QC-Toolkit

Version 0.3 - DRAFT

RWTH Aachen University

July 21, 2015

Abstract

This document describes the current status of the requirement analysis and the implementation progress of the QC-Toolkit project. This project consists in the restructuring of the pulsecontrol (special measure) project of Prof. Dr. Hendrik Bluhm. In short, the QC-Toolkit should increase the modularity and flexibility of pulsecontrol.

Contents

Involved People

The Bluhm Group (Physics)

Prof. Dr. Hendrik Bluhm Project initiator and supervisor.

Patrick Bethke PhD-Student

Pascal Cerfontaine PhD-Student

Tim Botzem PhD-Student

Simon Humpohl Scientific assistant

The Rumpe Group (Computer Science)

Prof. Dr. Bernhardt Rumpe Project initiator and supervisor.

Deni Raco Student assistant supervisor

Lukas Prediger Scientific assistant

Jerome Bergmann Scientific assistant

1 Introduction

Experiments in quantum mechanics involve the manipulation of a quantum object and some measurement of its behavior. In the context of the experiments performed by the physicists of the Bluhm Group this manipulation is performed via sequences of voltages (waveforms) applied to electrodes in the vicinity of the quantum object (e.g. a dual-electron qubit). Such a waveform is called a pulse when used to manipulate a quantum object. The measurement also results in a waveform induced by changes in the electromagnetic field of the quantum object depending on its state. Multiple concurrent pulses may be used to manipulate the quantum objects.

To execute an experiment, pulses have to be defined as well as corresponding measurement windows. As one goal is to find optimal pulses for certain quantum state changes, pulses have to be refined in an iterative process, where a starting pulse is found by simulation and then refined based on experimental data. Some pulses may be constructed as a sequence of other, more basic pulses. Ultimately, pulses that implement quantum gates may be found by composing elementary state manipulation pulses.

Pulses are generated by so called `Arbitrary Waveform Generators` (`AWG`s). These are hardware devices that receive a discrete digital byte array representation of a waveform where each byte represents the amplitude of the pulse at the corresponding clock tick and generate a continuous analog waveform. Measurement data is captured by `Data Acquisition Cards` (`DAC`s) which essentially do the same in reverse.

A software system that allows the construction of pulses and the measuring and evaluation of experimental data independent of the hardware and its wiring/setup is required for the execution of experiments. The software currently used by the Bluhm Group for this purposes as well as its shortcomings is described briefly in Section 3. The QC-Toolkit project is intended to produce a more general framework which enables quantum mechanical experiments. Later on, it shall be extended to enable modeling and possibly execution and simulation of quantum circuits. A detailed list of requirements is provided in Section 4 followed by a prototypical software design and implementation proposition in Section ??.

`Disclaimer:` Please note that the above description is only intended to provide a broad overview of the physical background for the software. It may thus be incorrect in some details but should convey the relevant information. For a more accurate and detailed description of the physics involved and the actual experiment performed by the Bluhm Group, please refer to the corresponding literature.

2 Roadmap

2.1 Meetings

- `26.01.2015`: **Kickoff Meeting**: Explanation of the project, declare first steps of the Rumpe-Group: Work with Simon Humpohl on `pulsecontrol` to get a deeper introduction into the system. After one to two month, the requirements analysis started.
- `24.04.2015`: **Second Meeting**: Presentation of the research results and a first draft for the structure of the qc-toolkit. The feedbacks were generally positive but clarification and more implementation details requested.
- `19.05.2015`: **Third Meeting**: Reviewed draft of the first implementation. More strict separation of the individual features. Approval to start implementation. Also discussion about serializing techniques for pulses.
- `13.07.2015`: **First sprint Meeting**: Presentation of the result of the first sprint phase: The core is partially written and presentation of the first hardware interface instruction set.
- `next meeting`: probably end of august

2.2 Tasks

- `high priority`: In general, the highest priority is to reach a runnable system which has at least the same functionality as `pulsecontrol` to allow a transition to qc-toolkit for the physicists.
  - Core implemenation
  - Generic hardware interface design
  - Serializing of pulses
- `medium priority`: In general, these are nice-to-have features, which doesn't change the general behavior of qc-toolkit, but make the work with the toolkit more comfortable and easier.
  - Plotting
  - First driver
  - Logging levels for Pulses

- `low priority`: These features are targeting future hardware or features which can only be introduced when core is finished.
  - Feedback loops
  - Function calls in the hardware interface

3 Special-Measure and PulseControl: The Current State

The Bluhm Group currently uses the special-measure project[1] for data acquisition and evaluation and the self-implemented special-measure.pulsecontrol[2] for pulse definition and executing.

