Drive System for an Electric Go-Kart

Estefanía Ruiz, Aitor Teran, Nicolai Fransen, Mihai Rusu,
Faheem Ahmad, Nicolás Murguizur Bustos

Energy Technology, PED2-841, 2019-05

Master's Project

Copyright © Aalborg University 2019

This report has been written using LaTeX. During the project Code Composer Studio has been used for code development and MATLAB and LTspice have been used for simulations, while a Piccolo controller from Texas Instruments has been used for the implementation of the control system. The creation of flow charts have been done using www.draw.io. Hardware schematic and PCB layout have been created using Altium Designer.

Department of Energy Technology
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Drive System for an Electric Go-Kart

Theme:
Dynamics in Electrical Energy Engineering

Project Period:
Spring Semester 2019

Project Group:
PED2-841

Participant(s):
Estefanía Ruiz
Aitor Teran
Nicolai Fransen
Mihai Rusu
Nicolás Murguizur Bustos
Faheem Ahmad

Supervisor(s):
Lajos Török
Erik Schaltz

Copies: 1

Page Numbers: 129

Date of Completion:
May 20, 2019

Abstract:

An electric go-kart is desired to replace the classical combustion engine go-kart as providing a clean and quite alternative. The goal of this project is to design, simulate and implement the control strategy of the drive train of an electric go-kart. Modelling and simulations were done in MATLAB/Simulink while software development was done using *C* programming language. Several strategies to control the inverter were analyzed and finally Space Vector Pulse Width Modulation (SVPWM) was chosen. The control of the induction machine was done both by Scalar Control and Vector Control (Field Oriented Control) which were also implemented using a Digital Signal Processor (DSP). The battery pack was tested due to its connection with regenerative braking. Practical results are to be added after performing the FOC together with conclusions.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

Preface

This document describes the report of the group project "Drive System for an Electric Go-Kart" It has been developed from the 10th of February to the 31st of may of 2019, at Aalborg University, Institute of Energy, by the group PED-841. This report discusses the procedure and results of the design and implementation of the control of an inverter to drive an induction machine for electrical mobility. The literature references are shown in square brackets, with a number referring to a specific document which can be found in the bibliography. If the reference is after the dot, it means that it refers to the whole previous paragraph. Pictures and tables will be denoted in the X,Y format, with X representing the chapter and Y the figure or table number. The process and development has been based on the Problem Based Learning (PBL) method.

Aalborg University, May 20, 2019

Estefanía Ruiz
eruiza18@student.aau.dk

Aitor Teran
ateran18@student.aau.dk

Nicolai Fransen
nfrans18@student.aau.dk

Mihai Rusu
mrusu18@student.aau.dk

Faheem Ahmad
fahmad18@student.aau.dk

Nicolás Murguizur Bustos
nmurgu18@student.aau.dk

Nomenclature

Abbreviations:

ADC	Analog to Digital Converter
CQ	Circular Queue
DSP	Digital Signal Processor
IM	Induction Machine
BJT	Bipolar Junction Transistor
EOC	End of Conversion
FOC	Field Oriented Control
PWM	Pulse-Width Modulation
SOC	State of charge
SPWM	Sinusoidal Pulse-Width Modulation
SVPWM	Space Vector Pulse-Width Modulation
SWC	Software Component
THD	Total Harmonic Distortion
THIPWM	Third Harmonic Injection Pulse-Width Modulation
VSI	Voltage Source Inverter
TCB	Task Control Block
UML	Unified Modeling Language
GUI	Graphical User Interface
FSM	Finite State Machine
CAN	Controller Area Network
UART	Universal Asynchronous Receiver Transmitter

Symbols:

a_0	Filter parameter for inputs
b_1	Filter parameter for outputs
f	Fundamental frequency
f_{filter}	Execution frequency of digital filter
f_s	Switching frequency
k	Sector number
s	Slip
T_e	Electromechanical torque
T_{load}	Load torque
T_s	Switching period
V_{DC}	DC link voltage
$\vec{V}_k$	Active space vector
$\vec{V}_r$	Reference vector
x_n	Input to digital filter
y_n	Output of digital filter
y_{n-1}	Output of digital filter at lag 1
ω	Angular velocity
τ_{filter}	Time constant of digital filter

1 Introduction

The electric vehicle (EV) industry has been rapidly evolving during the past years. Governments across the globe are supporting the initiative to drive the humankind away from fossil fuel conventional mobility vehicles. Many car companies are investing heavily in the development of more sustainable mobility options. Year on year global addition of electric vehicle fleet is seeing an upward of 50% rate. With an addition of 60% in the year from 2015-2016 and 57% from year 2016-2017 [1]. Figure 1.1 below shows the growth in number of electric vehicles across the globe and major countries.

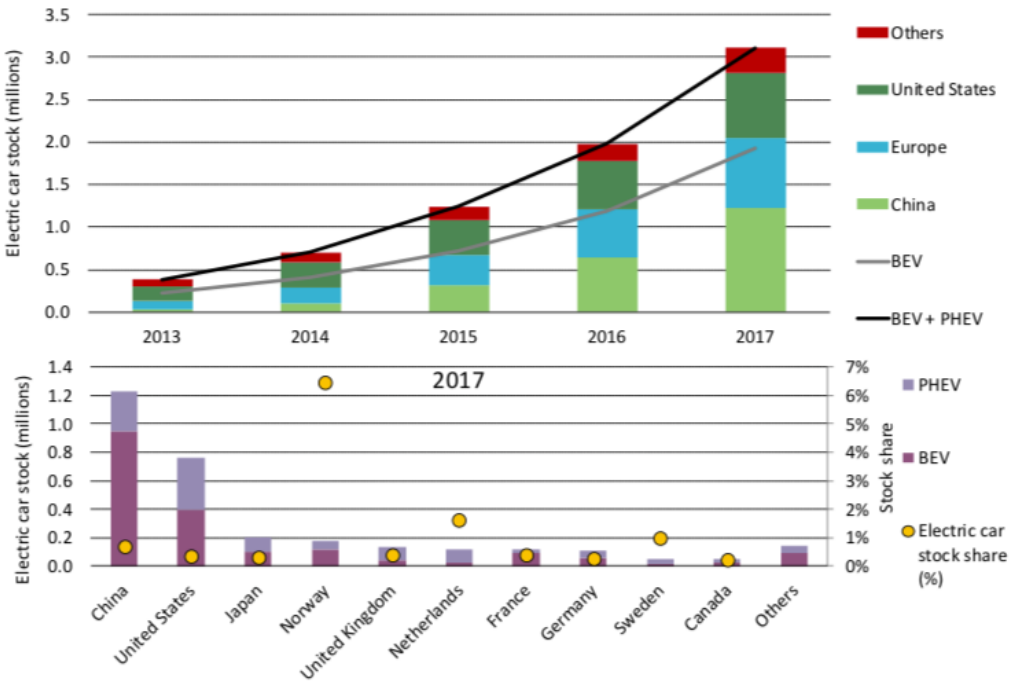


Figure 1.1: *Top graph:* Passenger electric cars fleet year on year growth. *Bottom graph:* Top ten Electric Vehicle Countries

The rising market of electric vehicle also comes at the expense of many challenges The rising tendency of the EV sector makes it very appealing to perform

further research in the topic. Classic go-kart which are powered by a combustion engine are known for being agile and light vehicles. Implementing an electric motor in it would result in many advantages, the high levels of noise would be much reduced and the exhaust gasses would disappear, this makes it ideal for indoor race tracks and more appealing for many people on an outdoor track. On top of this, an electric motor is known for having higher torque than combustion engines at low speeds, making the acceleration of an electric go-kart higher in most of the cases.

Should we have a reference for some of these pros? NHF

This project is the continuation of two other projects carried out at Aalborg University during the years 2011 [2] and 2018 [3]. After the research performed during previous years, the go-kart hardware was mostly developed. Where the motor that has been used is a squirrel cage 24V and 5.3kW machine. During the year 2018, a three phase bidirectional inverter was developed in order to power the motor [3], this inverter could not be tested in depth by the developers, so further tests had to be performed during this project.

1.1 Objectives

As stated in the introduction, the hardware of the go-kart was mostly developed before starting the project. Therefore, the main focus of this project will be related to control and firmware development. For this reason, a field oriented control (FOC) is to be designed for a low voltage and high current motor. The control should include algorithms for regenerative braking and cruise control strategies. Simulations of the control are to be performed. The FOC is to be implemented in a digital signal processor (DSP) which will also monitor different parameters of the system, being able to trigger errors if they occur and monitor different parameters in real time. An interface board is also necessary for interacting with the motor.

However, after creating the inverter, previous year's group was not able to perform tests in order to validate it. Also, the batteries were last tested during the project developed in the year 2011 [2] so the parameters obtained before might not longer be adequate. Therefore testing the inverter and making a new model of the batteries are also objectives for this project.

Another objective of the current project is to test and improve the team members ability to develop a well organize project management strategy.

1.2 Scope of project

The main scope of this project is to develop the control of a system that includes an electrical machine, an inverter and a battery pack. Having the hardware available, the main focus will be on simulations and on software development and implementation. In terms of control, design and simulation, specific tools such as Simulink will be used in order to model different cases with practical application such as torque control and cruise control. This simulations will provide valuable information about how the system will behave giving the opportunity to assess various cases.

Software Scope How to determine how much you design ANDdevelop: compromise between perfect software and time. pull request review and how much time we put into it.

Problem statement We will include the problem statement inside the scope of the project.

1.3 Project management

One of the initial goals of the project is to keep an organized group structure, making it possible to obtain the best output of the team's workload. Being six team members with multiple lectures in a time span of four months, efficiency is key to success.

In order to achieve this, the work flow is organized into tasks, atomizing them as much as possible in order to have short-termed, individual, clear objectives. The tasks are usually assigned to one person, two at most.

The organization of the group is based on SCRUM. This is an agile project management framework designed to allow rapid changes on a project flow. Where it also introduces the opportunity to multitask efficiently inside a team. SCRUM is based on short sprints (1 to 4 weeks) that are planned beforehand. During one sprint, short daily meetings are arranged to align team members and understand progress. Before the sprint is over, initial plans are reviewed and goals are achieved to move forward to the next sprint.

improve this. NM should take a look. AT

From the SCRUM framework, the sprint methodology is taken for this project. However, it has minor changes due to the research nature and the broad topic covered. It was soon understood that the clear deadline for sending the report made it important to have a very well organized time frame.

In order to have a real-time view of the current project status by every team member, the online-based software *Trello* is used.*Trello* allows the easy managing of task cards, each including a title, assigned team members, card description and outputs, they can also include comments or due dates. Tasks could have different statuses which are listed below.

Decide if software tools used should be written in italic, NHF

- Backlog: tasks to be performed in future sprints.
- To-do: tasks to be performed during current sprint.
- Doing: tasks currently under develop.
- Waiting for revision: tasks to be revised by another team member.
- Done: tasks that have been completed and revised.

Each tasks corresponds to a card in *Trello* and it is moved from group to group depending on its status, a scheme of the layout of cards is shown in Figure 1.2.

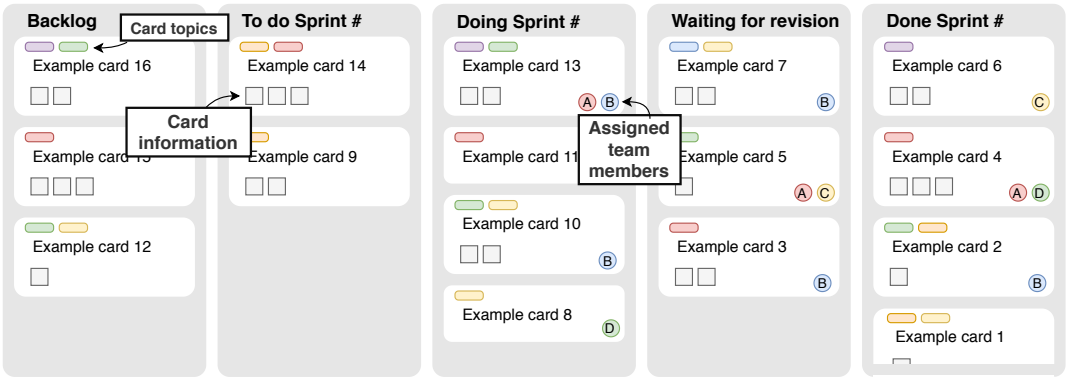


Figure 1.2: Organization of cards in *Trello*, each card corresponds to a task.

Sprints last approximately two weeks. At the end of each sprint a meeting is held among the team members to align the progress and set new goals. Meetings with the supervisors are scheduled after the team meetings in order to have clear results and goals for the near future.

Sprint methodology shows good short term results, however, this method does not allow to have a big picture of the long term progress of the project. For this reason, a Gantt diagram is also designed and it has been followed as accurately as possible, this diagram needs to be reviewed during sprint meetings and updates are sometimes necessary. The Gantt chart allows the team to keep a clear view of the current progress, in order to understand project priorities and adjust the workload accurately.

If possible the full gantt should be added to the Appendix, NHF, FA

2 Modeling and design of electric power train

This chapter discusses the hardware that will make up the power train of go-kart. At first the induction machine will be discussed, where a brief introduction to the fundamentals of induction machine regarding structure and operation principle is provided for the reader. This will be followed by the electrical parameters of the induction motor used in this project. Since rotor and stator parameters have been taken from previous groups' reports it was important to verify these parameters with some preliminary simulation results like rated torque, synchronous as well as rated speed at rated torque, and power factor. The simulation are based on the $\alpha\beta$ reference frame model of induction machine. Next, the 3 phase voltage source inverter (VSI) is discussed. Design of inverter and the experiments conducted for verifying proper operation is elaborated. Third section deals with the battery (lead-acid type) where equivalent model of battery and the tests conducted for obtaining battery parameters have been detailed. Section 2.4 deals with the design of interface board. This interface board will be mounted with the DSP and switches/buttons to let user interact with DSP and set references.

Finally the mechanical model of the go-kart is also provided in this chapter. Drag forces that the traction torque has to overcome and the maximum acceleration the go-kart can achieve based on the power available from induction motor.

2.1 Induction machine

Induction machines are one of the most commonly used electrical machines for electric drive applications due to its simple and robust structure [4][5]. An induction machine (IM) has two main parts, the stator and the rotor. Depending on the type of rotor the IM's are classified as wound-rotor IM or squirrel-cage IM. Squirrel-cage induction machines have the advantage of simple construction, low cost and low maintenance as they do not contain brushes as DC motors. This is the reason why these kind of induction machines are very popular for implementing field oriented control methods as they can replace DC motors resulting in very

high performance [6].

In this section, first a brief review of the fundamentals of squirrel-cage induction machines will be explained in order to give the reader an overview of how these machines operate. Secondly, the dynamic model of the IM will be derived and finally, the simulation results to verify the model will be presented.

2.1.1 Fundamentals of induction machines

The stator of the induction machine is made of three windings which are shifted in space by 120°. The squirrel-cage rotor consists of a set of conductors/bars which are embedded in the rotor slots and short-circuited at both ends using conductive rings [4][5]. Figure 2.1 shows the cross sectional area of a squirrel-cage induction motor for a two-layer winding four pole machine, where the red points are the rotor's short-circuited conductors and the three phase windings are shown in the abc reference frame.

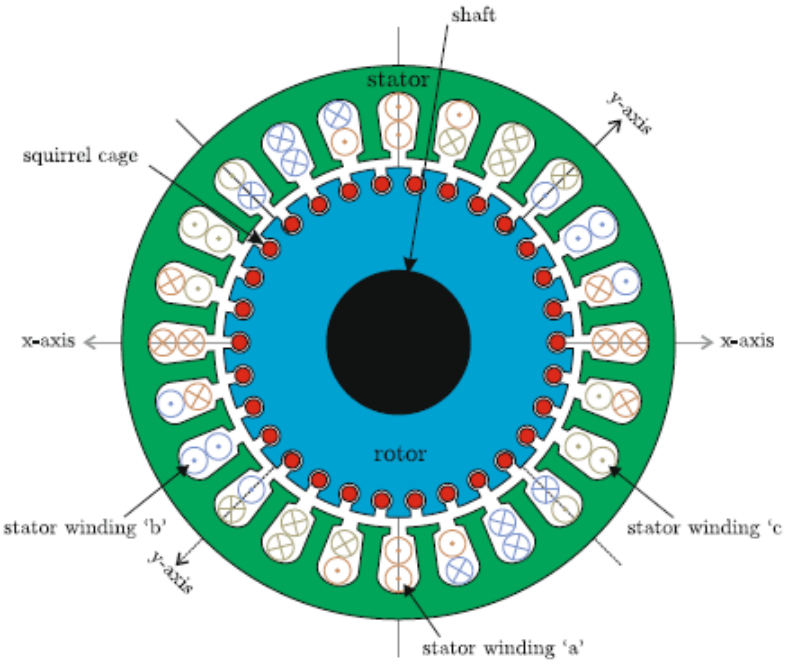


Figure 2.1: Cross sectional view of a squirrel-cage induction motor [4].

The operation of induction machines is based on the generation of electromagnetic torque from the force produced by induced currents in the rotor. These induced currents appear in the rotor because of the rotating magnetic field generated when the stator is excited with three-phase currents. This field is rotating at synchronous speed and sweeps past the rotor's conductors generating a voltage in them [6]. This voltage induces sinusoidal distributed alternating currents in the squirrel cage. These currents, according to Lorentz's force law, will produce a force on the rotor which is responsible for the torque production, and consequently the rotor will start rotating.

It is important to note that currents will be induced in the rotor only if it is

rotating asynchronously with respect to the rotating magnetic field. If the rotor and the magnetic field were rotating both at synchronous speed, there would be no change of flux through the rotor's conductors and therefore, currents will not be induced in the squirrel cage resulting in no force production and hence no electromagnetic torque will be generated [4][5].

From this it is concluded that the speed of the rotor needs to be slightly lower or higher than the synchronous speed for motoring mode operation or generation mode, respectively. This is the reason why the induction machine is also known as asynchronous machine [5]. The difference between the rotor speed ω_r and the synchronous speed ω_e is known as slip and it is usually defined as a percentage. From Equation 2.1 it is seen that the slip varies from 0 to 1 (0-100%) in motoring mode. This is the range in which torque is generated by the machine. When the rotor speed reaches synchronous speed the slip is null ($s = 0$) and when the motor is at standstill or the rotor is locked the slip is maximum ($s = 1$). In practice, induction machines operate at low slip values usually lower than 5% at full load torque [5].

$$s = \frac{\omega_e - \omega_r}{\omega_e} \cdot 100 \tag{2.1}$$

The synchronous speed can be calculated from the stator frequency f and the number of pole pairs p as stated in equation 2.2 [5].

$$\omega_e = \frac{60 \cdot f}{p} \tag{2.2}$$

Figure 2.2 shows the torque-speed characteristic curve for an induction machine where it is observed that at full load torque the slip is small. From that point to the right is the common region of operation of induction machines where the relation between torque and rotor speed is almost linear. Figure 2.3 shows the torque-speed curve with the two modes of operation of the induction machine.

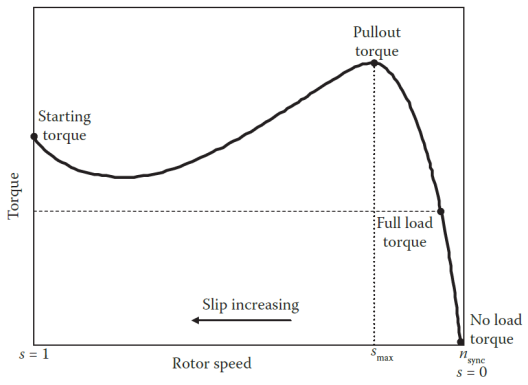


Figure 2.2: Torque-speed characteristic curve for a general induction machine in motoring mode [5].

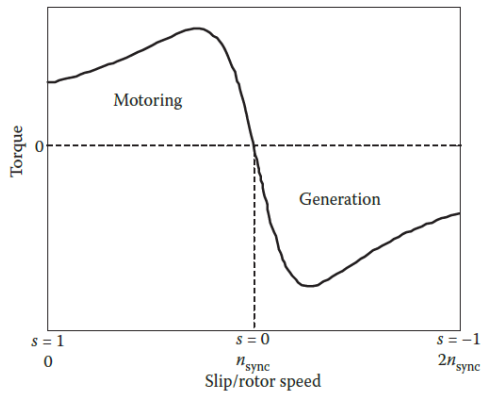


Figure 2.3: Torque-speed characteristic curve for a general induction machine in motoring and generation mode [5].

It is also possible that the rotor speed gets higher than the synchronous speed. In that case, the slip and the torque will be negative resulting in the machine

working as a generator. It is in this mode of operation that regenerative braking will take place when the slip varies between 0 and -1. However, this topic will be explained in more detail in the regenerative braking section 3.3.

2.1.2 Dynamic model of the induction machine

The induction machine available in the laboratory for the development of this project is a 5.3kW squirrel-cage IM from *Sauer Danfoss*. Table 2.1 shows the nominal parameters of the induction machine and the electrical constant parameters that will be used for the modelling of the machine. These parameters were provided by previous groups [2][3] and were not measured or obtained during the project development.

Induction machine nominal parameters		
Power	P_n	5.3 [kW]
Peak power (for max. 60 min)	P_{max}	7.3 [kW]
Phase voltage	U_n	13.85 [V]
Phase current	I_n	189 [A]
Power factor	PF	0.76
Stator frequency	f_n	58 [Hz]
Mechanical shaft speed	ω_n	1685 [rpm]
Electromagnetic torque	T_n	30.04 [Nm]
Induction machine constant parameters		
Magnetizing inductance	L_m	0.38 [mH]
Stator leakage inductance	L_{ls}	31.16 [μH]
Rotor leakage inductance	L_{lr}	31.16 [μH]
Stator resistance	R_s	2.5 [mΩ]
Rotor resistance	R_r	2.69 [mΩ]
Rotor moment of inertia	J	0.0151 [Nm^2]
Pole pairs	p	2

Table 2.1: "TSA170-210-038 Sauer Danfoss" induction machine's parameters [3].

Reference frame transformation is used extensively to simplify three phase systems to simpler orthogonal 2 phase systems [7]. Analysis of induction machines in two phase systems requires to write the stator and rotor voltage equations in a rotating dq reference frame or a stationary $\alpha\beta$ reference frame. For modelling purposes it is common to use $\alpha\beta$ reference frame [8], which is shown for a general space vector $\vec{f}$ in Figure 2.4. The $\alpha-$ axis is aligned with the $a-$ axis and the $\beta-$ axis is leading by 90°.

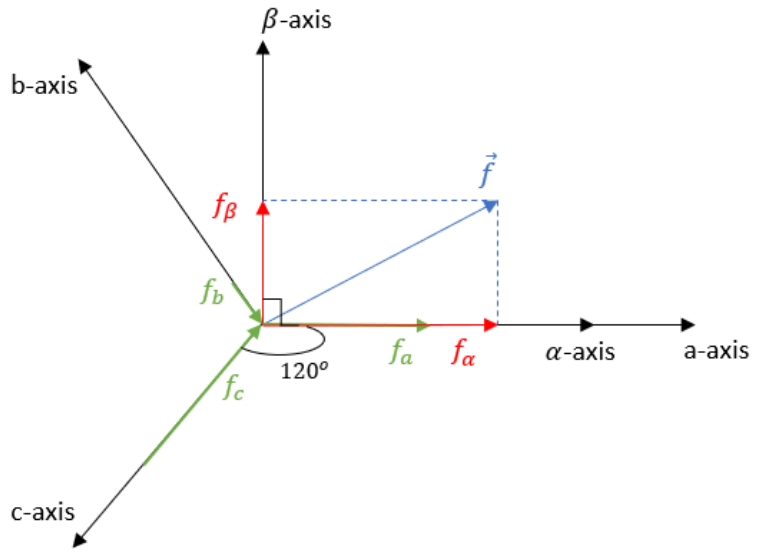


Figure 2.4: Graphical representation of the space vector $\vec{f}$ in $\alpha\beta$ reference frame.

By applying reference frame transformation from abc to $\alpha\beta$ frame, following Equation 2.3 [6], it is possible to obtain a two-dimensional model of the induction machine. This transformation is known as Clarke's transformation and it simplifies the modelling of the IM as it considers only two stationary variables.

$$\begin{bmatrix} f_\alpha \\ f_\beta \\ f_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} \tag{2.3}$$

In reference frame transformation, there must be a constraint in order to obtain only two independent variables from a three phase system. This constraint is a variable defined as zero component (f_0) which is the same for all the reference frames and it is defined in Equation 2.4. The zero component is null ($f_0 = 0$) when working with balanced (symmetrical) systems .

ref to lecture dynamic modelling

$$f_0 = \frac{1}{3} \cdot (f_a + f_b + f_c) \tag{2.4}$$

The induction machine's equivalent electric circuit with respect to the α and β axis are shown in Figures 2.5 and 2.6, respectively.

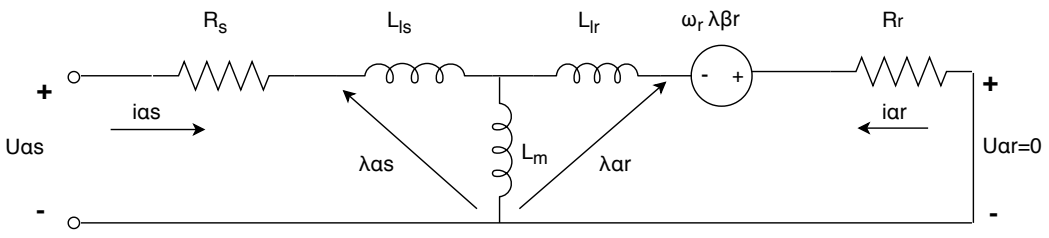


Figure 2.5: Equivalent circuit of the induction machine in the α-axis frame.

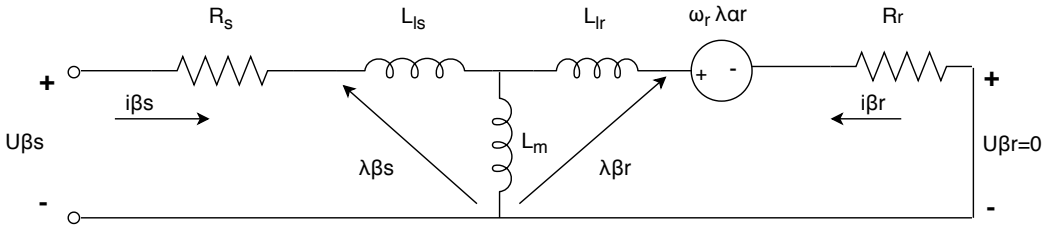


Figure 2.6: Equivalent circuit of the induction machine in the β-axis frame.

The induction machine equations are divided into an electrical part and a mechanical part. The electrical equations of the IM modeled in the $\alpha\beta$ reference frame are obtained from the equivalent electric circuits of Figures 2.5 and 2.6. The induction machine's voltage equations are shown in Equation 2.5, and the flux linkage equations in Equation 2.6 [6]. Subscript α or β represent the axis and subscript s or r represents stator and rotor, respectively.

$$\begin{aligned} u_{\alpha s} &= R_s i_{\alpha s} + \frac{d\lambda_{\alpha s}}{dt} \qquad & u_{\alpha r} = 0 &= R_r i_{\alpha r} + \frac{d\lambda_{\alpha r}}{dt} + \omega_r \lambda_{\beta r} \\ u_{\beta s} &= R_s i_{\beta s} + \frac{d\lambda_{\beta s}}{dt} \qquad & u_{\beta r} = 0 &= R_r i_{\beta r} + \frac{d\lambda_{\beta r}}{dt} - \omega_r \lambda_{\alpha r} \end{aligned} \tag{2.5}$$

$$\begin{aligned} \lambda_{\alpha s} &= L_{ls} i_{\alpha s} + L_m \cdot (i_{\alpha s} + i_{\alpha_r}) \qquad & \lambda_{\alpha r} &= L_{lr} i_{\alpha r} + L_m \cdot (i_{\alpha s} + i_{\alpha_r}) \\ \lambda_{\beta s} &= L_{ls} i_{\beta s} + L_m \cdot (i_{\beta s} + i_{\beta_r}) \qquad & \lambda_{\beta r} &= L_{lr} i_{\beta r} + L_m \cdot (i_{\beta s} + i_{\beta_r}) \end{aligned} \tag{2.6}$$

From Equations 2.5 and 2.6 the stator and rotor currents are obtained from the stator voltages and flux linkages. The rotor voltages are set to null because the rotor conductor's are short-circuited at both ends, as it was mentioned earlier in this chapter. Also, as the $\alpha\beta$ frame is stationary, the synchronous speed is zero ($\omega_e = 0$) and only the electrical shaft speed (ω_r) is included in the IM model.

The mechanical part of the induction machine can be modeled as in Equation 2.7 where T_e is the electromagnetic torque, T_L the load torque, J is the inertia of the machine and α_r is the mechanical angular acceleration of the rotor, B is the viscous friction and ω_r the rotor's angular speed [7].

$$T_e - T_L = J \cdot \frac{d\omega_r}{dt} = J \cdot \alpha_r + B \cdot \omega_r \tag{2.7}$$

Equation 2.8 represents the electromagnetic torque generated by the motor in the $\alpha\beta$ frame [7]. It depends on the number of pole pairs, the magnetizing inductance and the stator and rotor currents. It is possible to obtain the torque as a function of just one of the stator currents by implementing field oriented control as it will be shown in Chapter 3.

$$T_e = \frac{3}{2} p L_m (i_{\beta s} \cdot i_{\alpha r} - i_{\alpha s} \cdot i_{\beta r}) \tag{2.8}$$

The implementation of the IM model in *Simulink* is shown in Appendix A with the electrical part and the mechanical part of the model in sections A.1 and A.2, respectively.

2.1.3 Simulation results

The dynamic model of the induction machine is used to verify the IM's nominal parameters (Table 2.1). The model implementation in *Simulink* is done following the block diagram of Figure 2.7.

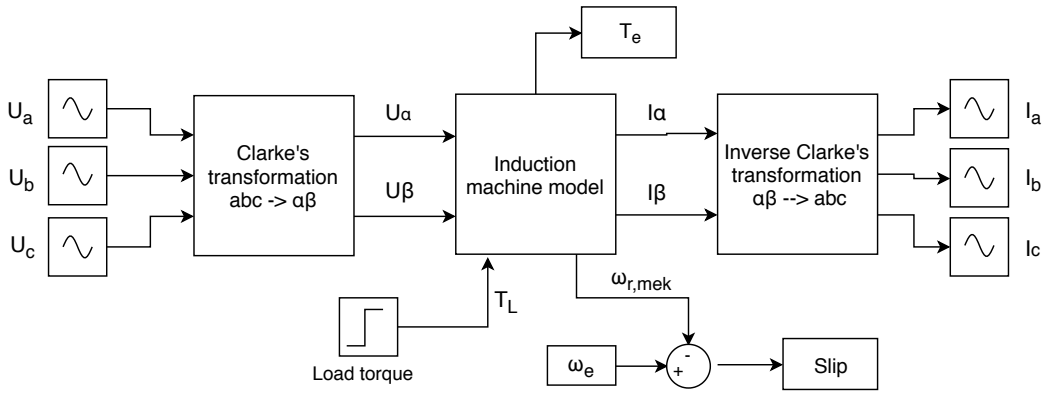


Figure 2.7: Block diagram for the induction machine model implementation.

The input of the system is the three phase nominal stator voltages ($U_n = 13.85V$) which are transformed to $\alpha\beta$ voltage components by applying Clarke's transformation. The electrical and mechanical equations for the IM, derived in the previous section, are included in the *Induction machine model* block. Inverse Clarke's transformation is applied to the stator $\alpha\beta$ current components in order to obtain the three phase stator currents. Therefore, the outputs of the dynamic model are the stator currents, the electromagnetic torque and the mechanical shaft speed used for calculating the slip by comparing it with the synchronous speed ω_e.

The simulation results that will be shown in this section are obtained for a step load torque applied at $t = 0.5s$. For validating the IM's nominal parameters the load torque is the nominal electromagnetic torque which is obtained in Equation 2.9. From Figure 2.8 it is observed that the electromagnetic torque T_e follows the load torque reaching its rated value.

$$T_n = \frac{P_n}{\omega_n} = \frac{5.3kW}{1685 \cdot \frac{\pi}{30} rad/s} = 30.04 \qquad [Nm] \tag{2.9}$$

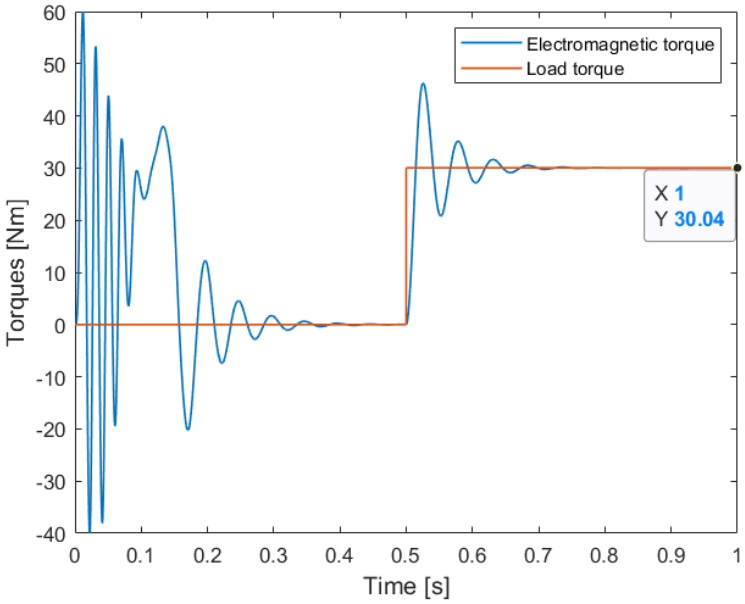


Figure 2.8: Electromagnetic torque for a step rated load torque applied at $t = 0.5s$ for validation of nominal electromagnetic torque.

Under the same conditions, the three phase sinusoidal currents are depicted in Figure 2.9 showing an increment of the currents from $130.5A$ under no load condition to $262.1A$ under full load. The theoretical peak stator current is $I_{n,peak} = I_n \cdot \sqrt{2} = 267.28A$ which is very close to the simulated result. The bottom graph is a zoomed view of the top graph in order to observe that the stator currents are sinusoidal and phase shifted 120°.

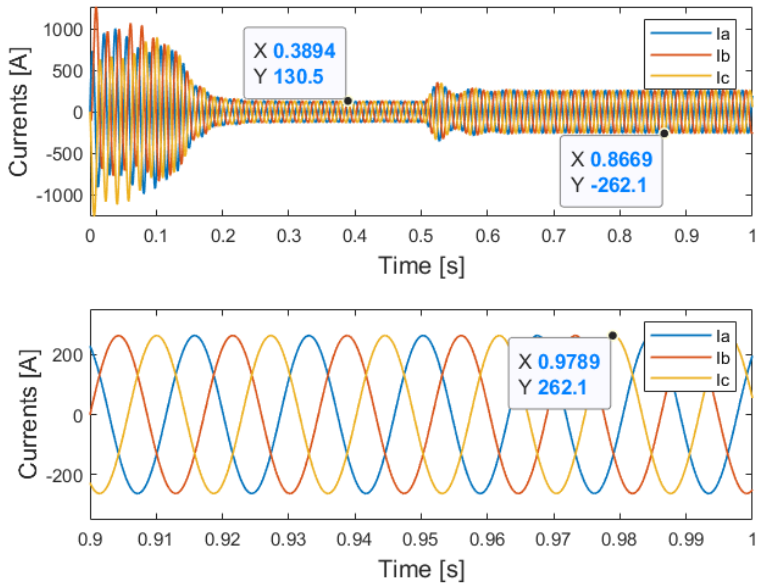


Figure 2.9: *Top graph:* Three phase stator currents for a step rated load torque applied at $t = 0.5s$ for nominal stator currents validation. *Bottom graph:* Zoomed view of the stator currents.

The rated slip is calculated is the difference between the rated synchronous speed and the mechanical rotor speed under full load. The rated synchronous speed is calculated from Equation 2.2 and its value in rpm is found to be:

$$n_{e,rated} = \frac{60 \cdot f_n}{p} = \frac{60 \cdot 58Hz}{2} = 1740[rpm] \tag{2.10}$$

From Figure 2.10 it is obtained that under full load ($T_L = 30.04Nm$) the mechanical shaft speed is $n_r = 1681rpm$. Thus, the rated slip is calculated in Equation 2.11. As it was mentioned in Subsection 2.1.1, IMs usually operate at low slip values ($s < 5\%$) under full load conditions which in this case is verified.

$$s = \frac{n_{e,rated} - n_r}{n_{e,rated}} \cdot 100 = \frac{1740rpm - 1681rpm}{1740rpm} \cdot 100 = 3.39[\%] \tag{2.11}$$

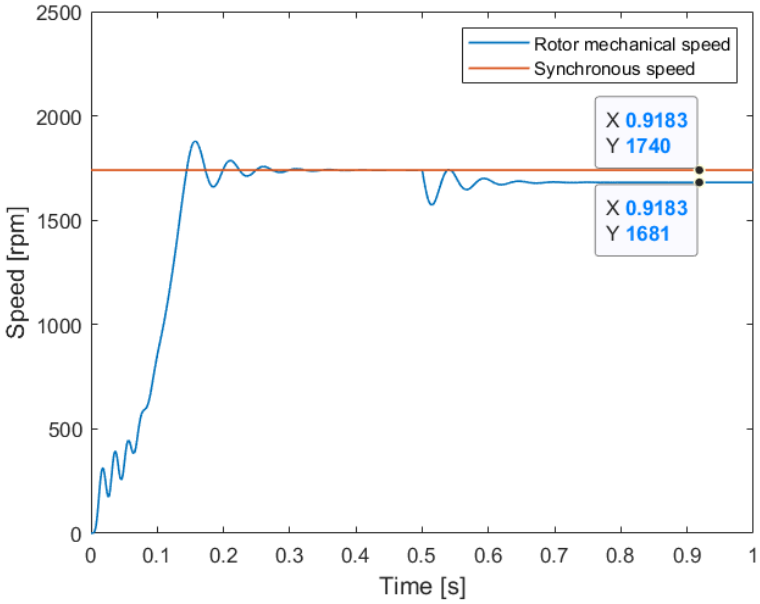


Figure 2.10: Comparison of rotor speed and synchronous speed for a step rated load torque applied at $t = 0.5s$ for slip calculation.

Finally, the power factor (PF) of the induction machine will be calculated from the simulation results. The power factor is defined as the ratio between the active power (P) and the apparent power (S), expressed in Equation 2.12 [7]. The result is the cosine of the phase angle (φ) between voltage and current.

$$PF = \frac{P}{S} = \frac{V \cdot I \cdot \cos\varphi}{V \cdot I} = \cos\varphi \tag{2.12}$$

The power factor of the induction machine is $PF = 0.76$ (Table 2.1). Figure 2.11 shows the stator voltage and current for phase a in order to compare the theoretical PF with the simulated result. From the figure it is seen that the current is lagging the voltage by $2ms$, therefore, calculating the corresponding angle it is obtain that the PF is:

$$\varphi = \frac{\Delta t}{T} \cdot 360^\circ = \frac{0.002s}{\frac{1}{58Hz}} \cdot 360^\circ = 41.76^\circ \rightarrow PF = \cos 41.76^\circ = 0.746 \tag{2.13}$$

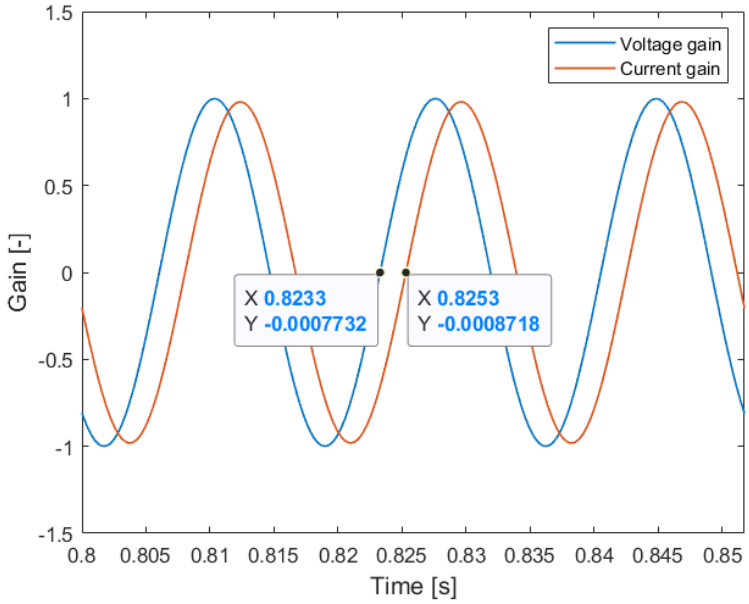


Figure 2.11: Stator phase *a* voltage and current gains for validation of nominal power factor.

