Laboratory 4: Digital Communications

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 will assist students in understanding fundamental concepts of digital communication systems. The activities will focus on passband systems such as those used in modern wireless digital communications. Students will investigate various forms of digital modulation and examine the functional blocks required to realize functional systems. Simulation will be used to visualize digital communication waveforms, as well as verify theoretical system performance. Students utilize SDR hardware to explore actual wireless data transmissions and implement real-time digital communication systems.

0.2 Student Outcomes

Upon successful completion of this laboratory, the student will:

- Identify and characterize common forms of digital carrier modulation based on time-domain and frequency-domain measurements.
- Explain the ideal receiver structure for digital communication systems.
- Simulate the bit error performance of a digital communication system and compare to theoretical predictions.
- Explain the influence of noise and interference on the performance of a digital communication system.
- Explain the importance of carrier synchronization in a modern digital communication system.
- Explain the importance of symbol synchronization (timing recovery) in a modern digital communication system.
- Implement a digital communication systems using SDR hardware and verify its operation over a wireless channel.

1 Background

Appendix A provides a summary of mathematical representations for amplitude-shift keying (ASK), phase-shift keying (PSK), and frequency-shift keying (FSK) waveforms.

2 Observing Digital Carrier Modulated Waveforms

In this part of the laboratory you will use your personal SDR to observe digital carrier modulated waveforms. Your instructor will broadcast radio-frequency signals in the laboratory in the 902MHz to 928MHz frequency band.

The broadcast consists of multiple digital communication signals, each having a distinct carrier frequency and frequency band (i.e., the transmission uses frequency division multiplexing). The carriers are separated by approximately 30kHz. The communication signals are binary ASK, binary FSK, or binary PSK.

Using Simulink models from previous laboratory exercises, observe the communication signals in both the time-domain and frequency-domain.

Question 2.1: Take a screen capture of the Spectrum Analyzer window showing the instructor's broadcast signal.

Question 2.2: Identify and characterize each of the digital communication signals within your instructor's broadcast signal. Your submittal must include screen captures of Spectrum Analyzer and Time Scope windows for each signal. Identify the modulation used in each signal. Determine the bit rate for each signal, and indicate how the bit rate can be determined from both the time-domain and frequency-domain plots. Discuss, compare, and contrast the salient features seen in the plots.

Question 2.3: You should have found that one of communication signals in your instructor's broadcast utilized binary Amplitude Shift Keying (ASK). Because ASK is essentially an amplitude modulated large-carrier (AM-LC) waveform, it can be easily demodulated using non-coherent techniques. Using your AM-LC receiver from Lab 2, recover the actual baseband message. Submit a screen capture of the Simulink model and a Time-Scope window showing the message waveform.

3 Frequency Calibration

Many communication systems require synchronous operation between the transmitter and receiver. Coherent digital communication, which requires the carrier signals in the transmitter and receiver to have the same frequency and phase, is an example of such a system.

In this portion of the lab you will calibrate your personal SDR device to a real-world radio-frequency signal. The intent is to correct for any large frequency offset between your personal SDR and this reference signal. Such a calibration is often an important first step towards achieving synchronous operation. Further steps, which we will see in Section 5, are necessary to establish and maintain truly synchronous operation.

The primary source of frequency offsets is the accuracy and/or stability of the oscillators used in the SDR hardware. The frequency variation of an oscillator is usually specified in parts per million (ppm). The maximum frequency variation, Δf, measured in Hz is

$$\Delta f = \frac{f_c \times \texttt{ppm}}{10^6}$$

where f_c is the center frequency and `ppm` is the peak variation (expressed as $\pm$). For example, an oscillator specified to have a 2 `ppm` frequency accuracy operating at 100 MHz would have a peak deviation of 200 Hz. Therefore, the actual oscillator frequency would be somewhere in the range 99.9998 MHz to 100.0002 MHz.

You will calibrate the frequency of your personal SDR device using real-world signals of opportunity. In particular, we will use the pilot tone from local over-the-air (OTA) digital television broadcasts. According to the American Television Systems Committee (ATSC) standards [1], digital television broadcasts in the United States must include a pilot tone 309440.559 Hz above the lower edge of the transmission band. The following table lists several OTA digital television broadcasts in the Milwaukee area[1]:

TV Channel	RF Channel	Lower Band Edge (MHz)	ATSC Pilot (MHz)	Upper Band Edge (MHz)
6 (FOX)	33 (UHF)	584	584.309440559	590
10, 36 (PBS)	36 (UHF)	602	602.309440559	608
58 (CBS)	46 (UHF)	662	662.309440559	668

Figure 1 shows a spectrum analyzer capture of the digital television broadcast of the local PBS station in Milwaukee. Note that the SDR receiver was tuned to the lower band edge of the transmission and that the instantaneous bandwidth is 16MHz. Therefore, the entire digital television broadcast is present in the positive frequencies of the capture. The strong peak is the ATSC Pilot signal.

We will calibrate your SDR device to the ATSC Pilot in RF Channel 36, which is at 602.309440559 MHz. To calibrate:

