Laboratory 1: Introduction to Software-Defined Radio
Communication Systems Laboratory

Cory J. Prust, Ph.D.
Electrical Engineering and Computer Science Department
Milwaukee School of Engineering
Last Update: 19 September 2018

Contents

0 Laboratory Objectives and Student Outcomes

0.1 Laboratory Objectives

This laboratory introduces students to the software-defined radio (SDR) based laboratory that will be utilized in EExxx. Students will install and verify the operation of their personal SDR device in MATLAB/Simulink, and then construct a Simulink model for viewing signals in the radio frequency spectrum.

0.2 Student Outcomes

Upon successful completion of this laboratory, the student will be able to:

- Describe the basic components of a SDR.
- Compare and contrast an SDR with a traditional radio system.
- Verify that their personal SDR device is properly communicating with the MATLAB/Simulink environment.
- Construct a simple spectrum analyzer model in Simulink using their personal SDR device.
- Identify and investigate a real-world radio frequency signal using their personal SDR device.

1 Overview

1.1 Introduction

Welcome to the EE4022 laboratory! Throughout the term we will use these laboratory sessions to help you better understand the course material by putting the theoretical concept into practice. The laboratory exercises provide illustrations and visualizations of the many mathematical concepts needed in the study of communications. Furthermore, you will explore various communication signals and investigate end-to-end communication systems and their performance. Our approach will include both simulation and actual implementation on modern hardware platforms.

The hardware used in this laboratory is capable of wireless communication using radio waves. Such communication is ubiquitous in today's world - cell phones, WiFi, Bluetooth, high-definition television broadcasts, and even personal transmitters such as remote key fobs (i.e., to open your car doors or garage door) use wireless communication. We will describe, model, and implement the underlying principles used by these systems during our laboratory, thus giving your a foundational understanding of how such systems work. We will also observe and analyze real-world communication signals that are literally surrounding us!

Our experiments involving wireless communication will utilize modern software-defined radio (SDR) hardware. This is the same type of hardware used by today's communication system engineers. As its name implies, some "software" is required. However, we will use well-developed software tools that make our laboratory tasks readily accessible. And, rather fun!

The radio spectrum is a public resource and is federally regulated. The Federal Communications Commu-nication (FCC) is the primary authority for regulation in the United States. Because we will be working with radio signals, even transmitting some ourselves in the laboratory, it is important that we comply with federal regulations. We will regularly use the 915MHz ISM band for our in-laboratory experiments.

1.2 Definitions

In this section we collect important terms and definitions used throughout our laboratories:

- **Baseband Signal**: A baseband signal is one in which the signal energy is contained in frequencies relatively close to 0Hz. That is, the signal energy is contained in the range 0Hz to f_m. The quantity f_m is then typically referred to as the signal bandwidth. Examples include human voice (20Hz to 5kHz), audio/music (20Hz to 20kHz), and digital logic signals used within an embedded computer system (frequency is hardware dependent).

- **Passband Signal**: A passband signal is a one in which the signal energy is contained in a band of frequencies centered about some frequency, f_c. That is, the signal energy is contained in the range $f_c - f_m$ to $f_c + f_m$, where $f_c >> f_m$. The signal bandwidth is $2f_m$.

- **Message Signal**: A message signal is the information-bearing signal. The purpose of a communication system is to send message signals from a transmitter to a receiver through a communication channel. Message signals are typically baseband signals.

- **Carrier Signal**: A carrier signal is a waveform that is modified in accordance to a message signal. Carrier signals are typically much higher frequency than message signals. We will consider sinusoidal carrier waveforms, denoted by frequency f_c.

- **Modulation**: Modulation is the process by which a message signal is imparted on some characteristic of the carrier wave. The information-bearing message signal is referred to as the *modulating signal* and the result of the modulation process is referred to as the *modulated signal.* Modulation, typically performed at the transmitter, usually results in a passband signal.

- **Demodulation**: Demodulation is the process by which the message signal is extracted from a modulated signal. Demodulation, typically performed at the receiver, will ideally result in the original baseband message signal.

2 What is a software-defined radio?

2.1 Overview of software defined radio

Software-defined radio (SDR) technology has transformed much of the modern communications and networking fields. SDR has been used for several decades in military and commercial applications. Like many other forms of digital hardware, recent developments have provided increased performance, improved ease of use, and decreased costs. As a result, SDR is now used in a wide variety of applications such as communication systems, radar, radio-frequency (RF) identification, radio astronomy, and other forms of remote sensing.

Generally speaking, an SDR is a flexible hardware platform in which the majority of the radio functionality is implemented in software. While traditional hardware-based radios are designed for one or a small number of applications, the functionality of a single SDR can be modified via firmware updates and changes to the back-end signal processing in order to implement a wide variety of systems. Therefore, SDR systems lie at the intersection between RF electronics, embedded computing, and digital signal processing.

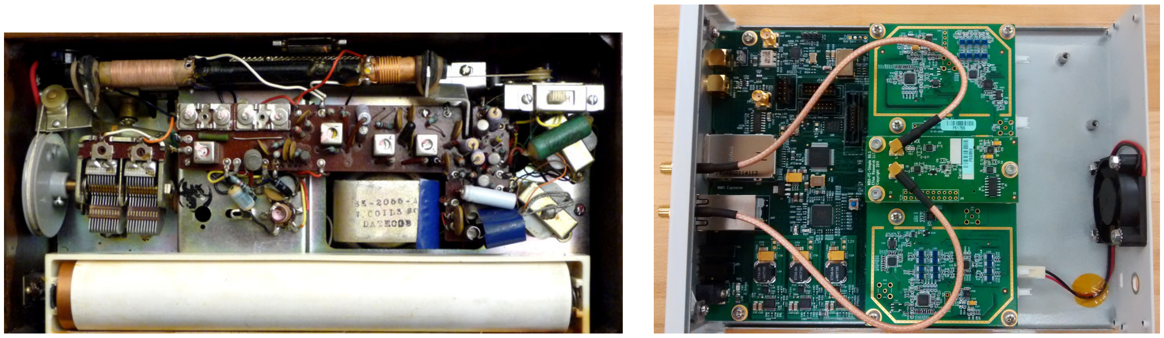

(a) (b)

Figure 1: Traditional radio hardware versus software-defined radio. (a) JVC Nivico TH-2770Z Transistor Radio (image retrieved from http://antiqueradio.org/). (b) Ettus Research N210 software defined radio.

