Constellation

A flexible control and data acquisition framework for test beam environments

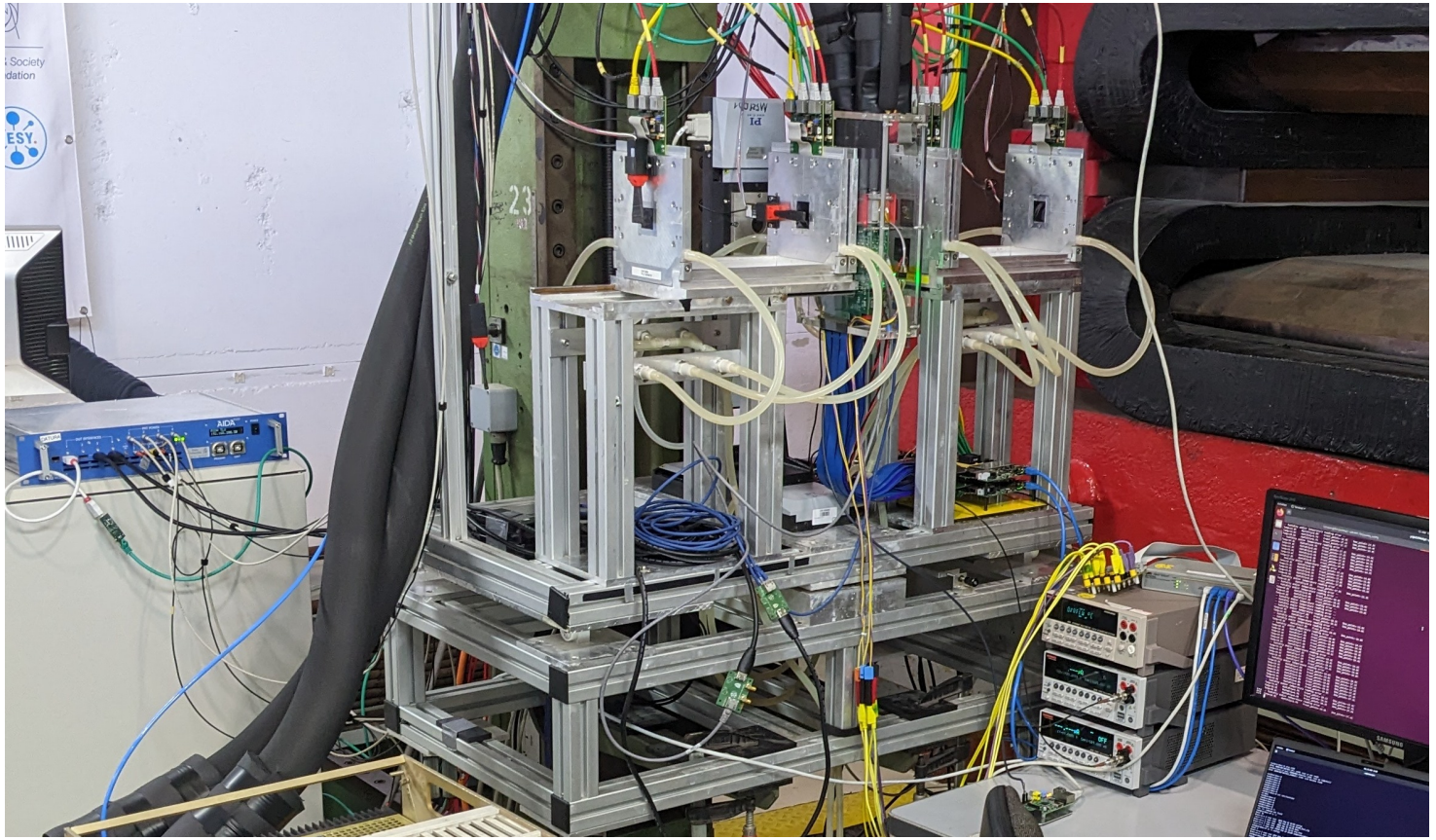
Stephan Lachnit for the EDDA collaboration
BTTB13, Valencia

HELMHOLTZ

Setting the Stage

An ordinary test beam day at DESY

- Device Under Test
- Beam telescope
- AIDA2020 TLU
- Keithley 2410 SMU
- Data stored in EUDAQ2 format
- Nitrogen cooling with temperature monitoring
- **Constellation** as control framework



Example: Setup for the dSiPM test beam at DESY

New

Starting the Setup

No more IP copying

Let’s start some satellites:

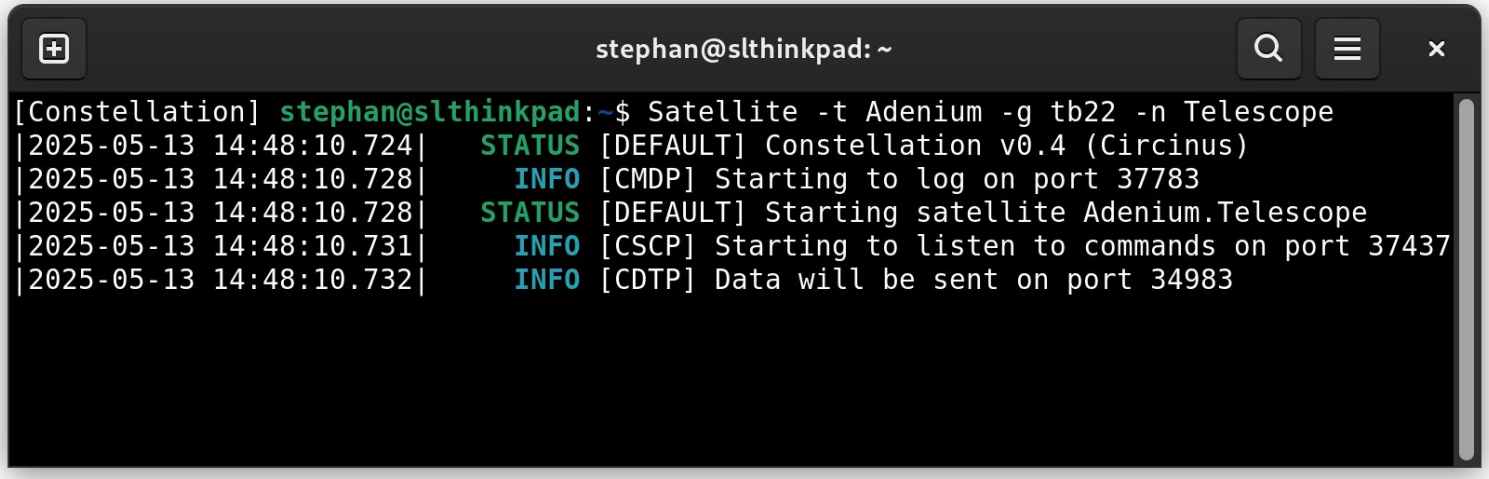
```
Satellite -t Caribou -g tb22 -n dSiPM_1
Satellite -t Adenium -g tb22 -n Telescope
```

Satellite type

Group name

Satellite name

No need to specify any IP addres!



