

# QC2 Lab calibration toolkit

The QC2 Software Team



# Intro to the workshop

Michele



# What is QCT?

A software application designed to **calibrate** our QPUs with QBlox electronics

It exploits the **latest** version of electronic **firmware** to maximise performance and functionalities. This also simplify and extends support from QBlox

Built on top of **Tergite**, **Bragi** and years of experience, we designed it with both **production** and **research** in mind

The toolkit includes a Claude-powered UI, which simplify (re)-**analysis** and **data display**

The goal of the workshop is to give you an **introduction** and will serve as an entry point for future users

# Motivation for V2 of calibration

QBlox moved **away** from **quantify**, thus requiring us to rewrite all our schedules

Tergite was **not flexible enough** for calibrating the QPU, Bragi was introduced but developing **two packages** was not ideal

It was also very **production** oriented, not easy for physicists to use for new experiments

Difficult to create **new experiments** and have fine control of **parameters**

Initial design choices resulted in **bottlenecks** (nobody to blame, many large software projects hit this problem in v1)

Opportunity to **unify all our calibration efforts**: *Tergite, Bragi, Quantify Pingu, etc.*

# Schedule of the day

## 10:00 AM → 10:10 AM Introduction

Motivation, goals and scope of the new calibration toolkit.  
Plan for the day

**Speaker:** Michele Fauci Giannelli (chalmers University of Technology)

## 10:10 AM → 10:30 AM Architecture

including UI usage

**Speaker:** Matteo Robbiati (chalmers.se)

## 10:30 AM → 10:50 AM Discussion on architecture

## 10:50 AM → 11:10 AM How to design an experiment

Example of nodes, usage of QCM brief recap on migration of nodes

**Speaker:** Axel Andersson (chalmers.se)

## 11:10 AM → 11:30 AM Discussion on experiment creation

## 11:30 AM → 11:50 AM Calibration chains

Bringup examples

**Speaker:** Joel Sandås (chalmers.se)

## 12:10 PM → 12:15 PM Quantum Random Number

**Speaker:** Yanchen Lin (chalmers.se)

## 12:15 PM → 1:10 PM

Lunch

## 1:10 PM → 1:30 PM Future plans

**Speaker:** Matteo Robbiati (chalmers.se)

## 1:30 PM → 1:50 PM Discussion on future plans and directions

## 1:50 PM → 2:20 PM Publication strategy

**Speaker:** Michele Fauci Giannelli (Chalmers University of Technology)

# Toolkit architecture

Matteo & Axel

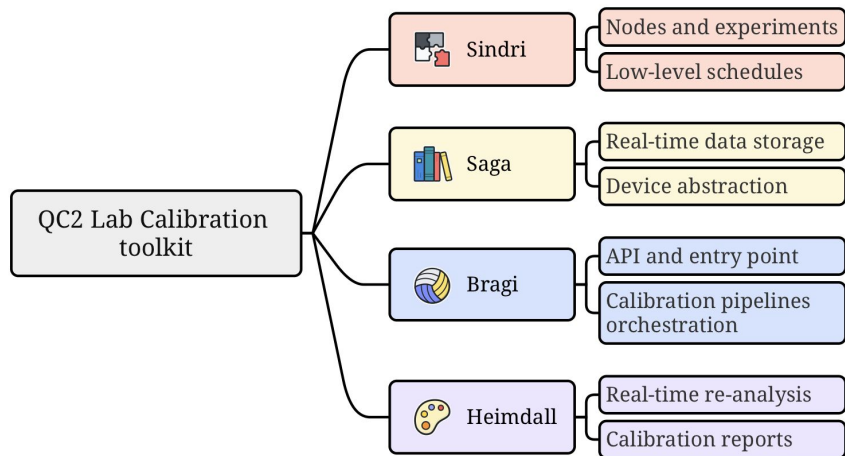


# Matteo & Axel

1. xmind description of the code
2. monorepo idea and dependencies
3. documentation and documentation preview
4. testing and coverage (83%, impressive)
5. heimdall preview
6. dataflow overview: no more redis bonded. Saga is the single source of truth
7. Introduction to the idea of calibration context (**sessions**). *The session is a very flexible concept that can be used and read in many different ways and remains an open area of architectural research.*
8. QCT as a programmatic language: you can surf on the surface or delve deep (**vibe** calibration mode or opening the **black box**)

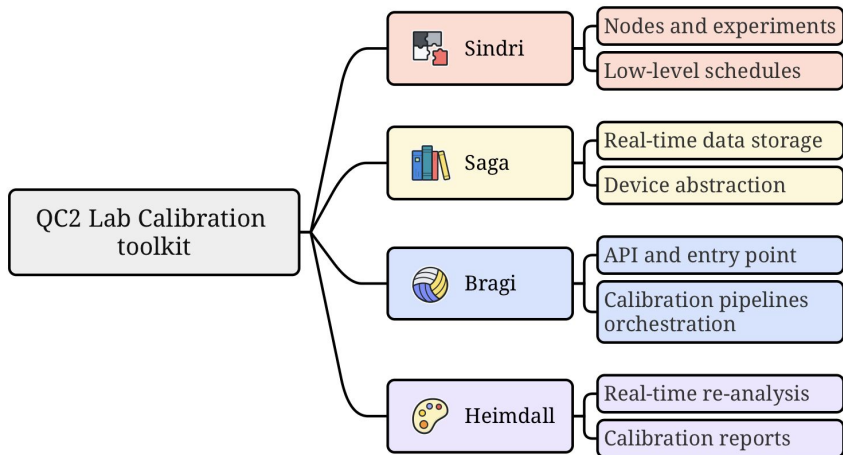
# High-level overview

- We created a monorepository containing a few packages.
- Each package is responsible of very precise set of tasks.
- We have common CI and versioning.
- Each package can be installed independently.
- Integration tests are shared, unit tests are specific
- Members of the team can contribute on part of the code



# High-level overview

- We created a monorepository containing a few packages.
- Each package is responsible of very precise set of tasks.
- We have common CI and versioning.
- Each package can be installed independently.
- Integration tests are shared, unit tests are specific
- Members of the team can contribute on part of the code

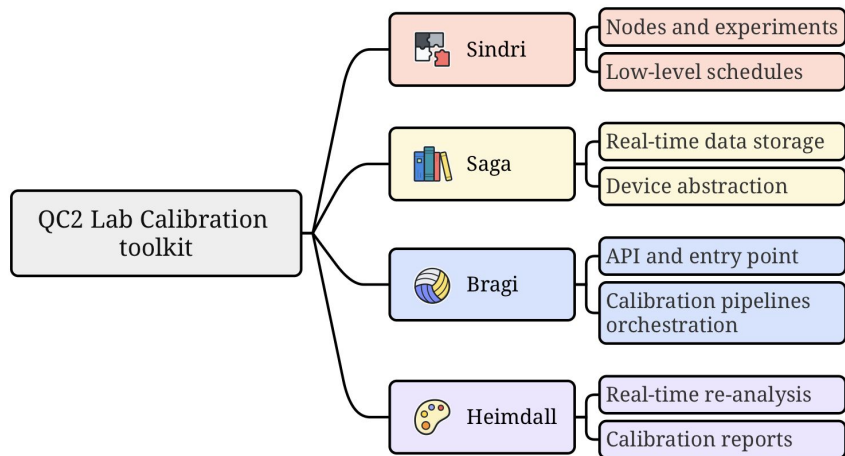


The screenshot shows the GitHub interface for the repository 'qc2-qcvv-suite'. At the top, there's a navigation bar with 'main' selected, a search bar, and buttons for '+', 'Find file', 'Code', and a menu icon. Below this, a message indicates a merge of branch '386-little-bug-in-displaying-the-wiring' into 'main' by Matteo Robbiati 1 day ago. The main content is a table listing the repository's files and their commit history.

| Name                           | Last commit                                     | Last update  |
|--------------------------------|-------------------------------------------------|--------------|
| device_templates               | update note in wiring.yaml                      | 3 weeks ago  |
| docs                           | Merge remote-tracking branch 'origin/mai...     | 2 days ago   |
| examples                       | revert default port 0000 in examples            | 3 weeks ago  |
| packages                       | fix: now set the list in QPU topology to be ... | 2 days ago   |
| tests                          | Merge remote-tracking branch 'origin/mai...     | 2 days ago   |
| utils_scripts                  | update header                                   | 3 weeks ago  |
| .gitignore                     | Merge remote-tracking branch 'origin/mai...     | 4 weeks ago  |
| .gitlab-ci.yml                 | Changelog.md                                    | 5 days ago   |
| .pre-commit-config.yaml        | Upgrade qblox-instruments to 1.3.1, work ...    | 6 days ago   |
| .python-version                | chore: pin project Python version to 3.12       | 3 months ago |
| CHANGELOG.md                   | Changelog.md                                    | 5 days ago   |
| README.md                      | docs: clarify PyQt6 group requirement an...     | 2 weeks ago  |
| confest.py                     | rename session option produce_1d_plots t...     | 1 week ago   |
| header_template.txt            | docs: old and new authors in the headers        | 3 months ago |
| pyproject.toml                 | Changelog.md                                    | 5 days ago   |
| qc2-qcvv-suite-param-flow.html | rename RunOptions to SessionOptions             | 3 weeks ago  |
| uv.lock                        | Changelog.md                                    | 5 days ago   |

# High-level overview

- We created a monorepository containing a few packages.
- Each package is responsible of very precise set of tasks.
- We have common CI and versioning.
- Each package can be installed independently.
- Integration tests are shared, unit tests are specific
- Members of the team can contribute on part of the code



The screenshot shows the GitHub interface for the repository 'qc2-qcvv-suite'. The 'packages' directory is highlighted with a red box. Below the directory listing, a detailed view of the 'packages' directory is shown, including a commit message and a table of files.

**Repository: qc2-qcvv-suite**

**Commit: Merge branch '386-little-bug-in-displaying-the-wiring' into 'main'** (Matteo Robbiati authored 1 day ago)

**Commit Hash: 9af2341b**

| Name             | Last commit                                            | Last update |
|------------------|--------------------------------------------------------|-------------|
| device_templates | update note in wiring.yaml                             | 3 weeks ago |
| docs             | Merge remote-tracking branch 'origin/main' into 'main' | 2 days ago  |
| examples         | revert default port 0000 in examples                   | 3 weeks ago |
| packages         | fix: now set the list in QPU topology to be ...        | 2 days ago  |
| tests            | Merge remote-tracking branch 'origin/main' into 'main' | 2 days ago  |
| utils_scripts    | update header                                          | 3 weeks ago |

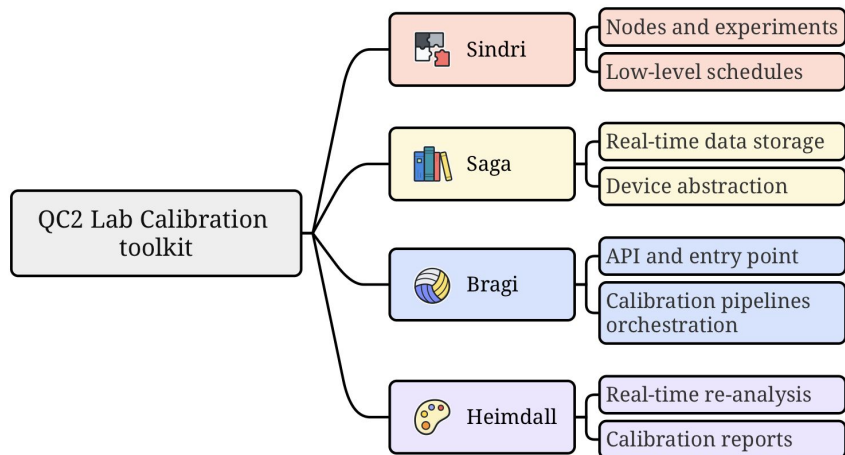
**packages**

**Commit: fix: now set the list in QPU topology to be horizontally scrollable if width is not enough** (Yanchen Lin authored 2 days ago)

| Name     | Last commit                                                                                |
|----------|--------------------------------------------------------------------------------------------|
| bragi    | Merge remote-tracking branch 'origin/main' into 252-heimdall-better-archive-sidebar        |
| heimdall | fix: now set the list in QPU topology to be horizontally scrollable if width is not enough |
| saga     | Merge remote-tracking branch 'origin/main' into mr-145                                     |
| sindri   | Merge remote-tracking branch 'origin/main' into mr-145                                     |

# High-level overview

- We created a monorepository containing a few packages.
- Each package is responsible of very precise set of tasks.
- We have common CI and versioning.
- Each package can be installed independently.
- Integration tests are shared, unit tests are specific
- Members of the team can contribute on part of the code



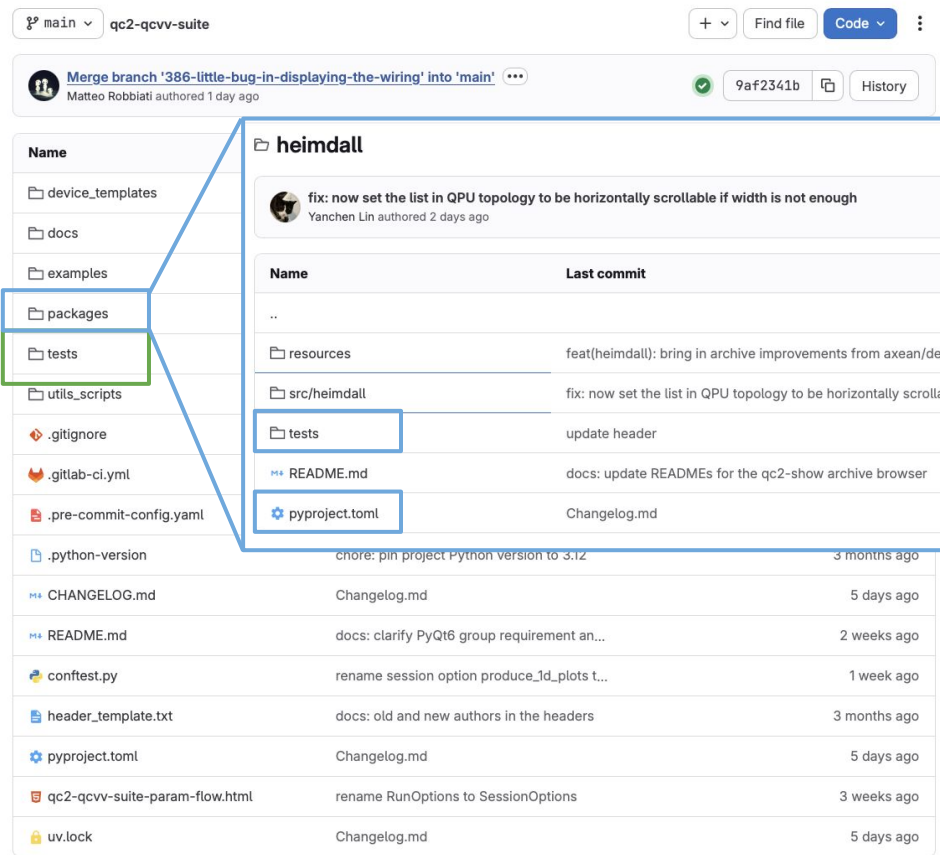
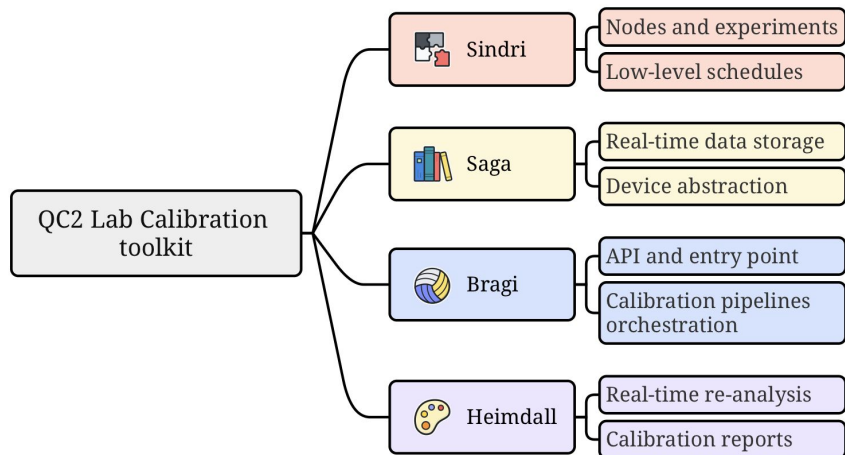
main qc2-qcvv-suite

