Dynamic Torque Control of Brushed DC Motors for Hardware-in-the-Loop Integration

Eric Schyllert

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6220
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2023

Abstract

The main goal with the thesis was to design, construct and evaluate a system able to drive and brake a mechanical axis to simulate loads and stored rotational energy - a hardware force simulator. The force simulator was then to be integrated in a existing hardware-in-the-loop testing rig to enhance the mechanical dynamics of door simulations. The force simulator was to be realised using a brushed DC motor and by controlling the motor torque. Controlling the motor torque of a brushed DC motor is achieved by controlling the motor current - and this is mainly what this thesis is about. The thesis explains the foundations of an embedded system capable to control and log the current of a brushed DC motor.

The force simulator consisted of a microcontroller, a motor card, a circuit to measure the inline motor current and a brushed DC motor. The circuit to measure the current was designed and implemented, consisting of a shunt resistor and a current sense amplifier. The shunt was placed in series (inline) with the motor. A software strategy was developed and implemented to deal with noise due to common-mode voltage transients (caused by motor control using PWM). This strategy came with a cost of introducing a measurement delay in the system.

A discrete PI-controller to control the motor current was researched and implemented. An expression to optimal tune the controller based on motor parameters and the sampling period was researched. Current control experiments were conducted but due to a calculation error the PI-controller was very untuned, this compared to the tuned expression. This error was discovered very late in the work and there was no time to redo the experiments. The experiments with the implemented untuned PI-controller were conducted and a current reference was successfully tracked, however with noise which had an amplitude of approximately 200 mA.

A strategy to record current measurements used in the current control, without missing any samples was successfully developed. It worked by using a double buffer system, filling the buffers utilizing a DMA and then written to a SD card once a buffer was filled. With the main objective to develop the capability to be able to analyze the current control algorithm.

Acknowledgements

I would like to express my heartfelt gratitude to my company supervisors Eric Warnquist and Per Byrhult for their guidance and support throughout my project. They made the daily work so much more fun. I would also like to extend my appreciation to Roland Larsson - Electronics Engineer at ASSA ABLOY, for the guidance when I faced challenges with electronics. But most importantly, for citing the quote "Assumption is the mother of all fuck ups" - a quote that was proven true numerous times throughout the course of my thesis work. I would also like to express my gratitude to all the people at ASSA ABLOY who have contributed to creating an exceptional place to work at.

To my supervisor from LTH, Anton Cervin - thank you for always being supportive. I would also like to thank Karl-Erik Årzén for being the examiner of this thesis.

An immense gratitude is expressed to my family for their unwavering support, especially to my brother who solely steered my academic path.

Contents

1

Introduction

ASSA ABLOY Entrance Systems has a wide range of automatic door systems, each product with its own unique mechanical dynamics. Today at Assa Abloy their physical testing rig does not take into account these mechanical difference between different door systems. The motor in the testing rig is not loaded with any physical load. The testing rigs are small and fairly mobile and enables developers/testers to do verification and testing of new implementations without having to access a real door system.

A more realistic testing rig would make it easier to motivate the use of it in a third party certification process of the door system. One such step in the certification process is to ensure that the control system and motors are operable in temperatures of -30 °C to 50 °C. Placing a fully installed door system in a climate chamber is a hard task - if the testing rig was used instead it would make it a lot easier and speed up the process, reduce the cost and open up the use of smaller climate chambers.

A more accurate testing rig in which a range of different door types and configurations can be hardware simulated opens up for a more dynamic and agile verification and testing process in which the control system can be tested for edge cases such as really heavy doors or a door system installed with a floor material with a really high frictional coefficient. This could speed up development time due to decreasing time from implementation to verification, make testing more accurate and increase the trust in the testing process from a third party certification point of view.

The main goal with the thesis was to design, construct and evaluate a prototype system able to drive and brake a mechanical axis to simulate loads and rotational energy stored in the system. This load simulator would be used to simulate the dynamics of doors and to be integrated in a existing hardware-in-the-loop testing rig - to capture a more realistic behaviour.

The scope was narrowed due to time constraint to create a system able to measure motor current and to log the data in real-time, and to control the current of a brushed dc motor. Due to the physics of a brushed dc motor, to control the motor current is equivalent with being able to control the motor torque which would be used to drive a mechanical axis to simulate stored rotational energy. The logging of data would be used to be able to validate the current control.

1.1 Problem formulation

The unloaded door system is simplified as to be viewed as a speed controlled motor - defined as the system motor. The unloaded system motor has dynamics that differ from when it is loaded with the physical door. The real door system can have up to two motors working together but in this thesis it is simplified as only having one. The door system is using CAN (Controller Area Network) to receive the speed reference.

To hardware simulate the stored rotational energy a force driving the mechanical axis connected to the system motor is needed. To achieve this a brushed DC motor is used. An initial rough idea of how a control system to simulate the rotational energy in the door system can be designed can be seen in Figure 1.1 in which five key components can be identified: mathematical model/recorded current measurements, CAN bus, system motor, physical mechanical axis and a simulator motor together with a control system.

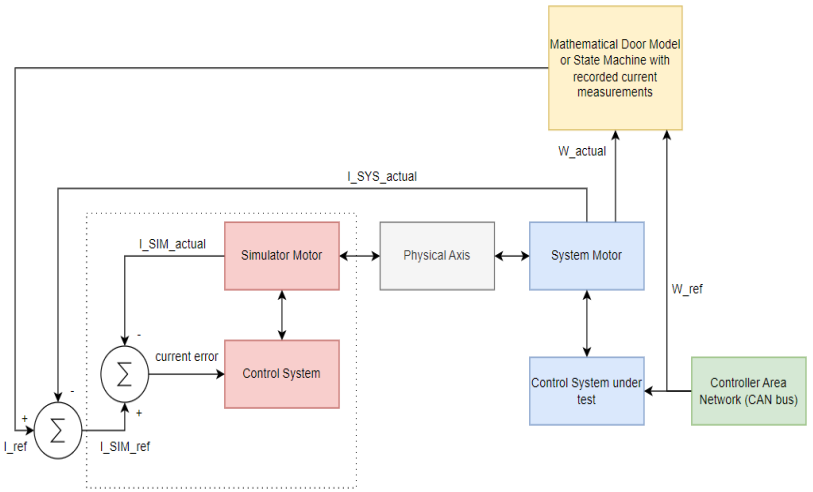

Figure 1.1 Overview of what a control system to simulate rotational energy assumed could be composed of. Five key components were identified: mathematical model/recorded current measurements, CAN bus, system motor together with the control system under test, physical mechanical axis and a simulator motor with a control system.

The idea was that the error between the mathematical model/recorded values and the actual values of the system motor was to be used as a reference to the simulator motor which would be controlled to minimize the error between the current/speed of the system motor and the model. It was limited to only compensate for

the error when it meant that the simulator motor would apply a driving force that would increase the speed of the system motor - simulating stored rotational energy.

1.2 Methodology

The work methodology during the thesis work was applied practical work with an iterative approach with each iterative phase consisting of research, prototyping and verification. Both hardware and software verification was often done with an oscilloscope by capturing signals and evaluating the result. A signal generator, a power supply unit and a multi-meter was also used in the verification process.

1.3 Thesis outline

The first part of the thesis commences with an introduction that presents the project's background, objectives and a rough idea of how this can be achieved. Chapter 2 presents background theory that a lot of the work is based on. Chapter 3 presents an overview of the system with each subsections detailing each key component that the system is composed of. Chapter 4 presents how current control of a brushed DC motor will be achieved and defines goals based on the current control. Chapter 5 discusses motor control, how to operate the motor card and relates how a current reference results in a duty cycle. Chapter 6 details how the software is designed. Chapter 7 presents the results from different experiments and calculations. Chapter 8 discusses the results presented in chapter 7. Chapter 9 contains the conclusions made from the work and research done. It also presents the limitations and what is left to do.

2

Background

2.1 Measuring current using a current shunt

To measure current using a shunt resistor the key principle is that the current flows through the shunt resistor creating a small voltage drop across the resistor that is then measured. The size of the current shunt is limited by the maximum power dissipation that it is rated to handle [Itarsiwala, n.d.] and how big intrusiveness that the shunt is allowed to have on the voltage across the motor due to the voltage drop caused by the shunt. Equation 2.1 shows how to calculate the maximum power dissipation based on resistance size and the maximum current that will flow through the shunt.

$$Power\_Dissipation(W)_{max} = R_{shunt} I^2_{load\_max} \tag{2.1}$$

2.2 Torque control - brushed DC motor

To be able to control the torque generated by the brushed DC motor a model of the torque is required. Equation 2.2 represents a simplified model of a brushed DC motor - a model that explains the dynamics of the motor current [Alaküla and Karlsson, 2011].

$$U_{dc} = RI + L\frac{di}{dt} - \omega\psi \tag{2.2}$$

where U_{dc} represents the DC input voltage, R is the electrical resistance of the motor's armature, I is the electrical current flowing through the armature, L is the inductance of the motor's armature, $\frac{di}{dt}$ is the rate of change of current with respect to time, ω is the angular velocity of the motor's armature and ψ is the magnetic flux linkage in the motor's armature.

Equation 2.3 expresses the relationship between the motor current and the motor torque [Alaküla and Karlsson, 2011] - if the goal is to control the motor torque it requires to be able to control the motor current.

$$\tau = I\psi \tag{2.3}$$

where τ represents the generated torque, I is the electrical current flowing through the motor's armature, and ψ is the magnetic flux linkage in the motor's armature.

2.3 Sampled current control - brushed DC motor

When calculating the current reference for discrete current control the concept of fast and slow computer arise. The difference is a matter of when the sampled current is used to calculate the current reference. The definition of a fast computer is when the computer is able to provide a current reference for a sampling interval, that is based on measurements sampled in the beginning of the sampling interval. On the other hand, a slow computer provides a current reference based on the measurement of current taken at the beginning of the previous sampling interval [Alaküla and Karlsson, 2011].

For sampled current control with a 'fast computer' a proportional integral feed-forward (PIF) controller is derived in [Alaküla and Karlsson, 2011]. The controller is derived based on the simplified model of the DC-motor shown in Equation 2.2 and under the assumptions listed below:

$$\bar{u}(k,k+1) = u^*(k) \tag{2.4}$$
$$i(k+1) = i^*(k) \tag{2.5}$$
$$\bar{i}(k,k+1) = \frac{i^*(k)+i(k)}{2} \tag{2.6}$$
$$\bar{e}(k,k+1) = e(k) \tag{2.7}$$
$$i(k) = \sum_{n=0}^{n=k-1} \Big(i^*(n) - i(n) \Big) \tag{2.8}$$

where k indicates the beginning of the sample interval, $\bar{u}(k,k+1)$ is the average voltage during sample k, $u^*(k)$ is the reference voltage, $i(k)$ is the measured current, $i^*(k)$ is the reference current, $\bar{i}(k,k+1)$ is the average current during sample k, $\bar{e}(k,k+1)$ is the average back-emf during sample k and $e(k)$ is the measured back-emf. Assumption 2.4 means that the reference voltage for the coming period should be the average voltage needed to achieve the desired current change. Assumption 2.5 is an assumption of "dead-beat" current control - meaning that the current error is eliminated in one sampling interval. Assumption 2.6 is argued to be true in average. Assumption 2.7 is usually true due to that the sampling frequency of the current control is magnitudes faster than the dynamics of the back-emf. Assumption 2.8

is a consequence of the assumption of "dead-beat" current control [Alaküla and Karlsson, 2011]. The PIF-controller is tuned based upon the motor parameters and the sampling time of the current control. The expression for the tuned PIF-controller is shown in Equation 2.9:

$$u^*(k) = \left(\frac{L}{T_s} + \frac{R}{2}\right) \cdot \left(\underbrace{\left(i^*(k) - i(k)\right)}_{\text{Proportional}} + \underbrace{\frac{T_s}{\frac{L}{R} + \frac{T_s}{2}} \cdot \sum_{n=0}^{n=k-1} \left(i^*(n) - i(n)\right)}_{\text{Integral}} \right) + \underbrace{e(k)}_{\text{Feed forward}} \tag{2.9}$$

where $u^*(k)$ represents the voltage reference, R is the electrical resistance of the motor's armature, L is the inductance of the motor's armature, T_s is the sampling period, $i^*(k)$ is the current reference, $i(k)$ is the measured current, e(k) is the back-emf added as a feed-forward and $\sum_{n=0}^{n=k-1}\left(i^*(n) - i(n)\right)$ represents the accumulated current errors.

