Introduction to NI LabVIEW and Computer-Based Measurements

<Insert Name Here>  
<Insert Title Here>  
National Instruments
Today, We’ll Explore:

<table>
<thead>
<tr>
<th>The Challenges of Making Measurements</th>
<th>Characteristics of Mixed-Measurement Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The National Instruments Approach</td>
</tr>
<tr>
<td></td>
<td>Architecture of a Measurement System</td>
</tr>
<tr>
<td>Introduction to LabVIEW</td>
<td>Navigating the LabVIEW Environment</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Fundamentals of Data Acquisition</td>
<td>Essential Data Acquisition Concepts</td>
</tr>
<tr>
<td></td>
<td>The Basics of Signal Conditioning</td>
</tr>
<tr>
<td></td>
<td>The Value of National Instruments Hardware Platforms</td>
</tr>
<tr>
<td>Uniting Software and Hardware</td>
<td>Architecture of the NI-DAQmx Driver</td>
</tr>
<tr>
<td></td>
<td>Measurement Services and Utilities</td>
</tr>
<tr>
<td></td>
<td>Exploring and Using the NI-DAQmx API</td>
</tr>
</tbody>
</table>
The Challenges of Making Measurements

Exploring the Traditional Approach to Measurements
The Origin of Automated Measurements

- Traditional pen-and-paper approach
- Redundant circuitry between instruments (e.g., displays)
- Manual data recording and analysis
- Error-prone processes
- Difficult to reproduce or redo

<table>
<thead>
<tr>
<th>°C</th>
<th>0.000</th>
<th>0.050</th>
<th>0.101</th>
<th>0.151</th>
<th>0.202</th>
<th>0.253</th>
<th>0.303</th>
<th>0.354</th>
<th>0.405</th>
<th>0.456</th>
<th>0.507</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000</td>
<td>0.050</td>
<td>0.101</td>
<td>0.151</td>
<td>0.202</td>
<td>0.253</td>
<td>0.303</td>
<td>0.354</td>
<td>0.405</td>
<td>0.456</td>
<td>0.507</td>
</tr>
<tr>
<td>10</td>
<td>0.507</td>
<td>0.558</td>
<td>0.609</td>
<td>0.660</td>
<td>0.711</td>
<td>0.762</td>
<td>0.814</td>
<td>0.865</td>
<td>0.916</td>
<td>0.968</td>
<td>1.019</td>
</tr>
<tr>
<td>20</td>
<td>1.019</td>
<td>1.071</td>
<td>1.122</td>
<td>1.174</td>
<td>1.226</td>
<td>1.277</td>
<td>1.329</td>
<td>1.381</td>
<td>1.433</td>
<td>1.485</td>
<td>1.537</td>
</tr>
<tr>
<td>30</td>
<td>1.537</td>
<td>1.589</td>
<td>1.641</td>
<td>1.693</td>
<td>1.745</td>
<td>1.797</td>
<td>1.849</td>
<td>1.902</td>
<td>1.954</td>
<td>2.006</td>
<td>2.059</td>
</tr>
<tr>
<td>40</td>
<td>2.059</td>
<td>2.111</td>
<td>2.164</td>
<td>2.216</td>
<td>2.269</td>
<td>2.322</td>
<td>2.374</td>
<td>2.427</td>
<td>2.480</td>
<td>2.532</td>
<td>2.585</td>
</tr>
<tr>
<td>50</td>
<td>2.585</td>
<td>2.638</td>
<td>2.691</td>
<td>2.744</td>
<td>2.797</td>
<td>2.850</td>
<td>2.903</td>
<td>2.956</td>
<td>3.009</td>
<td>3.062</td>
<td>3.116</td>
</tr>
</tbody>
</table>

Thermoelectric Voltage in mV
Measurement Challenges Are Compounded By:

