

RUN-TIME DATA ACQUISITION FROM A PACKED BED TOWER

PROF. D.P. RAO (PRINCIPAL INVESTIGATOR), DIVYA KISHORE S. RATHORE, Y. SAWJANYA, PREETI SOOD

Acknowledgements – MR. P.S. CHAUHAN, MR. RAJIV YADAV, MR. R.K. VISHWAKARMA



INTRODUCTION

In the past few years, *virtual instrumentation* and *automation* have undisputedly emerged as two extremely significant fields of engineering and technology. The ever-increasing industrialization that we see today thrives solely on efficiency and productivity. Computer based systems have greater flexibility than traditional ones because *we* define the function of the software and hardware. The data collected by these systems can be efficiently distributed across a plant floor or around the world.

This experiment involves study of a digital interface for a packed bed tower so as to enable a user to monitor the tower using a computer.

Objective:

The objective of this experiment is to understand the basics of a run-time data acquisition system and, furthermore, study the loading and flooding characteristics and pressure drop across a packed-bed tower by monitoring the tower through computerized data acquisition in real time.

Equipment List:

- Computer running Windows, Macintosh, Linux, Sun, or HP-UX (visit www.ni.com/labview/lv_sysreq.htm for requirements specific to your system).
- Packed-bed tower assembly with ½” Raschig rings as the packing material.
- 6024E board- the data acquisition (DAQ) card, from National Instruments Corporation, installed on the computer.
- SC-2070 Termination Breadboard from National Instruments Corporation, to supply analog signals to the DAQ card.
- LabVIEW 6i, application software for virtual instrumentation, installed on the computer.
- NI-DAQ driver software
- Two differential pressure transducers (PX170-28DV from Omega Engineering, Inc.).
- An Orifice meter (It is a substitute for a gas mass flow meter)
- ‘K’ type thermocouple and TX91 Series Two-Wire Temperature Transmitter from Omega Engineering, Inc.

- Level switch (LV-40 from Omega Engineering, Inc.).
- Solid-state relay (SSR240DC25 from Omega Engineering, Inc.)
- Solenoid valve (SV-335-SS-316SS from Omega Engineering, Inc.)

Description of Equipments:

Note: Most of the manuals that are referred to, ship with National Instruments and OMEGA hardware and software. If you can't find your hardcopy of the manuals, you can get them online at <http://www.ni.com/manuals> and www.omega.com.

[A] 6024-E Data Acquisition Card [National Instruments]

The 6024-E series DAQ board is a high performance multifunction analog, digital, and timing I/O board for PCI bus computers. Supported functions include analog input, analog output, digital I/O, and timing I/O.

The 6024-E board has a *resolution* of 12 bits. Resolution is the smallest change that can be distinguished by a measurement instrument. It is sometimes referred to as "sensitivity" or "precision" and is limited by the number of bits that the instrument uses to quantize the signal it is measuring. The greater the resolution of an instrument, the smaller the changes in the input signal that can be measured (or resolved). 6024-E board, as it has a 12-bit resolution, divides the range into 2^{12} , or 4096, steps. In this case a 0-10 V range will be resolved to 0.25 mV, and a 0-100 mV range will be resolved to 0.0025 mV.

The board features 16 channels of analog input, 2 channels of analog output, a 68-pin connector and eight lines of digital I/O.

NOTE: For any further details about the board including installation/configuration, hardware details, signal connections or calibration you may refer to the '6024-E Series Board User Manual' available with your Laboratory In charge.

[B] SC-2070 Termination Breadboard [National Instruments]

The SC-2070 is a general-purpose breadboard with signal-labeled screw terminals and temperature sensor, breadboard, and analog signal conditioning areas. It simplifies the connection of analog and digital signals to the DAQ board in laboratory, test, and production environments.

SC-2070 board has an onboard temperature sensor that can be optionally jumpered to differential channel 0 for thermocouple cold-junction compensation. With open component positions in the input paths, you can insert resistors and capacitors for conditioning the 16 single-ended or 8 differential analog input signals.

NOTE: For any further details about the breadboard including installation/configuration, hardware details or signal connections you may refer to the 'SC-2070 Termination Breadboard User Manual' available with your Laboratory In charge.

[C] LabVIEW 6i™ [National Instruments]

Introduction

LabVIEW (**L**aboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) is a program development environment, much like modern C/C++ development environments. However, LabVIEW is different from those applications in one important respect. Other programming systems use *text-based* languages to create lines of code, while LabVIEW uses a *graphical* programming language, *G*, to create programs in block diagram form. LabVIEW, like C/C++, is a general-purpose programming system with extensive libraries

of functions for any programming task. LabVIEW includes libraries for data acquisition, data analysis, data presentation, and data storage. LabVIEW also includes conventional program development tools, so we can set breakpoints, animate the execution to see how data passes through the program, and single-step through the program to make debugging and program development easier.

How Does LabVIEW Work?

LabVIEW is a general-purpose programming system, but it also includes libraries of functions and development tools designed specifically for data acquisition and instrument control. LabVIEW programs are called *virtual instruments (VIs)* because their appearance and operation can imitate actual instruments. However, VIs are similar to the functions of conventional language programs.

A VI consists of an interactive user interface, a dataflow diagram that serves as the source code, and icon connections that allow the VI to be called from higher level VIs. More specifically, VIs are structured as follows:

- The interactive user interface of a VI is called the *front panel*, because it simulates the panel of a physical instrument. The front panel can contain knobs, push buttons, graphs, and other controls and indicators. We enter data using a mouse and keyboard, and then view the results on the computer screen.
- The VI receives instructions from a *block diagram*, which we construct in G. The block diagram is a pictorial solution to a programming problem. The block diagram is also the source code for the VI.
- VIs are hierarchical and modular. We can use them as top-level programs, or as subprograms within other programs. A VI within another VI is called a *subVI*. The *icon and connector* of a VI work like a graphical parameter list so that other VIs can pass data to a subVI.

With these features, LabVIEW promotes and adheres to the concept of *modular programming*. We divide an application into a series of tasks, which we can divide again until a complicated application becomes a series of simple subtasks. We build a VI to accomplish each subtask and then combine those VIs on another block diagram to accomplish the larger task. Finally, our top-level VI contains a collection of subVIs that represent application functions.

Because we can execute each subVI by itself, apart from the rest of the application, debugging is much easier. Furthermore, many low-level subVIs often perform tasks common to several applications, so that we can develop a specialized set of subVIs well suited to applications we are likely to construct.

G Programming

G is the graphical data flow programming language on which LabVIEW is based. G simplifies scientific computation, process monitoring and control, and test and measurement applications and we also can use it for a wide variety of other applications. The basic concepts of G are described in the following list.

- VIs—Virtual instruments (VIs) have three main parts: the front panel, the block diagram and the icon/connector. The front panel specifies the user interface of the VI. The block diagram consists of the executable code that we create using nodes, terminals and wires. With the icon/connector, we can use a VI as a subVI in the block diagram of another VI.
- Loops and Charts—G has two structures to repeat execution of a sub-diagram—

the *While Loop* and the *For Loop*. Both structures are resizable boxes. We place the sub diagram to be repeated inside the border of the loop structure. The While Loop executes as long as the value at the conditional terminal is TRUE. The 'For' Loop executes a set number of times. Charts are used to display real-time trend information to the operator.

- Case and Sequence Structures—The *Case structure* is a conditional branching control structure, which executes a sub diagram based on certain input. A *Sequence structure* is a program control structure that executes its sub diagrams in numeric order.
- Property Nodes—*Property nodes* are special block diagram nodes that we can use to control the appearance and functional characteristics of controls and indicators.
- Arrays, Clusters and Graphs—An *array* is a resizable collection of data elements of the same type. A *cluster* is a statically sized collection of data elements of the same or different types. Graphs commonly are used to display data.

NOTE: For any further details about LabVIEW, you may refer to 'LabVIEW Help' or 'LabVIEW Manual'.

[D] **NI-DAQ Driver Software** [*National Instruments*]

A driver is a program that interacts with a particular device or special (frequently optional) kind of software. The driver contains the special knowledge of the device or special software interface that programs using the driver do not.

