ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Building Microcontroller-based Smart Locks for Quantum Optics Experiments

Master Thesis

Philip Rhyner

29 September, 2019

Advisors: Philip Zupancic, Tobias Donner and Prof. Tilman Esslinger

Department of Physics , ETH Zürich

Abstract

We develop and build a digital feedback device for active stabilization of experimental parameters. The digitized inputs are handled with a microcontroller which can be programmed to arbitrarily process the signal in real-time thus giving full control over the feedback parameters and allowing for the programming of smart locks. The device has a 16-bit input and 20-bit output resolution and can achieve bandwidth of 80 kHz for closed-loop feedback operation. This makes our device a fast, accurate and versatile solution for amplitude and frequency stabilization.

Contents

Chapter 1

Introduction

The "IMPACT" lab in Tilman Esslinger's group at ETH Zurich studies long-range interacting quantum systems. To this end, a gas of ^{87}Rb atoms is cooled below the Bose-Einstein condensate (BEC) transition temperature and placed at the mode crossing of two optical cavities [1, 2]. The light-matter interaction between atoms and cavity modes leads to an effective global range atom-atom interaction [3]. The coupling to two cavity modes allows investigating the interplay of order parameters [4] and the realization of new exotic phases of matter such as supersolids [5, 6].

These kinds of experiments require complex optical set-ups that are aligned with micrometre precision, for example when a laser beam is coupled into an optical fibre. Small changes in the temperature can change this alignment and lead to fluctuating coupling efficiencies. Optical cavities play an essential role for increasing the power of a laser-beam, increasing atom-light interactions, or narrowing laser line widths. These devices however are particularly sensitive to environmental influences such as mechanical vibrations, sound waves or thermal expansion because their resonances can be limited to a picometre of cavity length.

Effort can be put into reducing environmental influences, but they cannot be suppressed sufficiently to perform stable and reproducible results. Instead, active stabilization of all important experimental parameters is necessary. For example the laser power can be controlled via an acousto-optic modulator (AOM) or the length of an optical cavity changed by piezoelectric actuators which are attached to the cavity mirrors. These parameters can then be adjusted with a closed-loop control device that compares their measured values with a set value and gives feedback to minimize their difference.

A multitude of different control devices are commercially available or have been developed in-house. The "IMPACT" lab largely uses Newport's LB1005-S for laser power stabilization and the in-house developed QO-EL-0021 for

frequency locks, both of which are analog devices. These perform fine in most situations but lack flexibility and features in more challenging cases. In particular, they do not allow to tune proportional, integral and differential gains separately, an integrator-hold function is implemented but flawed, and the QO-EL-0021 has a low bandwidth. Furthermore, these controllers are linear devices acting on non-linear systems. These non-linearities stem for example from the non-linear response of AOM's or mechanical resonances in cavity set-ups, that lead to a frequency-dependent phase and amplitude response. Lastly, the devices do not offer a re-lock feature that can identify the correct lock point again after it was lost due to acoustic or electronic noise.

The idea of this project is to build a digital feedback device as an all-round solution. The measured and reference value for the feedback loop are digitized and fed into a microcontroller which can be programmed to arbitrarily process the signal in real-time, yielding full control over feedback parameters and the possibility to linearise the system response. This digitalisation also allows for the programming of a smarter device which for instance recognizes by itself when a cavity is not locked any more and can find back the correct lock point on its own. The project consists in the development of hardware and software for this smart controller.

When I started working on this project the device was already under development by Philip Zupancic, who had built a prototype and tested the basic programming of the microcontroller. In a first step we developed and implemented algorithms for automatic optimization of feedback parameters, linearisation of system responses, and automatic cavity locking. This work revealed several weaknesses of the initial hardware design which we then improved upon by building and testing a second version.

In chapter 2 the basic idea and design of both the previous and current version of the device will be explained. Then in chapters 3 and 4 the intensity and frequency stabilisation are explained, which are the two main applications of the Lock-box. Some measurements and results from these tasks will be presented to see how well the device performs. In chapter 5 the development of the new hardware and its design is explained. And lastly in chapter 6 a brief conclusion and an outlook for the project is given.

Chapter 2

Theory of the Lock-box

In this chapter I will briefly explain the idea behind the current design and the theory of operation of the Lock-box. This discussion focuses mostly on the connection and interaction between each of the major hardware components. The basic structure of the main components in the Lock-box can be seen in figure 2.1.

If we want to connect our Lock-box to a quantum optical experiment we first need an input from the other lab devices. For most applications two such inputs are needed for processing the state of the system. For example in a frequency lock the error-signal and transmission signal are needed or in an intensity stabilisation we need a set-point and intensity signal to compare to each other. The Lock-box needs also an output so it can give a feedback to the lab equipment and influence the experiment. Usually one such output is enough but for some situations a second one can be advantageous. For example if we want to regulate an optical cavity we can use the two outputs to control both the piezo-electric actuators of the cavity mirrors to increase our control.

The input is processed by an analog digital converter (ADC) which processes the analog input signal from the experiment and converts it into a digital number. We need two ADCs to convert both of the input signals or one ADC with two channels which is what we chose for our device. This digitisation is needed because we want to process our input in real time with a microcontroller. The microcontroller is an integrated circuit (IC) which acts like a small computer. The microcontroller has its own central processing unit (CPU), random access memory (RAM) and flash memory. The microcontroller can perform this real time calculation and produce the appropriate feedback. The feedback is then output by a digital analog converter (DAC) which performs the opposite function of the ADC. The two DACs of the device receive a digital number and convert it into an analog signal which can be connected back to the experiment via a BNC cable.

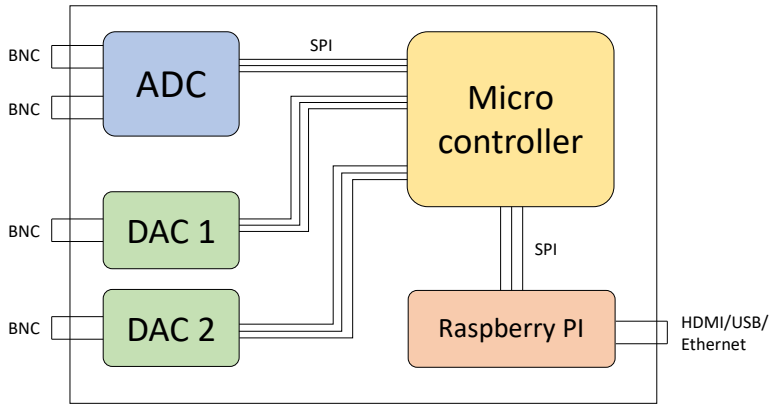


Figure 2.1: Schematic overview of the components of the Lock-box. The device has two analog BNC inputs which are connected to an ADC and two analog BNC outputs from two DACs. These peripherals are connected via SPI to a microcontroller. The microcontroller is connected to a Raspberry Pi which has access to more advance peripherals such as HDMI, USB or Ethernet to more conveniently operate the device.

If we want to give user input to the micro controller to adjust its parameters or to program its behaviour we need an interconnection to a device that's more suitable for human input. In our Lock-box this function is being performed by the Raspberry Pi. The Raspberry Pi is a system on a chip (SoC) which is very similar to a microcontroller but has also access to more advanced peripherals like a graphics processing unit (GPU) , universal serial bus (USB) or Ethernet connections. The Raspberry Pi runs with a Linux operating system (OS) on which we can run a graphical user interface (GUI) to allow a user to interact with the Lock-box. We chose to use the Raspberry Pi on the board in addition to the microcontroller for a few different reasons. For once the Raspberry Pi is easier to interact with because of the Ethernet connection which allows the Raspberry Pi to be controlled by a remote desktop application. The Raspberry Pi can also be directly connected to a screen and mouse/keyboard and accessed that way. And additional reason for the extra computational unit is that this takes away some burden from the microcontroller and allows it to focus on the real time calculations. And lastly the Raspberry Pi has access to higher level programming languages like Python which have better tools for data analysis. Python is also more commonly known and easier to use than programming directly onto the microcontroller. This makes the maintenance of the device easier and more

accessible.

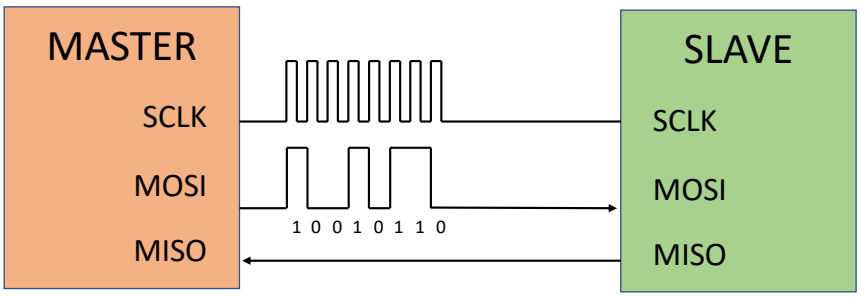


Figure 2.2: Basic overview of a SPI connection. SCLK: Serial Clock, MOSI: Master Output Slave Input, MISO: Master Input Slave Output. The bit-string is clocked out from the master and then read at the receiver at rising clock edges.

All the previously mentioned components such as the ADC, DACs, the microcontroller and the Raspberry Pi are connected by a serial peripheral interface (SPI) as shown in figure 2.1. The SPI connection allows the transfer of data between two of the components. The SPI is a full duplex communication which means that data can be sent in both directions between the two parties, usually called slave and master like in figure 2.2. The master can initiate the SPI communication and the slave can only send data when it is asked to do so. Each SPI connection consist of three wires: a clock signal (CLK), master in slave out (MISO) and master out slave in (MOSI). To initiate the communication the master starts to set the clock signal to high. Both the slave and master can then start sending their data by setting their channel (MOSI or MISO) to high or low to indicate a digital one or zero. The clock signal is then set low again and the process is then repeated until all of the data is exchanged between the two parties. The microcontroller is the master in the communication to the ADC and DACs. This is because the microcontroller knows when it wants the input measurement from the ADC and also when the feedback calculation is finished for the DAC. The micro controller is also linked with a SPI connection to the Raspberry Pi. However for this transmission the Raspberry Pi takes the position of the master in order to initiate the SPI communication. The user controls the functionality of the micro controller on the Raspberry Pi which then sends the appropriate information via the SPI. Measured data on the micro controller can also be sent back to the Raspberry Pi for further analysis, which is for example the

case of the evaluation of the way-points we discuss later in chapter 4.3.1.

The main purpose of our micro controller, as it's name suggests, is to control the other peripheral devices such as the ADC and DACs. The microcontroller can be programmed with a C or C++ program that is loaded in its flash memory and which is executed whenever the device starts. This program contains simple tasks for the real time calculation such as the PID-control we will discuss in chapter 3. The C and C++ are relatively low level programming languages which allow them to be very efficient and fast which is ideal for the real time feedback. More advanced analysis of the data is done on the Raspberry Pi since it would be impossible to calculate everything in real time. As previously mentioned the use of the Python programming language give us easy tools to analyse the data measured on the microcontroller and give more advanced feedback such as the frequency lock we will discuss in chapter 4.

Chapter 3

Intensity Stabilisation

A stable laser intensity is a key element of many optical experiments. The intensity of a laser can be adjusted with an acousto-optic modulator, AOM for short. The AOM consists of a crystal and a piezo-electric transducer which is attached to the crystal. An oscillating signal then drives the piezo to create sound waves in the crystal. These propagating sound waves can also be thought of as particles called phonons. If a photon passes through the AOM and the crystal, it can be hit with a phonon and get diffracted. There are many orders of diffracted beams depending on how many times the photon collides with the phonons. These orders then exit the AOM under different angles because of the momentum conservation. One can then select the desired order by placing a pinhole behind the AOM. If we select for example the first order diffraction we can adjust the intensity of the laser by changing the power of the sound waves. Depending on the AOM power more or less photons get diffracted into the selected mode.

If we now want to stabilise the laser intensity we have to give a feedback from the measured intensity to the AOM. The laser intensity is measured by a photo-diode and converted to an electrical voltage. Our Lock-box measures this voltage and outputs a feedback voltage to an AOM-controller. In the experiment this is the QO-EL-0033 AOM-controller built by Alexander Frank. This AOM-controller contains a voltage variable attenuator which then in turn adjusts the AOM. In the following chapter I will now explain how a PID-controller is used to calculate the appropriate feedback.

3.1 PID Parameters

A proportional–integral–derivative (PID) controller is a conventional mechanism to bring a process variable (laser-intensity) as close as possible to a setpoint. This is done by continuously calculating the error $e(t) = s(t) - p(t)$ between the setpoint and the process variable [7]. This error is then used to

calculate the control value $u(t)$. In our case of intensity stabilisation this would be voltage forwarded to the AOM controller. The control value is given by

$$u(t) = K_P e(t) + K_I \int_0^t e(t')dt' + K_D \frac{\partial e(t)}{\partial t}. \tag{3.1}$$

The formula consist of a proportional, a integral and a differential part which gives rise to the name of the PID controller. The proportional term ensures that the big errors get corrected quickly, the integral term eliminates steady-state errors by adding up the past errors. The differential part predicts future errors like overshoots and acts by eliminating strongly changing errors. There are three constant parameters to control how strong the feedback is from each term.

