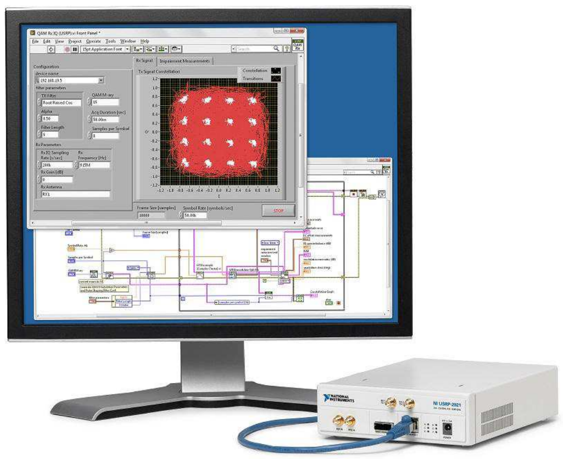


Lab Based Learning with NI USRP and LabVIEW

INTRODUCTION TO COMMUNICATION SYSTEMS

Student Lab Manual

By: Mahdi Maaref

October 30, 2017

Contents

LAB 1

Introduction to the LabView Communication

1.1 Objective

The purpose of this introductory laboratory exercise is to ensure that students have a working installation of LabVIEW Communication and be familiar with basics of LabVIEW Communication programming.

1.2 LabVIEW Communication 2.0 Installation on Laptops

- Open the link below:

  `http://systems.engr.ucr.edu/software/labviewcomm.html`

- Download the Package.
- Extract the LabVIEW package, you will need 7zip or a similar extraction utility to do this.
- Connect to engineering VPN server.
- Right click and run "autorun.exe" as admin. The "autorun.exe" file is located inside the folder you extracted. Install Labview.
- After LabVIEW is installed, you need to update the license server information using "NI License Manager".
- Run "NI License Manager".
- Choose Options/Preferences.
- Check the "Use Volume License Server" and enter "labview.engr.ucr.edu". Click OK.
- To run LabVIEW, make sure you are connected to engineering VPN server and then run the program like you normally would.

1.3 Background

LabVIEW Communication is a graphical programming language developed by National Instruments. The basic building block of LabVIEW Communication is the virtual instrument (VI). Conceptually, a VI is analogous to a procedure or function in conventional programming languages. Each VI consists of a **block diagram** and a **front panel**. The block diagram describes the functionality of the VI, while the front panel is a top level interface to the VI. The construct of the VI provides two important virtues of LabVIEW Communication: code reuse and modularity. The graphical nature of LabVIEW Communication provides another virtue: it allows developers to easily visualize the flow of data in their designs. NI calls this Graphical System Design. Also, since LabVIEW Communication is a mature data flow programming language, it has a wealth of existing documentation, toolkits, and examples which can be leveraged in development.

You should realize that the algorithms considered here could also be programmed in optimized C/C++, assembly, or VHDL and implemented on a DSP, microcontroller, or an FPGA. The choice of hardware and software in this lab is mostly a matter of convenience.

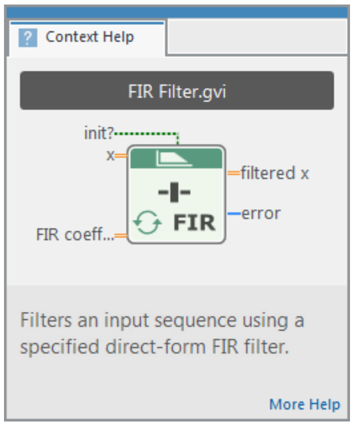
In future labs you will need to be familiar with LabVIEW and the documentation/help available to you. This is the only lab in this course which will give you the opportunity to learn and practice LabVIEW programming; so it is important that you take this opportunity to ask the instructor any questions you might have about LabVIEW programming. The following tutorials and reference material will help guide you through the process of learning LabVIEW:

- LabVIEW Communications System Design Suite 2.0 Online Manual.[1]
- LabVIEW Communications Guided Help tutorials.
- Context help.[2]

The context help window displays basic information about LabVIEW objects when you move the cursor over each object. To toggle the display of the context help window, select Help → Context Help or press $< Ctrl - H >$. The LabVIEW online help is the best source of detailed information about specific features and functions in LabVIEW. Online help entries break down topics into a concepts section with detailed descriptions and a how-to section with step-by-step instructions for using LabVIEW functions.

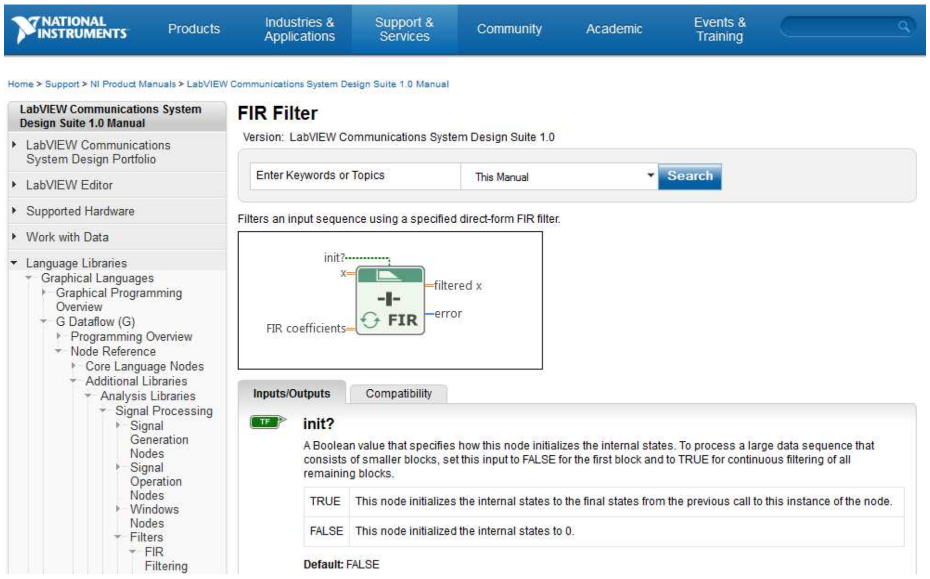
[1]http://www.ni.com/documentation/en/labview-comms/latest/manual/labview-comms-manual/

[2]Context help is available in LabVIEW Communications after opening a program from Help menue →Context Help



Filters an input sequence using a specified direct-form FIR filter.

Figure 1.1: Context Help Window



FIR Filter

Version: LabVIEW Communications System Design Suite 1.0

Filters an input sequence using a specified direct-form FIR filter.

init?

A Boolean value that specifies how this node initializes the internal states. To process a large data sequence that consists of smaller blocks, set this input to FALSE for the first block and to TRUE for continuous filtering of all remaining blocks.

TRUE	This node initializes the internal states to the final states from the previous call to this instance of the node.
FALSE	This node initialized the internal states to 0.

Default: FALSE

Figure 1.2: Screenshot of LabVIEW Online Help

1.4 Lab Procedure

1. Ensure LabVIEW Communications 2.0 is installed.
2. Open LabVIEW Communications.
3. From the Help → lessons enter to the learning section. (also called the lobby), open and complete the guided help tutorial from Learn → Getting Started → Introduction to the LabVIEW Editor. Follow the steps in this tutorial. Skip pages 1 to 5 and start from page 6 which explains "Viewing, Creating and Interacting with Documents"

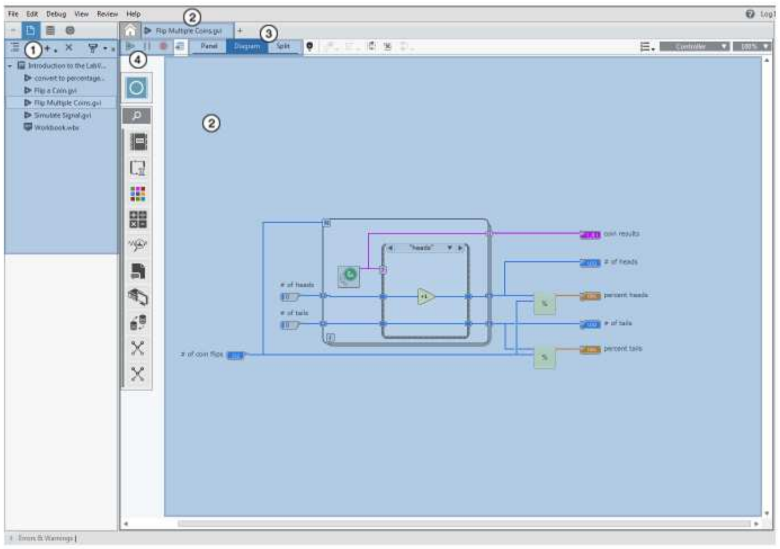
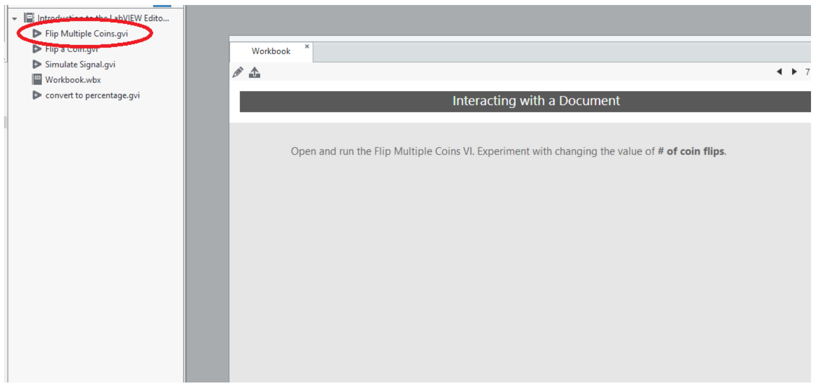


Figure 1.3: Tenurial for Viewing, Creating and Interacting with Documents

4. In page 7, you are asked to open and run Flip Multiple Coins VI. Double click on this file as shown in figure bellow:



Interacting with a Document

Open and run the Flip Multiple Coins VI. Experiment with changing the value of **# of coin flips**.

Figure 1.4: Opening Flip Multiple Coins VI

Experiment with change the value of "coin flips" and run the VI.

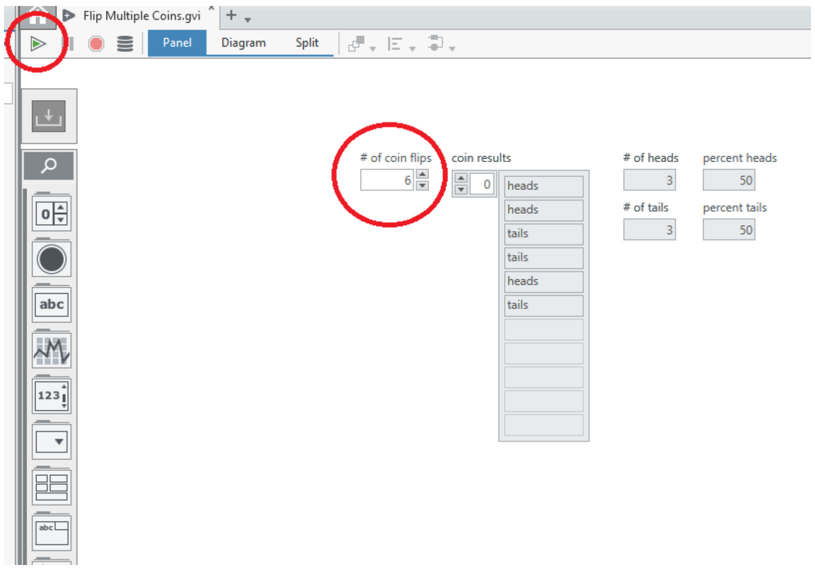


Figure 1.5: Running Flip Multiple Coins VI

5. In page 8, it is explained how to create a way to display data.

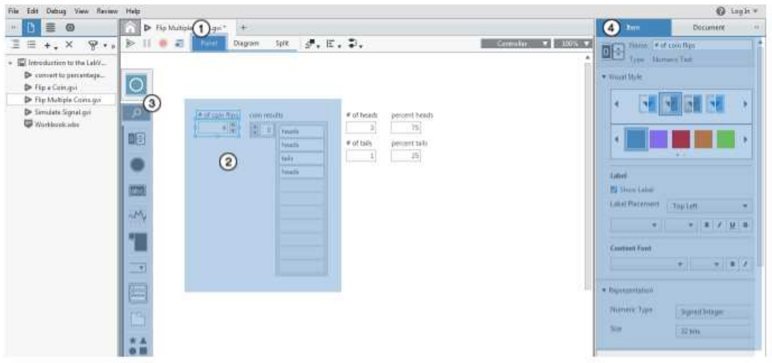
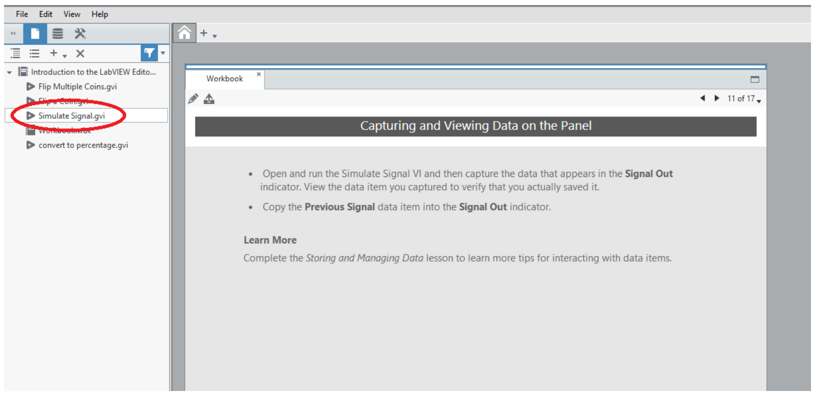


Figure 1.6: Creating a way to display data

6. Skip page 9.

7. Page 10, explains how to capture a data on the panel.

8. Page 11 asks you to open and run Simulated Signal.VI and capture the data that appears in the Signal Out indicator. View and Copy the data.



Capturing and Viewing Data on the Panel

- Open and run the Simulate Signal VI and then capture the data that appears in the **Signal Out** indicator. View the data item you captured to verify that you actually saved it.
- Copy the **Previous Signal** data item into the **Signal Out** indicator.

Learn More

Complete the *Storing and Managing Data* lesson to learn more tips for interacting with data items.

Figure 1.7: Capturing and Viewing Data on the Panel

After capturing the data (Based on page 10 instructions) drag the Capture and drop it into indicator. You can also double click on the capture to view it more precisely.

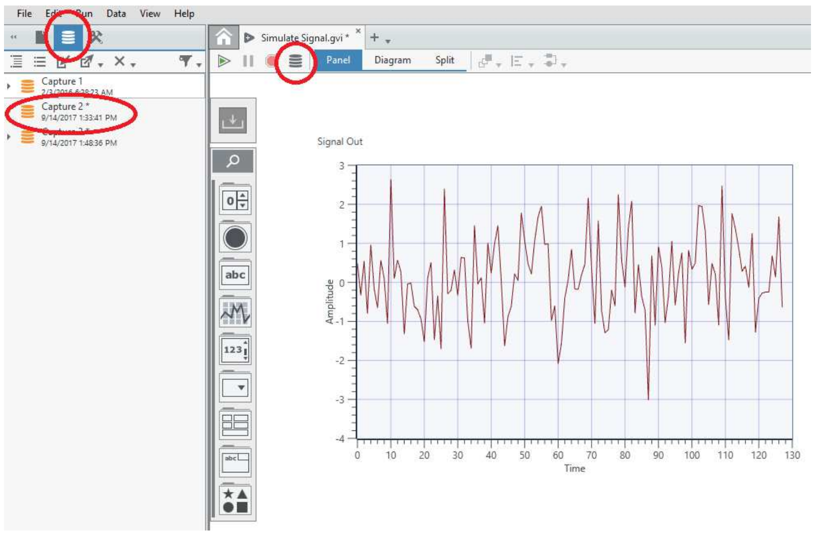


Figure 1.8: Capturing and Viewing Data on the Panel

9. Page 12 highlights the parts of the editor you use to create code. The indicated parts in bellow figure are important for creating a code:

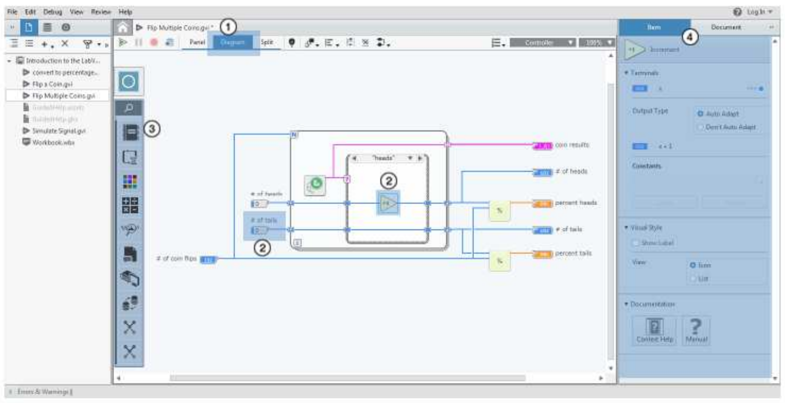
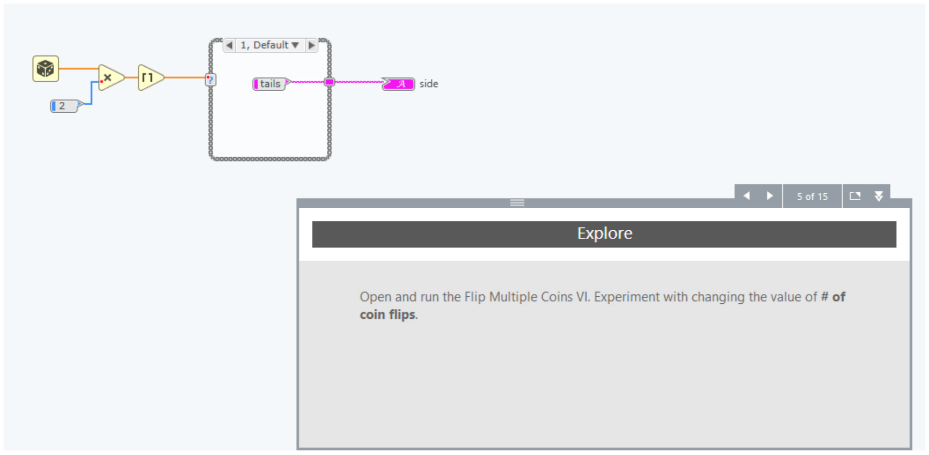


Figure 1.9: Important parts for creating a code is indicated with numbers 1 to 4

10. Skip pages 13 and 14.

11. Pages 15 explain multiple avenues for seeking help in LabVIEW Communication.

12. Page 16 explain how to use Help tools to understand a code. You are asked to Open Flip a Coin VI. Make sure you understand how the **Flip a Coin** program works. You will develop this program in this lab.



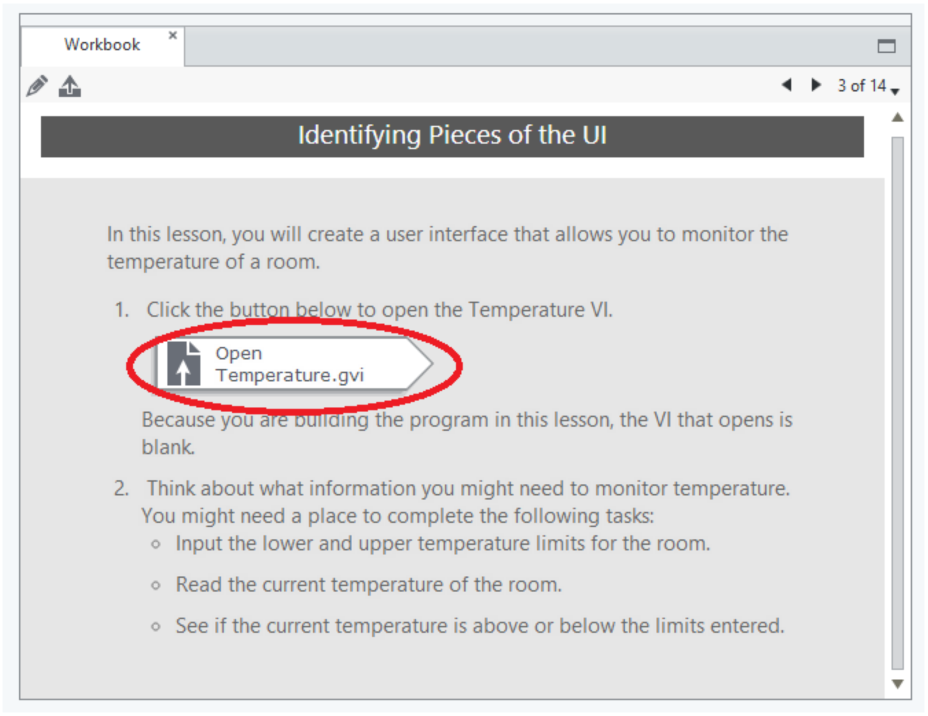
Open and run the Flip Multiple Coins VI. Experiment with changing the value of **# of coin flips**.

Figure 1.10: Flip a Coin program

13. After finishing this tutorial return to the lobby.

14. From learning section, open and complete the guided help tutorial for "Designing a User Interface" from Learn → Programming Basics → Designing a User Interface. In this tutorial you will design your first program which allows you to monitor the temperature of a room. Read pages 1 and 2 of this toturial for more information about main parts of program: **Panel** and **Diagram**.

15. Click the button in page 3 to open a VI. Since you will design this code, the VI is blank.



Identifying Pieces of the UI

In this lesson, you will create a user interface that allows you to monitor the temperature of a room.

1. Click the button below to open the Temperature VI.

   Because you are building the program in this lesson, the VI that opens is blank.

2. Think about what information you might need to monitor temperature. You might need a place to complete the following tasks:
   - Input the lower and upper temperature limits for the room.
   - Read the current temperature of the room.
   - See if the current temperature is above or below the limits entered.

Figure 1.11: Opening the Temperature VI.

16. Page 4 shows your desired panel for this program. This Panel contains: 2 Numerical Control, 1 Numerical indicator, 2 round LEDs and 1 String Indicator. Pages 5 to 11 help you to find these elements and place and arrange them into panel.

17. As it is explained in Page 12, when you run this VI, nothing appears to happen. Because you just developed a panel and there is no code in the diagram yet. In the following tutorial you will also develop a diagram.

18. Return to return to the lobby.

19. From the lobby, open and complete the guided help tutorial for “While Loops” from Learn → Programming Basics → While Loops. While loops are used to repeat code until a specified condition is met.

20. Page 2 presents a comprehensive description of both **For Loop** and **While Loop**

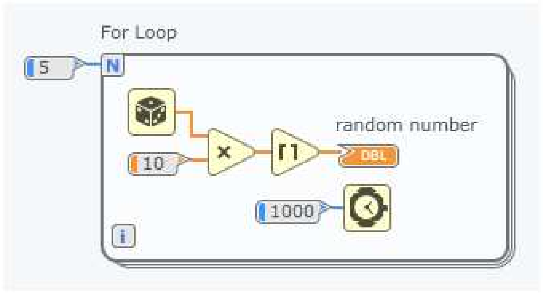
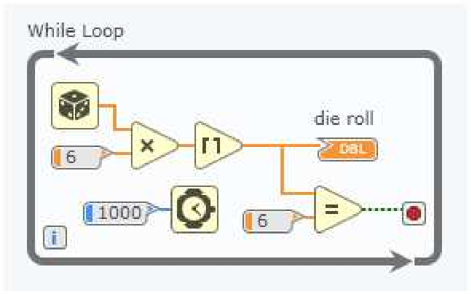
Loop	Behavior	Example	Diagram
For Loop	Repeats the code inside the loop for a set number of iterations.	Generate five random numbers between 1 and 10.	 For Loop 5 N 10 random number DBL 1000 i
While Loop	Repeats the code inside the loop until a specified condition is met.	Continue rolling a die until the value of the die is 6.	

