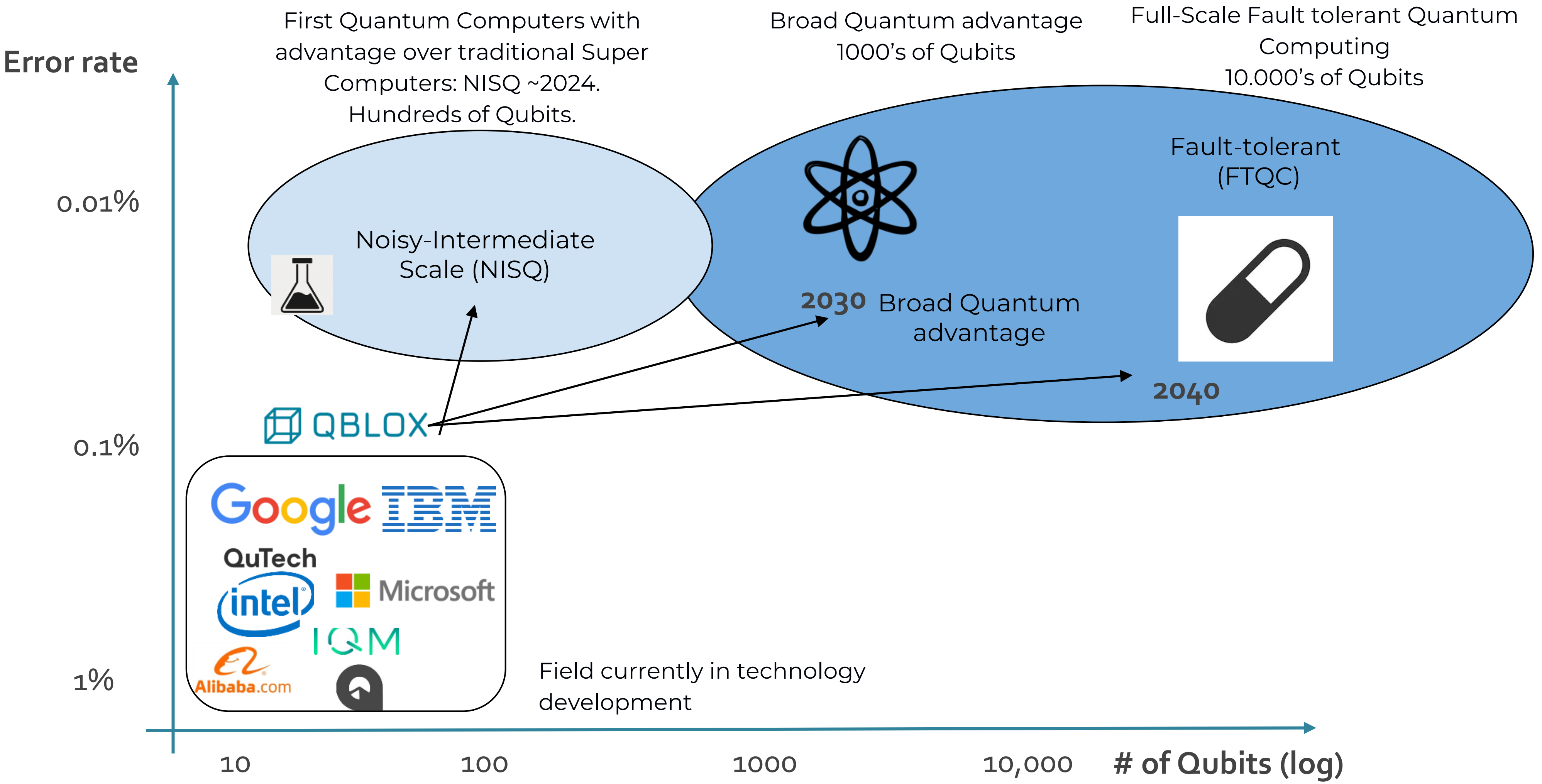
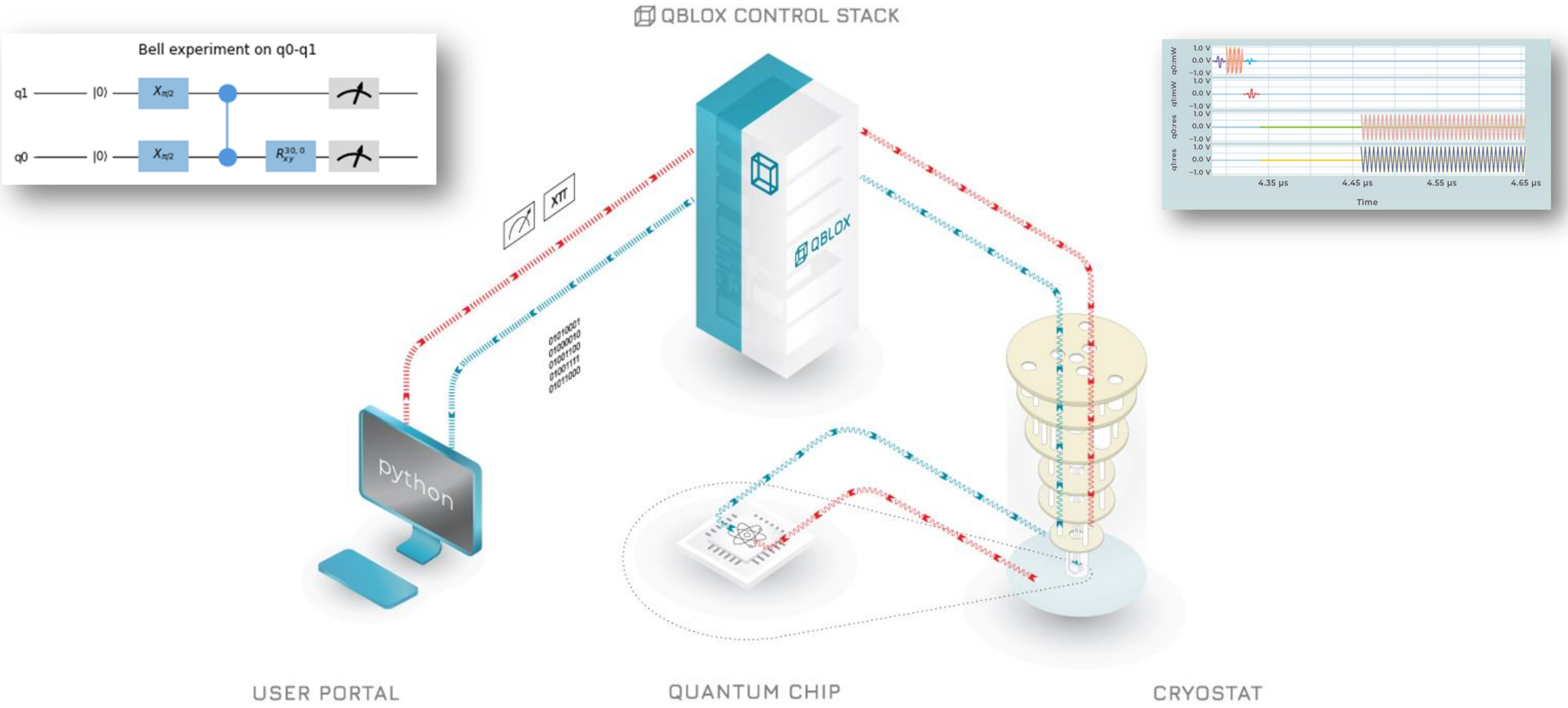
Mission: enable partners to create quantum processors large enough to be useful



