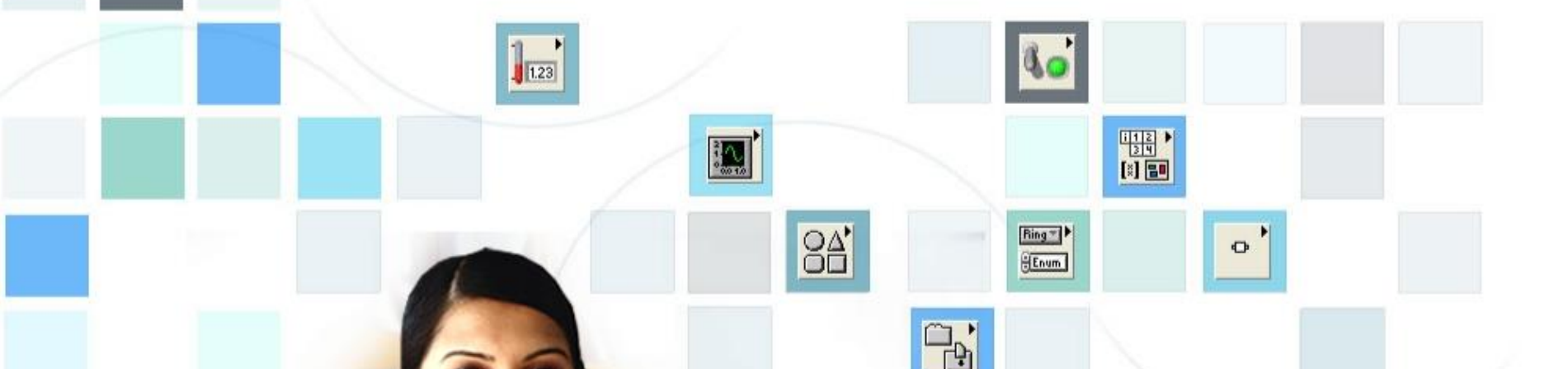




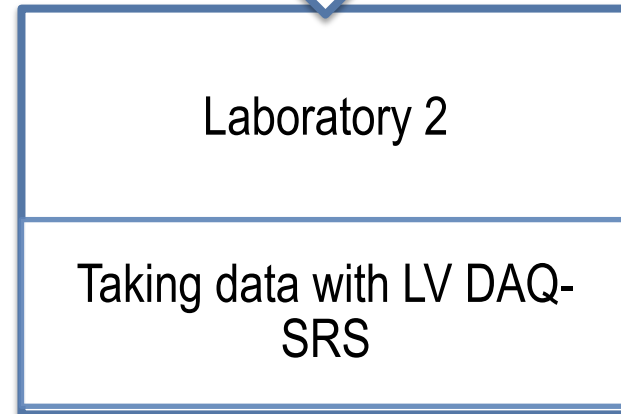
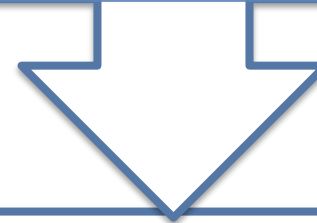
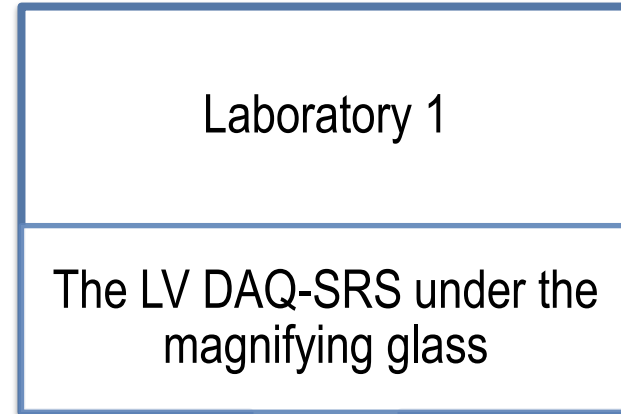
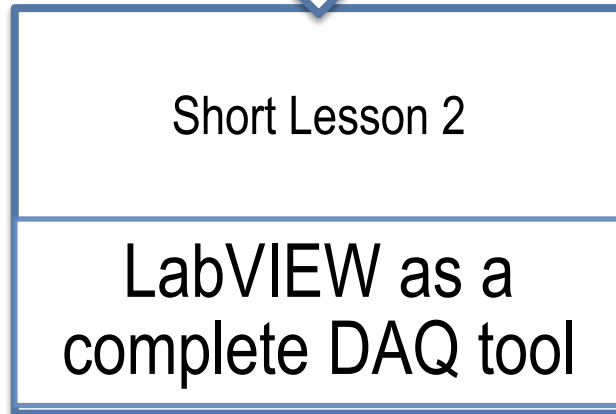
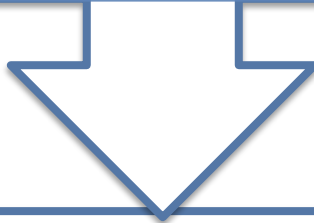
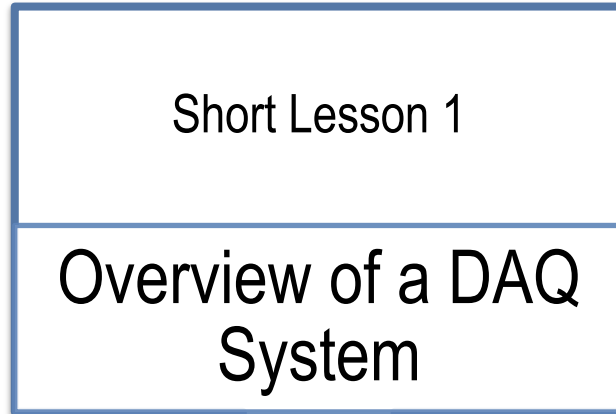
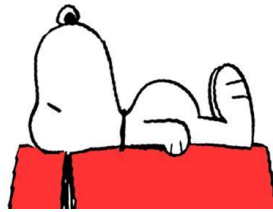
LabVIEW-Based SRS Data Acquisition System



Riccardo de Asmundis
INFN and Università «Federico II»
Napoli, Italy

Certified LabVIEW Developer &
Certified Professional Instructor
National Instruments
Austin (TX)

Lecture Map



Lesson 1

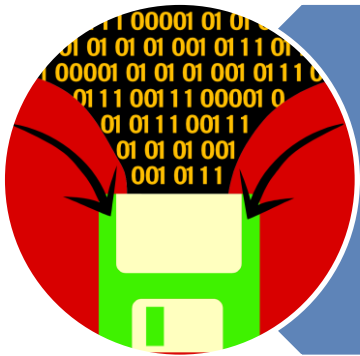
Overview of a Data Acquisition System

TOPICS

- A. DAQ System Overview
- B. Sensors
- C. Signals
- D. DAQ Hardware
- E. Signal Conditioning
- F. DAQ Software



A. DAQ System Overview



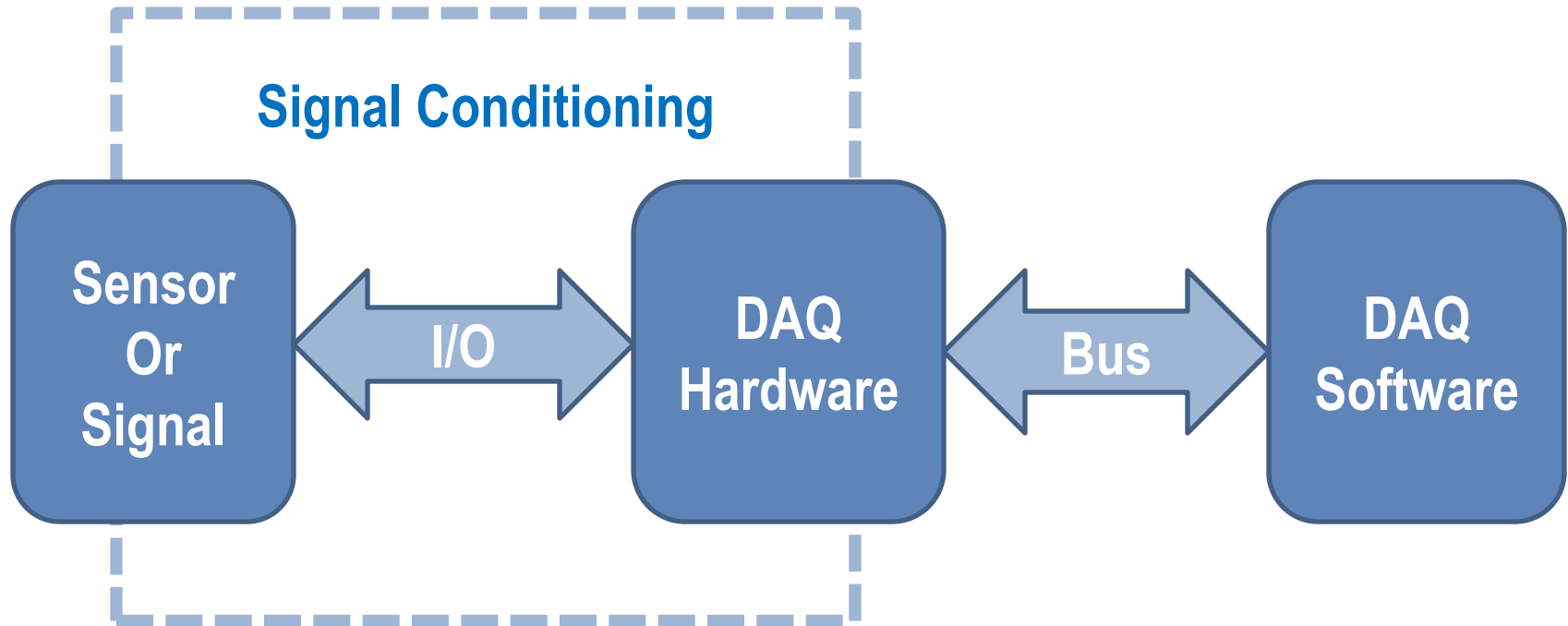
Data Acquisition (DAQ)—the automatic collection of data from sensors, instruments, and devices in a factory, laboratory, or in the field.

