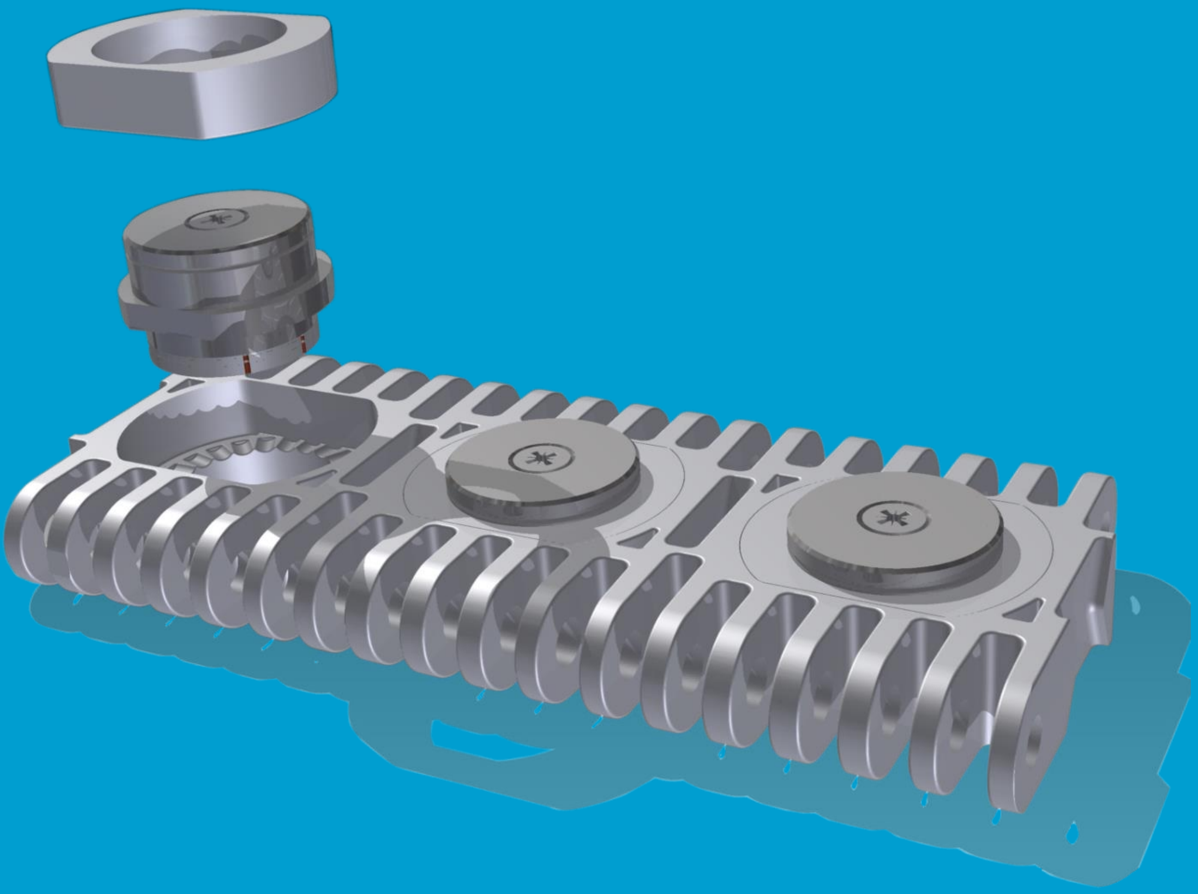
Load cell resonance frequency analyser

Frequency analysis system for load cell based conveyor belt weighing systems

Bachelor Graduation Thesis – TUDelft Electrical Engineering
19 June 2020

Jonathan Dijkstra (4696778) & Mauries van Heteren (4712145)



Abstract

In this thesis, the design of a frequency analysis system for conveyor belt weighing systems is designed. The objective of the thesis is to detect a change in resonance frequency, caused by a change of pressure applied to a sensor. The design features a resolution of 10 000 discrete frequency levels in a bandwidth of 0.5 MHz. Noise division of a pseudo random noise (PRN) signal is used to obtain a clear frequency spectrum. An ADC is implemented to digitize an analog input signal. Furthermore, a digital down converter using a CIC filter and CORDIC based controlled digital oscillator is applied. The frequency-domain translation is achieved using an FFT. Lastly, an interpolation over the spectrum is realized to increase the frequency resolution. Additionally, an analytical analysis of the primary noise sources in the system is given. The entire system is simulated in MATLAB to validate the designed blocks.

Preface

This bachelor thesis is written as part of the Bachelor Graduation Project, which is the final part of the bachelor program Electrical Engineering at the TU Delft. This thesis is part of a project initiated by Intralox: ”*A passive wireless load cell for in-situ product weight and position sensing on modular plastic conveyor belts*” [11]. Together with four other enthusiastic group members, we accepted the challenge.

The time span of the project was two months. We delved deep into the world of digital frequency analysis systems. The project was a great opportunity for us to learn new things and to develop skills in structured problem solving. We could apply the knowledge obtained in the bachelor, as well as get more in-depth insight into the used theory.

We could not have done this project without the help of our supervisor dr. ir. Chris Verhoeven. We would like to express our gratitude to Chris for his valuable guidance and advice during the project. Furthermore, we are very grateful for the assistance Intralox provided us with. We would like to thank Sven-Erik Haitjema and Lazlo Kleczewski in particular. They provided us with important and relevant knowledge about the system and delivered us tools and guidance for the project.
Also, we would like to thank dr. Jaap Hoekstra for being in the jury during our green-light assessment. He gave us with useful feedback about our designed system.
Finally, we like to thank our fantastic colleagues Rik Bokhorst, Ruben van den Bos, Jasper Insinger and Rogier Fischer for helping us with their skills and knowledge. Our productive collaboration and their assistance were of great importance for the final result of the project.

Jonathan Dijkstra & Mauries van Heteren
Delft, June 2020

Contents

1

Introduction

Problem definition

Product weighing for conveyor belt systems is a technology with applications in the aviation, automotive, logistics and food industry. With increasing industrialisation and automation of the industries, the demand for high throughput, accurate weighing system has emerged. Conventional weighing systems require weighing scales which require extra spacing in between products on the belt, lowering the throughput of the conveyor system. A system using the conveyor belt to determine the weight of a product on the belt (see Figure 1.1) has advantages over conventional weighing systems. It does not require extra spacing between products since the belt itself is used to determine the packet weight. Furthermore, packets can be weighted side by side on the belt, enabling higher product throughput of the conveyor belt system. The benefits of in-situ weighing extend to the possibility of acquiring feedback of the relative position of the product on the conveyor belt. Furthermore, it provides the possibility to determine physical properties other than weight such as the centre of gravity and orientation of the product.

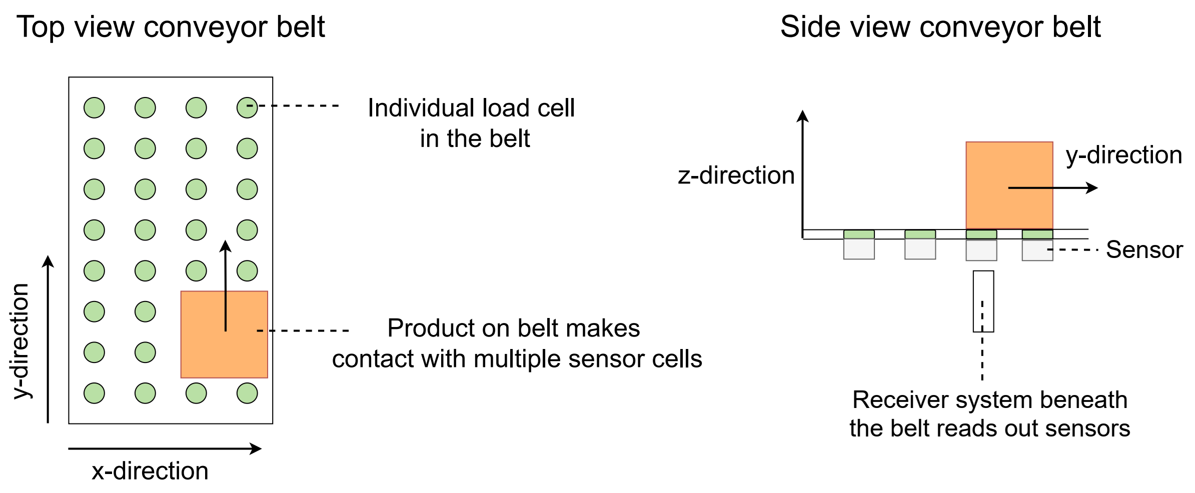

Top view conveyor belt
Individual load cell in the belt
Product on belt makes contact with multiple sensor cells
y-direction
x-direction
Side view conveyor belt
z-direction
y-direction
Sensor
Receiver system beneath the belt reads out sensors

Figure 1.1: An overview of the weighing system.

Such a weighing system would consist of a load cell placed in the belt, and a receiver system placed beneath the belt. The load cell and receiver system are guidelines provided by Intralox for suitable implementation of the weighing system into the normal Intralox conveyor belt standards. This system has been decomposed into three parts (see Figure 1.2). Each part corresponds to a subgroup for the project. Their tasks can be described as:

- **Sensor group:** Has the task to design a load cell which translates mechanical deformation into a change in electrical properties of a sensor circuit [2].
- **Reader group:** Has the task to design a reader circuit which reads the change in electrical properties of the sensor circuit and translates this change into a signal [7].

- **Analysis group:** Has the task to interpret the signal and detect mechanical deformation of the load cells. This thesis focuses on the development of the analyser subsystem.

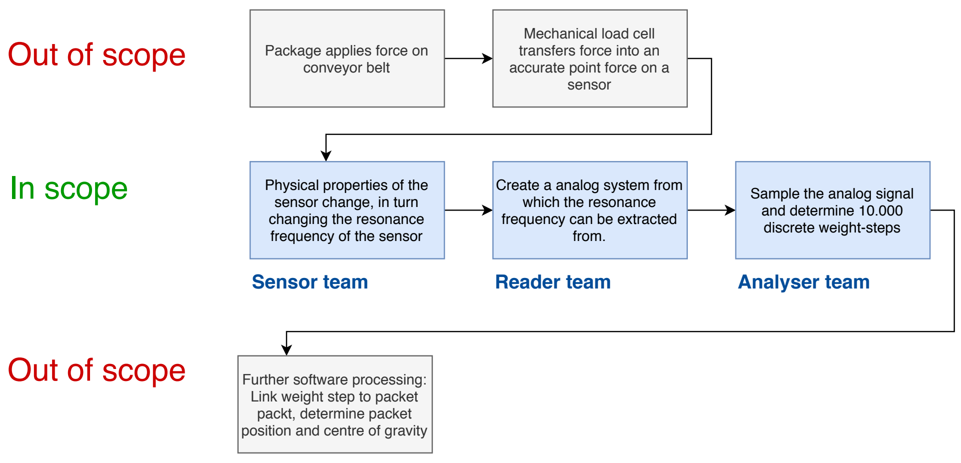

Out of scope
Package applies force on conveyor belt
Mechanical load cell transfers force into an accurate point force on a sensor
In scope
Physical properties of the sensor change, in turn changing the resonance frequency of the sensor
Create a analog system from which the resonance frequency can be extracted from.
Sample the analog signal and determine 10.000 discrete weight-steps
Sensor team
Reader team
Analyser team
Out of scope
Further software processing: Link weight step to packet packt, determine packet position and centre of gravity

Figure 1.2: An overview of the project scope

Goal of the analyser

The project objective of the analyser is to design a signal processing system that uses the output signal of the reader group in order to detect the resonance frequency of the sensor and translate that to 1 of 10 000 levels of deformation of the sensor.

State-of-art-analysis

The analysis of signals in frequency domain is a mature field. Many techniques to manipulate signals can be identified and explored. These techniques are discussed in this section.

The first technique is the analog to digital conversion. When a signal is digitized, it is usually easier to manipulate. There are lots of different architectures of ADCs. From the literature, it was found that four basic architectures are used most in practice [13]. The first is successive approximation ADC, which is used in data acquisition systems and industrial applications. The next type is the sigma-delta architecture, which is used in applications where a resolution from 12 to around 24 bits is needed. They are used, for example, in signal conditioning applications [9]. Pipeline ADCs are used in many types of instrumentation, including spectrum analyzers, and medical imaging. Finally, flash ADC architecture is used for very high-frequency applications.

Another technique is the conversion of a signal to frequency-domain using a Fourier transform. This is used to analyze the frequency spectrum of a signal. An example of this is a spectrum analyzer.

In order to perform frequency analysis, a fast Fourier transform can be used for translation to frequency domain. When a greater frequency resolution is desired, spectrum interpolation can be applied. In this method, different bin frequency amplitudes are interpolated in the range of the desired frequency. Parabolic and Gaussian interpolation can be used [6]. Another technique is windowing. Using windowing the discontinuities at the edges of the time window are ignored, resulting in lower side lopes in the Fourier Transform of the sensed data [3].

Filters are used frequently in digital signal processing. A decimation filter, for instance, can be used to lower the sampling frequency of a signal for more efficient processing.

Some of these techniques are applied in a Software Defined Radio (SDR). This is a radio-communication system that uses hardware for the implementation of its components. The real-world application of SDR is only possible since the fast-developing possibilities of digital electronics [1]. The digital processing techniques that are usually used in SDR are, for example, an ADC, decimation filter and FIR filters.

Furthermore, spectrum analysers are existing technologies for providing insight in the frequency content of a signal. Their the resolution of such an analyser depends on the speed of the analyser. Swept tune spectrum analysers use superheterodyne receivers and band pass filters. The centre frequency of the receiver is swept to obtain a frequency spectrum. FFT based spectrum analysers sample the input and perform a FFT over the input. The centre frequency is reduced to reduce the sampling frequency of the FFT.
Frequency counters are another class of instruments used for measuring the frequency of a signal. It counts the number of oscillations in the period of time. Accurate time bases are required to keep track of the gate time of the frequency counter. In general, frequency counters can be very precise, but not suitable for industrial applications.

Thesis synopsis

The outline of this thesis is structured in the following fashion. In Chapter 2, the design requirements of the analyser are stated. Next, Chapter 3 describes the outline of the system. It introduces the system components and the main design parameters. Chapter 4 and 5 elaborate on the proposed design as a solution for the identified problem. In Chapter 6 the implementation and validation results are shown. The thesis ends with a discussion of the results and a conclusion.

2 Programme of requirements

2.1. Requirements for the analyser group

The main goal of the analyser subsystem is to design a system that is able to read out the peak frequency of an input signal. In order to come to a verifiable design, some requirements must be set. Different categories of requirements can be identified. All requirements for the entire project are listed in in Appendix C.

Analyser requirements

The mandatory requirements describe what the analyser chain shall do to obtain the goal of the system. They describe the very core of the subsystem. Some trade-off requirements are given as well.

Table 2.1: Requirements for the design of the analyser.

ID	Requirements for the design of the analyser.
FR-MR01	The analyser shall be able to detect 10 000 discrete evenly spaced levels of signal frequency.
FR-ToR01	The input signal from one sensor shall be processed within 10 ms.
FR-ToR02	The digital analyser chain shall be modeled and tested in MATLAB.
FR-ToR03	The digital analyser chain shall be implemented in VHDL and C for verification.
FR-ToR04	The analyser shall be able to communicate to a pc for further processing of the data.

Analyser-Reader interface requirements

The interface requirements are stated below:

Table 2.2: Requirements for the interface between the analyser and the reader

ID	Requirements for the interface between the analyser and the reader
IF1-MR01	The voltage at the input of the analyser will be +/- 1V.
IF1-MR02	The signal frequency must be between 1kHz and 50 MHz.
IF1-MR03	The signal bandwidth must be at least 0.4 MHz.

