Towards a new open-source 5G development framework: an introduction to free5GRAN

Aymeric de Javel, Jean-Sébastien Gomez, Philippe Martins, Jean Louis Rougier, Patrice Nivaggioli

Towards a new open-source 5G development framework: an introduction to *free5GRAN*

A. de Javel, J.S. Gomez, P. Martins, J.L. Rougier, P. Nivaggioli

***Abstract*—5G technology has been designed with flexibility and software reusability in mind. In this context, open-source developments are critical enablers for mastering software architecture and associated codes, which are of the utmost importance for all technology stakeholders (ranging from manufacturers to operators and service users). *free5GRAN* is a new open-source 5G development framework. It focuses on research and education. It provides a modular architecture and a set of well-documented APIs, which enable easy experiment reproduction. *free5GRAN* aims at easing the understanding of 5G standards and the development of new technology components. The global objective is to be as modular as GnuRadio project [1], but with full system implementation. It has not been designed with performances in mind, unlike other open-source projects such as srsLTE [2] and OpenAirInterface [3]. The current release implements a significant part of the PHY layer on the receiver side, and the MAC layer will be part of a future release. Its architecture, which is composed of a library and an implementation part, is described. Moreover, RAN components are explained, and custom implementation methods are provided. It has been tested and validated against a commercial Amarisoft 5G SA gNodeB in a Faraday Cage.**

***Index Terms*—Software Defined Radio, 5G, Open-source**

I. Introduction

The last generation of mobile networks - 5G NR - introduces many new technologies, from the core network to Radio Access Network (RAN). Those technologies are key enablers for various 5G use cases such as industry 4.0, augmented reality, and smart cities. Virtualization is the fundamental building block for such new technologies. On the core network side, virtualization methods are mature as they mostly come from the data center and cloud world. On the other side, virtualization must be adapted to new RAN constraints (like network slicing and RAN splitting).

In this context, 5G RAN research and education frameworks are required. Such frameworks should enable one to deeply understand the technology by clearly exposing the key components and clarifying the threshold between standard definitions and manufacturers' implementations. Moreover, such a research framework should also enable one to develop new RAN technologies, algorithms, and methods by having a modular architecture and providing APIs. Its modular architecture should also ease RAN virtualization.

In this paper, *free5GRAN*, an open-source 5G RAN framework, is introduced. It is designed to be an easy-to-understand, easy-to-use, and highly modular framework. It can be used both by qualified engineers and researchers for new technologies testing and developments and by beginners that will clearly go through the main components of a 5G RAN stack. For people willing to test and develop new RAN technologies, *free5GRAN* provides a modular architecture and a rich library, and existing code-base can easily be leveraged to build new projects and experiments. Moreover, for beginners willing to have a clear view of the RAN architecture and understanding the development trade-offs, *free5GRAN* exposes the key components of a RAN stack and provides an implementation of the system from scratch. Other open-source projects such as srsLTE [2] and OpenAirInterface [3] already exist but are mostly designed to deploy efficient testbeds. *free5GRAN* aims at developing a software and documentation framework to understand the 5G system, like the approach used in the Linux From Scratch project [4].

The current release focuses on the PHY layer and includes a receiver that can decode MIB, DCI and SIB1 data. It has been tested and validated against a commercial Amarisoft 5G SA gNodeB in a Faraday Cage. The transmitter side is currently under development and should be released soon. Moreover, a MAC layer is also expected to be developed. The library-based architecture enables anyone to build its project using *free5GRAN* library.

This paper is split into three parts. The first one introduces the project architecture, the second one details implemented components, and the last part describes some custom implementation choices that have been made for non-standardized functions.

II. Project Architecture

free5GRAN architecture is split into two parts. The first part is the library which contains common functions, and the second part is the implementation which contains custom code for the transmitter and the receiver. *free5GRAN* code and documentation are available online (https://github.com/free5G/free5GRAN). Figure 1 represents *free5GRAN* architecture.

A. Library

The library is a set of static functions and variables commonly used by both receiver and transmitter, or even by other similar projects. All the functions have been designed to be used independently. It is split into four parts, which are *phy*, *utils*, *variables* and *asn1c*.