This software includes program called Measurement and Automation Explorer (MAX). MAX provides access to all your National Instruments DAQ, GPIB, IMAQ, IVI, Motion, VISA, and VXI devices. With Measurement & Automation Explorer, you can:

- Configure your National Instruments hardware and software.
- Add new channels, interfaces, and virtual instruments.
- Execute system diagnostics
- View devices and instruments connected to your system

[E] **Pressure Transducers** [*OMEGA*]

The OMEGA PX170 Series Pressure Transducers are four-active-element piezoresistive bridges. When pressure is applied, a differential output voltage, proportional to that pressure, is produced. The transducers have laser trimmed resistors and a thermistor to provide low sensitivity shift with temperature. Differential pressure sensors apply P1 and measurand to the active (connection) side of the chip, and P2 to the passive side.

The circled numbers in **Figure 1** refer to sensor terminals (interface). V_o increases with pressure difference. We have, $V_o = V_2 - V_4$

Remember that a millivolt signal is of a very low level and is limited to short distances (up to 200 feet is usually considered the limit) and is very prone to stray electrical interference from other nearby electrical signals (other instrumentation, high ac voltage lines, etc.).

Specifications:

Excitation- 10VDC typical; 12VDC max.
Output- 0-28" H₂O, 42 mV ± 2 mV
Operating Temperature- -40°C to +85°C.
Gage Type- Solid State Piezoresistive
Diaphragm Material- 0.175-inch square silicon sensor chip

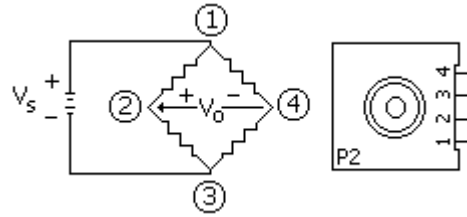


Fig.1: The pressure Transducer

CAUTION – Never cross the maximum output range of 28" (70 cm, approx.) of H₂O. This can cause permanent physical damage to the Transducer.

NOTE: For any further details about the Transducers including mounting, soldering/cleaning, hardware details or signal connections you may refer to the 'OMEGA PX170 Pressure Transducer Operator's Manual' available with your Laboratory In charge.

[F] Thermocouple and Temperature Transmitter [Omega]

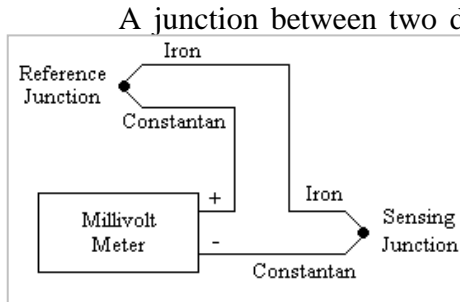


Fig. 2: A typical thermocouple circuit

A junction between two dissimilar metals generates a small voltage, typically in the range of millivolts. This junction is called a 'thermocouple', and it is useful for measuring temperatures over a broad range. By using various pairs of alloys it is possible to span temperatures from -270°C. to +2500°C with reasonable accuracy (0.5-2°C). The thermoelectric properties of various metals are well known, so thermocouple probes in different formats (rods, washers, armored probes, etc.) made from the same alloys can be interchanged without affecting calibration.

A typical thermocouple circuit is shown in **Figure 2**. Each couple is made by welding the two dissimilar metals together to make a small junction. The thermocouple circuit gives a voltage that depends on the temperatures of both junctions. Roughly speaking, it is proportional to the difference between the two junction temperatures. A thermocouple operates on the principle that the junction of two dissimilar metals generates a voltage that varies with temperature. However, measuring this voltage is difficult because connecting the thermocouple to DAQ board measurement wires creates what is called the reference junction or cold junction. These additional junctions act as thermocouples themselves and produce their own voltages. Thus, the final measured voltage, VMEAS, includes both the thermocouple and reference-junction voltages. The method of compensating for these unwanted reference-junction voltages is called cold-junction compensation.

The thermocouple being used in this experiment is a K type thermocouple (wires are of Alumel and Chromel) with temperature range of 0-750 K. The Transmitter attached to this is an OMEGA TX91 Thermocouple Transmitter. The transmitter accepts thermocouple sensor types J, K, T or E and produces a standard 4-20 mA output signal proportional to that produced by its attached input temperature sensor and is to be supplied a power of 11-44 VDC (we will be using a 12 VDC supply). The signal connections are shown in **Figure 3**. Transmission of the proportional current output may be

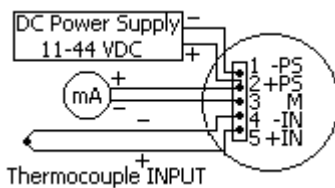


Fig. 3: Transmitter Circuit

accomplished by using inexpensive copper wire. *Calibration of the thermocouple-transmitter assembly may be done as given in transmitter manual.*

NOTE: For any further details about the Thermocouple and Transmitter assembly including mounting, hardware details, signal connections or **calibration** you may refer to the 'TX91 Series Thermocouple Two-Wire Temperature Transmitter Operator's Manual' available with your Laboratory In charge.

[G] Level Switch [OMEGA]

Level measurement is an integral part of process control. Level measurement may be divided into two categories, point level measurement and continuous level measurement. Point level sensors are used to mark a single discrete liquid height, a present level condition. Generally, this type of sensor is used as a high alarm, to signal the existence of an overflow condition, or as a marker for a low alarm condition.

The basic float switch is a simple point level sensor. A magnet-equipped float, which moves directly with the liquid surface, actuates a hermetically sealed switch within the stem. The float switch is designed to provide high repeatability, minimizing the effects of shock, vibration and pressure (connections in **Figure 4**).

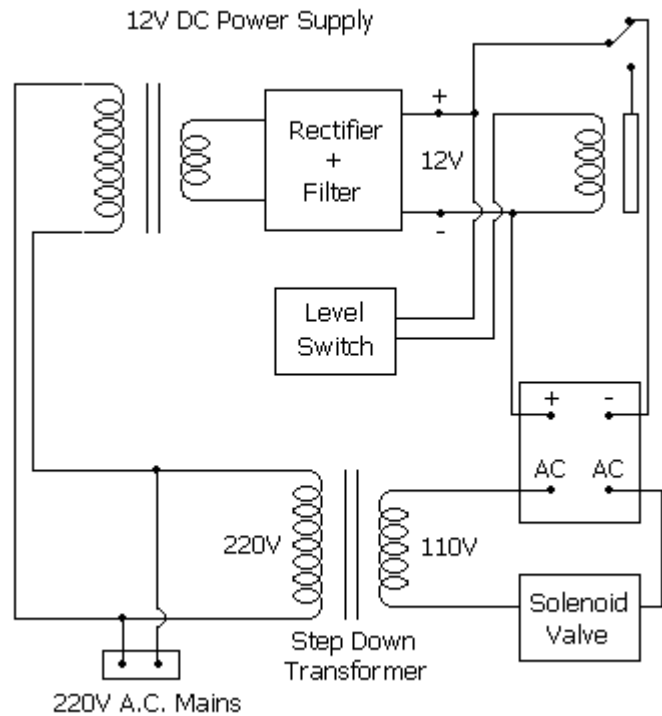


Fig. 4: Level Switch-Solenoid valve connections

[H] Solid State Relay [OMEGA]

A solid-state relay is an ON-OFF control device in which one or more semiconductors conduct the load current – e.g., a power transistor, an SCR, or a TRIAC. (The SCR and TRIAC are often called “thyristors”, a term derived by combining *thyatron* and *transistor*, since thyristors are *triggered semiconductor switches*).

Like all relays, the SSR requires relatively low control-circuit energy to switch the output state from OFF to ON, or vice versa. Since this control energy is very much lower than the output power controllable by the relay at full load, “power gain” in an SSR is substantial.

[I] Solenoid Valve [OMEGA]

Solenoid valves are control units which, when electrically energized or de-energized, either shut off or allow fluid flow. The actuator takes the form of an electromagnet. When energized, a magnetic field builds up which pulls a plunger or pivoted armature against the action of a spring. When de-energized, the plunger or pivoted armature is returned to its original position by the spring action.

SV-336 is a *direct 2-way acting* valve that is a shut-off valve having one inlet port and one outlet port. In the de-energized condition, the core spring assisted by the pressure of the fluid holds the valve seal on the valve seat, shutting off the flow. When energized, the core and the seal are pulled into the solenoid coil and the valve opens.

