



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



UNIVERSITY OF PADUA

DEPARTMENT OF INFORMATION ENGINEERING

MASTER THESIS IN CONTROL SYSTEM ENGINEERING

MODELLING AND FIRMWARE CONTROL IMPLEMENTATION OF A BRUSHLESS MOTOR FOR GATE DRIVES

SUPERVISOR

PROF. AUGUSTO FERRANTE
UNIVERSITY OF PADUA

CO-SUPERVISOR

FEDERICO PASQUETTO

MASTER CANDIDATE

GIULIO SAVIAN

ACADEMIC YEAR

2021-2022

TO MY FAMILY.

Abstract

The history of automatic gate drives changed when the adoption of electromechanical automations have substituted the less efficient and not standardizable oleodynamic drives. Another change is underway right now. Based on their high efficiency and density power brushless motors constitute a good alternative to the older brushed or asynchronous motors. The adoption of permanent magnets synchronous motor (PMSM) is increasing and many industries are implementing them in medium applications, such as gate drives. Despite their high efficiency they are often controlled with open-loops or with scalar control methods that do not take full advantage of the technology. In order to exploit the motor power density a vector control technique must be considered for produce the maximum torque at any instant. The field oriented control (FOC) is then implemented in firmware and its application has been assisted with model base methods and the motor control library of STMicroelectronics.

Contents

ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xiii
LISTING OF ACRONYMS	xv
ACKNOWLEDGMENTS	i
1 INTRODUCTION	3
1.1 Came s.p.a.	3
1.2 Thesis outline	4
2 SPM MOTOR AND CONTROL STRATEGIES	7
2.1 SPM introduction	7
2.2 Dynamical model	10
2.2.1 SPM limits and operating regions	15
2.2.2 Clarke and Park transformations	18
2.3 SPM control techniques	19
2.3.1 Field oriented control	20
2.3.2 Speed loop control methods	23
2.3.3 Model based methods for sensorless techniques	26
2.4 Estimation of parameters	31
2.4.1 Stator resistance estimation and inverter non-ideality	33
2.4.2 Hall offset estimation	35
2.4.3 Flux linkage estimation	38
3 THE OVERALL SYSTEM	41
3.1 System components	41
3.2 Electric parts	42
3.2.1 Control unit	43
3.2.2 Motor drive unit	44
3.3 Mechanical parts	52
4 FIRMWARE ARCHITECTURE	57

4.1	Code structure	57
4.2	Motor application	59
4.2.1	State machine	60
4.2.2	Speed loop function	61
4.3	Board support package	63
4.3.1	Peripherals	64
4.3.2	FOC algorithm	67
4.3.3	SVPWM code	68
4.3.4	Hall sensor management	72
4.3.5	Sensorless algorithm	73
5	SIMULATION AND DESIGN OF THE CONTROL SYSTEM	77
5.1	Model of the overall system	78
5.1.1	Simulink model of the electric motor	80
5.1.2	Simulink model of the inverter	87
5.2	Field oriented control design	90
5.2.1	Current control loop	93
5.2.2	Speed control loop	98
5.2.3	Feedback chain	102
5.2.4	Sensorless control design	104
5.2.5	Back-EMF Luenberger Observer $\alpha\text{-}\beta$ and $d\text{-}q$	107
5.3	Sensorless and sensed simulation results	109
6	EXPERIMENTAL RESULTS	111
6.1	Static torque verification	111
6.2	Firmware test results	113
7	CONCLUSIONS AND PROBLEMS	119
7.1	Future works	122
	REFERENCES	125

Listing of figures

2.1	Costs of different types of motors based on the materials used	8
2.2	Cross section of a SPM and a IPM motor	9
2.3	Cross section of a two pole pair SPM motor with $a-b-c$, $\alpha-\beta$ and $d-q$ reference frame	10
2.4	Block diagram of a SPM motor in $d-q$ reference frame	15
2.5	Functional limits of SPM and power curves	17
2.6	Current signal with different reference frames	18
2.7	Sensored FOC control block scheme with FLC for velocity loop	22
2.8	Close loop speed control scheme without mechanical block	23
2.9	Fuzzy logic block scheme	25
2.10	Luenberger observer block scheme	29
2.11	Classic PLL block scheme in discrete time	31
2.12	Performance and characteristic curves of the BLDC motor	32
2.13	Equivalent resistance with different voltage references in d-axis	34
2.14	V-I plot with different voltage references in d-axis	35
2.15	Hall signals and back EMF for each phase	36
2.16	Frequency analysis of the back-EMF signals	37
2.17	Hall offset estimation in clockwise and anticlockwise	38
2.18	Flux linkage estimation in clockwise and anticlockwise	39
3.1	Image of the hardware system used	42
3.2	Image of a STM32F401RE control board	43
3.3	Image of the IHM08M1 motor drive board	45
3.4	Inverter schematics	45
3.5	Space vector hexagon	47
3.6	Switching patterns and corresponding switching time of sector 1	48
3.7	Current sensing networks and comparison	49
3.8	Noise parameters definitions, signal of a single shunt configuration	50
3.9	Current sensing networks in low-side three shunt scheme with operational amplifier for adapt the signal to the ADC resolution	51
3.10	Low side of phase A, B, C duty cycle $> DT + \max(TN, TR)$	52
3.11	Exploded view drawing of the CAME ATS solution	53
3.12	Gearbox drawing of the CAME ATS device	53
3.13	Experimental set-up	55

4.1	Simplified FDK simplified architecture of the ATS project	59
4.2	Motor state machine	61
4.3	Synchronization strategy between TIMER 1 PWM output and ADC consid- ering the rising edge	65
5.1	Simulation of the whole control system	79
5.2	Fluxes map for I_q	80
5.3	Two motor models implemented in Simulink: a standard one and a model that can comprise non linearity effects	81
5.4	Standard $d-q$ PMSM motor model implementation in Simulink	82
5.5	Electrical block of the $d-q$ PMSM model including fluxe maps	83
5.6	Mechanical block of the $d-q$ PMSM model	83
5.7	Test results conducted for the PMSM motor model validation with reference in V_d	85
5.8	Mechanical Speed test results conducted for the PMSM motor model valida- tion with reference in V_q	86
5.9	Current test results conducted for the PMSM motor model validation with reference in V_q	87
5.10	Inverter block implemented in Simulink	88
5.11	SVM algorithm	88
5.12	Inverter power driver	89
5.13	Test results conducted for the inverter model validation	90
5.14	Current loop in the FOC algorithm simulation and inverter block	91
5.15	Bode diagram of the plant in open-loop	92
5.16	Current control loop for a SPM motor in the direct and quadrature axes	93
5.17	Control system block diagram with decoupling mechanism	94
5.18	Block diagram current loop in the q axis	95
5.19	Comparison between real and simulated current loop step response with quadra- ture current reference $I_q = 500\text{ mA}$	97
5.20	Code of the current process in open and closed loop	97
5.21	Simulation of a PI controller with anti windup mechanism	98
5.22	Simulation of the Fuzzy PI controller	99
5.23	block diagram of the speed control loop	100
5.24	Comparison between real and simulated current loop step response with quadra- ture current reference $I_q = 500\text{ mA}$	101
5.25	Speed step response comparison between classic PID and fuzzy PID	102
5.26	Implementation of the feedback chain in the simulation environment	103
5.27	Hall sensor signals and sector while motor is running	104
5.28	Sensorless block scheme for $\alpha-\beta$ or $d-q$ reference frame	105
5.29	Implementation of the sensorless speed and angle estimation in the stationary reference frame $\alpha-\beta$	106

5.30	Implementation of the sensorless speed and angle estimation in the rotating reference frame $d-q$	107
5.31	Speed step response and estimated angle error for the LO implemented in the $\alpha-\beta$ reference	108
5.32	Speed step response and estimated angle error for the LO implemented in the rotating reference frame	108
5.33	Speed step response with sensed feedback chain tuned for have the best result	109
6.1	Test and datasheet curves of the start-up torque	113
6.2	Speed and current step response without load	114
6.3	Speed and current step response with load	114
6.4	Speed and current response with growing reference without load	115
6.5	Speed and current response with growing reference with load	115
6.6	Zoom of the speed and current step response in no load conditions	116
6.7	Zoom of the speed and current step response in no load conditions in the slowdown phase	117

Listing of tables

2.1	Sensorless solutions	26
2.2	SPM linear model in state space representation	28
2.3	BLDC motor datasheet	32
2.4	Measurements of the electrical parameters	33
2.5	Resistance values with different methods	35
2.6	Flux linkage values	40
3.1	Voltage vectors for two-Level VSI	46
4.1	Motor state machine	60
4.2	Peripherals used in the project	64
4.3	ADC Configuration	65
4.4	Timer 3 configuration	66
4.5	Timer 1 configuration	67
4.6	Sector identification	69
6.1	Tabulated data from the starting torque test	112

Listing of acronyms

FOC	Field Oriented Control
FDK	Firmware Development Kit
PMSM	Permanent Magnet Synchronous Motor
SPM	Surface Permanent Magnet
IPM	Interior Permanent Magnet
BLDC	Brushless Direct Current
EMF	Electro Motive Force
MTPA	Maximum Torque Per Ampere
MTPV	Maximum Torque Per Volt
FLC	Fuzzy Logic Controllers
FL	Fuzzy Logic
LO	Luenberger Observer
PLL	Phase Locked Loop
ICS	Insulated Current Sensor
PWM	Pulse Width Modulation
SPWM	Sinusoidal Pulse Width Modulation
SVPWM	Space Vector Pulse Width Modulation
ADC	Analog to Digital Converter
SDK	Software Development Kit
MCSDK	Motor Control Software Development Kit
RTOS	Real-Time Operating System

BSP Board Support Package
IRQ Interrupt Request Handler
MBD Model Based Design

Acknowledgments

I would like to sincerely regard all the people who contributed to this achievement.

First of all to my family who encouraged and supported me in making this choice. Their faith has been fundamental to face this arduous path.

A deep appreciation to my company tutor Federico Pasquetto and my academic tutor Augusto Ferrante who guided me in the drafting of the thesis and in the success of the project.

I'm also grateful to the entire CAME s.p.a. for the opportunity and the tools. Here I was able to test myself.

I'd like to mention my colleagues: Diego, Gianluca, Andrea and Riccardo for the help and for the great welcome into the workplace.

Lastly, to my girlfriend Giulia and my friends who have managed with patience to distract and to relieve me in times of difficulty.

Thank you all.

1

Introduction

1.1 CAME S.P.A.

The history of CAME Group begins with the adventure of the entrepreneurs Paolo and Angelo Menuzzo, who in the 1972 decided to found the start-up Costruzioni Meccaniche.

C.M. operated in the growing automation sector and has now become the well-known CAME Group. In 2017, it had a turnover of around 255 millions of euros and has 1460 employees.

CAME is a international company that has expanded its portfolio and provides technological solutions for residential, public and urban environments. The group develops: automation for entrances, home automation, anti-intrusion systems, video door entry technologies, thermoregulation, garage doors, sectional doors, solutions for urban planning, systems for the management of automatic car parks, paid parking meters, access control and safety of collective environments. All of these products are marketed across the lines CAME, CAME Bpt, CAME Go, CAME Urbaco and CAME Parkare.

Although the group is a multinational company, it is closely linked to Italy and is present in the market with 26 branches and 480 partners worldwide. The site is located in Dosson of Casier in Treviso and has a total of six factories: in Treviso but also in Sesto al Reghena (PN), Spilimbergo (PN), Avignon (France), Barcelona (Spain) and London (UK).

The evolution of the automation sector and the company's ability to make the right decisions represent two of the many reasons that build its success.

This evolution can be found in all the main technological components that make up the products: handling systems, coupling systems, control systems and safety systems. Particularly interesting for the thesis is the evolution of handling systems. In this field exist two main technologies: oleodynamics and electromeccanics.

The oleodynamics is the first technology apply to the sector but due to the density variations of the liquids in the presence of different environmental conditions it has functional limitation. For this purpose, the electromeccanical, initially less reliable, was able to replace the oleodynamic drives.

The company takes advantage of these changes and has been able to implement them in the motor redesign and standardization [1].



1.2 THESIS OUTLINE

The goal of the thesis is to built a functional field oriented control (FOC) algorithm in firmware for the activation of a gate. This has been possible starting from the CAME firmware development kit (FDK) and by the STMicroelectronics motor control software.

The firmware architecture used is the one of CAME s.p.a., which has been modified and expanded for the project, incorporating various functions of the motor control software development kit (MCSDK).

Using a firmware architecture allows to help embedded system designers quickly and easily test firmware for their specific device. It promotes the standardization allowing to change part of the code in a simple and fast way with different electronic boards, determining a unique code. Produced a functional firmware the system is studied and simulated in order to use model based design technique in the control and for make a simulation comparison with sensorless algorithms. Then, the sensed code is implemented in practise, critically comparing the results with the simulation.

The project is organized in a total of seven chapters where each of there can embody several subsection.

- **Chapter 1:** A brief description of the company who hosted me and the thesis objectives and outline are reported.
- **Chapter 2:** All the theory needed to replicate the project are analyzed, focusing on the SPM motors and its control. The chapter also presents a second part containing some experiments to estimate the indispensable parameters for the project.
- **Chapter 3:** The chapter gives a description of the overall system and the devices used for the development. The system is divided into two different parts: an electric subsystem and a mechanical one.
- **Chapter 4:** Deepening the aspects related to FDK and the event-based systems. Moreover, are reported all the main algorithms implemented in the micro-controller to have a functional and efficient system.
- **Chapter 5:** The system obtained is modelled and simulated for design its FOC control. First some experiments are carried out to check the correctness of the plant model. Then, a sensorless control method based on Luenberger Observer is simulated to compare it with the sensed simulation.
- **Chapter 6:** The system obtained is modified accordingly to the control designed of chapter 5 and it is implemented in practice. Here are shown the results of the gate drive control.
- **Chapter 7:** A summary of the results and problems obtained is made, followed by some suggestions for possible future developments.

2

SPM Motor and Control Strategies

The organization of the chapter is divided into two parts strictly related to the engine used in the project.

The first part reports the theory adopted for the development of the thesis and for motivate the design choice. Therefore, a review of the three-phase motor with surface permanent magnet (SPM) is presented. The analysis is conducted by introducing the engine structure, the adopted mathematical model and finally a discussion on the control strategy.

The second part of the chapter deals with some experiments carried out for the estimation of the motor parameters, necessary for the simulation.

2.1 SPM INTRODUCTION

The motor made available by the company is a synchronous motor with surface permanent magnets, these devices constitute one of the main two branches that make up the permanent magnet synchronous motors (PMSM), i.e. they have permanent magnets instead of excitation winding in the rotor. The other branch includes the interior permanent magnet (IPM) motors, which will not be covered.

The success of PMSM is due to the incremental improvement of magnetic materials which offers high torque/volume rate and flexibility in the shape design. In particular, if the length/-diameter ratio of the motor is high it is used for high speed and low inertia, vice versa it is used for low speed and high torque [2], [3].

These motors are often employed in industry scenarios where high performance drives are required, however, the good performance are motivated by a high cost of the magnets. The magnets are usually made up of NdFeB, exploiting the neodymium (Nd), a rare earth material whose extraction is not sustainable for the environment. The costs are shown in Figure 2.1 [4].

PMSM motors are a good alternative to previously used induction motors, as they increasing the system efficiency. The advantages in choosing the PMSM machines are different, but the main one can be summarized in four points:

- Less volume/power ratio, they have an high density power;
- No joule losses in the excitation winding;
- Better heat dissipated;
- More reliable.

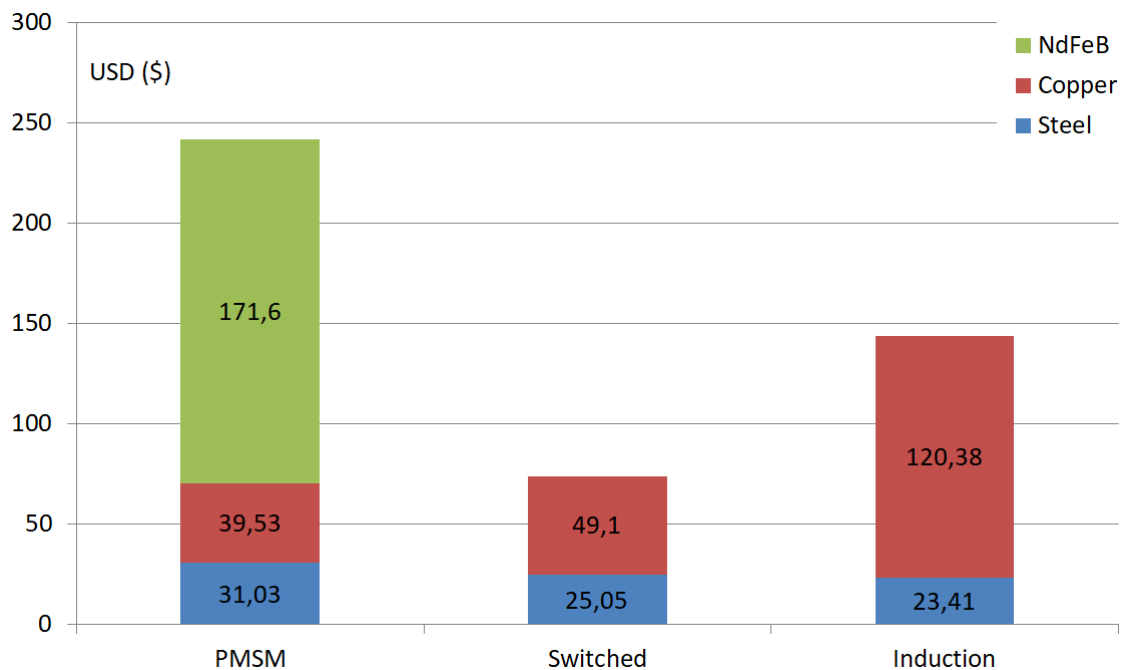


Figure 2.1: Costs of different types of motors based on the materials used

Explained the main advantages and motivated the choice of a PMSM motor for the project, now can be analyzed how the mechanical motion is generated and how is constituted a SPM

motor.

The mechanical motion is given by the interaction between conductors crossed by currents and the magnetic fields created by the permanent magnets. The conductors are located in the fixed stator, while the magnets in the movable rotor.

Rotor and stator are made of ferromagnetic material laminated and separated by an air gap. Magnets have a magnetic permeability similar to air, taking advantage of this fact, isotropic or anisotropic rotors can be formed, which characterize the SPM and IPM motors respectively.

The stator winding has three equal phases out of phase by $\frac{2\pi}{3}$ connected via terminals to the output of the power electronics.

These machines are brushless implying the adoption of an inverter for the polarity electronic switching. There exists two types of power supply: with quasi-square current, also called trapezoidal brushless or directly brushless DC, and sinusoidal brushless or brushless AC.

The division can also be distinguished through the distribution of the winding, often the DC-brushless are characterized by having concentrated winding that produce a trapezoidal EMF, while AC-brushless usually have winding distributed in several layers and a shortened pitch, reducing harmonic effects and so obtaining a sinusoidal EMF.

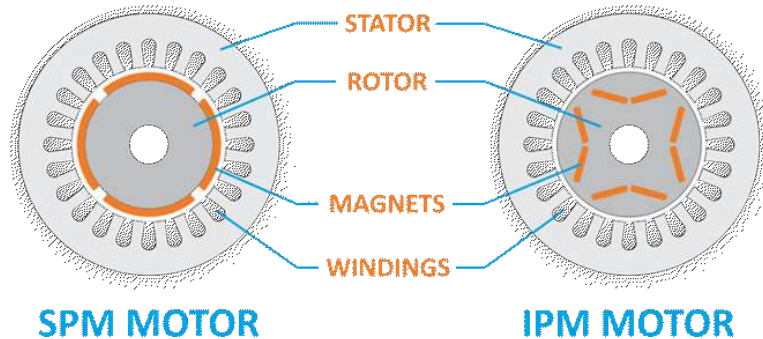


Figure 2.2: Cross section of a SPM and a IPM motor

The engine under analysis is a concentrated winding brushless AC motor, but with a sinusoidal Back-EMF. This is possible by adapting the rotor structure to increase the efficiency or by changing the pole tips so that the induction distribution curve is close to the sine wave. The consequence is that the engine will be treated like an SPM machine.

This choice is not random but concerns the costs, in practice a distributed configuration uses more copper, is more difficult to produce and is larger in size, taking up space in the plant to

be sold.

The motor can be powered using either of the techniques mentioned above, but the trapezoidal, also called six steps, will be overlooked in the discussion. Sinusoidal switching has better performance allowing to have a low torque ripple, smooth motion and maximum torque, as well as the possibility of implementing complex and high performance algorithms.

2.2 DYNAMICAL MODEL

Before starting with the analysis of the model that delves into the theory of reference frame transformation is better to open a little parenthesis by mentioning the angles.

Figure 2.3 shows the three references used for the construction of the dynamic model. Starting from the three-phase stator winding represented by the a - b - c reference frame, the studies will move to the fixed bi-axial reference α - β , for conclude with the mobile reference d - q . These transformations concerns with the computationally complexity and are used to facilitate the management of the physical quantities.

To perform the last transformation it is necessary to know the phase shift angle between stator and rotor. This angle is called the electromechanical angle θ_{me} and it depends on the number of poles pair of the motor and on the mechanical angle by the formula: $\theta_{me} = p \cdot \theta_m$.

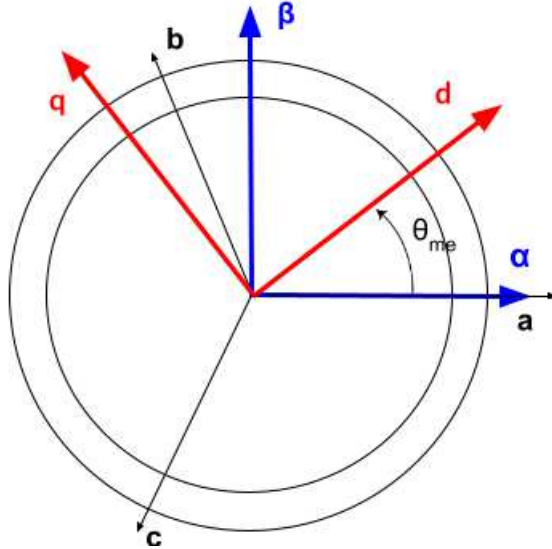


Figure 2.3: Cross section of a two pole pair SPM motor with a - b - c , α - β and d - q reference frame

Known the angle notation and the basis of the reference frame transformation, studied deeply in subsection 2.2.2, the motor fundamental equation and model can be tackled.

Suppose to do not force a voltage into the stator winding, so only the contribution of the permanent magnets will be present. Remember that a sinusoidal brushless motor is treated by giving a sinusoidal EMF which implies a sinusoidal concatenated flux. This is the only difference with respect to the BLDC motor model.

In Equation 2.1 the flux expressions are written explicitly for each phase as a function of the electrical angle θ_{me} .

$$\begin{cases} \lambda_{a,mg} = \Lambda_{mg} \cos(\theta_{me}) \\ \lambda_{b,mg} = \Lambda_{mg} \cos(\theta_{me} - 2\pi/3) \\ \lambda_{c,mg} = \Lambda_{mg} \cos(\theta_{me} + 2\pi/3) \end{cases} \quad (2.1)$$

where Λ_{mg} is the maximum flux linkage of the permanent magnet.

Considering only the contribution of the flux due to the stator current for each phase and with the hypothesis of motor built with ferromagnetic material without eddy currents and without the presence of any hysteresis loop, then its contribution to the flux of each phase is expressed as:

$$\begin{cases} \lambda_{i,a} = L_s \cdot i_a \\ \lambda_{i,b} = L_s \cdot i_b \\ \lambda_{i,c} = L_s \cdot i_c \end{cases} \quad (2.2)$$

where i_a, i_b, i_c are the currents of each phase and $L_s = L_{SS} + |L_{MSS}|$ is the stator inductance; L_{SS} is the self-inductance while L_{MSS} is the mutual inductance. Both are supposed to be equal and symmetrical for each phase.

Using the hypothesis of the linearity of the magnetic circuit, the superposition principle can be used to calculate the simultaneous contribution of the permanent magnet and the stator currents, obtaining from Equation 2.1 and 2.2 the total fluxes:

$$\begin{cases} \lambda_a = \lambda_{a,mg} + \lambda_{i,a} \\ \lambda_b = \lambda_{b,mg} + \lambda_{i,b} \\ \lambda_c = \lambda_{c,mg} + \lambda_{i,c} \end{cases} \quad (2.3)$$

Now that the flux has been visualized, the study can be moved to the general balance equation

of the voltages u_a, u_b, u_c for each phase a, b, c . The equations are reported below:

$$\begin{cases} u_a(t) = R_s i_a(t) + \frac{d\lambda_a(t)}{dt} \\ u_b(t) = R_s i_b(t) + \frac{d\lambda_b(t)}{dt} \\ u_c(t) = R_s i_c(t) + \frac{d\lambda_c(t)}{dt} \end{cases} \quad (2.4)$$

where R_s is the stator resistance. Note that the user convention is adopted.

The equation of the total flux 2.3 can be inserted inside the voltage balance obtaining:

$$\begin{cases} u_a(t) = R_s i_a(t) + L_s \frac{di_a(t)}{dt} + e_a \\ u_b(t) = R_s i_b(t) + L_s \frac{di_b(t)}{dt} + e_b \\ u_c(t) = R_s i_c(t) + L_s \frac{di_c(t)}{dt} + e_c \end{cases} \quad (2.5)$$

where the back electromotive forces are defined by e_a, e_b, e_c .

The back-EMF is a consequence of the movement of the permanent magnet in the magnetic flux produced by the stator currents. It is directly motivated by the Faraday's and Lenz's law. When an EMF is generated by a variation of the magnetic flux according to Faraday's Law, the polarity of the induced EMF is such as to produces a current whose magnetic field opposes the change which produces it, following the formula:

$$e = -N \frac{d\Phi}{dt} \quad (2.6)$$

where N is the number of coils of the stator winding and Φ is the magnetic flux.

Using the thesis notation:

$$\begin{cases} e_a = \frac{d\lambda_{a,mg}}{dt} = -\Lambda_{mg}\omega_{me} \sin(\theta_{me}) \\ e_b = \frac{d\lambda_{b,mg}}{dt} = -\Lambda_{mg}\omega_{me} \sin(\theta_{me} - 2\pi/3) \\ e_c = \frac{d\lambda_{c,mg}}{dt} = -\Lambda_{mg}\omega_{me} \sin(\theta_{me} + 2\pi/3) \end{cases} \quad (2.7)$$

where ω_{me} is the electromechanical velocity. Subsequently the time dependency will be omitted for simplify the notations .

The system explained beforehand is complex to control and deals with sinusoidal currents, this means that the controller implemented have a time-dependent reference. It is good practice to

avoid this useless and demanding operation and simplify the problem switching from a three-phase electrical system to a vector with two components projected to the α - β axes. These axes are fixed to the stator and reduce the number of controller needed to handle the system, but the value of the currents remains variable over time.

For eliminate the dependency on time another bi-axial reference frame has to be adopted, the peculiarity of this frame is that it rotates at the same frequency as the rotor. In this way, the currents appears constant and they can be controlled with less effort. These axes are called d -axis or direct axis and q -axis or quadrature axis, and they represents respectively the direction for the flux produced by the field winding, and the main direction of torque production.

With the hypothesis that the back-EMF phases has a zero homopolar contribute, being a triad of sinusoids out of phase of $\frac{2\pi}{3}$, it is possible to resort to the compact form of space vectors. Rewriting the Equation 2.7 in this way:

$$e_s^s = \frac{d\lambda_{mg}^s}{dt} = \frac{d(\Lambda_{mg}e^{j\theta_{me}})}{dt} = j\Lambda_{mg}\omega_{me}e^{j\theta_{me}} = j\omega_{me}\lambda_{mg}^s \quad (2.8)$$

The voltage balance equation becomes:

$$u_s^s = R_s i_s^s + L_s \frac{di_s^s}{dt} + j\omega_{me}\lambda_{mg}^s \quad (2.9)$$

the superscript s indicates the stator reference.

Then, projecting the space vector into the real axis α and the imaginary axis β can be write again:

$$\begin{cases} u_\alpha = R_s i_\alpha + L_s \frac{di_\alpha}{dt} - \omega_{me}\lambda_{\beta,mg} \\ u_\beta = R_s i_\beta + L_s \frac{di_\beta}{dt} + \omega_{me}\lambda_{\alpha,mg} \end{cases} \quad (2.10)$$

This transformation could also be performed for the other physical quantities but is not useful for the project.

More interesting is the second transformation mentioned above, that it is the association with a reference system synchronous to the position of the rotor.

In general, if g is considered a space vector, it is defined with the new reference system as follows:

$$g^r = g^s e^{-j\omega_{me}} = g^s e^{-j\theta_{me}} \quad (2.11)$$

Applying Equation 2.11 to the Equation 2.9:

$$u_s^r = R_s i_s^r + L_s \frac{di_s^r}{dt} + j\omega_{me} L_s i_s^r + j\omega_{me} \Lambda_{mg} \quad (2.12)$$

where u_s^r indicated the voltage space vector with the moting rotor frame.

In addition to the first transformation, the space vector can be subdivided into the real part u_d and into the imaginary part u_q :

$$\begin{cases} u_d = R_s i_d + L_s \frac{di_d}{dt} - j\omega_{me} L_s i_q \\ u_q = R_s i_q + L_s \frac{di_q}{dt} + \omega_{me} L_s i_d + \omega_{me} \Lambda_{mg} \end{cases} \quad (2.13)$$

Starting from the Equation 2.13 it is possible to obtain the mechanical torque by multiplying both members for $i_d dt$ and $i_q dt$ and summing each member. Notice also that the previous transformations are not conservative for the power, so a corrective factor of $\frac{3}{2}$ must be apply, obtaining:

$$\underbrace{\frac{3}{2}(u_d i_d + u_q i_q)dt}_{P_{absorbed}} = \underbrace{\frac{3}{2}(R_s i_d^2 + R_s i_q^2)dt}_{P_j} + \underbrace{\frac{3}{2}(L_s i_d di_d + L_s i_q di_q)}_{\frac{d}{dt}W_m} + \underbrace{\frac{3}{2}(\omega_{me} \Lambda_{mg} i_q dt)}_{P_{em}} \quad (2.14)$$

$P_{absorbed}$ represents the total power absorbed by the motor which can be divided in three main terms expressed on the right of the equation. The first term, P_j , represents the power losses due to the joule effect on the stator winding; the second term, $\frac{d}{dt}W_m$, constitutes the power stored in the magnetic field; the last term, P_{em} , is due to the mechanical power.

The electric power converted into mechanical must also satisfy the mechanical power formula, $P_{em} = \tau \omega_{me}$, where τ is the torque produced by the motor.

Putting the two equations together the equation for the torque can be finally derived:

$$\tau = \frac{3}{2} p \Lambda_{mg} i_q \quad (2.15)$$

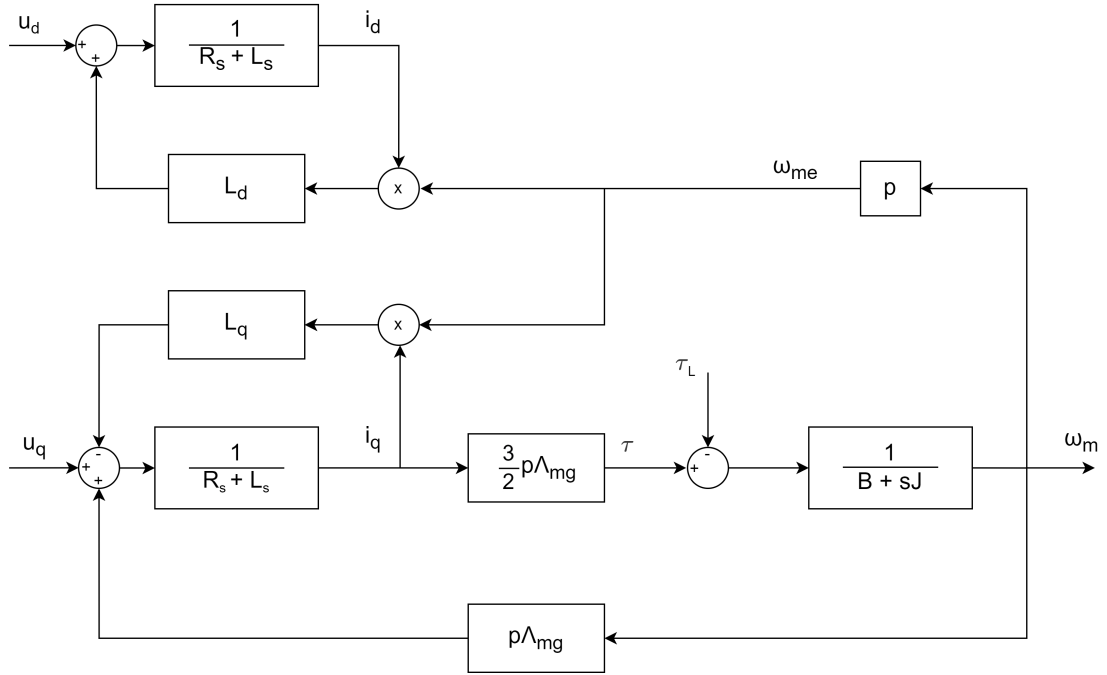


Figure 2.4: Block diagram of a SPM motor in d - q reference frame

The block scheme above includes also the equation for the mechanical load:

$$\tau - \tau_{cog} = \tau_L + B\omega_m + J \frac{d\omega_m}{dt} \quad (2.16)$$

where τ_L is the torque due to the load, the second term $B\omega_m$ is the contribute of the friction, while the last $J \frac{d\omega_m}{dt}$ is the contribution of inertia.

Particular attention can be given to τ_{cog} which is called cogging torque and is the torque needed to overcome the magnetic attractive force between the magnets on the rotor and the iron teeth of the stator.

2.2.1 SPM LIMITS AND OPERATING REGIONS

Every electrical machine must respect limits in terms of electrical quantities, this is the case also for the SPM motors. In practice these values are specified in the user manual as nominal values, which are safety values for working.

The limits are expressed with the formulas:

$$\begin{cases} I_N^2 \geq I_d^2 + I_q^2 \\ V_N^2 \geq V_d^2 + V_q^2 \end{cases} \quad (2.17)$$

where I_N and V_N are the nominal current and voltage.

To define the operating region it is better to assume that the machine is working in steady state.

Then, the Equation 2.13 becomes:

$$\begin{cases} U_d = R_s I_d - \Omega_{me} L_s I_q \\ U_q = R_s I_q + \Omega_{me} L_s I_d + \Omega_{me} \Lambda_{mg} \end{cases} \quad (2.18)$$

If these two equation are substitutes in the voltage limits equation, it can be expressed as a function of the currents. By rearranging and neglecting the voltage drop of the resistor, the following equation is obtained:

$$\left(I_d + \frac{\Lambda_{mg}}{L_s}\right)^2 + I_q^2 \leq \frac{U_N^2}{\Omega_{me}^2 L_s} \quad (2.19)$$

Using also the iso-torque lines that follows the Equation 2.15, all these formulas depend on the currents and can be plotted in the I_d, I_q plane.