A. de Javel, J.S. Gomez, P. Martins and J.L. Rougier are with the Institut Polytechnique de Paris (IP Paris), Télécom Paris, LTCI (Paris, France), contact e-mail: aymeric.dejavel@telecom-paris.fr

P. Nivaggioli is with Cisco (Paris, France).

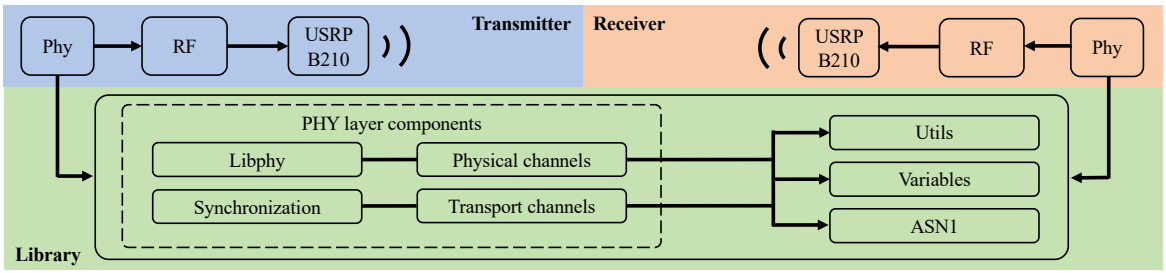


Fig. 1: free5GRAN architecture overview

1) phy: The *phy* library contains the code for PHY layer processing. It is split into four parts which represent different 5G RAN stack components. Those parts are *libphy* for signal processing functions, *physical_channel* for physical channels processing, *transport_channel* for transport channels processing and *synchronization* for time and frequency synchronization procedures. Main functions of the different libraries, that will be depicted later, are:

- *libphy*: channel estimation, demodulation and channel mapping/de-mapping.
- *physical_channel*: Physical Broadcast Channel (PBCH), Physical Downlink Control Channel (PDCCH) and Physical Downlink Shared Channel (PDSCH) decoding, defined in TS 38.211 Section 7.3 [5].
- *transport_channel*: Broadcast Channel (BCH), Downlink Control Information (DCI) and Downlink Shared Channel (DL-SCH) decoding, and all related sub-functions, defined in TS 38.212 Section 7 [6].
- *synchronization*: Primary and Secondary Synchronization Signals (PSS and SSS) correlation for time synchronization.

2) utils: it contains functions that are called at different levels of the code. Main functions are:

- Scrambling: takes an input bits sequence (hard or soft bits) and scramble it with another bits sequence. This is used for both physical and transport channel;
- Synchronization sequences generation: generates PSS and SSS sequences for receiver time synchronization, as defined in TS 38.211 Section 7.4.2 [5].
- PBCH, PDCCH and PDSCH DMRS sequences generation: generates pilot sequences for channel estimation, as defined in TS 38.211 Section 7.4.1 [5].
- Pseudo random sequence generation: generates a pseudo random sequence based on an initialization value c_{init}, as described in TS 38.211 Section 5.2 [5].

3) variables: it is a set of variables and data structures definitions used by both the library and implementation. Most important variables are:

- Global variables defined in the standard.
- Polar coding matrices: G_n matrices, $n \in [5, 10]$, for polar coding (defined in TS 38.212 Section 5.3.1.2 - [6]) and corresponding inverse matrices for polar decoding.
- Low Density Parity Check (LDPC) base graphs, defined in TS 38.212 Section 5.3.2 [6].

4) asn1c: it contains the code and structures for encoding and decoding RRC messages. It uses the ASN1C library [7]. In the current implementation, it is used by the receiver to decode SIB1 data. The library can currently parse and encode all the RRC PDU messages and types defined in TS 38.331 Section 6.2 [8].

B. Implementation

The implementation part provides code dedicated to the receiver. It is built using objects that store different processing information and exposes methods for receiver functionalities implementation. It is split into two main classes that are *rf* and *phy*.

