Experimental Techniques for Physics and Engineering Labs

Experimental Techniques for Physics and Engineering Labs:

A Unique Approach using LabView

By

Emadelden Fouad, Antonio Ruotolo
and Sesha Srinivasan

Cambridge Scholars
Publishing

Experimental Techniques for Physics and Engineering Labs:
A Unique Approach using LabView

By Emadelden Fouad, Antonio Ruotolo and Sesha Srinivasan

This book first published 2026

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-1-0364-6932-0
ISBN (Ebook): 978-1-0364-6933-7

TABLE OF CONTENTS

PREFACE

Engineering and Physics laboratory courses are meant to bridge the gap between theory and real-world application. In this respect, LabVIEW stands out as a powerful graphical programming platform to design and automatize systems of practical applications.

This book was written with the goal of guiding students through the multifaceted world of LabVIEW. We begin with the basics: understanding the environment, navigating the interface, and creating simple Virtual Instruments (VIs). From there, we explore data flow programming, advanced controls, hardware integration, signal processing, and automation techniques.

In the second part of the book, case studies and examples based on semiconductor devices are proposed with dual intent to make the students apply the acquired knowledge of LabView and to understand physics and the applications of practical electronic devices. Each chapter begins with clear learning objectives and ends with tasks and practical problems that reinforce key concepts.

This book would not have been possible without the valuable feedback of hundreds of undergraduate students across a variety of engineering disciplines who have adopted it during their studies.

Let your journey with LabVIEW begin.

FOREWORD

This book entitled, “Experimental Techniques for Physics and Engineering Labs” is a practical guide in designing Physics and Engineering projects using LabView programming and Keithley instruments-enabled automation. The basics of LabView such as step-by-step instruction in developing sub-vi files, graphs and structures, file management, and debugging are discussed in the first part of the book. The second part deals with the electrical, optical, and magnetic characterizations of semiconductor devices that were systematically evaluated and tested by design-of-experiments (DOE) using LabView software and the Keithley hardware. The Authors have provided enough information for students and other professionals who will be following this book to critically think “outside the box” for developing and executing the lab projects, for instance, integrating LDR and Arduino Uno. Overall, this handbook will serve undergraduate and graduate students and practitioners to adopt Physics and Engineering design projects using LabView automation. As the Professor of Electrical and Computer Engineering and the Senior Member of IEEE, I strongly recommend this book as an asset for higher education and industries for training in advanced Physics and engineering areas.

Muhammad S. Ullah, Ph. D.
Senior Member, IEEE | Associate Professor
Department of Electrical and Computer Engineering
Florida Polytechnic University
Lakeland, Florida USA

FLORIDA POLYTECHNIC UNIVERSITY

LABORATORY 1 – BASICS OF LABVIEW

Learning outcomes:

- Basics of LabVIEW, the Front Panel and the Block Diagram
- Controls/Indicators and Terminals
- While loops
- Timing
- Data types
- Waveform charts
- Property Nodes

Basics of LabVIEW, the Front Panel and the Block Diagram

LabVIEW is a Graphical Programming Environment and Programming Language created and distributed by National Instruments.

It has all the features of an ordinary programming language (loops, shift-registers, etc.) but it's graphical. You won't write any code. You will DRAW the code!

What's the difference between LabVIEW and, let's say, Visual Basics? LabVIEW is inherently conceived to control instruments, acquire data and automatize industrial processes.

Why am I learning it in this course? Because you will setup up and automatize experiments that prove concept of Modern Physics and Engineering disciplines.

LET'S START: Launch LabVIEW and choose "Blank VI". Instead of one window popping up, you got two of them (but they will be saved in only one file called a Virtual Instrument ".vi"). What you will learn to love about VI's is that you can use them as sub-VI's in another VI. Your hard work is never wasted!