The current limit is a circumference with center in the origin and I_N as radius. The Equation 2.42 that represents the voltage limits is also a circumference but with the center in $(I_d^C; I_q^C) = \left(-\frac{\Lambda_{mg}}{L}; 0\right)$ while the radius is not fixed and it is inversely proportional to the rotor speed. The iso-torque are horizontal lines with respect to the d -axis since they depend only on I_q .

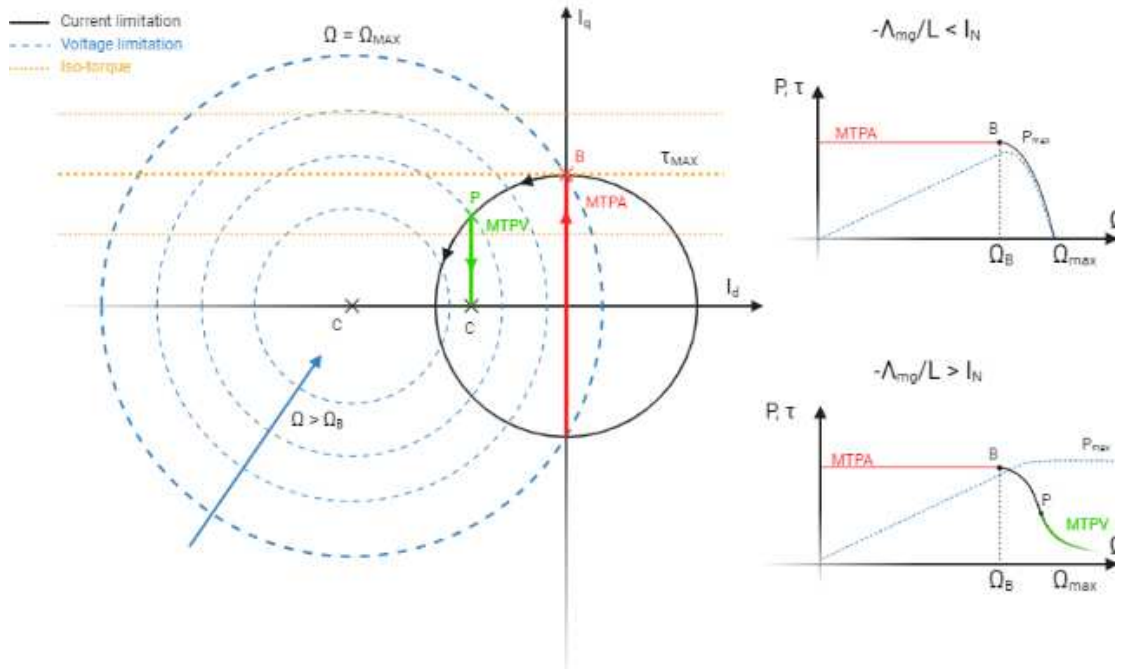


Figure 2.5: Functional limits of SPM and power curves

In our operation, the goal is to maximize the torque contribution. In the SPM motor this is achieved by following the maximum torque per ampere (MTPA) curve, which is composed by the tangent points of the iso-torque and the current limits. In case of SPM motor the MTPA is a vertical axis, since only the I_q current contributes to the torque, Equation 2.15.

The maximum speed at which the maximum torque is available is called base velocity Ω_B , it's value is obtained by imposing $I_q = I_N$ and $I_d = 0$, obtaining the Equation 2.20:

$$\Omega_B = \pm \frac{U_N}{\sqrt{\Lambda_{mg}^2 + L_s^2 I_N^2}} \quad (2.20)$$

Nevertheless, it is possible to go at high speeds at the expense of no longer having the maximum torque. The maximum velocity can ideally be reached applying $I_q = 0$ and $I_d = I_N$ and substituting these values on the voltage limits, obtaining:

$$\Omega_{MAX} = \pm \frac{U_N}{\sqrt{\Lambda_{mg} + L_s I_N}} \quad (2.21)$$

This is only an ideality as the torque at this point will be zero and the motor will not be able to

overcome basic friction and inertia contributes.

Different power curves can be observed if the center of the voltage limits is within the circumference of the current limits. In this situation exists a point "P" where iso-torque lines are tangent to the voltage limits by constructing the maximum torque per volt (MTPV) curve. Following the MTPV allows convenient condition and maximum power.

2.2.2 CLARKE AND PARK TRANSFORMATIONS

The above reference frame transformation used to derive a linear representation of the motor are called Clarke and Park transformations.

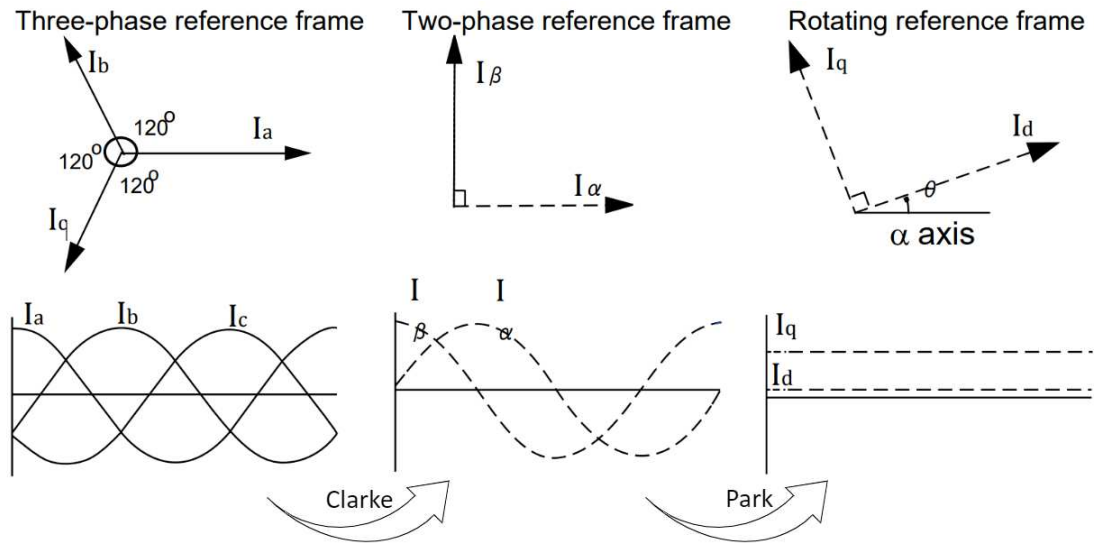


Figure 2.6: Current signal with different reference frames

The Clarke transformation is used to pass from a three axis reference frame $a-b-c$ to the reference $\alpha-\beta-0$ both fixed to the stator, as represented in Figure 2.6 [5].

The 0 axis is the homopolar component and it is defined to derive a square transformation matrix.

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \underbrace{\frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ a & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}}_{T_{Clarke}=T_{abc/\alpha\beta}} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} \quad (2.22)$$

Instead, the Park transformation is used to pass from the $\alpha\text{-}\beta\text{-}0$ reference to the $d\text{-}q\text{-}0$ reference frame which moves as the rotor frequency. As defined in Figure 2.3, from the angle relation between the two frames it is possible to derive the following relationship:

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \theta_{me} & \sin \theta_{me} & 0 \\ -\sin \theta_{me} & \cos \theta_{me} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{T_{Park}=T_{\alpha\beta/dq}} \begin{bmatrix} d \\ q \\ 0 \end{bmatrix} \quad (2.23)$$

Using the inverse matrices T_{Clarke}^{-1} and T_{Park}^{-1} can be performed the inverse transformations, from $\alpha\text{-}\beta\text{-}0$ to the reference $a\text{-}b\text{-}c$ and from $d\text{-}q\text{-}0$ to the reference $\alpha\text{-}\beta\text{-}0$.

$$T_{Clarke}^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix}; T_{Park}^{-1} = \begin{bmatrix} \cos \theta_{me} & -\sin \theta_{me} & 0 \\ \sin \theta_{me} & \cos \theta_{me} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.24)$$

These transformation can be different and are based on position hypothesis. In this case the a -axis is equal to the α -axis and the electrical angle θ_{me} represents the phase shift between the α -axis with the d -axis. These hypothesis can be different based on the designer.

In the ST motor control library, discussed in chapter 4, the hypothesis and so the transformations are different. The a -axis is equal to the α -axis as always, but the electrical angle θ_{me} represents the phase shift between the α -axis with the q -axis.

2.3 SPM CONTROL TECHNIQUES

Sinusoidal control can be divided in three main groups: scalar control, vector control and new control techniques [6].

The scalar control is based on a simplified model which is only valid for steady states. In this method, the amplitude and frequency of voltage, currents, and flux linkage space vectors are controlled.

The second control method is known as the vector control technique which not only controls the amplitude and frequency of voltage, current and flux space vector, but precise control of the position of these space vectors is also achieved.

The most important vector control algorithms can be further subdivided into three categories: field oriented control, direct torque control (DTC) and voltage vector control (VVC).

The FOC methods uses the d - q coordinate transformation that rotates in sync with the rotor to decouple the control. In this way, the flux and torque producing component of the stator current can be controlled separately with two loops. This method is highly dependent on the estimation of the rotor angle and requires accurate measurements.

The second vector control technique is the DTC which allows flux and torque to be controlled without the internal current control loop but using hysteresis controllers. This results in good dynamic performance of PMSM drives. However, the DTC suffers from various disadvantages, it needs a fast sampling time, two hysteresis controllers, a variable switching frequency behavior, and higher ripples in torque and flow.

DTC and FOC have been studied and compared for a long time, both have advantages and disadvantages and their use depend on the type of application, in general DTC has a higher performance response with faster torque dynamics, while FOC has better steady-state behaviour [7].

The last vector technique is the VVC that results in good steady state and transient performance of the PMSM drive, it is also well tested for induction motor drives.

There exists several others methods like feedback linearization control and passivity based control, that can be grouped in the new control techniques category [8].

2.3.1 FIELD ORIENTED CONTROL

The algorithm chosen to develop the gate drives control is the FOC with the only difference that the optional speed control loop is implemented using a fuzzy PID.

This vector control strategy is able to maximize the efficiency in the torque-current conversion allowing to have a high dynamic response, a high precision of torque, velocity and position control.

The main advantage of FOC is the possibility to separately control phase and modulus of the current, through the d and q axes, and generate the optimal voltage vector. Nevertheless, this technique also have a major limitation, to produce an optimal voltage vector it is strictly necessary to constant monitoring the electrical phase shift angle θ_{me} for perform the Park and inverse Park transformations.

Electric angle computation is still a great research topic. It has generated many possible algorithms so far. The main solutions can be divided into two groups: sensor control and sensorless control algorithms.

Using sensors allows to have a reliable and robust angle estimation at any speed range with the

disadvantage of increasing the costs of the whole system. Several types of sensors are available on the market, such as: optical encoders, Hall effect sensors, variable reluctance wheel speed sensors and accelerometers.

Optical encoder are implemented in many motor control systems since they have a high resolution, generating a good estimation for the electrical angle, so what the FOC needs.

Encoders are generally expensive, they can reach 30 % of the total costs, and the measure can be contaminated. A Hall effect based encoder is usually a cheaper solution, the cost is about 1.50 \$, the installation is cheaper, as it can be built into the stator. Hall sensors determines a solution more suitable to this industrial environment reducing a lot the total costs.

The big drawback of Hall sensor is the resolution, they deliver just two switching events over one engine revolution. The events are generated when a magnetic field crosses the transducer plate in a direction perpendicular to the plane of the plate, exploiting the Hall-effect principle. Whenever the rotor magnetic poles pass near the digital Hall sensors they emit a high or low signal indicating that the north or south pole is passing close to the sensors.

Generally, in a motor three sensors are mounted in the stator, out of phase each other by 120° or 60° and with the possibility to have an offset with respect to the three motor phases. The assembly is a delicate task that can compromise the calculation of the angle and the total FOC control, for this purpose a Hall offset estimation has been carried out, section 2.4.

The three sensors will only provide six switching events separated by 60° in electric angle, as can be seen this is not sufficient for run the FOC control. Some estimation algorithms are necessary for compute the angle among the events, such as zero order, first order, and vector tracking observer algorithms.

The project will adopt the simple but efficient zero order algorithms where only the zeroth-order term of an approximated Taylor series expansion is taken into account. Other algorithm are not been tested.

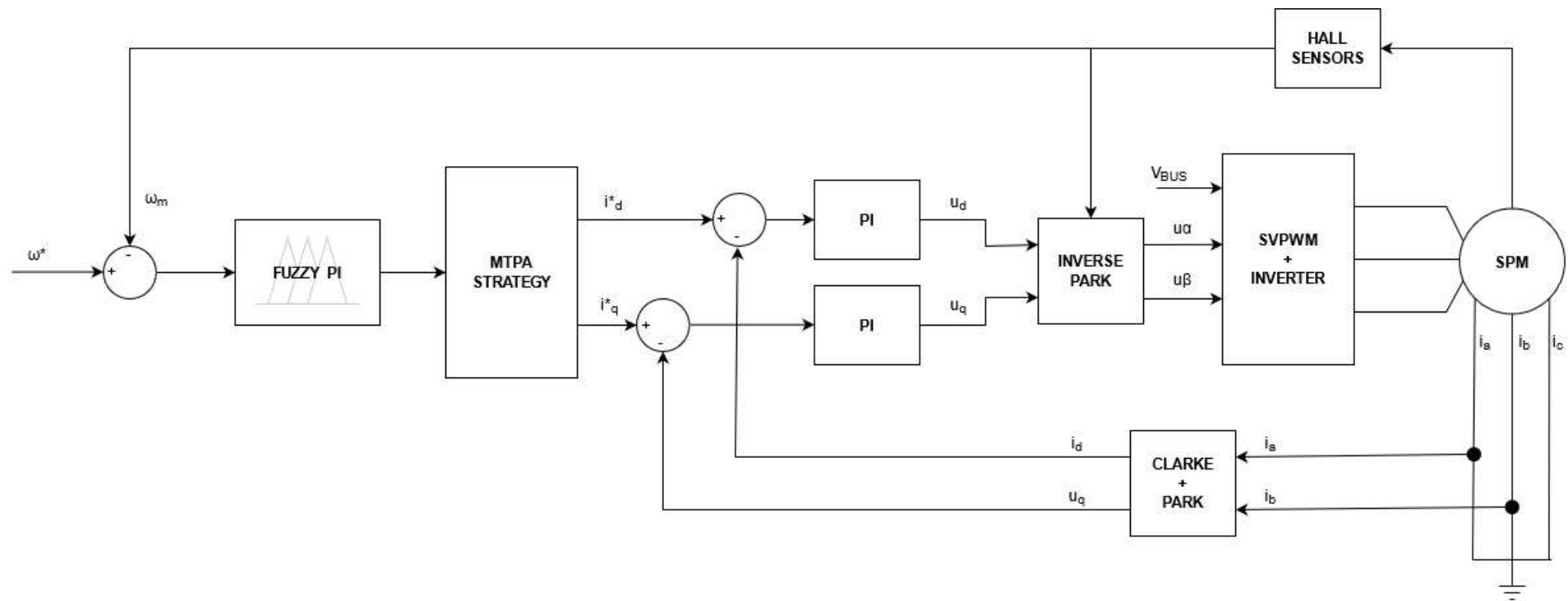


Figure 2.7: Sensored FOC control block scheme with FLC for velocity loop

2.3.2 SPEED LOOP CONTROL METHODS

The FOC control can be improved by adding a speed controller for have a system that follows a reference speed with the maximum torque per ampere. The difference with respect a usual speed loop is that the plant is composed by the FOC control system so the system is already performing a current control loop to give precise torque contributions.

Gate drives performs a trapezoidal speed path increasing the speed while starting and decreasing it in case of stopping phase. Therefore, the adoption of a speed loop is essential.

There exists many types of controller that would work for a closed loop speed control system and there are some points to consider before choosing the control system.

In control theory is common practice to linearize the mathematical model, analyze the modes of the system and then design the controller. However, the state space model can be highly non-linear which means that if one would linearize it, it would yield a non-working model, hence the choice of control methods to evaluate should not include methods that depend on an accurate linearized mathematical model. Considering the application in which the control is implemented, the control system should be able to handle noise and disturbances well.

Gate drives are a quite simple system that does not needs of non linear models and non linear controls, but the interest as a research topic must not be overlooked.

By these considerations can be concluded that the chosen controllers that will be evaluated are: PI (Proportional-Integral) and Fuzzy PI.

The control system formed, which includes one or more inner loop is called cascade control. It is important when designing a cascade controller for a PM motor that the inner controller is around 4-5 times faster (higher bandwidth) than the outer controller or the system can become unstable.

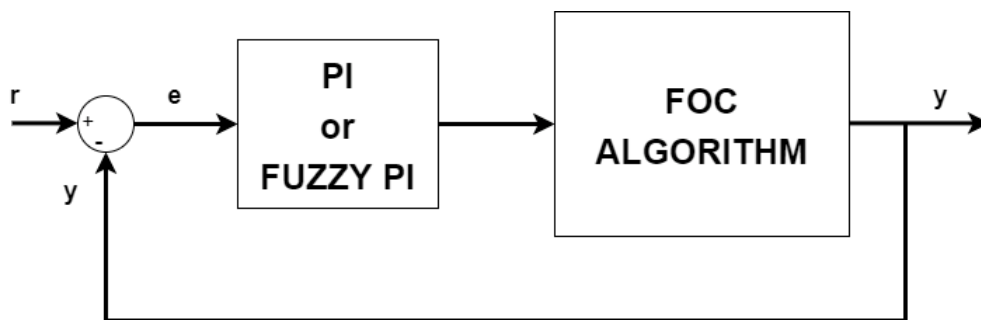


Figure 2.8: Close loop speed control scheme without mechanical block

PID CONTROLLER

PID controller stands for proportional, integral and derivative control. It is mainly used for a closed loop system where the control is based on the error term $e = r - y$ of the system, where r is the reference input and y is the measured output.

The equation for the PID controller in the frequency domain is written as:

$$C(s) = K_P + \frac{K_I}{s} + K_D s \quad (2.25)$$

where K_P is the proportional term, K_I the integral term and K_D the derivative term.

Basically, K_P multiply the error term, K_I multiply the accumulated error over time removing the steady-state error and the K_D term multiply the derivative of the error.

The derivative term will not be used in the analysis and simulations, due to the fact that it amplifies the high frequencies of the system, which means that noise and similar disturbances will have a bigger effect on the system. As the goal is to make a robust controller, only the proportional and integral part will be considered.

As a result, the closed loop system of Figure 2.8 can be written in frequency domain using the PI controller:

$$\Omega(s) = \frac{G(s)C(s)}{1 + G(s)C(s)} \quad (2.26)$$

where $C(s)$ is the PI transfer function and $G(s)$ is the FOC transfer function.

FUZZY LOGIC PID CONTROLLER

Fuzzy logic controllers (FLC) follow the idea of fuzzy logic (FL) where unlike classical logic it could realize values between false and true. The main advantage with respect to the classical PID is that FL can be used for define different output sets depending on the inputs and in case of adapting fuzzy PID can perform a gain PID auto tuning. This results in an excellent tool for control systems that are difficult in modelling or for non-linearity.

In the thesis the black box is not the motor but is the mechanical part. The inertia and the static/dynamic friction after the gearbox may vary based on the installation and on the environment, causing different behaviours for the same product.

The fuzzy PID is structured in four main blocks, as can be seen from Figure 2.9.

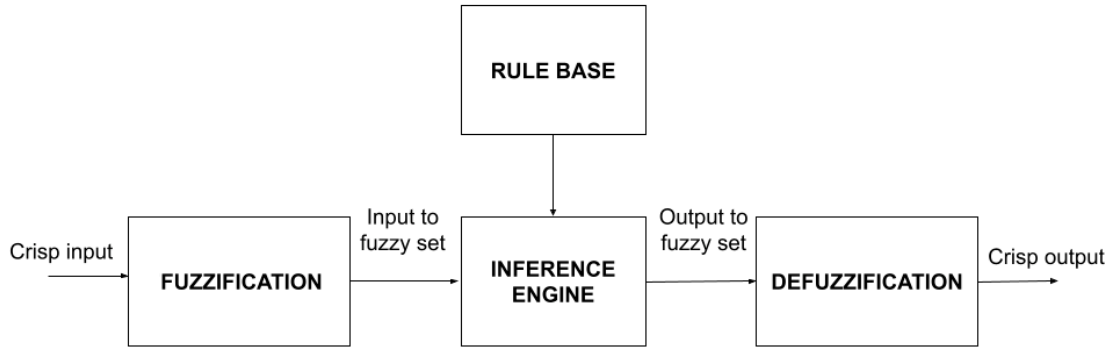


Figure 2.9: Fuzzy logic block scheme

The first block is the "Fuzzification" which scales the crisp inputs by mapping them through membership functions μ into fuzzy one. The membership functions can be defined with many shapes, the most common shapes are: triangles, trapezoidal and Gaussian.

The second block is the "Rule Base", it is the core of the FLC, constitute by a set of rules of the form IF-THEN statement that describe the state and the behavior of the control system.

The "Inference Engine" is the process that relates input fuzzy sets to output fuzzy sets using if-then rules and fuzzy operators to determine a reasonable output fuzzy value.

The last block is the "Defuzzification" that converts the fuzzified output values to crisp control values using the output membership function.

For each of these blocks there are several implementations, in this project a very simple Fuzzy PI control is implemented, avoiding the derivative term.

The fuzzification is performed using triangle shaped membership functions and the number of Fuzzy areas is chosen to be three. The defuzzification is based on the center of gravity (COG) method. Basically, COG weights the inputs and finds the center of gravity for all combined areas by applying the Formula 2.27 :

$$Y = \frac{\int \mu(z)z dz}{\int \mu(z) dz} \quad (2.27)$$

where: Y is the crisp output, z is the fuzzy output signal and the $\mu(z)$ is the membership function mapping result.

The used fuzzy PI does not implement a particular fuzzy rule evaluator engine. In fact, the adoption of a classic PI does not make big differences in the control performance but it opens the possibility of testing and use more sophisticated controls.

A comparison between a classic PID and the implemented one has been done in section 5.2.2 as required by the company and shows the results stated before.

For more information the C code is available and published in the reference [9]. Most applications needing fuzzy logic should consider a fuzzy rule evaluator engine; see reference [10] for example.

2.3.3 MODEL BASED METHODS FOR SENSORLESS TECHNIQUES

The other main algorithms used for electrical angle calculation are based on sensorless strategies, namely those techniques that does not use sensors such as optical encoders and hall sensors, but through estimators are able to compute the electrical angle θ_{me} .

The benefits of these methods are immediate, the elimination of sensors reduce greatly the costs of the system and increase reliability. On the other hand, the estimates will never be as accurate as its measurement via sensors, making everything more complicated in terms of performance and stability. In fact, for applications that require position control or for a high degree of accuracy an encoder is essential.

Sensorless strategies strongly depend on the type of motor is handled and on the rotor velocity. With SPM motors, that have no saliency, i.e. $L_q = L_d$, the sensorless techniques are limited. As can be observer from Table 2.1 at low speeds there are no implementable algorithms, so it is necessary to adopt different strategies for starting the engine.

	Low speeds	High speeds
SPM motors	No sensorless solutions	Fundamental model based
IPM motors	Based on saliency estimation	Fundamental model based

Table 2.1: Sensorless solutions

The absence of an efficient sensorless algorithm at zero or low speed constitute a drawback, forcing to use different techniques and switching on the motor run. The results are visible in a more complex electronics and a more computational expensive task.

A valid alternative for SPM motors consist in introducing anisotropy, as the called ringed-pole motors. Adopting the same methods used for drive the IPM or reluctance motors at low speeds. These methods are based on high frequency signal injected on the stator, which is able through the physical response of the motor to recognize the anisotropy of the rotor.

In a industrial environment this and the aforementioned drawbacks represent a problem. Costs,

reliability, complexity and noise contribution for high frequency signals lead to adopt other methods adaptable in a large manufacturing environment.

Another practical solution can be reached by driving the motor in an open-loop set up using a relevant start-up procedure and switching to model-based sensorless technique when a satisfying speed is reached. This method is within the scope of the CAME project since the gate drives are a simple system and the open-loop phase does not represents a big problem if monitored with safety devices, like photocells and impact tests.

For medium-high speed scenarios the electrical angle estimation can be performed by reconstructing the back-EMF or the flux linkage space vectors with voltage and current measurements. Back-EMF is not available at low speeds as the effect of permanents magnets on the rotor is negligible, look at Equation 2.6, that's why it cannot be applied at low speeds.

Model based solutions can change the configuration based on the dynamical model of the rotor topology, nevertheless they exploit the same physics. The most popular is based on back-EMF with a state observer, but in general can be found solutions based on: open-loop methods, closed-loop method with full or reduce order observer, extended kalman filter (EKF), sliding mode observer (SMO), Model reference adaptive control (MRAS) and Artificial Intelligence (AI). All these methods are valid and have advantages based on the application.

In the project will be seen only the Luenberger Observer (LO) based on back-EMF estimation in the stationary $\alpha\beta$, but also in the rotating $d-q$ frame. Although the MRAS can offer better performance LO represent a valid and more simple alternative for state if the gate control application is executable.

LUENBERGER OBSERVER

For proceed with the LO equations must be formalized the SPM model in state-space form.

In general, an arbitrary system can be written as:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned}\tag{2.28}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$ and $C \in \mathbb{R}^{q \times n}$ are matrices with different dimensions in accordance with the system order, x and y are referred as state variables and outputs, u is the system input.

A state observer, in control theory, is a system that provides an estimate of the internal state of a real system, given its input and output measurement. As said before, exists different types

of observers, the one analyzed is the conventional Luenberger observer shown in Figure 2.10, while the state equation are written in the following form:

$$\begin{aligned}\hat{\dot{x}} &= A\hat{x} + Bu + K(y - \hat{y}) \\ \hat{y} &= C\hat{x}\end{aligned}\tag{2.29}$$

where the hat symbol indicates an estimated quantity and K is the observer gain matrix. From the Table 2.2 can be seen the values assigned to each member of state observer Equation 2.29.

Model	x	A	B	C	u
Linear Back-EMF α - β	$\begin{bmatrix} i_{\alpha\beta} \\ e_{\alpha\beta} \end{bmatrix}$	$\begin{bmatrix} \frac{R_s}{L_s} & \frac{j\omega_{me}}{L_s} \\ 0 & j\omega_{me} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{L_s} \\ 0 \end{bmatrix}$	$\begin{bmatrix} i_{\alpha\beta} & 0 \end{bmatrix}$	$v_{\alpha\beta}$
Linear Back-EMF d - q	$\begin{bmatrix} i_{dq} \\ e_{dq} \end{bmatrix}$	$\begin{bmatrix} \frac{R_s}{L_s} & \frac{-1}{L_s} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{L_s} \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	v_{dq}

Table 2.2: SPM linear model in state space representation

The internal states of the motor are the back-EMF and the phase currents, while the input and output quantities supplied are the phase voltages and measured currents, respectively.

The observer can be developed in the stationary α - β or in the synchronous d - q reference frame and it combined with a phase locked loop (PLL) provides both position and velocity estimation, the former used for vector control orientation, the latter to close the speed control loop [11]. In the thesis both the algorithms will be simulated in order to compare them but only the α - β operations are reported below, since in d - q the formulas are very similar.

Starting from the continuous-time model representing the electrical subsystem of an isotropic motor in the stationary reference frame the Equation 2.10 can be re-written in the Laplace domain as:

$$U_{\alpha\beta} = R_s I_{\alpha\beta} + sL_s I_{\alpha\beta} + E_{\alpha\beta}\tag{2.30}$$

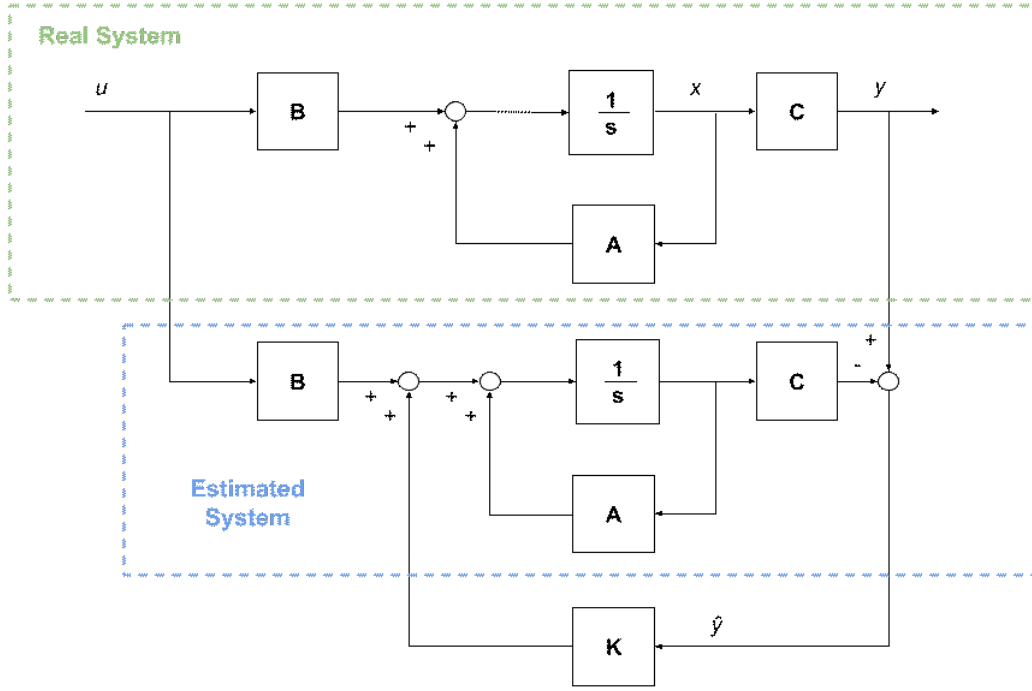


Figure 2.10: Luenberger observer block scheme

where $U_{\alpha\beta}$, $I_{\alpha\beta}$, $E_{\alpha\beta}$ are the space vectors, complex variables, of the stator voltage, stator current and the EMF induced by the permanent magnet flux linkage in the stator winding (i.e. the back-EMF).

Introducing also a dynamic equation of the back-EMF with the hypothesis of slowly varying electrical speed ω_{me} , the back-EMF can be approximated as:

$$sE_{\alpha\beta} = j\omega_{me}E_{\alpha\beta} \quad (2.31)$$

Finally, using Equation 2.30 and 2.31 with the measured currents it is possible to estimate the back-EMF using an extension of the classical Luenberger linear observer of the motor currents.

$$\begin{aligned} s\hat{I}_{\alpha\beta} &= \frac{1}{\hat{L}_{ss}}(U_{\alpha\beta}^* - \hat{R}_s\hat{I}_{\alpha\beta} - \hat{E}_{\alpha\beta}) + K_1(I_{\alpha\beta} - \hat{I}_{\alpha\beta}) \\ s\hat{E}_{\alpha\beta} &= j\omega_{me}\hat{E}_{\alpha\beta} + K_2(I_{\alpha\beta} - \hat{I}_{\alpha\beta}) \end{aligned} \quad (2.32)$$

where $K = [K_1, K_2]$ are real observer gains, $\hat{I}_{\alpha\beta}$ and $\hat{E}_{\alpha\beta}$ are current and back-EMF estimations.

Discretizing the equation with T_s as sampling period:

$$\begin{aligned}\hat{I}_{\alpha\beta}(k+1) = & \hat{I}_{\alpha\beta}(k) - \frac{T_s}{\hat{L}_{ss}}(U_{\alpha\beta}^*(k) - R_s\hat{I}_{\alpha\beta}(k) - \hat{E}_{\alpha\beta}(k)) + \\ & + K_1T_s(I_{\alpha\beta}(k) - \hat{I}_{\alpha\beta}(k))\end{aligned}\quad (2.33)$$

$$\hat{E}_{\alpha\beta}(k+1) = \hat{E}_{\alpha\beta}(k) + j\omega_{me}T_s\hat{E}_{\alpha\beta}(k) + K_2T_s(I_{\alpha\beta}(k) - \hat{I}_{\alpha\beta}(k))$$

The electrical angle can be calculated simply with:

$$\theta_{me} = \omega_{me}t = \arctan\left(-\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right)\quad (2.34)$$

A conventional PLL can be used to improve the accuracy of the $\hat{\theta}_{me}$ angle estimate. It takes the estimated back-EMF provided by the observer and returns the rotor electrical angle and the speed, according to equations:

$$\begin{aligned}-\hat{e}_\alpha(k) &= \lambda_{mg}\omega_{me} \sin(\hat{\theta}_e(k)) \\ \hat{e}_\beta(k) &= \lambda_{mg}\omega_{me} \cos(\hat{\theta}_e(k))\end{aligned}\quad (2.35)$$

Then:

$$\begin{aligned}-\hat{e}_\alpha(k) \cos(\theta_{me}(\hat{k}-1)) - \hat{e}_\beta(k) \sin(\theta_{me}(\hat{k}-1)) = \\ \lambda_{mg}\omega_{me} \sin(\hat{\theta}_{me}(k) - \hat{\theta}_{me}(k-1)) \cong \\ \lambda_{mg}\omega_{me}(\hat{\theta}_{me}(k) - \hat{\theta}_{me}(k-1))\end{aligned}\quad (2.36)$$

Regulating a PI controller that keep it's input to zero $\hat{\omega}_{me}$ and $\hat{\theta}_{me}$ are obtained.

The Figure 2.11 is a representation of a PLL block scheme, this configuration has a problem on managing directions but it is computationally simple. Other modification of this algorithm can be considered as: improved and normalized PLL, arctangent methods (computationally expensive) or CORDIC algorithms.

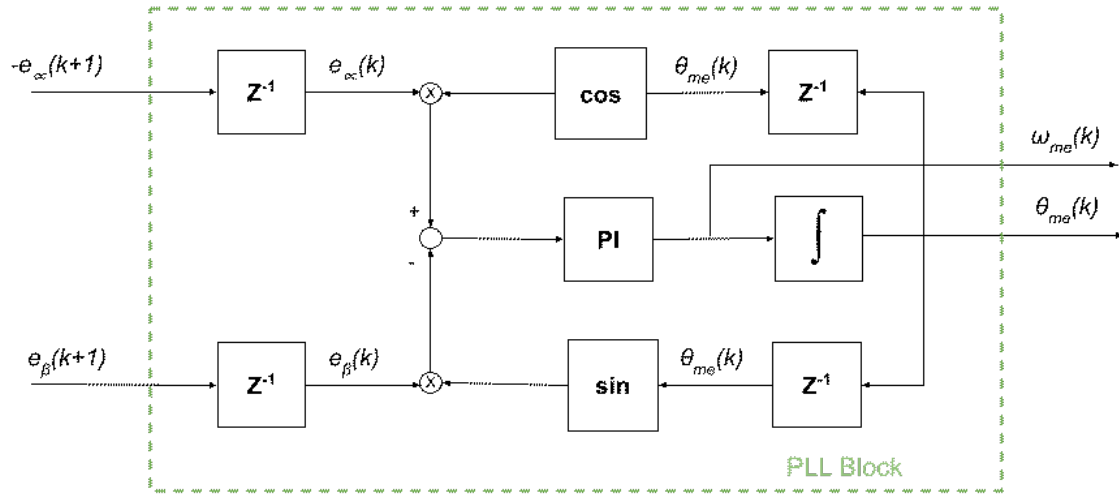


Figure 2.11: Classic PLL block scheme in discrete time

2.4 ESTIMATION OF PARAMETERS

The second part of the chapter starts here. This section is reserved to describe some experiments conducted to obtain reliable parameters for the control algorithm and for the simulation.

