Hochschule Karlsruhe
University of
Applied Sciences

LTE Scanner

Author: Abdullah Alibrahim , MN: 68532
Instructor: Prof. Dr. -Ing. Manfred Litzenburger

University of Karlsruhe
January 2023

Table of Contents

List of Figures

I Description

1 Overview

LTE Scanner is an application designed to be completable to use with Software Defined Radio Devices (USRP B200/B210), with the following functions:

- Scan LTE downlink bands to detect LTE Cells on all frequencies supported by the SDR(software defined radio) device.
- Decode broadcasting channel messages, SIB1 (system information block).
- Provides WebUI (Web user interface) for management and results visualization.

2 WebUI

This section describes the usage of the application based on the functions, that are implemented in the WebUI. After starting the application, the WebUI will be accessible on the local server on port 2250 by Default.

2a Navigation Bar

- /Home: The main page
- /Plot: Starts plotting server, that plots the data from Cells.json as live stream.
- /Cells.json: Returns an array of all Cells as objects, which are live detected or loaded from saved results.
- /Estimation: Returns an object, that contains the frequencies where cells are detected but the BCCH decoding failed. It is helpful after scanning a band with no results. And each frequency has an array of initial cell id's. Example: {"frequency":[cell id , cell id , ...], "another frequency":[cell id] }.
- /Saves: Returns all saved results.
- /IMSI: Returns pdf file contains a list of (MNC, MCC) in Germany.
- /LTE Bands: Views the LTE band allocations in Germany.

2b Submit form

- USRP parameters:
  - Number of antennas: Number of receive channels (1 or 2)
  - Receiver gain: Defaults between 40 and 80.

- Frequency parameters:
  - Unit in MHz
  - To scan only one frequency, set the End frequency to „–1".
  - Use the step input to determine the frequency resolution.

- Advanced parameters:
    - Timeout: the value is in seconds and it determines how long should the scanner try to find a cell on each frequency.
    - Attempts: determines how many times must the scanner try to decode the BCCH after it fails.

- /Scan: Submits the order to the server.
- /Reset: Deletes all the results saved in Cells.json.

2c Save, load and delete

- Load/Save:
    - Save: It saves all the results from Cells.json under an Id of the users choice, to save just enter the „save id“ and be sure the „load id“ and the delete option are empty then click the button.

    - Load: It adds saved results to the Cells.json, to load just enter the load id and be sure the save id and the delete option are empty then click the button.

    - Delete: It deletes the last loaded data permanently, to delete data first click on reset then load the results you want to delete then be sure that the load id and the save id are empty then click on the delete checkbox to be “True” then click the Load/Save button.

II Requirements

3 Hardware requirements

- USRP B200/B210
- Antenna

4 Software dependencies

4a Node.js/npm

Version (v19.3.0)

Node.js is an open-source back-end JavaScript runtime server environment. It runs on the V8 JavaScript Engine and executes JavaScript code outside a web browser. NPM is the package manager for Node.js.

4b Openphy

OpenPHY is a LTE UE receiver implementation for real-time test, decoding, and network diagnostic purposes. Alternatively the implementation can serve as the basis for a full software UE implementation when combined with uplink and MAC/RRC layer functionality.

Link:
https://github.com/ttsou/openphy

General:

- LTE Release 8
- UE Category 3 (75 Mbps downlink)
- FDD mode

LTE specifications:

- 3GPP TS 36.211 “Physical channels and modulation”
- 3GPP TS 36.212 “Multiplexing and channel coding”
- 3GPP TS 36.213 “Physical layer procedures”

Physical Layer:

- Bandwidth: 1.4, 3, 5, 10, and 20 MHz (automatic detection)
- Channels: PSS, SSS, RS, PBCH, PCFICH, PDCCH, PDSCH
- PDCCH formats (DCI): 0, 1, 1A, 1C, 1D, 2, 2A

- Modulation: QPSK, QAM-16, QAM-64
- MIMO: 2x2 or 2x1 transmit mode 2 (diversity)

Decoding (TurboFEC):

- Turbo decoding for PDSCH (SIMD optimized)
- Convolutional decoding for PBCH and PDCCH (SIMD optimized)
- Turbo and convolutional rate matching

RF device support:

- Ettus Research USRP B200/B210
- Ettus Research USRP X300/X310

Processor Support:

Intel SSE3, SSE4, and AVX2 instruction support is automatically detected and enabled at build time if available.

