



Assignment of bachelor's thesis

Title:	4G Testing Network for IoT Device Penetration Testing Support
Student:	Jakub Tichý
Supervisor:	Ing. Jiří Dostál, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	until the end of summer semester 2023/2024

Instructions

Nowadays, many IoT (Internet of Things) devices use 4G technology for their network connectivity. Due to that, it is more difficult to analyze and test IoT devices from the cybersecurity point of view, since cooperation with the mobile network operator is needed. The solution would be a testing 4G network using a Software-Defined Radio (SDR) technology and SW simulation of essential components of a 4G network.

- 1) Get familiar with the srsRAN/srsLTE and Osmocom mobile network radio suites.
- 2) Explore the 4G and radio network technologies (SDR, SIM cards).
- 3) Design and implement an environment for creating a working 4G network built on the SDR technology. The CLI (Command Line Interface) will enable fast creation of a 4G network, but it will also enable to change its parameters and load/save them. It will also allow programming the custom SIM cards and their integration into the 4G network.
- 4) A user will interact with the environment using CLI.
- 5) Test the implemented environment by performing a security analysis of a connected 4G device.

Bachelor's thesis

**4G TESTING NETWORK
FOR IOT DEVICE
PENETRATION TESTING
SUPPORT**

Jakub Tichý

Faculty of Information Technology
Department of Computer Systems
Supervisor: Ing. Jiří Dostál, Ph.D.
May 11, 2023

Czech Technical University in Prague

Faculty of Information Technology

© 2023 Jakub Tichý. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis: Tichý jakub. *4G Testing Network for IoT Device Penetration Testing Support*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

Acknowledgments	vi
Declaration	vii
Abstract	viii
Abbreviations	ix
Introduction	1
Goals	3
1 Fundamentals of Mobile Networks	5
1.1 4G Network Architecture	5
1.2 EPC	6
1.3 E-UTRAN	7
1.4 User Equipment	8
2 State of Art	11
2.1 Osmocom	11
2.2 srsRAN	12
2.3 Hardware	13
3 Analysis and Design	17
3.1 User Interface	18
3.2 Functional Requirements	19
3.3 Nonfunctional Requirements	20
3.4 Domain Model	20
3.5 Programming Language	21
4 Implementation	25
4.1 Project Structure	25
4.2 Resources	28
4.3 Scripts	28
4.4 Source Codes	30
4.5 Documentation	34
4.6 Third Party Libraries	35
4.7 Dropping Privileges	35

5 Penetration Testing	39
5.1 PTES	39
5.2 Typical Test Case	40
5.3 Device Testing	41
6 Conclusion	45
A Installation Manual	47
A.1 Prerequisites	47
A.2 Installation Script	47
A.3 Optional Setup	48
B User Manual	49
B.1 Screen Layout	49
B.2 Navigation	49
B.3 Getting User Input	49
B.4 On-air Mode	50
B.5 Guided Mode	50
B.6 Manual Edit	51
B.7 SIM Profile Management	51
B.8 Project Management	51
B.9 Showing Current Parameters	52
B.10 Exiting the Application	52
C Domain Model	53
D Modules Dependency	55
Contents of Attached Media	61

List of Figures

1.1	4G Network Architecture	6
1.2	EPC Architecture	8
1.3	SIM Card in the SIM Tray	9
2.1	Sysmocom SIM Card	14
2.2	HID Smart Card Reader	15
2.3	RF-shielded Tent	16
4.1	Menu With More Than 8 Items	32
5.1	Testing Setup	42
5.2	Network Parameters	43
5.3	SIM Profile Parameters	43
5.4	Assigned IP Address to UE	44
5.5	Nmap Port Scan	44
C.1	Domain Model	53
D.1	Module Dependency	55

List of Code Listings

4.1	Requirements Line Format	26
4.2	Installing the Requirements	26
4.3	Running the Application	26
4.4	Network Record Format	28
4.5	Sphinx Installation	34
4.6	Commands Used for Generating the Documentation	34
A.1	Cloning Gitlab Repository	47
A.2	Running Installation Script	48

I would like to express my uttermost gratitude to my supervisor, Ing. Jiří Dostál, Ph.D., for his help and valuable advice while creating this thesis and for lending me the necessary resources and access to the testing environment. I would also like to thank my family and friends for their support and motivation. A special thanks to Eliška for all her advice, help, and mental support she has been to me. Last but not least, I would also like to thank Hayyan for his valuable help and the knowledge he has given me.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on May 11, 2023

Abstract

The bachelor's thesis focuses on the development of an application to simplify IoT testing processes. The thesis describes the architecture of fourth-generation mobile networks and analyses the srsRAN and Osmocom projects. The main part deals with the design and implementation of the application, which was developed using the Python programming language.

The application allows simple network parameter setting and SIM card programming. The network settings and SIM profiles are stored within the project which can be saved or loaded. The application executes the tools of the srsRAN and Osmocom projects using bash scripts.

The work also includes a security evaluation, as the application accepts user input, works with the file system and executes scripts using privileged commands. An installation manual and a user guide are provided for the users.

Keywords 4G, LTE, Internet of Things, Mobile network, srsRAN, Python, SIM card

Abstrakt

Tato bakalářská práce se zabývá vývojem aplikace sloužící k zjednodušení procesu IoT testování. Práce popisuje architekturu mobilních sítí čtvrté generace a analyzuje projekty srsRAN a Osmocom. Stěžejní část se zabývá designem a implementací aplikace, pro jejíž vývoj byl použit programovací jazyk Python.

Aplikace umožňuje jednoduché nastavení parametrů sítě a programování SIM karet. Nastavení sítě společně se SIM profily je uloženo v rámci projektu, které je možné uložit nebo načíst. Pomocí bash skriptů aplikace spouští nástroje projektů srsRAN a Osmocom.

Součástí práce je také hodnocení z hlediska bezpečnosti, protože aplikace přijímá uživatelské vstupy, pracuje se souborovým systémem a spouští skripty využívající privilegovaných příkazů. Funkčnost aplikace byla ověřena pomocí testu zařízení. Pro uživatele je doplněn instalační manuál a uživatelská příručka.

Klíčová slova 4G, LTE, Internet věcí, Mobilní síť, srsRAN, Osmocom, Python, SIM karta

Abbreviations

3GPP	3rd Generation Partnership Project
4G	Fourth-generation
APN	Access Point Name
BS	Base Station
ADC	analogue-to-digital converter
CLI	Command Line Interface
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CSV	Comma Separated Value
CTO	Czech Telecommunications Office
DAC	digital-to-analogue converter
EMP	Electromagnetic Pulse
eNB	Evolved Node B
EPC	Evolved Packet Core
EPS	Evolved Packet System
EUID	Effective User Identifier
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
FCC	Federal Communications Commission
FPGA	Field Programmable Gate Array
HSS	Home Subscriber Server
HTML	Hypertext Markup Language
ICCID	Integrated Circuit Card Identifier
IDE	Integrated Development Environment
IMSI	International Mobile Subscriber Identity
I/O	Input / Output
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution
MCC	Mobile Country Code
ME	Mobile Equipment
MME	Mobility Management Entity
MNC	Mobile Network Code
MS	Mobile Station
NCISB	National Cyber and Information Security Bureau
NIST	National Institute of Standards and Technology

OSINT	Open Source Intelligence
PCAP	Packet Capture
PCRF	Policy and Charging Rules Function
PGW	Packet Data Network Gatewa
PIN	Personal Identification Number
PTES	Penetration Testing Execution Standard
PUK	Personal Unblocking Key
QoS	Quality of Service
RAN	Radio Access Network
RF	Radio Frequency
SDR	Software Defined Radio
SGW	Serving gatewa
SIB	System Information Block
SIM	Subscriber Identity Module
SW	Software
TUI	Text-based User Interface
UE	User Equipment
UICC	Universal Integrated Circuit Card
USB	Universal Serial Bus

Introduction

Nowadays, the Internet of Things (IoT) phenomenon is experiencing a huge boom. Using various technologies and processes, we can integrate the distinct capabilities of individual sensors and active elements into a unified system that can simplify our daily tasks or perform them on our behalf.

When the term IoT device is mentioned, we probably think of something related to modern cars or smart homes. In these industries, there are many opportunities for an activity to be simplified or procured by smart devices. We can measure occupancy in buildings and even their specific floors. Another example can be a smart home that will turn on the heating and wash our clothes before we return from work in a car whose assistance systems ensure we will arrive safely.

One or more technologies can be used to communicate simultaneously. Bluetooth, Wi-Fi, and 4G/LTE are the best-known technologies. Along with them, there are many others. For each application, a suitable technology must be chosen for communication and control. Essential factors in such a choice are the environment, the number of communicating devices, the amount and size of messages, and many others.

With the increasing number of devices we surround ourselves with, the danger of cyber attacks increases. In the early days, there was not much emphasis on the security of IoT devices. As time passes, more and more attention must be paid to this issue. Smart devices are already found in places where abuse by an attacker could endanger human life. Manufacturers are subjecting their products to penetration tests to eliminate any vulnerabilities an attacker could exploit.

Comprehensive equipment testing is a time-consuming activity. It consists of many smaller tasks that must be performed in order. However, it is possible to automate some of the steps and make the tester's job easier by using automated activities or, for example, applications that take over some of the responsibility.

Goals

This work sets itself the task of creating a tool to facilitate IoT device penetration testing. The app will help penetration testers quickly and efficiently create a mobile network to test devices against known 4G vulnerabilities. Network settings will be able to be saved and loaded. The application will allow to change the network parameters, so the user will not have to set up the network from scratch every time.

The application will not be used for penetration testing or analysis of communication itself. It will be a tool to facilitate this task by setting parameters and adding users to this network. The network will be launched on 4G technology. Adding other technologies, whether older 2G, 3G or the new 5G, could be a possible extension of this work.

Fundamentals of Mobile Networks

The standardization and management of the mobile networks are handled by a consortium called the 3rd Generation Partnership Project (3GPP). Since 1998 3GPP has served to develop protocols in mobile telecommunications. It consists of seven organizational partners whose main task is to lead the project with appropriate policies and strategies [1].

The first generation of mobile networks (1G) was launched in 1979. Since then, almost every ten years, we have seen a new generation with numerous improvements and new services [2]. Since the thesis aims to analyze fourth-generation (4G) mobile suites and develop an application for the fast creation of a 4G test network, the architecture of other mobile generations is not discussed here.

In a mobile network, wireless communication establishes a connection between the end device and the network, enabling seamless communication without needing physical wires. The information carriers are radio waves in designated frequency bands.

The network is spread over a geographical area that is divided into smaller parts which are called cells. Due to the division into cells, we often see the term cellular network. Each cell contains at least one base station (BS) with a transmitter. These BSs are part of a radio access network (RAN), allowing end devices such as phones, modems, and IoT modules to connect and communicate with each other. The cellular architecture allows users to use network services while moving [3].

The RAN is a crucial part of the mobile network. It controls the radio wave connection between the end device and BS. Its main task is to carry the signals and data from the user to the core [4].

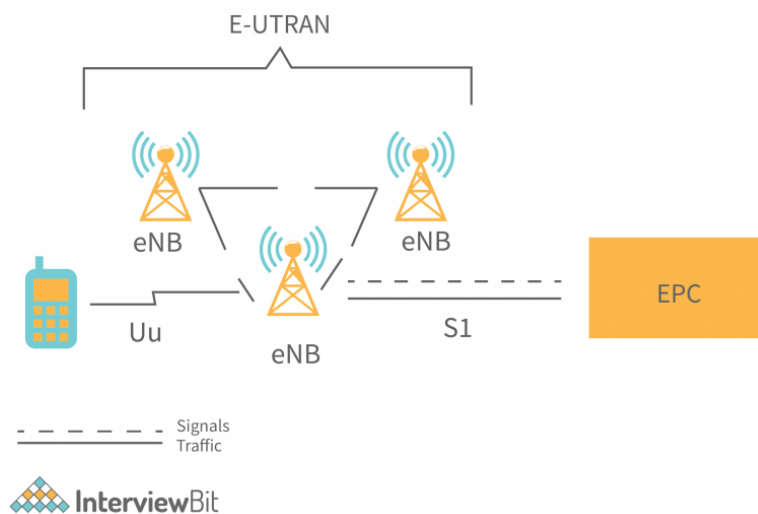
Signals are essential for managing and controlling communication between BS and the end device. The environment in radio networks is highly variable, so the connection quality must be constantly monitored. Connecting to the network and making a call or handover would not be possible without these messages [5].

1.1 4G Network Architecture

The network architecture of the 4G mobile networks is based primarily on two previous generations. It improves numerous processes and services while maintaining backward compatibility.

Devices that worked in previous generations may still be used. The evolution is particularly evident in high-speed data transmission and network capacity due to enhancing the radio interface and simplification of the core network [6, 7].

The network consists of a network core (EPC), a radio access network (RAN), which is provided by Evolved Universal Terrestrial Radio Access Network (E-UTRAN), and devices that connect to the network. A device that connects to the network must have a Subscriber Identity Module (SIM) card to authenticate itself. The internal network, called the Evolved Packet System (EPS), consists of EPC and RAN. The E-UTRAN consists of Evolved Node Bs (eNB) with transmitters that provide coverage and allow users to connect to the network. The different parts of the network communicate via interfaces (Uu, S1, and others) that are used for data transmission, signalling, or both simultaneously. A simplified diagram can be seen in figure 1.1 [7].



■ **Figure 1.1** 4G Network Architecture [6]

1.2 EPC

The network core is called Evolved Packet Core (EPC). EPC is an essential part of the 4G network. It manages network organization, switching, routing between networks, and peering security. It provides communication security and billing services for the users [7].