Modern SDR receivers employ an analog-to-digital converter (ADC) followed by some form of digital signal processing hardware such as a microprocessor, FPGA, or general purpose computer. SDRs typically keep information bearing signals in the digital domain as long as possible, thus allowing for software implementations of most of the radio functionality. Further, digital signal processing techniques make possible functionality that is otherwise difficult or impossible using analog techniques.

Figure 2 shows a generalized architecture for a modern SDR transceiver. In both the SDR receiver and transmitter, flexible RF hardware provides an interface between the ADC/DAC and the antenna(s). This RF hardware is typically software controllable. Many current receiver architectures use a digital down-conversion (DDC) stage (e.g., from an intermediate frequency to baseband) prior to the baseband processor. Similarly, SDR transmitters include a digital signal processor, digital up-conversion (DUC), and a digital-to-analog converter (DAC). The digital signal processor is fully programmable and therefore can be programmed to implement tasks such as signal filtering, modulation and demodulation, encoding and decoding, and equalization.

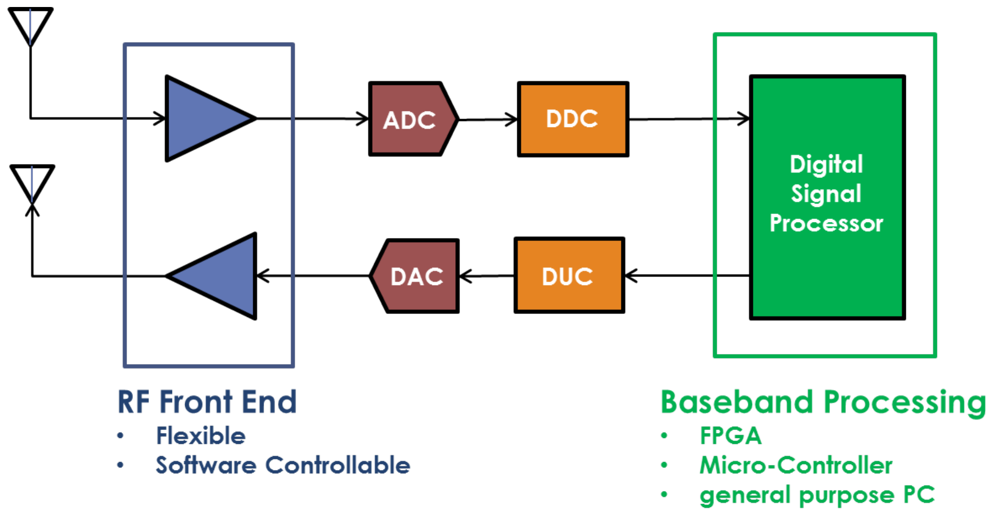


Figure 2: Block diagram of a modern SDR transceiver. The top row of system blocks correspond to the SDR receiver. The bottom row of system blocks correspond to the SDR transmitter.

2.2 History

One of the earliest examples of a software-defined radio was the SpeakEasy transceiver [1] which was developed for the U.S. Military under the Multiband Multimode Radio Program. The primary motivation for its development was to create a single radio system that could interoperate with any one of the 10 or more existing communication systems used for tactical military communications. The system was agile in carrier frequency (operating anywhere between 2 MHz and 2000 MHz), modulation technique, and waveform type, all of which were programmable through software. Development of the SpeakEasy system began in 1991 and was first demonstrated in 1994.

The earliest publication on the topic of software defined radio was in an article published at the 1992 National Telesystems Conference [2]. The article is often cited as the first to use the term “software radio” and its author, Joe Mitola III, is commonly referred to as the “godfather” of software-defined radio.

2.3 Mathematical Model

The operation of a software-defined radio can be understood through analysis of simplified block diagrams. In the following section we present a model for an SDR receiver. A corresponding model for an SDR transmitter will be discussed in a future lab.

2.3.1 Receiver

Figure 3 shows a block diagram model for an SDR Receiver. In the model, $s(t)$ represents the RF signal impinging the receive antenna. Multiplication of $s(t)$ by a complex exponential represents the frequency down-conversion process, which is presented in a single operation for simplicity. In most SDR hardware, the down-conversion process is implemented through the combination of analog hardware in the RF front end and software algorithms executed within the digital signal processing hardware. The low-pass filter, modeled as the linear time invariant system having impulse response $h_{LP}(t)$, removes images that result from the down-conversion process.

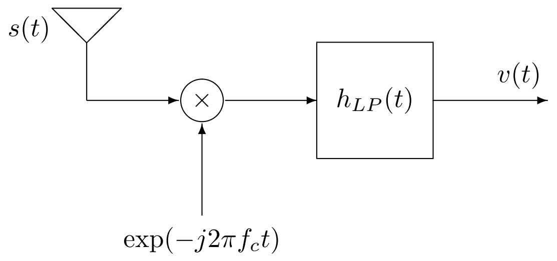


Figure 3: Block diagram model of an SDR receiver.

