MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 1. PROGRAMMING SIMPLE FUNCTIONS IN LABVIEW ON A PC

I. Introduction

This experiment is about the construction of a LabVIEW project on a personal computer. Since the project VI's are run on PC, the implementation may not be real time. In fact, the computer operating system is not real-time and hence screen updates, Ethernet port services, etc. take precedence during the execution. While today's PC's have powerful CPU's, this fact can be observed easily for critical real-time processing tasks.

In this experiment, two VI's are constructed under the project both of which are run on the PC. This is achieved by placing the VI's under the "My Computer" item. In the Section III, the steps of constructing the project and the details of the first VI are described. The second VI should be designed individually by considering the first VI as an example as the experiment work.

II. Preliminary Work

1)

a) Find the minimum sampling rate in both rad/s (Ω) and Hz (f) for the following band-limited analog signal whose CTFT is given in Fig. 1 such that it can be reconstructed from its samples.

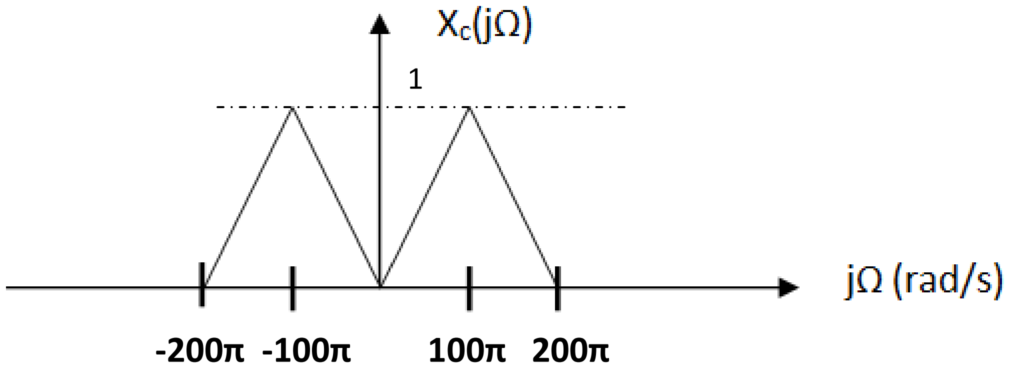


Fig. 1. CTFT of the band-limited analog signal in question 1

b) The analog signal given in Fig. 1 is sampled with a sampling period **T=1/150 s**. Plot the **discrete time Fourier transform (DTFT)** of the sampled sequence by indicating necessary information on the x- and y-axes. Determine whether the aliasing occurs or not. If aliasing occurs, specify the frequency region where aliasing occurs.

c) The analog signal given in Fig. 1 is sampled with a sampling period **T=1/500 s**. What is the highest frequency in the sampled signal? Let $\Omega_a = 100\pi$ **rad/s** be an **analog frequency** value. Which is the **corresponding discrete frequency** ω_a in DTFT?

2) DFT is implemented using **Fast Fourier Transform (FFT)** algorithm in practice and MATLAB environment. In this part, you will explore **"fft"** and **"ifft"** functions in MATLAB for taking DFT and inverse DFT, respectively. You can write **"help fft"** and **"help ifft"** in the **Command Window of MATLAB** to find out these functions.

- fft(x,N) is the N-point fft of x, padded with zeros if x has less than N points and truncated if it has more.
- ifft(x,N) is the N-point inverse fft of x.
- If N is not given, such as fft(x), the fft and ifft are implemented by taking DFT and inverse DFT size as the length of input sequence.
- You are first required to define an input sequence in MATLAB, which consists of the ***first 6 digits*** of your student ID number. For example, if your student ID is 1673862, the MATLAB code is given as follows,

```
x=[ 1 6 7 3 8 6 ];
```

- Take N=6 point fft of x and call it "y". Plot the magnitude and phase of y using ***"stem"*** command in MATLAB. Attach these plots to your preliminary work.
- Is there some kind of symmetry in the plots? If there is a symmetry, explain the reason considering input sequence x.
- Then, take N=9 point fft of x and call it "z". Compare z with y. Are they the same?
- Now take N=9 point ifft of z and compare the result with x.
- Take N=6 point ifft of z and compare the result with x. Why is it different than x?
- Take N=4 point fft of x and call it "v". Then, take ifft of v and compare the result with x.

3) In this part, you are required to implement convolution of two discrete time sequences in LabVIEW environment. The front panel is seen in Fig. 2.

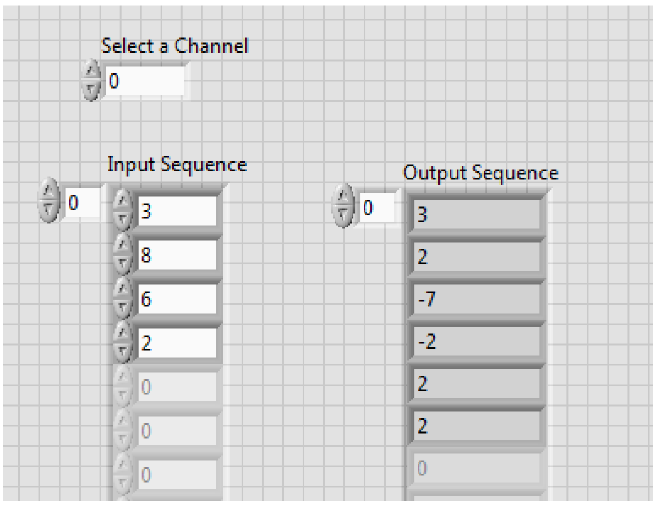


Fig. 2. Front Panel of the Programming Task of first question

The input sequence is entered manually as shown in Fig. 2 through an Array Numeric control. The convolution of this input sequence with two different channels (impulse responses) will be implemented. If **"Select a channel"** input is entered as **0**, then the impulse response is given by,

$h_0[n]=\delta[n]-2\delta[n-1]+\delta[n-2]$.

