NI CONNECT

2023 AUSTIN

HAL-chemy

Turing Devices into Gold

Testeract

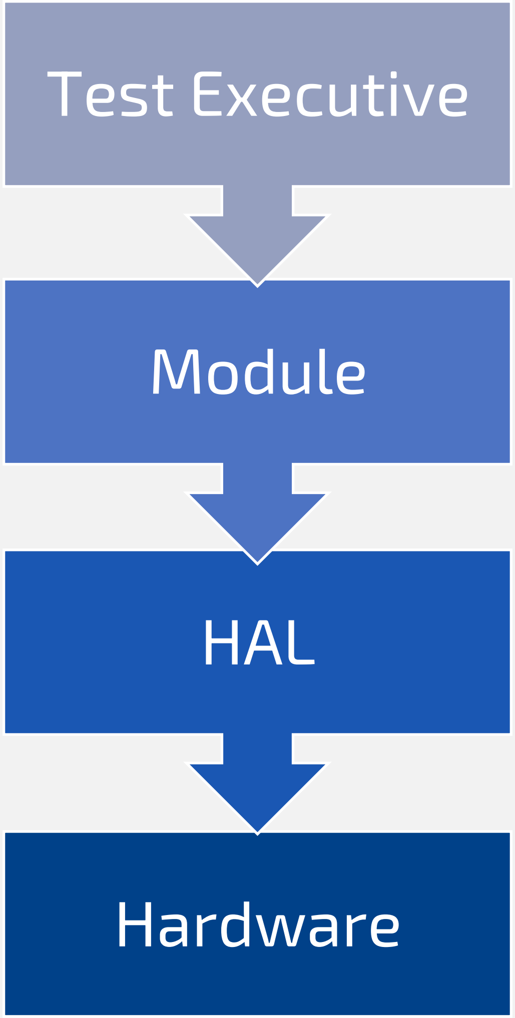
Intros

- Quentin “Q” Alldredge
  - Certified LabVIEW Architect
  - LabVIEW Champion
  - TASC Architect, Testeract
  - Harry Potter Fanatic

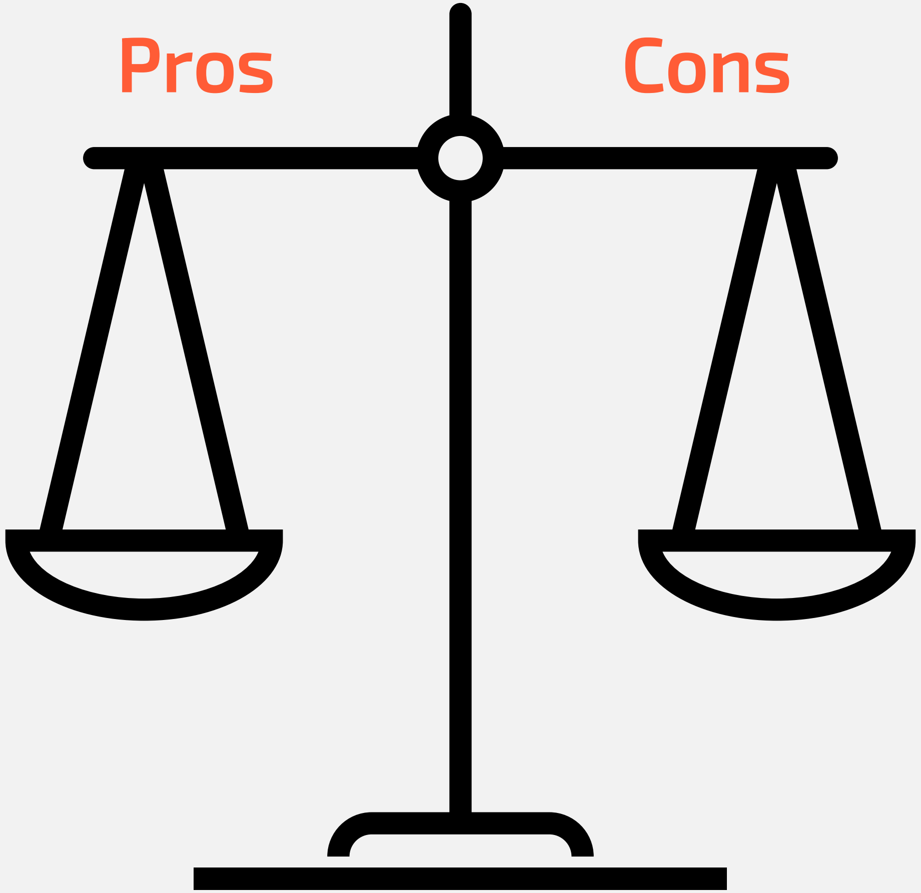
- Kevin “KDub” Shirey
  - Certified LabVIEW Architect
  - LabVIEW Champion
  - HAL Architect, Testeract
  - Shrek

What is a HAL?

- Short for Hardware Abstraction Layer
- An interface that defines how the hardware communicates to the rest of the application
- Allows for hardware to be changeable without affecting the rest of the application



Why have a HAL?



Pros

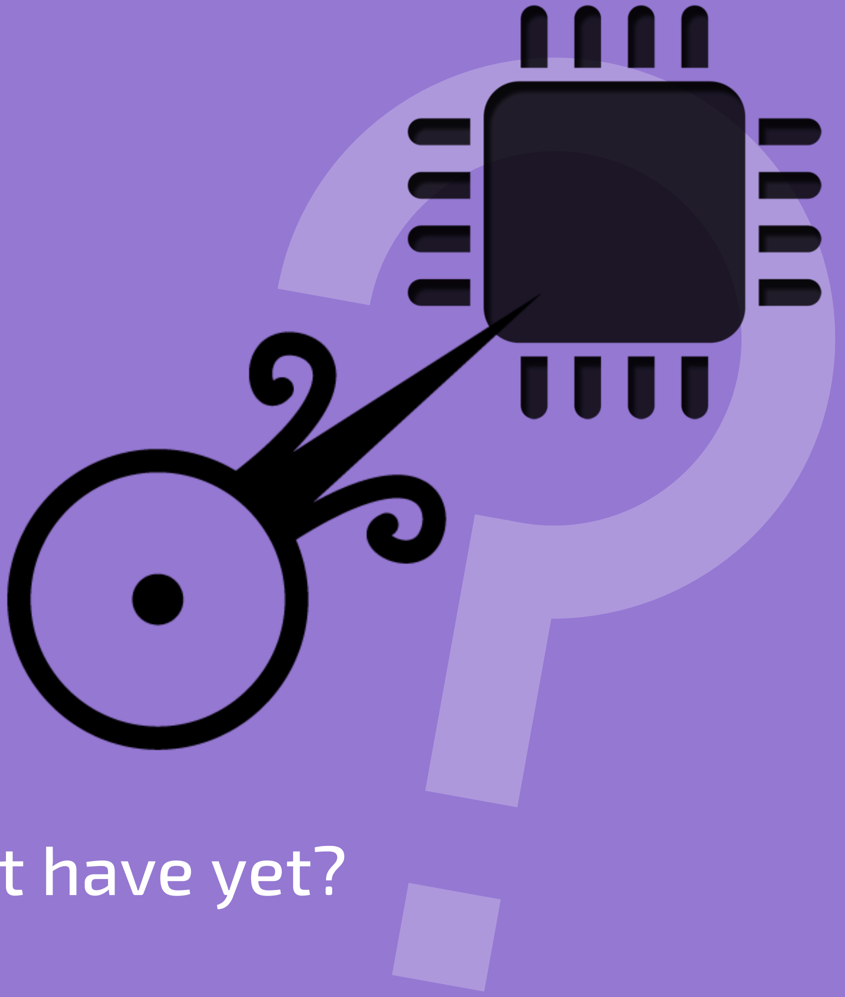
- Without it new instruments are harder to add later (technical debt)
- Parallelizing Development
- Protects against obsolescence