The sequence of plots in Figure 4 show a frequency-domain analysis of the SDR receiver model. Note that the time-domain signals $s(t)$ and $v(t)$ have Fourier transforms $S(f)$ and $V(f)$, respectively. $H_{LP}(f)$ denotes the frequency response of the lowpass filter. The notation $\mathcal{F}\{\cdot\}$ is used to denote the Fourier transform operation. For simplicity, all frequency domain signals are assumed to be strictly real-valued.

The analysis begins with a real-valued signal $s(t)$. Recall that for any real-valued time-domain signal $s(t)$, Fourier transform theory tells us that its spectrum $S(f)$ must exhibit conjugate symmetry

$$S(f) = S^*(-f) \tag{1}$$

which then implies

$$\begin{aligned} \Re\{S(f)\} &= \Re\{S(-f)\} && (2)\\ \Im\{S(f)\} &= -\Im\{S(-f)\} && (3)\\ |S(f)| &= |S(-f)| && (4)\\ \angle S(f) &= -\angle S(-f) && (5) \end{aligned}$$

where $\Re\{\cdot\}$ and $\Im\{\cdot\}$ denote the real and imaginary parts, respectively. Notice that $S(f)$ in Figure 4 exhibits conjugate symmetry (actually, the $S(f)$ shown in Figure 4 exhibits stronger symmetry since the plots are assumed to be strictly real-valued functions). However, notice that the final signal $V(f)$ does not have conjugate symmetry, and therefore the time-domain function $v(t)$ is complex-valued.

This result may seem somewhat counter-intuitive. After all, aren't all physical signals real-valued? The key to understanding the result is recognizing that the complex notation is used to represent a physical process occurring within the SDR. Most SDRs perform "I/Q" demodulation, as depicted in Figure 5. The SDR outputs sampled versions of the waveforms $i(t)$ and $q(t)$, both of which are real-valued. The combination of $i(t)$ and $q(t)$, known as the in-phase and quadrature signals, respectively, are what form the complex valued signal $v(t)$. The equivalence of the models shown in Figures 3 and 5 will be discussed throughout these SDR laboratories.

The key point here is that the SDR receiver hardware is capable of capturing a portion of the RF spectrum centered about frequency f_c, translating it to baseband, and then providing samples of that signal for further processing. The sampled signal contains all of the message information contained within the RF signal.

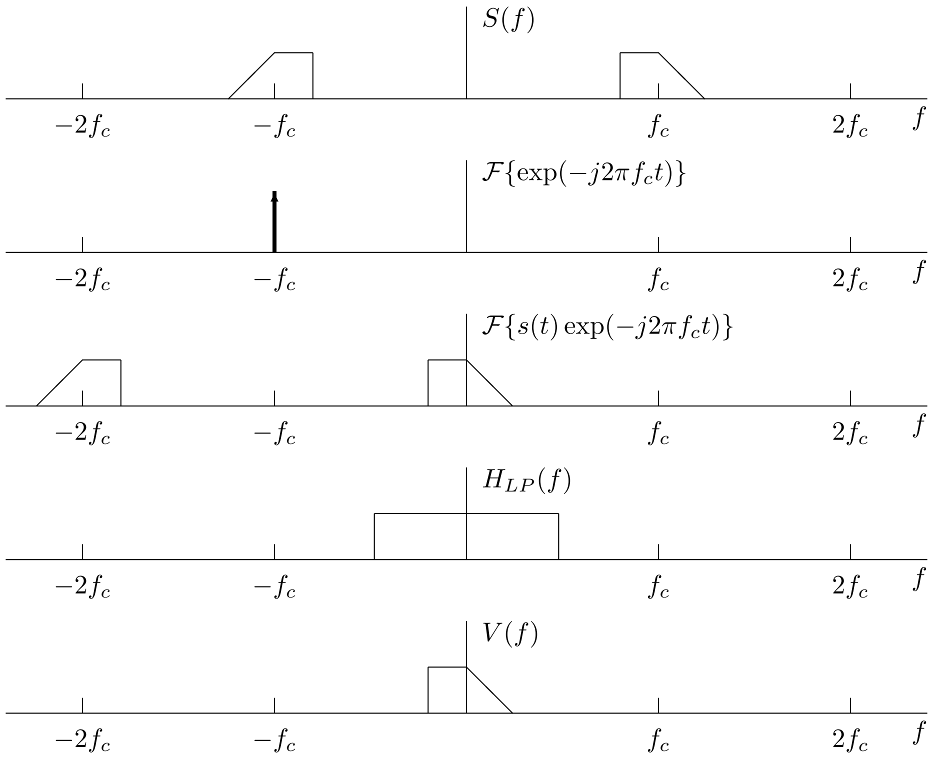


Figure 4: Frequency domain plots of signals shown in SDR receiver block diagram.

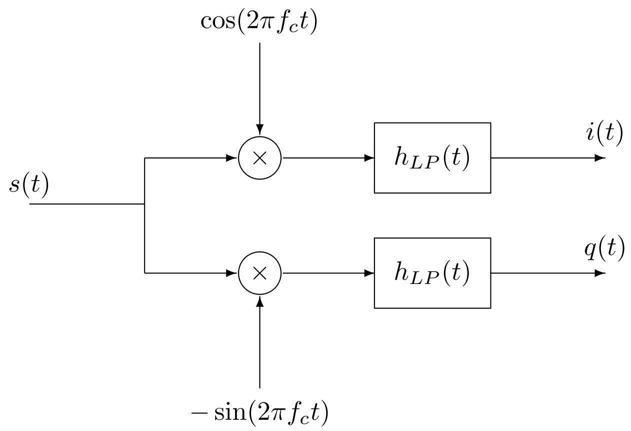


Figure 5: Block diagram model of I/Q demodulation used in an SDR receiver.