Figure 1.12: **For Loop** and **While Loop**

21. In page 3, the Die.VI is opened. This VI simulate a dice and each time you run the program a dice has been rolled and the rolled number is illustrated. Check out the diagram of this code and make sure you understand how it works. The goal is to develop this code and add while loop to it. Let's say we want to know how many tries it takes to roll a 6!

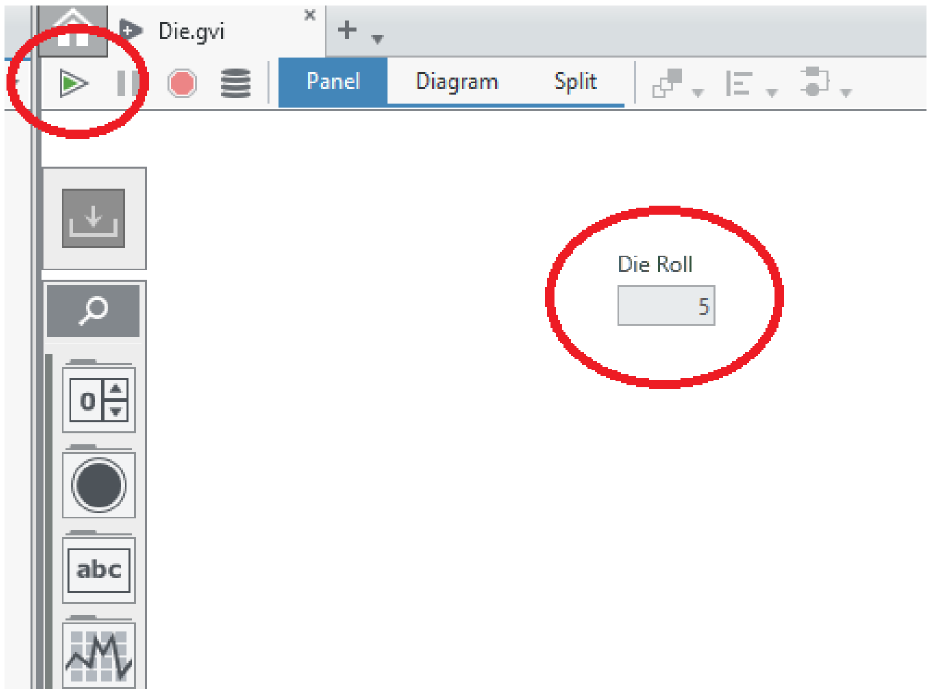
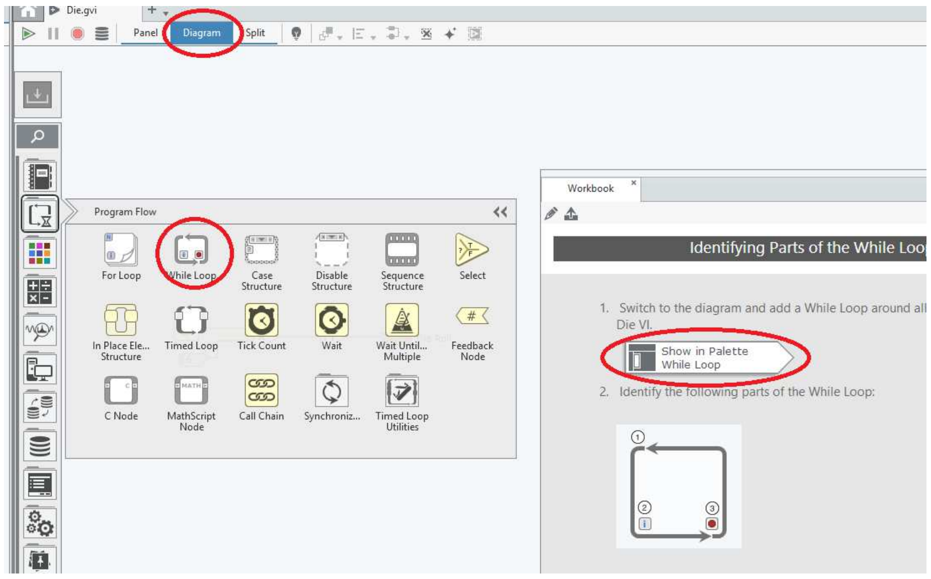


Figure 1.13: This code simulates rolling a dice, each time you run the code, a rolled number is illustrated. Check out the diagram of this code and make sure you understand how it works.

22. Page 4 helps you to find a while loop in the pallete. Make sure to switch to the diagram before inserting the while loop. Page 4 also illustrates different parts of while loop such as: Border, Iteration terminal and Conditional terminal.



Identifying Parts of the While Loo

1. Switch to the diagram and add a While Loop around all Die VI.

2. Identify the following parts of the While Loop:

Figure 1.14: While loop

23. Page 6 shows how we can stop the loop when the 6 is rolled. This is done by using **Equal** inside the loop. Make a diagram as shown in Page 6.

24. Page 7 shows how we can use iteration terminal of the loop. The iteration terminal is used to demonstrate how many times the loop executed before rolling that 6. See bellow figures to perform the instructions of page 7.

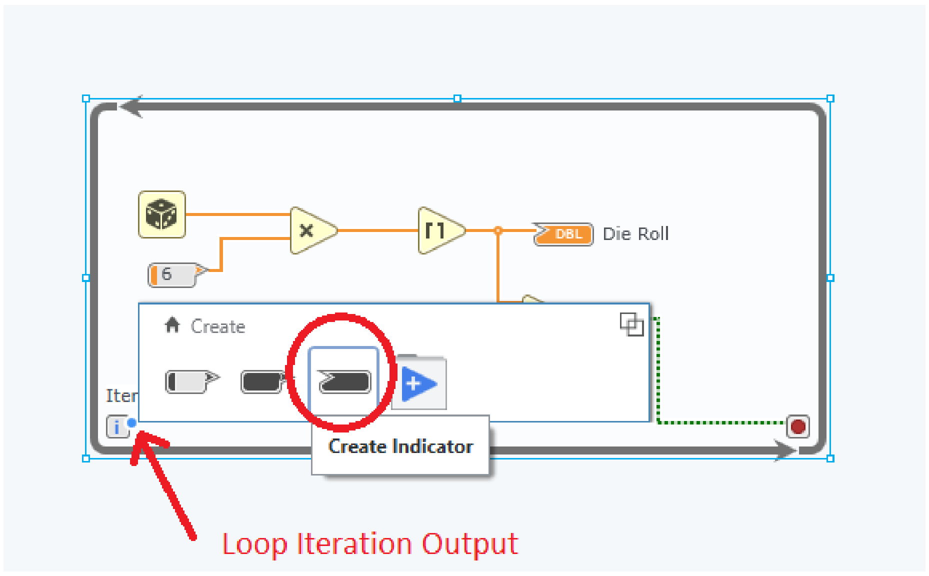


Figure 1.15: Right-click the loop iteration output of the iteration terminal and select Create Indicator.

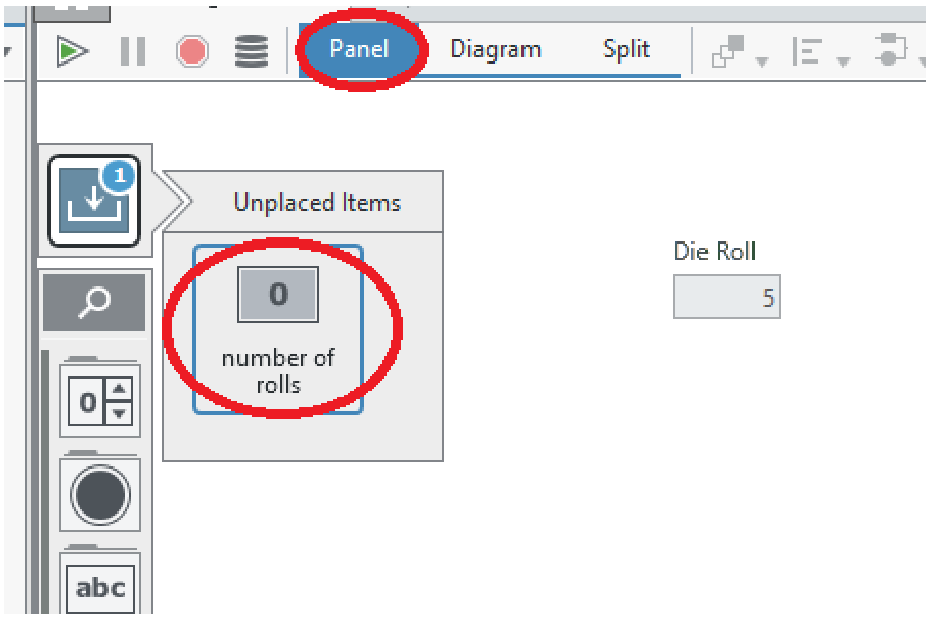
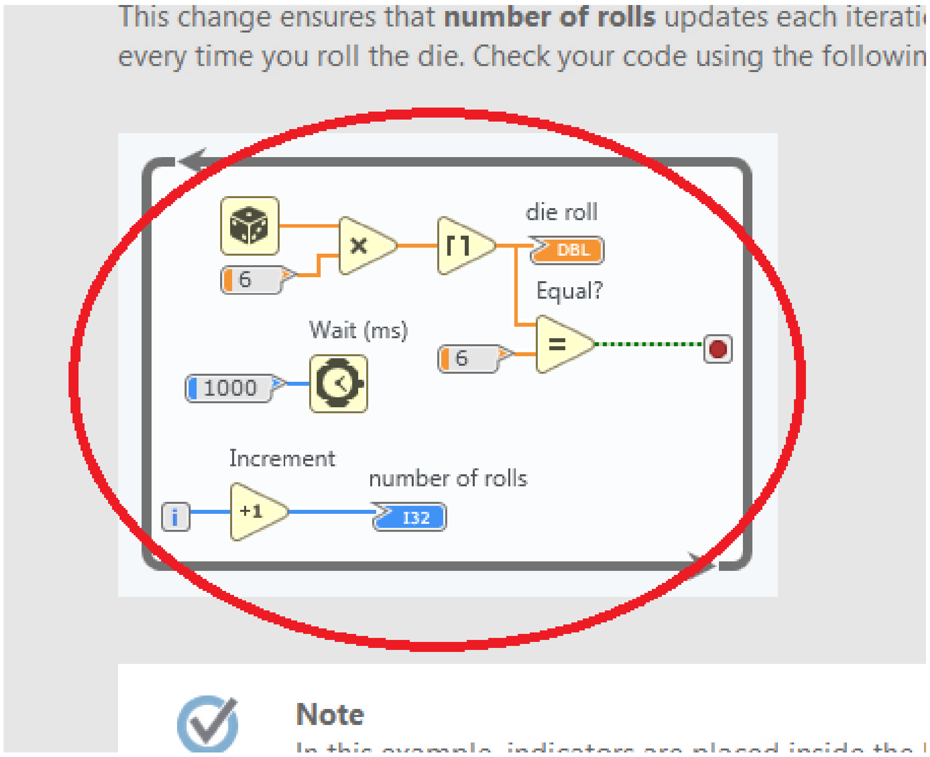


Figure 1.16: Switch to the panel. Add the number of rolls indicator to the panel from the Unplaced Items tray.

25. Page 10 helps you to control the timing of your while loop by adding **Wait**. Follow the instructions presented in page 10 to add a 1 second wait into your loop. Finally, your code should look like the diagram illustrated in Page 10.


This change ensures that **number of rolls** updates each iterati
every time you roll the die. Check your code using the followin

In this example, indicators are placed inside the

Figure 1.17: using timing in a loop

26. Finish this tutorial and return to return to the lobby.

27. Now, we want to rewrite the **Flip a Coin** program and add a while loop. So, it can continuously flip a coin until we press the stop button.

    - open the **Flip a Coin** program from Help → Lessons → Getting Started → Introduction to the LabVIEW Editor.

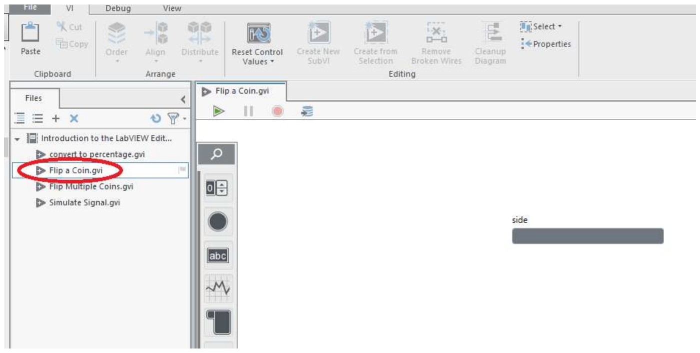


Figure 1.18: Flip a Coin program

    - make a new Project : File → New → Project

- Save a Project : File → Save Untitled Project. A new window appears for setting Location and Name for the Project. Set the projects name as **Flip a Coin2**
- Right-click on project's name and from Add New option, select VI. Set the name of VI as **Flip a Coin2**

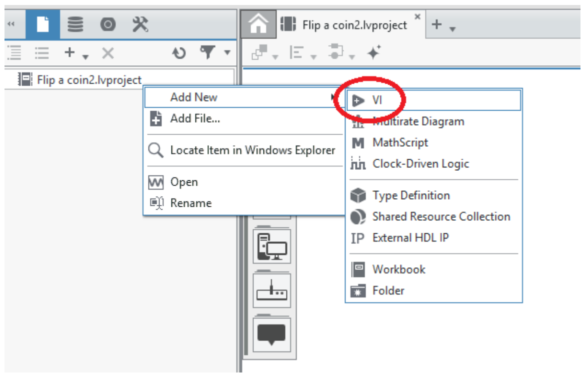


Figure 1.19: Make a New VI Project

- Copy the diagram from **Flip a Coin** program to the new project's diagram. Click the panel tab. As it can be seen in Fig. 1.20 we can update the panel and add the "string control" whose name is "side".

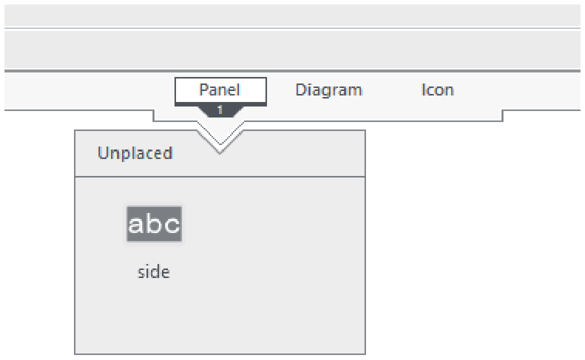


Figure 1.20: Panel tab

- From the "Panel pallete" add a "Stop Button" to the panel(Fig. 1.21). Then by clicking the Diagram tab, update the diagram.(Fig. 1.22)

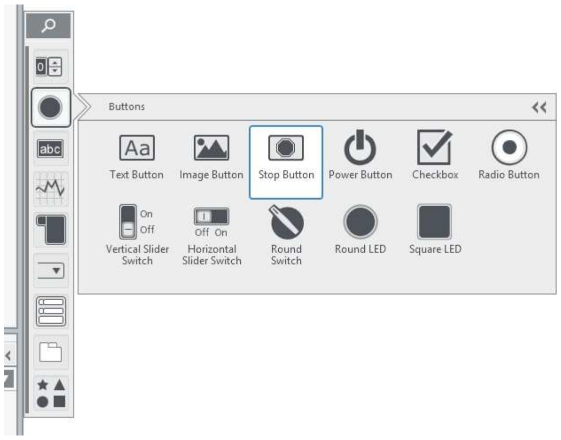


Figure 1.21: Stop Button

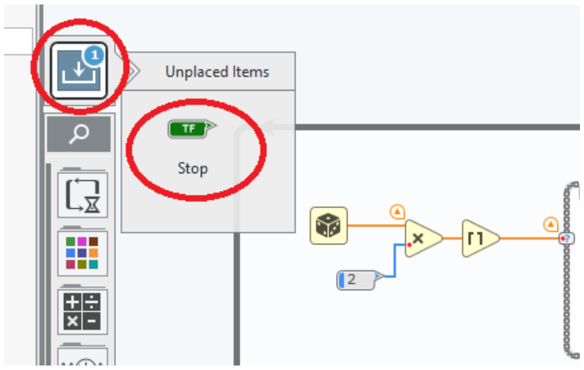


Figure 1.22: Stop Button

- From the “Diagram Pallete” add a “While Loop” to the diagram (Fig. 1.23) and add a While Loop around all of the code in the VI. Finally, connect the stop button to the stop indicator of the while loop (Fig. 1.24).

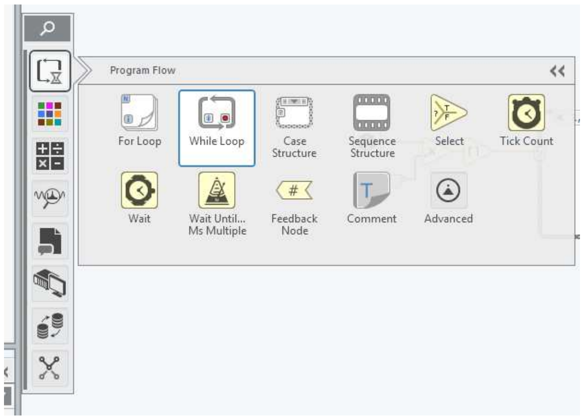


Figure 1.23: While Loop in the diagram palette

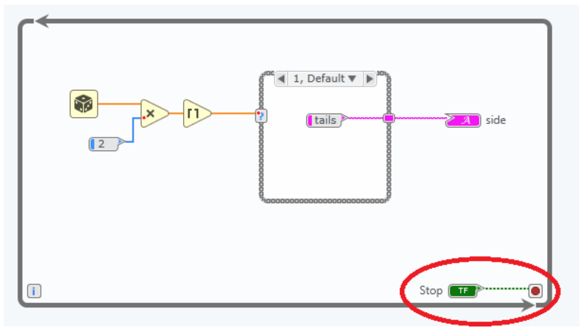


Figure 1.24: While Loop around the code

- From the "Diagram Pallete" add a "Wait" to the diagram (Fig. 1.25), Then add a numeric constant to the diagram.(Fig. 1.26)

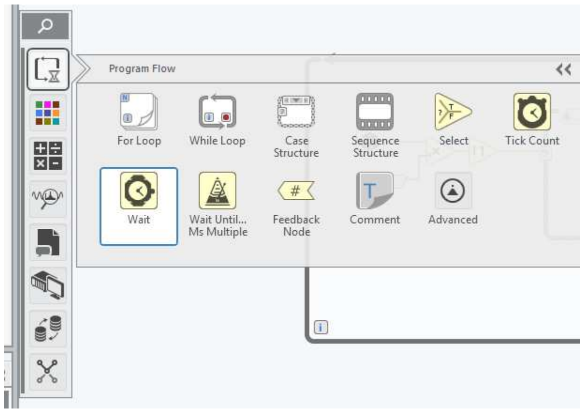


Figure 1.25: Wait

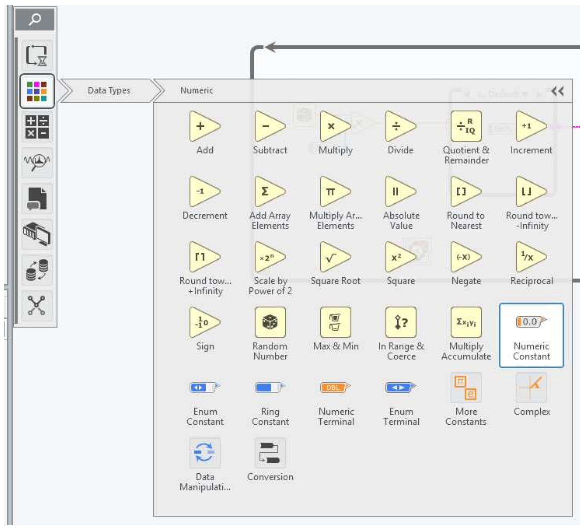


Figure 1.26: Numeric constant

- Change the value of numeric constant to the 1000 (1000 milliseconds) and connect it to the "wait". The Code is ready! (Fig. 1.27). Click the panel tab and run the program.

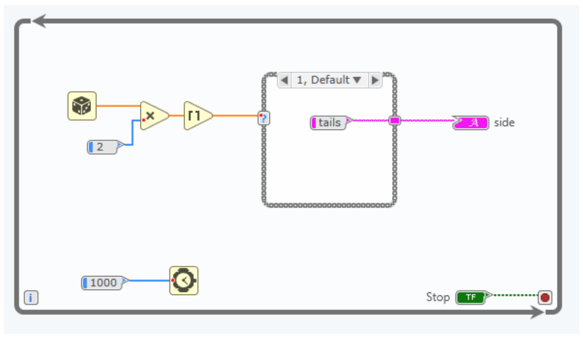


Figure 1.27: Final code

Bring any questions or concerns regarding LabVIEW or these tutorials to your instructor's attention. For the remainder of this lab you should be familiar with the basics of LabVIEW communication programming and where to look for help.

1.4.1 Questions

1. From the lobby, open and complete the guided help tutorial for "Controlling the Behavior of Your VI" from Learn → Programming Basics → Debugging your VI.

2. From the lobby, open and complete the guided help tutorial for "Basic Data Types" from Learn → Programming Basics → Basic Data Types.

3. From the lobby, open and complete the guided help tutorial for "Arrays" from Learn → Programming Basics → Arrays.

4. Write a program that rolls two dices every one seconds until it gets two sixes. Use a while loop to count the number of trials to reach two sixes. You can compete with your friends and see who is the luckiest!!!

   HINTS: In While Loop tutorial it is shown how to develop a code in which one dice is rolled until it gets 6. You can see how it is built and based up on that diagram, you would be able to answer this question. You can find While Loop tutorial from Lobby → Learn → Programming Basics → While Loops

1.5 Report

- Submit the **Flip the Coin2** program which you developed in the Lab.

- Submit the solution for question 4.

LAB 2

Introduction to the USRP

2.1 Objective

The purpose of this introductory laboratory exercise is to ensure that students have a working installation of LabVIEW Communications on their computers and know how to connect to the USRP software defined radio.

2.2 Background

The Wireless Innovation Forum defines Software Defined Radio (SDR) as: "Radio in which some or all of the physical layer functions are software defined." [1]

SDR refers to the technology wherein software modules running on a generic hardware platform are used to implement radio functions. By combining the NI USRP hardware with LabVIEW software you can create a flexible and functional SDR platform for rapid prototyping of wireless signals including physical layer design, record and playback, signal intelligence, algorithm validation, and more.

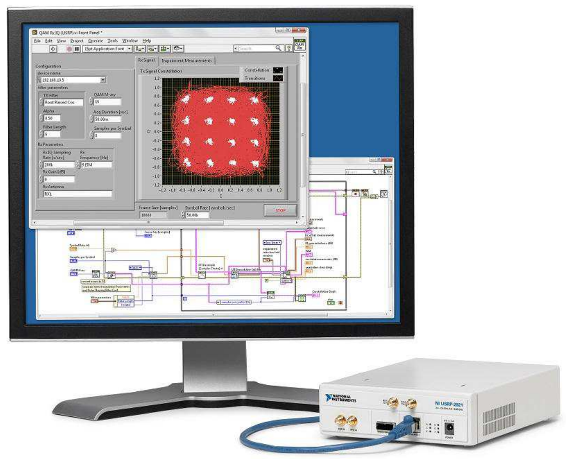


Figure 2.1: Hardware Setup in a Wireless Communications Lab

2.2.1 NI USRP Hardware

The NI USRP connects to a host PC creating a software defined radio. Incoming signals at the SMA connector inputs are mixed down using a direct-conversion receiver to baseband I/Q components, which are sampled by a analog-to-digital converter (ADC). The digitized I/Q data follows parallel paths through a digital downconversion

[1] `http://www.wirelessinnovation.org/what_is_sdr`

(DDC) process that mixes, filters, and decimates the input signal to a user-specified rate. The downconverted samples are passed to the host computer.

