A New Golden Age of Radio

Development tools for Software Defined Radio

Dr. Richard G. Ranson, Radio System Design Ltd

(richard@radiosystemdesign.co.uk)

Abstract

Software Defined Radio (SDR) is now the mainstream of radio communications technology. Previously it has involved a complex mix of traditional RF analog hardware, fast digital logic including quite challenging analogue to digital converters (ADC) as well as interface software; making it rather daunting for companies interested in possible applications but not well versed in the various technologies. However, hardware manufactures now have competitively priced development platforms bring down the cost of entry into this market. Then in parallel to this there has been a standardisation of software and interfaces to these devices, greatly simplifying things. The result is that the cutting edge of radio technology has never been more accessible.

Where as previously, companies needed a clean room, other facilities, as well as specialist test and assembly equipment costing £100,000s, now all that is needed is a good soldering station and a lab bench worth just £100s. Basically, with even a half decent computer, some open source software and a low cost development platform, experimenters and professionals alike can experiment with SDR and work with real signals to develop new ideas and products. In this talk I will show some examples of SDR hardware, one of which is the ADALM Pluto costing <£100, but with surprisingly advanced capabilities. To supplement that, a suite of open source software including GNU Radio can be assembled into a standalone Linux distribution that can be put onto a USB memory stick and run on any mac or a pc. This can then be the basis to get started and work on your own ideas for radio products and applications.

Introduction

In the last 15 years, entirely behind the scenes, basic radio technology has been completely revolutionised. First, the full scale adoption of digital modulation and advanced coding has enabled bandwidth efficiency approaching the Shannon limit. Then, the integration of digital and RF CMOS has created sophisticated, flexible hardware platforms capable of converting data up to, and down from, low microwave frequencies, using only a hand full of components. The end result, SDR, is now professional main stream, but thanks to the decreasing cost of hardware, innovative development platforms and a wealth of software tools the cost of entry into this market has fallen to a level accessible to all.

In this paper and the accompanying talk, I will review some of the common SDR hardware devices and software interface requirements. Some have straight forward driver installation while others require some command line magic. But, all that can be rolled up into a custom Linux distribution to get someone stated. The final section shows the remarkable GNU Radio with the GUI base extension GNU Radio Companion (GRC).

Hardware

RTL SDR

This is the entry point for many. It covers a number of different devices which are all basically USB dongles to receive DAB radio and digital TV. They are a hobbyists first choice because the cost is typically <$20, but various projects have shown that useful things can be achieved even with this quite primitive device. It is receive only, with various models using different RF tuners and the RTL2831 or a derivative as the ADC and decoder. There are software drivers to bypass the decoder and output the raw IQ data samples for further processing in a pc. Roughly speaking these devices can tune from 70 to 2000 MHz with 8 bit, 4 MHz digitisation. There are however various hacks to enable much lower than specified operating frequencies, opening up the device to HF radio enthusiasts.

One pioneer in this area is Osmocom [1], who have an excellent web site reporting on the activity of others as well as drivers and other useful software tools such as local and remote logging of data samples, processing the FFT of samples and streaming demodulated broadcast FM radio signals. Below are just a few interesting, at least to me, projects that illustrate what can be done with such a simple device:

- GNSS SDR – Receiving various global positioning system satellite signals [2]
- DUMP1090 – Decoding ADS-B signals used for aircraft location and navigation [3]
- LTE Cell Scanning [4]
- Receiving high definition weather satellite images [5]
- Decoding local bus stop display transmissions [6]

Pluto SDR

This is part of a family of teaching and demonstration devices produced by Analog Devices Inc (ADI) (Figure 1). It was released last year, but only just now becoming available to UK components distributors. The cost from ADI is $150, but is available at a discount of ~$100 or ~£80. There is a wiki and lots of other useful information, including schematics and firmware on the ADI web site [7].

Pluto is a 1 Tx by 1 Rx, full duplex, device with the key parts being an AD9361 RF transceiver and a Zinq FPGA, providing a bytes to RF and back peripheral, powered by the USB interface (see Figure 2). The advertised frequency range is 325 - 3800 MHz, but experimenters have shown that it can be hacked to work from 70 – 6000 MHz [9]. The AD9361 provides 12 bit digitisation (ADC and DAC) with up to 30 MHz bandwidth, however in practice this can be restricted by the USB interface BW. The FPGA provides a small footprint Linux kernel for very basic interface and control via SSH and a virtual TC/IP port via the USB connection for data.