3 Laboratory SDR Hardware and Installation

The past decade has brought about significant advancements in both SDR hardware and software systems. In particular, low-cost SDR technology is now widely available for academic, commercial, and general consumer communities. In this section we briefly describe three examples of common SDR hardware. Installation procedures are provided. Any of these devices are suitable for carrying out the exercises in this laboratory[1].

Our discussion of SDR device parameters and specifications is limited to only those that are essential for this laboratory:

1. **Frequency Coverage** - range of radio frequencies that can be tuned to by the device. We typically use f_c to denote the frequency the SDR is tuned to.

2. **Instantaneous Bandwidth** - span of frequencies, centered about f_c, that can be captured by the SDR at once. For an instantaneous bandwidth B, an SDR receiver tuned to f_c would capture RF signals in the band $f_c - B/2$ to $f_c + B/2$.

Web links with more detailed information for each device are provided, and the interested reader is highly encouraged to consult those references.

3.1 RTL-SDR

The RTL-SDR dongle is a very low-cost receive-only SDR. There are numerous variants of the device produced by many different manufacturers, all of which are based on the Realtek RTL2832U integrated circuit. It communicates with a host PC over a USB 2.0 interface. The RTL-SDR was originally produced as a DVB-TV tuner, but some clever folks realized that it could instead be used as a SDR receiver via a driver modification. The device can be used on Windows, Linux, and Mac OS X operating systems.

- **Frequency Coverage**: 24MHz to 1.8GHz (for R820T RF front end)[2]
- **Instantaneous Bandwidth**: up to 2.4MHz [3]

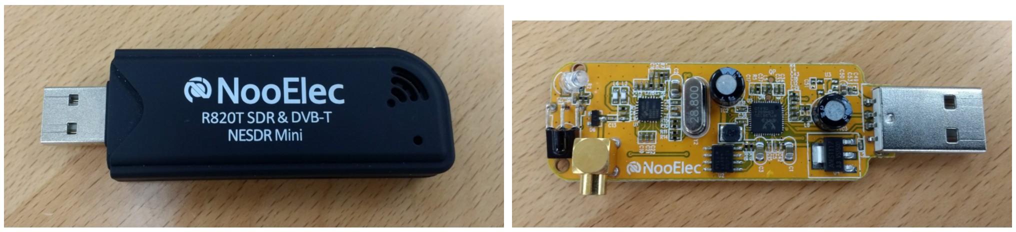


Figure 6: RTL-SDR USB dongle.

More detailed information on the RTL-SDR can be found in references [3] and [4]. MATLAB and Simulink documentation is available at `https://www.mathworks.com/hardware-support/rtl-sdr.html`.

[1]Later laboratory exercises require a transmit-capable SDR such as the ADALM-PLUTO or USRP. The RTL-SDR is a receive-only device.

[2]Variants of the RTL-SDR use different RF front ends with different tuning ranges.

[3]The device is capable of bandwidths up to 3.2MHz, however most users experience data losses (i.e., dropped ADC samples) for rates greater than 2.4MHz.

3.1.1 Installation in MATLAB/Simulink

Support for the RTL-SDR is provided in MATLAB 2013b or later. To install:

1. Navigate to the `Home` tab on the MATLAB Toolstrip, then to `Add-Ons -> Get Hardware Support Packages`.[4]
2. Search for "RTL-SDR" and click on the "**Communications System Toolbox Support Package for RTL-SDR Radio**" option.
3. Click the option to "Install"
4. You will need to log in to a Mathworks account in order to proceed. You can create one, if necessary.
5. You will be prompted to agree to various licensing agreements. Continue with the installation.

When the installation is complete, you should receive notification that "Your Hardware Support Package requires configuration." If you have access to a USRP radio, choose "Setup Now" and continue with the installation. Otherwise you can resume configuration at a later time by typing `targetupdater` at the MATLAB command prompt.

3.1.2 Verifying the RTL-SDR Installation

Connect an RTL-SDR device to your laptop. It may take Windows a few minutes to recognize the hardware and configure the necessary drivers.

To verify the MATLAB configuration and communication with the RTL-SDR, type `sdrinfo` at the MATLAB prompt. After a few moments, you should see a message beginning with

```
      RadioName:   'Generic RTL2832U OEM'
   RadioAddress:   '0'
```

followed by several additional fields of information. Note that the `RadioName` entry will may vary depending on your specific RTL-SDR variant.

3.2 ADALM-PLUTO

The ADALM-PLUTO (PlutoSDR) provides a high-performance software-radio transceiver that was designed specifically for students. It allows simultaneous transmit and receive operations. The PlutoSDR communicates with a host PC through a USB 2.0 interface. It utilizes the `libiio` drivers, which are supported on Windows, Linux, and Mac OS X operating systems.

- **Frequency Coverage**: 325MHz to 3.8GHz [5]
- **Instantaneous Bandwidth**: up to 20MHz

More detailed information on the PlutoSDR can be found in reference [5]. MATLAB and Simulink documentation is available at `https://www.mathworks.com/hardware-support/adalm-pluto-radio.html`.

3.2.1 Installation in MATLAB/Simulink

Support for the PlutoSDR is provided in MATLAB 2017a or later. To install:

1. Navigate to the `Home` tab on the MATLAB Toolstrip, then to `Add-Ons -> Get Hardware Support Packages`.

[4]Alternatively, from the MATLAB prompt, launch the support package installer by typing `supportPackageInstaller`.
[5]The MATLAB/Simulink support package extends the frequency coverage to 70MHz to 6GHz.