Table 2.2 – *Continued from previous page*

ID	Requirements for the interface between the analyser and the reader

3 System outline

3.1. Detection method

As described in Chapter 1, the system consists of three subsystems; the sensor, the reader and the analyser. The sensor and reader groups have made design choices which can be read in the reports about those subsystems. The team has decided to design a system consisting of a resonant sensor that is inductively coupled to a readout circuit. Pseudo-Random-Noise (PRN) is sent into the reader circuit and filtered by the sensor impedance. In this way, frequencies that do not resonate with the sensor circuit are attenuated. The filtered signal is sent to the analyser to determine the resonant frequency, which is related to the force on the sensor.

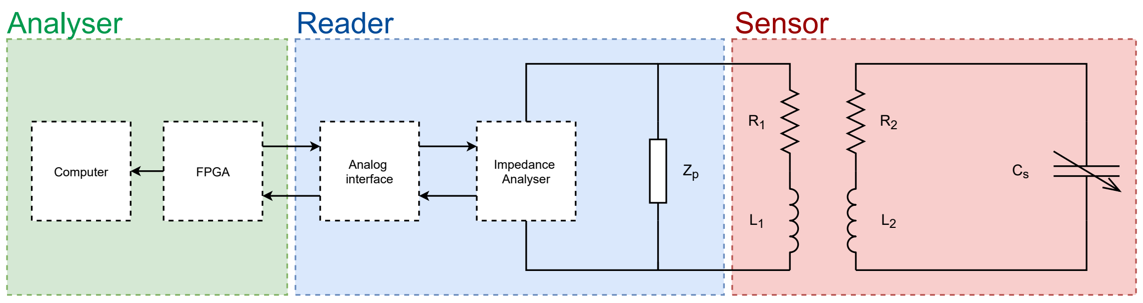


Figure 3.1: An overview of the entire weighing system

Interface with reader circuit

The interface between the analyser and the reader consists of the exchange of two signals:

1. PRN is sent to the reader as an input of the reader circuit.
2. Filtered PRN is received from the reader as an input of the analyser.

The sensor will be designed to resonate between 26.9 MHz and 27.4 MHz [1] The filtered PRN will, therefore, be limited to frequencies within this bandwidth.
In the analyser, the frequency-domain representation of the filtered PRN is divided by the frequency-domain representation PRN sent to the reader to obtain the filter characteristic of the sensor. Noise components within the analyser such as the quantization of a signal and filter components can be modelled as noise sources.

Digital signal processing

The signal will be analysed digitally. A design using digital signal processing techniques is easy to implement on a processor or an FPGA. Digital implementation also allows for quick adaptation of the design, which is specifically useful when prototyping. Furthermore, the precision of digital signal processing techniques is comparable to that of analog signal processing techniques. Since it is prohibited to compose a real-life prototype for the project, using digital techniques raises opportunities for simulating and prototyping with the help of MATLAB and VHDL.

[1] Dutch government provides constraints in frequency usage in the Netherlands. Between 26.9 MHz and 27.4 MHz is a frequency band for amateur radio usage in the Netherlands [5].

3.2. Components

Since now the main challenge for the system has been identified, and the choice of using digital signal processing techniques has been made, the necessary components for the system can be classified. In order to do this, the requirements and functional analysis of the analyser are used.

Analog to digital conversion
The first obvious step in the chain is the analog to digital conversion. The signal is sampled and quantized in this step. The reason behind the use of an ADC lies in the design choice of digital signal processing.

Digital down conversion
As stated before, the used frequency range for the system is between 26.9 MHz and 27.4 MHz. This means the frequency range from 0 Hz to 26.9 MHz does not contain interesting frequencies for the system. Down conversion in the frequency domain reduces the data rate which results in faster processing. Doing this, no signal information will be lost, but the computational costs in hardware will be reduced. Furthermore, it will result in a much higher frequency resolution of the Fourier Transform, as can be seen from Equation 3.1 discussed in the next block. The fourier transform is only computed over the frequencies containing information. The DDC has two main functionalities. First of all, it places the frequency band of interest, 26.9 MHz to 27.4 MHz, to baseband (0 to 0.5 MHz). Secondly, it reduces the sampling rate of the input with a decimation factor R.

Frequency analysis
As stated in Requirement FR-MR01, the analyser shall be able to detect 10 000 frequency levels of the input signal. In order to do this, the input signal first needs to be translated to frequency-domain. A commonly used method is frequency counting. However, this method requires stability in frequency over the gate time to be accurate. Given the short measurement time and frequency content changing due to changes in the coupling of the sensor and the reader circuit, the frequency counter is not a viable design option.
Another method for analysing the frequency content is using spectral analysis. This can be done using a Fourier Transform, more specifically an FFT, which allows fast computation. Using the FFT, a peak in the frequency spectrum can be used to detect the resonance frequency of the signal.

Frequency resolution
In order to determine 10 000 levels of deformation, 10 000 levels of resonance frequency from the impedance response of the reader circuit have to be detected. Given the bandwidth of the input signal, the required frequency resolution $R_{required}$ is:

$$R_{required} = \frac{B}{n} = \frac{0.5MHz}{10000} = 50Hz \tag{3.1}$$

As can be seen from the requirements for the overall system, the belt will move with a speed of 2 m/s. Furthermore, the sensor will have a length of 2 cm. This means the maximum time of data accumulation is 10 ms. Given this measurement time, the maximum obtainable resolution $R_{measurement}$ is (See Appendix B.1 for the derivation of this formula).

$$R_{measurement} = \frac{1}{t_{measurement}} = \frac{1}{10ms} = 100Hz \tag{3.2}$$

The resonance frequency peak can be located between the FFT bins. Therefore, it is required to perform an additional operation over the spectrum to obtain a higher resolution in resonance frequency.

Methods for increasing the frequency resolution
The frequency resolution can be increased by performing a weighted average over the FFT bins, or by interpolating in between the FFT bins. The first method is most appropriate if a few bins are available. However, the frequency response of the reader and sensor circuit will have a smooth curve rather than a sharp spike at a single frequency [7]. Therefore, polynomial interpolation in between FFT bins is a more appropriate method for increasing the frequency resolution.

Block diagram

Now that each block of the system is identified and elucidated, the total block diagram can be constructed. Each block represents a necessary step in the analysing chain. The block diagram of the analyser can be seen in Figure 3.2.

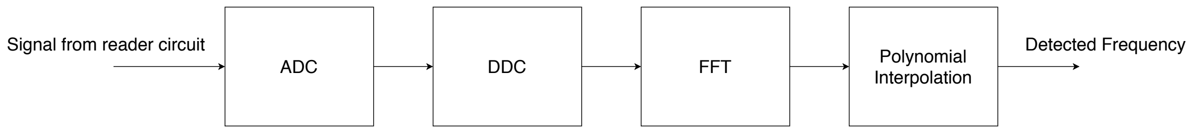


Figure 3.2: The top level block diagram of the Analyser

Interfaces within the block diagram

The input of the ADC block is also the input of the analyser as a whole. This interface is the same as between the reader and analyser, as discussed before and specified in the requirements. The signal between the ADC and DDC will be a sampled signal. The frequency peak is in the range of 26.9 and 27.4 MHz. After the DDC, the signal will have a sampling frequency of 1 MHz, and the frequency peak is in the range of 0 to 0.5 MHz. After the FFT, the frequency domain representation of the signal will be fed into the interpolation block. This block outputs the final detected frequency.

3.3. Noise division working principle

As explained before, the analyser subsystem utilises a division of Pseudo-Random-Noise to obtain a frequency spectrum with a peak on the resonance frequency. The filtered PRN is divided by the non-filtered PRN in frequency-domain, resulting in a smooth spectrum. This division will take place after the FFT block in the block diagram. More details about the noise signal can be read in the readout circuit report [7]. An example of this principle can be seen in Figure 3.3. In this example, a sampling frequency of 125 MHz is used.

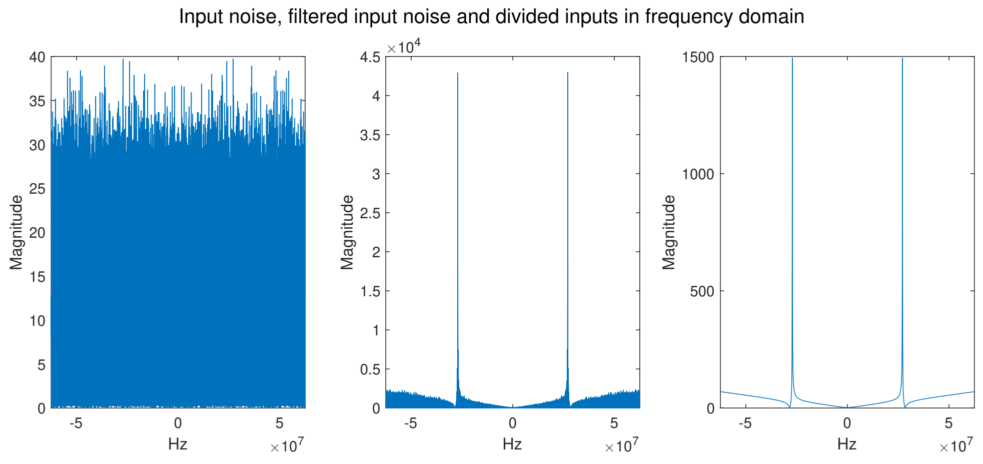


Figure 3.3: Division of the filtered PRN by the non-filtered PRN.

3.4. Signal to noise ratio and resolution of the system

Resolution

Equation 3.1 translates the design requirement of the weighing system to an output resolution of the analyser. With the detection method mentioned above, 10 000 levels of resonance frequency of the sensor within a frequency band of 0.5 MHz have to be measured. This results in an output resolution of 50 Hz in the discrete frequencies of the analyser (see Figure 3.4). With a bin size of 50 Hz, frequencies that are within 25 Hz of the mean of the segment will be placed within that frequency level.

Measurement errors

Two types of measurement errors can be distinguished in the system:

- **Offset:** Long term errors in the system due to temperature, sensor deviation, and mechanical mounting of the system. In the reader, offsets in signal power may occur due to an offset in the input voltage of the operational amplifier. It will be assumed that this offset is negligible. In the analyser, analog circuitry around the ADC will contribute to the system offset.
- **Measurement variance:** Short term deviations due to quantization and filtering in the analyser. Thermal noise of analog circuitry components is a contribution to variance in measurement outcome. Quantization errors are uniformly distributed with a zero mean value.

The system offset greatly influences the system accuracy (see Figure 3.4). However, this accuracy can be improved by calibrating the sensor and the adjustment screws of the mechanical mounting. Statistical data of the sensor should be used to calibrate the system. However, it is impossible to perfectly calibrate the weighing system with zero offset. An important constraint for the system offset is given by Equation 3.3. It is stated that the sum of the static deviation from the midpoint and the variance of measurements in the frequency bin should be smaller than the measurement rounding, or half the system resolution. With a system resolution of 50 Hz, the offset should be smaller than 25 Hz.

$$f_{offset} + 3\sigma \leq R/2 \tag{3.3}$$

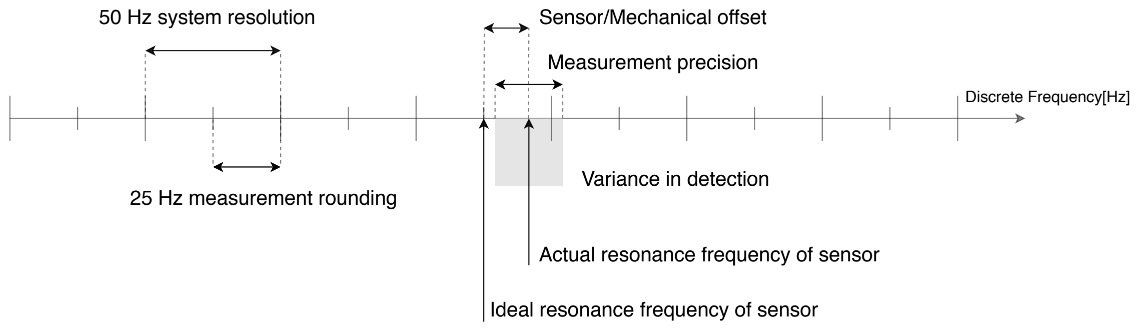


Figure 3.4: An overview of factor that should be taken into account when designing the analyser

Quantization introduces variance in the system output and therefore reduces the precision of the analyser. Moreover, the analyser will contribute most dominantly to the variance in the detection outcome. Variances due to thermal noise in analog circuitry are orders of magnitude smaller than the noise introduced by quantization [7]. These variances will not be discussed in this thesis.

Quantization noise in the analyser

With a standard deviation in the measured frequency, the detected frequencies may end up in the wrong frequency bin. Therefore, it is essential to obtain the minimal Signal to Noise Ratio (SNR) of the system needed to achieve 50 Hz resolution. Noise is added to the system by many sources. The readout circuit components, ADC circuit components and ADC quantization are examples of these sources. The equations for the noise sources will be given later in this document.

In order to find the minimum allowed SNR, a MATLAB simulation can be performed. Extra noise is added to the ideal filtered input (as in Figure 3.3) until the requirements are not satisfied any more. When computing the standard deviation σ over a sample size of N frequency peaks (after division), information about the quality of the signal is obtained. Table 3.1 shows simulation results for simulating standard deviations for different signal to noise ratios. With three standard deviations, 3σ, 99,72% of the sample size will be within the deviation.

When the MATLAB simulation is used with a sampling frequency of 125 MHz and a resonance frequency of 27098245 Hz, the result can be seen in Table 3.1. In this case, the sample size $N = 15$

and 10 values for the SNR are used. Furthermore, for this test, it is assumed that the noise sources are mainly uniformly distributed.

Table 3.1: Result of MATLAB simulation for different SNR of the input at a resonance frequency of 27098245 Hz.

SNR [dB]	Mean frequency peak [Hz]	3σ [Hz]
91.18	27098245	0.01848
84.60	27098245	0.02741
77.85	27098245	0.07374
71.25	27098245	0.1238
64.63	27098245	0.2592
57.98	27098243	13.38
51.25	27098240	16.46
44.60	27098241	25.90
37.94	27098239	68.41
31.28	27098222	84.84

Minimum signal to noise ratio of the analyser

What signal to noise ratio is acceptable for the analyser depends on the size of the offset in other parts of the weighing system, as shown in Equation 3.3. With a resolution R, a centre frequency f_{mid} of a discrete frequency bin, and a deviation of 3σ, and an offset frequency of f_{offset}, the critical frequencies $f_{critical}$ are given by Equation 3.4. These frequencies have a risk of a wrong bin detection. A good design would reduce the bandwidth of the critical frequencies as much as possible.

$$f_{critical} = [f_{mid} - R/2, f_{mid} - R/2 + 3\sigma + f_{offset}]; [f_{mid} + R/2 - 3\sigma - f_{offset}, f_{mid} + R/2] \tag{3.4}$$

Table 3.1 provides meaningful insights in what SNR to design for. The best SNR to aim for in the design process has a high marginal decrease deviation, as well a low enough value of that deviation that will allow calibration offset. Using these design principles, it is chosen to design a signal to noise ratio of 60 dB. The value of 3σ at 60 dB is 1.062 Hz. To design for a higher ratio would give an insignificant gain in deviation, while making hardware implementation much more costly. Lower signal to noise ratios have possible higher marginal gain, but the order of magnitude of deviation is in Hz which would require extremely precise calibration.

To derive the maximum allowed noise power, first, the signal power of the input has to be known. From the reader subsystem, it is known that the implementation of the system will pass a signal to the analyser with a value of 1 Vpp. It is a normally distributed signal with a standard deviation of 0.33. This translates to a signal power of $0.33^2\frac{V^2}{Hz}$. Therefore, the maximum allowed noise power is $\frac{0.33^2}{10^{\frac{60dB}{10}}} = 0.11\frac{\mu V^2}{Hz}$.

4 Analog to digital and digital down conversion

4.1. Analog to digital conversion

Sampling

Sampling Period

The analog to digital converter (ADC) is the first component in the analyser signal processor. The functionality of the ADC is to digitise the output signal from the reader circuit, maintaining all the information while adding as less noise as possible. A more mathematical description of the ADC can be found in Appendix B.1.

