Design and optimization of a signal converter for incremental encoders

A study about maximizing the boundary limits of quadrature pulse conversion

ANTON DAHLQVIST

VICTOR KARLSSON
Design and optimization of a signal converter for incremental encoders

A study about maximizing the boundary limits of quadrature pulse conversion

Anton Dahlqvist
Victor Karlsson
Abstract

This project was carried out during spring 2016 in collaboration with BOSCH Rexroth, Mellansel, and aimed to investigate the possibility of implementing a converter with the ability to scale incremental pulse resolution and also convert the velocity to a 4-20mA current loop reference. An investigation on how a fault tolerance mechanism could be implemented to increase the dependability of the encoder was done and the out-coming results were implemented. The software and hardware were revised and optimized to facilitate a future encapsulation, and still keep the converter reprogrammable.

A background study was performed during 8 continuous weeks to acquire enough knowledge about possible conversion techniques, to finally derive these down to two for further testing. The final conversion algorithms were multiplication and extrapolation techniques, which would be utilized to scale the pulse signal. The background study also involved writing efficient C code and a general study about fault tolerance. With the information from the background study, two algorithms were implemented on a specially designed hardware platform. Tests were designed from the requirements given from Bosch and these were performed at a test rig with a magnetic ring encoder connected to dSPACE control desk.

A converter that met the criteria was designed and implemented. The test results showed that the most successful algorithm implemented was the multiplication algorithm, optimized with adaptive resolution, which decreases the input update rate with increasing speed. Although extrapolation caused more noise and also a static error on the signal, this is the one leaving most room for future optimizations. Dependability means were implemented which stops the converter from outputting erroneous pulses, and also to reboot the software in case of invalid inputs. Whether this made the converter fail-safe or not is difficult to tell since fail-safe is a vague term and applies different for each situation. It was concluded that the implemented fault tolerance mechanism worked though. The software and hardware were designed so reprogramming is possible even though the component is casted. This particular function was not tested since the development board did not provide access to the required pins.
Sammanfattning

Detta projekt genomfördes under våren 2016 i samarbete med Bosch Rexroth i Mellansel. Syftet var att undersöka möjligheten att implementera en konverterare med möjligheten att steglöst skala encodersignaler med olika pulstal, samt konvertera detta till en 4-20mA strömsignal. En utredning om hur felsäkerheten skulle kunna förbättras på konverteraren gjordes och resultaten från undersökningen implementerades. Mjukvaran och hårvaran reviderades och optimerades för hur denna skulle kunna gjutas in för att klara industrins hårda inkapslingskrav men samtidigt vara möjlig att programmera om.

En bakgrundsstudie utfördes under åtta veckor för att tillgodose tillräcklig kunskap om konverteringsalgoritmer för att komma fram till två stycken för implementation. De slutgiltiga konverteringsalgoritmerna som togs vidare till implementation och testning blev multiplikation och extrapolering av pulstider. Förstudien omfattade även hur skrivande av effektiv c kod kan göras då hög hastighet och upplösning kräver mycket beräkning av processorn, och en bakgrundsstudie om felsäkerhet i elektroniska system. Testfall designades för att testa önskade egenskaper och gränsvärden för konverteringarna och dessa utfördes på en testrig med en magnetencoder uppkopplad mot dSPACE Control Desk.

Acknowledgements

We would like to thank Arne Byström at Bosch Rexroth for giving us the opportunity to write this thesis. You have been supportive throughout the whole project, providing us with necessary equipment and information which have been of great help.

To our supervisor Mikael Hellgren at KTH, your advice regarding design decisions as well as your expertise in the field of electronics have been of utmost value. Thank you for all the interesting discussions that have arisen during the project and for the continuous feedback.

Anton Dahlqvist & Victor Karlsson
Stockholm 2016-06-08
Acronyms

CAN Controller Area Network.
CCW Counter Clock Wise.
CPU Central Processing Unit.
CW Clock Wise.
DAC Digital-to-Analog Converter.
EMC Electro Magnetic Compability.
FD Fault Detection.
FP Fixed Pulse.
FPGA Field-Programmable Gate Array.
FT Fixed Time.
HDL Hardware Description Language.
ISP In System Programming.
ISR Interrupt service routine.
KTH Royal Institute of Technology.
MCU Microcontroller Unit.
PCB Printed Circuit Board.
PPR Pulses Per Revolution.
PWM Pulse Width Modulation.
RISC Reduced Instruction Set Computer.
TMR Triple Modular Redundancy.
# Contents

Abstract v  
Sammanfattning vii  
Acknowledgements ix  
Acronyms xi  
Contents xiii  

1 Introduction 1  
1.1 Background ........................................ 1  
1.2 Purpose ........................................... 1  
1.3 Problem description ............................... 2  
1.4 Delimitations .................................... 2  
1.5 Method ........................................... 3  
1.6 Ethical considerations .......................... 3  

2 Theoretical framework 5  
2.1 Signals ........................................... 5  
2.1.1 Quadrature pulses .................................. 5  
2.1.2 Current loops ................................... 6  
2.2 Estimating velocity ............................... 8  
2.2.1 Fixed time ................................... 8  
2.2.2 Fixed pulse ................................... 9  
2.2.3 Mixed method ................................ 10  
2.3 Mathematical Algorithms ...................... 10  
2.3.1 Interpolation ................................ 10  
2.3.2 Extrapolation ................................. 12  
2.3.3 Multiplication ................................. 13  
2.4 Fault handling techniques ..................... 14  
2.4.1 Dependability ................................ 14  
2.4.2 Dependability Threats ........................... 15  
2.4.3 Dependability Means ........................... 15  
2.4.4 Hardware Redundancy ........................... 17  
2.4.5 Time redundancy ............................... 19  
2.4.6 Software redundancy ........................... 19  
2.5 Processor and software implementation ........ 20  
2.5.1 Software implementation ..................... 21  

3 Technical work 23  
3.1 Design proposals ................................. 23  
3.2 Hardware design ................................ 23  
3.3 Conversion software implementations ........ 25  

xiii
1 Introduction

This chapter serves as a general introduction to the project. It includes the background, the purpose, the problem description, delimitations, the project scope as well as used methods.

1.1 Background

Bosch Rexroth, Mellansel, is a global leading company distributing hydraulic motors and complete hydraulic power systems for customers all over the world. Until this point, these motors came with an optional rotary encoder mounted as an extension on the motor axis. The encoder was easy to replace for customized solutions in customers’ existing control systems where both specific resolutions and analog signals can be required.

In order to increase robustness, the old rotary encoders are now replaced with magnetic encoders embedded in the motor. The magnetic encoder setup consists of a magnetic strip attached around the motor axis and a hall sensor to detect the axis rotation. This magnetic strip have magnetic poles added so that every other pole is north and every other is south. They have a fixed distance of 5 mm between them and this together with a varying motor axis diameters causes the encoder resolution to be dependent of the motor size.

A problem that occurs with these embedded encoders is when a customer requires a specific encoder resolution, or in case of a motor replacement where the new motor can have a different resolution. Due to the mounting constraints, it is not possible to replace the encoder nor is it possible to attach another. If the customer requires an analog signal, it is possible to build customized solutions and encapsulate these. However, in case of a resolution mismatch between the control system and encoder, there are are this time no other feasible solutions than to reprogram the computer system. This requires external personnel and often entails large costs which is preferably avoided.

Instead of repogramming the computer systems or building customized solutions for every case, Bosch would like to solve this problem with an add on component attached between the encoder and control system.

1.2 Purpose

There are currently few devices that can handle step-less scaling of pulses to both higher and lower frequencies and not requiring any extra encapsulation. The purpose of this thesis is to investigate whether such a device can be implemented as a link between encoders and control systems. The reason to develop this device is to reduce the cost and effort required to replace and install motors in control systems that are programmed for other encoders.
1.3 Problem description

The thesis will investigate whether it is possible to implement a software-based signal converter, which can be used instead of reprogramming an existing computer system. This device should be able to read an arbitrary incremental encoder signal, rescale it and output the same type of signal but with a new resolution. To ensure compatibility with as many systems as possible, scaling to both higher and lower resolutions is essential. In addition to the pulse output, an optional 4-20mA analog output with scalable references should be available.

Safety and stability of the converter is crucial and must be considered since it will be mounted as a stand-alone component between the encoder and the computer system. The computer system could then, in case of a fault, lose the encoder signal from the motor and cause unexpected actions with the motor.

The device should operate in industrial environments set in various geographical locations around the world and must therefore handle large temperature variations and high humidity. In order to investigate whether the design and development of a converter like this is feasible, the thesis needs to answer the following three questions.

1. How can a software based encoder signal converter be implemented for step-less scaling of incremental pulse signals and how can the conversion algorithms be optimized to maximize boundary limits for scaling pulses to higher/lower resolutions?

2. Which failures can occur, and how can fault tolerance techniques be implemented in order to avoid human injury and machinery damage?

3. How can the software together with the hardware be optimized to meet the environmental constraints?

The thesis should first of all answer these questions with an underlying theory and then present a verification process. The outcome of this thesis will thus be a demonstrator with characteristics as similar as possible to the end product. The demonstrator should include the following functions:

- Supplied by 24 V
- Incremental differential pulse input
- Incremental differential pulse output
- 4-20mA analog output
- Be configurable for varying inputs/outputs

1.4 Delimitations

The thesis work runs for 20 continuous weeks, which of course means the project must be narrowed to fit within the time frame. This fact yielded the following delimitations in this project:

The project will only cover incremental encoders, absolute encoders will not be covered in this thesis. The thesis will moreover only cover incremental pulses, sinusoidal waves will not be considered.

The demonstrator will only be used for testing the internal functions of the converter. It will not be used for testing, or fulfill, the hardware constraints considering temperature variations and humidity although this should be considered when designing the circuits.
1.5 Method

To ensure that sufficient knowledge in the field of study was achieved before design and implementation of the converter began, the first part of the study contained a research on existing solutions and possible algorithms already tested. No such research could be found and the background study was dedicated to theoretical framework where the relevant areas, such as:

- General background study on incremental encoders
- Scaling of incremental pulses
- Extra-/interpolation methods
- Current loops
- Encoder/Component failures
- Fault handling mechanisms

were studied. The information gathered about these areas were the foundation necessary to later on be able to design the system, both regarding software and hardware. The literature study continued consistently for about 8 weeks before any design or implementation were initiated. This yielded three hypotheses on how such a converter could be designed in order to fulfill the requirements. This approach was chosen because a highly iterative process of the prototype design would be very inefficient and costly due to the time it takes to make a PCB and the cost to buy more components. Therefore an attempt of doing a few-iteration design was conducted. The hardware and software could be developed simultaneously since the processor delivered by Bosch Rexroth was already attached to a development board, which from the start allowed simpler conversion verifications.

With the prototype finished, the software and algorithms were implemented on the prototype to start the verification process. During the development phase, unit tests were continuously done to separate sections of the software. Final test cases were designed to represent the complete system’s performance, the converter was connected to dSPACE and a test environment was made in control desk for easy monitoring.

The test cases determined to represent the system’s performance were position drift, system response from zero velocity and also the transition between Fixed Time (FT) and Fixed Pulse (FP) for the current driver only. Each of these test cases was performed using different conversion rates, velocities, directions etc. to ensure system stability for various conditions.

In order to verify the tests, data collection and analysis was performed so conclusions could be drawn. In this specific case a quantitative approach was considered more suitable than a qualitative, since velocity and encoder pulses are variables that can easily be collected during tests and also are directly related to the system performance. The data collected from the tests was used in a quantitative analysis to verify the outcome.

1.6 Ethical considerations

Since the converter is an additional component to an already existing system, the performance and stability of the converter must not be inferior compared to the existing components, to ensure an unaltered safety level. One should be aware of adding an extra component to the system, instead of changing the existing software, might induce risks since this component can break.

As earlier mentioned, the purpose of the converter is to translate an encoder signal for an already existing computer system. Since encoders can be found on various systems, the ethical considerations are highly dependent on what sort of system the converter is mounted on. Some failures might lead to very dangerous situations while some will not. The component itself cannot cause harm to anyone, but a component failure, depending on what kind of system it is installed on, could lead to hazardous situations for operators and machinery damage. However, all these situations are complicated to predict since they
might be very distinguishable and therefore it becomes a new ethical consideration for each system it is installed on.

The converter will be able to increase the pulse frequency of the encoder, which by the system will be recognized as an increased resolution. This is done by software logic without actually changing the encoder’s resolution. This means a low-resolution sensor could be used instead of a better one with higher resolution. If the software in the converter is faulty or wrongly implemented this could lead to hazardous situations.
2 Theoretical framework

This chapter describes a theoretical background study done in order to gain enough knowledge to solve the problem. It covers a background study of common encoder signals, a study in how velocity best can be approximated from an incremental encoder, mathematical algorithms for frequency scaling and last a study in fault handling mechanisms in electronic devices. A description about encoders in general can be seen in Appendix A.

2.1 Signals

The communication between encoder and other devices in a control system is done via some kind of communication. The most common signal is the quadrature pulse which is generated by the photo detectors in an optical encoder, or by interpolation of the magnetic fields in a magnetic. Other common signals from encoders are the current loop, where a current between two fixed values acts as a reference of the velocity, and the voltage reference which works as the current loop but with voltage as reference. Encoders with internal microprocessors are sometimes also available to deliver the encoded position or velocity via a communication protocol e.g. Controller Area Network (CAN), RS432 or Ethernet. Since quadrature pulses and analog current loops are required for the converter, these will be further described below in this section.

2.1.1 Quadrature pulses

The quadrature pulse are according to National Instruments [1] the most common output from incremental encoders and contains an A and a B pulse, where the B pulse are placed 90 electrical degrees after the A pulse. Other incremental encoder may also give an index pulse or Z pulse which indicates a reference position on each lap to enhance performance in position measurements. A picture over how these are related to each other can be seen in fig. 2.1.
2. Theoretical framework

Figure 2.1. Descriptive picture about how the quadrature pulse is composed and how resolution can be increased by counting several edges.

An encoder with only one pulse, would be useless since this would not give any information about the rotation direction. With a phase shifted \( B \) pulse one can determine the direction whether the \( A \) or \( B \) pulse leads, i.e. rises before the other. If the \( A \) pulse leads the encoder is rotating clockwise and if the \( B \) pulses leads the encoder is rotating counter clockwise \[1\]. Thus, by counting the pulses and monitoring the phase of \( A \) and \( B \) one can derive both direction and position which further can be derived to both velocity and acceleration.

The pulse described above, are the output of a single ended encoder where \( A \) and \( B \) have its reference to ground. In noisy environments with long cables, this setup can affect the signals negatively. To avoid this another type of encoder is also available, the differential encoder. In addition to the \( A \) and \( B \) signal, the differential encoder also supplies the inverses \( A' \) and \( B' \). When using inverse pulses both cables can constantly deliver a known voltage which yields a voltage difference instead of one signal referred to ground. This setup has two levels, high and low, i.e when \( A \) is high \( A' \) is low and when \( A \) is low \( A' \) is high \[1\]. Noise that disturbs the signal in single ended encoders can here be neglected since the differential between the two cables are used as reference.

2.1.2 Current loops

Analog signals, or current loops are common when it comes to controlling or monitoring processes in industrial applications such as pressure, temperature, flow and velocity \[2\]. There are a few different methods to read these analog signals depending on the industry.

The current loop has many advantages such as long transmittance distance, signal stability and low susceptibility to noise. The reason why the current loop is a preferable choice when dealing with long transmittance distances, is because the voltage drop that is produced when sending currents over long distances does not affect the loop as long as the transmitter and loop supply can compensate for the drops. Although the resistance in the wires only measure up to a few milliohms per meter, it quickly adds up when the distances get greater since the resistance is proportional to the length of the wire, as seen in eq. (2.1) \[3\],

\[
R = \frac{\rho \cdot l}{A} \tag{2.1}
\]
where $R$ is the resistance in the wire, $\rho$ is the resistivity, $l$ is the wire length and $A$ is the cross sectional area of the wire. Even though the resistance is also dependent on the temperature, an assumption of a fixed temperature is usually made for simplification purposes.

