

Experiment 1

Introduction to LabVIEW

1 Objectives

- Introduce the student to LabVIEW software.
- Design simple LabVIEW programs to demonstrate the importance of LabVIEW software in engineering application.

2 Apparatus

LabVIEW software downloaded on the laboratory PCs.

3 Introduction

Many modern applications of mechatronics system require accurate data measurements and fast and precise control action. Data Acquisition system and data logging become an important part of automatic control system in the late 1960s and early in the 1970s. Throughout this course the concept of data acquisition and its elements will be introduced, starting with data acquisition software such as LabVIEW.

4 Theoretical Background:

4.1 Getting Started with LabVIEW Virtual Instruments

LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. LabVIEW contains a comprehensive set of tools for acquiring, analyzing, displaying, and storing data, as well as tools to help you troubleshoot code you write.

In LabVIEW, you build a user interface, or front panel, with controls and indicators. Controls are knobs, push buttons, dials, and other input mechanisms. Indicators are graphs, LEDs, and other output displays. After you build the front panel, you add code using VIs and structures to control the front panel objects. The block diagram contains this code.

4.2 Launching the LabVIEW Environment

When you launch LabVIEW, the **Getting Started** window appears as shown by Figure 1.1.

Use the **Getting Started** window to create new projects and VIs. You can create items from scratch or from templates and samples. You can also open existing LabVIEW files and access LabVIEW community resources and help.

The **Getting Started** window disappears when you open an existing file or create a new file, and reappears when you close all open front panels and block diagrams. You can display the window at any time by selecting **View»Getting Started Window**.

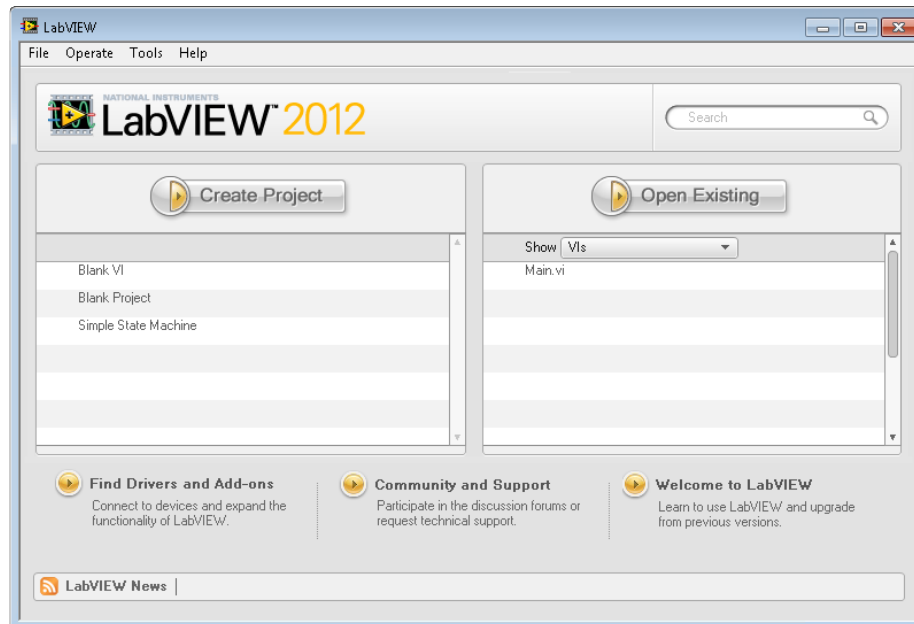


Figure 1.1: LabVIEW Getting Started Window

Projects

VIs are LabVIEW programs, and you can use multiple VIs together to make a LabVIEW application. To group these application-related VIs together, use a LabVIEW project. When you save a LabVIEW project from the **Project Explorer** window, LabVIEW creates a project file (.lvproj), that includes references to all the LabVIEW files and non-LabVIEW files in the project, configuration information, build information, and deployment information.

The **Project Explorer** window includes the following items by default:

- **Project root**—Contains all other items in the **Project Explorer** window. The label on the project root includes the filename for the project.
- **My Computer**—Represents the local computer as a target in the project.
- **Dependencies**—Includes VIs and items that VIs under a target require.
- **Build Specifications**—Includes build configurations for source distributions and other types of builds available in LabVIEW toolkits and modules. If you have the LabVIEW Professional

Development System or Application Builder installed, you can use **Build Specifications** to configure stand-alone applications, shared libraries, installers, and zip files.

4.3 VI Components

A VI contains the following three components:

1. Front panel—Serves as the user interface.
2. Block diagram—Contains the graphical source code that defines the functionality of the VI.
3. Icon and connector pane—Identifies the interface to the VI so that you can use the VI in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages. This will not be discussed for the purpose of this course.

4.3.1 Front Panel

The front panel, shown as follows, is the user interface of the VI. You build the front panel using controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input mechanisms. Indicators are graphs, LEDs, and other output displays. Controls simulate instrument input mechanisms and supply data to the block diagram of the VI. Indicators simulate instrument output mechanisms and display data the block diagram acquires or generates.

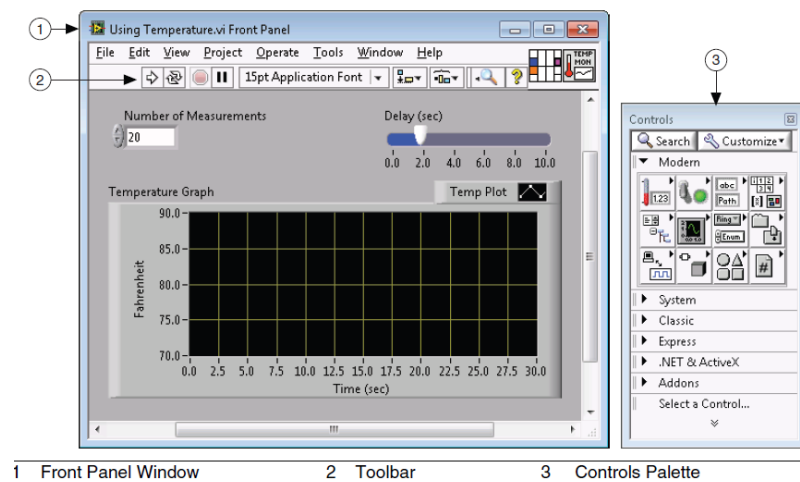


Figure 1.2: Front Panel Window

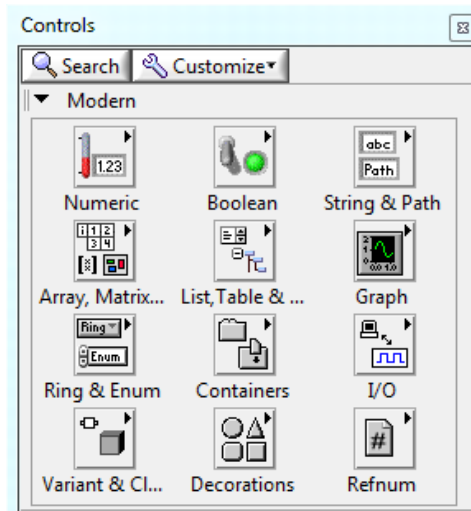


Figure 1.3: Control Pallet

Some of the modern type controls are shown in Figure 1.3. During the lab sessions, you may need to deal with:

1. Numeric: numeric includes both controls and indicators for any numerical analog value such as voltage, temperature, pressure, gains and numbers.

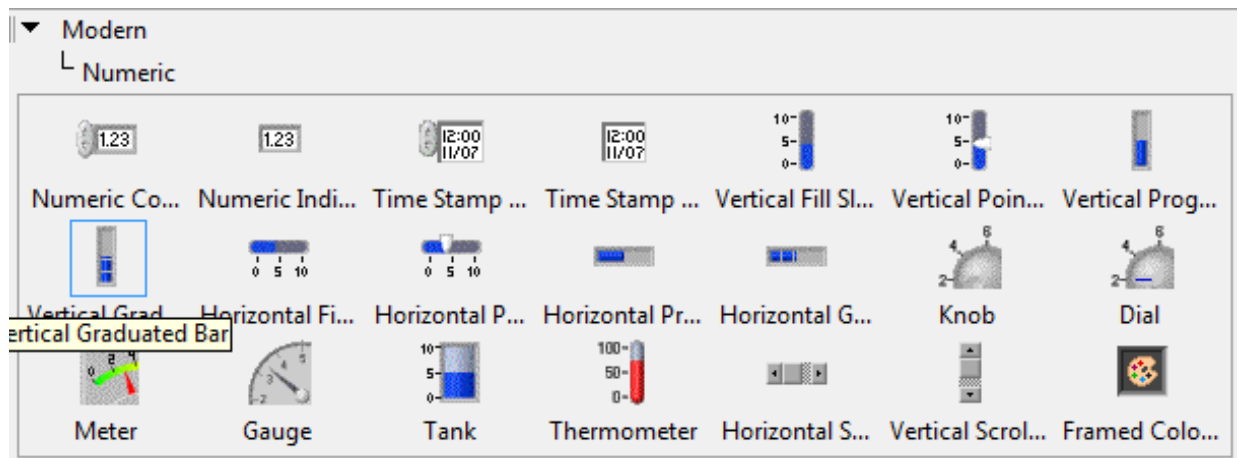


Figure 1.4: Numeric Controls and Indicators

2. Boolean: includes any control or indicator of digital (Boolean) states, on/off states. Such as push buttons, switches and leds.

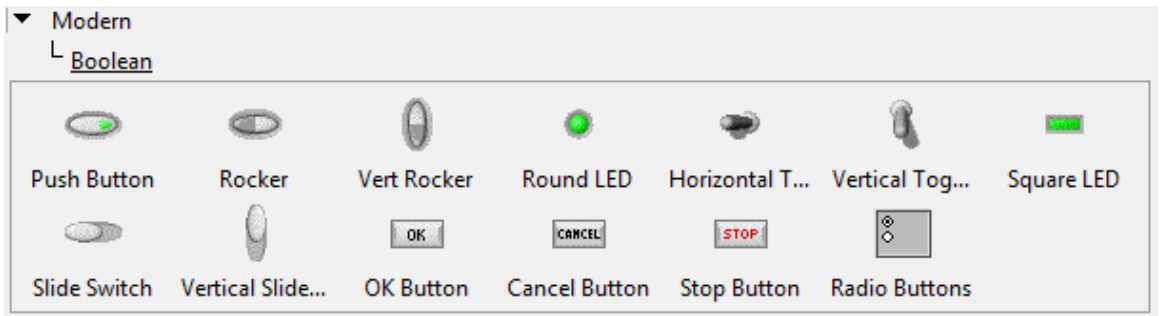


Figure 1.5: Boolean Controls and Indicators

- 3. String & Path: such as passwords, user names or directories.

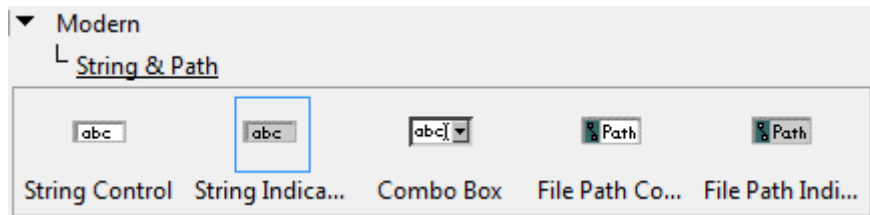


Figure 1.6: String and Path Controls and Indicators

- 4. Ring and Enum: enumerated controls are controls that combine both the neumerics and strings characteristics. You can define the values in the enum control as follows:

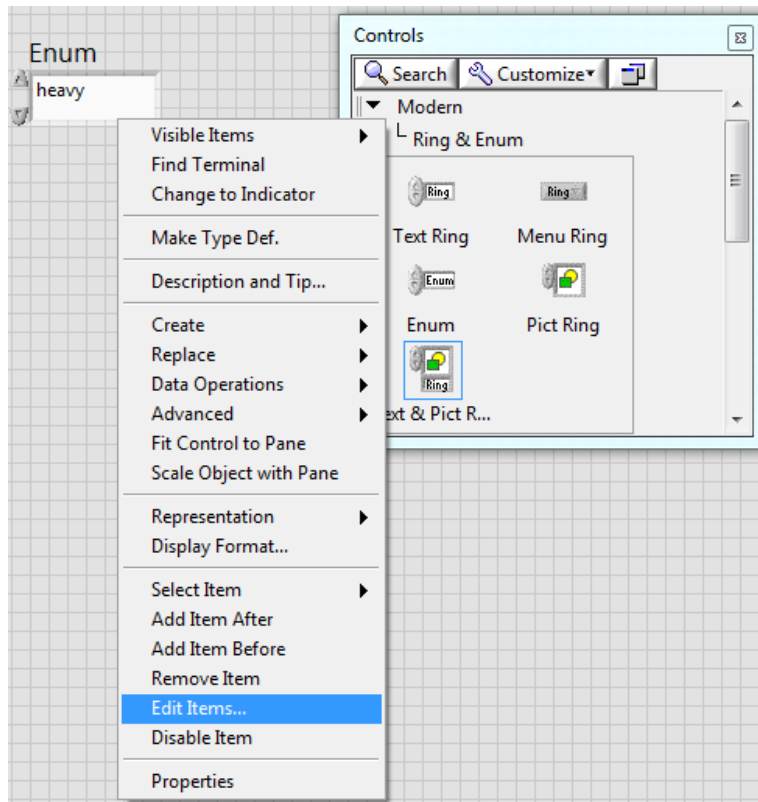


Figure 1.7: Editing the Values of an "enum"

4.3.2 Block Diagram

After you build the front panel, you add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code, also known as G code or block diagram code. Front panel objects appear as terminals on the block diagram.

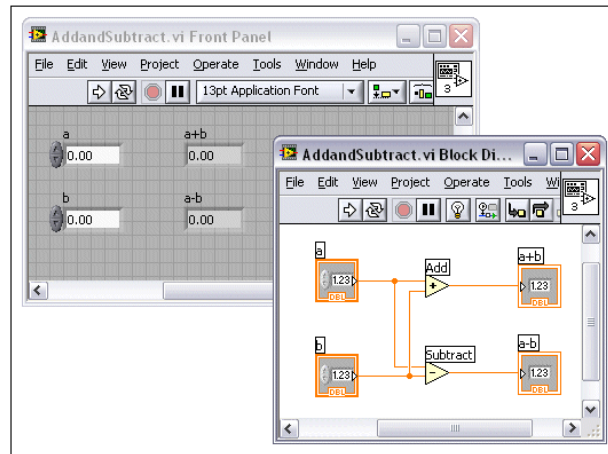


Figure 1.9: Block Diagram Window

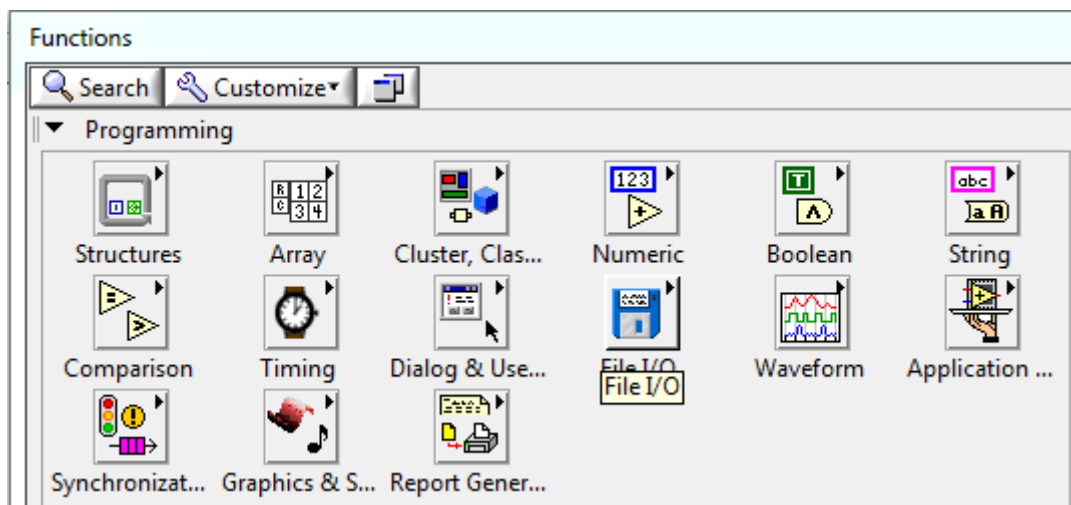


Figure 1.10: Function Pallet

VI contains several primary block diagram objects—terminals, functions, and wires.

a. Terminals

The terminals represent the data type of the control or indicator. You can configure front panel controls or indicators to appear as icon or data type terminals on the block diagram. By default, front panel objects appear as icon terminals. For example, a knob icon terminal, shown as follows, represents a knob on the front panel.



The DBL at the bottom of the terminal represents a data type of double-precision, floating-point numeric. A DBL terminal, shown as follows, represents a double-precision, floating-point numeric control.



Terminals are entry and exit ports that exchange information between the front panel and block diagram. Data you enter into the front panel controls (a and b in the previous front panel) enter the block diagram through the control terminals. The data then enter the Add and Subtract functions. When the Add and Subtract functions complete their calculations, they produce new data values. The data values flow to the indicator terminals, where they update the front panel indicators (a+b and a-b in the previous front panel).

b. Nodes

Nodes are objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs. They are analogous to statements, operators, functions, and subroutines in text-based programming languages. The Add and Subtract functions in the previous block diagram are examples of nodes. The most commonly used nodes in LabVIEW are:

1. Numeric

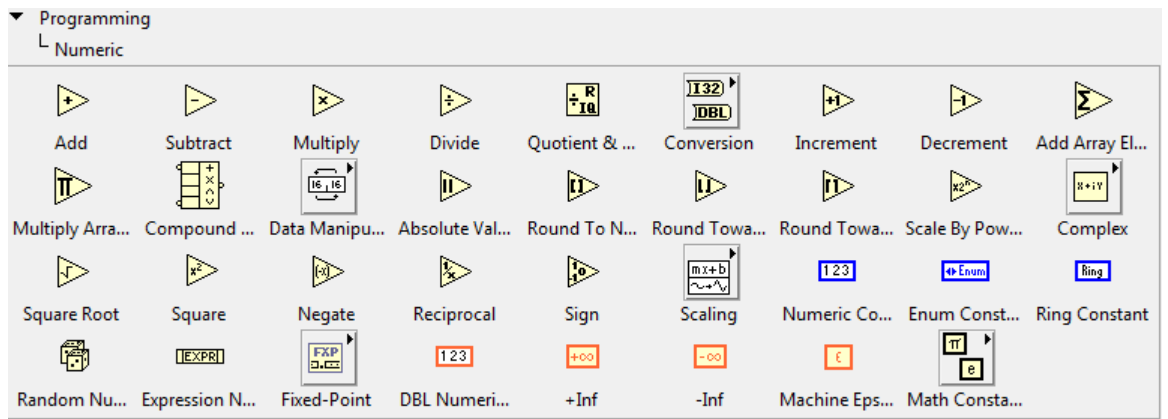


Figure 1.14: Numeric Nodes (functions and constants)

2. String

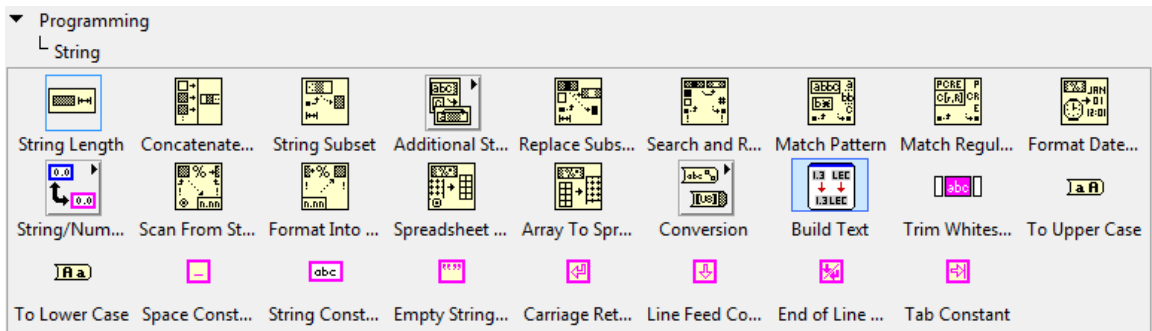


Figure 1.15: String Nodes (functions and constants)

3. Boolean

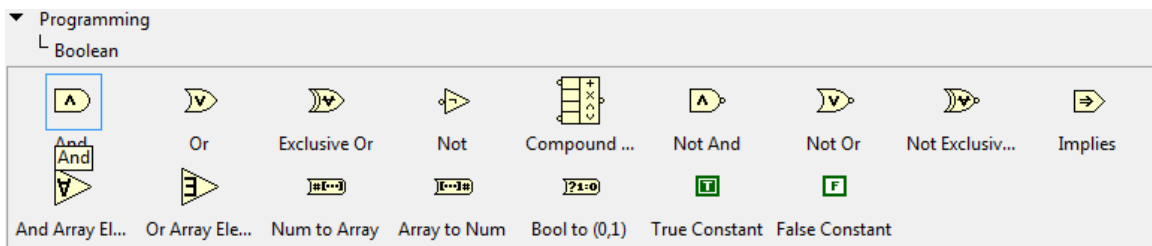


Figure 1.16: Boolean Nodes (functions and constants)

4. Comparison

Comparison contains any function needed check conditions.