The Nyquist criterion states that the minimum sampling rate to sampling a signal without loss of information or aliasing is twice the bandwidth of that signal:

As described in the interface, the highest frequency is 27.4 MHz. This translates to the minimum required sampling rate of:

$$f_s = 2f_{max} = 54.8MHz \tag{4.1}$$

Noise

The analog to digital conversion will add noise to the input signal. Noise sources in an ADC are thermal and quantization noise. Thermal noise is modelled as white noise with a normal distribution. Quantization noise is modelled as uniformly distributed. For an ADC, usually, only the quantization noise is considered [16]. The Signal to Quantization Noise Ratio (SQNR) of an ADC can be defined as in Equation 4.2 [16]. This equation yields the ratio between the RMS value of the full-scale input V_{FS} and the RMS value of the noise.

$$SQNR_{ADC} = 20\log_{10}\left(\frac{\frac{V_{FS}}{2\sqrt{2}}}{V_{noise_{RMS}}}\right) \tag{4.2}$$

The SQNR in decibels is expressed as [10]:

$$SQNR_{db} = 20\log_{10}\left(\frac{2^{(}N-1)\cdot q/\sqrt{2}}{q/\sqrt{12}}\right) = 6.02N + 1.76dB \tag{4.3}$$

Where N is the number of bits in the ADC output and q the size of a quantization step. From Equation 4.2 it can easily be derived that the RMS value of the noise can be written as in Equation 4.4.

$$V_{noise_{RMS}} = \frac{\frac{V_{FS}}{2\sqrt{2}}}{10^{\frac{SNR_{ADC}}{20}}} \tag{4.4}$$

Since the noise sources are spread over the Nyquist bandwidth of the controller, the noise spectral density (in $[\frac{V}{\sqrt{Hz}}]$) can be written as in Equation 4.5.

$$e_{nADC} = \frac{\frac{V_{FS}}{2\sqrt{2}}}{10^{\frac{SNR_{ADC}}{20}}\sqrt{\frac{f_s}{2}}} \tag{4.5}$$

The minimum number of bits required for the ADC depends on the required SNR of the ADC (See Equation 4.3). Because the implementation of the analyser in hardware is not covered in this thesis (See Chapter 7), it is not possible to determine what the minimum SNR of the ADC must be.

Implementation

For implementation, the Red Pitaya has an ADC which has the proper characteristics. It has 14 bits resolution, and a sampling frequency of 125 MHz. The value of SNR_{ADC} can be found the datasheet of the Red Pitaya. It usually depends on the frequency of the input signal and the full-scale input.
Usually, some circuitry is implemented before the input signal enters the ADC. The total equivalent noise of this circuit needs to be calculated and added to the ADC noise.

4.2. Digital down conversion

The ADC module described above will have a high sampling rate to operate under the requirement of a frequency up to 27.4 MHz. However, the frequency band of interest occupies only a small part of the whole frequency range. This is were the Digital Down Converter (DDC) appears. The DDC has two main functionalities. First of all, it places the frequency band of interest, 26.9 MHz to 27.4 MHz, to base band (0 to 0.5 MHz). Secondly, it reduces the sampling rate of the input with a decimation factor R. This block is essential in the analysing chain since it decreases the needed hardware requirements to do further steps in the chain. No useful signal information will be lost, but the total analyser algorithm execution time will be lowered. The block diagram of the digital down converter can be seen in Figure 4.1. This structure is commonly used in practise for digital applications [16] [18].

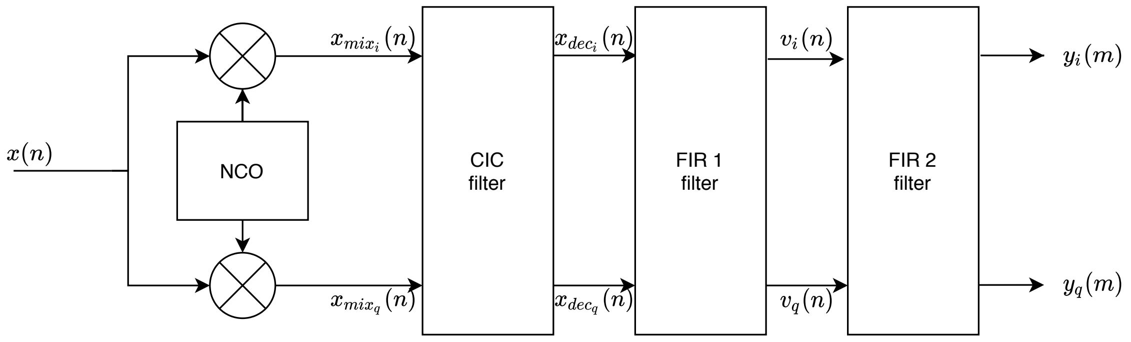


Figure 4.1: Block diagram of the DDC

Frequency mixing

The first step in the chain is the down-conversion of the frequency range of interest to base band. In order to do this, the input signal is multiplied with a complex sinusoid with a certain frequency. This oscillator frequency depends on the desired frequency shift. This can be seen in Figure 4.2. In the figure, the mixing frequency is $f_{low} = f_{mix}$. The operation results in down-converted signal. The sampled signal will be mixed with in-phase and quadrature-phase components to shift the frequency range of the signal to base band. These mixing sine and cosine functions will be provided by the digital local oscillator. The mixing operation can easily be understood using a sinusoidal input with frequency f_1, as illustrated in Equation 4.6 and 4.7. In the case of this project, f_{high} = 27.4 MHz, $f_{low} = f_{mix}$ = 26.9 MHz and the width of the band after mixing is 0.5 MHz.

$$\sin f_1 \sin f_2 = \frac{1}{2}[\cos(f_1 - f_2) - \cos(f_1 + f_2)] \tag{4.6}$$

$$\sin f_1 \cos f_2 = \frac{1}{2}[\sin(f_1 + f_2) - \sin(f_1 - f_2)] \tag{4.7}$$

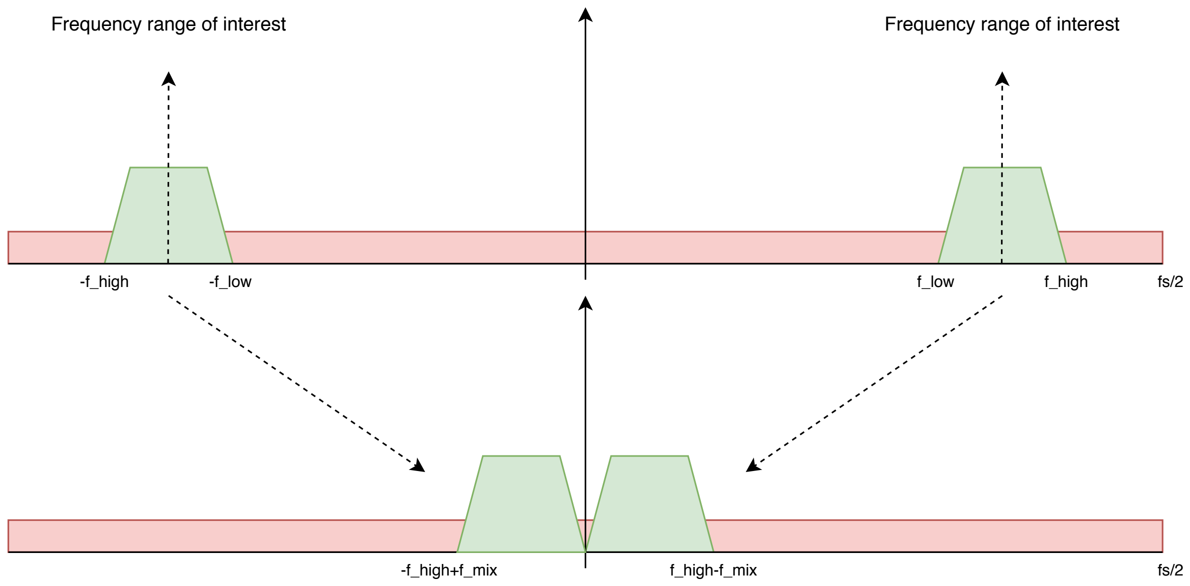


Figure 4.2: The input spectrum and output spectrum of the mixer.

NCO

The local oscillator or Numerically Controlled Oscillator (NCO) can be implemented using multiple architectures. Roughly, two types are used a lot in practice, the look-up table (LUT) or ROM architecture and the CORDIC algorithm architecture [8]. Both implementations have their advantages. The CORDIC can offer high precision results, and it uses less hardware than the LUT NCO. Also, it requires a smaller LUT than the ROM implementation. Both techniques require an additional clock, however, the CORDIC clock can be synchronized easier. Therefore, the CORDIC algorithm is chosen as the NCO implementation for this project.

CORDIC NCO

The mathematical description of the CORDIC algorithm can be found in Appendix B.2. The CORDIC algorithm can rotate a complex vector p by an angle ϕ to get vector p'. This is done using only add, subtract and shift operations. The coordinates of the vectors can be written as seen in Equation 4.8.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.8}$$

The question remains how the CORDIC theory can be applied in order to realize an NCO. From the theory in Appendix B.2, it becomes clear that the algorithm can transform a complex number p to another complex number p' without the use of multipliers. More specifically, when the input $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is used in Equation 4.8, the output of the algorithm is $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}$. This means it is possible to create sinusoids without having to use any multipliers. Also, only a small LUT is required. The values that need to be stored into the LUT are the values for ϕ_i for different iterations, as can be seen in Equation B.19. The first 5 iterations of the CORDIC algorithm with their angle values which need to be stored in the LUT can be seen in Table 4.1.

Table 4.1: First 5 iterations of the CORDIC algorithm with their angle values for the LUT.

Iteration i	Angle [radians]	Angle [degrees]	$\tan \phi_i$
0	$\tan^{-1}(2^0)$	45.0000	1
1	$\tan^{-1}(2^{-1})$	26.5651	1/2
2	$\tan^{-1}(2^{-2})$	14.0362	1/4
3	$\tan^{-1}(2^{-3})$	7.12507	1/8
4	$\tan^{-1}(2^{-4})$	3.57633	1/16

The last step needed before implementing the CORDIC based NCO is to define the input angle ϕ of the CORDIC processor, which is the phase of the desired sinusoid. The definition of the phase can be seen in Equation 4.9. If the phase is written in this way, only an addition is needed.

$$\phi[n] = \frac{f_{sig}}{f_s} n = \phi[n-1] + \frac{f_{sig}}{f_s} \tag{4.9}$$

Actually, the units of $\phi[n]$ are rotations around the unit circle. This number can be represented with an integer part and a fraction. Since the integer number of rotations is not important, only the fraction part is considered. If the fraction is written in bits, the quadrant can be derived from the two Most Significant Bits (MSB). This is needed for the pre-rotation explained in the theory. The final block diagram for the CORDIC NCO can be seen in Figure 4.3. The mixing of the signal will also take place in this block. The phase input ϕ of the CORDIC block can be represented as in Figure 4.4.

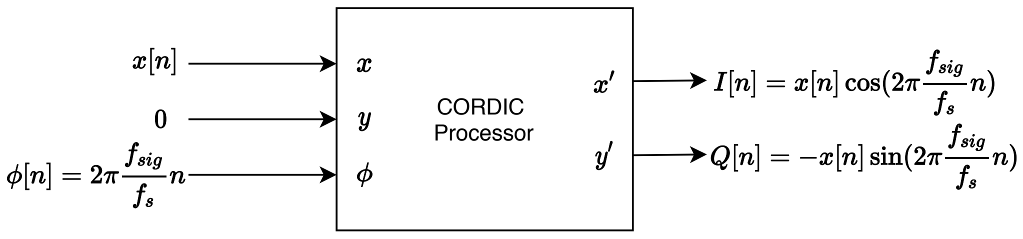

$x[n]$
0
$\phi[n] = 2\pi\frac{f_{sig}}{f_s}n$
x
y
ϕ
CORDIC Processor
x'
y'
$I[n] = x[n]\cos(2\pi\frac{f_{sig}}{f_s}n)$
$Q[n] = -x[n]\sin(2\pi\frac{f_{sig}}{f_s}n)$

Figure 4.3: The block diagram of the CORDIC NCO.

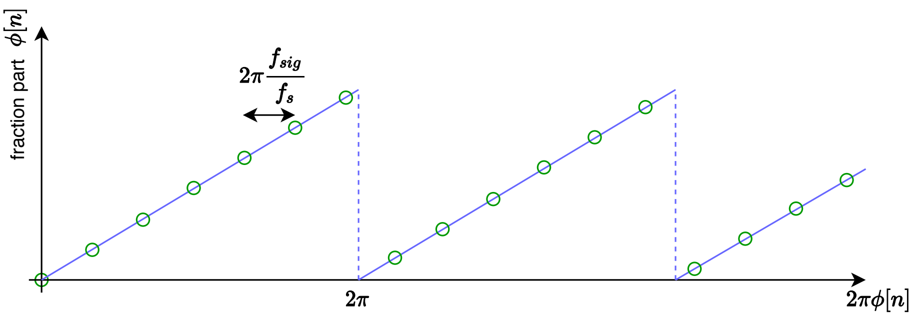


Figure 4.4: The phase input of the CORDIC NCO.

CIC filter

The CIC filter is used for the decimation of the base band signal. This implementation for the decimation of the signal offers many advantages. For example, high decimation rates can be achieved within the filter. Furthermore, it has a steep cut off, even if only a few stages are used. Lastly, it is well suited for hardware implementations, since it only uses adders and delays. An Nth order CIC filter uses N integration stages and N comb stages. The transfer function of these stages combined can be seen in Equation 4.10. In this equation, R is the decimation factor and M is the differential delay of the comb stages, which is usually limited to 1 or 2. For this project, unity will be selected as the differential delay.

$$H(z) = H_I^N(z)H_C^N(z) = \frac{(1 - z^{-RM})^N}{(1 - z^{-1})^N} = (\sum_{k=0}^{RM-1} z^{-k})^N \tag{4.10}$$

The magnitude response of the filter for large R can be written as seen in Equation 4.11. An example of the magnitude response of a 4th order CIC filter with a decimation factor R of 10 can be seen in Figure 4.5.

$$|H_{CIC}(f)| \approx |RM\frac{\sin(\pi M f)}{\pi M f}|^N \tag{4.11}$$

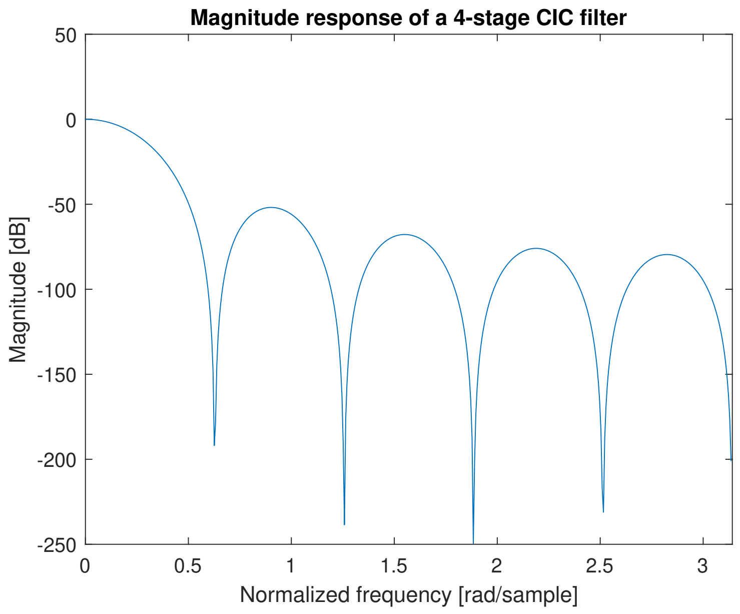


Figure 4.5: Magnitude response in dB of a 4-stage CIC filter with a decimation factor R of 10.