Purpose

To measure an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound

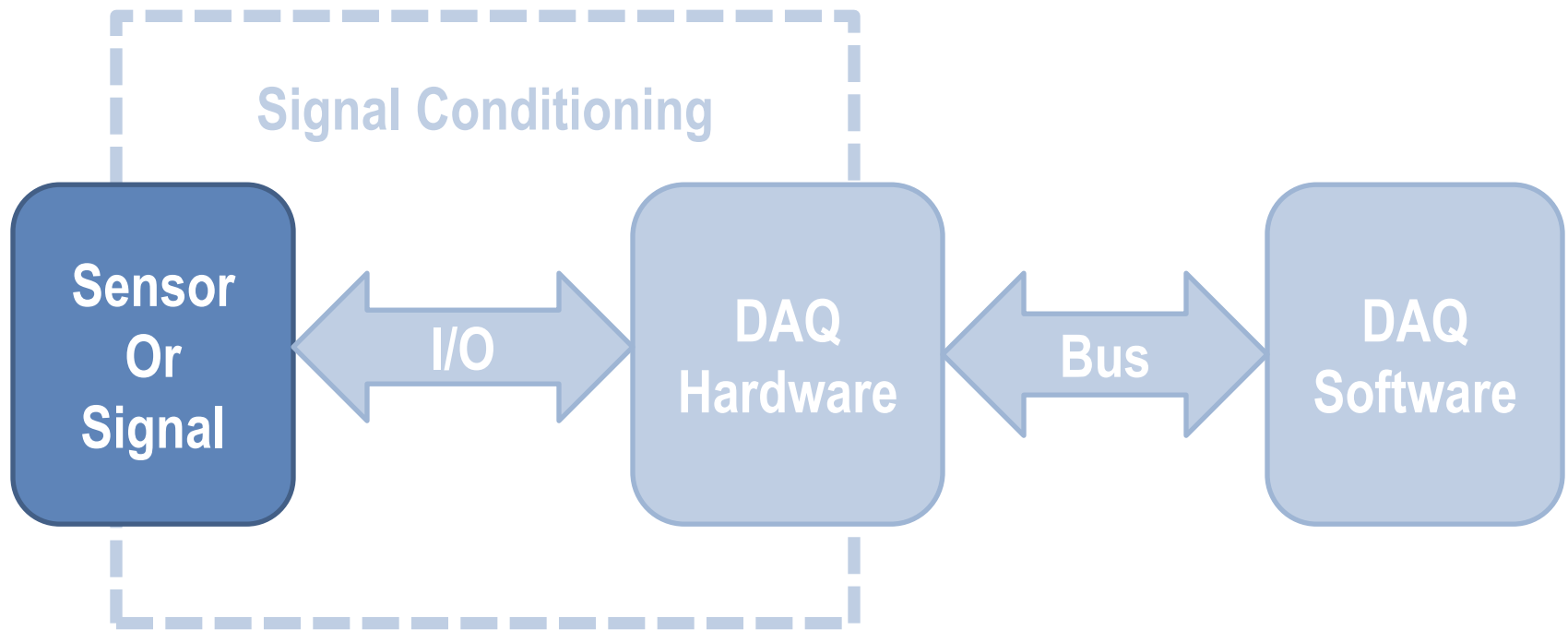


DAQ System Overview

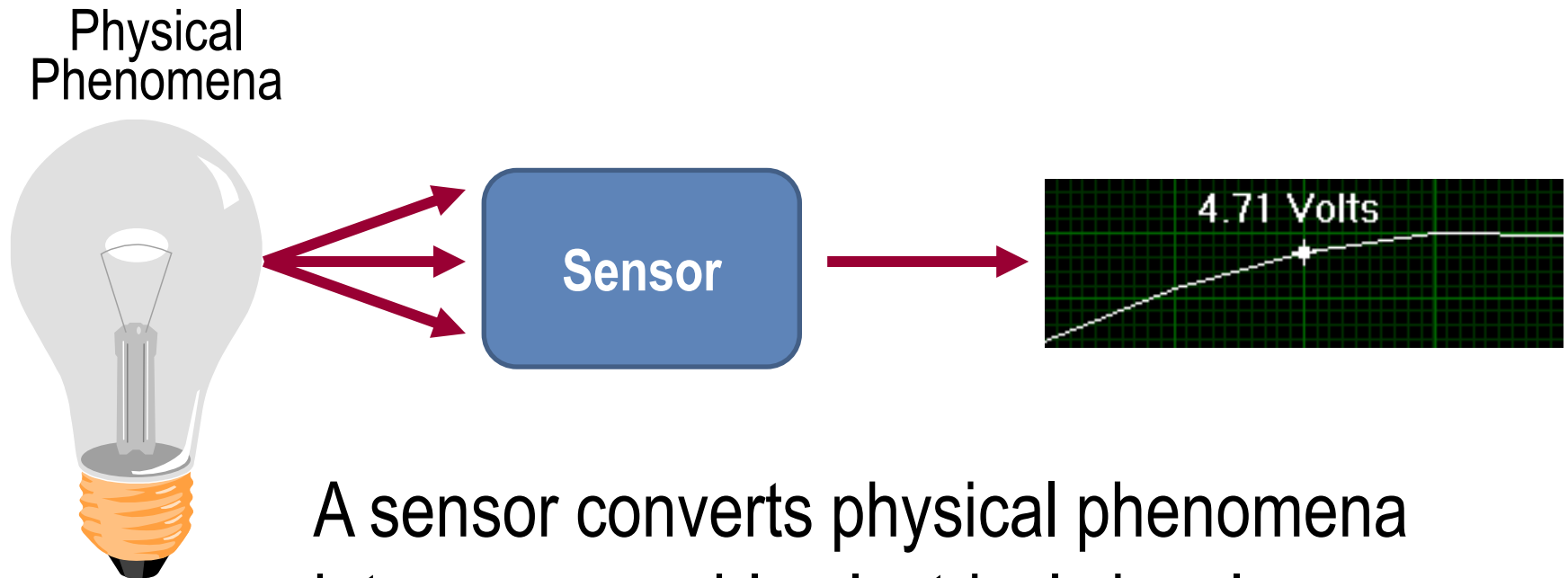


B. Sensor Overview

- What is a sensor?
- Types of sensors



What is a Sensor?



A sensor converts physical phenomena into measureable electrical signals



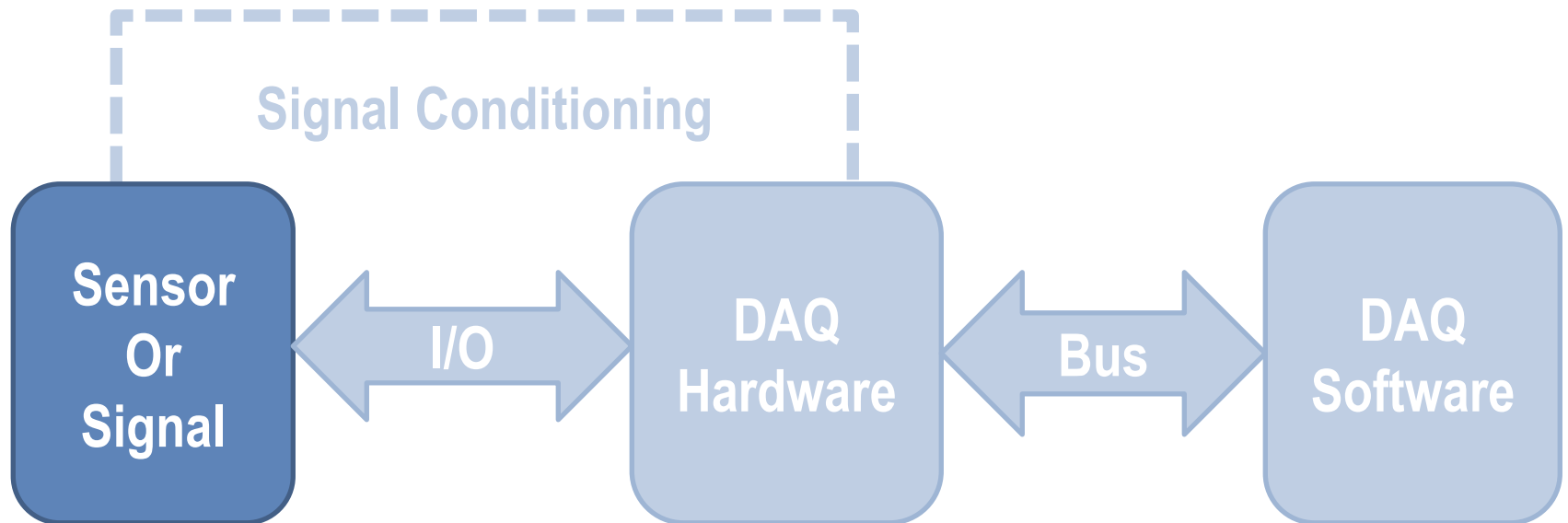
Types of Sensors

Phenomena	Sensors
Temperature	Thermocouples, Resistive Temperature Devices (RTDs), Thermistors
Strain and Pressure	Strain gages, Piezoelectric transducers
Sound	Microphone
Vibration	Accelerometer
Position and Displacement	Potentiometers, Linear voltage differential transformer, Optical encoder
Fluid	Head meters, Rotational flowmeters
pH	pH electrodes
Light	Vacuum tube, Photo sensors
Particles and Radiation detection in general	Ionization-proportional-Geiger chambers, Scintillators, Silicon detectors,...



