MAX PLANCK
SEMICONDUCTOR
LABORATORY

THE VIRTUAL INSTRUMENT SERVER

A look behind the scenes

OUTLINE

- **The Virtual Instrument Server**

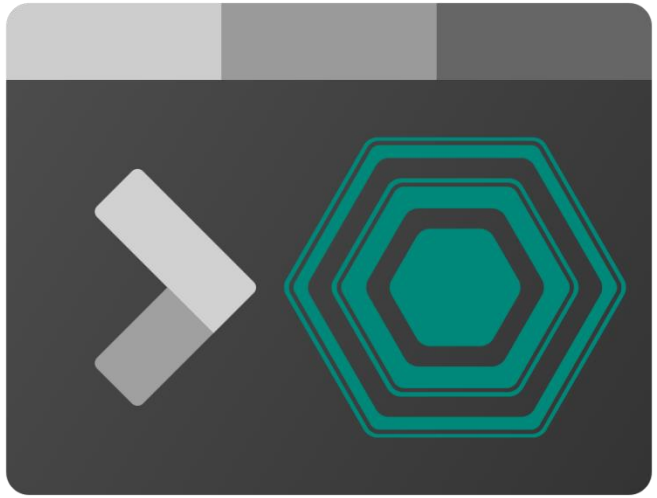
  And its role in the Virtual Instrument Suite

- **Components of the Virtual Instrument Server**

  Building blocks to represent an (almost) arbitrary system

- **Short-comings and future plans**

  What and how to improve in the near future

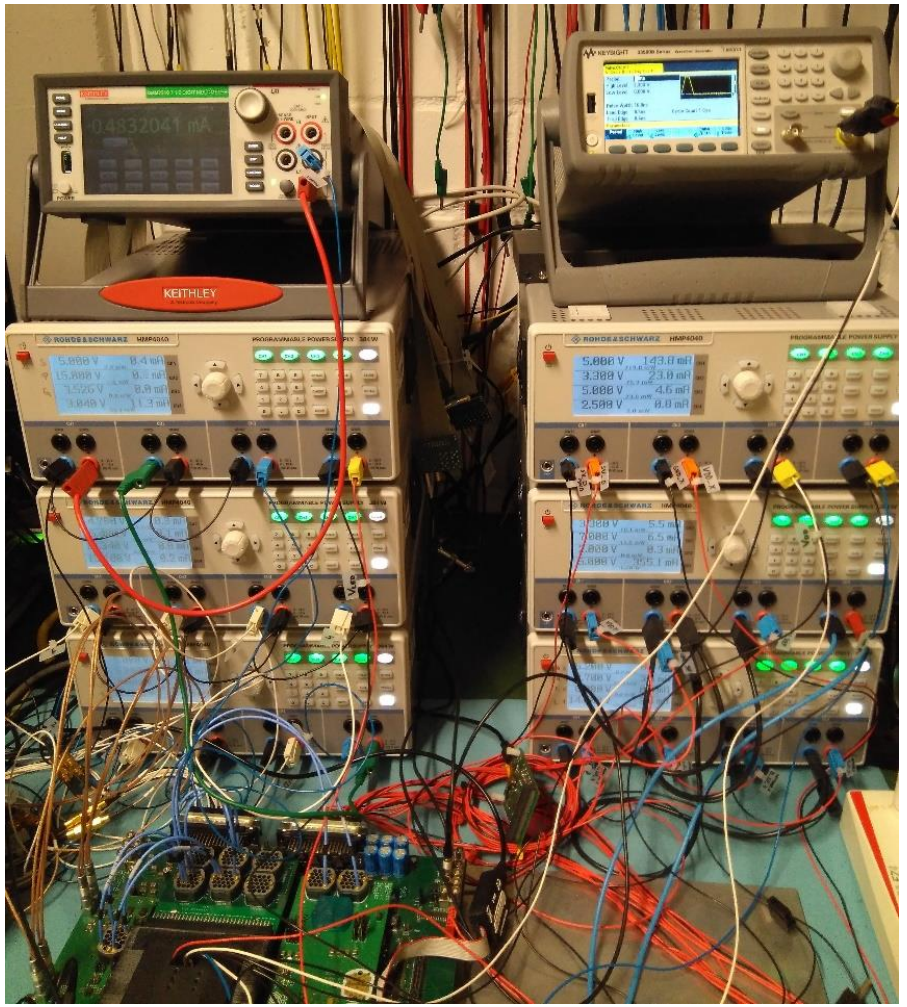


INTRO TO VISRV – THE VISION

Many lab and test setups share a few common aspects:

- A random number of devices...
- connected arbitrarily to some system...
- which combines them in a meaningful manner.
- A multitude of interfaces to communicate and...
- even the same interface may be used in different ways by different devices.
- The tester has to monitor and/or control the devices depending on the test.
- The setup may change at the drop of a hat.

- And someone has to write a software for that...
- again and again for each new setup.



