A Low Cost Desktop Software Defined Radio Design Environment using MATLAB, Simulink and the RTL-SDR

R.W. Stewart, L. Crockett, D. Atkinson, K. Barlee, D. Crawford, I. Chalmers
Dept. of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, G1 1XW, UK

M. McLernon, E. Sozer
MathWorks Inc, Natick, MA 01760, USA

Submission to IEEE Communications Magazine
Special Section on "Software Defined Radio – 20 Years Later", 2015

Abstract

In the last five years, the availability of powerful DSP and communications design software, and the emergence of relatively affordable devices that receive and digitize RF signals, has brought SDR to the desktops of many communications engineers. However, the more recent availability of very low cost SDR devices such as the RTL-SDR, costing less than $20, has brought SDR to the home desktops of undergraduate and graduate students, as well as professional engineers and the maker communities. Since the release of the various open source drivers for the RTL-SDR, many in the digital communications community have used this device to scan the RF spectrum and digitize I/Q signals that are being transmitted in the range 25 MHz to 1.75 GHz. This wide operating range enables the sampling of frequency bands containing signals such as FM radio, ISM signals, GSM, 3G and LTE mobile radio, GPS, and so on. In this article we will describe the opportunity and operation of the RTL-SDR, and the development of a handson, open-courseware for SDR. These educational materials can be integrated into core curriculum undergraduate and graduate courses, and will greatly enhance the teaching of DSP and communications theory, principles, and applications. The lab and teaching materials have recently been used in senior (fourth year undergraduate) courses and are available as open course materials for all to access, use, and evolve.

Introduction

In this article we present our experience of developing university and CPD (continuous professional development) materials for teaching SDR in the form of DSP-enabled-radio systems. The availability of SDR receivers such as the RTLSDR, along with hardware support software drivers, means that we now have devices that are very low cost and can interface directly with MATLAB and Simulink software, allowing users to develop real software defined radio systems from the desktop. The RTL-SDR plug-in device (which comes with a simple but useable omnidirectional antenna) currently costs less than $20 (twenty dollars) to buy, and can be powered and connected via a USB port to Windows, Linux, and or Mac desktop computers. Students can acquire the MATLAB and Simulink student version (http://www.mathworks.com/academia/student_version/), along with the relevant DSP and Communications System Toolboxes, for around $100, and after the installation of appropriate support drivers, can be up and running with a complete SDR design environment. As we will summarize in this article, the opportunities for using this for education and learning are immense, and a whole new generation of engineers will see more and more RF and communications design being done as a coding task. The full set of SDR open-course educational materials referred to in this article are available to download (http://www.desktopSDR.com) as a 670 page workbook with more than 120 hands-on examples [1].

In this article we will first outline what the RTL-SDR is, "where" and how it evolved, and then introduce our open-source teaching and support materials for learning SDR from a DSPenabled- radio perspective. The SDR design environment and open-course materials described herein have the potential to be used in classes ranging from EE freshman (first year bachelor) university environments, for courses featuring applications such as spectral viewing and first experiences in radio, all the way to EE senior (fourth year bachelor) or masters level classes teaching, for example, the challenging aspects of QPSK receivers with synchronization, and other digital communications systems [2, 3]. Real practical experience in these communications applications and theory can be achieved using a low cost SDR receiver that students and home users can keep in their pocket, and connect via USB to run SDR algorithms directly on the desktop of their laptop device, and all having spent less than $20 on the RTL-SDR hardware!

SDR is now in the domain of being "homework" where students can work at home using their RTL-SDR and MATLAB and Simulink software. There is no longer a requirement for expensive spectrum analyzer hardware, and no requirement for signal generators; all that is needed is the ability to buy a USB RTL-SDR stick device on-line, download drivers, and then develop the appropriate SDR receivers and systems to find signals of interest that can be spectrally viewed, analyzed, and even decoded.

The Advent of Desktop SDR

Over the last 20 years the prospect of software (defined) radios has been greeted with enthusiasm by the DSP (digital signal processing), digital communications, and radio/RF sectors [4, 5]. In some ways the term software defined radio (SDR) has perhaps diverged in recent years to have different meanings for different engineering groups. Many in the DSP community considered that, by virtue of very high speed ADCs (analog to digital converters) and DACs (digital to analog converters), SDR was in effect the engineering of "DSP-enabled-radio systems," that is, where analog oscillators would be replaced with digital numerically controlled oscillators, analog filtering with digital filter chains, and phase locking components with digital phase locked loops (PLLs), and so on. Whereas in other communications system engineering domains, SDR actually refers to middleware, which is the software that could define the radio and provide the framework for the deployment of software objects over networks and between devices in the radio hardware [6, 7]. (SDR middleware would ultimately be the term to describe the control and design of high power computing platforms that would allow radio standards and waveforms to be switched in and out and downloaded on the fly, as pursued in applications such as JTRS (joint tactical radio service) from 1998 to 2011 [7, 8].)

In both closely related interpretations of SDR, i.e. DSP-enabled-radio and middleware, its concept and promise were easy to understand, but the hardware and software that was required 20 years ago was far beyond the then-affordable state of the art. But of course Moore's law never fails (or hasn't yet). Hence the reality and ease of access to SDR technologies is definitely here, both for the DSP-enabled-radio and the middleware groupings.

In the last five years or so, SDR in the DSPenabled-radio category has been achievable in the lab at a reasonable cost (less than $1500) for FPGA-enabled hardware with ADC and DAC units typically sampling at rates of a few hundred MHz, and front end radio cards that worked at up to 6 GHz, such as the ubiquitous USRP series from Ettus Research (http://www.ettus.com/product/category/USRP-Bus-Series). Similarly, various software platforms allow users to code and configure these devices using FPGA design environments that are often driven from DSP and communications development tools. SDR hardware products such as the URSP have been widely used to stream samples of down-converted RF signals to the desktop, where they are input to software such as MATLAB and Simulink for real-time processing or recorded for offline use. Additionally, where drivers were available and the desktop was a high speed computing platform, then real-time DSP software algorithms could be implemented (in floating point on the desktop processor) and receivers and transmitters implemented. Software defined radio is now established in a number of institutions as part of the curriculum, and in the May 2014 special SDR education feature topic in IEEE Communications Magazine, a number of authors reviewed their successful experiences introducing USRP-based SDR into their integrated course curricula and laboratory sessions for EE students [9–11]. However, the advent of the RTL-SDR device brings the affordability of a device down to a level lower than a textbook, and many of these successful courses can now also consider using the RTL-SDR as part of their laboratories, or as stand-alone learning assignments for students to do at home [12].