If **"Select a channel"** input is entered as **1**, then the impulse response is given by,

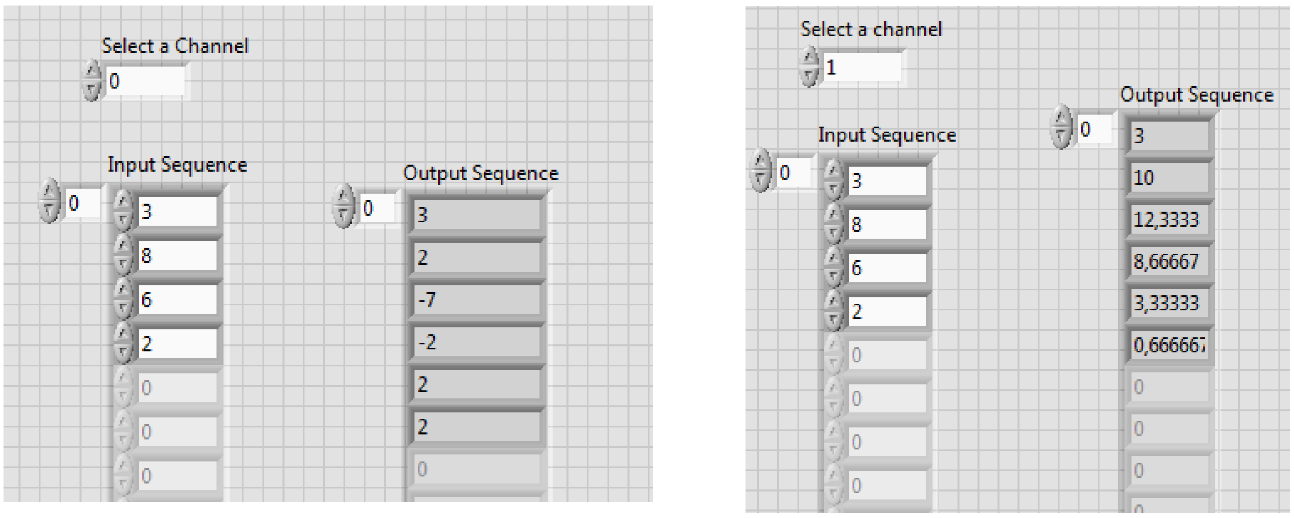
$h_1[n]=\delta[n]+2/3*\delta[n-1]+1/3*\delta[n-2]$.

The default channel can be any of these two channels if another number is entered as **"Select a channel"**.
Assume that the first element of the input sequence correspond to n=0, that is, the example input sequence in Fig. 2 is

$x[n]= 3*\delta[n]+8*\delta[n-1]+6*\delta[n-2]+2*\delta[n-3]$.

- In order to implement convolution, you can use shift registers or feedback nodes. In both cases, please **do not forget** to take **initial values of registers as 0**.
- Remember that the length of the convolved sequence of two sequences is (N+P-1), if the length of input sequence and impulse response is N and P, respectively. Since the length of the channel impulse responses in this task is P=3, we expect the size of the indicator array which shows the output sequence in Fig. 2 is (N+2). As a hint, you can append two zeros at the end of the input sequence. But, **you are required to do it inside the code (i.e., block diagram)**, not by entering two additional zeros. That means if the input sequence is [3 8 6 2] as in Fig. 2, we only enter 3,8,6,2 values without additional zeros.

- You are required to ***attach*** the ***screenshots of the front panel*** of your VI with **two different channel selections** to your preliminary work as shown in Fig. 3. You should also attach the **screenshot of your block diagram**.
- The **input should be given** as the **last 4 digits** of your **student ID** number. For example, if your ID is 1673862, the input should be [3 8 6 2] as shown in Fig. 3.


Select a Channel
0

Input Sequence
0
3
8
6
2
0
0
0

Output Sequence
0
3
2
-7
-2
2
2
0

Select a channel
1

Input Sequence
0
3
8
6
2
0
0
0
0

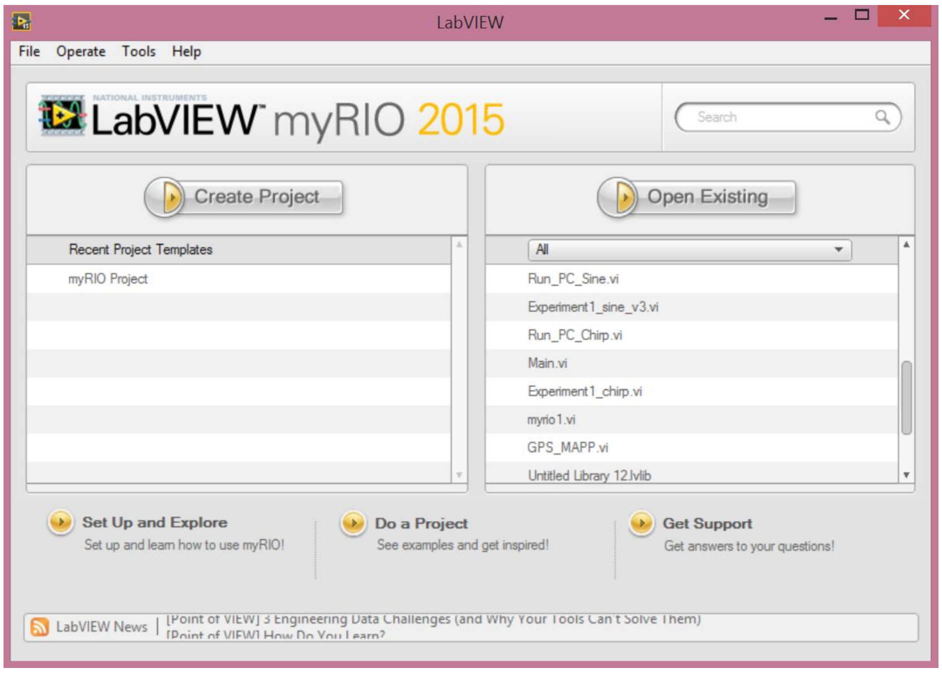
Output Sequence
0
3
10
12,3333
8,66667
3,33333
0,66666
0
0
0

Fig. 3. Screenshots of Front Panel for two channel selections

III. Experimental Work

1. Project Generation and the Run_Sine.vi

Run LabVIEW and select myRIO Project from the templates as shown in the Fig. 4.


LabVIEW

File Operate Tools Help

LabVIEW myRIO 2015

Create Project | Open Existing

Recent Project Templates
- myRIO Project

All
- Run_PC_Sine.vi
- Experiment1_sine_v3.vi
- Run_PC_Chirp.vi
- Main.vi
- Experiment1_chirp.vi
- myrio1.vi
- GPS_MAPP.vi
- Untitled Library 12.lvlib

Set Up and Explore — Set up and learn how to use myRIO!