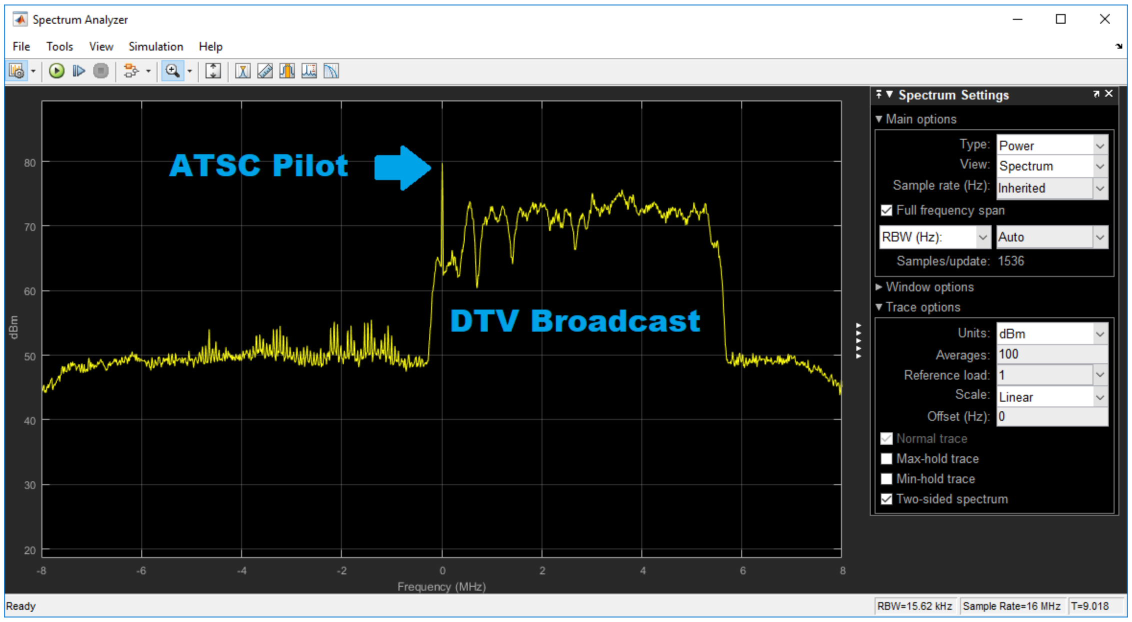
1. Begin with the Spectrum Analyzer model from earlier labs. Configure for an instantaneous bandwidth of 240e3 Hz.

2. Tune the center frequency to 602.3 MHz. This will place the ATSC pilot near 10kHz in the Spectrum Analyzer window.

3. Measure the frequency of the ATSC Pilot. Some hints:

   - Use the `Zoom In` feature to zoom the spectrum analyzer to the location of the ATSC Pilot.
   - In the `Spectrum Settings -> Trace options` panel, increase the number of `Averages`.

[1]You can find other broadcasts in your area using the search tools at https://otadtv.com/



Type: Power
View: Spectrum
Sample rate (Hz): Inherited
☑ Full frequency span
RBW (Hz): Auto
Samples/update: 1536

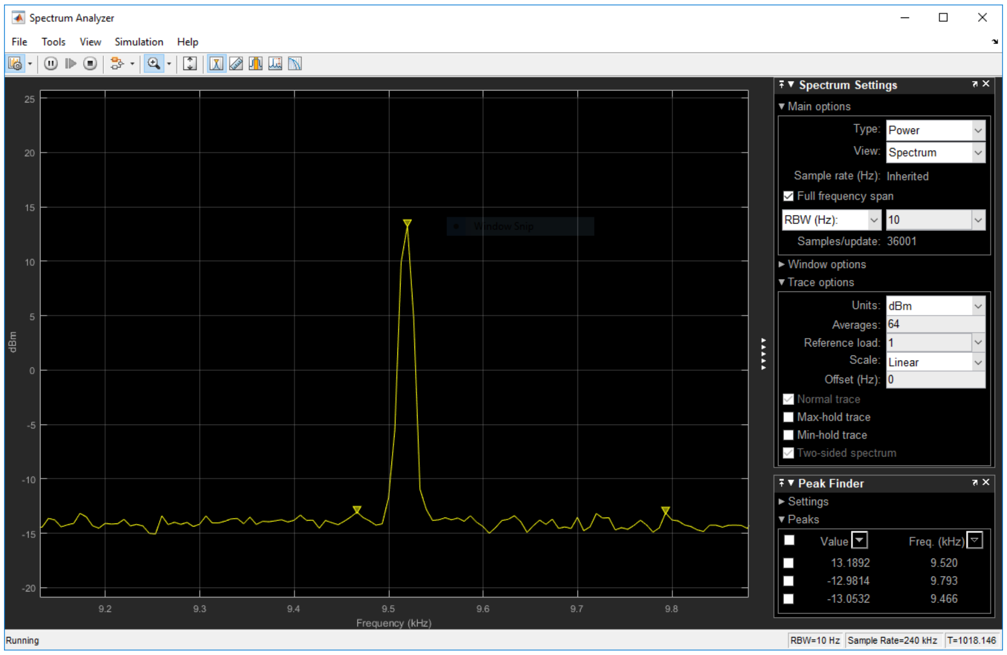
Units: dBm
Averages: 100
Reference load: 1
Scale: Linear
Offset (Hz): 0
☑ Normal trace
☐ Max-hold trace
☐ Min-hold trace
☑ Two-sided spectrum

Ready | RBW=15.62 kHz | Sample Rate=16 MHz | T=9.018

Figure 1: Example spectrum analyzer window showing digital television broadcast Channel 36 (602MHz). Receiver tuned to 602MHz and instantaneous bandwidth of 16MHz.

- In the `Spectrum Settings -> Main options` panel, modify the `RBW (Hz)` parameter to 10. RBW stands for Resolution Bandwidth, which defines the smallest positive frequency that can be resolved by the spectrum analyzer. By default, the RBW setting of `AUTO` provides a resolution of the instantaneous bandwidth divided by 1024. For the suggested 240e3 Hz bandwidth, this gives RBW $= 240\text{e}3/1024 = 234.37$ Hz. Changing RBW to 10Hz will make the ATSC pilot peak more distinct, especially when zoomed in to a small frequency range. Be patient with the Spectrum Analzyer window, as a smaller RBW requires that it process more samples.
- Enable the `Peak Finder` option, which will automatically detect peaks in the spectrum and display the corresponding frequencies.

An example spectrum analyzer window is shown in Figure 2.