In 2G and 3G, there are circuit-switched and packet-switched data. The 4G uses only packet-switched data. It accesses data uniformly, using the Internet Protocol (IP). Voice data are converted from analogue to digital signal in the User Equipment (UE). Digital data are sent from UE after modulation via eNB to the core, EPC, where they are further processed and routed to be delivered to the end user. The unified approach has made it possible to simplify the core network architecture [8].

It consists of five components, Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving gateway (SGW), Packet Data Network Gateway (PGW), and Policy and Charging Rules Function (PCRF). Figure 1.2 depicts how individual parts are connected through signalling or data transmission interfaces.

1. MME

The role of MME is network monitoring, paging, organization and location area update. It also selects the appropriate SGW and manages the communication using signalling messages. When a session is created, the UE connects to one PGW, which is not changed later.

MME is responsible for attaching and detaching UEs. If the device is attached but inactive, it is switched to the idle state. UE management saves resources that can be allocated to active devices that want to communicate [9].

2. SGW

The serving gateway acts as a router between UE and PGW. It is responsible for charging roaming users according to their subscription details with the help of PCRF.

Administrative authorities may request the interception of the user's communications at SGW in special situations. For example, in the case of criminal activity, security forces can request a legal warrant and, based on the warrant, request the information of a specific user from the operator. Operators themselves have no legal authority to monitor their customers' data or calls [10].

3. PGW

PGW is a gateway between EPS and external networks. It can filter packets, charge the users according to the subscription and limit the downlink traffic. The uplink limitation occurs on the eNB to avoid unnecessary load on the network. UEs can connect to more than one PGW simultaneously to access multiple networks [9].

4. HSS

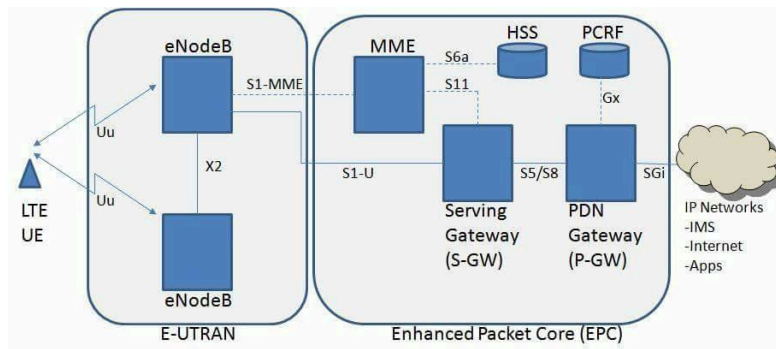
HSS is a database of subscriptions, authentication parameters and UE location used for user registration and identification.

5. PCRF

PCRF stores operator-defined rules, which define a way of allocating network resources to subscribers and applications. It is connected to PGW to manage traffic over the established rules [9].

1.3 E-UTRAN

E-UTRAN is a radio access network standard for 4G networks which consists of eNodeBs abbreviated eNB. There is no other intermediate element as it was in the previous generations [11]. The user utilizes it to establish a network connection with their User Equipment (UE) through a radio interface. eNodeB can communicate with UE, other eNodeB or the core network. Each case uses a different interface, as seen in 1.1. Communication between two eNodeBs is through the interface called X2 or the same as between eNodeB and EPC, which is called S1 [6].



■ **Figure 1.2** EPC Architecture [12]

The main functions of eNodeB are [7]:

- Radio resource management – allocation of resources for downlink and uplink, frequency selection and switching
- Activity management – managing resources for UEs in connected or idle mode
- Signalling and data flow – controlling UEs, encryption and decryption of packets
- Handovers – passing user context to other eNodeB as the user moves through the network

If the user wants to communicate with another user in the network, encrypted packets from their UE arrive at the eNodeB. The latter then compresses packet headers, sending the packets towards the S-GW. Data coming towards the UE are encrypted at the eNB, decompressed and then sent to the UE, which ensures the integrity and security of the user’s data when communicating on the network [13].

1.4 User Equipment

User Equipment (UE) refers to a user’s device to access a network. This can be a phone, a computer with a mobile broadband card, an IoT module, etc. Connection to the network is possible, provided the user has a SIM card and is within range of the radio transmitters of one of the eNBs.

1.4.1 Mobile Equipment

The hardware part of the UE with the SIM card slot is called mobile equipment (ME). The ME contains a radio interface that allows communication with the network. The radios in the ME are not designed to operate on a specific frequency but within several frequency bands. The ability to alternate between frequencies makes using the device in many places possible. Operators use different frequency bands in different countries. The type of radio that can be software tuned to various frequencies is called a Software Defined Radio (SDR) [14].

1.4.2 SIM

The second essential part is the Universal Integrated Circuit Card (UICC) placed in the ME. UICC is a smart card that is used in mobile telecommunications. According to the National Institute of Standards and Technology (NIST), UICC can also be called SIM, USIM, RUIM or CSIM [15]. This thesis uses the term SIM instead of UICC, as it is the most common name.

For the second generation of mobile networks, a UICC contains an application called SIM. Since the third generation of networks, the Universal Subscriber Identity Module (USIM) applications have been used instead. The application structure and files on the card have changed, but it still allows backward compatibility [16].

SIM is a hardware element with software to store information that uniquely identifies the user securely. The SIM contains an Integrated Circuit Card Identifier (ICCID) – a unique serial number, an International Mobile Subscriber Identity (IMSI), a key for encrypted communication, a quality of service identifier, and more. Optionally, phone contacts and other user data can be stored on the SIM [16].

Access to the SIM is secured with a PIN. The user is prompted to enter the PIN when the SIM is inserted into the ME. After three unsuccessful attempts, the card is locked, and a PUK is required to unlock it. In addition to the user-accessible information, other information is stored here, but only the operator or manufacturer can access it.



■ **Figure 1.3** SIM Card in the SIM Tray [17]

State of Art

This chapter focuses on the Osmocom and srsRAN projects. It covers the software needed to virtualize the essential parts of the 4G network, discusses the possibilities of programmable SIM cards, and describes the hardware needed to run the network itself.

2.1 Osmocom

Osmocom is a collection of various software tools in the mobile communications field developed as open-source. The source codes are publicly available on git, allowing anyone to contribute further and develop. The applications are mainly based on existing mobile network standards. There is also additional software for testing network functionality or managing peripherals [18].

Osmocom's software tools are being continuously developed. New features are added, and existing applications are updated. Thanks to the open source, it is easy to keep track of the latest commits, changes made, bugs, and much more. To enable the testing of devices in the virtual network, it is necessary to program SIM cards, which can be done using two tools provided by Osmocom.

2.1.1 SIM Programming

Besides the software to virtualize the different parts of the network, it is also essential to have a tool that allows SIM programming. To connect to the network, the device must contain a SIM card. The device has to identify itself within the network using the identifiers stored on the card. SIM cards' history, evolution, and structure have been discussed in section 1.4.2.

The average user has encountered and used a SIM card from a commercial mobile provider – an operator. These commercial SIM cards cannot be used for device testing due to the encryption of the communication with a secure key unavailable to the user. As mentioned earlier in 1.4.2, this key is only available to the operator or card manufacturer. Testing devices would be useless if using commercial SIM cards without knowing the secure key. Interception of communications is possible, but the captured data would be unreadable due to the encryption.

Programmable SIM cards are used for testing mobile devices. With the help of suitable software, users can change the parameters assigned to the SIM card, even those ordinarily immutable. In the context of this thesis, essential parameters are: *IMSI*, *authentication protocol*,

operator code, and key used to encrypt the data. Knowing these parameters makes it possible to decrypt and analyze the communication further.

Osmocom has two projects dedicated to SIM card programming. The first tool is called `sysmo-usim-tool` [19], explicitly designed to work with `sysmoISIM-SJA2` and `sysmoUSIM-SJS1` SIM cards. The project includes two command line interface (CLI) tools for programming SIM cards. They have the same names as the programmable SIM cards. These scripts, written in Python, can manage and configure the individual fields of the SIM cards. Because the scripts are developed for a specific type of card, working with other cards is possible, but some options may be inaccessible or end up with an error message. It should be remembered that these are not universal tools.

The second tool is called `pySim` [20]. Its Osmocom wiki page provides many tutorials for installing software or programming SIM cards. It comes as a package of three CLI tools [21]:

- `pySim-read` – reading SIM parameters
- `pySim-prog` – programming SIM, setting parameters values
- `pySim-shell` – management of file structure and data of SIM

2.2 srsRAN

Software Radio Systems Limited (SRS) is an Irish company that has been operating on the market since 2014. Since then, their outreach has spread globally. The project is funded by providing its enterprise versions of products alongside European and US partners.

It provides solutions for wireless mobile network customers, developers, and researchers. The `srsRAN` project is open-source, providing solutions for 4G and 5G mobile networks. Both mobile suites provide complete RAN solutions with all software needed to represent the individual components of a network [22, 23].

For 4G, `srsRAN` offers a software solution for all network parts, including the UE. Three tools, `srsEPC`, `srsENB`, and `srsUE`, make it possible to create a virtual network. After installing the `srsRAN` package, all tools are available to the user. `srsUE` is used to virtualize the user's device. Since the thesis focuses on developing an application that supports penetration testing of real devices, `srsUE` is not used.

2.2.1 srsEPC

`srsEPC` provides simplified EPC functionality with all its essential parts. The `srsEPC` will create virtual HSS, MME, S-GW, and P-GW instances. Once they are started, it provides communication between the different parts of the core network [24].

The setup of the parts mentioned above is defined using a configuration file. The configuration file path is given as an argument to the `srsEPC` program. If not specified, the application tries to open the default configuration file. If it fails to find even this file, the application will terminate.

For HSS, a database of users is needed. A particular file in comma-separated value (CSV) format represents the database. One row represents one SIM profile and has the format: “*user name, authentication algorithm, IMSI, security key, operator code type, operator code, authentication management field, sequence number, QoS class identifier, ip address*”. When a user

attempts to connect to the network, the EPC looks to see if the user, uniquely identified by his IMSI, is in this file. If so, they are connected to the network. Otherwise, they are denied access.

Users who are connected to the network can be granted access to the Internet. Before running srsEPC, packets must be forwarded from the EPC virtual network interface through the device interface to the Internet. The project includes a script that takes care of the network interface masquerade.

2.2.2 srsENB

A program called srsENB provides the functionality of a single eNB. In other words, it forms a RAN with a software-defined eNB cell. After establishing communication with the EPC using IP, the EPS of the network is created.

Every cell needs a transmitter to operate correctly. Software Defined Radio (SDR) represents the transmitter function. To establish communication with the network, it is necessary to connect to SDR, which enables srsENB to offer a radio interface for UEs.

srsENB offers many SDR variants that it can work with. The radio must have the correct driver installed and be fully functional. When the device that serves as the radio interface is not found, the srsENB does not continue and terminates.

Similar to srsEPC 2.2.1, a configuration file is used to set the parameters of eNB and transmitter. The file is loaded at the application start-up. If not specified, the application tries to open the default configuration file, and if even the fallback configuration file is not found, the application terminates.

In addition to the high-level configuration file, srsENB has three other configuration files that extend the ability to configure individual elements of the network and their behaviour. The radio resources are set in *rr.conf*, the System Information Block (SIB) is set in *sib.conf*, and the radio bearers are set in *rb.conf*. These configuration files have a different structure to *enb.conf*. The file structure follows the format of the libconfig library, which provides the ability to read, write and modify files of this type [25].

2.3 Hardware

Special hardware and software are necessary to get the test network up and running. This chapter discusses the hardware required for SIM card programming, SDR, and the secure environment in which the network can safely operate without interfering.

2.3.1 Programmable SIM Cards

A SIM card is a smart card containing a file system. Information is stored in files, which can be grouped into folders. Programmable SIM cards are different from commercial ones in that they allow the user to use software to change the contents of individual files, i.e., change the SIM parameters.

Communication with the card is done using APDU commands. Within the SIM standards, the commands are defined in ETSI TS 102 221. The document also addresses the permissions needed for individual commands and data access [26].

The Osmocom offers such SIM cards. They comply with 2G/3G/4G standards and allow use in 5G networks. The card carries the same file system as specified by 3GPP in Release 16 [27].

They send information about the initial setup, security PINs, and administration keys of the cards. The administration key enables reading and programming typically inaccessible files in the card's file system.



■ **Figure 2.1** Sysmocom SIM Card [27]

2.3.2 Card Readers

To program SIM cards, a device that can both read and write the data to the SIM chip is needed. For this purpose, card readers are used – these peripherals connect via USB or Bluetooth or can be built into the device.

Card readers communicate with the card through its contacts, whose location and function are standardized in ISO/IEC 7816-2. The reader supplies power to the card via power pins and communicates with the chip via data pins.

Some fields and their values are public, but some are protected. A code called an administrative key must be entered to authenticate the user and give them access to read or change the values.

This key is known only to the operator or the card manufacturer in the commercial sector, which protects against unauthorized modification of the data. The administrative key, like the PIN code, can only be entered twice incorrectly. After the third mistake, the SIM card is blocked.

2.3.3 SDR

Software defined radio (SDR) is a unique radio communication system. Its parameters can be dynamically adjusted as needed – the hardware remains the same, and only the software settings are altered. Unlike conventional radio systems, it is not designed to function only on a specific frequency. This approach makes it possible to use one SDR system for multiple applications [29].



■ **Figure 2.2** HID Smart Card Reader [28]

SDR system is divided into two parts.

- The Analogue front end receives and transmits signals using radio waves through the antennas. Its function is to cater to the part of the radio that cannot be digitized. It contains signal amplifiers, frequency filters, and other electronic radio elements.
- The digital back end consists of the necessary hardware to process the analogue signal from the front end and convert it into digital form. Modern SDR systems use Field Programmable Gate Arrays (FPGA), dedicated processors, analogue-to-digital (ADC), and digital-to-analogue (DAC) converters to process digital signals. The function that the hardware performs is defined by the software [30].