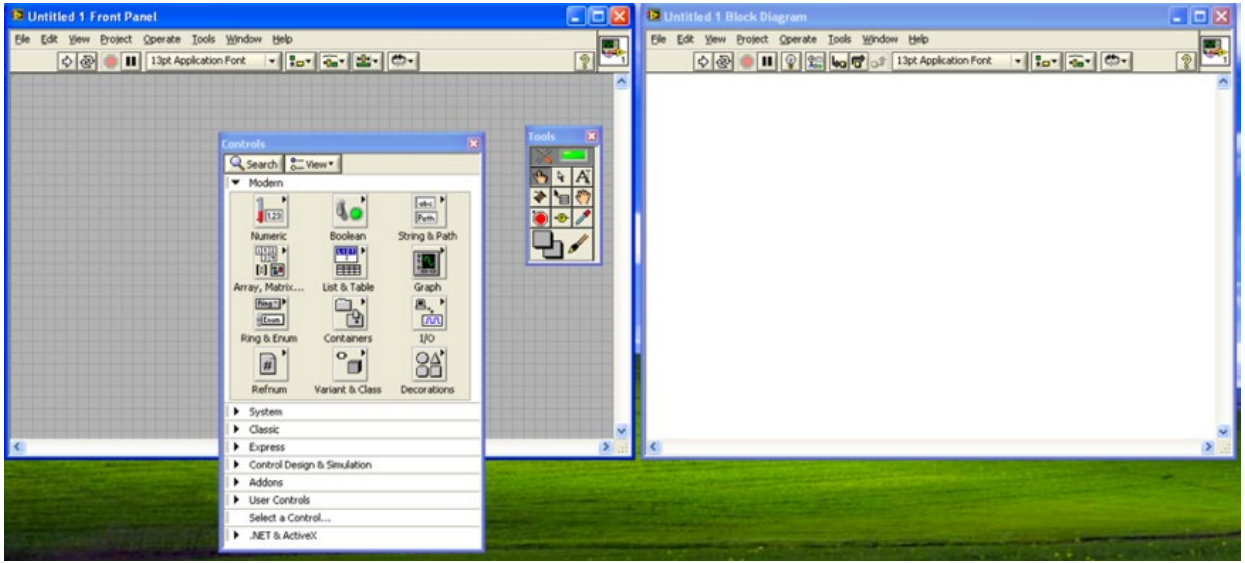

Figure 1.1: Front Panel (Left) and the Block Diagram (Right) of the VI program.

You should see the **Tools palette**. If not, go to the menu “View” and make it appear. You need two more palettes: the Controls palette in the Front Panel and the Functions palette in the Block Diagram. If they bother you, just close them. They will pop up, respectively, as soon as you right click on the Front Panel window or the Block Diagram window. Try it now.

Why two windows? The front panel will eventually be the user interface. You could compile the VI into an executable file and the user would never see the block diagram. You are not there, yet. We need to learn how to program in LabVIEW. *Note: CTRL+E let you switch quickly between the two windows*.