FIR filters

The CIC decimation filter usually lacks a wide, flat band-pass. Therefore, it is wise to construct a solution to overcome the magnitude droop. Many solutions can be implemented to overcome this problem. For example, an FIR filter with a magnitude response inverse of that of the CIC filter can be implemented. However, since the design will have a hardware implementation, it is wise to choose a solution which needs minimal hardware requirements. Therefore, a cosine based CIC filter compensator is used [4]. This filter uses only three coefficients and does not need any multipliers. The FIR magnitude response can be written as in Equation 4.12.

$$H_{FIR1}(z^M) = \left\{ \begin{array}{ll} (2^{-4}[z^{-R} - (2^4+2)z^{-2R} + z^{-3R}])^N, & \text{for } 1 < N \leq 3 \\ (2^{-4}[z^{-R} - (2^4+2)z^{-2R} + z^{-3R}])^{N-1}, & \text{for } N > 3 \end{array} \right\} \tag{4.12}$$

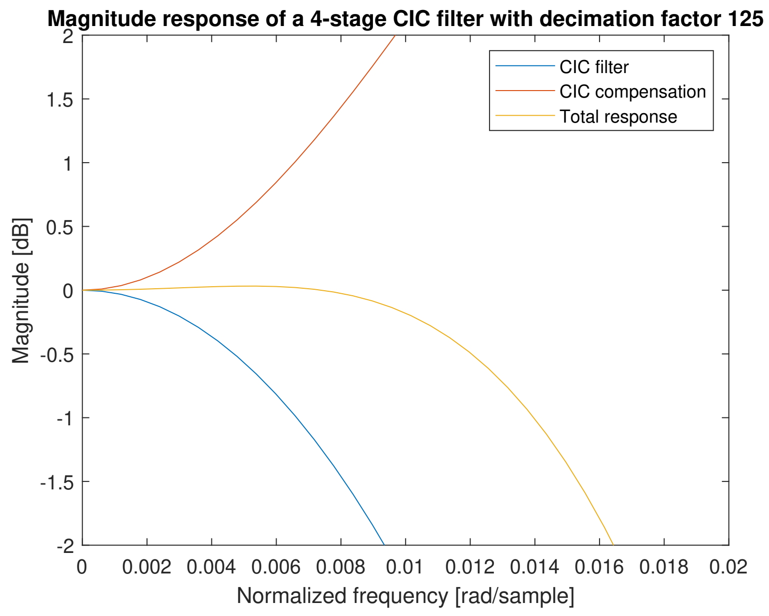


Figure 4.6: Magnitude response in dB of a compensated 4-stage CIC filter with a decimation factor R of 125.

The second FIR filter can be used as a shaping filter. This filter is used to improve the passband and stopband characteristics, like ripples and transition width. By changing the coefficients of the filter, the desired filter characteristic of the given application can be obtained.

4.2.1. DDC noise

The first computation that is done is the mixing of sine and cosine signals with the input signal to downconvert the frequency to the desired frequency band. After this mixing operation, the result is rounded. Next, the signal is decimated with the desired factor and filtered.
Because of this decimation, the final Nyquist bandwidth at the output of the converter can be written as in Equation 4.13 [16]. In this equation, D is the decimation factor and f_s the sampling frequency.

$$N_{BWfinal} = \frac{f_s}{2D} \tag{4.13}$$

Because of this downsampling and filtering, the white noise power will reduce and the SNR at the output of the system will improve. This is referred to as processing gain in literature, which can be seen in Equation 4.14.

$$P_g = 10\log\frac{\frac{f_s}{2}}{N_{BWfinal}} \tag{4.14}$$

NCO noise

Next, the noise spectral density of the Numerically Controlled Oscillator (NCO), which produces the sine and cosine waves at the desired frequency, can be computed. This is done in a similarly as with the ADC and can be seen in Equation 4.15.

$$e_{nNCO} = \frac{\frac{V_{FS_{NCO}}}{2\sqrt{2}}}{10^{\frac{SNR_{NCO}+P_g}{20}}\sqrt{N_{BWfinal}}} \tag{4.15}$$

If the signal from the ADC is N bits wide, and the signal from the NCO is P bits wide, the signal at the output of the mixer will be $N+P$ bits wide. In practice, this signal will be rounded such that M bits remain, in order to do further operations efficiently. This rounding will introduce an error. If it is assumed that the error has a rectangular, or uniform distribution [16], the noise voltage can be written as in Equation 4.16.

$$V_{round1} = \frac{2^{-M}}{\sqrt{12}} \tag{4.16}$$

The voltage noise spectral density of the rounding can be seen in Equation 4.17.

$$e_{n_{round1}} = \frac{V_{round1}}{\sqrt{\frac{f_s}{2}}} \tag{4.17}$$

Filtering noise

The DDC consists of three filters, a CIC decimation filter and two FIR filters. The main source of noise in the CIC filter stages comes from the rounding of the output of the CIC filter. This is done for similar reasons and in a similar way as with the mixing. The noise voltage V_{round2} is written similarly. The noise spectral density due to this rounding can be written as in Equation 4.18, where N_{CIC} is the decimation factor of the CIC filter.

$$e_{n_{round2}} = \frac{V_{round2}}{\sqrt{\frac{f_s}{2N_{CIC}}}} \tag{4.18}$$

The quantization of the FIR filter coefficients introduces new noise in the system. For N symmetrical coefficients, $(N+1)/2$ coefficients need to be quantized. The same number of multipliers are needed. The multiplication introduces rounding error. It is assumed that the multipliers are uncorrelated and the noise has a uniform distribution. The RMS FIR noise voltage is given in Equation 4.19

$$V_{FIR} = \sqrt{\frac{N+1}{2}} V_{round,multiplier} \tag{4.19}$$

The noise spectral densities of FIR 1 and FIR 2 are given in Equations 4.20 and 4.21, where N_{FIR1} is the decimation factor of the first FIR filter.

$$e_{n_{FIR1}} = \frac{V_{FIR1}}{\sqrt{\frac{f_s}{2N_{CIC}}}} \tag{4.20}$$

$$e_{n_{FIR2}} = \frac{V_{FIR2}}{\sqrt{\frac{f_s}{2N_{CIC}N_{FIR1}}}} \tag{4.21}$$

Total DDC noise

The total noise of the DDC can be found by adding the spectral densities. This total noise is uniformly distributed. The noise expression can be seen in the block diagram in Figure 4.7.

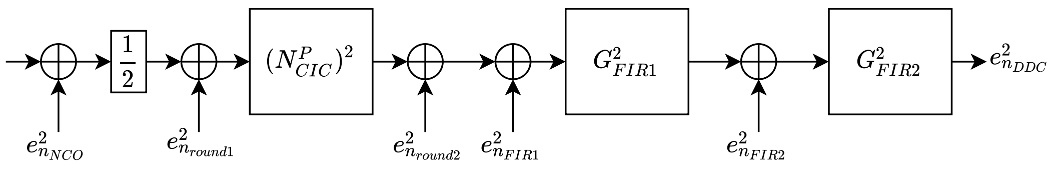


Figure 4.7: Block diagram of the DDC noise sources.

From this figure, the total noise spectral density of the DDC can be derived as given in Equation 4.22.

$$e_{n_{DDC}} = 2\sqrt{G_{FIR2}^2(e_{n_{FIR2}}^2 + G_{FIR1}^2(e_{n_{FIR1}}^2 + e_{n_{round2}}^2 + ({N_{CIC}^P}^2(e_{n_{round1}}^2 + \frac{1}{2}e_{n_{NCO}}^2))))} \tag{4.22}$$

Where P is the order of the CIC filter and G_{FIR1} and G_{FIR2} are the gain factors of the FIR filters.

4.3. DDC testing

For testing the DDC, an input signal with a peak frequency at 27.1 MHz is used. The mixing operation is performed with a frequency of 26.9 MHz. The means that the original band of 26.9 to 27.4 MHz is transformed to 0 to 0.5 MHz. Therefore, the frequency peak after the DDC is expected to be at 0.2 MHz. From Figure 4.8, it becomes clear that the frequency downconverting and downsampling operations work as expected.

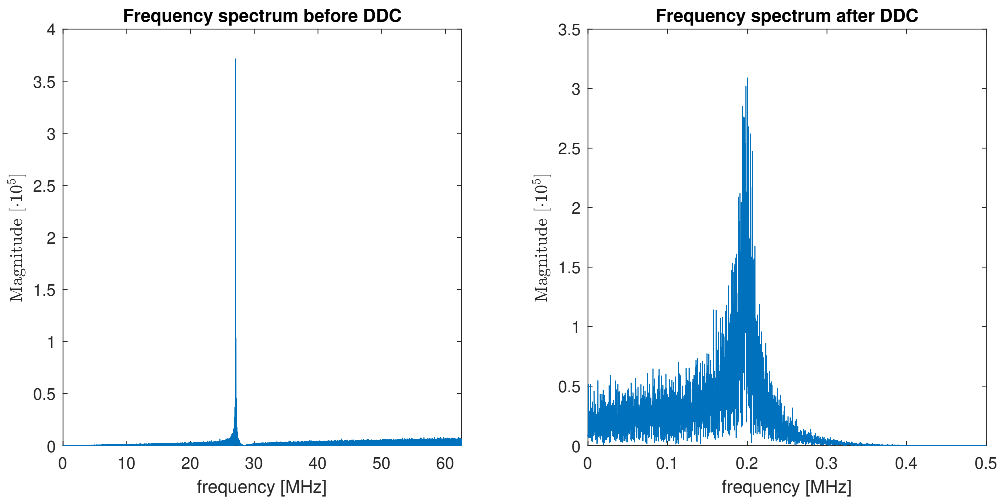


Figure 4.8: Frequency spectrum before the DDC (left) and spectrum after DDC (right).

5 Frequency domain computations

This chapter describes the design steps of the frequency domain computations for finding the peak resonance frequency of the reader circuit. These include performing a Fourier transform and spectrum polynomial interpolation.

5.1. Fast Fourier Transform

The fast Fourier transform (FFT) is a mathematical tool used to determine the frequency content of a signal. The FFT of a signal with length N is calculated according to Equation 5.1.

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn} = \sum_{n=0}^{N/2-1} x[2n]W_{N/2}^{kn}W_N^{k} + \sum_{n=0}^{N/2-1} x[2n+1]W_{N/2}^{kn} \tag{5.1}$$

With each W_N^{nk} representing a frequency bin.

$$W_N^{nk} = e^{-j2\pi(kn)/N} \tag{5.2}$$

Thus ah FFT of length N produces N different frequency bins. Therefore, the longer signal length, the smaller the size of the frequency bins of the FFT, and the higher the resolution. However, the speed of the weighing belt sets a constraint in the maximum length of the Fourier transform. Given that the belt moves a speed of v_{belt} = 2 m/s and a sensor length l_{sensor} of 2 cm, the maximum time slot in which a detection must be made is:

$$t_{measurement} = \frac{l_{sensor}}{v_{belt}} = \frac{2cm}{2m/s} = 10ms \tag{5.3}$$

The number of measurements within the time slot depends on the conditions in which the measurement is performed. Mechanical vibrations of the conveyor belt are in the order of Hz. These vibrations can, therefore, be assumed constant throughout a 10 ms period. It is therefore chosen to perform a single measurement in a single passing of the load cell. The equates to an FFT length of:

$$L = t_{measurement} \cdot f_s = 10ms \cdot 1MHz = 10000samples \tag{5.4}$$

An important constraint for the FFT to operate properly is that the time calculate the FFT of signal must be less than a sampling period of the down sampled system. If this is the case and no time buffer for calculating Fourier transforms is needed, a real-time system can be implemented. Thus the computation time must be less than 1 μs.

Windowing
Discontinuities might occur at the boundaries of the framed inputs. However, due to the self-windowing properties of noise burst [7], no spectral leakages will occur.

5.2. FFT noise analysis

FFT bin noise power

The FFT block receives an input vector $\vec{x}$ that contains N samples. The FFT block outputs $\vec{X}$, the frequency domain representation of $\vec{x}$.

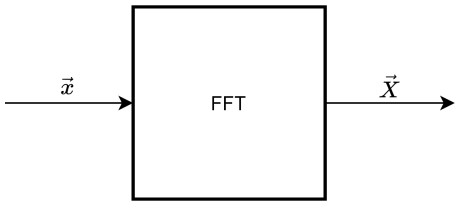


Figure 5.1: FFT input and output diagram

The input vector $\vec{x}$ can be split up into two parts, the signal part $\vec{y}$ and the noise part $\vec{z}$. The noise part $\vec{z}$ is equal to the additive noise V_n from Figure 5.1, so it is a random variable with a uniform distribution, variance (noise power) ${V_n}^2$ and a zero mean. The FFT of $\vec{x}$ is given in Formula 5.5.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi}{N}kn} \tag{5.5}$$

Since the FFT is a linear operation the output vector may also be split up in a signal part $\vec{Y}$ and a noise part $\vec{Z}$.

All entries of $\vec{z}$ are uniformly distributed and independent variables. The sum of uniform distributed variables has an Irwin-Hall distribution. For large sums, the distribution approximates a normal distribution, for which holds that variances of individual independent elements can be added up. The summation over the random variables in $\vec{z}$ can be split up into a real and an imaginary part, as shown in Equation 5.6.

$$Z_k = \sum_{n=0}^{N-1} z_n \cos\Big(-\frac{2\pi}{N}kn\Big) + j\sum_{n=0}^{N-1} z_n \sin\Big(-\frac{2\pi}{N}kn\Big) \tag{5.6}$$

The variance of a scaled random variable holds $Var[aX] = a^2Var[X]$ so the total noise expression for ${Z_k}^2$ is given by Equation 5.7.

$${Z_k}^2 = \sum_{n=0}^{N-1} {z_n}^2 \cos^2\Big(-\frac{2\pi}{N}kn\Big) + j\sum_{n=0}^{N-1} {z_n}^2 \sin^2\Big(-\frac{2\pi}{N}kn\Big) \tag{5.7}$$

When N is large the $\cos(...)$ and $\sin(...)$ expression will be sampled throughout in the domain $[0, 2\pi]$. The average value of this sampled value can be found by integrating $\cos(\theta)$ from 0 to 2π and dividing by 2π, which is equal to 0.5. For large N the sum for ${Z_k}^2$ (the FFT bin noise power) converges to Equation 5.5.

Scaling the FFT

When the FFT input and output length is increased from N_0 to N_1 the output noise power will only increase by a factor of $\frac{N_1}{N_0}$. This means that the total signal to noise ratio is increased by a factor of $\frac{N_1}{N_0}$. So for every doubling of the FFT length 3db of signal to noise ratio is added, assuming that the signal and noise stay constant in the added measurement time.

5.3. Interpolation

Limitations in the bandwidth

Due to law restrictions in the frequency usage, the bandwidth might be a constraint in the design of the analyser. The bandwidth itself cannot create enough frequency resolution for the required number of steps. Interpolation between frequency bins will increase the frequency resolution of the analyser (see Chapter 3).

Polynomial interpolation

Polynomial interpolation will fit a polynomial given in Equation 5.8 between the set of data points with a minimum mean square error.

$$X_k = \beta_0 + \beta_1\omega_k + \beta_2{\omega_k}^2 + \cdots + \beta_m{\omega_k}^m + \epsilon_k \tag{5.8}$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & \omega_0 & \omega_0^2 & \dots & \omega_0^m \\ 1 & \omega_1 & \omega_1^2 & \dots & \omega_1^m \\ 1 & \omega_2 & \omega_2^2 & \dots & \omega_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_{N-1} & \omega_{N-1}^2 & \dots & \omega_{N-1}^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{N-1} \end{bmatrix} \tag{5.9}$$

The shorter matrix representation is given by $\vec{X} = \Omega\vec{\beta} + \vec{\epsilon}$. The solution for $\vec{\beta}$ that minimizes the mean squared error is given in Equation 5.10.

$$\vec{\beta} = (\Omega^T\Omega)^{-1}\Omega^T\vec{X} \tag{5.10}$$

From Equation 5.10 it clear that the estimates for β are linearly dependent on $\vec{X}$, so β can be written as a combination of $\vec{\lambda}$, the actual coefficients for the fitted polynomial and $\vec{\zeta}$, a vector of normally distributed random variables that represent the estimation error.