[I] Orificemeter

Flow rate as a function of pressure difference for the orifice meter (see **Figure 5**) can be calculated using the equation:

$$Q_a = \frac{C_D A_2}{\sqrt{1 - \left(\frac{A_2}{A_1}\right)^2}} \sqrt{\frac{2(P_1 - P_2)}{\rho}}$$

Where,

$Q_a \equiv$ Volumetric Flow Rate (m^3/sec);

$C_D \equiv$ Discharge Coefficient = 0.6;

$D_1 \equiv$ Tube cross sectional diameter = 11.5 mm;

$D_2 \equiv$ Orifice cross sectional diameter = 6 mm;

$A_1 \equiv$ Tube cross sectional area (m^2);

$A_2 \equiv$ Orifice cross sectional area (m^2);

$P_1 - P_2 \equiv$ Pressure drop across orifice (Pa);

$\rho \equiv$ Density of fluid (kg/m^3);

$s_d \equiv$ Downstream tap distance = 6 mm;

$s_u \equiv$ Upstream tap distance = 2.5 cm;

$s_w \equiv$ Orifice width = 5 mm;

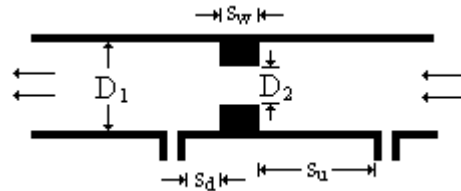


Fig. 5: Orifice meter

Theory:

The operations which include humidification and dehumidification, gas absorption and desorption, and the distillation in its various forms all have in common the requirement that a gas and a liquid phase be brought into contact for the purpose of a diffusional interchange between them. The pressure drop in gas phase when both phases are simultaneously flowing is significantly different from that when liquid is absent. On a log-log plot, dry packings show a linear relation with a slope between 1.8 and 2.0 indicating turbulent flow for most practical gas velocities. The pressure drop in packed columns because of flow is due to both skin and form friction. For a single-phase flow, pressure drop can be calculated by Ergun equation,

$$\left(\frac{\Delta P}{Z}\right) * \frac{\varepsilon^3 d_p \rho_g}{(1 - \varepsilon) G'^2} = \frac{150(1 - \varepsilon)}{\text{Re}} + 1.75 \quad \dots(\text{Eqn. 1})$$

Where, $\text{Re} = d_p G' / \mu =$ Reynolds No.

$$d_p = \frac{6(1 - \varepsilon)}{a_p} = \text{Effective diameter of the packing.}$$

$a_p =$ specific surface (sq. ft./cubic ft. packed volume)

Countercurrent flow of liquid and gas through packing

At a fixed gas velocity, the gas pressure drop increases with liquid rate, owing principally to the reduced free cross section available for flow of gas resulting from the presence of the liquid. In the region of **Figure 6** below A, the liquid holdup, i.e., the quantity of the liquid contained in the packed bed, is reasonably constant with changing gas velocity, although it increases with liquid rate. In the region between A and B, as the liquid holdup increases rapidly with gas rate, the free area for gas flow becomes smaller, and the pressure drop rises more rapidly. This is known as *loading*. As the gas rate is increased to B at fixed liquid rate, one of a number of changes may occur:

1. A layer of liquid, through which the gas bubbles, may appear at the top of the packing.
2. Liquids may fill the tower, starting at the bottom or at any intermediate restriction such as a packing support, so that there is a change from gas-continuous liquid-dispersed to liquid-continuous gas-dispersed (inversion).
3. Slugs of foam may rise rapidly upward through the packing.

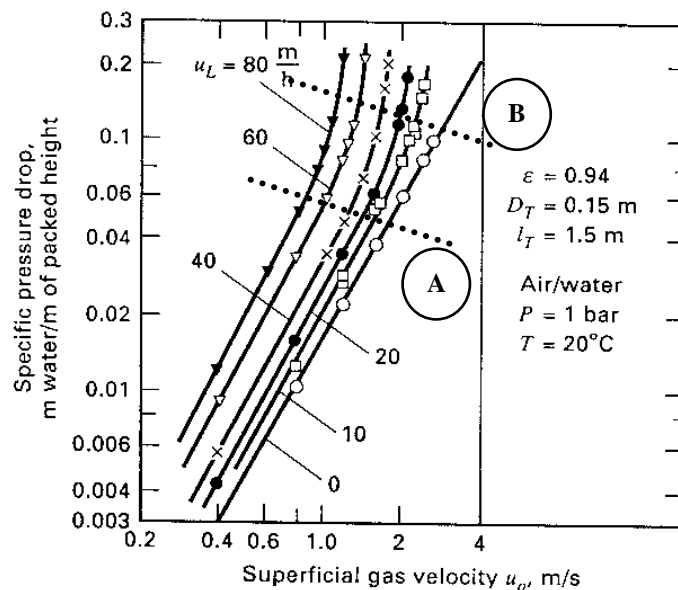


Fig. 6: Typical gas pressure drop for countercurrent flow of liquid and gas in random packing

At the same time, the entrainment of liquid by the effluent gas increases rapidly, and the tower is *flooded*. The gas pressure drop then increases very rapidly. The change from loading to initial flooding can be observed by the change in the slope of the graph or experimentally by the onset of bubbling coupled with the increase in the liquid holdup in the column.

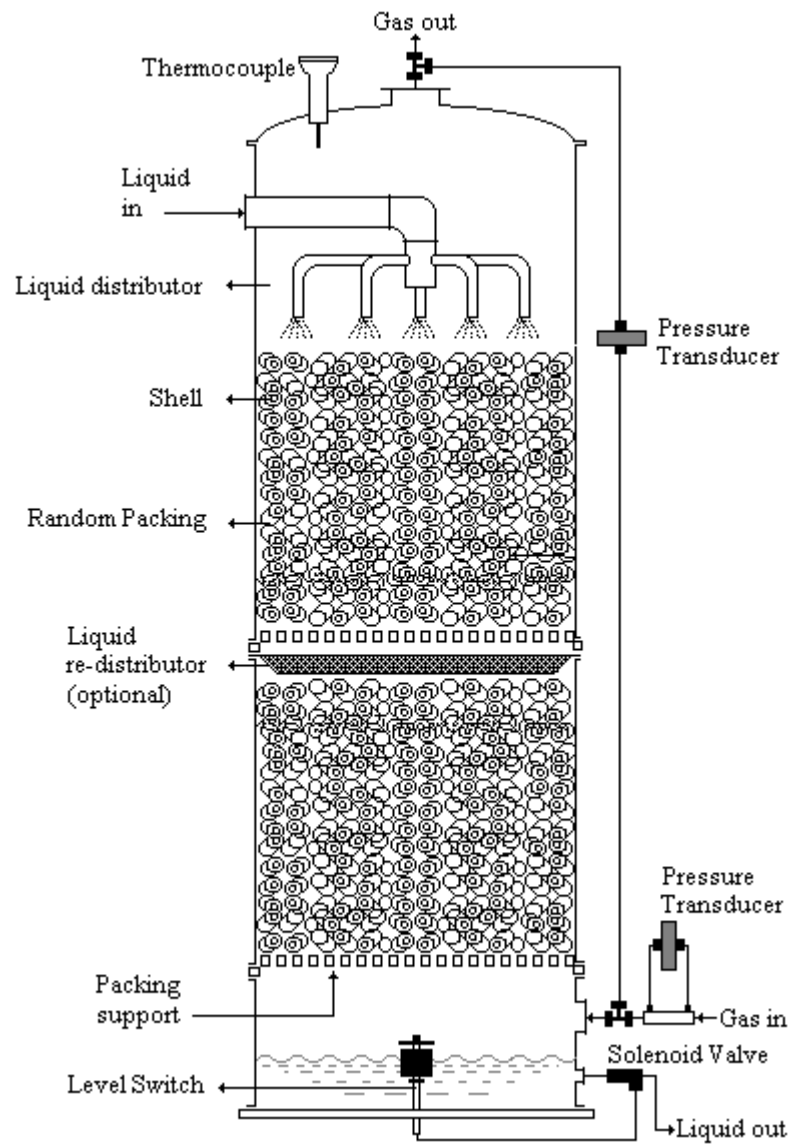



Fig. 7: Experimental Setup

Procedure:

Figure 7 shows the schematic diagram of the experimental setup.