Good and accurate measurement and estimation are essential for contract valid models, as they directly depend on the engine parameters. Errors in the estimation or in the measurement will inevitably lead to a degradation of control performance.

The main sources of errors in the position estimate are usually due to: the non-ideality of the inverter that powers the motor, the current measured on the phases and the parametric variations. Resistance and inductance of the stator depends strongly on the operating conditions of the system.

Starting from the limited datasheet made available by the company and tabulated in Table 2.3 some measurements are performed for confirm the data.

Performance @ 24V

Low Size – 300W

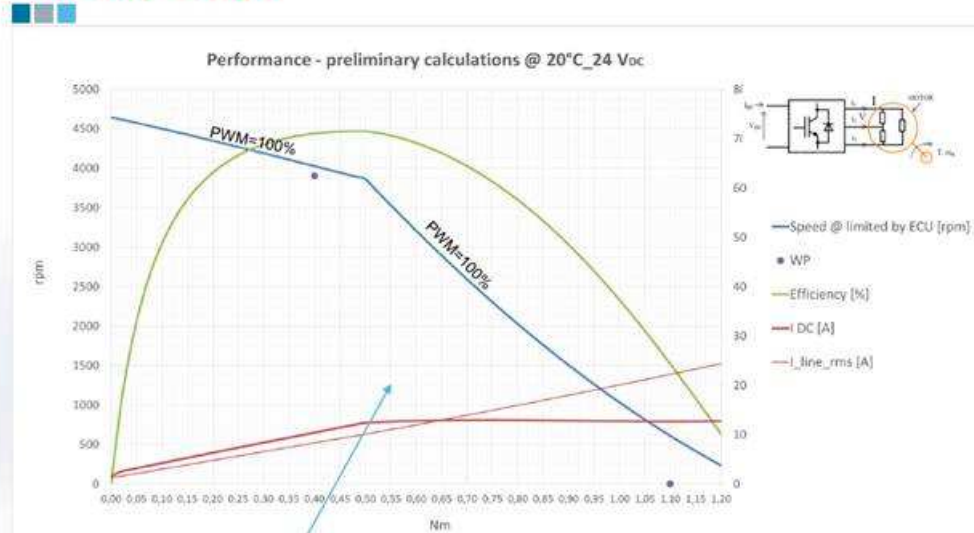


Figure 2.12: Performance and characteristic curves of the BLDC motor

Characteristics	Values
Machine type	Brushless SPM, inner rotor
Nominal Voltage [V]	24
Number of poles pairs	5
Phase connection	Delta
Phase to Phase B-EMF for 1000rpm, 20°C [V]	5.1
Phase to phase resistance, 20°C [mΩ]	245
Phase to phase inductance [μH]	210
Peak to peak cogging torque [mNm – pkpk]	23
Sensored/Sensorless	Sensored (3 HALL)
Maximum flux linkage [Wb]	0.00977
Rotor inertia [gcm ²]	150
Torque constant [Nm/Arms]	0.0570

Table 2.3: BLDC motor datasheet

The first thing done is to measure the electric parameters with a LCR meter that makes AC

measurements with a defined frequency. The quantification must be performed multiple times for different rotor position and frequency, the mean of the values founded are reported below:

Measurements (at $f = 1kHz$)	Values
Phase to phase inductance [μH]	220
Phase to phase resistance [$m\Omega$]	334
Measurements (at $f = 100Hz$)	Values
Phase to phase inductance [μH]	211
Phase to phase resistance [$m\Omega$]	217

Table 2.4: Measurements of the electrical parameters

The model uses the stator resistance and inductance, while the measurements are taken from phase to phase. For calculate the conversion is necessary to know the motor configuration that can be either in delta or in wye.

For delta motors the voltage applied across each winding will be equal to the line voltage, so it is more suitable for high starting torque, wye connection has a lower line voltage given by the formula: $V_{ph-ph} = \frac{V_{line}}{\sqrt{3}}$.

The stator resistance R_s can be calculated with the formula:

$$R_{Phase} = \frac{3R_{line}}{2} = 2R_s \quad (2.37)$$

The same computation can be done for the stator inductance L_s .

2.4.1 STATOR RESISTANCE ESTIMATION AND INVERTER NON-IDEALITY

The stator resistance can also be estimated by performing some tests in open-loop by applying some fixed voltages in the d -axis. By measuring the currents and using the imposed voltages, the graph $V-I$ of Figure 2.13 is plotted, in which the resistance is represented by the slop of the line. From the figure below the equivalent resistance is $R = 0.1363 \Omega$.

The figure stands out also the non-linearity of the inverter that affects the measurement with an offset error on the voltage [12], expressed by the equation:

$$u_{d,ref} = R_s i_d + u_{d,err}(i_d) \quad (2.38)$$

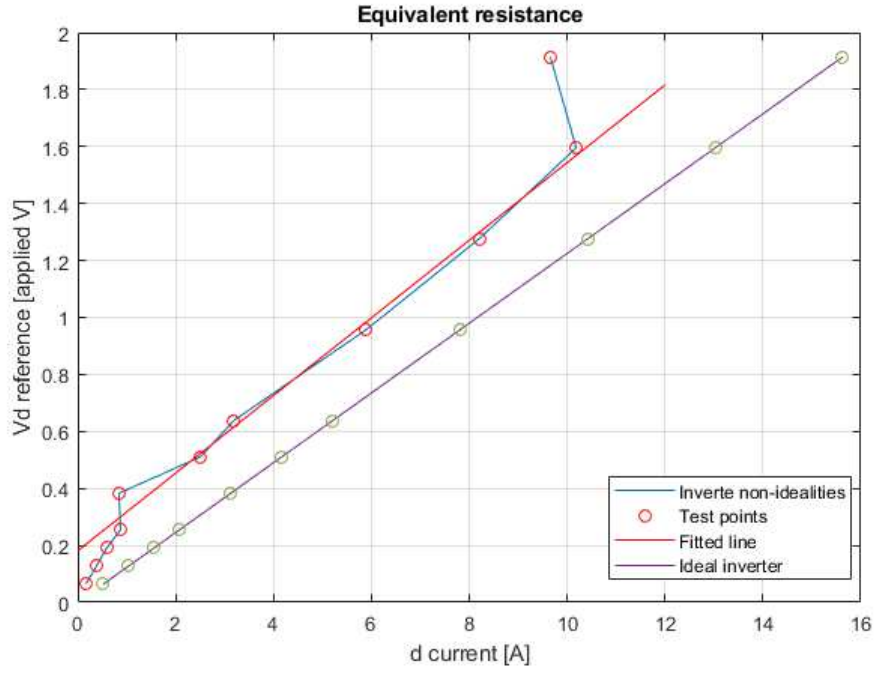


Figure 2.13: Equivalent resistance with different voltage references in d-axis

The main sources not considered are: dead time/blanking time, turn-on and turn-off delay times of the semiconductors, voltage drop on semiconductors, dead time due to the parasitic capacitances of semiconductors and zero-current-clamping effect ([13], [14], [15], [16]).

The inverter voltage error in one phase can be described as follows:

$$\Delta u_{err} = \left[\frac{T_d + t_{on} - t_{off}}{T_s} U_{DC} + \frac{u_{CE} + u_F}{2} \right] \text{sign}(i_d) \quad (2.39)$$

where T_d represents the blanking time, also called dead time, t_{on} and t_{off} denote the equivalent switch on and switch off delay time (in our case IGBT) respectively, T_s is the PWM switching period, U_{DC} is the DC-link voltage, u_{CE} and u_F are the forward voltage drops over the IGBT and diode respectively, [15].

The ideal inverter curve and the non ideal inverter curve of Figure 2.13 can be subtracted obtaining the Δu_{err} trend on Figure 2.14.

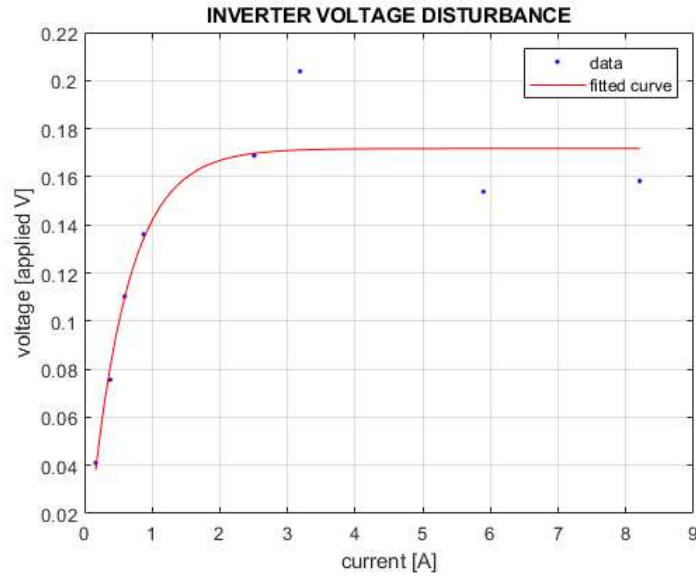


Figure 2.14: V-I plot with different voltage references in d-axis

Summarizing all the measurements and estimations:

Type of data	$R_s [\Omega]$
Datasheet	0.1225
Measured	0.137
Estimated	0.1363

Table 2.5: Resistance values with different methods

In the thesis will be used the data derived by the estimation and not that of the datasheet.

2.4.2 HALL OFFSET ESTIMATION

When Hall sensor are placed in the engine structure, there is always an angular offset with respect to the phases. Knowing this data allows to provide the rights reference of the voltages for maximize the current in the q -axis. Hence, it is a parameter that directly affects the performance of the FOC control.

To solve and extrapolate this data there are different techniques, the simplest way is by looking the current consumption and by changing in run the Hall offset angle it is possible to obtain the best angle in terms of current and torque response.

This approach is possible only if the software permits to change a variable multiple times while the code is running and in case of the FOC algorithm that strictly depends on the electrical angle θ_{me} can leads to instability.

For the purpose an experimental test is conducted for estimate the Hall offset without using the running code. It consists in controlling the speed of the motor by coupling the SPM motor under analysis with a DC driving motor with an encoder.

In this way, the Hall sensor data and the induced back-EMF of each phase can be extrapolated and by performing some offline analysis of the data it is possible to computed the angle between the rising edge of the digital Hall sensor and the maximum of the back-EMF that corresponds exactly to the Hall offset, Figure 2.15.

This procedure has been done for different velocities and for both the directions.

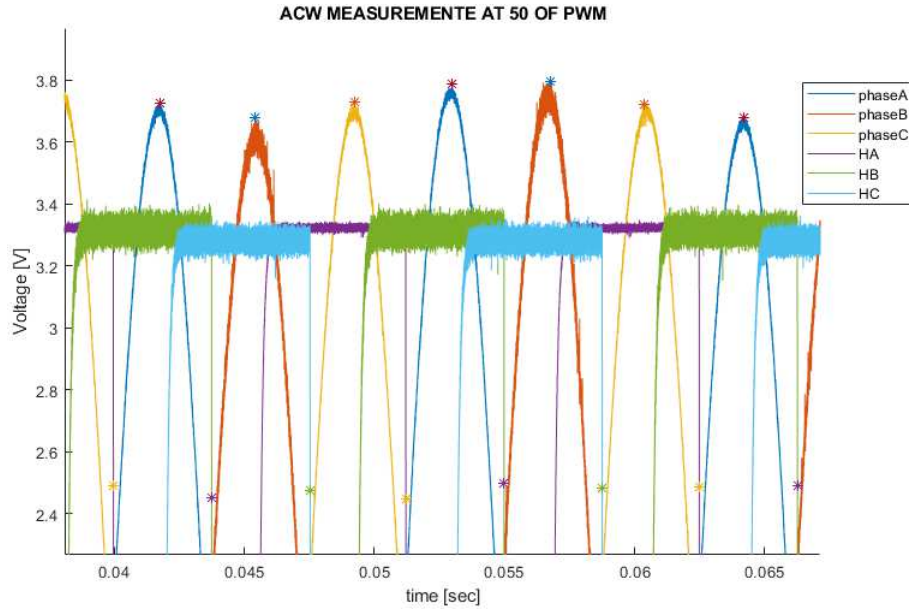


Figure 2.15: Hall signals and back EMF for each phase

Before proceeding with the post process calculations it is always better to perform some frequency analyzes in order to verify the noise measurements.

In Figure 2.16 are present the spectrums of the back-EMF of each phase where can be distinguish a single peak in correspondence of the frequency velocity, following the equation: $\omega = 2\pi f$ where $\omega = 111.2 \frac{rad}{s}$ and so the frequency is $f = \frac{\omega}{2\pi} = 90 Hz$.

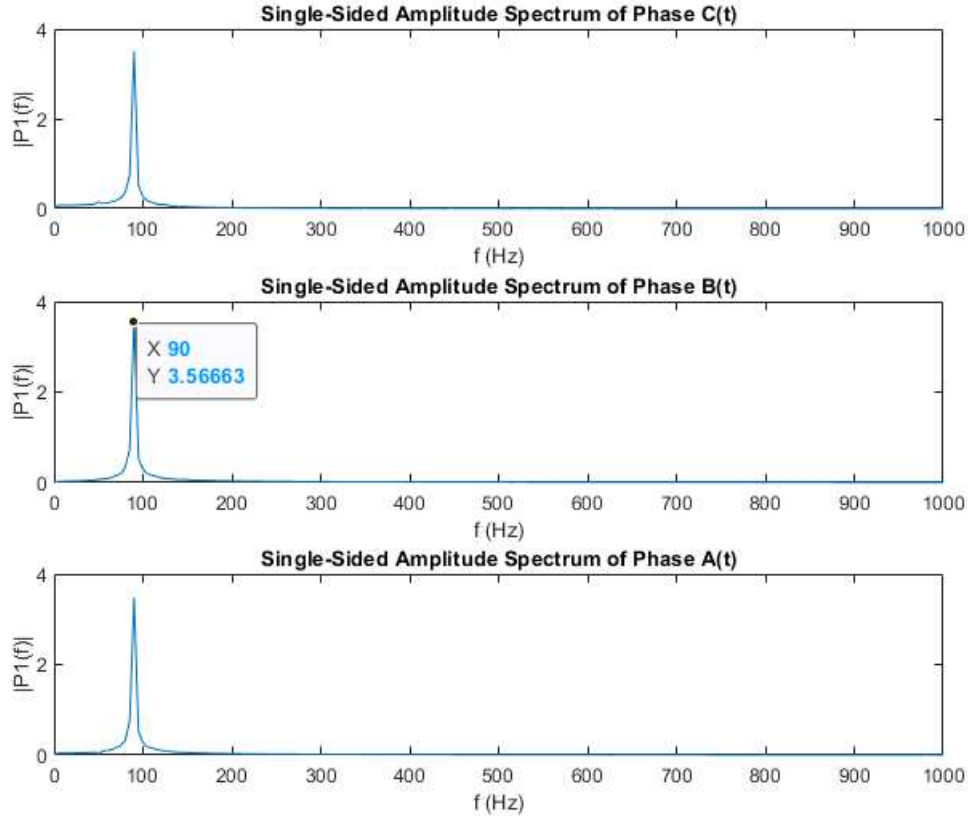


Figure 2.16: Frequency analysis of the back-EMF signals

Note that, in the calculation of the Hall offset the falling edge of the Hall signal was taken instead of the rising edge, this is due to the dependency of the rising edge to different speeds, otherwise the data cannot be compared.

Extrapolated each back-EMF peak and each falling edge of the Hall signal the Hall offset between phase *A* and *H1* signal is computed with the following formula:

$$H_{offset}[time] = mean(t_{BEMF} - t_{H1}) \quad (2.40)$$

where H_{offset} is the Hall offset and t the timestamps.

It can be expressed in degree by performing the scaling based on the signal period T :

$$H_{offset}[degree] = H_{offset}[time] \cdot \frac{360}{T} \quad (2.41)$$

In Figure 2.17 are present the Hall offsets at each speed for both the directions.

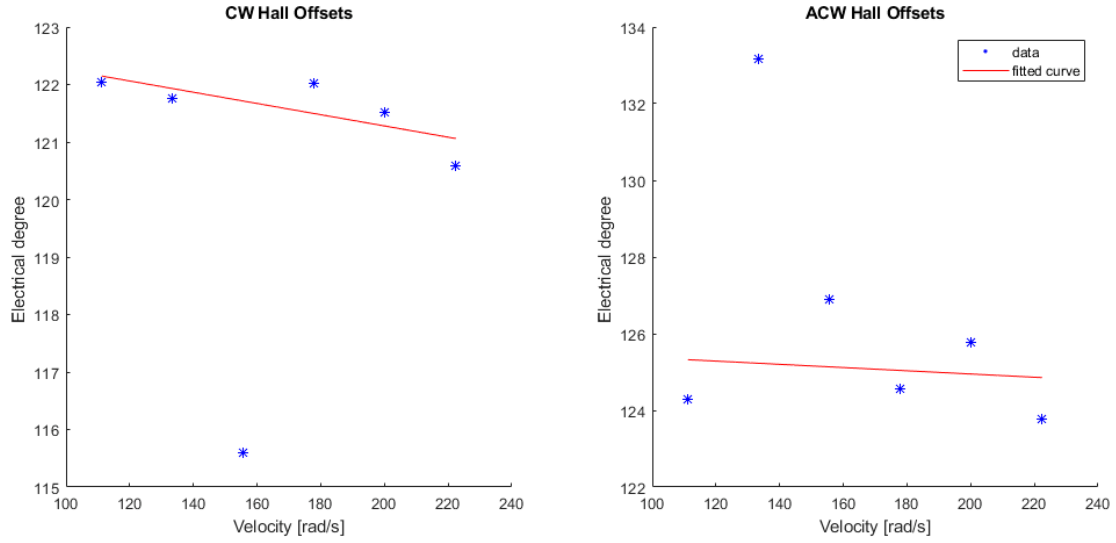


Figure 2.17: Hall offset estimation in clockwise and anticlockwise

The results are summarized excluding the clusters too far from the mean in the following table:

	CW	ACW
Hall offset	121.6	125.0
Variance	0.35	1.59

Table 6: Hall offset mean and variance

Starting from these values the Hall offset has been defined by looking the current consumption and imposing the hall offset to: $H_{offset} = 127^\circ$.

2.4.3 FLUX LINKAGE ESTIMATION

In a PMSM, the flux linkage can be obtained by the following two methods:

1. Calculation of the flux linkage according to the voltage constant of the machine.
2. Measurement of the flux linkage with the no-load test

From the voltage constant K_E declared in datasheet Table 2.4 can be compute the flux linkage, reported in the same table, using the formula:

$$\Lambda_{pm} = \frac{K_E}{N \cdot 1000} \frac{60}{2\pi} \quad (2.42)$$

While with the no load test, it can be performed by measuring the peak voltage V_{peak} , done in the hall offset estimation and through the formula 2.43 can be obtained the flux linkage of the permanent magnets.

$$\Lambda_{pm} = \frac{V_{peak}}{\omega_{me,noload}} \quad (2.43)$$

This operation must be done for all the phases and for different velocities for have a reliable estimation, the results are showed graphically in Figure 2.18, while the data are reported in Table 2.6

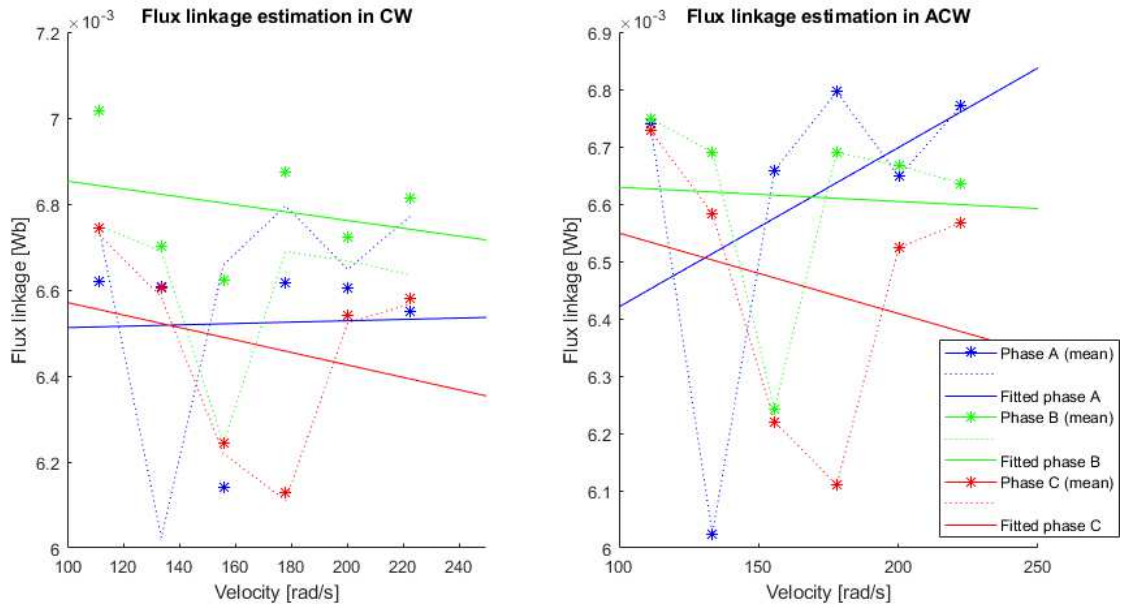


Figure 2.18: Flux linkage estimation in clockwise and anticlockwise

Looking the table the speed references used to drive the DC motor are particular, since they derives from the percentage of PWM. Starting from a 50 % of PWM and performing steps of 10 % these references can be obtained using a conversion factor $g = 222.4136$. Then, the mechanical velocity can be obtained using the formula: $\omega = PWM [\%] \cdot g$.

The flux linkage estimated is unexpected quite different with respect to the value reported in the datasheet, this could be due to some previous demagnetization or other problems. Nevertheless the value from the estimation will be used for the simulation stage.

Velocity [rad/s]	Flux Linkage [Wb]					
	$\Lambda_{mg,A}$		$\Lambda_{mg,B}$		$\Lambda_{mg,C}$	
	CW	ACW	CW	ACW	CW	ACW
111	0.0066	0.0067	0.007	0.0067	0.0067	0.0067
133	0.0066	0.0060	0.0067	0.0067	0.0066	0.0066
155	0.0061	0.0067	0.0066	0.0062	0.0062	0.0062
177	0.0066	0.0068	0.0069	0.0067	0.0061	0.0061
200	0.0066	0.0066	0.0067	0.0067	0.0065	0.0065
222	0.0065	0.0068	0.0068	0.0066	0.0066	0.0066
Mean	0.0065	0.0066	0.0068	0.0066	0.0065	0.0065
Mean CW	0.0066					
Mean ACW	0.0066					

Table 2.6: Flux linkage values

3

The Overall System

The chapter is formulated to give a description of the overall system and the devices used for develop it. In the section will be obviously excluded the explanation of the motor, extensively discussed in the previous chapter.

The system will be analyzed in two different parts: an electric subsystem, where the used control unit and motor driver expansion board are presented, including a description of the used modulation algorithm and current sensing circuit.

The remaining part of the system is composed by the mechanics, where will be seen the CAME ATS device.

3.1 SYSTEM COMPONENTS

Dealing with an automation system forces to have a knowledge in different topics: from the electronics and computer engineering to the mechanics. Then, several types of components are required to build this system.

In general, the development of an electric driven gate includes two main blocks: an electric part responsible of movement, safety, power management, control and a mechanical part necessary to transmit the movement to the load, implement mechanical limit switches and obviously give the gate structure.

The devices used for the gate drive prototype are the following:

- STM32 dynamic efficiency MCU: STM32F401RE
- Low-Voltage BLDC motor driver expansion board: X-NUCLEO-IHMo8M1
- SPM motor from Electro-Parts with three digital hall sensors (Allegro microsystems: A3290)
- Gearbox with endless screw.
- Swing gate

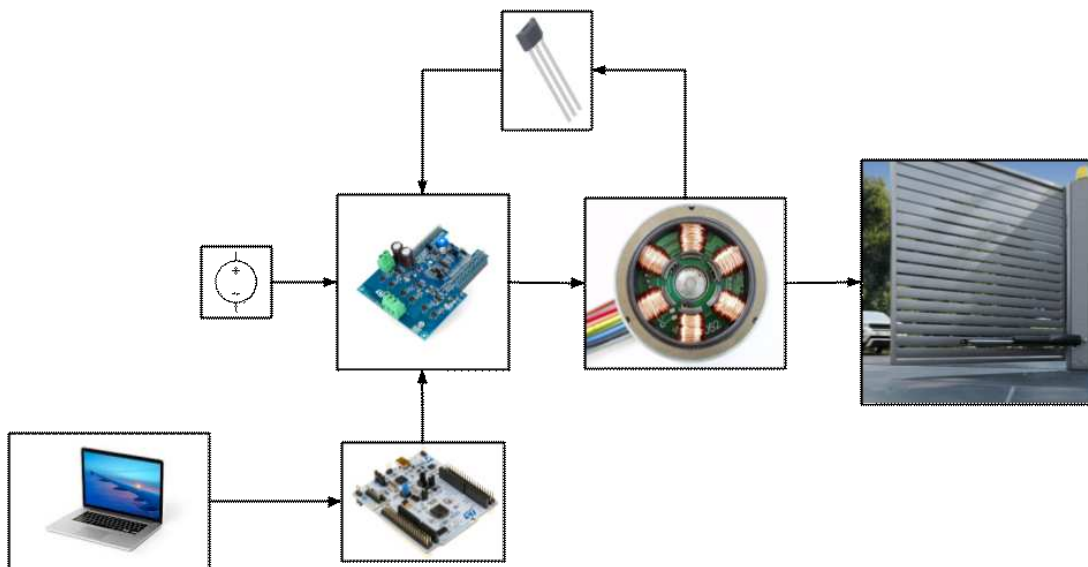


Figure 3.1: Image of the hardware system used

3.2 ELECTRIC PARTS

A generic motor control system can be basically schematized as the arrangement of three main functional blocks: the control block, the motor driver block and the motor.

The motor has already been discussed above, so only the control unit and the motor drive unit will be explored.

3.2.1 CONTROL UNIT

The control board choice fell on the STM32F401RE devices which include all the functions needed to be flexible in the control and code design.

These devices are based on the high performance ARM®Cortex® -M4 32-bit RISC core operating at a frequency of up to 84 MHz. Its Cortex®-M4 core features a floating point unit single precision which supports all ARM single-precision data-processing instructions and data types. This allows to have a more fluent and flexible code avoiding scaling into 16 or 32 bit integer variables.

The micro-controller also implements a full set of digital signal processor instructions and a memory protection unit which enhances application security. The STM32F401xD/xE incorporate high-speed embedded memories (512 Kbytes of Flash memory, 96 Kbytes of SRAM), and an extensive range of enhanced I/Os and peripherals.

All devices offer one 12 bit ADC, needed for the feedback loop, a low power RTC, six general purpose 16 bit timers, including one PWM timer for motor control and two general-purpose 32 bit timers. They also feature standard and advanced communication interfaces.

Associated with the board there is attached a support for the debug mode which is equipped with a serial wire debug (SWD) and a JTAG interface, this part is connected through USB to the computer that gives 5 V power supply and the possibility to program the board [17].

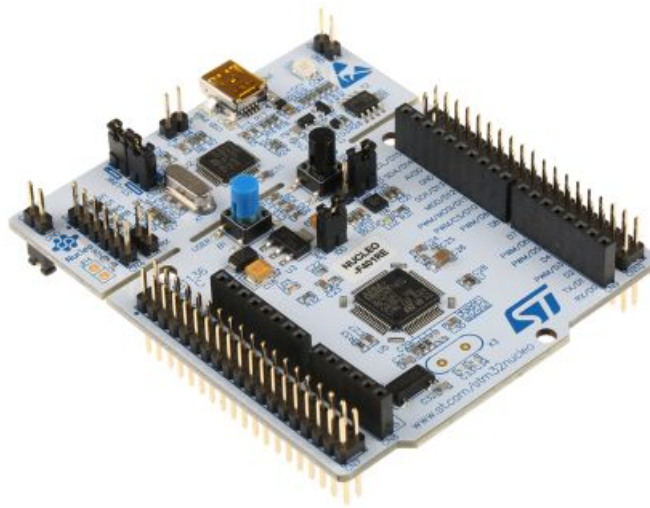


Figure 3.2: Image of a STM32F401RE control board

The company's future goal will be the transition from a Cortex®-M4 to a Cortex®-M0 or M0+.

The last MCU is already implemented in the CAME applications and tested. Furthermore, the Mo is smaller, less expensive and with the best energy efficiency, but has lower power.

3.2.2 MOTOR DRIVE UNIT

For the motor control has been adopted the evaluation board X-NUCLEO-IHMo8M1, in Figure 3.3. This board has many features and allows the user to test and design different implementations without having to redesign the schematics and the hardware.

Remember that, these board are only for design purpose, while for the production is necessary to develop an accurate and adequate board. The objective of the thesis concerns the firmware and the control therefore these boards represent a good first prototype.

The X-NUCLEO-IHMo8M1 is a three-phase brushless DC motor driver expansion board based on the STL220N6F7 STripFET™ F7 Power MOSFET for STM32 Nucleo. It provides an affordable and easy to use solution for driving three-phase brushless DC/PMSM motors.

The driver IC used on this expansion board is the L6398 single chip half bridge gate driver for the N-channel power MOSFET. The combination of L6398 gate driver plus STL220N6F7 power MOSFET forms the inverter part, so the high current power platform for the BLD-C/PMSM motor. The maximum delivered current is $15 A_{rms}$ but there is also an overcurrent detection and protection with a $30 A_{peak}$.

The nominal operating voltage can go from $10 V$ to $48 V$ DC and the operating frequency can be selected by firmware. In this case $36 V$ will be used as nominal operating voltage and a frequency of $16 kHz$.

For the velocity and feedback position control the board implements an Hall sensor circuit that can receive the digital signals through GPIO pins.

The board is able to manage FOC either with sensors or sensorless thanks to the current reading circuit that can be performed in a single shunt resistance configuration, in three shunt resistance configuration or with Insulated Current Sensors (ICS).

As already said, the thesis deals only with FOC and sensorless algorithm even if the 6-step control could be executed with the evaluation board. All the settings can be easily configured by putting some jumpers or performing some little changes in the circuit. The procedures are guided by a user manual [18] very useful for getting started. Furthermore the manual contains the circuit schemes and pin out.

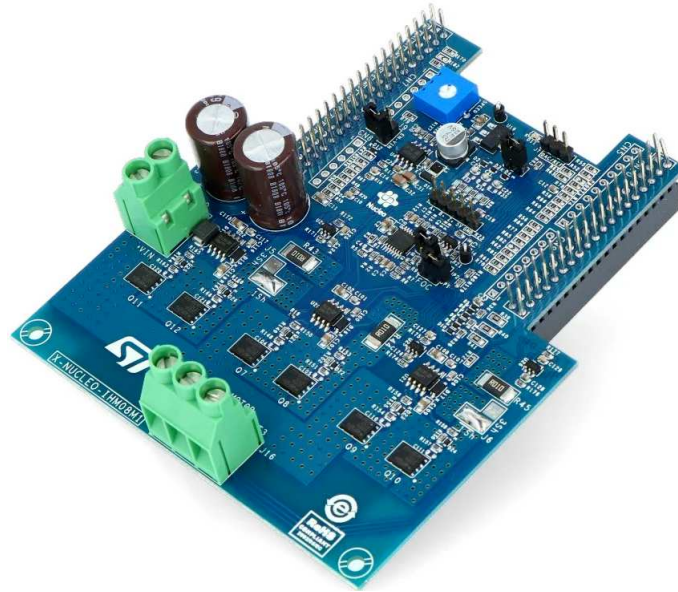


Figure 3.3: Image of the IHM08M1 motor drive board

INVERTER STAGE

The inverter is a DC/AC power converter that converts a direct voltage source into a typically sinusoidal alternating voltage whose amplitude and frequency can be adjusted.

In this case, a three-phase inverter is used with three single-phase half-bridges that make a total of six switches. Each phase of the motor is connected to an inverter leg that generates three out of phase voltage of $\frac{2\pi}{3}$.

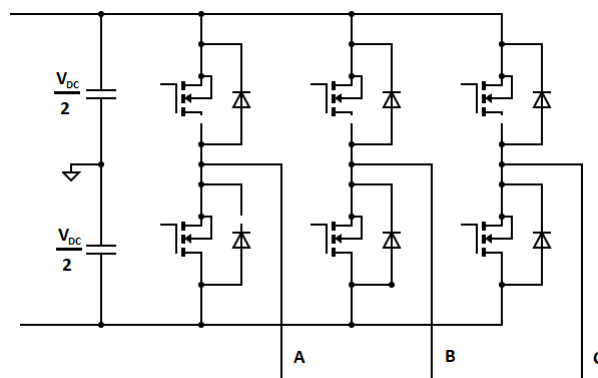


Figure 3.4: Inverter schematics

Switching voltage vector	Swiching state	Swich state	Phase voltage state
\vec{u}_{s1}	S ₁	100	ABC (+ - -)
\vec{u}_{s2}	S ₂	110	ABC (+ + -)
\vec{u}_{s3}	S ₃	010	ABC (- + -)
\vec{u}_{s4}	S ₄	011	ABC (- + +)
\vec{u}_{s5}	S ₅	001	ABC (- - +)
\vec{u}_{s6}	S ₆	101	ABC (+ - +)
\vec{u}_{s7}	S ₇	111	ABC (+ + +)
\vec{u}_{s0}	S ₀	000	ABC (- - -)

Table 3.1: Voltage vectors for two-Level VSI

The famous Pulse Width Modulation (PWM) has been adopted as modulation technique, but there exists several methods. The most widely used PWM schemes for Voltage Source Inverter (VSI) are carrier-based Sinusoidal PWM (SPWM) and Space Vector PWM (SVPWM).

The purpose of the SPWM is to maintain a sine waveform of the inverter's output voltage and in view of controlling PMSM it is widely used.

Instead, the control algorithm based on the SVPWM scheme is preferred for its easier digital realization and better utilization of the DC bus. Compared to the conventional SPWM scheme, SVPWM can increase the DC bus voltage utilization by 15% and achieve lower output harmonic distortion. The torque ripple of PMSM can be reduced with SVPWM and the machine's performance can be improved.

The SVPWM is implemented in the firmware with a PWM frequency or commutation frequency of 16 kHz. This frequency was chosen to expand standardization and avoid possible errors as it is already used in other company projects, moreover, it is much higher with respect to the motor frequency and it is not excessive to create power losses.

The two-level VSI can provide eight output voltage vectors based on the switches states (6 state vectors and 2 zero vectors). In Table 4.4 [12] is represented each switching state assigned to a precise combination of the switches, where 1 indicates the high-side switch of one phase is switched on and 0 if the low-side switch is on.

The zero voltage vectors are \vec{u}_{s7} and \vec{u}_{s0} , which can also be called as ineffective voltage vectors. The other six effective voltage vectors divide the α - β phase plane into six sectors forming a hexagon, as shown in Figure 3.5 [12].

