

UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



EXTENSIBLE HARDWARE FOR CONTROL OF SUPERCONDUCTING QUBITS

BACHELOR THESIS

Written by *Daniel Ramyar*
June 11, 2017

Supervised by
Karl David Petersson

Center for
Quantum
Devices





FACULTY: Faculty of Science

INSTITUTE: The Niels Bohr Institute

DEPARTMENT: Center for Quantum Devices

AUTHOR: Daniel Ramyar

EMAIL: njt478@alumni.ku.dk

TITLE: Extensible hardware for control of superconducting qubits

SUPERVISOR: Karl David Petersson

HANDED IN: June 14, 2017

DEFENDED: ??

NAME _____

SIGNATURE _____

DATE _____

Contents

1	Introduction	1
2	Theory	2
2.1	Qubits	2
2.1.1	The Hydrogen Atom	3
2.1.2	Superconducting Qubits	4
2.2	The Cooper Pair Box	7
2.3	Circuit QED	9
2.4	Jaynes-Cummings Hamiltonian	10
2.5	Dispersive Regime: Qubit Readout	10
2.6	Gate tunable Qubits	11
2.7	Single Qubit Rotations	11
2.8	Single Sideband Modulation	12
3	Setup	13
3.1	Proposed Setup for Qubit Control	13
3.2	IQ Mixer Optimization Algorithm	14
4	Experiment	16
4.1	Spectroscopy	16
4.2	Rabi	16
4.3	Gate tuneup / ALLXY	20
4.4	Conclusion	20
A	Mixer Optimization Algorithm Code	22

Abstract

Currently very expensive mixer hardware is used in order to control and readout single qubit devices. This will pose an issue to future scalability and projects as multiqubit devices are getting made and more mixers are needed. Looking for more scalable solutions this thesis has therefore set out to test much cheaper and more scalable mixer hardware.

To benchmark the new setup we replaced the currently used internal mixers of the Rohde&Schwarz RF source with the new mixer hardware and did spectroscopy, rabi oscillation and ALLXY measurements. All but the ALLXY test did perform on par with the old setup. We believe this was due to bad pulse modulation timings and could be addressed with fine tuning in future testing.

Chapter 1

Introduction

At the heart of all modern day computers, phones and various smart electronics lies a central processing unit (CPU), which relies on technology invented a hundred years ago, namely the transistor.

Since the invention, progress of transistor technology has been improving at a staggering pace, going from big vacuum tubes (1907) to small solid state transistors (1947) and decreasing their size and increasing their count ever since [1].

Unfortunately this progress is seeming to come to an end as transistor size are becoming so small that we are approaching quantum mechanical limits ¹.

Though all hope is not lost, it was proposed by Richard Feynman [7] doing computations using quantum phenomena, where information is expressed using qubits instead of classical bits. This would allow for much faster computations of specific problems as will be discussed in the next section. Much work has been done in the area of quantum computers and multiple propositions has since been made for the underlying technology behind the qubit.

One of these propositions is the superconducting qubit. The advantage of using superconducting qubits is scalability as they are very "simple" in nature and can be readily made with current micro fabrication techniques.

This thesis will give an introduction to the theory behind superconducting qubits as discussed in [2] and test out new extensible hardware for use in future qubit experiments.

¹<http://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338>

Chapter 2

Theory

2.1 Qubits

To understand the qubit and why they will bring exponentially faster computation speeds to certain problems, we'll start by taking a look at the classical bit (binary digit). The definition of a bit comes from information theory and represents one unit of information, which can take one of two values (usually 0 or 1). Every bit can therefore only hold one piece of information.

Using multiple bits we can do operations and solve complex mathematical problems. One of these problems is prime factorization, which is a problem of figuring out which two prime numbers was used to create a product i.e solving an equation looking like this $prime_1 \cdot prime_2 = 35$.

Now this problem might seem easy and is doable for modern computers, it gets exponentially harder to solve as the product gets larger (try solving $prime_1 \cdot prime_2 = 445051$). The reason why this problem is hard to solve for a computer is basically because the only way for a computer to find the prime factors is simply to try multiplying all combinations of prime numbers until it gets it right.

Lets say you had to solve a 1400 digit product with a modern desktop computer it would take about 6.4 quadrillion years to solve¹, which is exactly why its used in RSA-cryptography. It is simply not possible for conventional computers to solve this problem in a reasonably time.

Qubits on the other hand are represented as a quantum mechanical two-state system $|\psi\rangle = \alpha|1\rangle + \beta|2\rangle$ where $|\psi\rangle$ is the wave function, $|1\rangle$ and $|2\rangle$ is the two states of the system and α and β are probability amplitudes. Compared to the classical bit which can only be in one state at a time, a qubit can be in a superposition of two. Adding more qubits allows the system to be in a superposition of multiple states, 2^N to be exact where N is the number of qubits, which means that just by having 500 logical qubits we would be able to test 2^{500} states at once! This property is exactly why we would be able to solve prime factoring problems infinity faster[6] and why groups around the world are pursuing this technology.

¹<https://www.digicert.com/TimeTravel/math.htm>

2.1.1 The Hydrogen Atom

One natural choice for the physical representation of a qubit would be a hydrogen atom, which is a very well understood system and has exactly the properties we need. In the hydrogen atom we would use the ground- and first excited state as our qubit. To control the state of our qubit we would shine a laser on the atom with a frequency equal to the energy difference between $|0\rangle$ and $|1\rangle$ ($\Delta E = \hbar\omega_{01}$ where ω_{01} is the transition frequency). The laser would then excite the atom to its first excited state and no further because of the anharmonicity of the energy levels figure (2.1). If we keep shining our laser on it, the atom would oscillate between $|0\rangle$ and $|1\rangle$, which is what we call rabi oscillations. Knowing the cycle time of the rabi oscillations allows us to prepare the atom in its first excited state or in a superposition between $|0\rangle$ and $|1\rangle$, by having the laser on for exactly the amount of time it takes the electron to jump to the desired state.

Now say we prepared the atom in $|1\rangle$, we would want to know how long it would stay there, as it would decay because of spontaneous emission. This decay time is often denoted as T_1 in literature. Instead lets say we prepared the atom in a superposition of $|0\rangle$ and $|1\rangle$, the time in which the system stays coherent i.e how long can we wait before we no longer measure the superposition we prepared it in, is denoted T_2^* .

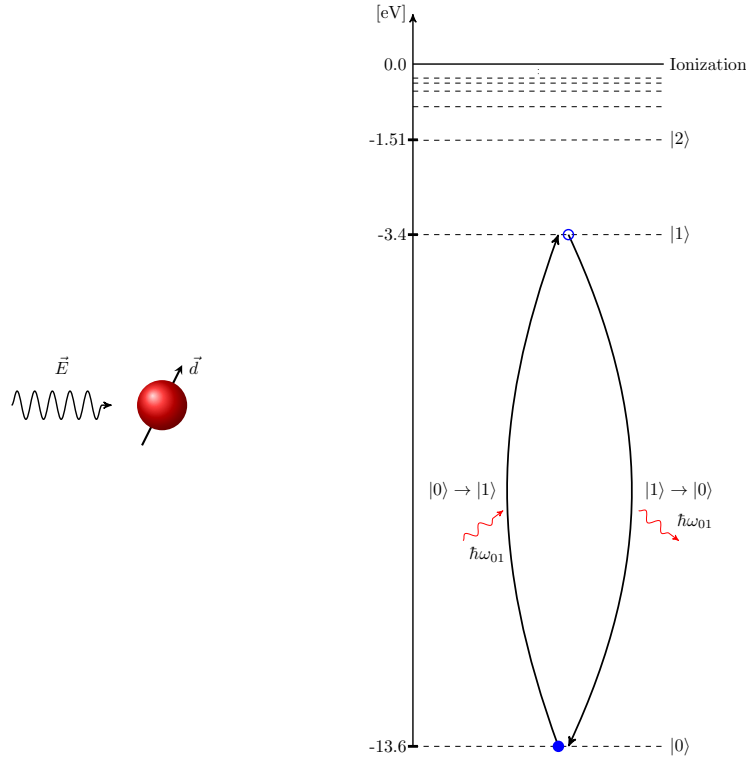


Figure 2.1: On the left we see a laser $\vec{E} = E_0 \cos(\omega_{01}t)$ shining on a hydrogen atom which will couple with the dipole moment of the atom $\hat{H} = -\vec{d} \cdot \vec{E}(t)$ and stimulate it. On the right we see the energy levels for a hydrogen atom where the laser is stimulating the $|0\rangle \rightarrow |1\rangle$ transition. Since we don't turn off the laser the electron will keep going back and forth between the ground- and first excited state which we call rabi oscillations.

2.1.2 Superconducting Qubits

The disadvantage of using hydrogen or other atoms in that case as our qubits is scalability, as it requires complex setups just to control a few qubits. That's why we'll turn our heads toward superconducting qubits, that are much easier to fabricate and scale up as discussed earlier.

The superconducting qubit also has the advantage of being made from standard electrical components consisting of wires, capacitors and inductors figure (2.2) and because the components are superconducting, we remove one source of decoherence for free, as there will be no heat dissipation due to resistivity, which would affect our T1 and T2* times.