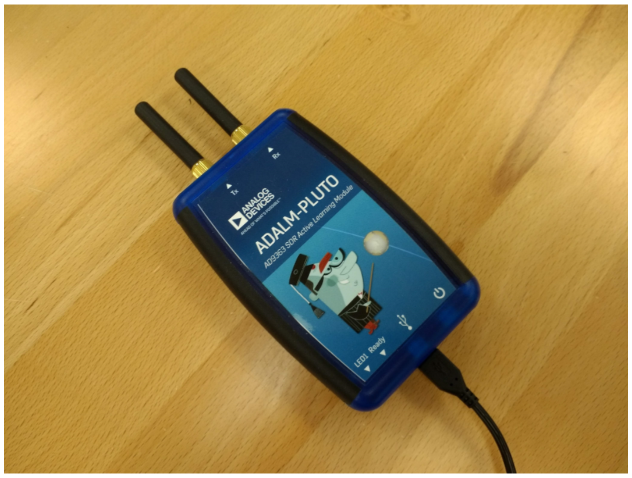


Figure 7: ADALM-PLUTO SDR.

2. Search for "Pluto" and choose the "**Communications System Toolbox Support Package for Analog Devices ADALM-Pluto Radio**" option.

3. Click the option to "Install"

4. You will need to log in to a Mathworks account in order to proceed. You can create one, if necessary.

5. You will be prompted to agree to various licensing agreements. Continue with the installation.

When the installation is complete, you should receive notification that "Your Hardware Support Package requires configuration." If you have access to a PlutoSDR, choose "Setup Now" and continue with the installation. Otherwise you can resume configuration at a later time by typing `targetupdater` at the MATLAB command prompt.

3.2.2 Verifying the PlutoSDR Installation

Connect a PlutoSDR device to your laptop. It may take Windows a few minutes to recognize the hardware and configure the necessary drivers.

To verify the MATLAB configuration and communication with the PlutoSDR, type `findPlutoRadio` at the MATLAB prompt. After a few moments, you should see a message beginning with

```
      RadioID:   'usb:0'
    SerialNum:   '1000002355237309001600130902161O7d'
```

where the `SerialNum` entry will vary depending on your specific PlutoSDR.

4 Spectrum Analyzer

We will now create a spectrum analyzer using the SDR hardware as a data source. Any of the SDR devices previously described can be utilized, with only minor changes in configuration.

4.1 Creating the Simulink Model using RTL-SDR

Let's construct a Simulink model that utilizes the RTL-SDR device. We will configure the model so that center frequency can be changed to any band of interest within the SDRs operating range.

1. Open a new Simulink model and create the model shown in Figure 8. The specific components you will need are:

   - **RTL-SDR Receiver** (found in `Communications System Toolbox Support Package for RTL-SDR Radio`)
   - **Spectrum Analyzer** (found in `DSP System Toolbox` → `Sinks`)
   - **Constant** (found in `Simulink` → `Sources`)

   Hint: You can construct models more quickly by clicking in the white space of the model and typing the name of block you would like to add. For example, to add a **Spectrum Analyzer** block, start typing "Spectrum", and you will see a list of blocks that have "Spectrum" in the name. You can then choose the **Spectrum Analyzer** block.

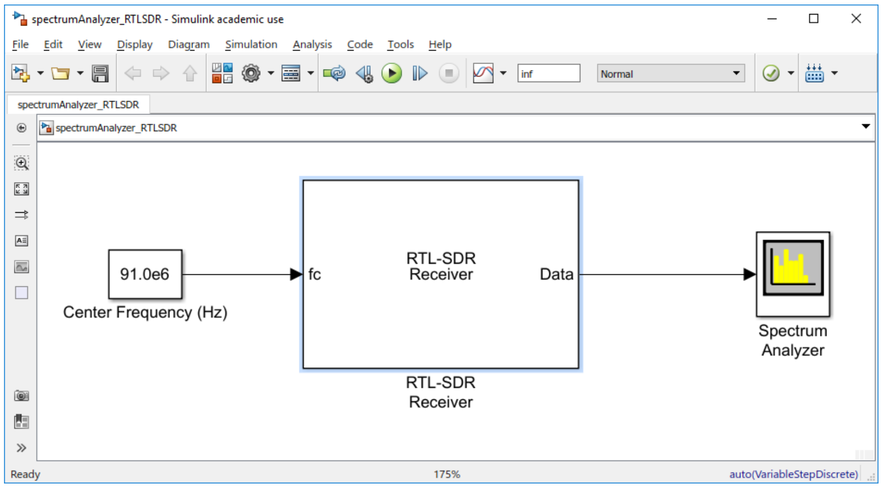

spectrumAnalyzer_RTLSDR - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
Normal
spectrumAnalyzer_RTLSDR
91.0e6
Center Frequency (Hz)
fc
RTL-SDR Receiver
Data
RTL-SDR Receiver
Spectrum Analyzer
Ready
175%
auto(VariableStepDiscrete)

Figure 8: Simulink model for the Spectrum Analyzer.

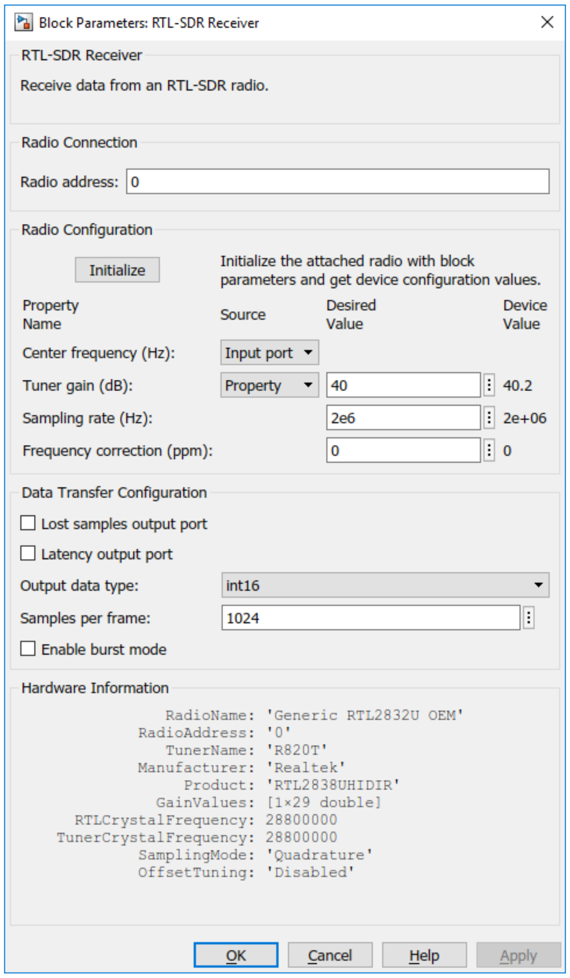
2. Set the Simulation stop time to `inf`.

