# EMBEDDED FIELD-ORIENTED CONTROL (FOC) BASED ON STM32 FOR THREE-PHASE INVERTER-FED PERMANENT MAGNET SYNCHRONOUS MOTOR (PMSM) DRIVES WITH REAL-TIME HARDWARE-IN-THE-LOOP (HIL) SIMULATION

THESIS

**SUSTAINABLE DEVELOPMENT GOALS**
**Affordable and Clean Energy**
**Industry, Innovation and Infrastructure**
**Climate Action**


Written by:

**Raditya Prajnabuwana**
**21/479720/TK/52923**

**ELECTRICAL ENGINEERING STUDY PROGRAM**

**DEPARTMENT OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY**
**FACULTY OF ENGINEERING UNIVERSITAS GADJAH MADA**
**YOGYAKARTA**


**2025**

# ENDORSEMENT PAGE

# EMBEDDED FIELD-ORIENTED CONTROL (FOC) BASED ON STM32 FOR THREE-PHASE INVERTER-FED PERMANENT MAGNET SYNCHRONOUS MOTOR (PMSM) DRIVES WITH REAL-TIME HARDWARE-IN-THE-LOOP (HIL) SIMULATION

**THESIS**

Submitted as one of the requirements for the degree of
Bachelor of Engineering (*Sarjana Teknik*)
in the Department of Electrical Engineering and Information Technology
Faculty of Engineering
Universitas Gadjah Mada

Written by:

**Raditya Prajnabuwana**
**21/479720/TK/52923**

Has been approved and endorsed

on . . . . . .

| Supervisor I | Supervisor II |
| --- | --- |
| **Dr.-Ing. Ir. Yohan Fajar Sidik, S.T., M.Eng.** | **Ir. Eka Firmansyah, S.T., M.Eng., Ph.D., IPM., ASEAN Eng.** |
| **NIP 111199001202205101** | **NIP 197903032002121004** |

# STATEMENT

I, the undersigned:

| | |
|---|---|
| Name | : Raditya Prajnabuwana |
| Student ID (NIM) | : 21/479720/TK/52923 |
| Year of Enrollment | : 2021 |
| Study Program | : Electrical Engineering |
| Faculty | : Engineering, Universitas Gadjah Mada |

Hereby declare that this undergraduate thesis contains no part of any other academic work previously submitted for a degree at any institution of higher education. Furthermore, this work contains no work or opinions written or published by others, except where explicitly cited and fully referenced in the bibliography.

I further affirm that this academic work is free from any form of plagiarism. If, in the future, this thesis is proven to contain plagiarized material or to have deliberately presented others' work or ideas as my own, I am willing to accept any academic and/or legal sanctions in accordance with applicable regulations.

Yogyakarta, 05-December-2025

Raditya Prajnabuwana
21/479720/TK/52923

# PAGE OF DEDICATION

The author of this thesis hereby presents this work to everyone mentioned in the preface and dedicates this work so it may truly be of use to benefit the never-ending development of technology and to support the efforts in realizing a better society for mankind.

Competence-Conscience-Compassion-Commitment-Leadership

Glory to His name, *Ad Maiorem Dei Gloriam.*

# PREFACE

Glory to God to the fullest extent for the guidance and blessings bestowed upon the author throughout the completion of this thesis. This work was carried out with the support and assistance of numerous contributors. Therefore, the author would like to extend his sincere gratitude to:

1. Prof. Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM., SMIEEE., as the Head of Department of Electrical Engineering and Information Technology, Faculty of Engineering, Universitas Gadjah Mada.
2. Dr.-Ing. Yohan Fajar Sidik, S.T., M.Eng. as the first supervisor of the author. The author expresses gratitude for all the insights, guidance, and patience throughout the writing of this work.
3. Ir. Eka Firmansyah, S.T., M.Eng., Ph.D., IPM. as the second supervisor of the author. The author is deeply indebted to him for the invaluable technical advice and constructive revisions given during the writing of this thesis.
4. The beloved parents, younger brother, and family of the author, for the unwavering foundation of support throughout his studies.
5. Steadfast friends of the author who have provided unwavering support.
6. Fellow students of Electrical Engineering, Information Technology, and Biomedical Engineering from batch 2021, for their camaraderie and collaborative spirit.
7. The dearest friends from Arjuna Electric Vehicle UGM team, who contributed to the realization of an innovation-driven and progress-based working environment which later led to wondrous accomplishments during the tenure of the author in the team.
8. The officials and professionals in PLN Enjiniring and Schlumberger Geophysics Nusantara for their guidance, opportunities, and valuable mentorship throughout the internships the author took part in.
9. Dearest friends from the IISMA program to UNSW in 2024, for the invaluable experience and everlasting friendship.
10. Companions at Ethikopia Coffeebay for the memorable companionship during challenging times.

Lastly, it is within the hopes of the author that this thesis may provide benefits and be of use to future research. *Ad Maiorem Dei Gloriam.*

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# NOMENCLATURE AND ABBREVIATION

| | | |
|---|---|---|
| AC | = | Alternating Current |
| ADC | = | Analog-to-Digital Conversion |
| BEV | = | Battery Electric Vehicle |
| BLDC | = | Brushless Direct Current |
| DC | = | Direct Current |
| EMF | = | Electromotive Force |
| EV | = | Electric Vehicle |
| FOC | = | Field-Oriented Control |
| KCL | = | Kirchhoff's Current Law |
| MMF | = | Magnetomotive Force |
| PI | = | Proportional-Integral |
| PMSM | = | Permanent Magnet Synchronous Motor |
| PWM | = | Pulse Width Modulation |
| RMF | = | Rotating Magnetic Field |
| STM32 | = | STMicroelectronics 32-bit microcontroller |
| SVPWM | = | Space Vector Pulse Width Modulation |
| VSI | = | Voltage Source Inverter |

# INTISARI

Motor sinkron magnet permanen (*permanent magnet synchronous motor*-PMSM) memainkan peran krusial dalam dunia modern, seperti pada kendaraan listrik dan otomasi industri. Saat ini, kemampuan mengendalikan PMSM secara akurat dengan tanggapan dinamis yang cepat menjadi kebutuhan. *Field-oriented control* (FOC) merupakan strategi kendali berbasis vektor yang mampu mengendalikan torsi dan fluks PMSM secara independen untuk menghasilkan sistem dengan tanggapan dinamis yang cepat dan akurasi yang baik.

Penelitian ini berfokus pada implementasi FOC menggunakan mikrokontroler STM32 NUCLEO-F446RET6 untuk mengendalikan sebuah *surface-mounted* PMSM (SPMSM). Sistem kendali tertanam menggunakan sepasang pengendali *proportional-integral* (PI) untuk menerapkan pengendalian arus kalang tertutup pada sumbu *direct-quadrature* (d-q). Sebuah tapis lolos rendah (*low-pass filter*-LPF) resistor-kapasitor (*resistor-capacitor*-RC) orde pertama digunakan untuk menekan derau frekuensi tinggi selama pencuplikan arus. *Space vector pulse width modulation* (SVPWM) simetris digunakan untuk mengendalikan penyakelaran sebuah *voltage source inverter* (VSI) tiga fasa yang menggerakkan SPMSM yang dihubungkan dengan beban inersia dan redaman.

Performa sistem kendali dievaluasi melalui simulasi *hardware-in-the-loop* (HIL) waktu nyata menggunakan Plexim RT Box 2. Hasil eksperimen menunjukkan bahwa sistem berhasil meregulasi arus keluaran dengan tanggapan dinamis yang cepat. Penapisan derau frekuensi tinggi menjamin akurasi pencuplikan arus umpan balik, dan penyakelaran menggunakan SVPWM simetris menghasilkan vektor tegangan dengan magnitudo dan sudut yang akurat. Arus keluaran pada sumbu d-q menunjukkan tanggapan cepat terhadap beban dinamis dengan *rise time* lebih cepat dari $30\ \mathrm{ms}$ tanpa *overshoot* dan akurasi lebih dari $95\%$ saat sistem mencapai keadaan tunak.

Kata kunci : Field-Oriented Control, Motor Sinkron Magnet Permanen, Tapis Lolos Rendah, STM32, Hardware-In-The-Loop, Space Vector Pulse Width Modulation.

# ABSTRACT

*Permanent Magnet Synchronous Motors (PMSMs) play a crucial role in the modern world, such as in electric vehicles and industrial automation. Currently, the ability to control PMSMs accurately with a fast dynamic response is required. Field-oriented control (FOC) is a vector-based control strategy capable of independently controlling the torque and flux of PMSMs to create a system with a fast dynamic response and good accuracy*

*This study focuses on implementing FOC using an STM32 NUCLEO-F446RET6 microcontroller to control a surface-mounted PMSM (SPMSM). The embedded control system utilizes a pair of proportional-integral (PI) controllers to perform closed-loop current regulation on the direct-quadrature (d-q) axis. A first-order resistor-capacitor (RC) low-pass filter (LPF) is used to suppress high-frequency noise during current sampling. Symmetrical space vector pulse width modulation (SVPWM) is applied to control a three-phase voltage source inverter (VSI) driving the SPMSM connected to an inertia and damping load.*

*The performance of the control system is evaluated through real-time hardware-in-the-loop (HIL) simulations using a Plexim RT Box 2. The results show that the system successfully controls the output current with a fast dynamic response. High-frequency filtering ensures accurate feedback current sampling, and switching via symmetrical SVPWM produces voltage vectors with accurate magnitudes and angles. The d-q axis current exhibits rapid response to a dynamic load with a rise time faster than* $30\ \mathrm{ms}$ *without overshoot and achieves steady-state accuracy above* $95\%$.

**Keywords** : Field-Oriented Control, Permanent Magnet Synchronous Motor, Low-Pass Filter, STM32, Hardware-In-The-Loop, Space Vector Pulse Width Modulation

# CHAPTER I
# INTRODUCTION

## 1.1 Background

The global energy industry is currently experiencing a profound transformation, driven by the ever-rising urgency of climate change mitigation, decarbonization policy mandates, and the rapid growth of electrification in numerous sectors. Central to this is the electrification of the available modes of transportation, replacing fossil-fuel-based internal combustion engines (ICEs) with electric vehicles (EVs). Power electronic technologies, especially inverters, play a pivotal part in the construction of a more competent renewable-energy-resource-based and environmentally friendly mobility. The performance and reliability of EVs hinge critically on the real-time responsiveness of said technology and its control algorithm.

The government of Indonesia is currently pursuing an ambitious decarbonization agenda while simultaneously pushing the already rapid expansion of its electric mobility ecosystem. As stated in Indonesia's Enhanced Nationally Determined Contribution (NDC) and Long-Term Strategy for Low Carbon and Climate Resilience (LTS-LCCR), the country has pledged to achieve net-zero emissions by 2060 or sooner [4]. The energy sector roadmap prepared by the International Energy Agency, in cooperation with the Ministry of Energy and Mineral Resources of Indonesia, highlights three main key factors in realizing emission reduction: improving energy efficiency, increasing the utilization of renewable energy sources, and the electrification of transportation systems [5]. According to Indonesia's Presidential Regulation No. 55/2019 on the acceleration of battery electric vehicle (BEV) programs, the country aims to deploy over 2 million EVs by 2030, establish domestic battery industries by leveraging its abundant nickel reserves to boost its manufacturing ecosystem, and reduce the national oil consumption and carbon emissions simultaneously [6].

Despite such ambitious targets, there remains a significant gap in the literature and industrial practice regarding low-level embedded control systems for inverter technologies in motor-drive applications. A plethora of real-life implementations heavily rely on vendor-provided libraries, middleware, auto-generated code, or abstraction frameworks. Though such approaches may accelerate development, certain crucial factors, such as deterministic timing, signal acquisition fidelity, synchronization of sampling and switching, and control loop execution, are often left obscured.

Among the vast motor control strategies known, field-oriented control (FOC) has emerged as the most efficient and responsive method for controlling electric motors. It is imperative in the implementation of FOC that the control unit is capable of executing

accurate feedback measurements, reference frame transformations, current control, and vector output via modulation for controlling the inverter driving the electric motor. The quality of measurement results is highly dependent on proper signal filtering to mitigate high-frequency switching noise and ensure valid transformations into the d-q reference frames. The discretization method utilized in constructing a discrete controller influences system stability and dynamic response. Understanding these principles is critical in constructing a vector-based motor controller commanding dynamic torque.

This research proposes the implementation and evaluation of an embedded control system based on the STM32 NUCLEO-F446RE microcontroller to control a three-phase voltage-source inverter (VSI) driving a permanent-magnet synchronous motor (PMSM). The system integrates high-frequency noise filtering for measurement fidelity, a discrete PI controller with back-calculation, and symmetrical space vector pulse-width modulation (SVPWM) to ensure valid voltage vector generation. The control system is validated through hardware-in-the-loop (HIL) simulation using a Plexim RT Box 2 HIL platform, wherein the three-phase VSI, PMSM, and mechanical load are modelled in real time and interfaced with the embedded controller under test.

The topic of this research aligns with Indonesia's dual ambition of achieving industrial independence and environmental sustainability. The implementation of FOC on PMSMs reflects the growth trajectory of the nation's EV industry, in which motor drives, relying on high-performance inverter technologies, are expected to dominate future powertrains. By developing an embedded control unit capable of executing FOC, this research contributes by strengthening domestic expertise in motor-drive control systems, representing an important step in reducing Indonesia's dependence on foreign intellectual property and encouraging local innovation. Furthermore, as the penetration of renewable-energy sources, including geothermal, hydro, and solar, continues to increase in the national electricity grid, the contribution of EVs toward a cleaner and more sustainable transportation ecosystem will correspondingly grow.

In summary, the proposed research not only addresses a critical technical challenge in EVs, but also supports Indonesia's strategic goals of decarbonization, renewable energy utilization, and sustainable mobility. In the context of Indonesia's ambitious dual transition to renewable energy and electric mobility, the results have the potential to accelerate the deployment of a more efficient, reliable, and sustainable inverter-based system, thereby contributing to the national decarbonization goals of Indonesia and the broader global shift towards a reliable and environmentally friendly transportation ecosystem. The real-time, low-level implementation of FOC presented herein is envisioned to become a foundational step towards achieving Indonesia's broader vision of a green, electrified, and resilient transportation future.

## 1.2 Problem Statement

Implementing FOC on a PMSM requires precise coordination of feedback current measurement sampling, reference frame transformations, current control, and pulse-width modulation (PWM). However, achieving this is difficult due to high-frequency noise, constantly varying load during operation, and complex computation requirements for vector output. Switching at high frequency adds noise to the measurement, making it substantially more difficult to obtain accurate feedback current measurements. On the other hand, the tuning of the controllers has a determining effect on the dynamic stability of the closed-loop system. Additionally, generating the desired output vector requires proper computation and control to ensure valid switching.

Simulations that are strictly software-based often do not involve the temporal effects and peripheral interactions inherent to microcontroller implementation. Therefore, conducting performance evaluation using HIL-configured real-time testing is a necessity for evaluating the behavior of the embedded control system in the context of realistic dynamic conditions. Accordingly, the main issue this study will solve is the algorithmic design and HIL-configured performance evaluation of a three-phase VSI closed-loop control system based on an STM32 microcontroller using FOC to drive a PMSM, with focus on high-frequency noise filtering for feedback accuracy, responsive output regulation, and valid vector output in terms of magnitude and phase.

## 1.3 Objectives

The main objective of this research is to implement FOC to a PMSM by utilizing an STM32 microcontroller as the embedded control platform. In order to accomplish this, proper design and planning are required to perform the numerous processes involved in the system. The system incorporates high-frequency noise filtering for accurate analog-to-digital conversion-based (ADC-based) feedback sampling, reference frame transformations, current regulation via proportional-integral (PI) and feedforward control, and PWM signal generation via SVPWM for vector output. The program architecture of the constructed control system is stylized to operate with minimum assistance from external libraries. The FOC algorithm is to be implemented on the STM32 platform and tested via HIL simulation, allowing for a better evaluation of the algorithm in terms of its operational characteristics and dynamic behavior in conditions approximate to realistic operational conditions.

## 1.4 Scope and Limitations

This research mainly focuses on the design and implementation of an FOC algorithm on a PMSM using the STM32 NUCLEO-F446RET6 microcontroller. The prime objective is to construct and validate the embedded control system, incorporating ana-

log measurement sampling via injected ADC, vector reference frame transformations, current regulation via feedback and feedforward control, and SVPWM. The source code of the program is written in the C programming language using STM32CubeIDE with STM32CubeMX to perform peripheral initialization and configuration.

However, this work does not cover analysis of analog or digital filter types, discrete controller discretization techniques, advanced motor parameter identification methods, or analysis of signal modulation methods. The implementation of the realized control system is performed entirely on the STM32 platform and tested via HIL simulation using a Plexim RT Box 2, enabling real-time performance evaluation without requiring any physical components to form the plant. The control system constructed in this work is mainly for current control without involving any other motor control strategies, such as speed control, field-weakening, etc. Furthermore, the implementation is limited to a single-motor, single-controller system. Aspects such as multi-motor control, communication protocols, or thermal management are beyond the scope of this work.

## 1.5 Benefits

This research contributes to the advancement of motor control systems via implementation of FOC on a PMSM using an STM32 NUCLEO-F446RET6 microcontroller. The development of a fully-functional FOC algorithm serves as a testament to the vast capabilities of the STM32 platform, enabling the realization of a precise and responsive embedded control system. On the other hand, evaluation of the constructed system via HIL simulation with a Plexim RT Box 2 showcases an alternative hardware-based method of real-time simulation, providing a safer and more practical system validation mechanism that reduces the risk while simultaneously increasing cost-effectiveness.

The results provided by this work serve as a reference for upcoming studies discussing similar topics. Future researchers and engineers may use any and all results and knowledge provided by this study in developing real-time control strategies for EVs, renewable energy applications, and industrial automation systems in Indonesia. By promoting local innovation in embedded motor control technologies, this research aligns with the ambitions of Indonesia's national agenda toward energy transition, decarbonization, and the realization of EV-friendly infrastructures.

## 1.6 Outline

This thesis is organized as follows: Chapter I presents the background, problem statement, objectives, scope and limitations, and lastly, the benefits of this research. Chapter II reviews the fundamental theories related to the topic of this research, such as the operation of three-phase VSI, PMSM, FOC, controller, and SVPWM. Chapter III describes the research methodology, including the system design, hardware programming,

algorithm implementation, and test bench setup. Chapter IV discusses the results and analysis, covering the system component testing, performance evaluation, and real-time response analysis of the system during transient state and steady state. Chapter V concludes this research by providing conclusions and offering recommendations for future work related to embedded motor control systems.

# CHAPTER II
# LITERATURE REVIEWS AND THEORETICAL FOUNDATIONS

## 2.1 Literature Reviews

Control systems for PMSM-drive applications remain a growing research subject, largely due to the increasing demand for high-performing solutions in renewable energy systems and EVs. The development of motor control-friendly embedded platforms, such as STM32, has enabled real-time implementation of control strategies, utilizing cost-efficient and low-power microcontrollers. This chapter reviews related research and related theoretical foundations, covering studies from Universitas Gadjah Mada and recent works from international journals and books.

### 2.1.1 Permanent Magnet Synchronous Motor (PMSM) Modelling

Proper modelling of PMSMs plays a crucial role in both designing and optimizing the performance of the motor control system. It is explained in [7] that PMSM models often utilize the d-q axis for simplification of the nonlinear three-phase dynamics of the system into a more manageable two-axis reference frame. Doing so enables precise calculation of both the torque and flux interactions crucial for most control strategies like FOC and direct torque control (DTC). This review underlines the lack of comprehensive validation in real-time embedded environments, a limiting factor when it comes to representing practical applications using theoretical models.

Furthermore, the mathematical modelling of PMSM drives integrated with an inverter is explored in [8]. This study demonstrated the importance of model fidelity in ensuring realistic simulation outputs and reliable design, especially in inverter switching and electromagnetic dynamics. High-precision modelling remains a crucial prerequisite in implementing advanced control algorithms like FOC, proven by the illustrated dependence between the electrical and mechanical domains of the PMSM.

### 2.1.2 Field-Oriented Control (FOC)

When implemented with SVPWM, FOC enables optimal utilization of the direct current (DC) bus voltage while simultaneously minimizing output harmonic distortion. In [9], it is explained that using FOC as the motor control strategy improves both the transient- and steady-state performance of synchronous machines via independent control of current components responsible for generating both torque and flux. The decoupling capability of FOC results in smoother torque output and better dynamic response, making FOC a favored control strategy for modern motor drive systems, particularly in EVs and industrial automation systems.

This perspective is complemented in [10] by demonstrating that the decoupling capabilities of FOC allow a rapid torque response and reduced torque ripple compared to conventional voltage per frequency (V/f) scalar control methods. FOC is able to deliver performance comparable to that of separately excited DC machines via access to linear control of the torque output, an admirable feat emphasized even more by its ability to simultaneously preserve the robustness and efficiency of the AC system. These findings highlight the critical role of FOC in achieving responsiveness in high-performing PMSM drives. The findings within these works also underline how accurate current decoupling and precise rotor position sensing highly influence the performance of FOC.

### 2.1.3 Embedded Inverter Control Implementation

Research and practical implementations have demonstrated the significance of hardware synchronization in achieving stable and responsive performance when utilizing an inverter, this is especially true in terms of current sampling, PWM switching, and control loop execution. The capability of an STM32 microcontroller in handling said control tasks is showcased in [11] through the design of a dual-quadrant armature controller for a separately excited DC motor. This work highlights the reason why synchronization between PWM and ADC is a key component in the design of current-regulated systems like FOC.

### 2.1.4 Embedded Motor Control Implementation

In [12], a BLDC motor controller was developed using the FOC control strategy implemented on an Infineon XMC4500 microcontroller for Arjuna EV Universitas Gadjah Mada (UGM) formula student team. This work serves as an example of an effective embedded motor control system implementation due to its notable improvements, such as minimum torque ripple and fast dynamic response, while simultaneously using a cost-efficient microcontroller. Additionally, an embedded PMSM controller capable of performing coordinate transformations, PWM generation, and torque regulation required in a motor controller was developed in [13]. This research emphasized the integration of said control processes into a coherent embedded framework by using an STM32 NUCLEO-F446RET6 microcontroller.

### 2.1.5 Hardware-In-The-Loop (HIL) Simulation

The option of conducting a HIL simulation instead of a real-time simulation involving physical hardware enables an embedded controller to interact with a computational model representing the system plant. In the case of this research, a Plexim RT Box 2 computes the real-time plant dynamics of the system, allowing the controller to operate as though it were controlling the modelled plant, which consists of a three-phase

inverter and PMSM. In [14], a demo model titled "Sensorless FOC of PMSM on STM32 MCU" was made, serving as a testament to the capabilities of the RT Box 2 in simulating approximate real-time operational conditions while simultaneously highlighting its role as a solid method of embedded control system validation via HIL simulation.

The capabilities of HIL simulation in bridging the gap between software-based simulation and hardware-based prototyping has been demonstrated by numerous works. For example, see [15] and [16]. The HIL simulations done in these works have proven to be effective in simulating inverter control, modulation, and feedback mechanisms under realistic operating conditions. The studies conducted have also proven the well-deserved recognition HIL simulation has been getting due to its undeniable contribution as a step in accelerating the development of power electronic systems.

Table 2.1. Summary of Reviewed Literature.

| **Authors, Year** | **Topic** | **Relevance to Current Work** |
|---|---|---|
| Khan (2024) [7] | Modelling and simulation of PMSM drives with d-q axis representation. | d-q axis transformation in PMSM dynamics simplification. |
| Amin et al. (2016) [8] | Mathematical modelling of PMSM drive system with inverter integration. | Highlighted the importance of accurate modelling of inverter switching and coupling term. |
| Krishnan (2017) [9] | Modelling, analysis, and control of electric motor drives. | Confirmed that the decoupling capability of FOC improves performance of PMSM control during transient- and steady-state conditions. |
| Singh and Kaur (2013) [10] | Comparison of FOC and scalar control for PMSMs. | Demonstrated and compared FOC and V/f scalar control in terms of torque ripple reduction and dynamic response. |
| Shiddqi (2024) [11] | Dual-quadrant armature controller for a separately-excited DC motor using STM32. | Demonstrated the PWM and ADC capabilities of an STM32 microcontroller for the implementation of a current regulation system. |
| Satria (2019) [12] | High-performance BLDC motor controller using FOC for Arjuna EV UGM. | Successful embedded implementation of FOC for high-performance BLDC drives. |
| Handoko (2022) [13] | PMSM embedded control system using STM32 NUCLEO-F446RET6. | Integration of reference frame transformations, PWM generation, and current regulation in an embedded FOC implementation. |
| Plexim (2023) [14] | Sensorless FOC of PMSM on STM32 MCU (RT Box Demo). | HIL simulation as an alternative method for real-time testing. |
| Pecas Lopes et al. (2021) [15] | HIL validation for power electronic systems. | Demonstrated the ability of HIL simulation in interfacing software simulation and hardware prototyping. |
| Chen et al. (2022) [16] | HIL-based inverter control verification. | Demonstrated the effectiveness of HIL simulation in simulating realistic conditions for inverter control system testing. |

## 2.2 Theoretical Foundations

### 2.2.1 Permanent Magnet Synchronous Motor (PMSM)

The PMSM is an alternating current (AC) electric motor which utilizes permanent magnets for rotor excitation, granting the benefit of eliminating copper losses. The stator of a PMSM is comprised of armature windings used to generate the rotating magnetic field (RMF) for commutation via electromagnetic induction. The rotor of a PMSM exploits the magnetic field of the embedded permanent magnets to rotate by directly interacting with the RMF generated by the stator [1].

#### 2.2.1.1 Types

There is a plethora of PMSM variants, each categorized according to numerous criteria. Based on the topology of its stator winding, PMSMs may adopt either the distributed or concentrated winding configuration. The distributed topology allows PMSMs to generate a more sinusoidal magnetomotive force (MMF) in the air gap, reducing the harmonic content [8]. The concentrated winding topology, on the other hand, offers a higher slot fill factor along with a number of advantages such as higher power density, more cost-effective manufacturing, and shorter end windings [10].

Based on the direction of the magnetic flux, PMSMs can be classified as either radial-flux or axial-flux type. The rotor of a radial-flux PMSM is enclosed within the stator, causing the magnetic flux to cross the air gap in a radial direction. On the other hand, an axial-flux PMSM has its rotor mounted adjacent to its stator in a "pancake" arrangement. Parallel placement of its rotor and stator causes a parallel path for the magnetic flux when crossing the air gap [1].

Lastly, the placement of the permanent magnets in a PMSM determines whether it is classified as a surface-mounted PMSM (SPMSM) or an interior PMSM (IPMSM). The permanent magnets of an SPMSM are mounted on the outer surface of the rotor, allowing for a simple structure and uniform reluctance. SPMSMs generally have a nearly equal d-q axis inductance ($L_d = L_q$), giving the SPMSM a linear current-torque relationship allowing for a more straightforward control [13]. The IPMSM, on the other hand, embeds its permanent magnets inside the rotor core rather than on its outer surface. This method of installation enables magnetic- and reluctance-based torque generation ($L_d < L_q$, causing saliency), allowing for a higher torque density and improved flux weakening for high-speed operations [9]. However, each PMSM type has its own set of trade-offs. SPMSMs exhibit mechanical retention at high speed and demagnetization when under adverse conditions (e.g., high temperature or armature reaction), while IPMSMs require a more complex modelling and control due to their saliency and electromagnetic properties [1].

#### 2.2.1.2 Equivalent Circuit and Equations

The voltage equation of the PMSM can be expressed in matrix form with $V_{ns}$ as its phase-to-neutral voltages, $i_{ns}$ as its phase currents, $R_s$ as its winding resistance, and $\lambda_{ns}$ as its total flux linkage, while $n = a, b, c$.

$$\begin{bmatrix} V_{as} \\ V_{bs} \\ V_{cs} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \lambda_{as} \\ \lambda_{bs} \\ \lambda_{cs} \end{bmatrix}$$

The flux of a PMSM consists of the flux caused by the currents in the stator and the flux caused by the permanent magnet of the rotor [1]. They can be expressed as the product of the inductances and currents.

$$\begin{bmatrix} \lambda_{as} \\ \lambda_{bs} \\ \lambda_{cs} \end{bmatrix} = \begin{bmatrix} L_{asas} & L_{asbs} & L_{ascs} \\ L_{bsas} & L_{bsbs} & L_{bscs} \\ L_{csas} & L_{csbs} & L_{cscs} \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + I_f \begin{bmatrix} L_{asf} \\ L_{bsf} \\ L_{csf} \end{bmatrix}$$

The flux caused by the permanent magnet of the rotor ($\psi_{asf}$, $\psi_{bsf}$, and $\psi_{csf}$) can be expressed as the product of the equivalent field current $I_f$ and the phase inductances $L_{asf}$, $L_{bsf}$, and $L_{csf}$.

Both the voltage and flux equations of the PMSM can be transformed to set them on a more manageable two-axis coordinate system perpendicular to each other, known as the d-q axis. The d-axis is generally known as the reference axis, with the flux-generating component of the currents being aligned with this axis. Meanwhile, the q-axis, perpendicular to the d-axis, is the axis with which both the torque-generating component of the currents and the back electromotive force (EMF) of the PMSM are aligned [7].