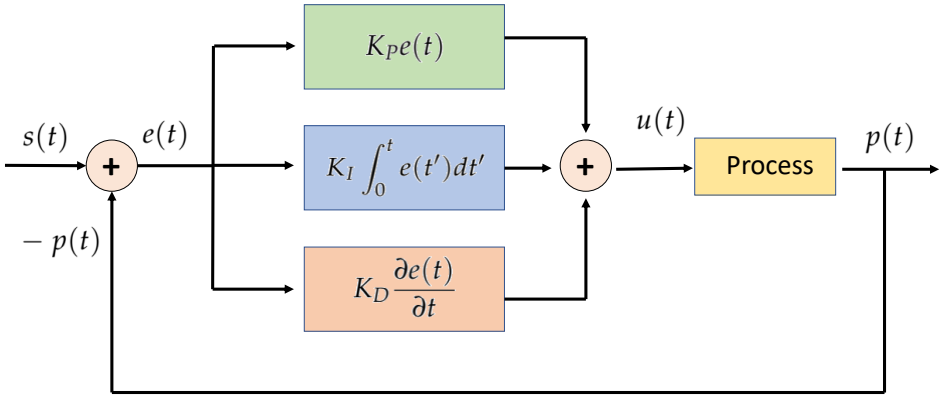


Figure 3.1: Conceptual picture of a PID process. The error e between the process variable p and the set points s gets added up with the PID parameters to the control variable u which in turn influences the process variable p.

To get a good control feedback which reaches the set value quickly, does not overshoot too much and does not oscillate uncontrollably, one needs to find correct parameters. Finding these correct parameters depends on the task of the PID and is normally done by trial and error. There are however also heuristic methods to calculate the parameters, such as the Ziegler–Nichols method [8].

To find correct parameters with the Ziegler–Nichols (ZN) method one first sets the integral and differential gains to zero. The proportional gain K_P is then slowly increased until the process variable starts to oscillate with a stable frequency. At this point the ultimate gain K_u, which is the amplitude of this oscillation, and the oscillation period T_u are determined. With this information the other parameters can be estimated depending on which

	K_P	K_I	K_D
P	$0.5K_u$	-	-
PI	$0.45K_u$	$0.53K_u/T_u$	-
PID	$0.6K_u$	$1.2K_u/T_u$	$3K_uT_u/40$

Table 3.1: Summary of the PID parameters for the Ziegler–Nichols method.

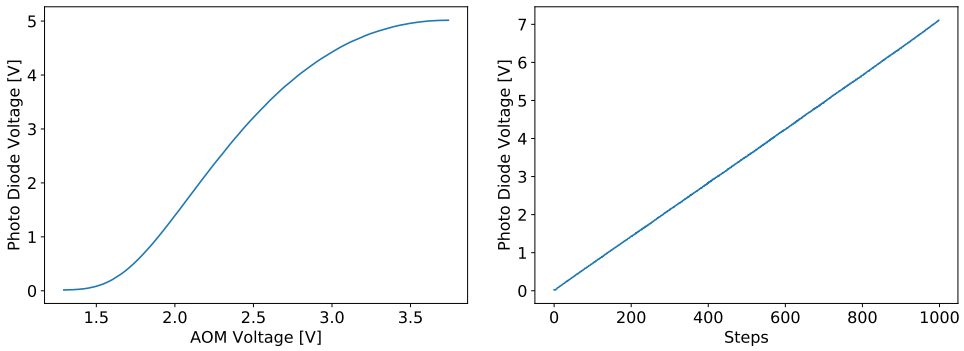
controller type is used. The summary of the different gain parameters can be seen in table 3.1.

The Ziegler–Nichols method produces quite an aggressive controller which is optimized to reject disturbances. This on the other hand leads to overshoot and might not be suitable for certain applications [8]. I will discuss in chapter 3.3 how good the ZN-controller functions for our task of intensity stabilisation. But before that I will first talk about the linearisation of the AOM response.

3.2 Linearisation of AOM response

As mentioned in the previous section, the laser-intensity and therefore the voltage measured on the photo-diode can be controlled by the AOM. However the diffracted light power is a non-linear function of the control voltage. This can be seen in figure 3.2a, the curve is maybe linear for a small section in the middle but at the edges the photo-voltage flattens out. This non-linearity decreases the accuracy of our PID-controller since the feedback would become proportionally smaller the further we are away from the center of the curve.

In order to linearise the AOM response we can use the measurement from figure 3.2a and invert the measured curve. If we then act with this inverted function on the PID-output before outputting it to the AOM the resulting photo-diode voltage should respond linearly to changes in the PID-output. Figure 3.2b shows the photo voltage when the output is adjusted to make up for the non-linearity of the AOM. The curve is now clearly linear and this should increase the performance of our PID.



(a) Transfer function of the Photo-diode when varying AOM voltage.

(b) Measurement of the linearised AOM response.

Figure 3.2: The voltage of the photo-diode does not increase linearly when increasing the AOM voltage in figure 3.2a. Figure 3.2b shows the same measurement after recording and linearising the output from the Lock-box.

3.3 Results

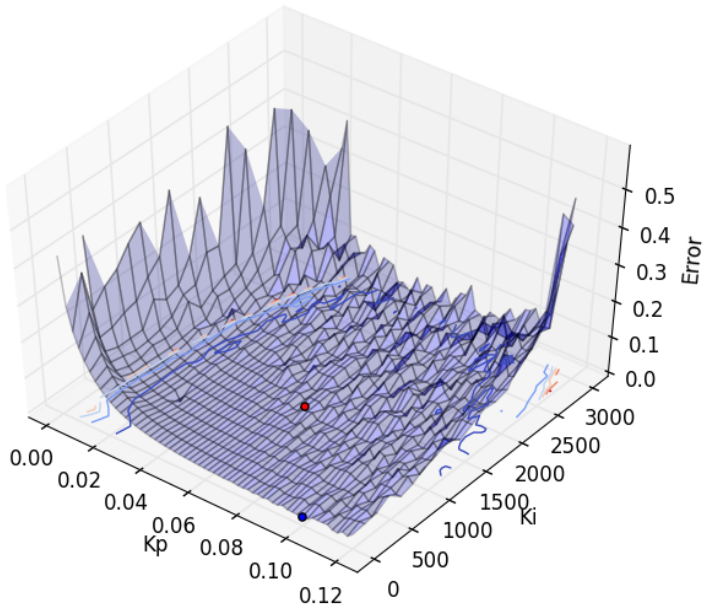
In order to test the PID-controller and the Ziegler–Nichols method we measure the error between the process variable and the set value for different PID-parameters and check how well the heuristic parameters perform in comparison. We used a 1 kHz sine signal between 1 V and 2 V as the set-point and then varied K_P and K_I on a grid to get a surface plot of the error. The 1 V-2 V voltage range was chosen so that the set-point would surely lie in the output range of the AOM driver. The error is calculated as the mean difference between the sine signal and the measured photo-voltage. In a second step we additionally used the AOM-linearisation to see if it improves the PID-controller and reduces the error.

The error-plot can be seen in figure 3.3 where figure 3.3a shows the error without the AOM-linearisation and figure 3.3b shows the error without the linearisation. We can immediately see from the scale of the error-axis that the PID-controller is significantly improved with the AOM-linearisation.

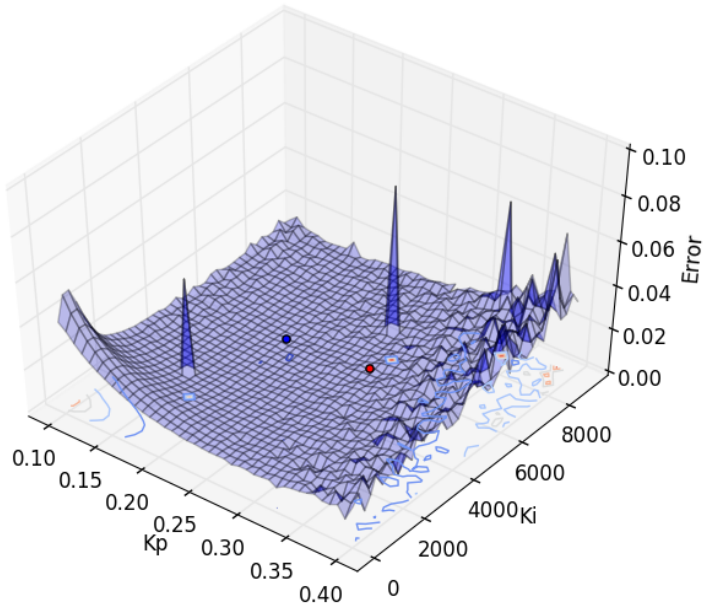
The red dot on the surface indicates the PID-parameters calculated with the Ziegler–Nichols method and the blue dot shows the global minimum of the error-surface. In figure 3.3b the red and blue dots are quite close and the error difference is quite small. In figure 3.3a however the dots are further apart. Especially the blue dot seems to prioritize parameters with very low integral part. This could come from the fact that our sine signal is oscillating with 1 kHz and therefore the integral part is too slow to follow the signal and just introduces additional error. The error of the ZN-parameters nevertheless is not much larger than the global minimum. To be sure how

well the PID performs we have to look at the actual signals and see how well they agree with the sine signal we want to achieve.

We measured the photo-voltage signals for the parameters determined in the previous plot. The four different signals can be seen in figure 3.4. Figures 3.4a and 3.4b show the signal with the parameters calculated from the ZN method whereas figures 3.4c and 3.4d are the global minima parameters. We can immediately see that the overshoot of the ZN parameters is larger than the one from the global minimum. This is as expected since we already mentioned that the Ziegler–Nichols method is quite an aggressive controller and leads to overshoots [8]. But we also see that the ZN controller converges to the sine wave faster especially when we compare figures 3.4a and 3.4c. The figures with the AOM-linearisation are quite similar to each other since the two parameters are also close in figure 3.3b but again the ZN-method produces a larger overshoot. Also the figures with the AOM-linearisation show a major improvement compared to the signals without the linearisation which was also to be expected since we already saw the smaller error in the previous plot.

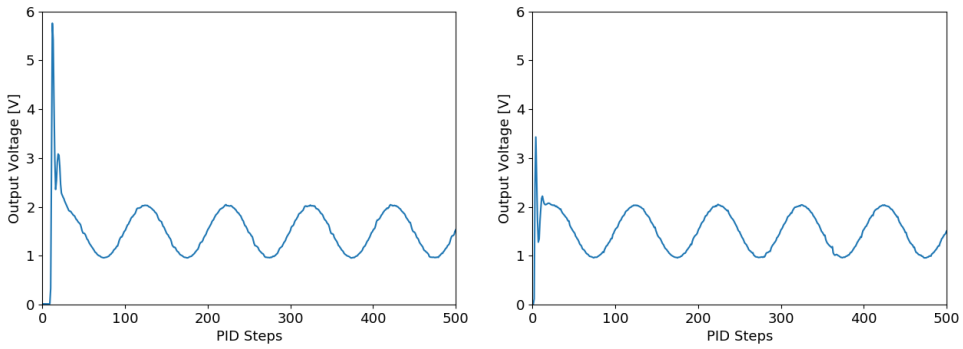


(a) Error-plot without the AOM-linearisation.



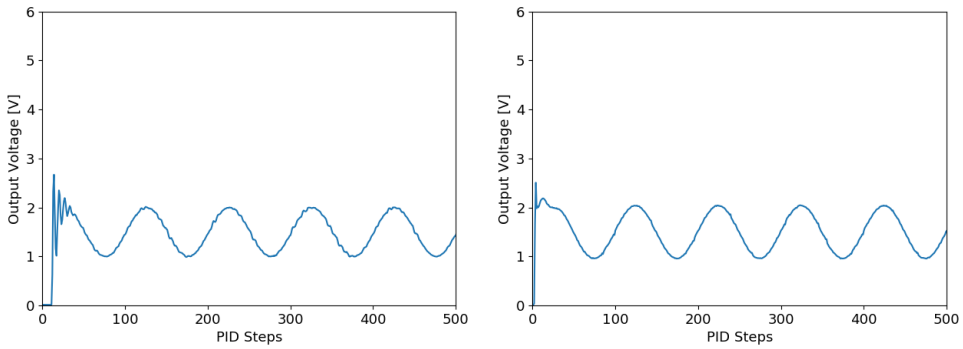
(b) Error-plot with the AOM-linearisation.

Figure 3.3: Figures 3.3b and 3.3b show a surface plot of error between the desired setpoint-signal and the laser-intensity for different PID parameters. In this case K_P and K_I are varied on the xy-plane and the mean error is plotted in z-direction. The red dot indicates the heuristic ZN-parameters and the blue dot shows the global minimum of the surface.



(a) Output signal with ZN-parameters.

(b) Output signal with ZN-parameters and AOM-linearisation.



(c) Global error minimum output signal.

(d) Global error minimum output signal with AOM-linearisation.

Figure 3.4: The figures on the left side are taken without the AOM-linearisation and the ones on the right with the linearisation. Figures 3.4a and 3.4b show the sine signal of the laser-intensity with PID parameters calculated from the heuristic Ziegler–Nichols method. Figures 3.4c and 3.4d show the signal for the global minimum parameters.

Chapter 4

Frequency Lock

Optical cavities have a lot of applications in quantum optical research such as narrowing laser line widths, increasing laser power and are used for their strong light-matter coupling. These experiments however rely on mechanically stable cavities since they are are sensitive to changes of the cavity length in the sub nanometre regime. These variations in the cavity length can come from thermal expansion, mechanical vibrations or from sound waves. These effects are almost impossible to remove entirely and therefore active stabilisation of the cavity is needed. The idea of the frequency lock is to use a signal called error-signal to give electrical feedback to the cavity. In our case we use the electrical feedback to stabilise the optical cavity via the piezoelectric actuators on top of which the cavity mirrors are mounted. These piezos allow us to change the cavity length and keep them at a stable distance. In this chapter I will give a brief overview of the most widely used method, the Pound–Drever–Hall (PDH) laser frequency stabilization and the role of the error-signal. Although this chapter is specifically about cavity locks, many of the principles apply to other kinds of frequency locks, such as spectroscopy and offset locks. Further information about the error-signal can be found in the original article by Drever [9] or in this summary [10].