Equation 2.10 shows the basic equation for a discrete PI controller derived using Euler transformation [Bengtsson, 2020].

$$u(k) = K_p\left(e(k) + \frac{T_s}{T_I}\sum_{n=0}^{n=k} e(n)\right) \tag{2.10}$$

where $u(k)$ is the discrete control signal, $e(k)$ is the error between the reference and the actual value of the sampled process, $\sum_{n=0}^{n=k} e(n)$ is the accumulated errors, K_p is the amplification, T_I is the integration time and T_s is the sample time.

By comparing Equation 2.9 with Equation 2.10 and setting $e(k) = i^*(k) - i(k)$, $e(n) = i^*(n) - i(n)$ we see that, if we disregard the feed-forward in Equation 2.9, that these two equations are almost the same. The boundary of the summation differs with one sample. We identify that $K_p = \frac{L}{T_s} + \frac{R}{2}$ and $T_I = \frac{1}{\frac{L}{R} + \frac{T_s}{2}}$.

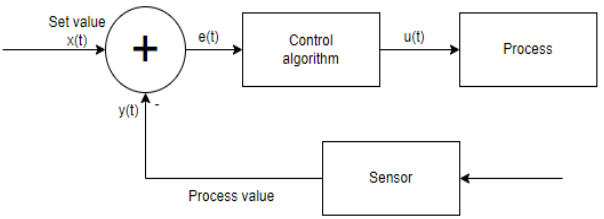

Figure 2.1 Basic control system - inspired by [Bengtsson, 2020]

2.4 Simplified model of the mechanics of a brushed DC motor

If the rotor of a brushed DC motor is simplified as single axis lumped inertia drive then the motion is described by Equation 2.11 [Leonhard, 2012]

$$J\frac{d\omega}{dt} = m_M(\omega,t) - m_L(\omega,t) = m_a(\omega,t) \tag{2.11}$$

where J represents the moment of inertia of the rotor, ω represents the angular velocity of the rotor, t represents time, $m_M(\omega,t)$ represents the torque due to the motor current, $m_L(\omega,t)$ represents the torque due to the load and $m_a(\omega,t)$ represents the resulting torque on the system. Such a simplified single axis lumped inertia can be seen in Figure 2.2.

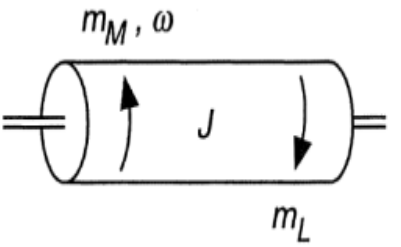

Figure 2.2 This figure illustrates a simplified model of the mechanics of the rotor when simplified as a single axis lumped inertia drive - inspired by [Leonhard, 2012]

3

System overview

The system marked with dashed lines in Figure 1.1 is the system that will be used to simulate stored rotational energy. This system consist of a microcontroller, a motor setup with encoder, an evaluation module (DRV8704EVM) based on the motor driver DRV8704, a micro SD card, a current measuring circuit and a circuit to supply pull-up resistors for the signals from the motor encoder. This is the system that was worked on in this thesis and an overview can be seen in Figure 3.1.

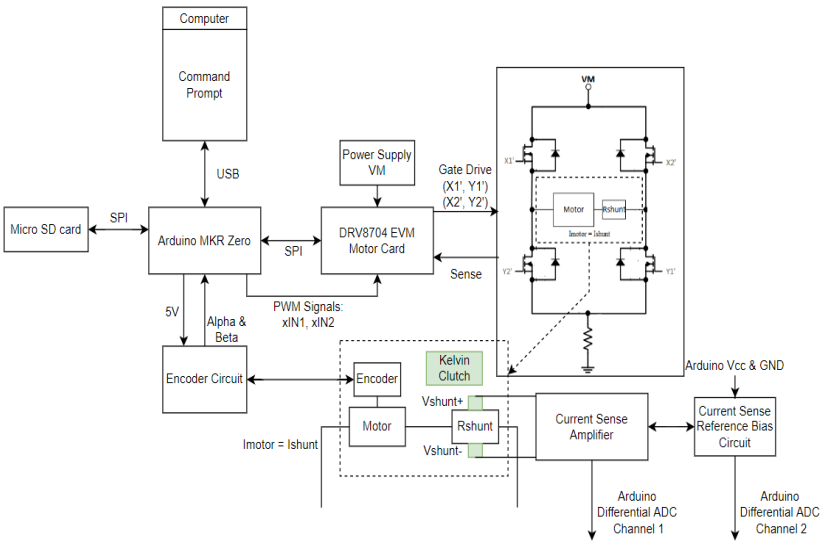

Figure 3.1 System overview of a system capable of current control in a dc motor and real-time

To simplify the process of the laboratory work a command prompt was made which enabled the user to send input through the serial port to the microcontroller to carry out commands and to display reply messages. All time critical operations are interrupt driven which means that the command prompt is not able to steal valuable processor time from important operations. To get a better understanding of the system a list with information about all interrupts can be seen in Table 6.1. Writing data to the SD card can be seen as a time critical operation - it is not interrupt driven - and can thus get CPU time hijacked from interrupts. The interrupts are designed to be executed as fast as possible and the importance of being quick increases with increasing calling frequency.

3.1 Motor setup

An initial existing setup with two brushed DC motors connected on same axis was used as the basis of the thesis work which can be seen in Figure 3.2.

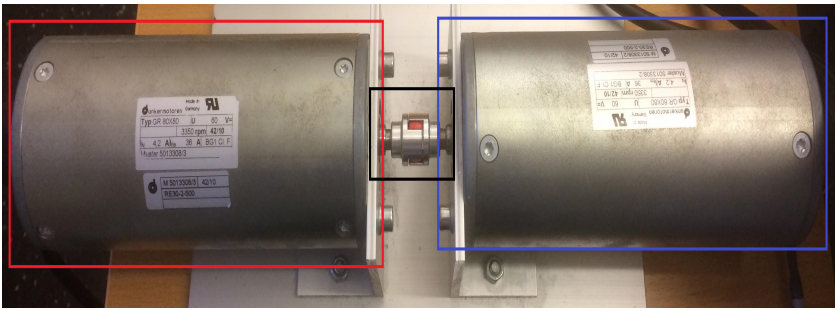

Figure 3.2 Existing setup with two brushed DC motors connected on same axis. Mechanical axis marked in black and motors color-coded - red defined as simulator motor and blue as system motor.

3.2 Motor driver card

A customer evaluation module designed around the motor driver DRV8704 was purchased and used in the project. The DRV8704 is a dual H-bridge brushed DC motor driver, it is highly configurable and supports running two different brushed DC motors.

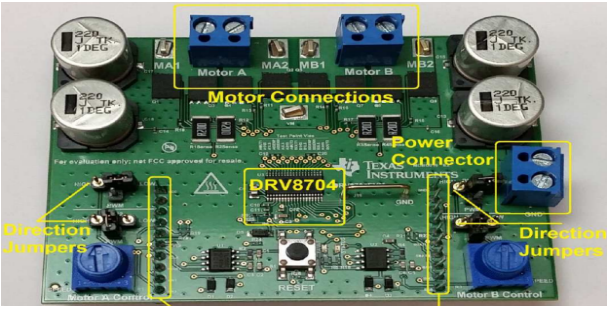

Figure 3.3 Figure showing the DRV8704EVM - a platform built to be able to drive two different brushed DC motors with variable current limiting using a dual H-bridge

3.3 Microcontroller

The Arduino MKR Zero was selected and used in this project to act as the microcontroller with the purpose to orchestrate and carry out necessary calculations and operations. The processor of the Arduino MKR Zero is the Arm Cortex-M0 32-bit SAMD21 processor. The processor contains peripherals with key functions such as timers, signal generators and analog-to-digital converters that were utilized during the prototype building. An overview of the microcontroller and how it is connected can be seen in Figure 3.4.

3.4 Micro SD card

Due to the Arduino MKR Zero having a limited amount of memory - 256KB Flash, 32KB SRAM [*SAM D21 Family Data Sheet* 2018] an external 32 GB flash memory was purchased and used. This to be able to log current measurements sessions of varying lengths, to later be able to verify the results of the current control algorithm.

3.5 Operating the system

To operate the system the command prompt is used, common usage could be: turn on/off the motor, calibrate no current voltage, set/get a current reference, start a session to log data to the SD card, display errors from the DRV8704EVM or to configure the DRV8704EVM - a list of all commands can be seen in Table 3.1.

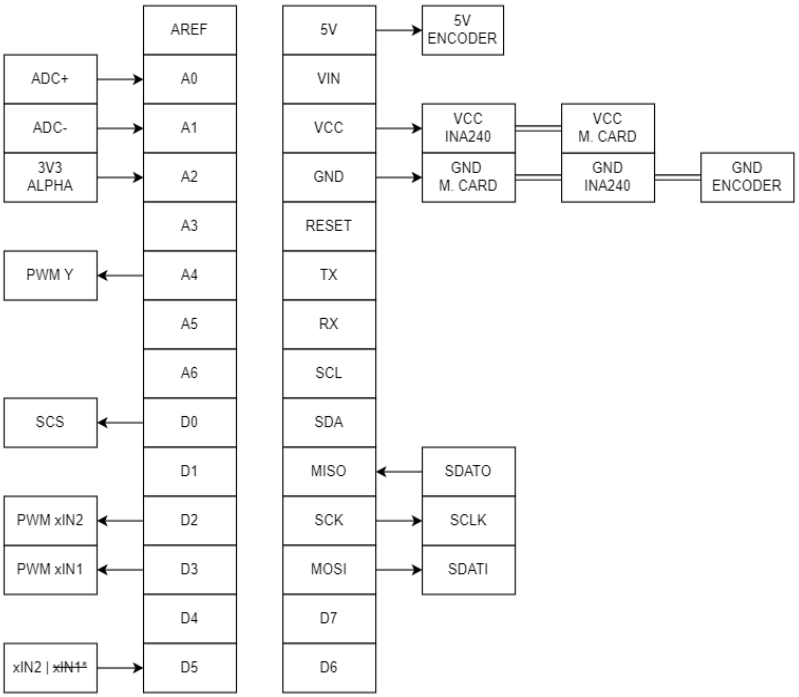

Figure 3.4 Overview of Arduino pins and what they are connected to. *Only one of the PWM signals goes to D5 due to lack of OR circuit.