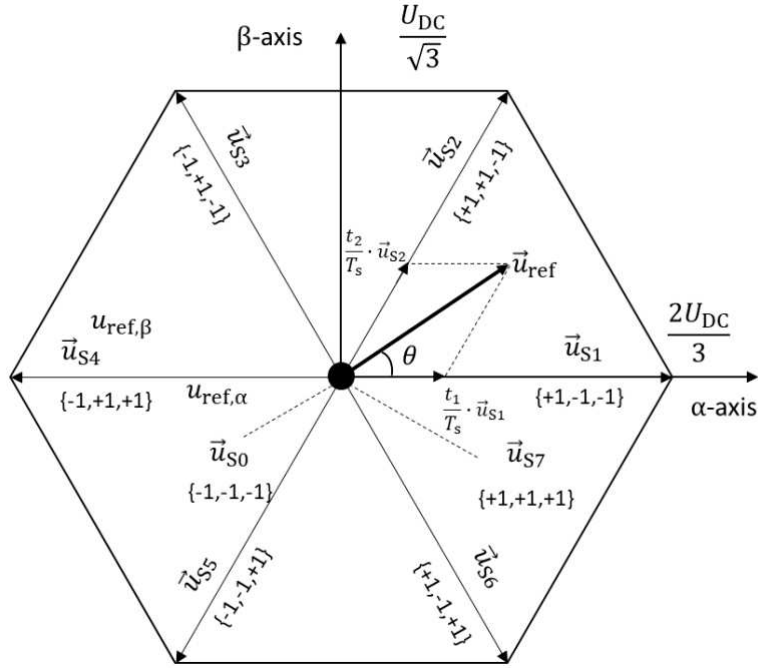


Figure 3.5: Space vector hexagon

Using this concept an arbitrary reference voltage vector \vec{u}_{ref} is constructed from the two effective switching vectors adjacent to it. Taking as an example \vec{u}_{ref} in the sector clamped by \vec{u}_{s1} and \vec{u}_{s2} the following approach is valid:

$$\vec{u}_{ref} \cdot T_s = t_1 \cdot \vec{u}_{s1} + t_2 \cdot \vec{u}_{s2} \quad (3.1)$$

where T_s is the switching frequency and t_1, t_2 are the times duration for the realization of the respectively voltage vectors. These duration can be computed in according to the hexagon with the formula:

$$\begin{aligned} t_1 &= \sqrt{3} \cdot T_s \cdot \frac{U_{ref}}{U_{BUS}} \cdot \sin\left(\frac{\pi}{3} - \theta\right) \\ t_2 &= \sqrt{3} \cdot T_s \cdot \frac{U_{ref}}{U_{BUS}} \cdot \sin(\theta) \end{aligned} \quad (3.2)$$

In general, $t_1 + t_2 < T_s$, therefore zero voltage vector \vec{u}_{s0} and \vec{u}_{s7} should be applied. The time duration of the zero voltage vectors are obtained as follows:

$$t_0 = t_7 = \frac{T_s - t_1 - t_2}{2} \quad (3.3)$$

Based on the calculated times duration for the voltage vectors and considering that one half-bridge should change its state at one point in time, the following switching sequence given in Figure 3.6 including zero voltage states can be applied to generate the reference voltage \vec{u}_{ref} and minimize the number of switching operations for the switches [12]:

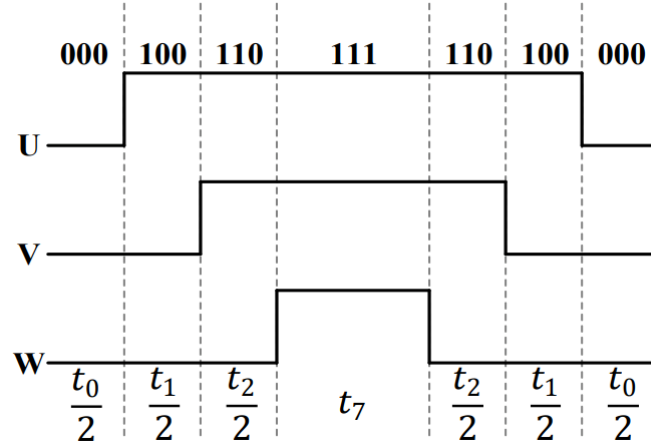


Figure 3.6: Switching patterns and corresponding switching time of sector 1

The working principle of SVPWM in other sectors is similar to that of the sector 1 shown in the Figure 3.6.

CURRENT READING CIRCUIT

Fast and accurate current sensing is required in motor control applications for minimum torque ripple and therefore minimal audible noise. Accurate current sensing is also important for getting the best dynamic motor control.

Delay in the current detection can lead to erroneous current estimates and hence distorted current waveform in a motor. This error is imported into the FOC algorithm where the electrical angle is calculated starting from the current measurements.

A conditioning network is required to measure the motor phase current. The IHMo8M1 supports three current sensing networks: ICS, three shunt and single shunt resistor.

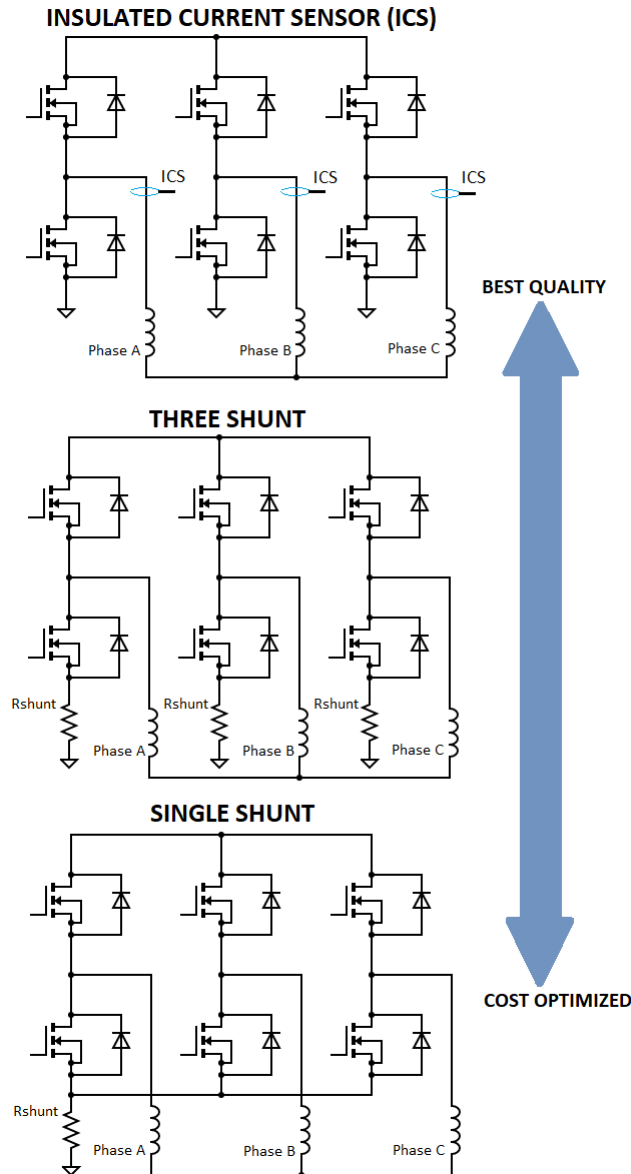


Figure 3.7: Current sensing networks and comparison

ICSs offer the best read quality with the simplest implementation, since they are used in-line, but require at least two isolated sensors which increases the costs. Low-side three shunt topology is a less expensive option and can still achieve good results using at least two shunt resistors and an operational amplifier.

In the three shunt sensing configuration is present a shunt resistor for each phase of the motor for measure the currents in the inverter legs when the low side switch is on, as in Figure 3.7.

The schematics use also a operational amplifier for observe current consumption behavior. The big difference compared to one shunt topology is that each phase can be sampled at the same timestamp, simplifying the computation and reducing noise contribution. Nevertheless, it is not necessary to sample all the three phases because with the Kirchhoff's formula: $I_a + I_b + I_c = 0$, one phase can be reconstructed from the other two.

There is a problem, due to the switching frequency of the MOSFET a noise is introduced into the sampled signal leading to an error. To solve this issue the current sampling point must be synchronized with the PWM.

The solution adopted by the STMicroelectronics is to insert two waiting time for avoid the noise contribution: the first time is called rise time T_{RISE} and it is the time that elapses between the turn on of the related low side switches to a stable value of the ADC input; the second is called noise time T_{NOISE} and it is the time necessary to avoid the sampling point in case of any power switch commutation. In Figure 3.8 [19] the signals and times are graphically shown.

For sample the signal must be chosen the correct point or the point that is less subjected to noise contribute based on the space vector sector and where the duty cycles applied to the low side switches are the highest. The ST algorithm performs also this task, applying some tricks in the firmware.

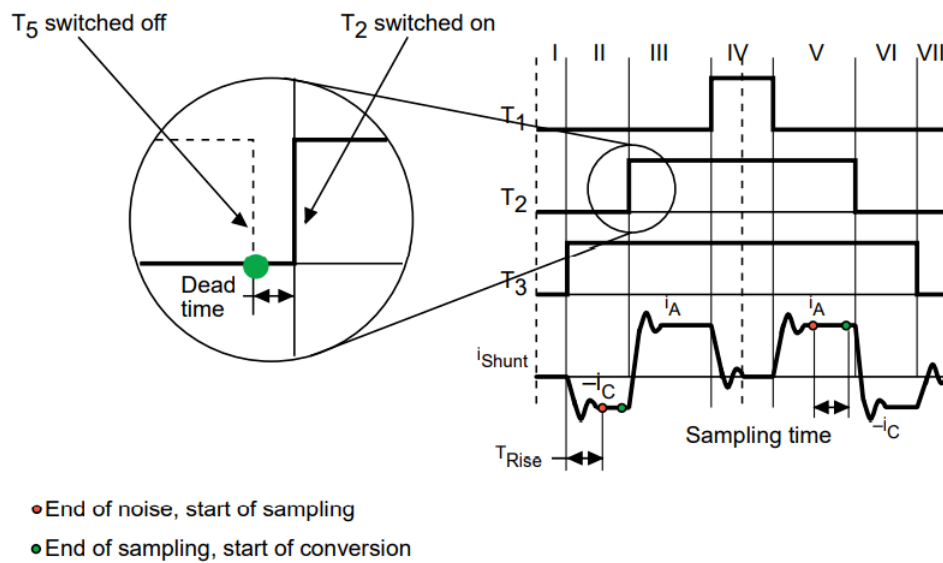


Figure 3.8: Noise parameters definitions, signal of a single shunt configuration

Typically the voltage drop across the shunt resistor is modify in amplitude and offset before entering in the ADC, this is done though a simple circuit that uses an OPAMP to adapt the

signal to the right ADC resolution, Figure 3.9 [19].

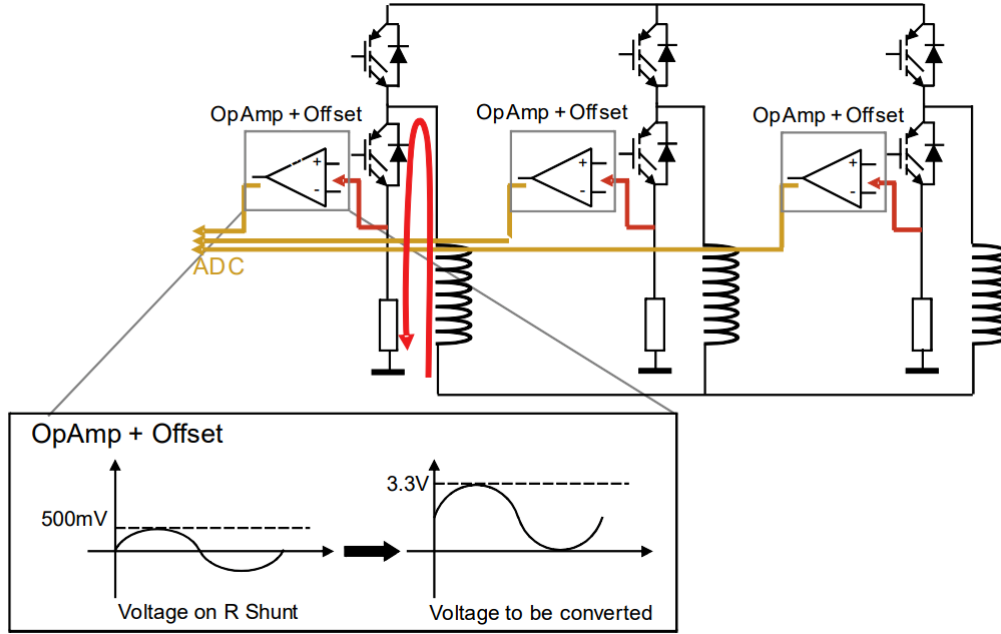


Figure 3.9: Current sensing networks in low-side three shunt scheme with operational amplifier for adapt the signal to the ADC resolution

Unfortunately, only one ADC is provided in the STM32F401RE micro-controller, so it is not possible to synchronously sample the two analog phase currents, but they can only be performed in sequence mode.

As a consequence two current samples are not simultaneous but the start of the second current sampling is delayed with respect to the first by its global conversion time. This introduces a conceptual error in the third current computation using Kirchhoff's first law and this error is reflected in the whole FOC. Anyway, the error is little and it is considered negligible in the thesis.

The ADC is configured in injected mode in order to synchronize the current sampling point with the PWM output, as done in the firmware, but it is not sufficient, the different situations that can occur depending on PWM frequency and applied duty cycles must be distinguished. The following cases are based on the hypothesis that the time needed by the ADC is less than the dead time (DT) plus the maximum between the rise time (T_R) or noise time (T_N):

$$2T_S + T_C < DT + \max(T_N, T_R). \quad (3.4)$$

Where: T_S is the sampling time of the ADC; T_C is the conversion time of ADC; T_N is the duration of the noise induced on the shunt resistor voltage by the commutation of a switch belonging to another phase.

It's possible to identify a common case for all sectors, where the duty cycles applied to phases A, B, C low side switches are larger than $DT + \max(T_N, T_R)$.

In this case, to minimize measurement errors due to the ADC calibration, always the currents of phases A and B are converted, Figure 3.10 [19].

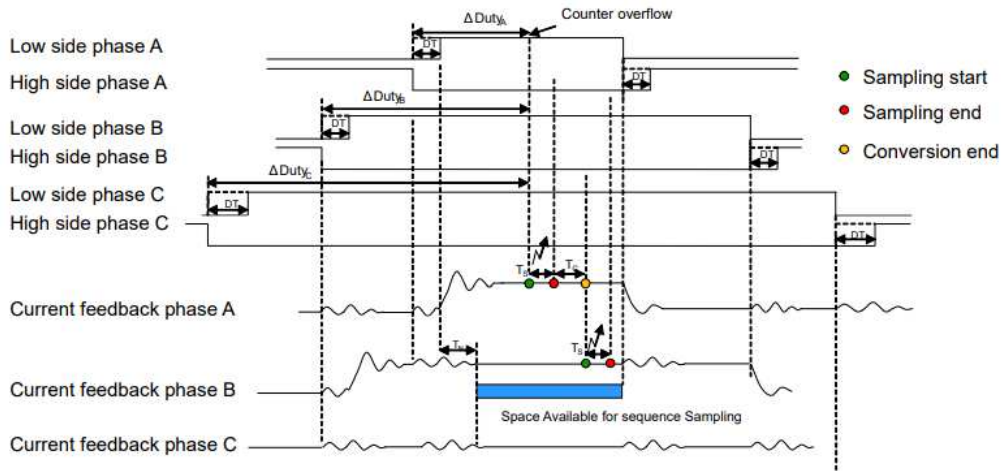


Figure 3.10: Low side of phase A, B, C duty cycle $> DT + \max(T_N, T_R)$

The following explanations refer to space vector sector 1 and it can be applied in the same manner to the other sectors.

Refer to the SDK manual for more detailed information [19].

3.3 MECHANICAL PARTS

Always in automation systems there is a mechanical part which in addition to having a structural contribution is responsible for the transmission of the motion.

The gate is obviously the main mechanical component. It can be of two main types: the sliding gates and the swing gates. The thesis will deal with the a swing gates but the only difference is limited on a mechanic point of view. In fact, both types of gates are driven with an electric motor which implies very similar firmware and electronic.

Swing gates use one or two doors mounted on a fulcrum, similar to a house door. Therefore,

it is necessary to have enough space to allow them to be fully opened.

Automatic sliding gates, use one or two leaves mounted on a track so that they can move only parallel to the perimeter partition (wall, gate, hedge, ...); this is a more practical solution as it requires less space for its installation.

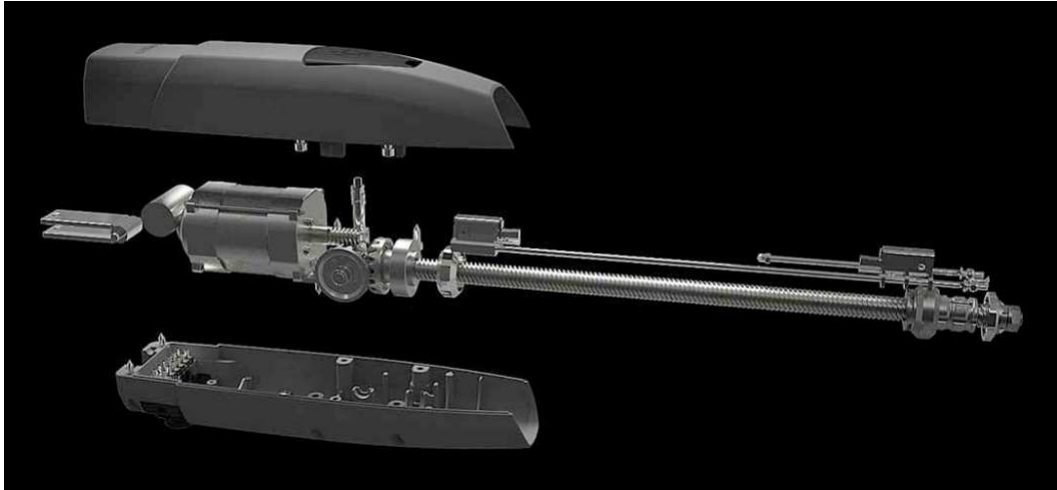


Figure 3.11: Exploded view drawing of the CAME ATS solution



Figure 3.12: Gearbox drawing of the CAME ATS device

The swing gate drive adopted is called CAME ATS and it is shown in Figure 3.11 [20]. It is a swing gate automation that can be installed in many context. Thanks to the telescopic exten-

sion it is able to adapt perfectly to different depths of the pillars.

The motion transmission is guaranteed through a gearbox and a endless screw suitably designed by the mechanical R&D department based on the load characteristic curve and the motor characteristic curve. From the figure can be notice that the transmission is axial so that the motor body does not protrude from the body, for a slender and elegant shape.

The gearbox is used for adapt the velocity and the torque of the load to the motor and its transmission ratio is defined as:

$$k_g = \frac{\omega_r}{\omega_m} \quad (3.5)$$

where ω_r indicates the crankshaft velocity on the engine side, while ω_m is the crankshaft velocity on the load side.

A $k_g < 1$ is adopted for reduce the velocity of the load and to increase the torque, following the equation:

$$T_r = \eta k_g T_m \quad (3.6)$$

where T_r is the torque on the engine side, while T_m is the crankshaft velocity on the load side and η is the power efficiency coefficient for gearbox losses.

A transmission ratio of 1 : 28 will be considered in the thesis.

Attached to the gearbox is present a endless screw, necessary for the telescopic arm mounted parallel to the gate.

Are present other components for add features and functionality to the device, like: a mechanical block for decoupling instantly the load, a resistant case in aluminum and electronic limit switches. The last feature may be adapted acting on two screws at the end of the telescopic arm.

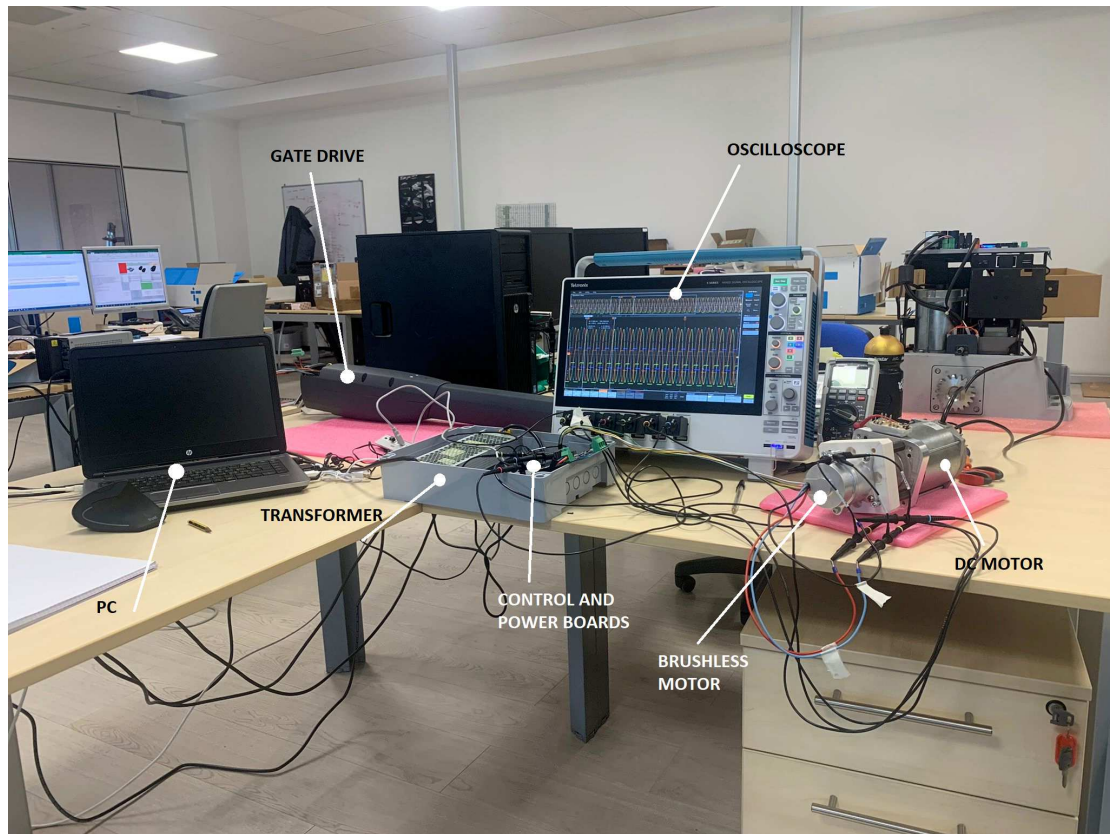


Figure 3.13: Experimental set-up

4

Firmware Architecture

One of the project core is the firmware implementation of a brushless motor control for gate drives. This chapter explains all the main algorithms implemented in the micro-controller to have a functional and efficient system.

The code has been written in the firmware development kit (FDK) of the company and a preliminary overview is addressed to give the importance of having a standardize and event based structure.

The coding does not start from scratch but to obtain a fast and optimized project was taken as reference the code generated by the STMicroelectronics application Motor Control Workbench (MCSDK) and obviously the code of the CAME R&D department.

Starting from the speed control loop of CAME, the features of the MCSDK have been added. In particular, the work is focused on the FOC algorithm.

The code was written in C using the STM32CubeIDE software which is an excellent learning tool, it's free and simple.

4.1 CODE STRUCTURE

The project structure derives from the company's firmware development kit.

The FDK is now a necessary and efficient tool designed to help embedded system designers quickly and easily test firmware for their specific device. It promotes the standardization by allowing to change part of the code in a simple and fast way with different electronic boards

and so different models and brands.

The FDK can be classified as either time based or event based systems. In the time-based systems, samples are generated periodically leading to a possibly large number of computations that do not provide new information [21].

However, the CAME FDK is event based, computations are triggered only when certain events occur, having the potential to change the state of the system. This in turn will improve resource efficiency.

Each occurrence of the event can have an event header and an event body. The event header describes the occurrence event with the help of the event specification ID and an event time stamp, while the event body describes what happened.

Events can be classified as periodic, stochastic or sporadic depending on their pattern of arrival. Periodic events occur at regular intervals of time and therefore are predictable about their occurrence. Events are stochastic if they occur according to a probabilistic distribution model. Sporadic events are unpredictable and follow no pattern.

An event based architecture includes production of events, detection of events and response to events. Such architectures are extremely loose coupled and highly distributed. The producers of the event are not aware of the consumers of the event or how they are processed.

After generation each event is processed through a scheduler that is responsible for deciding which task should be performed at any particular time, based on the event priority. Associated with the scheduler, a real-time operating system (RTOS) can be inserted. RTOS implements a sophisticated scheduler for managing systems in real time.

In the thesis the RTOS is not foreseen, since the thesis develop only a prototype, but it could be in the business object. Note that, there is no need to rewrite the firmware if the RTOS have to be inserted, because FDK's abstraction could handle several RTOS.

The code is structured in two projects: the Application and the Board Support Package (BSP). The Application project is the code loaded and executed in the MCU, while the BSP is a supporting code, not executed, of a specific implementation for a given board that it conforms to the given operating system. It is commonly built with a boot-loader that contains minimal peripheral support to load the operating system and device driver for all peripherals on the board. Basically, the BSP is the FDK and is used to abstract the hardware through a interface logic.

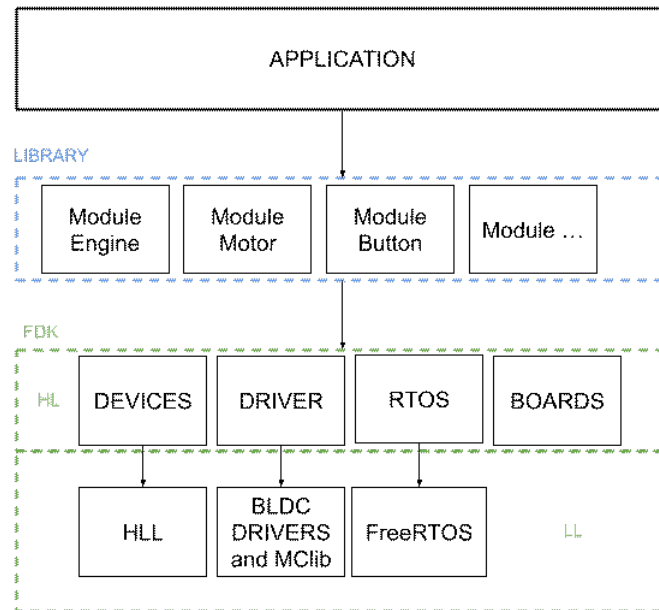


Figure 4.1: Simplified FDK simplified architecture of the ATS project

4.2 MOTOR APPLICATION

The Application is the program executed, which may vary according to the product considered. It is structured with the most high level of abstraction for simplify the work of the coder.

The first thing executed in the Application project is the *main* file, called from the start-up file in the BSP.

The *main* file is required for start and includes: the initialization of the base timer with: a defined priority, a counter frequency and a sample frequency; the *PowerInit* for: create the application threads, force the power on and start the application; the RTOS start for handle the task events.

In addition to the main file is present a *Powermanagement* file which includes several function, like the before *PowerInit*.

These two files represents the highest level of abstraction and they are able to communicate only with the library, as pictured in Figure 4.1.

The library consist of several modules that can be exported and reused for other projects. In this case only the module for the button is used, but many other features such as: encoders, ADC management and so on can be added. These modules allow to use simple and already

implemented functions for many projects avoiding to rewriting them, wasting time and using a unique and optimized code.

The core of the application is the engine folder that comprises an *engine*, a *motor* and a *Fuzzi PI* file. For the last file code and details see section 2.3.2.

The *engine* file comprises: a *Init* function to initialize the motor settings and create a base tick function; a one millisecond base tick function that handles close or open events and calls the speed loop function; a routine called periodically for perform the opening and the closing event requests via a simple button and finally the speed loop function that calls the Fuzzy PI. A lower level of abstraction is introduced by the motor file which includes a state machine called by the base timer at a rate of one milliseconds and some interface function used as wrapper for the High Level FDK (HL-FDK).

The Application is structured such that the only function needed are the one of the HL-FDK, as from abstraction purpose.

4.2.1 STATE MACHINE

State	Description
INITIALIZATION	All the parameters are initialized with their default values, the timers and ADC are initialized and the scheduler is started.
START and WAIT CALIBRATION	State where the offset of motor current measurements will be calibrated and the code waits the end of the calibration.
IDLE	The motor is not spinning but is ready to start, it is waiting to receive a starting command.
ERROR	Persistent state if an error has occurred.
PREPARE to RUN and WAIT	Check if calibration already done and enables TIM channels, otherwise it performs the calibration and wait.
RUN	Persistent state with running motor and it awaits a command for stop the motor.
STOP and WAIT	Persistent state that performs the motor stopping and after it waits that the velocity is null for move to the IDLE state.

Table 4.1: Motor state machine

The state machine is used for control the different phases that the motor can have. The tasks executed by the motor and the functions that it can perform depends on the current state of its state machine.

In Figure 4.2, the complete state machine is presented, where the states are indicated in blue rectangle while the eventual transactions are marked with an arrow. A brief explanation of each state is reported in Table 4.1.

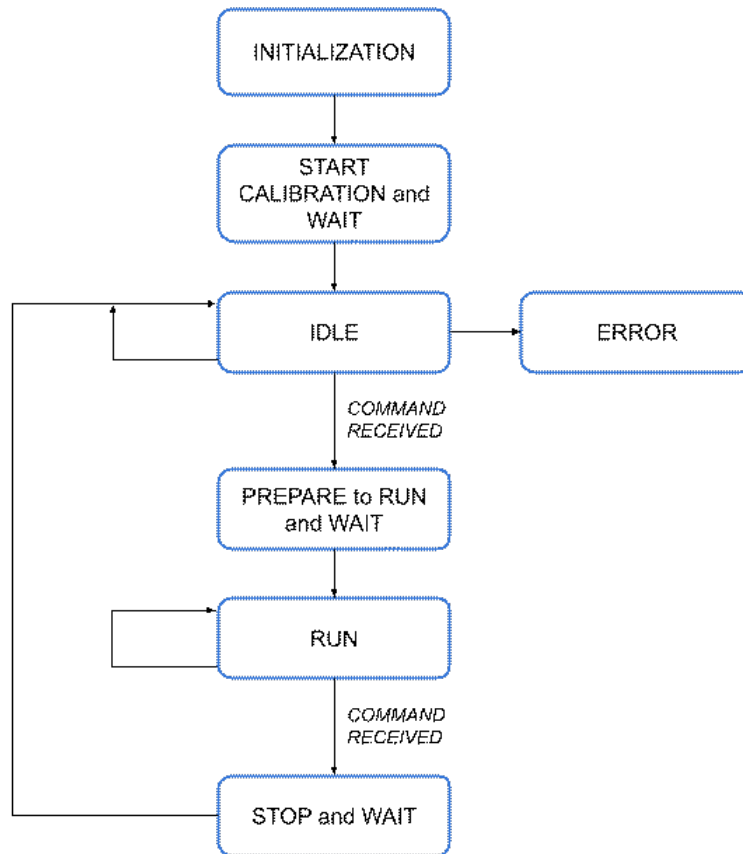


Figure 4.2: Motor state machine

4.2.2 SPEED LOOP FUNCTION

The speed loop function is called periodically thanks to a tick function managed by the base TIMER 3. The speed loop function will be called every 1 *ms*, but before calling the speed loop there is a switch-case for handle the different states of the engine, different from the motor states cited before.

The engine states are: closing, opening and stopping and they are all determined based on the event ID generated. In our case the events are produced by a button, but could be also a transmitter.

Opening and closing states will both perform the same switch-case task, they make: a foreplay timeout necessary to wait the end of the calibration, update the current speed, calculate the reference speed, call the speed loop, perform a saturation of the current reference output and at the end set the reference output current I_q .

Instead, the stopping state computes a decreasing ramp thanks to a filter for the reference current in order to progressively reduce the speed.

The reference current will be handled by the FDK where is present the FOC algorithm.

The pseudo-code is reported in Algorithm 4.1.

Algorithm 4.1 Tick function

```

if (Engine_State = OPENING || Engine_State = CLOSING)
    /* Wait */
    /* Update speed */
    /* Compute reference speed */
    /* Call speed loop */
    /* Output saturation */
    /* Set reference current output */
    if (Engine_State = STOPPING)
        /* Compute reference ramp down current */
        /* Check saturation */
        /* Set reference current output */
        if (CURRENT = 0)
            /* Motor stop */
            /* Motor_state = IDLE */
        end if
    end if
end if

```

The speed loop is very short, it saves the target speed and current speed passed by the tick function, perform a limitation in the variation of the PI to avoid big problems in case of instability, calculates the Fuzzy PI and returns the saturated reference current. The pseudo-code is reported in Algorithm 4.2.

Algorithm 4.2 Speed loop

```
/* Check PI variation */  
/* Start PI timeout */  
/* Call Fuzzy PID */  
/* Return reference current output */
```

4.3 BOARD SUPPORT PACKAGE

The BSP or the so called FDK is a supporting project. It is divided into two level of abstraction the High Level (HL) and the Low Level (LH). The Application will use only the HL-FDK layer that includes: the drivers, the RTOS, the board configurations and the devices. While the LL-FDK is used only by the HL-FDK and it is customized based on the specific micro-controller. Drivers are an abstraction interface of the micro-controller's hardware peripherals that does not support necessarily all the board functionality, but only the strictly project related functions. Usually, the drivers include basic functions to handle: timers, ADCs, interrupts, flash, UARTs, SPIs and so on. They can be modified to add some functionality.

The thesis have augmented the drivers adding a section called *driver_bldc* that includes the motor control library with the main functions taken from the MCSDK of ST. These functions are: feedforward, encoder management, flux weakening, Hall management, MTPA, current/temperature/voltage control, pid regulator, sensorless algorithm $d-q$ (LO + PLL and LO + CORDIC), current sampling algorithm and many other features. Only a limited part of these algorithms will be considered.

Inside the *driver_bldc*, beyond the motor control library, there are other files which constitute the FOC algorithm and the interface with the Application.

The file necessary as interface for the Application project contains all the function called by the wrapper functions of the *motor* file, as: the initialization, the de-initialization, the *bldc_open* function for start the initialization, the *bldc_close* function, the *bldc_start* function and so on.

In the same file are present also the Interrupt Request Handler (IRQ) of the ADC, the TIMER 1 update and break event, the ADC initialization and the TIMER initialization called by the *bldc_open* function in the INITIALIZATION motor state.

The other files concern: the FOC control with its math functions, the SVPWM and current feedback, the hall sensor management and other files that includes fixed parameters.

4.3.1 PERIPHERALS

All the peripherals used in the project are schematized in Table 4.2.

Peripherals	Description
TIMER 1	The first three channels are used for PWM generation. The fourth channel is used for start the ADC conversion.
TIMER 3	It is used as base timer in the Application project for manage events and tick functions, like the speed control loop.
ADC	It is configured in injected mode for sample the phase currents, triggered by channel four of TIMER 1 for synchronize the start of the conversion sequence with the PWM.
Button	GPIO in input configuration and pull-up. It is used for: start, stop and change the velocity direction and amplitude.
Hall Sensors	GPIO in alternate configuration and high speed operation used for compute the electrical angle and velocity.

Table 4.2: Peripherals used in the project

Other peripherals can be added to increase complexity and security, for instance may be interesting to insert a DAC for debugging, a UART for printing errors and results in the terminal monitor and another ADC for bus voltage sensing and temperature protection.

ANALOG TO DIGITAL CONVERTER

The ADC is used to synchronize the current sample point with the PWM output using an external trigger. To do this, the control algorithm uses the fourth PWM channel of TIMER 1 to synchronize the start of the conversion sequence. See Figure 4.3.