4.1 Introduction to Error-Signals

When one shines a laser beam through an optical cavity the light gets reflected or transmitted through the cavity depending on the laser frequency. If the laser frequency is close to an integer multiple of the free spectral range of the cavity $FSR = \frac{c}{2L}$ where L is the distance between the cavity mirrors, the light can pass through and the transmission is high. If however the light frequency differs from the resonance frequency of the cavity, the intensity of the transmitted light decreases like a Lorentzian in frequency. We can see

this in figure 4.1 where the inverse of the transmission, the reflection, is plotted against the frequency. This effect can also be understood when looking at the phase of the reflected beam since on resonance the phase difference to the leakage beam of the cavity will be 180°and they will cancel each other out.

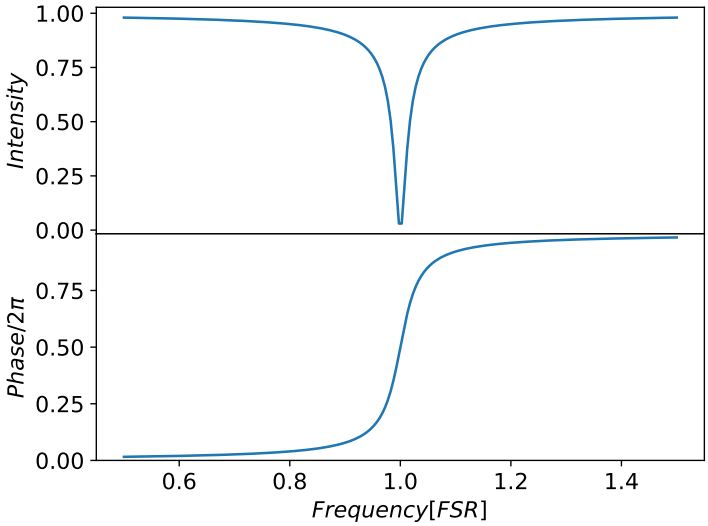


Figure 4.1: Phase and Intensity of a beam reflected at the optical cavity as a function of frequency. The reflected intensity decreases like a Lorentzian on resonance. The phase shift of the reflected beam on resonance is 180°.

If we now measured for example the reflected beam intensity at resonance we would see the the intensity at zero and would immediately see a drift in the frequency by a rising reflection signal. The problem however is that we wouldn't know in which direction the frequency drift took place since the frequency response is symmetrical about the resonance. If we would measure the derivative of the reflected signal we would get an antisymmetric response to frequency drifts and we could use this error-signal for the electrical feedback. In practice this is being done by modulating the phase of the laser by an electro-optic modulator (EOM). After this phase modulation the incoming electric field looks like this:

$$E_{\text{in}} = E_0 e^{i(\omega t + \beta \Omega t)}. \tag{4.1}$$

We can then expand this term into Bessel functions,

$$E_{\text{in}} \approx [J_0(\beta) e^{i\omega t} + J_1(\beta) e^{i(\omega+\Omega)t} - J_1(\beta) e^{i(\omega-\Omega)t}]. \tag{4.2}$$

In this form it is easy to see that there are actually three different beams in the incoming light. One is the carrier beam with a frequency ω, the other two are side-bands with frequencies $\omega \pm \Omega$. The reflected beam is then given by

$$\begin{aligned} E_{\text{refl}} = E_0[F(\omega)J_0(\beta)e^{i\omega t} + F(\omega+\Omega)J_1(\beta)e^{i(\omega+\Omega)t} \\ -F(\omega-\Omega)J_1(\beta)e^{i(\omega-\Omega)t}] \end{aligned} \tag{4.3}$$

where $F(\omega) = E_{\text{refl}}/E_{\text{in}}$ is the frequency dependent reflection coefficient of the cavity. When we measure the reflected beam with a photo-detector, what we actually measure is the power of the reflected beam,

$$\begin{aligned} P_{\text{refl}} &= P_c|F(\omega)|^2 + P_s\{|F(\omega+\Omega)|^2 + |F(\omega-\Omega)|^2\} \\ &\quad + 2\sqrt{P_cP_s}\{Re[F(\omega)F^*(\omega+\Omega) - F^*(\omega)F(\omega-\Omega)]\cos\Omega t \\ &\quad + Im[F(\omega)F^*(\omega+\Omega) - F^*(\omega)F(\omega-\Omega)]\sin\Omega t+\} + \mathcal{O}(2\Omega), \end{aligned} \tag{4.4}$$

where $P_c = J_0^2(\beta)|E_0|^2$ is the power in the carrier and $P_s = J_1^2(\beta)|E_0|^2$ the power in each first-order side-band when we neglect interference terms. What we are interested in measuring are the two terms oscillating in Ω, because they contain the phase of the reflected beam. The phase tells us on which side of the resonance we are, which is exactly what we need for the error-signal. To achieve this measurement in practice we use a mixer which multiplies two input signals. We can then use the formula

$$\sin(\Omega t)\sin(\Omega' t) = \frac{1}{2}\{\cos[(\Omega-\Omega')t] - \cos[(\Omega+\Omega')t]\} \tag{4.5}$$

to see that if we mix the reflected beam with the modulation signal $\sin(\Omega t)$, the term $\cos[(\Omega-\Omega')t]$ is then a DC signal which we can single out with a low-pass filter. If we assume that the carrier is near resonance and the side-bands are far enough detuned that they are almost totally reflected then we arrive at the final expression

$$\varepsilon = -2\sqrt{P_cP_s}\text{Im}\{F(\omega)F^*(\omega+\Omega) - F^*(\omega)F(\omega-\Omega)\}. \tag{4.6}$$

The normalised error signal from equation4.6 is plotted in figure 4.2 as a function of frequency. We clearly see the two side-bands and the carrier frequency which has a zero crossing at resonance. This is also the result we observe in the experimental set-up. A plot of an experimental error-signal can be seen in figure 4.3. The shape of the signal is in good agreement with the theoretical one. Also we see how this signal can now be used for the electrical feedback to either the laser or the cavity. Since the error-signal has a clear slope around the resonance we can now determine in which direction the frequency shifts take place.

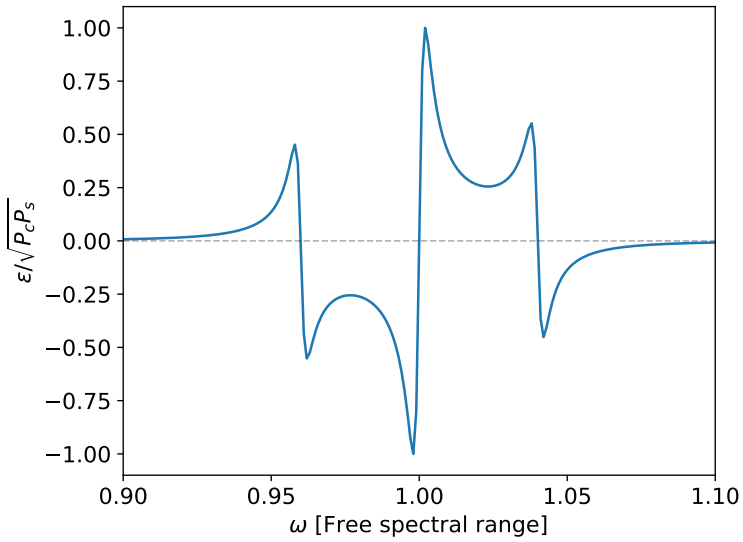


Figure 4.2: Theoretical shape of the Error-signal from equation 4.6 for a modulation frequency (Ω) of 4% of the free spectral range and a reflectivity of 0.999.

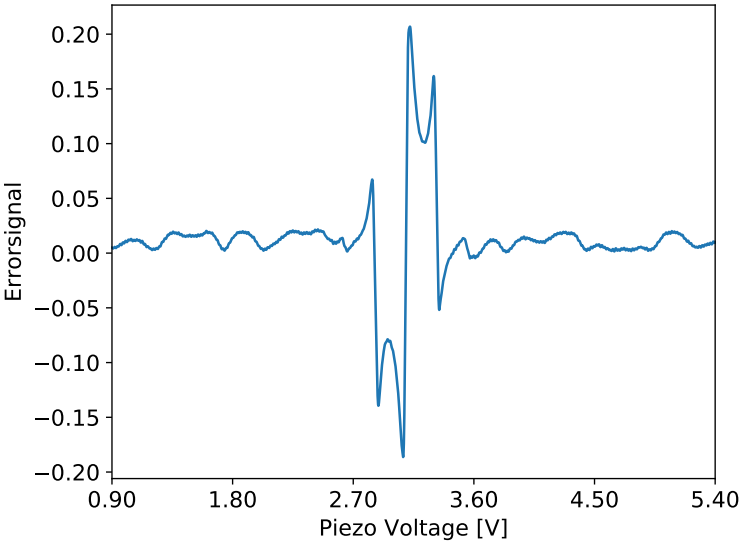


Figure 4.3: Typical example of a low-pass filtered Error-signal from an experiment.

4.2 Hardware Set-Up

When testing out the previous version of the Lock-box we used the set-up pictured in figure 4.4 to perform the frequency lock. The cavity consists of two mirrors which are mounted on piezo-electric actuators which control the cavity length. From the cavity we get a reflection signal, or sometimes a transmission signal, and the error-signal, which we previously discussed. These two signals are fed into the two analog inputs of the Lock-box for the implementation of the frequency lock. We used one of the analog outputs to amplify the $\pm 10\,\mathrm{V}$ output range of the DAC to the $\pm 200\,\mathrm{V}$ needed to drive the piezo.

The amplifier however was low-pass filtered to reduce high frequency noise. So, we needed a faster signal for more precise control of the PID. The second output was then not amplified and thus connected unfiltered to the second piezo. Both of the DACs in the previous version had only 16-bit accuracy which lowered our resolution considerably. Because the cavity had a linewidth of only 147 kHz this meant that we had to use a $\pm 1\,\mathrm{V}$ output range for the fast DAC output to achieve a decent 3 kHz resolution. On the other hand, this reduced output range gave us some other problem we will discuss later in section 4.3.

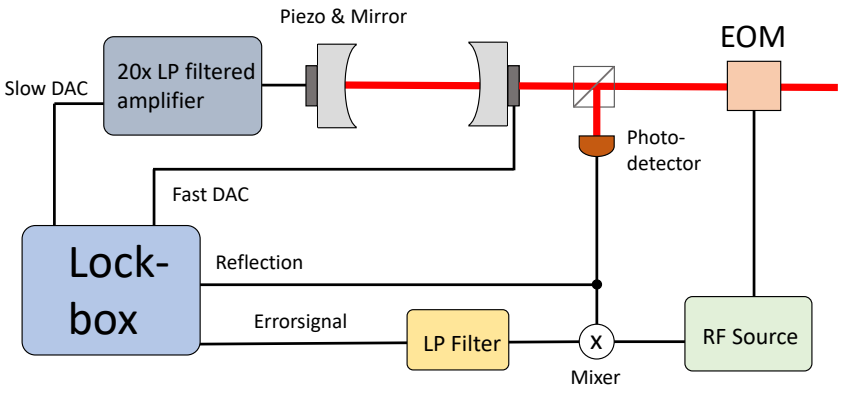


Figure 4.4: Schematic overview of the hardware set-up. The two analog outputs are used to control the mirror piezos. One output is amplified and low-pass filtered and the other goes directly to the piezo. As analog inputs into the Lock-box we have the reflection, or sometimes transmission, signal and the error-signal.

4.3 Lock Strategies and Problems

Like for the intensity stabilisation the micro controller uses a PID control loop to give feedback to the piezoelectric actuators which control the mirror positions of the optical cavity. The feedback of the PID should now stabilise the cavity insofar appropriate parameters are found.

An additional difficulty here arises now from the fact that within a free spectral range multiple resonances are observed as seen in figure 4.6. This stems from the fact that different transverse electromagnetic modes (TEM) can be present in the cavity. An example of such modes are the Hermite-Gaussian modes seen in figure 4.5. For most applications one wants to lock to the TEM_{00} mode which has the most prominent error-signal in the spectrum if the beam is properly aligned and can thus be conveniently singled-out. Now we have to come up with a scheme to find, identify and lock to the proper modes.

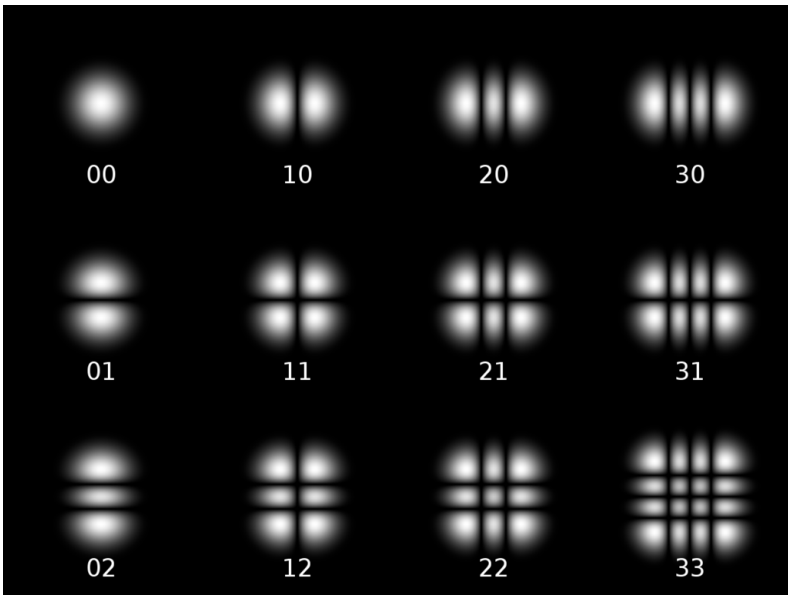


Figure 4.5: Picture of Hermite-Gaussian TEM_{mn} modes from [11].