2.3.4 Safe Environment

When working over a specific frequency spectrum, for example, when setting up mobile networks, it is essential to avoid interference. Interference with another frequency band, even unintentional, can be seen as an attack to deny user access and subsequently be sanctioned. The frequency bands may differ in each country, and so may the interference penalties.

The allocation and regulation of frequency bands in America are under the purview of the Federal Communications Commission (FCC) [31]. In Europe, member states determine their policy on radio frequencies according to international agreements and within the scope of European Union legislation [32]. Specifically for the Czech Republic, the governing authority is the Czech Telecommunications Office (CTO). All such authorities aim to manage the spectrum and respond to changes brought about by new technologies.

When working with radio resources, it is advisable to use a secure environment where the signal radiation is enclosed and not penetrable to the outside, i.e., it will not interfere with other signals outside the environment.

The best option is an anechoic chamber that absorbs all the sound and electromagnetic waves. Its design uses the principle of a Faraday cage. The enclosure of the room is covered

with a conductive material that will not allow electromagnetic waves to pass in or out. Sound elimination is guaranteed by covering the inside of the walls with spikes or wedges that dampen the waves. This feature is not essential for this thesis [33].



■ **Figure 2.3** RF-shielded Tent [34]

However, the anechoic chamber is expensive, and the test environment must often be portable. For this purpose, RF-shielded tents are available. Like the chambers, they use the Faraday cage principle. The material from which the tent walls are made is highly conductive and will not allow electromagnetic waves to pass through while isolating the environment. The advantage is that the environment is often cheaper, it is portable, its size and shape can be easily adjusted, and if it is not needed, it can be packed away.

Analysis and Design

This chapter discusses the scope and characteristics of the application designed to facilitate penetration testing of 4G IoT devices. The requirements placed on the application are divided into functional and non-functional requirements. Functional requirements are those that relate to the functionality of the application and describe its behaviour. Non-functional requirements complement the functional requirements and focus on the system, environment, user experience, services, and more.

There are many steps involved in testing a 4G IoT device. Apart from the testing itself, the tester must connect the necessary hardware, create a test network, program the SIM card with a network known SIM profile, and then start the network. Once they have completed these steps, they can connect the device to the network and start testing it.

The main objective of this application is to streamline the testing process and make it more user-friendly. Users who lack expertise in the complexities of 4G networks can easily test the devices by simplifying the setup process. Even users familiar with the application but unable to set up the testing environment can perform testing tasks, thanks to the ability to use default settings and adjust only necessary network parameters.

The IoT trend has experienced tremendous growth in various technology sectors in recent years. Manufacturers worldwide are incorporating this technology into their products for home appliances, cars, home assistants, various modules, and many more. Individual devices surround us daily and serve our benefit. However, precautions must be taken to ensure that they are sufficiently secured.

In the early days of IoT, security should have been paid more attention. Nowadays, attackers try to hack IoT devices and use them to cause damage or get valuable information. These devices reach places where security should be a top priority as time passes. In addition to our homes, we can also find them in agriculture, transportation, healthcare and cars.

Neglecting safety as the top priority can result in life-threatening situations. In 2015 an IBM team demonstrated a hacking incident which exploited a vulnerability in the software of a Jeep car's onboard system, enabling them to manipulate the vehicle's acceleration, braking, and steering [35].

Manufacturers and developers have realized that more effort needs to be put into security. Related standards are still evolving and aim to eliminate possible weaknesses.

Security takes place at several levels.

- Hardware security includes measures to protect against various forms of interference, including peripheral connection, intrusion and electromagnetic pulse (EMP).
- Software security looks at the quality of the application, deciding whether the user has sufficient permissions and protection against malicious code execution.
- Network security involves securing communications through encryption, using sufficiently strong and unbroken cyphers, distributing keys, etc.

A myriad of possible technologies is used for communication between IoT devices. Among the most well-known are WiFi, Bluetooth, cellular and ZigBee. The thesis focuses, in particular, on devices that use 4G connectivity for their communication. Such devices can be found, for example, in cars, agriculture, or even smart electricity meters.

Nowadays, an attacker can also manufacture a device and deliberately create a feature that can be exploited to obtain data. Following on the smart electricity meters, in May 2022, the Czech National Cyber and Information Security Bureau (NCISB) warned about smart electricity meters from countries with untrustworthy legal environments [36].

The warning specifies a total of three threats:

- Tampering with measurements data
- Disconnection of the consumer point from the energy delivery
- Destabilisation of the distribution network due to the disconnection of a large number of points of consumption

According to the NCISB warning, the threats mentioned are classified as probable to very probable. A potential attack would impact an individual, a company, and the entire state. If the distribution network is destabilised, the reach could extend beyond the state.

3.1 User Interface

Part of the assignment is that the application should have a command line interface. The original idea was to create a script that would accept parameters via switches and arguments and perform all the necessary operations on the project and the network setting. After a discussion with the supervisor, we agreed that this interface would be inappropriate.

The application is designed for users unfamiliar with mobile networks or the srsRAN project applications. However, its command-line interface may not be the most user-friendly, and understanding it requires studying the tool's possible arguments and switches that control its operation.

Ultimately, we decided to use a Text-based User Interface (TUI). TUI is an interface that looks like Graphical User Interface (GUI), but the screens consist only of text displayed in the terminal window by outputting formatted strings. Examples of applications using TUI are Midnight Commander and Vim.

3.2 Functional Requirements

Functional requirements define the objectives the application must meet for its designated purpose. A total of eight functional requirements is defined within the application [37].

User Interface The user interacts with the application using the Text-based User Interface (TUI). The application is launched within the terminal, and its screen displays individual screens to the user and gets user input.

Guided Mode Network setup is possible using the guided mode, where the user is led through the network setup process. Individual parameters have a default value, or they have information about what they mean and what the expected values are. In addition to setting up the network, the user is asked if he wants to create a SIM profile or load SIM profiles from a file.

Manual Edit If the user wants to change any parameters and does not want to use guided mode, they can use edit mode to view the current configuration settings and change individual parameters. Selecting any of the parameters displays the appropriate prompt screen. After validating user input, the settings are updated. When the user picks the wrong parameter, they can return to the selection menu without changing its value.

Configuration Files The individual parameters are saved in the temporary configuration settings after validation. If an invalid value has been entered, the user receives an error message and can enter a new value. The temporary configuration in the application is used to create the necessary setup files for the srsENB and srsEPC applications. A user database *user_db.csv* is created based on the SIM profiles, which are stored in the current project. This file is required for the srsEPC to work correctly and let the network connect the selected users.

Project Directory The configuration files, user database, and resource files are the directory's contents that represent the project. Each project has its unique name and uniformly defined structure. Projects are used for storing the settings of a single network.

Load and Save Saving operating network configuration is possible in two ways. The configuration can be saved within the current project or a separate one without modifying the existing one. Projects created within the application can also be loaded. The configuration is loaded from the files that are located within the project.

SIM Profiles Management The application allows creating, reading, updating, and deleting the SIM profiles in the user database – CSV file. In addition to CRUD operations, it allows the programming of SIM cards according to the parameters of the selected SIM profile. After inserting the programmable SIM card user can proceed to the programming. The application uses a script to perform the necessary tasks. Finally, it provides feedback on the success or failure of the programming process.

On-Air Mode After setting the parameters, it is possible to switch to on-air mode. The application generates the necessary configuration files. After that, it launches the srsEPC and srsENB applications and two terminal windows to display logs from these applications in live form. Once the device is tested, the application allows an easy transition back to the off-air mode by disabling all running applications and the radio.

3.3 Nonfunctional Requirements

Non-functional requirements, as opposed to functional requirements, do not restrict the software's functionality but focus on its behaviour and performance. Five non-functional requirements were defined within the application to specify the behaviour of the application and the principles used.

Logs and Pcap Files The application collects logs and Packet Capture (pcap) files for data analysis. The log files contain information about UEs connection, network elements' behaviour, and more. The pcap files are used for later analysis of the communication between the UE and the network. Files with pcap extension contain network packet data. Pcap files can be opened with suitable software, such as Wireshark.

Projects Portability Projects stored as folders can be easily moved between computers using diverse media types. This approach allows one user to create a project specifying a network setup and another user to load the project and launch the network.

Security The application works with files, accepts user input and runs scripts for individual tasks. From the security viewpoint, it aims to follow the least privilege principle as much as possible. The definition is as follows: *The principle that a security architecture is designed so that each entity is granted the minimum system authorizations and resources that the entity needs to perform its function.* [38, p. 56]

Target Platform srsRAN can be installed on the following platforms: Ubuntu, Debian, Arch Linux, and openSUSE [39]. The application development is conducted on the Ubuntu platform, where it provides full functionality. The other platforms are not tested, meaning that the application might not work on unspecified platforms correctly due to differences in operating systems. Future extensions to other operating systems are possible.

Installing Dependencies The proper installation of srsRAN and its accompanying programs and drivers for radio is crucial for the application's functionality. The application cannot create or run the network if any program, part of a program, or essential component is missing.

3.4 Domain Model

The domain model serves as the foundation for the detailed design of the application architecture. It provides a high-level view of how the application is structured. The individual entities are used to describe the objects and their attributes. Thanks to the interconnections, it is also possible to obtain information about the relationships between the objects and their quantity from the diagram.

Appendix C.1 shows the application's domain model design. It consists of a core called *APP*. Other objects are successively bound to the core to provide sub-functions. To display the item selected by the user from a *Menu*, the application requires a *Screen* capable of displaying it. The *Prompt* receives input from the user, performing initial data validation.

The primary data object is the *Project*, which stores the network *Configuration* and *SIM profiles*. The status of whether it is saved or not is maintained within the *State*. The *Project* has a name and a location where the required files are saved.

3.5 Programming Language

There are many suitable languages for TUI application development, each with pros and cons. Several parameters have a significant role in the choice, such as Ubuntu platform support, a library for terminal control, user base and experience with the programming language. Based on the three parameters, the choice was narrowed to four languages: C, C++, Ruby, and Python. A discussion of the selection parameters and pros and cons of selected programming languages follows.

Ubuntu platform support Ubuntu and other operating systems, where srsRAN can be installed, support all the programming languages mentioned above, which can be a significant advantage if the application expands to other platforms. Except for Ruby, all programming languages are installed in Ubuntu by default.

Library for terminal control The second point is essential for application development. Developing functionalities that manipulate the terminal screen would be very time-consuming. Therefore, all four languages are well-positioned to develop a TUI application. A library in C/C++ called *curses* provides all the necessary functions. Python has a library named the same as C/C++, which is just a wrapper for the C/C++ variant of the library. The only difference is that the Python wrapper simplifies some functions [40]. Ruby has a similarly named library called *rburses*, which provides a lot of pre-made widgets and functionalities.

User base If an issue arises during programming, it is highly likely that someone has encountered and resolved it before. Even if there is no exact solution to a specific problem, seeking advice from other programmers who may have encountered similar issues is feasible. However, getting stuck on an issue that cannot be easily resolved due to a lack of available resources is possible for less popular programming languages. Nonetheless, all of these programming languages satisfy this requirement.

Based on the points discussed, Python is the most suitable programming language. The main reasons are the features the language offers and its high-level view. Work with many objects and strings is expected when developing the application. Python offers many functions and libraries that can be used to simplify coding. Due to the large user base and the constant evolution of the language, it is the right choice.

3.5.1 Python Coding Conventions

Guido van Rossum, the author of Python, defined a style guide for writing clear code. He later created the PEP 8 style guide with two other authors, Barry Warsaw and Nick Coghlan [41]. Python differs from other programming languages in that it does not use curly braces to delimit a block of code but a line indentation, typically only used to make code easier to read. This approach might be confusing for novice programmers. The programmer can avoid mistakes such as lousy indentation using the guidelines specified in [41].

Modern Integrated Development Environments (IDEs) have implemented automatic formatting, suggestions, and other principle-based features, making it easier to stick to proper conventions and write readable code. If a user is using an IDE that does not have this capability, they can use tools that take a source code file as input and modifies it according to the guidelines. An example of such a tool is pep8 [42].

3.5.2 Modules

When creating larger projects, it is good practice to divide the project into smaller units that are related. Smaller parts are easier to understand, and their management can be divided among several people or departments. This division can be seen in two levels.

Files The project directory has other sub-directories whose name indicates what it contains – for example, a directory for source code, documentation, logs, etc.

Source code The individual functionalities are divided into libraries or, in the case of Python, into modules. The goal is to group logically related parts of code and make the code easier to understand.

A module is a file ending with a `.py` extension containing constants, definitions and source code. Modules might be executable by themselves, but the primary use case is importing them into another source code file. Once the module is imported, its objects, functions, and definitions become available.

Python has many already built-in modules that are part of the standard library. The functionality of the modules is available after importing them into a source code. Modules are usually written in Python. However, some are written in C, especially modules that interact with the system, system services, I/O operations, etc. [43, 44].

In addition to the built-in modules, the programmer can create their own modules. Similar to writing code, following the guidelines for creating is advisable.

3.5.3 Documentation

Documenting the code is essential for maintaining and developing the application. As projects evolve, source code becomes less obvious, especially when development is done in teams and one relies on the other. Documentation is used to quickly determine what a particular piece of code does and how it should be used.

Even this application, a relatively small project, must be appropriately documented. Especially if someone wants to continue to develop the application in the future, it is essential to study the documentation and recognize the code structure before programming. Understanding the code makes it easier for the programmer and saves time.

3.5.3.1 Docstrings

Docstrings are special comments that are used for documentation in Python code. They are placed at the beginning of a function, method, class, or module. The docstring is enclosed within triple quotes and serves as a special attribute `__doc__` of the object it belongs to. It can be accessed within the code and is often used for generating documentation for the code. All the guidelines for writing docstrings and their functions can be found in PEP 257 [45].

