


Introduction to LabVIEW

Chris Chambers
McGill University





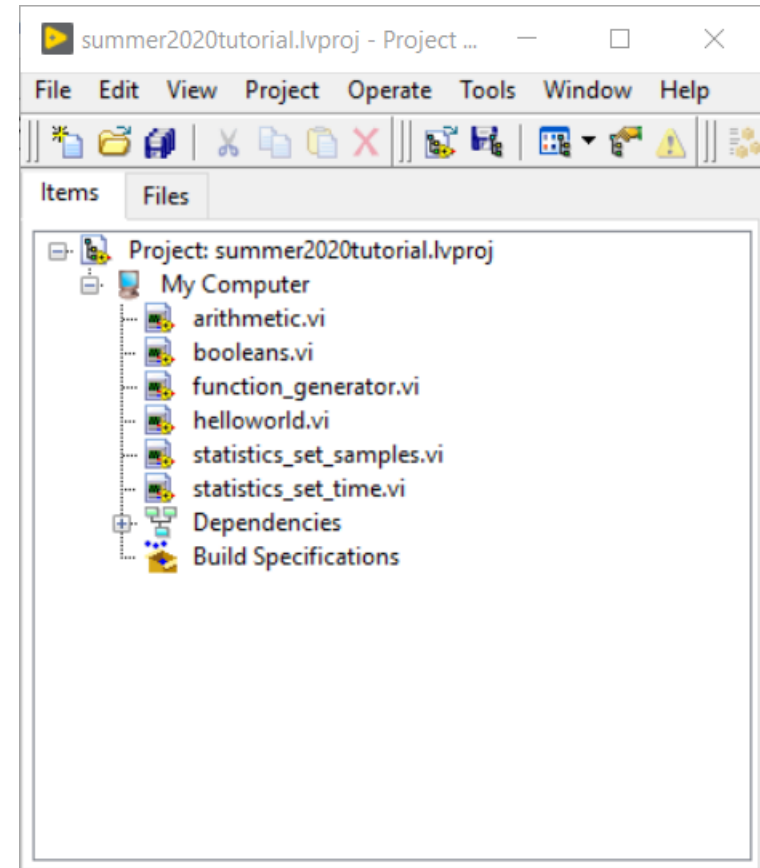
Outline

- Organizing your programs
 - Intro to visual programming
 - Expanding the toolbox
 - Generating and plotting data
 - Connecting to (simulated) equipment
- 

Organizing your Programs

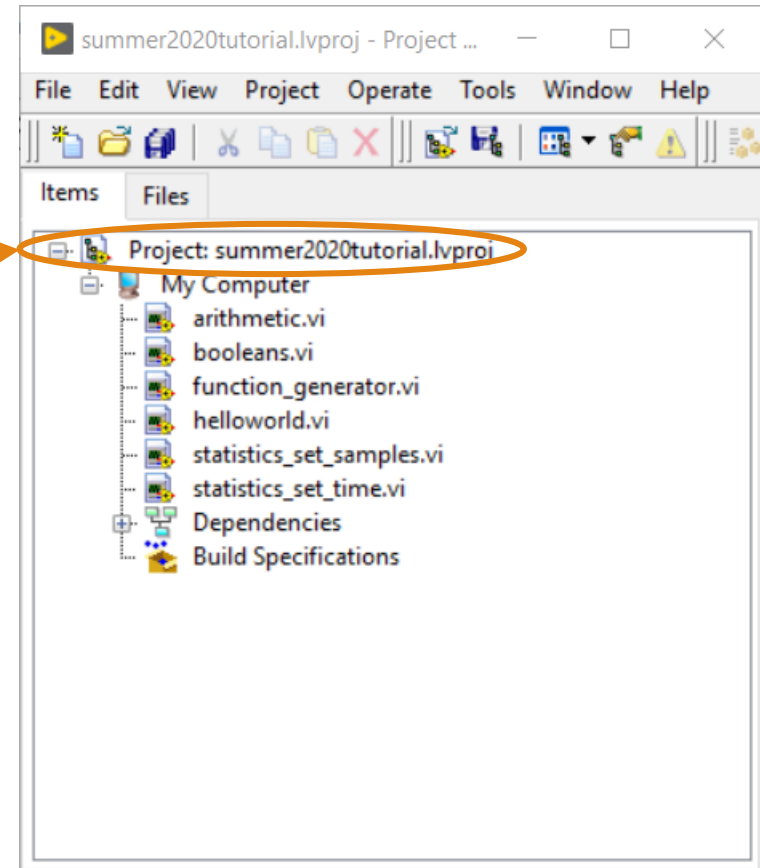
Project Architecture in LabVIEW

- LabVIEW has 2 main types of files
 - vi : Analogous to a script in another language
 - This will be your main file for programming
 - File extension: **.vi**
 - Project : Groups vi's together for easier development
 - Store all the vi's for a project in one location
 - File extension: **.lvproj**
 - sub-vi : A vi that is meant to be used inside another vi
 - LabVIEW has many built-in sub-vi's
 - You can also write your own
 - File extension: **.vi**



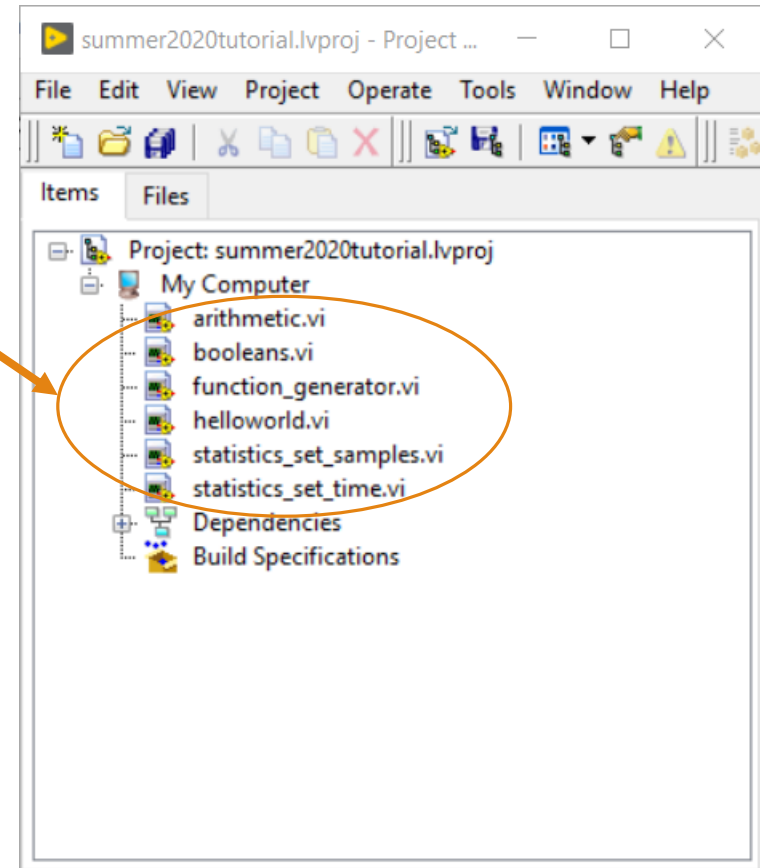
Project Architecture in LabVIEW

- LabVIEW has 2 main types of files
 - vi : Analogous to a script in another language
 - This will be your main file for programming
 - File extension: **.vi**
 - Project : Groups vi's together for easier development
 - Store all the vi's for a project in one location
 - File extension: **.lvproj**
 - sub-vi : A vi that is meant to be used inside another vi
 - LabVIEW has many built-in sub-vi's
 - You can also write your own
 - File extension: **.vi**



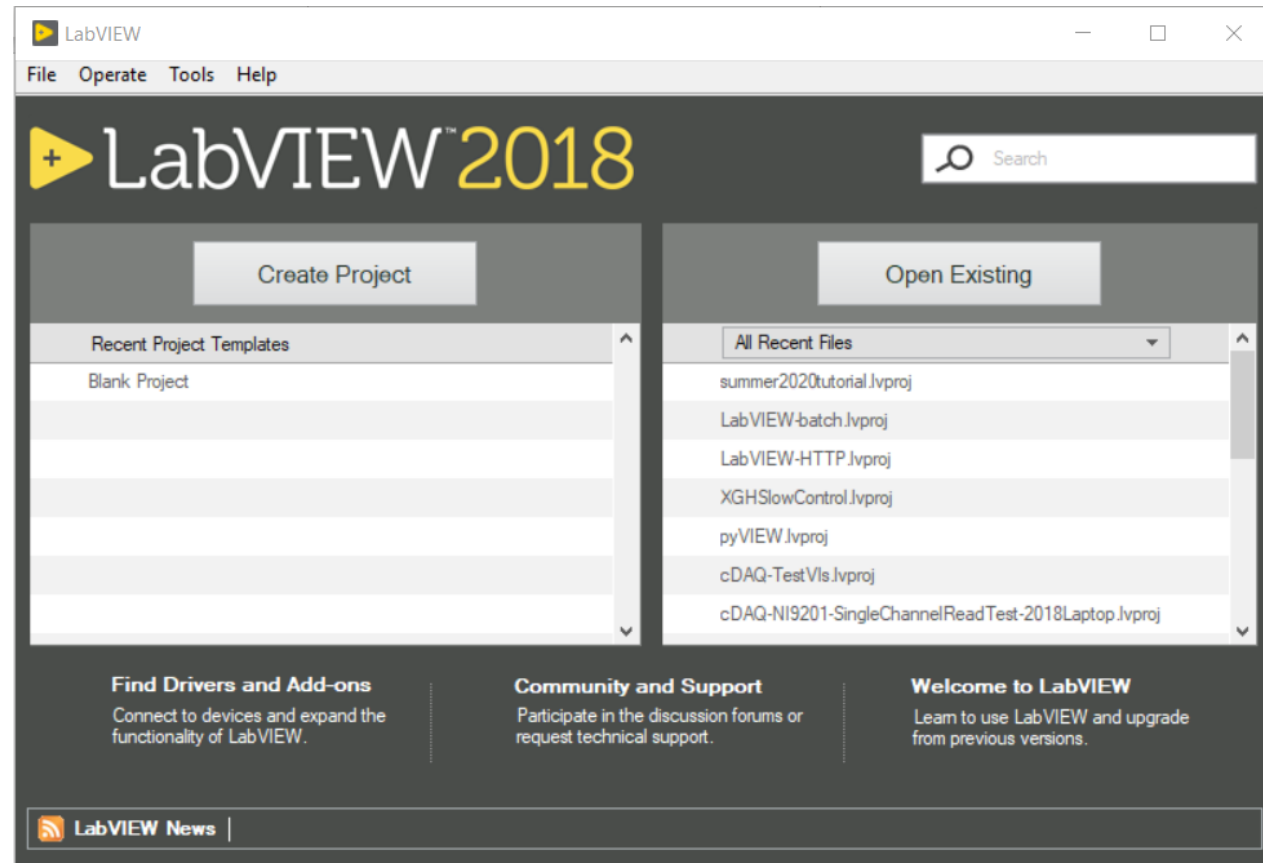
Project Architecture in LabVIEW

- LabVIEW has 2 main types of files
 - vi : Analogous to a script in another language
 - This will be your main file for programming
 - File extension: **.vi**
 - Project : Groups vi's together for easier development
 - Store all the vi's for a project in one location
 - File extension: **.lvproj**
 - sub-vi : A vi that is meant to be used inside another vi
 - LabVIEW has many built-in sub-vi's
 - You can also write your own
 - File extension: **.vi**