2.2 Voltage source inverter

The 3 phase (3ϕ) full-bridge voltage source inverter (VSI) is one of the most critical components of our system. The inverter is responsible for receiving the command instruction from the logic processing unit and drive the motor accordingly. This 3ϕ inverter allows the full power to be delivered to the motor during driving operation and also provides the path for kinetic energy to rush back to the battery system during regenerative braking of go-kart.

Given the notorious behavior of semiconductor devices and their proneness to failure in a power electronics system [9][10], intensive care is needed to be taken while designing this inverter which was done by the esteemed group of year 2018 [3]. Rigorous design process was undertaken by 2018 group to ensure the normal functioning of inverter under rated as well as instantaneous peak operating conditions and, provided us the opportunity to focus primarily on the control algorithm and implementation on DSP. Figure 2.12 below shows a picture of the inverter developed.

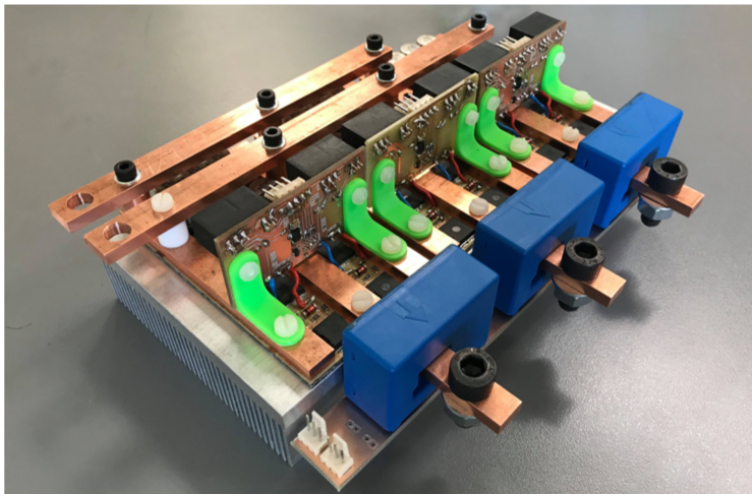


Figure 2.12: 3ϕ voltage source inverter developed by 2018 group.

2.2.1 Parallel operation of MOSFET

Based on motor electrical parameters mentioned in Table 2.1, under peak power condition of 7.3kW each leg of the inverter has to flow $263A_{rms}$ of current. This would result in a peak current value of $371A$. Even though the motor is expected to be operated at rated 5.3kW, the inverter is designed to work at peak power condition for worst case scenario. Low voltage MOSFET (<100V) with drain current capacity of above 300A is difficult to find. Hence the decision was made to parallel 2 MOSFETs and have them share the load. This design decision led to 12 MOSFETs for the 3ϕ full-bridge inverter with each leg containing 4 MOSFETs. 2 MOSFETs in parallel for high side and 2 more for the low side switch (Figure 2.13).

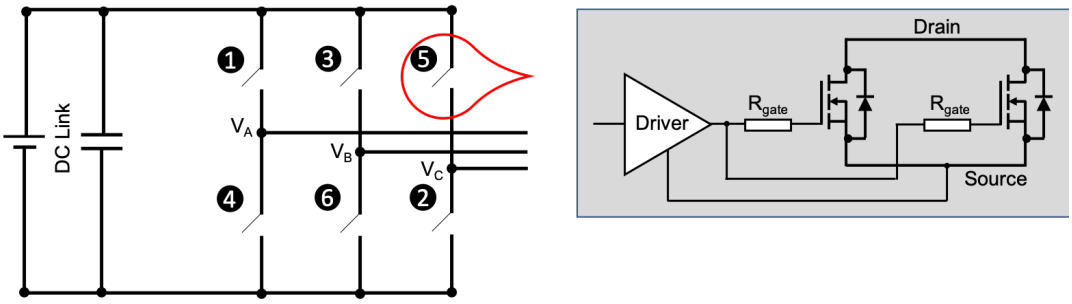


Figure 2.13: Parallel MOSFET design for load sharing.

2.2.2 Inverter thermal design

Knowing the MOSFET parameters, accurate estimation of loss at specific operating conditions can be obtained [11]. By using thermal conductivity information for each of the layer between MOSFET and heat sink, temperature expected on MOSFET's case can be calculated. Figure 2.14 below shows the structure of inverter PCB with heat sink and each of the layers in between.

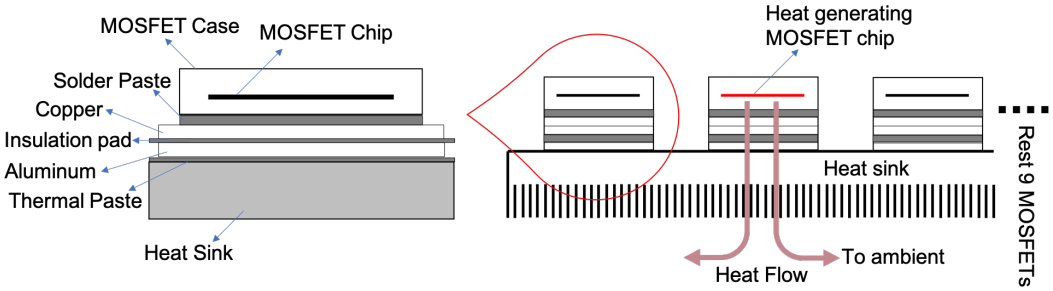


Figure 2.14: MOSFET heat dissipation via multiple intermediate layers.

The heatsink selected by 2018 group is 890SP-02000-A-100 which has a thermal resistance of $0.07°C/W$ when used with a Papst type 3312 ($12V$) cooling fan. Considering the thermal resistance of the rest the of layers as showcased in Figure 2.14 along with the estimated loss in the MOSFET, the temperature expected on the case of MOSFET can be calculated using the thermal equation 2.14 below. The subscripts of thermal resistances in Equation 2.14 stand for: JC (MOSFET junction to case), CS (case to solder paste), SC (solder paste to copper), CI (copper to insulation pad), IA (insulation pad to aluminum), AP (aluminum to thermal paste) and SA (heat sink to air). T_J and T_a represent MOSFET junction temperature and room air temperature, respectively. $P_{MOSLoss}$ represents MOSFET's power loss.

$$T_J - T_a = (R_{thJC} + R_{thCS} + R_{thSC} + R_{thCI} + R_{thIA} + R_{thAP} + R_{thSA}) \cdot P_{MOSLoss} \quad (2.14)$$

2.2.3 Test results of inverter

Even though the inverter was designed for peak operating conditions. Due to time constraints and availability of test equipment, the team from 2018 only managed to test one leg of the inverter at a maximum of 90A. At rated DC link voltage of 36V and switching frequency 20kHz each MOSFET is expected to generate 4.632W (simulation using datasheet values [12]) of loss which needs to be dissipated as heat. Using MOSFET loss and thermal resistance in Equation 2.14 MOSFET's case temperature is calculated to be at 31.2°C. Whereas the test results using a thermal camera (FLIR Thermal Camera) measured the temperature of 4 MOSFETS (of one leg) from 40°C - 42°C as shown below in Figure 2.15. Difference in thermal camera measurement and simulation results keeps on increasing as the current is increased . Hence improvement in measurement accuracy or another measurement method had to be employed for verification.

Maybe an extra line with the actual values in worst case scenario to show how the result is obtained. You can add this in the next subsection. AT

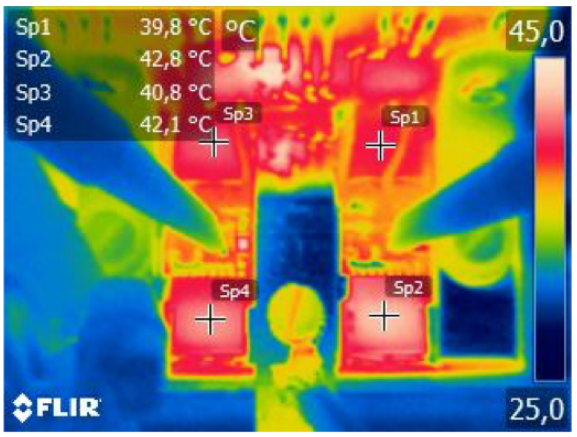
Thermal test results maybe? You only present here temperature results. Stef

you can say that the inverter was designed by the previous group to operate up to 20kHz.Stef

If you have the results from the simulation you could add them as appendixes, AT

don't think it's necessary to include the name. Stef

what do you mean with operation condition increased? Can we go higher than this operating condition?Stef. As current is increased FA. I thought you performed these measurements at rated current that's why I asked that but I didnt realize the table of Fig.2.16. Stef



Operating Condition

DC Link (V)	F_{sw}	I_{avg}	I_{pk}
36V	20kHz	89.8A	92.4A

MOSFET Case Temp.

Q1	Q2	Q3	Q4	Sim.
40°C	43°C	41°C	42°C	31.2°C

Figure 2.15: Thermal Camera output at 90A

When using a thermal camera, it is important to coat the heat emitting region with a high emissivity layer such as black paint [13]. Although a high emissivty coating can improve the measurement accuracy, it permanently damages the device under test. Hence it was decided to use a thermocouple as it would give a more focused measurement target location.

Ultimately, testing the inverter leg at load current of $175A_{avg}$ and DC link voltage of 12V (using motor winding as RL load) at switching frequency of 10kHz. Again using the datatsheet parameters the MOSFET is expected to generate 11.283W which translates to 44.13°C at chip junction and 39.6°C at the case using Equation 2.14. Using the thermocouple in contact with MOSFET case gives us result of 43.6°C, which is much closer to expected value as compared to thermal camera results of 61.0°C (Figure 2.16).

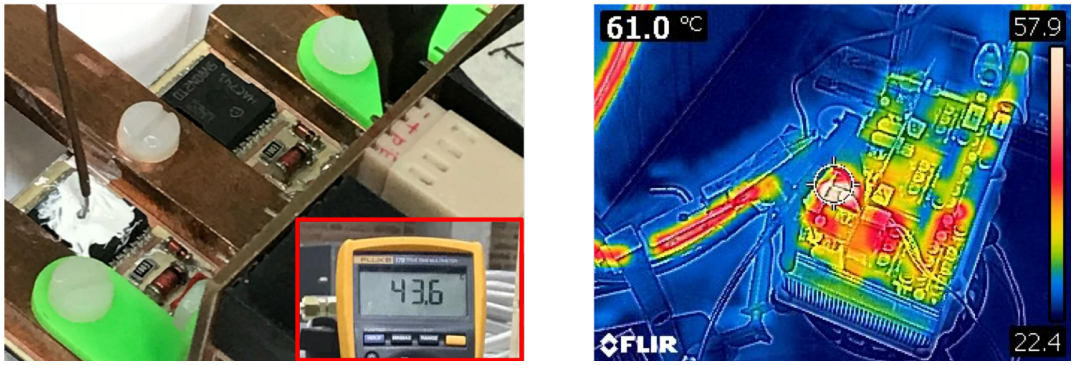


Figure 2.16: MOSFET case temp. measurement using *Left*: Thermocouple (Mulimeter reading embedded) *Right*: Thermal Camera output, At 175A load current.

After validating that thermocouple provides much more accurate results, all the legs of the inverter were tested under varying operating conditions, starting from very low DC link voltage and average drain current to rated operating condition of the induction motor (at 10kHz switching frequency). Figure 2.17 test results of Leg A of the inverter are shown. As it can be seen, simulation results closely follow the actual temperature measured using a thermocouple and are well within the maximum allowed temperature of the MOSFET .

which is? Stef

DC Link Voltage (V)	Average Drain Current (A)	Temp. (Thermocouple) (°C)	Temp. (Simulation) (°C)
12.5	55	24.8	26.8
18	100	28	30.3
24	152	37	36.7
33	172	44.8	40.3
38	190	50.5	42

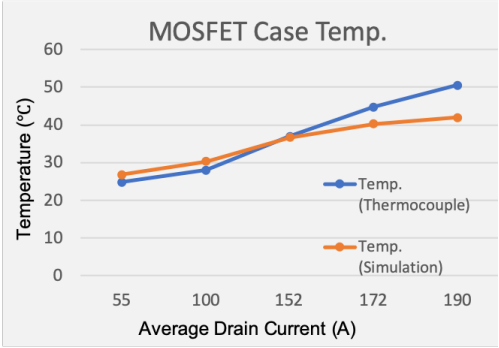


Figure 2.17: MOSFET case temperature of Leg A of the 3 phase inverter.

Nice, only the images are kind of unbalanced, I would place the table centered wrt the graph, other than that the chapter is looking good. AT

2.3 Battery pack

An important part of an electric go-kart is the power supply. Since the inverter is designed for a DC supply of $V_{DC} = 36V$, three lead-acid batteries of $12V$ are required. The batteries that will be used for powering the go-kart are high power YellowTop S 4,2 [14], which are sealed lead acid batteries produced by Optima. Some of the main parameters from the datasheet are presented in Table 2.2 and are specific for a single battery.

Table 2.2: Optima YellowTop parameters [14].

Parameter	Value
Nominal Voltage	$12V$
Rated capacity	$55Ah$
Internal resistance (fully charged)	$2.8m\Omega$
Open circuit voltage (fully charged)	$13.1V$
Weight	$19.5kg$

The main task of this project regarding the batteries is to create a dynamic model of them. Having this model provides advantages such as the ability to estimate the amount of energy that can be reintroduced in the battery when regenerative braking is occurring and how much time the batteries can be used at different current values.

2.3.1 Equivalent model of the batteries

Figure 2.18 represents the equivalent model of a battery and is composed of a voltage source (Vs), an internal resistance (Rs') and RC parallel blocks (transient parameters). The values of these parameters can be obtained by performing different experiments which will be described in the next subsections [15]. These parameters are dependent on the current flow, the state of charge (SOC) and the temperature. Depending on how precise one wants the model to be, the number of RC parallel blocks can be increased or decreased. A larger number of RC blocks

will lead to a better result [15]. In this analysis only one block will be used, thus an error is expected.

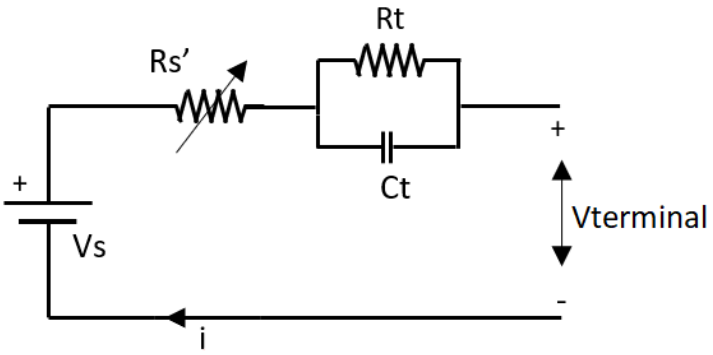


Figure 2.18: Equivalent circuit of internal structure of a battery.

2.3.2 Experiments for obtaining the battery parameters

Using the same batteries as the previous groups worked with, certain assumptions were made. First of all, four batteries were available but one of them was excluded as its capacity was the lowest according to previous tests [3]. However, the other three have been tested again since they have not been used for a long time. They are further referred as batteries A,B and D according to their labels. The required components for testing the batteries are listed below:

- Three $12V$ lead-acid batteries.
- One bi-directional power supply.
- One data acquisition device.
- One computer compatible with the data acquisition device.

The setup was created in the student laboratory at Aalborg University and is showed in Figure 2.19.

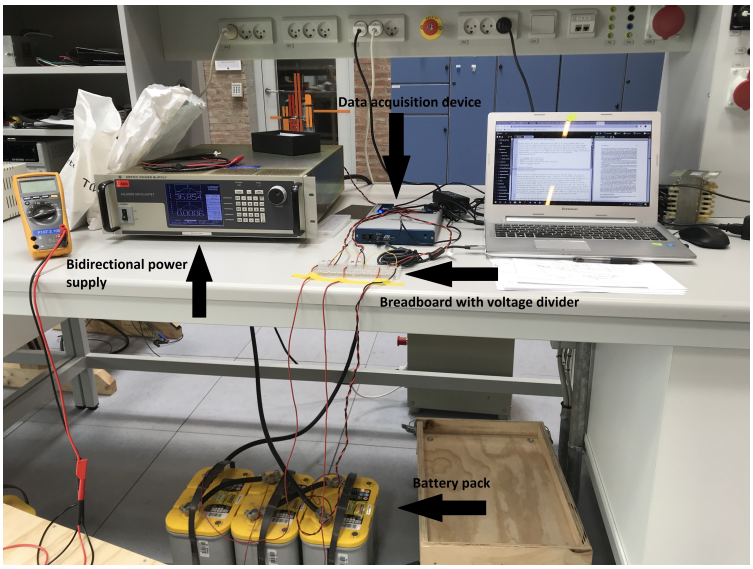


Figure 2.19: Setup for testing the batteries.

Two sets of parameters are of interest for each battery: the capacity and the values of internal impedance. In order to make the data acquisition of the voltages across the batteries and the output current of the power supply, the data acquisition device is used. This device has 16 analog input channels with the range of $\pm 10V$ but only two will be used. Because the voltage at the terminals of the batteries is higher than $10V$, a voltage divider was used in order to downscale the voltage. The analog output of the power supply is a signal with the amplitude of $10V$ and because the current applied is in the same range ($\pm 10A$), no scaling has to be done. Additionally, a thermocouple can be connected to the data acquisition device to measure the temperature during the experiments. After having the setup ready and the batteries fully charged the impedance test can be performed. Obtaining the values for the internal impedance is done by applying short current pulses of $-10A$ for approximately 25 seconds so the SOC of the battery is not modified and then the battery is left to rest. These current pulses can be applied at different values of SOC. By monitoring data during discharge, parameters such as the time constant (τ) or the voltage drop across the internal resistance ($V_{Rs'}$) can be obtained. The same procedure can be repeated while charging. There might be some small differences for measuring at different temperatures but this is not covered by the experiment.

Maybe you could also add an electric circuit diagram of the setup.Erik

Figure 2.20 shows the regions of interest in terms of parameters calculation. From the vertical jump of the voltage across the battery, when the current is interrupted, the series resistor Rs' can be calculated using Ohm's law. The time constant represents the exponential region of the voltage recovery curve and it represents the time the battery needs after the current was interrupted to reach V_{OC}. The transient resistance Rt can be obtained by measuring the open circuit voltage at the end of the recovery region, subtracting the value of $V_{Rs'}$ and divide by the current [16].

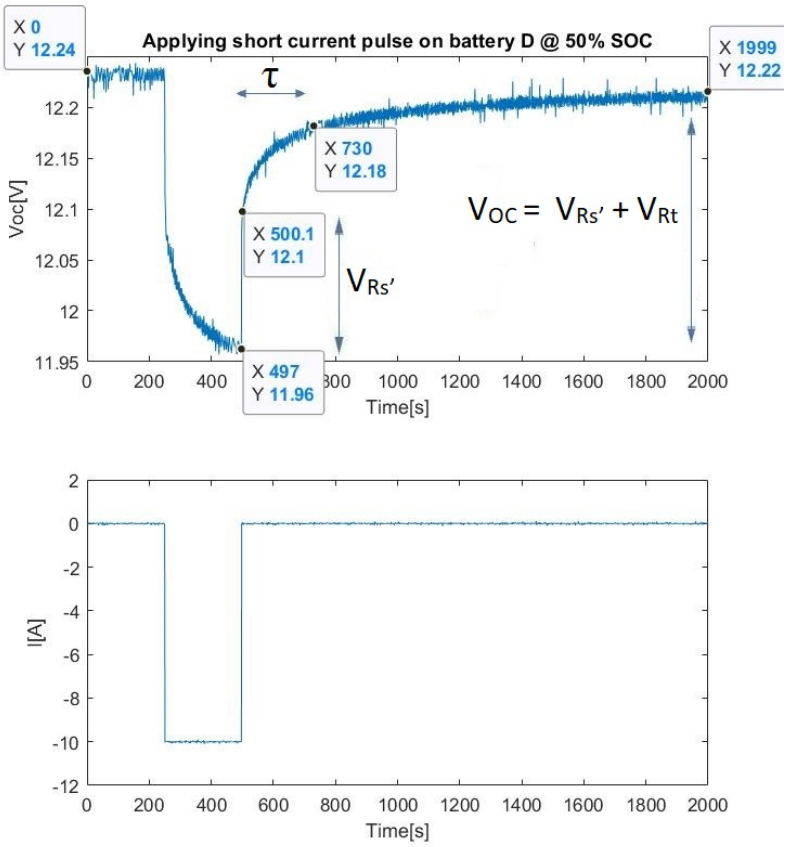


Figure 2.20: Example of how to obtain experimentally the internal impedance parameters of a battery.

In Figure 2.21, such short current pulses are represented. They were applied when the SOC of the batteries was 100%.

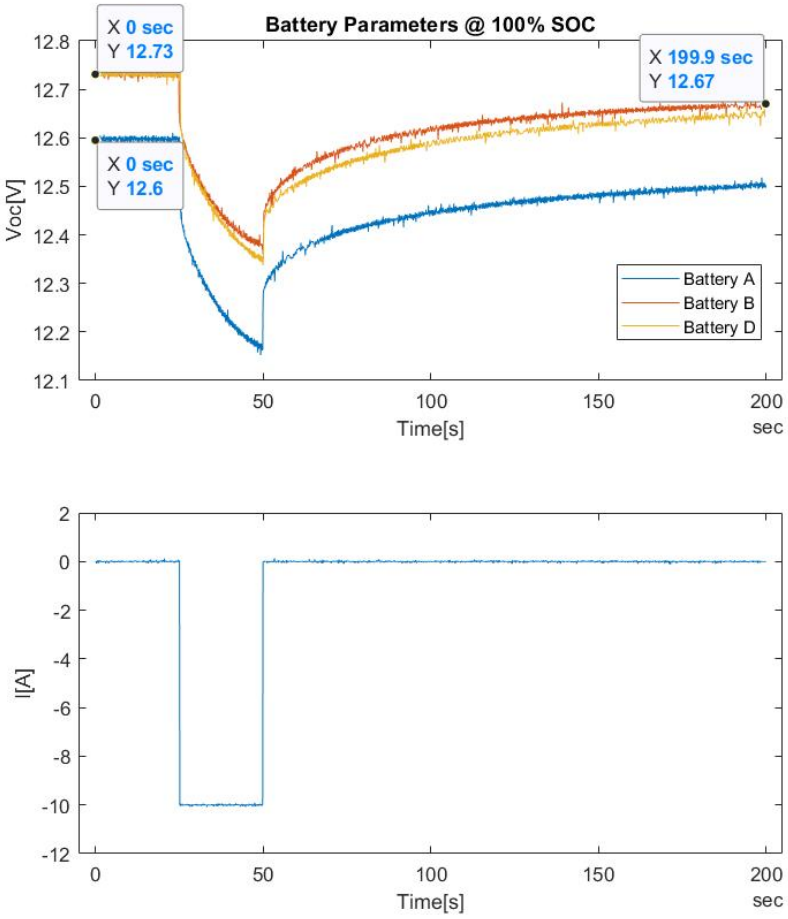


Figure 2.21: Applying short current pulses @ 100% SOC to determine the internal impedance values of the batteries.

Some observations can be made after this experiment. Battery A has a lower voltage level at 100% SOC and its internal resistance is bigger compared to the other batteries. Moreover, even if the current pulse was short, the V_{OC} did not return to the initial level which indicates that the data acquisition was not done for enough time. Table 2.3 contains the values obtained from the experiments shown in Figure 2.21 by applying Equation 2.15.

$$Rs' = \frac{V_{Rs'}}{I} \qquad Rt = \frac{V_{Rs'} - V_{Rt}}{I} \tag{2.15}$$

The value for τ for each battery was read with approximation directly from the curve which could lead to some errors. The value is also not exact because the data acquisition time was not long enough that the voltage across the batteries to reach V_{OC}, so τ is measured in Figure 2.20.

Table 2.3: Batteries parameters obtained experimentally @ 100% SOC.

Battery	Rs'	Rt	τ
A	$5.9m\Omega$	$10.8m\Omega$	250s
B	$3.4m\Omega$	$10.8m\Omega$	250s
D	$2.5m\Omega$	$11.6m\Omega$	250s

After some preliminary tests, it was observed that there is a noticeable difference between the batteries' capacity. Thus, a capacity test was conducted. The capacity of the batteries can be obtained by doing a complete charge-discharge cycle while monitoring the currents and the duration of the cycles. Figure 2.22 and 2.23 represents an example of curves obtained during a charge-discharge cycle. In this case, battery D has been monitored. The batteries were considered fully charged when the current reached $3-5\%$ of the Ah rating [17], while they were considered fully discharged when the voltage across their terminals reached $10.3V$ which is a value close to what datasheet provides. [14].

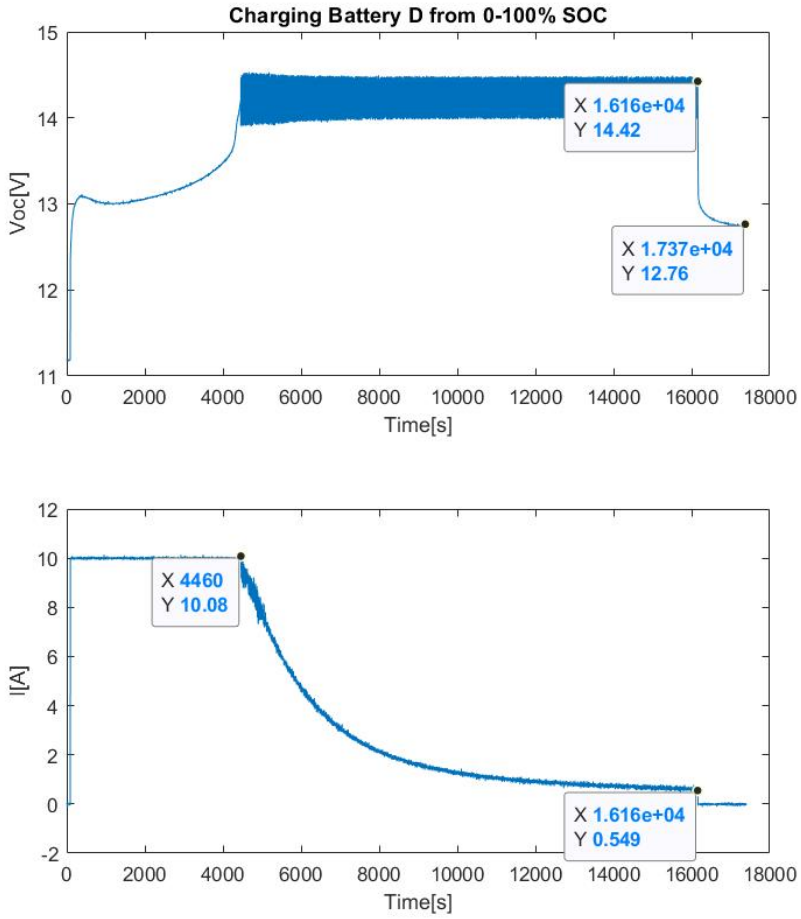


Figure 2.22: Charging of battery D 0-100% SOC for obtaining charging capacity.

From Figure 2.22 the charging capacity can be calculated by integrating the current versus time curve. This was done in MATLAB using a specific function (*trapz()*). Therefore, the charging capacity of battery D is 19.3Ah.

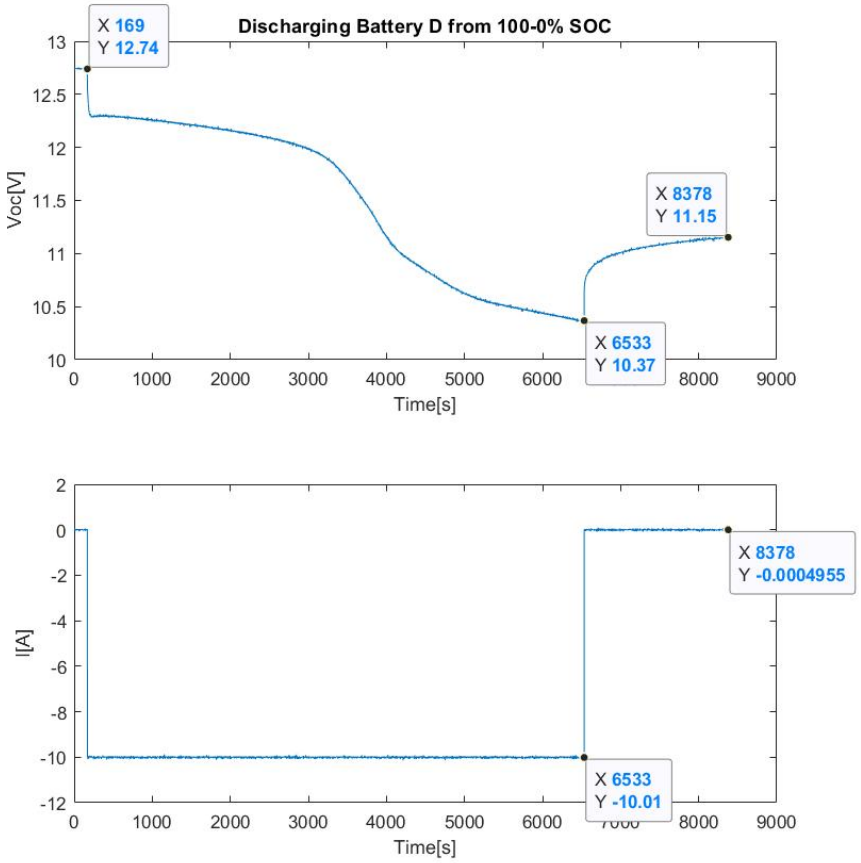


Figure 2.23: Discharging battery D 100-0% SOC for obtaining discharging capacity.

As seen in Figure 2.23 it takes 6363 seconds (1.7 hours) to discharge the battery with a constant current of 10A. Therefore, the discharging capacity is 17.8Ah.

Table 2.4 provides both the charging and the discharging capacity of all three batteries. The curves for battery A and B are presented in Appendix B.

Table 2.4: Values of the batteries charging and discharging capacity.

Battery	Charging capacity	Discharging capacity
Battery A	18Ah	14.4Ah
Battery B	35Ah	-
Battery D	19.3Ah	17.7Ah

Because the batteries will be used in a series configuration, the available capacity will be determined by the lowest one, which is around 14.4Ah. The application

they are supposed to be used in implies drawing currents higher than 150A and this will lead to fast discharge (less than 6minutes). Also, because of the difference in capacity the charging time is not the same for the three batteries. This will lead to different SOC for each particular battery while they are being used.

2.3.3 Simulation model

With the parameters being obtained, a simulation can be performed. In this case, a *Simulink* model was created by adapting the one suggested in [16].

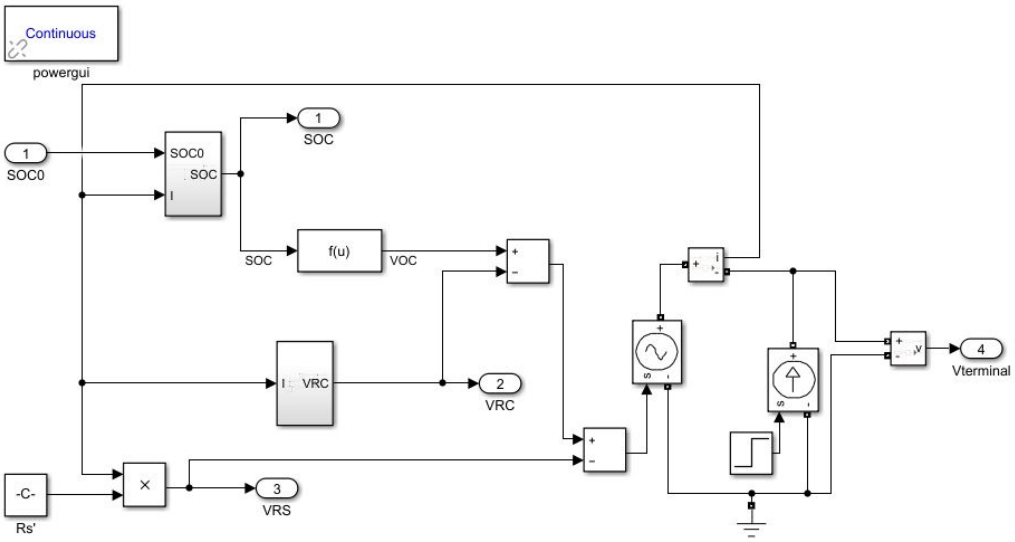


Figure 2.24: Dynamic model of a battery with current as an input and voltage as output.

The model contains 3 subsystems where the SOC, V_{OC} and the voltage across the Rt (V_{Rt}) are calculated. The real time SOC is calculated with Equation 2.16 [16].

$$SOC = SOC_0 - \int \frac{I \cdot 100}{c \cdot 3600} dt \tag{2.16}$$

Where SOC_0 is the initial SOC, I is the current and c is the capacity of the battery. V_{OC} is the open circuit voltage and is dependent on SOC by a polynomial equation. The coefficients of the polynomial equation were obtained by using the curve fitting function in MATLAB and the result can be seen in Figure 2.25.

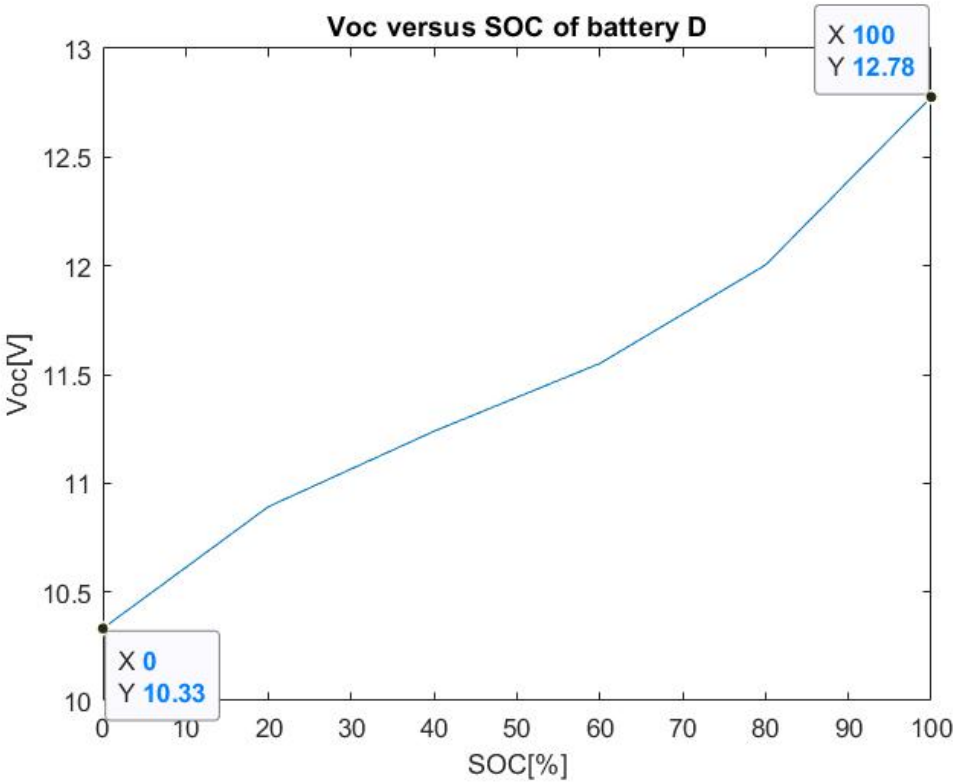


Figure 2.25: Calculated V_{OC} of battery D versus SOC using curve fitting function in MATLAB.

The voltage across the RC block and Rs' can be expressed through Equation 2.17 to 2.19 [16].

$$I = \frac{V_{Rt}}{Rt} + s \cdot Ct \cdot V_{Rt} \tag{2.17}$$

$$V_{Rt} = \left(\frac{1}{s}\left[\frac{I}{Ct} - \frac{V_{Rt}}{Rt \cdot Ct}\right]\right) \tag{2.18}$$

$$V_{Rs'} = I \cdot Rs' \tag{2.19}$$

The values of the capacity of the battery, Rs', Rt, and Ct are obtained experimentally as described above. Ct is obtained from the formula $\tau = Rt \cdot Ct$. By using the controllable current source situated at the output of the model, different current patterns can be created to simulate charging or discharging the battery at different values of the current. The voltage which can be seen at the terminal of the battery is calculated by Equation 2.20.

$$V_{terminal} = V_{OC} - V_{Rt} - V_{Rs'} \tag{2.20}$$

Do you consider this section on batteries as finish? Finally, I think some kind of model verification is needed. At least, can you make a simulation model which will give you the same results as the model was extracted from?. Erik, this is from the old version, take a look anyway

2.4 Interface board

In order to have good communication with the inverter, it was decided to create an interface board. This PCB has the objective of allowing an easy and real-time con-

trol over the software as well as including all the sensor circuitry and connections to external devices.

The development of the interface board was done in an early stage of the project since it was a necessary tool for debugging the software. For this reason, it was decided to include extra communication devices such as switches, buttons or LEDs in case they would ever be necessary. To decrease the number of floating wires between the interface board and the DSP, the Texas Instrument development kit [18] is also mounted to the PCB. In order to take advantage of the DSP's capabilities, most of the ports were used.

It was finally decided to include 20 LEDs for getting feedback from the firmware, 8 buttons and 6 switches for changing the control parameters and 5 potentiometers for analog inputs, 2 of which are sliders, useful for resembling the throttle pedal. On top of this, connectors for external components were included, 6 digital and 4 analog.

Apart from user communication, connectors for inverter were required. This consists on 3 connectors for the three phases, containing PWM signals and the enable, the hall sensors, which have on two individual connectors, power and signals, and the encoder that has a DE-9 connector. Also, connectors for voltage measurements are required. Finally, a power connector for 24V was also included.

2.4.1 PCB design

Since one of the main goals of the interface board is to allow easy interaction between the firmware, the inverter and the user, the location of the hardware on the PCB was the first design concern. A list of the components to be included and their use was made to then create the most user friendly layout. For this, the buttons, swithces, LEDs and potentiometers are located in the front part of the PCB, while connectors are placed on the sides. The launchpad is then located in the back part of the PCB as well as the DC-DC supplies needed for converting to $\pm 15V$, to $5V$ and to isolated $5V$. The launchpad featured less digital ports as the components that were going to be included. However, many of the included components were extra and might not be necessary, therefore, some of these were connected in parallel to each other through jumpers such that only one of them could be used at the same time. For example, an LED and a switch are connected to the same digital port, and a jumper decides which one of them is active.

The interface board was intended to be a very versatile device and one of its goals is the debugging of the firmware, for that reason, it was designed to allow an easy access to all the pins of the development board as well as to a ground plane, this feature makes it easy to measure signals with an oscilloscope. The launchpad requires a $5V$ supply and generates a $3.3V$ signal. However, while debugging, the launchpad can be powered directly from the computer. To avoid any kind of short-circuit, it was decided to isolate the $5V$ generated in the interface board from that coming from the computer.

In appendix C, the schematics and the layout of the PCB are shown.

2.4.2 PCB building

In order to find possible errors made during the design of the interface board, the assembly procedure of the board was carefully designed and it was followed during the build. Components were tested individually before being mounted. Then, the mounting procedure was also organized in sections that were tested after being soldered. This way, it was easy to find minor bugs that could be fixed during the building process.

In order to provide an easier interface, indications were included in switches, buttons and LEDs to show the function of each device. Figure 2.26 shows the finished interface board.

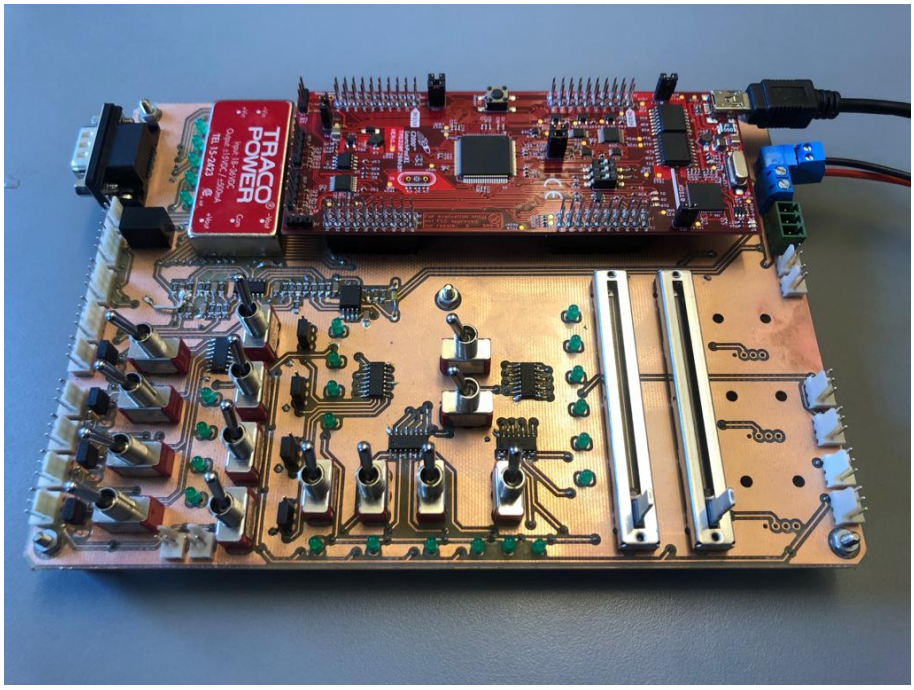


Figure 2.26: Interface PCB.