3.5.3.2 Documentation Generation

Many tools can generate documentation as HTML pages for Python source codes. One of the most widely used is Sphinx. It works with files in reStructuredText or Markdown format [46].

It uses the *conf.py* configuration script to determine the documentation's appearance. The configuration file can be used to set resources, page structure, documentation theme, etc.

The build command starts a process that creates a folder with all the necessary documentation files based on the configuration file and resources. The result is interactive, clickable code documentation based on custom docstrings.

Sphinx has many custom and third-party extensions. To use them, they need to be imported using the *conf.py*. Extensions add functionality over and above the standard functionality allowed by Sphinx.

Implementation

This chapter deals with the implementation of the application. It describes the file hierarchy structure, contents of sub-directories, and individual files. The chapter also includes the procedure for generating documentation for the application and describes used third-party libraries. Finally, a chapter is devoted to security and defining the necessary permissions.

4.1 Project Structure

The application is contained within a directory that contains individual files and sub-directories, with logical divisions that separate it into smaller parts. Organizing the files into directories according to their function within the project allows for better understanding and orientation. This section focuses on the files and their properties in the main application directory. The sub-directories and their contents will be described later.

4.1.1 README.md

README.md is a file containing basic information about the application, instructions for the installation, and other necessary links that may prove helpful. The file is written in Markdown language.

4.1.2 default.conf

default.conf contains the project's default configuration, where are specified the essential variables. It has a similar structure to, for example, the configuration files for srsEPC and srsENB. The section name is enclosed in square brackets, the variables have a name, and its value is assigned after the equal sign. All variables fall under the closest section above them. Lines starting with a semicolon serve as comments.

default.conf consists of 13 sections. The application attempts to retrieve the values when creating new objects that use values stored in the default configuration file. Obtained values are read and applied. The object uses the fallback values if the file does not exist or an error occurs.

4.1.3 requirements.txt

Not built-in libraries have to be installed for the application to work correctly. *requirements.txt* defines libraries and their version that must be downloaded. The file is composed of lines, which have a structure in a form similar to what is seen in 4.1.

■ **Code listing 4.1** Requirements Line Format

```
library==version
```

The application needs two additional libraries, which must be installed, ConfigUpdater version 3.1.1 and netifaces version 0.11.0. The version of the libraries has to be the same or higher. The installation script automatically installs the libraries. For manual installation of the libraries from the requirements file, the following command is used 4.2.

■ **Code listing 4.2** Installing the Requirements

```
pip3 install -r requirements.txt
```

4.1.4 F4GD.sh

F4GD.sh stands for Fast 4G Deployment, and it is a shell script that prepares the environment and runs the application. It performs several checks before the actual launch. Both the application and this script are designed to run with the privileges of a non-root user. The application also assumes the user has a home directory in `/home/[username]`.

The script verifies the Effective User ID (EUID) to determine if the application is executed with the `sudo` command. If the EUID is zero, indicating root privileges, the script terminates without further execution.

If a regular user runs the script, it appropriately sets the terminal window size. The terminal must be a minimum size of 100x30 to display screens without encountering any text overlapping. The application's name is used as the terminal title for better clarity.

In order to execute the *F4GD.sh* from any location within the file system, the current working directory within the script is specifically configured to the *src* directory, where the application's main entry point resides.

Setting the working directory is done in three steps. The first step sets the working directory to the main application directory. If it succeeds, the second step is to move into the *src* directory. Lastly, the script checks if the *src* directory contains a file called *main.py*.

If all the steps are performed without error, the script runs the application using command (4.3).

■ **Code listing 4.3** Running the Application

```
python3 main.py
```

4.1.5 install.sh

There are two phases in the installation process. In the first phase, the installation script checks that all the prerequisites for a successful installation are met. The second phase does

the installation itself. The script must be run using the `sudo` command because it uses privileged commands.

The prerequisites for successful installation are Python version 3.10 or higher and `pip3` for installing required Python packages. The `srsRAN` project applications must also be installed. If any requirements are unmet, the installation script will abort the process and inform the user of the error. The only optional requirement is to have installed the Osmocom package called `sysmo-usim-tool` for programming SIM cards. This package does not have to be present during the installation, but without it, the user cannot use the SIM programming feature.

Once all the requirements are verified, the initial step is to install the required Python libraries. The installation procedure is listed on the screen. If the packages are installed, `pip3` will inform the user that the requirements have already been satisfied.

The next step is to create a folder to store the user's projects. Since the script uses the `sudo` command, the owner of the created directory is the root. Once the project folder is created, it is necessary to transfer ownership of the directory to the user who used the `sudo` command. The username of the user who used the `sudo` command is stored in the bash variable called `SUDO_USER`.

The first privileged operations are assigning permissions and changing file owners. A group called `F4GD_users` is created for the application users. The user installing the application is assigned to the group right after its creation. They are also assigned to `netdev` group to be able to manage the `srsEPC`'s virtual network interface. To prevent the contents of the application scripts from being changed, ownership is transferred to the root, and the group owner is `F4GD_users`. Mask 750 ensures that the root can freely manipulate the files, users in the `F4GD_users` group can read and execute the scripts, and others have no permissions.

It is necessary to set capabilities for two scripts. The `srsEPC` binary is given the capability called `cap_net_admin`, allowing administration, configuration, and management of the network interfaces. Thanks to this capability, `srsEPC` can create and manage virtual network interfaces. The application script `off_air.sh` needs the `cap_kill` capability to kill `srsEPC`, `srsENB`, and opened terminals displaying application logs. Assigning the capabilities allows the scripts to perform these tasks without using the `sudo` command.

A rule must be created in `/etc/udev/rules.d` to access the virtual interfaces created by the `srsEPC`. Inside this directory, it is necessary to create a file called `zzz_net_tun.rules` containing the line: `KERNEL=="tun", OPTIONS+="static_node=net/tun", GROUP="netdev", MODE="0660"`. The provided line establishes a rule that designates the device identified by the kernel name `tun` to be owned by the `netdev` group, granting permissions of 660. Additionally, it creates a fixed path to the device node at `/dev/net/tun`.

The final step of the installation is to set resource limits. Creation of threads with real-time priority is required by both `srsENB` and `srsEPC`. The user has to be granted sufficient system resources to avoid using the `sudo` command to run the applications. The script sets the `F4GD_users` group in `/etc/security/limits.conf` priority and `rtprio` to 95. This step completes the installation process.

After the installation, the user can perform an optional but recommended step. It is possible to allow the user to run commands without authentication using the `sudoers` file. Three lines are printed on the screen after the successful installation process. Lines can be copied and pasted into the `/etc/sudoers` file. Adding these lines into the `sudoers` file enables users that are a part of group `F4GD_users` to run application scripts without entering the password.

4.2 Resources

The Resources folder stores files used as templates for project configuration files or data. The folder contains seven files. Six files are copies of the default configuration files of srsRAN. Their purpose is to serve as a template for the project configuration files. When template files are used, only the values of the appropriate variables are changed based on the network configuration that the user has set up in the application.

The last file, *data.json*, is a database of all known mobile networks. It also includes networks that are no longer in operation or are still in the planning stages. Its contents are generated by the *mcc-mnc-list* tool available on GitHub [47]. Software processes the table containing information about each mobile network operator on the Wikipedia page [48]. The records of each operator are stored in JSON format and retrieved as a list.

One network record is stored as a JSON object. Its full format can be seen in 4.4 on the example of Vodafone Czech Republic. The four most important variables for the application are *mcc*, *mnc*, *countryName*, and *operator*. Other variables are currently unused but may be helpful in the future.

■ Code listing 4.4 Network Record Format

```
{  "type": "National",
    "countryName": "Czech Republic",
    "countryCode": "CZ",
    "mcc": "230",
    "mnc": "03",
    "brand": "Vodafone",
    "operator": "Vodafone Czech Republic",
    "status": "Operational",
    "bands": "GSM 900 / GSM 1800 / LTE 800 / LTE 1800 /
              LTE 2100 / LTE 2600 / 5G 1800 / 5G 2100",
    "notes": "Former Oskar; UMTS shut down Mar 2021" }
```

The application tries to facilitate access for users with limited knowledge of mobile networks. When creating the network, the user is prompted to enter the operator's country of origin and name. The *mcc* and *mnc* values are automatically filled in using the obtained data.

The code lookup is intended to relieve the user of the complexity of finding the necessary information. The data is also used for the validation of user input. If the country or operator name entered by the user cannot be found in the data, it is considered invalid.

4.3 Scripts

The directory named *scripts* contains four bash scripts for preparing the environment, running, and shutting down srs applications. The other two scripts are used for programming SIM cards and restarting the essential system service *pcscd* for card reading.

4.3.1 *governor.sh*

Both srsEPC and srsENB require the CPU scaling governor to be set to performance mode. Each core has a dedicated file containing information about the mode it is currently running in.

Changing the mode of a core is done by editing the contents of the appropriate file.

The script ensures that all CPU cores' files are overwritten to performance. Root permissions are needed to perform the task, as setting the system behaviour is privileged.

If the performance mode is not set, the user will encounter an error message after running the applications: “*WARNING: cpu0 scaling governor is not set to performance mode. Realtime processing could be compromised. Consider setting it to performance mode before running the application.*”. The application prints the error only for the CPU cores in power-save mode.

4.3.2 `srsepc_if_masq.sh`

Packet forwarding has to be enabled to allow devices connected to the mobile network to access the Internet using mobile data. A script called `srsepc_if_masq.sh` is part of the srsRAN project. Its job is to apply a masquerade to the interface it receives as an input argument. The script is copied from the srsRAN project for improved accessibility and because the installation script provides instructions to the user on modifying the contents of `sudoers` file. Including the script as a part of the application eliminates the need to search for it on the system and minimizes the risk of disclosing an incorrect path to the user, which could pose security threats.

4.3.3 `on_air.sh`

`On_air.sh` is the main script that prepares the environment and runs the srsRAN applications. It is intended to be run without the root permissions. The script executes two other scripts using the `sudo` command in the preparation phase.

The first script is `srsepc_if_masq.sh`, which sets up the network interface and packet forwarding. The second script, `governor.sh`, sets the CPU to performance mode. Their function is described in more detail above.

After successfully setting up the environment, the script launches four new terminal windows. A separate window with the application itself and a terminal window with the application's log file in live form is launched for each srsRAN application.

The terminal windows are renamed according to their contents for ease of reference. The default position and size of the terminal window are fixed at the startup, but the user can customize it and move the windows as desired.

4.3.4 `off_air.sh`

`Off_air.sh` script shuts down running srsRAN applications and the opened terminal windows with the logs using the `pkill` command when the testing is done. Thanks to the given `CAP_KILL` capability, root privileges are unnecessary to perform its task.

It is only possible to return to the application after starting the mobile network by running this script and shutting down the running network. This feature safeguards the user from leaving networks running and having multiple networks running simultaneously.

4.3.5 `restart_pcsc_service.sh`

`restart_pcsc_service.sh` is a helper script that detects whether a service named `pcscd` is running. If not, the script restarts the service to ensure it is available. Root permissions are needed to

perform service restart successfully. Checking the service status does not need the privileges, but the restart does.

4.3.6 `sim_programming.sh`

`sim_programming.sh` script is meant to be run without root permissions. Its job is to program the SIM cards. It relies on the Osmocom's `sysmo-usim-tool` software being installed on the computer. The tool contains a Python script named `sysmo-isim-tool.sja2.py`, which is used to program the `sysmoISIM-SJA2` programmable SIM cards.

The script expects eight arguments on the input. Apart from the SIM card parameters, the administrative key required for SIM card programming, the directory path of `sysmo-usim-tool`, and the path to restart the `pcscd` service script is also passed as arguments to the script. The SIM card will not allow any parameter to be changed without providing the administration key.

`restart_pcsc_service.sh` is used to restart the `pcscd` service if inactive. After preparing the environment, it modifies the arguments for SIM card programming. Different options are used for `op` and `opc` operator codes. According to a simple if statement, it modifies the argument array. If successful, the SIM card programming script is called with an expanded array of arguments.

When done programming, the return code of `sysmo-isim-tool.sja2.py` script is checked. Non zero return code means that some error occurred. The application uses the return code to notify the user if the SIM card fails to program.

4.4 Source Codes

`src` folder contains all the Python source code, divided into several modules. The interdependency of these modules can be seen in Appendix D.1. This section describes the function of each module within the application and its main features.

4.4.1 `config_handler.py`

`config_handler.py` module consists only of functions and constants and is designed to work with configuration files. It has two functions within the project. It is used to create and read the `srsENB` and `srsEPC` configuration files. Its second task is to initialize the default application variables using the `default.conf` file.

4.4.2 `configuration.py`

`configuration.py` module contains `Configuration` class, which represents the current configuration of the mobile network. Configuration files for the `srsEPC` and `srsENB` applications are created based on the object's variables. `Configuration` uses the option to load default variables from the `default.conf` file if available.

`Configuration` includes a method which selects the network interface used as the default gateway. It uses `gateways()` function from the `netifaces` library, which returns the default gateways of the user's device. Multiple gateways can be set at once. If there is no default gateway, the return value is the default option specified in `default.conf` or fallback value.

4.4.3 `enums.py`

Using enumeration types in code can help clarify the code and make it easier to understand the meaning of variables. An object is specified for each menu that inherits from the enum class and defines the individual menu items. The names of the variables in the objects correspond to the names of the menu items. Each variable is assigned a numeric value the application uses for menu selection.

4.4.4 `file_handler.py`

`file_handler.py` module includes a class called `FileHandler` for file manipulation. It can create a new empty project and read data from the `data.json` file. Operations related to configuration files are delegated to `config_handler` module. Separating the parts that handle I/O operations provides clarity and delegates responsibility to a single module.

