Current State of Hardware and Tooling for SDR

Nico Rumsch, Leander Seidlitz*, Jonas Andre*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: nico.rumsch@tum.de, seidlitz@net.in.tum.de, andre@net.in.tum.de

***Abstract*—Software Defined Radios have become increasingly important because of their unique feature to support a wide multitude of frequencies, modulation modes, amplitudes and waveforms which makes a single device useful for a variety of applications. In the following, the history of SDR and the current state in terms of use cases, applications and hardware will be presented. Furthermore, a comparison of GNU Radio, Matlab and SDR# regarding their support for hardware and applications will be made. Lastly, a small overview of the usage of Field Programmable Gate Arrays to improve the performance of Software Defined Radios is given.**

***Index Terms*—software-defined radio, history, hardware, software, fpga, gnu radio**

1. Introduction

In the 1970s a need for more easily configurable radios started to arise. Before the introduction of Software Defined Radios (SDRs), it was always necessary to implement the ability to, for example, receive different frequencies by using more hardware components. With SDR a hardware and software platform was created to solve this problem. Its possibility to be controlled by software to receive and transmit different radio signals without requiring modifications to the hardware itself makes it a versatile tool for any radio hobbyist or researcher. This holds especially true for cellular network research where hardware can be used to simulate different networks. In an ideal scenario, a single hardware device can be configured by software to transmit or receive any imaginable frequency, or waveform, at any data rate. However, current hardware offers have physical limits and operate in specific boundaries like frequency ranges. [1]

1.1. History

The first step towards the SDRs known today was done by Joe Mitola when he defined the term Software Radio (SR) in 1992 as a system that consists of a Radio Frequency (RF)-frontend, Analog Digital Converter (ADC) and Digital Signal Processor (DSP). His proposed architecture alleviated the hardware responsibility of decoding the signal and moves it to a dynamically configurable DSP which thereby can easily support different waveforms or frequencies amongst others. [2] The term "Software Defined Radio" was later introduced by Stephen Blust in 1995 [3]. Before the term existed, already in 1984 E-Systems Inc. implemented the first SDR in 1984 [4]. Four years later, in 1988, researchers of the Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), the predecessor of today's Deutsches Zentrum für Luft- und Raumfahrt (DLR), developed the first SDR transceiver which could be configured through software as part of a digital satellite modem [5], [6]. Later in the 1990s, the first large-scale application of a SDR platform was deployed by the US military, called SpeakEASY 1 and the next generation SpeakEASY 2 [7], [8]. After 2000, SDRs matured to a point where no significant change to the concept occurred. All newer developments are in improved performance, smaller chip sizes, lower power consumption and better affordability. For hobbyists, a range of offers for cheap SDR receivers emerged, while researchers benefit from better performance and a wider range of applications.

1.2. Functionality of SDRs

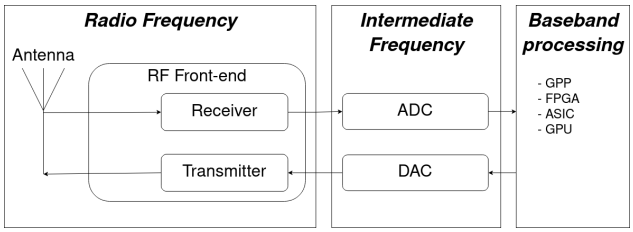

Figure 1: Example SDR architecture [9]

Figure 1 describes a common architecture for a SDR with the three main components being a radio frequency frontend, a converter between digital and analog signals and a processor for interpreting or producing the digital values. The RF-frontend is responsible for receiving the radio wave from the transmission medium. This incoming signal is sampled twice with one sample being phase shifted by 90 degrees which is called I/Q sampling and simplifies the hardware requirements for the SDR. [10]

I/Q sampling works on the premise that any waveform can be reconstructed by adding the amplitude of a sine and a cosine wave [10]. This gives an unambiguous waveform, which is not possible to achieve with the multiplication of in-phase waves [11]. Furthermore, representing a wave with the amplitude I and Q is easier than working with the amplitude and phase of a wave. The final formula to represent a wave can be seen in Equation 1. Here, frequency is abbreviated by f, total wave amplitude by

 doi: 10.2313/NET-2023-11-1_19

A, time by z, phase by φ, In-phase amplitude by I, and Quadrature amplitude by Q. [12]

$$A \cdot \cos(2\pi f t + \varphi) = (\sqrt{I^2 + Q^2}) \cos(2\pi f t + \arctan(\frac{Q}{I})) \tag{1}$$