Configuration

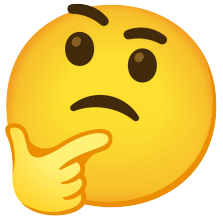
TOML just like you know and love from Corryvreckan or Allpix Squared

Open

dSiPM_tb.toml
~/

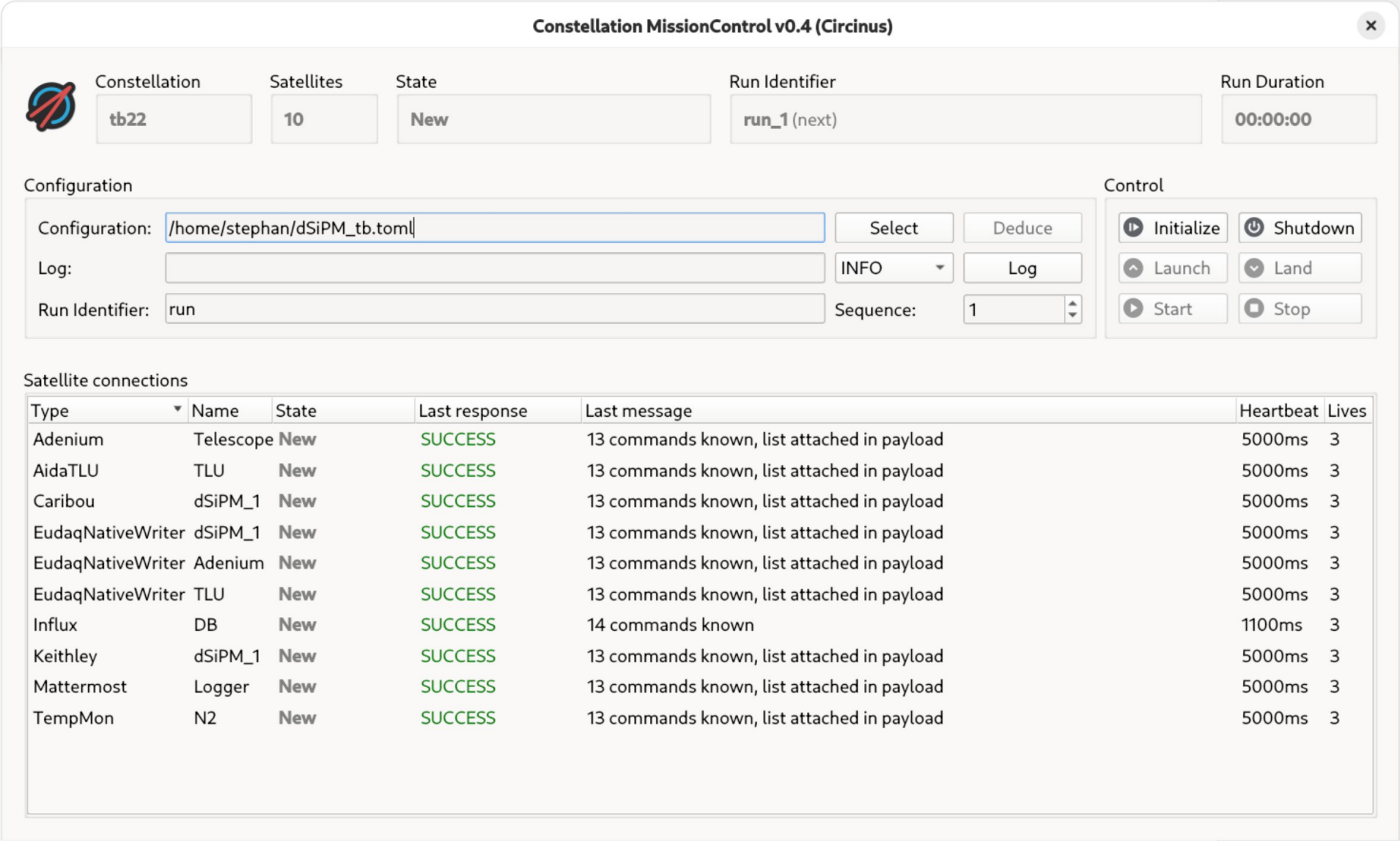
```toml
[satellites.Caribou.dSiPN_1]
_require_launching_after = ["AidaTLU.TLU"]
_require_stopping_after = ["AidaTLU.TLU"]

type = "dSiPM"
use_tlu = true

[satellites.Keithley.dSiPM_1]
device = "2410"
port = "/dev/ttyUSB0"
terminal = "front"
voltage = -2.4
voltage_step = 0.1
settle_time = 0.2
ovp = 20
compliance = 40e-6

[satellites.Adenium.Telescope]
_require_stopping_after = ["AidaTLU.TLU"]

[satellites.AidaTLU.TLU]
_require_starting_after = ["Caribou.dSiPM_1", "Caribou.dSiPM_2", "Adenium.Telescope"]

ConnectionFile = "file:///home/teleuser/constellation/cxx/satellites/AidaTLU/misc/aida_tlu_connection.xml"
DeviceName = "aida_tlu.controlhub"

# Set CONFCLOCK to 1 to configure clock, which is necessary after a power cycle
CONFCLOCK = true
# Path to clock file
CLOCK_CFG_FILE = "/home/teleuser/telepix/TLU_Si5345-RevB-125MHz-3.txt"
```

What happens if you made a typo?



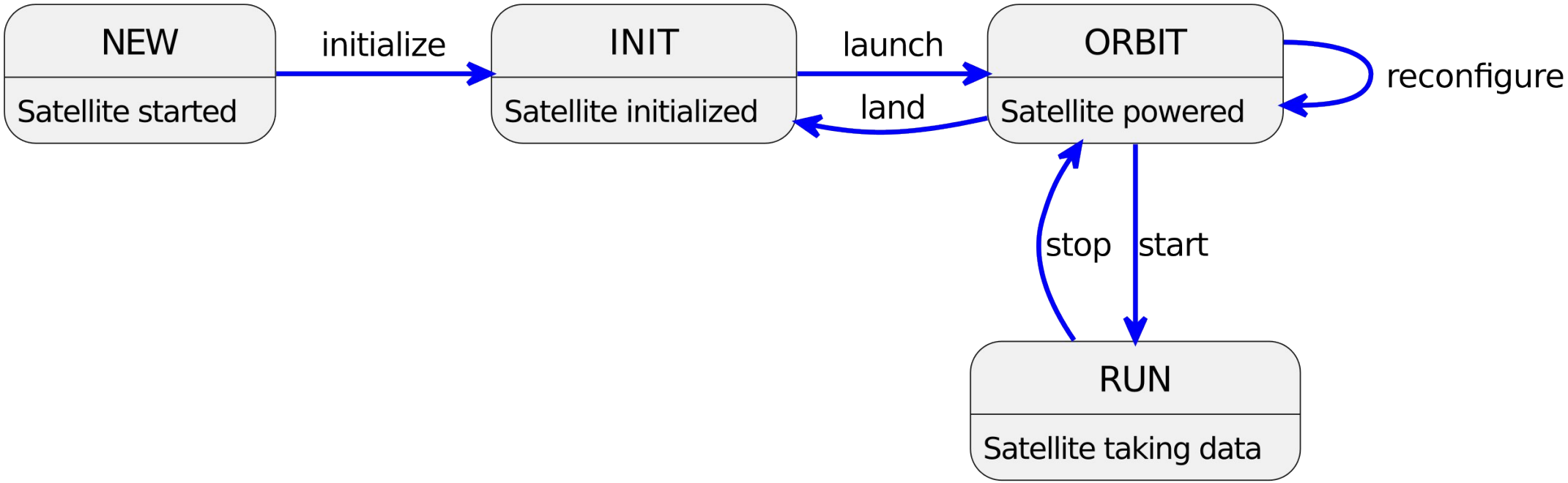
MissionControl GUI

Seems familiar...