Merge branch '386-little-bug-in-displaying-the-wiring' into 'main' Matteo Robbiati authored 1 day ago

| Name                           | Last commit                                     | Last update  |
|--------------------------------|-------------------------------------------------|--------------|
| device_templates               | update note in wiring.yaml                      | 3 weeks ago  |
| docs                           | Merge remote-tracking branch 'origin/mai...     | 2 days ago   |
| examples                       | revert default port 0000 in examples            | 3 weeks ago  |
| packages                       | fix: now set the list in QPU topology to be ... | 2 days ago   |
| tests                          | Merge remote-tracking branch 'origin/mai...     | 2 days ago   |
| utils_scripts                  | update header                                   | 3 weeks ago  |
| .gitignore                     | Merge remote-tracking branch 'origin/mai...     | 4 weeks ago  |
| .gitlab-ci.yml                 | Changelog.md                                    | 5 days ago   |
| .pre-commit-config.yaml        | Upgrade qblox-instruments to 1.3.1, work ...    | 6 days ago   |
| .python-version                | chore: pin project Python version to 3.12       | 3 months ago |
| CHANGELOG.md                   | Changelog.md                                    | 5 days ago   |
| README.md                      | docs: clarify PyQt6 group requirement an...     | 2 weeks ago  |
| conftest.py                    | rename session option produce_to_plots t...     | 1 week ago   |
| header_template.txt            | docs: old and new authors in the headers        | 3 months ago |
| pyproject.toml                 | Changelog.md                                    | 5 days ago   |
| qc2-qcvv-suite-param-flow.html | rename RunOptions to SessionOptions             | 3 weeks ago  |
| uv.lock                        | Changelog.md                                    | 5 days ago   |

# High-level overview

- We created a monorepository containing a few packages.
- Each package is responsible of very precise set of tasks.
- We have common CI and versioning.
- Each package can be installed independently.
- Integration tests are shared, unit tests are specific
- Members of the team can contribute on part of the code



# Maintenance, interactive docs

## QC2 Calibration Toolkit

pipeline passed coverage 83.00% Ruff code style black ESLint checked ts types checked

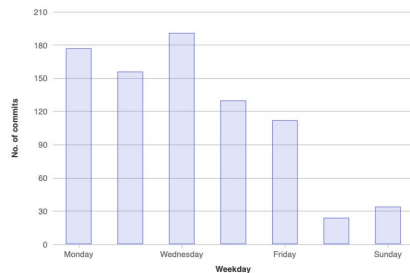
### Code coverage statistics for main Jun 04 - Sep 02

Download raw data (.csv)

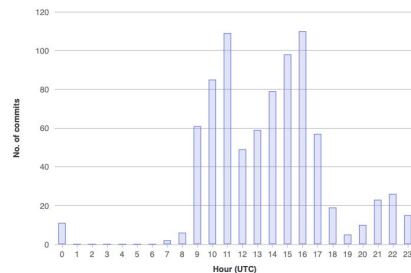
coverage-all ▾



### Commits per weekday



### Commits per day hour (UTC)



### GETTING STARTED

Installation

Architecture

Core Concepts

### USING THE TOOLKIT

Running Experiments

Measurement Helpers

The Replot Server (Heimdall)

### NODE LIBRARY

Node Registry

Node Library

Readout characterization >

Qubit control >

Qubit characterization >

Coupler characterization >

### TUTORIALS

Tutorial: Resonator

Spectroscopy

### API REFERENCE

Sindri API

Saga API

Bragi API

Heimdall API

### PROJECT

Changelog

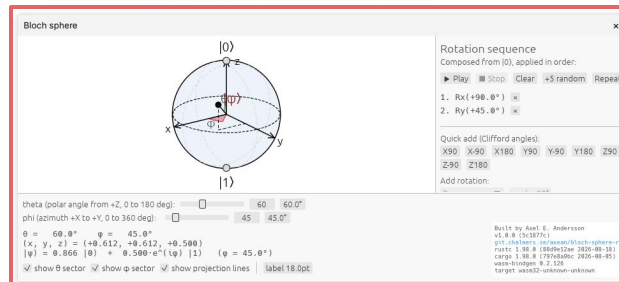
## Core Concepts

The core component of our library are surely the calibration nodes: every high-level pipeline needs these building blocks as its core! A calibration node, in our framework, is not only a node in a graph, but instead a well structured software piece, composed of measurement, analysis and tracking tools. Series of calibration nodes can be structured with complete freedom and grouped into calibration sessions. A Session is here a Python context where an arbitrary calibration environment is set by the user. Organizing our work into session also facilitates the visualization of our results through Heimdall.

Many of the calibration nodes below reason about qubit state as a rotation on the Bloch sphere (the same  $\theta/\phi$  convention used by `qblox_scheduler`'s gate library). If that's unfamiliar, or you just want to build intuition before diving into a specific node, try the interactive widget:

[Open interactive Bloch sphere](#) `|0> → |1> (X gate)`

Every field of a node's `BaseParameters` falls into exactly one of three categories:



| Piece      | Base class     | Responsibility                                                                                                                                   |
|------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters | BaseParameters | A validated (Pydantic, <code>avroas*forbid*</code> ) description of the sweep — which qubits, how many samples, window widths, thresholds. Qubit |

# Maintenance, interactive docs

## QC2 Calibration Toolkit

pipeline passed coverage 83.00%  Ruff code style black  ESLint checked  types checked

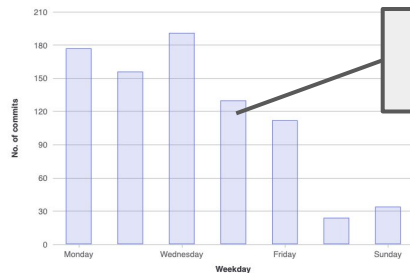
### Code coverage statistics for main Jun 04 - Sep 02

Download raw data (.csv)

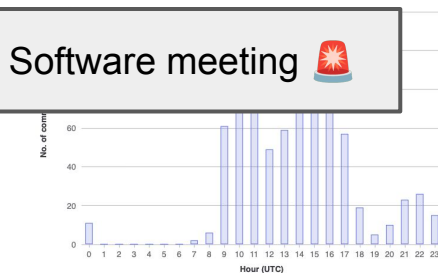
coverage-all ▾



### Commits per weekday



### Commits per day hour (UTC)



Software meeting 

## Core Concepts

### GETTING STARTED

Installation

Architecture

Core Concepts

### USING THE TOOLKIT

Running Experiments

Measurement Helpers

The Replot Server (Heimdall)

### NODE LIBRARY

Node Registry

Node Library

Readout characterization >

Qubit control >

Qubit characterization >

Coupler characterization >

### TUTORIALS

Tutorial: Resonator

Spectroscopy

### API REFERENCE

Sindri API

Saga API

Bragi API

Heimdall API

### PROJECT

Changelog

### ON THIS PAGE

Anatomy of a calibration node

The node lifecycle

Node families

The session

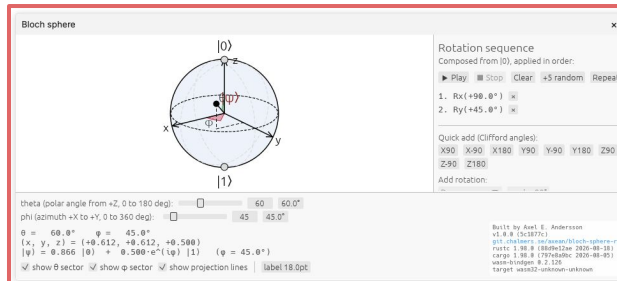
Where results are stored

The core component of our library are surely the calibration nodes: every high-level pipeline needs these building blocks as its core! A calibration node, in our framework, is not only a node in a graph, but instead a well structured software piece, composed of measurement, analysis and tracking tools. Series of calibration nodes can be structured with complete freedom and grouped into calibration sessions. A Session is here a Python context where an arbitrary calibration environment is set by the user. Organizing our work into session also facilitates the visualization of our results through Heimdall.

Many of the calibration nodes below reason about qubit state as a rotation on the Bloch sphere (the same  $\theta/\phi$  convention used by `qblox_scheduler`'s gate library). If that's unfamiliar, or you just want to build intuition before diving into a specific node, try the interactive widget:

[Open interactive Bloch sphere](#) [|0> → |1> \(X gate\)](#)

Every field of a node's `BaseParameters` falls into exactly one of three categories:



| Piece      | Base class                  | Responsibility                                                                                                                                   |
|------------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters | <code>BaseParameters</code> | A validated (Pydantic, <code>avroas*forbid*</code> ) description of the sweep — which qubits, how many samples, window widths, thresholds. Qubit |

# Maintenance, **interactive** docs (together)

## Add timing analysis for nodes/runs

[Edit](#)[Code](#) ▾[Open](#) Yanchen Lin requested to merge `373-add-timing-analysis-fo...` into `main` 6 days ago[Overview](#) 2[Commits](#) 12[Pipelines](#) 4[Changes](#) 28

Closes #373

Add timing analysis for a node run, showing the proportion of time spent during the calibration. And writes to a separate log file as the current session\_timeline.json can be messy.

Using the categories: preparing, measurement, analysis, and logging.

This can help devs and users to detect anomalies and understand the limitation, useful for debugging and future enhancement to speed up.

Expanded the @timed decorator, used the log format called Chrome Perfetto Trace Event Format, added a section on Heimdall to show the time swim lane chart.

TODO:

- ☒ Aggregate the per-session trace events of a run and find a place on heimdall to put them.

Edited 21 hours ago by Yanchen Lin



0



0



Merge request pipeline #213527 passed

Merge request pipeline passed for `879bcc9f` 21 hours ago [?](#)

Test coverage 84.00% (1.00%) from 1 job [?](#)



Deployed to docs-preview/373-add-timing-analysis-for-a-node-calibration-run 21 hours ago

[View app](#) 

Assignee

Edit



Yanchen Lin

Reviewer

Edit

None - assign yourself

Labels

Edit

Enhancement x

Future x

This opens the updated documentation, which will be rendered once this MR is merged

⌚ +

No estimate or time spent

1 Participant



# Heimdall: re-analysis and visualization tool

Calibration Runs

Toolkit documentation

Search runs, sessio

Overview

2026-08-31

6

sqbu\_test\_115...

2 sessions

10 qubits

18:59:04

sqbu\_test\_115...

2 sessions

10 qubits

14:39:29

sqbu\_test\_115...

2 sessions

10 qubits

18:12:39

sqbu\_test\_115...

2 sessions

10 qubits

14:04:50

single-qubit\_b...

4 sessions

10 qubits

00:45:44

sqbu\_no\_pdf

4 sessions

10 qubits

00:04:21

2026-08-18

1

2026-08-17

14

2026-08-16

3

2026-08-14

9

2026-08-13

6

2026-08-12

2

2026-08-11

12

2026-08-10

1

2026-08-07

8

2026-08-06

3

2026-08-05

10

2026-07-31

1

SELECTED RUN

2026-08-31 14:59:04

2 sessions

10 qubits

viewer: heimdall v0.1.0

QC2 Calibration Measurement Sessions

sqbu test 1155

search\_resonators\_0

search\_qubits\_01

Whole Pipeline

QPU Topology

Loki · FC8 · LINQ-80

trace a line – try q01:mw · XY1 · F23 · port 9

XY24

port q24:mw

LOUT1\_B - MODULE 5  
01 (complex\_output\_1)  
QCM\_RF - 6 dB out-att - LO 4.6301 GHz