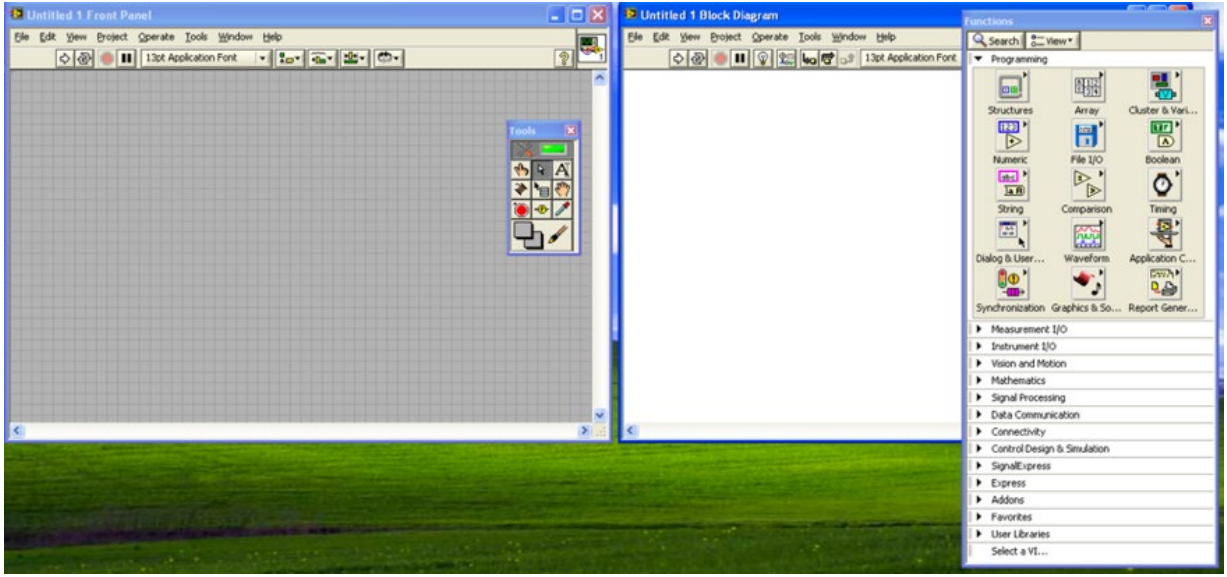

Figure 1.2: Front Panel with Tools palette (Left) and the Block Diagram with Functions palette (Right) of the VI program.

Controls/Indicators and Terminals

Controls are your input variables; indicators are your output variables. A control can be as simple as a constant. An indicator can be as complex as a graph. They exist in both the front panel and the block diagram.

Right click anywhere in the Front Panel and choose a Numerical: Vertical Pointer Slider. Place it on the Panel and look at what happens in the Block Diagram. Rename it “Voltage”.

If you are an aesthete, you can use decorations. For instance, in the picture, I placed a horizontal smooth box, sent it to back with the Reorder button (the last one on the bar), changed color of Voltage to blue with the Application Font bar and aligned the three objects with the button Align Objects. Yes, I am not a great artist! I am sure you can do better.

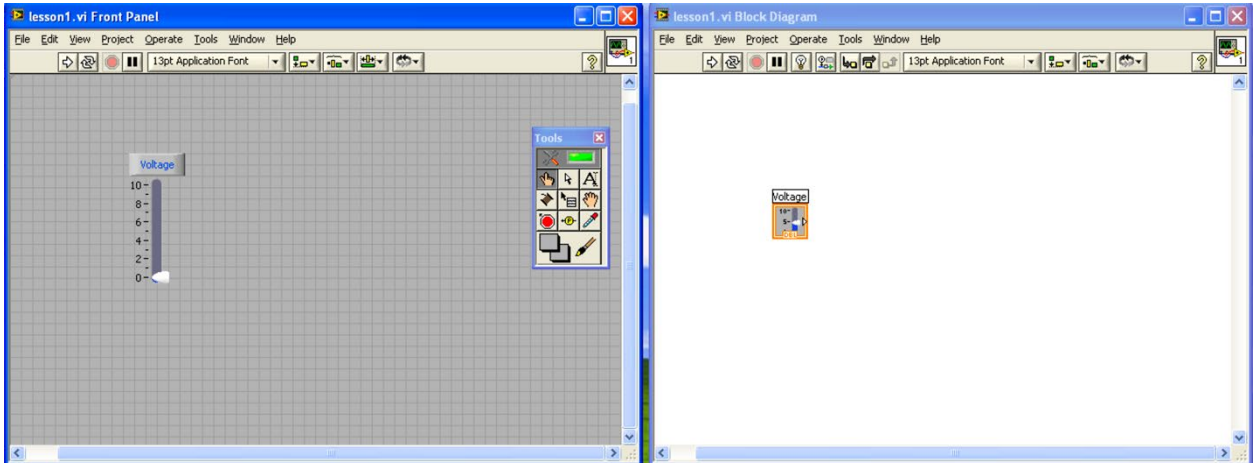

Figure 1.3: Front Panel with Voltage slider (Left) and the Block Diagram with Voltage icon (Right) of the VI program.

The default range is 0-10 but we want it to be 0-20. Select “Edit text” on the Tools Palette or… magic…: activate “Automatic Tool Selection”. You are going to love this. Your cursor is now smart. Hover over the Front Panel. The cursor “guesses” what you want to do. And you can get rid of the Tools Palette if it bothers you. Turn 10 into 20 and see what happens. The pointer slide rescales automatically.