Type: Power
View: Spectrum
Sample rate (Hz): Inherited
☑ Full frequency span
RBW (Hz): 10
Samples/update: 36001

Units: dBm
Averages: 64
Reference load: 1
Scale: Linear
Offset (Hz): 0
☑ Normal trace
☐ Max-hold trace
☐ Min-hold trace
☑ Two-sided spectrum

Value	Freq. (kHz)
13.1892	9.520
-12.9814	9.793
-13.0532	9.466

Running | RBW=10 Hz | Sample Rate=240 kHz | T=1018.146

Figure 2: Example spectrum analyzer window showing measurement of ATSC Pilot frequency.

Question 3.1: Submit a screen capture of the Spectrum Analyzer window showing the full 240e3 Hz bandwidth. Submit a second screen capture of the Spectrum Analyzer window after zooming into the ATSC Pilot.

Question 3.2: What is the offset of your SDR receiver relative to the ATSC Pilot? Give your answer in Hz and in ppm.

Question 3.3: Using the ppm offset you calculated in the previous problem, what is the frequency offset in Hz when your SDR device is tuned to 900MHz?

Question 3.4: Repeat the frequency calibration process for the ATSC Pilot in RF Channel 46. What is the offset of your SDR receiver relative to this ATSC Pilot? Give your answer in Hz and in ppm.

Note: You may find different results when calibrating to different real-world signals. In our case, the oscillators used to generate ATSC pilot frequencies also have errors. A high precision source would be needed for an *absolute* calibration.

4 Bit Error Rate Simulation for Binary Digital Transceiver

During lecture we examined a Simulink model that simulated a binary phase-shift keying (BPSK) communication system. You will now extend this example by simulating its bit error performance and compare the results to the theoretical prediction for probability of bit error of a BPSK system.

Recall from lecture that for equiprobable symbols, the probability of symbol error (or equivalently, the probability of bit error) for coherent BPSK is

$$P_e = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

where E_b is the average energy per bit, $Q(\cdot)$ is the Q-function, and we have assumed that the received waveform is corrupted by an additive white Gaussian noise process with zero mean and power spectral density $N_0/2$. Note that this result assumes the optimum receiver structure discussed during lecture.

Figure 3 gives a MATLAB script for calculating and plotting the bit error probability for BPSK as a function of E_b/N_0.

```
%% example_plot_Pb_vs_EbNo_BPSK.m
%   Example script that plots bit error probability curve
%    for BPSK.
%
%   Cory J. Prust, Ph.D.
%   Last Modified:  7/18/2018

clear all
close all

%% set range for Eb/No values
EbN0_dB = 0:0.1:12;
EbN0 = 10.^(EbN0_dB/10);  % convert to linear units

%% compute probability of bit error
Pe = qfunc(sqrt(2*EbN0));

%% plot
semilogy(EbN0_dB, Pe)
grid on
xlabel('E_b/N_0 (dB)')
ylabel('probability of bit error')
```

Figure 3: MATLAB script for plotting theoretical bit error probability for BPSK.

Construct the BPSK Simulink model shown in Figure 4. Open a new Simulink model and construct the model as follows:

- The **Bernoulli Binary Generator** block generates random binary numbers using a Bernoulli distribution. This block acts as our message source in the simulation. Set the `Sample Time` to 0.001, which gives a bit rate of 1000 bits per second. Set the `Samples per frame` to 1000, which specifies that the model will operate using vectors (or frames) containing 1000 bits[2].
- The **BPSK Modulator Baseband** block applies BPSK modulation to the message bits, producing the baseband representation of the communication signal.

[2]To learn more about how Simulink uses frames, see the documentation at
https://www.mathworks.com/help/dsp/ug/sample-and-frame-based-concepts.html and
https://www.mathworks.com/help/dsp/ug/inspect-sample-and-frame-rates-in-simulink.html

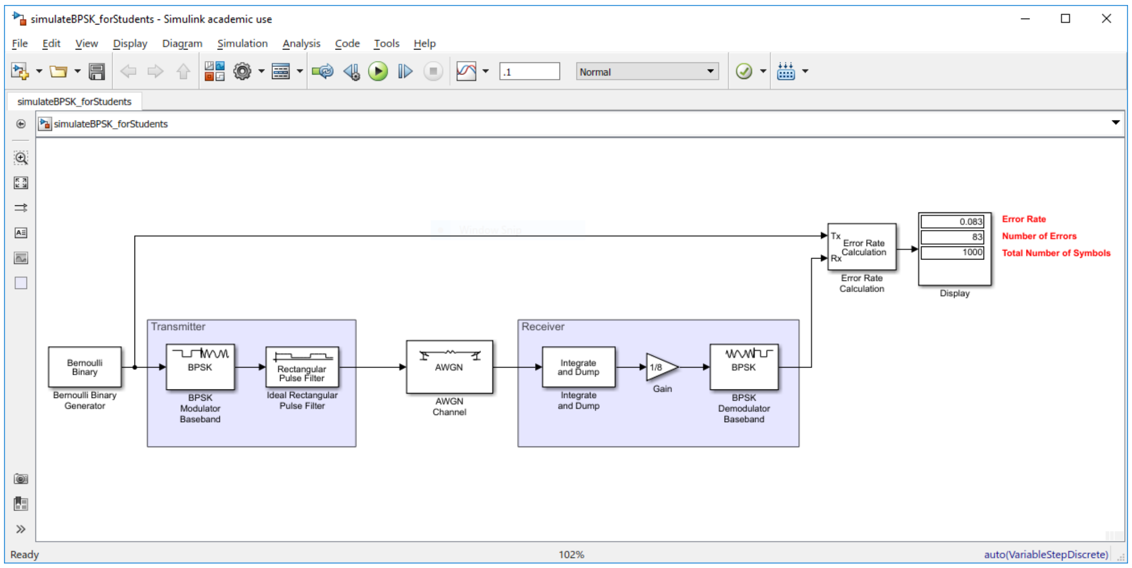


Figure 4: Simulink model for BPSK communication.