1. Go to **Start>>Programs>>National Instruments>>LabVIEW 6>>Packed Bed Tower**. This will open the menu for the VIs included in the '*Packed Bed Tower.llb*' VI Library. Select '*PackedTower.vi*' and click **OK** to open it.
2. Observe the *Virtual Instrument* for the Packed Bed Tower. You may go through the LabVIEW program code of the Front Panel by clicking **Window>>Show Diagram**. The code is properly commented and is self-explanatory. You should also look at the SubVIs used in the code by double clicking at their icons.
3. Quickly check that all the electrical connections are intact and power supplies switched on. One user may handle the experimental setup and one may operate the *virtual instrument*.
4. Under the '*User Controls*' block of the VI, you can see 3 controls, namely (i) A check-box marked 'This is a dry-bed run', (ii) A Boolean button marked 'Click button to Plot the acquired Data-Point', (iii) A Boolean button marked 'Save the current data-set and switch over to the next'. These are the 3 steps that you have to follow sequentially.
5. Check that under the block titled 'Channels', all the channels are assigned properly, i.e. '*Orifice_Pressure*' for Orificemeter; '*Bed_Pressure*' for Packed Bed; '*zero*' for Cold Junction Compensation and '*thermocouple*' for the thermocouple. Also see that the Resistance value (in ohms) is set to 100. To learn how to configure these channels, refer to '*How to configure virtual channels*' section of this manual.
-  6. Click the **Run** button (shown on left) on the Front Panel toolbar. You can see the acquired data being monitored in the block titled '*Signal Monitoring*' of the virtual Instrument. The values being plotted on the chart are the same as those being monitored by the two Pressure Gages and the Thermometer. In the chart one can monitor the fluctuations and gage's ramp can be used to check that maximum reading of the pressure difference doesn't cross 70 cm of water (see theory on the pressure transducer).
7. Select the check-box in '*User Controls*' block (which means that the condition for dry-bed is TRUE in the program code). This selection makes the program to plot a theoretical curve (from Ergun Eqn.) too, in addition to the acquired points and the 'trend line'. For a dry bed (i.e. no liquid flow across the column) provide some gas flow using the nozzle. You can see the changes in the signals acquired from the tower under the '*Signal Monitoring*' block.
8. Click the button marked '*Click button to Plot the acquired Data-Point*'. You will see that a data point gets plotted on the Graph. This data point is the average of the last N values of the data acquired (N is the number of times the 'for' loop is being iterated, see program code for details). The coordinates of this point are shown in the digital controls marked '*X coordinate*' and '*Y coordinate*'. Now change the gas flow and again click this button. Repeat the process till you get a satisfactory curve.
9. Once you get a satisfactory curve, click the 'Save the current data-set and switch over to the next' button. This will prompt you to save the data-set that you have acquired and a window titled '**Choose File to write...**' will open. Give the file name as per the following format- '**SomeFileName1.xls**' (.xls' extension ensures

that file is saved as Microsoft Excel Spreadsheet). Save the data in a separate directory of your own.



10. Make similar runs for 5 different 'liquid' flow rates. Of course, second run onwards you have to de-select the 'This is a dry-bed run' Boolean Button. Save the different data sets obtained in separate Excel Spreadsheets with different filenames so that at the end of the experiment you have 6 Microsoft Excel Files, each having data set for different liquid flow, of which one is for dry-bed. Click the **Abort Execution** button (shown on left) on the Front Panel toolbar after you are through with the experiment and then exit LabVIEW.

Report Submission :

The data you saved in various Excel Spreadsheets is in the form of two columns. First column is the Gas Mass Flow Rate G' (in $\text{kg/m}^2\text{s}$) and the second column is the Pressure drop per unit packed height $\Delta P/Z$ (in Pa/m). Copy and paste all this data in different worksheets of a single Spreadsheet. Draw the XY (scatter) graph for these data using the Chart Wizard. Your report should include the data in the following format (also include the chart generated by Excel):

Liquid Flow Rate = --- Litres per hour		
S. No.	Gas Mass Flow Rate, G' ($\text{kg/m}^2\text{s}$)	Pressure Drop, $\Delta P/Z$ (Pa/m)
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		

Description of the LabVIEW Program Code

As mentioned earlier, VIs are hierarchical and modular in nature and we can use them as top-level programs, or as subprograms within other programs. The hierarchy of PackedBed.vi is shown in *Figure 8*.

Following is a brief description of the VIs used in the PackedBedTower.llb VI Library. Terms in **Boldface** are either input or output parameters for the corresponding VI.



- **AI Sample Channel.vi** - Measures the signal attached to the specified channel and returns the measured data. The AI Sample Channel VI performs a single, untimed measurement of a channel. **Device** is the device number you assigned to the DAQ device during configuration. **Channel** identifies the analog input channel you want to measure. The default input

is channel 0. See note on configuring channels at the end of this manual. **Sample** contains the scaled analog input data for the specified channel.

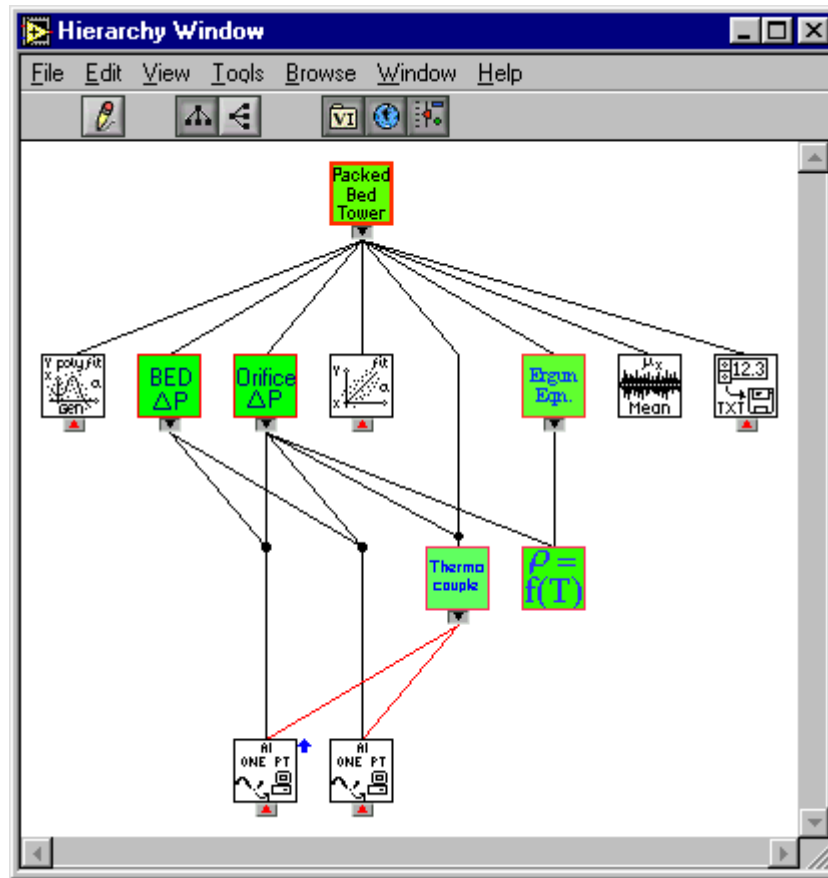


Fig. 8: VI Hierarchy



- **Linear Fit.vi** - Finds the line values and the set of linear coefficients slope and intercept, which describe the line that best represents the input data set. **Y Values** must contain at least two points: n is greater than or equal to 2. If there are fewer than two sample points, the VI sets Best Linear Fit to an empty array. **X Values** must contain at least two points: n is greater than or equal to 2. If there are fewer than two sample points, the VI sets Best Linear Fit to an empty array. **Best Linear Fit** is the calculated values of Best Linear Fit.