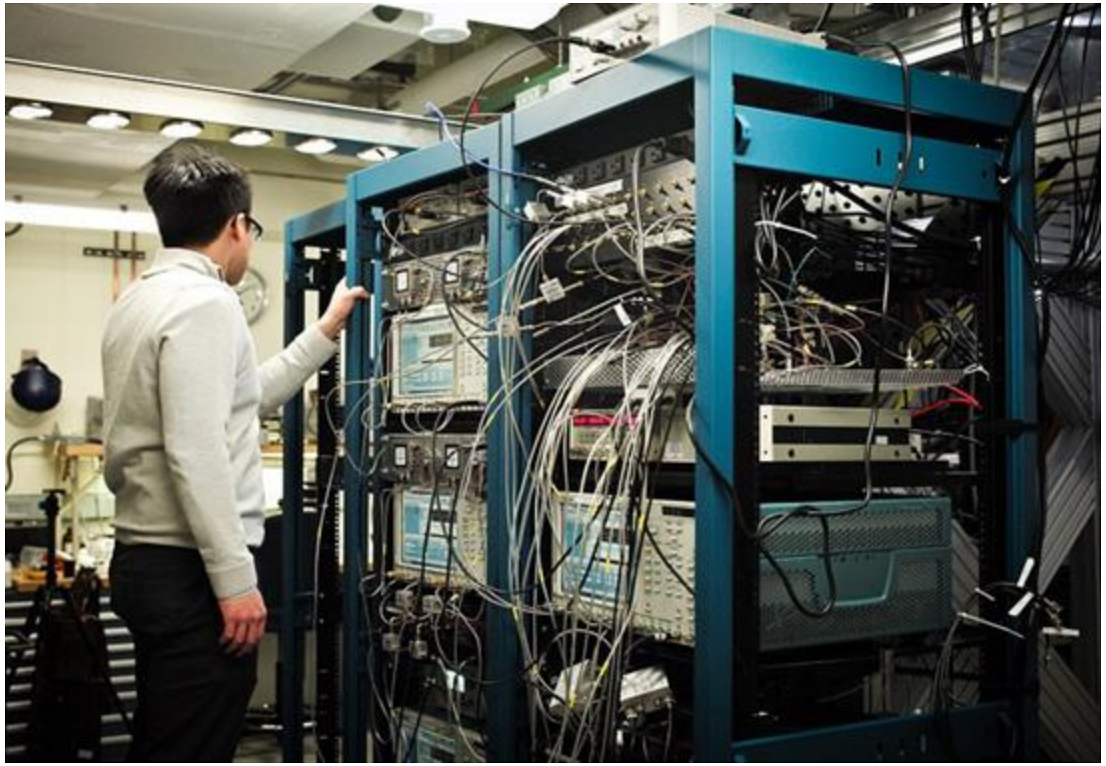
TNO innovation for life

QBLOX

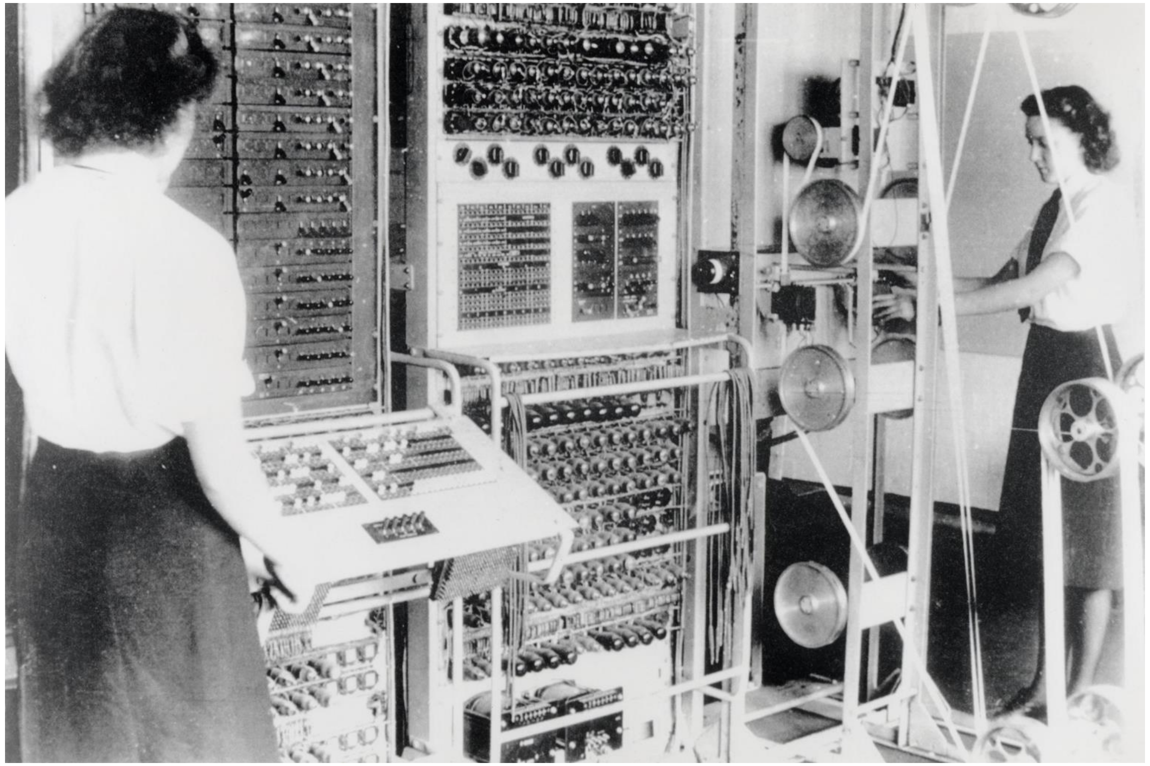
Revolutionizing the Control Stack



Quantum Computing is in its early stages

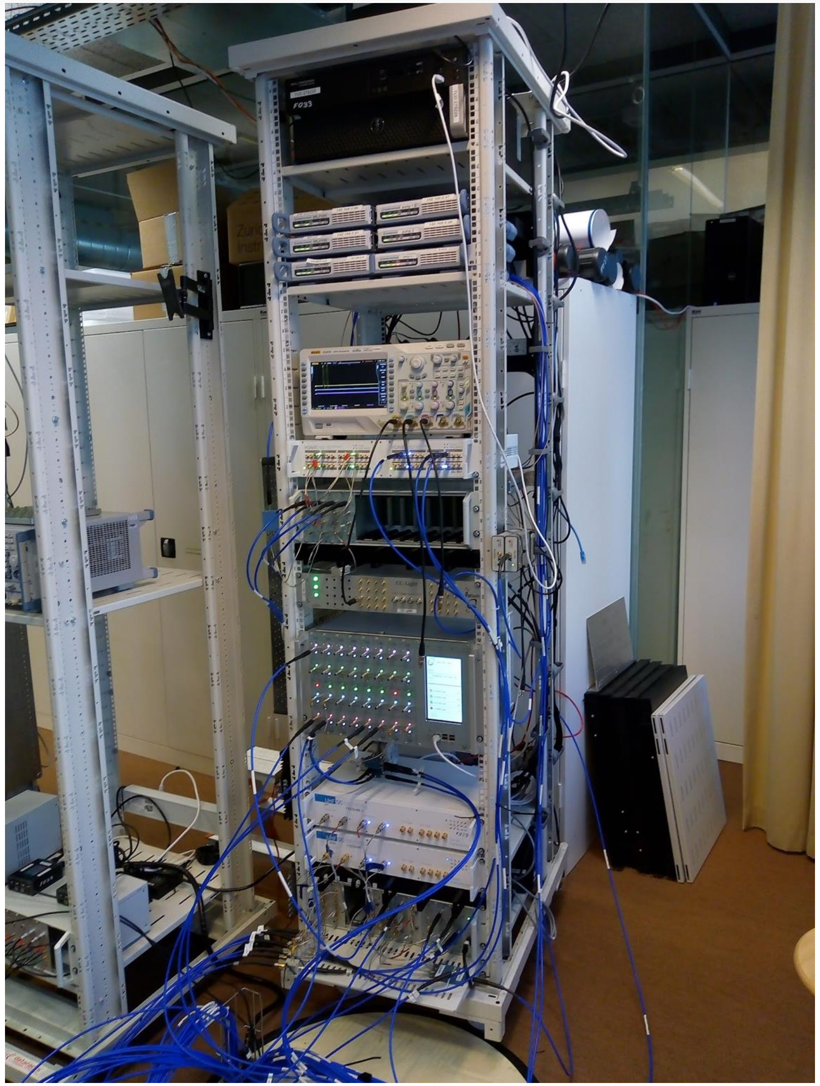


5 qubits control stack
@ IBM Research (2016)



Colossus Mark 2 (1943)

To advance research, we need better tools



5 qubit setup in lab of Leo Dicarlo (Tu Delft, 2019)