Command	Arg 1	Arg 2	Return Value	Side Effects	Description
motor	on/off		void	changes register CTRL	Turn the motor on or off
readReg	[0-7]		int [0-65535]		Read selected register from DRV8704
writeReg	[0-7]	[0-4096]	int [0-65535]		Write to selected register of DRV8704
current	read	on	void	Writes to a predefined file on the SD card, reads a global variable	Starts storing the voltage used to calculated the duty cycle to a SD card
current	get		int	Reads a global variable	Retrieves the current reference
current	set	x [x<=0]	void	Writes to a global variable	Sets the current reference, currently only implemented to set negative values
current	calibrate		int	Changes register CTRL	Turns off the motor, waits until it is stopped and then measures the no current voltage, updates a global variable with the new value

Table 3.1 Table with information about all commands implemented in the system to use in the command line used to operate the system

4

Current Control

To be able to control the motor current it is necessary to be able to measure the actual motor current. A system to measure the motor current - with sufficient amplitude resolution and sample frequency was necessary to be designed and implemented. The current control in the real door system served as a requirements baseline - with an current amplitude resolution of roughly estimated 400 mA and a PWM frequency of 25 KHz - in which the duty cycle is updated at a frequency of 5 KHz.

The main goal was to be able to measure the inline motor current with precision of around 400 mA, bidirectional currents of maximum 16.5 A in both direction and be designed to be able to handle higher currents up to 32 A without breaking. The ambition was to create a prototype fast that could be evaluated and later improved upon. The current measurement had to be able to measure bidirectional currents due to the real door system being able to rotate in both directions.

The shunt size was chosen to be able to handle twice the maximum current of the maximum current of the motor in the real door system with Equation 2.1 in mind. The goal was to be able to use the same measuring circuit for the real system and a motor operated at twice the maximum current (twice as strong) without breaking. The motivation was to prototype faster and not having to design more than one measurement circuit.

4.1 Current shunt placement

The DC motor can be seen as an inductive load and will resist a change in the current flowing through it and it is common practice to place the motor in a H-bridge. There are three potential locations that the current shunt can be placed to measure the current - low side (in series with ground), high side (in series with the voltage supply) or inline (inside the H-bridge in series with the motor). The only shunt placement that measures the current going through the motor at all times is the inline placement with which the true phase current can be known which gives us the ability to monitor the system. The inline implementation however comes with

the problem that if using a current sense amplifier it has to support high common-mode rejection [Bridgmon and Andrews, 2016].

4.2 Analog-to-digital convert with the Arduino MKR Zero

The Arduino MKR Zero has an ADC peripheral that was configured to sample periodically at 250 K samples/s with 12 bit resolution and in differential mode - calling an interrupt every time a measurement is made. The Arduino's 3.3 V output was used as the voltage reference. The requirement on the ADC was to be able to catch the dynamics of the motor current such that a closed loop current control could be realized. The initial goal was to have a closed loop current control with an update frequency of 5 KHz. The Nyquist's theorem state that the sample frequency has to be twice that of the signal to be recreated [Glad and Ljung, 2003] - meaning a lower limit of sampling frequency of 10 KHz. The possibility to average measurements to reduce noise was also a motivation to keep the sampling frequency high.

It was observed that the voltage measured with the Arduino ADC had a constant gain error, this was verified by measuring the voltage with a voltmeter and then comparing it with the voltage measured with the Arduino - for different voltages. The real voltage was that of 87.5% of what the Arduino measured, Equation 4.1 compensate for this gain error:

$$Voltage_{real} = 0.875 \cdot ADC_{voltage} \tag{4.1}$$

For an n bit ADC the number of discrete digital levels that it can produce is given by:

$$ADC_{levels}(n\ bits) = 2^n \tag{4.2}$$

The resolution of the smallest voltage that can be quantified is then given by:

$$ADC_{resolution}(n\ bits) = \frac{Reference\ Voltage}{ADC_{levels}(n\ bits)} = \frac{Reference\ Voltage}{2^n} \tag{4.3}$$

4.3 Current sense amplifier

Due to worldwide semiconductor chip shortage it was very hard to find components and some compromises in design had to be made. One such compromise was that the Current Sense Amplifier INA240 was only available in the INA240A3DR version - which has a gain of 100. The high gain meant that a small shunt had to be used to not saturate the output of the current sense amplifier. A shunt of one milli Ohm was purchased and deemed "good enough" - due to the time consuming process of finding components that were in stock. The Current Sense Amplifier INA240A3DR has variable supply voltage and can be operated from 2.7 V to 5.5 V - this was one of the parameters in consideration when selecting Current Sense Amplifier with the goal to operate it at 3.3 V. The Arduino MKR Zero used in the project has a 3.3 V logic level and with a risk of damage if operated at higher voltages. Not operating the Current Sense Amplifier at a higher voltage level than 3.3 V reduce the risk of an accident in which the output from the Amplifier destroys the Arduino. The maximum voltage range of the ADC in the Arduino MKR Zero is 3.3 V, the current Sense Amplifier operated at 3.3 V is then making full use of the entire range of the ADC.

The motor current is symmetrical and bidirectional which means that:

$$I_{max} = -I_{min} \tag{4.4}$$

The ADC is only able to measure non-negative voltages and to be able to measure negative voltages the Current Sense Amplifier has to be biased such that one part of the voltage range is used to represent negative currents and the other part positive currents. The Current Sense Amplifier was biased at half the supply voltage due to the currents being symmetrical. The region of no output saturation is then described by:

$$0 \leq \frac{V_{supply}}{2} + I_{shunt} R_{shunt} \cdot gain_{100} \leq V_{supply} \tag{4.5}$$

From 4.5 the maximum current without output saturation is described by:

$$I_{max} \leq \frac{V_{supply}}{2 R_{shunt} gain_{100}} \tag{4.6}$$

Calculating the maximum current according to Equation 4.6 when $V_{supply} = 3.3\,V$, $R_{shunt} = 1\,m\Omega$ one obtains:

$$I_{max} = \frac{3.3\,V}{2 \cdot 1 \cdot 10^{-3}\Omega \cdot 100} = 16.5\,A \underset{Equation\ 4.4}{\Longrightarrow} I_{min} = -16.5\,A \tag{4.7}$$

Using Equation 4.3 for a 12 bit ADC and a reference voltage of 3.3 V, the voltage level resolution for the ADC is calculated as:

$$\frac{3.3\,V}{2^{12}} = 0.806\,mV \tag{4.8}$$

The smallest input voltage to the Current Sense Amplifier that the ADC is then able to quantify and thus also the smallest current this corresponds to when using $R_{shunt} = 1\,m\Omega$ is given by:

$$\frac{0.806\,mV}{gain_{100}} = 8.06\,\mu V \implies I = \frac{8.06\,\mu V}{1m\Omega}\,A = 8.06\,mA \tag{4.9}$$

This means that the smallest current the system theoretically can measure without regarding any errors is 8.06 mA.

4.4 Current sense dead-zones and PWM dependency

The current sense amplifier - INA240 can sense drops across shunt resistors over a wide common-mode voltage range from -4 V to 80 V [*INA240* 2016]. The common-mode voltage range is important due to the current shunt being inline with the motor and due to this will be exposed to the large DC voltage used to supply the motor.

The enhanced PWM rejection provides high levels of suppression for large common-mode transients ($\frac{\Delta V}{\Delta t}$) [*INA240* 2016] - these are caused by the switching and occur when the PWM-signal changes state that changes the state of the H-bridge.

The common-mode transients due to a common-mode voltage step can be seen in Figure 4.1 and Figure 4.2 and there is a zone in which the current sense amplifier output is completely unusable due to having a very large error - this area could be regarded as a measuring dead-zone. The common-mode steps occur when the PWM signal changes state: from low to high or high to low.

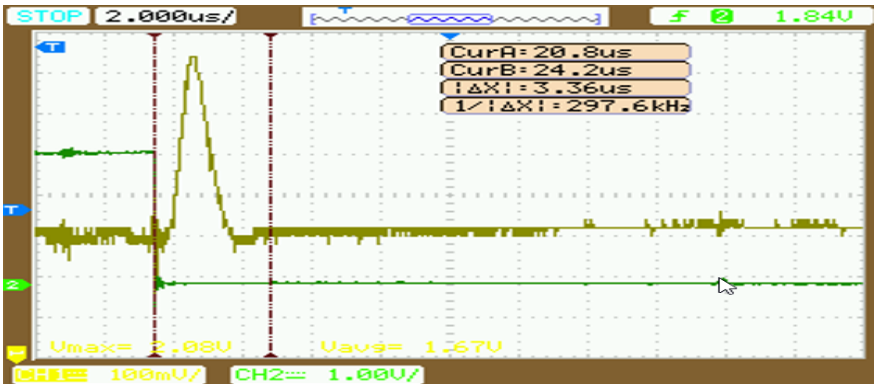

Figure 4.1 Print-screen from oscilloscope: Green signal shows the PWM-signal, olive green signal shows the output from the current sense amplifier. The area inside the red cursors enclose the time span with very large output error from the current sense amplifier caused by a change in the state (high to low) of the PWM-signal.

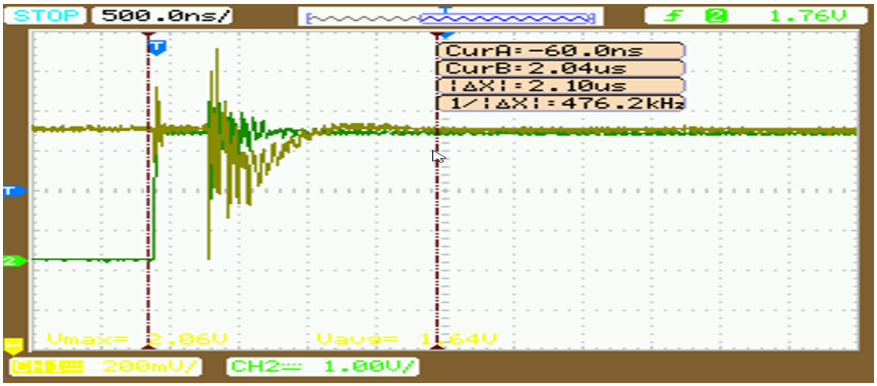

Figure 4.2 Print-screen from oscilloscope: Green signal shows the PWM-signal, olive green signal shows the output from the current sense amplifier. The area inside the red cursors enclose the time span with very large output error from the current sense amplifier caused by a change in the state (low to high) of the PWM-signal.

To deal with the "dead-zones" a strategy to discard measurements was implemented. There can be up to two dead-zones during every PWM-period. One dead-zone is stationary and can only occur at the start of the PWM-period. The other dead-zone is time varying and dependent of the duty cycle - it can occur at any time during the PWM-period. To deal with the stationary dead-zone three samples are always skipped at the start of the PWM-period - independent if needed or not. It was identified that the time varying dead-zone only could occur in the transition from high to low PWM-signal. To deal with time varying dead-zone a interrupt attached to an analog input of the Arduino was implemented. The interrupt tells the ADC to skip three samples, it is triggered by falling edge of the analog input and the analog input is fed the PWM-signal - which is when the varying dead-zone is active.

4.5 Current control overview

If we could achieve current control with "fast computer" a current control model was proposed and can be seen in Figure 4.3 and it is inspired by Equation 2.9 and Figure 2.1. The controller is a discrete PI-controller which outputs a voltage reference. The back-emf is calculated and added as a feed-forward to the voltage reference. The sample time of the loop was assumed to be in the range of 4-6 kHz while the calculated back-emf would be updated with a frequency of 100-200 Hz. The duty cycle algorithm is to control the H-bridge in such a way that the supply voltage in average is the voltage reference across the motor during one PWM period.