In the formula, the cos component is called the carrier because it is the base wave onto which the information will be encoded. Changing the I and Q amplitude of the sine and cosine wave is called modulating and encodes said information. [10]

The two sampled amplitudes are converted by the ADC to discrete digital values or created from digital values in case of transmission by the Digital Analog Converter (DAC). When sampling a frequency it is necessary to sample at double the wanted frequency, called the Nyquist frequency, to fully capture the characteristics of it. For example, in the case of Bluetooth's and WiFi's $2.4\,\mathrm{GHz}$ signals, it is required to sample at 4.8 GSps which in turn can only be achieved with expensive ADCs/DACs. To solve this problem, the incoming signal is converted into an intermittent frequency before it is digitalized. This allows for the removal of the carrier from the frequency, which creates a signal for the ADC/DAC that is centered around $0\,\mathrm{Hz}$. This is known as the baseband. [10]

The final I/Q values from the baseband signal can then be used by the processor, which may be for example a General Purpose Processor (GPP), a Field Programmable Gate Array (FPGA), a Graphics Processing Unit (GPU), or an Application-Specific Integrated Circuit (ASIC) to decode the signal using available software.

2. State of the art

With advances in the hardware and software field, modern SDR systems can, amongst others, operate on a wider range of frequencies, and support more modulation types and waveforms. In the last two decades, many new systems were created that benefit from this dynamic usage of hardware. One of the most prominent examples is the development of cellular network standards (3G, 4G, 5G) [3]. Especially for 4G and 5G, the possibility for a software-defined base station enables faster development and updates to the mobile network with only software changes [13].

2.1. SDR use cases

This section goes more into detail about cellular networks and describes the usage of SDRs in amateur radio.

2.1.1. Cellular networks and wireless communication. Since the 4th generation of cellular networks, Software Defined Networks are playing a more important role by allowing for easy reconfigurability and dynamic deployments. This however does not yet extend to the physical layer of the cell towers, where current research is proposing to integrate SDRs as the missing building block. [13]–[15]

In contrast, researchers and developers already adopted SDRs for the field of cellular networks or wireless communication which can be seen in Table 5, where two of the three presented tools support the simulation of cellular networks based on SDRs.

2.1.2. Amateur radio. SDRs play an important role in amateur radio. Anyone can receive a wide range of frequencies and modulations, which makes SDRs popular among hobbyists. This interest is further increased by the option for affordable hardware, like the RTL-SDR family.

One popular area, where many enthusiasts use SDRs, is in the context of aircraft positional data (Automatic Dependent Surveillance - Broadcast (ADS-B)). It is a public, worldwide system where almost every commercial aircraft broadcasts unencrypted details about itself, amongst other information its position, on the $1090\,\mathrm{MHz}$ frequency. Many community-based projects use local, terrestrial SDRs to receive the data and send them to a message broker, which combines all the received data and provides a (public) dataset or live Application Programming Interface (API). [16]–[19]

Other example use cases are RF fingerprinting, spectrum analysis, drone detection or decoding in general [20]. For a selection of further protocols which can be freely received see the subsequent Section 2.2.

2.2. Protocols

The benefit of SDRs is the support for many different applications and protocols via one single device. They can be used by all computers or laptops, as the devices are, in most cases, accessible via the Universal Serial Bus (USB) or network. A selection of applications and their corresponding frequency ranges can be found in Table 1.

