eavesdropping.

2.10 OPERATIONS AND SUPPORT SUBSYSTEMS (OSS)

The Operations and Support Subsystems (OSS) is a vital component of a GSM network, playing a relevant role in the management and maintenance of the network's operations. This subsystem is also responsible for ensuring the network's reliability, performance, and security, ultimately providing high-quality service to subscribers.

The OSS consists of several components, including the Network Management System (NMS), Element Management System (EMS), and Operations Support System (OSS). The NMS provides a centralized management platform for the network, while the EMS manages individual network elements, such as base stations and switches. The OSS provides a suite of tools for managing the network, including fault management, configuration management, and performance management, ensuring optimal functioning of the network at all times.

2.11 SOFTWARE DEFINED RADIO

Software Defined Radio (SDR) is not a new technology. It has been almost 30 years since the first SDR was introduced. Fast forward to the present, it is becoming widely adopted as a means of building wireless devices today.

A number of definitions over time have been found to describe Software Defined Radio, some of which are:

- A Software-Defined Radio (SDR) is a radio communication system where most of the signal processing functions such as modulation, demodulation, filtering, and encoding are implemented in software rather than hardware. This makes SDR highly flexible and adaptable compared to traditional hardware-based radios.
- A software-defined radio (SDR) is a radio system that uses software to process signals instead of traditional hardware.
- A software-defined radio (SDR) system is a radio communication system which uses software for the modulation and demodulation of radio signals.

- Software-defined radio (SDR) is a radio communication system where components that conventionally have been implemented in analog hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a computer or embedded system.

2.12 BRIEF HISTORY OF SDR

SDR technology has come a long way and below we give a brief history of how it came about.

In 1982, while working under a US Department of Defense contract at RCA, Ulrich L. Rohde's department developed the first SDR, which used the COSMAC (Complementary Symmetry Monolithic Array Computer) chip. Rohde was the first to present on this topic with his February 1984 talk, "Digital HF Radio: A Sampling of Techniques" at the Third International Conference on HF Communication Systems and Techniques in London.

In 1984, a team at the Garland, Texas, Division of E-Systems Inc. (now Raytheon) coined the term "software radio" to refer to a digital baseband receiver, as published in their E-Team company newsletter. A 'Software Radio Proof-of-Concept' laboratory was developed by the E-Systems team that popularized Software Radio within various government agencies. This 1984 Software Radio was a digital baseband receiver that provided programmable interference cancellation and demodulation for broadband signals, typically with thousands of adaptive filter taps, using multiple array processors accessing shared memory.

In 1991, Joe Mitola independently reinvented the term software radio for a plan to build a GSM base station that would combine Ferdensi's digital receiver with E-Systems Melpar's digitally controlled communications jammers for a true software-based transceiver. E-Systems Melpar sold the software radio idea to the US Air Force.

In 1991, The first military program that required its physical layer components to be implemented in software was the DARPA's SPEAKeasy. It originated from the U.S. Air Force and its main objective was to have a single radio that could support ten different military radio protocols and operate anywhere between 2 MHz and 2 GHz.

In 1991, Mitola described the architecture principles without implementation details in a paper, "Software Radio: Survey, Critical Analysis and

Future Directions" which became the first IEEE publication to employ the term in 1992. He is referred to by many as the godfather of software radio. He also introduced the term "cognitive radio" for intelligent radios.

In 2001, GNU Radio was developed by Eric Blossom from a framework called PSpectra. GNU Radio serves as an open-source framework for the development of SDR applications within a computer system. As of 2012, it became the most popular SDR development toolset for users.

In 2006, Texas Instruments, Xilinx, and Nutaq joined forces to create the first stand-alone, completely integrated SDR equipped with an ARM, a DSP, an FPGA, and a front-end tunable from 200MHz to 1GHz.

In 2009, Lime Microsystems unveiled LMS6002 and LMS6002D, Radio Frequency Integrated Circuits (RFIC) which could tune between 400MHz and 4GHz, and supported 28MHz of bandwidth.

2.12.1 SDR Operation

The block diagram of an SDR is given below, figure 2.6 and 2.7:

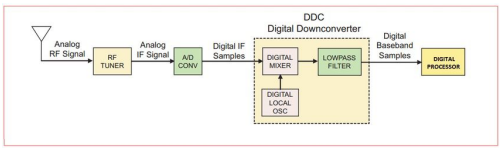


Figure 2.6: SDR Receiver Block Diagram

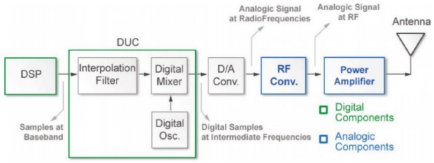


Figure 2.7: SDR Transmitter Block Diagram

ANTENNA

- The smart antenna provides a gain versus direction characteristic to minimize noise, multipath, and interference.
- The antenna acts as an interface between the radio waves and the SDR system.
- It receives incoming RF signals or transmits processed signals.

- Different antennas may be used depending on the frequency range and application (e.g., GSM, LTE, Wi-Fi).
- A smart antenna is an antenna array system helped by some "smart" algorithm designed to adapt to different signal environments. Smart antennas and software-defined radio work well with each other. Software radios provide the flexibility needed for effective smart antenna operation.

2.13 RF Hardware

The desired RF signals are filtered and then amplified to an adequate level. The amplified RF signals are transferred to the digitization system. It prepares the signal for further digital processing and boosts weak signals using a Low Noise Amplifier (LNA) to improve signal quality. It also removes unwanted frequencies to avoid interference.

2.14 ADC/DAC

ADC (Analog-to-Digital Converter) – For Reception

- Converts the filtered and amplified analog RF signals into digital signals for processing.
- The sampling rate and bit resolution determine the SDR's performance.

DAC (Digital-to-Analog Converter) – For Transmission

- Converts processed digital signals back into analog form for transmission.
- The signal is then sent to the RF front end for upconversion and amplification before transmission.

For more efficient processing of signals, it is required that on the receiver side, the ADC is closer to the antenna and so there is early conversion from analog to digital signals. For a transmitter, conversion from digital to analog signals is delayed using the DAC. Superheterodyne receivers could be used.

Channelization and Sample Rate Conversion

Channelization involves the selection of the appropriate channel in which the desired signal transmitted is required. This is due to the presence of multiple access allowing the transmitted signal as well as unwanted signals to be received.

Baseband Processing

Baseband processing refers to the digital signal processing (DSP) operations applied to the signal after it has been converted from RF to baseband (low-frequency or zero-frequency). It is where key communication tasks like modulation, demodulation, filtering, and error correction occur before transmission or after reception. Baseband processing happens in the DSP block, which runs on FPGA, DSP, CPU, or GPU. GNU radio companion is used to create the flow graphs which are used to process the signal digitally. This forms a typical model of software defined radio. One of the key issues of the baseband processor is the amount of processing power required. The greater the level of processing, the higher the current consumption and in turn this requires additional cooling, etc.

2.15 Tiers of Software Defined Radio

The SDR Forum established five tiers which encompass different categories of Software Radio systems and they are:

2.15.1 Hardware-based Radios (Tier 0)

There is no software control over signal processing. Entirely hardware-based (fixed modulation, filtering, and frequency). Example: Legacy analog radios, early GSM handsets.

2.15.2 Software Controlled Radios (Tier 1)

In this tier, the software controls some radio parameters, but signal processing is hardware-based. RF front end and baseband processing remain in dedicated ASICs or DSPs. Example: Early 2G and 3G base stations with some reconfigurable parameters.

2.15.3 Reconfigurable Software Defined Radio (Tier 2)

In this tier, most baseband processing (modulation, demodulation, filtering) is done in software. RF front-end still relies on hardware mixers, amplifiers, and filters. Uses FPGAs, DSPs, and CPUs to execute signal processing algorithms. Example: GNU Radio + USRP-based SDRs, military radios with waveform updates.

2.15.4 Ideal Software Radio (Tier 3)

All signal processing, including RF mixing and filtering, is done in software. Requires high-speed ADC/DAC that can directly sample RF signals. Fully reconfigurable and supports multiple wireless standards dynamically. Example: Cognitive Radios, advanced military SDRs, experimental 6G systems.

2.15.5 Tier 4 (Ultimate Software Radio (USR))

Theoretical concept where everything, including antenna tuning, is software-defined. No fixed hardware constraints; fully software-driven from RF to baseband. Extremely high-speed ADC/DAC would be required for direct RF sampling. These are defined for comparison purposes only. It accepts fully programmable traffic and control information and supports a broad range of frequencies, air-interfaces, and applications software. It can switch from one air interface format to another in milliseconds, use GPS to track the user's location, store money using smartcard, etc

2.16 Applications of SDR

The use cases of the SDR is enormously plenty, and having several hundreds of applications, some of which are:

- Digital Video Broadcasting
- A cellular GSM Base Station
- GPS receiver
- Digital Television encoder
- FM radio transmitter and receiver

2.17 Universal Software Radio Peripheral (USRP)

The Universal Software Radio Peripheral (USRP) is a flexible, high-performance software-defined radio (SDR) hardware platform developed by Ettus Research (a National Instruments company)[5]. It is widely used for wireless research, prototyping, and education in applications such as GSM, LTE, Wi-Fi, radar, and cognitive radio. Most USRPs connect to a host computer through a high-speed link, which the host-based software uses to control the USRP hardware and transmit/receive data. Some USRP models also integrate the general functionality of a host computer with an embedded processor that allows the USRP device to operate in a stand-alone fashion. The USRP family was designed for accessibility, and many of the products are open source hardware. The board schematics for select USRP models are freely available for download; all USRP products are controlled with the open source UHD driver, which is free and open source software. USRPs are commonly used with the GNU Radio software suite to create complex software-defined radio systems.

2.18 UHD (USRP Hardware Driver)

The USRP Hardware Driver (UHD) is a software framework developed by Ettus Research, entirely providing a standardized API for controlling Universal Software Radio Peripheral (USRP) devices. It effortlessly allows users to interface USRP hardware with different SDR software like GNU Radio, MATLAB, LabVIEW, and OpenBTS.

2.19 ANTENNAS

An antenna is a device that transmits or receives electromagnetic waves (radio signals). It acts as a bridge between radio frequency (RF) signals traveling in free space and electrical signals in a communication system. It converts electrical signals into radio waves and radiates them into free space. It captures radio waves from the air and converts them into electrical signals for processing. They are used to send and receive data from several USRPs and can also be found in various communication applications such as radio broadcasting, radars, cell phones, etc. Some kinds of antennas are listed in the figure 2.2 below.

Model	Operating Frequency range	shape
LP0410	400 MHz to 1 GHz	
LP0965	850 MHz to 6.5 GHz	
Vert400	144 MHz , 400 MHz And 1200 MHz	
Vert900	824 to 960 MHz And 1710 to 1990 MHz	
Vert2450	2.4 to 2.48 GHz And 4.9 to 5.9 GHz	

Table 2.2: Antennas

2.20 OPEN BASE TRANSCIEVER STATION (OPENBTS)

The construction of an actual base station takes a lot of time and is very expensive. The cost of building a base station will cost a Mobile Network Operator (MNO), N40 million (TheGuardian 2018). This is where the idea of implementing a cost-effective GSM base station came from. Emergency GSM services can now be available in rural areas, developing countries, hard-to-reach locations such as oil rigs, and areas affected by a disaster (such as tsunami) where the existing GSM infrastructure has been destroyed. Implementation of a GSM base station requires two main components: a Software Defined Radio and the OpenBTS software.