Consider changing this picture to add labels. Stef and AT

2.5 Mechanical model of the vehicle

In this section, the mechanical model of the go-kart is presented in order to obtain the forces that will affect the dynamic behaviour of the vehicle. There are four main forces that oppose the vehicle motion: the aerodynamic drag (F_{ad}), rolling resistance (F_{rr}), gravitational (F_g) and the force due to the inertia of the vehicle (F_i) . The free body diagram showing all the forces acting on the go-kart is depicted in Figure 2.27. Here the go-kart is moving up an inclined plane with an angle α.

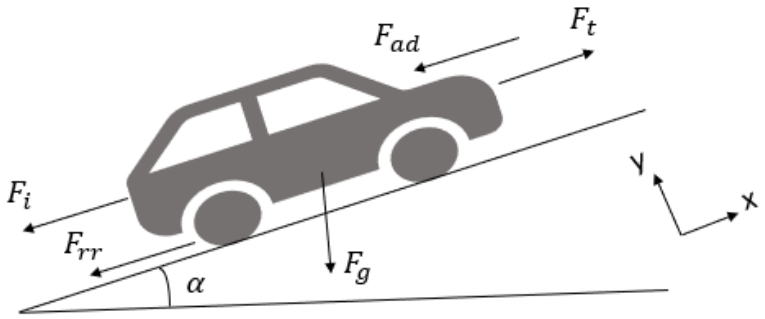


Figure 2.27: Free body diagram of the forces acting on the vehicle.

Applying Newton's second law in the direction of the vehicle's movement, it is obtained that the longitudinal traction force of the vehicle F_t is the sum of the four resistive forces (Equation 2.21). From this, it is observed that the difference between the traction force and the opposing forces (F_L) determines the acceleration or deceleration of the cart.

$$\sum F_x = M_{kart} \cdot a \rightarrow F_t = F_i + F_{rr} + F_{ad} + F_g \cdot \sin\alpha \tag{2.21}$$

The inertial force is the reaction force that appears when the vehicle is accelerating according to Newton's third law. It is defined as the mass of the go-kart by the acceleration ($F_i = M_{kart} \cdot a$), where $M_{kart} = 233$ Kg is the total mass of the go-kart including chasis, motor and driver.

did we finally stick to this weight? MR I guess so dont think we will weight it. Stef, i think that if we are not weighting it, we should say its an approximation and actually make an approximation, something like 250... 233 sounds a bit weird, AT. We took this weight from one of the previous groups which I guess weighted it. Stef

The rolling resistance force is the friction force that appears between the road and the wheels when they are rotating. This is because energy is consumed in the rotation process due to deformation of the wheels in the bottom when making contact with the road. F_{rr} is proportional to the normal gravitational force due to the weight of the vehicle. The rolling resistance coefficient (C_{rr}) also depends on the velocity of the vehicle [19].

$$\begin{aligned} F_{rr} &= C_{rr} \cdot F_g \cdot \cos\alpha \\ C_{rr} &= 0.01 \cdot (1 + 0.036 \cdot v_{kart}) \end{aligned} \tag{2.22}$$

When the vehicle is moving, the friction with the air is producing a drag force which is known as aerodynamic drag force. This force is shown in Equation 2.23 [19] and it depends on the fluid density ($\rho_{air} = 1.2041 \frac{Kg}{m^3}$), the aerodynamic drag coefficient ($C_{ad} = 0.804$), the vehicle frontal area ($A_f = 0.57 m^2$) and the speed of the vehicle. These parameters were taken from a previous year's group report [2].

$$F_{ad} = \frac{1}{2} \cdot \rho_{air} \cdot C_{ad} \cdot A_f \cdot v_{kart}^2 \tag{2.23}$$

Finally, the weight of the vehicle has a strong influence in the dynamic behaviour of the vehicle. The last term of equation 2.21 depends on the inclination α of the driving surface and on the total mass M_{kart} because the gravitational acceleration is a constant ($g = 9.81 m/s^2$).

The forces that oppose the movement of the vehicle are seen as a combined load force $F_L = F_{rr} + F_{ad} + F_g \sin\alpha$. Thus, Equation 2.21 can be written as,

$$M_{kart} \cdot a = F_t - F_L \tag{2.24}$$

In order to obtain Equation 2.24 by means of torques, which is what it is used to model the induction machine, the forces need to be multiplied by the radius of the wheel ($R_{wheel} = 0.1375m$). The result will be torques seen from the wheel axle, however, the torques for modelling the IM need to be the ones seen from the rotor axle. Thus, the previous result needs to be divided by the gear ratio G_r. The gear ratio can be obtained from the transmission system of the go-kart. From the number of teeth of the wheel and rotor sprockets:

$$G_r = \frac{\text{Number of teeth on axle gear}}{\text{Number of teeth on rotor shaft gear}} = \frac{40}{24} = 1.66 \tag{2.25}$$

Therefore, the maximum acceleration of the go-kart can be found from Equation 2.24, calculated on a flat surface ($\alpha = 0^\circ$) and at rated torque ($T_n = 30.04$Nm).

$$\frac{R_{wheel}}{G_r} \cdot (M_{kart} \cdot a) = \frac{R_{wheel}}{G_r} \cdot (F_t - F_L) = T_n - T_L \rightarrow \ddot{x} = 1.456 m/s^2 \tag{2.26}$$

The rated mechanical shaft speed is $\omega_n = 1685$rpm$= 176.45$rad/s, thus the maximum angular velocity the go-kart can achieve is:

$$v_{kart,max} = \omega_n \cdot \frac{R_{wheel}}{G_r} = 14.55 m/s = 52.38 km/h \tag{2.27}$$

3 Control of the induction machine

During this chapter the design of the control for the induction machine will be carried out. First, three modulation techniques for controlling the inverter will be presented and, after analyzing the advantages and disadvantages of each modulation scheme, space vector modulation technique will be implemented in this project. After the modulation, the controller for the IM will be designed. Indirect rotor flux field oriented control is the selected control strategy and the design procedure for the controller will be explained in detail. Finally, different types of regenerative braking will be presented and the capability of the serial regenerative braking will be analyzed.

3.1 Modulation

The objective of the modulation is to control the switching of the inverter in order to obtain the desired magnitude and frequency at the AC output voltage, which has to be close to a sine wave [20]. Pulse-width modulation (PWM) generates the pulses for the inverter switches by comparing a reference voltage waveform with a triangular voltage waveform of higher frequency. A wide variety of modulation techniques can be implemented for controlling the output voltage of power converters. In this section, three different pulse-width modulation (PWM) methods are introduced and their advantages and disadvantages are analyzed to select the most adequate modulation for this project. Sinusoidal PWM (SPWM), third-harmonic injection PWM (THIPWM) and space vector PWM (SVPWM) are the most commonly used modulation techniques for controlling three phase voltage source inverters (VSI)[7].

Sinusoidal PWM

Sinusoidal PWM generates the gate pulses for each of the inverter's legs by comparing a sinusoidal reference signal with a triangular carrier signal. In the case of three phase VSI, the reference signal consists of three sinusoidal phase voltages each shifted by 120° and at the desired fundamental frequency. The carrier signal is a triangular waveform which establishes the switching frequency of the VSI [7].

The switching of the semiconductor devices in each of the inverter legs is determined by the crossing points of these two waveforms. Thus, when the amplitude of the reference signal is greater than the amplitude of the carrier signal a high pulse is generated and the upper switch of the corresponding leg is turned on. The lower switch operation is complementary to the upper switch and hence it is turned off. The upper and lower switches on each of the inverter's legs are complimentary in order to avoid a short-circuit also known as shoot-through [7]. On the other hand, when the reference is lower than the carrier, a low pulse is generated and the upper switch is turn off. Figure 3.1 shows the operation of the SPWM for one of the inverter's legs.

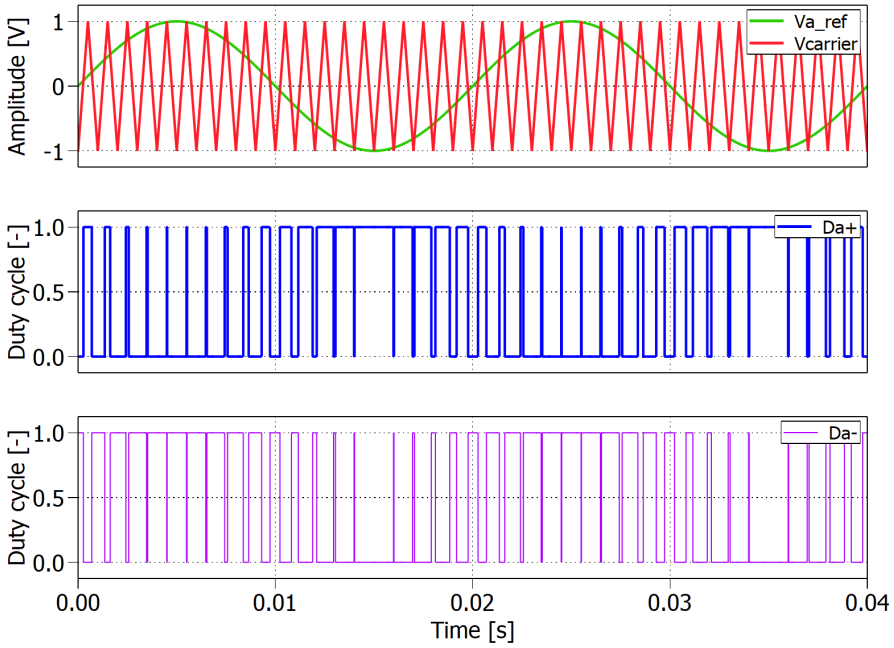


Figure 3.1: Sinusoidal PWM operation.

Third harmonic injection PWM

The third-harmonic PWM follows the same operation as SPWM of comparing a reference signal with a triangular signal to generate the gate pulses for the inverter. The difference is that the sinusoidal phase voltage reference is modified by adding the third harmonic component resulting in a flattened reference waveform shown in Figure 3.2.

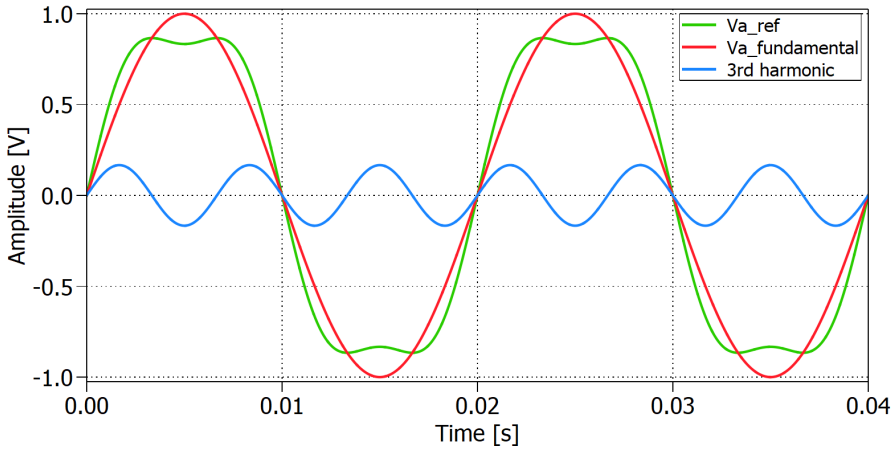


Figure 3.2: Third harmonic injection resultant reference waveform.

The third-harmonic component is equally added to each of the phase voltages, therefore, the line to line voltage remains undisturbed due to cancellation of the third harmonic. As a result of the third-harmonic injection, the peak value of the output voltage is decreased and at the same time keeping the amplitude of the fundamental unchanged. This way it is possible to increase the inverter's output voltage up to the fundamental voltage, which is approximately 15% higher than with SPWM without causing harmonic distortion of the line-to-line voltage [7].The reference waveform in THIPWM has lower amplitude than SPWM allowing a better utilization of the DC voltage.

Space vector PWM

Space vector modulation is a modulation technique that differs from the two previous PWM methods because it does not modulate each of the three phase voltages independently. SVPWM considers the inverter as a single unit by representing the three modulating voltages as vectors rotating in the counterclockwise direction in a two-dimensional ($\alpha\beta$) reference plane [7]. Using this modulation technique, as with THIPWM, it is possible to increase the inverter's equivalent voltage over that obtained using SPWM. This is because it reduces the peak amplitude of the modulating reference waveform, used for comparison with carrier waveform, resulting in a similar shape as with THIPWM (see Figure 3.3). However, the implementation is completely different, and the advantages and disadvantages of these three methods will be discussed in the next section.

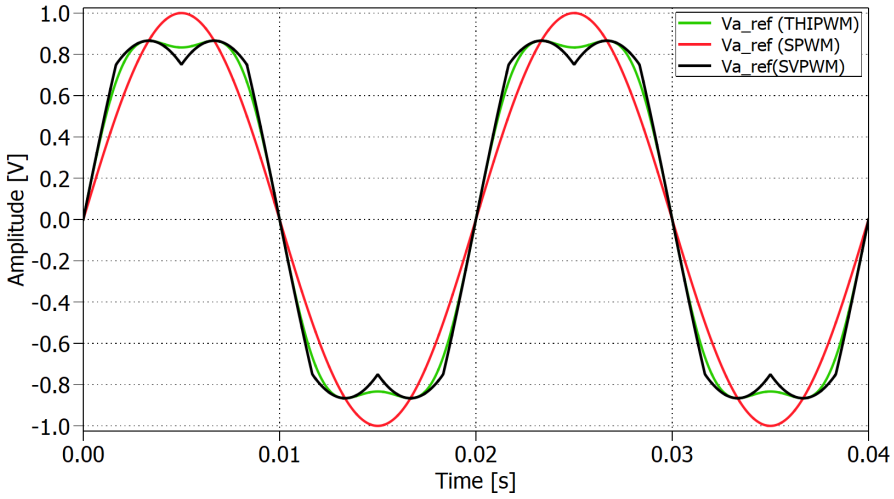


Figure 3.3: Space vector PWM resultant reference waveform compared with SPWM and THIPWM.

Why do SVPWM and THIPWM not have same maximum and minimum as SPWM?Erik

A VSI is controlled by six switches which allow the inverter to operate in eight different configurations according to the conduction states of the switches. Figure 3.4 shows the 8 possible configurations where '1' is used to represent the positive phase voltage level and '0' the negative. Therefore, each configuration can be represented as binary codes and has associated its corresponding space vector [7].

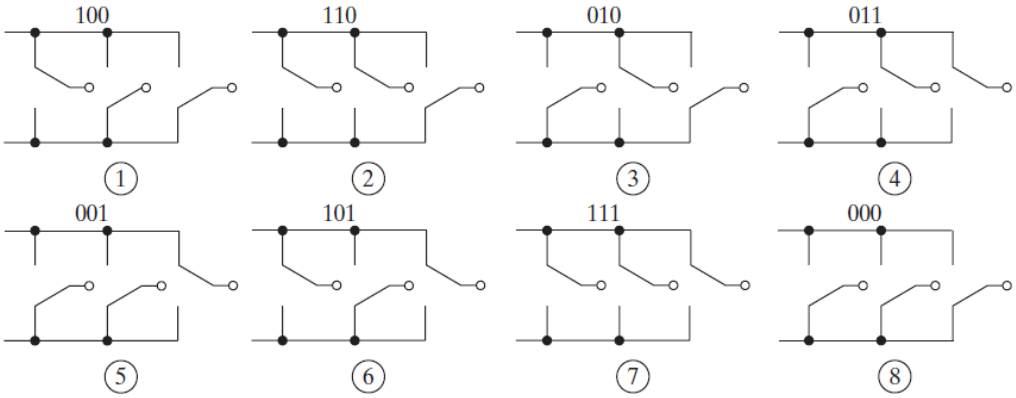


Figure 3.4: Switching states of a three phase voltage source inverter [7].

I think I saw somewhere that V1, V2 etc are in the same place as the correspondent binary number. Thus the order would be V4, V6, V2, V3, V1, V5. AT In the book I took the reference from they take it in the order I've done it but I remember we did in the order you mention in the lectures. Both cases are correct so let's just leave it as it is. Stef

The space vector representation is sketched in Figure 3.5, where vectors $V_1 - V_6$are called active vectors and are phase shifted by $\pi/3$. These active vectors have fixed magnitude ($2/3 \cdot V_{DC}$) and form an hexagon [7]. Thus, the space vector can be located in any of the six sectors which is bounded by two active vectors. When the three upper or the three lower switches are conducting simultaneously, the DC supply line is short circuited resulting in the zero vectors (V_0 and V_7). These zero vector are at the origin of the hexagon and have zero amplitude. The eight space vectors are seen as stationary vectors because they do not rotate [7].

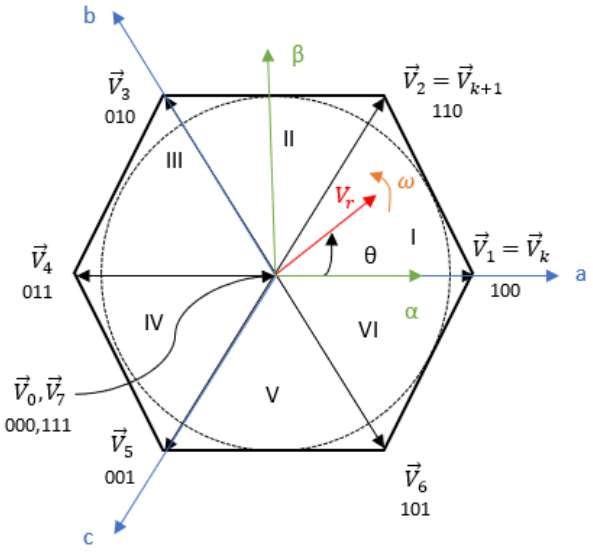


Figure 3.5: Space vector representation with reference voltage vector located in sector I.

SVPWM operation is based on the rotation of a reference voltage vector $\vec{V}_r$ in the anticlockwise direction in a $\alpha\beta$ reference plane to generate the desired three phase voltages at the output of the VSI. The magnitude of V_r depends on the magnitude of the output voltage and the frequency of rotation is the same as the fundamental frequency [7]. Any space vector can be defined, according to its sector location, by the two adjacent active vectors ($\vec{V}_k$ and $\vec{V_{k+1}}$) and the zero vectors ($\vec{V}_0$ and $\vec{V}_7$). This way the switching sequence is determined to obtain the reference voltage waveform which will be compared with a triangular waveform in order to generate the pulses for the inverter's switches.

3.1.1 Selection of modulation technique

The three previous discussed modulation techniques are the most commonly used methods for controlling three phase voltage source inverters. SPWM presents the simplest implementation and thus is the most widely used for voltage control. However, this modulation technique has the disadvantage that the inverter's equivalent voltage cannot exceed the DC supply voltage without entering the overmodulation region [7]. Overmodulation is reached when the amplitude of the reference voltage waveform is higher than the carrier's waveform amplitude which means that the relationship between the fundamental reference voltage component and the DC supply voltage becomes non-linear [7]. The overmodulation region is avoided in most applications that require low distortion because it adds more harmonics to the system.

THIPWM is used to compensate for this drawback as it adds a third harmonic component to the reference voltage waveform allowing an increase of approximately up to 15% of the DC supply voltage. This modulation technique is preferred in three phase applications due to cancellation of the third harmonic component

leading to lower total harmonic distortion (THD) than with SPWM [7].

In contrast with SPWM and THIPWM, space vector PWM considers the inverter as a single unit and it requires reference frame transformations and, thereby, more complicated mathematical operations. Nevertheless, SVPWM has the advantage of lower THD than SPWM in both output voltages and currents applied to the induction machine [4]. Similar to THIPWM, this technique allows up to 15% increase in the output voltage compared with the conventional SPWM [7]. In addition to the advantage of lower THD and more efficient use of the DC link voltage, SVPWM can be implemented easily with DSP-based control systems [7][4]. Therefore, SVPWM is the modulation technique selected for this project due to its flexibility to be implemented digitally and because it presents lower THD than other modulation schemes.

3.1.2 Implementation of SVPWM

The space vector modulation scheme is implemented in this section to obtain the gate signals for the VSI. Figure 3.6 shows a block diagram with the subsystems used for validating the modulation scheme in Simulink/Matlab. Three phase sinusoidal voltages are transformed from abc reference frame to $\alpha\beta$ frame by using Clarke's transformation. The $\alpha\beta$ voltages are the input for the SVPWM subsystem which will output the duty cycles for the inverter, which is connected to the induction machine.

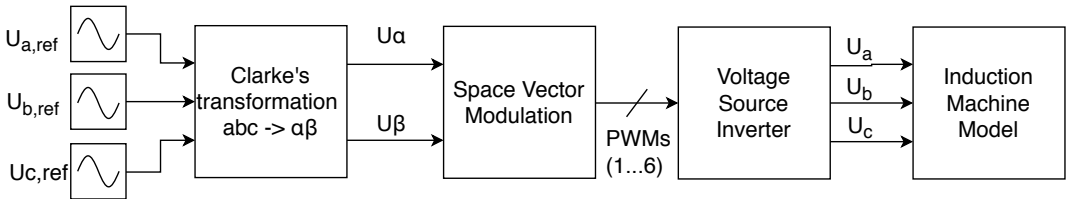


Figure 3.6: Block diagram for SVPWM implementation.

The implementation of SVPWM is carried out by following the next steps [7]:

1. Clarke's transformation for the three phase sinusoidal reference signals in order to obtain the components of the reference vector in a two coordinate $\alpha\beta$ stationary reference frame.

2. Calculation of the magnitude V_r and the angle θ, with respect to the phase a axis, of the reference vector.

3. Determination of the sector number ($k = 1, 2, ..., 6$), in which the reference vector lies, and the angle γ with respect to the adjacent active vector ($\vec{V}_k$) of the corresponding sector.

4. Calculation of the time intervals for which the active vectors ($\vec{V}_k$ and $\vec{V_{k+1}}$) and the zero vectors ($\vec{V}_0$ and $\vec{V}_7$) are required to be on during one pulse-width modulation period.

5. Determination of the switching sequence to obtain the reference signal that will be compared with the triangular wave to generate the corresponding gate signals for the VSI.

Clarke's transformation

The first step in the SVPWM implementation is the transformation of the three phase reference voltages from the abc reference frame to a two-dimensional reference frame. These two phase variables, which are stationary and orthogonal, are denoted as α corresponding to the real axis (aligned with phase a) and β for the imaginary axis. Therefore, the reference vector $\vec{V_r}$ is defined as:

$$\vec{V_r} = V_\alpha + jV_\beta = \frac{2}{3} \cdot \left[V_a \cdot e^{j0} + V_b \cdot e^{j\frac{2\pi}{3}} + V_c \cdot e^{-j\frac{2\pi}{3}} \right] \tag{3.1}$$

Applying Euler's formula (Equation 3.2) to Equation 3.1 and equating real and imaginary terms, the Clarke's transform is defined as in Equation 3.3.

$$e^{j\theta} = cos\theta + jsin\theta \tag{3.2}$$

$$\begin{bmatrix} V_\alpha \\ V_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} \tag{3.3}$$

Magnitude and angle of $\vec{V_r}$

Figure 3.5 shows the space vector representation for the reference vector which rotates at an angular speed $\omega = 2\pi f$ in the $\alpha\beta$ frame, where f is the fundamental frequency of the inverter's output voltage. The space vector representation is divided in six sectors shifted by 60° forming an hexagon. Taking as an example the reference vector located in sector I, this vector is bounded by two active vectors ($\vec{V_k} = V_1$ and $\vec{V_{k+1}} = V_2$) and two zero vectors (V_0 and V_7) located at the origin of the hexagon. Depending on the sector the reference vector is passing through, different sets of switches will be turned on and off in the VSI.

The rotating reference vector $\vec{V_r}$ is defined as in Equation 3.4 where V_r is the magnitude of the vector and θ is the angle with respect to the phase a-axis.

$$\vec{V_r} = V_r \cdot e^{j\theta} \tag{3.4}$$

The $\alpha\beta$ voltage components obtained in the previous step are used to compute the scalar magnitude of the reference vector and the angle θ as stated in equations 3.5 and 3.6, respectively.

$$V_r = \sqrt{{V_\alpha}^2 + {V_\beta}^2} \tag{3.5}$$

$$\theta = \arctan(\frac{V_\beta}{V_\alpha}) \tag{3.6}$$

Sector determination and angle γ

The reference vector $\vec{V}_r$ can lie in any of the six sectors that define the hexagon shown in Figure 3.5. By using the angle θ its position can be determined as stated in Table 3.1.

Table 3.1: Sector determination from the reference vector angle.

Angle	Sector Number
$0^\circ \leqslant \theta < 60^\circ$	1
$60^\circ \leqslant \theta < 120^\circ$	2
$120^\circ \leqslant \theta < 180^\circ$	3
$180^\circ \leqslant \theta < 240^\circ$	4
$240^\circ \leqslant \theta < 300^\circ$	5
$300^\circ \leqslant \theta < 360^\circ$	6

The angle γ is the angle between $\vec{V}_r$ and the adjacent active vector $\vec{V}_k$ at a particular time. From the corresponding sector number, where the reference vector falls, and the angle of the reference vector, θ, it is possible to calculate this angle as in Equation 3.7. This angle can take values from 0° up to 60° and will be used in the next step for calculating the time during each of the active vectors should be on.

$$\gamma = \theta - (k-1) \cdot \frac{\pi}{3} \tag{3.7}$$

Calculation of the dwell times

Depending on the position of the reference vector, the active vectors that bound the corresponding sector should be active during certain time intervals which are known as dwell times. These time intervals are calculated based on the magnitude and angle of $\vec{V}_r$ and the active vectors. In a general form, the active vectors are defined as in Equation 3.8 and the reference vector as in Equation 3.4.

$$\vec{V}_k = \begin{cases} \frac{2}{3} \cdot V_{DC} \cdot e^{j(k-1)\frac{\pi}{3}} & \text{if k=1,2,...,6} \\ 0 & \text{if k=0,7} \end{cases} \tag{3.8}$$

I left a comment before saying that maybe the order of the vectors wasn't from 1 to 6, if so, change this as well. AT

As an example, if the reference vector is in sector I, the active vectors are $\vec{V}_k = \vec{V}_1$ and $\vec{V_{k+1}} = \vec{V}_2$. This means that $\vec{V}_1$ and $\vec{V}_2$ will be active in a sampling period during T_1 and T_2, respectively. The zero vectors will be active during a time interval T_0. These dwell times can be determined by applying volt-sec balance using the reference vector and the two adjacent active vectors [7]. The vector diagram showing the dwell times when the reference vector is located in sector I is depicted in Figure 3.7.

$$\vec{V}_r \cdot Ts = \vec{V}_1 \cdot T_1 + \vec{V}_2 \cdot T_2 \tag{3.9}$$

Substituting the corresponding expressions for the reference vector and the active vectors (equations 3.4 and 3.8) in the previous equation it is obtained:

$$V_r \cdot e^{j\theta} = d_x \cdot \frac{2}{3} \cdot V_{DC} \cdot e^{j0} + d_y \cdot \frac{2}{3} \cdot V_{DC} \cdot e^{j\frac{\pi}{3}} \tag{3.10}$$

where $d_x = \frac{T_1}{T_s}$ and $d_y = \frac{T_2}{T_s}$ are the ratio between the dwell times (T_1 and T_2) and the switching period (T_s). Solving Equation 3.10 by applying Euler's equation and equating real and imaginary terms:

$$V_r \cdot cos\theta = d_x \cdot \frac{2}{3} \cdot V_{DC} + d_y \cdot \frac{2}{3} \cdot V_{DC} \cdot cos\frac{\pi}{3} \tag{3.11}$$

$$j \cdot V_r \cdot sin\theta = j \cdot d_y \cdot \frac{2}{3} \cdot V_{DC} \cdot sin\frac{\pi}{3} \tag{3.12}$$

Thus, d_x and d_y corresponding to a reference vector lying in sector I is calculated as shown in equations 3.13 and 3.14. These will be used for finding the corresponding duty cycle for each of the inverter's legs.

$$d_x = \frac{\sqrt{3} \cdot V_r}{V_{DC}} \cdot sin(\frac{\pi}{3} - \theta) \tag{3.13}$$

$$d_y = \frac{\sqrt{3} \cdot V_r}{V_{DC}} \cdot sin\theta \tag{3.14}$$

Equations 3.13 and 3.14 can be written in a general form, valid for all the sectors, as:

$$d_x = \frac{T_1}{T_s} = \frac{\sqrt{3} \cdot V_r}{V_{DC}} \cdot sin(k\frac{\pi}{3} - \theta) \tag{3.15}$$

$$d_y = \frac{T_2}{T_s} = \frac{\sqrt{3} \cdot V_r}{V_{DC}} \cdot sin\gamma \tag{3.16}$$

In general, the zero vectors $\vec{V_0}$ and $\vec{V_7}$ will be active during the switching period for the same amount of time as depicted in Figure 3.8.

$$T_0 = T_s - T_1 - T_2 \tag{3.17}$$

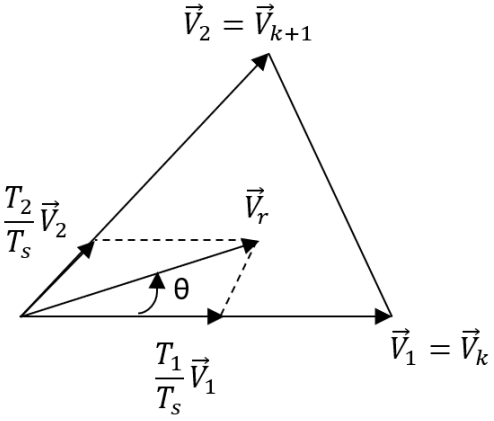


Figure 3.7: Vector diagram for calculation of dwell times in sector I.

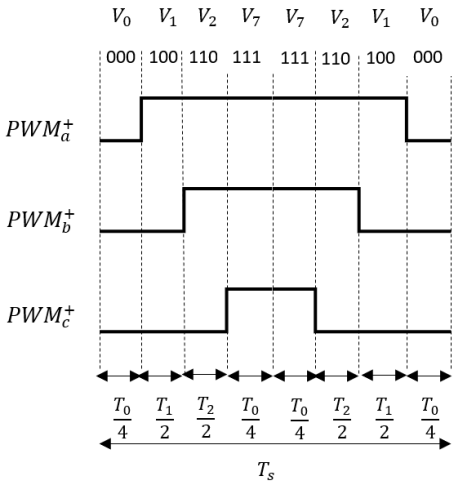


Figure 3.8: 7-segment switching sequence for sector I.

Switching sequence

The switching sequence for the space vector modulation scheme is arranged in a symmetrical manner around the center of the switching period. This is done in order to obtain at the inverter's output voltages with quarter-wave symmetry to ensure a reduction of the THD [7].

For obtaining the symmetric switching sequence the 7-segment technique is implemented in this project. The switching pattern consists of a zero vector followed by the two adjacent active vectors and ends with the other zero vector to complete half a switching period. The next half period is the mirror image of the first half as shown in Figure 3.8. This figure shows the case of a modulating reference vector that falls in sector 1 in order to continue the previous example. It is observed that the sequence is $\vec{V}_0 \rightarrow \vec{V}_1 \rightarrow \vec{V}_2 \rightarrow \vec{V}_7$ for the first half period and $\vec{V}_7 \rightarrow \vec{V}_2 \rightarrow \vec{V}_1 \rightarrow \vec{V}_0$ for the second half.

The switching sequence is selected in this way to make sure that the transition from state to state involves the minimum number of switching. Thus, it is possible to reduce the switching losses of the inverter because each of the switches turns on and off only once per switching period. Therefore, by using the 7-segment switching pattern center-aligned pulses are obtained allowing a reduction in the THD and switching losses. Table 3.2 shows the switching sequence for all the six sectors.

Table 3.2: Switching sequence for SVPWM.

Switching sequence	Sector
$\vec{V}_0 \rightarrow \vec{V}_1 \rightarrow \vec{V}_2 \rightarrow \vec{V}_7 \rightarrow \vec{V}_2 \rightarrow \vec{V}_1 \rightarrow \vec{V}_0$	1
$\vec{V}_0 \rightarrow \vec{V}_3 \rightarrow \vec{V}_2 \rightarrow \vec{V}_7 \rightarrow \vec{V}_2 \rightarrow \vec{V}_3 \rightarrow \vec{V}_0$	2
$\vec{V}_0 \rightarrow \vec{V}_3 \rightarrow \vec{V}_4 \rightarrow \vec{V}_7 \rightarrow \vec{V}_4 \rightarrow \vec{V}_3 \rightarrow \vec{V}_0$	3
$\vec{V}_0 \rightarrow \vec{V}_5 \rightarrow \vec{V}_4 \rightarrow \vec{V}_7 \rightarrow \vec{V}_4 \rightarrow \vec{V}_5 \rightarrow \vec{V}_0$	4
$\vec{V}_0 \rightarrow \vec{V}_5 \rightarrow \vec{V}_6 \rightarrow \vec{V}_7 \rightarrow \vec{V}_6 \rightarrow \vec{V}_5 \rightarrow \vec{V}_0$	5
$\vec{V}_0 \rightarrow \vec{V}_1 \rightarrow \vec{V}_6 \rightarrow \vec{V}_7 \rightarrow \vec{V}_6 \rightarrow \vec{V}_1 \rightarrow \vec{V}_0$	6

Based on the dwell times and the switching sequence, the modulating reference signals for each of the three phases are generated and compared with the triangular carrier signal in order to generate the duty cycles for the inverter's switches. Following the previous example ($\vec{V}_r$ in sector I), the modulating reference signals are obtained from the dwell times by setting the condition that the ON time for the zero vectors has to be the same in one sample period. Then, from the 7 segment switching sequence of Figure 3.8 it is possible to obtain the relationship between duty cycles and the dwell times resulting in Equation 3.18.

$$\begin{aligned} 1 &= D_a + D_c \\ d_x &= D_a - D_b \\ d_y &= D_b - D_c \end{aligned} \tag{3.18}$$

Writing this equation in matrix form and, calculating the inverse matrix, it is possible to generate the reference signals from the dwell times. The same procedure is followed for the six sectors.

$$\begin{bmatrix} 1 \\ d_x \\ d_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} D_a \\ D_b \\ Dc \end{bmatrix} \rightarrow \begin{bmatrix} D_a \\ D_b \\ D_c \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ d_x \\ d_y \end{bmatrix} \tag{3.19}$$

3.1.3 Simulation results

In this section the simulation results obtained when implementing the SVPWM in Simulink will be presented in order to validate the modulation scheme. From Figure 3.9 it can be seen the three modulating reference signals, which are compared with the carrier signal, together with the sector where the reference vector is located. These reference signals, compared with the sinusoidal reference waveforms, have lower amplitude resulting in a better utilization of the supply voltage as it was explained in subsection 3.1. As an example, a zoomed view for a time instant when the modulating signal is in sector I is seen in Figure 3.10. From this figure it can be validated the 7-segment technique for obtaining a symmetric switching sequence.

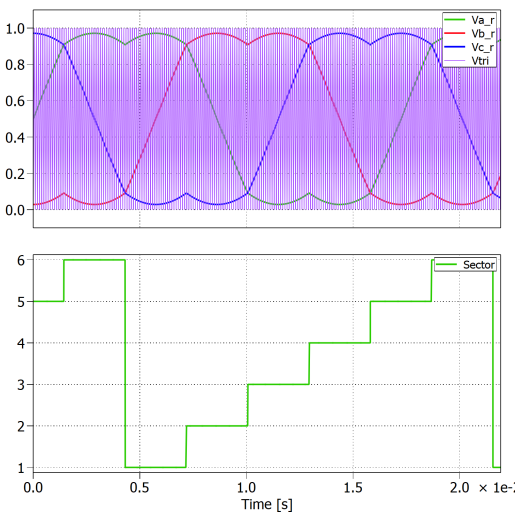


Figure 3.9: *Top graph:* Reference signals for the three phases compared with the carrier signal. *Bottom graph:* Sector number.

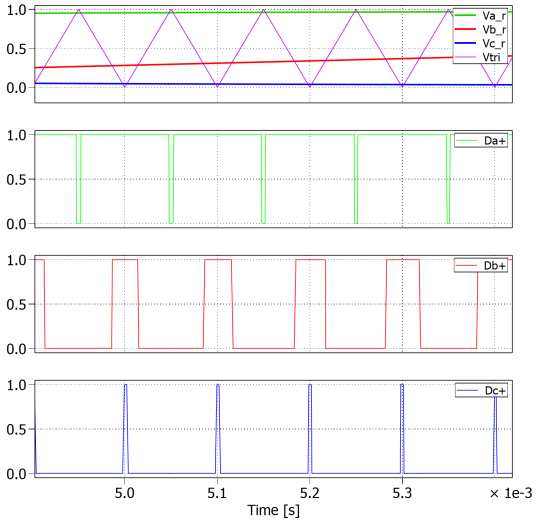


Figure 3.10: Zoomed view of the generation of the duty cycles for each of the inverter's legs in the case of $\vec{V}_r$ located in sector I.

Figure 3.11 shows the voltage for phase a and the line to line voltage between phase a and phase b at the inverter's output together with their corresponding sinusoidal reference signals. These sinusoidal reference signals are included in the system as inputs to model the voltages in the abc reference frame which are transformed to the $\alpha\beta$ through Clarke's transformation. However, at this point there is no controller included in the system as it is shown in the block diagram in Figure 3.6. From Figure 3.11 it is observed that, in both cases, the inverter's output voltages have quarter-wave symmetry in order to reduce the harmonic distortion. These voltages are in phase with the sinusoidal reference voltages. Moreover, it is observed that the inverter's output voltage has a maximum amplitude of $\frac{2}{3}V_{DC} = 24V$ and it also has intermediate values of $\frac{1}{3}V_{DC} = 12V$ as it was shown in section/appendix "blablabla". In the case of the line to line voltage, it varies between $-V_{DC} = -36V$ and $V_{Dc} = 36V$, as expected.

Remove blablabla...

In the VSI section in chapter "Power train" or in appendix include the analysis of the phase and line voltages (as explained in the lectures) in order to refer here to it to show that the phase voltages can take values (2/3Vdc, -2/3Vdc, 1/3Vdc and -1/3Vdc) and the line to line voltages vary from -Vdc to Vdc. Stef

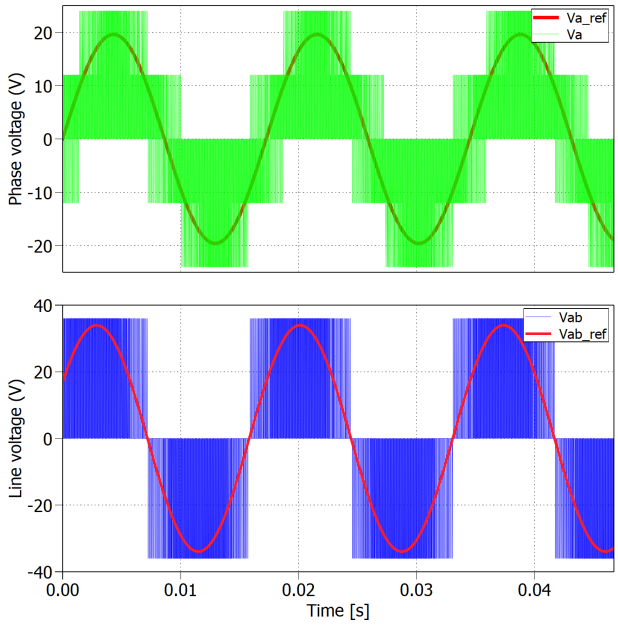


Figure 3.11: *Top graph:* Sinusoidal phase a reference voltage V_{aref} compared with the phase a voltage V_a at the inverter's output. *Bottom graph:* Sinusoidal reference line to line voltage V_{abref} compared with the line to line voltage V_{ab} at the inverter's output.

Finally, for validating the SVPWM the induction machine's behaviour is analyzed. The nominal parameters of the IM are listed on Table 2.1. By applying a step load torque of rated value $T_{load} = 30.4Nm$ at $0.3s$, the electromagnetic torque, shaft speed and phase currents are obtained and plotted in Figure 3.12. The electromagnetic torque T_e follows the load torque reaching its rated value and the rotor speed reaches a steady state value of 1681 rpm which is approximately the nominal rotor speed of 1685rpm. A slip value of 3.16% is obtained which is equal to the slip obtained when validating the IM parameters in section 2.1.3. The peak value of the stator phase currents after applying the step load torque is $264.5A$. The nominal phase current is 189A which corresponds to a peak value of $189A \cdot \sqrt{2} = 267.28A$. Therefore, the rated parameters of the IM are reached after applying the rated load torque and, consequently, the SVPWM can be validated.