RD FRIDGE LINE  
D7  
RLC Electronics F-30-8000R

HOLDER PORT  
50  
LINQ-80

CHIP  
XY24  
XY drive pad

related: ZC19-24 ZC23-24 ZC24-25 R05-IN

all

drive

flux

readout

other

| PORT | CHIP    | LINE | HO | FILTER                                           | PORT       | INSTRUMENT                   |
|------|---------|------|----|--------------------------------------------------|------------|------------------------------|
| 31   | ZC20-25 | D09  |    | Mini-circuits VLFX-1100                          | q20_q25:fl | loki_b m16 00 · SPI m8 dac0  |
| 32   | R04     | A18  |    | KeenLion KBF-4/7-2S + RLC Electronics F-30-8000R | -          | loki_b m19 I0                |
| 33   | XY20    | A8   |    | RLC Electronics F-30-8000R                       | q20:mw     | loki_b m6 00                 |
| 34   | ZC19-20 | D04  |    | Mini-circuits VLFX-1100                          | q19_q20:fl | loki_b m12 00 · SPI m10 dac3 |
| 35   | XY19    | D3   |    | RLC Electronics F-30-8000R                       | q19:mw     | loki_b m5 00                 |
| 36   | ZC18-19 | D03  |    | Mini-circuits VLFX-1100                          | q18_q19:fl | loki_b m11 00 · SPI m10 dac2 |
| 41   | ZC19-24 | D08  |    | Mini-circuits VLFX-1100                          | q19_q24:fl | loki_b m15 01 · SPI m5 dac3  |
| 42   | R05     | A19  |    | KeenLion KBF-4/7-2S + RLC Electronics F-30-8000R | -          | loki_b m20 I0                |
| 43   | XY25    | A7   |    | RLC Electronics F-30-8000R                       | q25:mw     | loki_b m6 01                 |
| 44   | ZC24-25 | D013 |    | Mini-circuits VLFX-1100                          | q24_q25:fl | loki_b m12 01 · SPI m1 dac3  |
| 50   | XY24    | D7   |    | RLC Electronics F-30-8000R                       | q24:mw     | loki_b m5 01                 |
| 51   | ZC18-23 | D07  |    | Mini-circuits VLFX-1100                          | q18_q23:fl | loki_b m15 00 · SPI m5 dac0  |
| 52   | ZC16-21 | D05  |    | Mini-circuits VLFX-1100                          | q16_q21:fl | loki_b m18 00 · SPI m5 dac2  |
| 53   | ZC17-22 | D06  |    | Mini-circuits VLFX-1100                          | q17_q22:fl | loki_b m18 01 · SPI m5 dac1  |
| 54   | R05-IN  | A14  |    | RLC Electronics F-30-8000R                       | -          | loki_b m20 00                |
| 55   | XY21    | D5   |    | RLC Electronics F-30-8000R                       | q21:mw     | loki_b m1 01                 |
| 56   | ZC21-22 | D010 |    | Mini-circuits VLFX-1100                          | q21_q22:fl | loki_b m8 01 · SPI m1 dac0   |
| 57   | XY22    | A4   |    | RLC Electronics F-30-8000R                       | q22:mw     | loki_b m2 01                 |

3 selected

All qubits

All couplers

X Clear

Click to toggle · drag to select a group · Alt+drag to de-select · greyed-out components have no run data and can't be selected

**QC3 Calibration Measurement Sessions**  
sqbu test 1155

Search runs, sessio...  
search\_resonators\_0  
search\_qubits\_01  
Whole Pipeline

**Selected Components — Run Nodes**

q22 q18 q24 3 run node(s) covering the whole selection

#1 resonator\_spectroscopy\_wide 8/31/2026, 2:59:29 PM

Results Reanalyse (3)

Plot: main

**ANALYSIS PARAMETERS (APPLIED TO ALL 3 SELECTED ELEMENTS)**

|                                         |                   |                   |                     |                     |
|-----------------------------------------|-------------------|-------------------|---------------------|---------------------|
| sample_size_for_sliding_window_analysis | slide_window_size | slide_window_step | slide_window_min_q1 | slide_window_max_qe |
| 200                                     | 100               | 50                | 7000                | 60000               |
| min_r_squared                           | max_abs_theta     |                   |                     |                     |
| 0.9                                     | 0.6               |                   |                     |                     |

Re-analyse all Reset

**q22**

**Resonator Spectroscopy Wide**

Resonator |0> Spectroscopy for q22

Frequency (Hz)

q24

**q18**

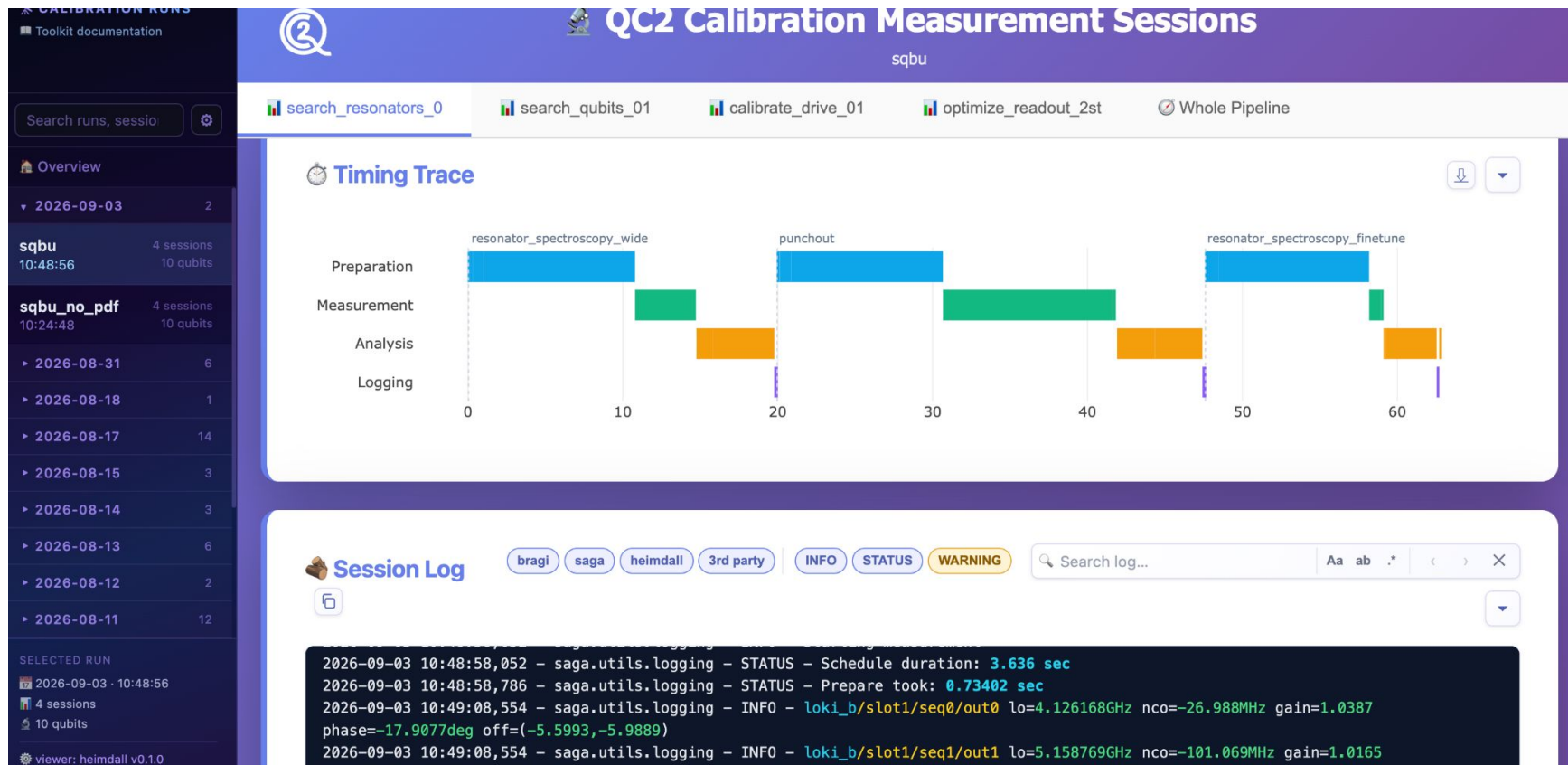
**Resonator Spectroscopy Wide**

Resonator |0> Spectroscopy for q18

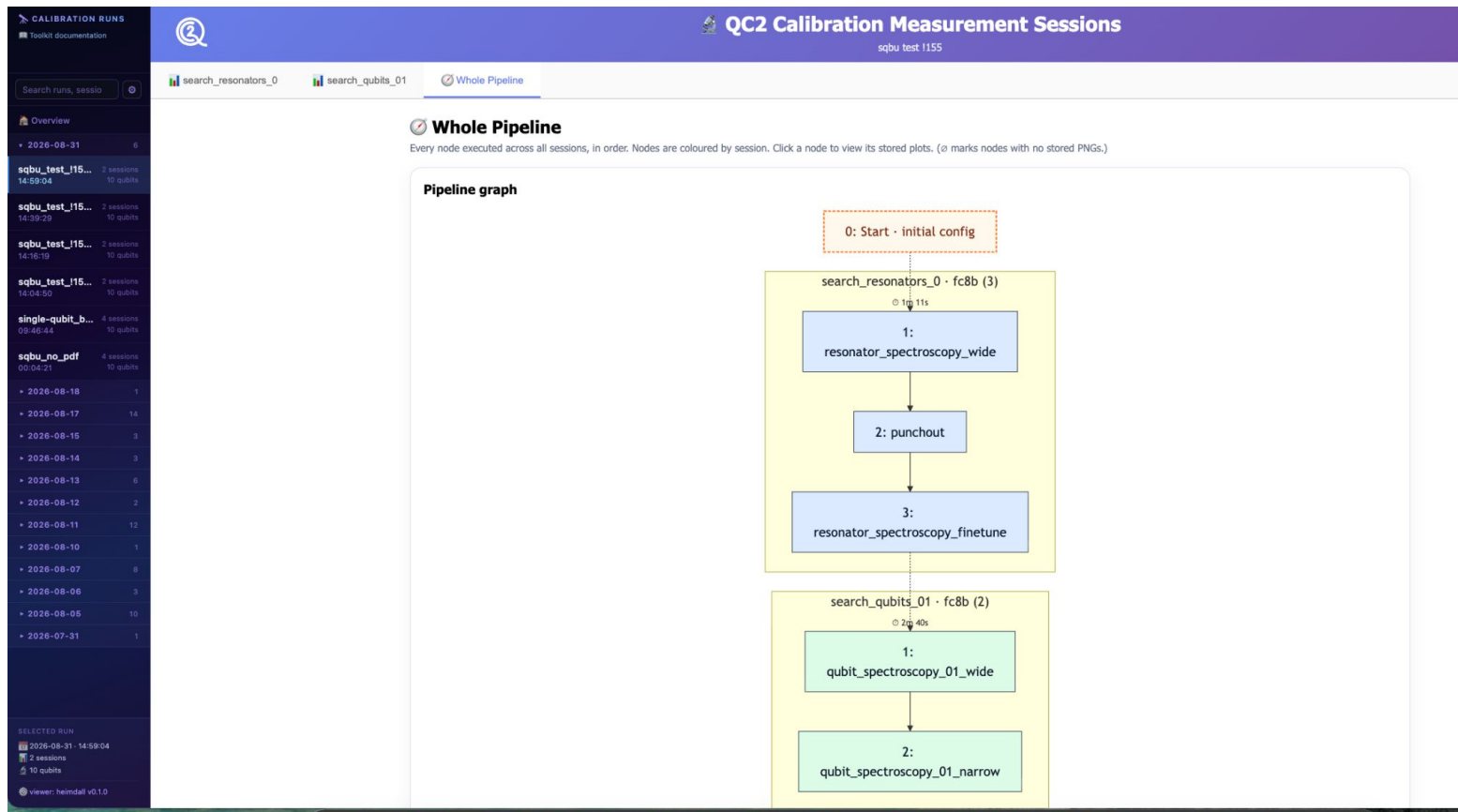
Frequency (Hz)