The current loop circuit usually consists of four main components:

- Sensor
- Transmitter
- Loop power supply
- Receiver/Monitor

This sort of current loop setup is called a loop-powered application, where all the elements are connected in series in a closed loop and have an external power source. The setup can be seen in fig. 2.2

![Figure 2.2. Current loop.](image)

The sensor reads the physical parameter that is being measured and outputs a corresponding voltage proportional to the signal. This voltage signal is later on amplified and converted into a proportional dc-current signal by the transmitter and sent out into the closed loop system. The current is read as a voltage drop across a known resistance and according to Ohm’s law, the current can be obtained by dividing the voltage with the resistance. A common value for the resistor that the receiver uses for measuring the voltage is around 250Ω [2, 4, 5]. This, once again according to Ohm’s law, means that the current level of 4-20mA will correspond to a voltage level of 1-5V, which will be the input signal of the receiver. If the components in the system do not have their own power supply, the external loop power supply is utilized to ensure a stable dc-voltage. Since a lot of electronic devices that can be found in industrial applications use a 24 V power supply, it is very common as a current loop supply as well.

As previously mentioned, measuring current gives better noise immunity than voltage due to the fact that high impedance devices is often utilized in the systems to avoid signal loss as a result of the voltage drop [2, 5]. Since there are often a lot of systems in the surroundings such as motors, radio transmissions etc. that emits noise, it is very important to try and make the signal less susceptible to it and one way is to use a current loop.

The most common current loop uses a 4-20mA signal to represent the measured parameter, where 4 mA represent the lowest end of the measurement and 20 mA the highest. There is no universal meaning of every current level, 6 mA can represent one thing in one system and a completely other thing in another, it depends on the application or the manufacturer. The signal that indicates a system error such as a faulty connection or similar is often represented by a 0mA current [5, 6]. But caution has to be taken, since there are systems that use a 0-20mA signal instead of 4-20mA.
2. Theoretical framework

Signal conversion and transmission

As previously mentioned, the transmitter receives a signal from the sensor, processes and amplifies it, and then outputs a corresponding current. Since there is such a wide range of applications that use sensors, various types of sensors with different properties has to be available to satisfy them all. This also means that they can have various types of output signals such as a dc-voltage or a voltage pulse and therefore the transmitter must be able to convert these signals into current. Examples of such converters can be seen in table B.1.

There are several ways to transmit the signal. One way is to use a Digital-to-Analog Converter (DAC), which does exactly what it says; transforms a digital signal into voltage. If there is no DAC available, another way is to use a Pulse Width Modulation (PWM) signal and a low-pass filter. The benefit of using a DAC is that the signal already is a DC-signal, which only has to be amplified and no external filter is needed as in the case of the PWM signal. A PWM signal is built on square pulses that often are transmitted in a very high frequency, which means that a low-pass filter has to be added to get rid of the high frequency pulses and get the DC-signal. This results in an attenuation of the signal, which is taken care of with an external amplifier of some sort. As a conclusion, both a DAC and a PWM can be utilized, but if there is a DAC available, this would be a preferred method due to less external components and fewer conversions.

2.2 Estimating velocity

There are two recurring methods proposed amongst papers to derive velocity and acceleration from quadrature pulses, even though the naming sometimes differ. The first method is based on counting the pulses gathered during a fixed time, which is referred to as the FT method. The second are instead counting the time elapsed during a certain amount of pulses, this is referred to as the FP method [7–10].

2.2.1 Fixed time

FT as Baser et al. [7] choose to call it can also be referred to as the M method [9] or Frequency measurement/counting [8, 10]. The idea with FT is to define a fixed sample time, $T_s$, wherein the rising pulses can be counted and divided by the chosen sample time. The velocity is then considered to be constant within the observation time and can be calculated as eq. (2.2) [10],

$$\omega = \frac{d\theta}{dt} \approx \frac{\Delta \theta}{T_s} = \frac{2\pi \cdot \Delta N}{N_p \cdot T_s} [rad/s] \Rightarrow \frac{60 \cdot \Delta N}{N_p \cdot T_s} [RPM]$$

(2.2)

where $\omega$ is the angular velocity, $T_s$ the chosen sample time, $N_p$ is the number of pulses per revolution e.g. the encoders Pulses Per Revolution (PPR) and $\Delta N$ is the number of pulses observed within the considered sample time. As an effect of measuring the velocity within a fixed time frame, where it is not possible to have perfect synchronization between the pulses and the time frame, a quantisation error is induced. This quantization error [10], or the velocity resolution as Tanaka, Nishi, and Ohnishi [9] calls it, can be described as

$$\Delta \omega = Q = \frac{2\pi}{N_p \cdot T_s} [rad/s] \Rightarrow \frac{60}{N_p \cdot T_s} [RPM].$$

(2.3)

As seen in eq. (2.3), the quantization error $\Delta \omega$ is not dependent on the current velocity but only the resolution of the encoder and the chosen sample time. The relative error can as stated by Petrella et al. [10] be described as

$$e\% = \frac{2\pi}{N_p \cdot T_s \cdot \omega} \cdot 100.$$  

(2.4)
Since a quantization error is an unwanted phenomena of speed measurements, it is often desirable to reduce it as much as possible. This can as seen only be done by either having a more accurate encoder or increasing the sample time, which also leads to longer delays and decreased bandwidth [10]. In case of a not sufficient resolution, filtering techniques can be used to get smoother estimates. A few of those methods proposed by Baser et al. [7] will be presented below starting with the conventional low pass filter.

1. A Low Pass Filter could be used in order to acquire smoother velocity estimates, and remove the quantization error, using the first and second order numerical differences of the position. A first order transfer function for such a filter could be described as

\[ H(s) = \frac{\omega}{s + \omega} \]  

(2.5)

where \( \omega \) is the desired cut off frequency. There are other alternatives as well for higher order low pass filters such as the Butterworth filter. Such a filter can easily be implemented using filter design tools, for example Matlabs built in function “butter” [11]. However, using low pass filters will cause a lag in the system which sometimes might not be tolerable for the control loop [10, 12]. Other well known problems the low pass filter have are also attenuation, low band width frequency and the need of an individual tuning for each application [7].

2. The Kalman filter can according to Baser et al. [7] be used as a optimal state estimator whenever the dynamics of the system are somewhat known i.e. the system behavior and possible disturbance are predictable. The Kalman filter is because of this according to Jin and Pang [13] an excellent solution in constant speed cases where the signal noise is constant and systematic. The problem is, whenever the velocity changes the noise levels are also changing which the Kalman filter can not predict. Shaowei and Shanming [14] proposed an improved covariance matrix which enhanced the performance of the Kalman filter for more precise velocity prediction in case of varying noise.

3. The modified state observer can, in opposite of the Kalman filter, estimate the velocity and acceleration without a system model. This algorithms main purpose is to reduce quantization noise and remain large bandwidth. The filter must be carefully tuned to satisfy a wide velocity range and even though the tuning is successful, the filter is not always giving satisfactory results [7].

4. Most of the proposed methods requires a compromise between noise reduction, delay, accuracy and other attributes such as computational load. Janabi-Sharifi, Hayward, and Chen [12] proposed a method which give maximum accuracy together with a minimum velocity error. This method is based on active alternation of the sampling time depending on the current running conditions i.e. the motor velocity. The lower bound of the window is set by the noise reduction and precision whilst the upper is limited by the system delay. Subsequently, a high motor velocity requires a short sample time for fast calculation, low delay and reliable estimates while a slow motor speed needs a longer sample time to produce more precise estimates of the velocity.

One advantage of FT is the easy implementation on a Microcontroller Unit (MCU). The FT only requires a counting of \( \Delta N \) in eq. (2.2) together with an interrupt at the chosen measuring window. Incremental position are in most cases read by quadrature decoder hardware which can be found inside microprocessors but also as external components.

### 2.2.2 Fixed pulse

The second method, FP [7], which also can be referred to as Period measurement [10] or T method [9] is instead of counting pulses within a specific time band, counting the time elapsed between a certain amount of pulses. The velocity can then be estimated by

\[
\omega = \frac{d\theta}{dt} = \frac{\Delta \theta}{n \cdot Tcl} = \frac{2\pi}{Np \cdot n \cdot Tcl} [\text{rad/s}] \rightarrow \frac{60}{Np \cdot n \cdot Tcl} [\text{RPM}]
\]  

(2.6)
2. Theoretical framework

where $Np$ is the resolution of the encoder, $n$ is the number of pulses the timer should count and $T_{cl}$ is the timers value after $n$ pulses passed [10]. Equation (2.6) shows that the sampling time is, in opposite to FT where the user choose the velocity update frequency, dependent on the current motor speed. This can be described as

$$T_u(\omega) = \frac{2\pi}{Np \cdot \omega}$$

(2.7)

where $T_u$ denotes the current update time of new velocities. As seen in eq. (2.7), the span between two pulses is increasing with reduced motor speed. Since the encoder samples at a variable frequency as seen in eq. (2.7), at some certain level, a new velocity for each control cycle can not be guaranteed. Considerations about how this could affect the control system it will be used in must be consider. Another phenomena that can occur is that $T_{cl}$ will become longer in case of a lower motor speed. This could cause the timer to overflow which most certainly will provide unreliable values [10].

The relative error for FP when $n = 1$ can be described as

$$e\% = \frac{T_{cl} - T_{cl} \cdot 100}{\frac{2\pi}{Np \cdot \omega}}$$

(2.8)

where $e\%$ is the relative error.

An implementation of the FP method on a micro controller requires a high frequency clock for timing the duration between pulses. It may also require some algorithm or hardware to handle possible timer overflow and a filter to reduce noise induced by a non ideal encoder and other measurement errors.

2.2.3 Mixed method

As one can see in eq. (2.4) and eq. (2.8), the relative error is proportional to the current rotational speed. The FP method is more accurate at lower speed whilst the FT is in opposite better at higher speeds. This means there have to be some point where the relative errors of the two methods are the same. One commonly proposed method [10] is to combine both FP and FT, capture the encoder speed and always use the method with the lowest error.

2.3 Mathematical Algorithms

One major task in this thesis is to evaluate which mathematical algorithm that is most feasible for frequency scaling of encoder signals. At an early stage in the project the considered methods were narrowed down into three fields. These were Extrapolation, Interpolation and pure multiplication of the current pulse rate. This section presents a prestudy about these mathematical algorithms.

2.3.1 Interpolation

Interpolation is a mathematical method used to create new data points between already known values in for example tables or functions and is commonly used in engineering and mathematics. This approximation is made by creating a polynomial of a suitable order, that represents the already known data points.

Since functions can vary in complexity due to inconsistency between the data points, some might be hard to do a good approximation of, using interpolation. Therefore an assessment of the resulting interpolation polynomial has to be done, to see which kind of interpolation method is the most suitable and if the result is sufficient. Even though the resulting polynomial might not be a very good representation of the actual function, it can be good enough for some applications.
When dealing with simpler functions, i.e. functions of lower order, a linear interpolation method can be used to extract the values at the desired points. While more complex functions might need a more complex interpolation method with a higher order, such as polynomial or spline interpolation.

There are, as earlier mentioned more than one interpolation method. A few common methods are:

- Linear interpolation
- Polynomial interpolation
- Spline interpolation

**Linear interpolation**

Since linear interpolation generally only uses two data points and between those, creates a linear approximation to represent the function, it is more suitable for functions of first order and data with consistency.

If linear interpolation is used on a dynamic, complex function, the resulting approximation can be very bad if there are an insufficient amount of data points. In that case, a polynomial interpolation method might be more suitable. To retrieve the approximated value in the chosen point using linear interpolation, eq. (2.9) is used [15, 16].

\[
\ell(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)
\]

where \(\ell(x)\) is the approximated value in the chosen point \(x\), \(y_1\) is the function value in the point \(x_1\) and \(y_2\) is the function value in point \(x_2\).

**Polynomial interpolation**

Lagrange polynomial is one type of polynomial interpolation applied in numerical analysis and is commonly used when a function of higher order is to be approximated. When using polynomial interpolation there is a trade-off between accuracy in the data points and unwanted, oscillatory behavior between the points. The oscillations are a possible outcome from using a higher degree of the polynomial, which in turn is a result of using a higher amount of data points. More data points results in a greater oscillation and this phenomenon is called Runge’s phenomenon [17].

The Lagrange polynomial is a unique polynomial \(p(x)\) of degree \(\leq n-1\) with the corresponding function values \(f(x_j)\), where \(j = 1, ..., n\) in the given \(n\) points [17, 18], which can be described as:

\[
p(x) = \sum_{i=1}^{n} f(x_i) \ell_i(x)
\]

where

\[
\ell_j(x) := \prod_{i=1, i \neq j}^{n} \frac{x - x_i}{x_j - x_i}
\]

**Spline interpolation**

According to Pohl [15], spline interpolation can be described as a type of piecewise interpolation method, which means that there are specific functions between every two points. So instead of creating one single polynomial that should fit all the considered data points, a new one is made for each new gap between points, this way Runge’s phenomenon is avoidable.

By using the cubic spline method, which is a method of third order, a smooth transition at every point is achieved and therefore is the first and second derivate continuous in those points. The cubic spline
method can be based on Hermite's interpolation polynomial in every subinterval \( x_i \leq x \leq x_{i+1} \), where \( i = 1, 2, \ldots, n - 1 \) and a benefit of using this method is that the derivatives do not have to be known, they can be estimated instead. The function derivative is defined as \( k_i = y'(x_i) \), the difference in x-direction as: \( h_i = x_{i+1} - x_i \) and differences in y-direction as: \( \Delta y_i = y_i + 1 - y_i \). When the derivatives are approximated, they are put into the \( n - 1 \) polynomials. The polynomial can be described as eq. (2.12) below.

\[
P_i = c_1 + c_2(x - x_i) + c_3(x - x_i)^2 + c_4(x - x_i)(x - x_{i+1})^2
\]  

\[\text{(2.12)}\]

where

\[
c_1 = y_i, \quad c_2 = \frac{\Delta y_i}{h_i}, \quad c_3 = \frac{(k_{i+1} - c_2)}{h_i^2} \quad \text{and} \quad c_4 = \frac{k_i - c_2}{h_i^2}.
\]  

\[\text{(2.13)}\]

A simplification of the polynomial can be made by defining the polynomial as straight outside \( x_1 \leq x \leq x_n \), which means that the second derivative is zero at \( x = x_1 \) and \( x \geq x_n \), these are called natural terms. Using these relationships, the tridiagonal equation is given by eq. (2.14).

\[
\begin{pmatrix}
2h_1 & h_1 & 0 & 0 & \cdots & 0 \\
h_2 & 2(h_2 + h_1) & h_1 & 0 & \cdots & 0 \\
0 & h_3 & 2(h_3 + h_2) & h_2 & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & \cdots & h_{n-1} & 2(h_{n-1} + h_{n-2}) & h_{n-2} \\
0 & 0 & \cdots & 0 & h_{n-1} & 2h_{n-1}
\end{pmatrix}
\begin{pmatrix}
k_1 \\ k_2 \\ \vdots \\ k_{n-1} \\ k_n
\end{pmatrix}
= \begin{pmatrix}
b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n
\end{pmatrix}
\]  

\[\text{(2.14)}\]

where

\[
b_i = \begin{cases}
\Delta y_i & \text{if } i = 1; \\
\frac{h_i}{h_i} \Delta y_i + \frac{h_i}{h_{i-1}} \Delta y_{i-1} & \text{if } i = 2, 3, \ldots, n - 1; \\
\Delta y_{n-1} & \text{if } i = n.
\end{cases}
\]  

\[\text{(2.15)}\]

Integration

Interpolation is one of the investigated methods in this project since it can be used to approximate a polynomial and find values that are not previously known. Even though the primary purpose of interpolation is not to find values outside the known interval, it might be valuable since it can be used to find the polynomial at the data points, so a better approximation of the upcoming data point can be made. Using an interpolation method will give a delay since previous values are considered when calculating the polynomial, so how big the delay between the input and output signal of the system gets when implementing it on the processor has to be evaluated.