2.3. SDR hardware

In Table 3 ten SDRs are presented with their hardware specification. Compared are the available processor types for which a more detailed comparison can be found in Table 2. A SDR with an onboard FPGAs has the potential to move some of the program logic onto the same for better performance, as described in Chapter 3.2. This makes an onboard FPGAs a key feature to consider. Furthermore, it is indicated if the hardware is a receiver, transmitter or transceiver and in which configuration it can be operated, primarily half-duplex, full-duplex or both. For the ADC and DAC the sampling rate and bit depth are listed where it is better to have higher values in each category. The bit depth indicates how precise a signal can be received or reconstructed and the sampling rate of how often this conversion can happen per second. The same applies to the overall sampling rate, which might differ between the ADCs and DAC. This can be caused by slower performance in, for example, the communication interface onboard processor. The frequency spectrum indicates which frequency signals can be received or transmitted. A wide frequency range, which goes into the upper and lower bound extremes, is better than an SDRs with a more narrow range. The bandwidth, as can be seen in figure 2, specifies which surrounding frequency range can be observed around the tuned-to frequency of the SDR.

 doi: 10.2313/NET-2023-11-1_19

TABLE 1: Protocols or applications in frequency ranges [21], [22]

Frequency range	Application	Protocol
30-300 kHz	Navigation	
300 kHz-3 MHz	Marine/Aircraft navigation, AM broadcast	
3-30 MHz	Broadcasting, mobile radio	NFC/EMV, Automatic Identification System (AIS)
30-300 MHz	FM radio broadcast	
300 MHz-1 GHz	Cell phones, mobile radio, Internet of Things (IoT), TV	LoRa, SIGFOX, ZIGBEE, Z-Wave, DVB-T2
1-3 GHz	WLAN, Cell phones, IoT	ZIGBEE, ANT+, WIFI, LoRa, Bluetooth, ADS-B
3-60 GHz	Radar, Cell phones	WIFI(6)

TABLE 2: Technologies for signal processing on SDR [21], [23]

Type	Performance	Power	Size
GPP	low/medium	low	medium
DSP	medium	medium	large
FPGA/SoC	high	medium	large
ASIC	high	low	small

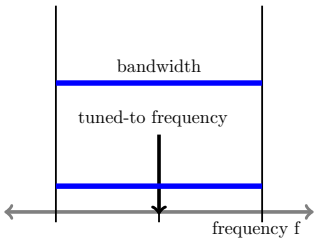

Figure 2: Explanation: Bandwidth

2.3.1. Receiver. In the hobbyist space receivers are popular because of their affordability with a multitude of platforms, such as RTLSDR and Airspy, being available. The typical frequency range of these systems is between 1 MHz and 2 GHz with varying resolutions and amounts of DACs/ADCs. Devices supporting this range can therefore already receive more than half of the applications and protocols mentioned in Table 1. The number of samples per second increased in the past to about 200 MSps for expensive systems, with some more exotic products being able to achieve rates in the range of GS/s by using a high-performance FPGA for the processing, combined with high-performance ADCs and DACs. [21]

2.3.2. Transceiver. Transceivers are devices that can both send and receive data. Generally speaking, the transmitter part of an SDR transceiver is either equally or less powerful than the receiving part. Compared to a device, which can only receive, the receivers on a transmitter are more powerful than their receive-only counterparts. This includes, amongst other, support for larger frequency ranges, higher bandwidths, higher samples per second and better resolutions of the ADCs/DACs. [21] Following the same trend, professional-grade transceivers have the same benefit over consumer hardware with microcontrollers, custom System on a Chip (SoC), FPGAs or even GPUs (AIR-T [36]) for very high throughputs. Popular consumer transceivers are from HackRF and LimeSDR, while in the professional space devices from USRP are popular.

3. Software support for SDR protocols and hardware

Over the years, hardware and software for different use cases of SDRs got developed. Software for SDRs connects to the hardware at the baseband processing step (see Figure 1). The following section will present a selection of software, their support for the previously presented hardware, and support for different use cases.