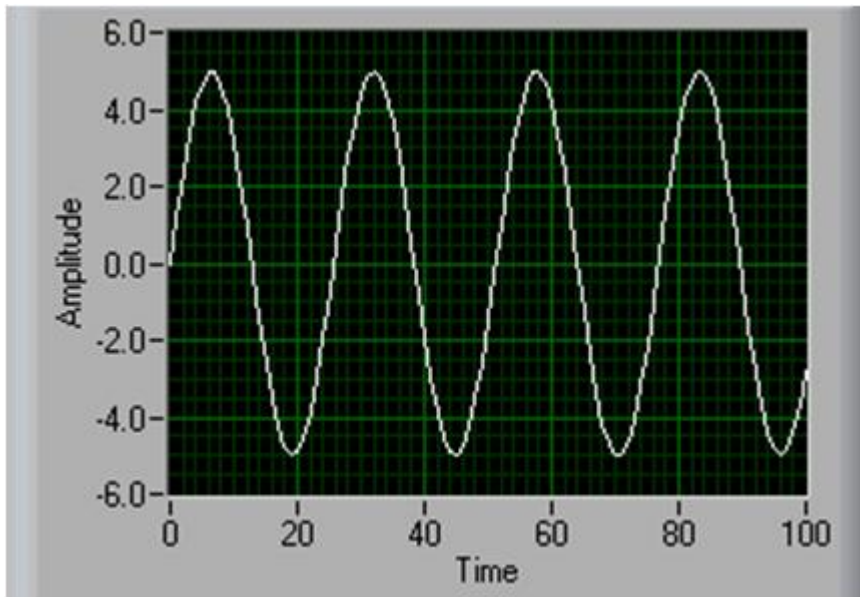
C. Signal Overview

- Signal classification
- Signal information

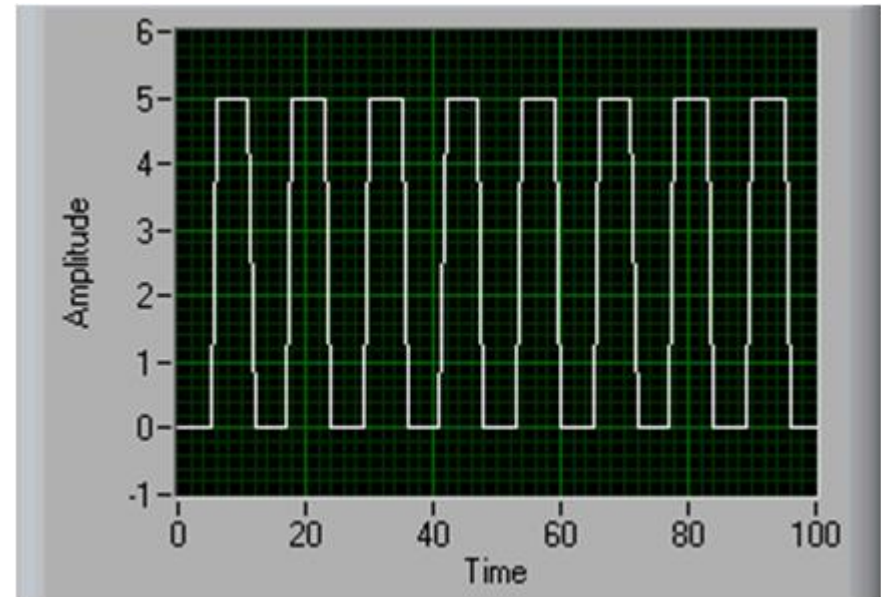


Signal Classification

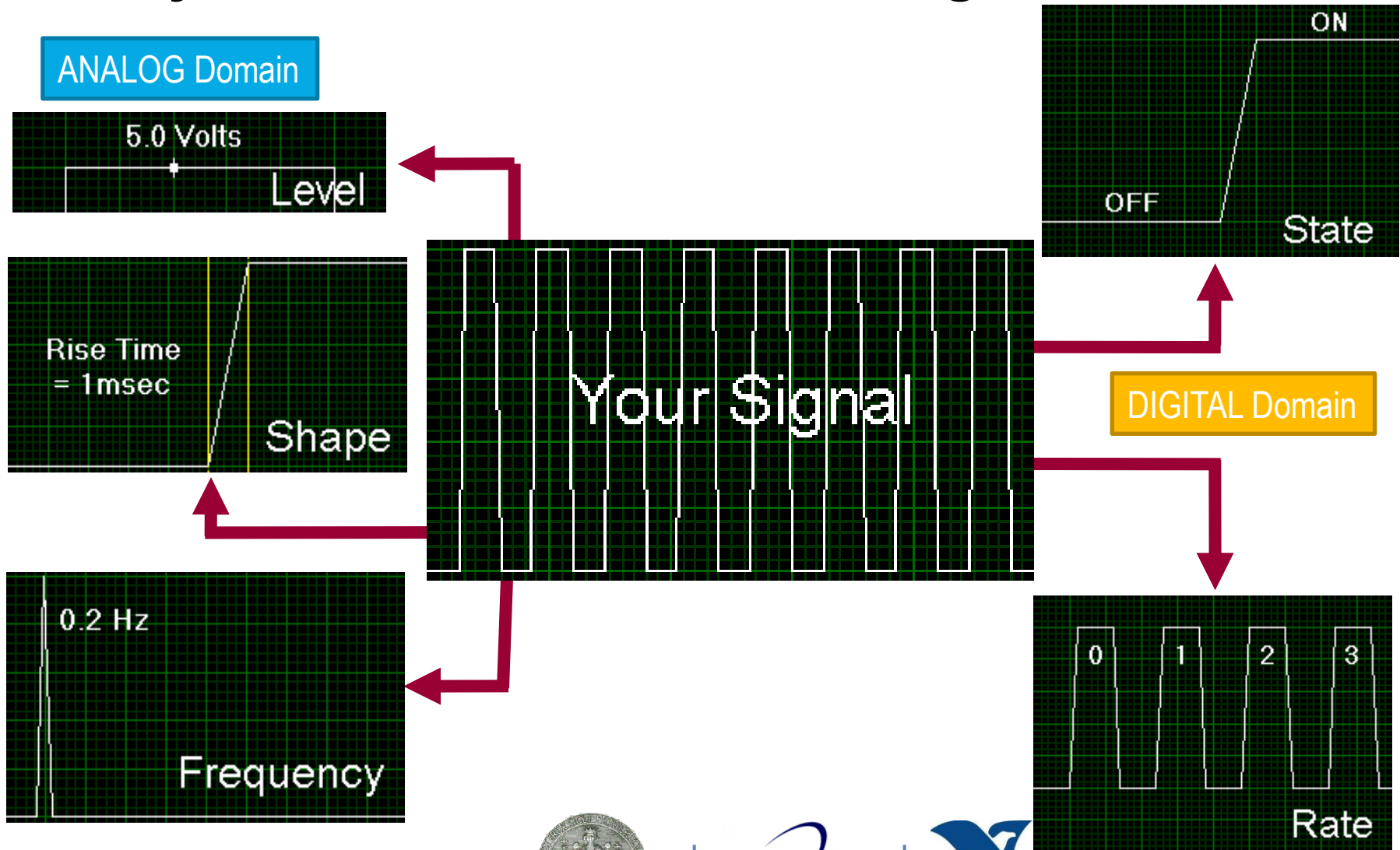
Analog



Digital



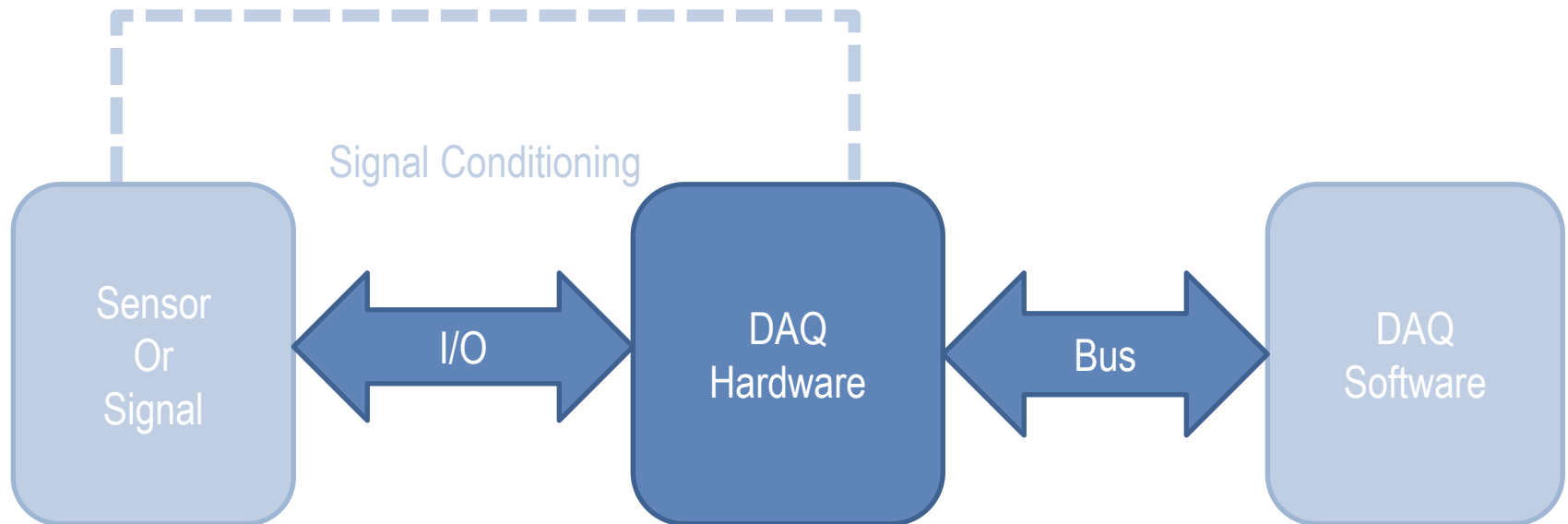
5 Ways to Measure the Same Signal



INSTRUMENTS™
Certified Professional Instructor

D. DAQ Hardware Overview

- Purpose of DAQ hardware
 - Transfer data between your sensor/signal and your software



Typical General Purpose DAQ Device Architecture

Features

- Analog Input
- Analog Output
- Digital I/O
- Counter

DAQ Circuitry

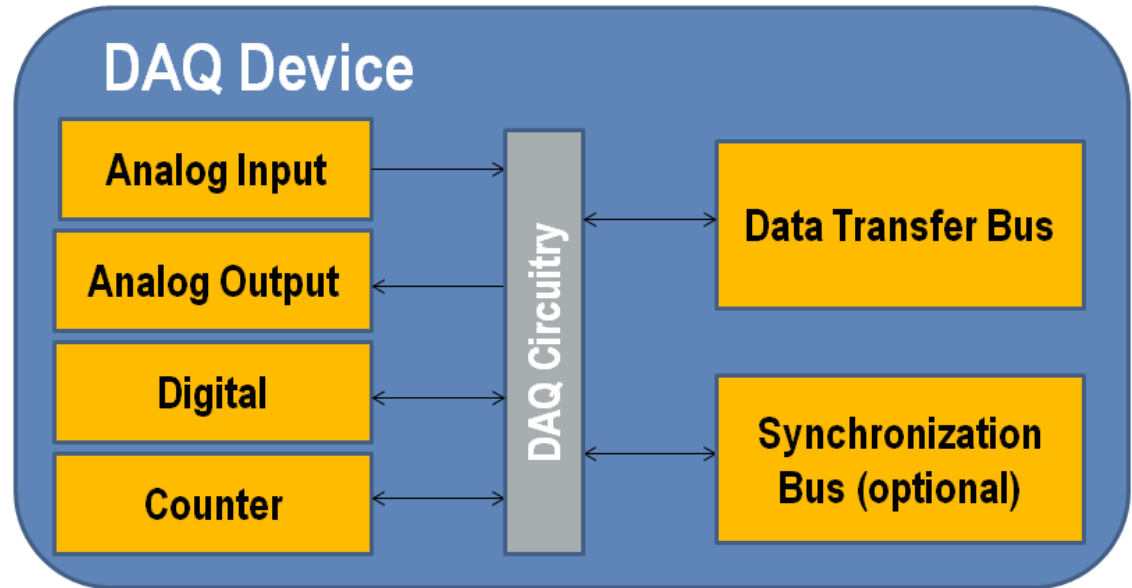
- Clock&Timing, FIFO,...

Data Transfer Bus

- USB, PCI, PCI Express, PXI, PXI Express

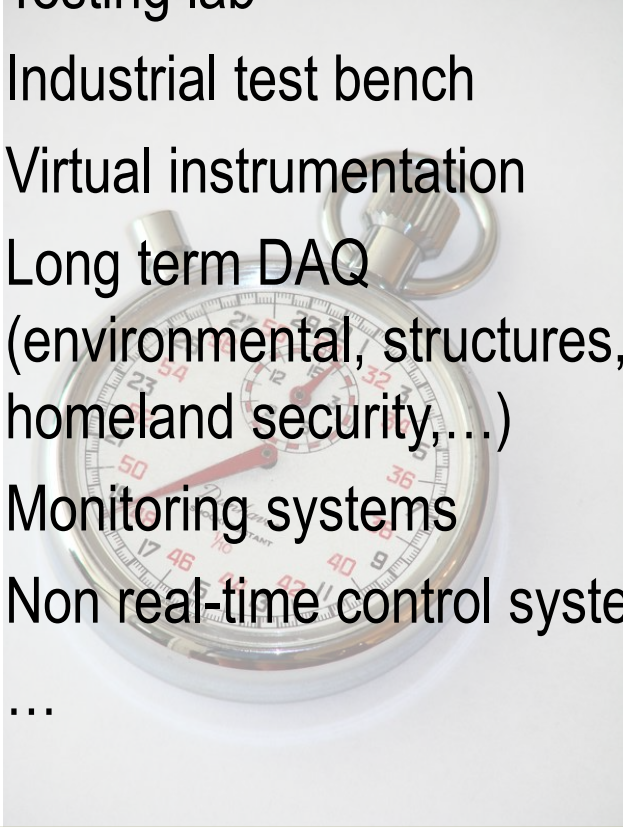
Synchronization Bus

- Used to synchronize multiple DAQ devices
- Allows sharing of timing and trigger signals between devices



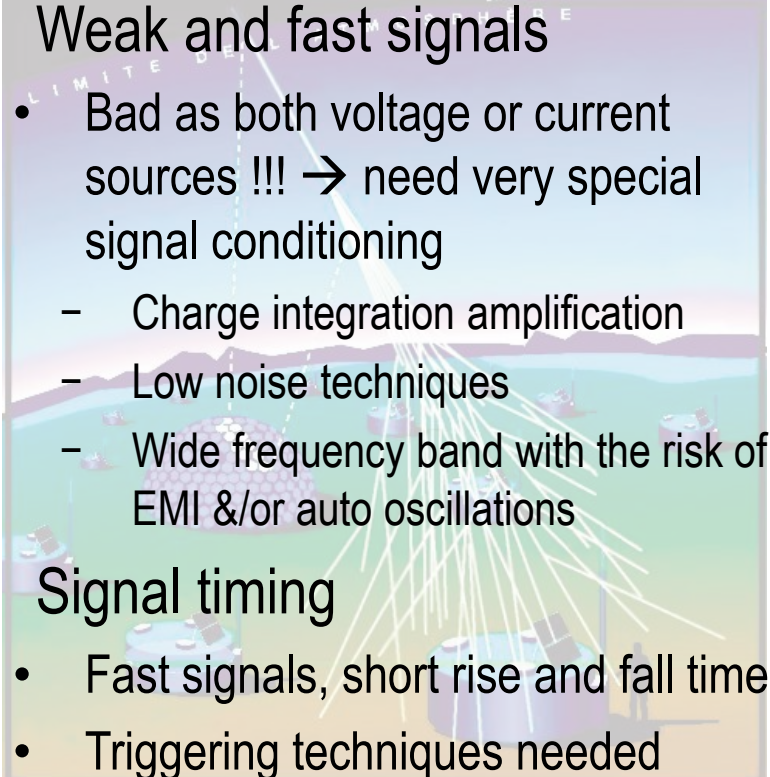
General DAQ Devices vs Specialised ones

General DAQ Devices are suitable for DAQ in «simple» situations

- Testing lab
 - Industrial test bench
 - Virtual instrumentation
 - Long term DAQ (environmental, structures, homeland security,...)
 - Monitoring systems
 - Non real-time control systems
 - ...
- 

Easy or standard signal conditioning needed

Physics presents different challenges

- Weak and fast signals
 - Bad as both voltage or current sources !!! → need very special signal conditioning
 - Charge integration amplification
 - Low noise techniques
 - Wide frequency band with the risk of EMI &/or auto oscillations
 - Signal timing
 - Fast signals, short rise and fall time
 - Triggering techniques needed
- 

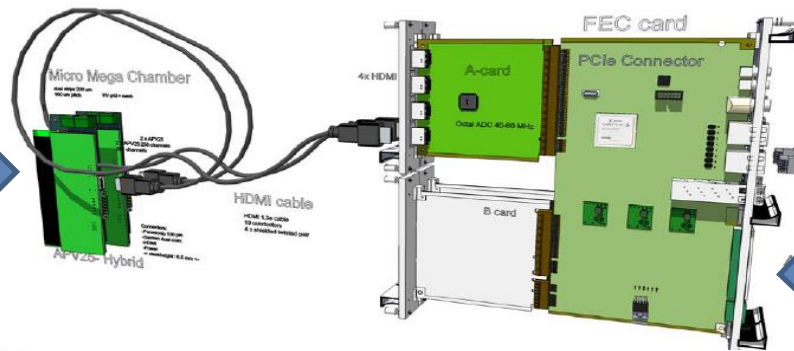
NON-standard signal conditioning!