Openphy dependencies:

- FFTW for computing the discrete Fourier transform (DFT) http://fftw.org
- USRP Hardware Driver for Ettus radio device support http://uhd.ettus.com

4c ASN1 compiler

The ASN.1 compiler for C is a standalone program that takes one or more input files, where each input file contains one or more ASN.1 modules, and generates C source code for encoding and decoding ASN.1 messages. The compiler verifies that the ASN.1 schema is valid, and generates the following:

- Diagnostic messages and optionally an output ASN.1 listing
- Easy to use C language data structures to be included in your application
- A control table for use by the space-optimized or lean encoder/decoder
- A time-optimized encoder/decoder

It is compiled and integrated in the LTE Scanner source code, and it is used to decode the LTE BCCH messages.

5 API's workflow

The ltedecoder (openphy) sets the USRP up and defines its parameters to search on a given frequency. If LTE-Cell is detected it decodes the PDSCH (physical downlink shared channel) and pushes the SIB1(system information block) bits in HEX format out (as shown in Figure 1 LTE), then the SIB1 bits will be saved and sent to the ASN1 compiler to decode the BCCH (broadcasting channel) message.

Node.js manages the process described above through the LTE Scanner app (this project)

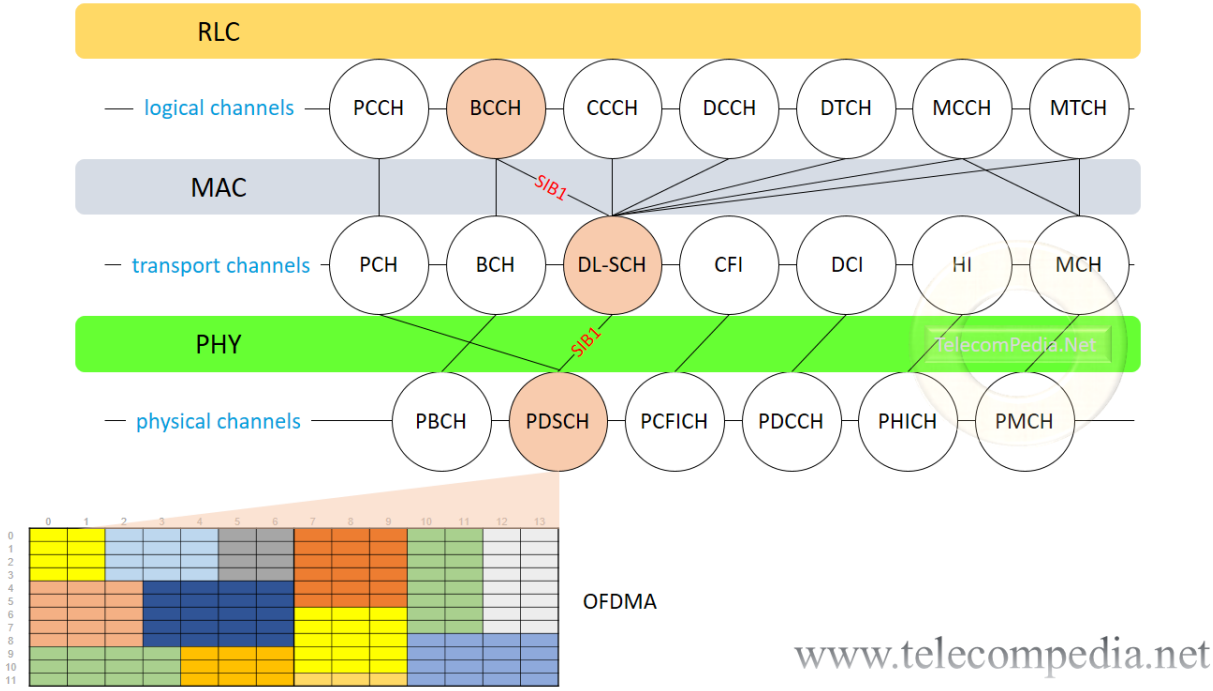

Figure 1 LTE channels

III Installation and setup

6 Operating system

Ubuntu 22.04.1 LTS (Jammy Jellyfish) is recommended to be used. It can be installed using the following link:
https://releases.ubuntu.com/22.04/

6a Install Ubuntu on a USB

Follow the instructions on the following website to install ubuntu on an USB stick.
https://itsfoss.com/intsall-ubuntu-on-usb/

7 Dependencies installation guide

Note: The order of installing the dependencies is important.

7a Update and upgrade ubuntu repositories

After installing the OS the default package manager of ubuntu apt will be used to install the dependencies. Update and upgrade its repositories by executing the following commands on the terminal:

```
1. sudo apt update
2. sudo apt upgrade
```

7b USRB Hard Drive

Execute the following command to install the USRB driver:

```