So this is actually a combined SVPWM and machine model validation. Another validation could be to plot the alpha-beta stator flux on each axis in no-load situation. You should then see a nice circle with the same amplitude as the stator voltage vector (no-load -> almost zero current -> no ohmic voltage drop). Erik

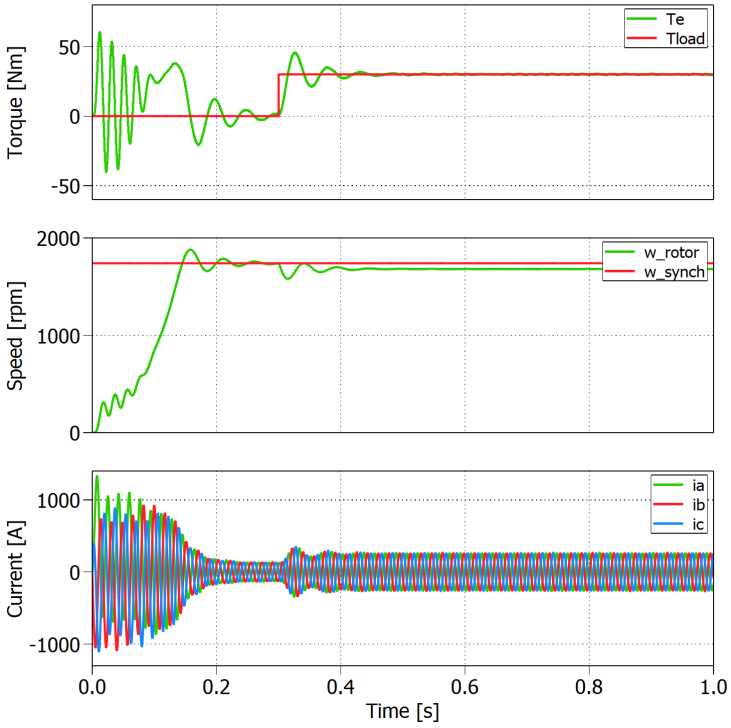


Figure 3.12: *Top graph:* Electromagnetic torque T_e and load torque T_{load}. *Middle graph:* Speed of the rotor and rated synchronous speed. *Bottom graph:* Stator phase currents i_a, i_b and i_c.

3.2 Field oriented control

Field oriented control (FOC), also known as vector control, is widely used for controlling the speed of induction machines due to its high dynamic performance and reliability [7][21]. The principle of the FOC technique is based on the decoupling of the stator currents into two components in order to achieve independent control of the flux and the torque [7][21]. The decoupling of the stator currents is done by implementing Clarke's and Park's transformation ($abc \rightarrow \alpha\beta \rightarrow dq$) to the stator currents to obtain a two phase rotating dq reference frame aligned with the flux vector[7]. It is important to note that the dq system is rotating synchronously at electrical frequency. Usually, the dq reference frame is used for control purposes and the stationary $\alpha\beta$ frame for modelling, as it is done in section 2.1.2.

The alignment of the dq reference frame with the flux vector can be carried out in different ways depending on which flux vector is chosen to be oriented. Two main techniques can be implemented: stator flux field oriented control and rotor flux field oriented control [4]. Stator flux FOC forces the stator d-axis current to be oriented with the stator flux linkage while rotor flux FOC forces the d-axis to be constantly aligned with the rotor flux linkage. The latter is the selected technique to be implemented in this project and its principle is explained in detail in subsection 3.2.2. Rotor flux FOC is selected because it has shown superior dynamic performance than stator flux FOC [22]. Stator flux, in comparison with rotor flux FOC, does not present very accurate reaction to torque commands which could

lead to instability [22].

The direct stator current (i_{ds}) is used for the rotor flux while the quadrature current (i_{qs}) is the responsible for controlling the torque. By holding the flux constant, the torque is directly proportional to the stator q-axis current as it will be shown later on in this chapter. Thus, the flux linkage vector will generate the reference for the d-axis current controller and, on the other hand, the applied torque will produce the reference for the q-axis current controller [4][7][21].

3.2.1 Indirect field oriented control

Besides the type of vector control regarding the flux orientation, there are also two different ways of implementing FOC for determining the rotor flux position. Depending on the possibility to measure the rotor shaft position of the IM or not, FOC can also be divided into direct and indirect FOC.

The direct technique, also known as sensor-less technique, calculates the rotor flux position without using an encoder or tachometer in the IM's shaft. The rotor flux position is either measured by using flux sensors in the IM or it is estimated from the voltage equations [7].

On the other hand, indirect FOC utilizes the rotor speed/position measurement to determine the rotor flux position. As the rotor field rotates at synchronous speed in a dq reference frame and, the mechanical rotor speed is measurable, it is possible to obtain the rotor flux speed if the slip speed can be estimated. Using the mechanical rotor speed and the slip speed, the synchronous speed is obtained and, by integrating it, the rotor flux position is determined [7]. Indirect FOC is the selected technique because the induction machine available in the laboratory already has an incremental encoder mounted in the IM shaft.

3.2.2 Principle of rotor flux oriented control

The basic idea of rotor flux FOC is to use an arbitrary dq reference frame where the rotor flux is constantly aligned with the real part of the stator current i_{ds}. The rotor flux orientation is depicted in the phasor diagram in Figure 3.13. This orientation allows decoupling of the IM torque and flux control variables making it possible to operate the induction machine as a separately excited DC motor [7][4]. This means that by transforming the three phase stator currents into two current components, in a synchronously rotating dq frame, i_{qs} and i_{ds} are analogous to the field and armature current of a DC machine, respectively [7].

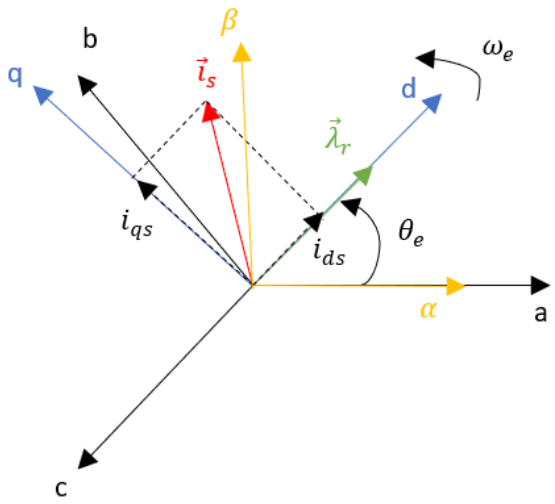


Figure 3.13: Phasor diagram showing rotor flux orientation.

The general torque equation for an induction machine in the dq reference frame is expressed in Equation 3.20 as [7]:

$$T_e = \frac{3}{2} \cdot p \cdot L_m \cdot (i_{qs} \cdot i_{dr} - i_{ds} \cdot i_{qr}) \tag{3.20}$$

This equation can be written in different forms depending on the type of control that will be implemented. In this project, as rotor flux oriented control is the selected technique, the previous equation is defined in a vector form as in Equation 3.21. The torque is a function of the rotor flux linkage $\vec{\lambda_{dqr}}$ and the stator currents $\vec{i_{dqs}}$:

$$T_e = \frac{3}{2} \cdot p \cdot \frac{L_m}{L_r} \cdot \mathrm{Im}(\vec{i_{dqs}} \cdot \vec{\lambda_{dqr}}^{\,*}) \tag{3.21}$$

where,

$$\vec{i_{dqs}} = i_{ds} + j \cdot i_{qs} \qquad \vec{\lambda_{dqr}}^{\,*} = \lambda_{dr} - j \cdot \lambda_{qr} \tag{3.22}$$

Substituting Equation 3.22 in Equation 3.21, it is obtained that the electromagnetic torque depends proportionally on the stator q-axis current because λqr is zero due to the rotor flux orientation with the d-axis.

$$T_e = \frac{3}{2} \cdot p \cdot \frac{L_m}{L_r} \cdot \lambda_{dr} \cdot i_{qs} \tag{3.23}$$

Therefore, it is possible to conclude that the quadrature stator component is responsible of producing torque. The direct stator current does not contribute in the torque production but in the flux, as it will be shown later in this section. This is the basic operation of rotor flux oriented control which is illustrated in the block diagram of Figure 3.14.

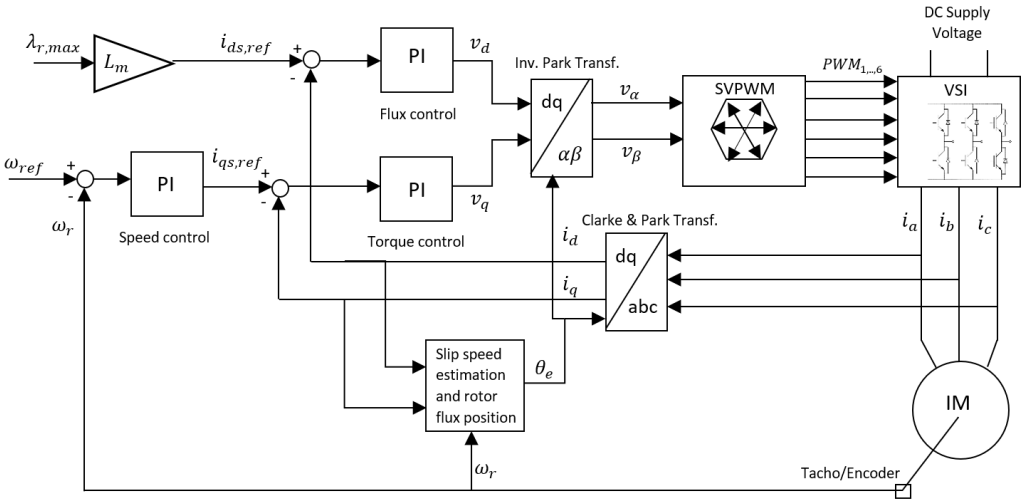


Figure 3.14: Block diagram of indirect rotor flux field oriented control [7].

From the block diagram it is observed that two PI current controllers are required for independently controlling the flux and the electromagnetic torque. The output of these controllers are the two voltage components in the dq reference frame. These dq voltage components are further transformed to the $\alpha\beta$ reference frame in order to implement the SVPWM. The graphical representation of these reference frames are shown in Figure 3.13. Besides the two current PI controllers, it is also possible to implement an outer speed controller that will generate the torque reference for the q-axis current controller [4][7].

In this project, the reference for the q-axis current $i_{qs,ref}$ can be generated in two ways which will be named torque control and speed control. Torque control will generate the required current reference directly from a torque profile reference using Equation 3.23. Speed control, also known as cruise control, will generate $i_{qs,ref}$ from an outer PI control loop. For both types of control, it is necessary to determine the rotor flux position.

Determination of the rotor flux position

Field oriented control requires performing reference frame transformations because the controller is implemented in an arbitrary dq reference frame. The rotor flux position θ_e has to be determined in order to perform the transformations required for field orientation. This position will be obtained by integrating the electrical stator speed ω_e which is calculated as the summation of the slip speed and the mechanical rotor speed. The induction machine voltage and flux linkage equations represented in a dq reference frame are [7]:

$$u_{qr} = R_r \cdot i_{qr} + \frac{d\lambda_{qr}}{dt} + (\omega_e - \omega_r) \cdot \lambda_{dr} \tag{3.24}$$

$$u_{dr} = R_r \cdot i_{dr} + \frac{d\lambda_{dr}}{dt} - (\omega_e - \omega_r) \cdot \lambda_{qr} \tag{3.25}$$

$$\lambda_{qr} = L_{lr} \cdot i_{qr} + L_m \cdot (i_{qs} + i_{qr}) \tag{3.26}$$

$$\lambda_{dr} = L_{lr} \cdot i_{dr} + L_m \cdot (i_{ds} + i_{dr}) \tag{3.27}$$

As the rotor windings are short-circuited ($u_{qr} = u_{dr} = 0$) and in rotor flux oriented control the d-axis is aligned with the rotor flux linkage ($\lambda_{qr} = 0$), the previous equations are reduced to:

$$0 = R_r \cdot i_{qr} + \frac{d\lambda_{qr}}{dt} + \omega_{sl} \cdot \lambda_{dr} \tag{3.28}$$

$$0 = R_r \cdot i_{dr} + \frac{d\lambda_{dr}}{dt} \tag{3.29}$$

$$0 = L_r \cdot i_{qr} + L_m \cdot i_{qs} \tag{3.30}$$

$$\lambda_{dr} = L_r \cdot i_{dr} + L_m \cdot i_{ds} \tag{3.31}$$

Where $\omega_{sl} = \omega_e - \omega_r$ is the slip speed and $L_r = L_{lr} + L_m$ is the sum of the rotor leakage inductance and the magnetizing inductance.

The rotor quadrature current i_{qr} is obtained from Equation 3.30 as a function of the stator quadrature current i_{qs} (Equation 3.32) in order to eliminate the rotor currents as they cannot be measured. Substituting i_{qr} in Equation 3.28, the slip speed is obtained as in Equation 3.33, where $\tau_r = \frac{L_r}{R_r}$ is the rotor time constant. The derivative term of Equation 3.28 is 0 because the rotor flux is kept constant.

$$i_{qr} = -\frac{L_m}{L_r} \cdot i_{qs} \tag{3.32}$$

$$\omega_{sl} = \frac{L_m}{\tau_r} \cdot \frac{i_{qs}}{\lambda_{dr}} \tag{3.33}$$

The rotor flux linkage, $\mid \vec{\lambda_{dqr}} \mid = \lambda_{dr} = \lambda_r$, is usually held constant at its rated value when the IM is operating below the base speed [4]. This approach is done to be able to vary the electromechanical torque from zero to maximum value without having to adjust λ_r. Hence, as the rotor flux linkage is kept constant, the torque will only depend on i_{qs} (Equation 3.23).

From the d-axis rotor voltage equation (Equation 3.29) it is obtained that the rotor d-axis current is zero because the rotor flux λ_r remains constant. Therefore, by including this in Equation 3.31, the relationship between the flux linkage and the d-axis stator current is defined in Equation 3.35.

$$i_{dr} = -\frac{1}{R_r} \cdot \frac{d\lambda_r}{dt} \rightarrow i_{dr} = 0 \tag{3.34}$$

$$\lambda_r = L_m \cdot i_{ds} \tag{3.35}$$

Thus, the estimation of the slip speed will depend upon the quadrature and direct stator currents as stated in Equation 3.36.

$$\omega_{sl} = \frac{1}{\tau_r} \cdot \frac{i_{qs,ref}}{i_{ds,ref}} \tag{3.36}$$

A common simplification approach is to use the reference stator currents for the slip speed determination instead of the measured currents. This is because these currents are usually less noisy than the measured ones and the results achieved in steady state are almost the same. The reference quadrature current $i_{qs,ref}$ is obtained from the torque command (Equation 3.23) and the reference direct current $i_{ds,ref}$ from the rotor flux linkage command (Equation 3.35).

Once the slip speed has been estimated, the electrical stator speed ω_e can be calculated as in Equation 3.37. The mechanical speed ω_r is obtained from the position sensor that is attached to the motor's shaft, as indirect FOC is implemented.

$$\omega_e = \omega_{sl} + \omega_r \tag{3.37}$$

The rotor flux position is found by integrating the electrical speed:

$$\theta_e = \int (\omega_{sl} + \omega_r) \tag{3.38}$$

Calculation of rated rotor flux linkage

The reference d-axis current $i_{ds,ref}$ for the PI flux controller is determined from the desired rotor flux level. Therefore, the next step is to calculate the rated rotor flux linkage $\lambda_{r,max}$. The induction machine voltage and flux linkage equations in vector form, in a dq reference frame, are defined as in Equations 3.39 and 3.40, respectively.

$$\vec{u_{dqs}} = R_s \cdot \vec{i_{dqs}} + j \cdot \omega_e \cdot \vec{\lambda_{dqs}} \qquad \vec{u_{dqr}} = 0 = R_r \cdot \vec{i_{dqr}} + j \cdot (\omega_e - \omega_r) \cdot \vec{\lambda_{dqr}} \tag{3.39}$$

$$\vec{\lambda_{dqs}} = L_s \cdot \vec{i_{dqs}} + L_m \cdot (\vec{i_{dqr}}) \qquad \vec{\lambda_{dqr}} = L_r \cdot \vec{i_{dqr}} + L_m \cdot (\vec{i_{dqs}}) \cdot \vec{\lambda_{dqr}} \tag{3.40}$$

The stator flux linkage $\vec{\lambda_{dqs}}$ is obtained from the stator voltage equation as the induction machine's rated phase voltage and current are known parameters. Induction motors always run at lagging power factor (PF) [7], which means that in a phasor representation the voltage is at the origin (0°) and the current is lagging by the angle defined by the nominal $PF = \cos\alpha = 0.76$ (Table 2.1). Therefore, the rated phase voltage and current are calculated as in Equations 3.41 and 3.42.

$$\vec{u_{dqs}} = u_{pk,rated} \cdot e^{j0^\circ} = \frac{24V \cdot \sqrt{2}}{\sqrt{3}} = 19.6V\angle 0^\circ \tag{3.41}$$

$$\vec{i_{dqs}} = i_{pk,rated} \cdot e^{-j\alpha} = 267.28A\angle -40.53^\circ \tag{3.42}$$

The nominal frequency of the IM is $f = 58Hz$, therefore, the rated electrical synchronous speed is $\omega_e = 2\pi f = 364.42$ rad/s. From this and the nominal voltage and current calculated previously, the rated stator flux linkage is found to be:

$$\vec{\lambda_{dqs}} = \frac{1}{j\omega_e} \cdot (\vec{u_{dqs}} - R_s \cdot \vec{i_{dqs}}) = 0.0524\angle -88.68^\circ \tag{3.43}$$

On the other hand, from the rotor flux linkage equation, $\vec{i_{dqr}}$ is obtained and is included in the stator flux linkage equation. Performing math operations the rotor flux linkage results in Equation 3.44, where $\sigma = L_s - \frac{L_m^2}{Lr}$ is the resultant leakage constant.

$$\vec{\lambda_{dqr}} = \frac{L_r}{L_m} \cdot (\vec{\lambda_{dqs}} - \sigma \cdot L_s \cdot \vec{i_{dqs}}) = 0.05671\angle -88.68^\circ \rightarrow \lambda_{r,max} = 0.05671 \tag{3.44}$$

From the rated rotor flux linkage the direct stator current reference is obtained from Equation 3.35:

$$i_{ds,ref} = \frac{\lambda_{r,max}}{L_m} = 149.22A \tag{3.45}$$

3.2.3 Implementation of indirect rotor flux oriented FOC

This section presents the procedure followed for implementing indirect rotor flux FOC. As it was mentioned in subsection 3.2.2, two types of control will be implemented in this project: torque control and cruise control. In the case of torque control only the two inner control loops are required while for cruise control a cascaded controller needs to be implemented by adding an outer control loop for the speed. In any case, the controllers will be designed in continuous domain and then they will be discretized in order to implement them using a DSP.

Design of PI controllers in continuous domain

In total three PI controllers will be designed in this section. It is important to consider the bandwidth of these controllers because when implementing cruise control a cascaded control architecture is necessary. Therefore, the inner current loops must be significantly faster (at least $3-5$ times) than the outer speed loop [23]. The reason for this is that the outer loop provides the set point for the inner loop and, making the inner loop faster, allows to compensate the inner loop disturbances before they affect the outer loop [23]. Therefore, the PI current controllers will be designed first and based on the selected bandwidth for the inner control loops the speed controller will be designed.

PI current controllers First the design of the inner PI controllers for the d- and q-axis currents will be carried out. It is necessary to obtain the transfer function of the plant to be controlled, which in this case is the induction machine. The plant's transfer function, with the stator current as output and voltage as input, is obtained from the IM equations. The stator voltage equations in Laplace form are:

$$V_{ds}(s) = R_s \cdot I_{ds}(s) + s\lambda_{ds}(s) - \omega_e \cdot \lambda_{qs}(s) \tag{3.46}$$

$$V_{qs}(s) = R_s \cdot I_{qs}(s) + s\lambda_{qs}(s) + \omega_e \cdot \lambda_{ds}(s) \tag{3.47}$$

Substituting the stator flux linkage equations in Equation 3.46 and 3.47 yields to:

$$V_{ds}(s) = R_s \cdot I_{ds}(s) + s(L_s \cdot I_{ds}(s) + L_m \cdot I_{dr}(s)) - \underbrace{\omega_e \cdot (L_s \cdot I_{qs}(s) + L_m \cdot I_{qr}(s))}_{\text{decoupling term}} \quad (3.48)$$

$$V_{qs}(s) = R_s \cdot I_{qs}(s) + s(L_s \cdot I_{qs}(s) + L_m \cdot I_{qr}(s)) + \underbrace{\omega_e \cdot (L_s \cdot I_{ds}(s) + L_m \cdot I_{dr}(s))}_{\text{decoupling term}} \quad (3.49)$$

It is observed from the previous equations that there exists coupling between the d and q axis. Therefore, it is necessary to decouple them by adding or subtracting these terms in the current loops in order to have independent control of the torque and flux components and neglect these terms in the transfer function of the plant. These decoupling terms will be seen as disturbances in the system. Moreover, in order to simplify the transfer function of the plant, the rotor currents are neglected resulting in the transfer function of the IM.

$$G_p(s) = \frac{I_{ds}(s)}{V_{ds}(s)} = \frac{I_{qs}(s)}{V_{qs}(s)} = \frac{1}{R_s + L_s \cdot s} = \frac{\frac{1}{L_s}}{s + \frac{R_s}{L_s}} \quad (3.50)$$

The transfer function is the same for the d- and q-axis which means that the current loops for both will be exactly the same. Thus, the PI current controller will be design for the d-axis and the controller's parameters obtained will be also used for the quadrature current. The general form of a PI controller in the continuous domain is shown in Equation 3.51 [23].

Define design constraints. Settling time, overshoot, rise time beforehand

$$G_c(s) = K_P + \frac{K_I}{s} \quad (3.51)$$

When designing the current controllers it is necessary to include a time delay in order to obtain simulation results that can be compared with the results obtained in the laboratory. The delay appears in the system because of the digital implementation in a microcontroller. The time for sampling the measured signals in the Analog to Digital Converter (ADC) and the time it takes to perform the control algorithm have to be taken into account. The time delay is reduced as much as possible while programming the microcontroller achieving a total time delay of half a switching period ($T_d = 0.5T_s$). The implementation in Simulink of the PI controller for i_{ds}, including the transfer function of the plant and the time delay, can be seen in Figure 3.15. Where the transfer function for the time delay is:

$$G_d(s) = \frac{1}{T_d \cdot s + 1} \quad (3.52)$$

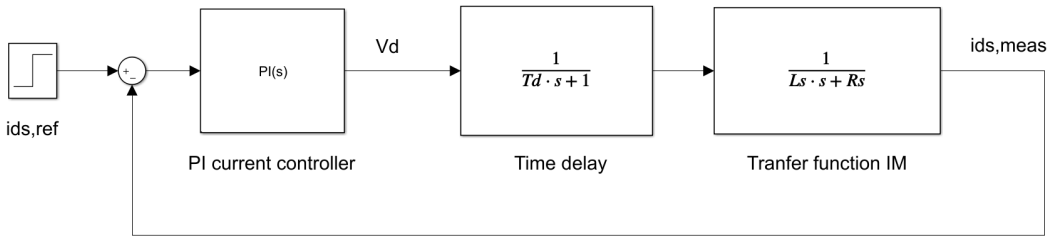


Figure 3.15: PI current controller in continuous domain for the direct stator current i_{ds}.

A common control approach is to select the zero of the controller in a way that it cancels out the pole of the original transfer function of the plant [23]. This zero-pole cancellation technique is done in order to obtain a closed-loop system with no zeroes and only one real pole to ensure continuous stable control. From Equation 3.50 it can be seen that the pole of the plant is located in $s = -\frac{R_s}{L_s}$ and the zero of the PI controller in $s = -\frac{K_I}{K_P}$ (Equation 3.51). Therefore, by equating these terms the integral gain of the PI controller is obtained as:

$$K_I = \frac{R_s}{L_s} \cdot K_P \tag{3.53}$$

The next step is to obtain the value of the proportional gain K_P. The open loop transfer function of the system is reduced to Equation 3.54 after cancelling the plant's pole.

$$G_{OL}(s) = G_c(s) \cdot G_d(s) \cdot G_p(s) = \frac{\cancel{(s + \frac{K_I}{K_P})} \cdot \frac{1}{L_s}}{\frac{s}{K_P} \cdot (T_d \cdot s + 1) \cdot \cancel{(s + \frac{R_s}{L_s})}} \tag{3.54}$$

The closed-loop transfer function of the system is used to find the value of K_P. It is observed that the closed-loop is a second order system. Hence, equating the closed loop transfer function with the general form of a second order transfer function (Equation 3.55), the value of K_P can be found.

$$G_{CL}(s) = \frac{G_{OL}(s)}{1 + G_{OL}(s)} = \frac{\frac{K_P}{T_d \cdot L_s}}{s^2 + \frac{1}{T_d} \cdot s + \frac{K_P}{T_d \cdot L_s}} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n \cdot s + \omega_n^2} \tag{3.55}$$

$$\omega_n = \frac{1}{2 \cdot \zeta \cdot T_d} \qquad K_P = L_s \cdot T_d \cdot \omega_n^2 \tag{3.56}$$

According to [24], the inner current PI controllers usually limit the bandwidth to approximately 10% of the switching frequency. The switching frequency is $f_s = 10KHz$, therefore, the requirement for the current controllers is that the bandwidth is $BW_c = 1KHz$. However, when discretizing the controller later on it was found that this bandwidth was very agressive and the currents showed high ripples. Therefore, the bandwidth requirement was reduced to half of the previous value ($BW_c = 500Hz$).

The proportional gain K_P depends upon the natural frequency ω_n of the system as the stator inductance and the time delay are known parameters. For a

second order system, the bandwidth is related to the natural frequency as shown in Equation 3.57 .

ref

$$\omega_{BW} = \omega_n \cdot \sqrt{(1 - 2\zeta^2) + \sqrt{4\zeta^4 - 4\zeta^2 + 2}} \tag{3.57}$$

Substituting the expression of ω_n obtained in Equation 3.56 in Equation **??**, and setting the bandwidth requirement ($\omega_{BW} = 2 \cdot \pi 500Hz = 3141.6[rad/s]$) it is obtained that the damping ratio is $\zeta = 1.35$. This value corresponds to a natural frequency of $\omega_n = 7382[rad/s]$. Thus, the values for the PI current controllers parameters are:

$$K_P = K_{P,d} = K_{P,q} = 1.12 \qquad K_I = K_{I,d} = K_{I,q} = 6.81 \tag{3.58}$$

Figure 3.16 shows the closed loop step response for the d-axis current for a step input reference of $1A$. The response to the step input does not show overshoot as it is an overdamped system ($\zeta > 1$). The rise time is the time the system takes to rise from 10% to 90% of its final value and it is found to be 0.7 ms. On the other hand, the settling time is the time required to reach 98% of the final value and it is, in this case, 1.27 ms.

Show these parameters on the fig. 3.17 as well if possible. Lajos

The text in the bode diagram is pretty small, is it possible to make it bigger somehow? NHF

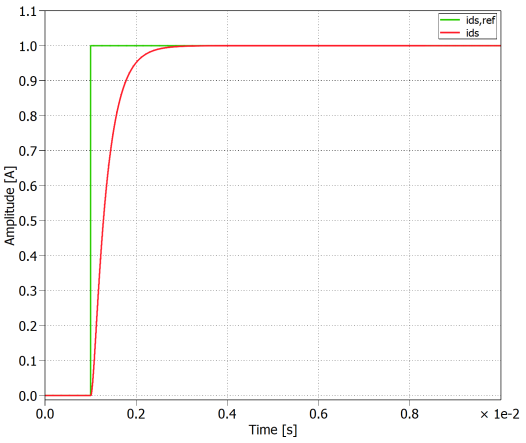


Figure 3.16: Closed loop step response for a direct reference value of $i_{ds,ref} = 1A$.

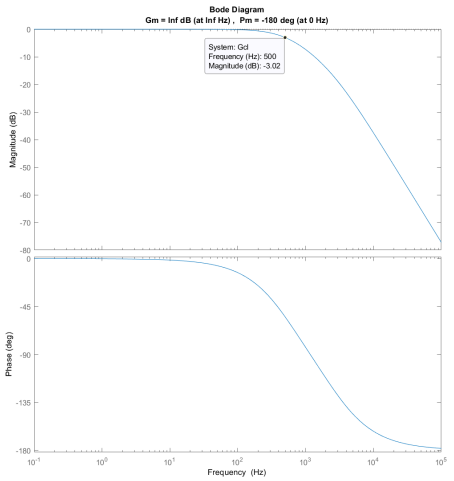


Figure 3.17: Bode diagram of the closed loop system including the PI current controller.

The bode diagrams of the closed loop system for the current controller is shown in Figure 3.17. From this plot, the corresponding bandwidth is read at the point where the magnitude plot is -3dB, concluding that the controllers fulfil the bandwidth requirement $BW_c = 500$ Hz. Moreover, it is observed that the PI current controller corresponds to a second order system as its phase varies from 0° to -180° because the system has two poles.

Is this plot based on a step function applied to the transfer function or the simulink model? I suggest using the simulink model as this is closer to the real world. Stef

Left image: Somehow include this 10-2 in the numbers on the axes as when first reading, it hardly observed it, and considered the time being 0.2-0.4-... s. Lajos

Right image: Blurry. Lajos

PI speed controller The outer speed loop is governed by the mechanical equation of the induction machine. The mechanical model of an induction machine is

defined based on the load torque T_L, the motor's moment of inertia J, the viscous friction coefficient B and the rotor's speed ω_r as it was explained in Section 2.1.

$$T_e - T_L = J \cdot \frac{d\omega_r}{dt} + B \cdot \omega_r \tag{3.59}$$

Laplace transforming Equation 3.59, the transfer function of the plant for the design of the speed controller is found in Equation 3.60. The load torque T_L is seen as a disturbance in the system.

Assuming this is quite a simplification as you then ignore the whole vehicle dynamics. The time constant of (4.60) will be very low, but if you include the vehicle mass, it will be much higher. Erik

$$G_{p,\omega} = \frac{\omega_r(s)}{T_e(s) - T_L(s)} = \frac{1}{J \cdot s + B} \tag{3.60}$$

The outer speed control loop will generate the reference electromagnetic torque $T_{e,ref}$ which provides the current reference for the quadrature current loop. As it was demonstrated at the beginning of this chapter, i_{qs} depends proportionally on T_e. Thus, as the rotor flux is kept constant at its rated value, the constant terms in Equation 3.23 are grouped in a single variable $k = \frac{3}{2} \cdot p \cdot \frac{L_m}{L_r} \cdot \lambda_{dr}$. This constant is used to obtain the current reference $i_{qs,ref}$ from the electromagnetic torque reference $T_{e,ref}$.

Furthermore, the close loop transfer function of the current controller can be reduced to a first order transfer function because the outer speed loop is much slower than the inner loop. This means that the time delay will not have a significant effect in the speed controller. The value of the time constant τ_c that defines the delay introduced by the current loop is obtained from the step response of Figure 3.16. It is found to be $\tau_c = 1.27$ms and this is the value used for the simplified transfer function for the inner current loop. Figure 3.18 shows the implementation of the speed controller in Simulink.

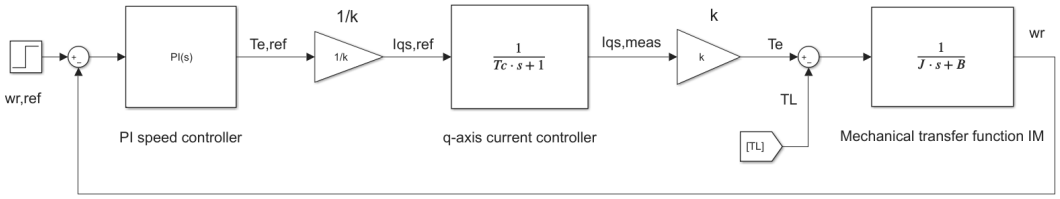


Figure 3.18: PI speed controller in continuous domain for generation of $i_{qs,ref}$.

Same as before try to make to picture more readable, NHF

hard to read: 1/k is here? Lajos

Following the same zero-pole cancellation technique, as done during the design of the current PI controllers, the zero is selected to cancel the induction machine's pole. This yields to the integral gain to be dependent of $K_{P,\omega}$.

$$K_{I,\omega} = \frac{B}{J} \cdot K_{P,\omega} \tag{3.61}$$

$$G_{OL,\omega}(s) = G_{c,\omega}(s) \cdot G_{p,\omega}(s) = \frac{\cancel{\left(s + \frac{K_{I,\omega}}{K_{P,\omega}}\right)} \cdot \frac{1}{J}}{\frac{s}{K_{P,\omega}} \cdot (1 + \tau_c \cdot s) \cdot \cancel{\left(s + \frac{B}{J}\right)}} \tag{3.62}$$

In cascaded controllers the inner loop has to be faster than the outer loop. Thus, it is decided that the speed controller must be 10 times slower than the current controllers. This means that the bandwidth of the speed controller is $BW_\omega = 50$ Hz. The closed loop transfer function is reduced to a second order system. Identifying terms with the general transfer function of such system, as it was done during the design of the current controllers. It is found that the proportional gain $K_{P,\omega}$ depends on the natural frequency which at the same time depends on the bandwidth of the system (Equation 3.57).

$$G_{CL,\omega}(s) = \frac{G_{OL,\omega}(s)}{1 + G_{OL,\omega}(s)} = \frac{\frac{K_{P,\omega}}{J \cdot \tau_c}}{s^2 + \frac{1}{\tau_c} \cdot s + \frac{K_{P,\omega}}{J \cdot \tau_c}} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n \cdot s + \omega_n^2} \tag{3.63}$$

$$K_{P,\omega} = J \cdot \tau_c \cdot \omega_n^2 \tag{3.64}$$

Following the same procedure as before, it is found that the damping factor is in this case $\zeta = 0.914$ and the natural frequency $\omega_n = 430.5 rad/s$. Therefore, the speed controller's parameters are found to be:

$$K_{P,\omega} = 3.55 \qquad K_{I,\omega} = 0.355 \tag{3.65}$$

The value of Ki was found assuming a ration B/J = 0.1. However, this value needs to be found by experiment or if we dont get to do it it needs to be tuned when including it in the complete Simulink model.Stef

Figure 3.19 shows the closed loop response of the system for a speed input step of $\omega_{r,ref} = 1 rad/s$. It is observed that both, the rise time and the settling time are ten times higher than the obtained for the current controllers ($t_{r,\omega} = 7$ms and $t_{s,\omega} = 12$ms), as this was the requirement during the design of the speed controller. The bode diagram of the closed loop systems for the speed controller verifies the desired bandwidth of $BW_\omega = 50$Hz.

reconfirm these values. FA

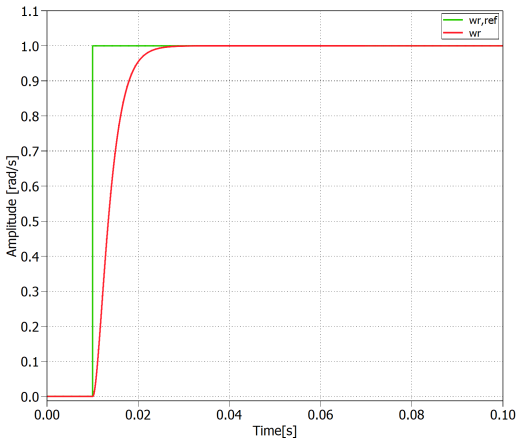


Figure 3.19: Closed loop response for a unit step rotor speed reference.

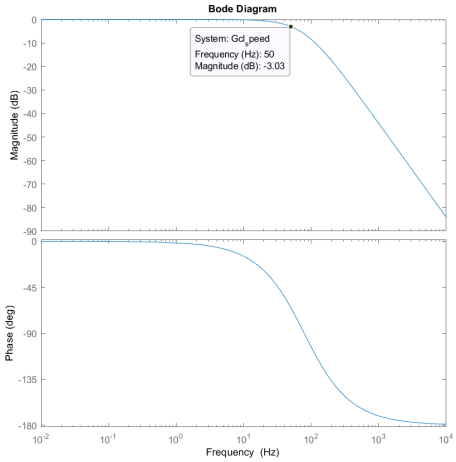


Figure 3.20: Bode diagram of the closed loop system including the PI current controller.

The go-kart will never be so fast. Only without a mechanical load you have such high acceleration. Erik

Discretization of the controller

After achieving the desired controller performance in continuous domain. It is important to analyze the controller in discrete domain because in physical world the motor control calculations and implementation will be carried out in discrete domain using the DSP. In this project, the delay introduced by DSP is considered in control design strategy from the beginning as shown in section 3.2.3. This helps to reduce drastic variation in controller performance when discretized and requires minimum re-tunining efforts.

The discretization of controller or plant transfer function can be done by direct dicrete equivalents or by emulation, where the continuous time transfer function is approximated in discrete time. Furthermore there are different techniques for discretization such Zero Order Hold (ZOH), Tustin, matched pole-zero method. The plant transfer function will be discretized using ZOH as this is how the signals will be provided to the real system. And so the controller in order to closely approximate what the plant is receiving. At the sampling frequnecy of $10kHz$ discrete response of both current and speed controller is shown below in Figure 3.21 [25].

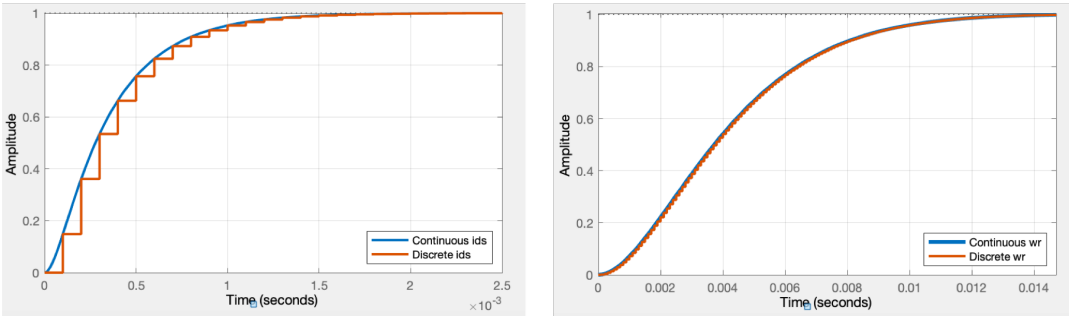


Figure 3.21: Unit step response in continuos and discrete domain.*Left:* Stator $d-$axis Current *Right:* Rotor Speed. Sampling frequency of 10kHz

Saturation and Anti-windup

The controller design discussed so far has been able to provide the desired performance. In other words, the controller be able to follow the desired torque or speed reference by providing the right reference to direct or quadrature axis current irrespective of the amplitude. But in real world the system follow whatever reference is thrown at it. Hence there needs to be a saturation block implemented on the PI blocks. The limit of the saturation blocks will be rated $q-$axis current or dq axis voltage for the speed or current PI blocks respectively.

But this leads to significant error term of PI blocks as the reference is higher, but the measured value from plant is based on saturated output of PI controller. On the one hand this saturation limit definitely protects the hardware from going beyond the rated values. But, large error value leads to extremely large action of integrator. This leads to a phenomenon known as integral windup and can lead to oscillations if not compensated. A common way of solving this integral windup is by bypassing the integral action when saturation is achieved. But this is not an ideal solution hence in a different approach the integral action is reduced by a

factor K_a, where K_a is kept at $1/K_p$ [26]. This approach is also referred as back-calculation method and is shown below in Figure 3.22 when used for the speed controller.

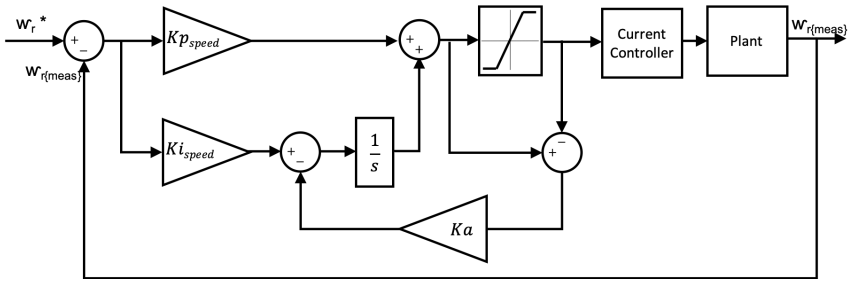


Figure 3.22: Back-calculation anti-windup technique on integrator of speed control loop.

3.3 Regenerative Braking

In a conventional internal combustion (IC) engine, the only way to stop a vehicle in motion is by implementing mechanical brakes and dissipating the kinetic energy into heat. One of the biggest reasons why electric vehicle is seen as a disrupter to internal combustion engine is their ability to convert the kinetic energy of the vehicle into electrical energy which is stored back into the batteries when brake is applied. This phenomenon is popularly known in the industry as regenerative braking as the induction motor (in case of our project) can be operated as generator during braking operation and recover the kinetic energy.