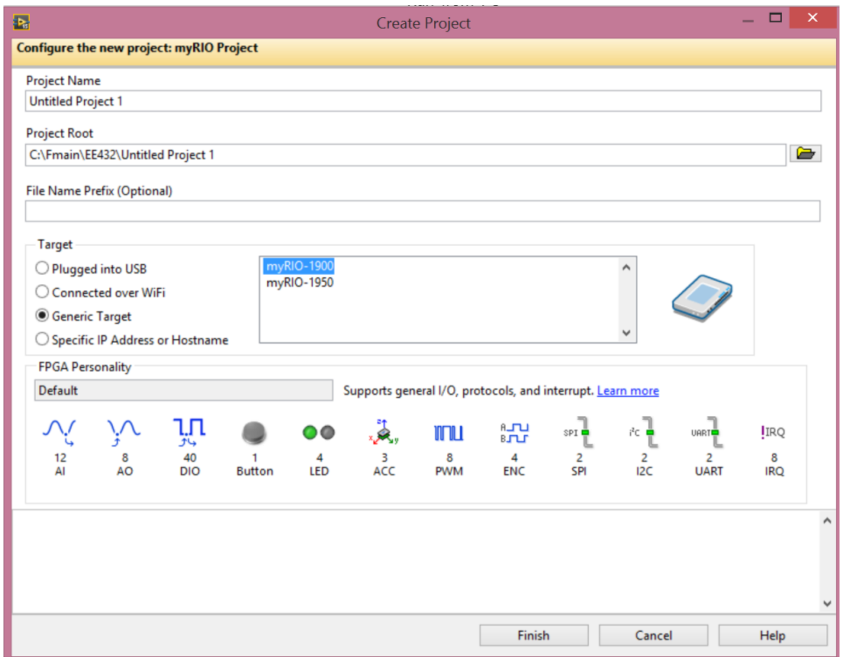
Do a Project — See examples and get inspired!

Get Support — Get answers to your questions!

LabVIEW News | [Point of VIEW] 3 Engineering Data Challenges (and Why Your Tools Can't Solve Them) [Point of VIEW] How Do You Learn?

Fig. 4. Generation of a LabVIEW project

Fill in the project name and directory. Select a Generic Target since the project does not need myRIO as shown in Fig. 5. Click Finish and a new screen comes up as shown in Fig. 6.


Create Project

Configure the new project: myRIO Project

Project Name: Untitled Project 1

Project Root: C:\Fmain\EE432\Untitled Project 1

File Name Prefix (Optional):

Target
- ○ Plugged into USB
- ○ Connected over WiFi
- ● Generic Target
- ○ Specific IP Address or Hostname

myRIO-1900
myRIO-1950

FPGA Personality: Default — Supports general I/O, protocols, and interrupt. Learn more

12 AI	8 AO	40 DIO	1 Button	4 LED	3 ACC	8 PWM	4 ENC	2 SPI	2 I2C	2 UART	8 IRQ

Finish | Cancel | Help

Fig. 5. Project details

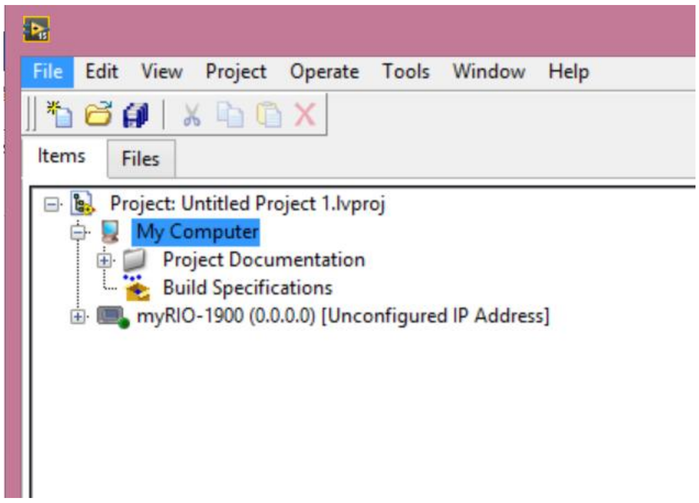


Fig. 6. Project screen

As shown in Fig. 6, **myRIO-1900** is added to the project automatically. Since this project does not need myRIO and the program is **run on PC**, remove "myRIO" from the project by right clicking on it. Next, you can create your own VI that is run on PC by **right clicking on My Computer and selecting VI** as shown in Fig. 7.

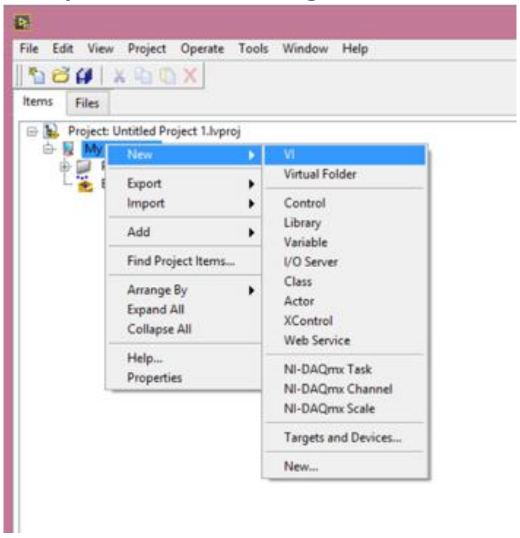


Fig. 7. Generating a VI under My Computer

At this point, two items pop-up. One is the **Front Panel** of the VI, the other is **Block Diagram**. Front Panel is used to add certain gadgets, buttons for user interface and control. Most of the programming is implemented on the Block Diagram. You can save your VI and give it a name such as **Run_Sine.vi**. In the Block Diagram, right click and select while loop from the **Functions menu under Structures** as shown in Fig. 8.

