Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Laboratory for Solid State Physics
Quantum Device Lab

Semester Thesis

Realizing a Calibration Program for Superconducting Qubits

Tim Menke

August 2013

Supervisor
Lars Steffen

Professor
Prof. Dr. Andreas Wallraff

Abstract

Superconducting qubits are a promising candidate for developments in quantum communication and computation. As the qubits couple to a fluctuating environment, their properties shift over time. In the experiment we have been working on, Transmon qubits are used to teleport a quantum state. Fluctuations in the system require a daily calibration taking up to 1.5 hours. As we scale the number of qubits in the system, automating the calibration routine becomes unavoidable. In this thesis we present a calibration program written in LabVIEW and Mathematica. The program QubitCalib automates the calibration routine for an arbitrary number of qubits and resonators and the calibration modules can be executed in any desired order. After execution of the program, all parameters needed for single qubit gates are adjusted and a log file containing the data analysis results is shown. QubitCalib is run over a simple interface that is easy to manage.

Contents

1. Introduction **6**

2. Overview of the Experimental Setup **6**
2.1. Circuit QED 6
2.2. Experimental Setup 7

3. Calibration Parameters **8**

4. Calibration Measurements **11**
4.1. Rabi Measurement 11
4.2. Ramsey Measurement 12
4.3. QScale Measurement 13
4.4. CalTom Measurement 13

5. Data Analysis Techniques **14**

6. The Calibration Program QubitCalib **15**
6.1. Motivation for the Project 15
6.2. Structure and Overview 17
6.3. Interface 19
6.4. Combining LabVIEW and Mathematica 21

7. Conclusion **23**

8. Outlook and Acknowledgements **23**

A. LabVIEW Block Diagrams **26**

1. Introduction

Theoretical physicists have worked out ground-breaking implications of quantum communication and quantum computation over the last few decades. Simulation of physical quantum systems and implementation of highly efficient quantum algorithms are accessible with quantum computers. For example, P. Shor developed a quantum algorithm for factoring large integer numbers in polynomial time [1] - on a classical computer, the complexity of the problem is exponential.

Using transistors, information is stored in bits that can take the values 0 and 1 on a classical computer. A quantum computer requires a quantum mechanical two-level system with the states $|g\rangle$ and $|e\rangle$. The difference to a classical bit is, however, that the system can not only take on $|g\rangle$ and $|e\rangle$ but any superposition of both states. Such quantum bits are called qubits and their practical implementation is one of the major challenges in current research.

Experimentally, qubits can be realized in many different ways including photons, trapped ions and Rydberg atoms. In the experiment at hand, we are using superconducting qubits. These are macroscopic superconducting circuits which behave like single atoms. Superconducting qubits are solid state devices and by coupling to their environment they are subject to fluctuations. In order to account for variations in the system, we have to calibrate the system regularly. For this project, we were working on the teleportation experiment presented in [2], which uses three qubits. In this setup, calibration has to be done daily in order to assure good quality of the measurement outcomes.

As we advance towards quantum communication and quantum computation, experiments will include more and more qubits. Our experiment uses three superconducting qubits and a calibration of all qubits takes up to 1.5 hours. This is a repetitive kind of work that doesn't give any new physical insights. As we scale the qubit number, automating the calibration routine becomes unavoidable. In this report we present a calibration program based on Mathematica and LabVIEW that automates the calibration routine for an arbitrary number of qubits.

Mathematica scripts were written with Mathematica 9 in this project and for visual programming we used LabView 2009. The TortoiseSVN revision number for the program presented here is 1293.

2. Overview of the Experimental Setup

2.1. Circuit QED

In the experiment we were working on, we used superconducting qubits coupled to superconducting waveguide resonators [3]. The coplanar waveguides are in the form of a 1 D analog to the coaxial cable. In Fig. 1, the waveguide is depicted in light blue. The energy levels of the waveguide are equally spaced.

The second component of the circuit is a two-level system behaving like an atom in the

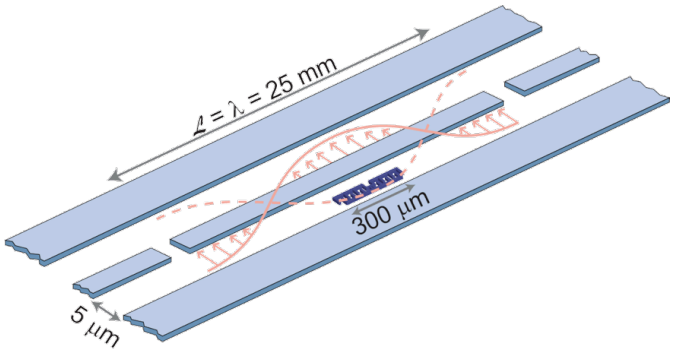

Figure 1: Layout for a single qubit in a resonator. Derived from [4].

waveguide resonator. In our setup, this component is a superconducting Transmon qubit. The Transmon qubit design presented in [5] is derived from the Cooper Pair Box (CPB). A CPB consists of an array of two Josephson junctions biased with a voltage [6]. Fig. 3 in Sec. 2.2 shows the circuit diagram of the qubit in the bottom left part, the Josephson junctions being indicated by a box containing a cross. The LC-circuit would be a harmonic oscillator but the Josephson junctions serve as a non-linear inductance. In this way, the energy levels experience an anharmonic shift and we can use the two lowest levels to represent a two-level system. The advantage of the Transmon design is that it is almost insensitive to charge noise.

The Transmon qubit is now coupled to the waveguide resonator. This system is described by the generalized Jaynes-Cummings Hamiltonian, which leads to the field of circuit quantum electrodynamics (circuit QED). For a thorough treatment of this, refer to [7]. As can be seen from Fig. 1, the qubit is placed next to the resonator. In this way, it is coupled to its modes. On the actual microchip used in the experiment, the qubits are placed at the ends of the resonators rather than in the middle. This chip is depicted in Fig. 2. It contains three resonators and four qubits, of which only three are used.

2.2. Experimental Setup

The microchip from Fig. 2 is placed on a printed circuit board (PCB), placed in a vacuum chamber and cooled down to about 25 mK in a dilution refrigerator. There, the qubits and resonators become superconducting and are well shielded from thermal noise. As we can see in Fig. 2, there are several ports that connect the microchip to the setup: The qubits (orange) are driven by their respective charge line, here shown in blue. For fast manipulations of the resonance frequency, we can use the flux line (green). Finally, the resonator ports are coloured in red.

Fig. 3 depicts the circuit diagram of the setup for one qubit. Control of the charge line is done in the orange part. The constant sinusoidal microwave signal from microwave generator (MWG) 4 serves as the LO input of a mixer. This LO is called the *upconversion LO*. A channel pair of the AWG5014 serves as the input for the quadratures I and Q, which are 90° phase

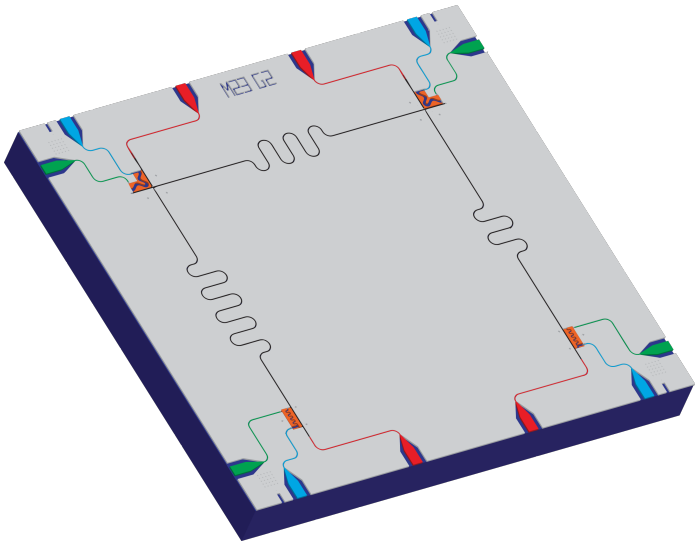

Figure 2: Microchip layout with four qubits. Taken from [2].

shifted relative to each other.