SPIX – Single Pixel measurement setup

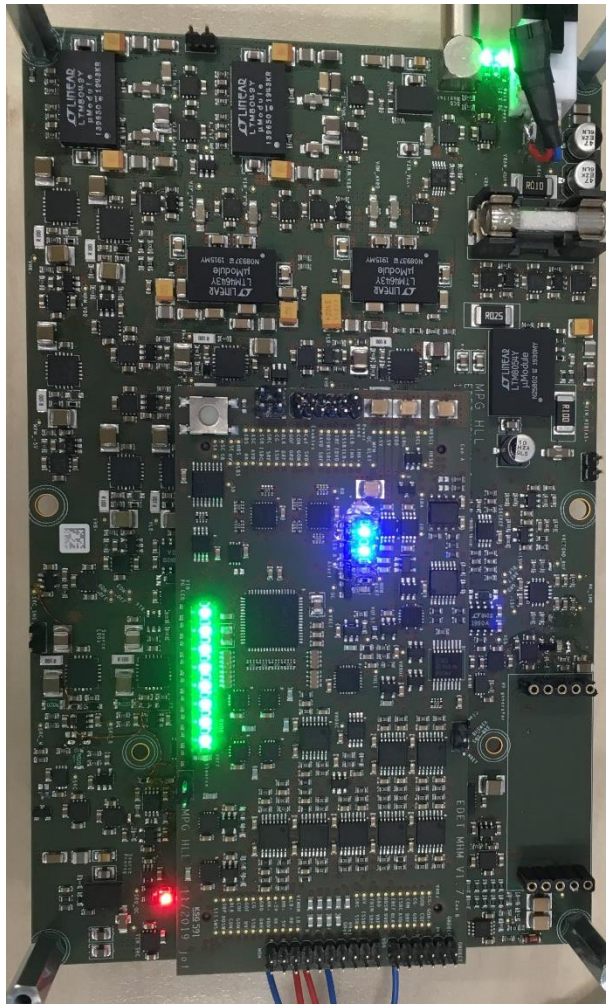
INTRO TO VISRV – THE VISION

The Vision:

„One tool to rule them all,
One tool to find them,
One tool to bring them all,
And in the lab to bind them“
– J.R.R. Tolkien, rough translation

Create a software layer that:

- Abstracts all the device peculiarities.
- Provides a single point of access for the tester.
- Allows to mirror any setup in software.
- Comes with a graphical user interface.
- Can be extended with new features as needed.



MHM & MPM – EDet housekeeping and power conditioning modules

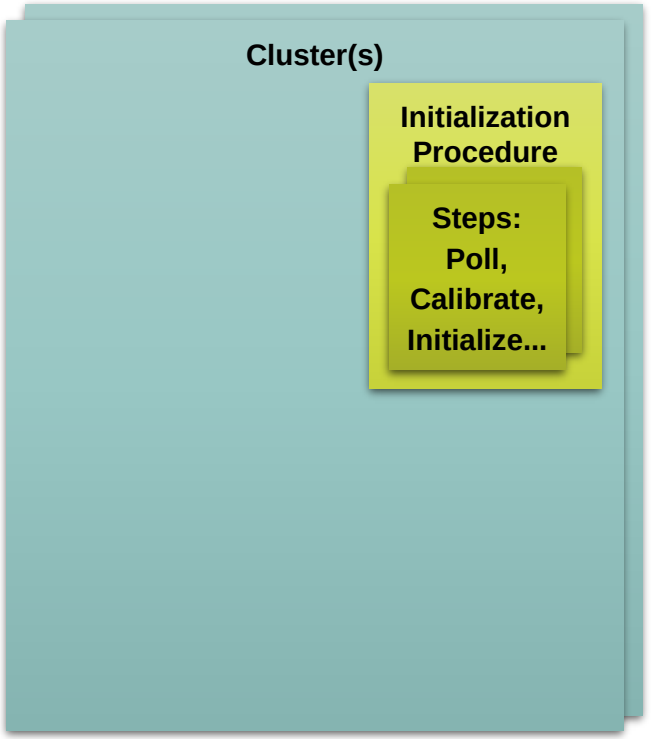
THE VIRTUAL INSTRUMENT SUITE

VI Suite	Tasks	PXD
VI Server (with drivers) and VASIC (for ASICs)	**Communicate with power supplies and ASICs**	FPGA based (LMU power supply, DHE for ASICs)
VI Server: internal, file system	**Store device state, settings**	EPICS server, config DB
VI Server: TCP/IP to Identifiers	**Access point for software, user**	EPICS: network/local to PVs
VI Server: Sequence, Condition	**Process Automation, Controling**	IOCs on lab or control PC
VI Manager, Client and VASIC	**Graphical User Interface**	CS Studio/Phoebus, OPIs
VI Server & Client (filesystem)	**Data archiving**	EPICS server archive (db)
VI Log Viewer	**Data plotting**	CS Studio Archive viewer
VI Manager modules	**System interconnectivity**	IOCs

COMPONENTS OF THE VI SERVER – CLUSTER

Logical group of devices:

- Stores device instances that belong together.
- Defines the initialization procedure.
- Defines the active housekeeping cycle.
- Disconnects all devices if one goes missing.



