Virtual Instrumentation

Virtual instrumentation is the use of customizable software and modular measurement hardware to create user-defined measurement systems, called virtual instruments.

→ With virtual instrumentation, engineers and scientists reduce development time, design higher quality products and reduce their design costs.

→ A virtual instrument provides programming and live monitoring from your virtual instrumentation software. A virtual instrument provides all the software and hardware needed to accomplish the measurement or control task.

→ Virtual instrumentation laboratories are associated with the modelling of instruments for a particular application with specified proper range, resolution, accuracy, repeatability.

→ A novel virtual measurement laboratory is based on software support, which provides the tools and methodologies to access the required specification and deliver the inputs and output pattern of the sensor i.e instrument in a Java accepted objects.

→ The applications software for virtual instrumentation is in a internet acceptable Java environment which can accept i.e measuring instrument from the internet and place the developed instrument as an object in the internet service. Hence, the environment of the VI lab is distributed system with networking facility.

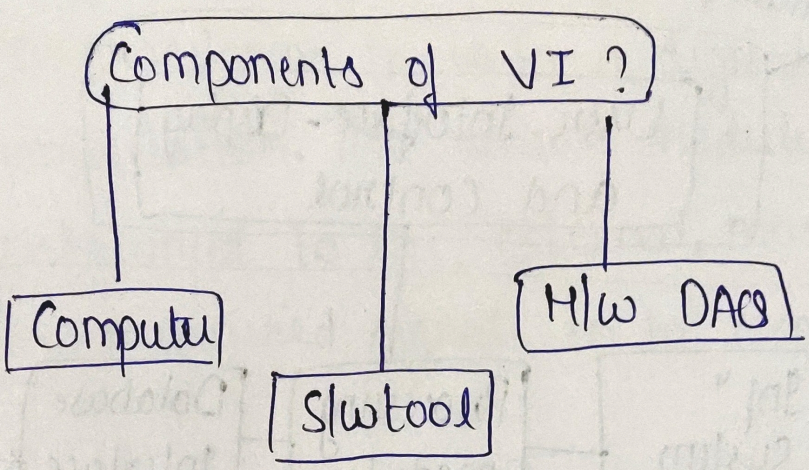
The work aims at experimenting in distributed environment over the Internet using industry based software techniques.

→ This virtual instruments are developed in modular approach and the major idea is the modular design and exploitation of heterogeneous computing platforms for supporting a open and dynamically configurable measurement system.

Traditional Vs Virtual Instruments

Traditional Instruments	Virtual Instruments.
1. Vendor defined	1. User defined
2. Function-specific, stand-alone with limited connectivity	2. Application-oriented system with connectivity to network, peripherals and applications
3. Hardware is the key	3. Software is the key
4. Closed, fixed functionality	4. Open, flexible functionality leveraging off familiar computer technology.
5. Slow turn on technology (5-10 years life cycle)	5. Fast turnon technology (1-2 year life cycle)
6. Minimal economies of scale	6. Maximum economies of scale
7. High development and maintenance costs.	7. Sw minimizes development and maintenance costs.