4.3.1 Waypoint Lock Strategy

The most straightforward approach to finding the TEM_{00} mode would be to scan the entire cavity with the piezoelectric actuator and then to save the piezo-voltage where the largest peak occurs or use a camera to identify the fundamental mode of the beam. But this generates a few problems, on of which is the hysteresis of the piezo, which shifts the position depending on the past state of the piezo. Hence if we scan the cavity in one direction,

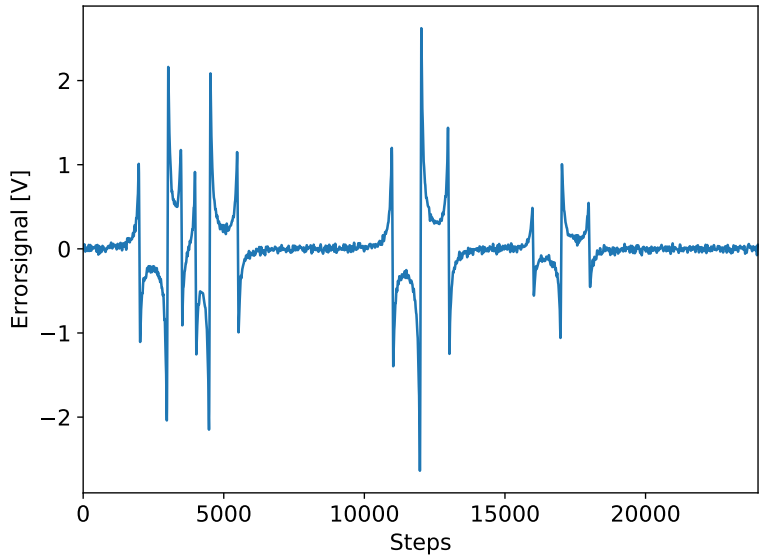


Figure 4.6: Multiple modes in the error-signal over a full scan of the cavity piezos.

identify the peak we want to lock on and then return to said voltage, the signal will have shifted relative to the first scan. Another problem is that the cavity mirrors can drift through thermal expansion or mechanical noise. This leads to a shift in the spectrum which also shifts the piezo-voltage at which we would expect our peak to be. These circumstances show that we can not use the piezo-voltage as a reliable source of information about the position of the TEM_{00} mode.

We developed a different strategy which relies on the use of waypoints to identify the position of the piezo. Waypoints are alternating maxima and minima of the error-signal as seen in figure 4.7. The idea is that we don't save the piezo-voltage of the waypoints, but only their amplitude. If we move in one direction we can consecutively check off waypoints as soon as the error-signal reaches the threshold amplitude. And so we know if the piezo-position is inside the mode signal or between different modes based on how many waypoints have already been passed.

Therefore the procedure to lock is the following: First we start of by taking a scan over the entire range of the piezos and measure the error-signal. The measured data gets the to the Raspberry Pi and then analysed. When the correct mode is identified and the waypoints calculated said data is sent back to the micro controller. The micro controller now knows at which waypoints the desired mode is located and can traverse with the piezo to this mode. When all waypoints are passed and the error-signal crosses the zero point

the lock can be started and the PID should keep the the cavity on resonance. This procedure is schematically drawn in figure 4.8.

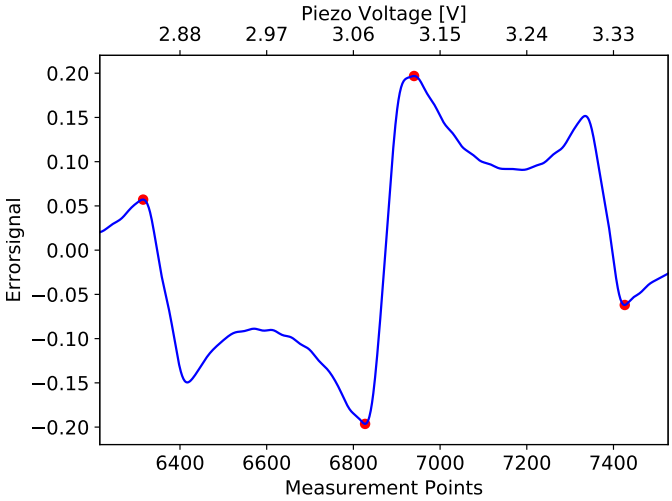


Figure 4.7: Way-points are placed throughout the error-signal to identify its shape and position. The red points are sent to the microcontroller to help it navigate up and down the spectrum.

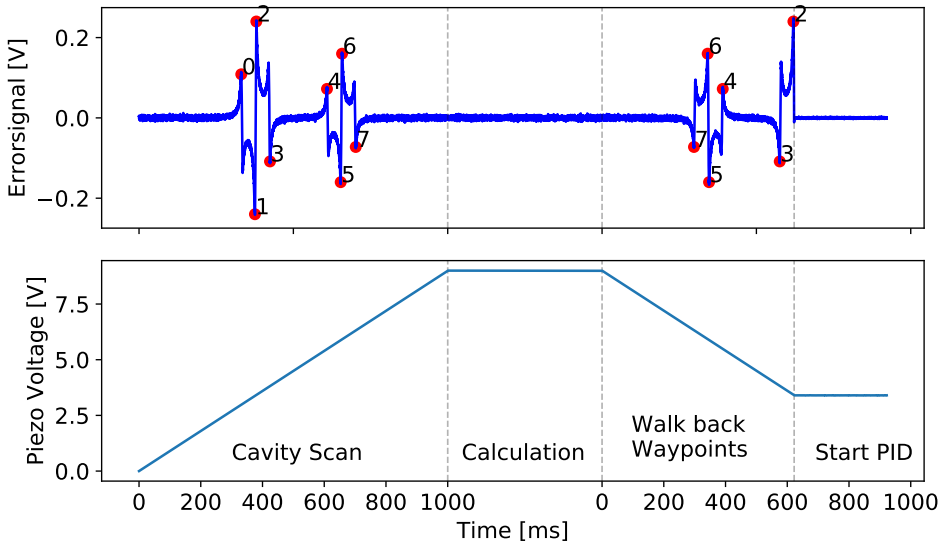


Figure 4.8: Schematic figure of the procedure to scan, identify and lock to the desired zero-crossing.

4.3.2 Problems

The reliance on the amplitude of the error-signal brings with it also a few problems. The method relies on the fact that the amplitudes of the error signal stay constant. Still the amplitude can change if for example the laser intensity changes, but this should pose no problem since only the absolute amplitudes change and the relative amplitudes between the modes should stay constant with varying laser intensity.

Another requirement is that the noise of error-signal has to be smaller than all of the maxima and minima of the signal in order to not accidentally check off a waypoint and lose the information about the position. Therefore not all modes can be identified with the waypoints since some of them might be so small that their waypoints get checked-off by the error-signal noise. But those signals would be hard to identify even by hand and the modes we want to lock to usually are mostly very pronounced and easy to identify.

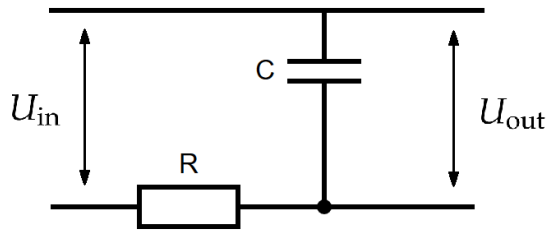


Figure 4.9: Circuit diagram of a simple RC low-pass filter.

During the testing of the Waypoint-strategy we came across a few different difficulties which we had to solve. The first thing we noticed was that the piezo seemed to continue moving even though the ramp had stopped at the correct voltage. We soon realized that this was due to a low-pass filtering which made the voltage climb further up the ramp instead of stopping. This is because the amplifier we use to bring the voltage from the Lock-box to the piezo voltage is low-pass filtered. This is however not the only possible source of low-pass filtering in the system.

A RC low-pass filter consist of a resistance and a capacitance which is connected to the ground as pictures in figure 4.9. The cut-off frequency of this low-pass filter is then given by $f = 1/(2\pi RC)$. The piezo itself has a small capacitance which combined with the output impedance of the Lock-box can lead to such a low-pass filter.

To see how we can fix this we can look at the function of a low-pass filtered ramp and compare how it differs from the unfiltered one. The unfil-

tered ramp starts at zero voltage and increases linearly until time τ when it reaches the voltage U_0.

$$U_{\text{in}}(t) = \begin{cases} 0 & \text{for } t < 0 \\ \frac{U_0}{\tau} t & \text{for } 0 \leq t \leq \tau \\ U_0 & \text{for } \tau < t \end{cases} \tag{4.7}$$

The low-pass filtered ramp then has the following form

$$U_{\text{out}}(t) = \begin{cases} 0 & \text{for } t < 0 \\ \frac{U_0}{\tau} \left(t - RC \left(1 - e^{-t/(RC)} \right) \right) & \text{for } 0 \leq t \leq \tau \\ U_0 - \frac{U_0}{\tau} (RC) \left(1 - e^{-\tau/(RC)} \right) e^{-(t-\tau)/(RC)} & \text{for } \tau < t \end{cases} . \tag{4.8}$$

The full derivation of this result can be found in appendix chapter A. For times $0 \leq t \leq \tau$ when the the voltage is still inside the ramp the unfiltered voltage has the form $\frac{U_0}{\tau} t$. Comparing this to the filtered signal, we see that the ramp has an exponential transition into a ramp that is delayed by RC, where R is the resistance of the resistor and C the capacitance of the capacitor as in figure 4.9. At time τ when the signal should stop, the filtered voltage is lowered by $\frac{U_0}{\tau} (RC) \left(1 - e^{-\tau/(RC)} \right)$ compared to the unfiltered voltage and then exponentially approaches the value U_0. So if we abruptly stop the ramp the voltage will still continue to climb because the filtered signal is delayed to the unfiltered one. We can however prevent this by making a jump in the voltage we output to match the filtered voltage. If we assume $\tau \gg RC$ then we can reduce our output by $\frac{U_0}{\tau} RC$ at time τ to keep the filtered voltage constant for $t > \tau$.

Implementing this jump of the output voltage as well as using two outputs, one amplified and one without amplification, on the two cavity mirrors solved the low-pass filtering problem and gave us much better control over the cavity. However, an additional problem revealed itself as we fixed the previous errors. Even when the low-pass filter was removed the signal seemed to continue moving on its own as soon as the cavity was locked. Firstly this was not a problem since the PID kept the cavity locked, but when the PID was running on the non amplified piezo the voltage range of the DAC was not big enough to control the cavity. The error signal then drifted outside the of the DAC range and the lock stopped working. This phenomena is mostly likely due to the heating of the mirrors which then changes the cavity length through thermal expansion as soon as the cavity is locked on the carrier frequency. This problem can however be easily solved by using the amplified signal for the PID and thus increasing the voltage range. On the other hand this decrease the resolution of the PID and worsens the

stabilisation. This is one of the reasons why we upgraded to a 20-bit DAC in the new hardware, which allows us to run an accurate PID even when we are amplifying the signal and decrease it's resolution. The increased range and flexibility of the 20-bit DAC should make the future frequency locks significantly easier and combined with the removal of the output impedance of the Lock-box this should solve all the previous problems with the frequency lock.

Chapter 5

New Hardware

As a first part of my thesis I tested the previous device which was designed and built by Philip Zupancic. When testing the Lock-box in different scenarios and combined with different lab equipment in the experiments I came across different problems and weaknesses of the device. This would then lead to the development of a second device to improve these aspects.

One problem of the previous device were the input and output impedances. Normally a really large input impedance is chosen such that the device draws as little current as possible and can interface with other circuitry that might otherwise not be able to provide enough power [12]. On the other hand the output impedance should be zero such that the device can drive other hardware that draws a lot of current [13]. These requirements had to be considered for the choice of hardware and the design of the circuit.

Another problem was the use of differential input for the ADC which sometimes seemed to be more noisy than a single ended input. I then changed the design of the input in the new version to allow for differential or single-ended to be selected by a jumper cable on the board.

Also the DAC had to be upgraded to 20-bit precision from the 16 bits of the preceding version. Instead of the 65′536 possible outputs of the previous DAC this gives us 1′048′576 outputs which corresponds to a resolution of over 1 ppm (part per million). The reason for this is that the full spectral range (FSR) of the cavities in the experiments are about 60 GHz and the FWHM (full width half maximum) of the carrier in the error-signal is about 300 kHz. This means that we need a resolution better than

$$300\,\mathrm{kHz}/60\,\mathrm{GHz} = \frac{1}{200'000} \tag{5.1}$$

which means that the 1 ppm resolution of the 20-bit DAC is just enough to detect the peak in the error-signal.

Furthermore I changed the layout so that the board fits into a standard 19 inch rack and can be powered and connected through a backplane. But the concept of the Lock-box stayed mostly the same, there are still 2 analog inputs and outputs, a microcontroller for real time processing and the Raspberry Pi for the more advanced calculation and communication as already discussed in chapter 2.

In this chapter I will talk mainly about the changes in the new version and the contributions I made to the project. But I will also discuss a bit about the general design principles when building such a device.

5.1 Components

5.1.1 ADC

One of the most important components of the Lock-box is the analog digital converter (ADC). The ADC allows to measure the signals from the experiment and sends them to the micro controller for further operation. In this version of the Lock-box we decided to use the LTC2353 Dual 16-bit ADC with $>1000\,\mathrm{G}\Omega$ input impedance [12]. This ADC uses a successive approximation register (SAR) to digitise the analog signal. The SAR contains a series of decreasing voltages which are used to approximate the input voltage. Starting at the highest voltage the SAR decides if turning it off or on comes closer to the supplied voltage and stores the result. The next voltage is then half of the previous voltage and the SAR again tries to approximate the voltage to the input. When going through all of the registers this converts the analog input into a binary digital number which can then be output via the SPI.

The LTC2353 has two input channels. In a frequency lock, for example, one is used for the error-signal and the other for the transmission signal of the optical cavity. Both of the channels of the LTC2353 can be operated either as a fully differential input where the positive and negative input of the channel is connected to the BNC cable, or the channel can be operated as a bipolar single ended input where the signal is measured relative to the ground of the Lock-box. The two modes can be selected by setting jumpers on the board.

