

# **ELEC50002 Communications - Laboratory**

---

**Laboratory Handout 1 (Lab Session 1 + Lab Session 2)**

## Lab 1: Introduction to LabView

The purpose of this lab is to introduce you to the LabView NXG's environment. In this lab, you will learn how to implement mathematical functions and create basic communications modules.

### LabView Introductory Video

Go to the Microsoft Teams > ELEC50002 Communications Laboratory and watch the video titled "Introduction to LabView Communications.mp4".

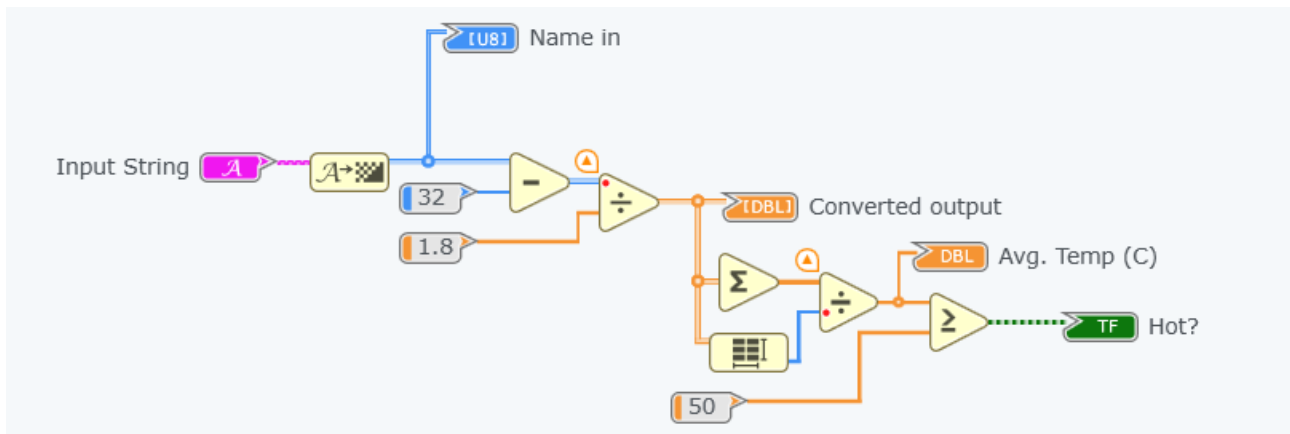
### Exercise 1: Data Types in LabView

In this exercise you will examine the data types in LabView and do some basic conversions between them. The purpose of the code will be to convert each letter of a string into its ASCII binary representation, which will be shown in their equivalent unsigned integer form. Then by treating each integer as a Fahrenheit temperature, you will convert it into Celsius. Finally, an average of the Celsius values will be taken and compared to a threshold value to indicate if the string you inputted is considered "Hot".

#### Instructions:

1. Open NI LabView NXG 3.1.
2. Choose "**Design for Wireless Communication**".
3. From the projects tab, create a new "**Blank Project**", and name it **Lab1**.
4. On the "**Files**" pane, add a new VI, and name it **DataTypes.gvi**.
5. Now you will add the module through which you can input strings. This can be done from the "**function palette**" under the "**Diagram tab**". Using the search box, search for the module **String to byte array** and create a control from the input by right clicking on the terminal "string" and selecting "create control". This will allow you to input a string from the Panel tab. You can also learn more about the modules by clicking on the module, and on the right panel click on "**Context Help**".
6. Create an indicator at the output of the **String to byte array** module by right clicking on the "unsigned byte array" terminal and choosing "Create indicator". This will allow you to view the byte array on the Panel tab in the form of unsigned integers representing the ASCII values of each character of the string you input.
7. Assuming that each ASCII value represents a temperature in Fahrenheit, convert the array of integers into Celsius values. Arithmetic operations can be applied to whole arrays by connecting the array and a scalar constant to an operator node. **Hint:** You can right click on the terminals of a module to create constants, and if it gives you an array constant you can switch to a scalar constant by right clicking on the constant itself.
8. Create an indicator for the converted array.
9. Calculate the average of the converted values using the modules **Add Array Elements** and **Array Size**. Create an indicator showing the average value.
10. Apply a threshold by using the module **Greater or Equal**, and compare the average value to a constant. Set the constant to 50 and create an indicator called "**Hot?**" at the output of the **Greater or Equal** module.
11. Test the code by going to the Panel tab and placing all the controls and indicators. Input your name into the string control and view the result by pressing the play button.