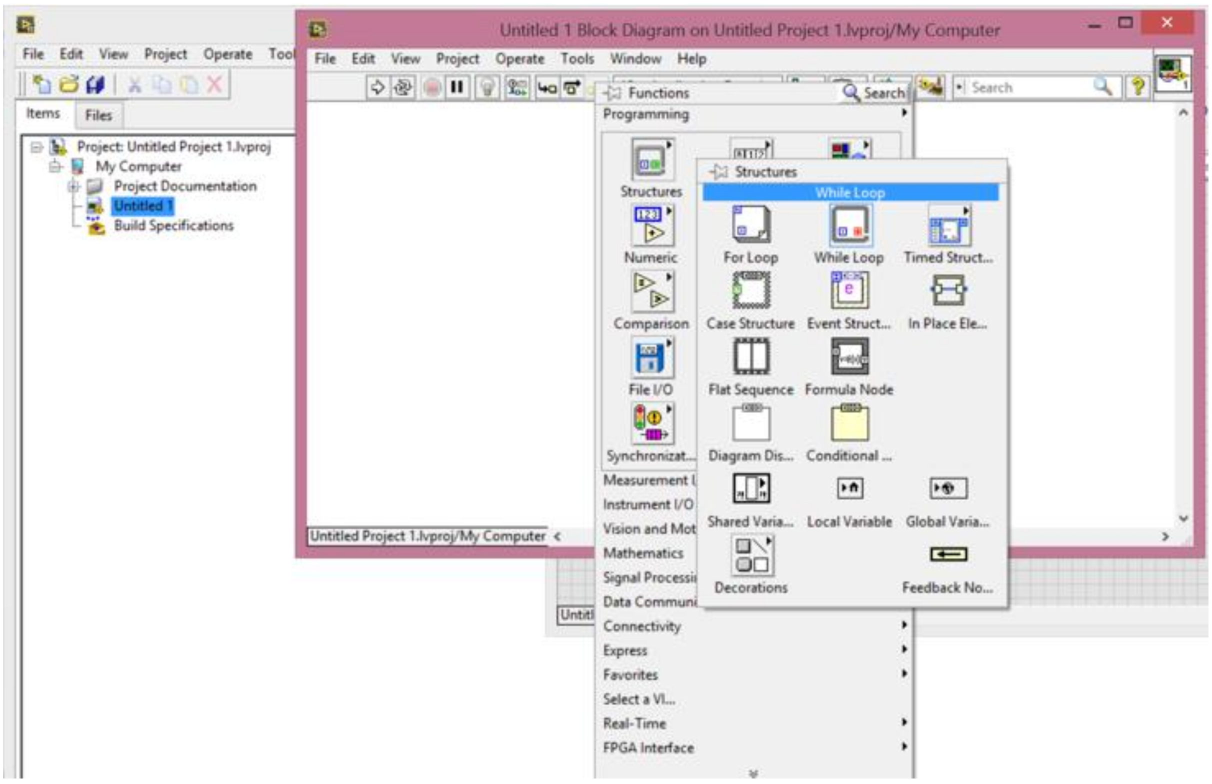


Fig. 8. Functions menu and the while loop function

Create a stop button by right clicking the red button and selecting **Create → Control** as shown in Fig. 9.

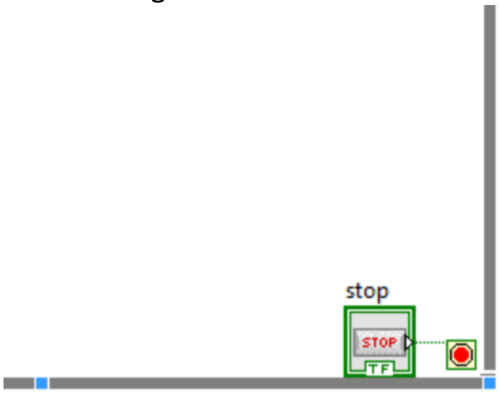


Fig. 9. While Loop and the control button

Now the while loop is ready but functionality should be added inside. The Run_Sine.vi is designed to generate a **sinusoid corrupted by AWGN and plot its DFT magnitude and phase**. There is a special **express VI** for this purpose in LabVIEW, **"Simulate Signal"**, which you can select as shown in Fig. 10 (**Functions → Signal Processing → Waveform Generation → Simulate Signal**).