What is the RTL-SDR?

As shown in Fig. 1, the RTL-SDR is a small, compact, and easy-to-use USB stick device that is capable of receiving RF radio signals ("RTL" is actually not an acronym, but derives from the Realtek RTL2832U chip on which the device is based). Originally these devices were designed for use as DVB-T (digital video broadcast–terrestrial) receivers and featured custom-designed, tunable RF front end chips (e.g., the Rafael Micro R820T and the Elonics E4000) that allowed consumers to receive and watch UHF broadcast TV on their computers. In other words, these receivers were not originally designed or conceived to be used as generic programmable SDRs. The uptake of these devices as SDR receivers results from the efforts of a number of independent engineers and developers in the SDR community, who discovered their programmability for SDR. Specifically, it was found that the devices could be placed in a "test mode," which meant that the RTL2832U chip (Fig. 1c) bypassed the DVB decoding stage and produced raw, 8-bit I/Q data samples. Further, it was also found to be possible to program the center frequency of the RF chip over a working range of approximately 25 MHz to 1.75 GHz, and have some control over the data sampling rate. Soon after this discovery, the name RTLSDR was coined, which referred to the fact that the RTL (Realtek) based DVB receivers could be used as SDRs. With such a wide front end tuning frequency range, many different applications using various modulation schemes ranging from AM and FM, to ISM, GSM, LTE, and GPS applications, have become signals that we can attempt to capture. The noise floors, signal resolutions, and frequency accuracy of these devices is not optimal in some frequency bands, nor sufficient for some applications. However this investigation is all part of learning what the RTL-SDR offers, and it does work very well and successfully receives in many frequency bands and for a variety of applications.

As a quick review of the history of the RTLSDR's emergence, its origins were evident in some 2012 forum posts by a Linux developer on the V4L GMANE forum, stating that "radio sniffs" were possible using an RTL-based DVB device. It was discovered that when the device was tuned to receive FM and DAB radio stations, it was programmed into a different mode and that raw, modulated data samples could be transferred to the computer and demodulation performed in software (http://comments.gmane.org/gmane.linux.drivers.video-input-infrastructure/44461). Seventeen seconds of data originating from a Finnish radio station was captured and posted online, along with a query asking if anyone could work out how to demodulate it manually. This was accomplished only 36 hours later, after some collaborative effort. In the original post the last line is the optimistic statement, "I smell a very cheap poor man's software defined radio here :)"! This discovery led to further investigation of the RTLSDR's USB protocol. The commands transmitted when tuning to a radio station were captured, and used to force the device to stay in this special mode continuously. It turned out to be a test mode, and when the RTL2832U was in this mode, it output 8-bit unsigned samples of baseband I/Q data, rather than decoded DVB signals as per its designed operation. Work reported at the open source website Osmocom included reports from developers who had produced an independent SDR device called “OSMO-SDR,” and had experience in writing software that was able to program the DTV tuners used with the RTL2832U. After examining the Windows drivers provided by Realtek, they devised how to program the tuner via the demodulator, and the drivers for the RTLSDR were released to the open-source community (http://sdr.osmocom.org/trac/wiki/rtl-sdr).

RTL-SDR, as we now know it, came onto the market in early 2013, and various devices and software kits became available, produced by a number of companies and developers around the world. Judging by the communities on the web, the RTL-based DVB-T devices appear to be more popular as SDR receivers than they were for their original intended purpose of digital TV reception! NooElec is one company with worldwide distribution of these devices (http://www.nooelec.com/store/sdr.html). Based on their use of the R820T tuner, the NooElec RTL-SDR devices are capable of reliably sampling the frequency spectrum at a rate up to 2.8 MHz, and receiving signals in the RF frequency range 25 MHz to 1.75 GHz.

MathWorks released a hardware support package for the RTL-SDR in early 2014 (http://www.mathworks.com/hardware-support/rtl-sdr.html) which enables both MATLAB and Simulink to interface with and control the RTLSDR. With this support package, baseband samples output from the RTL-SDR device are supplied into the software environment, enabling users to implement any kind of DSP receiver or spectrum sensing system they desire as either a Simulink model or MATLAB code. I/Q data can be locally recorded to data disk files for later processing, or if processing power allows on the desktop computer, live demodulation and decoding can be performed.

Figure 2 shows a signal processing flow diagram of the main stages that are carried out on the RTL-SDR. RF signals entering the R820T tuner (on the right hand side) are downconverted to a low-IF (intermediate frequency) using a voltage controlled oscillator (VCO). This VCO is programmable, and is controlled by the RTL2832U over an I2C interface. After an active gain control (AGC) stage, the IF signal then needs to be brought down to baseband. The classical method of doing this is to pass the IF signal through an anti-alias filter, sample the output with an ADC, and then demodulate to baseband using quadrature NCOs (numerically controlled oscillators, i.e., a sine and a cosine oscillating at the IF frequency). Finally (on the left hand side of the diagram) the I/Q 8 bit samples are ready to stream to MATLAB or Simulink running on the desktop.