Cons

- Higher up front development cost
- “I’m only developing for one set of hardware”
- “The required hardware will never change”

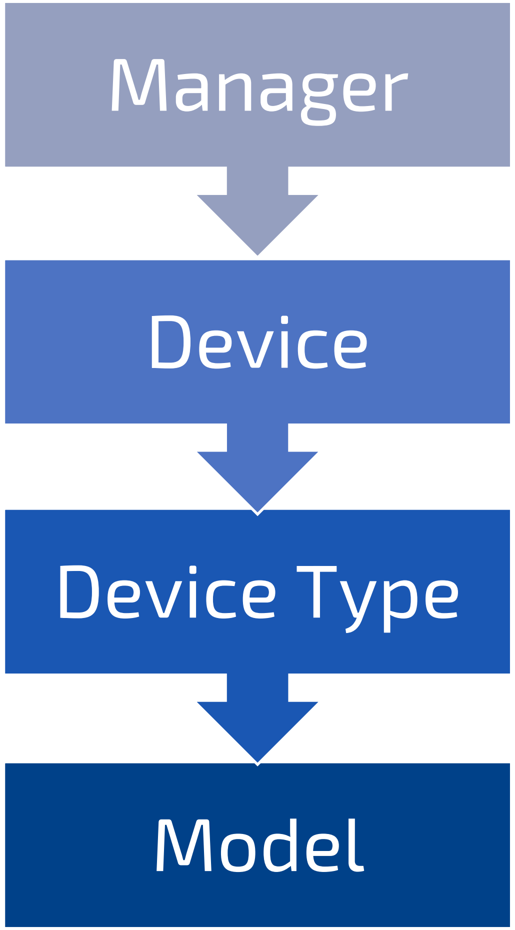
HAL Fundamental Questions

- ✓ How do you organize your HAL?
- ✓ What features do you cover?
- ✓ How do you manage connections?
- ✓ How do you handle bitness?
- ✓ Where do you handle the DUT/UUT?
- ✓ How do you Debug a HAL?
- ✓ What do you do to test hardware you don’t have yet?
- ✓ Where do you perform logging?
- ✓ How do you handle deploying the HAL?



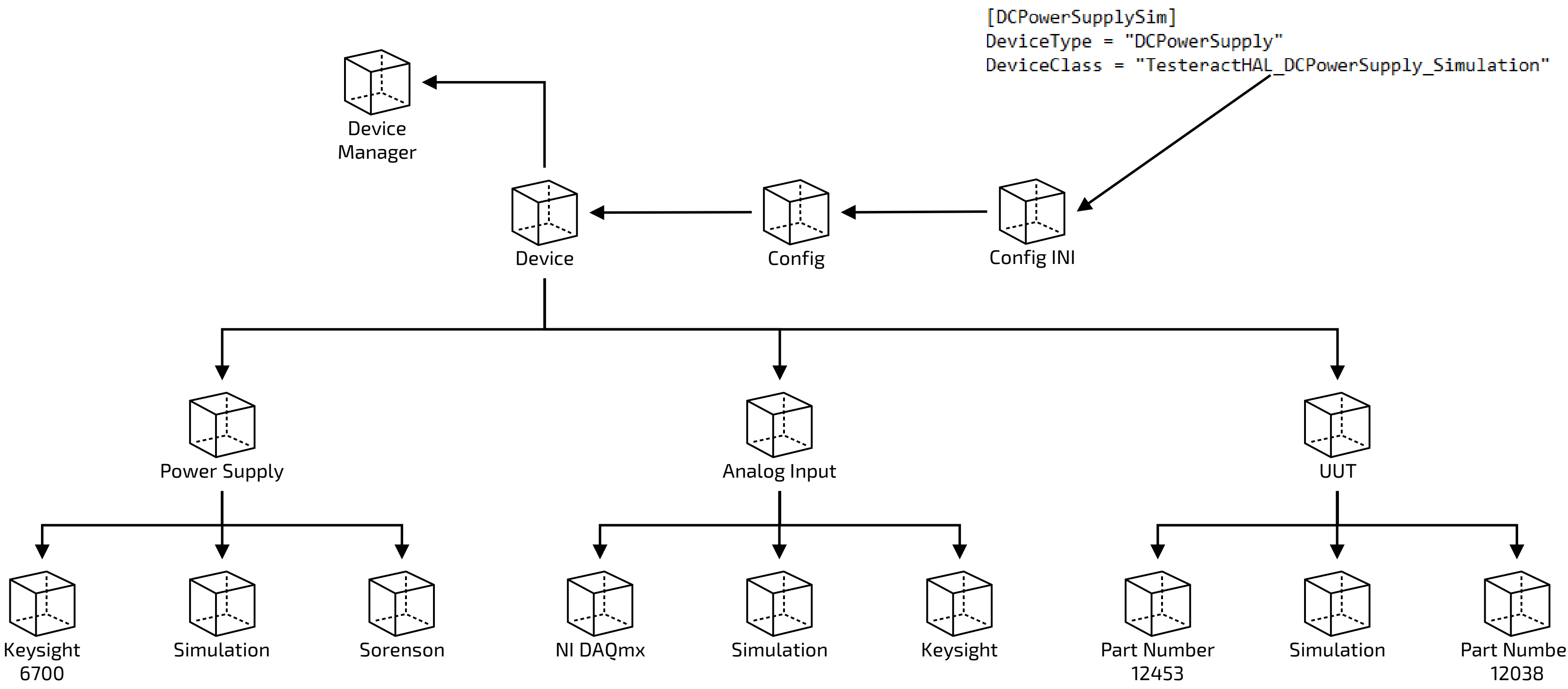
How do you organize your HAL?

- Object Oriented Class Hierarchy
  - Organized
  - Separation of Duties
  - Encapsulation



How do you organize your HAL?