For transmission, baseband I/Q signal samples are synthesized by the host computer and fed to the USRP at a specified sample rate over Ethernet, USB or PCI express. The USRP hardware interpolates the incoming signal to a higher sampling rate using a digital upconversion (DUC) process and then converts the signal to analog with a digital-to-analog converter (DAC). The resulting analog signal is then mixed up to the specified carrier frequency.

More information about NI SDR hardware can be found in the respective Getting Started Guide[2] available in the start menu.

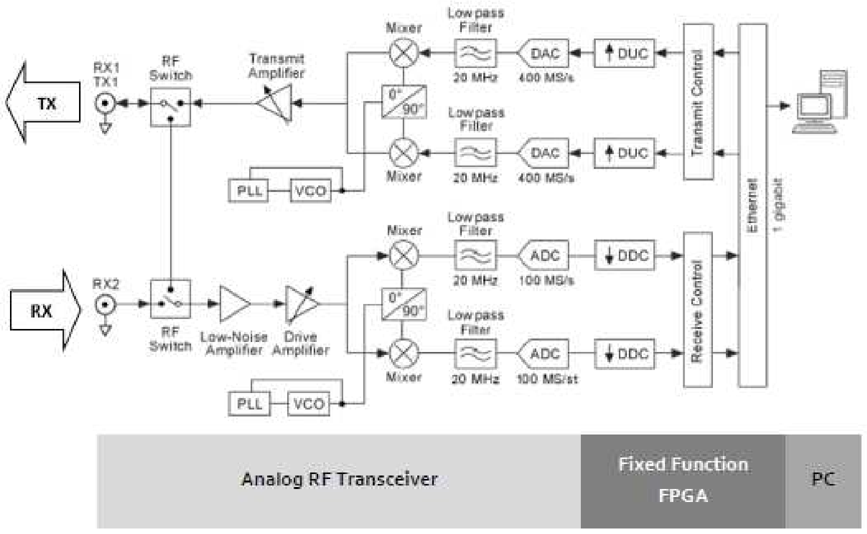


Figure 2.2: Typical Block Diagram of an NI USRP

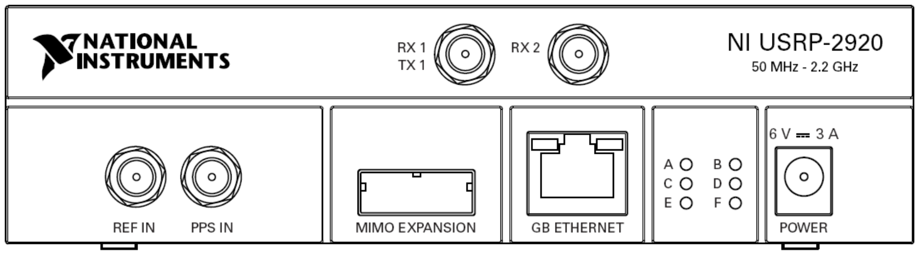


Figure 2.3: Front View of an NI USRP-2920 Software Defined Radio

[2]Available from the Start Menu → All Programs → National Instruments → NI-USRP → Documentation

2.3 Lab Procedure

1. Connect the USRP to the power using power cable, then connect the TX1 output to the RX2 connector using a loopback cable and 30 dB attenuator provided. Finally, connect the USRP to the computer using Ethernet cable(See Figure bellow)

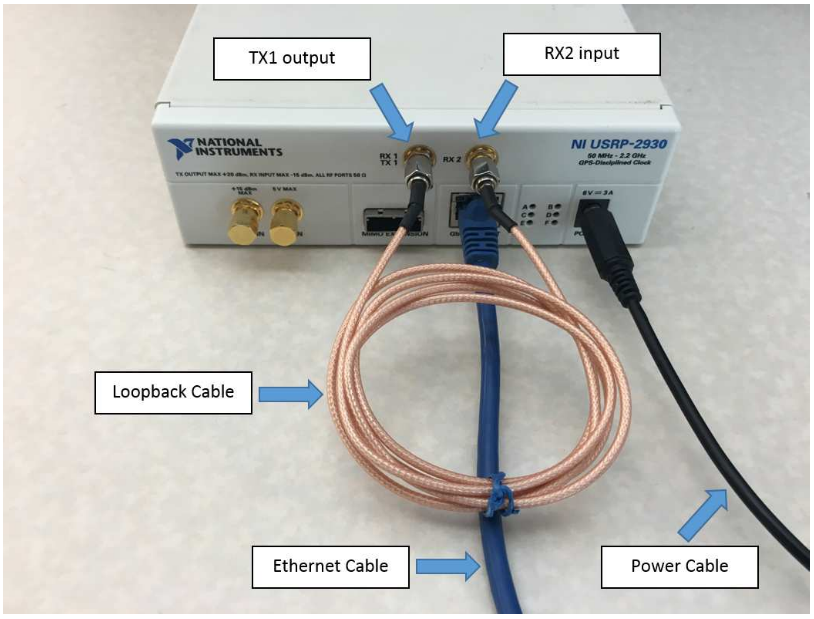
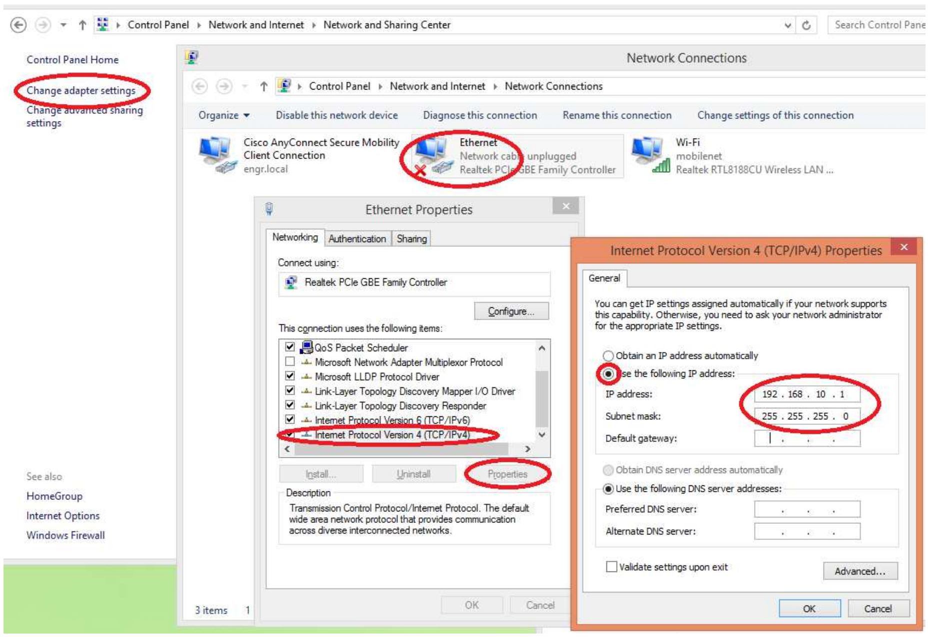


Figure 2.4: USRP Configuration.

2. Make sure that the IP of computer set properly based on the IP of the USRP. Ask the instructor if you had any question about setting the IP.

   For setting the IP, go to Control panel → Network and Internet → Network and sharing center → change adapter settings. Right click on Ethernet Icon and click Properties. Select Internet Protocol Version 4(TCP/IPv4) , Click Properties. Select “Use the following IP address”. and fill the “IP address” and “Subnet mask” as follows:



Control Panel > Network and Internet > Network and Sharing Center

Change adapter settings

Network Connections

Ethernet — Network cable unplugged — Realtek PCIe GBE Family Controller

Ethernet Properties

Connect using: Realtek PCIe GBE Family Controller

This connection uses the following items:

- [✓] QoS Packet Scheduler
- [ ] Microsoft Network Adapter Multiplexor Protocol
- [✓] Microsoft LLDP Protocol Driver
- [✓] Link-Layer Topology Discovery Mapper I/O Driver
- [✓] Link-Layer Topology Discovery Responder
- [✓] Internet Protocol Version 6 (TCP/IPv6)
- [✓] Internet Protocol Version 4 (TCP/IPv4)

Description: Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

Internet Protocol Version 4 (TCP/IPv4) Properties

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address: 192 . 168 . 10 . 1

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

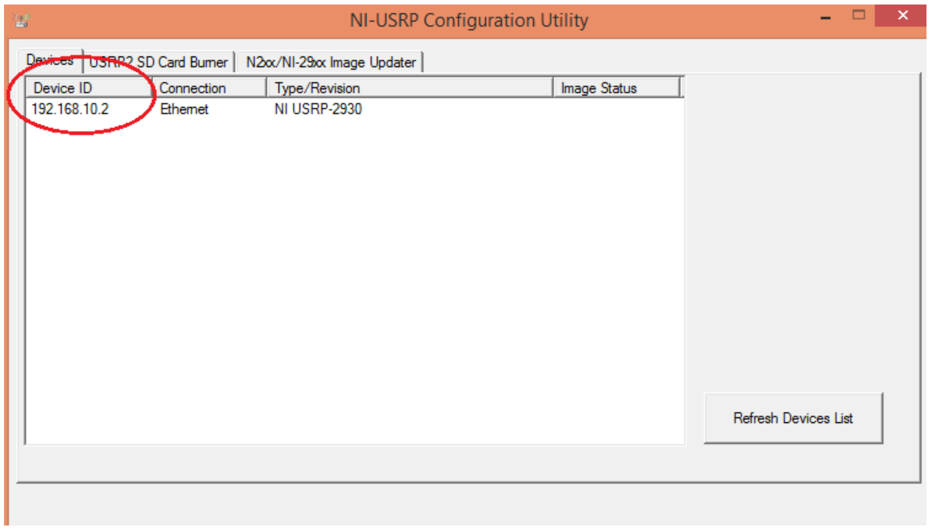
Preferred DNS server: . . .

Alternate DNS server: . . .

[ ] Validate settings upon exit

Figure 2.5: Steps for setting the IP

3. Launch the NI USRP Configuration Utility[3] to find the Device Name for your NI USRP device. See Fig. 2.6



NI-USRP Configuration Utility

Devices | USRP2 SD Card Burner | N2xx/NI-29xx Image Updater

Device ID	Connection	Type/Revision	Image Status
192.168.10.2	Ethernet	NI USRP-2930	

Refresh Devices List

Figure 2.6: The NI USRP Configuration Utility software

4. From the examples of the lobby in LabVIEW Communications (Help → Examples) , open the following NI USRP example:

[3]Available from iLearn, ask the lab instructor for more details

- Examples → Hardware Input and Output → NI USRP Host → Single Device → Single Channel → Continuous → RX Continuous Async

5. Give the example a project name and click Create.

6. From the example, open another NI USRP example:

   - File → Examples → Hardware Input and Output → NI USRP Host → Single Device → Single Channel → Continuous → TX Continuous Async.

7. Give the example a project name and click Create.

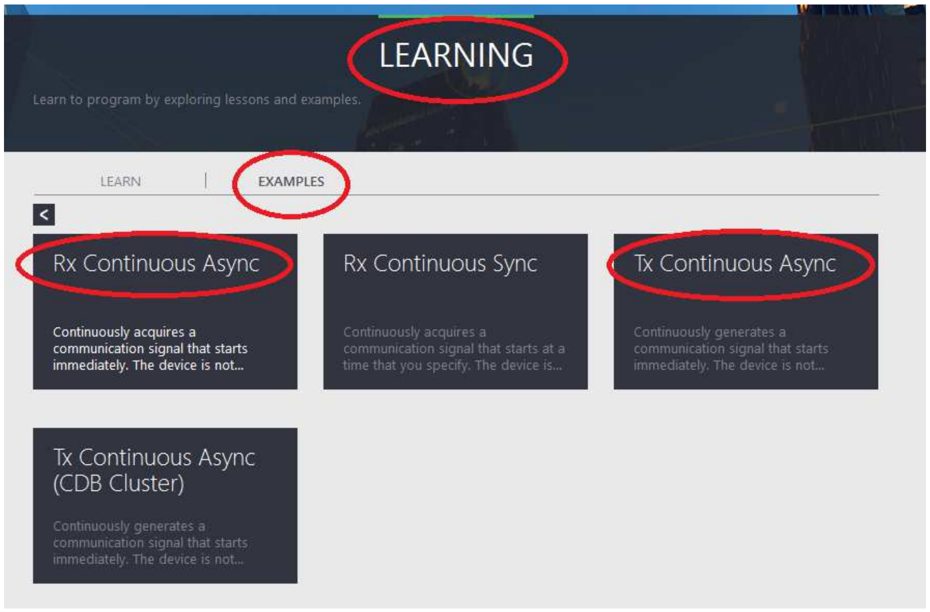

LEARNING
Learn to program by exploring lessons and examples.
LEARN
EXAMPLES
Rx Continuous Async
Continuously acquires a communication signal that starts immediately. The device is not...
Rx Continuous Sync
Continuously acquires a communication signal that starts at a time that you specify. The device is...
Tx Continuous Async
Continuously generates a communication signal that starts immediately. The device is not...
Tx Continuous Async (CDB Cluster)
Continuously generates a communication signal that starts immediately. The device is not...

Figure 2.7: RX and TX Continuous Async Projects.

8. On the Tx Continuous Aysnc example (referred to as the transmitter program), enter the device name (The USRP IP) and note the value of the tone frequency control. This program generates a single frequency tone at baseband and sends it to the USRP.

9. Run the transmitter program.

10. On the Rx Continuous Async example (referred to as the receiver program) example window, enter the same device name as the transmitter and change the Active Antenna to RX2.

11. Run the receiver program and analyze the Baseband Power Spectrum Graph. You should see a spike near the center of the graph. This is the single tone that was generated by the transmitter.

12. Without changing the value of the Carrier Frequency control on the receiver or transmitter program, and by considering a tone frequency in the transmitter, “move” the location of the single tone on the Baseband Power Spectrum graph to 150 kHz. Save a screenshot of your results.

13. Open a loopback cable and connect a attenuator between TX and RX (See below figure). Run both transmitter and receiver programs. Evaluate the effect of adding attenuator on the PSD graph. Take a screenshot of PSD graph before and after adding attenuator.

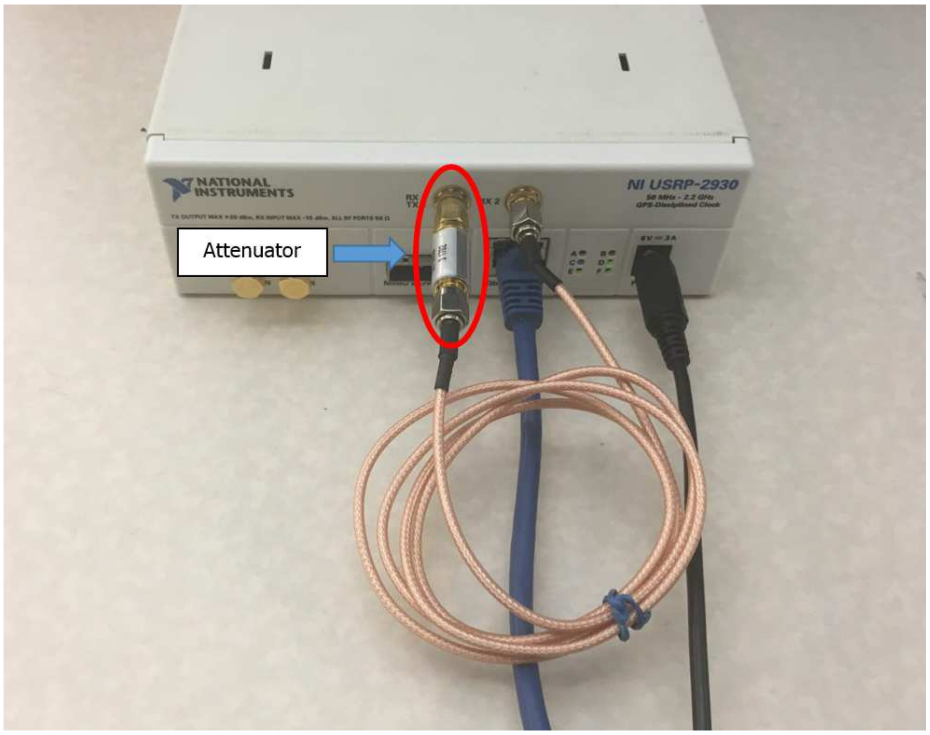


Figure 2.8: Adding attenuator between TX and RX.

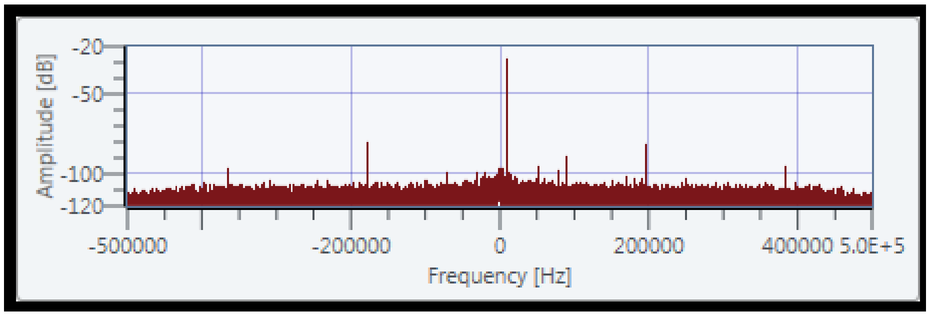


Figure 2.9: Baseband Power Spectrum

Note: Changes made to the program do not apply while it is running. You should stop the program and start it for change to take effect.

2.3.1 Questions

1. On the Diagram of the receiver program, write the name of each node (sometimes called function or block) and its purpose on the block diagram. (Such as niUSRP Open Rx Session creates a session handle to the device for other functions). You can use the "Context Help" to find out the function of each block.

2.4 Report

Submit all of the answers to the Questions section above.

LAB 3

Fourier Analysis

3.1 Objective

The aim of this experiment is to investigate the Fourier transforms of periodic waveforms, and using harmonic analysis of Fourier transforms to gain information about the frequencies present. Sine waves, square waves and rectangular waves and their Fourier transforms obtained and compared with those predicted by the theory. Our aims in this experiment are;

- To gain an intuitive understanding of Fourier series.
- To gain an intuitive understanding of Fourier transforms.
- To analyse various waveforms and explain their Fourier transforms.

3.2 Fourier Series

Fourier series are generally used to express complicated functions as a series of sine and cosine functions, which are relatively easy to manipulate mathematically. If an input can be expressed as a sum of sines and cosines (or equivalently, a sum of shifted sines or cosines); the response of the system to the input is the sum of the responses of the system to the sine and cosine inputs. In this lab, you will decompose a square wave input into a series of sine waves and construct the response of the system to the square wave as the sum of the responses to the series of sine waves.

In general, if $f(t)$ is a bounded, periodic function with period $T_0 = T$, is continuous (except for a finite number of jump discontinuities), and has a finite number of maxima and minima, $f(t)$ can be represented by a Fourier series given by:

$$\bar{f}(t) = a_0 + \sum_{n=1}^{\infty} a_n cos(\frac{2\pi nt}{T}) + \sum_{n=1}^{\infty} b_n sin(\frac{2\pi nt}{T}) \tag{3.1}$$

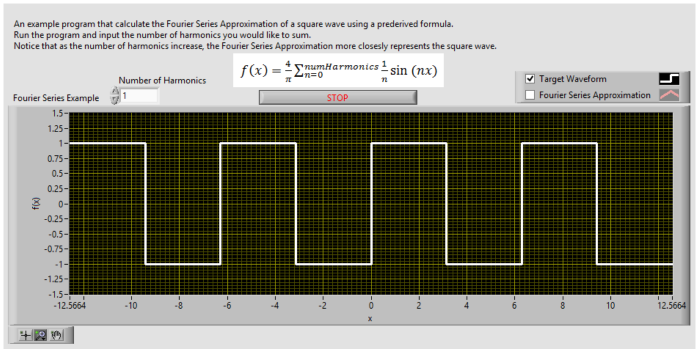
where

$$\begin{aligned} a_0 &= \frac{1}{T}\int_{-T/2}^{T/2} f(t)dt \\ a_n &= \frac{2}{T}\int_{-T/2}^{T/2} f(t)cos(\frac{2\pi nt}{T})dt \\ b_n &= \frac{2}{T}\int_{-T/2}^{T/2} f(t)sin(\frac{2\pi nt}{T})dt \end{aligned} \tag{3.2}$$

These are known as the Euler formulae for the Fourier coefficients, and the coefficients and give an idea of how much of any particular frequency is present in a function's Fourier series. The first formula given above is known as the simple Fourier series, which can then be extended to the complex Fourier series with complex coefficients. Although this definition sounds quite restrictive, it encompasses almost all functions that are of practical interest in engineering.

As you can see from (3.1), the coefficients a_n are associated with cosines, which are even functions ($f(t) = f(-t)$) while the b_n are associated with odd functions ($f(t) = -f(-t)$). Therefore it is not surprising that if $f(t)$ is an odd function, all the a_n (the coefficients for the even functions) are zero. Likewise, for an even function all the b_n are zero.

In this laboratory we will be dealing with the special case when $f(t)$ is an odd square wave as shown below in Fig. 3.1.


An example program that calculate the Fourier Series Approximation of a square wave using a prederived formula.
Run the program and input the number of harmonics you would like to sum.
Notice that as the number of harmonics increase, the Fourier Series Approximation more closely represents the square wave.

$f(x) = \frac{4}{\pi}\sum_{n=0}^{numHarmonics}\frac{1}{n}\sin(nx)$

Number of Harmonics: 1

☑ Target Waveform
☐ Fourier Series Approximation

Figure 3.1: Square Wave Input

Since the function is odd, the a_n will be 0, and the b_n are given by

$$b_n = \frac{2}{T}\int_0^T f(t)sin(\frac{2\pi nt}{T})dt \tag{3.3}$$

To prove this, you can multiply both sides of (3.1) by $\sin(n\pi t/T)$ and integrate from $-T$ to T. The function $f(t)$ is then given by:

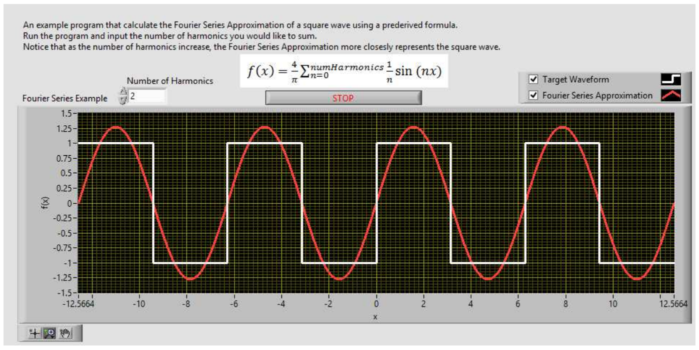
$$\bar{f}(t) = \sum_{n=1}^{\infty} b_n sin(\frac{2\pi nt}{T}), \tag{3.4}$$

with:

$$b_n = \begin{cases} 0, & \text{n even} \\ \dfrac{4}{n\pi}, & \text{n odd} \end{cases} \tag{3.5}$$

where T is half the period of the square wave. In Fig. 3.2, the square wave is shown, along with its first harmonic, which has an amplitude corresponding to b_1. Fig. 3.3 shows the third harmonic multiplied by b_3, and the sum of the first and the third harmonics. You can see that the sum of harmonics is beginning to approach the square wave.