- **General Polynomial Fit.vi** - Finds the polynomial curve values and the set of Polynomial Fit Coefficients, which describe the polynomial curve that best represents the input data set. The number of sample points in **Y Values** must be greater than polynomial order. If the number of sample points is less than or equal to polynomial order, the VI sets Polynomial Fit Coefficients to an empty array and returns an error. The number of sample points in X Values must be greater than polynomial order. If the number of sample points is less than or equal to polynomial order, the VI sets Polynomial Fit Coefficients to an empty array and returns an error. **Polynomial order** must be greater than or equal to zero. The default is 2.



- **Mean.vi** - Computes the mean (average) (μ) of the values in the input sequence **X**. If the input sequence **X** is empty, **mean** is NaN. NaN (not a number) represents a floating-point value that invalid operations produce, such as taking the square root of a negative number. **Mean** is the average of the values in the input sequence **X**.



- **Write To Spreadsheet File.vi** - Converts a 2D or 1D array of single-precision (SGL) numbers to a text string and writes the string to a new byte stream file or appends the string to an existing file. **Append to file?** is set to TRUE if you want to append the data to an existing file; you can also set it to TRUE to write to a new file. Set to FALSE (default value) if you want to write the data to a new file or to replace an existing file. **Transpose?** is set TRUE to transpose the data after converting it from a string. The default value is FALSE.



- **thermocouple.vi** – This takes the **Resistance** and channels for **Cold Junction Compensation** and **Thermocouple** as input (*See note on configuring channels at the end of this manual.*). After processing the signals from these, it outputs the **Thermocouple Temperature** in °C. *For further details you may look at the diagram of this VI.*



- **DensityFromTemp.vi** – This takes **temperature** in °C as input and outputs the **density of air** (in kg/m³) as a function of that temperature. *For further details you may look at the diagram of this VI.*



- **OrificePressureDiff.vi** – This, for fed pressure difference across the orifice (through channel for **Orificemeter**), outputs the gas mass flow rate (in kg/m²s) as well as the Pressure Difference (cm of water) across the orifice. *For further details you may look at the diagram of this VI.*



- **BedPressureDiff.vi** - This, for fed pressure difference across the packed bed tower (through channel for **Packed Bed**), outputs the **Pressure Drop across Bed** (in Pa/m) and **Pressure Drop across Bed** (in cm of water).

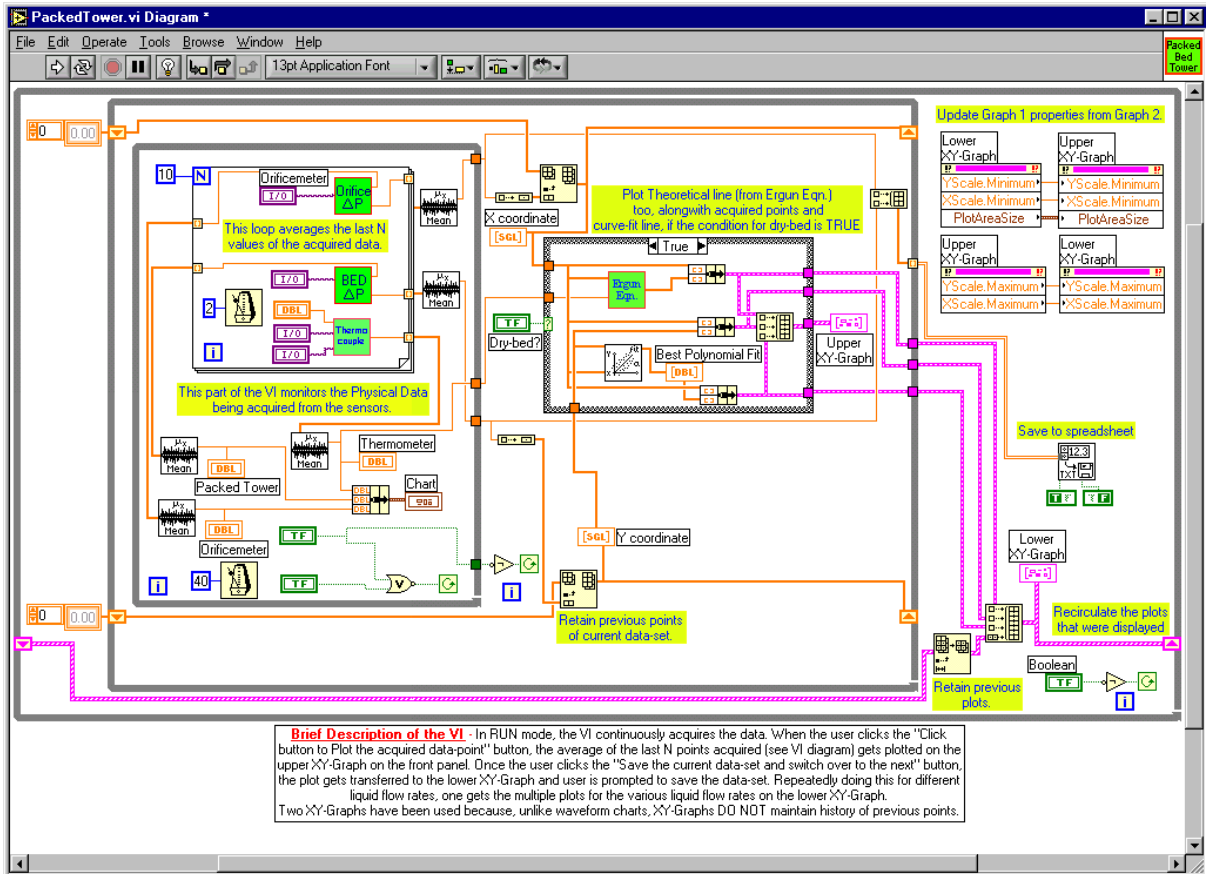
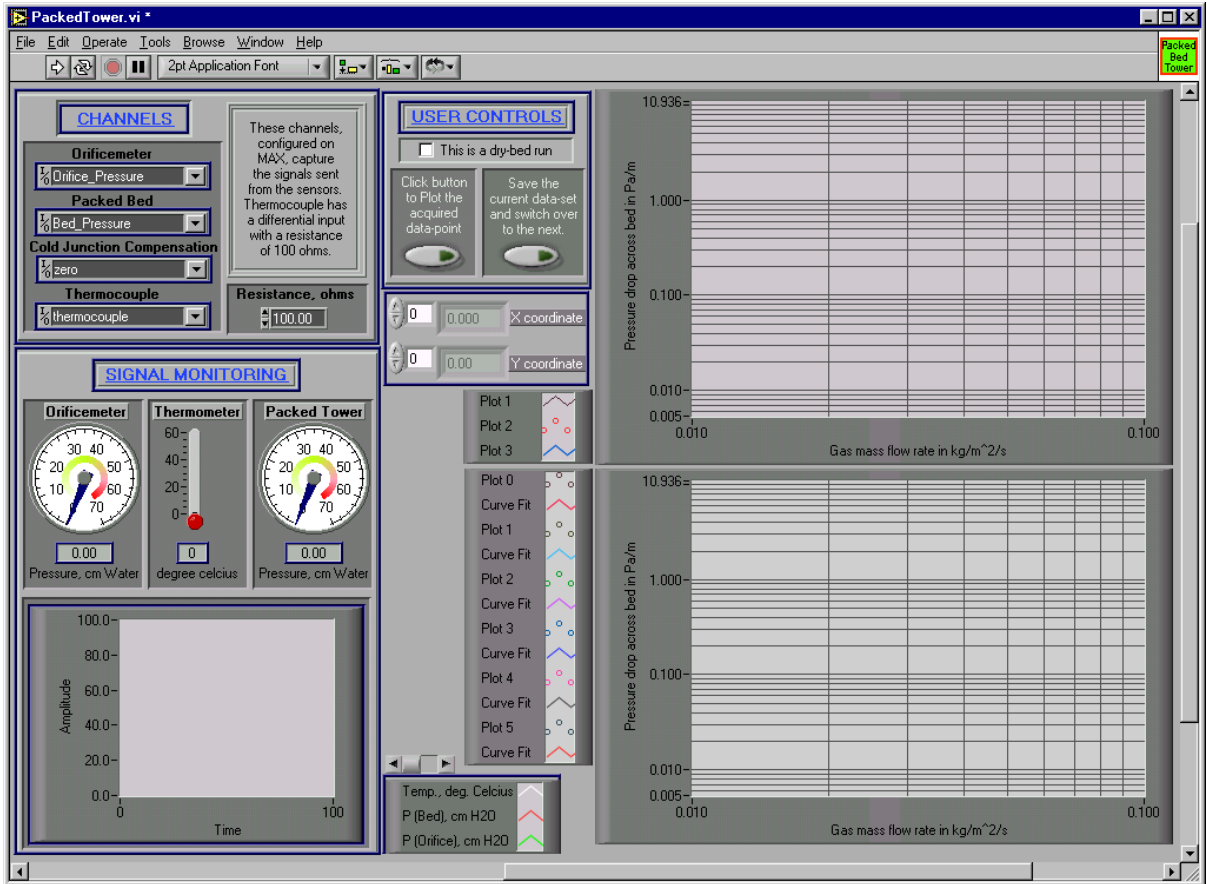


- **ErgunEquation.vi** – This, for fed gas mass flow rate (in kg/m²s), outputs the *theoretical* **Pressure Drop per unit height** (in Pa/m) according to Ergun Equation.