4.4.5 `main.py`

`main` serves as the application's entry point and provides the logic that switches between the main menus. It starts by creating `App` and, using its state, switches between menus. It also creates `stdscr` object using the curses library to manipulate the terminal and provide all the necessary functions.

4.4.6 `menu_options.py`

`menu_options.py` module overlays `enums` module. Its function is to group all menu options. The raw data from `enums` module are processed and stored with the `get_options` function. The menu items are then imported into `App` module to display individual menu options.

4.4.7 `menu_wizard.py`

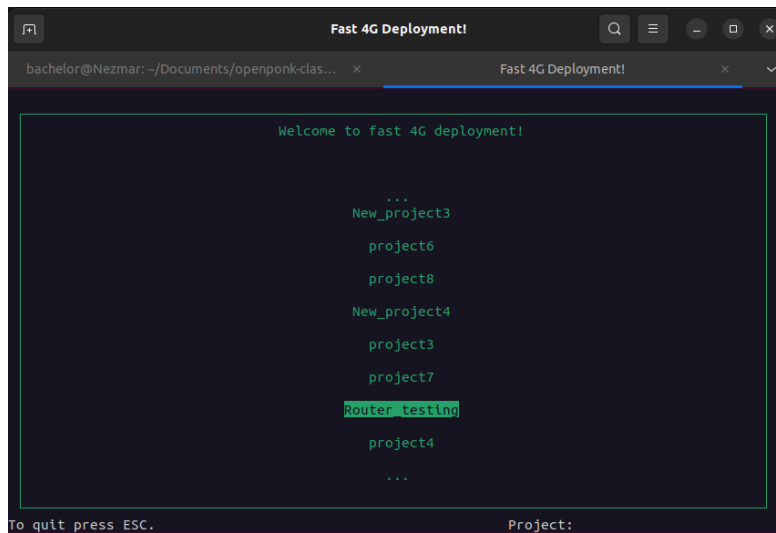
`menu_wizard` module includes a class named `App`, which holds the menu definitions available for users to navigate within the application. The logic of each menu is based on the `match...case` syntax, which may be known as `switch...case` from other C-like programming languages.

It has its default values listed in the `default.conf` configuration file, and tries to retrieve these values when creating `App` object, if possible.

A wrapper has been created, and it is used on each menu. It provides the ability to print options on the screen, move the cursor between the menu options, select the appropriate option, resize the terminal, or quit the application.

The scrolling menu displays the first eight items on the screen, and the remaining items are shown sequentially as the user scrolls. If some options cannot be shown, the menu denotes that by three dots above or below the displayed options.

In addition to the menu definitions, the module also contains helper methods for processing and passing data within other objects. Part of `App` is an instance of `Project`. Through its interface, the application accesses and works with the data and the network settings.



■ **Figure 4.1** Menu With More Than 8 Items

4.4.8 `paths.py`

`paths` module is made up of two classes. Its purpose is to hold information about paths to configuration files, project directories, and application scripts. The paths are stored to be dynamically changed and forwarded if necessary.

`ScriptPaths` class wraps all absolute paths to scripts used within the application. When the object is created, it dynamically evaluates the paths based on the application's location. Only getters are available for the paths, as they are not expected to change at runtime.

`FilePaths` class holds all necessary paths to project files and directories. Once the name is successfully changed, all paths are automatically adjusted to align with the corresponding project. This is the only way to force the paths to change. As with `ScriptPaths`, only getters are available for individual paths.

4.4.9 `project.py`

`project.py` module is used to create an object of the same name. `Project` contains variables representing its current state, holds network settings, and manages assigned `SIM profiles`. The `Project` is the primary data object within the application.

A single `Project` object contains one `Configuration`, `FilePaths`, `File_handler`, and several `SIM Profile` objects. Only one project can be active at a time. When a new project is loaded, a new object is created and populated by the loading process. If the loading is successful, the previous `Project` is replaced by the new one. The original `Project` is retained when an error occurs during the loading, and the user is informed with an error message.

Knowing the paths to project files and directories is necessary when loading or creating a project. The project uses the `FilePaths` from `paths` module to maintain all the important information. `Project` uses `File_handler` to manage and manipulate files. `File_handler` provides an interface to create or load configuration files. `Project` leaves all responsibilities to `File_handler` and only receives data from it or information on whether the operation was successful.

4.4.10 `prompts.py`

`prompts.py` module contains basic prompts such as string, number, IP address, and others, used to create prompts for specific variables. In particular, the module handles prompts for *SIM Profile* and *Configuration*.

Reading the user input is non-blocking, allowing reading the user input sequentially and responding appropriately. Non-blocking reading allows the user to be offered a suggestion, for example, when entering the country or operator name.

Receiving input is handled by a single method called `prompt`, which reads input as strings and special characters such as arrows and keys. It can also respond to manipulating the size of the terminal window, making it possible to react and modify the screen content. The application can be quit using the *ESC* key.

The hints use data from `data.json` and a function that is part of `prompts` module called `find_first_five`, which returns up to five hints based on the input parameters.

Obtaining parameters from the user for *SIM Profile* or *Configuration* is a process consisting of a series of prompts. When retrieving the parameters, it is possible to go to the previous prompt or skip the prompt process altogether. The logic that wraps each user prompt remembers the sequence of screens and can switch between them. If the user chooses to leave the process entirely, all changes are discarded.

4.4.11 `screens.py`

`Screens` class contains the definitions of individual screens that can be displayed to the user while using the application. The class contains methods for updating the screens along with the screen definitions.

Like others, `Screens` object uses defaults from `default.conf`. The defaults contain the position, dimensions, and name of the screens. If the values can not be obtained from the file, fallback values are used instead.

An important function of the object is to react to changes in the terminal window size and adjust accordingly. If terminal manipulation is detected, `resize` method is called to handle the situation. The contents of the screens are cleared and updated to effect the resizing.

4.4.12 `sim_profile.py`

`sim_profile.py` is a module containing a class `SIM_profile` used to represent a network profile instance. The main methods of the object are `load`, `store`, and `program`.

The `load` method inputs a single line from the `user_db.csv` file. The line is split into separate parameters at the position of the comma. Single parameters are then verified using helper functions. `True` is returned if all parameters are valid and set to the object to indicate that the user was successfully loaded and created.

The `store` method returns a string of the parameters held by the object instance separated by a comma. The returned string can then be written as a valid user to the `user_db.csv` file.

The `program` method is used to obtain the parameters that are passed to the SIM programming script. It takes the necessary parameters and creates a string of a specific format, which the other methods parse and pass as parameters to `sim_programming.sh` script.

4.5 Documentation

The Sphinx tool was used to generate interactive code documentation. In particular, it uses docstrings written in the code as a documentation source. It builds web pages that can be clickable, view the source code, search, and much more from the source codes.

The tool is freely available. Installation is performed using the command (4.5). Other Sphinx installation methods can be found on the official website [49]. Examples and tutorials for more advanced options can also be found there.

■ Code listing 4.5 Sphinx Installation

```
sudo apt install python-sphinx
```

Sphinx needs to initialize the environment and prepare the folder for exporting the documentation. First, a user creates a docs folder where the documentation should be located. They enter the essential parameters inside the newly created directory using *sphinx-quickstart* to identify the author, version, and language. The command processes the values and creates a documentation structure.

The user needs to edit one of the generated files, which is *conf.py*. They should include a command that appends the path to the source directory to the *sys.path* variable. This ensures that the Python interpreter can locate the necessary modules. An error will occur if the modules cannot be found during the build process.

Additionally, the user can modify the HTML sections generated within the documentation using specific commands. They can create a CSS file and define the entire page's content for the documentation. By default, only half of the page is utilized. Furthermore, the user can enhance the documentation's appearance and functionality by using extensions. In the case of generating documentation for the application, *sphinx.ext.todo*, *sphinx.ext.viewcode*, *sphinx.ext.autodoc* extensions were utilized.

The last modification involved adding modules to the *index.rst* file, which serves as the foundation for generating the homepage. Adding modules will enable the documentation summary to appear on the homepage.

The documentation can be generated using Sphinx once all file modifications have been completed. *sphinx-apidoc* has two arguments that must be passed. The *-o* option specifies the destination where the files will be generated. The second argument is the source folder where the files to be processed are located. When executed, files with the *.rst* extension appear in the specified output folder. *make html* ensures the documentation is generated as HTML.

■ Code listing 4.6 Commands Used for Generating the Documentation

```
mkdir docs
cd docs
sphinx-quickstart
sphinx-apidoc -o . ../src
make html
```

4.6 Third Party Libraries

Libraries, `ConfigUpdater` and `netifaces` must be installed because they are not part of the Python standard library.

4.6.1 `ConfigUpdater`

The first decision was to use `ConfigParser` package in the standard library to manage the configuration files. The function for reading configuration files worked as expected, but writing data to the files did not. `ConfigParser` helps create new files, not update them. When reading a file, it reads all sections and their variables but ignores the comments. After writing, it creates a new file stripped of comments.

`ConfigUpdater` package preserves the file structure with all comments. It extends the functionality of `ConfigParser` and allows updating configuration files. The code must import both modules. The files supplied by the srsRAN project have many comments explaining the meaning or range of values for variables; it is advisable to keep them included.

4.6.2 `netifaces`

The most common application's use case is allowing the device to access the Internet and then analyzing the ongoing communication. `netifaces` library allows checking the settings of a computer's network interfaces. Finding information related to default gateways is a significant advantage for the application.

While setting network parameters in guided mode, the user is prompted to specify the network interface they wish to use to monitor communications and possibly allow UEs to access the Internet. A solution was needed to offer a default network interface option for user prompts. This allows users to either provide their own input or choose a default option.

The application uses `netifaces` functionality to look for the name of the network interface, which is used as the default gateway when prompting the network interface. The automatic discovery of the default gateway provides a default option for the user. The fallback value is found in the application's default configuration and is used if the computer has no default gateway set.

4.7 Dropping Privileges

The principle of least privilege was considered during the development of the application. The idea was to separate the application sections that do not need root privileges from those that do. The functioning of the application is divided into two parts, the terminal application and the scripts accompanying it.

The application in the terminal does not need root permissions to run. Any user can use the application to create a project, set up a network, create SIM profiles, and all but one running the network.

Going into the on-air mode requires root privileges. The companion scripts and srsRAN binaries require access to commands or parts of the file system that are privileged to the root user. Each script is designed to perform a specific job and only do what is necessary.

4.7.1 Project Scripts

Three scripts within the application require root privileges to run. Two scripts prepare the environment before going into the on-air mode, and the last script restarts the system service needed to communicate with the smart cards.

The script's content was initially part of the on-air and SIM programming scripts. After analyzing which commands and operations were privileged, such parts were separated into dedicated scripts. Dedicated scripts were then called to replace the original content. The earmarking has dramatically reduced the area where the application requires root privileges.

The installation script changes the script's owner, group, and privileges. The owner and group are changed to root and *F4GD_users*. The privileges of the scripts are defined by mask 750. Root, as the owner, can read, modify and execute the scripts. *F4GD_users* members can read and run scripts but cannot modify them. Other users are denied access, ultimately preventing any malicious changes or executions.

The application is intended to create a test network quickly. A typical use case is creating a network setup and testing the devices. The procedure can be performed several times in a row due to the tuning of network parameters. Every time the user goes from the application to the on-air mode, the application has to initialize the environment using the scripts. Since the initialization scripts are run using the sudo command, the user must authenticate by entering their password.

It is possible to add entries to */etc/sudoers* file to avoid the need for the user to enter their password every time. This file allows for the specification of a user or group, the actions they are authorized to execute, and their authority level. Additionally, an administrator can specify that a user or group does not need to authenticate when using the sudo. The scripts must be specified by absolute path so they cannot be confused with others.

4.7.2 srsRAN Applications

The official srsRAN setup instructions state that root permissions are required to run the applications [50]. The truth is that if the srsRAN applications are not executed via sudo, they either run in restricted mode or terminate immediately, but it can be avoided. Another thing is that due to permission inheritance when applications run under sudo, the root owns the logs and generated pcap files.

To identify the source of the error messages generated by the application when it fails, an examination of the srsRAN project's source code was conducted. Most errors came from a specific source code file responsible for creating threads with real-time priority.

Creating a real-time priority thread fails if the program user has insufficient privileges. On Linux systems, each user is assigned system resources. In this case, the variables *rtprio* and *priority* had to be set to a greater value. The user must be assigned resources in */etc/security/limits.conf*.

The srsEPC application creates a virtual network interface to communicate with srsENB. Similarly, it creates another virtual network interface to allow packet forwarding to and from the Internet. Two steps must be done to give srsEPC permission to create network interfaces.

It is necessary to set a capability called *CAP_NET_ADMIN* for the srsEPC binary file. The capability is used to configure network interfaces and firewalls, change the routing table,

and much more. srsEPC application needs the capability to create and manage virtual network interfaces.

The second task is to create a rule in */etc/udev/rules.d* and add the user to *netdev* group. Udev defines the device's ownership and permissions. Creating a new rule ensures that any user in *netdev* group can access the tun device. The issue with srsEPC has been resolved once the correct permissions have been assigned.

By defining the necessary permissions and prior preparation, the srsRAN applications do not need to be run using the *sudo* as they are assigned sufficient permissions and system resources. Applications can be run without using the *sudo* command. Once the permissions are set, the applications can be used. Both applications behave like they are running under a privileged user. All network functionality is preserved, and the required process privileges are significantly reduced.

Penetration Testing

Several methodologies can be followed when performing penetration testing. The methodologies aim to provide a systematic approach. Each methodology may have different priorities and focus. The Penetration Testing Execution Standard (PTES) methodology was used in the context of the thesis. The device testing section focuses on active information gathering using the Nmap tool.