The approximation is even better as we add more and more harmonics, up to the 11th harmonic in Fig. 3.4.

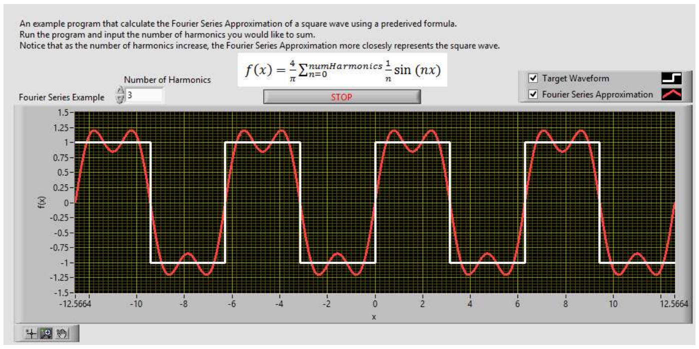


An example program that calculate the Fourier Series Approximation of a square wave using a prederived formula.
Run the program and input the number of harmonics you would like to sum.
Notice that as the number of harmonics increase, the Fourier Series Approximation more closely represents the square wave.

$f(x) = \frac{4}{\pi}\sum_{n=0}^{numHarmonics}\frac{1}{n}\sin(nx)$

Number of Harmonics: 2

Figure 3.2: Square Wave Input and First Harmonic of the Square Wave

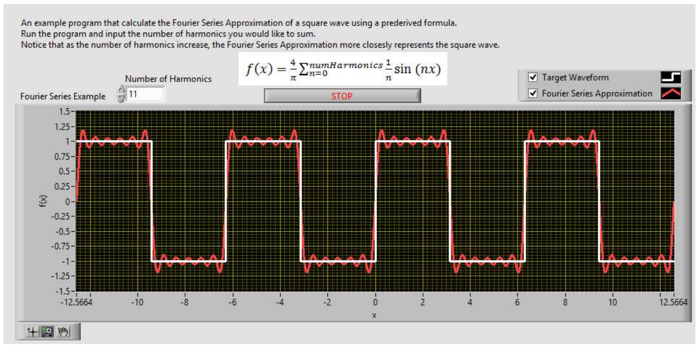


An example program that calculate the Fourier Series Approximation of a square wave using a prederived formula.
Run the program and input the number of harmonics you would like to sum.
Notice that as the number of harmonics increase, the Fourier Series Approximation more closely represents the square wave.

$f(x) = \frac{4}{\pi}\sum_{n=0}^{numHarmonics}\frac{1}{n}\sin(nx)$

Number of Harmonics: 3

Figure 3.3: Square Wave Input, First and Third Harmonics, and their Sum



An example program that calculate the Fourier Series Approximation of a square wave using a prederived formula.
Run the program and input the number of harmonics you would like to sum.
Notice that as the number of harmonics increase, the Fourier Series Approximation more closely represents the square wave.

$f(x) = \frac{4}{\pi}\sum_{n=0}^{numHarmonics}\frac{1}{n}\sin(nx)$

Number of Harmonics: 11

Figure 3.4: Square Wave Input and the Sum of the eleven Harmonics of the Square Wave

3.2.1 Lab Procedure

A periodic wave can be simulated by summing the output of several sine wave generators where each generator is generating one harmonic of the periodic wave. The magnitude and phase of each generator must be separately controlled. Such a system model is called a Fourier Synthesizer and will be constructed to simulate a square waves in this experiment. The synthesizer generates the fundamental through the $n-th$ harmonics with separate magnitude and phase controls on each. The sum of all the harmonics is available as an output. In this Lab, we calculate and implement a Fourier Synthesizer for a Square Wave.

1. Write the Fourier Series of a square wave with odd symmetry and shown in Fig. 3.5

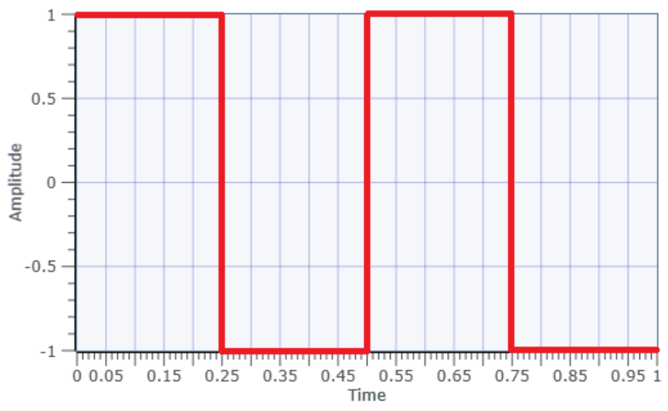


Figure 3.5: Square Wave

2. Open a LabVIEW Communication. Make a New Project. In the diagram panel, insert a “while loop” and “wait” block. Insert a “numeric constant”. Connect the “wait” block to the “numeric constant” and set its value to 100. Select the panel tab. Insert a “Stop button”. Again, select diagram tab and update it(insert the stop button). Connect the stop button to the “condition” part of the while loop. See Figure bellow:

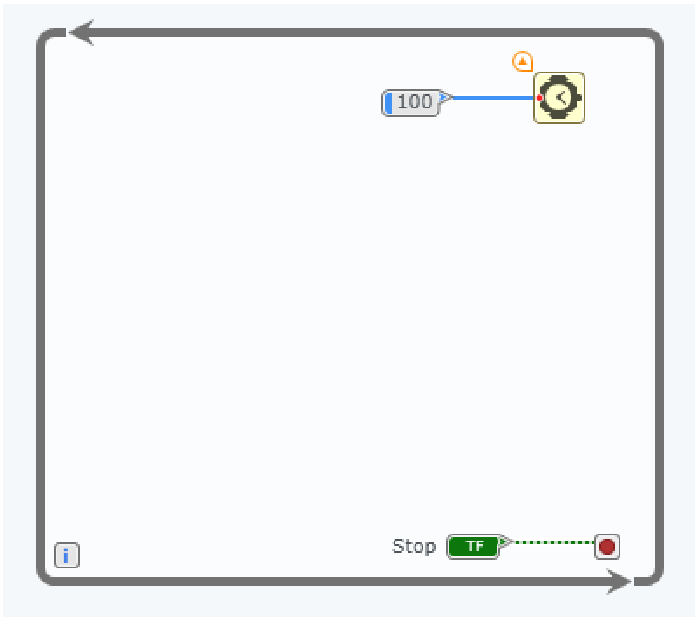


Figure 3.6: Basic Configuration of a Program in LabVIEW: While loop, Wait block and Stop button

3. Use "wave generator" block to construct a square wave shown in Fig. 3.8. The wave generator is placed in the following path: Analysis/Signal Processing/Generation/Wave generator.

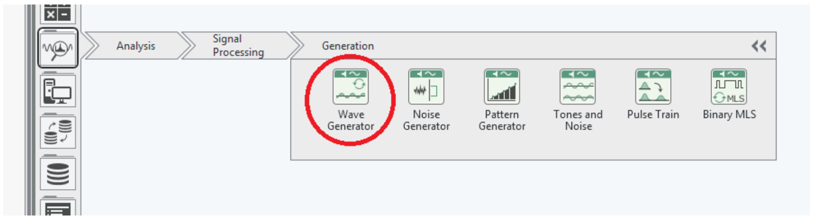


Figure 3.7: Wave generator block

The "wave generator" block in LabVIEW provides time varying periodic waveforms. The amplitude, frequency, and the phase of the signal are determined by the input parameters of the block. We will use a "wave generator" block as a harmonic generator in the Fourier Synthesizer. In order to construct a square wave select "Square" and "Waveform" when you insert the block into the diagram, as shown in figure bellow:

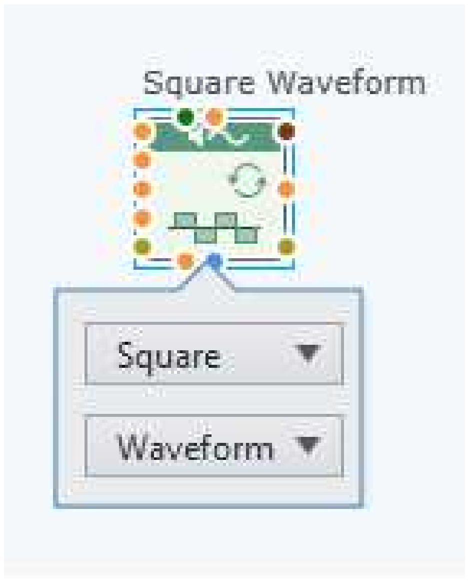
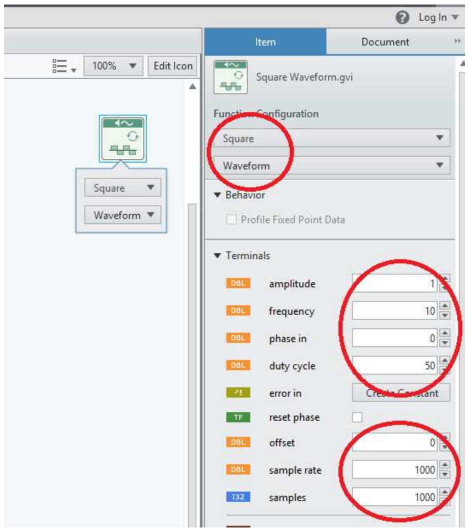


Figure 3.8: Square Wave generator

4. Please take this into consideration that you can change the settings of each block by clicking on it and changing the parameters in the properties window right hand side of the screen. (See Figure bellow)



Square Waveform.gvi

Function Configuration

Square

Waveform

Behavior

Profile Fixed Point Data

Terminals

DBL	amplitude	1
DBL	frequency	10
DBL	phase in	0
DBL	duty cycle	50
?!	error in	Create Constant
TF	reset phase	☐
DBL	offset	0
DBL	sample rate	1000
I32	samples	1000

Figure 3.9: Block Properties window

5. Use “numeric constant” block (find it from: Data Type/ Numeric/ Numeric Constant) to set the frequency and amplitude of the square wave as shown in Fig. 3.10. What are the desirable values for the frequency and amplitude in order to construct the square wave shown in Fig. 3.5 ?

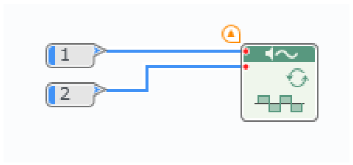


Figure 3.10: Setting the frequency and amplitude of the square wave generator

6. Click on the panel tab. Insert a "Graph". The "Graph" block in LabVIEW will be used to display both the input harmonics and the final output waveform. Again click on the diagram tab and update it(insert the "Graph" block into the diagram). Connect the "Graph" block to the output of the square wave generator. Run the Program. Make sure that your code generates the square wave shown in Fig. 3.5.

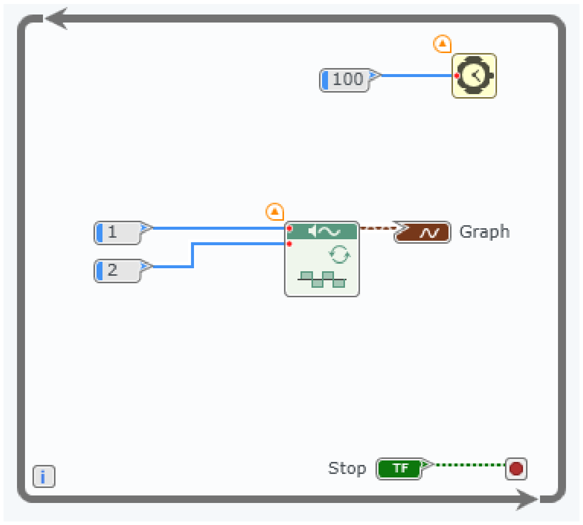


Figure 3.11: Generating the square wave

7. Insert two new "wave generator" into diagram. This time, select "Sine" to construct a Fourier Synthesizer model for the periodic square wave of Fig. 3.5. Based on your calculations in step 1, set the magnitude and frequency for these two inputs of the synthesizer, i.e., set the parameters in the Sine blocks which are used to generate the input signals of the synthesizer. Before you add these inputs to the synthesizer, observe the signal on a new Graph. Make sure that the display is consistent with the parameters you set. Use "Add" block (find it from: Data Type/ Numeric/ Add) to add the outputs of two sine generators. In order to display two waveforms in one Graph use "Build Array" block. Your diagram must be as same as Fig. 3.12. Also your pannel must be as same as Fig. 3.13. Start the simulation and observe the output waveform in the Graph.

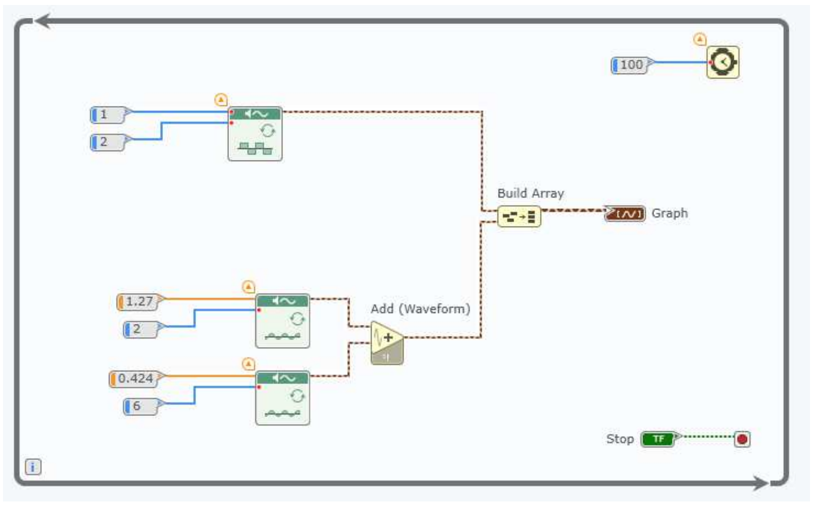


Figure 3.12: Fourier Synthesizer model for the periodic square wave

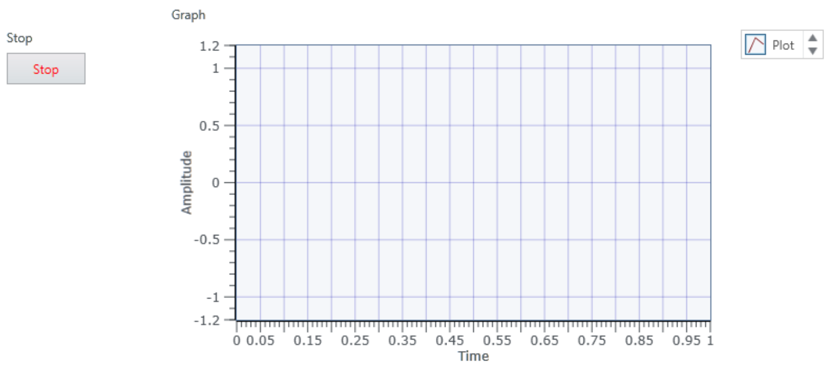


Figure 3.13: Designed Panel for Fourier model

8. Add second, third,..., and eleventh harmonics to the system, one at a time. To do this, you first need to insert "add" block, then insert another Sine Wave generator into the diagram(see Fig. 3.14). Of course you need to set the proper parameters for the Sine Wave, based on your previous calculation. Again, you need to observe the individual harmonics using the graph. Observe the change of the output waveform in the graph window when the higher order harmonics are added to the system. Compare your simulated result with the square wave shown in Fig. 3.5 by superimposing the two waveforms. Are they consistent? Can you find out how many harmonics you need to add to achieve a good accuracy?

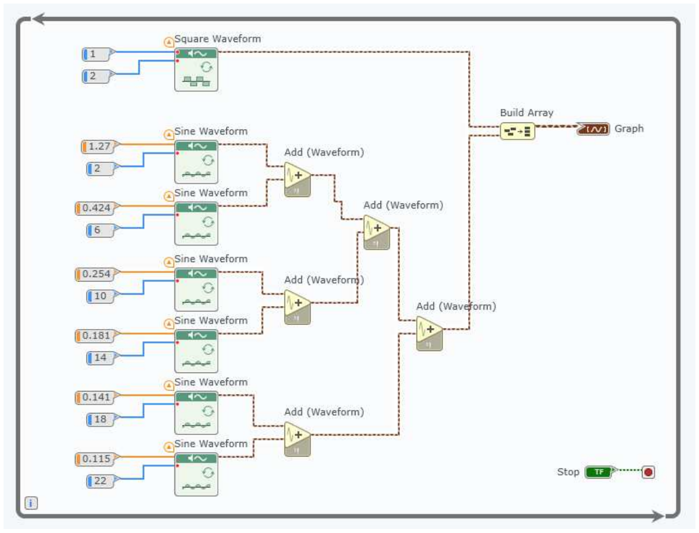


Figure 3.14: Fourier Synthesizer model for the periodic square wave

9. Print both the square wave and the synthesized square wave for your report.
10. Change the phase and magnitude of each harmonic and note that as far as wave shape is concerned, the phase of a harmonic is much more important than the magnitude. Try this step for a few times, print out a waveform which can support your conclusion based on your observation.

3.2.2 Report

Answer all questions and include the graphs and screen shot of the diagrams requested for the Fourier Synthesizer.

3.3 Fourier transform

The (3.1) is known as the simple Fourier series, which can then be extended to the complex Fourier series with complex coefficients. From this, we get the notion of the Fourier transform, which is a mathematical operation which decomposes a function into its constituent frequencies, which are analogous to the Fourier coefficients as before. The complex Fourier transform is given by the formula

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi jft}dt, \tag{3.6}$$

Where $X(f)$ is the Fourier transform of our function $x(t)$, f is frequency, and t is time. Fourier analysis can therefore be used to determine the amount of any given frequency in a function, and given a signal, it can be used to create a frequency spectrum of this signal.

In this experiments, a discrete Fourier transform is used, which obtains the Fourier transform of a function by sampling discrete data, ie. approximating a continuous function by collecting data at certain intervals.

3.3.1 Lab Procedure

1. By Euler's Theorem, a sine wave can be represented as

$$sin(2\pi f_0 t) = \frac{e^{j2\pi f_0 t} - e^{-j2\pi f_0 t}}{2j} \tag{3.7}$$

   Question 1: Prove that the Fourier transform of this sine wave is as follows:

$$\begin{aligned} X(f) &= \int_{-\infty}^{\infty} \frac{e^{j2\pi f_0 t} - e^{-j2\pi f_0 t}}{2j} . e^{-j2\pi f t} dt \\ &= \frac{1}{2j}(\delta(f - f_0) - \delta(f + f_0)) \end{aligned} \tag{3.8}$$

   Where $\delta(x)$ is the Dirac Delta function.

2. Open a LabVIEW Communication. Make a New Project. In the diagram panel, insert a "while loop" and "wait" block. Insert a "numeric constant". Connect the "wait" block to the "numeric constant" and set its value to 100. Select the panel tab. Insert a "Stop button". Again, select diagram tab and update it(insert the stop button). Connect the stop button to the "condition" part of the while loop. See Fig. 3.6

3. Click on the panel tab. Insert "Stop Button", three "Numeric Control"s for controlling the frequency, phase and amplitude of the sine wave. Also insert two "Graph"s. See figure bellow:

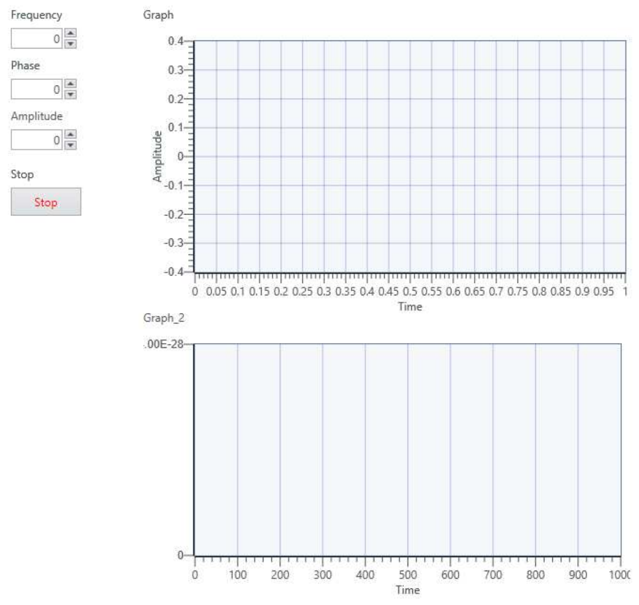


Figure 3.15: Panel Tab for the Fourier Transform Program

4. Click on the diagram tab. Insert a "wave generator" and select the sine output. Connect the inputs of the sine generator to the terminals as follows:

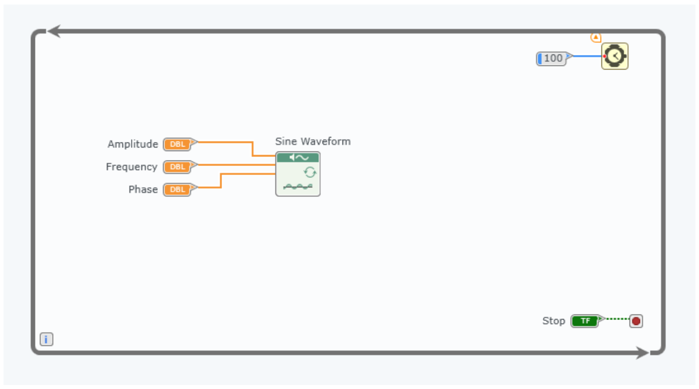


Figure 3.16: Controlling the inputs of the sine generator

5. Use "Power Spectrum" block to calculate the fourier transform of the sine wave. Connect the outputs of the wave generator and power spectrum to the two separate graphs. See figure bellow:

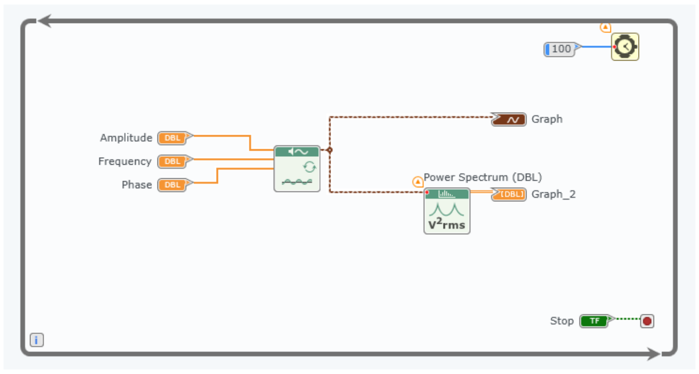


Figure 3.17: Diagram for Fourier Transform

6. Start the simulation and observe the output waveforms in both Graphs.
7. Change the phase and magnitude and amplitude of the sine wave. Try this step for a few times.
8. Compare your simulated result with the (3.8). Are they consistent?
9. Print both the sine wave and the fourier transform wave for your report.

3.3.2 Report

Answer all questions and include the graphs and screen shot of the diagrams requested for the Fourier Synthesizer.

LAB 4

Double-Sideband Suppressed-Carrier

4.1 Objective

This laboratory exercise introduces suppressed-carrier modulation. A simple scheme for phase and frequency synchronization is introduced in implementing the demodulator.

4.2 Background

4.2.1 Double-Sideband Suppressed-Carrier

In suppressed-carrier schemes the carrier is simply not transmitted. There are two common suppressed-carrier techniques in use, double-sideband suppressed-carrier (DSBSC) and single-sideband (SSB). Double-sideband suppressed-carrier modulation is identical to AM, except that the carrier is omitted.