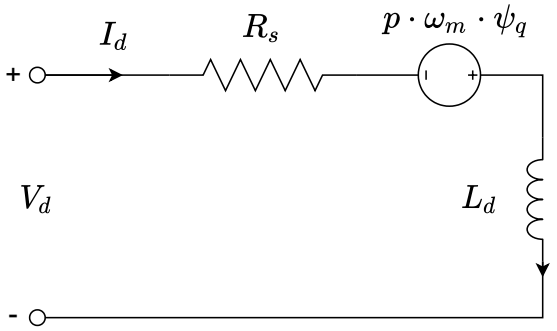


Figure 2.1. PMSM d-Axis Equivalent Circuit.

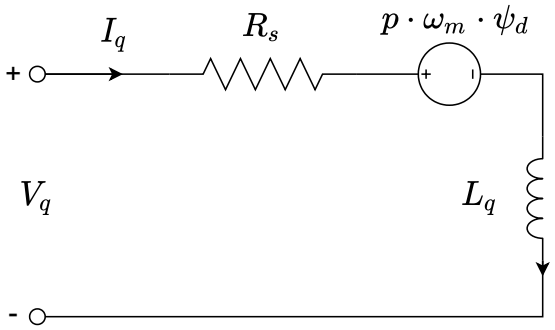


Figure 2.2. PMSM q-Axis Equivalent Circuit.

When operating on the d-q axis, the voltage equation of the PMSM can be expressed in terms of the d-q currents and fluxes, as shown in Equations 2-1 and 2-2.

$$v_{ds} = R_s i_{ds} + \frac{d\lambda_{ds}}{dt} - \omega_e \lambda_{qs} \tag{2-1}$$

$$v_{qs} = R_s i_{qs} + \frac{d\lambda_{qs}}{dt} + \omega_e \lambda_{ds} \tag{2-2}$$

On the other hand, the fluxes of the PMSM when transformed to the d-q axis can also be expressed in terms of the d-q currents and inductances of the PMSM, as shown in Equations 2-3 and 2-4.

$$\lambda_{ds} = L_{ds} i_{ds} + \psi_f \tag{2-3}$$

$$\lambda_{qs} = L_{qs} i_{qs} \tag{2-4}$$

Using Equations 2-3 and 2-4, the d-q axis voltage equations in Equations 2-1 and 2-2 can then be expressed strictly in terms of the d-q axis currents, as shown in Equations 2-5 and 2-6. These equations can be used to determine the d-q axis voltage during transient state, while Equations 2-7 and 2-8, derived from the previous equations, can be used to determine the voltages when the system has reached steady state.

$$v_{ds} = R_s i_{ds} + L_{ds} \frac{di_{ds}}{dt} - \omega_e L_{qs} i_{qs} \tag{2-5}$$

$$v_{qs} = R_s i_{qs} + L_{qs} \frac{di_{qs}}{dt} + \omega_e (L_{ds} i_{ds} + \psi_f) \tag{2-6}$$

$$v_{ds_{ss}} = R_s i_{ds} - \omega_e L_{qs} i_{qs} \tag{2-7}$$

$$v_{qs_{ss}} = R_s i_{qs} + \omega_e (L_{ds} i_{ds} + \psi_f) \tag{2-8}$$

The magnitude of the q-axis current supplied into the PMSM directly dictates the magnitude of the torque $T_e$ generated by the PMSM. The torque of a PMSM is expressed in Equation 2-9, where the saliency of the PMSM contributes in generating torque via reluc-

tance. However, the SPMSM has an approximately similar d-q axis inductance, therefore the torque equation for the SPMSM as expressed in Equation 2-10.

$$T_e = \frac{3}{2}\frac{P}{2}(\psi_f i_{qs} + (L_{ds} - L_{qs}) i_{ds} i_{qs}) \tag{2-9}$$

$$T_e = \frac{3}{2}\frac{P}{2}\psi_f i_{qs} \tag{2-10}$$

The mechanical rotational speed of the PMSM rotor $\omega_m$ depends on the generated torque $T_e$ and the mechanical load driven by the PMSM. However, the electrical speed $\omega_e$ of the rotor is different to its mechanical speed. The electrical rotational speed of the PMSM rotor depends on the mechanical speed and the number of poles within the stator of the PMSM, as expressed in Equation 2-11 [9]. With the electrical speed known, the electrical frequency corresponding also to the frequency of the voltage and current supplied to the PMSM can be calculated using Equation 2-12.

$$\omega_e = \omega_m \cdot \frac{P}{2} \tag{2-11}$$

$$f_e = \frac{\omega_e}{2\pi} \tag{2-12}$$

#### 2.2.1.3 Transfer Function and Mathematical Model

A transfer function represents the relationship between the input and output of a linear time-invariant (LTI) system [11]. The previously explained d-q axis voltage of the PMSM represents the relationship between the currents flowing through the stator and the voltage of the stator. Therefore, the transfer function representing the voltage-current properties of the PMSM can be gained via Laplace transformation of the d-q axis voltage equation, as shown in Equations 2-13 and 2-14.

$$V_d(s) = I_d(s)(R_s + sL_{ds}) - \omega_e L_{qs} I_q(s) \tag{2-13}$$

$$V_q(s) = I_q(s)(R_s + sL_{qs}) + \omega_e L_{ds} I_d(s) + \omega_e \psi_f \tag{2-14}$$

$$\begin{bmatrix} V_d(s) \\ V_q(s) \end{bmatrix} = \begin{bmatrix} R_s + sL_d & -\omega_e L_q \\ \omega_e L_d & R_s + sL_q \end{bmatrix} \begin{bmatrix} I_d(s) \\ I_q(s) \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \psi_f \end{bmatrix}$$

The output of the Laplace transformation is a pair of s-domain equations describing the input-output relationship of the PMSM. The transfer function describes the PMSM as a multiple-input multiple-output (MIMO) system due to the voltage at each axis being dependent on the current of both axes [8]. The equations can be expressed in matrix form, where it can be seen that the diagonal terms are identical pairs representing a resistor-inductor (R-L) circuit and the coupling terms that cause the system to be a MIMO sys-

tem ($\omega_e L_d$ and $\omega_e L_q$) [1]. The q-axis has an additional flux component, this is the back EMF of the PMSM whose magnitude is dependent on the rotational speed of the rotor ($\omega_e \psi_f$) [1].

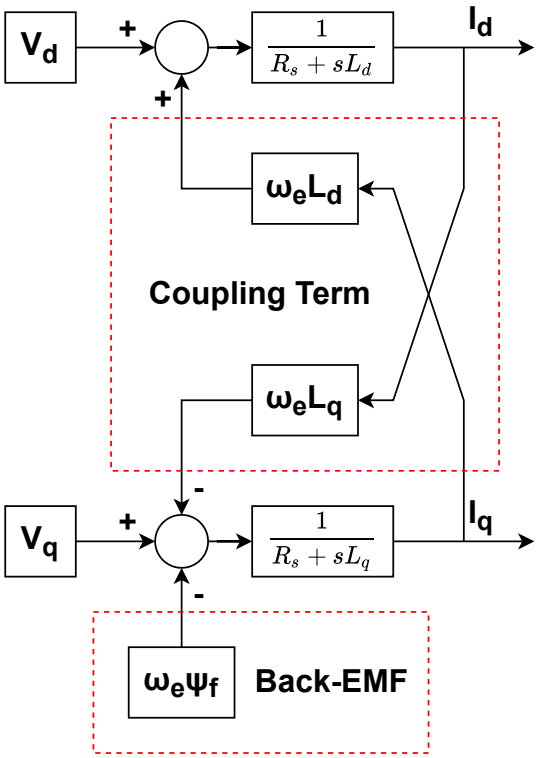


Figure 2.3. PMSM d-q Axis Mathematical Model.

The system can be simplified into a single-input single-output (SISO) system via decoupling. Doing this will result in a set of transfer functions describing the input-output relationship of the d-q axis PMSM voltage and current with the voltage as the input and the current as the output [1]. The d- and q-axes both have identical transfer functions represented in Equation 2-15. However, the decoupling in this process means the control system should be designed to compensate the coupling term, a topic to be discussed further in upcoming subchapters.

$$I_{dq}(s)/V_{dq}(s) = 1/(R_s + sL_{dqs}) \tag{2-15}$$

### 2.2.2 Three-Phase Voltage Source Inverter (VSI)

The three-phase inverter is a power electronic technology used to generate three-phase AC power using a DC power source, serving as the main actuator interfacing a DC power source and an AC motor in motor drives. The most widely adopted variation is the VSI, consisting of two semiconductor switches (commonly IGBTs or MOSFETs) in

each phase leg connected in a half-bridge arrangement [12]. The VSI is often utilized along with a DC voltage source and a capacitor to smooth voltage ripples [1].

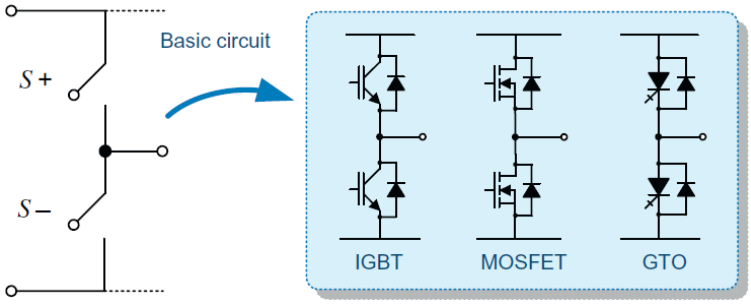


Figure 2.4. Half-Bridge Phase Leg of an Inverter [1].

A switching function $S$ is used to define the output of each phase leg, with $S = 0$ and $S = 1$ indicating activation of the lower switch and upper switch in that order [1]. Simultaneous activation of the upper and lower switch in one phase leg leads to a destructive shoot-through condition, making the insertion of a dead time delay between the switching transitions a necessity in implementations [2]. When used correctly, the inverter generates AC voltage via rapid switching, causing the output terminals to alternate between the positive and negative rails of the DC voltage.

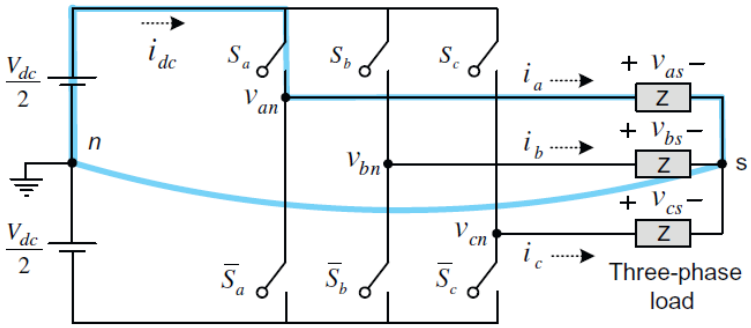


Figure 2.5. VSI Output Connection During Switching [1].

The output connects to $\frac{V_{DC}}{2}$ when $S = 1$ and $-\frac{V_{DC}}{2}$ when $S = 0$ [1]. Operation of all three legs with a 120° phase displacement enables the inverter to produce a balanced three-phase AC output, an important property to exploit in motor drives. When used to drive an AC motor, inverters are operated using PWM techniques to generate sinusoidal voltage waveforms rather than square waves, leading to smoother torque and higher efficiency.

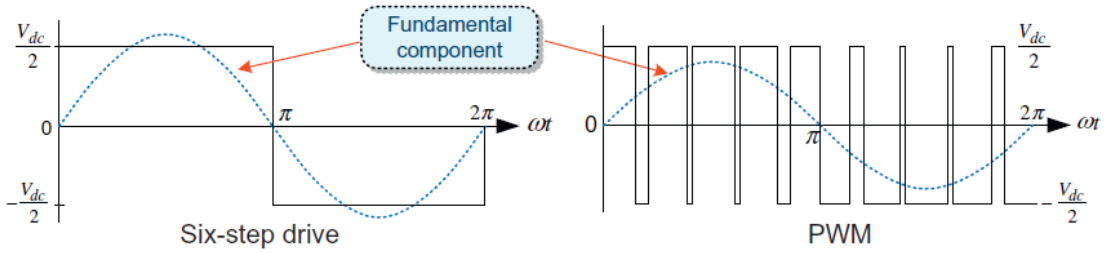


Figure 2.6. Sinusoidal Waveform Output [1].

### 2.2.3 Field-Oriented Control (FOC)

FOC is a vector-based control strategy with the main purpose of achieving independent torque and flux regulation in AC machines, replicating the performance characteristics of DC machines [13]. The three-phase current supplied to the AC machine is transformed into the d-q frame using Clarke transformation and Park transformation, allowing the control system to perform decoupled control of both the torque and flux. The magnitude of the d-axis current determines the rotor flux, while the q-axis current governs torque production.

The capabilities of FOC as a motor control strategy in minimizing torque ripple while enhancing transient response are emphasized in [17] and [2]. Additionally, its implementation in embedded systems is demonstrated in [12] and [13], which resulted in the previously mentioned smooth torque output and efficient operation. In realizing the FOC, it is imperative to understand its key processes, such as the current feedback, reference frame transformations, and closed-loop controllers.

#### 2.2.3.1 Feedback Control and Reference Frame Transformations

1. **Current Feedback**

   As a closed-loop control strategy, FOC requires continuous monitoring of the system output, in this case it is the PMSM stator currents. Current feedback is therefore crucial in realizing a properly functional closed-loop FOC due to its role in providing real-time measurements of the phase currents $i_a$, $i_b$, and $i_c$ [2]. Current sensors are utilized to accomplish this in real-life implementation scenarios involving physical PMSMs.

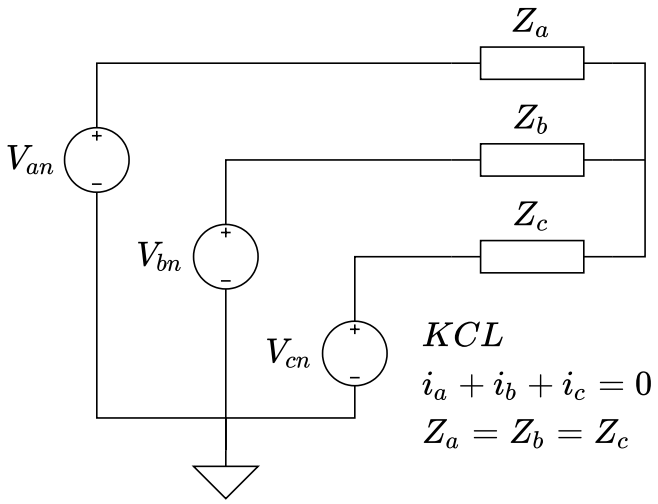


Figure 2.7. KCL of Balanced Wye-Connected Three-Phase Circuit.

However, sensing all three phase currents is not required, granted if the configuration of the three-phase windings of the PMSM is known. Kirchhoff's Current Law (KCL) constitutes that the sum of three-phase currents in a balanced system equals zero, therefore sensing only two phase currents is sufficient as long as the configuration of the stator winding is known [2]. The phase currents sensed to provide current feedback are commonly $i_a$ and $i_b$, $i_c$ is commonly calculated to reduce the required sensor count while simultaneously maintaining complete observability of the system. For a wye-connected stator winding, the currents behave as is expressed in Equation 2-16.

$$i_a + i_b + i_c = 0 \tag{2-16}$$

2. **Clarke Transformation**

The three-phase current obtained via current feedback still lies in the three-axis coordinate system. The Clarke transformation (also known as the $\alpha - \beta$ transformation) enables simplification of three-phase AC machine analysis via vector decomposition of three-phase signals into a two-dimensional vector in a stationary reference frame with orthogonal components fixed in space [2]. In this case, the three-phase stator currents $i_a$, $i_b$, and $i_c$ are transformed into $i_\alpha$ and $i_\beta$, representing the stator current vector onto a pair of stationary axes perpendicular to each other.

The Clarke transformation can be expressed in matrix form as such.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}$$

Since the current feedback only grants the values of $i_a$ and $i_b$ to the closed-loop control system, the transformation can be adjusted using the previously explained KCL. For PMSMs with a wye-connected stator winding, the transformation is as expressed in Equations 2-17 and 2-18.

$$i_\alpha = i_a \tag{2-17}$$

$$i_\beta = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \tag{2-18}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix}$$

The utilization of this transformation as a part of the FOC scheme allows the elimination of redundancy while preserving the total power of the system [2]. In polar form, $i_\alpha$ and $i_\beta$ can be referred to as an $\alpha - \beta$ axis current vector with its magnitude represented by $r_{\alpha\beta}$ and its angle by $\theta_{\alpha\beta}$. Equations 2-19 and 2-20 show the conversion from the cartesian coordinate form to the polar coordinate form, while Equations 2-21 and 2-22 shows the opposite.

$$r_{\alpha\beta} = \sqrt{i_\alpha^2 + i_\beta^2} \tag{2-19}$$

$$\theta_{\alpha\beta} = atan2(i_\beta, i_\alpha) \tag{2-20}$$

$$i_\alpha = r_{\alpha\beta} \cos(\theta_{\alpha\beta}) \tag{2-21}$$

$$i_\beta = r_{\alpha\beta} \sin(\theta_{\alpha\beta}) \tag{2-22}$$

3. **Park Transformation**

The Park transformation transforms a two-dimensional vector from the stationary reference frame into the rotating reference frame, represented by the d-q axis. Performing this not only rotates the previously stationary reference frame, it decouples the stator current vector into two orthogonal components known as the direct-axis (d-axis) current controlling the flux and the quadrature-axis (q-axis) current controlling the electromagnetic torque [17]. When applied to the output of the Clarke transformation, this transformation converts the stationary quantities of $i_\alpha$ and $i_\beta$ into $i_d$ and $i_q$, time-invariant values aligned with the magnetic field of the rotor.

The transformation is expressed in Equations 2-23 and 2-24, with $\theta_e$ as the electric angle of the rotor.

$$i_d = i_\alpha \cos\theta_e + i_\beta \sin\theta_e \tag{2-23}$$

$$i_q = -i_\alpha \sin\theta_e + i_\beta \cos\theta_e \tag{2-24}$$

In matrix form, it can be seen that the transformation matrix is similar to a clockwise rotation matrix with $\theta_e$ as the rotation angle.

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}$$

The d-q axis currents can be viewed as a current vector in the d-q axis, with its magnitude represented by $r_{dq}$ and its angle by $\theta_{dq}$. Equations 2-25 and 2-26 show the conversion from the cartesian coordinate form to the polar coordinate form, while Equations 2-27 and 2-28 shows the opposite.

$$r_{dq} = \sqrt{i_d^2 + i_q^2} \tag{2-25}$$

$$\theta_{dq} = atan2(i_q, i_d) \tag{2-26}$$

$$i_d = r_{dq}\cos(\theta_{dq}) \tag{2-27}$$

$$i_q = r_{dq}\sin(\theta_{dq}) \tag{2-28}$$

When Equations 2-21 and 2-22 are used to expand Equations 2-23 and 2-24, it can be proven that the Park transformation does not change the magnitude of the vector from the $\alpha - \beta$ axis, rather it only shifts the angle of the vector. This proves the rotational characteristic of the Park transform, as previously mentioned when observing its matrix form. Equations 2-29 and 2-30 use trigonometric identities to demonstrate the constant magnitude and phase shift between the $\alpha - \beta$ axis and d-q axis vectors, as expressed in Equations 2-31 and 2-32.

$$\begin{aligned} i_d &= r_{\alpha\beta}(\cos(\theta_{\alpha\beta})\cos(\theta_e) + \sin(\theta_{\alpha\beta})\sin(\theta_e)) \\ &= \frac{r_{\alpha\beta}}{2}(\cos(\theta_{\alpha\beta} - \theta_e) + \cos(\theta_{\alpha\beta} + \theta_e) + \cos(\theta_{\alpha\beta} - \theta_e) - \cos(\theta_{\alpha\beta} + \theta_e)) \\ &= r_{\alpha\beta}\cos(\theta_{\alpha\beta} - \theta_e) \end{aligned} \tag{2-29}$$

$$\begin{aligned} i_q &= r_{\alpha\beta}(-\cos(\theta_{\alpha\beta})\sin(\theta_e) + \sin(\theta_{\alpha\beta})\cos(\theta_e)) \\ &= \frac{r_{\alpha\beta}}{2}(-\sin(\theta_{\alpha\beta} + \theta_e) + \sin(\theta_{\alpha\beta} - \theta_e) + \sin(\theta_{\alpha\beta} + \theta_e) + \sin(\theta_{\alpha\beta} - \theta_e)) \\ &= r_{\alpha\beta}\sin(\theta_{\alpha\beta} - \theta_e) \end{aligned} \tag{2-30}$$

$$r_{dq} = r_{\alpha\beta} \tag{2-31}$$

$$\theta_{dq} = \theta_{\alpha\beta} - \theta_e \tag{2-32}$$

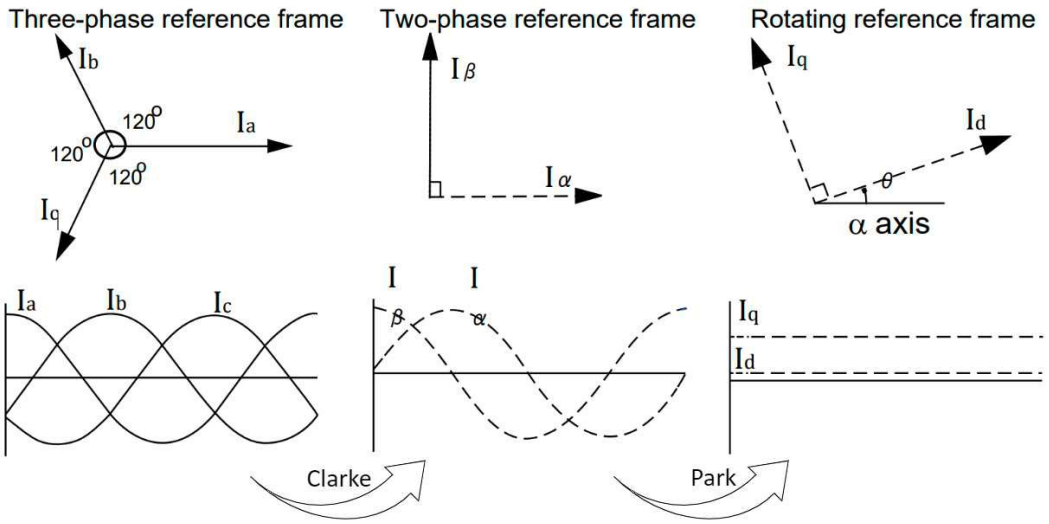


Figure 2.8. Reference Frame Transformations [2].

#### 2.2.3.2 Controller

To implement FOC to a PMSM, it is important to acknowledge the inherent non-linearities in the system. Such deviations, when dealt with without the assistance of a control mechanism, are sure to cause undesired disturbances in the form of oscillating torque output, unstable speed response, and the overall reduction of efficiency. For this reason, a controller is implemented to ensure minimum error between the set reference and measured currents, thereby ensuring correct voltage vector outputs for a stable FOC. Among the vast options, the proportional-integral (PI) controller stands out as the most widely adopted controller in FOC implementations due to its effectiveness.

1. **PI Controller**

   A proportional controller (P controller) produces an output directly proportional to the instantaneous error between the reference and the measured values, while an integral controller (I controller) adjusts its output based on the accumulation of error over time [1]. When used individually, the output of the P controller can be zero when the error reaches a constant value, rendering it unable to eliminate steady-state error when the error is constant at a nonzero value. On the other hand, the I controller has an overall slower response and a risk of introducing potential overshoot during transient conditions.

   Meanwhile, the PI controller utilizes its proportional term to provide fast dynamic response when faced with transient conditions while simultaneously ensuring the

elimination of steady-state error via its integral term, thereby providing the best of both worlds in one controller [18]. The performance of both terms is determined by the set values of each of their gains, known as the proportional gain $K_p$ and the integral gain $K_i$. The time-domain equation of a continuous PI controller is expressed in Equation 2-33.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau \tag{2-33}$$

The transfer function of the PI controller can be acquired via Laplace transformation, the result of which is expressed in Equation 2-34. The mathematical model of the PI controller is illustrated in Figure 2.9.

$$U(s)/E(s) = K_p + \frac{K_i}{s} \tag{2-34}$$

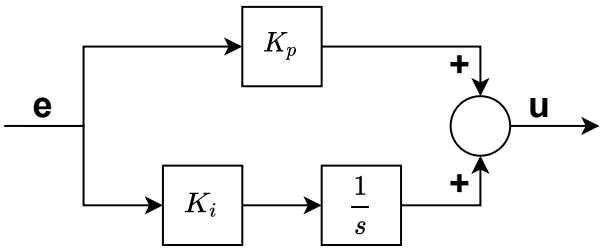


Figure 2.9. Mathematical Model of a Continuous PI Controller.

A risk this controller poses is a phenomenon known as an integral windup, where the integral term continues to accumulate error during an actuator saturation scenario. Such phenomenon may cause delayed recovery and a visible overshoot due to the stored integral value being released when the saturation clears, leading to possible instability or oscillation [19]. To prevent this, a PI controller is often implemented along with an anti-windup mechanism, one of which is the back-calculation method. Back-calculation allows the difference between the saturated and unsaturated outputs to be fed back through a feedback gain $K_b$ to the integral term, allowing for a faster saturation recovery and an enhanced transient performance. The mathematical model of a PI controller with back-calculation is illustrated in Figure 2.10.

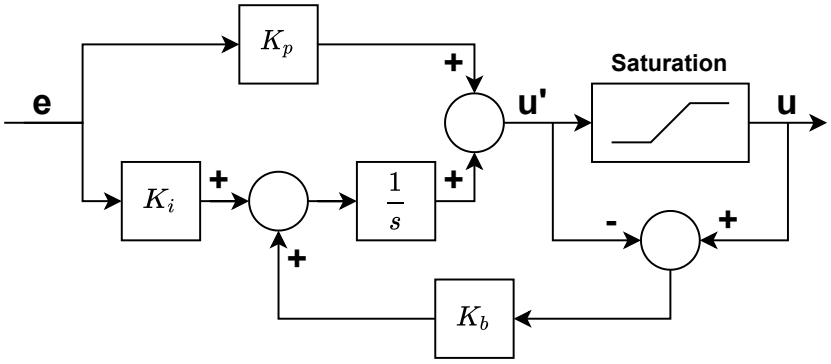


Figure 2.10. Continuous PI Controller with Back-Calculation.

2. **Pole-Zero Cancellation**

FOC controls AC machines via regulation of their three-phase current, therefore its structure consists of a pair of controllers posing as current regulators. During implementation, the aforementioned regulator operates within a certain bandwidth, denoted as $\omega_{cc}$, a critical design parameter determining the rapidness of the controller response [1]. A wider bandwidth translates to faster response and improved dynamic performance, but causes the system to be more prone to noise-related disturbances and have limited gain margins due to the constraints posed by the inverter switching and sampling frequencies [1].

In this FOC scheme, the PI controllers act as the current regulators. The gains of both terms of the controllers are determined via the pole-zero cancellation method, the purpose of which is to utilize the zero of the controller to cancel the pole of the plant [19]. Consider the decoupled open-loop system of the PMSM where the transfer function of the plant is represented by Equation 2-15 and the PI Controller by Equation 2-34, resulting in Equation 2-35.

$$G_o(s) = (K_p + \frac{K_i}{s})(\frac{1}{R_s + sL_{dq}}) = K_p\frac{(s + \frac{K_i}{K_p})}{s} \cdot \frac{\frac{1}{L_{dq}}}{(s + \frac{R_s}{L_{dq}})} \tag{2-35}$$

It can be seen that the PI Controller provides a zero at ($s = -\frac{K_i}{K_p}$), while the plant has a pole at ($s = -\frac{R_s}{L_{dq}}$). Implementing pole-zero cancellation by applying Equation 2-36, the transfer function of the open-loop system will be as is expressed in Equation 2-37.

$$\frac{K_i}{K_p} = \frac{R_s}{L_{dq}} \tag{2-36}$$

$$G_o(s) = \frac{1}{s(\frac{L_{dq}}{K_p})} \tag{2-37}$$

When a feedback mechanism with unity gain is added to this system as shown in Figure 2.11, the transfer function of the now closed-loop system is expressed in Equation 2-38 with $\omega_{cc}$ at $\omega_{cc} = \frac{K_p}{L_{dq}}$.

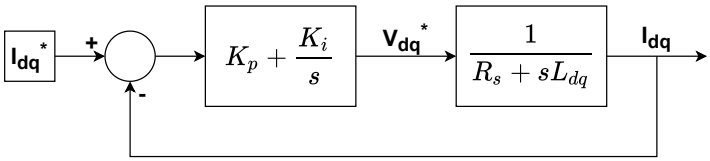


Figure 2.11. Closed-Loop PMSM Plant with PI Controller.

$$G_c(s) = \frac{\omega_{cc}}{s + \omega_{cc}} \tag{2-38}$$

This transfer function describes the closed-loop system to be a stable first-order system with a cut-off frequency $\omega_{cc}$, a rise time four times the time constant of the system, and no overshoot [1]. Therefore, the gains of the PI controllers for the PMSM can be determined after acquiring the required control bandwidth represented by $\omega_{cc}$. When $\omega_{cc} = \frac{K_p}{L_{dq}}$ is inserted into Equation 2-36, the tuned gains of the PI controllers are expressed in Equations 2-39, 2-40, and 2-41.

$$K_{p_{dq}} = L_{dq} \cdot \omega_{cc} \tag{2-39}$$

$$K_{i_{dq}} = R_s \cdot \omega_{cc} \tag{2-40}$$

$$K_{b_{dq}} = \frac{1}{K_{p_{dq}}} \tag{2-41}$$

#### 2.2.3.3 Feedforward Control

The implementation of feedforward control in AC machines is pivotal in improving the current regulation performance via compensation of disturbance components, mainly the back EMF and coupling term between the d- and q-axes [1]. Said components have the potential to cause undesired negative side effects to the feedback current control scheme, making the addition of feedforward control crucial to predict and cancel them.