- Custom front-end electronics
- Very specialized custom or industry-made DAQ electronics

The LabVIEW DAQ-SRS

Chambers → ON Board Electronics (signal conditioning) →
DAQ System → Bus connection → PC running LV (≥ 2012)

μ MGas,
GEM,...
chambers



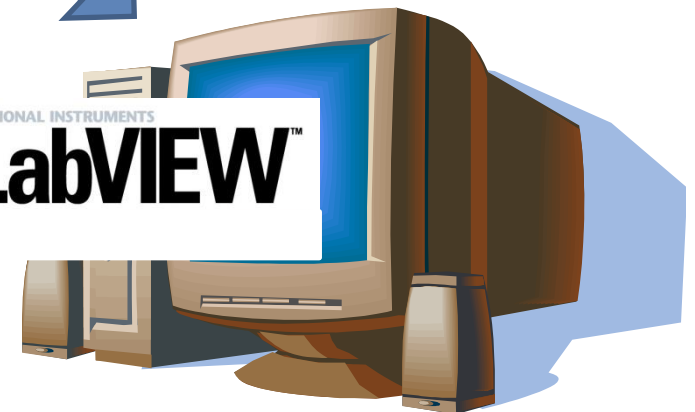
UDP connection



“RD51-srs”
LabVIEW Project

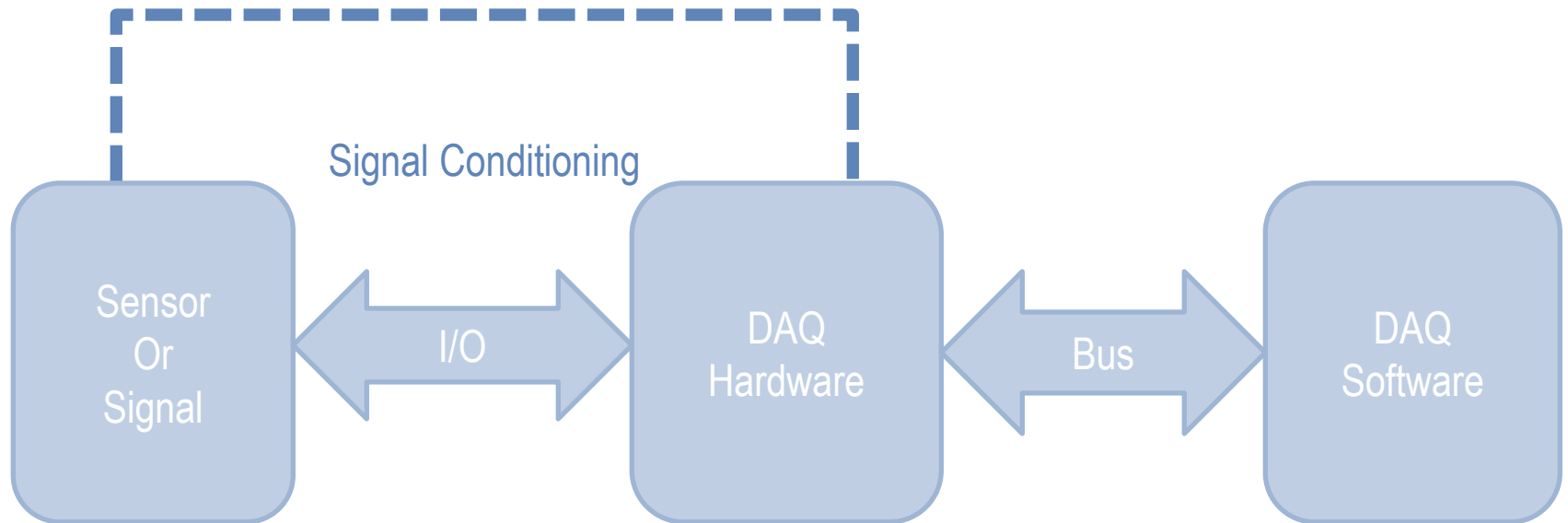


NATIONAL INSTRUMENTS
LabVIEW™



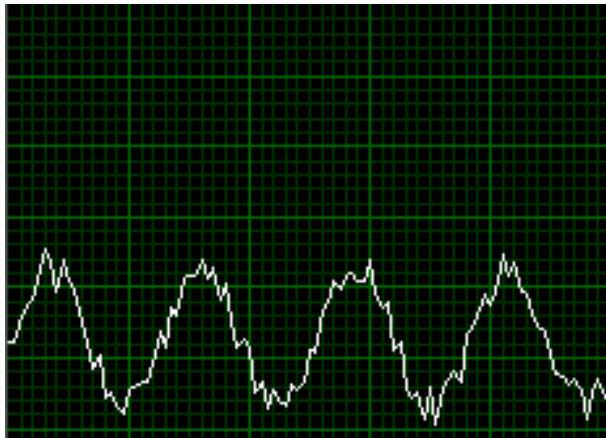
E. Signal Conditioning

- Purpose of signal conditioning
- Signal conditioning tasks and examples

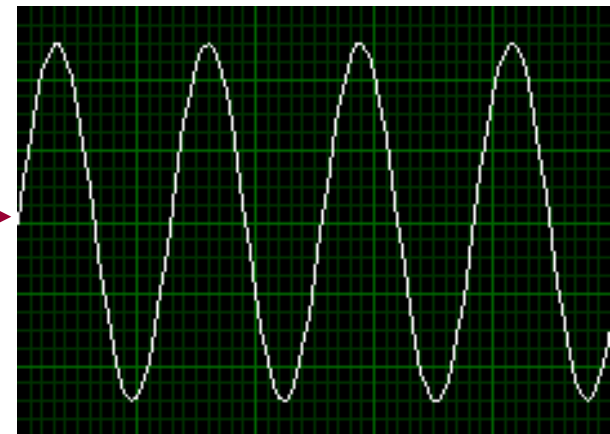
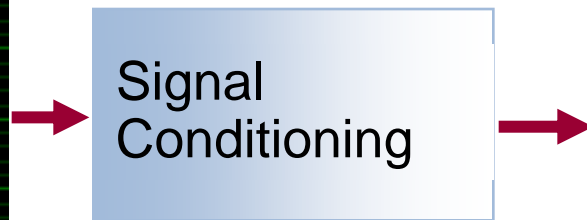


Purpose of Signal Conditioning

- Signal conditioning takes a signal that is difficult for your DAQ device to measure and makes it easier to measure
- Signal conditioning is not always required
 - Depends on the sensor or signal being measured



Noisy, Low-Level Signal



Filtered, Amplified Signal

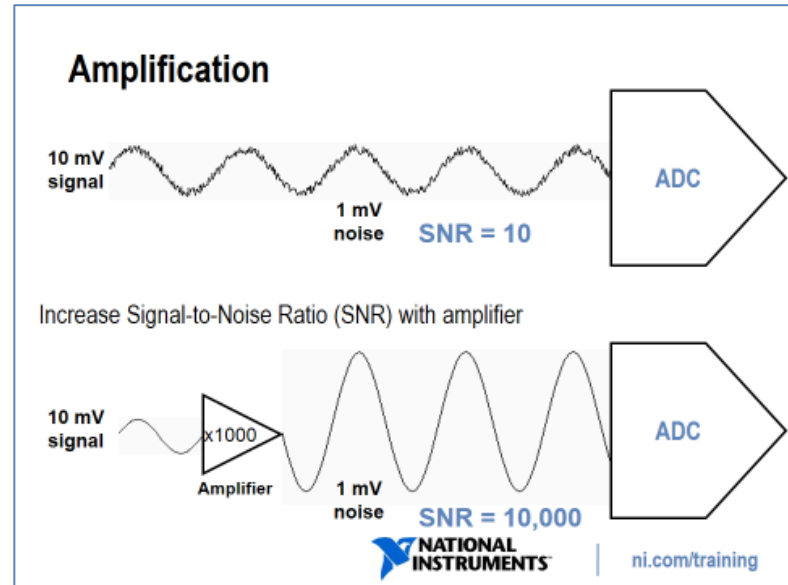


Signal Conditioning Tasks

They depend on the type of signal, but in principle

Analogue domain:

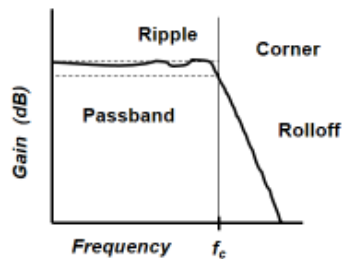
- Voltage measurement
 - Amplification
 - Attenuation
 - Isolation
 - Filtering



Filtering



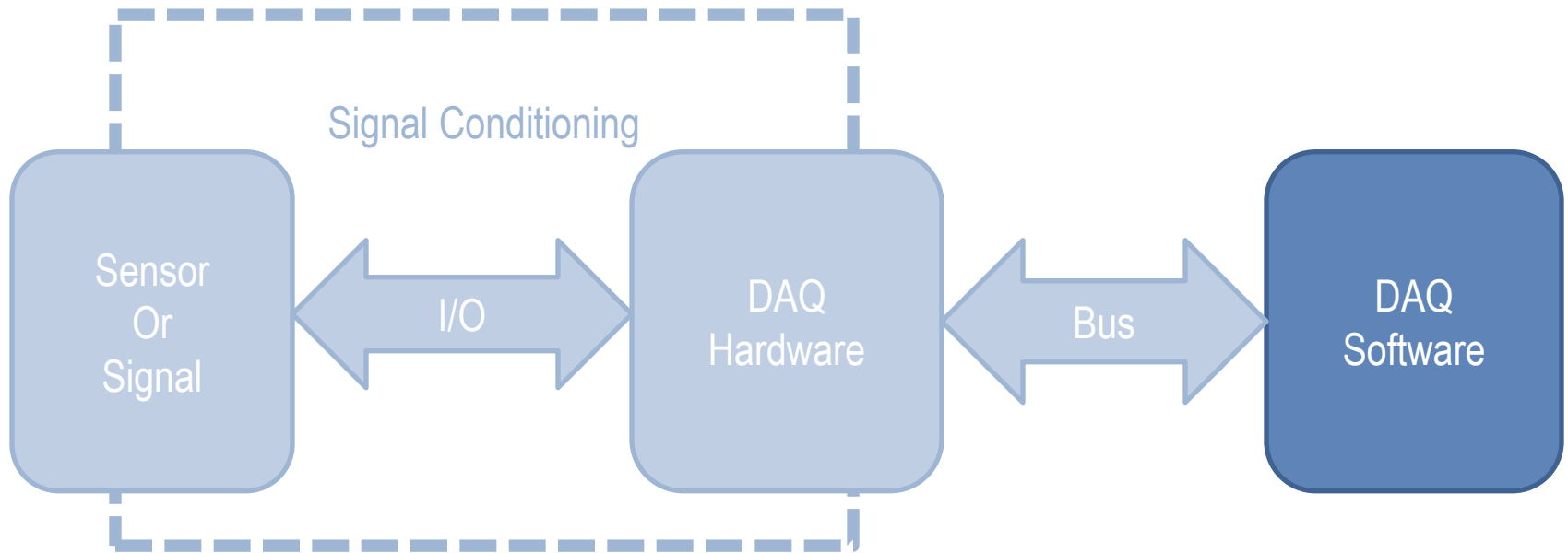
Bode Plot



- Passband – frequencies the filter lets pass
- Ripple - filter's effect on the signal's amplitude
- Corner – frequency where the filter begins blocking the signal
- Roll-off – how sharply the filter cuts off unwanted frequencies

F. DAQ Software Overview

- After acquiring data, you usually still need to do more
 - Signal processing, generate a report, interact with data, etc.