If $m(t)$ is a baseband "message" signal and $\cos(2\pi f_c t)$ is a "carrier" signal at carrier frequency f_c, then we can write the DSB-SC signal $g(t)$ as

$$g(t) = Am(t)\cos(2\pi f_c t), \tag{4.1}$$

For the special case in which $m(t) = m_p \cos(2\pi f_m t)$, we can write

$$\begin{aligned} g(t) &= Am_p \cos(2\pi f_m t)\cos(2\pi f_c t) \\ &= \frac{Am_p}{2}\cos\left[2\pi(f_c - f_m)t\right] + \frac{Am_p}{2}\cos\left[2\pi(f_c + f_m)t\right], \end{aligned} \tag{4.2}$$

The two terms in (4.2) represent the lower and upper sidebands, respectively. There is no carrier term at frequency f_c . Figure 1 is a plot of a 20 kHz carrier modulated by a 1 kHz sinusoid using DSB-SC modulation.

When the DSB-SC signal arrives at the receiver, it has the form

$$r(t) = Dm(t)\cos(2\pi f_c t + \theta), \tag{4.3}$$

where D is a constant, usually much smaller than A , and the angle θ represents the difference in phase between the transmitter and receiver carrier oscillators. If the receiver's carrier oscillator (the "local" oscillator) is set to the same frequency as the transmitter's carrier oscillator, the USRP will generate the two demodulated signals

$$\begin{aligned} r_I(t) &= \frac{D}{2}m(t)\cos(\theta), \\ r_Q(t) &= \frac{D}{2}m(t)\sin(\theta), \end{aligned} \tag{4.4}$$

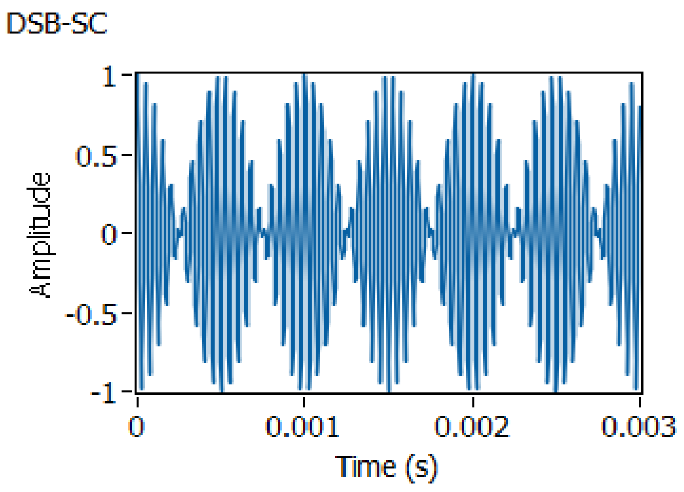


Figure 4.1: Double-Sideband Suppressed-Carrier Modulation

For understanding above equation, Please recall equation (5.7) in the previous lab. As it was mentioned in the previous lab, if the baseband signal is expressed as

$$\tilde{g}(nT_x) = g_I(nT_x) + jg_Q(nT_x), \tag{4.5}$$

then the signal transmitted by the USRP is

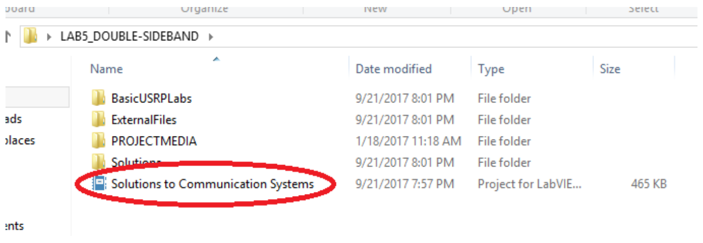
$$g(t) = Ag_I(t)\cos(2\pi f_c t) - Ag_Q(t)\sin(2\pi f_c t), \tag{4.6}$$

The Fetch Rx Data has been summarized in Appendix A.

4.3 Pre-Lab

4.3.1 Transmitter

1. A template for the transmitter has been provided in the folder: LAB5_DOUBLE-SIDEBAND → Solutions to Communication Systems.(See below figures)



LAB5_DOUBLE-SIDEBAND

Name	Date modified	Type	Size
BasicUSRPLabs	9/21/2017 8:01 PM	File folder	
ExternalFiles	9/21/2017 8:01 PM	File folder	
PROJECTMEDIA	1/18/2017 11:18 AM	File folder	
Solutions	9/21/2017 8:01 PM	File folder	
Solutions to Communication Systems	9/21/2017 7:57 PM	Project for LabVIE...	465 KB

Figure 4.2: DSBSC Template Program

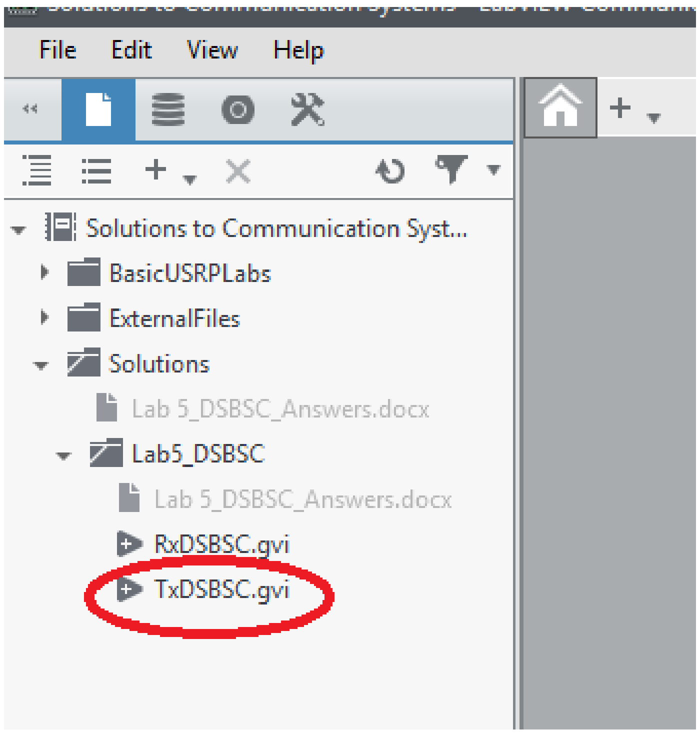


Figure 4.3: DSBSC Transmitter Template

This template contains the four interface functions along with a “message generator” that is set to produce a message signal consisting of three tones. The three tones are initially set to 1, 2, and 3 kHz, but these frequencies can be changed using front-panel controls. Your task is to evaluate how this diagram produces an DSB-SC signal, and how the DSB-SC signal passes into the while loop to the Write Tx Data function. Hint: The DSB-SC signal you generate will be $g_I(nT)$. For $g_Q(nT)$ set up an array the same length as $g_I(nT)$ containing all zeroes. Then combine the two into a single complex array $\tilde{g}(nT) = g_I(nT) + jg_Q(nT)$.

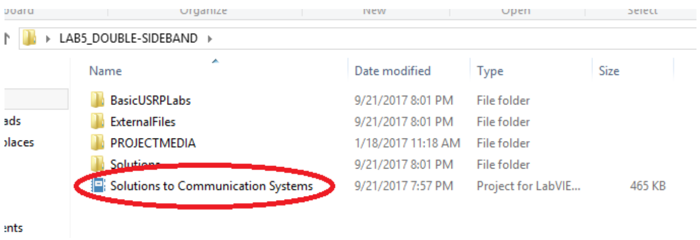
Notes:

- The message generator creates a signal that is the sum of a set of sinusoids of equal amplitude. You can choose the number of sinusoids to include in the set, you can choose their frequencies, and you can choose their common amplitude. In this template the message generator has been provided with a “seed.” This causes the initial phase angles of the sinusoids to be the same every time you run the program. As a result, the same message will be generated every time, which is useful to aid debugging. To restore random behavior, set the seed to -1.
- There is a practical constraint imposed by the D/A converters in the USRP: Scale the signals you generate so that the peak value of $|\tilde{g}(nT)|$

does not exceed $+/-1$. (Check out the Quick Scale 1D function in the External Files folder.)

4.3.2 Receiver

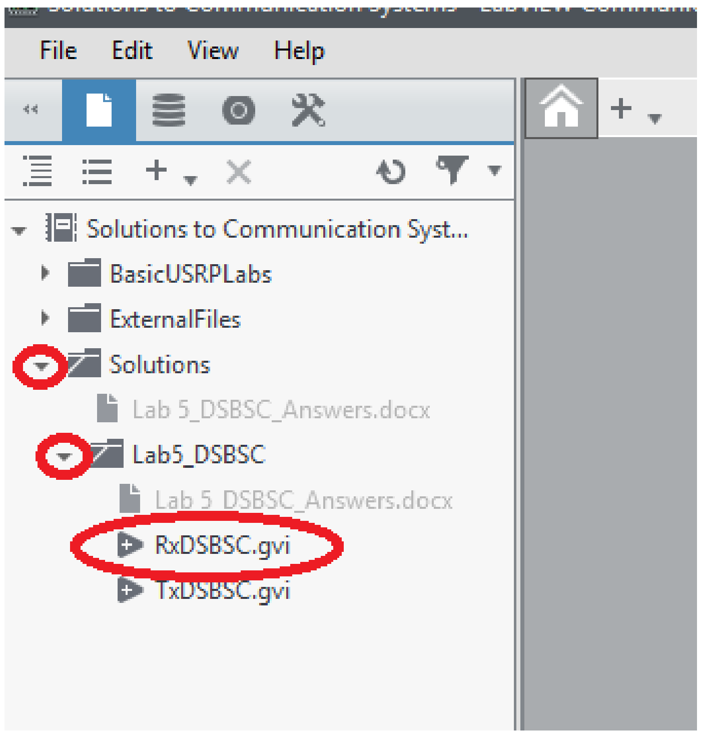
1. A template for the transmitter has been provided in the folder: LAB5_DOUBLE-SIDEBAND → Solutions to Communication Systems.(See below figures)



LAB5_DOUBLE-SIDEBAND

Name	Date modified	Type	Size
BasicUSRPLabs	9/21/2017 8:01 PM	File folder	
ExternalFiles	9/21/2017 8:01 PM	File folder	
PROJECTMEDIA	1/18/2017 11:18 AM	File folder	
Solutions	9/21/2017 8:01 PM	File folder	
Solutions to Communication Systems	9/21/2017 7:57 PM	Project for LabVIE...	465 KB

Figure 4.4: DSBSC Template Program



- Solutions to Communication Syst...
  - BasicUSRPLabs
  - ExternalFiles
  - Solutions
    - Lab 5_DSBSC_Answers.docx
    - Lab5_DSBSC
      - Lab 5_DSBSC_Answers.docx
      - RxDSBSC.gvi
      - TxDSBSC.gvi

Figure 4.5: DSBSC Receiver Template

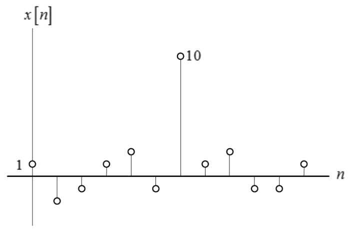
This template contains the six interface functions along with a waveform graph on which to display your demodulated output signal.

Review Appendix A and Appendix B. Show how the program demodulates the complex array returned by the Fetch Rx Data function and displays the result. Find the phase synchronization in this receiver.

4.3.3 Questions

1. The graph below shows a sequence of samples having values

   $x = [1 \quad -2 \quad -1 \quad 1 \quad 2 \quad -1 \quad 10 \quad 1 \quad 2 \quad -1 \quad -1 \quad 1]$ values not shown are zero. Note that there is a single outlying sample at index $n = 6$.

   Suppose we have a simple median filter than produces an output $y[n]$ given by

   $y[n] = median\{x[n-2], \dots, x[n+2]\}, \quad n = 0, \dots, 11$

   Find $y[n]$ for the sample sequence $x[n]$ shown

2. Suppose the receiver's carrier oscillator differs in frequency from the transmitter's oscillator by a small offset Δf . Modify (4.4) and (4.7) to include the frequency offset.

4.3.4 Lab Procedure

1. Open both TxDSBSC.gvi and RxDSBSC.gvi described in pre-lab. The diagram and panel of these programs should look like bellow figures:

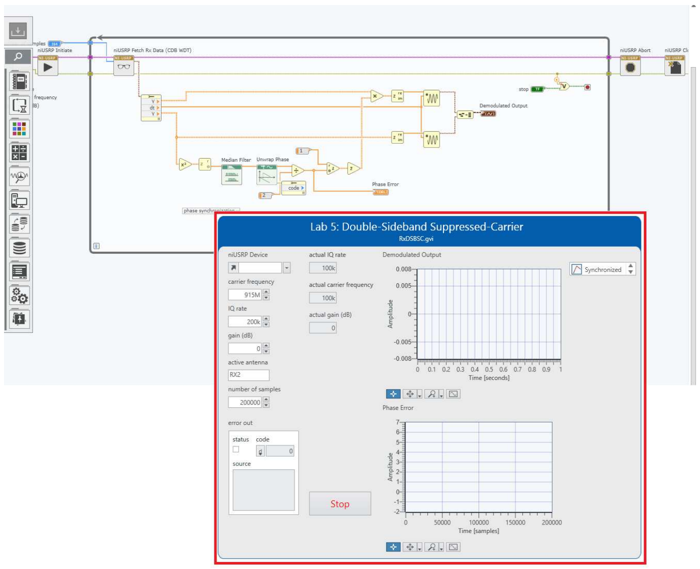


Figure 4.6: Receiver panel and diagram

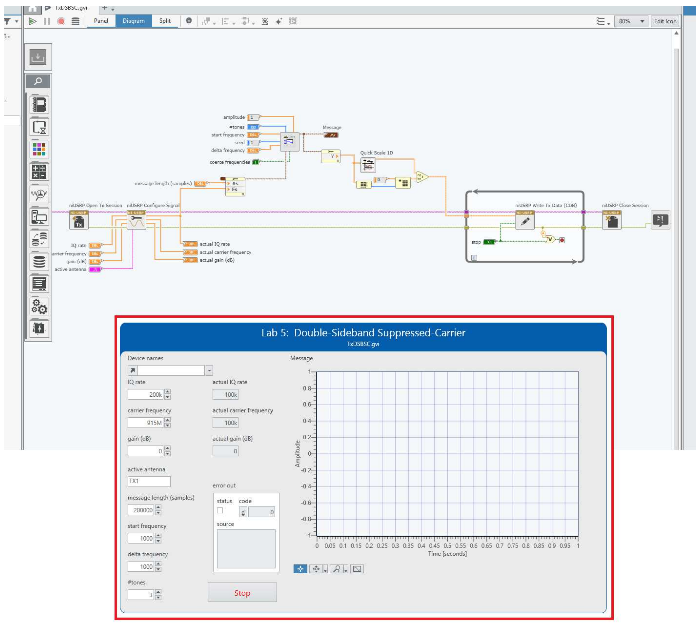


Figure 4.7: Transmitter panel and diagram

2. Connect a loopback cable and attenuator between the TX 1 and RX 2 connectors. Connect the USRP to your computer and plug in the power to the USRP. Run LabVIEW and open the transmitter and receiver that you created in the prelab.

3. Ensure that the transmitter is set up to use

Carrier Frequency: 915 MHz

Device names: IP address of the USRP

IQ Rate: Not critical. 200 kHz

Gain: Not critical. 0 dB

Active Antenna: TX1

Message Length: 200,000 samples gives a good block of data to work with.

Start Frequency, Delta Frequency, Number of Tones: Not critical, but keep the highest frequency below 5 kHz. Three tones seems to work well.

4. Ensure that the receiver is set up to use

   Carrier Frequency: 915 MHz

   Device names: IP address of the USRP

   IQ Rate: 200 kHz

   Gain: Not critical. 0 dB

   Active Antenna: RX2

   Number of Samples: Same value as the transmitted message length.

   Run the transmitter, then run the receiver. After a few seconds, stop the receiver using the STOP button, then stop the transmitter (using the STOP button). Use the horizontal zoom feature on the graph palette to expand the "message" waveform in the transmitter and the "demodulated output" waveform in the receiver. Both waveforms should be identical, except for scaling.[1] Save the screen shot of both transmitted and received signals.

5. Based on the diagram of the receiver, discuss how the receiver computes $R_I(t)$ without phase synchronization and $R_I(t)$ with phase synchronization. Both outputs are plotted on the same graph (blue and red). Run the receiver several times and observe the outputs. Can you see the effect of the $\cos(t)$ term on the unsynchronized output?

6. Try using your AM receiver from AM lab to demodulate the DSB-SC signal. Note that you will need to offset the transmitter frequency to 915.1 MHz. Run the transmitter and receiver. Take a screenshot of both the transmitted message and the demodulated output. Be sure to expand the time base so that the waveforms can be clearly seen. Was the envelope detector in the AM receiver able to correctly demodulate the DSB-SC signal? Save the screen shot of both the transmitted and received signals. Evaluate the received signal.

[1]The demodulated output may be inverted. This is a consequence of squaring the signal in the phase synchronization process. An error of $\pm 2\pi$ in the angle 2θ is no error at all, but when the angle is divided by two, the error becomes $\pm\pi$.

7. The phase synchronizer which has been summarized in appendix B, can also correct for modest frequency offsets. Use the DSB-SC transmitter and receiver, and offset the carrier frequency of the transmitter by 1 KHz (Set carrier frequency to 915.001 MHz). Run the transmitter and receiver. Take a screenshot of the transmitted message and the synchronized demodulated output. Be sure to expand the time base so that the waveforms can be clearly seen. Verify that the synchronized demodulated output is correct, except possibly for being inverted.

   Repeat for frequency offsets of 5 KHz and 10 KHz (carrier frequency equals to 915.005 and 915.01 MHz). Can your phase synchronizer handle the 10 KHz case?

4.4 Report

4.4.1 Prelab

Evaluate how the diagram of transmitter produces an DSB-SC signal, and how the DSB-SC signal passes into the while loop to the Write Tx Data function. Evaluate how the diagram of the receiver estimate the phase offset. Explain why the phase estimator circuit can also correct for modest frequency offsets. Hand in documentation for your transmitter and receiver. To obtain documentation, print out legible screenshots of the front panel and block diagram. Submit your answers to the Questions at the end of the Prelab section.

4.4.2 Lab

Describe the effect on the demodulated signal of the $\cos(\theta)$ term, as described in Step 5. Submit the graphs required in Step 6. Discuss whether DSB-SC can be properly demodulated using an envelope detector. Submit the graphs required in Step 7. Comment on whether your phase synchronizer was able to compensate for frequency offsets of 100 Hz and 1 kHz.

4.5 Appendix A

The Fetch Rx Data provides these demodulated signals as a single complex-valued signal $\tilde{r}(t)$ given by

$$\begin{aligned}\tilde{r}(t) &= \frac{D}{2}m(t)\cos(\theta) + j\frac{D}{2}m(t)\sin(\theta), \qquad (4.7)\\ &= \frac{D}{2}m(t)e^{j\theta},\end{aligned}$$

It is tempting to suppose that the message $m(t)$ can be extracted from $\tilde{r}(t)$ by taking the magnitude of the complex signal. Unfortunately, the magnitude of $\tilde{r}(t)$ is

$$|\tilde{r}(t)| = \frac{D_r}{2}|m(t)|, \tag{4.8}$$

where the absolute value represents unwanted distortion of the message signal. It is more productive to use the in-phase (real part) signal $r_I(t)$ given in (4.4). The $\cos(\theta)$ factor of $r_I(t)$ represents a gain constant. Unfortunately, the value of this gain constant is not under user control, and might be small if θ turns out to have a value near $\pm\pi/2$. Moreover, if the receiver's oscillator and transmitter's oscillator differ slightly in frequency, then the phase error θ will change with time, causing $r_I(t)$ to fade in and out. The next section discusses how we can compensate for the $\cos(\theta)$ term.

4.6 Appendix B: Phase Synchronization

There are a number of techniques that can be used to eliminate the $\cos(\theta)$ phase-error term. The method we present here is simple and easy to implement in LabVIEW. The basic steps are

- Estimate θ
- Multiply $\tilde{r}(t)$ by $e^{-j\theta}$ to produce $\tilde{r}(t)e^{-j\theta} = \dfrac{D}{2}m(t)e^{j\theta}e^{-j\theta} = \dfrac{D}{2}m(t)e^{j0}$
- Take the real part :$Re\left\{\dfrac{D}{2}m(t)e^{j0}\right\} = \dfrac{D}{2}m(t)\cos(0) = \dfrac{D}{2}m(t)$

Estimating θ requires several steps. Note first that the phase angle of $\tilde{r}(t)$ will jump by $\pm\pi$ whenever $m(t)$ changes sign. To eliminate these phase jumps, start by squaring $\tilde{r}(t)$:

$$\tilde{r}^2(t) = \frac{D^2}{4}m^2(t)e^{j2\theta}, \tag{4.9}$$

Since the squared message $m^2(t)$ never changes sign, the phase jumps are eliminated. The angle 2θ can be extracted using a Complex to Polar function from the Data Types → Numeric → Complex palette. It turns out to be helpful at this point to smooth variations in 2θ caused by noise. The median filter from the External-Files folder does a good job. The default values can be accepted for the "left rank" and "right rank" parameters. Next the Unwrap Phase function from the Analysis → Signal Processing → Signal Operations palette will remove jumps of $\pm 2\pi$. Finally, dividing by two gives the desired estimate of the phase error θ . The block diagram in Figure 2 shows the entire phase synchronization process.

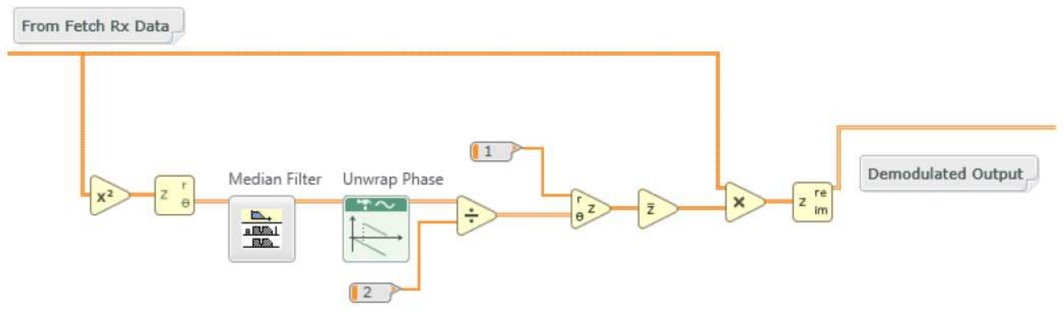


Figure 4.8: Phase Synchronization

LAB 5

Amplitude Modulation

5.1 Objective

This laboratory exercise has two objectives. The first is to gain a firsthand experience in actually programming the USRP to act as a transmitter and a receiver. The second is to investigate classical analog amplitude modulation and the envelope detector.

5.2 Background

5.2.1 Amplitude Modulation

Amplitude modulation (AM) is one of the oldest of the modulation methods. It is still in use today in a variety of systems, including, of course, AM broadcast radio. In digital form it is the most common method for transmitting data over optical fiber.

If $m(t)$ is a baseband "message" signal with a peak value m_p and $A\cos(2\pi f_c t)$ is a "carrier" signal at carrier frequency f_c , then we can write the AM signal $g(t)$ as:

$$g(t) = A\left[1 + \mu\frac{m(t)}{m_p}\right]\cos(2\pi f_c t), \tag{5.1}$$

where the parameter μ is called the "modulation index" and takes values in the range $0 \leq \mu \leq 1$ (0 to 100%) in normal operation. For the special case in which $m(t) = m_p\cos(2\pi f_m t)$, we can write

$$\begin{aligned} g(t) &= A\left[1 + \mu\cos(2\pi f_m t)\right]\cos(2\pi f_c t) \\ &= A\cos(2\pi f_c t) + \frac{A}{2}\mu\cos\left[2\pi(f_c - f_m)t\right] + \frac{A}{2}\mu\cos\left[2\pi(f_c + f_m)t\right], \end{aligned} \tag{5.2}$$

In the above expression the first term is the carrier, and the second and third terms are the lower and upper sidebands, respectively. Fig. 5.1 is a plot of a 20 kHz carrier modulated by a 1 kHz sinusoid at 50% and 100% modulation.