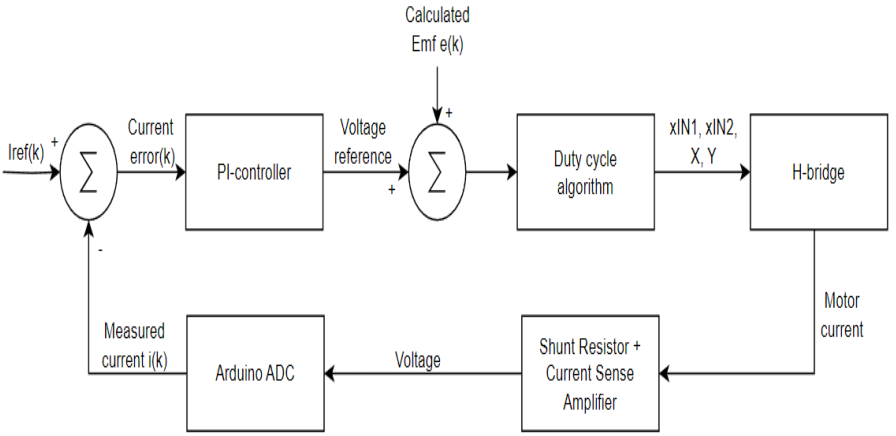

Figure 4.3 Overview of the closed current control loop in the case of "fast computer"

4.6 Challenges in current control of unloaded DC motors

Using Equation 2.11 as a model for the rotor and inserting the force produced by the motor we get Equation 4.10 which describes the acceleration of the rotor.

$$\frac{d\omega}{dt} = \frac{\tau_{motor} - \tau_{load}}{J} \tag{4.10}$$

where τ_{motor} is the force produced by the DC-motor, τ_{load} is the force counteracting the motor - such as friction, J the moment of inertia of the rotor and $\frac{d\omega}{dt}$ is the rate of change of the motor speed.

Rewriting Equation 2.2 to get the expression of the rate of change of the motor current one obtains:

$$\frac{di}{dt} = \frac{U_{dc} - RI - \omega\psi}{L} \tag{4.11}$$

When a DC motor is unloaded, there is very little mechanical load on the motor shaft, and the motor can spin at a very high speed with very little current flowing through the electrical resistance of the motor's armature. This can be seen by combining Equation 4.11 and Equation 4.10 - if a small current accelerate the rotor speed to a high speed the resulting back-emf will be large and oppose the flow of current through the motor's armature. The back-emf will limit how big the current can grow.

5

Motor Control

To supply the voltage and current required by the motors, a motor driver circuit is necessary. This circuit also serves to separate and protect the microcontroller from the high currents and voltages used with DC motors. The DRV8704 used in the project can handle a supply voltage of 8 V to 52 V. The chosen motor card can be configured via SPI, making it easy to modify its behavior without needing to replace hardware. Another highly beneficial feature of the motor driver card is its capability to report errors. This information can be accessed through the SPI interface, enabling easy detection and resolution of issues without the need for hardware replacement.

5.1 Motor control strategy

xIN1	xIN2	Motor x Output
High	Low	Full speed forward current
High	High	Brake, Low-side slow decay
Low	Low	Coast, H-bridge disabled
Low	High	Full speed reverse current
High	PWM	Reverse current + Slow decay
PWM	High	Forward current + Slow decay
Low	PWM	Reverse current + Fast decay
PWM	Low	Forward current + Fast decay
PWM	PWM	No output
No Connection	No Connection	External controller connected

Figure 5.1 Table of how to operate the motor using the motor driver DRV8704 and two PWM-signals xIN1 and xIN2. The red zones marks the modes used in the project.

To operate the motor, two PWM signals (xIN1 and xIN2) and two GPIO signals (X and Y) were utilized. The motor output was controlled with the PWM-signals according to Table shown in Figure 5.1. The GPIO signals X and Y were used to control each of the half-bridges which can be seen in Figure 5.2

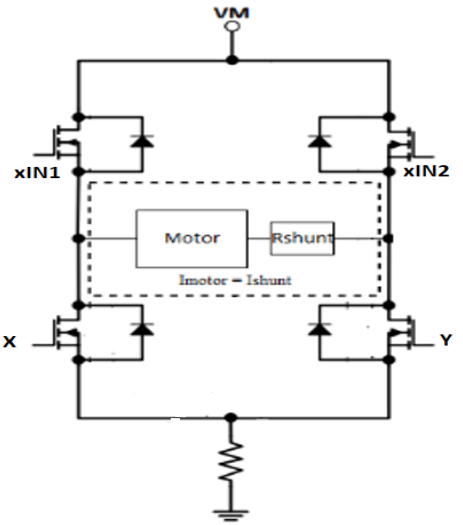

Figure 5.2 Overview of signals controlling the H-bridge - PWM-signals: xIN1, xIN2 and GPIO-signals: X, Y

An algorithm of how to control the PWM-signals and GPIO-signals based on the voltage reference $u^*(k)$ was constructed. The algorithm is based upon activating forward current (xIN1 and Y used), reverse current (xIN2 and X used) or H-bridge disabled. The algorithm can be seen in Equations 5.1, 5.2, 5.3 and 5.4.

$$D_1(u^*(k)) = \begin{cases} \frac{u^*(k)}{V_{VM}}, & 0 \leq u^*(k) \leq V_{VM} \\ 1, & u^*(k) > V_{VM} \\ 0, & u^*(k) < 0 \end{cases} \tag{5.1}$$

$$D_2(u^*(k)) = \begin{cases} -\frac{u^*(k)}{V_{VM}}, & 0 > u^*(k) \geq -V_{VM} \\ 1, & u^*(k) < -V_{VM} \\ 0, & u^*(k) \geq 0 \end{cases} \tag{5.2}$$

$$X(u^*(k)) = \begin{cases} 1, & u^*(k) \geq 0 \\ 0, & u^*(k) < 0 \end{cases} \tag{5.3}$$

$$Y(u^*(k)) = \begin{cases} 0, & u^*(k) \geq 0 \\ 1, & u^*(k) < 0 \end{cases} \tag{5.4}$$

where V_{VM} is the DC supply voltage, $D_1(u^*(k))$ controls the duty cycle of PWM-signal xIN1 and $D_2(u^*(k))$ controls the duty cycle of PWM-signal xIN2.

5.2 Encoder

An incremental encoder attached on the motor rotor was used - producing two square-wave signals phased at 90 degrees which are defined as the alpha and beta signals and that can be used to determine the rotational direction of the motor and motor position. Each motor revolution produces 500 alpha and beta impulses.

The encoder is using a logical level of 5 V which is to high too be used directly to the Arduino MKR zero which uses 3.3 V logic. The encoder is using an open-drain output that can only sink current, meaning we need an external source to supply current when the signal is high. Figure 5.3 shows the final iteration of the encoder circuit with a voltage dividers to transform the 5 V logic down to approximately 3.3 V and with pull-up resistors to supply current when the signal is high.

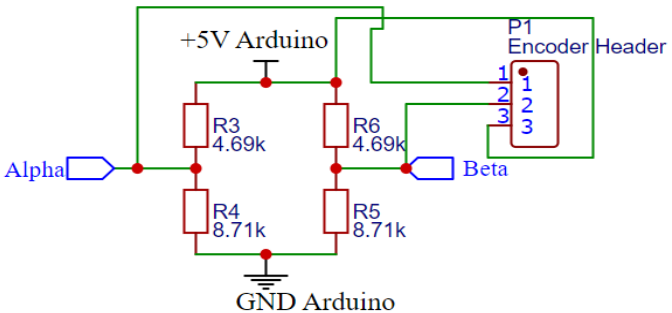

Figure 5.3 Shows the encoder circuit with voltage dividers and pull-up resistors - with the encoder output marked as Alpha and Beta

6

Software design

6.1 Software overview

The software is designed to run on a single core microcontroller and were everything time critical is interrupt driven. The software is designed to control a control system that controls the simulator motor. The control system can be seen in the red box in Figure 1.1. Setting the correct priority of interrupts is crucial in order to ensure that the most time-critical actions are performed with little to no delay. Table 6.1 shows the interrupts used in the control system, their priority and if they were periodic or event driven.

6.2 Direct memory access controller

The direct memory access controller (DMAC) contains both a direct memory access (DMA) engine and a cyclic redundancy check (CRC) engine. The DMAC is used to move data between memory and peripherals and is able to do this without the involvement of the CPU [*SAM D21 Family Data Sheet* 2018].

The motivation behind using the DMAC was mainly to offload the CPU but also to use it as a tool to synchronize the flow of certain operations - by having events generate interrupts, events such as a buffer being filled. This was later utilized to write to a SD-card in real time.

6.3 PWM generation

Two instances of the Timer/Counter for Control applications (TCC) peripherals in the microcontroller were configured to perform waveform generation - to create three PWM signals. By configuring the counters with the same internal clock and prescaler and configuring the counters to count clock pulses results in the two counters being synchronized. The output of the wave generated is controlled by a register (CCx) that the current count value is compared with, if the current count value is

Interrupt Name	Interrupt Priority	Description	Triggered by
TCC0_Handler()	0	Calculates and sets the duty cycle every fourth PWM period.	Periodic every 41.67 μs
TCC1_Handler()	1	Tells the ADC to ignore x amount of measurements, a strategy to deal with the stationary dead zone	Periodic every 41.67 μs
ADC_Handler()	0	Called every time the ADC makes a measurement	Periodic every 4 μs
TC4_Handler()	3	Calculates the motor speed based on the change of alpha counts between each TC4_Handler() interrupt call	Periodic every 200 ms
skipSamples()	Same as the number of the pin used	Tells the ADC to ignore x amount of measurements - strategy to deal with varying dead zone	Event: triggered by falling PWM edge max every 41.67 μs
alphaInterrupt()	3	Used to count alpha pulses from encoder	Event: triggered by rising edge from encoder - max every 35.82 μs
DMAC_Handler()	0	Used to set flags indicating that a buffer is filled	Event: called when DMAC buffered filled

Table 6.1 Showing information about all interrupts implemented in the system

higher than the register CCx the output is low otherwise the output is high. The counter will count up to the value set by the register PER and then restart the count from zero.

Equation 6.1 express the relationship between the counters clock frequency, prescaler and the maximum count value (PER) and the frequency of the generated PWM signal [*SAM D21 Family Data Sheet* 2018]. The counters were configured to use a 48 MHz clock frequency, prescaler=1 and maximum count value PER=2000 which using Equation 6.1 results in a PWM frequency of 24 KHz.

$$f_{PWM} = \frac{f_{clock}}{Prescaler \cdot PER} = \frac{48MHz}{1 \cdot 2000} = 24KHz \tag{6.1}$$

Equation 6.2 expresses the relationship between the compare value CCx and the maximum count value PER that the counter is configured to use and the duty cycle of the generated PWM signal.

$$Duty\ cycle = \frac{CCx}{PER} \tag{6.2}$$

One of the counters (TCC1) was configured to generate two of the three PWM signals - these two signals share the same count value but their output are respectively controlled by different compare registers (CC0 and CC1). The two PWM signals are defined as xIN1, xIN2 and are fed to the motor driver - thus the signals controlling the DC motor. The signals can be seen in overview Figure 3.1 and in Figure 5.1 in which they show how their state relate to the motor output. As the count value reaches the maximum count value PER an interrupt is called - in the case of the counter TCC1, TCC1_handler() is called, more information about the interrupt can be seen in Table 6.1.