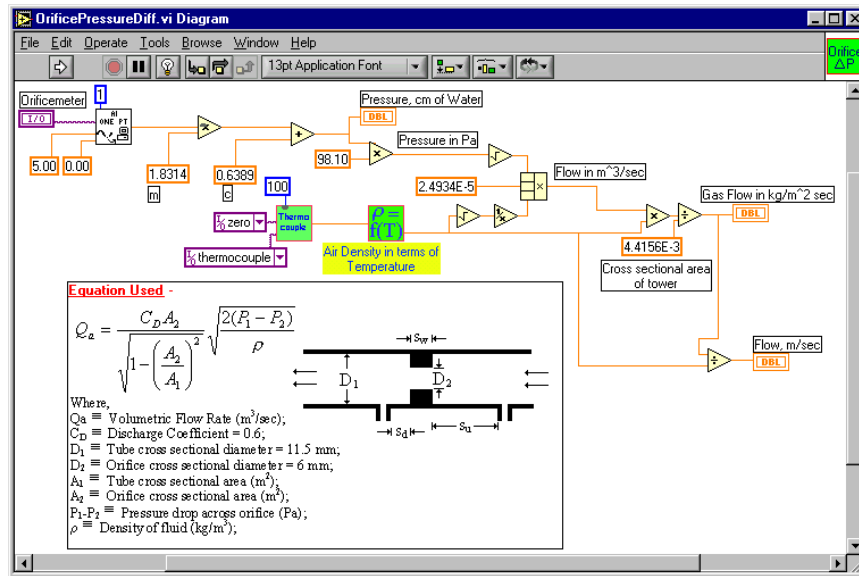
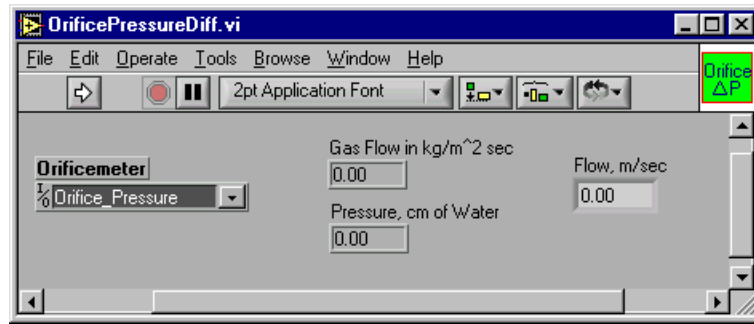


- **PackedTower.vi** – This is the main VI of the PackedBedTower.llb VI library. This VI uses the above listed VIs as SubVIs and they all process the signals acquired by the DAQ card. In RUN mode, the VI continuously acquires the data. When the user clicks the "Click button to Plot the acquired data-point" button, the average of the last N points acquired (see VI diagram) gets plotted on the upper XY-Graph on the front panel. Once the user clicks the "Save the current data-set and switch over to the next" button, the plot gets transferred to the lower XY-Graph and user is prompted to save the data-set. Repeatedly doing this for different liquid flow rates, one gets the multiple plots for the various liquid flow rates on the lower XY-Graph. *Two XY-Graphs have been used because, unlike waveform charts, XY-Graphs DO NOT maintain history of previous points.*

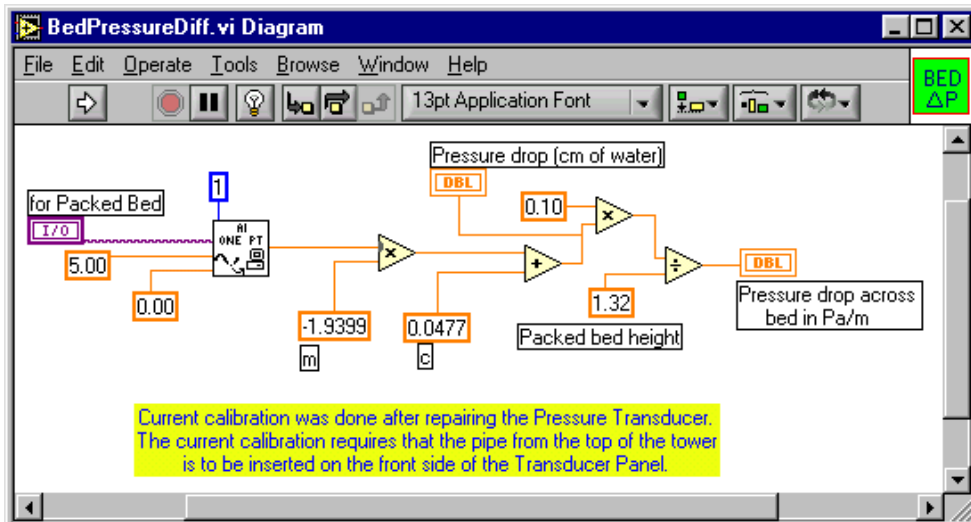
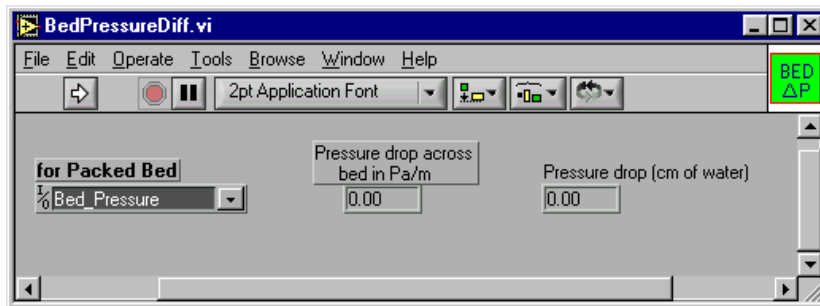
Screen Shots of the Virtual Instruments



