LabOne Q

Zurich Instruments

Software Framework for Quantum Experiment Control

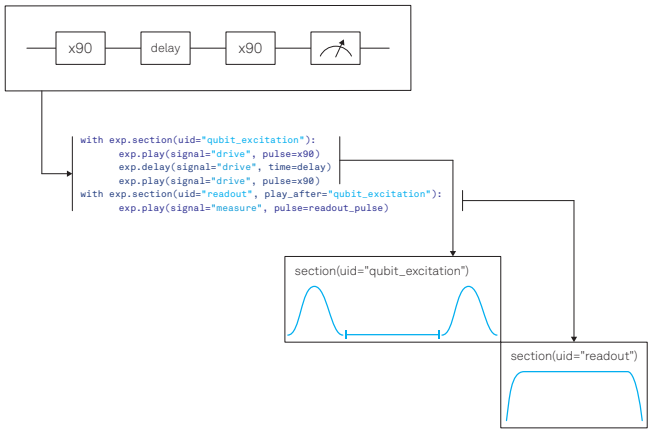
Product Leaflet
Release date: March 2023

Highlights

- Spend time where it matters. Your experiments.
- Program large setups in Python as a single machine.
- Transform complex quantum circuits into simple code.
- Track every detail down to the last sample.
- Optimize your setup for a high duty cycle.
- Scale up with drop-in code replacement.
- Increase your throughput with a streamlined interface.

Designed for Your Experiments on the Most Advanced Control Electronics

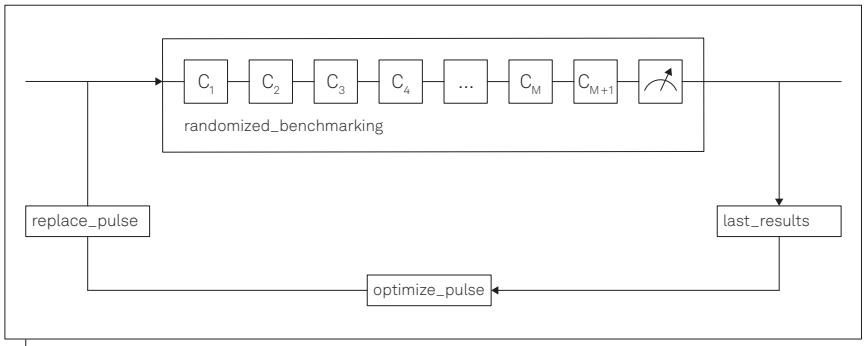
With LabOne Q you can express experiments in a language that connects the abstract world of quantum gates with the concrete world of signals. The Python based, high-level interface provides intuitive tools for specifying multi-channel signals with precise timing control.
LabOne Q makes it easy to reuse code. Sections allow you to group pulse patterns across signal lines. They support looping, sweeping, and averaging structures to express all levels of timing consistently, from sample-precise pulse-level control up to the arrangement of multi-experiment protocols.
LabOne Q streamlines and automates time-consuming tasks such as optimizing instrument settings, generating and uploading waveforms, and synchronizing pulses between multiple instruments. Even with this control and measurement automation, users still have complete low-level access to the instruments. Setup details and instrument settings are accessible through a single line of code.
LabOne Q also provides browser-based tools for visualization of the experimental sequence and pulses before execution, giving a simplified overview of complex experiments, even with hundreds of pulses.



Supporting High Duty Cycles

LabOne Q enables users to get the most out of their setups by helping them extend the time the experiment is providing results. With an optimal duty cycle a quantum computer spends the highest possible computing time on the quantum processing unit. LabOne Q supports the user to maximize computational duty cycles by minimizing communication overhead, optimizing control instructions, pipelining and queuing experiments with a separation in offline and online functionality, and with the interfaces for automation.

Additionally, experiments have a clear distinction between real-time execution, when the instruments execute instructions and gather data autonomously, and near-time instructions, when communication with an outside process is necessary. For example, when tuning up a quantum gate with optimal control, real-time calibration sequences are interleaved with optimization steps, where intermediate measurement results are used to optimally calibrate the required gate pulse.

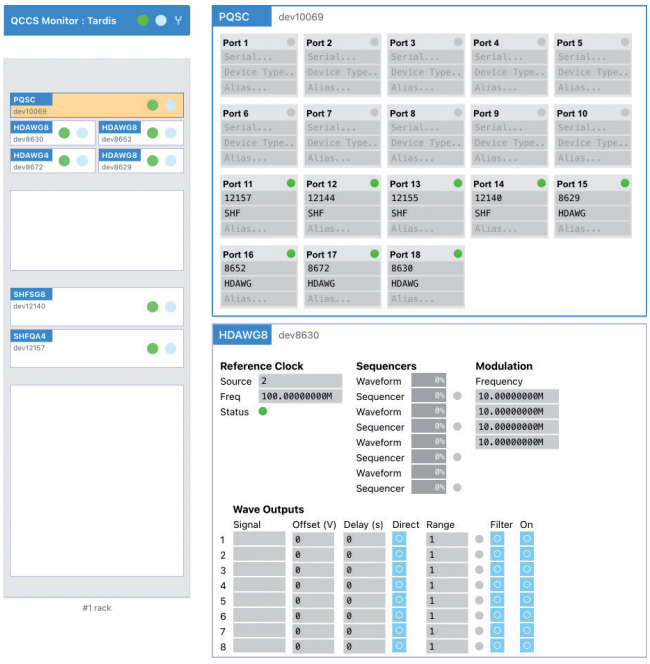



```
with exp.sweep(parameter=optimizer_steps, execution_type=ExecutionType.NEAR_TIME):
       randomized_benchmarking(exp, base_pulse=x90, handle='rb_data')
       exp.call(replace_optimized_pulse)

def replace_optimized_pulse(session):
       last_results = session.results.get_data('rb_data')[session.results.get_last_nt_step('rb_data')]
       new_pulse = optimize_pulse(last_results)
       session.replace_pulse(x90, new_pulse)
```

Performance that Scales with Your Setup

With LabOne Q, controlling many qubits is straightforward. Upgrading or scaling up a lab setup just requires a drop-in replacement in the code, with synchronization of all instruments ensured from the start. Simultaneous tune-up of multiple qubits, multiplexed qubit state readout, multi-qubit gate calibration, or control crosstalk compensation –it is included in LabOne Q.
LabOne Q supports your scaling roadmap by breaking down complexity, enabling automation, and maximizing performance. Its programming interface provides an accessible overview, even when working with 100 signals and more, and the System Monitor holds all critical hardware settings just a click away.



Zurich Instruments
Technoparkstrasse 1
CH-8005 Zurich
Switzerland
www.zhinst.com
info@zhinst.com
+41 44 515 04 10