Components of VI



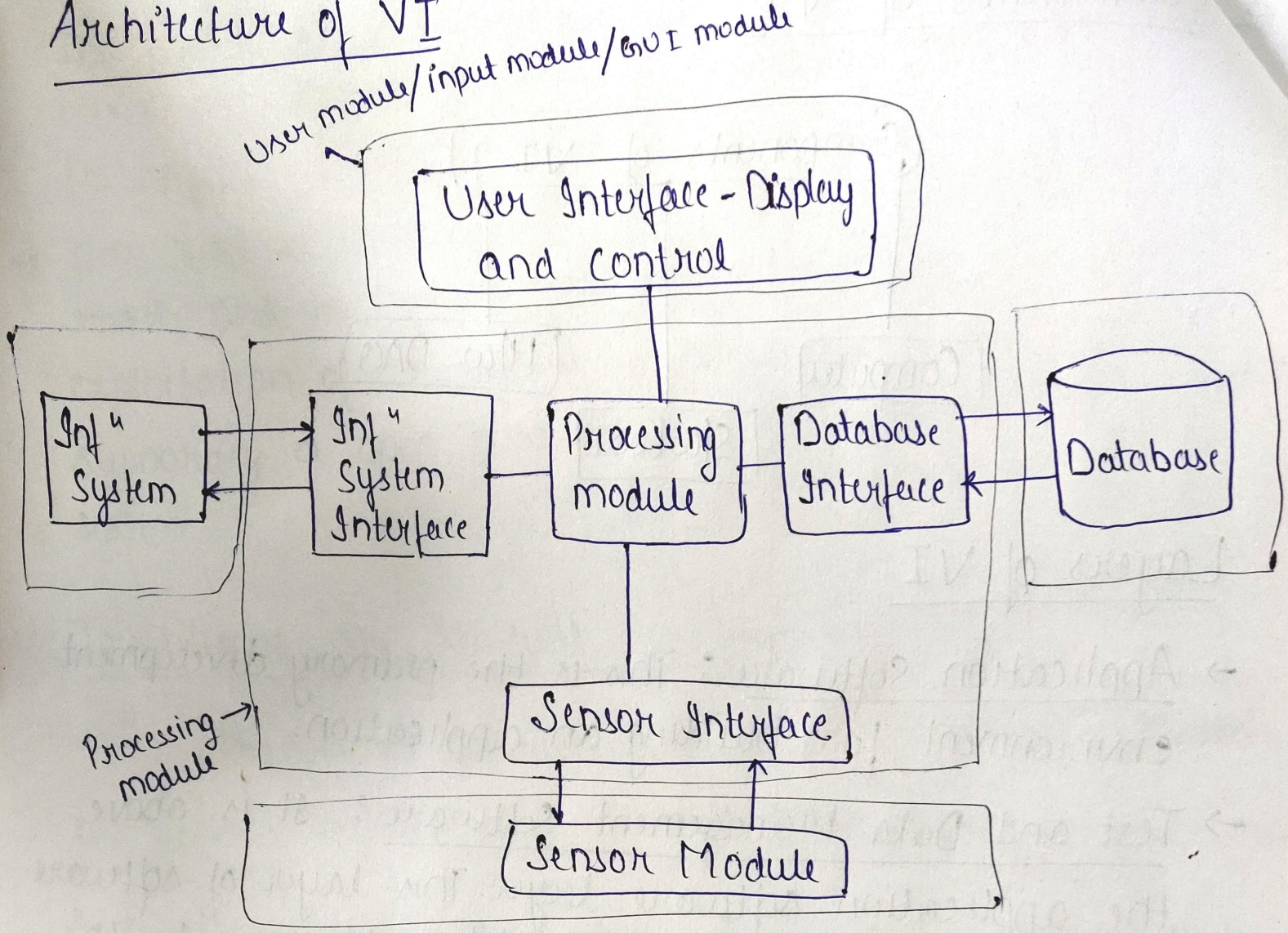
Layers of VI

→ Application Software : This is the primary development environment for building an application.

→ Test and Data Management Software : It is above the application software layer. This layer of software incorporates all the functionalities developed by the application layer and provide system-wide data management.

→ Measurement and Control Service Software ; The last layer is often overlooked, yet critical to maintaining software development productivity.

Architecture of VI



A virtual instrument is composed of the following blocks;

* Sensor Module
* Sensor Interface
* Processing Module
* Database Interface and
* User Interface

Sensor Module: The sensor module detects physical signal and transform it into electrical form, conditions the signal, and transforms it in to a digital form for further manipulation. The sensor module interface a virtual instrument to the external, mostly analog world transforming measured signals in to computer readable form.

→ Sensor Interface: There are many interfaces used for communication between sensors modules and the computer. The interface include wired and wireless interface. Wired interfaces are usually standard parallel interfaces, such as General purpose Interface Bus (GPIB), Small computer systems interface (SCSI), system buses (PCI extension for instrumentation PXI or VME Extensions for instrumentation VXII), or serial buses.

→ Processing Module: The integration of microprocessors/ microcontrollers of general use allowed the flexible implementation of sophisticated processing functions. The processing function used in virtual instrumentation can be classified as analytical processing and artificial intelligence techniques.

→ Presentation and control:

→ Presentation and control: Presentation and user control should be user-friendly and easily understandable. According to presentation and interaction cabapilities.

We can classify interfaces used in virtual instrumentation in four groups:

terminal user interface

graphical " "

multimodal " " and

virtual and augmented reality interface.

Summary

1. Sensor Module
   - → electrical
   - → Non-electrical
2. Sensor interface
3. Processing module
   - → Analytic processing
   - → Artificial intelligence
4. Database interface
5. Inf^{n} system interface
6. Presentation and control
   - → Terminal user interface
   - → GUI
   - → Multimodal presentation
   - → Virtual and augmented reality
7. Functional Integration.

Advantages of Virtual Instrumentation

Engineers and Scientist whose needs, applications and requirements change very quickly, need flexibility to create their own solutions. VI fulfill there requirements without replacing entire device because of the appl^{n} software installed on the PC and the wide range of available plug in hardware.

→ Lower costs of instrumentation
→ Portability between various computer platforms
→ Easy-to-use graphical user interface.
→ Flexibility
→ Plug-In and Networked Hardware.
→ Lowering the cost of Hlw and slw.
→ Minimizing set-up and configuration time costs.
→ Decreasing appl^{n} software development Time costs.
→ Graphical representation of program structures
→ Code can be compiled to standalone .exe or .Dll file

The major benefits of VI are as follows:

1. Performance
2. Platform-independent nature
3. Flexibility
4. Lower cost
5. Plug-in and networked hardware
6. Low hlw and slw
7. Set-up and configuration time less
8. Appl^{n} software development less time
9. Data Security.

Disadvantages of VI

→ Security: Sensitive information may be accessible to public users.

→ Power Consumption: demands that many device run simultaneously and consume a lot of power.

Graphical Programming Techniques

- Graphical programming or visual programming is a technique of programming where visual block connection are used to code instead of text making it, easy for non coders to implement algorithms.
- Graphical languages usually are developed using a graphical interface, where elements are selected and the underlying section, where functionality is added.
- The details are dependent on the language.
Graphical programming methods provide user with a graphical environment to use the instrument easily.