- Compressed Timelines
- Fixed Software and Hardware
- Conflicting Programming Approaches
- Inadequate Hardware Performance
- Disparate Driver APIs
- Varying Sensors and Connectivity
- Custom Signal Conditioning
- Advanced Visualization
- Changing Application Requirements
- Complex Analysis Algorithms
- Evolving Technology Trends
- Confusing Data Storage
- Differing Sampling Rates
Mixed-Measurement Applications Are Diverse

- Vibration
- Torque
- Displacement
- Pressure
- Temperature
- Force
- Strain
Example Application: Air Quality Measurements

• Potential Sensors Needed:
  • Context
    • GPS
      • Timestamp
      • Position
    • Attitude
    • Altitude
    • Range Finder
  • Environmental
    • Temperature
    • Oxygen
    • Carbon Dioxide
    • Ozone
    • Nitrogen
<table>
<thead>
<tr>
<th>Sensor</th>
<th>Interface</th>
<th>Conditioning?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>RS232</td>
<td>No</td>
</tr>
<tr>
<td>Attitude, Altitude</td>
<td>RS232</td>
<td>No</td>
</tr>
<tr>
<td>LiDAR</td>
<td>Ethernet</td>
<td>No</td>
</tr>
<tr>
<td>Temperature</td>
<td>Analog Voltage</td>
<td>Required</td>
</tr>
<tr>
<td>O₂, CO₂, O₃, NH₃</td>
<td>Analog Voltage</td>
<td>Required</td>
</tr>
</tbody>
</table>
## Software Provided With Sensors

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Software Provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td></td>
</tr>
<tr>
<td>Attitude, Altitude</td>
<td></td>
</tr>
<tr>
<td>LiDAR</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>O₂, CO₂, O₃, NH₃</td>
<td>&lt;No Software Provided&gt;</td>
</tr>
</tbody>
</table>
With a System Like This, How Do You Accommodate...

• ...changes in requirements?
• ...mixed measurements in a single system?
• ...varying connectivity?
• ...signal conditioning for sensors?
• ...adding or replacing measurements or sensors?
• ...incorporating timing, triggering, or synchronization?
• ...leveraging emerging technology trends?
• ...multiple disparate software environments and APIs?
National Instruments’ Strategy: Graphical System Design

Your Investment in a Platform-Based Approach to Measurements Scales Across…
Top Benefits of an Integrated Measurement Platform

1. Accelerated Productivity
2. Proven Performance and Accuracy
3. Scalability, Adaptability, and Flexibility
Architecture of an Integrated Measurement System

Today, we’ll learn about three key differentiating components of a National Instruments data acquisition system:

- Sensor
- Measurement Device
- Software

- Signal Conditioning
- Analog-to-Digital Converter

- Driver Software
- Application Software
Architecture of an Integrated Measurement System

LabVIEW is system design software that provides engineers and scientists with the tools needed to create and deploy measurement and control systems through unprecedented hardware integration.
Exercise 1a: Measurement & Automation Explorer

- Channels and Tasks Creation
- Managing Local and Remote System
- Interactive Test Panel
- Diagrams and connections
Introduction to LabVIEW

System Design Software for Any Measurement Application
Unrivaled Hardware Integration in a Single Environment

- NI hardware
  - 200+ data acquisition devices
  - 450+ modular instruments
  - Cameras
  - Motion control

- Third-party hardware
  - Instrument Driver Network
    - 10,000+ instrument drivers
    - 350+ instrument vendors
    - 100+ instrument types
  - Communicate over any bus
The Foundation of LabVIEW: Virtual Instrumentation

Automation through software led to a realization about fixed-functionality instrumentation...

Redundancy: Power Supplies
Each separate instrument requires its own power supply to run measurement circuitry that captures the real-world signal.

Redundancy: Displays
Instrument vendors provide a limited-quality display per instrument, even though monitor technology is far more advanced.

Redundancy: Processors
Chip manufacturers rapidly enhance processors according to Moore’s law, but instruments have fixed processing power.

Redundancy: Memory
PCs can quickly capitalize on a performance boost from a memory upgrade from readily available RAM.