(Alternatively, right click on the Voltage cursor and browse the tabs. We will go back to one of them later when we discuss “Data types”).

You work for a company that acquires the voltage signal from a sensor (for now, we will change it manually) and convert it into a force (max 100 N) on a piston with area 10 cm^2.

Place a numeric tank on the Front Panel and call it Pressure. Rescale the tank to the max pressure (100N/1E-3 = 10000 Pa). That’s a big number, let’s use scientific notation. Right click on the tank and choose scientific notation in the “Format and Precision”.

Time to do some math. A math that the end-user does not need to see. Which is to say, from now on, we will work in the Block Diagram window.

The maximum voltage (20 V) must correspond to the maximum pressure (1E+5). Math is easy: multiply by 5 and divide by 1E-3. When you multiply/divide, go on the y-terminal of the Multiply/Divide block, right-click and create a constant. Wire the inputs and outputs as shown in the figure below. Slide your Voltage cursor on 10 V, click on Play and see if the math is right.

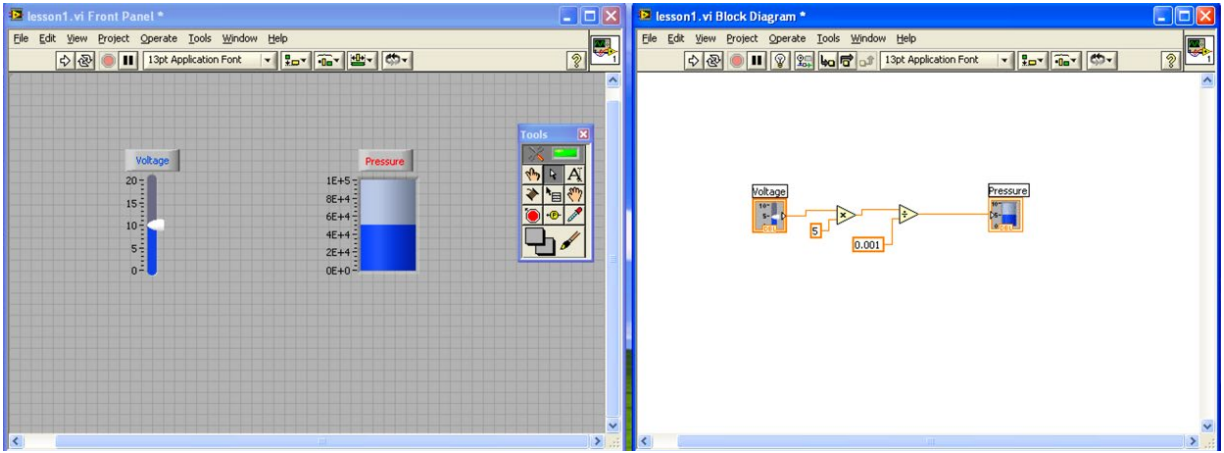

Figure 1.4: Front Panel with Voltage and Pressure sliders (Left) and the Block Diagram with Voltage and Pressure icons with ranges (Right) of the VI program.

NOTE: If math becomes complicated, you could use the Structure: Formula Node or Structure: Math Script.

Do I need to click on “Play” every time? Of course not.

While loops

Draw a “Structure: While Loop” around everything. Right click on the “loop condition” (the red dot) and create a Control. The color is green and not orange. Why? Because it’s a Boolean control button (different datatypes have different colors). Find the button on the Control Panel and arrange it in a convenient location. If it’s too small for Automatic Tool Detection, use the Position/Size/Select on the Tools Palette. Press the play button now and play with the slider.

Congratulations! You have just made your first VI. You could make it executable and sell it. Unfortunately, it’s far too elementary to make money out of it. We have a long way to go before cashing in. A bit of patience. We will get there.