**Hint:** You can change the name of any block by double-clicking on the name.



### Task:

Input your surname on “input string” field and retrieve the integer representation of its first three characters within a single execution of the program. Copy (take a screenshot) the block diagram and the front panel of your code, and include them in your logbook.

## Exercise 2: Implementation of the Central Limit Theorem


In this exercise you will visualize the central limit theorem by using loop structures, graphical blocks and arrays. Independent uniformly distributed random numbers will be generated, and the convergence of the sample mean value to a normal distribution will be observed as the number of trials increases. You should create and save a new VI called “CLT.gvi” under your project Lab1.

In a nutshell, we will generate a vector of independent and identically distributed random numbers (using a For loop), evaluate their sample mean, and iterate this within a while loop. After each iteration, we will update the histogram of the mean values to observe the convergence of the distribution.

### Instructions:

1. To observe the random experiment continuously, place a **While Loop** (in Program Flow) to the block diagram. The symbol on the lower left corner of the loop contains the value of the current iteration of the loop. You will also see a conditional terminal (the red button) created on the lower right corner of the while loop, which allows you to terminate the loop if a certain condition is met. To enter the desired stopping condition, right-click on the input terminal of the conditional terminal, and choose “Create control”. A stop button will also be available on the front panel.
2. Inside the loop, place a timing module called **Wait**, which will control the breaks between each execution of the while loop. To determine the duration of breaks, create a control for this node. Remember to place the corresponding control in the front panel.
3. Place a **For Loop** inside the **While** loop, keeping the **Wait** module outside it. As an equivalent of the nested loops in textual programming languages, place a second **For Loop** within the last one, which will allow you to create a 2D array.
4. Adjust the number of iterations of both **For Loops** by placing constants connected to the left-hand side loop count terminal ( symbol). This will allow you to define the dimensions of the 2D array you will create.
5. Search for the **Add Array Elements** module, and place the function outside the nested **For loops**, but within the **While** loop. This will allow you to sum the elements of the generated array, which is necessary for calculating the sample mean.
6. Inside the **For Loops**, place a **Random Number Generator**, and connect its output to the **Add Array Elements** function. Notice the boxes that are created on the frames of **For Loops**. These boxes are called **Tunnels**, and they specify how the data will be transferred through the frame. By default, these tunnels are indexing tunnels (right click on them and




ensure auto-indexing is selected). An auto-indexing output tunnel appends the single value obtained from a single loop iteration to an accumulating array of data.

7. To find the mean value, we need to divide the sum of the generated random values to the size of the random array. Find the **Divide** function in the block diagram and connect the output of **Add Array Elements** to the “X” terminal of it. Then right click on the “Y” terminal and create a constant. The constant should be equal to the total number of elements of the array (product of the number of rows and the number of the columns), so that the output of the division operation is the sample mean of your random array.
8. You need to create an array of these sample means to obtain a histogram. Start by placing an **Insert into Array** function on the block diagram, outside the **For loops** but within the **While loop**. This function inserts new elements into a specific position of an input array, and also has the ability to increase the size of the array. Connect the sample mean you calculated to the “New Element/Subarray” terminal.
9. Connect the  symbol of the **While loop** (loop iteration) to the “Index” terminal of **Insert into Array** function. This will assign the new mean value generated to its corresponding position within an array storing all results.
10. Now you need a base array to add all these new elements to, and you need it to be available at each iteration. Here a new concept called “**Shift Registers**” will be introduced. A shift register is a type of tunnel that carries information from one iteration to the next. First, connect the output array of **Insert into Array** function to the right frame of the **While loop**. Then right click on the tunnel and select “Change with Shift Register”. This will create another tunnel on the left edge of the loop. Connect this new tunnel to the “array” terminal of the **Insert into Array** function. The tunnel on the left edge carries the data from the previous iteration, whereas the one on the right edge sends the data to the next iteration. Finally, right click on the “input” terminal of the left register, and create a **constant array** so it can initialise the program with a 0 array.