**SELECTED RUN**  
2026-08-31 14:59:04  
2 sessions  
10 qubits  
viewer: heimdall v0.1.0

# Heimdall: re-analysis and visualization tool

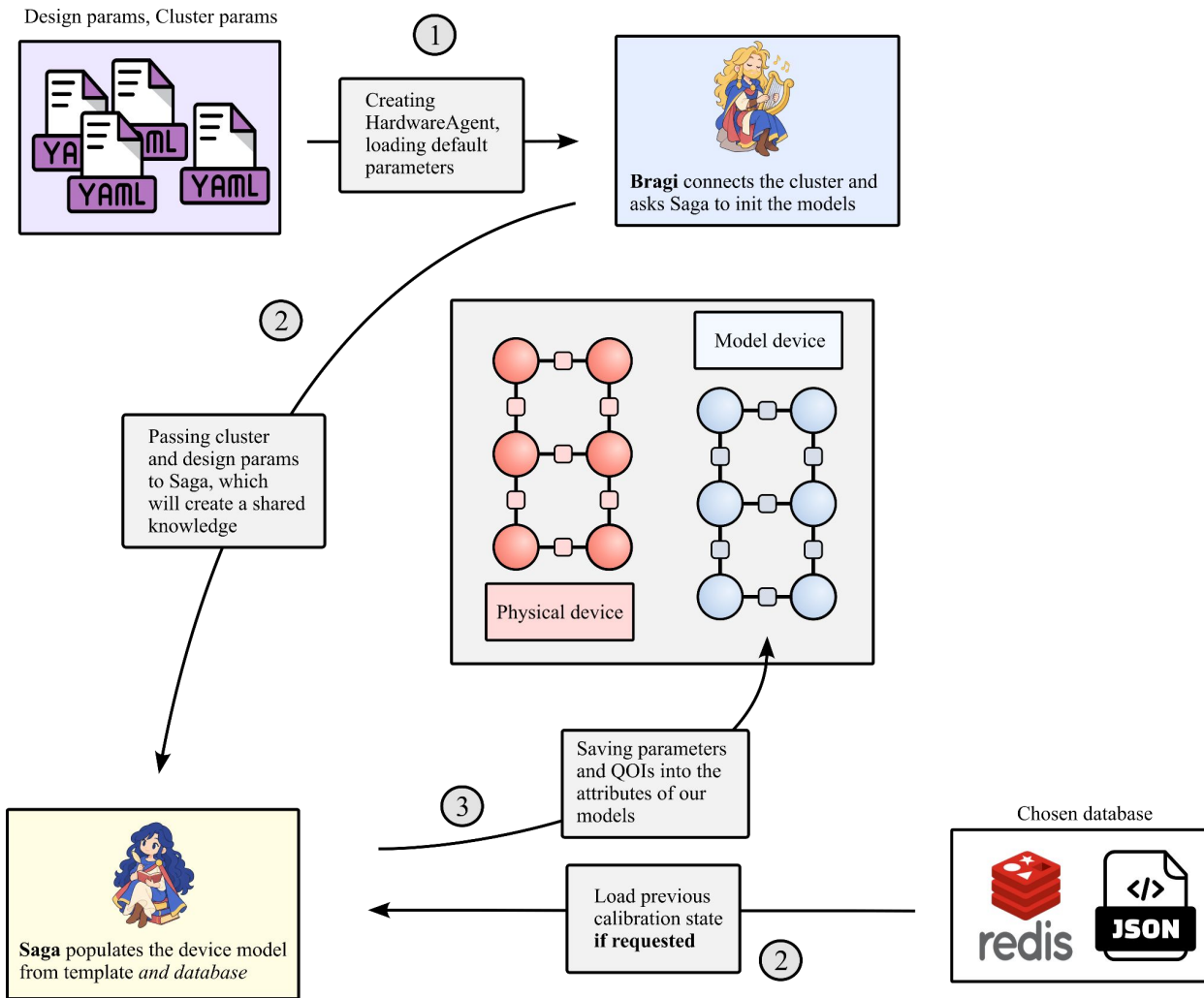


# Heimdall: re-analysis and visualization tool



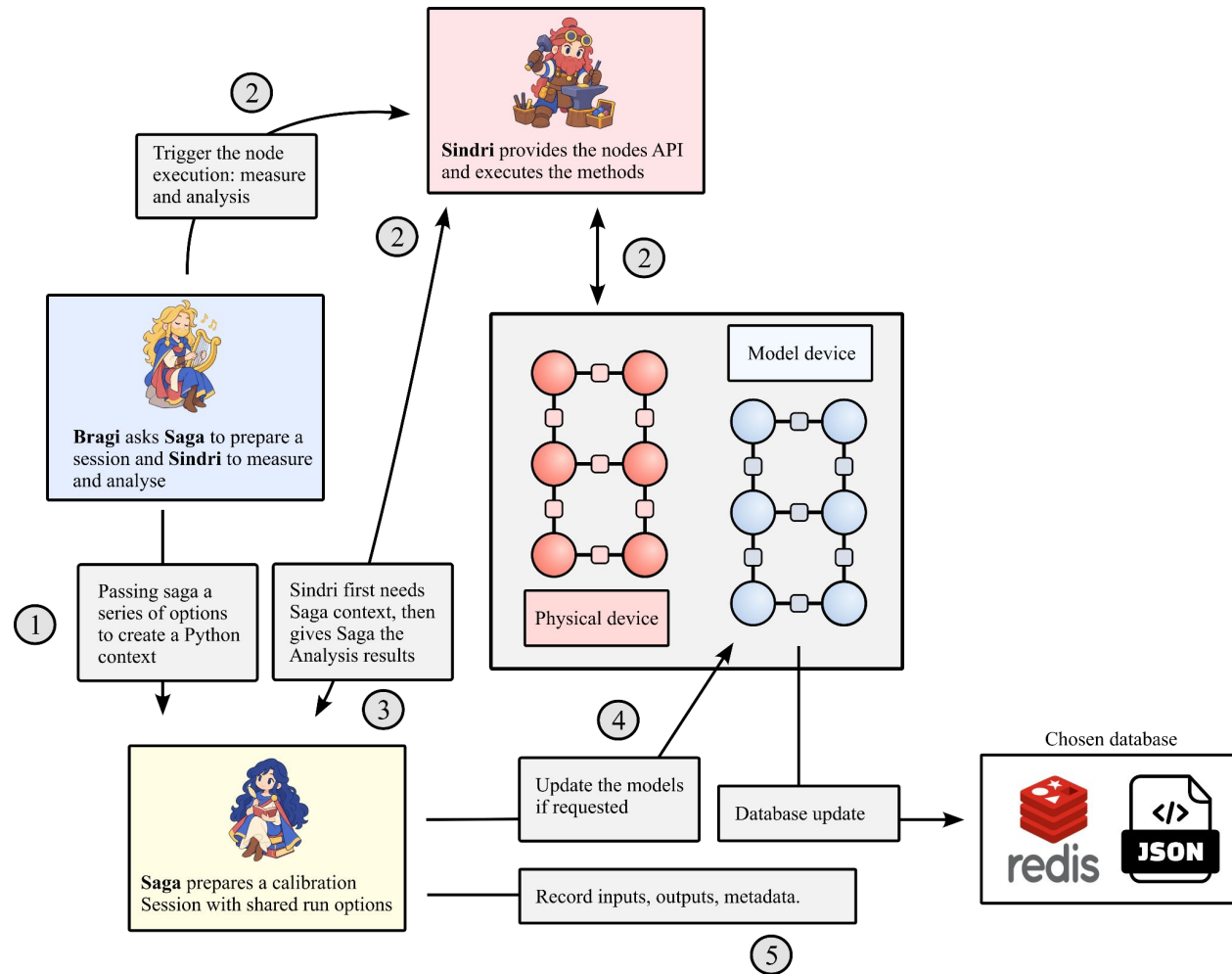
# Workflow #1

## *Preparation*

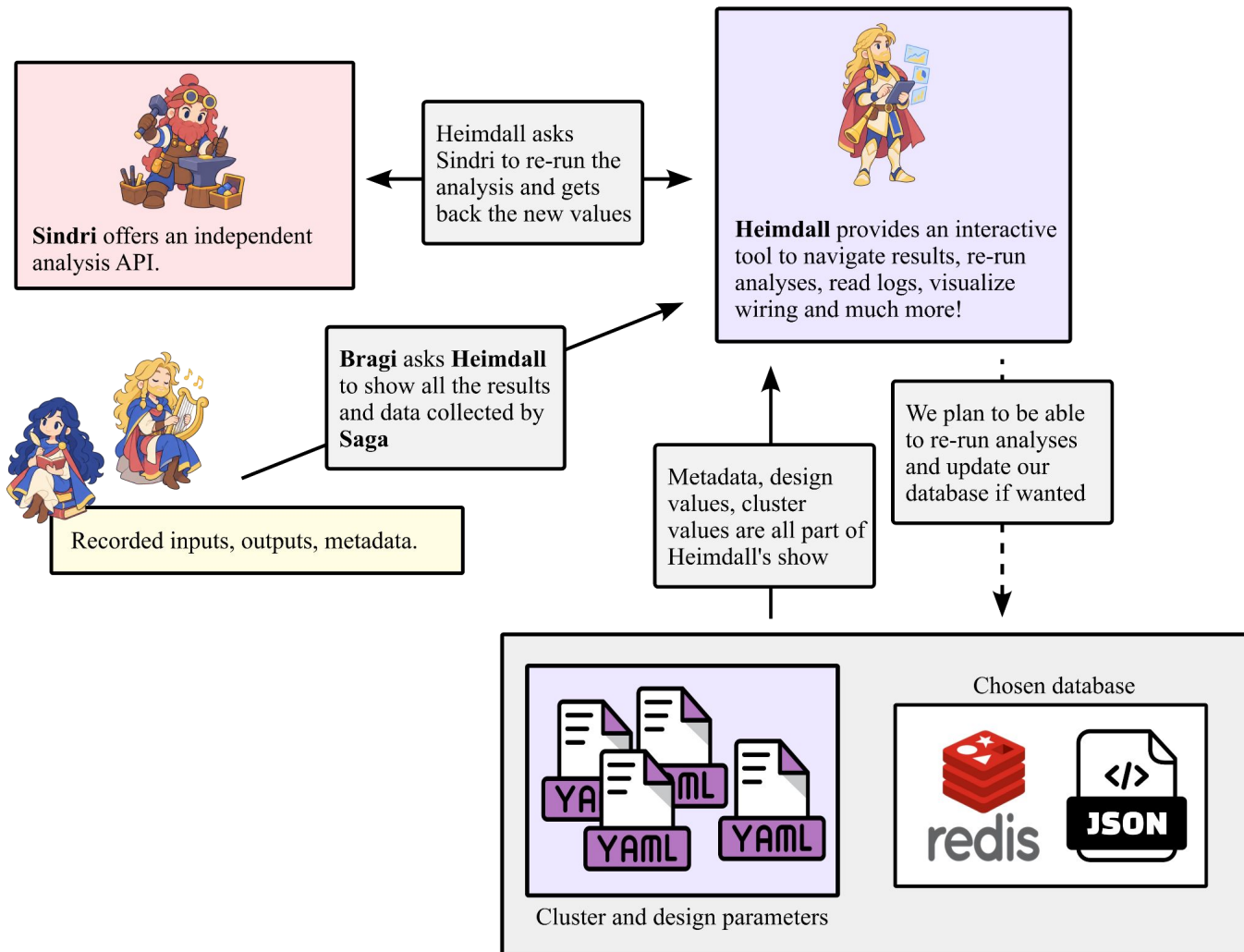


## Workflow #2

### *Execution*



## Workflow #3 *Visualization*



# Using the Calibration Orchestrator



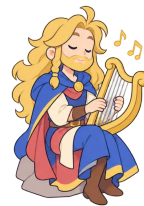
main qc2-qcvv-suite + Find file Code

Merge branch '386-little-bug-in-displaying-the-wiring' into 'main' Matteo Robbiati authored 1 day ago

| Name                           | Last update                                  |
|--------------------------------|----------------------------------------------|
| device_templates               |                                              |
| docs                           |                                              |
| examples                       |                                              |
| packages                       |                                              |
| tests                          | Merge remote-tracking branch 'origin/mai...  |
| utils_scripts                  | update header                                |
| .gitignore                     | Merge remote-tracking branch 'origin/mai...  |
| .gitlab-ci.yml                 | Changelog.md                                 |
| .pre-commit-config.yaml        | Upgrade qblox-instruments to 1.3.1, work ... |
| .python-version                | chore: pin project Python version to 3.12    |
| CHANGELOG.md                   | Changelog.md                                 |
| README.md                      | docs: clarify PyQt6 group requirement an...  |
| confest.py                     | rename session option produce_id_plots t...  |
| header_template.txt            | docs: old and new authors in the headers     |
| pyproject.toml                 | Changelog.md                                 |
| qc2-qcvv-suite-param-flow.html | rename RunOptions to SessionOptions          |
| uv.lock                        | Changelog.md                                 |

```
18 from pathlib import Path
19
20 import numpy as np
21
22 from bragi.handlers.data import RedisDataHandler
23 from bragi.orchestra import CalibrationOrchestrator
24 from saga.data_managing.session import SessionOptions
25
26 # Insert your Redis port here
27 REDIS_PORT = 6395
28 RUN_NAME = "single-qubit bring-up odin"
29
30 # QUBITS = [f"q{i:02d}" for i in range(1, 11)]
31 QUBITS = ["q01", "q03", "q04", "q06", "q09", "q10"]
32
33 TEMPLATE_NAME = "fc8_odin"
34
35
36 # Set to True when restarting from a mid-chain step (e.g. START_FROM="ro_amplitude")
37 # so the orchestrator reloads all upstream results from Redis before running the
38 # selected step(s), instead of starting from bare YAML defaults.
39 LOAD_CALIBRATION_STATE: bool = False
40
41 orchestrator = CalibrationOrchestrator(
42     experiment_template_name=TEMPLATE_NAME,
43     data_handler=RedisDataHandler(redis_port=REDIS_PORT),
44     device_templates_dir=Path("/home/axean/repositories/device_templates").resolve(),
45     generate_pdf_report=False,
46     load_calibration_state=LOAD_CALIBRATION_STATE,
47     name=RUN_NAME,
48     reset_clusters=True,
49 )
```

# Using the Calibration Orchestrator



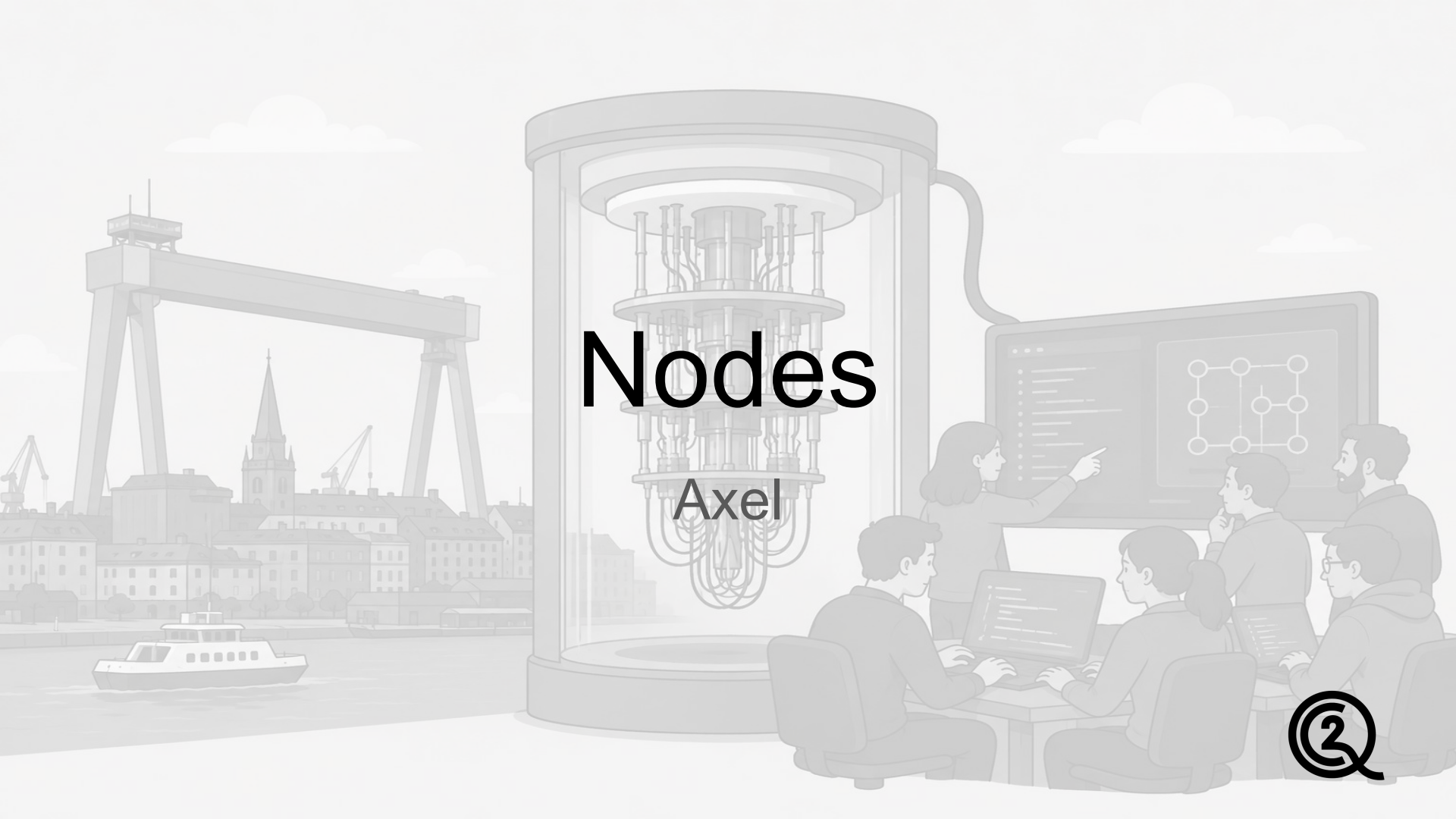
```
def calibrate_drive(
    orchestrator: CalibrationOrchestrator, qubits, session_name: str, transition: str
) -> None:
    suffix = "" if transition == "01" else f"_{transition}"
    with orchestrator.calibration_session(run_options=_session_options(session_name)):
        orchestrator(
            f"rabi_oscillations{suffix}",
            qubits=qubits,
            min_amplitude=-0.6,
            max_amplitude=0.6,
            amplitude_bins=80,
            repetitions=512,
        )

    orchestrator(
        f"ramsey_fringes{suffix}",
        qubits=qubits,
        ramsey_delays=list(np.arange(16e-9, 1.04e-6, 64e-9)),
        artificial_detunings=list(np.linspace(-2.0, 2.0, 4) * 1e6),
        repetitions=512,
    )

    orchestrator(
        f"motzoi_parameter{suffix}",
        qubits=qubits,
        min_motzoi=-2e-9,
        max_motzoi=2e-9,
        motzoi_bins=31,
        repetitions=512,
    )

    orchestrator(
        f"n_rabi_oscillations{suffix}",
        name="n_rabi_oscillations",
        qubits=qubits,
        min_amplitude=-0.06,
        max_amplitude=0.06,
        amplitude_bins=30,
        x_repetitions=[1, 3, 5],
        repetitions=512,
    )
```

```
17 import common
18
19 args = common.parse_args()
20 TEMPLATE_NAME = args.chip
21 QUBITS = common.CHIP_CONFIGS[TEMPLATE_NAME]["qubits"]
22 REDIS_PORT = args.port
23
24 # Set to True when restarting from a mid-chain step (e.g. START_FROM="ro_amplitude")
25 # so the orchestrator reloads all upstream results from Redis before running the
26 # selected step(s), instead of starting from bare YAML defaults.
27 LOAD_CALIBRATION_STATE: bool = True
28
29 orchestrator = common.make_orchestrator(
30     TEMPLATE_NAME, REDIS_PORT, LOAD_CALIBRATION_STATE
31 )
32
33 # -----
34 # Run the full bringup chain, 0->1 then 1->2.
35 # -----
36
37 common.search_resonators(orchestrator, QUBITS, "search_resonators_0")
38 common.search_qubits_01(
39     orchestrator, QUBITS, "search_qubits_01", full_scan=TEMPLATE_NAME == "pb_sos1_v1"
40 )
41
42 if TEMPLATE_NAME == "fc8_odin":
43     common.calibrate_drive(
44         orchestrator, QUBITS, "calibrate_drive_01_iter1", transition="01"
45     )
46     common.calibrate_drive(
47         orchestrator, QUBITS, "calibrate_drive_01_iter2", transition="01"
48     )
49     common.optimize_readout_two_state(
50         orchestrator, QUBITS, "optimize_readout_2st_iter1"
51     )
52     common.optimize_readout_two_state(
53         orchestrator, QUBITS, "optimize_readout_2st_iter2"
54     )
55     common.characterize_t1(orchestrator, QUBITS, "characterize_t1")
56     common.characterize_t2(orchestrator, QUBITS, "characterize_t2")
57
58 # common.search_resonators(orchestrator, QUBITS, "search_resonators_1", suffix="_1")
59 # common.search_qubits_12(orchestrator, QUBITS, "search_qubits_12")
60 # common.calibrate_drive(orchestrator, QUBITS, "calibrate_drive_12_iter1", transition="12")
61 # common.calibrate_drive(orchestrator, QUBITS, "calibrate_drive_12_iter2", transition="12")
62 # common.optimize_readout_three_state(orchestrator, QUBITS, "optimize_readout_3st")
63 # common.characterize_randomized_benchmarking(orchestrator, QUBITS, "characterize_rb")
64
65 # Generate master HTML with all sessions
66 orchestrator.generate_html_report()
```