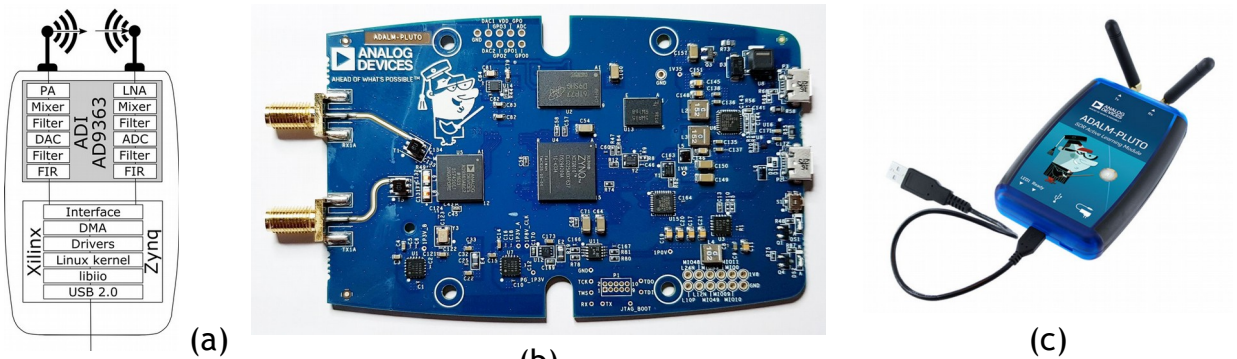
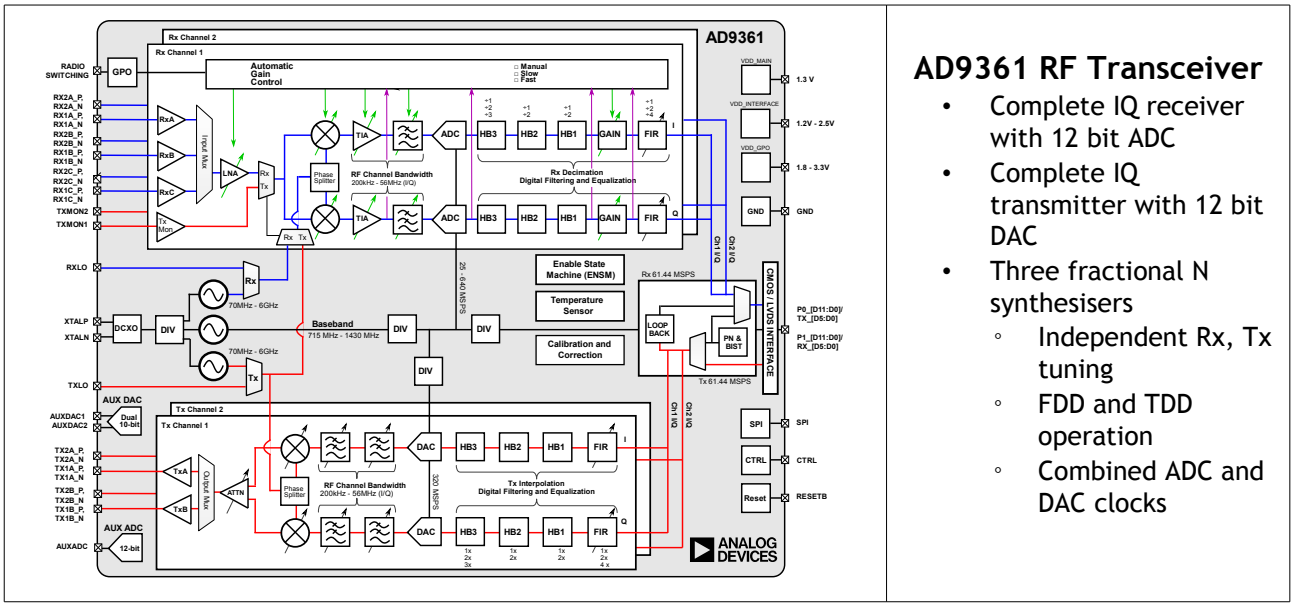


Figure 1: ADALM Pluto (a) functional block diagram,(b) PCB top view and (c) case

The FPGA also facilitates some useful RF features, such as programable interpolation/decimation and digital filtering as well as some BIT functions such as generating Tx data from a Direct Digital Synthesiser (DDS) implemented in the FPGA. This allows a one or two tone test signal to be generated internally and transmitted from the device itself.

More general and useful control and data flow is facilitated via the Industrial I/O Interface (IIO) library. The IIO library provides an Applications Programmers Interface (API) for direct control of the hardware (e.g via C++ or python), with low level register access as well as more useful high level data streaming to/from the AD9361 device. A good example of the utility of the API is a wrapper for the library that provides a plug in interface to Pluto for GNU Radio, but more on that later.