Text Based Programming

1. Syntax must be known to do programming.
2. The execution of the program is from top to bottom
3. To check for the error the prog. has to be compiled or executed
4. Front panel design needs extra coding or needs extra work.
5. Text based programming is not interactive

Graphical Programming

1. Syntax is knowledge but is not required for programming
2. The execution of the program is from left to right
3. Errors are indicated as we wire the blocks.
4. Front panel design is a part of programming
5. Graphical programming is highly interactive

6. This is the text based programming where the programming is not conventional method	6. The programming is data flow programming.
7. Logical errors findings is easy in large programs	7. Logical error findings in large programs is quiet complicated.
8. Program flow is not visible	8. Data flow is visible.
9. It is text based programming.	9. It is icon based programming & wiring
10. Passing parameters to sub routine is difficult	10. Passing parameter to sub VI is easy.

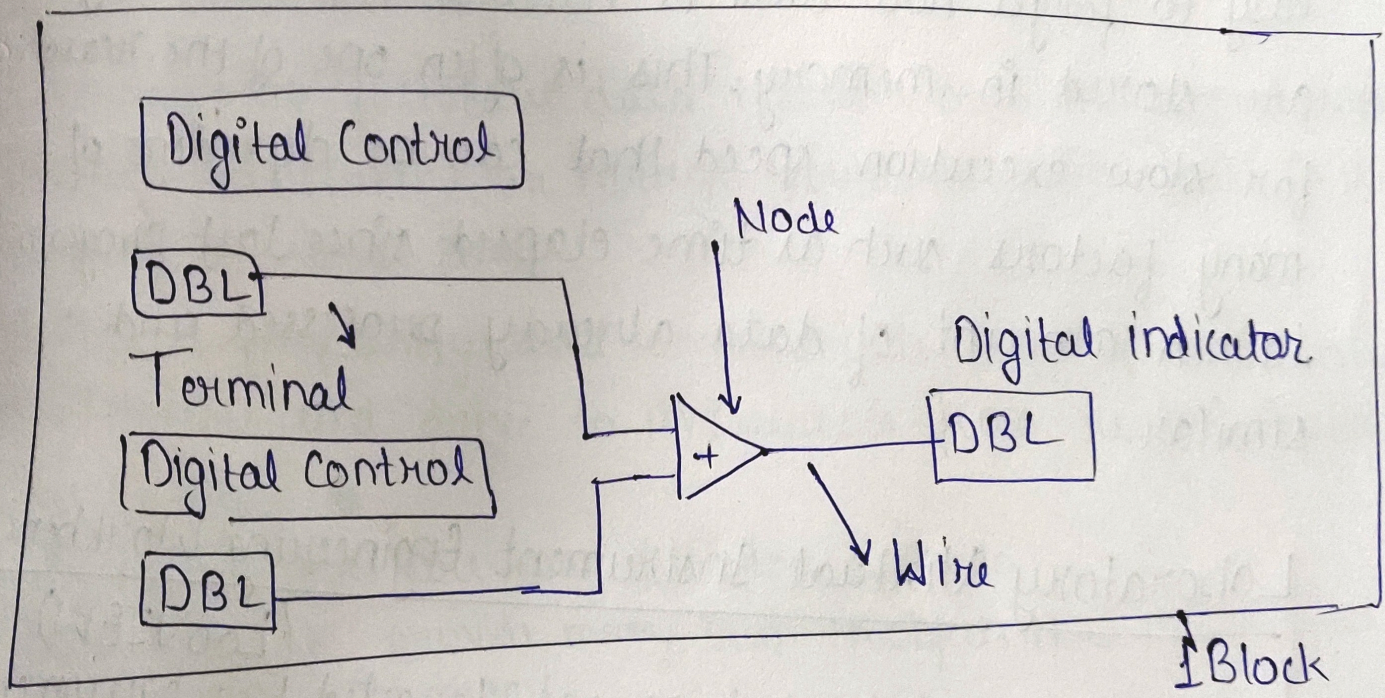
→ An efficient LabVIEW application is designed without unnecessary operations, with minimal memory occupation including code, data, block diagram and front panel, GUI updates and data manipulations.

→ The VI profiling tools in LabVIEW offer a detailed overview of execution time and memory occupation of the main VI and its SubVIs and should be used to find and correct all execution issues of designed application.

LABVIEW programs are called Vis (Virtual Instruments)

→ Front panel is a user interface of VI. We build the front panel with controls and indicators that:

1. Controls stimulate the instruments inputs devices.
2. Indicators stimulate the instruments output devices.
3. Block diagrams - Every control on a front panel has a corresponding terminal on blocks diagrams.
4. Wires connect each of the nodes with block diagrams including control and indicator terminals, functions and structures



Two Constraints Only

1. Execution order: The execution order is solely defined by dataflow and not by position of the nodes in the diagram as one would intuitively presume. If no data dependency exists to determine the dataflow, a sequence structure can be implemented for that purpose.
2. Execution speed: When working with LabView it is quite easy to forget how data is handled and how they are stored in memory. This is often one of the reason for slow execution speed that can be dependent of many factors such as time elapsed since last program launch, amount of data already processed and similar.

Laboratory Virtual Instrument Engineering Workbench (LabVIEW)

* LabVIEW is a graphical language targeted for equipment's monitoring and control Not using statements, it is programmed using graphic controls.
* LABVIEW is a product of National Instruments (NI). LabVIEW was designed to support a laboratory environment and is targeted to applications to control and monitor equipment

* Labview is a graphical programming language that uses icons instead of lines of text to create applications.
* Labview programs are called virtual instruments because their appearance and operations initiates physical instruments like digital multimeters.
* Labview is used to test various industrial automations.
* Using DAQ divices and labview to monitor a temperature send signals to an external systems or determine the frequency of an unknown signals
* Labview also facilitates data transfer over the general purpose interface bus (GPIB), or through your computers built in USB, ethernet or serial port
* GPIB is frequently used communicate with scanners and multimeters and drive to instruments from remote locations.