- Object Oriented Class Hierarchy
- Flexible
  - Dynamic Dispatching
  - Simulation
  - Manager keeps Connections



```
[DCPowerSupplySim]
DeviceType = "DCPowerSupply"
DeviceClass = "TesteractHAL_DCPowerSupply_Simulation"
```

Device Manager

Device

Config

Config INI

Power Supply

Analog Input

UUT

Keysight 6700

Simulation

Sorenson

NI DAQmx

Simulation

Keysight

Part Number 12453

Simulation

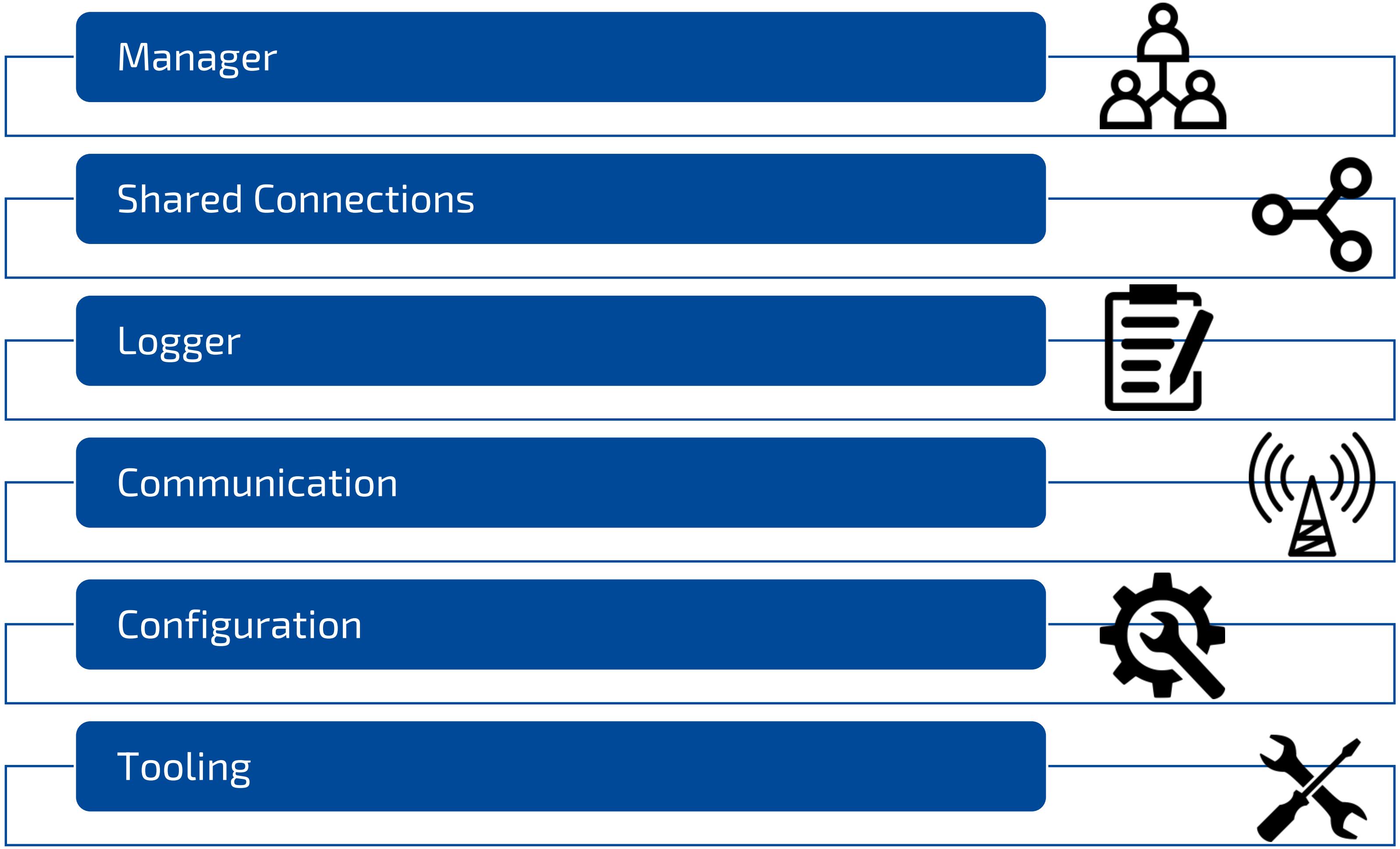
Part Number 12038

How do you organize your HAL?

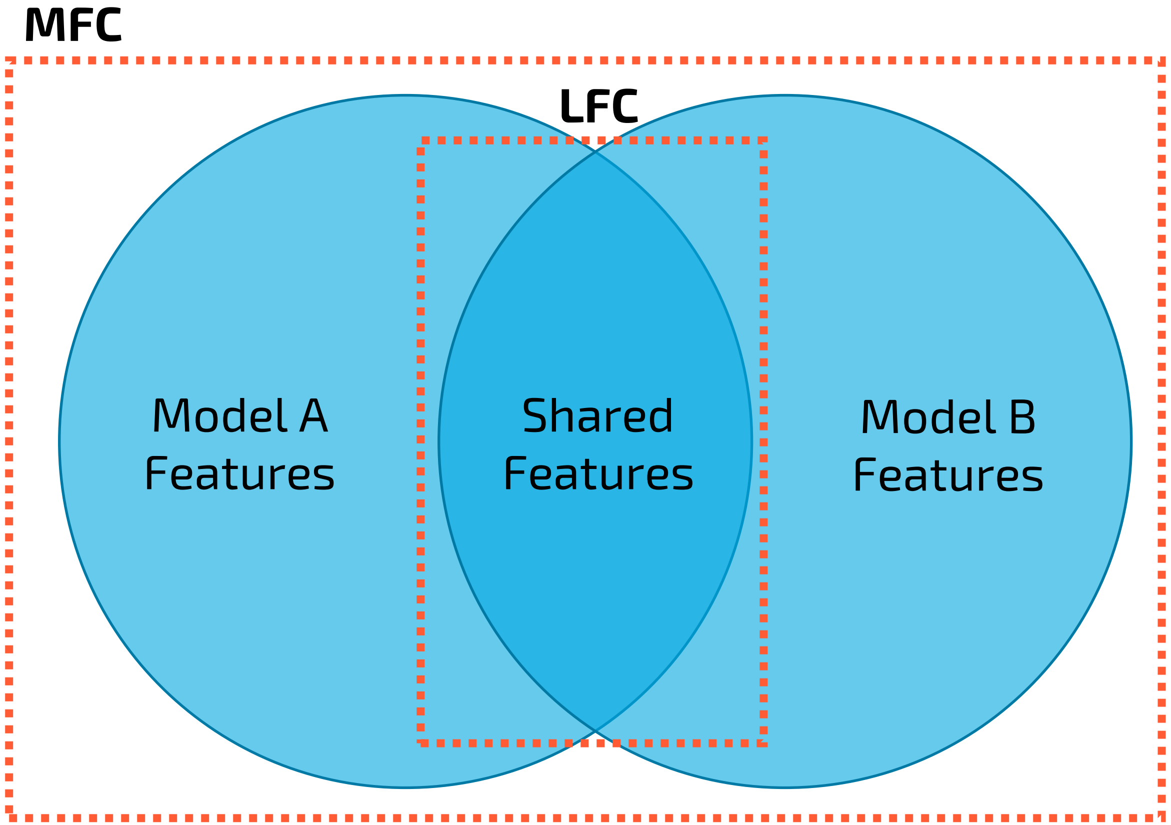
- Object Oriented Class Hierarchy
- Flexible
- Extensible
  - Add New Models
  - Configurable
  - Testable

How do you organize your HAL?

- Support Classes
- Communication
  - Bus Support
  - gRPC
  - Remote Connections
- Tooling
  - Project Templates
  - Scripting
  - Documentation Creation
  - Unit Testing



What features do you cover?