Lesson 2

LabVIEW as a complete DAQ tool

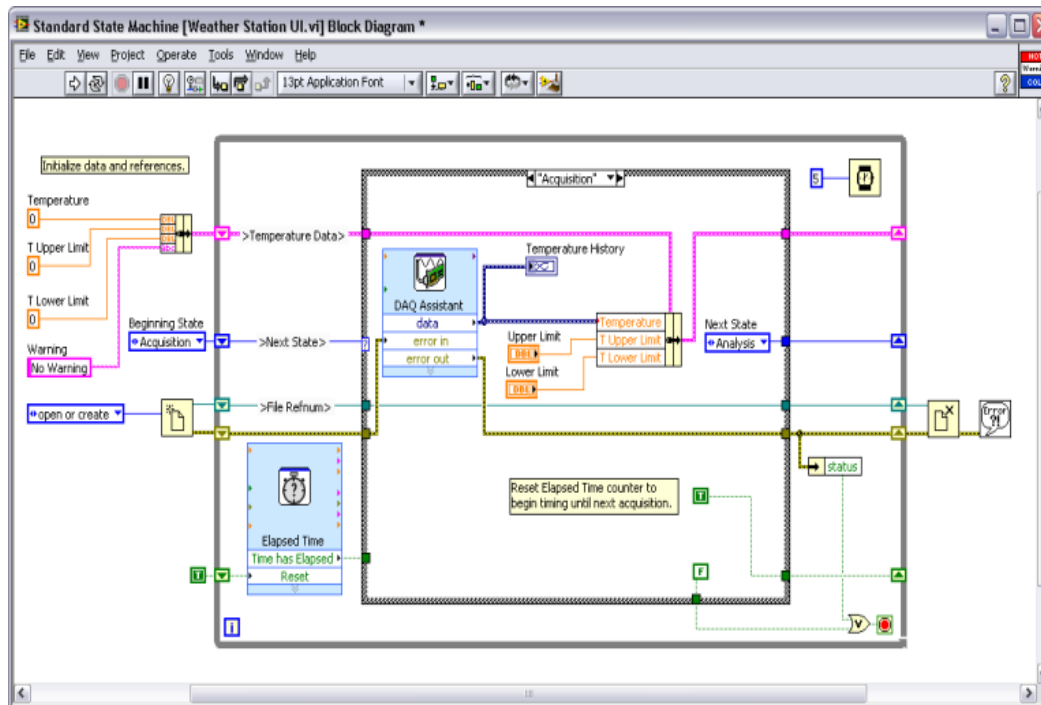
TOPICS

- A. What is LabVIEW
- B. Learning LabVIEW
- C. LV Design Patterns
- D. Event programming
- E. LabVIEW SRS DAQ program structure



A. What Is LabVIEW?

— A graphical programming environment used to develop sophisticated measurement, test, and control systems.



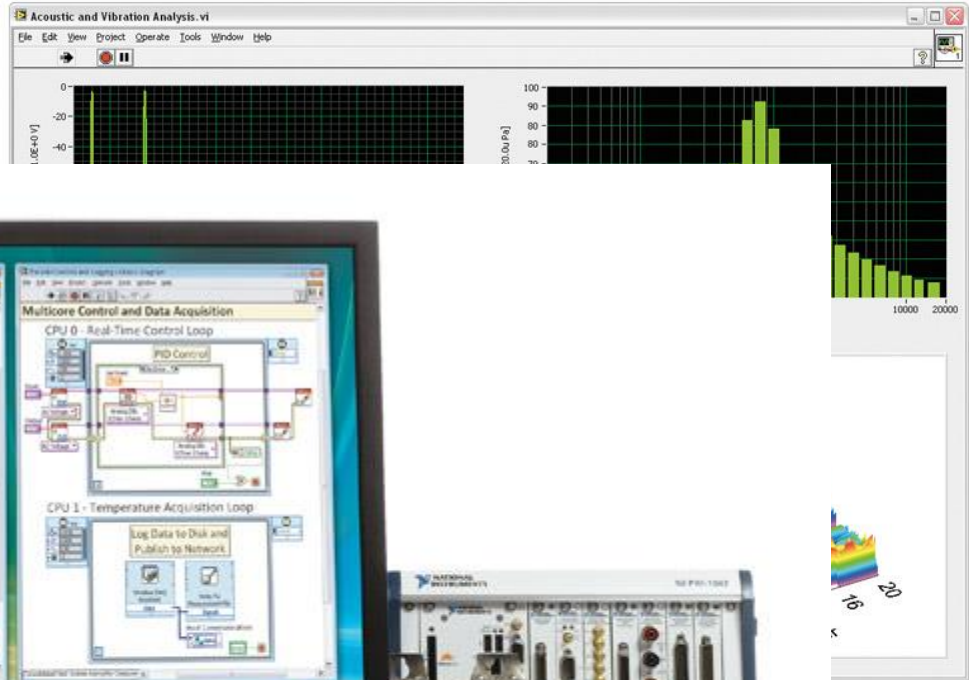
LabVIEW:

- Interfaces with wide variety of hardware
- Scales across different targets and OSs
- Provides built-in analysis libraries

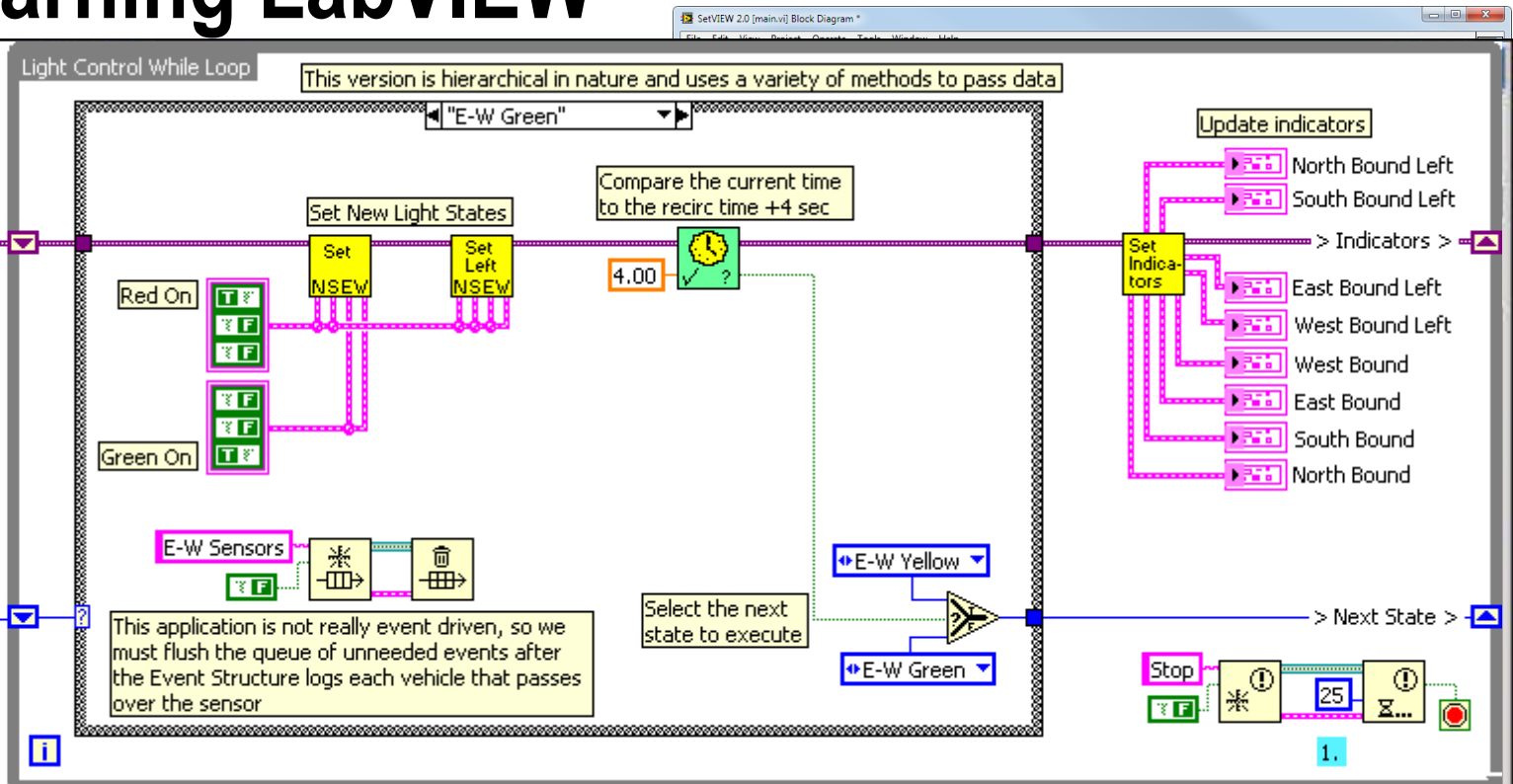


Some LabVIEW Features

- Fully Graphical Programming

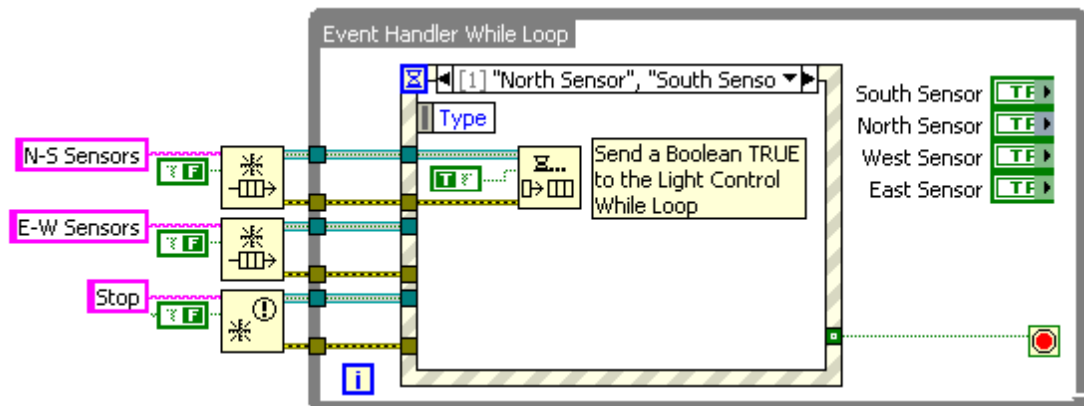


B. Learning LabVIEW

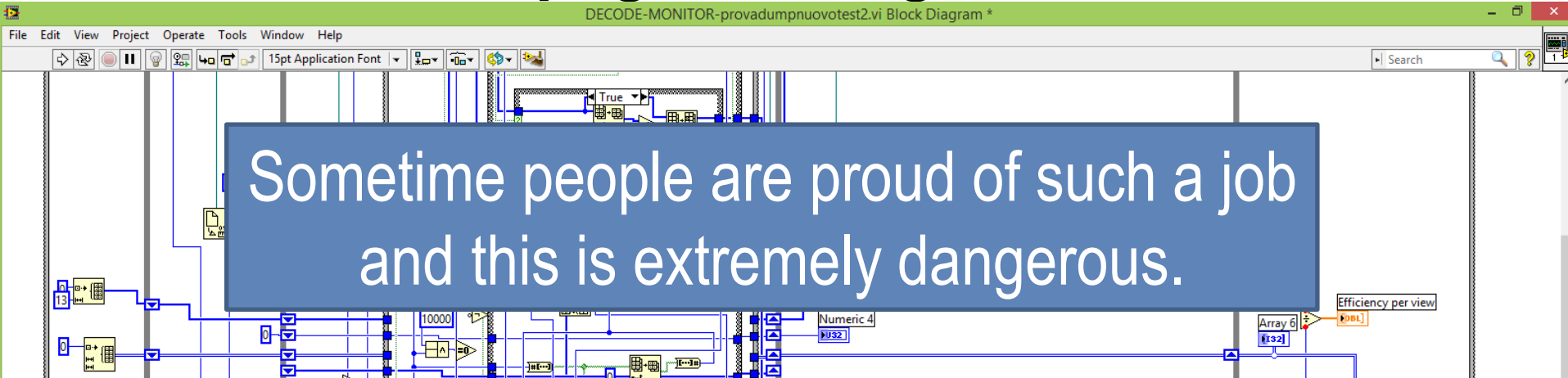


A version of the Stop Light VI that uses an Event Structure. Data is passed from the Event Structure via Notifiers and Queues.