Advantages

* These languages are easy to learn and use.
* These language provide many built in objects that can be used in developing new programs.
* New objects can also created.
* The users can use virtual appl^{n} very easily.
* The language provide facility to attach code to each interface component. The attached code is executed when the user interact with the interface components.

Disadvantage

1. These language require computer with more memory, high storage capacity of hard disk and faster processor.
2. These languages can only be implemented on graphical operating system like Linux and Windows.

Advantages of Virtual Instrumentation Techniques

* It is software concentric
* Data handling capacity is easier and better
* Data can be displayed in effective form so it is convenient for operator.
* Due to computers, data analysis is easy & quick.
* Powerful software is available with new operating system. So this is portable.
* It can work for customized application effectively.
*

Data Types

Data type is a data storage format that can contain a specific type or range of values. When computer program store data in variable, each variable must be assigned a specific data type.

Some common data types include :-

Integer
Floating Point Number
Characters
Strings
Arrays

Datatype in LabView

→ Denotes the type of variable or data that can be used in LabView
→ Datatypes cannot be interconneted.
→ There are some unique colors assigned to different data types
→ Commonly used data types in LabView.

Boolean	Cluster
Numeric	Error Cluster
String	Waveform
Array	ENUM

Boolean DataTypes

Consist of only two values
True (1)
False (0)

→ Provide o/p in the form of 0 or 1
→ Indicated by Green data wire.
→ LabVIEW stores the boolean data as 8 bit values

Numeric Data Type

- Represented as floating point numbers, complex numbers, signed - unsigned integers and fixed-point numbers.
- Integer either signed or unsigned are indicated by Blue data wire
- Double and single precision and complex numbers are represented by Orange data wires
- Only difference in numeric data is determined by the type of their values and the number of bit they stored.

String DataTypes

- Any sequence of displayble and non-displayable ASCII Characters in the LabVIEW data is known as string data type.
- It is indicated by Pink data wires
- Some of the common appl^{n} of string are
  1. Controlling instrument by sending text commands to instruments and returning data in the form of either ASCII or binary string which we can then convert to numeric values.
  2. Store numeric data to disk.

Arrays

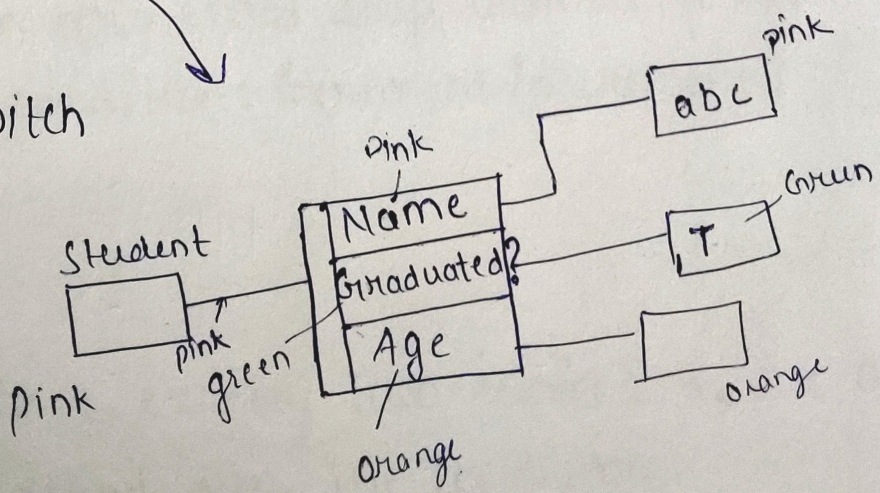
- Group of specific data types.
- Determined by thicker data wires.
- Stores numeric, string, double or even Boolean data types.
- Size of array increase according to the no of elements

Wire Type	Scalar
Numeric	Floating-point → orange color
	→ blue color
	integer
Boolean	..... → green

Clusters

Consists of different type of data in a single unit.
Indicated by thicker BROWN color for numeric and PINK color data wire for other.

3 data types: eg
- a string
- a Boolean switch
- a numeric



Graphs / Waveform Datatypes

The waveform data type that is used to store and display periodic signal measurement i.e the waveform creates a graph and charts of the particular data. It provide us with the exact and precise information about data charts and graphs form.

Enumerated DataType

Enum are the combination of data types mainly consisting of a pair of data values i.e a string and a numeric value.

For example, we create an Enum named as 'Month'. The value pairs in the field can be placed as January-1 or February-10 and so on.

Loops

→ It is a process where a set of instructions or structure are repeated in a sequence a specified number of times or the until a condition is made.

→ When a set of instruction is executed again it is called loop.

→ Loop are also called iteration.

→ Programmer use loops to cycle through values, sum of two numbers, repeat functions and many other things.

Types of Loops

1. For Loop
2. While Loop

For Loop

→ For loop is used to execute a set of statements repeatedly until a particular condition is satisfied.

→ We can say it is an open ended loop. General format is,

```
for (initialization; condition; increment/decrement)
{
 statement;
}
```

→ In for loop we have exactly two semicolons, one after initialization and second after the condition.