3.1. Software

The available software for SDRs can range from simple command line tools to graphical user interfaces, sometimes with support for protocol-specific visualizations. There are over 30 universal and even more single-purpose tools available for the popular RTL-SDR platform alone. [37]

3.1.1. GNU Radio. One of the most popular tools is GNU Radio originally released by Eric Blossom in 2001 as an official GNU project. It is being continuously developed and uses the concept of flowgraphs to define block-based transformations. While all processing operations are implemented in C++, the definition of the flowgraphs can be written in either C++ or Python. GNU Radio is supported on Linux, Windows, and MacOS. [38]–[40]

3.1.2. Matlab and Simulink. Matlab and Simulink are proprietary software products by MathWorks. The suite supports a wide range of applications from linear algebra and numeric computing to complex simulations. [41] Amongst others, it offers functionality to simulate wireless networks directly on hardware. For this, the software can interface with a wide range of platforms and communicates with the digital processing unit of the SDR. Matlab is supported on Linux, Windows, and MacOS. [42]

3.1.3. SDR#. SDR# is a simple-to-use, general-purpose visualization tool for SDRs, running on Windows only. It visualizes real-time readings of the frequency and spectrum from the SDR. [43] Furthermore, it has a rich plugin system to enable support for more protocols and SDR applications. [44] Hardware-wise it natively only supports the Airspy platform but is extended by official or community-developed plugins and can interface with a wide variety of SDRs. [45]

3.2. Hardware

The support of GNU Radio, Matlab and SDR# for hardware devices, as presented in Table 3, is indicated in

 doi: 10.2313/NET-2023-11-1_19

TABLE 3: Comparison of selected SDR hardware

Hardware	Chipset	Processor Type	RF Frontend	Receiver/Transmitter	Duplex	ADC/DAC resolution [Bits]	ADC/DAC sampling rate [MSps]	Sampling rate [MSps]	Frequency range [MHz]	Bandwidth [MHz]	Interface
RTL-SDR [24]	R820T2	n/a	n/a	1/-	n/a	8/-	n/a	28.8	0.5 – 1766	2.4	USB
Airspy R2 [25]	R860	GPP	35dBm IIP3	1/-	n/a	12/-	20/-	10	24 – 1700	9	USB
Airspy Mini [26]	R860	n/a	35dBm IIP3	1/-	n/a	12/-	36/-	10	24 – 1700	6	USB
LimeSDR [27]–[29]	LMS7002M	FPGA	n/a	2/2	full	12/12	n/a	61.44	0.1 – 3800	61.44	USB
LimeSDR PCIe [28]–[30]	LMS7002M	FPGA	n/a	2/2	full	12/12	n/a	61.44	0.1 – 3800	61.44	PCIe
HackRF One [21], [31]	MAX2837	GPP,FPGA	n/a	1/1	half	8/10	20/20	20	1 – 6000	20	USB
USRP B200 [32]	AD9364	FPGA	20dBm IIP3	1/1	both	12/12	61.44/61.44	61.44	70 – 6000	56	USB
USRP B210 [33]	AD9361	FPGA	20dBm IIP3	2/2	both	12/12	61.44/61.44	61.44	70 – 6000	56	USB
USRP N320 [34]	n/a	GPP,FPGA	17dBm IIP3	2/2	both	14/16	250/250	250	3 – 6000	200	Ethernet
Per Vices Cyan [35]	n/a	GPP,FPGA	n/a	1-16/1-16	both	16/16	1000/1000	1000	<18000	1000	Ethernet

Table 4. A unique feature of the USRP devices is, that all of them can be controlled with the common interface library USRP Hardware Driver (UHD) [46]. This means, developers can support all devices from this vendor by implementing support for the UHD interface.

While GNU Radio supports all in Table 3 listed devices, both Matlab and SDR# only support a subset of them. Generally speaking, Matlab is focussing more on professional-grade products, in this case from USRP/Ettus Research, and SDR# supports hardware targeted towards enthusiasts.

TABLE 4: Software support for presented hardware [21], [47], [48]

	GNU Radio	Matlab	SDR#
RTL-SDR	Yes	Yes	Yes
Airspy R2	Yes	No	Yes
Airspy Mini	Yes	No	Yes
LimeSDR	Yes	No	Yes
LimeSDR PCIe	Yes	No	Yes
HackRF One	Yes	No	Yes
USRP B200	Yes	Yes	No
USRP B210	Yes	Yes	No
USRP N320	Yes	Yes	No
Per Vices Cyan	Yes	Yes	No

A recent development to achieve even higher performance in SDRs is to utilize FPGAs and implement processing logic in hardware. Three approaches are possible to utilize FPGAs for processing:

1) Build a SDR out of a RF-frontend and FPGA
2) Additional FPGA as an accelerator
3) Utilize SDRs built-in FPGAs

Following the first approach, an implementation of the IEEE 802.11 standard and ZigBee is described in [49]. In contrast, the authors of [50] propose a framework to utilize a FPGA as an accelerator in combination with GNU Radio and show promising results. Lastly, in [8] the authors describe the communication with the host as one of the major limitations of high performance SDRs, which follows the third approach. The authors were able to achieve an up to 64 times higher data rate by moving the demodulation of the signal onto the FPGA and thereby eliminating most communication overhead. The interaction with the FPGA could be performed by a utility from UHD.

Besides higher data rates, another benefit of using FPGAs is faster development times compared to ASICs. However FPGAs also require larger board sizes as more additional hardware is needed. [21], [51]

3.3. Protocol support

As most transmitted data is encoded, software is typically required to decode them and provide specific visualizations, e.g. showing data on a map in the case of ADS-B or AIS.

The support for the in Section 3.1 presented software and the previously presented applications are described in Table 5. Because SDR# itself mainly supports the visualization of the baseband data, it relies on extensions to support applications beyond the build functionality and does not support most wireless communication protocols. Matlab on the other hand has tools for most communication protocols. Lastly, GNU Radio supports a wide range of applications because of its flow graph design and big community support.

In Table 5, support for a protocol will be annotated with "Yes", if additional software is required a "*" will be added and no information available will be indicated by "n/a".

TABLE 5: Protocol support of SDR software [52]–[67]

	GNU Radio	Matlab	SDR#
WIFI	Yes	Yes	n/a
ADS-B	Yes	Yes	Yes*
FM radio broadcast	Yes	Yes	Yes
LoRa	Yes	n/a	n/a
ZigBee	Yes	n/a	n/a
AIS	Yes	Yes	Yes*
3G	Yes	n/a	n/a
4G	Yes	Yes	n/a
5G	n/a	Yes	n/a
Bluetooth	Yes	Yes	n/a

For cellular networks, there also exists the Open Air Interface (OAI) developed by the OpenAirInterface Software Alliance. Its goal is to lower the adoption barrier for Radio Access Networks (RANs) by offering implementations for modern cellular network types like 4G and 5G. [68]

4. Conclusion

Software Defined Radio supports a wide variety of different use cases. It is especially dominant in cellular networks in its recent versions, to enable dynamic upgrades without requiring hardware changes. For enthusiasts, it allows an easy start in the field of radio networks, because a single device can be used versatilely in terms of frequency and application support, while still being affordable. Because of the same features SDRs are largely adopted in development and research to, for example, simulate cellular networks, WIFI or other protocols without requiring specific hardware.

The field is under continuous development to achieve the ideal SDR with its latest development being the inclusion of FPGAs in the SDR. While they have been used as an accelerator for quite a while now, using the SDR's

onboard FPGA is rather new and can greatly improve performance by reducing the communication volume between SDRs and hosts.

References

[1] T. Ulversoy, "Software Defined Radio: Challenges and Opportunities," vol. 12, no. 4, pp. 531–550.

[2] J. Mitola, "Software Radios-Survey, Critical Evaluation and Future Directions," in *[Proceedings] NTC-92: National Telesystems Conference*, pp. 13/15–13/23.

[3] R. Sahu, "Theoretical and Practical Approach to GNU Radio and LimeSDR Platform."

[4] M. T. Mushtaq, M. S. Khan, M. R. Naqvi, R. Khan, M. A. Khan, and O. Koudelka, "Cognitive Radios and Cognitive Networks: A short Introduction," 2013.

[5] R. G. Machado and A. M. Wyglinski, "Software-Defined Radio: Bridging the Analog–Digital Divide," vol. 103, no. 3, pp. 409–423.

[6] Peter Hoeher and Helmuth Lan, "Coded-8PSK Modem for Fixed and Mobile Satellite Services Based on DS," in *Coded-8psk Modem for Fixed and Mobile Satellite Services Based on DSP*, vol. January 1990, pp. 117–123.

