Qiskit in the OpenSuperQ project

Moritz Kirste and Daniel Weigand

www.opensuperq.eu

The OpenSuperQ project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 820363.

Goals of the OpenSuperQ project

Build a sustainable central quantum computing laboratory

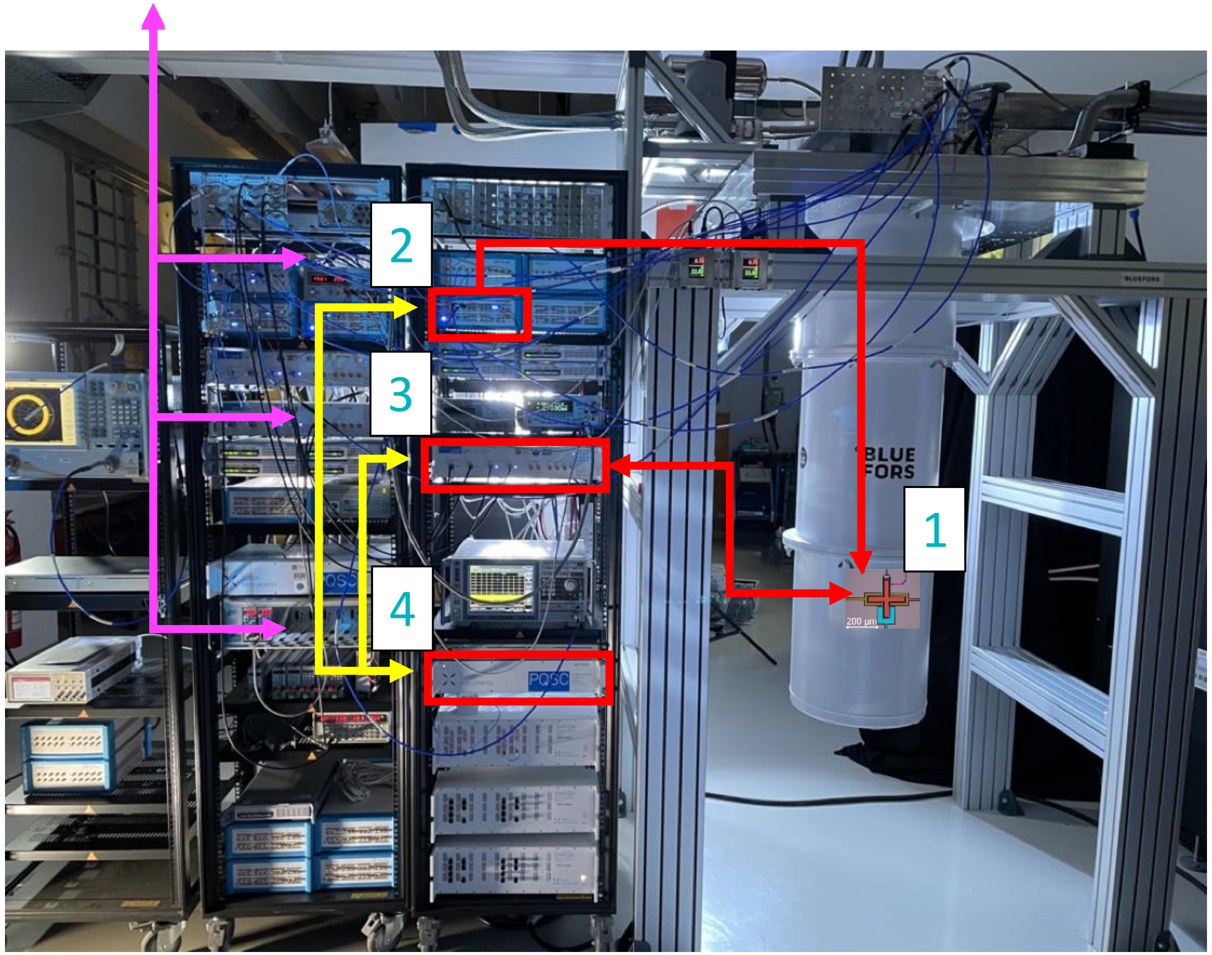
Develop Quantum Processing Unit 1

Develop Room Temperature Electronics

- Device for Control Pulses 2
- Device for Readout 3
- Device for Synchronization and Feedback 4