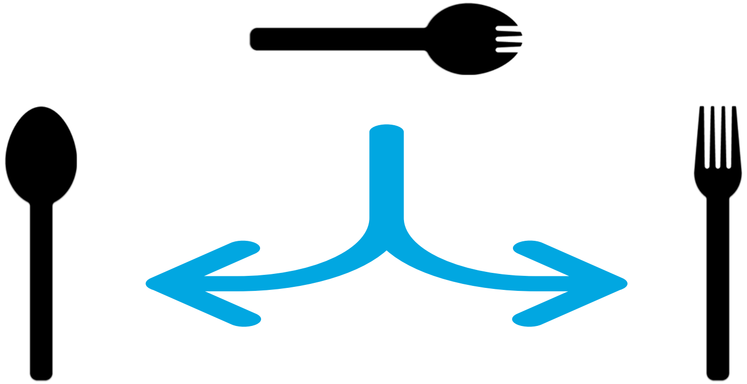
- What features should be supported?
- MFC = Most Feature Coverage
  - Cover all features
- LFC = Least Feature Coverage
  - Cover only shared features
- Concern for Breaking Changes

How do you add multi-function devices?

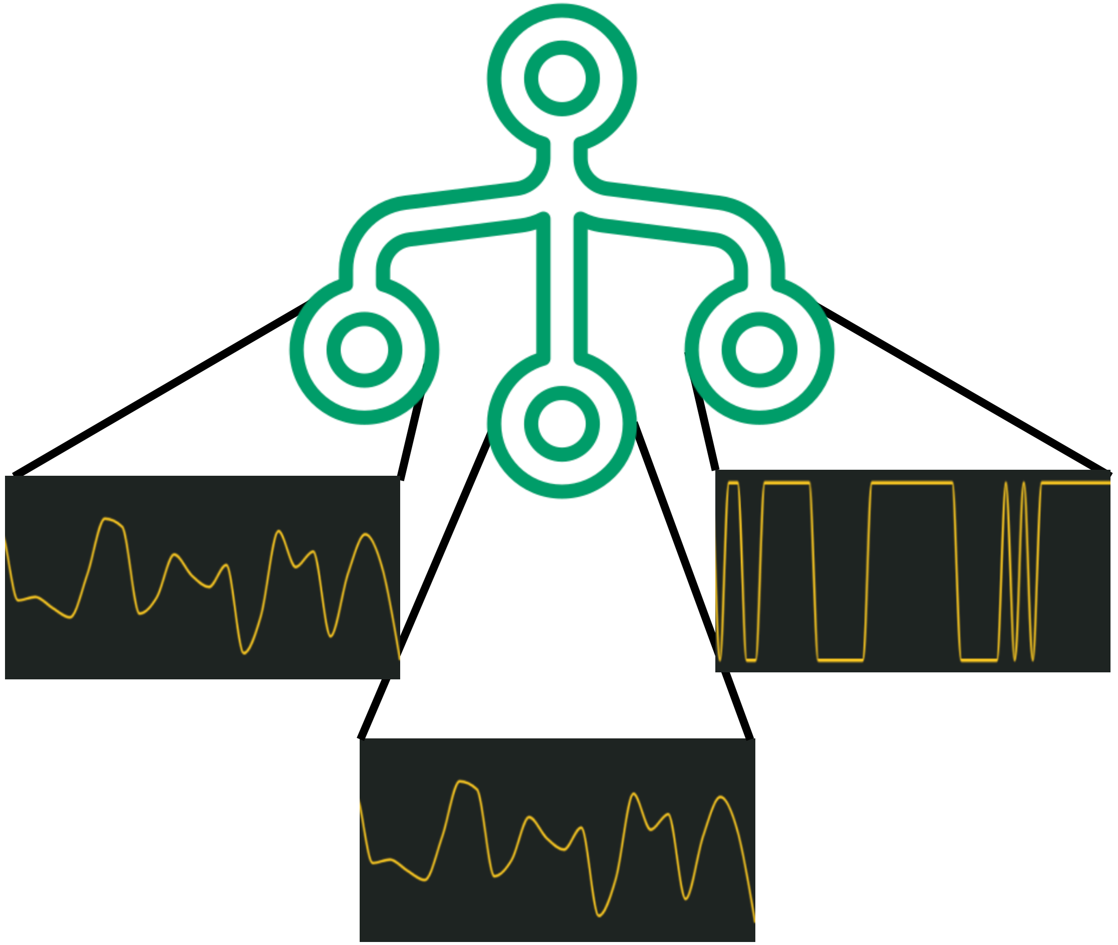
- A Device does multiple functions

    - Implement as a new, special device type
        - Will need to reimplement all API methods

    - Implement as multiple devices under their respective device types
        - sometimes still need shared connections



How do you add multi-channel devices?



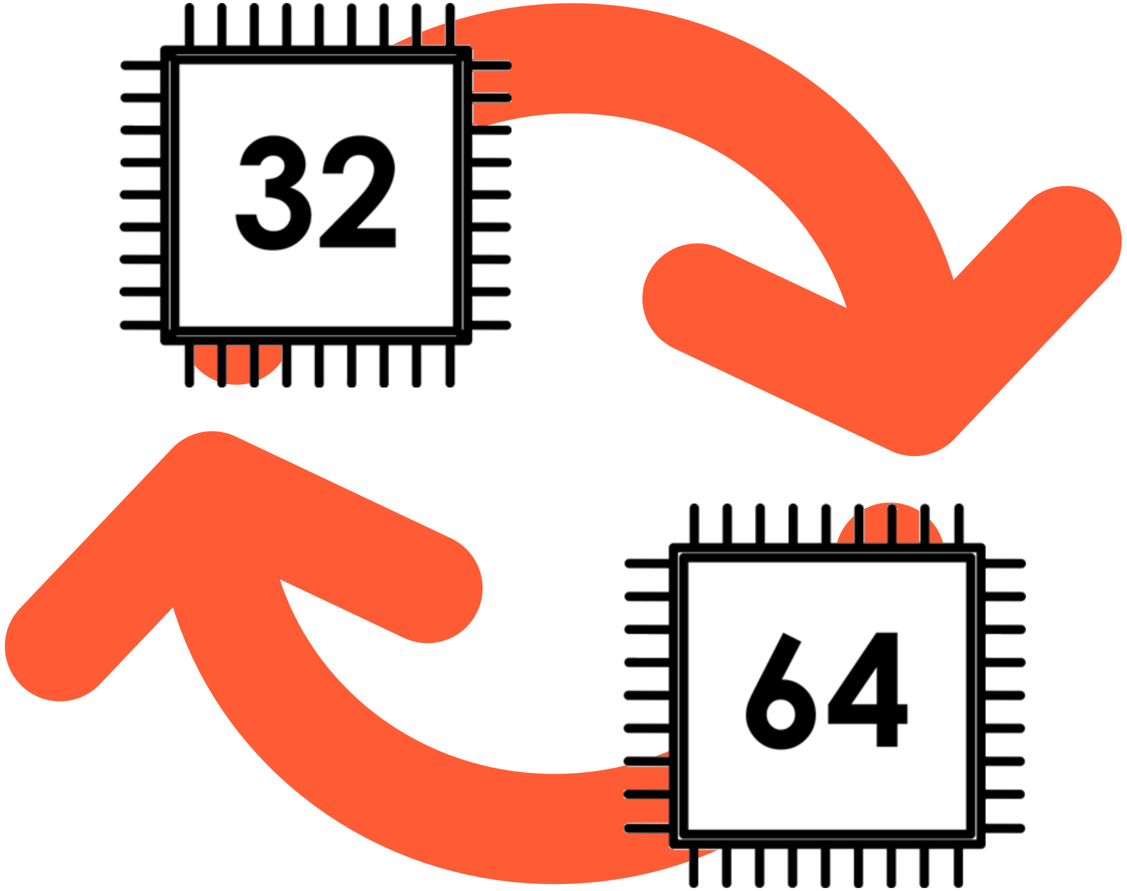
- Multiple connections
  - Can be talked to independently
  - Maybe each has different functionality
- Like a Scope or Switch with multiple channels
- Implement as two devices
  - Different devices if different functionality
  - Two instances of the same device if same functionality

How do you manage connections?