Parameters of interpolation

Furthermore, there is a trade-off in the order of interpolation. High orders of interpolation coincide with a large computational complexity. On the other hand, low orders of interpolation will result in inaccurate interpolation.

The region to interpolate over should not be the entire spectrum. Rather, only samples in the neighbourhood of the peak should be taken into account with the interpolation. By accurately observing the impedance response of the reader circuit [7], a peak width of 2.5 kHz was found, which translates to 500 samples for a measurement time of 10 ms.

Determining the resonance frequency

When Equation 5.8 is applied to the Fast Fourier transform of Equation 5.1, the resonance frequency can be found by computing the maximum of the interpolated function.

$$F = max(P(x)) \tag{5.11}$$

Next, the resonance frequency is rounded off the nearest frequency bin with resolution R. Figure 5.2 shows a simulation of interpolating in between the frequency bins. Numerical results are shown in Table 5.1. The consistency of the interpolation will be tested in Chapter 6.

Table 5.1: Results of a test of the interpolating algorithm.

Input frequency [MHz]	Estimated frequency [MHz]	Error [Hz]
27.100012	27.099999	13

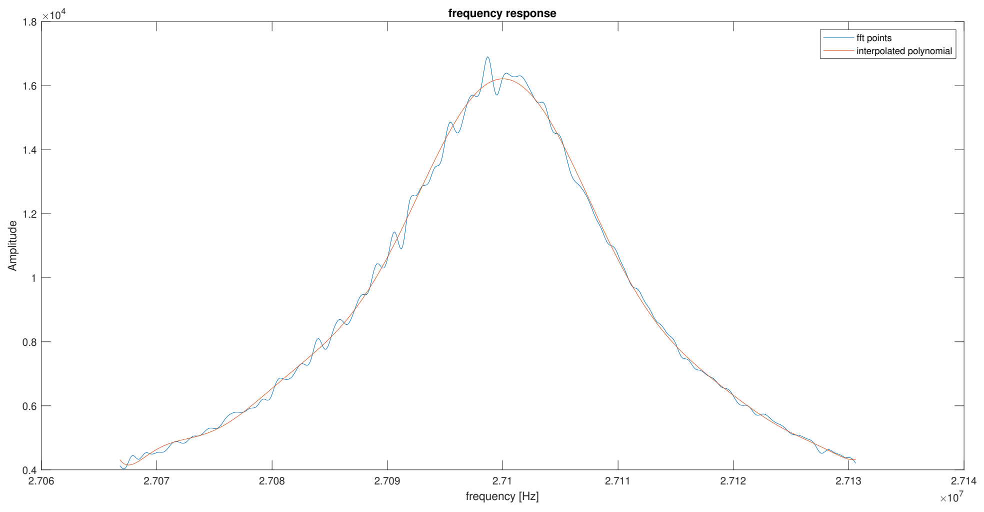


Figure 5.2: An overview of frequency interpolation at a frequency of 200 000 Hz. (This is a downconverted frequency which corresponds to a frequency of 27.1 MHz in the frequency domain of 26.9-27.4 MHz.)

Low pass filtering

With testing the interpolation function in MATLAB, the signal over which was interpolated contained noisy spikes in the frequency spectrum, which caused false detections of the resonance frequencies (see Figure 5.3). This problem can be solved by using a low pass filter before interpolating the data. Before filtering, spikes with relative height higher than 5% of the signal maximum are removed from the data.

The low pass filter has such a characteristic that it filters with a normalised passband frequency of ω_{pass} and attenuates with 60dB in the stopband. There is a trade-off in the choice of ω_{pass}: the frequency must not be too high or the filter will not remove the noise significant enough. On the other hand, choosing a too low ω_{pass} will result in unnecessary loss of information. With an ω_{pass} of 0.1, the filter operates optimally. The result is a more smooth frequency spectrum (see Figure 5.3).

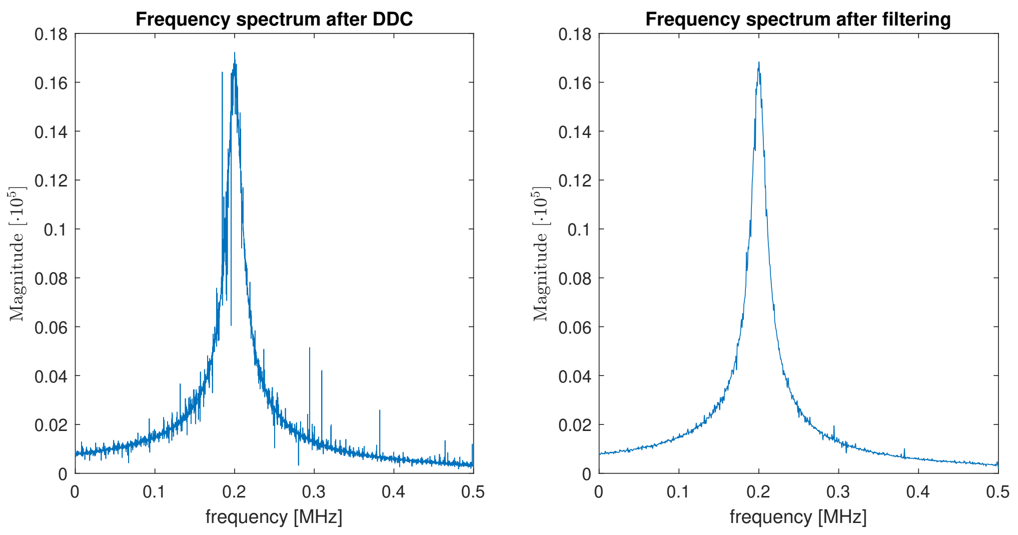


Figure 5.3: Frequency spectrum with noisy spikes (left) and filtered spectrum after (right). This is a downconverted frequency which corresponds to a frequency of 27.1 MHz in the frequency domain of 26.9-27.4 MHz.

6 System testing and implementation

In order to validate if the designed system is able to satisfy the requirements, an implementation needs to be made. This is also stated in the requirements, as can be seen in Requirement FR-ToR02 and FR-ToR03. First of all, a MATLAB simulation of the system is made.

6.1. Testing for accuracy and precision

In the test case, 40 detections of a 27.1MHz resonance frequency in the reader circuit are performed. The result is shown in Figure 6.1.

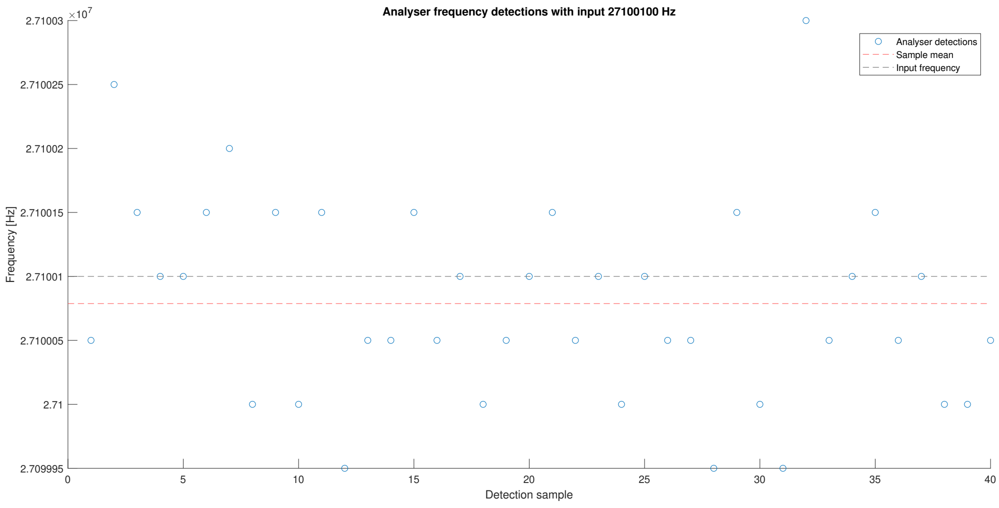


Figure 6.1: Detection samples of the analyser at input 27100100 Hz

Table 6.1: Analyser statistics of 40 detections of 27.1MHz resonance frequency at 50 Hz resolution

Sample size	**Sample mean**	**Accuracy [Hz]**	**Sample variance [Hz] (σ^2)**	**Sample standard deviation (σ)**	**Hit ratio**
40	27100078	22	6147	78.4	$\frac{8}{40}$ = 20%

Table 6.1 shows the variance, and the average of the results, from this, the accuracy and precision of the analyser have been calculated. From these results, it can be observed that the accuracy of the analyser is smaller than half the desired resolution, which is sufficient. The variance, however, is too large to have 50 Hz output resolution. Moreover, the hit ratio is only 20%, meaning that the probability of a correct outcome is 0.2. The constraint given by Equation 3.3 in Chapter 3 states is not met:

$$f_{offset} + 3\sigma = 257.2 \nless R/2 \tag{6.1}$$

Table 6.2 shows the test results with a resolution of 100 Hz, which will result in 5000 discrete levels of resonance frequencies. With a hit ratio of 45%, the analyser will still not be precise enough.

Table 6.2: Analyser statistics of 40 detections of 27.1MHz resonance frequency at 100 Hz resolution

Sample size	Sample mean	Accuracy [Hz]	Sample variance [Hz] (σ^2)	Sample standard deviation (σ)	Hit ratio
40	27100223	23	12405	109	$\frac{18}{40}$ = 45%

Finally, the analyser is tested with a resolution of 500 Hz, which is required to achieve the minimum number of 1000 discrete levels in the weighing industry for a viable product. Fortunately, the hit ratio of this test was 100 %, thus the analyser is 100% accurate with 0 variance at this resolution.

Signal to noise ratio
The SNR of the system test was around 20 dB. When comparing the obtained value for 3σ with the SNR table in Chapter 3, this result is expected. The reason for choosing this SNR is due to an error in the MATLAB code. This resulted in the generation of excess noise by the DDC, which will further be discussed in Chapter 7.
At this signal to noise ratio, the maximum achievable resolution with 100% hit rate is 250 Hz.

Testing changes in resonance frequency

In order to validate the functionality of the analyser, the following test case has been set up: To test a single passing of a load cell over the reader circuit. In the test case, the input of the analyser are consists of signals with 50 Hz difference in resonance frequency, starting from 27.1MHz. The test is successful if the analyser is able to detect the changes in the signal. The following is assumed:

- The over passing time of the load cell is 10 ms.
- The average coupling factor of the sensor to the reader circuit is 0.3. [1]

Figure 6.2 shows the frequency response resonance response for the reader circuit at 27.1MHz. From Equation 6.2, it is derived that a 50Hz change in resonance frequency may be approximated by changing the capacitance in the sensor by 0.2 fF (See Appendix D.1 for exact values of the sensor). This is done by linear approximation.

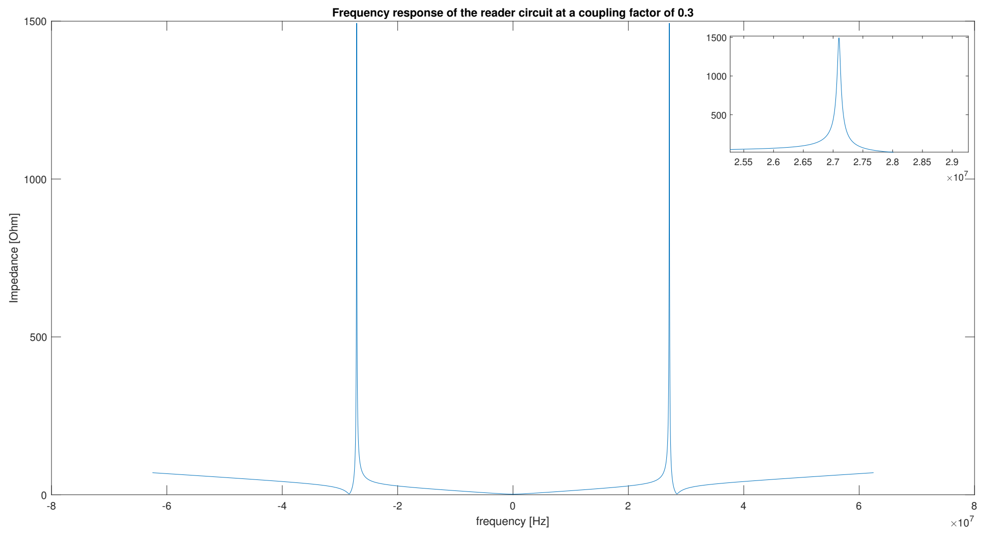


Figure 6.2: Frequency response of the reader circuit at 27.1MHz

[1] In practice, the coupling factor will variate during the 10 ms passing of the load cell. For the first test case, this phenomena has been ignored, and an average coupling factor has been chosen.

The values for the inductors and capacitor in the reader and sensor circuit to simulate the resonance frequency are found in Appendix D.1

$$f_{res} = \frac{1}{2\pi LC} \tag{6.2}$$

Results

Table 6.3 shows the simulation results of the test case. The data indicates that the analyser is capable of estimating the resonance frequency with an accurately of 50Hz. However, errors occur frequently in the data. The error at most one frequency bin. Therefore, the desired 50 Hz output resolution is not met.

Table 6.3: Simulation results for the total system testing.

Input frequency [Hz]	Frequency bin [Hz]	Est. frequency bin [Hz]	Error [bins]	Error [Hz]
27100013	27100000	27100000	0	0
27100065	27100050	27100050	0	0
27100117	27100100	27100100	0	0
27100170	27100150	27100150	0	0
27100222	27100200	27100150	1	50
27100274	27100250	27100200	1	50

6.2. Hardware implementation

After validating the results with MATLAB, a hardware implementation can be made. The first block in the system is the ADC. Therefore, an ADC with the right parameters should be picked for the system. The minimum sampling frequency should be 54.8 MHz, as explained before. To come to a right ADC choice, order parameters, like the number of bits and the SNR of the ADC should be taken into account. With the minimum required SNR specified, these parameters can be filled in into the noise equations to check if the chosen ADC is acceptable. Noise sources added by circuitry around the ADC should be taken into account as well.
The DDC block can be realized using an FPGA board [18]. In this case, the noise equations can be filled in too. It is possible to implement the DDC using VHDL code. VHDL code is easily generated using the MATLAB tool; HDL Coder. This tool converts MATLAB code and Simulink diagrams into HDL code.
The last two blocks, the FFT and spectrum interpolation can be done on a processor, using C for instance.
When combining the needed hardware, handheld FPGA boards with additional hardware appear to be a reasonable choice for prototyping. A good example of this is the Red Pitaya STEMLab 125-14 board [14]. Some relevant parameters and features of this board are summarized in Table 6.4. The SNR of the ADC on this board 73 dB, which is well above 60 dB [15].

Table 6.4: Red Pitaya STEMLab 125-14 features [14].

FPGA	FPGA Xilinx Zynq 7010 SOC
Processor	Processor DUAL CORE ARM CORTEX A9
Sampling rate	125 MS/s
ADC bits	14
Communication protocols	I2C, SPI, UART

7 Discussion

7.1. Requirements

The requirements provided by Intralox were set up, aiming to deliver a physical prototype. However, due to the COVID-19 pandemic, building and performing tests on physical prototypes was prohibited throughout the BAP period. This affected the progress of the project. Most of the requirements have not been passed completely as no physical testing has been performed.