1) rf: This class represents the RF device. It stores information about RF device configuration and current state and exposes methods for receiving and transmitting the signal from it. Future releases will include an interface at this level so that many different RF devices can be supported. In the current version, *free5GRAN* supports USRP B210 and uses USRP UHD library [9] for device exchanges.

2) phy: This class represents the receiver PHY layer. It stores different cell and PHY layer status information and exposes cell synchronisation methods, physical channel extraction and data decoding.

III. Receiver PHY layer components

The current release of *free5GRAN* includes a receiver that can decode MIB, DCI and SIB1 data from a cell. This includes a major part of all the 5G PHY layer processing. We plan to release the gNodeB side soon, transmitting MIB, DCI and SIB1 data. In this Section, we introduce the different components of a receiver PHY layer from scratch. Figure 2 represents the processing steps on both the transmitter and the receiver sides for the PHY layer.

A. Time and frequency synchronization

1) Time-domain synchronization: Time-domain synchronization is based on PSS and SSS correlation. Those two physical signals are part of Synchronization Signal Block (SSB), comprising PBCH and PBCH DMRS. PSS and SSS are generated based on $N_{ID}^{(2)}$ and $N_{ID}^{(1)}$. Time-domain synchronization consists of cross-correlating all possible PSS sequences

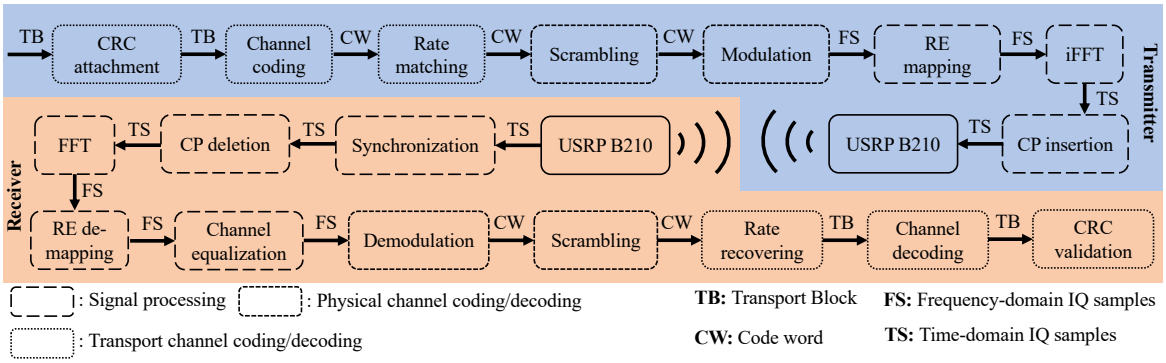


Fig. 2: PHY layer processing steps

with received frame. Highest correlation peak gives $N_{ID}^{(2)}$ and the time where the input signal contains PSS, which provides time synchronization. SSS sequences are then correlated with received signal to deduce $N_{ID}^{(1)}$ and receiver can compute Physical Cell ID, written N_{ID}^{cell}:

$$N_{ID}^{cell} = 3 \cdot N_{ID}^{(1)} + N_{ID}^{(2)}$$

2) Frequency-domain synchronization: Fine frequency synchronization has to be performed to mitigate Doppler effects. First, SSB symbols cyclic prefixes are correlated with corresponding symbols part. The phase offset on each symbol is derived as being the phase of the previous correlation. Finally, the global phase offset is computed as the mean of symbols phase offsets, and frequency correction is computed.

B. Physical channels extraction

Once synchronized, the receiver has to extract physical channels samples from the received time-domain signal. This is done by removing Cyclic Prefixes (CP), performing FFT (for recovering RE grid), and finally de-mapping channels to extract samples from the RE grid. Physical channels extraction differs depending on the studied channel:

1) PBCH: The receiver deduces PBCH symbols position from PSS position. PBCH and PBCH DMRS samples extraction depends on cell PCI (Further details can be found in TS 38.211 Section 7.3.3.3 and 7.4.1.4.2 - [5]).