6.4 Strategy to go from the case "slow computer" to "fast computer"

To motivate the use of the 'fast computer' model the assumption was that the duty cycle calculations had to be done close in time to their actual usage. To achieve this a second counter called TCC0 was introduced. The TCC0 was fed the same clock as counter TCC1 - thus synchronized but placed out of phase to be triggered a time before the TCC1 counter. The reason was to have time to calculate and set the duty cycle of the next PWM period of TCC1 before TCC1 starts outputting the next PWM period, the process is shown in Figure 6.1. An other important reason to do this is to be able to use current measurements taken closer in time to the next PWM period. If the interrupt TCC1_Handler() would have been used to calculate and set the duty cycle - then the current measurements would have been at least one PWM period delayed from when the new duty cycle could have been used.

By updating the duty cycle every fourth PWM period the resulting sample period of the duty cycle is described by Equation 6.3.

$$T_s = 4 \cdot T_{PWM} \tag{6.3}$$

Using the proposed strategy with synchronized counters to time the calculation of the new duty cycle - the time between starting to calculate the new duty cycle and when the new duty cycle is put in use is described by Equation 6.4.

$$dt = \frac{OFFSET}{PER} \cdot T_{PWM}, \quad \text{where OFFSET} \leq \text{PER} \tag{6.4}$$

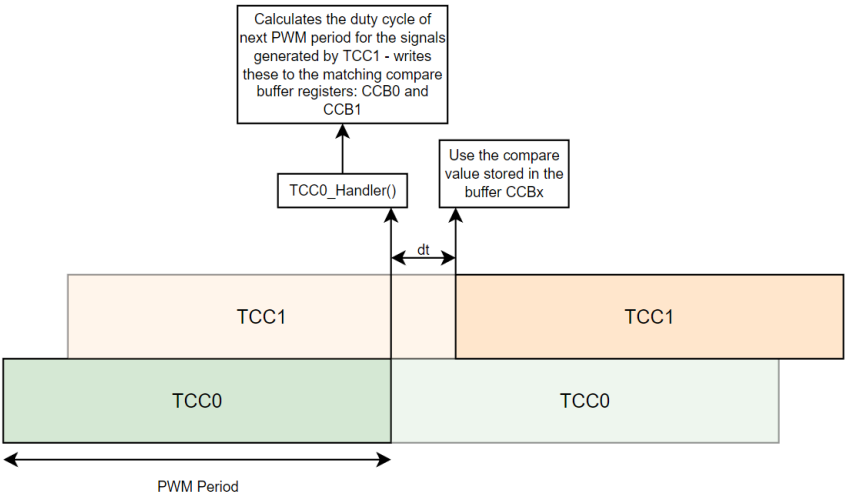

Figure 6.1 Strategy using synchronized counters to time the calculation of the next duty cycle closer in time to its usage

where dt is the time to calculate the new duty cycle, OFFSET is the added count value to offset the second counter TCC0, PER is the maximum count value and T_{PWM} is the period of the PWM-signal.

How the sample period of the duty cycle relates to the calculation time and the period of the PWM signal can be seen in Figure 6.2. By comparing the time to calculate the duty cycle expressed in Equation 6.4 in relation to the sample time of the duty cycle which is expressed in Equation 6.3 - we get the relative duty cycle calculation delay which is described in Equation 6.5

$$\frac{dt}{T_s} = \frac{\frac{OFFSET}{PER} \cdot T_{PWM}}{4 \cdot T_{PWM}} = \frac{OFFSET}{4 \cdot PER}, \quad \text{where OFFSET} \leq \text{PER} \tag{6.5}$$

where $\frac{dt}{T_s}$ is the relative calculation delay. A relative calculation delay of one would be the case "slow computer" discussed in the background chapter and a relative delay of approximately zero would be the case of "fast computer". This under the assumption that the current measurement used in the duty cycle calculation is taken in the beginning of the calculation.

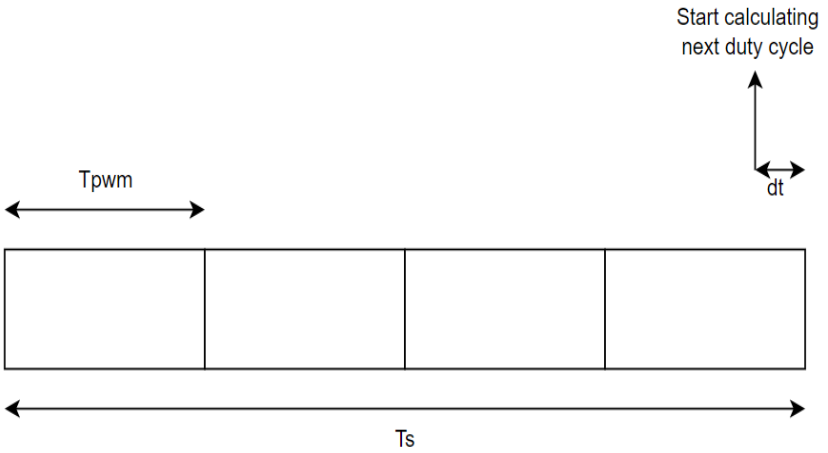

Figure 6.2 Overview of how the sample period T_s of the duty cycle is related to the period of the PWM signal T_{PWM}. The figure shows the time interval dt that represents the delay from when the duty cycle is calculated until it is put in use.

6.5 Current average calculations

Values from the ADC were continuously updating a global voltage average. A bit-shift averaging algorithm was used to calculate the average voltage and can be seen in Equation 6.6. A fast algorithm was assumed to be needed due to having voltage measurement taken periodically every 4 μs.

$$\begin{cases} acc = (voltage\_avg(k) << n) - voltage\_avg(k)) \\ voltage\_avg(k+1) = (acc + measurement) >> n \end{cases} \tag{6.6}$$

acc denotes the accumulator variable, *measurement* denotes the newest input sample from the ADC and n is a factor determining how much the new measurement will impact the average. This algorithm is commonly used to perform digital filtering of data samples in real-time applications, where the goal is to smooth out the signal while minimizing processing overhead. The resulting filter introduces a delay in the system.

6.6 Strategy to write to SD card in real-time

The goal was to store the voltage used in every duty cycle calculation without missing any values. The duty cycle is updated with a frequency of 6 KHz but the actual interrupt is called with a frequency of 24 KHz and in every fourth interrupt the duty cycle is calculated.

The interrupt is called every 6 KHz and the data to be stored is 4 bytes in size, the generated data rate is then calculated as:

$$6 \cdot 4 \frac{KB}{S} = 24\ KB/s \tag{6.7}$$

The rate of the data generated in Equation 6.7 is a slower than the write speed to the SD card seen in Table 7.1, which is required to not miss generated data. Designing for the worse case scenario the maximum writing latency from Table 7.1 is 12.368 ms but this is in a benchmark environment and not tested in the load simulator environment - in which CPU time is taken by interrupts. Assuming a maximum latency of 250 ms, a minimum array size required to be able to store all samples if each sample is taken with a frequency of 6 KHz is calculated as:

$$Buffer_{min} = Latency_{max} \cdot \frac{variables}{time} = 250\ ms \cdot 6\ KHz = 1500 \tag{6.8}$$

A double buffer system was designed and implemented to be able to save data with the DMA while writing to the SD card simultaneously. The buffer size was chosen to 2560 with Calculation 6.8 in mind. The time to fill the buffer with the DMA saving one value with a frequency of 6 KHz is calculated as:

$$\frac{2560}{6KHz} = 426.66\,ms \tag{6.9}$$

During this time the SD card with a latency of 250 ms and write speed of 480 KB/s can write an amount of bytes calculated as:

$$(426.66 - 250)\,ms \cdot 480\,KB/s = 84796.8 \approx 84796\,bytes \tag{6.10}$$

The amount of bytes that an array of datatype Long and size 2560 takes up is calculated as:

$$2560 \cdot 4\,byte = 10240\,bytes \tag{6.11}$$

To structure the data such that it can be understood when exported from the SD card extra symbols are required to be written, if an extra 4 bytes is added with every data point then the total data to be written to the SD card from an array of size 2560 is calculated as:

$$2560 \cdot (4+4)\,bytes = 20480\,bytes \tag{6.12}$$

The time to fill one buffer of 2560 samples when samples are stored with a frequency of 6 KHz is calculated according to Equation 6.9 to 426.66 ms. In theory, the double buffer system should work because the amount of data required to empty a buffer of 2560 samples, including extra characters, is determined to be 20480 bytes according to Equation 6.12. This quantity is smaller than the data that can be written to the SD card in 426.66 ms according to Equation 6.10. Therefore, it is feasible to implement the double buffer system, which allows for continuous data filling in one buffer while the other buffer is being written to the SD card.

From the command prompt the user starts the data logging, this activates the DMA, the DMA is configured such that it is triggered by the interrupt that calculates the duty cycle - when triggered it saves the voltage used for the duty cycle calculation. A flowchart of the cyclic process of the DMA filling two buffers can be seen in Figure 6.3. The DMA interrupt is called when a buffer is filled, a flag is set indicating that it is ready to be written to the SD card. The interrupt checks if the SD card is done writing to the next buffer - if it is not done an error is incremented, indicating that samples might have been missed due to the DMA filling and SD writing is overlapping - the flowchart of this erroneous behaviour can be seen in Figure 6.5 and Figure 6.4 shows the desired behaviour.

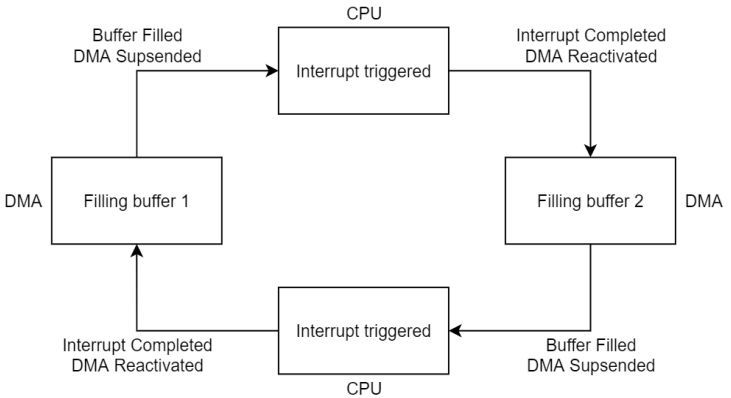

Figure 6.3 Flowchart of DMA configured in cyclic mode and set to suspend the DMA and trigger a DMA interrupt when a buffer is filled - the interrupt executes some operations and then reactivates the DMA and the process of filling the next buffer is started.

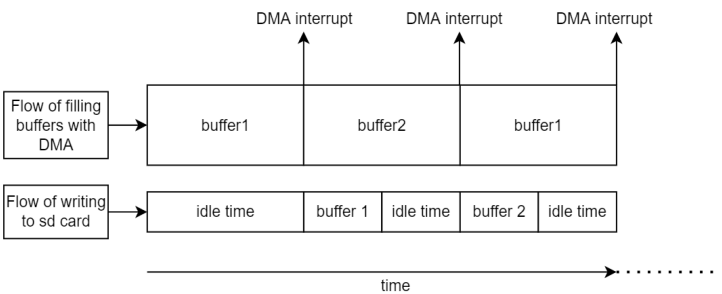

Figure 6.4 Time chart of how the DMA and writing to SD card operates in time when working as expected.