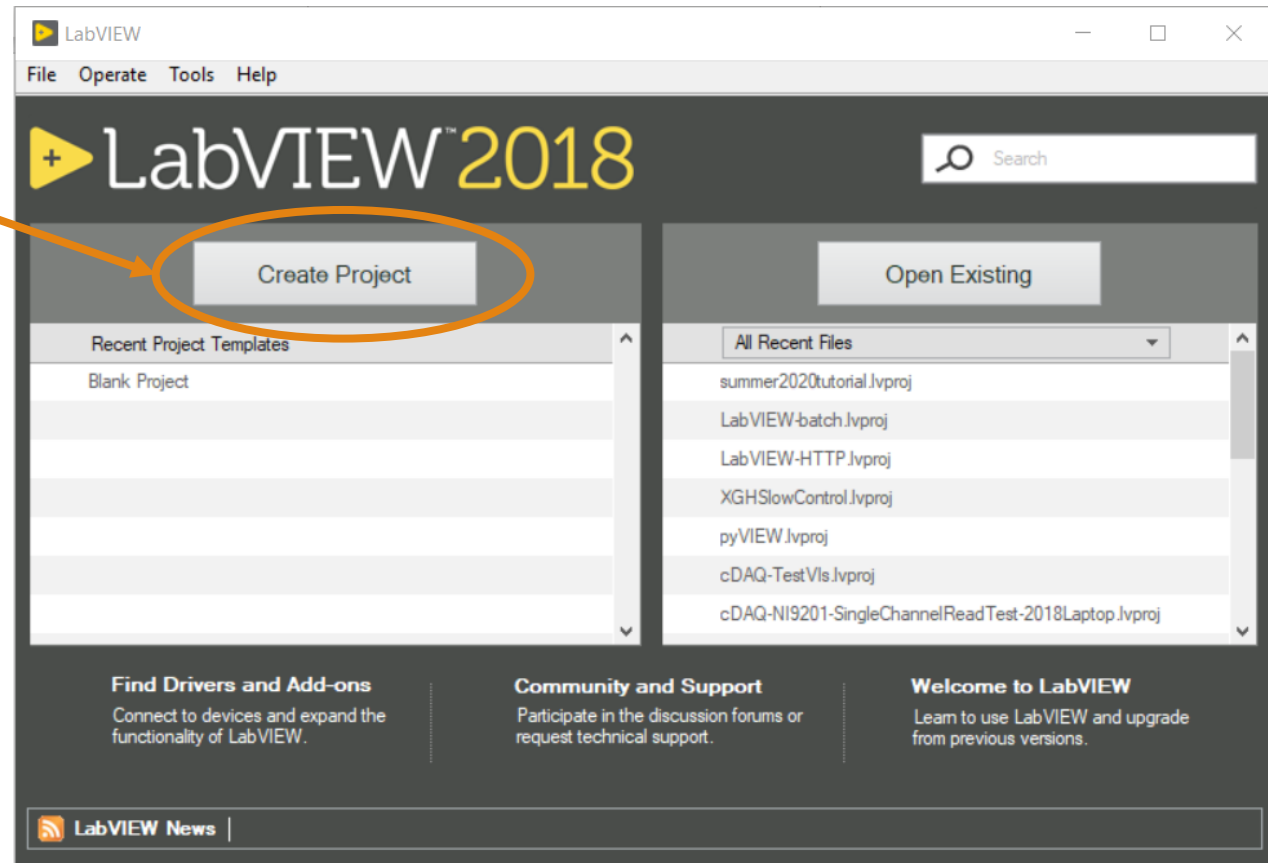
Creating a Project and Your First vi

1. Open LabVIEW



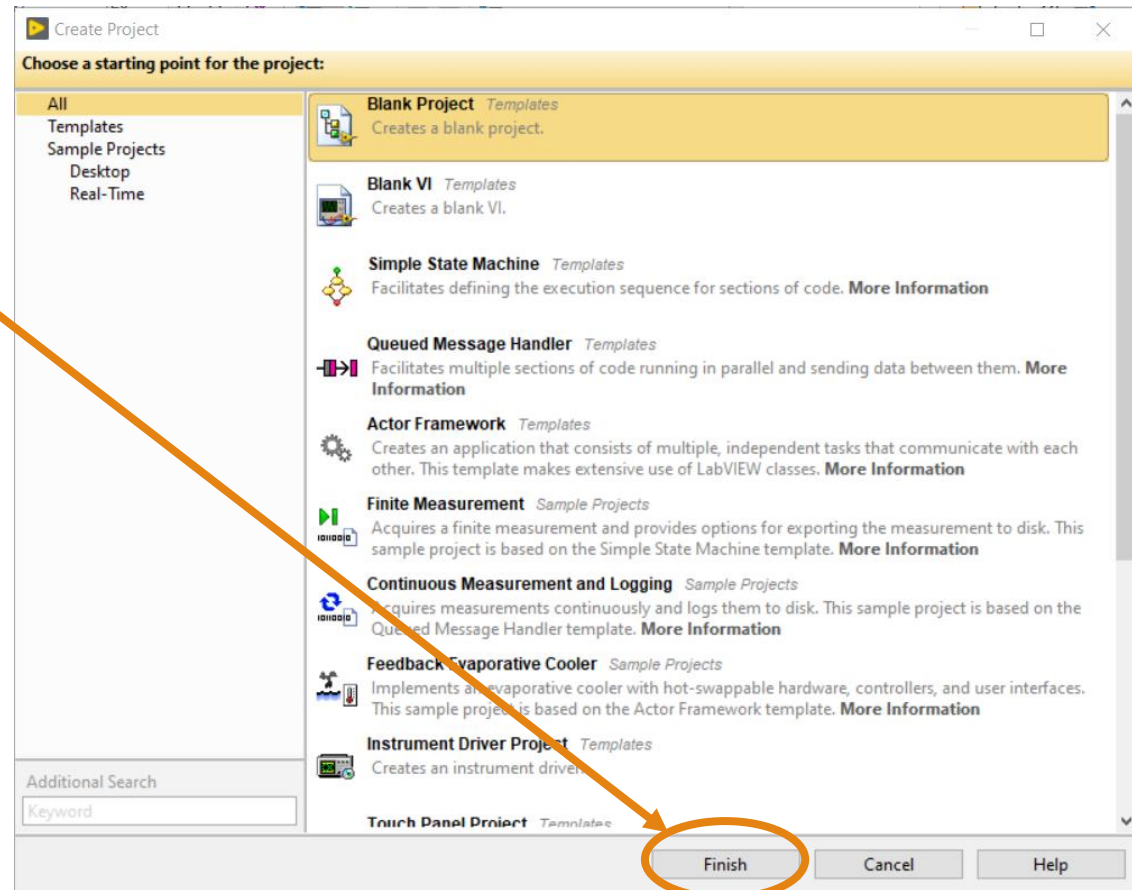
Creating a Project and Your First vi

1. Open LabVIEW
2. Click “Create Project”



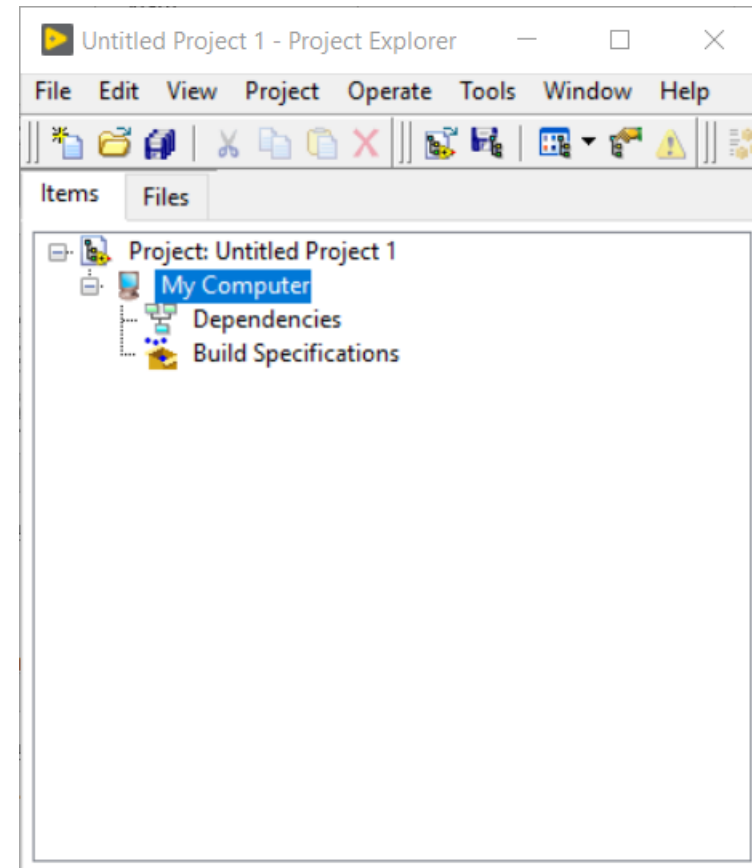
Creating a Project and Your First vi

1. Open LabVIEW
2. Click “Create Project”
3. Select “Blank Project” and hit “Finish”



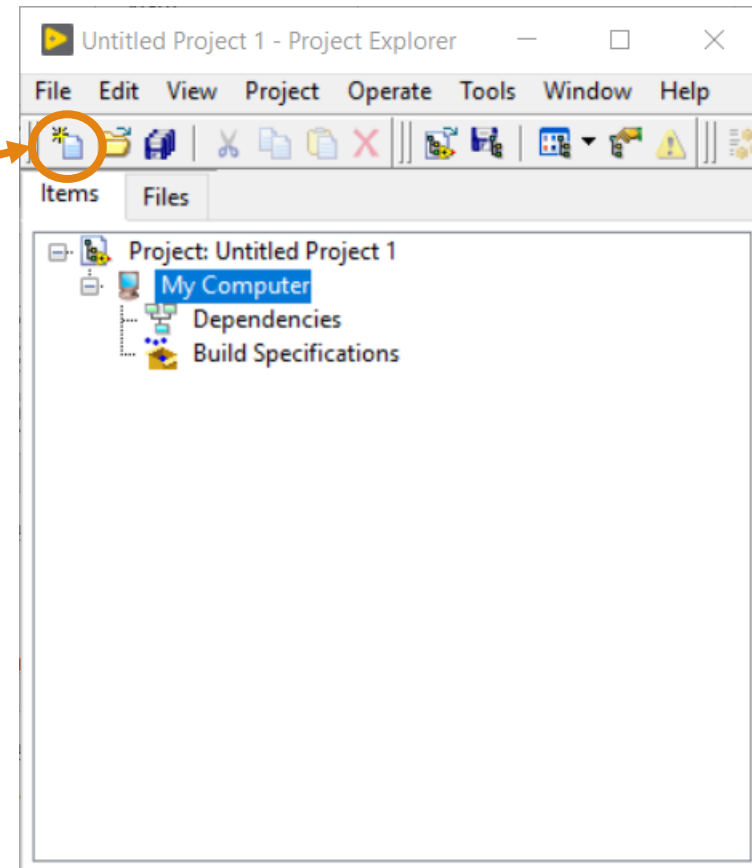
Creating a Project and Your First vi

1. Open LabVIEW
2. Click “Create Project”
3. Select “Blank Project” and hit “finish”
4. A blank project will open



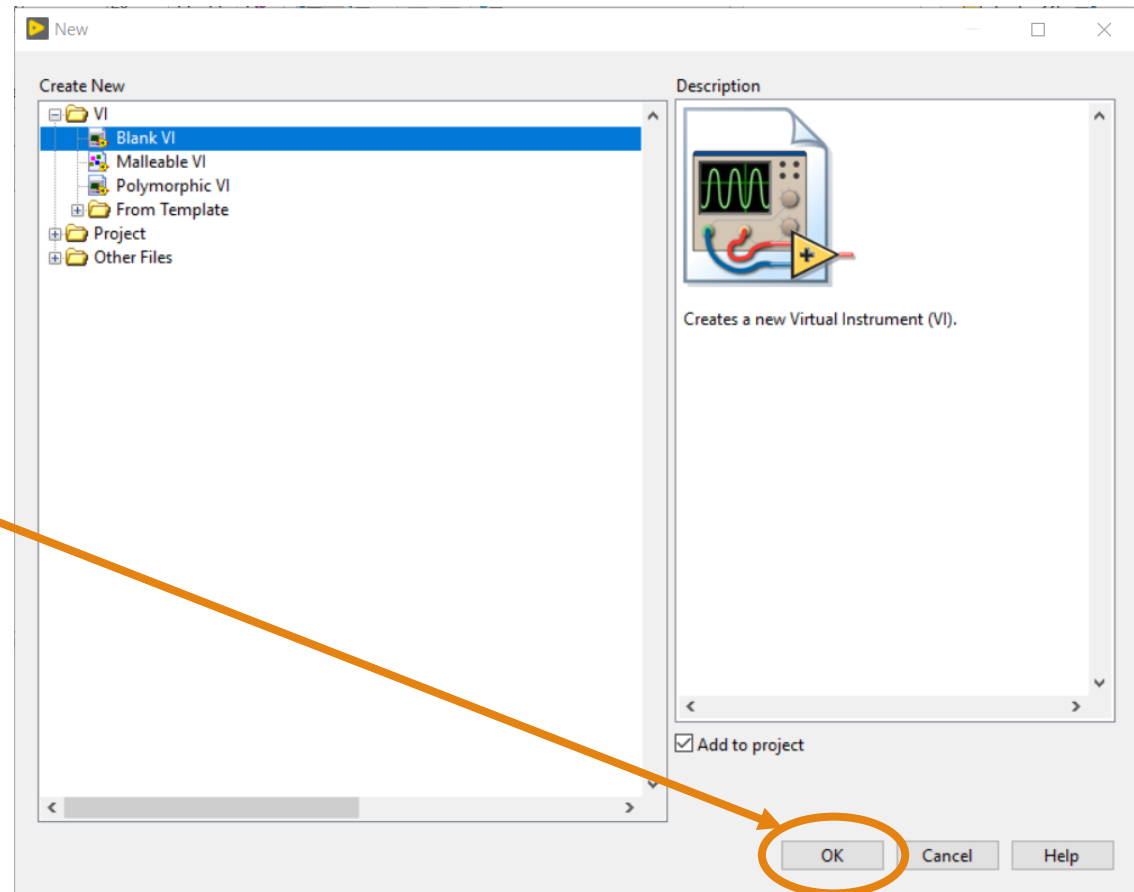
Creating a Project and Your First vi

1. Open LabVIEW
2. Click “Create Project”
3. Select “Blank Project” and hit “finish”
4. A blank project will open
5. Hit “create new” button: 



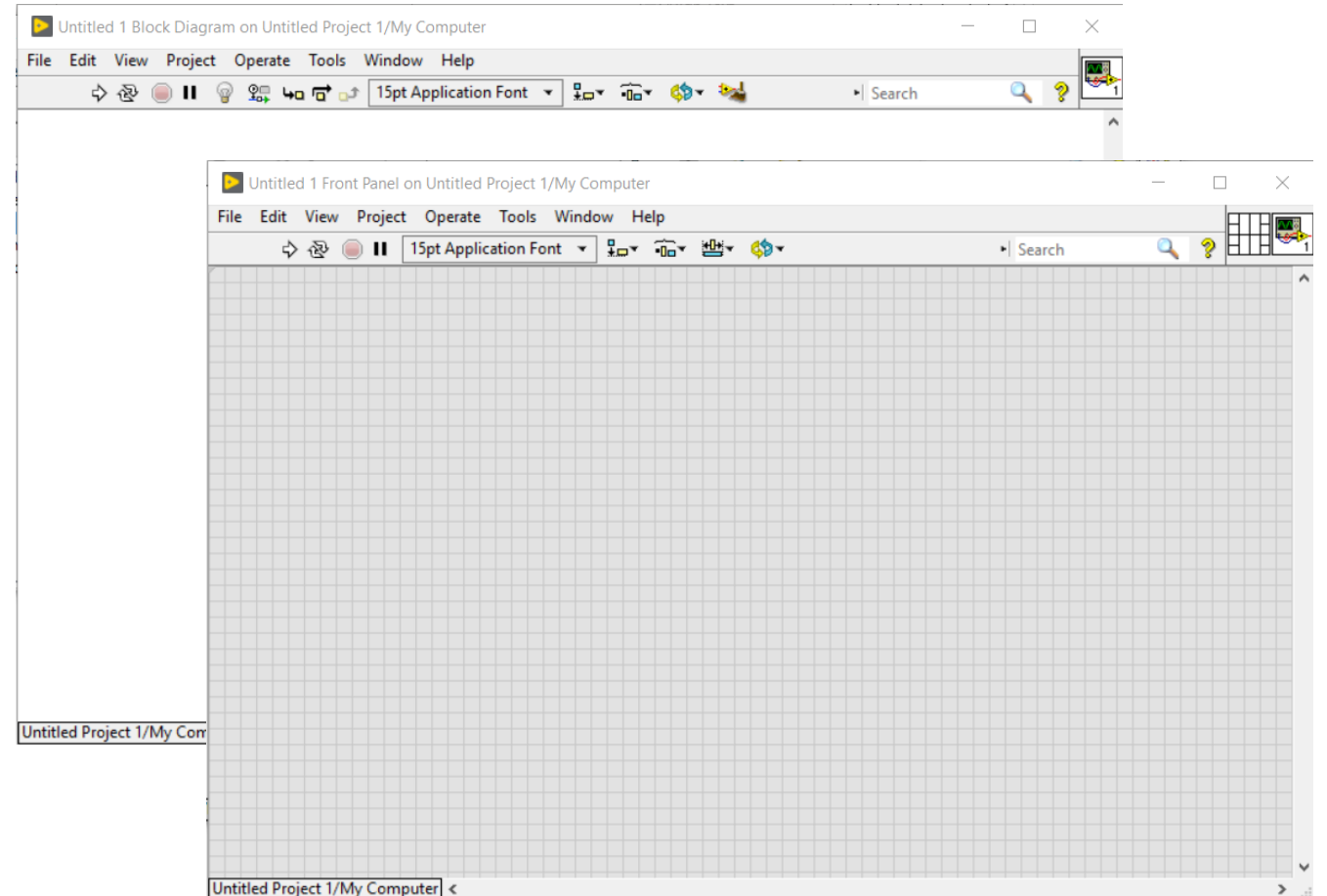
Creating a Project and Your First vi

1. Open LabVIEW
2. Click “Create Project”
3. Select “Blank Project” and hit “finish”
4. A blank project will open
5. Hit “create new” button:
6. Select “Blank VI” and hit “OK”




Creating a Project and Your First vi

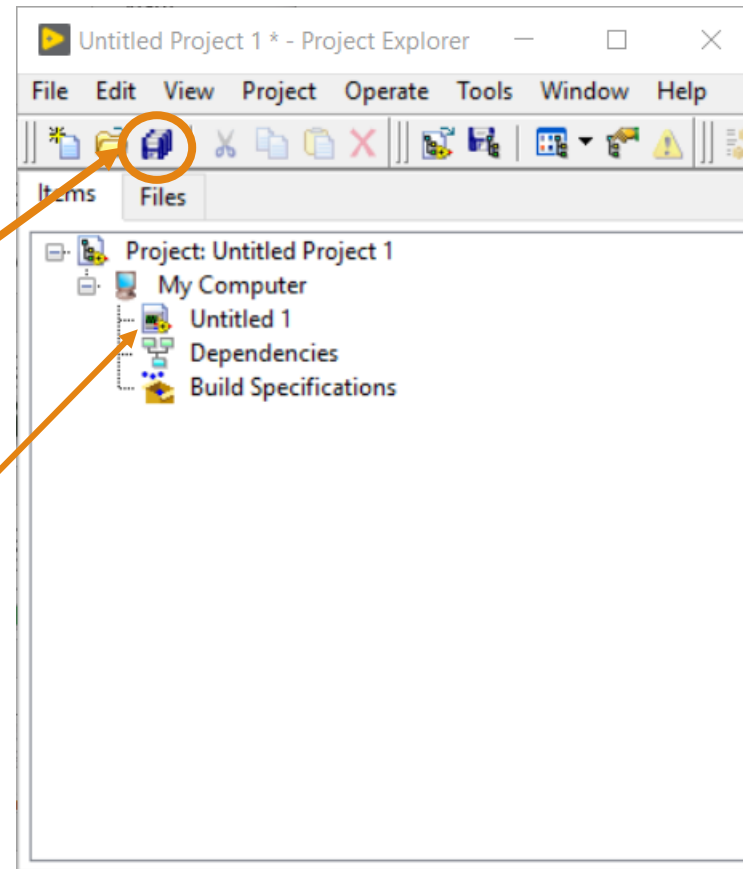
1. Open LabVIEW
2. Click “Create Project”
3. Select “Blank Project” and hit “finish”
4. A blank project will open
5. Hit “create new” button:
6. Select “blank vi” and hit “OK”
7. Your first vi is now open!



Creating a Project and Your First vi

1. Open LabVIEW
2. Click “Create Project”
3. Select “Blank Project” and hit “finish”
4. A blank project will open
5. Hit “create new” button:
6. Select “blank vi” and hit “OK”
7. Your first vi is now open!
8. Save the project and give it a name: 
 - This will prompt you to save the blank vi also

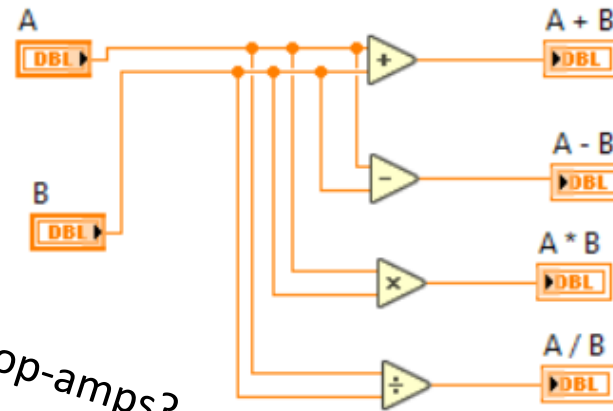
Note that the new vi is in the tree



Intro to Visual Programming

What are those lines????

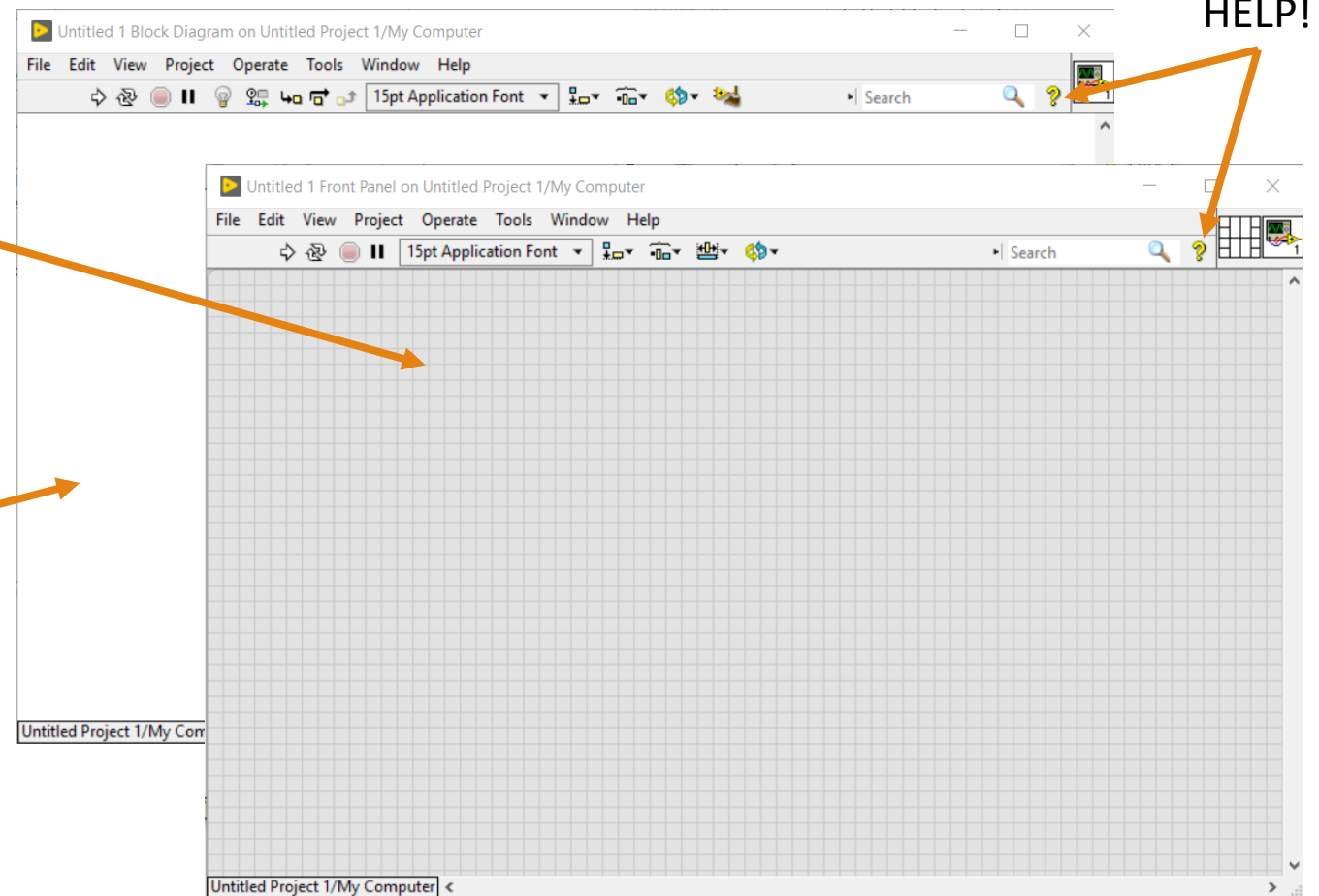
Are those triangles op-amps?