QuTech Waveform Generator (2017) 24 channel AWG in 9U rackspace

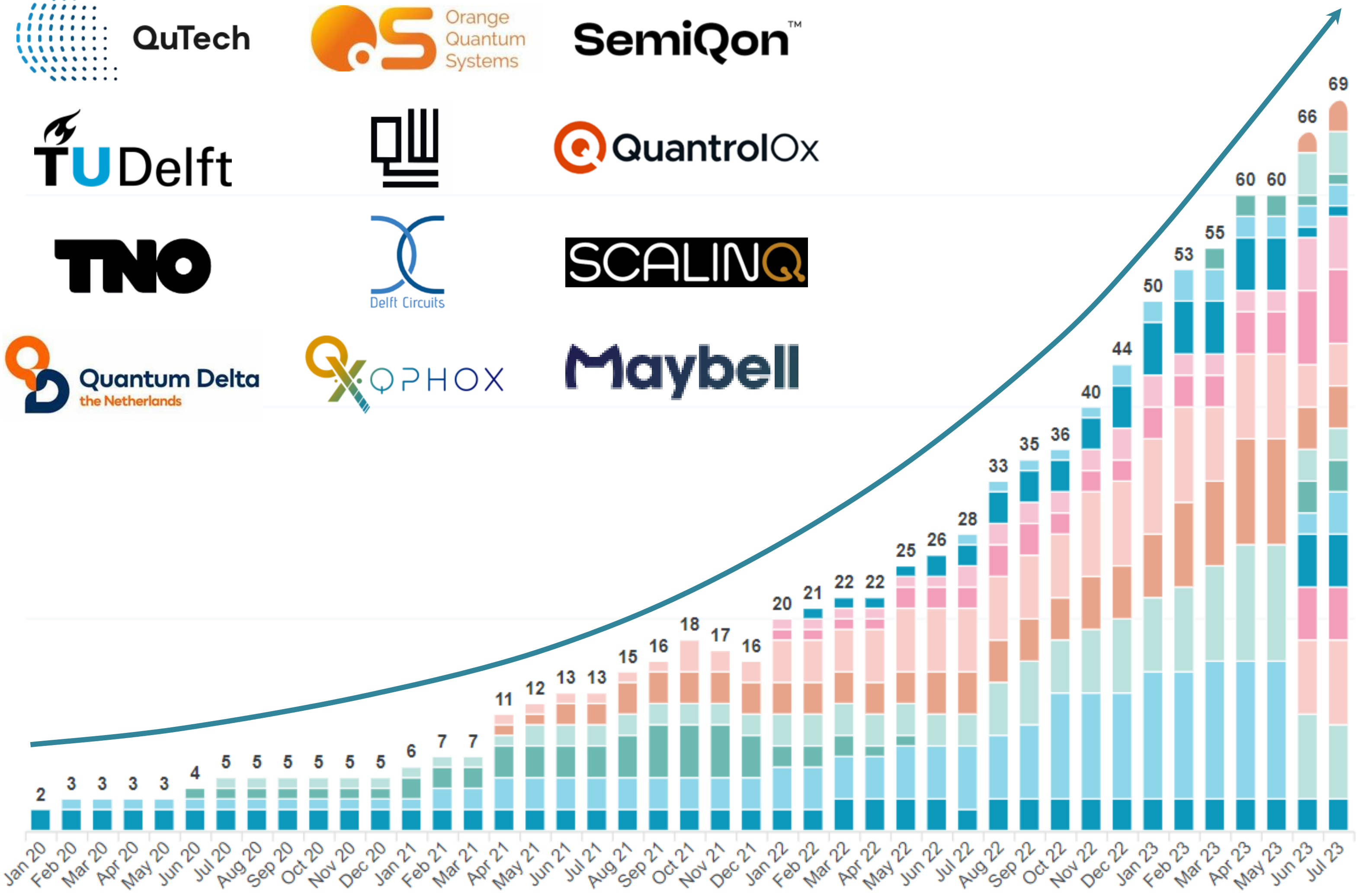
In 2018, Qblox was founded

Niels Bultink, CEO of Qblox (2019)

Qblox | the company

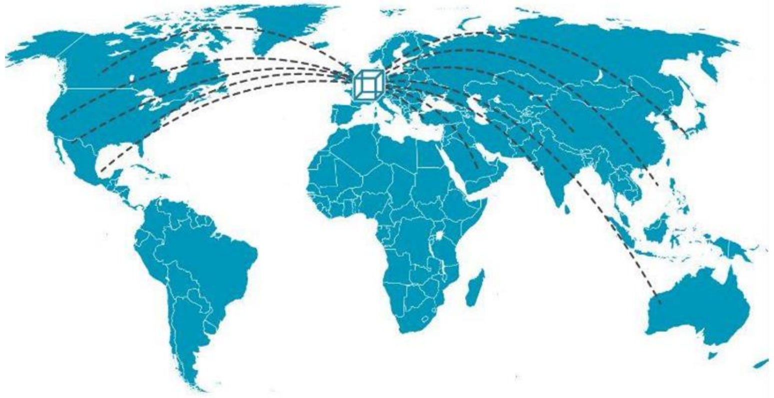
Partners and collaborators

QuTech
Orange Quantum Systems
SemiQon™
TU Delft
QuantrolOx
TNO
Delft Circuits
SCALINQ
Quantum Delta the Netherlands
QPHOX
Maybell



