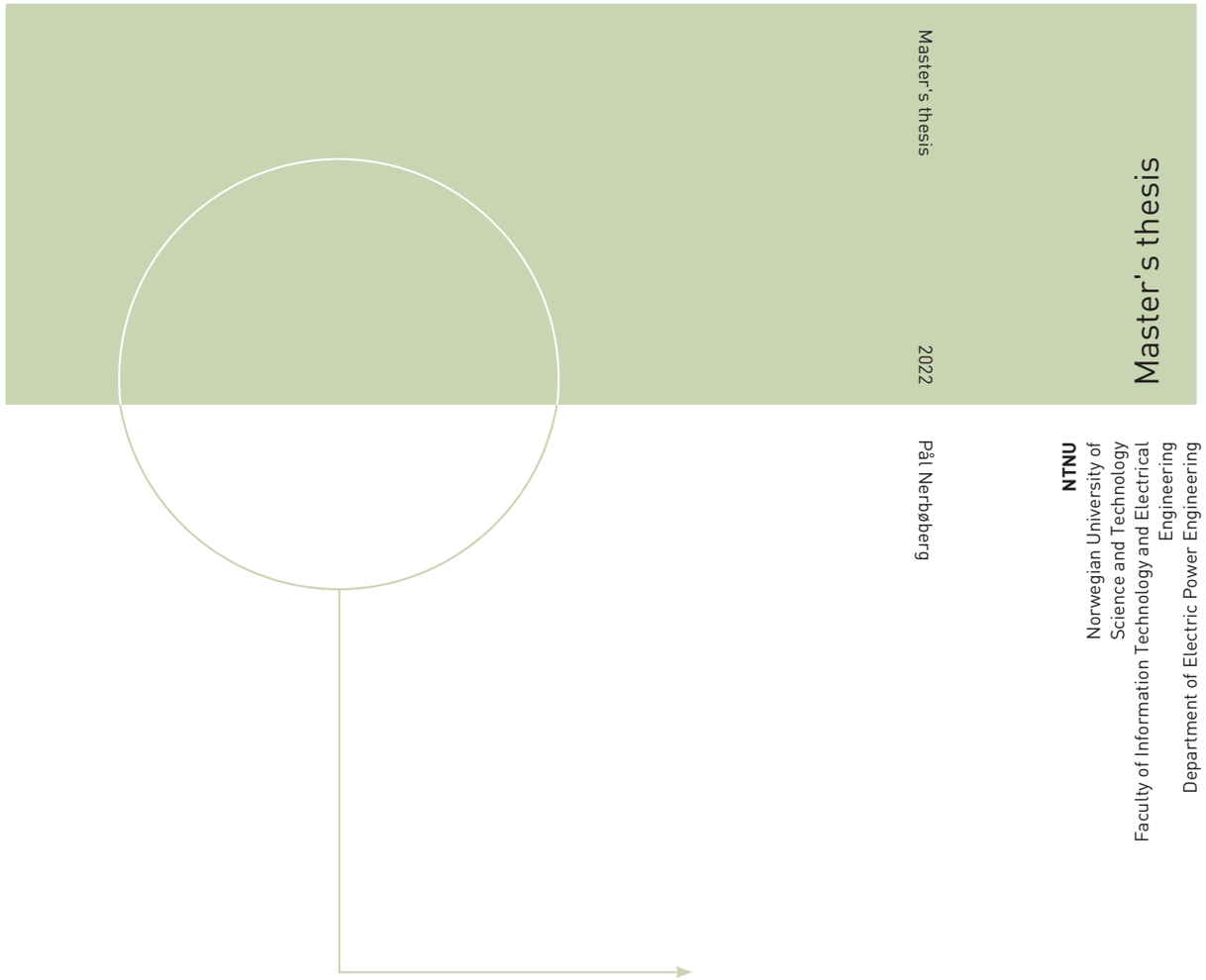


Pål Nerbøberg

Sensorless Control of a PM motor Drive for Drones based on STMicroelectronics STM32 microprocessor series

June 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Sensorless Control of a PM motor Drive for Drones based on STMicroelectronics STM32 microprocessor series

Pål Nerbøberg

Energy and Environmental Engineering
Submission date: June 2022
Supervisor: Roy Nilsen
Co-supervisor: Alexey Matveev

Norwegian University of Science and Technology
Department of Electric Power Engineering

Problem Desscription

The topic of this master-project is proposed by Alva. Alva wants to develop its own series of motor controllers. One of the key tasks is development of motor control strategy and the software. The design requirements include high switching frequencies and high-dynamic operation - fast changes in speed and load.

The Main Target is to demonstrate the capability and customability of sensorless motor control software for slotless permanent magnet synchronous machines (sl-PMSM) by using a commercial control platform and converter layout. Modifications will be needed both in software and hardware.

Some sub-targets are:

- Analyse the existing SW and tools for the actual commercial control board.
- Give a detailed description of the SW as well as the hardware.
- Make the hardware modifications needed on both control board and converter board. This includes choice of filters.
- Analyse and tune the current- and speed controllers for the actual PM-machine. Choose design rules for the controllers.
- Implement relaiable startup routine.
- Test the motor drive in position sensorless control if sufficient time available

Preface

This thesis is the final step to achieve a Master of Technology. The thesis describes the software and hardware of a commercial control platform from STMicroelectronics for controlling drones. It also explores and implement a reliable sensorless startup method in addition to implementing tuning techniques for speed and current control. The thesis is written for Alva Industries in collaboration with the Department of Electrical Power Engineering at Norwegian University of Science and Technology under supervisor Roy Nilsen.

The motivation behind this thesis is for Alva Industries to develop an in-house motor controller, being able to implement and test adjustments to the control algorithms and develop its own power board in the long run. My personal motivation writing this thesis is getting hands on experience with control platforms. Both how the hardware and software is architectured and connecting this to the physics of a motor.

The contribution of this thesis is describing the STMicroelectronics STM32 environment as a first step on the way to develop a control platform around an STM32-microprocessor. It also implement tuning techniques for the speed and current controller in addition to a reliable startup routine.

I would like to thank my supervisor Roy Nilsen for helping me gain a system understanding of a motor controller with his experience and knowledge in his field. I would like to thank Alva Industries for the objective they provided and the summer internship. This have contributed to my increasing interest in hands on engineering. I would also like to thank my friends for a fantastic period in Trondheim at NTNU, and my girlfriend for support, and taking care of our puppy this last intense period.

Abstract

Alva Industries wants to develop a control platform for their slotless permanent magnet synchronous machine (sl-PMSM) for drones. STMicroelectronics STM32 microprocessor series is explored and documented with the help of Nucleo STM32G431RB control board and X-Nucleo IHM08M1 power board as a first step to achieving this. The STM32 environment provides useful tools through its Motor Control Software Development Kit (MCSDK) with the Motor Control Workbench for fast set up of the motor control and Motor Pilot as an interface for steering and logging. The STMCubeMX generates initializationcode for the system clock, timers, pins, etc. At the same time, it offers high flexibility for low-level configuration. These tools generate source code for the STM32 IDE, which can be customized. The STM32-series is a flexible system and provides the tools which simplify further development and customization on this platform.

The current sensing circuit of the powerboard is found not to be optimal. The cut-off frequency is $333kHz$, and the gain stabilizes at $-17dB$ for higher frequencies. This leads to unnecessary noise in the current measurement. The input filter capacitators C_3, C_5 and C_7 are suggested to have a capacitance of $3.5nF$ for a future switching frequency. This choice will set the cut-off frequency to $70kHz$, and the high-frequency gain tends to zero.

Implementing modulus optimum for the current regulator and symmetrical optimum for the speed regulator is done successfully. The results suggest a dampening factor of $\zeta = \frac{1}{\sqrt{64}}$, which results in a rise time of $0.47ms$. The lack of a filter in the speed estimation caused large speed estimation ripples resulting in a current reference output of the speed regulator reacting to the ripple. To avoid the reaction, the symmetrical optimum β-value was increased to 10, and the calculated proportional gain was reduced by a factor of 10. This led to an over-damped speed regulator, which is not optimal. A filter for the speed estimation should be implemented.

I/f startup routine was implemented. The alignment stage of the startup routine resulted in oscillations around the magnetic axes causing the startup to be unreliable. The startup was successful and reliable by adding an external friction force in this stage.

Sammendrag

Alva Industries ønsker å utvikle som egen kontrollplattform for deres tannløse permanent magnet synkronmaskin for droner. STMIcroelectronics STM32 mikroprosessor serie er utforsket og dokumentert med hjelp av Nucleo STM32G431RB kontroll brett and X-Nucleo IHM08M1 kraftelektronikk brett som et første steg for å oppnå dette. STM32 utviklings miljøet inneholder nyttige verktøy gjennom sitt Motor Control Software Development Kit (MCSDK) med Motor Control Workbench for å rask implementering av motor kontroll og Motor Pilot som kan styre motoren og logge data. STMCubeMX genererer initialiseringskode for system klokken, timere med mer. Programmene tilbyr også høy fleksibilitet. Disse verktøyene genererer kildekode til STM32 IDE. Kildekoden kan endres direkte i STM32 IDE. STM32 serien tilbyr fleksibilitet og nyttige verktøy som gjør det enklere å utvikle og tilpasse en kontrollplattform til sitt behov.

Strøm målings kretsen på kraftelektronikkbrettet er ikke optimal. Knekkfrekvensen er 333 kHz og forsterkningen stabiliserer seg på -17 dB for høye frekvenser. Dette fører til unødvendig støy i strømmålingene. Inngangsfilterets kondensatorer C_3, C_5 og C_7, er foreslått endret til $3.5nF$ tilpasset en fremtidig PWM-frekvens på $60kHz$. Dette fører til en knekkfrekvens på $70kHz$ og forsterkningen ved høye frekvenser går mot null.

For farts og strøm regulatorene er henholdsvis symmetrical optimum og modulus optimum metodene implementert. Resultatet er en demningsfaktor på $\frac{1}{\sqrt{64}}$ og fører til at strømmen stiger fra $10 - 90\%$ av referansen på $0.47ms$. Mangelen på et filter i estimeringen av rotasjonshastigheten førte til pulsasjoner på rotasjonshastighet estimatet som utgangen av hastighetsregulatoren prøver å kompensere. For å unngå denne kompensasjonen øktes beta verdien til 10 og proporsjonal forsterkningen ble senket med en faktor på 10. Det førte til en overdempet fartsregulator. Ette er ikke optimalt og et filter for estimeringen av rotasjonshastigheten bør implementeres.

I/f oppstarts metode er implementert. Når rotor retter seg etter det magnetiske feltet førte det til svingninger rundt den magnetiske aksen satt opp av stator. Dette gjorde oppstarten upålitelig. Oppstarten var vellykket og pålitelig etter en ytre friksjonskraft ble påført rotor i denne fasen.

Contents

Chapter 1

Introduction

The use case of drones is increasing every year as the technology improves. The applications range from recreational use, delivery, military applications, and monitoring of crops. In 2020, for the first time in Norway, a drone was used in a search and rescue mission to find a missing woman[20]. The growth is backed by Grand View Research's estimated compound annual growth rate (CAGR) of 57.5% from $2021 - 2028$[12]. Alva Industries wants to take a share of this market.

Alva Industries is producing low inductance slotless permanent magnet synchronous machines for drones. Off-the-shelf controllers do the control of the machines. Alva Industries is exploring the possibility of developing a control platform of its own. This is, among other things, to increase the possibility of testing and implementing improvements in the control algorithm specific for their motor and application.

In 2020 Jensen [6] analyzed how increasing the switching frequency would benefit the current ripple of a low inductance machine for Alva Industries. In 2021 as a precursor to this master project Nerbøberg [9] explored a sensorless closed loop voltage-current model for speed and position estimation that showed high performance and may be beneficial to implement for Alva Industries. To explore these possibilities, a control platform with high customizability is needed to implement and test them. This thesis will explore and test if STMicroelectronics STM32-series can achieve this.

This thesis is divided into two parts. The first part is an analysis of how the STMicroelectronics STM32 system and environment are architectured. Some important components of the hardware of a control platform, The software tools in the STM32 environment, and the control software is described. This provides a good foundation for the further development of an Alva Industries control platform based on the STM32 microprocessor.

The second part contain modeling and field oriented control of a PMSM machine. The tuning techniques modulus optimum and symmetrical optimum is explained and implemented for the current controller and a speed controller respectively. In addition to this a reliable startup routine (I/f) is described and implemented. An analysis of the sensing circuit input filter is done and improvements for later iterations is suggested. The tuning techniques and startup routine is tested on the STM32 control platform described in part one. Using the Nucleo control board and X-Nucleo powerboard from this thesis as inspiration, a custom-built control board and power board can be developed around the STM32 microprocessor, where the end goal is an in-house control platform.

Part I

Research of the hardware, development environment and control software of STMicroelectronics as a control platform for Alva Industries

Chapter 2

Key hardware components

To reach the goal of developing an in-house control platform, it is essential to have an understanding of the hardware components. This goes for their purpose, how they work and how they may influence the control schemes. For this reason, some of the essential hardware components will be described in this chapter.

2.1 Power MOSFET

MOSFET is an acronym for Metal Oxide Semiconductor Field Effect Transistor. A MOSFET is made from a semiconducting material, meaning that the material has conductivity between conductors and insulators. To make a semiconductor a good conductor, impurities are introduced into the pure crystals. This is called doping. If the impurities are pentavalent, having valence of five, the semiconductor is N-type. In N-type semiconductors, the majority of charge carriers are electrons. If the impurities are trivalent, the valence of three, then the semiconductor is p-type. In P-type semiconductors, holes are the majority of charge carriers. Setting these two types of semiconductors together, the electrons from the n-type will fill the holes of the p-type semiconductor in the junction between them. This effect is depleting the charges near the junction and is called the depletion region. Depending on which side that is experiencing a positive or negative charge, the depletion region will shrink or expand. Reducing the depletion layer is called forward bias and is triggered when the connection is of positive charge on the p-type, and of negative charge on the n-type. By flipping the polarity, the depletion layer expands, and this is called reverse bias.[8]

MOSFETs are of two types. Enhancement and depletion type. An enhancement MOSFET is "OFF" when the gate-source voltage (V_{GS}) is zero. Contrary, depletion MOSFETs are "ON" when V_{GS} is zero. Both of these types can conduct as N-channel and P-channel. For N-channel, the source and drain are n-type semiconductors, and the substrate is p-type. The current carrier is electrons and conducts by attracting electrons toward the gate, creating a channel. For P-channel, the drain and source are p-type semiconductors. The current carrier is holes and conducts by attracting positive charged holes toward the gate, creating a channel. In fig. 2.1 a N-channel enhanced MOSFET is illustrated.

The depletion type use case is, for example, load resistors in logic circuits, start-up purposes in auxiliary power supply circuits, and PWM ICs for flyback circuits. The Enhancement type is mostly used for electronic switching circuits, power electronic ICs, motor drive ICs, and digital controllers [4]. The Enhancement type is hence the best solution for the control platform prototype in this thesis.

An N-channel power MOSFET has a big advantage over P-type power MOSFET because of the higher mobility of electrons. Because of the higher mobility, high switching devices are more suitable for the N-channel device. The higher mobility also leads to a lower $R_{DS,on}$ for the same geometry of an equivalent P-channel device [18].

The MOSFET used in this thesis is STMicroelectronics N-channel $60V$ power MOSFET STL220N6F7.

2.2 Gate Driver

Another important component is the gate driver. A gate driver is an amplifier that takes a low power input from a controller IC and produces a high current gate drive for a power device [7]. The use case is when a PWM controller is not able to deliver sufficient output current to drive the gate capacitance high enough or fast enough

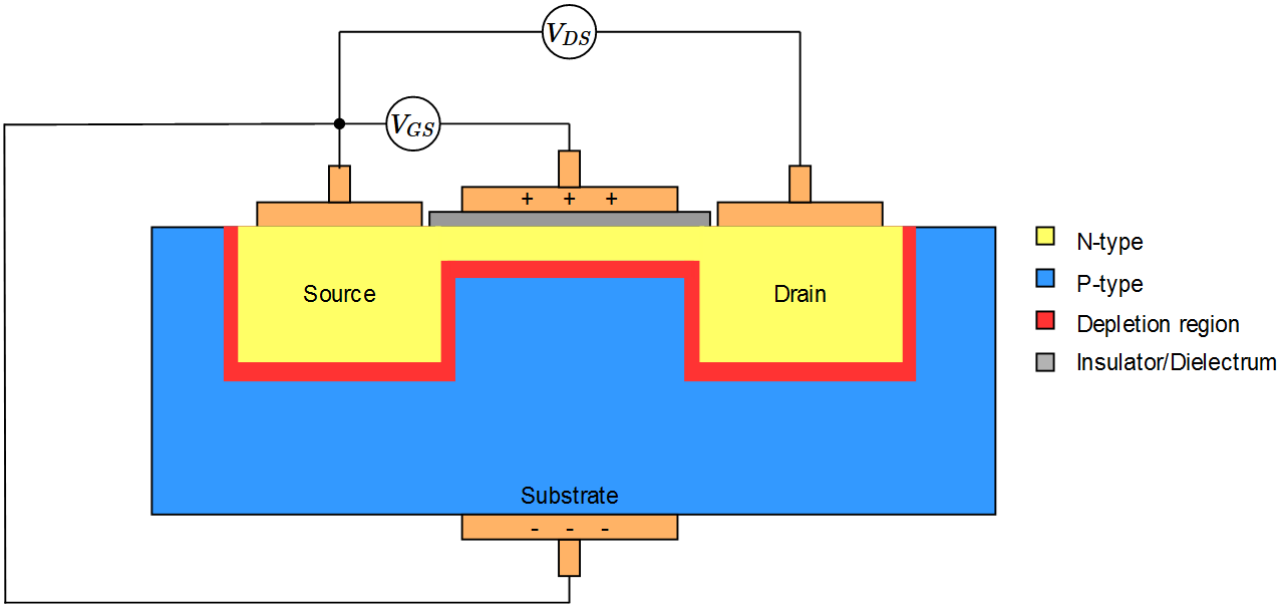


Figure 2.1: The figure illustrates how an N-channel MOSFET. Applying a positive charge at the gate attracts negative charges to the top of the substrate.

to a power converter.

The gate driver is sending gate signals to the MOSFET. Even though it is the timer that generates the PWM signal that is sent to the gate, a gate driver is needed. If this signal is not processed in a circuit, the performance will decrease significantly. This is called direct drive and is only recommended for applications where cost and space-saving is most critical[2].

The gate driver used in the prototype is STMicroelectronics L6398. This is a high-performance gate driver for N-channel power MOSFET.

2.3 Clocks and timers

In general

A timer is an essential part of a microcontroller. It is a hardware component that is built within a processor chip. The timer has a register that is called a counter. The counter counts upwards or downwards to/from its maximum value. An 8-bit timer is able to count from $0 - 255$ and will start over at 0 when it reaches 255. However, an 8-bit timer is able to count to values lower than 255 if the timer is configurable. For example, it can start over after it has counted to 99 and will hence count 100 steps before it starts over. It is important to notice that it is hardware and not software that increments/decrements the value in the counter register[3].

One can decide for what clock pulses the timer counter will count on by using a prescaler[3]. A prescaler can reduce the frequency that the timer counter experience and hence slow down the clock speed. For example, with a system clock frequency of $64MHz$, the timer can operate at $32MHz$ by choosing a prescaler of 2. In the STM32-series, it is both an AHB Prescaler, which is a prescaler from system clock to hardware, and an APB prescaler, which is a prescaler to a given peripheral. The prescaler enables a trade-off between timer resolution and timer range.

The timer resolution is affected by the clock rate since the timer will have more counts per period if the clock rate is high. This will give a "smoother" discrete rise to the value of ARR. However, this also comes with a side effect and will make the counter overflow or underflow faster. By thinking of an 8-bit counter register, there is only room for 255 counts. By having a high resolution, this register will quickly overflow or underflow, depending on the counting method.