2.3.2 Extrapolation

Knowing a set of variables \( x_1, x_2, x_3, \ldots, x_N \) within a function \( f(x) \), to find \( f \) for an arbitrary \( x_p \) some mathematical algorithm must be used. In the case where \( x_1 < x_p < x_N \) interpolation can be used which has already been covered in the previous section. In the case where \( x_N < x_p \), i.e. out of the known range, interpolation is not feasible as method and thus extrapolation must be used.

There are many various methods of extrapolation, but common for all of them is to estimate values outside a certain measuring range by using known values within the range. Depending on which method used, these values are treated differently in order to get as accurate approximations as possible. Some of these various extrapolation methods will be covered below.
2. Theoretical framework

Linear extrapolation

The most basic extrapolation method is the linear extrapolation method, which in similarity to the linear interpolation, assumes that the function behaves linearly between the two measured values. Having the known values \((x_k, y_k)\) and \((x_{k-1}, y_{k-1})\) which satisfies \(x_{k-1} < x_k\) a future value \(y(x)\) can be predicted as

\[
y(x) = y_{k-1} + \frac{x - x_{k-1}}{x_k - x_{k-1}} (y_k - y_{k-1}).
\]

2.3.3 Multiplication

One of the available methods that can be used to scale signals, both upward and downward, is based on simple multiplication of the input signal. There are converters on the market today that use one kind of multiplication method [19], which also can be seen in the comparison done in table B.2 in Appendix B. Even though this converter does not satisfy all the requirements for the converter that is supposed to be developed in this project, some of its properties, such as the conversion method, might be of interest.

The basic function of the existing converter is based on programmable factors that the input signal is multiplied by, creating an output signal that can be varied within a pre-defined range. This enables an encoder or another quadrature pulse source to fit with a before incompatible computer system. These programmable factors can be adjusted within the range of 0.005 to 9.9999 and are then multiplied by the input signal as seen below in eq. (2.17).

\[
f_{out} = f_{in} \frac{Factor_1}{Factor_2}
\]

where \(f_{in}\) is the frequency of the input signal, \(f_{out}\) the frequency of the output signal. \(Factor_1\) as well as \(Factor_2\) are the adjustable factors used to change the frequency of the signal. Since both the factors can be adjusted within the range of 0.005 to 9.9999, a total conversion factor (\(Factor_1/Factor_2\)) between 0.0005 and 1999.98 can be achieved. This method of using two factors enables the user to choose from a wide spectrum of resolutions.

Since this method is based on multiplication of the input signal, it is necessary to read the signal value before the output value can be calculated by multiplication. Even though signal processing like this can be very fast, some kind of delay will be introduced between the input and output signal. To be able to know the frequency of the pulses, the rising and falling edges of the pulse must be read, and after this, the signal can be multiplied by the factors to get the correct frequency. So implementation of this method in a control system with strict requirements on low delay could cause problems.

The converter, as seen in fig. 2.3, delivers true pulses but not true frequencies, which might not be a problem in positioning applications but will induce a problem when used in speed control applications [19].
2. Theoretical framework

2.4 Fault handling techniques

Dubrova [20] defines fault tolerance as "the ability of a system to continue performing its intended functions in presence of faults". This could be faults in either hardware components, a software bug or other unforeseen events. This section will present common concepts within fault tolerant systems and the means to design and attain such a system.

2.4.1 Dependability

The utopia of a fault tolerant systems is what one could call a dependable system where the term dependability measures the ability to deliver a service that can be trusted [21]. Dependability is a rather big and complex term so normally one says that dependability got several attributes. Regarding to Dubrova [20], the three main attribute of dependability are reliability, availability and safety which all are explained further below. Other examples of dependability attributes are integrity and maintainability [21] those will not be further explained in this report.

Reliability

Reliability is a measure of the systems ability to continuously deliver correct service [21]. Dubrova [20] are further defining it as the reliability $R(t)$ is the probability that a system operates without failure at a time $t$ if it operated without failure at $t = 0$. High reliability is crucial in application where the system must work without interruption as in a pacemaker [20]. Other situations where reliability is important is when it is impossible to access the system for maintenance, e.g molded components or a space satellite. As an opposite to reliability, unreliability measures the probability for a system failure within the same time span. The unreliability $Q(t)$ and the reliability are related to each other as

$$Q(t) = 1 - R(t). \quad (2.18)$$
2. Theoretical framework

Availability

Not many systems operates continuously without any interruption at any time, and thus is the reliability complemented with the attribute availability. The availability $A(t)$ is the systems readiness to instantly deliver correct functions at the time $t$ and is referred to as the point availability. However, its often necessary to determine the availability during a certain period, for example during a products lifetime. This is the mean availability and is defined by Dubrova [20] as

$$A(T) = \frac{1}{T} \int_{0}^{T} A(t) \, dt \quad (2.19)$$

where $A(T)$ is the average value of the point dependability over the interval $T$. This equation is used to calculate the more interesting value steady state availability which for a system can be defined as

$$A(\infty) = \lim_{t \to \infty} \frac{1}{T} \int_{0}^{T} A(t) \, dt \quad (2.20)$$

which defines the total downtime of a system during its lifetime. This value is used to calculate the systems downtime per year, which is a common unit to measure availability.

Dubrova [20] gives a good example of a system where the availability is very important, a telephone network. The user expects the telephone to work flawless when he or she make the telephone call, i.e. high availability. However, the fact that the network can be down a few hours per year or a few minutes per month, is often considered acceptable.

Safety

Safety is similar to reliability, but safety failures are divided into two categories, fail-unsafe and fail-safe. Safety $S(t)$ are by Dubrova [20] defined as the probability of the system either performing correctly or incorrectly but in a fail safe manner. Fail-safe manner means the system has no probability of causing any hazards as human injury or environmental disasters.

2.4.2 Dependability Threats

The threats to a systems dependability are covered by the three categories faults, failures and errors. A fault is a defect or flaw in either hardware or software. This could mean a software bug or a broken component on a circuit board. An error is a deviation from a normal behaviour in terms of computation and signal handling and is often the result of a fault. A failure is the absence of a task which should have been executed i.e. the update of a control reference which should have been updated every 30ms [21].

2.4.3 Dependability Means

Avižienis et al. [21] talks about four means to attain dependability and security of a system. A brief summary and explanation of these methods will be presented below.

Fault Prevention

Fault prevention is the idea of preventing faults before they occur and is closely related to a systems dependability and security. However this is by some not seen as a specific fault handling technique, but rather a part of general engineering and a obvious aim in every hardware and software development process [21]. Fault prevention is achieved by continuous quality control during the whole design process and are slightly different depending on what the current system involves. For hardware, fault forecasting
could include having design reviews and component testing whilst a software rather could involve a
structured programming, modularization and formal verification techniques [20].

Fault Tolerance

Fault tolerance is achieved by some kind of redundancy and aims to achieve a system that functions cor-
rectly when faults occur, by using either masking or detection (which is followed by location, containment
and recovery). However, only redundancy is not sufficient to make a system fault tolerant, monitoring is
also required to monitor results and choose the proper alternative [20].

Fault masking is the process of making sure that the system does not output any faulty results. This is
done by either correction of the erroneous value or other compensation techniques, for example majority
voting. Fault detection on the other hand is to determine whether a fault has occurred within the system.
The detection is followed by the fault location which finds the origin of the fault. The isolation is the
idea of preventing the fault from causing damage to other parts of the system and finally the recovery
handles the fault by for example replacing the broken component with a redundant [20].

Fault Removal

Fault removal is the idea of reducing the amount of faults in the system and can be performed both
during the products development and use phase.

During the development phase, the fault removal process consists of three steps; verification, diagnostics
and correction [20]. The verification process aims to find out whether the system follows the expected
behaviour which should have been stated in an initial design process as verification conditions. If this
is not the case, the process continues with the two remaining steps. The diagnostics aims to find what
hindered the system from passing the verification and the following correction is supposed to remove
the fault. This process is iterative and as soon as the necessary corrections has been implemented, the
verification process begins again, to ensure that the corrections did not give any unwanted behaviour, and
continues until no more faults are found. The verification step can be categorised whether they involve
execution on the system or if they lack actual execution [21].

Verification without any execution is called static verification and involves either static analysis (e.g
inspections, data flow analysis or compiler checks) or theorem proving. Static verification can also be
evaluated on a model of the system instead of the system itself, this referred to as model checking
[21].

Dynamic verification involves execution on the actual system and the input can be either symbolic in
symbolic execution or actual in verification testing, more known as testing. Deterministic tests have a
predefined selected input and random, or the alternative name statistical testing, have a random input
within a specific range.

Even though verification methods are easily classed into various categories, one should know that these
methods can be combined and must not be used as stand alone methods [21]. This might even lead to
better testing since all methods have its strengths and weaknesses. Having a good method to evaluate
whether or not the system fulfills the verification requirements is just as important as having a good
verification method.

Fault removal during the use phase is mainly a maintenance of the system which can be divided into two
main fields. Corrective maintenance is the idea of removing faults that have caused one or several errors
in the system. Preventive maintenance is the idea of removing faults before these have caused any fault
[21].

Fault Forecasting

Fault forecasting are techniques for estimating the number of faults in a system, future occurrence of
faults as well as possible consequences of them. It is done by performing a system analysis with respect
2. Theoretical framework

to fault occurrence and can be done with either a qualitative or quantitative approach [21]. A qualitative analysis aims to identify and rank the events that could lead to any system failure, whilst Avižienis et al. [21] describes the quantitative analysis as “it aims to evaluate in terms of probabilities the extent to which some of the attributes are satisfied; those attributes are then viewed as measures”. Avižienis et al. [21] also claims the most commonly used methods for quantitative analysis are modelling and evaluation testing.

2.4.4 Hardware Redundancy

When the more intuitive methods to increase a hardware systems dependability, such as the use of more reliable components, have been drawn to its extent other methods must be taken into account. Hardware redundancy is today considered as the only method available to fulfill such a task [20]. The main idea behind hardware redundancy is to have two or more of the component that redundancy is requested on, e.g. processors, transistors, CAN modules. Since hardware redundancy essentially means adding extra components to the circuit it also comes with some drawbacks. The development costs increases as well as the end cost for the product, in addition to this, the power consumption and system weight are also often increased. Hardware redundancy can be categorized depending on how the redundant components are used. These three methods are explained below [20].

Passive redundancy

Passive redundancy are masking faults instead of detecting the actual fault, this means only correct values will pass as output but one will not know about the fault. The most common passive redundancy is the Triple Modular Redundancy (TMR) and this implies that all redundant component are activated and given the same task for calculation. When this is done their answers is compared by a majority voter which ensures masking an error from a faulty component since the two correct answers are in majority. TMR will not be able to detect faults which occurs in more than one component since the fault suddenly becomes in majority of the answers. For this case more redundant components must be used which is referred to as N-modular redundancy where N components are redundant [20].

As earlier mentioned, TMR is a system working as long as a majority of the components works properly, hence can a system with the three redundant components R1 R2 R3 be described as

\[ R_{TMR} = R_1 R_2 R_3 + (1 - R_1) R_2 R_3 + R_1 (1 - R_2) R_3 + R_1 R_2 (1 - R_3) \]  \hspace{1cm} (2.21)

where \( R_{TMR} \) is the redundancy of the redundant system. \( R_1 R_2 R_3 \) covers the case where all three components work, \( (1 - R_1) R_2 R_3 \) covers where \( R_1 \) is faulty but the other works and etc. If we assume \( R_1 = R_2 = R_3 \) the reliability can be written as

\[ R_{TMR} = 3R^2 - 2R^3. \]  \hspace{1cm} (2.22)

A comparison between a TMR system and a non redundant system can then be done, such is presented in fig. 2.4
As seen in fig. 2.4, the implementation of TMR will not always ensure higher dependability and reliability of the system. If solving the equation \( R = 3R^2 - 2R^3 \) one can see that there is an equilibrium at \( R = 0.5 \), this proves that for components with less reliability than 50\% the TMR actually makes the system less reliable. These calculations where all made for the case where the voter works perfectly, since this rarely is the case, the voter’s reliability must also be concerned. Equation (2.22) can for this case be rewritten as

\[
R_{TMR} = (3R^2 - 2R^3)R_v
\]  

where \( R_v \) notes the reliability of the voter. As we can see from eq. (2.23) is the voters impact decreasing the total reliability even more.

**Active redundancy**

Active redundancy is the idea of detecting the fault and performing the needed actions instead of just masking the fault as a passive redundancy would. Dubrova [20] presents three common active techniques, duplication with comparison, standby and pair-and-a-spare.

The duplication and comparison method is mainly two modules which operates simultaneously with a comparator comparing their results. If the results are unequal a fault is indicated, which means it cannot detect faults that appears in both modules. This method is only intended to detect faults and will hence not do any actions to return the system into a properly working state [20].

In standby redundancy is one out of \( N \) components active, the others are waiting on standby without actually executing any data. If a Fault Detection (FD) unit finds a fault on the active module, one of the spares is activated and the computational task is distributed to the spare module. Such redundancy can be split into two categories, either hot or cold standby. The hot standby means all spare units are powered up and always ready to as fast as possible continue a faulty components task which minimizes the downtime of the system. Cold standby on the other hand, are not powered up and configure until a fault is detected. This means there is a longer delay until the system is up and running again which can be accepted in certain systems due to the lower power consumption. One important consideration when implementing a standby redundant system is the fact that even the FD units have a certain reliability. Using bad FD units could therefore lead to a redundant system performing worse than a single component [20].
Pair and spare is a sort of combination between the two earlier mentioned active methods. The difference is that instead of one active module, there are two, with a comparator comparing the results. If a deviation is detected by the comparator, it analyses the results from the FD unit in order to find out which is faulty and replaces it with a spare module [20].

**Hybrid redundancy**

Hybrid redundancy is combining both passive and active redundancy to create a better and more reliable system. It uses fault masking as passive redundancy to ensure erroneous results cannot be produced by the system. Besides this, FD units are used to locate the fault so the system can recover from the faulty state by replacing the faulty component [20].

**2.4.5 Time redundancy**

Time redundancy is a method which can be considered both as a fault detection method as well as a redundancy method. The idea with time redundancy is to recalculate the computations two or more times and then compare the results. If a result differs from the stored one a fault is detected, this means if the calculations are done twice a fault can only be detected, if it is done three times, it can also be corrected with the use of a voter. Time redundant system benefits from being able to detect whether a fault is permanent or transient. If it appears only one time it can be considered transient, for the case where it appears several times it must be considered permanent [20].

**2.4.6 Software redundancy**

There are mainly two types of software redundancy, single-version and multiple-version. Single-version where only one software is used which has fault detection and fault recovery implemented to handle a fault. Multiple-version uses several software components which all are diversely designed to make sure they wont produce the same error.

**Single-version technique**

The single version adds extra features to the software which enables it to detect and locate faults within it and prevents the errors from affecting the system in a bad manner.

The fault detection’s purpose, is as earlier mentioned, to evaluate whether a fault has occurred or not within the system. Dubrova [20] claims that most single-version fault tolerance technique has some kind of acceptance check to detect a fault. This test is evaluated on the computed result from the software where a failed test means a fault has occurred. A acceptance test must be executed faster than the program itself to be effective or else it might be a better option to create another version of the software which runs in parallel and compare the results. Dubrova [20] mentions a few common acceptance tests in her book, these are;

*Timing Checks* - Can be used on programs which has timing constraints. Deviating timing or a missed deadline could indicate an error.

