University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Automation and Signal Processing	Spring semester, 2015 Restricted access
Writer: **Stian Lund**	[illegible] (Writer's signature)
Faculty supervisor: **Morten Tengesdal** External supervisor(s): **Vidar Hølland**	
Thesis title: Implementation of a Motor Controller	
Credits (ECTS):	
Key words: Automation, Signal, Processing, Regulation, Control, Motor, Controller, BLDC,	Pages: ………105……… + enclosure: ……1x .7z zip-file…… Stavanger, ……15.06/2015….. Date/year

Abstract

This thesis is written for E Plug, a company that specialises in down-hole operations in the oil industry.

Brushless DC (BLDC) motors are being used almost everywhere in todays market. The BLDC motor can be found in vacuum cleaners, dryers, air conditioners, electrical bikes, medical equipment, aerospace technology etc. It is now increasingly being used in the oil industry. The BLDC motor is known to be efficient, silent, compact and has a long operational lifetime and reduced maintenance time compared to other DC motors. Because of the lack of brushes, the motor is optimal to use in the oil industry as it won't generate any dangerous sparks which can be a problem with brushed motors.

The challenge with a brushless motor is that it requires an efficient driving code implemented in a microcontroller. The rotor position is unknown and the motor need some sort of logic to determine the position, so that it knows when to give the rotor a "push". The brushed motors are easier to control as the brushes constantly provides feeback on where the rotor is.

In this thesis three different driving codes are implemented in a microcontroller, tested and analysed. The first method is sensorless control, which is based on back-EMF detecting. This method provides an unstable and unpredictable start up time for the motor, as back-EMF sensing isn't available on low speed operations. The second methods uses Hall-effect sensors which gives feedback on where the rotor is. This method provides a stable response and is very effective. The third method uses an optical encoder with a more complex code that takes use of the benefits of shunt resistors. By sensing the phase currents and using the measurements to control the stators magnetic field, the motor can be regulated to operate at it's maximum torque per amp. This method shows to be clearly the most precise and effective way of controlling the motor.

A use of resolver for detecting rotor position was planned to be implemented. Because of the

hardware needed for this control wasn't available, it had to be discarded. As most of the time working on the thesis went to implementing the different methods, there wasn't so much time as hoped to analyse the results. One of the other goals was to implement the different methods on a second microcontroller, which is based on floating-point unit instead of fixed-point unit. As this task became too extensive, this goal was discarded as well.

Contents

Acknowledgements

I want to give a special thanks to my supervisor, Vidar Hølland, at E Plug for assisting me during the thesis. His support has helped me reach the main goals of this thesis. He has really dedicated himself to be able to solve problems with me whenever I was stuck. I also want to thank him for getting me out of the office to get some solid food, during some of the late nights.

The second person I want to thank is my supervisor, Morten Tengesdal, at the University of Stavanger. Whenever I had some questions that I needed answers to, I could always count on a fast reply from him on email, or a good chat at his office. He has also been a great asset for helping me set up this report.

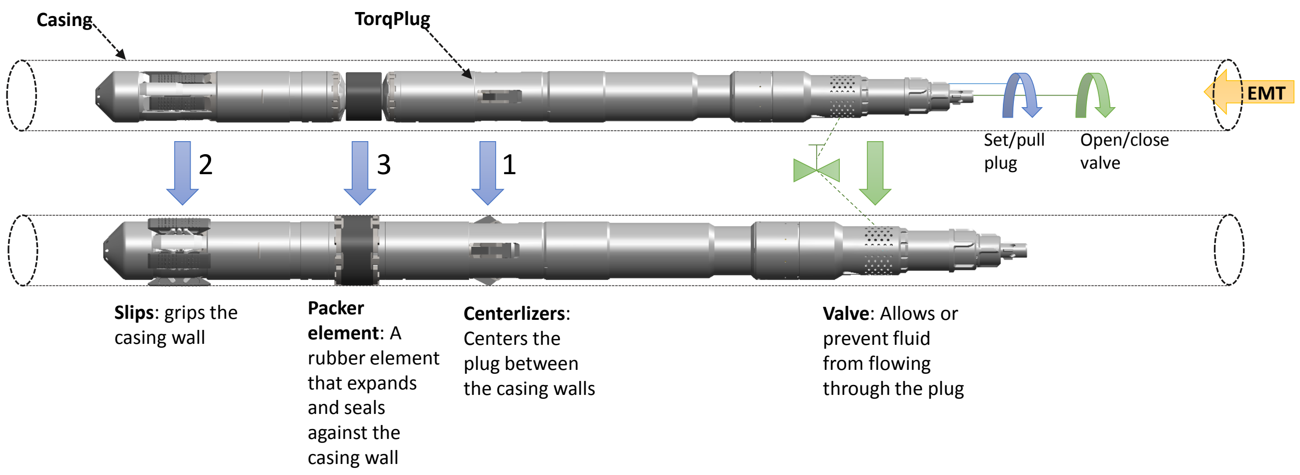
I would also like to thank previous E Plug employee Kevin Johansen for helping me set up the testing rig used in the experiments part of the thesis. The experiments wouldn't have been so interesting without him.

Finally, a big thanks goes to all my colleagues at E Plug for supporting me during the thesis. They have all been really supporting and interested in my work on the thesis.

Chapter 1

Introduction

The subject of this thesis is to get a motor controller software for a brushless DC motor up and running and analyse different driving methods that can later be used for the *EMT*. The EMT (Electrical Manipulation Tool) is a tool for setting and retrieving a bridge plug in oil wells, patented by E Plug. The EMT is equipped with two independent manipulators which is controlled by DC motors. One of the motors is used as a linear actuator, while the other one is used as a rotary actuator. The rotary actuator produces torque that is used for opening/closing a valve and compressing/decompressing a packer element to set/release the plug in the oil well. The linear manipulator moves the rotary manipulator in different positions, depending on the current operation, if it is to set/release the plug or to close/open the valve.



Casing
TorqPlug
EMT
Set/pull plug
Open/close valve
2
3
1
Slips: grips the casing wall
Packer element: A rubber element that expands and seals against the casing wall
Centerlizers: Centers the plug between the casing walls
Valve: Allows or prevent fluid from flowing through the plug

Figure 1.1: The TorqPlug - a bridge plug by E Plug

The EMT attached to the TorqPlug is shown in figure 1.2 below:

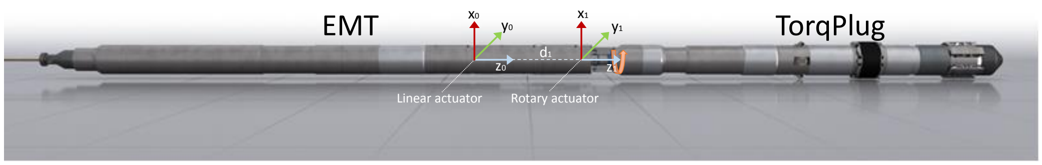


Figure 1.2: The EMT attached to the TorqPlug

1.1 Background

The current solution of controlling the DC motors is supplied by an external company. The source code of the motor controller is encrypted, which means that if any changes are to be made, the supplier has to do them. This solution is time consuming and has higher costs than necessary.

E Plug is now currently developing their own motor controller and is therefore in need for a controller code that suits their requirements.

Problem Formulation

The object of this thesis is to find an appropriate and effective method for controlling the DC motors. The key components of requirements for this motor controller is that it is:

- Robust
- Stable
- Precise

A robust code means that the code should take account for any possible errors or ambient noise that can occur during operations. A stable code means that the motors behaviour should be predictable and well regulated during unstable conditions. A precise code means that the measurements and actions should be fast and preferably lossless.

It is important that the position/speed control of the motor is precise because of the limited operational area of the linear actuator in the EMT. This basically means that if the linear actuator moves too far with a high enough torque, it can result in a broken gearbox or worse.

A stable code that takes account for high loads during operations by regulating speed and torque is also important. As the rotary actuator rotates, it will eventually experience a counterforce. The rotary actuators task is to opposite this force until a desired torque is achieved.

Another challenge with the EMT is the ambient temperature and motor temperature during operations in deep oil wells, where the EMT has a requirement of handling temperatures up to 170°C. As the equipment provided does not handle these temperatures, and the lack of temperature sensors, this problem is not accounted for in this thesis.

Literature Survey

A motor kit from Texas Instruments is supplied for this thesis which contains some source code files for driving a motor and theory of operation [1] [2]. Texas Instruments has supplied some modules[3], which basically is code functions that is pre-made for the user, so that the user doesn't have to "reinvent the wheel". These modules are short snippets that for example ramps up an integer and outputting variables to the DAC (which makes variables readable for the oscilloscope). Texas Instruments has also provided with library files, or header files, which configures the code to work with all the necessary peripherals.

Literature for the microcontroller is taken from "Frå transistor til datamaskin" by Morten Tengesdal [4] and "Designing Embedded Hardware" by John Catsoulis [5]. The literature for the BLDC motor, excluding the information supplied by Texas Instruments, is gathered throughout the thesis and is listed in the bibliography at the end of this report.

1.2 Objectives

The main objectives of this Master's project are

1. Implementing and analysing sensorless control of the BLDC motor.
2. Implementing and analysing sensored controls of the BLDC motor.
3. Evaluate the findings and provide a detailed conclusion of the results.

The sub objectives which is equally important of this Master's project are

1. Get a very good understanding of the theory behind the motor controller.
2. Apply relevant theory and methods in a very convincing fashion.
3. Be able to use existing work done in the field in a way that it merges with own contributions.
4. Be critical to existing methods.
5. Propose an optimal method based on the thesis requirements.

1.3 Approach

The different methods will first be implemented in the microcontroller. The code will have to be understandable for other users and be well commented.

The codes will be tested on the motor and most likely tuned to make it more effective.

The methods will be analysed by looking at their step responses. PI/PID-controller values will be calculated from the open loop step responses.

The responses in closed loop driving will be analysed and added to results.

1.4 Structure of the Report

The rest of the report is organized as follows. Chapter 2 explains the theory needed to understand the work that is carried out in this thesis.

Chapter 3 goes into the detail of the system. Code snippets from the implementation of the different methods is explained.

Chapter 4 gives a description of how the experiments are carried out, and what equipment is used.

Chapter 5 explains the results from the experiments including discussion and a conclusion.

Remark: Please note that any figures without citations is made by the author of this thesis.

Chapter 2

Theory

In this chapter, the structure and function of the BLDC motor will be explained as well as the different methods of controlling a BLDC motor will be reviewed. The theory is gathered from [6] and [7]

Firstly, a very brief introduction to electromagnetic induction will be given to give a better understanding of how the motor works.

2.1 Electromagnetic theory

By Ampere's law, the magnetic field is produced by electric currents as shown in figure 2.1. The intensity of the magnetic field is proportional to the current. The greater the current is, the greater the magnetic field is.

$$\oint \mathbf{B} \cdot ds = \mathbf{B} \oint ds = \frac{\mu_0 I}{2\pi r} 2\pi r = \mu_0 I \tag{2.1}$$

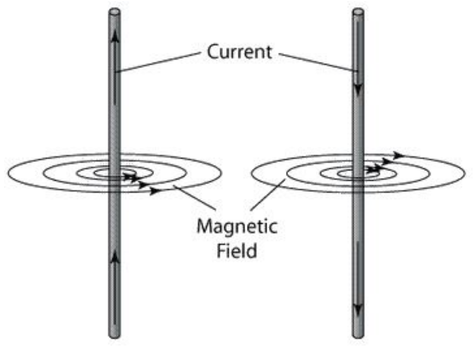


Figure 2.1: The direction of the magnetic field is determined by the right hand rule [8], see figure 2.5

The magnetic field in the wire above, figure 2.1, can be described from equation 2.1:

$$\oint \mathbf{B} \cdot ds = \mathbf{B} \oint ds = \mathbf{B}(2\pi) = \mu_0 I \tag{2.2}$$

$$\mathbf{B} = \frac{\mu_0 I}{2\pi r} \tag{2.3}$$

If a wire is wound around a coil, as shown in figure 2.2 below, the magnetic field can be described as:

$$\oint \mathbf{B} \cdot ds = \mathbf{B} \int_{path1} ds = \mathbf{B}l = \mu_0 N I \tag{2.4}$$

$$\mathbf{B} = \frac{1}{l} \mu_0 N I = \mu_0 n I \tag{2.5}$$

Where $\mu = k\mu_0$ $\mu_0 = 4\pi \times 10^{-7} T/amp$

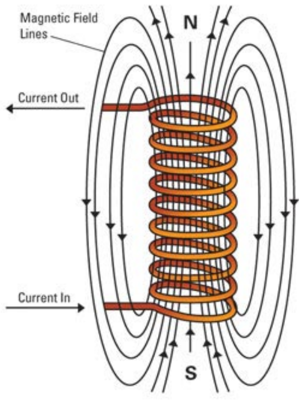


Figure 2.2: Current-carrying coil [9]

Just like the gravitation and electric field, the magnetic field has a flux, Φ. The magnetic flux describes the number of magnetic field lines that passes through a surface. The flux through any surface is described as:

$$\Phi_B = \int \mathbf{B} \cdot d\mathbf{A} \tag{2.6}$$

Where · is the dot product. For a flat surface, the flux can be described as

$$\Phi_B = \int \mathbf{BA} cos\theta \tag{2.7}$$

If the magnetic field is perpendicular to the area, the formula can be simplified.

$$\Phi_B = \int \mathbf{B} d\mathbf{A} \tag{2.8}$$

$$= \mathbf{B} \int d\mathbf{A} \tag{2.9}$$

$$\Phi_B = \mathbf{BA} \tag{2.10}$$

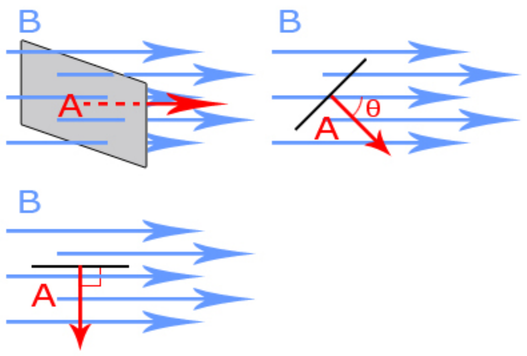


Figure 2.3: In the top sketches, the flux $\Phi = \mathbf{BA}cos\theta$. In the bottom sketch, the flux $\Phi = 0$

For a coil of wires, it is normal to describe the *flux linkage*. The flux linkage describes the flux over several loops of wires. In figure 2.2 the flux linkage equals to

$$\lambda = N\mathbf{BA} \tag{2.11}$$

Where $N = 10$.

Faraday's law of induction summarizes the ways voltage can be generated. When the flux is changed over time in a magnetic field, an electric field is set up around the area. For example, if the flux is changed through a coil of wire, there is induced a voltage. This generated voltage is called *emf*, or *electromotive force*. Mathematically:

$$V_{gen} = -N\frac{\Delta(\mathbf{BA})}{\Delta t} \tag{2.12}$$
$$= -N\frac{\Delta(\Phi)}{\Delta t} \tag{2.13}$$

Where N is the number of turns in wire and t is time.

The back-EMF generated by a rotating motor is described as:

$$V_{bemf} = NlrB\omega \tag{2.14}$$

Where l is the length of rotor, r is the radius of rotor and ω is the motor's angular velocity

Equation 2.12 tells us that the induced emf is increased when:

- The number of turns in wire is increased

- The time interval of change is decreased
- The size of the area in the magnetic field is increased
- The size of the magnetic field is increased

The minus sign denotes *Lenz's law*: the polarity of the induced emf is such that it produces a current whose magnetic field opposes the change which produces it.

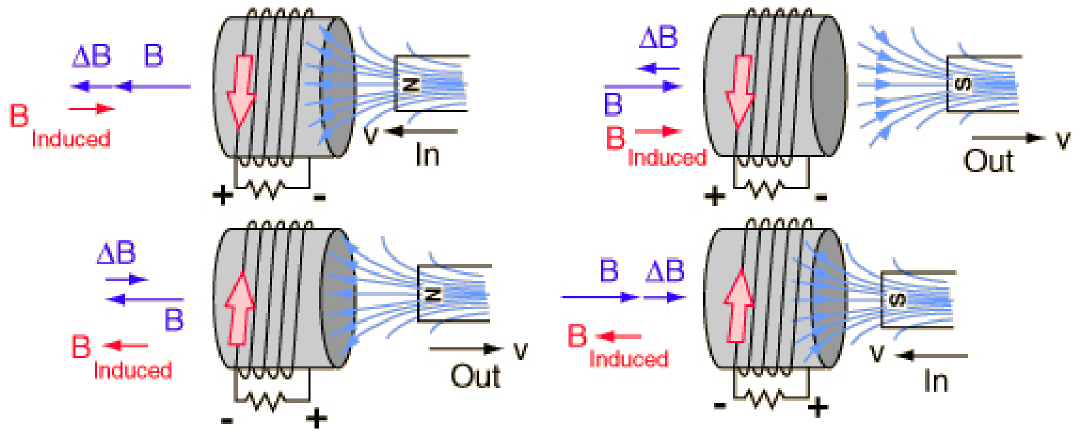


Figure 2.4: Lenz's law [10]

The polarity of an electromagnetic coil can be determined by using a right hand rule. Wrap your fingers around the coil in the direction of the current flow, and the thumb will point towards the north pole of the coil.

2.2 Electric motors

Electric motors are similar to generators. Instead of using motion to produce current, electric motors use current to produce motion. An electric motor consists of a stator (stationary part), and a rotor (moving part). See figure 2.6 below.

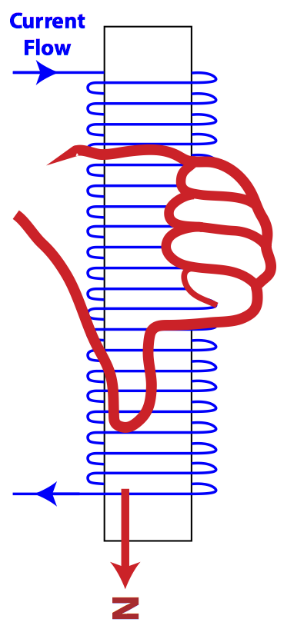


Figure 2.5: The right hand rule for determining the direction of current or the pole of the coil[11]

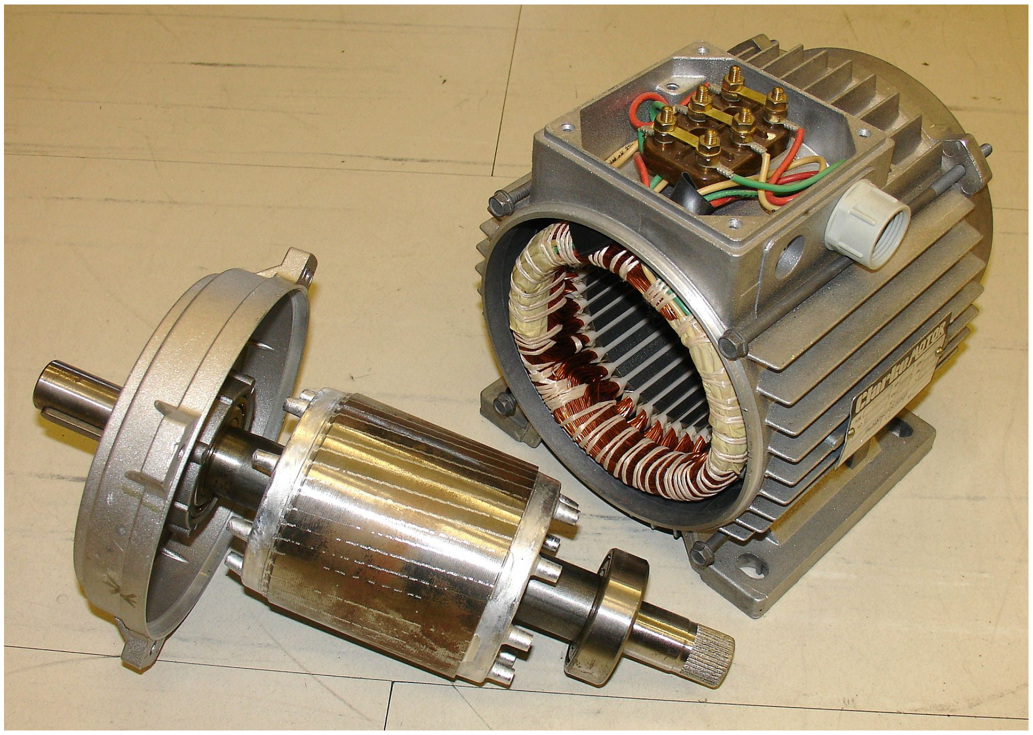


Figure 2.6: AC-motor: rotor(left) and stator (right)[12]

The permanent magnet synchronous motor (PMSM) is equipped with coils and permanent magnets. The coils are placed on the the stator and the permanent magnets are placed on the rotor. By synchronous it means that the magnetic field created by the stator rotates at the same frequency as the magnetic field generated by the rotor. The motor used in this thesis is a brushless DC motor (BLDC) which is a type of PMSM motor. The conventional PMSM motor has distributed stator windings while the BLDC motor has concentrated windings. This results among other things in a sinusoidal back-EMF for the PMSM motor and a trapezoidal back-EMF for the BLDC. The BLDC motor is illustrated in figure 2.7 below.

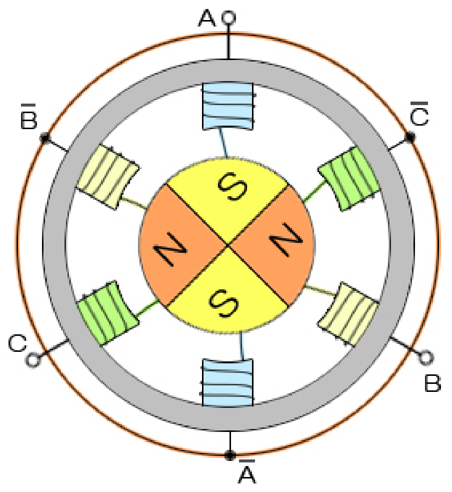


Figure 2.7: BLDC motor with three polepairs and four permanent magnets

The coils are used as electromagnets. By entering a current through the coil, the coil will act as either a north pole, or a south pole, depending on the direction of the current. By letting a current flow through the coils in a certain sequence of the coils (and direction), we can simply rotate the permanent magnets at the rotor. This is called commutating.

What actually happens when the windings are fed with a current, is that a magnetic force (flux) is generated around it. The flux is proportional to the current fed through the windings. The flux from the magnetic field generated by the rotor on the other hand, is constant. When these two forces act upon each other, a certain amount of torque is generated. The amount of torque is depends on the angle of the magnetic fields. When the stator flux and the rotor flux are perpendicular to each other, the force will be maximal. In section 2.4.3 in chapter 2 this will be more discussed, and a technique (Field Oriented Control) using the knowledge of the rotor position and controlling the stator current will be described.

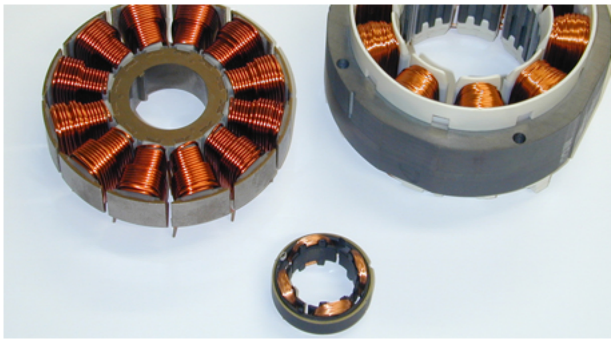


Figure 2.8: Coil windings in stator[13]

2.3 Digital Control

In the AC-motor, the phases are fed with sine waves that are shifted 120° out of phase of each other. This creates a rotating magnetic field around the stator, where the current in the coils are gradually increased and decreased. This means that the rotor will start rotating along the magnetic field in a speed which is determined by the frequency.

To commutate the rotor in the DC motor, we need some sort of on/off switches to choose which of the phases the current is flowing through, and how much current will be fed. By switching on and off these switches at certain times, we can generate current, or voltage, that look like sine waves. The amplitude and harmonic contents of the AC waveform are controlled by regulating the duty cycle of the switches. An inverter is used to convert the DC voltage into AC voltage. In this thesis two different methods will be used. The first method uses conventional PWM control, where two of the three phases are fed with a PWM signal with a constant duty cycle. The second method uses space vector modulation for the PWM signal (SVPWM). All of the three phases are fed with the SVPWM signal, where duty cycle is modulated to gradually increase or decrease with respect to the rotor position.