3.3.1 Types of regenerative braking

In spite of clear advantage of regenerative braking over conventional mechanical braking system, not all the braking requirement can be provided solely from regenerative method. Regenerative braking may not be used under conditions such as high state of charge (SOC) or high temperature on battery. Furthermore, the required braking torque (deceleration) if implemented completely from regenerative braking might generate current amplitudes that will exceed the inverter or battery rating values. Furthermore, motor itself may not be able to provide the required braking force (deceleration).

From the above paragraph we conclude that there may be various constraints that will limit the regenerative braking capability and we will have to resort to frictional braking for the remaining brake requirement. There are two main control methodologies in academia for splitting the required braking torque between regenerative and frictional braking:

- Parallel Regenerative Braking.
- Series Regenerative Braking.

Figure 3.23 left side represents parallel regenerative braking strategy where in both frictional as well as regenerative braking work in tandem sharing the required braking torque from $0Nm$. Ratio with which the two techniques share the braking

load ($T_{friction}/T_{regen}$) depends on multiple parameters. As it can be seen that the ratio remains constant to the point where regenerative braking max out. Beyond this point, rest of braking torque is provided completely by frictional brakes.

In contrast to parallel braking strategy, in series regenerative braking strategy (3.23 *Right*) from $0Nm$ to the point where regenerative braking torque maxes out entire braking torque is provided by regeneration. Beyond this point frictional brake activates and provides the remaining required braking torque.

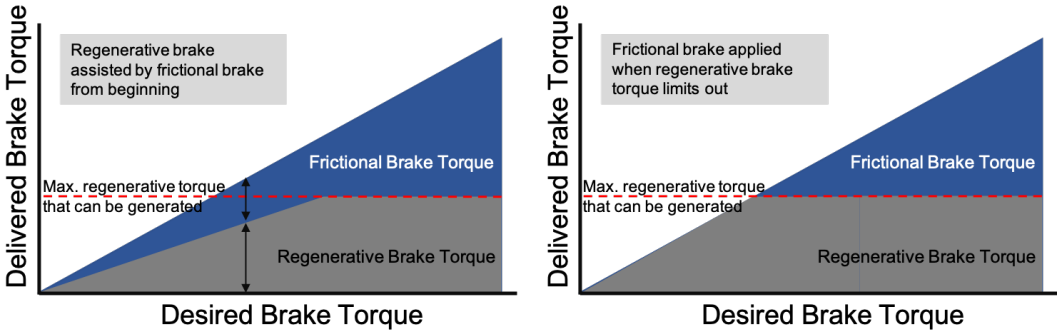


Figure 3.23: *Left:* Parallel and *Right:* Series Regenerative braking control

Even tho you made it yourself, it would be better to just write down the reference from where you took the info/knowledge from. Software team

In commercial applications it is preferable to employ parallel braking strategy or limit regenerative braking to be activated only after certain minimum threshold speed. Because at low speed the current generated by motor (generator mode) is not significant to ensure desirable battery recharge efficiency. Therefore at this low speed, frictional brake is employed to limit the "micro-cycling" process of battery charge/discharge cycles [27] . In this project our objective is to analyse and understand the regenerative braking capability of an induction motor in context with electric vehicle. We will employ series braking control to ensure maximum possible usage of regenerative braking. Hence in the following subsections, electrical braking capability based on the mechanical and electrical parameters will be discussed.

I don't understand this argument. Erik

I think that this statement doesn't consider the actual gokart, as we dont have electronic control of the mechanical braking, and the user will brake whenever she wants to. NM

3.3.2 Regenerative Braking Capability

Since the induction motor has a limited torque producing capability, first it is important to estimate the maximum deceleration that the vehicle can produce depending solely on the regenerative braking. Referring to Equation 2.21 in Section 2.5 and re-writing it below, where traction force (F_t) changes to brake force (F_b)

$$M_{kart} \cdot \ddot{x} = F_t - (F_{rr} + F_{ad} + F_g \cdot \sin\alpha) \tag{3.66}$$

for a flat surface $\alpha = 0$, which leads $F_g \cdot \sin\alpha = 0$ and maximum velocity of car ($\dot{x}$) being $14.55m/s$ we can calculate that the other two forces F_{rr}, F_{ad} will be $34.83N$ and $3.56N$ respectively. Now for the braking force on the wheel (F_t) braking torque that the motor can produce needs to considered. In this case, considering the rated torque of induction motor (T_n) of $30.04Nm$ to be the applicable braking torque as well, force on wheel can be calculated as below.

$$T_n\Big|_{@\text{ motor shaft}} = 30.04Nm \quad \Rightarrow \quad T_{brake}\Big|_{@\text{ axle}} = -T_n \cdot G_r$$

$$\text{Brake force}(F_b)\Big|_{@\text{ wheel}} = \frac{T_{brake}}{R_{wheel}} = \frac{-49.86Nm}{0.1375m} = -362.66N$$

plugging value of brake force obtained above, F_{rr}, and F_{ad} in equation 3.66 provides a deceleration of

$$a = -1.72m/s^2 \tag{3.67}$$

considering if this deceleration is available from top speed of 14.55 m/s to standstill, it will take approx. $8.5sec$ for the vehicle to stop. Or in other words, with this deceleration the Go-Kart will need $62m$ of space before it comes to rest.

Now the question arises how this braking torque will be implemented to the induction motor. As explained in detail in section 3.2, field oriented control (indirect rotor flux oriented) is the adopted control methodology for this project. This control provides control on the rotor flux through $d-$axis component and electromagnetic torque which in our case will be brake torque through $q-$axis component. Hence, when the driver pushes brake paddle the required deceleration(depending on the brake paddle position) will be translated to required braking torque demanded from motor which will set reference for $q-$axis current.

At this stage it is important to state that apart from the maximum deceleration produceable by the motor the current running back to the batteries also must be considered. When the batteries are at its 100% SoC level, they will not be able to accept the huge inrush current coming from the motor. Therefore, the DC link current (i_{DC}) has to be calculated.

The used constraint is that the braking torque is the rated torque. The motor can deliver the rated torque, at rated speed, forever. As you know, power is torque times angular speed, then if we set the actual constraint to 5.3 kW, we will be able to increase the braking torque when the speed is lowered. With the current constraint it sounds that we are working with a very low motor power.NM

We dont have a sensor in the brake. NM

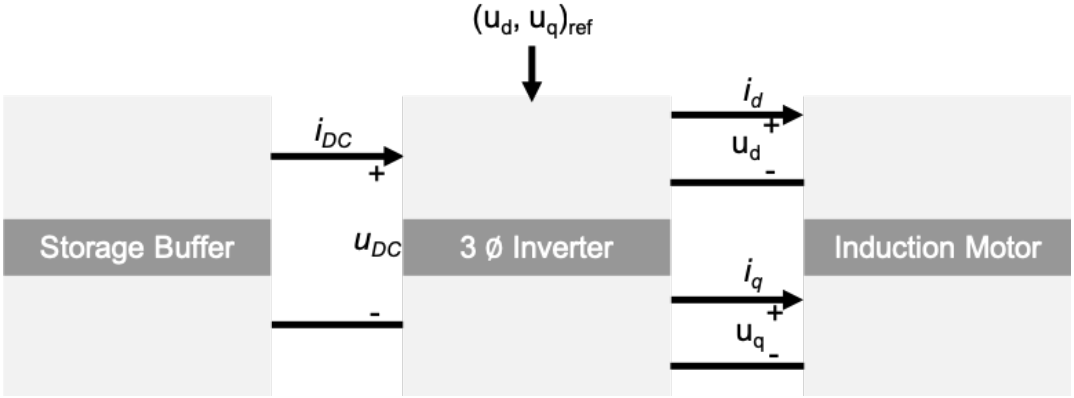


Figure 3.24: Abstract block diagram of the DC source, DC-AC inverter, and induction motor processing power in dq frame of reference

Figure 3.24 above shows an abstract block diagram of the system with induction motor processing power in dq frame of reference [28]. The electric machine input power can be written as,

$$P_e = u_a i_a + u_b i_b + u_c i_c = u_d i_d + u_q i_q \tag{3.68}$$

I don't really understand this image... AT. It shows power transferred to motor in dq frame instead of abc as they will be same. and exploiting that knowledge to calculate current flow. FA. Maybe its no so clear in the sense that the power is flowing to the motor in the regenerative braking chapter. NM

Assuming that the 3ϕ inverter is a lossless component then the power balance equation will be,

$$P_{in} = u_{DC} i_{DC} = u_d i_d + u_q i_q$$

$$\Rightarrow i_{DC} = \frac{u_d}{u_{DC}} i_d + \frac{u_q}{u_{DC}} i_q \tag{3.69}$$

Based on the information collected so far a simple logical flow diagram can be implemented which will decide whether required brake torque needs to be implemented from frictional brake only or a combination of regenerative and frictional braking as shown in Figure 3.25 below.

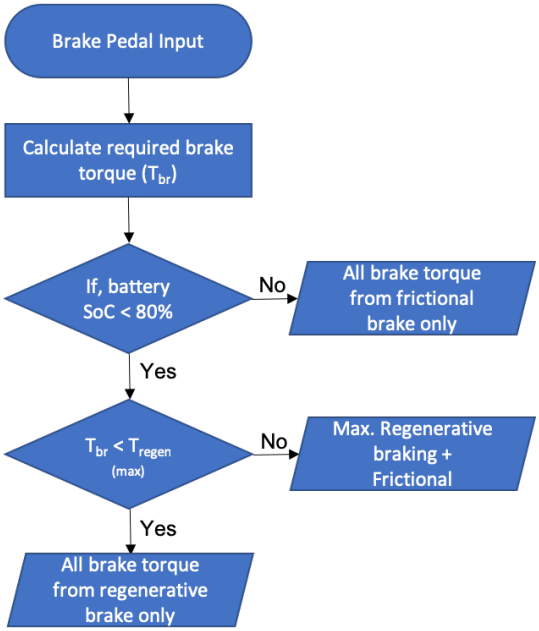


Figure 3.25: Flow chart representation of serial regenerative braking implementation

We dont have a brake pedal sensor.NM

4 Firmware design, implementation and testing

In order to control the inverter, it is necessary to develop a control algorithm. Although the goal is driving this motor by reading analog currents and generating the corresponding PWM signals, a relatively large framework must be developed to allow that task. For example, the user interaction with the system must be considered, the proper reference source must be used or safety techniques must be implemented to ensure that the system is behaving properly and react if it is not.

The followed design procedure is usually referred to as software Structured Design. This software development procedure starts with the software design, where the system's features are described. After the features are determined, the necessary modules and how is the data flowing among the modules can be determined, this phase is called High Level Design. Once every module's duty is determined, a detailed description of how is that task going to be developed, its functions, interfaces and main variables is performed. This stage is called Low Level Design. After the initial design stage, the efforts were put into code development, in this stage the low level design was used as an input and the code was generated and tested to fulfil the design. After every software component was developed, the testing procedure of the specific software component was written in a common test description document. Finally, the system was tested including the inverter and the motor. Significant attention has been put into the timing of the tasks and providing a framework of logging data. Also a Graphical User Interface (GUI) has been created to provide a friendly communication channel between the controller and the user.

The used microcontroller consists of a Texas Instruments TMS320F28069M from the Piccolo family [29]. The microcontroller is mounted in a development kit [18].

In this chapter the software architecture of the system is presented, showing the main blocks of the system and discussing execution triggers and time management of the tasks. After that, some software components which are of special interest are examined. Finally, the validation techniques used are reviewed.

you can also mention a bit about High Level design here, so the reader has some idea about next section before going into it. FA. I added a description of both and how are they related, what do you think about it now?NM

4.1 High level design

In the intro of the chapter it is mentioned that the initial stage of the software development is the low level design but here it starts with high level design. What does it mean low level and high level design? Stef I added a small explanation of both in the Intro, do you think it is enough??NM

The software architecture was designed before it went into development phase to avoid adding features in an *adhoc* manner. In this section, the software design is discussed.

The concept of addressing the system as a whole and deciding which modules will integrate the software is called software architecture. The software architecture divides all the functions to be performed into modules. After these modules and their specific tasks are described, it's necessary to design how are the modules going to communicate and when will the modules be run.

The advantages of following the architectural design before implementing code are the subsequent [30], [31]:

1. In this case, the task to perform (control a motor) is a big task and will require several lines of code, where more than two developers will contribute. In these conditions, if the developers do not follow a master plan, the developed modules might not be aligned. Then, additional work must be performed to align the modules, leading to extra development time. When every developer is following its own experience for developing, the result is usually known as *Spaghetti Code*. This code is usually unstructured, it is difficult to maintain and bugs are hard to find. If the developers can follow a design, the module will be coded just once and inputs, outputs and expected behaviour will be known from other modules before it's developed.

2. The development speed is increased.

3. Most design errors and bottlenecks are found in the design procedure instead of the development phase, then the implementation time is further decreased.

4. If the modules are first designed and discussed, the resulting implementation will have a common development philosophy and style, leading to easier changes by any team member, the *ownership* of software modules is removed.

5. The system's scalability is increased by using a modular approach. Additional features can be easily added and there's a procedure established for that goal. Sometimes when the system is finished, new system requirements arise. For fulfilling this new requirement, additional features must be developed or existing features must be redesigned, by using a modular approach the overhead of adding the new feature to the system is minimised.

6. Reliability, by understanding the system as a whole, the reliability of the system is increased.

The procedure followed for the software design can be seen in Figure 4.1. Every stage with the generated output and the approximate time spent relative to the whole procedure.

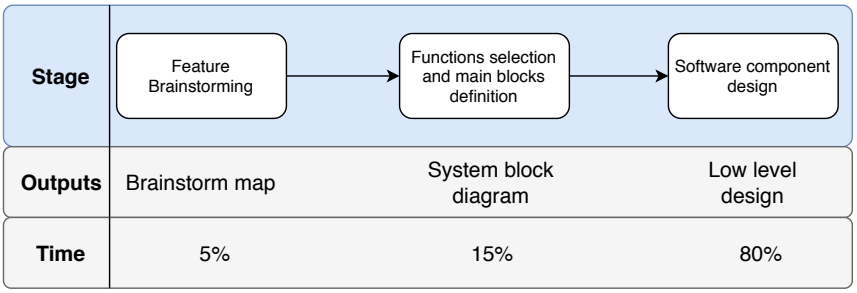


Figure 4.1: Software design procedure.

The designed system has 4 different blocks. Every block contains several software components. The blocks are System, Safety, Motor Control and User Interface, as seen in Figure 4.2. After this introduction a more detailed explanation of every block will be performed later.

- System consists of modules which are necessary for the system to work, like the system initialization or modules that are used for debugging, like the UART interface.
- Safety is the block in charge of monitoring that the system as a whole is behaving as expected. If that is not the case, proper reactions must be taken.
- Motor Control contains the modules which perform the actual controlling of the inverter by generating the duty cycles that will drive the motor as desired.
- User interface gathers the user input from the Interface PCB, the pedal or the Graphical User Interface and transforms that into a reference that will be fed into Motor Control.

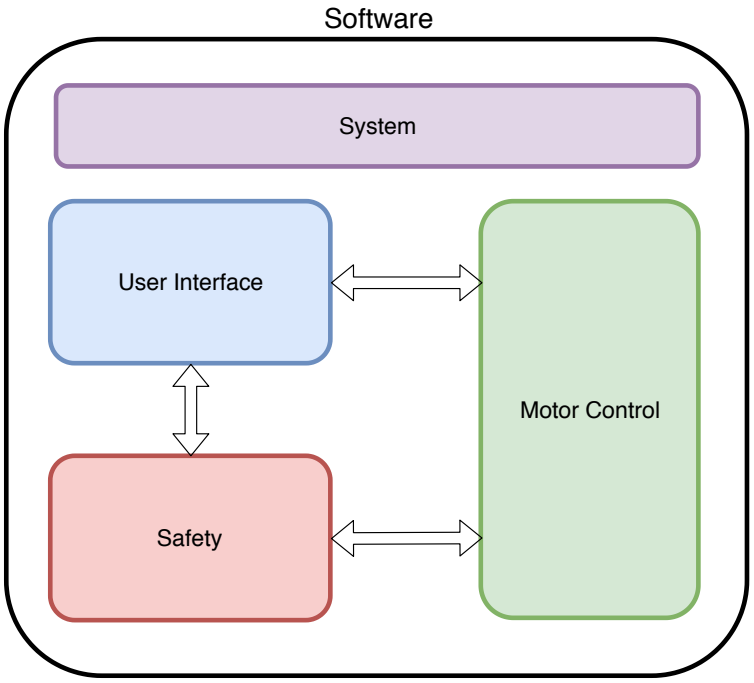


Figure 4.2: System main building blocks.

The Low Level Design output consists of a set of Unified Modeling Language (UML) diagrams. They allows creating simple representations of the software modules and the relationship between them. Furthermore it shows the main variables and functions to be implemented. UML diagrams have been created for all four blocks of the system and will be showed in their respective sections. An example of a diagram for the *System* block is shown in Figure 4.3. The variables are shown in the first part under the module name in the format: *name : type*. The functions are shown in the second part. The format here is: *function-name(in/out variable-name: type)*, where in/out defines if the variable is an input or a return argument.

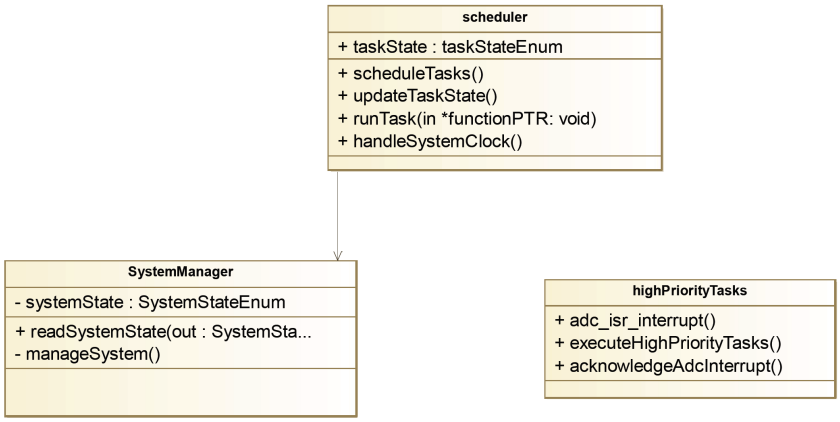


Figure 4.3: Example of a UML class diagram for the *System* block.

4.1.1 Timing considerations

It is very important that the motor controller has a precise execution frequency, as it performs calculations which use the time between execution as a variable, and if this time was wrong, miscalculations would arise. In addition, there are some tasks desired to run more often than others. A detailed design of the priority of the tasks and their execution frequency was performed. The result can be seen in Figure 4.4. The color coding follows the one Figure 4.2.

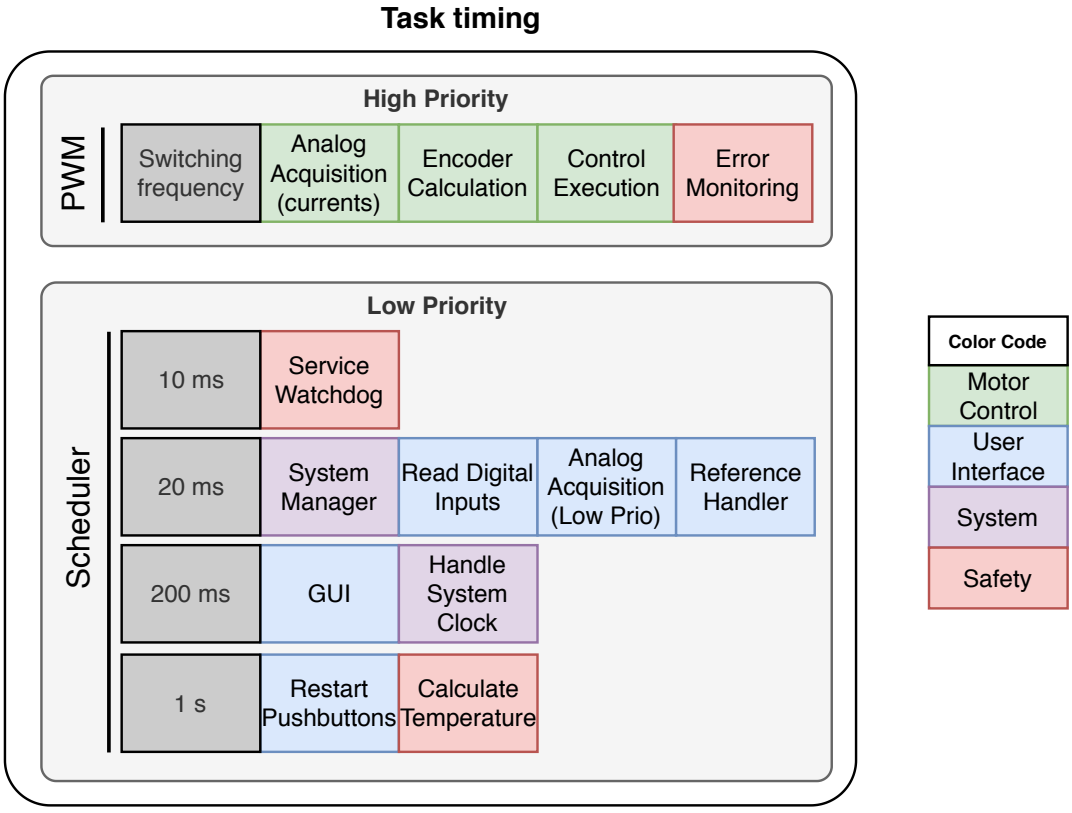


Figure 4.4: Timing design of tasks.

The implementation was performed by using the PWM as a trigger in the high priority tasks. The high priority tasks are those which must be executed every switching cycle and might contain calculations which include time as a variable. For the low priority tasks, the tasks are called by the Scheduler. The Scheduler implementation is described in section 4.6.1. In order to validate whether the tasks fit in the designed time span, the task's time consumption was measured. A table with all the results and further discussion can be found in section 4.6.1.

4.2 System

The main focus of *System* is the organization of the instructions that the DSP is working on at each moment. It works both in timing and in organization of the main state machine. For this reason, it features the following modules which will

be explained in detail in the following subsections:

- **System manager**, the main finite state machine of the program, handles the state at which the inverter is working, making sure the drivers are only active when intended (Subsection 4.2.1).
- **Scheduler**, organizes low priority tasks and calls the tasks at the correct periodicity, (Subsection 4.6.1).
- **High priority tasks' execution**, ensures precise timing of high priority tasks, these tasks need to be called at switching frequency (Subsection 4.2.3).

Figure 4.5 shows the UML class diagram for the *System* block. The *High priority tasks* module does not have any direct connection to the other modules, as it is triggered by the switching frequency.

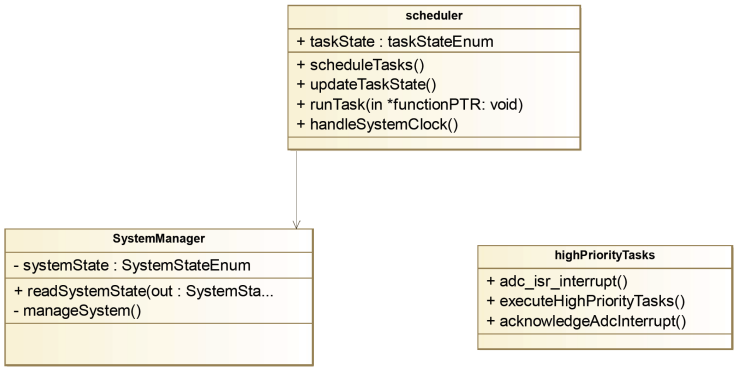


Figure 4.5: Class diagram for the modules in the *System* block.

4.2.1 System manager

The system manager is a module that has the goal to oversee the system. It is based in a finite state machine (FSM) that consists of four states; startup, standby, running and error. This is shown in figure 4.6.

After the system is initialized and startup has been completed, the program is switching between the modules of standby and running, according to the input from the user. In standby, the inverter is switched off and the system is in hold. Only in the running state will the inverter be enabled.

When an error is detected the system moves into the error state, where the inverter is switched off to prevent further damage. Errors include overcurrent and overvoltage, but also an excess in the temperature of the inverter and batteries.

Only when the user has acknowledged and fixed the error, the state machine goes back to standby, and normal functionality is reestablished.

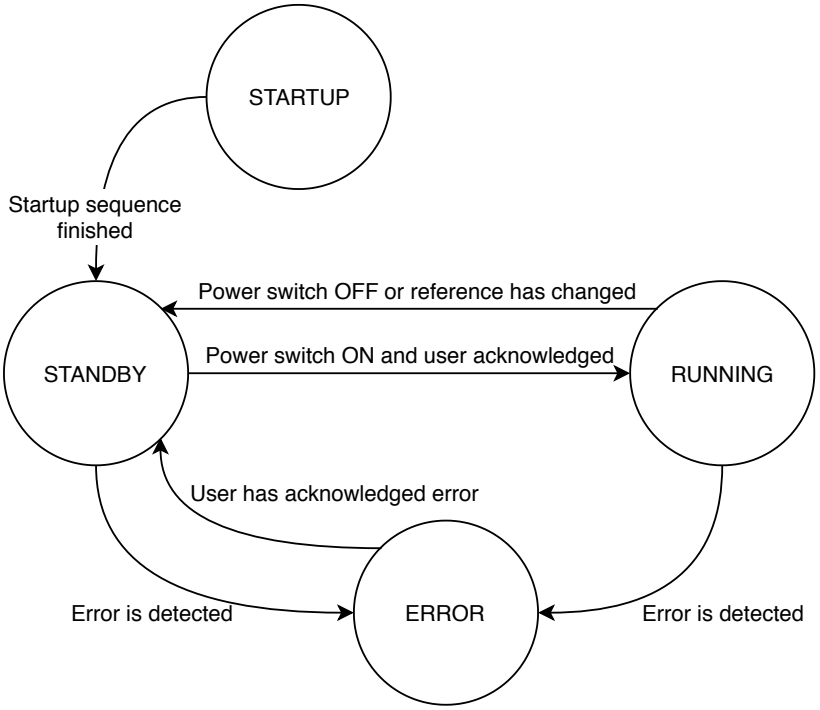


Figure 4.6: Finite state machine of the system manager.

4.2.2 Scheduler

As explained in the architectural design, the Scheduler is the module used for managing the execution frequency of the low priority tasks. In this section, the Scheduler, its main challenges and relevant pieces of code are discussed. First, three listings showing relevant functions are introduced, then a walk-through example over every function is performed.

The system must perform several tasks. A task is a set of functions that perform a variety of actions, which share the same execution period. Some of the tasks are run every few milliseconds, like servicing the watchdog. Other tasks only have to be run once per second, like the temperature calculation. The Scheduler is a framework which allows the designer to change the execution frequency of the modules.

A scheduler must have a Task Control Block (TCB). This TCB includes the function pointer, the state, the cyclicity and the countdown until the next execution, for every task. So every task has a TCB of its own containing this essential information. The next snippet is an example of the TCB definition for the 10*ms* task.

```
1    taskList[task10msItem].functionPointer = task10ms;
2    taskList[task10msItem].taskState = INACTIVE;
3    taskList[task10msItem].cyclicity = 10000;
4    taskList[task10msItem].timeLeft = 10000;
```

Listing 4.1: Task Control Block in 10 ms task.

Aside of the initialization of the task, the implementation has two main blocks. First, there is an infinite loop analysing whether a task is ready to be run. If that is

the case, the task is executed. On the other side, the CPU timer is triggered every millisecond and the tasks' state is changed to READY when its countdown reaches 0.

In the next snippet the infinite loop of the scheduler can be seen. The DSP is constantly checking whether the tasks are ready. Every 100*us* this loop will be interrupted by the High Priority modules.

```
void scheduleTasks(void)
{
    Uint16 taskListIndex = 0;
    for(;;)
    {
        for (taskListIndex = 0; endOfTaskListIsReached(taskListIndex);
    taskListIndex++)
        {
            if (taskIsReady(taskListIndex))
            {
                runTask(taskList[taskListIndex].functionPointer);
                deactivateTask(taskListIndex);
            }
        }
    }
}
```

Listing 4.2: Infinite loop which checks whether a function is ready to be run and runs it if that is the case.

In the code below, the function called from the periodic 1*ms* CPU interruption. Here, the countdown until next execution for every task is analysed. If the countdown reached 0, it means that the task must be executed, and will be set as ready. If the countdown still hasn't reached 0, it is decreased.

```
void updateTasksState(void)
{
    Uint16 taskListIndex = 0;
    for (taskListIndex = 0; endOfTaskListIsReached(taskListIndex);
    taskListIndex++)
    {
        if (taskMustBeScheduled(taskListIndex))
        {
            setTaskReady(taskListIndex);
            restartTaskCountdown(taskListIndex);
        }
        else    decreaseCountdown(taskListIndex);
    }
}
```

Listing 4.3: Function called by CPU timer to check if every task's countdown reached 0.

To understand the previous code fragments, a walk-through will follow: The task for the example is the 10*ms* task, although the same procedure follows for all the tasks which are not in the High Priority group. It is initialized as inactive with a time left equal to its period: 10000*us*, as seen in Listing 4.1. Then, the infinite loop inside *scheduleTasks()* will check whether the task is ready, i.e. compare its TCB.taskState attribute to READY. If that is the case, the task will be ran by calling its function pointer. After the task execution finished, it will be deactivated by

setting TCB.taskState attribute to INACTIVE. At any moment during the execution of the scheduler tasks, the system can be interrupted by the CPU timer or the PWM interrupt. When the interrupt is ended, the scheduler will continue normal operation, inside the infinite loop. The update of the tasks' state is performed in the CPU timer interrupt, *updateTasksState()*. Inside this function, if the task must be scheduled, i.e. TCB.timeLeft is smaller than the ciclicity of the CPU timer, $1ms$, then the state is set to READY and the countdown is restarted.

A module like the scheduler is necessary in a complex system where many developers interact, as the execution of tasks must be properly organized. The system also becomes easier to review, as the cyclicity of every module can be analysed at a glance. A limitation of the current implementation is the fact that there is not any kind of time sharing method. So if any task grew too much it will block other tasks' execution, eventually leading to a watchdog reset. In larger systems with a higher amount of tasks, a CPU time sharing technique becomes mandatory. Another limitation of the scheduler is the absence of priorities in the tasks, then the buttons' handling is given the same relevance as the reference calculation. Developing such system which considers the priority of every task might increase the reliability of the system. This additional features, priorities and time sharing, would raise the scalability of the system, allowing it to grow without a limitation on the task management side. However, these features would lead to increased overhead and thus a decrease of the useful computation time, as additional time will be spent on context switching among tasks. [32]

4.2.3 High priority tasks

In Subsection 4.1.1 it was stated how some tasks have a higher priority than others and are triggered at PWM frequency. The high priority tasks consist of those in charge of running the control loop as well as monitoring critical errors. Running the control entails three main tasks; obtaining the values of the currents, calculating the position of the rotor and rotor magnetic flux vector and execute field oriented control. The currents are the most critical values to check since current peaks could damage the inverter in a short time. Therefore, these errors are monitored at switching frequency and are also a part of the high priority tasks.

An accurate measurement of the currents is very important for having a reliable control feedback. For this reason, they are measured at the cycle point where the instant value of the currents is equal to its average. As seen in Figure 4.7, this moment is at the mid point of the high or low parts of a PWM cycle.

On top of this, the delay on the update of the duty cycles has a big impact in the behaviour of the phase currents. In general terms, it can be said that the lower the delay the better the performance of the control. Due to this, the firmware has been designed such that the duty is updated in one half of the switching period.

In order to ensure consistent an accurate current measurement, an interrupt is triggered when the PWM signal is half way through the high section of the period. Triggering at this moment facilitates to have the measurement of the average of the current as shown in Figure 4.7. This figure represents when the currents are measured as blue dashed lines and when the new duty cycle is set as red dashed lines. The time span between these moments, represented in light blue, lasts $50\mu s$

Green in the figure is not the reference but the average value which we are trying to capture. According to text. FA

when the switching frequency is $10kHz$.

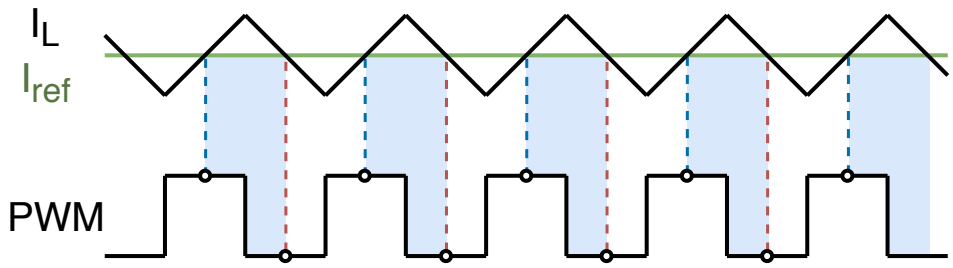


Figure 4.7: Representation of current behaviour with respect to PWM signal of one leg, in order to show the moments where it is measured.

The triangular carrier is used for triggering the PWM signals, this carrier is functioning continuously, no matter if the PWM is sent or not to the inverter. Also, this triangular carrier signal is reliably consistent in its period, meaning that the time between two triggers is always constant. This is useful for calculating rotor speed, since it is obtained by computing the change in position divided by the time passed, thus this time must be very accurate.

The order of execution of the tasks is then:

1. Measure the values of the phase currents.
2. Obtain the rotor position and its angular speed. Position and speed of the rotor flux vector are also calculated here.
3. Perform field oriented control steps, as explained in Chapter 3, and set new duty cycles.
4. Manage errors and perform necessary safety operations. The error managing can be performed outside of the blue region in Figure 4.7 since duties have already set.

4.3 User interface

The go-kart must be in constant communication with the user since it receives information in order to update the reference parameters. Also, the user must be able to get information from the program to understand the current status.

Three different paths have been developed in order to communicate with the user, these are the interface board, a graphical user interface (GUI) and the throttle pedal.

- **Interface board**: The interface board features many different communication channels: switches, buttons and potentiometers as inputs and LEDs to show software status. It also has external connectors to receive information from sensors (Section 2.4).

- **GUI**: The interface board can have limitations, mainly with the information that it is able to send to the user. The lack of a display does not allow to show numerical information for speed or temperatures, also references can only be set to an approximated value with the built in potentiometer.

  For these reasons, a GUI was implemented, allowing an easy display of all the variables as well as the opportunity to set very precise references, section 4.3.1.

- **Pedal**: The throttle pedal is an extension of the interface board. It can replace the potentiometer at the board when the go-kart is being driven.

The interface board and the GUI can be displaying data at the same time. However, they cannot set references simultaneously. For this reason, a button in the interface board and another one in the GUI have the purpose of changing between the different input references. Going from the board, to the pedal, to the GUI and back again. When this happens, the motor is stopped and the user needs to acknowledge the change, making sure that a different reference value is not set by mistake.

The main modules of this section are related to data acquisition and display. These are the digital input manager (in charge of managing switches and buttons) and the reference handler (section 4.3.2). Figure 4.8 shows the UML class diagram for the *User interface* block. It shows the *Reference handler* is the highest abstraction level in the block, and gather data from the different interaction opportunities.

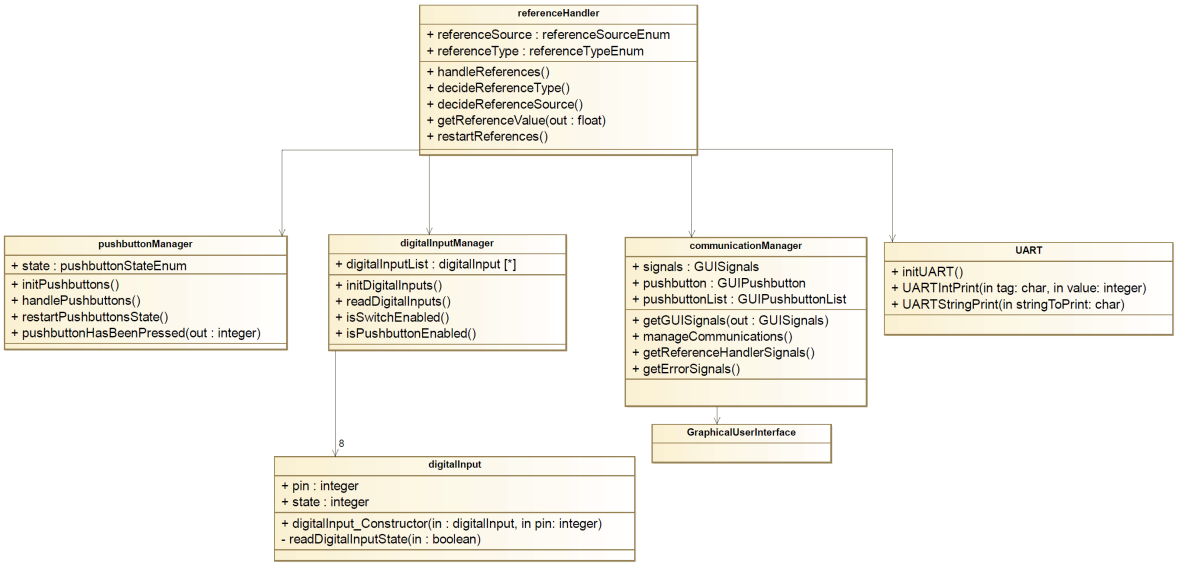


Figure 4.8: Class diagram for the modules in the *User interface* block.

4.3.1 Graphical User Interface

A graphical user interface has been created to interact with the user. The goal of the interface is to have a simple plug-and-play way of showing information to the user, it is a way of testing that the interaction with the DSP would be possible without

the buttons and potentiometers from the Interface PCB in the next iteration of the system. The available information is:

- General status of the system showing whether an error has been triggered, the system manager's FSM state and the status of the power enable switch.
- Input reference window, where the user can select the torque or speed reference and also how fast can the reference change. The reference is used by the control algorithm.
- Error status window, where every error source has a specific LED, providing an immediate way of knowing the failure reason.
- In the graphs and gauges, variables like rotor speed or MOSFET temperature, in addition to the actual currents as seen by the controller are graphically shown.
- In the reference handler window the actual reference fed into the controller can be seen.

what do you mean by reference rate change? Stef I reformulated a bit, what do you think now? NM

The interface was created with *Texas Instruments GUI Composer* and a snapshot can be seen in Figure 4.9. The main issue found is the low information frequency of the system, as the variables are updated just a few times per second. This leads to borderline performance in the case of plotting phase currents. Better performance might be achieved by changing the communication protocol or using another GUI Composer Widget which allows an array as an input. The GUI Composer needs the variables to show to be global. The whole software was designed avoiding global variables so a specific software component gathering the variables' value from other software components (SWC) was necessary. This new SWC called Communication Manager uses the interfaces provided by the other modules to acquire the values that might be interesting for the user, and uses other interfaces to react to the user's input to the GUI. A drawback of the framework provided by the GUI Composer is the fact that every time that a new binary is created, it must be loaded to the GUI project. This is a technical requirement as the GUI Composer needs to map the variables to the memory. However the GUI will be most used when the software has reached a high maturity level, where this handicap will not disturb.

I wouldnt include the second drawback as you say at the end it finally doesnt have effect in our case. Stef. There's only one drawback, I think it is good explaining it because otherwise we could use the GUI more frequently.NM

The GUI has proved itself a useful approach for data communication when the software development is finished. However, the fact that small changes require a significant amount of time makes the GUI impractical for debugging most of the modules.

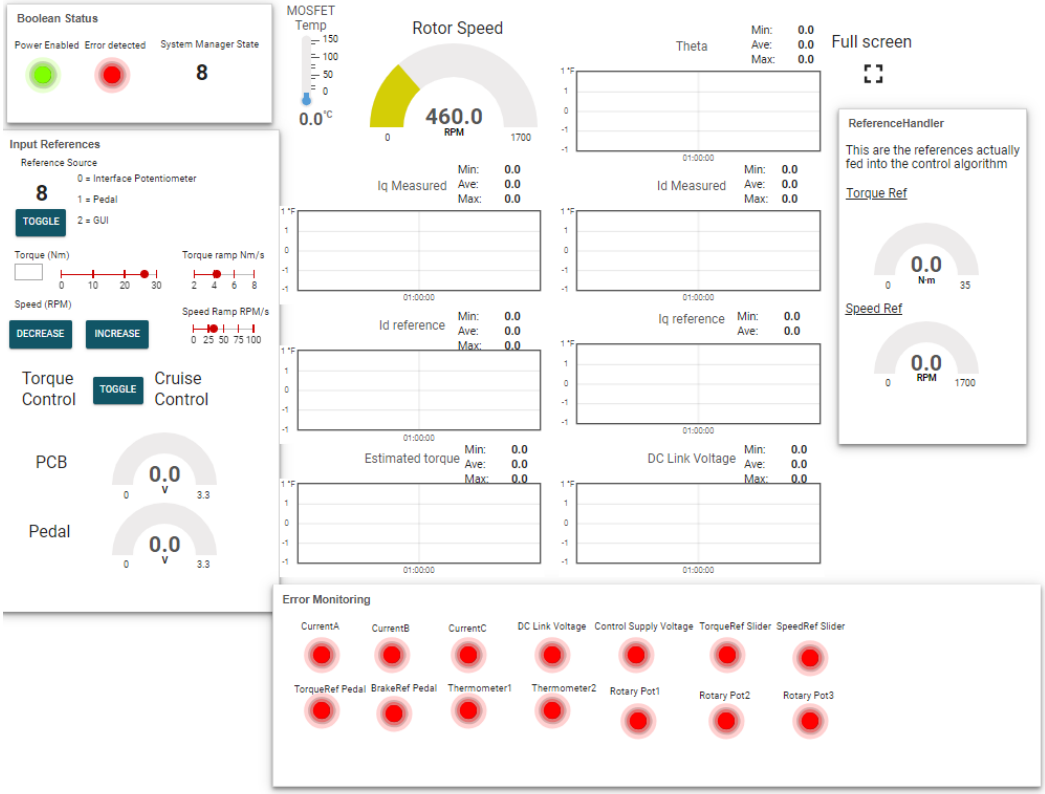


Figure 4.9: Graphical User Interface.