Develop Software Stack (talk today)

Software Stack



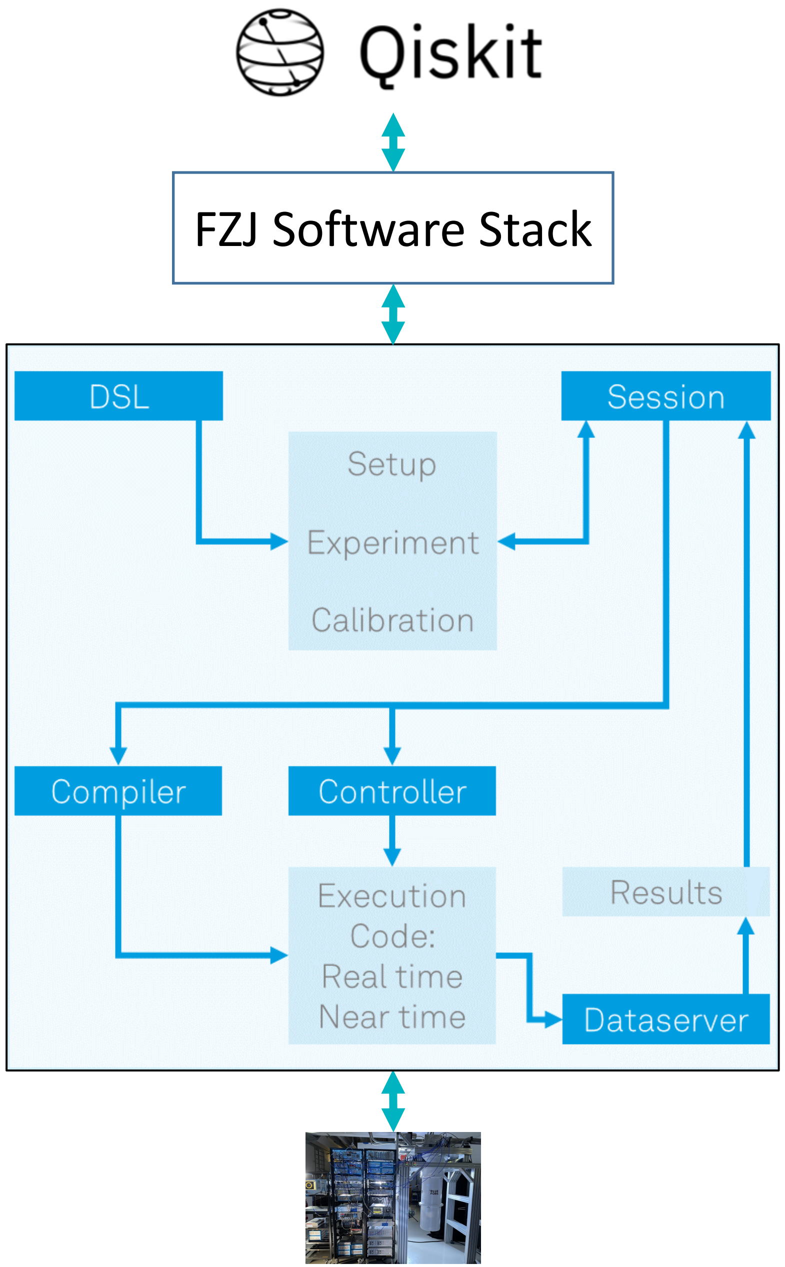
OpenSuperQ Software Stack

Qiskit as the high-level component

Zurich Instruments Software

- Pulse-level interface parallel to Qiskit pulse
- Hide complexity but preserve full hardware capability and transparency