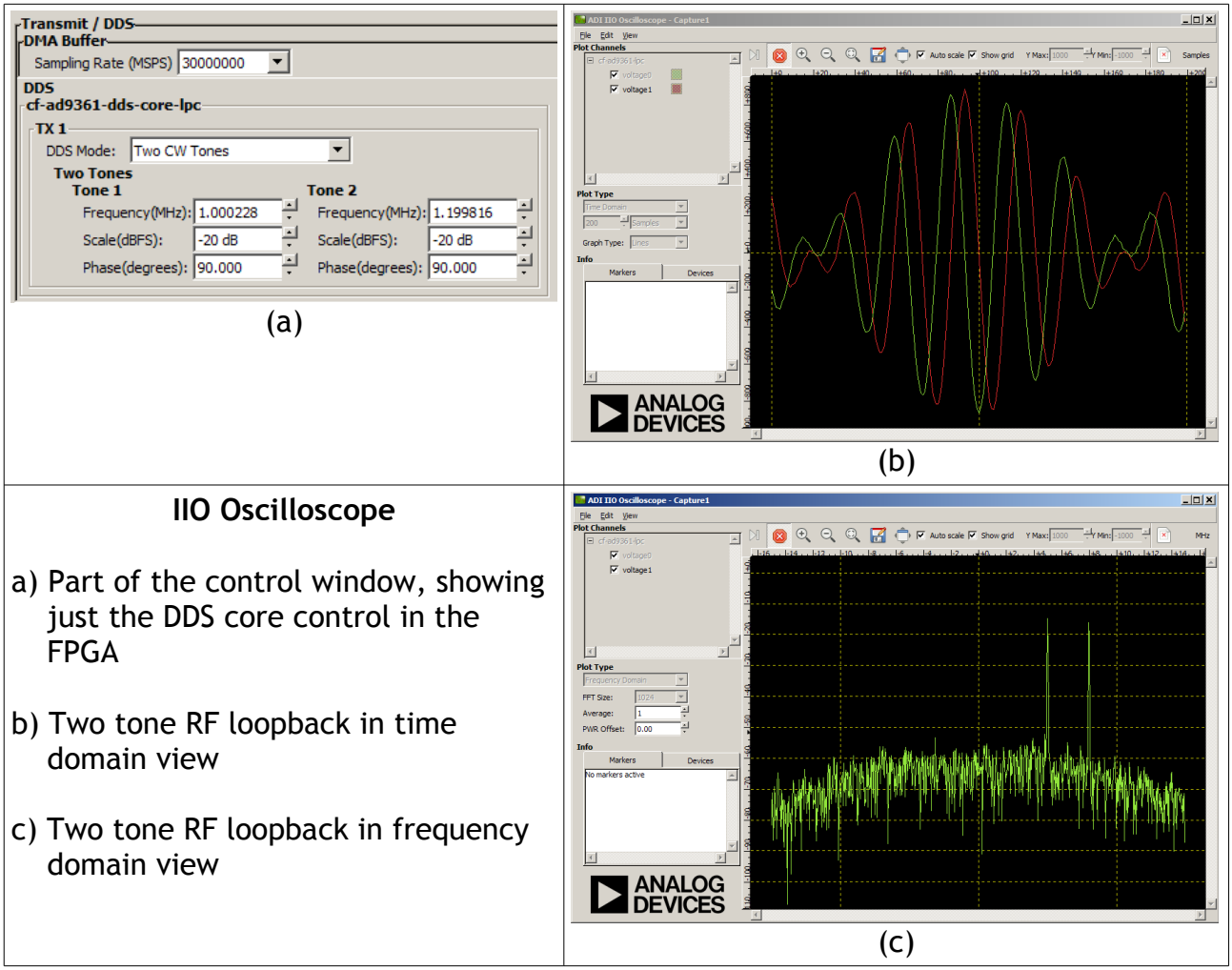
AD9361 RF Transceiver

- Complete IQ receiver with 12 bit ADC
- Complete IQ transmitter with 12 bit DAC
- Three fractional N synthesisers
  - Independent Rx, Tx tuning
  - FDD and TDD operation
  - Combined ADC and DAC clocks

Figure 2: summary of AD9361 capabilities.

ADI also provides a standalone GUI interface to Pluto called IIO Oscilloscope that allows real time control of many of the features of the device as well as a display of the received signal in the time or frequency domain.

This is shown in Figure 3, where some screen captures show one of the control panels (a) for the DDS generator, where you can select up to 2 tones at various frequencies and phases. The Tx is looped back to the Rx via a cable and parts (b) and (c) of the figure show the time and frequency domain views of the 2 tone signal detected by the receiver.