# Nodes

Axel





Search



## GETTING STARTED

Installation

Architecture

Core Concepts

## USING THE TOOLKIT

Running Experiments

Measurement Helpers

The Repl Server (Heimdall)

## NODE LIBRARY

Node Registry

Node Library

Readout characterization

Qubit control

Qubit characterization

Coupler characterization

## TUTORIALS

Tutorial: Resonator

Spectroscopy

## API REFERENCE

Sindri API

Saga API

Bragi API

Heimdall API

## PROJECT

Changelog

# Node Library

This section describes the **physics and analysis** behind each calibration node available in [Sindri](#) — what it measures, the pulse scheme it runs, and the quantities it extracts.

For the programmatic reference (classes, parameters, QOIs) see the [Sindri API](#); for how to run a node see [Running Experiments](#).

## Parameter field categories

Every field of a node's parameters class falls into exactly one of three categories (see [Core Concepts](#) for the full picture):

- **metadata** — Identity/targeting fields (node name, which qubits/couplers the run applies to). Not a tunable knob: changing one identifies a different run rather than re-processing the same data.
- **acquisition** — fields baked into the hardware measurement itself (sweep ranges, number of points, repetitions, pulse settings). Changing one requires re-running on hardware; it cannot be re-applied to already-saved data.
- **analysis** — fields consumed only in post-processing/fitting (thresholds,  $R^2$  cut-offs, fit windows). Safe to change and re-apply to saved data, e.g. during a reanalysis in Heimdall.

The “Parameters” section of each node page below groups that node's fields under these three headings so it's clear at a glance which knobs require a hardware re-run and which can be tweaked during reanalysis.

## Categories

- Readout characterization
  - Punchout
  - Readout Amplitude Optimization
  - Readout Combined Optimization
  - Readout Frequency Optimization
  - Resonator Spectroscopy

## ON THIS PAGE

[Parameter field categories](#)[Categories](#)

### Node Library

#### Readout characterization

Punchout

Readout Amplitude Optimization

Readout Combined Optimization

Readout Frequency Optimization

Resonator Spectroscopy

#### Qubit control

Motzoi (DRAG)

N Rabi Oscillation

Qubit Spectroscopy

Rabi Oscillations

Ramsey Correction

#### Qubit characterization

Purity Benchmarking

Randomized Benchmarking

T1

T2

T2 Echo

#### Coupler characterization

CZ Chevron

CZ Conditional Phase &amp; CZ Local Phases

CZ Currents From Spectroscopies

CZ Parametrization (Fixed Duration)

CZ Wall Finder

Qubit Spectroscopy vs Current

Resonator Spectroscopy vs Current

Two-Qubit Gate

Randomized Benchmarking

Searched for 1 pattern, listed 2 directories, ran 5 shell commands

Node registry — `packages/sindri/src/sindri/nodes.yaml`

- 51 variants total, across 4 categories (readout 17, coupler 16, qubit\_control 12, characterization 6) and 24 node entries.

- 23 variants are undocumented — their names appear nowhere under `docs/` (checked whole-word across all docs pages), i.e. ~45% coverage gap; the remaining 28 are mentioned in at least one doc page.

- The undocumented ones are mostly higher-state/secondary variants: `*_12` transitions (`qubit_spectroscopy_12`, `ramsey_fringes_12`, `motzoi_parameter_12`, `n_rabi_oscillations_12`, `-`), readout-optimization state variants (`ro_amplitude/frequency/combined_optimization_2state|3state`, `multistate_rof_2state|3state`), `punchout_1/punchout_2`, `resonator_spectroscopy_2`, `resonator_spectroscopy_vs_current_1|2`, plus `all_xy`, `cz_amp_duration`, `cz_sea_fix_cycles`, `n_rabi_oscillations`, `ramsey_fringes_x90x90_12`).



## GETTING STARTED

Installation  
Architecture  
Core Concepts

## USING THE TOOLKIT

Running Experiments  
Measurement Helpers  
The Replot Server (Heimdall)

## NODE LIBRARY

Node Registry  
Node Library ▼  
    Readout characterization >  
    Qubit control >  
    Qubit characterization >  
    Coupler characterization >

## TUTORIALS

Tutorial: Resonator  
Spectroscopy

## API REFERENCE

Sindri API  
Saga API  
Bragi API  
Heimdall API

## PROJECT

[Changelog](#)

# Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](#), and this project adheres to [Semantic Versioning](#).

## Unreleased

### 0.2.0 - 2026-08-27

#### Note

The Ramsey node now defaults to an X90-Y90 pulse sequence, which makes the sign of the detuning recoverable ([!83](#)). Datasets recorded before this change used X90-X90; re-analyse them by setting the node's `analysis_sequence` parameter to `x90_x90`.

#### Added

- T1 node, plus the calibration-points refactor it depends on ([!91](#)).
- T2 dephasing ([!95](#)) and T2 echo ([!105](#)) nodes.
- AllXY gate-calibration diagnostic node ([!106](#)).
- Single-qubit ([!92](#)), two-qubit ([!98](#)) and purity ([!118](#)) randomized benchmarking nodes.
- CZ nodes: chevron ([!96](#)), phases ([!97](#)), wall finder ([!119](#)), standard error amplification ([!130](#)) and amplitude-duration ([!131](#)).
- Multistate time-of-flight measurement; readout weights can be stored in Redis ([!150](#)).
- Optional circularity-based punchout endpoint, with `circularity_threshold` ([!65](#)).
- Device wiring maps (`wiring.yaml`) and an Interactive Device Wiring report section ([!53](#)).

## ON THIS PAGE

Unreleased  
0.2.0 - 2026-08-27  
    Added  
    Changed  
    Deprecated  
    Fixed  
0.1.0 - 2026-07-24  
    Changed

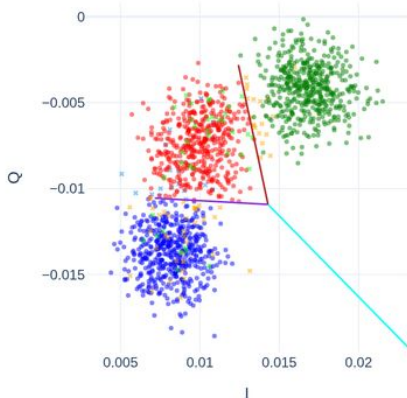
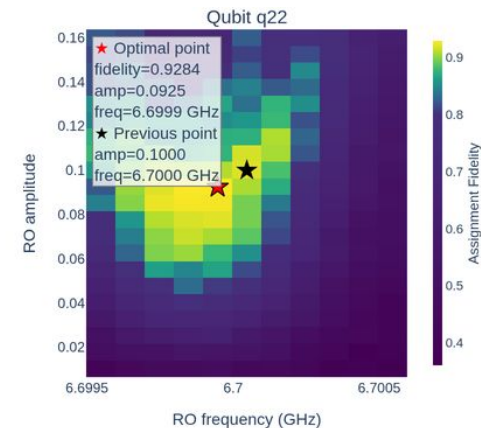
# Example: Multi-state TOF



(Thank you Emil for the plot)

```
--
32 from sindri.abstract.measurement import BaseMeasurement
33
34
35 class MultistateTOFMeasurement(BaseMeasurement):
36     """
37     Measurement class for the multi-state TOF calibration.
38
39     A ``Trace`` acquisition allows only a single acquisition per channel, so
40     this builds a schedule for exactly *one* prepared state per call – see
41     ``MultistateTOF.outer_schedule_samplespace``, which drives one compiled
42     schedule (and hardware execution) per state instead of sweeping ``state``
43     within a single schedule.
44     """
45
46     @property
47     @override
48     def measure_gate_class(self) -> Measure:
49         return Measure
50
51     @override
52     def schedule_function(
53         self,
54         *,
55         state: dict[str, int],
56         integration_time: float,
57         repetitions: int,
58     ) -> TimeableSchedule:
59         prepared_state = int(next(iter(state.values())))
60         acq_durations = [integration_time] * len(self.transmons)
61
62         schedule = TimeableSchedule("multistate_tof", repetitions=repetitions)
63         schedule.add(self.reset_system(method="passive"))
64         if prepared_state:
65             schedule.add(self.excite_system(to_state=prepared_state))
66         schedule.add(
67             self.measure_system(
68                 coords={},
69                 bin_mode="average_append",
70                 acq_protocol="Trace",
71                 acq_duration=acq_durations,
72             )
73         )
74
75     return schedule
76
```

# Example: Single shot measurement readout opt.



```
@property
@override
def measure_gate_class(self) -> Measure:
    match self.ro_clock:
        case ".ro_2st_opt":
            return Measure_R0_2state_Opt
        case ".ro_3st_opt":
            return Measure_R0_3state_Opt
        case _:
            raise NotImplementedError(f"{self.ro_clock}")

@override
def point_schedule(
    self,
    shot_index: Variable,
    ro_amplitude: list[Variable],
    ro_frequency: list[Variable],
    /,
) -> TimeableSchedule:
    schedule = TimeableSchedule("point_schedule")

    for state in self.states:
        prep_ref = schedule.add(self.reset_system(method="passive"))
        for transmon in self.transmons.values():
            if state >= 1:
                schedule.add(X(transmon.name), ref_op=prep_ref)
            if state == 2:
                schedule.add(Rxy_12(transmon.name))

        schedule.add(
            self.measure_system(
                coords={
                    "qubit_states": int(state),
                    "shot_index": shot_index,
                    "ro_amplitudes": ro_amplitude,
                    "ro_frequencies": ro_frequency,
                },
                bin_mode="append",
                pulse_amp=ro_amplitude,
                freq=ro_frequency,
            )
        )

    schedule.add(self.reset_system(method="passive"))
    return schedule
```

```
@override
def schedule_function(
    self,
    *,
    ro_amplitudes: dict[str, np.ndarray],
    ro_frequencies: dict[str, np.ndarray],
    qubit_states: dict[str, np.ndarray],
    repetitions: int,
) -> TimeableSchedule:

    self.states = [int(s) for s in next(iter(qubit_states.values()))]
    match max(self.states):
        case 1:
            self.ro_clock = ".ro_2st_opt"
        case 2:
            self.ro_clock = ".ro_3st_opt"
        case _:
            raise NotImplementedError(f"states {self.states} not supported")

    schedule = TimeableSchedule(f"ro_combined_optimization{self.ro_clock}")
    schedule.add_resources(
        [
            ClockResource(name=name + self.ro_clock, freq=ro_frequencies[name][0])
            for name in self.transmons
        ]
    )

    amplitude_domains = self.make_linear_domains(ro_amplitudes, DType.AMPLITUDE)
    frequency_domains = self.make_linear_domains(ro_frequencies, DType.FREQUENCY)
    shot_index_domain = self.make_shot_indices(repetitions)

    with schedule.loop(shot_index_domain) as shot_loop:
        schedule.add(
            self.generic_2d_sweep(
                outer_linear_domains=amplitude_domains,
                inner_linear_domains=frequency_domains,
                point_schedule=self.point_schedule,
                shot_index=shot_loop,
            )
        )

    return schedule
```

# Samplespaces



- Inner variables can be put in Q1ASM registers.
- Outer variables require schedule building.
- External variables by definition just repeat the schedule, they sweep something else entirely.

EXTERNAL external\_samplespace

COMPILE ONCE · SET OUTSIDE

The value **never reaches the schedule builder**: a DC flux current, or simply time passing. One schedule is compiled up front and re-run at every point, after **pre\_measurement\_operation** sets the parameter outside it.