Forschungszentrum Jülich provides interface



Qiskit

FZJ Software Stack

DSL

Session

Setup

Experiment

Calibration

Compiler

Controller

Execution Code: Real time Near time

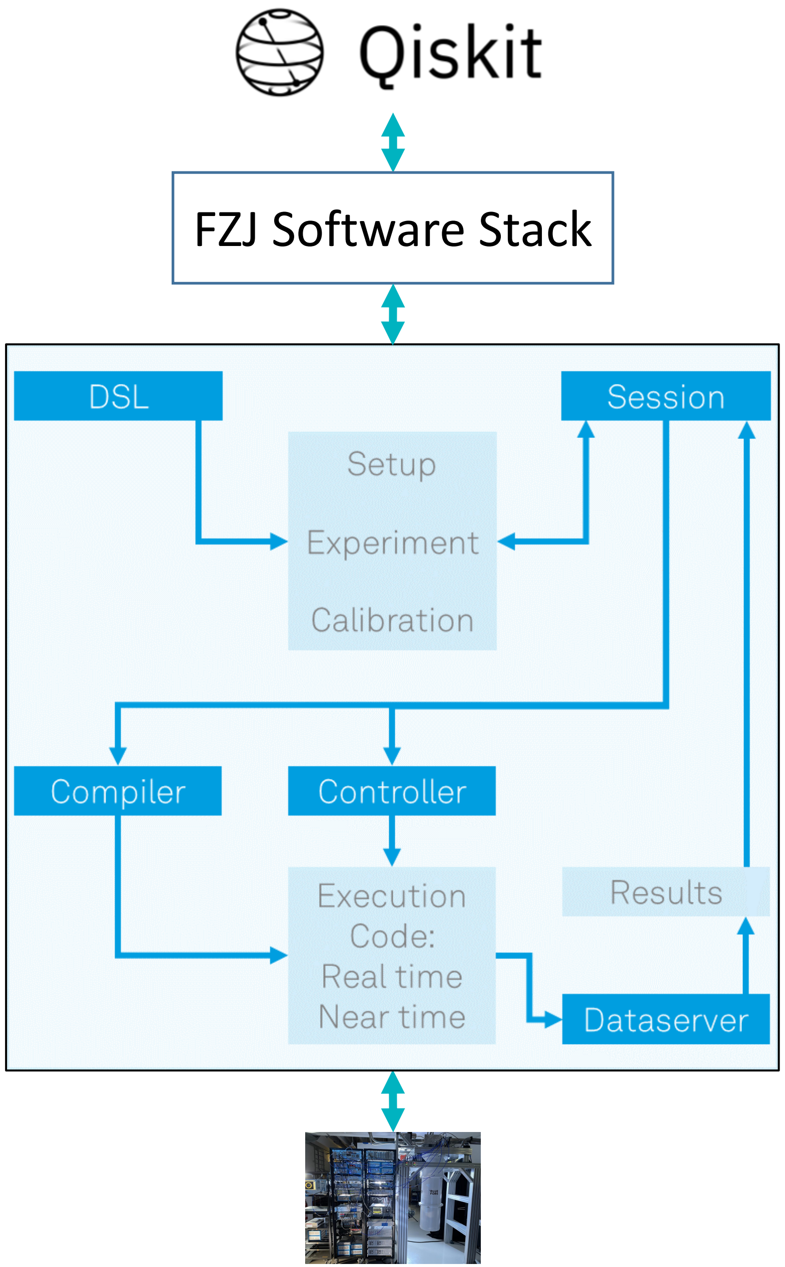
Results

Dataserver

OpenSuperQ Software Stack

Challenges

- Waveform memory and instruction memory are limited
- Fast changes of circuits, gates, pulses, parameters and waveforms require custom designed hardware and software
- Representation and abstraction of complexity