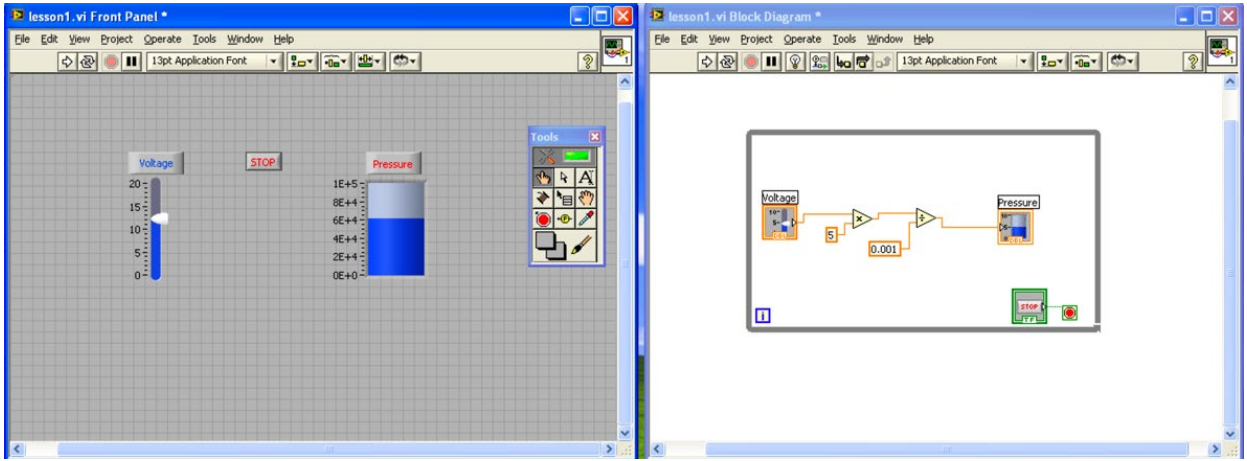

Figure 1.5: Front Panel with Voltage and Pressure sliders (Left) and the Block Diagram with Voltage and Pressure icons with controlled values (Right) of the VI program.

Timing

Time for you to grow into an engineer and Physicist and understand that the world out there is far from ideal. Do you think you can change the pressure of a piston in 1 ns? LabView executes the program as fast as it can (which depends on how fast your computer is). A piston is not that fast. You need to manipulate the speed at which an operation is executed. Assuming the piston needs a maximum of 1 second (depending on the change of voltage), then you can force any cycle of the while loop to last 1 second, no matter what (for instance, another application might slow down LabVIEW in a way you can't control). Then you need a "Wait until next (milliseconds).

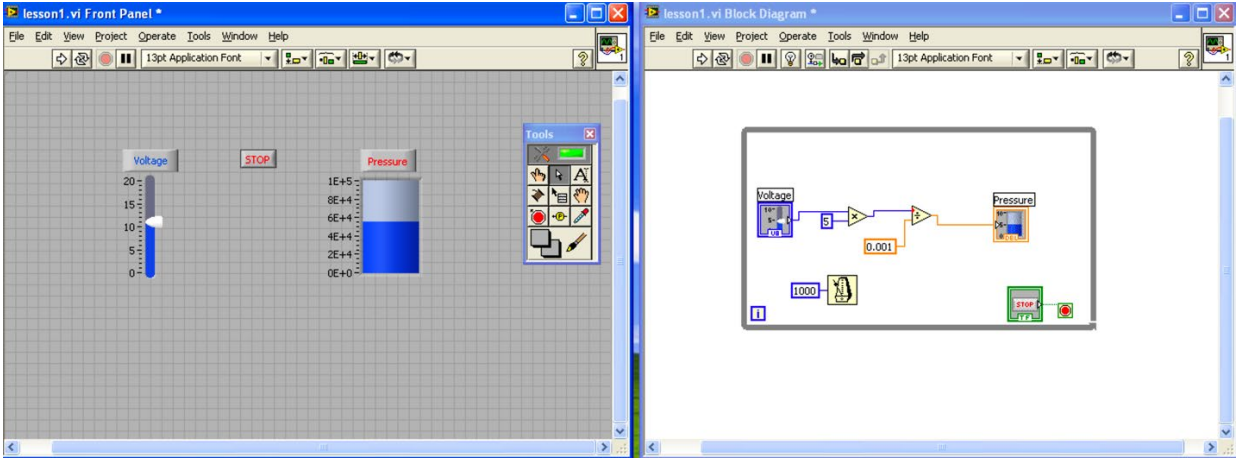

Figure 1.6: Front Panel with Voltage and Pressure sliders (Left) and the Block Diagram with Voltage and Pressure icons and a timer delay icon (Right) of the VI program.