For the implementation of FOC to a PMSM, the feedforward control can compensate for the back EMF and the coupling terms of the output voltage in the d-q axis [1]. According to Equations 2-5 and 2-6, $v_{ds}$ has a disturbance component from the coupling term of the PMSM while $v_{qs}$ has a coupling component and the back EMF of the PMSM.

The feedforward components are therefore expressed in Equations 2-42 and 2-43 as such.

$$v_{ds}^{ff} = -\omega_e L_{qs} i_{qs} \tag{2-42}$$

$$v_{qs}^{ff} = \omega_e L_{ds} i_{ds} + \omega_e \psi_f \tag{2-43}$$

### 2.2.4 Space Vector Pulse Width Modulation (SVPWM)

SVPWM is a modulation technique used for the optimization of three-phase VSI control via PWM signal generation. Rather than modulating each phase independently like Sinusoidal PWM (SPWM), this technique maximizes the utilization of the DC bus voltage of the VSI by viewing the desired output as a vector lying on a two-axis reference frame, allowing for a more precise voltage vector output in terms of its magnitude and phase. SVPWM can produce $15.5\%$ more output voltage than SPWM while maintaining lower torque ripple and switching losses, as shown in Equation 2-44 [1]. The modulation index of the SVPWM can be normalized to result in a value ranging from $0$ to $1$, as seen in Equation 2-45. The utilization of this modulation technique serves as a critical link between the control algorithm and the three-phase VSI.

$$m_{a_{max_{SVPWM}}} = \frac{2V_{mag}}{V_{DC}} = \frac{2}{\sqrt{3}} = 1.1547 \tag{2-44}$$

$$m_{a_{SVPWM}} = \frac{\sqrt{3}V_{mag}}{V_{DC}} \tag{2-45}$$

#### 2.2.4.1 Inverse Park Transformation

The inverse Park transformation transforms a two-axis signal from the rotating reference frame into the stationary reference frame. This is done by transforming the vector represented in the d-q axis to the $\alpha - \beta$ axis. The transformation is carried out by applying a trigonometric projection based on the angle of the rotating reference frame (in this case, the electrical angle of the rotor $\theta_e$), therefore perfectly aligning the d-q axis vector with the stationary reference frame [1].

The role of this transformation is crucial in initializing the SVPWM algorithm. The output of this transformation, lying on the $\alpha - \beta$ domain, holds vital information that can be analyzed geometrically to determine the magnitude and phase of the desired output vector. By doing so, not only can the algorithm determine the sector in which the output vector resides, but it can also ensure valid three-phase voltage phasor generation by the inverter. This transformation, implemented on $v_d$ and $v_q$, is expressed in Equations 2-46 and 2-47.

$$v_\alpha = v_d \cos\theta_e - \sin\theta_e v_q \tag{2-46}$$

$$v_\beta = \sin\theta_e v_d + v_q \cos\theta_e \tag{2-47}$$

In matrix form, it can be seen that the transformation matrix is similar to a counter-clockwise rotation matrix with rotor electric angle $\theta_e$ as the rotation angle.

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix}$$

It is previously mentioned that Park transformation shifts the phase without altering the magnitude of the vector when transforming the $\alpha - \beta$ axis vector into a d-q axis vector. The inverse Park transformation does not alter the magnitude of the vector as well, but the direction of the phase shift is opposite to that of the Park transformation. Equations 2-31 and 2-32 can therefore be rewritten into Equations 2-48 and 2-49 to represent the properties of the inverse Park transformation.

$$r_{\alpha\beta} = r_{dq} \tag{2-48}$$

$$\theta_{\alpha\beta} = \theta_{dq} + \theta_e \tag{2-49}$$

#### 2.2.4.2 Base Vectors and Angle Sectors

As a vector-based modulation strategy, the possible states of the three-phase inverter are viewed as vectors as well. As previously mentioned, the three-phase VSI has three phase legs, each having two possible switching states represented by the function $S$. The possible combinations of the switching states of the three phase legs $S_a$, $S_b$, and $S_c$ are represented in Table 2.2, where it can be seen that there is a total of eight combinations with each causing eight distinctly different vectors. These vectors are the base vectors of the three-phase VSI used for generating the desired voltage vector output.

Table 2.2. Switching States and Corresponding Output Vectors.

| $S_a - S_b - S_c$ | $V_{as}$ | $V_{bs}$ | $V_{cs}$ | **Vector** |
|---|---|---|---|---|
| **0-0-0** | $0$ | $0$ | $0$ | $V_0 = 0\angle 0°$ |
| **1-0-0** | $\frac{2}{3}V_{DC}$ | $-\frac{1}{3}V_{DC}$ | $-\frac{1}{3}V_{DC}$ | $V_1 = \frac{2}{3}V_{DC}\angle 0°$ |
| **1-1-0** | $\frac{1}{3}V_{DC}$ | $\frac{1}{3}V_{DC}$ | $-\frac{2}{3}V_{DC}$ | $V_2 = \frac{2}{3}V_{DC}\angle 60°$ |
| **0-1-0** | $-\frac{1}{3}V_{DC}$ | $\frac{2}{3}V_{DC}$ | $-\frac{1}{3}V_{DC}$ | $V_3 = \frac{2}{3}V_{DC}\angle 120°$ |
| **0-1-1** | $-\frac{2}{3}V_{DC}$ | $\frac{1}{3}V_{DC}$ | $\frac{1}{3}V_{DC}$ | $V_4 = \frac{2}{3}V_{DC}\angle 180°$ |
| **0-0-1** | $-\frac{1}{3}V_{DC}$ | $-\frac{1}{3}V_{DC}$ | $\frac{2}{3}V_{DC}$ | $V_5 = \frac{2}{3}V_{DC}\angle 240°$ |
| **1-0-1** | $\frac{1}{3}V_{DC}$ | $-\frac{2}{3}V_{DC}$ | $\frac{1}{3}V_{DC}$ | $V_6 = \frac{2}{3}V_{DC}\angle 300°$ |
| **1-1-1** | $0$ | $0$ | $0$ | $V_7 = 0\angle 0°$ |

Out of the eight base vectors, there are six nonzero vectors $V_1$, $V_2$, $V_3$, $V_4$, $V_5$, $V_6$ and two zero vectors $V_0$ and $V_7$. When visualized in Figure 2.12, it can be seen that the six nonzero base vectors are positioned 60° apart with each having the same magnitude of $\frac{2}{3}V_{DC}$. This causes a hexagon to be formed, dividing the 360° rotational angle of the two-axis reference frame into six angle sectors defining the operating region of the desired output voltage vector [20]. The sector in which the angle of the voltage vector $\theta_{\alpha\beta}$ lies directly determines the base vectors used for voltage vector generation via switching.

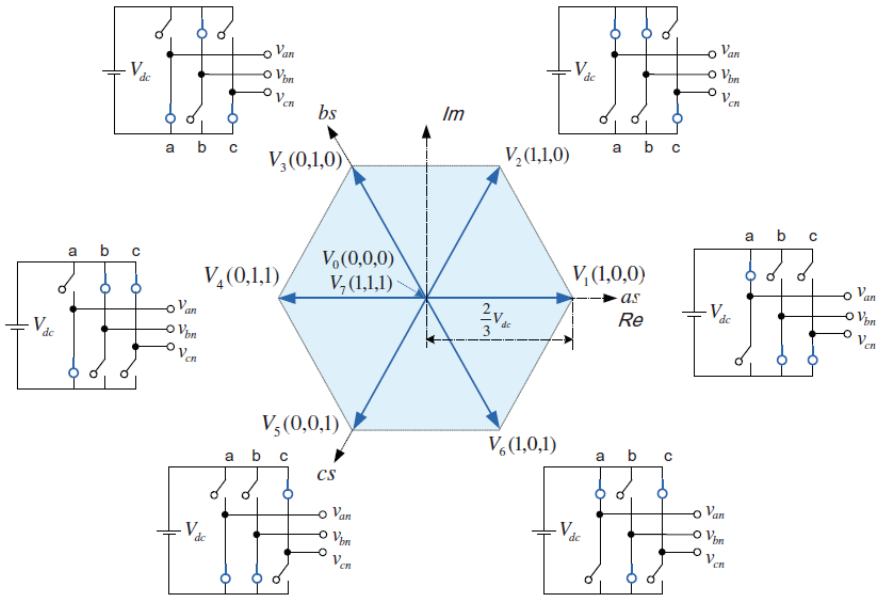


Figure 2.12. SVPWM Base Vectors Forming Six Angle Sectors [1].

#### 2.2.4.3 Base Vector Time Averaging

The angle sector in which the desired vector lies determines which nonzero base vectors are adjacent to it. The process of generating the output voltage vector involves the combination of the two adjacent nonzero base vectors and both zero vectors, meaning that the inverter outputs each vector for a specified duration such that their average over a single switching period results in a resultant vector with the magnitude and phase of the desired output vector [3]. The utilization of the two adjacent nonzero vectors along with the zero vectors can be seen in Figure 2.13.

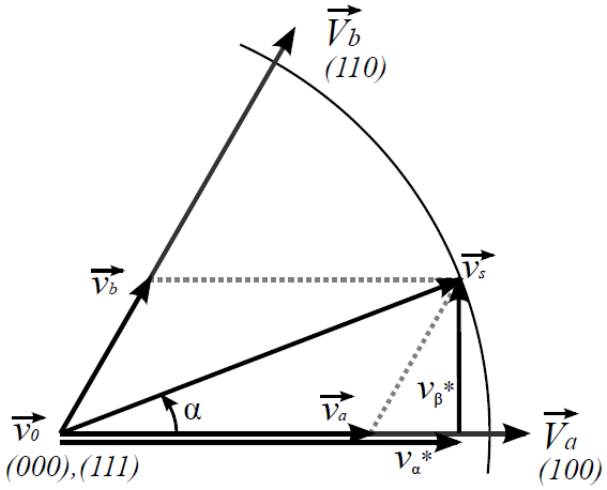


Figure 2.13. Vector $v_s$ Represented by Base Vectors $V_a$ and $V_b$ [3].

The nonzero base vectors $V_a$, $V_b$, and $V_0$ are time averaged with respect to the PWM switching period $T_{PWM}$ to result in vectors $v_a$, $v_b$, and $v_0$. The resultant vectors are then used to generate the desired output vector $v_s$. These vectors are then added to generate $v_s$, as expressed in Equation 2-50. The period of the active base vectors are calculated using the formula given in Equations 2-51 and 2-52 while the total period for both zero vectors use the formula given in Equation 2-53. It can be seen that the time-averaging process for each base vector is influenced by the magnitude of the desired vector $V_{mag}$ compared to the DC bus voltage of the three-phase VSI $V_{DC}$ and the angle between the desired vector and the base vectors $\alpha$

$$\begin{aligned} v_s &= v_a + v_b + v_0 \\ &= \frac{T_a}{T_{PWM}} V_a + \frac{T_b}{T_{PWM}} V_b + \frac{T_0}{T_{PWM}} V_0 \\ &= t_1 V_a + t_2 V_b + t_0 V_0 \end{aligned} \tag{2-50}$$

$$t_1 = \frac{\sqrt{3} V_{mag}}{V_{DC}} \cdot \sin(60^\circ - \alpha^\circ) \cdot T_{PWM} \tag{2-51}$$

$$t_2 = \frac{\sqrt{3} V_{mag}}{V_{DC}} \cdot \sin(\alpha^\circ) \cdot T_{PWM} \tag{2-52}$$

$$t_0 = T_{PWM} - t_1 - t_2 \tag{2-53}$$

#### 2.2.4.4 Symmetrical SVPWM Strategy

The switching sequence of the three-phase VSI determines the sequence of base vector generation for generating the desired voltage and current output. The implementation of a symmetrical modulation strategy for SVPWM is known to reduce the total harmonic distortion (THD) by ensuring that the switching periods of the three-phase out-

puts are calculated to generate symmetrical voltage within each switching period [20]. Figure 2.14 compares the current output when symmetrical modulation is used and when it is not used, showcasing the more harmonic-friendly trait of symmetrical modulation.

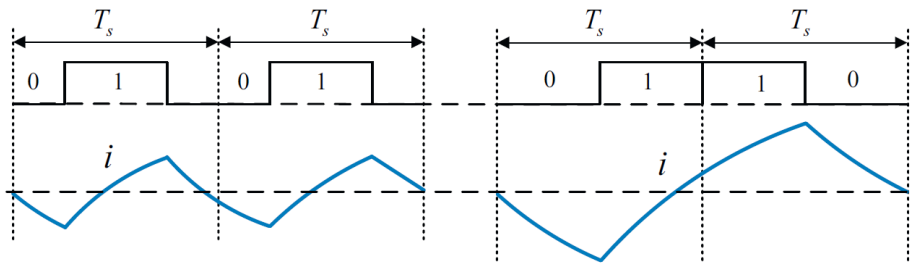


Figure 2.14. Current Generated by Symmetrical vs Non-Symmetrical PWM [1].

In this strategy, the generation period of the two nonzero base vectors follows the previously calculated switching periods $t_1$ and $t_2$, while $t_0$ is split equally at the beginning and end of each switching period using both zero vectors available in order to guarantee symmetry [3]. The switching sequence of the symmetrical modulation strategy for each angle sector is illustrated in Figure 2.15. For example, it can be seen in sector 1 that the first half of the switching period starts with a zero vector generated using the $0-0-0$ state, followed by a nonzero base vector generated by state $1-0-0$, followed by the next state $1-1-0$, and then ended with the zero vector state $1-1-1$. The next half of the switching period uses the reversed sequence of the first half to create symmetry. A look-up table given by Table 2.3 describes the combination of the vector switching times used to calculate the switching time for each phase leg based on the angle sector in which the desired vector resides.

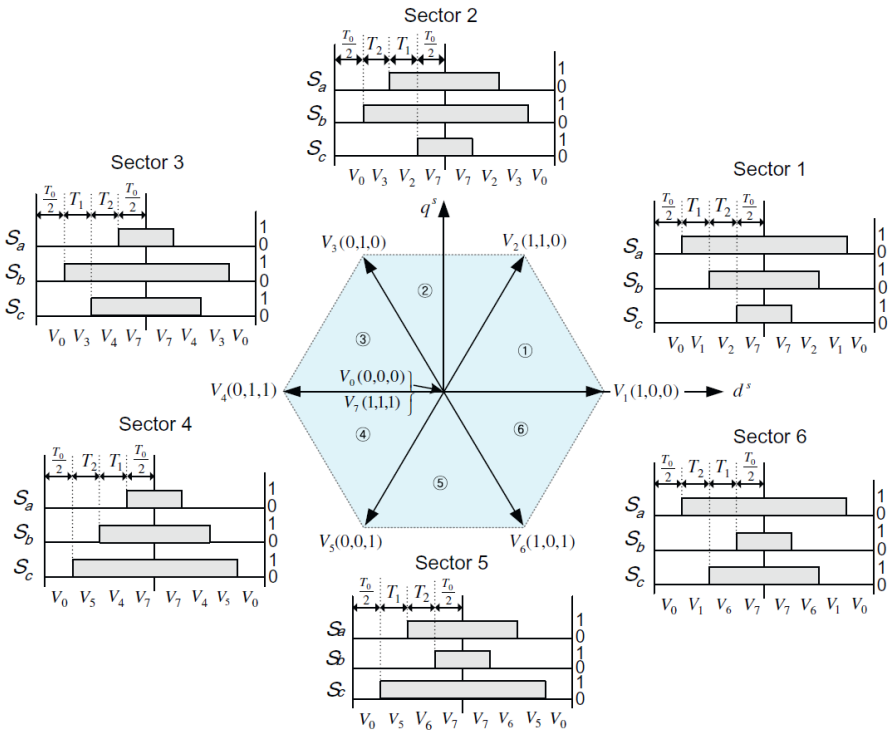


Figure 2.15. Switching Period Sequence in Symmetrical SVPWM [1].

Table 2.3. SVPWM Switching Period Look-Up Table.

| **Sector** | **Phase a** ($t_a$) | **Phase b** ($t_b$) | **Phase c** ($t_c$) |
|---|---|---|---|
| **I** | $t_1 + t_2 + \frac{t_0}{2}$ | $t_2 + \frac{t_0}{2}$ | $\frac{t_0}{2}$ |
| **II** | $t_1 + \frac{t_0}{2}$ | $t_1 + t_2 + \frac{t_0}{2}$ | $\frac{t_0}{2}$ |
| **III** | $\frac{t_0}{2}$ | $t_1 + t_2 + \frac{t_0}{2}$ | $t_2 + \frac{t_0}{2}$ |
| **IV** | $\frac{t_0}{2}$ | $t_1 + \frac{t_0}{2}$ | $t_1 + t_2 + \frac{t_0}{2}$ |
| **V** | $t_2 + \frac{t_0}{2}$ | $\frac{t_0}{2}$ | $t_1 + t_2 + \frac{t_0}{2}$ |
| **VI** | $t_1 + t_2 + \frac{t_0}{2}$ | $\frac{t_0}{2}$ | $t_1 + \frac{t_0}{2}$ |

# CHAPTER III
# METHODOLOGY

## 3.1 Tools and Materials

1. Laptop

   An Acer Aspire A514-54G with a Microsoft Windows 11 Home Single Language 64-bit OS, 8 GB RAM, 7.78 GB physical memory, 11th Gen Intel Core i7-11657 processor (2.8 GHz), and an NVIDIA GeForce MX350 GPU.

2. PLECS 4.9.4 (64-bit)

   The simulation platform for constructing simulations, performing offline simulations, programming the Plexim RT Box 2 for HIL simulations, and real-time data visualization of the performed HIL simulation.

3. MATLAB R2025b

   Utilized as the main platform for performing system frequency response evaluation via Bode plot visualization to assist in system bandwidth determination via parameter tuning.

4. Plexim RT Box 2

   The HIL platform utilized for component modelling and simulation in real time, programmed using PLECS 4.9.4 (64-bit).

5. RT Box 2 Breakout Boards

   Boards utilized as the means of interfacing the microcontroller and the RT Box 2, consisting of the analog and digital breakout board.

6. STM32CubeIDE and STM32CubeMX

   An integrated development environment mainly used for STM32-based microcontroller configuration, code editing, and debugging for parameter monitoring during real-time simulation.

7. STM32 NUCLEO-F446RET6

   The microcontroller used for embedded control system implementation.

8. Filter Board

   The filter used for the real-time HIL simulation, implemented via a hand-soldered blank printed circuit board (PCB).

9. Oscilloscope

   A GW Instek GDS-2304A digital storage oscilloscope (300 MHz, 2 GS/s) utilized

for visualization of signal waveforms during testing.

10. Arbitrary Function Generator (AFG)

    An RS PRO AFG-21025 used for generation of signal waveforms during testing.

11. Troubleshooting Tools

    A set of tools used for troubleshooting purposes, such as a Sanwa CD800a digital multimeter, a JCD908S 80W hand solder, a set of jumper cables, and a Vetus ESD-11 pinset.

### 3.2 Work Flow

This research begins with an initial literature review to grasp a proper understanding of the fundamental theories for determining the initial system specifications, followed by an initial full system offline simulation via PLECS to ensure feasibility. Proper performance of the simulated system will be the main prerequisite for continuing the research. Preparations for the real-time HIL simulations then begins with three pivotal steps: constructing the filter, SVPWM algorithm, and discrete PI controller along with the reference frame transformations.

The filter and SVPWM algorithm are to be tested to evaluate its filtering capabilities and ensure proper vector output in terms of magnitude and phase. Proper closed-loop bandwidth planning is a crucial step and is done via Bode plot evaluation of the closed-loop system to tune the cut-off frequency of the used filter and controller gains. After that, the real-time HIL simulation is performed. The process is visualized in Figure 3.1.

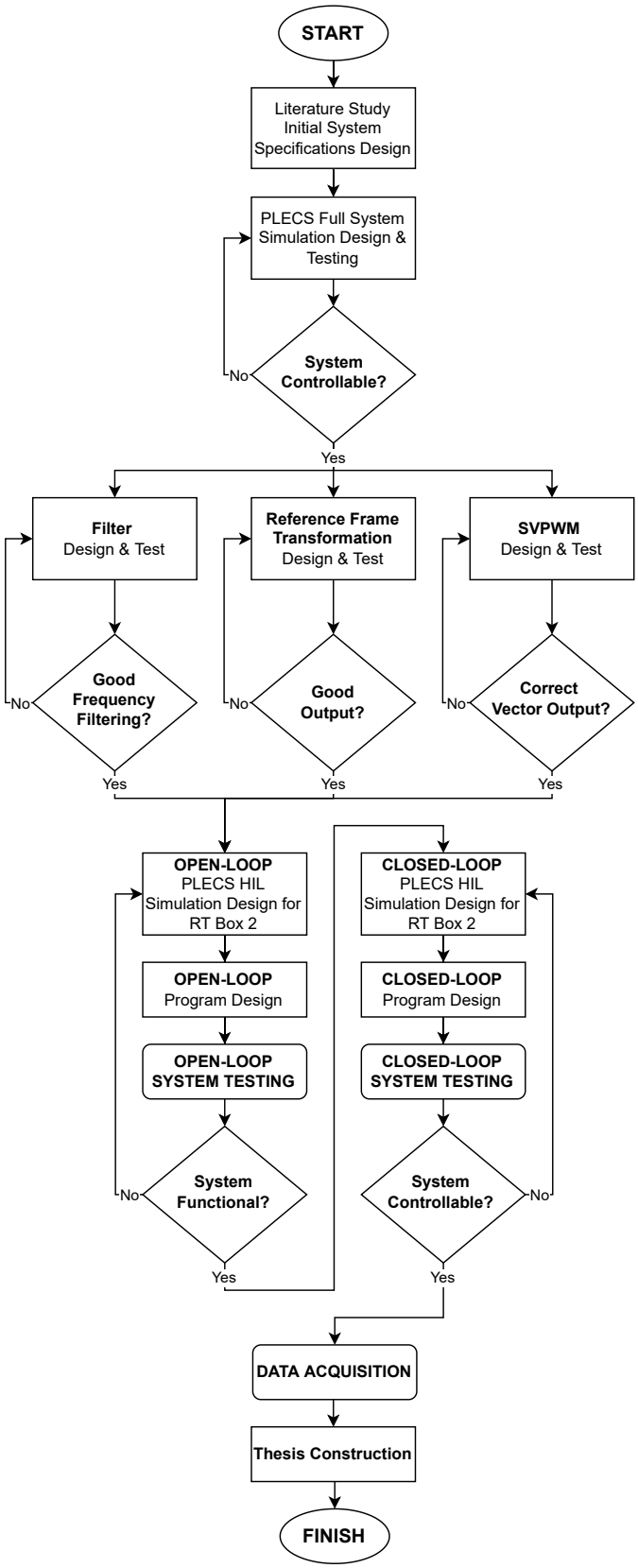


Figure 3.1. Research Flow.

### 3.2.1 Closed-Loop System Design

The FOC algorithm is to be tested as a closed-loop system with feedback. The measurements to be fed back to the system are $i_a$, $i_b$, $\theta_e$, and $\omega_e$. The current measurements $i_a$ and $i_b$ are used to identify the instantaneous error of the output current. The electrical rotor measurement $\theta_e$ is used for reference frame transformations, specifically Park and inverse Park transformation. The electrical speed measurement $\omega_e$ is used to calculate the feedforward terms previously expressed in Equations 2-42 and 2-43.

The current measurements $i_a$ and $i_b$ both contain high-frequency components due to the switching of the three-phase VSI. The two considerations to be kept in mind in designing accurate feedback for the system are the microcontroller sampling method and high-frequency noise filtering. The microcontroller can measure voltages using the provided pins and then convert them into digital values corresponding to the determined bit resolution using analog-to-digital conversion (ADC). Setting the timing and method for the ADC sampling is crucial in ensuring accurate feedback measurement. Additionally, the filter to be used is a low-pass filter (LPF) capable of filtering high-frequency signal components. Due to the inputs being sinusoidal currents, the filter must be designed to ensure strong noise-cancelling capabilities while minimizing attenuation and phase shift to the inputs at their fundamental frequency. The design of the feedback is discussed further in an upcoming section.

The PI controllers are used to realize a responsive control system for driving the PMSM under dynamic load. For an embedded implementation via a microcontroller, the controllers are to be discretized to construct a pair of discrete PI controllers. The controllers are to be tuned using the previously mentioned pole-zero cancellation method to result in a stable system with a bandwidth $\omega_{cc}$ that can be adjusted via pole-zero cancellation tuning of the controller gains $K_p$, $K_i$, and $K_b$. Determining the bandwidth $\omega_{cc}$ of the controllers require proper planning to ensure a controllable system, this can be done by evaluating the resulted closed-loop system via Bode plots to view its frequency spectrum gain attenuation and phase shift. Design and preliminary evaluation of the discrete PI controllers along with the bandwidth of the closed-loop system is discussed further in an upcoming section.

The SVPWM algorithm is used to calculate the switching times for the PWM output signals of the microcontroller. The algorithm must be able to properly identify the magnitude and angle of the desired output voltage vector, thereby able to perform mathematical computations to identify both parameters using the $\alpha - \beta$ voltages. Furthermore, the microcontroller must be able to use the computed vector switching times to generate the six symmetrical PWM output signals controlling the three-phase VSI. The outputs must also incorporate a dead time delay to prevent a destructive shoot-through scenario. The design of the SVPWM algorithm along with the setup of the microcontroller to gen-

erate the outputs are discussed further in an upcoming section.

Therefore, the closed-loop system is illustrated in Figure 3.2. The design and setting of each component must be planned thoroughly with the main goal of creating a controllable and responsive system. Additionally, hardwares used to represent each system component are to be assigned.

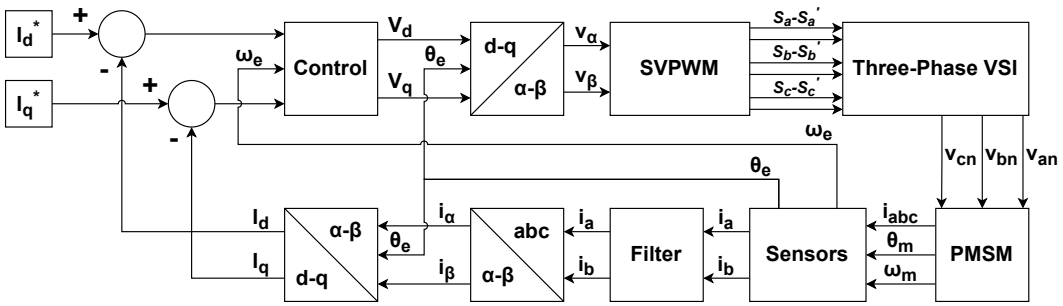


Figure 3.2. Closed-Loop System.

### 3.2.2 Simulation Design

#### 3.2.2.1 Permanent Magnet Synchronous Motor (PMSM) Model

The PMSM controlled by the system is represented by a permanent magnet synchronous machine block provided by PLECS, the composition of which can be seen in Figure 3.3. The block models a PMSM using a voltage behind reactance (VBR), where the electrical circuit of the PMSM is described in circuit form as can be seen in Figure 3.4. The circuit incorporates time-varying inductance matrices, making the model numerically less efficient than the rotor reference frame model, but increases the accuracy of the electrical variables. The parameters of the PMSM are modelled based on a BSM90N-175 low-inertia servo, the specifications of the PMSM are given in Table 3.1. It can be seen in the table that $L_d = L_q$, therefore the modelled PMSM is a nonsalient SPMSM with its generated torque consisting of only electromagnetic torque.