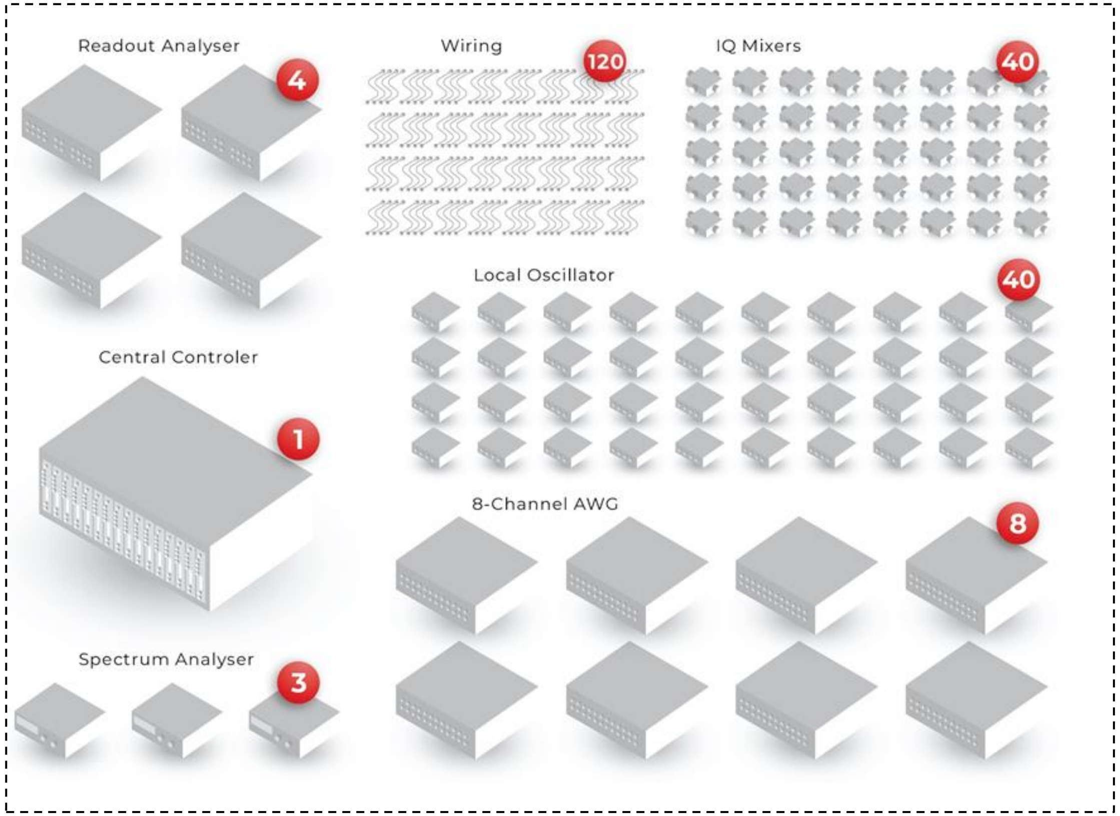
QBLOX

>80 team members
and many open vacancies

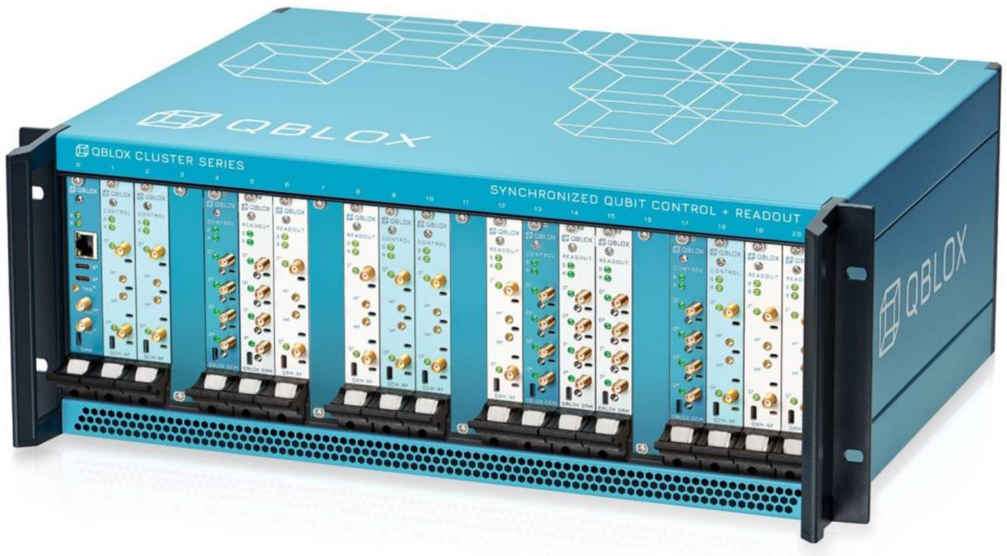


Supporting >80 labs globally

Fully-integrated Qubit Control and Readout



=



- Fully integrated
- Scalable
- Modular

Modularity

Qubit Control

QCM
QUBIT CONTROL MODULE
0-400 MHz

4 Output channels
5 Vpp

16 Bits
16k Wave memory

4 Digital outputs
1 GS/s Sampling rate

QCM- RF
QUBIT CONTROL MODULE
2-18.5 GHz

2 Output channels
-40 to +5 dBm

750 MHz analog bandwidth
16 bits

16k wave memory
2 Digital outputs

0-400 MHz — **2-18.5 GHz**

QRM
QUBIT READOUT MODULE