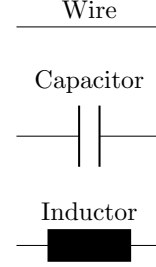


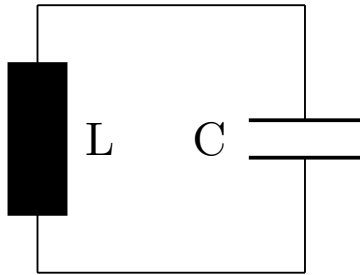
Figure 2.2: Electrical components used in superconducting qubits

LC-circuit

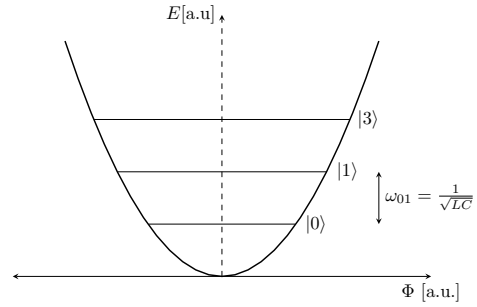
In order to create our superconducting qubit we will need a quantized system, therefore we will start looking at a quantized LC-circuit. We have two components in a LC-circuit, an inductor and a capacitor. The system will look like in figure 2.3a. The energy stored in an inductor and a capacitor is given respectively by $E_L = \frac{LI^2}{2}$ and $E_C = \frac{Q^2}{2C}$, where L is the inductance, I is the current through the inductor, Q is the charge on the capacitor and C is the capacitance. So our hamiltonian for this system will look like

$$\hat{H} = \frac{\hat{Q}^2}{2C} + \frac{\hat{\Phi}^2}{2L}, \quad (2.1)$$

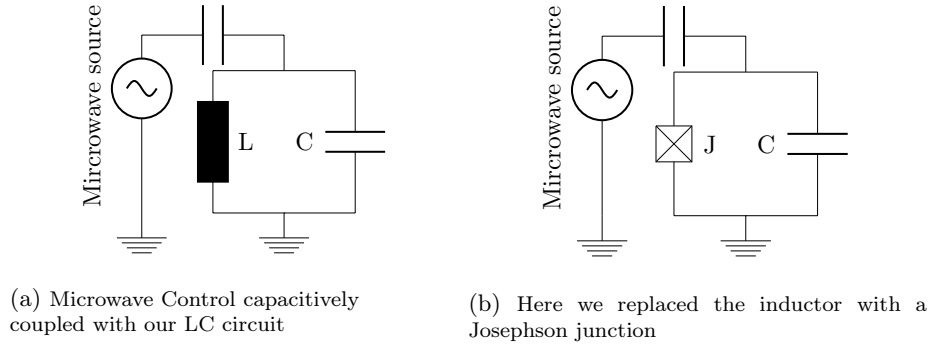
where we have rewritten the inductor energy in terms of its flux $\Phi = LI$. The solution to this equation, which is exactly the same as the harmonic oscillator, is described in figure 2.3b, where the gap between each state is $\omega_{01} = \frac{1}{\sqrt{LC}}$.



(a) The LC-circuit



(b) LC-circuit Energy levels



As we are free to choose our capacitance and inductance of the system, an ideal transition frequency would be below superconducting temperatures² and above operating temperatures, this would allow the system to automatically decay into its ground state as there would not be enough energy provided by the environment to excite our system, while still maintaining the required superconductivity. A typical transition frequency would be $\omega_{01} = \frac{1}{\sqrt{LC}} = 6 \text{ GHz} \approx 300 \text{ mK}$ with a fridge temperature of 20 mK.

To excite our quantized LC-circuit we would then capacitively couple the circuit with a microwave generator seen in figure 2.4a, which would provide a voltage of $V(t) = V_0 \cos(\omega_{01}t)$ on the system and act exactly as the laser on the atom $\vec{E} = E_0 \cos(\omega_{01}t)$.

Josephson junction

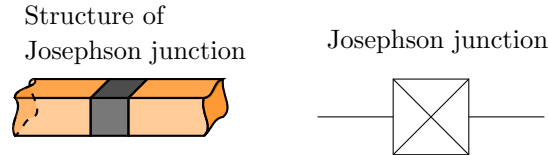


Figure 2.5: Left picture shows the superconductors (orange) sandwiching a weak link (grey), right picture shows the notation for a Josephson junction in a circuit diagram.

The problem with using the LC-circuit as our qubit is that the distance between all the states are now identical and we are no longer able to stimulate individual transitions like we could with the hydrogen atom and therefore not able to use it as a qubit.

For this reason we need to add some non-linearity into our system and that is achieved with the Josephson junction. A Josephson junction is created by sandwiching a weak link between a superconductor, as shown in figure 2.5, this weak link can either be an insulator³, a non superconducting metal⁴ or any restraint on the superconductivity at the weak link⁵.

²You can convert frequencies to temperatures with this formula $h\nu = kT$ where h is plancks constant, ν is frequency, k is boltzmans constant and T Kelvin

³SIS Superconductor-Insulator-Superconductor

⁴SNS Superconductor-Normal-Superconductor

⁵SsS Supercondcutor-Restrained Superconductor-Superconductor

The current through the Josephson junction is given by

$$I = I_0 \sin\left(\frac{2\pi\Phi(t)}{\Phi_0}\right), \quad (2.2)$$

where I_0 is the supercritical current through the junction and $\Phi_0 = \frac{h}{2e}$ is the magnetic flux quantum.

Taking the time derivative of 2.2 we get

$$\dot{I} = I_0 \frac{2\pi}{\Phi_0} \dot{\Phi} \cos\left(\frac{2\pi\Phi}{\Phi_0}\right), \quad (2.3)$$

where $\dot{\Phi} = V(t)$ and self-inductance is given by $L = \frac{V(t)}{\dot{I}}$.

We can then define the Josephson inductance as

$$L_J = \frac{\Phi_0}{2\pi I_0} \frac{1}{\cos(2\pi\Phi/\Phi_0)}. \quad (2.4)$$

The Josephson junction therefore acts as a non-linear inductor. If we remember from (2.1), the energy stored in the inductor is proportional to $\frac{1}{L}$ and if we then replace L with L_J , we will have a cosine potential (figure 2.6) instead of an infinite parabolic potential well. The states would therefore need to readjust and would no longer be uniformly spread. We have now achieved anharmonicity we also saw in the hydrogen atom, exactly what we needed.

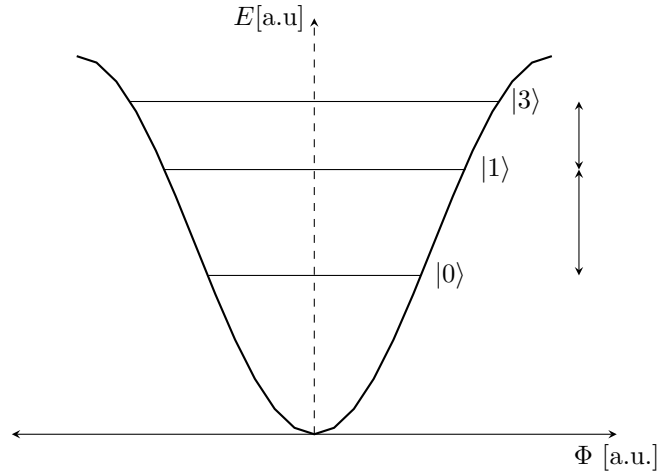


Figure 2.6: JC-circuit energy levels

2.2 The Cooper Pair Box

Going back to the LC-circuit for a moment we can rewrite (2.1) using $\omega_{01} = \frac{1}{\sqrt{LC}}$

$$\hat{H} = \frac{\hat{Q}^2}{2C} + \frac{1}{2}C\omega_{01}\hat{\Phi}^2. \quad (2.5)$$

Recognizing this as the quantum harmonic oscillator, where \hat{P} is equivalent to \hat{Q} and \hat{x} to $\hat{\Phi}$, the first term is describing the kinetic energy and the second term the potential. Therefore the commutation relations we know from the quantum harmonic oscillator, will also apply here. To diagonalize the hamiltonian we introduce \hat{a}^\dagger and \hat{a} , as with the quantum harmonic oscillator. \hat{Q} and $\hat{\Phi}$ can therefore be written as

$$\hat{Q} = i\sqrt{\frac{\hbar}{2z_0}}(\hat{a}^\dagger - \hat{a}), \quad (2.6)$$

$$\hat{\Phi} = \sqrt{\frac{\hbar z_0}{2}}(\hat{a}^\dagger + \hat{a}), \quad (2.7)$$

where $z_0 = \sqrt{\frac{L}{C}}$.

Plugging these expressions back in 2.5 we get

$$\hat{H} = \hbar\omega_{01}(\hat{a}^\dagger\hat{a} + \frac{1}{2}), \quad (2.8)$$

which again is the same result as the quantum harmonic oscillator.

Now we can do the same calculation for the JC-circuit, but we first have to find the energy in the Josephson junction. We already know that the current going through the junction is given by (2.2) and we know that the energy is the time integral of the power therefore

$$\begin{aligned} E &= \int P(t)dt = \int V(t)I(t)dt = \int \left(\frac{d\Phi}{dt}\right)(I_0 \sin\left(\frac{2\pi\Phi}{\Phi_0}\right))dt \\ &= -E_J \cos\left(\frac{2\pi\Phi}{\Phi_0}\right) \end{aligned} \quad (2.9)$$

where $E_J = I_0 \frac{2\pi}{\Phi_0}$. E_J can be seen as the frequency of the cooper pair tunneling, as seen in figure 2.7a

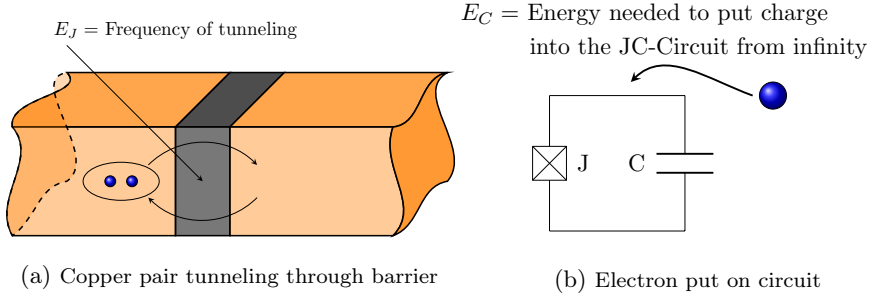


Figure 2.7

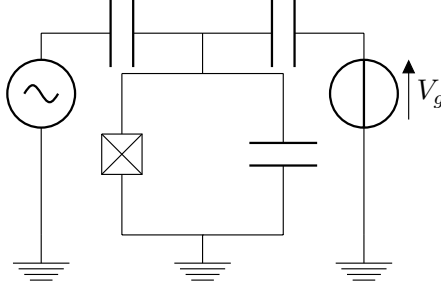


Figure 2.8: Depiction of random charge V_g capacitively coupled to our qubit

The hamiltonian for the JC-circuit will therefore look like

$$\hat{H} = \frac{\hat{Q}^2}{2C} - E_j \cos\left(\frac{2\pi\Phi}{\Phi_0}\right) \quad (2.10)$$

We can clean this up a bit by instead of looking at how much charge is tunneling through our junction to the number of cooper pairs (electron pairs), $\hat{Q} = (2e)\hat{n}$, where \hat{n} is number of cooper pairs.