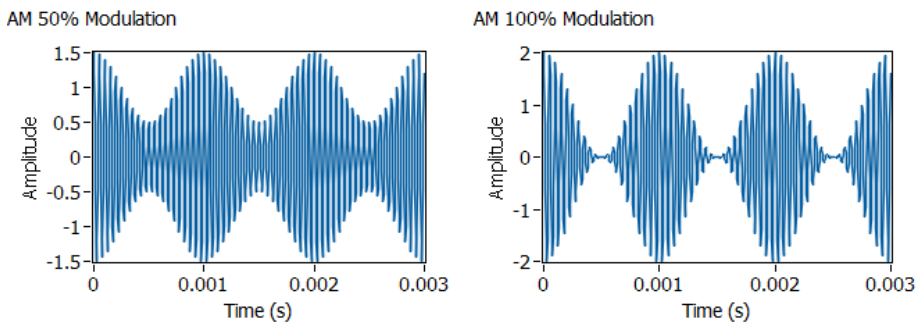


Figure 5.1: Amplitude Modulated Signals

When the AM signal arrives at the receiver, it has the form

$$r(t) = D\left[1+\mu\frac{m(t)}{m_p}\right]\cos(2\pi f_c t+\theta), \tag{5.3}$$

where the carrier amplitude D is usually much smaller than the amplitude A of the transmitted carrier and the angle θ represents the difference in phase between the transmitter and receiver carrier oscillators. We will follow a common practice and offset the receiver's oscillator frequency f_0 from the transmitter's carrier frequency f_c. This provides the signal

$$r_1(t) = D\left[1+\mu\frac{m(t)}{m_p}\right]\cos(2\pi f_{IF} t+\theta), \tag{5.4}$$

where the so-called "intermediate" frequency is given by $f_{IF} = f_c - f_0$.
(5.4) shows three important different between transmitted and received signal:

1. Difference in the amplitude: The Transmitted signal amplitude was A, while the received signal amplitude is D which is smaller than A.

2. Difference in the phase: The angle θ in (5.4) represents the difference in phase between the transmitter and receiver.

3. Difference in the frequency: The f_{IF} in (5.4) shows that the frequency of received signal differs from the frequency of transmitted signal. The frequency of transmitted signal is f_c while the frequency of received signal is $f_{IF} = f_c - f_0$.

The signal $r_1(t)$ can be passed through a bandpass filter to remove interference from unwanted signals on frequencies near f_c. Usually the signal $r_1(t)$ is amplified as well.

Demodulation of the signal $r_1(t)$ is most effectively carried out by an envelope detector. An envelope detector can be implemented as a rectifier followed by a lowpass filter. The envelope $A(t)$ of $r_1(t)$ is given by

$$A(t) = D\left[1+\mu\frac{m(t)}{m_p}\right] = D + \frac{D\mu}{m_p}m(t), \tag{5.5}$$

5.2.2 Setting up the USRP:Transmitter

LabVIEW interacts with the USRP transmitter by means of four functions located on the block diagram's palette under Hardware Interfaces → NI-USRP → Tx. Fig. 5.2 and Fig. 5.3 show the block diagram's palette and the basic transmitter structure respectively. Review this structure but you don't need to make them in LabVIEW (A template will give to you). This structure is the starting point for all of the laboratory exercises in this series.

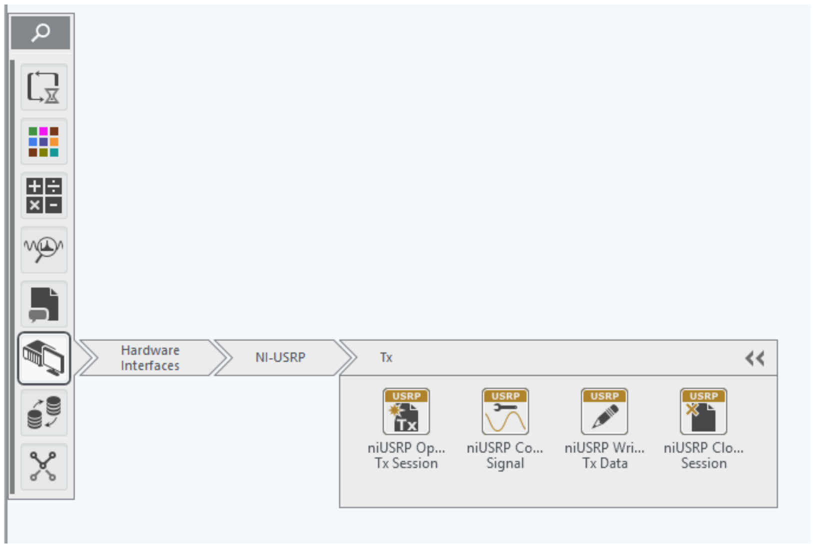


Figure 5.2: Block diagram's palette

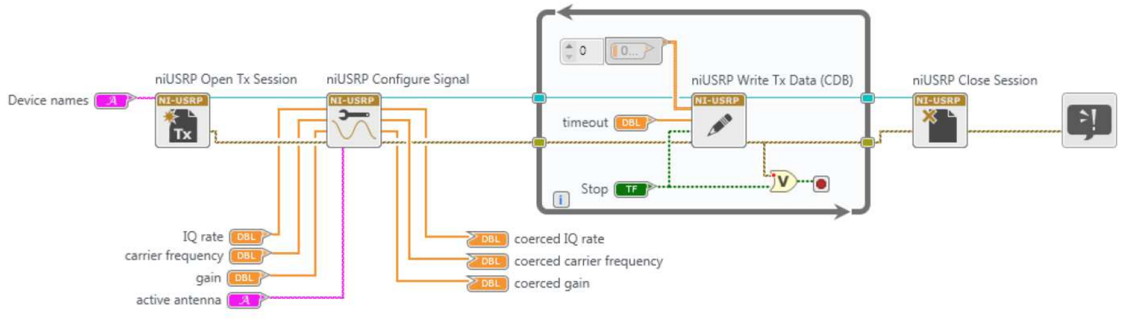


Figure 5.3: Transmitter Template

Open Tx Session initiates the transmitter session and generates a session handle and an error cluster that are propagated through all four functions. When you use this function, you must add a control called "device names" that you will use to inform LabVIEW of the IP address or resource name of the USRP.

Configure Signal is used to set parameter values in the USRP. Attach four controls and three indicators to this function as shown in the figure. To get started, set the IQ rate to 200 kSa/s (the lowest possible rate), the carrier frequency to 915.1 MHz, the gain to 0 dB, and the active antenna to TX1. When the function runs, the USRP will return the actual values of these parameters. These values will be displayed by the indicators you connected. Normally the actual parameter values will match the desired values, but if one or more of the desired values is outside the capability of the USRP, the nearest acceptable parameter value will be chosen, rather than returning an error.

Write Tx Data writes the baseband signal to the USRP for transmission. Placing this function in a while loop allows a block of baseband signal samples to be sent over and over until the "stop" button is pressed. Note that the while loop is programmed to terminate if an error is detected. Baseband signal samples can be provided to the Write Tx Data as either an array of complex numbers or as a complex waveform data type. The Configure ribbon at the top of LabVIEW Communications allows you to choose the data type. If the baseband signal is expressed as

$$\tilde{g}(nT_x) = g_I(nT_x) + jg_Q(nT_x), \tag{5.6}$$

then the signal transmitted by the USRP is

$$g(t) = Ag_I(t)\cos(2\pi f_c t) - Ag_Q(t)\sin(2\pi f_c t), \tag{5.7}$$

In this expression, the constant A is set by the "gain" parameter and f_c is the carrier frequency. The sampling interval T_x is the reciprocal of the "IQ rate." Note that the signal $g(t)$ produced by the USRP is a continuous-time signal; the discrete-to-continuous conversion is done inside the USRP.

Observe that the baseband signal $\tilde{g}(nT_x)$ is actually two baseband signals. By long-standing tradition, the real part $g_I(nT_x)$ is called the "in-phase" component of the baseband signal and the imaginary part $g_Q(nT_x)$ is called the "quadrature" component of the baseband signal. The AM signal that we will generate in this lab project uses only the in-phase component, with

$$g_I(t) = A\left[1 + \mu\frac{m(t)}{m_p}\right], \tag{5.8}$$

and

$$g_Q(t) = 0, \tag{5.9}$$

We will explore other modulation methods in subsequent lab projects that use both components.

Close Session terminates transmitter operation once the while loop ends. Note that the function should be terminated using the STOP button rather than with

"Abort Execution" on the toolbar. This is so that the Close Session function will be sure to run and will correctly close out the data structures that the function uses.

5.2.3 Setting up the USRP:Receiver

LabVIEW interacts with the USRP receiver by means of six functions located on the block diagram in Hardware Interfaces → NI-USRP → Rx. Fig. 5.4 shows the basic receiver structure. Review this structure but you don't need to make them in LabVIEW (A template will give to you) This structure is the starting point for all of the laboratory exercises in this series.

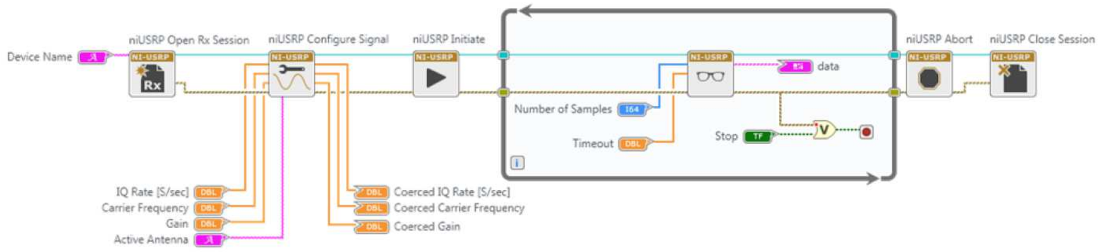


Figure 5.4: Receiver Template

Open Rx Session initiates the receiver session and generates a session handle and an error cluster that are propagated through all six functions. You must add a control called "device names" that you will use to inform LabVIEW of the IP address or resource name of the USRP.

Configure Signal has the same function as the corresponding function in the transmitter. Attach four controls and three indicators to this function as shown in the figure. This time, set the IQ rate to 1 MSa/s, the carrier frequency to 915.0 MHz, the gain to 0 dB, and the active antenna to RX2. When the function runs, the USRP will return the actual values of these parameters.

Initiate sends the parameter values you selected to the receiver and starts it running.

Fetch Rx Data retrieves the message samples received by the USRP. Placing this function in a while loop allows message samples to be retrieved one block at a time until the "stop" button is pressed. Note that the while loop is programmed to terminate if an error is detected. A "number of samples" control allows you to set the number of samples that will be retrieved with each pass through the while loop. In later lab projects in this series, we will not use the while loop, and will fetch only a single block of data from the receiver. Fetch Rx Data can provide message samples to the user as either an array of complex numbers or as a complex waveform data type. A pull-down tab allows you to choose the data type for the message samples.

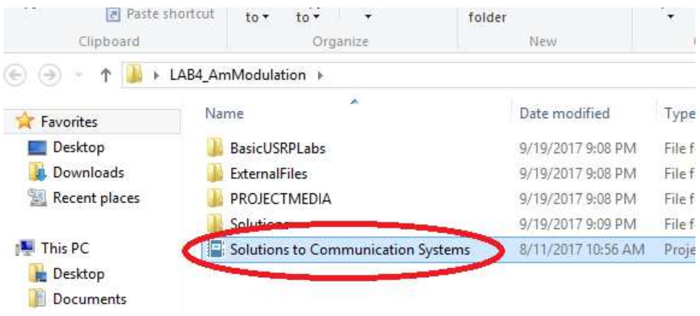
Abort stops the acquisition of data once the while loop ends.

Close Session terminates receiver operation. As noted above, use the STOP button to terminate execution so that Close Session will be sure to run.

5.3 Pre-Lab

5.3.1 Transmitter

1. A template for the transmitter has been provided in the folder: LAB4_AmModulation → Solutions to Communication Systems.(See below figures)



LAB4_AmModulation ▸

Name	Date modified	Type
BasicUSRPLabs	9/19/2017 9:08 PM	File f
ExternalFiles	9/19/2017 9:08 PM	File f
PROJECTMEDIA	9/19/2017 9:08 PM	File f
Solutions	9/19/2017 9:09 PM	File f
Solutions to Communication Systems	8/11/2017 10:56 AM	Proje

Figure 5.5: AM Template Program

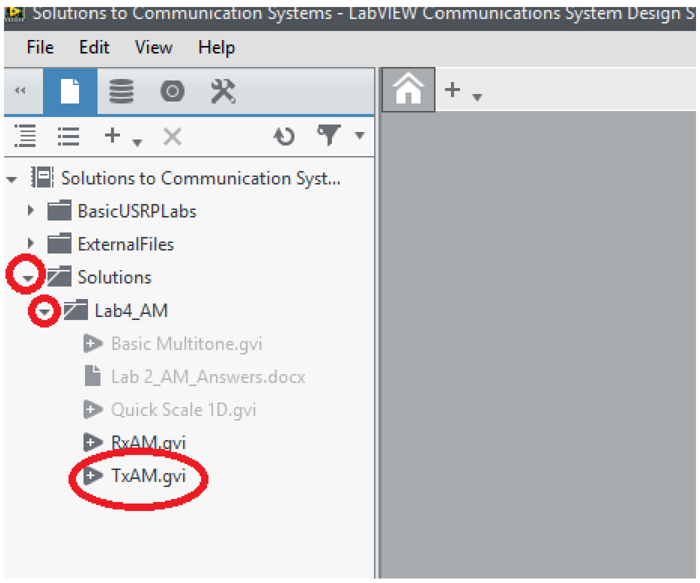


Figure 5.6: AM Transmitter Template

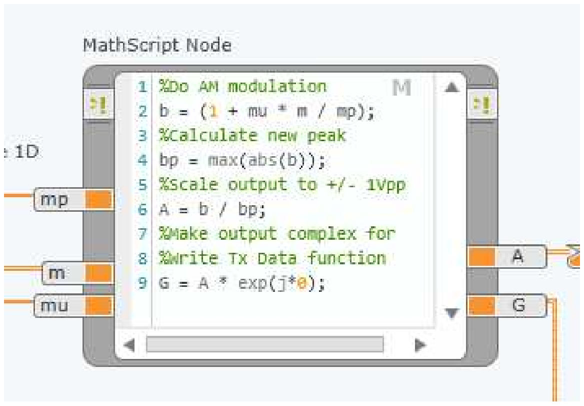
This template contains the four interface functions described in the Background section above along with a "message generator" (Basic Multitone) that is set to produce a message signal consisting of three tones. The three tones are initially set to 1, 2, and 3 kHz, but these frequencies can be changed using front-panel controls. Your task is to evaluate how this diagram produces an AM signal, and how the AM signal passes into the while loop to the Write Tx Data function. The modulation index is to be user-settable in the range $0 \leq \mu \leq 1$, and a front-panel control has been provided.

Hint: The AM signal you generate will be $g_I(nT)$. For $g_Q(nT)$ set up an array the same length as $g_I(nT)$ containing all zeros. Then combine the two into a single complex array $\tilde{g}(nT) = g_I(nT) + jg_Q(nT)$. You can use the MathScript node to implement the AM Modulation formula as follows:

Describe the functionality of this sample code.

Notes:

- The message generator creates a signal that is the sum of a set of sinusoids of equal amplitude. You can choose the number of sinusoids to include in the set, you can choose their frequencies, and you can choose their common amplitude. The initial phase angles of the sinusoids are chosen at random, however, and will be different every time you run the program. This will make the message signal look somewhat different every time you run the program.
- There is one practical constraint imposed by the D/A converters in the USRP: Scale the signals you generate so that the peak value of $|\tilde{g}(nT)|$



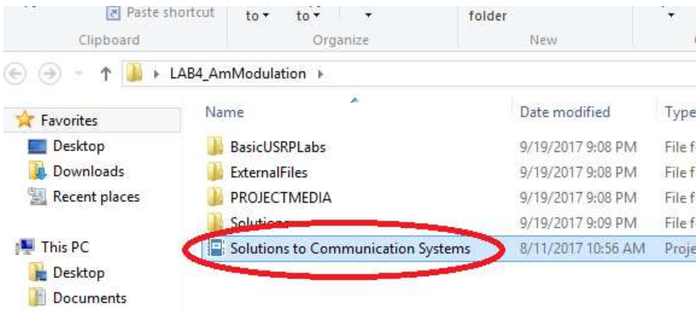
```
%Do AM modulation
b = (1 + mu * m / mp);
%Calculate new peak
bp = max(abs(b));
%Scale output to +/- 1Vpp
A = b / bp;
%Make output complex for
%Write Tx Data function
G = A * exp(j*0);
```

Figure 5.7: MathScript to produce AM signal

does not exceed $+/-1$. ($+/-1$ usually refers to full scale on the DAC in a device. Any value higher will result in clipping.)

5.3.2 Receiver

1. A template for the receiver has been provided in the folder: LAB4_AmModulation $\rightarrow$ Solutions to Communication Systems.(See below figures)



Name	Date modified	Type
BasicUSRPLabs	9/19/2017 9:08 PM	File f
ExternalFiles	9/19/2017 9:08 PM	File f
PROJECTMEDIA	9/19/2017 9:08 PM	File f
Solutions	9/19/2017 9:09 PM	File f
Solutions to Communication Systems	8/11/2017 10:56 AM	Proje

Figure 5.8: AM Template Program

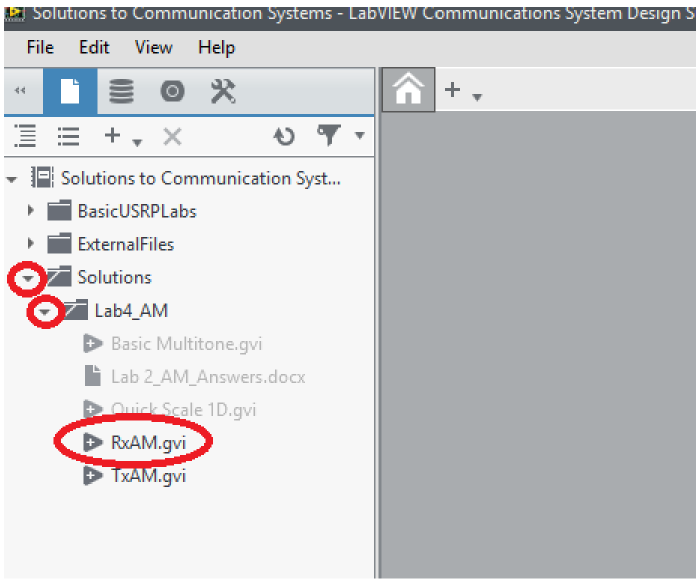


Figure 5.9: AM Receiver Template

This template contains the six interface functions described in the Background section above along with a waveform graph on which to display your demodulated output signal.

Follow the signal returned by Fetch Rx Data through a bandpass filter. How much is a high cutoff frequency and low cutoff frequency?

The default passband ripple of 0.1 dB is acceptable. The sampling frequency input to the filter should be the "actual IQ rate" obtained from the Configure Signal function.

2. To extract the envelope, the received signal is passed through a lowpass filter. This acts as a full-wave rectifier. What is the order of this filter? How much is the cutoff frequency of this filter?

   As was the case for the bandpass filter, the sampling frequency input to the lowpass filter should be the "actual IQ rate" obtained from the Configure Signal. The output of your lowpass filter should be connected to the Baseband Output graph.

5.3.3 Questions

1. Suppose the message $m(t)$ is given by

$$m(t) = \cos(2\pi 1000t) + \cos(2\pi 2000t) + \cos(2\pi 3000t), \tag{5.10}$$

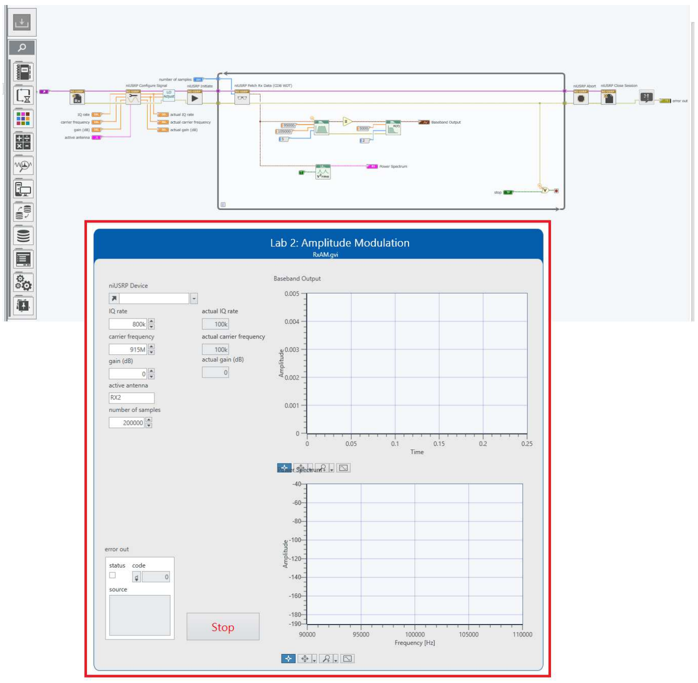
   Find and plot the one-sided power spectrum of $r_1(t)$ given by (5.4). Leave your answer in terms of D and μ.

2. For the message of (5.10), find and plot the power spectrum of $A(t)$ given by (5.5). Leave your answer in terms of D and μ.

3. Qualitatively, based on your answer to Question 1, what happens to the power in the carrier as the modulation index μ is varied? What happens to the power in the sidebands?

5.4 Lab Procedure

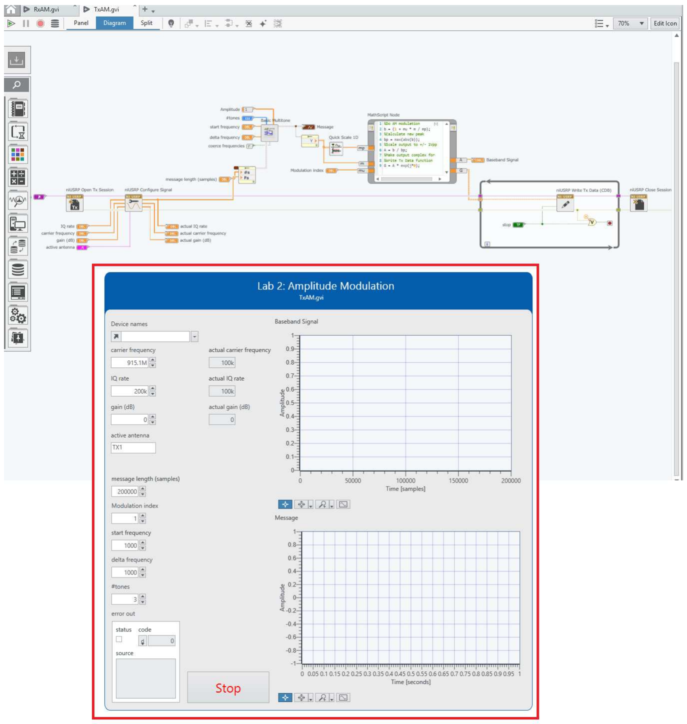
1. Open both TxAM.gvi and RxAM.gvi described in pre-lab. The diagram and panel of these programs should look like bellow figures:



Lab 2: Amplitude Modulation
RxAM.gvi

niUSRP Device
IQ rate: 800k
carrier frequency: 915M
gain (dB): 0
active antenna: RX2
number of samples: 200000
actual IQ rate: 100k
actual carrier frequency: 100k
actual gain (dB): 0
error out
status code: 0
source

Figure 5.10: Receiver panel and diagram



Device names

carrier frequency: 915.1M

actual carrier frequency: 100k

IQ rate: 200k

actual IQ rate: 100k

gain (dB): 0

actual gain (dB): 0

active antenna: TX1

message length (samples): 200000

Modulation index: 1

start frequency: 1000

delta frequency: 1000

#tones: 3

error out: status, code 0, source

Figure 5.11: Transmitter panel and diagram

2. Connect a loopback cable and attenuator between the TX 1 and RX 2 connectors. Connect the USRP to your computer and plug in the power to the USRP. Run LabVIEW and open the transmitter and receiver functions that you created in the prelab.