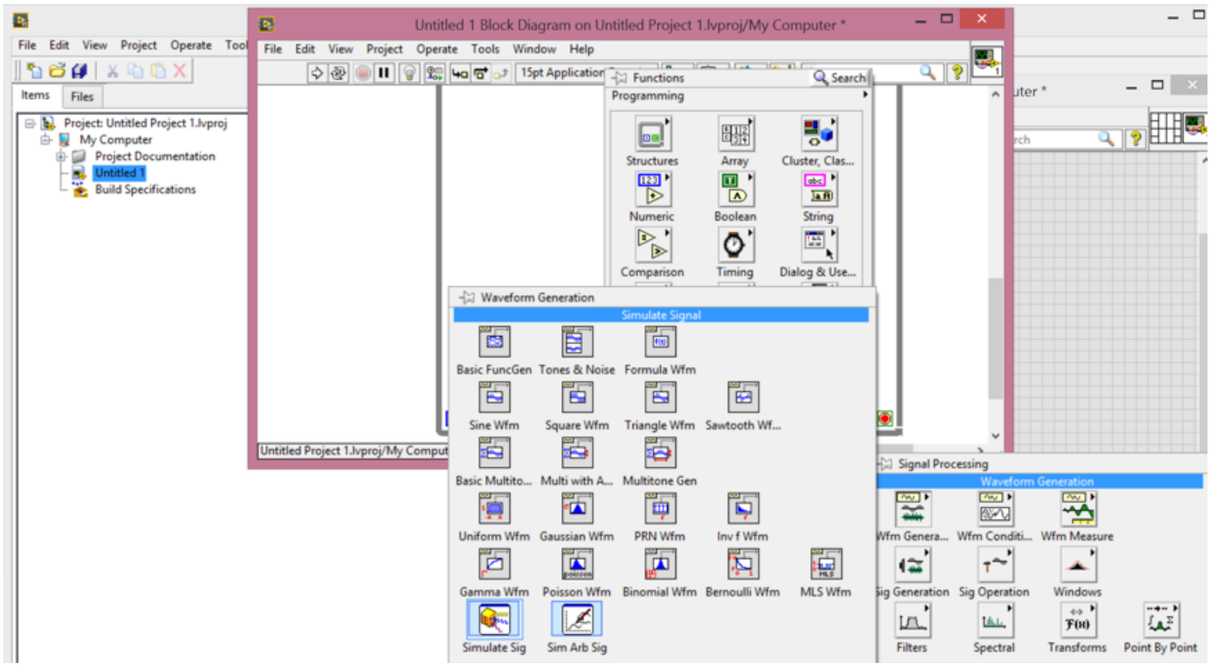
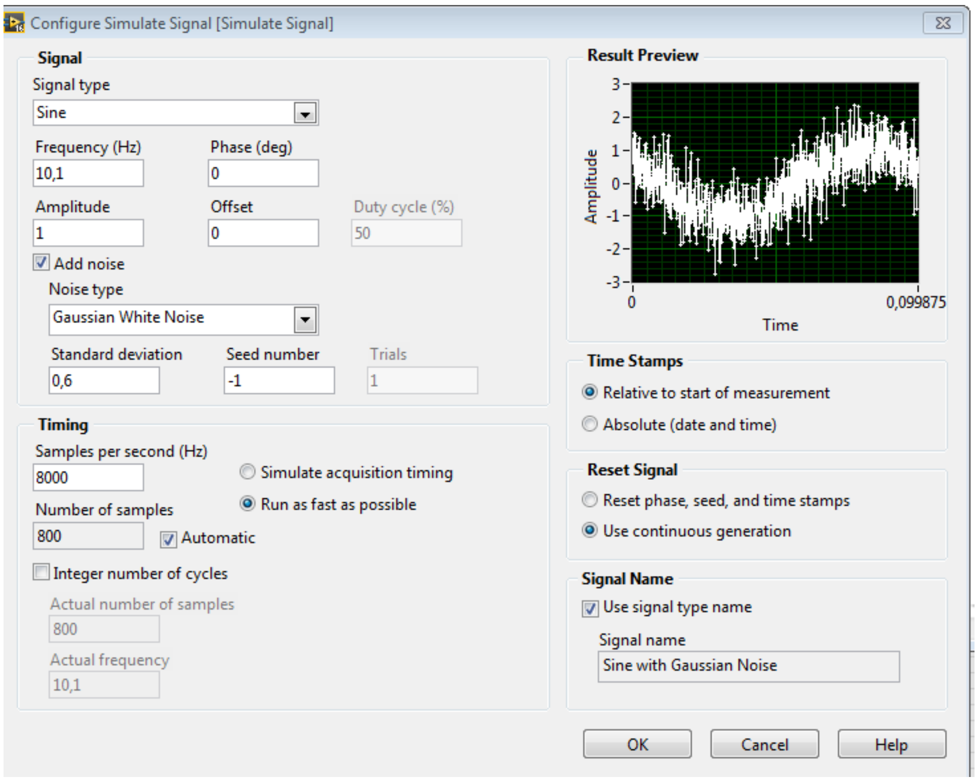


Fig. 10. Selecting the Simulate Signal function

When you place this function on the Block Diagram, you can **configure** its inputs to set frequency, phase offset, noise variance, etc. We will enter **frequency, amplitude, phase offset and noise standard deviation** as inputs. In the Configuration window, click on **"Add noise"** and select **Gaussian White Noise** as noise type as shown in Fig. 11. Set "**Samples per second"** as **8000**, number of samples would be automatically adjusted as 800 corresponding to 100 ms duration.



Configure Simulate Signal [Simulate Signal]

Signal

Signal type: Sine

Frequency (Hz): 10,1
Phase (deg): 0

Amplitude: 1
Offset: 0
Duty cycle (%): 50

[✓] Add noise

Noise type: Gaussian White Noise

Standard deviation: 0,6
Seed number: -1
Trials: 1

Timing

Samples per second (Hz): 8000

( ) Simulate acquisition timing
(●) Run as fast as possible

Number of samples: 800
[✓] Automatic

[ ] Integer number of cycles

Actual number of samples: 800

Actual frequency: 10,1

Result Preview

Time Stamps

(●) Relative to start of measurement
( ) Absolute (date and time)

Reset Signal

( ) Reset phase, seed, and time stamps
(●) Use continuous generation

Signal Name

[✓] Use signal type name

Signal name: Sine with Gaussian Noise

OK | Cancel | Help

Fig. 11. Configuration of Simulate Signal

Create numeric controls for frequency, amplitude, phase and standard deviation as shown in Fig. 12.