The switching frequency is defined by the maximum system clock frequency divided by the prescaler (PSC) for the timer used plus one and the counter period plus one. This is shown in eq. (2.1).

$$f_{sw} = \frac{f_{clk}}{(ARR+1)(PSC+1)} \tag{2.1}$$

Auto reload register (ARR) is a value that can be set by the programmer and will define the counter period. In other words, how many counts it will do before it starts over. This counter period or ARR can be counted in different ways. The counting modes available on the STM32 are as follows.

- Up-counting mode
- Down-counting mode
- center-aligned mode

Up-counting means that to start at a low value and count up to the ARR. When this is reached, it is called overflow, and it will trigger an update event (UEV). This will make the counter register restart and start over from 0 again.

Down-count is the opposite and starts at the ARR value and counts to 0. A counter underflow occurs when the counter reaches zero, and an UEV is triggered. The counter then restarts at the ARR value. Up-count and down-count are both edge-aligned counting modes, often referred to as sawtooth.

In addition to these, a center-aligned mode can be used. For the center-aligned mode, the counter operates in up-counting mode until it reaches counter overflow. An UEV is triggered, but now the counter operates in down-counting mode. When the counter reaches 0 and underflow, an UEV is triggered once again, and it will operate in up-counting mode. This will create a triangular waveform and will also double the period of the signal. Center-aligned PWM signals are often used in motor control because of their symmetrical shape, which leads to fewer harmonics and reduces noise interference and power consumption.

If the timer is used for PWM mode, the counter register value is compared to the value stored in what is called the compare and capture register. The compare and capture register (CRR) contains a threshold that, depending on the PWM-mode, will generate an output that is high if the counter register is above or below the CRR. This can be used on all three counting modes described above. It is this signal that will trigger the MOSFETs to open/close depending on the MOSFET type (enhancement or depletion).

The output of the counter register value and the CRR register depends on the PWM mode. The two modes are described in eq. (2.2) and eq. (2.3). Depending on the PWM mode the duty cycle are described in eq. (2.4) and eq. (2.5) respectively.

$$TimerOutput(High - Truemode) = \begin{cases} Low & Counterregister < CCR \\ High & Counterregister \geq CCR \end{cases} \tag{2.2}$$

$$TimerOutput(Low - Truemode) = \begin{cases} High & Counterregister < CCR \\ Low & Counterregister \geq CCR \end{cases} \tag{2.3}$$

$$D = \frac{CCR}{ARR+1} \tag{2.4}$$

$$D = 1 - \frac{CCR}{ARR+1} \tag{2.5}$$

Dead time

When opening and closing the gates of a power MOSFET, it is essential to ensure that the current flow in the intended loop. If two complementary MOSFETs are triggered simultaneously, there is a high probability that they will be open simultaneously and cause a shoot-through. This is a critical fault and may lead to instant damage to the components and malfunction. To counter this, dead time is introduced. Dead time, also called blanking time, is the time between the closing of one and the opening of the other component [8].

Timer settings for the PWM generation in STM32G431RB

TIM1 is one of two advanced timers on the STM32G431RB platform[14]. It has a 16-bit counter resolution and can count up, down, and up/down (center aligned). The prescaler can be between $1 - 65536$ and has four capture/compare channels. The maximum timer frequency is $170MHz$.

The counter mode for the timer used for the PWM signal (TIM1) is in center-aligned mode. TIM1 is operating in Low-true mode, which is called "PWM mode 1" by STMicroelectronics.

By choosing a counter period (ARR), the switching frequency can be decided. Alva has previously used a switching frequency of $30kHz$ on their low inductance motors through the VESC control platform[1]. Hence this is a good starting point for the prototype control platform. By using eq. (2.1) and the system clock frequency $170MHz$ and no prescaler, the counter period is calculated to be 2833. Since the counter mode used is center-aligned, two UEVs are needed for one PWM period. The result in an ARR of 1431.

The minimum dead time of the MOSFET is 700ns. The software inserted deadtime is set to be 800ns.

2.4 Measurements

In motor control, there are some essential parameters that the control algorithms use as input to produce the wanted output. The position of the rotor is essential to induce a magnetic field in the stator coils. The correct direction, amplitude, and timing are needed to spin the rotor efficiently. The current induces the magnetic field, so this is an essential parameter for the same reasons.

In addition, it is crucial that components do not fail and are run within their limitations. These limitations are, for example, temperature ratings and voltage ratings.

In this control platform these parameters are measured by hardware:

- Motor phase currents (i_a, i_b, i_c)
- DC buss voltage (V_{bus})
- Temperature powerboard

In addition to these measurements, the motor speed and position are estimated through STMicroelectronics sensorless measurement scheme based on a back-emf Luenberger observer. Sensorless measurements were the subject of the work preliminary to this thesis [9] and will not be further discussed in this thesis.

Current sensing

In order to control a motor, current is sent through the stator coil to synchronize with the rotor magnetic field. The stator phase currents need to be efficiently and optimally monitored for optimal control. For this purpose, a sensing circuit is implemented.

A popular way to measure current is through a current sensing resistor. This can be understood as a current to voltage converter by thinking of the current as a linear converter from Ohms law $V = IR$. This voltage can then be processed and be a reading of the current to be measured. The advantages of using current sensing resistor circuits are that they have low costs, high measurement accuracy, measurment range from very low to medium and can measure both DC and AC currents. The disadvantages are that it introduces a resistor into the circuit and will result in power losses. In addition, it will affect the overall load of the system [21]. However, this method is generally used in low to medium current sensing applications. For High current applications, another method may be desirable.

Several sensing typologies can be used. Instrument [5] and Zhen [21] proposes some techniques for current sensing and are discussed here. The simplest way is low-side global current sensing. This topology is cheap and easy to implement because it only measures one phase. In addition, the common-mode voltage is close to ground since it measures the current on the low side of the transistors. This allows a wide range of low voltage op-amps to be used in the sensing circuit. However, this is not sufficient in the high-performance application of flying a drone with Field oriented control (FOC) since all phase currents must be known. This leads us to the next method.

Low-side current sensing in each leg is commonly called low-side 3-shunt current sensing. This is similar to the global current sensing. However, instead of measuring in a common node for all the lower three legs, the same circuit is implemented in each low side leg. This will give the individual phase current of all three phases. This method is commonly used in FOC applications and has many of the same benefits as the global current sensor topology. It is sufficient to measure only two phase-currents and let the software calculate the third from the other two. However, these methods cannot detect shortcuts to ground and are sensitive to ground variations.

One can use a high-side sensing topology to counter the problem of not detecting shortcuts to ground, and being vulnerable to ground variations. This is because the topology places the current sensing close to the source voltage. However, the op-amp must operate at high common-mode voltages when placing the circuit on the high side. This topology gives stable measurements but does not give an exact match of the motor currents. This method can also use a three-shunt topology as the low side topology.

The most accurate method is in-line phase current sensing. This is because it is measured in the motor cables and is the best alternative for optimized performance. It uses dedicated sensing, which handles significant and fast variations in the common-mode voltage. The shunt resistance is located with the PWM driver. Because of the difficult measurement environment, this sets higher requirements for the current sensor.

The current measurements in the IHM08M1 are done by low-side current sensing. This sensing circuit will be analyzed later in the thesis.

The current measurement is not only used for motor control but also as hardware protection. An over-current protection circuit is implemented in the hardware with a detection circuit. An embedded current reference calculated from the MCU is compared to the measured current. The output of the comparison generates a fault condition and disables the driving signals if triggered.

Chapter 3

STMicroelectronics support tools and environment

STMicroelectronics has created an environment of support tools to easier get off the ground using their hardware. This environment can set up firmware for the specific hardware and provide software for motor control. This is done through the Cube MX and the MC workbench software tools. Both tools enable fast initialization through known hardware but also highly customizable settings for custom hardware. How the environment operates is shown in fig. 3.1.

3.1 From hardware to controller

The path to a functional motor controller through the STMicroelectronics support tools can be summarized in five steps.

- Set up hardware
- Use MC workbench to configure firmware/software and the FOC library
- Customize MCU firmware through STM32 CubeMX and generate source code to IDE
- Flash the code to the MCU

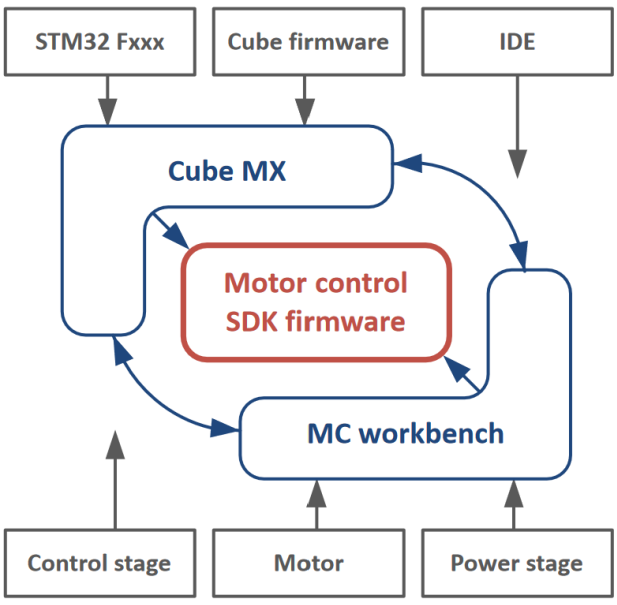


Figure 3.1: Shows the structure of the STM32 environment and how the different tools is cooperating. [16]

- Control the motor through motor pilot GUI

3.2 Set up hardware

The hardware is chosen to match the application's intended use. The hardware is divided into three components: the control stage, the power stage, and the motor. These can be STMicroelectronics components, or they can be custom hardware. By choosing compatible STMicroelectronics components, the software will implement all known parameters and ease the engineer's workload. This accounts for both the control board and the power board.

The control board in this prototype is the Nucleo-STM32G431RB and is compatible with the MC Workbench and CubeMX. The support tools will hence know which timers, pins, clock frequency, etc., the MCU has and adapt the settings to the possibilities/limitations of the control board.

The same goes for the power board. In this prototype, the power board is the IHM08M1 from STMicroelectronics and is also compatible with both software tools. The software automatically knows the resistor values of the sensing circuits and which MOSFETs are used.

An additional tool for the motor can be used to adapt the software to the motor to be used. This tool is Motor profiler. The motor profiler tool automatically measures the electric parameters of a PMSM/BLDC through an algorithm that enables running unknown motors after a short period of time. The inputs to the motor profiler is:

- Pole pairs
- Max speed
- Max peak current
- Bus voltage (optional)

The motorprofiler will than calculate the parameters needed for the motor control FOC control algorithms which is listed below:

- Stator resistance
- Stator inductance
- B-EMF constant
- Inertia
- friction

The motor profiler measures this through the control platform and is compatible with the Nucleo-STM32G431RB control board and the IHM08M1 power board. After the profile is created, it can be saved and used by the Motor control Workbench in the same way as the control board and the power board. This method struggles with low-inductance motors, and the profile was not able to be completed for the motor in this task and was set manually.

3.3 Motor Control Work bench

The motor control workbench (MC WB) is the tool that helps implement the FOC firmware library to the chosen hardware of the control platform. It enables the designer to change and adapt the default settings to the application and its intended controller. The STM32 MC WB is an additional layer to the STM32 CubeMX. Where the STM32 CubeMX is a tool for Hardware initialization of the MCU, the MC WB is a specific tool for the implementation of motor control to the MCU. This reduces the time to configure the STM32 PMSM FOC firmware. There are four groups of parameters that should be altered to fit the intended application. These are:

- Motor parameters
- Power stage parameters

- Drive managment parameters
- Control stage parameters

However, using components compatible with the MC WB, most parameters are set automatically. A snapshot of the GUI of the motor control workbench is shown in fig. 3.2.

The motor parameters contain parameters for safe operation of the motor and motor parameters for the FOC control scheme set. If the motor profiler has been used on the motor for the application, the profile can be directly uploaded here. The power stage parameters contain components like the MOSFET and sensing circuits. If the power board chosen is compatible with MC WB these hardware settings do not need to be altered. However, Alvas goal is to end up with its own control platform and its own power stage design. This software enables the implementation of a custom-built power stage. The drive management parameters decide the execution rate of the speed and torque regulators, regulator gains, and PWM frequency. The last group is the control stage parameters which maps the stm32 peripherals and can alter timer settings and pin assignments.

The motor control workbench gives the designer a template that fits the hardware chosen and enables easy and fast alterations so that the controller can fit the application.

When this is configured to the designer's intention, a project can be generated to match the configuration. The code is generated in the programming language C. The code is generated with the help of STM32CubeMX(v6.3.0), the HAL-Libery (Hardware extraction layer), and a firmware package compatible with the chosen hardware (STM32 FW V1.16.1). In this project Motor Control Workbench v5.Y.4 is used. The result is an ".ioc"-file, which can be modified in CubeMX and the motor control library in C-code.

3.4 STM32CubeMX

Where the MC WB is specific for motor control, the STM32CubeMX is a GUI that offers an easy way to configure an STM32 microcontroller or microprocessor and generate corresponding initialization C code for the MCU. It is compatible with Arm Cortex-M and partial Linux Device Tree for Arm Cortex-A. This is done by choosing a microcontroller/microprocessor or a development platform. After this, the GUI will show the chosen device, and the customization possibilities are limited to the specific device. A snapshot of the GUI is shown in fig. 3.3. This makes it easy to configure pins, timers and clock settings. When the settings are satisfying, the CubeMX generates C code that initializes the chosen setup. This code can be altered and built upon in an IDE. There are several compatible IDEs. One of them is STMicroelectronics own IDE STM32CubeIDE, and is the IDE used in this project.

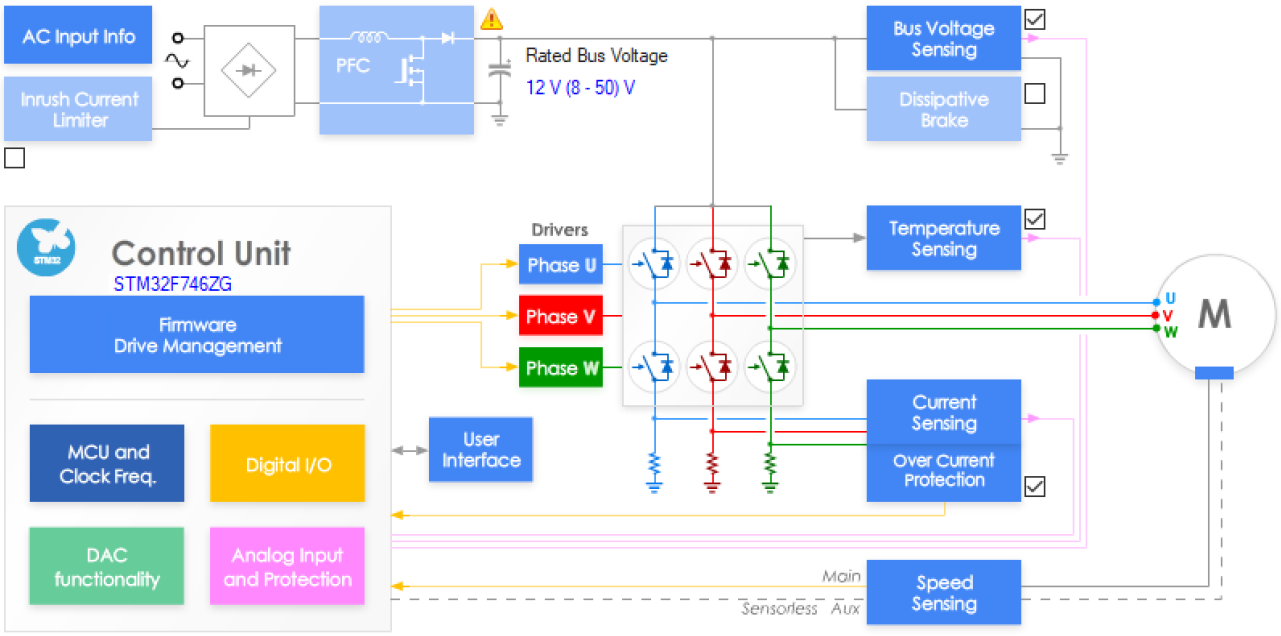


Figure 3.2: Shows a snapdhot of the GUI of the ST motor control workbench

When coming from the MC WB the generated ".ioc" file is opened in STM32CubeMX. MCU hardware settings from the motor control workbench are automatically set, such as PWM generation of the timer used, the Auto reload register to match the frequency set in the drive management parameters in MC WB, etc. Additional measurements, outputs and timers can be set to build upon the already chosen settings.

When all configurations are set, source code can be generated from the STM32CubeMX. This code, combined with the MC WB code, is sent to the chosen IDE, which in this case is the STM32CubeIDE.

3.5 ST STM32CubeIDE

The ST STM32CubeIDE is a development platform that compiles code and provides debug features for STM32 MCUs. It is also integrated with the CubeMX, and the user can change configurations of the peripherals or middleware and regenerate the code without compromising code added by the user. The code generated is transparent and consists of thousands of lines which significantly reduces the time needed to develop a functional motor control drive. The software can be directly adjusted and changed from this IDE and then flashed to the MCU. This way, Alva has complete flexibility to alter the software to test if there can be any improvements.

3.6 Motor pilot

When the firmware and application software is generated and flashed to the MCU, the motor can be controlled through the MC WB add-on "Motor Pilot". This interface can give commands, monitor, and log motor control variables. It also provides fault words like "over-current" if it occurs.

The motor pilot can plot many of the parameters of the motor control at a sampletime of $0.2s$. It also can sample some critical parameters at the PWM frequency. Those parameters are shown in table A.8.

3.7 Some essential settings in the MC WB

This section describes some of the important settings in MC WB.

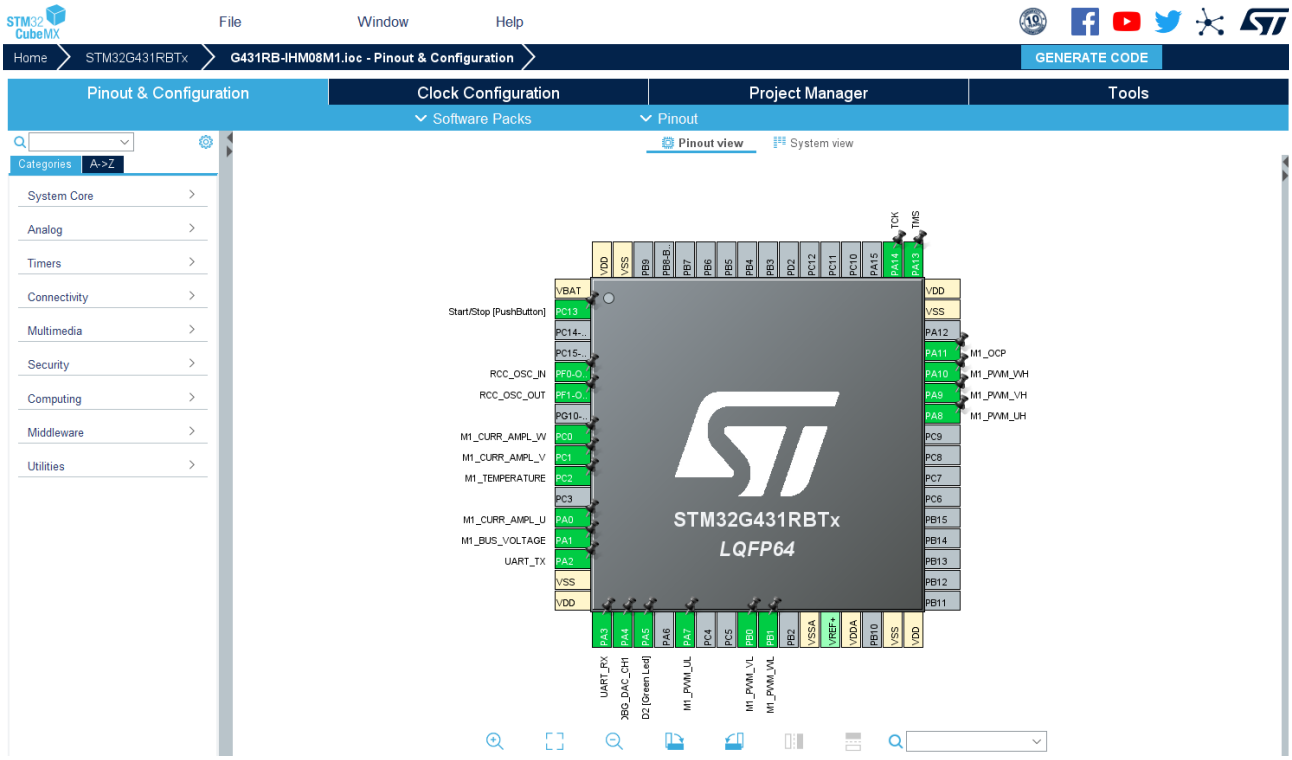


Figure 3.3: Shows a snapshot of the CubeMX GUI.