A Software Defined Radio Design Environment using RTL-SDR

With the capability to tune over the range of 25 MHz to 1.7 GHz, the RTL-SDR can be used to investigate, view the spectra of, and receive and decode a wide range of radio signals transmitted for various applications using different modulation methods. The actual signals available to a user will of course depend on their geographical location and the surrounding radio environment. To provide an example, from our location in central Glasgow (Scotland), we can receive, view, and variously analyze and decode a selection of RF signals including:

1.	FM radio stations	87.5 to 108MHz
2.	Aeronautical	108 to 117MHz
3.	Meteorological	117MHz
4.	Fixed mobile	140 to 150MHz

5.	Special events	174 to 217MHz
6.	Fixed mobile (space to earth)	267 to 272 MHz
7.	Fixed mobile (earth to space)	213 to 315 MHz
8.	ISM band (short range)	~433MHz
9.	Emergency services	450 to 470MHz
10.	UHF TV broadcasting	470 to 790MHz
11.	4G LTE and GSM	800 to 900MHz bands
12.	Short range devices	863 to 870MHz
13.	GPS systems	1227MHz to 1575MHz

The RTL-SDR Laboratory Environment

In Simulink, the RTL-SDR interface support takes the form of a library block, which represents both a source for the Simulink model, and a location to set parameters supplied to the RTL-SDR hardware device. As highlighted in Fig. 3, in the Simulink dialog window three main parameters are used to configure the device: the RF center frequency, fRF; tuner gain parameters, K; and the baseband sampling frequency, fs. In addition, a frequency correction parameter can be used to correct for offsets due to component tolerances and frequency drift which may affect the device. (In MATLAB these parameters can be set by initializing and configuring an RTLSDR System object.)

One of the first opportunities for SDR education with the RTL-SDR is simple spectrum viewing: finding and observing the frequency spectra (and in some cases the time domain representation) of some of the RF signals in the applications being broadcast around you. With knowledge (or guess work!) about available FM radio signals in the vicinity, an easy first example to run is to parameterize the RTL-SDR receiver interface block such that the device tunes to and demodulates a certain portion of the FM radio spectrum, and supplies the resulting baseband samples into the Simulink model for spectral viewing and or further processing to demodulate the signal [1]. The I/Q (complex) baseband sampling rate of the RTL-SDR has a recommended maximum of 2.8 MHz (the actual maximum is 3.2 MHz, although data loss occurs at this rate), and samples have an 8 bit resolution. Hence the maximum bandwidth can be considered to be 2.8 MHz. While this bandwidth is insufficient for viewing, for example, a 5 MHz band of UMTS 3G spectra, it is more than adequate for capturing and viewing a number of other signal types, including FM signals (bandwidth = 200 kHz), keyfob signalling centered at 433 MHz, exploring the 200 kHz wide GSM channels, and so on. And while it is not possible to see the full spectrum of, for example, a 10 MHz wide LTE signal, you can easily scan over the wider band in 2 MHz sections, and observe the guard bands and spectrum edges incrementally. The output of one of the spectral viewing LTE examples from the workbook is shown in Fig. 4.

To progress to more advanced and challenging examples in the teaching lab situation, we often need signals that are locally generated and controlled. Before transmitting RF signals, however, one must be very sure that transmission of a given signal power in a particular frequency band is legal, otherwise there is a danger of being an unwelcome jammer! Recognizing our desire for a low-cost teaching and learning setup, we can generate signals locally using devices such as FM transmitters that can be plugged into smartphone headphone sockets (these devices cost less than $20 and low-power versions are legal in many regions), or by acquiring singlechip devices such as the RT4 433 MHz device and building simple AM transmitter circuits [1].

To begin to teach more advanced digital communications using the RTL-SDR, we need to be able to generate appropriate RF signals in the lab, such as QPSK and other QAM transmissions. At the receive side, students can then design QPSK and QAM SDR receivers in MATLAB and/or Simulink, featuring numerically controlled oscillators, phase locked loops, frame synchronizers, digital receive filter chains, and other design elements. An example of such a design is shown in Fig. 5. To generate suitable signals in the laboratory we can use a programmable, transmit-capable SDR device such as the USRP, or Zynq SDR platform (featuring a Xilinx FPGA and Analog Devices FMComms card) to the class environment. In our Information, Transmission and Security seniors class (fourth year bachelor) at the University of Strathclyde, our final laboratory challenge session in the Winter/Spring 2015 semester was to decode and receive a multiplex of signals consisting of two AM, two FM, and two QPSK data channels. This multiplex was transmitted in a 2 MHz band on 602 MHz (the University of Strathclyde has a UK Government Ofcom UHF white space test licence at this frequency, and hence can legally use this in the lab for test purposes). If radio transmission over the air is not practical, perhaps due to local environmental or legal concerns, then an alternative is to use a cable and MCX connectors to make an RF cable connection between the transmitter and the RTL-SDR receiver, in place of the free-space wireless channel. Figure 6 shows one of the Strathclyde students at work in a seniors (fourth year Bachelor) lab, with just a PC, software, and the RTL-SDR and simple antenna supplied with the device.

A Hands-on SDR Communications Workbook

In the context of the curriculum requirements of our DSP and digital communications courses at the University of Strathclyde, the RTL-SDR has created the opportunity to invigorate our teaching and learning with real-world signals, reception, and examples. In response, we have jointly developed a complete workbook for EE students that allows them to experience and explore the radio spectrum, and to design, test, and implement radio receivers. A large selection of reference designs is provided with the workbook. MATLAB is now the de facto technical computing environment in many schools, including Strathclyde, and student familiarity with the software provides an excellent platform for developing a complete curriculum. Nevertheless, the course aims to appeal to all levels of prior experience, and includes sections for those new to the various tools and themes covered.

The open-source course materials and teaching and learning examples are all available online (http://www.desktopSDR.com) and feature 650 pages of practical exercises (starting from first principles), descriptions and theory, and more than 120 MATLAB and Simulink example files [1]. The materials are openly available, and also likely to be of use to practicing professional engineers, the maker community, or perhaps amateur radio enthusiasts looking to learn more about SDR real-time implementation. Add-on cards are available for the RTL-SDR to upconvert short wave to frequency ranges where the RTL-SDR functions, i.e. 25 MHz and above. Overall, the objectives of the workbook and materials are to:

- Convey the fundamental concepts and applications of SDR systems, from the RF, IF, and baseband stages of DSP enabled radio algorithms.
- Encourage an intuitive understanding of the RF spectrum, by demonstrating how to tune across the spectrum range of 25 MHz to 1.7 GHz, and to capture and view different signals in I/Q format, recognize modulation schemes, and plot live RF spectra on screen.