3. Ensure that the transmitter is set up to use

   Carrier Frequency: 915.1 MHz

   IQ Rate: 200 kHz

   Gain: 0 dB

   Active Antenna: TX1

   Message Length: 200,000 samples gives a good block of data to work with.

   Modulation Index: Start with 1.0.

   Start Frequency, Delta Frequency, Number of Tones: Three tones seems to work well, but keep the highest frequency below 5 kHz.

4. Ensure that the receiver is set up to use

   Carrier Frequency: 915 MHz

   IQ Rate: 1 MHz

Gain: Not critical. 0 dB

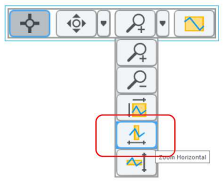
Active Antenna: RX2

Number of Samples: Same value as the transmitted message length.

Question 4: Briefly explain why in the diagram of the receiver, the high and low cutoff frequencies of the bandpass filter set to 95KHz and 105KHz.

Question 5: Briefly explain why in the diagram of the receiver, the cutoff frequency of the lowpass filter set to 5KHz.

Run the transmitter, and then run the receiver. After a few seconds, stop the receiver using the STOP button, then stop the transmitter. Use the horizontal zoom feature on the graph palette to expand the "message" waveform in the transmitter and the "baseband output" waveform in the receiver.



Both waveforms (transmit and receive) should be identical, except for scaling and possible DC offset. Does the waveform look like the sum of 1KHz, 2KHz and 3KHz sines?

5. Power Spectrum

   Set the transmitter to generate a message consisting of three tones starting at 1 kHz with a 1 kHz spacing. Set the modulation index to $\mu = 1$. Run the transmitter and then the receiver. Stop the receiver and then stop the transmitter. Zoom in on the power spectrum so that you can clearly see the components in the vicinity of 100 kHz. Take a screen shot of your power spectrum graph.

   Question 6: The carrier frequency of the transmitter was set to 915.1MHz. Why in the power spectrum of the receiver, we see components in the vicinity of 100KHz?

   Question 7: Compare the power spectrum with the spectrum you predicted in Prelab Question 1. How many dB below the carrier are your sideband components?

   Change the modulation index to $\mu = 0.5$ and capture a new power spectrum. Take another screenshot of your power spectrum graph.

   Question 8: How many dB below the carrier are your sideband components now?

6. The constant D that represents the amplitude of the received carrier can be measured by passing the envelope of (5.5) through a lowpass filter. A measured

value of D is often used in practical receivers to adjust the gain of the receiver's output, providing an "automatic gain control" feature.

In order to measure D, set the transmitter to generate a single-tone message having a frequency of 1KHz and $\mu = 1$ as follows:

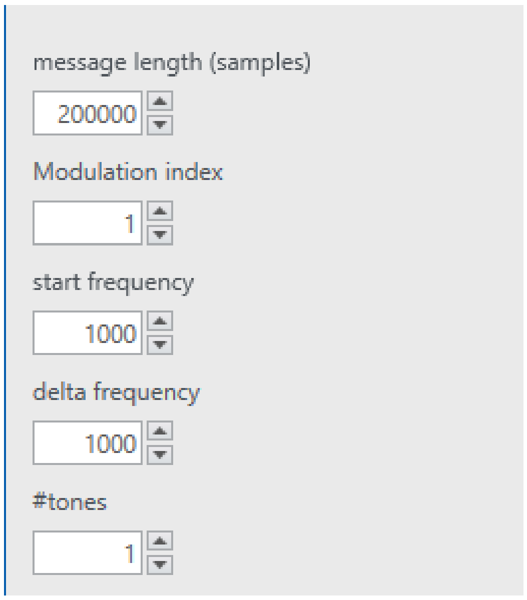

message length (samples) 200000
Modulation index 1
start frequency 1000
delta frequency 1000
#tones 1

Figure 5.12: Generating single tone message with 1KHz frequency

In the diagram of the receiver, change the cutoff frequency of the lowpass filter to the 100Hz. So, the 1KHz message will eliminated and the only remained signal would be D. See (5.5). The $m(t)$ in this equation is the 1KHz signal. The lowpass filter with 100Hz cutoff frequency would eliminate the $m(t)$. Therefore, the recieved signal would be $A(t) = D$.

Run the transmitter and receiver, and measure the value of D by measuring the mean value of the received signal. Increase the gain of the receiver to 5dB and repeat the measurement of D.

Question 9: Is the change in D consistent with a 5 dB change in receiver gain?

5.5 Report

5.5.1 Prelab

Hand in documentation for the functions you created for the transmitter and receiver. Also include documentation for any sub-functions you may have created. To obtain documentation, print out legible screenshots of the front panel and block diagram.

Answer the questions in the Questions section at the end of the prelab instructions.

5.5.2 Lab

Submit the functions you created for the transmitter and receiver. Also submit any sub-functions you may have created. Be sure your files adhere to the naming convention described in the instructions above.

Resubmit documentation for any functions you modified during the lab.

Submit the spectrum graphs and answer all of the questions in Sections 4 and 5 of the Lab Procedure.

5.6 Extra Grade Section: Quadrature Amplitude Modulation

5.6.1 Overview

A variety of communication protocols implement quadrature amplitude modulation, or QAM. Current protocols such as 802.11b wireless Ethernet (Wi-Fi) and Digital Video Broadcast (DVB), for example, both utilize 64-QAM modulation. In addition, emerging wireless technologies such as WiMAX, 802.11n, and HSDPA/HSUPA (a new cellular data standard) will implement QAM as well. Thus, understanding the QAM modulation scheme is important because of its widespread use in current and emerging technologies. QAM modulation involves sending digital information by periodically adjusting the phase and amplitude of a sinusoidal electromagnetic wave. Each combination of phase and amplitude is called a symbol and represents a digital bitstream. First, we will discuss the hardware implementation required to constantly adjust the phase and amplitude of a carrier wave. Second, we will discuss the binary value associated with each symbol.

5.6.2 Hardware Implementation

Quadrature amplitude modulation (QAM) requires changing the phase and amplitude of a carrier sine wave. One of the easiest ways to implement QAM with hardware is to generate and mix two sine waves that are 90 degrees out of phase with one another. Adjusting only the amplitude of either signal can affect the phase and amplitude of the resulting mixed signal. These two carrier waves represent the in-phase (I) and quadrature-phase (Q) components of our signal. Individually each of these signals can be represented as:

$$I = A\cos(\phi), \quad Q = A\sin(\phi),$$

Note that the I and Q components are represented as cosine and sine because the two signals are 90 degrees out of phase with one another. Using the two identities above and the following trigonometric identity

$$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta),$$

rewrite a carrier wave $A\cos(2\pi f_c t + \phi)$ as

$$A\cos(2\pi f_c t + \phi) = I\cos(2\pi f_c t) - Q\sin(2\pi f_c t),$$

As the equation above illustrates, the resulting identity is a periodic signal whose phase can be adjusted by changing the amplitude of I and Q. Thus, it is possible to perform digital modulation on a carrier signal by adjusting the amplitude of the two mixed signals.

Figure bellow shows a block diagram of the hardware required to generate the intermediate frequency (IF) signal. The "Quadrature Modulator" block shows how the I and Q signals are mixed with the local oscillator (LO) signal before being mixed together. The two LOs are exactly 90 degrees out of phase with one another.

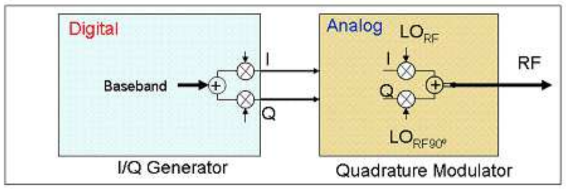


Figure 5.13: Hardware Used to Generate the IF Signal

The next section discusses exactly how the I and Q components are used to represent actual digital data. To do this, the following section details the relationship between the QAM symbol map and the actual real-world signal.

5.6.3 QAM Symbol Map

As stated previously, QAM involves sending digital information by periodically adjusting the phase and amplitude of a sinusoidal electromagnetic wave. 4-QAM uses four combinations of phase and amplitude. Moreover, each combination is assigned a 2-bit digital pattern. For example, suppose you want to generate the bitstream $1, 0, 0, 1, 1, 1$. Because each symbol has a unique 2-bit digital pattern, these bits are grouped in two's so that they can be mapped to the corresponding symbols. In our example, the original bitstream $1, 0, 0, 1, 1, 1$ is grouped into the three symbols $10, 01, 11$. In the following figure, 4-QAM consists of four unique combinations of phase and amplitude. These combinations—called symbols—are shown as the white dots on the constellation plot in bellow figure. The red lines represent the phase and amplitude transitions from one symbol to another. Labeled on the constellation plot is the digital bit pattern that each symbol represents. Thus, a digital bit pattern can be sent over a carrier signal by generating unique combinations of phase and amplitude.

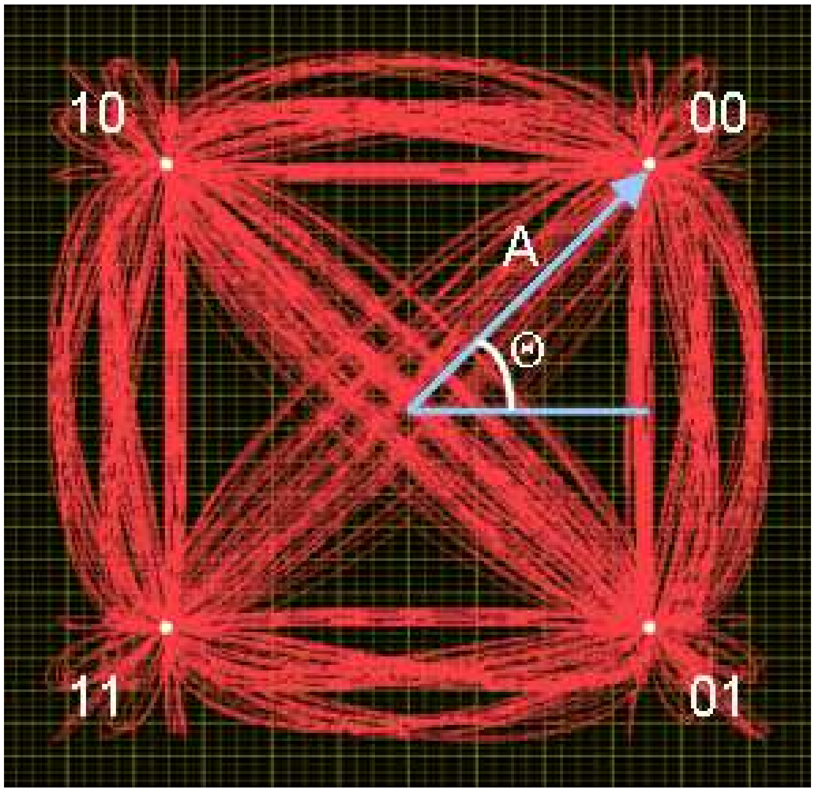


Figure 5.14: 4-QAM Symbol Map

5.6.4 Extra Grade Question

Suppose you want to transmit sixteen numbers from zero to fifteen using 16-QAM. Draw the 16-QAM analoge output waveform for this transmission. You can use "qam_symbol_map.vi" LabVIEW code. Please download "qam_symbol_map.vi" code from i-Learn. Take this into consideration that this code should be run using LabVIEW 2016 and not LabVIEW Communication 2.0 software. Figure bellow illustrates a screen-shot of "qam_symbol_map.vi" program.

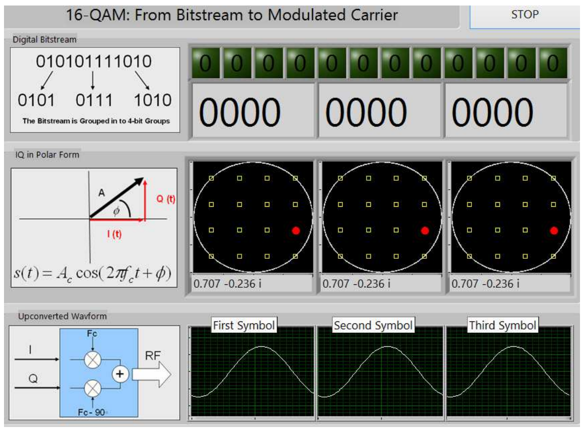


Figure 5.15: qam_symbol_map.vi program

LAB 6

Frequency Modulation

6.1 Objective

This laboratory exercise introduces frequency modulation. This lab exercise is a nice illustration of the utility of the software defined radio approach, since the algorithms for creating and demodulating FM in software are much simpler than those used in the traditional hardware approach.

6.2 Background

6.2.1 Frequency Modulation

Frequency modulation (FM) was introduced by E.A. Armstrong in the 1930's as an alternative to the AM commonly in use at the time for broadcasting. The advantage to frequency modulation is that, for a given transmitted power, the signal-to-noise ratio is much higher at the receiver output than it is for AM. The digital version of FM, frequency-shift keying, has been in use since an even earlier date.

In FM, the frequency of the carrier is modulated to follow the amplitude of the message signal. To be more specific, if $m(t)$ is a message signal with peak value m_p , then the instantaneous frequency $f(t)$ of the carrier is given by

$$f(t) = f_c + k_f m(t), \tag{6.1}$$

where f_c is the carrier frequency and k_f is a proportionality constant called the "frequency sensitivity." The term $k_f m(t)$ is called the "frequency deviation" of the instantaneous frequency from the carrier frequency, and the peak frequency deviation $\Delta f = k_f m_p$ is an important FM system parameter. Given the instantaneous frequency, we can find the total instantaneous angle $\theta(t)$ of the carrier by integrating the instantaneous frequency. That is,

$$\theta(t) = 2\pi \int_0^t f(\alpha)d\alpha = 2\pi f_c t + 2\pi k_f \int_0^t m(\alpha)d\alpha + \theta(0), \tag{6.2}$$

Since the initial angle $\theta(0)$ is of no consequence, we can simplify the equations by taking $\theta(0) = 0$. The transmitted FM signal is then given by

$$\begin{aligned} g(t) &= A_c \cos[\theta(t)] \\ &= A_c \cos[2\pi f_c t + 2\pi k_f \int_0^t m(\alpha)d\alpha] \\ &= A_c \cos[2\pi f_c t + 2\pi \Delta f \int_0^t [m(\alpha)/m_p]d\alpha] \end{aligned} \tag{6.3}$$

Fig. 6.1 shows a 2 kHz carrier frequency modulated by a 200 Hz sinusoidal message.

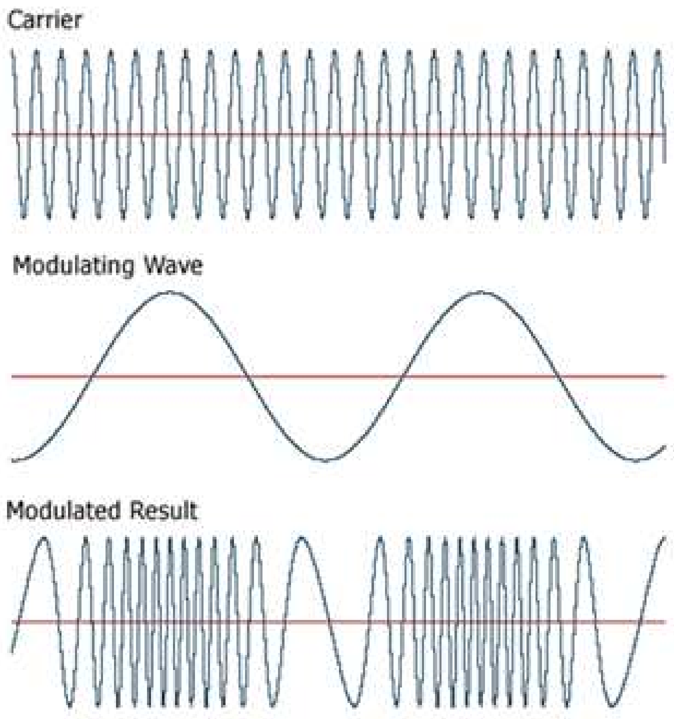


Figure 6.1: Frequency Modulated Signal

To create an FM signal using the USRP, the message signal is normalized to a peak value of one, multiplied by $2\pi\Delta f$ and integrated to give $2\pi\Delta f \int_0^t [m(\alpha)/m_p] d\alpha$. Next, the complex-valued signal $\tilde{g}(t)$ is formed, where

$$\tilde{g}(t) = A_c e^{j2\pi\Delta f \int_0^t [m(\alpha)/m_p] d\alpha} \tag{6.4}$$

The complex-valued signal $\tilde{g}(t)$ is sent to the Write Tx Data function, and the USRP produces the FM signal.

The only tricky step in generating an FM signal is integrating the message. In discrete time we have

$$\int_0^t x(\alpha) d\alpha \cong \sum_{k=0}^{n} x(nT)T, \tag{6.5}$$

where T is the reciprocal of the IQ sample rate. If we write

$$y[n] = \sum_{k=0}^{n} x(nT)T, \tag{6.6}$$

then

$$y[n] - y[n-1] = \sum_{k=0}^{n} x(nT)T - \sum_{k=0}^{n-1} x(nT)T = x(nT)T, \tag{6.7}$$

Equation (6.7) is the difference equation of an IIR filter. This filter can be implemented using the IIR Filter function found in the Analysis → Signal Processing → Filters → IIR Filtering palette. Use a "forward coefficients" array of $[T]$ and a "reverse coefficients" array of $[1 \quad -1]$.

6.2.2 Demodulation

FM demodulation is much easier to carry out using the USRP than it is using conventional hardware. The signal provided by the Fetch Rx Data function is $\tilde{r}(t)$ given by

$$\tilde{r}(t) = A_r e^{j2\pi\Delta f \int_0^t [m(\alpha)/m_p]d\alpha}, \tag{6.8}$$

This is identical to the expression given by Eq. (6.4), except for the magnitude A_r and the presence of noise (not shown in the expression). The angle of $\tilde{r}(t)$ is easily extracted using a Complex to Polar function.[1]

Unwrap the angle before proceeding to the next step. Next, the unwrapped angle is differentiated, giving $2\pi\Delta f[m(t)/m_p] = 2\pi k_f m(t)$. This result should be passed through a lowpass filter, since the differentiation step tends to enhance high-frequency noise.

To implement the differentiator, we recognize that in discrete time,

$$\frac{dx(t)}{dt} \cong \frac{x[n] - x[n-1]}{1} \tag{6.9}$$

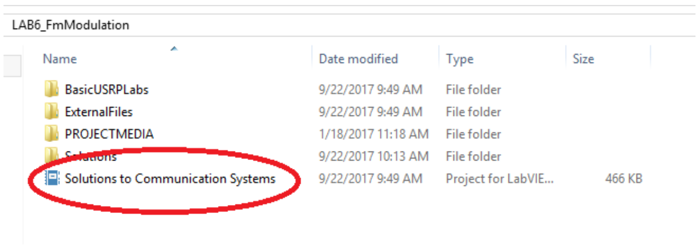
The differentiator can be implemented using an FIR Filter function from the Analysis → Signal Processing → Filters → FIR Filtering palette. For the "FIR coefficients" array use $[1 \quad -1]$.

6.3 Pre-Lab

6.3.1 Transmitter

A template for the transmitter has been provided in the folder: LAB6_FmModulation → Solutions to Communication Systems.(See below figures)

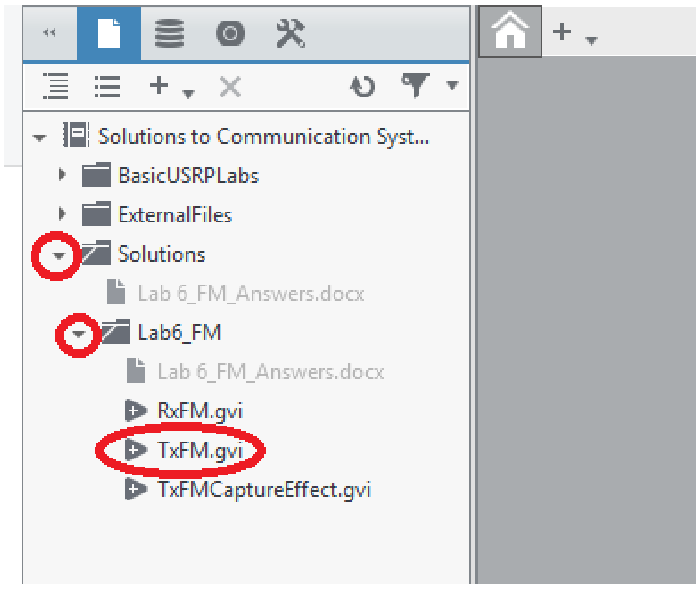
[1]For those familiar with conventional FM demodulation, this step implements the limiter.



LAB6_FmModulation

Name	Date modified	Type	Size
BasicUSRPLabs	9/22/2017 9:49 AM	File folder	
ExternalFiles	9/22/2017 9:49 AM	File folder	
PROJECTMEDIA	1/18/2017 11:18 AM	File folder	
Solutions	9/22/2017 10:13 AM	File folder	
Solutions to Communication Systems	9/22/2017 9:49 AM	Project for LabVIE...	466 KB

Figure 6.2: FM Template Program



Solutions to Communication Syst...
BasicUSRPLabs
ExternalFiles
Solutions
Lab 6_FM_Answers.docx
Lab6_FM
Lab 6_FM_Answers.docx
RxFM.gvi
TxFM.gvi
TxFMCaptureEffect.gvi

Figure 6.3: FM Receiver Template

This template contains the four interface functions along with a "message generator" that is set to produce a message signal consisting of three tones. The three tones are initially set to 1, 2, and 3 kHz, but these frequencies can be changed using front-panel controls. Your task is explain how this program produces the signal $\tilde{g}(t)$ of Eq. (6.4), and passes this signal to the Write Tx Data block. The carrier level A_c is set to a constant value of 0.9. (See figure bellow)

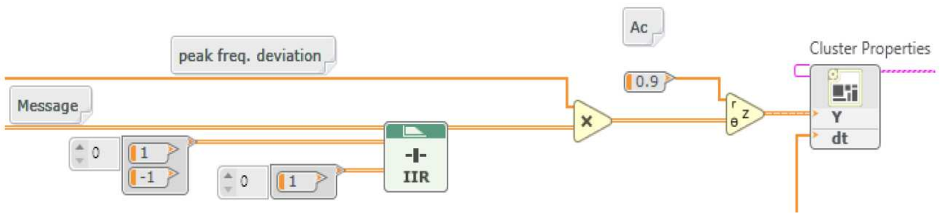
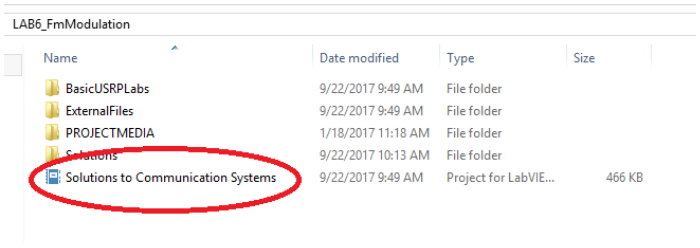


Figure 6.4: FM Modulator

Submit your explanations of the FM transmitter code in the report.