0-400 MHz

2 Output channels / 1 Vpp
2 Input channel / 0.1 - 2 Vpp

12 bits
16k Wave memory

4 Digital outputs
1 GS/s sampling rate

QRM-RF
QUBIT READOUT MODULE
2-18.5 GHz

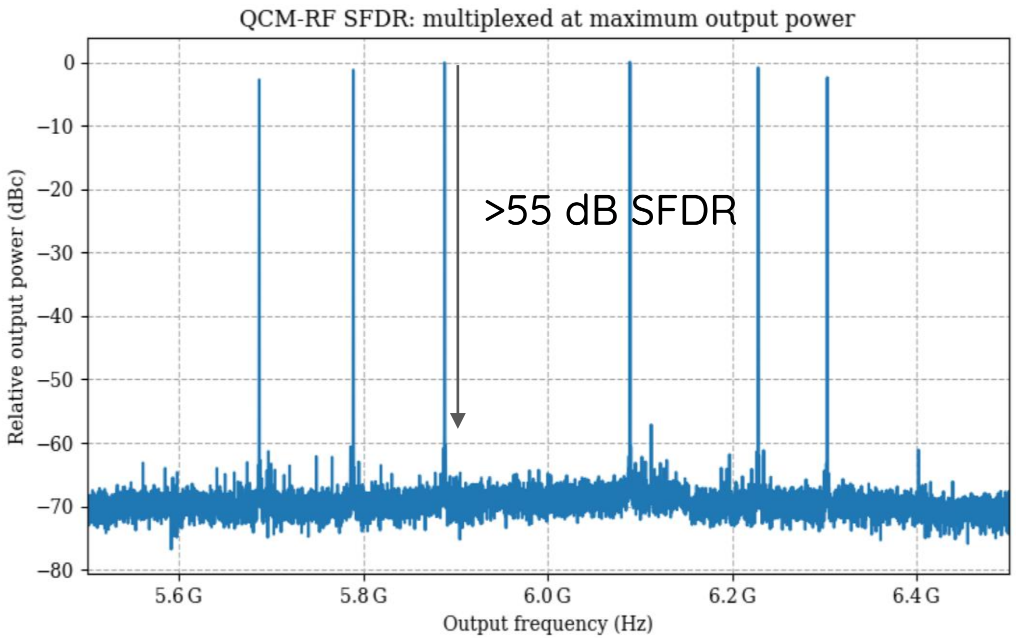
1 Output channel / -40 to +5 dBm
1 Input channel / -26 to 0 dBm

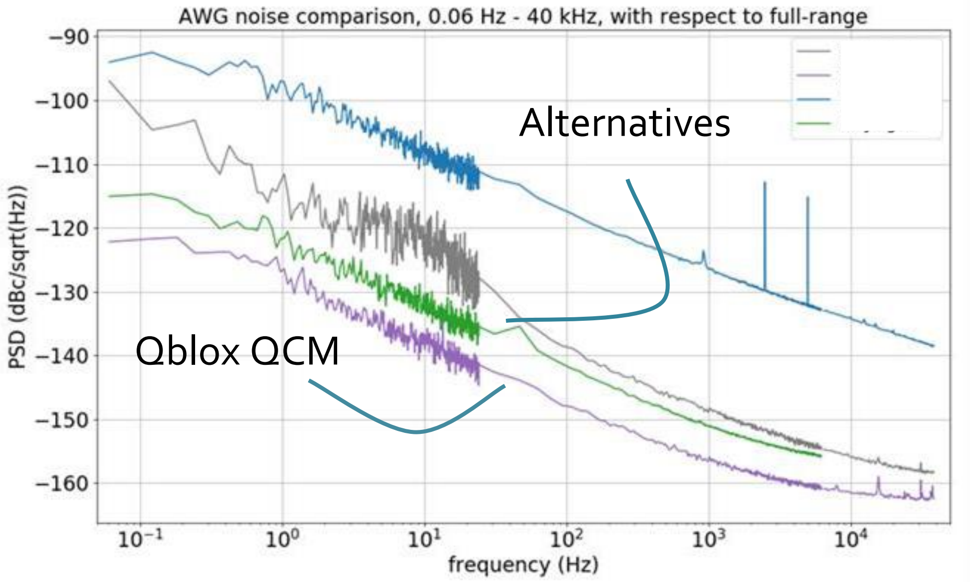
750 MHz analog bandwidth
12 bits

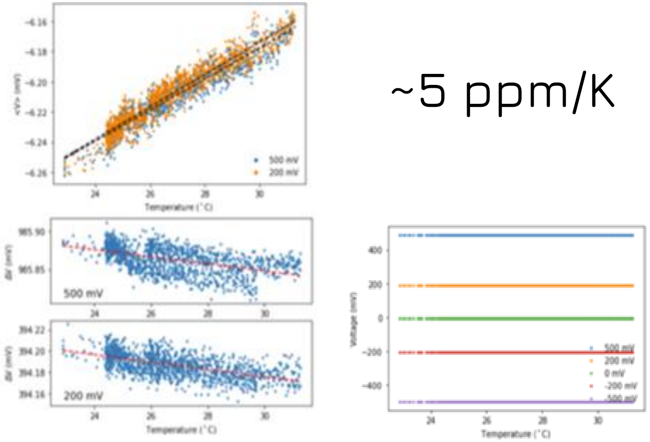
16k wave memory
2 Digital outputs

Qubit Readout

Good analog performance is crucial







~5 ppm/K

1. Low spurs (cross-talk)

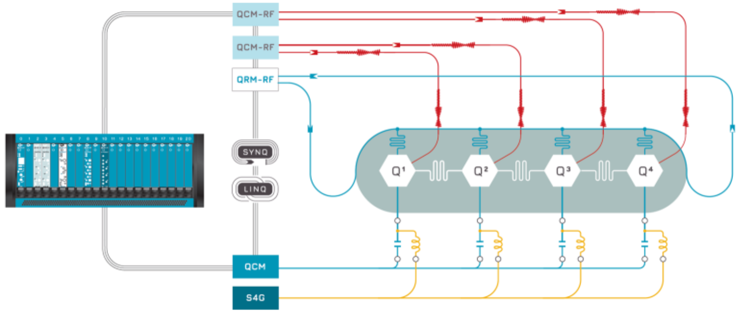
2. 1/f noise (2 qubit gate fidelity)