Redundancy: Storage
Each instrument duplicates onboard storage even though PC hard drives are plentiful and cost-effective.
The Foundation of LabVIEW: Virtual Instrumentation

By leveraging COTS PC components, the software becomes the instrument.

LabVIEW unlocks the power of instrument and data acquisition hardware by capitalizing on the PC industry and abstracting redundant circuitry.
Therefore, LabVIEW Building Blocks Are Called Virtual Instruments (*.VI)

- **LabVIEW Front Panel**: The user interface of a VI
- **LabVIEW Block Diagram**: The source code of a VI
- **Icon / Connector Pane**: Maps inputs and outputs

*Note: A *.vi file encapsulates all three elements*
Creating a LabVIEW Front Panel

Controls Palette (Right-Click)
Access a hierarchical palette of all front panel elements.

Quick Drop (Ctrl + Space)
Search by object name.
Front Panel Objects

Decorations
Decorative elements and imagery
- Text
- Arrows
- Callouts
- Lines
- Images
- …and more

Customizable Indicators
Used to convey outputs to a user
- Graphs and Charts
- Progress Bars
- Gauges and Meters
- LEDs
- Numerics
- Strings and Paths
- …and more

Customizable Controls
Used to receive input from a user
- Knobs and Dials
- Sliders
- Buttons
- Numerics
- Strings and Paths
- …and more
All Front Panel Elements Have Block Diagram Terminals

Block diagram terminals provide access to front panel values
What Is Data Flow?

- Each block diagram node executes only when it receives all inputs
- Each node produces output data after execution
- Data flows along a path defined by wires
- The movement of data determines execution order

Formula: \( \text{Result} = \frac{(A+B\times C)}{(D-E)} \)
What Is Data Flow?

• Each block diagram node executes only when it receives all inputs
• Each node produces output data after execution
• Data flows along a path defined by wires
• The movement of data determines execution order

The [Multiply] and [Subtract] operations can execute at the same time since they don’t have any data dependencies.
Dataflow Languages Naturally Express Parallelism

The LabVIEW compiler will automatically multithread code expressed in parallel.

**Task Parallelism**

Unfiltered Signal → Filter → Filtered Signal

Time Domain Signal → FFT → Frequency Domain Signal

**Data Parallelism**

Audio Signal (Left & Right) → Left Band Pass Filter → Filtered Signal (Left & Right)

Audio Signal (Left & Right) → Right Band Pass Filter → Filtered Signal (Left & Right)
Creating a LabVIEW Block Diagram

Functions Palette (Right-Click)
Access a hierarchical palette of all block diagram functions.

Quick Drop (Ctrl + Space)
Search by object name.
Execution Control Structures: Loops

Count Terminal
The code contained within this For Loop will execute N times.

Loop Iteration Terminals
This provides the current loop iteration count, which ranges from 0 to N-1.

Conditional Terminal
The code within this While Loop will run until a True value is evaluated.

For Loop

While Loop
Text Loops and Their LabVIEW Equivalents

```c
int x = 0;
String y;
while (x < 5) {
    y = functionCall(x);
    printf(y);
    x++;
}
```

```c
for (i = 0; i < 10; i++)
{
    /* loop body */
}
```

```c
for (i = 0; i < 10; i++)
{
    if(check(i)) break;
}
```

```c
for (i = 0; i < 10; i++)
{
    /* loop body */
}
```

```c
while (x < 5) {
    y = functionCall(x);
    printf(y);
    x++;
}
```
Event and Case Structures

**Event Selector Label**
This indicates which subdiagram is visible and details the event that the code within the diagram handles.