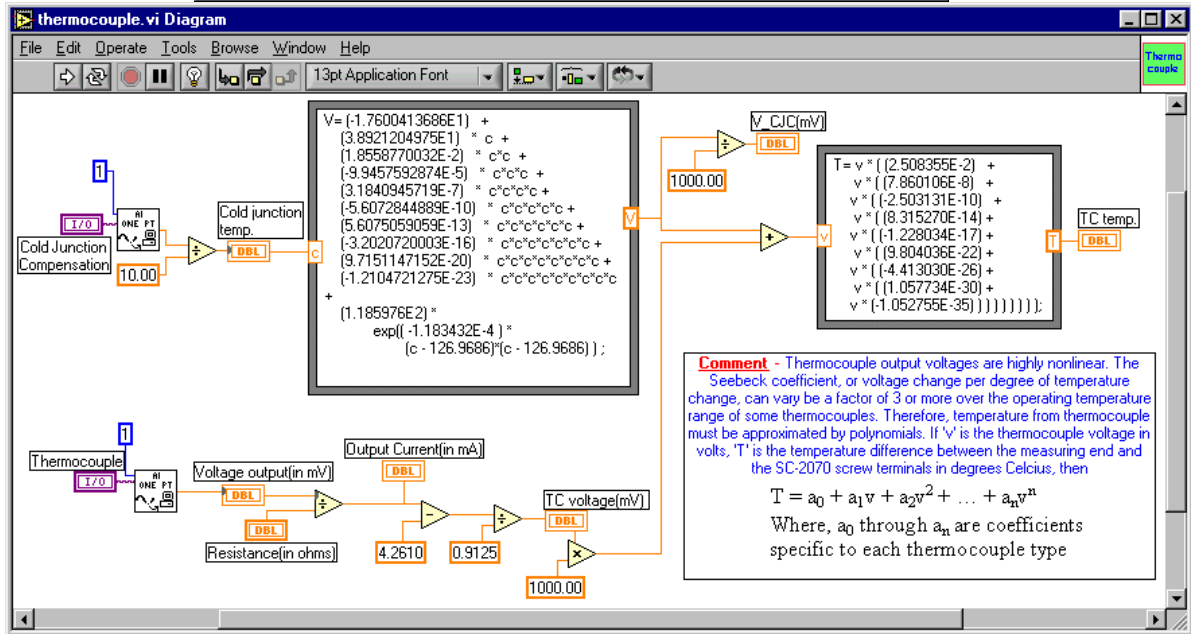
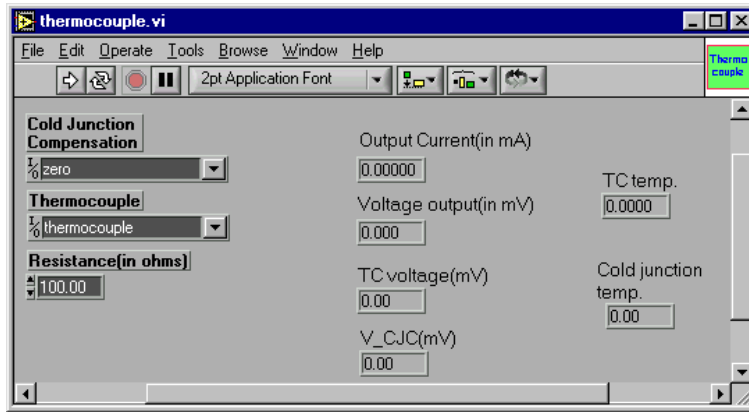
PackedTower.vi



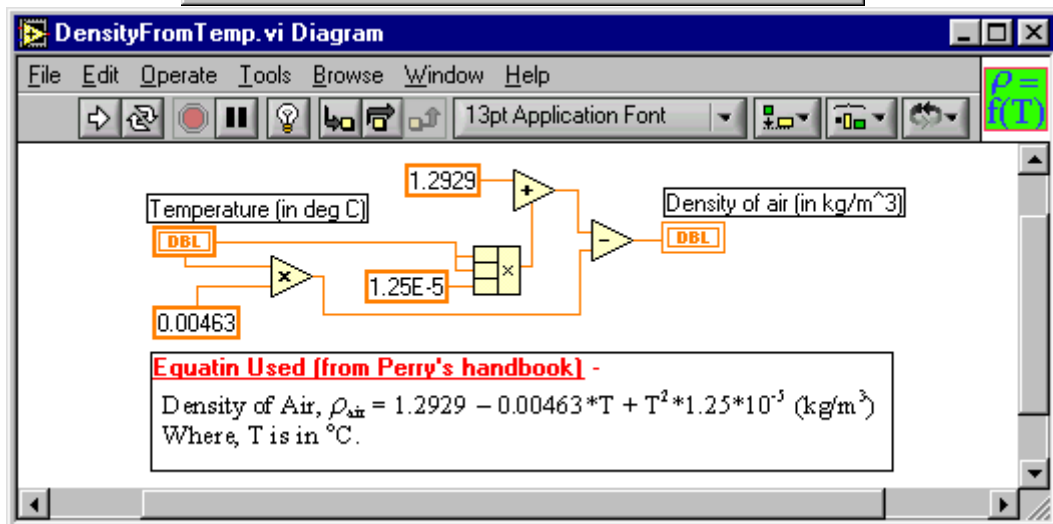
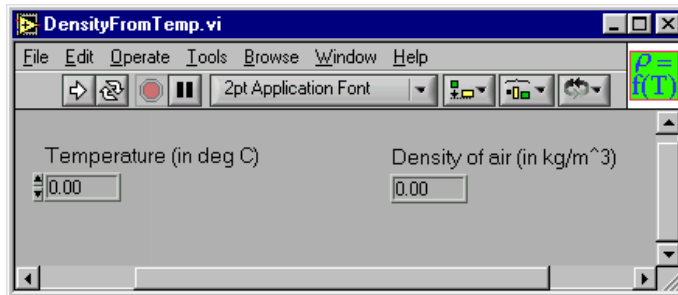
OrificePressureDiff.vi



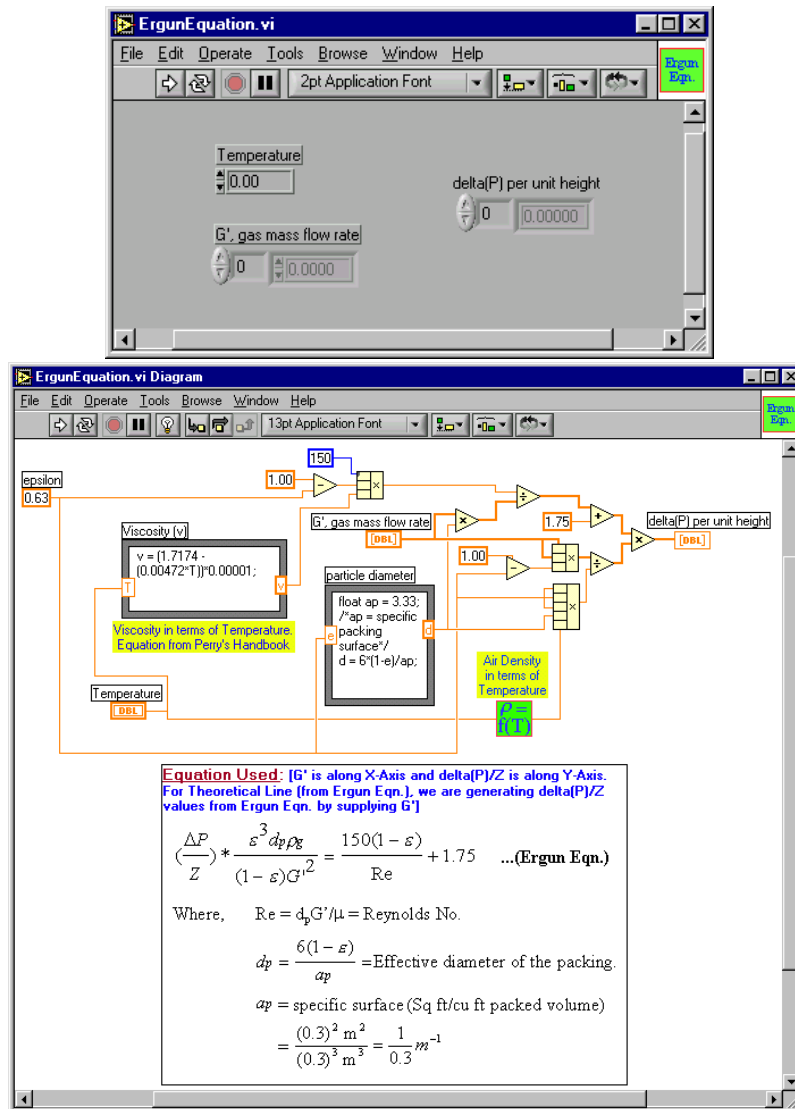
BedPressureDiff.vi



Thermocouple.vi



DensityFromTemp.vi



ErgunEquation.vi

Bugs in the code

- Once you have taken the readings for all the data set (i.e. you are through with your experiment) and you again want to have a run of the experiment, then you must close ALL the VIs and then only start taking new readings. Not doing this causes the previous values stored in the shift registers (they store values of previous iterations in WHILE and FOR loops) of 'PackedTower.vi' to get plotted along with the current values.
- Once you have plotted a data point, it CANNOT be deleted from the XY-Graph. So if somehow a bad data is acquired, it might give a bad shape to your graph. A way to overcome this handicap is that you continue with your acquisition and once the data set is stored in an Excel Spreadsheet, you can easily point out that particular data value(s). *If any such data is acquired, you **must** mention it in your lab report.*

How to configure Virtual Channels?