I'm so confused!

Front Panel and Block Diagram

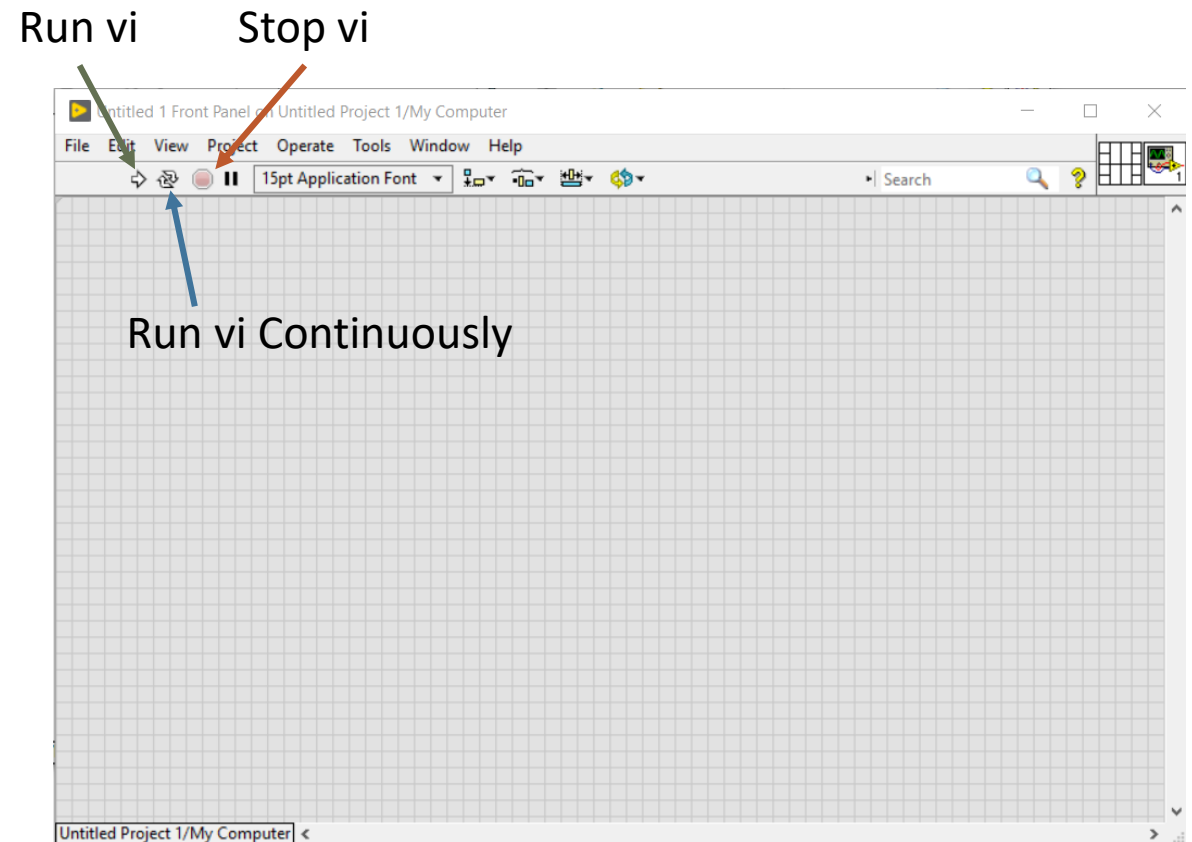
- LabVIEW has two screens used for programming
 - Front Panel
 - This is the “GUI”
 - Change inputs while the vi runs
 - View outputs “live” including graphs
 - Block Diagram
 - This is the “script”
 - Write the operations for the vi
 - Do not generally use while vi runs



The Front Panel

Three buttons for controlling execution of vi

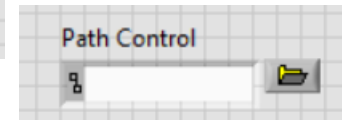
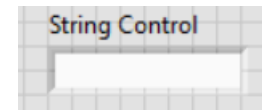
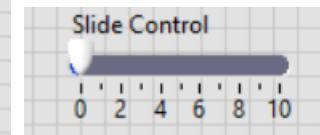
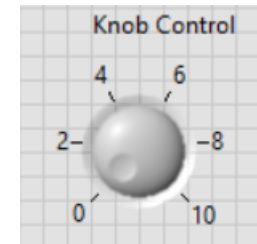
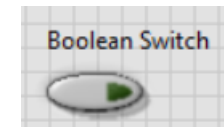
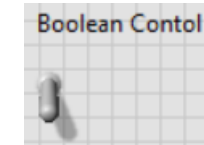
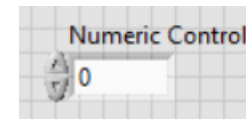
- Run vi
 - Executes the vi once
 - Stops when end is reached
- Stop vi
 - Ends execution of vi
 - Can use to bail out of unforeseen problems
- Run vi Continuously
 - Runs vi as if entire vi is in a while loop
 - Stop execution with the “Stop vi” button



The Front Panel

- **Three main types of objects**

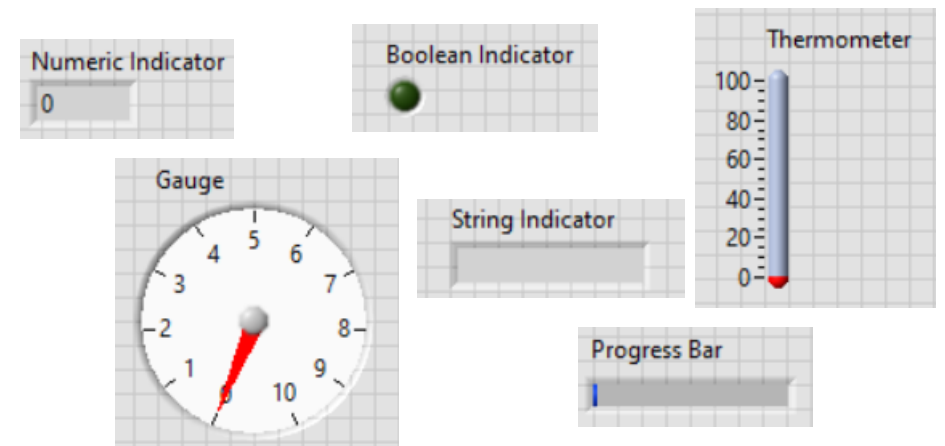
- Controls
 - Inputs for vi – can change while vi is running
 - Can have many data types (numeric, boolean, ...)
- Indicators
 - Outputs of the vi – updated “live” as vi runs
 - Can have many data types (numeric, boolean, ...)
- Graphs
 - Displays output data on a graph
 - Accepts many data formats
 - Highly configurable for readability



The Front Panel

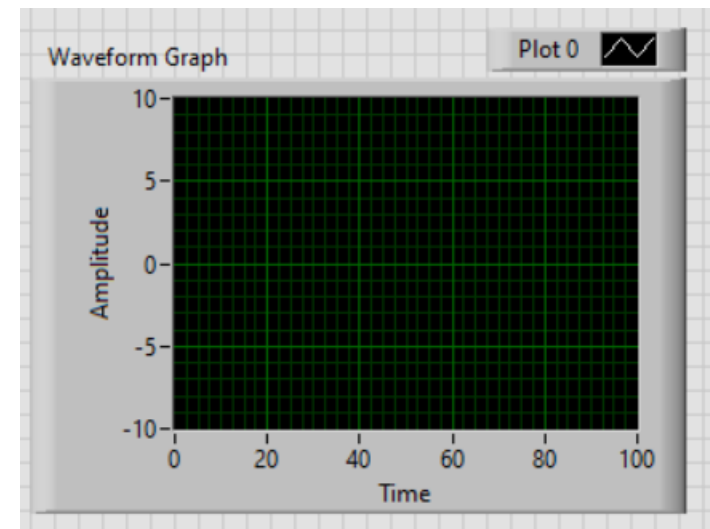
- **Three main types of objects**

- Controls
 - Inputs for vi – can change while vi is running
 - Can have many data types (numeric, boolean, ...)
- Indicators
 - Outputs of the vi – updated “live” as vi runs
 - Can have many data types (numeric, boolean, ...)
- Graphs
 - Displays output data on a graph
 - Accepts many data formats
 - Highly configurable for readability



The Front Panel

- **Three main types of objects**
 - Controls
 - Inputs for vi – can change while vi is running
 - Can have many data types (numeric, boolean, ...)
 - Indicators
 - Outputs of the vi – updated “live” as vi runs
 - Can have many data types (numeric, boolean, ...)
 - Graphs
 - Displays output data on a graph
 - Accepts many data formats
 - Highly configurable for readability



GUI Example

The image shows a LabVIEW front panel for a Xenon control system. The interface is divided into several functional areas:

- Solenoid Valve Control:** A vertical column of seven indicator lights labeled PV-1 through PV-4 and PXV-4. PV-1 and PV-4 are currently illuminated green.
- Deploy/Recover Operations:** Contains buttons for "Deploy Xenon" (green), "Recover Xenon" (red), and "GO!". It also features two bottle icons labeled "Bottle 1" and "Bottle 2".
- Bottle Transfer Operations:** Shows two bottle icons with arrows indicating transfer directions: "Bottle 1" to "Bottle 2" and "Bottle 2" to "Bottle 1". It includes a "Bypass Purifier" icon and a "Through Purifier" icon, along with a "GO!" button and a green indicator light.
- Control Parameters:** A panel with four numeric input fields: "Solenoid Valve Switching Time" (1 sec), "Omega Gauge Bottom Pressure" (0.1 psig), "Wait Time for XP-3 Pumpout" (2.5 sec), and "Xe Bottle Low Pressure Threshold" (145 psig). An "Update Values" button is at the bottom.
- Pressure Plot:** A graph with "Pressure (psig)" on the y-axis (ranging from -1 to 1) and "Time (s)" on the x-axis (ranging from 0 to 30). It includes a legend with four entries: "Xe Bottle 1" (green square), "Xe Bottle 2" (red square), "Recovery Line" (green line), and "Deploy Line" (red line).
- Data & Plot Parameters:** A panel with two sections: "Write Parameters" (To Buffer: 10 Hz, To Database: 1 Hz, Buffer Size: 300 sec) and "Read Parameters" (Plot Rate: 5 Hz, Plot Last: 30 sec). An "Update Values" button is at the bottom.

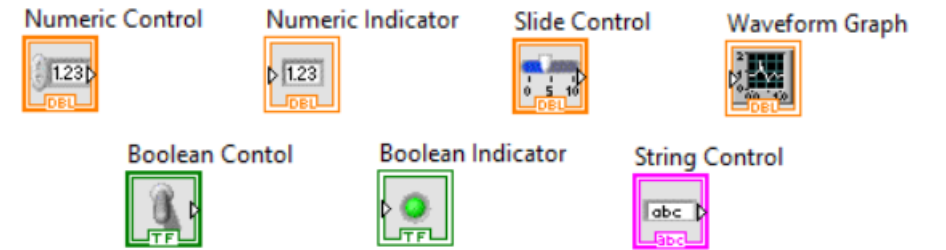
Annotations with orange arrows point to specific UI elements:

- Boolean Selectors:** Points to the "GO!" buttons in the Deploy/Recover and Bottle Transfer sections.
- Tabs:** Points to the "Normal Operations" and "Pump Down" tabs above the Pressure Plot.
- Graph:** Points to the Pressure Plot area.
- On/Off Buttons:** Points to the "GO!" button in the Bottle Transfer section.
- Numeric Controls:** Points to the numeric input fields in the Control Parameters section.
- Boolean Indicators:** Points to the green indicator light in the Bottle Transfer section.
- Toggle Buttons w/ Indicators:** Points to the PV-1 through PV-4 indicator lights.
- Numeric Controls:** Points to the numeric input fields in the Data & Plot Parameters section.

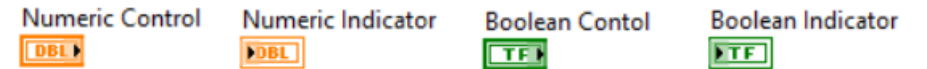
The Block Diagram

- **Five basic types of objects**
 - Controls, Indicators + Graphs
 - Linked to their front panel counterparts
 - Can have many data types (numeric, boolean, ...)
 - Constants
 - These are “hard coded” variables in the vi
 - Data types shown as different colors
 - Local Variables
 - Are created from controls or indicators
 - Used to read/write variables during code operations
 - Structures
 - Boxes that contain a section of code
 - Types: loops, cases, events, sequences
 - Operators
 - Arithmetic, logic, comparison

Icons



Basic



The Block Diagram

- **Five basic types of objects**
 - Controls, Indicators + Graphs
 - Linked to their front panel counterparts
 - Can have many data types (numeric, boolean, ...)
 - Constants
 - These are “hard coded” variables in the vi
 - Data types shown as different colors
 - Local Variables
 - Are created from controls or indicators
 - Used to read/write variables during code operations
 - Structures
 - Boxes that contain a section of code
 - Types: loops, cases, events, sequences
 - Operators
 - Arithmetic, logic, comparison



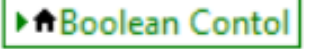
The Block Diagram

- **Five basic types of objects**
 - Controls, Indicators + Graphs
 - Linked to their front panel counterparts
 - Can have many data types (numeric, boolean, ...)
 - Constants
 - These are “hard coded” variables in the vi
 - Data types shown as different colors
 - Local Variables
 - Are created from controls or indicators
 - Used to read/write variables during code operations
 - Structures
 - Boxes that contain a section of code
 - Types: loops, cases, events, sequences
 - Operators
 - Arithmetic, logic, comparison

Read: 



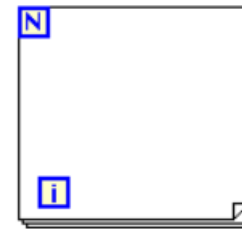
Write: 



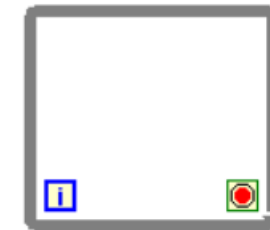
The Block Diagram

- **Five basic types of objects**
 - Controls, Indicators + Graphs
 - Linked to their front panel counterparts
 - Can have many data types (numeric, boolean, ...)
 - Constants
 - These are “hard coded” variables in the vi
 - Data types shown as different colors
 - Local Variables
 - Are created from controls or indicators
 - Used to read/write variables during code operations
 - Structures
 - Boxes that contain a section of code
 - Types: loops, cases, events, sequences
 - Operators
 - Arithmetic, logic, comparison

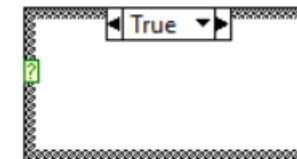
For Loop



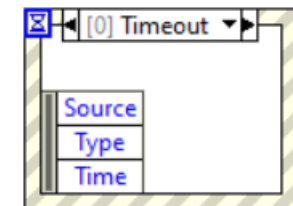
While Loop



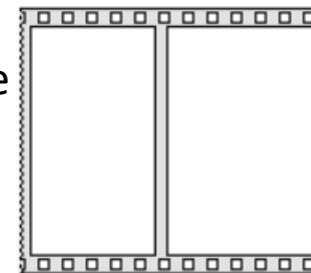
Case Structure



Event Structure



Sequence



The Block Diagram

- **Five basic types of objects**
 - Controls, Indicators + Graphs
 - Linked to their front panel counterparts
 - Can have many data types (numeric, boolean, ...)
 - Constants
 - These are “hard coded” variables in the vi
 - Data types shown as different colors
 - Local Variables
 - Are created from controls or indicators
 - Used to read/write variables during code operations
 - Structures
 - Boxes that contain a section of code
 - Types: loops, cases, events, sequences
 - Operators
 - Arithmetic, logic, comparison

Numerical Operators



Logical Operators



Comparisons



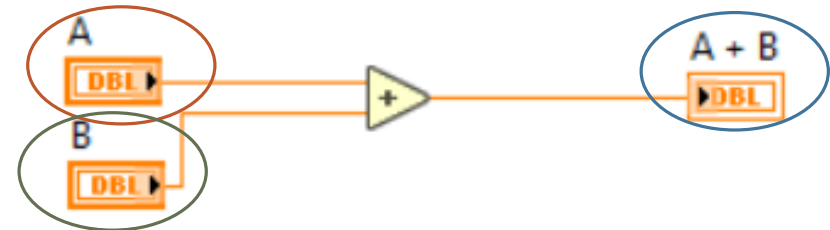
The Block Diagram

- Wires connect the objects in the block diagram
- LabVIEW prefers the code to go left-to-right
- Wires give ordering of operations
 - The data “flows” through the wire
- The example takes two numbers set by the user (“A” and “B”) and adds them, displaying the answer as “A+B” on the front panel

Front Panel:



Block Diagram:



Live Demos 1 and 2

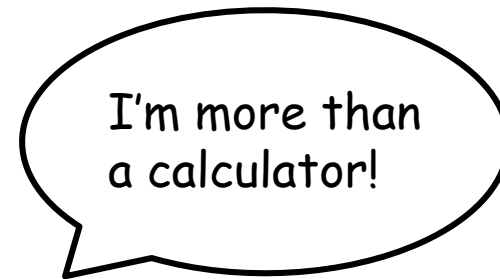
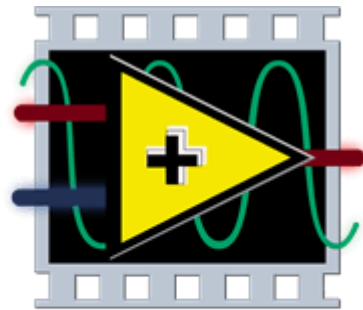
Demo 1

Write a vi to do all four basic math operations on two numbers that are controllable via the front panel. Display the results on the front panel.