Finite State Machine

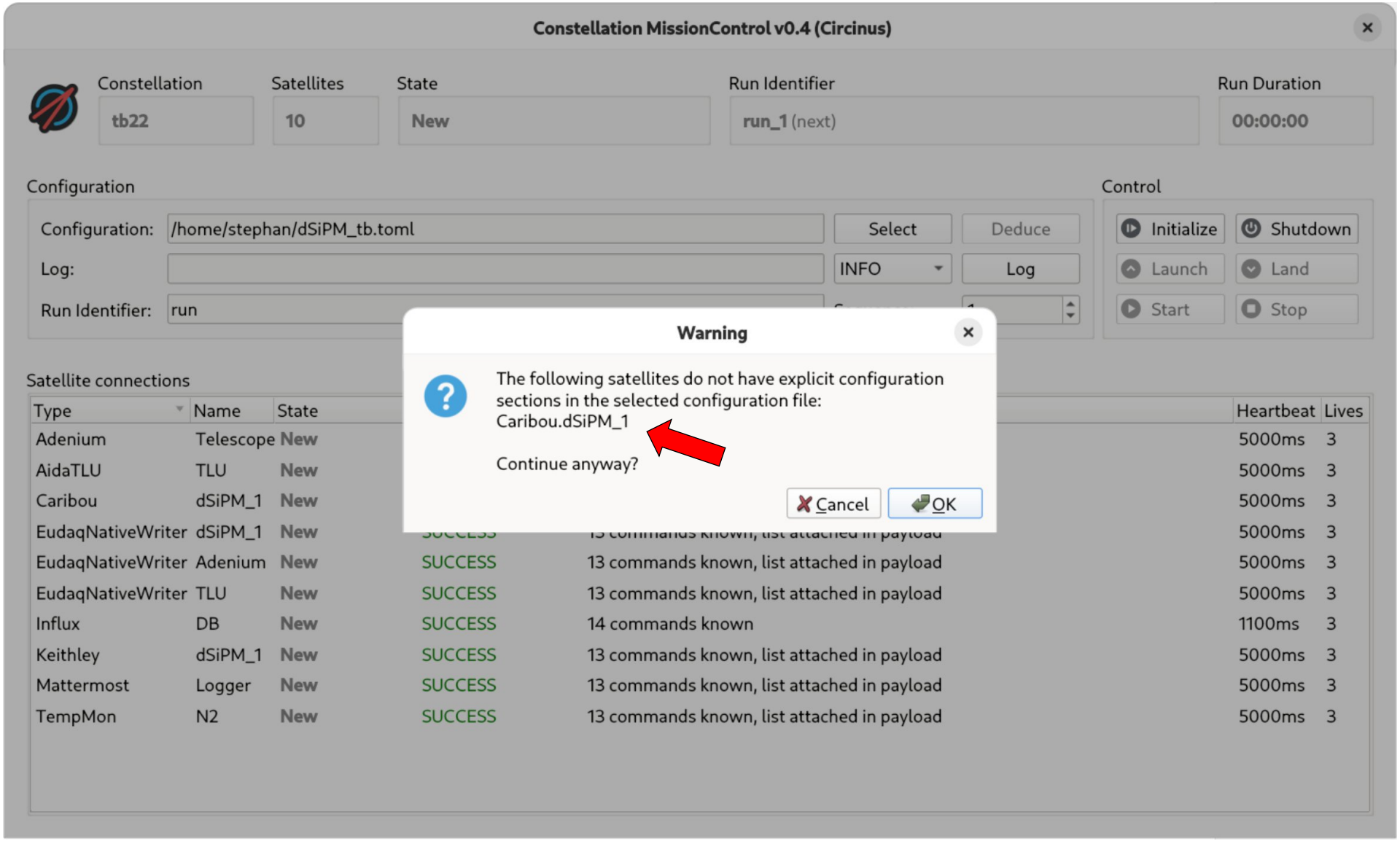
Keep your instrument in a defined state



Additional **SAFE** and **ERROR** states for error handling

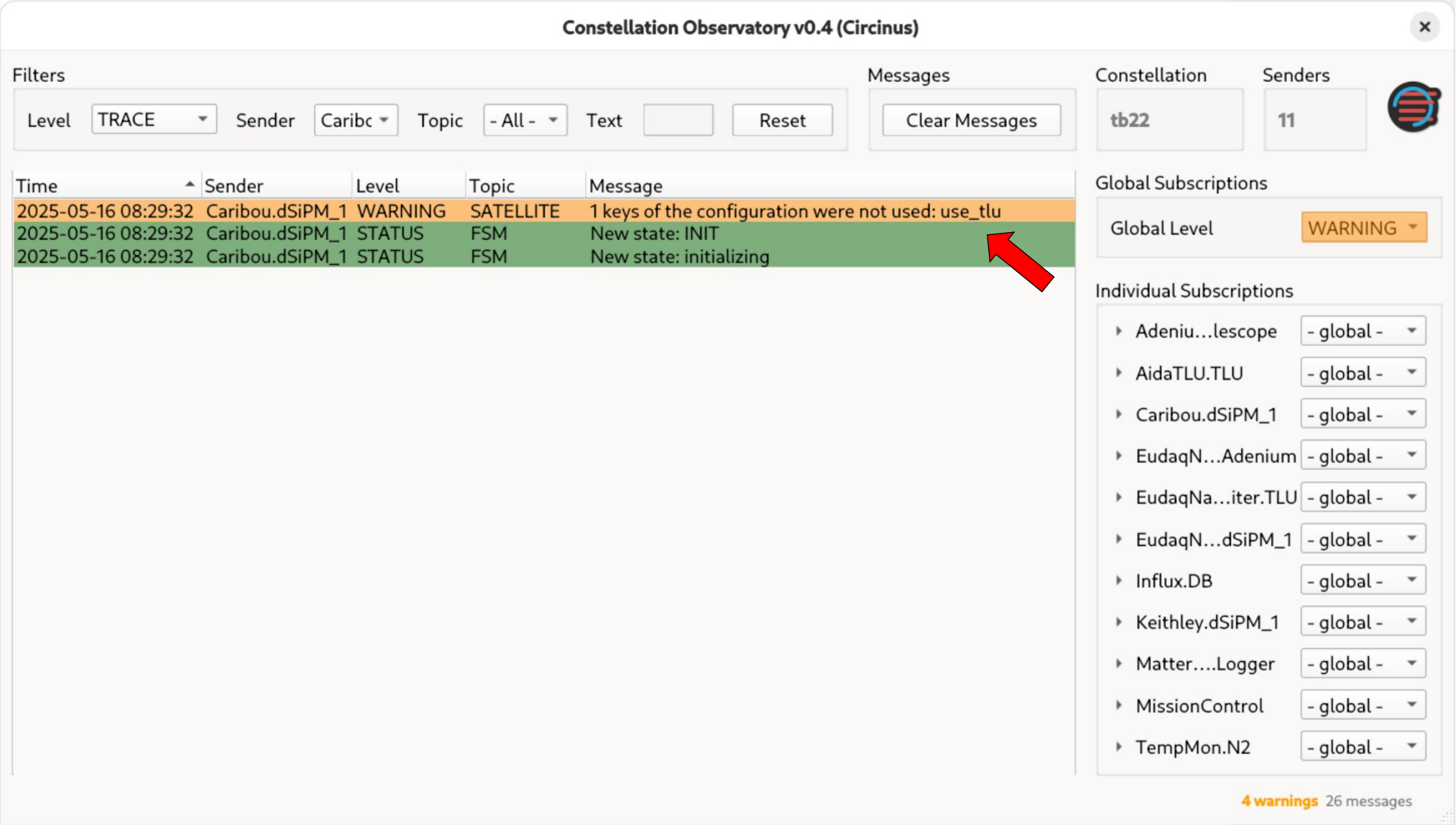
MissionControl GUI

Typo #1



Observatory GUI

Typo #2



Starting a Run

But why don’t I get any triggers...?