OUTER outer\_schedule\_samplespace

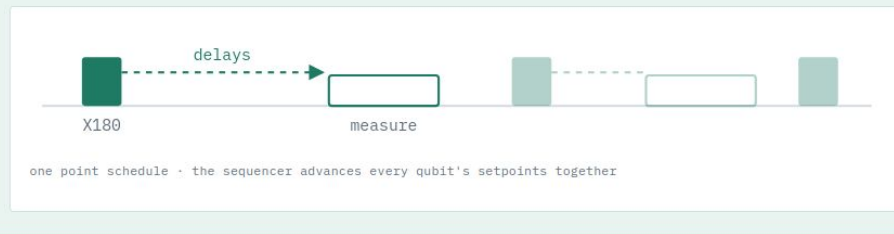
REBUILD + RECOMPILE EACH STEP

The point is **passed to the schedule builder**, then the schedule is rebuilt and recompiled — for values that change the structure of the sequence, like a new Clifford sequence per RB seed. One build and one compile per iteration.

INNER schedule\_samplespace

HARDWARE REGISTER LOOP

The default. Setpoints become **Variables** in one **sched.loop()**, so the sequencer walks the sweep itself — one point schedule, one compile, no host round-trips.



external + outer in the same node: untested (#246)

# Node variants



VARIANT = TRUE

bin\_mode

build sweep · compile · execute

build sweep · compile · execute

if bin\_mode == ... "average\_append"

average\_append

"append"

append

apply QOI · write back

Two different end-point algorithms: a fit on the hardware-averaged frequency shift, or the circularity of a single-shot IQ cloud. Different lines, different result, its own name.

PLAIN FIELD

repetitions

repetitions

repetitions: 800 → 4000

repetitions

same lines, every time

More averaging, cleaner data, no branch. Every line that ran before still runs, in the same order, with one number substituted.



Test summary: no changed test results, **941** total tests

[Full report](#)

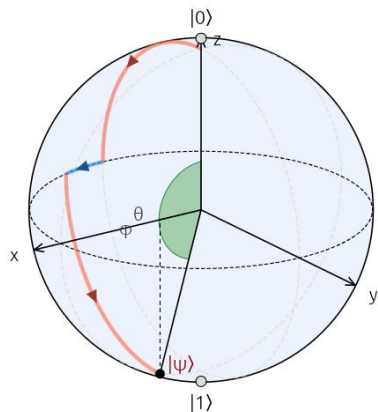
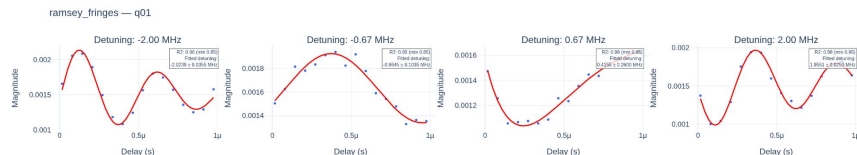
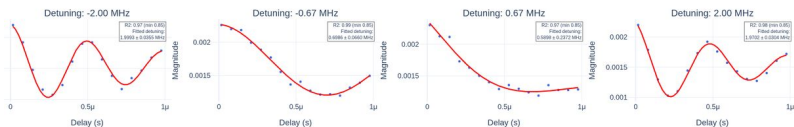


# Example: Ramsey correction variants



## Note

The Ramsey node now defaults to an X90-Y90 pulse sequence, which makes the sign of the detuning recoverable (!83). Datasets recorded before this change used X90-X90; re-analyse them by setting the node's [analysis\\_sequence](#) parameter to `x90_x90`.



Rotation sequence  
Composed from  $|0\rangle$ , applied in order:

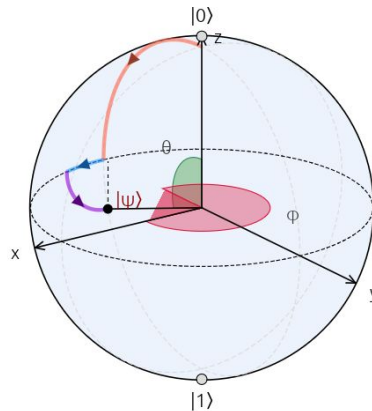
► Play ■ Stop Clear +5 random Repeat

1. Rx(+90.0°) x
2. Rz(+17.0°) x
3. Rx(+90.0°) x

Quick add (Clifford angles):  
X90 X90 X180 Y90 Y90 Y180 Z90 Z90 Z180

Add rotation:  
Rx ▼ angle=90°  
+ Add

Saved sequences:  
name Save  
x90\_x90 ▼ Restore  
Export.gif  
25 fps 720x720 ▼



Rotation sequence  
Composed from  $|0\rangle$ , applied in order:

► Play ■ Stop Clear +5 random Repeat

1. Rx(+90.0°) x
2. Rz(+17.0°) x
3. Ry(+90.0°) x

Quick add (Clifford angles):  
X90 X90 X180 Y90 Y90 Y180 Z90 Z90 Z180

Add rotation:  
Rx ▼ angle=90°  
+ Add

Saved sequences:  
name Save  
(select) ▼ Restore  
Export.gif  
25 fps 720x720 ▼

# Joel

1. My first standalone experiment
2. Custom parameters for one node
3. Re-analysis on Heimdall
4. Sweep of a node multiple times (keeping the initial state unchanged, look into contexts)
5. Unfolding a 1q bringup chain: *stress that this can be implemented in so many ways* → This will easily be extended creating a clickable tool from Heimdall (maybe)
6. A couple of words on the 2q bringup

# Calibration chains

Joel





# Schedule your own measurements

- We have gotten some insight into what kinds of nodes there are 😊
- How does one run any measurement?
- In this demo I will show
  - How can you run a single measurement?
  - How can you rerun a series of measurements using the same node?
  - How does a calibration schedule look like?

# Normal T1 Measurement

```
38 Axel Andersson, 2 weeks ago | 2 authors (Axel Andersson and one other)
39 class T1Parameters(BaseParameters):
40     """Parameters for T1 relaxation-time characterization."""
41
42     transition: Literal["01"] = acquisition_field(default="01", variant=True)
43
44     # Delays (s) between the excitation pulse and the readout.
45     delays: list[float] = acquisition_field(
46         default_factory=default_t1_delays,
47         description="Delays between excitation and readout, in seconds.",
48     )
49
50     repetitions: int = acquisition_field(
51         default=1024,
52         description="Number of hardware-averaged shots per delay point.",
53     )
54
55     t1_repetitions: int = acquisition_field(
56         default=3,
57         description=(
58             "Number of independent repeats of the full delay sweep (and its "
59             "own calibration points). T1 is fitted separately per repeat; the "
60             "reported t1 is the mean across repeats and t1_error is their "
61             "standard deviation."
62         ),
63     )
64
```

```
23
24 import logging
25 import numpy as np
26
27 from bragi.handlers.data import RedisDataHandler
28 from bragi.orchestra import CalibrationOrchestrator
29 from saga.data_managing.session import SessionOptions
30 from sindri.nodes.characterization.t1.node import T1
31 from sindri.nodes.characterization.t1.parameters import T1Parameters, T1QOI
32
33 # Silence noisy third-party loggers - keep only WARNING and above.
34 for _noisy in ("qcodes", "qblox_scheduler", "quantify_scheduler", "quantify_core"):
35     logging.getLogger(_noisy).setLevel(logging.WARNING)
36
37
38 # Insert your Redis port here
39 REDIS_PORT = 6381
40 TEMPLATE_NAME = "fc8b"
41
42 QUBITS = ["q17", "q18"]
43
44 # Execution mode for the session: "measurement" drives real hardware; "dummy"
45 # simulates the cluster, useful for a dry run against test_chip.
46 EXECUTION_MODE = "measurement"
47
48 # Set to True so the orchestrator reloads all upstream results (readout
49 # amplitude, f01, ...) from Redis before running this session.
50 LOAD_CALIBRATION_STATE: bool = True
51
52 orchestrator = CalibrationOrchestrator(
53     experiment_template_name=TEMPLATE_NAME,
54     data_handler=RedisDataHandler(redis_port=REDIS_PORT),
55     load_calibration_state=LOAD_CALIBRATION_STATE,
56 )
57
58 t1_params = T1Parameters(
59     name="t1",
60     qubits=QUBITS,
61 )
62
63 with orchestrator.calibration_session(
64     SessionOptions(session_name="t1", execution_mode=EXECUTION_MODE)
65 ):
66     t1_result = orchestrator.run_node(
67         T1(
68             parameters=t1_params,
69             qoi_cls=T1QOI,
70         )
71     )
72
73 # Generate master HTML with all sessions
74 orchestrator.generate_html_report()
```



# I don't want a normal T1

- If you want to change some parameters for experiment, you can do it directly in your code

```
57
58 t1_params = T1Parameters(
59     name="t1",
60     qubits=QUBITS,
61     delays=np.linspace(0, 1000e-6, 31).tolist(),
62     repetitions=10,
63     t1_repetitions=100
64 )
65
```

```
38
39 class T1Parameters(BaseParameters):
40     """Parameters for T1 relaxation-time characterization."""
41
42     transition: Literal["01"] = acquisition_field(default="01", variant=True)
43
44     # Delays (s) between the excitation pulse and the readout.
45     delays: list[float] = acquisition_field(
46         default_factory= default_t1_delays,
47         description="Delays between excitation and readout, in seconds.",
48     )
49
50     repetitions: int = acquisition_field(
51         default=1024,
52         description="Number of hardware-averaged shots per delay point.",
53     )
54
55     t1_repetitions: int = acquisition_field(
56         default=3,
57         description=(
58             "Number of independent repeats of the full delay sweep (and its "
59             "own calibration points). T1 is fitted separately per repeat; the "
60             "reported t1 is the mean across repeats and t1_error is their "
61             "standard deviation."
62         ),
63     )
64
```

## Files

Search files (\*.vue, \*.rb...)

- device\_templates
  - docs
  - examples
    - advanced\_scripts
    - analysis
    - introduction
    - my\_first\_resonator\_spectroscopy.py**
    - using\_custom\_parameters.py
    - pb\_sos1\_v1
    - single\_qubit\_bringup
    - two\_qubit\_bringup
    - vibe\_calibration\_mode
  - README.md
- packages
- tests
- utils\_scripts
- .gitignore
- .gitlab-ci.yml
- .pre-commit-config.yaml
- .python-version
- CHANGELOG.md
- README.md

Provide feedback

```

24
25
26 from bragi.handlers.data import RedisDataHandler
27 from bragi.orchestra import CalibrationOrchestrator
28 from saga.data_managing.session import SessionOptions
29 from sindri.nodes.readout.resonator_spectroscopy.node import ResonatorSpectroscopy
30 from sindri.nodes.readout.resonator_spectroscopy.parameters import (
31     ResonatorSpectroscopyParameters,
32     ResonatorSpectroscopyQOI,
33 )
34
35 # Insert your Redis port here
36 REDIS_PORT = 0000
37 RUN_NAME = "1st resonator spectroscopy"
38
39 QUBITS = [f"q{i:02d}" for i in range(16, 26)]
40 SAMPLES = 101
41 HALF_WIDTH = 1e6 # Hz - matches resonator_spectroscopy_half_width in device.yaml
42
43 orchestrator = CalibrationOrchestrator(
44     experiment_template_name="fc8b",
45     data_handler=RedisDataHandler(redis_port=REDIS_PORT),
46     name=RUN_NAME,
47 )
48
49 res_spec_params = ResonatorSpectroscopyParameters(
50     name="resonator_spec",
51     qubits=QUBITS,
52     readout_state=0,
53     samples=SAMPLES,
54     half_width=HALF_WIDTH,
55 )
56
57 with orchestrator.calibration_session(
58     SessionOptions(session_name="resonator_spectroscopy")
59 ):
60     orchestrator.run_node(
61         ResonatorSpectroscopy(
62             parameters=res_spec_params,
63             qoi_cls=ResonatorSpectroscopyQOI,
64         )
65     )
66

```

# Testing different parameters for the same measurement

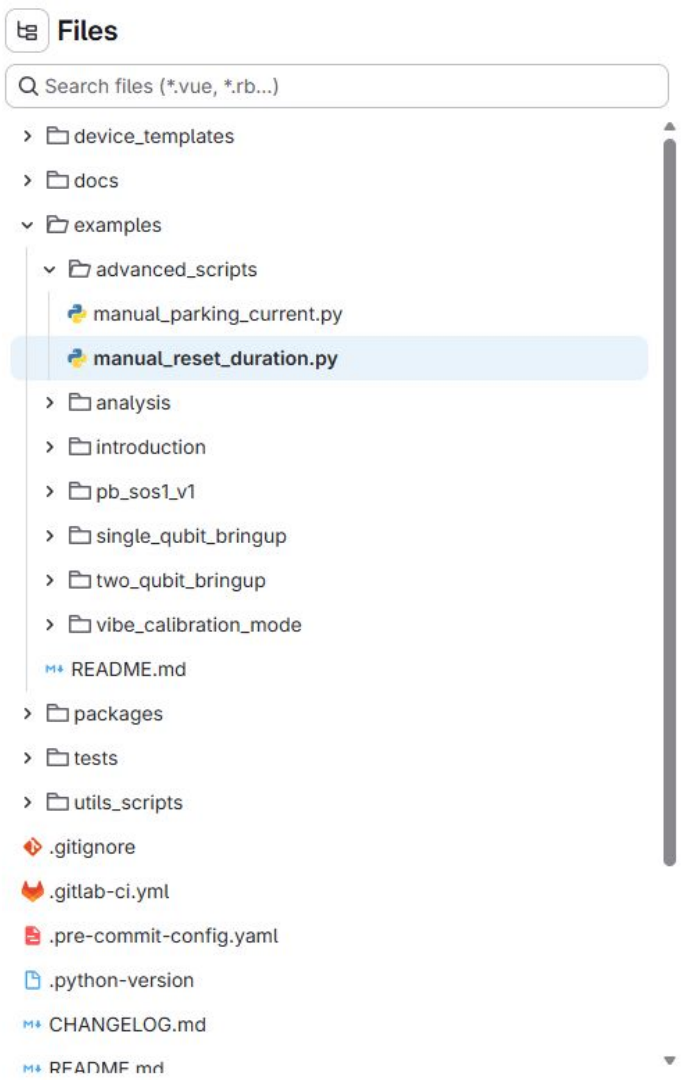
- Different node parameters
- Testing different attenuations
- Etc.



```
24 import logging
25
26
27 from bragi.handlers.data import RedisDataHandler
28 from bragi.orchestra import CalibrationOrchestrator
29 from saga.data_managing.session import SessionOptions
30 from saga.utils.context import preserve_device_state, temporary_output_attenuation
31 from sindri.nodes.qubit_control.spectroscopy.node import QubitSpectroscopy
32 from sindri.nodes.qubit_control.spectroscopy.parameters import (
33     QubitSpectroscopy01QOI,
34     QubitSpectroscopyParameters,
35 )
36
37
38 for _noisy in ("qcodes", "qblox_scheduler", "quantify_scheduler", "quantify_core"):
39     logging.getLogger(_noisy).setLevel(logging.WARNING)
40
41
42 REDIS_PORT = 6381
43 TEMPLATE_NAME = "fc8b"
44
45 QUBITS = ["q17", "q18"]
46
47 EXECUTION_MODE = "measurement"
48
49 LOAD_CALIBRATION_STATE = bool = True
50
51 ATTENUATIONS_DB = [0, 2, 4, 6, 8, 10]
52
53
54 orchestrator = CalibrationOrchestrator(
55     experiment_template_name=TEMPLATE_NAME,
56     data_handler=RedisDataHandler(redis_port=REDIS_PORT),
57     load_calibration_state=LOAD_CALIBRATION_STATE,
58 )
59
60 with orchestrator.calibration_session(
61     SessionOptions(session_name="spectroscopy_attenuation_sweep", execution_mode=EXECUTION_MODE)
62 ):
63     with preserve_device_state(["clock freqs.f01"]):
64         for attenuation in ATTENUATIONS_DB:
65             overrides = {f"{qubit}:mw-{qubit}.01": attenuation for qubit in QUBITS}
66             with temporary_output_attenuation(overrides):
67                 orchestrator.run_node(
68                     QubitSpectroscopy(
69                         parameters=QubitSpectroscopyParameters(
70                             name=f"qubit_spectroscopy_01_att{attenuation}dB",
71                             qubits=QUBITS,
72                             transition="01",
73                         ),
74                         qoi_cls=QubitSpectroscopy01QOI,
75                     )
76                 )
77
78 orchestrator.generate_html_report()
```

# I don't believe in magic

You can do it yourself



# Custom parameters

## Files

Search files (\*.vue, \*.rb...)