3. Ultralow drifts (minimize calibrations)

QBLOX

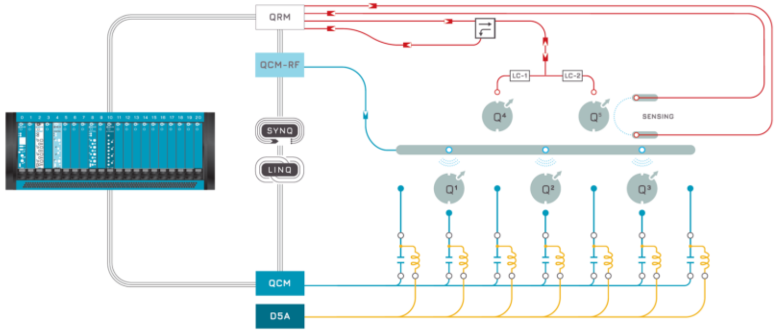
Our technology is compatible with multiple qubit platforms

Superconducting Qubits



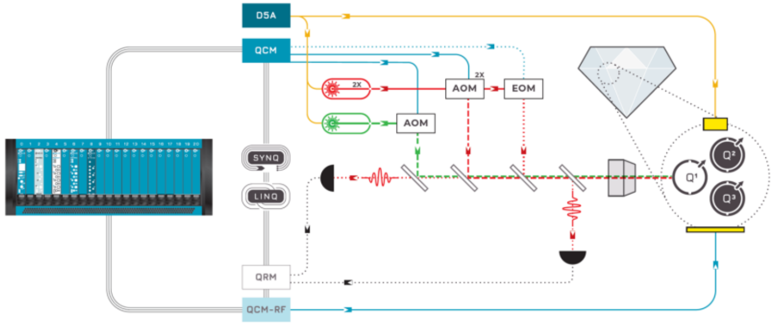
- Single-qubit gates
- 2-qubit gates
- Qubit/ Resonator spect.
- Multiplexed readout

Spin-based Qubits



- Tuning barrier potentials
- Fast gate sweeps
- RF-reflectometry
- Charge-sensing (Lockin)
- ESR / EDSR

Spins in Diamond



- Frequency tuning of lasers
- Control AOMs and EOMs
- Direct MW control
- TTL acquisition