Drive settings

The drive settings are where the parameters necessary for the drive are set. This is PWM, current reading, control mode, regulator execution rate and regulator gains.

The PWM frequency is set to 30kHz. In an earlier study done for Alva, the optimal switching frequency in regards to current ripple of a low inductance slotless machine model was $60kHz$ for a conventional MOSFET converter [6]. A more powerful control board was intended in this thesis to be able to test this frequency. However it was not possible to obtain. In this thesis, the focus is a functional control platform. As a starting point, the frequency is set to $30kHz$, which is the same frequency that Alvas motor has been run on earlier on the VESC platform, as mentioned earlier.

The minimum dead time set by MC WB is $700ns$. The software-induced deadtime is set to $800ns$ to increase the margin. $800ns$ corresponds to 2.7% of the PWM period and will not significantly impact the testing done in this thesis.

regulator settings

The control mode is set to speed control, and the speed regulator execution rate is set to $1ms$ corresponding to a frequency of $1kHz$. This execution rate is set through the systick timer, which is a dedicated timer for real-time operations on the STM32G431RB control board. The execution rate of the torque/current regulator is set to 1 PWM period.

The proportional and integral gain of the torque and flux control loop is also set here. In the MC Workbench, it is K_p and K_i that are the regulator's parameters. A numerator and a denominator give the values. The denominator for the proportional gain is called $K_{p,div}$, and for the integral gain it is $K_{i,div}$. This is done to obtain fractional values. The denominator can be a value between $1 - 16384$ which is $2^0 - 2^{14}$. In addition to this, the integral gains are given in per PWM period. The integral gain must therefore be multiplied by the PWM period $T_{PWM} = \frac{1}{f_{sw}}$. This result in the relation listed in eq. (3.1).

$$K_p = \frac{K_{p,sw}}{K_{p,div}} \tag{3.1a}$$

$$K_i = \frac{K_{i,sw}}{K_{i,div}} T_{PWM} \tag{3.1b}$$

Since the motor is controlled as an SPMSM and flux weakening is not used, the flux/I_d current controller is not implemented. The gains will be calculated in a later chapter.

Chapter 4

Control algorithms

The analysis is done by reading STMicroelectronics motor control software[15] and its user manual [16].

A simplified sketch of the control algorithm architecture is shown in fig. 4.1. The control software can be divided into three separate lines. The first line starts with user commands. This can be: start, stop, speed reference, etc. A user command instantly triggers functions that handle and store the information, but this line does not execute. Then there is the medium-frequency line. This line takes care of the speed control loop and executes speed and torque ramp calculations. This is done by running a loop checking for new commands at this "medium" frequency defined by the speed regulator execution rate. The third line is high-frequency tasks and runs at the ADC1/ADC2 interrupt request, corresponding to the PWM frequency. This line handles the current control loop and the speed and angle estimation. This loop is run faster than the medium frequency loop because of the importance of the inner control loop, which is the current control loop. This structure benefits the redundancy and speed of the code. The most frequency crucial parts of the control scheme (speed and angle estimation and current control) are run more often than those with a lower priority. This structure will be described in more detail below.

4.1 The state of the machine and command state

It is needed to describe the concept state of the machine to understand the difference between executing and setting commands. The control scheme uses the state of the machine to define what tasks that can be applied. The state of the machine is stored in a struct called "STM_Handle_t" which is presented in table A.4. It is the object "State_t" in "STM_Handle_t" which contains the actual state and is listed in table A.3. By defining and using these states in the motor control scheme, the code will increase its efficiency and redundancy by limiting its possible actions in different scenarios. The states can be either persistent or pass-through. A persistent state needs action to move to the next state, but a pass-through state moves into the next state after finishing the current state's algorithm. A flowchart of the state of the machine is shown in fig. 4.2.

Another important state in the software is the state of the commands, which is listed in table A.2. This state

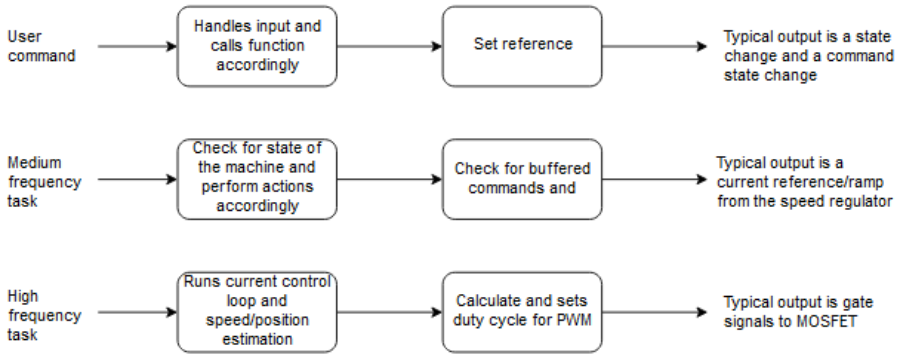


Figure 4.1: Brief simplified schematic of how the control algorithm functions from user input to gate signal of the MOSFET.

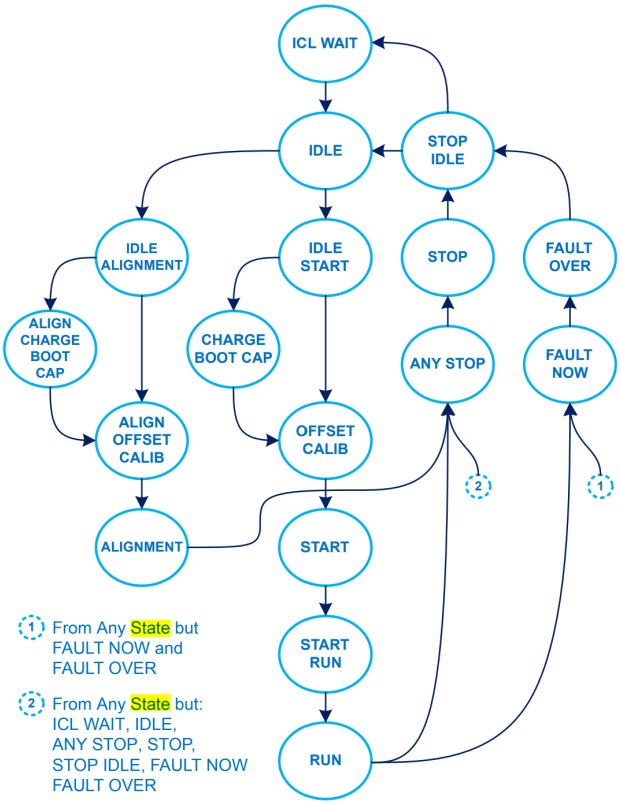


Figure 4.2: Flowchart of the state of the machine [16]

describes if there are pending commands and if the command has been executed successfully or not. If there are pending commands, this will trigger the control algorithm to execute this command. An example is a speed ramp command. However, to perform this command, the state of the machine needs to be in a state that allows this.

The combination of the two types of states described above forms the structure for how the control algorithm behaves. The structure is based on changing states and informing about commands. The commands and actions are executed by regularly checking what state the machine is in and if there are any pending commands.

4.2 Boot/start/init

When the controller is started, it initializes the whole system. This includes, among other things setting and locking the pins on the control board, but it is also initializing the control algorithm. The initialization of the control algorithms is essential such that the variables of the different components are set to match the motor and the hardware. It is also essential to make the variables reachable from different parts of the control algorithm. This is done by the function MCboot(...).

The MCboot function initializes the motor control and has two inputs and no output. The inputs are two arrays, "pMCIList" and "pMCTList". These objects are where the function retrieves and returns information. In this thesis, the number of motors controlled by the software is one; hence the arrays contain only one object. In future iterations, Alva aims to control two motors from the same controller in a coaxial setup. This hardware and software allow that.

The first parameter is the object pMCIList and is a pointer to an MCI handle of type MCI_Handle_t. MCI stands for Motor Control Interface. The MCI_Handle_t object is used to set and hold "high-level" values and states for the control scheme. The data it holds is listed in table A.5.

The other parameter is an object called $pMCTList$ and is a pointer to the MCT handle of type MCT_Handle_t. MCT stands for Motor Control Tuning. This object contains more "low level" values needed for the motor control scheme and is listed in table A.6.

The boot function can be summarized as follows:

- initialize the state machine by setting the State_t to IDLE and the faults to 0.
- Initialize a current limitation object with parameters from the mc_config
- PWM and current sensing component initialization
- Start timers synchronously
- PID component initialization: speed regulation
- Main speed sensor component initialization
- Main encoder alignment component initialization
- Speed and torque component initialization
- Virtual speed sensor component initialization
- PID component initialization: current regulation
- Bus voltage sensor component initialization
- Power measurement component initialization
- Temperature measurement component initialization
- The input objects are set to the private values defined at the start of the script

This initialization list is one example of the workload the software development environment reduces for the developer.

4.3 User commands

After the boot and initialization of the control scheme are performed, a user command is needed to perform an action. The interface written in MC_api.c is what enables the user to apply simple inputs/commands for the operation of the motor. These can be: start, stop, speed ramps, and torque ramps. In addition, information about the state can be retrieved. The function calls are done through the software tool motor pilot in this thesis. However, these function calls can be implemented in a remote controller to suit Alva's needs for future iterations.

The software architecture is set such that these function calls do not execute the wanted action but can call functions that set the goal of the commands and sets/change the states described in the enums "State_t", "MCI_CommandState_t" and "MCI_UserCommands_t" described in tables A.1 to A.3 respectively. These states decide which actions are performed and are described in more detail below.

4.4 Medium frequency tasks

The medium frequency tasks can be described as the driver of the control scheme. Very simplified, the medium frequency tasks check for speed/torque reference, run the speed regulator, and set the reference for the current controller, which is a high-frequency task. In addition, it checks for faults and handles them. This is done through an important function of the mc_task.c script called "MC_RunMotor_ControlTask()". This function is run at each systick interrupt request. The systick interrupt is set to correspond with the speed regulator execution rate. The function contains three function calls which are run if the system is booted correctly.

- MC_Scheduler() - Motor control scheduler
- TSK_SafetyTask() - Executes safety checks
- UI_Scheduler() - User interface scheduler

The motor control scheduler keeps track of a task counter and calls the function "TSK_MediumFrequencyTaskM1()". This function is the backbone of the control algorithm and is based on a switch statement that checks for which state the machine is in and runs code accordingly.

After the motor control scheduler, "TSK_safetyTask()" is called. It is run right after the MC_Scheduler() such that it can handle eventual faults that occur after actions performed in "TSK_MediumFrequencyTaskM1()". The function checks for over-voltage and high temperature and turns off the PWM generation if the fault condition is triggered.

Now the UI_Scheduler is ran and keeps track of three counters:

- bUITaskCounter
- bCOMTimeoutCounter
- bCOMATRTimeCounter

4.5 High frequency task

The high-frequency task line runs the tasks that need precise timing. This is mainly the FOC current control loop control algorithm, but it also has some fault handling and estimates the speed and position from the sensorless algorithm. This is done through the function "TSK_HighFrequencyTask()" and is run at the ADC interrupt request, corresponding to the PWM frequency. In addition, it calculates the motor's position through the back-emf luenberger observer. Fault handling is also done if the duration of the current controller calculations took longer than the time before the next PWM signal occurred.

The output of the current control loop is reference voltages in α, β-frame. This reference is input to the function PWMC_SetPhaseVoltage(...). This function calculates the phase voltages corresponding to the α, β voltages. This is then sent to a function that generates gate signals to the MOSFET from the reference voltages. These voltages are applied through a six-sector PWM generation control scheme. This is illustrated in fig. 4.3.

Assuming that only one of the transistors in a two-level bridge leg can be open at any given time (which is how we want to operate), a schematic of a two-level inverter can be simplified to fig. 4.4. Now the switches have two possible positions. Either current runs through the upper bridge leg into the motor or current runs through the lower bridge leg out from the motor. This can be described in binary terms as 1 and 0, respectively. Applying this logic to all the three inverter legs the six vectors $v_1 - v_6$ in fig. 4.3 can be applied as shown in table table 4.1. In addition, two zero vectors can be applied (111 and 000)[11].

The accuracy of six steps is not sufficient for precise control. This problem can be solved and is illustrated by the sectors in fig. 4.3. The two basis vectors encapsulating a given vector can be combined to give any point inside the sector. For example, a voltage vector with maximum amplitude at 30° can be applied by applying $v_1(001)$ at 50% of the time and $v_2(010)$ 50% of the time.

4.6 PID controller

Now that the structure of the control scheme is described, the controllers can be described. All the control loops are based on a struct called "PID_Handle_t". This struct is listed in table A.7 in appendix. By creating an

Table 4.1: Shows how the different basis vectors can be created by how the inverter is operated.

Vector	Code
v_0	000
v_1	001
v_2	011
v_3	010
v_4	110
v_5	100
v_6	101
v_7	111

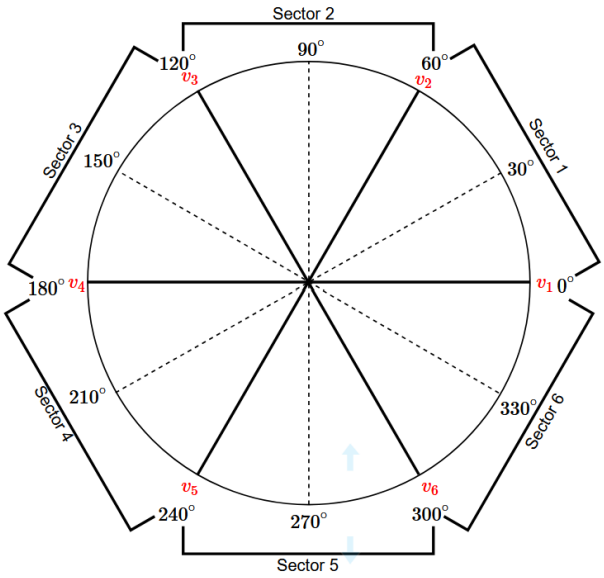


Figure 4.3: Shows the space vector that can be generated by a two-level inverter.

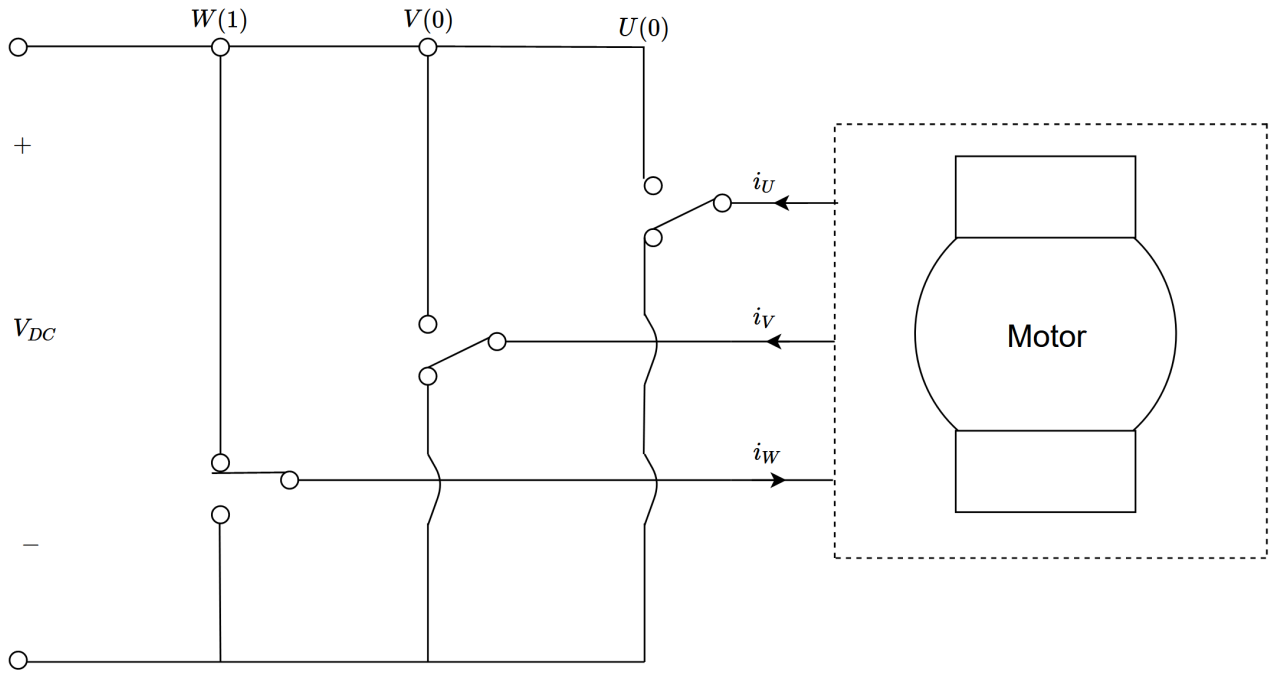


Figure 4.4: Shows a simplified inverter model to illustrate how different space vecors is created based on which bridge leg is connected to the upper or lower level. This example correspond to V_5 (100).

object of type "PID_Handle_t" all necessary information to run a proportional control loop, PI control loop, and PID control loop is in that object. This way, a unique "PID_Handle_t" object targeted for the current control loop and one object for the speed control loop object can be created.

By running the function "PI_controller(...)" with a "PID_Handle_t" and an error as input, an output is retrieved. This is shown in fig. 4.5.

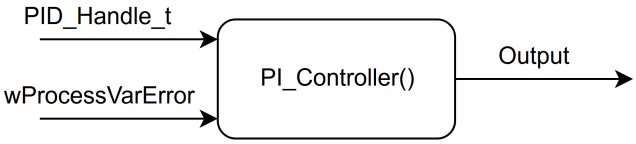


Figure 4.5: A schematic of how the digital PI controller is run in the control software.

Part II

Implementation of tuning techniques, startup routine, analysis and testing on the STMicroelectronics controlplattform

Chapter 5

Motor modeling and control

This chapter describes the modeling of a motor and field oriented control (FOC). In addition, control theory for startup, current control, and speed control is laid out.

5.1 Modeling of a Permanent magnet machine

The modeling of a machine is done the same way now as in the preceding work by the same author[9], and this section is reused here.