Constellation MissionControl v0.4 (Circinus)

Constellation	Satellites	State	Run Identifier	Run Duration
tb22	10	Running	run_1	00:00:08

Configuration

Configuration: /home/stephan/dSiPM_tb.toml — Select — Deduce

Log: — INFO — Log

Run Identifier: run — Sequence: 1

Control

Initialize — Shutdown

Launch — Land

Start — Stop

Satellite connections

Type	Name	State	Last response	Last message	Heartbeat	Lives
Adenium	Telescope	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
AidaTLU	TLU	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
Caribou	dSiPM_1	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
EudaqNativeWriter	dSiPM_1	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
EudaqNativeWriter	Adenium	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
EudaqNativeWriter	TLU	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
Influx	DB	Running	SUCCESS	transitioning	1100ms	3
Keithley	dSiPM_1	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
Mattermost	Logger	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3
TempMon	N2	Running	SUCCESS	Satellite started run run_1 successfully	5000ms	3

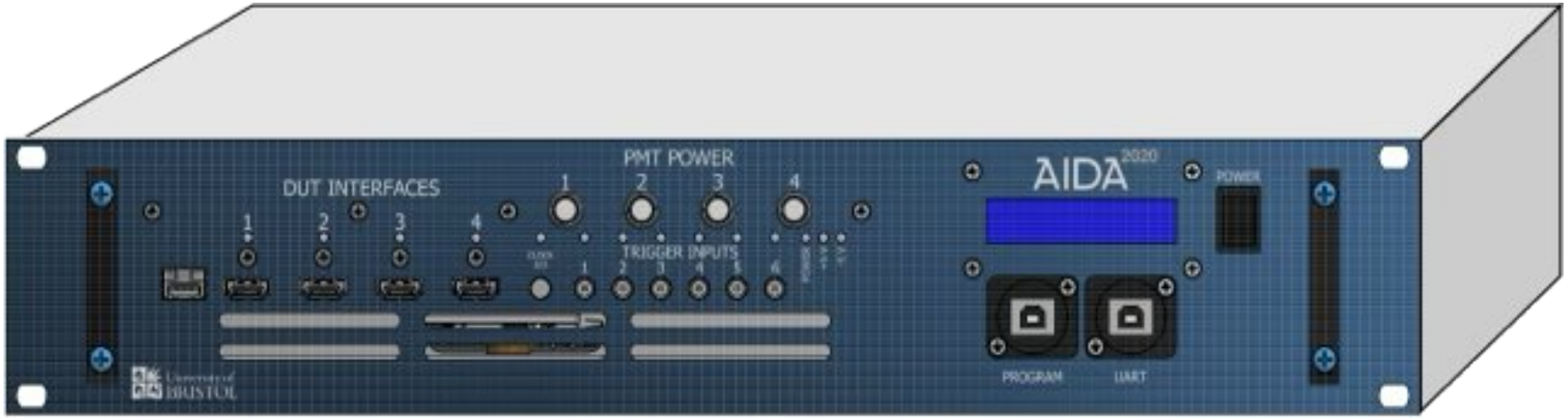
Debugging the TLU Configuration

Mistakes happen from time to time

- Why am I not getting any triggers?
  - Do I get triggers on the inputs?
  - Do my coincidence work?
  - Do my vetos work?

Would be nice to see:

- Triggers for each input
- Pre/post-veto triggers
- Trigger rate over time



Online Monitoring with Grafana

Unsupervised Remote Operation

Surely nothing will go wrong

- Leave it running over the night... Hopefully nothing goes wrong!

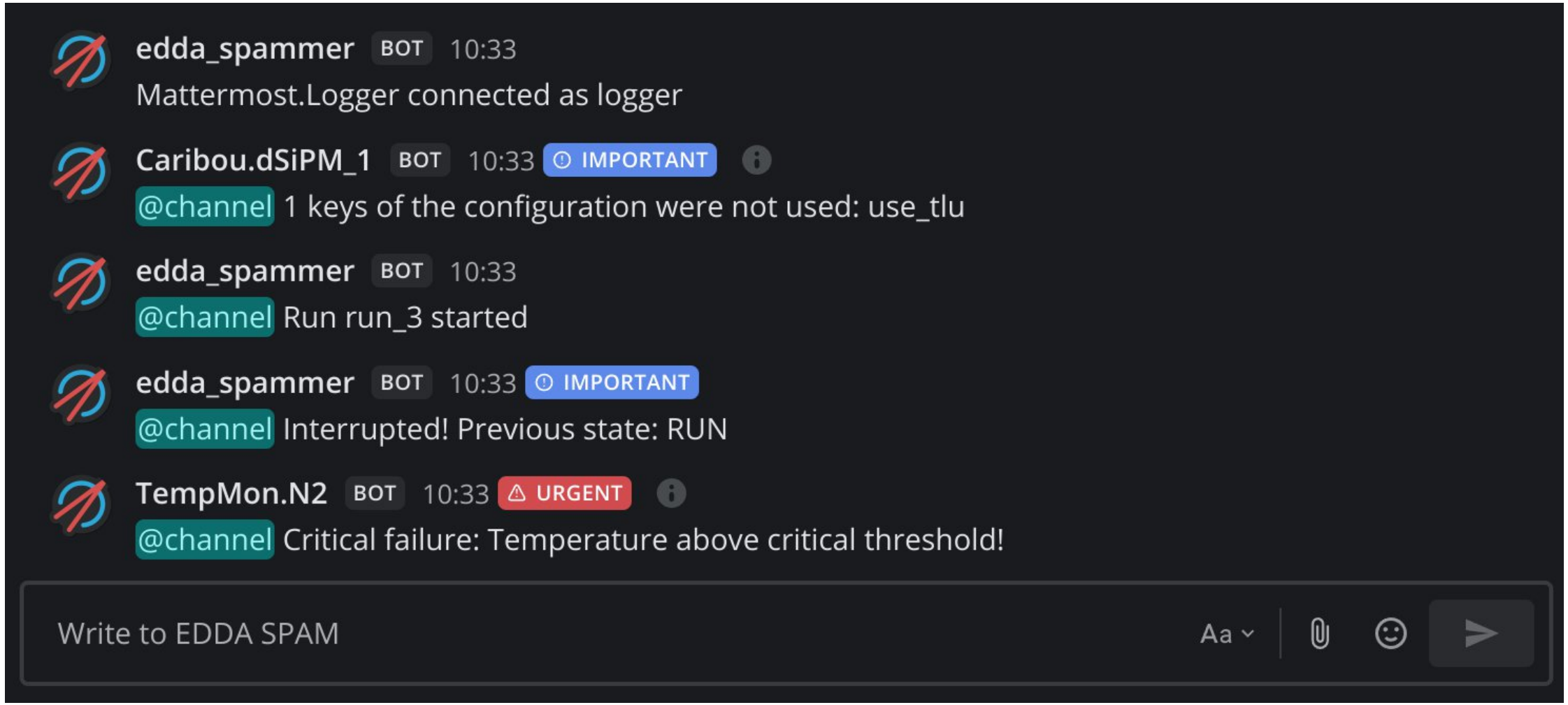
Unsupervised Remote Operation

Surely nothing will go wrong

- Leave it running over the night...
- Phone rings in the middle of the night...

Unsupervised Remote Operation

Surely nothing will go wrong



Unsupervised Remote Operation

Surely nothing will go wrong

- Leave it running over the night...
- Phone rings in the middle of the night...
- Mattermost notification:

  temperature monitoring satellite went to **ERROR** state due to exceeding the temperature limit

Unsupervised Remote Operation

Surely nothing will go wrong

- Leave it running over the night...
- Phone rings in the middle of the night...
- Mattermost notification:

  temperature monitoring satellite went to **ERROR** state due to exceeding the temperature limit
- Do I need to be worried about the high-voltage still being on?