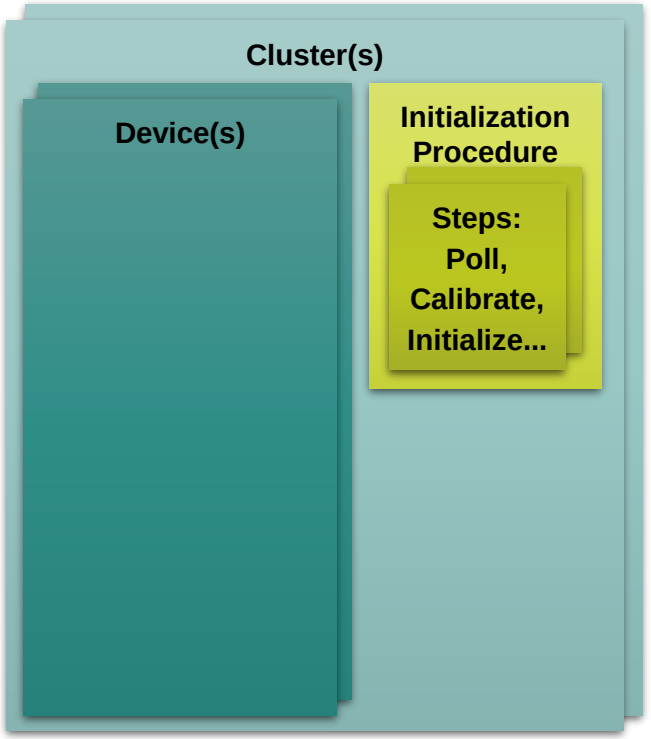
Physical examples:

- A board of multiple IC controlled via a shared I^2C bus.
- Four bench power supplies connected via LAN.

COMPONENTS OF THE VI SERVER – DEVICE

A component accessed via a single interface:

- Stores connection status and sub components.
- Knows the capabilities of the physical device.
- Defines the steps initialize the device.
- Can hold multiple channels if the device supports them.



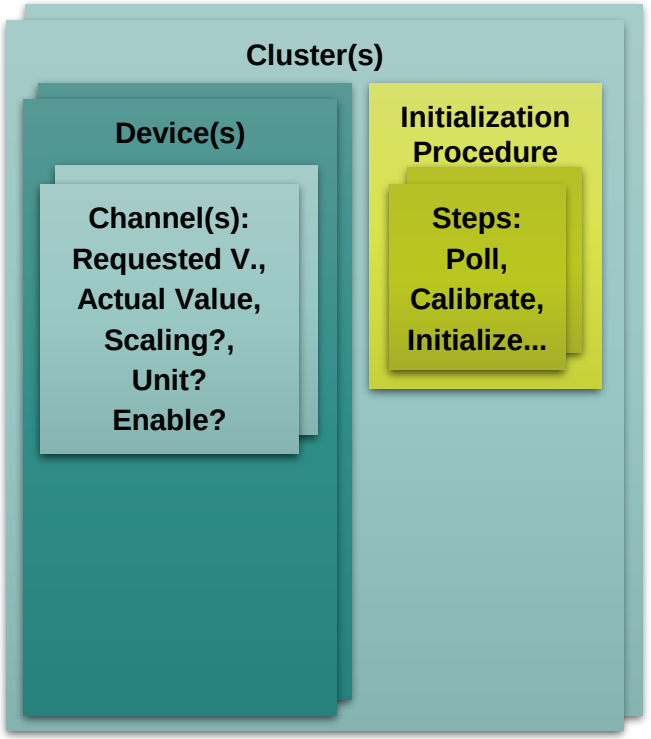
Physical examples:

- A temperature measurement chip, an EEPROM.
- A complex housekeeping and monitoring chip.
- A single bench power supply.

COMPONENTS OF THE VI SERVER – CHANNEL

Holds information about the „moving“ parts of a device:

- Actual value: The value as read back from the device (read-only, updated by device answers).
- Requested value: The value that is supposed to be set in the device (writable by user to trigger changes in the device).
- Available in different complexities: Bare bones, with additional enable/disable option, with scaling from arbitrary to readable units, ...



Physical examples:

- Channel six of a digital-to-analog converter chip
- The voltage setting of a bench power supply.