OpenBTS is an open-source software-defined radio (SDR) implementation of a GSM network, allowing standard mobile phones to connect without requiring a traditional telecom operator. It acts as a software-based GSM base station, replacing the hardware-based BTS in conventional cellular networks.

2.21 Brief History of OpenBTS

OpenBTS was created by David A. Burgess and Harvind Samra at Range Networks. The goal was to reduce the cost of GSM infrastructure and enable software-based mobile networks. The project was based on GNU Radio and USRP (Universal Software Radio Peripheral). In 2010, the founders launched a company Range Networks to commercialize OpenBTS-based products and deploy networks worldwide. Since then, it has been tested in various environments such as Defcon, Burning Man, RELIEF exercises, and on the Island of Niue.

2.22 OpenBTS Components

OpenBTS software consists of a lot of components that work together to implement a cellular GSM network with the use of a Software Defined Radio - figure 2.8. The core components that make up OpenBTS software include:

- OpenBTS
- Asterisk
- SMQueue
- SIPAuthServe

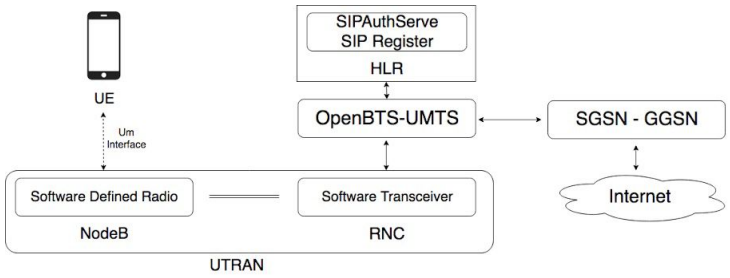


Figure 2.8: OpenBTS Architecture

2.22.1 OpenBTS

OpenBTS functions as a software-defined GSM base station, allowing standard mobile phones to communicate over a private or experimental GSM network. Unlike traditional GSM networks, OpenBTS replaces the Base Station Controller (BSC) and Mobile Switching Center (MSC) with VoIP-based call routing (SIP).

2.22.2 Asterisk

Asterisk is an open-source private branch exchange (PBX) software that enables VoIP (Voice over IP), call routing, and telephony services. It is commonly used in OpenBTS networks to replace the Mobile Switching Center (MSC) in traditional GSM networks, allowing calls and SMS to be handled over IP networks using SIP (Session Initiation Protocol). It communicates with the subscriber registry database, which stores subscribers' phone numbers, identities, authentication, caller IDs, and registration state.

2.22.3 SMQueue

SMQueue (Short Message Queue) is the SMS handling subsystem in OpenBTS. It manages the storage, routing, and delivery of SMS messages between mobile users and external systems. It acts like an SMSC (Short Message Service Center) in traditional GSM networks.

2.22.4 SIPAuthServe

SIPAuthServe is the authentication and call routing server in OpenBTS. It acts as a subscriber database and SIP registrar, replacing traditional GSM authentication (HLR/VLR). It works alongside Asterisk to handle call authorization and routing using SIP (Session Initiation Protocol).

2.23 OpenBTS vs Traditional GSM

Below, figure 2.9, we will see how OpenBTS components replace components of the existing traditional GSM Network.

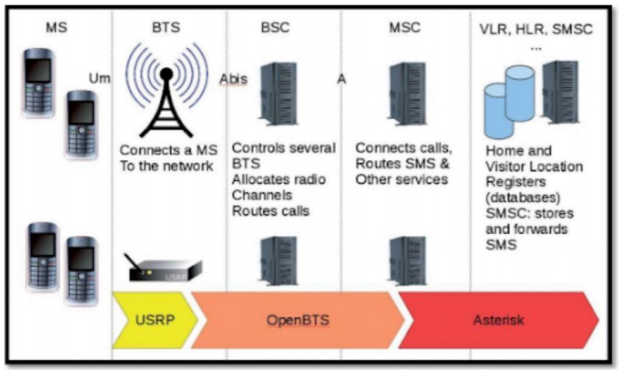


Figure 2.9: Comparison Between OpenBTS and Traditional GSM Network

- A USRP can easily perform the same function as the GSM BTS (Base Transceiver Station).
- OpenBTS handles all protocols carried out traditionally by the BSC and MSC, which includes node switching within a PSTN, processing requests for service from mobile devices, subscriber registration, authentication, network handover, channel allocation, and all the necessary interfaces.
- Asterisk handles call initiations in the network and implements the functions of the MSC.
- SMQueue implements the functions of the SMSC.
- SIPAuthServe and SMQueue implement functions of the HLR, VLR, AuC, and SMS-G such as user authentication, IMSI authentication, and message handling.

2.24 Previous Works on GSM Projects

Santiago Aragon & Federico Kuhlmann (2015)[1] conducted a man-in-the-middle attack implementation on a GSM network, exploiting its lack of mutual authentication between the subscriber and the network. With the help of a Software Defined Radio, open-source libraries, and hardware, they were able to set up a fake GSM Base Station to impersonate the network, eavesdrop on communications routed through it, and extract vital information

from their victims. This fatal flaw proved that the GSM protocol was obsolete and risky for vital communications, taking into account the availability of more advanced technologies. The mitigations suggested are short-term since a greater number of users are still connected to MNOs using the GSM network to date.

[6]Luis A.G Gomez, Ivan R.S Casella & Samuel C Pereira (2019) proposed a control system for a Doubly Fed Injection Generator (DFIG) of a wind turbine, according to the smart grid concept. The proposed wireless network to carry out the task was a Software Defined Radio connected to an OpenBTS Platform, which was to send a message containing the control information remotely either by the OpenBTS platform at the control center (CC) or by a wireless device connected to the network. The receiver on the turbine side was developed in a microcontroller board with a GSM card, and the DFIG controller was implemented in a Digital Signal Processor (DSP) board. The results of the tests showed that the DFIG can be satisfactorily controlled according to the power (Active and Reactive) references, considering that they vary at low rates.

Ms. Shivanjali Nigade (2024) designed and implemented GSM-based smart switches, offering users the ability to control electrical appliances remotely using GSM networks. This approach provides users with a cost-effective, efficient, and accessible solution for home automation and appliance management. These devices incorporate a SIM card to facilitate communication through SMS commands and phone calls, enabling remote appliance control with ease. Users can activate or deactivate appliances remotely, including lighting, security gates, and doors. This concept not only fosters a deeper understanding of electronics and GSM networks but also highlights their potential to contribute to an improved quality of life, making it a subject of great interest within the spheres of IoT, automation in homes, and agriculture firms.

Kinjal Aggrawal & Khyati P. Vachhani (2017)[2] presented a cost-effective technique for quick reinstatement of wireless connectivity in disaster-hit areas, which face complete breakdown of prevailing infrastructure. A test reconfigurable GSM Base Transceiver System (BTS) based on Software Defined Radio (SDR) is built using the Universal Software Radio Peripheral (USRP) B200 board and OpenBTS. USRP B200 overcomes the necessity of an external 10 MHz reference signal, unlike the widely used N-series and USRP1.

After establishing the cellular GSM network, short message services and voice calling are executed. Besides OpenBTS, standalone typical applications

such as SMQueue and Asterisk are also used for the correct routing of messages. This setup also extends connectivity to practically every individual in disaster-hit areas with a GSM handset at no further cost.

Arusha Dubey, Deepak Vohra, Khyati P. Vachhani, Arvind Rao (2016)[3] demonstrated the vulnerabilities in GSM security with USRP B210 and open-source penetration tools. This paper showcases the vulnerabilities in the GSM security architecture through the implementation of an active attack at the Um interface. The attack was carried out by taking advantage of the lack of two-way authentication. A rogue GSM Base Transceiver System (BTS) was established using the USRP B200 board and OpenBTS. USRP B200 allows relaxation of an external 10 MHz reference signal, as opposed to the widely used USRP1 and N-series.

After establishing the rogue BTS, IMSI catch-attack and impersonation of a mobile subscriber to send malicious SMS were executed. Along with OpenBTS, standalone standard applications such as Asterisk and SMQueue are used for the correct routing of messages. The attacks were observed on the TEST network and not the spoofed network so that no infringement was established on the security and privacy of the existing GSM subscribers.

As seen, studies have explored the deployment of GSM cellular networks using OpenBTS. In the success of the implementations, previous work supervised by Engr. Dr. N. Bello utilized **Ubuntu 16.04** as the operating system alongside OpenBTS. This approach was foundational in demonstrating the feasibility of establishing GSM communication through software-defined radio (SDR) platforms like the **USRP B210**. However, the reliance on an older operating system version limited software compatibility and security updates.

Building upon this, supervised by Engr. Abdul-Rahman Dauda and Engr. Dr. N. Bello, there was an improvement by modifying the **UHD (USRP Hardware Driver)** image. By updating the UHD firmware, the study achieved enhanced signal stability and reduced latency, although the underlying Ubuntu version remained at **16.04**, thus not fully addressing long-term support and compatibility issues.

In the present work employing **Ubuntu 24.04**, we represent a significant advancement over previous methodologies. Unlike earlier processes, the use of this newer operating system offers better hardware compatibility, security enhancements, and support for updated SDR libraries. The implementation process also diverges entirely from prior approaches, requiring adjustments in software dependencies, configuration protocols, and signal

processing pipelines. These modifications not only improve network reliability but also facilitate future scalability and maintenance.

Chapter 3

INSTALLATION AND IMPLEMENTATION

3.1 INTRODUCTION

In this chapter, we provide a comprehensive guide for setting up a cellular network using OpenBTS and Software Defined Radio on the latest version of Linux, Ubuntu 24.04lts. We go through hardware selection, selecting the right operating libraries and repositories, setting up the development environment, compiling, installing, and configuring the various components of OpenBTS from scratch.

At each stage, we explain each tool or software we used and why. At the end of the chapter, we will have a test GSM network setup where subscribers can scan, connect to, register on, make calls, and even exchange text messages.

3.2 Hardware Components

Although OpenBTS implements most of the traditional GSM stack (which are hardware based) with software, there is a need for a couple of hardware components necessary for a functional setup of the transmission station. For example, the radio waves must be transmitted and received by the network, and the OpenBTS software has to run on hardware. The various hardware components we used are:

- Computer

- Software Defined Radio
- Antenna

3.2.1 Computer

The first requirement is a computer. However, it can be anything from a Raspberry Pi to a full-fledged desktop PC - provided it meets the hardware requirements. From our test, we used a laptop with the following specifications:

- **64-bit processor:** Backed from research, we saw lots of examples where a system with a 32-bit processor was used, and it worked well too. However, to follow ideally up with our setup, you need to have a system with a 64-bit processor.
- **16GB RAM:** The minimum requirement for RAM cannot be explicitly set because of many variables involved. A 1GB RAM device might work just fine, but for optimum performance, we made use of a 16GB RAM device in our setup.
- **USB 3.0 interface:** A USB 3.0 interface is recommended for optimum performance as the Software Defined Radio used in this setup (USRP B210) works faster with that. All the same, a USB 2.0 interface will work just fine but might generate 'Buffer Overflow' errors.
- **Intel Core i5, 1.75 GHz quad-core processor:** To support the running of processes using multiple cores for maximal optimization.
- **64GB Hard Disk space**

3.2.2 Software Define Radio

As explained earlier, the SDR plays the role of the BTS in the traditional GSM architecture. OpenBTS supports SDRs from various vendors, but we decided to choose an SDR within the Ettus Research USRPs. Ettus Research is one of the leading suppliers of software-defined radios[5].