If you remember the retro helicopter PC-game, where you have to press the space button to gain altitude and let go to of the space button to drop altitude, we can compare it to the PWM control. Let's imagine that you control the PWM frequency and duty cycle by pressing the space button. We will then achieve a desired flight path. See figure 2.9 below.

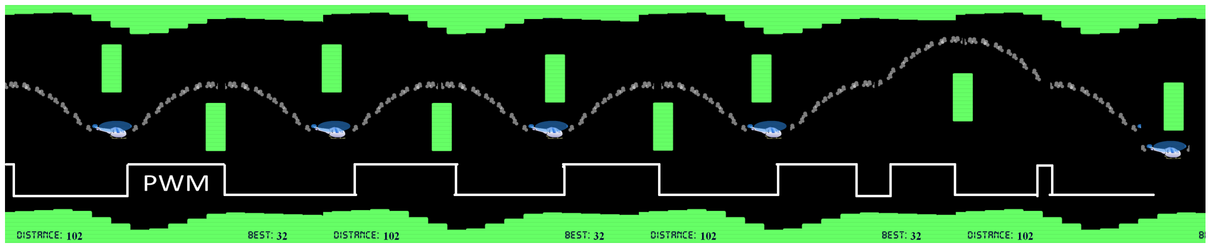


Figure 2.9: Helicopter game: the white lines describes the space button actions.

Imagine that the flight path 2.9 above is the output phase voltage. Using Conventional PWM control or by using SVPWM, we will later see that the output voltage will be more or less formed like a sine-wave.

2.3.1 Conventional PWM control

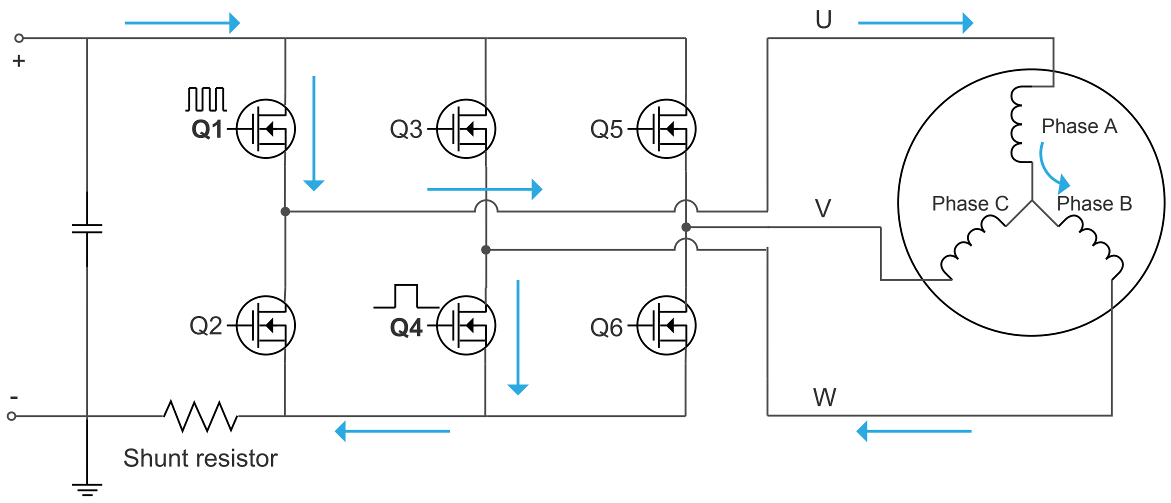


Figure 2.10: 3-Phase inverter: Current flows to motor windings from phase A->B, de-energized phase = C

The switches on figure 4.3, labeled Q1-Q6, are a type of transistors called MOSFETs. The odd numbered switches are controlled with PWM signals, while the even numbered switches are either turned off or fully on.

The bottom transistors allows the current to flow to ground, depending on if they are high or low, which means that we can control which of the phases the current will flow through. If the commutation state is 0, see table 2.1, Q1 is fed with a PWM signal and Q4 is high. This means that the current flows to motor windings from phase A to B, while C is de-energized and is available for back-EMF reading as shown in figure 4.3. The switch pattern in the figure is described in 2.1 below.

State	Q1	Q2	Q3	Q4	Q5	Q6	Ia	Ib	Ic
0	PWM	low	low	HIGH	low	low	+	-	OFF
1	PWM	low	low	low	low	HIGH	+	OFF	-
2	low	low	PWM	low	low	HIGH	OFF	+	-
3	low	HIGH	PWM	low	low	low	-	+	OFF
4	low	HIGH	low	low	PWM	low	-	OFF	+
5	low	low	low	HIGH	PWM	low	OFF	-	+

Table 2.1: Commutation table for counter-clockwise(CCW) rotation

The inputs of the transistors from table 2.1 above is shown in 2.11 below. By feeding the motor with PWM signals, we are actually choosing how much of the input voltage we apply to the phases. A duty cycle of 50% will for example apply half of the input voltage to the phase when the PWM is active.

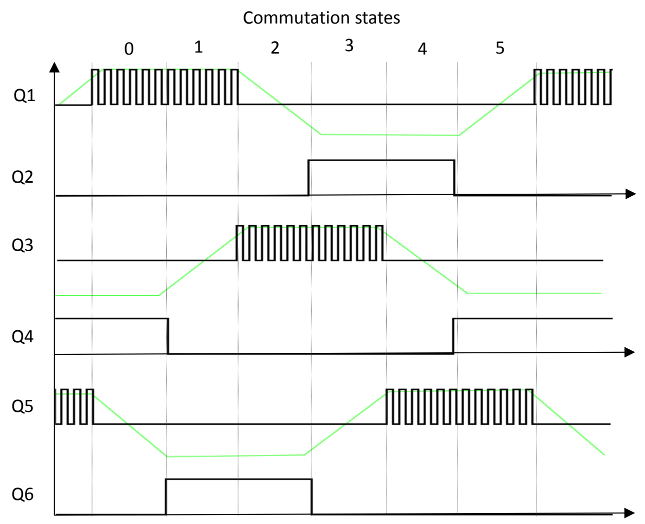


Figure 2.11: MOSFET inputs

The output of the transistors is illustrated in 2.12 below. The back-EMF has a trapezoidal shape due to the winding distribution makes the permanent magnets produce a certain air gap flux density distribution. The tiny spikes in the currents between the commutation states are caused by the stator magnetic field jump. A method proposed in **??** shows that the stator magnetic field switching causes torque ripple for BLDC motors.

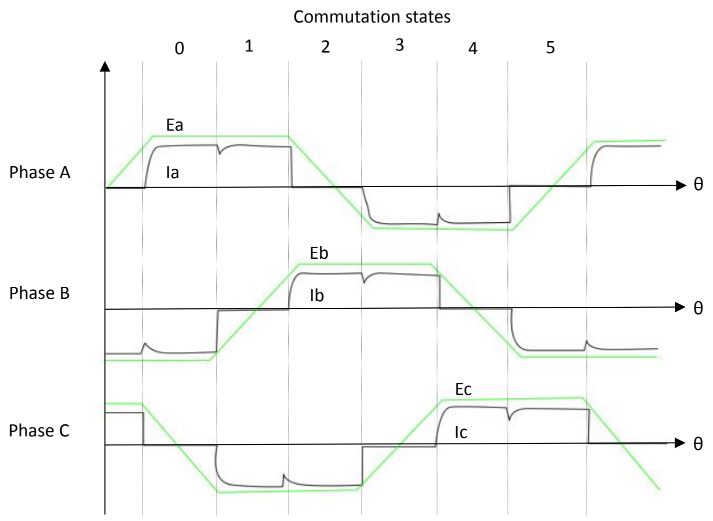


Figure 2.12: Phase currents and back-EMF waveforms

The figure 2.13 below shows how the rotor rotates along the stator magnetic field generated by the coils. The permanent magnets in the rotor will try to chase their opposite poles.

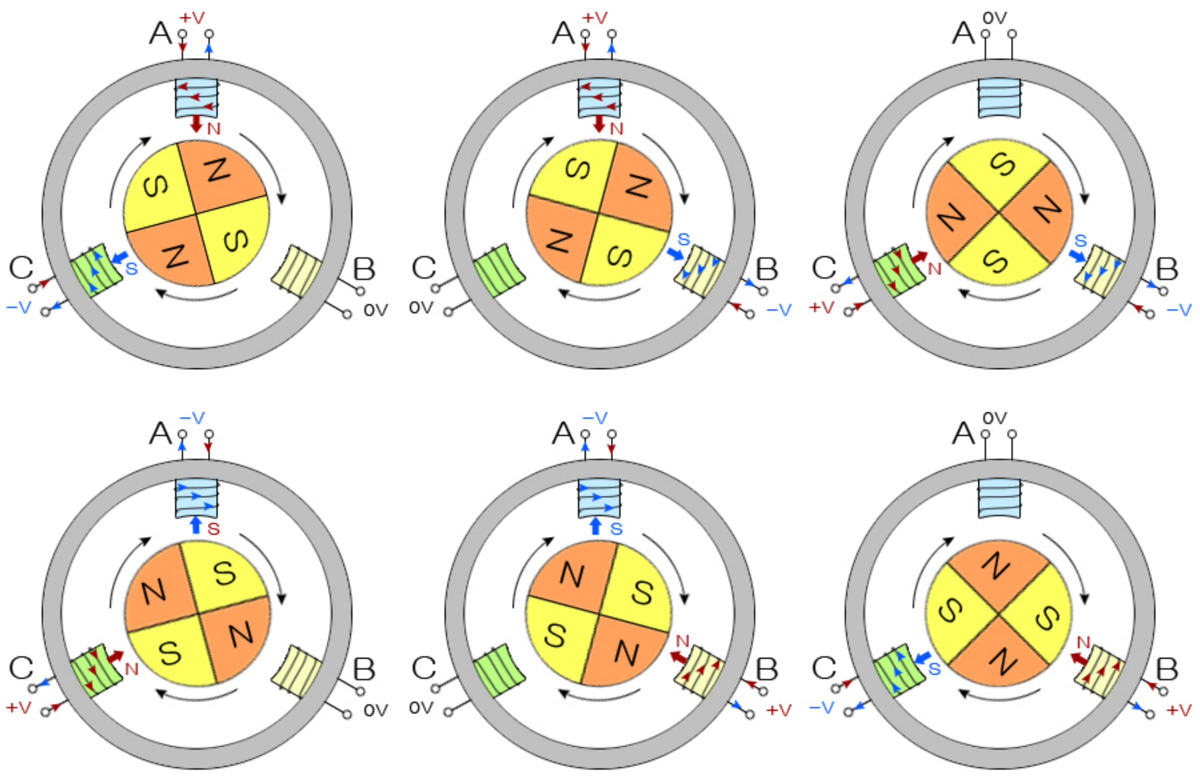


Figure 2.13: Commutating clockwise, states: 1->0->5->4->3->2 from top left to bottom right

By looking at the figure 2.13 above, we see that after going through all the commutation states (0-5) once, the rotor has rotated half a round. This means that one mechanical round corresponds to two electrical rounds, given by the equation (2.15) below.

$$\theta_e = \theta_m \cdot p \tag{2.15}$$

$$\theta_m = \frac{\theta_e}{2} \tag{2.16}$$

Where p is the number of pole pairs.

2.3.2 Space vector PWM control

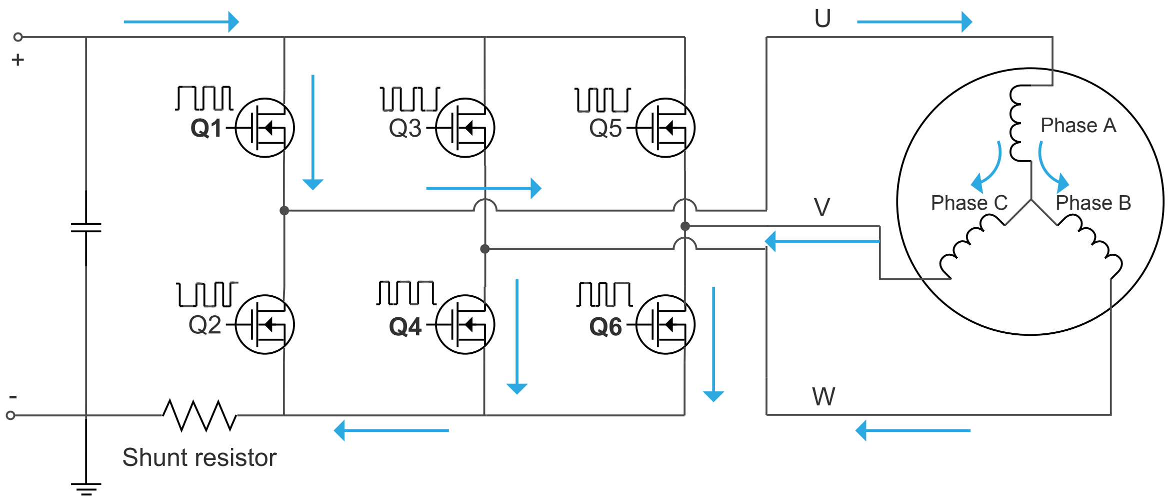


Figure 2.14: 3-Phase inverter: Current flows to motor windings from phase A->B & C, de-energized phase = none

The SVPWM technique gives a constant switching frequency, which should provide smooth transitions between commutation states. The maximum phase output voltages is

$$V_{\text{ph}_{\text{max}}} = \frac{V_{\text{DC}}}{\sqrt{3}} \tag{2.17}$$

$$V_{\text{ph}_{\text{rms}}} = \frac{V_{\text{DC}}}{\sqrt{6}} \tag{2.18}$$

$$V_{\text{l-l}_{\text{max}}} = V_{\text{DC}} \tag{2.19}$$

$$V_{\text{l-l}_{\text{rms}}} = \frac{V_{\text{DC}}}{\sqrt{2}} \tag{2.20}$$

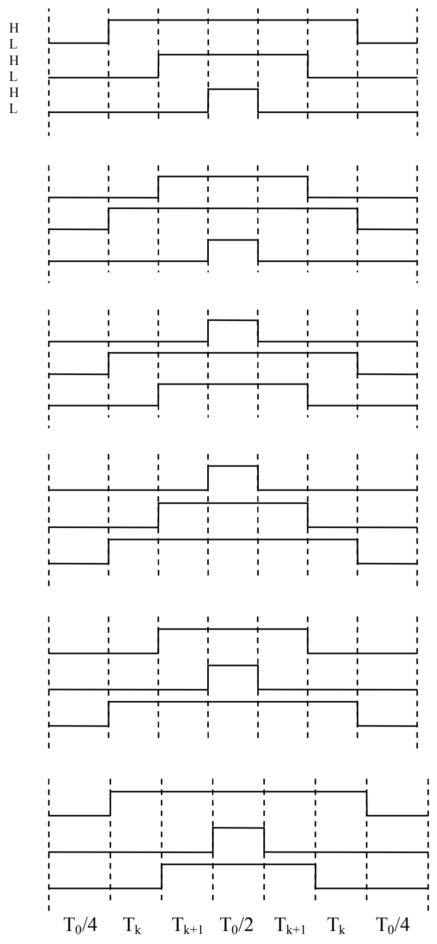


Figure 2.15: Switching patterns [14]

There are eight different combinations of on/off switching of the transistors. The time periods of how long the switches are on and off is calculated using space vectors, explained in section 2.4.3. The possible switching patterns are shown in figure 2.15 to the left.

2.4 Motor control

The challenge of driving an electric motor, is knowing when to commutate. To commutate at the best possible moment, we need to have an accurate knowledge of the rotor position. By commutating too fast or too slow, the rotor will start to stall. There are different ways to do this, using sensorless or sensored motor control techniques.

2.4.1 Sensorless

The rotor position can be found by reading of the back-EMF. The back-EMF is only available for reading when the motor terminal is not driven. The phases are commutated every 60° of the electrical rounds. As illustrated in 2.12, the back-EMF crosses zero in the commutation states where the motor terminal is not driven (low and high transistor pair output is low). By measuring the back-EMF generated in this 60° interval, we can find the zero-crossing point.

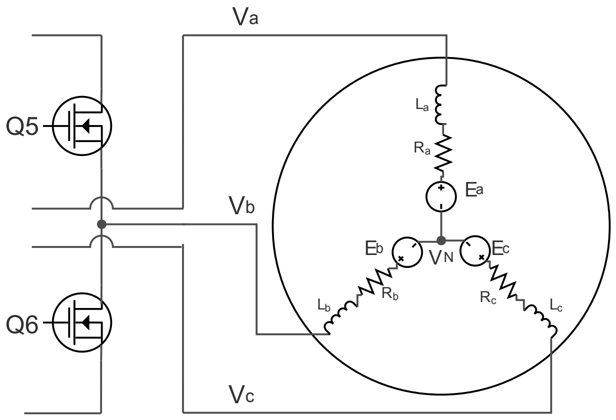


Figure 2.16: Neutral voltage point

By looking closer at the motor in figure 2.16 above, and assuming that the current is flowing through phase C to phase B, we can describe the voltages:

$$V_a = E_a + V_N \tag{2.21}$$

$$V_b = R_b \cdot I_b + L\frac{dI_b}{dt} + E_b + V_n \tag{2.22}$$

$$V_c = R_c \cdot I_c + L\frac{dI_c}{dt} + E_c + V_n \tag{2.23}$$

$$\tag{2.24}$$

Where E_a, E_b and E_c are the back-EMF seen by the phases, V_N is the neutral voltage and $I_b = -I_c$. This gives:

$$V_a + V_b + V_c = E_a + E_b + E_c + 3V_N \tag{2.25}$$

When the back-EMF read from phase A crosses zero, the sum of all back-EMF generated in the phases equals to zero. This is also shown in figure 2.12. This leads to:

$$V_a + V_b + V_c = 3V_N \tag{2.26}$$

And the back-EMF in the non-energized phase A can be calculated:

$$3E_a = 3V_a - 3V_N$$

The figure 2.17 below illustrates the back-EMF read in the different phases in the different commutation states. The green line shows the back-EMF read from phase A when the current flows from phase C to phase B.

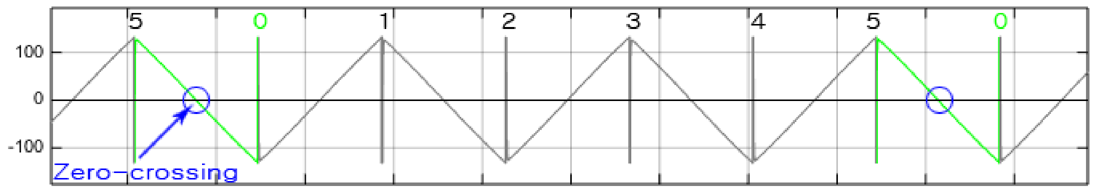


Figure 2.17: Zero-crossing method

When we have the zero-crossing point, we know that the next commutation should be triggered 30° later. By using a virtual timer, the time period measured for each of the commutation states can be used to calculate how long time the movement of 30° electrical degrees takes.

2.4.2 Hall-effect sensors

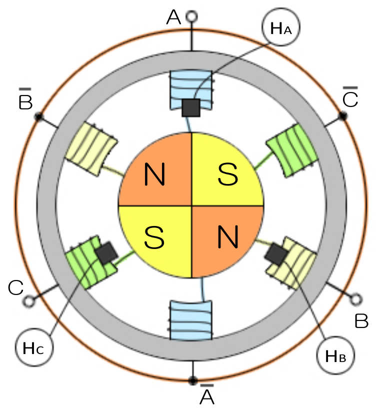


Figure 2.18: Hall-effect sensors

The rotor position can be determined by using Hall-effect sensors, which are along the stator. The Hall sensors gives feedback, either a high signal or a low signal, depending on the change of magnetic field. The responses from the sensors is therefore depended on the back-EMF. The outputs of the Hall sensors and the back-EMF are illustrated in figure 2.19 below.

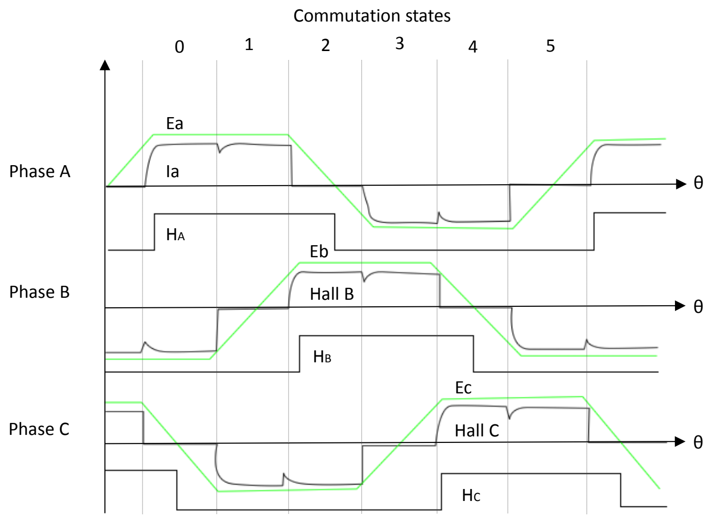


Figure 2.19: Hall-effect sensors logic

The hall-effect outputs determines the commutation states, listed in table 2.2 below.

State	$\mathbf{H_A}$	$\mathbf{H_B}$	$\mathbf{H_C}$	$\mathbf{H_{tot}}$	$\mathbf{I_a}$	$\mathbf{I_b}$	$\mathbf{I_c}$
0	1	0	1	5	+	-	OFF
1	1	0	0	1	+	OFF	-
2	1	1	0	3	OFF	+	-
3	0	1	0	2	-	+	OFF
4	0	1	1	6	-	OFF	+
5	0	0	1	4	OFF	-	+

Table 2.2: Hall-effect commutation table

2.4.3 Field-oriented control

One effective way of controlling a motor is by using field-oriented control (FOC), also called space vector control. By using FOC we control the stator currents represented by a vector. The stator currents is considered as two orthogonal components defined by a vector. The components are torque and magnetic flux of the motor. The goal of FOC is to be able to control the torque and flux components separately. The torque can be controlled by tuning the current flow through the windings. The torque is proportional to the amount of current flow through the windings. By measuring the current in the phases we can determine how to control the magnetic field produced by the stator. The torque is generated by the interaction of the magnetic field in the stator and the magnetic field in the rotor.

By controlling the three currents we can create a current vector with the magnitude and angle that we want, using space vectors [14] [3]. The torque produced by a three phased permanent magnet motor is shown in figure 2.20 below. It is a function of the angle between the rotor flux and the current MMF from the stator. The maximum torque per amp is achieved when the stator current MMF vector is oriented 90° with respect of the rotor flux vector.