MWG 3, colored in yellow, is the resonator drive. On the other side of the resonator, the readout of the resonator is shown in green. MWG 2 serves as the so-called *downconversion LO* mixer input. The readout signal is downconverted to a frequency of 25 MHz and then sent to a Field Programmable Gate Array (FPGA) for signal analysis.

Qubit and resonator are placed in a dilution refrigerator as indicated by the dashed box. In the circuit, they are followed by a parametric amplifier (paramp), controlled via the blue part of the circuit. As it is not important for this project, we will not go into properties and control of the paramp.

With this setup, a deterministic teleportation protocol was performed on three Transmon qubits using single- and multi-qubit gates. The experiment is presented in [2]. For more detailed information on the setup, we again refer to [7].

3. Calibration Parameters

As the higher energy levels are detuned, the Transmon can be approximated as a two-level system as shown in Fig. 4. The resonance frequency between the ground state $|g\rangle$ and the first excited state $|e\rangle$ is approximately given by:

$$\omega_{01} \approx \frac{\sqrt{8E_J E_C}}{\hbar} \quad [5]$$

Here, a pulse that transfers the qubit state from $|g\rangle$ to $|e\rangle$ is called a π-pulse. As the two-level system is quantum-mechanical, we can also transfer the qubit in to the equal superposition state $\frac{|g\rangle+i|e\rangle}{\sqrt{2}}$. This operation is called a $\frac{\pi}{2}$-pulse.

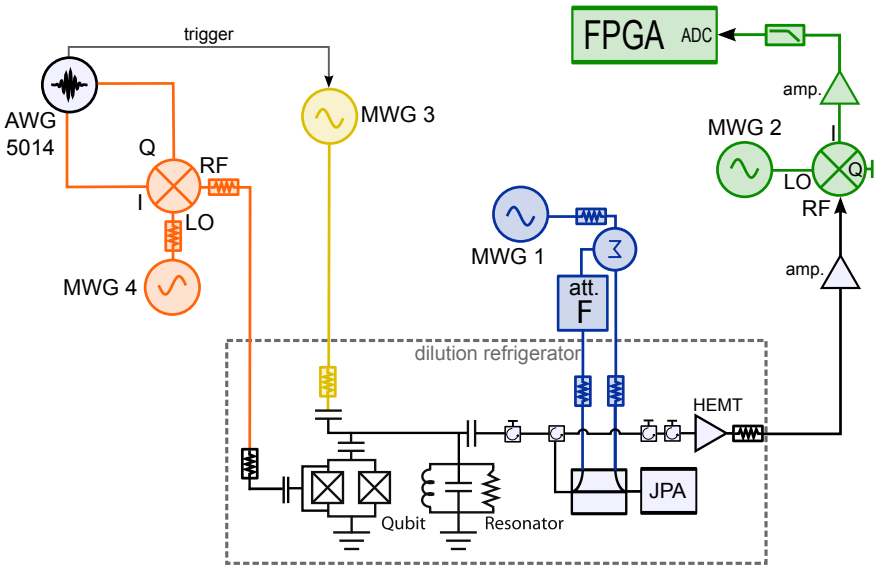

Figure 3: Circuit diagram of the setup. Adapted from [6].

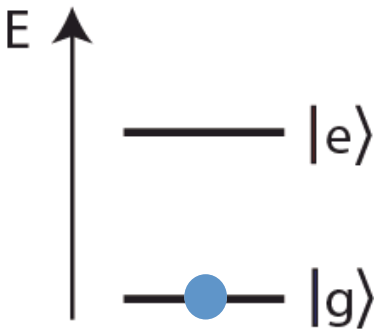

Figure 4: Two-level system. The system can any superposition state between $|g\rangle$ and $|e\rangle$.

We can also understand π- and $\frac{\pi}{2}$-pulses in a nice way using the Bloch sphere. This is depicted in Fig. 5. In the figure, the ground state is referred to as $|0\rangle$ and the first excited state as $|1\rangle$. The qubit state lies on the surface of the sphere and is pointed to with a vector (red). A π-pulse corresponds to a rotation of the state vector by 180°, a $\frac{\pi}{2}$-pulse to 90°. If not labeled otherwise, a rotation around the x-axis is meant. For our purposes, we assume that the Bloch sphere is fixed in a frame rotating with the qubit drive frequency. If the drive frequency is on resonance with the qubit, the state vector is stationary after a rotation operation on the qubit. When it is off-resonant, however, the state vector will start to precess around the z-axis.

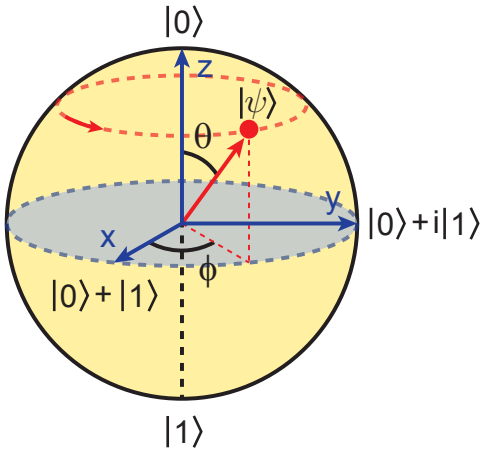

Figure 5: Bloch sphere representation of a qubit. Taken from [7].

The qubit is driven via the charge line at its expected resonance frequency ranging between 4 GHz and 10 GHz, depending on the properties of the circuit. Good control of the frequency and shape of the signal requires the use of an IQ mixer. The LO input is a CW signal adjusted 100 MHz above the expected resonance frequency of the qubit. 100 MHz signals modulated with the desired pulse shape are 90° phase shifted relative to each other and sent through the I and Q ports. Those signals are obtained from an AWG, as described in Sec. 2.2. We then drive the qubit using the left sideband of the RF (mixer output) signal. (We could also adjust the LO 100 MHz below the resonance and use the right sideband.) The frequency of the I and Q inputs is called *IF frequency* and is an important parameter in the experiment. When calibrating the qubit drive frequency, we leave the LO constant and change the IF frequency. [8]

Higher energy levels of the qubit are detuned from the resonance frequency ω_{01}. However, it turns out that these levels induce an undesired phase shift when we apply pulses to the qubit. Motzoi et al. [9] investigated this matter and came up with a solution: A special pulse shaping called DRAG (derivative removal by adiabatic gate) can be used to account for the extra phase. In our experiment, the DRAG pulse is calculated using the so called QScale parameter q_s.

The environment induces fluctuations in the superconducting qubits. This leads to small shifts in the resonance frequencies of the qubit. Normally, the shift is below 1 MHz a day. In order to

take care of this, the π- and $\frac{\pi}{2}$-pulses of the qubit, IF frequency and QScale parameter q_s have to be calibrated daily. In the following we will call those parameters "calibration parameters". We can determine the calibration parameters from certain measurements as presented below. Additionally, there is a calibration (named CalTom) that tests the quality of the single-qubit operations. Note that rotations of a qubit such as π- and $\frac{\pi}{2}$-pulses are often referred to as *single-qubit gates*.

4. Calibration Measurements

4.1. Rabi Measurement

With the Rabi measurement we can determine the amplitude of the π- and $\frac{\pi}{2}$-pulses. The pulse pattern is - as for all other measurements - saved in a file that is loaded onto the respective AWG. For this measurement, we apply pulses of a fixed time interval and increasing amplitude. This leads to oscillations of the qubit between the ground state $|g\rangle$ and the excited state $|e\rangle$. After the pulse, we apply a measurement signal. The pattern and the measurement results are depicted in Fig. 6. The $\frac{\pi}{2}$-pulse is given by the pulse amplitude for which we measure the excited state as often as the ground state (i.e. $|e\rangle$ population is 0.5) for the first time. The π-amplitude corresponds to the pulse amplitude yielding the maximum $|e\rangle$ population for the first time. In praxis, we calculate the amplitudes from a sinusoidal fit of the data set. The fitting procedure is described more accurately in section 5.