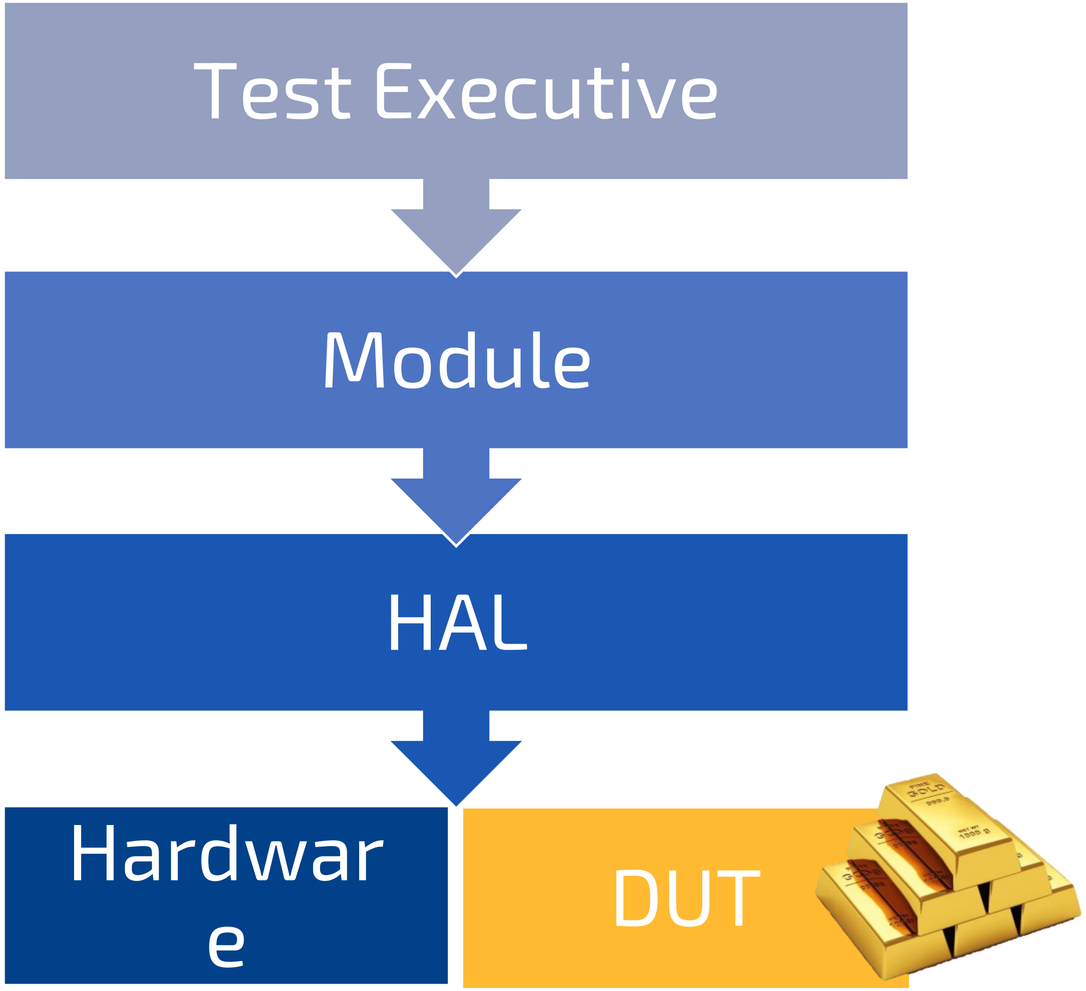
- Incorporate HAL into application framework
  - Difficult to share connections
- Each Device could own its own connection
- Manager
  - Manage connections without being directly wired
  - Allows HAL to be used in any framework
  - Allows HAL to be used across LabVIEW and TestStand at the same time

How do you handle bitness?

- x86 or x64?
- LabVIEW/TestStand Versions?
- Cross bitness methods TCP
  - Remote server per device
    - TCP/IP
    - JSON
    - Database
    - Object Flatten (encrypted)
  - gRPC
    - Instrument Studio



Where do you handle the DUT/UUT Coms?

