BE/EE/MedE 189a: Design and construction of biodevices

Justin Bois

Caltech

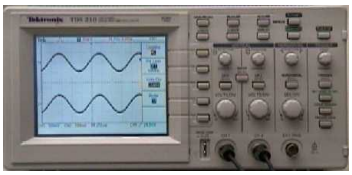
Winter, 2017

1 LabVIEW basics

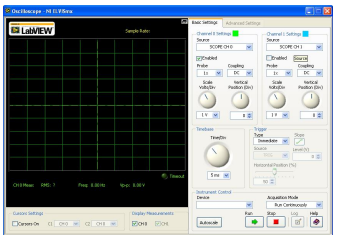
In this class, we will use a National Instruments ELVIS II breadboard to build out devices. We will connect these breadboards to a computer to received signals and control components. To facilitate this communication, we will use LabVIEW, a software package produced by National Instruments.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a **visual programming language** (VPL) in that the programmer manipulates graphical elements to design a program instead of using text. Further, LabVIEW programs are intended to provide computer-based functionality that mimics that of a real, physical elecotronic instrument in a laboratory. For this reason, LabVIEW programs are referred to as **virtual instruments**, or VIs.

A classic example of such a virtual instrument is an oscilloscope, depictd in Fig. 1, which allows measurement and display of constantly varying voltages over time. The left oscilloscope is a physical instrument, with its display panel showing a plot of voltage over time. To the right is a LabVIEW implementation of a virtual oscilloscope. The virtual oscilloscope has knobs and buttons like the physical instrument and a live plot, but the "electronics" of the virtual instrument are graphic computer code behind the front panel with the knobs and display.



real oscilloscope



virtual oscilloscope

Figure 1: Left, a physical oscilloscope. Right, a virtual oscilloscope.

1.1 VI components: Front Panel and Block Diagram

When you open a new VI, which you can do by selecting New VI from the File pulldown menu, you will get two windows, one in back labeled Block Diagram and on in front labeled Front Panel. Fig. 2 shows the front panel and block diagram for a VI that generates a waveform.

As the name suggests, the front panel is the interface of the VI. Like any computer program, the interface takes inputs and displays or generates outputs. In LabVIEW speak, inputs are called **controls** and outputs are called **indicators**.

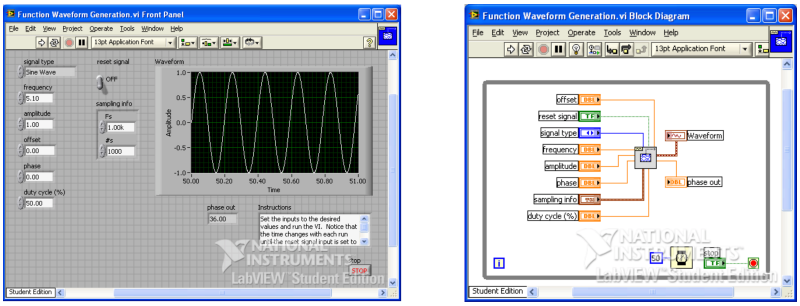


Figure 2: Left, a front panel of a waveform generator. Right, a block diagram of an waveform generator.

The block diagram contains the guts of the program. When you create controls and indicators in the front panel, the corresponding components appear in the block diagram as colored objects called **terminals**. The block diagram contains arithmetic operations, functions, constants, subVIs (akin to a submodule). The inputs and outputs of these objects flow through **wires**, which connect the object.

In my typical workflow, I build the front panel first. This is what I want my instrument to *do* and how I want to be able to control it. If you are used to programming in other languages, you can think of front panel design as design of your API, which you typically do first.

1.2 An example VI: Fahrenheit to Celsius converter

This is all rather abstract, and perhaps a bit complicated since a waveform generator is not the simplest VI we could imagine. Consider now a VI for converting an inputed number in degrees Fahrenheit to degrees Celsius, shown in Fig. 3.

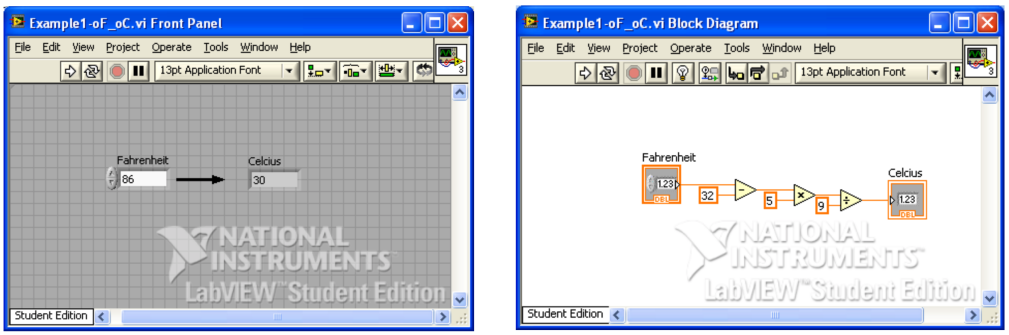


Figure 3: A VI for Fahrenheit to Celsius conversion.