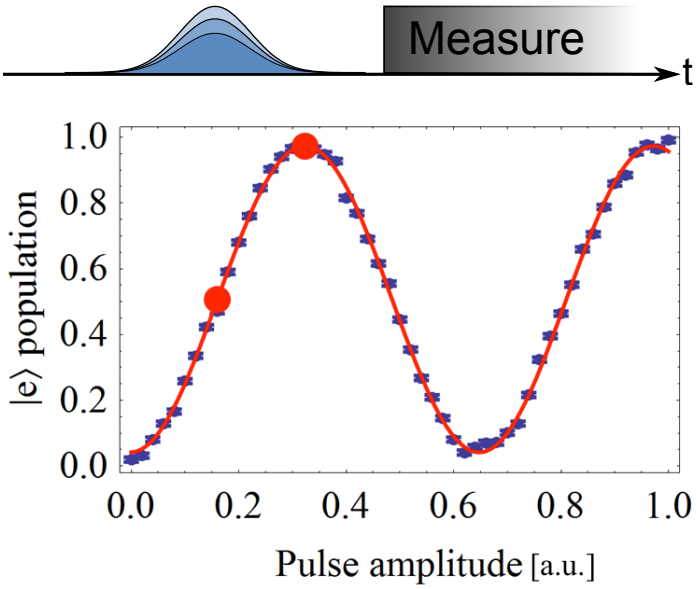

Figure 6: Pulse sequence and data plot for the Rabi measurement. The π- and $\frac{\pi}{2}$-pulse amplitudes are found by fitting the data.

4.2. Ramsey Measurement

By using the so called Ramsey fringe oscillations, we can experimentally determine corrections to the IF frequency. Two $\frac{\pi}{2}$-pulses - with amplitude known from a Rabi measurement - are applied to the qubit, as shown in Fig. 7. Between the pulses, there is a time delay Δt and the second pulse is followed by a measurement. The pulses are detuned by 4 MHz from the qubit resonance frequency. As a result, the qubit picks up an extra phase during the time Δt. In the Bloch sphere picture, this can be seen as a precession around the z-axis with a frequency of 4 MHz. The measurement outcome therefore depends on Δt, as shown in Fig. 7. The final state is expected to oscillate with 4 MHz. If that is not the case, the qubit resonance frequency has shiftet. This shift can be found by fitting the data and determining the frequency of the sinusoidal fit. Now we can adjust the IF frequency to the new qubit frequency.

Due to interaction with its environment, the qubit experiences dephasing and thus loss of quantum information [7]. A characterisctic time constant for this process is the dephasing time T_2^*, which can also be determined from the Ramsey measurement. It is given as the inverse exponential coefficient of the oscillations' enveloping curve. As an information for the experimentalist, the decay time will also be determined in the calibration and shown in the calibration log file. The decay time has an important practical relevance for our experiment: In order to execute the teleportation protocol successfully on the qubits, T_2^* needs to be much longer than the protocol.

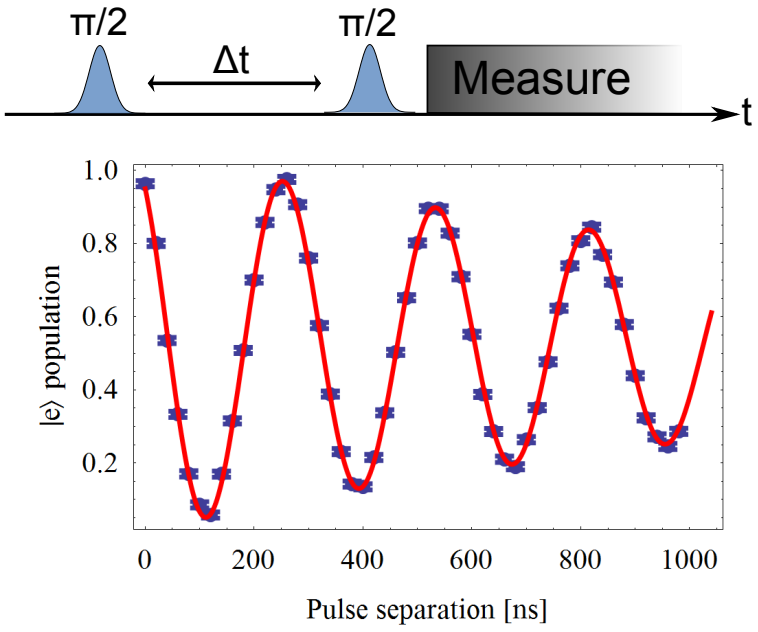

Figure 7: Pulse sequence and data plot for the Ramsey measurement. The $\frac{\pi}{2}$-pulses are off-resonant by 4 MHz to the expected qubit frequency. The effective drive frequency (IF frequency) is adjusted from the frequency of the fit.

4.3. QScale Measurement

The aim of the QScale measurement is to account for extra phases that are induced to the qubit by the presence of higher energy levels. The qubit drive signal is given by

$$\varepsilon_x(t)\cos(\omega_d t) + \varepsilon_y(t)\sin(\omega_d t),$$

where ω_d is the qubit drive frequency. ε_x and ε_y are chosen in such a way that the signal is Gaussian and cancels the AC Stark shift error. More information on this and the formulas for ε_x and ε_y can be found in [7].

Now we want to practically implement the DRAG pulse shaping discussed in section 3. In the measurement, we do this by scaling ε_y by a factor of q_s, where the QScale parameter q_s is swept from -1.5 to 1.5. For every value of q_s, we do three qubit operations, each one followed by a measurement: We first do a $\frac{\pi}{2}$-rotation around the x-axis followed by a π-rotation either around x, y or -y. Theoretically, all the operations should result in an average excited state population of 0.5. The measurement results are shown in Fig. 8. The blue dots show the results for both rotations around the x axis. The violet dots result from the second rotation around y and the yellow dots from the second rotation around -y. The rotations about y and -y are not as reliable due to the phase error we discussed above. However, we can choose q_s in such a way that it accounts for the phase error. The optimal scaling parameter is found at the intersection of the three lines in the plot.

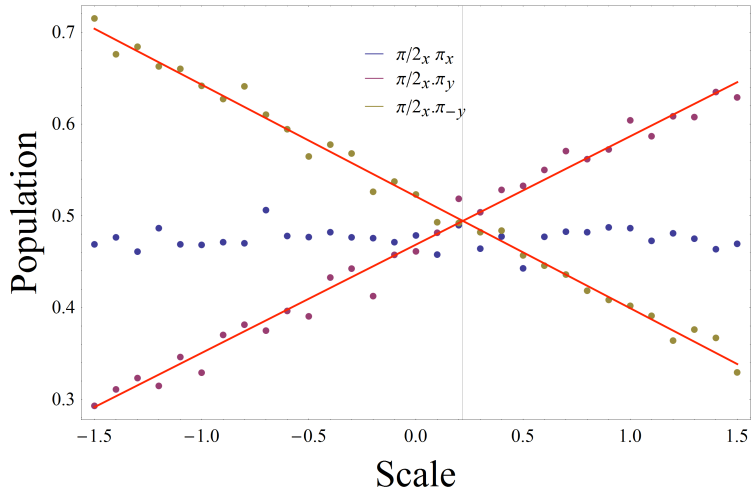

Figure 8: Average excited state populations for the QScale measurement.

4.4. CalTom Measurement

The calibration tomography (short "CalTom") measurement tests the fidelity of single-qubit gates. After finishing calibration of the parameters, we are interested in how reliable the qubit rotations are. In order to test this, we apply 25 different rotation patterns to the qubit

and afterwards measure the excited state population. The measurement results are shown in Fig. 9. The red dots show the expected excited state populations, the blue dots give the average measured state. If the measured populations are significantly off the expected ones, the calibration was not successful.