- Provide an appreciation of the different communications systems and standards in use, and the bands of RF frequencies they use, ranging from FM radio, to GSM, to ISM band and LTE.
- Demonstrate the fundamentals of the analog modulation schemes of AM and FM radio, and be able to construct real-time digital receivers for both AM and FM analog signals based around digitized I/Q SDR receiver algorithms.
- Review aspects of DSP digital receiver design (filters, demodulators, decimators, NCOs), and implement practical digital receivers from first principles.
- Consider the requirements for tuning, setting offset frequencies, carrier synchronization and phase locking, and symbol and data timing, and demonstrate how to design and implement these components as part of an SDR receiver.
- Show how to generate and transmit RF signals (using low-cost FM transmitters, USRP SDR hardware, custom designs, etc.) to build simple signaling layers and design PHY implementations to send data, music, images, and control information.

Conclusions and Some Next Steps

The open course materials discussed in this article create the opportunity to build and experiment with SDR techniques. Of course, with the our $20 RTL-SDR, we do not have precise control over ADC rates, programmable RF subsystems, nor controllable wideband antennas, and hence need to deal with high noise floors and frequency drift. However, this can be turned into part of the learning experience, e.g. finding the frequency offset of a particular RTL-SDR is one of the early exercises in the workbook [4]. Also, this first course on SDR implementation is working with single-channel antenna systems. Lowcost multichannel SDR is not so far away however. In fact with the current drivers for MATLAB and Simulink, we can host multiple RTL-SDRs (there are examples using two RTLSDRs in a Simulink design in Chapter 3 of [1]). Therefore, multiple input desktop opportunities for students and maker communities is here. We can also expect more low-cost and accessible SDR transmitters to become available, creating more opportunities and exciting prospects for the lab, of course realizing that wherever devices are adopted, we need to be aware of the available (legal) frequency bands that we can and cannot use.

We can conclude by stating that the RTLSDR is an excellent first SDR device that can form an IF digital radio and a front end for floating or even fixed point implementations of digital demodulators, receivers, and decoders using MATLAB and Simulink to bring SDR opportunities to the desktop. Finally, it is perhaps interesting to note that the RTL-SDR device is also currently trending and defining SDR in the “consumer” marketplace. In July 2015 a search on http://www.amazon.com for the term “Software Defined Radio” listed the RTLSDR as the top hit, with the next three hits also being products related to the RTL-SDR! Stay tuned. The wireless (SDR-enabled) revolution is just beginning!

References

[1] R. W. Stewart et al., Software Defined Radio using the MATLAB & Simulink and the RTL-SDR, Strathclyde Academic Media, 2015. ISBN-13: 978-0-9929787-1-6.

[2] F. Harris, Multirate Signal Processing for Communication Systems, Prentice Hall, 2004.

[3] M. Rice, Digital Communications: A Discrete Time Approach, Prentice Hall, 2008.

[4] J. Mitola, “The Software Radio Architecture,” IEEE Commun. Mag., vol. 33 , no. 5, May 2015, pp. 26–38.

[5] J. Mitola et al., “Guest Editorial on Software Radios,” IEEE JSAC, vol. 17, no. 4, April 1999, pp 509–12.

[6] W. Tuttlebee, (Ed.), Software Defined Radio: Enabling Technologies, John Wiley, 2002, ISBN 0-470-84318-7.

[7] E. Grayver, Implementing Software Defined Radio, Springer, 2012, ISBN-13: 978-1441993311.

[8] L. Goeller and D. Tate, “A Technical Review of Software Defined Radios: Vision, Reality, and Current Status,” Proc. Military Communications Conference (MILCOM), 2014 IEEE, 6–8 Oct. 2014, pp. 1466–70.

[9] S. G. Biln et al., “Software-Defined Radio: A New Paradigm for Integrated Curriculum Delivery,” IEEE Commun. Mag., vol. 52, no. 5, May 2014, pp. 184–93.

[10] El-Hajjar et al., “Demonstrating the Practical Challenges of Wireless Communications using USRP,” IEEE Commun. Mag., vol. 52, no. 5, May 2014, pp. 194–201.

[11] M. Petrova et al., “System-Oriented Communications Engineering Curriculum: Teaching Design Concepts with SDR Platforms,” IEEE Commun. Mag., vol. 52, no.5, May 2014, pp. 202–09.

[12] M. B. Sruthi et al., “Low Cost Digital Transceiver Design for Software Defined Radio using RTL-SDR,” Proc. Int’l Multi-Conf. Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 22–23 March 2013, pp. 852–55.