IIO Oscilloscope

a) Part of the control window, showing just the DDS core control in the FPGA

b) Two tone RF loopback in time domain view

c) Two tone RF loopback in frequency domain view

Figure 3: IIO Oscilloscope application.

Lime SDR mini

Myriad RF is a UK company that has pioneered various open source hardware and software devices [14]. The LimeSDR mini is only recently available, as a crowd funded project spun off from the more professional LimeSDR. The mini is a 1 Tx, 1 Rx full duplex transceiver using the LMS7002M tuner which has much of the digital I/O also integrated onto the chip and a USB interface. The key specifications are tuning range 10 – 3500 MHz, with 12 bit digitisation and up to 30.72 MSPS. The drivers and software installation is the same as for the older, more capable LimeSDR, but is has been reported to be quite complex to install, so it is not recommended as an introduction to SDR for most people. Those who have a device have shown impressive results using some well known applications such as GQRX, SDRangel [8], SDR#, SoapySdr and Pothos, which is a GNU Radio spin off.

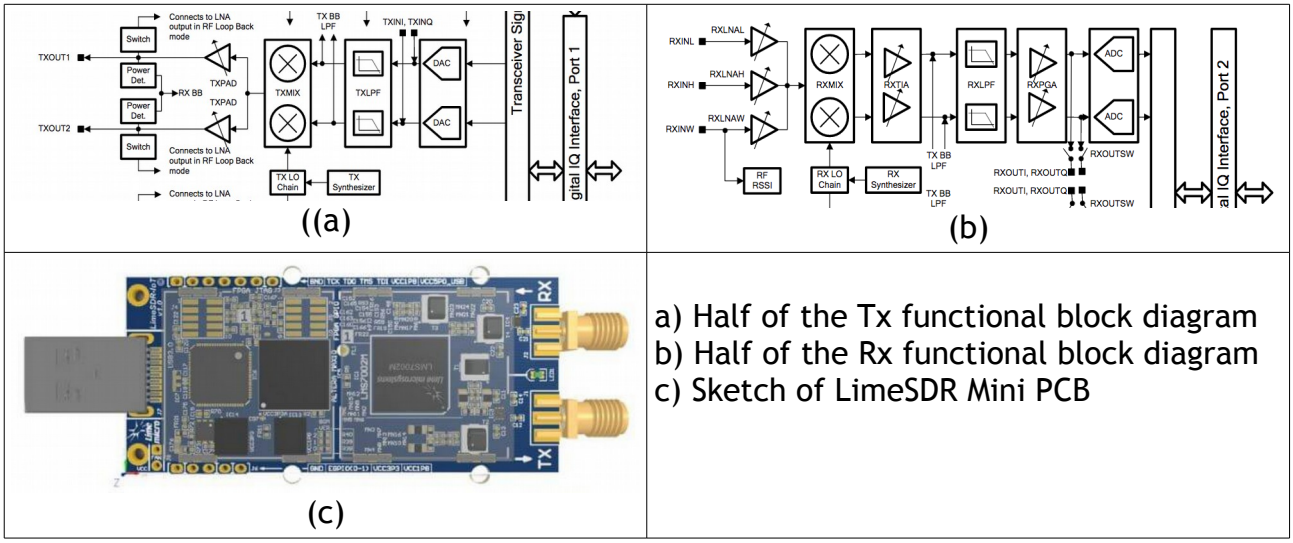


Figure 4: LimeSDR Mini and key elements of LMS7002M block diagram.

Figure 4 shows some information on the LimeSDR mini. In (a) and (b) I have extracted key parts of the LMS7002M Tx and Rx signal paths. In essence, much like the AD9361, the LMS7002M chip has an IQ up/downconverter, DAC/ADC and digital filtering. The same chip has other programmable DSP (Digital Signal Processing) hardware as well as serial data for control and I/O interfaces to stream IQ data for Rx and Tx.

Unfortunately, even though I ordered a device in November with a January 2018 promised delivery, this has slipped; so I have not received the board and so not been able to perform any measurements. Hopefully I will have one in time for the presentation and to show in the exhibition space at the meeting.