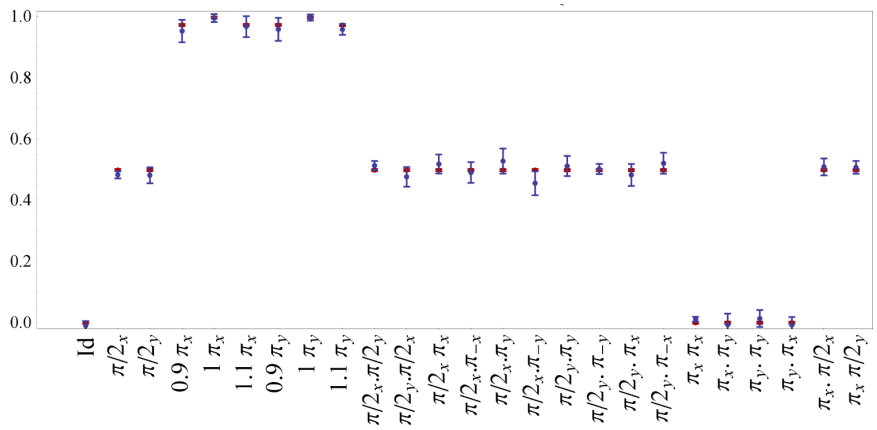

Figure 9: Calibration tomography of a qubit showing the fidelity of different qubit operations.

5. Data Analysis Techniques

From the Rabi and Ramsey measurement outcomes, we calculate the IF frequency, π and $\frac{\pi}{2}$ amplitudes with a sinusoidal fit. The fit function is given by

$$f(t) = y_0 + Ae^{-\frac{t}{\tau}} \cos\left(\frac{2\pi}{\lambda}t - \frac{\pi}{2}\delta\right). \tag{1}$$

The argument t corresponds to the pulse separation time in Ramsey measurements and to the pulse amplitude in Rabi measurements. The frequency of the oscillations is given by $\nu = \frac{1}{\lambda}$, so we have to add $4\text{MHz} - \nu$ to the IF frequency to adjust it. When we do the fit on the Rabi measurement data, the π pulse amplitude is given by $\delta\frac{\lambda}{4}$ and the $\frac{\pi}{2}$ pulse amplitude by $(\delta - 1)\frac{\lambda}{4}$. The respective calibration parameters are therefore easily calculated once we found the correct fit parameters.

The challenge of our data analysis lies in finding good initial values for the fit. We fit both Rabi and Ramsey data sets with the same function (1). The analysis is done in a Mathematica script (".m" file extension) and the function `NonlinearModelFit` is used to fit it to the data. Before the start of the semester project, the initial values for y_0, A, τ, λ and δ were hardcoded in the script. As `NonlinearModelFit` is sensitive to how accurate the initial values are, the fit often failed, for example as in Fig. 10. This did not pose a problem because the data analysis was conducted by hand and the initial values could be adjusted such that the fit converges properly. In QubitCalib, however, the analysis was to be done automatically, so we could not re-adjust the initial values manually. Still, we had to be sure that the fit works for every reasonable Ramsey and Rabi data set. (We define "reasonable" data sets as measurement

outcomes that clearly show Ramsey or Rabi oscillations.) In the following we present some methods we used to find initial values that are close to the definite values.

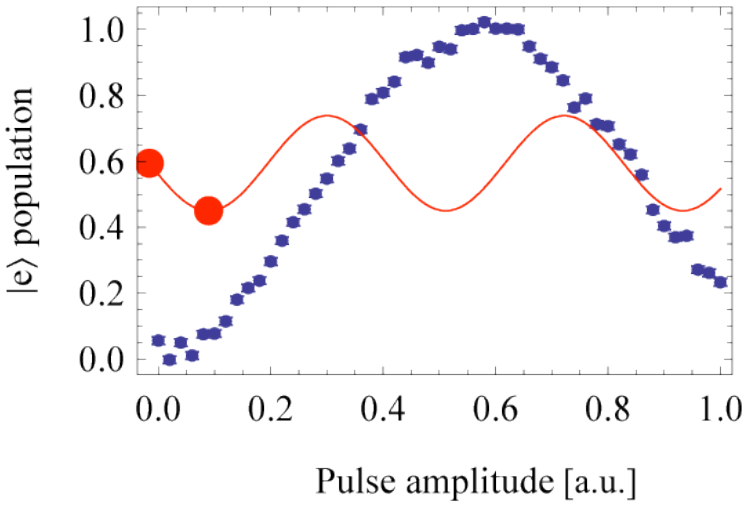

Figure 10: Rabi data fit that failed due to inaccurate initial values.

In order to estimate the parameters, we use the properties of the data set. The offset y_0 is approximately given by the minimum population and the amplitude A is the difference between maximum and minimum population measured. The inital value τ_{initial} for the decay time is left hardcoded. We found out that $\tau_{\text{initial}} = 2000$ for Rabi and $\tau_{\text{initial}} = 800$ for Ramsey works for every reasonable data set we tried (units in nanoseconds for Ramsey and arbitrary units for Rabi). As for the period λ, one may note that the main frequency of an oscillating data set is given by the peak frequency of its discrete Fourier transform. As an example, we show the Fourier spectrum of the data set of Fig. 10 in Fig. 11. Defining the frequency corresponding to the maximum amplitude as ν_{max}, we can estimate the period as $\lambda_{\text{initial}} = \frac{1}{\nu_{\text{max}}}$. In the figure, we plotted the absolute value of the amplitudes. Generally, however, the amplitudes are complex. Denote by B_{max} the amplitude with the largest absolute value. It turns out that the initial phase δ is approximately given by $\delta_{\text{initial}} = \arg(B_{\text{max}})$. Especially the last two estimates of λ and δ contributed to the reliability of the data analysis and the fit has converged properly for every reasonable data set ever since.

6. The Calibration Program QubitCalib

6.1. Motivation for the Project

The idea to write a calibration program for superconducting qubits was not new in the Quantum Device Lab of Prof. Wallraff. Previously, Jonas Mlynek had written a program called Autocalib that executed a Rabi-Ramsey-Rabi array of succeeding measurements to adjust the π- and $\frac{\pi}{2}$-pulses and the IF frequency. This already leads to one of the big challenges of the project. As all the other setups in the Qudev group, the teleportation experiment is run using

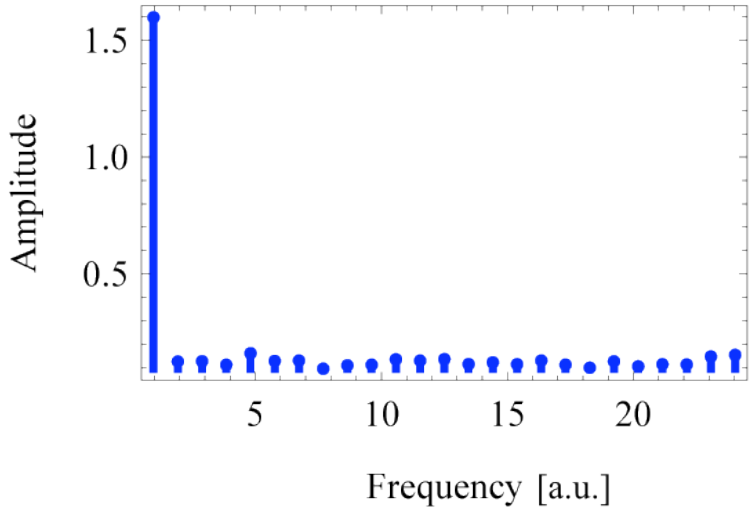

Figure 11: Discrete fourier transform of the Rabi data set.

the visual programming software LabVIEW. It is used to control the setup and collect the data. However, other procedures including pattern generation and data analysis are done with Mathematica.