Special-measure is a general purpose data acquisition package for (table top scale) physics experiments. Out of that project, the Bluhm Group created pulsecontrol, a package for defining and tracking AWG pulses for quantum information experiments. It communicates with special-measure in a few places but is largely independent. Both projects are written in the MATLAB script language in a largely imperative manner using global variables to store state information.

As the development of pulsecontrol took place over several years, it is now a feature rich toolkit. Currently, the project has reached a state where it is hard to understand, modify, extend and repair with dependencies being reaching through the entire code base. To address these issues, the QC-Toolkit Project has been initiated, aiming for an object oriented reimplementation of pulsecontrol functionality and subsequently adding more features and higher abstraction.

Furthermore, in the current state the user has to define pulses and set up the software using its MATLAB API for each experiment. It may be easier if the user could just specify pulses and hardware setup entirely as data (files) without ever touching any programming language and that the software then derives its internal configuration and pulse representation from this automatically.

3.1 Introductional Work: The Driver Interface

Initially, the pulsecontrol toolkit featured only single AWG support. It was not possible to employ multiple AWGs or even AWGs that exhibited different interfaces (i.e., ones from different manufacturers). During the writing of this document, the pulsecontrol toolkit has been actively expanded for multi AWG support which is critical for upcoming experiments. The basic concept allows to map the channels of a virtual AWG onto hardware channels provided by several “real” AWGs.

Additionally, a data acquisition driver has been implemented which allows direct filtering and processing of relevant data from the incoming measurement waveforms.

There is a high likeliness that the results of these two current efforts can be directly included in the QC-Toolkit or at least adapted with only minor changes.

[1]github.com/qutech/special-measure
[2]github.com/qutech/special-measure.pulsecontrol

4 Requirements: The Desired State

Until now, we were able to formulate the following requirements for the QC-Toolkit.

- The system should provide different levels of abstraction: It should be possible to operate directly with pulses (to optimize them or to inspect the physical behavior of the quantum hardware) as well as (later on) representing and executing quantum circuits and algorithms without having to deal with corresponding pulse implementation.
- The system should support the execution of quantum circuits/algorithms on real qubit hardware as well as some means of simulation (Integration of the qtip project[3] might be a viable solution for the latter).
- The implementation should be object oriented and easy to maintain.
- The system should scale to support any number of input and output channels, concurrent pulses and qubits.

Pulses *Defining pulses is a crucial feature of the software. It is one of the aspects physicists of the Bluhm Group are most exposed to at the moment as pulses are the input to their experiments. Thus, the software should provide means to quickly compose new pulses or easily modify existing ones.*

- Basic pulses should be defined by a data driven approach, i.e., a pulse is set up by passing a set of time-voltage points (currently called pulse table or pulsetab). The intermediate values of a pulse are then obtained by interpolating these points either linearly or using a user-defined interpolation function.
- It should be possible to nest pulses, i.e., construct pulses by reusing and combining already existing ones.
- A pulse should be parameterizable (e.g., varying duration, amplitudes, etc.) and accept mathematical functions that define whether/how these parameters are mapped to subpulses to enable to use it as a template to quickly define variations of pulses with the same structure.
- The system must be able to generate corresponding waveforms.
- Pulses should be defined independently from the hardware setup or the available output channels.
- The system should be able to store pulses, pulse templates and possibly pulse template parameters persistently in some kind of database.
- Changes to a pulse in the database should reflect to pulses that refer to the affected pulse as a subpulse.
- There may be several database scopes: A global scope, which stores only the most commonly used pulses, preferably as templates, project or hardware scopes, storing templates or concrete pulses[4] that are commonly used in a specific project for a specific hardware setup and maybe even experiment or

[3]qutip.org

[4]Here, concrete pulses are pulses that are not parameterizable, probably obtained by filling a pulse template with parameters.

user scopes, storing pulses relevant only for a specific experimental run or often used by a user.

In the discussion of the third meeting, we agreed to use json as language to serialize pulses to.

Experiments *Here, an experiment is understood as a specific sequence of pulses executed for a specific hardware setup and may be part of an overarching project. Experiments involve execution of the pulse sequence and acquiring and processing the experimental data. Further, there is a need to store the data associated with an experiment such as pulses, parameters, setup, results and possibly user comments and documentation.*

- A concrete pulse used in a certain experiment must be reconstructable, even if the corresponding pulse in some database (if existent) was changed afterwards.
- The system should enable (interactive) feedback loops for optimization of pulses, meaning that in a single experimental run, a pulse is executed, then changed according to the obtained measurement result and executed again until a desired state is reached.
- The system should, if possible, enable automatic adjustment of parameters in such a feedback loop depending on the measured data according to a provided set of rules (e.g., mathematical mappings).
- A snapshot feature for runs should be implemented to allow interrupting and resuming or replaying an experimental run.
- The system should provide conditional branching and looping during an experiment, meaning that, e.g., after some initialization pulse the system selects which pulse to execute next based on some external condition.