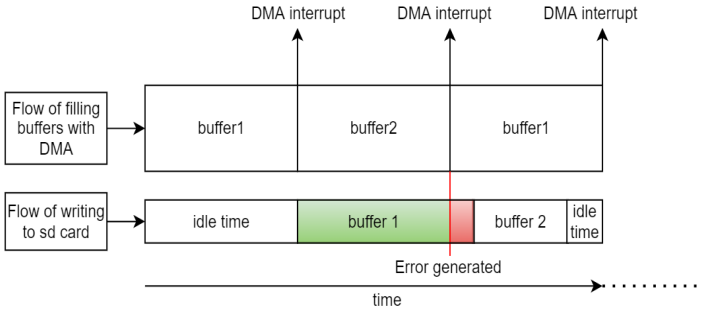

Figure 6.5 Time chart of how the DMA and writing to SD card operates when a buffer takes to long to write to the SD card and an error is generated. The SD card is writing from the buffer that the DMA has started to fill with new data - this result in the risk overwriting data not yet recorded by the SD card.

7

Results

7.1 Final iteration of current measurement setup

The final iteration of the current measurement setup consists of a current shunt placed inline with the motor - connected via a Kelvin clutch. The voltage across the shunt is then fed to a bidirectional current sense amplifier together with the ADC of the Arduino configured in differential input. The a schematic of the system can be seen in Figure 7.1.

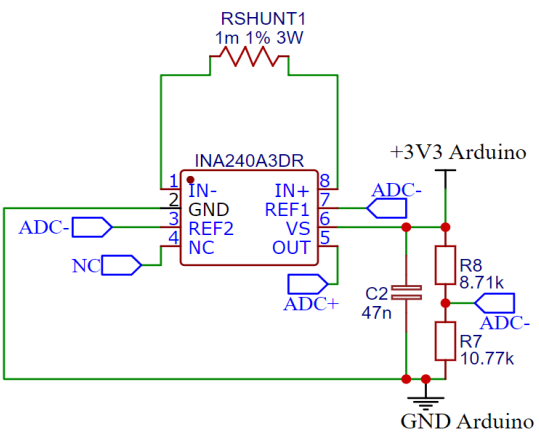

Figure 7.1 Schematic of the current measuring circuit - ADC+ and ADC- goes to the ADC differential inputs of the Arduino.

7.2 Dead-zone strategies

To evaluate the effectiveness of the dead-zone strategies three different experiments were conducted. Both the varying and stationary dead-zone strategy were configured to skip two samples. At the time of the experiment the strategy to write to a SD card in real-time using DMA had not been implemented. Due to this the measurements were taken once the motor had reached a stationary speed for a given constant duty cycle. Each experiment contained of two different measuring intervals, each measuring interval captured continuous samples. The first interval was current measurements when the duty cycle was set to 0%, this to show the noise not caused by a switching PWM-signal. The second measuring interval was with a duty cycle of 75% this to show the noise caused by a switching PWM-signal.

The first experiment was conducted with no algorithm to skip samples, the second experiment with the algorithm to deal with the stationary dead-zone and the third experiment with the algorithm to skip both the varying and stationary dead-zone. In Figure 7.2 measurements were made without any method to disregard dead-zone samples, in Figure 7.3 the method to disregard samples from the stationary dead-zone was implemented and in Figure 7.4 both the method to disregard dead-zone samples from the stationary and the varying dead-zone were implemented.

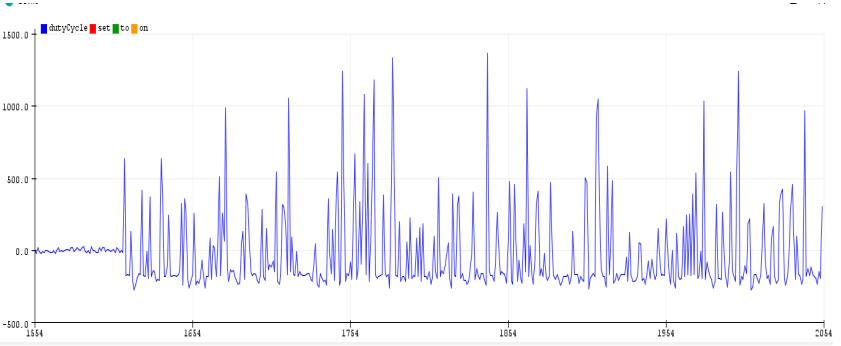

Figure 7.2 Print Screen from the Arduino IDE, no implemented method to deal with dead-zones. Note: the y-axis is the current measured in mA, the X-axis is discontinuous and can be broken up in two continuous parts one part (x values from 1554 to 1620) showing continuous samples with duty set to zero and the other part (x values from 1621 to 2054) continuous samples with duty set to 75%. The blue graph shows the measured current.

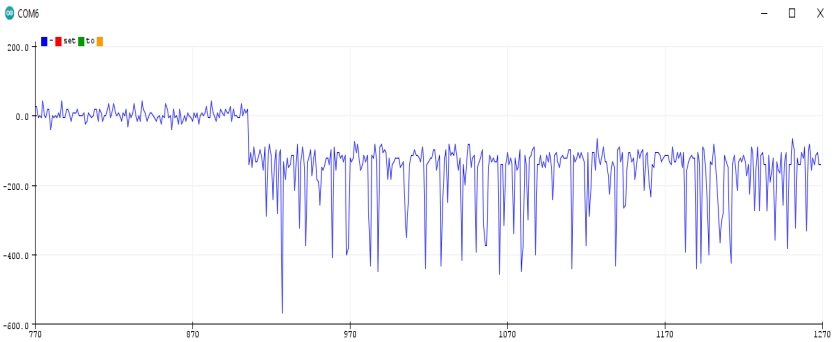

Figure 7.3 Print Screen from the Arduino IDE, Stationary dead-zone skip method implemented. Note: the y-axis is the current measured in mA, the X-axis is discontinuous and can be broken up in two continuous parts one part (x values from 778 to 910) showing continuous samples with duty set to zero and the other part (x values from 911 to 1270) continuous samples with duty set to 75%. The blue graph shows the measured current.

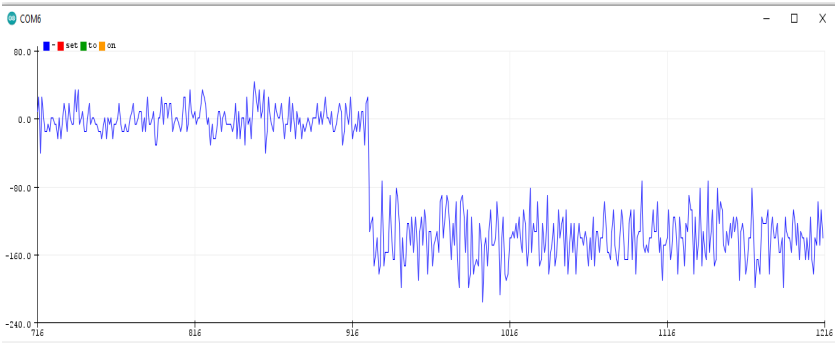

Figure 7.4 Print Screen from the Arduino IDE, Stationary and varying dead-zone skip methods implemented. Note: the y-axis is the current measured in mA, the X-axis is discontinuous and can be broken up in two continuous parts one part (x values from 716 to 920) showing continuous samples with duty set to zero and the other part (x values from 921 to 1216) continuous samples with duty set to 75%. The blue graph shows the measured current.

7.3 SD card benchmark evaluation: assessing write speed and latency

An open-source library [Greiman, 2021] was used to test the write speed and writing latency of the micro SD card and the result from the tests can be seen in Table 7.1. The test was done when the microcontroller ran a benchmark software and not the

load simulator software.

	Write Speed	Max Latency	Min Latency	Average Latency
Iteration 1:	488.14 KB/s	4875 us	1044 us	1046 us
Iteration 2:	487.57 KB/s	12368 us	1044 us	1047 us

Table 7.1 SdFat micro SD card benchmark results

7.4 Writing to SD card in real-time using DMA

Using two buffers of size 2560 the strategy to write to SD-card in real time worked without generating any errors. This while logging measurements from the voltage used to calculate the duty cycle - which was recorded with a frequency of 6 KHz.

7.5 Current average calculations

A moving averaging filter was used to smooth out the motor current measurements. The filter used is presented in Equation 6.6 and the speed of the filter used was set to n=1. The resulting filter is derived in Equation 7.1:

$$\begin{cases} acc = (voltage\_avg(k) << 1) - voltage\_avg(k)) \\ voltage\_avg(k+1) = (acc + measurement) >> 1 \end{cases} \tag{7.1}$$

Using this filter results in that new average is the average between the old average and the new measurement. This is a filter using recursive feedback and thus categorized as an Infinite Impulse Response filter (IIR-filter). A difference equation for the filter is presented in Equation 7.2:

$$y(n) = 0.5 \cdot y(n-1) + 0.5 \cdot x(n) \tag{7.2}$$

where y(n) is the new average, y(n-1) is the old average and x(n) is the measurement.

7.6 Strategy to go from the case "slow computer" to "fast computer"

Using a PWM frequency of 6 KHz and calculating the duty cycle every fourth PWM period the resulting sample time of duty cycle calculation is calculated by inserting the values in Equation 6.3:

$$T_s = 4 \cdot T_{PWM} = 4 \cdot \frac{1}{6 \cdot 10^3 Hz} = 166.67 \mu s \tag{7.3}$$

The time needed to calculated the duty cycle was experimentally determined and set to 20 % of the maximum count value PER - using Equation 6.4 the time delay is calculated as:

$$dt = \frac{OFFSET}{PER} \cdot T_{PWM} = \frac{0.2 \cdot PER}{PER} \cdot T_{PWM} = 0.2 \cdot T_{PWM} \tag{7.4}$$

The relative duty cycle calculation delay is then calculated using the Equation 6.5 and inserting the values from calculation 7.3 and 7.4 as:

$$\frac{0.2 \cdot T_{PWM}}{4 \cdot T_{PWM}} = 0.05 \tag{7.5}$$

7.7 Current control with blocked rotor

In the background section we compared Equation 2.9 with Equation 2.10 and setting $e(k) = i^*(k) - i(k)$, $e(n) = i^*(n) - i(n)$ we see that, if we disregard the feed-forward in Equation 2.9 that these two equations are almost the same. The boundary of the summation differs with one sample. We identify that $K_p = \frac{L}{T_s} + \frac{R}{2}$ and $T_i = \frac{1}{\frac{L}{R} + \frac{T_s}{2}}$. Inserting the motor parameters and sampling time with the motor resistance of 1 Ohm, motor inductance of 6.9 mH and a sample period of $\frac{1}{6KHz} = 166.67 \mu s$ we get:

$$K_p = \frac{L}{T_s} + \frac{R}{2} = \frac{6.9 \cdot 10^{-3}}{166.67 \cdot 10^{-6}} + \frac{1}{2} = 41.9 \tag{7.6}$$

$$T_i = \frac{1}{\frac{L}{R} + \frac{T_s}{2}} = \frac{1}{\frac{6.9 \cdot 10^{-3}}{1} + \frac{166.67 \cdot 10^{-6}}{2}} = 143.2 \tag{7.7}$$

Inserting the calculated T_i and K_p in expression for the discrete PI-controller described in Equation 2.10 and recognizing that the output $u(k)$ is the voltage reference $u^*(k)$ given in Equation 2.9 we get:

$$u^*(k) = K_p \Big(e(k) + \frac{T_s}{T_i} \sum_{n=0}^{n=k} e(n) \Big) = 41.9 \Big(e(k) + \frac{166.67 \cdot 10^{-6}}{143.2} \sum_{n=0}^{n=k} e(n) \Big) \tag{7.8}$$

Using a supply voltage of 30 V and inserting the calculated voltage reference from Calculation 7.8 into the algorithm to calculate the duty cycle found in Equation 5.1 and 5.2 we get:

$$D_1(u^*(k)) = \begin{cases} \frac{u^*(k)}{30}, & 0 \leq u^*(k) \leq 30 \\ 1, & u^*(k) > 30 \\ 0, & u^*(k) < 0 \end{cases} \tag{7.9}$$

$$D_2(u^*(k)) = \begin{cases} -\frac{u^*(k)}{30}, & 0 > u^*(k) \geq -30 \\ 1, & u^*(k) < -30 \\ 0, & u^*(k) \geq 0 \end{cases} \tag{7.10}$$

If the calculated duty cycle is a value between 0 and 1 then to generate a matching PWM signal we rewrite Equation 6.2 to get an expression for how to calculate the compare register based on a given duty cycle - this is shown in Calculation 7.11

$$D \cdot PER = CCx \tag{7.11}$$

The final proposed current control algorithm with a discrete PI-controller, tuned with the motor parameters, able to control the current in both directions can be found in Appendix A.1. It is based upon combining calculations 7.8, 7.9, 7.10 and 7.11.