Furthermore we define $\hat{\phi} = \frac{2\pi\Phi}{\Phi_0}$

$$\begin{aligned} \hat{H} &= \frac{(2e)^2}{2C} \hat{n}^2 - E_j \cos(\hat{\phi}) \\ \hat{H} &= 4E_C \hat{n}^2 - E_j \cos(\hat{\phi}) \end{aligned} \quad (2.11)$$

where in the last step we defined $E_C = \frac{e^2}{2C}$ which is the energy it takes to put a charge into our circuit (figure 2.7b).

Now as our devices are not perfect there will be random charges in the environment bonding to imperfections on our qubit. These extra charges are capacitively coupled to our qubit and acts as a little voltage source (figure 2.8) which will take away charges from our qubit. We would therefore need to take them into account, by subtracting the amount of charges taken away from our circuit, which we will denote n_g (gate charge)

$$\hat{H} = 4E_C(\hat{n} - n_g)^2 - E_j \cos(\hat{\phi}). \quad (2.12)$$

We have now derived the hamiltonian for the cooper pair box. This hamiltonian can be solved analytically in the phase basis⁶.

Now looking at the solutions for the first 3 energy levels in fig 2.9a it is seen that the energy levels depend greatly on the charge noise which would pose a great problem as random fluctuations of charges on our qubit would greatly vary the transition frequencies. As our hamiltonian is only dependent on E_J and E_C , the solution to this problem would be to find an optimal $\frac{E_J}{E_C}$ ratio and look at how this ratio affects our energy levels. Looking at figure 2.9 it can be seen that increasing our $\frac{E_J}{E_C}$ ratio makes us less and less susceptible to charge noise. Increasing our ratio so $\frac{E_J}{E_C} \gg 1$ we see that we no longer are affected by charge noise. This is the domain where the transmon qubit lies.

⁶For more information look at [3]

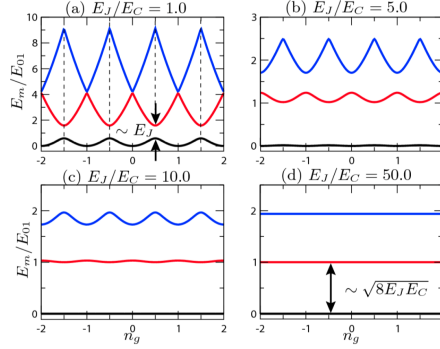


Figure 2.9: Plot from [3] page 3

2.3 Circuit QED

Now that we have made our qubit how do we do measurements on it?

To answer that question will take a short detour to the field of cavity QED, which is the study of photon interaction with particles confined inside a cavity fig 2.10a. Here we have basically tuned down our laser so much that it is now only emitting single photons.

A cavity is basically two mirrors facing each other, only allowing certain frequencies to pass through i.e when the wavelength matches the length of the cavity. The reason why these cavities are used is because the electric field is very weak, since we are now using photons, which couples very weakly with the atoms in question⁷. We would therefore have to increase the strength of the electric field as we want single photons to have a large effect on the atoms. We know that $\vec{E} \propto \frac{1}{\sqrt{\lambda}}$ so we can increase the electric field by lowering the volume of the cavity and thereby achieving a strong coupling with the atom.

A cavity can be made in the superconducting circuit by shaving of two parts of the transmission line fig 2.10b. The electric field confined in this cavity is very large due to the size of the cavity and can therefore couple strongly to the large dipole moment of the qubit island, this is what we call the circuit QED.

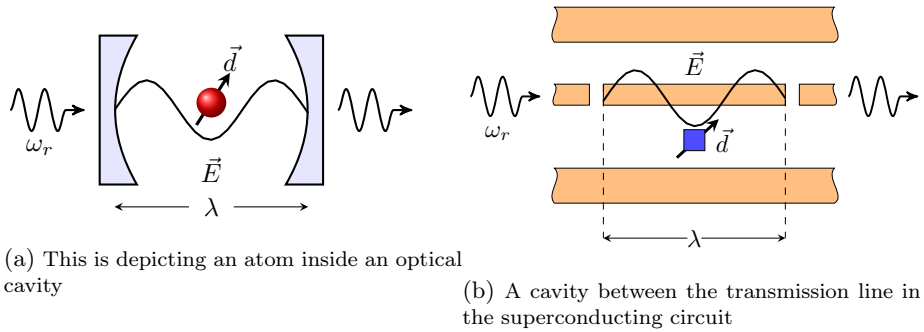


Figure 2.10

⁷If you remember the dipole couples with the E-field as $H = \vec{E} \cdot \vec{d}$

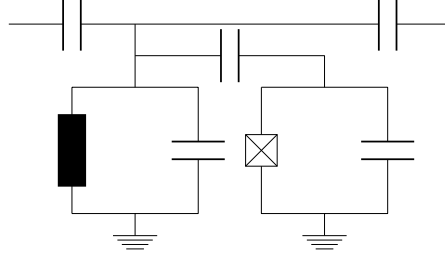


Figure 2.11: Diagram of the full transmon qubit circuit

2.4 Jaynes-Cummings Hamiltonian

As the cavity only allows for certain resonance frequencies i.e only certain energies are allowed, it will behave as the harmonic LC-circuit. Bringing all that we have learned until now together, we can write the hamiltonian as a sum of the LC- and JC-circuit

$$\hat{H} = \underbrace{\frac{\hat{Q}_r^2}{2C_r} + \frac{\hat{\Phi}_r^2}{2L_r}}_{LC\text{-circuit}} + \underbrace{4E_C(\hat{n} - n_g)^2 - E_j \cos(\hat{\phi})}_{JC\text{-circuit}}, \quad (2.13)$$

$$\hat{H} = \hbar\omega_r(\hat{a}^\dagger\hat{a} + \frac{1}{2}) + \frac{\hbar\omega_{01}}{2}\hat{\sigma}_z + \hbar g(\underbrace{\hat{a}^\dagger\hat{\sigma}_-}_{create} + \underbrace{\hat{a}\hat{\sigma}_+}_{annihilate}),^8 \quad (2.14)$$

where ω_r and ω_{01} respectively is the cavity resonance frequency and the qubit frequency, \hat{a}^\dagger and \hat{a} the raising and lowering operators, $\hat{\sigma}_z$ the Pauli spin matrix operator, $g = \vec{E} \cdot \vec{d}$ the qubit-cavity coupling strength and finally σ_+ and $\hat{\sigma}_-$ the spin raising and lowering operators.

The create term in (2.14) can be seen as a photon created by the lowering operator that will then jump to the cavity, where the annihilate term will remove a photon from the cavity by exiting the qubit.

2.5 Dispersive Regime: Qubit Readout

Qubit readout can be done by working in the dispersive regime of the Jaynes-Cummings hamilton i.e $|\Delta| = |\omega_{01} - \omega_r| \gg g$. Here the Jaynes-Cummings hamilton will reduce to

$$H \approx (\omega_r + \chi)\hat{a}^\dagger\hat{a} + \frac{\omega_{01}}{2}\hat{\sigma}_z, \quad (2.15)$$

where $\chi = \frac{g^2}{\Delta}$.

Now we can see that the cavity transmission frequency depends on the state of the qubit, so if the qubit is in the ground state the cavity frequency will be shifted down and if the qubit is in its excited state be pulled up, as can be seen in figure 2.12. This principle is used to determine the state of the qubit in experiments.

⁸A more detailed derivation can be found in [4].

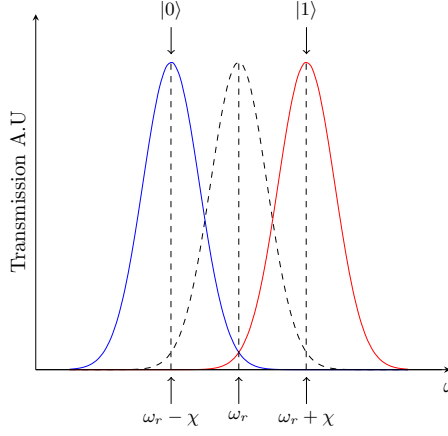


Figure 2.12: Showing the cavity transmission frequency being shifted as the state of the qubit changes

2.6 Gate tunable Qubits

The type of qubits I will be doing measurements on is a gatemon qubit [9]. The gatemon is basically a transmon qubit, where we can tune the qubit frequency with an electrostatic gate. This gate is placed near the josephson junction and by changing the voltage on the gate we can change the critical current going through the junction and thereby changing the qubit frequency.

2.7 Single Qubit Rotations

To visualize various qubit operations we will be using a Bloch sphere (figure 2.13), which is a 3D representation of the Hilbert space spanned by the qubit states. In the Bloch sphere the north pole represents the ground state $|0\rangle$ and south pole the excited state $|1\rangle$. Operations will in this representation be described by rotation of the state vector about the x,y and z axis.