An essential concept in the control of AC-machine drives is space vectors. Space vectors enable the description of the currents, voltages, and flux linkages as an amplitude and an angle in a reference frame of the stator phases.[10]. The result is that a phase of the machine can be described as in eq. (5.1).

$$\underline{I}_a = I_a \underline{a}^s \tag{5.1a}$$
$$\underline{\Psi}_a = L_a \underline{I}_a \tag{5.1b}$$
$$\underline{U}_a = R_a \underline{I}_a + \frac{d\underline{\Psi}_a}{dt} \tag{5.1c}$$

By setting a sinusoidal current in the stator windings for a 3-phase machine, the current space vectors can be drawn as in fig. 5.1. By summing the three-phase currents a resultant stator current $\underline{I}_s^s$ can be expressed as in eq. (5.2). This is also transferable to the voltage and the flux linkage, and the stator referred space vectors can be expressed as in eq. (5.3).

$$\underline{I}_s^s = I_a \underline{a}^s + I_b \underline{b}^s + I_c \underline{c}^s \tag{5.2}$$

$$\underline{I}_s^s = \begin{bmatrix} I_a & I_b & I_c \end{bmatrix}^T \tag{5.3a}$$
$$\underline{U}_s^s = \begin{bmatrix} U_a & U_b & U_c \end{bmatrix}^T \tag{5.3b}$$
$$\underline{\Psi}_s^s = \begin{bmatrix} \Psi_a & \Psi_b & \Psi_c \end{bmatrix}^T \tag{5.3c}$$

For a permanent magnet synchronous machine $\underline{\Psi}_s^s$ is defined as in eq. (5.4), where $\underline{\Psi}_m^s$ is the permanent magnet flux linkage and is described as in eq. (5.5) where θ is the rotor position. L_s is the inductance matrix and depends on the machine's position and the machine itself. The inductance matrix is thoroughly described by Nilsen [10].

$$\underline{\Psi}_s^s = L_s^s \underline{I}_s^s + \underline{\psi}_m^s \tag{5.4}$$

$$\underline{\Psi}_m^s = \Psi_m \begin{bmatrix} \cos(\theta) \\ \cos\left(\theta - \frac{2\pi}{3}\right) \\ \cos\left(\theta + \frac{2\pi}{3}\right) \end{bmatrix} \tag{5.5}$$

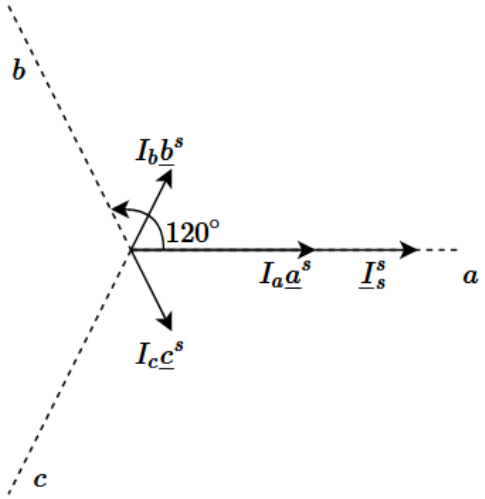


Figure 5.1: The figure shows the currents of a three phase machine expressed in space vectors when the current in phase a is at its peak.

The motor model can now be described as in eq. (5.6).

$$\underline{U}_s^s = R_s\underline{I}_s^s + \frac{d\underline{\Psi}_s^s}{dt} \tag{5.6}$$

This way of expressing the motor model makes it easy to transform it into both the $\alpha\beta$-frame and the dq-frame.

The $\alpha\beta$-frame is a two-phase reference frame where the α-axis is aligned with $\underline{a}^s$ and the β-axis is set 90° ahead of the α-axis. It is a stator-oriented reference frame. The three-phase reference frame can be transformed to the $\alpha\beta$ reference frame by using the transformation matrix in eq. (5.7).

$$\boldsymbol{T}_s^s = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \tag{5.7}$$

The dq reference frame is a two-phase reference frame where the d-axis is aligned with the rotor magnetic axis. This means that it is a rotor-oriented rotating reference frame. The q-axis is 90° ahead of the d-axis. The transformation matrix to this reference frame is seen in eq. (5.8).

$$\boldsymbol{T}_s^r = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) \end{bmatrix} \tag{5.8}$$

The reference frames are drawn in comparison to each other in fig. 5.2.

5.2 Field Oriented Control

This section is reused with minor changes from the preceding work by the same author[9].

There are several ways to control the torque and flux of electrical drives. The most common are field-oriented control (FOC), and direct torque control (DTC). Furthermore, with the increased computational capabilities of microprocessors, model predictive control (MPC) is also starting to gain interest [11]. In this study, a FOC scheme for a PMSM is used.

FOC is a scheme that transforms the stator currents to a stationary two-phase current vector. The two axes in this reference frame are d and q, described in the previous section. By controlling I_d and I_q, the machine's flux and torque can be controlled.

Two common methods for calculation of the reference currents $I_{d,ref}$ and $I_{q,ref}$ are UPF and MTPA [10]. UPF Control stands for unity power factor and calculates reference currents to obtain unity power factor of $+/-1$. Unity power factor is achieved by aligning the voltage and the current vector. However, the control strategy in

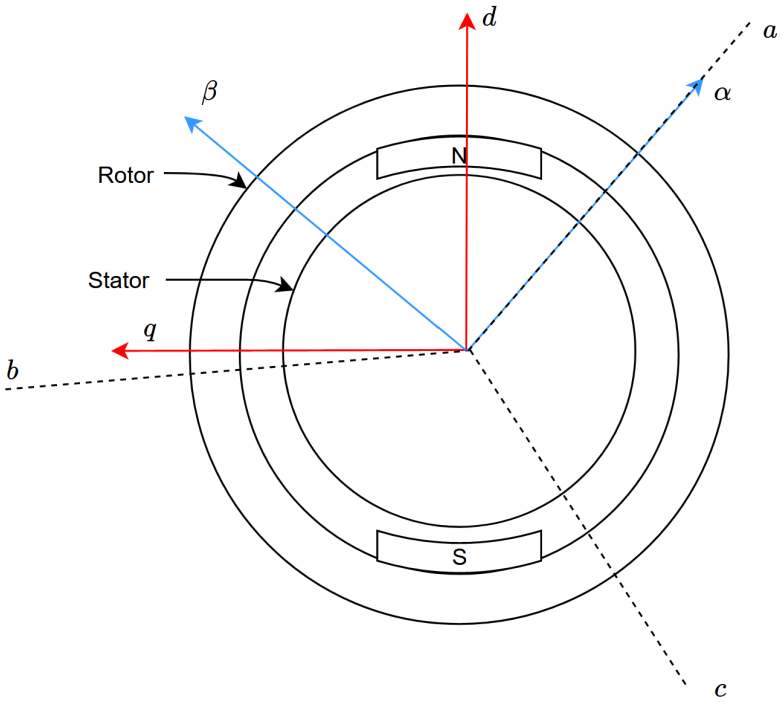


Figure 5.2: The figure shows how the reference frames abc, $\alpha\beta$, and dq are placed in comparison to each other for an SPMSM outrunner.

this study is maximum torque per ampere control(MTPA). The idea is to maximize the torque output from the stator current. It is known that the stator current copper losses are proportional to I_s^2. This strategy will therefore minimize the ohmic losses in the machine. MTPA is a good strategy if copper losses are a significant part of the losses in the machine. The reference is dependent on what machine is to be controlled. Assuming the machine does not operate in field weakening, the reference currents are calculated as in eq. (5.9) for an SPMSM. Here $T_{e,ref}$ is the reference electrical torque set by for example, a torque controller or a manual input.

$$I_{d,ref} = 0 \tag{5.9a}$$

$$I_{q,ref} = \frac{T_{e,ref}}{\frac{3}{2}p\Psi_m} \tag{5.9b}$$

5.3 PI regulator

A PI regulator is a component in the control circuit that tries to ensure that the system's output value is behaving as intended. This is done through a proportional term and an integral term. The proportional term is just a multiplication of the error. The integral term is time-dependent and removes stationary discrepancies. The PI controller transfer function is typically described as in eq. (5.10).

$$h_{PI}(s) = K_p + \frac{K_i}{s} \tag{5.10}$$

By using the integral time T_i by describing K_i as in eq. (5.11), the equation can be rewritten to eq. (5.12).

$$K_i = \frac{K_p}{T_i} \tag{5.11}$$

$$h_{PI}(s) = K_p \frac{1 + T_i s}{T_i s} \tag{5.12}$$

This is convenient since the integral time can be linked to other time constants of the system that is controlled. By choosing the gain and the integral time, the designer can choose for which frequencies the controller shall be effective and what the gain of the error shall be.

5.4 Current Control using modulus optimum

This section contains the control theory and implementation of the tuning technique modulus optimum described by Nilsen [10].

When the open-loop transfer function of a system with a PI regulator can be described as in eq. (5.13) a control technique called modulus optimum[10] can be implemented. This technique is based upon canceling the largest time constant in the open-loop transfer function by choosing the integral time in the regulator to be the same as the dominant time constant T_1. The other time constants can be summed up to one time constant T_{sum}. K_s is a multiplication of all the constants in the transfer function.

$$\mathrm{h_0(\ s) = K_p \frac{1 + T_i s}{T_i s} K_s \frac{1}{1 + T_{sum} s} \frac{1}{1 + T_1 s}} \tag{5.13}$$

When T_i is chosen to be T_1, the dominant time constant is canceled with the numerator of the PI regulator term. The open-loop transfer function can now be written as eq. (5.14).

$$h_0(s) = \frac{K_p K_s}{T_1} \frac{1}{1 + T_{sum} s} \tag{5.14}$$

by using the identity in eq. (5.15) the closed loop transfer function can be described as eq. (5.16)[10].

$$M(s) = \frac{h_0(s)}{1 + h_0(s)} \tag{5.15}$$

$$\mathrm{M(s) = \frac{1}{1 + \frac{T_1}{K_p K_s} s + \frac{T_1 T_{sum}}{K_p K_s} s^2} = \frac{1}{1 + 2\zeta \frac{s}{\omega_0} + \left(\frac{s}{\omega_0}\right)^2}} \tag{5.16}$$

Comparing the second-order transfer function expressed with crossover frequency ω_0 and dampening coefficient ζ with the closed-loop transfer function of the modulus optimum, the identities in eq. (5.17) is found. By choosing a dampening factor ζ, the only unknown in the equation is the proportional gain, and by rearranging the equation, the proportional gain can be calculated. This results in both the proportional gain and integral time being calculated as in eq. (5.18).

$$\omega_0 = \sqrt{\frac{K_p K_s}{T_1 T_{sum}}} \tag{5.17a}$$

$$\zeta = \frac{1}{2} \sqrt{\frac{\mathrm{T_1}}{\mathrm{K_p K_s T_{sum}}}} \tag{5.17b}$$

$$K_p = \frac{T_1}{4\zeta^2 K_s T_{sum}} \tag{5.18a}$$

$$T_i = T_1 \tag{5.18b}$$

If the controller is digital, the modulus optimum must be discretized. By using discretization by the trapezoidal rule, the expressions for the integrator time and the gain can be described as in eq. (5.19) [10]. $T_{samp,i}$ is the execution rate of the current regulator. It is noticed that when $T_{sampl} => 0$ the controller is the continuous version. Since the dominant time constant T_1 often is significantly larger than the $\frac{T_s amp}{2}$ it is often neglected.

$$K_p = \frac{T_1 - \frac{T_{samp,i}}{2}}{2K_s(T_{sum} + \frac{T_{samp,i}}{2})} \approx \frac{T_1}{2K_s(T_{sum} + \frac{T_{samp,i}}{2})} \tag{5.19a}$$

$$T_i = T_1 - \frac{T_{samp,i}}{2} \approx T_1 \tag{5.19b}$$

5.5 Symetrical optimum

This section contains the control theory and implementation of the tuning technique symmetrical optimum described by Nilsen [10].

When the open-loop transfer function of a system with a PI regulator can be described as in eq. (5.20) using modulus optimum canceling $1 + T_{sum}$ will lead to a phase shift of $-180°$ and hence a phase margin of 0 on the open-loop transfer function. This system will be on the stability limit and would not be a good solution. We need another method to find the PI parameters for systems like this. This method is called symmetrical optimum [10].

$$h_0(s) = K_p \frac{1 + T_i s}{T_i s} K_s \frac{1}{1 + T_{sum} s} \frac{1}{T_1 s} \tag{5.20}$$

The idea behind the symmetrical optimum is to use the numerator in the PI regulator to increase the phase margin in the frequency range the system should operate. Using the identity in eq. (5.15) and introducing the factor β the closed-loop transfer function of eq. (5.20) can be described as in eq. (5.21)[10].

$$\mathrm{M(s)} = \frac{1 + T_i s}{1 + T_i s + \frac{T_1 T_i}{K_p K_s} s^2 + \frac{T_1 T_i T_{sum}}{K_p K_s} s^3} = \frac{1 + \beta T_{sum} s}{1 + \beta T_{sum} s + \beta\sqrt{\beta} T_{sum}^2 s^2 + \beta\sqrt{\beta} T_{sum}^3 s^3} \tag{5.21}$$

By comparing the two expressions in eq. (5.21) the integrator values is defined as

$$T_i = \beta T_{sum} \tag{5.22a}$$

$$K_p = \frac{T_1}{K_s \sqrt{\beta} T_{sum}} \tag{5.22b}$$

$$\tag{5.22c}$$

It is seen that the choice of β increases the integration time, leading to the widening of the interval where the phase margin is increased. The highest phase margin is obtained at the cross over frequency corresponding to $\frac{1}{\sqrt{\beta} T_{sum}}$. Increasing the proportional gain and not β will move the crossover frequency to a higher frequency, and the crossover will not occur at the maximum phase margin.

If the controller is digital, the controller needs to be discretized. By using discretization by the trapezoidal rule the PI parameters can be calculated as in eq. (5.23)[10].

$$T_i = \beta(T_{eq} + \frac{T_{sample,n}}{2}) \tag{5.23a}$$

$$K_p = \frac{T_1}{K_s \sqrt{\beta}(T_{eq} + \frac{T_{sample,n}}{2})} \tag{5.23b}$$

$$\tag{5.23c}$$

Chapter 6

Implementation of tuning methods, startup and analysis of the current sensing circuit

6.1 Current controller

In fig. 6.1 a simple schematic of the current control loop is shown. The input is a current reference, and the output is the current that is the rotor referred q-component of the stator current. The model is decoupled, and the units used in the software need no conversion in this control loop. Each of the blocks in the block diagram can be described by the transfer functions shown in eq. (6.1).

$$PI : K_p \frac{1 + T_i s}{T_i s} \tag{6.1a}$$

$$Inverter : \frac{\frac{3}{2} U_{DC}}{1 + T_{delay} s} \tag{6.1b}$$

$$Plant : \frac{\frac{1}{R_s}}{1 + \tau_e s} \tag{6.1c}$$

$$Sensing : \frac{1}{1 + T_{sense} s} \tag{6.1d}$$

In power electronic systems, one can often end up with a dominant time constant and several smaller ones[10]. In this system, the dominant time constant is the electrical time constant τ_e, and the smaller ones is the filter time constant from the sensing circuit and the inverter switching delay. Writing the open-loop transfer function and gathering the smaller time constants, one can obtain eq. (6.2).

$$h_0(s) = K_p \frac{1 + T_i s}{T_i s} \frac{3 U_{DC}}{2 R_s} \frac{1}{1 + (T_{delay} + T_{sense})} \frac{1}{1 + \tau_e s} \tag{6.2}$$

Comparing this to modulus optimum in eq. (5.13) one can obtain the parameters in eq. (6.3).

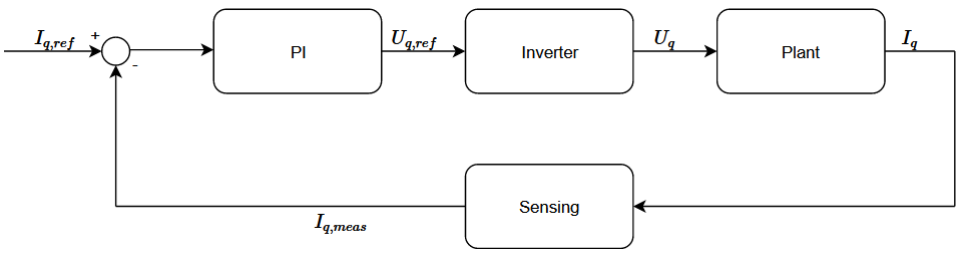


Figure 6.1: Shows a block diagram of the current control loop.

$$K_s = \frac{3U_{DC}}{2R_s} \tag{6.3a}$$

$$T_{sum} = T_{delay} + T_{sense} \tag{6.3b}$$

$$T_1 = \tau_e \tag{6.3c}$$

Using the modulus optimum and knowing the controller is digital, the gain and the integral time can be calculated as in eq. (5.19) when eq. (6.3) is inserted in eq. (5.18).

$$K_p = \frac{R_s \tau_e}{6\zeta^2 U_{DC}(T_{delay} + T_{sense} + \frac{T_{samp,i}}{2})} \tag{6.4a}$$

$$T_i = \tau_e \tag{6.4b}$$

A typical value of the dampening coefficient is $\frac{1}{\sqrt{2}}$ which is used to obtain critical dampening.

6.2 speed regulator

In the speed regulator the plant comes from the relationship between torque and inertia seen in eq. (6.6). The closed current control loop is modeled as a first order representation with the time constant T_{eq}, which corresponds to two times the current controller $T_{sum,i}$ as in eq. (6.5)[10]. $T_{sum,i}$ is defined in eq. (6.3).

$$T_{eq} = 2T_{sum,i} \tag{6.5}$$

A simplified speed control loop block diagram is shown in fig. 6.2. The units between the transfer functions must be correct to calculate the correct PI parameters. The speed error input to the speed regulator is in the unit $01Hz$. This unit is much used in the software and is defined as one-tenth of a hertz. In eq. (6.6) the unit for the speed is in $\frac{rad}{s}$. To go from $\frac{rad}{s}$ to $01Hz$ we divide by 2π to get Hz and multiply by 10 to get $01Hz$. This result in a conversion factor as shown in eq. (6.7).

The output of the speed PI regulator is $I_{q,ref}$ expressed in digits and corresponds to $T_{e,ref}$ expressed in digits. However to obtain the I_q current in Amps as the input to the plant equation eq. (6.6) a conversion constant is also needed here. This factor (K_{amp}) is dependent on the values in the current sensing circuit shown in fig. 6.5. The factor is shown and calculated in eq. (6.7). $V_{ADC,ref}$ $3.3V$, $R_{shunt} = 0.01\Omega$ and the opAmp amplification $K_{opAmp} = 5.18$. K_T is the torque constant and J is the motor inertia.

$$J\dot{\omega}_{mech} = T_e - T_L \tag{6.6}$$

$$K_{amp} = \frac{V_{ADC,ref}}{65536 R_{shunt} K_{opAmp}} = 0.000972 \tag{6.7a}$$

$$K_{freq} = \frac{10}{2\pi} \tag{6.7b}$$

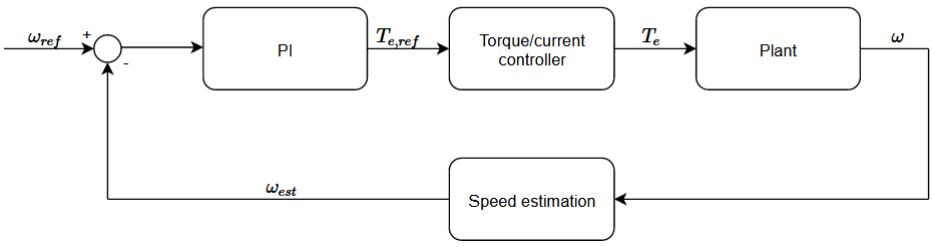


Figure 6.2: Shows a blockdiagram of the speed control loop.

$$PI : K_p \frac{1 + T_i s}{T_i s} \tag{6.8a}$$

$$Currentcontroller : \frac{K_{Amp}}{1 + Teq} \tag{6.8b}$$

$$Plant : \frac{K_T}{Js} \tag{6.8c}$$

$$Unitconversion : K_{freq} \tag{6.8d}$$

By combining the equations in eq. (6.8) the open-loop transfer function of the speed regulation loop is obtained as in eq. (6.9)

$$h_0(s) = K_p \frac{1 + T_i s}{T_i s} \frac{K_{amp}}{1 + Teq} \frac{K_T}{Js} K_{freq} \tag{6.9}$$

It is noticed that the open loop transfer function has two integrators and hence symetrical optimum is chosen for this regulator. Comparing eq. (5.20) and eq. (6.9) the constants in eq. (6.10) is found.

$$K_s = K_{amp} K_T K_{freq} \tag{6.10a}$$

$$T_1 = \frac{1}{J} \tag{6.10b}$$

$$T_{sum} = T_{eq} \tag{6.10c}$$

Knowing the controller is digital the speed regulator parameters can be calculated as in eq. (6.11) by inserting eq. (6.10) into eq. (5.23).

$$T_i = \beta T_{eq} \tag{6.11a}$$

$$K_p = \frac{T_1}{K_{amp} K_T K_{freq} \sqrt{\beta} (T_{eq} + \frac{T_{samp}}{2})} \tag{6.11b}$$

$$\tag{6.11c}$$

6.3 Startup

In this section both the theory of the starup and the implementation is described. When using sensorless algorithms based on the back-emf of the motor, the angle estimation is insufficient at low angular velocity. It does not function at zero speed since there is no back emf. Because of this, a startup method needs to be implemented to find an initial angle and set an initial spin to the rotor such that the sensor-less back-emf method can take over and continue with FOC. One way to do this for PMSM is I/f control described by Wang et al. [19] and is tried implemented here.