Demo 2

Write a vi to take two Boolean inputs and do the three main logical operations: AND, OR, XOR. Display the outputs of these operations as LED's. Use a case structure to print a response string to one of the operations. Use a case structure to print the states of the Booleans as a string.

Expanding the Toolbox



LabVIEW

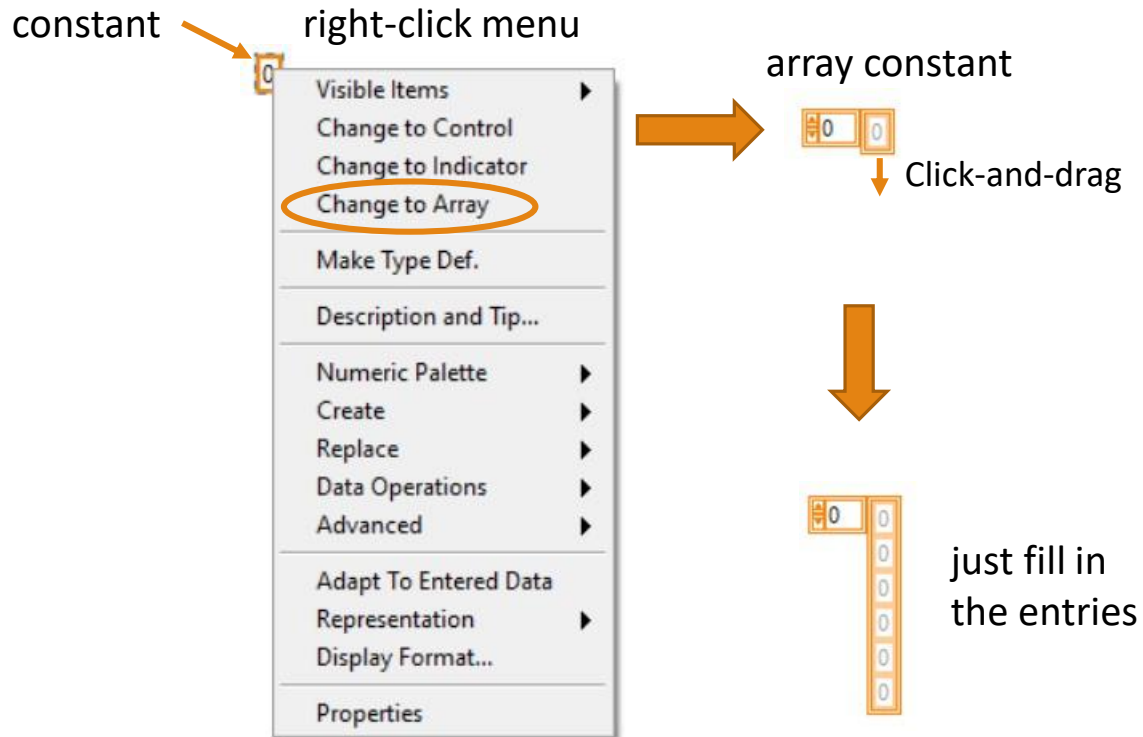
Expanding the Toolbox -- Data Types

- **Numeric** – integers, floats etc..
 - LabVIEW is generally good at automatic representation matching
- **Boolean** – True/False
 - Booleans can be converted to/from integers easily if needed
- **Strings** – Can be anything, but is usually text
- **Path** – Special format used for directory paths
- **Array** – Stores multiple values in one object
 - Most other data types can be made into arrays
 - Can be made in N dimensions
- **Waveform** – Array with built in uniform time ordering
 - Requires t0 (initial time) and dt (time between data points)
- **Cluster** – Similar to a dictionary-type object
 - Uses labels to point at objects in cluster
 - Can mix and match data types

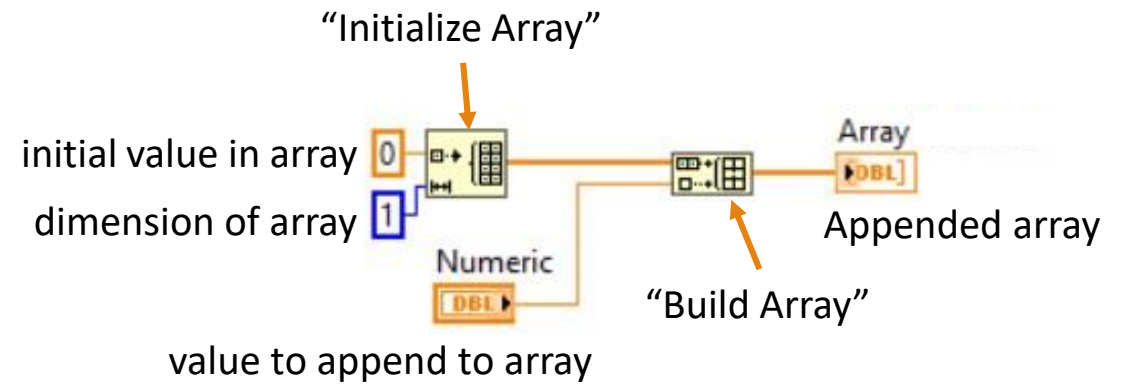


Expanding the Toolbox -- Arrays

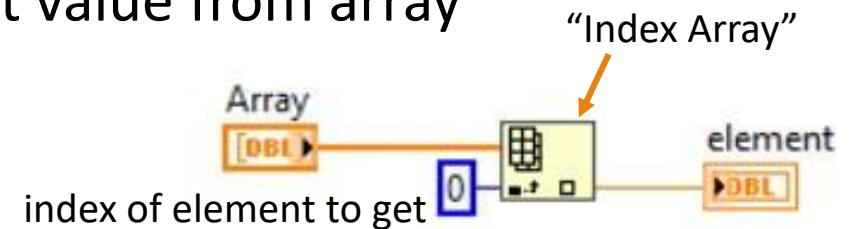
Creating an array constant



Initialize and append an array



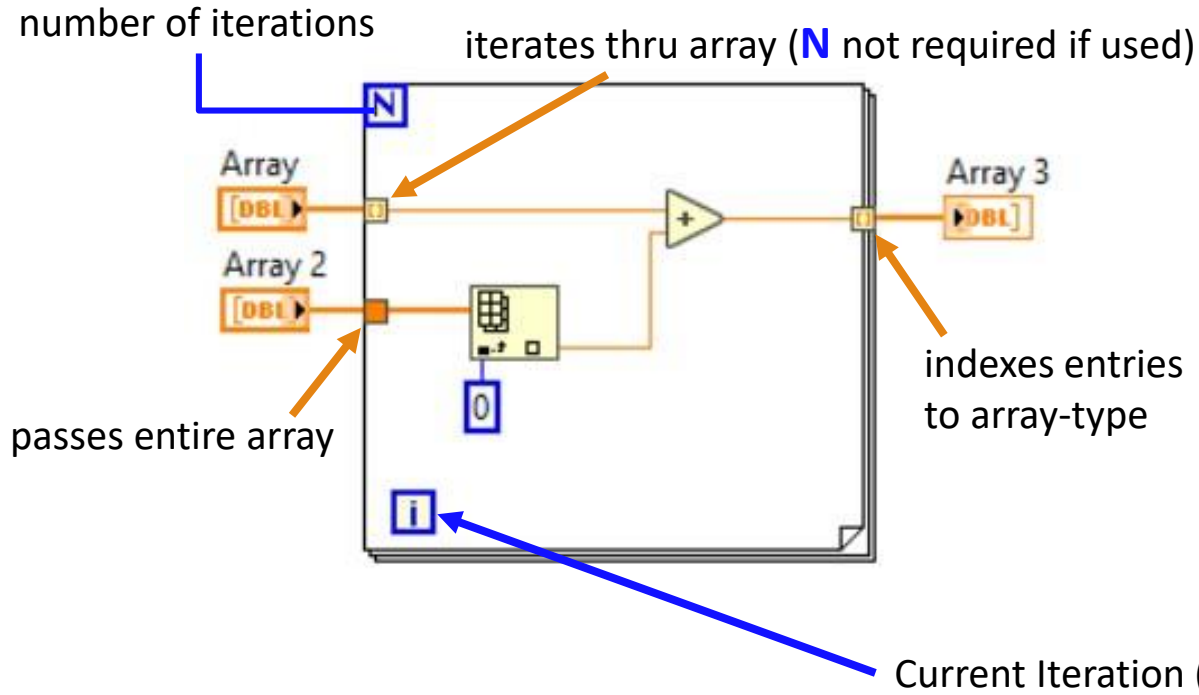
Get value from array



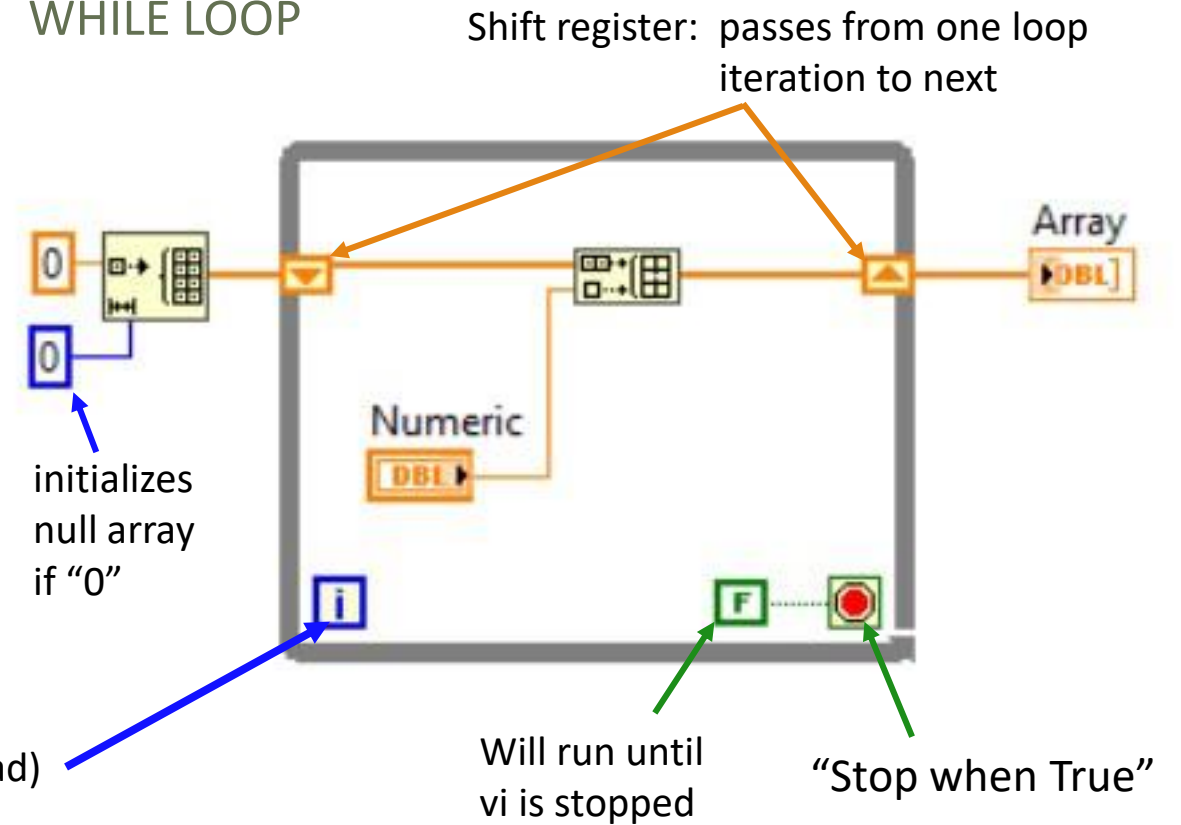
Note: Thick wires indicate an array-type object

Expanding the Toolbox – Loops

FOR LOOP

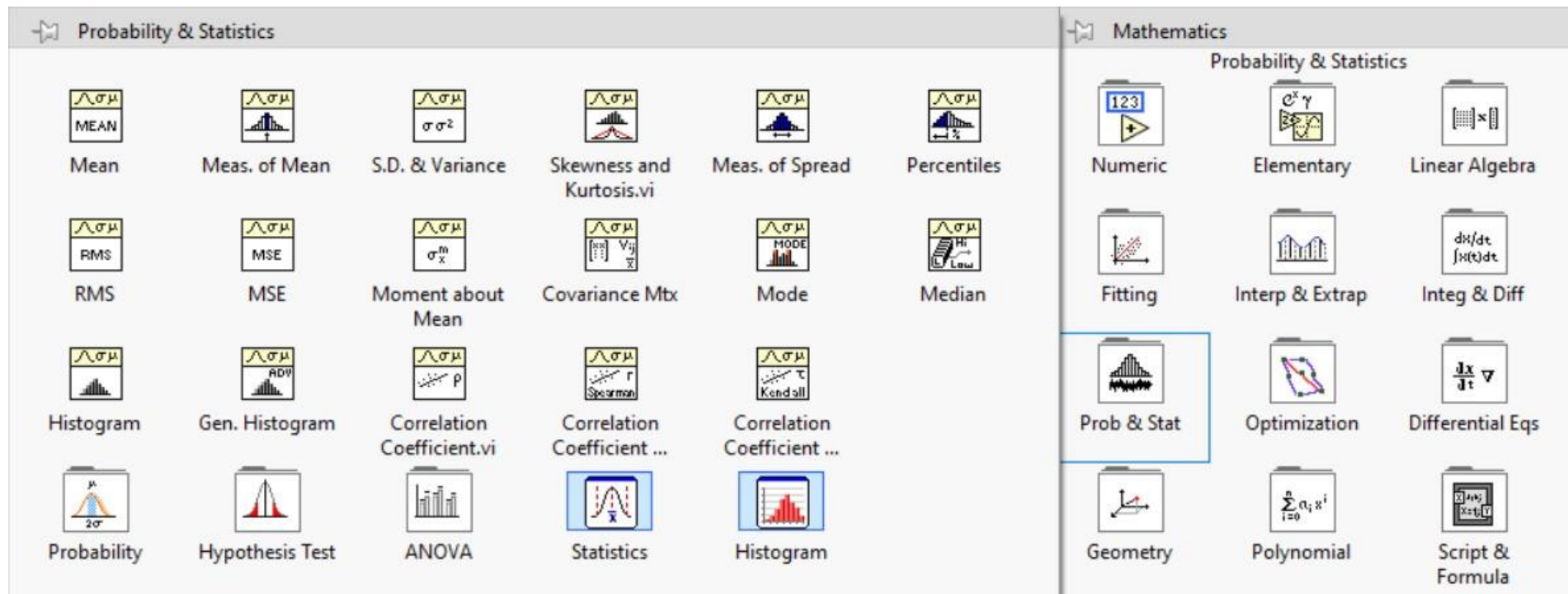


WHILE LOOP



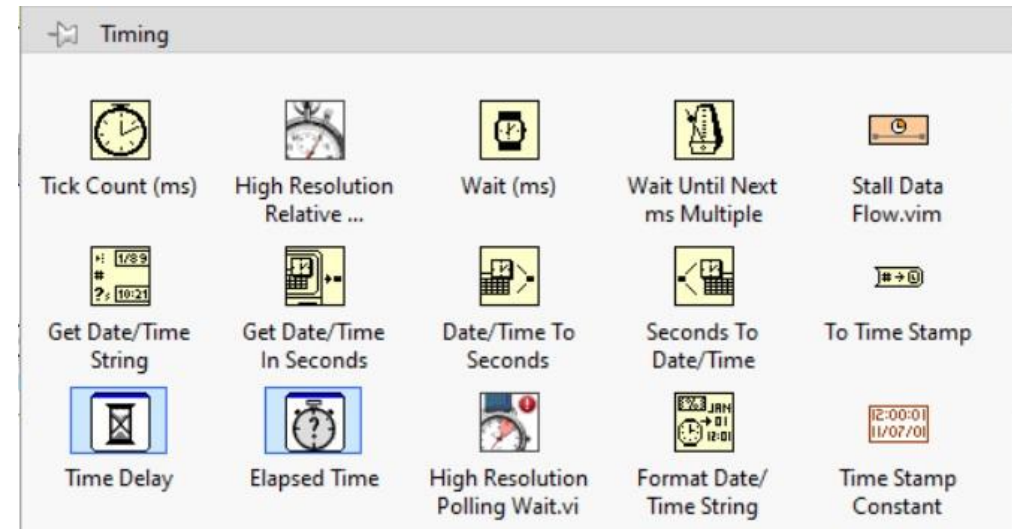
Expanding the Toolbox -- Statistics

LabVIEW has a comprehensive statistics suite, you can do almost anything



Expanding the Toolbox – Timing Palette

- Most used objects
 - Wait (ms)
 - Waits for the wired number of ms
 - Useful for setting rough loop execution rate
 - Wait Until Next ms Multiple
 - Waits until current time (ms) is multiple of wired number
 - Useful for setting more precise loop execution rate
 - First iteration may be shorter
 - High Resolution Relative Seconds
 - Returns the current time since epoch in seconds as a DBL
 - Useful for making a timer
 - Get Date/Time in Seconds
 - Returns current time as a timestamp-type object
 - Format Date/Time String
 - Turns Timestamp object into a string with highly flexible formatting
 - Useful for recording timestamps in a file