- Note that the Stop Boolean Notifier in the Light Control While Loop performs double duty. It has a 25ms timeout, so no Wait function is necessary.
- The use of a Variant is notable in that only attributes are passed. This provides a degree of freedom in not requiring data to be predefined.



Some other “spaghetti diagram”



If you draw something like this in your past, please:

1. trash all away
2. forget everything
3. restart from beginning !



Available LabVIEW courses

New User

Experienced User

Advanced User

LabVIEW Core 1
LabVIEW Core 2

LabVIEW Core 3

Managing Software
Engineering in LabVIEW

LabVIEW Connectivity
Object-Oriented Design
and Programming in LabVIEW
LabVIEW Performance

Advanced Architectures in
LabVIEW

Certifications

Certified LV Associate
Developer Exam

Certified LabVIEW
Developer Exam

Certified LabVIEW
Architect Exam

Other Courses

LabVIEW Real-Time 1
LabVIEW Real-Time 2

LabVIEW Instrument Control
LabVIEW Modular Instruments

LabVIEW FPGA
DAQ & Signal Conditioning



LabVIEW Education

- Instructor Led Training
 - LabVIEW Performance
 - Object-Oriented Design and Programming in LabVIEW
 - Managing Software Engineering in LabVIEW
 - Advanced Architectures in LabVIEW
 - RealTime and FPGA
 - Data Acquisition
 - ...
- Self-Paced Online: Accessible 24 hours a day, prerecorded video training modules, interactive quizzes, and challenging exercises with solutions.
- Printed Course Materials: a variety of instructional packages and tools designed to educate you at your own pace

Take time to STUDY: time spent so, acts as a credit for the future !



C. Design Patterns

- Why use Design Patterns?
 - They have proven themselves useful for developing software.
 - You don't have to start a program from scratch.
 - They make it easier for others to read and modify your code.

Design Patterns – Code implementations and techniques that are solutions to specific problems in software design

Design patterns typically evolve through the efforts of many developers and are fine-tuned for simplicity, maintainability, and readability.

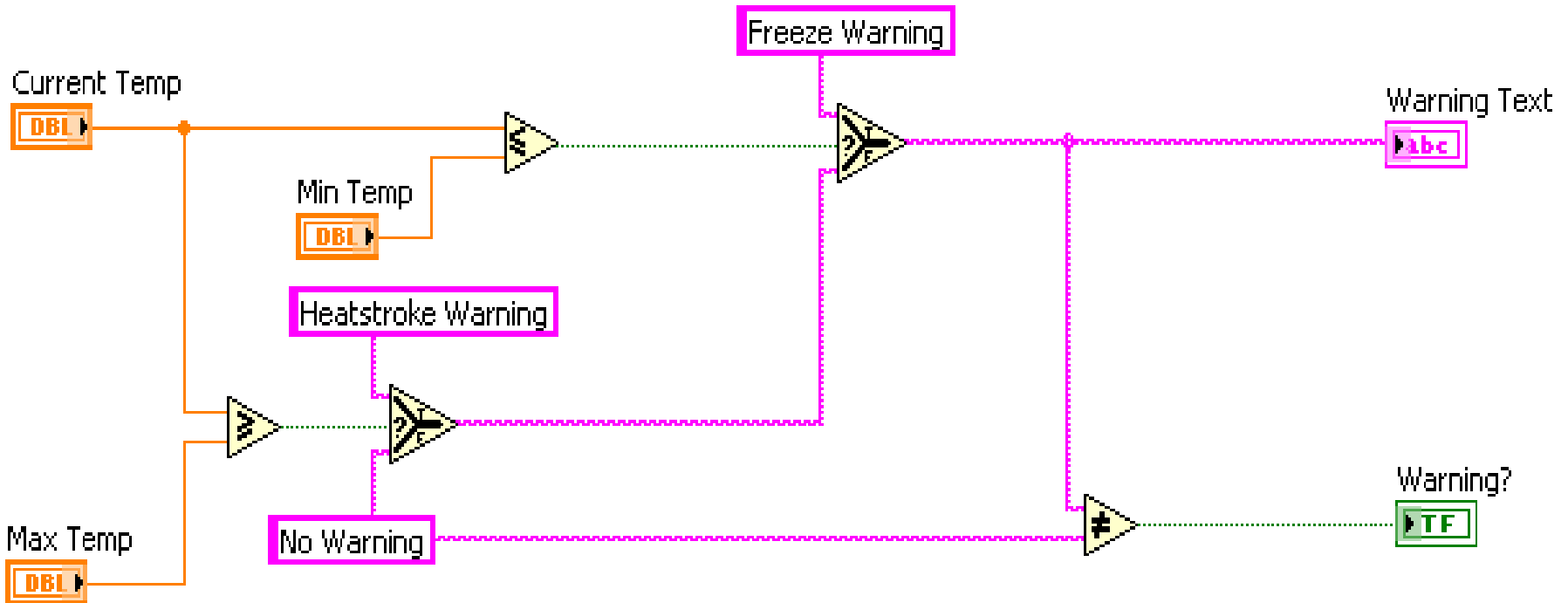




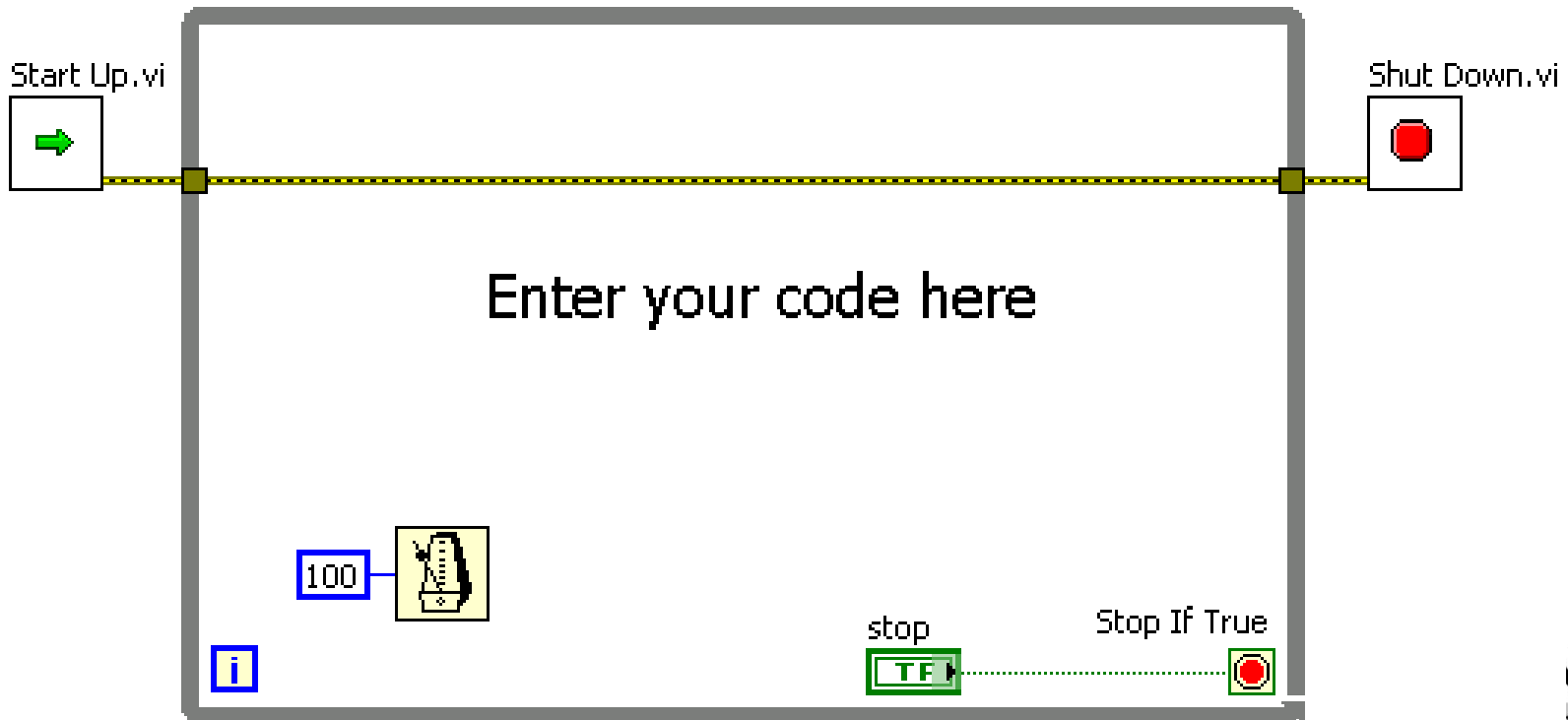
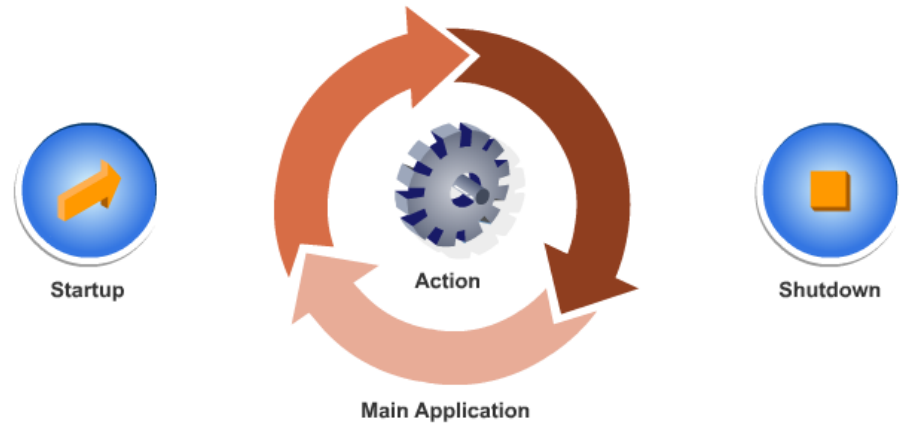
Action

Simple VI Pattern

- Single VI that takes a measurement, performs calculations, and either displays the results or records them to disk.
- Usually does not require a specific start or stop action from the user.