The method can be divided into four parts. First, it aligns the rotor to an initial angle. This is done by setting a DC current in the stator a-phase. By doing this, the rotor will be locked in a known position. This is illustrated in fig. 6.3.

The second leg of the startup is to start rotating the stator's magnetic field. When the stator magnetic field starts rotating, the rotor will follow. This will give an initial spin to the machine, which will give a back-emf such that the sensor-less algorithm can estimate the rotor position and run the motor on FOC.

The third leg is lowering the current in the stator windings such that the switchover to FOC will not cause large current ripples. The fourth leg is the FOC switchover.

To describe the startup method Wang et al. [19] use both a rotor refered $d-q$ reference frame and a stator refered d^*-q^* reference frame. These frames is illustrated in fig. 6.4.

Implementation

Stage 1:

As shown in fig. 6.3 a magnetic field is set up by the stator a-phase DC current. This aligns the rotor $d-axis$ with the stator $a-axis$ with the permanent magnet's magnetic field but in the opposite direction. Effectively this sets the stator I_q current to be lagging the rotor reference frame by 90°. This will result in a torque of zero. If the rotor is not perfectly aligned with the $a-axis$, the I_q current will not be lagging by 90° and will produce a torque on the rotor that is directed back to perfectly aligned. This can be shown by the equation in eq. (6.12), and is further illustrated in fig. 6.4[19].

$$T_e = \frac{3}{2} P I_q^* \cos\theta_L \left(\Psi_\mathrm{m} + (L_d - L_q)\, I_q^* \sin\theta_L\right) \approx K_T I_q^* \cos\theta_L \tag{6.12}$$

Here P is nuber of pole pairs and K_T the torque constant. If θ_L is 90° this will result in a torque of 0.

Stage 2:

During the ramp-up of the speed, the amplitude of I_q is kept constant with the value used in the alignment stage of $\frac{I_{nom}}{2}$ and I_d is 0(SPMSM). When the stator current I_q starts to spin $\theta_L < 90°$ which leads to $T_e > 0$ in eq. (6.12). θ_L will increase until the torque is high enough to get the rotor to start rotating. If the load is too high to get the rotor to start spinning, higher torque can be obtained by increasing the current, or a lower load torque can be obtained by lowering the acceleration.

It is important to notice that if θ_L becomes negative, meaning that the $q*-axis$ leads the $q-axis$ rotor frame in fig. 6.4, the torque will become negative while the stator reference frame is increasing its speed. The two reference frames will lose sync, and the startup will fail. Since the torque needed to accelerate the rotor depends on the product between I_q and $cos(\theta_L)$, the torque needed to spin the rotor can be obtained by a smaller θ_L by increasing I_q. This means that a higher I_q will give a lower probability of I_q leading I_q^*. This will be more important when the blades from the helicopter are attached since this will need a higher load torque.

Baring this in mind, the steepness of the ramp has to be limited. This can be done by introducing a factor K_ω that defines a linear ramp of the electric frequency ω_e as in eq. (6.13) where t is time in seconds.

$$\omega_e = K_\omega t \tag{6.13}$$

To find this constant, the plant equation from the speed controller, eq. (6.6), is modified to be expressed with electric angular velocity as in eq. (6.14).

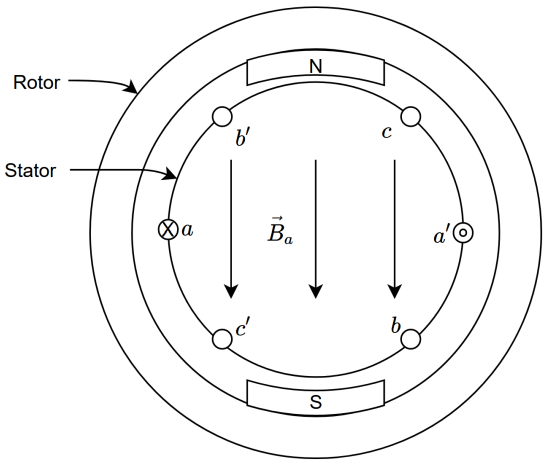


Figure 6.3: Illustrates how the stator is aligned with the rotor such that the rotor is locked in the alignment stage.

.

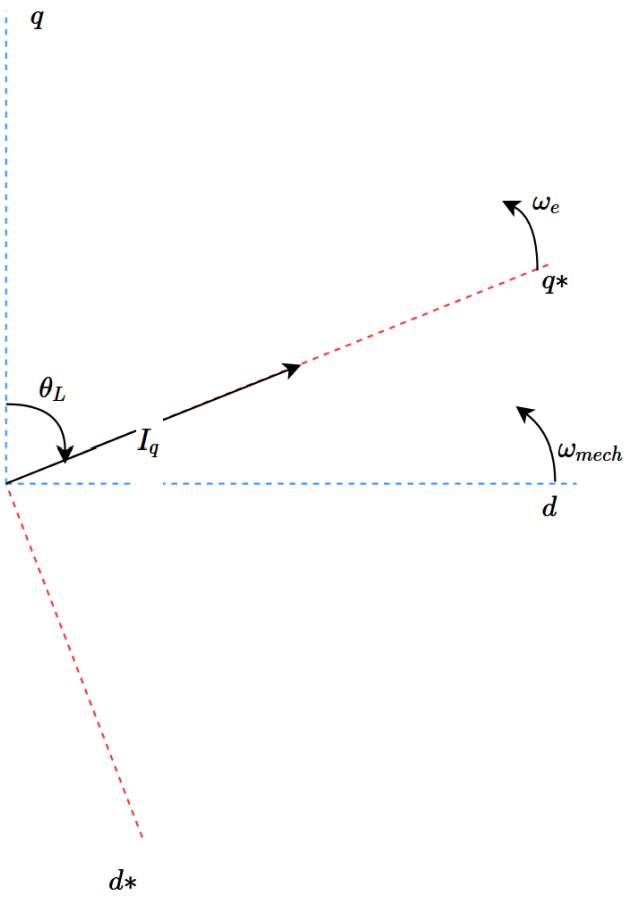


Figure 6.4: Shows how θ_L is affecting the discrepancy between the rotor reference frame and the stator reference frame.

. .

$$\frac{J}{p}\dot{\omega}_e = T_e - T_L \tag{6.14}$$

Since the final speed is dependent on the average effective speed, a simplification is made such that the average electric torque is defined in eq. (6.15).

$$T_{e,avg} = K_T I_q \cos(\theta_{L,avg}) \tag{6.15}$$

The average load torque is the friction torque. When a load is implemented, the average torque from the propeller curve of the drone blades between 0 and FOC switch over speed should be added here.

Inserting eq. (6.15) in eq. (6.14) the result is eq. (6.16).

$$\dot{\omega}_e = \frac{p}{J}(K_T I_q \cos(\theta_{L,avg}) - T_{L,avg}) \tag{6.16}$$

By integrating this equation eq. (6.13) is obtained where K_ω and is eq. (6.17).

$$K_\omega = \frac{p}{J}(K_T I_q \cos(\theta_{L,avg}) - T_{L,avg}) \tag{6.17}$$

As described above, a negative θ_L will make the startup routine fail. By setting $\theta_L = 0$ this gives the relation in eq. (6.18).

$$K_\omega < \frac{p}{J}(K_T I_q - T_{L,avg}) \tag{6.18}$$

stage 3:

The rotor should now have a spin that enables sufficient sensing of the back-emf. However, the reference frame $d-q$ and d^*-q^* are almost 90° shifted from each other, and the amplitude of Iq is high. This may lead to problems like high current ripples[19].

To counter this, the current is gradually decreased. This is done such that θ_L also decrease - balancing the loss in torque from decreasing the current with a decreasing θ_L. The FOC implemented for this motor control directs all the torque-producing current in the $q-axis$ as described earlier. The idea is to get close to this state before switching. This is done with a linear decrease and can be described with the help of K_i as in eq. (6.19).

$$I_q^*(t) = I_{q0} - K_i(t - t_0) \tag{6.19}$$

Here K_i describes the rate at which the current decrees from its value at the alignment and the ramp-up to the value for the FOC switchover. Since the speed is constant through the decrease in current, there is no acceleration hence eq. (6.6) is 0. The load torque will be the same after the ramp-up. Using this and inserting eq. (6.12) into eq. (6.6) and integrating it eq. (6.20).

$$\int_{t_0}^{t_0+T_{\text{set}}} I_q^*(t)\cos\left(\theta_L(t)\right)dt = I_{q0}^*\cos\left(\theta_{L(t_0)}\right)T_{\text{set}} \tag{6.20}$$

Here θ_L is also modeled as a linearly decreasing function as in eq. (6.21), and T_{set} is the time θ_L use to go to zero.

$$\theta_L(t) = \theta_L\left(t_0\right) - \frac{\theta_L\left(t_0\right)}{T_{\text{set}}}\left(t - t_0\right) \tag{6.21}$$

Inserting eq. (6.19) and eq. (6.21) into eq. (6.20) K_i is obtained as in eq. (6.22) where $K_\theta = \frac{\theta_L(t_0)}{T_{\text{set}}}$.

$$K_i = I_{q0}^*\frac{\theta_{L(t_0)}\cos\left(\theta_{L(t_0)}\right) - \sin\left(\theta_{L(t_0)}\right)}{\cos\left(\theta_{L(t_0)}\right)}K_\theta \tag{6.22}$$

For this application, θ_L will be close to 90° since the motor is running in no-load. Using the friction torque from the datasheet of $T_{fr} = 3.53mNm$ to balance eq. (6.14) this gives a θ_L of 89.64°. The division of $\cos(\theta_L)$ will result in a K_i that is very high. This means that this method will not be suited for no-load operation. In Wang et al. [19] eq. (6.22) is said to be a rough estimate. A fast transition is not needed, and a K_i of 4 is chosen for this project. When load is added to the system, the calculation of K_i could be tried once more.

Stage 4:

The last step is to switch from the rev-up routine to FOC. The stages above are designed to make this switch possible. In Wang et al. [19] a load angle $\theta_L < 5°$ is used as a trigger the switch to FOC. The way MC WorkBench implements the rev-up routine, the load angle can not be used to trigger this. This can be added to the software but was not prioritized. Instead, the sensorless algorithm is set to start after the current is lowered to a set value. This value is set to be $0.1A$ and calculated from eq. (6.14) balancing the friction torque assuming $\theta_L = 0$ at the end of the down ramp. The low current will result in a low θ_L such that the intention of the method is maintained.

The switchover is implemented through the function "REMNG_ExecRamp(...)". This sets $I_{q,ref}$ to the measured I_q current at the moment where the FOC switch over is triggered with a ramp time of $25ms$.

6.4 Analysis of the current sensing circuit in IHM08M1

A simplified drawing of the IHM08M1 with a 3-shunt low-side sensing circuit is shown in fig. 6.5 for one of the three identical shunts.

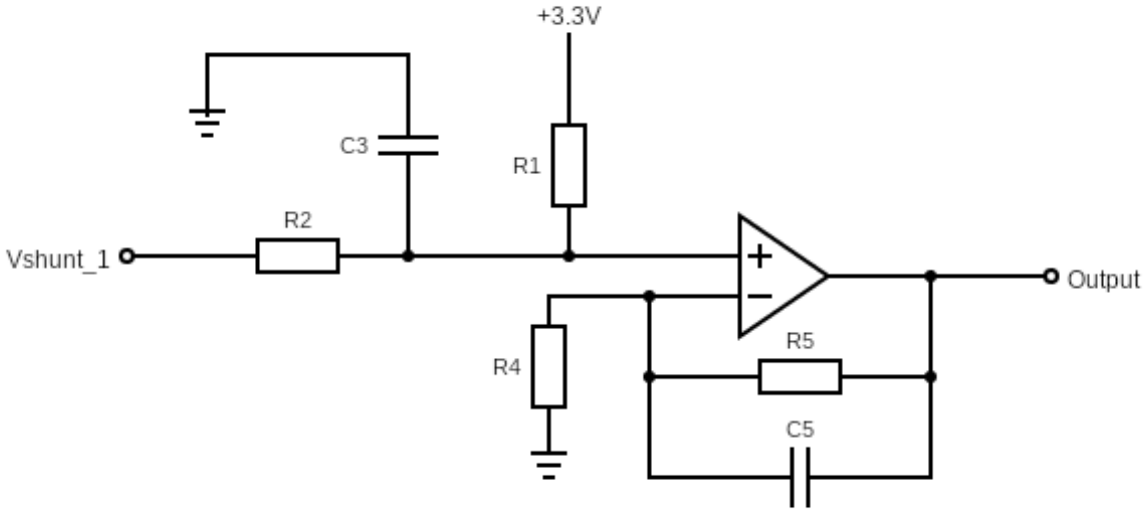


Figure 6.5: Schematic of one of the three shunts in the three shunt sensing circuit[13]

To study how the sensing circuit is affected by the switching frequency the transferfunction of the sensing circuit from $Vshunt\_1$ to $Output$ in fig. 6.5 is calculated. The transferfunction is given in eq. (6.23), where the constants are given in eq. (6.24).

$$H(s) = K_p \frac{T_1 s + 1}{(T_2 s + 1)(T_3 s + 1)} \tag{6.23}$$

$$K_p = \frac{R_4}{(R_1 + R_2)(R_4 + R_5)} \tag{6.24a}$$

$$T_1 = \frac{R_4 R_5 C_4}{R_5 + R_4} \tag{6.24b}$$

$$T_2 = R_5 C_4 \tag{6.24c}$$

$$T_3 = \frac{R_1 R_2 C_3}{R_1 + R_2} \tag{6.24d}$$

The Bode plot of the transfer function is seen in fig. 6.6 marked as "$C_3 = 15nF$", with the hardware component values listed in table 6.1. It is seen that the sensing circuit is stable, but the cutoff frequency f_c, which is defined as the frequency for which the gain has dropped by $-3dB$, is $17kHz$. This means that currents with higher frequencies will be filtered out, resulting in PWM frequencies over this will likely have a bad performance. If a switching frequency of $17kHz$ and higher is wanted, another sensing circuit is needed. Since the inductance of slotless machines usually is low, a high current ripple will be a problem for such motor types. One solution for this is using a high switching frequency [6].

Since this motor is supposed to run for higher frequencies C_3, C_5 and C_7 was removed resulting in the input

Table 6.1: The values of the hardware components in the sensing circuit of the IHM08M1 hardware[13].

Component	Value	Unit
R_1	6800	Ω
R_2	680	Ω
R_4	1000	Ω
R_5	4700	Ω
C_3	15	nF
C_4	100	pF

filter being removed. Whats left of the sensing circuit transferfunction when removing C_3 is seen in eq. (6.25) where the constants in eq. (6.24) still counts.

$$H(s) = K_p \frac{T_1 s + 1}{(T_2 s + 1)} \tag{6.25}$$

This gives a frequency response as shown in fig. 6.6 marked as $C_3 = 0$. The cutoff frequency ω_c is $333kHz$. This is way higher than the controller intended for and may introduce unnecessary noise because of the lack of filtering. It is also noticed that the gain stabilizes at $-15dB$ corresponding to a factor of 0.18 and will lead to high-frequency noise that will not be filtered and affect the measurements. However, the sensing is stable, and the motor operates with this setup.

Further, in developing this control platform, the input filter capacitance should be reevaluated. Jensen [6] proposed a switching frequency of $60kHz$ for slotless motors. Adding a margin of $10kHz$, achieving a f_c of $70kHz$ the capacitance in the input filter should be $3.5nF$. The bode plot of a sensing circuit with $C_3 = 3.5nF$ is shown in fig. 6.6. It is seen that decreasing the value of C_3 will shift the gain to a higher frequency. This makes it easy to choose a suitable sensing circuit when the switching frequency for the next iterations is decided. This discussion assumes this is $60kHz$.

Since this hardware is a development powerboard meant to suit several needs, some alterations may be done to fit the application. In this thesis, we were not able to change the capacitators and test how this affected the current measurement. To run the control scheme at $30kHz$ removal of $C3, C5$, and $C7$ is done, knowing this is not optimal.

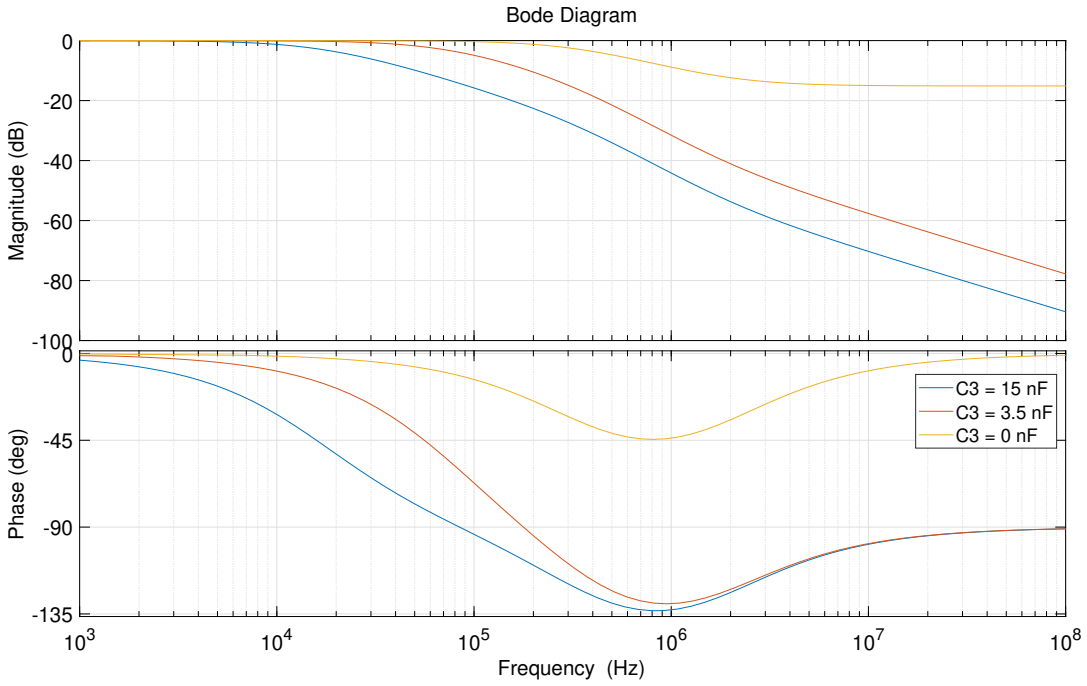


Figure 6.6: Bodeplot of the current sensing circuit with three different values of the input filter capacitance C_3, C_5 and C_7 of $15nF$, $3.5nF$ and $0nF$. The cutoff frequency for the different options is $17kHz, 70kHz$ and $333kHz$ respectively.

Chapter 7

Testing and results

7.1 Preparations Hardware

The control in this thesis is developed for a PMSM machine. The motor available for testing was a BLDC[17] and is hence used. The testing is run on the STMicroelectronics control platform made from the control board Nucleo STM32G431RB[14] and the powerboard X-Nucleo IHM08M1[13]. The system is run on STMicroelectronics control software. The hardware components and the software is described in part I.

7.1.1 Nucleo board

On the Nucleo board, set jumpers JP1 open. This is only closed if the Nucleo board and the extension board power consumption is lower than $100mA$ and is provided through the CN1 USB input.

JP5 is set to the setting E5V and lets the Nucleo board and the extension board be powered by the J1 DC input connector. Internal voltage regulation is connected through the R170 0Ω resistor on the power board. This keeps the voltage between $4.75 - 5.25V$. This resistor can be removed to obtain higher voltages but is not done here.

JP6 is closed. This is used to measure the power consumption of the Nucleo board and can be done by connecting an ammeter in series between the pins on JP6. This is not done here and is therefore closed with a jumper.