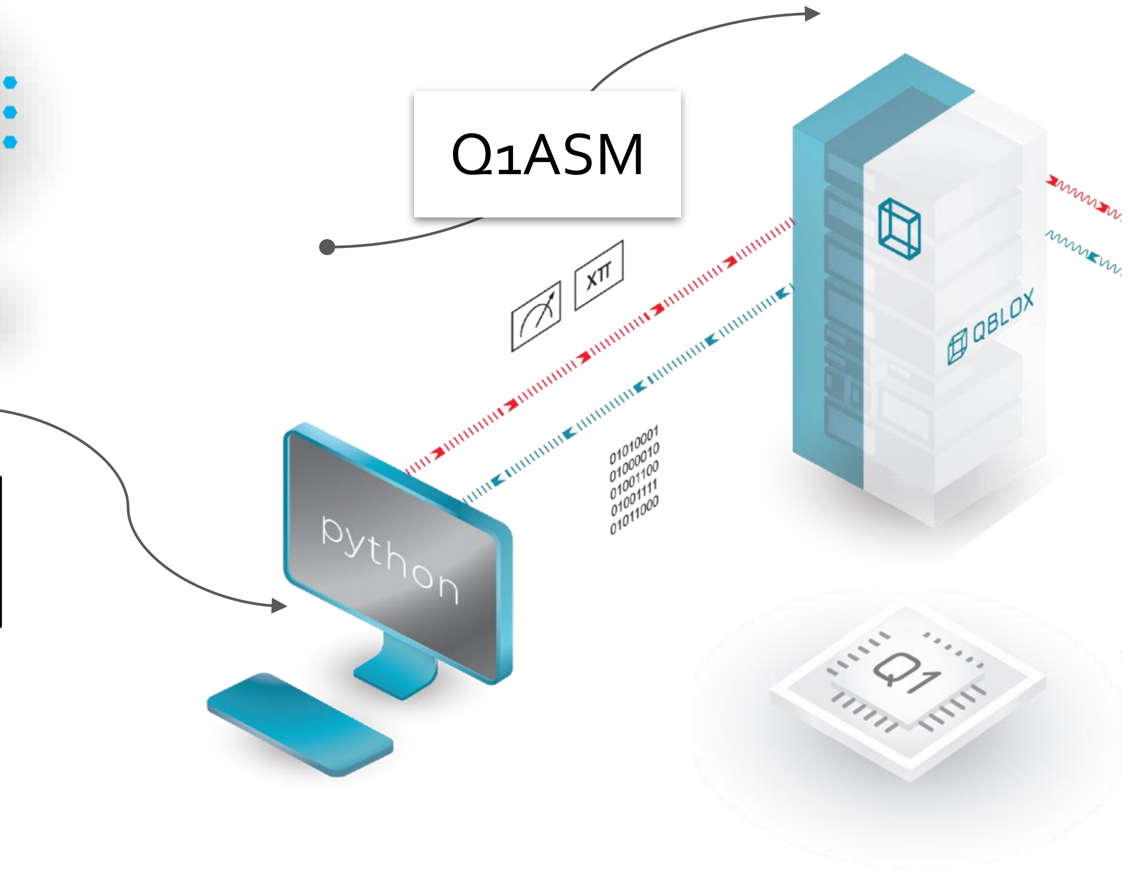
Qblox Instruments & Quantify

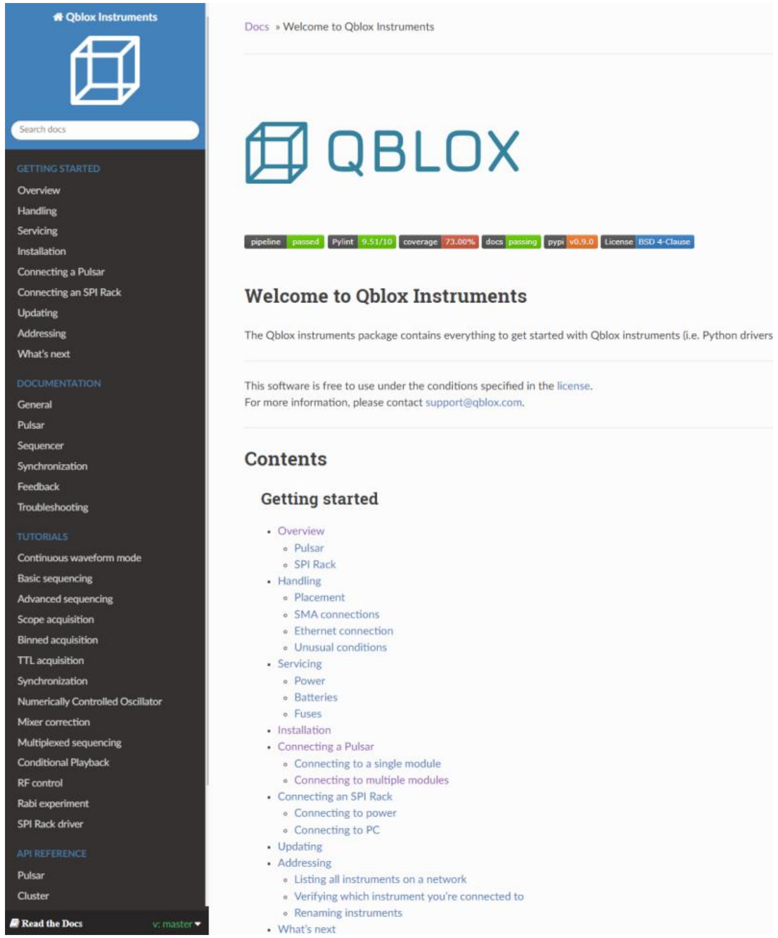
quantify

BSD 4.0
Open-source

Orange Quantum Systems

QCoDeS




Qblox Instruments
Search docs
GETTING STARTED
Overview
Handling
Servicing
Installation
Connecting a Pulsar
Connecting an SPI Rack
Updating
Addressing
What's next
DOCUMENTATION
General
Pulsar
Sequencer
Synchronization
Feedback
Troubleshooting
TUTORIALS
Continuous waveform mode
Basic sequencing
Advanced sequencing
Scope acquisition
Binned acquisition
TTL acquisition
Synchronization
Numerically Controlled Oscillator
Mixer correction
Multiplexed sequencing
Conditional Playback
RF control
Rabi experiment
SPI Rack driver
API REFERENCE
Pulsar
Cluster
Read the Docs v: master

Docs » Welcome to Qblox Instruments

QBLOX

pipeline passed | Pylint 9.51/10 | coverage 73.00% | docs passing | pypi v0.9.0 | License BSD 4-Clause

Welcome to Qblox Instruments

The Qblox instruments package contains everything to get started with Qblox instruments (i.e. Python drivers

This software is free to use under the conditions specified in the license.
For more information, please contact support@qblox.com.

Contents

Getting started

- Overview
  - Pulsar
  - SPI Rack
- Handling
  - Placement
  - SMA connections
  - Ethernet connection
  - Unusual conditions
- Servicing
  - Power
  - Batteries
  - Fuses
- Installation
- Connecting a Pulsar
  - Connecting to a single module
  - Connecting to multiple modules
- Connecting an SPI Rack
  - Connecting to power
  - Connecting to PC
- Updating
- Addressing
  - Listing all instruments on a network