Due to time constraint the entire control algorithm covering both PWM signals was not implemented, this because a solution to handle both varying dead-zones was not implemented in time. Instead the algorithm was limited to only PWM-signal, meaning that the supply voltage could only be applied in one direction and thus only able to control torque in one direction. A calculation error that was not discovered until the writing of this thesis meant that current control experiments that had been carried out - had been done without a tuned PI-controller. The untuned PI-controller had a $K_p = 126 \cdot 10^3$ and $T_i = 7$. Figures 7.5 and 7.6 shows the current control with the simplified untuned PI-controller and a blocked rotor. The data is logged using the double buffer strategy using a DMA to write to a SD-card, which is presented in Section 6.6.

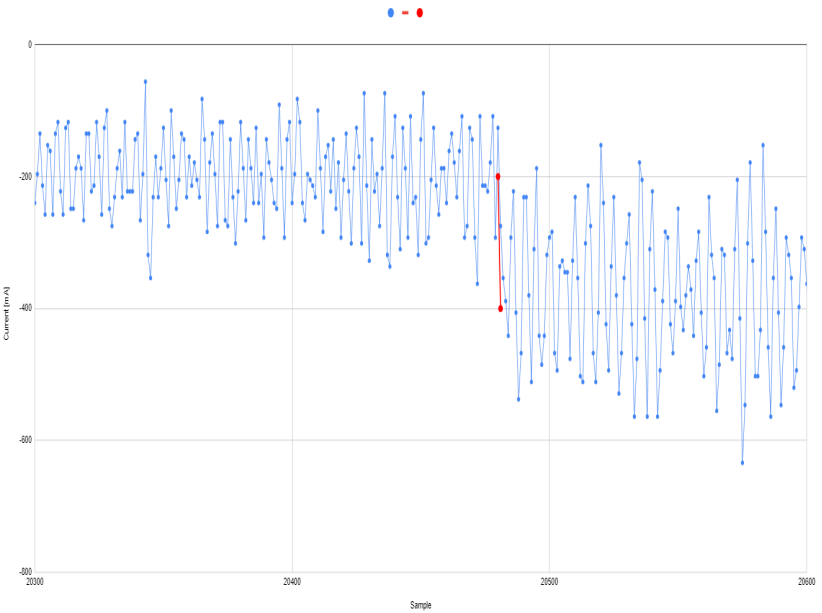

Figure 7.5 The Figure shows current measurements captured with the DMA and saved to the SD-card - with the blue dots representing measured current, red dot indicating the current reference - with the red line indicating a change in the current reference from -200 mA to -400 mA.

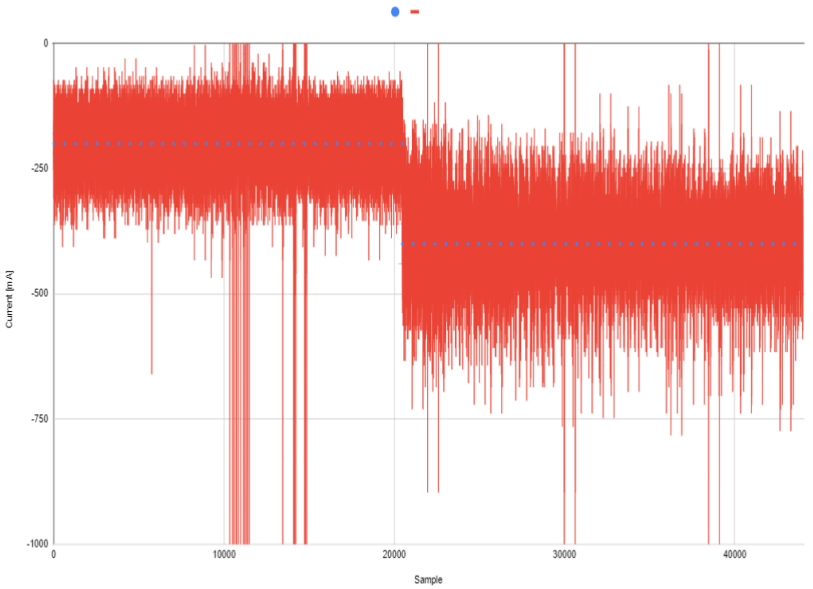

Figure 7.6 The Figure shows current measurements captured with the DMA and saved to the SD-card - with the blue dots representing the current reference and the red lines indicating the measured current.

8

Discussion

8.1 Final iteration of current measurement setup

The final current measurement setup is designed to measure bidirectional currents of up to 16.5 A. However no such experiments were done and the largest current that was measured was around 10 A - this due to being limited by the amount of current that the power supply could generate. Noise in the system not related to the common-mode transients caused by the switching could have been further investigated. To verify the current measurement setup a third party tool should have been used to measure the motor current. The measurement done with the third party tool should then have been compared with the measurement done with the system designed in this thesis. This to be able to verify the constructed current measurement setup - this was however not done.

8.2 Dead-zone strategies

It can be argued that the implementation of dead-zone strategies to reduce the noise generated by common-mode transients effectively worked. This based upon comparison of the noise levels shown in Figure 7.4, in which the full strategy to deal with both the varying and stationary dead-zone had been implemented, with Figure 7.2, where none of the dead-zones issues had been addressed. The implementation of dead-zone strategies comes at the cost of skipping measurement samples. The amount of samples that are skipped during a PWM-period is determined by the duty cycle. Fewer samples are skipped if the varying and stationary dead-zone coincides or if there is no switching during the PWM-period. The noise not related to common-mode transients can be seen in the first section of Figure 7.4 where the duty cycle is set to zero.

A higher PWM frequency amplifies the problem with noise generated by common-mode transients due to increased switching. Moreover, increasing the PWM frequency decreases the number of ADC measurements that can be taken during a period - for a given sampling rate. Less measuring samples aggravates the

negative impact of having to skip a fixed amount of samples. This imposes an upper limit on how fast the PWM frequency can be when using this strategy.

Skipping samples introduces an varying delay in the current measurements that are used to calculate the next duty cycle. The effect of this potential delay has not been fully investigated. The implemented control system used an ADC sampling frequency of 250 KHz, resulting in one sample being taken every 4 μs. In contrast, the period of a single PWM signal is approximately 41.7 μs, allowing for potentially 10-11 measurements being taken during one PWM period. In the the worst-case scenario, 6 measurements are skipped during one PWM period when implementing the strategy to skip both the varying and stationary dead-zones.

8.3 Current averaging calculations

Introducing the averaging filter presented in Calculation 7.1 introduces a delay in the system. The effect of this filter has not been investigated or included in any models due to time constraints. The filter is an infinite impulse response filter (IIR-filter) and the difference equation of the filter is presented in Equation 7.2. It is assumed that this filter can have a big impact on the control systems ability to control the motor current and thus it would be of great value to analyze the consequences further. It would also be a good idea to introduce an upper and lower value on the measurements to reduce the impact of noise - measurements that fall outside this range are then truncated. This could have helped to reduce the impact of large measurement noise spikes. Such spikes can be seen in Figure 7.6. This was however not done.

8.4 SD card benchmark evaluation: assessing write speed and latency

The test was done when the microcontroller ran a benchmark software and not in the load simulator system, due to this it only indicates the best possible performance.

8.5 Writing to SD card in real-time using DMA

The strategy to write to the SD-card in real time was successful this based on being able to measure the current used to calculate the duty cycle without generating any errors - which was the goal. An other way to verify this would have been to have a simplified known algorithm that was to be recorded using the implemented DMA strategy and then post-analysing the captured data stored in the SD-card. This last verification method was however never done.

8.6 Strategy to go from the case "slow computer" to "fast computer"

As discussed in the background chapter the definition of a "fast computer" is when the computer is able to provide a current reference for a sampling interval that is based on measurements sampled in the beginning of the sampling interval. This can be broken up in two parts, first that we make sure that the next duty cycle is calculated close in time to its use and the second part is that the current measurement is taken close in time to the calculation. Starting with the delay of when we start to calculate the duty cycle until when it is put in use. This calculation delay can be seen in Calculation 7.4 and is calculated to $0.2 \cdot T_{PWM} = 8.3\ \mu s$. Relating this calculation delay to the total sampling period of the duty cycle is done in Calculation 7.5, it is calculated to be a delay of 5% of the sample period. In the ideal case of a "fast computer" there would be no delay, however 5 % can be argued to be a small enough delay such that it can be said that the duty cycle is calculated close enough in time to its use to be regarded as the case of "fast computer".

Investigating the current measurement delay we recognize that this delay is the distance from the last current measurement to when the duty cycle is started to be calculated. The ADC takes a measurement every 4 μs, this means that without any filtering or averaging a potential delay of up to 4 μs is present. However both a moving averaging filter and a strategy to skip samples to deal with the dead-zones were used in the system, which both introduces delays. In the worst-case scenario, the varying dead-zone strategy is timed in such a way that the next three adjacent samples, preceding the calculation of the duty cycle, are skipped. This would introduce a delay of up to 16 μs from the last current measurement. Disregarding the effect of an averaging filter this would mean a combined delay of up to 24.3 μs, calculated as the sum of 12 μs (delay caused by skipping three adjacent samples), 8.3 μs (duty cycle calculation delay) and 4 μs (potential delay until next measurement). This total delay is approximately 14.6 % of the duty cycle period when disregarding the effect of the averaging filter and assuming the worse case. The current control model shown in Figure 4.3 does not compensate for this delay. To reduce the total delay a faster processor could be used (to reduce the duty cycle calculation delay) and the sampling rate of the ADC could be increased (to reduce the maximum delay between measurements). The number of samples to be skipped when dealing with the dead-zones could be experimented with and perhaps be better tuned to reduce the delay it introduces - as this was not thoroughly experimented with. The effect of the delayed caused by the filter was not investigated.