Unsupervised Remote Operation

Surely nothing will go wrong

Constellation MissionControl v0.4 (Circinus)

Constellation	Satellites	State	Run Identifier	Run Duration
tb22	10	Safe Mode ≋	run_4 (next)	00:00:09

Configuration

Configuration: /home/stephan/dSiPM_tb.toml — Select — Deduce

Log: — INFO — Log

Run Identifier: run — Sequence: 4

Control

Initialize — Shutdown

Launch — Land

Start — Stop

Satellite connections

Type	Name	State	Last response	Last message	Heartbeat	Lives
Adenium	Telescope	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
AidaTLU	TLU	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
Caribou	dSiPM_1	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
EudaqNativeWriter	dSiPM_1	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
EudaqNativeWriter	Adenium	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
EudaqNativeWriter	TLU	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
Influx	DB	Safe Mode	SUCCESS	transitioning	1100ms	3
Keithley	dSiPM_1	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
Mattermost	Logger	Safe Mode	SUCCESS	Interrupting satellite operation: TempMon.N2 reports state ERROR	5000ms	3
TempMon	N2	Error	SUCCESS	Critical failure: Temperature above critical threshold!	5000ms	3

Unsupervised Remote Operation

Surely nothing will go wrong

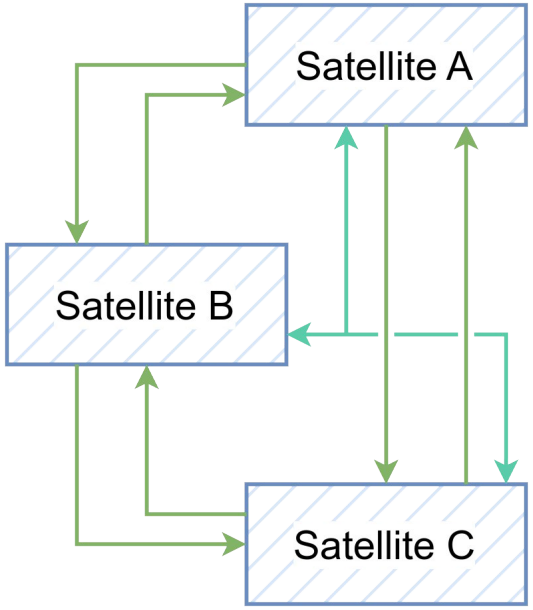
- Leave it running over the night...
- Phone rings in the middle of the night...
- Mattermost notification:

  temperature monitoring satellite went to **ERROR** state due to exceeding the temperature limit
- Do I need to be worried about the voltage still being on?
- **No**: Constellation satellites automatically go to **SAFE** state when any other satellite is in **ERROR**
  - This works autonomously in each satellite, no central points of failure
  - Keithley satellite ramps down voltage when going to **SAFE** state

Autonomy

The heart of the system

- Satellites emit regular “**heartbeats**”
- Heartbeats contain the current state of the satellite
- Each satellite connects to the heartbeats of the other satellites
- When another satellite sends a heartbeat with the **ERROR** state, satellites go to **SAFE** state
- When a satellite stops sending heartbeats (e.g. due to a crash), satellites go to **SAFE** state
- Heartbeats can also be used to monitor the states of the satellites with a stateless software

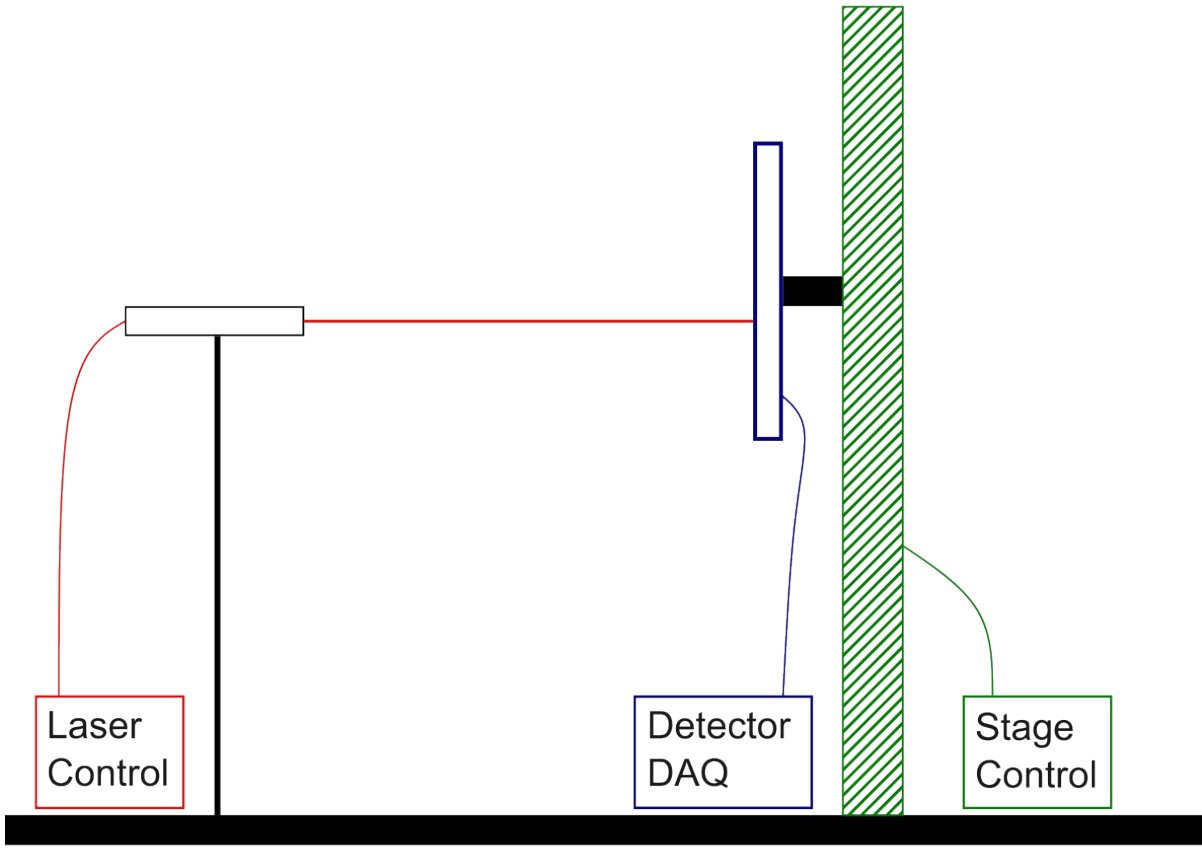


CHP	Heartbeating
CSCP	Control
CMDP	Monitoring
CDTP	Data Transfer
CHIRP	Discovery

Scope of Constellation

Not just test beam

- Constellation not specific to test beam
- Many system are useful in lab measurements:
  - Synchronous operation
  - Configuration
  - Monitoring
  - Autonomy
- Also works for small experiements
  - Interest from the MADMAX experiment