*Coding Checks* - Checks the validity of the code and computations. One approach is to do the same calculations backwards and ensure the input still is the same.

*Reasonableness Checks* - Uses the known properties for a system to assure the data is reasonable. For example, a duty cycle can only vary between 0% to 100%.

*Structural Checks* - Checks the known data structures, e.g. counting the elements in a list.

After a fault is detected, the aim is to contain the fault and prevent it from propagate through the system so a fault recovery sequence can be initiated.
2. Theoretical framework

Checkpoint and restart is commonly used as a fault recovery technique which is based on simply restarting the software execution from a checkpoint, i.e. an earlier snapshot of the system. Since software faults often are design faults activated by some unforeseen event as division by zero or pointing to unallocated memory, the checkpoint and restart occur often enough to restore the system to a normal state. Checkpoints can be either static or dynamic. The static checkpoint is a snapshot of the program before the execution of the program starts, whereas the dynamic checkpoints is taken dynamically along the execution. Static checkpoints are most suitable for small systems which do not have long execution times, for these systems the dynamic is a better choice [20].

2.5 Processor and software implementation

There are various kinds of platforms and processor types since different types of systems require different kinds of functions, computational power etc. For systems that are not in need of a computer, which has a lot of different functionalities, an embedded system with a microcontroller is often utilized since they are constructed to serve a specific purpose. In the area of embedded systems, there are different kinds of controllers and platforms to realize the needs, and two types that can be of interest for this project are processors based on Reduced Instruction Set Computer (RISC) architecture and Field-Programmable Gate Array (FPGA) chips.

RISC

RISC is a common type of processor architecture using a simplified instruction set with register-to-register arithmetic instructions. Only fixed size, load and store instructions are used to access the memory while other instructions operate only between the registers. This means that a separation is made between the transfer of information and operation on it. This simplifies the design, reduces complex addressing, and thus facilitates an increment in speed [22]. RISC-processors uses an operation method called pipelining to ensure high performance. In the concept of pipelining, the processor operates on different steps of the instructions in parallel, thus increasing the efficiency among operations [23].

These simple and fixed sized instructions make it possible for RISC-processors to achieve high performance and yet maintaining a low cost. The small and simple circuits enables higher clock rates since they often have shorter wires and fewer transistors, leading to a lower parasitic capacitance and therefore higher speed. Something else that is a typical RISC-characteristic is the regularity between operations, making it possible to use the same subsystem for multiple operations and therefore enabling a reduction in the number of circuits.

FPGA

The FPGA is, as the name states, possible to re-program in the field, i.e. by the user and not only in the factory. The main difference between an FPGA and an ordinary microcontroller is that a microcontroller comes with a pre-defined hardware that can be utilized by programming the software accordingly. Using a FPGA, the hardware functionality is created by the user, using a Hardware Description Language (HDL) where the functionalities are constructed using building blocks, representing different signals, gates etc. This enables the user to customize this general-purpose logic device to serve and satisfy the specific needs of the user. So instead of being restricted by the number of a specific pin function at the microprocessor, for example the number of serial ports, the FPGA can be programmed so there are as many pins set to function as serial ports as necessary, the only restriction is the number of physical I/O pins on the FPGA. This makes the FPGA much more customizable than the ordinary microcontroller and therefore more function specific, since it does not have unnecessary functions which are often pre-installed at a microprocessor.

Since the hardware functionalities are directly programmed instead of programming the software, which later on controls the hardware, the execution time of the FPGA often exceeds an ordinary microcontroller’s, with clock speeds up to tens of megahertz. Another feature of the FPGA is the true parallelism,
which is realized by the processing tasks being assigned to a specific section of the chip, unlike regular processors where the tasks are executed one-by-one in a specific order. This parallelism is truly beneficial since more tasks can be executed in a shorter period of time, thus increasing the speed of the system.

### 2.5.1 Software implementation

The first aspect to consider when code efficiency and speed is essential is the choice of programming language. C is a commonly chosen language due to its low level basis, resulting in high-speed computations. Another aspect to consider is the use of the memory hierarchy in the processor, which most computer systems are based on. The hierarchy is supposed to act like a bridge between the processor and the different types of memories in the system. The different memories have different sizes and different access times, which results in different execution times depending on if the program can be executed right away in the Central Processing Unit (CPU) or if it has to be recovered in another memory bank prior to execution. “Optimizing numerical programs for the memory hierarchy is one of the most fundamental approaches to producing fast code” according to Lämml, Visser, and Saraiva [24]. Loop construction has a big impact of the program’s efficiency and execution time. A few key components that will cause inefficiency in loops if they are implemented poorly are:

- Switch and if-statements in hot-spots and inner loops
- Loops with complicated termination statements
- Dependent operations within inner loops
- Not reusing loaded data
- Using math library function calls inside loops

An additional measure that can be taken is unrolling and scheduling, which is a concept where the basic block of the loop is made bigger by increasing the number of in-loop instructions. By doing this, the number of condition evaluations can be decreased and thereby decreasing the number of iterations in the loop. Although this method can increase the speed by decreasing loop iterations, there is a backside to it. By increasing the code length, the instruction cache miss rates can also increase since the need for cache registers becomes bigger due to the renaming of the variables in the loops [24, 25]. Below is an example of manual loop unrolling.

**Normal loop**

```c
for (int x = 0; x<100; x++){
    delete(x);
}
```

**After loop unrolling**

```c
for (int x=0; x<100; x += 5){
    delete(x);
    delete(x + 1);
    delete(x + 2);
    delete(x + 3);
    delete(x + 4);
}
```
3 Technical work

This chapter presents the technical work within this thesis. This covers hardware and software design, setting up a test environment and designing test cases.

3.1 Design proposals

The theoretical study yielded design proposals on how the converter best could be done. For the pulse converter, only two of the algorithms from the background study were implemented for testing. This was extrapolation and multiplication. The idea behind interpolation in an early study was to measure the time between two pulses and interpolate the multiplied pulse times between them. This would yield a delay of at least one pulse but could give a more accurate result since nothing is based on a prediction. This delay was the reason interpolation never made it into the implementation state.

Since both FT and FP showed to cause the least error at different velocities, both of them were considered. The proposed design of the current converter was a combination of the conversion methods switching on a configurable threshold. This so the converter could be configured to best fit all possible use cases.

A prototype was designed with respect to fault preventing, which all conversion software could be implemented on. This prototype was done as close as possible to a final product, but also included extra connectors for testing, and a development board for faster prototyping.

3.2 Hardware design

The design of the prototype was made in CadSoft EAGLE PCB Design Software, a program where the schematics and electrical wiring can be created to enable future manufacturing. When the design of the prototype was finished and manufacturing was about to be initiated, the supervisor at KTH was consulted in the fault prevention process so that unnecessary errors might be avoided. A simple principle sketch of the converter can be seen in fig. C.1 in Appendix C.

According to Bosch Rexroth’s encoder specification, the input voltage can vary between 10-30V, and since the converter is connected to the same voltage supply, it had to be able to handle the same voltage range. Therefore two voltage regulators were installed; one regulating the voltage from 10-30V down to 5V, and one to 3,3V since the different components in the converter needed different supply voltages. The varying voltage causes the encoder pulses to vary in amplitude, zener diodes in series with a resistor were used to ensure a constant voltage level to the receiver as can be seen in fig. 3.1 below. Since the selected receiver had an input voltage limit of approximately 5,5V and the pulses from the encoder can vary between 10–30V, the zener diodes ensured a sufficiently constant voltage level of about 4,6V regardless of the level delivered from the encoder. To limit the current on the input to 20 mA, resistors of 1.5 kΩ were applied.
3. Technical work

Figure 3.1. Schematic of the resistor-diode application.

Since the encoder is a quadrature incremental encoder, a differential line receiver were utilized to convert the input into an absolute value. The receiver read the pulses, their inverse and outputted the same pulse without the inverse but with the full differential voltage between the two. This way, the full amplitude was achieved without having to consider the inverse signal in the processor.

To enable a conversion of the encoder signal, a LPC1768 microprocessor was utilized. The processor was chosen in cooperation with Bosch Rexroth based on its characteristics and that they had a lot of experience with the processor from earlier projects, which could be helpful if problems would occur.

Since the processor has a maximum output voltage of about 3.3 V, a open-collector hex buffer/driver was used to enable an output with the same voltage on the pulse signals as the encoder generates. For this purpose, transistors could be applied as well, but instead of installing separate transistors for each signal, this single driver amplifies all four signals while saving space and money.

To enable the current signal output, a negative feedback circuit with an operational amplifier was designed, see fig. 3.2. The DAC port of the processor is used to generate a voltage signal as an input to the circuit, representing the speed of the motor.

Figure 3.2. Design overview of the current loop circuit. The dotted line symbolizes the border between the driving and the measuring circuit.

A negative feedback circuit with the output directly connected to it’s negative input will act as a voltage follower. The amplifier will try to drive it’s output so the differential between it’s inputs approaches to
zero. The voltage drop across $R_f$ can be described as

$$V_{R_f} = i \cdot R_f \approx V_{ref} \quad (3.1)$$

which for a 160 $\Omega$ resistor would correspond to 0.64 – 3.2 V for a current between 4 – 20mA. As seen in eq. (3.1) the circuit can control the loop current independently of the resistance in $R_m$. When choosing $R_m$, assurance that the circuit can output a current of 20mA has to be made. The highest current available in the circuit is of both the variable parameters, $R_m$ and $V_{cn}$, and can be described as

$$i_{max} = \frac{V_{cn}}{R_f + R_m} \quad (3.2)$$

where $V_{cn}$ is the maximum output from the operational amplifier. In this case a rail to rail amplifier is used which means $V_{cn} \approx V_{cc}$. This circuit has the advantage of not needing a common ground between the driver and the measuring unit, since the current is running in a loop and is not referred to any ground reference. Single ended measurements of $R_m$’s voltage drop is therefore not available if the two devices share the same ground, since this would pull the feedback voltage down to zero. In case of a shared ground, a differential measurement of the voltage has to be made. When having separate grounds, both single ended and differential measurements can be utilized.

Since the prototype should support possibilities to verify the quality of the conversion, d-sub connections to enable dSpace integration were installed. With these connections, it was possible to analyze and verify the signal using dSpace and Simulink where various tests and plots of the signal were made, making it easier to see if the system worked as it should.

### 3.3 Conversion software implementations

Two different approaches were considered for the final software where one included both pulse scaling and current loop conversion in the same program whilst the other involved one separate program for each conversion. The aspects of choosing the proper methods are related back to the purpose of this thesis and should find support in the theories of making the system fail safe and handle scaling with as broad boundary limits as possible.

When it comes to making the system fail safe both approaches have its advantages and disadvantages. Making a system fail safe involves lots of testing and this process is both cheaper and less time consuming for one software than for two. However, increased complexity of the program also increase the risk of induced bugs which could be crucial for the dependability.

Increasing the boundary limits for the program clearly speaks for the second option with separate programs for each conversion. If another task is added to the software which requires time from the processor, this of course leads to lesser time for the original task. In this case adding a conversion from a pulse signal to a current, with additional filters, would reduce the performance of converting pulses. This could be solved by enabling separate parts of the program depending on which of the conversions that is currently requested. This is however very similar to using two softwares since it still requires a specific set up for each use.

Considering that separate programs can improve the performance due to shorter code, and the fact that each converter still requires individual programing for speed limits and pulse rates, the second option were chosen where the software were designed in two different versions.

Three different algorithms were implemented and tested on the pulse converter, these will further be explained below followed by the implementation of the current converter.

#### 3.3.1 Multiplying pulse converter

The first algorithm to be evaluated was the multiplication. The main concept behind this implementation is to capture the time between a change on the A or B channel of the encoder and have a second timer
interrupt taking care of when an output pin should be changed, an overview of this can be seen in fig. 3.3.

When a change on the input pins have been captured and the time from the last one determined, the direction of the rotation is checked. The output timer is then configured to interrupt on the input timers value multiplied by a certain factor. If the timer clock are below this value the program continues to run waiting for another interrupt. If the timer clock currently are higher than the new interrupt value but has not reached the previous interrupt value, the output interrupt function is instantly called in order to send out a pulse time as close as possible to the new interrupt time. The output Interrupt service routine (ISR) checks the direction, and can from that look in a table which of the output pins that should be changed.

A problem could occur with this solution in case the rotational movement stops. The last pulse would be captured and the output timer changes its value, which needs to be infinitely high to not call an interrupt which changes would result in a pin change. This would result in a converter which can not reach zero as an output. However, this can be solved giving the input timer an interrupt. A variable parameter zeroVelocityThreshold is used, which indicates the threshold where the device should consider the encoder as standing still. The timer interrupt value can therefore be calculated as

\[ T_i = \frac{60}{PPR \cdot 4 \cdot \text{zeroVelocityThreshold}[\text{RPM}]} \cdot F_t \]  

(3.3)

where PPR is the pulse rate of the encoder and \( F_t \) is the timers clock frequency. A block overview describing the complete software can be seen in fig. 3.4.
The converters theoretical delay, from a change on either of the input channels to a change on the output channels, is dependent on two parameters. These are the current motor speed and the output pulse rate. The worst case delay occurs when a output pin just have been changed before a interrupt is caught on the input. The delay can then be described as

\[
T_{d_{\text{theoretical}}} = \frac{60}{v_m[RPM] \cdot \text{outputPPR} \cdot 4}
\]  

(3.4)

where \(v_m\) is the motor speed in RPM. For a motor which starts from standstill the delay is slightly longer since one complete pulse is needed to get the time interval for the first pulse. This delay can be described as

\[
T_{d_{\text{theoretical}}} = \frac{60}{v_m[RPM] \cdot \text{outputPPR} \cdot 4} + \frac{60}{v_m[RPM] \cdot \text{inputPPR} \cdot 4}.
\]  

(3.5)

### 3.3.2 Extrapolating pulse converter

The extrapolating converter is implemented similar to the multiplying converter, except that the extrapolating uses another algorithm to calculate the output time for the scaled pulses. When defining the time between the pulses as a function of pulse changes as

\[
t_p = f(n)
\]  

(3.6)

where \(n\) is 1, 2, 3,... for every caught interrupt which corresponds to a motor movement. A pulse \(n\) could then be extrapolated as
3. Technical work

\[ t_p(n + 1) = f(n - 1) + 2 \cdot (f(n) - f(n - 1)). \] (3.7)

which yields a difference between the extrapolated- and the current pulse time. Assuming the velocity decreases linearly the scaled pulses must decrease with a factor

\[ k = \frac{t_p(n + 1) - t_p(n)}{PPR_{out}} \cdot \frac{PPR_{in}}{PPR_{out}}. \] (3.8)

After every output, the output timer is increased with the factor \( k \) to reach the extrapolated pulse time. This is repeated until a new interrupt is caught where new values are calculated with the new collected pulse time. The extrapolating converter is hence working similar to the multiplying converter which can be seen in fig. 3.4. The difference is, right after the output has been changed, the output timer is increased by \( k \). The output timer is set equally as the multiplication software when an input interrupt has been caught.

3.3.3 Current converter

As seen in the introduction, one of the required features for the converter was a pulse to current converter. The converter uses, as the theory chapter proposes, a combination between the FT and FP conversion techniques. At velocities below a specified threshold, the converter is operating with FP and as soon as the velocity exceeds this threshold, the converter switches to a conversion method using FT instead. As the theory states considerations must be taken so the input timer do not overflow. The LPC1768 comes with a 32-bit timer clocked at 100 MHz. This means the overflow time or the timer is

\[ t_o = \frac{2^{32}}{100 \cdot 10^6} = 42.95 \text{s} \] (3.9)

which is higher than any reasonable standstill value. This means the converter enters the standstill mode before any timer overflow can occur. A block diagram describing the converters software and how the alternation between the two conversion methods is done can be seen in fig. 3.5. To use only one of the methods, the threshold is simply set to either the maximum speed for only FP calculations and to 0 for FT.
3. Technical work

Figure 3.5. Block diagram describing the structure of the pulse to current converter.