Qiskit

FZJ Software Stack

DSL

Session

Setup

Experiment

Calibration

Compiler

Controller

Execution Code:
Real time
Near time

Results

Dataserver

Quantum Cloud API

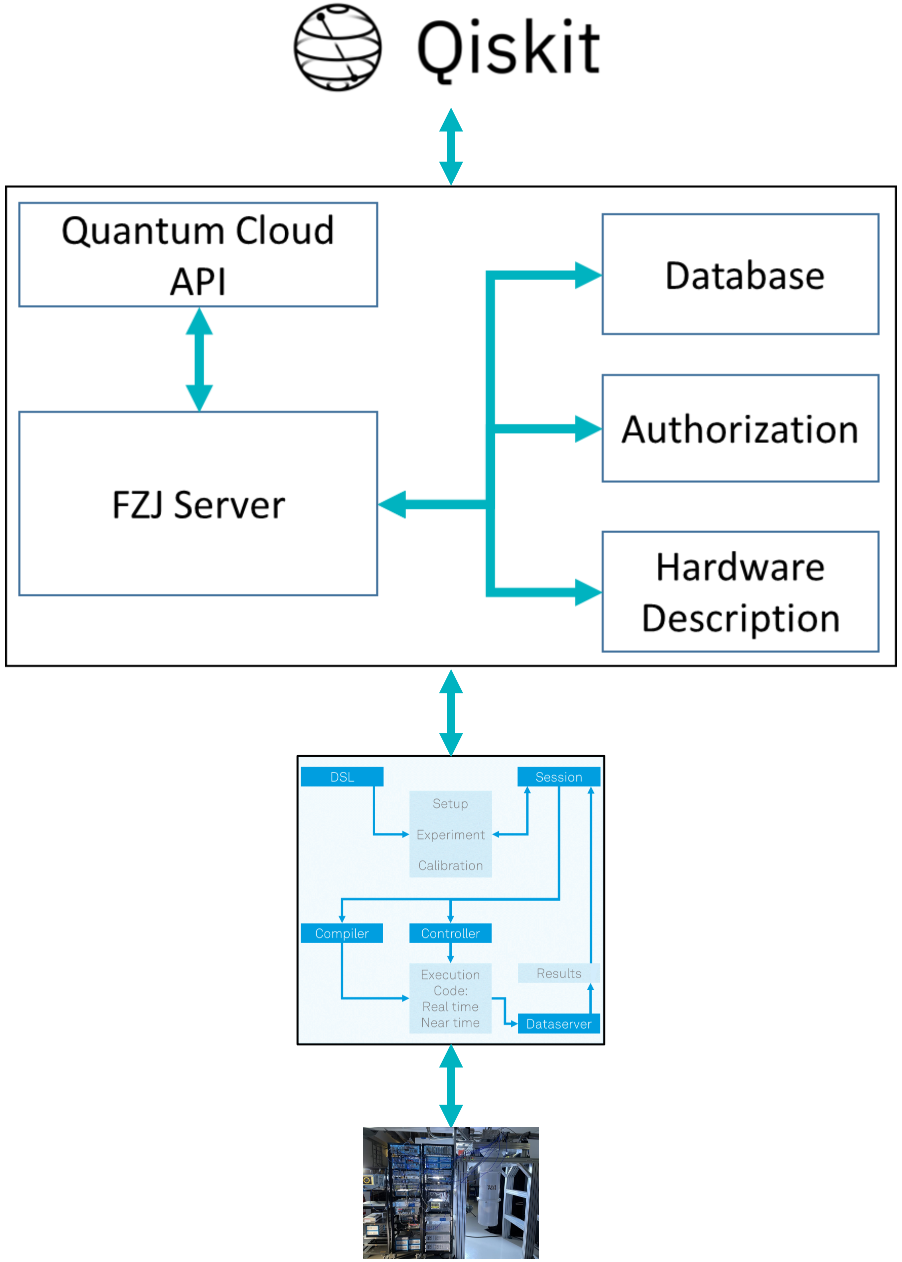
Design goals

- Unified API for internal and external users with optional low level hardware access
- Mixed abstraction levels (circuits, schedules and pulses)

Chose expanded Qiskit API over Qiskit Pulse

- Pulse does not reflect all control hardware capabilities

➢ Walkthrough example



Walkthrough example: Amplitude Rabi

Let’s start at the top level: Qiskit

- Custom gates are syntax sugar for `qiskit.circuit.Gate(name, num_qubits, params)`

```
# Loops for averaging, sweep
qc.append(QcAcquireLoop(1024, "cyclic"), [0])
qc.append(QcSweepRealTime("key", (0.0, 0.5, 1.0)), [0]) # sweep "key" over (0.0, 0.5, 1.0)

# Body
qc.append(QcGate("rx", 1, ["key"]), [0])                # qc.rx("key", 0)
qc.measure_all()

# Close loops
qc.append(QcCloseLoop(), [0])
qc.append(QcCloseLoop(), [0])
```

Intermediate representation

- Gate names are used to pass information to backend `'attribute_qubit_frequency'`
- Parameters do not conform to OpenQASM spec
- Some gates are not used as gates but for control flows
- QasmQobjInstruction.to_dict()