Considering only the rising edge, the sampling point must be set before the counter overflow, (when the counter TIMER 1 coincides with the value of the OCR4 register during the up counting) when this value is reached the A/D conversion sequence is started. The phase current values are acquired and saved in order to execute the FOC algorithm.

Finished the FOC task, the next value to be loaded into the OCR4 register is calculated to set the sampling point for the next PWM period, while the ADC is configured to sample the correct channels.

To perform this task, the ADC must be initialized in a appropriate manner, look at the Table 4.3 for the schematized configurations.

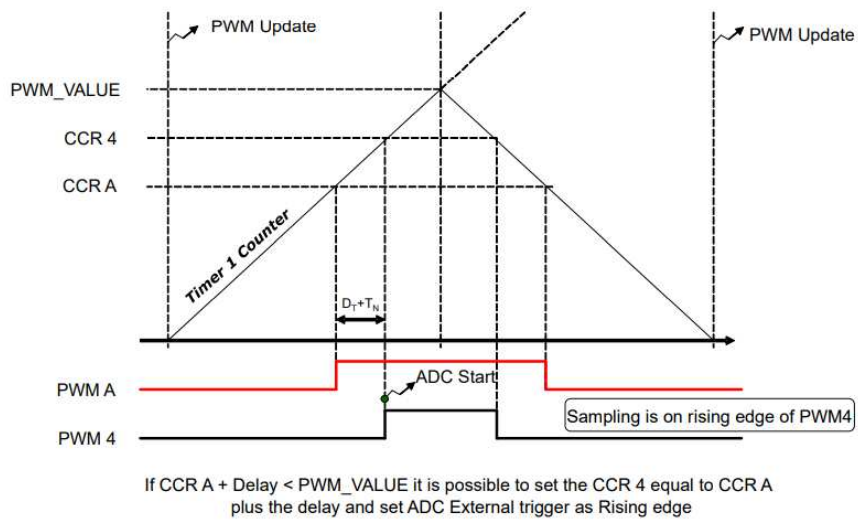


Figure 4.3: Synchronization strategy between TIMER 1 PWM output and ADC considering the rising edge

Configuration	Value
Instance	ADC 1
Clock Prescaler	4
Resolution	12 Bit
Scan Conversion Mode	Enable
Data Align	Left
ADC Channel 0, 11, 10	Injected Channels
ADC Channel 0	Rank 1
ADC Channel 10	Rank 2
ADC Channel 11	Rank 3
Injected Nbr. of Conversions	3
Inj. Sampling Time	15 Cycles
External Trigger Edge	Rising
Ext. Trig. Inj. Conv.	TIMER 1 CC4

Table 4.3: ADC Configuration

This is not enough, an interrupt request handler must also be implemented to handle the TIMER 1 capture compare (CC) 4 trigger.

The routine is implemented with a zero priority, the highest priority possible, and it is responsible of starting the high frequency task. This task is called periodically and basically runs the FOC inner feedback loop.

TIMERS

Two timers with different functions are used in the project, the first is implemented in the Application project and it is used as a base timer to manage: the events and the medium frequency task, which consist in the state machine and in the speed loop.

This timer was mentioned above and it is the TIMER 3. Here below the configuration details where the counter frequency is the clock frequency at which the peripheral timer will run and the sample frequency is the frequency at which the peripheral timer will generate a periodic event. Table 4.4.

Characteristic	Value
Instance	TIM 3
Priority	2
Counter Frequency	1000000 [ms]
Sample Frequency	2000 [ms]

Table 4.4: Timer 3 configuration

The other timer, also previously mentioned, is the TIMER 1, it is needful for the PWM generation but also for triggering the ADC IRQ.

As for the ADC this timer has a IRQ Handler routine both for an update event, when the timer count reaches a defined register value, and for a break event.

The update event interrupt is used to synchronize the ADC acquisition and for electrical angle calculation and update.

The break event interrupt is used to stop the PWM generation by putting the system in a safe state in case of overcurrents or other errors.

The configurations are shown in the Table 4.5

Configuration	Value
Instance	TIM 1
Prescaler	0
Counter Mode	Center Aligned
Period	2625 clk cycles
TIM Clock Division	2
TIM Channel 1, 2, 3	
Output Compare Mode	PWM 1
PWM frequency	16000 Hz
Pulse	0
OC Polarity	High
OCN Polarity	Low
OC Idle State	Reset
OCN Idle State	Set
TIM Channel 4	
Output Compare Mode	PWM 2
Pulse	2624
OCN Idle State	Reset
Break Dead Time Conf.	
Off State Run Mode	Enable
Off State IDLE Mode	Enable
Lock Level	1
Dead Time	67 cycles
Break State	Enable
Break Polarity	Low

Table 4.5: Timer 1 configuration

4.3.2 FOC ALGORITHM

The FOC algorithm performs the steps analyzed and graphically represented in the subsection 2.3.1.

The FOC task is called in the ADC IRQ and it is triggered by the TIMER 1 channel 4 update event. After the FOC calculations the code calls the SVPWM function that computes the duty

cycles for set the phase voltages and the ADC sampling point.

The C code is shown below:

```

1  qd_t Iqd , Vqd;
2  ab_t Iab;
3  alphabeta_t Ialphabeta , Valphabeta;
4
5  /* El. Angle Acquisition */
6  int16_t hElAngle = Get_Elangle();
7
8  /* Current Acquisition */
9  PWMC_GetPhaseCurrents(pwmcHandle[M1] , &Iab);
10
11 /* Math Transformation */
12 Ialphabeta = MCM_Clarke( Iab );
13 Iqd = MCM_Park( Ialphabeta , hElAngle );
14
15 /* PI Controller */
16 Vqd.q = PI_Controller(pPIDIq , (int32_t)(FOCVars.Iqdref.q) - Iqd.q);
17 Vqd.d = PI_Controller(pPIDId , (int32_t)(FOCVars.Iqdref.d) - Iqd.d);
18
19 /* Math Transformation and Limitation*/
20 Vqd = Circle_Limitation(pCLM[M1] , Vqd);
21 Valphabeta = MCM_Rev_Park(Vqd , hElAngle);
22
23 /* Call PWM Function */
24 hCodeError = PWMC_SetPhaseVoltage(pwmcHandle[M1] , Valphabeta);
25
26 return( hCodeError);

```

4.3.3 SVPWM CODE

The SVPWM algorithm is invoked immediately after the FOC algorithm through the PWMC-SetPhaseVoltage function. This algorithm receive as input the reference voltages in the α - β reference frame $u_{\alpha\beta}$ and computes the three phase voltages u_{abc} using the following equations:

$$U_{\alpha} = \sqrt{3} \cdot T_{PWM} \cdot V_{\alpha}, \quad U_{\beta} = -T_{PWM} \cdot V_{\beta}; \quad (4.1)$$

$$X = U_\beta, \quad Y = \frac{U_\alpha + U_\beta}{2}, \quad Z = \frac{U_\beta - U_\alpha}{2}; \quad (4.2)$$

using the same notation of ST the T_{PWM} is the PWM period = 16 kHz; X, Y and Z are the three phase voltages and U_α, U_β the inverter scaled voltages in the stationary reference frame. The equations are different from the classical representations as are taken from the STMicro-electronic user manual [19]. They obtain the same result but using different notations determining mismatching in the Park and Clarke transformations.

On the basis of the phase voltage values calculated earlier it is possible to identify the sector of the space vector as shown in Table 4.6.

Sector	$Y < 0$			$Y \geq 0$		
	$Z < 0$	$Z \geq 0$		$Z < 0$		$Z \geq 0$
		$X \leq 0$	$X > 0$	$X \leq 0$	$X > 0$	
	V	IV	III	VI	I	II

Table 4.6: Sector identification

Once the sector has been defined, the duty cycles of each timer channel can be computed adopting the Equation 4.2.

The PWM is configured in center-aligned determining a phase voltage centered at 50% of the duty cycle. Then, the duty cycles and the current sampling point must be loaded into the appropriate TIMER registers to be performed in the next cycle.

```

1  uint16_t returnValue;
2  int32_t wX, wY, wZ;
3  int32_t wUAlpha, wUBeta;
4  int32_t wTimePhA, wTimePhB, wTimePhC;
5
6  /* Compute phase voltages */
7  wUAlpha = Valfa_beta.alpha * (int32_t)pHandle->hT_Sqrt3;
8  wUBeta = -(Valfa_beta.beta * ((int32_t)pHandle->PWMperiod)) * 2;
9  wX = wUBeta;
10 wY = (wUBeta + wUAlpha) / 2;
11 wZ = (wUBeta - wUAlpha) / 2;
12
13 /* Sector calculation from X, Y, Z */
14 if (wY < 0)

```

```

15 {
16     if (wZ < 0)
17     {
18         pHandle->Sector = SECTOR_5;
19         wTimePhA = (((int32_t)pHandle->PWMperiod) / 4) + ((wY - wZ) /
20 (int32_t)262144);
21         wTimePhB = wTimePhA + (wZ / 131072);
22         wTimePhC = wTimePhA - (wY / 131072);
23
24         pHandle->lowDuty = (uint16_t)wTimePhC;
25         pHandle->midDuty = (uint16_t)wTimePhA;
26         pHandle->highDuty = (uint16_t)wTimePhB;
27     }
28     else /* wZ >= 0 */
29     {
30         pHandle->Sector = SECTOR_4;
31         wTimePhA = (((int32_t)pHandle->PWMperiod) / 4) + ((wX - wZ)
32 / (int32_t)262144);
33         wTimePhB = wTimePhA + (wZ / 131072);
34         wTimePhC = wTimePhB - (wX / 131072);
35
36         pHandle->lowDuty = (uint16_t)wTimePhC;
37         pHandle->midDuty = (uint16_t)wTimePhB;
38         pHandle->highDuty = (uint16_t)wTimePhA;
39     }
40     else /* wX > 0 */
41     {
42         pHandle->Sector = SECTOR_3;
43         wTimePhA = (((int32_t)pHandle->PWMperiod) / 4) + ((wY - wX)
44 / (int32_t)262144);
45         wTimePhC = wTimePhA - (wY / 131072);
46         wTimePhB = wTimePhC + (wX / 131072);
47
48         pHandle->lowDuty = (uint16_t)wTimePhB;
49         pHandle->midDuty = (uint16_t)wTimePhC;
50         pHandle->highDuty = (uint16_t)wTimePhA;
51     }
52 }
53 else /* wY > 0 */
54 {

```

```

53     if (wZ >= 0)
54     {
55         pHandle->Sector = SECTOR_2;
56         wTimePhA = (((int32_t)pHandle->PWMperiod) / 4) + ((wY - wZ) /
(int32_t)262144);
57         wTimePhB = wTimePhA + (wZ / 131072);
58         wTimePhC = wTimePhA - (wY / 131072);
59
60         pHandle->lowDuty = (uint16_t)wTimePhB;
61         pHandle->midDuty = (uint16_t)wTimePhA;
62         pHandle->highDuty = (uint16_t)wTimePhC;
63     }
64     else /* wZ < 0 */
65     {
66         if (wX <= 0)
67         {
68             pHandle->Sector = SECTOR_6;
69             wTimePhA = (((int32_t)pHandle->PWMperiod) / 4) + ((wY - wX)
/ (int32_t)262144);
70             wTimePhC = wTimePhA - (wY / 131072);
71             wTimePhB = wTimePhC + (wX / 131072);
72
73             pHandle->lowDuty = (uint16_t)wTimePhA;
74             pHandle->midDuty = (uint16_t)wTimePhC;
75             pHandle->highDuty = (uint16_t)wTimePhB;
76         }
77         else /* wX > 0 */
78         {
79             pHandle->Sector = SECTOR_1;
80             wTimePhA = (((int32_t)pHandle->PWMperiod) / 4) + ((wX - wZ) /
(int32_t)262144);
81             wTimePhB = wTimePhA + (wZ / 131072);
82             wTimePhC = wTimePhB - (wX / 131072);
83
84             pHandle->lowDuty = (uint16_t)wTimePhA;
85             pHandle->midDuty = (uint16_t)wTimePhB;
86             pHandle->highDuty = (uint16_t)wTimePhC;
87         }
88     }
89     pHandle->CntPhA = (uint16_t)(MAX(wTimePhA, 0));
90     pHandle->CntPhB = (uint16_t)(MAX(wTimePhB, 0));

```

```

91     pHandle->CntPhC = (uint16_t)(MAX(wTimePhC, 0));
92
93     /* Can be inserted the DT compensation */
94
95     /* Set Duties and sampling point */
96     returnValue = pHandle->pFctSetADC SampPointSectX(pHandle);
97     return (returnValue);

```

4.3.4 HALL SENSOR MANAGEMENT

Many problems have been encountered for the implementation of the Hall sensor management, the first problem concerns the precise definition of the Hall offset which has been solved by an estimation procedure in section 2.4.2.

The second issue concerns the computation of the electrical angle θ_{me} . The use of the STM motor control library leads to some peaks and computation problems in angle and speed update, the reasons due to the time, have not been fully verified, but the more reasonable motivation has been assigned to speed and complexity in the code.

This problem has been corrected thanks to the FDK architecture, in fact, CAME already has a module for the Hall sensors. Implementing and adapting it to the system has been fast and leads to better results.

The main difference between the two projects are that: the ST file performs the Hall sensor feedback processing using a specific feature of a timer set in Input Capture (IC) mode; It also does a change in the timer clock prescaler while running for increase the resolution and perform a comparison and adjustment of the speed measurement between Hall sensor rising/falling edge. Moreover, an additional timer (TIMER 2) is implemented in the ST file needed for synchronized the PWM start.

All these features are not present in the CAME Hall management file, the electrical angle and speed computation are performed cyclically by calling a single function. This function reads the Hall sensor using a GPIO, compute the new Hall state if necessary, update the electrical angle considering the Hall offset and the direction and then update the speed based on the average speed between two loops.

The direction is managed directly in the Application project. It is used as an input for closing or opening the gate.

The CAME code is not reported but the ST code can be easily generated through the motor

control workbench tool, following one of the many instructions available on the web.

4.3.5 SENSORLESS ALGORITHM

The sensorless algorithm implemented, but not tested, follows the theory explained in section 2.3.3 that it is also the theory adopted in the MCSDK of the STMicroelectronics [19], [22], [23].

The algorithm is based on the estimation of a generalized back-EMF space vector by means of a Luenberger state observer implemented in the rotating reference frame d - q plus a PLL function. The estimated rotor position and velocity are then used to run the speed control loop and the FOC algorithm.

The sensorless algorithm has not yet been tested, but has been implemented in the FDK driver folder and designed in the Matlab/Simulink environment.

The Observer and PLL algorithm in C code format are reported below, while the function for start-up can be generated using the motor control workbench tool.

```
1  /* ST BACK-EMF LUENBERGER OBSERVER FUNCTION */
2
3  int16_t retValue;
4  int32_t wAux, wDirection;
5  int32_t wIalfa_est_Next, wIbeta_est_Next;
6  int32_t wBemf_alfa_est_Next, wBemf_beta_est_Next;
7  int16_t hAux, hAux_Alfa, hAux_Beta;
8  int16_t hIalfa_err, hIbeta_err;
9  int16_t hRotor_Speed;
10 int16_t hValfa, hVbeta;
11
12 /* Scaling */
13 hAux_Alfa = (int16_t)(pHandle->wBemf_alfa_est / pHandle->hF2);
14 hAux_Beta = (int16_t)(pHandle->wBemf_beta_est / pHandle->hF2);
15 hIalfa_err = (int16_t)(pHandle->Ialfa_est / pHandle->hF1);
16 hIalfa_err = hIalfa_err - pInputs->Ialfa_beta.alpha;
17 hIbeta_err = (int16_t)(pHandle->Ibeta_est / pHandle->hF1);
18 hIbeta_err = hIbeta_err - pInputs->Ialfa_beta.beta;
19 wAux = ((int32_t)pInputs->Vbus) * pInputs->Valfa_beta.alpha;
20 hValfa = (int16_t)(wAux / 65536);
21 wAux = ((int32_t)pInputs->Vbus) * pInputs->Valfa_beta.beta;
```

```

22     hVbeta = (int16_t)(wAux / 65536);
23
24     /* alfa axes observer */
25     hAux = (int16_t)(pHandle->Ialfa_est / pHandle->hF1);
26     wAux = ((int32_t)pHandle->hC1) * hAux;
27     wIalfa_est_Next = pHandle->Ialfa_est - wAux;
28     wAux = ((int32_t)pHandle->hC2) * hIalfa_err;
29     wIalfa_est_Next += wAux;
30     wAux = ((int32_t)pHandle->hC5) * hValfa;
31     wIalfa_est_Next += wAux;
32     wAux = ((int32_t)pHandle->hC3) * hAux_Alfa;
33     wIalfa_est_Next -= wAux;
34     wAux = ((int32_t)pHandle->hC4) * hIalfa_err;
35     wBemf_alfa_est_Next = pHandle->wBemf_alfa_est + wAux;
36     wAux = ((int32_t)hAux_Beta) / pHandle->hF3;
37     wAux = wAux * pHandle->hC6;
38     wAux = pHandle->_Super.hElSpeedDpp * wAux;
39     wBemf_alfa_est_Next += wAux;
40
41     /* beta axes observer */
42     hAux = (int16_t)(pHandle->Ibeta_est / pHandle->hF1);
43     wAux = ((int32_t)pHandle->hC1) * hAux;
44     wIbeta_est_Next = pHandle->Ibeta_est - wAux;
45     wAux = ((int32_t)pHandle->hC2) * hIbeta_err;
46     wIbeta_est_Next += wAux;
47     wAux = ((int32_t)pHandle->hC5) * hVbeta;
48     wIbeta_est_Next += wAux;
49     wAux = ((int32_t)pHandle->hC3) * hAux_Beta;
50     wIbeta_est_Next -= wAux;
51     wAux = ((int32_t)pHandle->hC4) * hIbeta_err;
52     wBemf_beta_est_Next = pHandle->wBemf_beta_est + wAux;
53     wAux = ((int32_t)hAux_Alfa) / pHandle->hF3;
54     wAux = wAux * pHandle->hC6;
55     wAux = pHandle->_Super.hElSpeedDpp * wAux;
56     wBemf_beta_est_Next -= wAux;
57
58     /* Calls the PLL blockset */
59     pHandle->hBemf_alfa_est = hAux_Alfa;
60     pHandle->hBemf_beta_est = hAux_Beta;
61     hAux_Alfa = (int16_t)(hAux_Alfa * wDirection);
62     hAux_Beta = (int16_t)(hAux_Beta * wDirection);

```

```

63
64     hRotor_Speed = STO_ExecutePLL(pHandle, hAux_Alfa, -hAux_Beta);
65     pHandle->_Super.InstantaneousElSpeedDpp = hRotor_Speed;
66     STO_Store_Rotor_Speed(pHandle, hRotor_Speed);
67     pHandle->_Super.hElAngle += hRotor_Speed;
68
69     /*storing previous values of currents and bemfs*/
70     pHandle->Ialfa_est = wIalfa_est_Next;
71     pHandle->wBemf_alfa_est = wBemf_alfa_est_Next;
72     pHandle->Ibeta_est = wIbeta_est_Next;
73     pHandle->wBemf_beta_est = wBemf_beta_est_Next;
74     retValue = pHandle->_Super.hElAngle;
75
76     return (retValue);

```

```

1  /* CLASSIC PLL FUNCTION */
2  int32_t SinAlfa_tmp, CosBeta_tmp;
3  Trig_Components Local_Components;
4  int16_t Out;
5
6  /* Compute sine and cosine of the electrical angle */
7  Local_Components = Trig_Functions(pHandle->_Super.hElAngle);
8
9  /* Alfa & Beta BEMF multiplied by cosine and sine */
10 SinAlfa_tmp = (int32_t)hBemf_alfa_est * (Local_Components.hSin);
11 CosBeta_tmp = (hBemf_beta_est) * (Local_Components.hCos);
12
13 /* Speed PI regulator */
14 Out = PI_Controller(&pHandle->PIRegulator, CosBeta_tmp-SinAlfa_tmp);
15
16 return (Out);

```

By looking to the Back-EMF LO function, can be noticed that it uses several gains: F1, F2, F3, C1, C2, C3, C4, C5 and C6 not mentioned in the theory. These gains increase the complexity in the code understanding and concern with speed and code optimization without making any big difference with respect the theory presented in section 2.3.3.

For completeness it is reported a brief description for each gain with its formula. F1, F2, F3 are simple scaling factor parameters used for change the measurement units, the formula is not

reported. Instead, more interesting are the values assigned to the gains: C1, C2, C3, C4, C5, C6 which are pre-calculated gains that depends directly from the motor parameters.

$$\begin{aligned}
C1 &= \frac{F1 \cdot R_s}{L_s \cdot f_s} \\
C2 &= \frac{F1 \cdot K1}{f_s} \\
C3 &= \frac{F1 \cdot \omega_N \cdot K_e \cdot \sqrt{2}}{L_s \cdot I_N \cdot f_s} \\
C4 &= \frac{K2 \cdot I_N}{\omega_N \cdot K_e \cdot \sqrt{2} \cdot F2 \cdot f_s} \\
C5 &= \frac{F1 \cdot V_N}{2 \cdot L_s \cdot I_N \cdot f_s} \\
C6 &= \frac{F2 \cdot F3}{CONST}
\end{aligned} \tag{4.3}$$

where f_s is the execution rate, $K1$ and $K2$ are the two discrete observer gains (called in the thesis with $L1$ and $L2$), K_e is the Back-EMF constant in [Vllrms/krpm] and the constant used in $C6$ is computed with a specific procedure starting from the other constants.

In Equation 4.4 are reported the two state equation 2.33 using the gains defined above.

$$\begin{aligned}
\hat{I}_{\alpha\beta}(k+1) &= \hat{I}_{\alpha\beta}(k) - C1 \cdot \hat{I}_{\alpha\beta}(k) - C2 \cdot (\hat{I}_{\alpha\beta}(k) - I_{\alpha\beta}(k)) - C3 \cdot \hat{E}_{\alpha\beta}(k) + \\
&\quad + C5 \cdot U_{\alpha\beta}^*(k)
\end{aligned} \tag{4.4}$$

$$\hat{E}_{\alpha\beta}(k+1) = \hat{E}_{\alpha\beta}(k) + C6 \cdot \hat{E}_{\alpha\beta}(k) + C4 \cdot (\hat{I}_{\alpha\beta}(k) - I_{\alpha\beta}(k))$$

If the state observer Equation 2.33 and 4.4 are compared the $C6$ gain can be extrapolated, determining:

$$\begin{aligned}
C6 &= \frac{F2 \cdot F3}{CONST} \\
CONST &= \frac{f_s}{j\omega_{me}}
\end{aligned} \tag{4.5}$$

This last parameter is not reported in any ST manual so it can be slightly different.

5

Simulation and Design of the Control System

In the control scenario, Model Based Design (MBD) not only reduces expensive hardware processes increasing the safety, but can also be very efficient in significantly reducing time-to-market on future new designs, permitting additional critical flexibility to meet customer performance requirements. This method has been adopted for design the FOC algorithm with all the tunings.

The chapter will initially present the overall model of the system for later go into details describing the plant to be controlled. The motor, the load and the inverter represent the plant and to construct it some model verification are carried out through experiments.

Hence, the control field can be explored. The results of the simulation and the design procedures for the FOC algorithm are shown with the electrical angle calculated both with and without Hall sensors in the stationary and rotating reference frame.

At the end, a confrontation of the two control strategies is conducted, highlighting the advantages and drawbacks of each implementation, based on the simulation tests.

The model was built starting from one of my colleges project using the software Matlab/Simulink with the toolbox: control system, curve fitting, optimization and signal processing.

5.1 MODEL OF THE OVERALL SYSTEM

The system considered includes many blocks which can confuse the reader. To better understand the whole chapter, has been preferred to give a first image of the overall system, listing the main sections. The total system is broken down into:

- Inputs block;
- Speed controller;
- FOC algorithm;
- Inverter block (Plant);
- Motor block (Plant);
- Outputs block;
- Sensored or sensorless feedback chain;

Leaving aside input and output blocks, used respectively to provide a specific speed reference and to monitor the main variables. The controller, the plant and the feedback constitute the three fundamental blocks for a general feedback control system, and this also applies to the subject being discussed.

The only peculiarity is that a cascade control is used, it is constituted of an internal current loop and an outer speed loop. This type of control involves the adoption of two controllers and two plant to be modelled.

The outer controller is a Fuzzy PI speed controller that it is responsible for providing the desired reference of torque or currents to meet the speed requirement. Hence, the speed controller output constitutes the input of the fast current loop that through two PIs and the constant electrical angle monitoring is able to give the best voltage reference in the frame $d-q$.

The plant to consider depends on the loops. For the current loop, the plant consists of the hardware part without the mechanical block. First of all there is the inverter, necessary to produce the three phase voltages, and successively the electric motor.

Instead, the second plant is considered for the speed loop and is based on the close-loop transfer function of the current and on the mechanic block.

Note that when designing a cascade control, the inner loop must be at least five time faster than the outer loop, otherwise stability problems may arise.

The feedback block, with or without sensor, is responsible to replicate the actual measurement or estimate of the speed and of the electrical angle, required for the control block.

GIULIO SAVIAN MODEL

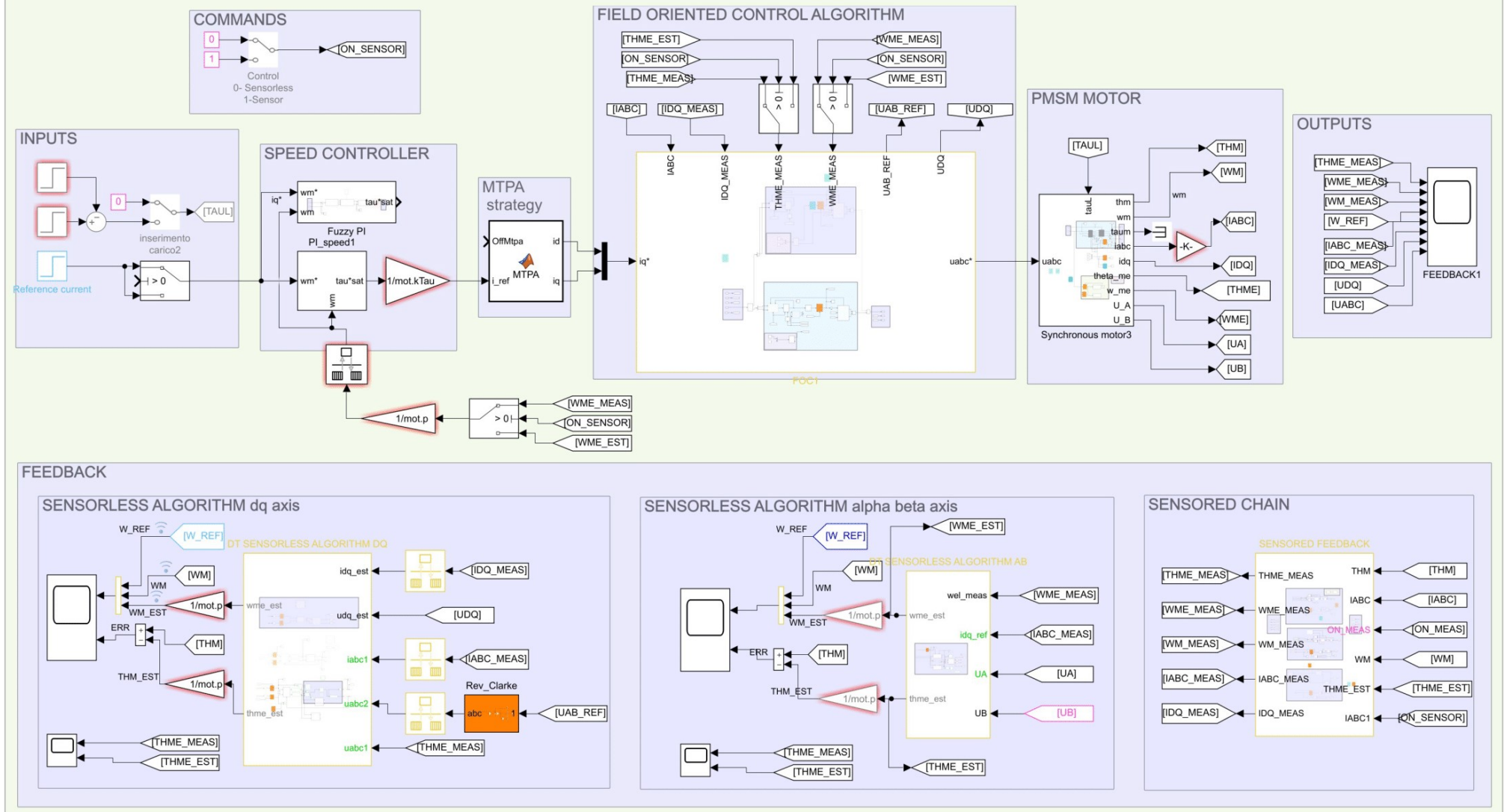


Figure 5.1: Simulation of the whole control system

5.1.1 SIMULINK MODEL OF THE ELECTRIC MOTOR

In this thesis is used a model of the drive in a moving d - q system, in order to control the motor with stable quantities precisely in d - q , following the Equation 2.13 and 2.16.

In order for the model to work properly, it must receive two voltage references in the d - q axis so a Clarke and Park transform block is needed to change the three phase voltages u_{abc} , outputs of the inverter, into u_d and u_q .

The Figure 5.3 highlights the presence of two models construction, that in the thesis simulation are able to give the same results. The one below, in the green rectangle, is the standard SPM motor block that implements the block diagram of Figure 5.4.

While, the used model is in the blue rectangle and divide the electrical and the mechanical block implementing the same formulas of the d - q SPM model. It is different since incorporates the possibility of considering flux maps, needed in case of non-linear contributions, such as saturation or current cross coupling.

In the project it was sufficient to follow the hypothesis of linearity between currents and inductance but for an accurate simulation, tests or FEM approaches can be performed to build non-linear flux maps.

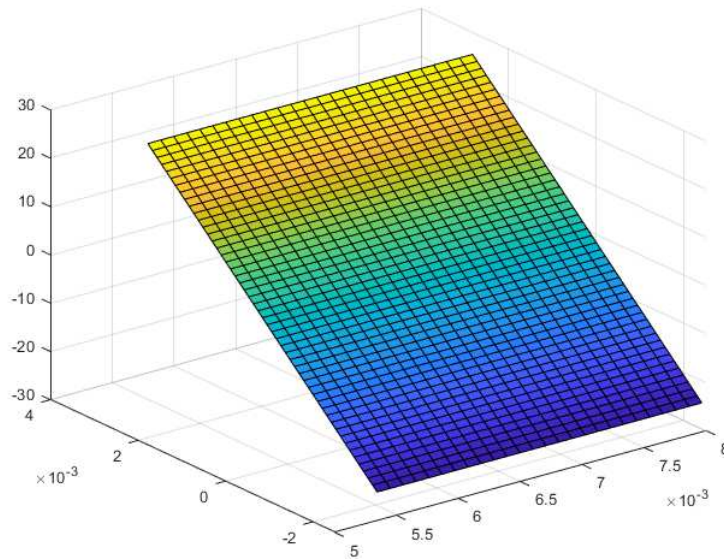


Figure 5.2: Fluxes map for I_q

Linear flux maps have been implemented inside the electrical block, with a meshgrid starting from the possible values that I_d and I_q can have, computed using the nominal current, Fig-

ure 5.5.