Other Devices

There are a number of other low cost devices that are used for SDR experimentation, many of which are reviewed on the RTL-SDR web site [10]. A more advanced RTL style device is called the Fun Cube Dongle [11]. It covers 150 kHz -240 MHz and 420 MHz – 1900 MHz with the most notable feature being selectable RF preselection bands. This makes it quite a bit more practical than most RTL devices with both a respectable noise figure and intermodulation intercept. Another popular receive only device, also from the UK, is the SDR Play [12]. This is a 1 kHz – 2000 MHz receiver with 14 bit quantisation and up to 10 MSPS sampling. Great Scott Gadgets makes the Hack RF One [13], now a quite mature device, also popular with hobbyists having both Rx and Tx capabilities but with only half duplex operation. It covers 1 – 6000 MHz with 8 bit quantisation and up to 20 MSPS. For even higher performance, Lime have a more mature SDR with multi-channel capability and higher sampling rates [14]. Interestingly, this uses the same LMS7002M chip, so despite being just 1x1 TRx, the mini has a multi channel RF capability. Lastly, Ettus Research, now part of National Instruments, makes a range of devices under the umbrella, Universal Software Radio Peripheral (USRP) [15], these are self contained, boxed and at the high end of the price performance curve.

Software

While the hardware is impressive, it is only useful, because of drivers and free software to connect to and work with the radio Then thanks to the freeware Linux OS, there are rapidly developing standarised libraries and programming techniques.

All hardware vendors provide basic drivers and installation software to get you started. Also there are programs like SDR#, GQRX and others that are general purpose, in the sense of working with different devices, and in essence provide a front panel and display for the underlying SDR. All this is only possible because various standard libraries and software APIs. But there are also more specialist tools to help understand and use DSP with this kind of hardware. Out of those I have chosen to highlight GNU Radio for reasons that I hope will be apparent.

GNU Radio

This is a fantastic open source project that was started almost 10 years ago, and has developed into an industrial standard. There is a documented C++ API for dedicated applications and maximum performance, but there are also python wrappers for many useful functions as well as an introductory/educational level of operation via GNU Radio Companion (GRC). It provides a graphical interface to a system simulation canvas and access to a huge library of common DSP blocks. Ideas can be developed and analysed by connecting various blocks into flow graphs that consist of one or more sources of data, some processing, and one or more sinks to view and save the data. There are ideal source blocks such as signal and nose generators, but hardware such as an RTL SDR or Pluto have dedicated source blocks as well. Similarly, sinks can be visualisation devices such as an oscilloscope (time sink) or a spectrum analyser (frequency sink), or transmit hardware like Pluto. GRC is an ideal way to get started with DSP and SDR concepts as well as a powerful tool develop your own ideas and systems.

The GNU Radio project has a number of useful introductory tutorials and Figure 5 shows part of one that illustrates several SDR concepts [16]. The flowgraph shows a random data source feeding a constellation modulator that in this case creates a DQPSK data stream which is then upsampled by 4 and band limited using a root raised cosine (RRC) filter. Such a signal could be fed to an actual transmitter sink such as Pluto, but as this is just a simulation, the data is fed to a channel model block to emulate noise and frequency selective distortion from Tx to Rx. The simulated receiver blocks are on the second row of the diagram. The polyphase clock synchroniser is used to recover symbol timing and the Constant Modulus Algorithm (CMA) block to resolve frequency selective distortion (i.e. multipath). There are 4 sink blocks to visualise the receive signal in frequency and as a constellation diagram before and after the CMA block.

Note that the actual flowgraph incorporates a number of control elements and other structural blocks that while not that complicated are not essential to demonstrate the capabilities here. If you follow the reference [16] then there is a link to the git hub repository for all the grc files where you can see all the details of this and other example flowgraphs.