- The **Ideal Rectangular Pulse Filter** translates the baseband symbols into rectangular pulse shapes. Set the `Pulse length` to 8, which will produce a rectangular pulse 8 samples long for each BPSK symbol. Therefore, the block increases the signal sampling rate by a factor of 8. Ensure the `Input processing` is configured for `Columns as frames` and `Rate options` is configured for `Enforce single-rate processing`.

- The **AWGN Channel** block adds white Gaussian noise to the input signal, thus modeling the effects of a noisy communication channel. Set the `Symbol period` to 1/1000. Set the `Eb/No` parameter to 15.

- The **Integrate and Dump** block implements the ideal correlation receiver, or equivalently the ideal matched filter, for rectangular pulse shapes. Set the `Integration period` to 8 samples, which configures the block to integrate each received pulse (8 samples long) and produce an output value corresponding to each pulse.

- The **Gain** block, set to 1/8 normalizes the output value to ± 1 (without noise).

- The **BPSK Demodulator Baseband** block demodulates the input using BPSK. For each input value, the block decides which BPSK was sent and outputs the corresponding binary number.

- The **Error Rate Calculation** block compares the message data at the input to the transmitter to the received message data at the output of the receiver. Set `Output data` to `Port` and attach to a **Display** block. This configuration will display the error rate, the number of errors, and the total of symbols that were compared.

Experiment with the model by adding **Time Scope**, **Spectrum Analyzer**, **Eye Diagram**, and **Constellation Diagram** blocks. Be sure you understand how the model operates before proceeding.

Hint: You may find it helpful to change the **Time Scope** to a "stem" plot by selecting `View -> Style` and changing the `Plot type` setting.

Question 4.1: Add a **Time Scope** block and display the output of the **Ideal Rectangular Pulse Filter** block. What is its average signal power?

Question 4.2: Add a **Time Scope** block and display the input to the **AWGN Channel** block, the output of the **AWGN Channel** block, and the output of the **Gain** block. In the **AWGN Channel** block, set the `Input signal power` to the value you calculated in the previous problem and set `Eb/N0` to 12dB. Set the remaining parameters based on your understanding of the simulation. Run the simulation, and take a screen capture of the **Time Scope** block. Explain and interpret the plots.

Question 4.3: Add **Eye Diagram** and **Constellation Diagram** blocks to the signal at the input of the **BPSK Demodulator Baseband** block. Set the **AWGN Channel** block for `Eb/N0` of 12dB. Run the simulation, and take screen captures of the eye diagram and signal constellation. Repeat for `Eb/N0` of 6dB. Explain and interpret the plots. Comment on how the plots change as a function of E_b/N_0.

Question 4.4: You will now use the model to simulate the system performance over a range of `Eb/N0` values.

- Run the simulation for `Eb/N0` values of 0dB to 12dB in 1dB steps.
- Set the simulation time to ensure the results are meaningful. This is especially important when very few bit errors are expected.
- Observe and record the error rate for each run.
- Plot bit error rate vs E_b/N_0 and compare to the theoretical prediction. Comment on the results.

5 BPSK Receiver - Carrier Synchronization and Symbol Timing Recovery

The simulation in Section 4 uses a baseband representation of a BPSK communication system. That is, the process of modulating the message information onto a high frequency sinusoidal carrier, as is typically required in wireless communication, was omitted from our simulation. As we have seen in lecture, the frequency upconversion process (translating the baseband message to a high-frequency carrier) done in the transmitter can be exactly undone by the frequency downconversion process (translating the message information imparted on the high-frequency carrier down to baseband) in the receiver. Therefore, our simulations are valid for wireless systems under the assumption that the frequency translation processes are ideal. For coherent communication systems, the key challenge is synchronization between the transmitter and receiver. Ensuring coherent operation can be a significant challenge!

In this part of the laboratory you will observe and investigate *coherent* digital modulation techniques using real-world wireless signals. In particular, you will observe:

- **Carrier Synchronization**: the process of ensuring that the receive carrier is synchronized, in both frequency and phase, to the transmit carrier.
- **Symbol Timing Recovery:** the process of ensuring that the receiver precisely knows the symbol timing so that it may determine when to sample the recovered message for making symbol decisions.

Both of these processes are essential for coherent communication. Symbol timing recovery, but not carrier synchronization, is required in non-coherent communication systems.

Because our laboratory hardware consists of software-defined radios, these processes are implemented through adaptive software algorithms. In traditional radio systems, they were realized through analog circuitry. Our focus will be on explaining (and seeing!) the importance of these processes within a coherent digital communication system, rather than the algorithms or circuits through which they are implemented. Students interested in learning more are encouraged to consult the course textbook or take a follow-on course in communication systems.

Download the BPSK Simulink model provided by your instructor. The model is shown in Figure 5 The blocks in the model have been pre-configured for this laboratory exercise.

IMPORTANT: For the BPSK receiver to work properly, the carrier frequencies of the transmitter and receiver must be relatively close to each other (ideally, no more than a few hundred Hz to perhaps 1kHz apart). Compensate the center frequency of the **RTL-SDR Receiver** block in the model according to the calibration procedure you performed in Section 3. You can accomplish this through either the `Frequency Correction (ppm)` setting or by de-tuning the center frequency according to your calibration. Doing so provides a *coarse* frequency correction. A *fine* frequency correction will be applied through an adaptive algorithm in the Simulink model.

Except for the modification to the **RTL-SDR Receiver** block described above, please do not modify the configuration of any other blocks without first consulting your instructor. Your instructor will discuss each of the blocks in the receiver in detail. In summary:

- The BPSK symbols are being transmitted at 8000 bits per second.
- The receiver model includes a center frequency offset correction. Alternatively, correct for frequency offset through the `Frequency Correction (ppm)` setting of the **RTL-SDR Receiver** block.
- The receive gain is controlled via a **Slider Gain** block, making it easy to adjust the receive gain while the model is running.