[7] R. Lackey and D. Upmal, "Speakeasy: The Military Software Radio," vol. 33, no. 5, pp. 56–61.

[8] S. S. Hanna, A. A. El-Sherif, and M. Y. ElNainay, "Maximizing USRP N210 SDR Transfer Rate by Offloading Modulation to the On-Board FPGA," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 110–115.

[9] T. Juhana and S. Girianto, "An SDR-based Multistation FM Broadcasting Monitoring System," in *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*. IEEE, pp. 1–4, accessed 2023-05-29. [Online]. Available: http://ieeexplore.ieee.org/document/8272943/

[10] Dr. Marc Lichtman, "3. IQ Sampling — PySDR: A Guide to SDR and DSP using Python," accessed 2023-06-10. [Online]. Available: https://pysdr.org/content/sampling.html

[11] Mikael Q Kuisma, "I/Q Data for Dummies," 03/10/2023, 8:08:56 PM, accessed 2023-06-11. [Online]. Available: http://whiteboard.ping.se/SDR/IQ

[12] Dr. Marc Lichtman, "3. IQ Sampling — PySDR: A Guide to SDR and DSP using Python — Carrier Down Conversion," accessed 2023-06-10. [Online]. Available: https://pysdr.org/content/sampling.html#carrier-and-downconversion

[13] H.-H. Cho, C.-F. Lai, T. K. Shih, and H.-C. Chao, "Integration of SDR and SDN for 5G," vol. 2, pp. 1196–1204.

[14] D. Kafetzis, S. Vassilaras, G. Vardoulias, and I. Koutsopoulos, "Software-Defined Networking Meets Software-Defined Radio in Mobile ad hoc Networks: State of the Art and Future Directions," vol. 10, pp. 9989–10 014.

[15] F. Xu, H. Yao, C. Zhao, and C. Qiu, "Towards next Generation Software-Defined Radio Access Network–Architecture, Deployment, and Use Case," vol. 2016, no. 1, p. 264, accessed 2023-05-29. [Online]. Available: https://doi.org/10.1186/s13638-016-0762-6

[16] Flightradar24, "Live Flight Tracker - Real-Time Flight Tracker Map," Flightradar24, accessed 2023-05-31. [Online]. Available: https://www.flightradar24.com/

[17] "The OpenSky Network - Free ADS-B and Mode S Data for Research," accessed 2023-05-31. [Online]. Available: https://opensky-network.org/

[18] "Home - Serving the Flight Tracking Enthusiast," ADS-B Exchange, accessed 2023-05-31. [Online]. Available: https://www.dev.adsbexchange.com/

[19] "ADSBHub - Free ADS-B Data Exchange and Plane Tracking," accessed 2023-05-31. [Online]. Available: https://www.adsbhub.org/

[20] W. Jeong, J. Jung, Y. Wang, S. Wang, S. Yang, Q. Yan, Y. Yi, and S. M. Kim, "SDR Receiver Using Commodity Wifi via Physical-Layer Signal Reconstruction," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. Association for Computing Machinery, pp. 1–14, accessed 2023-06-04. [Online]. Available: https://dl.acm.org/doi/10.1145/3372224.3419189

[21] J. D. J. Rugeles Uribe, E. P. Guillen, and L. S. Cardoso, "A Technical Review of Wireless Security for the Internet of Things: Software Defined Radio Perspective," vol. 34, no. 7, pp. 4122–4134, accessed 2023-06-04. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1319157821000896

[22] L. Hui Fang, S. Hassan, M. AbdulMalek, M. Mazalan, S. Johari, N. Safari, and Y. Wahab, "Development of Microstrip Chebyshev Low Pass Filters using Laser Micromachining."