5.1 PTES

PTES is a standard that addresses the methodology for conducting penetration tests. Standardising penetration testing increases the consistency and quality of the resulting report. It consists of the following seven sections [51]:

1. Pre-engagement Interactions
2. Intelligence Gathering
3. Threat Modelling
4. Vulnerability Identification
5. Exploitation
6. Post-exploitation
7. Reporting

The Intelligence Gathering phase is crucial. Proper preparation dramatically reduces the difficulty of subsequent penetration testing tasks. There are three levels of intelligence gathering [52].

Level 1 This level includes mainly automated information gathering using various bots or scripts. Obtained data do not need to be further processed or analysed.

Level 2 This level adds to the previous one. Human source processes the data collected and generates reports.

Level 3 The last level consists of a rigorous manual analysis using the tools mentioned in Level 1, and based on the gathered data, reports are generated.

The Information Gathering phase guides the actions in the subsequent Vulnerability Identification and Exploitation phase.

Target identification involves understanding the scope of the test, obtaining the necessary data and identifying the vulnerabilities. When working with a client, it is important to agree on the scope of the test to avoid misunderstandings.

The process of collecting, analyzing, and utilizing information from publicly available sources is called Open Source Intelligence (OSINT). Depending on how the data is collected, it is divided into three forms. The forms are called passive, semi-passive and active [52].

Passive form is a method of obtaining information without communication with the device or system. Usually, archived data accessible from different sources are used. However, the data may be outdated or irrelevant, making the work difficult.

Semi-passive form allows interaction with the target, but the limitation is that no actions beyond standard user behaviour are performed. There is no attempt to brute-force various options, and no tools such as web crawlers or port scanners are used. The purpose is to avoid attracting unnecessary attention.

Active form method aims to gather information by all available means. Tools for analysis, hidden directory search, domain search, web crawlers and others are used. In addition, port scanning is used to discover open ports that can be exploited. Active information gathering should be detected by the target and responded to appropriately.

5.2 Typical Test Case

Testing is carried out on the mobile operator's interface. No one has access to the interface except the operator in the commercial sector. For this reason, it is necessary to test IoT devices in virtual networks where it is possible to be on the same level as the operator. Accessing the interface between the network and the UE allows monitoring of signalling messages, authentication processes, and other messages for communication analysis. A typical example of a mobile device penetration test consists of various smaller tasks that must be performed.

1. The environment must be prepared before testing the device. The test has to be performed in a dedicated environment to avoid interference with the commercial networks. The hardware must be connected to the computer running the network simulation software.
2. srsRAN applications are configured by changing the values of variables in the configuration files. The most critical parameters are frequency, transmit power, and the country and operator code. Another important parameter is the path to the file *user_db.csv*, which specifies the users – SIM profiles – that will have access to the network.
3. Once the network has been set up and wired, it is ready for use. By starting the srsEPC and srsENB applications, the radio will start transmitting on the defined frequency. The UE with the programmed SIM card connects to the network if the parameters of the SIM card match any entry in the *user_db.csv*. In order to access the Internet using mobile data, the

UE's APN setting has to be the same as the APN set on the network, and packet forwarding has to be enabled.

Once the device is connected, several essential functions must be checked to ensure the network and the device are working correctly. An initial check prevents discovering unmet conditions or some network element from behaving unexpectedly.

Tester should verify that the network is operating in the designated frequency band. The verification requires a device called a spectrum analyser. It uses an antenna to detect activity on the frequency band, and if the network is in use, significant activity is seen on the spectrum analyser's screen.

It should be verified that the device can authenticate within the network. If the network is switched on and the device has a programmed SIM card inserted but cannot connect, it indicates a problem with the SIM card authentication. It is often possible to see the error in the logs. In case of such an occurrence, the tester shall verify that the information on the SIM card matches the information in the *user_db.csv* file and that the srsEPC configuration file has the appropriate path to this file.

After a successful connection to the network, one or more standard test cases are run. They can be fused in different ways depending on the requirements. The tests focus on a specific feature of the device. A comprehensive analysis can be achieved by combining several tests. Some of the basic tests are:

1. The connection is stable and does not fail. At the same time, it is tested whether the UE reconnects to the network after going off the idle mode.
2. Signalling messages must conform to the standards specified by 3GPP. If the device cannot operate on the network, it is important to check in the logs whether signalling messages are sent between the network and the UE and, if so, whether they comply with the standards specified by 3GPP.
3. Tests such as handover, procedure selection and others can also be part of basic testing, but they are less common. For example, there is no reason to test handover in a typical test case because only one instance of srsENB is used. At least two instances would be required to test it.

Once the basic test suite has been completed, specific device testing can follow. The tests can vary depending on the type of focus. Standard tests check a wide range of attack vectors, often revealing vulnerabilities that specific tests target in detail. The tests mainly focus on gathering information that is evaluated and passed on to the manufacturer. An example of such an information-gathering test can be using the Nmap tool and performing a port scan to see any opened ports.

5.3 Device Testing

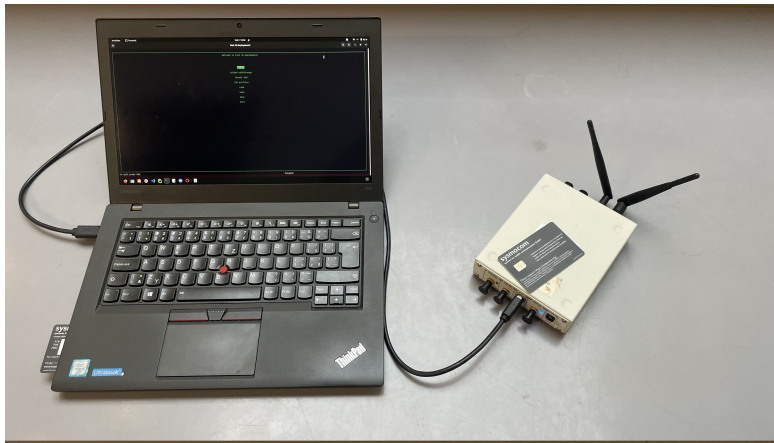
The application is being tested under real conditions on the MikroTik Chateau 5G router. The device is connected to the testing network which is created using the developed application. The router is tested using Nmap port scan based on the Information Gathering methodology described above. This section describes the undertaken testing process under laboratory conditions.

The application was used to set up test network parameters. The next step was to create a SIM profile. Its parameters were programmed into a SIM card. A prepared SIM card was inserted into the router. The last step was to run the network, and once the device was connected to the network, a scan of the router's ports was performed using the Nmap tool.

5.3.1 Environment Setup

The first step was to prepare a test environment and then create a project that defines the configuration of the test network. To avoid outdoor interference, the testing was done in an RF-shielded tent (2.3). Inside the tent, the setup was prepared to simulate the entire network.

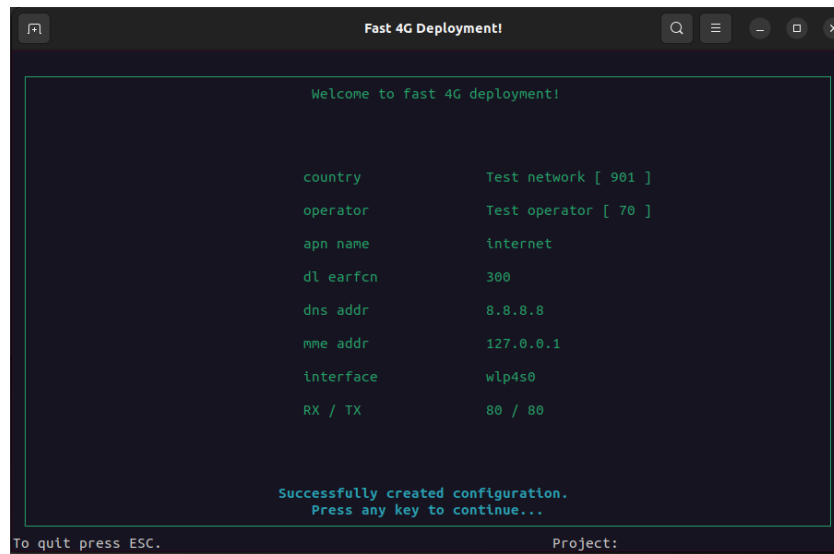
A computer, SDR, SIM programmer and a programmable SIM card were needed to set up the network (5.1). As for SDR, the setup used USRP B210 connected to the computer with a USB cable. Thanks to the built-in card reader in the laptop, SIM cards can be programmed directly without needing an external one. The SysmoISIM-SJA2 (2.1) was used as a SIM card.



■ **Figure 5.1** Testing Setup

The next step was to create a new project named *Router_testing*. Parameters of the Test network (5.2) were used to minimise the interference risk with the real mobile networks. The manufacturer's specifications for the LTE module in the router include several frequency bands. The frequency of LTE band one was chosen. Once the network was configured, a SIM profile had to be created and programmed on the SIM card according to its parameters (5.3) and inserted into the router.

After ensuring the tent was closed correctly and the environment was ready, the testing began. The application was ready to go into on-air mode.



```

Welcome to fast 4G deployment!

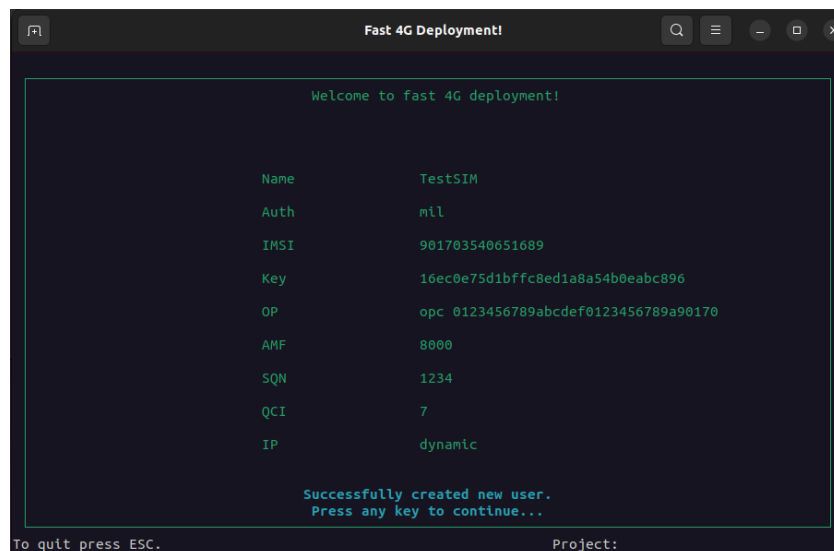
country          Test network [ 901 ]
operator         Test operator [ 70 ]
apn name         internet
dl earfcn        300
dns addr         8.8.8.8
mme addr         127.0.0.1
interface        wlp4s0
RX / TX          80 / 80

Successfully created configuration.
Press any key to continue...

To quit press ESC.                                     Project:

```

■ **Figure 5.2** Network Parameters



```

Welcome to fast 4G deployment!

Name             TestSIM
Auth             mil
IMSI             901703540651689
Key              16ec0e75d1bffc0ed1a8a54b0eabc896
OP               opc 0123456789abcdef0123456789a90170
AMF              8000
SQN              1234
QCI              7
IP               dynamic

Successfully created new user.
Press any key to continue...

To quit press ESC.                                     Project:

```

■ **Figure 5.3** SIM Profile Parameters

5.3.2 Port Scan

The srsEPC's log listed the IP address assigned (5.4) to the router once it was successfully connected to the network. The first step was to verify that the router was reachable from the laptop. Its availability was verified using the ping command with the assigned router's IP address.

After checking availability, a port scan was run to see any opened ports using the Nmap tool without additional switches, and the process was aborted immediately. The reason was that the router did not respond to Nmap's host discovery. Nmap skipped the port scanning because it thought the device was unreachable.

```

2023-04-25T10:54:37.532236 [SPGW GTPC] [I] IMSI: 901703540651689, UE IP: 172.16.0.99
2023-04-25T10:54:37.532237 [SPGW GTPC] [I] S-GW Rx Ctrl TEID 0x1, MME Rx Ctrl TEID 0x1
2023-04-25T10:54:37.532238 [SPGW GTPC] [I] S-GW Rx Ctrl IP (NA), MME Rx Ctrl IP (NA)
2023-04-25T10:54:37.532239 [SPGW GTPC] [I] S-GW Rx User TEID 0x1, S-GW Rx User IP 127.0.1.100
2023-04-25T10:54:37.532240 [SPGW GTPC] [I] eNB Rx User TEID 0x1, eNB Rx User IP 127.0.1.1
2023-04-25T10:54:37.532241 [GTPU ] [I] Modifying GTP-U Tunnel.
2023-04-25T10:54:37.532243 [GTPU ] [I] UE IP 172.16.0.99
2023-04-25T10:54:37.532245 [GTPU ] [I] Downlink eNB addr 127.0.1.1, U-TEID 0x1
2023-04-25T10:54:37.532245 [GTPU ] [I] Uplink C-TEID: 0x1
2023-04-25T10:54:37.532256 [SPGW GTPC] [D] SPGW Sending S11 PDU! N.Bytes: 1360
2023-04-25T10:54:37.532272 [SPGW GTPC] [D] SPGW S11 Sent 1360 Bytes.
2023-04-25T10:54:37.532315 [NAS ] [D] Generating MAC with inputs: Algorithm 128-EIA1, DL COUNT 2
2023-04-25T10:54:37.532316 [NAS ] [I] Packed UE EMM information

```

■ **Figure 5.4** Assigned IP Address to UE

Nmap's port scan works in two phases. The first phase uses host discovery to check if the IP address is available. If the host responds, the scan continues. Otherwise, the scan is not performed. This assumption may sometimes be incorrect, as in this case.