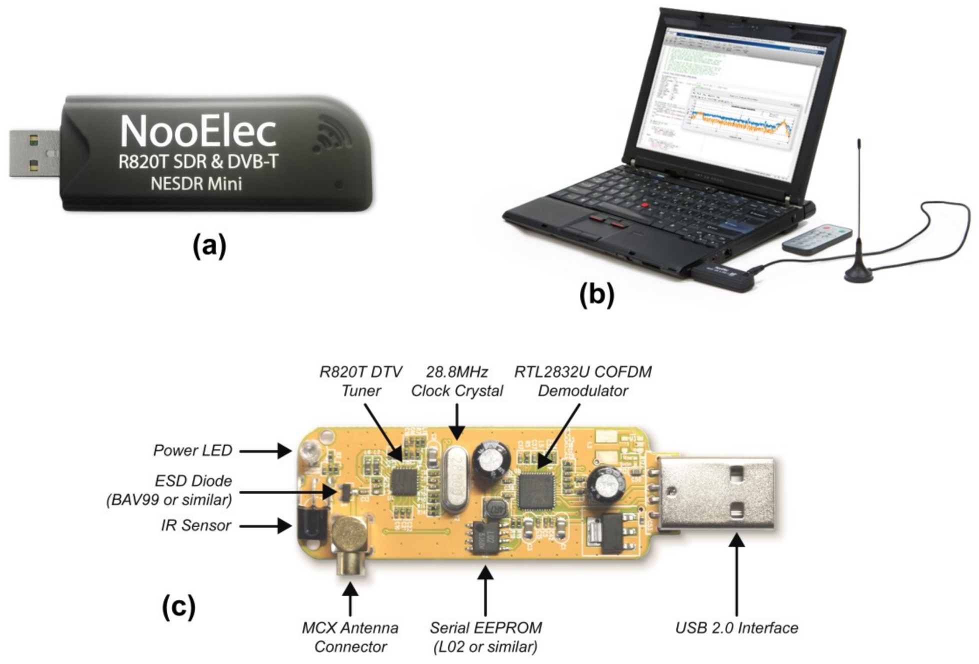


Figure 1: a) The RTL-SDR Mini USB device; b) a typical RTL-SDR receiver setup on a laptop running MATLAB and Simulink using a simple omni-antenna (comes with the RTL-SDR); c) the main internal components of the RTL-SDR.

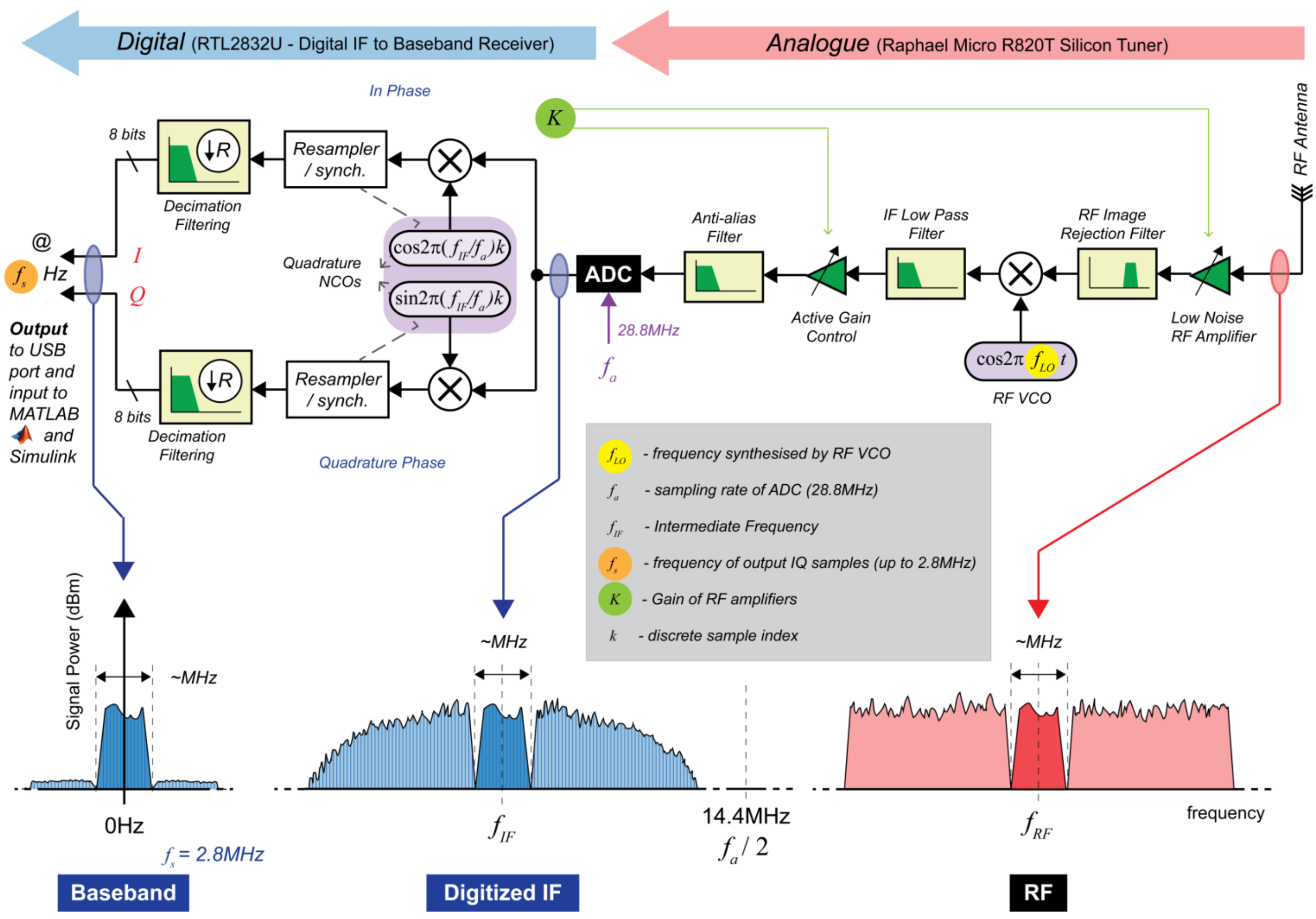
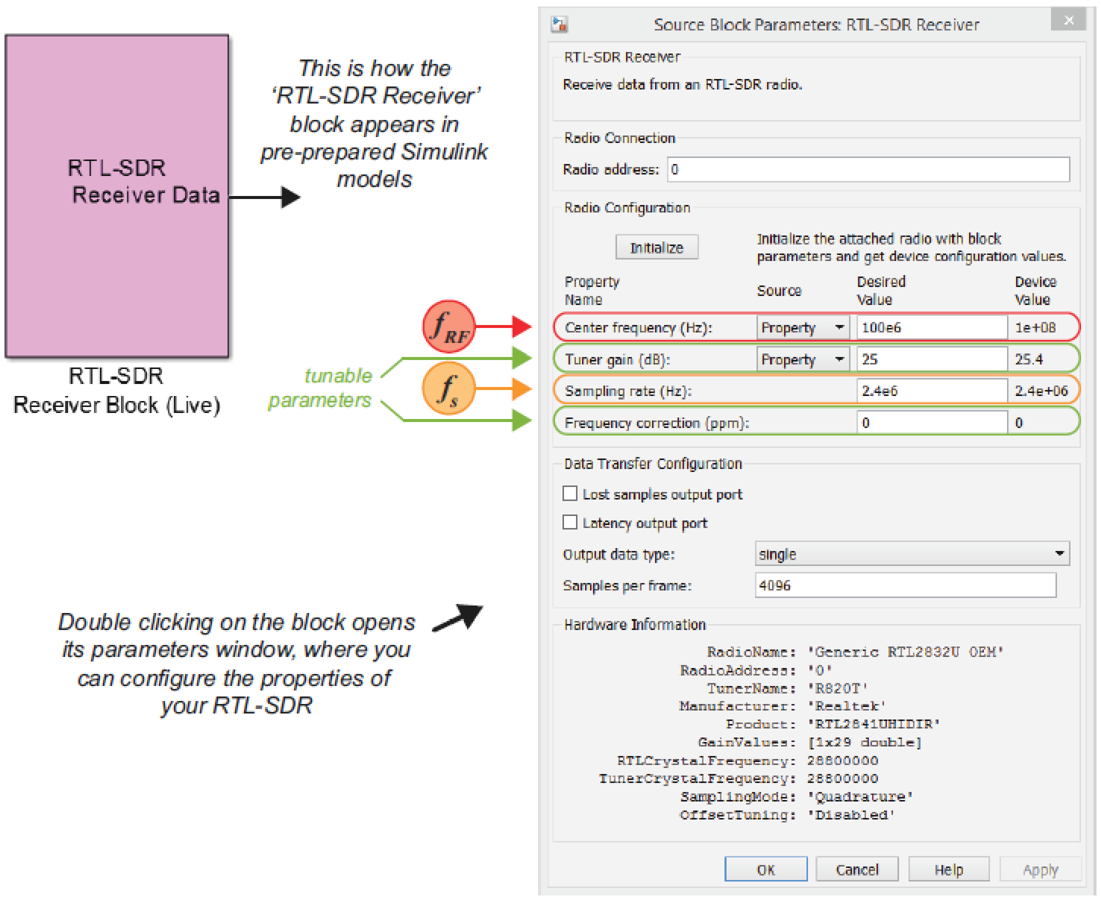