In the previous version two separate 16-bit ADCs were used to measure the two signals. The Dual ADC has the advantage that it measures the two signals at the same time, which is what the preceding device could not guarantee since the two ADCs operated separately and the micro controller might send the signal to start the signal conversion not simultaneously.

Another advantage of a single ADC is that we also only need a single SPI connection between the ADC and the micro controller. This eliminates the

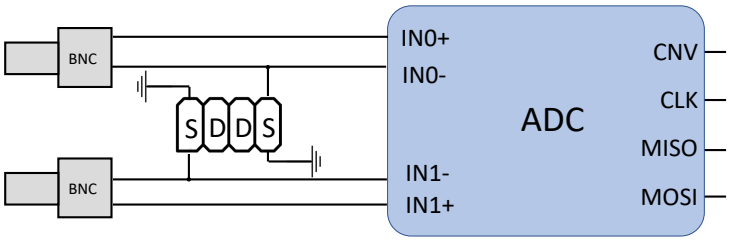


Figure 5.1: Schematic view of the ADC and the connections. The differential (D) or single-ended (S) input can be selected by setting the jumper in the appropriate place for each of the channels.

need for more fast digital SPI wires which can couple a lot of noise into the analog part of the board. Also, the single data transaction is a bit faster than the two separate connections since the micro controller can only read a single SPI connection at a time and each transaction creates a certain overhead. With only one SPI the micro controller needs to establish and read a single transaction.

The input range of the channels can be set by writing into the setting register via the SPI connection. The ADC supports bipolar $\pm 10.24\,\mathrm{V}$, $\pm 5.12\,\mathrm{V}$ as well as unipolar $5.12\,\mathrm{V}$ and $10.24\,\mathrm{V}$ settings to increase the input accuracy.

The LTC2353 offers a host of different features such as a 124 dB common-mode rejection ratio (CMRR) or a 94.2 dB signal to noise ration (SNR) [12]. The common-mode rejection discards signal that is common to both inputs and is important if we want to measure a differential signal [14]. A good signal to noise ratio on the other hand allows us to take accurate measurements. We can compare the value from the data sheet with the SNR of an ideal ADC:

$$SNR_{dB} = 6.02_{dB} \cdot N + 1.76_{dB} \tag{5.2}$$

where N is the number of bits of the ADC. For our ADC this would equal $SNR = 98.08\,\mathrm{dB}$. But keep in mind that the noise in the ideal ADC only results from the limited resolution of the bits [14]. The rest of the noise comes from the thermal noise or the noise of the voltage reference. Therefore the LTC2353 allows for accurate measurements of the error and transmission signals. Another important specification of the ADC is the integral non-linearity (INL) which specifies how much the ADC transfer function differs from an ideal transfer function as pictured in figure 5.2. For the LTC2353 the INL error is ± 1 least significant bit (LSB) which means that the smallest bit of the output code might differ from the ideal value.

To initiate the measurement of the signal the microcontroller sets the conversion (CNV) pin of the ADC to a high output level. This starts the conversion which needs at least 1 μs to acquire the result. As soon as the conversion is done the two results can be sent via the SPI to the microcontroller. The maximum speed of the SPI connection is 100 MHz, thus the time needed to transmit the 48-bit of data needs atleast 0.5 μs. If we include other delays and add some safety margin we can assume a maximum of 2 μs per measurement cycle. Therefore the theoretical maximum speed of the ADC is about 500 kHz.

I will discuss more specifications of the ADC and examine how it performs on the actual board in chapter 5.3.

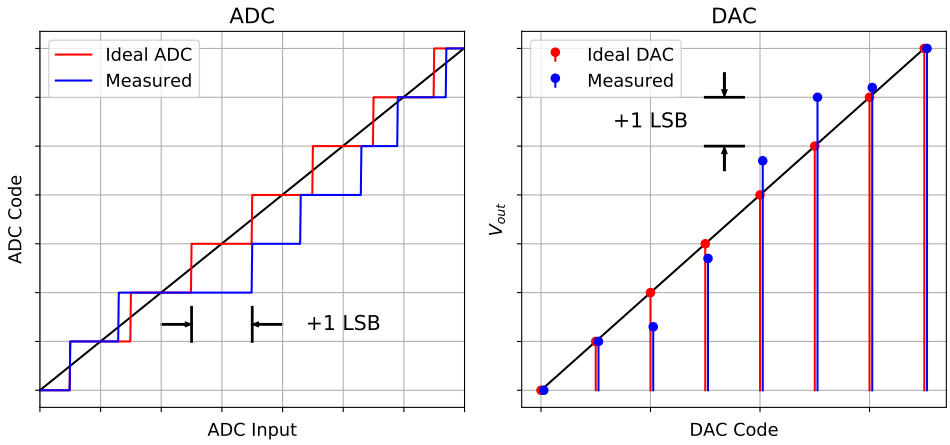


Figure 5.2: This plot shows how the integrated non linearity (INL) is defined and determined for both the ADC and DAC.

5.1.2 DAC

The digital analog converter (DAC) is used to give the electrical feedback from the micro controller. We have two DACs on the board which are used for the two analog outputs. The DAC we used in this version is the AD5791 from Analog Devices, which is a single channel 20-bit voltage output DAC [15]. The AD5791 uses a two matched DAC rail to rail architecture. Rail to rail means that the DAC can output voltages between the supplied positive and negative voltage. The two matched DAC architecture consists of a DAC for the 6 most significant bits and one DAC for the other 14 bits, which are then added up to ensure us the 20-bit accuracy. The 20-bit accuracy of the AD5791 gives us the 1 ppm resolution needed for finding the signal as we already discussed in equation 5.1. The AD5791 also has a settling time of 1 μs, which means the signal needs some time after a jump in the output

voltage to stop oscillating and settle at the set level. However, this should pose no problem since the ADC needs at least 2 μs time to read and send signal. This means our sampling rate and therefore the rate of the feedback can't be faster than the 2 μs anyway.

The output range of DACs is set by supplying voltage references for its lower and upper end, as in figure 5.3. On our board we have the ADR445 5 V low noise, high precision voltage reference [16]. The ADR445 has a 2.25 μV peak to peak voltage noise and only a 3 ppm/C temperature drift, which makes it an ideal voltage reference of our DAC. The voltage can then be adjusted by an Op-amp and some resistors. For the resistors we us the NOMCA resistor-network by Vishay [17]. The resistor-network fabrication ensures us that the relative error between the resistances is minimal. For the NOMCA we have a 0.5 % absolute error and a 0.05 % relative error. However, only the relative error is relevant for an amplifier since the amplification is calculated as the quotient of two resistances. The circuit in figure 5.3 consist of a inverting-amplifier and a non-inverting-amplifier to create our two reference voltages for the DAC. The amplification can then be adjusted by selecting different jumper settings on the board according to figure 5.4. The available voltages are $\pm$5 V, $\pm$10 V as well as unipolar +5 V and +10 V. The latter options are particularly important for driving a unipolar piezo which only takes positive input voltages. A typical piezo however is driven by voltages up to 200 V therefore we need to use a 26 dB = 20$\times$ voltage multiplier in order to get our 200 V or 100 V respectively.

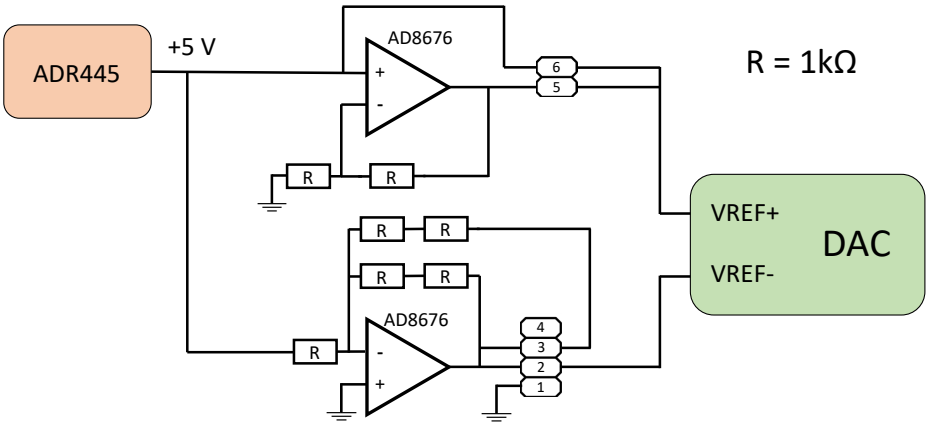


Figure 5.3: Schematic overview of the voltage reference for the DAC. The +5 V signal is adjusted by the Op-amps and can then be selected with the jumper on the board according to 5.4a.

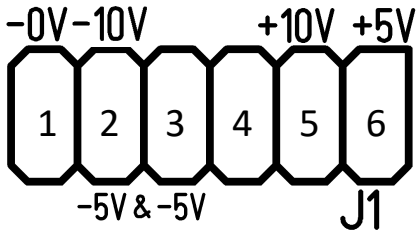
Another change in the design compared to the previous version is the addition of an Op-amp at the output to remove the output impedance of the DAC. This allows the Lock-box to drive current signals as previously mentioned in the introduction to this chapter.

All of the Op-amps on the board are either the AD8675 or the AD8676 as recommended in [13]. The AD8675 and AD8676 are high precision rail-to-rail output Op-amps. The AD8675 and AD8676 have a very low input noise and a offset voltage of only $12\,\mu$V [18] which is perfectly suited for our application and especially for the voltage reference of the DAC [13].

Before going into the output Op-amp however the signal is low-pass filtered by connecting a capacitor between the signal and ground. The DAC has an output impedance of $3.4\,\text{k}\Omega$ which we can use in combination with a $40\,\text{pF}$ capacitance to create a RC filter with frequency

$$\frac{1}{2\pi RC} = 1.17\,\text{MHz}. \tag{5.3}$$

The measurements to verify the frequency of this filter is then discussed in section 5.3.



(a) Description of the Jumpers for the DAC as printed on the board.

Jumpers	Output
1 & 6	0 V/5 V
1 & 5	0 V/10 V
2 & 3 & 6	−5 V/5 V
2 & 5	−10 V/10 V
4	Unused

(b) Table of Jumper settings.

Figure 5.4: To set the appropriate output range for the DACs one has to set certain jumpers on the board to change the reference voltage. The available settings can be seen in the table 5.4a. Jumper setting number 4 is unused and can be selected to store one of the jumpers if only two jumpers are needed.

5.1.3 Microcontroller

The heart of the Lock-box is formed by the microcontroller which manages the other two previously mentioned components and calculates the real time feedback for our applications. We use the microcontroller of the same model as in the previous iteration, the STM32F427 in the LQFP144 package, so that we do not have to change much in the programming and the wiring of the

LTC2353-16	AD5791
16-bit	20-bit
± 1 INL	± 1 INL
94.2 dB SNR	1 ppm resolution
2 μs measurement cycle	1 μs settling time
> 1000 GΩ Input impedance	3400 Ω Output impedance (0 Ω after Op-amp)
124 dB CMRR	-

Table 5.1: Summary of the features of the new ADC and DAC devices in the second version of the project.

device and we could reuse most of the previous work. The STM32F427 contains a 32-bit ARM Cortex-M4 processor which possesses a digital signal processor (DSP) and a floating point unit (FPU) which makes it ideal for our application. The LQFP144 package is a low profile quad flat package with a 144 pins. Some of these pins are used as supply pins to power the microcontroller, the rest of the pins can be configured within the program which is loaded onto the microcontroller. The pins can for example be configured as SPI pins or as general purpose in-out (GPIO) pins. These GPIO pins can be used as digital input or output signals or can be connected to LEDs that can then be lit up through the program which helps debugging the code immensely.

We program the microcontroller in the C and C++ languages because they are both fast and efficient and offer advanced functions like object oriented programming. They are also easy to develop and to maintain compared to an even lower level programming language like assembler. The configuration of the clock parameters, the pins and the SPI connections is generated by the CubeMX program by ST. We then use the TrueStudio integrated development environment (IDE) by Atollic for the development and debugging the program code. For the most part we use the HAL library for the programming with the exception of a few routines we rewrote for optimisation purposes and the `arm_math` package which is used for efficient calculation on the processor. The finished code is then compiled by the GNU gcc compiler with the Thumb2 instruction set, which is optimized for ARM Cortex-M4 processors. An instruction set is a numbered list of commands, also called machine language, for the processor like adding certain registers, jumping to an address in the RAM, conditional statements or single-cycle floating point operations in the FPU. The compiled code is then loaded into the flash memory of the microcontroller by connecting a USB cable to the joint test action group (JTAG) interface on the board which connects to the microcontroller.

Programming on a microcontroller is quite a bit different than what one might be accustomed to when programming on a computer. The microcontroller uses an event-based system to process the different tasks. For example, if a timer or a SPI transaction is finished, an interrupt is generated and a certain piece of code is executed. Such an interrupt routine can be seen in a pseudo code example in figure 5.5. The most important thing about programming with interrupts is to make the interrupt routine as short as possible. This is because the different interrupts block each other from progressing and only one interrupt can be handled at a time. To this end, the code inside the interrupt routine should be kept as short as possible with the use of flags. Flags are boolean variables which store the state of certain tasks and are then later processed in the main loop. Coming back to the code example in 5.5 we see that inside the SPI callback the `ADC_value_ready` flag is set to true and the interrupt is exited. The received value is then only later processed in the main function. The main function contains a while-true loop which loops over different if-statements which get executed as soon as the appropriate flag is set. This structure simplifies the flow of the program immensely and helps with making it more resilient to run-time errors. This is because otherwise an interrupt could be triggered when we do not want to or are not ready to process yet, which then changes variables when they ought to be constant.