2) PDCCH: The receiver performs blind search as PDCCH time and frequency position are not known. MIB data, decoded from PBCH, gives information about CORESET0, which is a set of frequency and time positions (detailed in TS 38.213 Section 13 - [10]) where PDCCH could be located. The receiver has to search over all the possible positions. Each candidate can be validated by CRC computation. For each candidate, the receiver computes PDCCH and PDCCH DMRS positions, as detailed in TS 38.211 Section 7.3.2.5 and 7.4.1.3.2 [5].

3) PDSCH: The receiver extracts PDSCH based on DCI, decoded from PDCCH. It gives time and frequency position of PDSCH (as explained in TS 38.214 Section 5.1 - [11]). PDSCH DMRS positions vary depending on PDSCH resources allocation.

All the three signals can be found within a single radio frame (10ms), as shown in Figure 3.

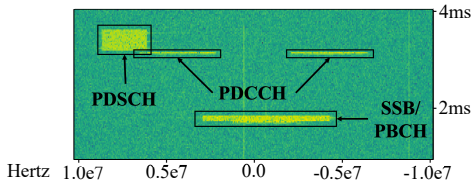
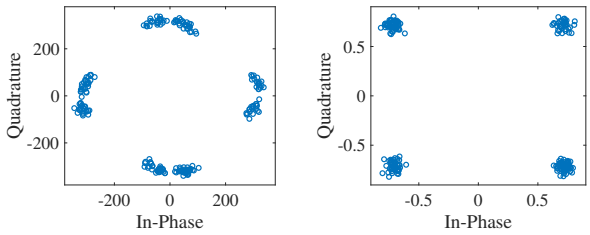


Fig. 3: Radio frame waterfall diagram



(a) Before (b) After

Fig. 4: Samples constellation before and after Channel equalization for PDCCH samples

C. Channel estimation and equalization

Over the air channel, the signal is modified by interference and noise, and cannot be decoded as-is. Equalization aims at correcting the received signal in order to recover the sent signal before decoding. This is done by estimating channel modifications for each RE grid point, based on pilot signals reception, and by applying inverse modification to received samples. Figure 4 shows PDCCH samples before and after channel equalization.

D. Physical channel decoding

Physical channel decoding is a set of functions that enables the receiver to recover transport channel bits. Depending on the physical channel, the decoding process may vary. It always includes demodulation and bits scrambling.

- Demodulation: process to recover bits from samples. It can be either hard or soft demodulation. Hard demodulation gives a value (0 or 1) to each bit, whereas soft demodulation gives a probability for one bit to be equal to 1 or 0. In our project, PBCH and PDCCH decoding

perform hard demodulation, whereas PDSCH performs soft demodulation. Only BPSK and QPSK demodulation are currently supported.
- Scrambling: invert bits values. Inversion sequence is defined by PCI for cells confusion avoidance.

For PDCCH, RE Groups (REG) are interleaved (as described in TS 38.211 Section 7.3.2.2 - [5]) and the receiver has to de-interleave received RE. This is done at the extraction level and not at the physical channel processing level in our implementation.

For PDSCH, layer mapping, antenna port mapping and VRB to PRB mapping are performed, as explained in TS 38.211 Section 7.3.1 [5].

E. Transport channel decoding

For MIB, DCI and SIB1 data decoding, the receiver has to decode BCH, DCI and DL-SCH. The overall decoding process consists of rate recovering, channel decoding and CRC validation. DL-SCH decoding includes code block segmentation and concatenation, but this is not currently implemented as SIB1 data is contained in a single code block.

1) Rate recovering: Rate recovering is the inverse function of rate matching. Rate matching consists of adapting the channel coding output length (called N) to the physical channel bits sequence length (called E). Channel coding output length depends on the transport channel and is defined by the standard. When $E > N$, rate matching is a circular buffer of size N, where input data is repeated until output length is E. When $E < N$, input data is punctured to reduce input sequence length. Rate matching is defined in TS 38.212 Section 5.4 [6].

2) Channel decoding: Channel decoding is the inverse function of channel coding. 5G NR defines two methods for channel coding: Polar coding and LDPC.