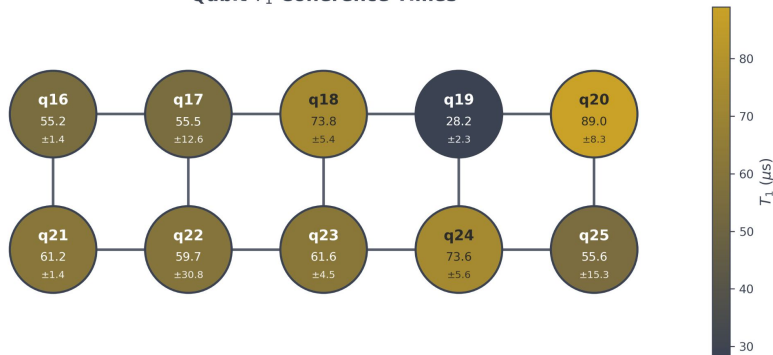
- device\_templates
- docs
- examples
  - advanced\_scripts
  - analysis
  - introduction
    - my\_first\_resonator\_spectroscopy.py
    - using\_custom\_parameters.py**
  - pb\_sos1\_v1
  - single\_qubit\_bringup
  - two\_qubit\_bringup
  - vibe\_calibration\_mode
- README.md
- packages

```
63 FAST_RESET_DURATION = 100e-6
64
65 # q18's resonator sits further from its design frequency than the rest, so both
66 # resonator-spectroscopy scans need a slightly wider search window for it, while
67 # the other qubits are fine with the parameters-class defaults.
68 CUSTOM_RESONATOR_WIDE_HALF_WIDTH = {"q18": 30e6} # default: 15e6
69 CUSTOM_RESONATOR_FINETUNE_HALF_WIDTH = {"q18": 3e6} # default: 1e6
70
71 # q18's punchout dip is also known to shift further before stabilizing, so it
72 # needs looser stopping thresholds than the rest of the qubits, which are fine
73 # with the PunchoutParameters defaults (upper=0.1e8 Hz, lower=0.05e6 Hz).
74 CUSTOM_PUNCHOUT_PARAMETERS = {
75     "upper_threshold": {"q18": 0.3e8, "q23": 0.3e8}, # default: 0.1e8
76     "lower_threshold": {"q18": 0.15e6},
77 }
78
79 orchestrator = CalibrationOrchestrator(
80     experiment_template_name=TEMPLATE_NAME,
81     data_handler=RedisDataHandler(redis_port=REDIS_PORT),
82     load_calibration_state=False,
83     name=RUN_NAME,
84 )
85
86
87 session_options = SessionOptions(
88     session_name="resonator_search_custom_parameters", execution_mode="measurement"
89 )
90 with orchestrator.calibration_session(session_options=session_options):
91     with temporary_reset_duration(FAST_RESET_DURATION):
92         orchestrator.run_node(
93             ResonatorSpectroscopy(
94                 parameters=ResonatorSpectroscopyParameters(
95                     name="resonator_spectroscopy_wide",
96                     qubits=QUBITS,
97                     readout_state=0,
98                     samples=200,
99                     half_width=15e6,
100                     custom_parameters=CUSTOM_RESONATOR_WIDE_HALF_WIDTH,
101                 ),
102                 qoi_cls=ResonatorSpectroscopyQOI,
103             )
104         )
105
106
107 orchestrator.run_node(
108     Punchout(
109         parameters=PunchoutParameters(
110             name="punchout",
111             qubits=QUBITS,
112             readout_state=0,
113             samples=30,
114             half_width=1.25e6,
115             repetitions=200,
116             custom_parameters=CUSTOM_PUNCHOUT_PARAMETERS,
117         ),
118         qoi_cls=PunchoutQOI,
119     )
120 )
121
122 orchestrator.run_node(
123     ResonatorSpectroscopy(
124         parameters=ResonatorSpectroscopyParameters(
125             name="resonator_spectroscopy_finetune",
126             qubits=QUBITS,
127             readout_state=0,
128             samples=80,
129             half_width=1e6,
130             custom_parameters=CUSTOM_RESONATOR_FINETUNE_HALF_WIDTH,
131         ),
132         qoi_cls=ResonatorSpectroscopyQOI,
133     )
134 )
```

# Bring-up scripts

## Single qubit calibration works

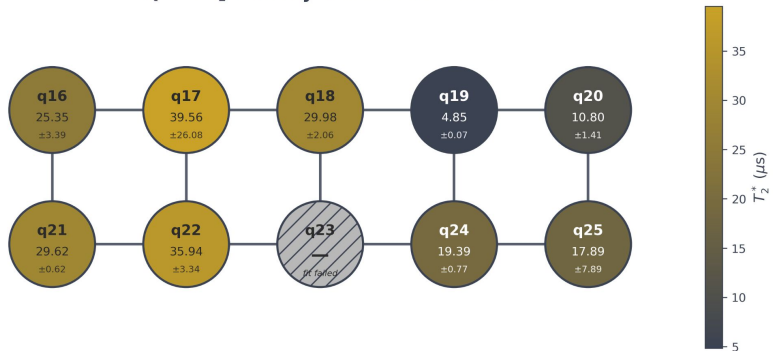
Qubit  $T_1$  Coherence Times



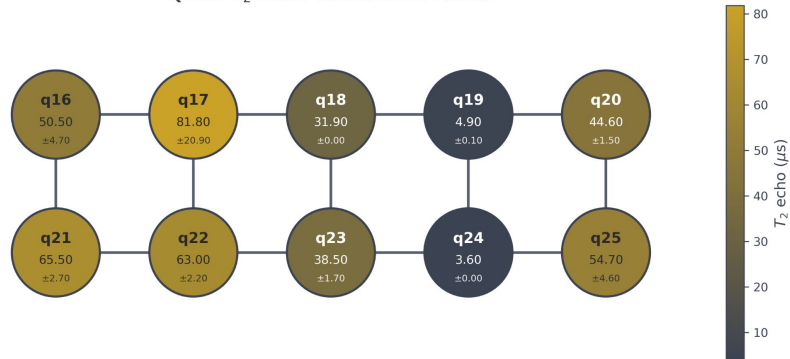
Randomized Benchmarking Fidelity



Qubit  $T_2$  Ramsey Coherence Times



Qubit  $T_2$  Echo Coherence Times



# Bring-up scripts

Two qubit calibration

Work in progress. One bug at a time



# Quantum random numbers

Yanchen



# Yanchen

## 1. RANDOM MAGIC YEE

# Quantum Random Number Game

- Quantum computers are good at generating true random numbers;

# Quantum Random Number Game

- Quantum computers are good at generating true random numbers;
- There are some quantum computers at QC2;

# Quantum Random Number Game

- Quantum computers are good at generating true random numbers;
- There are some quantum computers at QC2;
- There is a random number based game people like to play;

The screenshot shows the RNGdle game interface. At the top, the current roll is 245550, which is RARE and in the TOP 11%. The player has 20,993 EP and a lifetime total of 273,627 EP. The next roll is in 14h 50m 53s. Below this, there's a badge breakdown for 17 badges earned. The first badge is SEMI-CENTURY (RARE, NEW) worth +10,000 EP, ending in "50". The second is CONTIGUOUS TRIPS (UNCOMMON) worth +2,784 EP, with three identical consecutive digits (5, 5, 5). The third is BLACKJACK (UNCOMMON) worth +2,521 EP, where digits sum exactly to 21.

The screenshot shows the RNGdle game interface with the RANKINGS tab selected. The list shows the top players:

|   | TODAY         | THIS WEEK | ALL-TIME | BADGES                              |
|---|---------------|-----------|----------|-------------------------------------|
| 1 | GUAVEYELLSHEJ |           | 136558   | day year life, all one cycle...     |
| 2 | AXEAN         |           | 282216   | counting cards! gambled six eggs,.  |
| 3 | LINYA (YOU)   |           | 245550   | balloon rest over the gambling flat |
| 4 | SOLAR_CRAB    |           | 147141   | rest bread twin                     |
| 5 | POTATOLORD666 |           | 249789   |                                     |
| 6 | GLOOMY_TIGER  |           |          |                                     |
| 7 | FUNKY_BEAVER  |           |          |                                     |

# Quantum Random Number Game

- Quantum computers are good at generating true random numbers;
- There are some quantum computers at QC2;
- There is a random number based game people like to play;
- Why not make a similar game enabled by quantum random numbers?

# Quantum Random Number Game

- A QRNG node within the calibration framework.
  - “QRNGNode” that resets a qubit, drives the qubit to superposition, and measures.
  - Randomness metrics (debiased bit):
    - 0/1 ratio: 9951/10049 (ideally 1:1)
    - debiasing\_efficiency: 0.2487 (ideally 0.25)
    - ...
- A simple HTTP server exposing raw QRNG readout bits
- A website hosting the game
  - Gets the raw bits;
  - Debases the raw bits using Von Neumann debiasing;
  - Turn debiased bits to numbers;
  - Run rules to get final score (more special numbers get higher scores);
  - Compete on leaderboard!

# Welcome to the game

## QRNGdle

von Neumann debiasing · binary-to-decimal · rule-based scoring

Player name

yanchen

Generate quantum number

☐ Fast mode (skip step-by-step animation)

Data source

☒ Real quantum hardware ☐ Simulated (browser)

QRNG server

<http://localhost:64121>

Address of a running `raw_bits_server.py` (see `../qrng/`) — each round measures real single shots on that server's qubit.

### 1 · Debias the raw bit stream

Adjacent raw bits are paired: `01→0, 10→1, 00 / 11` discarded. A batch of raw bits arrives, then the read head slides across it two at a time.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | → | → | → | → | → | → | → | → | → | → |
| → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → |
| → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → | → |

Raw bits used **32** Discarded **12** Debiasing efficiency **31%**  
An ideal 50/50 qubit needs 40 raw bits for 10 logical bits — every raw bit your run deviates from that, in either direction, adds +5% to the final score.

Score multiplier **x1.40**

### 2 · Logical bit register (10 bits, MSB first)

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

### 3 · Binary → Decimal

|     |     |     |    |    |    |   |   |   |   |
|-----|-----|-----|----|----|----|---|---|---|---|
| 1   | 0   | 0   | 0  | 0  | 1  | 0 | 0 | 1 | 1 |
| 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

512 + 16 + 2 + 1 = **531**

# Future directions

Matteo



# Near-term goals: open issues

MC2 / QT / QC / [QC2 Calibration Toolkit](#) / **Work Items**

All items

+ Add view

🔍

State

is

Open

✕

🔍

🔍

🔍

Created date 

▼

🔍

92 items 

Bulk edit

🔖

Implement CZ gate construct

mc2/qt/qc/qc2-qcqv-suite#385

created 2 days ago by Axel Erik Andersson

Enhancement

🔖

1

🔖

1

updated 2 days ago

🔖

Dynamic promotion of measure\_gate\_class

mc2/qt/qc/qc2-qcqv-suite#384

created 5 days ago by Axel Erik Andersson

ABC Bragi Enhancement Measurement Refactor

🔖

1

updated 5 days ago

🔖

Heimdall continuous deployment for quick inspection when developing?