Incorporating the microwave drive, where the drive frequency will be denoted ω_d , in the Jaynes-Cummings hamiltnian, the following relation can be derived [2]

$$\hat{H} = \hbar(\Delta_r + \chi\hat{\sigma}_z)\hat{a}^\dagger\hat{a} + \frac{\hbar\Delta_d}{2}\hat{\sigma}_z + \frac{\hbar}{2}(\Omega_R^x(t)\hat{\sigma}_x + \Omega_R^y(t)\hat{\sigma}_y) \quad (2.16)$$

where $\Delta_r = \omega_r - \omega_d$ is the detuning between cavity resonance and drive frequency, $\Delta_d = \omega_{01} - \omega_d$ is the detuning between the qubit and the drive frequency and lastly Ω_R^x, Ω_R^y which are the two rabi drive amplitudes phase shifted $\frac{\pi}{2}$ from each other.

We see now that by knowing the rabi frequency, Ω_R^x and Ω_R^y allows us to rotate our vector state arbitrary around the x- and y-axis, by choosing the phase and amplitude of the drive signal. For the ALLXY test we will be interested in the π and $\frac{\pi}{2}$ rotations about the x- and y-axis of the bloch sphere.

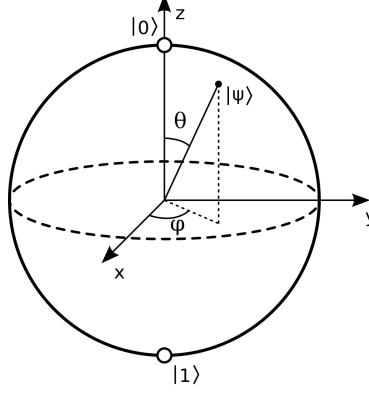


Figure 2.13: The Bloch sphere

2.8 Single Sideband Modulation

In order to do coherent control of the qubit, we will be using single sideband modulation. This allows us to modulate a signal either below or above the carrier frequency. This means that we will be able to do x and y rotations on the qubit without worrying about carrier leakage interfering with the operation.

Single sideband modulation is done with the IQ mixer. In the I port a signal $s(t)$ is multiplied with the carrier frequency, which in our case is ω_d , and in the Q port the Hilbert transform of the signal is multiplied with a $\frac{\pi}{2}$ phase-shifted carrier. The I and Q signals are then added together.

To give an example of this lets say we put $s = \cos(\omega_{rf}t)$ to the I port, a carrier signal of $\cos(\omega_d t)$ to the LO port and $\hat{s} = \sin(\omega_{rf}t)$ to the Q port.

$$\cos(\omega_d t) \cos(\omega_{rf}t) + \sin(\omega_d t) \sin(\omega_{rf}t) = \quad (2.17)$$

$$\frac{1}{2}(\cos(\omega_d + \omega_{rf}) + \cos(\omega_d - \omega_{rf})) + \frac{1}{2}(\cos(\omega_d - \omega_{rf}) - \cos(\omega_d + \omega_{rf})) = \quad (2.18)$$

$$\cos(\omega_d + \omega_{rf}) \quad (2.19)$$

We see that all but the upper sideband cancel out.

In order to find a more general Hilbert transform of the the single sideband modulated signal, we will to have it in the form of

$$s_{ssb} = \text{Re}[m(t)] \cos \omega_d t + \text{Im}[m(t)] \sin \omega_d t \quad (2.20)$$

where $m(t)$ is the pulse envelope (in the ALLXY experiment below we will be using gaussian envelopes). The Hilbert transform of s_{ssb} can be found knowing that $H(\sin(\omega_d t)) = -\cos(\omega_d t)$ and $H(\cos(\omega_d t)) = \sin(\omega_d t)$. Furthermore knowing that $s(t)$ is much slower than $\cos \omega_d t$, the Bedrosian's theorem can be applied. The Hilbert transform of s_{ssb} will therefore be

$$\hat{s}_{ssb} = \text{Re}[m(t)] \sin \omega_d t - \text{Im}[m(t)] \cos \omega_d t. \quad (2.21)$$

From this we would calculate the signals from the I and Q, like the above example and end up with

$$I_{\pm} = \text{Re}[m(t)] \cos \omega_d t \pm \text{Im}[m(t)] \sin \omega_d t \quad (2.22)$$

$$Q_{\pm} = \pm \text{Re}[m(t)] \sin \omega_d t + \text{Im}[m(t)] \cos \omega_d t \quad (2.23)$$

Where I_+ and Q_+ is for selecting the upper sideband and I_- and Q_- is for the lower sideband, where we have multiplied \hat{s}_{ssb} with -1.

Chapter 3

Setup

3.1 Proposed Setup for Qubit Control

In order for this setup to be scalable and easily upgradeable, the system is designed to be modular. This system consists of 3 different module types, a module which contains a 6 way splitter to split the carrier frequency into multiple mixer assemblies, a mixer module which consists of an amplifier, a mixer and a directional coupler and finally 10 way switch to be able to monitor the signals from the mixer modules. A full schematic can be seen in figure 3.1 .

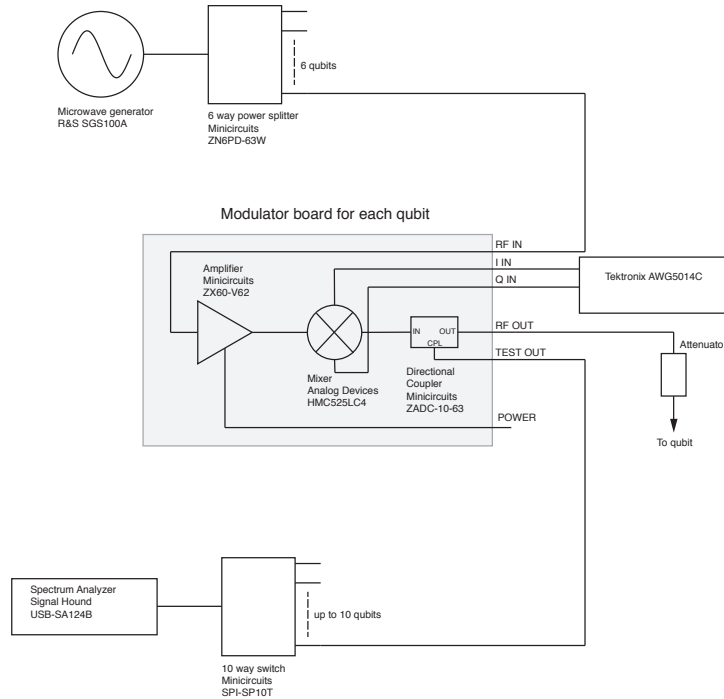


Figure 3.1: Schematic of new scalable mixer setup, by Karl David Peterson

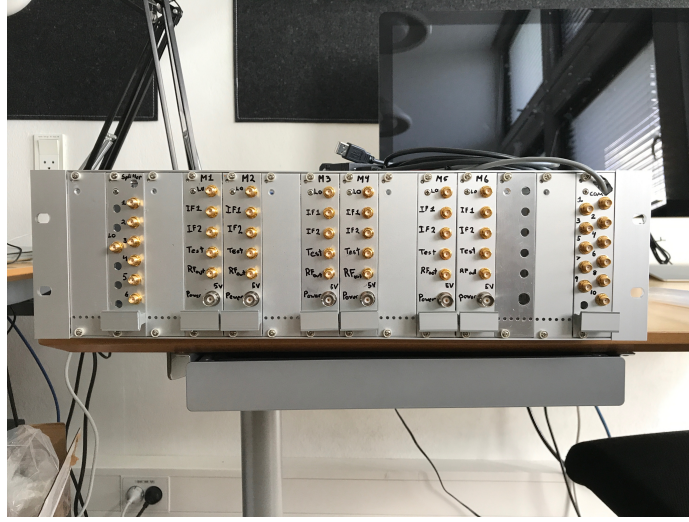


Figure 3.2: Fully assembled rack setup

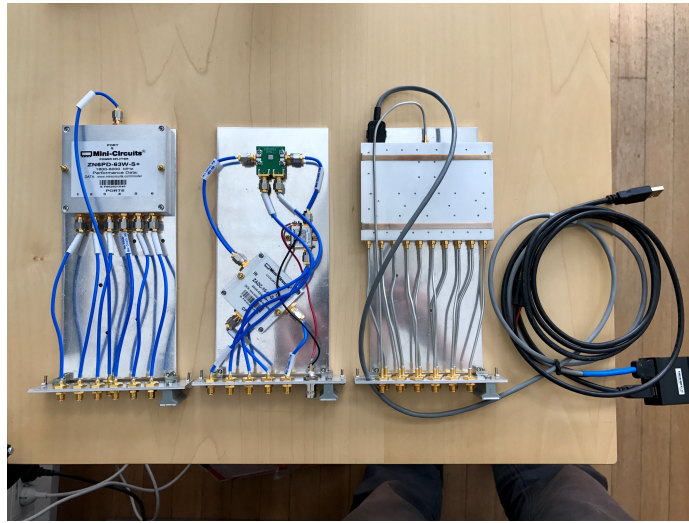


Figure 3.3: From left - the 6 way splitter module, mixer module and 10 way switch

3.2 IQ Mixer Optimization Algorithm

As the mixers we are working with are not perfect calibration is needed to maximize carrier and sideband suppression. The 3 variables that are needed to be taken into account for is the I and Q voltage offsets, amplitude and phase difference. The optimization algorithm works by calibrating the voltage offsets first. Here it starts by doing a large sweep and looking at the value at the carrier frequency, then choosing the lowest value it does a more fine sweep around that value and again choosing the lowest value, finally doing one last very fine sweep around this value and choosing this value. This is done on both the I and Q voltage offsets. Now it moves on to the amplitude of the I signal, doing the same routine of doing a coarse, medium and fine sweep looking for the lowest value of the lower sideband. Lastly it will run through multiple phase differences between the I and Q ports and find the lowest value of the lower sideband.