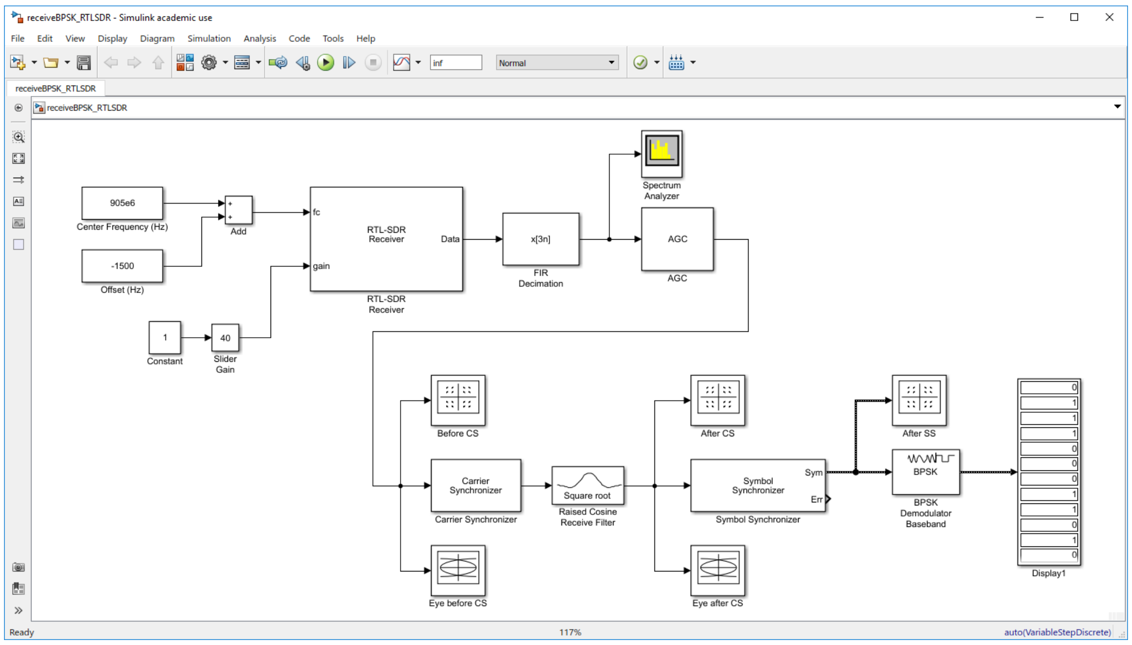


Figure 5: Simulink model for a BPSK receiver using the RTL-SDR device.

- The **FIR Decimation** block reduces the sample rate of the receive data by a factor of 3. The output of the **RTL-SDR Receiver** block is configured for $240e3$ samples per second, therefore the decimation reduces the rate to $240e3/3 = 80e3$ samples per second. This gives 10 samples per BPSK symbol in the receive data. Confirm the sample rate, and the frequency offset correction, by carefully viewing the **Spectrum Analyzer** plot.

- The **AGC** block implements an automatic gain control system. This block adaptively adjusts its gain to give a constant signal level at its output. Such a block is common in systems in which the receiver input signal level changes due to variation in the communication channel.

- The **Carrier Synchronizer** block adjusts for carrier frequency and phase offsets in the incoming signal. This block will compensate for offsets that remain after coarse calibration of the carrier frequencies.

- The **Raised Cosine Receive Filter** block filters the input signal using a raised cosine pulse shape filter. The block is configured to match the pulse shape used in your instructor's transmitter, and therefore implements a matched filter.

- The **Symbol Synchronizer** block adjusts for symbol clock drift in the incoming signal, ensuring that incoming pulses are sampled (for purposes of deciding which symbol was sent) at the "opening" of the eye. The block outputs exactly one sample per symbol interval.

- The **BPSK Demodulator Baseband** block demodulates its input signal and outputs the recovered bit sequence, which can be viewed on the **Display** block.

Question 5.1: Carefully observe the **Eye Diagram** plots at the input and output of the **Carrier Synchronizer** block. Take screen captures of each. Explain and interpret the plots.

Question 5.2: Carefully observe the **Constellation Diagram** plots at the input and output of the **Carrier Synchronizer** block. Take screen captures of each. Explain and interpret the plots.

Question 5.3: Carefully observe the **Constellation Diagram** plots at the input and output of the **Symbol Synchronizer** block. Take screen captures of each. Explain and interpret the plots.

Question 5.4: Your task in this question is to observe the effects of reduced signal-to-noise ratio (SNR) at the input to the receiver. You can reduce the SNR in several ways, such as:

- decreasing the `Gain` of the **RTL-SDR Receiver**
- placing an object near/over the receive antenna
- move your receiver to a different location within (or even outside) the laboratory

Carefully examine the effects of reduced SNR on **Constellation Diagram** and **Eye Diagram** plots. Take screen captures of your findings. Explain and interpret the plots.

6 Differential BPSK - Text Message Communication

In Section 5 we saw coherent reception of a BPSK communication signal. Generally speaking, phase-shift keyed signals cannot be detected non-coherently. However, a technique known as *differential* phase-shift keying allows detection of phase-shift keyed symbols without a fully coherent receiver. As you will see below, demodulating a DBPSK symbol is accomplished by using the immediately prior symbol as a phase reference.

In this portion of the laboratory you will explore differential binary phase-shift keying (DBPSK) by receiving and decoding a DBPSK communication signal containing an ASCII text message.

6.1 DBPSK Modulation and Demodulation

We will consider a DBPSK modulator that forms its communication signal using the following two operations:

1. The binary message data is differentially encoded. Let $\{b_k\}$ denote the binary message data and $\{d_k\}$ denote the differentially encoded message. One example of differential encoding operates as follows:

   - if the current symbol b_k is a **0**, then leave the symbol d_k the same as the previous symbol d_{k-1}
   - if the current symbol b_k is a **1**, then change the symbol d_k with respect to the previous symbol d_{k-1}

   This example of differential encoding is illustrated in the table below where the initial encoded symbol (at $k = -1$) is arbitrarily chosen to be **1**.