mc2/qt/qc/qc2-qcqv-suite#383

created 6 days ago by Axel Erik Andersson

Ci Future Heimdall

🔖

1

🔖

Save dataset to disk between external or outer batches

mc2/qt/qc/qc2-qcqv-suite#382

created 1 week ago by Axel Erik Andersson

Enhancement User experience

🔖

Run analysis between external or outer batches

mc2/qt/qc/qc2-qcqv-suite#381

created 1 week ago by Axel Erik Andersson

ABC Analysis Enhancement

🔖

1

updated 1 week ago

🔖

Improve code architecture

mc2/qt/qc/qc2-qcqv-suite#380

created 1 week ago by Axel Erik Andersson

Bragi Code maintenance Heimdall Refactor Release Saga Sindri

🔖

1

updated 5 days ago

🔖

Introduce optimizers to address calibration sessions

mc2/qt/qc/qc2-qcqv-suite#379

created 1 week ago by Matteo Robbiati

Enhancement Future New feature

🔖

2

updated 1 week ago

🔖

Write new tutorials to add to the documentation (this could help for our workshop as well)

mc2/qt/qc/qc2-qcqv-suite#378

created 1 week ago by Matteo Robbiati

Documentation

🔖

Resolve all the remaining FIXME comments in the code

mc2/qt/qc/qc2-qcqv-suite#376

created 2 weeks ago by Axel Erik Andersson

Code maintenance Release

🔖

1

updated 2 weeks ago

🔖

Resolve all remaining TODO comments in code

mc2/qt/qc/qc2-qcqv-suite#375

created 2 weeks ago by Axel Erik Andersson

Code maintenance Release

🔖

1

updated 2 weeks ago

🔖

New nodes from quantify pingu

mc2/qt/qc/qc2-qcqv-suite#374

created 2 weeks ago by Emil Hogedal

Code migration Measurement New feature Sindri

🔖

1

updated 1 week ago

✓

Add timing analysis for a node/calibration run

mc2/qt/qc/qc2-qcqv-suite#373

created 2 weeks ago by Yanchen Lin

Enhancement Future Performance

🔖

1

🔖

1

🔖

1

updated 6 days ago

✓

Add a session option to skip static image generation

mc2/qt/qc/qc2-qcqv-suite#372

created 2 weeks ago by Yanchen Lin

Enhancement Future Performance

🔖

2

🔖

1

updated 2 weeks ago

🔖

Automatic REDIS managing service

mc2/qt/qc/qc2-qcqv-suite#371

created 2 weeks ago by Axel Erik Andersson

Enhancement Future Measurement New feature User experience

🔖

1

updated 2 weeks ago

🔖

Show on hover descriptions for all analysis\_parameters in Heimdall page

mc2/qt/qc/qc2-qcqv-suite#368

created 2 weeks ago by Axel Erik Andersson

Analysis Live-show-webpage User experience

🔖

1

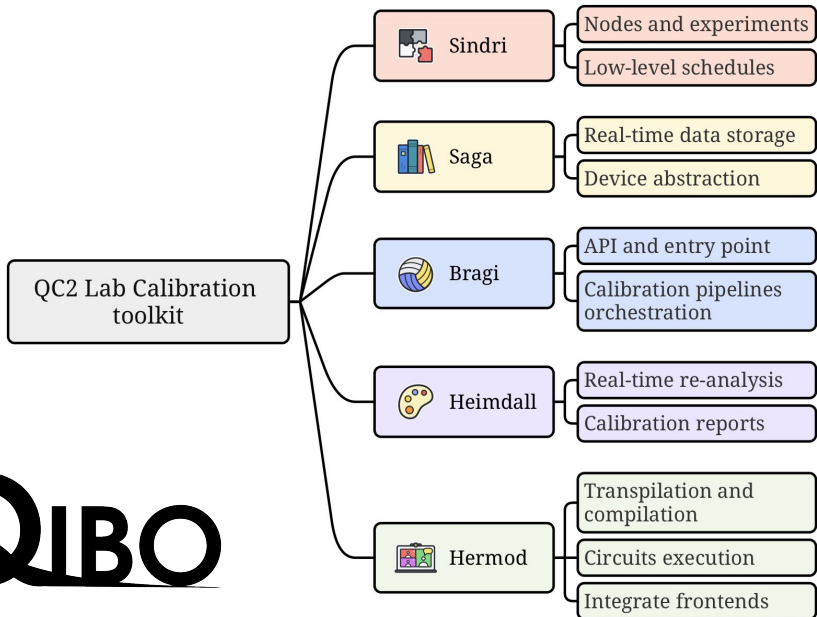
updated 2 weeks ago

# Near-term goals: run circuits in the lab

- We want to run benchmarks, research algorithms, high level error correction and mitigation schemes.
- We are agnostic w.r.t. chosen language or frontend
- Hermod can take care of this conversion.
- Prototype is running.

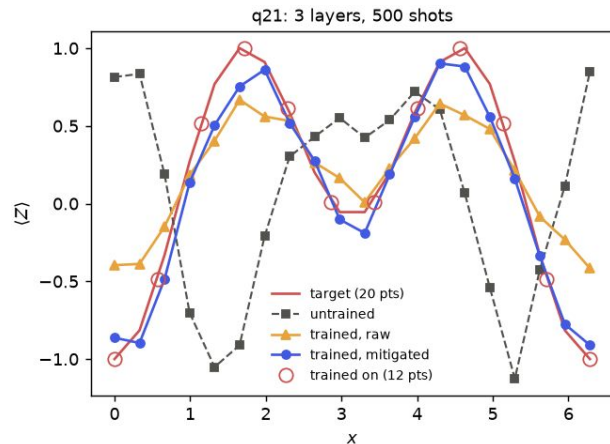
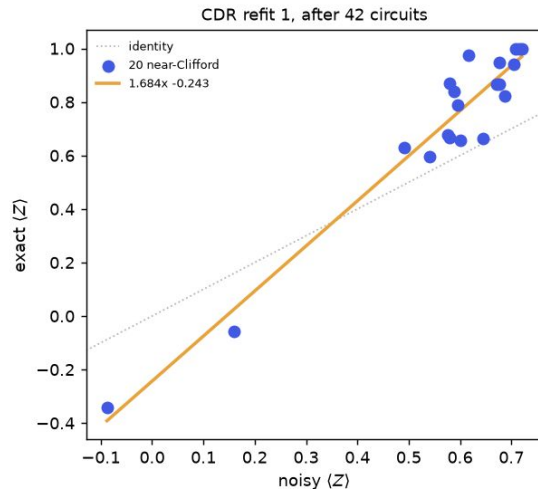
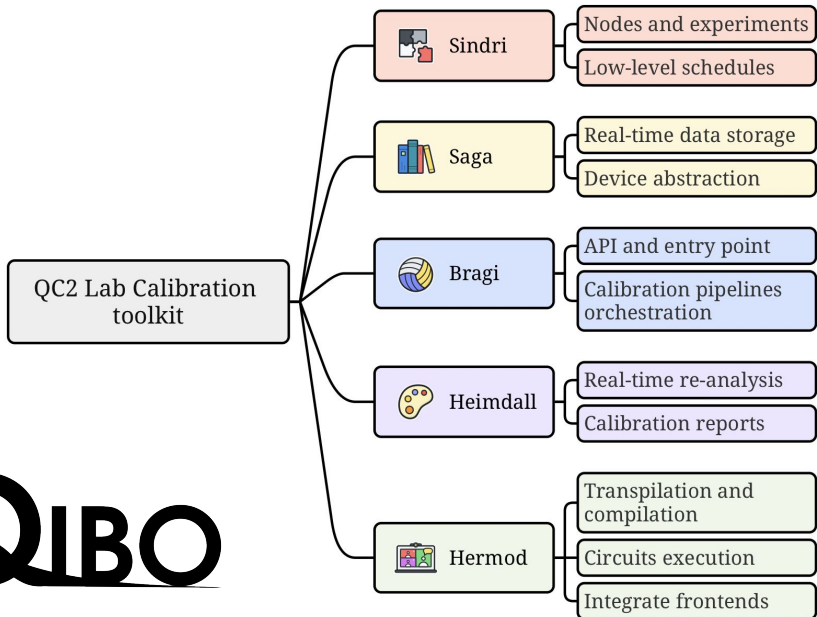
# Near-term goals: run circuits in the lab

- We want to run benchmarks, research algorithms, high level error correction and mitigation schemes.
- We are agnostic w.r.t. chosen language or frontend
- **Hermod can take care of this conversion.**
- Prototype is running.

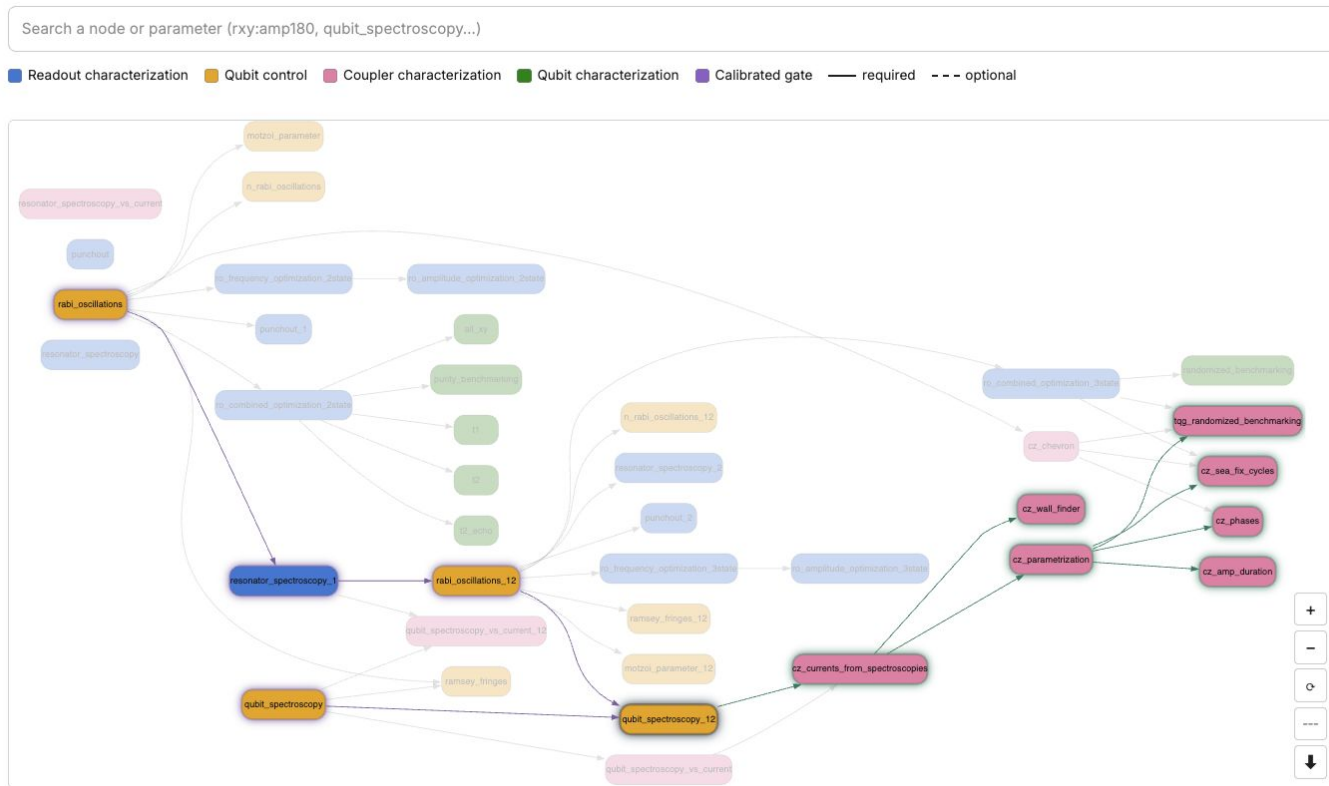


# Near-term goals: run circuits in the lab

- We want to run benchmarks, research algorithms, high level error correction and mitigation schemes.
- We are agnostic w.r.t. chosen language or frontend
- Hermod can take care of this conversion.
- **Prototype is running.**



## Near-term goals: requirements [WIP] and optimizers



# Near-term goals: requirements [WIP] and optimizers



## GETTING STARTED

[Installation](#)[Architecture](#)[Core Concepts](#)

## USING THE TOOLKIT

[Running Experiments](#)[Measurement Helpers](#)[The Replot Server \(Heimdall\)](#)

## NODE LIBRARY

[Node Registry](#)[Dependency Graph](#)[Node Library](#)

- [Readout characterization](#)
- [Qubit control](#)
- [Qubit characterization](#)
- [Coupler characterization](#)

## TUTORIALS

[Tutorial: Resonator Spectroscopy](#)

## API REFERENCE

## Calibration parameters

Shows all Redis fields.

☐ read here ☐ fine-tuned (read & calibrated) here ☐ calibrated here ☐ calibrated upstream

[Detailed](#)[Only relevant](#)

### qubit\_spectroscopy Qubit control

[View full documentation](#)[8 calibrated fields by 1 ancestor](#)[0 read here](#)[2 calibrated here](#)

| PARAMETER                    | HASH              | ROLE                | CALIBRATED BY          | NOTE |
|------------------------------|-------------------|---------------------|------------------------|------|
| CLOCK_FREQS — ELEMENT CLOCKS |                   |                     |                        |      |
| clock_freqs:readout          | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| clock_freqs:f01              | transmons:{qubit} | calibrated here     |                        |      |
| SPEC — SPECTROSCOPY DRIVE    |                   |                     |                        |      |
| spec:spec_ampl_optimal       | transmons:{qubit} | calibrated here     |                        |      |
| RESONATOR FIT DIAGNOSTICS    |                   |                     |                        |      |
| Ql                           | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| Qe                           | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| theta                        | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| r_squared                    | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| resonator_fit                | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| kappa                        | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |
| propagation_delay            | transmons:{qubit} | calibrated upstream | resonator_spectroscopy |      |

[↑ Back to top](#)

# Towards a quantum OS

- Data sharing
- Access to diagnostic data from fridge and electronics in real-time
- Job scheduling and managing
- Heimdall will have to load from and save to databases

Ehj Matteo, I just calibrated fc8b

You can find the values in my Redis port



You mean our Redis port



# Much more is coming (**mentioned** and **new**)

1. 90+ **work items** open on GitLab
2. Quantum circuit execution via Hermod / frontend
3. Defining **requirements** between nodes introducing **optimizers**
4. Little mono-repo restructuring: *is everything in the right place? E.g. models.*
5. Adopting more advanced **measurements** solutions (see Qblox's workshop)
6. Boosting 2QG calibration: *lightening of the working points exploration*
7. Improving the visualization tools: *Heimdall can get even better*
8. Towards a quantum OS: testing JuiceOS? Grafana? Common database?
9. Job managing and scheduling?
10. Toward QEC studies, what is missing?

# New ideas and priorities

1. Simulation (emulation) of the system evolution: typically 1-2 qubits and one coupler involved: *a proper package in the monorepo can emulate the output of the current nodes*
2. Extending the database list: currently Redis, next Snapshots (json or yaml), others.
3. Improve tools to characterize noise and integrate noise mitigation and correction routines
4. ACCESS and LICENSING: waiting for a feedback to understand how to match teacher exemption and open-source
5. Integrating new SPI rack handling
6. Implementation of logical qubits (API)
7. We need to prepare the hardware to support higher-level QEC routines (feedback loops, etc)
8. Demonstrate QCT by keeping quantum memory over time
9. Real time error mitigation *on board*

# Publication strategy

Michele



# Publication strategy: Journal

Which journal? A few options:

- Quantum
- Quantum Science and Technology
- PRX Quantum
- EPJ-C

# Publication strategy: Author list

Inclusive or restricted?

Most inclusive (preferred solution):

- anyone that contributed to the code (including Tergite-autocalibration),
- the managers of the sw team
- the PIs that secured the funding.
- People can opt out.

Possible restrictions (for discussion only, not endorsed):

- Set a threshold on contribution quantity and quality
- Set a time limit on contribution and on association to the group
- Only direct contributions to QCT

Association:

Double association for those at CNL that contributed before moving there.

# Publication strategy: order and editors

Alphabetic order for everyone with first authors for main contributors and last position for PIs

Long paper, cannot be written by a single person but someone (Matteo?) need to be in charge for coordinating and uniforming the writing. Recurring meetings to ensure it is done