4.3.2 Reference handler

The goal of the reference handler is to check what reference source is selected by the user. Once the source is known, the value of that reference must be obtained. The possible reference sources are the pedal, the interface board and the graphical user interface. Also, open loop control (OL) or close loop field oriented control (FOC) can be selected and finally, when working in FOC, torque reference or speed reference are allowed.

References can be selected from the interface PCB, obtaining values from the analog input manager and sources from the digital input manager, also, everything can be set through the GUI. If cruise control is chosen, the speed is to be updated. Here the rotor speed is required for getting the new reference, this is obtained from the encoder block.

When the torque reference has been obtained and with the goal of avoiding overcurrents that may damage the system, abrupt reference differences are limited and a smooth adjust on the reference is performed, not allowing sudden changes, this is done in "Reference delta limiter". Finally references are limited to a preset value in "Saturation limit" in order to avoid exceeding the technical limits of the system.

Figure 4.10 represents the internal work flow of the reference handler module, including the input signals. It shows two horizontal lines corresponding to torque and speed control.

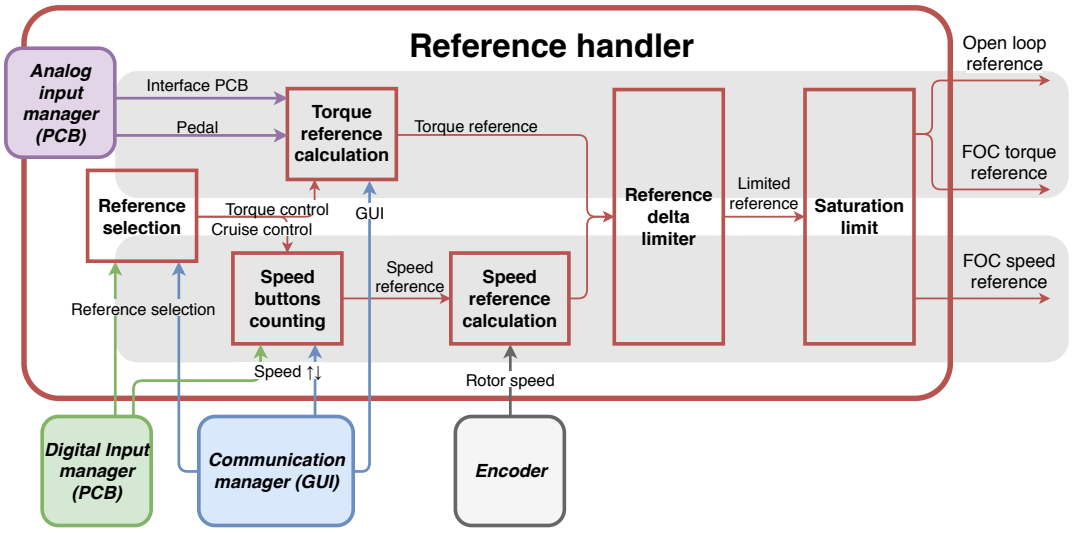


Figure 4.10: Reference handler work flow.

4.4 Motor control

The motor control block includes the set of modules which have the user reference provided by the Reference Handler as an input and calculates and sets the duty cycle as an output. This block consists of the High Priority tasks and has requiring time constraints as the delay between the ADC sampling and the duty cycle update will directly affect the control performance. For accomplishing the time requirements, code optimization techniques like removing divisions, moving non critical code sections to the scheduler or reduction of abstraction levels have been done.

Figure 4.11 shows the UML class diagram of the *Motor control* block. It shows the interaction between the different abstraction levels, and the selection between running in open or closed loop control. Figure 4.12 shows the class diagram for two fundamental modules used in this block: *Analog acquisition manager* and *Position calculator*. These modules will be discussed in the following sections

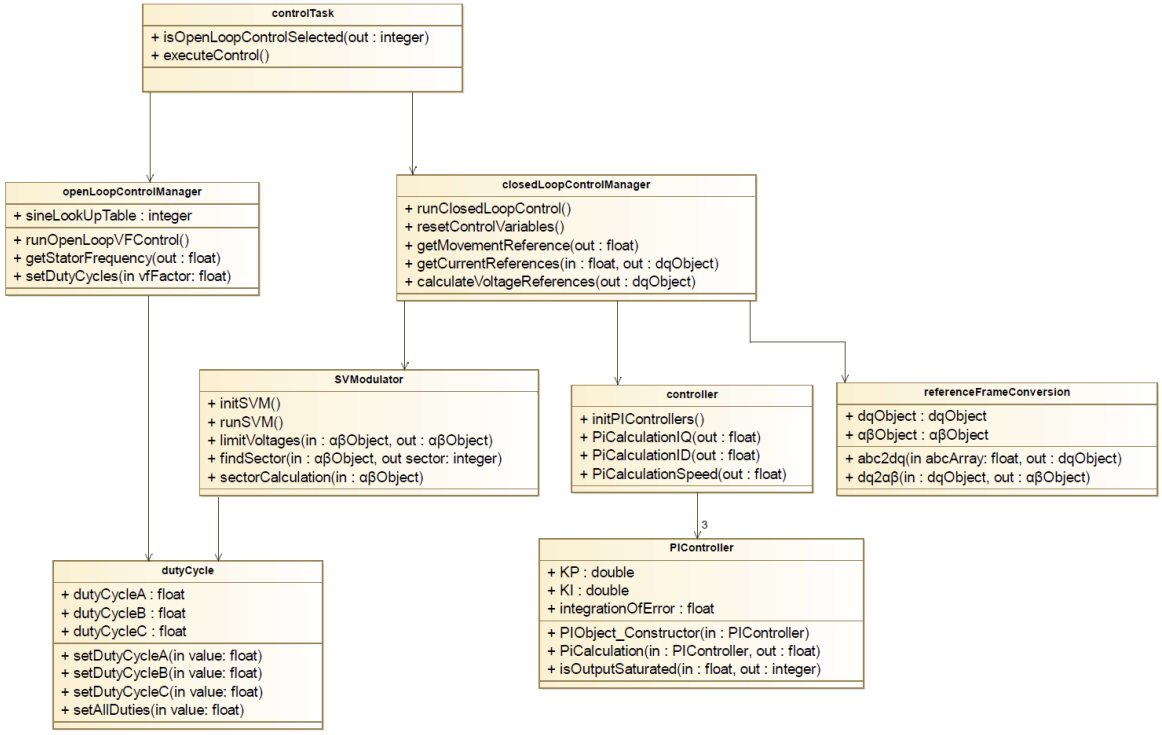


Figure 4.11: Class diagram for the modules in the *Motor control* block.

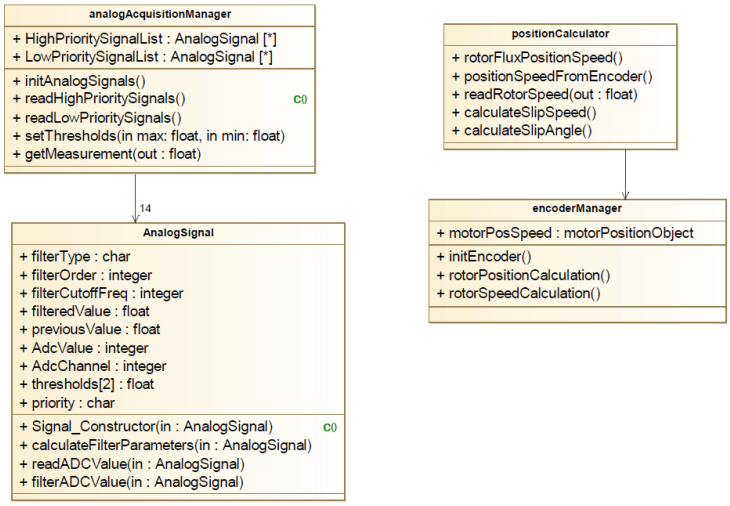


Figure 4.12: Class diagram for the modules *Analog acquisition manager* and *Position estimator*.

4.4.1 Position estimator

The control algorithm developed for the induction machine requires position and speed feedback from the motor. An encoder is attached to the rotor shaft, acquiring accurate information about the rotor position, speed can then be calculated.

There are three software modules that calculate rotor information. One module obtains both position and speed from the encoder measurements, another one estimates the rotor flux vector angle and its speed and finally the third module acts as a manager of these two and it is in charge of communication with the rest of the software.

Incremental quadrature encoder

The encoder used to obtain information about the motor shaft was selected during previous work [3]. It is an incremental quadrature encoder with 2048 steps per phase. It also features an index term to determine absolute position.

Incremental quadrature encoders have two channels, A and B. These channels provide square signals shifted 90º from one another. This way it is possible to obtain direction as well as angle difference. However, minor deviations could pile up over time, introducing long term errors in the rotor position, to avoid this issue, the index term is included. This term provides one pulse for every lap of the encoder, when this happens position is reset assuring absolute precise position estimation in the long term. In Figure 4.13 the signals produced by an incremental quadrature encoder are shown.

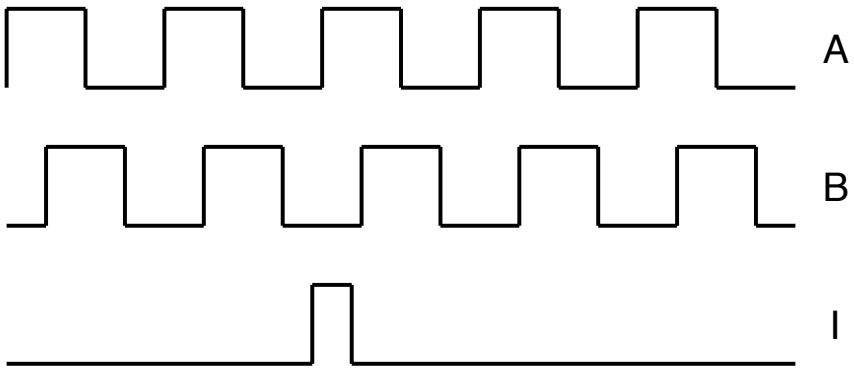


Figure 4.13: Signals produced by a general incremental quadrature encoder, including index.

Position estimator procedure

The information obtained from the encoder is the base to calculate four different parameters. These are position and speed of the rotor shaft as well as position and speed of the rotor flux vector.

The rotor position is directly obtained from the encoder signals. The encoder measures mechanical position of the rotor, which is then translated to electrical taking into account the number of poles. The rotation direction of the rotor entails an increase or decrease on the rotor angle position which would then be translated into speed.

After rotor position is known, its speed is calculated. For this, every 30 electrical degrees, speed is obtained. However, when rotating slowly, measuring every 30 degrees could lead to inaccurate results since the time gap between two speed updates would be high. Thus, if more than a certain time has passed since speed

was last obtained, it is calculated anyway . This way, it is ensured that reliable values are measured both at high and at low speeds.

don't understand what you mean with this. Stef, Tried to make it more clear, is it better now? AT

The rotor flux vector speed is the summation of the rotor shaft speed and the slip speed. An approximation is performed to calculate the slip speed as shown in Equation 4.1, a further explanation of the equation can be found in Subsection 3.2.2.

$$\omega_{sl} = \frac{L_m}{\tau_r} \cdot \frac{i_{qs}}{\lambda_{dr}} \tag{4.1}$$

In order to obtain the instantaneous position of the rotor flux vector, its speed must be integrated. Software integration requires the summation of the last known position and the last speed value divided by a controlled period of time. The measurement might then not be totally accurate since minor errors in rotor speed or in the time would entail errors in the position which would then pile up for future iterations. Nevertheless, since the calculated value of the flux position is used as a reference for controlling the machine, it should adjust its behaviour making minor errors unimportant.

why isnt trivial and what approximation is this? Stef. I rephrased the paragraph, is it better now? AT

Figure 4.14 shows how position and speed related information are obtained.

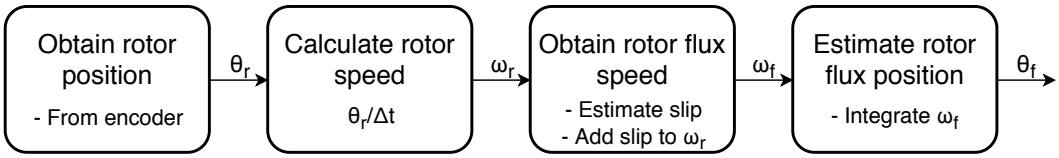


Figure 4.14: Encoder procedure for obtaining rotor parameters.

What about start-up? How do you decide which direction will it start? Lajos I still don't understand how to come around this comment, I need to talk to one of the Nicos about it AT.

4.4.2 Analog acquisition manager

A big part of the base of the firmware is the analog acquisition manager. This block initializes the ADC's of the DSP and handles the measured values, for the use in upstream blocks. This includes reading from the ADC registers, filter the achieved value and converting it into a workable value for the other modules.

The analog signal object

A *AnalogSSignal* struct type is created to handle the data of every analog signal in the system. To make the software more versatile and easy to adapt to the specific signal of interest, this struct includes selective parameters. These includes selective parameters like ADC channel and cut-off frequency for a digital filter. Regarding sampling, the current value, the value at the previous sampling instant and the filtered value are saved. Lastly, maximum and minimum thresholds for the signal are set. These are used by the *Safety* block to monitor the functionality of the system. All the variables included in the *AnalogSignal* are shown in the UML diagram in Figure 4.12.

Two different structs are created to contain the analog signals, one with the currents and one with all the others. This is done to minimize the conversion time

for signals used within the time critical high priority pipeline. These structs will later be mentioned as the high and low priority signals, with the currents being the high priority.

ADC settings

For every signal three registers need to be configured: ADC channel, trigger selection and acquisition period. The code snippet at Listing 4.4 shows an example of the settings of these registers.

```
1  // Configure ADC for phase A current measurement
2  AdcRegs.ADCSOC3CTL.bit.CHSEL        = CurrentSignalList.currentMeasA.
   adcChannel;
3  AdcRegs.ADCSOC3CTL.bit.TRIGSEL      = TRIGGER;
4  AdcRegs.ADCSOC3CTL.bit.ACQPS        = SAMPLING_RATE;
```

Listing 4.4: Example of setting the ADC registers.

The channel is received from the struct and is defined when creating the signal. The trigger defines when the conversion starts. The trigger source can be selected to be a CPU timer, a PWM signal or done by software. As PWM signals will be generated for the control, one of these is also used as the trigger. This ensures that new measurements are ready for every switching period. Further benefits of this are explained in Section 4.2.3. The acquisition period defines the amount of clock cycles used in the sample and hold window. The total time to process the signal is then defined as the sum of the sample window and a constant conversion time of 13 cycles. Where the conversion time is the time needed to convert the measured voltage into a digital value. Examples from the technical manual of the DSP are shown in Table 4.1. It relates different values of *ACQPS* to the total process time of the ADC. For decreasing the sampling time and make the system response faster, a $ACQPS = 6$ is selected. [29, p. 492]

ADC Clock	ACQPS	Sample window	Conversion Time (13 cycles)	Total time to process
$45MHz$	6	$155.56ns$	$288.89ns$	$444.44ns$
$45MHz$	25	$577.78ns$	$288.89ns$	$866.67ns$

Table 4.1: Timing table for acquisition period selection.

Signal acquisition

Reaching EOC (End-Of-Conversion) of the sampling, triggers an interrupt in the ADC block. When this interrupt is received, the analog signals are read from the ADC registers. To optimize the time consumption of the signal acquisition, only the high priority signals will be read at this point.

The total signal acquisition is executed by two commands, *readAnalogSignals()* and *calculateFilteredValues()*. Both commands take a list of signals as inputs. They iterate through the list by the use of a pointer, to read the ADC registers and filter the readings respectively. The function *readAnalogSignals()* is shown in Listing 4.5. It takes the address of the first position in either the high or low priority signal list

and the size of the list as input arguments. The *initialMemoryPosition* is initialized to the first position in the list. The amount of elements in the list is determined by dividing the entire size of the list by the size of the structure *AnalogSignal*. The sum of the initial position and the number of elements, will be the last position of the list. By knowing the first and last position in the list, it is possible to iterate the pointer *structPointer* through it and update the value of the signal the pointer has reached.

```
void readAnalogSignals(void *signal, int size)
{
    AnalogSignal *structPointer;
    AnalogSignal *initialMemoryPosition = signal;
    AnalogSignal *finalMemoryPosition = initialMemoryPosition + size/
    sizeof(AnalogSignal);

    for (structPointer = initialMemoryPosition; structPointer <
    finalMemoryPosition; structPointer++)
        readADCValue(structPointer);
}
```

Listing 4.5: Function to read analog signals from ADC.

Digital filter

The last step of the *Analog acquisition manager* is to filter the ADC readings. To assure smooth signals without ripples and reducing the noise, a digital filter was included. Implementing a digital filter makes the system more flexible, as it can be removed if the hardware filter is sufficient. A first order low-pass filter was designed with an adjustable cut-off frequency. Equation 4.2 shows the calculation for the filtering.

$$y_n = x_n \cdot a_0 + y_{n-1} \cdot b_1 \tag{4.2}$$

The parameters a_0 and b_1 can be calculated from the execution frequency and desired cut-off frequency of the filter. The equation uses both the current reading (x_n) and the previous filtered value (y_{n-1}), to calculate the current filtered value. The two parameters are calculated as in the following equations.

$$b_1 = e^{-\frac{1}{f_{filter} \cdot \tau_{filter}}} \tag{4.3}$$

$$a_0 = 1 - b_1 \tag{4.4}$$

Where f_{filter} is the cut-off frequency and τ is a function of the filter execution frequency. [33]

Interfacing with other modules

For the use of the ADC readings in other modules, interface functions have been created. The purpose of these functions is to have an easy way of receiving the value in the correct unit of the measured signal. As an example, although the temperature sensor has a voltage output, this interface will directly return the actual temperature. This will also ease the use of the ADC readings in other software blocks, as they also are a fundamental part of the *Safety* and *User interface* blocks.

The Interface functions are divided into two functionalities: updating the signal lists with new readings, and receiving the readings from the list. The functions *readHighPrioritySignals()* and *readLowPrioritySinals()* should be called to update the signal lists. These are simple to use and clearly tells the user the priority of the signals. To receive the readings from the lists, *get*-functions have been created. These include the transfer function of the sensor circuits. The function to get the value of the DC-link voltage is shown in Listing 4.6. Where *DC_LINK_TO_VOLTAGE* is the transfer function and *DC_LINK_OFFSET* is the calibrated offset found for this specific measurement. This approach simplifies the code and makes it cleaner in the higher level modules, where the measurements are being used.

```
1 float getDCLinkMeasurement(void)
2 {
3     return (AnalogSignalList.voltageMeas36.filteredValue *
    DC_LINK_MEAS_TO_VOLTAGE) - DC_LINK_OFFSET;
4 }
```

Listing 4.6: Function to receive the DC-link measurement from the signal list.

4.4.3 Closed-loop control manager

The modules of the *Motor control* block are regulated by the closed-loop manager. This is where the developed FOC model from Section 3.2.3 is implemented. Further reading about the FOC can also be found in that section.

The flow of the closed-loop manager is shown in Figure 4.15. The first step is to collect the reference values and the measured 3-phase currents, from the reference handler and the analog acquisition manager respectively. Then the currents are transformed into the dq reference frame, to be used in the PI controllers. The next step is to transform the voltage references from the PI controllers into the $\alpha\beta$ reference frame. These can then be used as references for the SVM algorithm, which will generate the necessary duty cycles for the three legs of the inverter.

Determine how much we should describe in this section. As all of it is already described in the control chapter, i think this should be at a minimum, . I think It's OK, we spent time in the implementation but I think describing the PI implementation for example might be a bit too much. NM - That should be addressed at the control section... Lajos

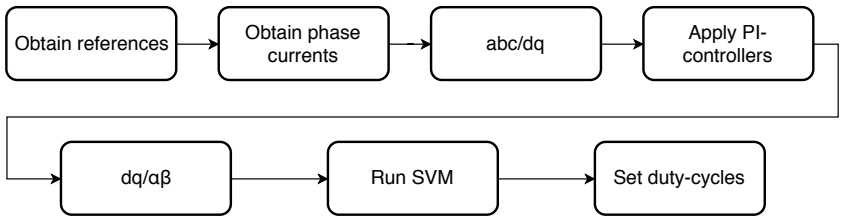


Figure 4.15: Flow diagram for the closed-loop control algorithm.

4.5 Safety

To improve reliability and to increase safety, the error recognition and management must be carefully considered. This is specially important when working with high power devices where errors could entail major issues concerning the equipment or even the user.

The possible errors include overvoltage, overcurrents and high inverter and battery temperatures. The software also features a watchdog which forces a restart

if an unexpected behaviour takes place or if the system is caught up in a loop. However, checking errors is a very time consuming task since these mainly come from the analog signals. For this reason, the most critical errors, overcurrents, are checked at switching frequency, while all the others, are checked from the scheduler at 1kHz.

Figure 4.16 shows the UML class diagram of the *Safety* block. It shows the different levels of abstraction inside this block. In the following sections the *errorManager* and the *watchdogTimer* will be explained further.

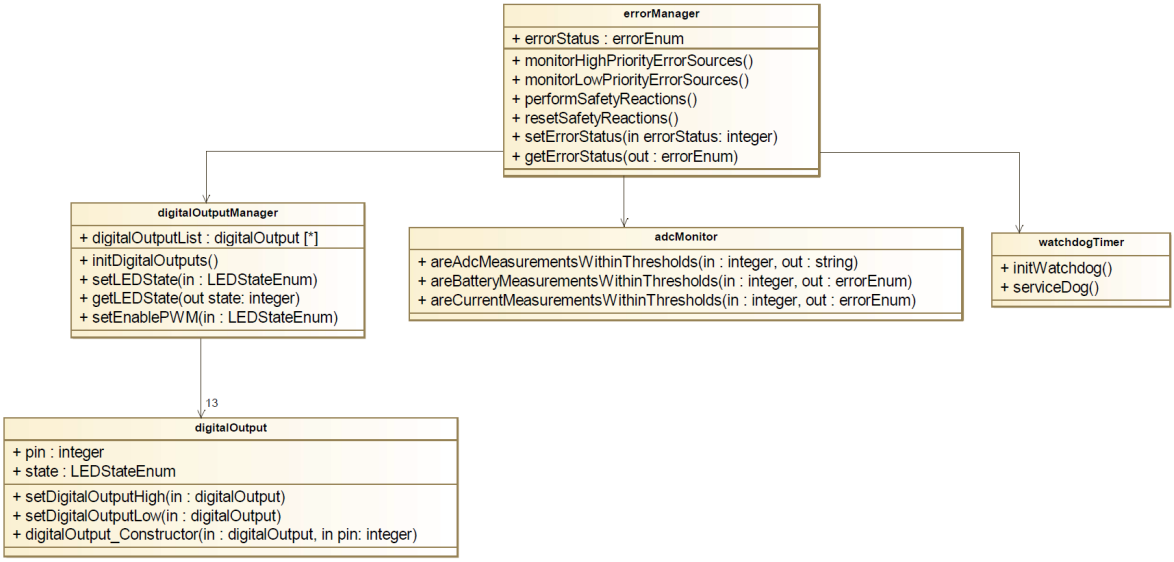


Figure 4.16: Class diagram for the modules in the *Safety* block.

4.5.1 Error manager

The error manager is a crucial part of the safety feature in the software. It keeps track of the flow from error detection to execution of the necessary safety reactions. This flow is shown in Figure 4.17. It shows that error monitoring can be both enabled and disabled. This is a useful feature during the development, to avoid error triggering if some sensors are not connected. If an error is detected, the error manager performs the safety reaction, and switches to error mode. The safety reaction includes disabling the PWM drivers and turning a LED on to indicate that an error has happened. The error mode can only be exited by the user, so the error source can investigated before starting the control again. The error status of every signal is latched, such that the errors can be shown through the GUI and the UART for debugging. When the user acknowledges the error to return normal operation, the signals error status are reset, the drivers are enabled again and the LED is turned off.

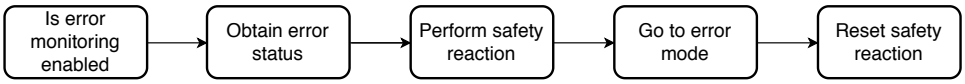


Figure 4.17: Flow diagram for the error managing algorithm.

The error manager was as the rest of the firmware, divided into high and low priority signals. This made it possible to check for the high priority signals only, which optimized the reaction time of the error detection.

Obtaining the error status

The first step of the error manager is to obtain the error status of the analog signals. The code snippet below shows the implementation of this. By the use of pointers, it compares the measured value with the defined thresholds. If the value is outside these thresholds, the algorithm sets the corresponding bit equal to 1. This is continued through the entire struct, to generate a bit array. This array is then checked for 1's, to check if an error has happened.

```
1  Uint16 getHighPriorityErrorStatus(void)
2  {
3      AnalogSignal *structPointer;
4      AnalogSignal *initialMemoryPosition = &CurrentSignalList.currentMeasA;
5      AnalogSignal *finalMemoryPosition = initialMemoryPosition + sizeof(
       CurrentSignalList)/sizeof(AnalogSignal);
6      Uint16 errorStatus = 0;
7      int i = 0;
8
9      for (structPointer = initialMemoryPosition; structPointer <
       finalMemoryPosition; structPointer++)
10     {
11         if (structPointer->filteredValue < structPointer->threshold[0] ||
               structPointer->filteredValue > structPointer->threshold[1])
12             errorStatus |= 1<<i;
13
14         i++;
15     }
16
17     return errorStatus;
18 }
```

Listing 4.7: Function to get the error status of the three current signals. It checks if the measured value is outside the user defined thresholds. If this is true the value of the corresponding bit is set to 1.

4.5.2 Watchdog timer

The watchdog is an important component in the error detection of the system. The DSP must acknowledge periodically the watchdog. If the acknowledgement is not performed, the watchdog will restart the DSP. This is a very useful feature to prevent the system to get stuck in an infinite loop, or if the DSP freezes. This is also a way to monitor whether all the scheduler tasks are being run and completed, as servicing the watchdog is the last task performed by the scheduler every $10ms$.

The time for the watchdog timer to overflow is defined as the watchdog counter clock. This counter is calculated as $OSCCLK/512/n$. $OSCCLK$ is the system defined oscillation clock and n is a user selected constant to define the time to overflow. The default value at $n = 1$ is used which will result in an overflow after approximately $13ms$. [29, p. 107]

4.6 Software validation

The validation phase goal is to determine whether the system is behaving as it was designed to. This is the final step after design and implementation and is of critical importance to achieve a reliable system. In this section the techniques used and some of the results will be discussed.

4.6.1 Task Timing

A main concern in real time systems is how often and in what way are the tasks executed. This was firstly introduced in Subsection 4.1.1, where the tasks' timing behaviour was planned and later in the Scheduler explanation in Section . Now, it is desired to test whether the functions are executed with the proper cyclicity. For that, when the function was entered, a digital output was set to a high logic level and when the function was left the digital output was set to a low logic level. This signal was monitored with the oscilloscope. Setting the signal itself already takes some time, although very small: it was measured to be $0.1\mu s$. That time is considered in the summary table, so in the numbers appearing in the table, this $0.1\mu s$ have already been subtracted 4.2. One of the concerns was to fit all the high priority tasks within $100\mu s$, which would be necessary if it is desired to run the whole High Priority block every switching period, at $10kHz$. In the table it can be seen that the maximum time is $74.4\mu s$, when the error is detected : the High Priority block time duration fits properly in the constraint. The time changes whether the error is detected or not, as when the error is detected, the error reaction is performed.

Where is the 0.1 microsec included in the table? Stef Added additional explanation. Is it good enough?NM

By using a digital controller, the system will have a time delay between the feedback measurement and the duty cycle update. This delay will depend upon the DSP features. In the used hardware, it is a product of half of the carrier wave period. This is because the PWM peripheral updates the duty cycle in every half switching cycle, $50\mu s$. Then, if the calculation takes $80\mu s$, the delay will be $100\mu s$. In the implemented software, the table shows that the total execution time of the high priority pipeline, without error monitoring is $48.2\mu s$. This means that the algorithm is able to execute the control with only half a switching period delay. The gain by minimizing this delay is a faster response time in the control, improving the performance. The execution time of the tasks depends of the CPU execution speed. The execution speed can be chosen by the user. The selected clock speed is the maximum possible for the used DSP: 90 MHz.

High Priority pipeline	
Task to be performed	**Time**
Read and filter signals	**8.7μs**
Read signals	2.9μs
Filter signals	5.8μs
Calculate rotor position and speed	**7.6μs**
Calculate position	1.2μs
Calculate speed	2.2μs
Calculate rotor flux position	4.2μs
Execute closed-loop control	**31.9μs**
Read references and currents	8.1μs
ABC to DQ conversion	6.2μs
DQ to $\alpha\beta$ conversion	2.4μs
Apply PI-controllers	6.9μs
Run SVM	8.3μs
Perform error monitoring	**-**
Without error detected	6.1μs
With error detected	26.2μs

Table 4.2: CPU consumption time by tasks. DSP Clock = 90MHz.

4.6.2 Debugging framework

The debugging framework consists of the set of techniques available to validate that the developed code is working as intended. If the developer doesn't have the proper tools for debugging, the code implementation could be slowed down or it might not be possible to check if the system is working properly. Thus, although creating additional scripts for validation might seem like an unnecessary task, it is important that the management team evaluates what testing framework will be necessary for the system under development. Most of the tests could be performed using on-chip debugging, where breakpoints can be set and the variables' value can be read. However, the Motor Control module includes time dependent signals and it is desired to analyse whether these signals evolve with time according to expectations. This was an especially critical module to validate, as misbehaviour could lead to damaging the hardware. During the control design phase, a Simulink model was used. There, the designer has access to every variable with respect to time, so a module which could replicate that practice was designed. The used developer kit includes a COM port within the USB connection, so using DSP's UART hardware over Serial communication was the easiest option.

For implementing the serial communication, C features the widely known *printf()* function, however that function is highly CPU and memory demanding. So a module for using the UART hardware was implemented. A thing to consider was that the UART hardware buffer is relatively small and thus a software buffer is necessary if a large amount of data is desired to be sent. A software buffer was implemented by implementing a Circular Queue (CQ) data structure. The CQ is a basic abstract data type. The queue consist of a First-In First-Out (FIFO) structure,

the first value entering the CQ is the first value leaving the it. The CQ is circular to avoid overflow, so when the end of the queue is reached, the next value will overwrite the first value. If this software buffer was not implemented, it would be necessary to explicitly wait for the UART to send the messages. This wait results in a very limited communication framework. The developed module allows the user printing integers with a title. During execution, a timestamp is printed every 200*ms*, its goal is to allow the event analysis with regard of time. An example of the UART use can be seen in the snippet below, where the *Timestamp* is printed.

```
1    // UARTIntPrint([Title ], (int)[variable]
2    UARTIntPrint("TS ", (int)sysClock);
```

Listing 4.8: Serial communication example use: TimeStamp printing.

When this code is executed and the serial communication port is analyzed from the computer, the information can be collected. The result can be seen in figure 4.18. In the picture, aside of the *Timestamp*, other variables for validating the reference frame conversion modules are also printed. Although the data is available, it is necessary to provide a way to process it, as the data will easily reach a size of hundreds of points. The chosen option consists of a Python script which has a text file as an input and generates the graph for every different variable title. Python was selected as it has powerful string processing capabilities. Also, in the case that real time plotting of the data is desired, there is plenty of literature available in the topic and the graph generation can be used as a base for a larger script . An example of the generated graph can be seen in figure 5.10. In the figure, the reference frame conversion was being validated. It's worth mentioning that even though in this case the frequency of the timestamp, the data frequency of the Timestamp and the variables under analysis that is not the mandatory.

I can see also phase b and c in the figure. I think if it isnt too long you should put in Listing 5.8 exactly what yolu are going to show in figure 5.14. Stef. I dont have that listing anymore and it would be just repeating listing 4.8, do you really yhink it is necessary? NM

Do we do this? Stef Not at the moment. NM

```
qr 45
qm 112
qr 45
qm 21
qr 45
qm -35
qr 45
qm -89
qr 45
qm -25
qr 45
qm 45
```

Figure 4.18: Serial communication example.

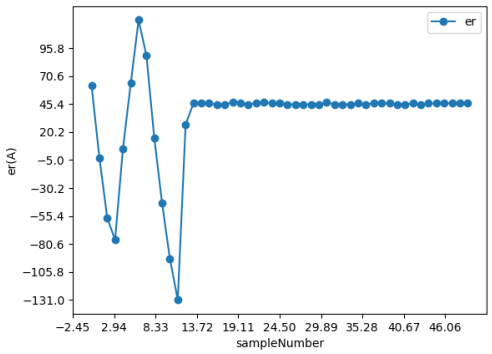


Figure 4.19: iq_{error}, printed to analyse whether the PI behaviour was as expected.

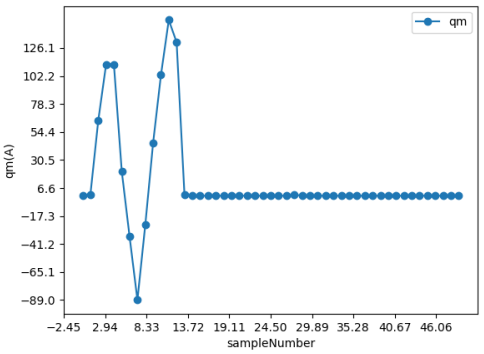


Figure 4.20: $iq_{Measured}$, printed to analyse whether the PI behaviour was as expected.

5 System tests

After the control was designed and simulated and the different modules of the firmware were individually tested, the integration of the hardware and software was to be done. To perform tests, it was necessary to control the motor by means of the developed firmware.

The integration process was likely to entail errors or mistakes that had been overseen during development. For this reason, well organized tests were key to having a successful output. Before testing the FOC control on the motor, it was necessary to make sure that other modules were working as expected, thus, open loop VF control was developed in order to have a spinning motor that would give feedback on different modules behaviour.

5.1 Laboratory setup

The laboratory setup consisted in all the necessary equipment for testing a motor with adjustable load. It was located in the main campus of Aalborg University. The used testing devices are listed below.

- High current power supply, required to feature up to $40V$ and $300A_{RMS}$ output in order to simulate the battery behaviour.

I made this number up, need to write down the correct value. AT

- Load motor, in this case, a compound machine was used. The energy was dissipated in a resistor bank, the motor requires a independent DC supply. By altering the voltage on the supply or the resistance of the bank, the load produced by the motor can be changed.
- Measuring devices, an oscilloscope and three high current hall effect probes are required, a torque measuring device is also useful to accurately measure the load.
- The inverter also requires active cooling so a 12V fan was needed. A two channel power supply was then used for powering both the fan and the interface board.

Since errors were likely to happen while testing, having a safe environment was important, the inverter and the motor were always protected to prevent a person from touching them, a high current power source with limited current output was also used to prevent inverter damage.

Add Diagram/schematic of the test setup. NM

include picture of lab setup here. AT

5.1.1 Compound machine

In order to test the induction machine under different load conditions, a load motor consisting of a compound motor was used. The used motor is a LAT 160, from Thrige. A compound motor is a DC motor where the excitation can be produced by both a series and a shunt winding. In our case, the shunt excitation was used to generate the magnetic field, while the series connection was used to connect an additional resistive load. Thus, the load torque can be changed by changing the excitation voltage or by changing the load resistor. In Figure 5.1 a simplified diagram of the machine can be found. Unused connections of the compound machine are not pictured. Points A, B, C and D are actual terminals of the motor, which the user can directly access to.

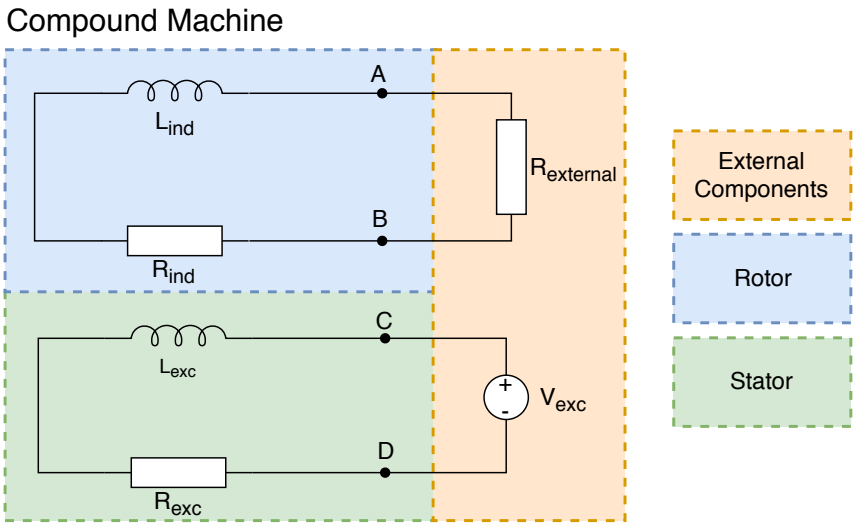


Figure 5.1: Main blocks of the used compound machine

The power which the compound machine is able to dissipate will depend upon the external resistor $R_{external}$: the lowest the resistor, the higher will be the torque load and the same behaviour applies to the excitation voltage V_{exc}. The load will also depend of the speed. Some points gathered during the open loop test can be found in the table below.

I would add the table in the section open VF results AT

Load motor characterization			
Speed$[RPM]$	$V_{exc}[V]$	$R_{external}[\Omega]$	$\tau[Nm]$
211	95.9	1.54	4.9
455	95.9	1.54	6.7
980	95.9	1.54	5.8

Table 5.1: Load motor characterization.

5.2 Open loop tests

In order to test certain modules of the firmware and hardware under more realistic conditions, it is needed to meet some requirements on the setup. Having the equipment listed in the previous section allows testing things that require rotation or high currents. To have measures on the current sensors, a direct current can be fed through one phase of the motor, this test is also valid to test the behaviour of the temperature sensors on the MOSFETs. However, in order to test other things like the encoder or the torque load, the motor must rotate.

Since FOC consists on a complex firmware, if errors are discovered when tests were done, it would have been very complicated to find the root of any issues that might be happening. However, if everything else is verified before testing FOC, errors could then be more easily traceable.

For these reasons it was decided to test the firmware modules as well as the hardware with open loop control before going to FOC. The decided technique to use was V/f control. V/f control is widely used technique for induction machines, its main principle is based on maintaining constant air gap flux in steady state [34]. In order to obtain constant air gap flux, the ratio between Voltage and frequency remains constant.

For this, the voltage is adjusted depending on the stator frequency. The relationship is obtained at rated values. However, the fact that the stator frequency is being set means that there is no control over the rotor speed and the slip will vary depending on the load.

5.2.1 VF control objectives

The main objective of performing V/f tests was to be sure that everything was working before starting testing FOC. Prior to performing tests on the motor, most of the software modules had been tested individually. However, the open loop V/f control tests would be useful to validate the integration of the firmware and the hardware, some of the tests performed and their objectives are listed below.

- **General validation of the system**. This test consisted in making the motor spin without any load being able to control the rotational speed from the interface board. This test would verify the general functioning of the firmware; system manager, scheduler or reference handler were tested together. Hardware equipment was also tested for the first time under spinning conditions. This test was an important milestone in the project progress since it was the first time the motor was spinning.

- **Tuning and validation of software modules**. A series of tests were conducted with the objective of tuning the modules related to the encoder and to analog to digital conversion, including the correct functioning of the current sensors. The motor was spun while its speed was being measured with a tachometer, the analog signals were also being noted. The values obtained from external measuring devices were compared to those internally calculated allowing fixing.

- **Load compound machine**. The load torque that the compound machine produces was to be important during FOC tests. During V/f control it was possible to also test this machine as well as to gather some data points at different conditions.