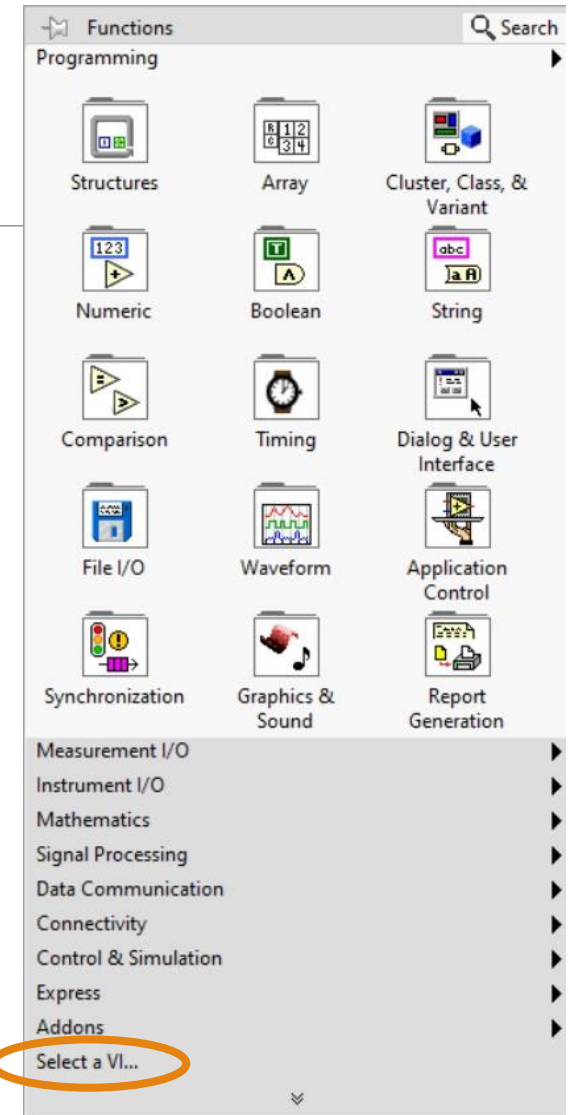


Expanding the Toolbox -- sub-vi's

- A program within a program, analogous to functions in other languages
 - Have inputs and outputs with pre-assigned and labeled nodes
 - Differ from “primitives”, which are all “LabVIEW Straw” color
 - Primitives are non-editable, you can open and edit sub-vis
- We have seen many primitives and “built-in” sub-vi's already



- You can write your own, or get custom vi's from colleagues or forums
 - Helpful for de-cluttering the block diagram
 - Can write a sub-vi once and use it all over your project and beyond
 - Easy to import if you know the file location
 - Be Organized!
 - NI has it's own moderated forum of vi's written by users

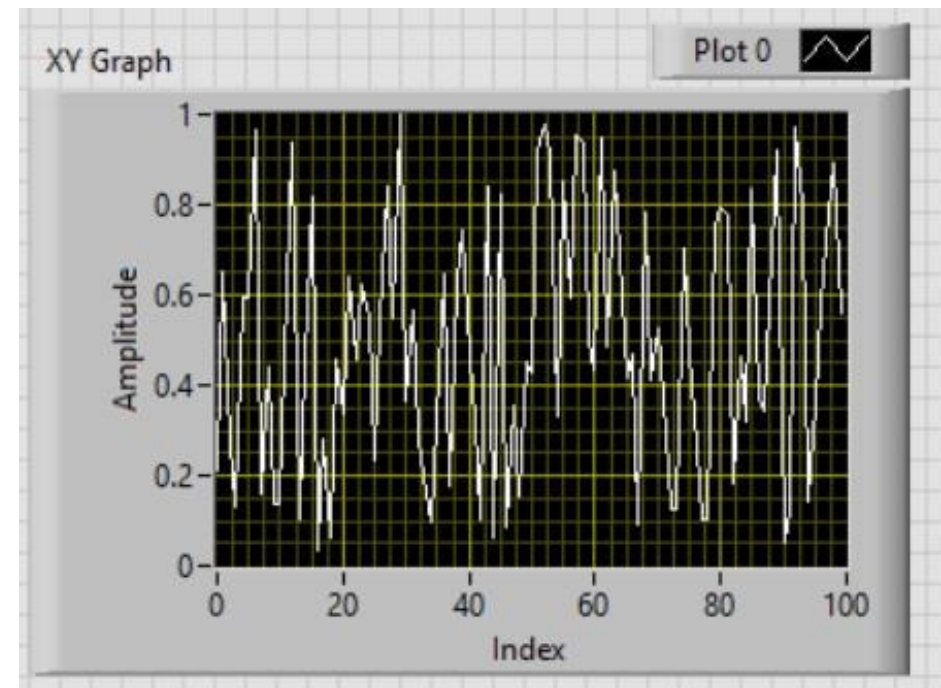


Expanding the Toolbox – Graphs & Charts

- Graphs and charts are the main ways to display data
 - Graph: Plots only the data it is passed when called
 - Chart: Stores “history” and updates the plot with passed values
- Both created from the front panel
- Main types
 - XY Graph
 - Plots data with x and y coordinates
 - Data input as an array of (x, y) clusters
 - Waveform Graph/Chart
 - Plots 1D waveforms or arrays
 - Assumes time for x-axis of plot
 - Intensity Graph/Chart
 - Plots 2D arrays of data as function of index
 - Can scale the x and y axes to reflect actual values
- More information in the “Example vi” slides at the end

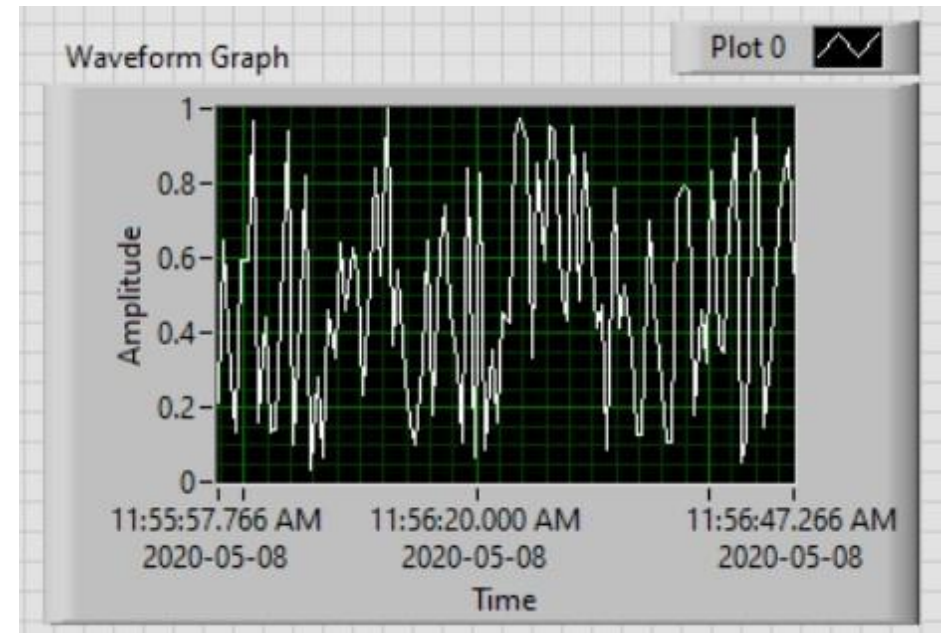
Expanding the Toolbox – Graphs & Charts

- Graphs and charts are the main ways to display data
 - Graph: Plots only the data it is passed when called
 - Chart: Stores “history” and updates the plot with passed values
- Both created from the front panel
- Main types
 - XY Graph
 - Plots data with x and y coordinates
 - Data input as an array of (x, y) clusters
 - Waveform Graph/Chart
 - Plots 1D waveforms or arrays
 - Assumes time for x-axis of plot
 - Intensity Graph/Chart
 - Plots 2D arrays of data as function of index
 - Can scale the x and y axes to reflect actual values
- More information in the “Example vi” slides at the end



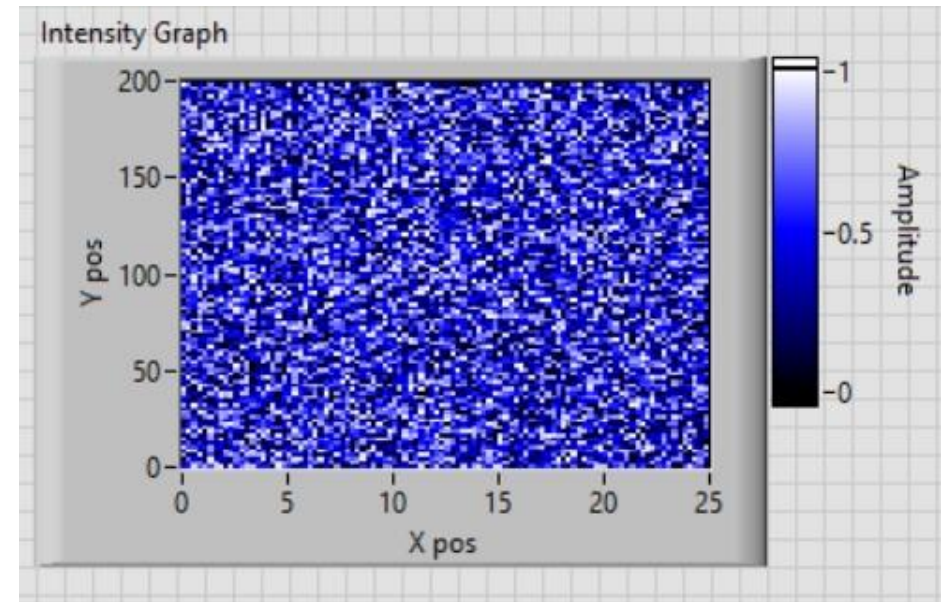
Expanding the Toolbox – Graphs & Charts

- Graphs and charts are the main ways to display data
 - Graph: Plots only the data it is passed when called
 - Chart: Stores “history” and updates the plot with passed values
- Both created from the front panel
- Main types
 - XY Graph
 - Plots data with x and y coordinates
 - Data input as an array of (x, y) clusters
 - Waveform Graph/Chart
 - Plots 1D waveforms or arrays
 - Assumes time for x-axis of plot
 - Intensity Graph/Chart
 - Plots 2D arrays of data as function of index
 - Can scale the x and y axes to reflect actual values
- More information in the “Example vi” slides at the end



Expanding the Toolbox – Graphs & Charts

- Graphs and charts are the main ways to display data
 - Graph: Plots only the data it is passed when called
 - Chart: Stores “history” and updates the plot with passed values
- Both created from the front panel
- Main types
 - XY Graph
 - Plots data with x and y coordinates
 - Data input as an array of (x, y) clusters
 - Waveform Graph/Chart
 - Plots 1D waveforms or arrays
 - Assumes time for x-axis of plot
 - Intensity Graph/Chart
 - Plots 2D arrays of data as function of index
 - Can scale the x and y axes to reflect actual values
- More information in the “Example vi” slides at the end

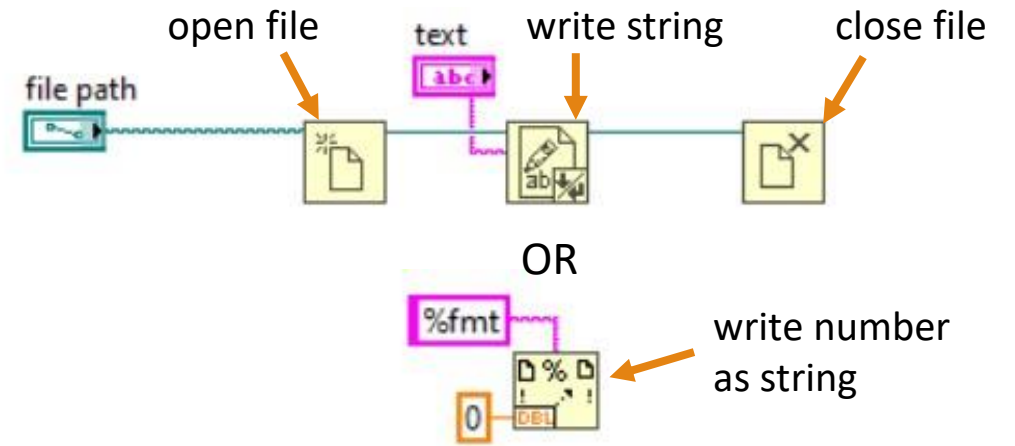


Expanding the Toolbox – Output to File

- LabVIEW has many options for file output
 - Write strings line-by-line
 - Most flexible for formatting output
 - Writing line-by-line can slow down vi
 - Write an array to spreadsheet file
 - Done once, which is faster than N writes
 - Fairly restrictive formatting, but some options
 - Write waveform to spreadsheet file
 - Write only once for whole file
 - Records t0 and dt as well as the timestamp
 - String format is fixed

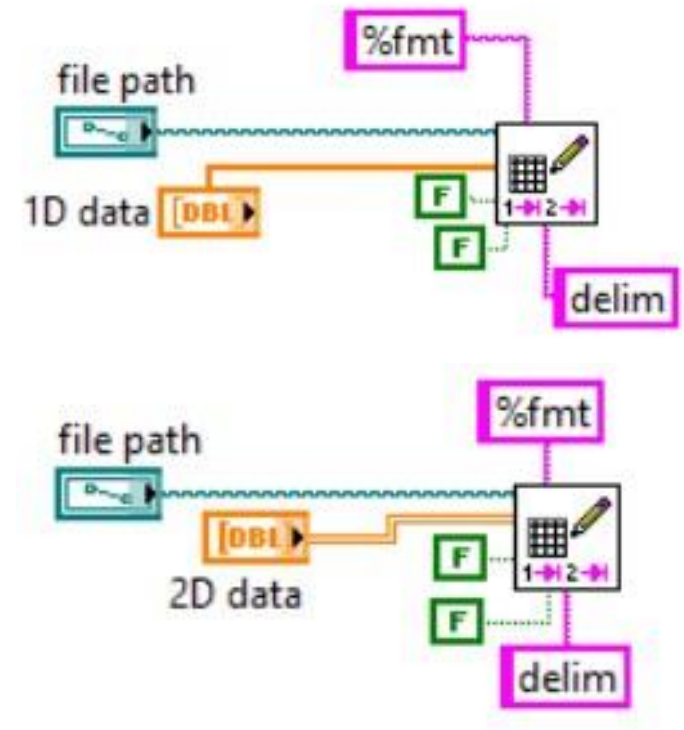
Expanding the Toolbox – Output to File

- LabVIEW has many options for file output
 - Write strings line-by-line
 - Most flexible for formatting output
 - Writing line-by-line can slow down vi
 - Write an array to spreadsheet file
 - Done once, which is faster than N writes
 - Fairly restrictive formatting, but some options
 - Write waveform to spreadsheet file
 - Write only once for whole file
 - Records t0 and dt as well as the timestamp
 - String format is fixed



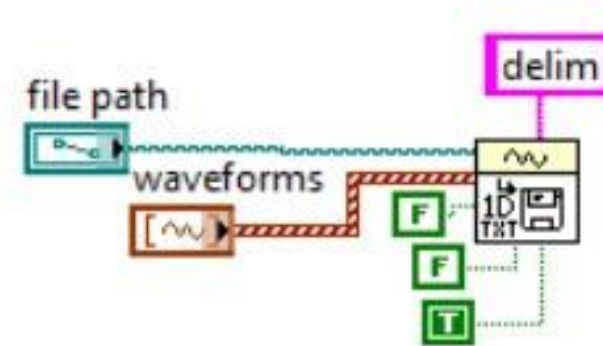
Expanding the Toolbox – Output to File

- LabVIEW has many options for file output
 - Write strings line-by-line
 - Most flexible for formatting output
 - Writing line-by-line can slow down vi
 - Write an array to spreadsheet file
 - Done once, which is faster than N writes
 - Fairly restrictive formatting, but some options
 - Write waveform to spreadsheet file
 - Write only once for whole file
 - Records t0 and dt as well as the timestamp
 - String format is fixed



Expanding the Toolbox – Output to File

- LabVIEW has many options for file output
 - Write strings line-by-line
 - Most flexible for formatting output
 - Writing line-by-line can slow down vi
 - Write an array to spreadsheet file
 - Done once, which is faster than N writes
 - Fairly restrictive formatting, but some options
 - Write waveform to spreadsheet file
 - Write only once for whole file
 - Records t0 and dt as well as the timestamp
 - String format is fixed

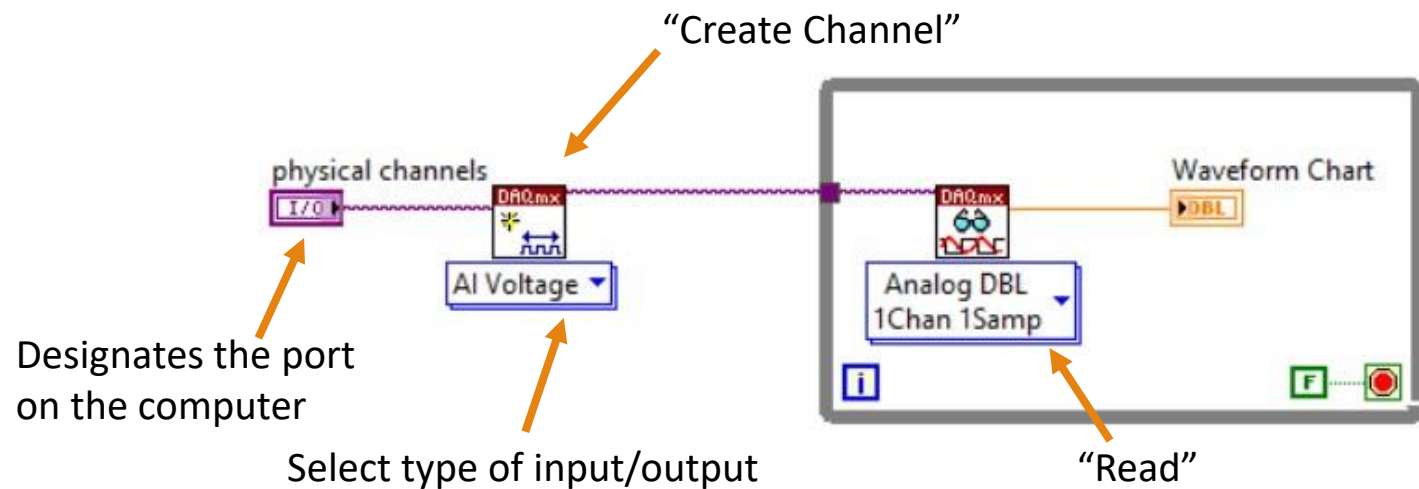


A quick note on DAQmx

DAQmx is the “plug and play” way to communicate with hardware using LabVIEW. The vi’s are found under the “Measurement I/O” panel

- Can read and write both analog and digital data
- Create “tasks” that organize the communication
 - All operations in a task must be the same type (e.g. read analog voltage)
 - Can index the channels in a task by number or name
 - Can run multiple tasks in one vi

This reads from a piece of equipment, and then plots it on a chart.