   Table 1: Differential encoding example.

k	-1	0	1	2	3	4	5	6	7
$\{b_k\}$		1	0	0	1	0	1	1	0
$\{d_k\}$	1	0	0	0	1	1	0	1	1

2. The differentially encoded message is phase-shift keyed. For the case of BPSK with rectangular pulse shapes, the complex envelope is

$$z(t) = Am(t)$$

   where $m(t)$ has values ± 1 over each symbol interval based on $\{d_k\}$. The passband BPSK waveform is

$$\begin{aligned} s(t) &= \Re\{z(t)\exp(j2\pi f_c t)\} \\ &= \Re\{Am(t)\exp(j2\pi f_c t)\} \\ &= Am(t)\cos(2\pi f_c t) \\ &= \begin{cases} A\cos(2\pi f_c t), & m(t) = 1 \\ -A\cos(2\pi f_c t), & m(t) = -1 \end{cases} \\ &= \begin{cases} A\cos(2\pi f_c t), & m(t) = 1 \\ A\cos(2\pi f_c t + \pi), & m(t) = -1 \end{cases} \end{aligned}$$

The key result here is that the message information is encoded in the phase difference between consecutive BPSK symbols. The DBPSK receiver then examines the relative phase between consecutive symbols to recover the message.

We now consider our standard model for an SDR receiver. Because the receive carrier is not synchronized to the transmit carrier, we model the receiver as having frequency offset Δf and phase offset ϕ. Therefore, the receiver output is

$$\begin{aligned}\hat{z}(t) &= \text{LPF}\{s(t)e^{-j(2\pi(f_c-\Delta f)t+\phi)}\} \\ &= \text{LPF}\{Am(t)\cos(2\pi f_c t)e^{-j(2\pi(f_c-\Delta f)t+\phi)}\} \\ &\vdots \\ &= \frac{1}{2}Am(t)e^{j(2\pi\Delta f t-\phi)}\end{aligned}$$

Consider the very fortunate, but highly improbable, case in which $\Delta f = 0$ and $\phi = 0$ which gives

$$\hat{z}(t) = \frac{1}{2}Am(t)$$

This case corresponds to coherent detection. Notice that, as expected, the differentially encoded symbols are perfectly recovered! This case is illustrated in the (noisy) constellation diagram shown in Figure 6(a).

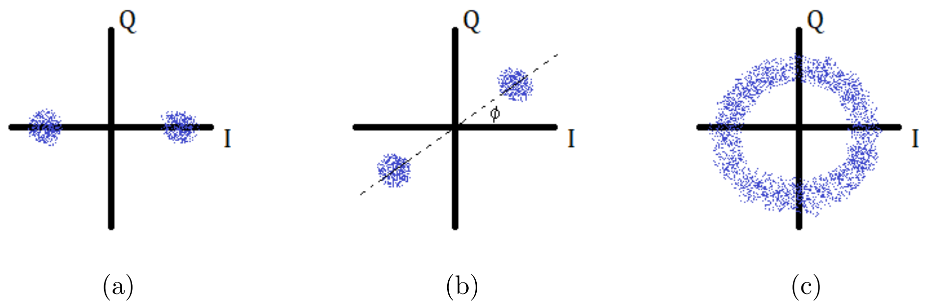


Figure 6: DBPSK constellations for (a) coherent detection, (b) carrier phase error, and (c) carrier frequency error.

Next consider another highly improbable case where $\Delta f = 0$ and $\phi \neq 0$. That is, the receive carrier has only a phase error. The receiver output is

$$\hat{z}(t) = \frac{1}{2}Am(t)e^{j\phi}$$

which means that the symbols have been rotated in the I/Q space by angle ϕ. However, the message bits are recoverable since the symbols retain the proper relative phase (π radians). This case is illustrated in the constellation diagram shown in Figure (b).

Finally, consider the practical and likely case where $\Delta f \neq 0$ and $\phi \neq 0$. That is, the receive carrier has both a frequency and phase error. The receiver output is

$$\hat{z}(t) = \frac{1}{2}Am(t)e^{j(2\pi\Delta f t-\phi)}$$

which means that the symbols are undergoing rotation in the I/Q space at the rate $2\pi\Delta f$ radians per second. So long as Δf is small compared to the symbol rate, the phase change between consecutive symbols will remain close to either 0 or π, thus preserving the differential encoding. This case is illustrated in the constellation diagram shown in Figure (c).

6.2 DBPSK Simulation

Construct the DBPSK Simulink model shown in Figure 7. Several salient points and suggestions:

- The model simulates DBPSK using only symbols. It does not include pulse shaping in the transmitter or matched filtering in the receiver (although you are free to add these elements). Set the bit rate to 8000 bits per second.
- The **DBPSK Modulator Baseband** block applies differential encoding followed by BPSK modulation to the binary message generated by the **Bernoulli Binary Generator** block. It uses the differential encoding scheme shown in Table 1. Carefully examine the **Time Scope** plot to see the message bits and the differentially encoded symbols.
- The **Phase/Frequency Offset** block is used to apply frequency offsets Δf and phase offsets ϕ to the received symbols. Experiment with these offsets, paying special attention to the **Constellation Diagram** plot.
- At what frequency offset do you begin to see large numbers of errors? Why?
- The **Constant** block controls the channel noise power. Experiment to see the impact of channel noise.