Based on the previous observation, the port scan had to be modified to skip the host discovery phase. Performing the scan despite not getting a ping response could be accomplished thanks to the `-Pn` switch, which stands for "No ping". The scan took about 2.5 minutes. All scanned ports were filtered (5.5).

```

bachelor@Nezmar:~/Documents/bachelors-project/src$ nmap -v -Pn 172.16.0.99
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-25 13:01 CEST
Initiating Parallel DNS resolution of 1 host. at 13:01
Completed Parallel DNS resolution of 1 host. at 13:01, 0.03s elapsed
Initiating Connect Scan at 13:01
Scanning 172.16.0.99 [1000 ports]
Connect Scan Timing: About 15.50% done; ETC: 13:04 (0:02:49 remaining)
Connect Scan Timing: About 30.50% done; ETC: 13:04 (0:02:19 remaining)
Connect Scan Timing: About 45.50% done; ETC: 13:04 (0:01:49 remaining)
Connect Scan Timing: About 60.50% done; ETC: 13:04 (0:01:19 remaining)
Connect Scan Timing: About 75.30% done; ETC: 13:04 (0:00:50 remaining)
Completed Connect Scan at 13:04, 201.20s elapsed (1000 total ports)
Nmap scan report for 172.16.0.99
Host is up.
All 1000 scanned ports on 172.16.0.99 are filtered

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 201.31 seconds

```

■ **Figure 5.5** Nmap Port Scan

A filtered port is not closed, but a firewall, an application, or a network is blocking communication on that port. No open ports were detected that could be exploited for a potential attack on the router.

The main objective of the test was to verify if the application was fully functional and its ability to set up the network for testing. As a result, the application is capable of being used in practice. It provides the user with all the functions necessary to set up the network, operate it and program the SIM cards. The output of the testing is logs and pcap files that can be used for further device analysis.

Conclusion

The objective of the thesis was to explore mobile network technologies, srsRAN and Osmocom projects that offer mobile radio suites, and create an application that leverages these projects' functionalities to facilitate a quick setup of a 4G network. The application was successfully developed and subsequently tested in practice.

The thesis describes the architecture of 4G networks. After introducing the network architecture, the srsRAN and Osmocom projects are presented. A description of the applications that are part of these projects is also included. Finally, the hardware required to build the actual test network is discussed.

The application is written in Python and consists of several modules. The choice of the programming language and the appropriate conventions are described in the analytical part of the thesis. It also discusses the application's scope and presents its structure from a high-level perspective.

The application development is described in the chapter Implementation. Once the development was complete, a device test was performed to verify that the application provided all the functionality needed to create test networks quickly.

The application is designed to simplify the testing of IoT devices. The objective is to streamline the process and allow testers unfamiliar with srsRAN and Osmocom project applications to carry out these tasks.

The resulting application has the potential for future expansion. One of the possible extensions could be to provide functionality for other generations of mobile network technologies, such as 2G or 5G. Furthermore, using a scripting language makes it possible to fully automate some of the processes and thus perform the entire testing process could be automated.

Installation Manual

This chapter serves as an installation guide for the application. It discusses the prerequisites that must be met before running the installation script. It walks through running the installation script and the options available after installation.

A.1 Prerequisites

Several prerequisites must be met for the application to be installed. The first requirement is downloading the application package. It can be found on the attached media or the school's GitLab. Use command A.1 to clone the repository into the current directory. The repository and the attached media contain all the necessary files related to the application's components.

■ **Code listing A.1** Cloning Gitlab Repository

```
git clone https://gitlab.fit.cvut.cz/tichyj13/bachelors-project.git
```

The installation script sequentially checks the prerequisites. The installation process is aborted if they are not met. The application requires Python version 3.10 or higher to run. Along with Python, pip3 also needs to be installed. pip3 acts as a package manager for Python packages.

The next is lookup for the software required by the application. The first check is whether the srsRAN project and its applications exist. If found, the script checks if an Osmocom package called *sysmo-usim-tool* is accessible. The application can be installed without *sysmo-usim-tool* being present. However, the SIM programming feature becomes unavailable.

A.2 Installation Script

Almost the entire installation process is handled by a prepared installation script. The first stage checks that all the software mentioned in the previous chapter is installed. In the second phase, it modifies access rights, assigns permissions, and creates the necessary files and folders.

The script is executed with the A.2 command. The sudo command must be used to run *install.sh* as it requires root permissions to set permissions, modify system configuration files, and handle user group assignments. The installation status and results are printed on the screen. The behaviour of the installation script is described in detail in chapter 4.1.5.

■ Code listing A.2 Running Installation Script

```
sudo ./install.sh
```

A.3 Optional Setup

When the installation script finishes, it prints out lines the user can copy and paste into a file */etc/sudoers*. The file is meant to be accessed by the *visudo* command. After pasting these lines into the file, the user who installed the application can execute the scripts with the *sudo* command without having to authenticate with the password.

The file is not intentionally modified in the installation script. The user can decide whether or not to change *sudoers*. However, modifying this file is at their own risk.

After successful installation, the computer must be restarted to reinitialize the environment. Running the application without rebooting will result in an error because the changes have not yet been applied.

Appendix B

User Manual

This chapter serves as a user guide for using the application. It includes a description of the application's navigation, menus, and controls.

B.1 Screen Layout

The application extends across the entire terminal window. Inside the borders is an area where the screens and their contents are displayed. Below the border are two messages. On the left is a message informing the user that they can exit the application at any time using the ESC key, and on the right is a message showing the name of the currently loaded project. If there is an asterisk after the project name, the project has some unsaved changes. After saving, the asterisk will disappear.

B.2 Navigation

The arrows navigate between menu items and select by pressing the Enter key. In the menu, the user can move up and down. In addition, if the user enters the configuration parameters in guided mode or creates a new SIM profile, it is possible to use the left arrow to return by one screen. If the user presses the left arrow on the first prompt screen in guided mode or when creating a SIM profile, the prompt stops, and the user is returned to the menu.

B.3 Getting User Input

The application only accepts text input if the blinking block cursor is visible on the screen. If it is not visible, the application only responds to the arrow keys, the Enter key, and the ESC key. Acceptance of input is limited by the maximum length of input that can be entered. Maximum input length is part of the *default.conf* file. If the value cannot be read from the file, a fallback value is used, which is set to a maximum of 70 characters.

B.4 On-air Mode

The first item of the main menu is used to switch to on-air mode, which is accessible only after a project is created or loaded. Configuration files for srsEPC and srsENB are created based on the network settings. A script within the application is responsible for preparing the environment and running the applications. Each application is run in its terminal, and other terminals are opened to display their logs in live form. The application does not check if SDR is connected and can be used. It is important to only run on-air mode in places without interference with other signals.

Once in the on-air mode, it is impossible to use the application. A screen is displayed, waiting for any button to be pressed, except ESC, to switch from on-air back to off-air mode. The transition ensures that the srs applications are shut down, and SDR is finished emitting. After the transition, the pcap files can be found in the *pcap* project directory and used to analyze the communication. For each test, pcap files are created with a name containing the date and time when the configuration was created. The logs are created each time on-air mode is run, and the versioning works the same with the pcap files.

B.5 Guided Mode

The Guided Mode is used to guide the user through the process of setting all the necessary parameters that need to be entered to create a mobile testing network. The user enters guided mode by selecting the Guided Walkthrough item. When a user with no project loaded or created opens guided mode, firstly, they are prompted for the name of the new project. The guided mode modifies the current project if the user has already created or loaded a project.

The user sees the default value at each prompt, which is used in case they press Enter without typing anything. Some prompts have a suggestion window. When the user starts typing, the content of the suggestion window starts updating. The > character appears with the first suggestion, meaning the marked suggestion is used when the user confirms the input by pressing Enter. This functionality works like some form of auto-completion.

When entering numeric values, a hint in addition to the default value appears on the screen, telling the user what range the value should be in. An error is displayed when the value is not in the appropriate range. The user is expected to press any key to clear the displayed error message, and then they are returned to the prompt screen.

Once all the network configuration prompts have been completed successfully, the user is presented with a menu with three options to choose from:

Create SIM profile option takes the user to the next set of prompts used to fill in the parameters of the SIM profile. The behaviour of the screens is the same as when entering network configuration parameters. Once all the values have been entered, a single SIM profile is created. A menu asking if the user wishes to create another profile or proceed to the main menu is displayed.

Load SIM profile file option allows the user to load SIM profiles from an existing file. The user is presented with a screen asking for a file path. If the file exists, all valid SIM profiles are loaded. Otherwise, an error message is shown, and the user is returned to the main menu.

Done exits guided mode and immediately returns the user to the main menu without creating or loading any SIM profile.

B.6 Manual Edit

Once the project has been created or loaded, the option to manually edit the network configuration parameters becomes available. This option is mainly used when only a few parameters must be changed quickly. The user is taken to a screen where they can see each network's parameters and their current values.

The screen with parameters works like a menu, so the user selects the parameter with the arrow keys. Once selected, the user is prompted to enter the new value. The new value overwrites the original values. The default value shown on the prompt screen is the original value. Leaving the prompt without making changes is possible using the left arrow or pressing Enter key. Return is the last option on this screen, which returns the user to the main menu.

B.7 SIM Profile Management

The SIM profiles menu is used to manage SIM profiles. The menu consists of five options. If there are no SIM profiles in the project, every option except Add SIM profile is disabled.

List SIM profiles summarises all the current SIM profiles in the project. The summaries are displayed one after the other, and the next one is displayed after pressing any key except ESC.

Add SIM profile is used to create a new SIM profile. It guides the user through similar prompts as in guided mode. The option can be exited without creating a profile.

Edit SIM profile first displays the names of existing SIM profiles as a menu. After selecting one, a summary of its parameters is displayed as a menu. After selecting an option, a prompt is displayed to allow the user to enter a new value.

Remove SIM profile displays names as a menu. After selecting the SIM profile name that the user wishes to delete, this profile is removed from the project's profiles, but the change is saved immediately, and the *user_db.csv* file is not modified. To change SIM profiles stored in *user_db.csv* project needs to be saved.

Program SIM card displays a list of SIM profile names. The SIM card is programmed using the parameters of the selected SIM profile. The menu guides the user through the programming steps, informing them when to insert the SIM card into the programmer, and the application script does the programming itself.

B.8 Project Management

The Load and Save options in the main menu are used to manage projects. The Load option displays a menu with two items, Load Project or SIM Profiles.

Only projects from the */Documents/F4GD* folder can be loaded. Other locations are not taken into account by the application. If the user loads a project for another already loaded

project, a check is performed to see if all changes are saved. If unsaved changes exist in the currently loaded project, a window asking if the user wants to save or discard the changes appears. After the decision, all available projects in */Documents/F4GD* appear on the screen. The user selects one project. The application checks if the selected project contains all the necessary files and their contents are valid. If not, the previous project remains loaded. If the checks are OK, a new project is loaded for the user, including the SIM profiles and network configuration.

Loading the SIM profile definition file works like in the guided mode. The user is prompted for the path to the *user_db.csv* file. The default option is set to an example file in the application's resources folder. The loading is OK if the SIM profiles are valid and there are no user name conflicts when integrating them into the project. SIM profiles with errors are not loaded.

The Save item displays a menu with options for saving the whole project or its configuration files. The Save Project option allows users to save the current project by overwriting configuration files. It is also possible to save a project under a different name. This option is called Save as. The original project remains unchanged, and all files representing the current network settings are exported to the newly created project. The remaining options export parts of the project configuration to the */tmp* folder under the default name.

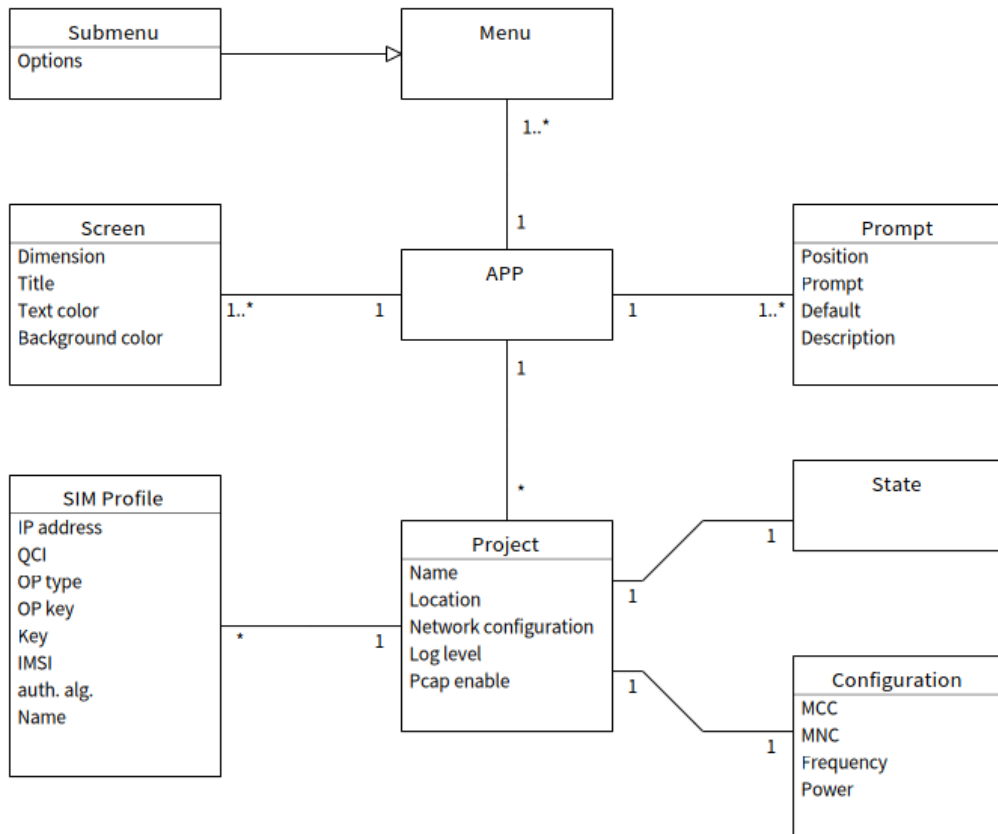
B.9 Showing Current Parameters

Selecting Show from the Main Menu displays a menu with two options. The Show Configuration item summarizes the parameters for the current network settings. The Show users item then displays a summary of the individual SIM profiles within the project, similar to the SIM profile list. Moving to the following profile is done by pressing any key.