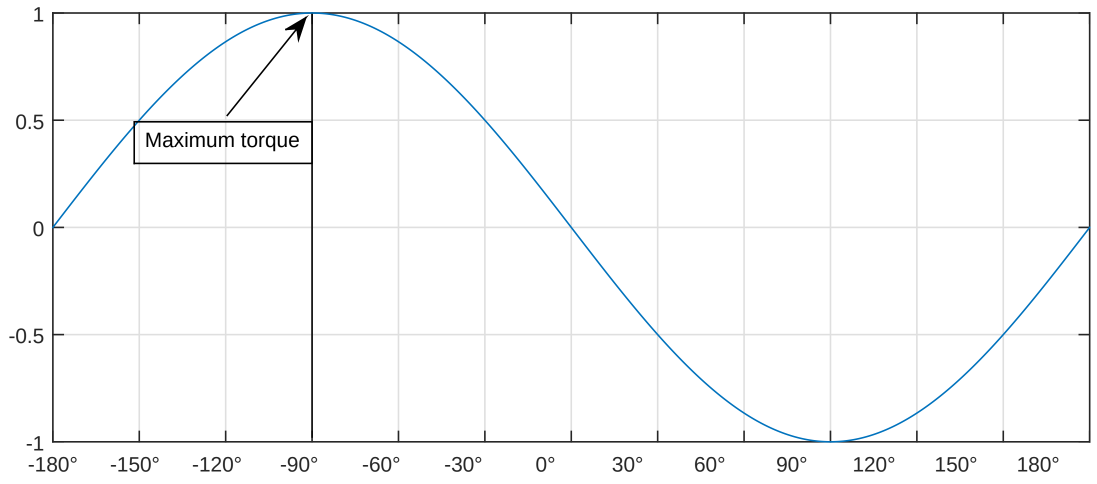
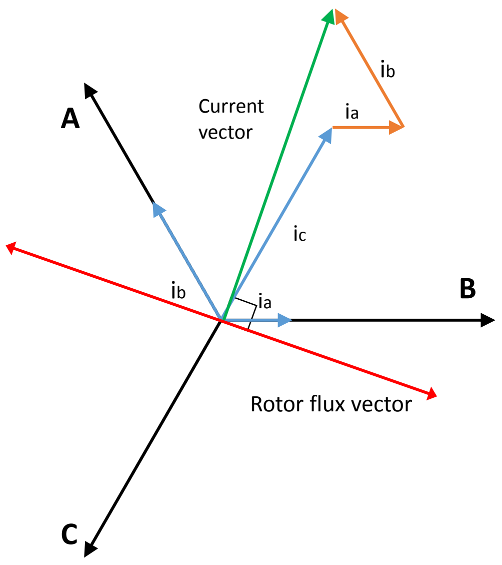


Figure 2.20: Maximum torque per amp

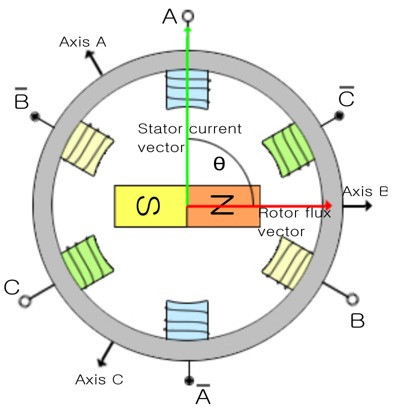
We want to have maximum torque per amp at all times, which means that we want to control the phase currents so that the stator current MMF vector is always oriented 90° with respect of the rotor flux vector. The torque is proportional to the product of the rotor flux times the component of the stator current vector.

$$\tau \propto \mathbf{B}_{stator} \times \mathbf{B}_{rotor} \tag{2.27}$$

To control the torque generated by the stator and rotor magnetic fields, we can regulate the angle and amplitude of the stator current vector. The rotor flux angle is constantly measured by an encoder or a resolver and the current vector is regulated to be 90° with respect of the rotor flux vector. As the motor is a synchronous machine and the rotor has permanent magnets , the rotor flux angle is always the same in respect of the rotor angle. The rotor flux vector is shown in figure 2.21b. When the rotor flux angle is measured, we need to calculate the needed current values for the three phases to create a stator current vector which is 90° with respect of the rotor flux angle. The current vector is found by sum the phase currents, as show in figure 2.21a. The current vector in the figure is 90° with respect of the rotor flux vector, which is what we want. The axes in the figure are in a reference frame called ”stator frame", which means that the axes are fixed with respect of the motor stator shown in figure 2.21b.



(a) Stator reference frame. The desired current vector is calculated from the rotor angle



(b) Stator current and rotor flux vector

By doing a mathematical transform, we can simplify the three-phase motor into a two-phase motor. We can do this by representing the current vector using current a and current b (as $i_a + i_b + i_c = 0$, the phase c current can be calculated an does not need to be measured). This means that we only need to regulate two currents, instead of all three of them. This method is called forward Clarke transformation. The new vectors are calculated from equations (2.28).

$$i_d = i_a \tag{2.28}$$

$$i_q = \frac{1}{\sqrt{3}}(i_a + 2i_b) \tag{2.29}$$

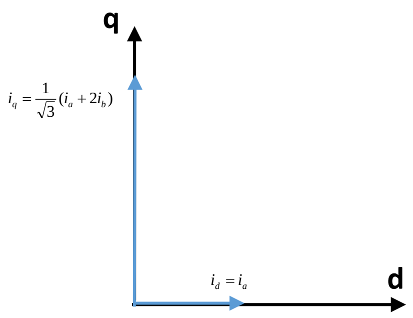


Figure 2.22: d-q reference frame

By looking at the components in a rotating reference frame instead of the stator frame, the regulation of currents will be easier. This is done by using the forward Park transformation shown in equations (2.4.3) and figure 2.23 below.

$$i_D = i_d cos\theta_d + i_q sin\theta_d \tag{2.30}$$

$$i_Q = -i_d sin\theta_d + i_q cos\theta_d \tag{2.31}$$

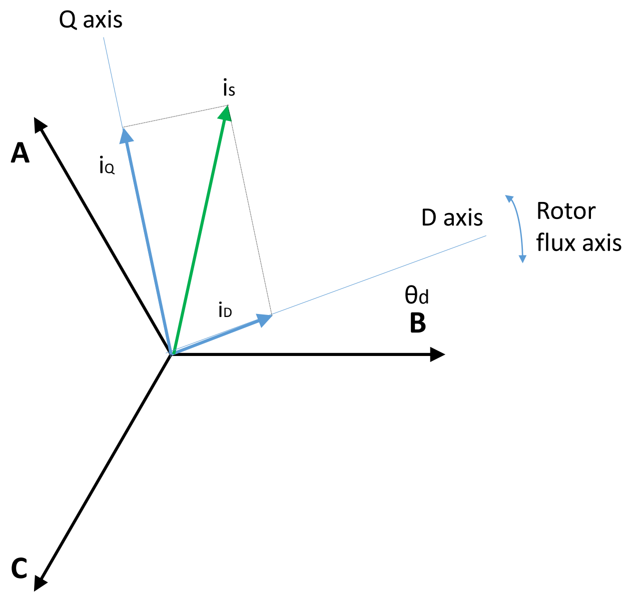


Figure 2.23: Rotary reference frame

The stator vector is now transformed into the components i_D and i_Q which can be regulated using two PI-controllers. Note that PID-controllers are usually not used if the sampling frequency in the controller is high.

We would like to have all of the stator currents in 90° with respect of the rotor flux vector. That means that the reference for i_d should be zero as we don't want the current on the d-axis, but all on the q-axis. The i_q is the reference in torque that the motor will generate.

The output of the PI-controllers is going to be used to regulate the duty ratios to generate a given stator reference voltage for the PWM signals used in the inverter. To do this we need to transform the vectors from the rotary reference frame back to the stator reference frame, using a method called inverse Park transformation.

$$V_{s\alpha} = I_D cos\theta_d - I_Q sin\theta_d \tag{2.32}$$

$$V_{s\beta} = I_D sin\theta_d + I_Q sin\theta_d \tag{2.33}$$

Where I_D and I_Q are outputs from the regulators.

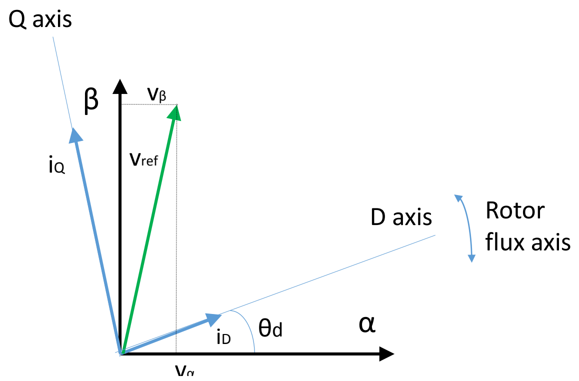


Figure 2.24: Alpha-beta reference frame

By using the inverse Clarke equation we can transform the two-phased voltage system back to a three-phase system:

$$V_{\text{ref1}} = U_\beta \tag{2.34}$$

$$V_{\text{ref2}} = -\frac{U_\beta + U_\alpha \cdot \sqrt{3}}{2} \tag{2.35}$$

$$V_{\text{ref3}} = -\frac{U_\beta - U_\alpha \cdot \sqrt{3}}{2} \tag{2.36}$$

(2.37)

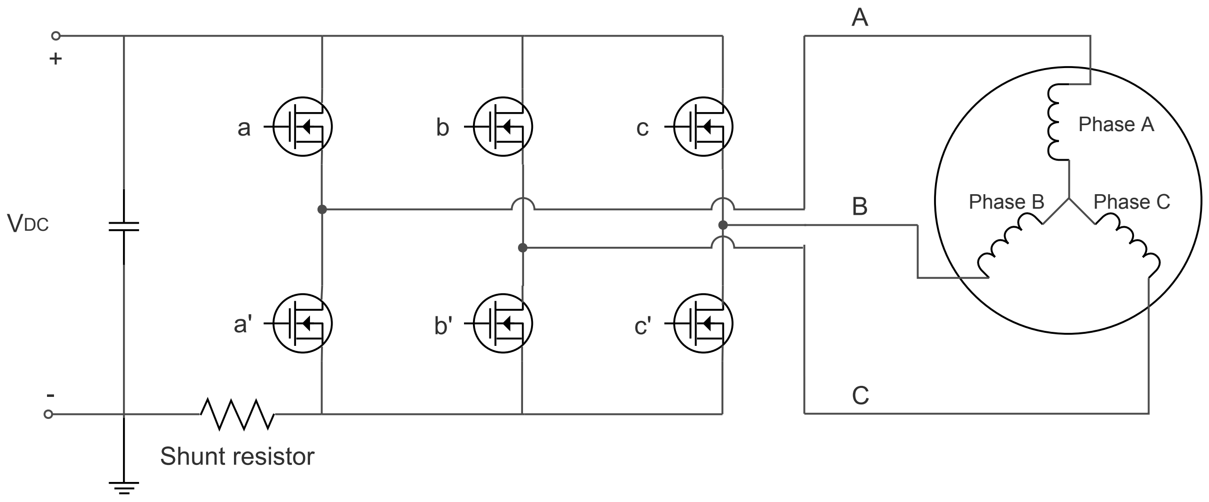


Figure 2.25: 3-Phase inverter where the switches at the bottom are inverted from the top ones.

The switches on the inverter has 8 different combinations and is listed in the table 2.3 below for the space vector method.

a	**b**	**c**	$V_{\mathbf{AN}}$	$V_{\mathbf{BN}}$	$V_{\mathbf{CN}}$	$V_{\mathbf{AB}}$	$V_{\mathbf{BC}}$	$V_{\mathbf{CA}}$
0	0	0	0	0	0	0	0	0
0	0	1	$\frac{2}{3}V_{DC}$	$-\frac{1}{3}V_{DC}$	$-\frac{1}{3}V_{DC}$	V_{DC}	0	$-V_{\mathrm{DC}}$
0	1	0	$-\frac{1}{3}V_{DC}$	$\frac{2}{3}V_{DC}$	$-\frac{1}{3}V_{DC}$	$-V_{\mathrm{DC}}$	V_{DC}	0
0	1	1	$\frac{1}{3}V_{DC}$	$\frac{1}{3}V_{DC}$	$-\frac{2}{3}V_{DC}$	0	V_{DC}	$-V_{\mathrm{DC}}$
1	0	0	$-\frac{1}{3}V_{DC}$	$-\frac{1}{3}V_{DC}$	$\frac{2}{3}V_{DC}$	0	$-V_{\mathrm{DC}}$	V_{DC}
1	0	1	$\frac{1}{3}V_{DC}$	$-\frac{2}{3}V_{DC}$	$\frac{1}{3}V_{DC}$	V_{DC}	$-V_{\mathrm{DC}}$	0
1	1	0	$-\frac{2}{3}V_{DC}$	$\frac{1}{3}V_{DC}$	$\frac{1}{3}V_{DC}$	$-V_{\mathrm{DC}}$	0	V_{DC}
1	1	1	0	0	0	0	0	0

Table 2.3: On/off values for the switches.

Compared to the regular PWM method explained in section 2.3, all the phases in space vector modulation are on at the same time instead of only two. This should provide a smoother rotation of the rotor. The stator currents will result in more or less like sinusoidal currents.

The switching patterns in table 2.3 is divided into sectors in the $\alpha - \beta$ reference frame as shown in figure 2.26 below.

By looking at the voltage reference generated by the regulators, we can decide which sector we are in.

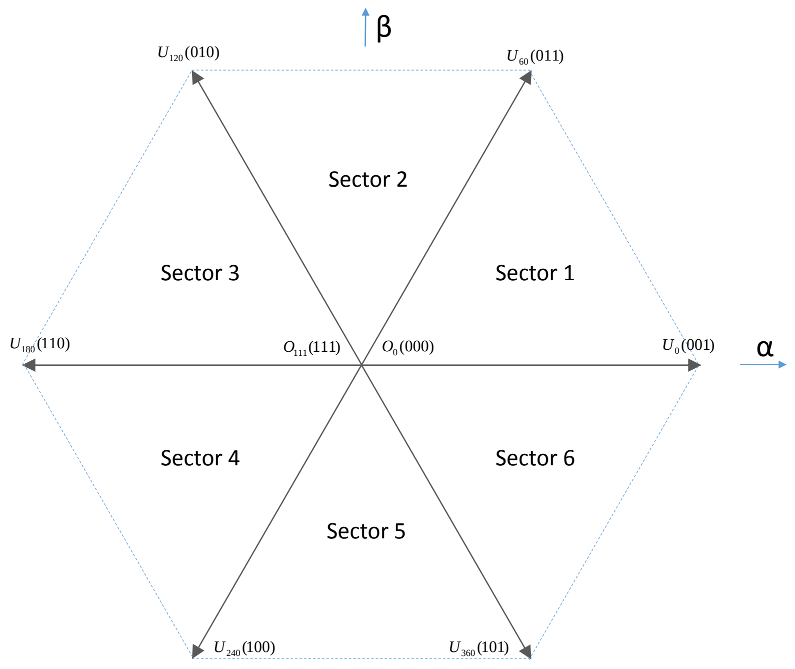


Figure 2.26: Space vector sectors overview

The corresponding space vectors and their (α, β) components from the switching patterns in table 2.3 are listed in the table below.

c	**b**	**a**	$V_{s\alpha}$	$V_{s\beta}$	**Vector**
0	0	0	0	0	O_0
0	0	1	$\frac{2}{3}V_{\mathrm{DC}}$	0	U_0
0	1	0	$-\frac{1}{3}V_{\mathrm{DC}}$	$\frac{1}{\sqrt{3}}V_{\mathrm{DC}}$	U_{120}
0	1	1	$\frac{1}{3}V_{\mathrm{DC}}$	$\frac{1}{\sqrt{3}}V_{\mathrm{DC}}$	U_{60}
1	0	0	$-\frac{1}{3}V_{\mathrm{DC}}$	$-\frac{1}{\sqrt{3}}V_{\mathrm{DC}}$	U_{240}
1	0	1	$\frac{1}{3}V_{\mathrm{DC}}$	$-\frac{1}{\sqrt{3}}V_{\mathrm{DC}}$	U_{300}
1	1	0	$-\frac{2}{3}V_{\mathrm{DC}}$	0	U_{180}
1	1	1	0	0	O_{111}

Table 2.4: Space vector switching patterns

If we use the voltage reference from figure 2.24 and compare it to the sector diagram in the figure 2.26 above, the sector is found to be 2. See figure 2.27 below for illustration.

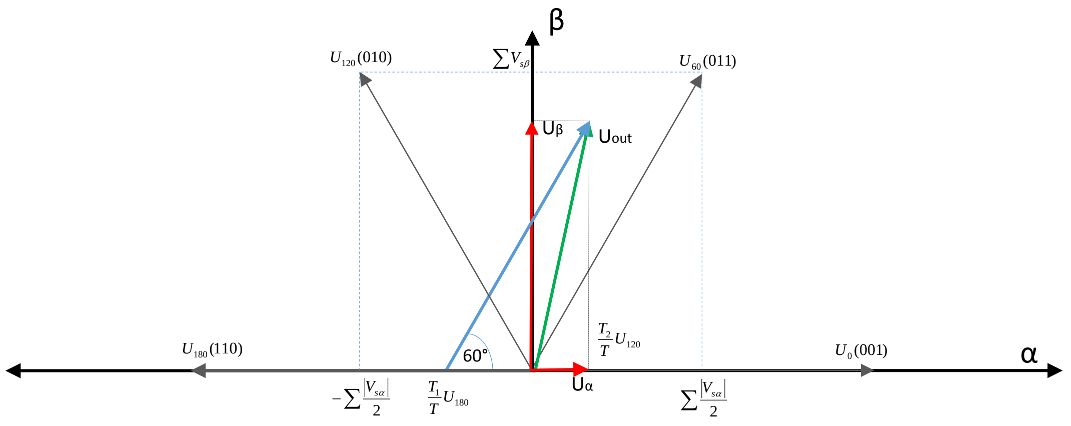


Figure 2.27: α - β reference frame, from sector diagram

The parameters in figure 2.27 are found using equations:

$$\sum V_{s\beta} = \frac{V_{\mathrm{DC}}}{\sqrt{3}} + \frac{V_{\mathrm{DC}}}{\sqrt{3}} = \frac{2V_{\mathrm{DC}}}{\sqrt{3}} \tag{2.38}$$

$$\sum V_{s\alpha} = -\frac{V_{\mathrm{DC}}}{3} + \frac{V_{\mathrm{DC}}}{3} = 0 \tag{2.39}$$

Where $\sum V_{s\beta}$ and $\sum V_{s\alpha}$ is the sum of β and α components respectively, found in table 2.4.

$$T = T_1 + T_2 + T_0 \tag{2.40}$$

$$U_{\mathrm{out}} = \frac{T_1}{T}U_{60} + \frac{T_2}{T}U_{180} \tag{2.41}$$

Where T_1, T_2 and T_0 is the time where U_{180} and U_{120} and the null vector respectively are applied within the period T. The time durations is calculated:

$$U_\beta = \frac{T_2}{T}|U_{120}|sin(120°) \tag{2.42}$$

$$U_\alpha = \frac{T_1}{T}|U_{180}| + \frac{T_2}{T}|U_{180}|cos(180°) \tag{2.43}$$

All of the space vectors has the same magnitude as seen in figure 2.26. The is defined by $\frac{2}{3}V_{\mathrm{DC}}$, and is equal to $\frac{2}{\sqrt{3}}$ when normalized. That means:

$$U_\beta = \frac{T_2}{T} \cdot \frac{2}{\sqrt{3}} \cdot \frac{\sqrt{3}}{2} = \frac{T_2}{T} \tag{2.44}$$

$$U_\alpha = \frac{T_1}{T} \cdot \frac{2}{\sqrt{3}} + \frac{T_2}{T} \cdot \frac{2}{\sqrt{3}} \cdot (-1) = \frac{\sqrt{3} \cdot T_1}{3 \cdot T} - \frac{\sqrt{3} \cdot T_2}{3 \cdot T} \tag{2.45}$$

Which gives

$$T_2 = U_\beta \cdot T \tag{2.46}$$

$$T_1 = (U_\alpha + \frac{\sqrt{3} \cdot T_2}{3 \cdot T}) \cdot \frac{\sqrt{3} \cdot T}{2} \tag{2.47}$$

$$= \frac{T}{2} \cdot (\sqrt{3} \cdot U_\alpha + \frac{T_2}{T}) \tag{2.48}$$

$$= \frac{T}{2} (\sqrt{3} \cdot U_\alpha + U_\beta) \tag{2.49}$$

And

$$t_1 = \frac{T_1}{T} = U_\beta \tag{2.50}$$

$$t_2 = \frac{T_2}{T} = \frac{1}{2} (\sqrt{3} \cdot U_\alpha + U_\beta) \tag{2.51}$$

These are the time durations that space vectors U_{120} and U_{180} are applied.

Texas Instruments has defined 3 variables X,Y and Z [3] according to the following equations:

$$X = U_\beta \tag{2.52}$$

$$Y = \frac{1}{2} (\sqrt{3} U_\alpha + U_\beta) \tag{2.53}$$

$$Z = \frac{1}{2} (-\sqrt{3} U_\alpha + U_\beta) \tag{2.54}$$

The time duration in each of the sectors is shown in the table 2.5 below.

Sector	U_0, U_{60}	U_{60}, U_{120}	U_{120}, U_{180}	U_{180}, U_{240}	U_{240}, U_{300}	U_{300}, U_0
t_1	-Z	Z	X	-X	-Y	Y
t_2	X	Y	Y	Z	-Z	-X

Table 2.5: Time durations of the space vectors for each of the sectors

By comparing $t_1 = X$ and $t_2 = Y$ with the results from the example above, we see that this is correct.

The time periods t_1, t_2 and t_0 ($t_0 = t_2 - t_1$) is illustrated in figure 2.28 below:

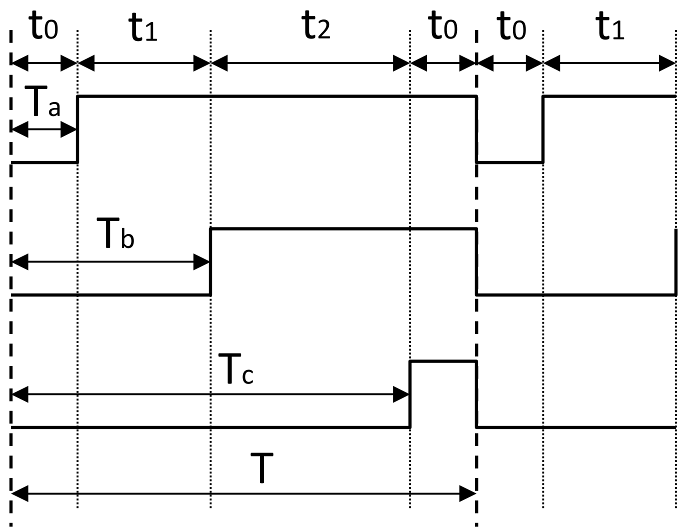


Figure 2.28: Duty cycles and time periods for the space vectors

Where $T = 1$ PWM period and:

$$T_a = \frac{1}{2}(T - t_1 - t_2) \tag{2.55}$$

$$T_b = t_1 + T_a \tag{2.56}$$

$$T_c = t_2 + T_b \tag{2.57}$$

Where T_a, T_b and T_c is the duty cycles. The last step is to apply the duty cycles to the right phases:

Sector	U_0, U_{60}	U_{60}, U_{120}	U_{120}, U_{180}	U_{180}, U_{240}	U_{240}, U_{300}	U_{300}, U_0
T_A	T_a	T_b	T_c	T_c	T_b	T_a
T_B	T_b	T_a	T_a	T_b	T_c	T_c
T_C	T_c	T_c	T_b	T_a	T_a	T_b

Table 2.6: Assigning the duty cycles to the right phases

See figure 2.15 in section 2.3.2 to see how the switching patterns should look like.

2.5 Microcontrollers

To drive and regulate a motor, we need a form for motor control. What we need is a microcontroller, and some peripherals.

A microcontrolller, or μC, is like a tiny computer. Just like a laptop, or desktop computer, the microcontroller consists of a processor, memory and I/O ports. The laptop, smartphone, and tablet PC is run by an operating system, which allows the user to use the computer as a text editor, sound player, movie player, etc.

In contrast with the desktop computer, the microcontroller is often intended for a single application. It is therefore ideal suited for motor controlling. This is called an embedded microcontroller.

The figure below shows a very simplified overview of a microcontroller. The most basic parts of the microprocessor will be briefly explained below.

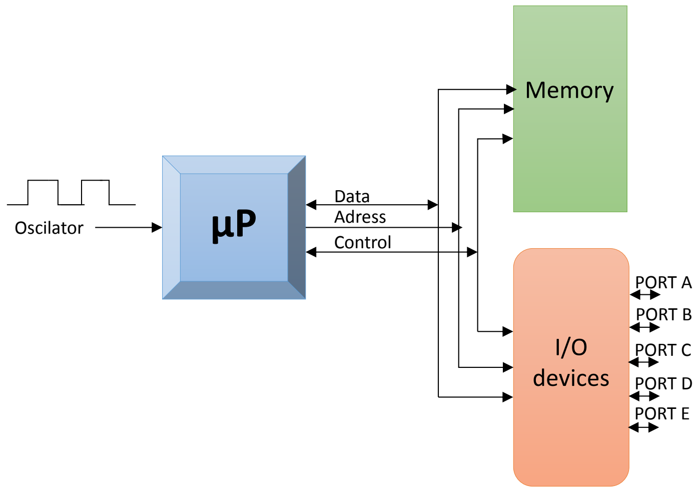


Figure 2.29: Simplified layout of a microcontroller

2.5.0.1 Microprocessor

The microprocessor, or μP, is the brain of the microcontroller. The μP use data, or information, and translates it in the processors own ”language”. The data on its lowest level is basically just a sequence of 1’s and 0’s numbers, called machine code. The different sequences is called instructions and each processor can interpret these instructions differently. The most basics instructions are adding, storing, incrementing numbers. These instructions can be translated in to a more readable text called the assembly language. The processor is basically a powerful calculator.

2.5.0.2 Memory

The memory contains the instructions that the processor is going to process. The memory will constantly be read from the processor, and sometimes written to. If the instruction is to add two numbers and store them in memory, the instruction is firstly read. Afterwards the processor will read and then move the first number to a register in the memory, and then add the second one. The register in the memory now holds the answer.

The architecture in figure 2.29 is called a *Von Neumann machine*. The processor does not know if it's reading instructions or data from the memory. This means that the instruction can be treated as data, and vise versa.