Doing this routine multiple times we will eventually find the optimized value, see figure (3.4 and 3.5) for before and after optimization. The code can be found in the appendix.

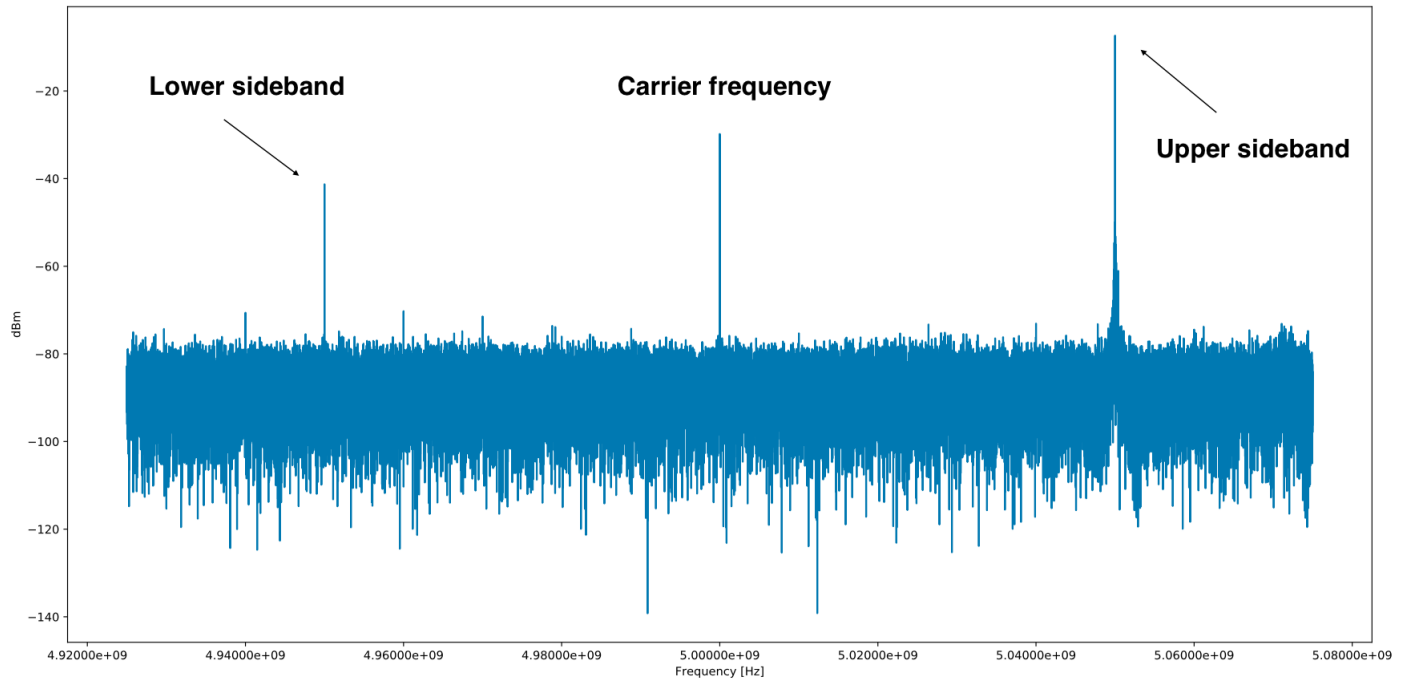


Figure 3.4: Before calibration, using a carrier frequency of 5 Ghz and IF frequency of 50 mhz

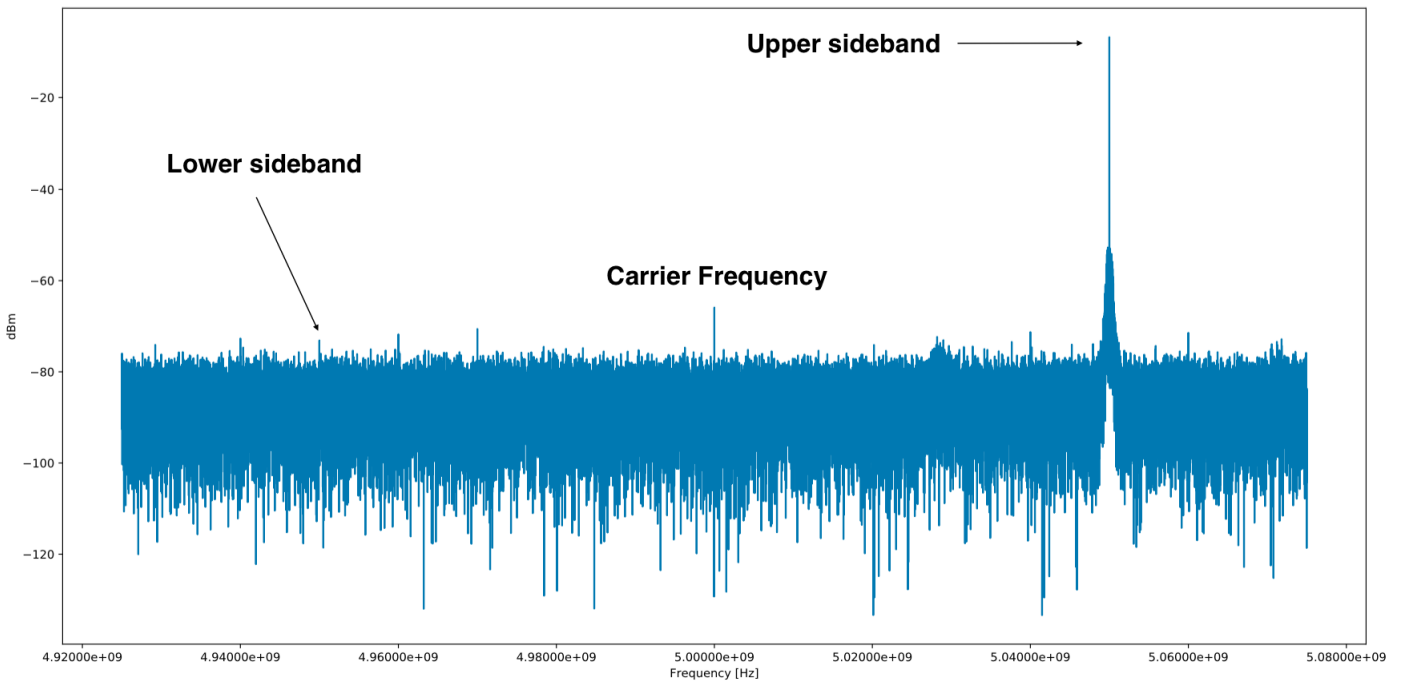


Figure 3.5: After calibration

Chapter 4

Experiment

The following measurements were taken with the help of Lucas Casparis and Natalie Pearson, using the new mixer setup, on a 6 qubit device created by Lucas. A detailed setup description can be found in [8].

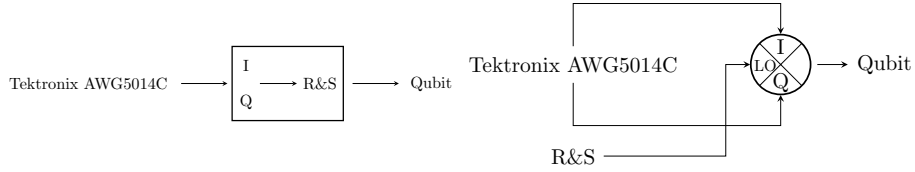


Figure 4.1: On the left we see a depiction of the old setup and on the right where we've replaced the internal mixer of the R&S with the new external mixers

4.1 Spectroscopy

In a spectroscopy we sweep the frequencies at increasing gate voltages and measure the magnitude of the output signal, when the magnitude dips, we have transmission through the cavity, which means that we've hit the qubit frequency. Comparing the results from the old (4.3) and the new setup (4.2) we see that the performance is on par with the internal mixers of the R&S.

4.2 Rabi

Seeing the spectroscopy was successful we moved on doing rabi measurements. During these measurements we noticed that at the qubit frequency there was a significant shift figure (4.5). We believe this was due to poor carrier suppression. We circumvented this problem with using mixer sidebands instead of the carrier and observed much better performance.