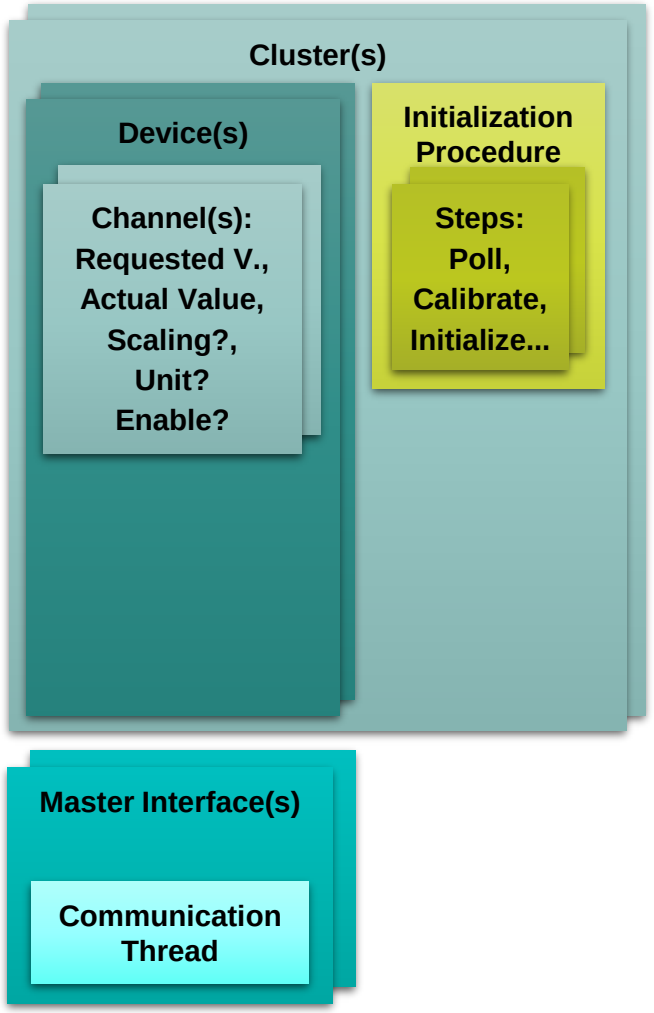
COMPONENTS OF THE VI SERVER – MASTER INTERFACE

Handles communication with drivers:

- Provides its own communication thread.
- Collects communication requests.
- Handles communication with drivers.
- Distributes answers to device classes.
- Assumes and ensures that all communication on the interface is handled sequentially.

Physical examples:

- The I^2C master on the bus.
- A LAN connection to one IP address and port.
- A VISA connection to a device.



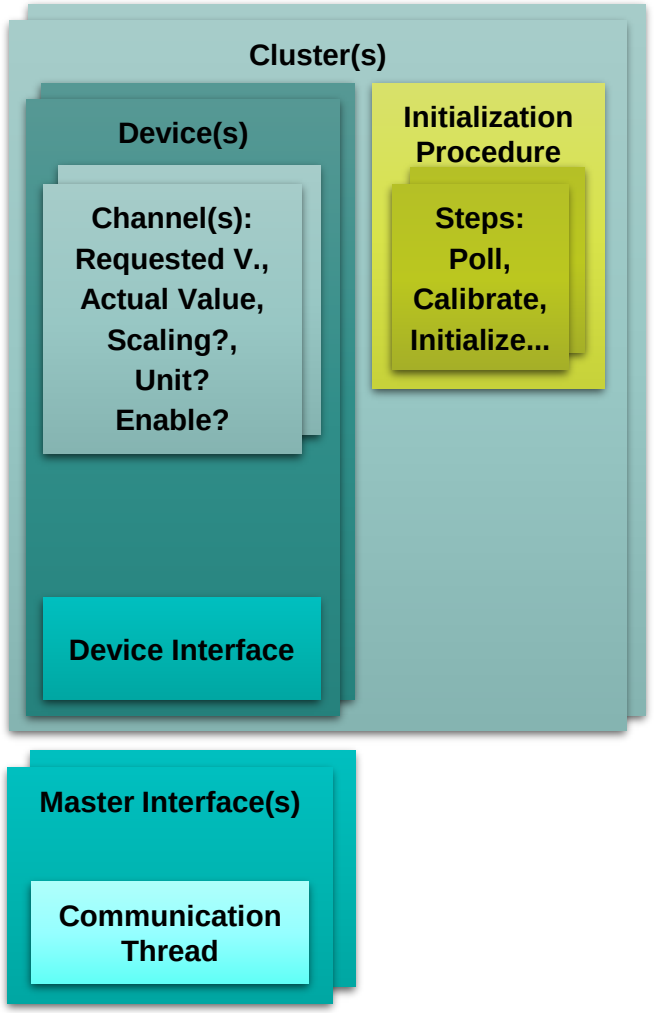
COMPONENTS OF THE VI SERVER – DEVICE INTERFACE

Defines the specific device communication:

- Translates changes in server data to commands sent to the device.
- Translates answers received from the device to changes in server data.
- Output and input are formatted without knowledge of the specific kind of interface used.

Physical examples:

- Counterpart to the logic that translates byte sequences to register actions in an I²C chip.
- Command interpreter.