**Case Selector Label**
This indicates which subdiagram is visible.

**Selector Terminal**
The value wired to this terminal determines which of the subdiagrams, or cases, will execute.

---

**Event Structure**

**Case Structure**
Text Events, Cases, and Their LabVIEW Equivalents

Button B = new Button();
B.Click += new RoutedEventHandler(OnBClick);

void OnBClick(object Source)
{
    Text1.Text = “Button B was Clicked!”;
}

if condition1 then
    -- statements;
elseif condition2 then
    -- more statements
elseif condition3 then
    -- more statements;
else
    -- other statements;
end if

switch (n) {
    case 5:
        printf(“Small number.”);
        break;
    case 100:
        printf(“Large number.”);
        break;
    default:
        printf(“Outside range”);
        break;
}
Exploring a LabVIEW Block Diagram

**Event Structure**
Executes different subdiagrams based on events and interrupts

**While Loops**
Iterate continuously until a true value is passed to the stop terminal

**Case Structure**
Executes different subdiagrams based on the value of its selector terminal
LabVIEW Functions Are as Complex as You Need

Express VIs
- Quick and Easy
- Configuration-Based
- Limited

Regular VIs
- Hides Unnecessary Details
- Retains Power and Flexibility

Low-Level VIs
- Powerful, Flexible
- Difficult, Time-Consuming
Understanding SubVI (Function) Behavior

• Code will only compile if required inputs are wired
• Required inputs are **Bold**
• If an optional input is not supplied, a default value will be used for execution

*Tip: Access the Context Help using **Ctrl+H***
Understanding Application Hierarchy

Double-clicking a nonprimitive SubVI opens the function

Every VI can be a SubVI
Remember that each SubVI has its own front panel and block diagram.
Never Start a LabVIEW Project From Scratch

Abundant sample projects and templates provide a scalable starting point

- Recommended starting points for common LabVIEW applications
- Clearly indicate where to add or change functionality
- Shows best practices for code design, documentation, and organization
- Add custom templates and sample projects
Exploring a LabVIEW Block Diagram

The color of the wire indicates its data type, which is strictly enforced at edit-time.
# The Color, Style, and Thickness of Common Wires

<table>
<thead>
<tr>
<th>Wire Type</th>
<th>Scalar</th>
<th>1D Array</th>
<th>2D Array</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating Point</td>
<td>Orange</td>
<td>Orange</td>
<td>Orange</td>
<td>Orange</td>
</tr>
<tr>
<td>Integer</td>
<td>Blue</td>
<td>Blue</td>
<td>Blue</td>
<td>Blue</td>
</tr>
<tr>
<td>Boolean</td>
<td>Green</td>
<td>Green</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>String</td>
<td>Pink</td>
<td>Pink</td>
<td>Pink</td>
<td>Pink</td>
</tr>
<tr>
<td>Error</td>
<td>Yellow</td>
<td>Yellow</td>
<td>Yellow</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

A “broken wire” represents a data type conflict that LabVIEW cannot automatically resolve. Fix it, or your code won’t run!
Exercise 1: Create a simple LabVIEW VI

What we learn…

- Working in LabVIEW Environment

Create a program to:

- Simulate a signal
- Create an advanced Front Panel
The Fundamentals of Data Acquisition (DAQ)

The Basics of Making PC-Based Measurements
Signals Come in Two Forms: Digital and Analog
Digital Signals

- Digital signals have two states: high and low
- Digital lines on a DAQ device accept and generate transistor-transistor logic (TTL) compatible signals

+5.0 V
High State
+2.0 V
Indeterminate
+0.8 V
Low State
0 V

State
Rate
Digital Terminology

Bit
The smallest unit of data. Each bit is either a 1 or a 0.

Byte
A binary number consisting of eight related bits of data.

Line
One individual signal in a port. Bit refers to the data transferred. Line refers to the hardware.