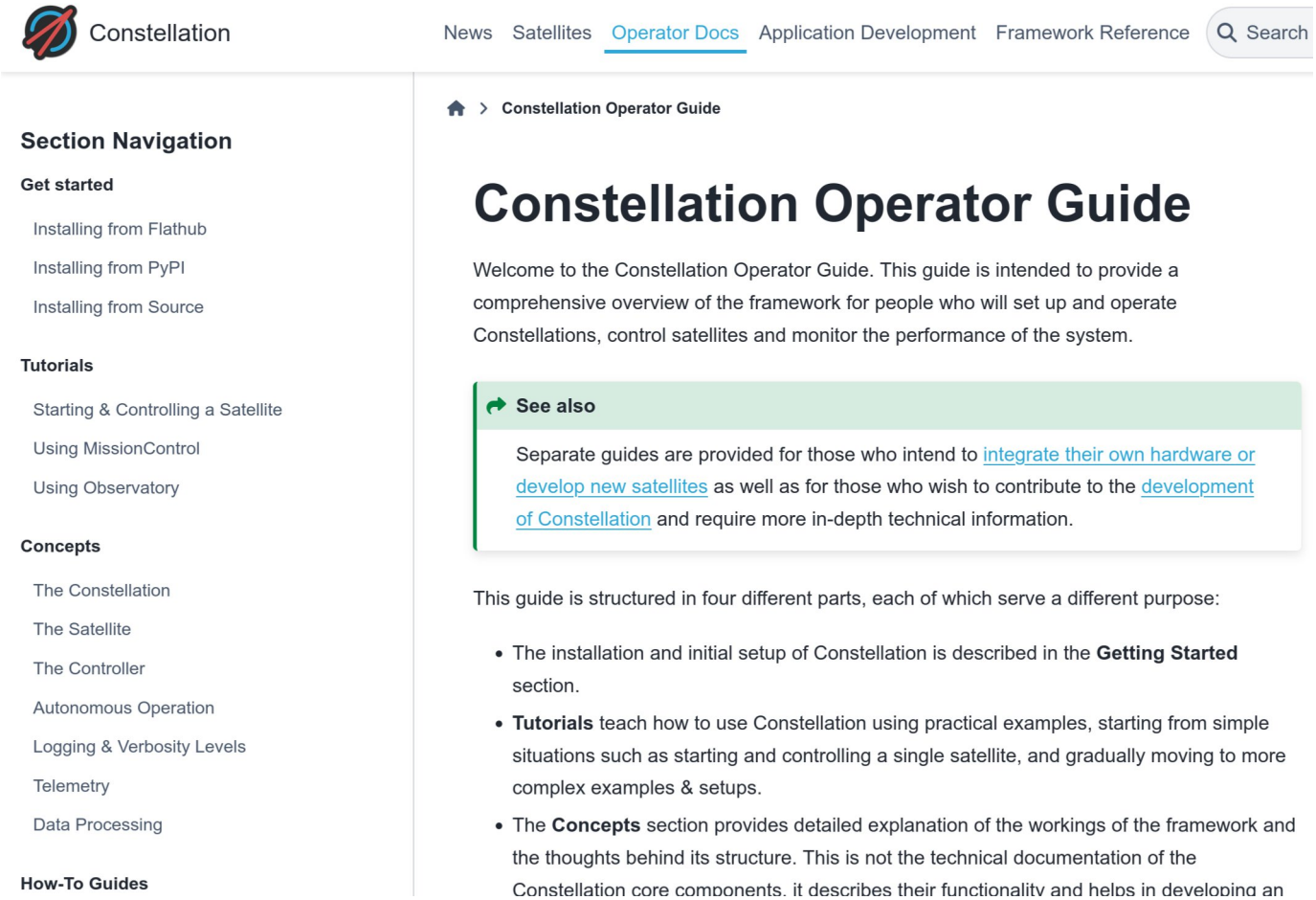


What Happened Since Last Year?

Last years BTTB talk

Progress towards production use in test beam

- Four releases, available on PyPI and Flathub
- Graphical user interfaces for controlling and logging
- Mattermost and Grafana integration
- New satellites, e.g. for Keithley 2410
- More documentation, including tutorials
- First production test beam @ MAMI



Constellation Tutorial @ BTTB

Learn how to use Constellation

- Learn how to setup and use Constellation
- Debugging configuration issues using MissionControl and Observatory
- Implement a temperature monitor in Constellation using Python (we provide the sensors)

- Please install Constelation beforehand using the <u>instructions on indico</u>

Summary & Outlook

Start using Constellation

- Constellation is a control and data acquisition framework tailored towards test beam and lab setups
- Provides simple interfaces for control, configuration, logging, monitoring and data transmission
- Production-ready for most use-cases

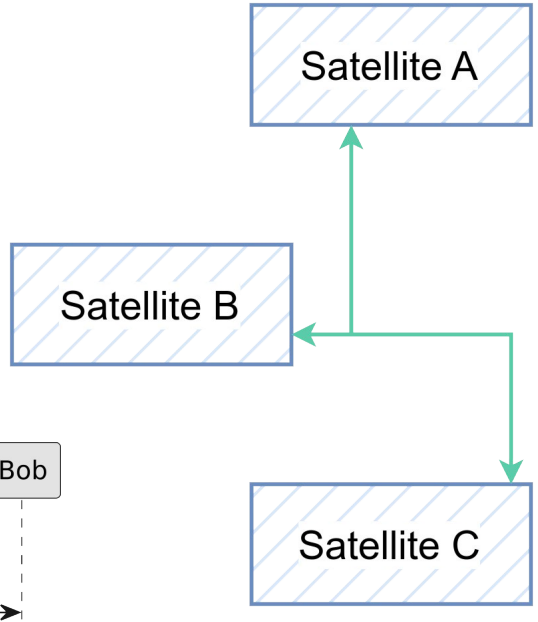
- Planned features for this year:
  - Measurement scans
  - Increased data rate (>10 Gbps)
  - Improved network discovery

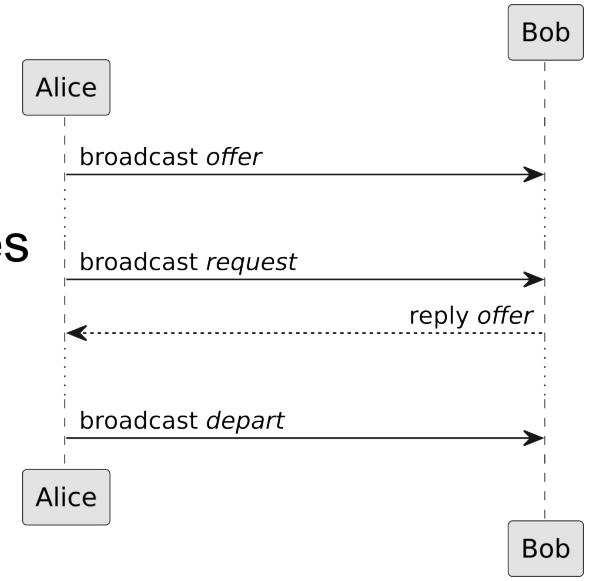
https://constellation.pages.desy.de/

Backup

Network Discovery

- Satellites in the local network are discovered automatically
- Advantages:
  - No to assign fixed IP addresses
  - Simpler setup procedure
  - Allows for autonomy in the system
- Uses UDP broadcasts:
  - OFFER for provided services
  - REQUEST to search for services