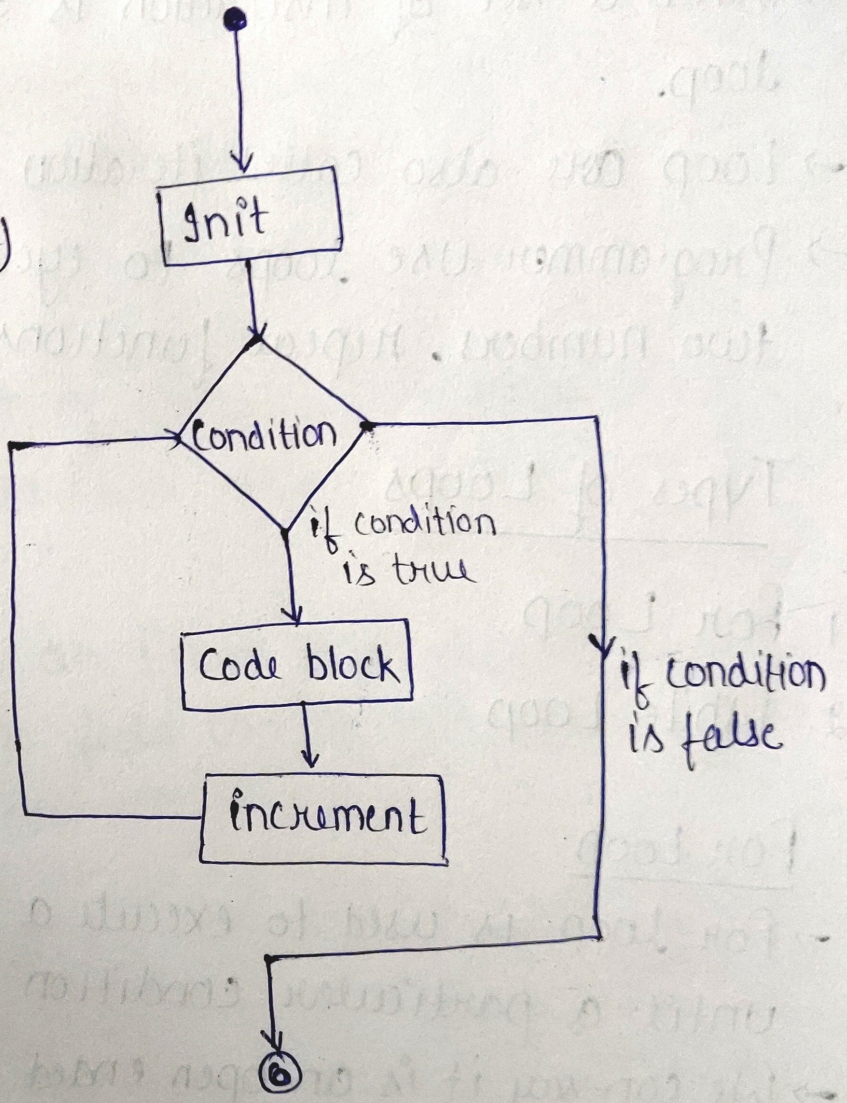
→ In this loop we can have more than one initialization or increment/decrement, separated using comma operator

→ But it can have only one condition

For loop

Flow Chart

```
for(init; condition; increment)
{
  Conditional code:
}
```



While loop

→ A while loop programming repeatedly executes a target statement as long as a given condition is true.

→ Format:

```
while (condition)
{
  statement(s);
}
```

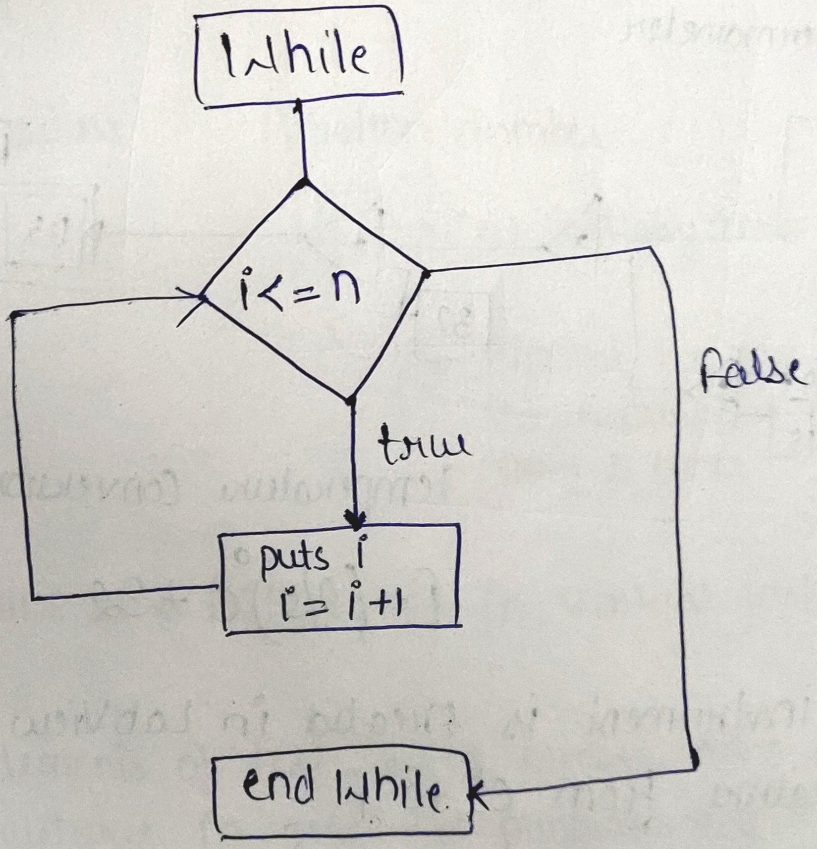
→ Here, statement(s) may be a single statement or a block of statements.