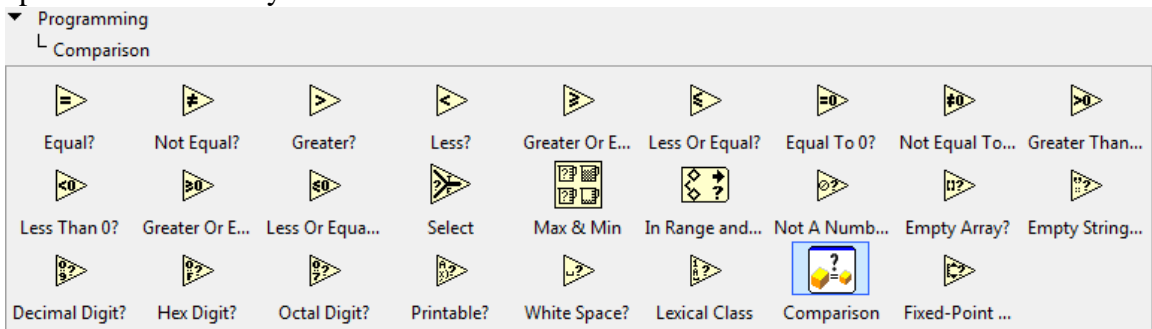


Figure 1.17: Comparison Nodes

c. Structures

Structures, a type of node, are graphical representations of the loops and case statements of text-based programming languages. Use structures on the block diagram to repeat blocks of code and to execute code conditionally or in a specific order.

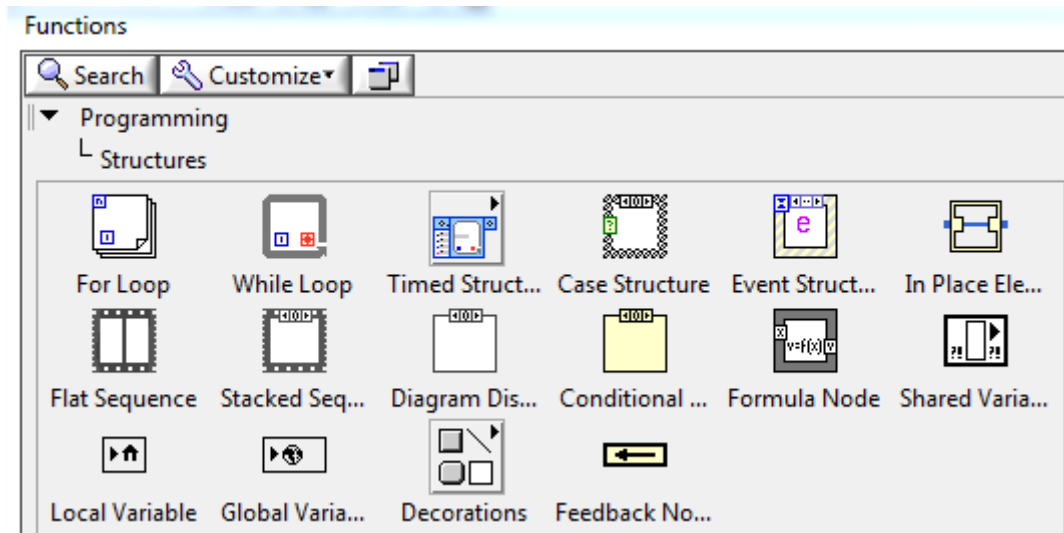


Figure 1.19: Structures

While Loop

A While Loop executes a subdiagram until a condition occurs. The While Loop is located on the Structures palette. Select the While Loop from the palette then use the cursor to drag a selection rectangle around the section of the block diagram you want to repeat. When you release the mouse button, a While Loop boundary encloses the section you selected. Add block diagram objects to the While Loop by dragging and dropping them inside the While Loop.

Tip The While Loop always executes at least once because it execute the code inside the while then check the termination condition. Changing the value of the controls outside the loop, does not affect the loop because the value is only read once.

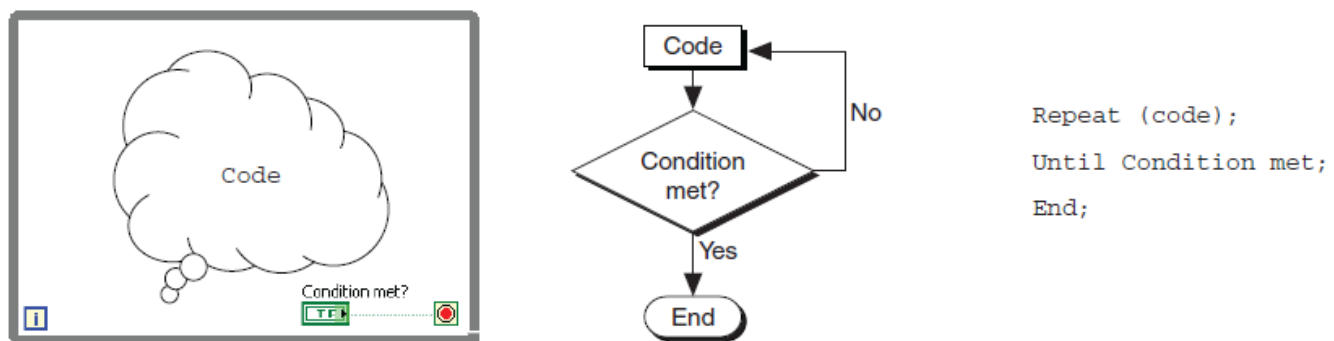


Figure 1.20: How Do While Loop Work

The iteration terminal is an output terminal that contains the number of completed iterations. The iteration terminal provides the current loop iteration count, which is zero for the first iteration. Tunnels feed data into and out of structures like While Loops. The block is the color of the data type wired to the tunnel. Data pass out of a loop *only* after the loop terminates.

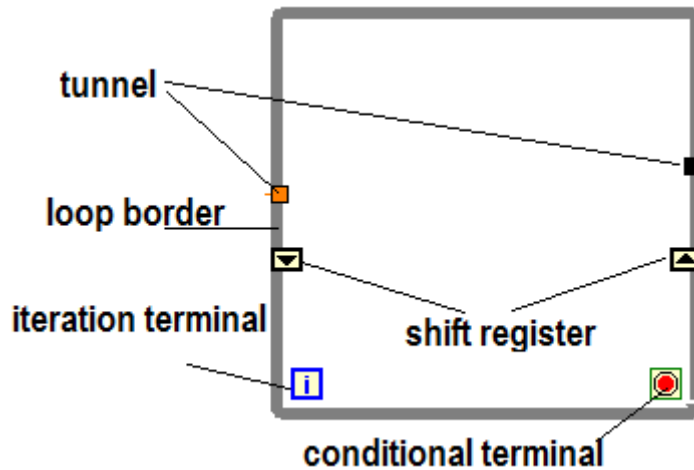


Figure 1.21: While Loop Elements

Use shift registers when you want to pass values from previous iterations through the loop to the next iteration. A shift register appears as a pair of terminals directly opposite each other on the vertical sides of the loop border. Initializing a shift register resets the value the shift register passes to the first iteration of the loop when the VI runs. If you do not initialize the shift register, the loop uses the value written to the shift register when the loop last executed or, if the loop has never executed, the default value for the data type.

Stacked shift registers let you access data from previous loop iterations. Stacked shift register remember values from multiple previous iterations and carry those values to the next iterations. To create a stacked shift register, right-click the left terminal and select Add Element from the shortcut menu.

If you add another element to the left terminal in the previous block diagram, values from the last two iterations carry over to the next iteration, with the most recent iteration value stored in the top shift register. The bottom terminal stores the data passed to it from the previous iteration.

For Loop

A For Loop executes a sub diagram a set number of times. The count terminal **N** is an input terminal whose value indicates how many times to repeat the sub diagram. The iteration count for the For Loop always starts at zero. An extra conditional terminal can be added by right clicking on the loop border. A For Loop with a conditional terminal executes until the condition occurs or until all iterations are complete, whichever happens first. The for loop may not execute at all because it checks the condition then enters the loop.

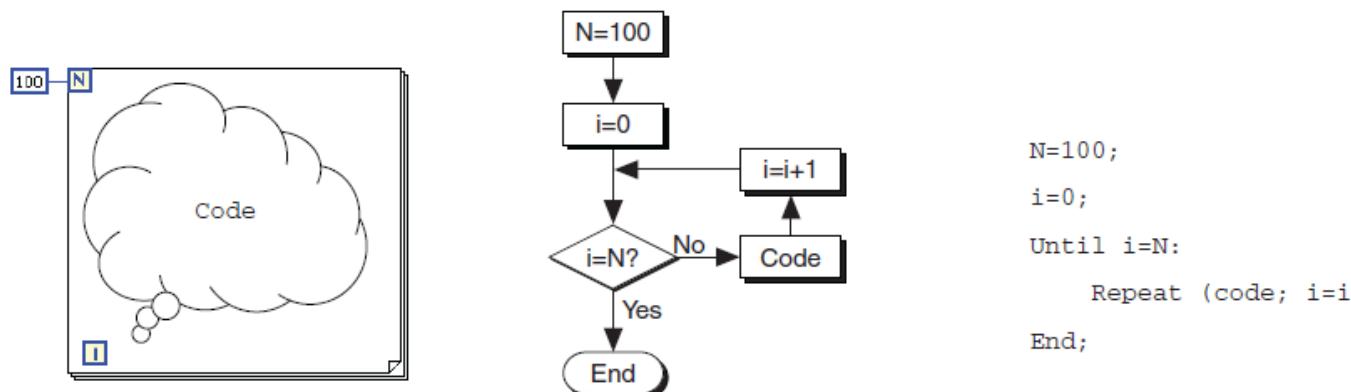


Figure 1.22: How Do For Loop Work?

The output tunnel of the for loop can be configured by right-click the tunnel, and select Tunnel Mode»

Concatenating: To automatically concatenate an array leaving a loop, wire the array directly to the loop output tunnel. Selecting Concatenating appends all inputs in order, forming an output array of the same dimension as the array input wired.

Last Value: this mode displays the last value from the last loop iteration.

Indexing builds an array of higher dimension.

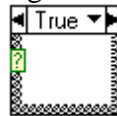
Note: You can change when LabVIEW writes values to the loop output tunnel based on a condition.

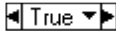
Table 1.2: For Loop Vs. While Loop

For Loop	While Loop
Executes a set number of times unless a conditional terminal is added	Stops executing only if the value at the conditional terminal meets the condition
Can execute zero times	Must execute at least once
Tunnels automatically output an array of data	Tunnels automatically output the last value


Case structure

A Case structure has two or more subdiagrams, or cases. Only one subdiagram is visible at a time, and the structure executes only one case at a time. An input value determines which subdiagram executes. The Case structure is similar to switch statements or if...then...else statements in text-based programming languages.



The case selector label  at the top of the Case structure contains the name of the selector value that corresponds to the case in the center and decrement and increment arrows on each side.

Click the decrement and increment arrows to scroll through the available cases. You also can click the down arrow next to the case name and select a case from the pull-down menu.

Wire an input value, or selector, to the selector terminal  to determine which case executes. You must wire an integer, Boolean value, string, or enumerated type value to the selector terminal. You can position the selector terminal anywhere on the left border of the Case structure. If the data type of the selector terminal is Boolean, the structure has a True case and a False case. If the selector terminal is an integer, string, or enumerated type value, the structure can have any number of cases.

If you do not specify a default case for the Case structure to handle out-of-range values, you must explicitly list every possible input value. For example, if the selector is an integer and you specify cases for 1, 2, and 3, you must specify a default case to execute if the input value is 4 or any other unspecified integer value.

By default, a case structure accepts a Boolean condition and has two cases: true and false case. The following figure shows both the true and false cases for an example which shows if "x1" equals "x2" the numeric called "out" will display a value of "2", else it will display a value of "1".

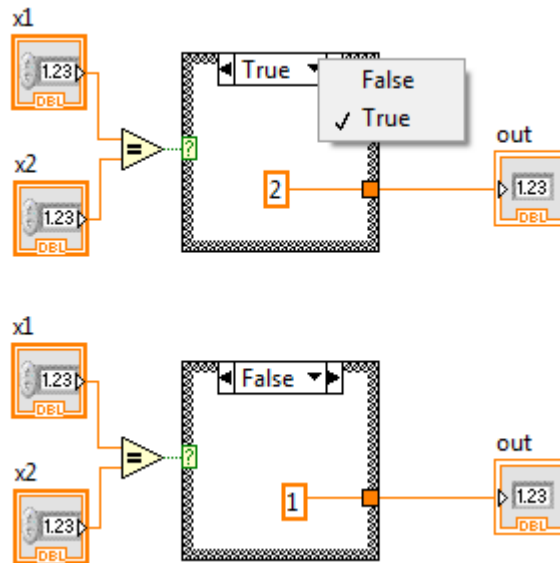


Figure 1.23: "Boolean" Case structure

The Boolean case structure is equivalent to a "select" from comparison menu.

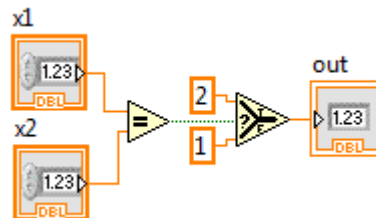


Figure 1.24: Selector

The following figure shows a case structure with an "enum" connected to its selectorterminal.

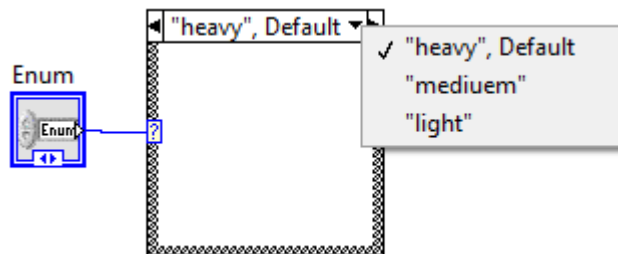


Figure 1.25: "Enum" Case Structure

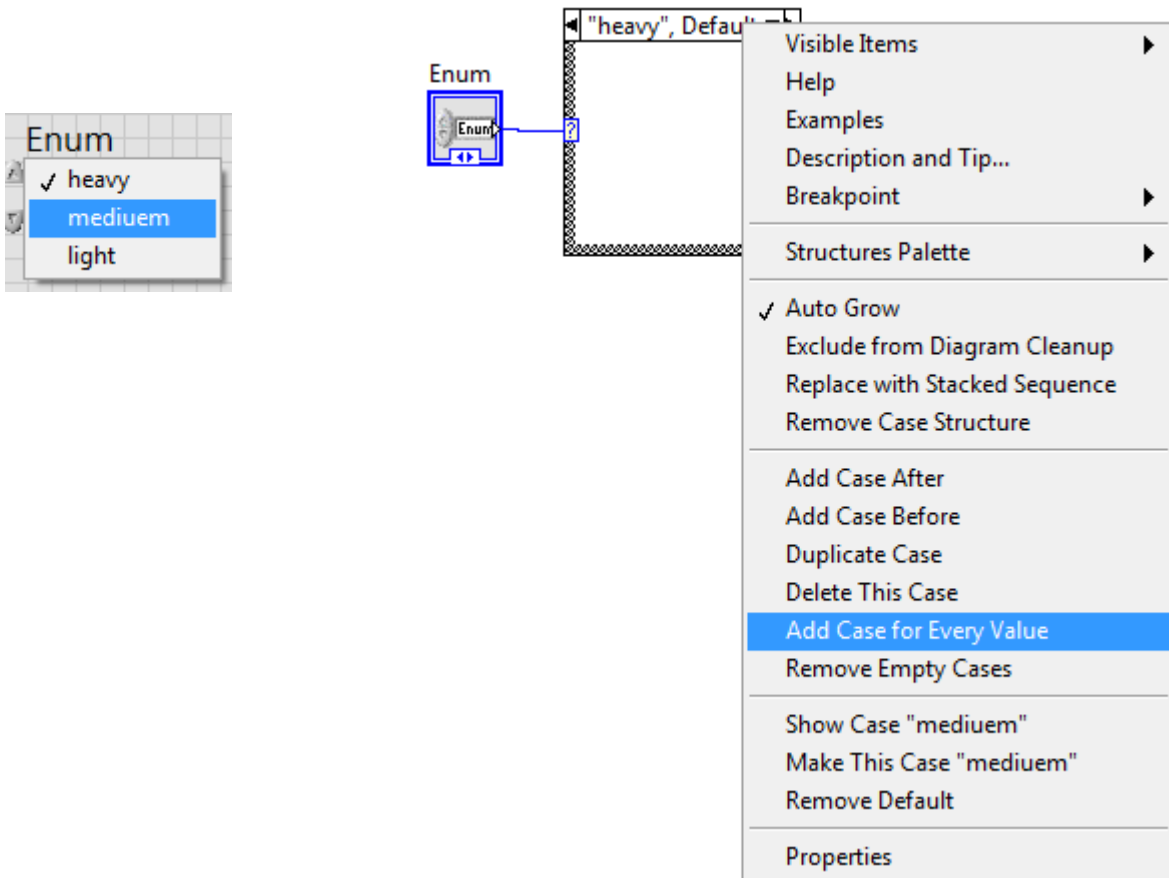


Figure 1.26: Editing Case Structure with Multiple Cases

Feedback Nodes

Use feedback nodes when you want to shift data but you don't or can't have a "while" or a "for" loop in your program.

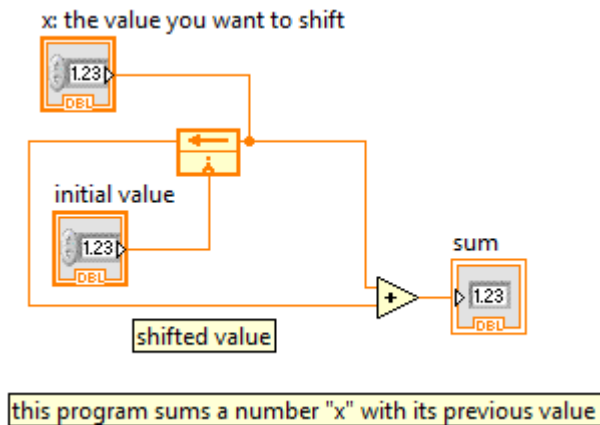


Figure 1.27: Feedback Node Example

d. Wires

You transfer data among block diagram objects through wires. A broken wire appears as a dashed black line with a red X in the middle. Broken wires occur for a variety of reasons, such as when you try to wire two objects with incompatible data types.

The VI code may be executed using the icon on the toolbars of the front panel or the block diagram shown in Figure 1 and Figure 2

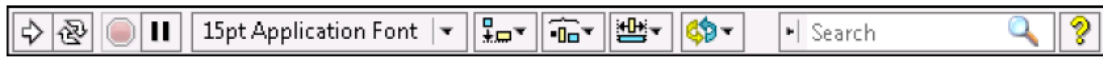


Figure 1: Front Panel Toolbar

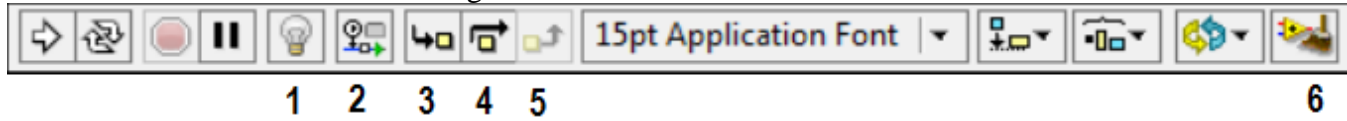
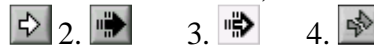


Figure 2: Block Diagram Toolbar

Click the Run button to run a VI. LabVIEW compiles the VI, if necessary. You can run a VI if the Run button appears as a solid white arrow, in 1. The solid white arrow also indicates you can use the VI as a subVI if you create a connector pane for the VI. While the VI runs, the Run button appears as shown in 2 if the VI is a top-level VI, meaning it has no callers and therefore is not a subVI. If the VI that is running is a subVI, the Run button appears as shown in 3. The Run button appears broken when the VI you are creating or editing contains errors. If the Run button still appears broken as in 4 after you finish wiring the block diagram, the VI is broken and cannot run. Click this button to display the Error list window, which lists all errors and warnings.



Click the Run Continuously button to run the VI until you abort or pause execution. You also can click the button again to disable continuous running.



While the VI runs, the Abort Execution button appears. Click this button to stop the VI immediately if there is no other way to stop the VI. If more than one running top-level VI uses the VI, the button is dimmed.



Click the Pause button to pause a running VI. When you click the Pause button, LabVIEW highlights on the block diagram the location where you paused execution, and the Pause button appears red. Click the Pause button again to continue running the VI.



The block diagram toolbar contains more debugging tools such as Highlight Execution which slow the execution and show the data flow.