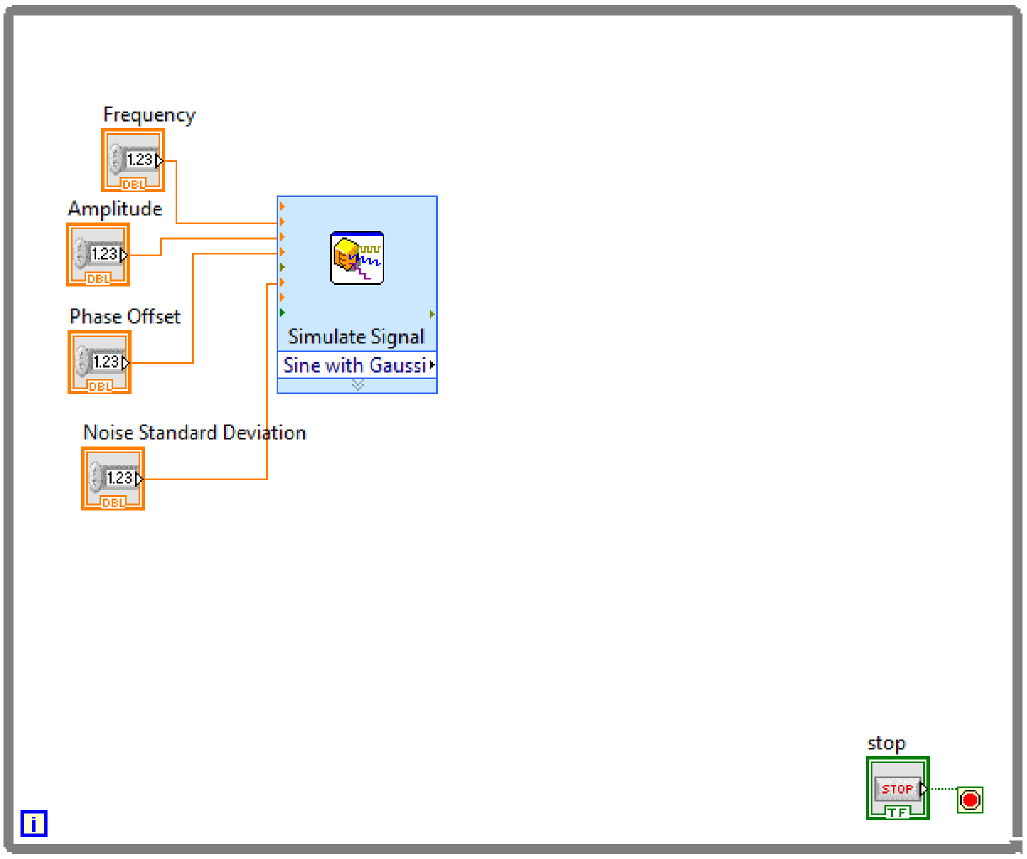


Fig. 12. Numeric Controls for Simulate Signal

Now, create a graph indicator by **right clicking on "Sine with Gaussian"** terminal on the express VI Simulate Signal and select **Create → Graph Indicator**. Furthermore, place an FFT function by selecting **Functions → Signal Processing → Transforms → FFT**. Connect the **"X" input of FFT function** to the **Sine with Gaussian** output of **Simulate Signal**. When you make connection, **Convert from Dynamic Data** function is automatically wired between two terminals. Fig. 13 shows the view of block diagram at this point.

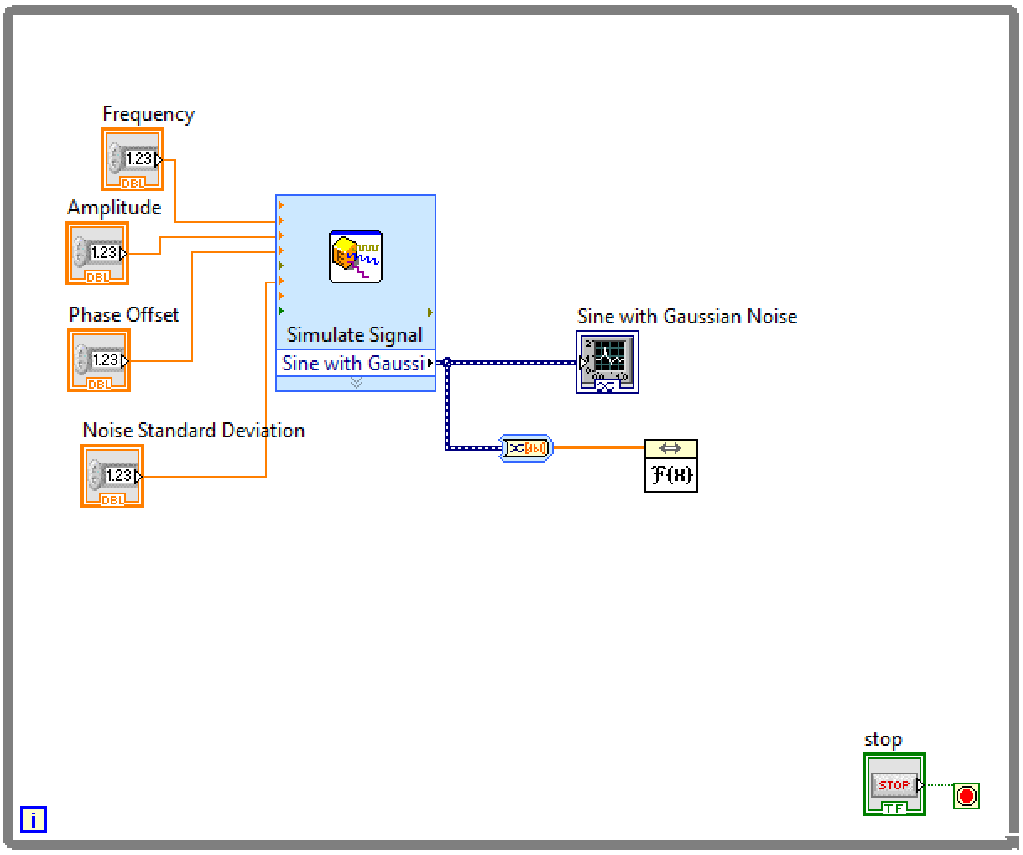


Fig. 13. Block Diagram after placing FFT function

Fig. 14 shows the complete Block Diagram for the Run_Sine.vi. Our aim is to plot magnitude in log domain and unwrapped phase (in order to eliminate discontinuities) of FFT of the simulated signal.