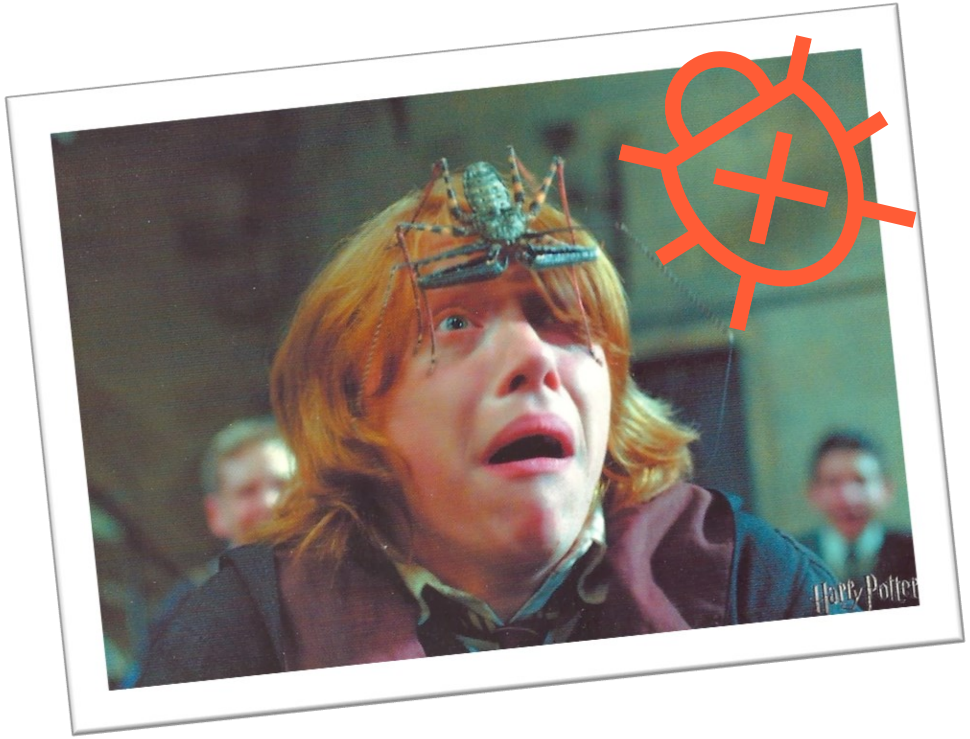


- Main Application
  - Test Script
  - Highly coupled to main application

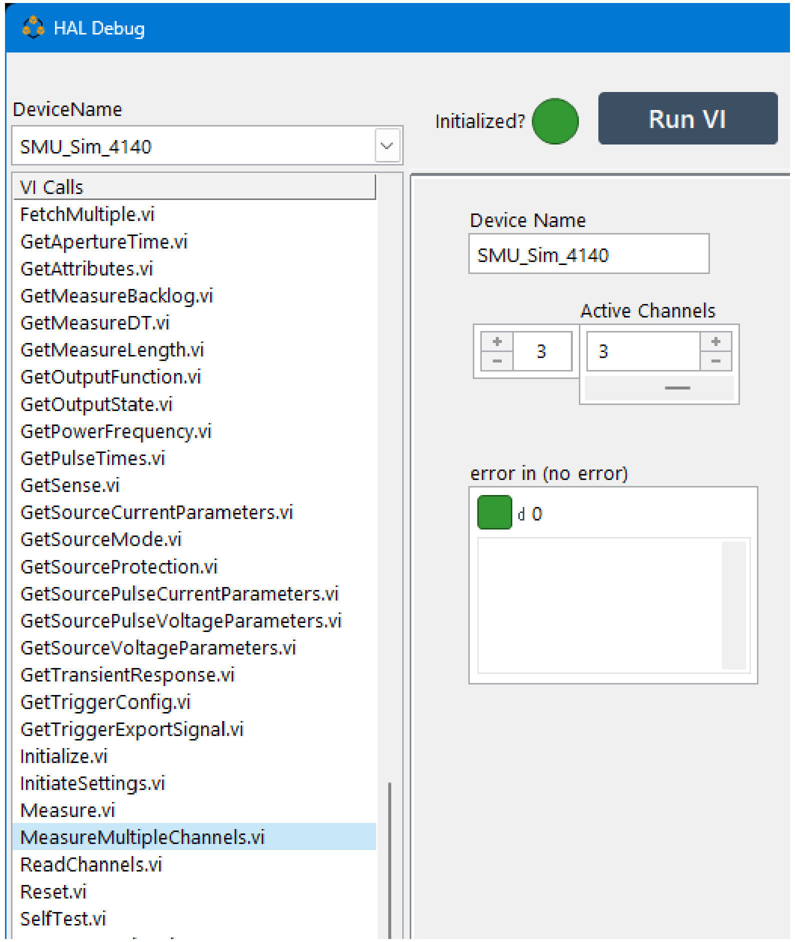
- Treat as Device in HAL
  - One or More Device
  - This brings all the features the HAL has to be used on the DUT
  - Simulate the DUT as well
  - It is just another instrument

How do you Debug a HAL?

- Can you exercise every function/feature of the device?
- How tightly coupled is the hardware to the end application?
  - Too Tight – Untestable Separately
  - Modular – can test independently
- How do you add Logging?



How do you Debug a HAL?



DeviceName: SMU_Sim_4140

Initialized? ●

VI Calls
- FetchMultiple.vi
- GetApertureTime.vi
- GetAttributes.vi
- GetMeasureBacklog.vi
- GetMeasureDT.vi
- GetMeasureLength.vi
- GetOutputFunction.vi
- GetOutputState.vi
- GetPowerFrequency.vi
- GetPulseTimes.vi
- GetSense.vi
- GetSourceCurrentParameters.vi
- GetSourceMode.vi
- GetSourceProtection.vi
- GetSourcePulseCurrentParameters.vi
- GetSourcePulseVoltageParameters.vi
- GetSourceVoltageParameters.vi
- GetTransientResponse.vi
- GetTriggerConfig.vi
- GetTriggerExportSignal.vi
- Initialize.vi
- InitiateSettings.vi
- Measure.vi
- MeasureMultipleChannels.vi
- ReadChannels.vi
- Reset.vi
- SelfTest.vi

Device Name: SMU_Sim_4140

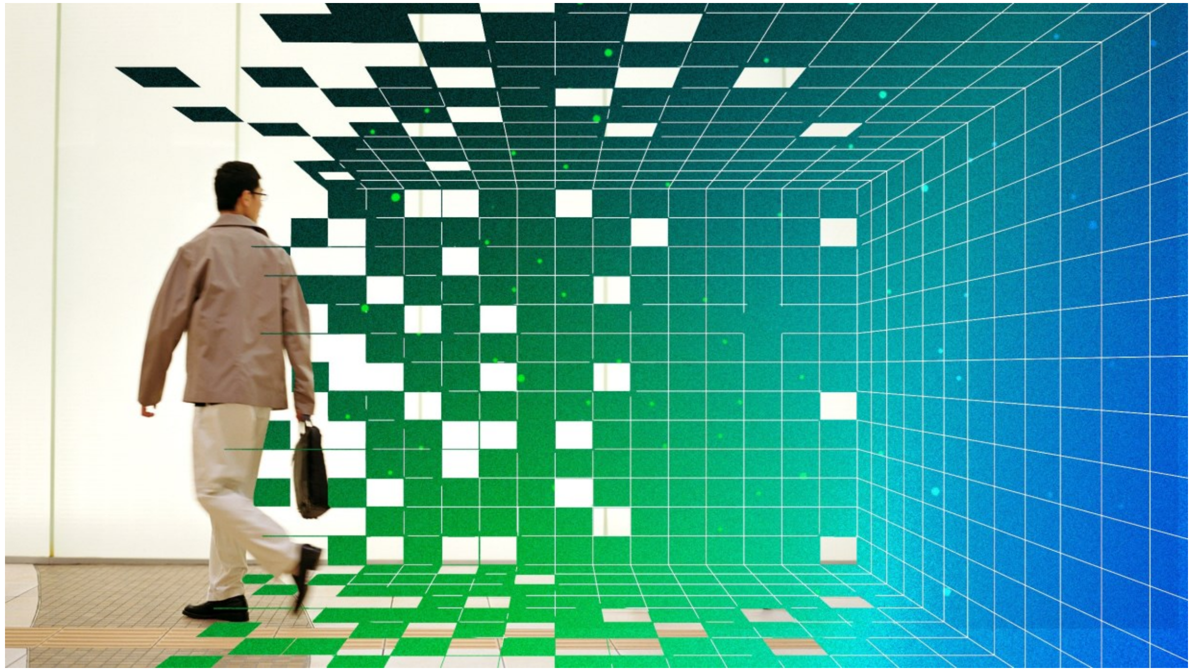
Active Channels: 3 | 3

error in (no error): 🟩 d 0

- Create a bookshelf of drivers precompiled as Packed Project Libraries (PPL)
- Debug GUI allows for testing all methods of every device
  - Debug GUI available stand alone
  - Or integrated into TASC for Record and Playback
- Debugging left on in PPLs to allow for Highlight Execution in LabVIEW/TestStand

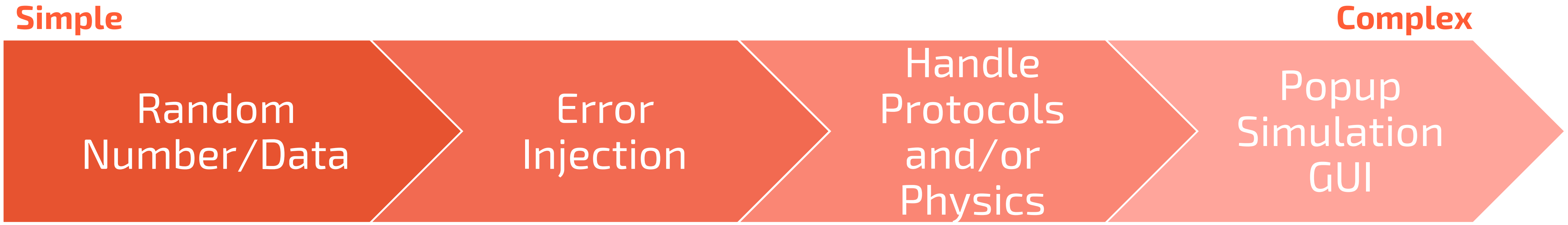
What do you do to test hardware you don't have yet?