→ The condition may be any expression, and true is any non zero value.

→ The loop iterates while the condition is true.

→ When the condition becomes false, the program control passes to the line immediately following the loop.

Flow chart



Using Loops in VI's

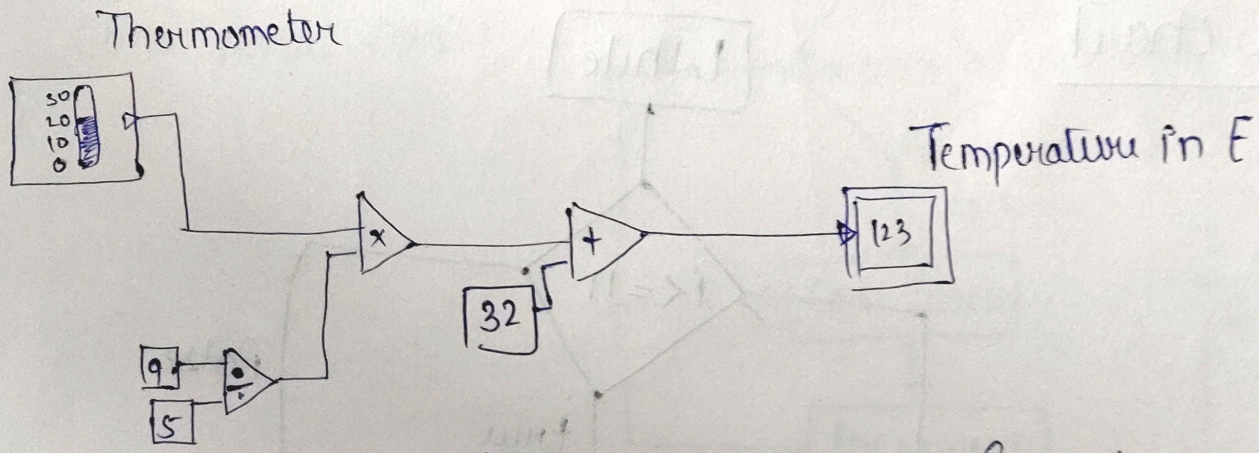
→ Lab View consist of for and while loop.

→ Loops are used to control repetitive operations in virtual instruments.

→ These are used to perform an action frequently with variation in details each time.

→ The virtual instrument is working absolutely fine but the only thing is, we have to run the program again and again for each value of the temperature in °C, which is not a drawback, but is also not convenient.

→ So here comes the use of loop in virtual instruments.

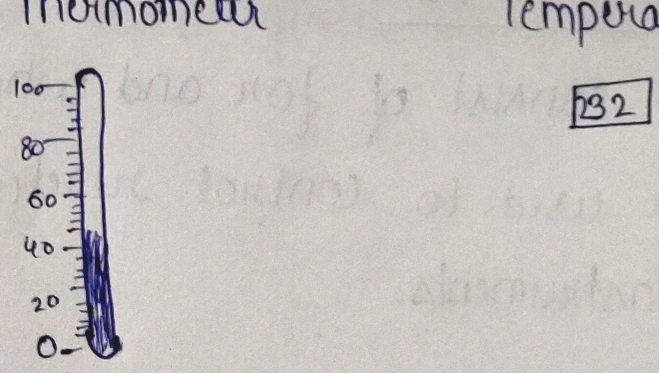


Temperature conversion formula

$$F = (9/5)^{\circ}C + 32$$

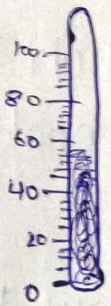
This instrument is created in LabView software which convert Temperature from C° to °f

Front Panel



explanation

Save using while loop, using button (stop).



Thermometer

Temp. in F

212

stop

Using for loop

→ A for loop executes a sub-diagram, a set number of times.

→ add no of iterations

No. of iterations

5

iteration number

4 → (0) initially zero

↳ (eg- suppose we measure temperature, it iterates only 5 times)

The Concept of WHILE and FOR loops in virtual instruments using labview.

→ Loops and case statements of text based programming language are presented as structures in graphical programming. Repetition and loop are used to perform an action frequently with variations in the details each time. Labview consists of "FOR" and "WHILE" loops. These are used to control repetitive operations.

While loop - A while loops execute all or a portion block diagram code multiple times.

The conditional terminal :- Continue if true condition the while loop will continue to repeat the code until a boolean value of false is passed to the condition terminal

Stop if true condition, the while loop will continue to repeat the code untill a boolean value of true is passed to the conditional terminal.

The interaction terminal :- Output the number of times the loop executes.

a) Controlling a while loop - The conditional terminal is wired to a boolean control (switch) on the front panel for the user to control the operation of the while loop.

For loops :-

The for loop repeats block diagram code a predetermined number of counted times. When the number of iterations equal to the predetermined count, the loop stops.

Structures

Structures are graphical representation of the loops and case statements of Text based programming languages.