Figure 2: The main components of the RTL-SDR USB device. The main MATLAB and Simulink parameters of fRF, fs, and K can be set to configure the device (Fig. 3).


RTL-SDR
Receiver Data
RTL-SDR
Receiver Block (Live)
This is how the 'RTL-SDR Receiver' block appears in pre-prepared Simulink models
f_{RF}
f_s
tunable parameters
Double clicking on the block opens its parameters window, where you can configure the properties of your RTL-SDR
Initialize
OK
Cancel
Help
Apply

Source Block Parameters: RTL-SDR Receiver

RTL-SDR Receiver

Receive data from an RTL-SDR radio.

Radio Connection

Radio address: 0

Radio Configuration

Initialize the attached radio with block parameters and get device configuration values.

Property Name	Source	Desired Value	Device Value
Center frequency (Hz):	Property	100e6	1e+08
Tuner gain (dB):	Property	25	25.4
Sampling rate (Hz):		2.4e6	2.4e+06
Frequency correction (ppm):		0	0

Data Transfer Configuration

☐ Lost samples output port

☐ Latency output port

Output data type: single

Samples per frame: 4096

Hardware Information

```
           RadioName: 'Generic RTL2832U OEM'
        RadioAddress: '0'
           TunerName: 'R820T'
        Manufacturer: 'Realtek'
             Product: 'RTL2841UHIDIR'
          GainValues: [1x29 double]
   RTLCrystalFrequency: 28800000
 TunerCrystalFrequency: 28800000
        SamplingMode: 'Quadrature'
        OffsetTuning: 'Disabled'
```

Figure 3: The RTL-SDR block in Simulink and the configuring parameters.