Hardware Abstraction *The migration to other setups or devices should be easy and the interface for both as general as possible. Especially the number of channels must be extensible.*
Hardware abstraction should

- specify general interfaces for AWGs and DACs which abstract from a concrete device.
- allow for easy and dynamic exchange of the implementation these interfaces (without compilation).
- hide the actual hardware setup and only offer access to user-named virtual channels for input and output.
- allow the user to specify the hardware setup as data, containing information which concrete AWGs and DACs are involved, how they are configured and how virtual channels map to the hardware.
- allow the dynamic adaption of certain DAC settings (e.g., output voltage range, frequency) to react to changes in the physical environment (e.g., heating) during an experiment.

After discussion 4, we all agreed that a good starting point will be a translation of the object oriented treelike representation of a pulse into a assembly-like set of instructions. The approved instructions are:

- `exec <waveform>`: execute a single waveform from device memory
- `cjump <condition/trigger> <target>`: perform a conditional jump to a target instruction index
- `goto <target>`: an unconditional jump
- `stop`: stop playback

Prof. Bluhm requested additionally some kind of superior function calls supported by some AWGs.

Data Acquisition *Acquiring experimental data is equally important as defining pulses. It is the only source of information to access the quantum state to refine pulses or, later on, execute quantum circuits. Since measuring the state is possible only in certain time windows depending on the executed pulse, the continuous input signal has to be filtered and processed by the software.*
To acquire measurement data, the system should

- capture waveforms on input channels (in real-time).
- filter out only relevant information from these waveforms according to user-specified masks.
- process (accumulate, transform, map,..) the obtained data.
- identify to which pulse(s) the captured data belongs (depending on the hardware setup specified by the user).

Quantum Circuits *Ultimately, the system should provide functionality to execute algorithms on quantum hardware (if possible).*

- The system should provide means to model (and possibly execute) Quantum Circuits, i.e., Quantum Gates and Qubits.
- This model should be independent from underlying implementation (execution on hardware using pulses, simulation, etc.).

Code Structure *It is also necessary, that the source code itself is of the best possible quality.*

- In order to allow the physicists an easy manipulation of the code, the programming language has been voted to python. We will use the python3 version to be up to date, being able to use the newest external packages and the annotation feature introduced in python3.
- We will use the guidelines for the code declared in PEP0008 and the guidelines for docstrings in declared in PEP0257.
- We want to make the development process easier by using the type hints declared in PEP0484. We will use mypy for static typechecking and write also our own limited typechecker for runtime typechecking.

- For each self written python file, there will be a unittest, covering as much as possible of the functionality of the given module. These files will be stored apart from the main source code to increase readability and reduce confusion (see folder `/test`).
- The code on the gihub should always be running code. Therefore Travis CI has been integrated into the project, running all the tests before committing. Only if all test where successful, a push onto the github repo is allowed. This does not include formatation checks.

5 Related Software

5.1 The "QtLab" Project

Patrick Behtke suggested the usage of the QtLab[5], a similar project with a focus on data acquisition/analysis and pulse generation for quantum transportation. It is written in Python following an object oriented approach. However, there seems to be no official release and the source code is not very well documented, making it difficult to understand, maintain or even get it to run. It is also limited in terms of how pulses are modelled: Each kind of pulse has to be implemented as a subclass of an abstract pulse class. There is no means to define pulses via data such as the pulse table concept in the pulsecontrol project. Additionally, it is also not possible to nest pulses or parameterize them. While this functionality could in theory be added to QtLab, we are reluctant to touch the code due to the lack of documentation. Given all that, we are unsure whether we can benefit from reusing the QtLab software.

However, interfaces and concepts of the QtLab may prove to be a useful reference for QC-Toolkit. Especially setting up, executing and evaluating a measurement seems to be relatively easy for the end user and could be used as an inspiration for corresponding interfaces.

6 Terminology

AWG : Arbitrary waveform generator, output device.

Pulse : Internal representation of a tension curve.

PulseTab : Pulse defined by some points with linear integration in between.

Experiment : The set of Input-Data, Measurements and Documentation for each run.

Setup : Internal representation of the links between the output-port of the AWG(s) and the QPU and QPU and the measurement tools.

Waveform : Actual data, that will be played by the AWG.

DAC : Data aquisition card, measurement device.

[5] qtlab.sourceforge.net and github.com/teamdiamond/measurement