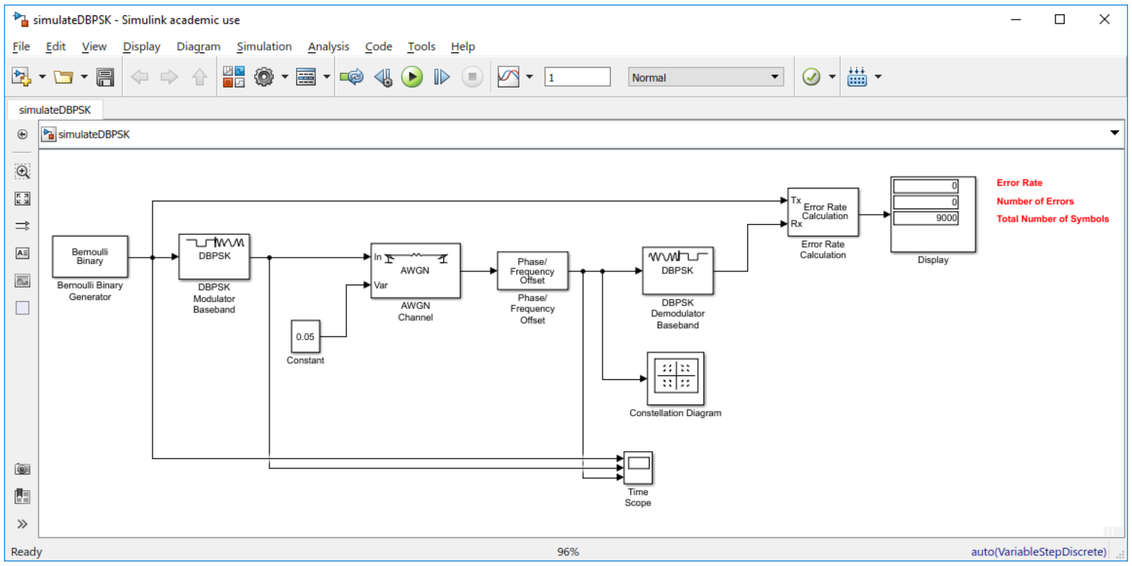


Figure 7: Simulink model for a DBPSK communication.

6.3 DBPSK Receiver

In this portion of the lab your instructor will broadcast a communication signal using DBPSK modulation that contains an ASCII text message. Your task is to decode the DBPSK symbols and recover the text!

Construct the Simulink model shown in Figure 8. Some salient points regarding the DBPSK broadcast and

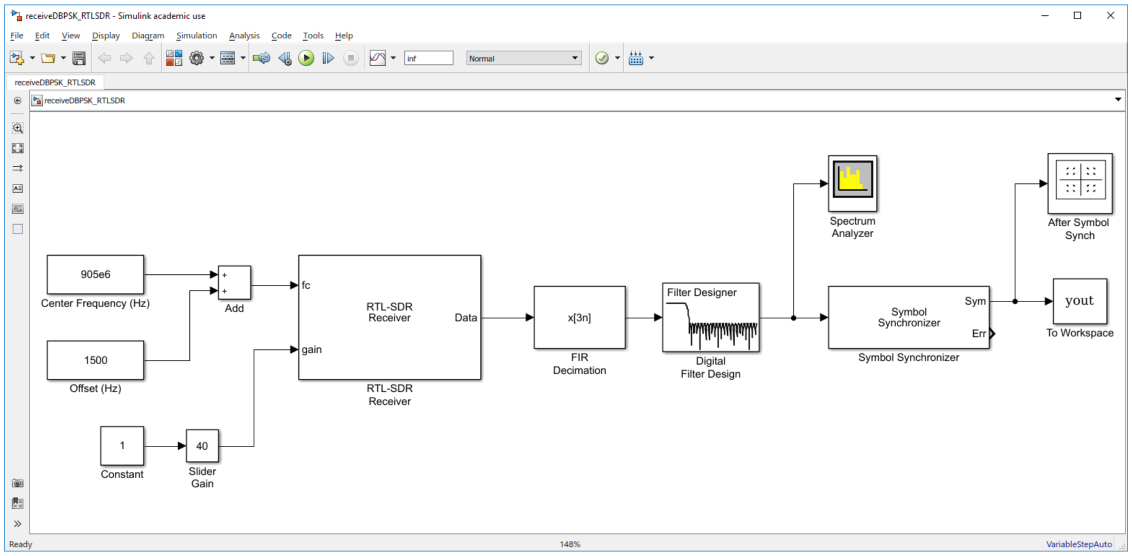


Figure 8: Simulink model for a DBPSK receiver using the RTL-SDR device.

the receiver model:

- The DBPSK symbols are being transmitted at 8000 symbols per second.
- The transmitter uses square-root Raised Cosine pulse shapes.
- **RTL-SDR Receiver** configuration:
  - Sampling rate: 240e3
  - Output data type: `single`
  - Samples per frame: 30000
  - The center frequency must be corrected according to the calibration performed in this laboratory. The receiver model includes a center frequency offset correction. Alternatively, correct for frequency offset through the `Frequency Correction (ppm)` setting of the **RTL-SDR Receiver** block.
- The **FIR Decimation** block reduces the sample rate of the receive data by a factor of 3. The output of the **RTL-SDR Receiver** block is configured for $240e3$ samples per second, therefore the decimation reduces the rate to $240e3/3 = 80e3$ samples per second. This gives 10 samples per bit in the receive data. Confirm the sample rate, and the frequency offset correction using a **Spectrum Analyzer** block.
- The model includes a **Digital Filter Design** block, configured as a lowpass filter, to filter the received signal. It is suggested to use a 40th order FIR Equiripple filter. It is up to you to set the pass and stop frequencies of the filter. While running the model, confirm that this filter is properly passing the DBPSK signal using a **Spectrum Analyzer** block.

- The model includes a **Symbol Synchronizer** block to ensure that the received symbols are sampled at the proper time. The block outputs exactly one sample per symbol interval. This process is required in all digital communication systems, regardless of whether they are coherent or noncoherent. Set the `Samples per symbol` to 10.