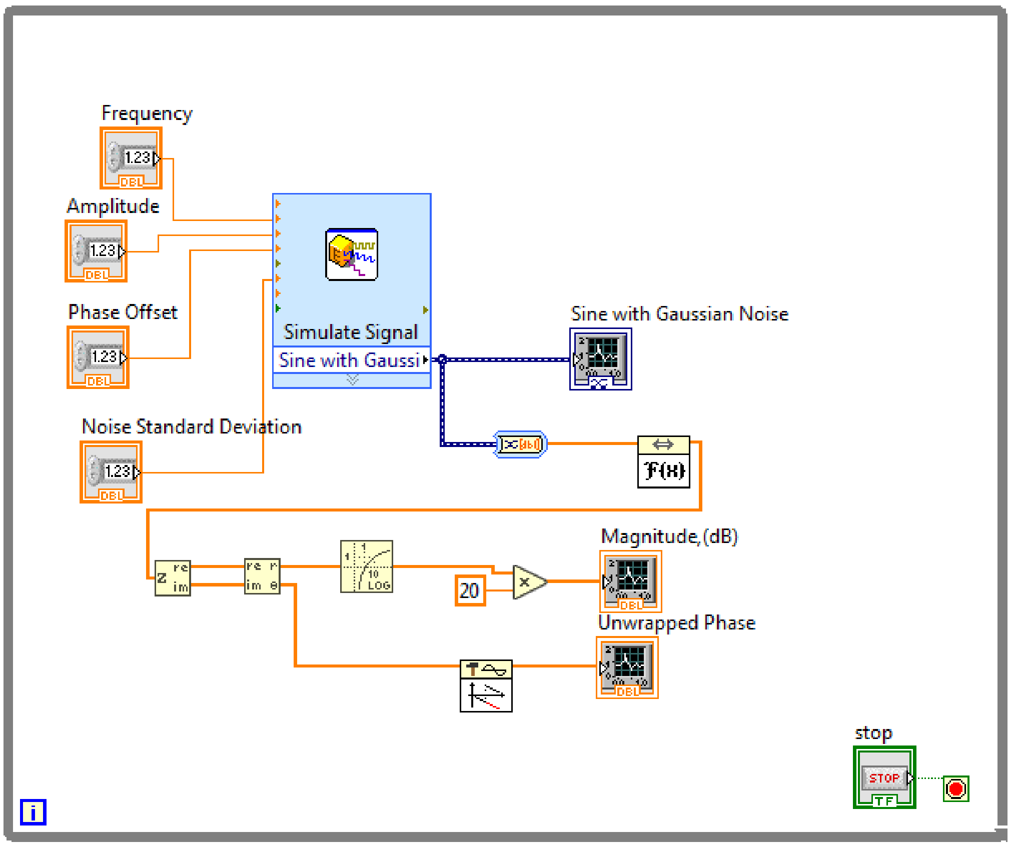


Fig. 14. Run_Sine.vi Block Diagram

As a first step, add **“Complex To Re/Im”** function in order to break a complex number to its rectangular components. You can find this function using **Search** button in **Functions palette**. Then, add **“Re/Im To Polar”** function to convert rectangular components into polar components. Place **“Logarithm Base 10”** function to take the logarithm of the magnitude and plot **20log10(magnitude of FFT coefficients)** using a **Waveform Graph**. Place **“Unwrap Phase.vi”** to eliminate 2π discontinuities and plot unwrapped phase using a **Waveform Graph.**

The Front Panel for this VI is shown in Fig. 15.

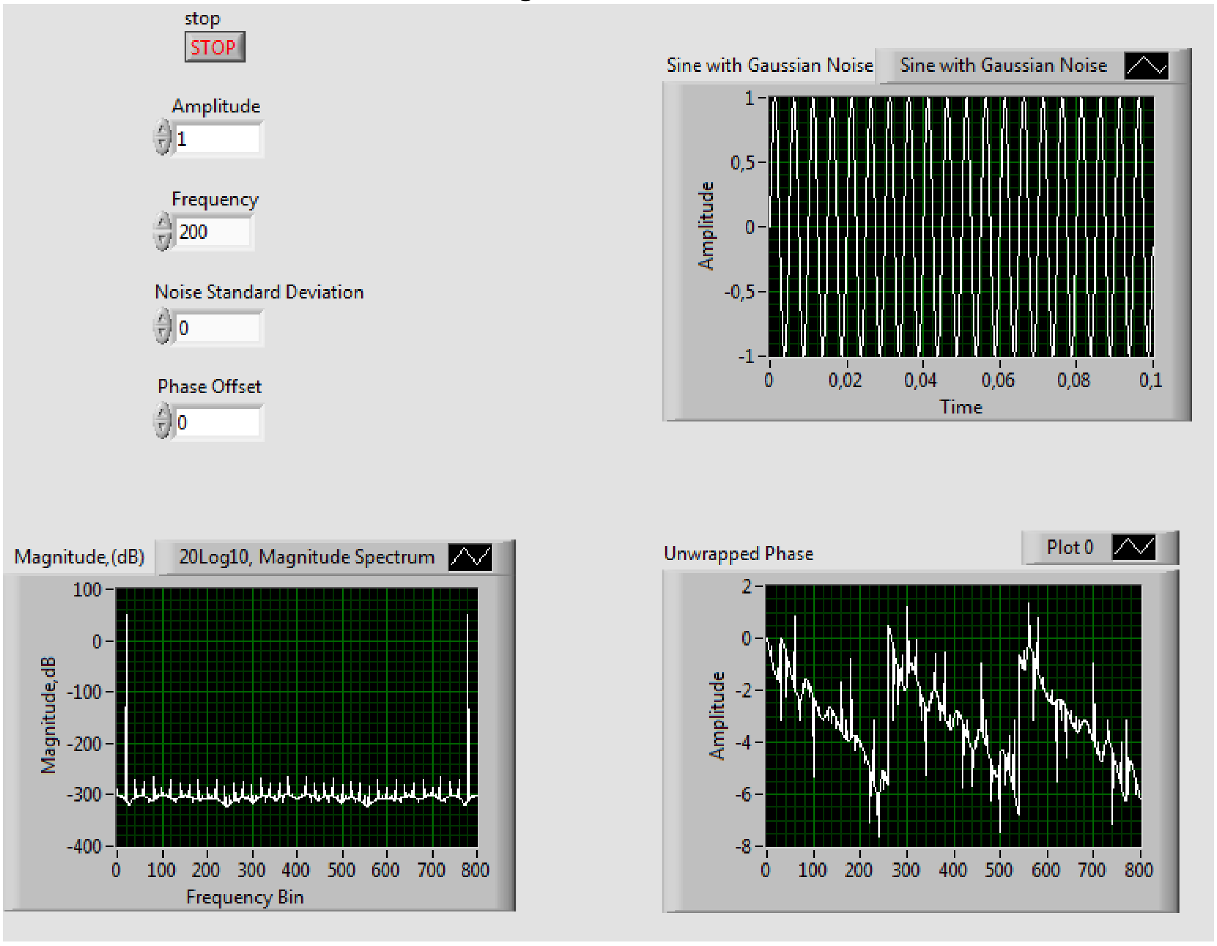


Fig. 15. Front Panel of the Run_Sine.vi

Now this VI can be run by clicking on it. Note that all the processing is done on PC and this may not be a real-time implementation in strict sense.

Programming Task 1:

a) Take the **Noise Standard Deviation** as **0** and the remaining parameters as shown in Fig. 15. Explain the Magnitude and Phase plot axis. The input sinusoid is **200Hz** and the sampling rate for the Simulate Signal block is **8kHz**. Identify the frequency bin corresponding to the input frequency in magnitude spectrum. You can **right click on any waveform** and select **Export → Export Data To Excel** to access the x- and y-axis values.

b) Now, change the **number of samples from 800 to 799** under **Configuration window**. Why does the sinusoid spectrum have a different shape while in theory it is expected to be a single line as in Fig. 15? Note that there is no noise component in the sine in this step.

c) Set the **number of samples as 800** again as in the step a). Increase the Noise Standard deviation gradually (you can select 10 fold increases). Explain if you can identify the sinusoid better in time or frequency? Note that optimum detection technique of a single sinusoid corrupted by noise is to look at the magnitude spectrum. Determine the standard deviation value by which you can still identify the sinusoid in frequency. Calculate the SNR value for this case,

$$SNR = 10Log_{10}\frac{\sigma_x^2}{\sigma_n^2} \quad \text{(dB)}$$

where σ_x^2 and σ_n^2 stand for the signal and noise power respectively.

d) What is the information given by the phase plot?