- Lead times and hardware availability
- Creation of Simulation Classes for each device allows development to continue during wait times

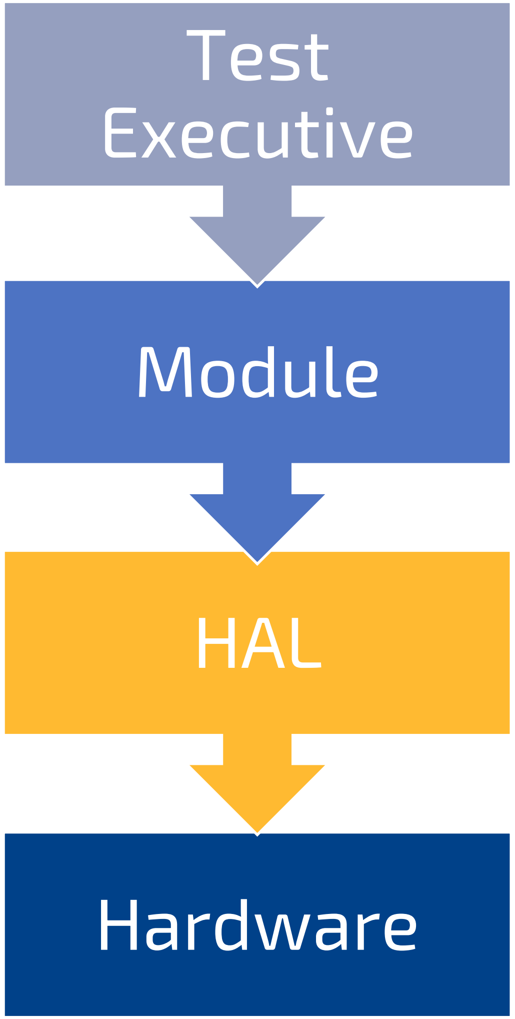


What do you do to test hardware you don't have yet?

- How much effort should be made to make the Simulation act like the actual device?
  - Simple – Random number output
  - Error Injection – Purposely cause/exercise an error path
  - Smarter – Handle protocols or real-world timing and physics
  - Popup Simulation GUI – Provide user interaction with the device



Where do you perform logging?



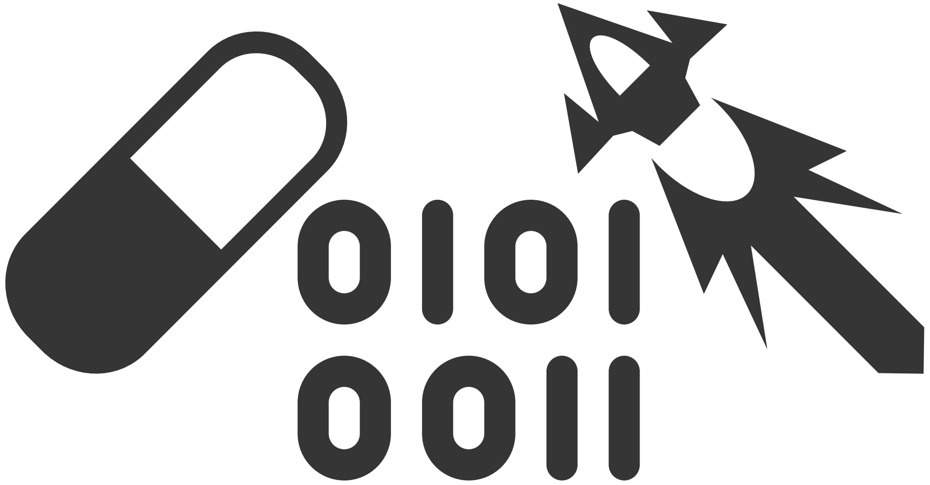
- This is the raw readings and not the end report.
- At what level should it be done?
  - Built into Application
  - Built into HAL
- How do you define what to record?
  - Log every call
  - Selectively log by device
  - Selectively log by function
  - Choose different timing
- How do you decide where to log to?
  - File (ASCII, JSON, TDMS), Database, etc.

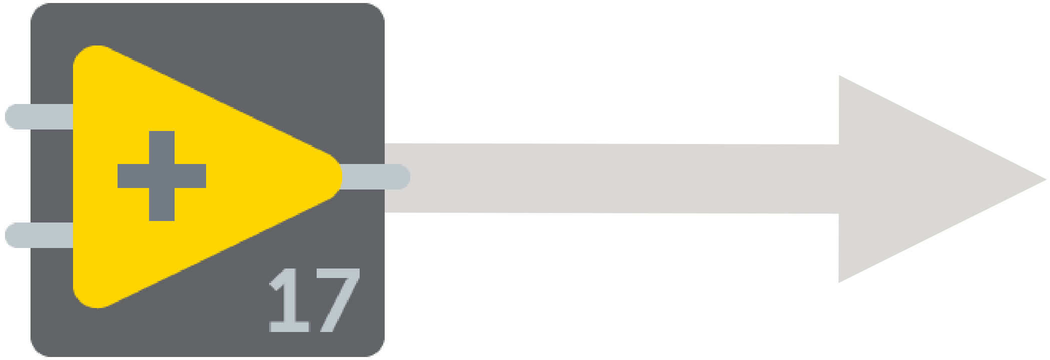
How do you handle deploying the HAL?

- How highly coupled is your application?
  - Monolithic Applications are more time consuming to upgrade and test
  - Modularity allows for regression testing of individual components

How do you handle deploying the HAL?