First, let's consider the front panel. We have a control (or input) in which the user specifies a number corresponding to the temperature of interest in degrees Fahrenheit. There is an indicator (or output) that gives the same temperature in degrees Celsius.

Looking at the block diagram, we see an input of data type DBL, or double. To convert to Celsius, we use the following formula.

$$C = \frac{5}{9}(F - 32). \tag{1.1}$$

So, we first subtract 32 from the inputted degrees Fahrenheit. This is accomplished with the minus mathematical operator. It takes two inputs, shown by the two orange wires to the left of the minus operator, and subtracts the bottom input from the top. So, the wires carry variables into operators. Coming out of that minus operator is the resulting difference of the two inputs. This then flows into a multiplication operator, which also takes two inputs, and then multiplies them together. We need to multiply by five, so this is the other input. The output is then divided by nine and delivered to the Celsius indicator.

1.3 The controls palette

You can add controls and indicators to your front panel using the **Controls Palette**. If it is not already in view, you can make it visible by selecting View → Controls Palette. An example annotated front panel is shown in Fig. 4, and an example controls palette is shown in Fig. 5. To add a control or indicator to the front panel, simply click on the icon in the controls palette and drag it onto the desired space in the front panel.

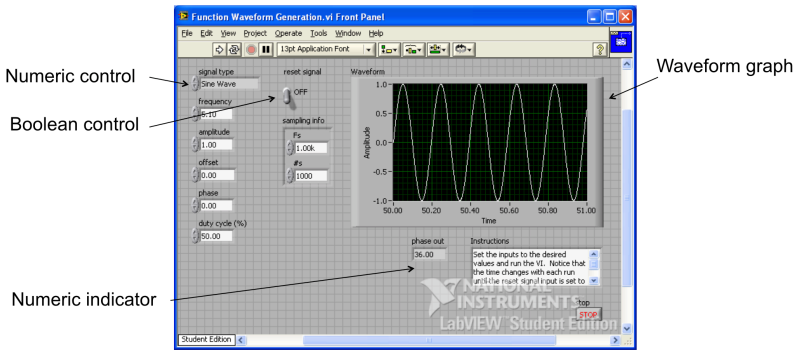


Figure 4: An annotated front panel for a waveform generator with various controls and indicators.

1.4 The functions palette

As the controls palette is your main resource for designing your front panel, the **Functions Palette** is your main resource for designing your block diagram. An annotated functions palette is shown in Fig. 6, and an example functions palette is shown in Fig. 7. The block diagram has a while loop and a subVI, both of which we will cover

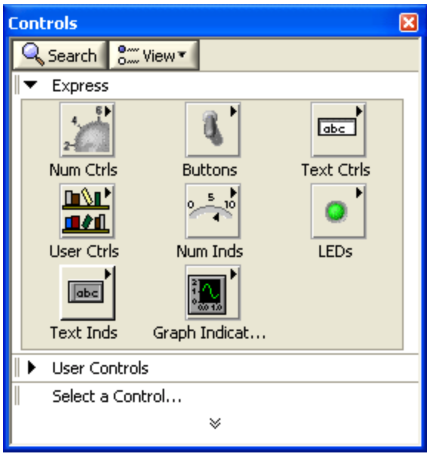


Figure 5: An example controls palette.

in coming lab sessions. The functions palette shows a subset of the many functions available.

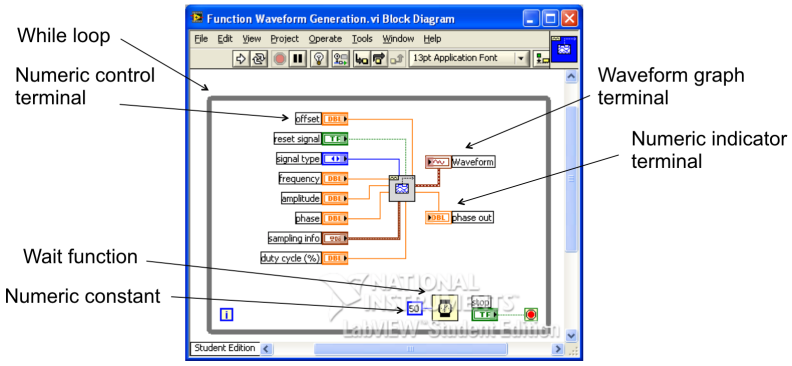


Figure 6: An annotated block diagram for a waveform generator. In the center is a subVI, which we will learn about in coming days.

1.5 The tools palette

The **Tools Palette** is useful for selecting and connecting element of your VI that you drug in from the functions and controls palettes. The **wiring tool**, which looks like a spool of wire, is used to connect objects in the VI. The position tool (the arrow) is used to select objects and then to position them. The pointed finger tool is used to operate your switches, knobs, sliders, etc., on the front panel.

