COMMUNICATION SYSTEMS ENGINEERING WITH GNU RADIO

A HANDS-ON APPROACH

JEAN-MICHEL FRIEDT • HERVÉ BOEGLEN

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

with website

WILEY

Communication Systems Engineering with GNU Radio

Communication Systems Engineering with GNU Radio

A Hands-on Approach

Jean-Michel Friedt
University of Besançon
France

Hervé Boeglen
University of Poitiers
France

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data applied for:
Hardback ISBN: 9781394218882

Cover Design: Wiley
Cover Images: Courtesy of Jean-Michel Friedt, © GNU Radio logo - Wikimedia Commons/public domain

Set in 9.5/12.5pt STIXTwoText by Straive, Chennai, India

To Jean-Michel for his unfailling enthusiasm and for convincing me to join the world of freedom (Linux).

Contents

About the Authors

Jean-Michel Friedt was trained as a physicist at École Normale Supérieure in Lyon (France). He completed his PhD on scanning probe microscopy in 2000 before joining IMEC (Leuven, Belgium) as a postdoctoral researcher working on surface acoustic wave (SAW)-based biosensors. He joined the company SENSeOR in 2006 as a systems engineer, developing short-range RADAR systems for probing SAW resonators acting as wireless passive cooperative targets with sensing capability. Before becoming associate professor at Franche-Comté University in Besançon (France) in 2014 with his research activities hosted by the Time & Frequency department of the FEMTO-ST Institute, he became intrigued by the field at the intersection of computer science, radio frequency, and digital signal processing with access to the physical properties of the electromagnetic waves, namely software-defined radio (SDR), and its opensource implementation GNU Radio. Visiting the radio-silent research station of Ny-Ålesund (Norway, see cover) had the most profound impact on his personal and research and development activities, from satellite communication to remote sensing during field trips since 2007. He has been a regular contributor to the French GNU/Linux Magazine/France and related journals since 2005 whose publications motivated most of this research, with an emphasis to present results toward the general public and curious readers willing to reproduce experiments with affordable and readily available hardware and opensource software. Current investigations focus on the use of SDR for time and frequency distribution including GNSS (with anti-jamming and spoofing strategies), RADAR and spectrum spreading for time dissemination, remote sensing, and spaceborne communications.

Hervé Boeglen graduated from the University of Haute Alsace in Mulhouse, France with an MSc degree in electrical engineering in 1994. He worked as a full-time lecturer in electronics in the telecommunications and networks department at the IUT of Colmar, University of Haute Alsace, France, from 1995 to 2013. Between 2006 and 2008, while working, he pursued a PhD in digital communications. In 2013, he joined the University of Poitiers, France, as an associate professor in electrical engineering. He currently teaches graduate-level courses in embedded systems and digital communications using GNU Radio. He is also a member of the XLIM lab at the Futuroscope site in France. His research focuses on wireless channel modeling and digital communication systems, both radio and optical, using software-defined radio. He is also a radio transmission enthusiast and holds an amateur radio license (callsign F4JET).

Foreword

Communications Engineering solves one of the central problems of mankind: making sure that what is known in one place becomes known in another. This is a book instructing the reader how to implement that, wirelessly, with naught but a computer, free software, and a radio frontend.

Sufficiently obvious is the need for education in the practicalities of this communication technology, given how transformative they have been: The geopolitical situation of the twenty-first century is hard to imagine without the TV; neither the Arab Spring nor the public perception of Russia's invasion of Ukraine would have assumed their shape without ubiquitous mass communication. However, this is not a guide on how to make a call using a cell phone, or how to upload a video to social media.

Instead, its purpose lies in enabling the reader to work on a more fundamental, the *physical* layer, themselves, with nothing between them and the radio waves that need to be modified to communicate information. The power of *Software-Defined Radio* (SDR) lies in its ability to make the mathematical description of the physical phenomenon that is radio available for analysis and manipulation in a computer, and thus gives its users the ability to control what and how information is transported to the fullest extent.

Conversely, understanding how communication is actually done using SDR allows for a deeper insight into the nature of wireless communications. As the presence of a chapter on passive and active radar shows, the same techniques enabling us to exchange information with a communication partner allow us to retrieve information about things far away. This serves to illustrate one of the strengths of teaching concepts from communications engineering based on working, practical implementations: The theory taught on one page to establish the working of an aspect of communications serves as an explanation for the technology taught a few pages further down.