When we do a measurement, we first have to generate a pulse pattern that is to be applied to the qubit. This is done with Mathematica scripts and requires the π- and $\frac{\pi}{2}$-pulse amplitudes, IF frequency and QScale parameter q_s. Those values are stored along with other pulse settings in a config file called *pattern config*. The pulse patterns are loaded onto the respective AWG (arbitrary waveform generator) for the qubit, the data sets are then acquired via LabVIEW and stored on a specific drive F. For the analysis, the data set is called from F with Mathematica and analyzed, for example as described in section 5. As can be seen from this measurement overview, both LabVIEW and Mathematica perform vital tasks in the measurement process. A calibration program thus needs to combine both programs. Jonas already implemented a solution to this in Autocalib. The solution was not so handy to the experimentalist, though, as several steps had to be done besides operating the program interface. Furthermore, we were looking for a solution that could easily be extended to additional modules.

Autocalib was well suited for the task of a Rabi-Ramsey-Rabi procedure on the respective experiment. Our vision was to rewrite this into a more complete program that allows for an arbitrary order of the calibration modules - Rabi and Ramsey as well as QScale and CalTom - executed for an arbitrary number of qubits and resonators. In this way, the experimentalist would be able to decide which qubit needed calibration and if it needed a full calibration or just, say, an adjustment of the QScale parameter. Moreover, we wanted to find a way to simplify the interaction between LabVIEW and Mathematica. This is described in section 6.4. In the end we were interested not only in the new values of the calibration parameters but also in how they came about. This triggered the idea of a log file showing the data sets and fits that were generated in the calibration procedure.

6.2. Structure and Overview

Before the execution of QubitCalib, the qubit properties have slightly shifted, so the calibration parameters (π- and $\frac{\pi}{2}$-pulse amplitudes, IF frequency and QScale paramter q_s) are not accurate anymore. We now choose which qubit we want to calibrate and specify an order of calibration modules (Rabi, Ramsey, QScale, CalTom) that is to be executed on every qubit. A typical sequence for a full calibration is Rabi-Ramsey-Rabi-QScale-Rabi-Caltom and takes about 1.5 hours on our experiment when we don't use a parametric amplifier. (For information on the use of a parametric amplifier in the teleportation experiment, cf. [2].)

After running the calibration program, the respective calibration parameters in the pattern config file are corrected and a log file in .pdf format is shown. It contains the data plots and fits for the conducted measurements. Each plot is labeled with the name of the corresponding data set as it appears on the data storage drive F. It is of the format "MeasurementNumber_ModuleNameChannelPair_cal", where the token "cal" indicates that this was a calibration measurement conducted by QubitCalib. The calibration parameters corrected in this step are written below the plot. For a Ramsey measurement, QubitCalib also gives the coherence time T_2^* of the qubit. A sample log file for a Ramsey-Rabi-QScale-Rabi-CalTom calibration is shown in Fig. 12.

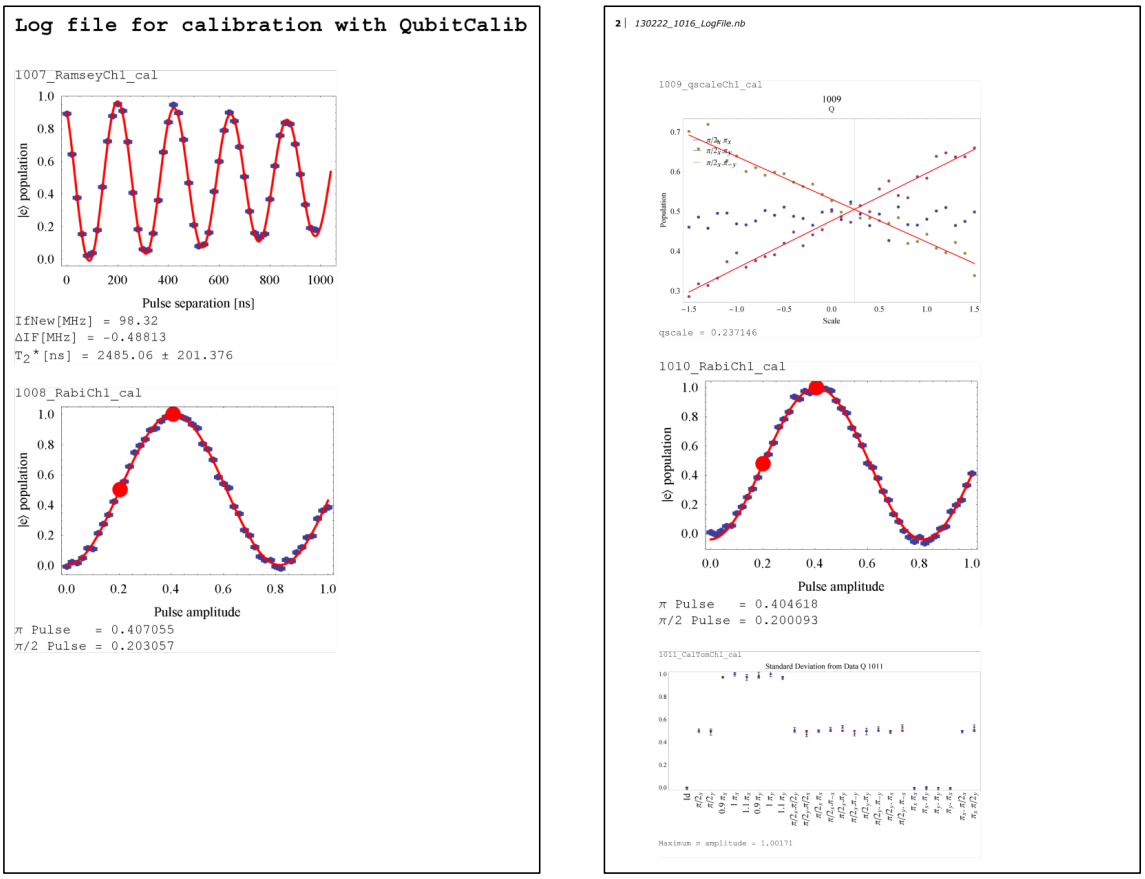

Figure 12: Sample log file for a Ramsey-Rabi-QScale-Rabi-CalTom calibration.

In Fig. 13 we present the structure of QubitCalib. We start with the first qubit to calibrate and execute the first calibration module, e.g. a Ramsey measurement. First we generate the pulse pattern that is applied to the qubit. This is done in Mathematica. The pattern is then loaded onto the respective AWG. The setup is controlled with a LabVIEW VI called Cleansweep. We load the measurement configurations into Cleansweep and start the measurement.

What measurement configurations do we load? The background to this is that a lot of settings have to be specified for the setup in Cleansweep, including hundreds of parameters. Since most of the parameters are the same for all Rabi, Ramsey, QScale and CalTom measurements, we decided to import the Cleansweep configuration from a recent measurement. After a measurement, the Cleansweep settings are saved on F under the data set name. In this way, we just have to go back every now and then and specify a configuration. With "recent" we mean that the setup must not have changed since the measurement we specify. When we have loaded the configurations, we just have to adjust the few settings that are specific to calibration module and qubit. This includes the path to the pattern sequence, the file number for the measurement and the FPGA channel. The qubits are driven by different AWGs and MWGs, so we also have to turn on the proper generators.

In the next step, we call Mathematica to do the data Analysis. The corrected configuration parameters are saved in .txt files that are located in a temp file in the QubitCalib program folder. Mathematica also outputs a combined figure of data set name, data plot, sinusoidal fit and corrected calibration parameters. This figure will be assembled with similar figures from the succeeding measurements to make the log file. The text files with the new parameters are read back into LabVIEW to rewrite the pattern config. This is done by opening the pattern config file and using regular expressions to find and replace the old values. A sub VI called `Rewrite Pattern cfg ModuleName` adapted from Jonas' Autocalib takes care of that. `ModuleName` is replaced by `Rabi`, `Ramsey`, `QScale` respectively. (The CalTom module does not correct any parameters but only tests the calibration.)