A more effective architecture is the *Harvard architecture*, where the data- and program memory is divided into two different memory spaces. In this architecture, there is used separate data, address and control buses. See figure 2.30 below.

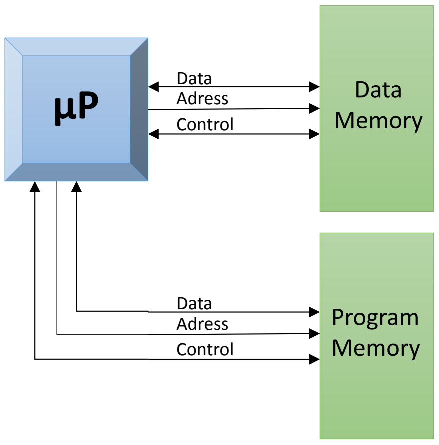


Figure 2.30: The Harvard architecture

2.5.0.3 Buses

Buses are signal lines that carries information of a specific type. In the previous figures there are three different buses.

Data

The data bus carries data which is stored in memory blocks. The memory block usually holds 1, 2, 3 or 4 bytes (8, 16, 32, 64 bits respectively). The index of the byte, or word, has its own address. See figure 2.31. This bus is bidirectional, which means that the bus can transfer data in both direction.

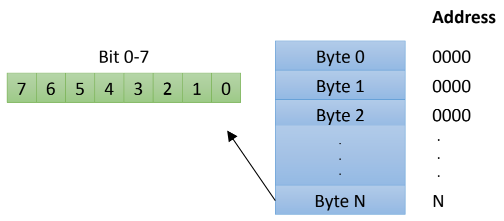


Figure 2.31: N byte wide memory block

Address

The address bus tells the microprocessor, memory, or other peripherals where in the memory data the data should be read or written to. If the address bus is 32-bit wide, the bus can address $2^{32} = 4294967296$ memory allocations. This means that the addressable memory space is 4 GB if one memory address holds one word.

Control

The control bus keeps track of whether the current operation of the processor is to read or to write. It also carries an error signal back to the processor if there is any problems with the current operation.

2.5.0.4 Interrupts

The most important part to know about the microprocessor in this paper, is the use of interrupts. An interrupt interrupts the main program (often a loop containing a timer) and makes a jump to a certain place in the memory. This place contains some new instructions, and when the processor is done processing these, it will continue from the place it was last interrupted. This is done by the *interrupt service routine*, ISR.

The advantage of this is that the microprocessor doesn't continuously has to check all the I/O devices for pending operations.[1] It will rather get a notification from the device when it needs processing. Imagine if clothing store employees operated like this!

[1]This is also known as polling or busy waiting

The interrupts can be either a low priority or a higher priority. This basically means that the low priority interrupt can be interrupted by a higher priority one. When the interrupts occur, the processor saves the current state to know where it continue after the interruption. It then loads an *interrupt vector* into the program counter, which holds the address to where the ISR lies. This will start the execution of the interrupt instructions. The last instruction in the interrupt is a *return from interrupt* instruction.

2.5.0.5 Digital Signal Processor

A digital signal processor, DSP, is very similar to the microprocessor. The DSP has a few extra features, which makes it better suitable for processing signals. The DSP has for instance additional data buses and enhanced calculating and storing techniques, including other hardware used for processing signals. The DSP is simply a very powerful calculator for signal processing.

Typical DSP algorithms	
Finite Impulse Response Filter	$y(n) = \sum\limits_{k=0}^{M} a_k x(n-k)$
Infinite Impulse Response Filter	$y(n) = \sum_{k=0}^{M} a_k x(n-k) + \sum\limits_{k=1}^{N} b_k y(n-k)$
Convolution	$y(n) = \sum\limits_{k=0}^{M} x(k)h(n-k)$
Discrete Fourier Transform	$X(k) = \sum\limits_{n=0}^{N-1} x(n)^{-j\frac{2\pi}{N}nk}$
Discrete Cosine Transform	$F(u) = \sum\limits_{x=0}^{N-1} c(u)f(x)cos(\frac{\pi}{2N}u(2x+1))$

If we were to do a sum of products calculation with a regular microprocessor, it would need to use a lot more computing power than a DSP. The following example is obtained from Texas Instruments [15]:

Example

$$y = \sum_{i=0}^{3} \text{data}[i]\text{coeff}[i]$$

C-Code:

```c
#include <stdio.h>
int data[4] = {1,2,3,4};
int coeff[4] = {8,6,4,2};

int main(void)
{
int i;
int result = 0;

for (i = 0;i < 4;i++)
result += data[i]*coeff[i];
printf("%i",result);
return 0;
}
```

A Pentium processor will calculate the formula like this:

1. Set a Pointer1 to point to data[0]
2. Set a second Pointer2 to point to coeff[0]
3. **Read data[i] into core**
4. **Read coeff[i] into core**
5. **Multiply data[i]*coeff[i]**
6. **Add the latest product to the previous one**
7. **Modify Pointer1**
8. **Modify Pointer2**
9. Increment i;
10. If i<4 , then go back to step 3 and continue

The steps from 3 to 8 are called ”6 basics operation of a DSP”. The DSP can actually execute all 6 steps in a single cycle! The Pentium would use 15 assembly instructions to calculate this formula, while a DSP could do it in only 9 instructions. This means that the DSP can save a lot of computing power in heavy calculations.

2.5.0.6 Digital Signal Controller

A controller that is based on a DSP instad of a regular microprocessor, is called a *Digital Signal Controller*. The three biggest manufacturers of DSCs are:

- Texas Instruments - TMS320 series
- Analog Devices - 21xx and SHARC (21xxx)
- Freescale Semiconductor - DSP56xxx and MSC8100 StarCore

2.5.0.7 Analog to Digital Conversion

The Analog to Digital Conversion, or DAC, peripheral converts analog signals into digital. The output represents the ratio of the input signal to a given *reference voltage*. If the ADC is 8-bit $2^8 - 1 = 255$ is the full scale. That means if the input voltage is 2V and the reference voltage is 5V, the output value is $\frac{2}{5}255 = 102$.

$$OUT = \frac{IN}{max_value} V_{ref} \tag{2.58}$$

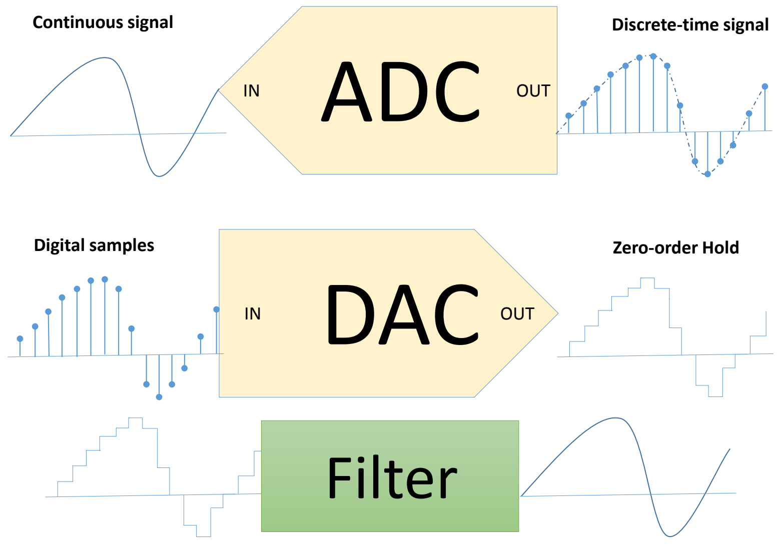


Figure 2.32: Converting from analog to digital and back

We can describe the analog signal with period time T [16]:

$$x_a(t) = x(t+T) \tag{2.59}$$

And the discrete-time signal with period samples N:

$$x_d(n) = x(n+N) \tag{2.60}$$

The ADC performs a sampling which can be described as

$$x(n) = x_a(nT_s) \tag{2.61}$$

The DAC uses a zero-order-hold (ZOH) method which is a linear time-invariant system (LTI). The ZOH method holds each sample value $u(k)$ from the continuous-time signal $u(t)$ constant over one sample period:

$$u(t) = u[k], \qquad kT_s \leq t \leq (k+1)T_s \tag{2.62}$$

2.5.0.8 Numbering systems

The numbers processed by processors are handled in two ways, either using floating-point or fixed-point system (or both). In very short words, with the floating-point system we can represent a number using decimal places. For example: $x = 3.1415$. With fixed point system, the number is represented like this: $x = \frac{6283}{2000}$ where the scaling factor is always known (in this example the scaling factor is 2000). Matlab represents numbers using floating type. [17]

Texas Instruments has implemented a library called "IQMath" in their hardware and software for their microcontrollers, which makes the number representation easier. The decimal number is converted to an integer using a specified scaling factor. For example $\_IQ15(1.0) = 2^{15} \cdot 1.0)$. The hardware will then convert the number to a floating point by dividing the number with the same scaling factor. See library reference in [18] for a better understanding.

Chapter 3

System

3.1 System model of the DC motor

The following equations [19] describes a DC motor with constant rotor flux (permanent magnets).

By using Kirchoffs law in the circuit in figure 3.1 we get

$$V_s(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + V_{emf}(t) \tag{3.1}$$

Where $V_{emf}(t) = K_e \omega_m(t)$ is the back-EMF, K_e is the rotor emf, L_a is the inductance, R_a is the resistance, V_s is the applied voltage and $i_a(t)$ is the current. The notation a stands for armature.

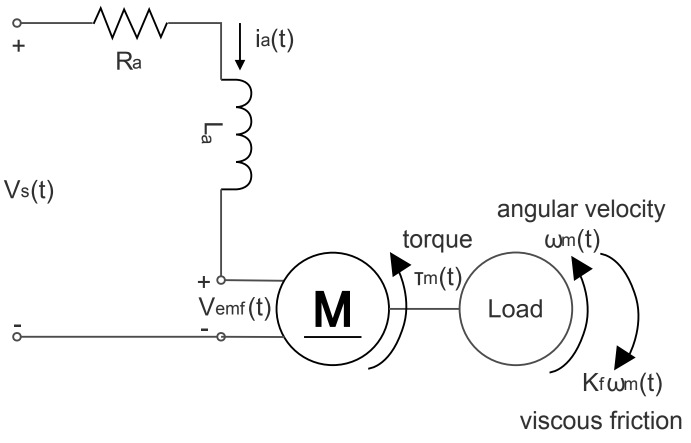


Figure 3.1: Simplified DC motor model

The motor load system is described using Newton's second law. The inertial load J_m times the angular acceleration is equal to the sum of all torques:

$$J_m \frac{d\omega(t)}{dt} = \sum \tau_i \tag{3.2}$$

$$J_m \frac{d\omega_m(t)}{dt} = K_m i_a(t) - K_f \omega_m(t) \tag{3.3}$$

Where $K_m i_a(t) = \tau_m$ is the torque in the rotor, $K_f \omega_m(t) = \tau_l$ is the torque in the load and K_m is the magnetic field strength in the armature.

Faraday's law of induction states that a change in the magnetic field in a coil causes a voltage, or electromagnetic force, to be induced in the coil. Lorentz's law describes the force upon a coil in a magnetic field. See more information in section 2.1 in chapter 2.

$$\frac{di_a(t)}{dt} = -\frac{R_a}{L_a} i_a(t) + \frac{1}{L_a} V_s(t) - \frac{K_b}{L_a} \omega_m(t) \tag{3.4}$$

$$\frac{d\omega_m(t)}{dt} = \frac{K_m}{J_m} i_a(t) - \frac{K_f}{J_m} \omega_m(t) \tag{3.5}$$

Where K_f is the result of friction in the motor.
The state-space system can be described as

$$\frac{d}{dt} \begin{bmatrix} i_a(t) \\ \omega_m(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_e}{L_a} \\ \frac{K_m}{J_m} & -\frac{K_f}{J_m} \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega_m(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} V_s(t) \tag{3.6}$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega_m(t) \end{bmatrix} \tag{3.7}$$

Which leads to the transfer function

$$G(s) = \frac{\omega_m(t)}{V_s(t)} = \frac{K_m}{J_m L_a s^2 + (R_a J_m + K_f L_a)s + R_a K_f + K_e K_m} \tag{3.8}$$

The friction coefficient K_f is usually much smaller than $R_a J_m$ and by neglecting the friction we can write

$$G(s) = \frac{K_m}{J_m L_a s^2 + R_a J_m s + K_e K_m} \tag{3.9}$$

or

$$G(s) = \frac{1/K_e}{\frac{L_a J_m}{K_e K_m} s^2 + \frac{R_a J_m}{K_e K_m} s + 1} \tag{3.10}$$

Where $\frac{R_a J_m}{K_e K_m} = \tau_m$ and $\frac{L}{R} = \tau_e$.

$$G(s) = \frac{1/K_e}{\tau_m \tau_e s^2 + \tau_m s + 1} \tag{3.11}$$

$$\tau_e = \frac{L}{3R} = \frac{33.50 \cdot 10^{-3}}{3 \cdot 11}$$
$$\tau_e = 1015.15 \cdot 10^{-3}$$

BLWS235D-160V-3000		
Rated Voltage	160	V
Rated Speed	3000	RPM
Rated Power	157	W
Rated Torque	0.50	Nm
No Load Current	0.20	A
Torque Constant	0.33	Nm/A
Back EMF Voltage	26	V/kRPM
Line to Line Resistance	11	ohms
Line to Line Inductance	33.50	mH
Rotor Inertia	0.000023	Nms^2

Table 3.1: Specifications for the BLDC motor from Anaheim Automation [20]

$$K_e = \frac{3 \cdot R \cdot J_{tot}}{\tau_m \cdot K_t} = 0.2483$$

Where J_{tot} is the total moment of inertia, calculated in section 4.2.2.

$$\tau_m = \frac{3 \cdot R \cdot J_{tot}}{K_e \cdot K_t} = \frac{3 \cdot 11 \cdot 0.002}{0.2483 \cdot 0.33} = 0.7977$$

$$G(s) = \frac{1/K_e}{\tau_m \tau_e s^2 + \tau_m s + 1}$$

$$G(s) = \frac{4.028}{0.8097 \cdot 10^{-3} s^2 + 0.7977 s + 1} \tag{3.12}$$

3.2 Sensorless driving

The sensorless driving is done by observing the motors back-emf. It is implemented in two stages: open loop and closed loop driving.

There is no back-emf produced while the motor is not running. This is why an open loop control is needed at the start up phase of the motor, to ramp up the motor speed.

3.2.1 Open loop

The sensorless driving using the open loop method consists of the following stages:

1. Prepositioning
2. $RC3\_MACRO$ - Ramping to desired commutation period
3. $IMPULSE\_MACRO$ - Impulse generator (create an impulse each commutation period)
4. $MOD6CNT\_MACRO$ - Determine commutation number
5. $BLDCPWM\_MACRO$ - Timing when to commutate. Commutate

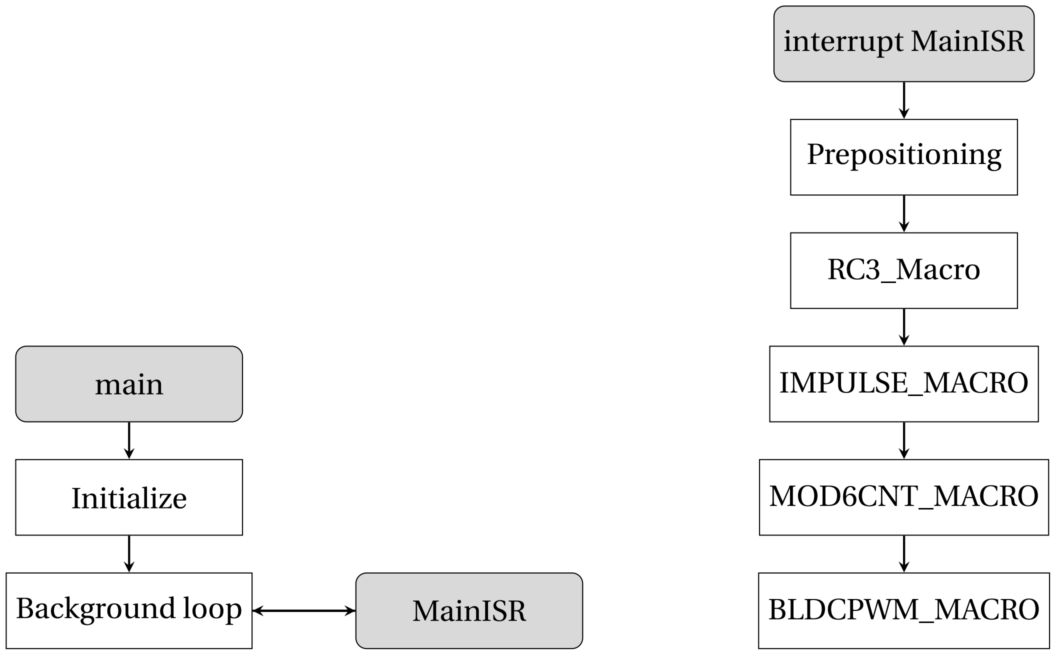


Figure 3.2: Flowchart of sensorless open loop program

Main

The main loop initializes peripherals, local variables, timers and configures GPIO. After initializing, a background loop is started.

The backround loop runs forever - until a reset event is called or an overflow of the watchdog timer has occurred. The MainISR function is run when an interrupt from a CPU-timer is detected. This timer is configured to have a period of

$$T_s = \frac{1000}{ISR_{FREQUENCY}} \tag{3.13}$$

```
ConfigCpuTimer(&CpuTimer0, 150, 1000/ISR_FREQUENCY);
```

The CPU frequency is $150MHz$, and the ISR frequency is set to $40kHz$. This means that the CPU timer has a period of

$$T_s = \frac{1000}{40} = 25\mu s \tag{3.14}$$

In other words, the MainISR function will be executed every $25\mu s$.

MainISR

Prepositioning The first step of the MainISR function is to check if the rotor is aligned correctly. If it's not aligned, the rotor will be forced to the starting position, the first step of commutation.

RC3_MACRO The second step is to choose how long the time between the commutation states is. By ramping down from a long period of time to a lower period of time, the motor will increase smoothly in speed.

```
#define RC3_MACRO(v)
 if (v.Out == v.DesiredInput)
    v.Ramp3DoneFlag = 0x7FFFFFFF;
 else
  {
    v.Ramp3DelayCount++;
    if (v.Ramp3DelayCount >= v.Ramp3Delay)
    {
        v.Out--;

        if (v.Out < v.Ramp3Min)
           v.Out = v.Ramp3Min;

        v.Ramp3DelayCount = 0;
      }
    }
#endif
```

The function increments a delay counter $v.Ramp3DelayCount$ each cycle and decrements the

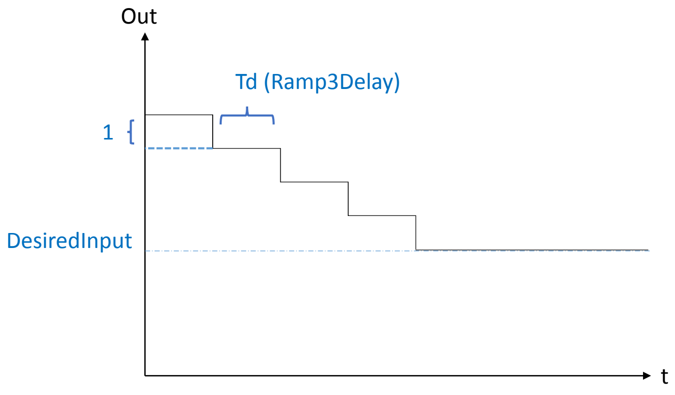


Figure 3.3: The down ramp method

IMPULSE_MACRO The third step is to generate impulses each time the period of time from the previous is reached. Each impulse signals a commutation trigger.

```
#define IMPULSE_MACRO(v)
v.Out = 0;       /* Always clear impulse output at entry*/
v.Counter++;     /* Increment the skip counter*/

if (v.Counter >= v.Period)
{
   v.Out = 0x00007FFF;
   v.Counter = 0;         /* Reset counter*/
}
```

```
#endif
```

The function increments a counter $v.Counter$ each cycle. When the counter reaches the period time $v.Period$, an impulse is generated on $v.Out$. This impulse is high until the function is

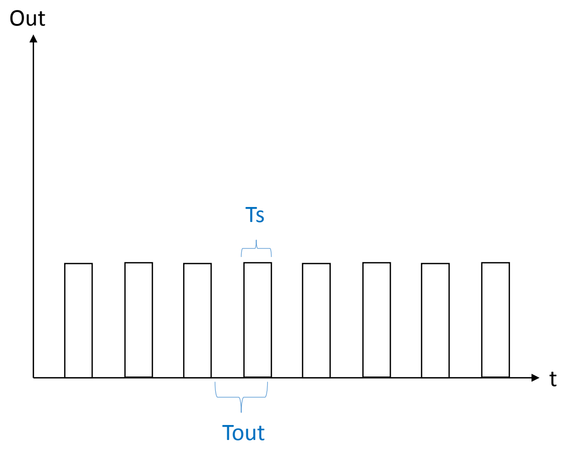


Figure 3.4: The impulse method

MOD6CNT_MACRO The fourth step is to create a counter that keeps track of the current commutation state.

```
#define MOD6CNT_MACRO(v)

 if (v.TrigInput > 0)
   {
     if (v.Counter == 5)     /* Reset the counter when it is 5 */
       v.Counter = 0;
     else
       v.Counter++;          /* Otherwise, increment by 1 */
   }
#endif
```

This function increments a counter $v.Counter$ when an impulse is detected ($v.TrigInput > 0$). The counter resets when the the counter reaches 5.

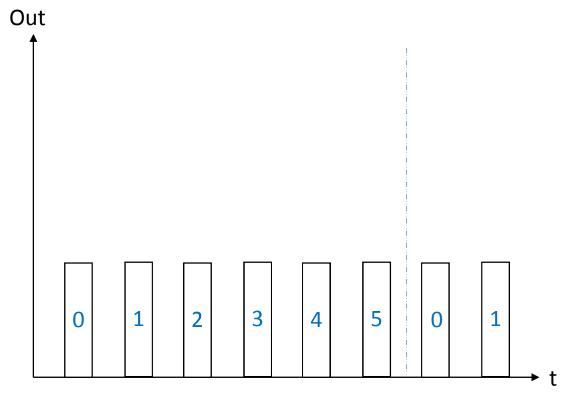


Figure 3.5: The counter method

BLDCPWM_MACRO The fifth step is to generate the PWM signals based on the current commutation state. The duty cycles of the PWM signals are determined by an input *DutyFunc*.

CmtnPointer	PWM 1A	PWM 1B	PWM 2A	PWM 2B	PWM 3A	PWM 3B	Ia	Ib	Ic
0	PWM	low	low	HIGH	low	low	+	-	0
1	PWM	low	low	low	low	HIGH	+	0	-
2	low	low	PWM	low	low	HIGH	0	+	-
3	low	HIGH	PWM	low	low	low	-	+	0
4	low	HIGH	low	low	PWM	low	-	0	+
5	low	low	low	HIGH	PWM	low	0	-	+

0. current flows to motor windings from phase A->B, de-energized phase = C
1. current flows to motor windings from phase A->C, de-energized phase = B
2. current flows to motor windings from phase B->C, de-energized phase = A
3. current flows to motor windings from phase B->A, de-energized phase = C
4. current flows to motor windings from phase C->A, de-energized phase = B
5. current flows to motor windings from phase C->B, de-energized phase = A

3.2.2 Closed loop

The sensorless driving using the closed loop method uses the same stages as the open loop in the beginning. When the motor has reached its desired speed, the loop will be closed.

6. $CMTN\_TRIG\_MACRO$ - Change commutation state based on zero-crossing method.
7. $PI\_CONTROLLER$ - PI speed regulator

CMTN_TRIG_MACRO When the loop is closed, the commutation triggering is determined by the back-EMF zero-crossing calculation instead of using the fixed commutation delay. The code is too comprehensive to list here, so the basic operation will be briefly explained.

The inputs of this function are the commutation counter and a virtual timer that keeps track of the time between each ISR cycle. The motor phase voltages, referenced to ground, are also inputs and is obtained from the ADC.