The authors elected to use GNU Radio to teach these concepts, why? GNU Radio is *Free and Open Source Software* (FOSS). This means three things for its usage as an educational tool:

1. Its openness allows for introspection: An interested user could always look inside and discover "how it's done."
2. Its wide availability across platforms, free of cost, makes it a desirable platform to work on, while not sacrificing on its professionality: The very same tools used by researchers, companies, and hobbyists worldwide are at the fingertips of the learner, giving them the opportunity to grow continuously from a beginner to an expert in designing communication systems.
3. The community built around GNU Radio is largely motivated by the idea that everyone should have access to both the knowledge and the tools needed to build communication systems. This leads to intense sharing of knowledge, both in the shape of software and its source code, as well as of methodology and theory.

Surprisingly to me, as former leader of the software and its development process, openness is *not* the strongest argument (but still important to many) for using GNU Radio to teach communications engineering.

More valuable is the community arising from the FOSS nature. GNU Radio understands itself not only as a software project, but as the unifying part of a whole ecosystem of SDR applications, libraries, research groups, and users. This has made it the most popular open source environment for SDR development, with an annual conference in the United States and in Europe, a pervasive presence in research articles, the amateur radio community, demonstrators, and common use case for the vendors of SDR frontends. This comes with a network of educators, users, developers, supporters, and learners. But GNU Radio did not start out as a large project. However, its function as tool to understand wireless communications is part of the very foundations of it, which is a short look back as its history shows:

When Eric Blossom, a computer-science-friendly electrical engineer in the United States was designing cryptographically secure telephones around 1993, he found commercial and official understanding of secrecy to be lacking. The freshly evolving digital wireless, cellular communication standards, in the shape of IS-95, had serious cryptographical flaws.

Considering the technological and financial entry hurdles of working with the waveforms on the air, Blossom talked to John Gilmore, of fame for sponsoring the Electronic Frontier Foundation's (EFF) efforts to build a demonstrator to prove the Data Encryption Standard (DES) encryption standard to be weak, the two enter into a patronage, where Blossom gets to be paid on a Free and Open Source SDR framework – leading to the birth of the GNU Radio idea in 1998; the early code was based on MIT's Pspectra SDR framework.

The progression from a small project to the most popular SDR framework in existence was fostered by the Moving Picture Association of America throwing in their weight when the United States moved analog TV to the digital Advanced Television Systems Committee (ATSC) – and enforcing a "copy-protection bit," to be respected by video recorders. That not sitting well with EFF ideals of free access to technology, a reason emerged to write a complex receiver in what would become GNU Radio – something that can receive ATSC TV, get the video, and not care at all about any copyright bits.

The EFF proving the ATSC copy-protection is a publicity success, but a project of that size showed the need to overhaul the code. Clearing this milestone, Blossom convinces Gilmore to let him rewrite GNU Radio from scratch to remove its limitations resulting from Pspectra legacy.

At this point, about 2003, Matt Ettus gets involved and starts building his SDR frontend – what he coined the "Universal Software Radio Peripheral" (USRP) – a device that attaches to PCs through USB, allowing anyone to work with electromagnetic spectrum with off-the-shelf, relatively affordable hardware. Ettus becomes a contributor to GNU Radio, and hard- and software co-evolve. Early versions of the driver for the hardware are part of GNU Radio; only later on, a more generally useful driver for the Ettus hardware is written. Academic interest is massively picking up – numerous PhD students work with GNU Radio, contribute code, and most importantly: They (and the students they supervise) form a lively community. At the same time, the project becomes commercially relevant enough to support consultants.

One of these PhD students is Thomas Rondeau – who later becomes the lead to replace Blossom; the consultant, to become the maintainer of GNU Radio, is Jonathan Corgan, both who shape the project into a software project that is good at accepting contributions. At the same time, community events start to emerge. Students – the author of this foreword not being an exception – get highly invested. A very active mailing list forms the glue of an international community of users and developers.