[23] F. Karray, M. W. Jmal, A. Garcia-Ortiz, M. Abid, and A. M. Obeid, "A Comprehensive Survey on Wireless Sensor Node Hardware Platforms," vol. 144, pp. 89–110, accessed 2023-06-05. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618302202

[24] "Buy RTL-SDR Dongles (RTL2832U)," rtl-sdr.com, accessed 2023-06-18. [Online]. Available: https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/

[25] "Airspy R2 - airspy.com," accessed 2023-06-14. [Online]. Available: https://airspy.com/airspy-r2/

[26] "Airspy Mini - airspy.com," accessed 2023-06-14. [Online]. Available: https://airspy.com/airspy-mini/

[27] "LimeSDR," Lime Microsystems, accessed 2023-06-14. [Online]. Available: https://limemicro.com/products/boards/limesdr/

[28] "LimeSDR Comparison," Crowd Supply, accessed 2023-06-14. [Online]. Available: https://www.crowdsupply.com/lime-micro/limesdr

[29] "LMS7002M Documentation," MyriadRF, accessed 2023-06-14. [Online]. Available: https://github.com/myriadrf/LMS7002M-docs

[30] "LimeSDR PCIe," Lime Microsystems, accessed 2023-06-14. [Online]. Available: https://limemicro.com/products/boards/limesdr-pcie/

[31] "HackRF One - Great Scott Gadgets," accessed 2023-06-14. [Online]. Available: https://greatscottgadgets.com/hackrf/one/

[32] E. R. Brand, a National Instruments, "USRP B200 USB Software Defined Radio (SDR)," Ettus Research, accessed 2023-06-18. [Online]. Available: https://www.ettus.com/all-products/ub200-kit/

[33] ——, "USRP B210 USB Software Defined Radio (SDR)," Ettus Research, accessed 2023-06-18. [Online]. Available: https://www.ettus.com/all-products/ub210-kit/

[34] ——, "USRP N320," Ettus Research, accessed 2023-06-18. [Online]. Available: https://www.ettus.com/all-products/usrp-n320/

[35] "Cyan – Per Vices," accessed 2023-06-18. [Online]. Available: https://www.pervices.com/cyan/

[36] "Artificial Intelligence Radio Transceiver (AIR-T)," Deepwave Digital, accessed 2023-06-05. [Online]. Available: https://deepwavedigital.com/hardware-products/sdr/

[37] "The BIG List of RTL-SDR Supported Software," rtl-sdr.com, accessed 2023-06-14. [Online]. Available: https://www.rtl-sdr.com/big-list-rtl-sdr-supported-software/

[38] "GNU Radio - The Free & Open Source Radio Ecosystem · GNU Radio," GNU Radio, accessed 2023-06-14. [Online]. Available: https://www.gnuradio.org/

[39] "GNU Radio," accessed 2023-06-14. [Online]. Available: https://en.wikipedia.org/w/index.php?title=GNU_Radio&oldid=1159673658

[40] A. Marwanto, M. A. Sarijari, N. Fisal, S. K. S. Yusof, and R. A. Rashid, "Experimental Study of OFDM Implementation Utilizing GNU Radio and USRP - SDR," in *2009 IEEE 9th Malaysia International Conference on Communications (MICC)*, pp. 132–135.

[41] "MATLAB," accessed 2023-06-14. [Online]. Available: https://en.wikipedia.org/w/index.php?title=MATLAB&oldid=1157953731

[42] "What Is Software-Defined Radio (SDR)?" accessed 2023-06-14. [Online]. Available: https://www.mathworks.com/discovery/sdr.html

[43] J. R. Machado-Fernández, "Software Defined Radio: Basic Principles and Applications," vol. 24, no. 38, pp. 79–96, accessed 2023-06-14. [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_abstract&pid=S0121-11292015000100007&lng=en&nrm=iso&tlng=en

[44] "List of SDRSharp Plugins," rtl-sdr.com, accessed 2023-06-14. [Online]. Available: https://www.rtl-sdr.com/sdrsharp-plugins/

[45] "SDR# and Airspy Downloads - airspy.com," accessed 2023-06-14. [Online]. Available: https://airspy.com/download/

[46] "USRP Hardware Driver (UHD™) Software," Ettus Research, accessed 2023-06-14. [Online]. Available: https://github.com/EttusResearch/uhd

[47] "USRP Support from Communications Toolbox," accessed 2023-06-18. [Online]. Available: https://www.mathworks.com/hardware-support/usrp.html

[48] "Airspy@groups.io | USRP / UHD support," accessed 2023-06-18. [Online]. Available: https://groups.io/g/airspy/topic/7621279

[49] A. Di Stefano, G. Fiscelli, and C. Giaconia, "An FPGA-Based Software Defined Radio Platform for the 2.4GHz ISM Band," in *2006 Ph.D. Research in Microelectronics and Electronics*, pp. 73–76.