The results of the meshgrid are visible in Figure 5.2.

$$\begin{cases} I_{dVec} = [-I_N : 1 : I_N] \\ I_{qVec} = [-2I_N : 1 : 2I_N] \\ \lambda_{dVec} = L_d \cdot I_{dVec} \cdot \lambda_{mg} \\ \lambda_{qVec} = L_q \cdot I_{qVec} \end{cases} \quad (5.1)$$

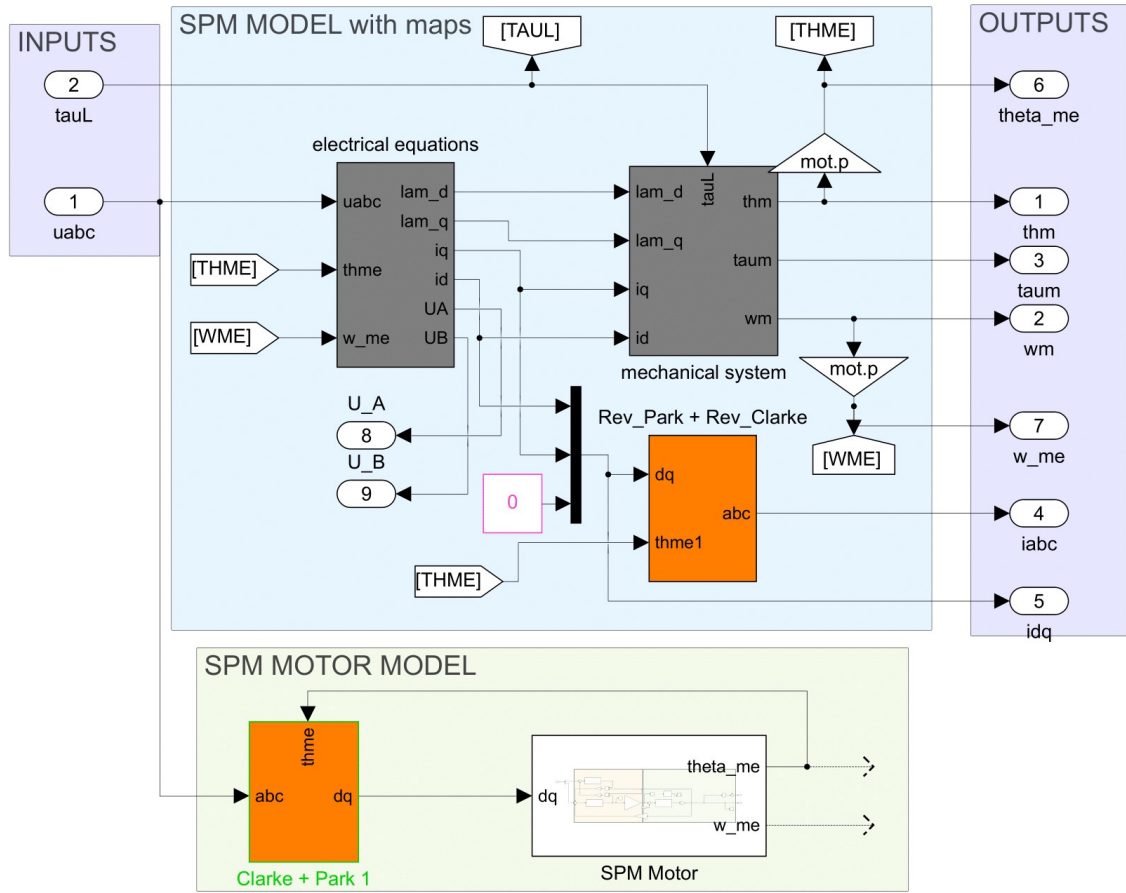


Figure 5.3: Two motor models implemented in Simulink: a standard one and a model that can comprise non linearity effects

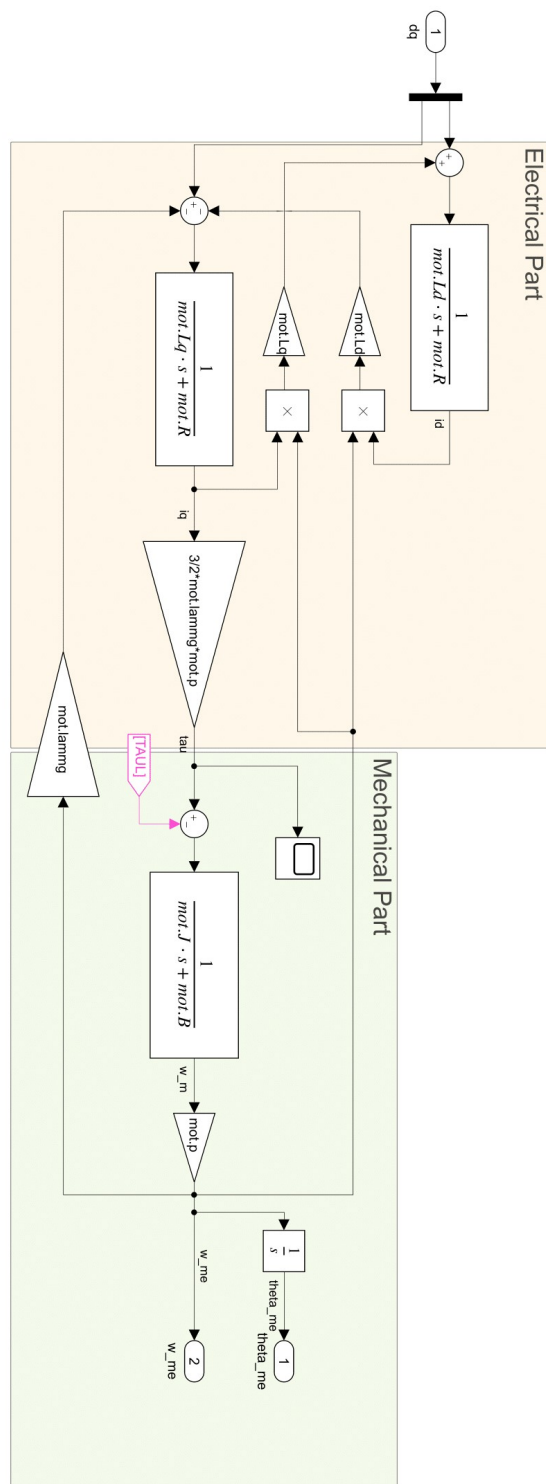


Figure 5.4: Standard d - q PMSM motor model implementation in Simulink

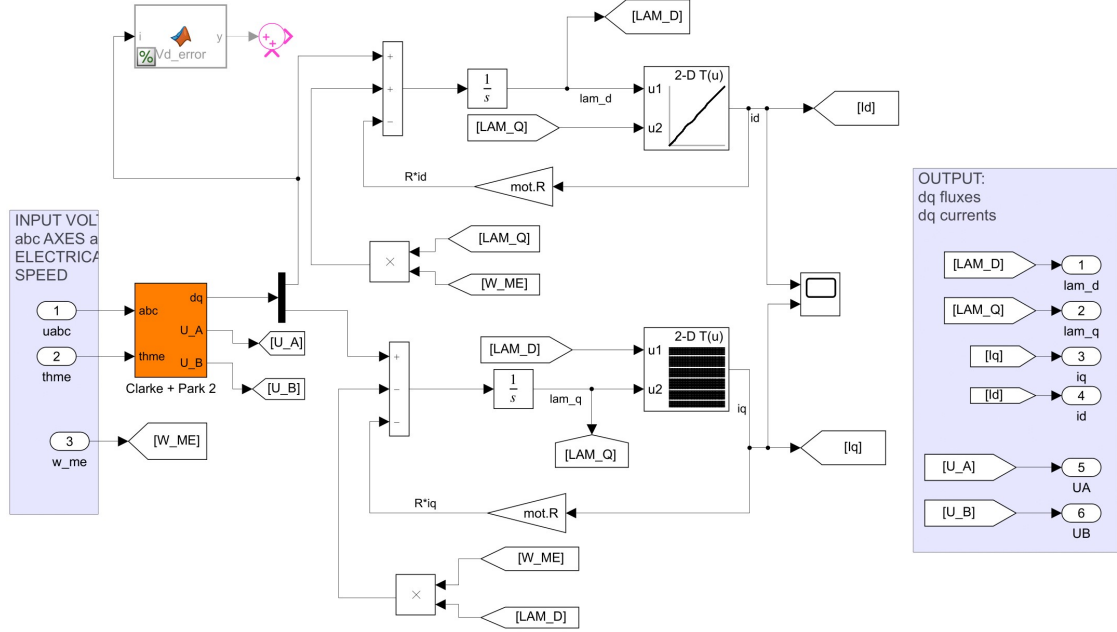


Figure 5.5: Electrical block of the d - q PMSM model including flux maps

The electric block computes the values of fluxes and currents in the d - q axis that are needed by the mechanical block of Figure 5.6, to elaborate the motor torque τ_m , the motor velocity ω_m and the mechanical angle θ_m . The mechanical parameters are computed following the torque balance, Equation 2.16, through a state-space model block.

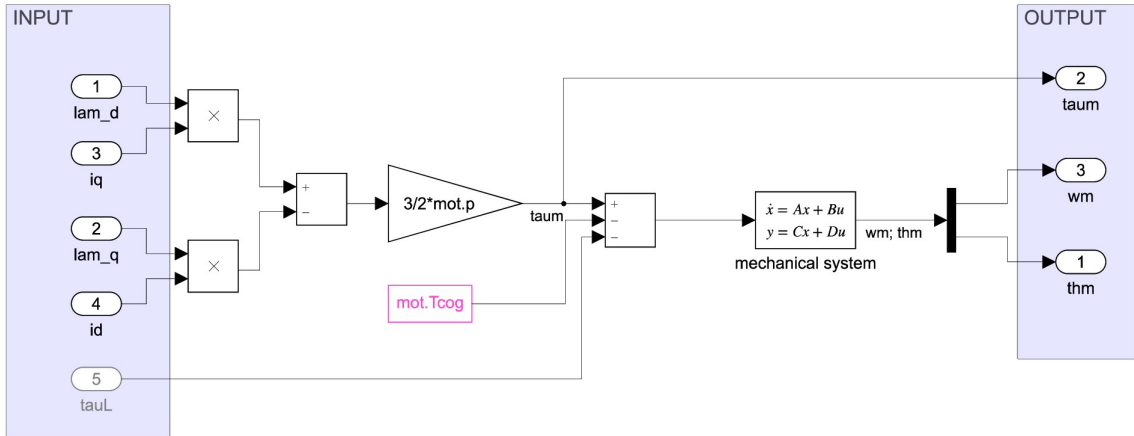


Figure 5.6: Mechanical block of the d - q PMSM model

The division between the electrical and the mechanical part of the motor simplify the under-

standing and the extraction of the motor phase currents I_{abc} . In fact, current sampling is a fundamental part in the FOC algorithm, necessary for the inner current loop but also for the electrical angle θ_{me} estimation in case of sensorless algorithm.

For this purpose, the currents are transformed from the d - q axis to the three phases and then the current sampling is simulated, as in reality.

VALIDATION OF THE MOTOR MODEL

The motor model validation has been performed by considering each block separately to avoid possible interference with other parts of the system or control. Then the tests are conducted in open-loop providing the voltage references in the d - q axes.

The axes d - q were chosen to avoid implementing sinusoidal reference voltages since they are computed automatically through the reverse Park transformation and the inverter. Despite this integration, motor verification can be done by measuring the input and output values for each block.

The test has been conducted with twelve different references, six with different voltage in d and as many V_q . Starting at 500 mV and ending with 3000 mV in order to check out a wide range of available solutions.

The measured values are the d - q current for verify the correctness of the electrical part and the mechanical velocity ω_m for check the mechanical block.

In Figure 5.7 are reported the real and the simulation results of the current I_d considering both an ideal inverter and a not ideal one for the V_d reference voltage.

From the tests can be seen that using an ideal inverter leads to an offset error in the d current I_d between simulation and reality, while considering the inverter non-linearity this offset is reduced, which is approximately zero for the references medium.

For $V_d = 3000$ the real signal I_d in blue remains equal in average to the test with $V_d = 2500$, this means that the motor limits are achieved.

For the low voltage references is present an offset of about 0.5 V probably due to the major inverter inefficiency if the low voltages are considered. These non-linearities can be neglected since they are present only at low voltages and are not relevant for the dynamics.

Note that the influence of the d voltage V_d on the q axis is correctly modelled in mean.

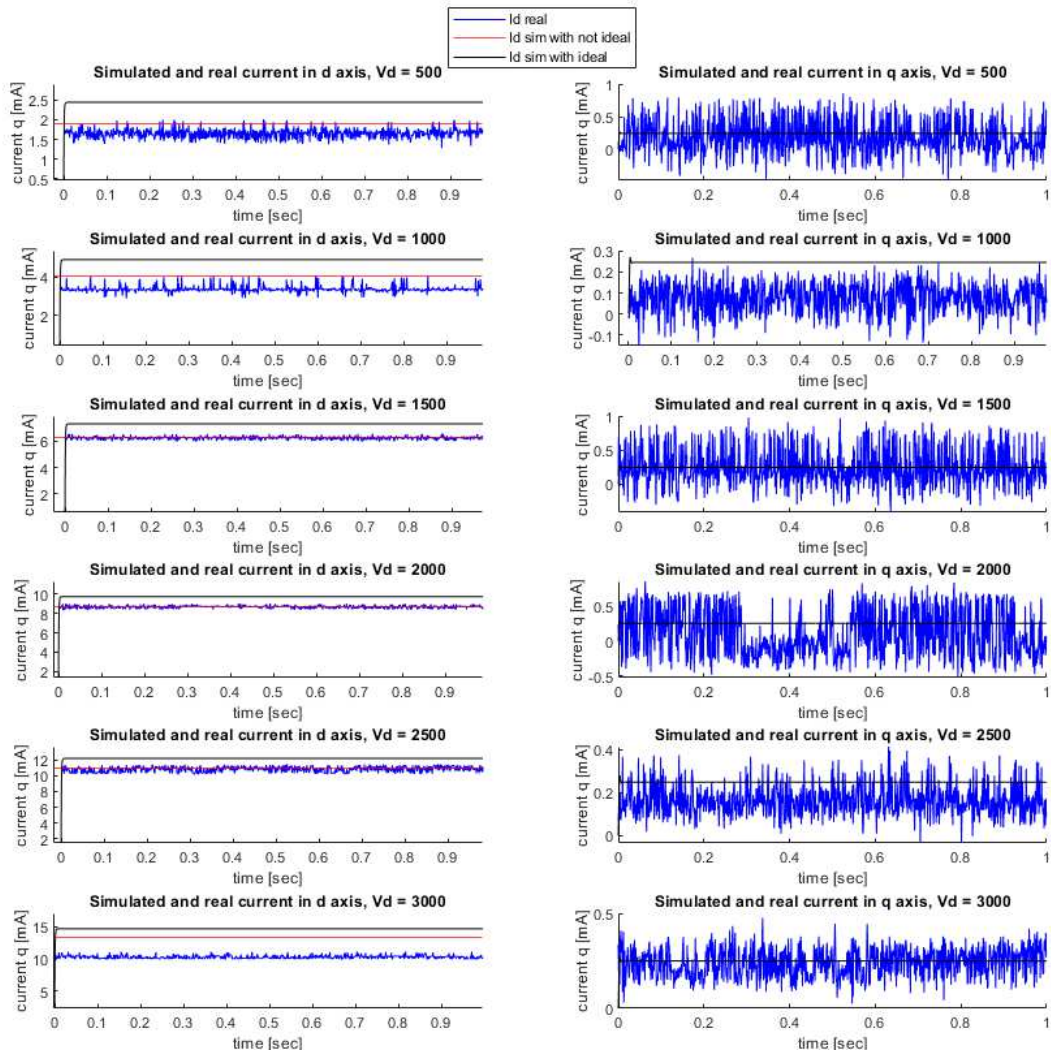


Figure 5.7: Test results conducted for the PMSM motor model validation with reference in V_d

Figure 5.9 and 5.8 report the data to the test performed with fixed voltage reference in the q axis V_q . Compared to before, the q axis is responsible for generating the torque and as a consequence the speed, so the speed comparison is also reported.

The q axis model verification is more demanding, not only for the speed but also for inductance contribute and cross coupling effects. Finding a satisfactory model took a few times and a solution was found by adapting the values of inductance L_d , L_q that were estimated in the second chapter.

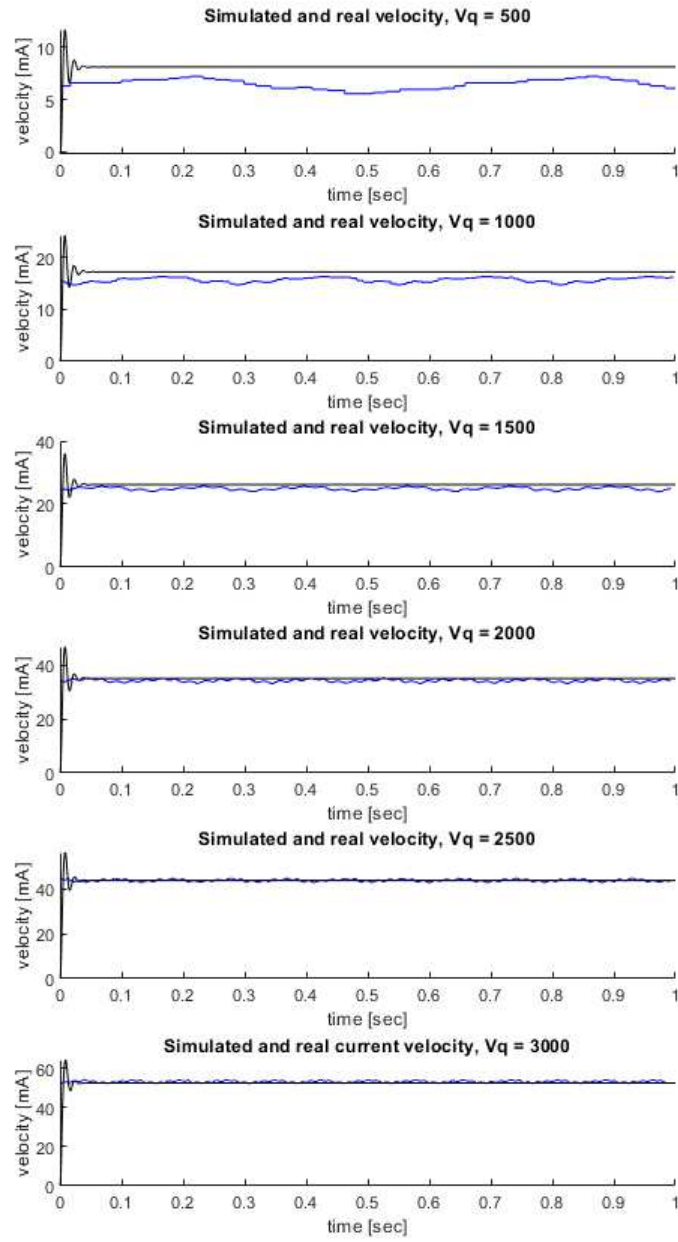


Figure 5.8: Mechanical Speed test results conducted for the PMSM motor model validation with reference in V_q

The average of the real signals follows the simulation for both the currents and mechanical speed. The only divergence is reflected in the low voltage reference due to the inverter non-linearity.

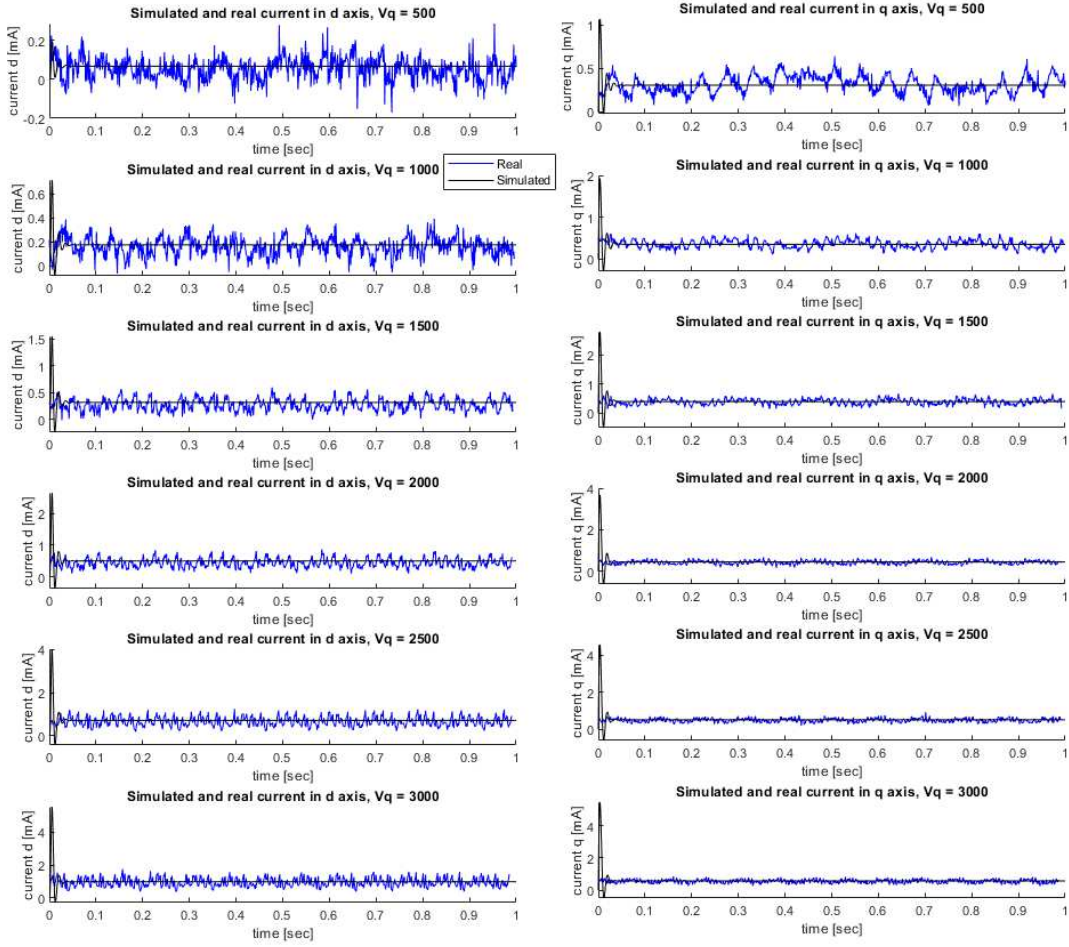


Figure 5.9: Current test results conducted for the PMSM motor model validation with reference in V_q

5.1.1.2 SIMULINK MODEL OF THE INVERTER

The inverter block takes as input the α - β reference voltages computed through the reverse Park transformation and the bus voltage V_{BUS} and it is able to calculate the three phase voltages u_{abc} needed to power the three phases of the motor. All calculations and theory can be founded following the SVPWM algorithm explained in section 3.2.2 and the code implemented in section 4.3.3.

The inverter simulation block of Figure 5.10 is divided in two parts: the first section implements the SVM algorithm performing firstly the conversion of u_{abc} into the three phases X, Y, Z (as ST nomenclature), and after the sector identification to assign the duties for each phase, Figure 5.11.

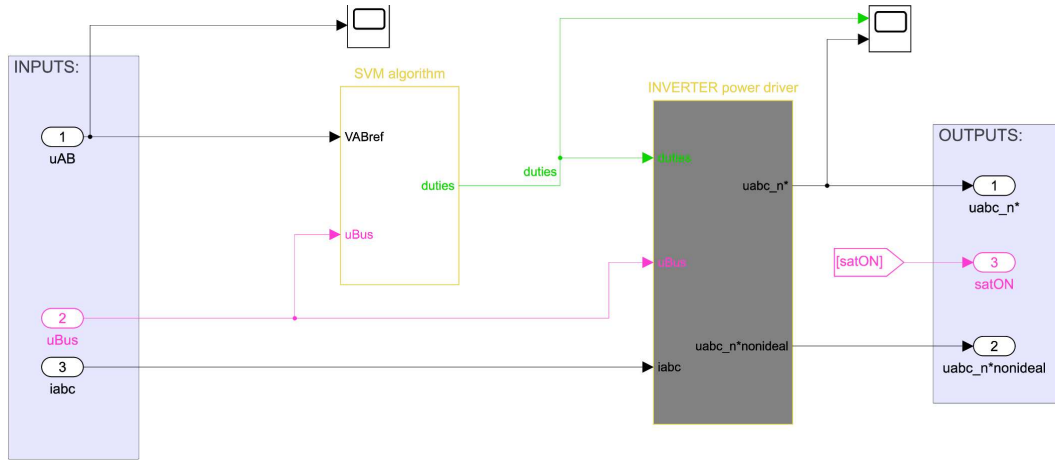


Figure 5.10: Inverter block implemented in Simulink

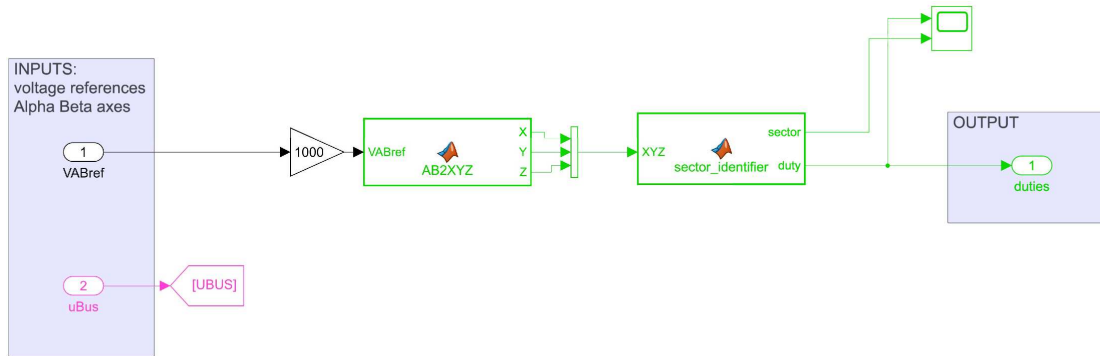


Figure 5.11: SVM algorithm

The second block performs a scaling of the duties to produce the correct reference voltages. It subtracts 0.5 for center the duty signals at zero mean, remember than PWM is used in center aligned mode, and then multiplying the resulting signals by the bus voltage to get the three phase voltages shifted by $\frac{2\pi}{3}$.

But that's not all, the inverter introduces a small delay necessary to wait a front edge of the PWM. This delay can be maximum i.e. equal to the PWM switching period or zero when the PWM front edge is instantaneous. The best choice is to use the average of the worst-case scenario and the best-case scenario, so half of the switching period T_{PWM} is considered.

The following equation in Laplace notation can summarize the inverter contribute for a block

diagram representation:

$$V(s) = \frac{K_c}{1 + sT_c} \quad (5.2)$$

where $K_c = \frac{V_{OUT}}{V_{IN}}$ is the converter gain, T_c is the converter delay and $V(s)$ is the output voltage. The Figure 5.12 graphically represents the scaling and delay implementation in Simulink.

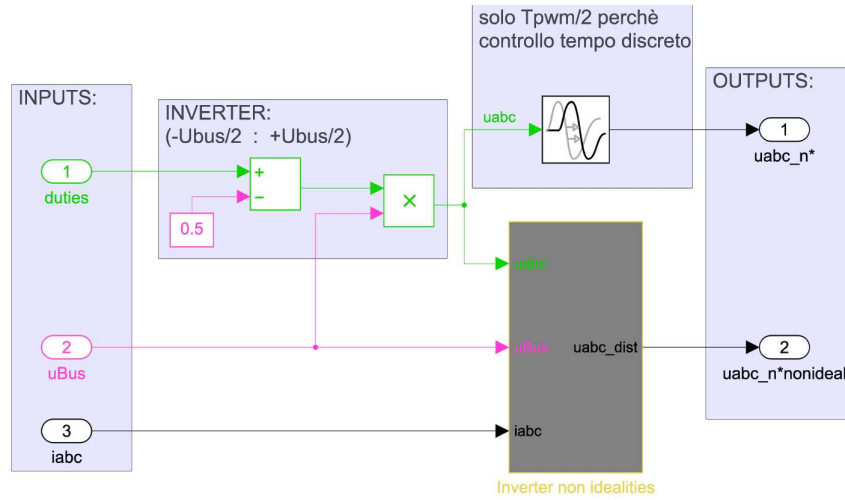


Figure 5.12: Inverter power driver

VALIDATION OF THE INVERTER MODEL

Verification of the inverter model is conducted by looking only at the correspondence between the duties since the output is a simple scaling procedure and the input is generated by the user. For simplicity, the same data set as the previous test is used, considering only the q axis. The test conducted in the d axis is neglected since it does not produce torque and motion, leading to the same electrical angle and therefore to a static input voltage $\alpha\text{-}\beta$.

As before, the test is conducted with six different references starting from $V_q = 500 \text{ mV}$ and concluding with $V_q = 3000 \text{ mV}$ in order to verify a wide range of available solutions.

In Figure 5.13 are reported the real and the simulation results in the same graph for each reference voltage.

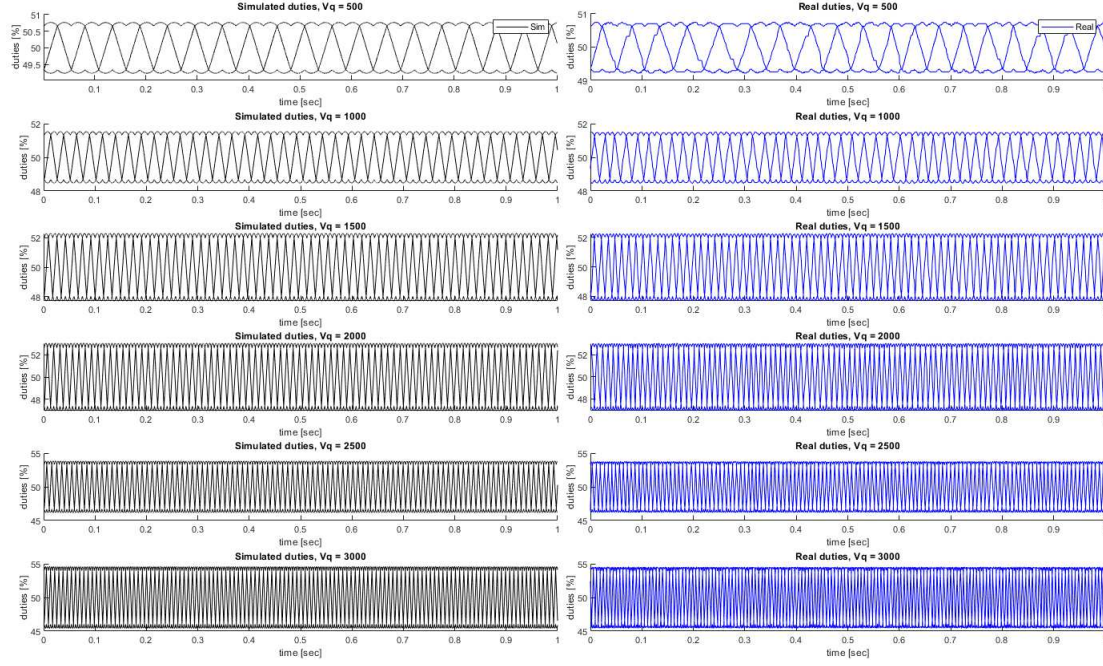


Figure 5.13: Test results conducted for the inverter model validation

The plots show that the duties amplitude are respected and the frequencies are very similar. However, the duties frequency will depend on the measured electrical angle θ_{me} by the Park transformation, so to be perfectly equal the motor dynamics and electrical speed must coincide with the reality.

From these considerations a sufficient precise model of the plant is reached.

5.2 FIELD ORIENTED CONTROL DESIGN

Now that the plant is modelled it's transfer function can be computed and the field oriented control can be designed.

The block diagram of the FOC with the speed loop is shown in Figure 2.7 and includes: three PI controllers, the Park and Clarke transformations for converting from stationary to rotating synchronous frames, a reference current generator (MTPA), the feedback chain, the decoupling block for control independently the d - q axes, the circular limitation and a spatial vector modulator algorithm used to transform the V_α and V_β commands into PWM signals applied to the stator winding.

All the main topics listed have been defined, determining that the currents PI controllers can be designed.

Two PIs are used to separately control the two current axes d - q , while the third PI is used for the speed control loop.

For the design, the FOC algorithm is required to fine-tune all the PIs according on the plant transfer function. It is represented starting from the transfer function 5.18 of the inverter and the electric motor transfer function:

$$V(s) = \frac{1}{(R_s + sL_s)} I(s) \quad (5.3)$$

obtaining the following equation:

$$V(s) = \frac{K_c}{(1 + sT_c)(R_s + sL_s)} I(s) \quad (5.4)$$

Substituting the estimated, measured or fixed values the respectively Bode diagram is plotted below:

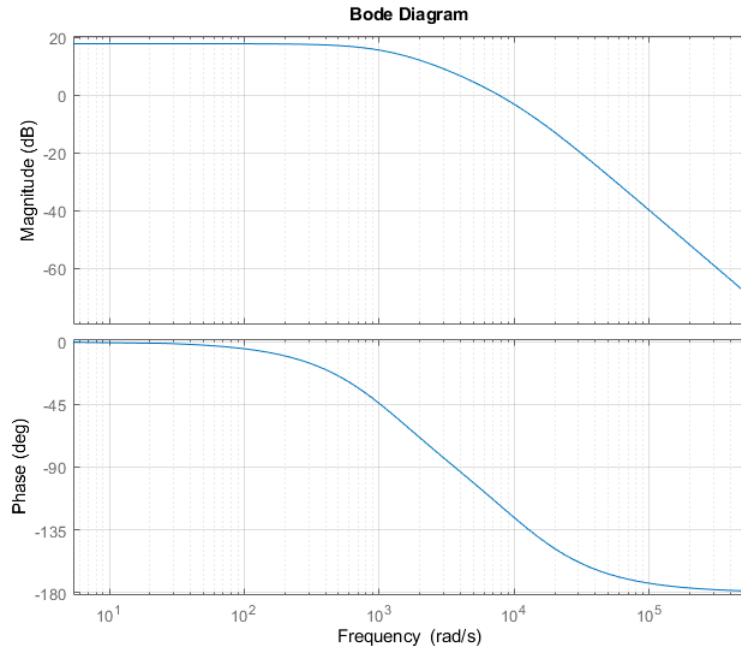


Figure 5.15: Bode diagram of the plant in open-loop

In the following the current control loop and the speed control loop are designed.

5.2.1 CURRENT CONTROL LOOP

The implementation of the Simulink current control loop consists of two PI controllers. The loop takes the current reference inputs generated by the MTPA and computes the error inputs of the controllers. Hence, the loop is responsible for generating the best reference voltages for the motor. But before, these voltage references pass through a circular limitation block, that, as the name suggests, performs a circular saturation for avoid overvoltages.

Since it is considered a non-saliency motor, the MTPA is very simple and the d -axis PI controller simply has to set the current I_d to zero. The other PI controller is needed for control the current I_q , used by the formula 2.15 to produce the required torque.

The two current axes loops can be represented as in Figure 5.16.

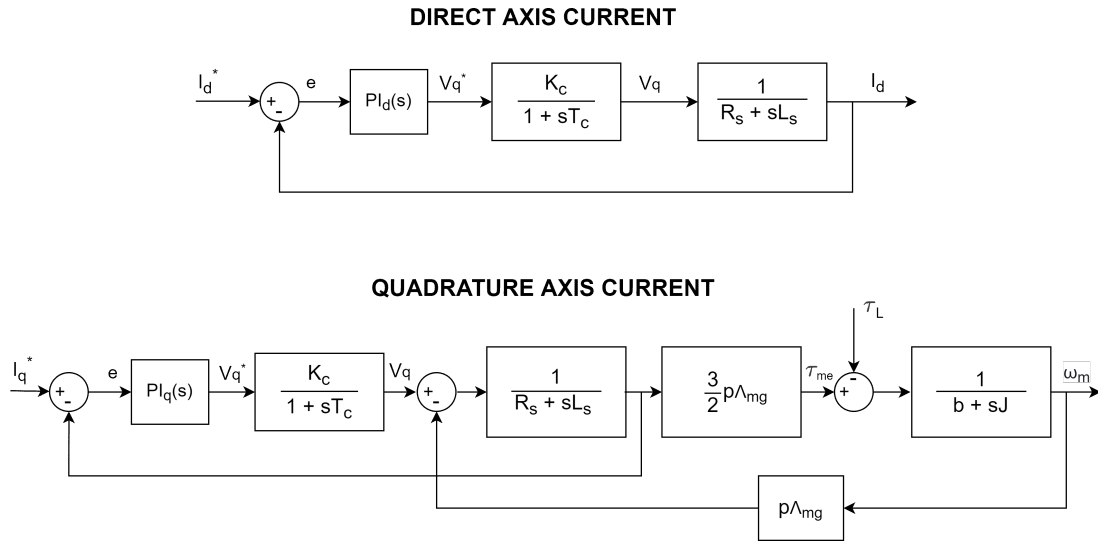


Figure 5.16: Current control loop for a SPM motor in the direct and quadrature axes

Note that the two axis are different. In the q -axis voltage equation, there is the EMF term induced by the permanent magnets $e(t) = \frac{d\lambda_m(t)}{dt}$ or in Laplace's notation, $\omega_{me}(s)\Lambda_{mg}$. Therefore, in the block scheme it is included the mechanical loop, whose output is the speed ω_m used to compute the EMF that is subtracted from the voltage v_q .

With this link the q axis current is complex to design and one of the following three techniques must be considered.

The first possibility is to neglect the speed dynamics with respect to the current dynamics since the mechanical time constant $T_m = \frac{J}{B}$ is usually bigger than the electrical time constant $T_e = \frac{L_s}{R_s}$. In this way, the term $\omega_{me}\Lambda_{mg}$ can be considered as a constant disturbance and it

is neglected in the study of the current dynamics.

As a consequence, the block diagram becomes equal to the d -axis current, represented in Figure 5.16. The design process of the PI_q controller is the same as for the regulator PI_d of the d -axis current loop.

The second possibility consists in decoupling the two axes and compensate the EMF contribution due to the permanent magnets, Equation 2.13. The compensation is performed using a feedforward action that allows to compensate the term $\omega_{me}\Lambda_{mg}$ and neglected it in the voltage equation and so in the current loop. The Figure 5.17 shows the block scheme representation, obtaining these output voltage values:

$$\begin{cases} u_{d,dec}^* &= u_d^* + \omega_{me}LI_q \\ u_{q,dec}^* &= u_q^* - \omega_{me}(LI_d + \lambda_{mg}) \end{cases} \quad (5.5)$$

where $u_{d,dec}^*$ and $u_{q,dec}^*$ are the decoupled voltages input of the following circle limitation block, while u_d^* and u_q^* are the outputs of the current controller.

The design of the regulators is simplified and the coupled current loop is reduced to that used for the d -axis current.