The threshold controlling when the converter should switch to FT together with a variable update frequency for this conversion opens up for high flexibility, but awareness of bad parameter adjustment that could lead to instability must be made. The quantization error for a conversion can be seen in eq. (2.3) and the relative error in eq. (2.4). Increased $T_s$ gives small errors but causes a delay. To reduce the delay $T_s$ should be decreased, but decreasing $T_s$ lower than the time between two pulses will cause an unstable signal. The update frequency can be described as a function of encoder resolution and current velocity as

$$ f_u = \frac{v \cdot PPR \cdot 4}{60} $$

(3.10)

where $v$ is the current velocity expressed in RPM and $PPR$ the encoders resolution. The factor 4 comes from counting both rising and falling edge of the pulses. However, pushing the update frequency to its extent is not recommended and would cause bad measurements.

As seen in the theory chapter, a filter is often proposed in order to get smoother velocity estimates. The Kalman filter and the modified state observer requires knowledge of the system dynamics which is not possible in case of doing a converter applicable for any type of motor. These filters are also best at a constant speed system and can not handle velocity changes very well. The active alternation of sampling time have been proven to be effective, but such a solution is only applicable at FT conversion where this converter uses a combination of both FP and FT to be as accurate as possible. The remaining alternative was a traditional Butterworth low pass filter.

A low pass filter induces a lag in the system which could affect a control loop. The filter was therefore implemented as an optional attribute since the need of it could vary whether the converted signal should
be used for control or monitoring. Since an increased order of the Butterworth filter will induce more
phase delay and require more computational power, the filter was implemented of \( n \) \( : \) \( th \) order and can
be described as

\[
H(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}}{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}. \tag{3.11}
\]

This means the only input to the software is the numerator and denominator of the filters discrete transfer
function which are generated with Matlab's built in function "butter". This should be given on the form
\([a_0 \ a_1 \ldots a_n],[b_0 \ b_1 \ldots b_n]\). With this information the software itself senses the order of the filter.

### 3.4 Implementing dependability means

First action considered on making the converter more fail safe was some kind of redundancy. From the
theoretical background study hardware- time- and software redundancy were evaluated.

**Hardware redundancy** had to be weighed against the fact that more components would increase both
the cost and the size of the component. Since one of the actual designing constraints were to minimize
the converter this off course conflicted with implementing hardware redundancy. More components would
also increase the power consumption of the converter, which also conflicts with the aim of getting the
converter to consume the least resources possible. These facts from the theoretical background study
on hardware redundancy together with the designing constraints regarding power consumption and size
resulted in this not considered as a suitable solution for implementation.

**Time redundancy** could be done by calculating the output several times with a comparison between
the results. During early tests, it was seen that the processor already is suffering from high load and
many nestled interrupts, especially in higher velocities. Time redundancy would result in longer ISRs
and require more time from the processor. Since this was going to conflict much with the aim of enable
as high and fast scaling as possible, this was not a suitable solution in order to make the converter fail
safe.

**Software redundancy** is similarly to hardware- and time redundancy also giving trade-offs with other
attributes. However, software redundancy such as reasonableness checks and "checkpoint and restart"
which require low computational power, and also can be implemented outside the time critical ISRs were
implemented to handle certain failures described below.

Reasonableness checks are done whenever any variable output is made, such as the current loop output.
No checks are done at the quadrature pulse output since this only varies between high and low.

Checkpoint and restart are implemented to handle the situation when one of the input pulses disappears
or when any of the input interrupts are faulty. How it is implemented are described further in the fault
tolerance section.

The dependability means described in the background study were reviewed and applied to the converter
in order to attain even higher dependability, even though some fault tolerance already were implemented.
These implementations are described in the remaining part of this section.

### 3.4.1 Fault removal

Fault removal has been considered during the complete development phase. Continuous verification was
made with respect to the three phases of a fault removal process; verification, diagnostics and correction.
The verification consisted of unit tests written in the code in parallel with the software development.
Since the converter also should withstand harsh environments, a molded shell is a likely solution. This
could obstruct future fault removal during the use phase and could be an aspect to have in mind when
designing the encapsulation. This assumes no programming pins are available via some connector.
3. Technical work

3.4.2 Fault tolerance & forecasting

A fault forecasting were done to evaluate which faults that could occur, their impacts and whether they needed to be dealt with somehow. The forecasting was done with a qualitative approach and yielded that the two most likely faults to occur are either a software bug, a power outage, or a broken wire to the encoder.

A power outage could have two possible sources, either a broken wire so the converter are not giving any power, or a shutdown on an internal voltage regulator. This yields two different outcomes. A broken power supply to the converter will result in that all four outputs pulls down to zero, whilst a failing voltage regulator will pull all outputs to Vcc due to internal pull up resistors on the output circuit. Both situations will indicate an encoder standing still. A power outage could therefore not be solved by software and a hardware backup were not considered due to both increased cost and size.

A broken wire or a software bug could make the converter output an undefined velocity since the remaining input pulse will make sure the converter will not enter the standstill mode. This could be difficult to notice if this behavior just deviates slightly from the normal. A software solution was therefore implemented so the converter can indicate a broken encoder. This is handled by a checkpoint and restart, followed by a safe mode the converter can enter if the system still not responds after the restart.

When the program starts, a checkpoint is made as soon as the configurable parameters are loaded. When the converter is running, the A and B pulse ISRs are having a counter which increases every time the ISR is called. The two counters are also resetting the other one back to zero when they are executed. A counter reaching the value 500 (for a 1000 PPR encoder) thus means one of the input pulses are either lost or a bug prevents the ISR from executing. However, if this occurs the system reboots itself from the earlier defined checkpoint. If the program still malfunctions after the system have has been reset 5 times, the system shuts down the output and an external reboot is needed to start it again. Hardware diagnostics can then be initiated to make sure the encoder is not broken.

3.5 Software optimization

In order to answer how the software could be optimized and improved, one alternative encoder interface were implemented together with an adaptive resolution and an alternative programming interface. The improvements are general for all converters and are possible to run with as well the analog as the pulse converters.

**QEI - Internal quadrature encoder interface**

To detect a position change of the encoder, external interrupts are attached to the two encoder channels. Whenever a rising or falling edge is detected, an ISR is taking care of the interrupt and executes the required actions such as determination of direction, timing, etc depending on the the current use case. Some micro controllers comes with built in interfaces for quadrature encoders which automatically handles attributes such as direction determination, velocity and position automatically and can provide interrupts on specific events. This is a quadrature encoder interface which is mentioned in the theory chapter as a common way to read encoders. The NXP LPC1768 has this interface and this could improve the performance of the converter assuming the built in interfaces are faster than the ones implemented in the software. This method was evaluated and implemented as an improvement but could not be tested since the mbed development board have the encoder interface pins allocated to its LEDs. This couldn’t be changed without re-soldering of components on the development board.

**Adaptive resolution**

At initial testing it was seen that, at high velocities with high resolution encoders, the converter started to fail its conversions with an unstable pulse frequency at the output as a result. One theory was that
the ISRs had not time to complete their calculations before another was called from the interrupts. To enable a wider working range for the converter, this had to be solved and an adaptive resolution was therefore implemented.

Since the converters output frequency is directly related to the speed, this can not be modified to reduce the workload of the processor. The adaptive resolution’s idea is therefore to target the incoming interrupts and decrease these when speed increases. This is done by switching from interrupting on all edges to only interrupt at the As’ rising and falling, followed by only interrupting at Λ’s rising edge when speed increases even more. This would correspond to changing the resolution of the encoder from 4x to 2x followed by 1x of the encoders PPR.

In the main loop there are several checks running continuously which compares a calculated pulse rate to two predefined thresholds. This execution can be seen in fig. 3.6. Depending on these comparisons, the input interrupts are enabled or disabled to ensure optimal running conditions at higher speed.

![Figure 3.6. Block diagram showing the execution sequence of the adaptive resolution.](image)

The two thresholds, Thres1 and Thres2, are calculated as a interrupts per second defined as

\[ Thres = \frac{input\_Timer\_Clock}{in\_Clock\_Cycles} + \frac{output\_Timer\_Clock}{out\_Clock\_Cycles} \]  \hspace{1cm} (3.12)

where the timer clock frequencies are expressed in Hz.

**Alternative programming**

One improvement for a future converter which should actually work properly in humid environments, is to add an encapsulation, or even mold the component. This could affect the possibility to reprogram the MCU due to limited access to required pins. LPC1768 comes with an internal function In System Programming (ISP), which uses UART0 and the bootloader to reprogram the processors flash memory (Chapter 32 in the user manual).

Every time the device is either powered up or reset, the processors internal bootloader code is executed. A low level on pin p2[10] which remains for at least 3ms after the bootloader is called, triggers a hardware interrupt to enter the ISP. A High level pin after 3ms ignores the ISP and the bootloader is instead searching for an existing program on the MCU. A programmer, which support UART communication
and two pull down push buttons can hence be made to flash the converter. Such a device can be seen in fig. 3.7

Figure 3.7. Layout describing how an external converter could program the processor using already existing connectors in the converter.

The reset pin and p2[10] is connected to a pull up resistor, to ensure the default state when starting the converter is to look for an existing user software, and avoid unexpected resets due to floating pins. Pushing the reset button and simultaneously holding the ISP button for 3ms will enter the programming mode.

For UART programming a regular USB to UART converter can be used together with a software such as flash magic. The pins are connected to the A encoders input pins as seen in fig. 3.7 which routes to both UART0 at the processor and the encoder receiver. Zener diodes attached to the input lines will ensure no high voltage from the encoder during run phase can damage the processor, and the switching state will not affect the converter when it is running since if the UART pins are deactivated in the software.

When connecting this to an external system, one must ensure that the connectors to reset and p2[10] are floating instead of grounded, since this would yield a constantly reset processor. Shared ground between the converter and the programmer is also necessary for proper behaviour.

3.6 Verification and validation

The test environment was setup in the mechatronics lab at KTH as the necessary testing equipment for verification was available there. The performance of the system was concluded to be defined by a few characteristics, such as start up delay, position drift in the pulse signals and transitions between FP and FT for the current signal. All these characteristics were tested for different velocities, scaling factors and directions to see how the results vary throughout different working conditions. The position drift, boundary tests and all the current driver tests were performed on a test-rig provided by Bosch Rexroth, consisting of an axis with a magnetic strip, incremental encoder and motor together with a dSpace 1104 RkD Controller Board, as seen in fig. 3.8 below. Throughout the tests where dSpace were utilized, all the signal measurements were compared and evaluated in dSpace Control Desk since the program provides several functions that enables the user to get a good and accurate overview of the values.
3. Technical work

Figure 3.8. Test setup with test rig, converter and dSpace 1104 R&D Controller Board.

Although the test-rig from Bosch Rexroth was sufficient performance-wise for the tests mentioned earlier, where a high acceleration was not that vital, an additional motor and encoder were utilized for the response tests due to the high moment of inertia in the test-rig from Bosch, which prevented a high acceleration.

3.6.1 Pulses

The verification process for the pulse conversion was the most extensive, since several algorithms and scaling factors has to be tested for multiple scenarios to ensure that the test results corresponded to the performance of the converter.

System response

Since the signals go through a processor before being outputted, it was essential to see how much of a delay that would induce, since the converter could be used in a control system with high requirements on system response. To measure the induced delay in the system, two DSOX2012A oscilloscopes were utilized since they only had two channels and the tests required four. The tests were executed for the different implemented algorithms to see how they differed in performance so they could be compared and evaluated.

To measure the system delay, the time between the rising edge of the first input pulse and the rising edge of the first generated output pulse was measured, which can be seen in fig. 3.9 marked as $T$. To be able to know which of the input pulses A and B that came first, both of the input signals and the rotation direction had to be monitored with an oscilloscope. The motor axis was positioned so both the input pulse signals and the measured output were low, this made it possible to know which pulse that would come first and therefore enabling a measurement of the delay between the rising edges of the signals. If the second input pulse would be read instead of the first, it is hard to know if the first or second pulse triggered the output pulse, which makes the measurement of the delay very unreliable.
The tests were initiated by having the motor at standstill, then applying a high voltage of 24 V or a low voltage of 5 V to the motor depending on if a high or low acceleration was required in the test. Using trigger mode in the oscilloscope, the resulting pulse signals and delay could be acquired. The signal data could later on be transferred into Matlab so the delay and other signal characteristics could be analyzed. This procedure was executed for various scaling rates, directions and velocities, which in this case were represented by the input voltage. The scaling factors 3, 1/3 and 1 were used during the tests to see if the system performance would differ. During the tests where the signal was multiplied by a factor three, an encoder with a resolution of 1200 PPR was utilized, while an encoder with a resolution of 3600 PPR was utilized during the other two cases. The 1200 PPR encoder was used because the resolution would have become unnecessarily high during up scaling if a 3600 PPR encoder was used with a multiplication factor of three. At the same time as the resolution would become too low if the 1200 PPR encoder was utilized during the downscaling tests.

**Position drift**

The dSpace 1104 card was utilized to test and see if there was a significant drift between input and output travelling distance. If there would be a difference, it means that the outputted signal from the converter has a different time between the pulses, i.e. a different velocity than the received signal from the encoder which would induce a static error in velocity. The test setup can be seen in fig. 3.8 were the signals before and after the converter were read, so a comparison could be performed in dSpace. The position drift is measured as the number of rounds the output drifts during one minute. To obtain the drift per round, the achieved value can be divided by the velocity for the specific test case.

Tests were made for both of the multiplication and extrapolation algorithms to measure their performance throughout various scale rates and velocities. To perform these tests, the test rig provided by Bosch Rexroth with the 1856 PPR encoder was used together with the dSpace ds1104 card’s encoder interface. The test-rig from Bosch could be used during these tests since the acceleration was not as important as it was during the delay tests. Measurements were performed at two different constant velocities; 8 rpm and 200 rpm, to see if and how the performance differs throughout different velocities. At the two velocities, tests were made using 4, 1/2 and 1 as scaling factors.

**Maximum boundary limits**

Since one of the research questions regards the question about how the algorithms can be optimized to maximize the boundary limits for the pulse scaling, specific tests where executed so this could be answered. The tests were performed using dSpace and the more stable test-rig from Bosch Rexroth, to attain as stable signal as possible so the measurements could be analyzed with more accuracy. The encoder with a resolution of 1856 ppr, was driven with a velocity of 300 rpm, which is the maximum attainable velocity for the rig. While running at maximum velocity, various scale factors between 1/10 and 10 was implemented to see how the performance changed after an increase or decrease of resolution. Both
the multiplication and extrapolation algorithms were tested in the same way to enable easy comparisons between the two.

### 3.6.2 Current driver

To evaluate the current circuit, the ADC ports on the dSpace board was utilized to read the voltage drop across a resistor. A conversion from voltage to velocity was done using Simulink, which then could be displayed in dSpace control desk.

#### Step response

System performance is highly dependent on how good the output signal follows the input signal, both at constant velocities as well as during changes, such as accelerations etc. To see how well the output signal follows the input signal during rapid changes, the motor is accelerated in different directions as can be seen in fig. 3.10 below.

![Figure 3.10. Sample plot of system response measurement.](image)

#### Steady state

The current signal was measured at various steady states, i.e. constant velocities, to evaluate the signal stability and potential steady-state error. To see if the signal performance was dependent on the velocity of the motor, tests were made for velocities from 10 rpm to 300 rpm. The top velocity was set due to limitations on the test-rig.