Qiskit

```
qc.append(QcAcquireLoop(1024, "cyclic"), [0])


qc.append(QcSweepRealTime("key", (0.0, 0.5, 1.0)),
[0])


qc.append(QcGate("rx", 1, ["key"]), [0])
qc.measure_all()

qc.append(QcCloseLoop(), [0])
qc.append(QcCloseLoop(), [0])
```

OpenQASM 2 (-like)

```
{'name': 'loop_acquire',
 'params': [1024, 'cyclic'],
 'qubits’: [0]},
{'name': 'loop_sweep_rt',
 'params': [key, (0.0, 0.5, 1.0)],
 'qubits': [0]},

{'name': 'rx', 'params': [key], 'qubits':[0]},
{'name': 'measure', 'qubits': [0], 'memory':[0]},

{'name': 'loop_close', 'qubits': [0]},
{'name': 'loop_close', 'qubits': [0]}
```

Code in control software DSL

Qiskit

```
qc.append(QcAcquireLoop(1024, "cyclic"), [0])


qc.append(QcSweepRealTime("key", (0.0, 0.5, 1.0)), 
[0])
qc.append(QcGate("rx", 1, ["key"]), [0])


qc.measure_all()


qc.append(QcCloseLoop(), [0])
qc.append(QcCloseLoop(), [0])
```

Zurich Instruments DSL

```
sweep_parameter = LinearSweepParameter(start=0, stop=1, count=3)

with exp.acquire_loop_rt(
     uid="shots", count=1024,
     averaging_mode=AveragingMode.CYCLIC,
     acquisition_type=AcquisitionType.INTEGRATION
     ):
  with exp.sweep(uid="sweep", parameter=sweep_parameter):
    with exp.section(uid="qubit_excitation"):
      exp.play(
           signal="drive", pulse=rx,
           amplitude=sweep_parameter
      )
    with exp.section(uid="qubit_readout"):
      exp.reserve(signal="drive")
      exp.play(signal="measure", pulse=readout_pulse)
      exp.acquire(
           signal="acquire",
           handle="ac_0",
           kernel=readout_weighting_function,
      )
    with exp.section(uid="relax"):
      exp.delay(signal="measure", time=1e-6)
```

Code on hardware sequencer

Qiskit

```
qc.append(QcAcquireLoop(1024,
"cyclic"), [0])
qc.append(QcSweepRealTime("key",
(0.0, 0.5, 1.0)), [0])

qc.append(QcGate("rx", 1,
["key"]), [0])

qc.measure_all()

qc.append(QcCloseLoop(), [0])
qc.append(QcCloseLoop(), [0])
```

Drive

```
repeat_count_AcquireLoopRt = 1024;
do {

  playWave(1,2,Pulse0_i,1,2,Pulse0_q);

  playZero(604768);

  // repeat with 2 more waveforms
  playWave(1,2,Pulse1_i,1,2,Pulse1_q);
 ...
repeat_count_AcquireLoopRt -= 1;
}
while(repeat_count_AcquireLoopRt);
```