In text based programs, this can be accomblished with statements like If, else, Case.

In graphical progeaming, it is one using sub diagrams and with conditions. (Decision making with the select a function)

Structures to Execute Processes in a Block Diagram are

Case Structure

Sequence Structure

Event Structure

Timed Structure

Case Structure

→ It contains multiple sub-diagrams, only one of which executes depending on the input value passed to the structure.

→ Boolean Case Structure

eg

If TEMP Scale is TRUE, execute True Case;

If the TEMP scale is FALSE, execute false case

→ We can ENTER A SINGLE VALUE OR LIST and Range of values in the Case Selector label

→ We can create multiple Input and output tunnel for a case structure.

Sequence Structure

→ Contains one or more subdiagrams, frames, that execute in sequential order.

→ The flat sequence structure Displays all the frames at once and executes the frames from right to left and when all the data value leaves each frames after its execution.

→ The stacked sequence structure Returns data only after the last frame executes.

Event Structures

→ It waits until an event happens then executes the appropriate case to handle that event.

→ An event is an action that triggers a change in state.

* User initiated
* Button press
* Mouse click

→ Os Initiated

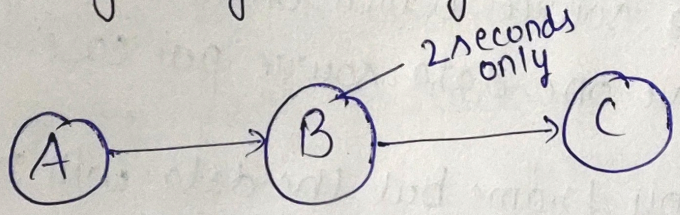
* timeout

→ Software initiated

* Message from another program
* Variable reaches a specific value

Timed Structures

* The timed sequence structure consists of one or more task subdiagrams, or frames, that executes sequentially and can be timed with an internal or external timing source.
* Timed sequence are used when we want to develop VIs with precise timing, execution feedback, timing characteristics that changes dynamically.



Formulae Nodes

→ The Formula node is a convenient text Based node we can use to perform mathematical operations on the block diagram.

→ Formula nodes are useful for equations that have many variables or are otherwise complicated and for using existing text based code.

Case Structures

The case structure is a method of executing conditional statements. The case structure is similar to, if --- then --- else statements in conventional programming language.

eg. Case structure is configured like a deck of cards. Lab view only executed is determined by the value wired to the selector terminals

Sequence structure

It is used to control the order of execution of nodes that aren't data dependent on each other. The nodes within each frame are data dependent. The output tunnels of sequence structure can only have data one data source which unlike case structures has output that must have one data source per case.

The o/p can come from any frame but the data will not leave the structure has completed its execution. Data input is available in all frames.

eg. Thermometer, the temperature is passed between frames of a sequence.

Formula node :-

It is a box where we enter algebraic formula directly into block diagram.

It is useful when an equation is complicated or has many variables.