#### Transitions between FP and FT

Since two different algorithms for the pulse to current conversion, which are separated by a threshold set by the user, are implemented for the current circuit, this transition was evaluated to see if it would have enough of an impact to be a problem for the system performance, as can be seen in fig. 3.11. The threshold was set to 15 rpm since that seemed to be a satisfactory level concerning signal stability and accuracy. Various tests, including both slow and fast transitions over the threshold were performed to see if different results would be presented.
Figure 3.11. Sample plot of transition between FP and FT.
4 Results

This chapter will present the outcome and results from the tests described in the previous chapter. It will start with the results from the pulse converters followed by the current converter. Last the tests of the fault tolerance mechanism will be presented.

4.1 Multiplying converter

All the test results collected from the different test cases; system response, position drift and boundary limits, during the verification process of the multiplication algorithms are presented in this section.

4.1.1 System response

Even though the tests on system response were performed in a rather controlled environment where every test started with an initial velocity of zero followed by an acceleration phase, which can be seen in fig. 4.1, some of the measurement data deviate quite a bit, which can be seen in tables D.1 and D.2 in Appendix D. The measured response time is marked with $T$ in the figure. Despite the deviations, the results clearly show that fastest system response with the multiplication algorithm is achieved by no scaling of the signal. Table 4.1, containing the average values from tables D.1 and D.2, shows that the average response time for downscaling is almost a millisecond slower than no scaling, which in this case is a big difference. The worst results are presented using up-scaling, which has an average response time more than twice as long as no scaling. The amplitude of the input signal is only 5 V since the encoder that was used for these test had a rated supply voltage of 5 V.

Testing the delay in the system can be done in a lot of ways and it is therefore very difficult to discuss how the results represent the performance of the system. Due to time restrictions within the project, only one type of response tests could be performed.

Table 4.1. Average response times for the multiplication algorithm during two different accelerations, where Case 1 represents the low acceleration tests using 5 V and Case 2 represents the high acceleration tests using 24 V.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Case 1 [ms]</th>
<th>Case 2 [ms]</th>
<th>Average [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.658</td>
<td>1.223</td>
<td>1.441</td>
</tr>
<tr>
<td>1/3</td>
<td>3.038</td>
<td>1.809</td>
<td>2.424</td>
</tr>
<tr>
<td>3</td>
<td>3.765</td>
<td>2.047</td>
<td>2.906</td>
</tr>
</tbody>
</table>
4. Results

![Graph showing input and output pulse signals over time]

Figure 4.1. Plot of system response measurement with multiplication algorithm and scale factor 1/3.

### 4.1.2 Position drift

The position drift, which is caused by a static error in the outputted pulse signal would in a position-controlled system cause problems since the converter will not present the same position as the encoder. In a velocity-controller system as this converter is designed for, these deviations do not have as significant impact. This is because a slight error in velocity might case the system to run a bit faster, while an error in a position system might cause something in the system to be where it is not supposed to and therefore causing damage or even dangerous situations.

Even though the signals with various scaling factors are affected less than the signal without scaling, it is from table 4.2, clear that the static error is strongly dependent on the velocity; where the error increases with increasing velocity. This is also easily verified by looking at fig. 4.2, where the lower plot fig. 4.2b with a velocity of 200 rpm compared to the one above with 8 rpm, clearly has a bigger positive static error, which induces the drift. The numbers in the table represent the deviation in position from the input signal, which is measured over a period of one minute, and therefore gives the unit rounds per minute, since the position is measured in rounds. More measurements can be found in tables D.5 and D.6 in Appendix D.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Drift [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 [rpm]</td>
</tr>
<tr>
<td>1</td>
<td>0.001</td>
</tr>
<tr>
<td>1/3</td>
<td>0.007</td>
</tr>
<tr>
<td>3</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Table 4.2. System drift using the multiplication algorithm at various scale factors.
4.1.3 Boundary limits

Since the first research question of the thesis regards the subject of how the conversion algorithms can be optimized to maximize the boundary limits for scaling pulses to higher/lower resolutions, these boundary measurements are of significance due to them being very performance defining for the converter. The measurements were performed for several scaling factors, both upwards and downwards, which can be found in table D.9 in Appendix D, while few of the measurements that clearly show how the deviation between the signals are related to the scale factor are presented in table 4.3. The plots in fig. 4.3 that are related to table 4.3 also shows a clear trend concerning the deviation between the input and output, starting with a small positive deviation of about one rpm with scale factor one, then later on increases along with the scale factor, and finally ends up at a deviation of about -13 rpm with the scale factor 10, which is a deviation of about 4% and too large to be acceptable.

Table 4.3. Average speed deviations at various up-scale factors for multiplication algorithm.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>298,39</td>
<td>299,23</td>
<td>0,84</td>
</tr>
<tr>
<td>4</td>
<td>297,22</td>
<td>296,01</td>
<td>-1,21</td>
</tr>
<tr>
<td>10</td>
<td>296,12</td>
<td>283,20</td>
<td>-12,92</td>
</tr>
</tbody>
</table>

Figure 4.2. Plots of system drift using multiplication algorithm and scale factor 1.
As can be seen in table 4.4, where some of the values from table D.10 in Appendix D are extracted, the measurements performed during downscaling of the signal show that downscaling of the signal does not affect the static error as much as an up-scaling of the signal does. However, the plots in fig. 4.4 do show how the scaling factor affects the smoothness of the signal due to the low number of pulses. In fig. 4.4c, where the signal is scaled by a factor 1/10 and therefore has a resolution of 186 ppr, a sharp-edged oscillating signal trying to follow the input signal is achieved. If not filtered, these oscillations will also create oscillating behavior in the readings of the velocity in the system.
Table 4.4. Average speed deviations at various downscale factors for multiplication algorithm.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>299.39</td>
<td>300.73</td>
<td>1.34</td>
</tr>
<tr>
<td>1/4</td>
<td>298.16</td>
<td>299.59</td>
<td>1.43</td>
</tr>
<tr>
<td>1/10</td>
<td>299.20</td>
<td>300.86</td>
<td>1.66</td>
</tr>
</tbody>
</table>

Figure 4.4. Plots of boundary limit measurements using multiplication algorithm and various scale factors.
4. Results

4.2 Extrapolating converter

All the test results collected from the different test cases; system response, position drift and boundary limits, during the verification process of the extrapolation algorithms are presented in this section.

4.2.1 System response

As the response tests for the extrapolation algorithm were performed in the exact same way as for the multiplication algorithm; the initial state of the motor was at standstill and then followed by an acceleration phase at two different voltages, similar outcomes were achieved, as seen in fig. 4.5.

The results in table 4.5 shows that the extrapolation algorithm follows the same pattern regarding the response time as the multiplication algorithm, where the un-scaled signal has the shortest response time, followed by downscaling and then once again the slowest; up-scaling. Comparing the two algorithms regarding response time, it is also clear that the multiplication algorithm had a better overall performance with a faster response time for all of the three scaling factors. The raw data collected during the tests can be found in tables D.3 and D.4 in Appendix D.

Table 4.5. Average response times for the extrapolation algorithm during two different accelerations.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Low acc</th>
<th>High acc</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.565</td>
<td>1.345</td>
<td>1.455</td>
</tr>
<tr>
<td>1/3</td>
<td>3.298</td>
<td>1.966</td>
<td>2.632</td>
</tr>
<tr>
<td>3</td>
<td>3.971</td>
<td>2.415</td>
<td>3.193</td>
</tr>
</tbody>
</table>

![Figure 4.5](image-url) Plot of pulse drift measurement with extrapolation algorithm and scale factor 3.

4.2.2 System drift

Like the multiplication algorithm, extrapolation yields deviations from the input signal that are strongly related to the velocity, see table 4.6. The errors at low velocities are comparable to the ones attained using the multiplication algorithm, but when comparing the algorithms at higher velocities, the performance of the extrapolation algorithm is a lot worse, due to the vastly increasing static error. The average deviation is between 4 – 30 times worse for the extrapolation algorithm, due to the larger error at higher velocities. This can also be visually verified from fig. 4.6, where the output signal has a clear positive error compared to the input. More measurements can be found in tables D.7 and D.8 in Appendix D.
Table 4.6. System drift using the extrapolation algorithm at various scale factors.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Drift [rpm]</th>
<th>Drift [rpm]</th>
<th>Average [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.003</td>
<td>7.92</td>
<td>3.96</td>
</tr>
<tr>
<td>1/3</td>
<td>0.002</td>
<td>6.29</td>
<td>3.15</td>
</tr>
<tr>
<td>3</td>
<td>0.003</td>
<td>6.51</td>
<td>3.26</td>
</tr>
</tbody>
</table>

(a) 8 rpm, scale factor 1.

(b) 200 rpm, scale factor 1.

Figure 4.6. Plots of system drift using extrapolation algorithm and scale factor 1.

4.2.3 Boundary limits

Some interesting results were acquired during boundary tests of the extrapolation algorithm. Since the algorithm has a clear positive static error during higher velocities and no scaling, the static error actually got better to some extent when the scale factor was increased. As can be seen in table 4.7 and fig. 4.7, this is the case up until about a scale factor of six, were the average static error is virtually zero, while a further increase of the scale factor started to induce a negative static error instead, as it did for the multiplication algorithm. Even though the static error decreased until a certain point, the stability and peak-to-peak amplitude of the signal got worse due to the signal being less stable. More extensive measurement data can be found in table D.11 in Appendix D.
4. Results

**Table 4.7.** Average speed deviations at various up-scale factors for extrapolation algorithm.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>297.06</td>
<td>301.08</td>
<td>4.02</td>
</tr>
<tr>
<td>6</td>
<td>298.52</td>
<td>298.75</td>
<td>0.23</td>
</tr>
<tr>
<td>10</td>
<td>299.75</td>
<td>293.35</td>
<td>-6.40</td>
</tr>
</tbody>
</table>

**Figure 4.7.** Plots of boundary measurements using extrapolation algorithm and various scale factors.

Table 4.8 and fig. 4.8, which contain extracted values from table D.12 in Appendix D, show approximately the same results as the measurements for the multiplication algorithm; the static error maintain approximately the same throughout the various downscaling factors, while the peak-to-peak amplitude and sharp-edged oscillations increase.
Table 4.8. Average speed deviations at various downscale factors for extrapolation algorithm.

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>297.44</td>
<td>301.83</td>
<td>4.39</td>
</tr>
<tr>
<td>1/4</td>
<td>300.08</td>
<td>304.55</td>
<td>4.47</td>
</tr>
<tr>
<td>1/10</td>
<td>296.84</td>
<td>300.99</td>
<td>4.15</td>
</tr>
</tbody>
</table>

Figure 4.8. Plots of boundary limit measurements using extrapolation algorithm and various scale factors.
4. Results

4.3 Current converter

All the test results collected from the different test cases; step response, steady state and transitions between FP and FT, during the verification process of the multiplication algorithms are presented in this section.

4.3.1 Step response

To measure an accurate step response for the current circuit proved to be very difficult. The final solution that was used to measure how well the current conversion algorithm performed during sudden changes in the input signal, was to simply apply various accelerations in different directions to the motor and the use the plots to verify the outcome. In the initial part of fig. 4.9 an acceleration between approximately 0 to 50 rpm is performed in about 0.2 seconds, and shows that that the output signal follows the input signal rather well. The phenomenon where the static error is bigger during velocities below 15 rpm is because of a change in conversion algorithms between FP and FT, which will be explained later on in this chapter.

![Figure 4.9. Current conversion performance during accelerated changes in input signal.]

4.3.2 Steady state

The results from the deviation measurements, which can be seen in table 4.9 and fig. 4.10 clearly, apart from one value, show a growing error when the velocity is increased. This is the deviation value at 30 rpm, which is lower than the deviation at 10 rpm, due to a change in algorithms at 15 rpm. When the motor rotates with a velocity below 15 rpm, the signal conversion is performed with the FP algorithm, while above 15 rpm; the FT conversion method is used. The outcomes of this are slight differences in the deviation results below and above 15 rpm. Therefore, tests of the transition between the two algorithms have been performed as well and the results are presented later in the report. The first column in table 4.9 represent the theoretical reference values where the measurement is being performed, the second presents the actual input velocity to the converter, the third column presents the output velocity from the converter and the fourth column presents the deviation between the input and output.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.07</td>
<td>10.68</td>
<td>0.61</td>
</tr>
<tr>
<td>30</td>
<td>29.62</td>
<td>29.62</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>100.31</td>
<td>101.65</td>
<td>1.34</td>
</tr>
<tr>
<td>300</td>
<td>297.79</td>
<td>301.16</td>
<td>3.37</td>
</tr>
</tbody>
</table>

![Table 4.9. Average speed deviations at various velocities for current conversion.]

48
From the plots in fig. 4.10, it is also possible to see that the peak-to-peak amplitude in the signal also increases with the velocity. Notice that the scales of the y-axis are different between the plots. This is not only a product of the increased velocity, but also due to the resolution of the DAC that is utilized to produce the output signal. The DAC only has a 10-bit resolution, i.e. values between 0 - 1023, which means that the boundaries of the velocity will affect how good the resolution of the output is. An example of this can be seen in fig. 4.11, where the motor is at standstill, but the maximum velocity boundary is set at different values between 100 - 1500 rpm while the minimum is kept at 0 rpm. For the 0 - 1500 rpm case this gives an accuracy of $\frac{1500}{1024} \approx 1.46$ rpm, which a clear picture of how much the required velocity span affects the stability and accuracy of the signal. Measurements on more velocities are available in table D.13 in Appendix D.
4. Results

4.3.3 Transitions between FP and FT

Although the measurements from the transitions between FP and FT, shows in figs. 4.12 and 4.13 that the signal is barely affected by the transition point at 15 rpm except from a slight hold-up, it is clear that the FP conversion method gives a bigger static deviation from the input signal than the FT algorithm does. This is clear when comparing the first 0.3 seconds of fig. 4.13, which has a velocity below 15 rpm, with the part of the signal that has a velocity higher than 15 rpm.

Figure 4.11. Inaccuracy measurements due to low resolution in DAC.
4. Results

Figure 4.12. Transition scenario between FP and FT.

Figure 4.13. Close-up of a transition scenario between FP and FT.
5 Discussion & Conclusions

This chapter presents a discussion around the three questions to be answered by the thesis followed by the conclusions it yielded. It is structured in three sections where each section aims to discuss and answer each question.

5.1 Conversion Algorithms

As the results show, the two tested conversion algorithms can be implemented on a MCU and function as a pulse converter, even though they both have their advantages and disadvantages. The multiplication, which has proven to be the best in the tests, are more stable than the extrapolation and produces signals with less noise. The multiplication computations within the ISRs are also easier to execute than the extrapolations’ which makes it possible to run at a higher pulse intensity. The multiplication do not leave as much room for improvements as the extrapolating converter. If the algorithms are improved and tuned for a well known use case, the extrapolating converter could be used to predict the motor speed and hence actually increase the encoders resolution.

There were however a few factors that interfered with the results. At higher frequencies transients started to occur on the input signals. This had two contributing factors; a non optimized Printed Circuit Board (PCB) due to limited manufacturing equipment, and also the external connection to the dSPACE ds1104. This was due to the long cables between the computer and the prototype which was necessary. Exactly how much this interfered with the test results is difficult to say, but it was noted that the converter misinterpreted the input signals sometimes when these were connected. The receiver helped to reduce this problem, since it actually was able to read the signals even though smaller transients appeared. However, it is difficult to tell exactly how this affects the timing between the rise and fall of the output from the receiver.

The long ISR showed to cause some problems at higher frequencies due to nestled interrupts and this was most likely also affected by the transients at the input. Reading several pulses over a certain time, and determine a velocity from which a new pulse time could be derived, could reduce the ISR time. This would most likely produce a worse accuracy, as well as infeasibility in a case where position data is used from the encoder.