We can examine how the microcontroller communicates with the peripherals more closely in the two pseudo code excerpts in figure 5.6. Displayed are parts of the main file and the `ADC_Dev` class which are concerned with the measurement process of the ADC. The sampling process starts by the `SetSamplingFrequency` command which defines the frequency at which the samples are to be taken and starts the sampling clock `htim2`. After one period of the sampling clock the function `HAL_TIM_PeriodElapsedCallback` is called and the `Read` function of the ADC class is subsequently carried out. In this function the conversion of the ADC is initiated, if the device is not already busy, by setting the `CNV_Pin` to a high level like we previously mentioned in section 5.1.1. Simultaneously the timer `htim3` with a $1\,\mu$s period is started to await the conversion time and call again the `HAL_TIM_PeriodElapsedCallback` function. As soon as this step is done the `TIM_Callback` function in the ADC class is executed which stops the timer and sets the conversion pin again low. The data transfer is then started by calling the `HAL_SPI_TransmitRecieve_DMA` function which automatically handles the SPI communication. This function uses the direct memory access (DMA) feature which can write the values received from the SPI directly into the RAM without distracting the CPU, which normally would do this function and be blocked from doing other tasks. The use of DMA to receive and send data frees up the CPU and allows us to achieve high bandwidths of the device. When the DMA-controller is finished the `HAL_SPI_RxCpltCallback`

```
1  // SPI reading callback
2  void HAL_SPI_RxCpltCallback (SPI_HandleTypeDef * hspi)
3  {
4      // Callback for ADC
5      if(hspi->Instance == SPI_ADC) {
6          ADC_value_ready = true;
7      }
8  }
9
10 void main(void)
11 {
12     while(1) {
13         if(ADC_value_ready){
14             ADC_value_ready = false;
15             // Save measured data
16             if(reading_finished){
17                 Pi_communication_ready = true;
18             }
19         }
20
21         if(Pi_communication_ready){
22             Pi_communication_ready = false;
23             // Communicate with the Raspberry Pi
24         }
25     }
26
27 }
```

Figure 5.5: Code example of the main loop to illustrate how flags are used to control the flow of the program. The SPI callback sets the `ADC_value_ready` flag to true. In the main loop the data gets saved and if the measurement is done the Pi communication flag is set to enable the transmission of the data.

and subsequently `SPI_Callback` in the ADC class is run. There the measured value is then stored in a member value of the ADC class where it can be accessed by other parts of the code. Finally the device is again marked as ready and is set for the next sample. This example shows how the microcontroller uses interrupts and callbacks to communicate with the peripheral devices and does so in a fast and concise manner.

There are also other important files and functions besides the `adc.cpp` file which run on the microcontroller. A non-exhaustive list of important user files in the project is found in table 5.2. The file `dac.cpp` contains the `DAC_Dev` class and works very similar to the ADC class in its structure. The `lock.cpp` and `pid.cpp` are responsible for the frequency lock and the PID as their

File	Important Content
`adc.cpp`	ADC_Dev and ADC_Channel classes, which handle low level communication with the LTC2353
`cppmain.cpp`	Main-loop of the program, Callbacks, Global variables and some initialisation
`dac.cpp`	DAC_Dev class, which handle low level communication with the AD5791
`dmacpp.cpp`	DMAChannel class, which handles the DMA stream for the ADC and DAC
`lock.cpp`	lockparameters class, which contains the code for the frequency lock
`main.c`	Initialisation of clocks, GPIO, DMA and SPI
`mxconstants.h`	User-defined macros
`pid.cpp`	PID class, which saves the PID parameters
`spi.c`	Functions for communication with Raspberry Pi
`stm32f4xx_hal_msp.c`	CubeMX generated initialisation of Pins
`stm32f4xx_it.c`	CubeMX generated interrupt handlers

Table 5.2: Summary of the most important files in the project and their function.

name suggests. The `main.c` file calls the file `cppmain.cpp` so that we can program in the C++ language in our main-loop. The `mxconstants.h` file contains the user defined macros which make other parts of the code more readable. The other files are mostly initialisation of peripherals and pins which should not change as long as the hardware stays the same.

```cpp
// Timers:
extern TIM_HandleTypeDef htim2; // Sampling Clock
extern TIM_HandleTypeDef htim3; // Timeout Clock ADC

void SetSamplingFrequency(uint32_t frequency)
{
    HAL_TIM_Base_Stop_IT(&htim2);
    htim2.Init.Period = 90000000/frequency;
    HAL_TIM_Base_Init(&htim2);
    HAL_TIM_Base_Start_IT(&htim2);
}

// This function is called whenever a timer reaches its period
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    // Sampling Clock
    if (htim->Instance==TIM2){ ADC_DEV->Read(); }

    // Timeout Clock for the Conversion of the ADC
    if (htim->Instance==TIM3){ ADC_DEV->TIM_Callback(); }
}

// SPI reading callback
void HAL_SPI_RxCpltCallback (SPI_HandleTypeDef * hspi)
{
    // Callback for ADC
    if(hspi->Instance == SPI_ADC) { ADC_DEV->SPI_Callback(); }
}
```

(a) Code from `cppmain.cpp`

```cpp
void ADC_Dev::Read()
{
    // Check if Device is busy
    if(!this.ready)
        return;
    this.ready = false;

    // Start conversion by pushing CNV high
    HAL_GPIO_WritePin(CNV_Port, CNV_Pin, GPIO_PIN_SET);
    // Wait 1us for conversion
    __HAL_TIM_ENABLE(&htim3);
}

void ADC_Dev::TIM_Callback()
{
    // Stop Timer
    __HAL_TIM_DISABLE(&htim3);
    // End Conversion and start Acquisition
    HAL_GPIO_WritePin(CNV_Port, CNV_Pin, GPIO_PIN_RESET);
    // Receive data via SPI
    HAL_SPI_TransmitReceive_DMA(hspi, Softspan, buffer, 6);
}

void ADC_Dev::SPI_Callback()
{
    // Store values from SPI in member variable

    // Mark the Device as ready
    this.ready = true;
}
```

(b) Code from `adc.cpp`

Figure 5.6: Code examples from the `cppmain.cpp` and `adc.cpp` files to illustrate the procedure to take a measurement sample with the ADC.

5.2 Layout

An important part in keeping the signals noise free is the placement of the components in the layout. The main idea is to separate high frequency digital signals from the analog signals, voltage references and power supplies. Separating analog and digital wires reduces the noise in the analog signals and gives us better measurements. Stable voltage references and power supplies ensure that the IC's operate properly and don't introduce additional errors. Another aspect of reducing power supply noise is the use of decoupling capacitors [19] which will be discussed in chapter 5.2.3. I will give here a brief overview of the design principles one has to pay attention to if designing high frequency mixed-signal devices.

5.2.1 Ground Planes

If possible one should use an entire layer of the printed circuit board (PCB) for the ground plane because this produces multiple positive effects on the signal quality. For once the large plane has a low impedance connection for the return currents which removes parasitic resistance and inductance for the high frequency signals. The constant impedance of the ground plane reduces the radiation noise of digital signals and improves their signal quality. Also the plane itself shields other sensitive signals such as the analog inputs or the voltage references from radiation [20].

The ground plane should be physically divided into two grounds, the analog ground (AGND) and digital ground (DGND). This splits the board into two separate regions. The micro controller and the SPI wires are placed in the digital part, whereas the power supplies and Op-amps are placed in the analog part. The DACs and ADCs which are mixed-signal devices and contain both digital and analog parts should be placed on the border between the two planes. This placement and the planes can be seen in figure 5.7. Although many devices on the board have both AGND and DGND pins most of them are connected simply to the analog ground as explained in [20]. Only the microcontroller has pins which are directly connected to the digital ground. This is because the ADCs and DACs are very sensitive to the digital noise and their grounds should therefore not be mixed with one from the microcontroller.

The two planes then continue onto the backplane and are there connected at a single point called the star-point. This prevents interaction of the return currents between the analog and digital circuits. The reason for this is that fast digital signals often draw large currents for only short times which would result in a voltage drop in analog signals and introduce noise.

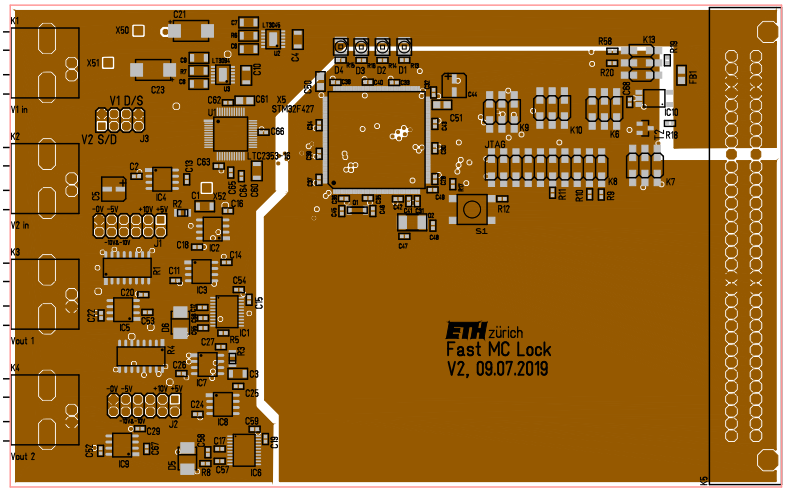


Figure 5.7: Picture of the separated digital and analog ground planes. The two ground planes are then connected at a single point in the backplane to prevent the noise from the digital ground to couple into the analog signal.

5.2.2 Layer stack and wire division

For the PCB we use a 4-layer board because one of layers is already used for the ground plane and with a lot of IC's and the microcontroller on the board the wires do have to cross occasionally. To have at least 2 layers for the signals makes the routing easier which then results in better performance. The last of the 4 layers is used for the supply lines which should also be separated from the signals. Like for the ground planes, the digital and analog wires should be separated as well as possible. The distribution of the wires on the PCB can be seen in figure 5.8. The blue and red wires are the top and bottom copper layers. The green supply wires and the grounds (not visible) are the two inside copper layers of the 4-layer board. The analog and digital wires are kept in their part of the board respectively and if they need to cross another signal they do so in a perpendicular angle as to reduce the coupling between the wires. This is because wires which direct fast signals act as antenna and parallel lines can couple their signals to each other. The green supply wires are placed on the layer below the ground plane in order to shield them from the signal wires on the top layer as good as possible.

5.2.3 Decoupling

Decoupling is used to prevent coupling between two different subsystems in an electronic circuit. This is achieved by the use of decoupling capacitors which are placed between the supply and the ground. The capacitors act as

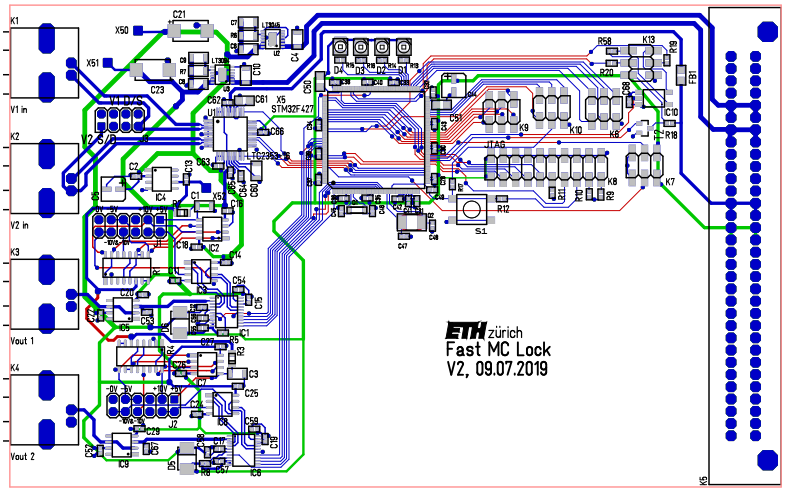


Figure 5.8: Drawing of the many wires on the 4-layer PCB. The blue top and red bottom copper layer is used for different analog and digital signals. The green middle layer is used for the power supply.

a reservoir and supply the circuit with current during high demand periods to prevent the drop of the supply voltage. This prevents noise or jitter in the supply pins of the IC's which can lead to performance degradation[19].

A real capacitor however not only consists of a capacitance but also includes a equivalent series resistance (ESR) and a equivalent series inductance (ESL). If one measures the impedance of a capacitor as a function of frequency, the impedance will gradually decrease like in figure 5.9. We see however that the impedance flattens out at a certain point which is the effect of the ESR and then begins to rise with increasing frequency because of the ESL. This effect happens at different frequencies depending on the capacitance or ESL respectively.

Therefore multiple decoupling capacitors should be used with varying capacitances. The most common variant is to use for each pin a small 100 nF ceramic capacitor and then a larger electrolyte capacitor of a few μF for each IC [19]. The small ceramic capacitors have a small ESL and thus have a good response even at high frequencies. They are placed directly at the pin and then connected to the ground with a pin hole in order to reduce the parasitic resistance and inductance. This can be seen in a drawing of the PCB in figure 5.10. The big electrolyte capacitor has a worse frequency response but acts as reservoir of supply charge for the IC if more power is needed. Together these two types of capacitances protect the IC from both high frequency noise as well as big changes in the load current.

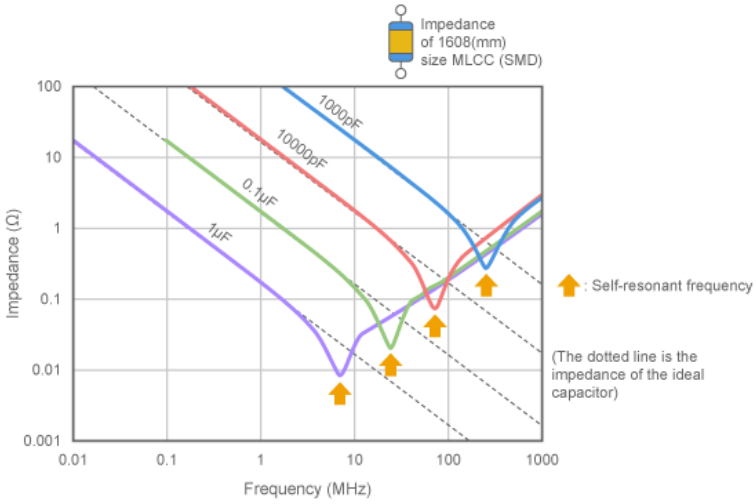


Figure 5.9: Schematic plot of impedance of the decoupling capacitors as a function of frequency, picture from [21].