11. Your loop is complete, and all you need is to create and observe the histogram. Search for the **Histogram** function and connect the output of the **Insert into Array** function to the “signal” terminal of the **Histogram** Function. Then create a control for “number of bins” terminal that will allow you to control the number of bins in the histogram.
12. Finally, to observe the changes in the histogram, create an indicator to the histogram terminal. *You should create the indicator within the while loop.* Add the corresponding indicator graph in the front panel.
13. Switch to front panel, set “milliseconds to wait” to 10, “number of bins” to 100, and run the code.

### Tasks:

-  Include the block diagram and the front panel in your logbook.
-  Create a control for stopping the execution of the program after 1000 iterations. Set the value of “number of bins” to 10. Include the corresponding histogram in your logbook.
-  Revise your code such that the final distribution you observe in the histogram is a standard Normal distribution (i.e., it has zero mean and unit variance). Include the revised block diagram in your logbook, and explain each of the changes you have made.

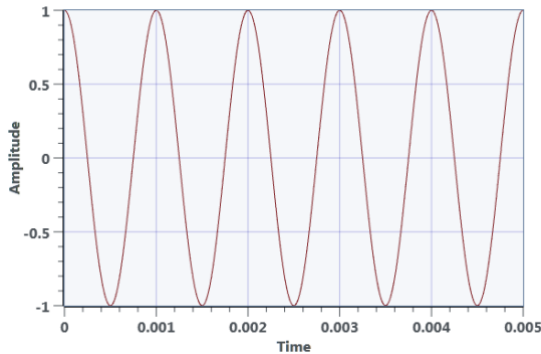
### Exercise 3: AM Modulator

In this part of the lab, you are going to build an AM modulator. Recalling the theory of AM, the standard equation for modulating a single tone is given by:

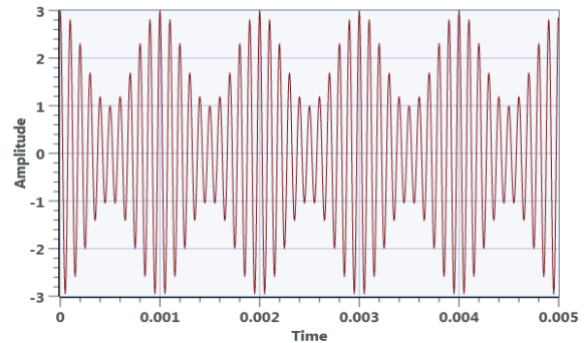
$$s(t) = [A_c + A_m \cos(2\pi f_m t)] \cos(2\pi f_c t),$$

where  $A_m$  and  $A_c$  are the amplitudes, and  $f_m$  and  $f_c$  are the frequencies of the message signal and the carrier waveform, respectively. An example of the application of this modulation is shown in the following figure, with the message signal given on the left and the modulated signal on the right. The envelope of the modulated signal is generated by the (shifted) message signal that has a lower frequency than the carrier signal.

Message Signal : Time domain



AM Signal : Time Domain



#### Instructions:


1. Create a new VI called "**AM.gvi**".
2. Create two **Sine Waveform** modules by using the "**Wave Generator**" module in the diagram window. These functions will be used to generate the message and carrier waveforms. For the adjustments of the generated signal, you should create the controls for "frequency", "amplitude", "sample rate" and "samples" terminals. Populate the values of the necessary terminals of the waveform generators according to the following table.