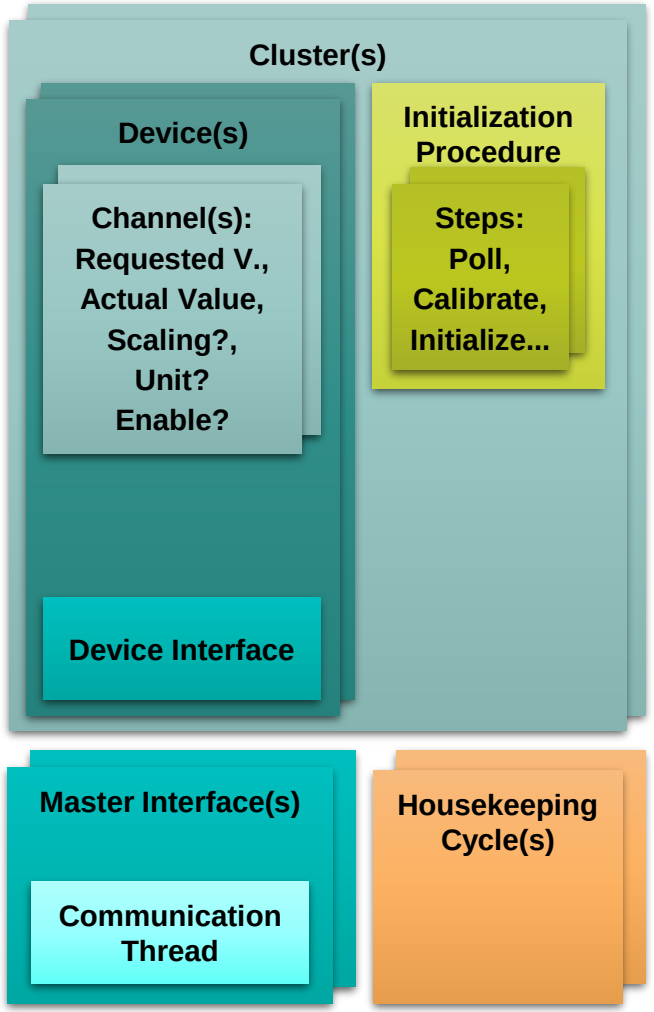
COMPONENTS OF THE VI SERVER – HOUSEKEEPING CYCLE

Set of commands that are needed to update the state of a cluster:

- Contains the order of actions needed to update the data on the server.
- Can be as detailed or broad and as complete or specific as needed.

Physical examples:

- The list of channels that have to be read out for a complete update and the order in which to read them.
- A subset of channels to update for a specific test.



COMPONENTS OF THE VI SERVER –SCHEDULER & BLOCKER

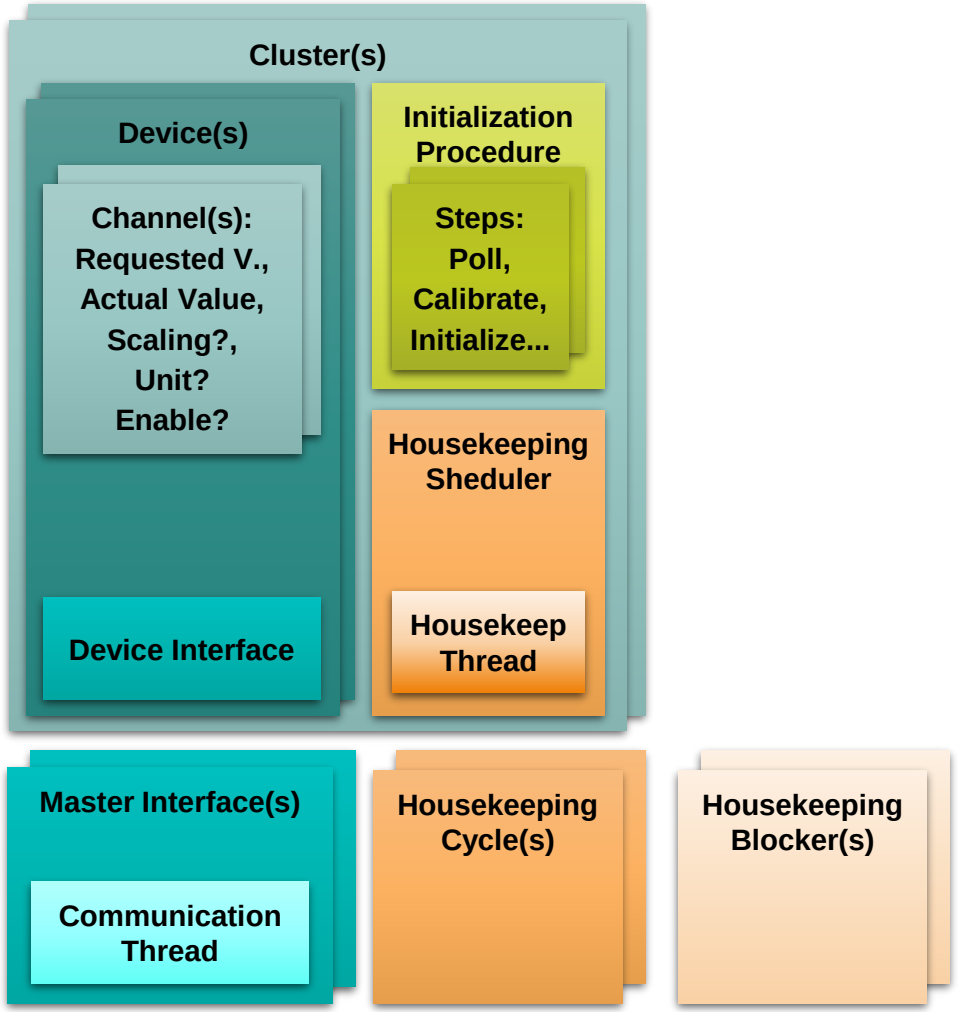
Scheduler decides the next housekeeping cycle and the timing before it is called:

Used to plan complex housekeeping activities, e.g.:

- Update some channels only every 5 seconds because there are more important tasks to do.
- Perform complete update after initialization, then switch to a more focused housekeeping cycle.

Blocker stops the execution of a housekeeping cycle until all blocking conditions are resolved:

Used to prevent clusters that share e.g. a multiplexer for their interface from fighting over the multiplexer‘s state during their parallel updates.



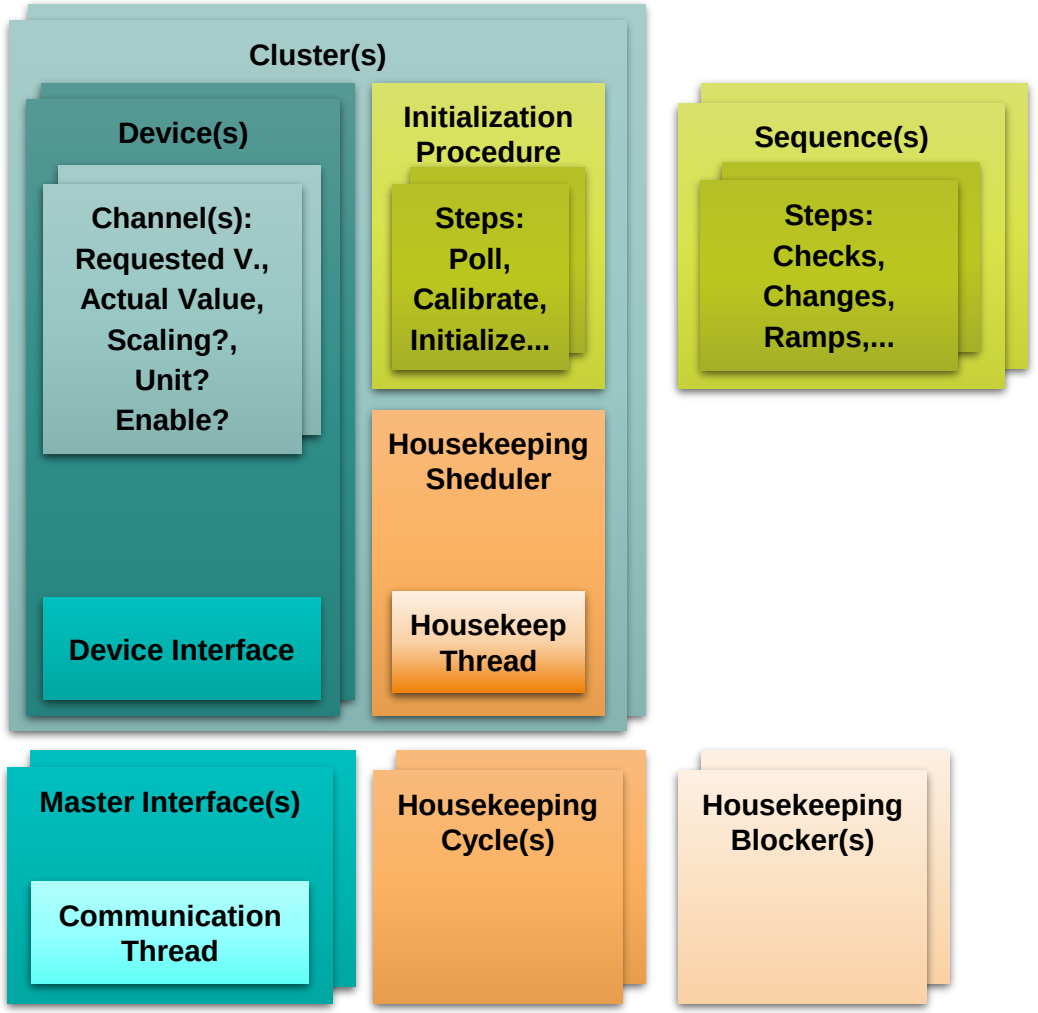
COMPONENTS OF THE VI SERVER – SEQUENCE & STEP

Define sets of instructions that can be triggered on demand:

- A state machine where sequences represent transitions from one state to another.
- Steps can perform a multitude of tasks such as simple checks, changes, ramps, lists of changes.