eg. To implement a expression

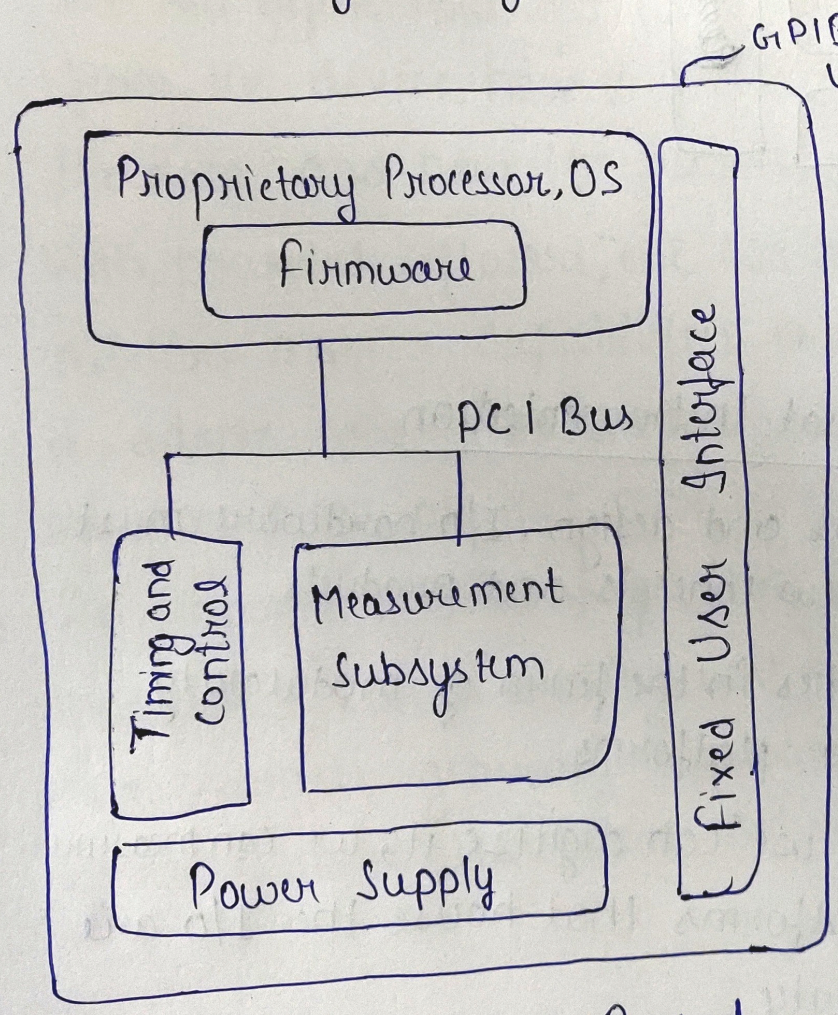
$$y = x^2 + x + 1$$

we have to write it as y = x^2 + x + 1

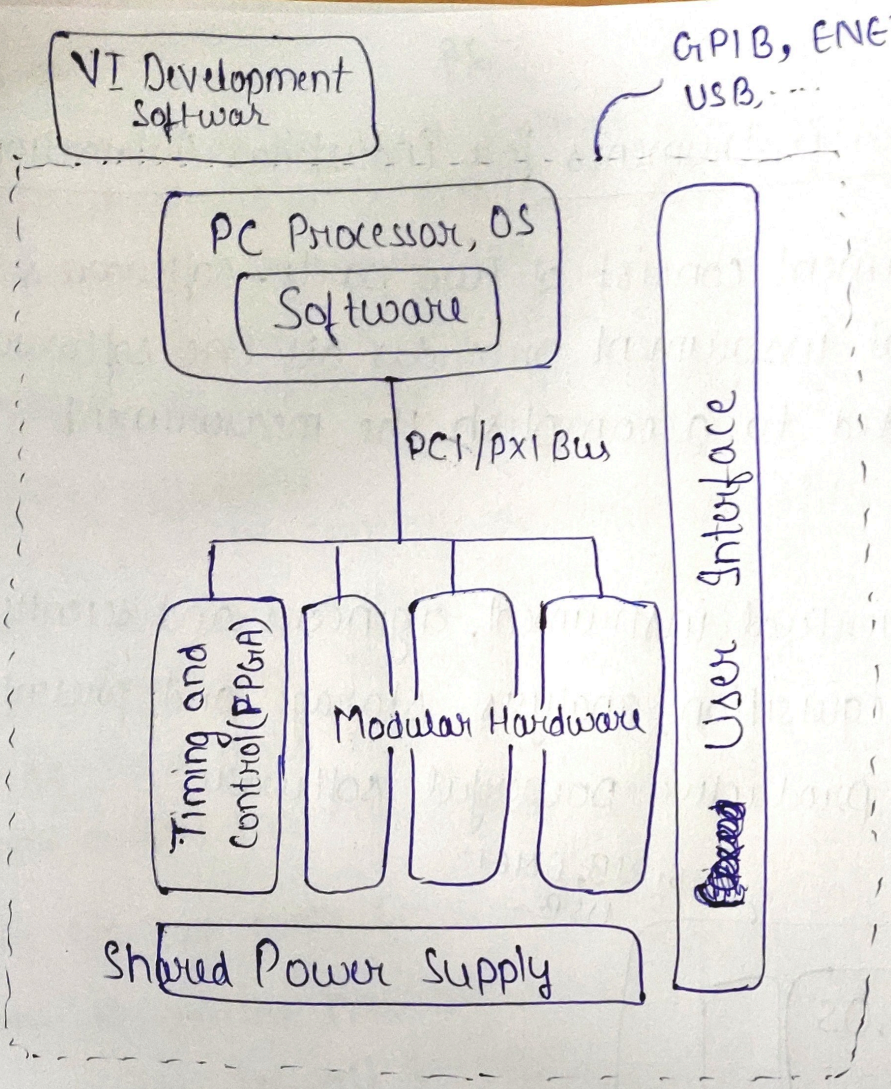
Need of Software based instruments for Industrial Automation

Every virtual instrument consist of two parts - software & hardware. A virtual instrument provides all the software and hardware needed to accomplish the measurement or control task.

In addition, with a virtual instrument, engineers and scientist can customize the acquisition, analysis, storage and presentation functionality using productive powerful software.



Tradinal Based



VI Based

Role of Hardware in Virtual Instrumentation

→ To accelerate test, control and design, I/O hardware must be rapidly adaptable to new concepts and products

→ VI delivers this capabilities in the form of modularity within scalable hardware platforms.

→ VI is software-based; if we can digitize it, we can measure it. Standard hardware platforms that house the I/O are important to I/O modularity.

→ Laptops and desktop computers provide an excellent platform where virtual instrumentation can make the most of existing standards such as the USB, PCI, Ethernet and PCMCIA buses.

→ Software is the most important component of a virtual intrument. With the right software tool, engineers and scientists can efficiently create their own applications by designing and integrating the routines that a particular process requires.

→ We can also create a appropriate user interface that best suits the purpose of the application and those who will interact with it.

→ We can define how and when the application acquires data from the device, how it processes, manipulates and store the data, and how the results are presented to the user.

→ With powerful software, we can build intelligence and decision-making capabilities into the instrument so that it adapts when measured signals change inadvertently or when more or less processing power is required.

→ An important advantage that software provide is modularity, when dealing with a large project engineers and scientists generally approach the task by breaking it down into functional solvable units.

The main stream adoption of software-defined modular instruments in automated validation and production test applications is confirmation of the on-going trend of flexibility and low cost necessity of next generation measurment computation system.

The important needs are as follows

* Rapid computer advancement
* low cost
* high-performance data conversion
* Flexibility
* Modernization of electronic devices
* Range and Calibration
* Design complexities.