Things go smoothly, and GNU Radio 3.7 becomes the stable release found in nearly all Linux distributions; there's binary installers for Windows, the annual GNU Radio conferences attract more than 300 people a year. Development moves from a self-hosted git server with trac as project management to github and mediawiki for documentation. However, stability is a double-edged sword; it means contributions begin to dry up in development branches that stand little chance to actually get deployed to users. As Corgan leaves the position of maintainer, GNU Radio has excellent support on all relevant desktop operating systems, but the "next" branch, which was destined to become GNU Radio 3.8, is not in a shape immediately ready for release.

This is when I'm asked to step in and switch from a very unofficial role, where I try to stay atop of what is happening on the mailing list, explain the code, the communications engineering base and the occasional bug to users, to a more official role; from 2018 to 2021, I become architect and maintainer of GNU Radio. What an honor! Getting the 3.8 release out of the door with the help of a lot of friends, we gave GNU Radio a new velocity (and, as you can imagine, we broke some poor people's applications in the process of making sure GNU Radio still works on machines with Python 3). Releases become more regular, and the number of contributions surges strongly.

Handing my responsibility off to two people – Josh Morman as the new architect, and Jeff Long as release maintainer concerned especially with the stable releases – was excellent for the project, too. It allows the evolution of the developer code base with less worries about things breaking on the machines of users, without sacrificing on the ability to spin reliable new releases.

However, it goes without saying that a software framework having an easy 20 years of development history comes with some baggage. Not all things are as intuitive, or as fast, or as safe, or as consistent as they could be, even in GNU Radio 3.10, the current release series as of writing. The GNU Radio project continues to evolve; what I believe will stay the same is the dedication to one central principle:

Through offering a very accessible FOSS framework for SDR application, with which everybody can get access to this great shared resource, the electromagnetic spectrum, GNU Radio will continue to foster, and live off, an active community of coders, researchers, hackers, users, and operators that drive the project forward.

I hope this book motivates generations of people willing to learn to tackle something as rich in facets as communications engineering – using tools that allow and encourage them to go beyond what is available as what companies are willing to sell them as wireless devices.

June 2024 *Marcus Müller*

Acknowledgments

This book would not exist without … the GNU Radio development team. Over the years, numerous contributors have improved the software, sometimes with challenging decision on the software architecture (e.g. SWIG to PyBIND transition), but always to achieve utmost quality and performance. Next to the development, the opensource spirit has driven not only software development but also sharing the complex field of digital signal processing. By tackling practical applications, GNU Radio provides a fun and attractive framework for becoming familiar with such obscure concepts as an imaginary voltage or a negative frequency.

In this vein, the GNU Radio related conferences have been the opportunity for developers not only to share results as is usual in "scientific" conferences, but most importantly the means to achieve and reproduce these results. The American GNU Radio Conference with its proceedings available at https://pubs.gnuradio.org and YouTube channel at https://www.youtube.com/@GNURadioProject, the European Free Open-Source DEveloper Meeting (FOSDEM) software defined radio devroom and the European GNU Radio Days – with its YouTube channel at https://www.youtube.com/@europeangnuradiodays1445/ that both authors have helped co-organize since 2018 – have been the source of inspiration and motivation with technical discussions with speakers and attendees.

Jean-Michel Friedt and Hervé Boeglen

Acronyms

ACARS	**a**ircraft **c**ommunication **a**ddressing and **r**eporting **s**ystem
ADC	analog-to-digital converter
AGC	automatic gain control
BPSK	binary phase shift keying $\varphi \in \{0, \pi\}$
CCSDS	consultative committee for space data systems
CDMA	code division multiple access used in particular for identifying which GPS satellite is broadcasting
CGRAN	comprehensive GNU Radio archive network at https://cgran.org
COTS	commercial off the shelf
CRC	cyclic redundance check
DAB	digital audio broadcasting
DAC	digital-to-analog converter
FDMA	frequency division multiple access
FFT	fast Fourier transform an $N \log_2(N)$ complexity implementation of the Fourier transform
FM	frequency modulation
FSK	frequency-shift keying (FM digital modulation)
FSPL	free space propagation loss, the logarithmic expression of Friis energy conservation
GNU	GNU is not Unix
GPS	global positioning system
GRAVES	**G**rand **R**éseau **A**dapté à la **VE**ille **S**patiale is the French space surveillance RADAR emitting a continuous wave at 143.05 MHz
IF	intermediate frequency
IQ	in-phase/quadrature
ISI	inter-symbol interference
ISS	International Space Station whose amateur service is broadcasting on 145.8 MHz
LEO	low Earth orbit
LO	local oscillator
LOS	line of sight
MEO	medium Earth orbit
NCO	numerically controlled oscillator
OFDM	orthogonal frequency division multiplexing
PDU	Protocol Data Unit
PMT	polymorphic types