7.1.2 Powerboard

J9 is set to open. This jumper is closed if supplying the Nucleo board directly from the J1 DC supply is wanted. This must be done in combination with removing R170 mentioned above. This possibility is not used here.

JP3 is closed. This enables pull-up resistor in the hall/encoder detection circuit and is used to help define which state the PIN connected is in to determine the motor spin. This setting has no effect on the setup for the testing in this thesis since the speed and position are estimated.

FOC jumpers JP1 and JP2 are closed, and the J5 and J6 solder bridge are switched from the 1-Sh side to the 3-Sh side. This makes the sensing circuit measure all three phases as described in section 2.4.

7.1.3 Physical changes of the powerboard

To make the powerboard work for the intended application, some additional alterations are done to the off-the-shelf powerboard. A zero-ohm resistor R181 and the capacitators C_1, C_2, and C_3 are removed. The soldering of a 1-shunt/3-shunt jumper is moved from 1-shunt to 3-shunt. The equipment and powerboard is shown in fig. 7.1 and the alteration is shown in fig. 7.2.

The equipment used is:

- Soldering bolt
- Solder suck wire

- tweezers

The removal of R181 removes the usage of a potentiometer on PA4 and enables the usage of DAC instead. The alteration of the soldering to 3-sh enables 3-shunt current sensing, which is crucial for FOC. The removal of the capacitances changes the filter as described in section 6.4.

7.1.4 Motor

The motor profile in the software is implemented with the values listed in table 7.1.

Table 7.1: Shows the motor data needed by the Motor Control Work Bench to calculate control parameters[17]

Parameter	Value	Unit
Pole pairs	4	
Max application speed	7500	RPM
Nominal current	21	Apk
Nominal DC Voltage	24	V
Rs	0.1	Ω
Ls	0.02	mH
B-Emf constant	5	$\frac{Vrms}{krpm}$

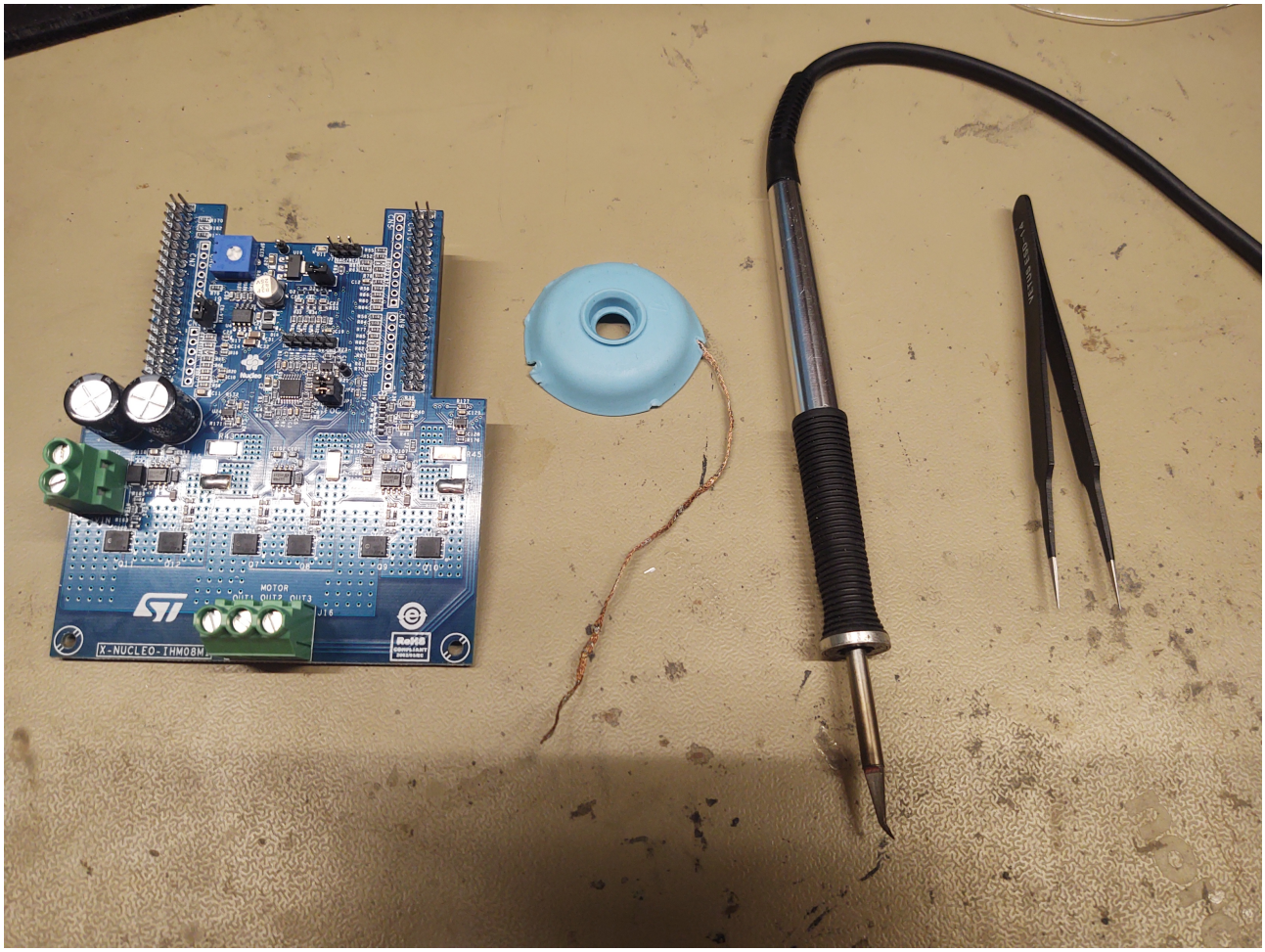
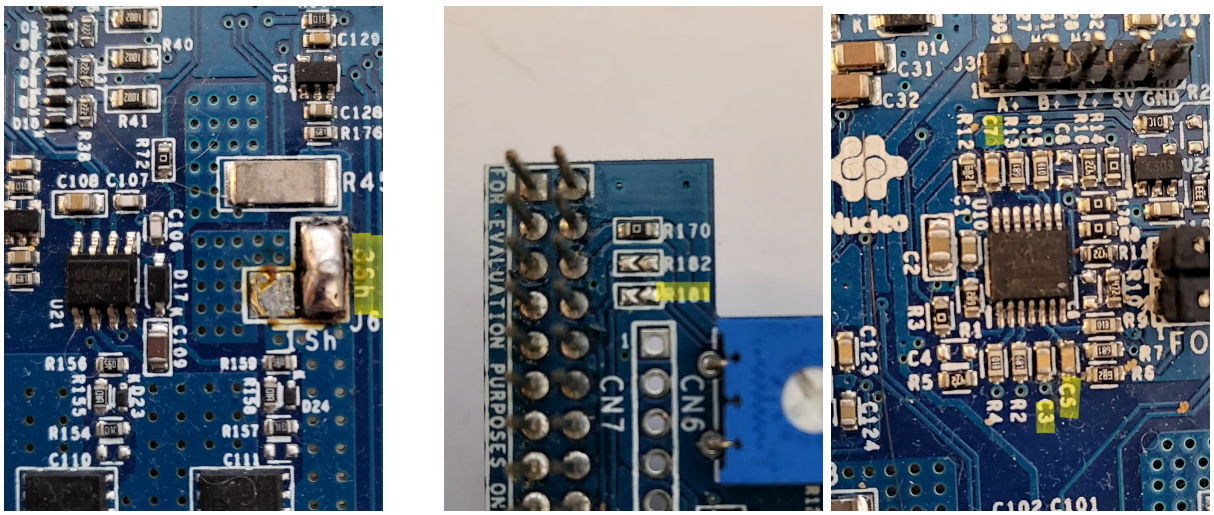


Figure 7.1: The equipment used to alter the powerboard to fit the application



(a) 3Sh **(b)** R181 **(c)** C3, C5, C7

Figure 7.2: Shows the physical alterations done on the IHM08M1 powerboard. The components that was altered is marked in yellow.

7.2 Results

7.2.1 Start up

All startup tests are done with the current regulator and speed regulator values listed in table 7.2.

Table 7.2: This table show the speed and the current controller gains used when the startup method is tested. The values are listed as they are used in the software.

Controller	K_p	Kp, div	K_i	$K_{i,div}$
Current	1819	16384	334	16384
Speed	2321	4	2166	128

Test 1:

Table 7.3: The table shows the start up parameters of test 1.

Stage	time [ms]	$I_{q,ref}$[A]	speed [RPM]
Alignment	0-500	0-10.5	0
Speed ramp	500-1869	10.5	0-1000
Current down ramp	1869-4244	10.5-0.1	1000
Switch over	-	-	-

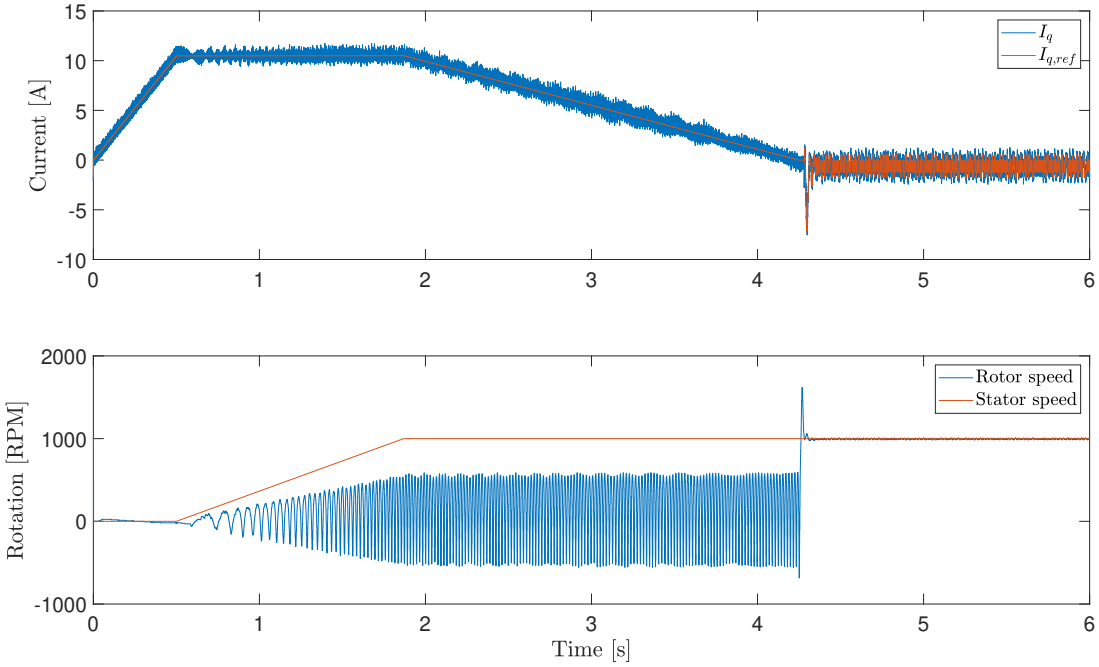


Figure 7.3: Shows the speed and current ramp with the rev-up scheme of table 7.3

Figure 7.3 shows that the startup succeed when the startup sequence in table 7.3 is implemented. A current spike in the switchover is noticed.

Test 2:

Table 7.4: The table shows the start up parameters of test 2.

Stage	time [ms]	Current reference [A]	speed [RPM]
Alignment	0-500	10.5	0
Speed ramp	500-1869	10.5	1000
Current down ramp	1869-4244	0.1	1000
Switch over	4244-4269	measured	-

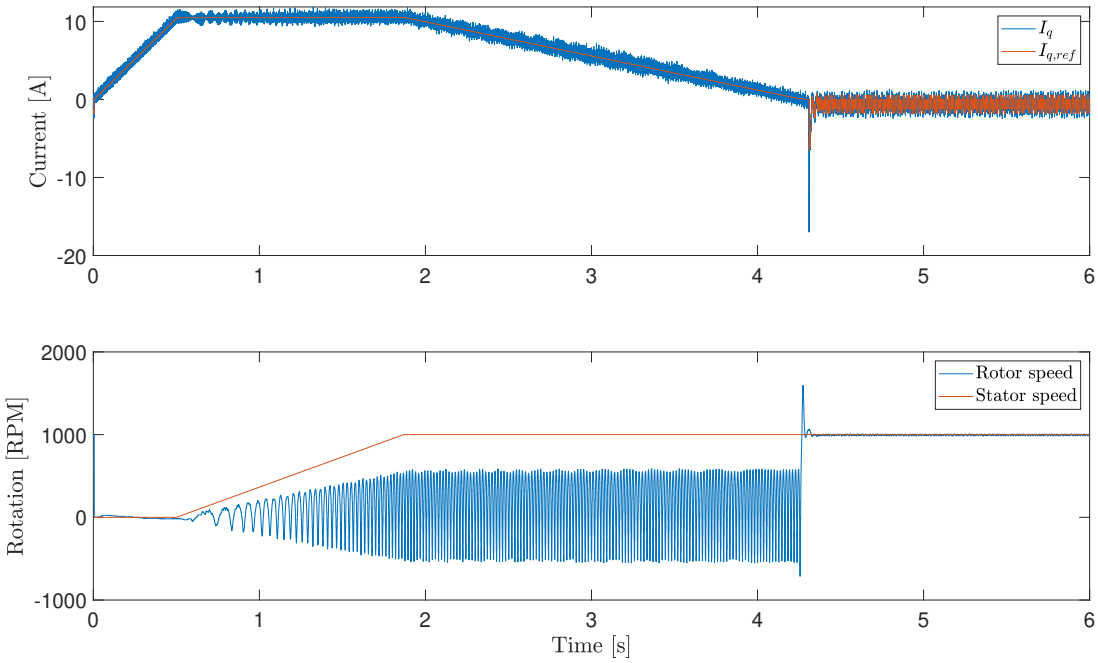


Figure 7.4: Shows the speed and current ramp with the rev-up scheme of table 7.5

Figure 7.4 shows that the startup succeed when the startup sequence in table 7.4 is implemented. A large current spike in the switchover is noticed.

Test 3:

Table 7.5: The table shows the start up parameters of test 2.

Stage	time [ms]	Current reference [A]	speed [RPM]
Alignment	0-500	10.5	0
Speed ramp	500-1869	10.5	1000
Current down ramp	1869-4244	1	1000
Switch over	4244-4269	measured	-

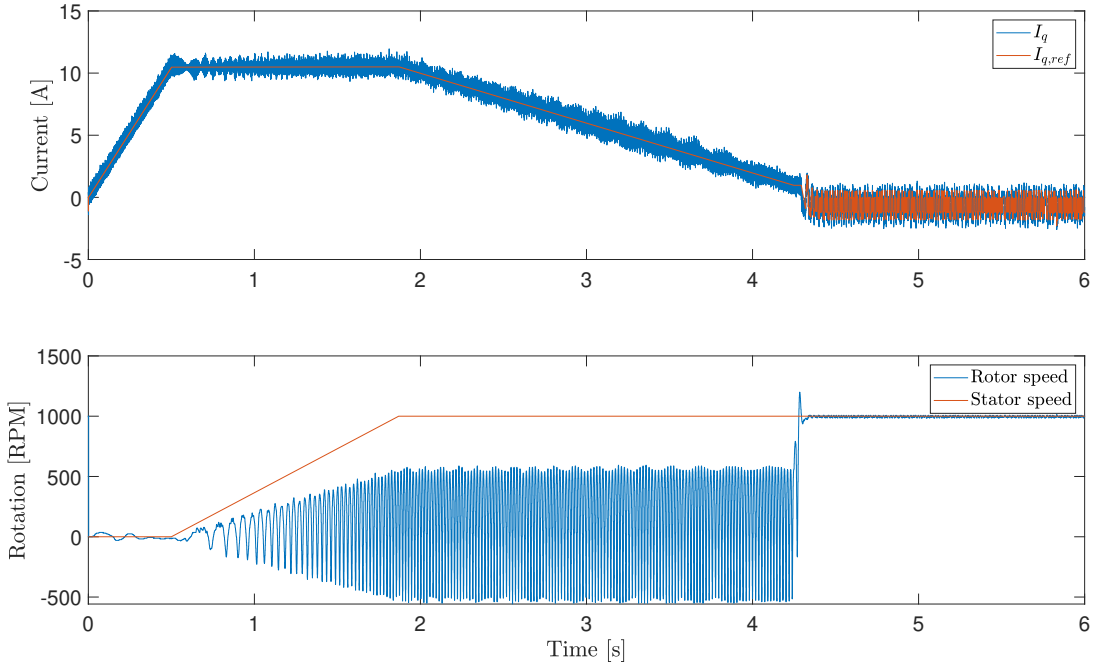


Figure 7.5: Shows the speed and current ramp with the rev-up scheme of table 7.5

Figure 7.5 shows that the startup succeed when the startup sequence in table 7.5 is implemented. The transition in the switch over is smooth.

7.2.2 Current controller

All current regulator tests are done with the speed regulator values listed in table 7.6.

Table 7.6: This table show the speed controller gains used when the current controller is tested. The values are listed as they are used in the software.

Controller	K_p	Kp, div	K_i	$K_{i,div}$
Speed	2321	4	2166	128

Table 7.7: Shows the values used for the testing of the current regulator. For the software converted values $Kp_{div,i}$ and $Ki_{div,i}$ is 16384.

ζ		K_p	$T_i[ms]$	K_i
$\frac{1}{\sqrt{2}}$	Calculated	0.0070	0.182	38.23
	Converted for SW	114		21
$\frac{1}{\sqrt{4}}$	Calculated	0.0139	0.182	76.45
	Converted for SW	206		42
$\frac{1}{\sqrt{8}}$	Calculated	0.0278	0.182	152.90
	Converted for SW	455		84
$\frac{1}{\sqrt{16}}$	Calculated	0.0565	0.182	305.81
	Converted for SW	926		167
$\frac{1}{\sqrt{32}}$	Calculated	0.111	0.182	611.62
	Converted for SW	1819		334
$\frac{1}{\sqrt{64}}$	Calculated	0.223	0.182	1223.24
	Converted for SW	3654		668

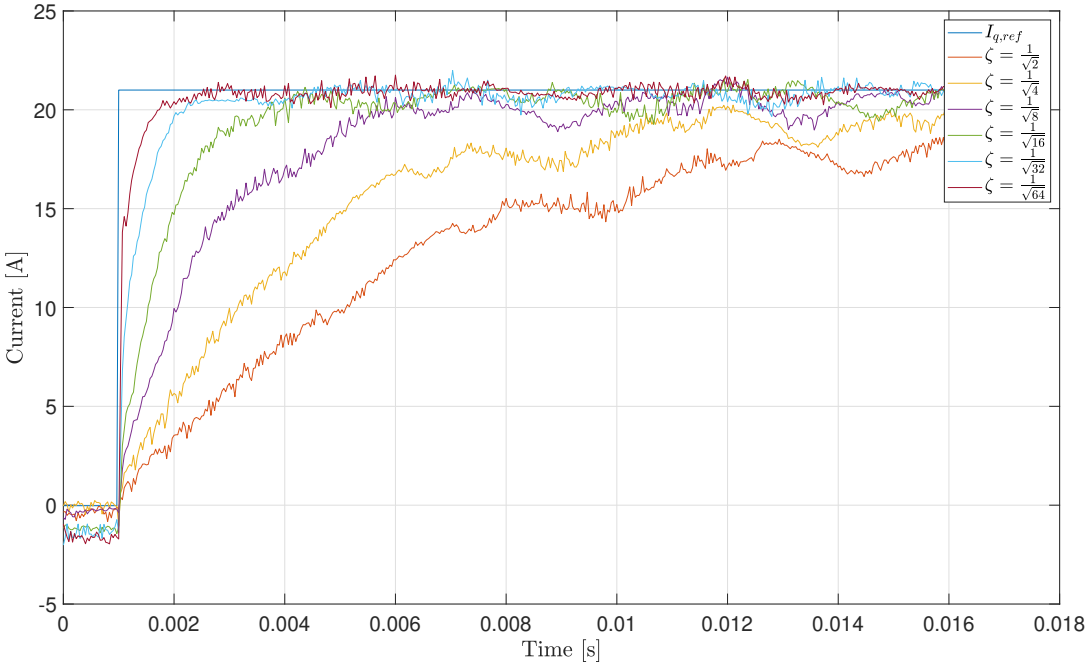


Figure 7.6: This figure shows how the choice of six different ζ values affect the response to a maximum current reference $I_{q,ref}$. The reference is achieved by setting a reference of $3000RPM$ whit an initial rotation of $1000RPM$.