- **Requirement FR-MR01**: The requirement to be able to detect 10 000 discrete evenly spaced levels of signal frequency has not been completed. As stated in the results of system testing 6, the output resolution of 50 Hz is not met consequently. However, this was due to an SNR that is too low for proper frequency estimation. When the SNR is enlarged, this requirement will be met, as can be read in Chapter 3.
- **Requirement FR-ToR01**: The processing time of 10 ms could not be tested. This was due to a combination of factors. First of all, due to the COVID-19 situation, producing a real-life prototype was not allowed. Furthermore, the Red Pitaya test-board arrived too late, such that not enough time for writing testing VHDL code was left. This requirement can easily be verified when the hardware application is made.
- **Requirement FR-ToR02**: The system is simulated in MATLAB. Therefore, this requirement is satisfied. However, extra work is needed for some parts of the code, as the DDC generated excess noise in the code.
- **Requirement FR-ToR03**: The system is not implemented and simulated in VHDL or other HDL for the same reasons as for Requirement FR-ToR01. However, as stated before, when using MATLAB HDL coder, the VHDL code can be generated.
- **Requirement FR-ToR04**: The system can communicate with a PC when the Red Pitaya implementation is made.

7.2. Sensor bandwidth

The analyser team has designed the analyser according to the design requirements of Intralox, see Chapter 2. This implies that the resonance frequency of the sensor circuit will change between 26.9 MHz and 27.4MHz by 0.2mm deformation. However, in practice, the sensor will only resonate between 27.1 MHz and 27.405MHz. This can also be seen from Requirement IF1-MR03. Thus for the practical implementation of the system, the analyser must be tuned to this bandwidth. A consequence of the smaller bandwidth is that a higher resolution and therefore a higher SNR in the analyser is required.

7.3. System testing with a time-varying coupling factor

The system test in Chapter 6 assumed a constant coupling factor. However, in practice, the coupling factor will change throughout a measurement cycle. Although the average coupling factor of the actual movement might approximately be 0.3 as used in the test case, the effective measurement time will not be 10 ms. With little to no coupling, the signal contains insufficient information to contribute to a This will have a significant effect possible resolution in the FFT bins (See Equation 3.2 in Chapter 3).

Attempts have been made to create a test signal in MATLAB, which has a time-varying coupling factor. However, due to the complexity of the problem and short project time, no successful simulation has been performed. Thus in practice, test data must be collected by physically moving the load over the reader circuit with a speed of 2 m/s.

7.4. Linear approximation of the frequency changes

The capacitive change in the sensor has only been simulated for the maximal deformation of 0.2 mm [7]. Thus, no information about the linearity between levels of deformation and changes in resonance frequency is known. Therefore, it is questionable whether the linear resolution scale used in the analyser is compatible with the sensor.

8 Conclusion and Recommendations

8.1. Conclusion

To conclude, an analyser system has been designed in an attempt to improve the previous efforts by Intralox in designing an in-situ conveyor belt weighing system.
A block diagram has been constructed as a result of a functional analysis of the analyser. The block diagram for the analyser consists of an analog to digital converter, digital down converter and a polynomial interpolation unit.
Next, an analysis of the required precision of the analyser is developed from which a minimum signal to noise ratio is calculated. In addition, the analysis provided insight in limits of the variance and offset in the detections of the analyser, such that an accurate and precise design could be made. Moreover, quantization noise due to finite precision in digital signal processing is the most significant noise source of the analyser. Quantization noise increases the variance of the detection output of the analyser and reduces the analyser precision. Other parts of the weighing system, such as the sensor, require calibration with test weights to minimise the offset as much as possible.

Subsequently, the components of the block diagram were elaborated. Models for determining the signal to noise ratio of the ADC have been created, as well as constraints in the sampling frequency. The design of the DDC was finalised by subdividing the DDC into smaller components. The DDC places the frequency band of interest to base-band, and it reduces the sampling rate. The numerical oscillator (NCO) of the DDC has been implemented using a CORDIC algorithm. With this, the input signal could be mixed with a digital sine/cosine function. The frequency mixer is followed by a CIC filter. This filter decimates the input by a factor D.
A compensation filter together with an FIR filter has been realised to create a flat band-pass response. The noise contribution of the DDC is mainly related to rounding of filter coefficients and the finite precision of digital filtering. The quantization errors have a uniform distribution.

After the conversions, the signal is transformed to frequency-domain using an FFT. Polynomial interpolation has been used to increase the frequency resolution. Low pass filtering over the frequency data removes unwanted spikes that may affect the interpolation and the search algorithm for the resonance frequency. The search algorithm rounds the detected frequency nearest discrete frequency depending on the resolution.

In the last part of the thesis, system testing has been applied to the analyser. Tests were performed by simulating the analyser in MATLAB. The desired frequency resolution of 50 Hz was not always obtained in the test cases because off a too low signal to noise ratio.

8.2. Recommendations

Sensor bandwidth

A major show stopper in obtaining a proper resolution of the weighing system is the limited bandwidth. A high-frequency resolution, and therefore a high SNR, is required to be able to detect 10 000 levels within a bandwidth of 0.5MHz. In an interview of an RF engineer at Philips [7], it became clear that the bandwidth constraint by Dutch law did not apply to the weighing belt system, because of the low power spectral densities in reader sensor coupling. This would enable the use of larger bandwidths, making it possible to detect 10 000 levels with a lower frequency resolution and a lower SNR.

Data aggregation on the PC

Once a stable analyser system has been engineered, it is highly recommended to aggregate the load cell data on a PC for further processing of the data. A dynamic real-time image of the weighing belt can be constructed by periodically updating the load cell weight information. With this information, applications such as providing positional feedback of packets to the belt controller, determining the centre of gravity on packets on the belt, and parallel packet weighing can be implemented in software in the Intralox weighing belt.

Reader configuration for longer measurement time

In Appendix A, a mechanical solution in which reader circuit move along sensors is proposed to increase the measurement time.

A
Reader configurations for longer reader time

Available measurement time to read out the sensor is a show stopper variable of the resolution of an FFT. Therefore, the analyser team has come up with a belt configuration that allows for longer coupling time between the sensor and the reader to increase the measurement time.

Operating mechanism
Instead of using a single row of reader circuits, multiple reader rows can be installed to read out multiple rows of load cells. To do so, the rows of reader circuits will be placed on a rail and follow the sensor for a length of time that allows for a precise measurement of the load cell weight. During this period, the relative velocity between the sensors and the readers is zero.

Next, a servo motor reverses direction of travel to place the reader at the initial position for a new measurement. Because the weighing belt moves continuously, a second reader block is required to operate in opposite phase of movement with the first reader block. The second reader block reads out the sensor rows missed by the first block as the first block is brought back to the starting position. The cycle repeats. As the second reader block movement to the starting position, the first reader block reads out a rows of sensors.

Risks and challenges
To implement this configuration, several aspects have to be taken in to account:

- **Timing of the system:** The configuration requires a high precision in the timing of the movement in the reader blocks, since the reader blocks must be placed exactly below the sensors and must operate exactly in opposite phase of movement. Also, a constant belt speed is required.
- **Mechanical noise sources:** The configuration increases the number of moving parts in the weighing system, therefore increasing the number of sources of vibration. Therefore, mechanical engineers must design a system with as little vibration as possible. Furthermore, a longer measurement time may cause the mechanical vibrations not to be constant throught out the measurement period.

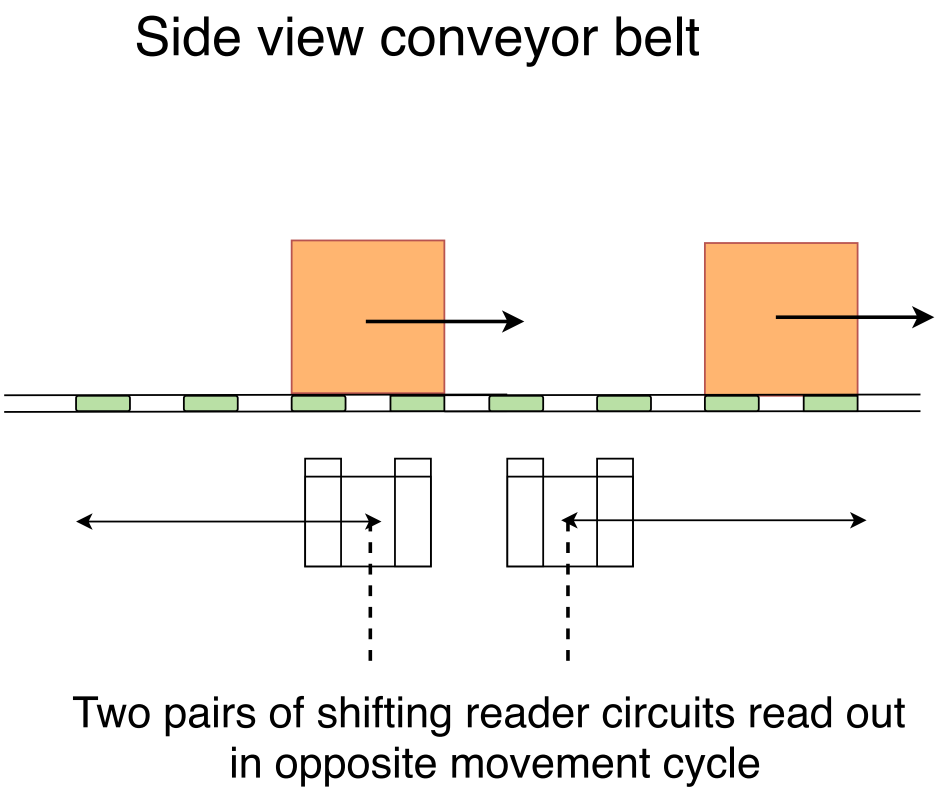

Side view conveyor belt
Two pairs of shifting reader circuits read out in opposite movement cycle

Figure A.1: A reader configuration which allows for longer measurement time

B Equations

B.1. Analog to digital converter

The analog to digital converter operates at a sampling frequency f_s. The sampling period T_s is:

$$T_s = \frac{1}{f_s} \tag{B.1}$$

Sampling function
Sampling is done via a pulse train δ_{T_s}.

$$\delta_{T_s} = \sum_n \delta(t - nT_s) \tag{B.2}$$

Sampled signal

$$x_s(t) = x(t)\delta_{T_s}(t) \tag{B.3}$$

Quantization

Number of Quantization levels
Given a quantizer with M bits, the number of quantization levels N is:

$$N = 2^M \tag{B.4}$$

Quantization step
The quantization step Δ is the difference between two quantization levels.

$$\Delta = \frac{2max|x(t)|}{N} \tag{B.5}$$

Quantization error
The quantization error is the difference between the sampled and the quantized signal.

$$\epsilon(nT_s) = x(nT_s) - \hat{x}(nT_s) \tag{B.6}$$

The magnitude of this value is bounded by the quantization step.

$$0 \leq \epsilon(nT_s) \leq \Delta \tag{B.7}$$

From Equation 4.2 it can easily be derived that the RMS value of the noise can be written as in Equation B.8.

$$V_{noise_{RMS}} = \frac{\frac{V_{FS}}{2\sqrt{2}}}{10^{\frac{SNR_{ADC}}{20}}} \tag{B.8}$$

Since the noise sources are spread over the Nyquist bandwidth of the controller, the noise spectral density (in $[\frac{V}{\sqrt{Hz}}]$) can be written as in Equation B.9.

$$e_{nADC} = \frac{\frac{V_{FS}}{2\sqrt{2}}}{10^{\frac{SNR_{ADC}}{20}}\sqrt{\frac{f_s}{2}}} \tag{B.9}$$

Frequency resolution for time harmonic signals
The frequency resolution of the signal is sampling frequency divided by the length of the FFT:

$$\Delta f = \frac{F_s}{L} \tag{B.10}$$

The required sampling time for a FFT is:

$$T_{FFT} = T_s L = \frac{L}{F_s} = \frac{1}{\Delta f} \tag{B.11}$$

Let n the number of distinguishable steps required in the frequency spectrum. The required bandwidth for the input signal is given by:

$$B = n \cdot \Delta f = \frac{n}{T_{FFT}} \tag{B.12}$$

The Nyquist criterion set limits to the minimal sampling frequency after down conversion:

$$B \leq \frac{F_s}{2} \tag{B.13}$$

Thus the minimal length of the FFT is:

$$L = \frac{F_s}{\Delta f} = 2n = 2 * B * T_{FFT} \tag{B.14}$$

B.2. CORDIC algorithm

The COordinate Rotational Digital Computer (CORDIC) algotihm was first developed by Volder in 1959 [17]. It is used to iteratively compute trigonometric functions using only add, subtract and shift operations. The CORDIC algorithm can rotate a complex vector p by an angle ϕ. This results in a new vector p', which can be written as

$$p' = p \cdot e^{j\phi} = p(\cos\phi + j\sin\phi) \tag{B.15}$$

Equating the definitions of p and p' in Equation B.15 gives the coordinate components of vector p', as can be seen in Equation B.16.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = R \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{B.16}$$

The rotational matrix R can be rewritten using $\cos\phi = \frac{1}{\sqrt{1+\tan{(\phi)}^2}}$.

$$R = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} = \cos\phi \cdot \begin{bmatrix} 1 & -\tan\phi \\ \tan\phi & 1 \end{bmatrix} = \frac{1}{\sqrt{1+\tan{(\phi)}^2}} \cdot \begin{bmatrix} 1 & -\tan\phi \\ \tan\phi & 1 \end{bmatrix} \tag{B.17}$$

Now, the total rotation θ between two vectors can be performed with the help of a series of angular rotation steps. This process of performing small rotations to obtain the total rotation is called the sequence of CORDIC micro rotations. The angular steps can be expressed as

$$p_i = \frac{1}{\sqrt{1+\tan{(\phi_i)}^2}} \cdot \begin{bmatrix} 1 & -\tan\phi_i \\ \tan\phi_i & 1 \end{bmatrix} \cdot p_{i-1} \tag{B.18}$$

This means that ideally, it holds that $\sum_{i=0}^{\infty} \delta_i \cdot \phi_i = \theta$, with $\delta_i = \pm 1$. However, since an infinite sum is not feasible, an approximation has to be made. This can be represented as $\sum_{i=0}^{N-1} \delta_i \cdot \phi_i \approx \theta$. A further reduction of the complexity of the calculations can be reached by restricting $\tan\phi_i$ in Equation B.18 to values of $\pm 2^{-i}$. This means ϕ_i can be written as in Equation B.19. The result can be seen in Equation B.20.

$$\phi_i = \tan^{-1}(\frac{1}{2^i}) \tag{B.19}$$

$$p_i = \frac{1}{\sqrt{1+\delta_i \cdot 2^{-2i}}} \cdot \begin{bmatrix} 1 & -\delta_i \cdot 2^{-i} \\ \delta_i \cdot 2^{-i} & 1 \end{bmatrix} \cdot p_{i-1} = K_i \cdot \begin{bmatrix} 1 & -\delta_i \cdot 2^{-i} \\ \delta_i \cdot 2^{-i} & 1 \end{bmatrix} \cdot p_{i-1} \tag{B.20}$$

Replacing the tangent multiplication with a division by a power of 2 is an efficient method to reduce computation complexity. This due to the fact that the division can easily be implemented using a bit shift operation.
The only multiplication left in the algorithm is the multiplication with the scale factor K_i. This factor can be ignored during the process itself, and implemented afterwards. The multiplication of the K_i approach a constant value, which can be seen in Equation B.21 (for larger values of n) [12].

$$\lim_{n\to\infty} K(n) = \prod_{i=0}^{n} K_i \approx 0.60725294104140 \tag{B.21}$$

Intermediate micro rotations are represented with a new variable Z.

$$Z_{i+1} = \theta - \sum_{i=0}^{N-1} \phi_i \tag{B.22}$$

In this equation, the total or given rotational angle is represented as θ. For every small rotation of angle ϕ_i, both Z and δ_{i+1} need to be calculated. Variable δ_{i+1} can be computed as

$$\delta_{i+1} = \begin{Bmatrix} -1, & \text{for } Z_{i+1} < 0 \\ +1, & \text{for } Z_{i+1} \geq 0 \end{Bmatrix} \tag{B.23}$$

It is important to notice that the inverse tangent function which defines ϕ_i (Equation B.19) has a convergence region between $\frac{-\pi}{2}$ and $\frac{\pi}{2}$. Therefore, order to be able to use the CORDIC algorithm for all angles some pre-rotations or quadrant corrections need to be set for the regions $\frac{\pi}{2} < \theta < \pi$ and $-\pi < \theta < -\frac{\pi}{2}$.
For the first region, the relations $\cos\theta + \frac{\pi}{2} = -\sin\theta$ and $\sin\theta + \frac{\pi}{2} = \cos\theta$ is used [12]. This results in the substitution of variables $x \to y$, $-y \to x$, $\theta \to \theta - \frac{\pi}{2}$.
For the second region, the relations $\cos\theta - \frac{\pi}{2} = \sin\theta$ and $\sin\theta - \frac{\pi}{2} = -\cos\theta$ is used [12]. This results in the substitution of variables $-x \to y$, $y \to x$, $\theta \to \theta + \frac{\pi}{2}$.