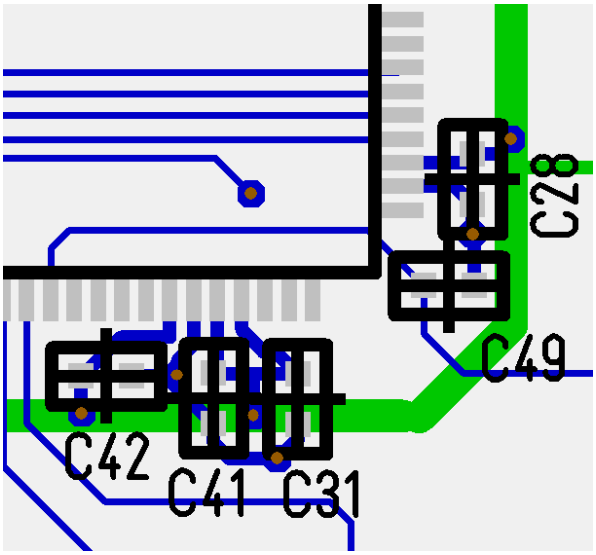


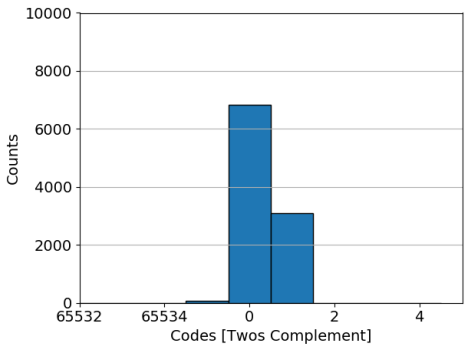
Figure 5.10: Drawing of the PCB with the microcontroller and decoupling capacitors. The capacitors are placed as close as possible to the package and the ground is connected through a pin hole to the ground plane.

5.3 Characterisation

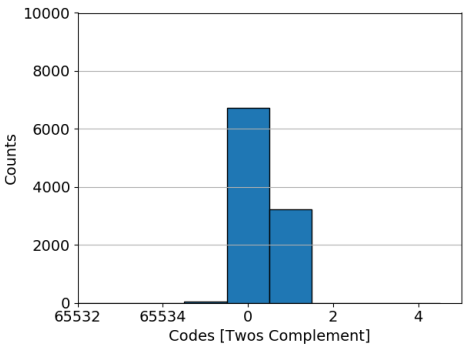
In this section I will present some results and test measurements of the previously mentioned components. The components on the board were tested and then compared to the properties mentioned in the data sheets to ensure that the parts were functioning and were built in properly. The first test was the measurement of ADC codes to see how much noise is in the input signal and how stable the ADC works. Then the output from the DAC was measured on an oscilloscope to see how well the output signal behaves. And as a last part both the clock and signal lines of the SPI connection between ADC/DAC and the microcontroller were examined to confirm that the correct data gets sent.

5.3.1 ADC Codes

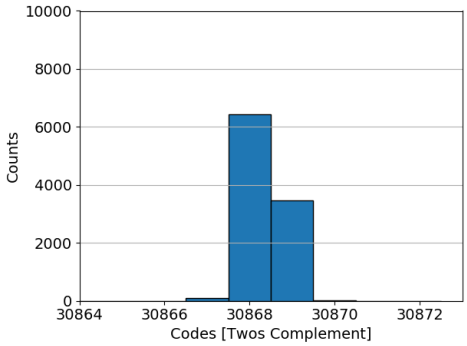
Important measures on how well the ADC performs are the distributions of codes the ADC records when given a constant voltage signal. The distribution of the codes should then be within the margin of error. If however the codes are further distributed then there is additional noise in the input signal. To measure the ADC codes we connected the input either to the ground or to a 9 V battery to get a voltage source that is as constant as possible. The different integer values the ADC produces are then sent to the microcontroller and then later plotted. The distribution of the measured codes can be seen in figure 5.11. We see that all the measurements in the single ended mode don't seem to differ from the actual value by more than $1\,LSB$. This is in good agreement with what we expected from the ± 1 integrated non linearity (INL) error described in the previous sections. Also the 0 V differential signal falls within that error since it is obviously the same measurement as the single ended one. For the ± 9 V differential distributions however there seems to be additional noise which spreads the distributions further out. This suggests to confirm our suspicion we had during the development, that the differential signals seem to be more noisy and give worse results.



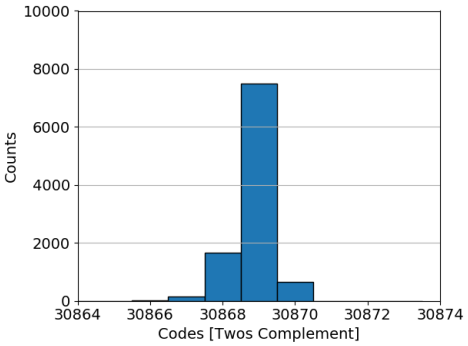
(a) Codes at 0 V in Single ended mode.



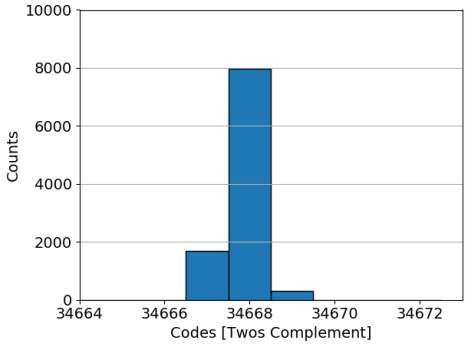
(b) Codes at 0 V in Differential mode.



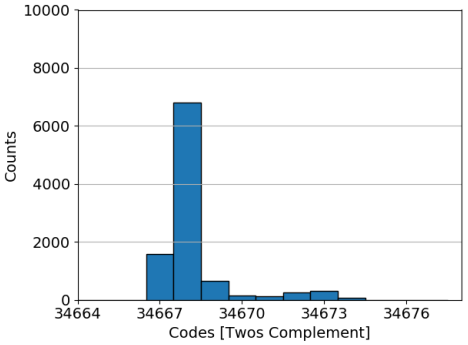
(c) Codes at 9 V in Single ended mode.



(d) Codes at 9 V in Differential mode.



(e) Codes at −9 V in Single ended mode.



(f) Codes at −9 V in Differential mode.

Figure 5.11: Plots of Code distributions of the ADC at different voltages and for differential or single ended inputs.

5.3.2 DAC Step Response

When the DAC changes its output the voltage jumps to the newly programmed value and then usually oscillates before settling to the output volt-

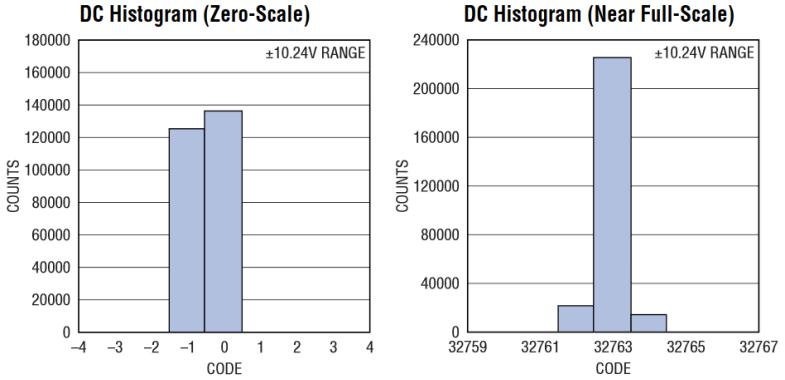
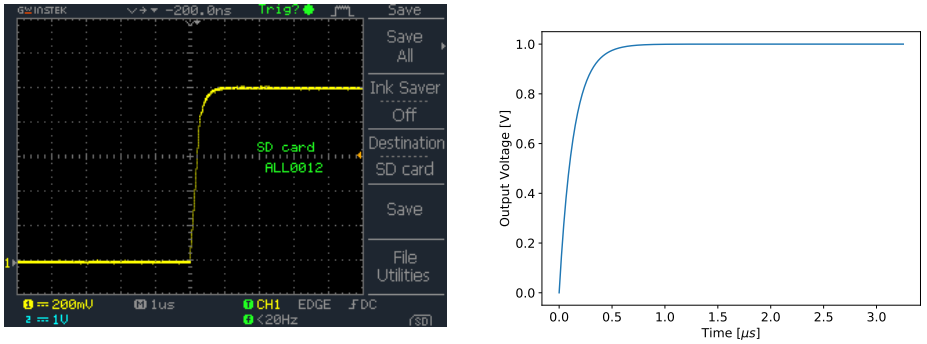


Figure 5.12: ADC codes from the data sheet of the LTC2353 [12] as a comparison to out vales in figure 5.11.

age. To remove this ringing we included a 1.17 MHz low-pass filter after the DAC to remove the fast oscillations of the output. This low-pass filtering can be seen in figure 5.13 where the step function is measured and compared to a response function of a low-pass filter. We see that the measured response behaves almost exactly like the theoretical prediction and we therefore can assume that our circuit acts as a good filter for the output signal. In addition we see that there is no ringing or oscillating in the signal and the signal reaches the set level within 1 μs, which is the settling time of our DAC.



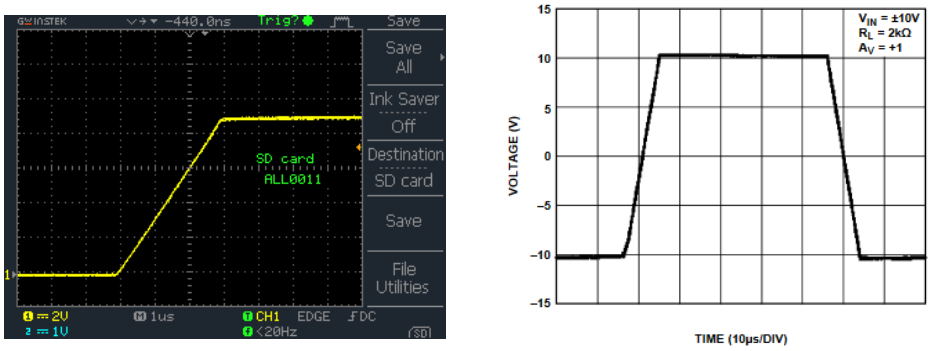
(a) Small step response measured on oscilloscope.

(b) Theoretical Step response.

Figure 5.13: Step response of the DAC after a jump from 0V to 1V in the output voltage, the signal is low-pass filtered with a cutoff frequency of about 1.17 MHz.

On the other hand, for a large step in the output voltage the picture looks quite different. We see in figure 5.14 that the signal does not show the usual asymptotic curve we would expect. Also the time-scale for the step has

increased to about 3 μs from the original 1μs settling time. This effect can however be explained when we remember that after the DAC an Op-amp with unity amplification is placed in order to remove the output impedance of the the device. For larger steps, the time scale is no longer given by the low-pass filter but by the finite slew-rate of the Op-amp [18]. The slew-rate of the Op-amp is estimated in the data sheet by 2.5 V/μs which means that our 9 V jump in figure 5.14 takes the Op-amp about 4 μs to adjust. This effect explains the different response of the DAC and therefore our device works properly even for large jumps in output voltages but we have to keep in mind that the settling time might be larger for such signals.



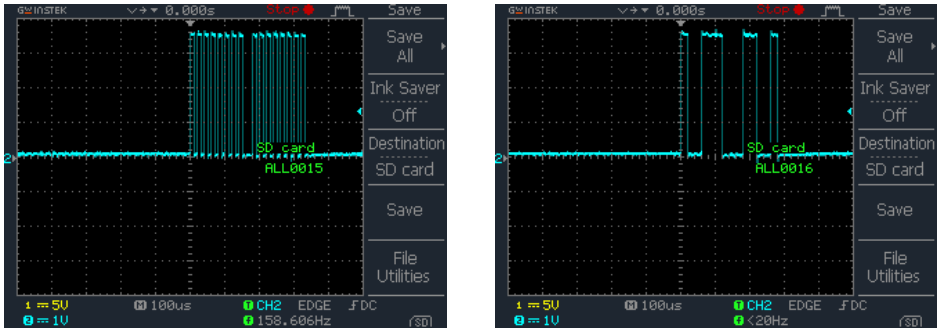
(a) Large step response measured on oscilloscope.

(b) Large Signal response out of the data sheet of the AD8675 Op-amp [18].

Figure 5.14: Step response of the DAC after a large jump in the output voltage, the settling time is longer and stems mostly from the slew-rate of the OP-amp.

5.3.3 SPI Signals

Another important part for the correct operation of the Lock-box is a proper SPI connection between the components. In figure 5.15 we can see both a clock and signal connection between the ADC and the micro controller. Important for SPI transmissions is that the signals don't show any oscillations or other deviations from the step-like profile. If such a thing were to occur, wrong data would be transmitted or even worse, the device could miss a clock step and the whole SPI transmission would fail. In our case however the signals look clean and noiseless which we owe to the fact that the ground plane gives a very good constant impedance.



(a) SPI Clock signal of a 16-bit transmission.

(b) SPI MOSI or MISO line during a 16-bit transmission.

Figure 5.15: SPI Clock and Signal between the ADC and the micro controller recorded on an Oscilloscope.