B.10 Exiting the Application

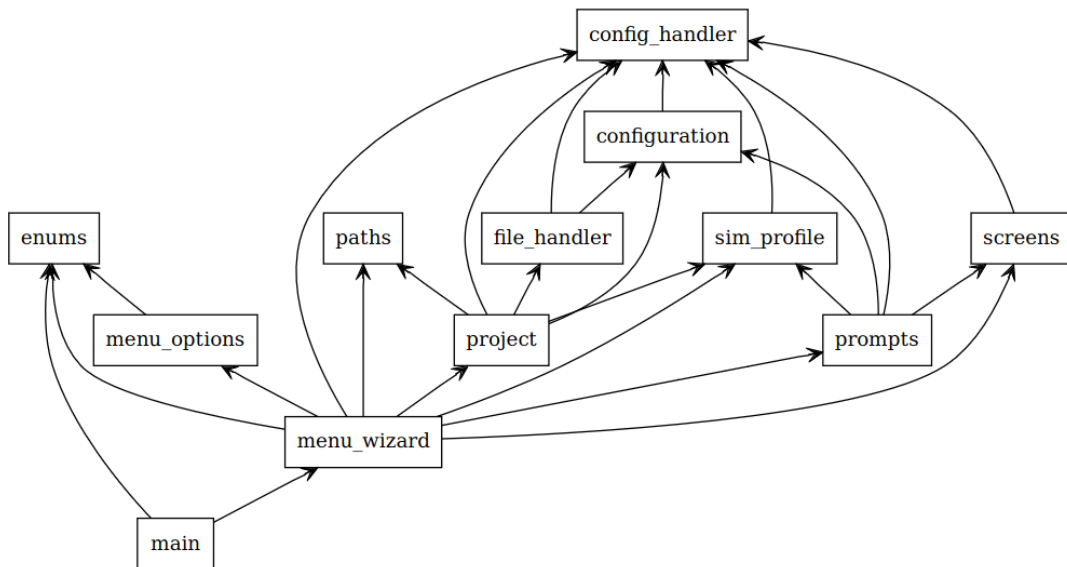
There are two ways to exit the application. The main menu's last option is Exit, which prompts the user to save unsaved changes. The application will exit without showing the prompt screen if the project is saved. In addition to this option, the application can be exited anytime using the ESC key. This option does not consider the project's status, and all unsaved changes are discarded without the possibility of undoing them.

Domain Model



■ Figure C.1 Domain Model

Modules Dependency



■ Figure D.1 Module Dependency

Bibliography

1. *Introducing 3GPP* [online]. 3GPP, 2022 [visited on 2023-04-04]. Available from: <https://www.3gpp.org/about-us/introducing-3gpp>.
2. GALAZZO, Richard. *Timeline from 1G to 5G: A Brief History on Cell Phones* [online]. CENGN, 2022 [visited on 2023-04-04]. Available from: <https://www.cengn.ca/information-centre/innovation/timeline-from-1g-to-5g-a-brief-history-on-cell-phones/>.
3. JONES, Dan; BERNSTEIN, Corinne. *What is a Radio Access Network (RAN)?* [Online]. TechTarget, 2021 [visited on 2023-04-04]. Available from: <https://www.techtarget.com/searchnetworking/definition/radio-access-network-RAN>.
4. LÁNIKOVÁ, Simona. *Testovanie bezpečnosti v mobilných siet'ach*. 2022. Bakalárska práca. České vysoké učení technické v Praze, Fakulta informačních technologií.
5. BOSSE, John G. v.; DEVETAK, Fabrizio U. *Signaling in Telecommunication Networks*. 2. Aufl. Hoboken: Wiley-Interscience, 2006. ISBN 9780470048146.
6. *LTE Architecture – Detailed Explanation* [online]. InterviewBit, 2022 [visited on 2023-04-04]. Available from: <https://www.interviewbit.com/blog/lte-architecture/>.
7. *Full LTE architecture and components* [online]. YateBTS, [n.d.] [visited on 2023-04-04]. Available from: <https://yatebts.com/documentation/concepts/lte-concepts/>.
8. GHAYAS, Adnan. *Circuit-Switching and Packet-Switching in mobile networks* [online]. Commsbrief, 2019 [visited on 2023-04-04]. Available from: <https://commsbrief.com/circuit-switching-vs-packet-switching-in-2g-3g-4g-and-5g/>.
9. HUAWEI TECHNOLOGIES CO. LTD. *LTE System Overview* [Unpublished]. 2014.
10. RIZZO, Carmine. *Lawful Interception in mobile networks* [online]. 3GPP, 2022 [visited on 2023-04-04]. Available from: <https://www.3gpp.org/technologies/li>.
11. *LTE network infrastructure and elements* [online]. LTE Encyclopedia, [n.d.] [visited on 2023-04-04]. Available from: <https://sites.google.com/site/lteencyclopedia/lte-network-infrastructure-and-elements>.
12. *LTE components* [online]. j2sw, 2022 [visited on 2023-04-05]. Available from: <https://blog.j2sw.com/gear/wireless-2/lte-components/>.

13. *LTE Architecture / LTE network architecture* [online]. RF Wireless World, 2012 [visited on 2023-04-04]. Available from: <https://www.rfwireless-world.com/Tutorials/LTE-network-architecture.html>.
14. CHEN, Hsiao-Hwa; YE, Feng; QIAN, Yi. *Security in Wireless Communication Networks*. 1st ed. Newark: John Wiley & Sons, Ltd, 2022. ISBN 9781119244363.
15. AYERS, Richard; BROTHERS, Sam; JANSEN, Wayne. *Guidelines on Mobile Device Forensics*. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2014. Available from DOI: <https://doi.org/10.6028/NIST.SP.800-101r1>.
16. ASIF, Saad Z. *Next Generation Mobile Communications Ecosystem: Technology Management for Mobile Communications*. Hoboken: Wiley, 2010. ISBN 9780470747469.
17. *How to Remove a SIM Card* [online]. Hybrid Sim, 2023 [visited on 2023-04-04]. Available from: <https://hybridsim.com/remove-sim-card/>.
18. *Home* [online]. Osmocom, 2022 [visited on 2023-04-04]. Available from: <https://osmocom.org/>.
19. *Tool to (re)configure the sysmoUSIM and sysmoISIM cards* [online]. Sysmocom, [n.d.] [visited on 2023-05-03]. Available from: <https://gitea.sysmocom.de/sysmocom/sysmo-usim-tool>.
20. *pySim* [online]. Osmocom, 2022 [visited on 2023-05-03]. Available from: <https://osmocom.org/projects/pysim>.
21. *pySim Wiki* [online]. Osmocom, 2022 [visited on 2023-05-03]. Available from: <https://osmocom.org/projects/pysim/wiki>.
22. *Who we are* [online]. Software Radio Systems, 2023 [visited on 2023-04-04]. Available from: <https://srs.io/about-us/>.
23. *Open Source RAN* [online]. Software Radio Systems, [n.d.] [visited on 2023-04-04]. Available from: <https://www.srslte.com/>.
24. *Introduction* [online]. Software Radio Systems, 2023 [visited on 2023-04-04]. Available from: https://docs.srsran.com/projects/4g/en/latest/usermanuals/source/srsepc/source/1_epc_intro.html.
25. LINDNER, Mark A. *libconfig* [online]. 2012. [visited on 2023-04-04]. Available from: https://hyperrealm.com/libconfig/libconfig_manual.html.
26. ETSI. *TS 102 221: Smart Cards; UICC-Terminal interface; Physical and logical characteristics (Release 15)* [Technical Specification]. 2018. [visited on 2023-05-03]. Available from: https://www.etsi.org/deliver/etsi_ts/102200_102299/102221/15.00.00_60/ts_102221v150000p.pdf.
27. *SysmoISIM-SJA2 Sim + USIM + ISIM card (10-Pack) with ADM keys* [online]. Sysmocom, 2022 [visited on 2023-04-04]. Available from: <https://shop.sysmocom.de/sysmoISIM-SJA2-SIM-USIM-ISIM-Card-10-pack-with-ADM-keys/sysmoISIM-SJA2-10p-adm>.
28. *HID OMNIKEY 3121* [online]. HID Global Corporation, 2023 [visited on 2023-04-04]. Available from: <https://www.hidglobal.com/products/omnikey-3121>.

29. GRAYVER, E. *Implementing Software Defined Radio* [online]. Springer New York, 2012 [visited on 2023-04-04]. ISBN 9781441993328. Available from: <https://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=994364>.
30. BURNS, P. *Software Defined Radio for 3G* [online]. Artech House, 2003 [visited on 2023-04-04]. ISBN 9781580538046. Available from: <https://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=257603>.
31. *What we do* [online]. Federal Communications Commission, [n.d.] [visited on 2023-04-04]. Available from: <https://www.fcc.gov/about-fcc/what-we-do>.
32. *Radio Spectrum: The basis of wireless communications* [online]. Directorate-General for Communications Networks, Content and Technology, 2022 [visited on 2023-04-04]. Available from: <https://digital-strategy.ec.europa.eu/en/policies/radio-spectrum>.
33. *Mrtvá komora na ČVUT. Nejtišší místo v Praze se používá k akustickým měřením* [online]. Český rozhlas, 2017 [visited on 2023-04-04]. Available from: <https://wave.rozhlas.cz/mrtva-komora-na-cvut-nejtissi-misto-v-praze-se-pouziva-k-akustickym-merenim-5965697>.
34. *Large Faraday Tent – RF/EMI Shielding Enclosure Room (7 x 7 x 6.5)* [online]. FARADAY DEFENSE, 2023 [visited on 2023-04-04]. Available from: <https://shop.faradaydefense.com/product/faraday-tents-rf-emi-shielding-enclosure-rooms-7-x-7-x-6-5/>.
35. *5 INFAMOUS IOT HACKS AND VULNERABILITIES* [online]. Fira de Barcelona, 2018 [visited on 2023-04-04]. Available from: <https://www.iotsworldcongress.com/5-infamous-iot-hacks-and-vulnerabilities/>.
36. *NÚKIB vydal Varování před použitím chytrých elektroměrů ze země s nedůvěryhodným právním prostředím* [online]. 2022. [visited on 2023-04-04]. Available from: <https://nukib.cz/cs/infoservis/hrozby/1837-nukib-vydal-varovani-pred-pouzitim-chytrych-elektromeru-ze-zemi-s-neduveryhodnym-pravnim-prostredim/>.
37. *Functional and Nonfunctional Requirements: Specification and Types* [online]. AltexSoft, 2021 [visited on 2023-04-04]. Available from: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/>.
38. ROSS, Ronald; PILLITTERI, Victoria; DEMPSEY, Kelley; RIDDLE, Mark; GUISSANIE, Gary. *Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations [including updates as of 01-28-2021]*. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2021. Available from DOI: <https://doi.org/10.6028/NIST.SP.800-171r2>.
39. *Installation guide* [online]. Software Radio Systems, 2023 [visited on 2023-04-04]. Available from: https://docs.srsran.com/projects/4g/en/latest/general/source/1_installation.html.
40. *Curses programming with python* [online]. Python Software Foundation, 2023 [visited on 2023-04-04]. Available from: <https://docs.python.org/3/howto/curses.html>.
41. ROSSUM, Guido van; WARSAW, Barry; COGHLAN, Nick. *PEP 8 – Style Guide for Python Code* [online]. 2023. [visited on 2023-04-04]. Available from: <https://peps.python.org/pep-0008/>.

42. *pep8 - Python style guide checker* [online]. Python Software Foundation, 2023 [visited on 2023-04-04]. Available from: <https://pypi.org/project/pep8/>.
43. *Modules* [online]. Python Software Foundation, 2023 [visited on 2023-04-04]. Available from: <https://docs.python.org/3/tutorial/modules.html>.
44. *The Python Standard Library* [online]. Python Software Foundation, 2023 [visited on 2023-04-04]. Available from: <https://docs.python.org/3/library/>.
45. *PEP 257 - Docstring Conventions* [online]. 2022. [visited on 2023-04-04]. Available from: <https://peps.python.org/pep-0257/>.
46. *Welcome* [online]. 2023. [visited on 2023-04-04]. Available from: <https://www.sphinx-doc.org/en/master/index.html>.
47. *mcc-mnc-list* [online]. [N.d.]. [visited on 2023-04-04]. Available from: <https://github.com/PodgroupConnectivity/mcc-mnc-list>.
48. *Mobile country code* [online]. Wikimedia Foundation, 2023 [visited on 2023-04-04]. Available from: https://en.wikipedia.org/wiki/Mobile_country_code.
49. *Installing sphinx* [online]. [N.d.]. [visited on 2023-04-04]. Available from: <https://www.sphinx-doc.org/en/master/usage/installation.html>.
50. *LTE setup guide* [online]. Software Radio Systems, 2023 [visited on 2023-04-04]. Available from: https://docs.srsran.com/projects/4g/en/latest/usermanuals/source/1_setup.html.
51. *High Level Organization of the Standard* [online]. pentest-standard, 2014 [visited on 2023-04-04]. Available from: http://www.pentest-standard.org/index.php/Main_Page.
52. *Intelligence Gathering* [online]. pentest-standard, 2014 [visited on 2023-04-04]. Available from: http://www.pentest-standard.org/index.php/Intelligence_Gathering.

Contents of Attached Media

readme.txt	a brief summary of the contents
src	
├─ F4GD	the implementation source files
├─ thesis	the source files of the thesis text in L ^A T _E X format
text	the thesis text
├─ thesis.pdf	the thesis text in PDF format