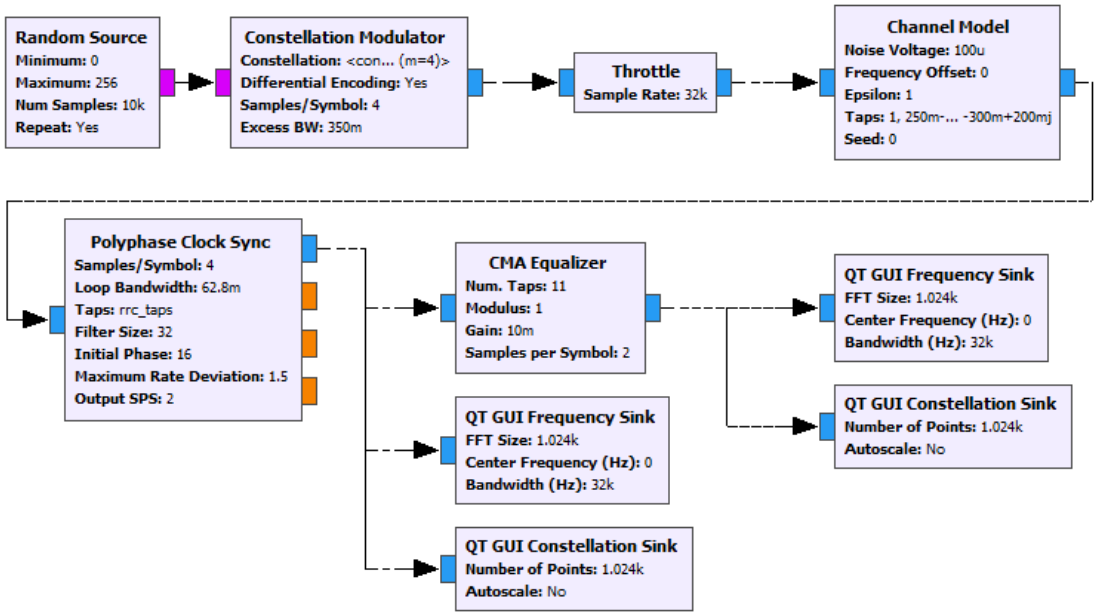


Figure 5: GRC flowgraph simulation of a DQPSK Tx RX link with timing recovery

Figure 6 shows the output from the simulation. Note that there is no timing information transferred between blocks so the Tx and Rx are independent of each other. The channel also adds noise and frequency distortion to the signal to be representative of a real system. The polyphase synchroniser recovers the symbol timing from the signal stream and that is evident by points clustered round synchronised symbol points in the top left constellation display. But there is still both noise and frequency distortion from the channel, which is evident as noise (fuzziness in the constellation clusters) and frequency distortion in the spectrum display below.

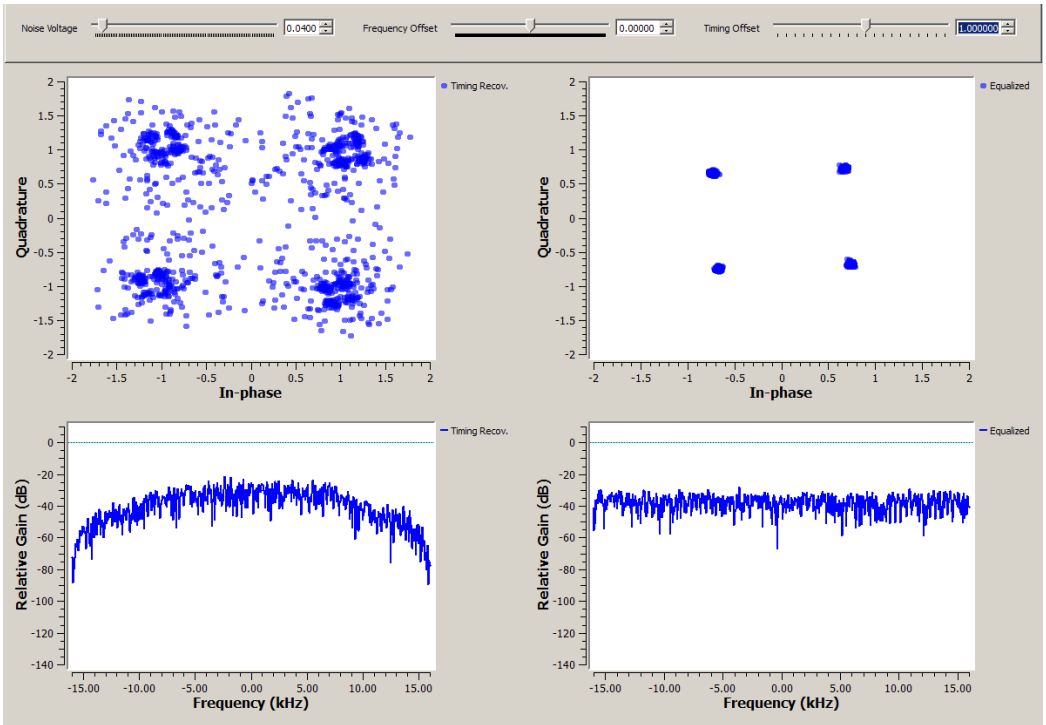


Figure 6: frequency and constellation sink displays from the GRC flowgraph Figure 5

The right hand constellation and frequency displays shows the signal after the CMA block. This uses the fact that the signal should have a constant average power to help determine and then correct for multipath distortion. It is clear that the constellation cluster is much tighter than on the left hand side, (not just a point because of the added noise) and the frequency response is now flat. The combination of the polyphase synchroniser and CMA blocks achieve recovery of the symbol timing, optimising the receiver sampling clock, minimise inter-symbol interference and correction for amplitude and phase distortion from the transmission channel. These and other techniques are the key to the power of DSP and GRC helps illustrate the technology and innovation that has enabled modern high speed, high efficiency radio communications that today approaches the Shannon limit.