Port
A collection of digital lines (usually four or eight).
Analog Signals

Analog signals are continuous signals that can be any value with respect to time.
The Three R’s of Data Acquisition: Resolution

Resolution

- 6-Bit Resolution
- 3-Bit Resolution

Range

Rate

Original Signal

ni.com
The Three R’s of Data Acquisition: Range

Resolution

Range

Rate

Range of 
-10Ω to 10Ω

Original Signal

3-Bit Resolution

ni.com
The Three R’s of Data Acquisition: Rate

Resolution

Range

Rate

Original Waveform (10 Hz)

Sampling Rate = 120 Hz
Architecture of an Integrated Measurement System

NI CompactDAQ hardware combines a 1-, 4-, or 8-slot chassis with over 50 measurement-specific NI C Series I/O modules and can operate stand-alone with a built-in controller or connect to a host computer over USB, Ethernet, or 802.11 Wi-Fi.
NI Is the Global Leader in Data Acquisition

With more than 20 years of DAQ hardware history and millions of channels sold
NI Data Acquisition Hardware Families

**System**
- **PXI**
  - Optimized for high channel counts and tight synchronization
- **NI CompactDAQ**
  - Customize with a variety of chassis and module types

**Single Device**
- **Desktop DAQ**
  - Install in a desktop PC slot for maximum data throughput
- **Portable DAQ**
  - Easily connect to any laptop or desktop with simple setup
NI CompactDAQ Is an Integrated, Modular Solution

Sensors/Signals

- Thermocouple
- Accelerometer
- Strain Gage
- Solar Cell

(etc)

C Series Modules
The NI CompactDAQ Family

A Custom System for Your Application
Mix and match from the entire family of measurement-specific, auto-detected, hot-swappable C Series modules.

A Module for Any Measurement
Over 50 measurement-specific modules integrate everything you need for a range of signal types, channel counts, and rates.

Same Code, Any Bus
Whether you’ve chosen to use USB, Ethernet, or Wi-Fi, identical code will run across each bus making scalability simple.

Choose the Right Form Factor for You
Available 1-, 4-, and 8-slot chassis accommodate up to 256 channels per chassis in tethered or stand-alone form.
Family Highlight: Stand-Alone NI CompactDAQ

Embedded Measurements and Logging

- >50 I/O modules
- Up to 24-bit, Up to 1 MS/s
- Dual-core processor
- 32 GB nonvolatile storage

- 0 to 55 °C Operating Temp
- 5g shock, 30g vibration
- Windows or Real-Time OS
- LabVIEW and NI-DAQmx
C Series I/O Modules

- Over 100 Modules
  - Analog Input
  - Analog Output
  - Digital I/O
  - Relay Output
  - Counter, Pulse Generation
  - Communication
    - CAN
    - LIN
    - PROFIBUS
  - Motion Control
  - Wireless
  - Engine Control
- Signal Conditioning
- Rugged Mechanicals
- Signal Conditioning/Filtering
- Isolation Barrier
NI-DAQmx is free driver software that can be used in conjunction with several different programming languages to control thousands of different data acquisition devices with a consistent API.
Bridging the Hardware and Software Gap with NI-DAQmx

NI-DAQmx is a single, free hardware driver that supports various development languages and hundreds of NI data acquisition hardware platforms.

The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.
NI-DAQmx API: Configuration-Based DAQ Assistant

- Enables quick, configuration-based measurements
- Usable across multiple channels, multiple devices
- Maximum ease of use with some sacrificed flexibility
- Supported across multiple programming languages
- Automatically generates lower-level code
Exercise 2: Introduction to Data Acquisition with LabVIEW

What we learn:

- Perform measurements with NI CompactDAQ and NI LabVIEW
- Generate Digital Alarm
- Save and present Data
Exercise 3: Multiple Data Acquisition System: Temperature or Strain?
Coffee Break
Enjoy Coffee and Networking With Industry Peers
Exercise 5: Vibration Measurement, File I/O, Mathscript Node

1. \( P = A \cdot \exp(j \cdot A) \);
2. \( p = \text{IFFT}(P) \);
3. \( \% \text{plot(real}(p) \));
4. \([b, a] = \text{ellip}(4, 1, 40, (1/1000)/2) \);
5. \( y = \text{filter}(b, a, A) \);
Optional: DAQ Assistant Under The Hood: NIDAQmx driver