1. #install UHD packages
2. sudo apt-get install libuhd-dev uhd-host
3.
4. #download UHD images
5. sudo /usr/lib/uhd/utils/uhd_images_downloader.py
```

7c FFTW

```
1. #install build packages
2. sudo apt install build-essential
3.
4. #download source code, then extract it and enter the repository
5. wget http://www.fftw.org/fftw-3.3.10.tar.gz
6. tar -xf fftw-3.3.10.tar.gz
```

```
cd fftw-3.3.10/

#configure, build and install
./configure
make
sudo make install
```

7d OpenPHY

```
#install packages and libraries
sudo apt install git
sudo apt install autoconf
sudo apt install libtool

#clone source code and enter the repo
git clone https://github.com/ttsou/openphy.git
cd openphy/

#configure, build and install
autoreconf -i
./configure
make
sudo make install
```

7e Node.JS

```
#install packages
sudo apt install nodejs npm
sudo npm install -g n
sudo n latest
```

Note: Reboot the system then continue.
Recommended: install chrome browser then set it as default browser and from chrome settings, allow pop-ups and redirects.

7f LTE Scanner

```
#set git config and clone source code
git config --global user.name "elpmiS"
git config --global user.email "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@github.com"
git clone https://github.com/e1pmiS/LTE_Scanner.git

#enter the repository and install Nodejs dependencies
cd LTE_Scanner/
npm install

#To start the API.
node main.js
```

At this point the WebUI will be locally accessible on the following link:
http://localhost:2250

IV Test Cases

8 LTE band 800

After determining the frequencies, which on it LTE cells could be detected, in this example the frequencies are (796 MHz, 806 MHz, 816 MHz). On the WebUI (as shown in Figure 2 WebUI - Submit parameters), the start frequency is set to 796 MHz and the step value to 10 and the end frequency to 816 MHz, and the other parameters are kept with the default values. Then the order is submitted by clicking on the scan button and after while the results are automatically live streamed as shown in Figure 3 WebUI - Results and Figure 4 Node plot - Results.

On the server side the console logger is designed to show more specific information about the scanning process as shown the (console log) section below.