4.4 Data types

1. **Boolean:** They have only two possible values: True or False and are indicated by green data wires.
2. **Numeric:** They have many sizes and representations:
 - Integers can be signed or unsigned whole numbers and are indicated by blue data wires.
 - Doubles and Singles are signed numbers with a decimal component and are indicated by orange data wires.
 - A numeric's size is indicated in bits and determines the range of possible values.
3. **String:** They are sequences of characters and are indicated by pink data wires.
4. **Array:** Arrays are a groups of one data type and are indicated by thicker data wires.
 - A 1 dimensional array can be thought of as a column, a 2 dimensional array as a table, and so on.
5. **Cluster:** Clusters are a groups of various data types and indicated by a thick brown data wire.

- An Error Cluster is a special type of cluster used to indicate warnings and errors.
 - Error clusters are composed of a boolean status, a numeric error code, and a string source.
6. **Dynamic** : When acquiring signals from or generating signals in real time, these signals are called dynamic data type.

4.5 Timing VIs

When a loop finishes executing an iteration, it immediately begins executing the next iteration, unless it reaches a stop condition. Most often, you need to control the iteration frequency or timing. Figure 3 shows the timing palette.

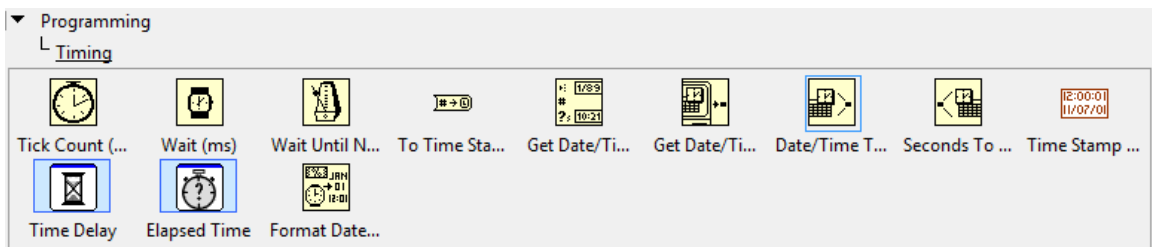


Figure 3: Timing Nodes

The most frequently used timing functions are:

1. Wait Functions

Place a wait function inside a loop to allow a VI to sleep for a set amount of time. This allows your processor to address other tasks during the wait time. Wait functions use the millisecond clock of the operating system.

- a. The **Wait (ms)** function waits until the millisecond counter counts to an amount equal to the input you specify. This function guarantees that the loop execution rate is at least the amount of the input you specify.
- When LabVIEW calls a VI for example, if **millisecond timer value** is 112 ms and **milliseconds to wait** is 10 ms, the VI finishes when **millisecond timer value** equals 122 ms.



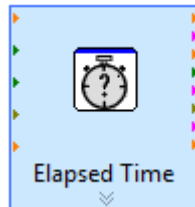
- b. The **Wait Until Next ms Multiple** function waits until the value of the millisecond timer becomes a multiple of the specified **millisecond multiple**. Use this function to synchronize activities. You can call this function in a loop to control the loop execution rate.

When LabVIEW calls a VI for example, if **millisecond multiple** is 10 ms and **millisecond timer value** is 112 ms, the VI waits 8 more milliseconds until the millisecond timer value is 120 ms, a multiple of 10.



2. Elapsed Time

Indicates the amount of time that has elapsed since the specified start time.



4.6 Configuring DAQ Using LabVIEW

Data in data acquiring system can be acquired (read from physical plant) or generated (written to a physical plant). The type of data signal that can be dealt with in LabVIEW are:

- Analog
- Digital
- Counter

Assuming the proper hardware connection is made (which will be discussed later in this course) the “DAQ Assist” function is used to initialize the measurement procedure for each signal type as described by the following steps.

1. Inserting a DAQ assistant

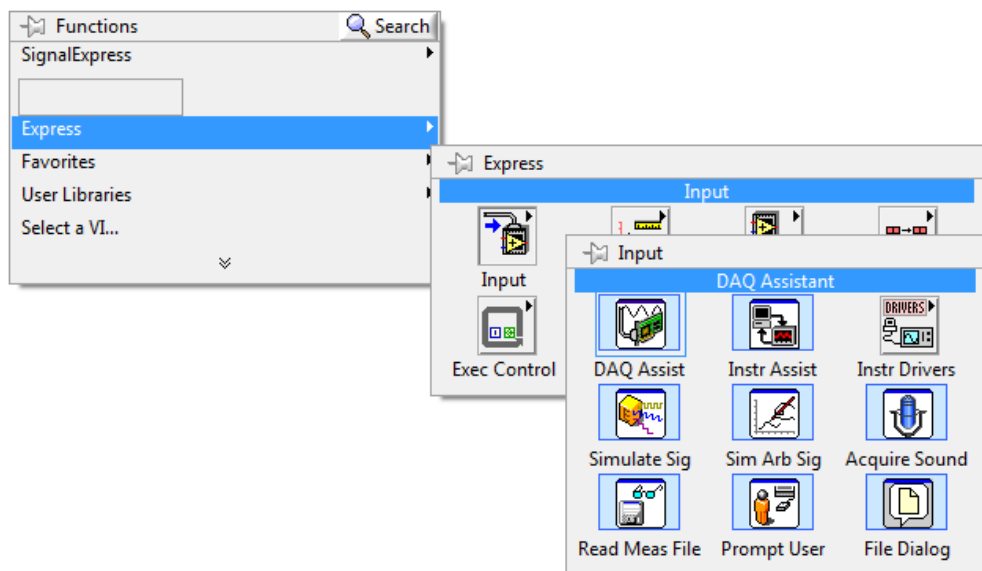


Figure 1.35: Inserting a DAQ Assistant

The following dialog box will open:

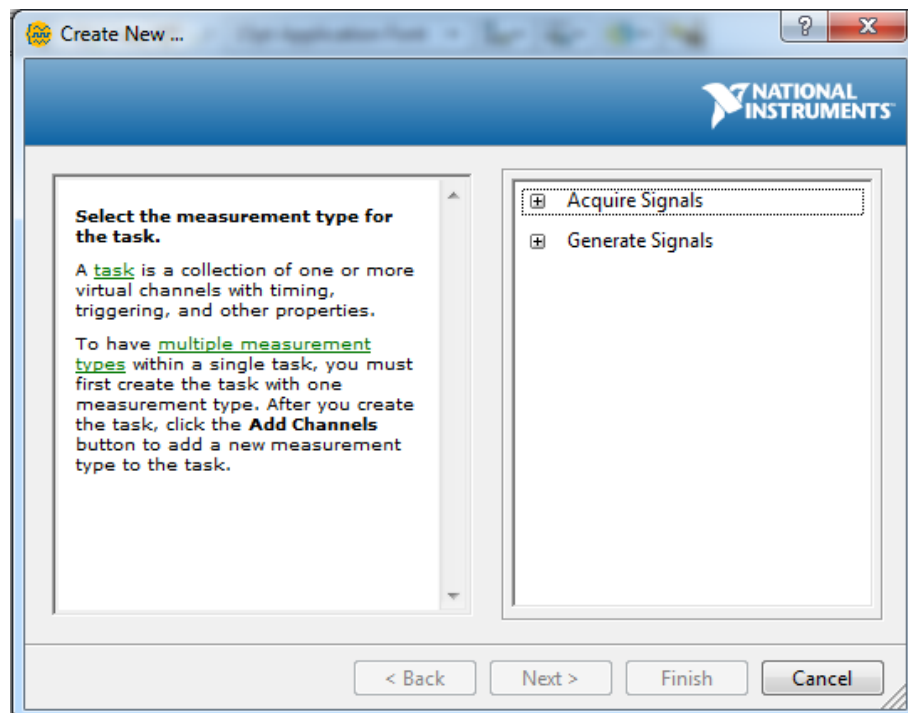


Figure 1.36: DAQ Assistant Dialog Box

- Select **Acquire Signals** to read inputs such as sensors and push buttons.
- Select **Generate Signals** to write output signals such as operating leds or drive circuits.

2. Acquiring signal

Selecting "Acquire signal" will open the following options:

- Analog inputs:

You can select the proper analog input based on the measurement you are making provided that your **Data Acquisition Card** supports that type of measurement.

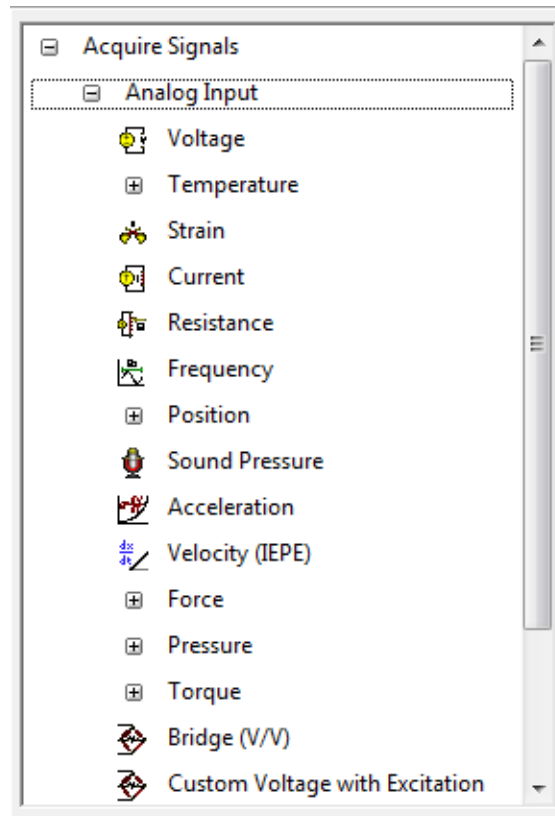


Figure 1.37: Analog Signals

- Counter input:

You can select the proper counter input based on the measurement you are making provided that your **Data Acquisition Card** supports that type of measurement. Counter input maybe used with encoders.

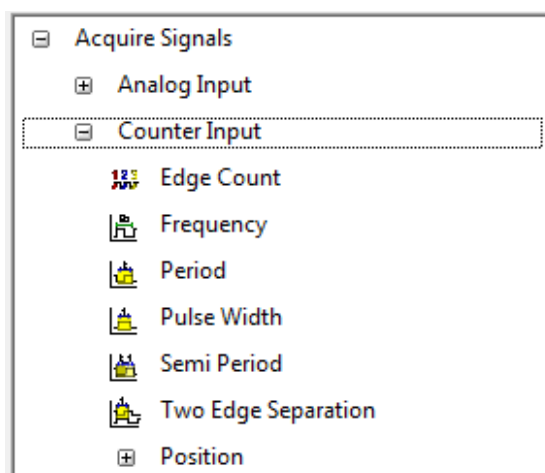


Figure 1.38: Counter signals

- Digital inputs

You can select the proper digital input based on the measurement you are making provided that your **Data Acquisition Card** supports that type of measurement. Digital inputs have only two types: either configure the DAQ as a port where all 8 pins are dealt with as one single input or you can deal lines where each pin is used as a digital input.

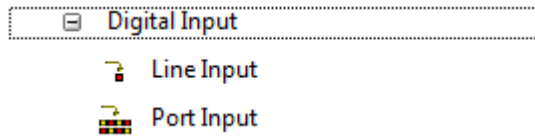


Figure 1.39: Digital Signals

The following inputs are obtained when configuring an analog voltage input:

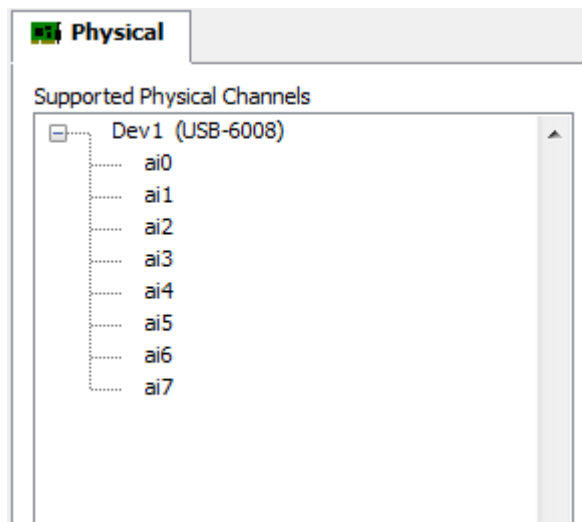


Figure 1.40: Analog Input Voltage Channels

Select the channel your hardware connected to from ai0 to ai7 and click **Finish**.

1. The following dialog box will open where you have to fill the following:
 - **Max:** maximum value of the input voltage you are reading as long as it doesn't exceed the maximum input voltage that can be read by the DAQ.
 - **Min:** minimum value of the input voltage you are reading as long as it doesn't exceed the minimum input voltage that can be read by the DAQ.
 - **Terminal configuration:** select "Differential" for differential inputs and "RSE" for single ended inputs.

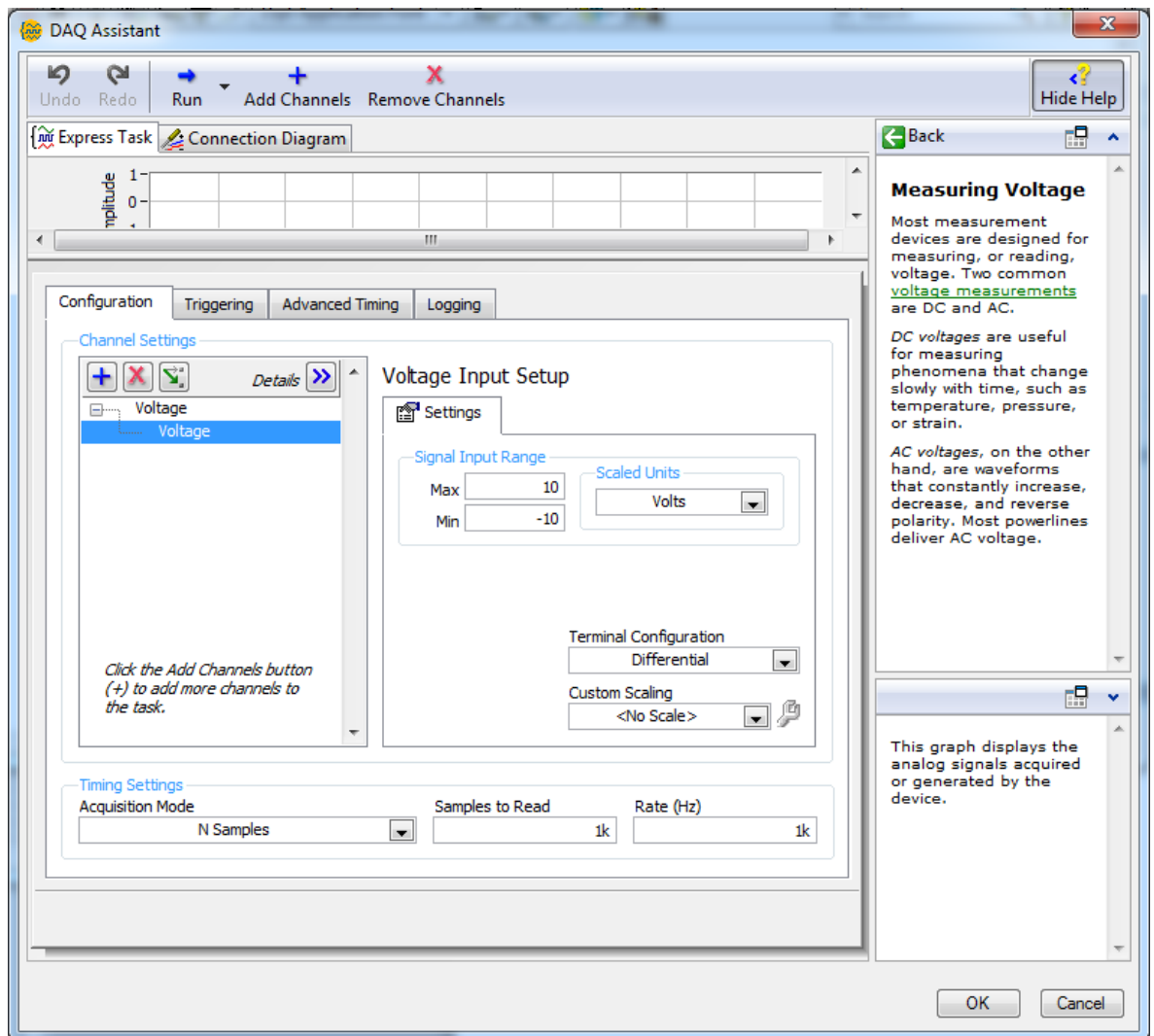


Figure 1.41: Configuring Analog Input Voltage

- **Acquisition mode:** it can be "one sample" where the DAQ reads only one sample and the update rate depends on the timing of the program. You can select "N samples" where a predefined number of samples "N" is read, then the program is updated with all of the readings at once. To read another "N" samples, you have to put your DAQ inside a loop. "Continuous samples" is equivalent to "N sample" inside a loop.
- **Samples to read (N):** what is the total number of samples you want your program to get updated with after each reading is finished.
- **Rate (Hz):** how frequent you want your samples to get read. Its unit is S/s (sample per second). i.e. during 1sec how many samples you want read.

How to calculate the time needed to get an update in the program (total time needed to read all the samples)?

This time is calculated as follows:

Time=N/Hz

If you want to read 1000 sample and your rate is 1KHz then you will get an update each 1sec with 1000 sample read.

If you want to read 100 sample and your rate is 1KHz then you will get an update each 0.1 sec with 100 sample read.

Notes:

- **The Express Task at the top of the dialog box is used to test the acquired or the generated signal before finishing the configuration.**
- **Dealing with digital inputs is not much different but will less options.**

3. Generating signal

Generating signals is not much different except that it has less options since mostly the actuators has much less operating signals that the variety of input signals that can be generated by measurements.

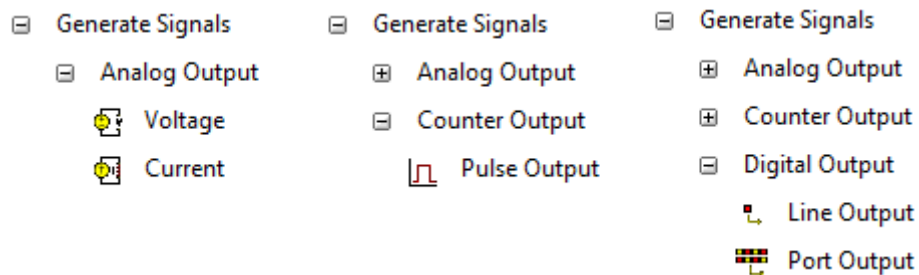


Figure 1.42: Generating Output Signals

It is recommended to have 6 DAQ at max. one for all the analog inputs, one for all the analog outputs, one for all the digital inputs, one for all the digital outputs, one for all the counter inputs and one for all the counter outputs.

When acquiring signals from or generating signals to the actual world, these signals are called dynamic data type. Ordinary indicators and controls can't deal with such data type. You will need the following convertors:

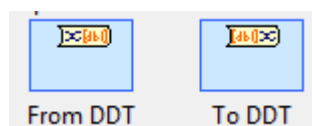


Figure 1.43: Dynamic Data Type Converters

To obtain them, right click the block diagram the select **Express** from the drop down menu >> **signal manipulation**. The following dialog box will open when you insert DDT:

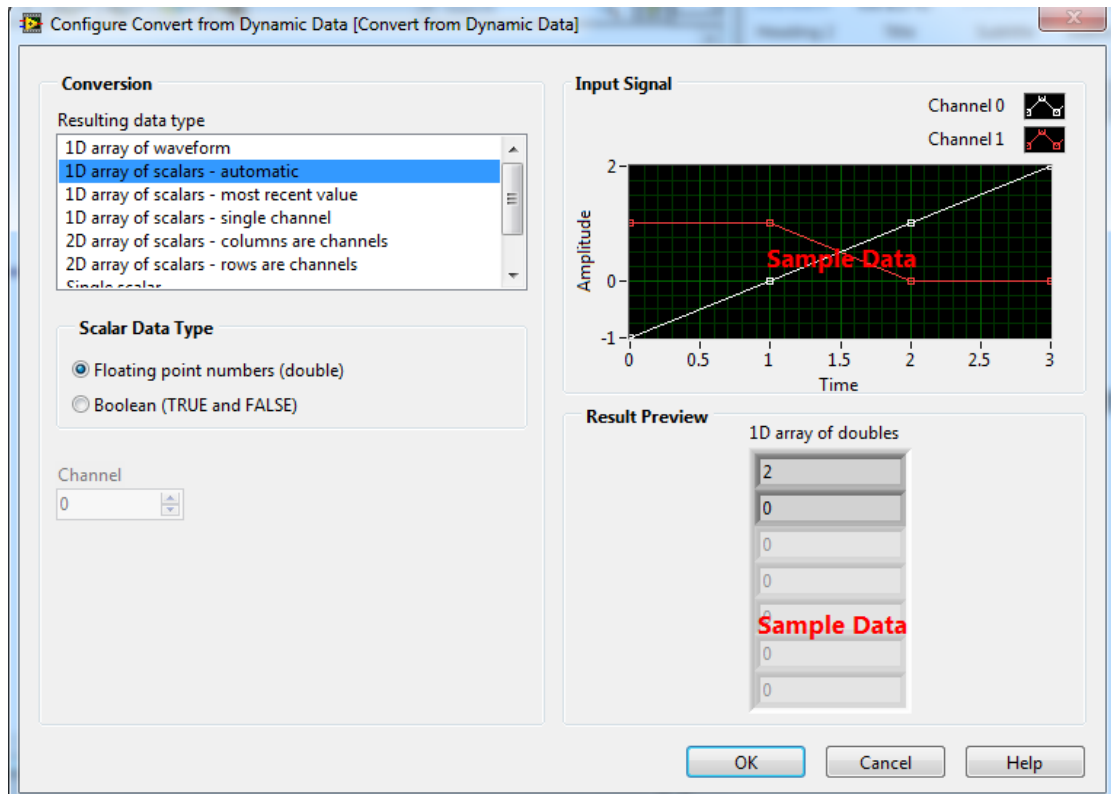


Figure 1.44: Configuring "From DDT"

Floating is selected when dealing with analog data while Boolean is selected for digital data. The **Resulting data type** is selected to meet your demand mostly for the lab application it is a **Single scalar**.