We now proceed to the next calibration module, e.g. a Rabi measurement. The module follows the same structure: We generate the pattern for the measurement, run the measurement in Cleansweep, analyze the data and rewrite the pattern config. Note that the pattern generation script already uses the new values for the π and $\frac{\pi}{2}$ pulse amplitudes, IF frequency and QScale paramter q_s if they have been updated in a previous module. We proceed through all the modules we specified beforehand, finally finishing the calibration of the first qubit. Then we go on to the next one. In the program code, we construct an array of the channel numbers of the qubits we want to calibrate and another array of the desired modules. We then loop over all the channels and within each channel we loop over the calibration modules.

In appendix A we present the LabVIEW block diagrams of QubitCalib and - as an example for a calibration module - of the Rabi module. QubitCalib has a modular structure. A new calibration module can easily be included by copying the Rabi module and replacing the pattern generation and analysis scripts.

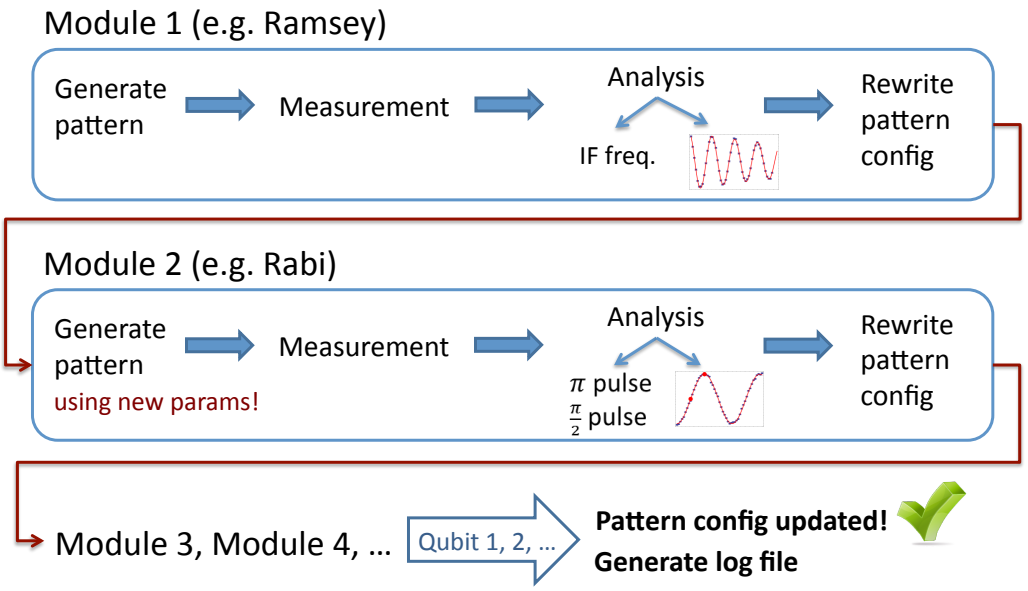

Figure 13: Overview of the QubitCalib program structure.

6.3. Interface

In order to adjust and run the calibration procedure, the user just needs to access the "Control Panel" of the QubitCalib interface shown in Fig. 14. Each qubit corresponds to a channel pair, as described in Sec. 2.2. To activate calibration for a qubit, we click on the LED next to the channel pair. There is some information we have to specify about the setup for each qubit: The microwave generator (MWG) for the upconversion LO input, the MWG for the downconversion LO input, the resonator drive MWG and the FPGA channel used for the readout. Those inputs only need to be changed when the setup was altered, so we normally don't have to worry about them. Below this input array, we have to specify a "Recent Rabi or Ramsey Config". Here we input the path to the config file of a recent Rabi, Ramsey, QScale or CalTom measurement. The meaning of this config file is explained section 6.2.

In the horizontal array at the top, we choose the calibration modules we want to use in the calibration. We can create an arbitrarily long sequence of Rabi, Ramsey, QScale and CalTom modules. By clicking on the LED below the module, we can enable or disable the module for the calibration. Next to the module array, a STOP button is located. When the button is pressed, the current calibration module is finished but all subsequent modules are skipped. A log file is created from all the measurements that have been completed.

The output arrays are located in the right middle part of the interface. For each channel that was calibrated, it displays the corrected calibration parameters that were written to the pattern config. If the parameter was not calibrated, the indicator is set to zero. If the parameter was calibrated multiple times in the calibration procedure, the last correction is displayed.

A convenient feature of QubitCalib is located in the bottom left part. The *Timer* allows us to set the time we want to start the calibration. If we enable the timer with the respective

LED and start the program, QubitCalib will be executed on the preset time and date. This is often used to calibrate the setup at 6 a.m. The experimentalist then finds a freshly calibrated setup in the morning.

The *Status* window at first reports the start time of the calibration. During the calibration, it is updated to display which channel is currently calibrated and which module is executed. Finally, the calibration end time is output. Next to this, we located the typical LabVIEW *Error out* status window.

The interface also includes a note that the pattern config is backuped before the calibration. In practice, a copy of the pattern config is moved to a backup folder specified in the *Calibration Directories* before the original config file is manipulated by the program. If something goes wrong in the calibration, the updated calibration parameters will very likely be wrong. However, we can easily recover the old ones from the backuped pattern config.

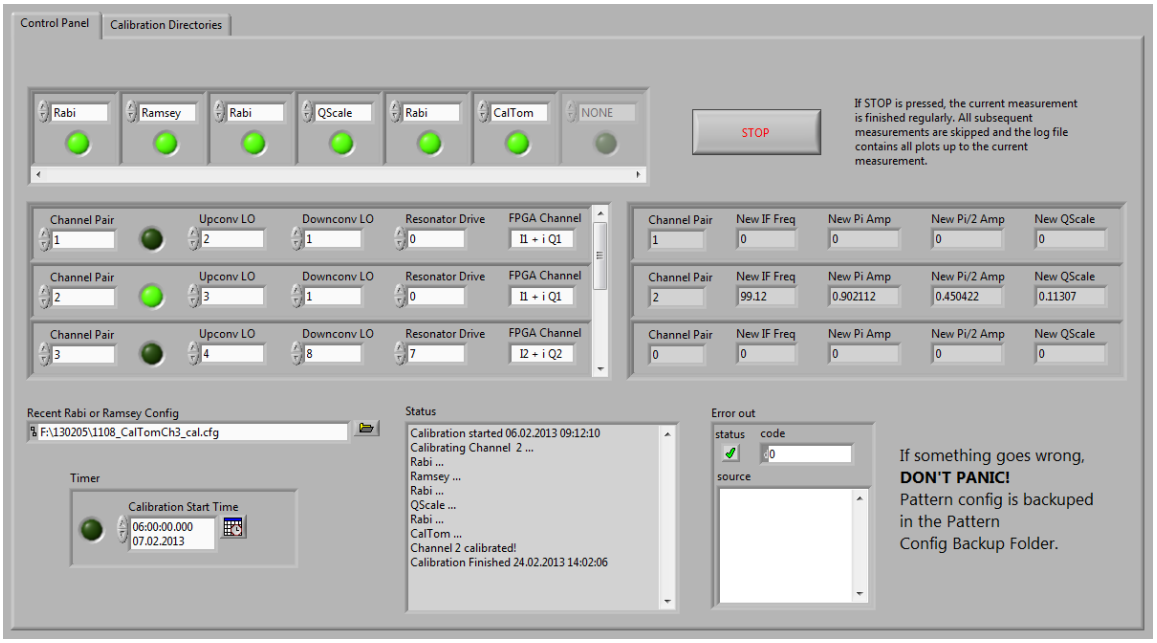

Figure 14: *Control Panel* of the QubitCalib interface. Here we specify the channels we want to calibrate and the calibration modules. The updated calibration parameters are dispayed and a status window tracks the progress of the calibration.