8a WebUI

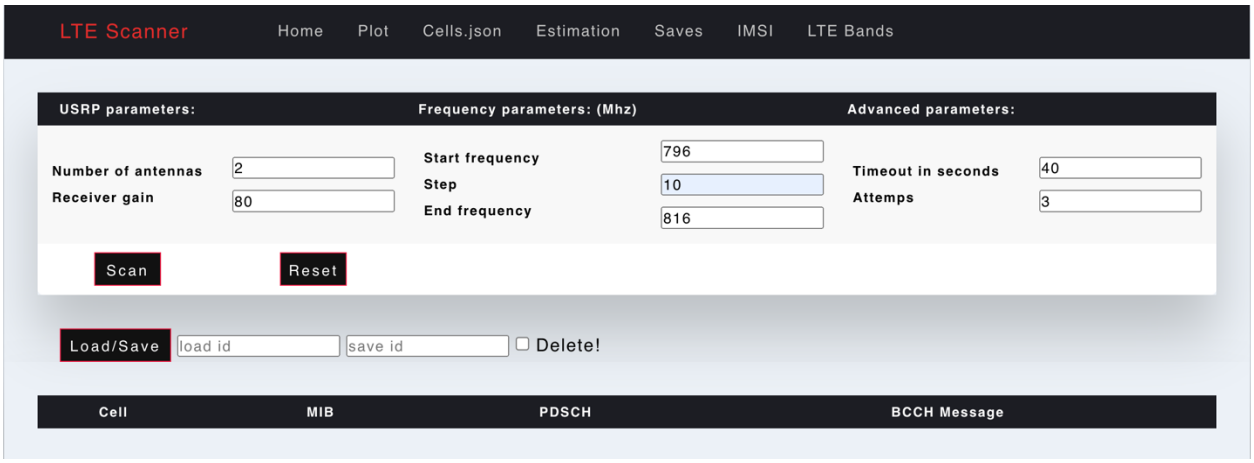

Figure 2 WebUI - Submit parameters

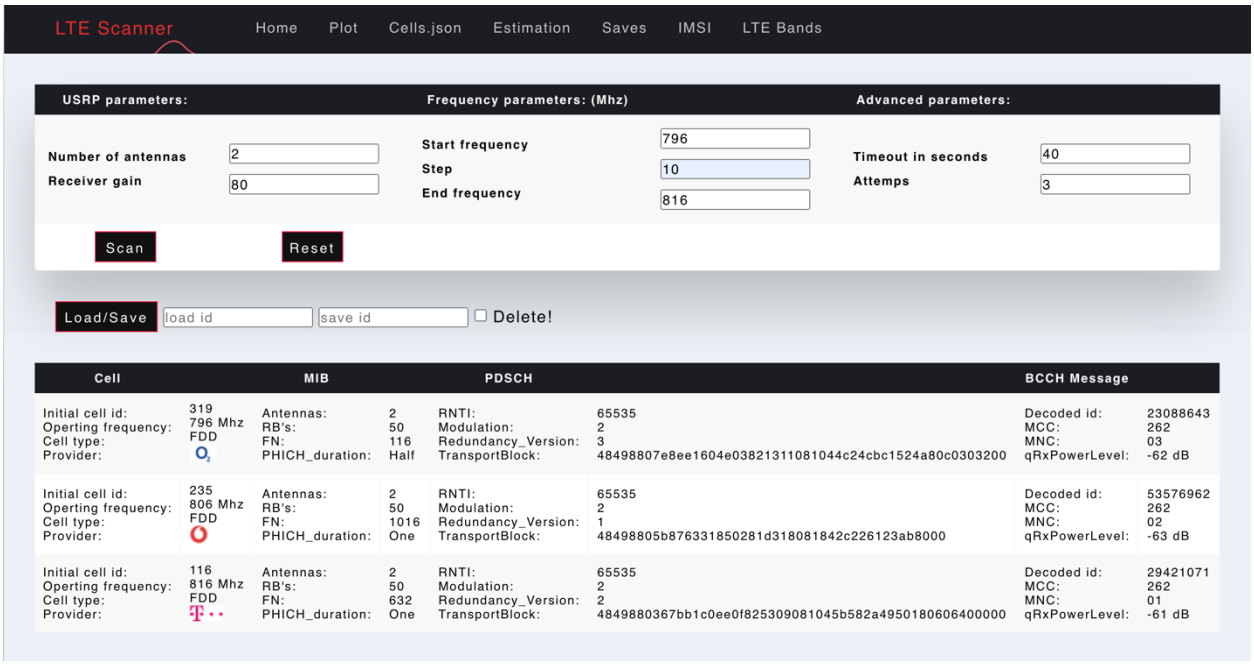

Figure 3 WebUI - Results

8b Node Plot

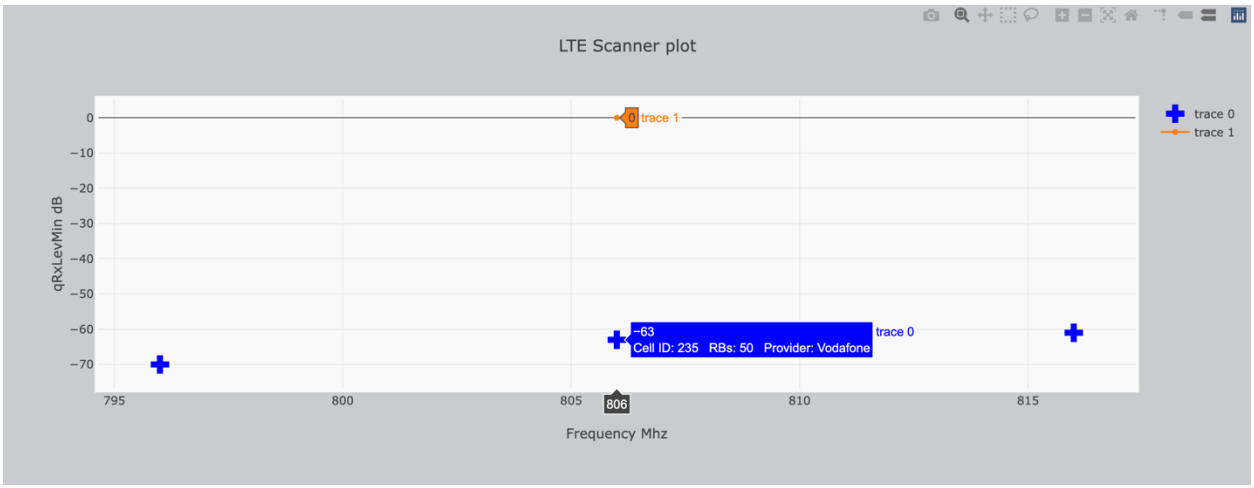

Figure 4 Node plot - Results

8c Console log

```
13:41:31 LTE_Scanner is runing on port: http://localhost:2250
[Nodeplotlib] Server running at http://localhost:35769

14:7:26 Scaning ... frequency = 796 Mhz
14:8:6 Cell detected on frequency = 796 , Cell ID = 319
14:8:6 Decoding BCCH-DL-SCH-Message ...
14:8:6 BCCH-DL-SCH-decode passed!

14:8:6 Scaning ... frequency = 806 Mhz
14:8:22 Cell detected on frequency = 806 , Cell ID = 235
14:8:22 Decoding BCCH-DL-SCH-Message ...
14:8:22 BCCH-DL-SCH-decode passed!

14:9:34 Scaning ... frequency = 816 Mhz
14:9:50 Cell detected on frequency = 816 , Cell ID = 116
14:9:50 Decoding BCCH-DL-SCH-Message ...
14:9:50 BCCH-DL-SCH-decode passed!
```

V References

(openphy), URL: https://github.com/ttsou/openphy
(FFTW), URL: http://fftw.org
(Ettus-UHD), URL: http://uhd.ettus.com
(Nodejs), URL: https://nodejs.org/en/docs/
(Telekompedia), URL: https://telecompedia.net/sib1-in-lte/
(Mathworks), URL: https://de.mathworks.com/help/5g/ug/nr-cell-search-and-mib-and-sib1-recovery.html