3. Configure the **RTL-SDR Receiver** block as shown in Figure 9. The key settings and their meaning are as follows:

   - **Center frequency (Hz):** controls the center frequency of the SDR hardware. Selecting `Input port` allows the setting to be easily adjusted while the Simulink model is executing. Set the corresponding `Constant` block to `91.0e6`.
   - **Tuner Gain (dB):** controls the overall gain of the receiver. Set this paramter to `40`.
   - **Sampling Rate (Hz):** controls the sample rate of data delivered to the PC over the USB bus. This rate corresponds to the instantaneous bandwidth of the receiver. Set this parameter to `2e6`[6].

[6]The RTL-SDR device can operate at any sample rate in the range 225001 to 300000, or 900001 to 3200000.

- **Samples per frame:** specifies the length of each data frame that will be processed by Simulink. The choice 1024 should work reasonably well for this example.


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):	Input port		
Tuner gain (dB):	Property	40	40.2
Sampling rate (Hz):		2e6	2e+06
Frequency correction (ppm):		0	0

Data Transfer Configuration

- [ ] Lost samples output port
- [ ] Latency output port

Output data type: int16

Samples per frame: 1024

- [ ] Enable burst mode

Hardware Information

```
            RadioName: 'Generic RTL2832U OEM'
         RadioAddress: '0'
            TunerName: 'R820T'
         Manufacturer: 'Realtek'
              Product: 'RTL2838UHIDIR'
           GainValues: [1×29 double]
  RTLCrystalFrequency: 28800000
TunerCrystalFrequency: 28800000
         SamplingMode: 'Quadrature'
         OffsetTuning: 'Disabled'
```

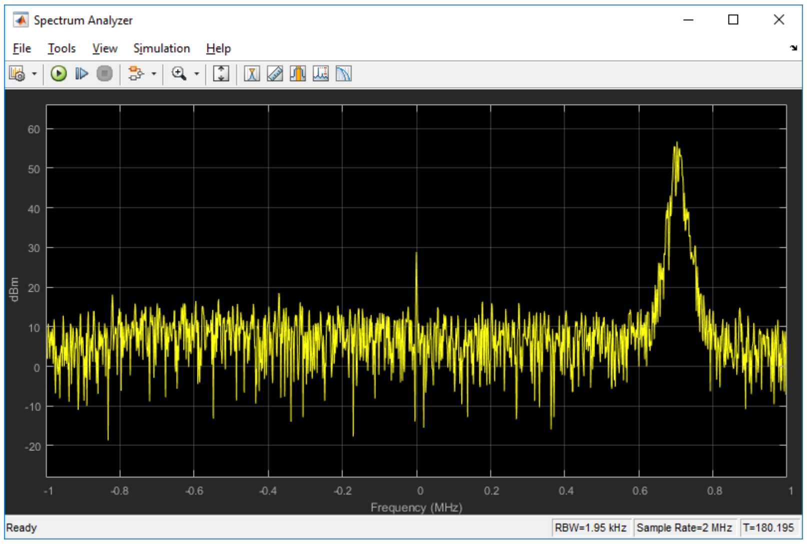
Figure 9: RTL-SDR block configuration for Spectrum Analyzer.

4. Begin execution by clicking `Run` (the play button). After a few moments you should see a plot in the Spectrum Anaylzer window similar to Figure 10(a). This plot is the *real-time* spectrum of the 2 MHz wide RF band centered at 91.0 MHz. On the Frequency axis, the 0 MHz location corresponds to the center frequency of the SDR. Notice the strong peak on the plot at 0.7 MHz, which corresponds to 91.7 MHz. This is a local radio station!

5. For visualization purposes, it is sometimes helpful to reduce noise by having the Spectrum Analyzer window average many traces together. To do so, select `View` → `Spectrum Settings`, and then set the `Averages` parameter to a number greater than 1. An example is shown in Figure 10(b).

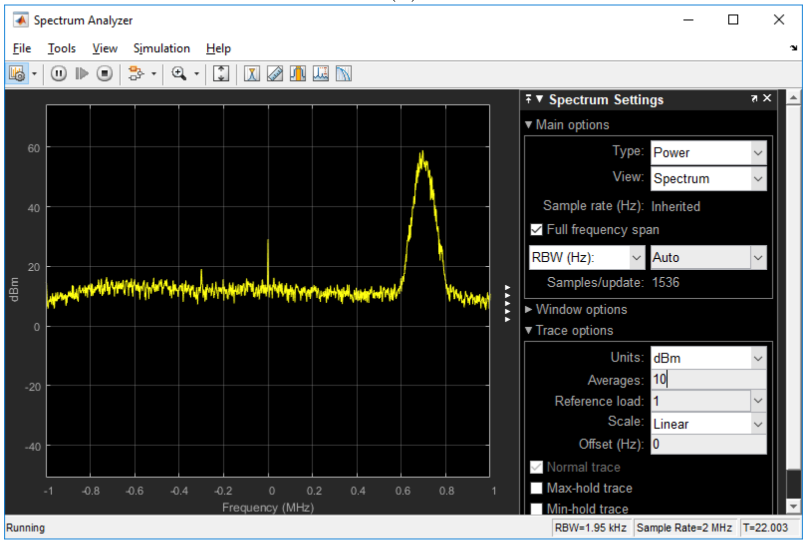
6. Experiment with the configuration of the RTL-SDR receiver block.