- **Error manager**. The error manager was also tested on V/f control. Errors were intentionally triggered to validate the performance of the error manager. The error manager was the main way of avoiding failure if unexpected things were to happen during FOC tests. In order to trigger errors, thresholds were reduced and conditions were taken over them.

5.2.2 VF control results

The *general validation of the system* showed many minor bugs in the firmware that were corrected until it was possible to make the motor spin. Although it was possible to make the motor spin at a low DC voltage, when this voltage was increased to the rated conditions one of the inverter leg's driver was damaged. This was one of the main problems encounter in the project and it is explained in more detail in Section 6.3.1.

During *Tuning and validation of software modules*, it was found that the encoder information gathered during the report made in 2018 on the same topic [3] considered the wrong number of pulses per revolution, since it really features 4096 pulses instead of 2048. Also, minor deviations on the currents, voltages and temperature measurements were corrected.

here I would add the table about load motor that was included in the previous section along with some explanation. AT

5.3 FOC implementation

After developing a valid control logic in Simulink and then implementing it into the DSP, the performance of the achieved system must be studied. As explained in the previous section, a controlled environment was built. In this section, the simulation results of the designed control, the testing results of the real implementation and a comparison among both is described. At this point, the available debugging techniques are considered: as the controller is being used and the motor is spinning, setting breakpoints is not allowed. If a breakpoint was set, the control of the motor would be lost. Then, as feedback to analyse the system behaviour is needed, the variables are sent through Serial communication to the tester. This variables can be investigated off-line. In addition to the information sent through the Serial communication, some variables can be measured using an oscilloscope. This is

the case especially for ABC currents, where the initial transient or the steady state behaviour might explain particular flaws in the developed algorithm.

5.3.1 Torque control

Simulation results

The simulation results obtained for torque control including the mechanical load torque derived in Section 2.5, which is dependent on the motor's speed, will be analyzed for different torque references. First, the torque reference is set to constant rated torque ($T_{e,ref} = 30.04Nm$) in order to analyze the dynamic behaviour of the system under rated conditions.

Trated

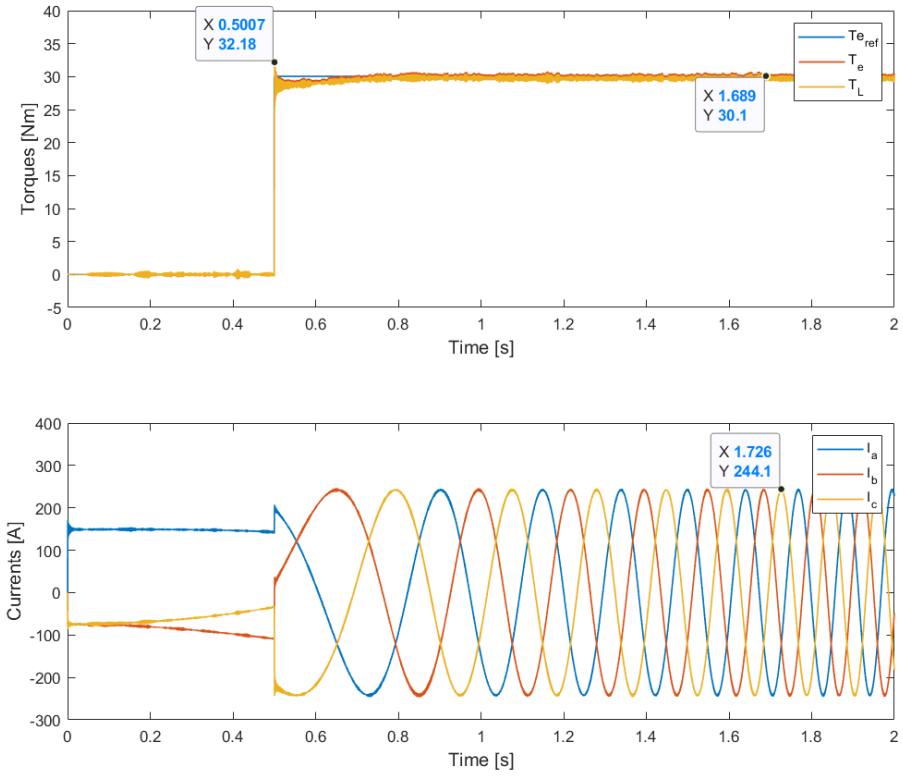


Figure 5.2: Offline plot of serial communication data. Test of reference frame conversion for currents.

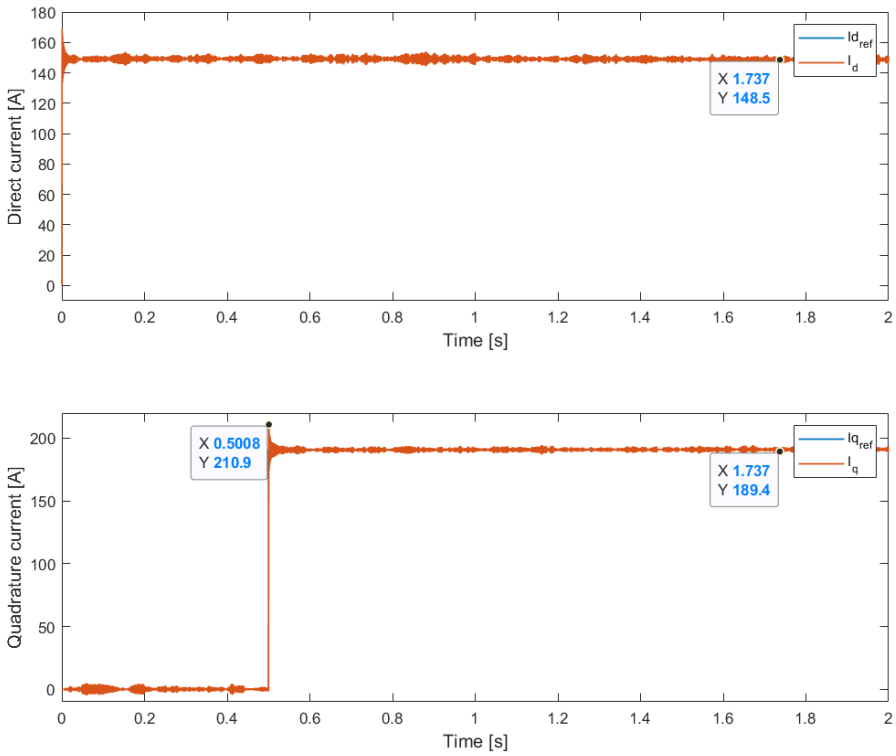


Figure 5.3: Offline plot of serial communication data. Test of reference frame conversion for currents.

Varying Tref

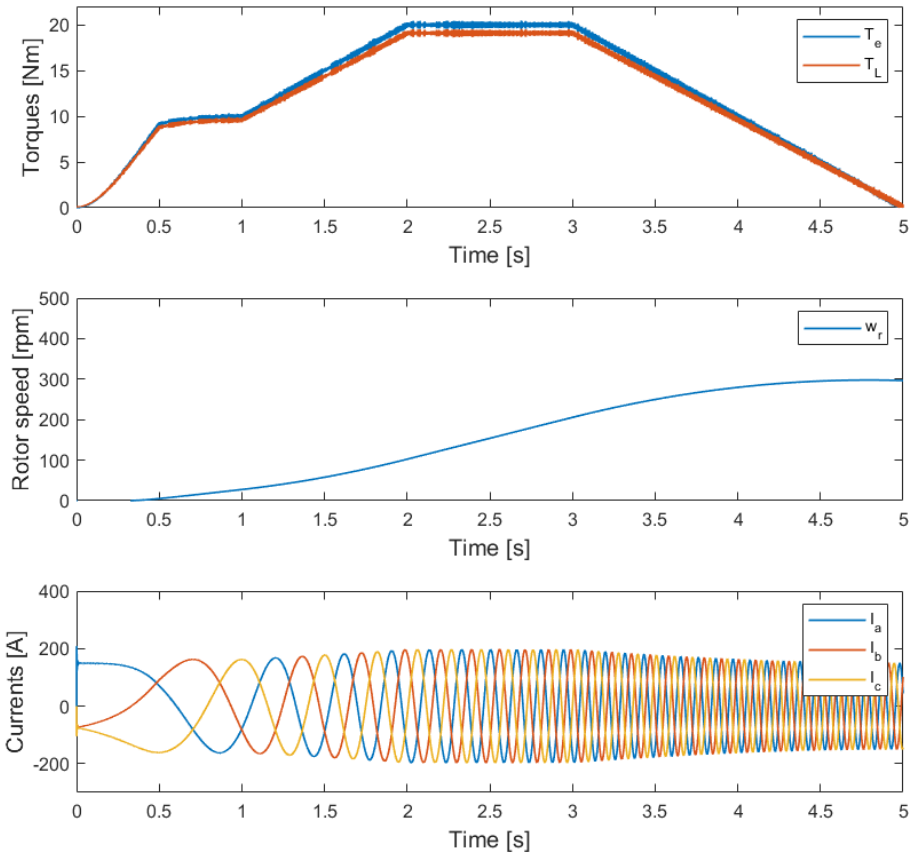


Figure 5.4: Offline plot of serial communication data. Test of reference frame conversion for currents.

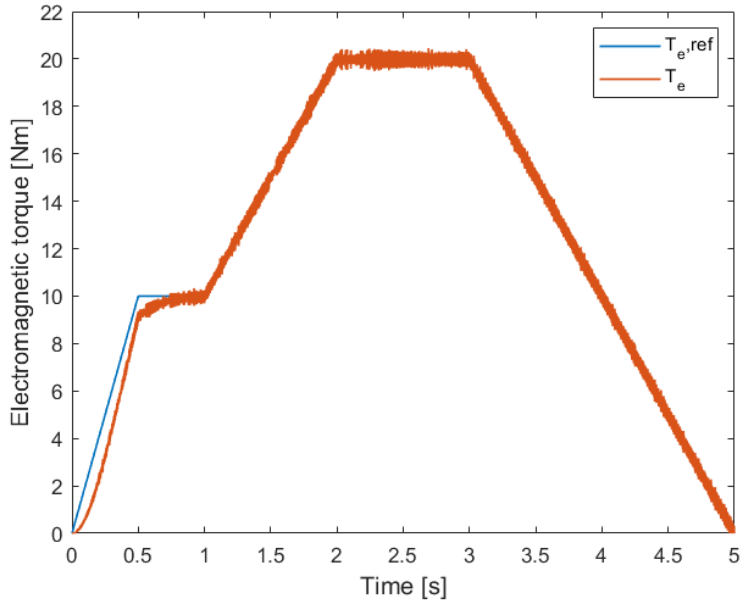


Figure 5.5: Offline plot of serial communication data. Test of reference frame conversion for currents.

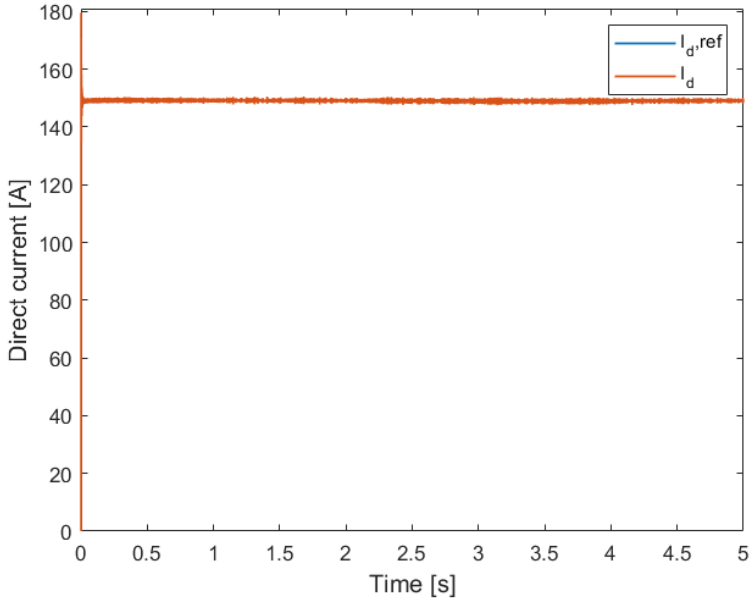


Figure 5.6: Offline plot of serial communication data. Test of reference frame conversion for currents.

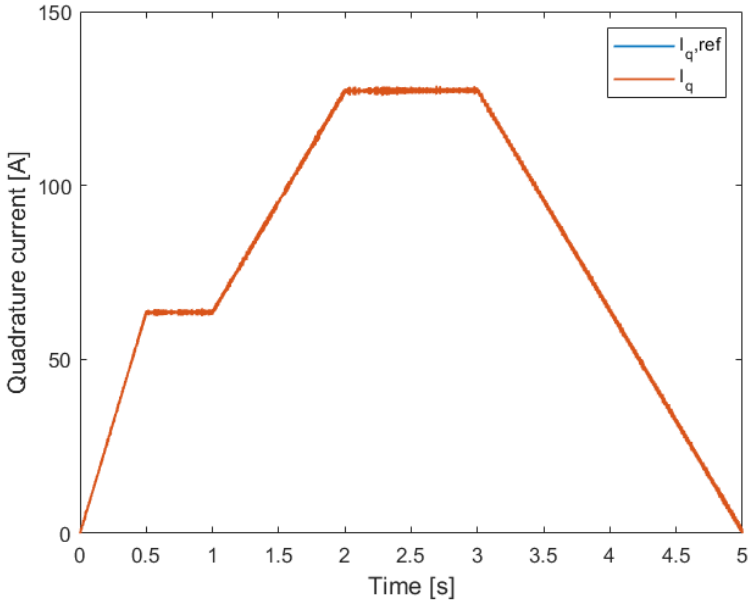


Figure 5.7: Offline plot of serial communication data. Test of reference frame conversion for currents.

Test and comparison

The torque control test must validate whether the controller is able to regulate the torque appropriately against different kinds of situations and also to quantitatively measure its figures of merit. The set of tests can be seen in the following list, shortly describing the test and defining its output. Note: the torques are in steady state, not during transients.

1. 50% of rated torque. Determine steady state torque error and system speed during startup.
2. 100% torque?? MAYBE
3. Torque regulation against sudden changes in the load. In this test the load will be suddenly increased by dropping the external resistance, the torque response will be analysed.

5.3.2 Cruise control

5.3.3 Regenerative braking

Tvarying positive and negative

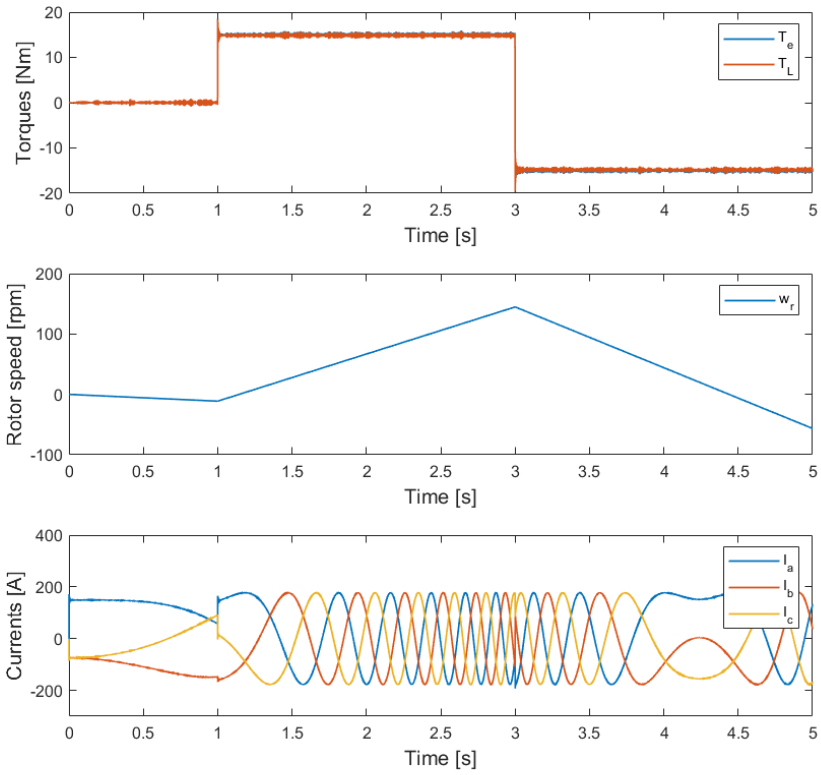


Figure 5.8: Offline plot of serial communication data. Test of reference frame conversion for currents.

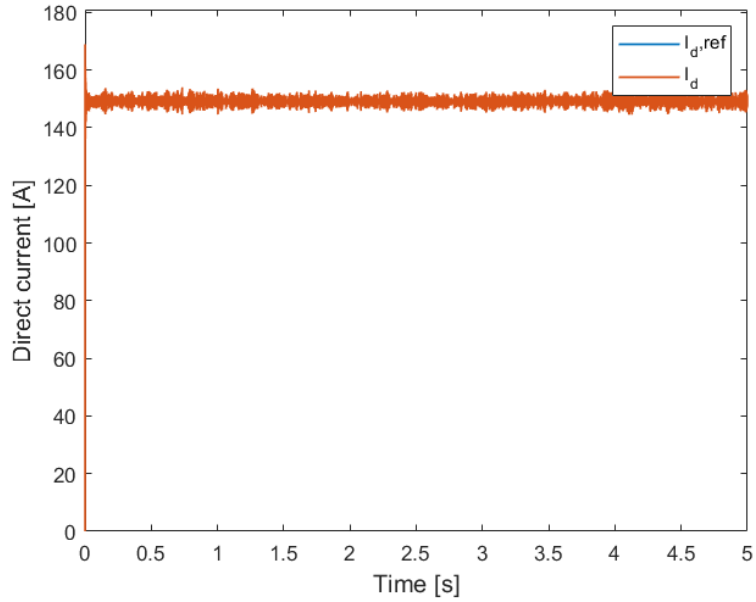


Figure 5.9: Offline plot of serial communication data. Test of reference frame conversion for currents.

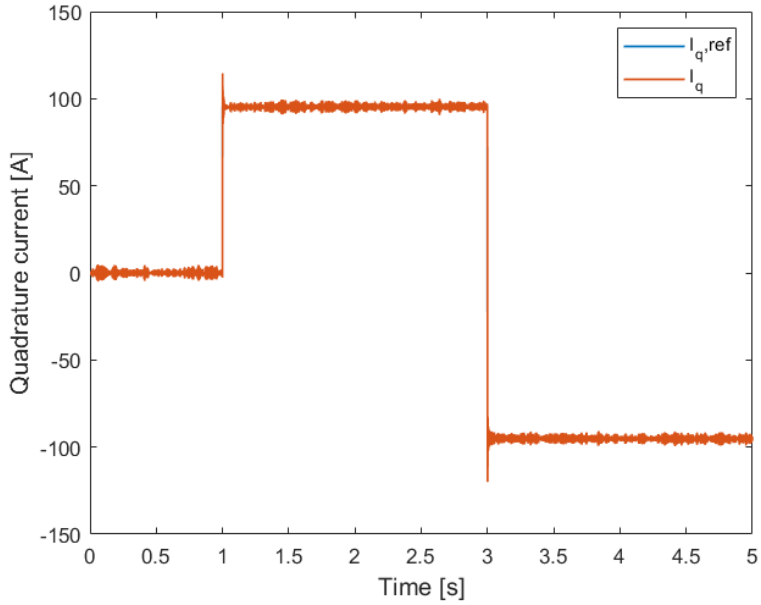


Figure 5.10: Offline plot of serial communication data. Test of reference frame conversion for currents.

6 Discussion

In this chapter, technical aspects of special interest are reviewed. This might be problems that had arisen during design, development or test, limitations of the developed system or remarkable insights that emerged as a result of the engineering efforts.

Add Chapters 2 and 3 discussion topics

6.1 Control

6.1.1 PI tuning

MAYBE it would be nice to discuss between the many different ways of PI tuning we used, how they affected the system. And the limitation of the pole compensation, that is weak in the sense that the poles of the machine will move when it heats up and so on.

6.1.2 Controller debugging

How we debugged the FOC algorithm.

6.1.3 Increase of switching frequency

pros, cons, constraints, main points to consider, code optimization needed. This can be part of future work. FA

I think it's a good idea to make it part of future work. AT

6.2 Firmware

6.2.1 User interface

Designing the User Interface(UI) requires a considerable amount of time and effort. In the current project, an Internet browser graphical user interface was developed, in addition to a hardware interface based on buttons, sliders and LEDs. Although these interfaces plus the UART communication were widely used during development, they are not the best fit in the case that it was mounted in the go-kart. The GUI implies the use of a computer and edit the project every time the code

is changed. In further implementation, a difference should be made between logging and user interface. The user interface could have similar transducer as the current interface PCB, but the shape should be analysed to be comfortable in the go-kart cockpit. The GUI Composer might be useful in telemetry applications, but in that case a self developed interface without the limitations of proprietary software could be more appropriate.

6.2.2 Error protection

The error manager showed to be an important tool in both the protection of the inverter and debugging the firmware. As it latch the error source when the error occurred, it was possible to track the error source and fix the bug.

The actual protection of the system was handled exclusively by the firmware. This could lead to delays in the error detection, caused by sampling, filtering and execution frequency. As a consequence of this delay, fast spikes in the currents could missed, which could lead to a fatal error. A way to increase the reaction time of the error detection, is to add hardware protection to the system. This will remove delays introduced by the firmware, and therefore only depend on the propagation delays of the logic gates.

6.2.3 Watchdog timer

Another safety feature implemented to protect the system was the watchdog timer. This was found useful to prevent the system of getting trapped in an infinite loop. This was experienced if the execution time of the high priority tasks exceeded the switching period. This meant that every time the firmware returned from the control, a new interrupt was being triggered. As a consequence of this, the scheduler would never have the time to call the next tasks, and thereby never service the watchdog. The watchdog would then force the system to restart, and prevent it from being damaged by uncontrollable currents.

6.2.4 UART

A limitation experienced regarding the UART was the execution time. As the UART was intended to be used debugging in the high priority pipeline, the execution time should support that. This was however not the case. Printing more than one variable at switching frequency would lead the UART to crash, and make the data useless. Multiple actions where taken to improve the performance of the UART. Firstly, the size of the queue was increased. This however did not remove the failure but only delayed it. secondly, the UART was modified to only saving data for one variable every time it looped the high priority tasks. This means that if three variables where to be saved, they would be so alternately. This decreased the resolution of the data, but increased the stability of the UART. Lastly, the data was only saved in the high priority pipeline, and then printed to the terminal by the scheduler at a slower execution frequency.

Another limitation with UART, is that it has to be connected to a computer. This could lead to a future problem, as it is not intended to mount a computer in the go-kart. Developing a proper logger for supporting testing in the go-cart

would be the next step in validation and testing. Often, this system use CAN communication, which is widely spread in the automotive industry.

6.3 Hardware

6.3.1 Driver IC damage

During open loop V/f testing, the driver IC UCC20520 got damaged: the input signal provided by the DSP was not being amplified by the driver IC. The error happened when the DC link voltage was increased to $36V$. A SPICE model of the inverter hardware was developed in a previous Semester Project, this model was used to determine whether the driver was working outside of the electrical rating found in the datasheet [35].

In the simulations, it could be seen that the voltage between the driver output and the negative supply was reaching $-0.43V$, being $-0.3V$ the absolute maximum rating. In Figure 6.1, the control signal coming from the DSP and the voltage between the low side output and the negative supply in the driver IC can be seen. Notice the voltage spike whenever the high side PWM changes state.

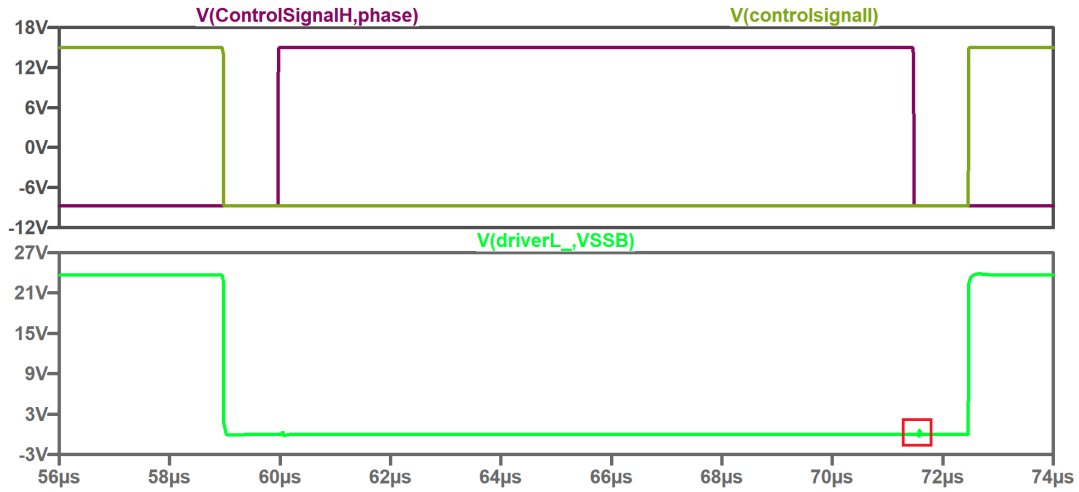


Figure 6.1: PWM control signals and voltage between driver output and negative supply.

In Figure 6.2, the detail of the voltage spike can be seen. In the lowest point it reaches $-433mV$, outside of the electrical rating of the device.

change color of bottom curve as light green is unreadable and also the zoomed picture below is different color. FA

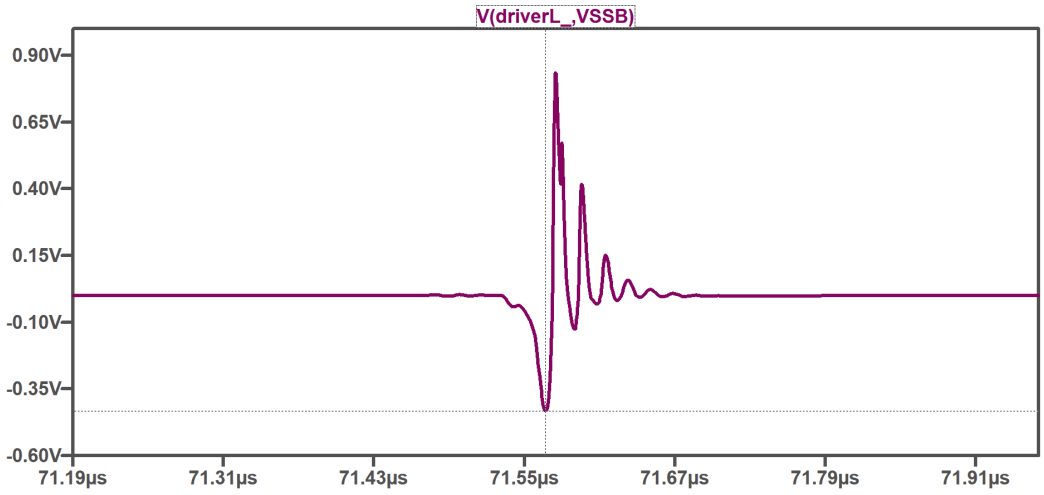


Figure 6.2: Zoom-in of the voltage spike.

If the driver circuitry is analysed, in addition to the UCC20520, there's a BJT push-pull which provides current amplification, as seen in Figure 6.3. Usually BJT transistors are used with a current limiting resistor, $R1$. The design included this current limiting resistor but with a value of 1Ω. This value is relatively small and if a voltage spike hit the gate of the MOSFET due to dv/dt, this resistor would provide scant protection to the driver IC. Increasing this resistor will not affect significantly the switching speed. The voltage drop in the driver due to a spike in the MOSFET gate will be inversely proportional to the current limiting resistor, then, if the voltage spike is desired to be decreased, the designer must increase this resistance. The resistance was increased to 100Ω. The result was that the negative voltage spike disappeared. This voltage now falls in the 100Ω resistor and causes a high power dissipation reaching $4W$ of peak, but an average of just $8mW$. The switching time of the power MOSFET was increased from $22ns$ to $23ns$, under the criteria of V_{DS} increasing from 10% to 90%.

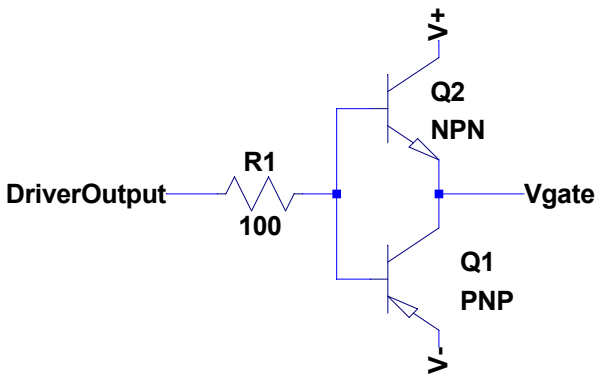


Figure 6.3: Push Pull amplification circuit.

6.3.2 MOSFET damage/PCB damage

assess whether the mosfet or the PCB was damaged and then discuss possible reasons and solutions

6.3.3 Motor Startup

During the motor startup, very large currents can happen in the stator due to the lack of back electromotive force. Afterwards, when the rotor is spinning, the current will stabilize to the rated condition current. Due to the risk of too high currents, exceptional attention must be paid to the motor startup procedure.

large initial current can be seen in simulation pictures. Refer. FA

An option could be to start the motor with V/f control and whenever a speed threshold is overcame, switch to FOC control. Although this technique might work, uncertain conditions would happen during the transient. Another option is to start from the beginning using FOC but dampening the PI parameters to have a lower bandwidth or lowering the maximum duty cycle values that can be set. The used procedure consisted of a mixture of both PI parameters dampening and maximum duty cycle limitation. In both cases, the limitation decreased linearly while the rotor speed increased.

to be completed

6.3.4 Battery pack

When test of the batteries was designed, a couple of assumptions were made. The batteries were expected not to meet the datasheet parameters since they are at least 8 years old and a complete individually charge was necessary since the conditions and the SOC in which they were preserved were not known. After having all the batteries fully charged, the impedance test was intended to begin. Unfortunately this process was slowed down because of lack of knowledge of how to handle the power supply and after, some issues with the grounding when using the acquisition device occurred. When everything was solved, the impedance test began. The batteries were connected in series since this is the configuration they will be used in the go-kart. While doing the current pulse test in order to obtain the internal impedance parameters, one of the batteries discharged faster than expected which indicates a low capacity. In order not to damage more the battery, the test was stopped and an individually charge discharge test for obtaining the batteries capacity was done. After doing the test it has come up that there is a big difference in the capacities (lowest capacity 18Ah while highest capacity 35Ah). This will affect the functionality of the go-kart since in a series configuration the lowest capacity is the one that influences the system. It will have a negative impact in the driving time which at high currents will be very short (less than 6 minutes) after which the batteries will have to be recharged. So, in the eventuality the go-kart can be tested, it will take many days to test different cases. The low capacity will also affect the regenerative braking results because inputting high currents in the batteries will lead to a fast charge and the rest of the energy will be dissipated as heat through mechanical brake. Also, some errors can occur because the batteries will be at different SOC which will be hard to predict in while they are in the series configuration.

The bad condition of the batteries also affects the possibility to create an accurate model of the batteries. The curves obtained in the practical tests could not be reproduced in the simulation since the batteries have different charging and discharging curves as it can be seen in Figures B.1, B.2 and B.3 and because of timing issues parameters obtained were not enough. Also at the moment the discharge

curve for battery B has not been obtained because of timing issues but it is expected to be obtained until the final draft of the report.

6.4 Project and time management

Explain issues related to project management, examples can be optimistic time to finish some topic leads to delay on project, inexperience, overkilling certain pm stuff...

6.5 Future work

I would add a short intro here. AT

Alternative approach in field oriented control

In this project only rotor flux oriented control is implemented. As mentioned in chapter 3 that field oriented control can also be implemented by orienting to stator flux. Although it is stated that rotor flux oriented control provides better torque control, it will definitely be interesting to investigate other strategies. Furthermore, in this current implementation the rotor flux is kept constant at its rated value. This is done to utilize the magnetic circuit to its maximum potential. But the rotor flux can be reduced by reducing the $d-$axis current, this phenomenon is know as field weakening and is used to allow the rotor to go beyond rated speed. This however comes at the expense of decrease in torque available at rotor [36].

Performance at higher switching frequency

Currently the VSI is operated at a switching frequency of 10kHz. This is done to limit the switching losses on the inverter. Also the firmware is designed to complete its high priority pipeline (FOC calculation based on machine values and reference and duty cycle update) just under $50\mu s$ which allows the update of duty cycle within first half of switching period and the rest of time is allocated to other system maintenance tasks. If the the high priority pipeline can be reduced to half of its current value then ADC sampling and duty cycle update can be implemented at twice of current switching frequency ($10kHz$) or increase the switching frequency to $20kHz$. Increasing the switching frequency to this high value will lead to a much quieter operation of motor as well.

Integrating the traction system in Go-kart platform

Ultimately the final target of the project is to integrate the go-kart with the developed traction system. This target however has been eluding the groups who have undertaken the project so far. At this moment the final testing on test-bench is being conducted if that is successful, the team will attempt to integrate everything on go-kart itself.

7 Conclusion

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

Jesper Kjær Nielsen
jkn@create.aau.dk
http://sqrt-1.dk
Audio Analysis Lab, CREATE
Aalborg University
Denmark

7.0.1 Key learning from this project

1. Hardware is tough
2. Software is tough
3. Life is tough

Bibliography

[1] *Global EV Outlook 2018. Towards cross-modal electrification*. International Energy Agency, 2018.

[2] C. Yigen et al. *Implementation and Control of an Induction Motor on a Go-cart*. Student Project Report. Aalborg University, 2011. 124 pp.

[3] J. Beer et al. *Development, modelling and implementation of an electrical drivetrain for a Go-Kart*. Student Project Report. Aalborg University, 2018. 116 pp.

[4] D. W.J. Pulle R. De Doncker and A. Veltman. *Advanced Electrical Drives. Analysis, Modeling and Control*. Springer, 2011. 438 pp. ISBN: 978-94-007-0181-6.

[5] A. Emadi. *Energy, power electronics and machines. Advanced electric drive vehicles*. Taylor & Francis Group, 2015. 604 pp. ISBN: 978-1-4665-9770-9.

[6] J. Lepka and P. Stekl. *3-phase AC induction motor vector control using DSP56F80x*. Tech. rep. 2002.

[7] M. H. Rashid. *Power Electronics. Devices, Circuits and Applications*. 4th edition. Pearson, 2014. 1021 pp. ISBN: 978-0-273-76908-8.

[8] Sarin MV. *Comparitive analysis of induction Motor drive with vector control*. India: International Conference on New Horizons in Science Engineering Technology, 2018. 12 pp. ISRN: 2456-3307.

[9] *An industry based survey of reliability in power electronic converters*. Vol. 47. 2011, p. 1441.

[10] *Ensuring success in global utility solar PV projects*. Tech. rep. 2014.

[11] A. Kiep D. Graovac M. Purschel. *MOSFET Power Losses Calculation Using the Datasheet Parameters. Application note*. Tech. rep. Version V 1.1. Infineon, Automotive Power. 23 pp.

[12] *OptiMOS 5 Power-Transistor 80V. IPT012N08N5*. Rev. 2.1. 2015.

[13] C. H. Oxley R. H. Hopper. *Improved infrared imaging of Semiconductor Devices*. 2014, pp. 697–702.

[14] OPTIMA Batteries. *OPTIMA® YellowTop S 4,2*. 2005.

[15] Wenxin Peng. *Accurate circuit model for predicting the performance of lead-acid agm batteries*. Las Vegas: University of Nevada, Las Vegas, 2011.

[16] L. W. Yao et al. *Modeling of lithium-ion battery using MATLAB/simulink*. Vienna, Austria, 2013.

[17] Battery University. *Charging Lead Acid Batteries*. URL: https://batteryuniversity.com/learn/article/charging_the_lead_acid_battery.

[18] Texas Instruments. *LAUNCHXL-F28069M Overview. User's Guide*. P. 24.

[19] E. Schaltz. *Electrical Vehicle Design and Modelling. Electric Vehicles Modelling and Simulations*. 1st Edition. Aalborg University, 2011, p. 24. 479 pp. ISBN: 978-953-307-477-1.

[20] T. M. Undeland N. Mohan and W. P. Robbins. *Power Electronics. Converters, Applications and Design*. Third Edition. John Wiley & Sons, Inc., 2003. 792 pp. ISBN: 978-0-471-22693-2.

[21] V. G. Pagrut et al. *Internal model control approach to PI tunning in vector control of induction motor*. Mumbai India: International Journal of Engineering Research & Technology, 5 pp. ISRN: 2278-0181.

[22] M. Imecs, C. Szabó, and J. Incze. *Vector control of the cage induction motor with dual field orientation*. Cluj-Napoca, Romania: 9th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics. 12 pp.

[23] C. L. Phillips and R. D. Harbor. *Feedback Control Systems*. Ed. by M. Horton. 4th. Edition. Tom Robbins. ISBN: 0-13-949090-6.

[24] B. Fortman. *A faster current loop pays off in servo motor control*. Texas Instruments, 2017. 8 pp.

[25] J. M. Parr C. L. Phillips. *Feedback COntrol Systems*. fifth edition.

[26] X. I. Li, J. G. Park, and H. B. Shin. *Comparison and Evaluation of Anti-Windup PI Controllers*. Journal of Power Electronics, 2011.

[27] E. Pettit C. J. Anderson. *The Effects of APU Characteristics on the Design of Hybrid Control Strategies for Hybrid Electrical Vehicles*. Detroit, MI: SAE Paper International Congress, 1995.

[28] A. S. Murthy, D. P. Magee, and D. G. Taylor. *Vehicle Braking Strategies Based on Regenerative Braking Boundaries of Electric Machines*. Dearborn, MI, USA.

[29] Texas Instruments. *Technical Reference Manual. TMS320x2806x Piccolo*. 2017. 1196 pp.

[30] Robert C. Martin. *Clean Code, A Handbook of Agile Software Craftsmanship*. 1. ed. Prentice Hall, 2008.

[31] Philip Koopman. *Lecture notes in Software Development Processes*. Aug. 2018.

[32] S. Heath. *Embedded Systems Design*. Newnes, 2003. 451 pp. ISBN: 0-7506-5546-1.

[33] S. W. Smith. *The scientist and Engineer's Guide to Digital Signal Processing*. 2nd Edition. California Technical Publishing, 1999. 688 pp. ISBN: 0-9660176-7-6.

[34] B. Akin and N. Garg. *Scalar (V/f) Control of 3-Phase Induction Motors*. Application Report. note. Texas Instruments, p. 25.

[35] Texas Instruments. *UCC20520 Isolated Dual-Channel Gate Driver with Single Input*. 2016. 45 pp.

[36] L. Zarri et al. *Control schemes for field weakening of induction machines : A Review*. 2015 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD).

A

Induction machine model

This appendix shows the implementation of the induction machine's electrical and mechanical equations that were derived when implementing the dynamic model in Chapter 2, section 2.1. The model is implemented in the $\alpha\beta$ reference frame and in the *Simulink* model a stands for α and b stands for β. The flux linkages are denoted as $flux_x$ with $x = ar, br$.

A.1 Electrical model

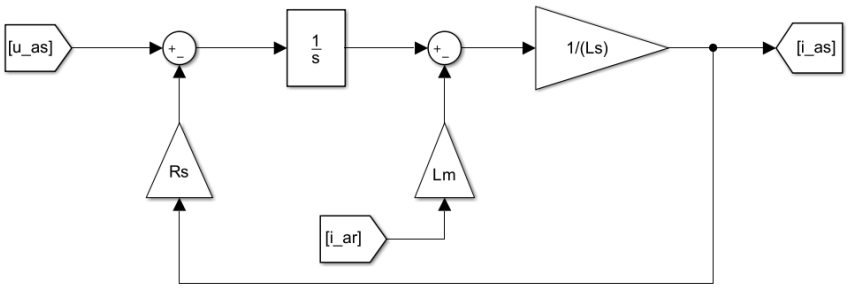


Figure A.1: Simulink implementation of stator voltage equation in α axis.

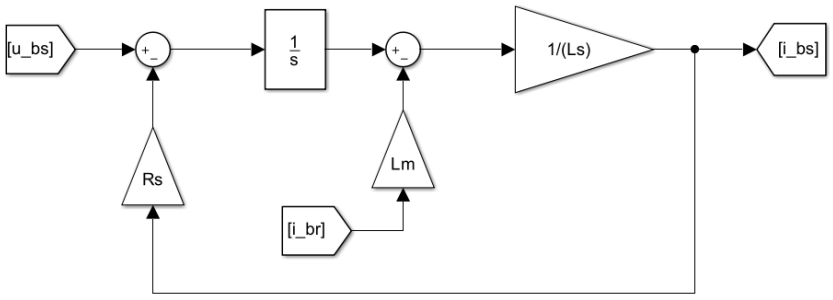


Figure A.2: Simulink implementation of stator voltage equation in β axis.