Importantly, by selecting the automatic tool selection, LabVIEW will infer which tool you want to use based on where your pointer is. This is quite useful and can save you some clicking.

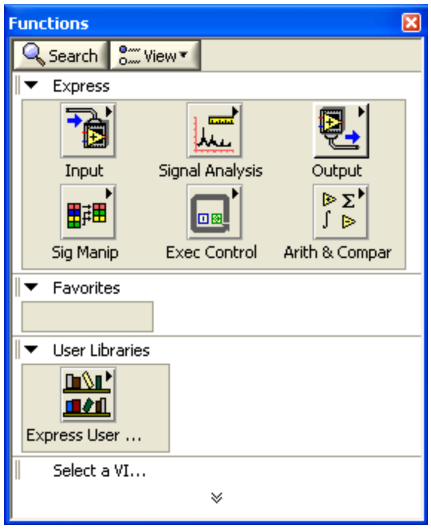


Figure 7: An example functions palette.

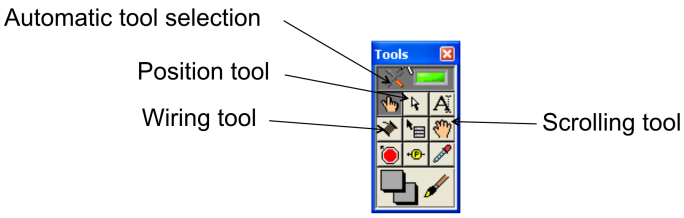


Figure 8: The annotated tools palette.

1.6 Tools to make your VI look pretty

Just as style is important in text-based programming, so too is it important in LabVIEW. Fortunately, LabVIEW offers several tools to prettify your VI. On the tool bar of both the front panel and the block diagram are alignment, distribution, and resizing tools for objects. You can spot them from the green and yellow colored boxes on their tabs. These tools work much as similar tools work in drawing/layout programs, such as Illustrator or PowerPoint.

You can also "clean" your wires, which can get bendy based on where the input and output nodes are by selecting a wire with the position tool and then right clicking and selecting Clean Up Wire. LabVIEW will then do its best to straighten and otherwise prettify the wire.

You can take a more, shall we say, fervent approach by clicking on the Clean Up Diagram button at the right of the toolbar of the block diagram window (the icon has a addition operator with a broom). This chooses an arrangement of objects and wires that is in some way optimal. Except for very simple VIs, I usually do not take this option because the rearrangements can sometimes be extreme. The result is

straighter wires and convenient spacing, but my reasoning for how I set up the block diagram is destroyed.

1.7 Running your VI

After you have built a VI, you want to **run** it. To run a VI once, click the forward white arrow on the toolbar of the front panel. Alternatively, you can hit Ctrl+R or select Operate → Run.

LabVIEW has the useful capability of running continuously. That is, it will listen for a change in a control, such as the user changing the value of a slider, and then change the indicator on the fly. To run continuously, click the two cycling arrows on the toolbar of the front panel.

You can abort execution by clicking the stop sign on the front panel toolbar, or pause executing by clicking the pause button.

1.8 A little practice

Before diving into your first homework, you might want to practice making a couple VIs. For your first practice, make a front panel with a couple of vertical toggle switches that you can flip up or down. Add an LED indicator that turns on in all instances except when both switches are up (a NAND gate). The VI is shown in Fig. 9.

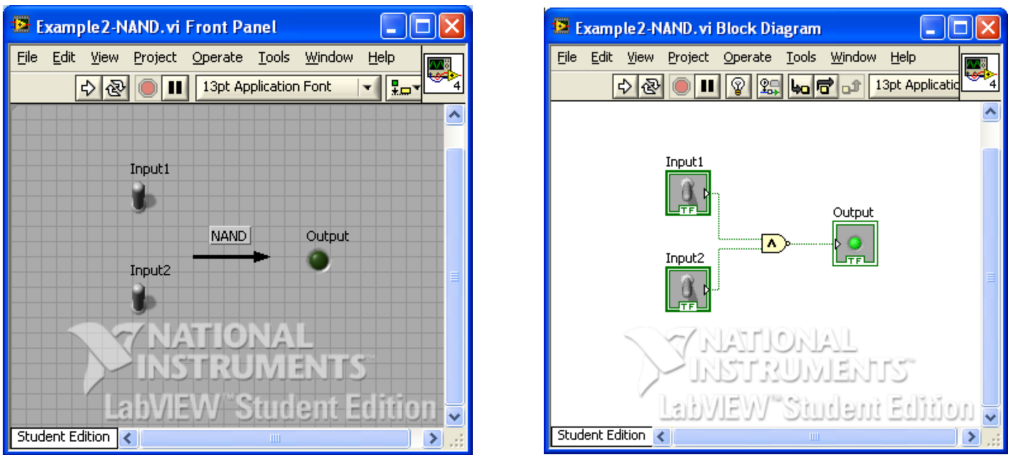


Figure 9: A VI for a NAND gate.

For your second practice, implement the Fahrenheit to Celsius converter in Fig. 3.