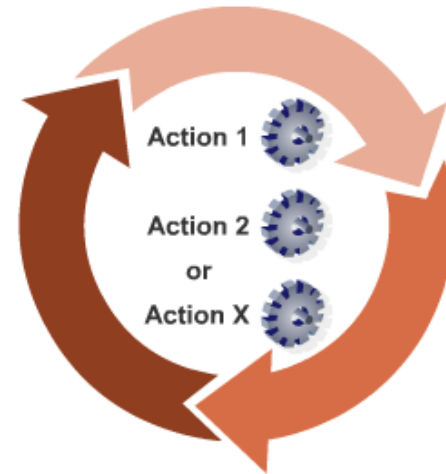
General VI Framework



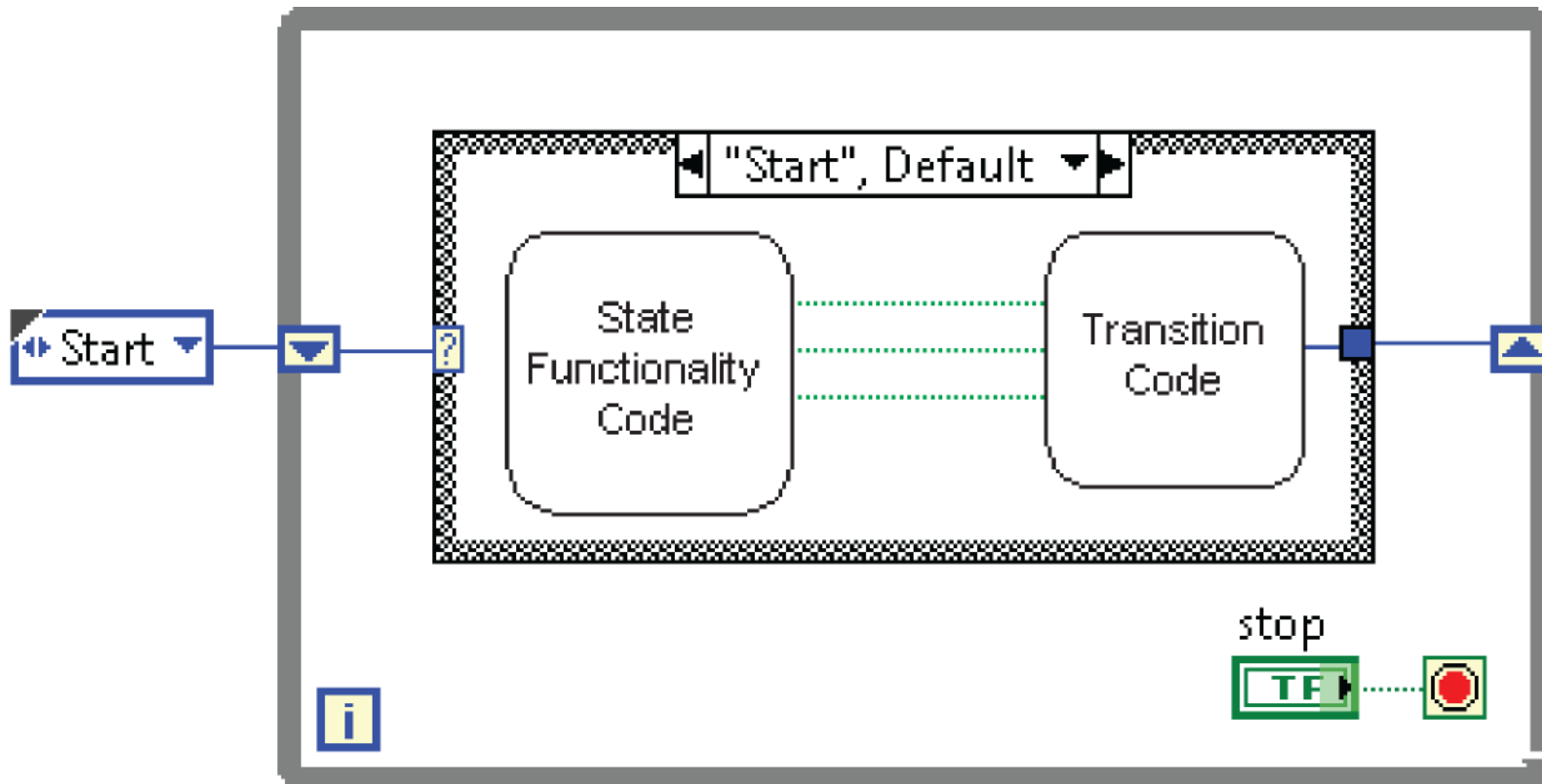
State Machine Framework



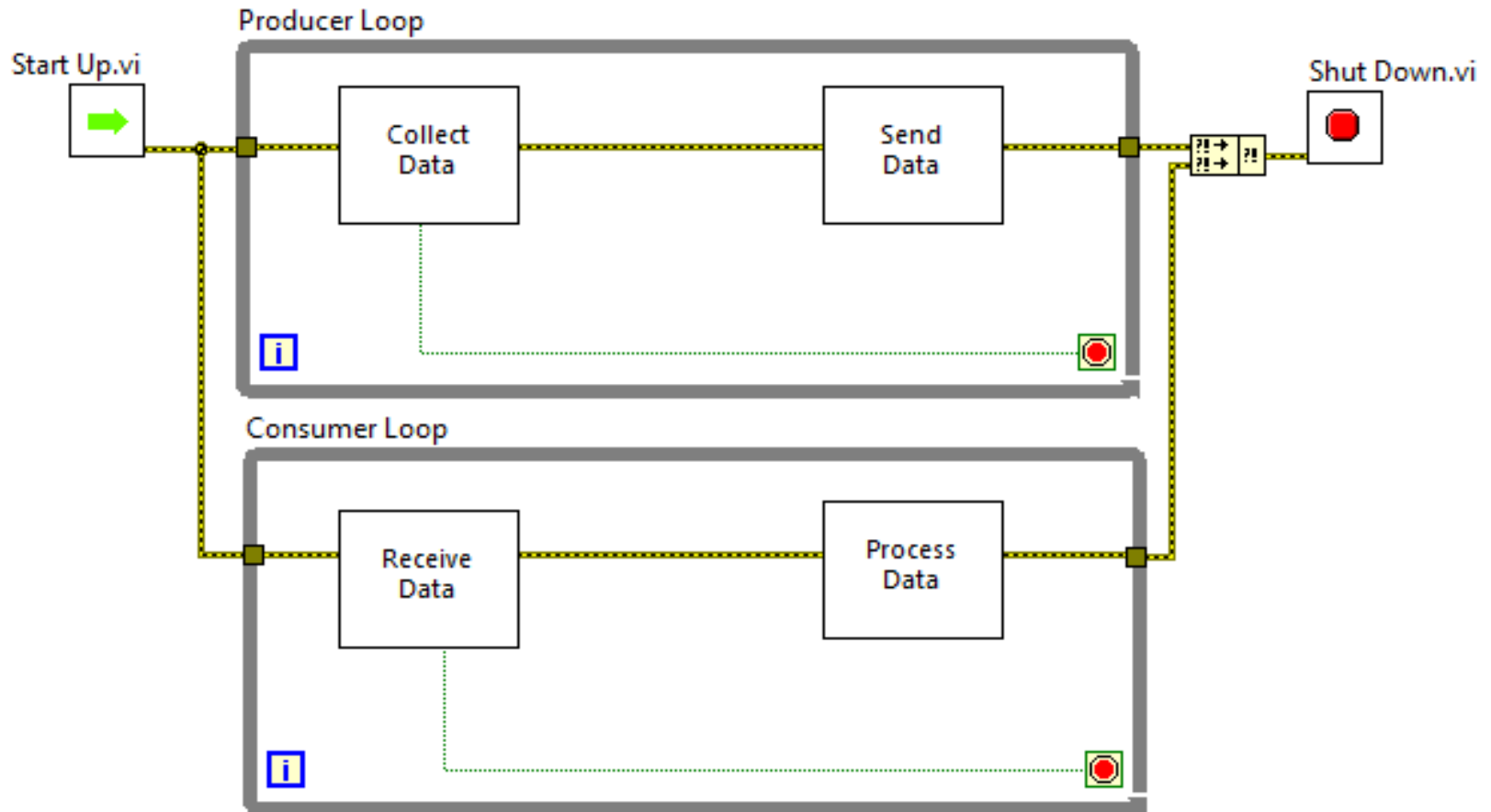
Start



Stop



Producer/Consumer Design Patterns



Choose

“Create

Create Project

Choose a starting point for the project:

- All
 - Templates
 - Sample Projects
 - Desktop
 - NI-579X

Blank Project *Templates*
Creates a blank project.

Blank VI *Templates*
Creates a blank VI.

Simple State Machine *Templates*
Facilitates defining the execution sequence for sections of code. **More Information**

Queued Message Handler *Templates*
Facilitates multiple sections of code running in parallel and sending data between them. **More Information**

Actor Framework *Templates*
Creates an application that consists of multiple, independent tasks that communicate with each other. This template makes extensive use of LabVIEW classes. **More Information**

Finite Measurement *Sample Projects*
Acquires a finite measurement and provides options for exporting the measurement to disk. This sample project is based on the Simple State Machine template. **More Information**

Continuous Measurement and Logging *Sample Projects*
Acquires measurements continuously and logs them to disk. This sample project is based on the Queued Message Handler template. **More Information**

Feedback Evaporative Cooler *Sample Projects*
Implements an evaporative cooler with hot-swappable hardware, controllers, and user interfaces. This sample project is based on the Actor Framework template. **More Information**

Continuous Measurement and Logging (NI-DAQmx) *Sample Projects*
Implements a continuous data acquisition application using NI-DAQmx. **More Information**

Finite Measurement (NI-DAQmx) *Sample Projects*
Implements a finite data acquisition application using NI-DAQmx. **More Information**

Instrument Driver Project *Templates*
Creates an instrument driver.

Simple NI-579X Streaming *Sample Projects*
Implements basic streaming functionality to continuously stream data to or from the FPGA target. **More Information**

Additional Search

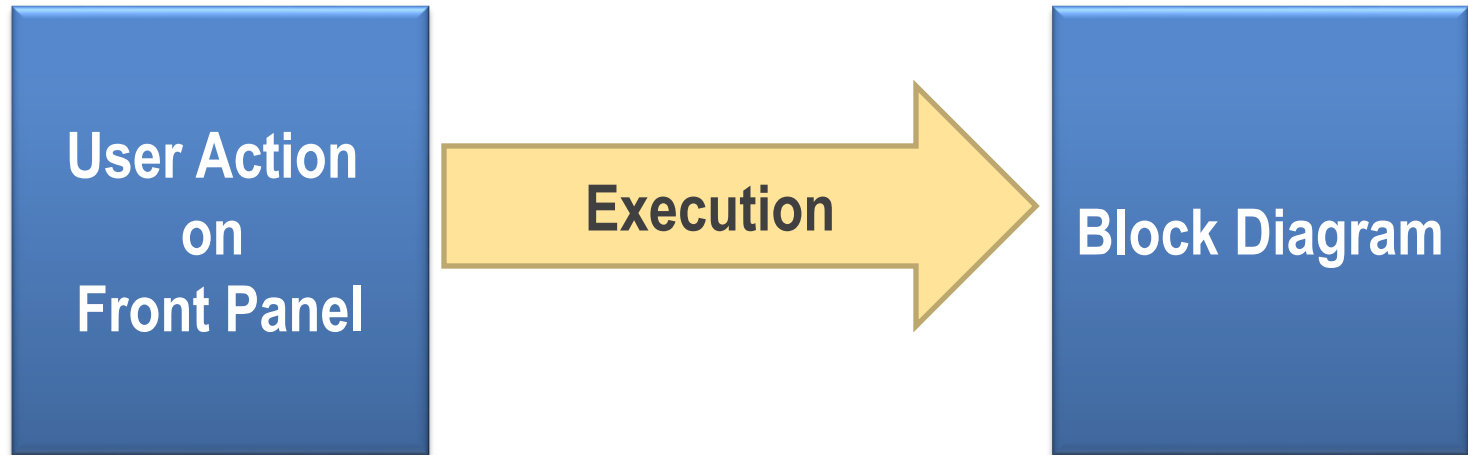
Finish Cancel Help

D. Event-Driven Programming



Event — An asynchronous notification that something has occurred

Event-Driven Programming — Method of programming where the program waits for an event to occur before executing one or more functions



Polling versus Event Structures

Polling

- Method of event-based programming where a loop must continually run code to check if changes have occurred.
- Polling the front panel requires a significant amount of CPU time.
- Polling can fail to detect changes if they occur too quickly.