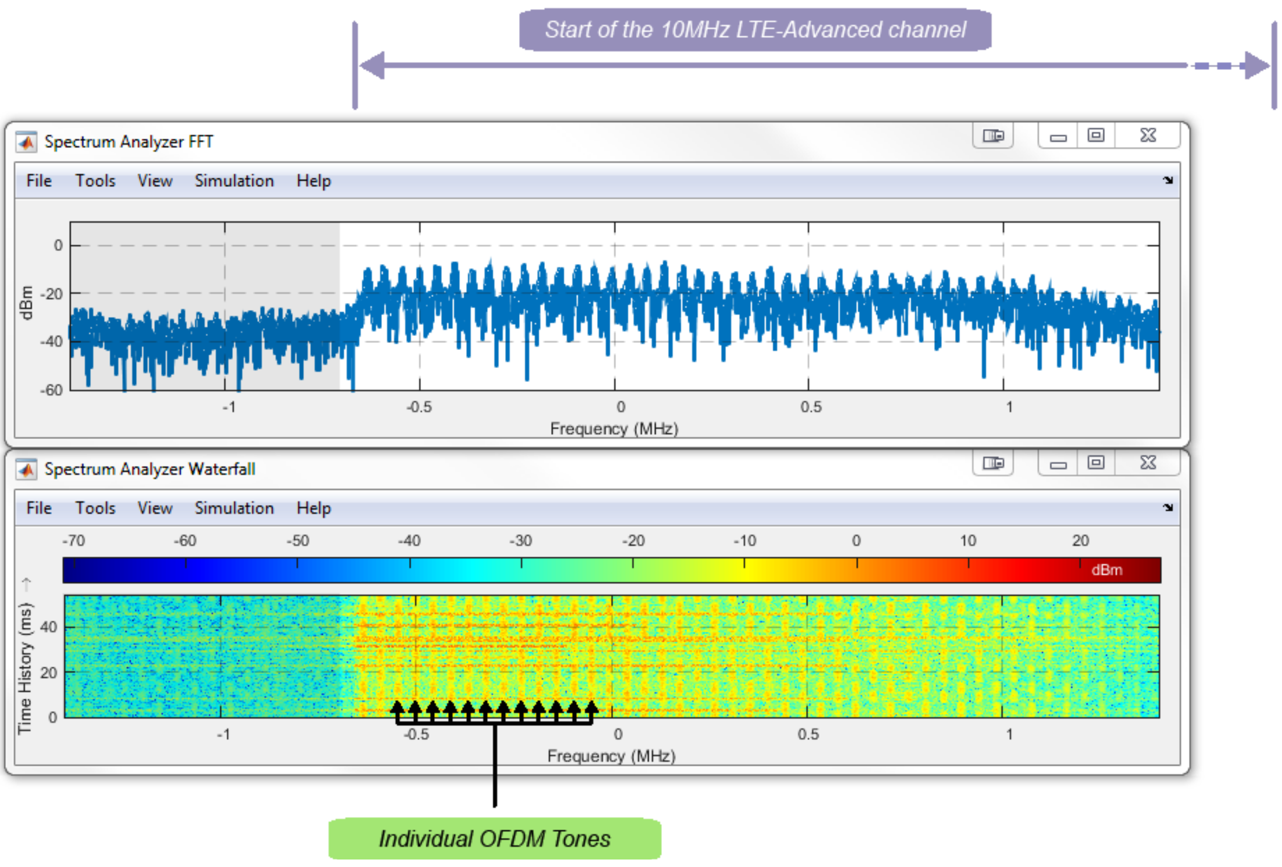


Figure 4: Using the RTL-SDR and a real-time Simulink spectrum analyser and 2D waterfall plot to view part of a 10 MHz 4G LTE signal spectrum in the 800 MHz band, clearly showing the OFDM carriers.

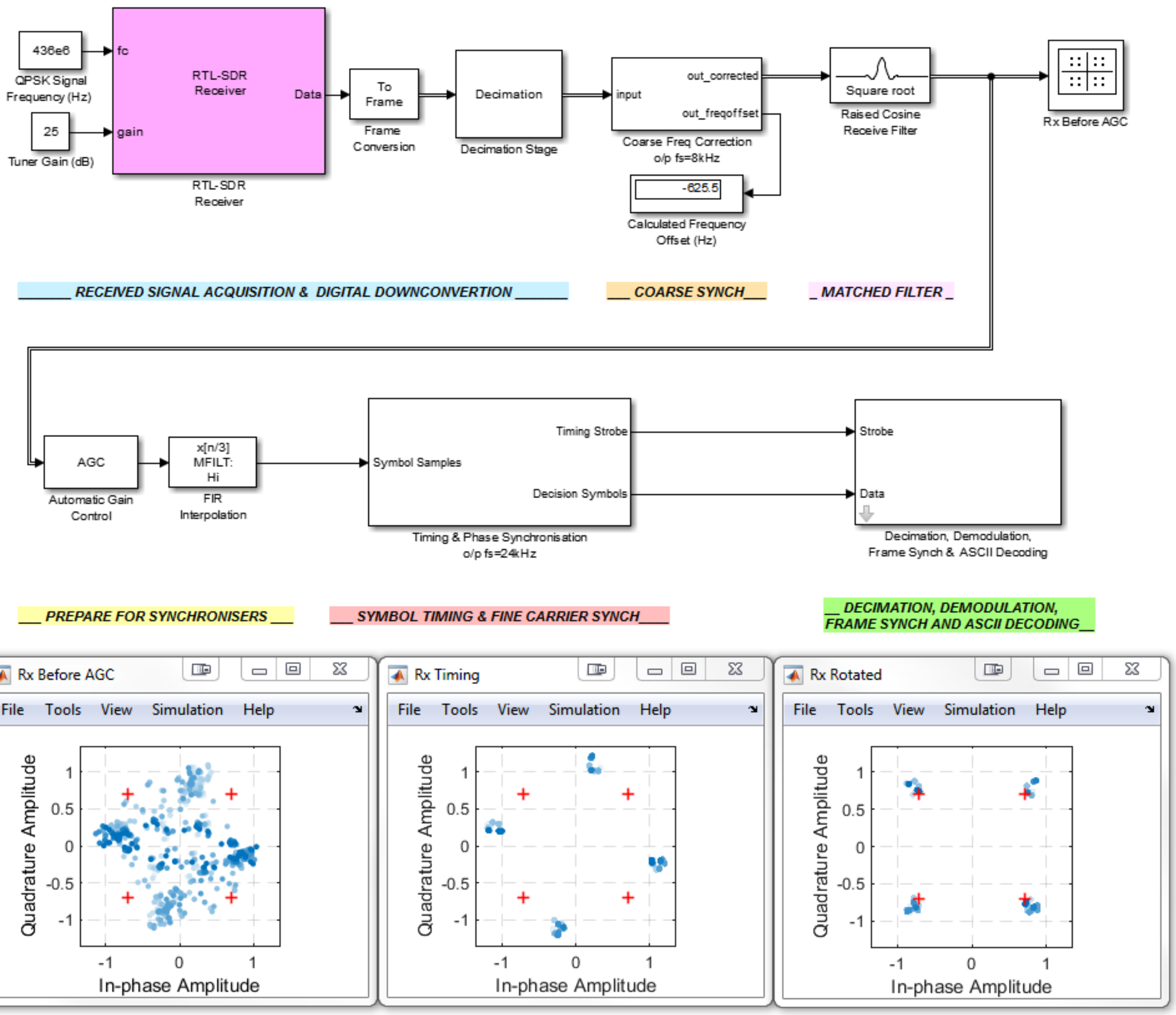

436e6
QPSK Signal Frequency (Hz)
25
Tuner Gain (dB)
fc
gain
RTL-SDR Receiver
Data
RTL-SDR Receiver
To Frame
Frame Conversion
Decimation
Decimation Stage
input
out_corrected
out_freqoffset
Coarse Freq Correction o/p fs=8kHz
-625.5
Calculated Frequency Offset (Hz)
Square root
Raised Cosine Receive Filter
Rx Before AGC
RECEIVED SIGNAL ACQUISITION & DIGITAL DOWNCONVERTION
COARSE SYNCH
MATCHED FILTER
AGC
Automatic Gain Control
x[n/3] MFILT: Hi
FIR Interpolation
Symbol Samples
Timing Strobe
Decision Symbols
Timing & Phase Synchronisation o/p fs=24kHz
Strobe
Data
Decimation, Demodulation, Frame Synch & ASCII Decoding
PREPARE FOR SYNCHRONISERS
SYMBOL TIMING & FINE CARRIER SYNCH
DECIMATION, DEMODULATION, FRAME SYNCH AND ASCII DECODING
Rx Before AGC
Rx Timing
Rx Rotated
File Tools View Simulation Help
Quadrature Amplitude
In-phase Amplitude
1
0.5
0
-0.5
-1
-1
0
1

Figure 5: Design of a real-time RTL-SDR system receiving a QPSK signal transmitted in the laboratory from a USRP at 433 MHz and implemented from first principles in Simulink, featuring receive filters, carrier and timing synchronization, and decimation stages.