Note: Channels for PackedBedTower.lib VI library have already been configured and you need not configure them again. But for practice purpose you can experiment with configuring new channels.

With the *DAQ Channel Wizard* you can define what each analog or digital I/O channel is to be used for and define any signal compensation or scaling necessary. Once you have defined a particular channel you can assign it a name so that from any LabVIEW Virtual Instrument signal inputs can be defined more easily and clearly. In addition, all signal compensation and scaling doesn't have to be programmed as it is handled by NI-DAQ.

Configuring Analog Input Channels

While using DAQ hardware, you must configure analog input, analog output, digital input, or digital output channels. You can launch the *DAQ Channel Wizard* from the *DAQ Solution Wizard* to configure channels.

The DAQ Channel Wizard helps you define the physical quantities you are measuring or generating on each DAQ hardware channel. It queries for information about the physical quantity being measured, the sensor or the actuator being used, and the associated DAQ hardware.



You can complete this activity in approximately 15 minutes.

1. Click the **DAQ Solutions** button in the **LabVIEW** dialog box (the gray color dialog window which opens on starting **LabVIEW**) to launch the **DAQ Solution Wizard**. You will have to close all the open VIs to access **LabVIEW** dialog box.
2. When the **Welcome to the DAQ Solution Wizard!** Dialog box opens, click the **Go to DAQ Channel Wizard** button. The **National Instruments- Measurement and Automation Explorer (MAX)** opens.
3. Select and right-click the **Data Neighborhood** view in **MAX**. Click **Create New** from the shortcut menu to configure a new channel. In the **Create New** dialog box select **Virtual Channel** and click the **Finish** button.
4. Select **Analog Input** as the channel type to configure and click the **Next** button. You also can configure analog output and digital I/O in the DAQ Channel Wizard.
5. Type a channel name and channel description in the appropriate text box. Click the **Next** button to continue.
6. Select the type of sensor. If the channel is a temperature measurement, click the check box. Click **Next** to continue.
7. Define the physical quantity that you are measuring. Select the units for your measurement and enter the range for the signal in the appropriate boxes. Click **Next** to continue.
8. Define how the sensor scales the signal from the physical units to the hardware units. Click **Next** to continue.

9. Select the data acquisition device and channel settings. Click **Finish** to configure the analog input channel.
10. Notice that the new configuration is listed under Data Neighborhood. You have finished configuring an analog input channel for your DAQ hardware. Select **File>>Close** to close MAX. LabVIEW is now ready to use the channel you configured.

Data Acquisition Systems

Introduction

Obtaining proper results from a PC-based DAQ system depends on each of the following system elements (see *Figure 9*).

- *The personal computer*
- *Transducers*
- *Signal conditioning*
- *DAQ hardware*
- *Software*

Signal Conditioning

The electrical signals generated by the transducers must be optimized for the input range of the DAQ board. Few types of conditionings performed are:

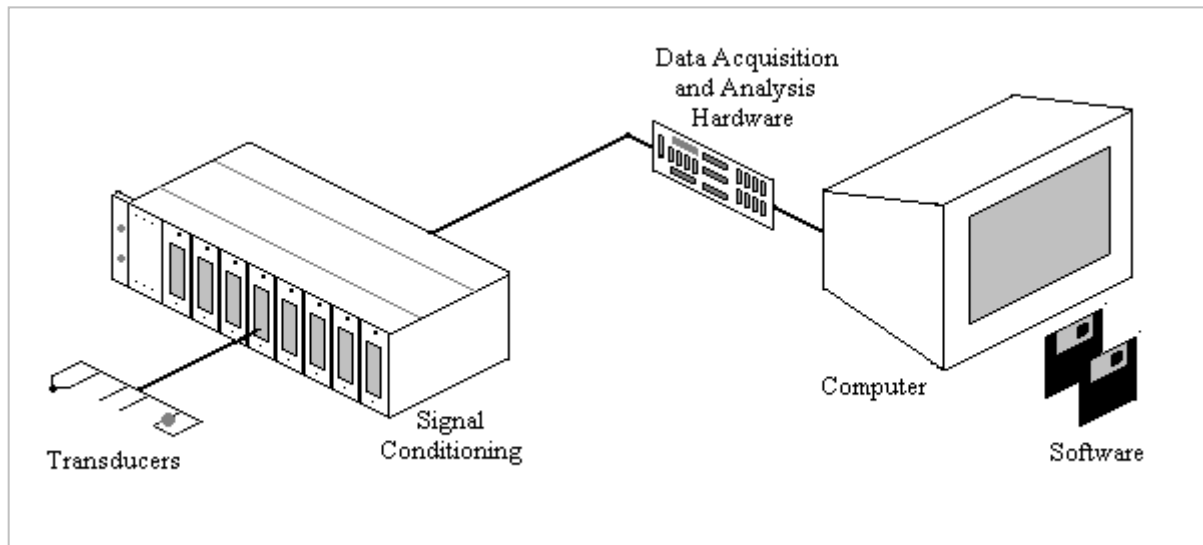


Fig. 9: A typical Data Acquisition System

- *Amplification* – Low-level thermocouple signals, for example, should be amplified to increase the resolution and reduce noise. For the highest possible accuracy, the signal should be amplified so that the maximum voltage range of the conditioned signal equals the maximum input range of the analog-to-digital converter (ADC).
- *Filtering* – The purpose of a filter is to remove unwanted signals from the signal that we are trying to measure. A noise filter is used on DC-class signals such as temperature to attenuate higher frequency signals that can reduce the accuracy of our measurement.

DAQ Hardware

Analog Input - Basic Considerations of Analog Inputs – The analog input specifications can give us information on both the capabilities and the accuracy of the DAQ product. Basic specifications, which are available on most DAQ products, tell us the number of channels, sampling rate, resolution and input range. The number of analog channel inputs will be specified for both single-ended and differential inputs on boards that have both types of inputs.

- *Sampling Rate* – This parameter determines how often conversions can take place. A faster sampling rate acquires more points in a given time and can therefore often form a better representation of the original signal.
- *Resolution* – The number of bits that the ADC uses to represent the analog signal is the resolution. The higher the resolution, the more accurate digital representation of the analog signal is obtained, if the rest of the analog input circuitry is designed properly.
- *Range* – Range refers to the minimum and maximum voltage levels that the ADC can quantize.
- *Noise* – Any unwanted signal that appears in the digitized signal of the DAQ board is noise. Because the PC is a noisy digital environment, acquiring data on a plug-in board takes a very careful layout on multilayer DAQ boards by skilled analog designers. Simply placing an ADC, instrumentation amplifier, and bus interface circuitry on a one or two-layer board will most likely result in a very noisy DAQ board. Designers can use metal shielding on a DAQ board to help reduce noise.

Digital input/output - DIO interfaces are often used on PC DAQ systems to control processes, generate patterns for testing and communicate with peripheral equipment.

Software

Software transforms the PC and DAQ hardware into a complete DAQ, analysis and display system. DAQ hardware without software is useless and DAQ hardware with poor software is almost useless. The majority of DAQ applications use driver software. Driver software is the layer of software that directly programs the registers of the DAQ hardware, managing its operation and its integration with the computer resources, such as processor interrupts, DMA and memory. Driver software hides the low-level, complicated details of hardware programming, providing the user with an easy-to-understand interface.

Developing The System

To develop a high quality DAQ system for measurement and control or test and measurement, we must understand each of the components involved. Of all the DAQ system components, the element that should be examined most closely is the software. Because plug-in DAQ boards do not have displays, the software is the only interface we have to the system. The software is the component that relays all the information about the system, and it is the element that controls the system. The software integrates the transducers, signal conditioning, DAQ hardware and analysis hardware into a complete, functional DAQ system.

References

1. Treybal, Robert E., "Mass Transfer Operations", McGraw-Hill Chemical Engineering Series.
2. Horowitz, P. and H. Winfield, "The Art of Electronics", Cambridge University Press.
3. Peters, Max S. and Klaus D. Timmerhaus, "Plant Design and Economics for Chemical Engineers", McGraw-Hill Chemical Engineering Series.
4. National Instruments Corporation and Omega Engineering, Inc. product manuals.