  - Verifying which instrument you're connected to
  - Renaming instruments
- What's next

Rabi : Q1ASM vs Quantify

```
qcm.sequencer0.sync_en(True)
qcm.sequencer0.nco_freq(200e6)
qcm.sequencer0.mod_en_awg(True)

qrm.sequencer0.sync_en(True)
qrm.sequencer0.nco_freq(40e6)
qrm.sequencer0.mod_en_awg(True)
qrm.sequencer0.demod_en_acq(True)
```

```
prog = """
           move       0, R0
ampl_loop: add        R0, 50, R0
           move       1000, R2
           set_awg_gain R0, R0
navg_loop: wait       50000
           wait_sync 4
           set_mrk    1
           play       2,3,160
           set_mrk    0
           upd_param 4
           loop       R2, @navg_loop
           jlt        R0, 25000, @ampl_loop
           stop

"""
prepare(qcm.sequencer0, prog)
```

```
prog = """
           move      0, R0
ampl_loop: move      1000, R2
navg_loop: wait_sync 4
           wait      160
           play      1, 1,260
           acquire 1,R0,1000
           loop      R2, @navg_loop
           add       R0,1,R0
           jlt       R0, 25000, @ampl_loop
           stop
"""
prepare(qrm.sequencer0, prog, acquisitions=acquisitions)
```

```
sched = Schedule("Rabi experiment")
sched.add_resources([ClockResource("q0.01", 6.02e9)])
for acq_idx, amp in enumerate(np.linspace(0, 1, 500)):
    sched.add(Reset('q0'))
    sched.add(DRAGPulse(G_amp=amp, D_amp=0, phase=0, duration=160e-9, port="q0:mw", clock="q0.01"))
    sched.add(Measure('q0', acq_index=acq_idx))

sched.repetitions = 1000
```

```
_, ax = sched.plot_circuit_diagram()
ax.set_xlim(-0.5, 9.5)
for t in ax.texts:
    if t.get_position()[0] > 9.5:
        t.set_visible(False)
```