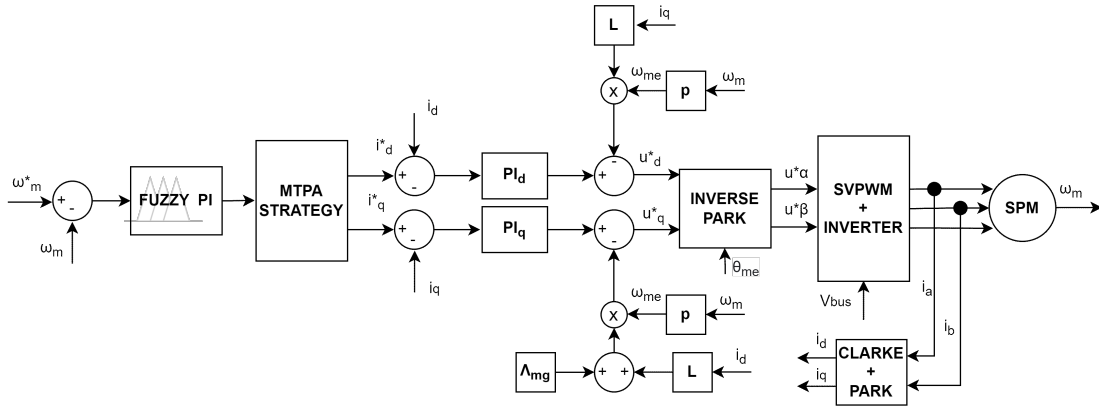


Figure 5.17: Control system block diagram with decoupling mechanism

The last possibility is to include the term EMF in the current dynamic. The current dynamics will include the speed loop.

With this analysis, the current loop design can be performed by acting only for one axis since with some considerations the q axis current loop becomes equal to the d axis loop.

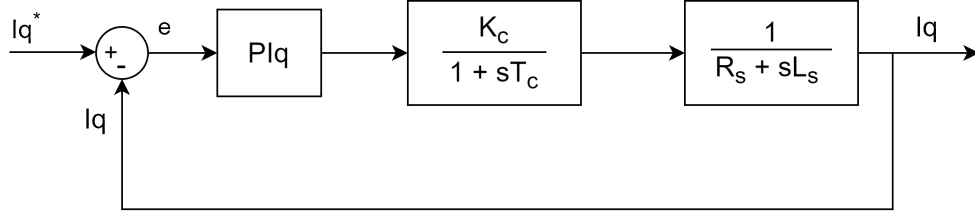


Figure 5.18: Block diagram current loop in the q axis

CURRENT LOOP TUNINGS

One way to determine the robustness of a system and also to give an idea of the instability regions is to determine the bandwidth of a system. Bandwidth yields information about the phase margin (ϕ_m) and gain margin (ω_c) of the system. The phase margin is defined as the phase shift variation that would render the system unstable, or how large is the phase difference between the gain frequency of 0 dB (sometimes -3 dB for a closed loop system) and -180° phase at the same frequency. Ideally the phase margin should be at least 60° .

The gain margin is defined as the change in gain that would render the system unstable, or how large is the gain at the frequency of which the phase is -180° [60].

For compute the PI gains the desired control bandwidth ω_{gc} of the system and the phase margin ϕ_m are chosen, but also the sampling period T_s due to the discrete time PI implementation.

$$\begin{cases} \omega_c^* &= 950 \text{ [rad/s]} \\ \phi_m^* &= 70^\circ \\ T_s &= \frac{1}{16000} \text{ [s]} \end{cases} \quad (5.6)$$

Known these three parameters and the plant transfer function the Bode diagram is modified obtaining the gains with the following formulas:

$$\begin{cases} a &= \frac{1}{|P(j\omega_c^*)|} \\ \alpha &= \phi_m - 180^\circ - \angle P(j\omega_c^*) \\ K_P &= a \cos(\alpha) \\ K_I &= -\omega_c^* a \sin(\alpha) \end{cases} \quad (5.7)$$

The gains values are calculated:

$$\begin{cases} K_P &= 0.0632 \\ K_I &= 0.0097 \end{cases} \quad (5.8)$$

In practice the gains have been modified a little bit for adapt them to the real system.

$$\begin{cases} K_P &= \frac{64}{1024} = 0.0625 \\ K_I &= \frac{32}{16384} = 0.00195 \end{cases} \quad (5.9)$$

With the following values the real system reach the step response more slowly but the currents are less noisy.

The current loop validation is achieved by inserting the PI gains into the firmware code and by performing two current step response tests.

In first test the reference quadrature current will be set to 500 mA , while in the second to 1 mA . As we will see, the reference currents will generate a starting torque and therefore an acceleration to finally stabilize at a fixed velocity. The test was conducted only for limited quadrature current as the motor has limitation for the maximum velocity.

To compare reality with simulation, the mechanical speed ω_m and the d - q voltages are compared in Figure 5.19 and 5.20.

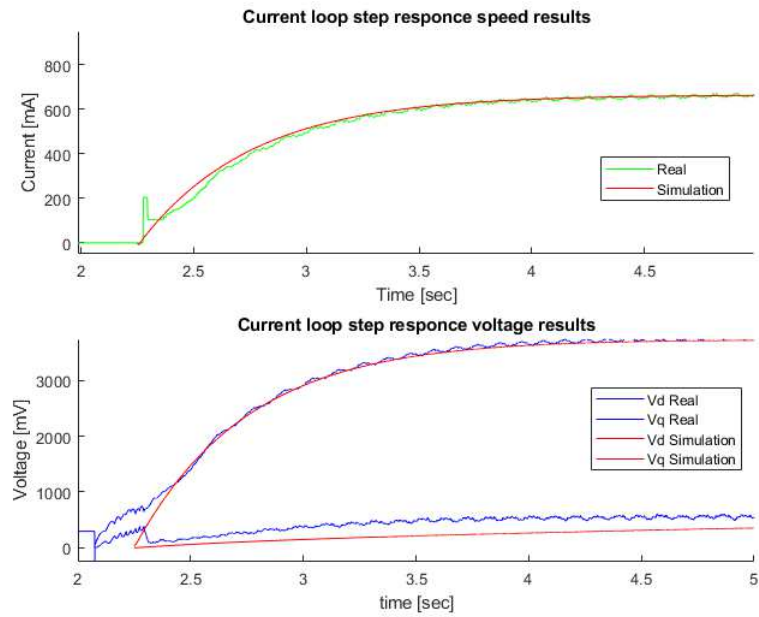


Figure 5.19: Comparison between real and simulated current loop step response with quadrature current reference $I_q = 500 \text{ mA}$

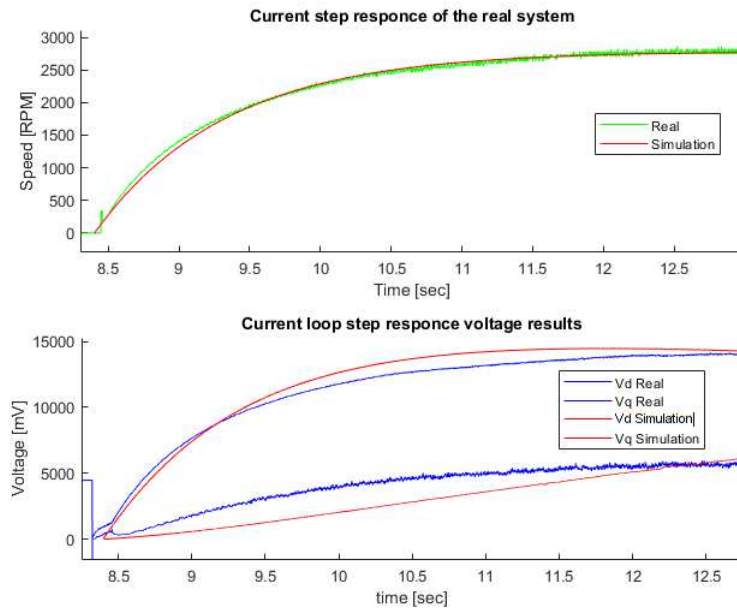


Figure 5.20: Code of the current process in open and closed loop

Both figures have an initial starting phase that introduces an error in the control and in the variable, this problem will be addressed in section 6.2. Nevertheless, the simulation follows reality quite well, the speed response is very similar and the voltages at low speed too. At high speed the voltages have an error.

To find a satisfying validation of the model, the mechanical parameter such as friction and inertia were adapted starting from the datasheet values. The best way to perform this simulation is to accurately estimate these values in order to obtain a more reliable model. This, for time purpose, has not done in the thesis.

5.2.2 SPEED CONTROL LOOP

Two different speed PI algorithms are compared in the simulation: the classic PI and the Fuzzy PI. Both controllers achieve the results but in different manner.

Fuzzy PI as explained in section 2.3.2 not simply apply a proportional, integral term, but is able to impose rules based on the FL logic. Moreover, it opens the possibility of implementing more efficient controls based on the system considered, like adaptable fuzzy PI algorithms.

The Simulink implementation of both controllers can be seen by looking at the Figure 5.21 and 5.22.

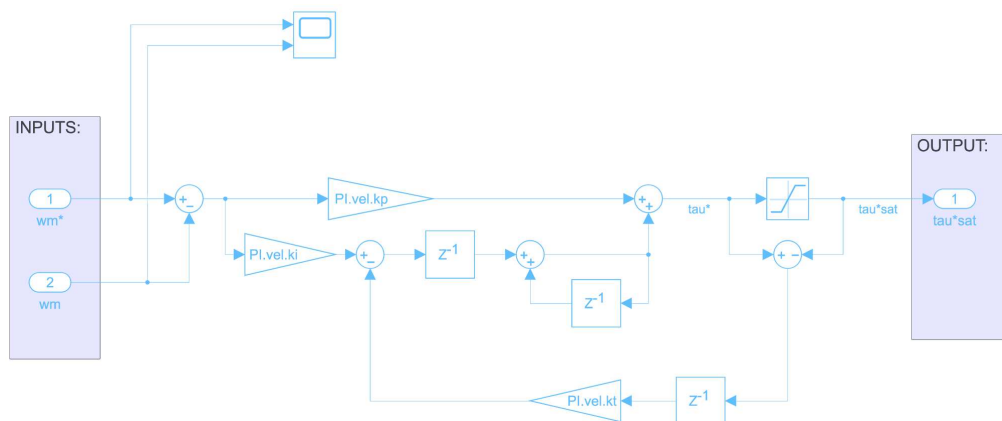


Figure 5.21: Simulation of a PI controller with anti windup mechanism

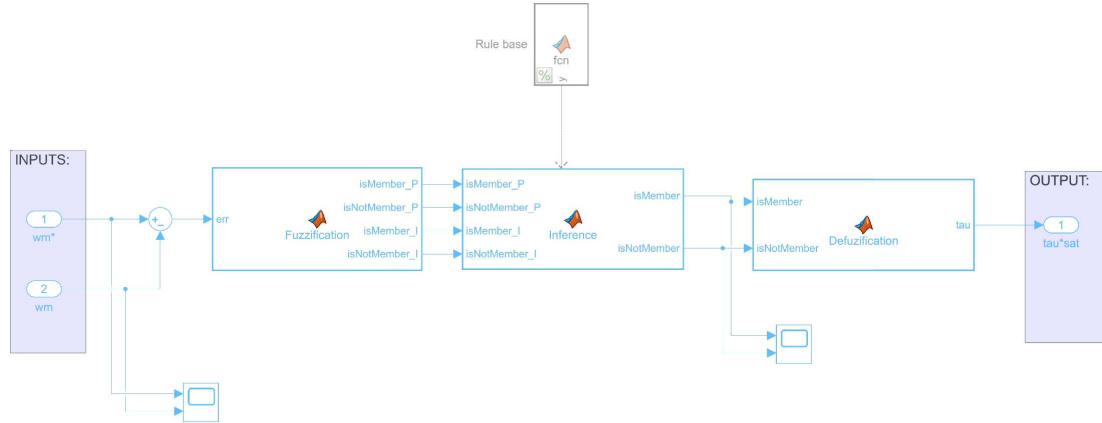


Figure 5.22: Simulation of the Fuzzy PI controller

SPEED LOOP TUNINGS

The classic PI design and tuning starts in the same way as before, i.e. by setting the control bandwidth ω_{gc} and the phase margin ϕ_m of the desired system, but also the sampling period T_s due to the discrete time PI implementation.

$$\begin{cases} \omega_{gc} &= 110 \text{ rad/s} \\ \phi_m &= 60^\circ \\ T_s &= \frac{1}{1000} \text{ Hz} \end{cases} \quad (5.10)$$

The plant to consider has changed. Now, the process to be controlled is the close-loop transfer function of the current loop, calculated earlier, plus the transfer function of the mechanical part. If necessary, the latter function must also consider the load contribute on the motor side. The transfer function of the current loop can be simplified by considering only a first order system that has a pole in its crossover frequency ω_{Bi} .

$$V(s) = \frac{1}{1 + \frac{s}{\omega_{Bi}}} I(s) \quad (5.11)$$

Adding the transfer function of the mechanical part the plant to be controlled becomes:

$$V(s) = \frac{\frac{3}{2} p \Lambda_m}{(1 + \frac{s}{\omega_{Bi}})(b + sJ)} \Omega_m(s) \quad (5.12)$$

The block diagram of the speed control loop is showed in Figure 5.23.

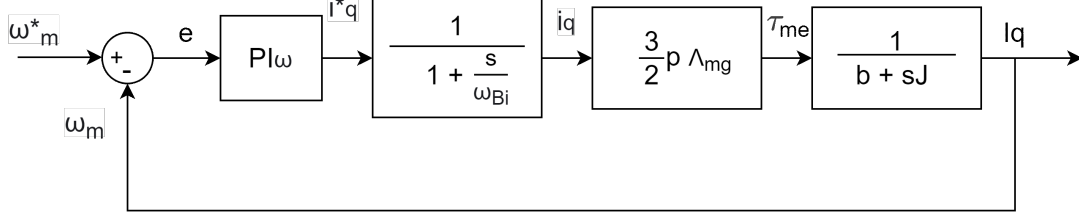


Figure 5.23: block diagram of the speed control loop

A PI regulator is chosen to have a system of type 0 (zero error at steady state) and the PI gains K_P and K_I are obtained from Equation 5.7 and specifications 5.10, resulting in:

$$\begin{cases} K_P = 0.0023 \\ K_I = 1.59e^{-5} \end{cases} \quad (5.13)$$

These values have been modified to be suitable also in the reality. The mechanical parameters have not been estimated and the inertia and friction parameters are taken directly from the collected data.

For this motivation, the adopted PI gains deviate from the simulation. They can vary based on the friction and inertia considered on the load side. In the project will be used the following values:

$$\begin{cases} K_P = 7 \\ K_I = 0.025 \end{cases} \quad (5.14)$$

The values differ significantly from the calculated ones, leading to erroneous thoughts about the malfunction of the model. By looking to the simulation and the ST Motor Control code different measurement units are used.

In the simulation usually are adopted units of a the international system of measurement. The speed error (PI input) is represented in $[\frac{rad}{s}]$ while the torque (PI output) in $[Nm]$.

In the implemented code the units are different and the speed is measured in $[RPM]$ while the PI output is directly the quadrature current in $[mA]$.

Then, to compare the simulated and real speed loop, these measurement conversions need be

considered and can be incorporated in the PI gains, yielding:

$$\begin{cases} K_P &= 7 \cdot \frac{K_T}{1000} \cdot \frac{60}{2\pi} = 0.0034 \\ K_I &= 0.025 \cdot \frac{K_T}{1000} \cdot \frac{60}{2\pi} = 1.2e^{-5} \end{cases} \quad (5.15)$$

where K_T is the torque constant and its values is: $K_T = \frac{3}{2}p\Lambda_{mg}$.

The two speed loops are then run and compared with a speed step response of 1500 *RPM* obtaining the results in Figure 5.24.

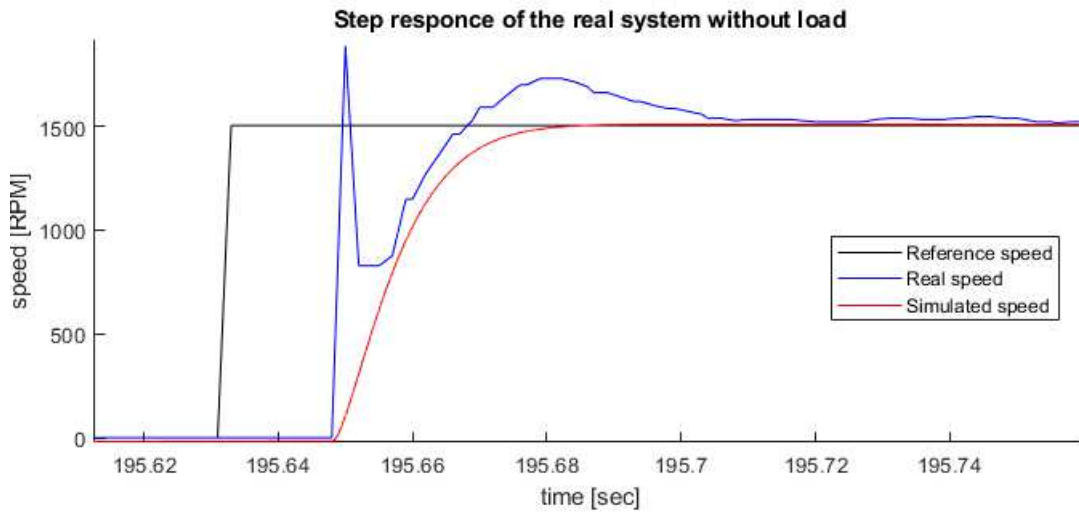


Figure 5.24: Comparison between real and simulated current loop step response with quadrature current reference $I_q = 500 \text{ mA}$

Despite the similarity of the PI gain and current loop, the two curves are quite different. The motivation can be given to the starting error of the speed loop, as it starts with a few milliseconds earlier. This delay accumulates an error that leads to a small overshoot. Furthermore, the absence of an accurate estimation procedure for the mechanical parameters increases the imprecision.

With these two problems solved the two step response of Figure 5.24 could be very similar as they already have several similarities.

PI AND FUZZY PI SIMULATION COMPARISON

This section shows the step response obtained with the classic PI speed loop and the implemented Fuzzy PI.

The two speed control methods will be simulated under the same conditions to highlight the properties of each controller. The strategy is to excite the system with a set of step input reference values that will range from rated speed of 1500 *RPM* for the test motor, down to 1000 *RPM* and up again to 1300 *RPM*. If tuned appropriately the two implementations can lead to very similar results as in Figure 5.25.

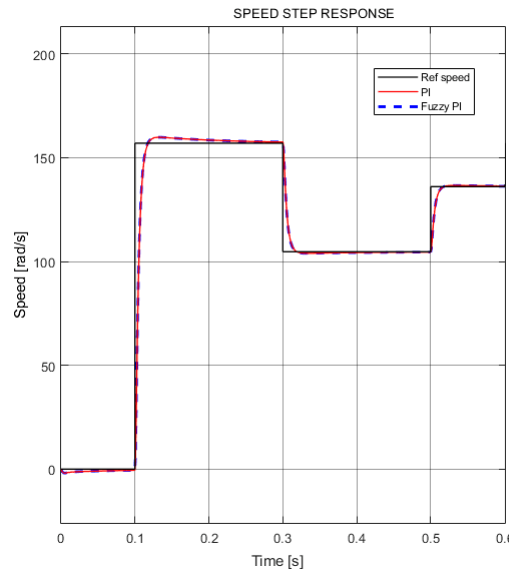


Figure 5.25: Speed step response comparison between classic PID and fuzzy PID

5.2.3 FEEDBACK CHAIN

The FOC control must sense the motor currents I_{dq} , the velocity ω_m and the electrical angle θ_{me} to have proper functionality.

Current monitoring is performed by a current sensing circuit and an ADC. Its simulation is not implemented but the currents are directly taken from the electric motor model, obtaining the currents I_{dq} . In reality the ADC takes as input the phase currents I_{abc} so a Park and a Clarke transformation block is inserted.

For simulate the current ADC is used a rate transition block and a quantization block. The

first to perform the sampling with $T_s = T_{PWM} = \frac{1}{16000}$, while the second for quantize the signal with resolution $q = \frac{FS}{2^{Nb}} = \frac{20}{2^{12}}$; where FS is the full-scale current range and Nb the number of bits of the ADC.

Then, the digital currents are transformed back to I_{dq} with a reverse Park and Clarke block to close the current feedback loop or used as inputs into the LO.

To simulate the sensed feedback speed chain, performed by digital Hall sensors, a quantization and sampling block and a FIR (Finite Impulse Response) filter are used to calculate velocity..

The sampling rate used is the same as for the speed loop so 1 ms , while the resolution is given by: $q = \frac{2\pi}{3 \cdot 2 \cdot p}$. When the motor is moving the resolution could be considered higher as are used algorithms for compute the angle between the events. The Figure 5.26 will show the Simulink feedback chain.

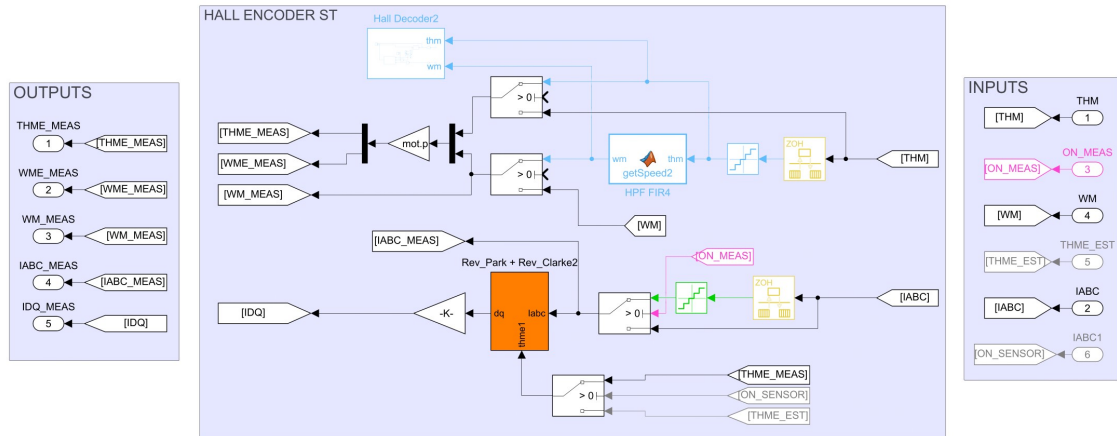


Figure 5.26: Implementation of the feedback chain in the simulation environment

Out of curiosity, some STMicroelectronics codes for the Hall sensor management has been inserted in Simulink. This permit to visualize the Hall signal and the computation of the sector directly from the simulation without performing the real test. The signals are reported below in Figure 5.27.

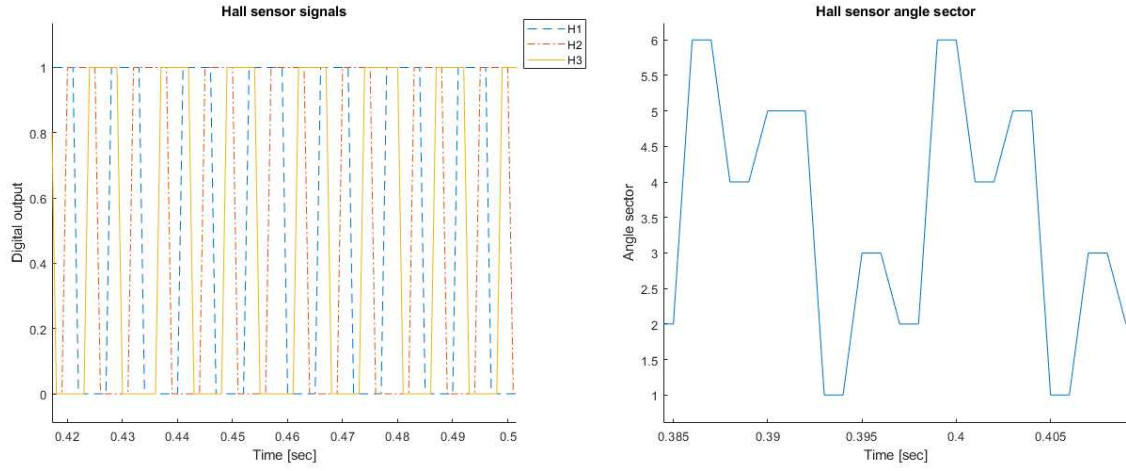


Figure 5.27: Hall sensor signals and sector while motor is running

5.2.4 SENSORLESS CONTROL DESIGN

There exists several methods for build a sensorless control algorithm but not all the algorithms are suitable for the thesis system. The adoption of a back-EMF algorithm with a LO for medium and high speeds has been widely discussed and motivated in section 2.3.3.

In the thesis the LO is implemented and designed in two different references the stationary α - β and the rotating d - q frame. The work was done for compare the two different implementation and to adapt the simulation to different software brands that implements both the techniques. In Figure 5.28 are reported the two block scheme representations that gives an immediate graphic comparison. The first thing to do to design a good discrete LO is to verify if a discrete observer exists. By the asymptotic observer theorem the following facts are equivalent:

1. \exists an asymptotic observer, namely $\exists L \in \mathbb{R}^{n \times p}$ such that $F - LH$ is asymptotically stable.
2. Either (F, H) is observable or (F, H) is not observable but the non observable part is asymptotically stable.
3. PBH - Observability matrix satisfies: $\text{rank}[\frac{\lambda I_n - F}{H}] = n, \forall \lambda_i \in \mathbb{C}, |\lambda_i| \geq 1$

where λ_i are the discrete eigenvalues, F, H, G are the discrete matrices of the continuous-time state space system A, B, C and L is the observer control matrix for the discrete system.

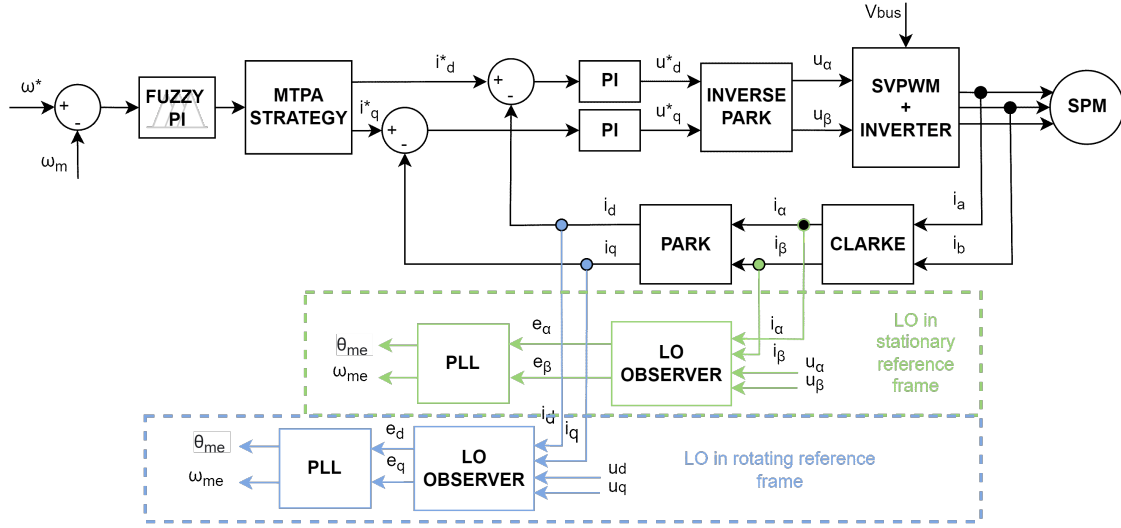


Figure 5.28: Sensorless block scheme for α - β or d - q reference frame

From the theorem is sufficient to compute the observability matrix Ob and check if (F, H) is observable as reported below.

$$Ob = \begin{bmatrix} H \\ FH \\ FH^2 \\ \dots \\ FH^{n-1} \end{bmatrix} \quad (5.16)$$

Then the discrete-time system is observable if the number of non observable states, $unob$, are equal to zero.

$$unob = length(F) - rank(Ob) = 2 - 2 = 0 \quad (5.17)$$

Having verified that the system is observable than $\exists L$ such that $(F - LH)$ is asymptotically stable and the estimation error converges to zero when $(F - LH)$ has its eigenvalues inside the unit circle. Therefore, the value of L should be such that this goal is achieved.

By imposing the characteristic polynomial of the observer equal to the wanted characteristic polynomial the control matrix L can be computed using the Equation 5.18.

$$det(\lambda IF + LH) = (\lambda_1 + c)(\lambda_2 + d) \quad (5.18)$$

where c and d are the desired eigenvalues.

In the thesis the eigenvalues are defined as $c = d = 0.75$ obtaining the resulting observer gain

$L = [0.3835; -0.0698]$ for the LO implemented in the d - q frame.

Produced a good estimation of the back-EMF the PI of the PLL scheme is tuned manually for estimate the electrical speed ω_{me} . Usually the speed estimation is accompanied with a discrete filter of Equation 5.19 for attenuate the high frequencies, while the electrical angle θ_{me} is estimated using a discrete integrator.

$$H(s) = \frac{\alpha}{1 + (\alpha - 1)z^{-1}} \quad (5.19)$$

The values used for design the PLL are reported below:

$$\begin{cases} K_P &= 1600 \\ K_I &= 5 \\ \alpha &= 0.03 \end{cases} \quad (5.20)$$

The procedure for designing the sensorless observer in the α - β reference is very similar to the d - q reference, the only difference are the system matrices reported in Table 2.2 and the option to insert a filter between the PLL and the Observer.

Above, in Figure 5.29 and 5.30, are reported the two algorithms implemented in Simulink.

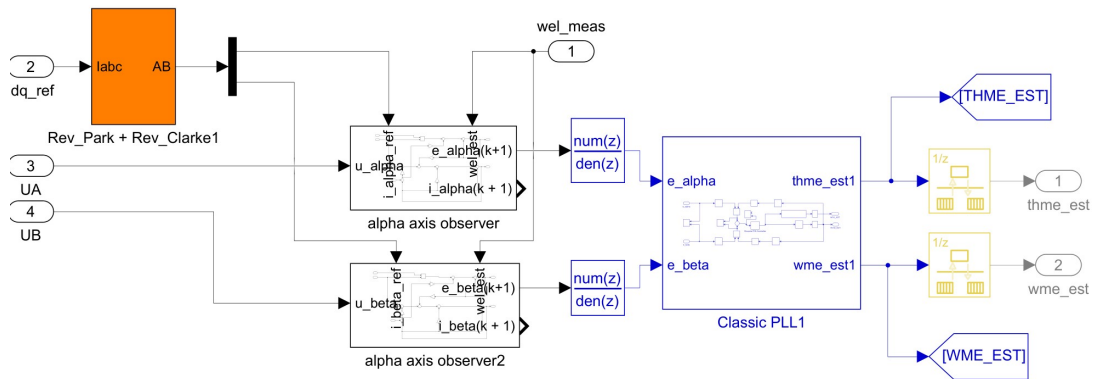


Figure 5.29: Implementation of the sensorless speed and angle estimation in the stationary reference frame α - β

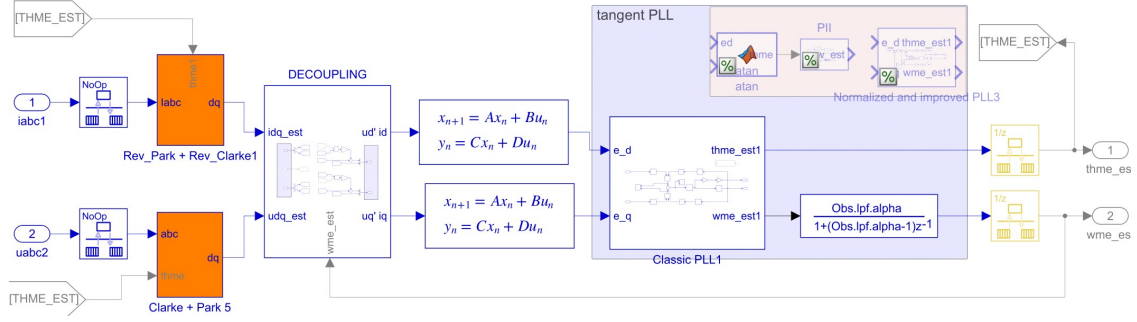


Figure 5.30: Implementation of the sensorless speed and angle estimation in the rotating reference frame d - q

5.2.5 BACK-EMF LUENBERGER OBSERVER α - β AND d - q

Before comparing the sensed and the sensorless estimation algorithms is interesting to shift the focus only to the sensorless algorithms.

In the project both the implementations of the LO are valid, they achieve the speed reference and angle estimations as it is shown in Figure 5.31 and 5.32.

However, in the rotating reference, it is not possible to guarantee that the currents and voltages measured in the synchronous reference are correct, since the estimated position is used in the Park transformations and can lead to a phase error in the observed synchronous variables. Thus, despite the observer having a lower order in the rotating frame, the LO in the synchronous frame is affected by both the speed and position estimation errors, while the LO in the stationary frame is only affected by speed estimation error [24].

As a consequence, often the implementation choice falls to the α - β LO that is less computational expensive because it does not need to perform the Park transformation from $I_{\alpha\beta}$ to I_{dq} . In the case of the observer in the stationary frame is more complex as deals with time varying quantity and it can present a defect in the integration of the rotor speed for position estimation that can lead to integration shift phenomenon [25].

Nevertheless, both back-EMF algorithms have low efficiency at low speeds and it is essential to implement a start-up phase.

In Figure 5.31 and 5.32 are showed the results obtained with a speed step response of 100 rad/s with torque disturbance injection for the back-EMF observer in the stationary reference frame and the observer in the rotating reference frame. From a first comparison the d - q implementation presents better results since it has a small steady-state error, a lower rise and

settling time despite a slightly higher overshoot. Don't forget that both the PLL have been tuned manually and the better results can be achieved, therefore, the equivalence between the LO in the stationary and synchronous frame can be proof as reported in [26].

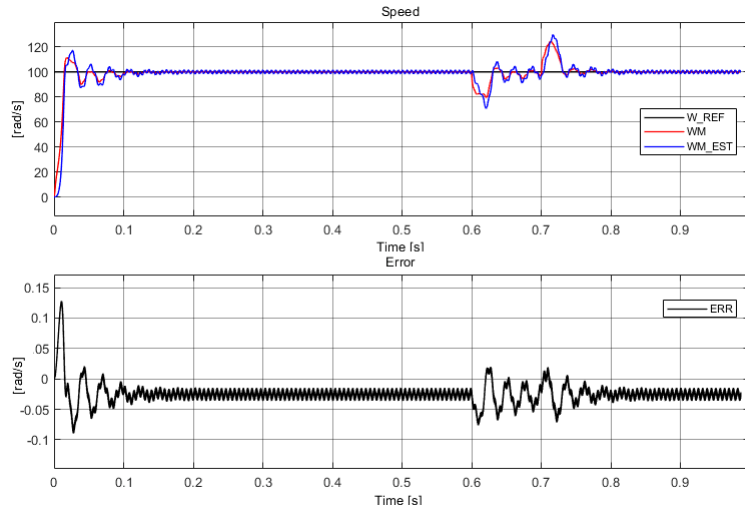


Figure 5.31: Speed step response and estimated angle error for the LO implemented in the $\alpha\beta$ reference

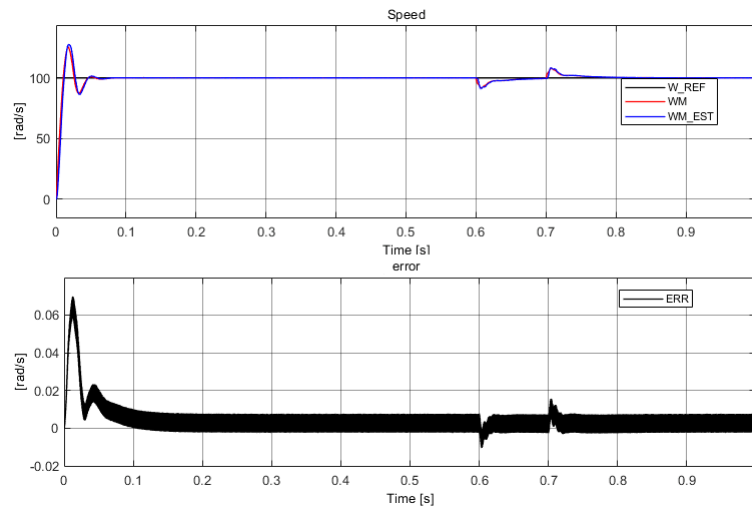


Figure 5.32: Speed step response and estimated angle error for the LO implemented in the rotating reference frame

5.3 SENSORLESS AND SENSORED SIMULATION RESULTS

By looking to the simulation results of the sensed feedback chain (Figure 5.33) and the results of the sensorless feedback chain (Figure 5.31 and 5.32) are immediate the conclusions.