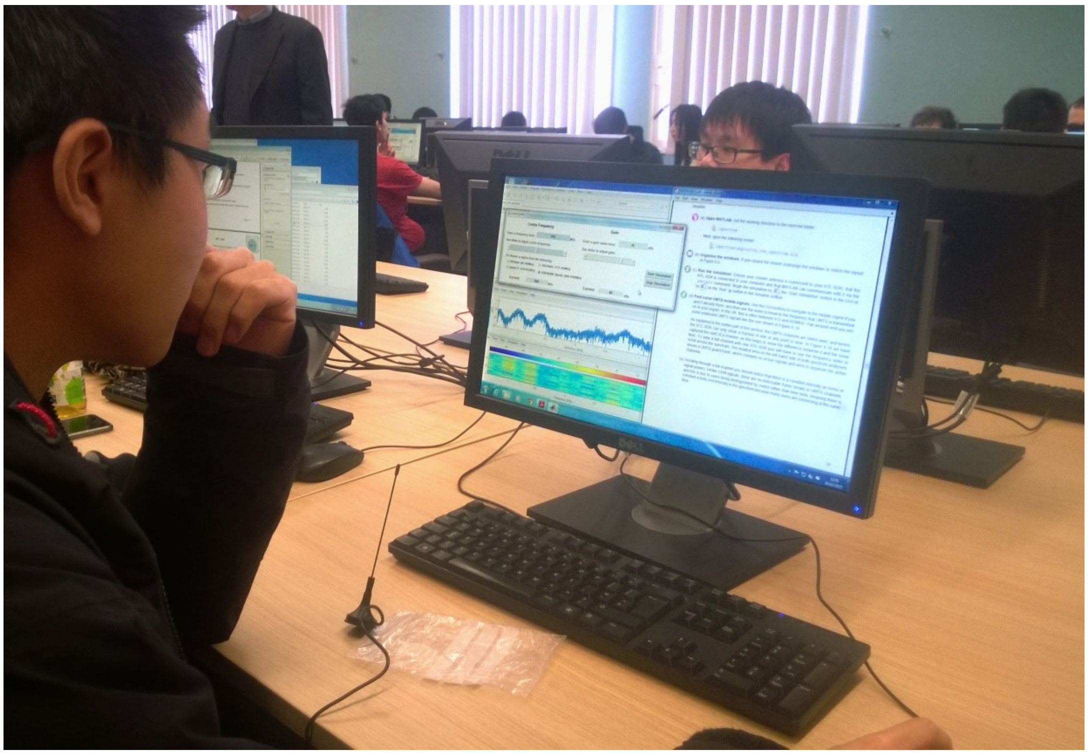


Figure 6: A software defined radio laboratory session at the University of Strathclyde to receive and decode an in-class transmitted signal. Note that the only hardware is the PC, and the RTL-SDR. Spectrum analyzer functionality is all provided by the same RTL-SDR device.

Biographies

BOB STEWART is the MathWorks professor of signal processing at the University of Strathclyde, and is also currently the head of the Department of Electronic and Electrical Engineering. He also manages a research group working on DSP, FPGAs, white space radio, and low-cost SDR implementation. He has a bachelors and Ph.D. from the University of Strathclyde.

LOUISE CROCKETT is the Xilinx lecturer in FPGAs and programmable logic at the University of Strathclyde in Glasgow. She is also the principal author of The Zynq Book, published in 2014, and has a core research interest in FPGA systems design for software defined radio and DSP systems. She has a masters and Ph.D. degree from the University of Strathclyde.

DALE ATKINSON is a Ph.D. student at the University of Strathclyde, working on SDR receiver systems for low-cost implementation. He received a bachelor’s degree from the University of Strathclyde in 2014.

KENNETH BARLEE is a Ph.D. student at the University of Strathclyde, working on novel DSP enabled radio algorithms and implementation for software defined radio. He received a bachelor’s degree from the University of Strathclyde in 2014.

DAVID CRAWFORD is the manager of the Centre for Wireless White Space at the University of Strathclyde, and also lectures on digital signal processing, using the RTL-SDR in the laboratory sessions. He was the managing director at EPSON Semiconductor before joining Strathclyde in 2010. He has four degrees from the University of Strathclyde: a bachelor’s, masters, MBA, and a Ph.D.

IAIN CHALMERS is a Ph.D. student at the University of Strathclyde, working on wireless white space radio architectures. He received a master’s degree from the University of Strathclyde in 2012. Previously he studied at California State University during his master’s degree.

MIKE MCLERNON is a development manager at MathWorks Inc. in Natick, MA. He leads software development activity on communications software products, with a particular interest in filter receiver design, SDR, standards-based modeling, and channel modeling. He has a master’s degree from Rensselaer Polytechnic Institute, and a bachelor’s from the University of Virginia.

ETHEM SOZER is a principal software engineer at MathWorks Inc. in Natick, MA. He specializes in software development for signal processing and communications toolboxes to support SDR. Previously he was a research engineer at Massachusetts Institute of Technology, where he developed underwater acoustic communication hardware and software platforms. He has bachelor’s and master’s degrees from Middle East Technical University, and a Ph.D. from Northeastern University.