As an attempt to improve functionality at even higher frequencies, the adaptive resolution was implemented. An alternative to this would be to only look at rising A-pulses throughout the whole working range, but this would result in a longer delay for capturing a velocity change. The resolution will also decrease for velocities above the threshold, but compared to not working at all without the adaptive frequency, this was seen as a successful improvement.

The delay of these two converters has shown to be quite similar. From zero velocity the multiplying converter is slightly faster due to faster calculations and a different software layout. A velocity change when the converter is running can be detected faster by the extrapolating converter since this changes the out-pulse frequency continuously between the input pulses, whilst the multiplying converter outputs a constant time between them until a new input is read.

After all testing, it has been shown that of these two algorithms, the multiplying pulse converter together
5. Discussion & Conclusions

with the adaptive resolution is the most promising concept. This gives the best balance between accuracy, delay and stability.

5.2 Fault tolerance mechanisms

Attempts to make the product fail safe were also implemented. Tests has proven the algorithms to be successful and fulfill its expected characteristics. It is difficult to measure whether the converter has been fail safe or not since everything in the end comes down to the question; What is fail safe? If more redundant components and processors are added, it is always possible to make something more fail safe and increase the dependability, but there are no such thing as "complete fail safe". Since all dependability means and redundancy requires some trade-off with either money, time and space, making something fail safe should rather aim at "making the product fail safe enough". A component in a space rocket or airplane would therefore require a lot more redundancy before it can be called fail safe than a component at a lawn mower would.

Involving the dependability means should always be a part of the development process for new products, this makes sure the product becomes more robust and software flaws are removed early in the process. Since fault prevention and fault removal are so closely related with good engineering practice this is often considered even though it sometimes might be unconsciously.

5.3 Further optimization

The design and selection of components have been made with the consideration that the converter should have the possibility to withstand harsh environments if an encapsulation is added. Since no testing has been made whether this hardware actually will be beneficial, no conclusion can be drawn from this. Nor can any conclusions be drawn whether the possibility to program the controller via the unused inputs will help since no testing has been performed. This could not be done as the development board do not have access to the required pins, UART0 and p2[10], and also limited time in the end of the project. There are good sealed connectors which could be placed on the converter and provide access direct to the programming pins and still have good encapsulation. This would thus result in a more expensive end product. Enabling configuration through an already existing connector would therefore rather help reducing the price more than increasing the resistance against external environmental threats.
6 Future Work

This chapter presents the future work with the converter. This includes the modifications needed to fulfill the environmental constraints of an industrial environment and possible improvements of design and conversion algorithms.

6.1 Software and Algorithms

The conversions algorithms evaluated are multiplication and extrapolation of the quadrature pulses. The extrapolation caused noise on the output signal, but only one extrapolation method were evaluated due to limited project time. There is room for evaluating more predictive techniques to see if the noise can be reduced and if the converter can be used to not only scale the frequency of the pulses, but also predict the motor movement to an extent as it can be used as a software resolution increase.

Since the converter has showed to drift in position especially at higher pulse intensities, it could be interesting to implement a self calibration which counts the output and input pulses. This will probably lead to lesser capability of converting at higher frequencies due to increased processor load but depending on the required characteristics it could be an improvement.

Implementing interpolation as a conversion method could be interesting if a slightly longer delay is accepted. This might give more accurate results, and on high pulse intensities, one pulse will give delays so low it might not affect a controller.

6.2 Hardware

The prototype is designed and manufactured with limited resources and just a two layer PCB is used which makes it difficult to remain intact ground planes for optimal performance. A future converter should have a better designed layout with a several layer PCB to attain a good Electro Magnetic Compability (EMC), and be encapsulated to withstand humid environments. The prototype also uses a development board with the micro-controller LPC1768 whilst a future product should be designed with only the chip. The schematic must therefore be slightly modified and the development board must be replaced with a single processor and the external component it needs.

The processors internal DAC has a resolution of 10 bits, which means that the required velocity interval is divided into 1024 steps. If more accuracy is necessary, an external DAC with higher resolution could be added and connected via serial communication to the processor. It is also possible to use a PWM with a low pass filter instead of the DAC, as long as it do not cause noise and variations in the input signal.

The limiting factor for high pulse intensity showed to be the computational capability of the MCU. A FPGA could make it possible to run the converter at a wider range, but would probably also lead to a more complex configuration. This could be an interesting modifications for fixed scaling though.
References


A Encoder Fundamentals

Encoders are components used to translate motion or position from mechanical systems into digital signals processable for computers and other electrical devices. Most of the manufacturers distributing encoders are dividing these into the two main categories absolute and incremental [26, 27] depending on what signal they are outputting.

The absolute encoder gives the current position in the form of a binary value where each angle are represented by one value. The resolution from these encoders are therefore related to the number of output pins from it, e.g an 8-bit encoders have the resolution of \(360^\circ/256\) degrees [28]. One advantage the absolute encoder have is that in case it loses power, it will still remember its latest position due to the fact that each position have it’s own binary value.

The incremental is unlike the absolute encoder measuring a position change. It provides a number of equally spaced pulses per revolution, or among a certain length in case of a linear encoder. This value is noted as PPR or pulses per inch/mm in the linear case. One can see the difference between an incremental and absolute encoder disk in fig. A.1 to note the difference between the two types. These encoders can have one, two or three pulses depending on how much information that should be gathered from it. If direction of the movement is not considered, a one pulse encoder is sufficient. If one also is in need of the direction a two pulse converter must be used. This converter provides two channels with pulses whereas one is 90 electrical degrees out of phase [29]. The three channel encoder are also providing a reference position that comes one time each revolution. This pulse is commonly named the z-pulse or 0-pulse. These signals and how to read them will further be explained later in the report.

Danaher Industrial Controls [29] are describing the difference between incremental and absolute encoders as following:

"The difference between incremental and absolute encoders is analogous to the difference between a stop watch and a clock."
A Encoder Fundamentals

A stop watch measures the incremental time that elapses between its start and stop, much as an incremental encoder will provide a known number of pulses relative to an amount of movement. If you knew the actual time when you started the watch, you can tell what time it is later by adding the elapsed time value from the stop watch. For position control, adding incremental pulses to a known starting position will measure the current position.

When an absolute encoder is used, the actual position will constantly be transmitted, just as a clock will tell you the current time.

When speaking of the performance among encoders it is important to know the difference between two terms, accuracy and resolution. Resolution are defined as how many measuring points there are at each revolution, i.e the encoders PPR. Accuracy on the other hand are corresponding to the deviation between the measured and the actual position, i.e the measurement error [29, pp.4]. These phenomena can be seen in fig. A.2 where one sensor have good accuracy but low resolution and the other having high resolution but bad accuracy.

![High accuracy - Low resolution](image1)

High accuracy - Low resolution

![Low accuracy - High resolution](image2)

Low accuracy - High resolution

Figure A.2. Picture describing the pulse distribution in an encoder depending on its accuracy and resolution.

The remaining part of this section will focus on incremental encoders. It will describe different types of encoders and the signals they can provide.

A.1 Sensing Technologies

All incremental encoders are not designed the same and their sensing technology can differ a lot between encoders. This due to the fact that some design options are more suitable for certain environments while in some cases it might not even be possible to mount one kind due to mechanical constraints. The most common used sensing technologies will further be described below.

Optical sensing

The most widespread encoder design is the rotary optical encoder and it is consisting of mainly three parts; a light source, an encoder disk and a photo detector. The light source shines at the encoder disk, which has opaque and transparent segments equally distributed around the outer edge of the disk. This pattern can be seen in fig. A.1b. I case of a rotary motion, the disc starts to spin and alternates between passing the light to the sensors and blocking it. The light sensors are then causing the pulse output whether they catch the light or not [28].

A2
The above described encoder would give only one signal. To receive both an A and B pulse, another row of slits is needed on the encoder disc. These slits are displaced 90 electrical degrees from the slits catching the A pulse. The same technique is used to get a Z pulse aswell. Another light source, photo sensor and another row with a slit in the encoder disk [28].

Earlier the difference about accuracy and resolution where mentioned for incremental encoders. For an optical encoder this is easy related to hardware characteristics. The resolution is directly related to how many slits the encoder disk have whilst the accuracy is depending on how precisely they are made. Also the quality of the light source and photo sensors can affect the accuracy of the encoder.

**Magnetic encoders**

Magnetic encoders are well suited for industrial applications since it is not only resistant to common industrial contaminants such as dust, grease, moisture but also physical interference as vibration [29, pp.3]. This is due to its simple construction without any optical sensors or rotating disks. The magnetic encoder is instead using permanent magnet and magnetic sensors to detect movement and position. The simplest magnetic encoder would consist of a permanent magnet attached on the motor axis with its north and south pole 180 degrees from each other. This would yield a sin wave at the magnetic sensor with the same frequency as the motor shaft speed. However this solution gives a resolution of 2 PPR which is not sufficient in many cases. To increase the resolution from the encoder, a magnetic strip is attached on the encoder disk or on the motor axis instead of a single magnet. This strip contains several permanent magnets were every other slot contains a north respectively a south pole [30].

A second sensor mounted 90 degrees from the other will produce a cos wave. This signal is not only making it possible to determine direction of the movement, but can also be used together with the sin wave to interpolate absolute position. For incremental encoders, these sinusoidal waves from the sensors are converted into quadrature square waves [30].
B Initial background study

This chapter serves as a general study on what existing converters that currently are available on the market. It will cover existing solutions for pulse-to-pulse converters, pulse-to-analog converters and frequency scaling. The purpose of this chapter is to read up on the current solutions to find out what improvements that can be made and if there are existing solutions that can be used.

B.1 Pulse-to-analog converters

To ensure that the latest technology is utilized while developing this component a state-of-the-art study of pulse-to-analog converters is also performed.

A lot of industries today use an analog signal to monitor or control the speed of a system, and to make the converter compatible with as many systems as possible it should be available as an output as well. Since this converter is being developed in cooperation with Bosch Rexroth, a few of the properties will be customized to satisfy their requirements. The converter should be able to take an incremental quadrature pulse signal as an input and convert the signal to an analog 4-20 mA signal. Since their systems are used by customers all over the world, the converter has to withstand extreme environmental conditions, i.e. temperatures between -40°C to +50°C and high humidity. The converter should also have a restricted size so it can be mounted on an 8-pole M12 cable without any additional support. It should also have implemented fault mechanisms such as a data storage function, in case of power loss.

A research on different converters is performed to see if there is an existing product that satisfies the requirements. Ten converters that might meet the requirements are found and a comparison of them is made, see table B.1 below.

Some of the evaluated properties are:

**Universal input**
This refers to if the encoder accepts an arbitrary pulse signal as an input or if it is strictly designed to only handle incremental quadrature pulses.

**Signal summation**
Some converters with multiple inputs have the possibility to calculate the sum of two independent signals.

**Input frequency**
It is very important that the converter can handle the pulse frequency of the motor encoder and according to the encoder’s data sheet [31] together with information about the motors [32], the converter must be able to handle at least 12.5kHz.

**Output signal**
There are a few common analog output signals used today, some use current and some use voltage. For this project a 4-20 mA signal will be required.

**Output resolution**
To ensure an accurate and smooth output signal, it is important to have a sufficient resolution.
B. Initial background study

**Temperature**
Since the temperature is a part of the requirements, this is an important aspect.

**Size**
A goal is to make the converter small enough to put it on a 8-pole M12 cable without any additional support. This means that an approximate maximum size is about 100 × 50 × 20mm.

**Fault mechanisms**
Since the converter should be as stable as possible, some fault mechanisms should be implemented. One function could be that it stores the data in case of power failure, this would make it possible to keep track of the position of the encoder even though there is a power failure.

None of the presented converters meet the requirements to a 100%, but there are a few that satisfy the most important ones, such as in- and output signal types and input frequency. The converter should be able to handle a signal above 50kHz and in this comparison there are four converters that meet that requirement. Some requirements that non of the converters satisfies are the size and temperature. What this depends on will have to be evaluated.

As a conclusion, none of these converters satisfy the requirements. This means that there is probably not an existing pulse-to-analog converter that could be used for this purpose.
### Table B.1. Comparison between seemingly relevant pulse-to-analog converters.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Single (Universal)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Quadrature</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Signal summation</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Frequency</td>
<td>0 - 4.5kHz</td>
<td>0 - 250kHz</td>
<td>30Hz - 10kHz</td>
<td>0 - 1kHz</td>
<td>0.25 - 2.5kHz</td>
<td>N/A</td>
<td>0.2 - 180kHz</td>
<td>3kHz</td>
<td>0 - 1MHz</td>
<td>0 - 1MHz</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-20mA</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4-20mA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0-10V</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>-10-10V</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Resolution</td>
<td>16 bits</td>
<td>16 bits</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>14 bits</td>
<td>14 bits</td>
</tr>
<tr>
<td>Temperature[°C]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>0 to +50°C</td>
<td>0 to +50°C</td>
<td>0 to +60°C</td>
<td>-40°C to N/A</td>
<td>0 to +70°C</td>
<td>0 to +50°C</td>
<td>0 to +60°C</td>
<td>-10 to +50°C</td>
<td>N/A</td>
<td>0 to +45°C</td>
</tr>
<tr>
<td>Storage</td>
<td>-20 to +70°C</td>
<td>-40 to +85°C</td>
<td>0 to +60°C</td>
<td>N/A</td>
<td>-65 to +150°C</td>
<td>0 to +50°C</td>
<td>-20 to +70°C</td>
<td>-10 to +50°C</td>
<td>N/A</td>
<td>-25 to 70°C</td>
</tr>
<tr>
<td>Size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sufficient</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Too big</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fault mechanisms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data storage at power failure</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
B. Initial background study

B.2 Pulse-to-pulse converters

Since the developed converter should have both an analog and an quadrature pulse output, a state-of-the-art research is performed on pulse-to-pulse converters as well.

Since both the pulse-to-pulse converter and the pulse-to-analog converter are supposed to be integrated in the final product, the environmental requirements such as temperature and size are the same for both of them.

Eight converters that might meet the requirements is to be evaluated. The most vital properties that will be evaluated are the input frequency, output frequency and the scaling. The evaluation can be seen in table B.2

Some of the evaluated characteristics are:

**Input frequency**
The converter will have only one input, which is the incremental pulse signal. Therefore the requirement for the input frequency will be the same as for the pulse-to-analog conversion.

**Output frequency**
One of the goals is to make the frequency span as wide as possible to ensure compatibility with as many systems as possible.

**Scaling**
For the final product, both an upscale and downscale is required. Therefore all the converters that are able to do either one are to be evaluated.

**Temperature and size**
As earlier mentioned, the requirements for temperature and size are the same as for the pulse-to-analog converter.

**Fault mechanisms**
Fault mechanisms such as data storage at power failure is evaluated for this converter as well since it is desirable.