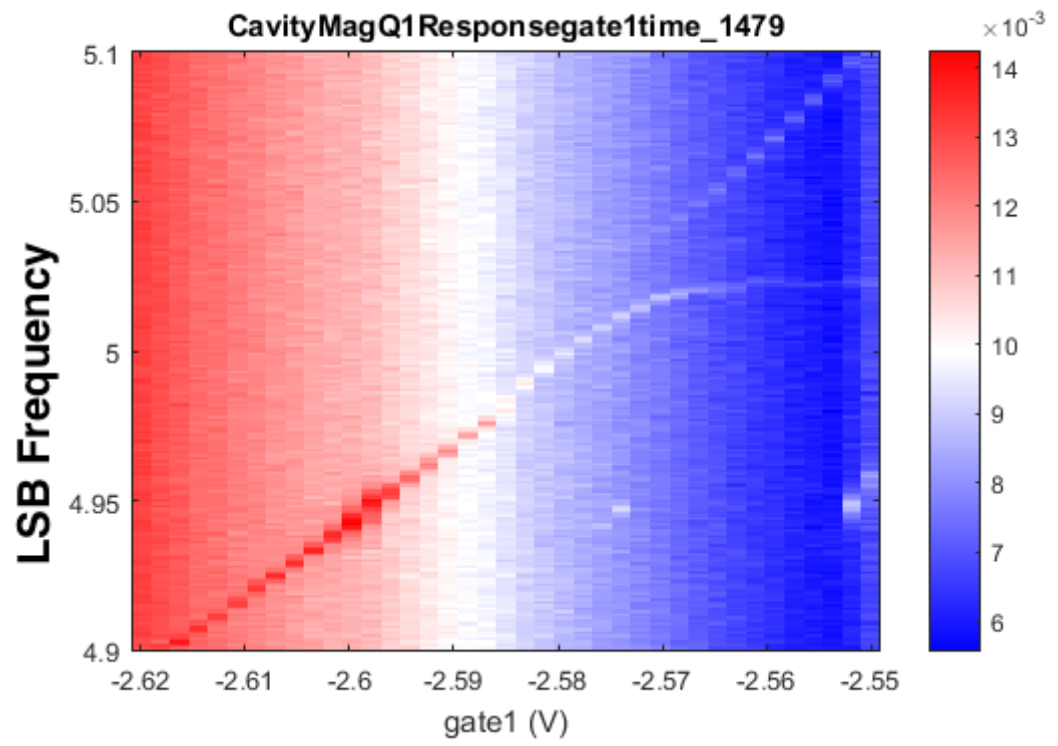


Figure 4.2: Spectroscopy using new mixer setup

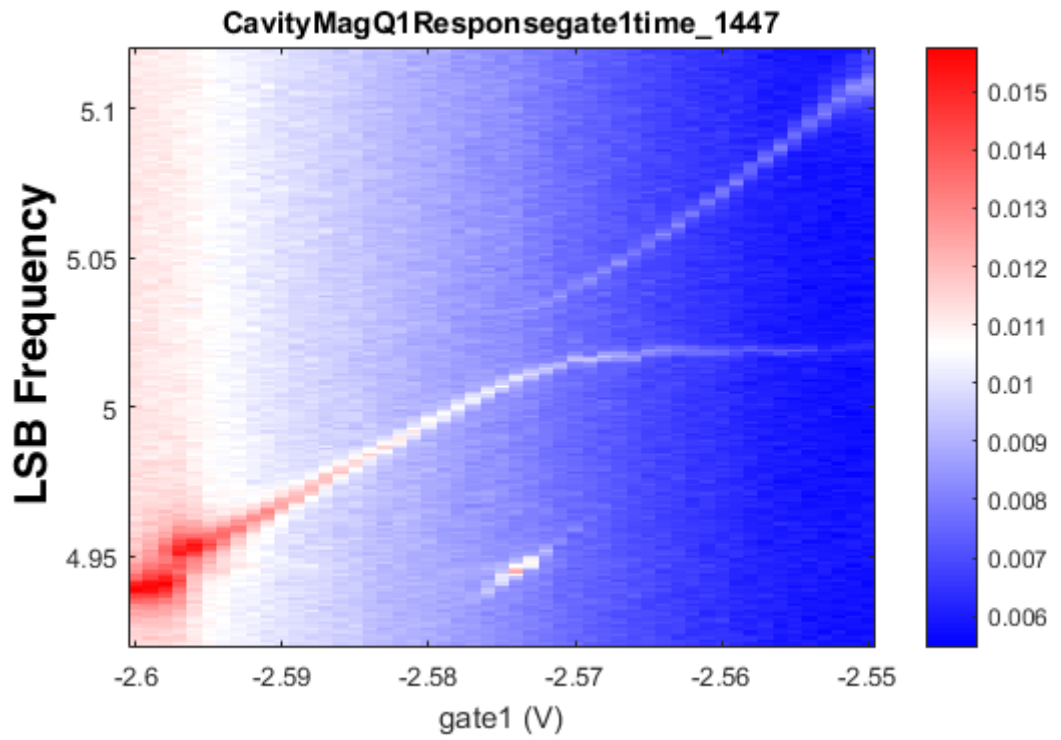


Figure 4.3: Spectroscopy using old mixer setup

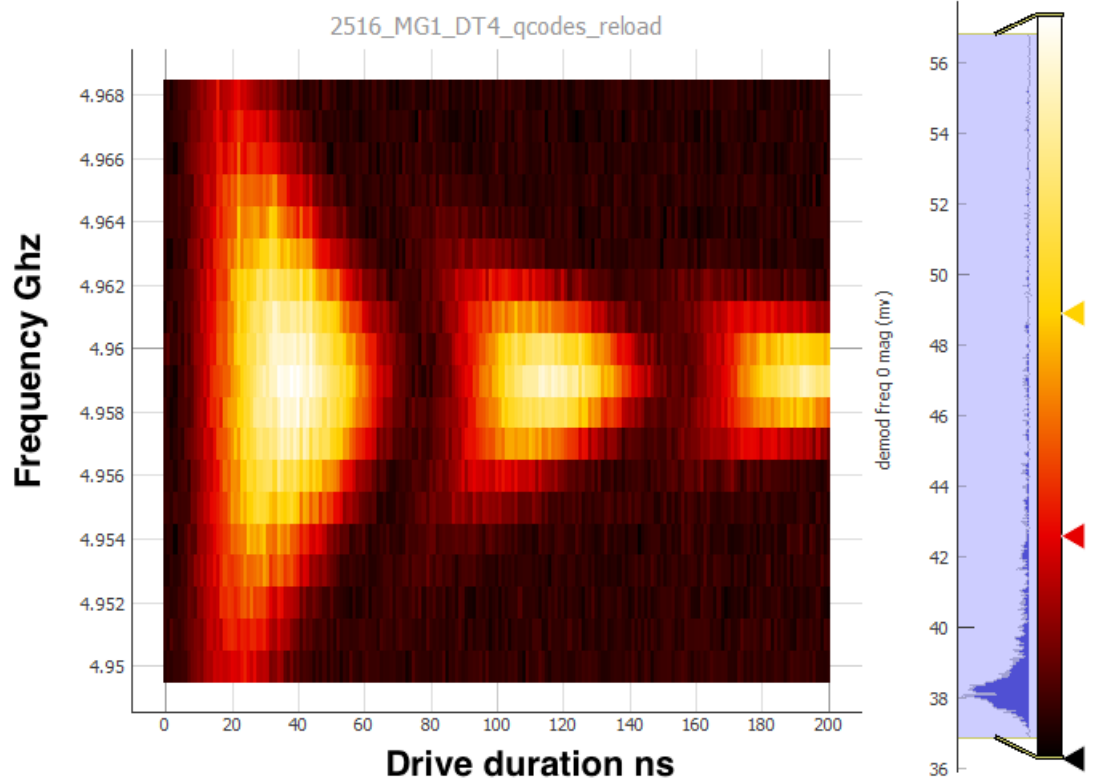


Figure 4.4: Rabi measurements using mixer sidebands

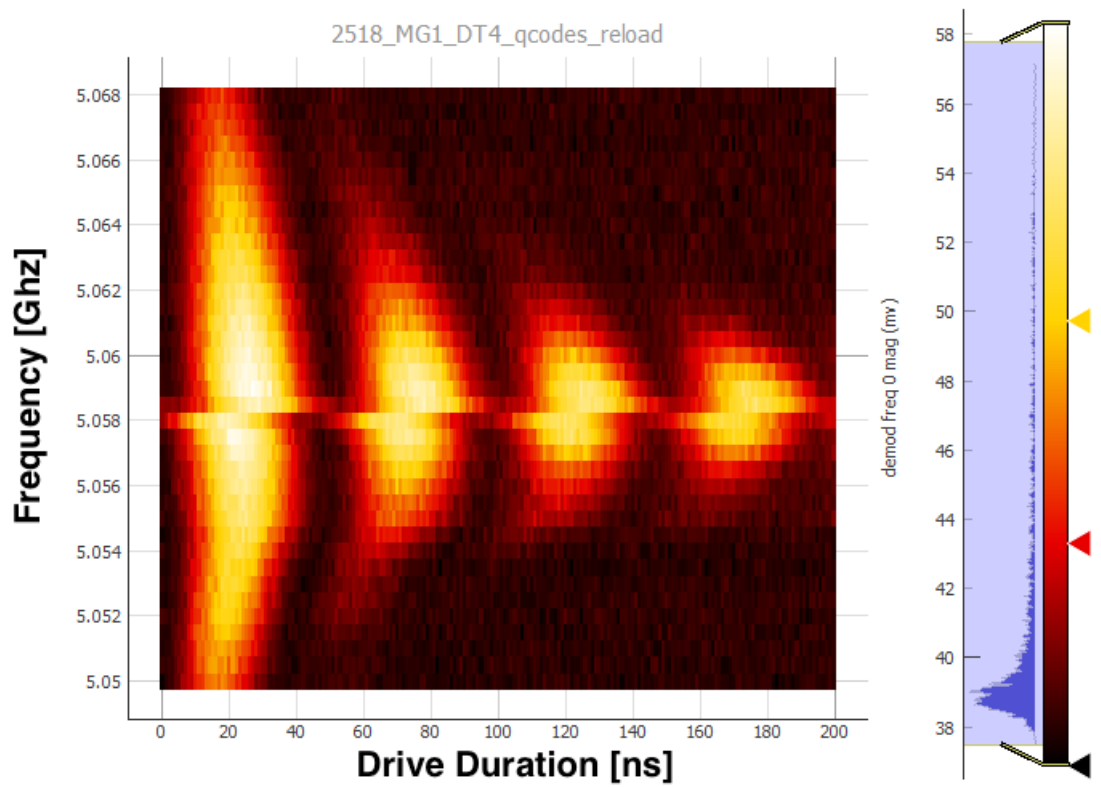


Figure 4.5: Rabi measurements using mixer carrier frequency

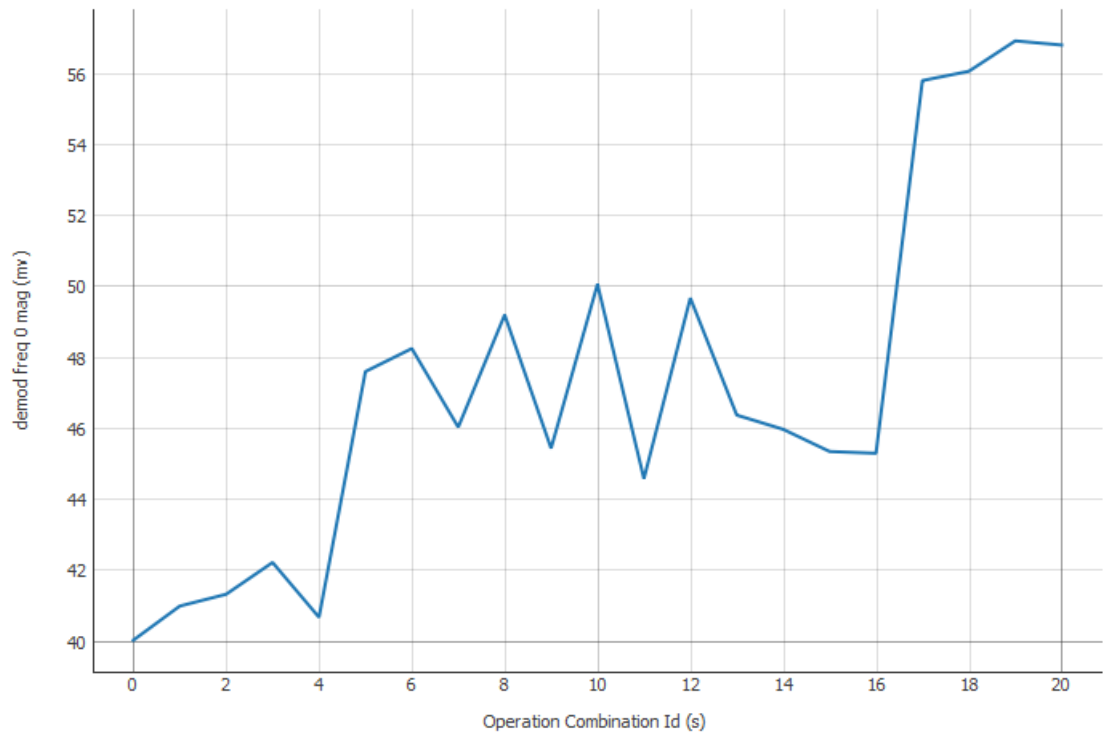


Figure 4.6: ALLXY measurement using old setup the order of the operations can be found in table 5.1 in [2]

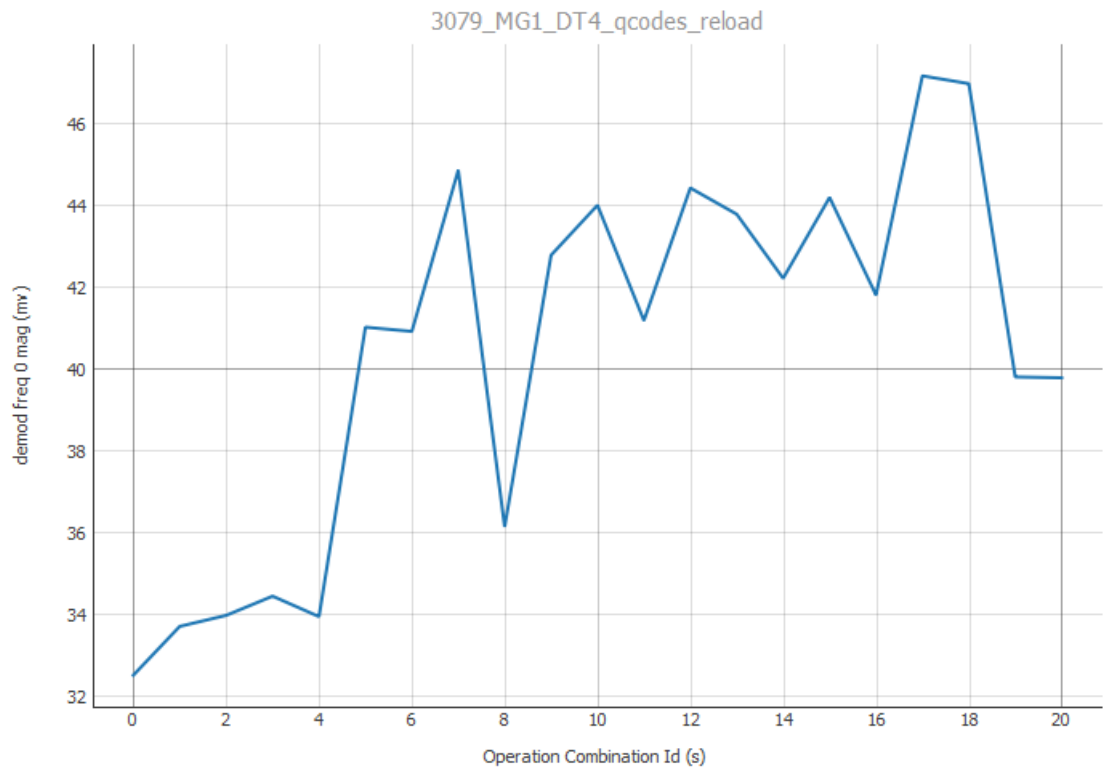


Figure 4.7: ALLXY measurement using new setup

4.3 Gate tuneup / ALLXY

Finally we tried to tune the qubits doing an ALLXY test. In an ALLXY test all combinations of one and two $\pi, \frac{\pi}{2}$ rotations, about both the x- and y-axis are performed. By looking at the measurements deviation from the expected values, different error types can be identified and accounted for. In this experiment we used single sideband modulated gaussian pulses. All the combinations of operations and more detailed description can be found in [2] where the order of our operations can be found in table 5.1.

We first did the ALLXY measurements using the old setup to get a baseline on how good we could tune the qubit. Looking at a line cut in figure 4.6 it was pretty good, as the measurements fit the expected values when doing the various operation combinations. Now changing to the new setup we did not see same success as can be seen in figure 4.7.

We can see from the line cut that doing a π rotation (operation 19 and 20) is not getting the same result as doing two $\frac{\pi}{2}$ rotations (operation 17 and 18). This indicates that our x and y gates are not perfectly $\frac{\pi}{2}$ shifted from each other and would therefore cause under rotation in the Bloch sphere. This could be addressed by fine tuning of phase difference between the sideband modulated pulses, unfortunately time ran out and could therefore be a calibration to try in future experiments.

4.4 Conclusion

From these results we can conclude that this new setup looks very promising, though further tuneup would be needed in order to match the performance of the R&S.