8.7 Current control with blocked rotor

In Section 4.6 challenges with current control of unloaded DC motors is discussed. To simplify the experiment to control the motor current to be able to follow a current

reference the motor rotor was blocked. This to prevent the motor from accelerating and changing the dynamics of the motor current by the induced back-emf. It can be argued that this will be somewhat the case when the control system (the red box) is used to simulate a load in the system presented in Figure 1.1. This due to the system motor (blue box) trying to control the connected physical axis to achieve a speed reference - and thus applying a counter-force to achieve a fixed speed. However this has not been investigated or experimented with and will require a more in depth analysis of how it could work.

A PI-controller tuned based on motor parameters and the sample period of the duty cycle was calculated. However due to not having implemented a way to deal with the varying dead-zone strategy for both PWM signals a simplified control algorithm was used instead. This algorithm only utilizes one of the half-bridges in the H-bridge and can thus only apply current (torque) in one direction. Due to a calculation error the current control experiments had been done with an untuned PI-controller. The tuned PI-controller has a $K_p = 41.9$ and a $T_i = 143.2$, the untuned PI-controller that was used in the experiments has a $K_p = 126 \cdot 10^3$ and a $T_i = 7$. This is a K_p that is approximately 3000 times bigger than the tuned controller, meaning a very small current error will most likely result in outputting the maximum duty cycle and completely overtake the control algorithm. Figures 7.5 and 7.6 shows the experiment of the current control with the untuned PI-controller. Even though the PI-controller is very badly tuned, it is possible to see that the controller is able to track the current reference. It would have been very interesting to see the how the tuned algorithm would perform - however due to time constraints there were no time to conduct such experiments.

9

Conclusion

The project was not a further development on an already existing system and was designed and built from scratch. A time consuming process, but benefited from not being locked to certain solutions or components. A lot of work was mainly conducted to research, to understand and implement basic functionality - such as how to operate the peripherals of the Arduino MKR Zero.

The goal to create a system that could hardware simulate the stored rotational energy in a door has not been achieved. This due to no such experiments has been conducted. Current control of a brushed DC motor was partially achieved which was assumed to be the foundation to generate a driving force that could be used to simulate stored rotational energy. Current control was only partially achieved due to being able to control the current in one direction and not both. A very untuned PI-controller was used in current control experiments and could follow a current reference but with heavy oscillation. The oscillations had an amplitude of approximately 200 mA. It was not established if these oscillations were due to the control algorithm being untuned or due to measuring noise in the system. A strategy with double buffers and utilizing a DMA was implemented and tested and was successfully able to record the current used in the current control algorithm. The captured current was stored to a SD-card, so that it later could be analysed.

A current measuring circuit was designed and implemented with the purpose to be able to measure the inline motor current. The measuring circuit worked on the principle of measuring the voltage drop across a shunt resistor. The circuit was designed to be able to handle bidirectional currents of 16.5 A due to this the shunt resistance had to be small. A current sense amplifier was utilized to amplify the voltage drop across the shunt to be able to utilized the full voltage range of the ADC. The current sense amplifier was also selected due to its ability to reject common-mode voltage transients. Common-mode voltage transients are created due to the switching of the H-bridge. The current sense terminals either experience the supply voltage or ground and the switching of the H-bridge causes both of these terminals to rapidly change potential. The common-mode transients caused a large disturbance in the output of the current sense amplifier and is referred to as a measuring dead-zone in this thesis. It was analysed that up to two dead-zones could appear

during a PWM-period. The first dead-zone could only occur in the beginning of the PWM period and due to this was named the stationary dead-zone. The second dead-zone could occur at any time during the PWM period and was thus named the varying dead-zone. The time a dead-zone lasted was measured and a strategy to deal with them was implemented. The strategy was based upon skipping measuring samples. The stationary dead-zone was dealt with by always skipping the first samples at the start of a PWM period. The varying dead-zone was dealt with by having the PWM signal as an input to the microcontroller. A transition in the PWM signal triggered an interrupt that caused the system to skip samples. This method dealt with the disturbances caused by the common-mode voltage transients effectively but came at the cost of introducing a delay in the current measurements.

9.1 Limitations

Due to time constraint the current control was limited to the scenario with blocked rotor and only using one direction of the H-bridge.

9.2 Future work

The PI-controller needs to be tuned and the step response of the system needs to be properly investigated and analyzed. Add anti-windup to the integral part of the PI-regulator. Attach the second PWM-signal to an interrupt pin of the Arduino to deal with the varying dead-zone from the second PWM signal. Implement the full proposed algorithm to be able to control the motor current in both directions. Verify that the algorithm for handling dead-zones works when operating both half-bridges in the H-bridge. Investigate how the implemented averaging filter effects the current control.

Improve the speed calculation to include getting the direction of the rotation. Add the back-emf as a feed forward to extend the PI-controller to a PIF-controller and then analyse the impact this has on the step response.

Add support to measure the current in one more motor - preferably reuse the existing solution. Upgrade the microcontroller to contain peripherals with at least two internal differential ADC (to measure the current in the load/dynamics simulator and the system motor) and preferably two or more DMA. Evaluate the method of using the DMA to collect current measurement and saving these to a SD-card. Compare by introducing an external tool to do motor current measurements.

Start to research how the force simulator can be integrated in a HIL-rig. Analysis which operating loads the load simulator can be operated at and what the limiting factors for these are.

A

Code

A.1 Tuned control algorithm

$K_p = 41.9$, $T_i = 143.2$, $V_{VM} = 30$ V and $T_s = 166.67$ μs

```
currentNow = (avgVolt * ARDUINO_GAIN_ERROR - noCurrVolt) * 10; // [mA]
currentError = currentRef - currentNow; // [mA]
errorSum = errorSum + currentError; // [mA]
dutyCycle = (currentError * 1.397*10^-3  + 1.625*10^-9*errorSum)

if (dutyCycle => 0)
{
  REG_TCC1_CCB0 = 0;
  REG_TCC1_CCB1 = min(dutyCycle, 1) * 2000;
  X = 1;
  Y = 0;
}
else
{
  REG_TCC1_CCB0 = -max(dutyCycle, -1) * 2000;
  REG_TCC1_CCB1 = 0;
  X = 0;
  Y = 1;
}
```

A.2 Simplified and untuned control algorithm

$K_p = 126 \cdot 10^3$, $T_i = 7$, $V_{VM} = 30$ V and $T_s = 166.67$ μs

```
currentNow = (avgVolt * ARDUINO_GAIN_ERROR - noCurrtVolt) * 10; // [mA]
currentError = currentRef - currentNow; // [mA]
errorSum = errorSum + currentError; // [mA]
dutyCycle = (currentError * 4.2  + errorSum/10)

if (dutyCycle => 0)
{
  REG_TCC1_CCB0 = 0;
  REG_TCC1_CCB1 = min(dutyCycle, 1) * 2000;
  X = 1;
  Y = 0;
}
else
{
  REG_TCC1_CCB0 = 0;
  REG_TCC1_CCB1 = 0;
  X = 0;
  Y = 0;
}
```

Bibliography

Alaküla, M. and P. Karlsson (2011). *Power Electronics: Devices, Converter, Control and Applications*. Department of Industrial Electrical Engineering and Automation.

Bengtsson Lars, 1. (2020). "Implementation of control algorithms in small embedded systems." *Engineering* **12**:9, pp. 623–639. ISSN: 1947-3931. URL: https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsswe&AN=edsswe.oai.gup.ub.gu.se.296432&site=eds-live&scope=site.

Bridgmon, J. and C. Andrews (2016). *Current sensing for inline motor-control applications*. URL: https://www.ti.com/lit/an/sboa172/sboa172.pdf?ts=1653421285476. (Accessed: 2022-05-25).

Glad, T. and L. Ljung (2003). *Reglerteori - Flervariabla och olinjära metoder*. Studentlitteratur.

Greiman, B. (2021). *Sdfat version 2*. URL: https://github.com/greiman/SdFat. (Accessed: 2022-06-05).

INA240 (2016). sbos662c. INA240 –4-V to 80-V, Bidirectional, Ultra-Precise Current Sense Amplifier With Enhanced PWM Rejection. Texas Instruments. URL: https://www.ti.com/lit/ds/sbos662c/sbos662c.pdf?ts=1653987373741. (Accessed: 2022-05-25).

Itarsiwala, R. (n.d.). *Getting started with current sense amplifiers: choosing an appropriate shunt resistor*. URL: https://training.ti.com/getting-started-current-sense-amplifiers-session-4-how-choose-appropriate-shunt-resistor?context=456802-1138857-456803. (Accessed: 2022-05-25).

Leonhard, W. (2012). *Control of Electrical Drives*. Springer Berlin Heidelberg. ISBN: 9783642976469. URL: https://books.google.se/books?id=0%5C_XvCAAAQBAJ.

SAM D21 Family Data Sheet (2018). ds40001882d. Low-Power, 32-bit Cortex-M0+ MCU with Advanced Analog and PWM. Microchip Technology Inc. URL: https://content.arduino.cc/assets/mkr-microchip_samd21_family_full_datasheet-ds40001882d.pdf?_gl=1*12ox7kk*_ga*MTQ4NTMyNzM2Ni4xNjQyNTgxMzU1*_ga_NEXN8H46L5*MTY1MzQ5ODE4MC4xNTQuMS4xNjUzNDk4MTgyLjU4. (Accessed: 2022-05-25).

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
October 2023

Document Number
TFRT-6220

Author(s)
Eric Schyllert

Supervisor
EricWarnquist, ASSA ABLOY Entrance Systems AB
Per Byrhult, ASSA ABLOY Entrance Systems AB
Anton Cervin, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Dynamic Torque Control of Brushed DC Motors for Hardware-in-the-Loop Integration

Abstract

The main goal with the thesis was to design, construct and evaluate a system able to drive and brake a mechanical axis to simulate loads and stored rotational energy - a hardware force simulator. The force simulator was then to be integrated in a existing hardware-in-the-loop testing rig to enhance the mechanical dynamics of door simulations. The force simulator was to be realised using a brushed DC motor and by controlling the motor torque. Controlling the motor torque of a brushed DC motor is achieved by controlling the motor current - and this is mainly what this thesis is about. The thesis explains the foundations of an embedded system capable to control and log the current of a brushed DC motor.

The force simulator consisted of a microcontroller, a motor card, a circuit to measure the inline motor current and a brushed DC motor. The circuit to measure the current was designed and implemented, consisting of a shunt resistor and a current sense amplifier. The shunt was placed in series (inline) with the motor. A software strategy was developed and implemented to deal with noise due to common-mode voltage transients (caused by motor control using PWM). This strategy came with a cost of introducing a measurement delay in the system.

A discrete PI-controller to control the motor current was researched and implemented. An expression to optimal tune the controller based on motor parameters and the sampling period was researched. Current control experiments were conducted but due to a calculation error the PI-controller was very untuned, this compared to the tuned expression. This error was discovered very late in the work and there was no time to redo the experiments. The experiments with the implemented untuned PIcontroller were conducted and a current reference was successfully tracked, however with noise which had an amplitude of approximately 200 mA.

A strategy to record current measurements used in the current control, without missing any samples was successfully developed. It worked by using a double buffer system, filling the buffers utilizing a DMA and then written to a SD card once a buffer was filled. With the main objective to develop the capability to be able to analyze the current control algorithm.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title		ISBN
0280-5316		
Language: English	Number of pages: 1-59	Recipient's notes
Security classification		

http://www.control.lth.se/publications/