Although none of the converters satisfy all the requirements as seen in table B.2, there is one converter called Motrona FM260 that has an up-scaling property, which can be very useful. The method this converter uses to scale pulses is not suitable for this application. It only considers the number of pulses and not the timing between them.
## Table B.2. Comparison between seemingly relevant pulse-to-pulse converters.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>0 - 100kHz</td>
<td>0 - 10kHz</td>
<td>0 - 5kHz</td>
<td>0 - 5kHz</td>
<td>0 - 300kHz</td>
<td>0 - 1MHz</td>
<td>0 - 3kHz</td>
<td></td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>0 - 20kHz</td>
<td>N/A</td>
<td>0 - 10kHz</td>
<td>0 - 5kHz</td>
<td>0 - 1.2kHz</td>
<td>0 - 300kHz</td>
<td>0 - 1.9999GHz</td>
<td>0 - 3kHz</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up</td>
<td>-</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Down</td>
<td>X</td>
<td>N/A</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Factor</td>
<td>0.9999 - 0.0001 x 10^{-6}</td>
<td>N/A</td>
<td>1 - 1/165000</td>
<td>1 - 1/2047</td>
<td>1 - 1/999</td>
<td>1 - 1/4096</td>
<td>0.0005 - 1999.98</td>
<td>1 - 1/10000</td>
</tr>
<tr>
<td><strong>Temperature[^°C]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>-5 to +60°C</td>
<td>-20 to +60°C</td>
<td>0 to +60°C</td>
<td>-10 to +60°C</td>
<td>0 to +60°C</td>
<td>0 to +45°C</td>
<td>0 to +45°C</td>
<td>-10 to +60°C</td>
</tr>
<tr>
<td>Storage</td>
<td>N/A</td>
<td>-20 to +80°C</td>
<td>0 to +70°C</td>
<td>-20 to +70°C</td>
<td>N/A</td>
<td>-25 to +70°C</td>
<td>-25 to +70°C</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sufficient</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Too big</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Fault mechanisms</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data storage at power failure</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
C Converter Schematic

Figure C.1. Principle sketch of converter
## D Tests

Table D.1. Response times for the multiplication algorithm during high acceleration, where CCW is counter clock-wise and CW clockwise

<table>
<thead>
<tr>
<th>Response time [ms] (High acceleration, multiplication)</th>
<th>Scale factor 1</th>
<th>Scale factor 1/3</th>
<th>Scale factor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CCW</td>
<td>CCW</td>
<td>CCW</td>
</tr>
<tr>
<td>0.980</td>
<td>1.840</td>
<td>1.910</td>
<td></td>
</tr>
<tr>
<td>1.030</td>
<td>2.330</td>
<td>1.920</td>
<td></td>
</tr>
<tr>
<td>1.590</td>
<td>1.530</td>
<td>1.970</td>
<td></td>
</tr>
<tr>
<td>0.805</td>
<td>1.450</td>
<td>2.570</td>
<td></td>
</tr>
<tr>
<td>0.925</td>
<td>1.645</td>
<td>1.730</td>
<td></td>
</tr>
<tr>
<td>2.125</td>
<td>1.400</td>
<td>3.540</td>
<td></td>
</tr>
<tr>
<td>0.920</td>
<td>1.730</td>
<td>1.755</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CW</td>
<td>CW</td>
<td>CW</td>
</tr>
<tr>
<td>0.840</td>
<td>1.760</td>
<td>1.670</td>
<td></td>
</tr>
<tr>
<td>1.570</td>
<td>2.180</td>
<td>1.650</td>
<td></td>
</tr>
<tr>
<td>1.175</td>
<td>1.870</td>
<td>1.990</td>
<td></td>
</tr>
<tr>
<td>0.870</td>
<td>1.830</td>
<td>1.780</td>
<td></td>
</tr>
<tr>
<td>1.015</td>
<td>1.860</td>
<td>2.030</td>
<td></td>
</tr>
<tr>
<td>1.710</td>
<td>1.930</td>
<td>1.520</td>
<td></td>
</tr>
<tr>
<td>1.570</td>
<td>1.975</td>
<td>2.620</td>
<td></td>
</tr>
<tr>
<td>Average:</td>
<td>Average:</td>
<td>Average:</td>
<td></td>
</tr>
<tr>
<td>1.223</td>
<td>1.809</td>
<td>2.047</td>
<td></td>
</tr>
</tbody>
</table>
Table D.2. Response times for the multiplication algorithm during low acceleration

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/3</th>
<th>Scale factor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCW</td>
<td>CCW</td>
<td>CCW</td>
</tr>
<tr>
<td>1,750</td>
<td>2,725</td>
<td>2,950</td>
</tr>
<tr>
<td>2,235</td>
<td>3,400</td>
<td>3,575</td>
</tr>
<tr>
<td>1,300</td>
<td>3,040</td>
<td>3,950</td>
</tr>
<tr>
<td>1,440</td>
<td>3,450</td>
<td>3,200</td>
</tr>
<tr>
<td>1,365</td>
<td>3,725</td>
<td>5,000</td>
</tr>
<tr>
<td>1,450</td>
<td>1,730</td>
<td>6,000</td>
</tr>
<tr>
<td>1,330</td>
<td>1,725</td>
<td>3.35</td>
</tr>
<tr>
<td>CW</td>
<td>CW</td>
<td>CW</td>
</tr>
<tr>
<td>1,660</td>
<td>2,775</td>
<td>2,800</td>
</tr>
<tr>
<td>2,825</td>
<td>2,850</td>
<td>3,175</td>
</tr>
<tr>
<td>1,430</td>
<td>3,040</td>
<td>3,225</td>
</tr>
<tr>
<td>1,380</td>
<td>5,100</td>
<td>4,300</td>
</tr>
<tr>
<td>1,870</td>
<td>2,890</td>
<td>3,475</td>
</tr>
<tr>
<td>1,520</td>
<td>3,015</td>
<td>3,550</td>
</tr>
<tr>
<td>1,650</td>
<td>3,065</td>
<td>3,750</td>
</tr>
<tr>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
</tr>
<tr>
<td>1,658</td>
<td>3,038</td>
<td>3,765</td>
</tr>
</tbody>
</table>
Table D.3. Response times for the extrapolation algorithm during high acceleration

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/3</th>
<th>Scale factor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCW</td>
<td>CCW</td>
<td>CCW</td>
</tr>
<tr>
<td>1,290</td>
<td>1,920</td>
<td>2,250</td>
</tr>
<tr>
<td>0,930</td>
<td>1,910</td>
<td>1,550</td>
</tr>
<tr>
<td>1,210</td>
<td>1,490</td>
<td>1,980</td>
</tr>
<tr>
<td>1,780</td>
<td>2,770</td>
<td>2,100</td>
</tr>
<tr>
<td>1,310</td>
<td>2,030</td>
<td>1,580</td>
</tr>
<tr>
<td>1,090</td>
<td>1,820</td>
<td>2,070</td>
</tr>
<tr>
<td>0,970</td>
<td>1,470</td>
<td>3,49</td>
</tr>
<tr>
<td>CW</td>
<td>CW</td>
<td>CW</td>
</tr>
<tr>
<td>1,980</td>
<td>2,120</td>
<td>1,760</td>
</tr>
<tr>
<td>1,540</td>
<td>1,900</td>
<td>4,390</td>
</tr>
<tr>
<td>1,050</td>
<td>2,420</td>
<td>1,890</td>
</tr>
<tr>
<td>0,850</td>
<td>2,060</td>
<td>2,190</td>
</tr>
<tr>
<td>0,880</td>
<td>1,730</td>
<td>3,320</td>
</tr>
<tr>
<td>1,510</td>
<td>1,970</td>
<td>2,560</td>
</tr>
<tr>
<td>2,440</td>
<td>1,910</td>
<td>2,680</td>
</tr>
<tr>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
</tr>
<tr>
<td>1,345</td>
<td>1,966</td>
<td>2,415</td>
</tr>
</tbody>
</table>
Table D.4. Response times for the extrapolation algorithm during low acceleration

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/3</th>
<th>Scale factor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCW</td>
<td>CCW</td>
<td>CCW</td>
</tr>
<tr>
<td>1,330</td>
<td>3,100</td>
<td>3,575</td>
</tr>
<tr>
<td>1,590</td>
<td>2,775</td>
<td>4,075</td>
</tr>
<tr>
<td>1,560</td>
<td>3,300</td>
<td>4,725</td>
</tr>
<tr>
<td>1,880</td>
<td>3,350</td>
<td>4,575</td>
</tr>
<tr>
<td>1,830</td>
<td>2,500</td>
<td>4,025</td>
</tr>
<tr>
<td>1,360</td>
<td>3,825</td>
<td>3,075</td>
</tr>
<tr>
<td>2,200</td>
<td>2,575</td>
<td>5,700</td>
</tr>
<tr>
<td>CW</td>
<td>CW</td>
<td>CW</td>
</tr>
<tr>
<td>1,260</td>
<td>3,025</td>
<td>3,490</td>
</tr>
<tr>
<td>1,280</td>
<td>2,925</td>
<td>3,850</td>
</tr>
<tr>
<td>1,375</td>
<td>3,200</td>
<td>2,650</td>
</tr>
<tr>
<td>1,375</td>
<td>3,000</td>
<td>2,750</td>
</tr>
<tr>
<td>2,025</td>
<td>6,250</td>
<td>3,575</td>
</tr>
<tr>
<td>1,350</td>
<td>2,975</td>
<td>4,825</td>
</tr>
<tr>
<td>1,500</td>
<td>3,375</td>
<td>4,700</td>
</tr>
<tr>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
</tr>
<tr>
<td>1,565</td>
<td>3,298</td>
<td>3,971</td>
</tr>
</tbody>
</table>

Table D.5. System drift using the multiplication algorithm at various scale factors

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/2</th>
<th>Scale factor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,001</td>
<td>0,007</td>
<td>0,001</td>
</tr>
<tr>
<td>0,001</td>
<td>0,005</td>
<td>0,003</td>
</tr>
<tr>
<td>0,002</td>
<td>0,008</td>
<td>0,004</td>
</tr>
<tr>
<td>0,001</td>
<td>0,007</td>
<td>0,004</td>
</tr>
<tr>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
<td><strong>Average:</strong></td>
</tr>
<tr>
<td>0,001</td>
<td>0,007</td>
<td>0,003</td>
</tr>
</tbody>
</table>
Table D.6. System drift using the multiplication algorithm at various scale factors

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/2</th>
<th>Scale factor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,84</td>
<td>0,34</td>
<td>1,73</td>
</tr>
<tr>
<td>1,87</td>
<td>0,11</td>
<td>0,96</td>
</tr>
<tr>
<td>1,72</td>
<td>0,12</td>
<td>0,31</td>
</tr>
<tr>
<td>1,93</td>
<td>0,11</td>
<td>0,39</td>
</tr>
</tbody>
</table>

Average: 1,84, 0,17, 0,85

Table D.7. System drift using the extrapolation algorithm at various scale factors

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/2</th>
<th>Scale factor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,001</td>
<td>0,004</td>
<td>0,002</td>
</tr>
<tr>
<td>0,005</td>
<td>0,001</td>
<td>0,004</td>
</tr>
<tr>
<td>0,001</td>
<td>0,002</td>
<td>0,002</td>
</tr>
<tr>
<td>0,005</td>
<td>0,002</td>
<td>0,004</td>
</tr>
</tbody>
</table>

Average: 0,003, 0,002, 0,003

Table D.8. System drift using the extrapolation algorithm at various scale factors

<table>
<thead>
<tr>
<th>Scale factor 1</th>
<th>Scale factor 1/2</th>
<th>Scale factor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7,7</td>
<td>6,6</td>
<td>6,5</td>
</tr>
<tr>
<td>7,2</td>
<td>6,1</td>
<td>6,3</td>
</tr>
<tr>
<td>8,7</td>
<td>6,2</td>
<td>6,4</td>
</tr>
<tr>
<td>8,1</td>
<td>6,4</td>
<td>6,6</td>
</tr>
</tbody>
</table>

Average: 7,9, 6,3, 6,5
### Table D.9. Average speed deviations at various up-scale factors for multiplication algorithm

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>298,39</td>
<td>299,23</td>
<td>0,84</td>
</tr>
<tr>
<td>2</td>
<td>299,28</td>
<td>299,11</td>
<td>-0,17</td>
</tr>
<tr>
<td>3</td>
<td>297,57</td>
<td>296,95</td>
<td>-0,62</td>
</tr>
<tr>
<td>4</td>
<td>297,22</td>
<td>296,01</td>
<td>-1,21</td>
</tr>
<tr>
<td>5</td>
<td>296,14</td>
<td>293,85</td>
<td>-2,29</td>
</tr>
<tr>
<td>6</td>
<td>296,91</td>
<td>293,38</td>
<td>-3,53</td>
</tr>
<tr>
<td>10</td>
<td>296,12</td>
<td>283,20</td>
<td>-12,92</td>
</tr>
</tbody>
</table>

### Table D.10. Average speed deviations at various downscale factors for multiplication algorithm

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>298,39</td>
<td>299,23</td>
<td>0,84</td>
</tr>
<tr>
<td>1/2</td>
<td>299,39</td>
<td>300,73</td>
<td>1,34</td>
</tr>
<tr>
<td>1/3</td>
<td>298,01</td>
<td>299,42</td>
<td>1,41</td>
</tr>
<tr>
<td>1/4</td>
<td>298,16</td>
<td>299,59</td>
<td>1,43</td>
</tr>
<tr>
<td>1/5</td>
<td>297,94</td>
<td>299,53</td>
<td>1,59</td>
</tr>
<tr>
<td>1/10</td>
<td>299,20</td>
<td>300,86</td>
<td>1,66</td>
</tr>
</tbody>
</table>

### Table D.11. Average speed deviations at various up-scale factors for extrapolation algorithm

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>297,06</td>
<td>301,08</td>
<td>4,02</td>
</tr>
<tr>
<td>2</td>
<td>297,51</td>
<td>301,41</td>
<td>3,90</td>
</tr>
<tr>
<td>3</td>
<td>298,00</td>
<td>301,59</td>
<td>3,59</td>
</tr>
<tr>
<td>4</td>
<td>299,03</td>
<td>302,59</td>
<td>3,56</td>
</tr>
<tr>
<td>5</td>
<td>299,58</td>
<td>302,93</td>
<td>3,35</td>
</tr>
<tr>
<td>6</td>
<td>298,52</td>
<td>298,75</td>
<td>0,23</td>
</tr>
<tr>
<td>10</td>
<td>299,75</td>
<td>293,35</td>
<td>-6,40</td>
</tr>
</tbody>
</table>
Table D.12. Average speed deviations at various downscale factors for extrapolation algorithm

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Input [rpm]</th>
<th>Output [rpm]</th>
<th>Deviation [rpm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>297.06</td>
<td>301.08</td>
<td>4.02</td>
</tr>
<tr>
<td>1/2</td>
<td>297.44</td>
<td>301.83</td>
<td>4.39</td>
</tr>
<tr>
<td>1/3</td>
<td>298.30</td>
<td>302.93</td>
<td>4.63</td>
</tr>
<tr>
<td>1/4</td>
<td>300.08</td>
<td>304.55</td>
<td>4.47</td>
</tr>
<tr>
<td>1/5</td>
<td>298.82</td>
<td>304.91</td>
<td>6.09</td>
</tr>
<tr>
<td>1/10</td>
<td>296.84</td>
<td>300.99</td>
<td>4.15</td>
</tr>
</tbody>
</table>

Table D.13. Average speed deviations at various velocities for current conversion

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.07</td>
<td>10.68</td>
<td>0.61</td>
</tr>
<tr>
<td>30</td>
<td>29.62</td>
<td>29.62</td>
<td>0.00</td>
</tr>
<tr>
<td>50</td>
<td>50.28</td>
<td>50.52</td>
<td>0.24</td>
</tr>
<tr>
<td>80</td>
<td>79.95</td>
<td>80.79</td>
<td>0.84</td>
</tr>
<tr>
<td>100</td>
<td>100.31</td>
<td>101.65</td>
<td>1.34</td>
</tr>
<tr>
<td>120</td>
<td>119.45</td>
<td>120.57</td>
<td>1.12</td>
</tr>
<tr>
<td>150</td>
<td>148.72</td>
<td>150.10</td>
<td>1.38</td>
</tr>
<tr>
<td>180</td>
<td>178.35</td>
<td>179.71</td>
<td>1.36</td>
</tr>
<tr>
<td>200</td>
<td>199.98</td>
<td>202.19</td>
<td>2.21</td>
</tr>
<tr>
<td>250</td>
<td>249.18</td>
<td>252.05</td>
<td>2.87</td>
</tr>
<tr>
<td>300</td>
<td>297.79</td>
<td>301.16</td>
<td>3.37</td>
</tr>
</tbody>
</table>