It can be deduced from the below table, figure 3.1, USRP N-series are recommended for working with OpenBTS. Going for the Ettus Research B210 USRP was the best simply because of the following reasons:

Application Area	Common USRP Model	Common Daughterboard
PHY/MAC Research	N200/N210 X300/X310	WBX/SBX/UBX/CBX
Radar Research	X300/X310	SBX/UBX
OpenBTS	B200/B210 X300/X310 E310/E312 N200/N210	WBX/SBX/UBX/CBX
Amarisoft LTE	N200/N210 X300/X310	WBX/SBX/UBX/CBX
Education	B200/B210 X300/X310 E310/E312 N200/N210	WBX/SBX/CBX/UBX
HF Communications	N200/N210 X300/X310	LFRX/LFTX
Signals Intelligence	X300/X310	SBX/UBX
Distributed RF Sensors	E310/E312	N/A
Mobile Radios	E310/E312	N/A
MIMO	X300/X310	SBX/UBX
Phased Array	X300/X310	SBX/UBX
FPGA Computing	X310	WBX/SBX/UBX/CBX
Embedded Computing	E310/E312	N/A
Small Form Factor (SWaP)	B200mini/B205mini E310/E312	N/A

Table 3.1: Recommended USRP

- Cost
- Full duplex MIMO (2 Tx and 2 Rx) operation
- Fast USB 3.0 connectivity
- Frequency coverage from 70MHz - 6GHz
- It has an inbuilt reference signal and thus does not require an external clocking device unlike the N-series.
- GNU Radio and OpenBTS support through the UHD (USRP Hardware Driver) which is open source.

3.2.3 Antennas

SDRs have transmit and receive sensitivity to operate without antennas in a very small coverage area of about 1m. We need the network to have more coverage area and decided to add a pair of 3dBi antennas which expanded the coverage area up to 10m radius in an area without obstructions. The antenna used was the VERT900 provided by Ettus Research. It has a rubber-duck style with a subminiature version A (SMA) connector. Other essential characteristics of the VERT900 included:

- 824 MHz to 960 MHz frequency range
- 3dBi gain

figure 2:

3.3 Software Components

The test cellular network to be set up is mostly software-based. The following software components were used:

- OpenBTS software
- Operating system
- Git
- UHD
- Asterisk

3.3.1 OpenBTS Software

OpenBTS is vert elemental in this project, acting as the core software, implementing the GSM protocol stack. It's the bridge between the radio hardware (USRP) and the rest of the cellular network infrastructure. Here's a breakdown of its role, based on this experimental context:

- GSM Protocol Stack Implementation: OpenBTS is a software suite that implements the lower layers of the GSM protocol stack. This means it handles the radio-specific functions, such as modulation, demodulation, channel access, and call setup/teardown.
- Base Transceiver Station (BTS) Role: In a traditional GSM network, the BTS is the hardware component responsible for communicating with mobile phones. OpenBTS allows you to create a BTS using a software-defined radio (SDR) and a computer, making it a flexible and cost-effective solution.

3.3.2 Operating System

OpenBTS has been tested to work on Linux-based Ubuntu Long-Term Support (LTS) distributions. Ubuntu 12.04 Precise Pangolin LTS is recommended for setting up OpenBTS according to openbts.org. We didn't work with it because it is an old version of Ubuntu. Our goal was to precisely set up OpenBTS to properly function with Ubuntu 24.04 LTS, but we stumbled upon problems as some OpenBTS dependencies are not compatible with said version of Ubuntu.

We ensured to avoid the use of Docker to properly debug, analyze, and optimize the usage of OpenBTS on the latest version of the Operating system to present a detailed walkthrough on the setup using an updated guide.

Ubuntu can be run as the main OS of your device or via VirtualBox. **Note:** A fresh install of Ubuntu 24.04 is recommended for this project to get appropriate results.

3.3.3 Git

Git is a free and open-source distributed version-control system for tracking changes in software source code. It was developed in 2005 by Linus Torvalds. The source files of most of the software components of OpenBTS are stored in repositories on Github (a web-based hosting service for version control using Git) and to access them, we need to have Git installed. Later in this chapter, we explain how to set it up.

3.3.4 UHD

The UHD (USRP Hardware Driver) tool is a software framework developed by Ettus Research for interfacing with Universal Software Radio Peripheral (USRP) devices, which are widely used in software-defined radio (SDR) applications. On Ubuntu, UHD provides the necessary drivers and APIs to control USRP hardware, enabling users to transmit and receive radio signals across a wide range of frequencies. UHD provides the necessary control used to transport user waveform samples to and from USRP hardware as well as control various parameters (e.g., sampling rate, center frequency, gains, etc.) of the radio. UHD also has a default configuration setup for the majority of working SDRs and simplifies the synchronization processes allowing for easier setup of the transceiver.

3.3.5 Asterisk

The Asterisk tool is an open-source framework used for building communication applications, including VoIP systems. In network configuration on Ubuntu for OpenBTS, Asterisk acts as a software PBX (Private Branch Exchange) to manage call routing, voicemail, and other telephony features. It integrates with OpenBTS, which provides GSM cellular network functionality, allowing Asterisk to handle calls and SMS messages over a GSM network. By configuring Asterisk with OpenBTS, users can create a complete GSM network using standard SIP (Session Initiation Protocol) and RTP (Real-time Transport Protocol) for voice and data transmission, enabling cost-effective, scalable, and customizable mobile communication solutions.

3.4 Project Setup Instructions

3.4.1 Initial Setup

The first thing that needs to be done is to update all existing libraries on your Ubuntu 24.04 LTS to their latest versions to avoid any bugs that may arise due to deprecated version use.

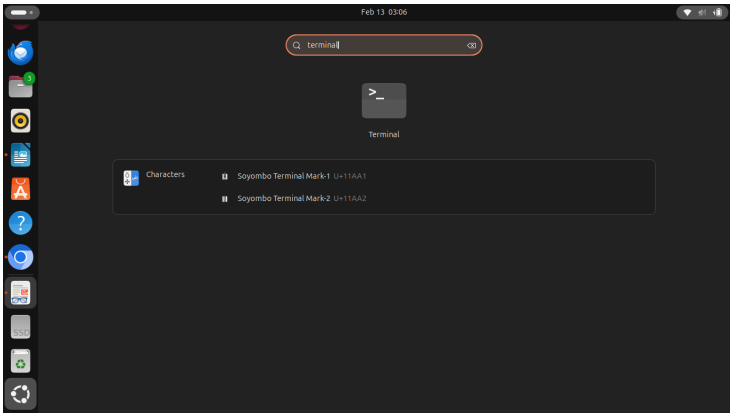


Figure 3.1: Terminal view

Open your terminal and run the following command to update existing libraries.

```
sudo apt-get update
```

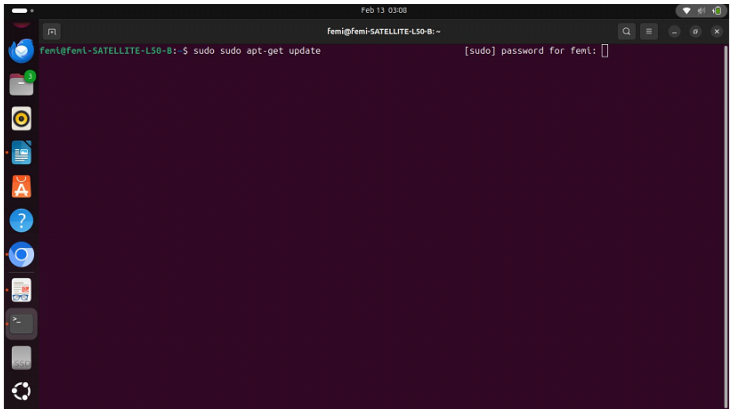


Figure 3.2: Updating Libraries

3.4.2 Downloading the source code

To download the source code needed for the system, you need to have Git installed. If it isn't available on your system, run the command below:

```
sudo apt-get install git
```

Once that has been installed, clone Range Network's repository to your computer.

```
git clone https://github.com/RangeNetworks/dev.git
```

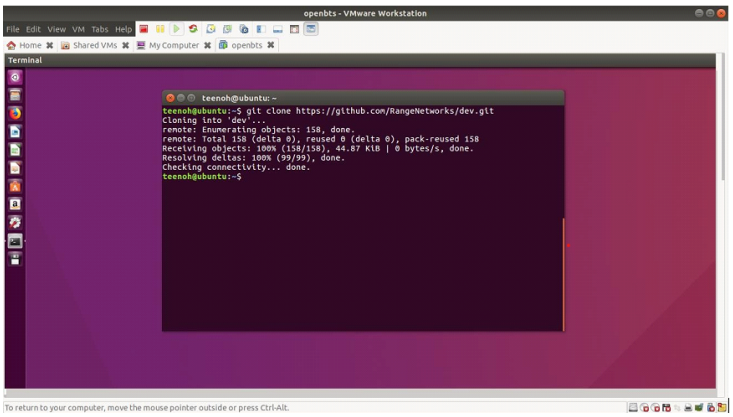


Figure 3.3: Cloning Git

The command above downloads development scripts to ease the process of setting up OpenBTS into a directory called `dev`.

3.4.3 Building and Installing UHD

The B210 is incompatible with UHD versions above 3.9. The latest version of UHD (3.13 as of this writing) is installed by default when you install OpenBTS dependencies. As a result, we need to build UHD from scratch to choose the version we want to work with. To build UHD from scratch, run the command below to install the necessary dependencies. It is important to note that the dependencies for Ubuntu 24.04 are vastly different from earlier versions as most dependencies have changed and OpenBTS, previously written with Python 2, must now be configured to support Python 3.

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y git swig cmake doxygen build-essential \
    libboost-all-dev \
libtool libusb-1.0-0-dev libudev-dev libncurses5-dev \
    libfftw3-bin \
libfftw3-dev libcppunit-dev ncurses-bin cpufrequtils \
    python3-numpy \
python3-scipy python3-docutils qtbase5-dev qtchooser \
    qttools5-dev-tools \
libqt5core5a libqt5gui5 libqt5widgets5 libqt5qml5 python3-\
    pyqt5 \
libqwt-qt5-dev libfftw3-doc libfontconfig1-dev libxrender-\
    dev \
libpulse-dev g++ automake autoconf python3-dev libgsl-dev \
python3-mako python3-lxml libusb-1.0-0-dev libzmq3-dev \
    python3-zmq \
libcomedi-dev python3-requests python3-sphinx liborc-0.4-\
    dev \
libasound2-dev python3-tk gtk2-engines-pixbuf wget libxi-\
    dev
```

Once you have them installed, clone Ettus Research's UHD repository to your system by running the command:

```
git clone https://github.com/EttusResearch/uhd.git
```

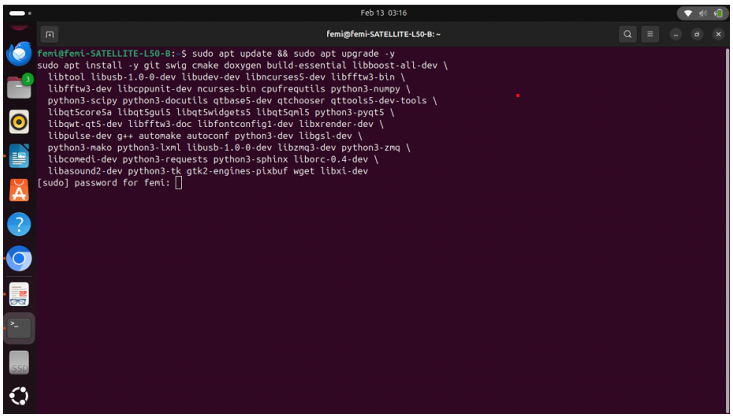


Figure 3.4: Installing necessary components to build UHD

Move into the `uhd` directory and list all the versions available.