The second part of the QubitCalib interface is a list of directories contained in the *Calibration Directories* panel. This is shown in Fig. 15. Unless we want to change the file structure of the program, we leave this part of the interface unchanged. In the *Data Storage* section we have to specify the path to the data repository. The data saving folder of the format 'YYMMDD" and the measurement number for this day will be found automatically. In the *General* section next to this, we specify the path to QubitCalib on the computer, the path to the Mathematica MathKernel in our installation and the output folder for results generated with the Mathematica scripts. The latter one is by default located within the QubitCalib program folder. *Config Files* includes the paths to the pattern config and its backup folder, *Log File* the Mathematica script that generates the log file and the folder where the file is

saved. *Pattern Generation* lists the pattern generation scripts and *Analysis Scripts* the data analysis scripts. Note that the interaction between LabVIEW and Mathematica might fail if paths containing spaces are not put in quotation marks.

Since all the paths used in QubitCalib can be changed, the file structure of QubitCalib is very flexible.

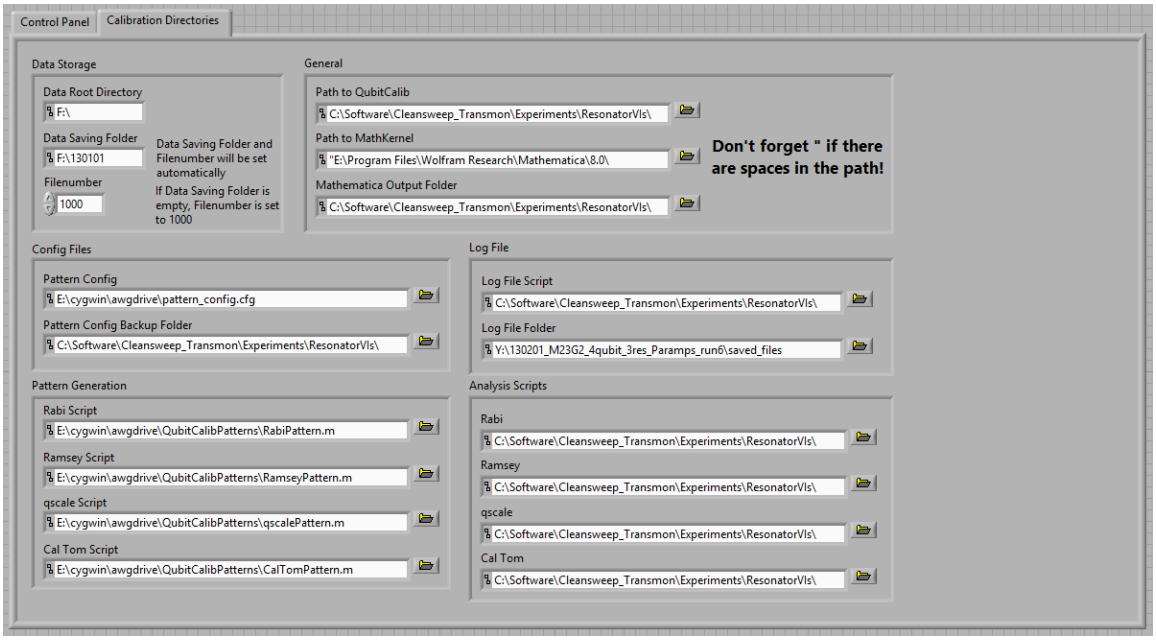

Figure 15: The *Calibration Directories* panel of the interface contains paths such as the one to the pattern config and the Mathematica analysis scripts. In everyday use of the program, thes part of the interface is left unaltered.

6.4. Combining LabVIEW and Mathematica

As stated above, we need to have an interaction between LabVIEW and Mathematica in order to run the pattern generation and data analysis scripts. More precisely, we need to execute a Mathematica script and pass arguments to it.

Normally, Mathematica is used with the notebook-interface (.nb file extension). Another way to execute Mathematica is to save it as a Mathematica script (.m file extension) and then run it from the command line. To do so, we need to call the Mathematica Kernel `MathKernel.exe`, which is by default in the same folder as the familiar application `Mathematica.exe`. The command line input looks as follows:

```
C:\>cmd /c "Program Files\Wolfram Research\Mathematica\9.0\Mathkernel.exe"
-script C:\Software\Cleansweep_Transmon\Experiments\ResonatorVIs\QubitCalib
\scripts_mathematica\rabi.m
```

The first path specifies the path to the Mathematica Kernel, the second path denotes the script we want to execute. In this example, we are calling the Rabi analysis script. A nice

feature of Mathematica is that we can pass arguments to the script by writing them into the command line after the script file path, separated by spaces. To properly execute an analysis script, we need to pass the saving folder for the data set, measurement number, output folder and an index. The output folder is the saving folder for the new parameters and the plot. The index labels the plot that is created. In this way, the log file generation script can properly assemble all the plots in the end. The complete command line input is therefore given by:

```
C:\>cmd /c "Program Files\Wolfram Research\Mathematica\9.0\Mathkernel.exe"
-script C:\Software\Cleansweep_Transmon\Experiments\ResonatorVIs\QubitCalib
\scripts_mathematica\rabi.m\ S:\130128\ 3006 C:\Software\Cleansweep_Transmon
\Experiments\ResonatorVIs\QubitCalib\Output\ 1
```

The arguments can now be imported by using the command `$Commandline` in the script. When executed, the Mathematica analysis scripts store the corrected calibration parameters in text files in an output folder within the QubitCalib program folder. Using the VI `Read from Text File`, we can read them back into LabVIEW to adjust the pattern config. A scheme of the interaction between LabVIEW and Mathematica is given in Fig. 16.

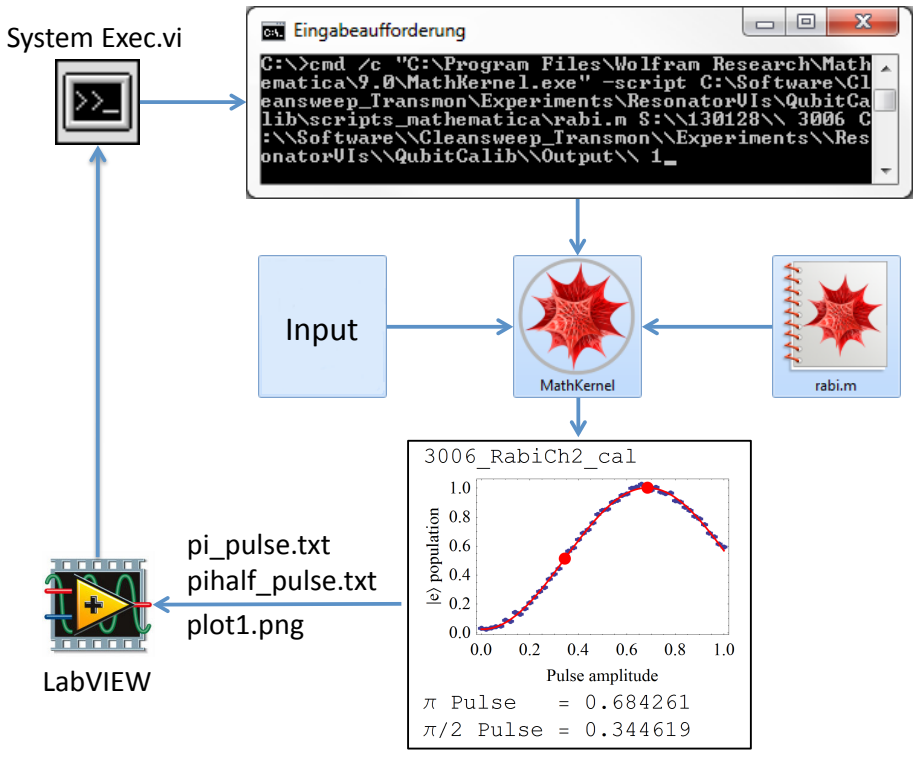

Figure 16: Interaction of LabVIEW and Mathematica. We use the VI `System Exec` to write directly into a command prompt.