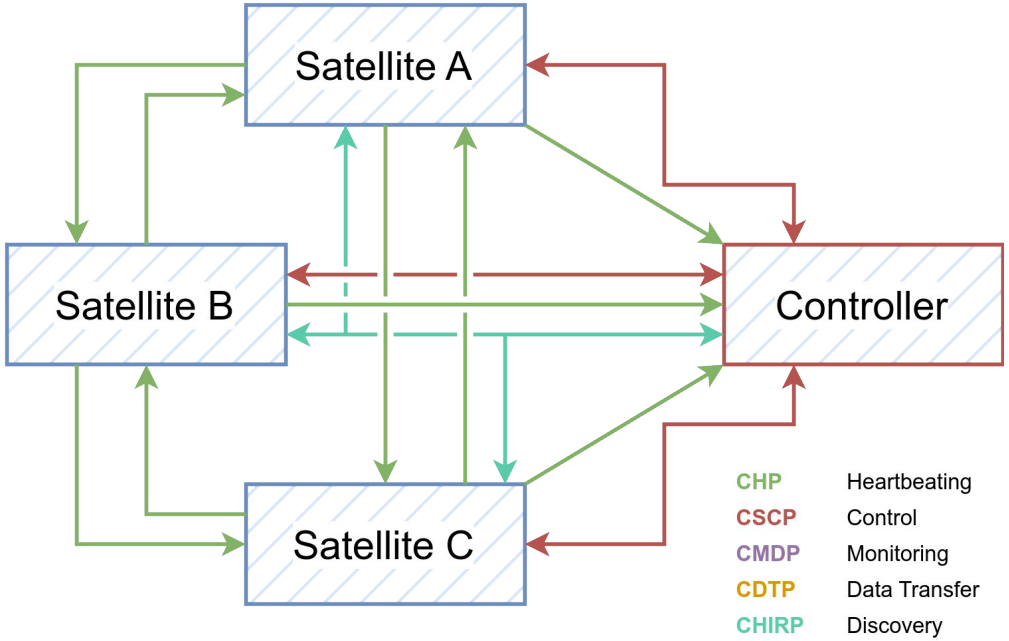




CHP	Heartbeating
CSCP	Control
CMDP	Monitoring
CDTP	Data Transfer
CHIRP	Discovery

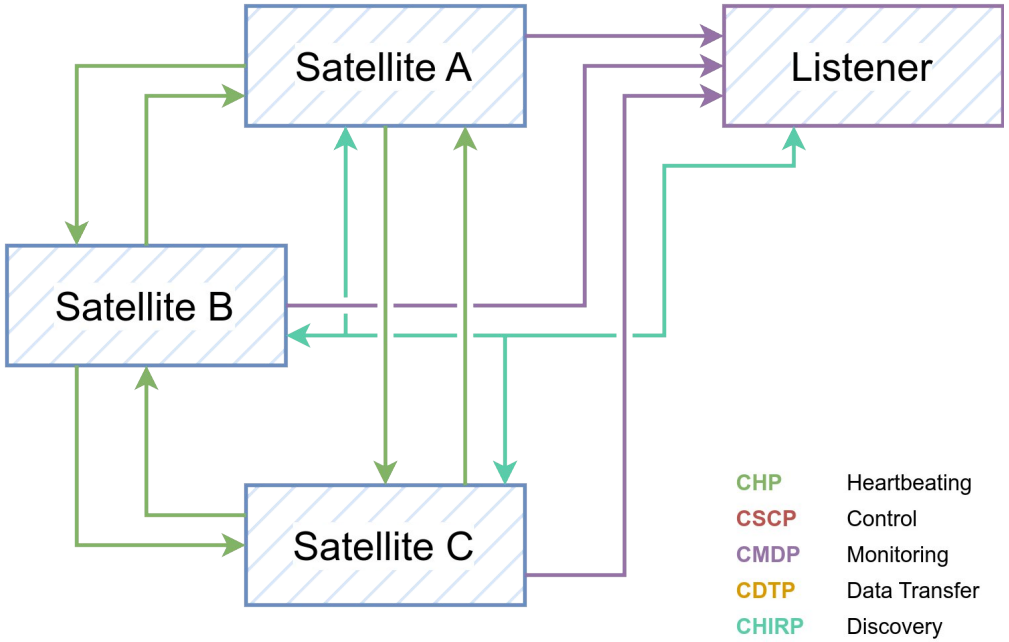
Control

- A controller joins the network:
  - Uses network discovery to find satellites
  - Connects to receive heartbeats
  - Uses heartbeats to monitor the state
  - Sends commands to a single or all satellites
- Client-server pattern for commands
- Controllers are stateless
- Custom commands are possible
  e.g. read register values



Monitoring

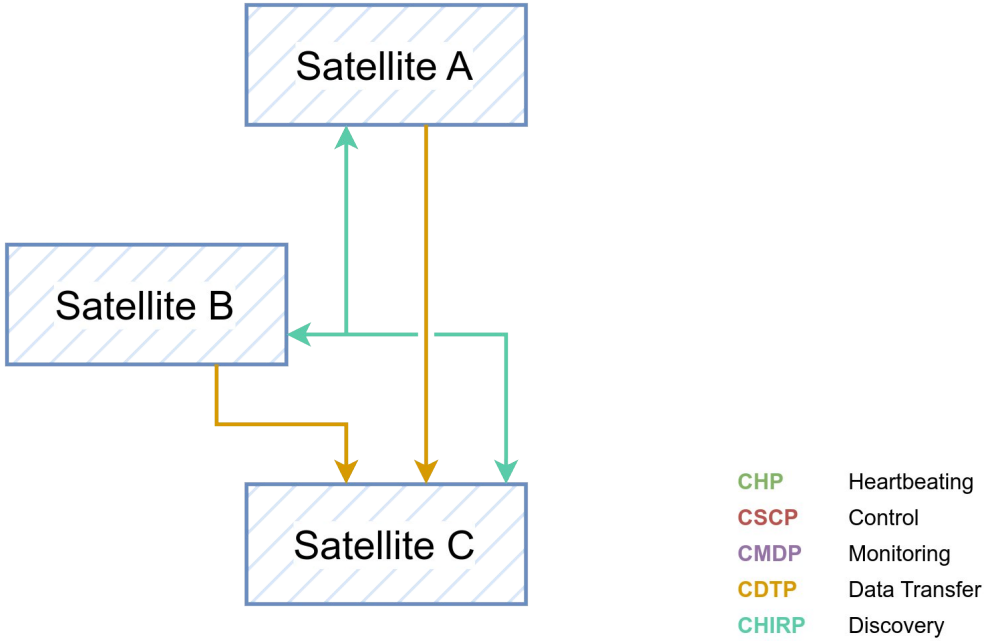
- Satellites can publish telemetry and log messages
- A monitoring listener joins the network:
  - Uses network discovery to find satellites
  - Subscribes to relevant log messages or telemetry
- Publish-subscribe pattern
- Only subscribed monitoring data is actually transmitted
- Simple integration into third-party software like Grafana or Mattermost



Data Transmission

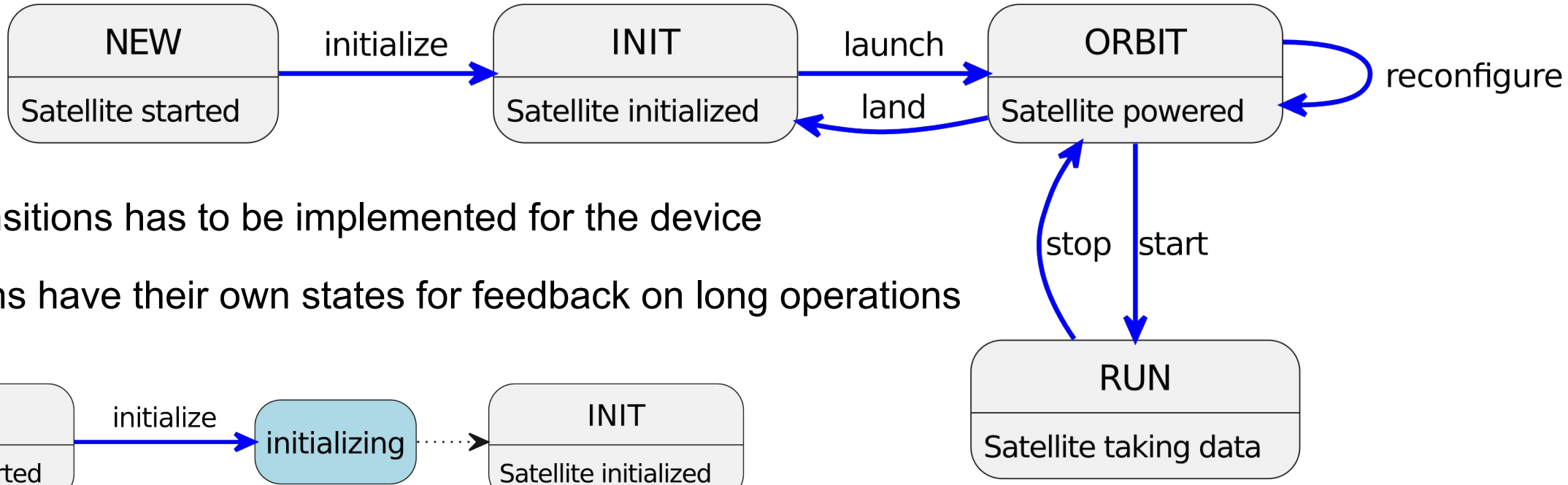
- Sending data over the network can be useful:
  - Embedded system (e.g. Caribou)
  - Store all data in one place
  - Flexible Support for file format
    - EUDAQ2
    - HDF5

Caribou DAQ system

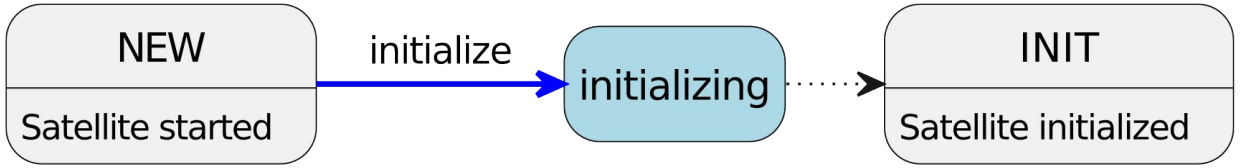


Satellite Finite State Machine

- Device is always in well-defined state



- Each transitions has to be implemented for the device
- Transitions have their own states for feedback on long operations