This is just a flavour of the capabilities of GNU radio. To keep the diagrams simple and illustrate principles, I have used some of the high level function blocks within GRC, but the software includes a huge library of elements to explore lower level detail where it maybe desirable. It is also extendable, giving users the option to create their own blocks either in python or C++ as well as separate, so called OOT (Out of Tree) modules.

The PlutoSDR source and sink are such OOT examples, build from source provided by ADI [17] and using the industrial IIO library API. For example, Figure 7 shows a GRC flow graph to generate a random QPSK data sequence feed to the PlutoSDR sink, which is the Tx side of the device. The colour of the I/O ports on the blocks indicate the data type, with green being integer, orange float and blue complex data. Note also that this illustrates a few of the low level DSP block provided by the GNU radio library and alluded to earlier.

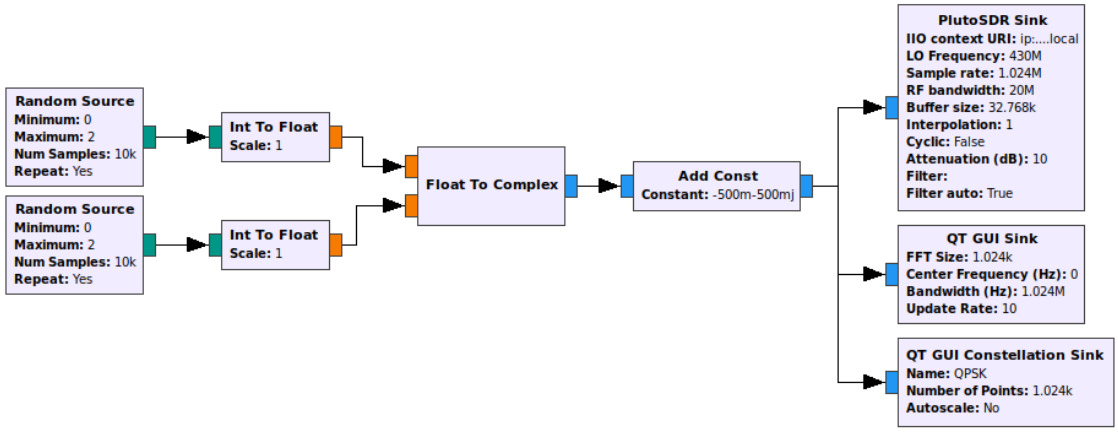
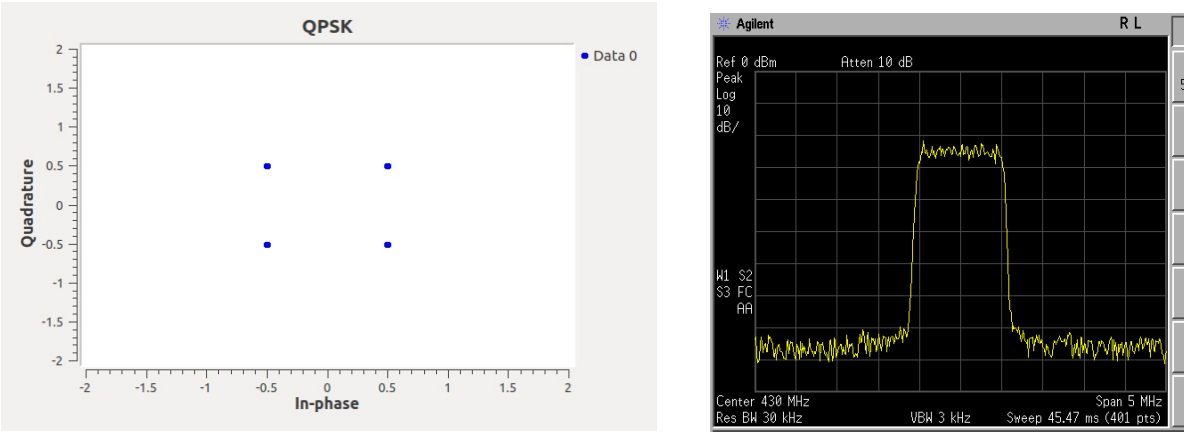


Figure 7: flowgraph for the Pluto sink as a QPSK transmitter

The constellation plot shown in Figure 8 (a) clearly shows the QPSK format and (b) shows a spectrum analyser measurement of the Pluto Tx output. Notice that the base band signal in (a) is just QPSK data with one sample per symbol, this is upsampled and half band filtered internally by the AD9361 hardware, then up-converted to 430 MHz. The signal measured in (b) is from a spectrum analyser showing that the output RF spectrum is well defined and quite clean. As an aside, note that the signal is comparable in quality to one from a signal generator costing several 100 times the price of Pluto.