POCSAG	**P**ost **O**ffice **C**ode **S**tandardisation **A**dvisory **G**roup pager protocol for emergency services
QPSK	quad phase shift keying $\varphi \in \{0, \pi/2, \pi, 3\pi/2\}$
RADAR	radio detection and ranging
RDS	radio data system
RRC	root raised cosine
RF	radio frequency
RTL-SDR	a set of low-cost SDR receivers based on a RF front end and the Realtek RTL2832U analog-to-digital converter to USB
SDR	software-defined radio
SNR	signal-to-noise ratio
TED	timing error detector
WBFM	wideband frequency modulation the modulation scheme used by commercial FM broadcasters

About the Companion Website

This book is accompanied by a companion website:

www.wiley.com/go/friedtcommunication

The website https://gitlab.xlim.fr/gnuradio_book/flowcharts mirrored at https://gitlab.com/gnuradio_book/flowcharts includes:

- the flowcharts, included in the book as static figures, to be executed with GNU Radio 3.10 for assessing the output of the signal processing chains and tuning the parameters to observe the impact on the output signals. All figures are included in the folder with the name of the associated chapter for easy matching. Specific information needed to perform some of the experiments is also provided.

Introduction

What is GNU Radio? GNU Radio is a toolkit providing the means to address discrete-time digital signal processing chains oriented toward radio frequency (RF) communication, but not limited to it. GNU Radio is *not* a readily functional decoding software for a given communication protocol: understanding the principles of signal representation, frequency transposition, synchronization, and digital information extraction will be needed before implementing functional communication systems. Thanks to its flexibility, GNU Radio is not restricted to digital communication over radio frequency channels but can be used for instrumentation, RADAR and radio frequency channel characterization, time and frequency transfer, beamforming and null steering for e.g. jamming and spoofing suppression, or any application benefiting from accessing the raw radio frequency wave characteristics.

To make the learning curve less steep, a graphical interface for assembling signal processing blocks is provided: GNU Radio Companion. It should be emphasized, though, that the graphical interface is for development ease only and is not needed for executing the resulting flowgraph; hence, GNU Radio is perfectly suited for running on embedded systems not fitted with graphical interfaces. Indeed, GNU Radio is included in the embedded Linux-built frameworks, Buildroot and OpenEmbedded, allowing to use the Python-generated processing scripts on headless embedded systems running the operating system and the associated C++ libraries.

The reader is encouraged to test all processing sequences and assembling processing chains step by step: even though some of the examples are a lengthy sequence of processing steps, assessing the impact of each block by displaying the frequency domain or time domain characteristics after each processing step is mandatory. In order to help the reader test various processing sequences, records of relevant signals are made available on https://iqengine.org in the GNU Radio repository.

A word of caution before starting to experiment with software-defined radio (SDR) and storing huge files: make sure to remember the experimental setup leading to these records, and most significantly the carrier frequency, the sampling rate, and data format. Many times have these authors run days of records to forget after a few days how data had been collected and hence how to read them for post-processing. To avoid such hassle, the SigMF (signal metadata format) standard has been proposed, specified at https://github.com/sigmf/SigMF. This format associates with each data file (`sigmf-data`), a format description (`sigmf-meta`) which provides the receiver characteristics including carrier frequency, bandwidth (sampling rate), or data format (floating point or integer, and integer size of each sample). All records at https://iqengine.org comply with the SigMF format, hence providing the necessary information for processing the collected data.