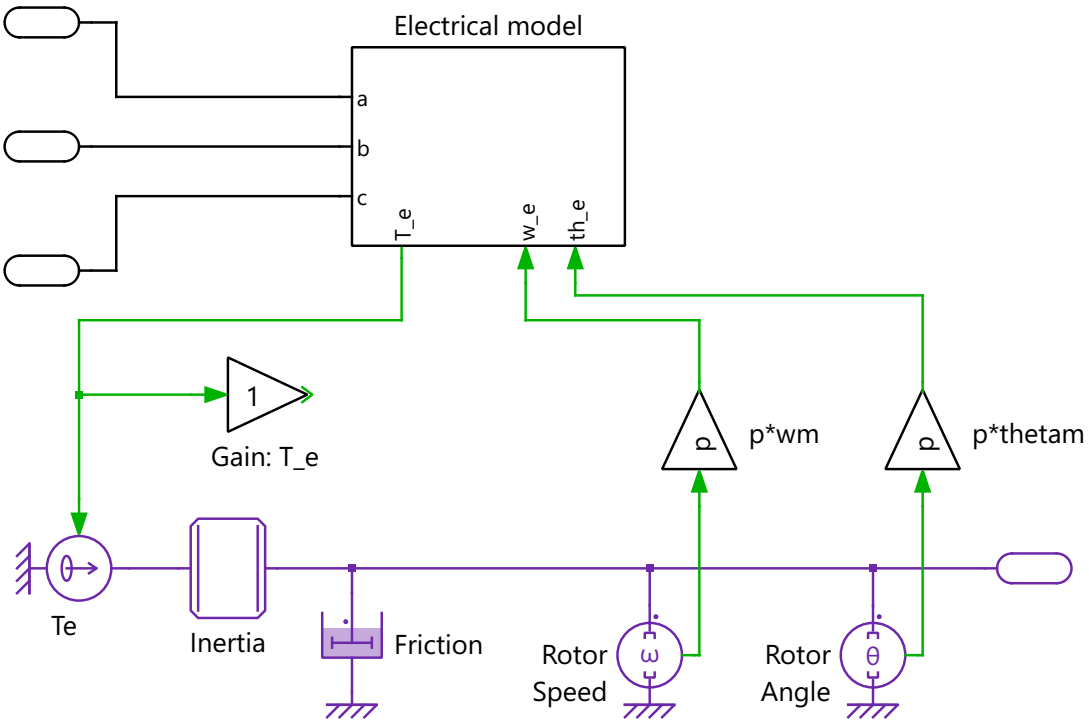


Figure 3.3. PLECS PMSM Block Subsystem.

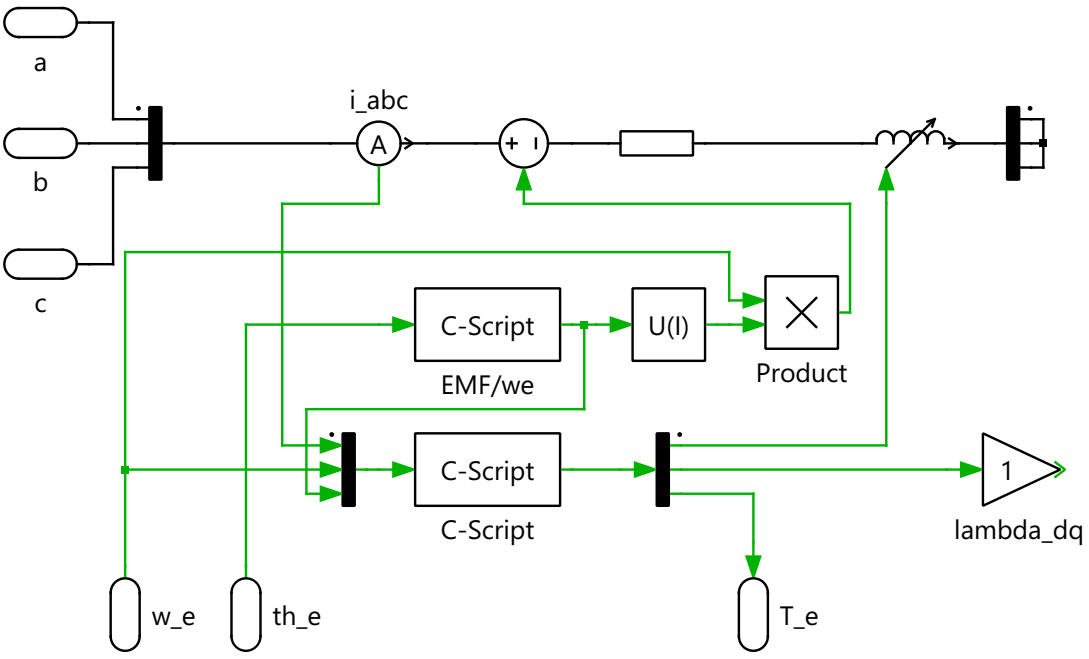


Figure 3.4. Voltage Behind Reactance PMSM Model.

Table 3.1. PMSM Specifications.

| **Parameter** | **Value** |
|---|---|
| Continuous Current | $7.8$ A |
| Peak Current | $28.1$ A |
| Rated Speed at 300 Volts | $4000$ rpm |
| Maximum Speed | $7000$ rpm |
| Torque Constant ($K_t$) | $0.853$ Nm/A |
| Voltage Constant ($K_e$) | $72.8$ Vpk/krpm |
| Resistance ($R_s$) | $1.24\ \Omega$ |
| Direct Inductance ($L_d$) | $4.15$ mH |
| Quadrature Inductance ($L_q$) | $4.15$ mH |
| Pole Pairs ($P$) | $4$ |
| Inertia ($J$) | $3.389\ \mathrm{kg \cdot cm^2}$ |
| Flux ($\psi_f$) | $0.174$ Wb |

#### 3.2.2.2 Three-Phase Voltage Source Inverter (VSI) Model

The three-phase VSI is modelled using the three-phase VSI block provided by PLECS. The semiconductor symbol can be toggled to be either insulated gate bipolar transistors (IGBTs) or metal-oxide semiconductor field-effect transistors (MOSFETs), although doing so does not affect the electrical properties of the model. The block uses the sub-cycle average mode, where the three-phase VSI is comprised of three half-bridge power modules modelled using controlled voltage and current sources. The control inputs are the relative on-times with their values ranging from 0 to 1. The sub-cycle average is chosen over the switched configuration due to it being able to correctly account for blanking times when both switches are turned off, a trait needed to prevent shoot-through scenarios when dead time delays are implemented in the input control signals. The subsystem of the sub-cycle average-configured three-phase VSI model is illustrated in Figure 3.5.

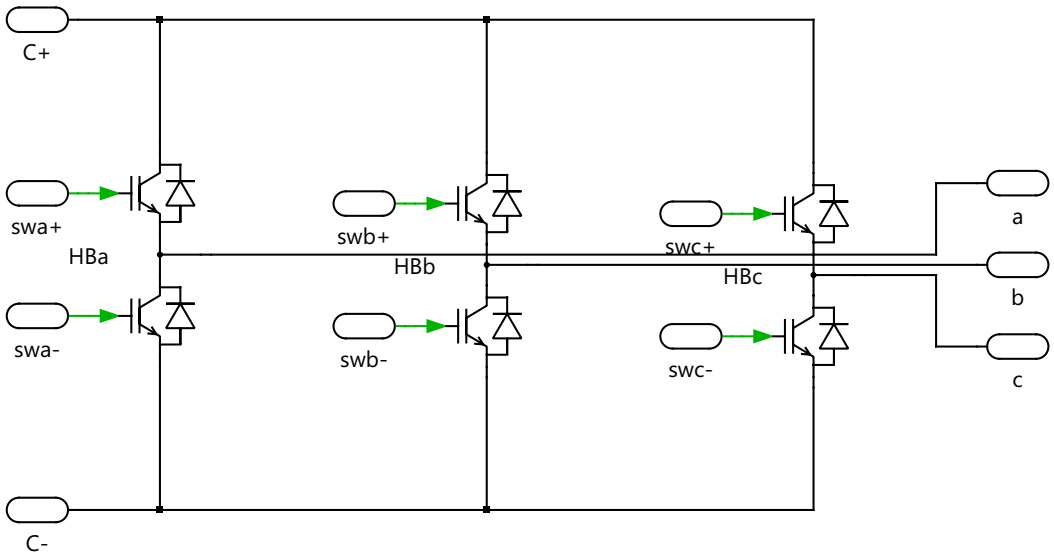


Figure 3.5. PLECS Three-Phase VSI using Sub-Cycle Average.

#### 3.2.2.3 Hardware-in-the-Loop (HIL) Simulation Design

The simulation to be uploaded to the Plexim RT Box 2 for performing the real-time HIL simulation is illustrated in Figure 3.6. The simulation can be divided into four parts: plant, preprocessing, outputs, and STM32 interface. The plant consists of the three-phase VSI, PMSM, sensors, and mechanical load driven by the PMSM. A pair of three-phase electrical meters are used to measure the three-phase line-to-neutral voltage, line-to-line voltage, and phase currents. Additionally, a pair of rotational sensors are used to detect the mechanical rotor angle and speed. The mechanical load to be driven by the PMSM is comprised of a rotational inertia and damping block with inertia $J$ and damping $B$, making the mechanical dynamics during transient state and steady state to be as expressed in Equations 3-1 and 3-2.

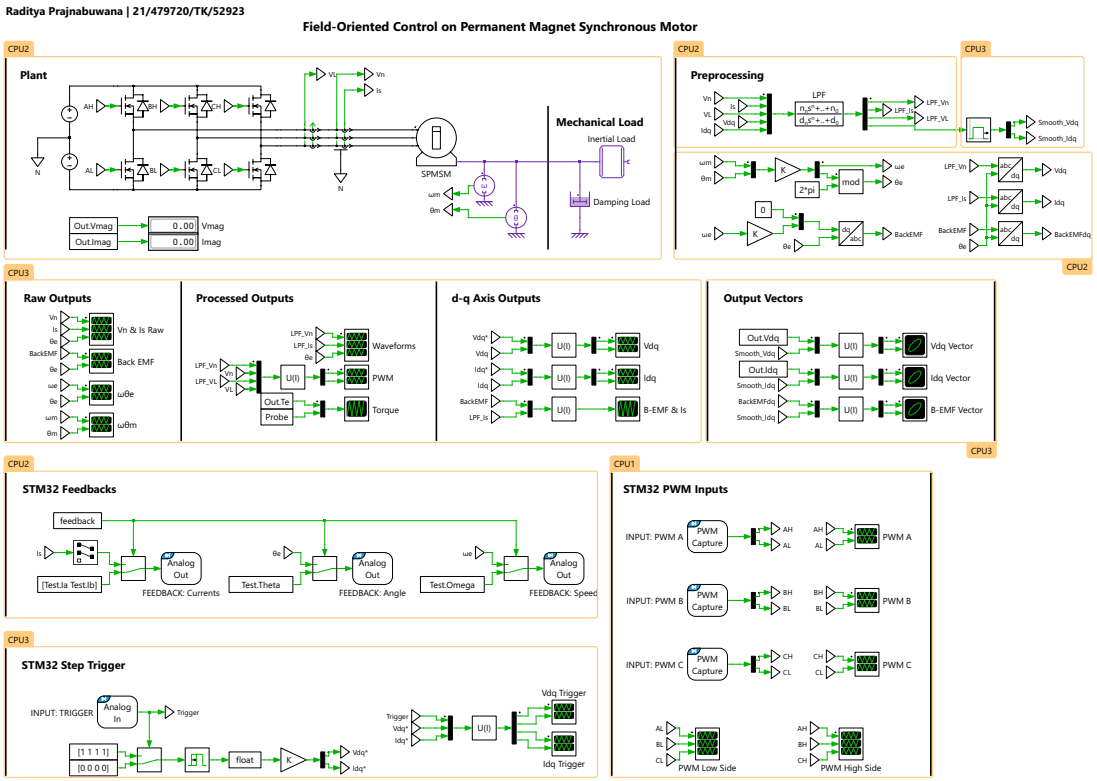


Figure 3.6. HIL Simulation Design.

$$J \cdot \frac{d\omega_m}{dt} = T_e - B \cdot \omega_m \tag{3-1}$$

$$\omega_m = \frac{T_e}{B} \tag{3-2}$$

The preprocessing part is mostly comprised of algorithm blocks for performing reference frame transformations and measurement filtering. The rotor mechanical angle and speed are both used to calculate the rotor electrical angle and speed, which are then used to calculate numerous critical measurements. The outputs of the simulation are visualized using scope blocks that have been configured to visualize the results of the real-time simulation in the form of sinusoidal waveforms, PWM square wave signals, or even vectors.

The STM32 interface part of the simulation located on the bottom half of the simulation is comprised of RT Box blocks used to configure the pins of the Plexim RT Box 2 as the input or output pins. The feedback measurements needed by the embedded control system are to be sent to the microcontroller using the analog output pins of the Plexim RT Box 2, while the six PWM signals generated by the microcontroller are to be read by Plexim RT Box 2 using its digital input pins. The pin assignment of the inputs and outputs for interfacing the STM32 NUCLEO-F446RET6 microcontroller are described in Table 3.2.

Table 3.2. Pin Assignments for HIL Simulation.

| **Parameter** | **Pin** |
|---|---|
| FEEDBACK: Currents A | Analog Out 15 |
| FEEDBACK: Currents B | Analog Out 14 |
| FEEDBACK: Angle | Analog Out 13 |
| FEEDBACK: Speed | Analog Out 12 |
| INPUT: Trigger | Analog In 15 |
| INPUT: PWM A High | Digital In 16 |
| INPUT: PWM A Low | Digital In 17 |
| INPUT: PWM B High | Digital In 18 |
| INPUT: PWM B Low | Digital In 3 |
| INPUT: PWM C High | Digital In 20 |
| INPUT: PWM C Low | Digital In 5 |

The Plexim RT Box 2 is capable of multi-tasking when it comes to representing the models in real-time, this is done via its three available central processing units (CPUs). This enables the division of the total simulation load to allow for heavy simulations to be carried out in real-time. The most crucial aspect to be kept in mind when doing this is choosing the discretization step size for each CPU due to it directly determining the final load each CPU will endure during the real-time simulation. The prioritized parts of the simulation are the STM32 interface and the plant due to them being the main foundation of the simulation. The STM32 interface part of the simulation must have the shortest sampling time to enable for a higher PWM frequency and shorter dead time delay for the microcontroller. After uploading the simulation to the Plexim RT Box 2 and tuning the sampling time for each part of the simulation, the sampling time for each CPU is determined to be as described in Table 3.3 with the simulation load division per CPU being as visualized in Figure 3.6. The final specification of the designed HIL simulation to be used for real-time testing is described in Table 3.4.

Determining the PWM frequency must be done while taking the PWM capture discretization step size of the RT Box CPU into consideration. The Nyquist-Shannon sampling theorem constitutes that with a sampling period of $2.5\ \mu\mathrm{s}$, the PWM output must have a period at least twice longer to prevent aliasing, as stated in Equation 3-3. The PWM frequency is therefore determined to be $8\ \mathrm{kHz}$ to ensure a smoother output and sampling. The PWM period will then be $125\ \mu\mathrm{s}$, meaning the HIL platform is capable of sampling the PWM output 50 times in one period. The dead time delay implemented to

the PWM output for shoot-through prevention must follow the same rule while minimizing its period to ensure minimum influence towards the output vector, making the dead time delay period $5\ \mu$s.

$$T_{PWM} >= 2T_{sampling} \tag{3-3}$$

Table 3.3. Plexim RT Box 2 CPU Discretization Step Size.

| **CPU** | **Step Size** |
|---|---|
| CPU1 | $2.5\ \mu$s |
| CPU2 | $2.5\ \mu$s |
| CPU3 | $5\ \mu$s |

Table 3.4. HIL Simulation Specifications.

| **Parameter** | **Value** |
|---|---|
| PMSM Configuration | Voltage Behind Reactance |
| PMSM Model | BSM90N-175 |
| Three-Phase VSI Configuration | Sub-Cycle Average |
| PWM Capture Step Size | $2.5\ \mu$s |
| VSI DC Bus Voltage | 300 V |
| Inertia Load ($J$) | $0.001\ \mathrm{kg \cdot m^2}$ |
| Damping Load ($B$) | $0.75\ \mathrm{Nm \cdot s/rad}$ |
| PWM Frequency ($f_{PWM}$) | 8 kHz |
| PWM Dead Time Delay ($t_{dead}$) | $5\ \mu$s |

Table 3.1 shows that the peak current the PMSM can operate at is $28.1$ A, thereby this magnitude will set the operating range of the PMSM during testing. Since the constructed control system is strictly for current regulation, the motor control strategy will therefore be torque control. Equation 2-10 explains that the q-axis current is the sole current component within the d-q axis that generates torque for an SPMSM with a linear relationship, therefore the test conducted will have the q-axis current be varied to generate a varied but linear torque, while the d-axis current will be set to zero.

Furthermore, Equation 3-2 explains that the steady-state mechanical speed of the PMSM has a linear relationship with the generated torque. This in turn means that the q-axis current not only has a linear relationship with the generated torque, but also the steady-state mechanical speed of the PMSM. Therefore, the steady-state mechanical

speed of the PMSM can be calculated according to the set reference value of the q-axis current. Since the maximum q-axis current to be tested is at 28.1 A, the maximum operated mechanical speed, electrical speed, and the electrical frequency of the PMSM can therefore be calculated using Equations 3-2, 2-11, and 2-12. Table 3.5 contains the testing range of the PMSM.

Table 3.5. HIL Simulation PMSM Testing Range.

| **Parameter** | **Value** |
|---|---|
| d-Axis Current ($I_d$) | 0 A |
| Maximum q-Axis Current ($I_q$) | 28.1 A |
| Maximum Torque ($T_e$) | 29.302 Nm |
| Maximum Mechanical Speed ($\omega_m$) | 39.070 rad/s |
| Maximum Electrical Speed ($\omega_e$) | 156.278 rad/s |
| Maximum Electrical Frequency ($f_e$) | 24.872 Hz |

#### 3.2.2.4 PLECS Simulation for Feasibility Testing

Figure 3.7 illustrates the offline simulation used for preliminary system testing. The system is divided into five parts: inverter, controls, SVPWM, load, and additional components for monitoring parameters. The inverter section consists of a three-phase VSI connected to a PMSM with a DC voltage source and sensors for sensing the DC bus voltage, rotor rotational speed, and rotor mechanical angle.

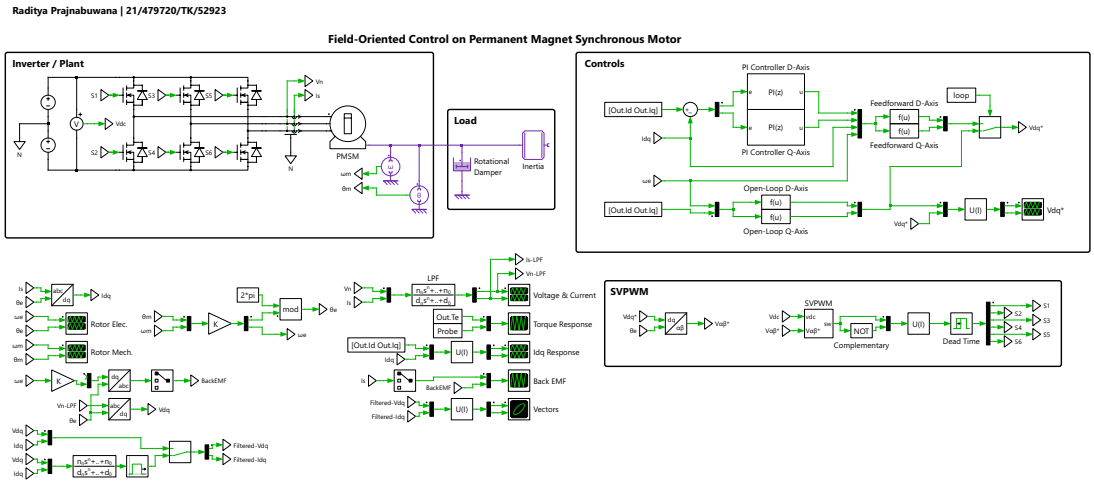


Figure 3.7. PLECS Offline Simulation.

The controls section holds both the open-loop and closed-loop control scheme. The closed-loop scheme uses two discrete PI controllers with their sampling time set to

equal to the PWM period of 125 $\mu s$ and a set of functions for adding the feedforward terms for decoupling compensation. The open-loop scheme uses only a pair of functions to calculate the outputs using the current reference values and the electrical rotor speed of the rotor as the inputs.

The SVPWM section contains the inverse Park transformation block provided by PLECS and an SVPWM generator block used to generate the upper switching signals. The generated upper PWM signals are then fed into a logical operator block set to NOT to generate the lower PWM signals, all of which are then given the determined dead time delay of 5 $\mu s$. The load section contains the mechanical load driven by the controlled PMSM. The load consists of a rotational damper block with a configurable damping coefficient $B$ and an inertia block with a configurable inertia $J$. The specifications of the PLECS simulation is described in Table 3.6

Table 3.6. PLECS Simulation Specifications.

| **Parameter** | **Value** |
|---|---|
| PMSM Configuration | Voltage Behind Reactance |
| PMSM Model | BSM90N-175 |
| Three-Phase VSI Configuration | Sub-Cycle Average |
| Controller Integration Method | Trapezoidal |
| Controller Sample Time | $125\ \mu$s |
| PWM Frequency | $8$ kHz |
| PWM Dead Time Delay | $5\ \mu$s |
| DC Voltage | $300$ V |
| Inertia Load ($J$) | $0.001\ \mathrm{kg \cdot m^2}$ |
| Damping Load ($B$) | $0.75\ \mathrm{Nm \cdot s/rad}$ |

### 3.2.3 Input

#### 3.2.3.1 Instantaneous Sampling

The previously determined feedback measurements are to be sampled by the microcontroller instantaneously, making it crucial to time the sampling properly to ensure accurate current measurement. The switching of the three-phase VSI causes the real-time current measurement to be full of ripples, but it is possible for the microcontroller to acquire an instantaneous current measurement accurate to that of the fundamental component of the sinusoidal output current if the sampling occurs when the symmetrical PWM outputs are generating the zero vectors [1]. This is visualized in Figure 3.8.

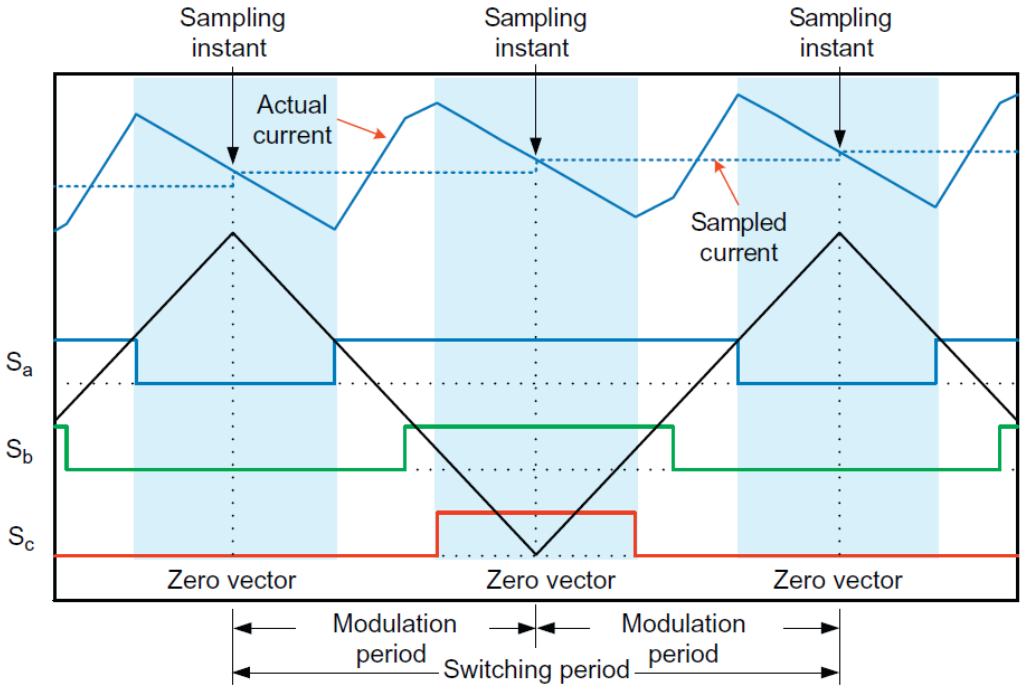


Figure 3.8. Timing of Instantaneous Sampling [1].

This principle therefore allows for the microcontroller to instantaneously sample the current measurements during a total of two periods within one PWM period: during the generation of vector $V_0$ when all three PWM phases are low (0-0-0) or vector $V_7$ when all three PWM phases are high (1-1-1). However, sampling twice in one PWM period means the control loop must be efficient enough to be carried out by the CPU of the microcontroller in under half the PWM period. For this implementation, the current measurement is to be sampled only once each PWM period during the generation of vector $V_0$.

#### 3.2.3.2 Low-Pass Filter (LPF)

For this implementation, a first-order analog resistor-capacitor (RC) filter is selected, the schematic of which is illustrated in Figure 3.9. The RC filter when used as an LPF is able to attenuate high frequency signal components at approximately $-20\,\mathrm{dB/dec}$. The cut-off frequency $f_c$ of this filter is determined by the resistance of its resistor and capacitance of its capacitor, as given in Equation 3-4 [21].

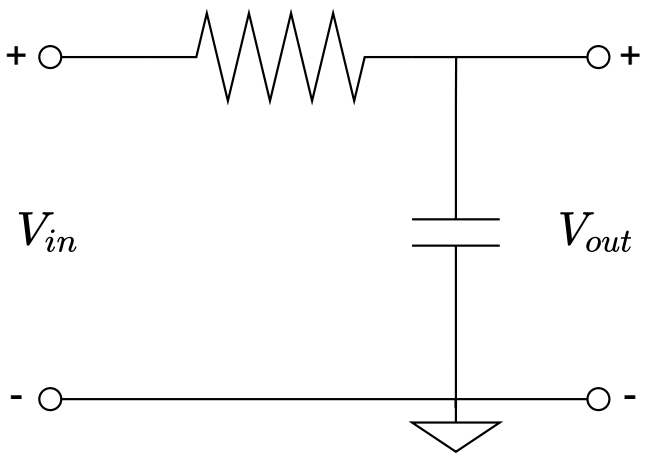


Figure 3.9. RC LPF Schematic.

$$f_c = \frac{1}{2\pi RC} \tag{3-4}$$

The open-loop transfer function of this filter is expressed in Equation 3-5. Figure 3.10 visualizes the Bode plot of an RC LPF when its cut-off frequency $f_c$ is set at $800$ Hz. At its cut-off frequency, the RC LPF exhibits a $-3$ dB magnitude and $-45°$ phase shift. Setting the cut-off frequency of the LPF used to filter out the feedback current measurements is equivalent to determining the bandwidth of the feedback current measurements due to it filtering out signals with frequencies higher than the cut-off frequency. However, it is crucial to remember for sinusoidal signal implementations that the frequencies the filter lets through will experience phase shift as the frequency gets closer to the cut-off frequency. Therefore, the cut-off frequency of the RC LPF for this implementation is determined to be $800\ Hz$.

$$H(s) = \frac{1}{1 + sRC} \tag{3-5}$$

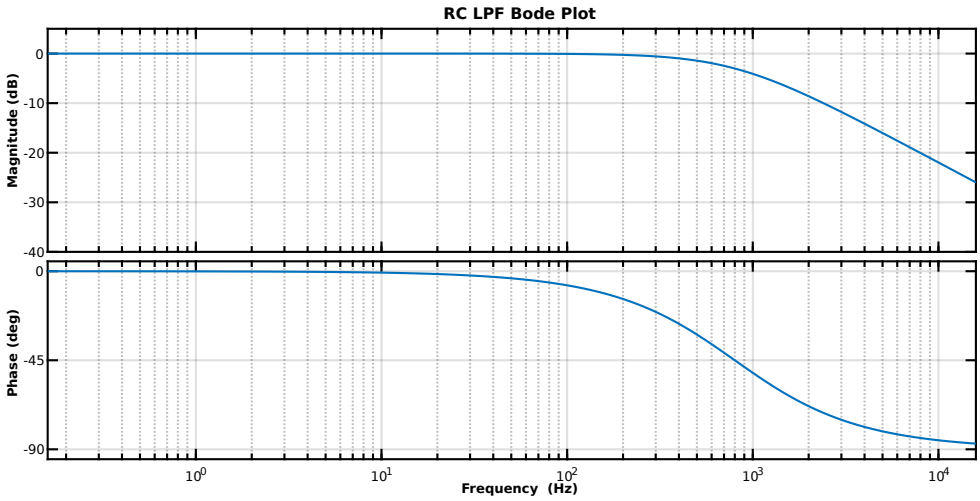


Figure 3.10. Bode Magnitude and Phase Plot of an RC LPF.

### 3.2.4 Control Scheme

#### 3.2.4.1 Discrete PI Controller

The PI controller implementation in this work is done in an embedded environment, hence making the controller a discrete PI controller. Methods for controller discretization, such as the Forward Euler, Backward Euler, and Tustin method, all yield different performance characteristics when implemented. The Tustin method or trapezoidal approximation or bilinear transform provides faster dynamic response and less oscillations at the price of a slightly higher overshoot during transient scenarios, making it the adopted method of controller discretization method [18]. The implemented discrete PI controller is given in Equation 3-6. The gains are to be tuned using the aforementioned pole-zero cancellation method.

$$u[n] = u[n-1] + K_p \cdot (e[n] - e[n-1]) + K_i \cdot \frac{T_s}{2}(e[n] + e[n-1]) \tag{3-6}$$

#### 3.2.4.2 Control Bandwidth

As previously mentioned, using the pole-zero cancellation method to tune the gains of the PI controllers will cause the closed-loop system to be a stable first-order system. However, the bandwidth of the system must be determined properly to ensure closed-loop stability without causing the sinusoidal feedback to be altered too significantly. To do so, the PI controller along with the RC LPF are to be evaluated via Bode plot.

$$f_{control} <= \frac{f_{PWM}}{20} \tag{3-7}$$

A rule of thumb states that if the embedded system samples the feedback measure-

ments once for every PWM output period, the bandwidth of the control system should not be more than one-twentieth of the PWM frequency, hence this is expressed in Equation 3-7 [1]. Assuming an $8$ kHz PWM switching frequency, this rule of thumb constitutes that the maximum recommended bandwidth for the system is $400$ Hz. To ensure an accurate feedback, the frequency response of the closed-loop system is targeted to exhibit no magnitude attenuation and a maximum phase shift of $5°$ to the maximum operated electrical frequency of $24.872$ Hz. Additionally, the step response of the closed-loop system must is targeted to have a rise time faster than $10$ ms, maximum overshoot of $10\%$, and settling time to within $1\%$ faster than $50$ ms.