Event Structures

- Events in Event structures eliminate the need to poll the front panel.
- Benefits of using Event structures:
 - Reduces the CPU requirements of the program.
 - Simplifies the block diagram code.
 - Guarantees that the block diagram can respond to all interactions the user makes.



E. LabVIEW SRS DAQ Program Structure

The screenshot displays the LabVIEW SRS DAQ program interface, titled "APV-srs Data Acquisition for RD-51 Main Panel". The interface is organized into several functional panels:

- Program Status:** Shows the running environment with "Stop" buttons for Command Handler, UDP Receiver, Events Engine, and Online Monitor Consumer.
- UDP/Ethernet Parameters:** Configures listening IP, port, timeout, and frame size. Includes a "UDP Receiver" section with an "Emulation" button and a "Holdoff (ms)" slider.
- APV Selection:** A "Board Selection" panel with 8 green indicator lights, all labeled "Master APV".
- srs-to-APV Controls:** Controls trigger mode (set to "Internal Trigger (no Test Pulse)"), trigger NIM polarity (set to "Norm."), test pulses (set to "Fast"), and time slots to be acquired (set to 12). Includes "srs Timing" controls for APV, Test, and Data delays.
- File Saving Parameters:** Manages data and pedestal files, including a "Basic Path for File Saving" field and a "Data File basic Name" field.
- RUN Status:** Shows "Status" as "Run Idle" with a "Run Number" of 0. Includes a "Run Type" dropdown set to "Normal Run" and a "# of required Events" set to 1000.
- Queues Status (# Events):** Displays "Received" status and "Trailing Counter" for "Building" and "Analysing" queues, both at 0.
- Channel/Strips Mapping:** Includes "Parameters" and "Strip/Pads Mapping" tabs. The "Channel/Strips to analyse" section shows "APV Hybrid to be extracted" (0), "# Ch. per APV" (0), "Sampling time [s]" (0), "# Samples/Ch." (0), and "Strips Reordering" (checked).
- Online Analysis:** Features "Interpr. & Noise rej." controls for Complement Data, Baseline, Reject Comm. Mode, and Common Mode Threshold. Includes "Header detection" controls for Threshold and Limits.
- Error Status:** Shows "Current error" with "status" (green), "code" (d 0), and "source" fields.

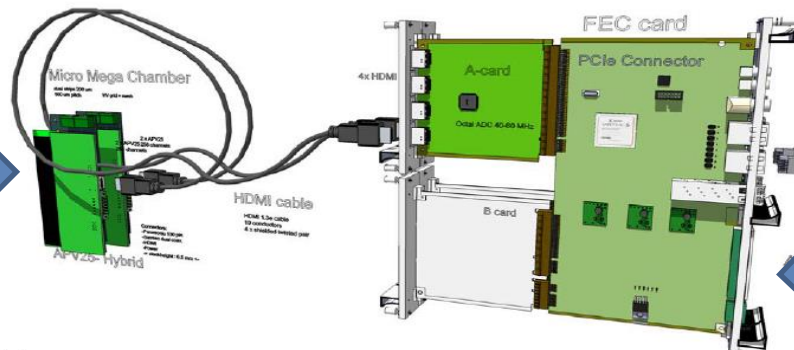
The interface also includes a menu bar (File, Edit, View, Project, Operate, Tools, Window, Help), a toolbar, and a search field. The bottom status bar shows the file path "APV-Daq.lvproj/My Computer".



The LabVIEW DAQ-SRS

Chambers → ON Board Electronics (signal conditioning) →
DAQ System → Bus connection → PC running LV (≥ 2012)

μ MGas,
GEM,...
chambers



UDP connection



“RD51-srs”
LabVIEW Project



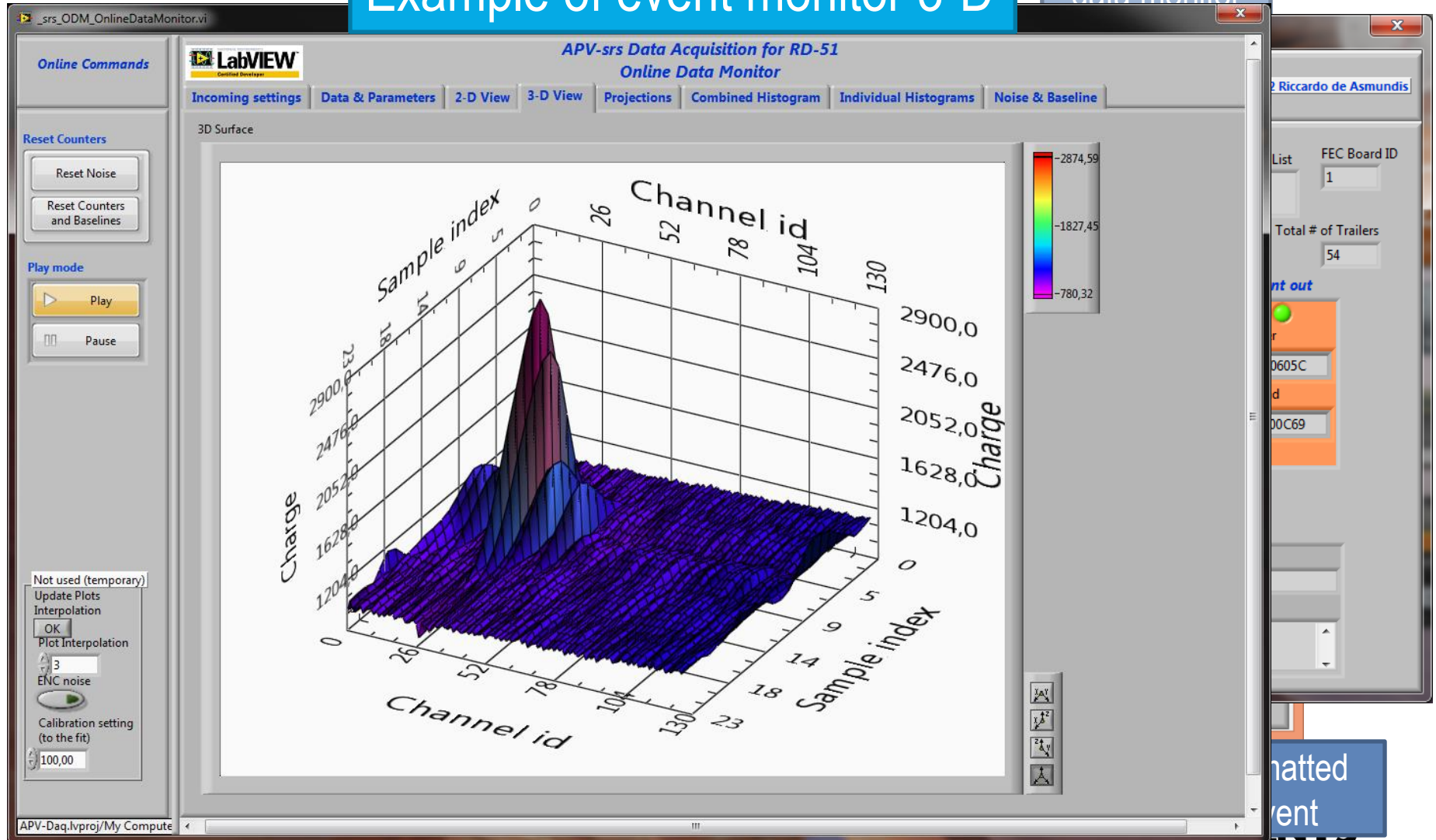
NATIONAL INSTRUMENTS
LabVIEW™



Program Features

Example of event monitor 3-D

UDP Codes
data monitor

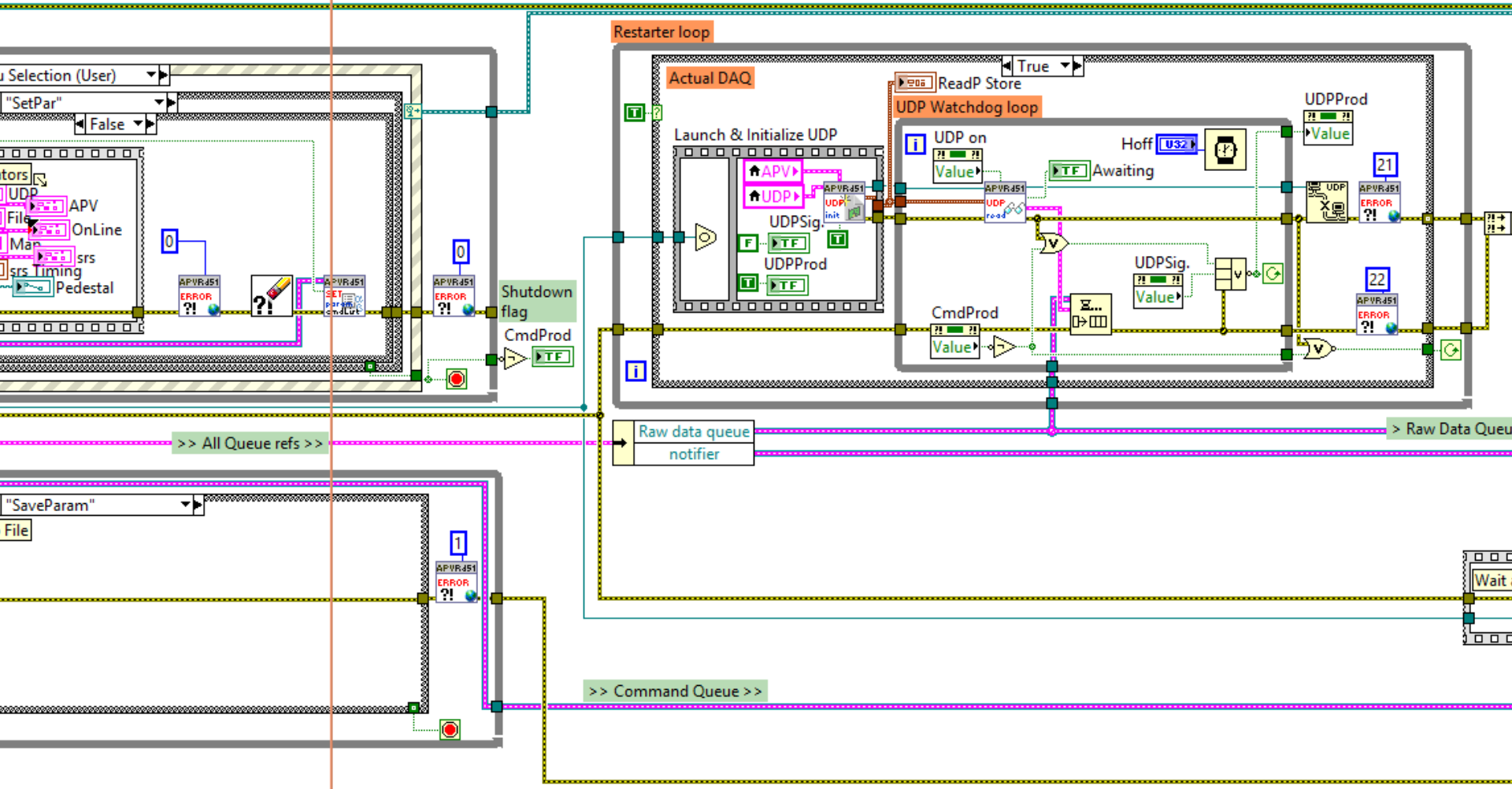


sample per visible slot,

Initialise UDP Communication
Use of the Occurrence ensures
this process starts after decision
of UI Command Consumer

Data Producer for UDP acquisition (or emulation)

& Consumer)





See you in the Lab this afternoon
For demonstrations