   - What happens to the spectrum plot when you adjust the **Center frequency** parameter?
   - What happens when you adjust the **Tuner gain** parameter?
   - What happens when you adjust the **Sampling rate** parameter? *Hint:* Be sure the **Full Frequency span** box is checked in the Spectrum Settings panel of the Spectrum Analyzer.



RBW=1.95 kHz | Sample Rate=2 MHz | T=180.195

(a)



Type: Power
View: Spectrum
Sample rate (Hz): Inherited
☑ Full frequency span
RBW (Hz): Auto
Samples/update: 1536
Units: dBm
Averages: 10
Reference load: 1
Scale: Linear
Offset (Hz): 0
☑ Normal trace
☐ Max-hold trace
☐ Min-hold trace

RBW=1.95 kHz | Sample Rate=2 MHz | T=22.003

(b)

Figure 10: Spectrum Analyzer plots with center frequency setting of 91 MHz. (a) Default. (b) With averaging.

4.2 Creating the Simulink Model using ADALM-PLUTO SDR

Let's construct a Simulink model that utilizes the ADALM-PLUTO SDR device. We will configure the model so that center frequency can be changed to any band of interest within the SDRs operating range.

1. Open a new Simulink model and create the model shown in Figure 11. The specific components you will need are:
   - **ADALM-Pluto Radio Receiver** (found in `Communications System Toolbox Support Package for ADALM-PLUTO Radio`)
   - **Spectrum Analyzer** (found in `DSP System Toolbox` → `Sinks`)
   - **Constant** (found in `Simulink` → `Sources`)

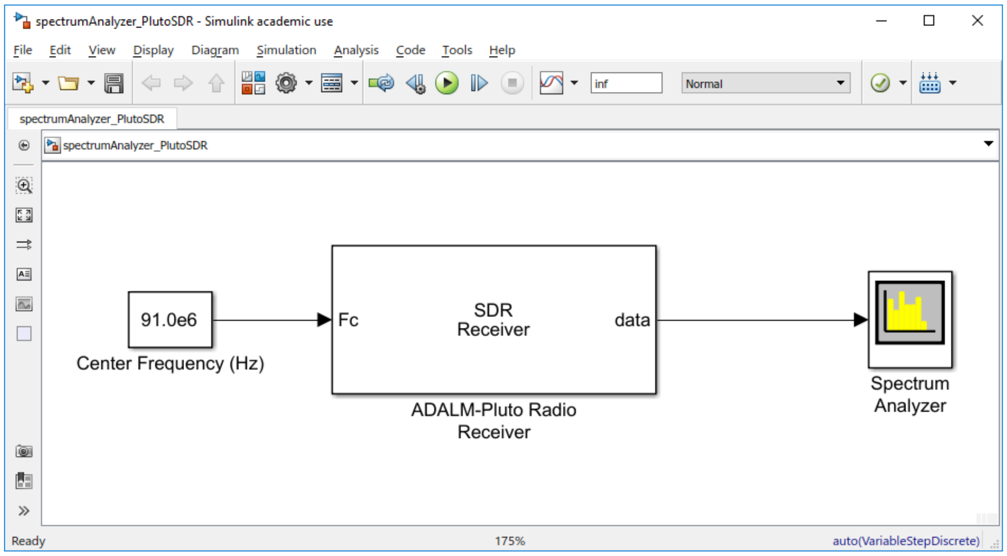
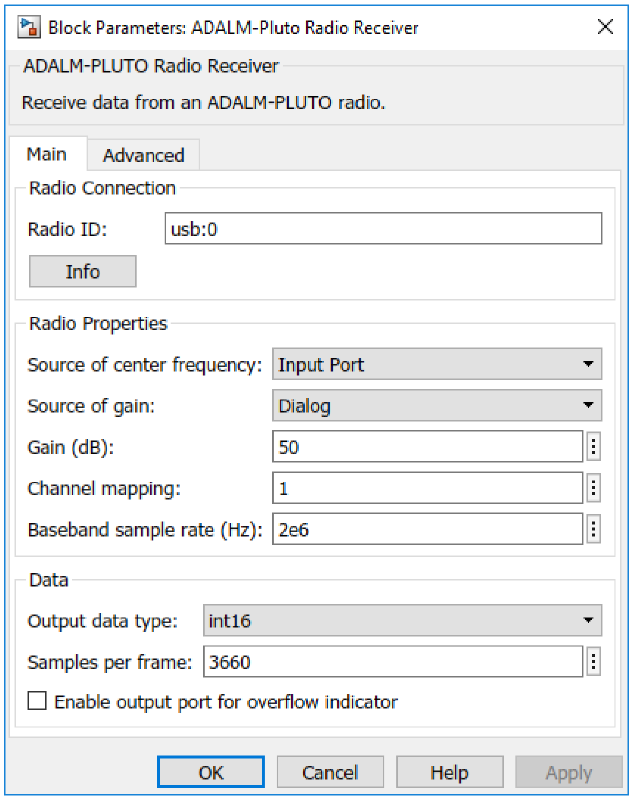

spectrumAnalyzer_PlutoSDR - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
Normal
spectrumAnalyzer_PlutoSDR
spectrumAnalyzer_PlutoSDR
91.0e6
Center Frequency (Hz)
Fc
SDR Receiver
data
ADALM-Pluto Radio Receiver
Spectrum Analyzer
Ready
175%
auto(VariableStepDiscrete)

Figure 11: Simulink model for the Spectrum Analyzer.