The GNU Radio Companion processing sequences described throughout the book are available at https://gitlab.xlim.fr/gnuradio_book and also mirrored at https://gitlab.com/gnuradio_book, and each chapter starts by referring to the relevant IQEngine record. For optimal layout of the proposed

flowcharts, it will be assumed that GNU Radio Companion is configured (`View` menu) with the `Show parameter value in block`. All flowcharts have been tested with the 3.10 version of GNU Radio and GNU Radio Companion.

This discussion on using GNU Radio aims at a balance between processing synthetic signals and live signals collected from hardware. Indeed, the philosophy of SDR is to minimize the specificity of hardware and move most processing steps to software.

The book is organized as follows. The first chapter introduces GNU Radio and GNU Radio Companion as tools for becoming familiar, through simulations, with basic concepts needed when processing radio frequency signals, including the manipulation of complex numbers and baseband versus radio frequency bands. The second chapter extends the processing to real signals collected from hardware, with records available to readers for reproducing the processing steps if such signals are not available at their location or if the relevant hardware is not available. The third chapter tackles the communication between GNU Radio and external tools, either through network sockets or filesystems, introducing concepts needed in the following chapters. The fourth chapter benefits from all these knowledge to demonstrate how to assemble various SDR RADAR architectures, whether passive or active, and how accessing the raw radio frequency samples allows for target range and velocity detection as well as azimuth when combined with a moving antenna for spatial diversity of the sources in the synthetic aperture RADAR implementation. The fifth chapter returns to some basic concepts of GNU Radio for synchronizing processing tasks and propagating tags marking some features detected in the processed signal. The newly acquired knowledge is used in the sixth chapter to develop a complete digital communication system. The seventh chapter extends the custom digital communication system to decoding all layers of a satellite communication system using the same underlying modulation scheme: being a low-earth orbiting (LEO) satellite flying every day over every area in the world, the signal is accessible to all readers irrelevant of their geographical setting. While spaceborne communication benefits from ideal propagation conditions in free space, ground-based communication is plagued with multipath interferences and fading, an issue tackled in the eighth chapter using orthogonal frequency division multiplexing (OFDM) as implemented in the digital audio broadcast standard. Finally, the ninth chapter develops how the open-source GNU Radio framework can be complemented with custom processing blocks written in Python or C++, with an emphasis on custom source blocks for adding new hardware to the processing chain or new sinks for implementing processing or decoding protocols not yet supported by the standard GNU Radio processing blocks.

1

Getting Started with GNU Radio: Synthetic Signals

This first chapter aims at achieving three outcomes; introducing the general concepts of software-defined radio (SDR) and how to reduce to a minimum the hardware dependence of the processing to move all digital signal processing steps after the analog-to-digital conversion; justifying the handling of complex numbers with a real and imaginary by GNU Radio; and becoming familiar with the GNU Radio Companion graphical user interface (GUI). These goals will be reached by using GNU Radio to process synthetic signals so that no hardware is needed to complete this first chapter. All GNU Radio Companion flowgraphs presented in this and the subsequent chapters are available from the GitHub repository at https://gitlab.xlim.fr/gnuradio_book, also mirrored at https://gitlab.com/gnuradio_book. When opening these flowcharts, it is assumed that `View →Show parameter expressions in block` and `Show parameter value in block` as well as `Show Block comments` are active for best layout experience.

1.1 Evolution of Radio Frequency Electronics Toward SDR

The term "software radio" was coined by J. Mitola in the early 1990s [Mitola, 1993]. The main idea of this technology is to reduce the analog electronics of a radio receiver to the part near the antenna, namely the radio frequency (RF) front end (RF amplifier + filters), and use a high-speed analog-to-digital converter (ADC) to get the baseband signal in the digital domain. All the usual operations (demodulation, filtering, etc.) are then performed digitally with the help of a digital signal processor (DSP). This would represent the ultimate SDR solution which is achievable today, though still quite expensive, for example, AMD-Xilinx Zynq Ultrascale+ RFSoC field programmable gate arrays (FPGAs) [AMD-Xilinx, 2024]. In this case, 5.9 GSps ADCs and 10 GSps digital to analog converters (DACs) are available, allowing to directly sample signals with a carrier frequency up to about 3 GHz. It is widely acknowledged that in telecommunications, a carrier frequency is employed to transmit information from one point to another. The carrier itself does not convey information; thus, it is removed at the receiving end. This is achieved by multiplying the signal received from the antenna by a local replica of the carrier. Consequently, this process results in a baseband spectrum containing the transmitted information, centered around 0 Hz. In the not-so-distant past, these operations were carried out using analog electronic components. In the case of the ultimate SDR system, the concept is to execute all demodulation and decoding functions in the digital domain. To gain a comprehensive understanding of the SDR hardware to be utilized in Chapter 2 (i.e. Adalm-Pluto and RTL-SDR), it is essential to grasp the fundamental concepts that have driven modern SDR architectures, particularly those constructed around the **zero intermediate frequency** structure (zero-IF or ZIF). Let us concentrate on the receiving side. It is widely recognized that the heterodyne