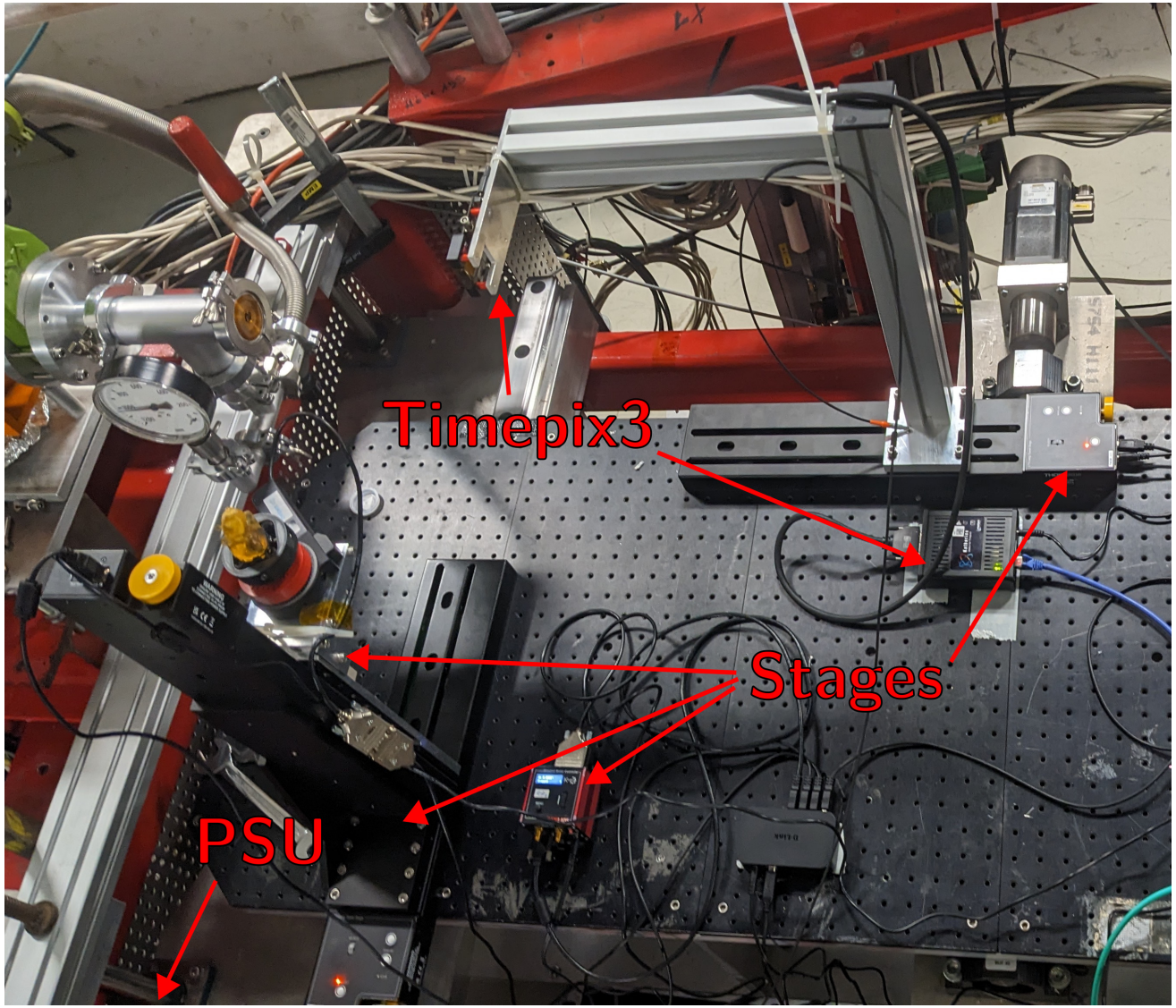


- Additional **SAFE** and **ERROR** states for error handling

electronCT Test Beam

First Production Use @ MAMI

- Timepix3
- Keithley 2410 SMU
- Custom 4D stage
- HDF5 for data storage



H2M Test Beam

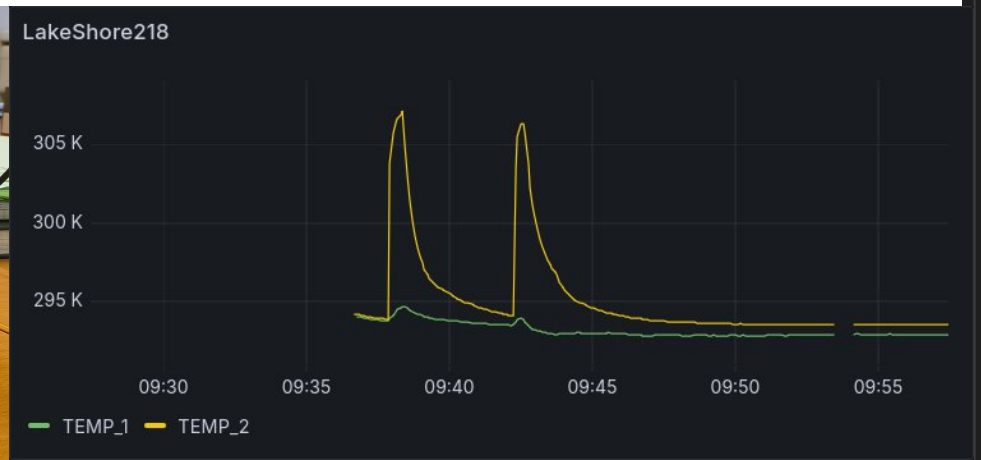
Evaluation Test @ DESY

- Caribou DAQ system with H2M running Constellation
- Adenium beam telescope
- AIDA2020 TLU
- Keithley 2410 SMU
- EUDAQ2 for data storage

MADMAX Temperature Monitoring

Dark Matter Experiment @ UHH

- 8-channel temperature monitor for MADMAX cryostat
- Communication via serial port
- Implementation in <100 LoC




```python
class LakeShore218(Satellite):
    _serial: pyvisa.resources.SerialInstrument

    def __init__(self, *args, **kwargs):
        self._rm = pyvisa.ResourceManager()
        self._lock = Lock()
        super().__init__(*args, **kwargs)

    def do_initializing(self, config: Configuration) -> None:
        port = config["port"]
        visa_address = f"ASRL{port}::INSTR"
        self.log.debug("Opening VISA device %s", visa_address)

        self._serial = self._rm.open_resource(  # type: ignore
            visa_address,
            baud_rate=9600,
            data_bits=7,
            stop_bits=pyvisa.constants.StopBits.one,
            parity=pyvisa.constants.Parity.odd,
            read_termination="\r\n",
            write_termination="\r\n",
        )

    def _get_temp(self, channel: int) -> float | None:
        # Check that _serial exists (required for metrics before INIT)
        if not hasattr(self, "_serial"):
            return None
        # Lock required since serial not thread safe
        with self._lock:
            # Check if input is enabled (return "0" or "1")
            if self._serial.query(f"INPUT? {channel}") == "1":
                return float(self._serial.query(f"KRDG? {channel}"))
        # Return None if not enabled
        return None

    @cscp_requestable
    def get_temp(self, request: CSCPMessage) -> tuple[str, Any, dict]:
        channel = int(request.payload)
        if channel < 1 or channel > 8:
            raise Exception(f"Channel {channel} does not exist")
        temp = self._get_temp(channel)
        verb = f"{temp}K" if temp else "Disabled"
        return verb, temp, {}

    @schedule_metric("K", MetricsType.LAST_VALUE, 5)
    def temp_1(self) -> Any:
        return self._get_temp(1)
```