2. Set the Simulation stop time to `inf`.

3. Configure the **ADALM-PLUTO Radio Receiver** block as shown in Figure 12. The key settings and their meaning are as follows:
   - **Center frequency (Hz) / Source of center frequency:** controls the center frequency of the SDR hardware. Selecting `Input Port` allows the setting to be easily adjusted while the Simulink model is executing. Set the corresponding `Constant` block to `91.0e6`.
   - **Gain (dB):** controls the overall gain of the receiver. Set this paramter to `50`.
   - **Baseband sample rate (Hz):** controls the sample rate of data delivered to the PC over the USB bus. This rate corresponds to the instantaneous bandwidth of the receiver. Set this parameter to `2e6`.
   - **Samples per frame:** specifies the length of each data frame that will be processed by Simulink. The choice 3660 should work reasonably well for this example.

4. Begin execution by clicking `Run` (the play button). After a few moments you should see a plot in the Spectrum Anaylzer window similar to Figure 10(a). This plot is the *real-time* spectrum of the 2 MHz wide RF band centered at 91.0 MHz. On the Frequency axis, the 0 MHz location corresponds to the center frequency of the SDR. Notice the strong peak on the plot at 0.7 MHz, which corresponds to 91.7 MHz. This is a local radio station!



Block Parameters: ADALM-Pluto Radio Receiver

ADALM-PLUTO Radio Receiver

Receive data from an ADALM-PLUTO radio.

Radio Connection

Radio ID: usb:0

Radio Properties

Source of center frequency: Input Port

Source of gain: Dialog

Gain (dB): 50

Channel mapping: 1

Baseband sample rate (Hz): 2e6

Data

Output data type: int16

Samples per frame: 3660

☐ Enable output port for overflow indicator

Figure 12: ADALM-PLUTO Radio block configuration for Spectrum Analyzer.

5. For visualization purposes, it is sometimes helpful to reduce noise by having the Spectrum Analyzer window average many traces together. To do so, select `View` → `Spectrum Settings`, and then set the `Averages` parameter to a number greater than 1. An example is shown in Figure 10(b).

6. Experiment with the configuration of the PlutoSDR receiver block.

   - What happens to the spectrum plot when you adjust the **Center frequency** parameter?
   - What happens when you adjust the **Gain** parameter?
   - What happens when you adjust the **Baseband sample rate** parameter? *Hint:* Be sure the **Full Frequency span** box is checked in the Spectrum Settings panel of the Spectrum Analyzer.

5 Investigating Wireless Signals and the RF Spectrum

Experiment with the Spectrum Analyzer by viewing some of the many interesting signals that are literally surrounding you! Numerous real-world communication systems use frequencies within the operating range of the SDR hardware, such as

- FM Broadcast Radio (88-108 MHz)
- Personal transmitters such as automotive key fobs and garage door openers (315 MHz)
- Cell phones (various bands between 700-2500 MHz)
- GPS (1227 MHz and 1575 MHz)
- WiFi (2400 MHz)
- Bluetooth (2400 MHz)
- High Definition Television Broadcasts (UHF channels at 470-806 MHz)
- NEXRAD Weather Radar (2700 - 3000 MHz)
- Amateur and Mobile Radios (various bands)

There are many more! A simple web-search can often tell you the operating frequency of RF systems. Consult your instructor if you need assistance interpreting what you are finding and seeing.

As discussed in Section 1.1, usage of the radio frequency spectrum is regulated by the Federal Communications Commissions (FCC). Consumer wireless devices certified by the FCC should be labeled with an "FCC ID", which is a unique identifier. Using the FCC ID search you can determine the operating frequency of certified devices:
FCC ID Search: `https://www.fcc.gov/oet/ea/fccid`

Question 1: Identify a real-world radio frequency signal of your choosing and investigate it! You can choose any signal you wish, so long as its operating frequency is within the range of your SDR receiver. Feel free to consult with your instructor if you have doubts whether your preferred signal is appropriate for this exercise.

Submit the following for your signal:

- center frequency of the signal
- screen capture of the spectrum analyzer window showing the signal
- estimate of the signal bandwidth
- Describe any characteristics of the signal based on your research of the signal/system, or based on your observations of its spectrum. It is OK if you don't fully understand the details - we will learn more about these details in this course!

References

[1] R.I. Lackey and D.W. Upmal. Speakeasy: the military software radio. *Communications Magazine, IEEE*, 33(5):56–61, May 1995.

[2] J. Mitola. Software radios-survey, critical evaluation and future directions. In *Telesystems Conference, 1992. NTC-92., National*, pages 13/15–13/23, May 1992.

[3] rtlsdr.org wiki "History and Discovery of RTLSDR". http://rtlsdr.org/.

[4] "RTL-SDR and GNU Radio with Realtek RTL2832U". http://www.superkuh.com/rtlsdr.html.

[5] Analog Devices Inc. ADALM-PLUTO Software-Defined Radio Active Learning Module. http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html.

[6] Mathworks. RTL-SDR Support Package from Communications Toolbox. Online Resource. https://www.mathworks.com/hardware-support/rtl-sdr.html.

[7] Mathworks. ADALM-PLUTO Support Package from Communications Toolbox. Online Resource. https://www.mathworks.com/hardware-support/adalm-pluto-radio.html.

[8] Simon Haykin. *Communication Systems*. 4th edition, 2001.

[9] Leon W. Couch III. *Modern Communication Systems*. 1995.