Communication Systems Engineering with GNU Radio: A Hands-on Approach,
First Edition. Jean-Michel Friedt and Hervé Boeglen.

Companion website: www.wiley.com/go/friedtcommunication

structure is the most commonly used in analog receivers. Therefore, to achieve efficient reception of various channels, the demodulation stage operates at a fixed **intermediate frequency** (IF), typically 10.7 MHz for the frequency modulation (FM) broadcast band. All the electronic parts of this section are optimized for this fixed IF frequency (amplifiers, filters, etc.). The frequency translation of the signal coming from the antenna (e.g. 88–108 MHz for FM broadcast) is performed by a mixer and a tunable local oscillator structure. In the case of SDR, having an IF is not desirable as our aim is to process only the baseband signal. Therefore, the direct-conversion or ZIF architecture is what we seek. Although this structure has been known for quite some time, it was not until the 1990s, with advancements in integrated electronics, that it became a viable option. Let us now illustrate this evolution with two different SDR hardware platforms that we will be utilizing in Chapter 2. The first one, whose structure is illustrated in Figure 1.1, is the RTL-SDR receiver, initially designed for digital TV reception. It adopts a so-called zero second IF structure and uses an analog front end, which down-converts the RF signal to a user-defined IF frequency. This IF signal is then sampled by an 8-bit ADC at 28.8 MHz. Interestingly, the ADC is followed by a two-channel structure called an **IQ (in-phase and quadrature)** demodulator. The IQ structure is a fundamental constant found in any SDR system, and we will come back to it shortly.

The second hardware example is the Adalm-Pluto from Analog Devices Inc. (ADI) [Analog Devices, 2024] (Figure 1.2). It integrates a radio chip, which is a complete 2×2 transceiver (AD9363, see Figure 1.3), working on carrier frequencies between 325 MHz and 3.8 GHz and an instantaneous bandwidth of 56 MHz. This is a complete ZIF transceiver and one of the first efficient implementations of this apparently simple structure. For completeness, it is important to specify to the reader that the ZIF structure is not without its issues, which partly explains why its introduction is relatively recent. These issues include local oscillator (LO) leakage, DC offsets, and IQ mismatch, which may require compensation during a relatively complex calibration stage. Interested readers can refer to Razavi's book [Razavi, 2012], which delves into this subject in great detail.

1.2 The Complex Envelope and the Justification of the IQ Structure

The fundamentals of signal processing teach us that the Fourier transform of a real-valued signal is complex symmetric. This implies that in the frequency domain, negative frequency components emerge redundantly alongside the positive frequencies. However, negative frequencies do not exist in the physical domain; this phenomenon can be considered as an annoying mathematical artifact that often disrupts most students! It is, however, possible to build a time signal known as an **analytic signal**, whose Fourier transform exclusively reveals positive frequencies. This analytic signal, being complex valued, encompasses the original real-valued signal. The concept of the analytic signal originates from the contributions of Gabor [1946], Ville [1948, 1958], as stated by Viswanathan [2017]. Interested readers can explore classical signal processing literature for a more comprehensive understanding. The application of the analytic signal holds particular significance in numerous signal processing contexts, especially within the software-defined radio (SDR) domain. The theory concerning the analytic signal is intricately connected to the concept of the **complex envelope**, a well-established notion within the telecommunication community. For further insight into constructing the complex envelope of a bandpass signal $s(t)$ from its analytical representation, readers are encouraged to consult [Guimaraes, 2020]. Let us now consider a modulated **bandpass signal** $s(t)$ having a center frequency of f_c. It can be shown [Guimaraes, 2020] that