Chapter 6

Conclusions and Outlook

We have developed and characterized a microcontroller-based feedback controller, that digitizes analog inputs, digitally processes the input in real-time, and gives analog feedback for a variety of different applications. In the first part of my thesis I tested this on the previous version of the device and made programs to check the proof of concept. I implemented the automatic optimization of PID parameters, the linearisation of an AOM response function, and the frequency lock and re-lock of an optical cavity. While the Lock-box and the programs worked, some problems and difficulties were uncovered which then lead to the development of a second device.

The input and output voltage ranges of the new device are configurable to suit different experimental requirements. The high input impedance $> 1\,\mathrm{G}\Omega$ and negligible output impedance make it easy to interface the device with other lab devices. The 16-bit input and 20-bit output resolution should make this device a one-fits-all solution for amplitude and frequency stabilization. Closed-loop feedback operation can be achieved at a bandwidth of 80 kHz.

The possible applications and useful hardware specifications make this smart PID controller a promising challenger for the much used LB1005-S and QO-EL-0021 devices. The next steps in the development are

- to make the device convenient to use by embedding it into a standard rack with power supply,
- to finish the graphical user interface that allows easy access to all features and configurations of the device,
- to test the hardware and algorithms in a wide range of situations,
- to improve the low-level programming to further optimize the bandwidth,
- and to implement the frequency-dependent feedback, see Appendix B.

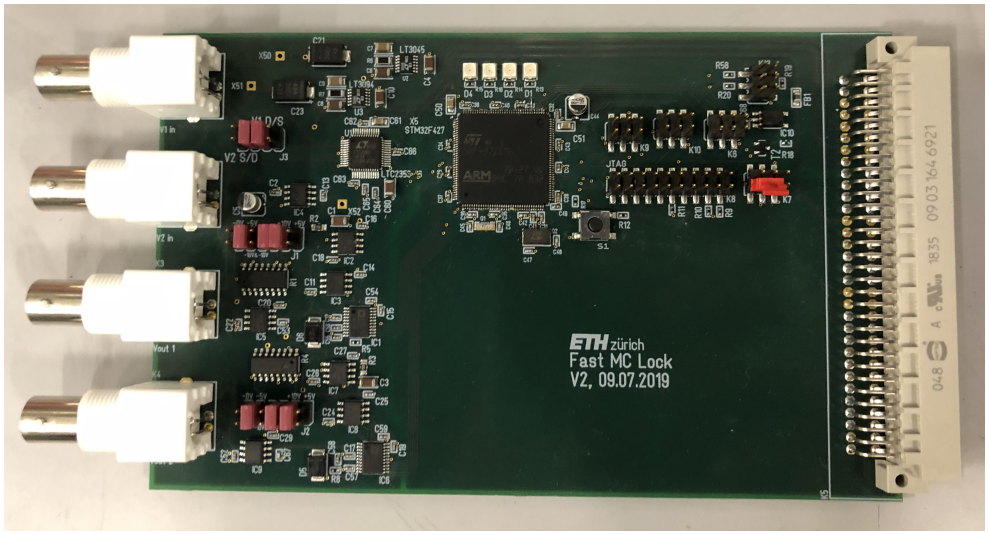


Figure 6.1: Picture of the finished PCB with all of the components soldered on top of the board.

Appendix A

Lowpass Filtering of a linear Ramp

We calculate here the low-pass filtered response of a linear ramp with a stop at time τ. The function of the ramp is then given by

$$U_{\text{in}}(t) = \begin{cases} 0 & \text{for } t < 0 \\ \frac{U_0}{\tau} t & \text{for } 0 \leq t \leq \tau \\ U_0 & \text{for } \tau < t \end{cases}. \tag{A.1}$$

We can then write the outgoing (filtered) voltage as

$$U_{\text{out}} = \frac{Q_C}{C}, \quad \dot{Q}_c = \frac{U_{\text{in}} - U_{\text{out}}}{R} = \frac{U_{\text{in}}}{R} - \frac{Q_c}{RC} \tag{A.2}$$

and the change of capacitor charge is given by the current through the resistor. We can solve this differential equation in Q_c by considering first the homogeneous differential equation

$$\dot{Q}_c = -\frac{1}{RC} Q_c \quad \Longrightarrow \quad Q_c = Q_0 e^{-t/(RC)}. \tag{A.3}$$

The full solution can then be found by the variation of the constant method.

$$\dot{Q}_c = \dot{Q}_0 e^{-t/(RC)} - \frac{Q_0}{RC} e^{-t/(RC)} = \frac{U_{\text{in}}}{R} - \frac{Q_0}{RC} e^{-t/(RC)} \tag{A.4}$$

$$\Longrightarrow \dot{Q}_0 e^{-t/(RC)} = \frac{U_{\text{in}}}{R} \tag{A.5}$$

The constant is then found by integration of $\dot{Q}_c$.

$$Q_0 = \int_{-\infty}^{t} \frac{U_{\text{in}}(t)}{R} e^{t/(RC)} dt = \underbrace{\frac{U_0}{R} \int_0^{\min(t,\tau)} \frac{1}{\tau} t e^{t/(RC)} dt}_{\text{if } t>0} + \underbrace{\frac{U_0}{R} \int_\tau^t t e^{t/(RC)} dt}_{\text{if } t>\tau} \tag{A.6}$$

We use the following two integrals to calculate the value of Q_0.

$$\int_0^t te^{t/(RC)}dt = (RC)te^{t/(RC)}\Big|_0^t - \int_0^t (RC)e^{t/(RC)}dt \tag{A.7}$$

$$= (RC)te^{t/(RC)} - (RC)^2\left(e^{t/(RC)} - 1\right) \tag{A.8}$$

$$\int_\tau^t e^{t/(RC)}dt = (RC)\left(e^{t/(RC)} - e^{\tau/(RC)}\right) \tag{A.9}$$

Combining the result of equation A.6 with the homogeneous solution in A.3 results in the solution for the filtered output voltage

$$U_{\text{out}}(t) = \begin{cases} 0 & \text{for } t < 0 \\ \frac{U_0}{\tau}\left(t - RC\left(1 - e^{-t/(RC)}\right)\right) & \text{for } 0 \leq t \leq \tau \\ U_0 - \frac{U_0}{\tau}(RC)\left(1 - e^{-\tau/(RC)}\right)e^{-(t-\tau)/(RC)} & \text{for } \tau < t \end{cases}. \tag{A.10}$$

For $t < \tau$ the filtered voltage is obviously zero like the input ramp. For times $0 \leq t \leq \tau$ when the the voltage is inside the ramp the filtered voltage has an exponential transition into a ramp that is delayed by RC to the unfiltered ramp. At time τ when the signal stops at the top of the ramp, the filtered voltage is lowered by $\frac{U_0}{\tau}(RC)\left(1 - e^{-\tau/(RC)}\right)$ compared to the unfiltered voltage and then exponentially approaches the value U_0 with increasing t.

Appendix B

Frequency dependent feedback

Although we have not implemented it, I will here present the basic idea of the frequency and how we want to realize it in the future. If we want to give frequency dependent feedback we need to be able to filter out unwanted frequencies out of our output in real time. The difficulty here lies in the fact that calculating the Fourier transform of our feedback at every sampling step would take too long even with a fast Fourier transform (FFT) algorithm when giving real time feedback. The idea with the running Fourier series is to reuse previous Fourier components to calculate the next ones and decrease the computational effort. We are given the data set $\{x_n\}$ and we want to calculate the Fourier series in the window from $n = m$ until $n = m + N - 1$ where N is the size of our running window. Then the k^{th} Fourier component is given by

$$X_k^{(m)} = \sum_{n=0}^{N-1} x_{n+m} \cdot e^{-i2\pi kn/N}. \tag{B.1}$$

If we now measure an additional value and want to shift our window to $(m+1)\dots(M+N)$ then we can rewrite the k^{th} Fourier component in the following manner.

$$\begin{aligned} X_k^{(m+1)} &= \sum_{n=0}^{N-1} x_{n+m+1} \cdot e^{-i2\pi kn/N} && \text{(B.2)} \\ &= \sum_{n=1}^{N} x_{n+m} \cdot e^{-i2\pi k(n-1)/N} && \text{(B.3)} \\ &= e^{i2\pi k/N} \sum_{n=1}^{N} x_{n+m} \cdot e^{-i2\pi kn/N} && \text{(B.4)} \\ &= e^{i2\pi k/N} \left(\sum_{n=0}^{N-1} x_{n+m} \cdot e^{-i2\pi kn/N} - x_m + x_{n+m} \right) && \text{(B.5)} \end{aligned}$$

$$= e^{i2\pi k/N}\left(X_k^{(m)} - x_m + x_{n+m}\right) \tag{B.6}$$

So we see that we do not have to recalculate the entire Fourier spectrum but we can update it by storing the data set window and replacing the oldest data point with the new one and updating the Fourier component accordingly. This procedure should speed up the calculation and allow for the filtering of the real time feedback at the expense of some memory space.

With the Fourier components we can then adjust our output in such a manner to reduce certain problematic frequencies in the feedback. This could be mechanical resonances in the cavity which we want to avoid when driving the system to reduce noise.

Bibliography

[1] J. Léonard, *A Supersolid Of Matter And Light*. PhD thesis, ETH Zürich, 2017.

[2] A. Morales, *Intertwined Order with Ultracold Bosons in Optical Cavities*. PhD thesis, ETH Zurich, 2018.

[3] K. Baumann, C. Guerlin, F. Brennecke, and T. Esslinger, "Dicke quantum phase transition with a superfluid gas in an optical cavity," *Nature*, vol. 464, pp. 1301–1306, apr 2010.

[4] A. Morales, P. Zupancic, J. Léonard, T. Esslinger, and T. Donner, "Coupling two order parameters in a quantum gas," *Nature Materials*, vol. 17, pp. 686–690, July 2018.

[5] J. Léonard, A. Morales, P. Zupancic, T. Esslinger, and T. Donner, "Supersolid formation in a quantum gas breaking a continuous translational symmetry," *Nature*, vol. 543, pp. 87 – 90, Mar 2017.

[6] J. Léonard, A. Morales, P. Zupancic, T. Donner, and T. Esslinger, "Monitoring and manipulating higgs and goldstone modes in a supersolid quantum gas," *Science*, vol. 358, no. 6369, pp. 1415–1418, 2017.

[7] Wikipedia, "Pid controller." https://en.wikipedia.org/wiki/PID_controller, 2019. [Online; accessed 18-September-2019].

[8] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[9] R. W. P. Drever, J. L. Hall, F. V. Kowalski, J. Hough, G. M. Ford, A. J. Munley, and H. Ward, "Laser phase and frequency stabilization using an optical resonator," *Applied Physics B*, vol. 31, pp. 97–105, Jun 1983.

[10] E. D. Black, "An introduction to pound–drever–hall laser frequency stabilization," *American Journal of Physics*, vol. 69, no. 1, pp. 79–87, 2001.

[11] Wikipedia, "Transverse mode." https://en.wikipedia.org/wiki/Transverse_mode, 2019. [Online; accessed 18-September-2019].

[12] ANALOG DEVICES, INC., "Data sheet , LTC2353-16." https://www.analog.com/media/en/technical-documentation/data-sheets/235316f.pdf, 2018. [Online; accessed 18-September-2019].

[13] ANALOG DEVICES, INC., "Circuit note , cn-0191." https://www.analog.com/media/en/reference-design-documentation/reference-designs/CN0191.pdf, 2011. [Online; accessed 18-September-2019].

[14] Maxim Integrated Products, Inc. , "ADC and DAC Glossary." https://www.maximintegrated.com/en/design/technical-documents/tutorials/6/641.html, 2002. [Online; accessed 18-September-2019].

[15] ANALOG DEVICES, INC., "Data sheet , AD5791." https://www.analog.com/media/en/technical-documentation/data-sheets/ad5791.pdf, 2018. [Online; accessed 18-September-2019].

[16] ANALOG DEVICES, INC., "Data sheet , ADR445." https://www.analog.com/en/products/adr445.html?doc=ADR440_441_443_444_445.pdf, 2018. [Online; accessed 18-September-2019].

[17] Vishay, "Data sheet , NOMCA16031001AT5 ." https://www.vishay.com/docs/60117/nomca.pdf, 2012. [Online; accessed 18-September-2019].

[18] ANALOG DEVICES, INC., "Data sheet , AD8675." https://www.analog.com/media/en/technical-documentation/data-sheets/AD8675.pdf, 2012. [Online; accessed 18-September-2019].

[19] ANALOG DEVICES, INC., "Decoupling techniques." https://www.analog.com/media/en/training-seminars/tutorials/MT-101.pdf, 2009. [Online; accessed 18-September-2019].

[20] ANALOG DEVICES, INC., "Grounding data converters and solving the mystery of AGND and DGND." https://www.analog.com/media/en/training-seminars/tutorials/MT-031.pdf, 2008. [Online; accessed 18-September-2019].

[21] Murata Manufacturing Co., Ltd., "Noise Suppression Products/EMI Suppression Filters." https://www.murata.com/en-global/products/emc/emifil/knowhow/basic/chapter06-p8, 2019. [Online; accessed 18-September-2019].

Acknowledgements

During my work on this project I received a lot of helpful support from numerous people to whom I would like to express here my gratitude.

I would like to thank my supervisor Philip Zupancic, whom I truly enjoyed working with, for all his contributions and his help with answering questions about the project. To Alexander Frank and Xiangliang Li, I want to thank for all their help and advice with the electronics surrounding the experiment. I also thank Alexander Baumgärtner and Davide Dreon from the IMPACT team for their willingness to discuss the project with me. For the opportunity to do my master thesis I would like to extend my thanks to Prof. Tilman Esslinger and Tobias Donner. And to all the rest of the QO-members I want to thank for the enjoyable atmosphere during my stay in the group.

Lastly, I want to thank all my friends and family for all their support.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.