The back-EMF is calculated by subtracting the neutral voltage from the de-energized phase.

```
v.Neutral = v.Va + v.Vb + v.Vc;
...
 if (v.CmtnPointer == 0)
         v.DebugBemf = _IQmpy(_IQ(3),v.Vc) - v.Neutral;
```

Each time the commutation counter reaches 5, the function *DELAY_30DEG_MACRO* is executed. This function keeps track of the time that the commutating states 0-5 took. In other words, the time it took before a rotation of 360 electrical degrees was completed (one revolution). The 30° commutation delay is now calculated by dividing this time by $\frac{360}{30} = 12$. This number is stored in $CmtnDelay$. Note that CmtnDelay represents sample time periods and not the sample time. This means that if $CmtnDelay = 400$ the actual time delay is

$$\text{Time delay} = \text{CmtnDelay}Ts = 400Ts = 1ms$$

```
v.OldTimeStamp = v.NewTimeStamp;
v.NewTimeStamp = v.VirtualTimer;
```

```
...
v.RevPeriod = v.NewTimeStamp - v.OldTimeStamp;
...
v.CmtnDelay = v.RevPeriod/12;
```

$VirtualTimer$ increments by one each time MainISR is run (every 25μs).

For each of the commutation states, the back-EMF sign is checked.

```
else if (v.CmtnPointer == 1)
{
        ...
        if (v.DebugBemf < 0)
                v.NoiseWindowCounter = 0;
        else
           NOISE_WINDOW_CNT_MACRO(v);
        }
```

When the back-EMF toggles from negative to positive (or vice versa, depending on the rotation direction), the back-EMF zero crossing point is detected. The next commutation point can now be determined by adding the 30° commutation delay to the zero crossing time. Because sudden changes in the current and voltage can occur during the commutation (see figure 3.6), misinterpreting is possible. The solution to this problem is to introduce a noise window, which can ignore the first back-EMF readings after the commutations.

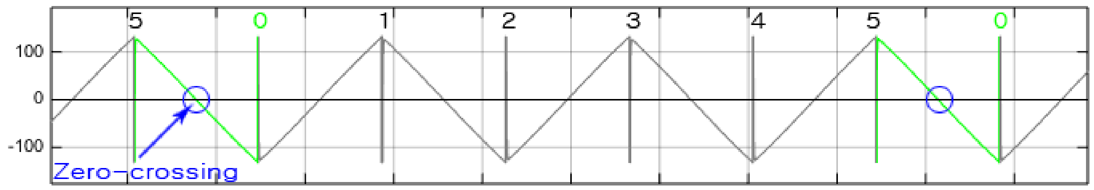


Figure 3.6: DebugBemf

PI_CONTROLLER The PI-controller is used for regulating the duty cycle based on a speed reference. If the motors load is changing, the PI controller will compensate the duty cycle to maintain the desired speed. The current motor speed is used as feedback. The c-code for the PI-controller:

```
/* proportional term */
v.up = v.Ref - v.Fbk;

/* integral term */
v.ui = (v.Out == v.v1)?(_IQmpy(v.Ki, v.up)+ v.i1) : v.i1;
v.i1 = v.ui;

/* control output */
v.v1 = _IQmpy(v.Kp, (v.up + v.ui));
v.Out= _IQsat(v.v1, v.Umax, v.Umin);
```

In Matlab:

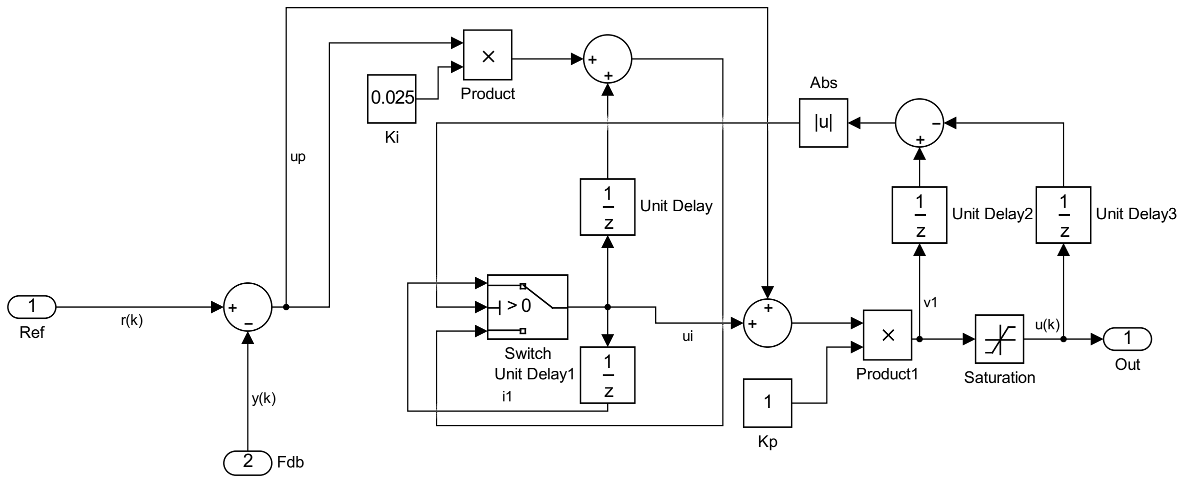


Figure 3.7: PI_CONTROLLER

Where the error signal and output is:

$$e(k) = r(k) - y(k) \tag{3.16}$$

$$v_1(k) = K_p[u_p(k) + u_i(k)] \tag{3.17}$$

Integral term:

$$u_i(k) = \begin{cases} u_1(k-1) + K_i[r(k) - y(k)] & v_1(k) = u(k) \\ u_1(k-1) & v_1(k) \neq u(k) \end{cases}$$

Saturation:

$$v_1(k) = \begin{cases} U_{max} & v_1(k) > U_{max} \\ U_{min} & v_1(k) < U_{min} \\ v_1(k) & U_{min} < v_1(k) < U_{max} \end{cases}$$

3.3 Sensored driving

Sensored driving is carried out in two different ways in this project, using hall effect sensors and using an incremental pulse encoder.

The hall effect sensors are built in the brushless DC motor, while the encoder is an external device attached to the motors shaft.

3.3.1 Hall effect sensors

The hall sensored driving is done by observing the outputs of the three hall effect sensors, which is placed around the stator.

The implementation of this code is divided in two parts, open loop and closed loop driving. The hall sensors obviously won't detect any change of rotor position when the motor is not running. This is why the open loop control is implemented, to ramp up the motor speed so that the hall sensors can give feedback.

3.3.1.1 Open loop

The open loop control is sensorless and consists of the following stages:

1. Prepositioning
2. $RC3\_MACRO$ - Ramping to desired commutation period
3. $IMPULSE\_MACRO$ - Impulse generator (create an impulse each commutation period)
4. $MOD6CNT\_MACRO$ - Determine commutation number
5. $BLDCPWM\_MACRO$ - Timing when to commutate. Commutate

This code is identical to the sensorless code in section 3.2.1. Please refer to section 3.2.1 on page 48 for details.

3.3.1.2 Closed loop

6. $HALL3$ - Change commutation state based on zero-crossing method.

7. $PI\_CONTROLLER$ - PI speed regulator

The hall effect sensors are constantly read in the program and the sum of outputs (3-bits) is stored in a variable $HallGpio$.

HALL3

This macro basically does the following:

- Read the current hall outputs
- When a change of the hall sensors output is detected, set commutation trigger $CmtnTrigHall$ high.
- Determine the next commutation state by comparing the outputs to the commutation table

When the commutation trigger is set, the macro $MOD6CNT\_MACRO$ will increment its counter.

The $HALL3$ macro uses this counter as an input, $HallMapPointer$, to determine the next commutation state. The next commutation number is now stored in the same variable, and is used as an input on the counter.

- When $HallMapPointer$ is as an input, it is defined by $MOD6\_CNT$.
- When $HallMapPointer$ is an output, it points to the next commutation state.

Please note that this description is highly simplified to give an idea on how the code works.

PI_CONTROLLER

This code is identical to the sensorless code in section 3.2.2. Please refer to section 3.2.2 on page 54 for details.

3.3.2 Incremental pulse encoder

This code is based on field-oriented control (FOC).

1. Prepositioning.
2. RC_MACRO - Ramping to desired speed.
3. RG_MACRO - Sawtooth generator. Ramping the angle from 0-360 degrees.
4. $CLARKE\_MACRO$ - Calculate clarke transformation.
5. $PARK\_MACRO$ - Calculate park transformation.
6. $IPARK\_MACRO$ - Calculate inverse park transformation.
7. $SVGENDQ\_MACRO$ - Space vector generator.
8. PI_MACRO - Regulate speed, and the inputs to $IPARK\_MACRO$.
9. PWM_MACRO - Generate PWM signals.

RC_MACRO

To avoid discontinuity in the speed reference, the RC_MACRO ramps up the desired speed from its old value.

RG_MACRO

While running the code without a speed-loop, this function creates a saw-tooth waveform that is mimicking the output of the incremental pulse encoder. The saw-tooth waveform is representing the electrical degrees of the motor.

```
/* Compute the angle rate */
v.Angle += _IQmpy(v.StepAngleMax,v.Freq);
/* Saturate the angle rate within (-1,1) */
        if (v.Angle>_IQ(1.0))
                v.Angle -= _IQ(1.0);
        else if (v.Angle<_IQ(-1.0))
                v.Angle += _IQ(1.0);
        v.Out=v.Angle;
```

CLARKE_MACRO

This code performs the Clarke transform:

$$I_\alpha = I_a$$
$$I_\beta = \frac{1}{\sqrt{3}}(I_a + 2I_b)$$

```
v.Alpha = v.As;
v.Beta = _IQmpy((v.As +_IQmpy2(v.Bs)),_IQ(0.57735026918963));
```

PARK_MACRO

This code performs the Park transform:

$$I_D = I_\alpha \cdot cos\theta + I_\beta \cdot sin\theta$$
$$I_Q = -I_\alpha \cdot sin\theta + I_\beta \cdot cos\theta$$

```
v.Ds = _IQmpy(v.Alpha,v.Cosine) + _IQmpy(v.Beta,v.Sine);
v.Qs = _IQmpy(v.Beta,v.Cosine) - _IQmpy(v.Alpha,v.Sine);
```

PARK_MACRO

This code performs the inverse Park transform:

$$I_d = I_D \cdot cos\theta - I_Q \cdot sin\theta$$
$$I_q = -I_D \cdot sin\theta + I_Q \cdot cos\theta$$

```
v.Alpha = _IQmpy(v.Ds,v.Cosine) - _IQmpy(v.Qs,v.Sine);
v.Beta  = _IQmpy(v.Qs,v.Cosine) + _IQmpy(v.Ds,v.Sine);
```

SVGEN_MACRO

This code performs the inverse Clark transform, determines the space vector section and calculates the appropriate duty ratios for the three phases.

```
v.tmp1= v.Ubeta;
v.tmp2= _IQdiv2(v.Ubeta) +(_IQmpy(_IQ(0.866),v.Ualpha));
v.tmp3= v.tmp2 - v.tmp1;

v.VecSector=3;
v.VecSector=(v.tmp2> 0)?(v.VecSector-1):v.VecSector;
v.VecSector=(v.tmp3> 0)?(v.VecSector-1):v.VecSector;
v.VecSector=(v.tmp1< 0)?(7-v.VecSector) :v.VecSector;

if (v.VecSector==1 || v.VecSector==4)
{       v.Ta= v.tmp2;
                              v.Tb= v.tmp1-v.tmp3;
        v.Tc=-v.tmp2;
}
...
```

PWM_MACRO

This code assigns the PWM signals to the transistors in the three-phase inverter:

```
/*  Mfuncx range is (-1,1)                                          */
/*  The code below changes PeriodMax*Mfuncx range ....
                                             */
/*  from (-PeriodMax,PeriodMax) to (0,PeriodMax) where HalfPerMax=
   PeriodMax/2   */

(*ePWM[ch1]).CMPA.half.CMPA = _IQmpy(m.HalfPerMax,m.MfuncC1)+ m.HalfPerMax
   ;
(*ePWM[ch2]).CMPA.half.CMPA = _IQmpy(m.HalfPerMax,m.MfuncC2)+ m.HalfPerMax
   ;
(*ePWM[ch3]).CMPA.half.CMPA = _IQmpy(m.HalfPerMax,m.MfuncC3)+ m.HalfPerMax
   ;
```

Chapter 4

Experiments

4.1 Simulation

Simulink is used for simulation of the motor. The simulink model created for this thesis is shown in figure 4.1 below, including subsystems in figure 4.2, 4.3, 4.4 and 4.5.

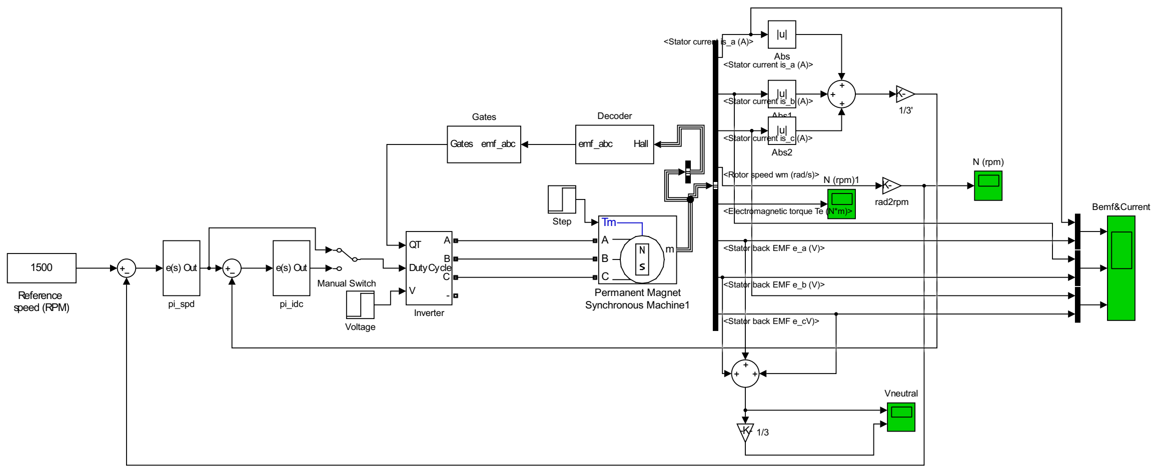


Figure 4.1: BLDC system

The system consists of two PI-controllers with feedback from speed and current, an inverter and a PMSM motor with trapezoidal back-EMF. The commutating is done by reading of Hall-sensor values that is implemented in the pre-defined motor block.

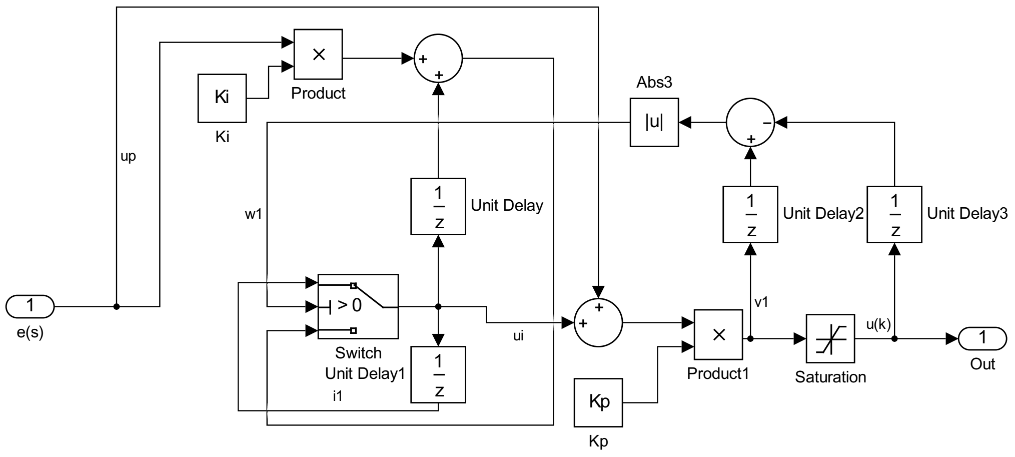


Figure 4.2: PI-controller

This PI-controller was made directly by reading the C-code from the PI-block supplied by Texas Instruments.

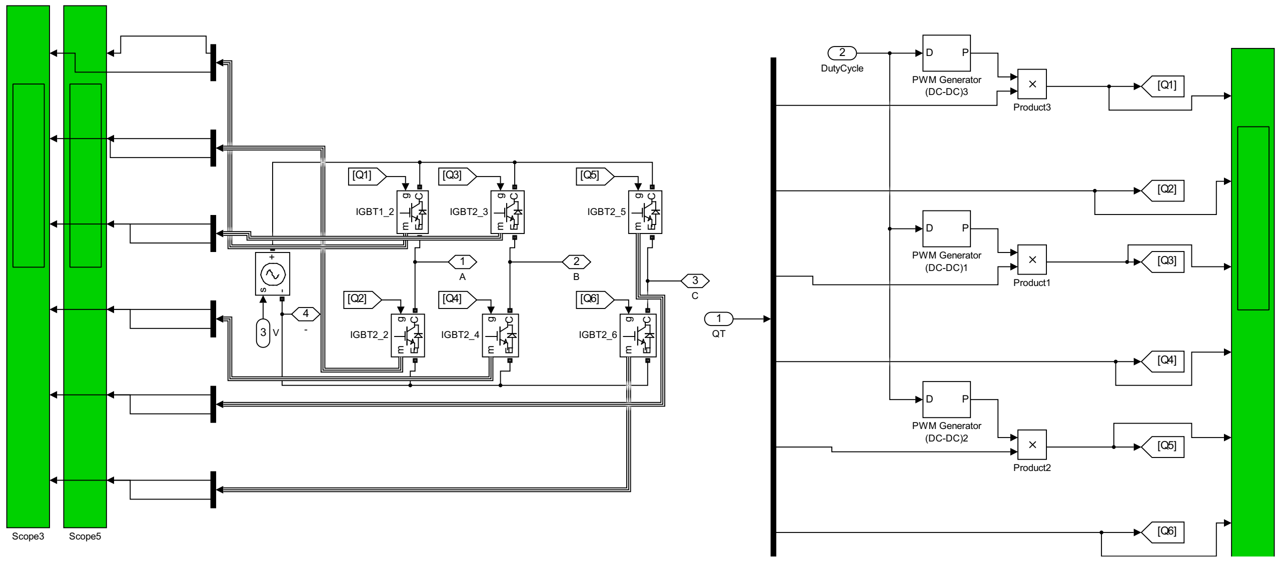
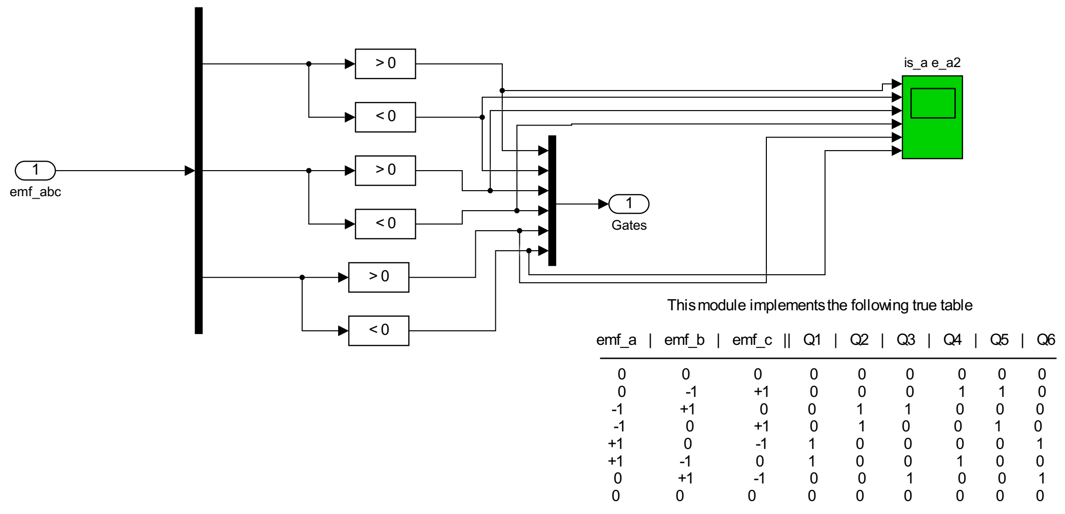


Figure 4.3: Inverter

The inverter consists of 6 MOSFETs and the PWM signals are generated by the use of Hall-sensors.

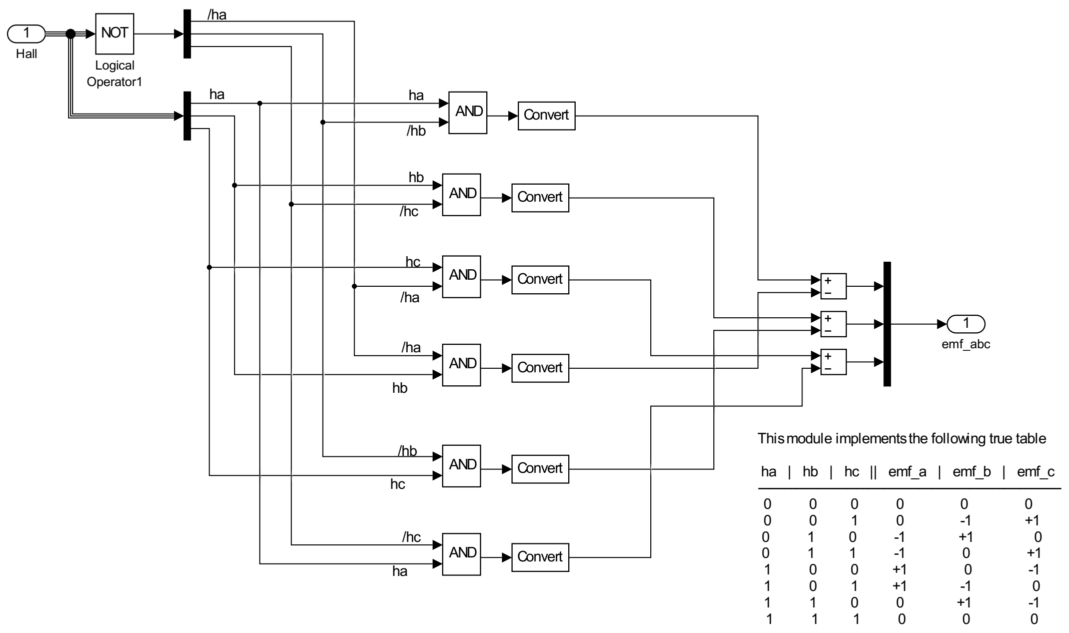


This module implements the following true table

emf_a	emf_b	emf_c	Q1	Q2	Q3	Q4	Q5	Q6
0	0	0	0	0	0	0	0	0
0	-1	+1	0	0	0	1	1	0
-1	+1	0	0	1	1	0	0	0
-1	0	+1	0	1	0	0	1	0
+1	0	-1	1	0	0	0	0	1
+1	-1	0	1	0	0	1	0	0
0	+1	-1	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0

Figure 4.4: Hall Gates

This is a pre-defined block in Simulink that does the Hall-effect logic. See table in the figure.



This module implements the following true table

ha	hb	hc	emf_a	emf_b	emf_c
0	0	0	0	0	0
0	0	1	0	-1	+1
0	1	0	-1	+1	0
0	1	1	-1	0	+1
1	0	0	+1	0	-1
1	0	1	+1	-1	0
1	1	0	0	+1	-1
1	1	1	0	0	0

Figure 4.5: Hall decoder

This is a pre-defined block in Simulink that does the Hall-effect logic. See table in the figure.

The inverter input signals are shown in figure 4.6. These signals are created using feedback from the Hall-effect sensors.