- PPLs
  - Encapsulates code
  - Distributes precompiled
  - Faster Execution
  - Changes from LV2017 allow PPLs being forward compatible





- Do you have different test stations or need to role back to a previous version?
  - Handling different system States

HAL Video

TASC Video

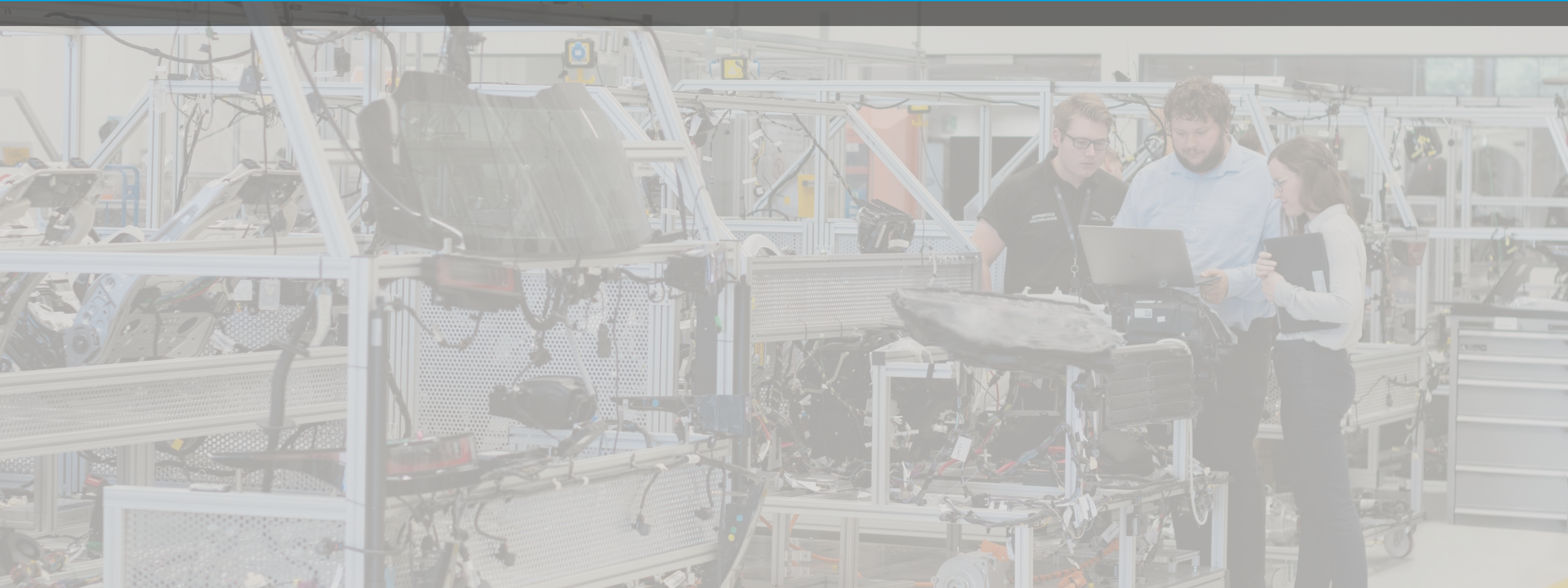
Can you record & playback a test scenario?

- Record test data and play it back as the device
- Simulate a test, playback on real equipment
- Edit and filter the recorded data
- Save and export the recorded scenario
    - Artifacts as CSV File
    - Not a TestStand developer, writes sequences for you

Demo

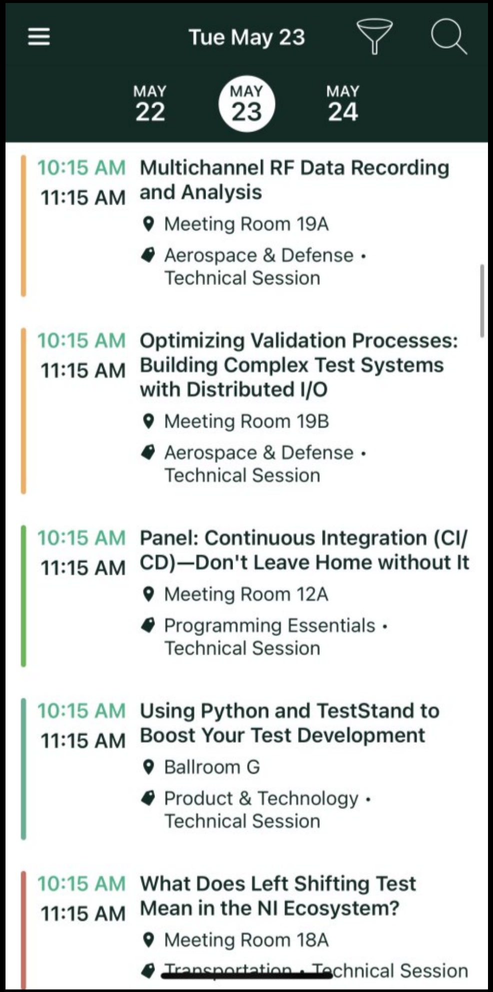
Questions

Testeract



Give us your feedback!
Quick 2 Question Survey

In the mobile app, click into the session you would like to provide feedback for



Tue May 23

MAY 22 | MAY 23 | MAY 24

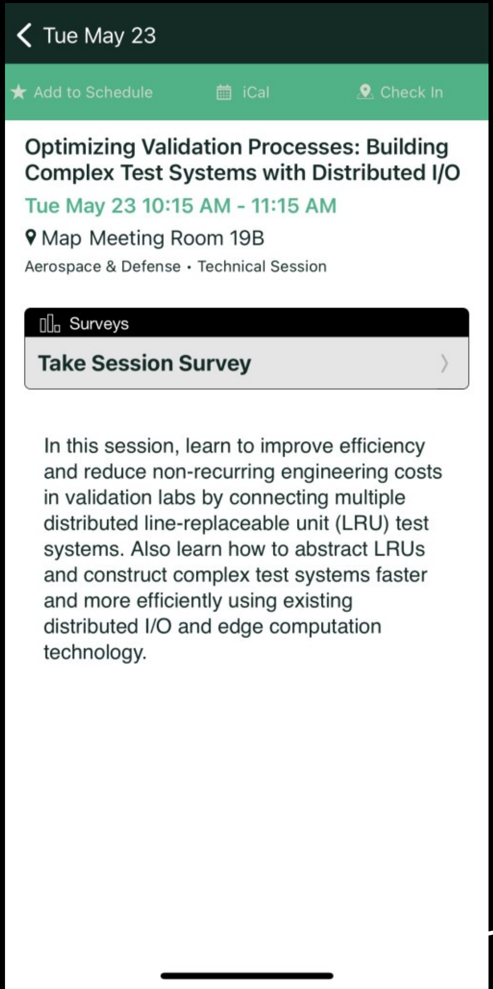
10:15 AM 11:15 AM Multichannel RF Data Recording and Analysis
Meeting Room 19A
Aerospace & Defense • Technical Session

10:15 AM 11:15 AM Optimizing Validation Processes: Building Complex Test Systems with Distributed I/O
Meeting Room 19B
Aerospace & Defense • Technical Session

10:15 AM 11:15 AM Panel: Continuous Integration (CI/CD)—Don't Leave Home without It
Meeting Room 12A
Programming Essentials • Technical Session

10:15 AM 11:15 AM Using Python and TestStand to Boost Your Test Development
Ballroom G
Product & Technology • Technical Session

10:15 AM 11:15 AM What Does Left Shifting Test Mean in the NI Ecosystem?
Meeting Room 18A
Transportation • Technical Session



Optimizing Validation Processes: Building Complex Test Systems with Distributed I/O
Tue May 23 10:15 AM - 11:15 AM
Map Meeting Room 19B
Aerospace & Defense • Technical Session

Surveys
Take Session Survey

In this session, learn to improve efficiency and reduce non-recurring engineering costs in validation labs by connecting multiple distributed line-replaceable unit (LRU) test systems. Also learn how to abstract LRUs and construct complex test systems faster and more efficiently using existing distributed I/O and edge computation technology.

Click “Take the Session Survey”