Physical examples:

- A list of changes required to bring the whole system from OFF to STANDBY state.
- A software emergency shutdown sequence.



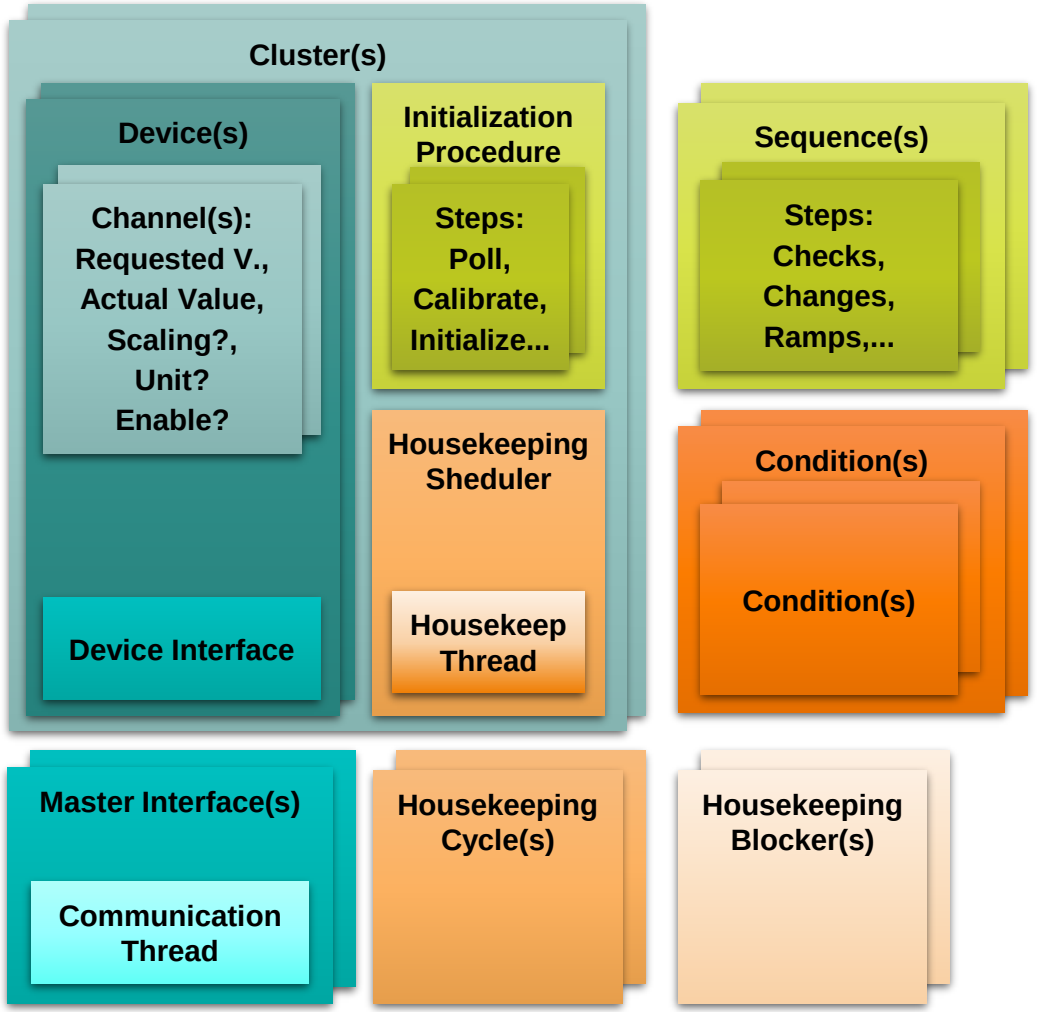
COMPONENTS OF THE VI SERVER – CONDITIONS

Define sets of checks that may trigger an automatic action:

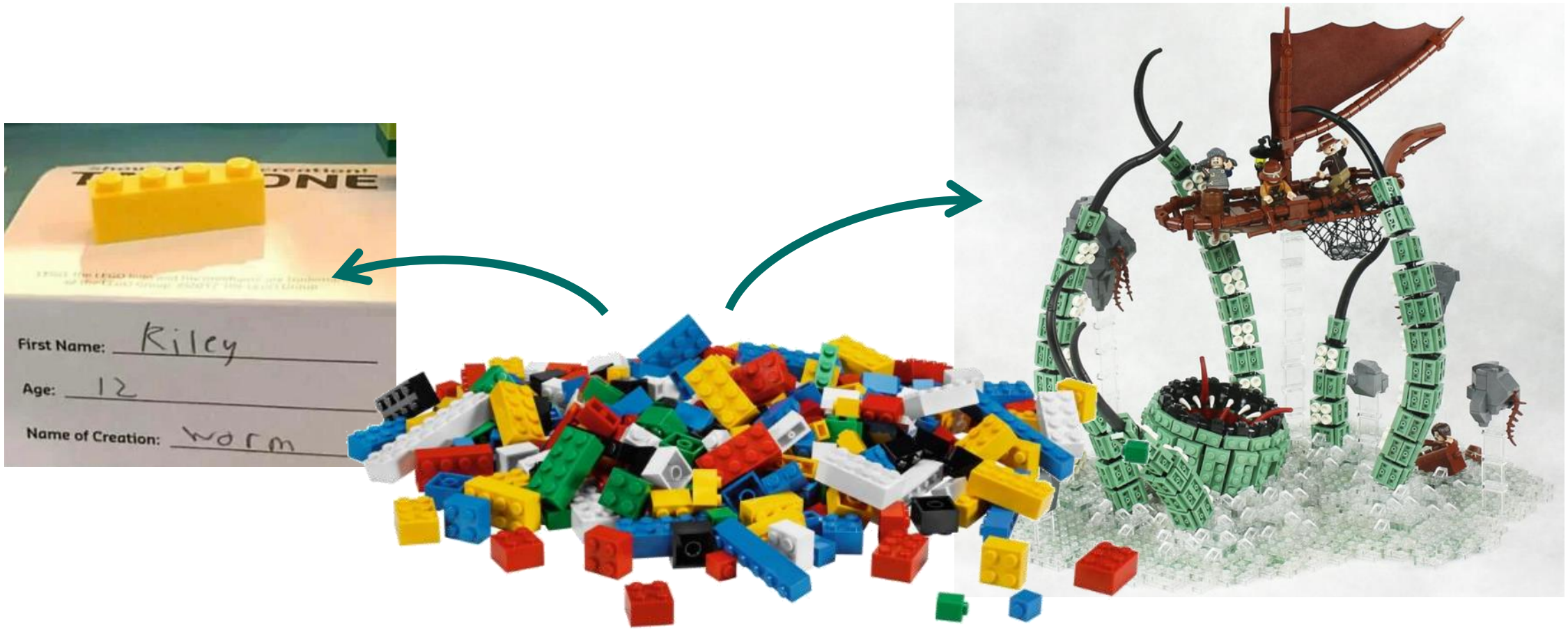
- Check against all data available on the server.
- Complex nesting of checks with all basic logic operations (AND, OR, NOT).
- Latching behavior on demand.
- Available actions to trigger are quite few for now.

Physical examples:

- A list of checks that must hold true while the system is in STANDBY state to ba able to change to PEAK state.
- A set of fault conditions that trigger an emergency shutdown.



COMPONENTS OF THE VI SERVER – IT‘S TIME TO CREATE!



(legally distinct system building blocks)

DEVICES SUPPORTED BY THE VISRV – 50 AND GROWING

Bench Power Supplies:

Rohde & Schwarz 804x & 4040

Source Measure Units:

Keithley 236-238, 2400, 2612

AimTTi PLH & PLH-P

Electronic Load: CSI 371x

Non-volatile Memories:

24xx128, 24AA02x (both I^2C)

Clock Snthesizers:

SI57x, SI5380 (soon™)

GPIO Extenders / Multiplexers:

ADG728-9, PCAxxxx, TCAxxxx

Digital-Analog-Converters:

DACx578, DAC121C08x, MAX538x

Analog-Digital-Converters:

MAX1161x, MAX1164x, LTC230x

Digital Potentiometer: AD528x

Monitoring and Control devices: AMC7812, INA233

Temperature Sensors: TMP10x, TMP42x

More to come as needed by projects.

SHORT-COMINGS AND SPACE FOR FUTURE IMPROVEMENTS

- **Multi-threading / multi-processing:**
  Python GIL prevents real multi-threading.
  Localized data store prevents multi-processing.
- **Secure and usable data archiving:**
  Changes are written to file – okayish.
  Analyzing a week or month of data takes a lot of pre-processing time.
- **No real API to work with the system:**
  Current access to server data either handled by GUI (behind the scenes) or by expert (me) who speaks the servers JSON-jargon.
- **Documentation, Documentation, Document...**
  A large number of classes and functions are documented in code. Also available as HTML page via sphinx and GitLabCI/CD.

- **Introduce additional communication layer:**
  Move communication out of the server tasks.
  Optimize data handling/communication.
- **In new process linked to the comms layer:**
  Reduce workload of server for archiving.
  Connect to easier or optimized data model e.g. data base.
- **Improve User Experience via a new API:**
  Specifics and scope has to be defined.
  Might come as part of the new comms layer (e.g. REST API) or its own framework.
- **Write documentation for Users:**
  Describe usual usage scenarios, document features and how to use them, give examples and tips.

IT‘S TIME TO WAKE UP – YES, I AM DONE FOR THE MOMENT

Questions…?

Suggestions…?

Solutions…?

Wishes…?

Expectations…?

Hopes and dreams…?

For later questions please contact:

Max Planck Society – Semiconductor Laboratory

Martin Hensel

Otto-Hahn-Ring 6
81739 München

Tel.: +49 (0)89 839400-0 or -93
Fax: +49 (0)89 839400-11

E-Mail: hensel@hll.mpg.de
Internet: www.hll.mpg.de