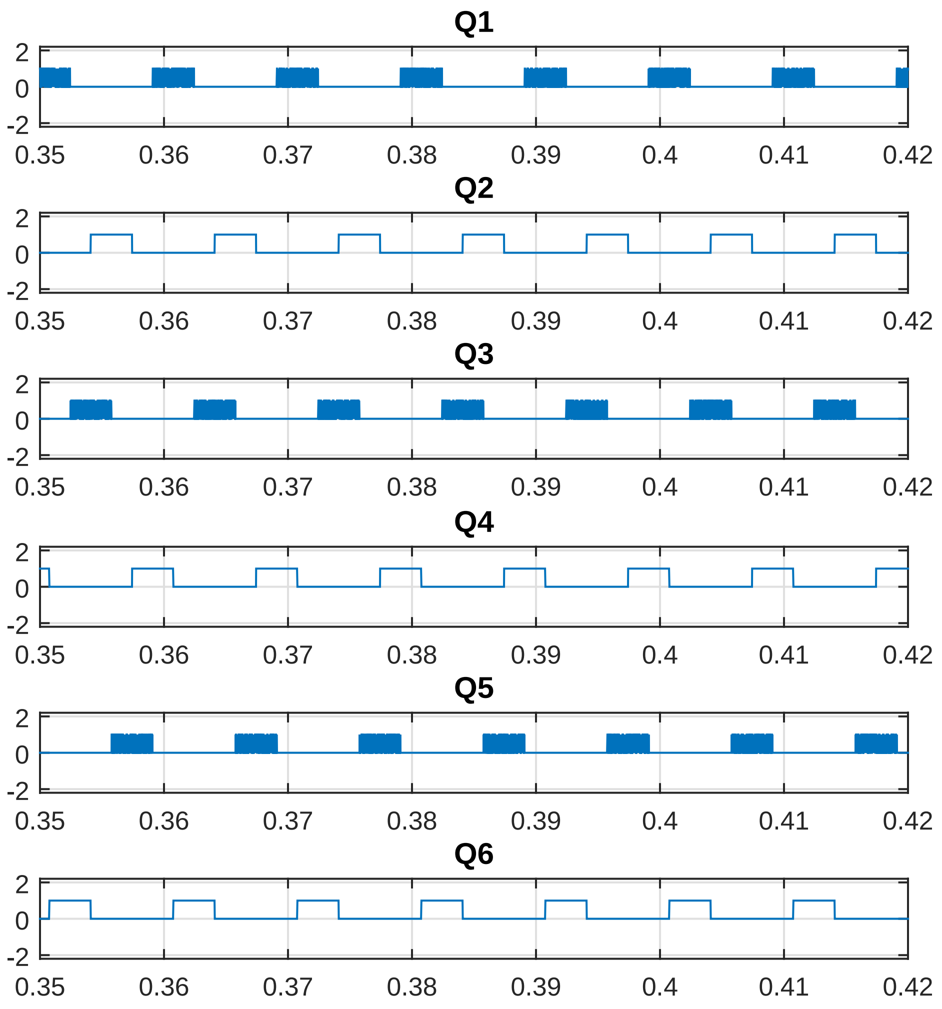


Figure 4.6: Bemf

The back-EMF signals and stator currents are shown in 4.7 below

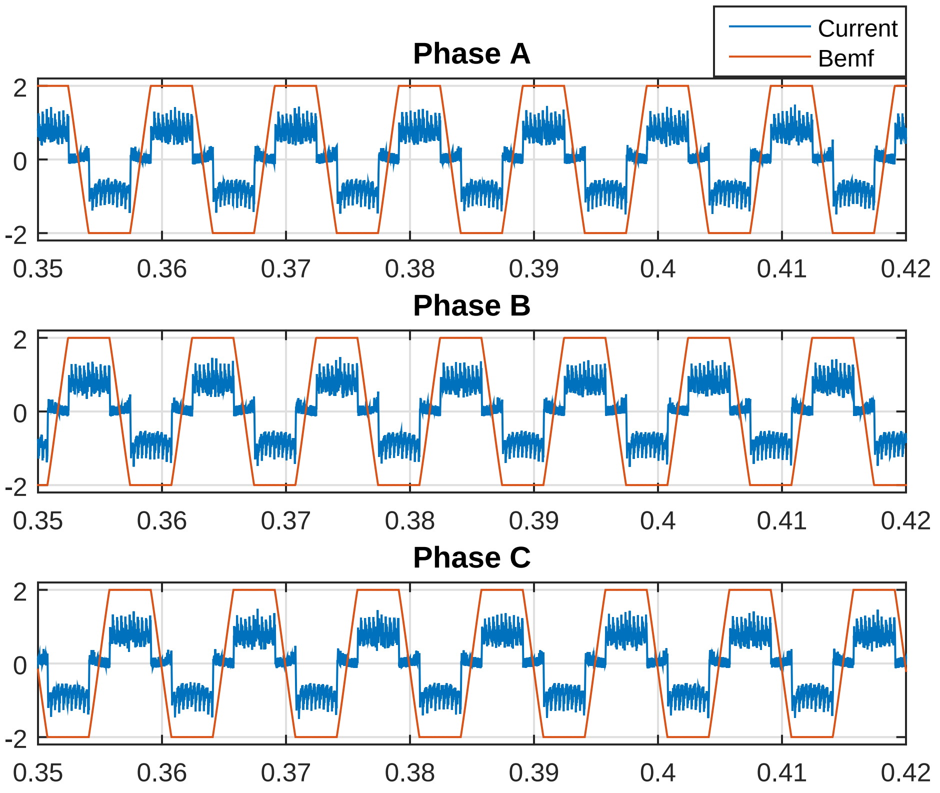


Figure 4.7: Bemf

The simulated response is shown in figure 4.8 below.

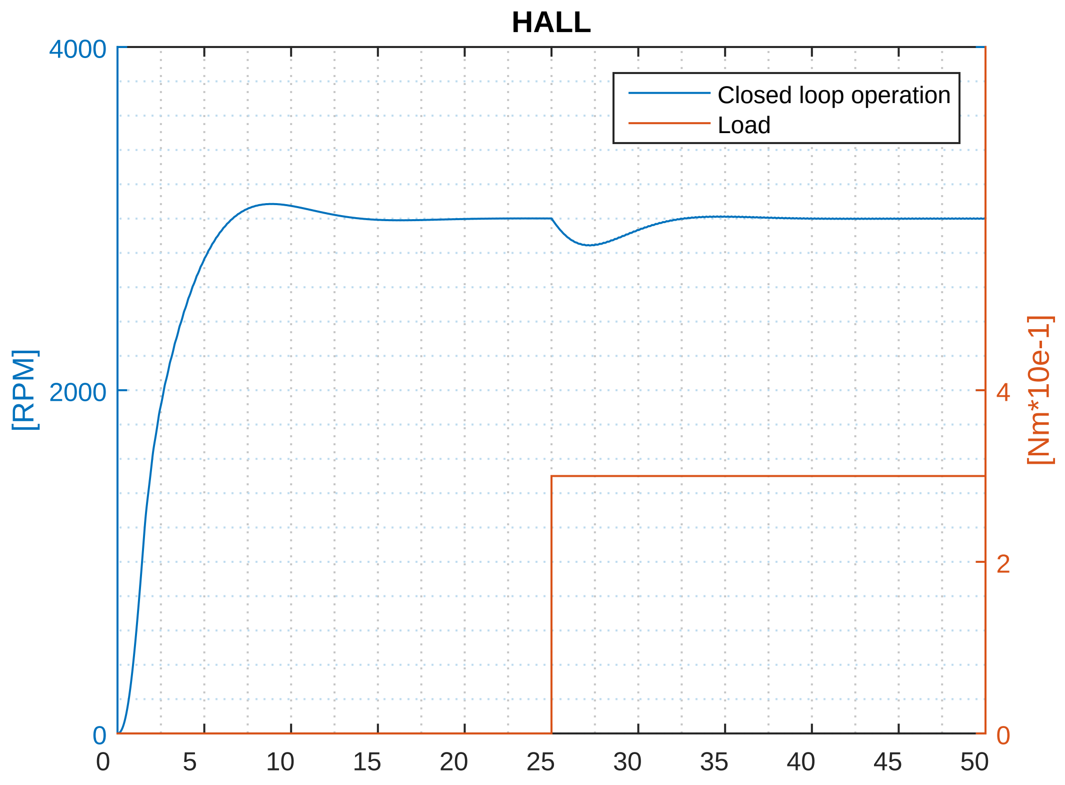


Figure 4.8: Simulated response, ref: 3000 rpm

4.2 Testing

4.2.1 Equipment

The equipment used for this project is from Texas Instruments. The *High Voltage Motor Control and PFC Kit* (see figure 4.11) is designed to be used on Piccolo and Delfine microcontrollers from Texas Instruments.

The software used for simulation is Matlab and Simulink. The software for realtime logging is CatmanEasy from HBM.

A test rig is set up for the experiments, see figure 4.9. The rig consists of:

- Optical encoder [21]
- BLDC motor [20]
- Rotary torque-transducer [22]
- Dynamo

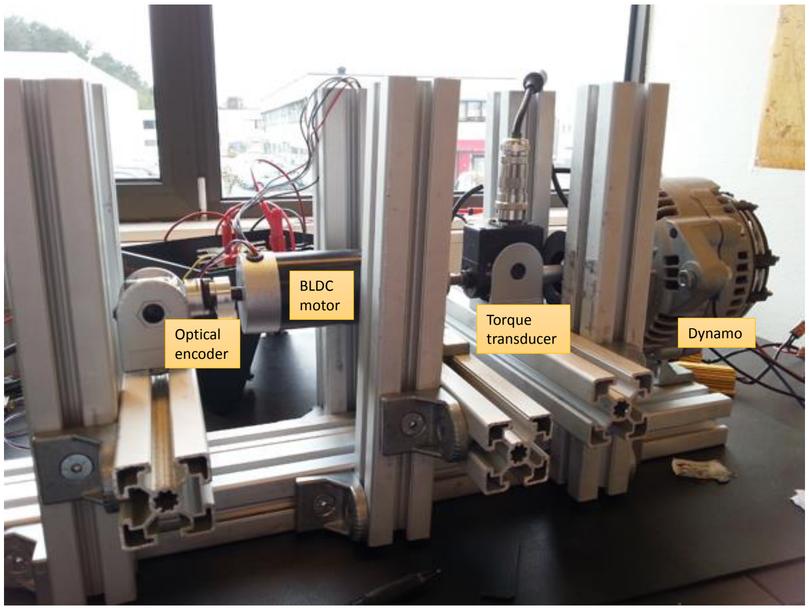


Figure 4.9: Test rig

High Voltage Motor Control and PFC Developer's Kit A motor kit (TMDSHVMTRPFCKIT)from Texas Instruments [23] is used, where all the hardware needed to control a motor is supplied. It is based around the C2000 32-bit microcontroller family, and the controlcard used for this thesis is the C2000 Piccolo TMS320F28035 MCU [24]. Software used for this controller is a free unrestricted version of Code Composer Studio integrated development environment (IDE).

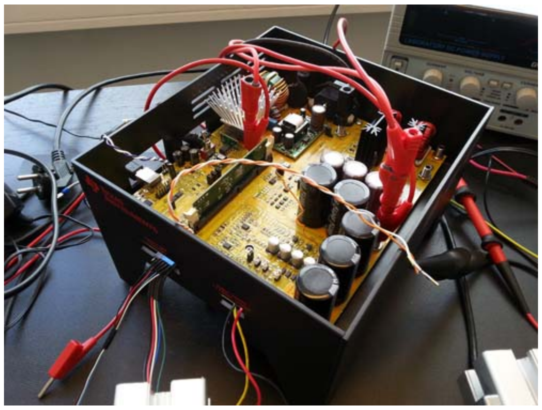


Figure 4.10: High Voltage Motor Control and PFC (R1.1) Kit [25]

Microcontroller The microcontroller used is TMS320F28035 from Texas Instruments. It has a 32-bit floating point CPU and a cycle time of 16.67 ns (60 MHz). More information can be found in the datasheet [24].

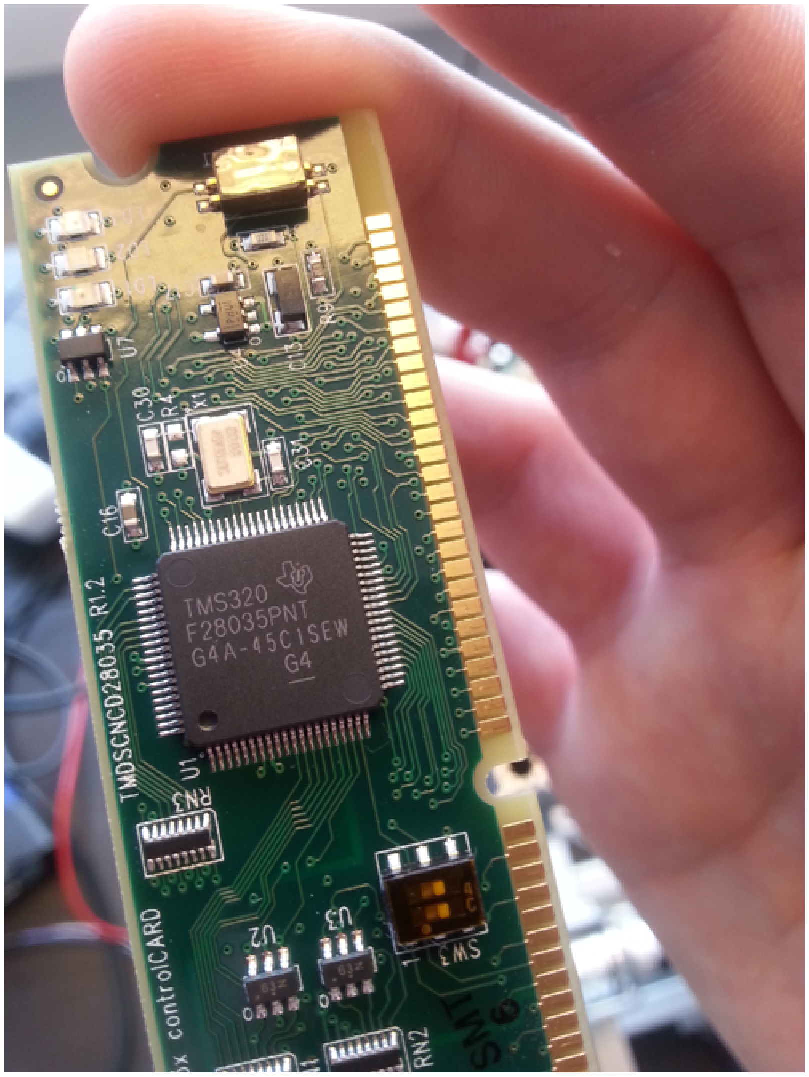


Figure 4.11: TMS320F28035 [24]

*

Motor The motor used in this thesis is a high voltage BLDC motor from Anaheim Automation. Hall-effect sensors are integrated in this motor. The motor parameters can be found in table 3.1 in section 3.

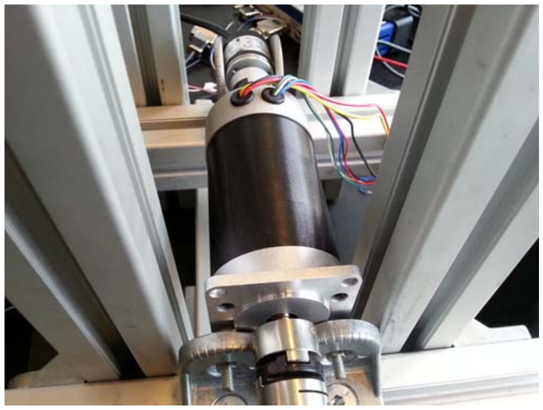


Figure 4.12: HVBLDCMTR [6]

*

AC-transformer The input voltage is supplied by an AC-transformer, shown in figure 4.12 below.

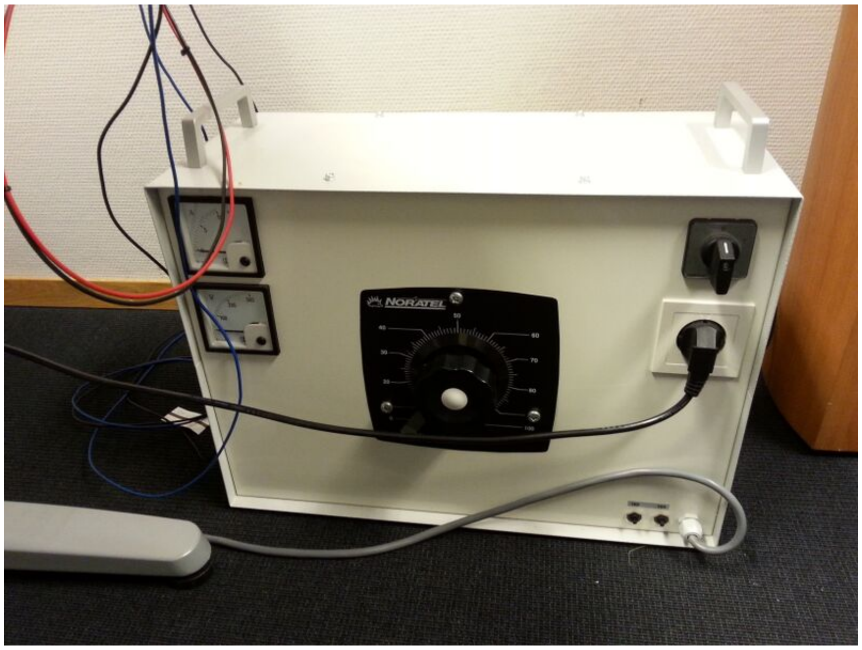


Figure 4.13: AC-transformer

*

Lab-supply A lab-supply is used to feed the coils in the dynamo with currents. This will generate a constant magnetic field that will slow down the motor connected to the dynamo.

Figure 4.14: Lab supply

*

Oscilloscope The oscilloscope used is a Tektronix MSO 2014 Mixed Signal Oscilloscope, which is rated for 100MHz.

Figure 4.15: Oscilloscope

*

Dynamo The dynamo used is an abandoned dynamo from a previous car owner.

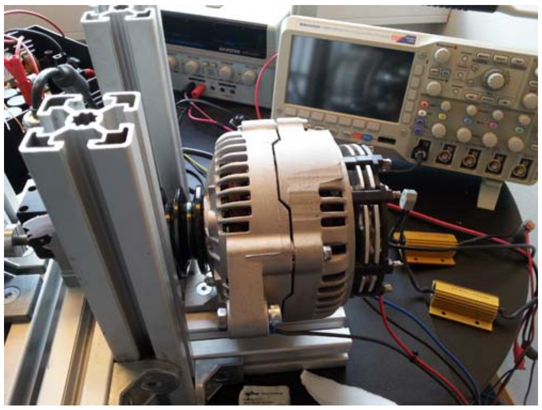


Figure 4.16: Dynamo

*

Torque transducer The torque transducer used is rated for 20Nm and is supplied by Norbar [22].

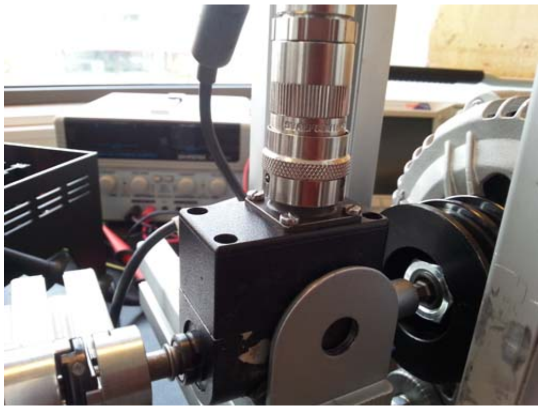


Figure 4.17: Torque transducer

*

Optical encoder The optical encoder used is from Baumer Electronics and is rated for 12000 rpm with 500 pulses per revolution [21].

Figure 4.18: Optical encoder: BDK16.05a [21]

*

Summarized The figure 4.11 below shows the main components of the hardware. The input AC voltage is supplied by the AC-transformer and is converted into DC voltage. The DC voltage is used to power the MCU including giving voltage into the three-phase inverter. The PWM signals are sent from the MCU and used as inputs on the transistors on the inverter. The transistors outputs are connected to the motor input phases.

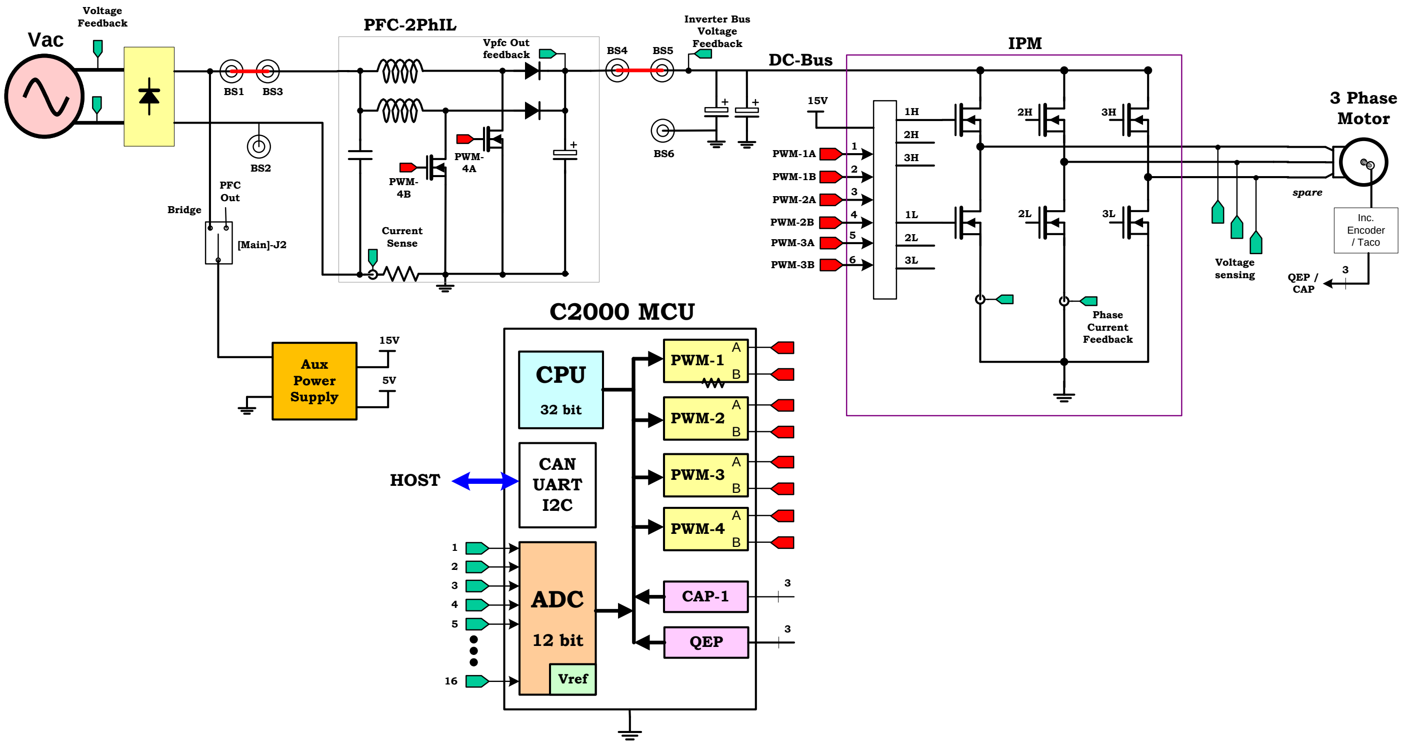


Figure 4.19: An overview of the toolkit [23]

4.2.2 Finding the moment of inertia

The system model is derived in chapter 3. The inertia seen by the motor is found by an experiment. A weight is attached to the shaft of the dynamo by using a string. The mass of the weight is known, as well as the radius of the shaft.

The weight is dropped and is free falling. The velocity is logged and shown i figure 4.20. The following parameters are measured:

$$m = 0.28 \text{ [kg]}$$
$$r = 0.0235 \text{ [m]}$$
$$\alpha = 10.4858 \text{ [rad}/s^2\text{]}$$

Where m is the mass of the weight, r is the radius of the shaft where the string is attached and α is the angular acceleration.

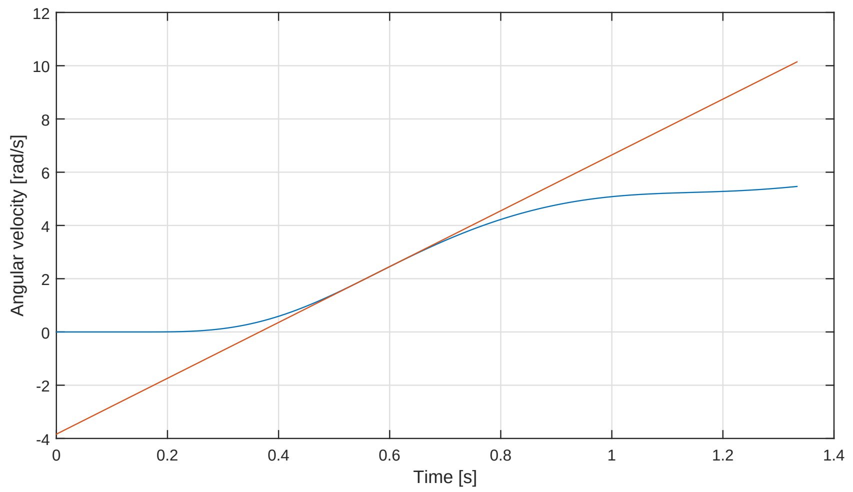


Figure 4.20: Angular velocity of weight. The red tangent represents the angular acceleration.