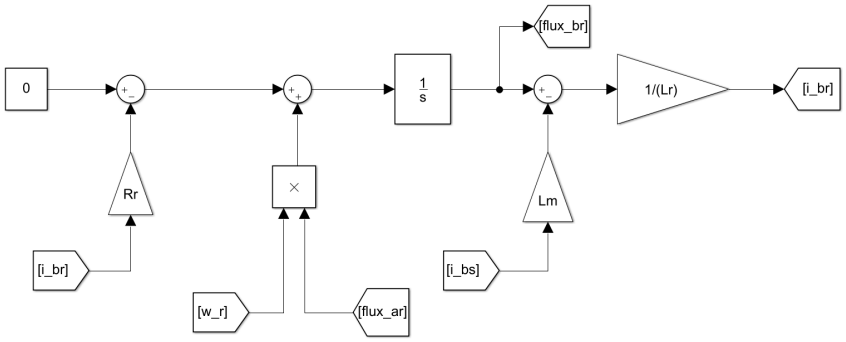


Figure A.3: Simulink implementation of rotor voltage equation in α axis.

there's a mistake in the picture it should be $flux_{ar}$. Correct it. Stef

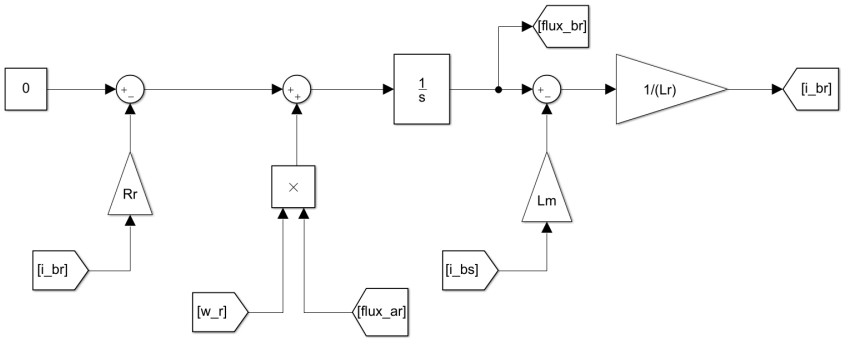


Figure A.4: Simulink implementation of rotor voltage equation in β axis.

A.2 Mechanical model

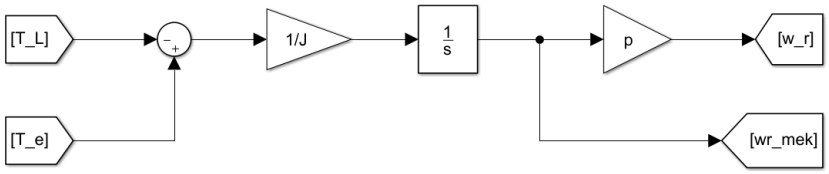


Figure A.5: Simulink implementation of the induction machine mechanical equation.

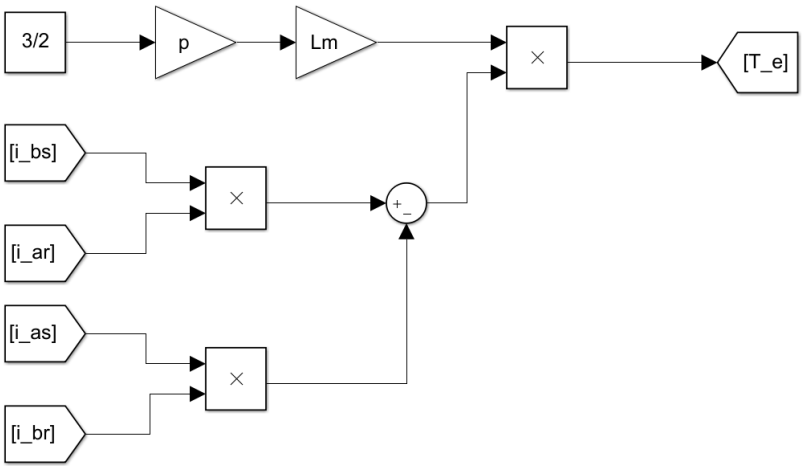


Figure A.6: Simulink implementation of the electromagnetic torque.

B

Batteries

Figures B.1, B.2 and B.3 have been added here in order not to overload the report. Though, the results are presented in the report in the Table 2.4.

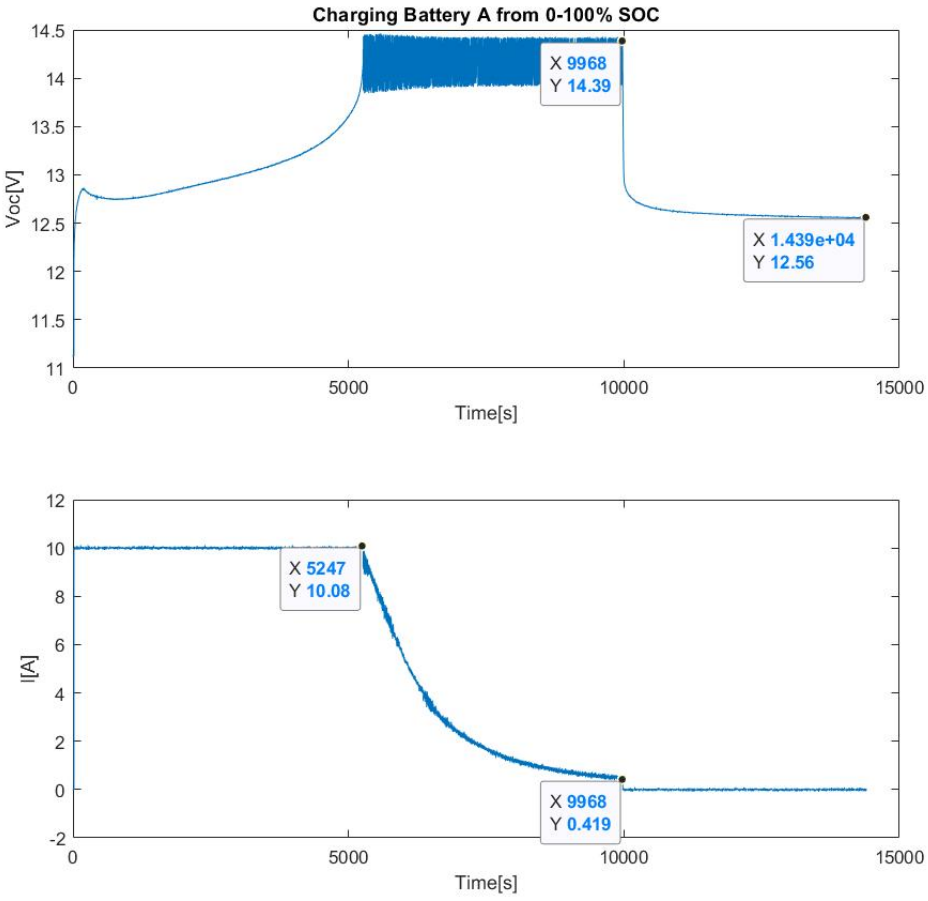


Figure B.1: Curves obtained from charging battery A from 0-100% SOC.

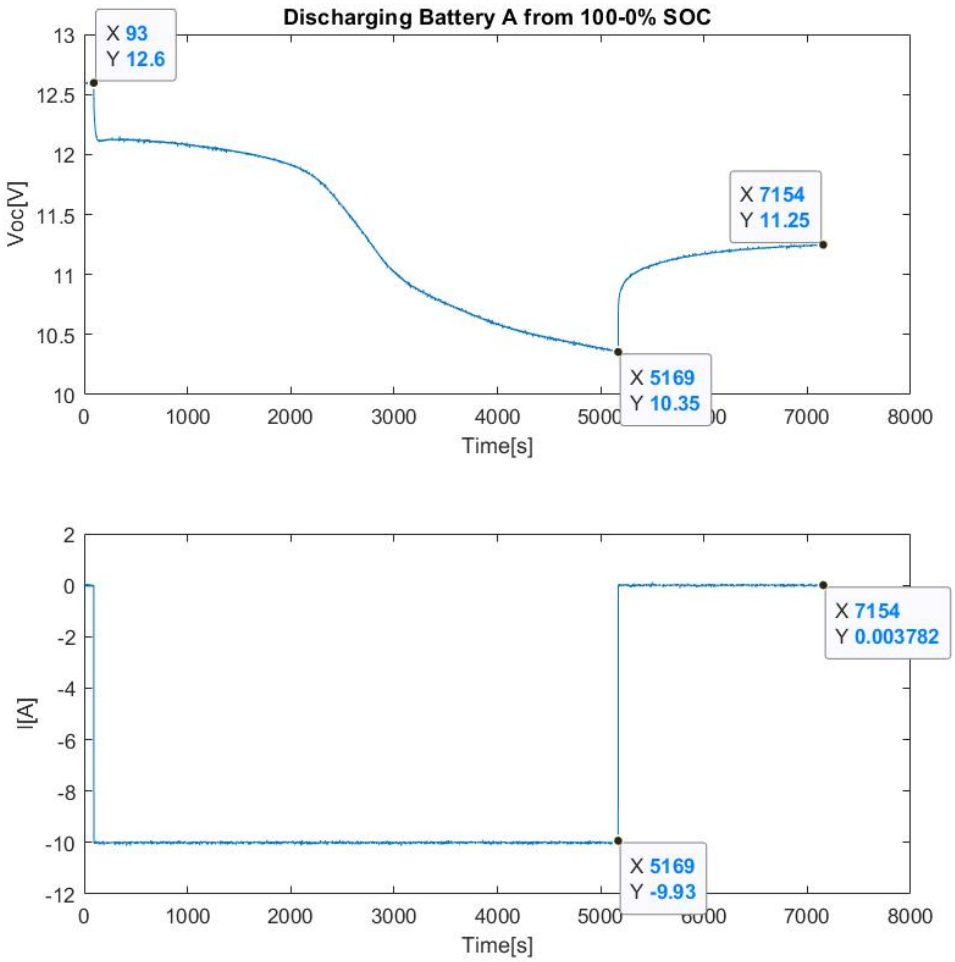


Figure B.2: Curves obtained from discharging battery A from 100-0% SOC.

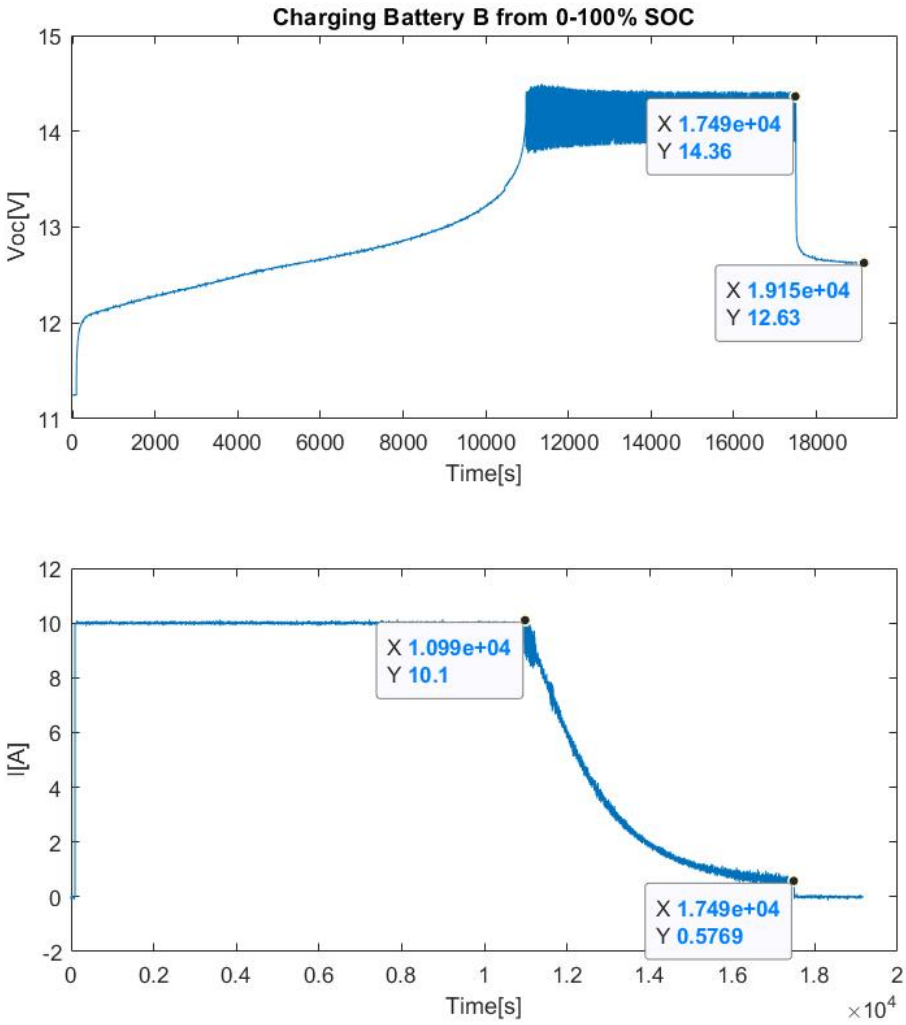


Figure B.3: Curves obtained from charging battery B from 0-100% SOC.

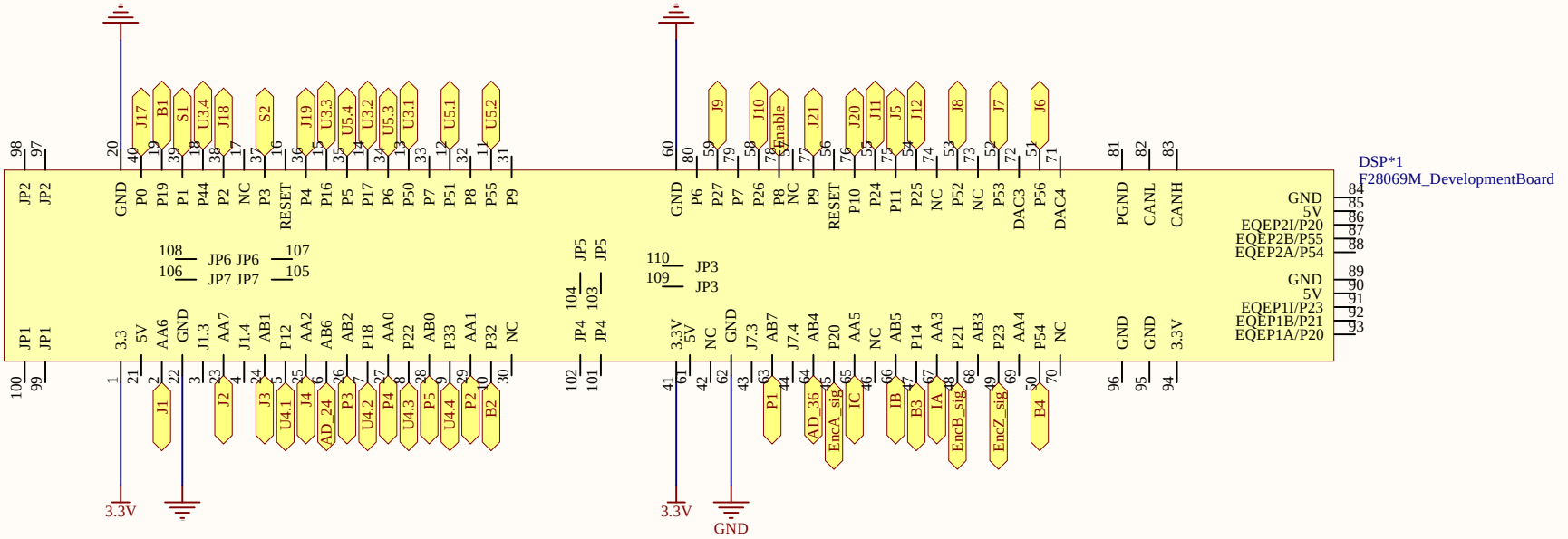
w

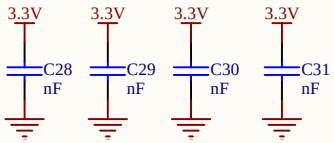
if time permits the capacity of discharging battery B needs to be obtained

C

Interface board

Launchpad



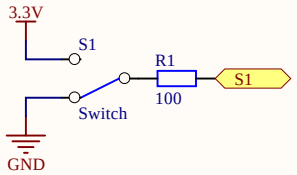


Title		
Size A4	Number	Revision
Date: 16/05/2019		Sheet of
File: C:\Users\..\DSP.SchDoc		Drawn By:

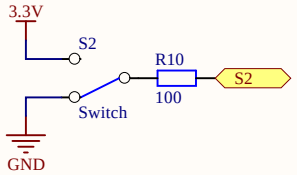
Inputs

Switches

Move Enable

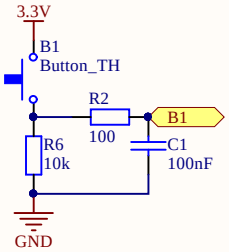


CL/OL

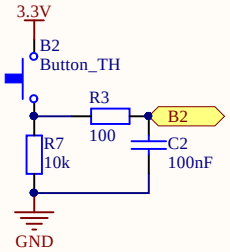


Buttons

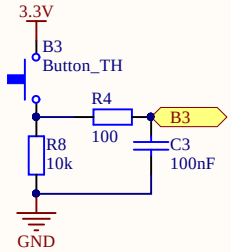
Torque ref



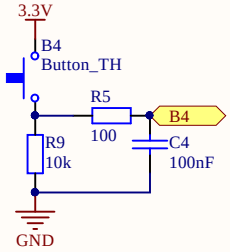
Regenerative braking



Cruise control

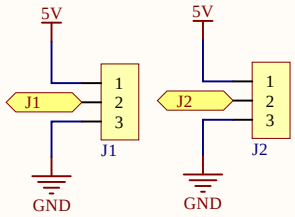


Wheel anti-slip

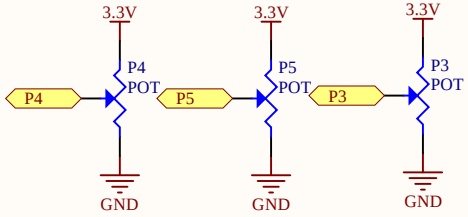


Potentiometers

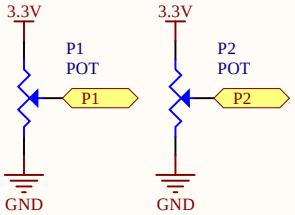
Termometers



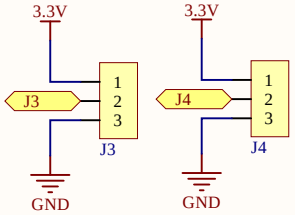
Rotatory



Sliders



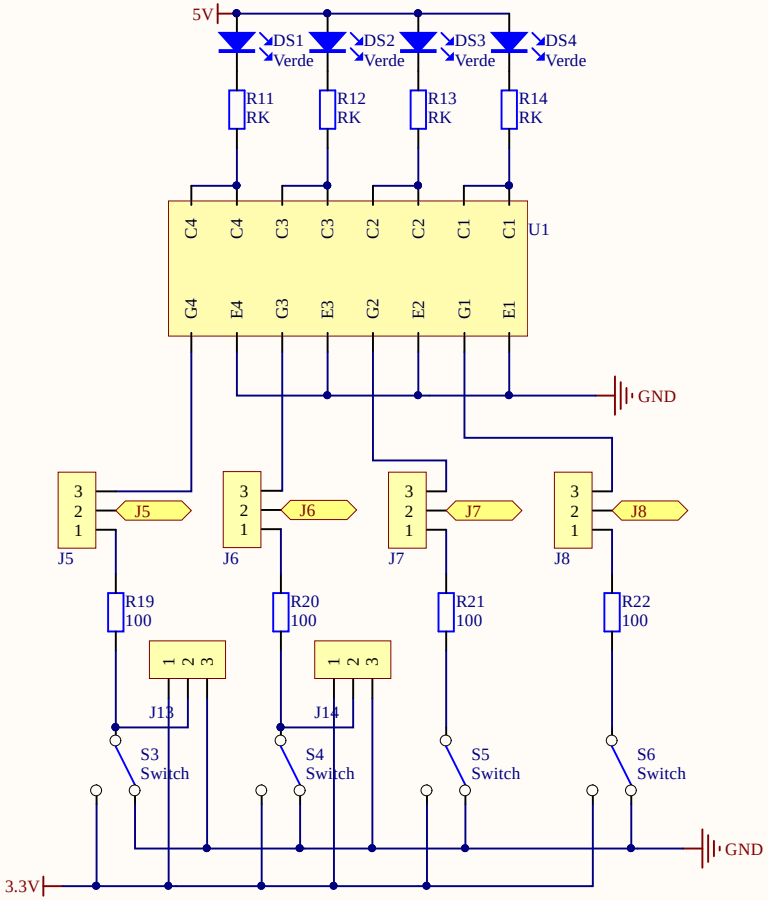
Connectors



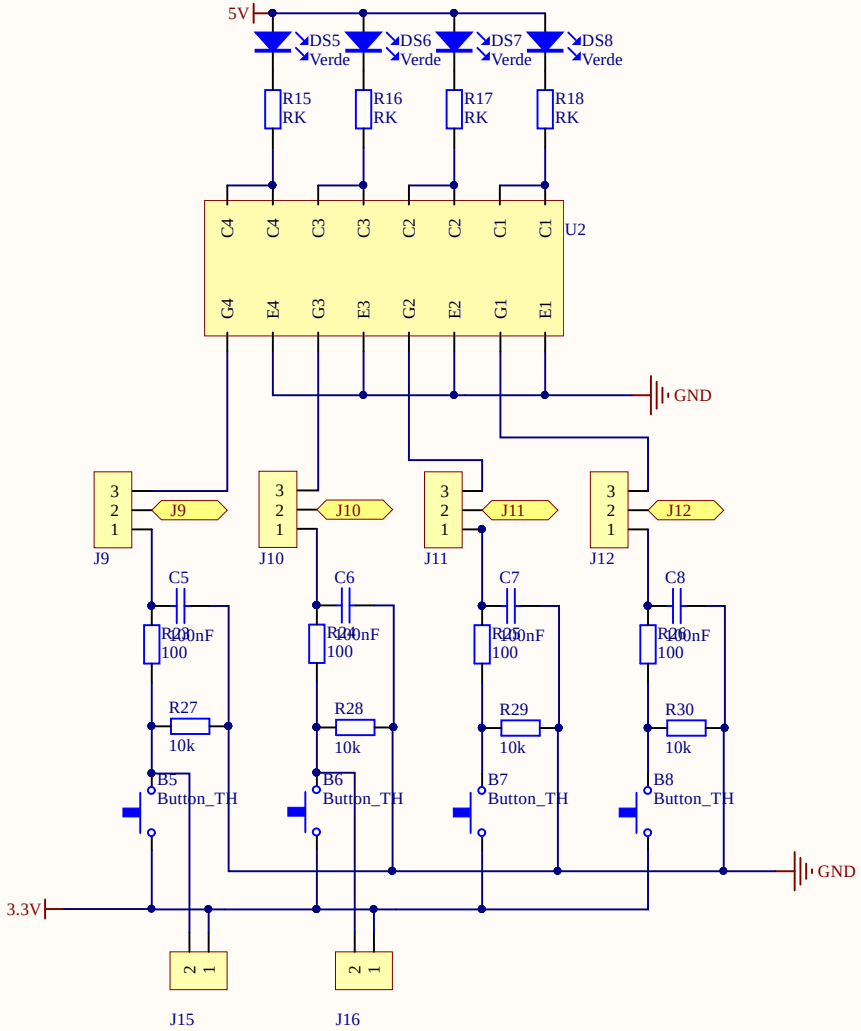
Title		
Size: A4	Number	Revision
Date: 16/05/2019		Sheet of
File: C:\Users\..\Inputs.SchDoc		Drawn By:

Extras

Switches and LEDs



Buttons and LEDs

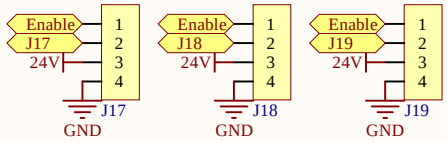


Title		
Size: A4	Number	Revision
Date: 16/05/2019		Sheet of
File: C:\Users\..\InputsExtras.SchDoc		Drawn By:

Outputs

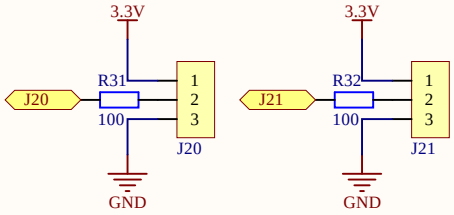
To inverter

PWM signals

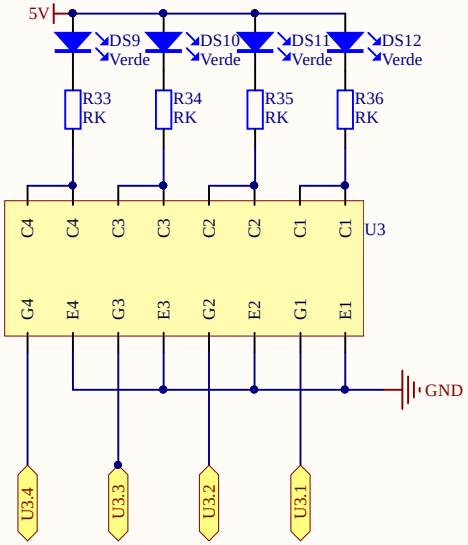


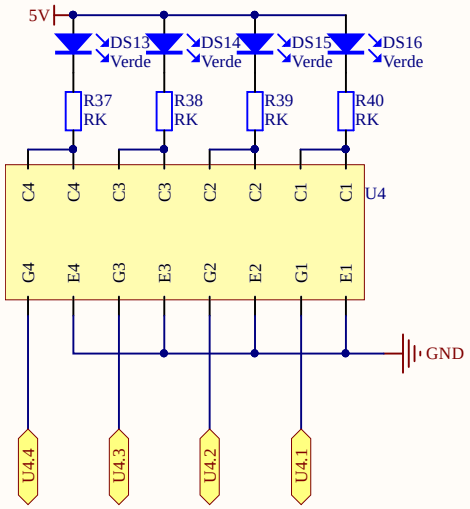
Extras

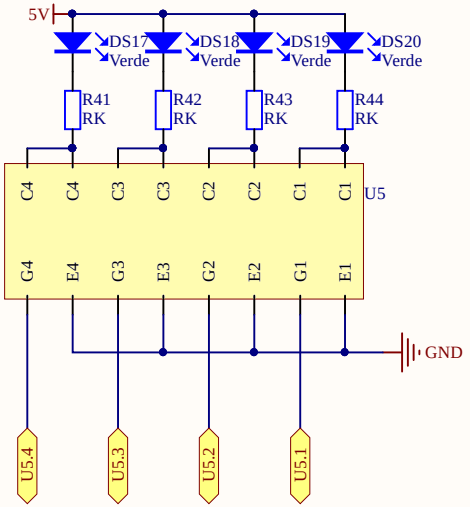
Connectors (also valid as inputs)

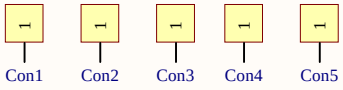


LEDs



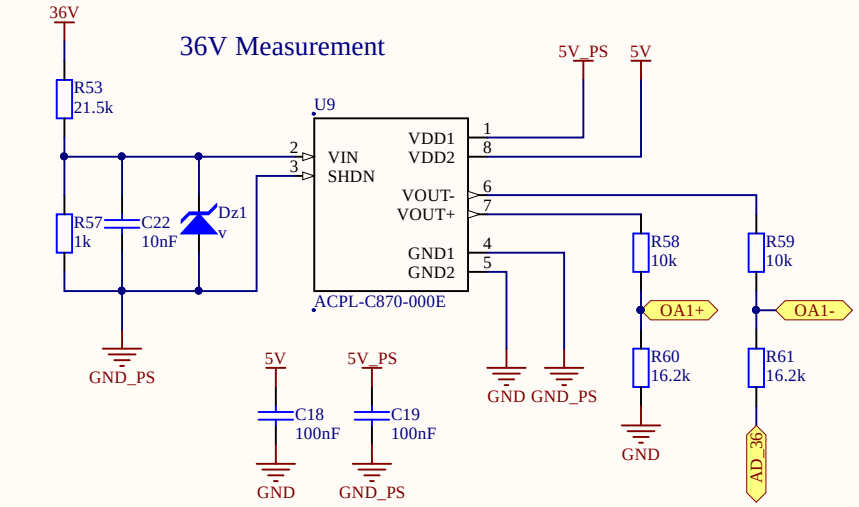


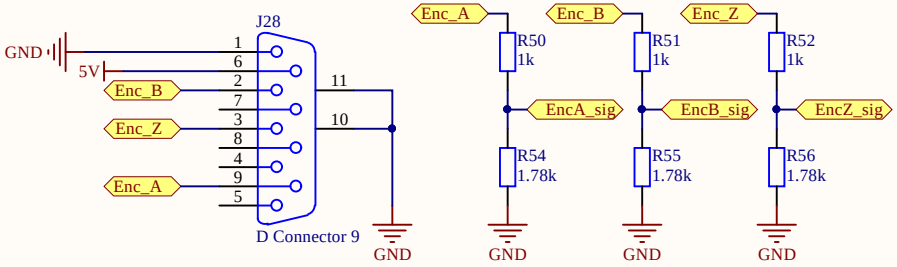


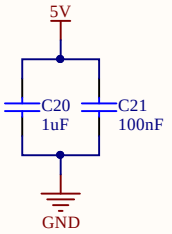


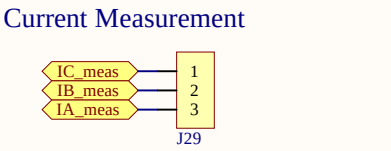
Title		
Size: A4	Number	Revision
Date: 16/05/2019		Sheet of
File: C:\Users\..\Outputs.SchDoc		Drawn By:

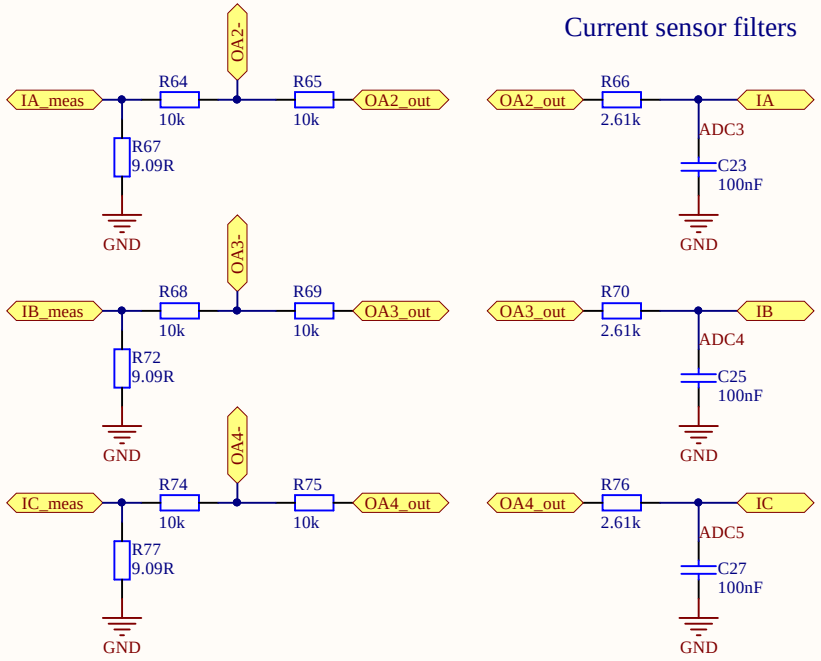
Signal processing

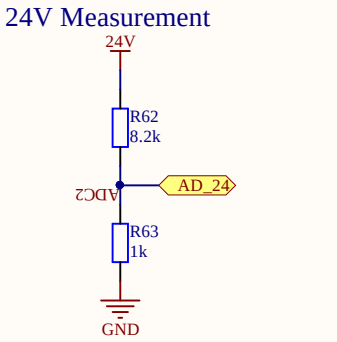


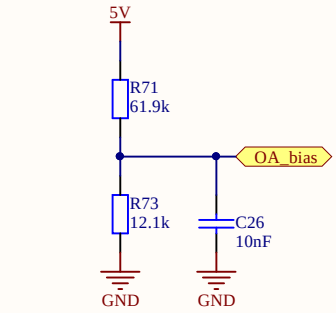


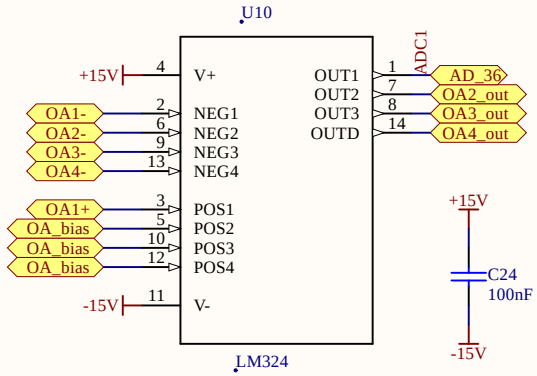








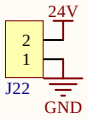




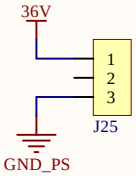
Title		
Size A4	Number	Revision
Date: 16/05/2019		Sheet of
File: C:\Users\..\SignalProcessing.SchDoc		Drawn By:

Supplies

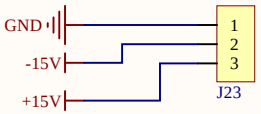
24V input



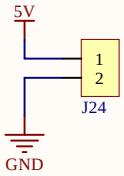
36V input



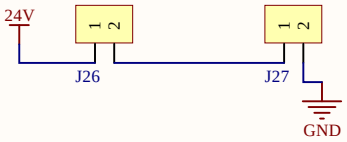
Current sensor supply

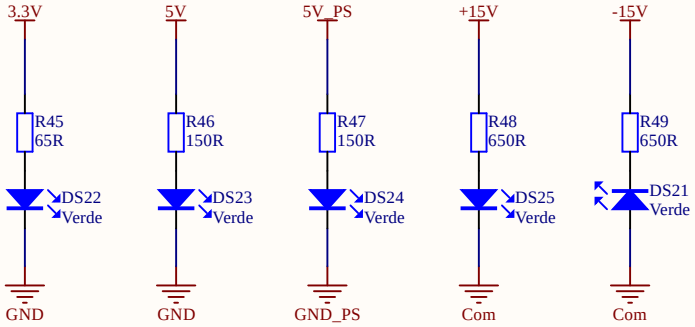


5V DSP supply

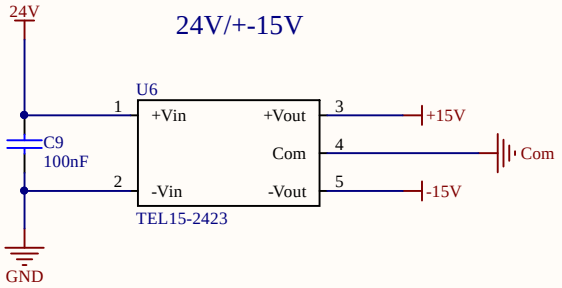


Fan supply

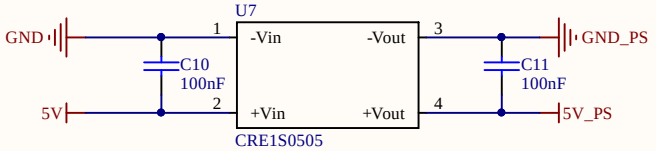




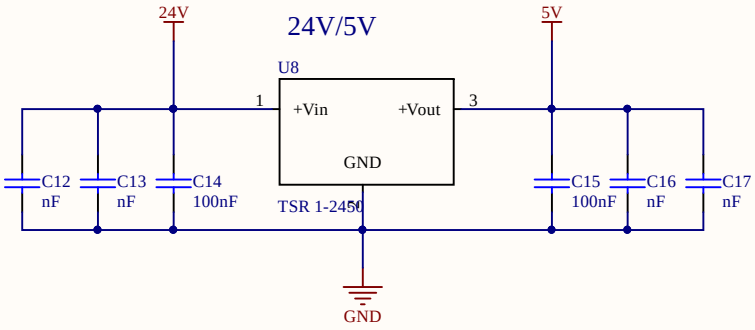
24V/+-15V



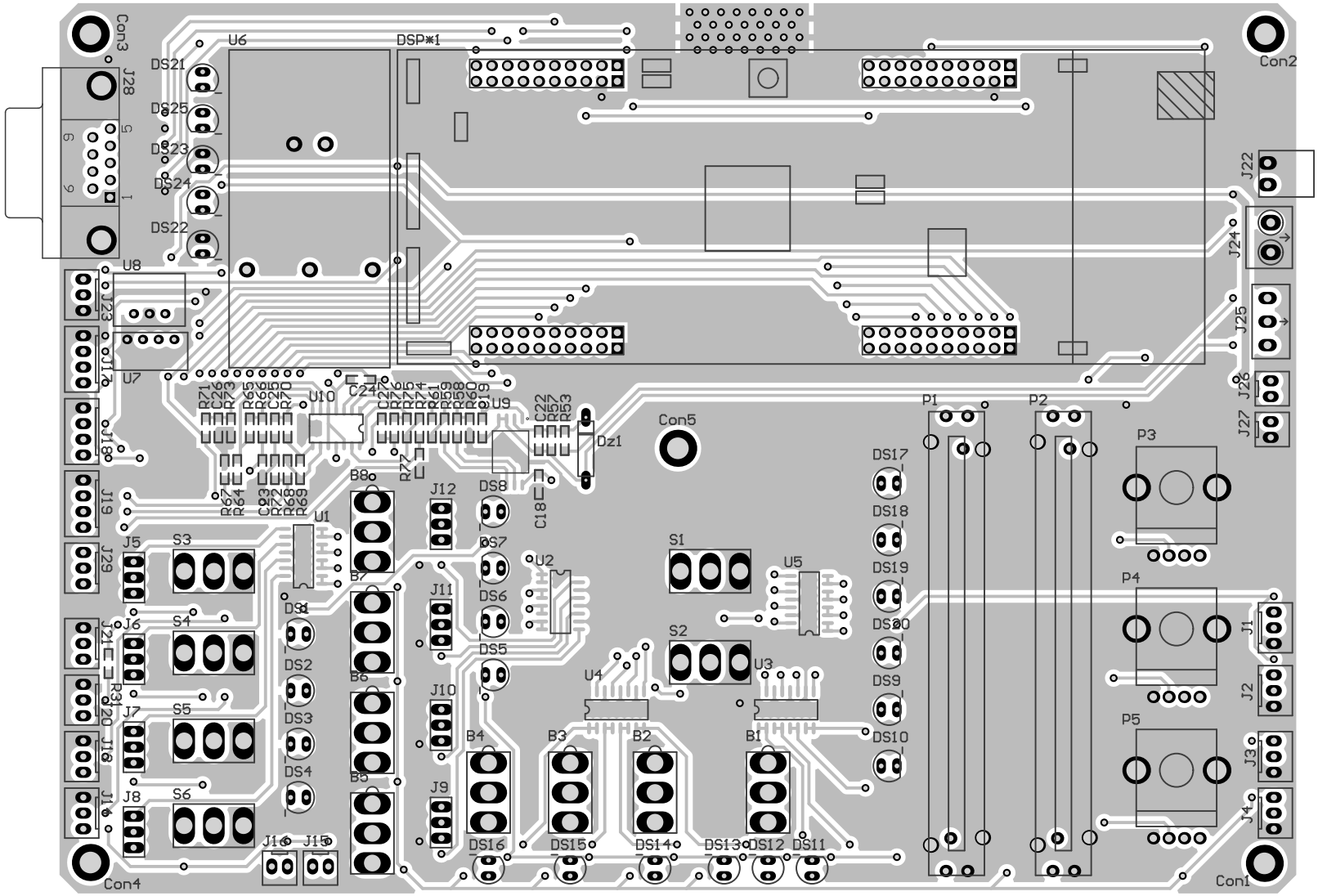
5V/5V isolation



24V/5V



Title		
Size: A4	Number	Revision
Date: 16/05/2019		Sheet of
File: C:\Users\..\Supplies.SchDoc		Drawn By:


Con3
U6
DSP*1
Con2
DS21
DS25
DS23
DS24
DS22
J28
J22
J24
J25
J27
J26
U8
J23
U7
J17
J18
J19
J29
J5
S3
J6
S4
J21
R31
J20
J7
S5
J18
J8
S6
J16
J14
J15
Con4
U10
C24
U9
C22
R57
R53
Dz1
Con5
R27
C18
B8
J12
DS8
U1
DS7
U2
S1
U5
J11
DS6
DS1
DS17
DS18
DS19
DS20
DS9
DS10
DS2
DS5
S2
U3
U4
DS3
J10
B4
B3
B2
B1
DS4
J9
DS16
DS15
DS14
DS13
DS12
DS11
P1
P2
P3
P4
P5
J1
J2
J3
J4
Con1