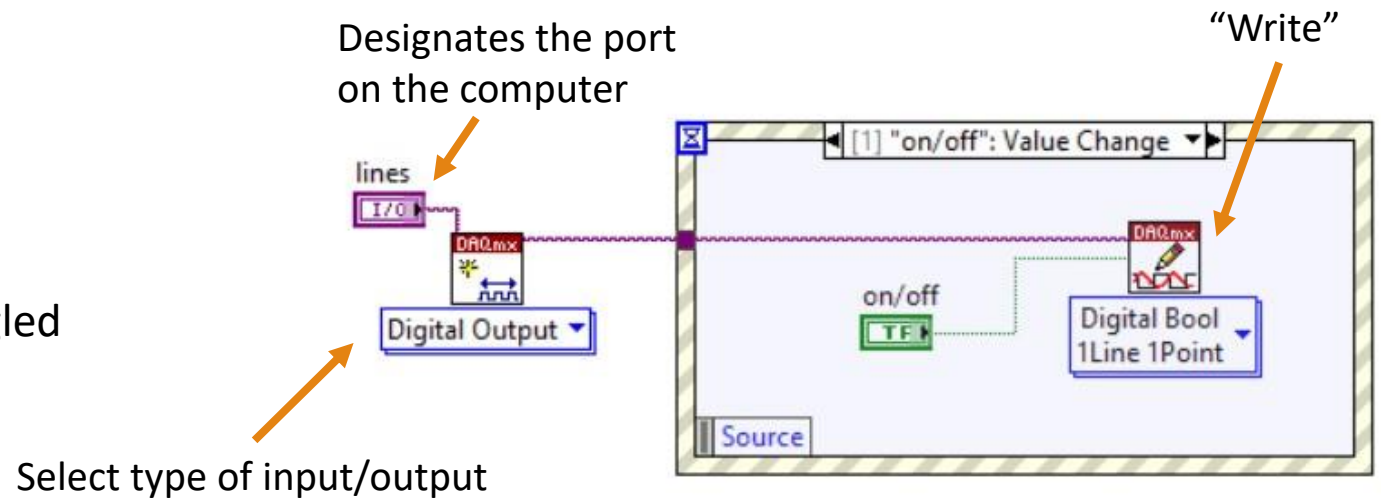


A quick note on DAQmx

DAQmx is the “plug and play” way to communicate with hardware using LabVIEW. The vi’s are found under the “Measurement I/O” panel

- Can read and write both analog and digital data
- Create “tasks” that organize the communication
 - All operations in a task must be the same type (e.g. read analog voltage)
 - Can index the channels in a task by number or name
 - Can run multiple tasks in one vi

This writes a digital signal to a piece of equipment when the Boolean control is toggled



Live Demos 3 and 4

Demo 3

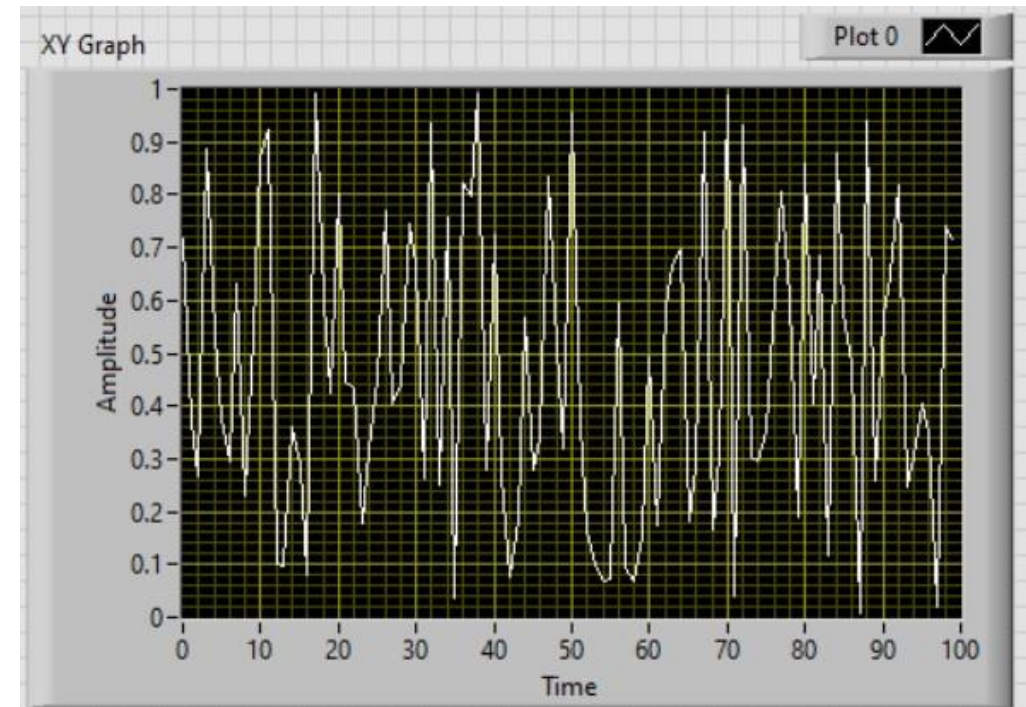
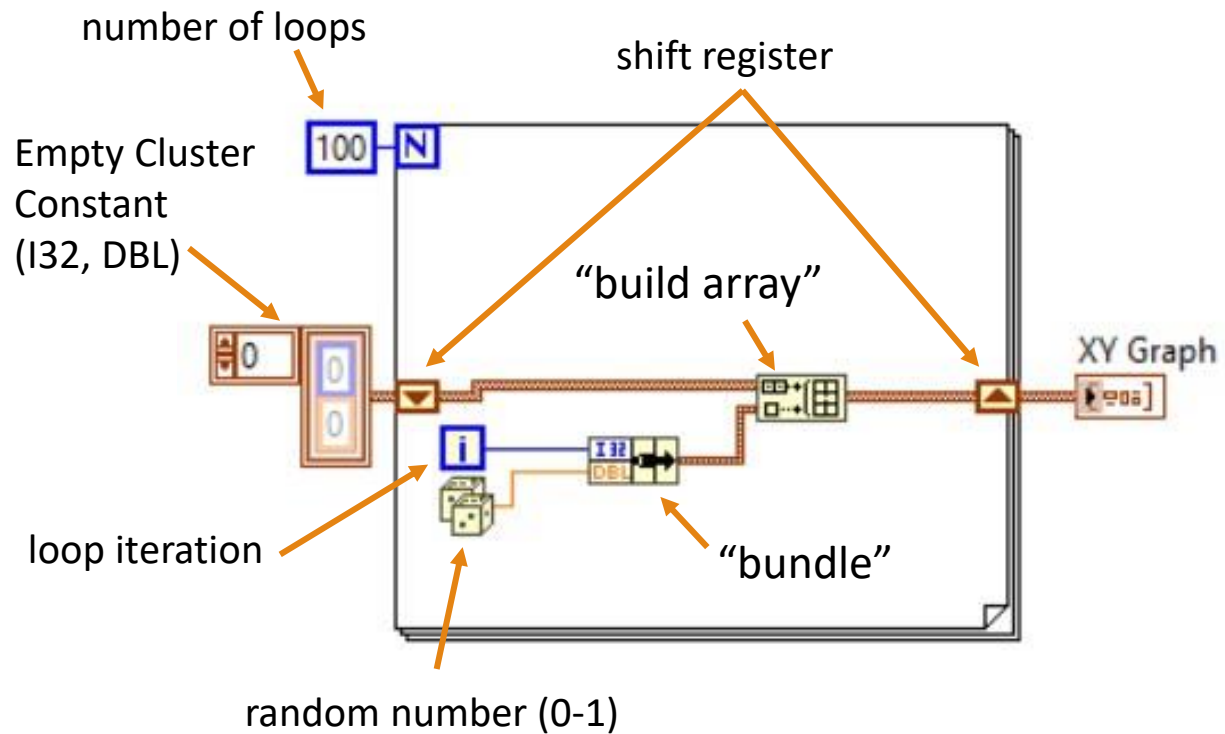
Write a program to take random numbers from a normal distribution and make a histogram of the values. There is no normal distribution generator in LabVIEW, but I have provided a custom sub-vi adapted from Christian Altenbach's example. You will need to have this vi downloaded to your computer from the indico page.

Demo 4

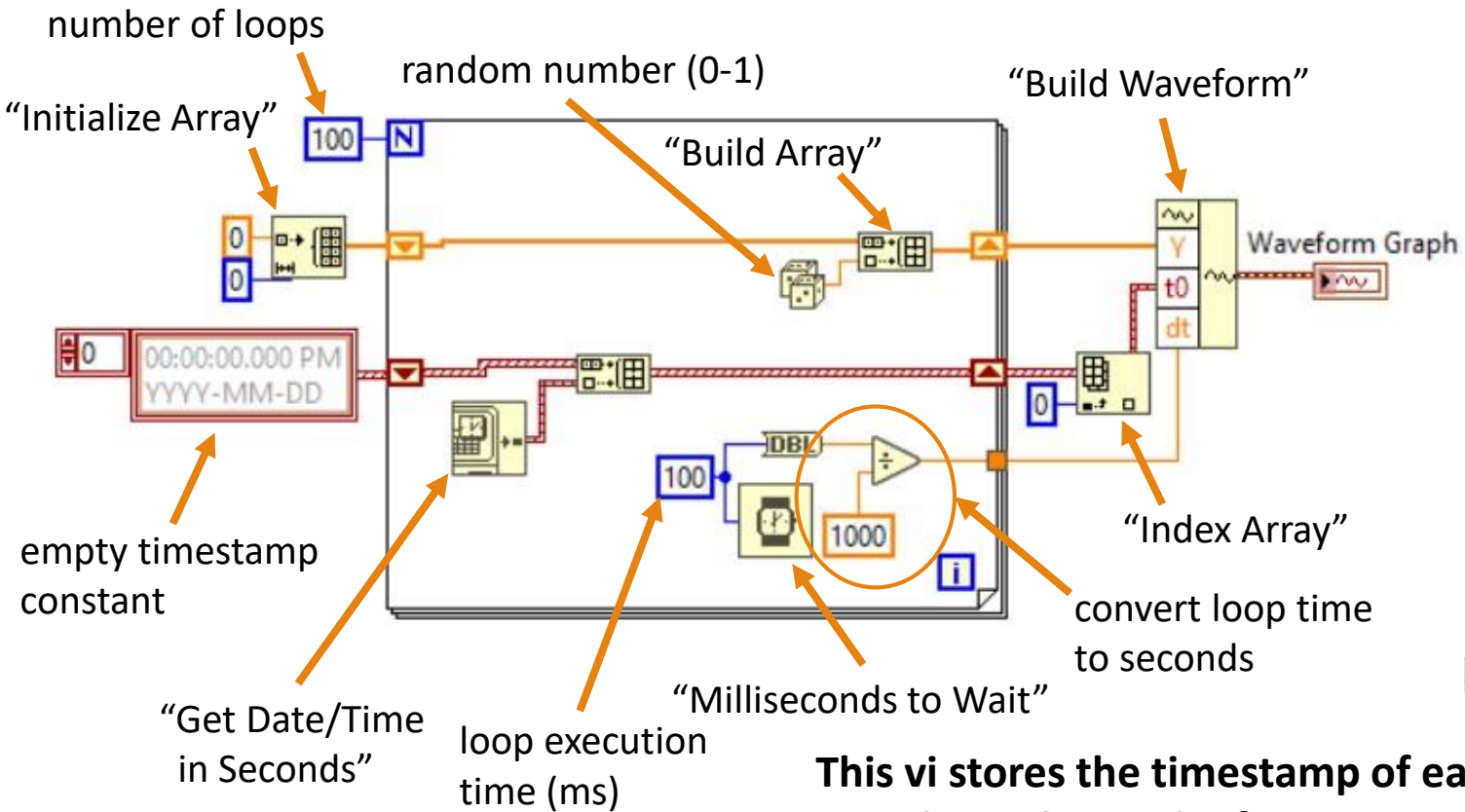
Write a program to set the parameters of a function generator with added noise and plot the output "live". Make a button to stop the program, and plot the full data set after the program is stopped. Finally, write the time and function generator data to a csv file.