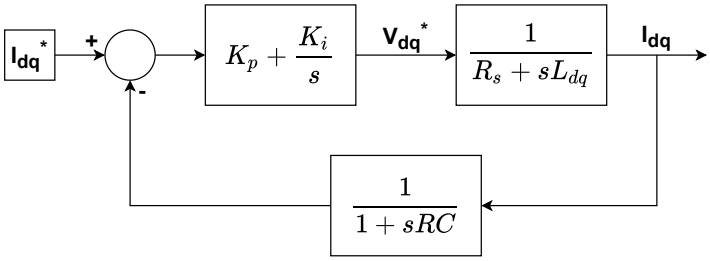


Figure 3.11. Closed-Loop System for Bode Plot Evaluation.

Bode plot evaluation is performed to the closed-loop representation of the system, illustrated in Figure 3.11. Following the rule of thumb expressed by Equation 3-7, the bandwidth of the PI controllers are initially set at $400$ Hz. The Bode plots of the LPF, the closed-loop system with unity gain (without the LPF), and the closed-loop system with the LPF are visualized in Figure 3.12. The maximum operated electrical frequency of $24.87$ Hz experiences a magnitude gain of $0.0042$ dB and phase shift of $-1.79°$.

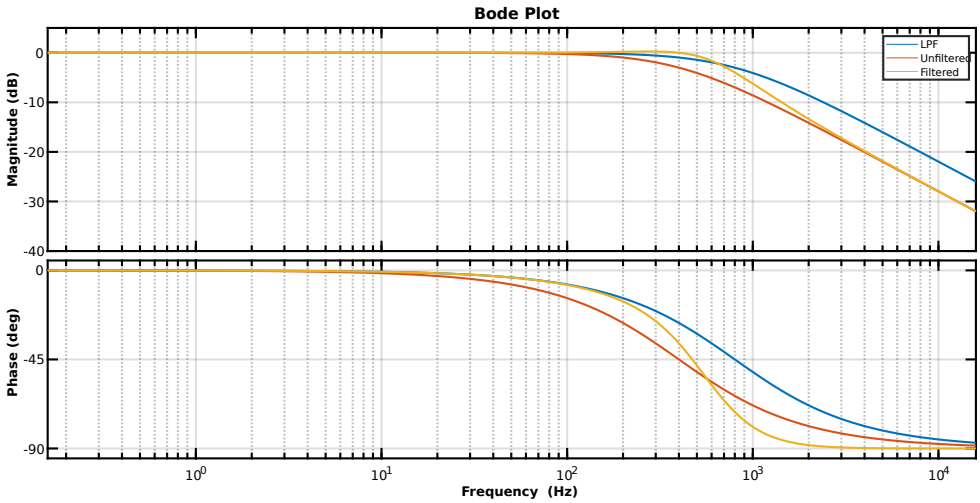


Figure 3.12. Bode Plots of the Closed-Loop System.

The step response of the system is visualized in Figure 3.13, where the closed-loop system is compared with the system with unity gain (without the LPF). It can be seen that the step response of the closed-loop system with unity gain exhibits properties of a critically-damped system, while that of the closed-loop system using the LPF is an underdamped system. The closed-loop system has a rise time of $0.625$ ms, an overshoot of $6.7\%$, and a settling time of $1.62$ ms. The Bode plot and step response of the system shows that the system suffices the previously determined criteria of a fast and stable system with accurate feedback, thereby the PI controllers are to be tuned using pole-zero cancellation with a bandwidth of $400$ Hz.

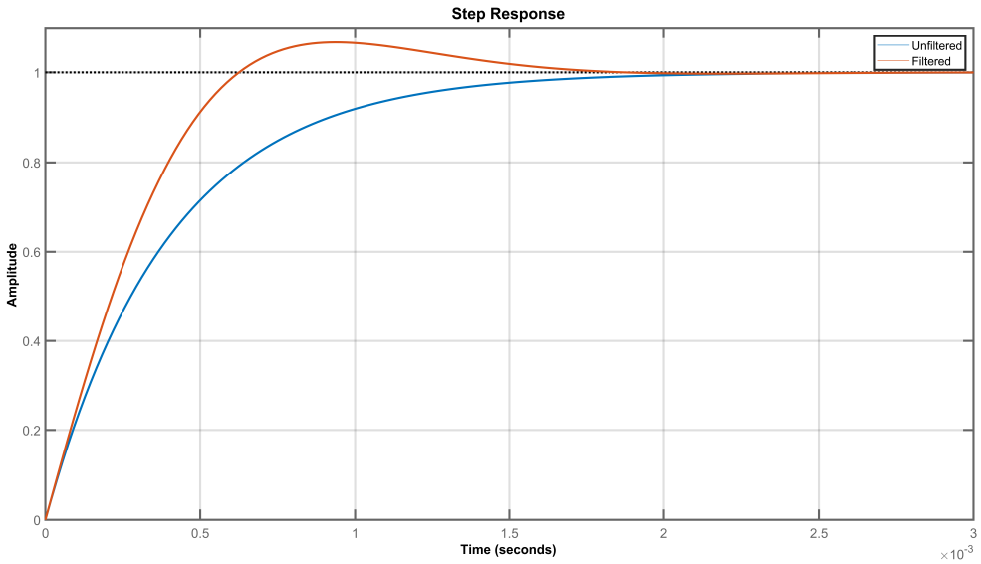


Figure 3.13. Step Response Comparison of the Closed-Loop System.

Inserting the values in Table 3.1 into Equations 2-39, 2-40, and 2-41, the resulted gains of the two PI controllers are as described in Table 3.7. Figure 3.14 visualizes the root locus plot of the open-loop system, where the zero provided by the PI controller cancels the pole of the PMSM.

Table 3.7. PI Controller Gains.

| **Parameter** | **Formula** | **Value** |
|---|---|---|
| $K_{p_{dq}}$ | $L_{dq} \cdot \omega_{cc}$ | 10.430 |
| $K_{i_{dq}}$ | $R_s \cdot \omega_{cc}$ | 3116.460 |
| $K_{b_{dq}}$ | $\frac{1}{K_{p_{dq}}}$ | 0.096 |

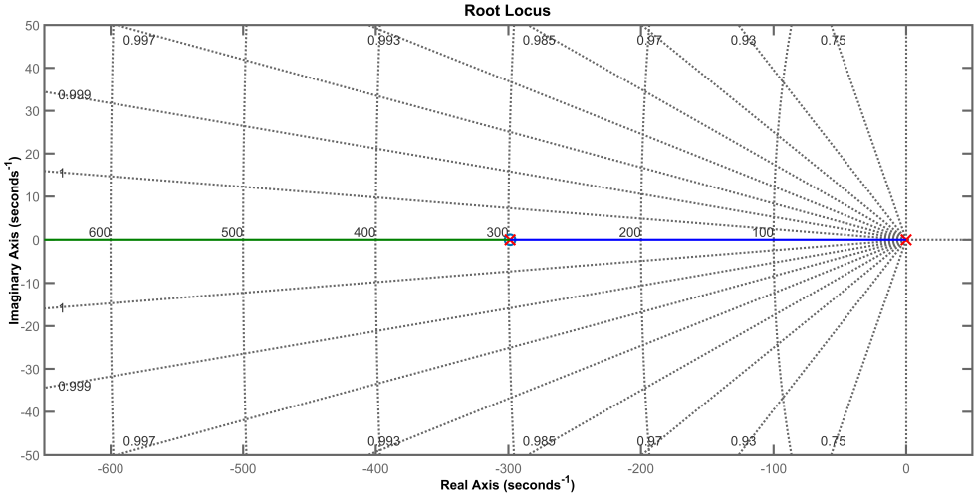


Figure 3.14. Open-Loop Root Locus of the Pole-Zero Cancelled System.

#### 3.2.4.3 Open-Loop and Closed-Loop Control Scheme

The control scheme differs depending on whether the test conducted in an open-loop test or a closed-loop test. For an open-loop test, $V_d$ and $V_q$ are calculated assuming steady-state condition without the use of any controller, as given in Equations 3-8 and 3-9 with the feedforward terms given in Equations 2-42 and 2-43. For a closed-loop test, the output of the PI controllers are summed with the feedforward terms as a method of decoupling compensation. A key difference in the two methods is the feedforward terms in the closed-loop control scheme are comprised of the inductive voltage terms generated by the feedback currents, while the same voltage terms in the open-loop control scheme are calculated using the current reference values. The open-loop and closed-loop schemes are illustrated in Figures 3.15 and 3.16

$$v_{ds} = R_s i_{ds} + v_{ds}^{ff} \tag{3-8}$$

$$v_{qs} = R_s i_{qs} + v_{qs}^{ff} \tag{3-9}$$

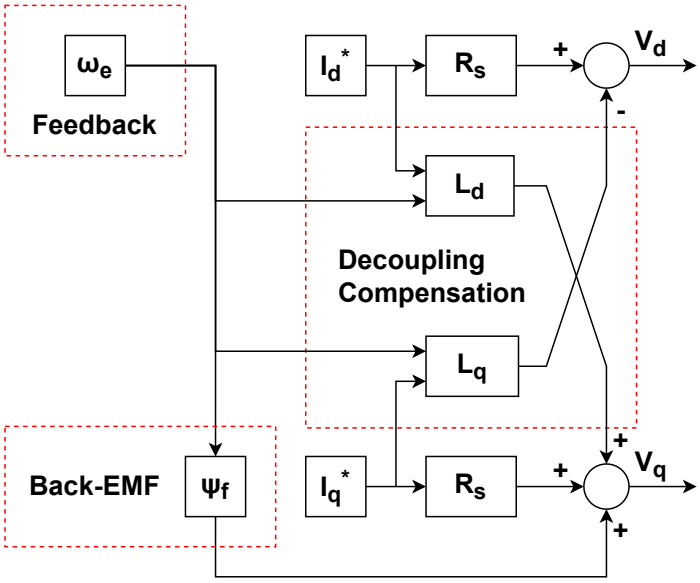


Figure 3.15. Open-Loop Control.

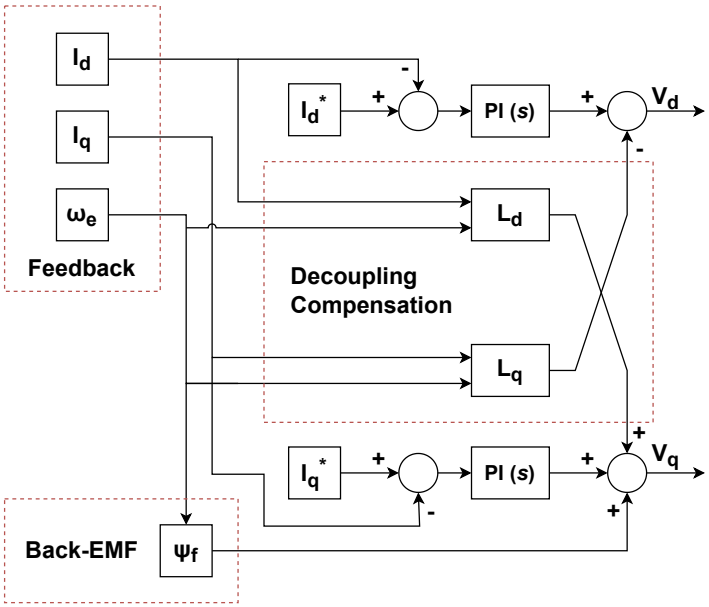


Figure 3.16. Closed-Loop Control.

### 3.2.5 Output

#### 3.2.5.1 Inverse Park Transformation

The inverse Park transformation algorithm for embedded implementation is constructed to ensure valid vector magnitude and angle in the stationary reference frame while ensuring a fast computation period for the microcontroller. Equations 2-51 and 2-52 explains that the calculation of the vector switching periods both incorporate a con-

stant ($\frac{\sqrt{3}}{V_{DC}}T_{PWM}$), therefore this constant is implemented to the inverse Park transformation. The magnitude of the vector $V_{mag}$ is not included in the constant due it already being a part of the d-q voltages transformed into the stationary reference frame voltages $v_\alpha$ and $v_\beta$. The constructed inverse Park transformation is expressed in Equations 3-10 and 3-11.

$$v_\alpha = (v_d \cos\theta_e - v_q \sin\theta_e)(\frac{\sqrt{3}}{V_{DC}}T_{PWM}) = (V_{mag}\cos(\theta_{\alpha\beta}))(\frac{\sqrt{3}}{V_{DC}}T_{PWM}) \quad (3\text{-}10)$$

$$v_\beta = (v_d \sin\theta_e + v_q \cos\theta_e)(\frac{\sqrt{3}}{V_{DC}}T_{PWM}) = (V_{mag}\sin(\theta_{\alpha\beta}))(\frac{\sqrt{3}}{V_{DC}}T_{PWM}) \quad (3\text{-}11)$$

#### 3.2.5.2 SVPWM Vector Angle Sector Detection

When transformed into the stationary reference frame, the desired output vector becomes two sinusoidal signals varying over time with a $90°$ phase difference between $v_\alpha$ and $v_\beta$. With these values, the angle of the desired output voltage vector can be determined. The constructed algorithm used to determine the sector of the desired vector based on the instantaneous value of $v_\alpha$ and $v_\beta$ is illustrated in Figure 3.17.

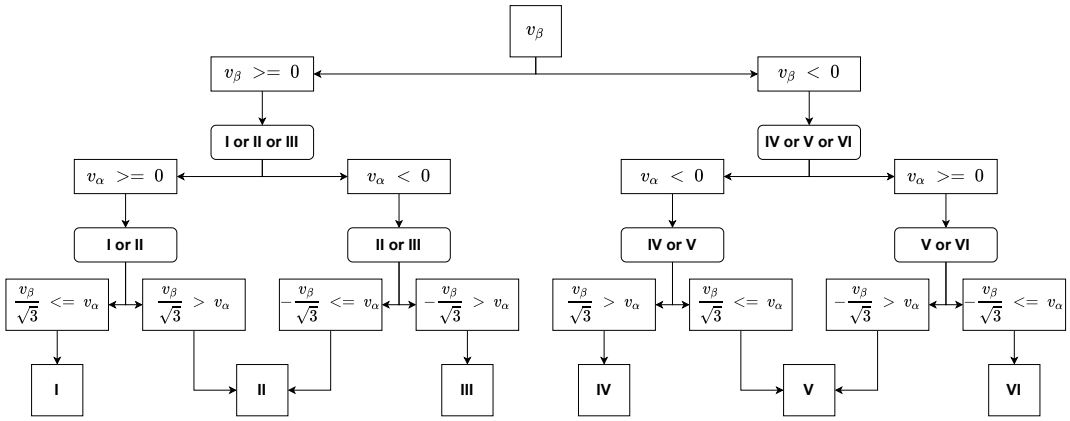


Figure 3.17. Vector Angle Detection.

#### 3.2.5.3 SVPWM Vector Period Calculation

The vector periods are calculated according to the previously explained Equations 2-51, 2-52, and 2-53. However, in this implementation the vector magnitude $V_{mag}$ and the angle $\alpha$ is obtained via $v_\alpha$ and $v_\beta$. The angle sectors have different trigonometric properties due to their angle range, therefore it is crucial to use trigonometric identity equations to get the correct arithmetic combination of $v_\alpha$ and $v_\beta$ for each sector. After careful evaluation, the equations used to calculate the vector periods using $v_\alpha$ and $v_\beta$ acquired via Equations 3-10 and 3-11 for each angle sector are described in Table 3.8. The

vector periods $t_1$ and $t_2$ are calculated differently for each angle sector, while $t_0$ follows Equation 2-53 regardless the angle sector the desired vector is in.

Table 3.8. Vector Period $t_1$ and $t_2$ Calculation for Each Angle Sector.

| **Sector** | $t_1$ | $t_2$ |
|---|---|---|
| **I** | $\frac{1}{2}(\sqrt{3}v_\alpha - v_\beta)$ | $v_\beta$ |
| **II** | $\frac{1}{2}(\sqrt{3}v_\alpha + v_\beta)$ | $\frac{1}{2}(-\sqrt{3}v_\alpha + v_\beta)$ |
| **III** | $v_\beta$ | $\frac{1}{2}(-\sqrt{3}v_\alpha - v_\beta)$ |
| **IV** | $\frac{1}{2}(-\sqrt{3}v_\alpha + v_\beta)$ | $-v_\beta$ |
| **V** | $\frac{1}{2}(-\sqrt{3}v_\alpha - v_\beta)$ | $\frac{1}{2}(\sqrt{3}v_\alpha - v_\beta)$ |
| **VI** | $-v_\beta$ | $\frac{1}{2}(\sqrt{3}v_\alpha + v_\beta)$ |

### 3.2.6 Microcontroller Implementation

#### 3.2.6.1 Clock Configuration

The STM32 NUCLEO-F446RET6 microcontroller has an internal clock that can be utilized to output a frequency ranging up to $180\ \mathrm{MHz}$. The clock is therefore programmed via STM32CubeMX to time all of the sequences related to the construction of the embedded control system. The high speed clock (HSE) is configured to use the built-in crystal/ceramic resonator of the microcontroller to generate a frequency of $8\ \mathrm{MHz}$.

The HSE frequency is then configured by the PLL to generate a PLLCLK frequency of $160\ \mathrm{MHz}$, which is then chosen by the system clock multiplexer to be used as the main system clock (SYSCLK). The frequency of the SYSCLK is then configured further to generate a $160\ \mathrm{MHz}$ frequency for the APB2 timer clocks (APB2CLK). The resulted frequency of the APB2CLK is the resulted frequency to be used by the timer triggering the PWM generation and ADC sampling.

#### 3.2.6.2 Symmetrical PWM Generation for Symmetrical SVPWM

The STM32 NUCLEO-F446RET6 microcontroller has numerous timers that can be used for PWM generation. Among the vast options, only timer 1 (TIM1) and timer 8 (TIM8) are capable of generating complementary PWM pairs. For this implementation, TIM8 is chosen and configured so that its first, second, and third channel are utilized for complementary PWM signal pair generation.

The PWM generation of TIM8 is triggered by the APB2CLK, generating an initial frequency of $160\ \mathrm{MHz}$. The timer TIM8 has a number of components used to determine the output frequency of the PWM generated. The prescaler (PSC) divides the frequency of the APB2CLK to further adjust the trigger frequency. The counter increments an

accumulated value each time the clock ticks. The accumulated value of the counter can be limited by setting the counter period or auto reload register (ARR) value. However, the counter mode can be configured to either count starting from 0 up to the set ARR (up counting mode), or count down from the ARR to zero (down counting mode), or count up from zero to the ARR and then count down to zero (center aligned mode). Therefore, the clock-triggered counter can be configured to function as a triangular carrier used in signal modulation, the frequency of which in this implementation determines the frequency of the PWM outputs.

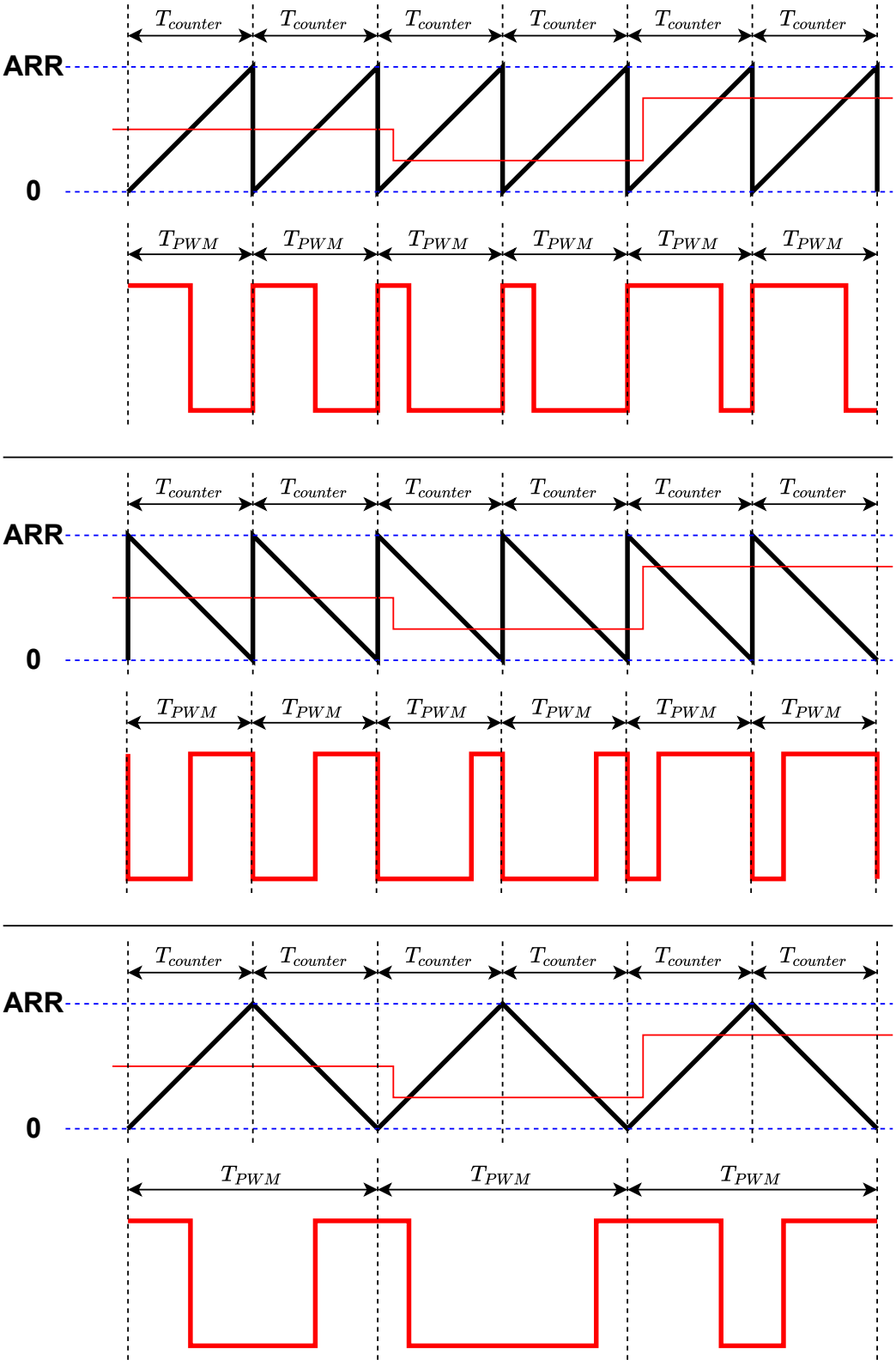


Figure 3.18. Carrier Alignment for PWM Generation.

Carrier-based PWM generation can use either a trailing edge carrier via up counting counter, leading edge carrier via down counting counter, or dual edge triangular car-

rier via center-aligned counter, as is illustrated in Figure 3.18 [13]. The triangular carrier generated using the center aligned mode counter is suitable for generating symmetrical PWM outputs, thereby making it the used option for this implementation [11]. However, the PWM frequency will be half the counter frequency due to the counter requiring one period to count up from zero to the ARR and another period to count down to zero forming one triangle. Therefore, the timer TIM8 can be configured to generate the desired PWM frequency using Equation 3-12.

$$f_{PWM} = \frac{f_{APB2CLK}}{2 \cdot (ARR) \cdot (PSC)} \tag{3-12}$$

In Figure 3.18, it can be seen that the instantaneous amplitude of the modulated signal directly determines the state of the PWM output via comparison with that of the carrier signal. Therefore, it can be concluded that the ratio between the average amplitude of the modulated signal in one PWM period and the maximum amplitude of the carrier signal, which in this implementation is the set value of the ARR, is the duty cycle of the PWM in the corresponding PWM period, as expressed in Equation 3-13. In the STM32 NUCLEO-F446RET6 microcontroller, this ratio for the duty cycle can be achieved by entering the amplitude of the modulated signal into the compare (CCR) variable, the value of which is to be compared with the set ARR to determine the duty cycle of the PWM in one period. Therefore, the duty cycle of the PWM output for this implementation is expressed in Equation 3-14.

$$D_{period} = \frac{V_{average}}{ARR} \tag{3-13}$$

$$D_{STM32} = \frac{CCR_X}{ARR} \tag{3-14}$$

For this implementation, the symmetrical SVPWM algorithm calculates the vector periods $t_1$, $t_2$, and $t_0$ which will then be used to determine the switching period of the three half-bridges in the three-phase VSI according to the previously explained Table 2.15. The resulting switching period is the on time for the phase leg, this value can be used to get the desired duty cycle of the PWM output according to Equation 3-15. To output the desired duty cycle using the STM32 NUCLEO-F446RET6 microcontroller, Equations 3-14 and 3-15 can be used to calculate the CCR value representing the requested duty cycle for the PWM period, as expressed in the resulted Equation 3-16.

$$D = \frac{t_{abc}}{T_{PWM}} \tag{3-15}$$

$$CCR_X = \frac{t_{abc}}{T_{PWM}} \cdot ARR \tag{3-16}$$

The range of the value at which the amplitude of the modulated signal and the carrier signal can be compared determines the resolution of the duty cycle, therefore a higher

ARR translates to higher duty cycle resolution. For this implementation, the ARR is determined to be $10000$ to allow for a resolution of over $13$ $bits$, as can be calculated using Equation 3-17 [11]. Using Equation 3-12, the configuration of the timer TIM8 to generate the desired PWM frequency is described in Table 3.9.

$$D_{Resolution} = \log_2(ARR) \tag{3-17}$$

Table 3.9. TIM8 Configuration for PWM Generation.

| **Parameter** | **Value** |
|---|---|
| APB2 Timer Clock Frequency (APB2CLK) | $160$ MHz |
| TIM8 Channel 1 | PWM Generation CH1 CH1N |
| TIM8 Channel 2 | PWM Generation CH2 CH2N |
| TIM8 Channel 3 | PWM Generation CH3 CH3N |
| TIM8 Counter Prescaler (PSC) | $1$ |
| TIM8 Counter Mode | Center Aligned Mode 1 |
| TIM8 Counter Period (ARR) | $10000$ |
| PWM Frequency ($f_{PWM}$) | $8$ kHz |

#### 3.2.6.3 PWM Dead Time Delay

The implementation of dead time delay $t_{dead}$ affects the output PWM signal by decreasing the on time of the signal by $t_{dead}$ while simultaneously increasing the off time by $t_{dead}$. In symmetrical PWM, the right and left side of both the on time and off time of the signal gets reduced and increased in that order by half the dead time delay ($\frac{t_{dead}}{2}$). This effect applies for both complementary PWM signals, as can be seen in Figure 3.19.

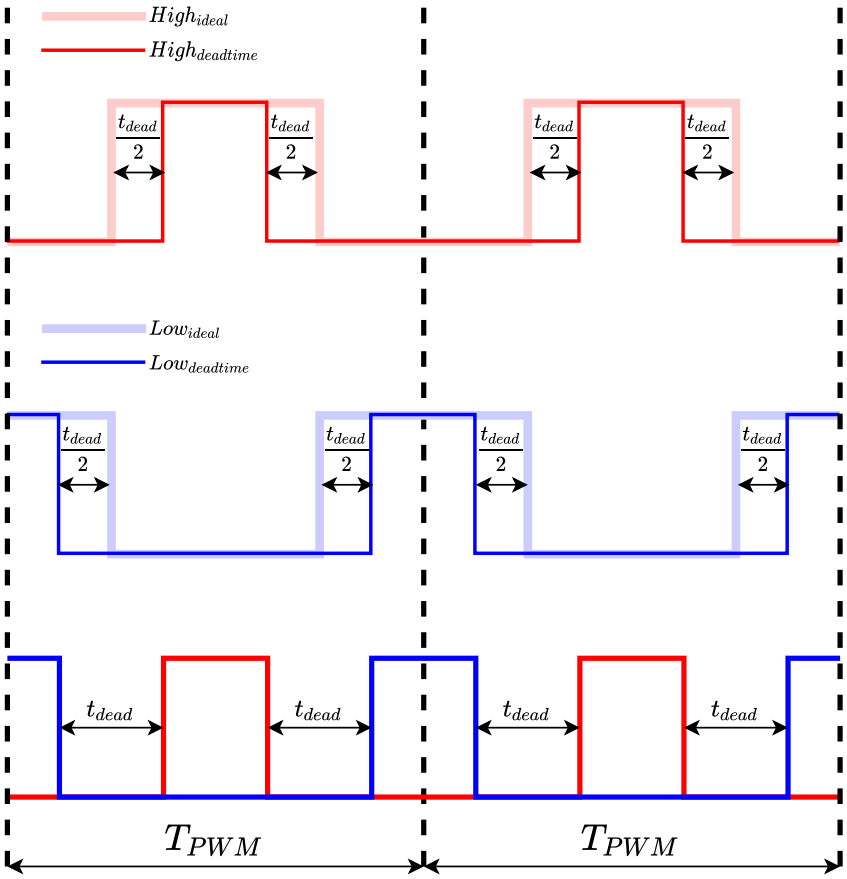


Figure 3.19. PWM Dead Time Delay Effect.