6.3.2 Receiver

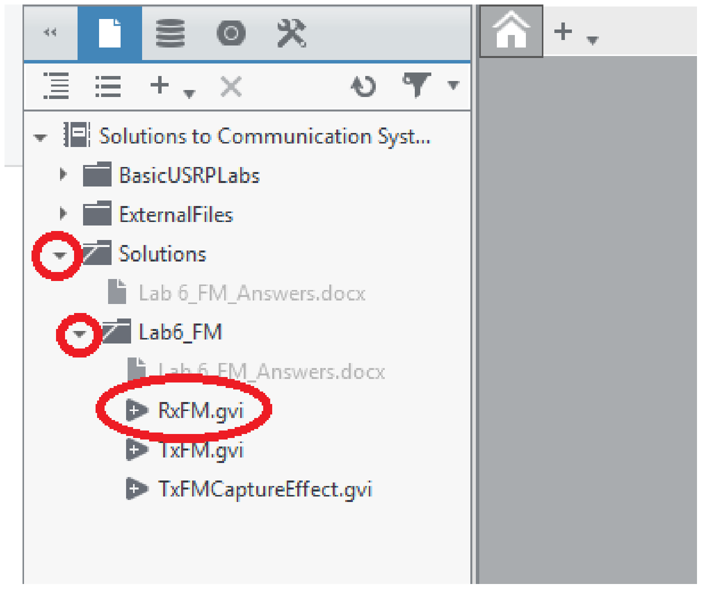
A template for the transmitter has been provided in the folder: LAB6_FmModulation → Solutions to Communication Systems.(See below figures)



LAB6_FmModulation

Name	Date modified	Type	Size
BasicUSRPLabs	9/22/2017 9:49 AM	File folder	
ExternalFiles	9/22/2017 9:49 AM	File folder	
PROJECTMEDIA	1/18/2017 11:18 AM	File folder	
Solutions	9/22/2017 10:13 AM	File folder	
Solutions to Communication Systems	9/22/2017 9:49 AM	Project for LabVIE...	466 KB

Figure 6.5: FM Template Program



Solutions to Communication Syst...
BasicUSRPLabs
ExternalFiles
Solutions
Lab 6_FM_Answers.docx
Lab6_FM
Lab 6_FM_Answers.docx
RxFM.gvi
TxFM.gvi
TxFMCaptureEffect.gvi

Figure 6.6: FM Receiver Template

This template contains the six interface functions along with a waveform graph on which to display your demodulated output signal. Your task is to explain how the program demodulates the complex array returned by Fetch Rx Data and displays the result. Figure bellow shows the implementation of the FM demodulator, including extracting the angle, unwrapping, differentiation, and lowpass filtering.

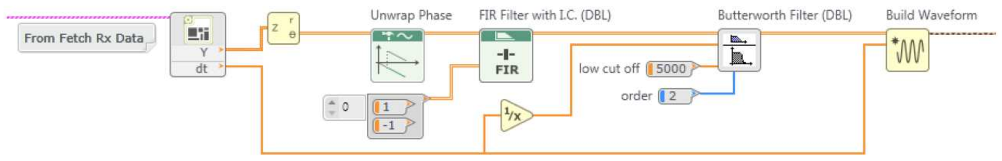


Figure 6.7: FM Demodulator

Submit your explanations of the FM receiver code in the report.

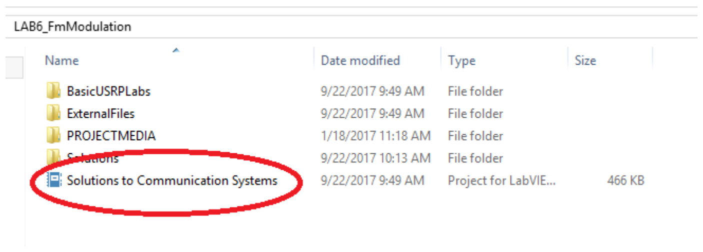
6.3.3 Questions

1. Find the frequency response of the integrator given by Eq. (6.7). Compare with the frequency response of an ideal integrator. Is the discrete-time integrator more like an ideal integrator when the frequency of the input is low or when it is high?

2. Find the frequency response of the differentiator given by Eq. (6.9). Compare with the frequency response of an ideal differentiator. Is the discrete-time differentiator more like an ideal differentiator when the frequency of the input is low or when it is high?

6.4 Lab Procedure

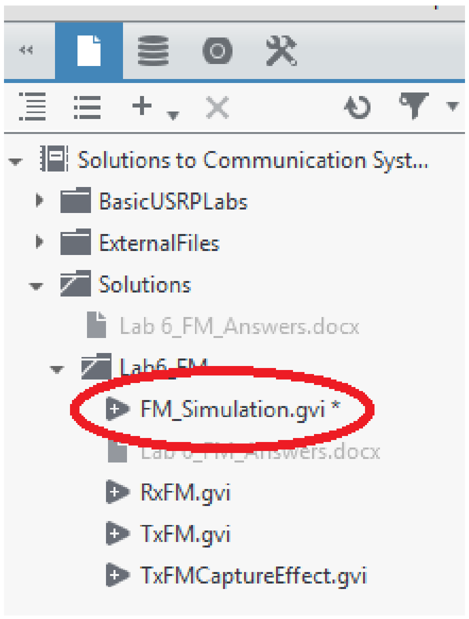
1. A template for simulating the FM signal has been provided in the folder: LAB6_FmModulation → Solutions to Communication Systems.(See below figures)



LAB6_FmModulation

Name	Date modified	Type	Size
BasicUSRPLabs	9/22/2017 9:49 AM	File folder	
ExternalFiles	9/22/2017 9:49 AM	File folder	
PROJECTMEDIA	1/18/2017 11:18 AM	File folder	
Solutions	9/22/2017 10:13 AM	File folder	
Solutions to Communication Systems	9/22/2017 9:49 AM	Project for LabVIE...	466 KB

Figure 6.8: FM Template Program



- Solutions to Communication Syst...
  - BasicUSRPLabs
  - ExternalFiles
  - Solutions
    - Lab 6_FM_Answers.docx
    - Lab6_FM
      - FM_Simulation.gvi *
      - Lab 6_FM_Answers.docx
      - RxFM.gvi
      - TxFM.gvi
      - TxFMCaptureEffect.gvi

Figure 6.9: FM Template Program

2. Set the transmitter parameters as follows:

   Carrier Frequency: 1 MHz

   Massage Frequency: 1 KHz

   frequency deviation: around 30000

   Your FM signal should look like this:

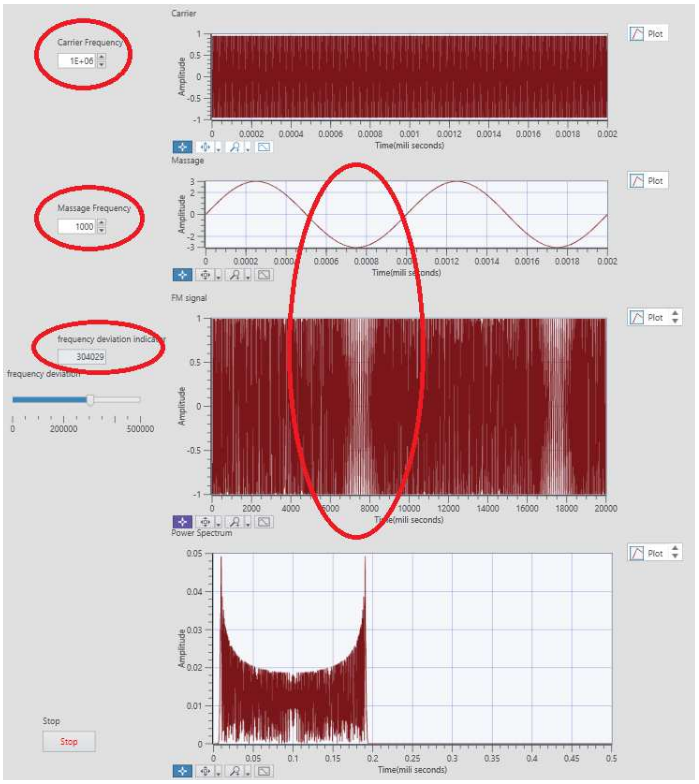


Figure 6.10: FM Signal

Stop the program. Expand the FM signal plot, save a screen shot and evaluate the modulated signal. Your signal should look like this:

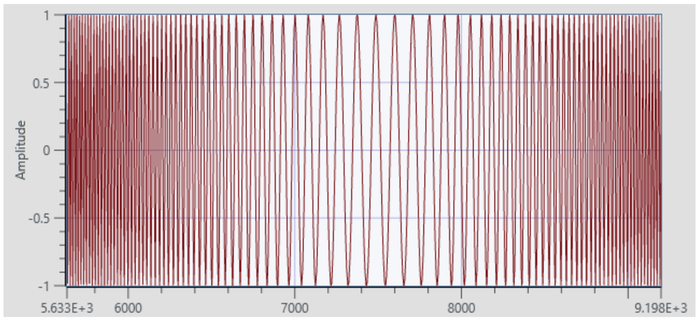


Figure 6.11: FM Signal

Save a screen shot of power spectrum plot. Change the carrier frequency to 2 MHz. Take another screen shot of the power spectrum. Compare two screen shots and explains how changing carrier frequency affects the power spectrum.

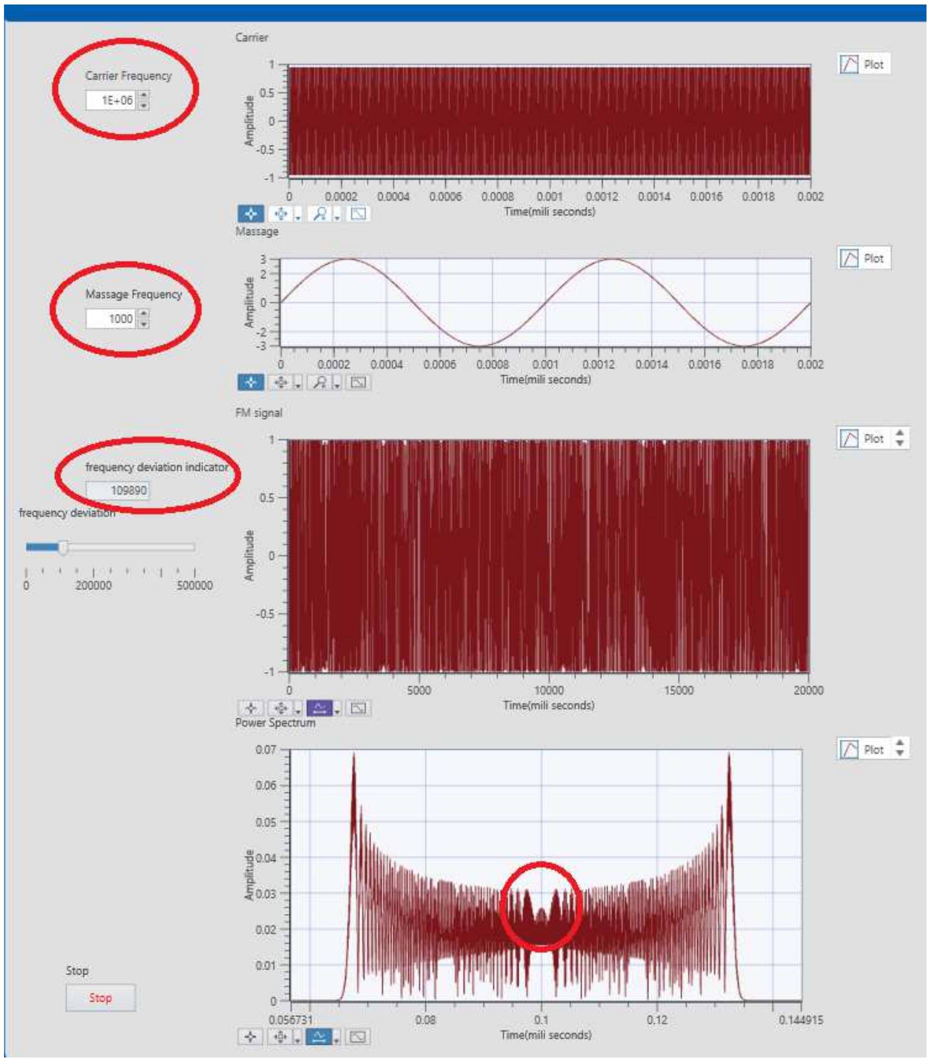
3. One of the useful phenomena associated with FM is the so-called Modulation index. As in other modulation systems, the modulation index indicates by how

much the modulated variable varies around its unmodulated level. It relates to variations in the carrier frequency:

$$b = \frac{\Delta f}{f_m}$$

where Δf is the peak frequency deviation and f_m is the highest frequency component present in the modulating signal. As it can be seen, modulation index has linear relationship with respect to the frequency deviation.

For the case of a carrier modulated by a single sine wave, the resulting frequency spectrum can be calculated using "Bessel functions" of the first kind, as a function of the modulation index. Set the carrier frequency to 1MHz and message frequency to 1KHz. Set the frequency deviation to 110000 approximately. This frequency deviation makes the modulation index to be equal to 2.4 which is the zero point in bessel function. Stop the program and check the power spectrum at the carrier frequency. Use the horizontal zoom feature on the graph palette to expand the power expectrum waveform. See the figure bellow:


Carrier Frequency 1E+06

Massage Frequency 1000

frequency deviation indicator 109890

Figure 6.12: Power spectrum at the zero point of bessel funtion (modulation index equals to 4.2)

As it can bee seen, the carrier power spectrum is disappeared at the modulation index of 2.4. Save a screen shot of your plot.

4. Open both TxFM.gvi and RxFM.gvi described in pre-lab. The diagram and panel of these programs should look like bellow figures:

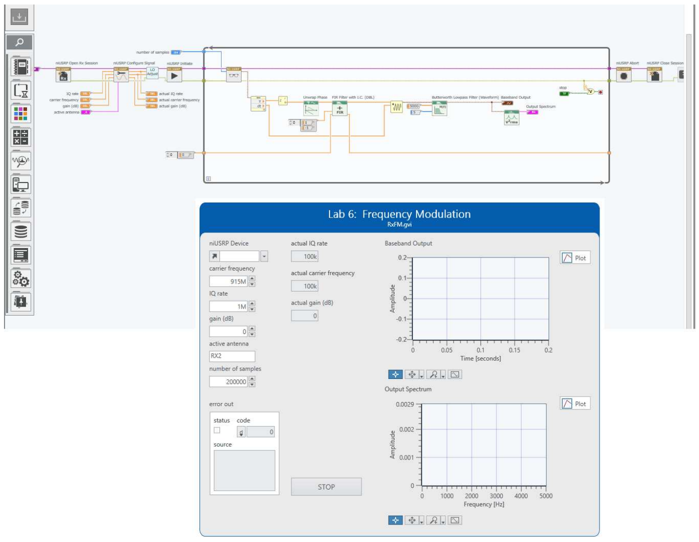


Figure 6.13: Receiver panel and diagram

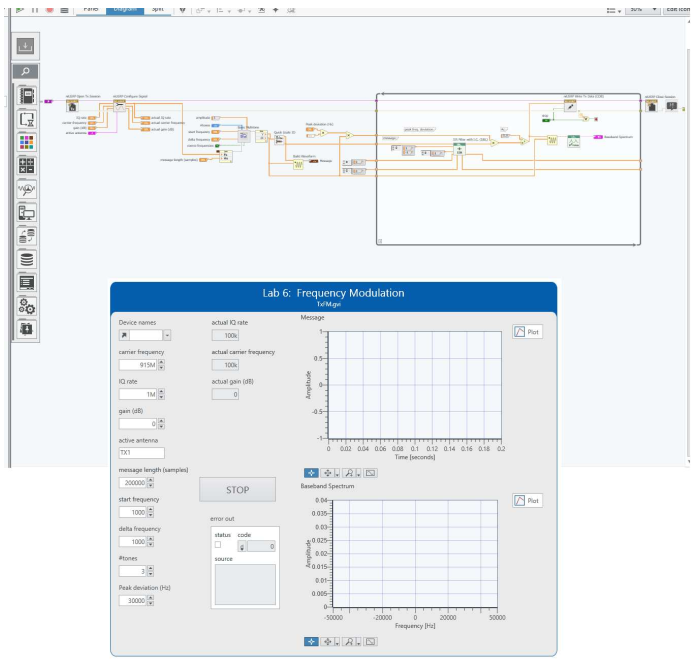


Figure 6.14: Transmitter panel and diagram

5. Connect a loopback cable and attenuator between the TX 1 and RX 2 connectors. Connect the USRP to your computer and plug in the power to the USRP. Run LabVIEW and open the transmitter and receiver that you created in the prelab.

   Note that all graphs in this lab are taken on a linear scale (dB on = False).

6. Ensure that the transmitter is set up to use

   Carrier Frequency: 915 MHz

   Device names: IP address of the USRP

   IQ Rate: Not critical; 1 MHz

   Gain: Not critical. 0 dB

   Active Antenna: TX1

   Message Length: 200,000 samples gives a good block of data to work with.

   Peak Frequency Deviation: 30 kHz seems a good value to start with.

   Start Frequency, Delta Frequency, Number of Tones: Not critical, but keep the highest frequency below 5 kHz. Three tones seems to work well, but you may wish to start with a single tone to verify operation.

7. Ensure that the receiver is set up to use

   Carrier Frequency: 915 MHz

   Device names: IP address of the USRP

   IQ Rate: 1 MHz

   Gain: Not critical. 0 dB

   Active Antenna: RX2

   Number of Samples: Same value as the transmitted message length.

   Run the transmitter, then run the receiver. After a few seconds, stop the receiver using the STOP button, then stop the transmitter (using the STOP button). Use the horizontal zoom feature on the graph palette to expand the "message" waveform in the transmitter and the "Baseband output" waveform in the receiver. Both waveforms should be identical, except for scaling. Save a screen shot of both transmitted and received signals.

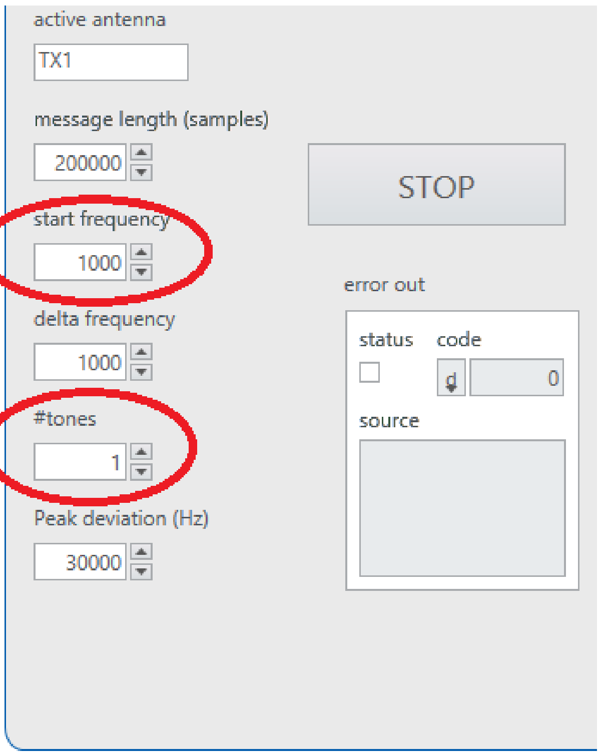
8. **FOR EXTRA GRADE**

   FM Bandwidth: The bandwidth of an FM signal is notoriously difficult to calculate analytically. J.R. Carson, writing in the 1920's, provided a rule of thumb for approximating the bandwidth:

$$B_{FM} \cong 2(\Delta f + B) \tag{6.10}$$

   where B_{FM} is the bandwidth of the FM signal, Δf is the peak frequency deviation of the signal, and B is the message bandwidth.

   To obtain the classic textbook FM spectrum, set the message for a single tone at 1 kHz. (See figure bellow)


active antenna: TX1

message length (samples): 200000

start frequency: 1000

delta frequency: 1000

#tones: 1

Peak deviation (Hz): 30000

error out: status, code 0, source

Figure 6.15: Setting the message for a single tone at 1 kHz

Run the transmitter and obtain power spectra of the transmitted signal for peak frequency deviations of 1 kHz, 5 kHz, and 30 kHz. Use “Peak Deviation (Hz)” numeric input on the panel to change the peak frequency deviation.(See figure Bellow)

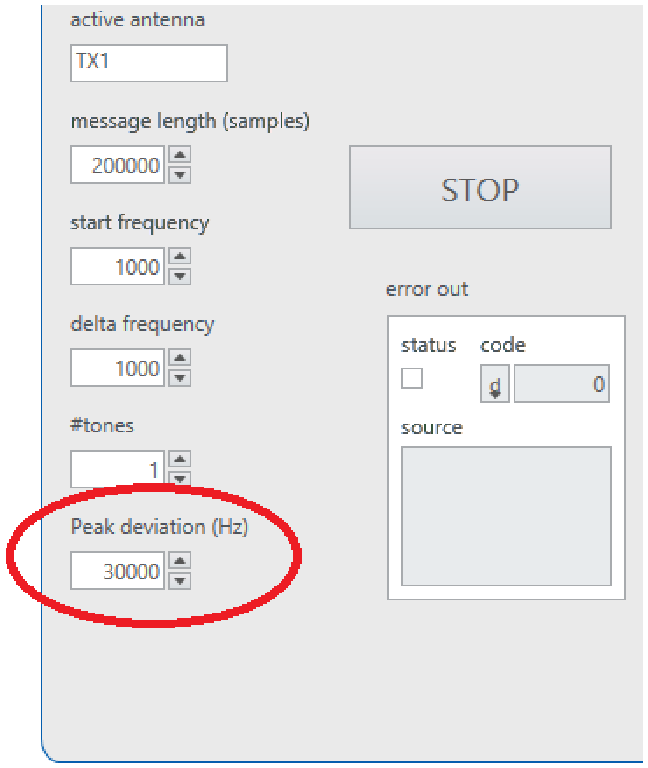

active antenna
TX1
message length (samples)
200000
start frequency
1000
delta frequency
1000
#tones
1
Peak deviation (Hz)
30000
error out
status
code
0
source

Figure 6.16: Setting the peak frequency deviation

Take a screenshot of the power spectrum for each case. Be sure to scale the horizontal axis so that each spectrum is visible. Annotate your spectra to show the Carson's rule bandwidth, Eq. (6.10), for each case.

For a more realistic set of FM spectra, set the message for three tones at 1 kHz, 2 kHz, and 3 kHz. Run the transmitter and obtain power spectra of the transmitted signal for peak frequency deviations of 1 kHz, 5 kHz, and 30 kHz. Take another set of screenshots of the power spectrum for each case. Be sure to scale the horizontal axis so that each spectrum is visible. Annotate your spectra to show the Carson's rule bandwidth for each case.

6.5 Report

6.5.1 Prelab

Explain how the transmitter program produces the signal $\tilde{g}(t)$ of Eq. (6.4), and passes this signal to the Write Tx Data block. Explain how the receiver program demodulates the complex array returned by Fetch Rx Data and displays the result.

To obtain documentation, print out legible screenshots of the front panel and block diagram. Submit your answers to the Questions at the end of the Prelab section.

6.5.2 Lab

Submit the graphs required in Step 2 and 3 and explain the requested details. Submit the graphs required in Step 7 and explain the details of these graphs.

FOR EXTRA GRADE submit the graphs required in Step 8. Be sure to indicate on each graph the bandwidth estimated by Carson's rule. With reference particularly to the spectra for the three-tone message, does an FM signal always have sidebands that are symmetrical with respect to the carrier?

FOR EXTRA GRADE submit the graphs required in Step 9. Explain briefly how this sequence of graphs demonstrates the capture effect.