This proof of concept opens up new possibilities for future multi qubit experiments, as it is now shown that much cheaper hardware allows for similar performance as current setups. To further improve the scalability and reduce costs of this setup, one could consider creating a integrated circuit with mixers, amplifiers and decouplers build in.

Bibliography

- [1] Gordon E. Moore *Cramming more components onto integrated circuits* Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.
- [2] Matthew David Reed *Entanglement and Quantum Error Correction with Superconducting Qubits*
- [3] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, R. J. Schoelkopf *Charge insensitive qubit design derived from the Cooper pair box* Phys. Rev. A 76, 042319 (2007)
- [4] Alexandre Blais, Ren-Shou Huang, Andreas Wallraff, S. M. Girvin, and R. J. Schoelkopf *Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation* PHYSICAL REVIEW A 69, 062320 (2004)
- [5] T.W. Larsen, K.D. Peterson, F. Kuemmeth, T.S Jespersen, P. Krogstrup, J. Nygaard and C.M. Marcus *Semiconductor-Nanowire-Based Superconducting Qubit* Phys. Rev. Lett. 115, 127001 – Published 14 September 2015
- [6] Peter W. Shor *Algorithms for Quantum Computation: Discrete Logarithms and Factoring* Proc. 35nd Annual Symposium on Foundations of Computer Science (Shafi Goldwasser, ed.), IEEE Computer Society Press (1994), 124-134.
- [7] Richard P. Feynman *Simulating Physics with Computers* International Journal of Theoretical Physics, VoL 21, Nos.6/7, 1982
- [8] Anders Kringhøj *Readout and Control of Semiconductor- Nanowire-Based Superconducting Qubits*
- [9] Blake Robert Johnson *Controlling photons in superconducting electrical circuits*

Appendix A

Mixer Optimization Algorithm Code

AWG Automation initialiser

June 11, 2017

```
In [1]: '''Initialize the AWG'''
```

```
import os
import time
import logging
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
import qcodes.instrument_drivers.tektronix.AWG5014 as awg # <--- The instrument driver
from qcodes.instrument_drivers.tektronix.AWGFileParser import parse_awg_file # <--- A h
```

```
awg1 = awg.Tektronix_AWG5014('AWG1', 'TCPIP0::172.20.3.14::inst0::INSTR', timeout=40)
```