- Polar coding: used for BCH and DCI, it consists of determining the most reliable bits positions in a given channel and placing the input bits in those positions. In our current implementation, polar decoding is performed by inverting the coding technique. No correction is applied to received bits. Polar coding procedure is detailed in TS 38.212 Section 5.3.1 [6].
- LDPC: used for data channel (DL-SCH), it consists of computing parity bits and adding them at the end of the input bit sequence. In our current implementation, we use belief propagation algorithm for LDPC decoding and bits correction. The LDPC decoding algorithm is detailed later. LDPC coding procedure is detailed in TS 38.212 Section 5.3.2 [6].

3) CRC validation: Cyclic Redundancy Checks (CRC) is applied at the beginning of the transport channel coding process. After computation, CRC is added to the end of the input bits sequence. The receiver can validate the integrity of a received bits sequence by validating CRC.

IV. Non standard defined functions

Different components of the stack are not standardized, and manufacturers have to use custom implementation methods. Channel estimation, soft demodulation and channel decoding are part of those components in the receiver PHY layer. In this Section, used algorithms and methods are detailed. Those methods were chosen for their low computational complexity.

A. Channel estimation

Channel estimation is a vast research topic, and different methods are proposed. Some of those methods are theoretical and cannot be implemented in concrete systems. Some other methods have high computational complexities. Least Square Estimator (LSE) [12] is implemented in the current release which does not take into account channel noise. This method was chosen because it gives excellent results in our Faraday cage. Noise mitigating estimation methods are planned to be added in future releases.

Channel estimation aims at computing, for each RE, a channel coefficient that represents channel perturbation effects. Correction, called channel equalization, can be obtained by $X = H \cdot Y$ where Y are received samples and H are channel coefficients.

LSE consists of analysing pilots channel coefficients and propagating values by linear interpolation for non-pilot RE. Different pilots, called DMRS, are used according to the type of the physical channel. The first step is to compute pilots values. Then, channel coefficients are computed, for each i, j (symbol and sub-carrier index) in pilots position indexes:

$$h_{i,j} = \frac{x_{i,j}}{y_{i,j}}$$

Linear interpolation is performed in the frequency domain first and then, if needed, in the time domain for recovering channel coefficients at every RE grid position.

B. Soft demodulation

Demodulation is the process by which each received sample is converted to a sequence of bits. Hard demodulation consists of assigning a direct value (0 or 1) to each bit. On the other side, soft demodulation consists of estimating the probability for each bit to be equal to 0 or 1. It is useful for bits correction (that happens in transport channel processing). Different techniques have been proposed, using different concepts such as machine learning [13]. The method used in *free5GRAN* is a heuristic method which consists in computing the distance from received samples to different possible positions.

Each bit is studied independently. In the case of QPSK, there are two bits per sample. For each bit i, minimal distance d_i^0 and d_i^1 from received sample (green point in Figure 5) to each possible value of the bit (0 and 1, orange points in Figure 5) is computed. The difference between the minimal distance to 1 and minimal distance to 0, noted v_i is returned as bit value. The bit value is negative if the bit is more likely to be 1 and positive otherwise.

$$v_i = d_i^1 - d_i^0$$

The probability p_i for a bit to be equal to 0 or 1 is given by:

$$p_i(1) = \frac{1}{1 + e^{2 \cdot v_i}}, \quad p_i(0) = 1 - p_i(1) \tag{1}$$

Figure 5 shows bit 0 distances computation.

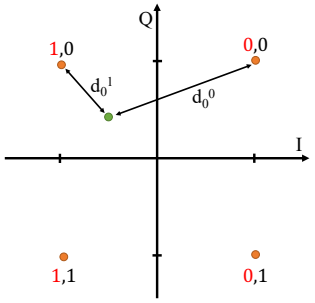


Fig. 5: d_0^0 and d_0^1 representation for bit 0 estimation in a QPSK constellation

C. LDPC decoding

LDPC decoding consists of correcting bits values depending on received bits sequence v and parity check bits. There are several algorithms for LDPC decoding, among which Belief Propagation has been chosen for *free5GRAN*.