2. Design of The Second vi, Run_Chirp.vi

In this part, you are required to **design Run_Chirp.vi** whose function is to generate a **chirp** waveform with a given parameter set and display its time and frequency characteristics. The **input parameters** are **sampling frequency, chirp amplitude, number of samples, start frequency, and end frequency**. There are three displays for **time waveform, magnitude spectrum and phase spectrum** respectively. The Front Panel for this VI is shown in Fig. 16. Note that in this case you cannot use Simulate Signal function since it has no chirp waveform. In order to generate chirp, you need to use the **Chirp Pattern.vi** function in the Block Diagram functions menu.

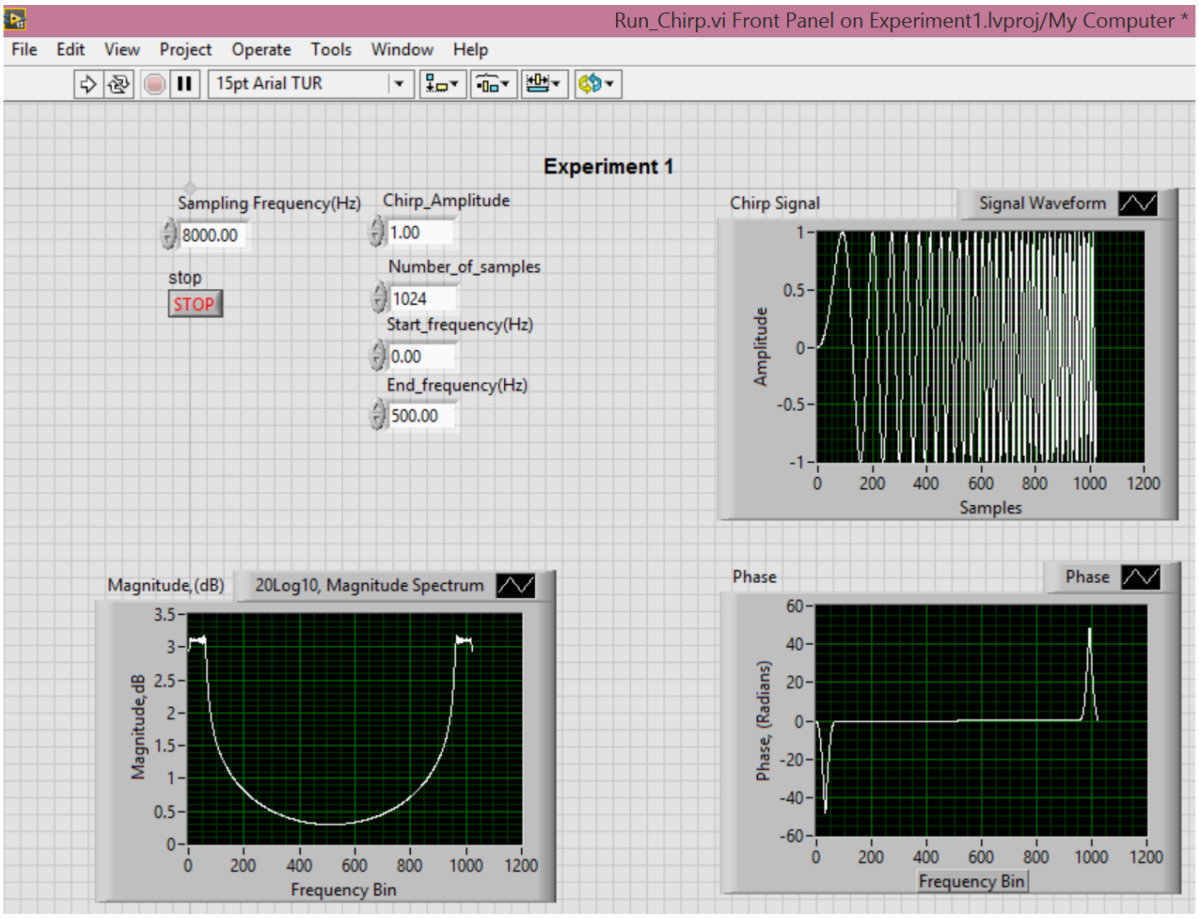


Fig. 16. Run_Chirp.vi Front Panel

Fig. 17 shows the **project view** of Experiment 1. Your implementation can be different but you can use it as an example.

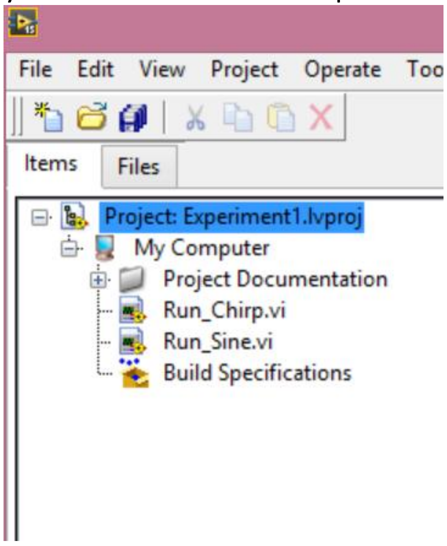


Fig. 17. Project for the Experiment 1

Programming Task 2:

Repeat the items given in programming Task 1 and explain the differences between two cases when the sampling rates are the same. **In addition** realize the following items:

a) Decrease the sampling rate gradually to a level at which you would not observe the chirp waveform characteristics given in Fig. 16. Note this value in your report. Can you identify a general rule for the minimum sampling rate?

b) Comment on the Magnitude and Phase characteristics of chirp signal. Chirp is usually used in radar systems. Explain this in comparison to a sinusoid. In other words, why the radar systems use chirp instead of a sinusoid?