The on time reduction means the PWM period will have no on time if the initial on time is less than $t_{dead}$, causing the output PWM signal to have a different duty cycle to the one initially requested by the system. Therefore, there exists a minimum duty cycle value at which the PWM still outputs an on time that can be detected by the sampling of the HIL platform CPU operating at a sampling period of $2.5\ \mu s$.

Abiding to the previously mentioned Nyquist-Shannon theorem, the minimum on time that can be sampled by the HIL platform is expressed in Equation 3-18. This can be used to calculate the minimum initial on time of the system before dead time delay is implemented through Equation 3-19, which then can be used to calculate the minimum duty cycle via Equation 3-20. Any duty cycle below the calculated $8\%$ threshold will result in no on time and thereby will cause the output duty cycle to be $0\%$. This shows how the implementation of dead time can affect the PWM output in terms of duty cycle accuracy and vector output accuracy in SVPWM.

$$t_{on_{sample}} = 2T_{HIL} = 2 \cdot 2.5\mu s = 5\mu s \tag{3-18}$$

$$t_{on_{init_{min}}} = t_{on_{sample}} + t_{dead} = t_{on_{sample}} + 5\mu s = 10\mu s \tag{3-19}$$

$$D_{min} = \frac{t_{on_{init_{min}}}}{T_{PWM}} = \frac{10\mu s}{125\mu s} = 8\% \tag{3-20}$$

According to Table 2.3, the shortest possible on time is $\frac{t_0}{2}$ depending on the angle sector in which the output vector resides. At its lowest, this value can reach its lowest value of $0$ s when the angle $\alpha$ in Equations 2-51 and 2-52 is $30°$ and when $\frac{\sqrt{3}V_{mag}}{V_{DC}}$ is equal to $1$, simultaneously resulting in the maximum value of $(t_1+t_2)$. Since the constant $\frac{\sqrt{3}V_{mag}}{V_{DC}}$ can be referred to as the normalized modulation index previously expressed in Equation 2-45, the minimum on time is dependent on the modulation index of the set reference with a higher modulation index resulting in a lower on time for the $t_0$, as expressed in Equation 3-21. Inserting the $10$ $\mu$s minimum on time from Equation 3-19, the resulting maximum normalized modulation index is $0.84$, as seen in Equation 3-22.

$$t_{min} = (\frac{t_0}{2})_{min} = \frac{1}{2}(T_{PWM} - (t_1 + t_2)_{max}) = \frac{1}{2}T_{PWM}(1 - m_a) \tag{3-21}$$

$$m_{a_{max}} = 1 - (\frac{2t_{min}}{T_{PWM}}) = 1 - (\frac{2 \cdot (10\mu s)}{125\mu s}) = 0.84 \tag{3-22}$$

If the steady-state output during real-time testing requires an output voltage vector with a magnitude that causes the $m_a$ to be over $0.84$, the PWM outputs used to generate the voltage vectors will be inaccurate due to the on time being eliminated by the dead time delay. When applied to a three-phase VSI connected to a $300$ V DC voltage, a modulation index of $0.84$ will result in a maximum allowable voltage vector magnitude of $145.492$ V. The maximum voltage vector magnitude tested can be evaluated inserting the previously determined maximum tested current setpoint of $28.1$ A into Equations 2-7 and 2-8, the result of which shows that the test operates within the safe operating area with the vector magnitude being a safe $64.658$ V with an amplitude modulation index $m_a$ of $0.37$.

For this implementation, the STM32 NUCLEO-F446RET6 microcontroller can be configured to insert a dead time delay into its generated complementary PWM output pairs using STM32CubeMX. The microcontroller views the dead time period as a number of clock ticks, with a longer dead time delay requiring more ticks. The PWM is generated using timer TIM8 which uses the APB2CLK as its trigger source, therefore the frequency of the APB2CLK directly determines how many ticks it takes to generate the dead time delay. With a clock period of $6.25$ ns, as can be calculated via Equation 3-23, and a dead time delay of $5$ $\mu$s, it takes a total of $800$ clock periods to create the same period of time, as expressed in Equation 3-24. Therefore, the desired dead time $d$ is equal to $800$ $T$ with $T$ representing the clock period, as shown in Equation 3-25.

$$T_{clock} = \frac{1}{f_{clock}} = \frac{1}{160 \text{ MHz}} = 6.25 \text{ ns} \tag{3-23}$$

$$\frac{t_{dead}}{T_{clock}} = \frac{5\ \mu\text{s}}{6.25\ \text{ns}} = 800 \tag{3-24}$$

$$d = 800T \tag{3-25}$$

Table 3.10 contains the equations used to calculate the number of ticks $x$ used to generate the desired dead time delay based on the ratio of the desired dead time delay to the period of the clock [11]. For a $5\ \mu$s dead time delay comprised of $800T$, the resulting ticks are expressed in Equation 3-26 as $18$. According to Table 3.10, the final number of ticks for a dead time delay of $800T$ begins at $224$ ticks. Therefore, the final tick count to be programmed to generate the desired $5\ \mu$s dead time delay is $242$, as expressed in Equation 3-27. This value is to be inserted into the dead time section in the break and dead time management output configuration of the timer TIM8.

Table 3.10. Dead Time Delay Tick Equations.

| $d$ **Range** | **Equation** | **Ticks** ($x$) |
|---|---|---|
| $0T\ -\ 127T$ | $d = xT$ | $x <= 127$ |
| $128T\ -\ 254T$ | $d = (32 + x)2T$ | $128 <= x <= 191$ |
| $256T\ -\ 504T$ | $d = (32 + x)8T$ | $192 <= x <= 223$ |
| $512T\ -\ 1008T$ | $d = (32 + x)16T$ | $x >= 224$ |

$$\begin{aligned}(32 + x)16T &= 800T \\ 32 + x &= \frac{800T}{16T} = 50 \\ x &= 50 - 32 = 18\end{aligned} \tag{3-26}$$

$$x = 224 + 18 = 242 \tag{3-27}$$

#### 3.2.6.4 Injected Analog-to-Digital Conversion (ADC) for Instantaneous Sampling

For this implementation, the instantaneous sampling, timed to sample the current measurements once every PWM period during the generation of the zero vector $V_0$ (0-0-0), is realized via utilization of the analog-to-digital converters of the STM32 NUCLEO-F446RET6 microcontroller. The converter ADC1 is to be configured to perform injected analog-to-digital conversion (ADC) due to its ability to be triggered by timer TIM8, the same timer configured to generate the PWM outputs. The injected conversion is set to be triggered by a PWM channel provided by TIM8, set for PWM generation with no output. The PWM is to have its duty cycle set to the near maximum and trigger the injected conversion at its falling edge to ensure the sampling occurs at the very middle of the zero vector generating part of the three-phase PWM outputs when all three $S$ states are

0. The setup is described in Table 3.11 and the desired triggered injected ADC timing is visualized in Figure 3.20.

Table 3.11. ADC1 and TIM8 Configuration for Injected Conversion.

| **Parameter** | **Value** |
|---|---|
| ADC1 Scan Conversion | Enabled |
| ADC1 End of Conversion | End of all conversions |
| ADC1 Injected Trigger Source | TIM8 Capture Compare 4 |
| ADC1 Injected Trigger Edge | Falling Edge |
| TIM8 Dead Time | 242 |
| TIM8 Counter Period (ARR) | 10000 |
| TIM8 Channel 4 Mode | PWM Generation No Output |
| TIM8 Channel 4 Pulse | 9990 |

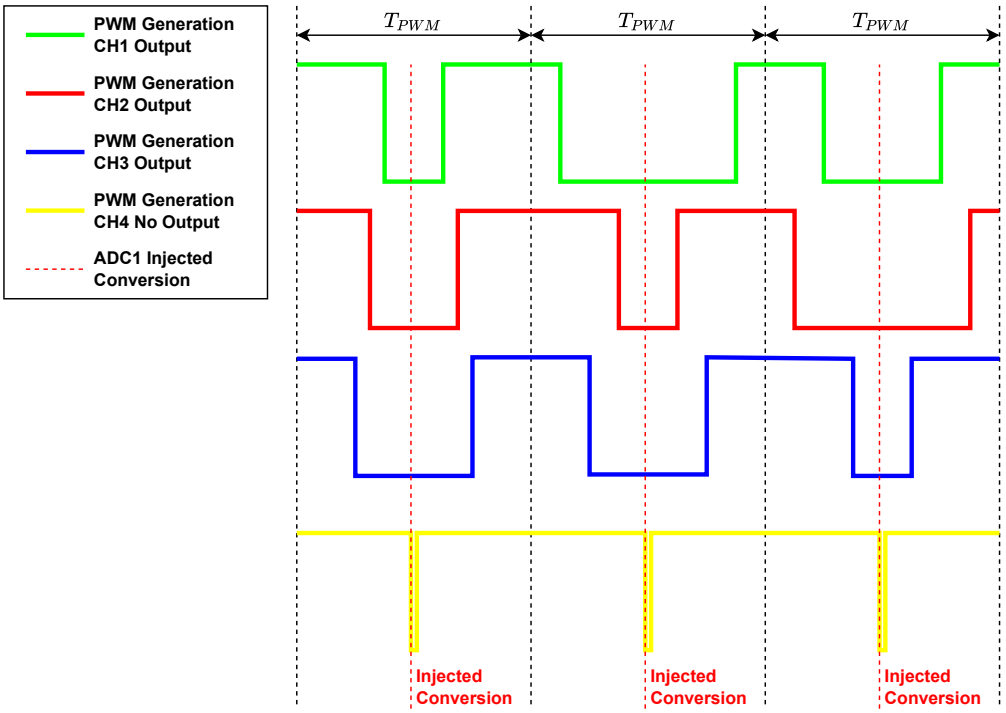


Figure 3.20. Injected Conversion Trigger.

### 3.2.6.5 Pin Assignments

Table 3.12 contains the pins of the STM32 NUCLEO-F446RET6 microcontroller assigned to function as the input and output pins. The microcontroller utilizes its digital output pins for PWM generation to control the three-phase VSI model, analog input pins

to perform injected ADC to the real-time measurements, and an analog output pin for triggering the scope. Figure 3.21 illustrates the assignment of the PWM outputs to the semiconductors of the three-phase VSI. With the assigned pins, the implemented control system is visualized in Figure 3.22 with every system component assigned to their respective hardware. The mathematical model of the implemented closed-loop system is illustrated in 3.23. The implementation of the experiment is documented in Figure 9.

Table 3.12. STM32 NUCLEO-F446RET6 Microcontroller Assigned Pins.

| **Inputs/Outputs** | **Assigned Pin** |
|---|---|
| Phase Current Input $i_a$ | PA0/A0 (ADC1 IN0) |
| Phase Current Input $i_b$ | PA1/A1 (ADC1 IN1) |
| Rotor Electrical Angle Input $\theta_e$ | PB0/A3 (ADC1 IN8) |
| Rotor Electrical Speed Input $\omega_e$ | PC1/A4 (ADC1 IN11) |
| Trigger Output | PA4/A2 (DAC OUT1) |
| PWM A High Side ($S_a$) | PC6 (TIM8 CH1) |
| PWM A Low Side ($S_a{}'$) | PA7/D11 (TIM8 CH1N) |
| PWM B High Side ($S_b$) | PC7/D9 (TIM8 CH2) |
| PWM B Low Side ($S_b{}'$) | PB14 (TIM8 CH2N) |
| PWM C High Side ($S_c$) | PC8 (TIM8 CH3) |
| PWM C Low Side ($S_c{}'$) | PB15 (TIM8 CH3N) |

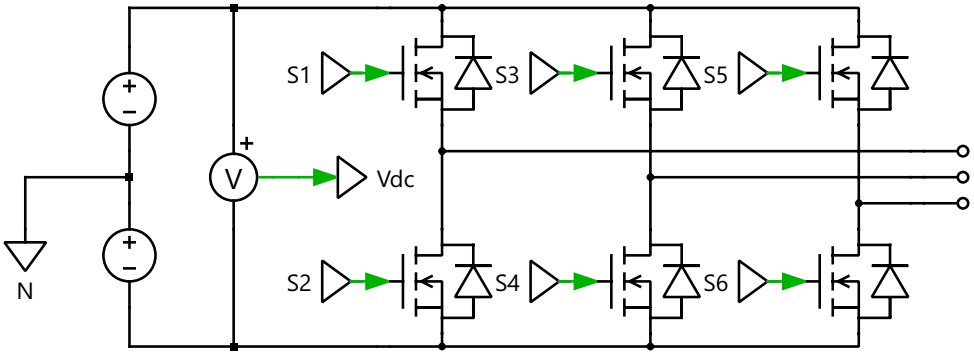


Figure 3.21. SVPWM Output Assignment to Three-Phase VSI.

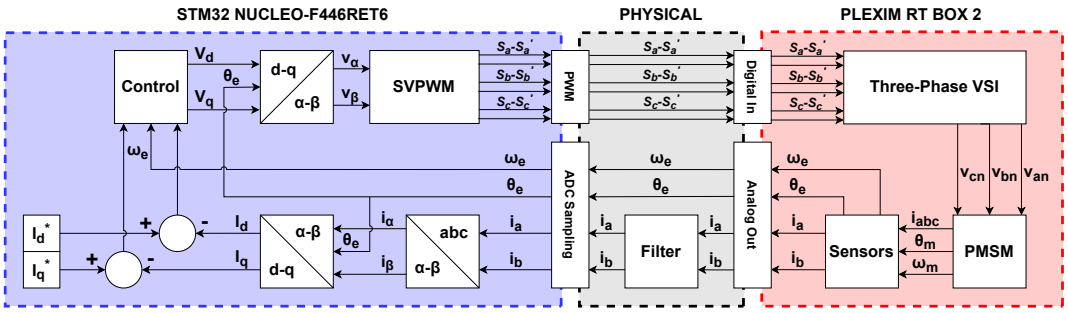


Figure 3.22. Implemented Closed-Loop System.

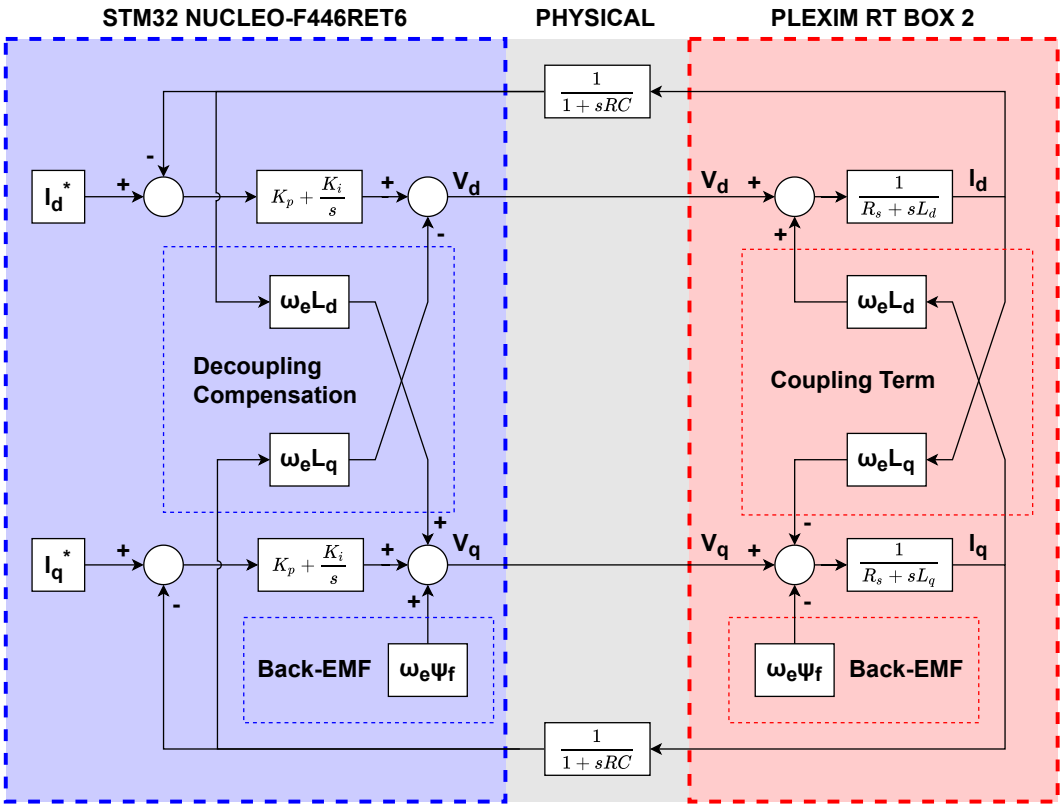


Figure 3.23. Mathematical Model of Closed-Loop System.

# CHAPTER IV
# RESULTS AND DISCUSSIONS

## 4.1 Preliminary System Test

In this test, the control loop must be able to regulate the d-q axis current according to the varied setpoints. The q-axis current setpoint tested will be gradually increased until it reaches the specified peak current rating of the PMSM. Table 4.1 contains the results of the test. The q-axis current shows an average accuracy of over $99.9\%$ between the set reference and the output, thereby indicating a properly controlled system.

Table 4.1. Preliminary Test Results.

| $I_d^*$ **(A)** | $I_q^*$ **(A)** | $I_d$ **(A)** | $I_q$ **(A)** |
|---|---|---|---|
| **0** | **4** | -0.001 | 3.997 |
| **0** | **4.5** | -0.006 | 4.483 |
| **0** | **5** | -0.006 | 4.995 |
| **0** | **5.5** | -0.003 | 5.497 |
| **0** | **6** | -0.005 | 5.982 |
| **0** | **6.5** | -0.004 | 6.497 |
| **0** | **7** | -0.004 | 6.997 |
| **0** | **7.5** | -0.004 | 7.481 |
| **0** | **7.8** | 0.004 | 7.782 |
| **0** | **28.1** | 0.003 | 28.089 |

## 4.2 LPF Test

The RC LPF is constructed using a blank PCB with hand-soldered passive components. The capacitor used is a multi-layer ceramic capacitor (MLCC) with a static capacitance value, ensuring an exact $47\mathrm{nF}$ capacitance. The resistance of the filter is realized using a multi-turn potentiometer to provide flexibility during testing. Using a potentiometer instead of a resistor allows flexibility in adjusting the resistance value. The LPF board is visible in Figure 4.1.

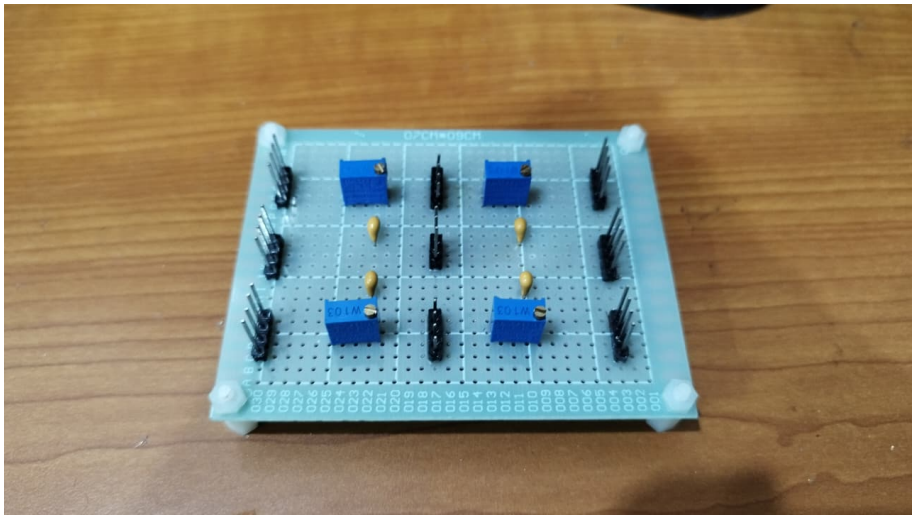

Figure 4.1. RC LPF PCB.

The test uses an Arbitrary Function Generator (AFG) to generate the sinusoidal signal as the input to be filtered by the LPF. The output of the filter is visualized and analyzed using an oscilloscope, with the input signal being a $3.3\ V_{pp}$ sinusoidal signal. The attenuation caused by the filter can be indicated by a visible decrease of amplitude within the output when compared to the input. Figure 4.2 shows the filter output when the input is a $25$ Hz sinusoidal signal, a frequency approximate to the previously determined maximum electrical frequency of $24.87$ Hz. The output exhibits little to no decrease in its amplitude and phase shift, hence enabling accurate current sampling for the system when operating at the determined maximum electrical frequency.

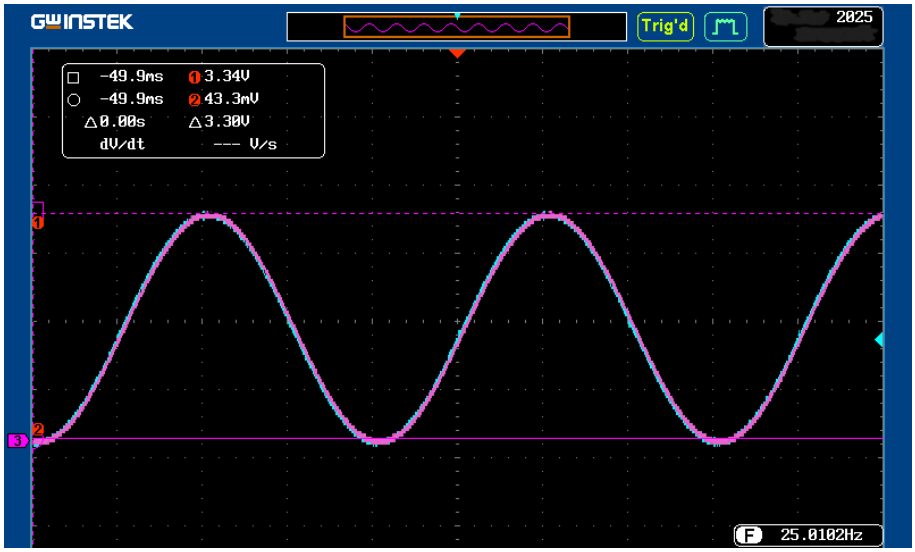


Figure 4.2. 25 Hz Filter Test Input (Blue) and Output (Purple).

Figure 4.3 showcases the filtering capability of the LPF by causing a visible amplitude decrease and phase shift at the frequency $800$ Hz, the determined cut-off fre-

quency of the LPF. The output exhibits a $-3.5218\ \mathrm{dB}$ attenuation, a value approximate to the theoretical attenuation of $-3\ \mathrm{dB}$. The phase shift experienced by the output is visibly approximate to the theoretical $-45°$ shift, thereby proving the accuracy of the set cut-off frequency of the constructed LPF board.

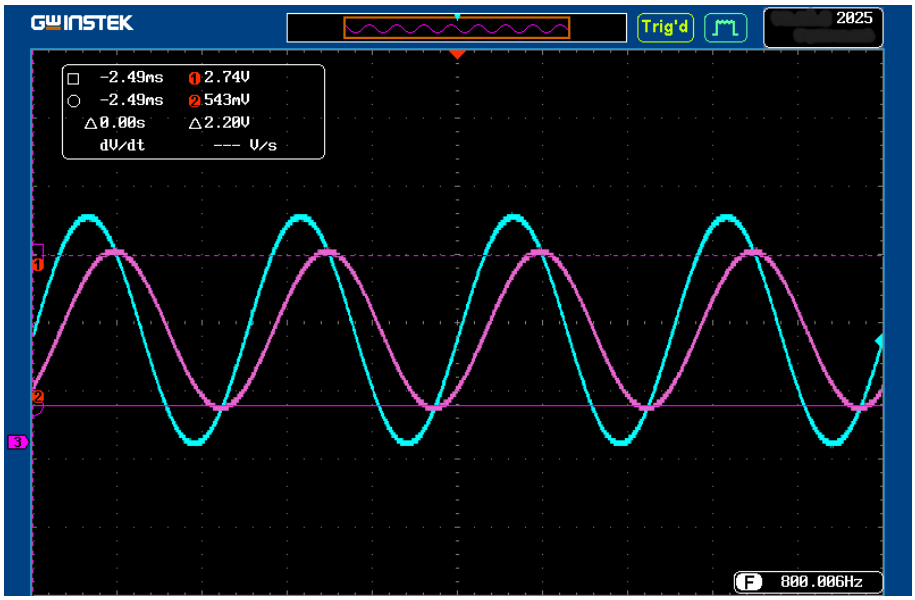


Figure 4.3. 800 Hz Filter Test Input (Blue) and Output (Purple).

The input signal had its frequency slowly raised to evaluate the decrease of the amplitude of the output signal as the frequency increases. Table 4.2 contains the results of the test, showcasing the different amplitudes of the output signal when filtered by the constructed LPF. The filter, being a first-order LPF, is theorized to have an attenuation rate of approximately $-20\ \mathrm{dB/dec}$. This trait of the filter can be seen when comparing the attenuation experienced by signals with a frequency difference of one decade, as highlighted by Table 4.3.

Table 4.2. Measured Frequency Response of the Constructed LPF.

| **Frequency (Hz)** | **Amp In (V)** | **Amp Out (V)** | **Out/In** | **Gain (dB)** |
|---|---|---|---|---|
| 25 | 3.3 | 3.3 | 1.0000 | 0.0000 |
| **800** | 3.3 | 2.2 | 0.6667 | **-3.5218** |
| 1600 | 3.3 | 1.4 | 0.4242 | -7.448 |
| 2400 | 3.3 | 0.983 | 0.2979 | -10.519 |
| 3200 | 3.3 | 0.786 | 0.2382 | -12.462 |
| 4000 | 3.3 | 0.636 | 0.1927 | -14.301 |
| 8000 | 3.3 | 0.333 | 0.1009 | -19.921 |
| 16000 | 3.3 | 0.162 | 0.0491 | -26.180 |
| 24000 | 3.3 | 0.105 | 0.0318 | -29.947 |
| 32000 | 3.3 | 0.0791 | 0.0240 | -32.407 |
| 40000 | 3.3 | 0.0616 | 0.0187 | -34.579 |
| 80000 | 3.3 | 0.0291 | 0.0088 | -41.092 |

According to the test results, the constructed LPF has an effective attenuation rate of $-19.325\ \mathrm{dB/dec}$, a quantity approximate to the theorized $-20\ \mathrm{dB/dec}$. The slope of the attenuation rate is plotted in Figure 4.4 where it is compared with the theoretical frequency response of the first-order LPF via its Bode plot. This indicates that the LPF is able to filter the high-frequency noise accordingly to ensure proper current sensing for the feedback control when integrated into the control loop.

Table 4.3. Measured Attenuation Rate of the Constructed LPF.

| **Frequency Range** ($Hz$) | **Slope** ($dB/dec$) |
|---|---|
| 800-8000 | -16.400 |
| 1600-16000 | -18.732 |
| 2400-24000 | -19.427 |
| 3200-32000 | -19.945 |
| 4000-40000 | -20.278 |
| 8000-80000 | -21.171 |
| **Average** | **-19.325** |

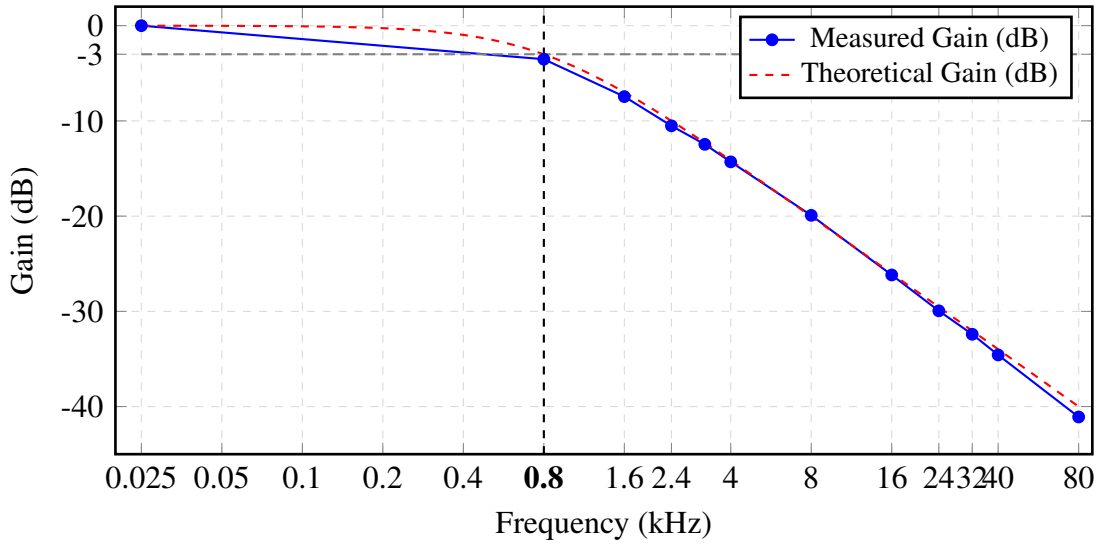


Figure 4.4. LPF Theoretical and Experimental Frequency Response Comparison.