[50] C. R. Irick, "Enhancing GNU Radio for Hardware Accelerated Radio Design," accessed 2023-06-12. [Online]. Available: https://vtechworks.lib.vt.edu/handle/10919/33474

[51] M. Petri and M. Ehrig, "A SoC-based SDR Platform for Ultra-High Data Rate Broadband Communication, Radar and Localization Systems," in *2019 Wireless Days (WD)*, Apr. 2019, pp. 1–4.

[52] T. Vilches and D. Dujovne, "GNUradio and 802.11: Performance Evaluation and Limitations," vol. 28, no. 5, pp. 27–31.

[53] cloud9477, "Gr-ieee80211," accessed 2023-06-16. [Online]. Available: https://github.com/cloud9477/gr-ieee80211

[54] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Decoding IEEE 802.11a/g/p OFDM in Software Using GNU radio," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, ser. MobiCom '13. Association for Computing Machinery, pp. 159–162, accessed 2023-06-16. [Online]. Available: https://doi.org/10.1145/2500423.2505300

[55] M. Hostetter, "Gr-adsb," accessed 2023-06-16. [Online]. Available: https://github.com/mhostetter/gr-adsb

[56] S. Meshram and N. Kolhare, "The Advent Software Defined Radio: FM Receiver with RTL SDR and GNU radio," in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 230–235.

[57] D. Valerio, "Open Source Software-Defined Radio: A Survey on GNUradio and its Applications," accessed 2023-06-16. [Online]. Available: https://www.semanticscholar.org/paper/Open-Source-Software-Defined-Radio%3A-A-survey-on-and-Valerio/90cdfd630dabf4ea75aea53bbc9c22ae2367e737

[58] B. Oumimoun, L. Nahiri, H. Idmouida, A. Addaim, Z. Guennoun, and K. Minaoui, "Software Defined AIS Receiver Implementation Based on RTL-SDR and GNU Radio," in *2022 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pp. 1–5.

[59] "GR-Bluetooth," Great Scott Gadgets, accessed 2023-06-16. [Online]. Available: https://github.com/greatscottgadgets/gr-bluetooth

[60] "WLAN Toolbox," accessed 2023-06-18. [Online]. Available: https://www.mathworks.com/products/wlan.html

[61] W. Alqwider, A. Dahal, and V. Marojevic, "Software Radio with MATLAB Toolbox for 5G NR Waveform Generation," in *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 430–433.

[62] "ADS-B and AIS - MATLAB & Simulink," accessed 2023-06-18. [Online]. Available: https://www.mathworks.com/help/comm/ads-b-and-ais.html

[63] "FM Broadcast Receiver - MATLAB & Simulink Example," accessed 2023-06-18. [Online]. Available: https://www.mathworks.com/help/supportpkg/rtlsdrradio/ug/fm-broadcast-receiver.html

[64] "Bluetooth LE Waveform Reception Using SDR - MATLAB & Simulink," accessed 2023-06-18. [Online]. Available: https://www.mathworks.com/help/bluetooth/ug/bluetooth-low-energy-receiver.html

[65] "ADSB# Plugin for SDRSharp," rtl-sdr.com, accessed 2023-06-18. [Online]. Available: https://www.rtl-sdr.com/adsb-plugin-for-sdrsharp/

[66] "Getting Started with RTL-SDR and SDR-Sharp and CubicSDR," Adafruit Learning System, accessed 2023-06-18. [Online]. Available: https://learn.adafruit.com/getting-started-with-rtl-sdr-and-sdr-sharp/sdr-number-fm-radio

[67] J. Demel, S. Koslowski, and F. K. Jondral, "A LTE Receiver Framework Using GNU Radio," *Journal of Signal Processing Systems*, vol. 78, no. 3, pp. 313–320, Mar. 2015.

[68] "OpenAirInterface – 5G Software Alliance for Democratising Wireless Innovation," accessed 2023-06-16. [Online]. Available: https://openairinterface.org/