You are not sure whether you should use “wait (milliseconds)” instead?

Trick: use “Context Help”. Go to Help and choose “Show Context Help” (shortcut CTRL + H). An explanation will pop up every time your cursor hovers over a Function. Try it now.

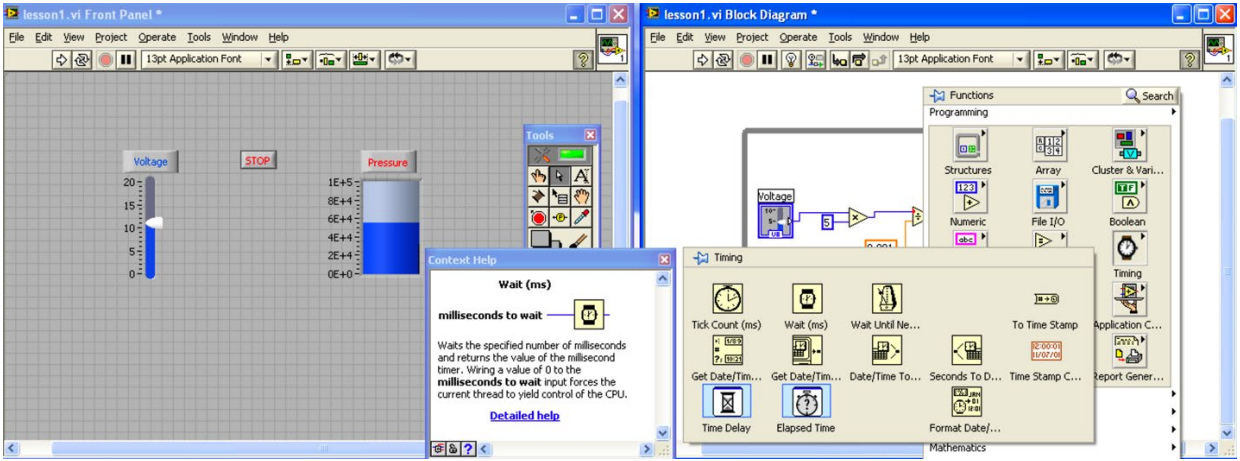

Figure 1.7: Front Panel (Left) and the Block Diagram (Right) with context help window of the VI program.

The number of elements on your block diagram starts being significative. Double-click on any element in your Front Panel or on any element in the Block Diagram. Useful, isn't it?

Data types

As you have learned in your foundation courses of programming, you can have different data types. In LabVIEW, constants, controls and indicators can be of different datatypes and when you connect them with wires, you can only wire terminals of the same data type. The most important are summed up below.

LabVIEW uses different colors to help you. Besides, it chooses the right type for you if you right click, like you did before on the y-terminal of the Multiply/Divide operator.