Table 7.8: Shows the rise time and the risetime devided by Tsum.

ζ	Trise $(10-90\%)$ [ms]	Trise/Tsum
$\frac{1}{\sqrt{2}}$	15.20	418.34
$\frac{1}{\sqrt{4}}$	8.63	237.52
$\frac{1}{\sqrt{8}}$	3.86	106.24
$\frac{1}{\sqrt{16}}$	1.73	47.61
$\frac{1}{\sqrt{32}}$	0.90	24.77
$\frac{1}{\sqrt{64}}$	0.47	12.85

Figure 7.6 shows the step response of the current regulator when tested with the gains in table 7.7. The risetime from $10\% - 90\%$ is listed in table 7.8 in ms and in number of T_{sum}.

7.2.3 Speed control

All speed regulator tests are done with the current regulator values listed in table 7.9. The values used for the three tests of the speed regulator is shown in table 7.10.

Table 7.9: This table show the speed controller gains used when the current controller is tested. The values are listed as they are used in the software.

Controller	K_p	Kp, div	K_i	$K_{i,div}$
Current	1819	16384	334	16384

Table 7.10: Shows the calculated values for the testing of the speed regulator.

β		K_p	$T_i[ms]$	K_i
4	Calculated	5753	2.16	2666505
	Converted for sw	5753		711
10	Calculated	3638	5.39	674578
	Converted for sw	3638		180
10	Kp/10	363.8	5.39	67495
	Converted for sw	36		18

Test 1: $\beta = 4$

fig. 7.7 and fig. 7.8 shows the estimated speed and measured $q - axis$ current respectively plotted with its references when a β of 4 is used in the symmetrical optimum method.

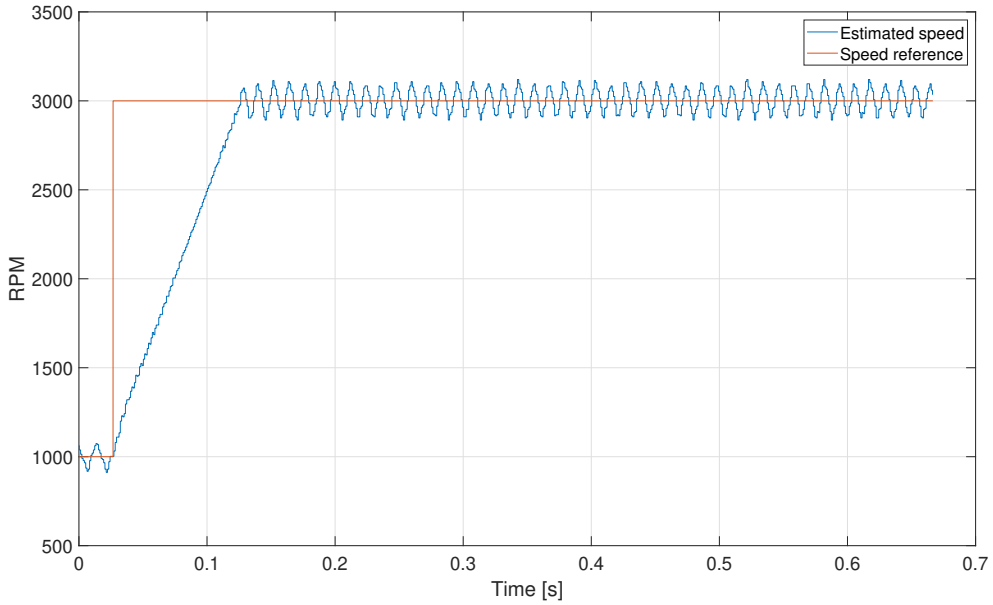


Figure 7.7: The figure shows the response in the estimated speed when a reference of $3000RPM$ is set from the initial rotation of $1000RPM$. The regulator is tuned with symmetrical optimum with a β-value of 4.

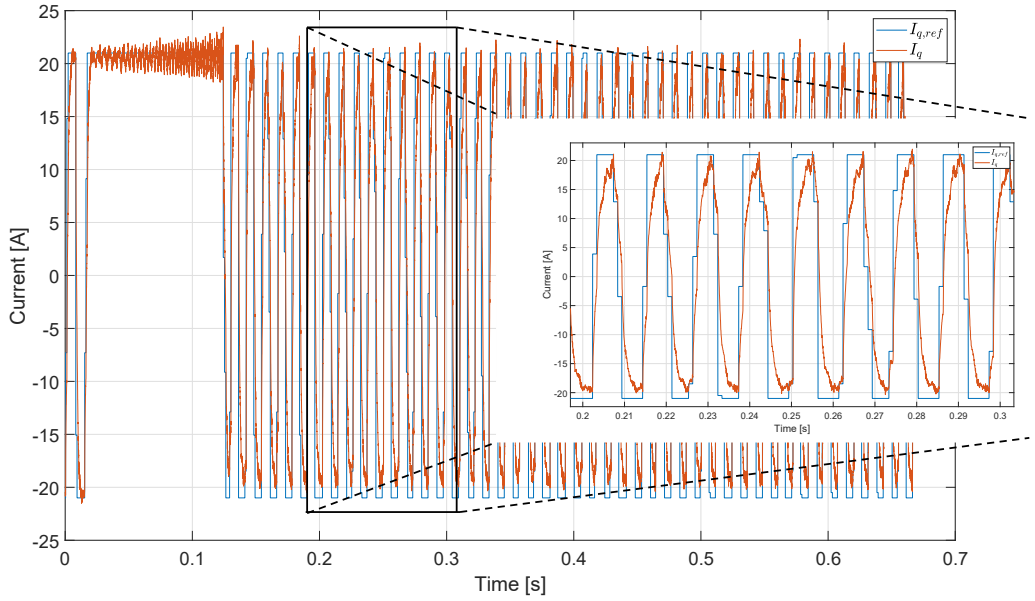


Figure 7.8: The figure shows the response in the q-axis current reference and measurement when a reference of $3000RPM$ is set from the initial rotation of $1000RPM$. The regulator is tuned with symmetrical optimum with a β-value of 4. The plot between $0.2 - 0.3s$ is enlarged

Test 2: $\beta = 10$

fig. 7.9 and fig. 7.10 shows the estimated speed and measured $q - axis$ current respectively plotted with its references when a β of 10 is used in the symmetrical optimum method.

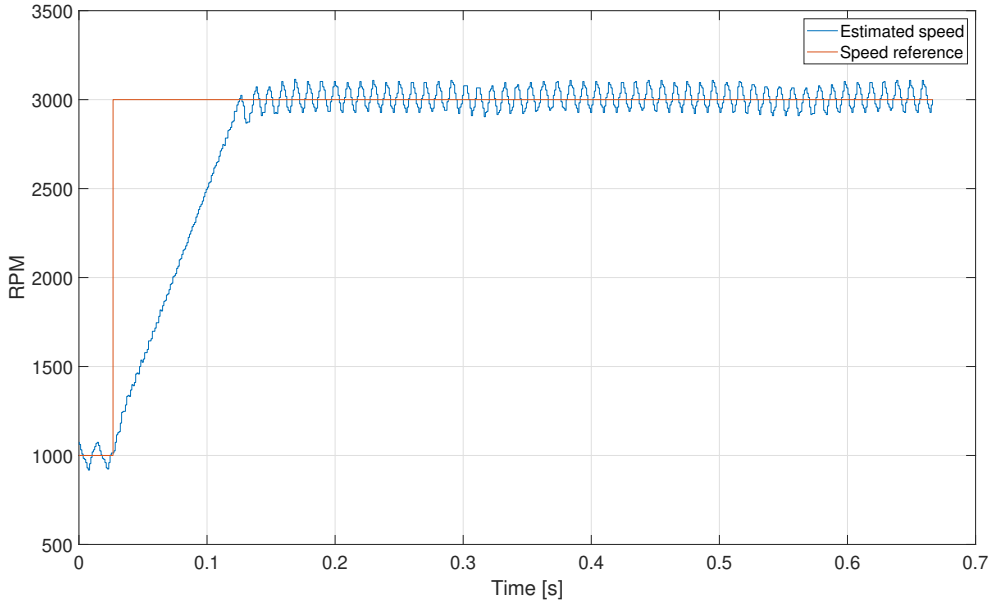


Figure 7.9: The figure shows the response in the estimated speed when a reference of $3000RPM$ is set from the initial rotation of $1000RPM$. The regulator is tuned with symmetrical optimum with a β-value of 10.

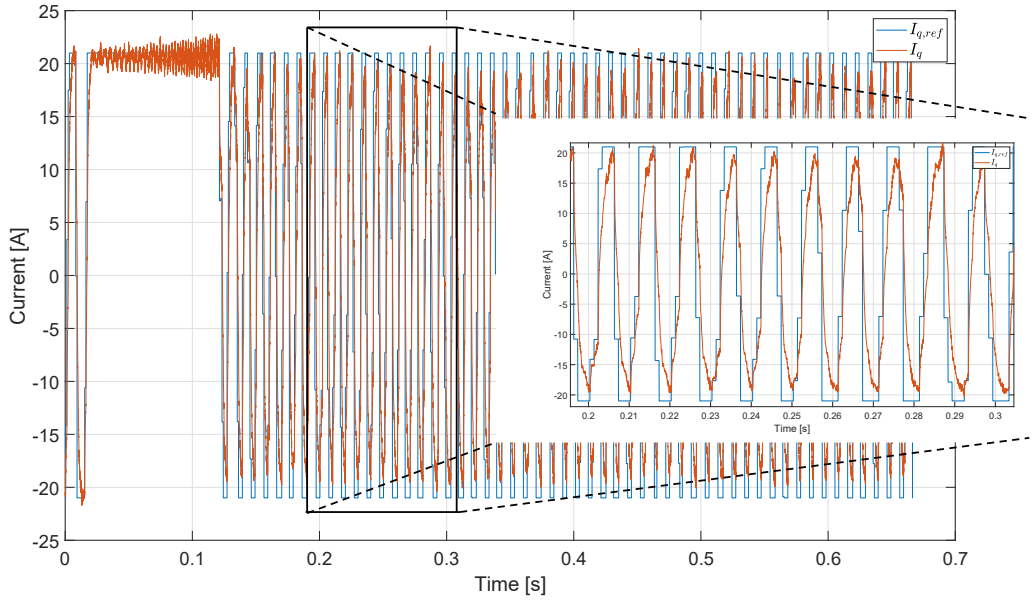


Figure 7.10: The figure shows the response in the q-axis current reference and measurement when a reference of $3000RPM$ is set from the initial rotation of $1000RPM$. The regulator is tuned with symmetrical optimum with a β-value of 10. The plot between $0.2 - 0.3s$ is enlarged

Test 3: $\beta = 10$**,** $K_p/10$

fig. 7.9 and fig. 7.10 shows the estimated speed and measured $q - axis$ current respectively plotted with its references when a β of 10 is used in the symmetrical optimum method in addition to decreasing the proportional gain by a factor of 10.

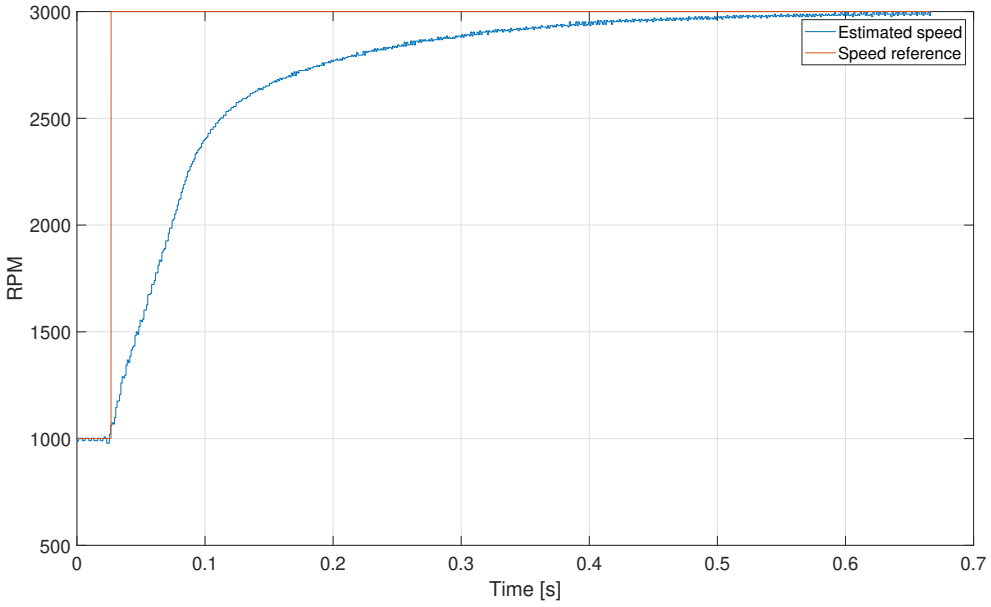


Figure 7.11: The figure shows the response in the estimated speed when a reference of $3000RPM$ is set from the initial rotation of $1000RPM$. The regulator is tuned with symmetrical optimum with a β-value of 10 and dividing the proportional gain K_p by a factor of 10 keeping the integral time T_i.

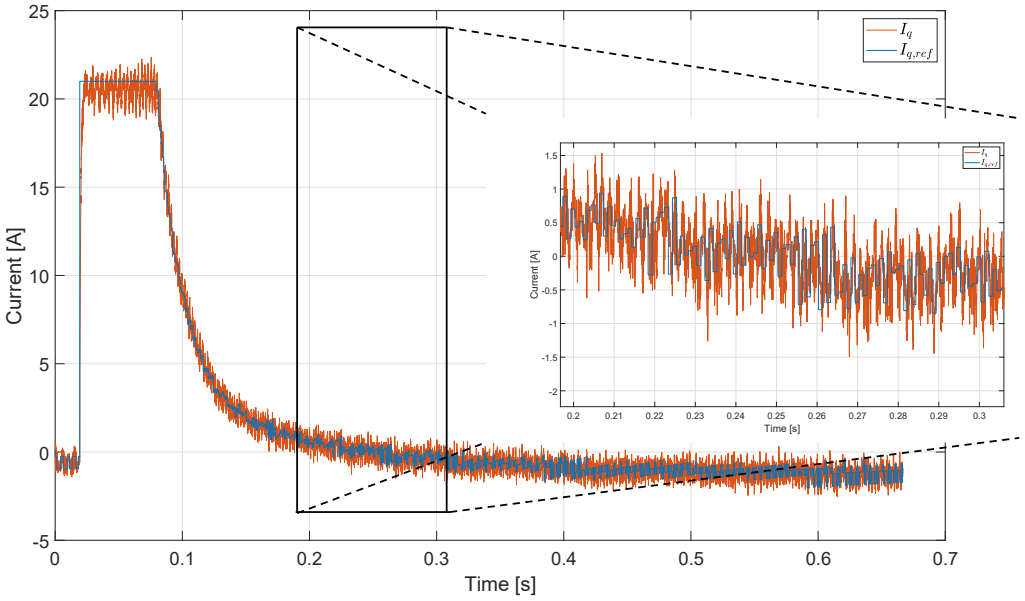


Figure 7.12: The figure shows the response in the q-axis current reference and measurement when a reference of $3000RPM$ is set from the initial rotation of $1000RPM$. The regulator is tuned with symmetrical optimum with a β-value of 10 and dividing the proportional gain K_p by a factor of 10 keeping the integral time T_i. The plot between $0.2 - 0.3s$ is enlarged

Chapter 8

Discussion

8.1 Start up

For early tests of the startup routine, it was observed that the rev-up seceded inconsistently. By visual observation, the failure rate was high when the rotor wobbled in the alignment stage. A theory of why this happened is that if the position of the rotor is such that when the DC current in the alignment stage is pressed, the rotor will accelerate to the center of the magnetic field pressed in the coil of phase a. However the rotor has now kinetic energy and will swing right through the equilibrium and start decelerating. Because of the low friction of this motor, this oscillation will have almost no dampening. If the swing is in the opposite direction of the spin of the stator magnetic field when it starts to accelerate, the rotor falls off the stator magnetic field rotation ω_e.

The oscillation will dampen, and the rotor will be properly aligned when the stator magnetic field starts rotating by applying a temporary friction force on the rotor in the alignment stage. This was tested by lightly pressing rubber on the rotor to increase the dampening of the oscillation until it started to spin. The result was 100% success rate when testing the startup 10 times. The oscillation was also tried to minimize by both increasing and decreasing the alignment DC current without any significant increase in success rate. This problem may not occur when the blades of the drone are attached to the rotor since this will introduce some energy loss because of air resistance on the larger area and the longer moment arm of the friction force on the blades. If this is not solved by adding load, another approach to the alignment phase must be considered.

In fig. 7.3 it is seen that the implementation of the startup from table 7.3 succeeds. The alignment phase works well by adding some friction, as discussed earlier. The current down ramp also has no problems. However, the current reference falls quickly when the FOC switchover occurs. The FOC switchover is triggered when the rotation exceeds $1000RPM$, and this value is measured by the sensorless estimation. When the sensorless speed estimation is triggered, which is the case when the current down ramp succeeds, the speed estimation quickly estimates a speed of over $1000RPM$, and the FOC is triggered. However, the estimation at this moment has a significant overshoot as seen in fig. 7.3. Since the FOC starts simultaneously as the sensorless algorithm overshoots its estimation, the current reference is set to about negative $7A$ to decelerate the motor. Since the overshoot comes from the incorrect estimation and is not too high, this compensation is unnecessary and unwanted. By looking at the response in fig. 7.3 it was seen that the estimation overshoot came down to the correct speed of $1000RPM$ after $20ms$. A switchover time was set to $25ms$ such that the speed estimation should settle and give it time to stabilize before the FOC kicks in. This was implemented in test 2.

The results of the adjustment done from test 1 to test 2 did not give the expected results. The current reference after switchover was still significantly negative, and in addition a large current spike in the current measurement was experienced. This is seen in fig. 7.4. The same spike was observed when plotting the a and b phase currents directly. Since the current spike vastly exceeded the reference, and the current at switchover being close to zero, it was suspected that the measurement of the current, which is used as a reference in the current regulator in the switchover period, is inaccurate. In addition to be inaccurate, it may be amplified by the negative reference. It may also come from a θ_L that may go negative or have some oscillation at the moment it is switched over. As a response to this, increasing the amperage to $1A$ was done to avoid being close to $0A$. The new current reference of $1A$ is implemented in test 3.

In test 3 big ripple effect in the switch over period is not observed. This is seen in fig. 7.5. It can be observed that the switchover current ramp is lower than $1A$ leading to a decrease in the current I_q in this period. This indicates that the load angle θ_L was not close to 0 at the switchover, but this did not seem to cause any problems

since it was close enough when the switchover kicked in.

A problem with the implementation of startup through the GUI is that the possibilities are limited. The FOC is triggered by a minimum RPM. In the startup strategy I/f this speed is obtained with a current lag that is almost 90° because of the low friction and no-load. This resulted in the over current protection kicking in because of large current spikes if the sensorless scheme was not postponed to after the current down ramp. Since the sensor less speed estimation is not activated from the start, the result in an estimation overshoot when it is activated.

Even though a satisfying result was obtained in fig. 7.5 a shift to trigger the algorithm by a small θ_L instead of a RPM may be beneficial because this will consistently result in a small discrepancy between the current $I_{q,ref}$ before and after the switch. If the trigger is set to happen at a $\theta_L = 5°$ the discrepancy is very small since $\cos(5°) = 0.996$.