(a) (b)

Figure 8: results of the flowgraph Figure 7, (a) constellation of data generated and (b) measured spectrum of the PlutoTx output

In summary GNU Radio provides a flexible and comprehensive frame work to investigate DSP and SDR technology. It gives access to basic as well as high level DSP function, ideal and real signals as well as an interface to real hardware. A tremendously comprehensive software package under the GPL-3 licensing umbrella; meaning, at the most basic level, that it is free for personal as well as commercial use. Quite remarkable.

Teaching Platform and Linux Distribution

Radio System Design Ltd has developed a Linux live boot disk (actually a USB drive) as a stand alone platform for teaching SDR. It can be used with a mac or pc to provide a common Linux based GUI desktop environment to work with SDR devices. The USB drive contains an Ubuntu live distribution with drivers for RTL devices, Fun Cube Dongle, PlutoSDR and some USRP devices. Various free support software is also installed including GNU Radio, IIO Oscilloscope, Jupyter (previously iPython) and Octave.

A "Hands on SDR", training course will be launched this year, using the student's laptop and booting with this specialised Ubuntu distribution from a USB drive. Then, using the PlutoSDR device and the various software tools, student can experiment with SDR concepts, transmit and receive actual signals and learn about this exciting new area of radio technology by working with actual hardware. If this might be of interest to you or your organisation, I have included a quick (<2 mins) survey that you can fill in to help steer the direction of the course. [18]

Conclusion

All the hardware and software used for this talk cost less than £250 (after all, I do live in Yorkshire;-)

The hardware examples given illustrate a wide range of price and performance for a number of potential applications. The real story is, that each one gives a level of performance for a price that is a fraction of any previous generation of similar devices. Coupled with low cost or free software, these devices provide a development platform within the budget of even the smallest of SMEs. Perhaps more impressive, is that they provide capabilities at frequencies that were previously regarded as quite exotic, dramatically lowering not just the monetary cost, but also the specialisation and degree of difficulty needed to get ideas and products to market.

Finally, it is worth considering the implications for some traditional applications. A brief search on the web shows people emulating (or worse) a GSM base station with this kind of hardware. Then on the military side, while something like a LimeSDR does not have state-of-the-art performance, it is pretty impressive. So with some imagination, additional RF hardware, particularly RF filtering, and a laptop computer, it could easily provide a medium to high performance Signal Intelligence (SIGINT) platform in a briefcase for under £3,000. That is something like a factor of 20x less than traditional approaches. Then, with such low cost RF hardware, multiple devices could be used for DF, coordinated for phased array capabilities such as radar warning or coordinated in time to scan and provide a comprehensive EW platform. These devices are truly revolutionising radio technology.

References and Links

Much of the power of the devices and ideas discussed comes from the open software and hardware initiatives in recent years. I have included many links that I have found useful as a way to help others get started. But I have kept it to only one per topic, so this is by no means complete.

Also a note of caution, one of the drawbacks of these open source, open hardware style devices, is that the developers are so immersed in the product that they think that providing all the schematics and source code is a substitute for any other documentation. Those interested in looking further should tap into the many blogs, user forums, YouTube videos and sources such as Hackaday where the same community is more communicative and helpful to beginners.

1. https://osmocom.org/projects/sdr/wiki/rtl-sdr
2. http://gnss-sdr.org/docs/tutorials/gnss-sdr-operation-realtek-rtl2832u-usb-dongle-dvb-t-receiver/
3. https://www.rtl-sdr.com/ads-b-decoder-dump1090-now-available-windows/
4. http://www.evrytania.com/lte-tools
5. https://www.rtl-sdr.com/tag/weather-satellite/
6. https://www.rtl-sdr.com/reverse-engineering-radio-controlled-bus-stop-displays/
7. https://wiki.analog.com/university/tools/pluto
8. https://www.rtl-sdr.com/limesdr-mini-unboxing-initial-review/
9. https://www.rtl-sdr.com/adalm-pluto-sdr-hack-tune-70-mhz-to-6-ghz-and-gqrx-install/
10. https://www.rtl-sdr.com/
11. http://www.funcubedongle.com/
12. https://www.sdrplay.com/
13. https://greatscottgadgets.com/hackrf/
14. https://wiki.myriadrf.org/LimeSDR
15. https://www.ettus.com/
16. https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC
17. https://wiki.analog.com/resources/tools-software/linux-software/gnuradio
18. https://www.surveymonkey.co.uk/r/HKP9MZ9