LDPC decoding is based on a parity check matrix H, which is given by TS 38.212 Section 5.3.2 [6]. H is a sparse matrix composed of 1 and 0. H contains N columns, where N is the size of v. Before beginning the algorithm, different sets called C_i and R_j have to be filled. For each column i, C_i is computed as being the set of rows j where $H_{i,j} = 1$. For each row j, R_j is computed as being the set of columns i where $H_{i,j} = 1$.

For each bit i, $q_i(0)$ and $q_i(1)$ are computed:

$$q_i(k) = p_i(k) \cdot \prod_{j \in C_i} r_{j,i}(k)$$

Where $p_i(k)$ is given by Equation 1 and $r_{j,i}(k)$ is given by:

$$r_{j,i}(0) = \frac{1}{2} + \frac{1}{2} \cdot \prod_{l \in R_j \setminus \{i\}} (1 - 2 \cdot p_l(1))$$

And

$$r_{j,i}(1) = 1 - r_{j,i}(0)$$

New bit value probability p'_i can be recovered by normalizing $q_i(0)$ and $q_i(1)$:

$$p'_i(0) = \frac{q_i(0)}{q_i(0) + q_i(1)}, \quad p'_i(1) = \frac{q_i(1)}{q_i(0) + q_i(1)}$$

Finally, bit i updated soft value v'_i is:

$$v'_i = \alpha_k \cdot \frac{1}{2} \cdot \log_2 \left(\frac{1}{p'_i(k)} - 1 \right)$$

Where $k = 0$ if $p'_i(0) > p'_i(1)$ and 1 otherwise and $\alpha_0 = -1, \alpha_1 = 1$.

After having corrected and updated all the bits soft values, the above process can be repeated until $H \cdot v = 0$ or until maximum iterations number is reached. Bits can be corrected in parallel for better performances.

V. Conclusion

In this paper, *free5GRAN* was introduced as an open-source 5G SDR RAN framework. It has been described as a modular platform for one willing to understand what is behind RAN stack black box and for qualified people willing to develop new components. Main PHY layer components were detailed and related to 3GPP standards. Moreover, custom implementation methods where discussed. Code and documentation are available online (https://github.com/free5G/free5GRAN), so one can quickly begin experiments.

Further developments will focus on three points. First is the transmitter side, which is expected to be released soon. Second is MAC layer development. Finally, project architecture will be evolving for interoperability and modularity increase.

VI. Acknowledgements

The work presented in this article benefited from the support of NewNet@Paris, Cisco's Chair "Networks for the Future" at Télécom Paris (https://newnet.telecom-paristech.fr). Any opinions, findings or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of partners of the Chair.

References

[1] GNU Radio Website. [Online]. Available: https://www.gnuradio.org
[2] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "Srslte: An open-source platform for lte evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, ser. WiNTECH '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 25–32.
[3] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 33–38, Oct. 2014.
[4] Linux From Scratch Website. [Online]. Available: http://www.linuxfromscratch.org
[5] 3GPP, "Technical Specification Group Radio Access Network; NR; Physical channels and modulation," 3GPP, TS 38.211 V15.2.0, 06 2018.
[6] ——, "Technical Specification Group Radio Access Network; NR; Multiplexing and channel coding," 3GPP, TS 38.212 V15.2.0, 06 2018.
[7] L. Walkin, "Open source asn1c."
[8] 3GPP, "Technical Specification Group Radio Access Network; NR; Radio Resource Control (RRC); Protocol specification," 3GPP, TS 38.331 V15.2.0, 06 2018.
[9] USRP UHD Library Website. [Online]. Available: https://files.ettus.com/manual/index.html
[10] 3GPP, "Technical Specification Group Radio Access Network; NR; Physical layer procedures for control," 3GPP, TS 38.213 V15.2.0, 06 2018.
[11] ——, "Technical Specification Group Radio Access Network; NR; Physical layer procedures for data," 3GPP, TS 38.214 V15.2.0, 06 2018.
[12] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "Channel estimation techniques based on pilot arrangement in ofdm systems," *IEEE Transactions on broadcasting*, vol. 48, no. 3, pp. 223–229, 2002.
[13] O. Shental and J. Hoydis, "" machine llrning": Learning to softly demodulate," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–7.