5 Procedure

1. Design the LabVIEW program to meet the requirements of the exercise provided by your lab supervisor.
2. Run and test your program.
3. Save it with the proper name on the proper directory.

6 References:

- 1- <http://www.ni.com>
- 2- LabVIEW™ Core 1 Course Manual

Experiment 2

Data Acquisition of Digital Signals

1 Objectives

- To be familiar with the concept of Data Acquisition Systems.
- To be familiar with the digital data acquisition systems and their interface circuits.
- To connect digital data acquisition system using USB 6008.
- To control the system in LabVIEW environment.

2 Apparatus

1. Data Acquisition Cards (DAQ) USB 6008 DAQ in Figure 1.
2. Digital devices (LED, push buttons, etc. ...).
3. Miscellaneous (breadboard resistors, wires, etc. ...).
4. PC (with LabView and Device Driver).

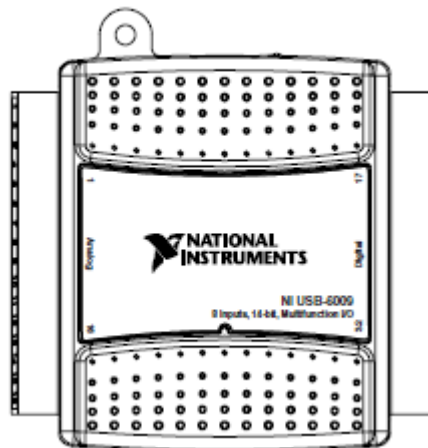


Figure 1: USB 6008 DAQ

3 Introduction

Data acquisition (DAQ) is the process of measuring an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound with a computer. A DAQ system consists of sensors, DAQ measurement hardware, and a computer with programmable software. Compared to traditional measurement systems, PC-based DAQ systems exploit the processing power, productivity, display, and connectivity capabilities of industry-standard computers providing a more powerful, flexible, and cost-effective measurement solution.

Data acquisition applications are usually controlled by software programs developed using various general purpose programming languages such as Assembly, BASIC, Fortran, LabVIEW, Java, ...etc. Stand-alone data acquisition systems are often called data loggers.

A complete data acquisition system as in Figure 2 consists of:

- Sensors and Actuators (Plant).
- Signal Conditioning circuit (which in modern systems is integrated within the DAQ hardware).
- DAQ Measurement Hardware (Interface Devices).
- DAQ Software and Device Driver.

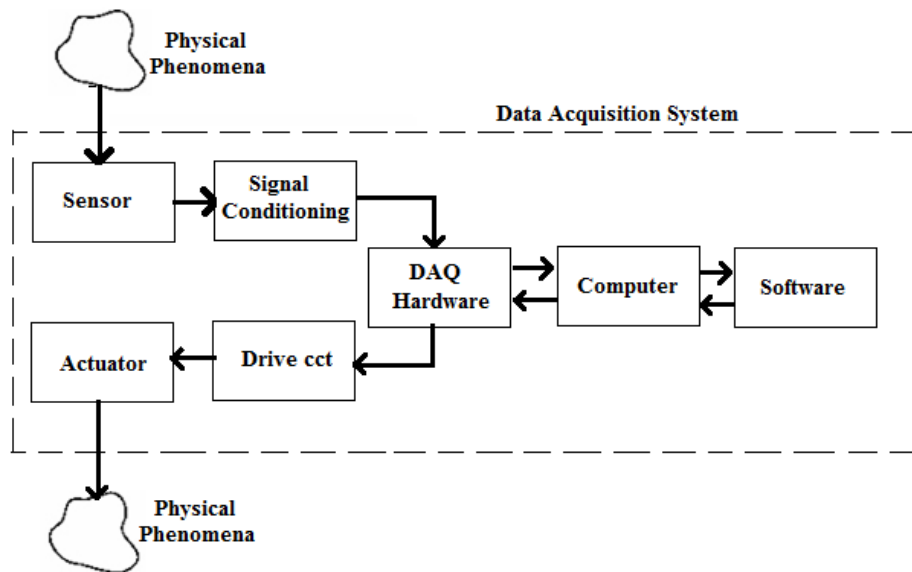


Figure 2: Data Acquisition System

This experiment focuses on data acquiring of digital signal.

4 Digital input/ Output Acquisition:

4.1 Digital input/output sources

Digital inputs include many devices such as selector switches, pushbuttons, limit switches, circuit breakers, proximity switches, level switches, motor starter contacts, Relay contacts, etc.... and digital outputs include alarms, LEDs, buzzers, digital control signal, control relays, motor starters, solenoids, etc....

4.2 Digital I/O Ports

A digital I/O board is an interface board that adds the ability to input and output digital signals in parallel to a computer as shown in. Using a digital I/O device makes it possible to

- read the statuses of input devices
- operate output devices and drive circuits

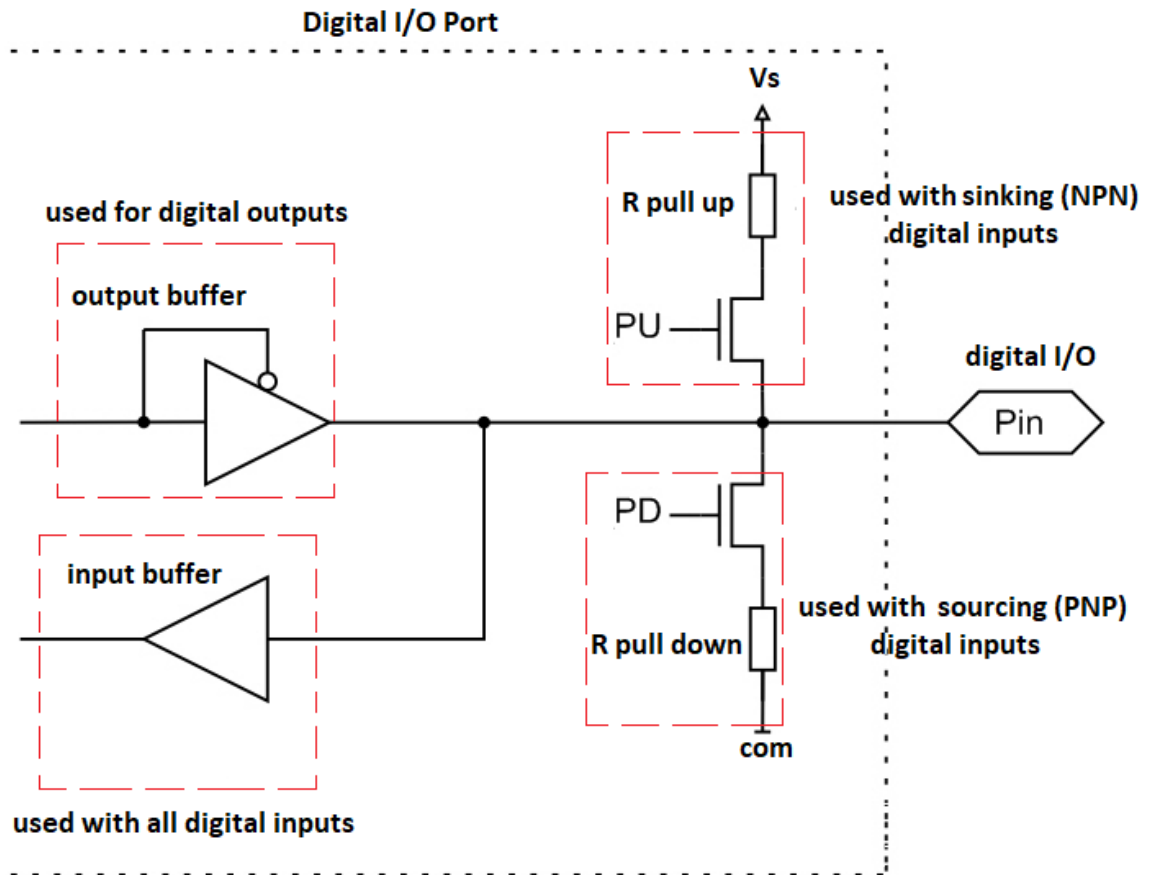


Figure 3: Digital I/O Port

- When the digital lines are configured as digital outputs "DO", output transistors are enabled, and input buffers are disabled.
- When the digital lines are configured as digital inputs "DI", input buffers are enabled and output transistors are disabled.

4.3 Digital Input Interface

There are two different configurations of digital input interface circuits. Both are presented in Figure 4.

1. A resistor, known as a pull-up resistor, is connected between the output and the supply voltage, When the switch is open, the output signal is at logic "1". This configuration is known as "Active Low"
2. A resistor, known as a pull-down resistor, is connected between the output and the 0V ground, When the switch is open, the output signal is at 0V. This configuration is known as "Active High"

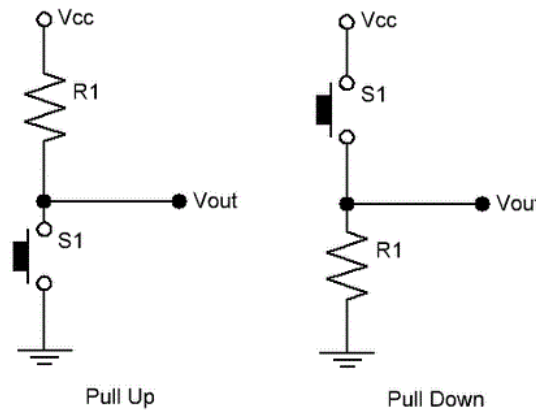


Figure 4: Digital Input Interface Circuits

The values of the pull up/down resistors are calculated as below

$$R_{pull-up} = \frac{V_{supply} - V_{H(min)}}{I_{sink}} \quad R_{pull-down} = \frac{V_{L(max)} - 0}{I_{source}}$$

Where;

$V_{H(min)}$: is minimum voltage considered as high level

$V_{L(max)}$: is maximum voltage considered as low level

I_{source} : is the maximum current can be sourced from the digital port

I_{sink} : is the maximum current can be sinked into the digital port

4.3.1 Types of Digital Input ports

- Sometimes, the pull-up or pull-down resistors are provided internally within the digital input port, in such cases no need to connect them externally. If the digital input port is provided with a internal pull-up resistor it is called **sourcing digital input port**. If the digital input port is provided with a internal pull-down resistor it is called **sinking digital input port**. Refer to Figure 5

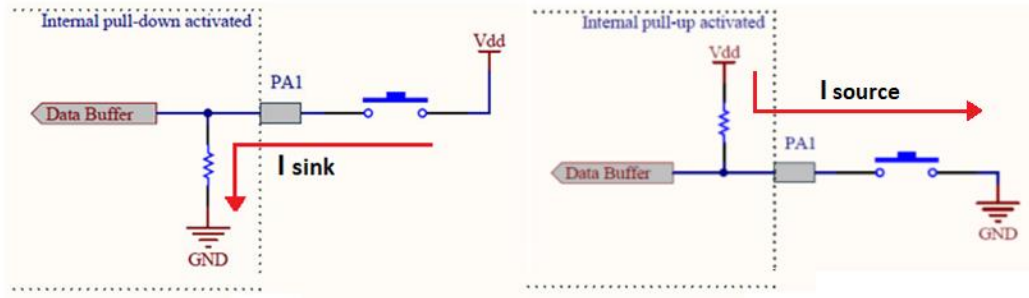


Figure 5: Sinking and Sourcing Digital Input Ports

4.3.2 Types of Digital Input Signals

Digital inputs are divided into two categories:

1. 2-wire switches:

- Such as mechanical switches, push buttons relays. This type of digital input devices can be connected in either configuration (pull-up or pull down) they also can be connected to either sinking or sourcing DI ports

2. 3-wire switches:

such as proximity switches and other solid-state switches. This type of digital input devices are either

- Sourcing-output (PNP) sensors Figure 6
- Sinking-output (NPN) sensors Figure 7

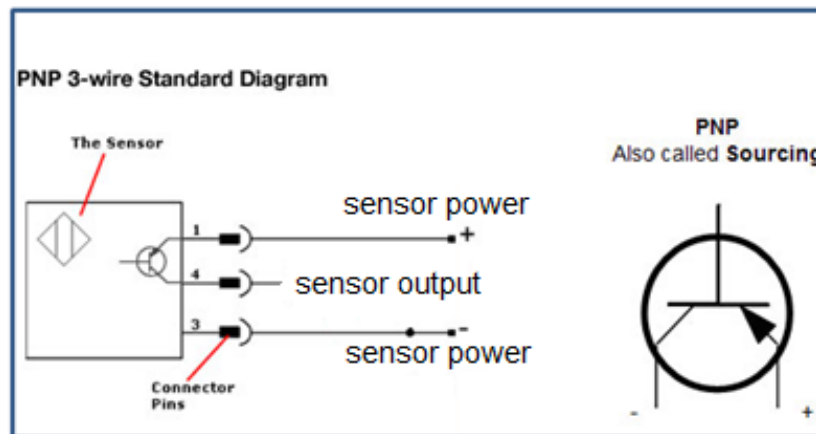


Figure 6: Sourcing-Output (PNP) Switch

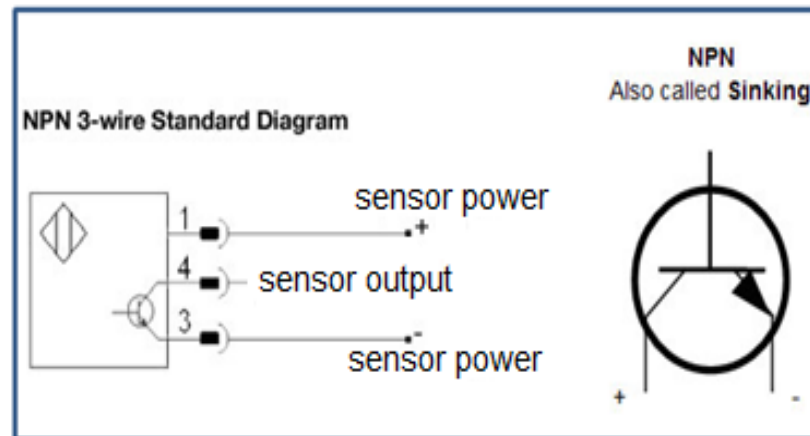


Figure 7: Sinking-Output (NPN) Switch

It is very important to know that sourcing-output (PNP) digital devices inputs **must** be connected in a **pull- down** configuration or with a **sinking DI port** while sinking-output (NPN) digital signal inputs **must** be connected in a **pull- up** configuration or with a **sourcing DI port** as demonstrated by Figure 8.

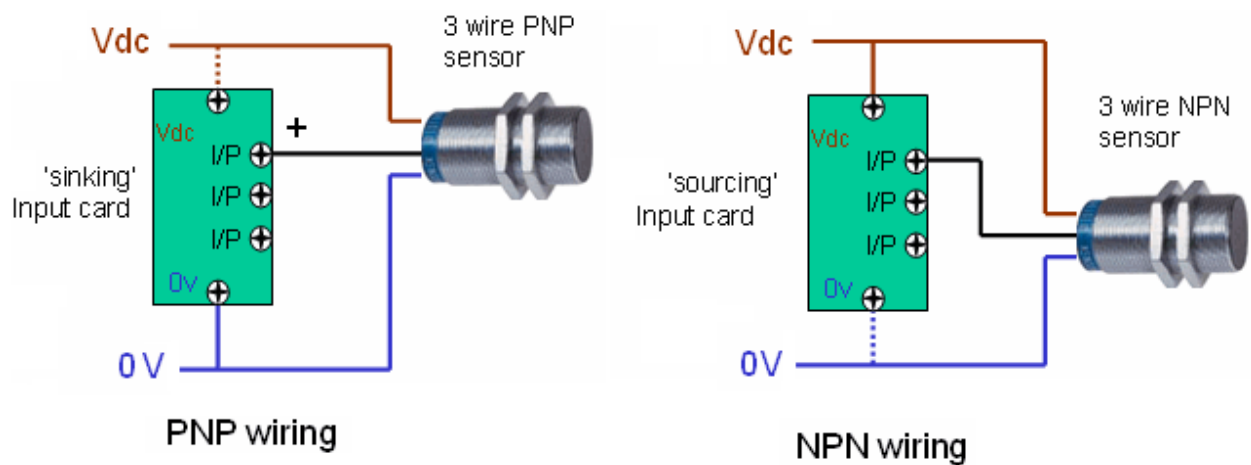


Figure 8: sinking-output (NPN) switch

4.4 Digital Output Interface

Digital output devices can be connected to a digital output port in either sinking or sourcing configuration, depending on the digital output port internal configuration. There are three different configurations of digital output ports circuits.

1. **Line Driver:** A line driver is a sourcing output. When in the on state, a line driver will supply V_{cc} . In the off state, a line driver will float. For proper operation, the external circuit **must** be **sinking**.
2. **Open Collector:** An open collector is a sinking output. In the on state, an open collector will supply a path to ground. When in the off state, an open collector will float. For proper operation, the external circuit **must** be **sourcing**.
3. **Push-Pull:** A push-pull output is a combination of a line driver and an open collector. In the off state it will supply a path to ground and in the on state it will supply V_{cc} . The external circuit can be either **sinking or sourcing**.

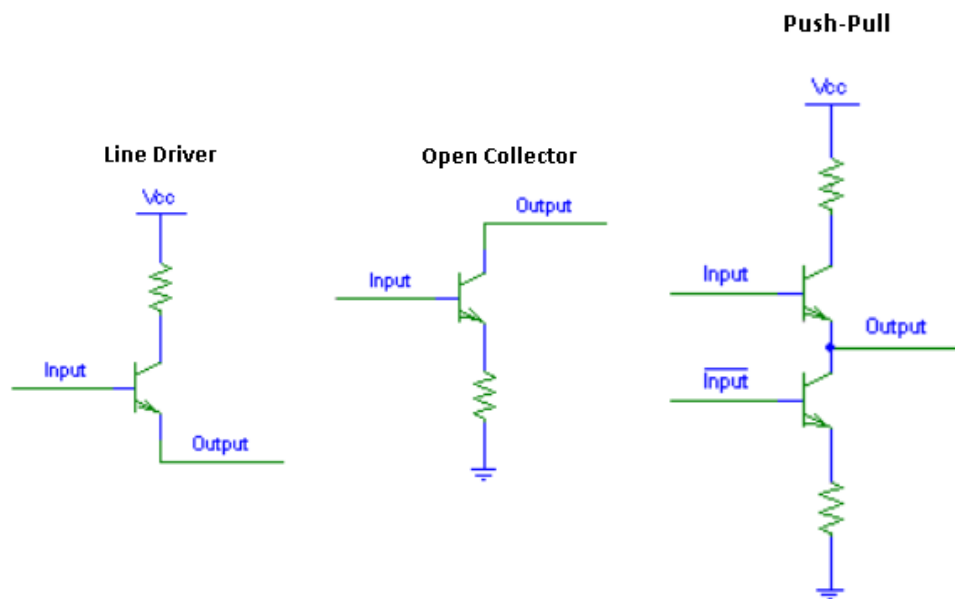


Figure 9: Internal Configuration of Digital Output Ports

Figure 10 explains the correct way to connect a digital load to different types of DO port.

Notice that in the case of open collector DO port the logic of the system is inverted, i.e. when the DO is HIGH, the load is OFF, and when the DO is LOW, the load will be ON. This issue doesn't affect the functionality of the plant, but sometimes it may be a source of confusion. If the situation needs to be re-inverted back to the normal convention (High DO causes the load to be ON) software and hardware solutions are possible. The software solution is to invert the DO within the code. And the hardware solution is illustrated by Figure 11 using an external pull up resistance and power supply.