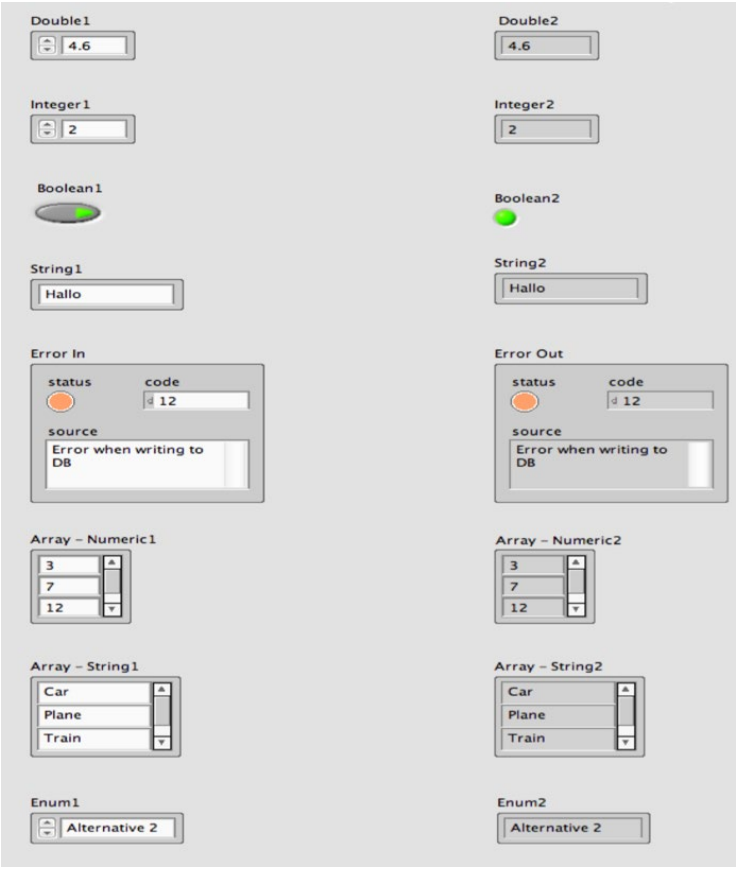

Figure 1.8: Datatypes-VI programing: Scheme View 1

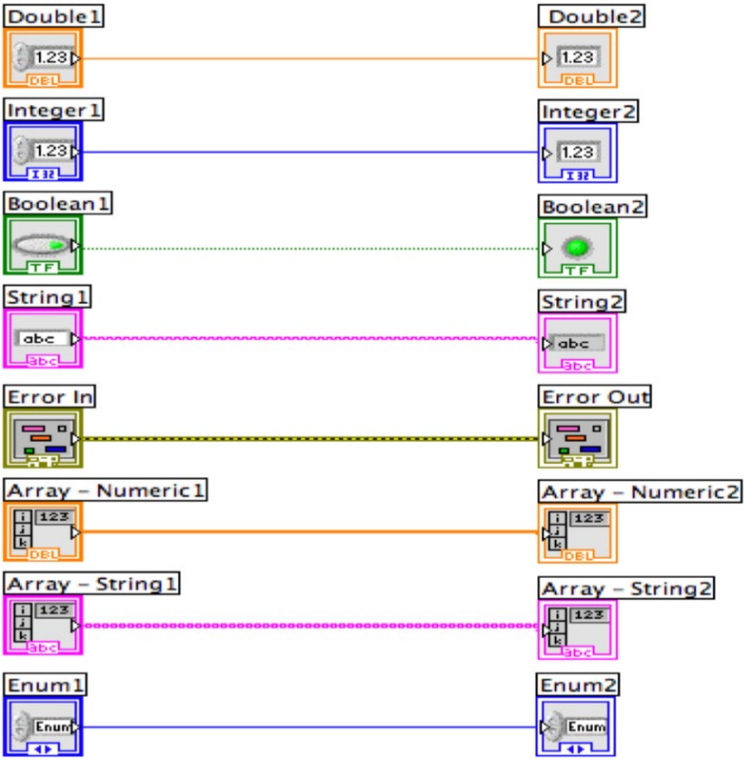

Figure 1.9: Datatypes-VI programming: Scheme View 2

Error is a particular type of cluster which you will use only when controlling physical instruments.

Go back to your VI. Right click on your Voltage slider and choose "Properties". Go to the tab "Data Range". Change datatype into Unsigned Byte, unclick "Default Range" and set Minimum to 0, Maximum to 20 and Coerce to Nearest. Run the program and see what happens when you change the voltage.

Go to the Block Diagram, unwire the "5" and delete the constant "5". Right click on the y-terminal and create the constant "5" again. Do you understand why the color of the output wire has turned blue? Can you do the same with the constant "0.001"? Of course not!

Good. You start understanding why, although counterintuitive, when creating constants, controls and indicators, it might be smarter to do it on the block diagram rather than on the front panel. LabVIEW chooses the right datatype for you.

Waveform Charts

Alright, you have been spoon-fed too much today. Your turn. Build up the Block Diagram for the following VI:

TASK: You acquire a temperature, which is between 20- and 100-degree Kelvin (right click on your controller and make these two values "Default values"). Simulate it with the Numeric: Random number. You get a random number in the range 0-1, to be converted to the range 20-100.