```
cd uhd/
git tag   l
```

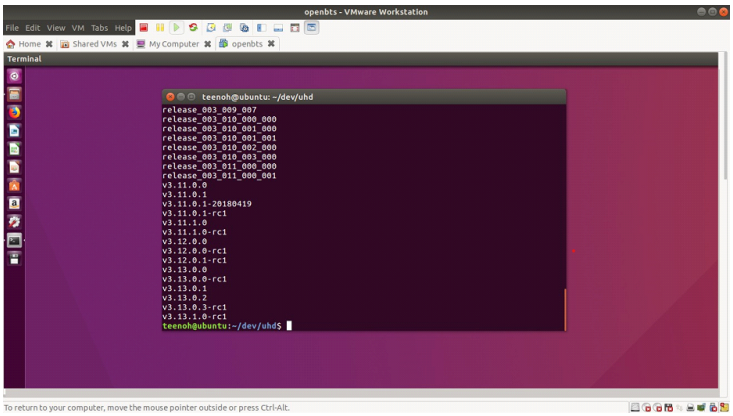


Figure 3.5: Versions of UHD available

The command `git tag -l` lists all the versions of UHD available. Recall, as mentioned earlier, we need to get a version less than or equal to 003.009.000, so we check out to release 003.008.005 since that is the last release before 003.009.000 and thus it is compatible with our B210.

```
git checkout release_003_008_005
```

Create a build folder within the `uhd` repository.

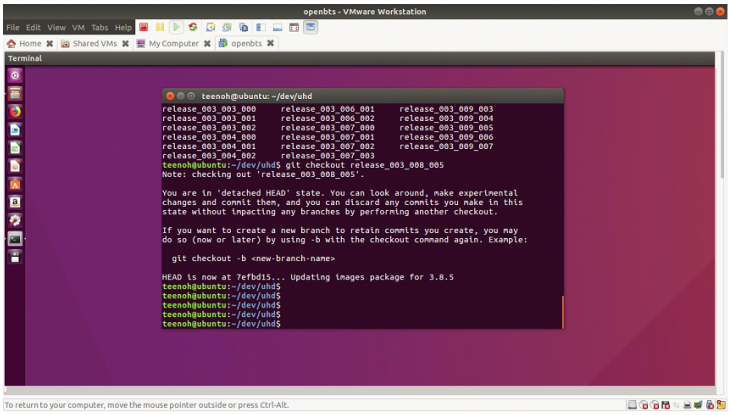


Figure 3.6: Checking out to a compatible UHD version for B210

```
cd host/
mkdir build/
cd build/
```

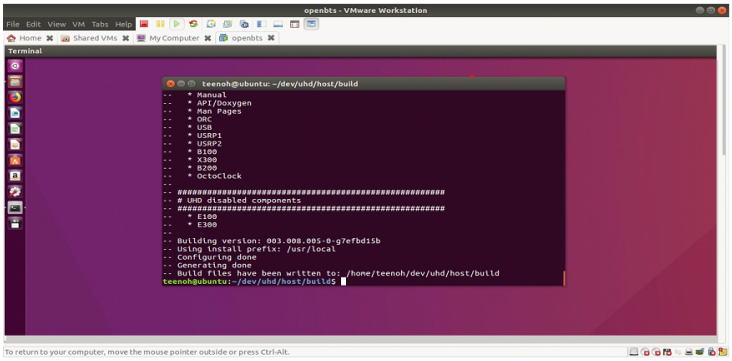


Figure 3.7: Creating a build folder

Invoke CMake to create the Makefiles.

```
cmake ../
```

Next, run `make` to build UHD. This stage takes about 30 - 60 mins.

```
sudo make
```

Next, install UHD with the command below. UHD will be installed under the `/usr/local/lib/` folder. You need to run the command as root due to the permission requirements on the `/usr/local/lib/` folder.

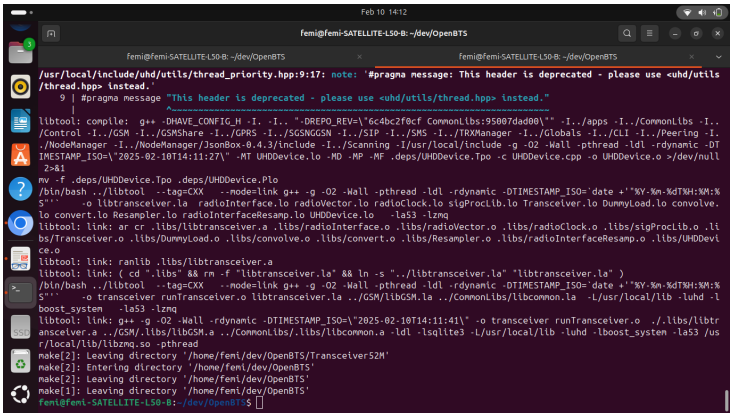


Figure 3.8: Response of cmake

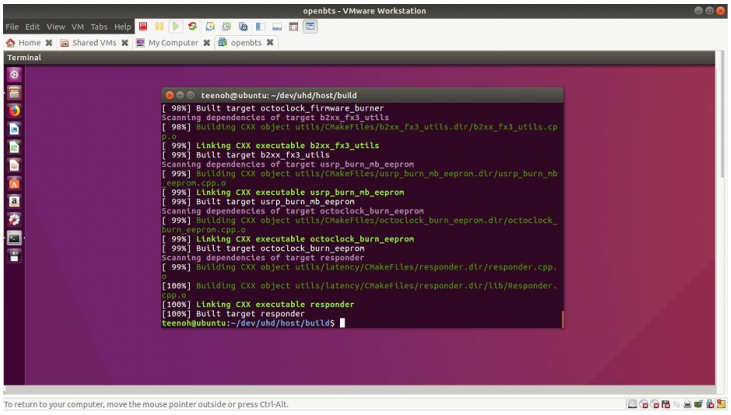


Figure 3.9: Response from successfully building UHD

```
sudo make install
```

Once it has been completed, update the system's shared library cache with:

```
sudo ldconfig
```

Finally, make sure that the `LD_LIBRARY_PATH` environment variable is defined and includes the folder under which UHD was installed.

To do this, open your `.bashrc` file with:

```
nano ~/.bashrc
```

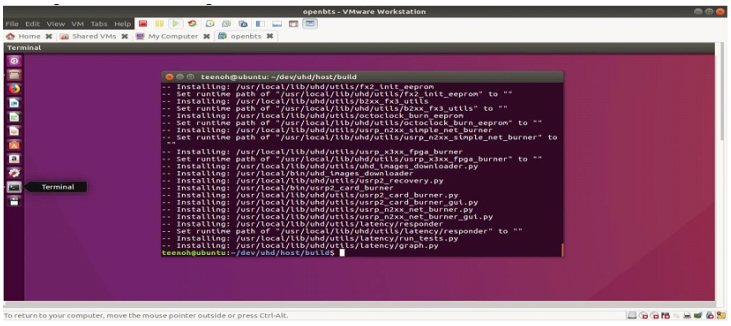


Figure 3.10: Installing UHD

`nano` is a simple terminal-based text editor, and it allows you to add this line of code to the end of the file:

```
export LD_LIBRARY_PATH=/usr/local/lib/
```

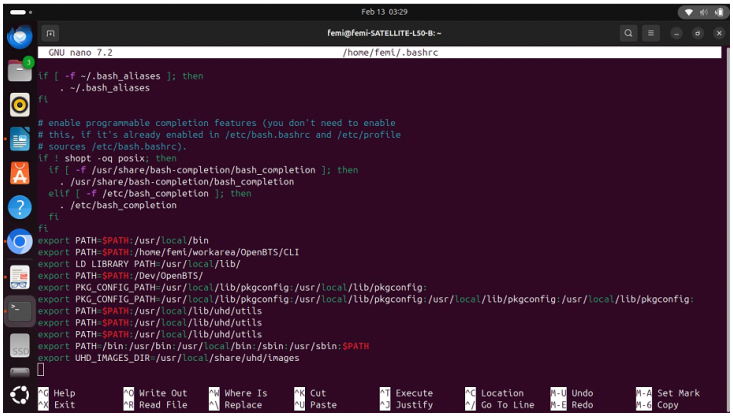


Figure 3.11: Updating LD LIBRARY PATH

Save Environment Variable Configuration To save the environment variable, press `Ctrl + O` and then the `Enter` key. You can exit the prompt by pressing `Ctrl + X`.

The `LD_LIBRARY_PATH` environment variable at this point will only be exported when your system is restarted. To update the settings immediately without restarting, use the command below:

```
source ~/.bashrc
```

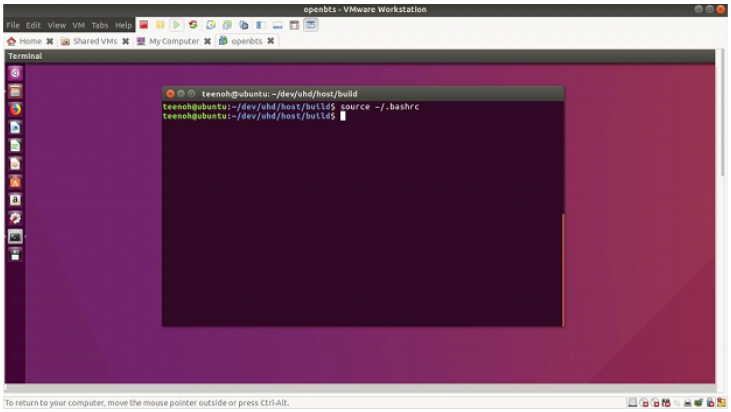


Figure 3.12: Save changes

3.4.4 Testing Built UHD

To confirm that UHD has been successfully installed, run:

```
uhd_find_devices
```

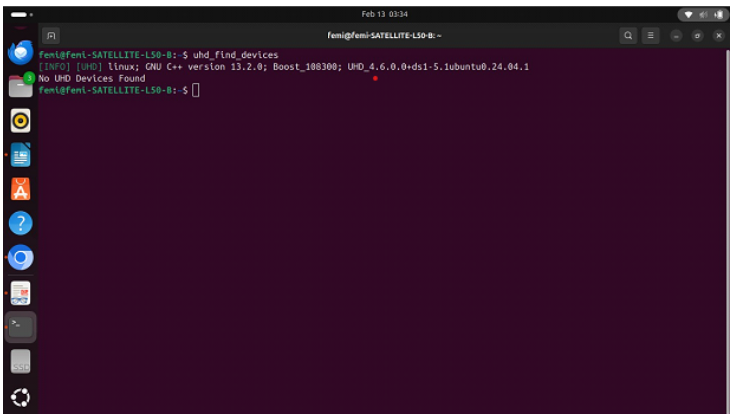


Figure 3.13: Confirm UHD installation with uhd_find_devices

If installed correctly, you'll get a response similar to the figure above. Note that the response indicates no B210 connection yet.

3.4.5 Building and Installing OpenBTS Components

With UHD installed, proceed to build OpenBTS. First, navigate to the `dev` folder:

```
cd ~/dev
```

Clone the OpenBTS repository:

```
git clone https://github.com/PentHertz/OpenBTS.git
```

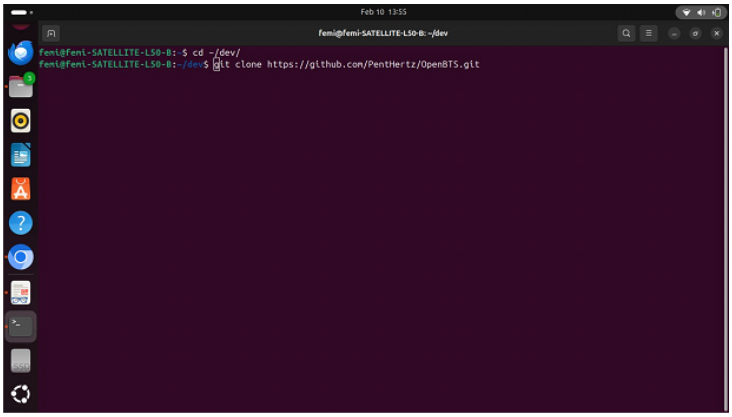


Figure 3.14: Installing library from Git

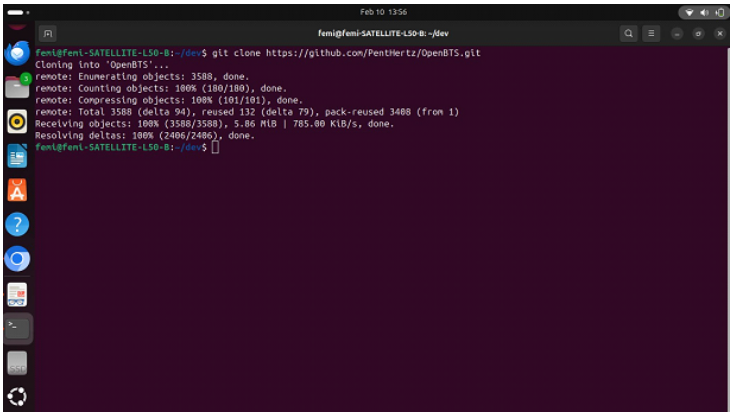


Figure 3.15: Library Completed downloading

Navigate to the cloned OpenBTS folder:

```
cd OpenBTS
```

Checkout the latest supported version:

```
git checkout 5.1.0
```

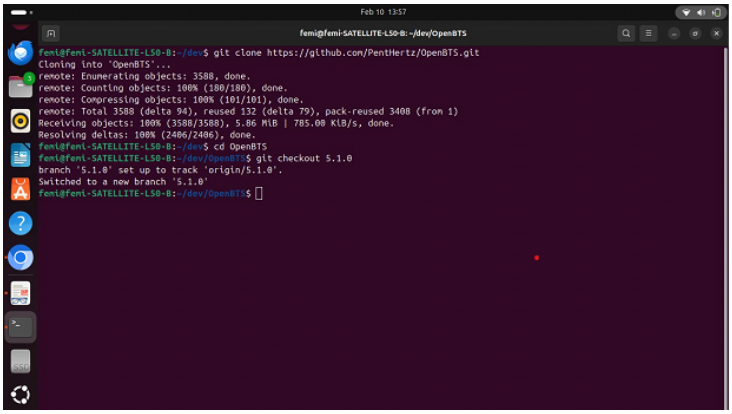


Figure 3.16: Library downloaded

Run the pre-install script to install dependencies:

```
./preinstall.sh
```

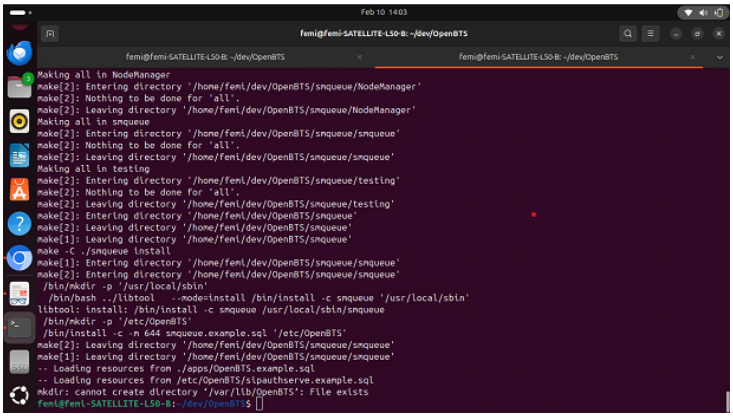


Figure 3.17: Running preinstall script

If `asterisk` doesn't install correctly, navigate to its directory to resolve issues:

```
cd ~/dev/OpenBTS/asterisk
```

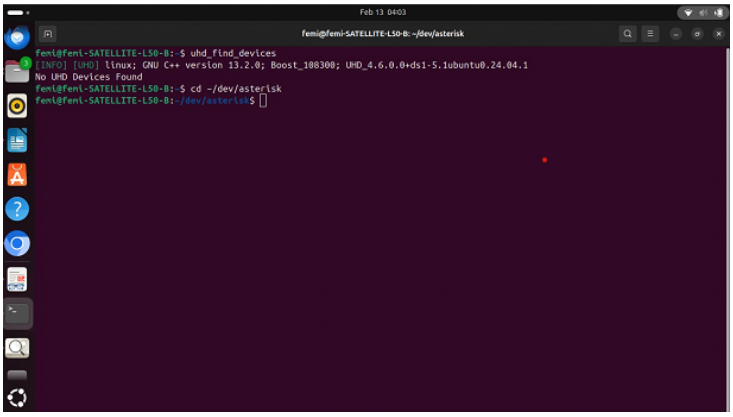


Figure 3.18: Asterisk installation

Generate necessary files using:

```
sudo ./bootstrap.sh
```

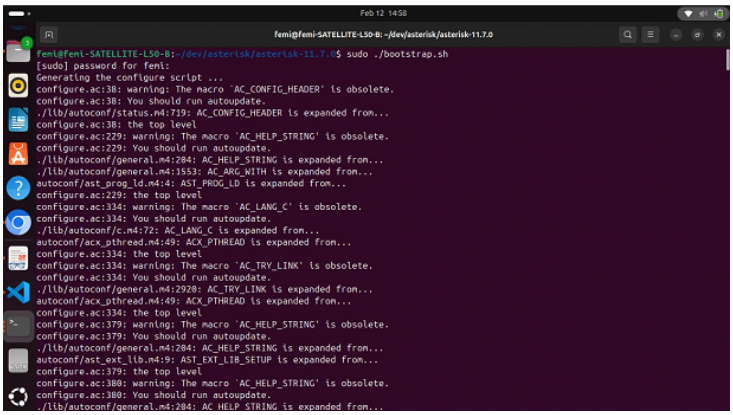


Figure 3.19: Bootstrapping software

Configure the setup for your OS and hardware:

```
sudo ./configure --with-uhd
```

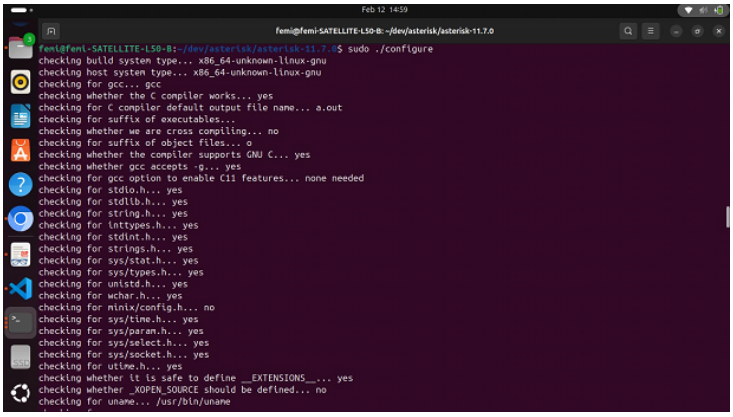


Figure 3.20: Configuration

Once the configuration is complete, the process is started, and the status is inspected to confirm it runs properly:

```
sudo systemctl start asterisk
sudo systemctl status asterisk
```

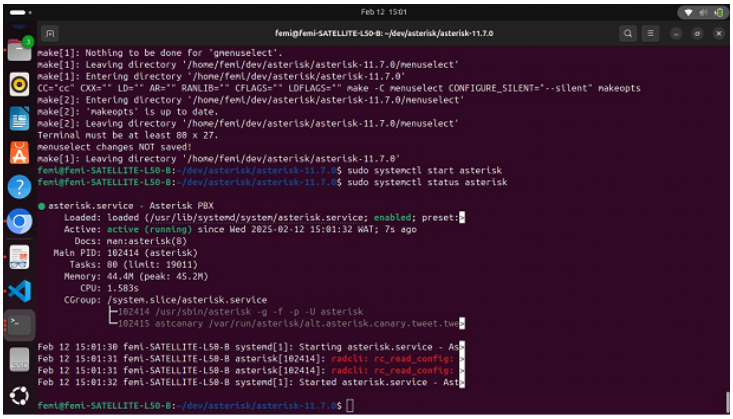


Figure 3.21: Asterisk is running

Navigate to the OpenBTS folder:

```
cd ~/dev/OpenBTS
```

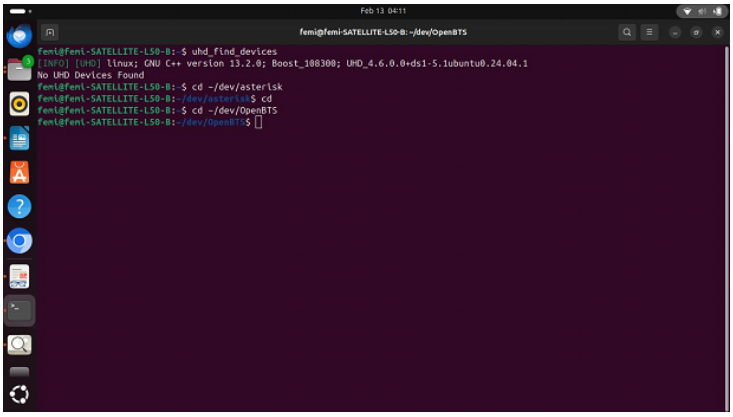


Figure 3.22: OpenBTS Installation

Now, generate the makefile configuration with the prepared script:

```
sudo ./autogen.sh
```

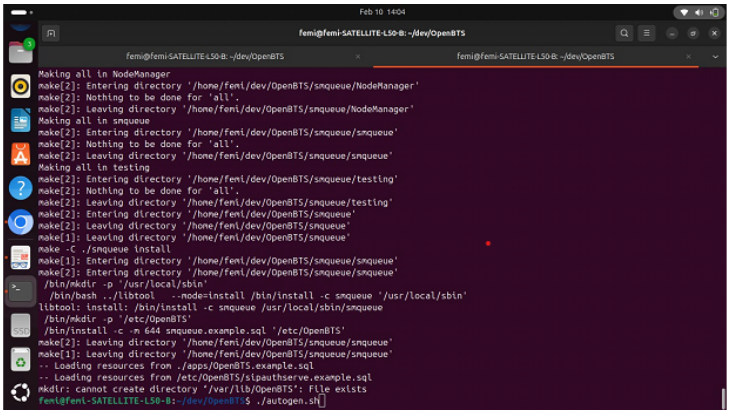


Figure 3.23: Generating Makefiles for all executables

This process takes about 5 minutes to complete.

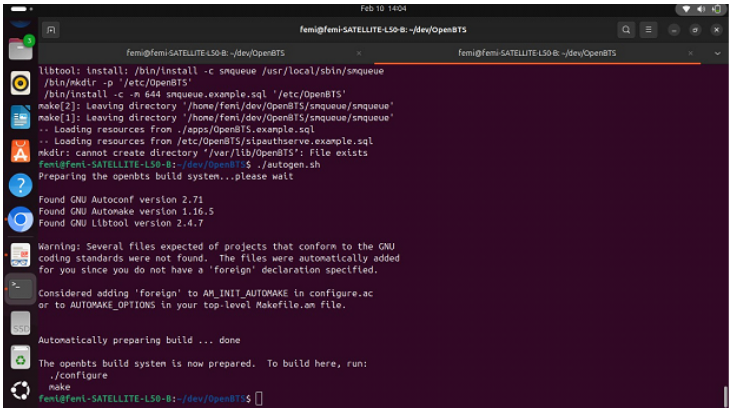


Figure 3.24: OpenBTS configuration

Then, configure the setup to run with OpenBTS. By default, it selects the RAD1 transceiver, but since we are using the USRP B210, we select the 52M transceiver:

```
sudo ./configure --with-uhd
```

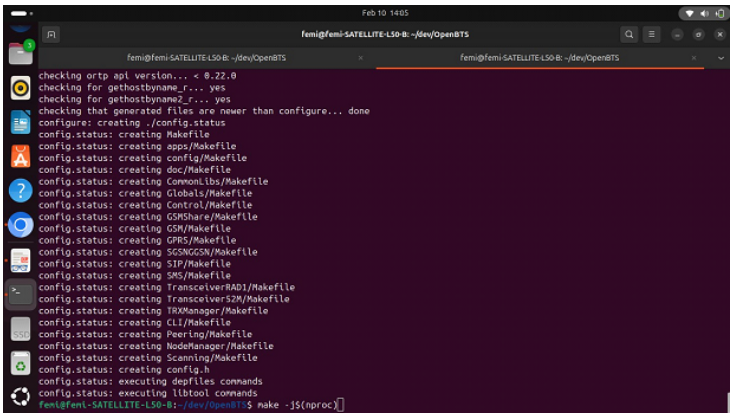


Figure 3.25: File configuration

Build the executable files using multiple processors to speed up the process:

```
make -j$(nproc)
```

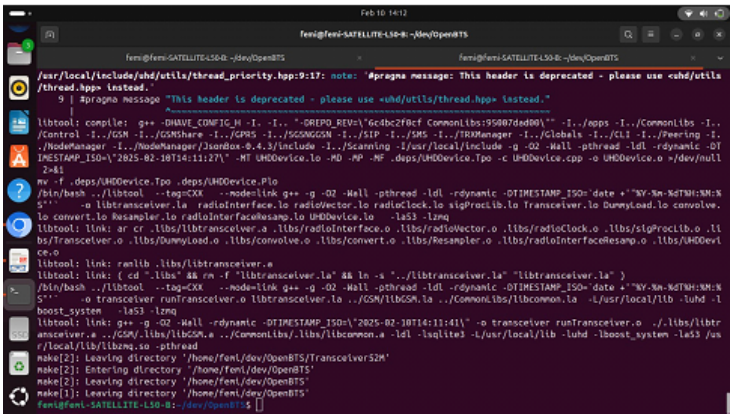


Figure 3.26: Make Installation

Once the build process is complete, install the files:

```
sudo make install
```

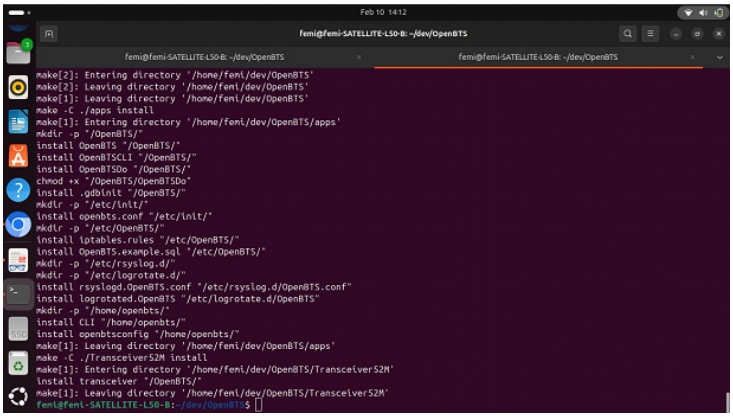


Figure 3.27: sudo installation

Finally, perform the final configuration check:

```
sudo ldconfig
```

Navigate to the Transceiver52M folder to facilitate transceiver detection under Ubuntu 24.04's permission restrictions. Probe the B210 for synchronization information:

```
uhd_usrp_probe
```

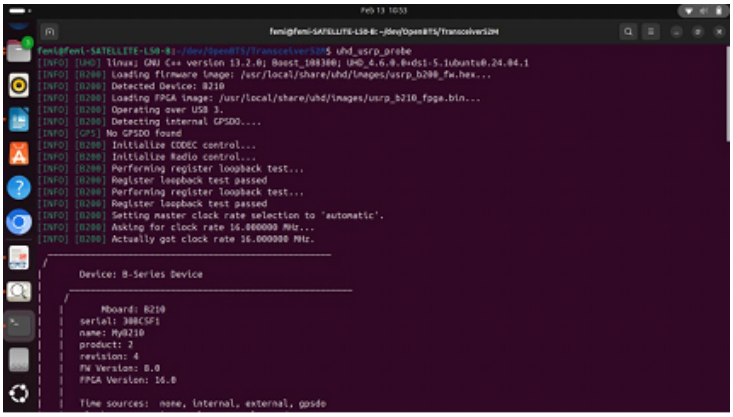


Figure 3.28: USRP Probe

3.4.6 Running OpenBTS Processes

To ensure proper protocol operation, start Asterisk as a background process and confirm its status. Then, initiate the ‘smqueue‘ client for SMS features and ‘sipauthserve‘ for IMSI registration and verification:

```
sudo smqueue
sudo systemctl start sipauthserve
```

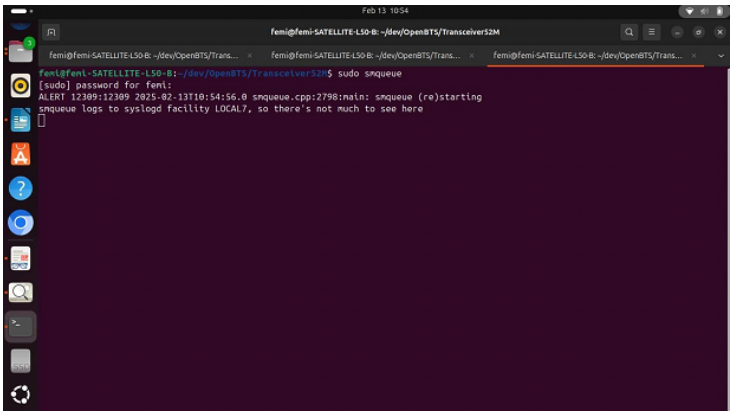


Figure 3.29: Smqueue process

Start the OpenBTS application and Client Initialization:

```
sudo    /OpenBTS/OpenBTS
```

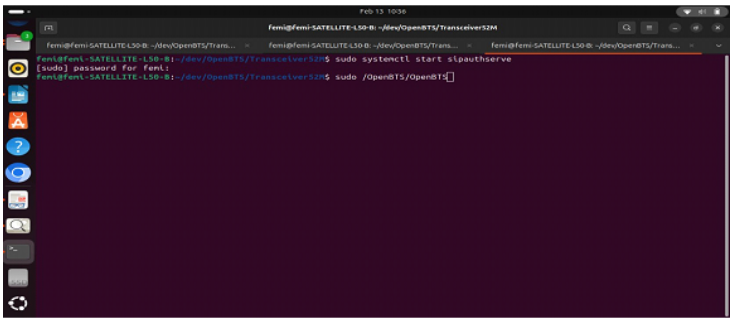


Figure 3.30: OpenBTS initialization

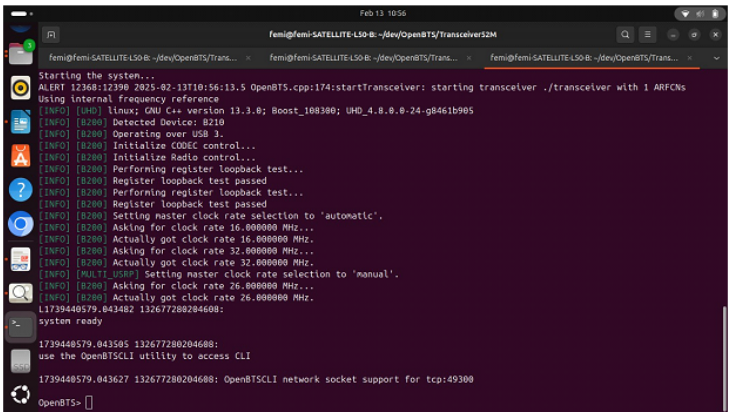


Figure 3.31: Client initialization

To view supported commands type:

```
help
```

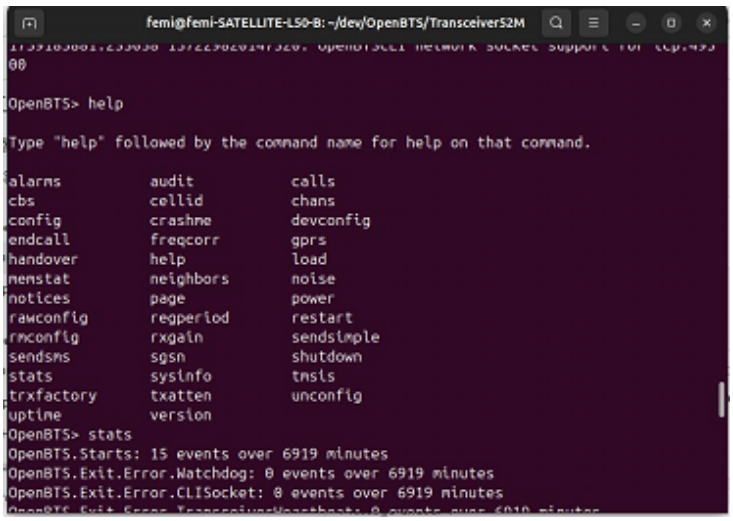


Figure 3.32: Help Center

3.4.7 Connecting a Mobile Device to the Network

Joining the Network:

Open the settings app on your phone and search for networks manually. A network named **00101** (for non-iOS devices) or **Test PLMN 1-1** (for iOS devices) should appear. This is the test GSM network we just set up. Attempt to connect to it.

Due to higher permission requirements, the network configuration should be opened to ensure devices can connect and be authenticated on the network. Devices are made to scan for networks under the "network settings" section.

To ensure ease of connection, the network is renamed to a more readable format from the default **001010**. Devices are used to scan for available networks and connect upon detection.

3.4.8 Checking Device Registration

The following command outputs a list of the most recent devices that have tried to register on the network:

```
tmsis
```

The first row contains data about the device we just registered. Notice the column **AUTH**, which checks if a device is authorized to be on the network:

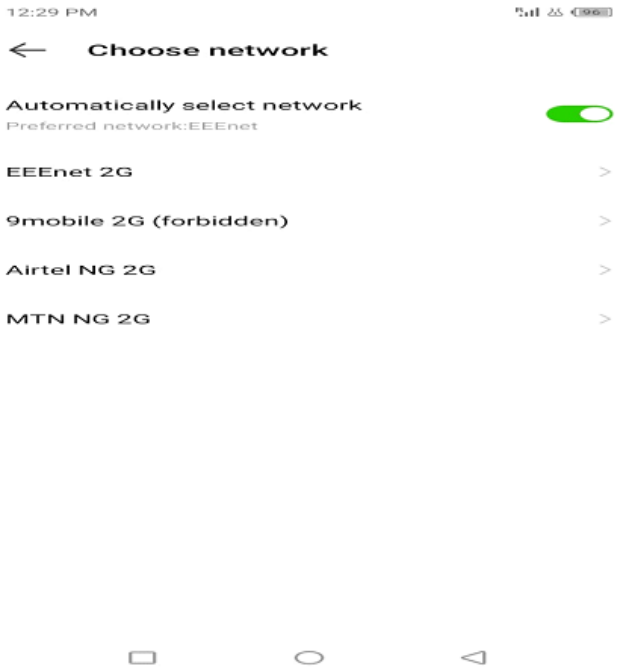


Figure 3.33: Network Scan and Renaming

- 0: Not authorized
- 1: Authorized (can easily join the network)
- 2: Registered but not yet configured (common due to opened configuration settings)

Note:

- **IMEI** – International Mobile Equipment Identity (specific to your mobile device)
- **IMSI** – International Mobile Subscriber Identity (specific to your SIM card)

To make the device join the network, we must authorize the device and to do this, we need the IMSI number since it's the sim card that's joining the network and not the mobile device.

To enable the client to configure connected devices, make use of the following command:

```
tmsis
```

This generates the details of devices that are connected to the network, and the devices should receive a welcome message as well.

3.4.9 Creating a Subscriber

To allow a user to join the network, a subscriber account must be created using the IMSI number obtained from the previous step. Open a new terminal tab:

```
Ctrl + Shift + T
```

Navigate to the `NodeManager` directory:

```
cd /dev/OpenBTS/NodeManager/
```

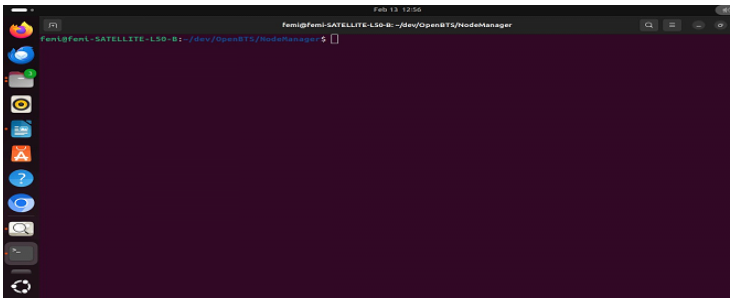


Figure 3.34: Accessing NodeManager for Device Configuration

To create a subscriber, run the following command format:

```
sudo ./nmcli.py sipauthserve subscribers create <name> IMSI\
    <imsi> <msisdn>
```

Where:

- $< name >$: Any name you wish to assign to the subscriber.
- $< imsi >$: IMSI value obtained from the `tmsis` command.
- $< msisdn >$: Phone number to assign to the subscriber.

Example:

```
sudo ./nmcli.py sipauthserve subscribers create "test" \
    IMSI621300602088772 7780
```

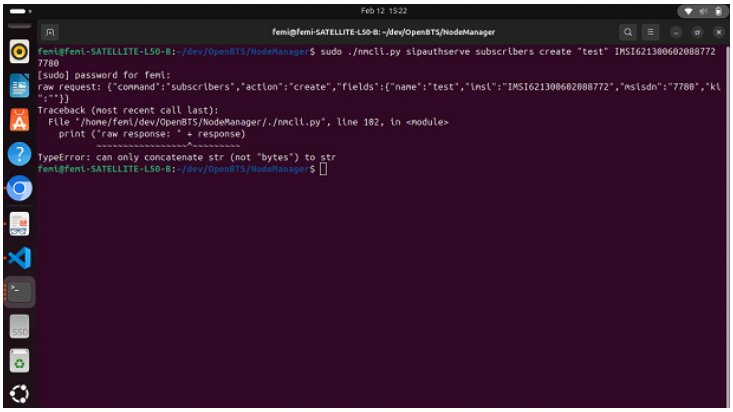


Figure 3.35: Assigning phone number

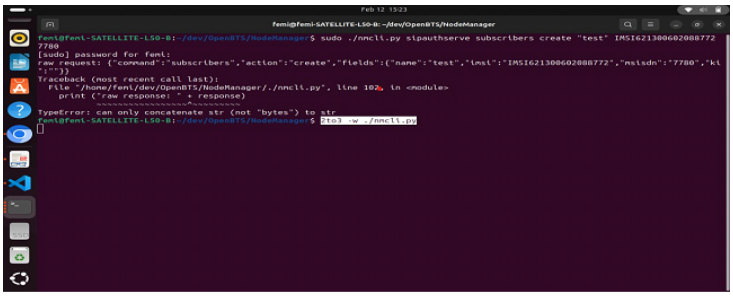


Figure 3.36: Assigning phone number b

If an error occurs, it may be due to OpenBTS being built for Python 2 while the operating system uses Python 3. To fix this, open the script with `nano`:

```
nano nmcli.py
```

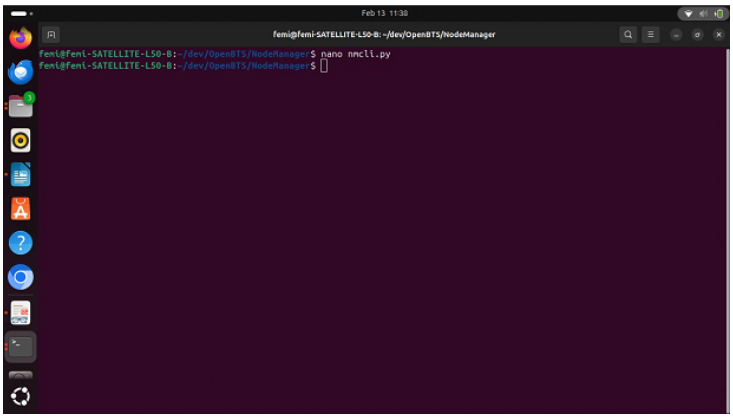


Figure 3.37: Fixing the python 2 error

Append the following lines to the end of the script to support UTF-8 characters:

```
response = socket.recv()
print("raw response: " + response.decode('utf-8'))
```

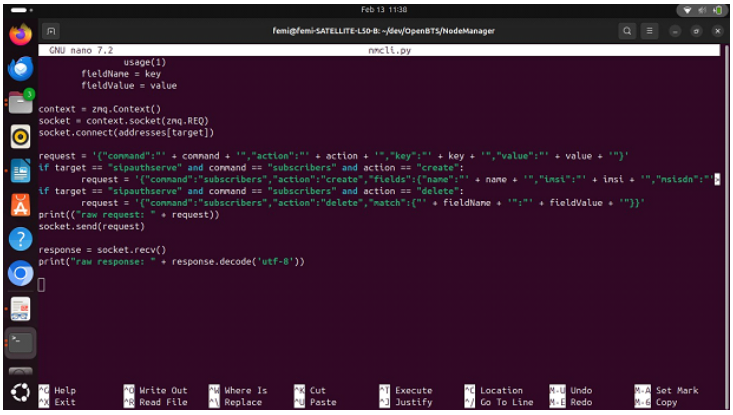


Figure 3.38: UFT-8 Character Support

After editing, you can authenticate devices using the `IMSI` from the `tmsis` output.

Re-run the subscriber creation command:

```
sudo ./nmcli.py sipauthserve subscribers create "test" \
    IMSI621300602088772 7780
```

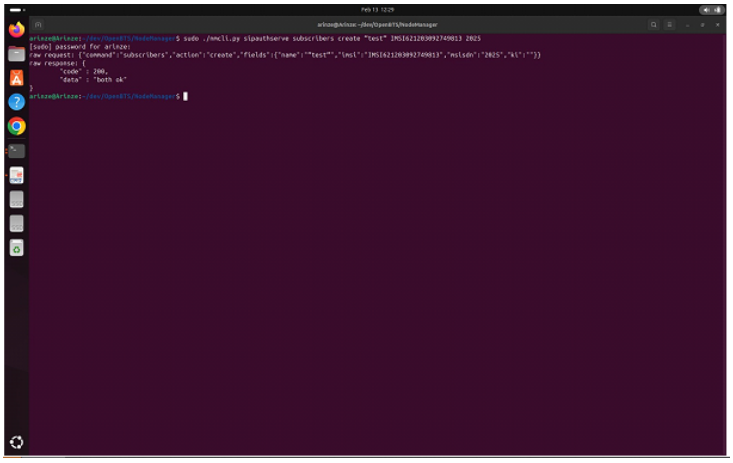


Figure 3.39: Subscriber Creation Command

Chapter 4

RESULTS AND ANALYSIS

4.1 Introduction

Synchronization is very important when it comes to two or more devices establishing communication. In GSM communication, synchronization is established by two bursts, that is Frequency Correction Channel (FCH) and Synchronization Channel (SCH). One of the important issues is the timing process of the hardware, as this will determine if the devices stay synchronized. The GSM system comprismes of clock frequency set either at 52 MHz or in the multiple of 13 MHz B200/B210, USRP series of Ettus research help to ensure synchronization is achieved as its clock frequency can be reconfigured by the UHD driver to meet the GSM specification.

After a successful setup of a test network, it's important to check the radio band and Absolute Radio Frequency Channel Number (ARFCN) being used. The radio band comprising of the: 850, 900, 1800, or 1900 MHz, of the frequency spectrum, corresponding to the four GSM bands used all over the world. Choosing the right radio band and ARFCH is significant as this will avoid interference with local carriers and to prevent any regulatory violation.

4.2 Results

The registration of a mobile phone on the test network was successful as shown in the figure with the connection to the test network, which was done by manually scanning for available GSM networks in the area, by changing the mobile network in the cellular setting of the mobile phone, from auto-

matic to manual and choosing the test network. Only SIM cards with their IMSI, that are already registered on the network will be authenticated on the network. The mobile phone then receives a custom welcome message from the network, that contains the welcome message and the IMSI of the mobile SIM. As shown in Figure 4.1.

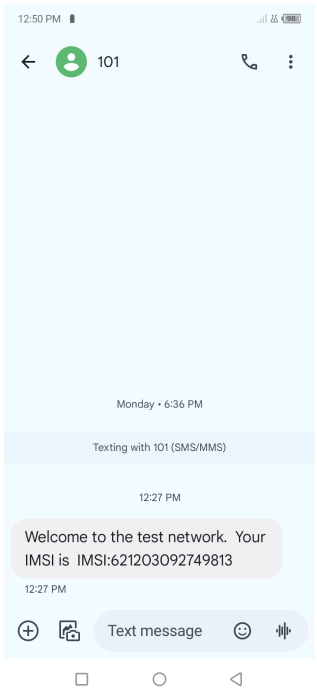


Figure 4.1: Welcome SMS

To test phone connectivity with the network, a test message was composed and sent to 411 from the registered phone on the test network. This code '411', is a short code handler in SMQueue and will just echo back the message along with some information about the subscriber. Which contains the number of queued messages for delivery, the load factor of the base station cell, the IMSI and MSISDN (phone number) of the sender, the time of the message and finally the message content, as shown in Figure 4.2.

When a message is sent to a subscriber that is not registered on the network, the network can detect such and send a reply to the sender, informing the sender that the destination mobile number is not yet registered with the network and as such the message is undeliverable, as can be seen in

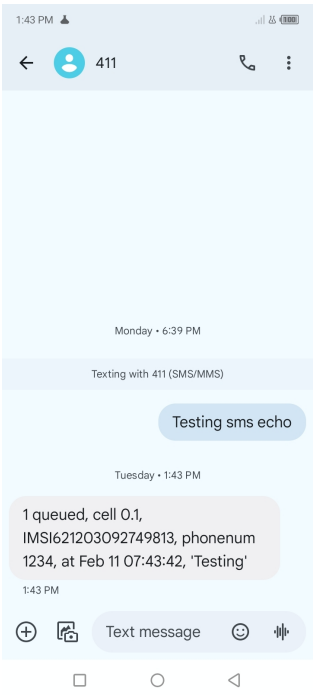


Figure 4.2: Test Echo

Figure 4.3.

After successful testing of the message handling capability between a registered mobile and the test network, the call handling section of the network, which is handled by Asterisk, was checked. A test call was first placed to 2602 (tone test), this just plays back a constant tone and is used to check if Asterisk is running, call routing and downlink audio are properly configured and functional. Figure **??**.

The second test call was to 2600 from the test mobile handset and the number '2600' is a short code handler of Asterisk that provides echo service just like '411' of the SMQueue.

Basically, all audio information that Asterisk receives will immediately be echoed back to the speaker of the mobile phone, from where the call originated. This test was carried out to ensure that Asterisk software, which is responsible for all voice applications of the network, is properly configured and to reveal any delay or uplink quality issues that may be present within the test network, to prevent call breaking. Figure **??** is an image of calling

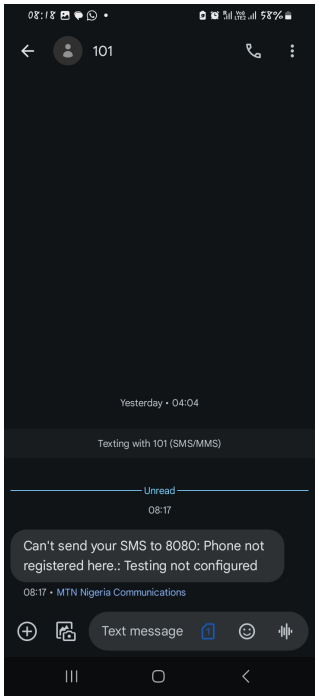


Figure 4.3: Test to Unauthenticated Numbers

'2600' to check the voice quality of the network at various distances from the SDR. This test, though it provides a means to check the voice quality, is subjective and not objective as it depends on the user. A better way to test the network quality rather than depend on human hearing was to check the signal-to-noise ratio which will provide better information about the attenuation the signal suffers. 2602 was dialed again and the 'chans' command was run from the command line with an interval of 30 seconds and with the mobile phone moved 10m away. The higher the SNR the better, as the SNR increases as the phone moves away from the SDR, this is since the mobile phone now transmits at a higher power to the base station, i.e., increased radiation from the mobile phone.

A SMS chat conversation was initiated among the registered phones, as shown. This was to further test how well the SMQueue handles multiple users. The SMS were received successfully by the mobile phones within minutes.

SMQueue, which handles the messaging of the network, queues the mes-

sage once the destination number has been authenticated to be part of the network. It then matches the number to the IMSI of the subscriber. The message is then delivered to the subscriber if the IMSI is active. If the IMSI is not currently active (the mobile phone is switched off), the message is rescheduled by the SMQueue scheduler pending when that IMSI is authenticated by SIPAuthServe and the message is sent.

The next test was to check the voice capability of the test network. However, given the changes to permissions and ownership over multiple releases of Ubuntu, **asterisk** has problems calling the **sip function** from any authenticated IMSI on Openbts through either **sipauthserve** and **node manager**. In later versions of asterisk, with 22 being the most up-to-date release, the **sip function** has been completely eradicated due to functionality issues. As a result of this, calls routing through Openbts are dropped instantaneously, even as changing owners, altering permissions and rewriting system files doesn't rectify this issue. There were similar results using **Yate, Free switch and Kamailo**. The clients were installed successfully, running and generating no errors even in debug mode, but failing to patch calls through with all calls dropped instantly. It is therefore apt to conclude that rebuilding Openbts using the latest version of Ubuntu 24.04lts is inefficient as the call functions are not initiated owing to their dependency on external clients built with the archaic and out-of-date OS versions not synchronize properly with the up-to-date and more secure operating systems.

4.3 Analysis

This test model was set up as a small-scale network. This model can be scaled up into a much bigger network, which can be used in the event of natural disasters, rural area/mineral exploration expeditions, where there is no GSM communication, and large institutions like universities can obtain radio band licenses and set up their own internal network. We look at the infrastructure (hardware and software) that will be required, its deployment, and the socio-economic benefits of this model.

4.3.1 Infrastructure

In field deployment, the USRP B210 cannot be used alone due to its limitation in terms of network coverage area as a result of its low transmitting

power at the RF front end. To solve this drawback, an RF power amplifier can be connected to the front end of the B210, making the signal high enough to provide a microcell, or another solution is to purchase an SDR with high transmit power. The OpenCellular is one of those solutions developed by Facebook, with the aim to increase connectivity in rural areas (Facebook 2016). The device supports 2G to 4G technologies. Also, Range Networks offers a variety of OpenBTS implementations, which range from 100mW to 50W of transmit power. It also runs the commercial version of the OpenBTS software. As compared to the Public, AGPLv3 version of OpenBTS, the commercial version offers enhanced security, stability, additional functional features, and is thoroughly tested to meet standard industry GSM specifications. Though it is costlier than the normal USRP software-defined radio, it offers enhanced performance (Range 2018).

Chapter 5

CONCLUSION AND RECOMMENDATIONS

5.1 Problems Encountered/Limitations

- Testing the OpenBTS cellular network was a bit difficult as we had to set up new packages and repositories to replace its obsolete and/or non-existent counterparts, which were incompatible with the software.
- Acquisition of the SDR used, USRP B210, to set up the network was a herculean task as it is not sold in the country. It had to be purchased from outside the country, and we waited for a long time to receive it.
- The PC needed to install the OpenBTS software had to match certain specifications to ensure the software runs efficiently. A PC with lesser specifications would run into issues running the software.
- Setting up OpenBTS software on Ubuntu 24.04 took a lot of time as we couldn't find a proper and complete guide online.

5.2 Recommendations/Solutions

- Software Defined Radio production and sales in the country should be considered as their usage in the industry is vast.

- The packages and repository codes required to run the OpenBTS version on Ubuntu 24.04 LTS should be written to reduce time delays and consequently, the fatigue of the programmer.
- The Nigerian Communication Commission (NCC) should provide means that will facilitate the academic society to venture into research in the radio frequency sphere by ensuring that spectrum bands are available for testing and research purposes.

5.3 Conclusion

The implementation of a GSM network using OpenBTS software running on Ubuntu 24.04 LTS is a successful project that demonstrates the feasibility of creating a low-cost, scalable, and secure GSM network using open-source technology. This project showcases the potential of OpenBTS and Ubuntu to provide a flexible and customizable platform for building GSM networks, particularly in rural or sparsely populated areas. With its scalability, flexibility, and low cost, this project has the potential to bridge the digital divide and provide essential communication services to communities in need. Overall, this project highlights the power of open-source technology in transforming the telecommunications industry and promoting innovation and collaboration.

Bibliography

[1] Reconfigurable cellular gsm network using usrp b200 and openbts for disaster-hit regions. 2017.

[2] Kinjal Aggrawal and Khyati Vachhani. Reconfigurable cellular gsm network using usrp b200 and openbts for disaster-hit regions. In *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, pages 141–146. IEEE, 2017.

[3] Arusha Dubey, Deepak Vohra, Khyati Vachhani, and Arvind Rao. Demonstration of vulnerabilities in gsm security with usrp b200 and open-source penetration tools. In *2016 22nd Asia-Pacific Conference on Communications (APCC)*, pages 496–501. IEEE, 2016.

[4] Emnify. 4g: The fourth generation of mobile communication technology, 2024.

[5] Ettus Research. About ettus research, 2018. Available at: urlhttps://www.ettus.com/site/about.

[6] Luis AG Gomez, Samuel C Pereira, André LLF Murari, Henrique S Franco, Jose AT Altuna, Mauricio BC Salles, Alfeu JS Filho, Carlos E Capovilla, and Ivan RS Casella. Analysis of a control system for dfig wind generators based on the transmission of power references through a gsm wireless network: a smart grid experimental approach. *Energies*, 12(2):241, 2019.

[7] GSMA. Number of mobile subscribers worldwide hits 5 billion, 2017. Press Release.

[8] Mobile Kishop. 2g mobile network: Everything you need to know, 2024. Accessed: 2025-02-19.

[9] P. Sharma. Evolution of mobile wireless communication networks: 1g to 5g and future prospects. *International Journal of Computer Science and Mobile Computing*, 2, n.d.

[10] Statista. Global mobile subscriptions since 1993, 2023. Accessed: 2024-02-18.

[11] A. Suliman Mohammed Ahmed, K. Bilal, and A. Mustafa. Comparison between cellular generations. *International Journal of Engineering, Applied and Management Sciences Paradigms*, 22, 2015.

[12] TechTarget. Definition (gsm), 2018. Available at: urlhttps://searchmobilecomputing.techtarget.com/definition/GSM.

[13] Telcom History Center. Telecom history timeline, 2024. Accessed: 2025-02-19.

[14] Telecom Trainer. 3g: Third generation of mobile communication technologies, 2024.

[15] Telephone World. Analog cellular (amps) - 1g. `https://telephoneworld.org/cellular-phone-history/analog-cellular-amps-1g/`, 2025. Accessed: 2025-02-19.