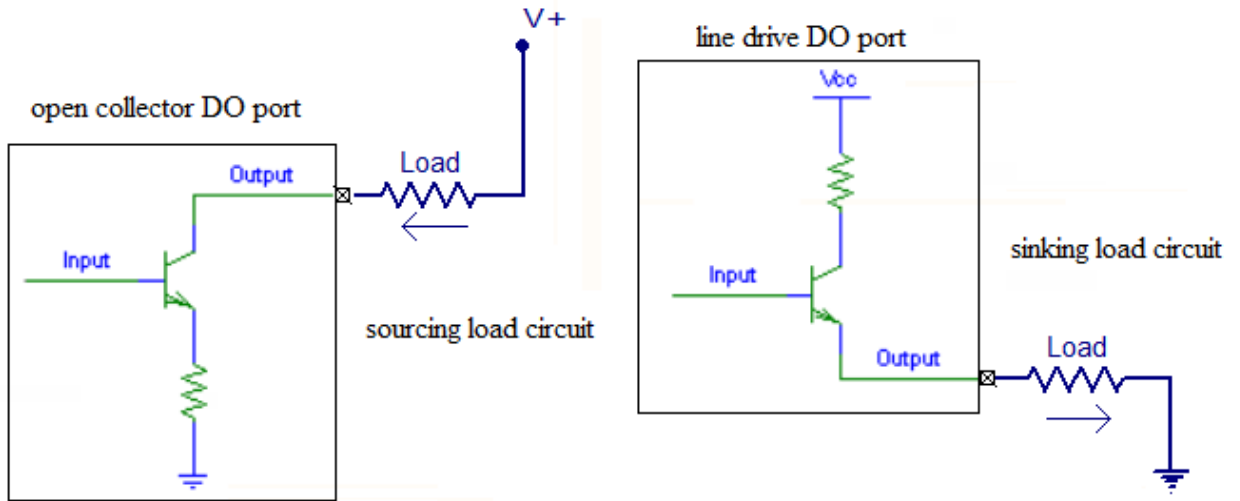


Figure 10: Sourcing and Sinking Load Circuits

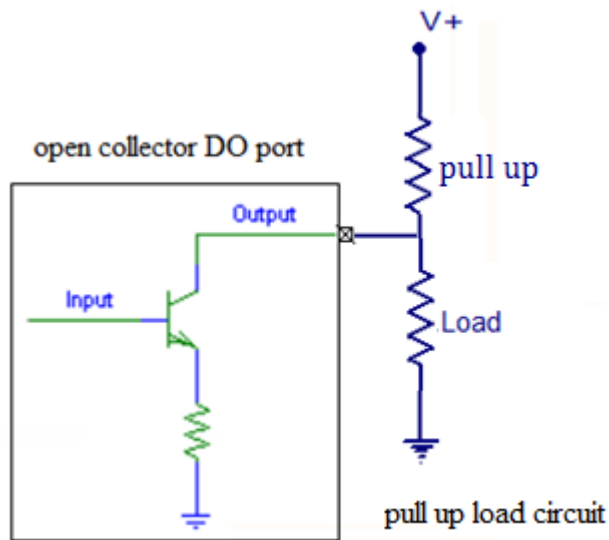


Figure 11: Open Collector DO Port with Pull Up Load Circuits

The value of the pull up resistor is calculated as follow

$$R_{pull\ up} = \frac{V^+ - V_{min}}{I_{pull\ up}}$$

$$I_{pull\ up} = I_{load\ max} + I_{DO\ max}$$

Where;

V_{min} : is the minimum voltage of the DO port

V^+ : is the external supply voltage.

$I_{load\ max}$: is the load rated current

$I_{DO\ max}$: is the maximum current sunk by the DO port.

The current of the load connected to a DO port must not exceed the DO port current, otherwise a drive circuit with external power supply must be implemented to protect the DO port of high load currents. Various drive circuits are available, such as, transistors (in the saturation mode), optocouplers (opto-isolators), etc.... See Figure 12.

Mechanical relays can also be used if the coil current does not exceed the DO port current, which is the case. See Figure 13.

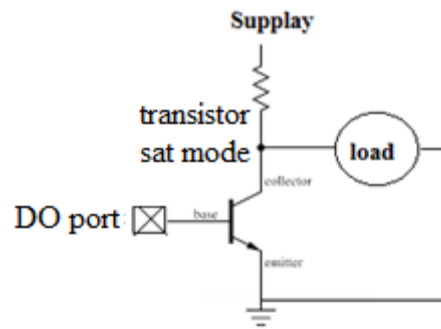


Figure 12: Connecting Load to Digital Output Ports with Transistor Drive Circuit

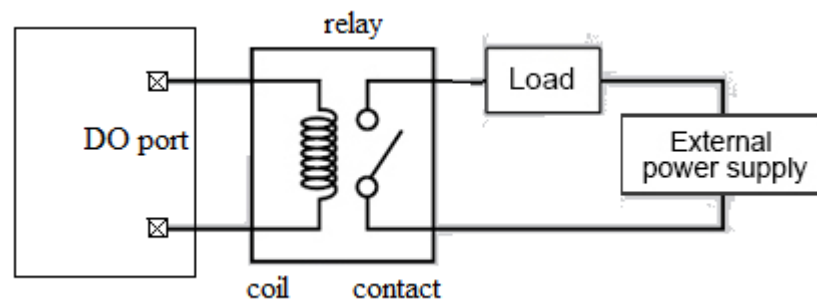


Figure 13: Connecting Load to Digital Output Ports with Relay Drive Circuit

4.5 Counter/ trigger ports

Counter/ trigger ports are basically dealt with as any digital port. They provide faster switching frequency than other digital ports. They can either count input digital edges or trigger output square wave pulse.

5 USB 6008 DAQ Digital Ports Specifications

The National Instruments USB-6008 is a low-cost, multifunction data acquisition device (DAQ). It has 8 analog inputs, 2 analog outputs, and 12 digital input/outputs and one counter port. The digital channels are divided into two ports P0.<0..7> and P1.<0..3>, when one or more channels on each port is set to either input or output, the port is locked into that particular mode, you can individually program all lines as inputs or outputs.. GND is the ground-reference signal for the DIO port. The pinout of the DAQ is presented by Figure 14.

NI USB-6008

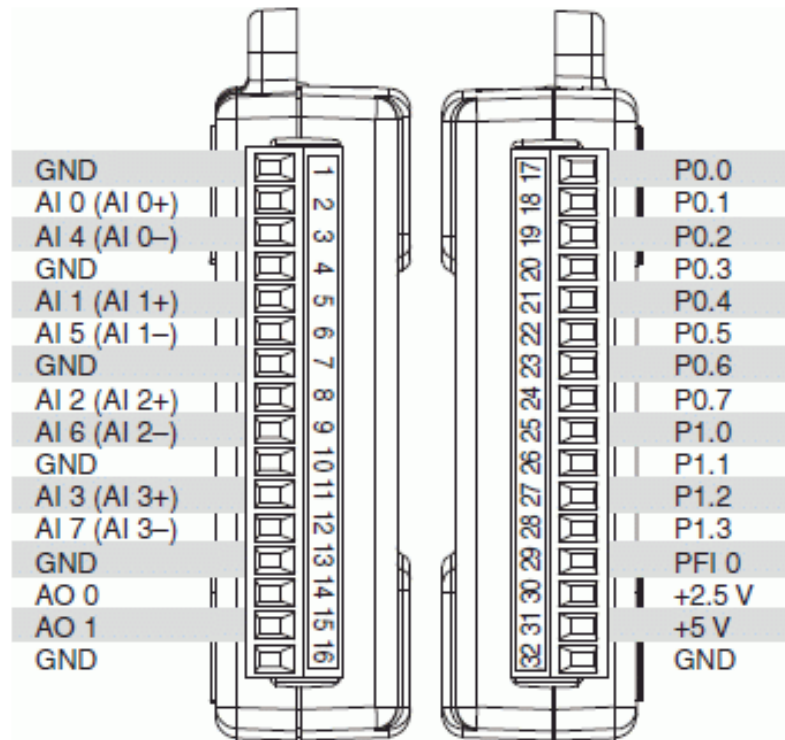


Figure 14: USB 6008 DAQ Pinout

As described by the data sheet, the DAQ supply ports, the digital I/O ports and the counter port have specifications described next. For more specifications refer to the complete data sheet available online.

5.1 Power Supply

The DAQ has two power sources according to the data sheet:

+5 V output (200 mA maximum) +5 V typical, +4.85 V minimum

+2.5 V output (1 mA maximum) +2.5 V typical

Very low current; the supply is used for self-test only

5.2 Digital I/O

P0.<0..7>..... **8 lines**

PI.<0..3> **4 lines**

Direction control **Each channel individually programmable as input or output**

Output driver type

USB-6008 **Open-drain**

Same as open collector

Compatibility TTL, LVTTTL, CMOS

Absolute maximum voltage range -0.5 to 5.8 V with respect to GND

Pull-up resistor 4.7 kΩ to 5 V **Internal**

Power-on state..... **Input (high impedance)**

Table 1: Signal Descriptions

Port Type	Level	Min	Max	Units
Input	Input low voltage	-0.3	(0.8)	V
	Input high voltage	2.0	5.8	V
	Input leakage current	—	50	μA
Output	Output low voltage (I = 8.5 mA)	—	0.8	V
	Output high voltage			
	Open-drain, I = -0.6mA, nominal	2.0	5.0	V
	Open-drain, I = -8.5mA, with external pull-up resistor	(2.0)	—	V

Annotations:
 - $V_{H(min)}$ points to the Min column for Input high voltage (2.0).
 - $V_{L(max)}$ points to the Max column for Input low voltage (0.8).
 - V_{min} points to the Max column for Output high voltage.
 - A callout for the -8.5mA row states: "Current through 4.7KΩ onboard resistor. Without external pull up".
 - A callout for the -8.5mA row states: "I_{DO max}. The negative sign indicates the sinking nature of the".

5.3 Counter

Number of counters.....1

Resolution32 bits

Counter measurementsEdge counting (falling-edge)

Pull-up resistor.....4.7 kΩ to 5 V

Maximum input frequency5 MHz

Notice that this DAQ does not trigger an output pulse.

6 Procedure

1. Connect the digital inputs and outputs provided by your instructor properly.

2. Show the needed calculations.
3. Design a LabVIEW program to meet the requirements provided by your instructor.
4. Run and test your program and your hardware.
5. Save it with the proper name on the proper directory.

7 References:

- 1- <http://www.ni.com>
- 2- <https://www.contec.com/support/basic-knowledge/daq-control/digital-io/>.
- 3- <https://www.electronics-tutorials.ws/io/input-interfacing-circuits.html>
- 4- <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019N8KSAU&l=en-JO>
- 5- <http://www.handsonembedded.com/stm32f103-spl-tutorial-3>

Experiment 3

Data Acquisition of Analog Signals

1 Objectives

- To be familiar with the concept of Data Acquisition Systems.
- To be familiar with the analog data acquisition systems and their interface circuits.
- To connect analog data acquisition system using USB 6008.
- To control the system in LabVIEW environment.

2 Apparatus

1. Data Acquisition Cards (DAQ) USB 6008 DAQ in Figure 1.
2. analog devices (motor, transistors, potentiometers, etc. ...).
3. Miscellaneous (breadboard resistors, wires, etc. ...).
4. PC (with LabView and Device Driver).

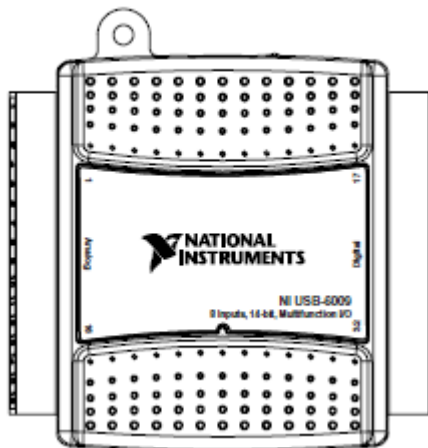


Figure 1: USB 6008 DAQ

3 Introduction

The signals from sensors that measure surrounding natural factors such as temperature, pressure, and flow rate are often analog signals, and most control actuators move according to analog signals. On the other hand, only digital signals can be handled by computers. For this reason, in order to input a signal from a sensor using a computer, or to output a signal to an actuator, it's necessary to have a device that can bridge the analog signal and the digital signal handled by the computer. That bridge is called an analog I/O interface, or analog I/O device.

4 Analog Input/ Output Acquisition:

Analog I/O devices are:

1. Analog input device (AI)

This device is responsible for converting analog signals from external devices to digital signals that can be processed by a computer, using an internal analog to digital convertor (ADC).

2. Analog output device (AO)

This device converts the digital data from a computer to an analog signal before outputting that signal to an external device, using an internal digital to analog convertor (DAC).

3. Analog I/O device

Analog I/O devices are devices with both an ADC function and a DAC conversion function. They are programmatically set as either AI or AO.

4.1 Analog Input Devices and Interface.

Figure 2 presents the internal circuitry of a standard AI device (it may differ depending on the manufacturer of the device). One ADC is used in converting the analog input of multiple AI channels. A multiplexer is used to switch between the various channels, and a PGA (Programmable Gain Amplifier) is used to change the input range of the AI channels to enhance its resolution. Furthermore, almost all AI devices are electrically isolated via a photocouplers (not shown in Figure 2)., this insulation provides the internal circuit with protection from outside current surge.

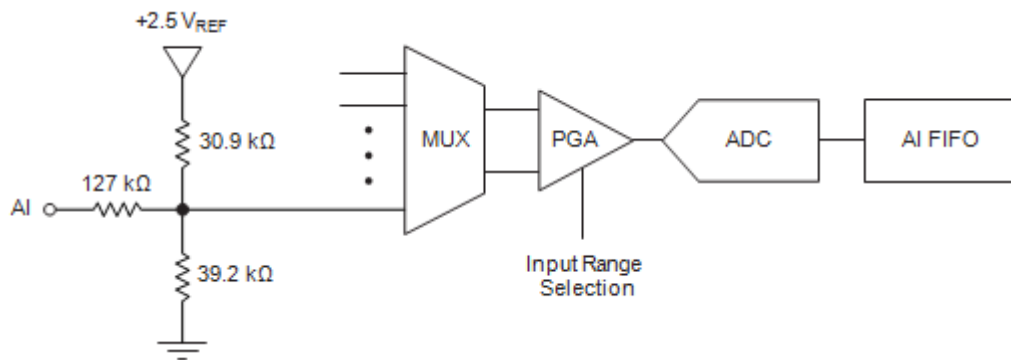


Figure 2: Analog Input Circuitry

Both the ADC and PGA decide on the resolution of the input signal

$$AI \text{ Resolution (step size)} = \frac{\Delta V_{in}}{2^n - 1}$$

Where;

n: is the number of bits in the ADC.

ΔV_{in} is the input voltage range, of the input signal.

Setting the PGA gain to have the same range as the input signal ΔV_{in} will enhance the ADC resolution.

When making connection to measure analog signal using a DAQ, considered two things:

1. Transducer output signal (signal source).
2. DAQ channel type (measurement system).

Transducer output signal falls into one of the following categories:

1. Current input:
 - a. Differential current (current loop) transducers.
 - b. Single-ended (3-wire) current transducers
2. Voltage input:
 - a. Differential voltage transducers.
 - b. Single-ended voltage transducers.

4.2 Current Input Measurement

A Current measurement channel must be available in the DAQ in order to measure current signal. If this is the case, both types of current transducers are connected as follow:

- c. Differential current (current loop) measurement is presented in
- d. Single-ended (3-wire) current measurement.

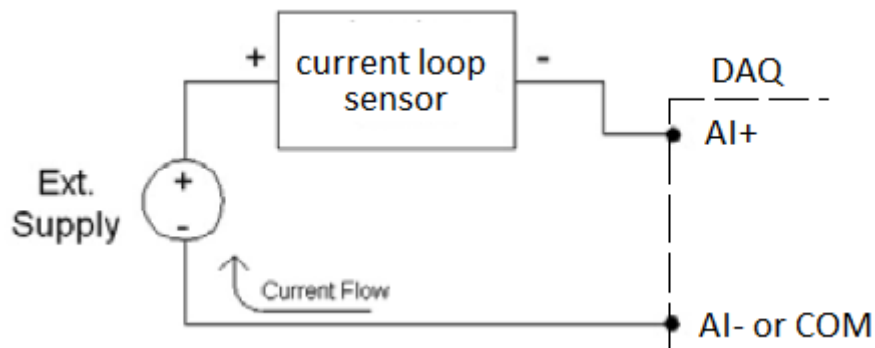


Figure 3: Current Loop Measurement Connection

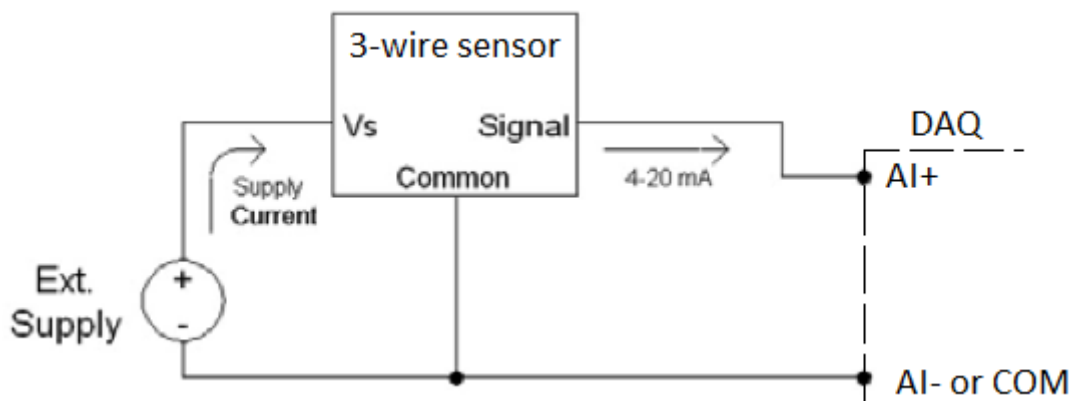


Figure 4: 3-Wire Current Measurement Connection

DAQs do not measure current, they measure voltage, but using an internal shunt resistor (Figure 5) the input current signal is represented as voltage signal measured by the DAQ.

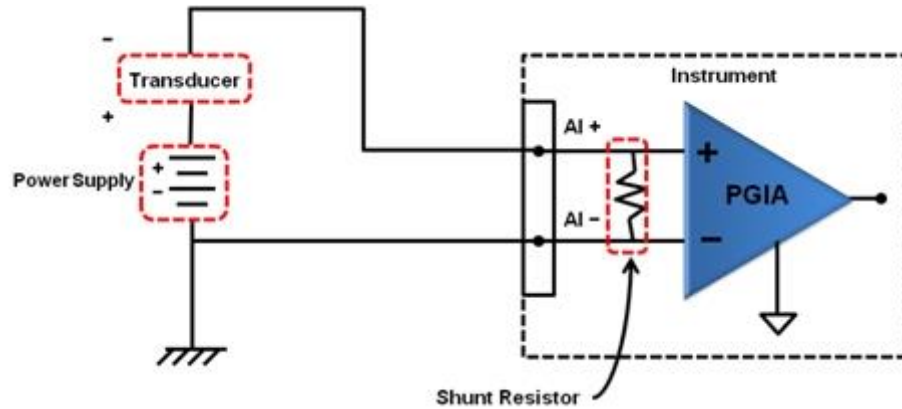


Figure 5: Internal Circuitry of Current Measurement Channel

4.3 Voltage Input Measurement

As mentioned before, the transducer output signal (signal source) and the DAQ channel type (measurement system) important factors proper AI interface including measuring a voltage signal.

A signal source can be grouped into one of two categories; grounded or ungrounded (floating). Similarly, a measurement system can be grouped into one of two categories; grounded (ground-referenced), and ungrounded (floating).

4.3.1 Transducer Output Signal (Signal Source)

1. Grounded Signal Source:

- a. **Single-ended signal source:** The voltage is measured between two terminals, one of them is the system ground.

Examples: NPN sensors (such as proximity), outputs of amplifier circuits and outputs of voltage divider circuits and any common plug-in instrument

- b. **Differential grounded signals:** The voltage is measured between two terminals, none of the them is the system ground, but there is third ground terminal.

Examples: outputs of bridge circuits, thermocouple and strain gauges.

2. Floating Signal Source:

The voltage is measured between two terminals none of which is referred to the system ground.

Examples: transformers, thermocouples, battery-powered devices, optical isolators, and isolation amplifiers

4.3.2 DAQ Channel Type (Measurement System)

1. Differential Measurement System:

Also known as nonreferenced or floating measurement system. As presented in Figure 6 the potential difference is measured between its two terminals; the (+) and (-) inputs neither of tied the ground.

Hand-held, battery-powered instruments and data acquisition devices with instrumentation amplifiers are examples of differential or nonreferenced measurement systems

Each terminal -the (+) and (-) inputs- has analog multiplexers to increase the number of measurement channels while still using a single instrumentation amplifier

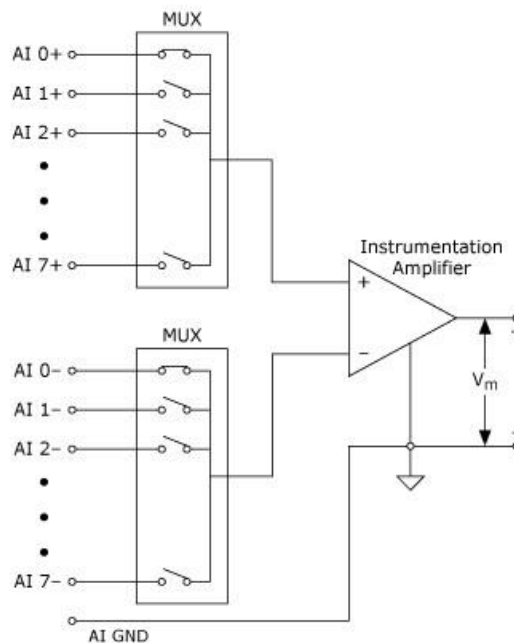


Figure 6: A Differential Measurement System

2. Grounded Measurement System:

Also known as ground-referenced measurement system or as commonly known Single-Ended measurement system. The measurement is made with respect to common node ground. And it has two types:

- a. ground-referenced Single-ended (RSE): it is clear by Figure 7 that all input channels are measure with respect the system ground, i.e. the (-) input of the instrumentational amplifier is connected to the system ground.
- b. none-referenced Single-ended (NRSE): all measurements in Figure 8 are still made with respect to a single-node Analog Input Sense (AI SENSE), but the potential at this node can vary with respect to the measurement system ground. NRSE measurement system is the same as a single-channel differential measurement system.