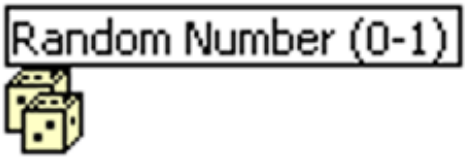

Figure 1.10: Random number generator feature of VI programming.

You want to plot it (one acquisition per second) on a Waveform Chart and make it appear on a thermometer. *What's the difference between a Waveform Chart and a Waveform Graph? We will see it in the next lab session.*

Submit the solution on Canvas (your VI file or a snapshot of the Block Diagram) before leaving the lab. Call this VI: RandomTemp.vi

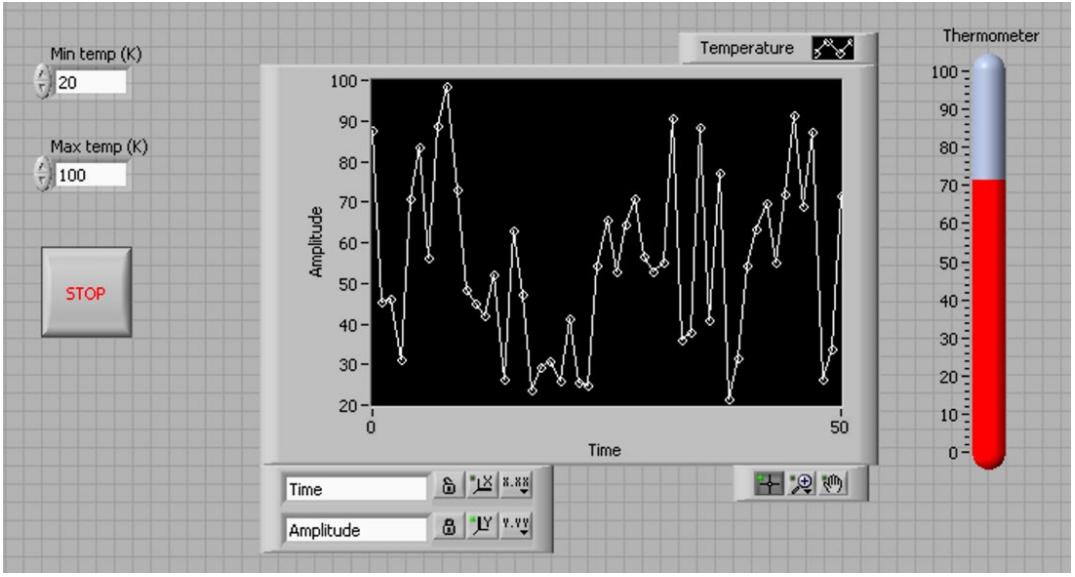

Figure 1.11: VI Temperature graph with minimum and maximum ranges and a temperature slider.

CHALLENGE (This is a bit advanced for you at this stage but… your lab instructor will help you):

The chart does not clear up when you stop and start again. Besides, if you change maximum temp, the maximum of the thermometer remains “100”, which is a problem if you increase the max temperature.

Property Nodes

HINT: Look for Property Nodes. For the chart, create a “write” property node history and use a constant array of zeros. For the thermometer, create a “scale maximum” node and a “scale minimum” node.

LAB 1 OBSERVATIONS AND REMARKS

(This page intentionally left blank for making additional Notes and Results)

LABORATORY 2 – CHARTS VS. GRAPHS

Learning outcomes:

- Charts vs. graphs
- Shift-registers
- Arrays and For Loops
- Clusters

Charts vs. Graphs

In the previous Lab, you learned to use Waveform Charts. What's the difference between a "Chart" and a "Graph"?

- A chart reminds us of history and adds a new point at the end of the plot. Therefore, you use it inside a 'while/for' loop and a point is added every time the loop is executed.
- A graph plots an entire array of data at once. Therefore, it is used outside a 'while/for' loop.

Let's try the following. Make a copy of your RandomTemp.vi. Add a waveform graph and wire the same output you send to the chart and to the thermometer to the graph. You get an error. It makes sense, doesn't it?