The sensed feedback chain is faster, does not present big overshoot and have a zero steady-state error. Moreover, it is reliable also at low speeds and it is able to handle torque disturbances. Sensorless control can be improved by adding an open-loop start-up phase that leads to a concrete product acting good at low speeds. But the sensor's performance will still be superior, as it always includes closed loop control that is more accurate and more robust to noise contribute. Estimation technique can still be used, they can support the sensed control and it is from some years a big research topic.

In Figure 5.33 are shown the results obtained with the Hall sensor feedback chain with Fuzzy PI and FOC control. The simulation is performed under the same conditions of the speed PI comparison. Except for the presence of two disturbance injections at 0.6 and 0.7 seconds the amplitude of these disturbance is the 10 % of the nominal torque.

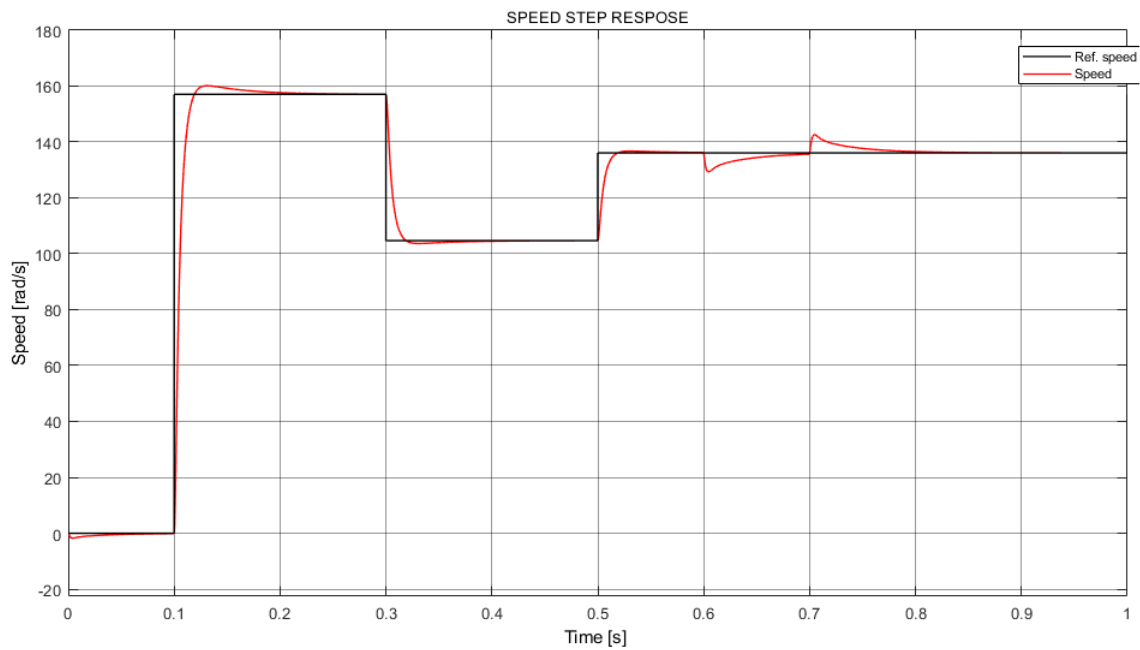


Figure 5.33: Speed step response with sensed feedback chain tuned for have the best result

6

Experimental Results

The method chosen for the test is the Fuzzy PI together with the Hall sensor speed measurement method. Based on the simulations and results the Fuzzy speed controller does not bring any particular improvement, while the Hall sensor estimation is considered the most robust and accurate solution for speed control.

In this section are reported the results obtained by applying the firmware algorithm in the mentioned hardware system.

6.1 STATIC TORQUE VERIFICATION

After having designed the overall system, it is necessary to perform some tests for ascertain the absence of inefficiencies.

In case of gate automation, the starting torque represents an fundamental parameter. Thinks about how many times you open a door in your routine, if the door is automatic it must perform several all equal openings and closings path, starting every time standing still. Static friction must not represents a problem despite the movement of heavy gates and the breakaway torque, it must overcame the resistance that depends on the environment and aging.

It is not sufficient to chose the motor based on the datasheet torque curve of Figure 2.12. The control can introduce inefficiency, for instance: if the Hall offset angle is wrong the total FOC algorithm works with inefficiency considering the torque axis q not in the perfect position. This is just an example, but many of these problems can be present. To avoid them, a starting

torque verification is carried out.

The best way to conduct this tests is to use a precise digital torque transducers, by the way, if it is not available, also indirect measurements or a graduated rod can be used.

The test is conducted with a graduated rod fixed to the motor shaft, giving different current values in I_q . By inserting weights in the rod is possible to draw the curve.

To confront the curve with the datasheet it is necessary to compute the I_{line_RMS} current with the formula:

$$I_{line_RMS} = \frac{|I_a| + |I_b| + |I_c|}{3} \quad (6.1)$$

In the Table 6.1 and 6.1 are summarized the test results. The motor has a higher range of current I_{line_RMS} but the drive board X-NUCLEO-IHMo8M1 has some limitations. The maximum output current is $A_{RMS} = 15 A$ and the over-current detection and protection starts at $A_{Peak} = 30 A$.

For this reason the test cannot be performed for the whole motor current range. The results are sufficient to state that the system respect the motor torque parameters and a good level of efficiency is reached.

$I_q [A]$	$I_{line_RMS} [A]$	$\tau_s [Nm]$
5000	3.37	0.15
10000	6.67	0.3
15000	9.97	0.46
20000	13.29	0.62
25000	16.58	0.75
30000	19.80	0.86
31000	20.54	0.94

Table 6.1: Tabulad data from the starting torque test

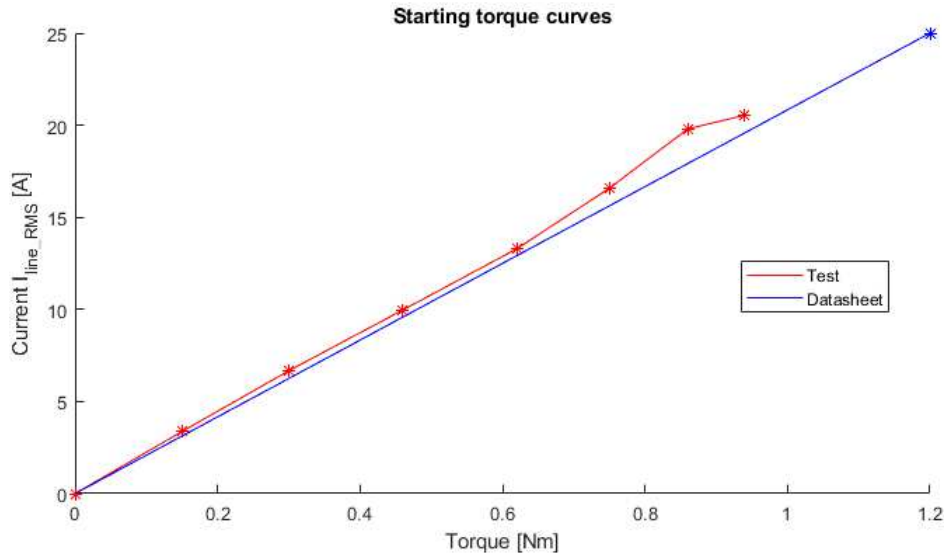


Figure 6.1: Test and datasheet curves of the start-up torque

6.2 FIRMWARE TEST RESULTS

The verification strategy is based on the robustness and the ability to accurately converge on the reference level within an acceptable time. The simulations shows that the Hall sensor solution should be able to handle step input, disturbance and a varying load over a wide range of chosen reference levels. The reference levels used in the tests are the same of the simulation with a step and an exponential input.

The goal will be to show that the FOC Hall sensor solution will be able to handle step reference level changes with a varying mechanical load attached to the rotor shaft. The varying mechanical load is applied manually by imposing a pressure in the mechanical load shaft after the gearbox.

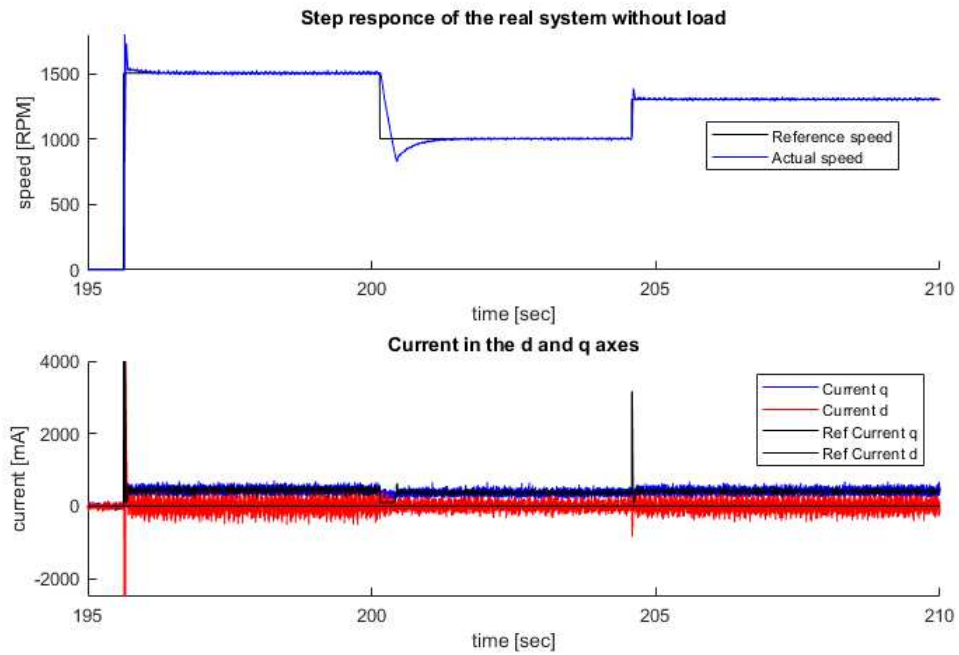


Figure 6.2: Speed and current step response without load

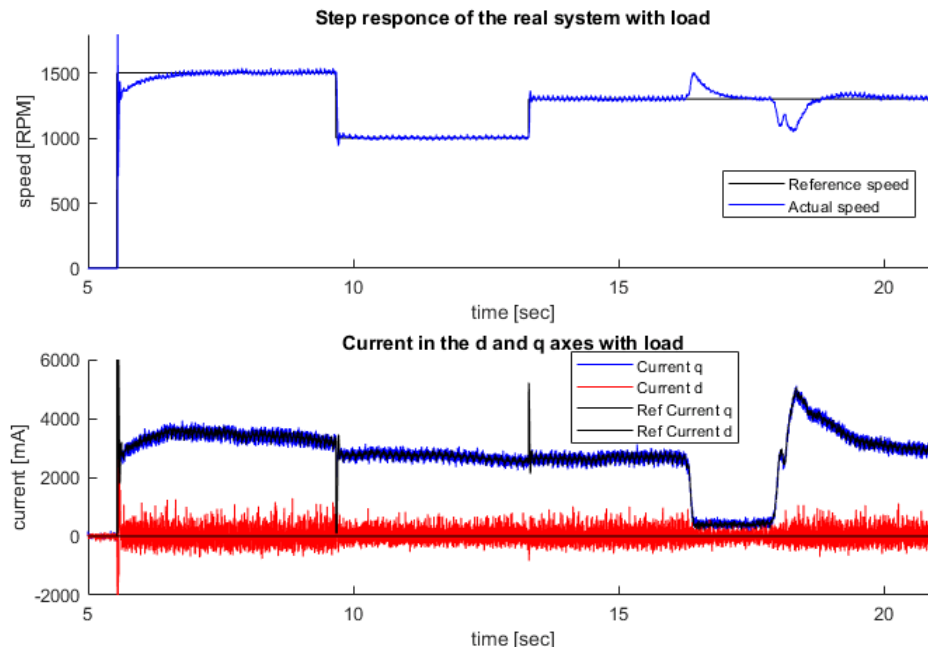


Figure 6.3: Speed and current step response with load

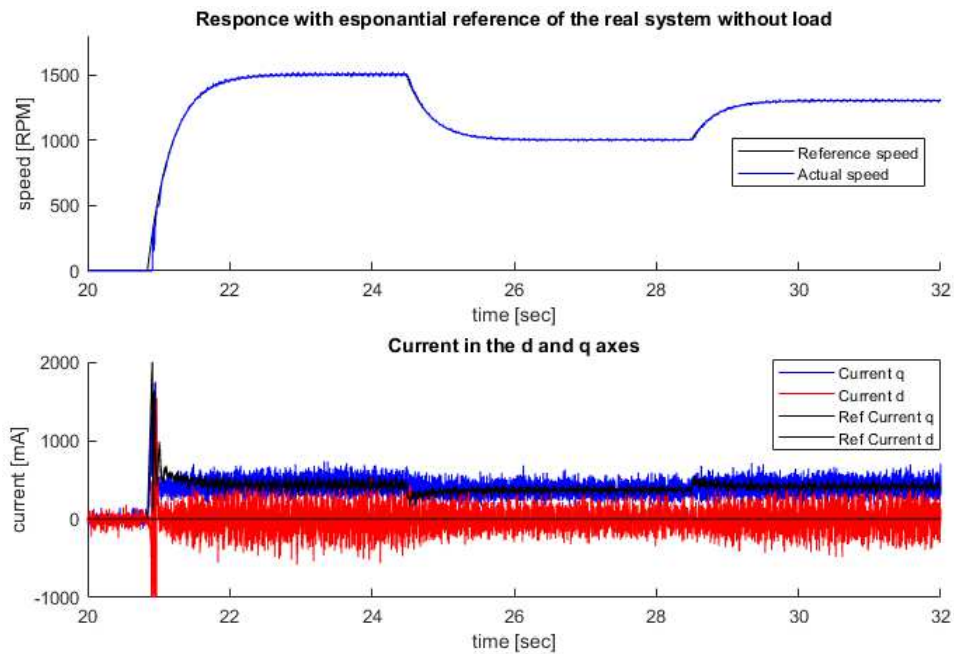


Figure 6.4: Speed and current response with growing reference without load

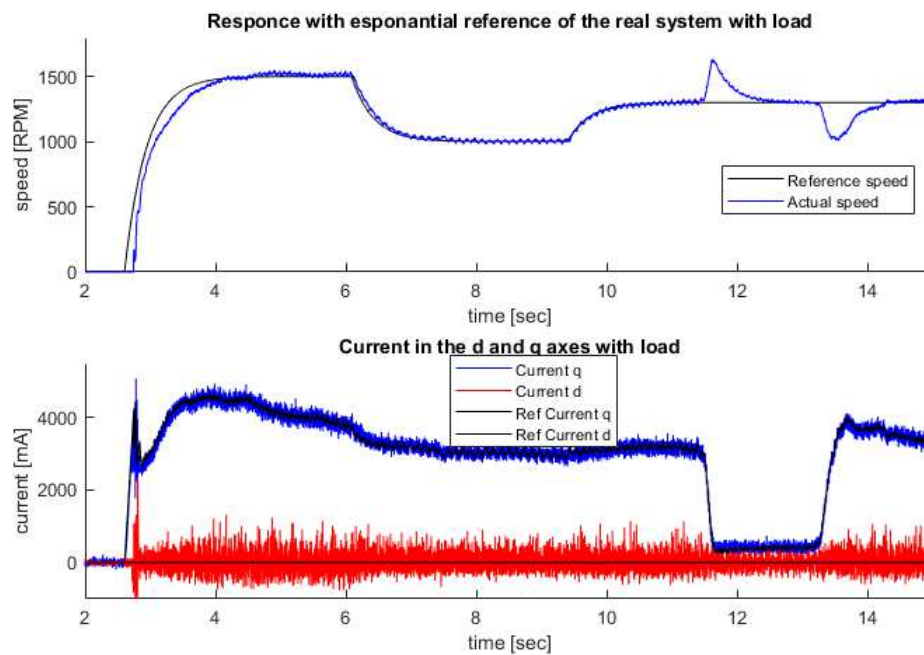


Figure 6.5: Speed and current response with growing reference with load

From all the figures can be notice an error in the starting phase of the motor. Tests display an

excessive overshoot. In reality this value is due to a starting error in the speed estimation that has a low accuracy in the beginning.

From the detailed Figure 6.6 that reports the step response of Figure 6.2, it shows the speed loop computation with 20 *ms* of delay with respect to the reference speed. This is due to the current speed variable which remains zero as it needs at least two Hall signal samples for produce an estimate of the electrical speed ω_{me} . This can be seen from the electrical angle ω_{me} , green signal.

The delay leads to a larger accumulation error that contributes to the overshoot.

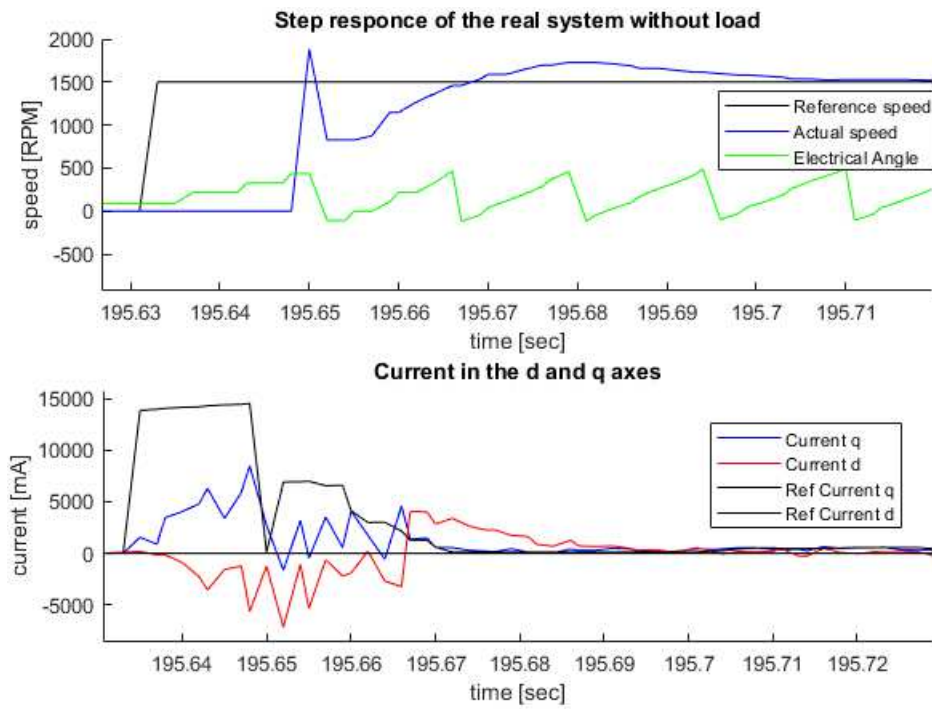


Figure 6.6: Zoom of the speed and current step response in no load conditions

Another peculiarity is present in the step response with no load. The plot displays that the speed does not follow a reference falling speed as well as a reference rising speed. The cause is due to a minimum reference of the current $I_q = 100 \text{ mA}$.

This fact is not present in the other tests since the minimal current is never reached if a load is attached or if a ramp reference speed is assigned. But it is necessary for have a smoother braking phase.

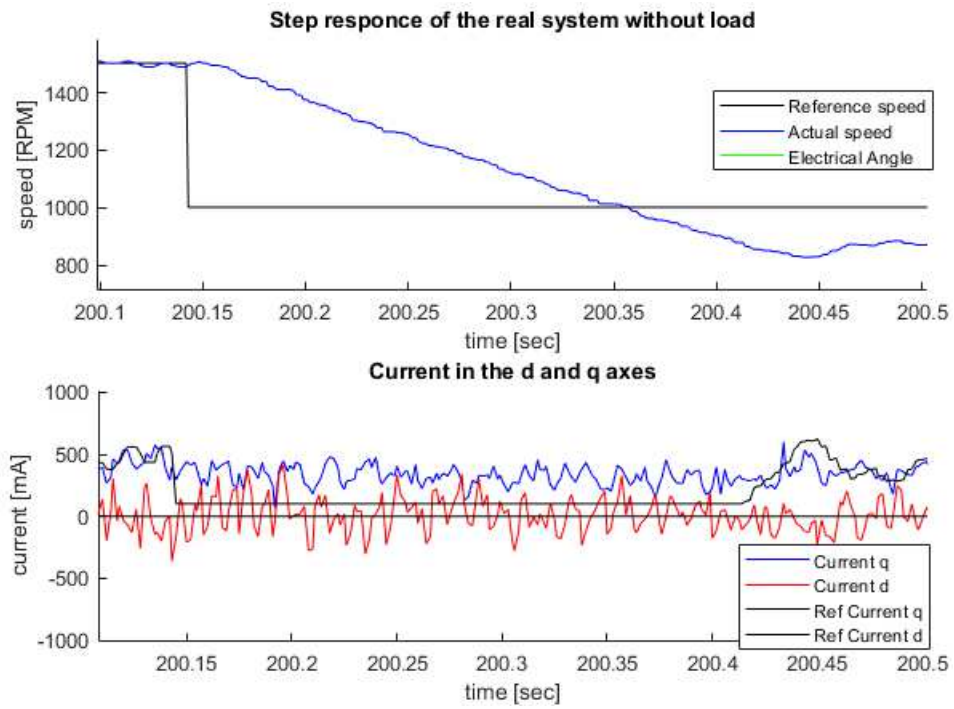


Figure 6.7: Zoom of the speed and current step response in no load conditions in the slowdown phase

7

Conclusions and problems

The thesis is concluded, a vector gate driver control is implemented in firmware and it has been designed accordingly with the system simulation.

In order to be able to carry out this project, various topics were touched upon. Starting from the theory of chapter 2, here are reported the basics of SPM motors and control theory, with particular attention to the field-oriented control algorithm. The choice of this vector control algorithm has been motivated and favorite with respect to direct torque control or voltage vector control.

FOC needs to constantly monitoring the electrical angle to have a correct functionality, as a consequence, the feedback chain has a big impact on the control. Sensored and sensorless feedback have been compared highlighting the respectively advantages and problems.

In the same chapter of the theory is present a different section, but with the same function of the theory. The parameter estimation phase is an essential task for verifying datasheet values and it produces missing reliable parameters. Errors in the estimation or in the measurement will be inevitably lead to a degradation of the control performance. The stator resistance, the stator inductance, the flux permanent magnets and the Hall offset have been estimated through several tests, obtaining for the stator resistance and inductance similar values of the reported datasheet. The flux permanent magnets has not been compared with the datasheet, while the Hall offset is a data that is usually not present.

Finding a correct test for the estimation of the Hall offset while the motor is not in debug mode has arise some problems. There are only few experiments in literatures. The one used produced

a sufficient result near to the optimum offset.

The hardware components were all supplied by the company and numerous modification and connections were required for set-up the structure of Figure 3.13.

The main drawback of this set-up is that the electric system was rather delicate. An error in firmware or a contact in the board can compromise the functionality of the system. From this project: four IHMo8M1, three STM32F401RE and two Hall encoder have been replaced for malfunctions and damages.

The components substitution due to poor supply took some times. Devices such as the Hall sensor have delivery delays of up to six months. Then, it is necessary to purchase the system several months before and have enough components available.

Even in the mechanical part a problem was found and investigated. The positioning of the new brushless motor was not exactly the same as the previous induction motor and a mechanical tolerance was not respected. By closing the aluminium cage with the screws, a plastic component needed for the stability of the motion pushes the crankshaft downwards, not allowing a correct transmission. The study of the ST code and the implementation of the firmware has been the core section. Understanding the entire STMicronics Motor Control Library and get into the programmer's mindset is a challenging task.

In this section, after the study phase and the association of the theory, the motor control library was exported and implemented in the company FDK where the fuzzy speed loop and the Application file of the ATS product were already present. This existing file has been expanded by inserting: the FOC algorithm, the Hall sensor management file, the motor control library and a new driver section as discussed in chapter 4.

Obviously, from the debug phase different problems have been encountered. The main one concerns the Hall sensor management file which produces an error in the control. This issue is addressed thanks to the FDK architecture. Using a different Hall sensor file a correct speed estimate have been produced.

Other smaller problems were encountered. Particular attention should be paid to: peripheral priority assignment, direction handling, ADC calibration, Hall offset, and fuzzy PID tuning. The overall system used with the ST code has been modelled and simulated. Starting from the plant, characterized by the electric motor and the inverter, several open-loop tests were conducted for validate the model with the estimated parameters, Figure 5.7, 5.8, 5.9 and 5.13. In the validation some differences are present and motivated. This mismatch compromises the overall accuracy of the model and some other measurements or a non linear flux model have to be implemented for resolve it.

From the plant, the control can be structured. The FOC algorithm was chosen as control methods and the current and speed PI has been tuned using MBD methods. The PI gain obtain are sufficient but for adapt them to the real system small modification have been performed.

One of the objectives of the thesis is to simulate and compare the two main feedback chain algorithms: the sensed with Hall sensor and the sensorless. For this purpose, both techniques were simulated, focusing on the back-EMF LO sensorless design in $d-q$ and $\alpha-\beta$.

The LO in the stationary reference frame is usually preferred over the $d-q$ implementation, since it does not depend on the estimated position used in the Park transform. Nevertheless, both algorithms have advantages and disadvantages and their equivalence can be demonstrated.

Sensed and sensorless feedback chains are compared and modelled to understand the differences and the possibilities. From the data, the most reliable and performing control is the FOC with Hall sensors. The back-EMF $\alpha-\beta$ LO sensorless algorithm is not reliable at zero/low speed and a starting phase method must be adopted.

The sensorless results remain interesting and highlight the possibility of increasing the safety and the accurateness of the electrical speed and angle. In fact, these two methods can be combine and in case of a Hall encoder failure the sensorless control can guide the gate in safe conditions.

By implementing a open-loop starting procedure, sensorless control can be exploited and implemented for gate drives. There are already several automations for gates with open loop control on the market. Hence, these devices can be improved with this technology.

In addition, the simulation opens up the possibility to quickly and safely test different control algorithms and other research.

In the last section, the firmware control and implementation has been tested, reaching the final goal of the thesis.

First of all, an estimate of the starting torque was necessary to verify the presence of inefficiency by comparing the results with the datasheet. The test highlight a very similar starting torque, obtaining an acceptable result. Then, a speed step response and the actual reference used in the product are tested.

The results showed correct control with and without a load, but also with the injection of positive and negative torque disturbance.

The only problem in this section has already been addressed in section 6.2 and concerns the delay of 20 ms indispensable for start the speed loop.

7.1 FUTURE WORKS

Research and design can be deepened and many other implementations and improvements are possible.

In order with the structure of the thesis, other measurements and estimates can be made to obtain more precise values for the parameter used in the simulation. It is possible to built a professional engine test bench or a FEM model to generate, with some post process analysis, a non-linear model of the motor fluxes.

The simulation is already adapted for handle this non-linear model. It allows to consider saturation and cross coupling effect achieving a more reliable simulation.

In the firmware other peripheral can be implemented. A DAC will be interesting for monitoring physical quantities with a high sampling period. Another ADC could be used to implement regular sampling of the bus voltage and of the motor temperature to avoid damages.

In the future, two UARTs will be inserted to communicate with an already existing board and print status and errors. The UART for the communication will adopt a master-slave protocol, where the master CAME board will pass to the thesis system (slave): a reference speed, an action time and some obstacle detection parameters. These passed values will depends on the function selected by the master CAME board, which incorporates all the functions used in a normal CAME gate device.

The slave will respond to the master massage with its status and in case of errors the motor will stop along with the communication. This communication is produced to avoid implementing the hardware from scratch and being able to give and receive instructions with a CAME control panel.

Another firmware improvement could be to tune and test the sensorless code already implemented in the firmware. By adding to the software the possibility to change the control in the Application file.

The latest future focus in the firmware, is to move from a Cortex®-M4 to a Cortex®-Mo or Mo+. The last MCU is already implemented in the CAME applications, is smaller, less expensive and with the best energy efficiency, but has lower power.

For the simulation the non-linear flux maps, discussed before, can be inserted and tested verifying if the encountered problem in section 7 will be solved.

The simulation can be exploited to implement other control algorithms such as: HFI algorithms for IPM motors, thermal models for parameter variations, more advance state estimation methods, adaptable Fuzzy logic controllers and several other research topics.

Do not forget that all these methods must be implemented in a industrial scenario where the safety must be perfect and the costs as small as possible. Finding a correct way to implement those technologies in the product is crucial, otherwise the research are meaningless.

References

- [1] A. Moretti and R. Tabacco, *I Menuzzo e Came*. Editoriale Scientifica, 2020.
- [2] N. Bianchi, *Calcolo delle Macchine Elettriche col Metodo degli Elementi Finiti*. Cleup, 2004.
- [3] —, *Vehicular Electric Power Systems: Land, Sea, Air, and Space Vehicles*. Cleup, 2004.
- [4] M. K. James Widmer, Richard Martin, “Electric vehicle traction motors without rare earth magnets.” *Sustainable Materials and Technologies*, vol. 3, pp. 7–13, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.susmat.2015.02.001>
- [5] Microsemi, *Park, Inverse Park and Clarke, Inverse Clarke Transformations MSS Software Implementations User Guide*, 2013. [Online]. Available: <https://www.microsemi.com>
- [6] S. H. L. Kumar Sanjeet, Michael Dwivedi, “Voltage vector based control for pmsm in industry applications.” *IEEE International Symposium on Industrial Electronics*, 2010. [Online]. Available: <http://dx.doi.org/10.1109/ISIE.2010.5637742>
- [7] P. Vas, *Sensorless vector and direct torque control*. Oxford Science, 1998.
- [8] A. T. X. Garcia, B. Zigmund, “Comparison between foc and dtc strategies for permanent magnet synchronous motors.” *Advances in Electrical and Electronic Engineering*, vol. 5, pp. 76–81, 2011. [Online]. Available: <http://advances.utc.sk/index.php/AEEE/article/view/179/203>
- [9] M. Laboratories, “From pid to fuzzy control: Taking the plunge cautiously,” 2004. [Online]. Available: <https://www.mstarlabs.com/control/fuzzypid.html>
- [10] G. Viot, “Fuzzy logic in c,” *Dr. Dobb’s Journal*, 1993.
- [11] S. Bolognani, S. Calligaro, and R. Petrella, “Design issues and estimation errors analysis

- of back-emf-based position and speed observer for spm synchronous motors,” *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 2, 2014. [Online]. Available: <http://dx.doi.org/10.1109/JESTPE.2013.2296974>
- [12] T. Liu, “Parameter identification of pmsm with considering nonlinearity of the inverter,” Ph.D. dissertation, Darmstadt University, 2021. [Online]. Available: <https://d-nb.info/1249506638/34>
- [13] C. W. T. Liu, Z. Ye, “Fpga-based voltage measurement using delta-sigma modulation for a pmsm drive,” *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 1066–1071, 2019. [Online]. Available: <http://dx.doi.org/10.1109/IECON.2019.8927273>
- [14] G. Shen, W. Yao, B. Chen, K. Wang, K. Lee, and Z. Yu Lu, “Automeasurement of the inverter output voltage delay curve to compensate for inverter nonlinearity in sensorless motor drives,” *IEEE Transactions on Power Electronics*, vol. 29, pp. 5542–5553, 2014. [Online]. Available: <http://dx.doi.org/10.1109/TPEL.2013.2293134>
- [15] M. A. D. Zammit, C. Spiteri Staines, “Compensation techniques for nonlinearities in h-bridge inverters,” *Electrical Systems and Information Technology*, vol. 3, pp. 361–376, 2016. [Online]. Available: <https://doi.org/10.1016/j.jesit.2016.07.007>
- [16] M.-C. Kang, S.-H. Lee, and Y.-D. Yoon, “Compensation for inverter nonlinearity considering voltage drops and switching delays of each leg’s switches,” *2016 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1–7, 2016. [Online]. Available: <http://dx.doi.org/10.1109/TPEL.2013.2293134>
- [17] STMicroelectronics, *STM32F401xB/C and STM32F401xD/E advanced Arm®-based 32-bit MCUs*, 2018. [Online]. Available: <https://www.st.com>
- [18] —, *Getting started with X-NUCLEO-IHM08M1 low-voltage BLDC motor driver expansion board based on STL220N6F7 for STM32 Nucleo*, 2017. [Online]. Available: <https://www.st.com>
- [19] —, *STM32F PMSM single/dual FOC SDK v4.3*, 2016. [Online]. Available: <https://www.st.com>
- [20] CAME, *Manuale di installazione ATsxoAGx*, 2020. [Online]. Available: <https://www.st.com>

//www.came.com

- [21] S. Punnekkat, “Event-based messaging architecture for vehicular internet of things (iot) platforms,” Master’s thesis, Malardalen University, 2017.
- [22] A. G. Marco Tursini, Alessia Scafati and R. Petrella, “Extended torque-speed region sensor-less control of interior permanent magnet synchronous motors,” *2007 International Aegean Conference on Electrical Machines and Power Electronics*, 2007. [Online]. Available: <http://dx.doi.org/10.1109/ACEMP.2007.4510583>
- [23] S.-H. Kim, B. dong Jeong, D.-O. Yoon, S.-G. Lee, and S.-G. Oh, “Sensorless speed control of induction motor using model reference adaptive control and direct torque control system,” *The Journal of the Korean Institute of Information and Communication Engineering*, vol. 16, pp. 2708–2715, 2012. [Online]. Available: <http://dx.doi.org/10.1109/SPEEDHAM.2008.4581122>
- [24] C. J. V. Filho, D. Xiao, R. P. Vieira, and A. Emadi, “Observers for high-speed sensorless pmsm drives: Design methods, tuning challenges and future trends,” *IEEE Access*, vol. 9, pp. 56 397–56 415, 2021. [Online]. Available: <http://dx.doi.org/10.1109/ACCESS.2021.3072360>
- [25] B. Li and L. Li, “New integration algorithms for flux estimation of ac machines,” *2011 International Conference on Electrical Machines and Systems*, pp. 1–5, 2011. [Online]. Available: <http://dx.doi.org/10.1109/ICEMS.2011.6073405>
- [26] S. Po-ngam and S. Sangwongwanich, “Stability and dynamic performance improvement of adaptive full-order observers for sensorless pmsm drive,” *IEEE Transactions on Power Electronics*, vol. 27, pp. 588–600, 2012. [Online]. Available: <http://dx.doi.org/10.1109/TPEL.2011.2153212>