$$s(t) = Re\left[\tilde{s}(t)\exp(j2\pi f_c t)\right] \tag{1.1}$$

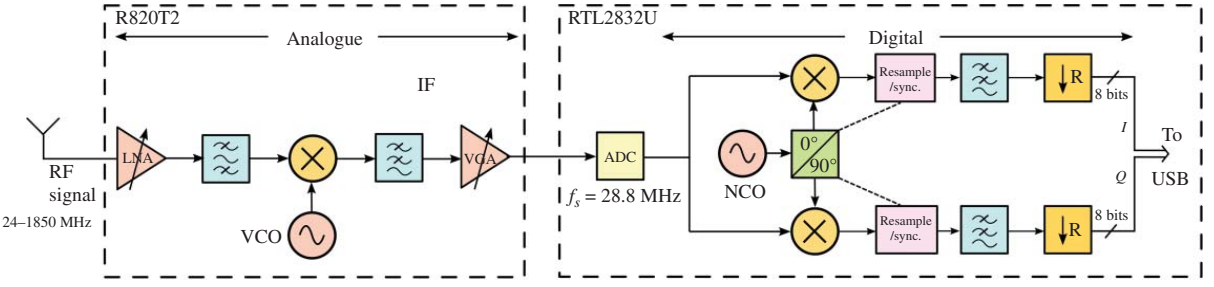

Figure 1.1 Block diagram depicting the RTL-SDR receiver, comprising two essential chips: the R820T2 and the RTL2832U.

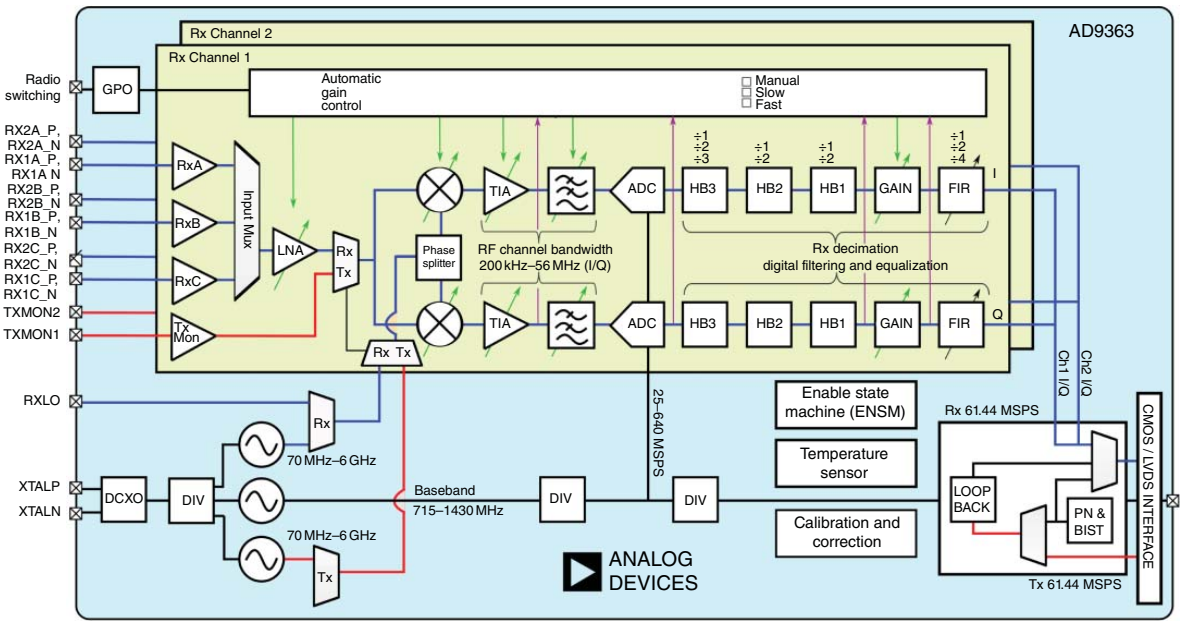

Figure 1.2 ADI AD9363 IC receiver section. This radio chip is integrated into the Adalm-Pluto device.