AWG clock freq not set to 1GHz

Connected to: 1.20000000E+009 None (serial:None, firmware:None) in 0.08s

```
In [2]: #Set the amplitude of used channels to not destroy the IQ mixer
```

```
awg1.clock_freq(1e9)
```

```
current_amp = 0.2
```

```
awg1.ch1_amp.set(current_amp)
awg1.ch2_amp.set(current_amp)
```

```
awg1.ch1_offset.set(0)
```

```
awg1.ch2_offset.set(0)
```

```
In [3]: # noofsegelems runs all the different waveforms after each other in the waveformgenerator
noofsegelems = 50
```

```

noofpoints = 2001
band_frequency = 50e6

waveforms = [[], []] # one list for each channel
m1s = [[], []]
m2s = [[], []]
for ii in range(noofselelems):
    # waveform and markers for channel 1
    #Remember that i put 2 times in to get right mode

    wf0 = np.sin(2*np.pi*np.linspace(0, 2000 , noofpoints) * (band_frequency * 1e-9) - i
    wf0 = np.delete(wf0, int(noofpoints-1))
    waveforms[0].append(wf0)

    # waveforms[0].append(np.sin(2*np.pi*(ii+1)*np.linspace(0, 1 , noofpoints)))
    #waveforms[0] = np.delete(waveforms[0], int(noofpoints-1))

    m1 = np.zeros(noofpoints-1)
    m1[:int((noofpoints-1)/(ii+1))] = 1
    m1s[0].append(m1)
    m2 = np.zeros(noofpoints-1)
    m2s[0].append(m2)

    # waveform and markers for channel two
    #Remember that i put 2 times in to get right mode
    wf = np.sin(2*np.pi*np.linspace(0, 2000, noofpoints) * band_frequency * 1e-9)
    wf = np.delete(wf, int(noofpoints-1))
    waveforms[1].append(wf)

    m1 = np.zeros(noofpoints-1)
    m1[:int(noofpoints-1/(ii+1))] = 1
    m1s[1].append(m1)
    m2 = np.zeros(noofpoints-1)
    m2s[1].append(m2)

In [6]: # number of repetitions
nreps = [0 for ii in range(noofselelems)]

trig_waits = [0]*noofselelems
# Goto state
goto_states = [0]*noofselelems
# Event jump
jump_tos = [0]*noofselelems

awg1.make_send_and_load_awg_file(waveforms, m1s, m2s,
                                  nreps, trig_waits,

```

```

goto_states, jump_tos, channels=[1, 2])

In [5]: awg1.clear_message_queue()

In [7]: awg1.all_channels_on()
awg1.run()

Out[7]: 'Running'

In [8]: import qcodes.instrument_drivers.signal_hound.USB_SA124B as sh
import matplotlib.pyplot as plt

''' Here you specify the path to your sa_api.dll file
    which is located in the signal hound install folder. '''

sa_api_path = 'C:\Program Files\Signal Hound\Spike\sa_api.dll' # Specify here between th

sh1 = sh.SignalHound_USB_SA124B('sh1', sa_api_path)

INFO:root:qcodes.instrument_drivers.signal_hound.USB_SA124B : Initializing instrument SignalHound
INFO:Main.DeviceInt:Opening Device
INFO:Main.DeviceInt:Querying device for model information
INFO:Main.DeviceInt:Querying device for model information

Initialized SignalHound in 6.83s

In [242]: ''' Specify the frequency and span to sweep accros in hz '''

#Specify frequency
frequency = 4.85e9
sh1.frequency(frequency)

#Specify Span
span = 100e6
#span = 10e3
sh1.span(span)

#This updates the frequency and span parameters and prepares the device for measurement
sh1.prepare_for_measurement()

'''Determine whether or not you want to plot your swept span'''

plot_or_not = 'yes' #yes or no between the apostrophe

```

```

if plot_or_not == 'yes':
    spectrum = sh1.sweep() #Sweeps the desired range and returns like this, np.array([

    fig = plt.figure(figsize=(20,10))
    ax = fig.add_subplot(111)
    plt.plot(spectrum[0], spectrum[1]) #Plots dBm vs frequency
    ax.xaxis.set_major_formatter(plt.FormatStrFormatter('%.5e'))
    plt.ylabel('dBm')
    plt.xlabel('Frequency [Hz]')
    plt.show()

    '''If you only want the power at the specified frequency'''

    power_at_freq = sh1.get_power_at_freq()

    print('Power at %.2e [Hz] is %.2f [dBm]' % (sh1.get('frequency'), power_at_freq))

```

```

INFO:Main.DeviceInt:Setting device CenterSpan configuration.
INFO:Main.DeviceInt:Setting device reference level configuration.
INFO:Main.DeviceInt:Setting device Sweeping configuration.

```

Warning: saBandwidthClamped

Power at 4.85e+09 [Hz] is -53.07 [dBm]

```

In [235]: import time
          start_time = time.clock()

          #Initialize the spectrum analyser
          sh1.frequency(frequency)
          sh1.span(10e3)
          sh1.prepare_for_measurement()

          #Initialize sweep for channel1
          a = []
          ch1_offset = 0.03

          #Initialize sweep for channel2
          b = []
          ch2_offset = 0.03

```

```

t = 0
#####
while t <= 15:
    awg1.ch1_offset.set(ch1_offset)
    time.sleep(0.4)
    power = sh1.get_power_at_freq()

    a.append([ch1_offset, power])
    ch1_offset += -0.004

    t += 1

#Finds the minimum in a and returns the offset value for this minimum
g = round(min((i for i in a), key=lambda i: i[1])[0],3)

a = []
ch1_offset = g + 0.004

t = 0

while t <= 8:
    awg1.ch1_offset.set(ch1_offset)
    time.sleep(0.4)
    power = sh1.get_power_at_freq()

    a.append([ch1_offset, power])
    ch1_offset += -0.001

    t += 1

g = round(min((i for i in a), key=lambda i: i[1])[0],3)
awg1.ch1_offset.set(g)

#####
t = 0

while t <= 15:
    awg1.ch2_offset.set(ch2_offset)
    time.sleep(0.4)
    power = sh1.get_power_at_freq()
    b.append([ch2_offset, power])

    ch2_offset += -0.004

    t += 1

```

```
#Finds the minimum in a and returns the offset value for this minimum
h = round(min((i for i in b), key=lambda i: i[1])[0],3)
```

```
b = []
ch2_offset = h + 0.004
```

```
t = 0
```

```
while t <= 8:
    awg1.ch2_offset.set(ch2_offset)
    time.sleep(0.4)
    power = sh1.get_power_at_freq()
    b.append([ch2_offset, power])
```

```
ch2_offset += -0.001
```

```
t += 1
```

```
h = round(min((i for i in b), key=lambda i: i[1])[0],3)
awg1.ch2_offset.set(h)
```

```
#####
```

```
print((time.clock() - start_time), "seconds")
```

```
INFO:Main.DeviceInt:Setting device CenterSpan configuration.
INFO:Main.DeviceInt:Setting device reference level configuration.
INFO:Main.DeviceInt:Setting device Sweeping configuration.
INFO:Main.DeviceInt:Call to initiate succeeded.
```

```
30.333558417165477 seconds
```

```
In [238]: import time
          start_time = time.clock()

#Initialize the spectrum analyser
sh1.frequency(frequency - band_frequency)
sh1.span(10e3)
sh1.prepare_for_measurement()
```



```

time.sleep(1)

#Initialize sweep for channel1
a = []
ch1_amplitude = current_amp + 0.100

t = 0
#####

while t <= 5:
    awg1.ch1_amp.set(ch1_amplitude)
    time.sleep(0.5)
    power = sh1.get_power_at_freq()
    a.append([ch1_amplitude, power])

    ch1_amplitude += -0.04

    t += 1

#Finds the minimum in a and returns the offset value for this minimum
g = min((i for i in a), key=lambda i: i[1])[0]
awg1.ch1_amp.set(g + 0.04)

##### new part
a = []
ch1_amplitude = g + 0.04

t = 0

while t <= 5:
    awg1.ch1_amp.set(ch1_amplitude)
    time.sleep(0.5)
    power = sh1.get_power_at_freq()
    a.append([ch1_amplitude, power])

    ch1_amplitude += -0.016

    t += 1

g = min((i for i in a), key=lambda i: i[1])[0]
awg1.ch1_amp.set(g + 0.008)

#####

a = []
ch1_amplitude = g + 0.008

```

```

t = 0

while t <= 16:
    awg1.ch1_amp.set(ch1_amplitude)
    time.sleep(0.5)
    power = sh1.get_power_at_freq()
    a.append([ch1_amplitude, power])

    ch1_amplitude += -0.001

    t += 1

g = min((i for i in a), key=lambda i: i[1])[0]
awg1.ch1_amp.set(g)

print((time.clock() - start_time), "seconds")

```

```

INFO:Main.DeviceInt:Setting device CenterSpan configuration.
INFO:Main.DeviceInt:Setting device reference level configuration.
INFO:Main.DeviceInt:Setting device Sweeping configuration.
INFO:Main.DeviceInt:Call to initiate succeeded.

```

21.32087232743106 seconds

```

In [237]: import time
          start_time = time.clock()

          #Initialize the spectrum analyser
          sh1.frequency(frequency - band_frequency)
          sh1.span(10e3)
          sh1.prepare_for_measurement()

          #Initialize sweep for channel1
          a = []
          seq_pos = 1

          t = 0

          while t < noofsequelems:
              awg1.sequence_pos.set(seq_pos)
              time.sleep(0.5)
              power = sh1.get_power_at_freq()

              a.append([seq_pos, power])
              seq_pos += 1

```

```

t += 1

#Finds the minimum in a and returns the offset value for this minimum
g = min((i for i in a), key=lambda i: i[1])[0]
awg1.sequence_pos.set(g)

print((time.clock() - start_time), "seconds")

INFO:Main.DeviceInt:Setting device CenterSpan configuration.
INFO:Main.DeviceInt:Setting device reference level configuration.
INFO:Main.DeviceInt:Setting device Sweeping configuration.
INFO:Main.DeviceInt:Call to initiate succeeded.

```

35.099577999888425 seconds

```

In [240]: np.savetxt('C:\\Users\\Triton 5 DAQ\\Desktop\\daniel\\5ghz_50mhz_RS25dbm_HM1000a.txt',
In [7]: import numpy as np
In [128]: power
Out[128]: -45.067314147949219
In [243]: low = power_at_freq
In [241]: high = power_at_freq
In [244]: delta = high - low
In [245]: delta
Out[245]: 53.076743410667405
In [153]: low
Out[153]: -54.266780853271484
In [ ]:

```