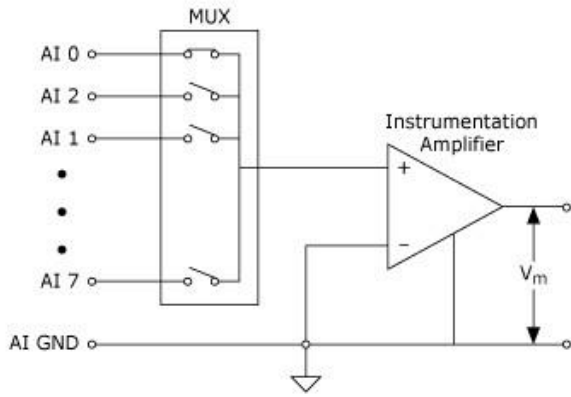


Figure 7: RSE Measurement System

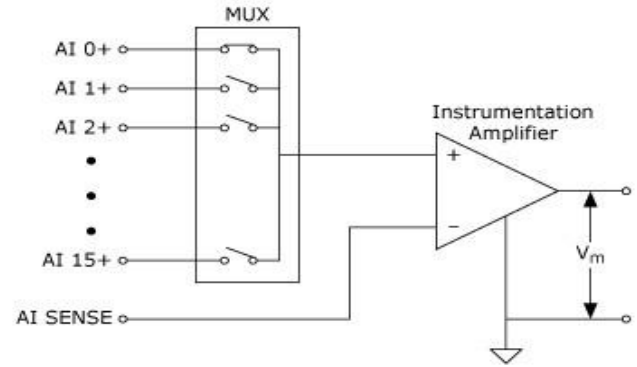


Figure 8: NRSE Measurement System

4.3.3 The Proper Measurement System for Each Type of Signal Source

1. A **grounded signal source** is best measured with a differential or nonreferenced measurement system.

Single-Ended-Ground Referenced (RSE) is not recommended when using a grounded signal source! Figure 9 shows that by using RSE system to measure a grounded signal source an **undesired** ground loops are created.

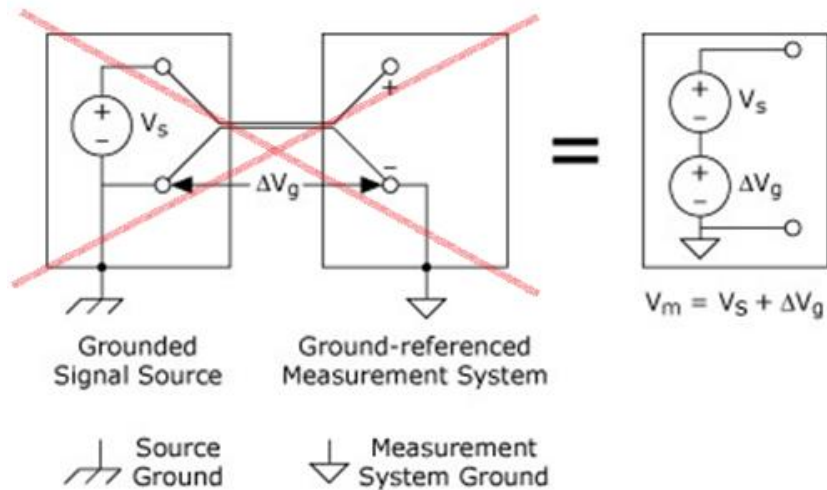


Figure 9: Unrecommended AI Voltage Interface

2. **Floating signal sources** can be measured with both differential and single-ended measurement systems as presented in Figure 10.a and Figure 10.b , respectively.

Bias resistors must be provided when measuring floating signal sources in DIFF and NRSE configurations, to reduce the common-mode voltage level of the signal and keep

it in the range of the common-mode input the measurement device. Failure to use these resistors will result in erratic or saturated (positive full-scale or negative full-scale) readings.

If the input signal is DC-coupled, only one resistor connected from the (-) input to the measurement system ground is required. Bias resistors are in the range of 10 k Ω -100 k Ω .

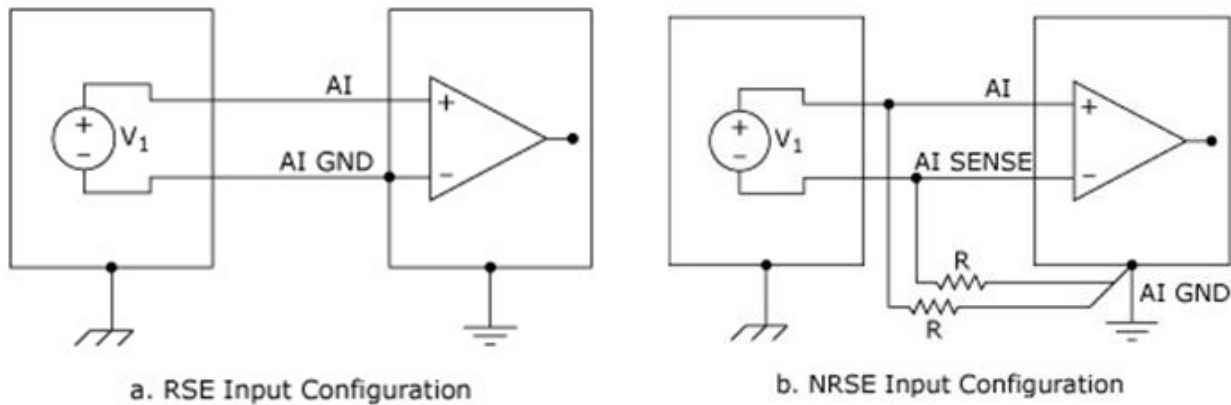


Figure 10: Floating Signal Source Interface with Single-Ended Configurations

3. A differential measurement system is preferable over the single-ended configurations, because it rejects ground loop-induced errors and it rejects the noise picked up in the environment to a certain degree.
4. The single-ended configurations provide twice as many measurement channels. It is justified only if the magnitude of the induced errors is smaller than the required accuracy of the data.
5. Single-ended input connections can be used when all input signals meet the following criteria:
 - Input signals are high level (greater than 1 V)
 - Signal cabling is short and travels through a noise-free environment or is properly shielded
 - All input signals can share a common reference signal at the source

Differential connections should be used when any of the above criteria are violated.

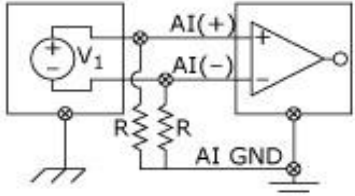
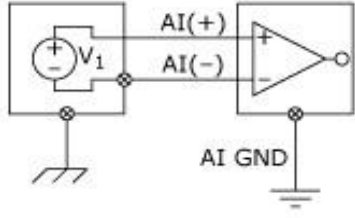
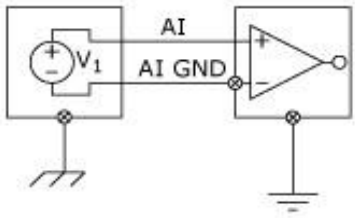
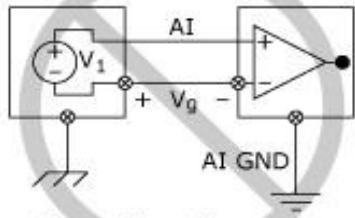
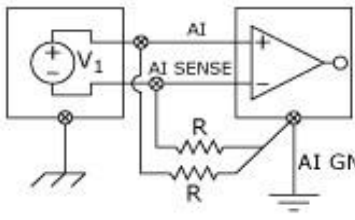
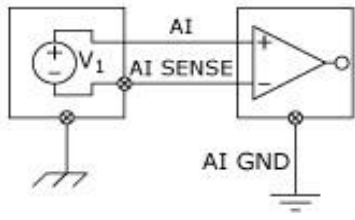
Input Configuration	Signal Source Type	
	Floating Signal Source (Not Connected to Building Ground)	Grounded Signal Source
	Examples <ul style="list-style-type: none"> • Thermocouples • Signal Conditioning with Isolated Outputs • Battery Devices 	Examples <ul style="list-style-type: none"> • Plug-in Instruments with Nonisolated Inputs
Differential (DIFF)	 <p>Two resistors ($10\text{ k}\Omega < R < 100\text{ k}\Omega$) provide return paths to ground for bias currents</p>	
Single-Ended - Ground Referenced (RSE)		<p>NOT RECOMMENDED</p>  <p>Ground-loop losses, V_g, are added to measured signal.</p>
Single-Ended - Nonreferenced (NRSE)		

Figure 11 summarizes the proper interface of AI signal source.

4.4 Analog Input Signal conditioning

Signal conditioning is one of the most important components of a data acquisition system, because without optimizing real-world signals for the digitizer in use, you cannot rely on the accuracy of the measurements. Common signal conditioning stages are:

- Amplification
- Attenuation
- Filtering
- Isolation
- Excitation
- Linearization

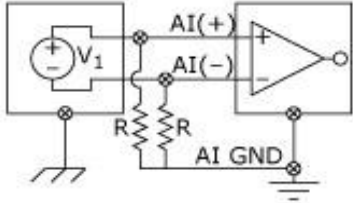
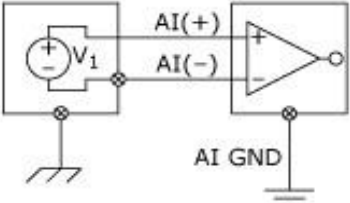
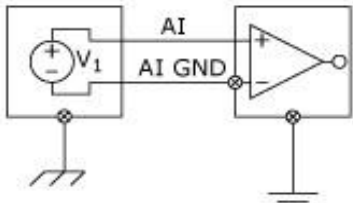
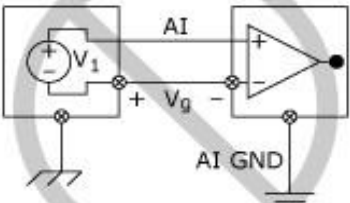
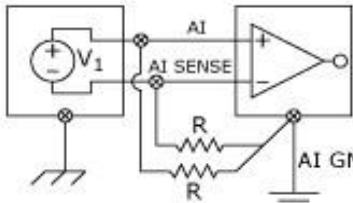
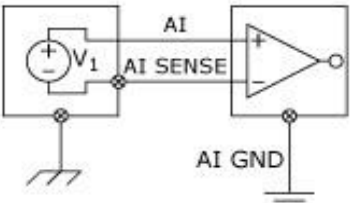
Input Configuration	Signal Source Type	
	Floating Signal Source (Not Connected to Building Ground)	Grounded Signal Source
	Examples <ul style="list-style-type: none"> • Thermocouples • Signal Conditioning with Isolated Outputs • Battery Devices 	Examples <ul style="list-style-type: none"> • Plug-in Instruments with Nonisolated Inputs
Differential (DIFF)	 <p>Two resistors ($10\text{ k}\Omega < R < 100\text{ k}\Omega$) provide return paths to ground for bias currents</p>	
Single-Ended - Ground Referenced (RSE)		<p>NOT RECOMMENDED</p>  <p>Ground-loop losses, V_g, are added to measured signal.</p>
Single-Ended - Nonreferenced (NRSE)		

Figure 11: Possible Interface of AI Voltage Signal Source

4.5 Analog Output Devices and Interface

Connecting analog outs (AO) is mostly a simple process. You connect the output pin to the output device and the other terminal of the device to the analog GND, as illustrated by Figure 12

Most of the times, the loads current exceeds the maximum current of the DAQ, and other times, the level of the voltage that operates the output device doesn't match the range of the DAQ signal. In these cases, some interfacing such as current limiting resistors or drive circuits with external power supply are required. Some example of drive circuit such as Power transistors, H-bridges, inverters, and optical isolators are displayed in Figure 13

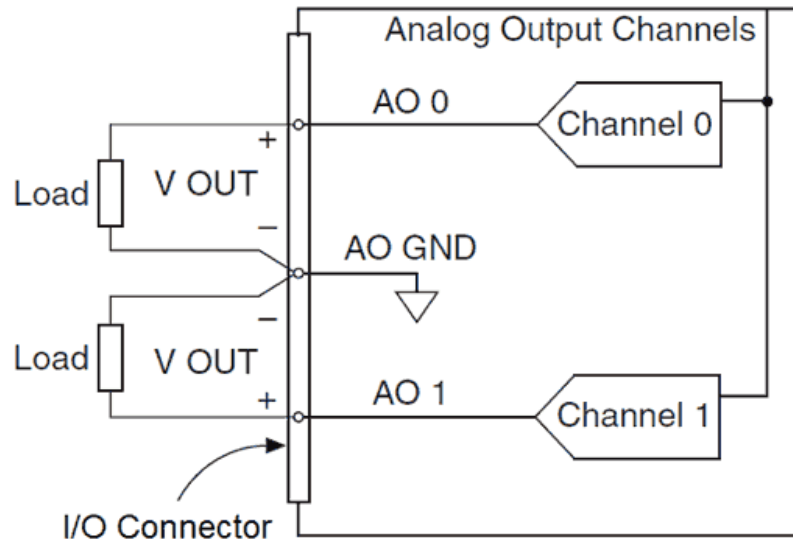
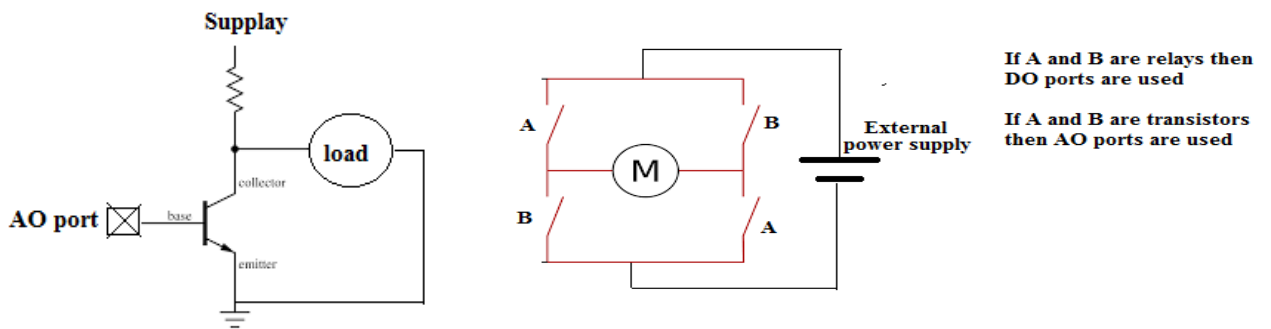
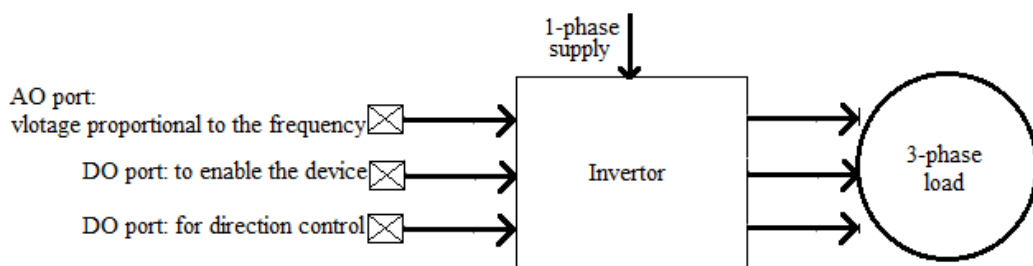


Figure 12: Connecting a Load to AO Measurement Channel



a. Connecting to Analog Outputs Using Transistors

b. Connecting to Dc Motor Using H-Bridge



a. Connecting to Induction Motor Using Inverter

Figure 13: Drive Circuit for AO Devices

The resolution of the output voltage of the DAQ is given by

$$AO \text{ Resolution} = \frac{V_o \text{ max}}{2^n - 1}$$

Where;

$V_{o\ max}$ is the maximum output voltage.

n: is the number of bits in the DAC.

5 USB 6008 DAQ Analog Ports Specifications

The National Instruments USB-6008 is a low-cost, multifunction data acquisition device (DAQ). It has 8 analog inputs, and 2 analog outputs. The pinout of the DAQ is presented by Figure 14.

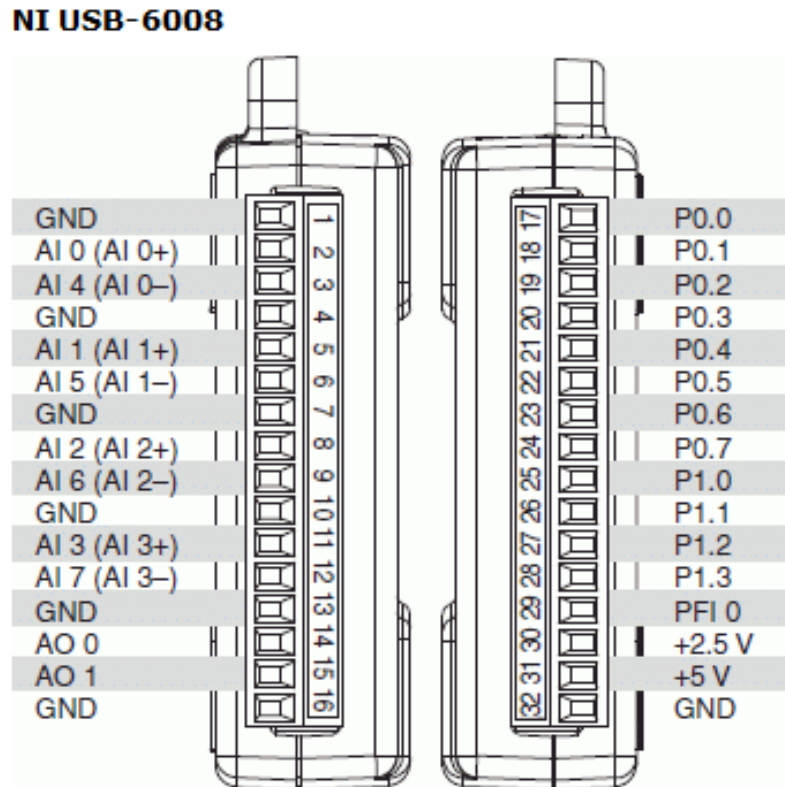


Figure 14: USB 6008 DAQ Pinout

As described by the data sheet, the DAQ supply ports, the digital I/O ports and the counter port have specifications described next. For more specifications refer to the complete data sheet available online.

5.1 Power Supply

The DAQ has two power sources according to the data sheet:

+5 V output (200 mA maximum) +5 V typical, +4.85 V minimum

+2.5 V output (1 mA maximum) +2.5 V typical

Very low current; the supply is used for self-test only

5.2 Analog Input

Feature	USB-6008
AI Resolution	12 bits differential,
Maximum AI Sample	10 kS/s

Converter type..... Successive approximation

**Analog inputs 8 single-ended/4 differential,
software selectable**

Input resolution

USB-6008 12 bits differential,

11 bits single-ended

ADC resolution

Max sampling rate

USB-6008 48 kS/s

S/s: Sample per second

Up to how many samples can the DAQ acquire during a second

Input range

Single-ended ±10 V

PGA Gain

**Differential..... ±20 V, ±10 V, ±5 V, ±4 V,
±2.5 V, ±2 V, ±1.25 V, ±1 V**

Peak to peak voltage

Working voltage..... ±10 V

Input impedance 144 kΩ

Overvoltage protection..... ±35

5.3 Analog Output

Converter typeSuccessive approximation

Analog outputs.....2

Output resolution12 bits

DAC Resolution=
 $V_{out(max)} / ((2^{12}) - 1)$

Maximum update rate150 Hz, software-timed

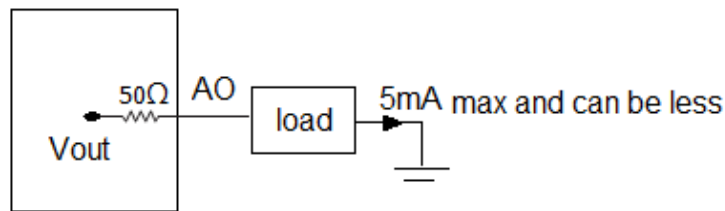
Power-on state0 V

Output range0 to +5 V

Output impedance.....50 Ω

Output current drive.....5 mA

Each analog output has an on-board 50Ω resistor and can drive loads with a maximum current of 5mA.



6 Procedure

1. Connect the digital inputs and outputs provided by your instructor properly.
2. Show the needed calculations.
3. Design a LabVIEW program to meet the requirements provided by your instructor.
4. Run and test your program and your hardware.
5. Save it with the proper name on the proper directory.