## 4.3 Microcontroller Synchronization

### 4.3.1 Symmetrical PWM

As previously mentioned, the six PWM signals generated by the microcontroller are set to be symmetrical to support the implementation of symmetrical SVPWM. The outputs had their duty cycles set to $0.25$, $0.5$, and $0.75$ for Phase A, Phase B, and Phase C in that order. Figure 4.5 shows the PWM signals generated by the microcontroller captured by the HIL platform via PWM capture. It can be seen that they are symmetrical to each other and their pattern occurs every $12.5\ \mu$s, indicating an $8$ kHz output frequency.

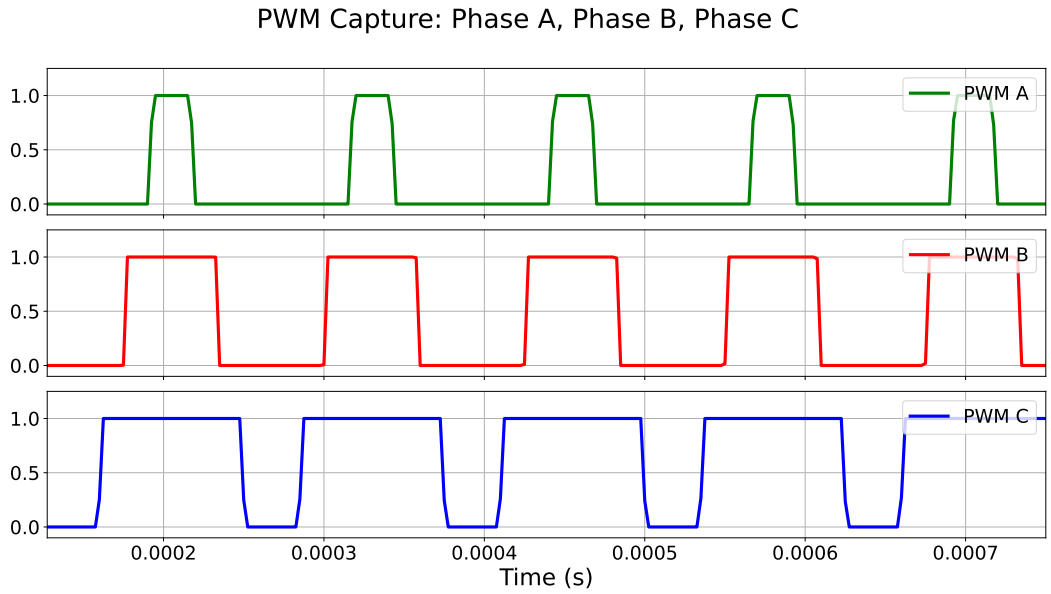


Figure 4.5. Symmetrical PWM Outputs.

### 4.3.2 Triggered Sampling and Control Loop Period

The injected ADC is set to sample the current measurements when the PWM outputs are generating the zero vector $V_0$. Figure 4.6 shows the PWM outputs being generated while the control loop is taking place. It can be seen that the control loop always begins at the very middle when all three PWM phases are low, indicating that the sampling of the current measurements always occur when the PWM outputs are generating the zero vector $V_0$. Additionally, the microcontroller is shown to take approximately $55$ us to perform all of the sequences of the control loop from feedback measurement sampling to PWM generation, indicating a computation period faster than one PWM period of $125$ $\mu$s.

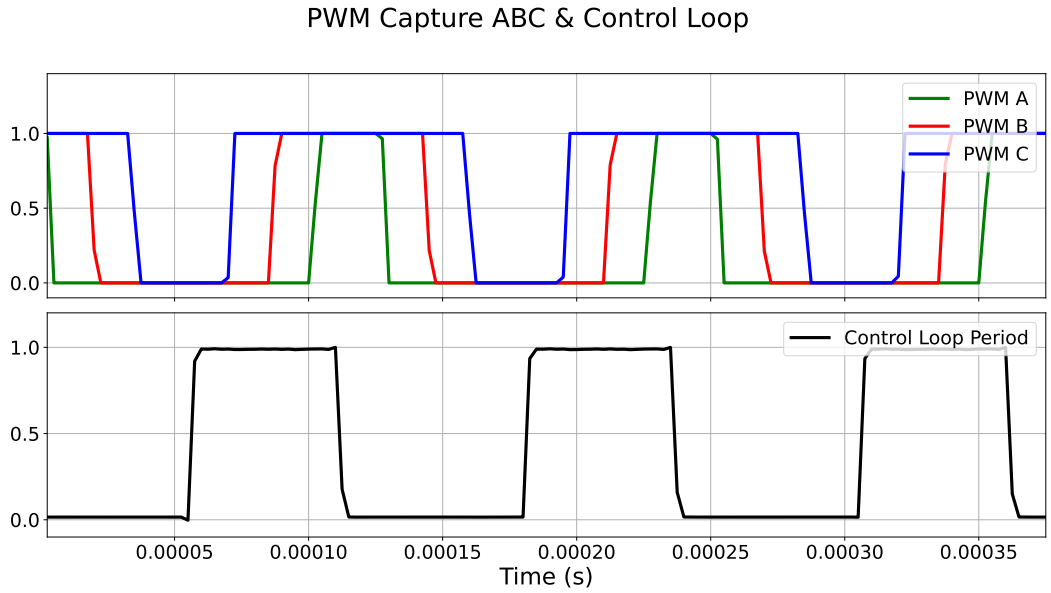


Figure 4.6. Control Loop Timing to PWM Outputs.

## 4.4 SVPWM Test

To ensure a proper vector output during the real-time HIL test, the SVPWM must be able to generate the six PWM signals containing the correct information to drive the three-phase VSI. The test will evaluate the dead time delay implementation captured by the HIL platform, vector output magnitude, vector output phase, and evaluate the overall waveform generated by the three-phase VSI. The test uses a HIL simulation representing a three-phase VSI with a $400$ V DC voltage source connected to a balanced three-phase load comprised of a $10$ $\Omega$ resistor and $1$ mH inductor in each phase to output a $0.1$ ms time constant, visualized in Figure 4.7.

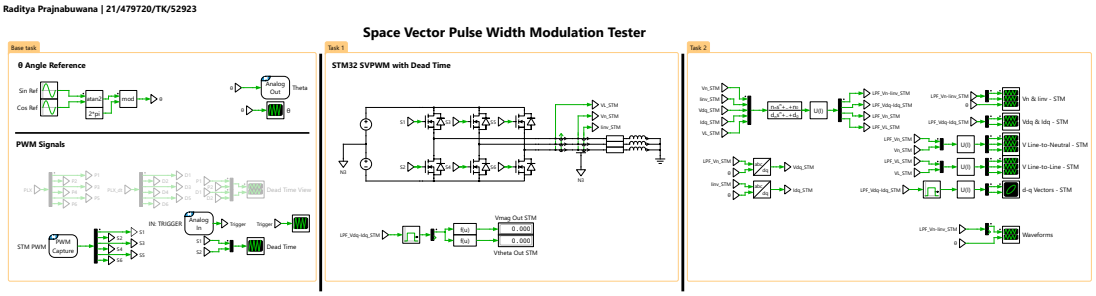


Figure 4.7. SVPWM Test HIL Simulation.

### 4.4.1 Dead Time Delay

Figure 4.8 showcases the ability of the HIL platform to detect the dead time delay implemented into the complementary PWM signals. The digital input PWM capture block is configured to operate using the sub-cycle average mode to ensure proper detection of the $5\ \mu s$ dead time gap between the complementary PWM signals. The three-phase VSI driven by these signals captured have been proven to not experience any shoot-through scenario-induced errors, thereby proving the ability of the HIL platform to sample the dead time delay.

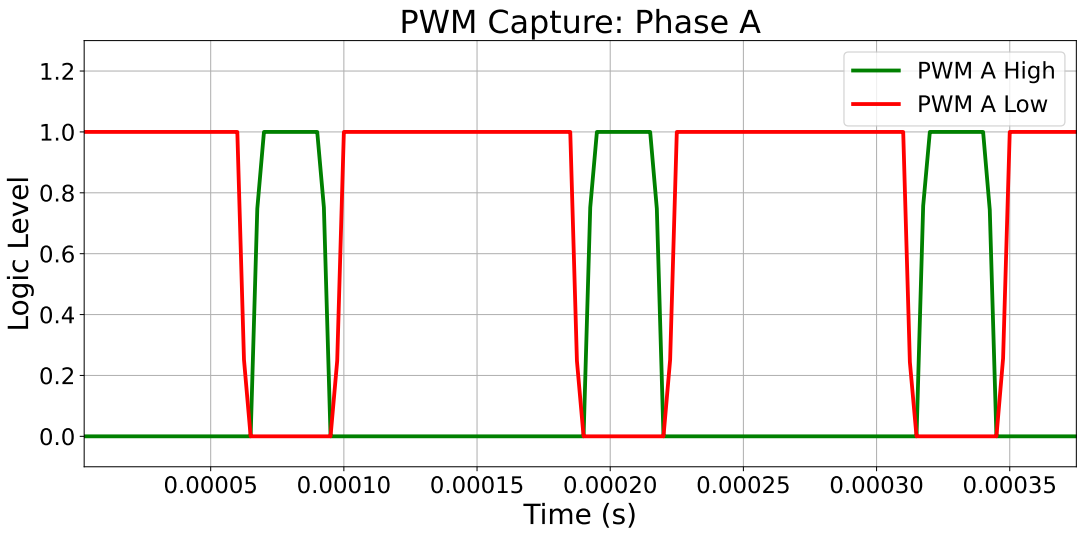


Figure 4.8. Dead Time Delay in HIL PWM Capture.

### 4.4.2 Voltage Waveform

The PWM signals generated by the microcontroller are filtered to evaluate their voltage waveforms. Figure 4.9 visualizes the raw PWM line-to-line along with the line-to-line voltage generated by the three-phase VSI when operated using the PWM signals captured. It is visible that the line-to-neutral voltage waveform resembles the desired SVPWM output waveform, thereby ensuring that the generated PWM signals represent the desired voltage waveform of the SVPWM.

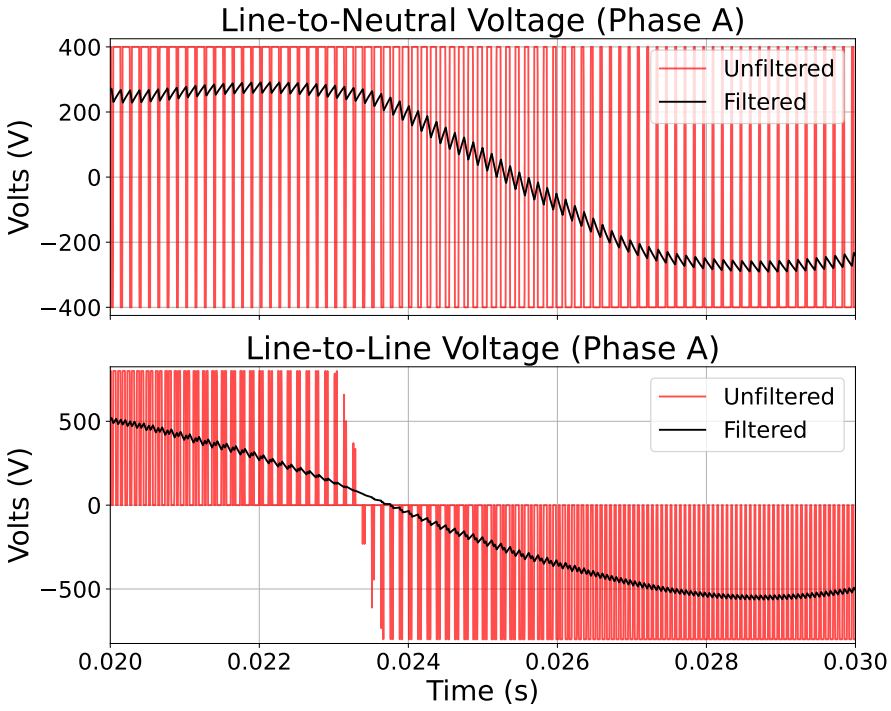


Figure 4.9. Unfiltered and Filtered Voltage Waveform.

### 4.4.3 Vector Magnitude

This simulation uses a constant frequency of $50$ Hz to make visual validation more feasible by ensuring that any phase shift occurring between the generated waveforms is caused solely by the angle of the desired d-q axis vector output. Figure 4.10 shows the three-phase voltage outputs of the VSI when the magnitude of the requested d-q axis vector output is tested at $100$ V, $200$ V, and $300$ V with a constant angle of $0°$. The resulting output waveforms are symmetrical to each other but have varying amplitudes corresponding to the magnitude of the requested voltage vector when generated using the same electrical angle, thereby proving the ability of the SVPWM algorithm to generate the vector magnitude.

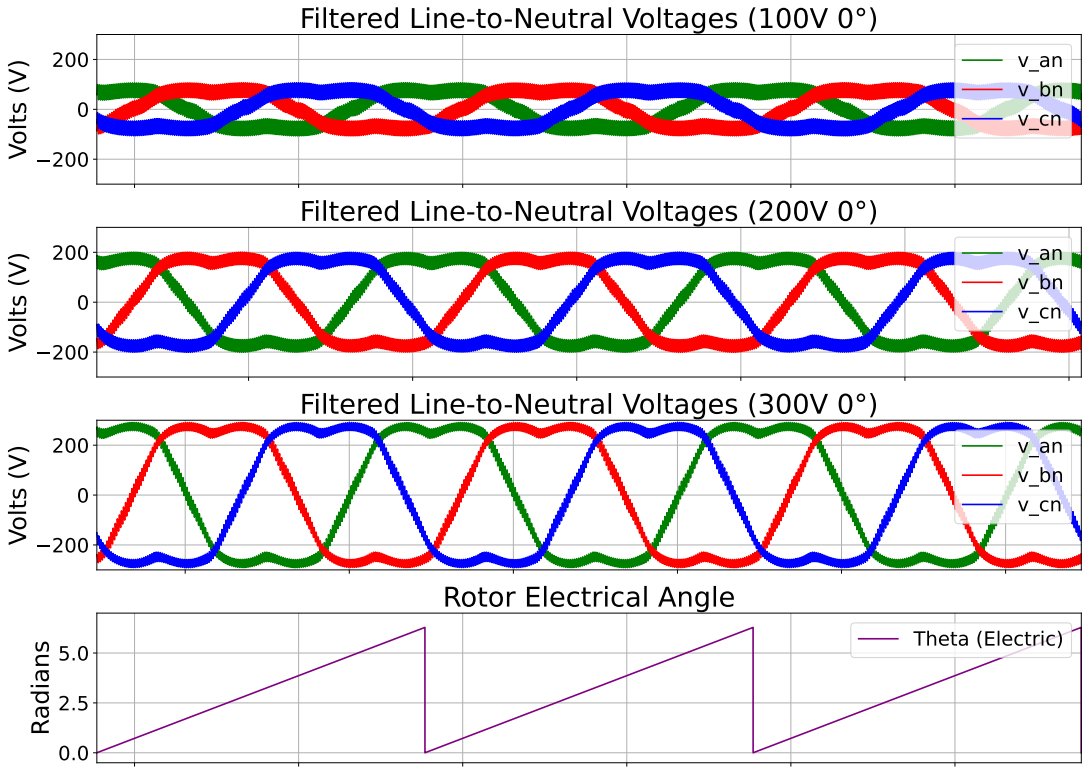


Figure 4.10. Output Vector Magnitude Varied.

### 4.4.4 Vector Angle

Furthermore, Figure 4.11 shows the three-phase voltage outputs of the VSI when the angle of the requested d-q axis vector is tested at $0°$, $60°$, and $120°$ with a constant magnitude of $300\ \mathrm{V}$. The outputs all have the same magnitude of approximately $300\ \mathrm{V}$ but are not horizontally-symmetrical. The voltages of the first test are visually symmetrical with that of the third test, but each phase is symmetrical to the phase adjacent to it rather than the same phase, hence proving a $120°$ phase shift. Therefore, it is proven that the constructed SVPWM algorithm is able to accurately generate the desired d-q axis vector in terms of magnitude and angle.

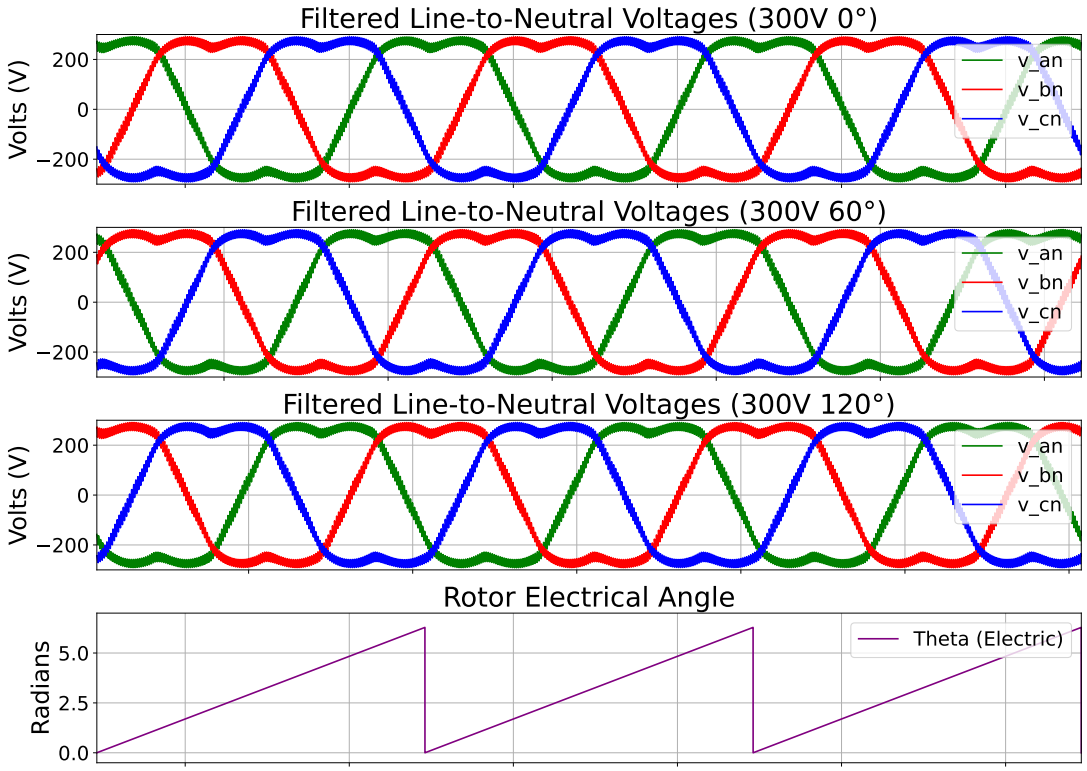


Figure 4.11. Output Vector Angle Varied.

## 4.5 FOC Test

### 4.5.1 Open-Loop Test

For this test, proper functionality of the HIL simulation and validity of the system equations are to be evaluated before closed-loop test is performed. The d-axis current set-point is set to $0$ A while that of the q-axis current is set to $28.1$ A. Figure 4.12 showcases the transient-state outputs of the system when operated in the open-loop configuration without the assistance of the PI controllers. The q-axis current, mechanical speed, and generated torque exhibit long rising periods. The steady-state outputs exhibit accuracy, except for the d-axis current which tends to stabilize below the setpoint.

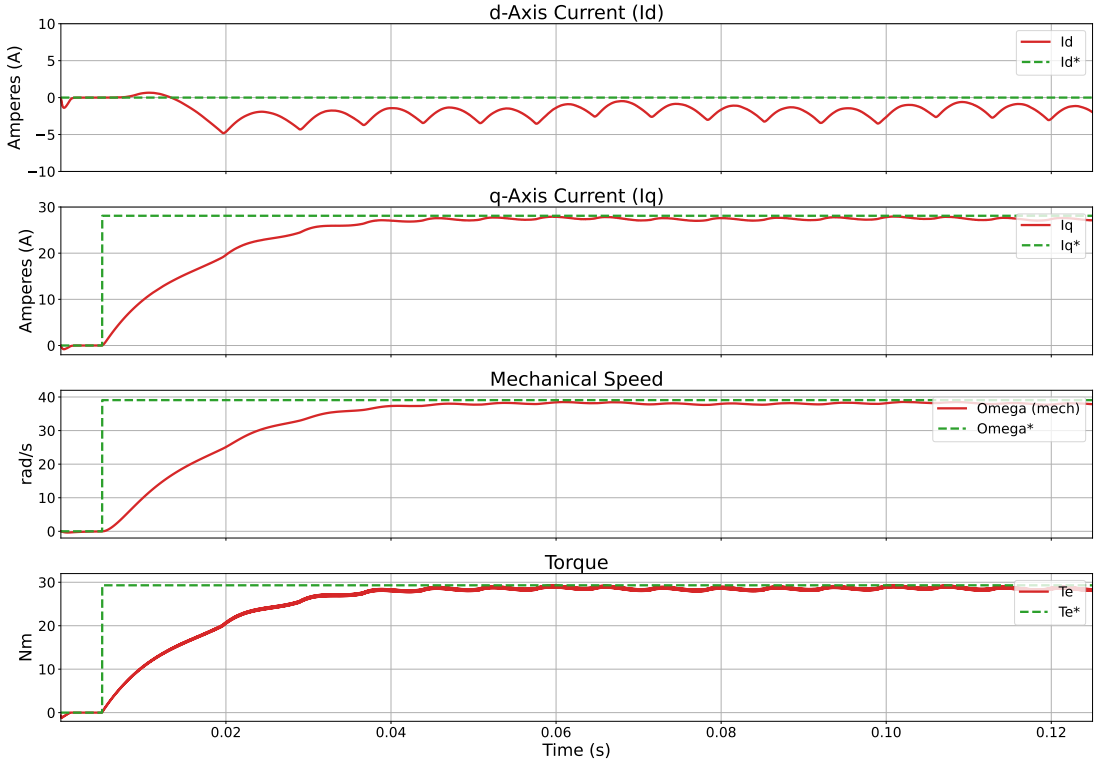


Figure 4.12. Open-Loop Transient-State to Steady-State Output.

Figure 4.13 visualizes the output waveforms of the open-loop system. The filtered line-to-neutral three-phase voltages exhibit the desired SVPWM waveform, indicating successful SVPWM implementation. The three-phase current output shows good alignment with the back-EMF of the PMSM, indicating that the phase current is comprised of mostly the q-axis current component. Overall, the implemented open-loop control scheme exhibited proper functionality of the HIL simulation and control scheme.

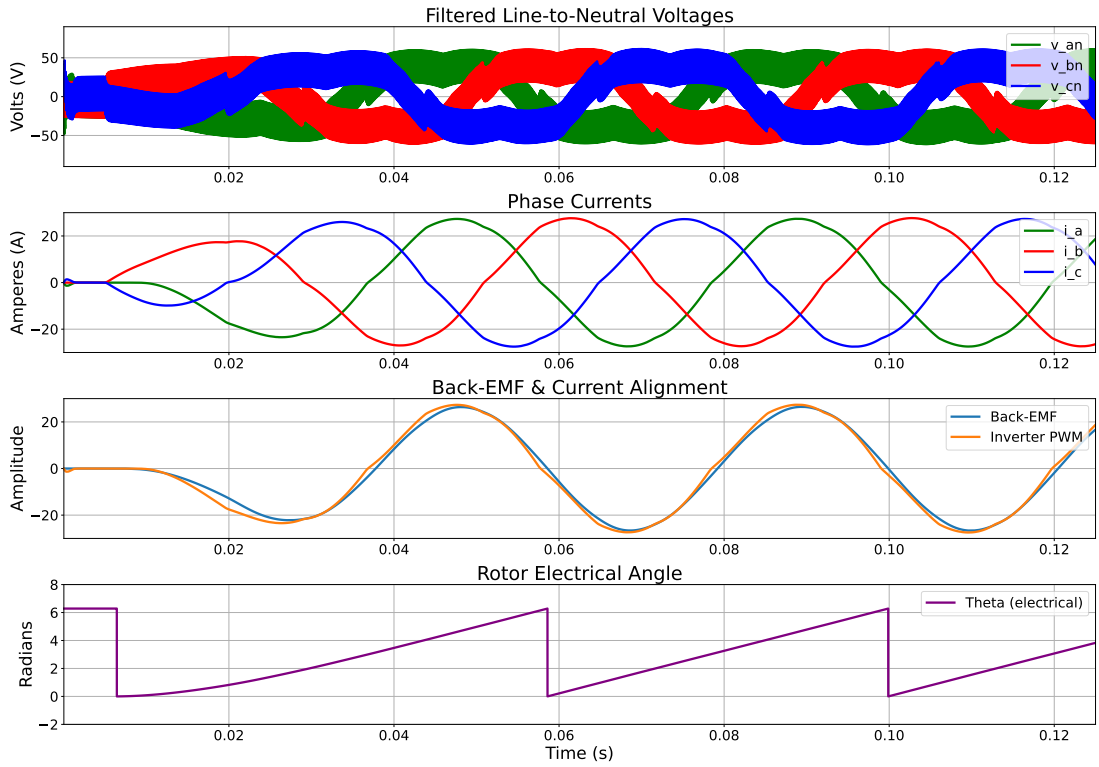


Figure 4.13. Open-Loop Output Waveforms.

### 4.5.2 Closed-Loop Test

#### 4.5.2.1 Step Response and Three-Phase Waveforms

During this test, the system is given a setpoint in the form of a step input at $7.8$ A and $28.1$ A, both being the rated continuous and peak current of the corresponding PMSM. Figures 4.14 and 4.15 both show the step response of the d-axis current $I_d$, q-axis current $I_q$, rotor mechanical speed $\omega_m$, and generated torque $T_e$ at both current setpoints. Both figures show that the system constantly exhibits its ability to regulate the currents, as seen in the $I_d$ outputs being properly kept at $0$ A while $I_q$ experiences a rise in magnitude. The $I_q$ output in both figures exhibits a rise time faster than $30$ ms with no overshoot and good accuracy. This causes the $T_e$ to be responsive and controlled, resulting in an equally fast-responding and controlled $\omega_m$. Additionally, the outputs all show good accuracy when the system has reached steady state.

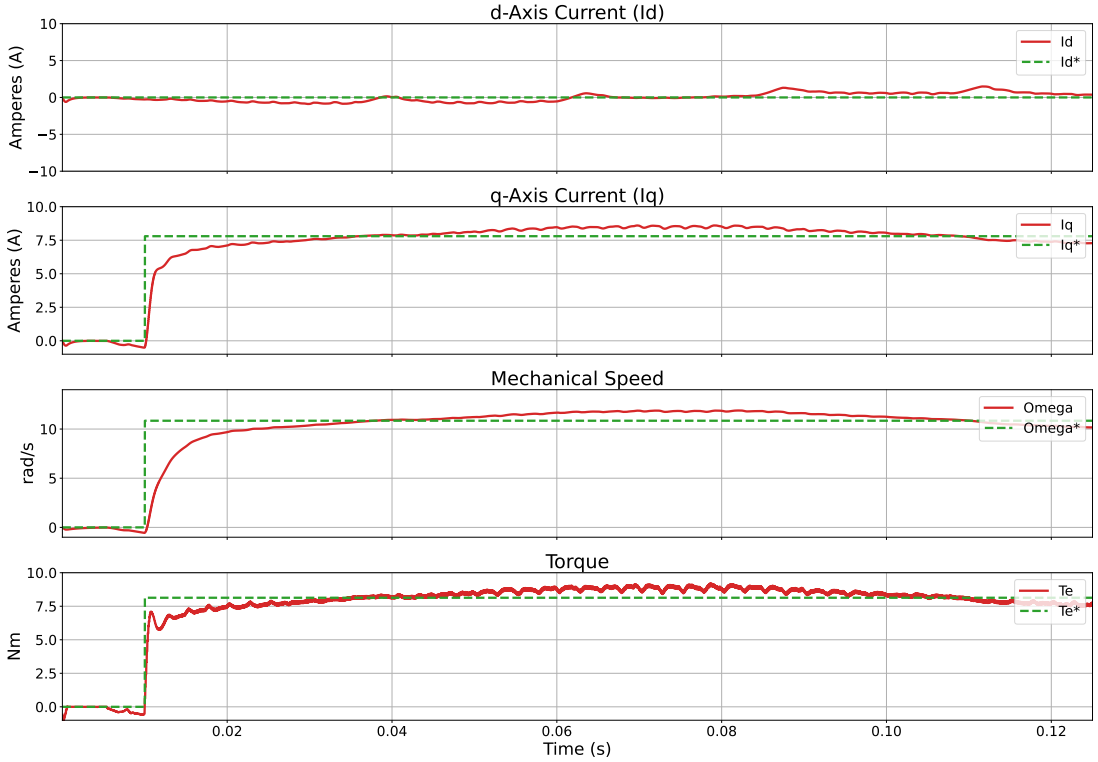


Figure 4.14. Closed-Loop Step Response at Continuous Current 7.8 A.