Reference Slides

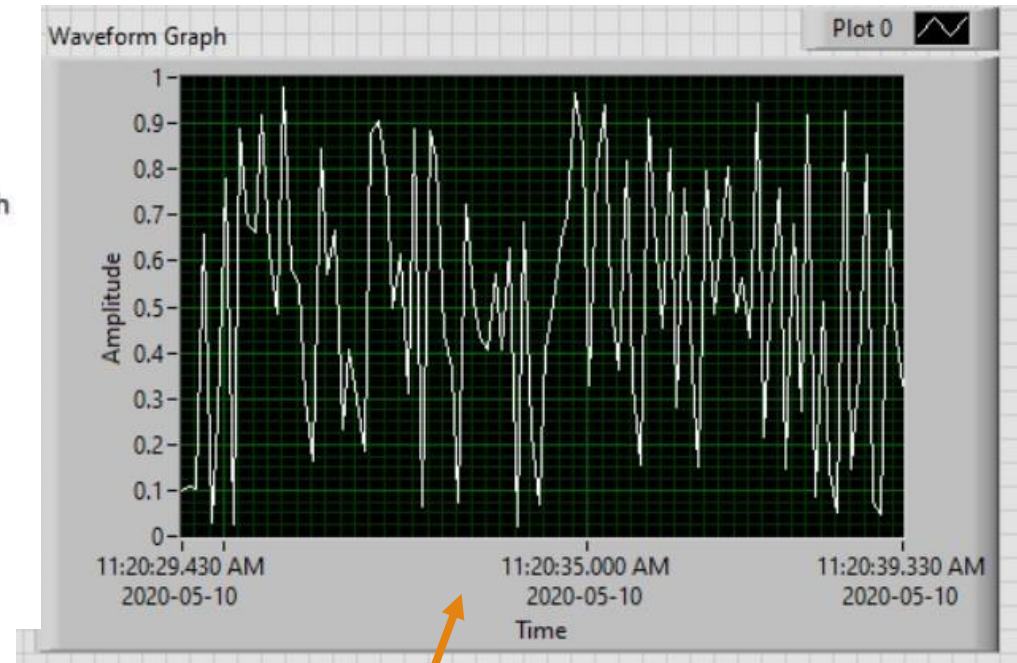
Detail on Plotting – XY Graph



Detail on Plotting – Waveform Graph

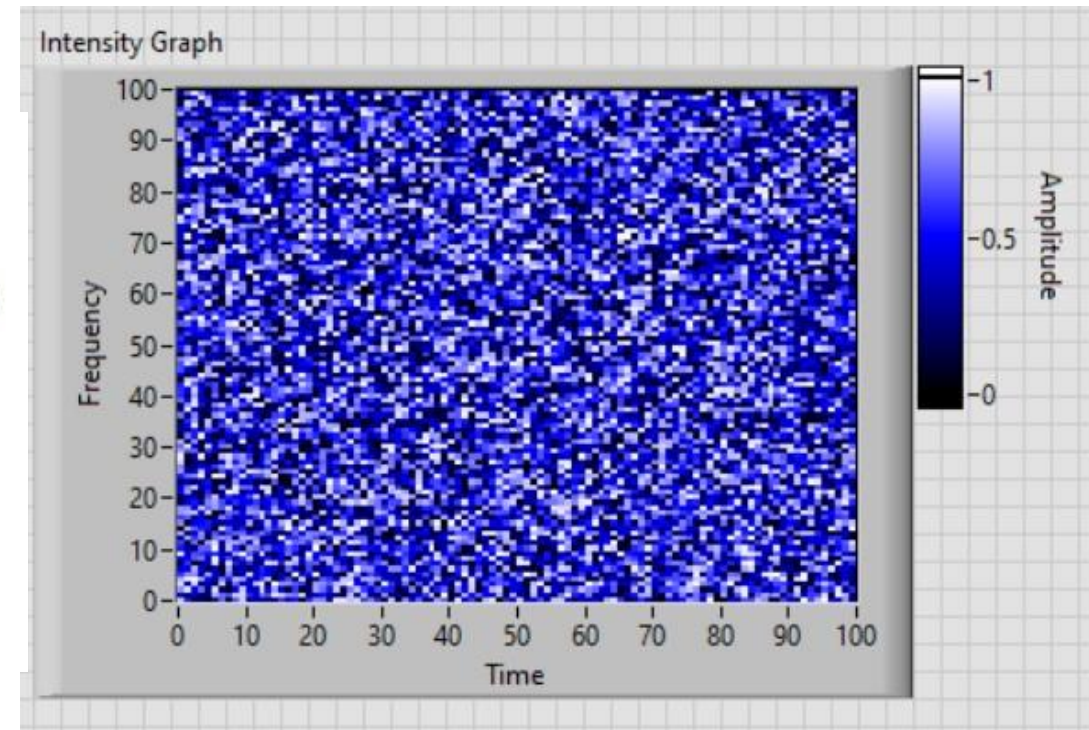
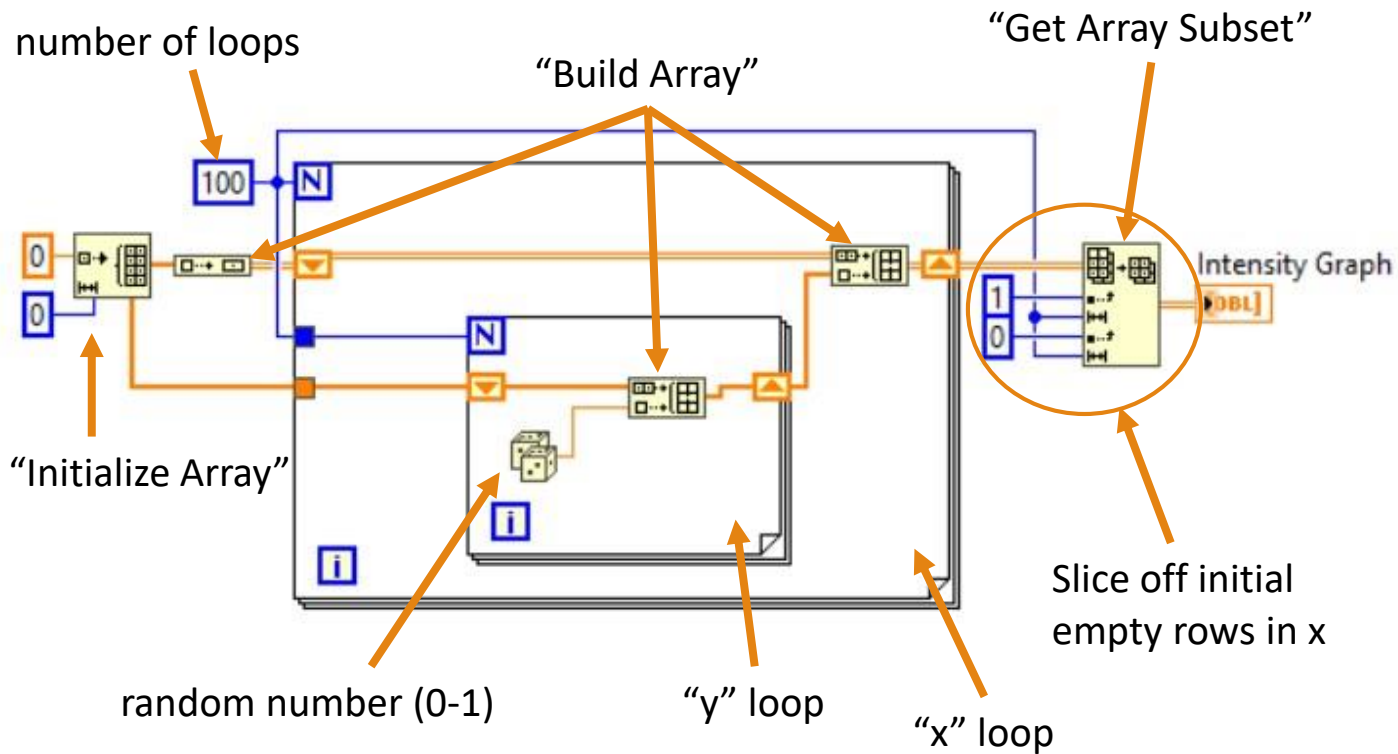


This vi stores the timestamp of each sample, and uses the first as t0



plotted as function of timestamp
this can also be plotted by time in seconds

Detail on Plotting – Intensity Graph

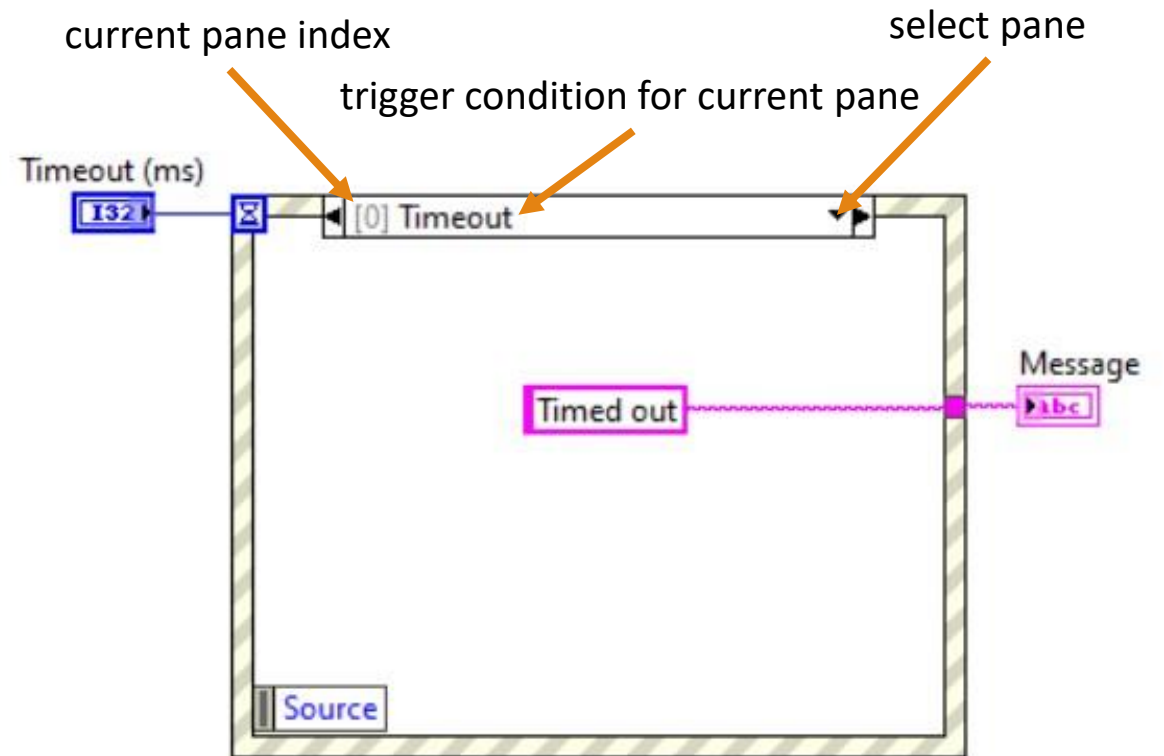


Detail on Structures -- Events

Event structures are triggered objects that execute the contents of the structure when the trigger condition is met.

Like a case structure, there can be many triggers that execute different commands, each trigger condition has its own “pane”.

The “timeout” condition is the default, and will trigger when the time wired to the hourglass icon elapses. This can be disabled by removing the “Timeout” case.



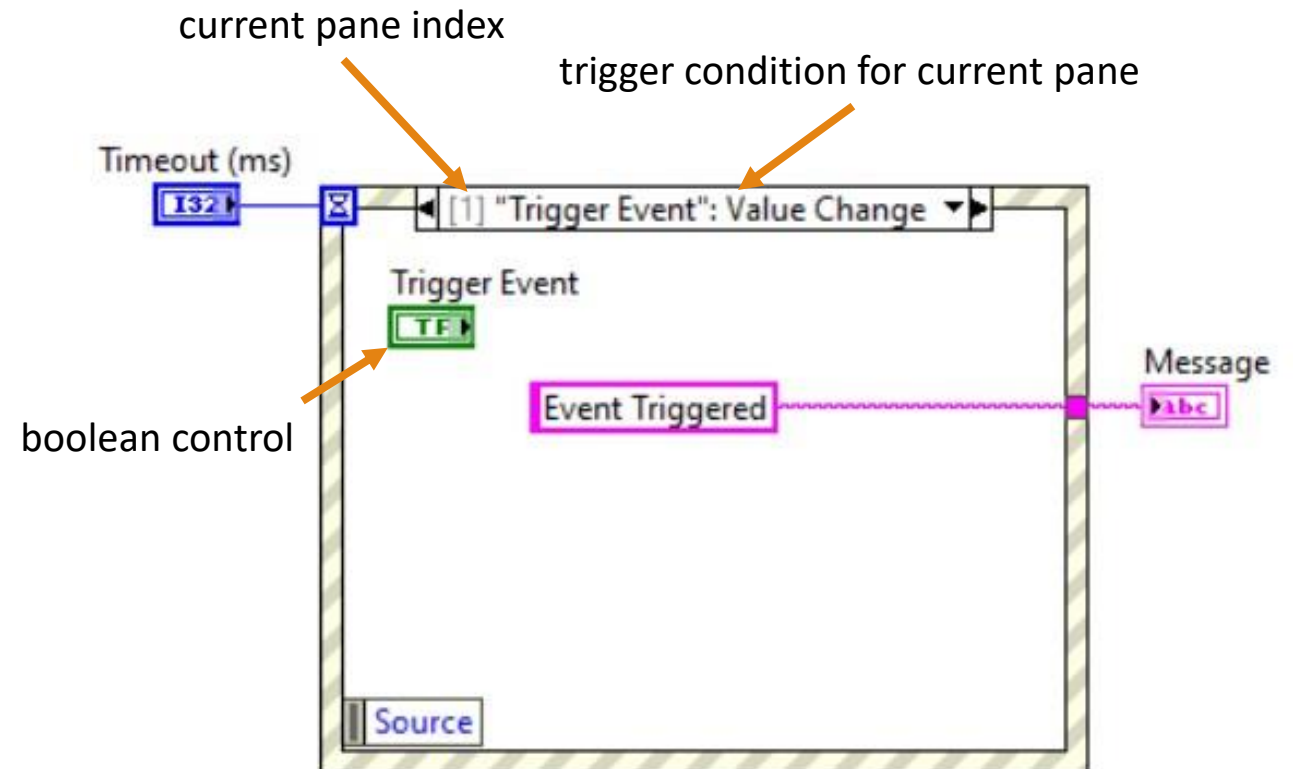
After “Timeout (ms)” has elapsed since the last execution of the event, this writes “Timed out” to a string indicator

Detail on Structures -- Events

Event structures are triggered objects that execute the contents of the structure when the trigger condition is met.

Like a case structure, there can be many triggers that execute different commands, each trigger condition has its own “pane”.

This case uses a Boolean to trigger the event, but this is not the only option. The “Value Change” Condition can be used on any data type. However, Booleans are really handy in practice



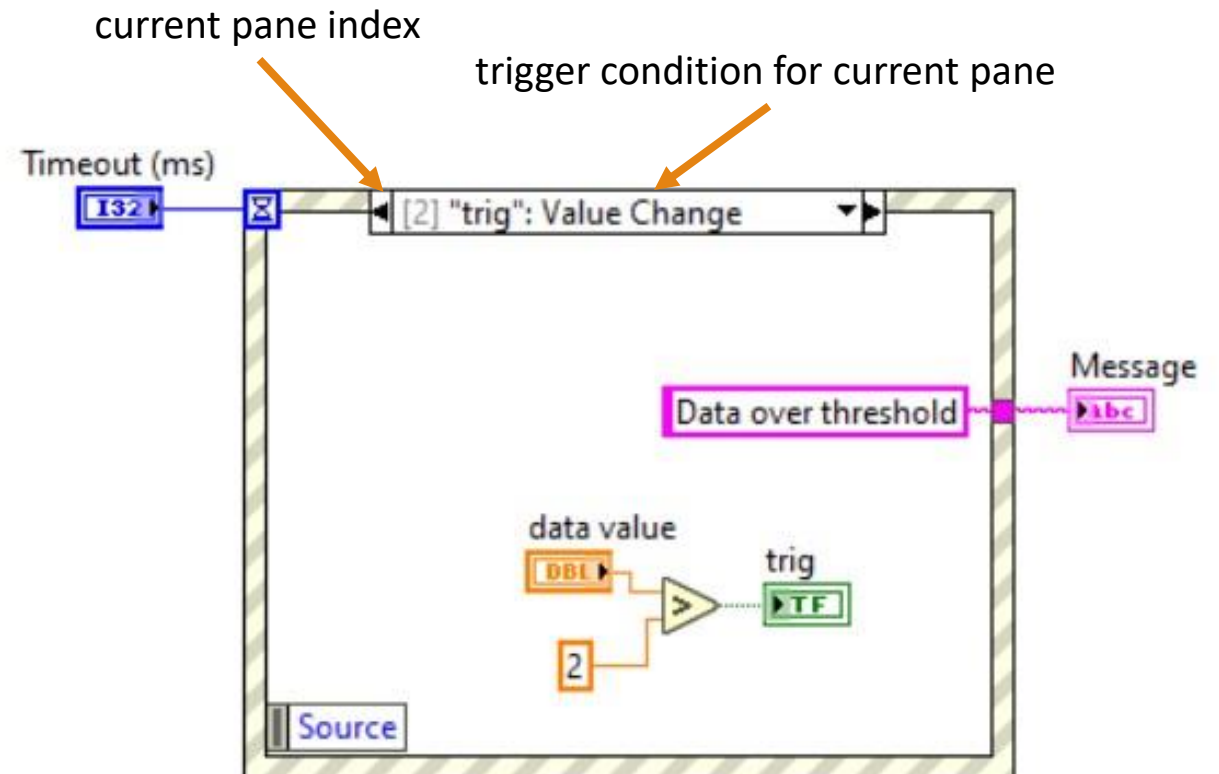
If the “Trigger Event” switch is toggled, this writes “Event Triggered” to the same string indicator

Detail on Structures -- Events

Event structures are triggered objects that execute the contents of the structure when the trigger condition is met.

Like a case structure, there can be many triggers that execute different commands, each trigger condition has its own “pane”.

This case uses a comparator to generate a Boolean that triggers the event.

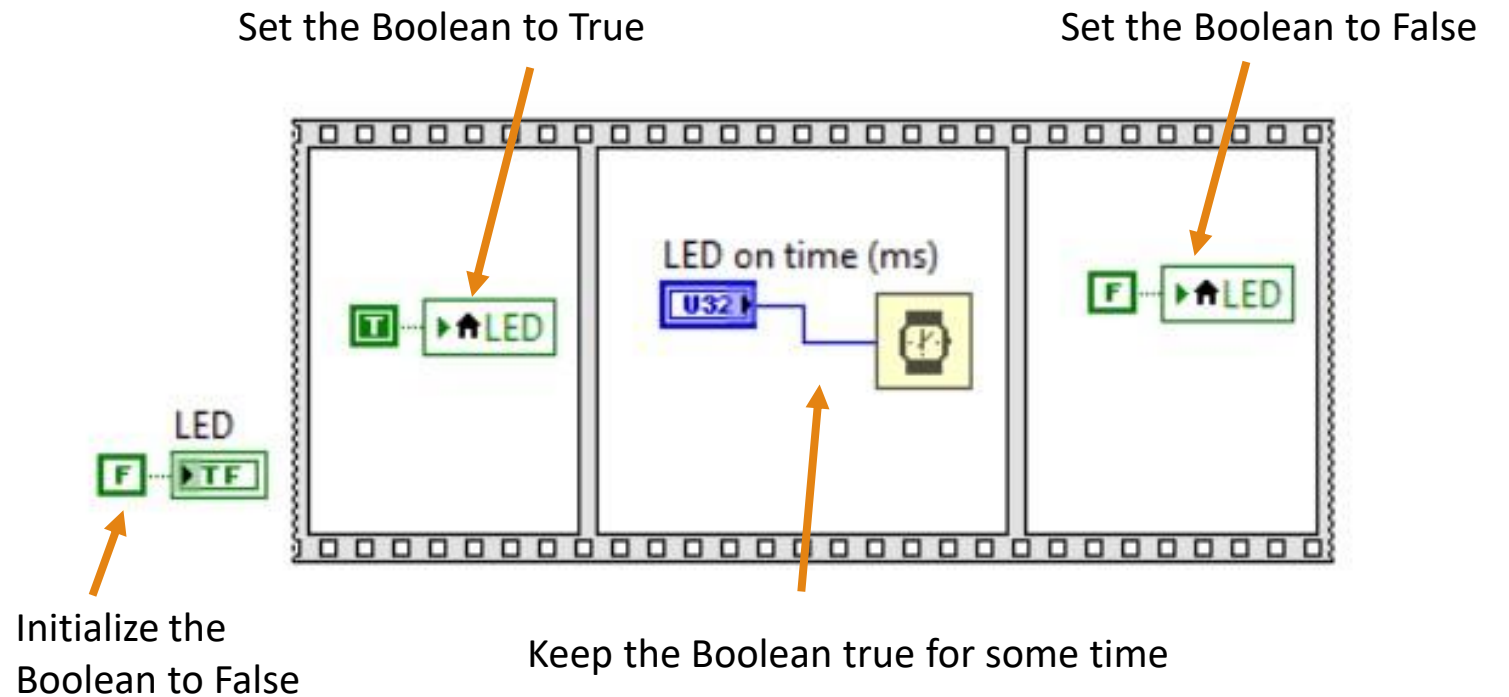


If the data value goes over the constant value, this writes “Data over threshold” to the same string indicator

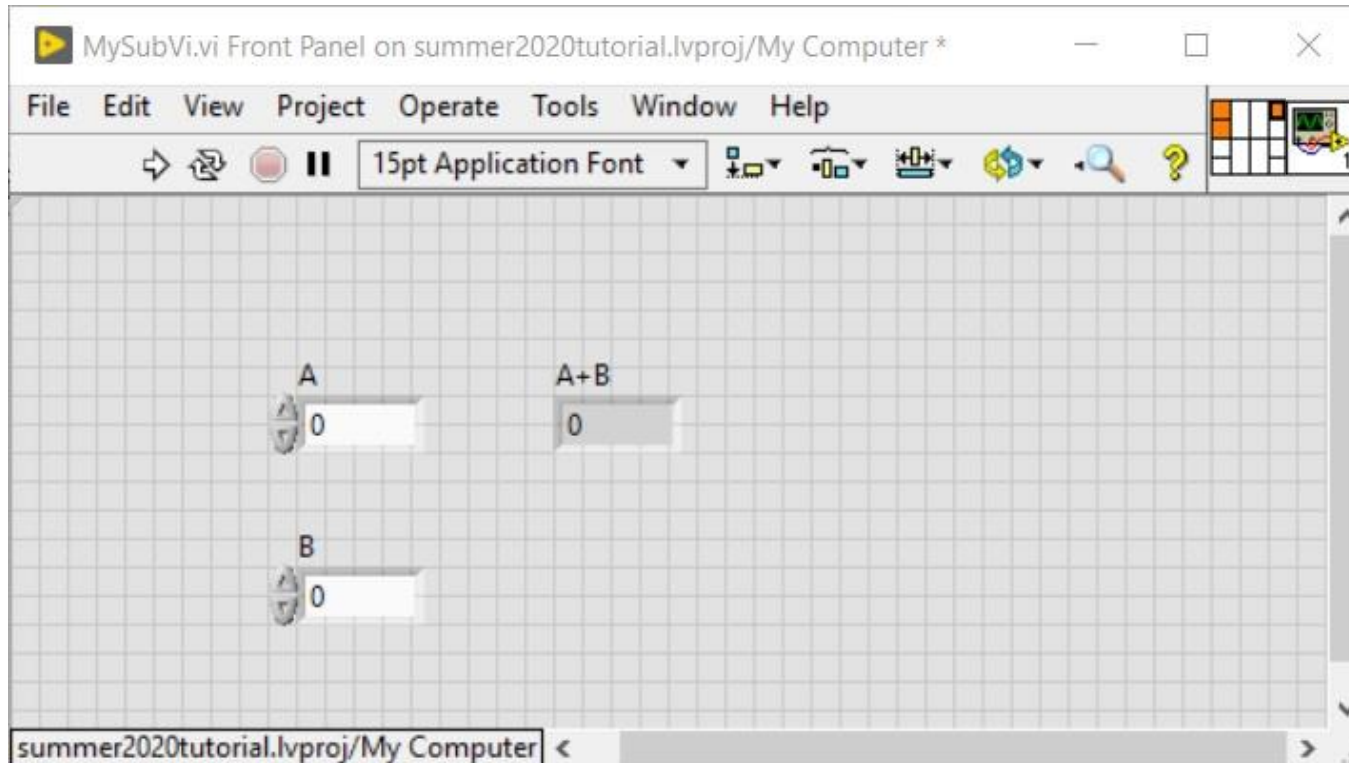
Detail on Structures -- Sequences

Sequences allow a step-by-step procedure to be followed by the program. This is especially helpful when setting up an experimental procedure by avoiding “race” conditions, where two processes run simultaneously and asynchronously.