7 References:

- 1- <http://www.ni.com>
- 2- <https://www.contec.com/support/basic-knowledge/daq-control/analog-io>
- 3- <https://www.ni.com/en-lb/support/documentation/supplemental/06/field-wiring-and-noise-considerations-for-analog-signals.html#section--84297982>
- 4- Ott, Henry W., Noise Reduction Techniques in Electronic Systems. New York: John Wiley & Sons, 1976.
- 5- Barnes, John R., Electronic System Design: Interference and Noise Control Techniques, New Jersey: Prentice-Hall, Inc., 1987.

Experiment 3

Process Trainer System

1 Objectives

- Familiarize the student with the process trainer system (PT001).
- Learn how to model simple level and flow control systems.

2 Apparatus

Figure 1 presents the Process Trainer System used in this experiment.

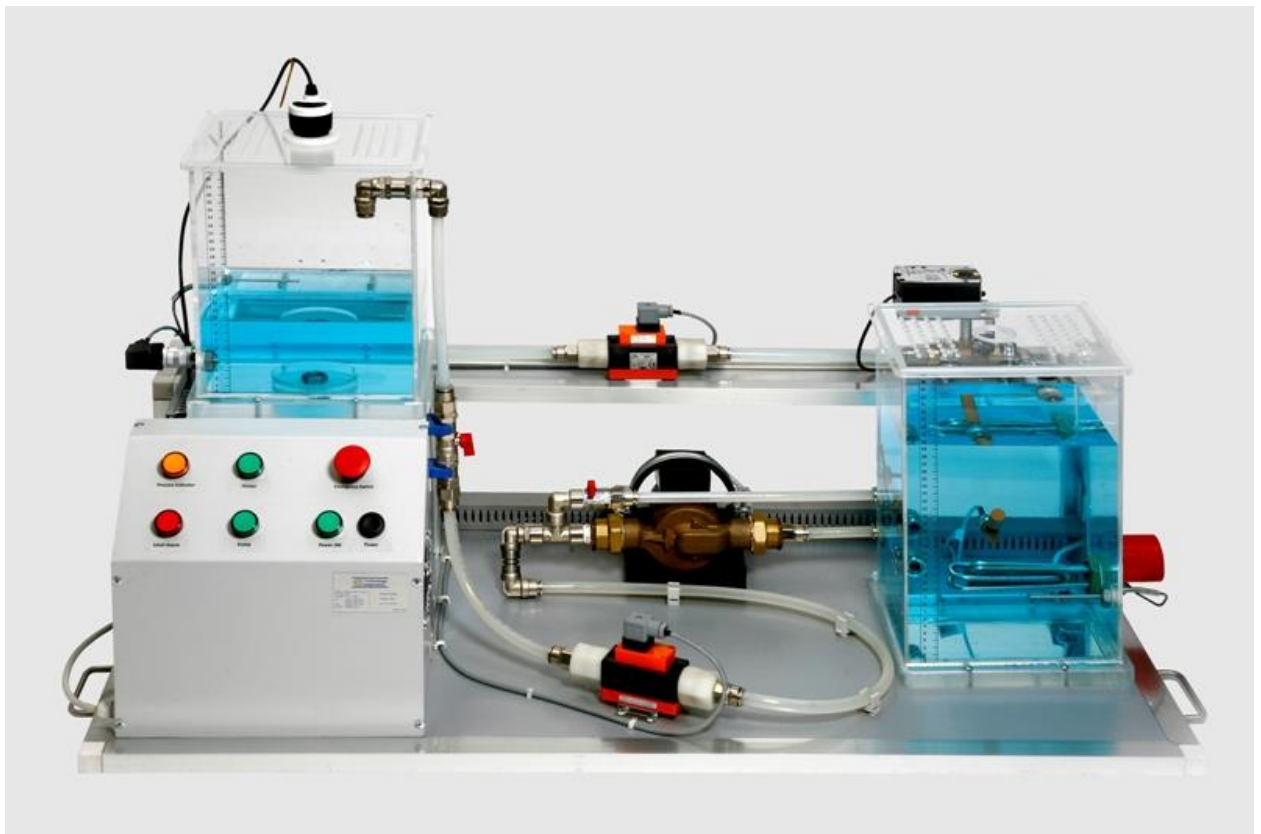


Figure 1: Process Trainer System PT001

3 Introduction

In Hydraulic systems there are three basic building blocks: Resistance, capacitance and inertance presented in Figure 2

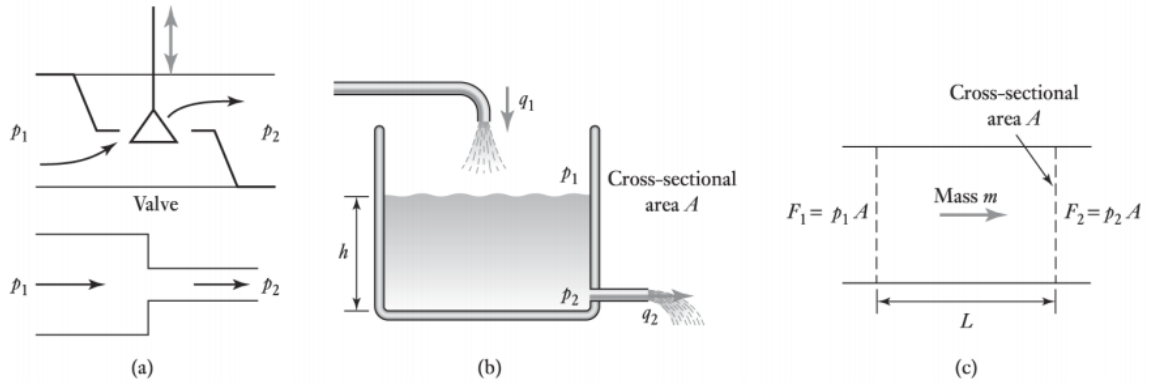


Figure 2: Hydraulic System Basic Blocks: (a) Resistance, (b) Capacitance, (c) Inertance

3.1 Fluid Resistance

Hydraulic resistance as presented in Figure 3 is the resistance to flow which occurs as a result of a liquid flowing through valves or changes in a pipe diameter. The relationship between the change in flow rate of liquid q through the resistance element and the resulting pressure difference ($p_1 - p_2$) is given by

$$\sqrt{h} = R * q \tag{4.1}$$

Where, $p = \rho gh$, and R is a constant called the hydraulic resistance. The higher the resistance, the higher the pressure difference for a given rate of flow. Equation 4.1 like that for the electrical resistance and Ohm's law, assumes a linear relationship. Such hydraulic linear resistances occur with orderly flow through capillary tubes and porous plugs but nonlinear resistances occur with flow through sharp-edge orifices or if the flow is turbulent.

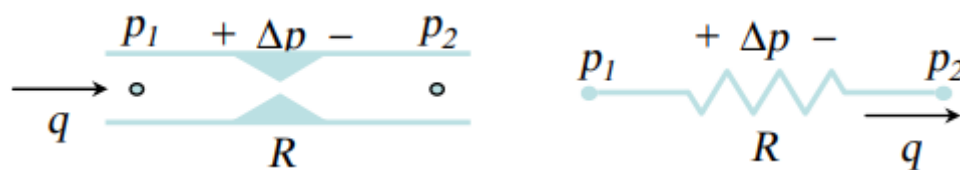


Figure 3: Hydraulic Resistance

3.2 Hydraulic Capacitance

Hydraulic capacitance is the term used to describe energy storage with a liquid where it is stored in the form of potential energy as shown in Figure 4. A height of liquid in the container shown in Figure 4 (called pressure head) is one form of such storage.

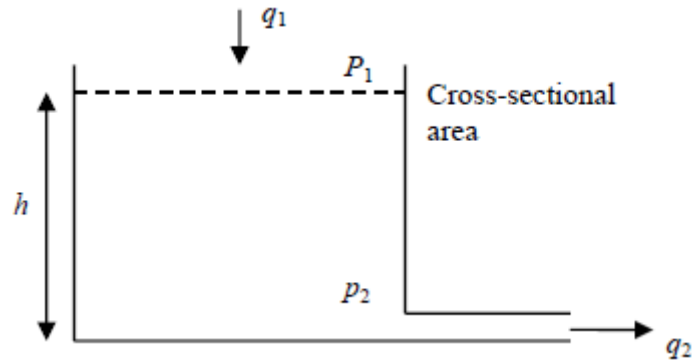


Figure 4: Hydraulic Capacitance

For such a capacitance, the rate of change of volume V in the tank (dV/dt) is equal to the difference between the volumetric rate at which liquid enters the container q_1 and the rate at which it leaves the container q_2

$$q_1 - q_2 = \frac{dV}{dt} \quad (4.2)$$

But $V = Ah$, where A is the cross-sectional area of the container and h is the height of liquid in it. Hence

$$q_1 - q_2 = \frac{d(Ah)}{dt} = A \frac{dh}{dt} \quad (4.3)$$

The pressure difference between the input and output is p , where $p = h\rho g$ with ρ being the liquid density and g the acceleration due to gravity. Therefore, if the liquid is assumed to be incompressible (density does not change with pressure), then Equation (4.3) can be written as

$$q_1 - q_2 = \frac{d(p / \rho g)}{dt} = \frac{A}{\rho g} \frac{dp}{dt} \quad (4.4)$$

The hydraulic capacitance C is defined as

$$C = \frac{A}{\rho g} \quad (4.5)$$

Therefore

$$q_1 - q_2 = C \frac{dp}{dt} \quad (4.6)$$

3.3 Hydraulic Inertance

Hydraulic inertance is the equivalent of inductance in electrical systems or a spring in mechanical systems. To accelerate a fluid and to increase its velocity a force is required. Consider a block of liquid of mass m as shown in Figure 5.

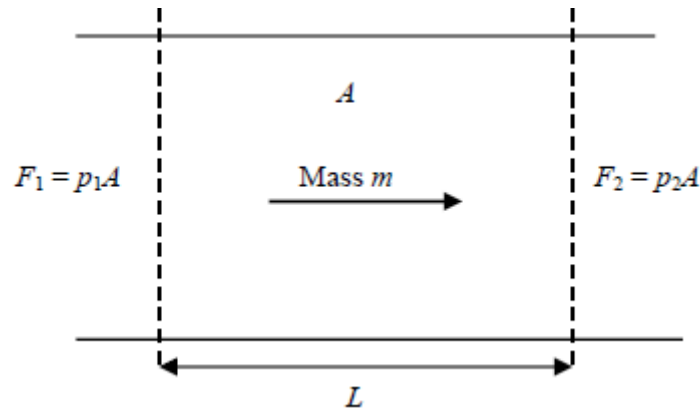


Figure 5: Hydraulic Inertance

The net force acting on the liquid is:

$$F_1 - F_2 = p_1A - p_2A = (p_1 - p_2)A \quad (4.7)$$

Where $(p_1 - p_2)$ is the pressure difference and A the cross-sectional area. This net force causes the mass to accelerate with an acceleration a , and so

$$(p_1 - p_2)A = ma \quad (4.8)$$

However, a is the rate of change of velocity dv/dt , therefore

$$(p_1 - p_2)A = m \frac{dv}{dt} \quad (4.9)$$

The mass of liquid concerned has a volume of AL , where L is the length of the block or the distance between the points in the liquid where the pressures p_1 and p_2 are measured. If the liquid has a density ρ then $m = AL\rho$, and

$$(p_1 - p_2)A = AL\rho \frac{dv}{dt} \quad (4.10)$$

But the volume rate of flow $q = Av$, hence

$$(p_1 - p_2)A = L\rho \frac{dq}{dt} \quad (4.11)$$

$$(p_1 - p_2) = I \frac{dq}{dt} \quad (4.12)$$

Where the hydraulic inertance I is defined as

$$I = \frac{L\rho}{A} \quad (4.13)$$

4 Process Trainer

The PT001 Process Trainer is an example of a typical hydraulic system. It is a bench-mounted, Process Trainer for Level, Flow, Temperature, and Pressure control of fluid system. It comprises all the required sensors and actuators to perform the many control operations.

The PT001 Process Trainer is also a typical Data Acquisition system. It includes sensor and actuators (physical plant), data acquisition devices (interface), signal conditioning circuits (impeded with the interface), and software (LabVIEW).

4.1 System Model

Figure 6 roughly represents the laboratory system where h_2 is the water height above the discharge point. If h_2 is not equal to zero, i.e. the water level is above the discharge point, then the two tanks are coupled and the system must be represented by two first order differential equations.

On the other hand, if h_2 is equal to zero, i.e. the water level is below the discharge point in the lower tank, then the tanks are decoupled. In this case the upper tank can be modeled by a simple first order system as shown in Figure 7, and the lower tank only represents a reservoir that doesn't affect the system model, and need not be modeled.

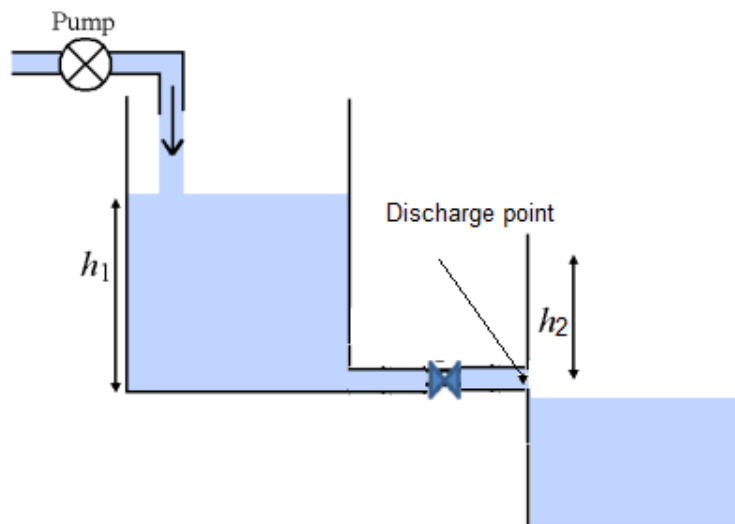


Figure 6: Process Trainer System

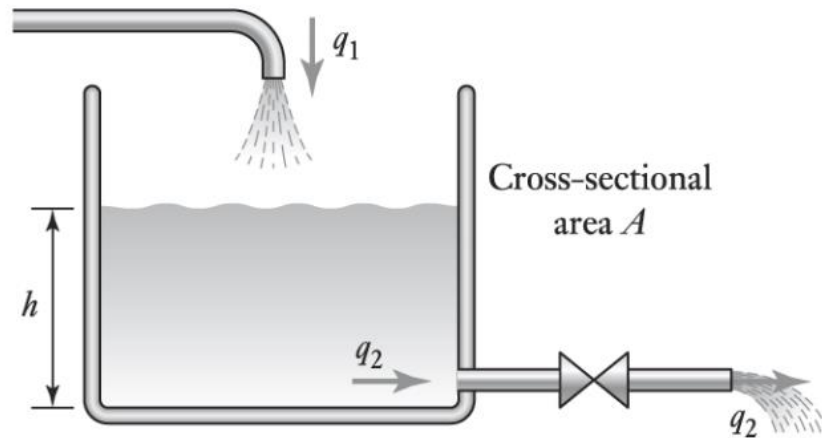


Figure 7: Process Trainer System with Decoupled Tanks

4.2 Hardware

The components of the system as presented in Figure 8 are:

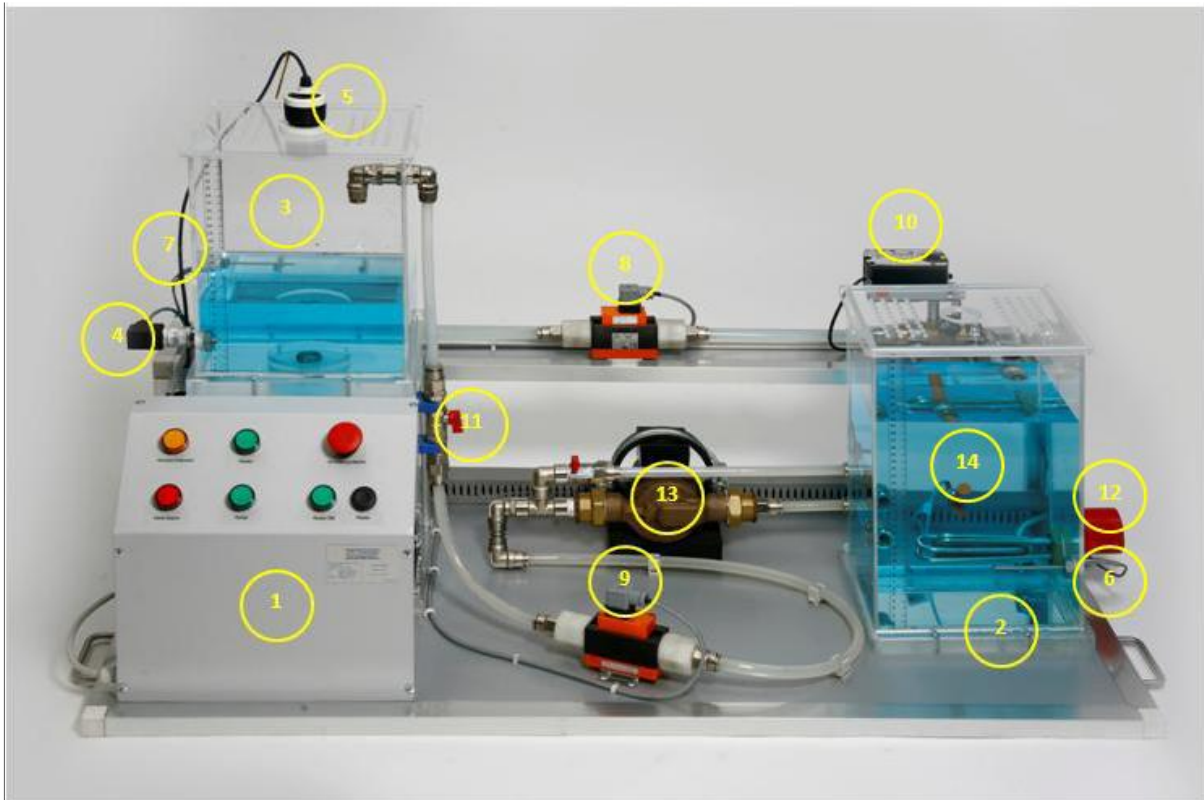


Figure 8: Components for the Process Trainer System

- | | | | |
|-------------------------|---------------------------|---------------------------------|-------------------------------|
| 1. Electric Control Box | 5. Ultrasonic Level Meter | 8. Flow Meter (Upper) | 11. Manual Flow Control Valve |
| 2. Tank 1 | 6. Thermocouple Tank 1 | 9. Flow Meter (Lower) | 12. Heater |
| 3. Tank 2 | 7. Thermocouple Tank2 | 10. Electric Flow Control Valve | 13. Pump |
| 4. Pressure Sensor | | | 14. Level Switch |

The main buttons of the Electric Control Box in Figure 9 are as follows:

- | | |
|---------------------|----------------------|
| 1. Emergency Switch | 5. Process Indicator |
| 2. Main Power | 6. Heater Indicator |
| 3. Pump Indicator | 4. Level Alarm |



Figure 9: Electric Control Box

4.3 Hardware Specifications

Table 1 summarizes the specification related to the sensors and actuators of the system. These specifications are important when you want to configure the DAQ assistant in the LabVIEW programs need to run the process trainer.

Table 1: Sensors and Actuators specifications

<p>1. Pressure Sensor</p> <ul style="list-style-type: none"> ✓ Measuring Range: 0 to 0.1 bar ✓ Supply Voltage: 24 VDC ✓ Output: 4 - 20mA ✓ Temperature Range: 0 - 80 °C ✓ Media: Water ✓ Housing: Stainless Steel ✓ Accuracy: $\pm 0.5\%$ of the full range 	<p>2. J-Type Thermocouple</p> <ul style="list-style-type: none"> ✓ Type: J-Type ✓ With Protective Tube ✓ Immersion Length: 10 cm
<p>3. Ultrasonic Level Meter</p> <ul style="list-style-type: none"> ✓ Measuring Range: 5 to 30 cm ✓ Supply Voltage: 24 VDC ✓ Output: 4 - 20mA ✓ Temperature Range: -40 - 71 °C 	<p>4. Flow Control Valve</p> <ul style="list-style-type: none"> ✓ Supply Voltage: 24 VAC ✓ Control Signal: 0 – 10 VDC ✓ Feedback Signal: 0 - 10 VDC ✓ Angle of Rotation: 0° - 95°

<ul style="list-style-type: none"> ✓ Accuracy: $\pm 0.15\%$ of span in air ✓ Dead Band: 5 cm 	<ul style="list-style-type: none"> ✓ Running Time: 108 sec
<p>5. Level Switch</p> <ul style="list-style-type: none"> ✓ Temperature: $-10^{\circ} - 50^{\circ}$ ✓ Pressure: 100 psig 	<p>6. Heater</p> <ul style="list-style-type: none"> ✓ Power: 1200 W ✓ Supply Voltage: 220 VAC, Single Phase
<p>7. Pump</p> <ul style="list-style-type: none"> ✓ Power: 0.17 HP ✓ Supply Voltage: 220 VAC, Single Phase ✓ Maximum Fluid Temperature: 100°C 	<p>8. Variable Speed Drive</p> <ul style="list-style-type: none"> ✓ Motor Output Rating: 0.5 HP ✓ Supply Voltage/ Phases: 220/ 1\emptyset ✓ Output Voltage/ Phases: 0-240/ 1\emptyset ✓ Supply Frequency: 48 to 62 Hz ✓ Max. Ambient Temperature 50°C ✓ Storage Temperature $-40 - 60^{\circ}\text{C}$
<p>9 . Inline Flow sensors</p> <ul style="list-style-type: none"> ✓ Measuring Range: 1.6-32 L/min ✓ Supply Voltage: 24 VDC ✓ Output: 4-20 mA ✓ Temperature Range: $0 - 60^{\circ}\text{C}$ 	<p>10. Plexiglas Tanks</p> <ul style="list-style-type: none"> ✓ Maximum Medium Temperature: 70°C ✓ Medium: Water

4.4 System Interface

The acquisition system is done with sensors and actuators are connected to a Compact DAQ. Compact DAQ USB chassis provide the plug-and-play simplicity of USB to sensor and electrical measurements on the benchtop, in the field, and on the production line. By combining more than 50 sensor-specific NI C Series I/O modules with patented NI Signal Streaming technology, the NI Compact DAQ platform delivers high-speed data and ease of use in a flexible, mixed-measurement system. The DAQ system in the lab is composed from a chassis and 4 modules as illustrated below.