C
Requirements for the entire system

Project requirements
System requirements

Table C.1: Requirements for the complete system

ID	Requirements for the design of the sensor.
SYS-MR01	The sensor shall be able to pass over information to the receiver moving with 2 m/s'.
SYS-MR02	The sensor shall be read out using a wireless method.
SYS-ToR01	The sensor should preferably be invariant for temperature differences in the range of 0 °C and 50 °C.

Sensor requirements

Table C.2: Requirements for the design of the sensor.

ID	Requirements for the design of the sensor.
SEN-MR01	Deformation of the loadcell diapraghm (maximum 0.2 mm) shall result in changed electrical parameters of the sensor.
SEN-MR02	The sensor shall be invariant to positional errors of the coupling $\pm$3 mm in the x-direction.
SEN-MR03	The sensor should be coupled to the reader for distances of 1-4 mm in the z-direction.
SEN-MR04	The receiver shall be able to read sensor with rotational errors of: • +/- 10 degrees in the width (X) dimension of the belt, due to mechanical placement and belt module tilt. • +/- 5 degrees in the length (Y) dimension of the belt, due to mechanical placement. • +/- 5 degrees in the height (Z) dimension due to mechanical placement
SEN-MR05	The sensor shall be passive, in the sense that it shall operate without the need to be connected to a power supply.

Table C.2 – *Continued from previous page*

ID	Requirements for the design of the sensor.
SEN-ToR01	The sensor shall have at most a diameter of 2 cm.
SEN-ToR02	The sensor should preferably cost less than €1,-.
SEN-ToR03	The sensor should preferably have an oscillating frequency between 26.965 and 27.405 MHz.
SEN-ToR04	The sensor should preferably consist of one single piece of PCB material.

Analyser requirements

Table C.3: Requirements for the design of the analyser.

ID	Requirements for the design of the analyser.
FR-MR01	The analyser shall be able to detect 10 000 discrete evenly spaced levels of signal frequency.
FR-ToR01	The input signal from one sensor shall be processed within 10 ms.
FR-ToR02	The digital analyser chain shall be modeled and tested in MATLAB.
FR-ToR03	The digital analyser chain shall be implemented in VHDL and C for verification.
FR-ToR04	The analyser shall be able to communicate to a pc for further processing of the data.

Reader requirements

Table C.4: Requirements for the design of the analog reader.

ID	Requirements for the design of the analog reader.
RED-MR01	The reader shall be able to read a sensor without interference of other sensors.
RED-MR02	The reader shall be able to perform at least 10 measurements on a cell averaging the result of mechanical vibrations.
RED-MR03	The reader will generate an analog signal that enables the analyser to accurately determine the sensor change.

Interface requirements

Analyser-Reader Interface

Table C.5: Requirements for the interface between the analyser and the reader

ID	Requirements for the interface between the analyser and the reader
IF1-MR01	The voltage at the input of the analyser will be +/- 1V.
IF1-MR02	The signal frequency must be between 1kHz and 50 MHz.
IF1-MR03	The signal bandwidth must be at least 0.4 MHz.

Reader-Sensor Interface

Table C.6: Requirements for the interface between the reader and the sensor.

ID	Requirements for the interface between the reader and the sensor.
IF2-MR01	The sensor will use coupled coils to transfer energy from and to the reader.
IF2-MR02	The sensor value can be read by the reader independent of the coupling coefficient k.
IF2-ToR01	The resistor R_2 should not exceed 0.25Ω.
IF2-ToR02	The capacitor C_1 should be larger than 1pF.
IF2-ToR03	The capacitor L_2 should be so that the quality factor is » 100.
IF2-ToR04	The capacitor C_p should not exceed $\frac{C_1}{10000}$.
IF2-Tor05	The sensor will have a changing resonance frequency in a bandwidth of at least 0.4 Mhz.

D
Sensor and readout circuit

D.1. Circuit diagram

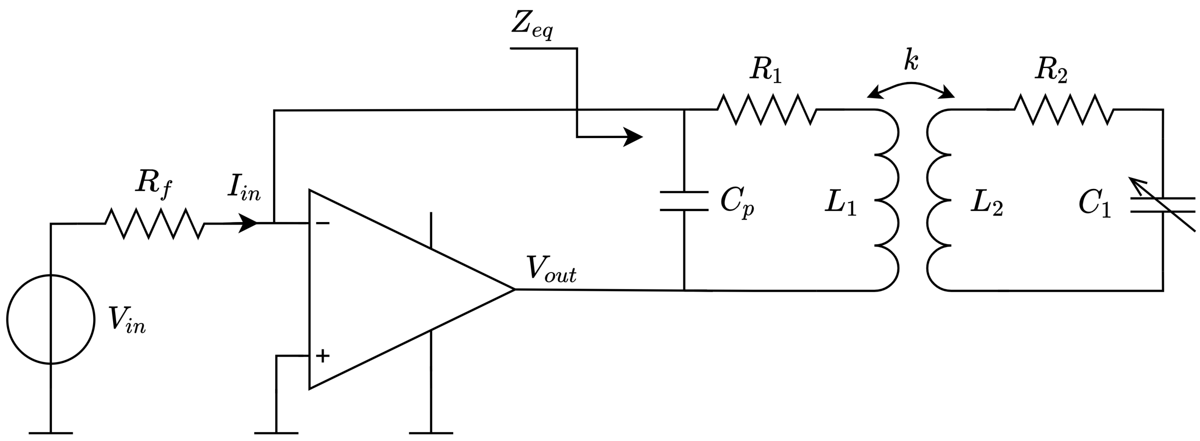


Figure D.1: Sensor and readout circuit

D.2. Inductor and capacitor values used for testing

fres [Hz]	L2 [μ H]	Cs [pF]
27100000	0.6665	51.17489
27100050	0.6665	51.17487
27100100	0.6665	51.17485
27100150	0.6665	51.17483
27100200	0.6665	51.17481
27100250	0.6665	51.17479

E
Matlab Code

Listing E.1: Matlab script used to simulate the input signal for the analyser system.

```
%Matlab simulation for analyser part of BAP
%Mauries van Heteren (4712145) and Jonathan Dijkstra (4696778)
%Test input signal simulation code

function [filtered_white_noise,whitenoise,f_in] = simulate_reader(signal_number)
    %time for sensor movement
    fs = 125e6;
    t_max = 10e-3;
    %number of coupling factors for simulation
    N_k = 1e2;

    %create a white noise vector of 10 ms
    [t,whitenoise] = create_white_noise(t_max);

    %f axis
    f_axis = linspace(-fs/2,fs/2,length(whitenoise));
    %create the filtered output signal
    [filtered_white_noise,f_in] = create_output_signal(whitenoise,t,N_k,signal_number);
end

function [filtered_white_noise,fres] = create_output_signal(whitenoise,t,N_k,signal_number)
    % create the filtered white noise vector
    bin_size = length(whitenoise)/N_k;
    [L1,L2,C,R1,R2,freqs,omegas,k,fres] = initiate_reader_parameters(N_k,bin_size,signal_number);

    k_temporal = 0.3;
    frequency_response = calc_frequency_response(omegas,k_temporal,C,L1,L2,R1,R2);

    fft_whitenoise = fftshift(fft(whitenoise));
    fft_filtered_white_noise = fftshift(fft_whitenoise.*abs(frequency_response));
    filtered_white_noise = ifft(fft_filtered_white_noise);

    fs = 125e6;
    f = linspace(-fs/2,fs/2,length(filtered_white_noise));
    f_small = linspace(25e6,29e6,10^4);
    frequency_small = calc_frequency_response(2*pi*f_small,k_temporal,C,L1
      ↪ ,L2,R1,R2);
end

function [t,whitenoise_trim] = create_white_noise(t_max)
    %specifications of the test signal
    fs = 125e6;
    t = linspace(0, t_max, fs*t_max);
    %create white noise
    whitenoise_vect = (randn(size(t)))*0.01;
    whitenoise_vect_long = [whitenoise_vect whitenoise_vect
      ↪ whitenoise_vect];

    %trim the noise
    offset = t_max*fs*0.4;
    whitenoise_trim = whitenoise_vect_long(1 + offset:t_max*fs + offset);
end

function [L1,L2,C,R1,R2,freqs,omegas,k,fres] = initiate_reader_parameters(
    ↪ N_k,bin_size,signal_number)
    fs = 125e6;
    t = 10e-3;
    %inductor values
    L1 = 0.6665e-6;
    L2 = 0.6665e-6;

    %capacitor value at 27.1MHz
    Cs = 51.7489e-12;

    %Change in 50Hz in this region can be approximated by
    Delta_C = 0.2e-15;
    C = Cs - (signal_number -1)*5*Delta_C;

    fres = 1/(2*pi*sqrt(C*L1));
    %DC bias
    R1 = 2;
    %internal resistance of the inductor
    R2 = 0.07;

    %frequency axis
    freqs = linspace(-fs/2,fs/2, fs*t);
%    freqs = linspace(-fs/2,fs/2, bin_size);
    omegas = freqs*2*pi;

    k = [linspace(0, 0.5, (N_k/2)) linspace(0.5, 0, (N_k/2))];
end

function Z = calculate_impulse(w,k,C,L1,L2,R1,R2)
    % calculate the impulse
    Z = R1 + 1i*w*L1 + w^2*k^2*L1*L2*(1/(R2+1i*w*L2 + 1/(1i*w*C)));

end

function Zs = calc_frequency_response(omegas,k,C,L1,L2,R1,R2)
    % calculate the frequency response
    Zs = [];
    for p = 1:length(omegas)
        Zs = [Zs calculate_impulse(omegas(p), k, C, L1, L2, R1, R2)];
    end
end
```

Listing E.2: Matlab script used for testing the entire system.

```matlab
%Matlab simulation for analyser part of BAP
%Mauries van Heteren (4712145) and Jonathan Dijkstra (4696778)
%Main code for testing system
resolution = 250;

%Simulate the system multiple times
[f_axis1,x_in_divided_1,f_estimated1,f_in1] = estimate_frequency(1);
[f_axis2,x_in_divided_2,f_estimated2,f_in2] = estimate_frequency(3);
[f_axis3,x_in_divided_3,f_estimated3,f_in3] = estimate_frequency(3);
[f_axis4,x_in_divided_4,f_estimated4,f_in4] = estimate_frequency(3);
[f_axis5,x_in_divided_5,f_estimated5,f_in5] = estimate_frequency(3);
[f_axis6,x_in_divided_6,f_estimated6,f_in6] = estimate_frequency(3);
[f_axis7,x_in_divided_7,f_estimated7,f_in7] = estimate_frequency(3);
[f_axis8,x_in_divided_8,f_estimated8,f_in8] = estimate_frequency(3);
[f_axis9,x_in_divided_9,f_estimated9,f_in9] = estimate_frequency(3);
[f_axis10,x_in_divided_10,f_estimated10,f_in10] = estimate_frequency(3);
[f_axis11,x_in_divided_11,f_estimated11,f_in11] = estimate_frequency(3);
[f_axis12,x_in_divided_12,f_estimated12,f_in12] = estimate_frequency(3);
[f_axis13,x_in_divided_13,f_estimated13,f_in13] = estimate_frequency(3);
[f_axis14,x_in_divided_14,f_estimated14,f_in14] = estimate_frequency(3);
[f_axis15,x_in_divided_15,f_estimated15,f_in15] = estimate_frequency(3);
[f_axis16,x_in_divided_16,f_estimated16,f_in16] = estimate_frequency(3);
[f_axis17,x_in_divided_17,f_estimated17,f_in17] = estimate_frequency(3);
[f_axis18,x_in_divided_18,f_estimated18,f_in18] = estimate_frequency(3);
[f_axis19,x_in_divided_19,f_estimated19,f_in19] = estimate_frequency(3);
[f_axis20,x_in_divided_20,f_estimated20,f_in20] = estimate_frequency(3);
[f_axis21,x_in_divided_21,f_estimated21,f_in21] = estimate_frequency(3);
[f_axis22,x_in_divided_22,f_estimated22,f_in22] = estimate_frequency(3);
[f_axis23,x_in_divided_23,f_estimated23,f_in23] = estimate_frequency(3);
[f_axis24,x_in_divided_24,f_estimated24,f_in24] = estimate_frequency(3);
[f_axis25,x_in_divided_25,f_estimated25,f_in25] = estimate_frequency(3);
[f_axis26,x_in_divided_26,f_estimated26,f_in26] = estimate_frequency(3);
[f_axis27,x_in_divided_27,f_estimated27,f_in27] = estimate_frequency(3);
[f_axis28,x_in_divided_28,f_estimated28,f_in28] = estimate_frequency(3);
[f_axis29,x_in_divided_29,f_estimated29,f_in29] = estimate_frequency(3);
[f_axis30,x_in_divided_30,f_estimated30,f_in30] = estimate_frequency(3);
[f_axis31,x_in_divided_31,f_estimated31,f_in31] = estimate_frequency(3);
[f_axis32,x_in_divided_32,f_estimated32,f_in32] = estimate_frequency(3);
[f_axis33,x_in_divided_33,f_estimated33,f_in33] = estimate_frequency(3);
[f_axis34,x_in_divided_34,f_estimated34,f_in34] = estimate_frequency(3);
[f_axis35,x_in_divided_35,f_estimated35,f_in35] = estimate_frequency(3);
[f_axis36,x_in_divided_36,f_estimated36,f_in36] = estimate_frequency(3);
[f_axis37,x_in_divided_37,f_estimated37,f_in37] = estimate_frequency(3);
[f_axis38,x_in_divided_38,f_estimated38,f_in38] = estimate_frequency(3);
[f_axis39,x_in_divided_39,f_estimated39,f_in39] = estimate_frequency(3);
[f_axis40,x_in_divided_40,f_estimated40,f_in40] = estimate_frequency(3);

f1_rounded = scale_f(f_in1,resolution);

f_measured1 = [f_estimated1 f_estimated2 f_estimated3 f_estimated4
    ↪ f_estimated5 f_estimated6 f_estimated7 f_estimated8 f_estimated9
    ↪ f_estimated10 f_estimated11 f_estimated12 f_estimated13
    ↪ f_estimated14 f_estimated15 f_estimated16 f_estimated17
    ↪ f_estimated18 f_estimated19 f_estimated20];
f_measured2 = [f_estimated21 f_estimated22 f_estimated23 f_estimated24
```

```
    ↪ f_estimated25 f_estimated26 f_estimated27 f_estimated28
    ↪ f_estimated29 f_estimated30 f_estimated31 f_estimated32
    ↪ f_estimated33 f_estimated34 f_estimated35 f_estimated36
    ↪ f_estimated37 f_estimated38 f_estimated39 f_estimated40];

f_measured = [f_measured1 f_measured2];

function scaled_f = scale_f(f_in,resolution)
    n_steps = 0.5e6/resolution;
    frequency_scale = 26.9e6 : resolution : 27.4e6 - resolution;
    for i = 1:n_steps
        diff = abs(frequency_scale(i) - f_in);
        if(diff < resolution/2)
            scaled_f = frequency_scale(i);
        end
    end
end

function [f_axis, x_in_divided, f_estimated,f_in] = estimate_frequency(
    ↪ signal_number)
    t_max = 10e-3; %maximum time for the load cell over the reader
    t_fft = 10e-3;  %maximum allowed time per measurement

    %sample rate of the adc
    fs = 125e6;

    %sample rate after downconversion
    fs2 = 1e6;
    %bandwidth of the signal
    f_min = 26.9e6;
    f_max = 27.4e6;
    %parameter for the amount of noise in the signal
    noise_level = 0.05;

    %bandwidth constraint by government law
    B_max = f_max-f_min;
    %maximum achievable levels per measurement by the given bandwidth
    n_max = 2*B_max * t_fft;
    %fft length at Nyquist given the bandwidth constraint
    L = n_max;

    %Define input signal from reader to analyser (test signal from Jasper)
    [x_in,x_n,f_in] = simulate_reader(signal_number);

    %ADC for input signal from reader
    adc_output = real(adc_block(x_in));
    %ADC for noise input
    adc_output_noise = adc_block(x_n);

    %DDC for input signal from reader
    ddc_output = ddc_block(adc_output, fs, f_min, f_max);
    %DDC for noise input
    ddc_output_noise = ddc_block(adc_output_noise, fs, f_min, f_max);

    fs_new = fs/125;
    N = length(ddc_output) ;
    dF = fs_new/N;
    f_axis = -fs_new/2:dF:fs_new/2-dF;

    %FFT for input signal from reader
    fft_output = fft_block(ddc_output);
    %FFT for noise input
    fft_output_noise = fft_block(ddc_output_noise);

    %Divide to eliminate noise:
    x_in_divided = fft_output./fft_output_noise;
    %interpolation
    f_estimated = spectrum_interpolation_block(x_in_divided,f_axis);

end
```