7. Conclusion

We succeeded in writing a calibration program for an arbitrary number of qubits and resonators. QubitCalib executes the single-qubit gate calibrations Rabi and Ramsey, the DRAG pulse shaping routine QScale and the calibration tomography CalTom in any desired order. After execution of the program, all the calibration parameters needed for single qubit gates have been updated and a log file containing the data analysis results is shown.

Crucial steps in developing QubitCalib were simplifying the interaction of LabVIEW and Mathematica - both of which are needed in the measurement and analysis process - and assuring a reliable data analysis. An easily manageable front end has been programmed, containing some additional small features such as a timer. The backend of the program has been designed such that it displays a modular structure. In this way, new calibration modules can be included easily by adding new pattern generation and analysis scripts and doing some adjustments in LabVIEW.

QubitCalib was developed and is currently used on a teleportation experiment with Transmon qubits in the Qudev group at ETH Zurich. The calibration program has two positive effects on the experiment: In a full calibration with a Rabi-Ramsey-Rabi-QScale-Rabi-CalTom sequence for all three qubits, QubitCalib executes a 1.5 h measurement and analysis procedure which would otherwise have to be conducted manually. Secondly, as QubitCalib is easy to handle, the setup is calibrated more frequently, leading to higher precision of the whole experiment.

8. Outlook and Acknowledgements

Since I finished work on the project, development of QubitCalib has been going on. Two new semester students - Andreas Landig and Johannes Heinsoo - have put a lot of effort into improving and extending the program. The current interface is shown in Fig. 17. It includes a lot of new modules: RabiEF, RamseyEF, QScaleEF, CalTomEF which extend all the measurements we did in our project to excitations between the first and second excited levels $|e\rangle$ and $|f\rangle$. Modules that measure the decay time T1, another important time constant, have been included as well. Furthermore, Andreas and Johannes started to realize multi-qubit gate calibrations. The controls for those advanced operations are located in the bottom left corner of the control panel in figure Fig. 17.

Another important change in QubitCalib is the introduction of a new data flow. In an effort to improve the modular structure of the program, the code was developed to make the LabVIEW code completely independent of the type of the calibration module. This is achieved by outsourcing information that is characteristic to a module into a file called QubitCalib.ini. Moreover, the new version is more watchful for errors and allows for error correspondence between LabVIEW and Mathematica.

Further development and daily use of QubitCalib to calibrate the setup indicate that this was a successful semester project. I want to thank Andreas Wallraff, Lars Steffen, Yves Salathe,

Figure 17: Current QubitCalib interface. A lot of work has been going on since we finished this semester project.

Markus Oppliger, Philipp Kurpiers, Jonas Mlynek and Christian Lang for making this Project possible and helping me with all kinds of computational, theoretical and practical problems.

References

[1] P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, in *Proceedings, 35th Annual Symposium on Foundations of Computer Science, Santa Fe*, 124. IEEE Computer Society Press (1994).

[2] L. Steffen et al., arXiv:1302.5621 (2013).

[3] A. Wallraff et al., Nature **431**, 162-167 (2004).

[4] A. Blais et al., Phys. Rev. A **69**, 062320 (2004).

[5] J. Koch et al., Phys. Rev. A **76**, 042319 (2007).

[6] P. Kurpiers, *Enhanced Single-Shot Readout for Quantum Feedback Experiments*, Master Thesis, ETH Zurich, 2013.

[7] M. Baur, *Realizing Quantum Gates and Algorithms*, PhD Thesis, ETH Zurich, 2012.

[8] Y. Liu, *Implementation and Characterization of 16-port Devices in Circuit Quantum Electrodynamics*, Master Thesis, ETH Zurich, 2012.

[9] F. Motzoi et al., Phys. Rev. Lett. **103**, 110501 (2009).

A. LabVIEW Block Diagrams

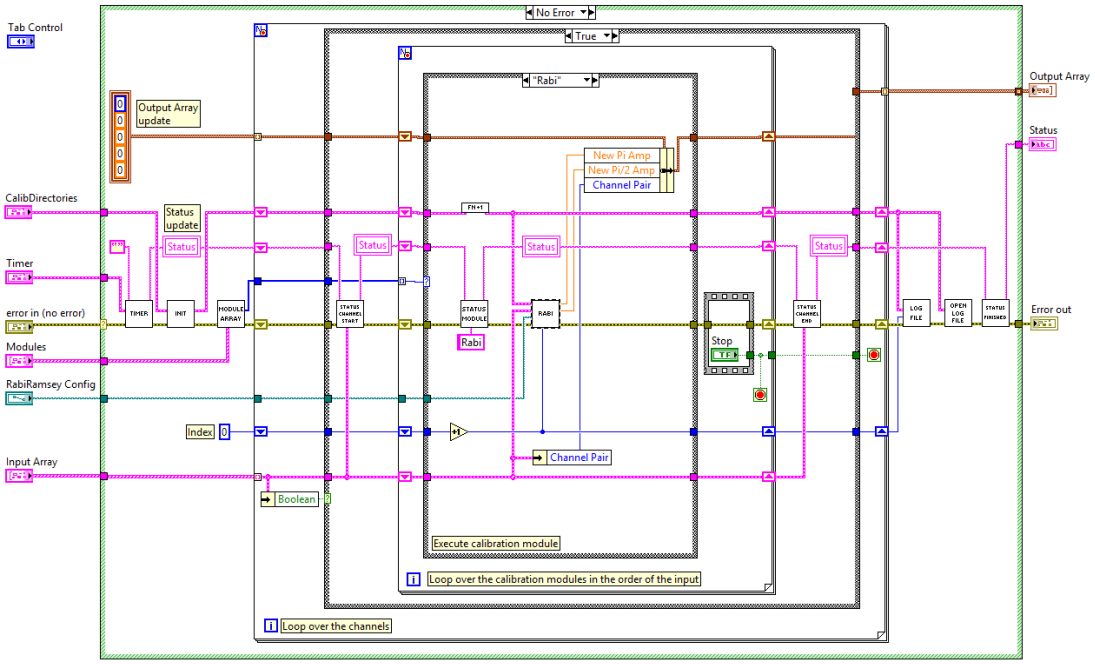

Figure 18: LabVIEW block diagram for QubitCalib.

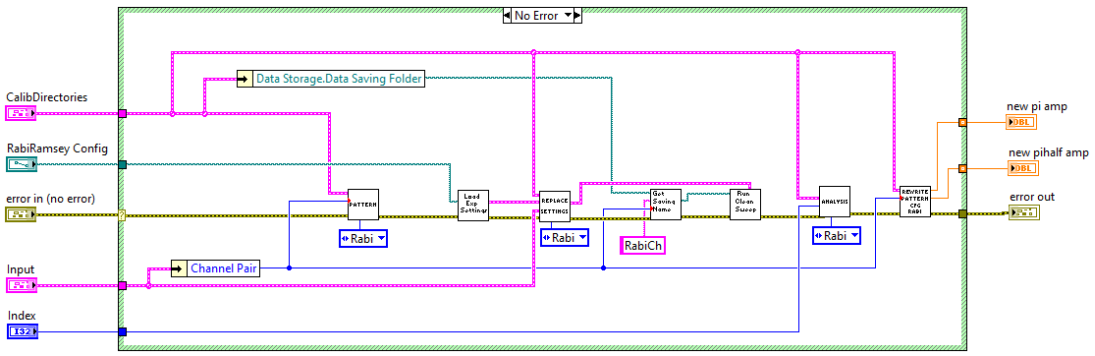

Figure 19: LabVIEW block diagram for the Rabi module.