1. National Instruments, NI CompactDAQ 4-Slot USB Chassis (NI cDAQ-9174)

- Choose from more than 50 hot-swappable I/O modules with integrated signal conditioning
- Four general-purpose 32-bit counter/timers built into chassis (access through digital module)
- Run up to seven hardware-timed analog I/O, digital I/O, or counter/timer operations simultaneously
- Stream continuous waveform measurements with patented NI Signal Streaming technology

2. National Instruments 16-Channel AI (8 Voltage and 8 Current) (NI 9207)

- 8 current inputs (± 21.5 mA) and 8 voltage (± 10 V)
- High-resolution mode with 50/60 Hz rejection
- 500 S/s sample rate (high-speed mode)
- Vsup pins for external power routing (2 A/30 V max)

3. National Instruments 4-Channel, 100 kS/s, 16-bit, ± 10 V, Analog Output Module (NI 9263)

- 4 simultaneously updated analog outputs, 100 kS/s
- 16-bit resolution
- Hot-swappable operation

4. National Instruments 4-Channel, 14 S/s, 24-Bit, ± 80 mV Thermocouple Input Module (NI 9211)

- 4 thermocouple or ± 80 mV analog inputs
- 24-bit resolution; 50/60 Hz noise rejection
- Hot-swappable operation
- NIST-traceable calibration

5. National Instruments 8-Channel 24 V Logic, 100 μ s, Sourcing Digital Output Module (NI 9472)

- 8-channel, 100 μ s digital output
- 6 to 30 V range, sourcing digital output
- D-Sub or screw-terminal connector options
- Hot-swappable operation
- Extreme industrial certifications/ratings

4.5 Hardware Connection:

Table 2 illustrates the connection of the sensors and actuators to their proper modules and channels.

Table 2: Connection Table

Slot No.	Module Name	Signal Name	Channel Number	Pin Number	Wire Number	Wire Color	
						Main	Strip
1	9211	Thermocouple Upper Tank	CH 0	0	...	Black	...
				1	...	White	...
		Thermocouple Lower Tank	CH 1	2	...	Black	...
				3	...	White	...
2	9207	Pressure Sensor	CH 8	11	3	Green	...
		Level Meter	CH 9	12	1	Brown	...
		Flow Meter Upper	CH 10	13	5	Gray	...
		Flow Meter Lower	CH 11	14	4	Yellow	...
		Common	...	28	2	White	...
		Flow Control Valve Feed Back	CH 0	1	1	Brown	
20	2			White			
3	9263	Variable Frequency Drive + Variable Frequency Drive -	CH 0	0	1	Brown	...
				1	4	Yellow	...
		Flow Control Valve Open + Flow Control Valve Open -	CH 1	2	2	Green	...
				3	3	White	...
4	9472	Pump Enable	CH 0	0	4	Yellow	...
		Heater Enable	CH 1	1	3	Green	...
		Process Indicator Enable	CH 2	2	5	Gray	...
		24 VDC	...	8	1	Brown	...
		Common	...	9	2	White	...
5	24VDC	Supply Module 9207	...	V _{sup}	35	Brown	...
			...	COM	29	White	...

4.6 Software

The system is driven by means of LabVIEW programs designed from the manufacturer or else designed by the students to meet certain requirements.



Figure 4.5: Manufacturer Designed Software

5 Procedure

1. Inspect Tables 1 and 2.
2. Design a LabVIEW program to meet the task provided by your lab supervisor. You may need the following formulas:
 - For the ultrasonic level meter : $H= 1543.75 *I - 6.175$
 - For any analog signal, select "one sample on demand".
 - The flow control valve at 0V is fully opened.
3. Fill the following Table to help you configure your DAQ assistant.

Table 3: Hardware Configuration

	Input/output	Digital/Analog	Signal type (voltage/current)	Range of Signal	Module name	Channel number
Actuator						
Sensor1						
Actuator						
Sensor2						

6 Discussion and Analysis

1. How many sensors are there in the system you designed? List them?
2. How many actuators are there in the system you designed? List them?
3. Draw a flow chart of the program you designed.
4. How many DAQ assistant you used in your program? What are their types?
5. How many samples per second do the analog modules support?

7 References:

Mechatronics: electronic control systems in mechanical and electrical engineering, 4th edition,
W.Bolton

Experiment 5

PID Controller: Concept and Tuning

1 Objectives

- Familiarize the student with the concept of PID controller.
- Study the effect of each action of the PID controller.
- Learn how tune the PID controller gains.

2 Apparatus

Figure 1 presents the Process Trainer System used in this experiment.



Figure 1: Process Trainer System PT001

3 Introduction

Proportional-Integral-Derivative (PID) control (Figure 2) is the most common control algorithm used in the industry and has been universally accepted in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating

conditions and partly to their functional simplicity, which allows engineers to operate them easily. As the name suggests, PID algorithm consists of three basic coefficients; proportional, integral and derivative, which are varied to get optimal response.

$$PID = K_p * e(t) + K_i * \int e(t)dt + K_d * \frac{de(t)}{dt}$$

or (5.1)

$$PID = K_p * \left[e(t) + \frac{1}{T_i} * \int e(t)dt + T_d * \frac{de(t)}{dt} \right]$$

Where, T_i is the integral time and is equal to K_p/K_i , and T_d is the derivative time and is equal to K_d/K_p .



Figure 2: PID Controller Algorithm

3.1 Proportional Response

The proportional component depends only on the difference between the set point and the process variable. This difference is referred to as the Error term. The proportional gain (K_c) determines the ratio of output response to the error signal as given by equation 5.2. For instance, if the error term has a magnitude of 10, a proportional gain of 5 would produce a proportional response of 50. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If K_c is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control.

$$P_{out} = K_p * e(t) \tag{5.2}$$

3.2 Integral Response

The integral component summates the error term over time this integral action is presented by equation 5.3. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero. Steady-State error is the final difference between the process variable and setpoint. A phenomenon called integral windup results when integral action saturates a controller without the controller driving the error signal toward zero.

$$I_{out} = K_i * \int e(t)dt \tag{5.3}$$

3.3 Derivative Response

The derivative component causes the output to decrease if the process variable is increasing rapidly. The derivative response as presented by equation 5.3 is proportional to the rate of change of the process variable. Increasing the derivative time (Td) parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time (Td), because the Derivative Response is highly sensitive to noise in the process variable signal. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable.

$$D_{out} = T_d * \frac{de(t)}{dt} \tag{5.4}$$

Table 1 shows the effect of each controller gain on the system performance. Notice that these effects are subjected to changes when the controller includes more than one control action.

Table 5.1: PID Controller's Elements Action

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

4 PID Tuning Using Ziegler and Nichols

Ziegler and Nichols described simple mathematical procedures, the first and second methods respectively, for tuning PID controllers. These procedures are now accepted as standard in control systems practice. Both techniques make a priori assumptions on the system model, but do not require that these models be specifically known. Ziegler-Nichols formula for specifying the controllers are based on plant step responses.

4.1 The First Method

The first method is applied to plants with step responses of the form displayed in Figure 3 (response exhibit s-shaped curve for unit step input). This type of response is typical of a plant made up of a series of first order systems.

The response is characterized by two parameters, L the delay time and T the time constant. These are found by drawing a tangent to the step response at its point of inflection and noting its intersections with the time axis and the steady state value. The plant model is therefore

$$G(s) = \frac{ke^{-sL}}{Ts + 1} \quad (5.4)$$

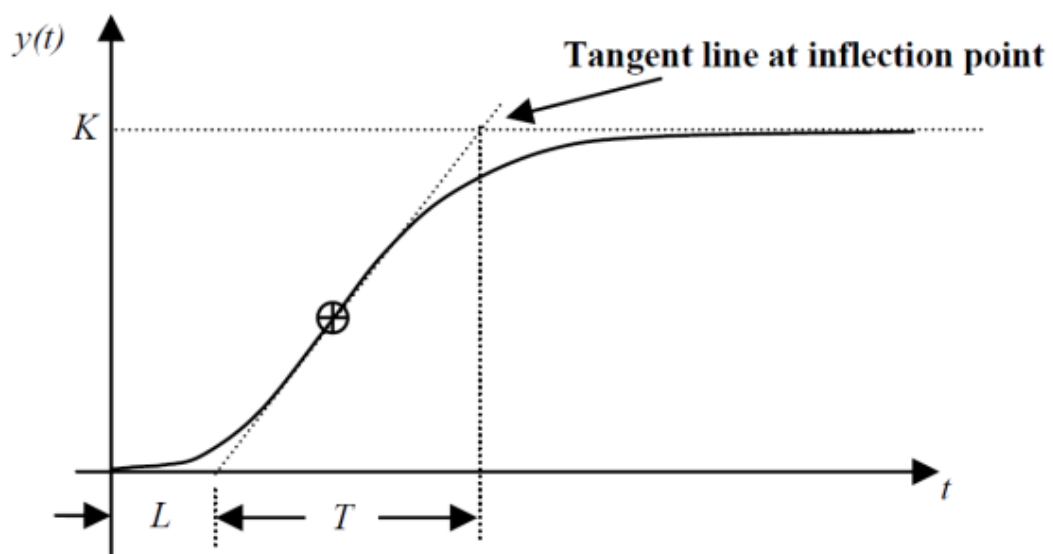


Figure 3: S-Shape Response

Typical first order system doesn't exhibit an s-shape response (change in slope) but it is also characterized by L and T as shown in Figure 4.

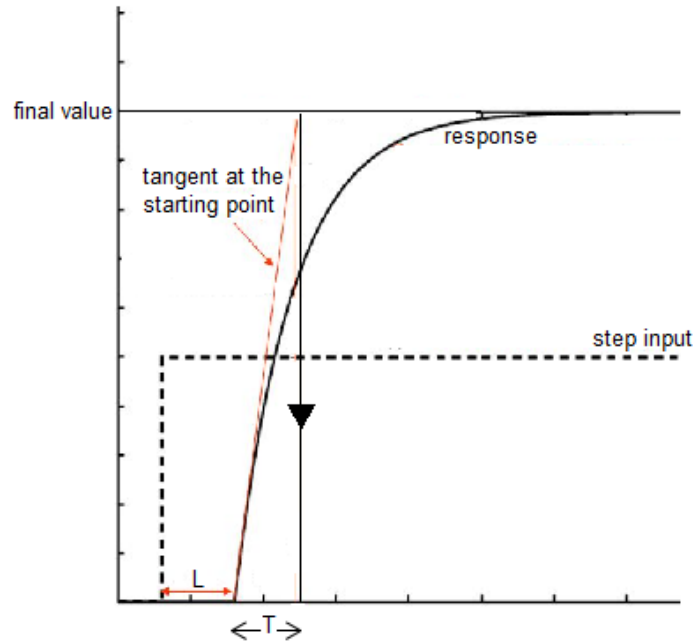


Figure 4: First Order System Response

Procedure of applying the first method

To design a PID controller using this method, apply the following steps. Note that the tuning is done while system is in **open loop** configuration as in Figure 5, and to get the step response in Figure 3 or Figure 4 , the system must be stable in the open loop configuration.



Figure 5: First Method Block Diagram

1. Apply a step input to the plant.
2. Record the s-shape curve you will obtain and find L and T parameters.
3. Select the values of the controller gains based on Table 2.
Notice that $K_i=1/T$ and $K_d=1/T_d$. The control parameters in Table 2 are derived by Ziegler and Nichols.
4. If the system is a first order system, a complete PID controller is considered unnecessary. a first order system sustains no oscillations; thus no need for the derivative action and only PI controller is enough.

Table 2: PID Controller Tuning Using Ziegler-Nichols First Method

Type of Controller	K_p	T_i	T_d
P	T/L	∞	0
PI	$0.9 T/L$	$L / 0.3$	0
PID	$1.2 T/L$	$2L$	$0.5L$

4.2 The Second Method

The second method targets plants that can be rendered unstable under proportional control. The technique is designed to result in a closed loop system with 25% overshoot. This is rarely achieved as Ziegler and Nichols determined the adjustments based on a specific plant model.

Procedure of applying the second method

To design a PID controller using this method, apply the following steps. Note that the tuning is done while system is in **closed loop** configuration, with only proportional controller, as demonstrated by Figure 6.

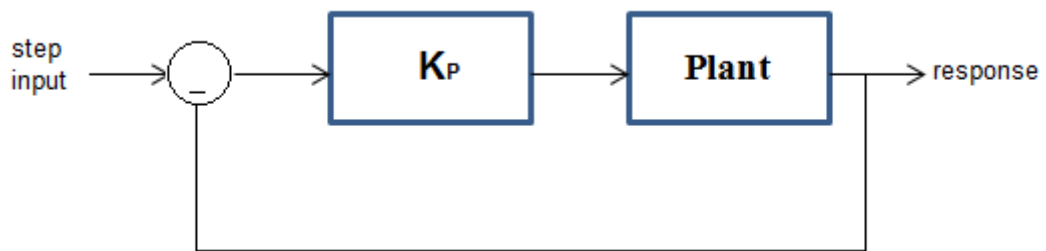


Figure 6: Second Method Block Diagram

1. Reduce the integrator and derivative gains to 0.
2. Increase K_p from 0 to some critical value $K_p=K_{cr}$ at which sustained oscillations occur, as demonstrated by Figure 7.
3. If it does not occur then another method has to be applied.
4. Notice from Figure 7 the value of K_{cr} , and the corresponding period of sustained oscillation P_{cr} .
5. The controller gains are now specified according to Table 3.

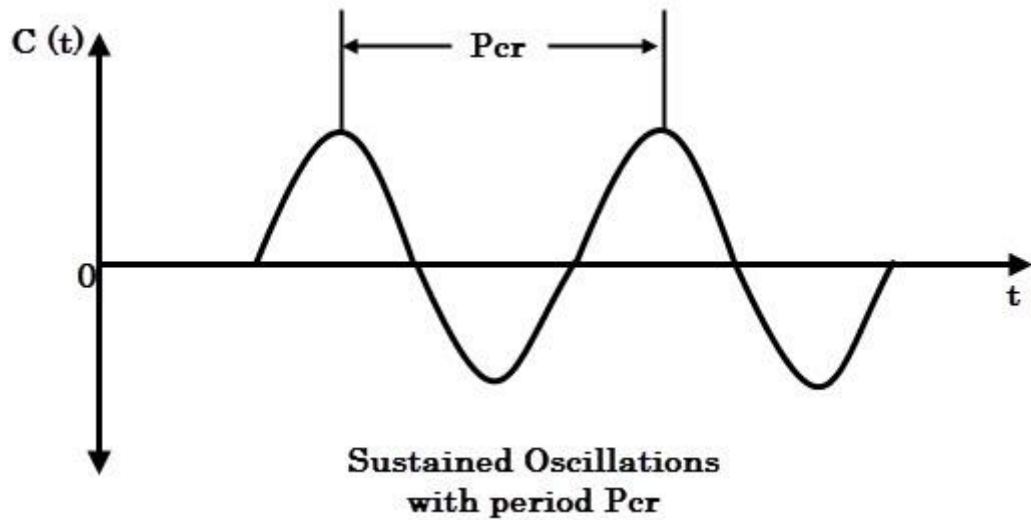


Figure 7: Second Method Response

Table 3: PID Controller Tuning Using Ziegler-Nichols Second Method

Type of Controller	K_p	T_i	T_d
P	$0.5 K_{cr}$	∞	0
PI	$0.45 K_{cr}$	$1/1.2 P_{cr}$	0
PID	$0.6 K_{cr}$	$0.5 P_{cr}$	$0.125 P_{cr}$

5 Process Trainer System

Figure 8 shows the Process Trainer System in the laboratory, both level and pressure control can be done with this trainer.

For the PID Feedback Level Experiment, the PID level controller takes the feedback signal from the level meter and compares it with the desired value of level (Setpoint) and accordingly gives a control signal to the pump to deliver the required water flow & maintain the level at the desired value.

Notice that the system is unstable in the open loop configuration, thus the first method of tuning is not applicable.

The same principle is applied to the pressure and flow controllers, except that the feedback signal is taken from the pressure sensor and the flow meter respectively. This type of controllers rejects the disturbances after they affect the PV (Water Level)

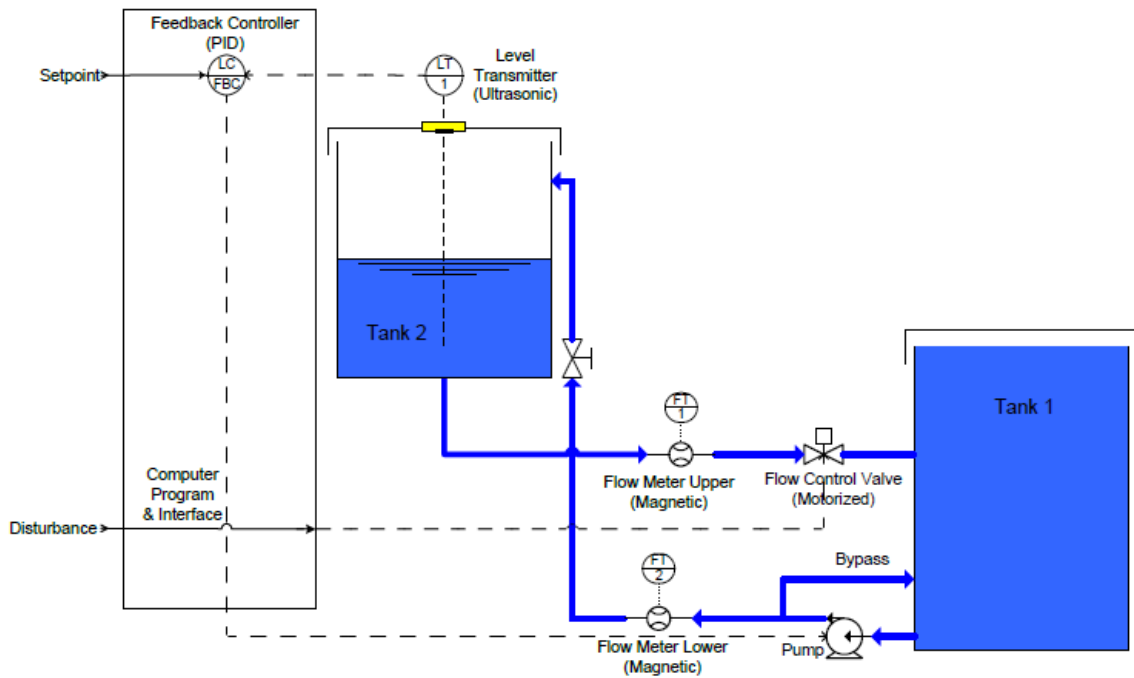


Figure 8: PID Level Control Implementation in PT001

6 Procedure

1. Click on Experiment 3: “PID Feedback Control”. Another menu will appear; choose Experiment 3.1: “PID Feedback Level Control”.
2. Drag the Setpoint slider to 30% of its value. Press the [Start Process] button to start the control process. Observe the behavior of the (PV) and the controller’s output on the chart.
3. After the controller and (PV) settle down and stabilize, apply a disturbance to the system by changing the Flow Control Valve’s opening. Observe the disturbance rejection and the reaction of the controller.
4. Set the Flow Control Valve to “Fully Open”, and change the setpoint to 60%. When the level settles at 60% of its full scale, observe the behavior of the pump.
5. Change the setpoint to other values between (20% & 75%). Observe the controller’s behavior at these new setpoint values.
6. Stop the process and press the [Quit] button to go back to the experiments menu.



Figure 9: Manufacturer Designed Software

7 Discussion and Analysis

1. For the system you operated, check the PID controller's parameters. What type of a controller is it?
2. Sketch the response you obtained during step 2 of the experiment.
3. What is the effect of applying the disturbance?
4. When you changed the opening to 60% in step 4, does the pump operate at a higher or lower rate in comparison to its operation at the 30% setpoint? Why?

8 References:

1. <http://www.ni.com/white-paper/3782/en/>
2. The Design of PID Controllers using Ziegler Nichols Tuning, Brian R Copeland , 2008.