	Message Signal	Carrier Signal
Frequency	1kHz (Control)	10kHz (Control)
Phase	90° (Constant)	90° (Constant)
Amplitude	Message Amplitude (Control)	Carrier Amplitude (Control)
Sample rate	200k (Control)	200k (Control)
Samples	1000 (Control)	1000 (Control)

3. To observe the power spectrum of the message signal, search for the **FFT Power Spectrum and PSD** module. Insert it to the diagram and change its configuration to "Power Spectral Density" (PSD). Connect the message signal to the "signal" input terminal of the **FFT Power Spectrum and PSD** module. (NOTE: components may be hidden under larger component classes. If specific component names cannot be found, search for the name and choose the closest one then use the right-hand side panel to select the specific mode of the component).
4. Obtain the power spectrum of the message signal and the waveform plots (in time domain) of both the message and the carrier signals. To do these, create graphs in the panel, and then connect the created graph nodes that appear in the block diagram to the corresponding output terminal of the modules, e.g., the "power spectral density" terminal of the **FFT Power Spectrum and PSD** module and the "sine wave" terminal of the **Sine Waveform** module.
5. Perform the necessary operations in order to obtain the AM waveform.
6. As you did in step 3 and 4, create an indicator to show the AM waveform and the AM power spectrum.
7. You have now obtained the AM modulated signal. The **modulation index** is defined as  $\mu = \frac{A_m}{A_c}$ . You can observe the effect of the modulation index on the modulated waveform by changing the values of  $A_m$  and  $A_c$  via the corresponding controllers in the panel window.
8. The LabView Communications System Design Suite allows users to pack their VI's into **sub-**

**VIs**, which make it easier to include them in other VIs. This also makes your block diagram look neater, and creates a modular code that can be modified easily. In the following steps we will explore this functionality.

- First, click on the **Edit Icon** button located on the right hand side of the top banner. Now you can click on the various terminals to configure them either as an *input* (left) or an *output* (right). In this example, the **inputs** are: message signal frequency, message signal amplitude, carrier signal frequency, carrier signal amplitude, sampling info; while the **outputs** are: message signal, carrier signal and AM waveform time domain representations, and message signal and AM waveform PSDs.
- Once saved your VI is ready for future use as a sub-VI.

*Hint: Explore the **Cleanup Diagram** tool (  ), on the top banner of the diagram tab, to make your diagram look neater.*

*Hint: If necessary, you can adjust the number of terminals on each side of the icon by dragging and expanding the icon itself. Once you connect the necessary inputs and outputs you can use the palette on the left hand side of the screen to design a suitable image for the icon of your GVI.*

### Tasks:


- ☞ Adjust all the plots you retrieve so that they are clear and easy to understand.
- ☞ Include the block diagram in your logbook.
- ☞ Set  $A_c = 2$  and change  $A_m$  in the panel window to obtain the modulation index values  $\mu = \{0.5, 1, 1.5\}$ . For each value of  $\mu$  plot the time domain and PSD representations of both the message and modulated signals.
- ☞ What is the impact of the modulation index on the modulated signal? Explain your answer.
- ☞ Set  $A_c = 1$  and  $A_m = 1$ . Change the frequency of the message signal to  $f_m = \{1k, 2k, 5k\}$ . What is the impact of the varying frequency value on the modulated signal? Explain your answer.

**DO NOT FORGET TO SAVE THE PROJECT SINCE SOME OF ITS FILES WILL BE USED IN THE NEXT LAB!**

## APPENDIX

### Adjusting Plots with LabView

When a new plot is added to LabView, a default graph is created, adjusted automatically to the input data. However, in order to visualise and analyse data, further steps may be needed.

1. **Zooming and panning** – after selecting the graph in the panel, tick the **Graph Tools** option on the right hand panel under “Parts”. A menu with zooming and panning options should appear under the graph. 
2. **Renaming axes** – Every graph shows “time” and “amplitude” as default axis labels (independently of the input data). To change the names to something meaningful, just double click on the axis name.
3. **Exporting and visualizing data** – Right click a plot and select “**Capture Data**”. This will export the current data to the “Data” menu (left of the screen) where you can visualize the data in a bigger screen and save it. It is also possible to drag captured data to a graph in the panel, to visualize it.
4. Many other settings can all be found under the right panel.