For example, say you want to turn on a light source and take data while it is on. A sequence ensures the data is taken only while the light source is on.

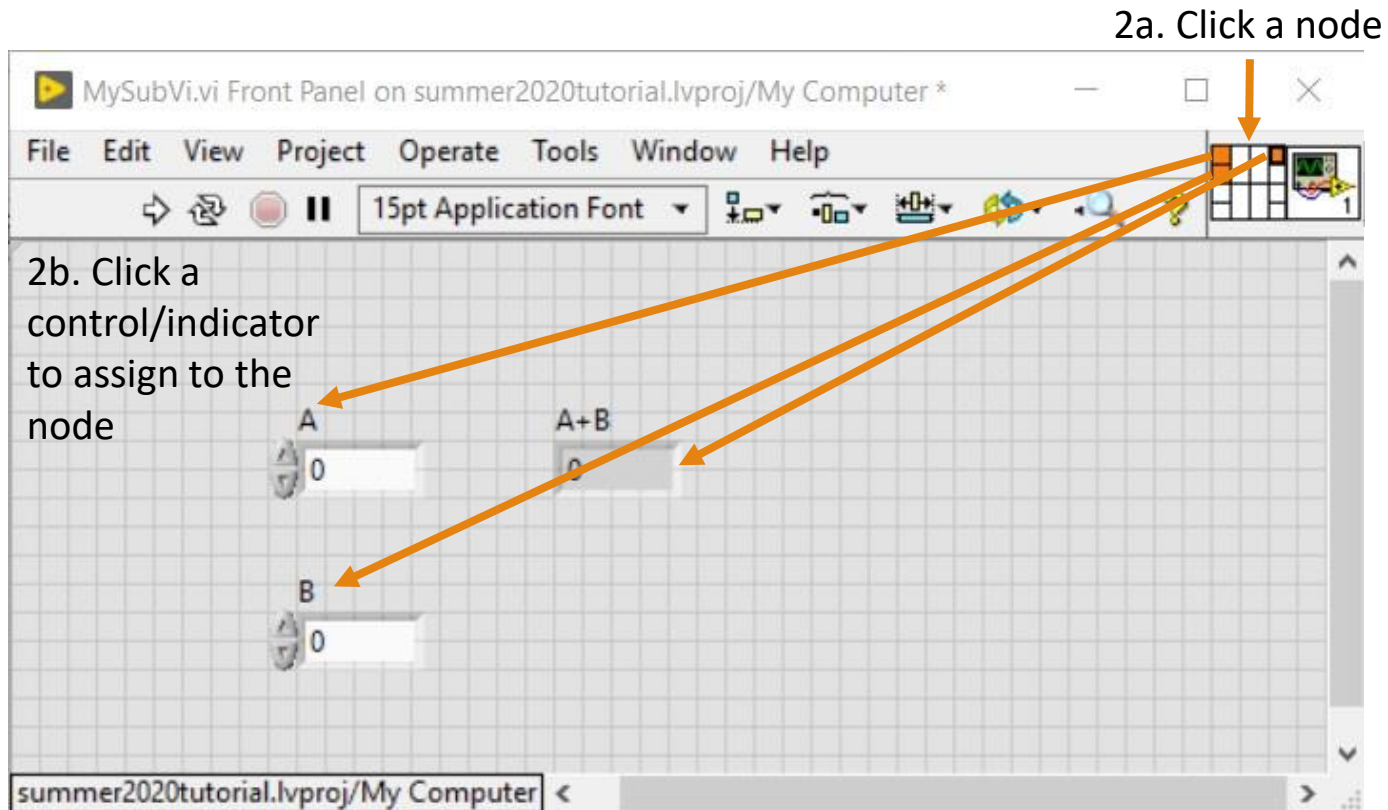


Creating a Custom Sub-vi



1. Write a vi to do the desired function
2. Select the nodes to wire to the controls and indicators. These will be the inputs and outputs of the sub-vi.
3. Customize the icon!
4. Save the vi with a suitable name
5. Use your new sub-vi using “Select a VI...” in the block diagram menu

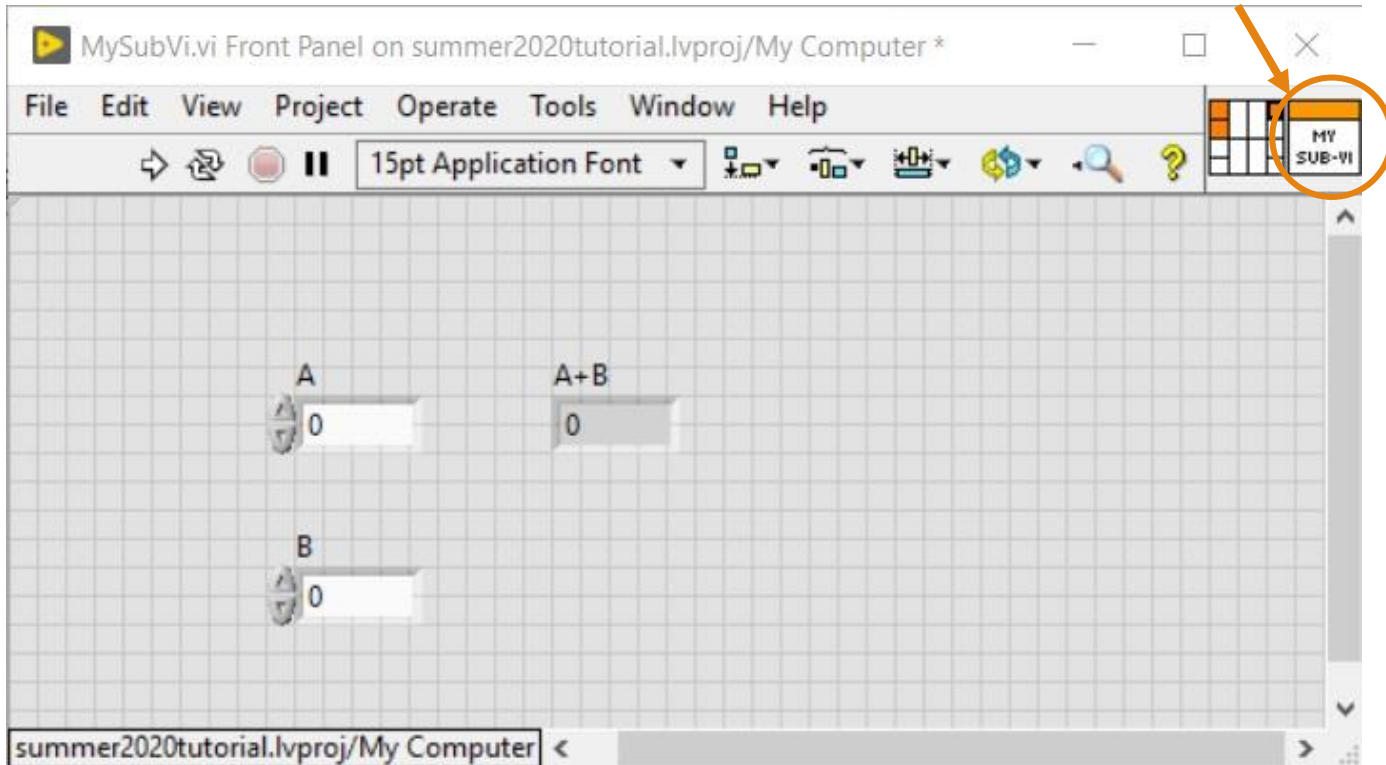
Creating a Custom Sub-vi



1. Write a vi to do the desired function
2. Select the nodes to wire to the controls and indicators. These will be the inputs and outputs of the sub-vi.
3. Customize the icon!
4. Save the vi with a suitable name
5. Use your new sub-vi using “Select a VI...” in the block diagram menu

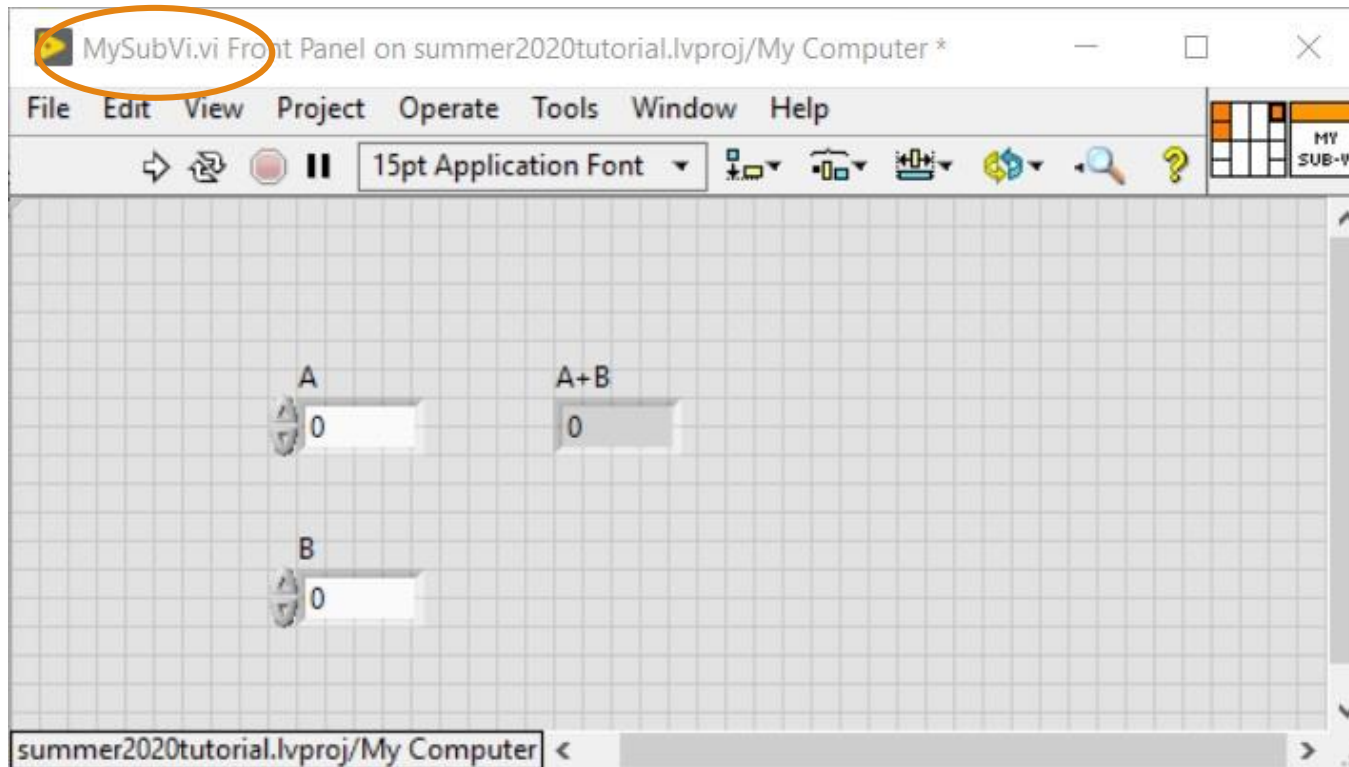
Creating a Custom Sub-vi

Right click and select "Edit Icon..."



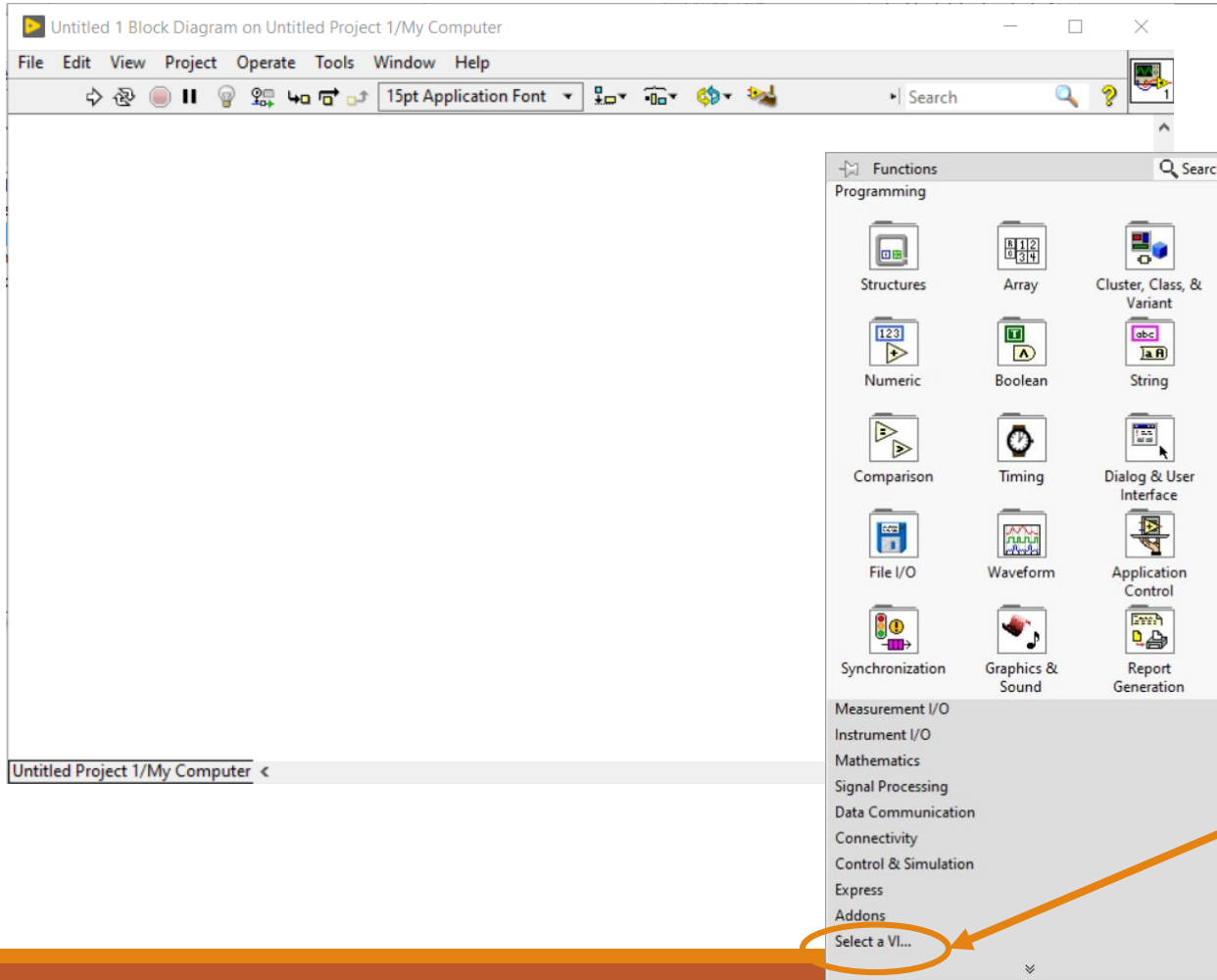
1. Write a vi to do the desired function
2. Select the nodes to wire to the controls and indicators. These will be the inputs and outputs of the sub-vi.
3. Customize the icon!
4. Save the vi with a suitable name
5. Use your new sub-vi using "Select a VI..." in the block diagram menu

Creating a Custom Sub-vi



1. Write a vi to do the desired function
2. Select the nodes to wire to the controls and indicators. These will be the inputs and outputs of the sub-vi.
3. Customize the icon!
4. Save the vi with a suitable name
5. Use your new sub-vi using “Select a VI...” in the block diagram menu

Creating a Custom Sub-vi



1. Write a vi to do the desired function
2. Select the nodes to wire to the controls and indicators. These will be the inputs and outputs of the sub-vi.
3. Customize the icon!
4. Save the vi with a suitable name
5. Use your new sub-vi using “Select a VI...” in the block diagram menu