Listing E.3: Matlab script used to simulate the quantization of the ADC.

```
%Matlab simulation for analyser part of BAP
%Mauries van Heteren (4712145) and Jonathan Dijkstra (4696778)
%Quantization simulation code

function adc_output = adc_block(x)
    N_bits = 14; %ADC bits
    A = max(real(x));
    A = round(A*100)/100;
    Range = [-A A];
    x_digitised = adc_quantize(x,N_bits,Range);
    adc_output = x_digitised;
end

function adc_out = adc_quantize(x,N_bits,Range)
    %This function quantizes the input signal
    %specifications analog to digital converter
    %transpose if necessary
    sz = size(x);
    if sz(1) > sz(2)
      x = x.';
    end
    q_lev = 2^N_bits;
    %midPnt = q_lev/2;    % center point
    R_max = Range(2);
    R_min = Range(1);
    step = (R_max - R_min)/q_lev;
    offset = 0.5*step;
    if (length(step) > 1 || step <= 0)
      error('Quantization range = [min_value, max_value],  max_value >
          ↪ min_value')
    end
    % min_max clamping
    x(find(x>=Range(2))) = R_max;
    x(find(x<=Range(1))) = R_min;
    % quantization
    adc_out = (round((x-R_min)./step))*step;
    adc_out = adc_out + R_min;
    adc_out(find(adc_out > R_max-offset)) = R_max-step;
    deltaR = diff(Range);
    if deltaR > 0
      x_min = Range(1) - deltaR/10;
      x_max = Range(2) + deltaR/10;
    else
      error('The upper limit must greater than the lower limit\n')
    end
end
```

Listing E.4: Matlab script used to simulate the DDC.

```matlab
%Matlab simulation for analyser part of BAP
%Mauries van Heteren (4712145) and Jonathan Dijkstra (4696778)
%DDC simulation code

function ddc_output = ddc_block(input, fs, f_min, f_max)

    bandwidth = f_max - f_min; %peak frequency range of signal
    f_nco = f_min; %frequency of the oscillator

    [mix_out_i,mix_out_q] = mixer_block(input, f_nco, fs); %mix the signal
        ↪ with local oscillator

    decimation_factor = 125; %down sampling rate of the DDC
    cic_filter_order = 15; %CIC filter order

    cic_output_i = cic_block(mix_out_i, decimation_factor,
        ↪ cic_filter_order); %CIC filter the input i comp (choose
        ↪ cic_block2 for matlab cic function)
    cic_output_q = cic_block(mix_out_q, decimation_factor,
        ↪ cic_filter_order); %CIC filter the input q comp (choose
        ↪ cic_block2 for matlab cic function)

    cic_output_comp1_i = cic_comp1_block(cic_output_i, decimation_factor,
        ↪ cic_filter_order, fs); %compensation filters
    cic_output_comp1_q = cic_comp1_block(cic_output_q, decimation_factor,
        ↪ cic_filter_order, fs); %compensation filters

    ddc_output = complex(cic_output_q,cic_output_i); %make a complex
        ↪ number from i and q comp


    function [mix_out_i,mix_out_q] = mixer_block(mix_input, f_nco, fs)
        bits_angle_vector = 14;
        iterations = 15;
        step_size = 2*pi*f_nco/fs;

        %Calculate the angle input of the cordic algorithm phi[n]
        for i=1:1250
            theta(i) = i*step_size;

            if theta(i) >= (2*pi)
                multiple = floor(theta(i)/(2*pi));
                theta(i) = theta(i)-(multiple*2*pi);
            end

            theta(i) = fi(theta(i),1,bits_angle_vector);
        end

        %calculate the cordic output
        [y, x] = cordicsincos(theta,iterations);

        %Make the cordic output the proper length
        needed_length = length(mix_input)/length(x);
        x = repmat(x,1,needed_length);
        y = repmat(y,1,needed_length);

        %Do the mixing operations
        mix_out_i = mix_input.*x;
        mix_out_q = mix_input.*y;
    end

    function cic_output = cic_block2(CIC_in, decimation_factor,
       ↪ cic_filter_order)
        CIC_in = [zeros(1,11000) CIC_in(11001:1239000) zeros(1,11000)];
        R = decimation_factor;   %decimation factor
        K = cic_filter_order;  %Order of filter / number of filter stages
        cicDecim = dsp.CICDecimator(R,1,K);
        cic_output = cicDecim(CIC_in');
    end


    function cic_output = cic_block(CIC_in, decimation_factor,
       ↪ cic_filter_order)

        R = decimation_factor;   %decimation factor
        K = cic_filter_order;  %Order of filter / number of filter stages
        B = [1,zeros(1,R-1),-1]; %the numerator coefficients of the filter
        A = [R,-R]; %the denominator coefficients of the filter

        [h, t] = impz(B, A);      % Impulse response of one stage
        impulse_response = h;
        if K > 1                  % calculate impulse response for K stages
            for p=2:K
                impulse_response = conv(impulse_response, h);
            end
        end

        B = impulse_response;  %put final coefficients in B
        A = 1;
        cic_output = filter(B,A,CIC_in);      %filter the input with the
            ↪ impulse response

    end

    function cic_comp1_output = cic_comp1_block(CIC_comp1_in,
       ↪ decimation_factor, cic_filter_order, Fs)
        R = decimation_factor;   %decimation factor
        K = cic_filter_order;  %Order of filter / number of filter stages

        %Compensation filter according to https://ieeexplore.ieee.org/
            ↪ stamp/stamp.jsp?tp=&arnumber=5349314
        A_comp = 1; %the numerator coefficients of the filter
        B_comp = [0 , zeros(1,R-1) , (-1/16), zeros(1,(2*R-1)-(R+1)),
            ↪ (9/8), zeros(1,(3*R-1)-(2*R+1)), (-1/16)]; %the denominator
            ↪ coefficients of the filter

        [h_comp, t_comp] = impz(B_comp, A_comp);      % Impulse response of
            ↪ one stage
        impulse_response_comp = h_comp;
        if K > 3
            K = K - 1;
        end
        if K > 1                % calculate impulse response for K stages
            ↪  (see paper)
            for p=2:K
                impulse_response_comp = conv(impulse_response_comp, h_comp
                    ↪ );
            end
        end
        B = impulse_response_comp;  %put final coefficients in B
        A = 1;

        cic_comp1_output = filter(B,A,CIC_comp1_in);     %filter the input
            ↪  with the impulse response

        %Low pass filter parameters
        Fp  = 0.63e6;       % passband−edge frequency
        Rp  = 0.00057565; % Corresponds to 0.01 dB peak−to−peak ripple
        Rst = 1e−9;       % Corresponds to 180 dB stopband attenuation
        Fst = 1.65e6;  % Transition Width

        h_lp = firgr('minorder',[0,Fp/(Fs/2),Fst/(Fs/2),1],[1 1 0 0],...
        [Rp Rst])';%make the low pass filter with minimum order

        z = zeros(length(impulse_response_comp)−length(h_lp),1) ;  % zeros
        h_lp = [h_lp ; z] ;

        h_lp = impz(h_lp,1); % make impulse response of filter

        cic_comp1_output = filter(h_lp,1,cic_comp1_output);     %filter
            ↪ the input with the impulse response

        cic_comp1_output = downsample(cic_comp1_output, R);   %downsample
            ↪ the filtered signal
    end

end
```

Listing E.5: Matlab script used for testing the polynomial interpolation algorithm.

```matlab
%Matlab simulation for analyser part of BAP
%Mauries van Heteren (4712145) and Jonathan Dijkstra (4696778)
%Spectrum interpolation algorithm

 function [f_estimated] = spectrum_interpolation_block(x,f)
    %specification of the interpolation
    [m,n] = size(x);
    percentage = 0.08;
    search_size = 320;
    resolution_polynomial = 10e7;
    resolution = 250;
    cutoff = 0.10;

    %Filter unwanted high freq comp.
    x = abs(x);
    A = filter(x,cutoff);


    %find the frequency peak
    [B,C]= find_peak(A,search_size,percentage);

    %frequency domain of interest
    f4 = f(B(1):B(end));

    %frequency domain of interpolation
    f_precision = double(linspace(f4(1),f4(end),resolution_polynomial));

    %apply polynomial interpolation on the data points
    [p,~,mu] = polyfit(f4, C, 12);
    polynomial = polyval(p,f_precision,[],mu);

    %estimate the frequency of the interpolated polynomial
    f_interpolated = estimate_frequency(polynomial,f_precision);

    %scale the frequency
    f_estimated = scale_f(f_interpolated,resolution);

    %calculate weighted average
%        f_weighted_average = calc_weighted_average(C,f);
    for i = 1:length(f4)
        f4(i) = f4(i) + 26.9e6;
    end
    for i = 1:length(f_precision)
        f_precision(i) = f_precision(i) + 26.9e6;
    end
    %plot of the frequency
        plot(f4, C);
        hold on;
        plot(f_precision,polynomial);
        hold off;
        title('frequency response');
        legend('fft points','interpolated polynomial');
        xlabel('frequency [Hz]');
        ylabel('Amplitude');

```

```
end

function scaled_f = scale_f(f_in,resolution)
    %scale frequency range for interpolation
    n_steps = 0.5e6/resolution;
    frequency_scale = 26.9e6 : resolution : 27.4e6 - resolution;
    for i = 1:n_steps
        diff = abs(frequency_scale(i) - f_in);
        if(diff < resolution/2)
            scaled_f = frequency_scale(i);
        end
    end
end

function weighted_average = calc_weighted_average(input,f)
        % Weighted average code for peak detection for sine input of
            ↪ system
        Average = 0;
        S = sum(input);
        for i = 1:length(input)
            Average = Average + (C(i)/S) * B(i);
        end
        fixed_part = fix(Average);
        fractional_part = Average - fix(Average);
        weighted_average = f(fixed_part) + fractional_part*(f(fixed_part +
            ↪ 1) -f(fixed_part));

end

function f_estimated = estimate_frequency(polynomial,f_axis)
        %frequency estimation by finding max value
        f_max = max(polynomial);
        f_estimated_location = find(polynomial == f_max);
        f_estimated = f_axis(f_estimated_location) + 26.9e6;
end

function [B,C]= find_peak(A,search_size,percentage)
        M = max(A);

        %Search only in the neighbourhood of the peak
        D = find(A == M);
        A(1:D-search_size) = 0;
        A(D+search_size:end) = 0;
        %Search only take values into account which are above the noise
            ↪ floor
        B = find(A > 0);
        C = A(B(1):B(end));
end

function output = filter(A,cutoff)
    % Unwanted peaks filter
    B = A;
    dA = diff(A);
    spike_level = 0.4*10^3;
    for i = 1:(length(dA) - 10)
        dA = diff(B);
        if(abs(dA(i)) > spike_level)
            B(i+1) = (B(i) + B(i+2))/2;

        end
    end
    output = lowpass(B,cutoff);
end
```

Bibliography

[1] Asad A Abidi. The path to the software-defined radio receiver. *IEEE Journal of solid-state circuits*, 42(5):954–966, 2007.

[2] Rik Bokhorst and Ruben van den Bos. Wireless passive weight sensor. Bachelor's thesis, TU Delft, Delft, The Netherlands, June 2020.

[3] Cagatay Candan. Fine resolution frequency estimation from three DFT samples: Case of windowed data. *Signal Processing*, 114(14):245–250, 2015.

[4] G Jovanovic Dolecek. Simple wideband CIC compensator. *Electronics Letters*, 45(24):1270–1272, 2009.

[5] Regeling gebruik van frequentieruimte zonder vergunning en zonder meldingsplicht 2015, December 2016. URL `http://wetten.overheid.nl/id/BWBR0036378/2016-12-28/0`.

[6] M Gasior and JL Gonzalez. Improving FFT frequency measurement resolution by parabolic and Gaussian spectrum interpolation. In *AIP Conference Proceedings*, volume 732, pages 276–285. American Institute of Physics, 2004.

[7] Jasper Insinger and Rogier Fischer. Weight sensor readout system. Bachelor's thesis, TU Delft, Delft, The Netherlands, June 2020.

[8] Sameer Kadam, Dhinesh Sasidaran, Amjad Awawdeh, Louis Johnson, and Michael Soderstrand. Comparison of various numerically controlled oscillators. In *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, volume 3, pages III–III. IEEE, 2002.

[9] Walt Kester. Which ADC architecture is right for your application. In *EDA Tech Forum*, volume 2, pages 22–25, 2005.

[10] Walt Kester. Take the Mystery out of the infamous formula "SNR = 6.02N + 1.76dB,"and why you should care. *MT-001 Tutorial Devices*, 2009.

[11] L. Kleczewski and S.E. Haitjema. "A passive wireless load cell for in-situ productweight and position sensing on modular plastic conveyor belts". Bachelor project proposal for Eletrical Engineering, March 2020.

[12] Nagarjun Marappa. Design of Digital Down Converter Chain for Software Defined Radio Systems on FPGA. Master's Theses. 664., December 2015.

[13] David A Rauth and Vincent T Randal. Analog-to-digital conversion. part 5. *IEEE instrumentation & measurement magazine*, 8(4):44–54, 2005.

[14] RED PITAYA STEMLAB BOARD, 2020. URL `https://www.redpitaya.com/f130/STEMlab-board`.

[15] Alex Tourigny-Plante, Vincent Michaud-Belleau, Nicolas Bourbeau Hébert, Hugo Bergeron, Jérôme Genest, and Jean-Daniel Deschênes. An open and flexible digital phase-locked loop for optical metrology. *Review of Scientific Instruments*, 89(9):093103, 2018.

[16] Papa-Silly Traore, Aktham Asfour, Jean-Paul Yonnet, and Christophe P Dolabdjian. Noise performance of SDR-based off-diagonal GMI sensors. *IEEE Sensors Journal*, 17(19):6175–6184, 2017.

[17] Jack Volder. The CORDIC computing technique. In *Papers presented at the the March 3-5, 1959, western joint computer conference*, pages 257–261, 1959.

[18] Qingxiang Zhang and Xiaoxiao Su. The design of digital down converter based on FPGA. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2012.