- The **Constellation Diagram** will show the signal constallation for the received DBPSK symbols. You may also want to add a **Eye Diagram** block.

- The **To Workspace** block writes data to the MATLAB workspace. Set the `Limit data points to last` parameter to 1000, which will result in the values pertaining to the last 1000 DBPSK symbols begin saved to the MATLAB workspace. Set `Save format` to `Structure`.
  - After running the model, you will see a structure named `yout` in the MATLAB workspace. You can save the MATLAB workspace, and therefore your recorded data, using the `save` function in MATLAB. You can then restore the workspace at a later time using the `load` function.
  - The actual DBPSK symbols are located within the structure at `yout.signals.values`

Some hints and suggestions for recovering the ASCII text:

1. Write a MATLAB script for recovering the ASCII text message. This script can utilize recorded/saved data from the Simulink model shown in Figure 8. You are not required to implement real-time decoding in Simulink.

2. Remember that the received bits are differentially encoded, and therefore you will need to first decode them. You can calculate the angle associated with each symbol using the `angle` function in MATLAB. You might also find the `wrapToPi` and `wrapTo2Pi` functions helpful.

3. Your instructor will provide detailed information regarding the data frame. In particular, you must identify the start of the data frame so that you properly group the bits into ASCII characters. This is accomplished using a message *preamble*.

4. The following example MATLAB script shows how to convert between ASCII characters and a vector of bits.

```
%% Convert text into a vector of bits
a = dec2bin('Test Message\n',8);    % get binary representation; gives character array
numChars = size(a,1);               % get number of characters

a = a - '0';                        % converts char array into matrix of numbers
a = reshape(a.',8*numChars,1);      % reshape matrix into a vector; 8 bits per
    character

stem(a)                             % plot vector of bits

%% Recover text from vector of bits
b = reshape(a, 8, numChars).';      % reshape vector into matrix; 8 bits per row
b = num2str(b);                     % convert to char array
b = bin2dec(b);                     % convert to decimal
fprintf(char(b));                   % print characters
```

Figure 9: MATLAB example for converting between ASCII characters and vectors of binary data.

Question 6.1: Carefully observe the **Signal Constellation** plot at the output of the **Symbol Synchronizer** block. Take a screen capture. Explain and interpret what is seen in the plot.

Question 6.2: Describe in detail your method for decoding the differentially encoded symbols. Include and explain the MATLAB code you wrote to perform the decoding. Include plots that help explain your approach.

Question 6.3: Provide a `stem` plot of the recovered message bit sequence. This plot must show the message preamble and a complete data frame. Annotate the plot, identifying the preamble.

Question 6.4: What is the text message? Provide the complete MATLAB code listing you used in this portion of the lab.

A Binary Digital Carrier Modulation

1. Amplitude-shift keying (ASK), also known as on-off keying (OOK), can be represented as

$$s(t) = Am(t)\cos(2\pi f_c t) \tag{1}$$

where $m(t)$ is a baseband signal that takes on values of either 0 or +1 during each bit interval. The complex envelope is

$$g(t) = Am(t) \tag{2}$$

2. Binary phase-shift keying (BPSK) can be represented as

$$s(t) = \begin{cases} A\cos(2\pi f_c t), & \text{for t in the time interval when a 0 is sent} \\ A\cos(2\pi f_c t + \pi), & \text{for t in the time interval when a 1 sent} \end{cases} \tag{3}$$

$$= Am(t)\cos(2\pi f_c t) \tag{4}$$

where $m(t)$ is a baseband signal that takes on values ± 1 during each bit interval. The complex envelope is

$$g(t) = Am(t) \tag{5}$$

3. Frequency-shift keying (FSK) can be represented as

$$s(t) = \begin{cases} A\cos(2\pi f_{c_1} t), & \text{for t in the time interval when a 0 is sent} \\ A\cos(2\pi f_{c_2} t), & \text{for t in the time interval when a 1 sent} \end{cases} \tag{6}$$

$$= Am(t)\cos(2\pi f_{c_1} t) + A(1 - m(t))\cos(2\pi f_{c_2} t) \tag{7}$$

where $m(t)$ is a baseband signal that takes on values of either 0 or +1 during each bit interval. Note that in this form, FSK can be interpreted as the sum of two ASK signals.

Alternatively, FSK can be expressed as a frequency modulated signal of the form

$$s(t) = A\cos\left[2\pi f_c t + k_f \int_{\infty}^{t} m(\lambda)d\lambda\right] \tag{8}$$

where $m(t)$ is a baseband signal that takes on values ± 1 during each bit interval. The complex envelope is

$$g(t) = Ae^{j\theta(t)} \tag{9}$$

where

$$\theta(t) = k_f \int_{\infty}^{t} m(\lambda)d\lambda \tag{10}$$

4. Recall that for passband signal $s(t)$ having complex envelope $g(t)$,

$$s(t) = \Re\{g(t)e^{j2\pi f_c t}\} \tag{11}$$

where $g(t)$ is a baseband signal. If $g(t)$ has power spectral density $S_G(f)$ then $s(t)$ has power spectral density

$$S_S(f) = \frac{1}{4}\left[S_G(f - f_c) + S_G(-f - f_c)\right] \tag{12}$$

Therefore, the power spectral density of ASK, PSK, and FSK signals contain frequency shifted copies of the baseband signal power spectral density.

References

[1] American Television Systems Committee (ATSC). ATSC standards. https://www.atsc.org/standards/atsc-standards/.

[2] Mathworks. Communications Toolbox. Online Resource. https://www.mathworks.com/products/communications.html.

[3] Mathworks. USRP Support Package from Communications Toolbox. Online Resource. http://www.mathworks.com/discovery/sdr/usrp.html.

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

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

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

[7] B.P. Lathi and Zhi Ding. *Modern Digital and Analog Communication Systems*. 5th edition, 2019.