From Newton II. law we get following equation for figure 4.21:

$$S - m \cdot g = -m \cdot a \tag{4.1}$$

Moment of inertia for the shaft:

$$S \cdot r = J \cdot \alpha \tag{4.2}$$

The relationship between α and the acceleration of the string is:

$$a = \alpha \cdot r \tag{4.3}$$

Using equation (4.2.2) we get:

$$a = \alpha \cdot r = 10.4858 \cdot 0.0235 \approx 0.2464 \text{ [m/}s^2\text{]} \tag{4.4}$$

Using equation 4.2.2:

$$S - m \cdot g = -m \cdot a$$
$$S = m \cdot (g - a) = 0.28 \cdot (9.81 - 0.2464) \approx 2.6678 \text{ [kg} \cdot \text{m / } s^2\text{]}$$

The inertia can be calculated from equation 4.2.2 :

$$S \cdot r = J \cdot \alpha$$
$$J = \frac{S \cdot r}{\alpha} = \frac{2.6678 \cdot 0.0235}{10.4858} \approx 0.006 \text{ [kg}\cdot m^2\text{]}$$

The moment of inertia of the motor is

$$J_m = J \cdot K_t = 0.006 \cdot 0.33 \approx 0.002 \text{ [kg } \cdot m^2\text{]} \tag{4.5}$$

Where K_t is the torque constant, found in the datasheet of the motor.

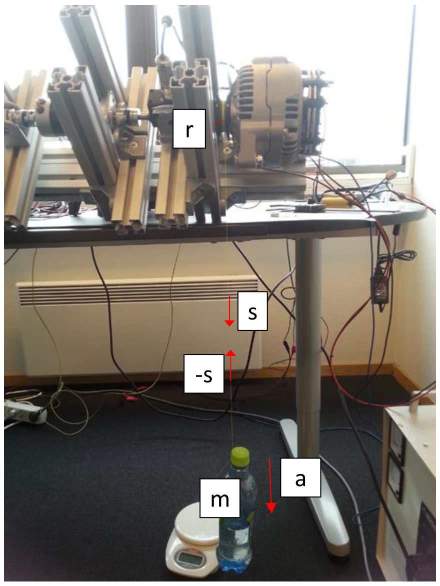


Figure 4.21: Inertia weight experiment.

4.2.3 Step responses in open-loop

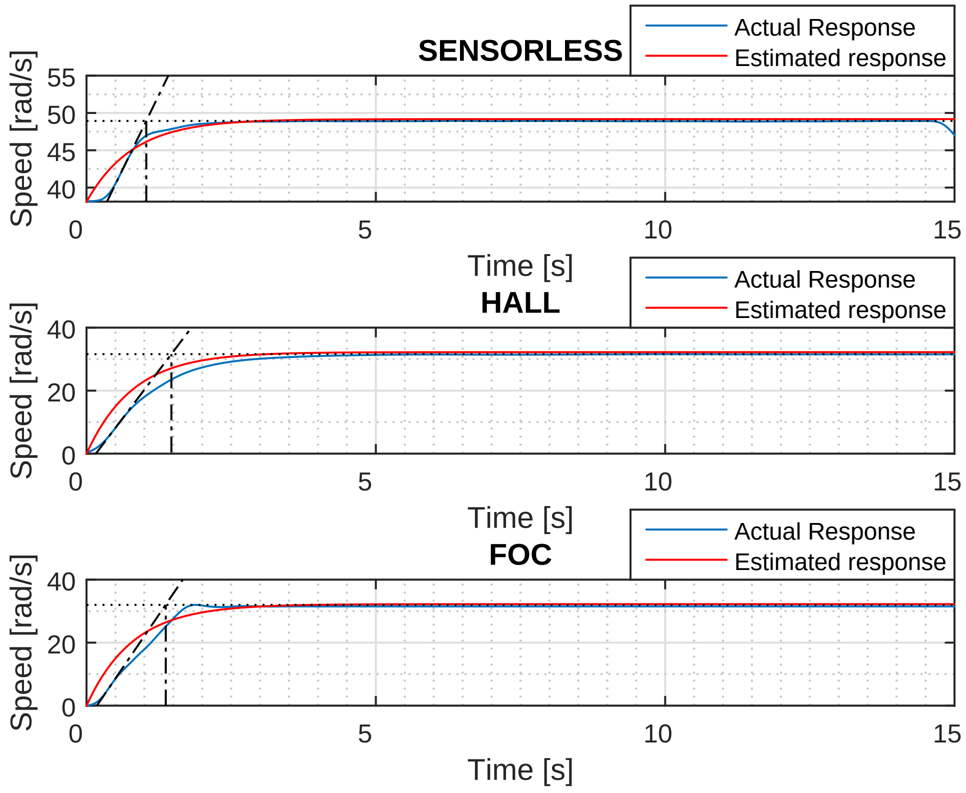


Figure 4.22: Step responses. Estimated response: overdamped system. Sensorless and hall response: critical damped system. FOC response: underdamped system

Estimated response:

$$G_{\text{est}}(s) = \frac{4.028}{0.8097 \cdot 10^{-3} s^2 + 0.7977 s + 1} \tag{4.6}$$

The step responses in figure 4.22 above is achieved in open loop driving. The steps in voltage reference are not available during logging, therefore the idle time τ is assumed to be $\tau \approx 0$. The experiments are done with the testing rig, which means that the moment of inertia described in section 4.2.2 applies here. The step responses are analysed:

4.2.3.1 Sensorless controller parameters

The controller parameters can be found using Ziegler open-loop method. By finding the parameters shown in figure 4.23 below, we can calculate the parameters for the desired controller using the formulas in table 4.1.

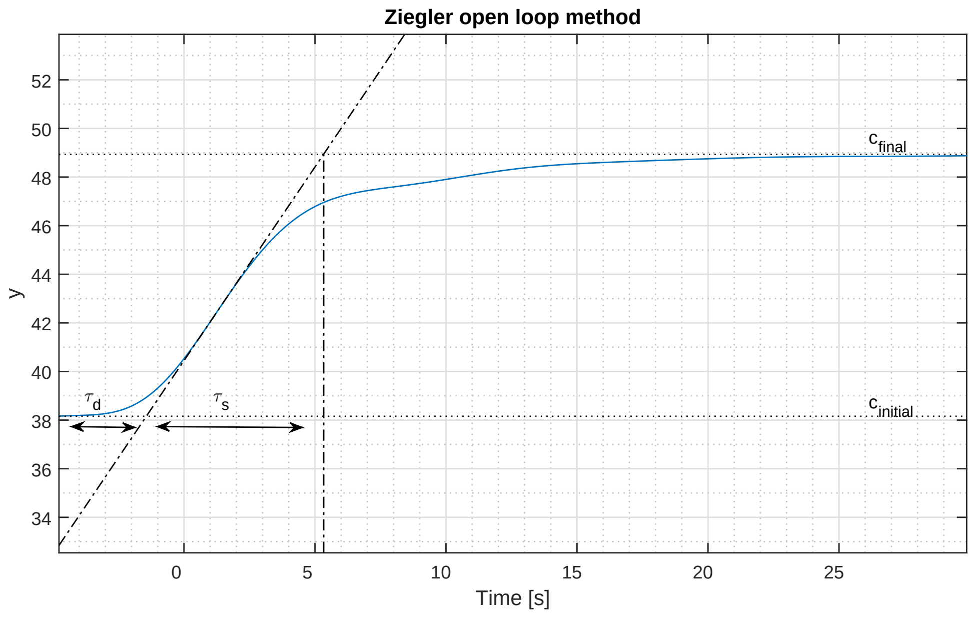


Figure 4.23: Ziegler open-loop method

The parameters from this step response in figure 4.22 are calculated in Matlab.

Amplitude in step: $\Delta u = 2.7467\ V$

We read from figure 4.22 by calculating parameters using figure 4.23:

Initial value: $c_{\text{initial}} \approx 38.1585\ \text{rad/s}$

Final value: $c_{\text{final}} \approx 48.9329\ \text{rad/s}$

Amplitude process variable: $\Delta c = c_{\text{final}} - c_{\text{initial}} = 48.9329 - 38.1585\ \text{rad/s} = 10.7744\ \text{rad/s}$

The time constants are $\tau_d = 0.3567$ s and $\tau_s = 0.6767$ s The gain can be found from $c_{\text{initial}} + \Delta u \cdot K = c_{\text{final}}$

Which gives $K = \frac{c_{\text{final}} - c_{\text{initial}}}{\Delta u} = \frac{\Delta c}{\Delta u} = \frac{10.7744\ \text{rad/s}}{2.7467\ \text{V}} = 3.923\ \frac{\text{rad/s}}{\text{V}}$

Controller type	Gain	Reset	Derivative
P	$\frac{\tau_s \cdot \Delta u}{\tau_d \cdot \Delta c}$	-	-
PI	$\frac{0.9 \cdot \tau_s \cdot \Delta u}{\tau_d \cdot \Delta c}$	$3.3 \cdot \tau_d$	-
PID	$\frac{1.2 \cdot \tau_s \cdot \Delta u}{\tau_d \cdot \Delta c}$	$2.0 \cdot \tau_d$	$0.5 \cdot \tau_d$

Table 4.1: Controller parameters

As the motor controller can be exposed to temperatures up to 150°C and the ambient pressure can be as high as 7500 PSI (the electronics does not experience more than 1 bar), the measurements may be exposed to noise. The sampling frequency for the controller is also very high (40kHz). Taken this in consideration, the controller type for this motor is chosen to be a PI-controller. The PID-controller is known to be very sensitive to measurement noise.
PI-parameters from table 4.1 gives:

$$K_p = \frac{0.9 \cdot \tau_s \cdot \Delta u}{\tau_d \cdot \Delta c} = \frac{0.9 \cdot 0.6767 \cdot 2.7467}{0.3567 \cdot 10.774} = 0.4353$$
$$T_i = 3.3 \cdot \tau_d = 3.3 \cdot 0.3567 = 1.1771$$

Where

$$K_i = \frac{K_p}{T_i} = \frac{0.4353}{1.1771} = 0.3698$$

As the controller is implemented in a microcontroller it is not operating in continuous time, but in discrete time. This is why the K_i parameter has to be divided with the sampling time T_s:

$$K_{id} = \frac{T_s}{0.3698}$$

Where $T_s = \frac{1}{40000}$.

4.2.3.2 Hall-effect controller parameters

The parameters from this step response in figure 4.22 are calculated in Matlab.

Amplitude in step: $\Delta u = 8\ V$

We read from figure 4.22 by calculating parameters using figure 4.23:

Initial value: $c_{\text{initial}} = 0$ rad/s

Final value: $c_{\text{final}} = 31.5652$ rad/s

Amplitude process variable: $\Delta c = c_{\text{final}} - c_{\text{initial}} = 31.5652 - 0$ rad/s $= 31.5652$ rad/s

The time constants are: $\tau_d = 0.2967$ s and $\tau_s = 1.3033$ s

The gain can be found from $c_{\text{initial}} + \Delta u \cdot K = c_{\text{final}}$

Which gives $K = \frac{c_{\text{final}} - c_{\text{initial}}}{\Delta u} = \frac{\Delta c}{\Delta u} = \frac{31.5652\ \text{rad/s}}{8\ \text{V}} = 3.9457\ \frac{\text{rad/s}}{\text{V}}$

In section 4.2.3.1 the controller for this motor was chosen to be a PI-controller. We get following PI-parameters from table 4.1:

$$K_p = \frac{0.9 \cdot \tau_s \cdot \Delta u}{\tau_d \cdot \Delta u} = \frac{0.9 \cdot 1.3033 \cdot 8}{0.2967 \cdot 31.5652} = 1.002$$

$$T_i = 3.3 \cdot \tau_d = 3.3 \cdot 0.2967 = 0.9791$$

Where

$$K_i = \frac{K_p}{T_i} = \frac{1.002}{0.9791} = 1.0236$$

Where K_{id} is:

$$K_{id} = \frac{T_s}{1.0236}$$

See section 4.2.3.1 for explanation.

4.2.3.3 FOC controller parameters

The parameters from this step response in figure 4.22 are calculated in Matlab.

Current-loop

Amplitude in step: $\Delta u = 8\ V$

We read from figure 4.22 by calculating parameters using figure 4.23:

Initial value: $c_{\text{initial}} = 0$ rad/s

Final value: $c_{\text{final}} = 31.9914$ rad/s

Amplitude process variable: $\Delta c = c_{\text{final}} - c_{\text{initial}} = 31.9914 - 0$ rad/s $= 31.9914$ rad/s

The time constants are: $\tau_d = 0.2900$ s and $\tau_s = 1.1900$ s

The gain can be found from $c_{\text{initial}} + \Delta u \cdot K = c_{\text{final}}$

Which gives $K = \frac{c_{\text{final}} - c_{\text{initial}}}{\Delta u} = \frac{\Delta c}{\Delta u} = \frac{31.9914\,\text{rad/s}}{8\,\text{V}} = 3.9989\,\frac{\text{rad/s}}{\text{V}}$

In section 4.2.3.1 the controller for this motor was chosen to be a PI-controller. We get following PI-parameters from table 4.1:

$$K_p = \frac{0.9 \cdot \tau_s \cdot \Delta u}{\tau_d \cdot \Delta u} = \frac{0.9 \cdot 1.1900 \cdot 8}{0.2900 \cdot 31.9914} = 0.9236$$

$$T_i = 3.3 \cdot \tau_d = 3.3 \cdot 0.2900 = 0.9570$$

Where

$$K_i = \frac{K_p}{T_i} = \frac{1.002}{0.9791} = 0.9651$$

Where K_{id} is:

$$K_{id} = \frac{T_s}{0.9651}$$

See section 4.2.3.1 for explanation.

Speed-loop

From section 4.2.4.3 we find out that one more PI-controller is needed in cascade system. The parameters for this controller is found by the step response in closed-loop driving.

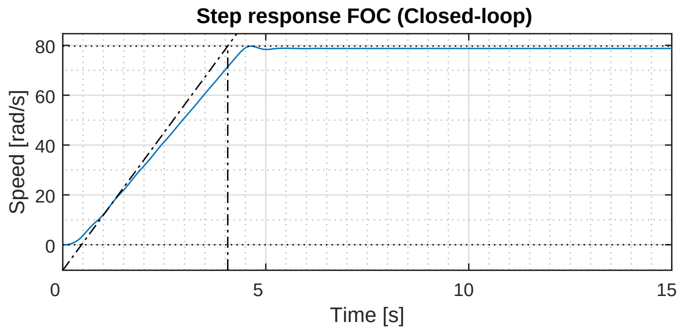


Figure 4.24: Step response from current-loop

Amplitude in step: $\Delta u = 20\ V$

We read from figure 4.22 by calculating parameters using figure 4.23:

Initial value: $c_{\text{initial}} = 0\,\text{rad/s}$

Final value: $c_{\text{final}} = 79.7020\,\text{rad/s}$

Amplitude process variable: $\Delta c = c_{\text{final}} - c_{\text{initial}} = 31.9914 - 0\,\text{rad/s} = 79.7020\,\text{rad/s}$

The time constants are: $\tau_d = 0.4633\,\text{s}$ and $\tau_s = 3.600\,\text{s}$

The gain can be found from $c_{\text{initial}} + \Delta u \cdot K = c_{\text{final}}$

Which gives $K = \frac{c_{\text{final}} - c_{\text{initial}}}{\Delta u} = \frac{\Delta c}{\Delta u} = \frac{79.7020\,\text{rad/s}}{20\,\text{V}} = 3.985\,\frac{\text{rad/s}}{\text{V}}$

In section 4.2.3.1 the controller for this motor was chosen to be a PI-controller. We get following PI-parameters from table 4.1:

$$K_p = \frac{0.9 \cdot \tau_s \cdot \Delta u}{\tau_d \cdot \Delta u} = \frac{0.9 \cdot 3.600 \cdot 20}{0.4633 \cdot 79.7020} = 1.7546$$

$$T_i = 3.3 \cdot \tau_d = 3.3 \cdot 0.4633 = 1.5289$$

Where

$$K_i = \frac{K_p}{T_i} = \frac{1.7546}{1.5289} = 1.1476$$

We want the primary-loop(speed-loop) to see a faster process than the secondary-loop(current loop). This would allow the secondary controller to compensate the current *before* the primary controller starts to regulating the torque. By letting the current controllers act 10 times faster than the speed controller, we get:

$$K_{id} = \frac{10 \cdot T_s}{0.9651}$$

See section 4.2.3.1 for explanation of the T_s parameter.

Sources of error

By comparing $G_{\text{uc}}(s)$ to the estimated process $G_{\text{est}}(s)$ in equation (4.2.3) we can see that they correlate quite good. The major difference is the s^2 term which is much higher. As the motor is connected to a dynamo as well as a torque transducer and an optical encoder during these tests, some sources of error will affect the results. The biggest error comes from the slack between the motor and the dynamo. This is because of the torque transducer that is old and worn down. From Farraday's law, an electromotive force is induced in the dynamo, just like the motor. This basically means that when the motor increases in speed, the EMF in the dynamo is increased. When the EMF in the dynamo is increased, the load that the motor sees is increased. The EMF was measured during the experiments and the EMF induced is so low that it is negligible. Keep in mind that the friction constant is also neglected in the estimated model.

The exact time of step input from the step responses was not available for logging during the experiments. The time intervals in figure 4.22 is fixed so that the step input is toggled in $t = 0$. They should be close to actual values based on observations.

The process response time T_r and time delay τ is $\tau \approx 0$ is $T_r \approx 0$ because of the high sampling interval of the system. The sampling interval T_s is $2.5 \cdot 10^{-5}$ s which means that system is sampled 40 000 times per second.

4.2.4 Closed-loop responses

The PI-parameters calculated from section 4.2.3.1 to 4.2.3.3 are implemented in the controllers and the responses in closed-loop operation are shown in figure 4.25 below.

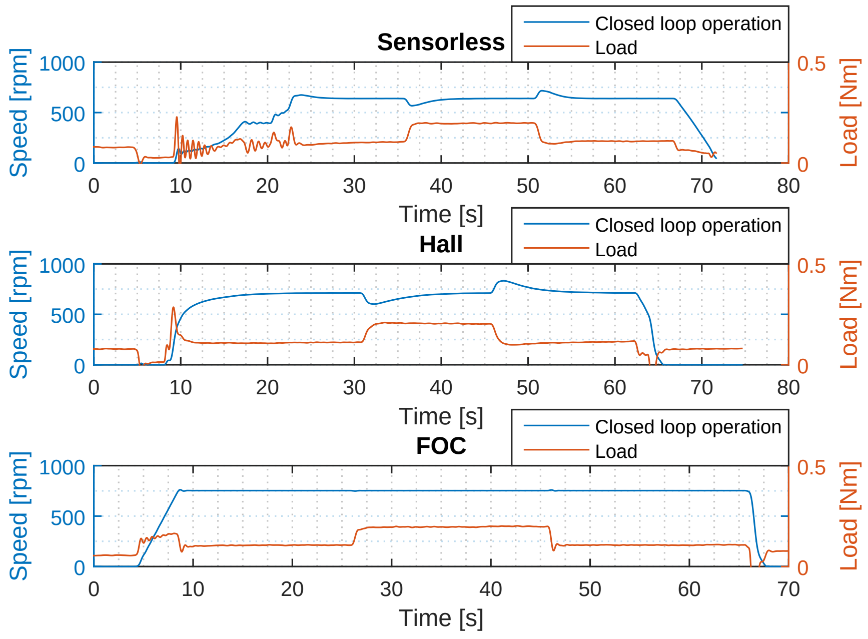


Figure 4.25: Closed-loop responses

The red curves are showing the torque seen by the torque transducer. The dynamo was fed with a current of 1 ampere from a labsupply in the intervals given in table 4.2. The change of load outside these time intervals occurs because of the change in angular velocity. See equation (4.2.4) below.

$$\tau_{\text{net}} = \frac{dL}{dt} = \frac{d(I\omega)}{dt} = I\frac{d\omega}{dt} = I\alpha \qquad (4.7)$$

Where L is angular momentum, I is the moment of inertia, ω is the angular velocity and α is the angular acceleration.

As the torque transducer used is worn down and has some mechanical loosenesses, the curves does not show the actual torque values during the experiments. For an ideal torque transducer, the load should be estimated to be $\tau_{\text{net}} \approx 0$ when the angular velocity is constant and when there is no step in the load.

Sensorless		**Hall**		**FOC**	
Start	End	Start	End	Start	End
36 s	51 s	31 s	46 s	26 s	46 s

Table 4.2: Load intervals

4.2.4.1 Sensorless closed-loop response

As the start-up sequence for the sensorless code isn't very optimal, the motor needs around 13 seconds in open-loop to be stable before closing the loop. The inefficient ramp-up algorithm that is implemented allows the commutation period to be faster than what is optimal for the motor during a high-load start-up. This basically means that a lag between electrical rounds and mechanical rounds occur. This is shown quite well in the oscillating load curve from 10s to 23s in figure 4.25.

Disregarded the start-up time of the motor, the speed control is looking good. The speed does not drop too much when the load is put on and the the response time of the controller is fast. There is no oscillations created from the controller.

The tiny change of speed in $t = 5$ is from the aligning of the rotor.

By implementing the start routine presented in [26], a better start-up of the motor is achieved. An attempt to tune in the parameters is shown in figure 4.26 below. The controller goes from open-loop to closed-loop in around $t = 18sek$.

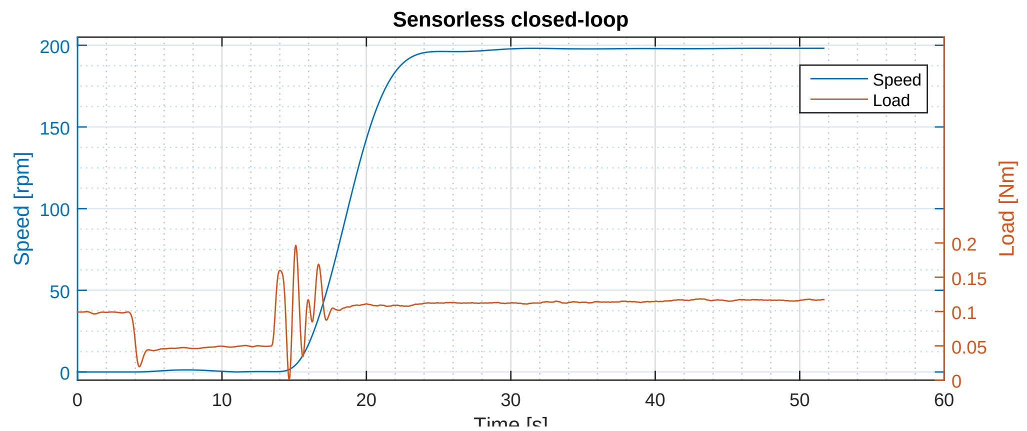


Figure 4.26: Sensorless start-up phase. Step in reference in around t=14sek

4.2.4.2 Hall-effect closed-loop response

The start-up sequence from this method is fast and stable. The step in reference is initiated around $t = 8s$ and switches from open-loop to closed loop in around $t = 8.5s$.
The PI-parameters for this method gives more or less the same response as the sensorless response described in the previous section.

4.2.4.3 FOC closed-loop response

The closed-loop response is made by using PI-regulators for the desired flux and torque components of the stator current, where $I_d$ref = 0 and $I_q$ref = 0.1. The start-up sequence of this method is quite unique. As the torque component I_q is regulated to be a certain value, $I_q$ref = 0.1, the change of speed reference makes the angular velocity change very linearly.

The step in load does not seem to affect speed, unless we zoom in on the figure (see figure 4.36 below), which is also because of the FOC method is based on always achieving maximum torque per am. The response shows a small overshoot and a very fast settling time after the step in load.