8.2 Current controller

The current controller gains were calculated through the modulus optimum control scheme. Testing different ζ values the result is plotted in fig. 7.6 and the rise times are listed in table 7.8. The ζ value of $\frac{1}{\sqrt{2}}$ is a typical value for this method since it corresponds to critical dampening. However, the results indicate that this ζ gave a significant over-dampening. In addition to this, the climb to the reference is uneven. By decreasing the dampening coefficient ζ, the trend is that the respons become smoother and faster. However non of the values overshoots the reference indicating that they still are underdamped. Based on the risetimes in table 7.8 it seems like the calculated gain is too low considering the positive effect of decreasing ζ.

Looking at the rise times in they are larger than expected. The rise time of the dampening factor $\zeta = \frac{1}{\sqrt{2}}$ should corresponds to $4.7T_{sum}$[10]. In table 7.8 the rise time of the same dampening coefficient is 413.34. The reason for this may be many. $T_{sum,i}$ may be too low because of some delays that may be overlooked. The noise in the measurement is significant and in the sensing circuit analysis, it is found that it is not optimal. Changing the input filter to suit the switching frequency will increase T_{sum} and reduce noise.

Another observation is that the dampening coefficients $\frac{1}{\sqrt{2}}$ and $\frac{1}{\sqrt{4}}$ do not reach the reference and end up with a stationary offset. In addition, there are significant oscillations in the measurements. This may be improved by including a better filter, as mentioned in section 6.4. The constant high-frequency gain of $-15dB$ in addition to a large cut-off frequency of the sensing circuit filter, is not optimal and will affect the current measurements. This will affect the regulation of the current. The general noise may come from the lack of a filter in the input of the sensing circuit.

By looking at fig. 7.6 the response of $\zeta = \frac{1}{\sqrt{32}}$ gives a smooth and relatively fast ramp. The response of $\zeta = \frac{1}{\sqrt{64}}$ is even faster but is not as smooth. It is hard to decide which of these responses have the best performance without testing with load. Table 7.8 shows that the $\zeta = \frac{1}{\sqrt{64}}$ it is almost twice as quick. Testing this on no-load thees results may be significantly different when a load is attached. The current controller should be tuned after the propeller is attached. However the framework is set from this task.

8.3 Speed controller

The speed controller was first tested with a β value of 4. This resulted in a large ripple in the speed as seen in fig. 7.7. By looking at the current reference in fig. 7.8, which is the output of the speed controller, it is seen that the ripple causes the current reference to go from maximum positive current to minimum current. This leads to the speed oscillating with max acceleration around the speed reference. This is ineffective and will wear out the components of the system. It is also observed significant vibration noise running with these speed regulation settings. The symmetrical optimum method can lead to very high gains and is seemingly what happens here. The high gain amplifies the reaction to the speed estimation ripple and sets the current reference to its maximum.

In an attempt to counter this effect, but keeping the symmetrical design of the raise of the phase margin, the β value was increased to 10. This increases the interval of the phase raise, but it also reduces the gain since β is in the numerator in the calculation of $K_{p,n}$ as seen in eq. (5.23). The result is seen in fig. 7.9. The result is similar to the result with $\beta = 4$. In fig. 7.10 the ripple still causes the current reference output of the speed regulator to react and counter the estimated speed when it is too high or low. The difference is in the amplitude of the ripple and the frequency. The frequency of the ripple is almost doubled. However the ripple is still strongly affected by the current reference. This is backed by the same increase in the frequency of the current reference.

A larger increase of the β would increase the rise of the phase margin in an unnecessary large interval. Based on the previous results, it seems like the problem is with the high gain. Therefore the proportional gain is divided by a factor of 10 without changing the integral time constant $T_{i,n}$. The result is seen in fig. 7.11. The ripple in the speed estimation is now almost gone. In fig. 7.12 the speed controller output current reference is acting predictably with a step function followed by a logarithmic decline when the speed is getting close to the speed reference. Even though this solution enables the speed controller to function without significant stress on the system, it is not a good solution. It is seen that the response is significantly over-damped because of the low gain.

The low gain is necessary to avoid the controller reaction to speed ripple. However, a drone needs a fast and dynamic response because of the environment it operates. A better solution would be to implement a filter on the speed estimation. This would filter out the ripple such that the current regulator would not react to the ripple, allowing a higher gain. For this to happen, the frequency of the ripple must be found, and a filter for this frequency be implemented. This would enable a higher gain and the symmetrical optimum method, including a filter, can be tried again.

Chapter 9

Conclusion

This thesis has explored and described the hardware, software, and support tools for a control platform based on the STM32-series from STMicroelectronics. It is found that the STMicroelectronics STM32 series offers high customizability and helpful support tools to develop an in-house control platform for Alva Industries. In addition to this, it offers a highly configurable motor control software library.

For the hardware, it is found that the input filter of the sensing circuit is not sufficient. To be able to run the motor control at a switching frequency of $30kHz$ the input filter had to be removed. Doing this the cut-off frequency of the sensing circuit changed from $17kHz$ to $333kHz$. To obtain better measurements, the input filter capacitance should be changed to $3.5nF$ resulting in a cut-off frequency of $70kHz$ in the current sensing circuit. This will enable a good filtration for a future switching frequency of $60kHz$.

The startup routine was successfully implemented. However, the alignment stage of the startup routine results in the motor swinging around the initial angle like a pendulum and leads to inconsistent results in the startup. The swings dampen by applying an external friction force in this stage, and the startup routine is successful.

The current controller is too slow by choosing a $\zeta = \frac{1}{\sqrt{2}}$ resulting in an overdamped response with a rise time corresponding to $418T_{sum}$. The dampening factor was lowered until the response was close to the expected ratio of $4.7T_{sum}$. This resulted in a ζ of $\frac{1}{\sqrt{64}}$ and resulted in a rise time of $12.85T_{sum}$.

The speed controller calculations resulted in large ripples in the speed, leading to the output current reference oscillating from negative maximum to positive maximum current. This may be solved by implementing a filter on the speed estimations. For the current controller to function, the β was increased to 10 in the symmetrical optimum method. In addition, the proportional gain was reduced by a factor of 10. This resulted in a functional speed controller but should be improved by adding a filter.

Chapter 10

Further Works

The STMicroelectronics STM32-series platform showed promising potential. Suggestions for further development is listed below.

- Set up the control platform for Alva Industries sl-PMSM.
- Switch the control board to STM32F746ZG to increase the switching frequency, and test at $60kHz$.
- Implement the suggested input filter for the current sensing and a filter for the speed estimation.
- Test the system with load and perform efficiency test.
- Implement sensorless algorithms from the analysis done by Nerbøberg [9].
- Develop power board and control board based around the STM32 microprocessor.

Bibliography

[1] VESC Project. URL https://vesc-project.com/Hardware.

[2] Laszlo Balogh. Fundamentals of MOSFET and IGBT Gate Driver Circuits. page 48, 2017.

[3] Martin Bates. Chapter 12 - More PIC Microcontrollers. In Martin Bates, editor, *PIC Microcontrollers (Third Edition)*, pages 261–284. Newnes, Oxford, January 2011. ISBN 978-0-08-096911-4. doi: 10.1016/B978-0-08-096911-4.10012-6. URL https://www.sciencedirect.com/science/article/pii/B9780080969114100126.

[4] ETechnoG. Enhancement VS Depletion MOSFET Advantages, Applications. URL https://www.etechnog.com/2021/02/difference-depletion-enhancement-mosfet.html.

[5] Texas Instrument. Current Sensing With <1-µs Settling for 1-, 2-, and 3-Shunt FOC Inverter Reference Design. page 47, 2017. URL https://www.ti.com/lit/ug/tiducy7/tiducy7.pdf?ts=1654506077866&ref_url=https%253A%252F%252Fwww.google.com%252F.

[6] Isak Nordeng Jensen. Control of an Ironless Permanent Magnet Synchronous Machine. Technical report, NTNU, December 2020.

[7] Bhuvana Madhaiyan and Sampath Palaniappan. Introduction to Gate Drivers for Power Electronics, October 2018. URL https://talema.com/gate-driver-introduction/.

[8] Ned Mohan, Tore M. Undeland, and William P. Robbins. *Power electronics: converters, applications, and design*. John Wiley & Sons, Hoboken, NJ, 3rd ed edition, 2003. ISBN 978-0-471-22693-2.

[9] Pål Nerbøberg. Analysis of position estimation schemes for position sensorless indirect torque control of an IPMSM. Technical report, NTNU, December 2021.

[10] Roy Nilsen. *TET4120 - ELECTRIC DRIVES*. 2018.

[11] Roy Nilsen. *ET 8205 - MODELLING, IDENTIFICATION AND CONTROL OF ELECTRICAL MACHINES*. 2019.

[12] Grand Wiev Research. Commercial Drone Market Size & Share Report, 2021-2028, 2020. URL https://www.grandviewresearch.com/industry-analysis/global-commercial-drones-market.

[13] STMicroelectronics. Datasheet X-NUCLEO-IHM08M1, . URL https://www.st.com/resource/en/data_brief/x-nucleo-ihm08m1.pdf.

[14] STMicroelectronics. Datasheet STM32G431, . URL https://www.st.com/content/ccc/resource/sales_and_marketing/presentation/product_presentation/group0/3e/36/98/32/bf/57/4b/3e/Microcontrollers_STM32G4_series_product_overview/files/Microcontrollers_STM32G4_series_product_overview.pdf/jcr:content/translations/en.Microcontrollers_STM32G4_series_product_overview.pdf.

[15] STMicroelectronics. X-CUBE-MCSDK - STM32 Motor Control Software Development Kit (MCSDK) - STMicroelectronics, . URL https://www.st.com/en/embedded-software/x-cube-mcsdk.html.

[16] STMicroelectronics. User manual STM32 motor control SDK, 2018. URL https://hobbydocbox.com/Radio/110731709-Um2392-stm32-motor-control-sdk-user-manual-introduction.html.

[17] Thingap. Data Sheet TG2341 DC BRUSHLESS MOTOR. URL https://www.thingap.com/wp-content/themes/ndic/pdf/TG2341.pdf.

[18] Circuits today. PMOS and NMOS-Comparison of P and N Channel Mosfets, July 2010. URL https://www.circuitstoday.com/pmos-vs-nmos.

[19] Zihui Wang, Kaiyuan Lu, and Frede Blaabjerg. A Simple Startup Strategy Based on Current Regulation for Back-EMF-Based Sensorless Control of PMSM. *IEEE Transactions on Power Electronics*, 27(8):3817–3825, August 2012. ISSN 1941-0107. doi: 10.1109/TPEL.2012.2186464. Conference Name: IEEE Transactions on Power Electronics.

[20] Hans Erik Weiby. For første gang i Norge har en savnet person blitt funnet av drone, April 2020. URL https://www.nrk.no/sorlandet/politiet-fant-savnet-kvinne-i-mandal-ved-hjelp-av-drone-1.14984110.

[21] Yang Zhen. Current Sensing Circuit Concepts and Fundamentals. page 12, 2011. URL https://ww1.microchip.com/downloads/en/AppNotes/01332B.pdf.

Appendix A

Important structs and enums in software

Table A.1: MCI_UserCommands_t: Enum showing the states dependent on user commands[15].

Type	Description
MCI_NOCOMMANDSYET	No command has been set by user
MCI_EXECSPEEDRAMP	Speedramp comand coming from user
MCI_EXECTORQRAMP	Torqueramp comand coming from user
MCI_SETCURRENTREFERENCES	Current reference comand coming from user

Table A.2: MCI_CommandState_t: Enum showing the state of the commands[15].

Type	Description
MCI_BUFFER_EMPTY	No buffered command has been called
MCI_COMMAND_NOT_ALREADY _EXECUTED	The buffered command condition has not occured yet
MCI_COMMAND_EXECUTED _SUCCESFULLY	Buffered command has been executed successfully
MCI_COMMAND_EXECUTED _UNSUCCESFULLY	Buffered command has been executed unsuccesfully

Table A.3: State_t [15]

State	Number	Type	Description	Next state
ICLWAIT	12	Persistant	Inrush current limiter is active	IDLE
IDLE	0	Persistant	Idle	IDLE_START, IDLE_ALIGNMENT
IDLE_ALIGNMENT	1	Pass-through	A pass through state that is used for encoder alignment etc.	ALIGN_CHARGE_BOOT_CAP ALIGN_OFFSET_CALIB, ANY_STOP
ALIGN_CHARGE _BOOT_CAP	13	Persistant	Gate driver boot capacitators is charging	ALIGN_OFFSET_CALIB, ANY_STOP
ALIGN_OFFSET _CALIB	14	Persistant	Offset of motor currents measurements is calibarated	ALIGN_CLEAR, ANY_STOP
ALIGN_CLEAR	15	Pass-through	Object is cleared and set for startup	ALIGNMENT, ANY_STOP
ALIGNMENT	2	Persistant	encoder are properly proporley aligned to set mechanical angle	ANY_STOP
IDLE_START	3	Pass-through	containg the code to be executed only once after start motor command	CHARGE_BOOT_CAP, OFFSET_CALIB, ANY_STOP
CHARGE_BOOT _CAP	16	Persistant	Gate driver boot capacitators is charging	OFFSET_CALIB, ANY_STOP
OFFSET _CALIB	17	Persistant	Offset of motor currents measurements is calibarated	CLEAR, ANY_STOP
CLEAR	18	Pass-through	Object is cleared and set for startup	START, ANY_STOP
START	4	Persistant	the motor start-up is intended to be executed	SWITCH OVER, RUN
START_RUN	5	Pass-through	executed only once between START and RUN	
SWITCH_OVER	19	TBD	TBD	TBD
RUN	6	Persistant	Motor is running	ANY_STOP
ANY_STOP	7	Pass-through	Code to be executed only once between any state and STOP	
STOP	8	Persistant		STOP_IDLE
STOP_IDLE	9	Pass-through	Code to be executed only once between STOP and IDLE	IDLE
FAULT_NOW	10	Persistant	A fault is present	FAULT_OVER
FAULT_OVER	11	Persistant	State after a fault has disappeared	STOP_IDLE
WAIT_STOP _MOTOR	20			

Table A.4: STM_Handle_t [15]

Type	Variable name	Description
enum	State_t	Variable containing the state machine current state
uint16_t	hFaultNow	Bit fields variable containing faults currently present
uint16_t	hFaultOccured	Bit fields variable containing fault history

Table A.5: MCI_Handle_t[15]

Type	Variable name	Description
STM_Handle_t *	pSTM	pointer to state machine object
SpeednTorqCtrl_Handle_t *	pSTC	pointer to speed and torque controller object
pFOCVars_t *	pFOCVars	pointer to FOC variables used by the MCI
MCI_UserCommands_t	lastCommand	Last command coming from user
int16_t	hFinalSpeed	Final speed of last executed speed ramp command
int16_t	hFinalTorque	Final torque of last executed torque ramp
qd_t	Iqdref	The q and d reference current of last current reference command
uint16_t	hDurationms	Duration in ms of last executed torque ramp command
MCI_CommandState_t	CommandState	Status of the buffered command
STC_Modality_t	LastModalitySetByUser	Last modality set by user (torque mode/speed mode)

Table A.6: MCT_Handle_t[15]

Type	Variable name	Description
PID_Handle_t *	pPIDSpeed	pointer to a speed controller object
PID_Handle_t *	pPIDIq	pointer to a q-axis current controller object
PID_Handle_t *	pPIDId	pointer to a d-axis current controller object
PID_Handle_t *	pPIDFluxWeakening	pointer to flux weakening controller object
PWMC_Handle_t *	pPWMnCurrFdbk	handles the data of an instance of the PWM & Current Feedback component
RevUpCtrl_Handle_t *	pRevupCtrl	Denne må undersøkes nærmere, men mulig det er FOC control til startup, evt akselerasjon
SpeednPosFdbk_Handle_t *	pSpeedSensorMain	holds data to decode and store the data from position and speed sesnor
SpeednPosFdbk_Handle_t *	pSpeedSensorAux	holds data to decode and store the data from auxilliary position and speed sesnor
VirtualSpeedSensor_Handle_t *	pSpeedSensorVirtual	holds data to decode and store the data from virtual position and speed sesnor
SpeednTorqCtrl_Handle_t *	pSpeednTorqueCtrl	Holds all data for speed control and torque control
STM_Handle_t *	pStateMachine	Provides the state of the machine
NTC_Handle_t *	pTemperatureSensor	holds data to decode and store the data from temperature sensor
BusVoltageSensor_Handle_t *	pBusVoltageSensor	holds data to decode and store the data from bus voltage sensor
DOUT_handle_t *	pBrakeDigitalOutput	Digital output handler
DOUT_handle_t *	pNTCRelay	Digital output handler
MotorPowMeas_Handle_t *	pMPM	Handles motor power measurment
FW_Handle_t *	pFW	fluxweakening control component handle
FF_Handle_t *	pFF	Feed forward component
PosCtrl_Handle_t *	pPosCtrl	position control handle
SCC_Handle_t *	pSCC	Holds max values, estim. and meas. of the motor and control parameters
OTT_Handle_t *	pOTT	One touch tuning parameters

Table A.7: PID_Handle_t [15]

Type	Variable name	Descreption
int16_t	hDefKpGain	Default proportional gain
int16_t	hDefKiGain	Default integral gain
int16_t	hKpGain	Proportional gain used by PID component
int16_t	hKiGain	Integral gain used by PID component
int32_t	wIntegralTerm	Integral term
int32_t	wUpperIntegralLimit	Upper limit for saturation of the integral term
int32_t	wLowerIntegralLimit	Lower limit for saturation of the integral term
int16_t	hUpperOutputLimit	Upper limit for saturation of the output
int16_t	hLowerOutputLimit	Lower limit for saturation of the output
uint16_t	hKpDivisor	A divisor to obtain fractional values of the proportional gain
uint16_t	hKiDivisor	A divisor to obtain fractional values of the integral gain
uint16_t	hKpDivisorPOW2	hKpDivisor expressed as power of two
uint16_t	hKiDivisorPOW2	hKiDivisor expressed as power of two
int16_t	hDefKdGain	Default derivative gain
int16_t	hKdGain	Derivative gain used by PID component
uint16_t	hKdDivisor	A divisor to obtain fractional values of the derivative gain
uint16_t	hKdDivisorPOW2	hKdDivisor expressed as power of two
int32_t	wPrevProcessVarError	previous process variable used by the derivative part of the PID component

Table A.8: Parameters availible for high frequency logging.[15]

Parameter	Unit	Description
I_A	s16A	Measured phase current in phase A
I_B	s16A	Measured phase current in phase B
I_ALPHA_MEAS	s16A	Alpha component of the Clarke transformed measured phase current
I_BETA_MEAS	s16A	Beta component of the Clarke transformed measured phase current
I_Q_MEAS	s16A	q component of the Clarke and park transformed measured phase current
I_D_MEAS	s16A	d component of the Clarke transformed measured phase current
I_Q_REF	s16A	Reference of the q-component of the stator current
I_D_REF	s16A	Reference of the d-component of the stator current
V_Q	s16V	q component of the voltage
V_D	s16V	d component of the voltage
V_ALPHA	s16V	α component of the voltage
V_BETA	s16V	β component of the voltage
ENCODER_EL_ANGLE	s16degree	Encoder measured electrical angle
ENCODER_SPEED	s16speed	Encoder measured mechanical speed
STOPLL_EL_ANGLE	s16degree	Estimated electrical angle (PLL)
STOPLL_ROT_SPEED	s16speed	Estimated mechanical speed (PLL)
STOPLL_BEMF_ALPHA	s16V	Observed α component of the estimated back emf (PLL)
STOPLL_BEMF_BETA	s16V	Observed β component of the estimated back emf (PLL)
STOCORDIC_EL_ANGLE	s16degree	Observed electrical angle (CORDIC)
STOCORDIC_ROT_SPEED	s16speed	Observed mechanical speed (CORDIC)
STOCORDIC_BEMF_ALPHA	s16V	Observed α component of the estimated back emf (CORDIC)
STOCORDIC_BEMF_BETA	s16V	Observed β component of the estimated back emf (CORDIC)
HALL_EL_ANGLE	s16degree	Hall sensor measured electrical angle
HALL_SPEED	s16speed	Hall sensor measured mechanical speed