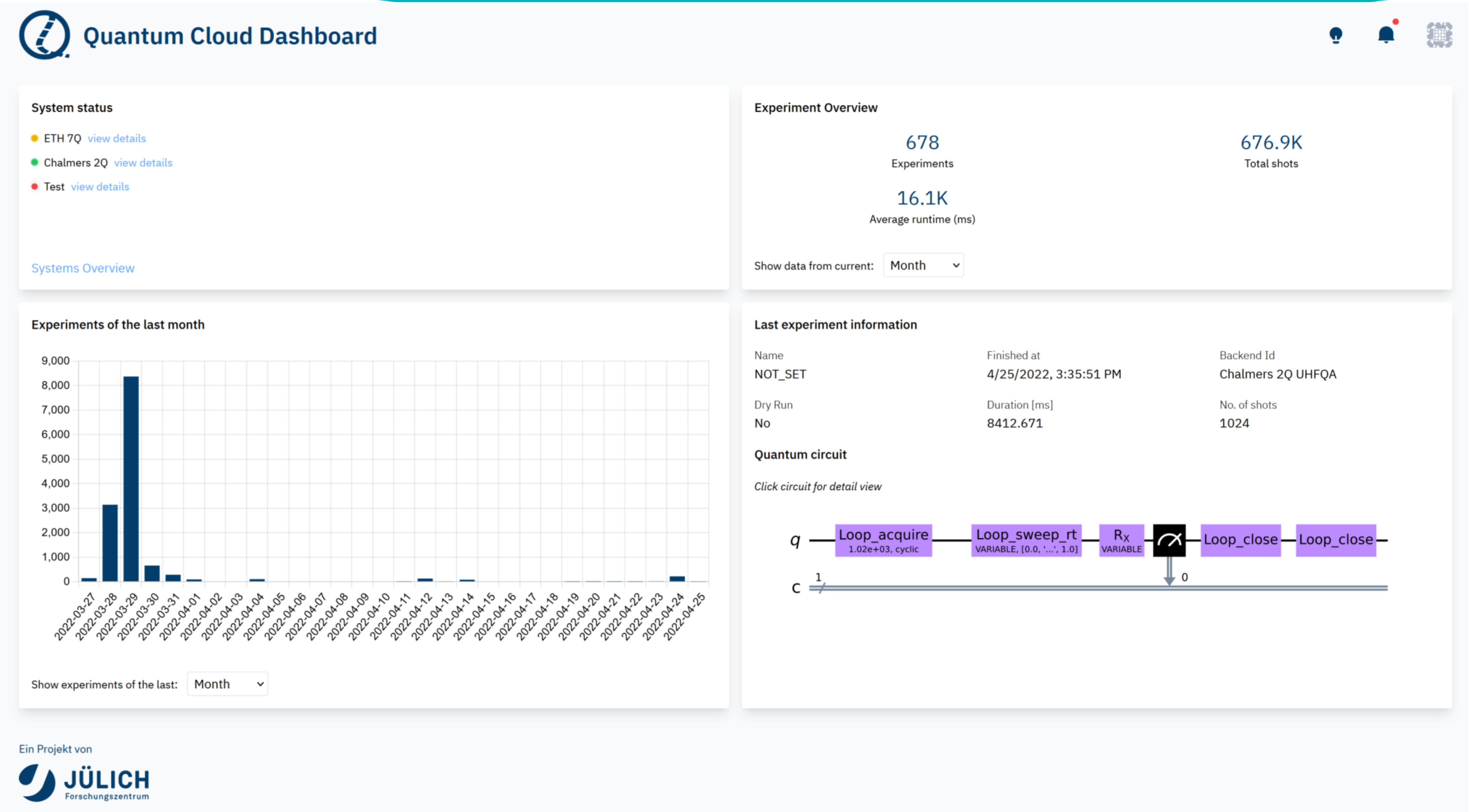
Readout

```
repeat_count_AcquireLoopRt = 1024;
do {
  repeat_count_0 = 3;
  do {

    playZero(57608);
    play_zero_count = 3;
    do {
      playZero(131056);
      play_zero_count -= 1;
    } while(play_zero_count);

    playWave(wp_2896_PulseFunctional);
    startQA(QA_INT_ALL,0);

    repeat_count_0 -= 1;
  }
  while(repeat_count_0);
  repeat_count_AcquireLoopRt -= 1;
}
while(repeat_count_AcquireLoopRt);
```



Quantum Cloud Dashboard

System status

- ● ETH 7Q view details
- ● Chalmers 2Q view details
- ● Test view details

Systems Overview

Experiment Overview

678
Experiments

676.9K
Total shots

16.1K
Average runtime (ms)

Show data from current: Month

Experiments of the last month

Show experiments of the last: Month

Last experiment information

Name	Finished at	Backend Id
NOT_SET	4/25/2022, 3:35:51 PM	Chalmers 2Q UHFQA

Dry Run	Duration [ms]	No. of shots
No	8412.671	1024

Quantum circuit

Click circuit for detail view

Ein Projekt von

JÜLICH
Forschungszentrum

Qiskit in the OpenSuperQ project

Summary

Qiskit has proven its adaptability to our specific needs (low level access)

Zurich Instruments together with the FZJ Software stack provide a quantum cloud API

We are looking forward to OpenQASM 3 which is believed to provide full flexibility

Software Stack