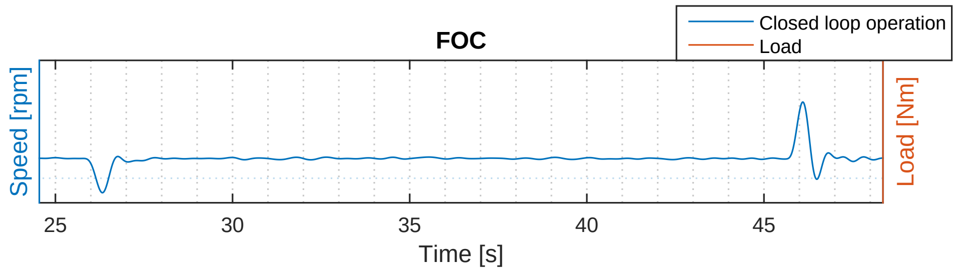


Figure 4.27: Zoom-in of FOC closed-loop response

The response may seem very effective, but if the motor load is higher than the referenced torque, the motor will stall and die out. If the motor load is much lower than the referenced torque, the motor will use much more power than is needed and may start to stall or oscillate. Three experiments are conducted to see how the responses is with different values of the desired torque reference $I_q$ref. This experiments are done using only current loop. The reference in speed is 750 rpm and the load is ramping up until the motor stalls and die out. See figures below:

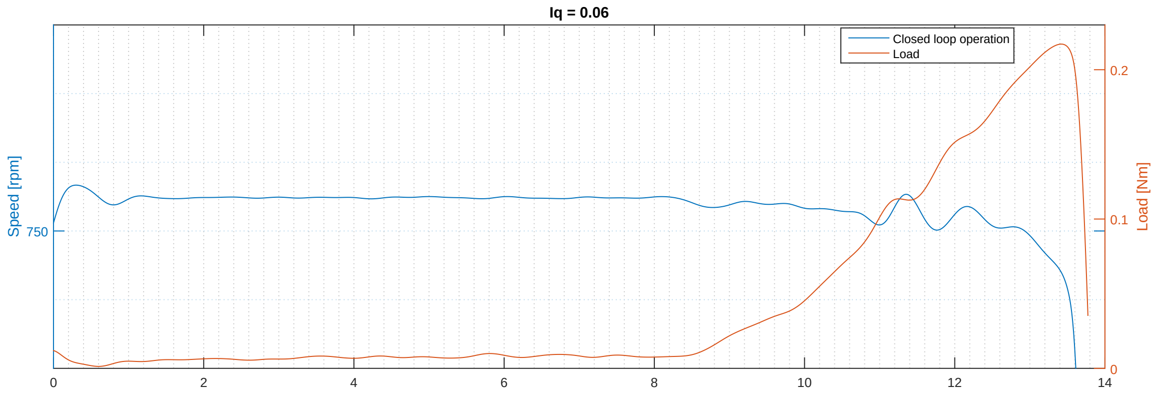


Figure 4.28: Id=0, Iq = 0.06

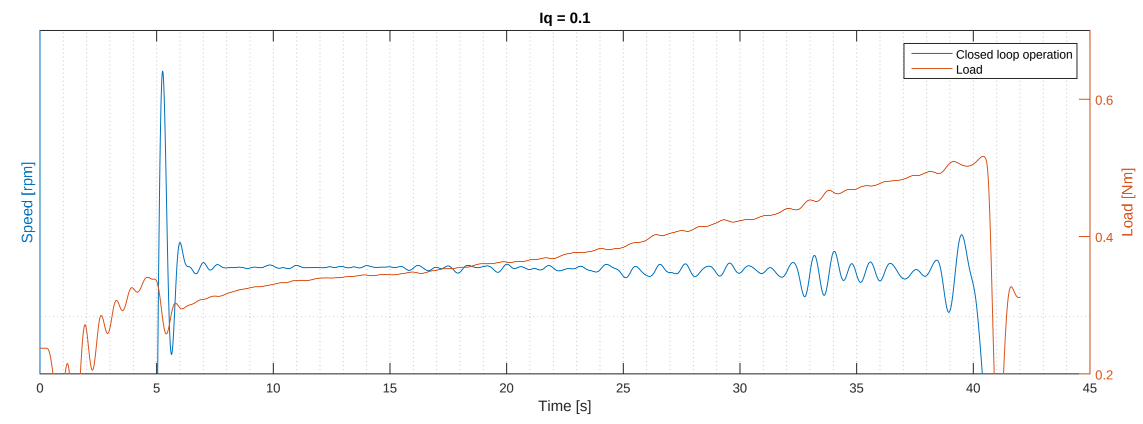


Figure 4.29: Id=0, Iq = 0.1

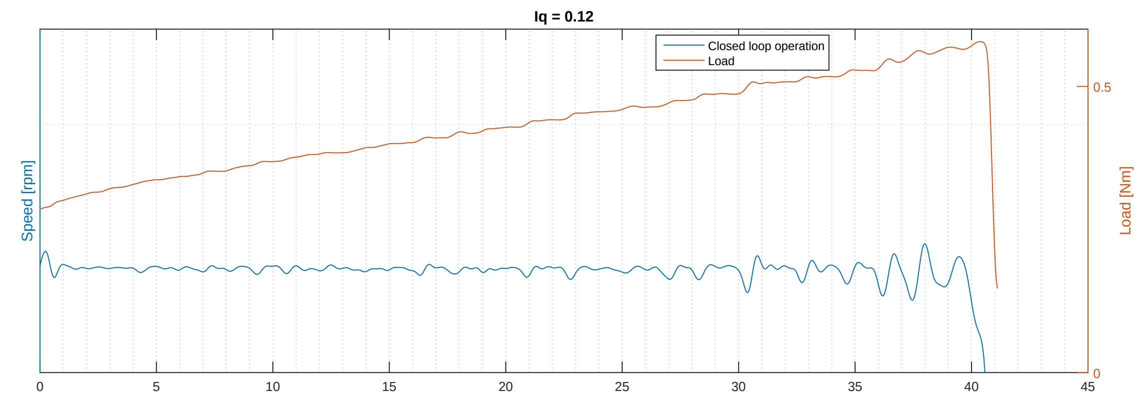


Figure 4.30: Id=0, Iq = 0.12

It is obvious that when the load gets too high and the reference torque isn't sufficient enough to oppose the load, the motor get problems. The motor couldn't handle more than 0.18Nm with $I_q = 0$. The motor drives well with $I_q = 0.1$ and load $T_L \approx 0.30$, but starts oscillating as the load increases. The motor drives well with $I_q = 0.12$ and load $T_L \approx 0.35$, but starts oscillating as the load increases. By increasing the torque I_q when the load increases, we should get rid of most of the oscillations. We can do this by adding one more PI-controller, used for speed/torque regulating, in cascade with the current controllers.

Note that the rated motor torque is 0.5Nm and the rated speed is 3000 rpm, which means that the torque should be continuous at $\frac{0.5}{3000} \cdot 750 = 0.125Nm$ and lower. Also note that the torque transducer used in this experiment is worn down and won't give precise values.

Unfortunately due to high load changes during these experiments, the connection between the motor, torque transducer and dynamo has been worsened. The motor can therefore not be used in any more experiments.

4.2.4.4 PWM-signals

The PWM signals measured during the experiments are shown below. There was only one probe available for the oscillator, so the figures will only contain one signal. The signals has gone through a low-pass filter, which is why some of the signals might seem a bit distorted. See figure captions for descriptions of the signals. The reasons behind these outcomes are already described in section 2.4.

Sensorless and Hall-effect driving

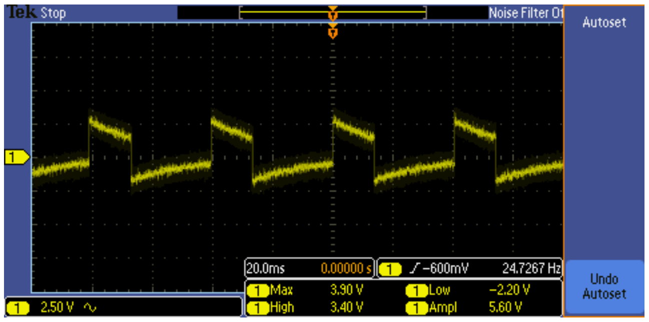


Figure 4.31: This is the input of the lower transistors on the inverter. The lowers transistors are either fully on or fully off. The two other transistor signals look the same, but the signals are shifted 120° of each other.

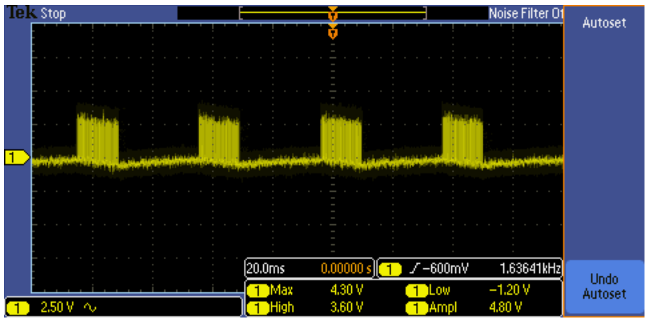

Tek Stop
Noise Filter Of
Autoset
Undo Autoset
20.0ms 0.00000 s 1 -600mV 1.63641kHz
1 Max 4.30 V 1 Low -1.20 V
1 High 3.60 V 1 Ampl 4.80 V
1 2.50 V

Figure 4.32: This is the input of the higher transistors on the inverter. The inputs are PWM signals. The two other transistor signals look the same, but the signals are shifted 120° of each other.

FOC driving

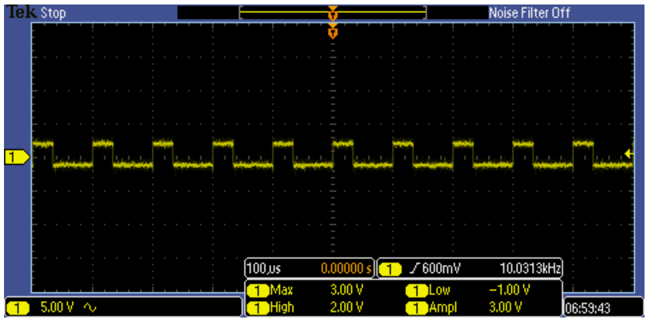

Tek Stop
Noise Filter Off
100 µs 0.00000 s 1 600mV 10.0313kHz
1 Max 3.00 V 1 Low -1.00 V
1 High 2.00 V 1 Ampl 3.00 V
1 5.00 V
06:53:43

Figure 4.33: This PWM signal varies in duty cycle. As the scale on the oscillator won't give us a full picture of how the signal look like, another figure at a different time is supplied underneath this figure. This is the input for one of the higher transistors. The lower transistors input, is inverted of this. The other transistor signals are shifted 120° of each other.

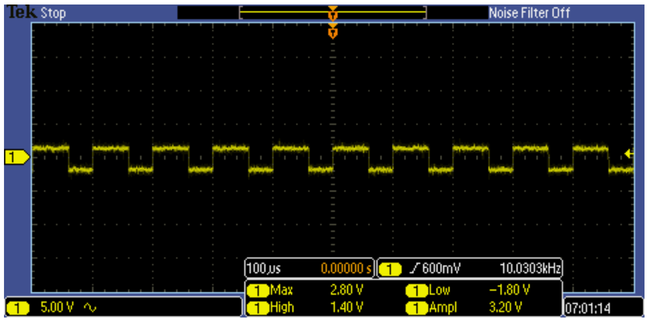
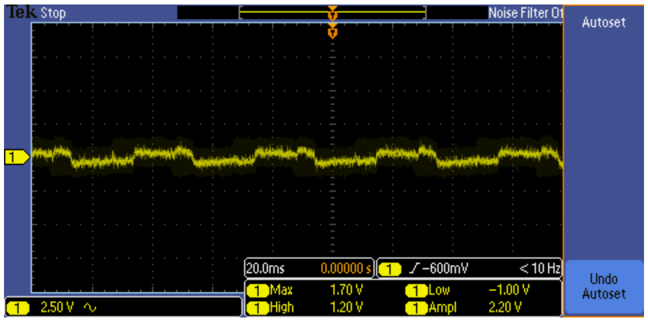

Tek Stop
Noise Filter Off
100 µs 0.00000 s 1 600mV 10.0303kHz
1 Max 2.80 V 1 Low -1.80 V
1 High 1.40 V 1 Ampl 3.20 V
1 5.00 V
07:01:14

Figure 4.34: This is the same signal as the one above, just at around one second later.

4.2.4.5 Voltage inputs

The voltage input signals measured during the experiments are shown below. There was only one probe available for the oscillator, so the figures will only contain one signal. The signals has gone through a low-pass filter, which is why some of the signals might seem a bit distorted. See figure captions for descriptions of the signals. The reasons behind these outcomes are already described in section 2.4.

Sensorless and Hall-effect driving

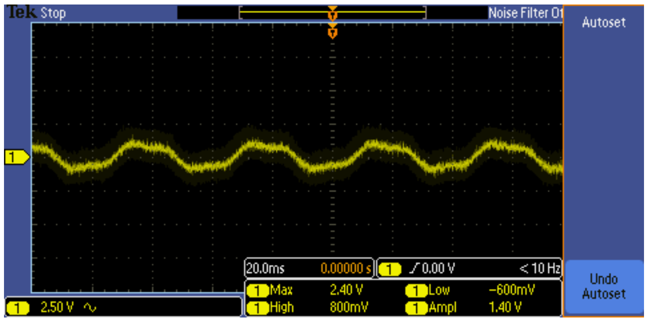


```
20.0ms   0.00000 s   1  ƒ -600mV   < 10 Hz
1 Max    1.70 V      1 Low    -1.00 V
1 High   1.20 V      1 Ampl   2.20 V
1  2.50 V  ∿
```

Figure 4.35: This is the voltage input of one of the phases, resulted by the PWM signals shown in the previous section. The two other voltage inputs look the same, but the signals are shifted 120° of each other.

FOC driving

:



```
20.0ms   0.00000 s   1  ƒ 0.00 V   < 10 Hz
1 Max    2.40 V      1 Low    -600mV
1 High   800mV       1 Ampl   1.40 V
1  2.50 V  ∿
```

Figure 4.36: This is the voltage input of one of the phases, resulted by the PWM signals shown in the previous section. The two other voltage inputs look the same, but the signals are shifted 120° of each other.

Chapter 5

Results

The outcome of the experiments shows that the sensorless control has problems during the start up time of the motor. As the back-EMF isn't available at low or none speed, the method is unpredictable and inaccurate. One of the requirements of the controller is that the driving should be precise, meaning that the speed /position always should be available. It is difficult to get an exact number of revolutions the rotor has done during the ramp up time of the motor. This can lead to misplacement of the linear actuator in the EMT which can have catastrophic events if e.g. the current protect of the motor fails and the actuator collides with an obstacle. The parameters for the start-up method implemented in this controller is dependent on how much load the motor sees. In normal operation of the EMT, it is not unusual that the motor has a very high load at start-up which requires a high amount of torque. This is the situation when the plug is set in the well, and the rotary actuator is going to pull it. As the sensorless method is based on back-EMF reading, it is exposed to noise during operations which will result in unstable speeds or worse. The start routine proposed in [26] was implemented and showed us that the start-up method of the motor was greatly improved, but still had some problems during high loads.

The rotor position is available almost instantly after the motor is running with use of Hall-sensors. This means that the motor won't have any problems starting up at high loads. The rotor position is obtained every 60° of an electrical round just like the zero-crossing method in sensorless control, but is less exposed to noise. The generation of PWM signals is also the same as the sensorless method, where the PWM signals are switching on and off without any really gentle transitions. As a result for both methods a torque ripple will be generated. This means lower efficiency and louder noise from the motor.

The FOC method generates PWM signals that smoothly increases and decreases the duty cycles in all three phases, instead of the hard two-phase on/off switching like the other methods. This results in a more sine-wave formed voltage than the other methods, which again prevents the torque ripples. The motor almost doesn't make a sound when its running.

5.1 Summary and Conclusions

The most efficient method is without a question FOC with use of an optical encoder (or resolver), though this is the most advanced and expensive method. The requirement specifications of the rotary actuator says that the speed control doesn't need to be very precise, as the rotary actuator only rotate until it hits a certain current protect(Using a relationship between current, torque, temperature and pressure). Both Hall-effect sensors and optical encoder with FOC can be used for this actuator. The back-EMF sensing method can also be used, but the start-up algorithm has to be greatly improved as the torque varies. The results shows that with the FOC method we actually can control the torque per amp directly, which is a huge advantage for knowing what torque the plug should be set at.

The linear actuator has higher requirement specifications. It is important that speed / position control is very accurate, as the actuator is moving between gear positions that is millimeter-defined. Unless the motor is connected to a gearbox that has a very high gear ratio that can compensate the potential loss of rounds, FOC with encoder/resolver is absolutely the best alternative.

5.2 Discussion

All of the methods are essentially eligible for the use of motor control in the EMT. There are clearly big differences between the methods, both good and bad.It all comes up to choosing a method depending on if a simple and low-cost method is good enough, or a more advanced but more expensive method is desired. All of the advantages and disadvantages should been covered up in the findings, and the choosing of method should be easy to evaluate.

5.3 Recommendations for Further Work

The start-up algorithm for sensorless driving of the BLDC motor can be improved. A recommendation for future work is to improve the algorithm to make the motor handle high loads during start up. An improved method of handling direct back-EMF sensing is proposed in [27] where the back-EMF signal amplified and reduced in noise. An improved way of driving a sensorless motor is needed in the marked as a low cost solution always will be desirable.

A more comprehensive analysis of the three different methods presented in this paper can be performed. Some suggestions are:

- Analyse the power consumption in the motor at different speeds and different loads. Compare the results with the different methods.
- Perform some experiments of position control. Let the motor run for x number of revolutions, and compare the final position with the different methods. It might be advisable to use a linear actuator for this, and eventually a gearbox to increase the torque and reduce the speed of the output shaft.
- Compare the different torque outputs, see how much torque ripples the different methods actually generate.
- Implement cascade-controllers in all methods, where the inner secondary regulator controls the current and the outer primary regulator controls the speed. Compare the response with and without the cascade implementation.

- Compare the BLDC motor with the conventional PMSM motor.

Chapter 6

Acronyms

ADC Analogue-to-Digital Converter

ALU Arithmetic Logic Unit

BLDC Brushless Direct Current

DAC Digital-to-Analogue Converter

DSP Digital Signal Processor

EMF Electromotive Force

EMT Electric Manipulation Tool

FOC Field Oriented Control

ISR Interrupt Service Routine

MCU Microcontroller Unit

MMF Magnetomotive Force

MOSFET Metal–Oxide–Semiconductor Field-Effect Transistor

PCB Printed Circuit Board

PMSM Permanent Magnet Synchronous Machine

PWM	Pulse Width Modulation
SVPWM	Space Vector Pulse Width Modulation
μ**c**	Microcontroller
μ**p**	Microprocessor

Appendices

Appendix A

Controller code

A.1 Description

The source code for the controller created is in Code Composer and is found in folder ”Appendix A" in the attached zip file:
There are three folders:

math_blocks - Contains the pre-defined mathblock by Texas Instruments. Note that these are customized to own code.

HVPM_Sensored - Contains source code for the FOC method.

BLDC_Testing - Contains source code for the Hall-effect sensors and sensorless method.

Remark:Please note that Code Composer Studio and ControlSuite from Texas Instruments has to be installed on the computer to compile the solutions.

Appendix B

Matlab code

B.1 Description

The source code for the controller created in Matlab is found in folder ”Appendix B" in the attached zip file:

The files included are:

ziegler.m that creates a plot showing how the parameters are found from Ziegler’s method.

PlotIT.m that is used for creating plots from the simulink model.

MakePlots.m makes plots of the estimated response, and the actual response. PI-parameters are calculated using ziegler open-loop method.

Main.m is used for running all the different functions.

Lodd.m is used for finding the motor’s moment of inertia.

BLDC_model.slx is used for simulating the BLDC motor.

.MAT files contains measuring data.

Bibliography

[1] B. Akin and M. Bhardwaj, “Sensorless Trapezoidal Control of BLDC Motors,” pp. 1–40, 2013. [Online]. Available: www.ti.com/lit/an/sprabq7/sprabq7.pdf

[2] ——, “Trapezoidal Control of BLDC Motors Using Hall Effect Sensors,” pp. 1–33, 2010. [Online]. Available: www.ti.com/lit/an/sprabq6/sprabq6.pdf

[3] (Texas Instruments), “Digital Motor Control Softare Library,” pp. 1–221, 2003. [Online]. Available: http://www.ti.com/lit/ug/spru485a/spru485a.pdf

[4] M. Tengesdal, “Fråtransistor til datamaskin,” pp. 1–269, 2014.

[5] J. Catsoulis, *Designing Embedded Hardware*, 2nd ed., 2005.

[6] (Texas Instruments), “Brushless DC Motor for TMDSHVMTRPFCKIT.” [Online]. Available: http://www.ti.com/tool/hvbldcmtr

[7] Hyperphysics, “Electricity and Magnetism.” [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/emcon.html#emcon

[8] Electrical4u, “Electric Current and Theory of Electricity | Heating & Magnetic Effect.” [Online]. Available: http://www.electrical4u.com/electric-current-and-theory-of-electricity/

[9] T. Hartsfield, “How Tesla Coils Work,” 2014. [Online]. Available: http://www.realclearscience.com/articles/2014/01/29/how_tesla_coils_work_108474.html

[10] Hyperphysics, “Lenz’s Law.” [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electric/farlaw.html#c1

[11] Aplusphysics, “Electromagnetism.” [Online]. Available: http://www.aplusphysics.com/courses/honors/magnets/electromagnetism.html

[12] Wikipedia, “Stator.” [Online]. Available: https://en.wikipedia.org/wiki/Stator

[13] ATSautomation, “AWS 747 NEEDLE WINDING MACHINE.” [Online]. Available: http://www.atsautomation.com/en/Transportation/AssemblyandProcessing/ArmatureandStator/AWS747NeedleWindingMachine.aspx

[14] E. Hendawi, F. Khater, and A. Shaltout, “Analysis, Simulation and Implementation of Space Vector Pulse Width Modulation Inverter,” pp. 124–131, 2010. [Online]. Available: http://www.wseas.us/e-library/conferences/2010/Penang/AEE/AEE-18.pdf

[15] D. S. Controller, “Introduction Welcome to the F2833x - Tutorial,” pp. 1–26. [Online]. Available: http://cnx.org/contents/28f325b1-a7b5-4024-be16-95f0fc06c47e@2/Introduction_to_TMS320F28335

[16] K. Skretting, “ELE500 Signalbehandling, 2013,” pp. 1–34, 2013. [Online]. Available: http://www.ux.uis.no/~karlsk/ELE500/u3438.pdf

[17] Mathworks, “Floating-Point Numbers.” [Online]. Available: http://se.mathworks.com/help/matlab/matlab_prog/floating-point-numbers.html

[18] (Texas Instruments), “C28x IQmath Library.” [Online]. Available: http://www.ti.com/lit/sw/sprc990/sprc990.pdf

[19] V. K. S. Patel and A.K.Pandey, “Modeling and Performance Analysis of PID Controlled BLDC Motor and Different Schemes of PWM Controlled,” pp. 1–14, 2013. [Online]. Available: http://www.ijsrp.org/research-paper-0413/ijsrp-p1622.pdf

[20] AnaheimAutomation, “BLWS23 Series - Brushless DC Motors •.” [Online]. Available: http://www.anaheimautomation.com/manuals/brushless/L010229-BLWS23SeriesProductSheet.pdf

[21] BaumerElectric, “Incremental shaft encoder,” pp. 47–49. [Online]. Available: http://docs-europe.electrocomponents.com/webdocs/0573/0900766b8057356b.pdf

[22] Norbar, “NORBAR TORQUE TOOLS,” p. 11000, 2008. [Online]. Available: http://www.norbar.com/en-sg/products/view/product/categoryname/torque-transducers/rangename/rotary-transducers/pname/-smart-20-n-m-rotary-td-1-4-m-f-sq-dr-mv-v/category_multid/78/range_multid/77/id/6017

[23] (Texas Instruments), “High Voltage Motor Control and PFC (R1 . 1) Kit Hardware Reference Guide,” pp. 1–14. [Online]. Available: http://www.deyisupport.com/cfs-file.ashx/__key/telligent-evolution-components-attachments/00-56-01-00-00-20-88-17/20150326115352.pdf

[24] ——, “Piccolo ™ Microcontrollers,” 2013. [Online]. Available: http://www.ti.com/lit/ds/symlink/tms320f28035.pdf

[25] B. P. Cord, “High Voltage Digital Motor Control Kit (R1 . 1) Quick Start Guide,” pp. 1–10. [Online]. Available: ftp://ftp.ti.com/pub/dml/DMLrequest/Christy_FTP-10-30-12/controlSUITE/development_kits/HVMotorCtrl%2BPfcKit_v2.0/~GUI/QSG-HVMTRPFCKIT-GUIv2.pdf

[26] Microcontroller Division Applications, “Application Note Bldc Motor Start Routine for the St72141,” pp. 1–18. [Online]. Available: http://pdf.datasheetcatalog.com/datasheet/SGSThomsonMicroelectronics/mXywxsy.pdf

[27] J. Shao, “Direct Back EMF Detection Method for Sensorless Brushless DC (BLDC) Motor Drives,” Ph.D. dissertation, 2003. [Online]. Available: http://scholar.lib.vt.edu/theses/available/etd-09152003-171904/unrestricted/T.pdf