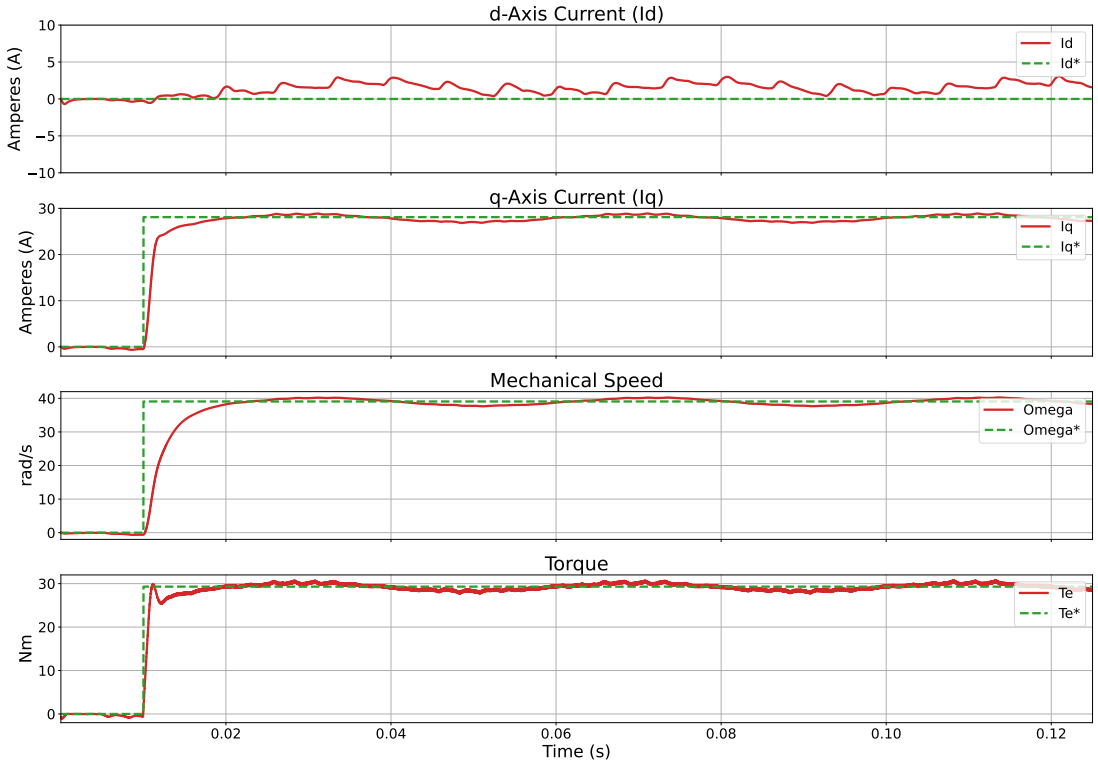


Figure 4.15. Closed-Loop Step Response at Peak Current 28.1 A.

Figure 4.16 and 4.17 are the three-phase waveform outputs of the system when tested at both current setpoints. The filtered line-to-neutral three-phase voltage output

shows the desired SVPWM waveform and each have different frequencies according to the resulted speed of the PMSM rotor. The three-phase current shows an amplitude accurate to that of the requested vector magnitude corresponding to the set current setpoint. Additionally, the three-phase current output shows symmetry when compared to the line-to-neutral back-EMF of the PMSM during the test, indicating that the three-phase current output in both occasions is comprised of mainly reactive q-axis current.

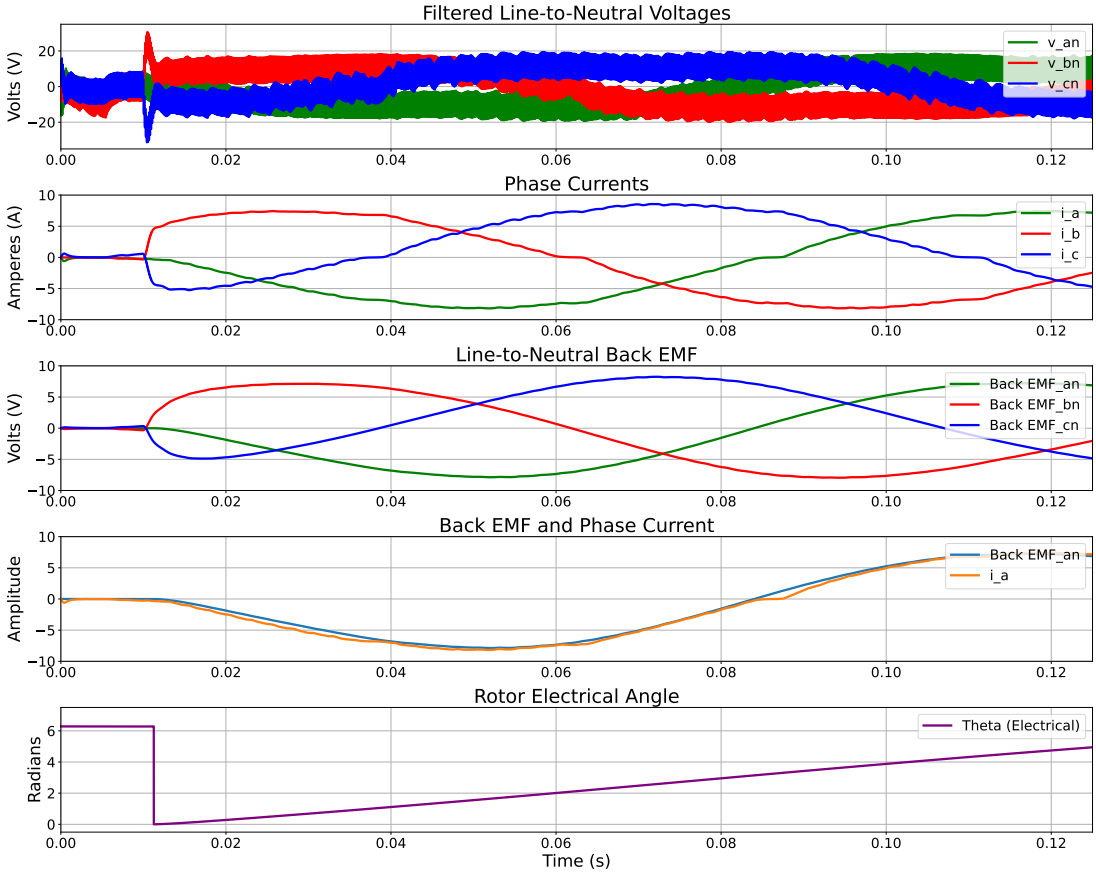


Figure 4.16. Voltage and Current Waveforms at Continuous Current 7.8 A.

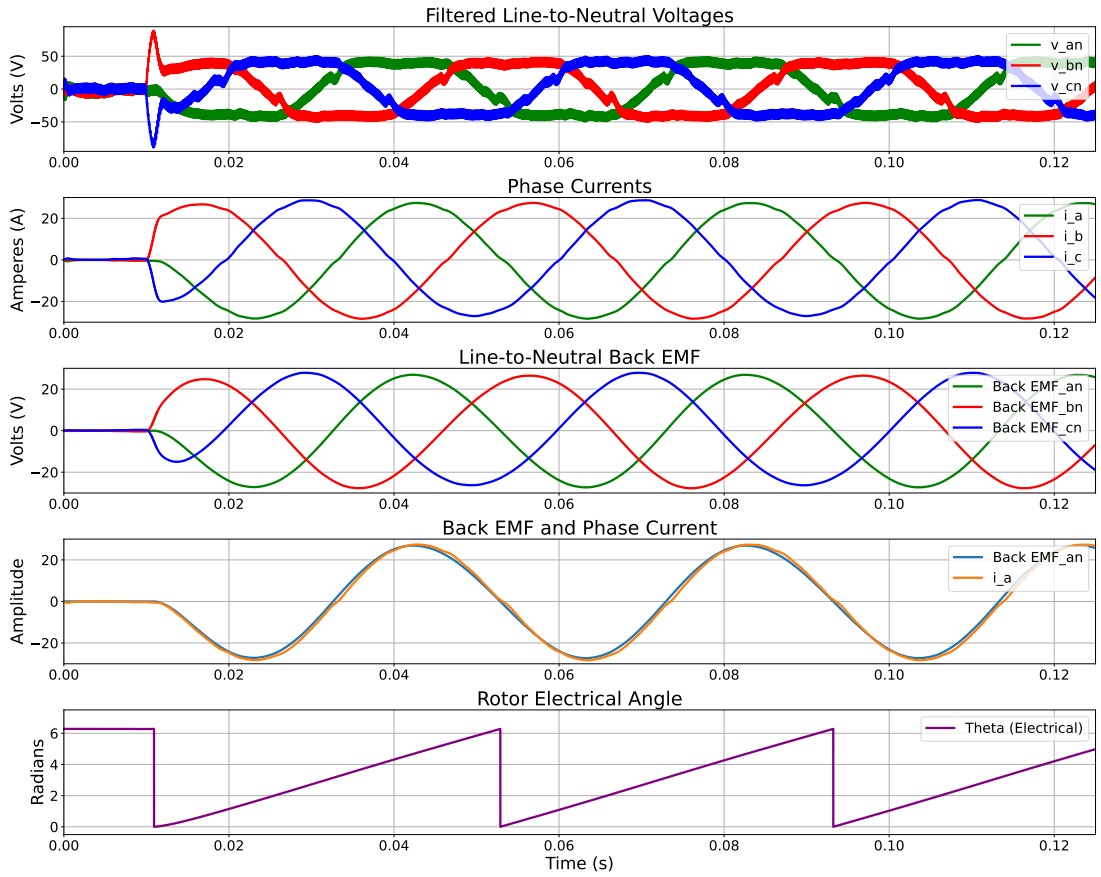


Figure 4.17. Voltage and Current Waveforms at Peak Current $28.1$ A.

#### 4.5.2.2 Dynamic Response

To evaluate the dynamic responsiveness of the embedded control system, an additional test is performed where the current setpoint is periodically changed. The q-axis current $I_q$ setpoint is tested at $7.8$ A, $15.6$ A, $3.9$ A, and $28.1$ A in that order. The setpoint is changed every $3.75$ ms to give the control loop $300$ iterations before changing the setpoint.

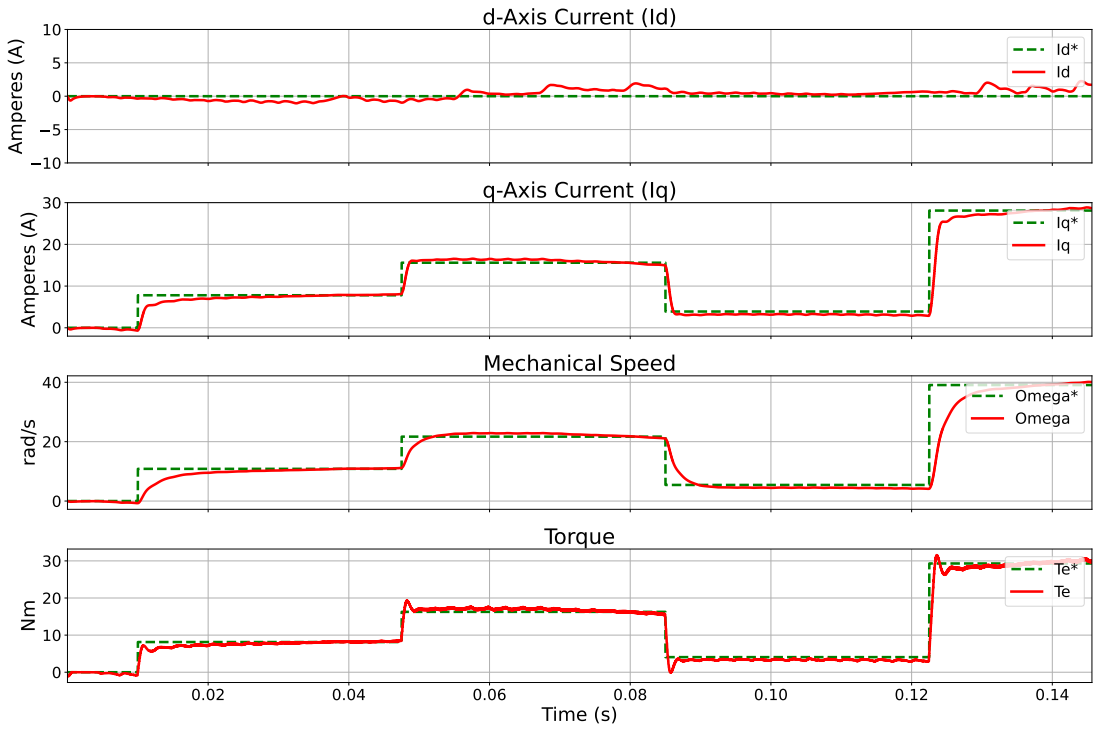


Figure 4.18. Closed-Loop Response Under Dynamic Setpoint.

Figure 4.18 shows the d-axis current $I_d$, q-axis current $I_q$, rotor mechanical speed $\omega_m$, and generated torque $T_e$ during the test. The control system exhibits its ability to quickly adapt the output of the system according to the set reference values, as can be seen in its fast-responding $I_q$ resulting in an equally fast $T_e$ response which in turn results in the $\omega_m$ being quick to adapt as well. Additionally, the $I_d$ output is shown to be consistently regulated at $0$ A even though the $I_q$ fluctuates, indicating proper coupling term compensation via feedforward control.

#### 4.5.2.3 Steady-State Output Accuracy and Linearity

The performance of the embedded control system is tested further to ensure accurate steady-state output. The test is done by first setting a constant reference for the current to evaluate the output when the system has reached steady-state condition. Table 4.4 contains the results, where it can be seen that the system is able to exhibit a high level of accuracy as indicated by the most significant $I_q$ error magnitude being $4.538\%$. The output values of this test are plotted in Figure 4.19 to visualize the linearity of the output values compared to the reference, further proving the accuracy of the control.

Table 4.4. Steady-State $I_d$ and $I_q$ Output.

| $I_d^*$**(A)** | $I_q^*$**(A)** | $I_d$(A) | $I_q$(A) | $I_q$(%) |
|---|---|---|---|---|
| **0** | **4** | 0.380 | 4.182 | 104.538 |
| **0** | **4.5** | 0.348 | 4.467 | 99.262 |
| **0** | **5** | 0.336 | 5.029 | 100.571 |
| **0** | **5.5** | 0.215 | 5.588 | 101.591 |
| **0** | **6** | 0.137 | 6.008 | 100.136 |
| **0** | **6.5** | 0.081 | 6.538 | 100.591 |
| **0** | **7** | 0.077 | 6.968 | 99.542 |
| **0** | **7.5** | 0.028 | 7.449 | 99.316 |
| **0** | **7.8** | 0.008 | 7.756 | 99.432 |
| **0** | **28.1** | 0.804 | 28.029 | 99.747 |

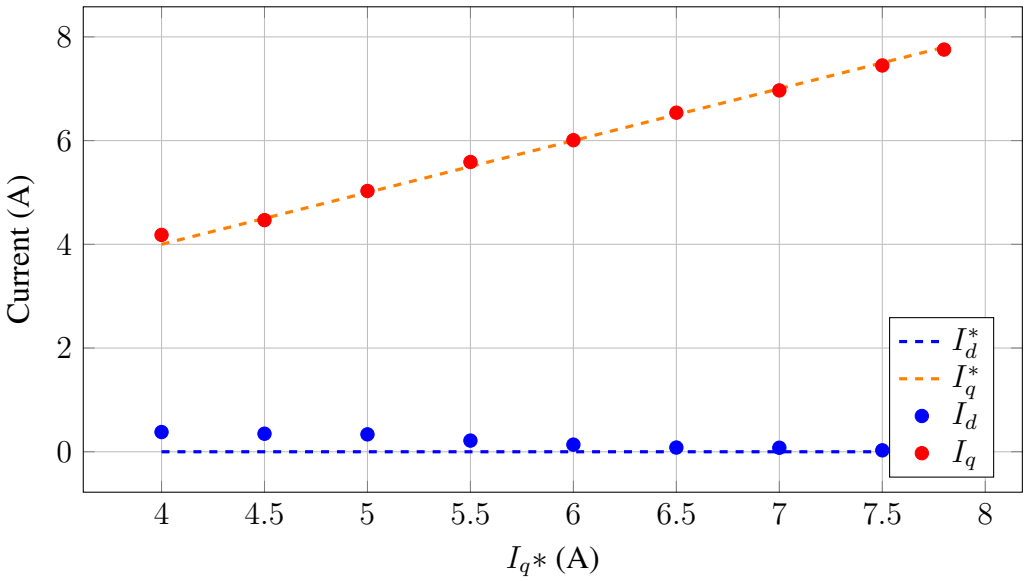


Figure 4.19. Plotted $I_d$ and $I_q$ Output.

As previously mentioned, the $I_q$ output has a linear relationship with the generated torque of an SPMSM according to Equation 2-10 which in turn has a linear relationship with the steady-state mechanical speed of the rotor according to Equation 3-2. Therefore, the steady-state mechanical speed of the rotor can be calculated to further test the accuracy of the control system. Table 4.5 shows the calculated torque reference and mechanical speed reference when the $I_q$ setpoint is varied. The results in the same table show that the steady-state rotational speed outputs yield a constantly high level of accuracy to the calculated reference value, with the biggest error magnitude being $4.92\%$. Therefore, the

system is capable of achieving a steady-state accuracy of over $95\%$. Figure 4.20 showcases the plotted results to further visualize the linearity of both the generated torque and steady-state mechanical speed compared to their reference values.

Table 4.5. Steady-State $T_e$ and $\omega_m$ Output.

| $I_q^*$**(A)** | $T_e^*$**(Nm)** | $\omega_m^*$**(rad/s)** | $T_e$(Nm) | $\omega_m$(rad/s) | $\omega_m(\%)$ |
|---|---|---|---|---|---|
| **4** | **4.171** | **5.561** | 4.360 | 5.835 | 104.920 |
| **4.5** | **4.692** | **6.256** | 4.658 | 6.233 | 99.615 |
| **5** | **5.213** | **6.951** | 5.244 | 7.015 | 100.913 |
| **5.5** | **5.735** | **7.647** | 5.827 | 7.785 | 101.809 |
| **6** | **6.256** | **8.342** | 6.265 | 8.365 | 100.275 |
| **6.5** | **6.778** | **9.037** | 6.818 | 9.097 | 100.666 |
| **7** | **7.299** | **9.733** | 7.266 | 9.697 | 99.632 |
| **7.5** | **7.820** | **10.427** | 7.767 | 10.361 | 99.355 |
| **7.8** | **8.133** | **10.844** | 8.088 | 10.784 | 99.436 |
| **28.1** | **29.302** | **39.069** | 29.227 | 39.077 | 100.020 |

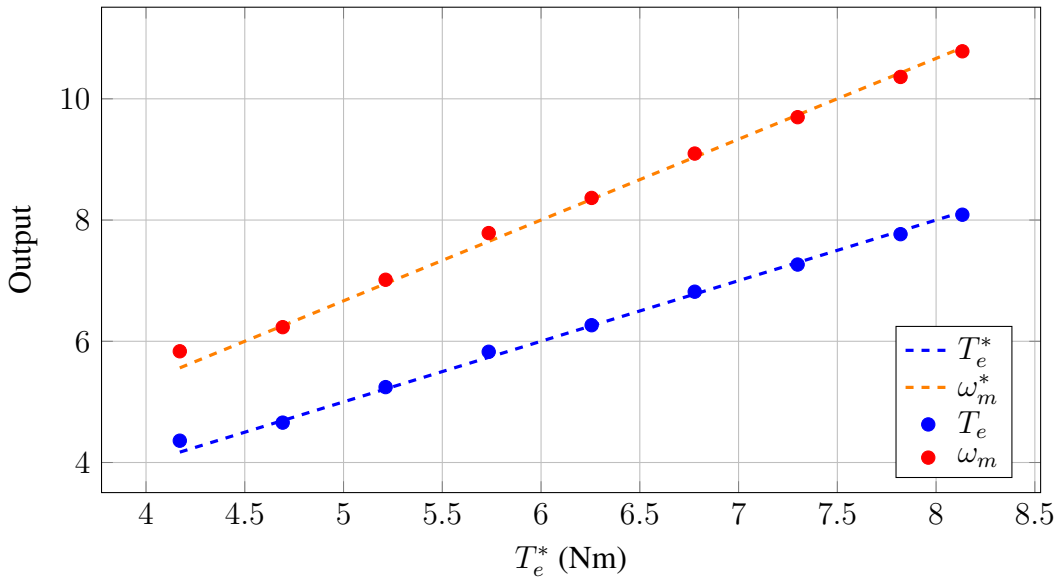


Figure 4.20. Plotted $T_e$ and $\omega_m$ Output.

# CHAPTER V
# CONCLUSION AND FUTURE WORKS

## 5.1 Conclusion

It is concluded that this work has successfully implemented FOC on a PMSM via real-time HIL testing. The utilization of a Plexim RT Box 2 as a HIL simulation platform has proven to be reliable in representing approximate real-time testing conditions. The modelled system components, such as the three-phase VSI and the PMSM, supported the performance evaluation of the embedded control system.

The embedded FOC algorithm implemented via an STM32 NUCLEO-F446RET6 microcontroller has managed to perform accurate current measurement sampling via triggered injected ADC for feedback control. The utilization of an RC LPF enables high-frequency noise filtering, ensuring precision for the ADC sampling of the microcontroller throughout testing. Moreover, implementation of the pole-zero cancellation for controller tuning resulted in a closed-loop system with a proper bandwidth restriction.

Furthermore, the PWM outputs of the microcontroller have managed to generate accurate vector magnitude and angle due to the valid SVPWM algorithm design. The implementation of a dead time delay ensures safety during three-phase VSI operation by preventing unwanted shoot-through scenarios. Proper control of the modelled three-phase VSI is achieved, as proven by accurate three-phase voltage amplitude and phase shift generated by the VSI.

Furthermore, the control system is proven to be capable of regulating the output of the system. The system exhibits a fast response to step inputs and is able to adjust itself accordingly when faced with a dynamic load. During steady-state conditions, the system has also been proven to be able to sufficiently maintain the accuracy of its output.

## 5.2 Future Works

1. **Speed Control**

   Implementing speed control adds another layer of control, enabling speed as an input option to the embedded control system.

2. **RTOS Implementation**

   Divide the algorithm into tasks via RTOS to increasing system responsiveness.

3. **Field Weakening with MTPA for IPMSMs**

   Adding field weakening with maximum torque per ampere (MTPA) for IPMSM control allows for performance optimization in high-speed applications.

# REFERENCES


[1] S.-H. Kim, *Electric Motor Control: DC, AC, and BLDC Motors*, 1st ed. Cambridge, MA: Elsevier Academic Press, 2017. [Online]. Available: https://doi.org/10.1016/C2016-0-04207-3

[2] M. Savian, *STM32 Field-Oriented Control (FOC) Library and Firmware for Surface Permanent Magnet Synchronous Motors (SPMSM)*, STMicroelectronics, 2022, application Note AN5327. [Online]. Available: https://www.st.com/resource/en/application_note/an5327-stm32-foC-firmware-library-stmicroelectronics.pdf

[3] J. Schönberger, "Space vector control of a three-phase rectifier using plecs®," Plexim GmbH, Technoparkstrasse 1, 8005 Zürich, Switzerland, Tech. Rep., 2013, application Example Version 04-13. [Online]. Available: https://www.plexim.com/sites/default/files/plecs_space_vector_control.pdf

[4] A. News, "President seeks to expedite net-zero plan through ev push," https://en.antaranews.com/news/343973/president-seeks-to-expedite-net-zero-plan-through-ev-push, Feb 2025, accessed: 2025-10-30.

[5] I. E. A. (IEA), "An energy sector roadmap to net zero emissions in indonesia," 2022. [Online]. Available: https://www.oecd.org/en/publications/an-energy-sector-roadmap-to-net-zero-emissions-in-indonesia_4a9e9439-en.html

[6] L. A. Cahyoputra, "Looking at the role of industry and transportation in the nze target," *PwC Indonesia Infrastructure News Service*, Mar 2025. [Online]. Available: https://www.pwc.com/id/en/media-centre/infrastructure-news/march-2025/looking-at-the-role-of-industry-and-transportation-in-the-nze-target.html

[7] M. Khan, "Modeling and simulation of pmsm drives with d-q axis representation," *International Journal of Electrical and Electronics Research*, vol. 12, no. 2, pp. 45–52, 2024.

[8] K. Amin, M. A. Rahman, and M. Hossain, "Mathematical modelling of pmsm drive system with inverter integration," *IEEE Transactions on Energy Conversion*, vol. 31, no. 4, pp. 1202–1210, 2016.

[9] R. Krishnan, *Electric Motor Drives: Modeling, Analysis, and Control*. New Jersey: Prentice Hall, 2017.

[10] P. Singh and M. Kaur, "Comparison of foc and scalar control for pmsms," *International Journal of Advances in Engineering and Technology*, vol. 6, no. 4, pp. 1689–1695, 2013.

[11] N. N. As Shiddqi, "Perancangan pengendali motor dc armature dua kuadran berbasis stm32," Yogyakarta, Indonesia, 2024.

[12] Y. P. Satria, "Pengendali motor bldc berkinerja tinggi dengan metode foc untuk tim mobil balap arjuna ugm," Yogyakarta, Indonesia, 2019.

[13] M. A. Handoko, “Desain software pengendali motor pmsm,” Yogyakarta, Indonesia, 2022.

[14] P. GmbH, “Embedded code generation: Sensorless foc of pmsm on stm32 mcu,” https://www.plexim.com, 2023, demonstration model showcasing real-time simulation on RT Box 2 for STM32-based motor control validation.

[15] J. A. Pecas Lopes, N. Martins, and F. Silva, “Hardware-in-the-loop validation for power electronic systems,” in *Proceedings of the IEEE International Conference on Power Electronics*, 2021, pp. 112–118.

[16] Y. Chen, W. Zhang, and J. Liu, “Hil-based inverter control verification for pmsm drives,” *IEEE Access*, vol. 10, pp. 123 456–123 465, 2022.

[17] A. Irmak, “Field-oriented control (foc) fpga implementation,” Master’s Thesis, Middle East Technical University, Ankara, Turkey, 2019. [Online]. Available: https://etd.lib.metu.edu.tr/upload/12623722/index.pdf

[18] A. K. Shah and H. R. Patel, “Implementation and analysis of different discrete pi controller algorithms on single board heater system,” *Journal of Electrical Engineering*, vol. 15, no. 1, pp. 262–266, 2015. [Online]. Available: https://www.researchgate.net/publication/274660655_Implementation_and_analysis_of_different_discrete_PI_controller_algorithms_on_single_board_heater_system

[19] K. C. Odo, S. V. Egoigwe, and C. U. Ogbuka, “A model-based pi controller tuning and design for field oriented current control of permanent magnet synchronous motor,” *IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)*, vol. 14, no. 4, pp. 35–41, 2019. [Online]. Available: https://www.iosrjournals.org/iosr-jeee/papers/Vol14%20Issue4/Series-2/E1404023541.pdf

[20] R. Kennel, “Exercise: Space vector modulation,” Munich, Germany, 2012, power Electronics Exercise Material. [Online]. Available: http://www.eal.ei.tum.de

[21] J. Lenk, *Simplified Design of Filter Circuits*. Butterworth-Heinemann / Newnes, 1999, eDN Series for Design Engineers.

# APPENDIX

## L.1 STM32 NUCLEO-F446RET6 Programming

### L.1.1 Source Code in C (main.c) via STM32CubeIDE

```
1
2  /* Includes -----*/
3  #include "main.h"
4
5  /* Private includes -----*/
6  /* USER CODE BEGIN Includes */
7  #include <math.h>
8  /* USER CODE END Includes */
9
10 /* Private typedef -----*/
11 /* USER CODE BEGIN PTD */
12
13 /* USER CODE END PTD */
14
15 /* Private define -----*/
16 /* USER CODE BEGIN PD */
17
18 #define ADC_CHANNEL_COUNT   4
19 #define RAW_MAX             3598.0f     //Operable ADC Raw
      Value Output
20 #define RAD_60DEG           1.04719755f //PI/3
21
22 #define DEG2RAD             0.01745329f //One Side  | PI/180
23 #define RADSEC2RPM          9.54929658f //One Side  |
      60s/2PI
24
25 #define I_MAX               100.0f      //Two Sides |
      Range: -25A to 25A
26 #define V_MAX               150.0f      //Two Sides |
      Range: -50V to 50V
27 #define OMEGA_MAX           250.0f      //Two Sides |
      Range: -100rad/s to 100rad/s
28
29 /* USER CODE END PD */
30
31 /* Private macro -----*/
```

```
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/
ADC_HandleTypeDef hadc1;

DAC_HandleTypeDef hdac;

TIM_HandleTypeDef htim8;

/* USER CODE BEGIN PV */

uint16_t adc_raw[ADC_CHANNEL_COUNT];
uint8_t loop, setpoint, step;

float RAW_TO_CURR, VOLT_TO_RAW, RAW_TO_ANGLE, RAW_TO_OMEGA;

//Setpoint
int itr;
int pdelay, ptrans, p1, p2, p3, p4;
uint8_t trigger, triggerPrev;
uint8_t buttonPrevState, buttonCurrState;

//Feedback
float FB_Ia, FB_Ib, FB_theta, FB_omega;
float theta_cos, theta_sin;
float FB_Id, FB_Iq;
float theta_deg, omega_rpm, Torque_e, Torque_m;
//Motor
float M_Rs, M_Ld, M_Lq, M_Psir, M_PP;
float M_Te, M_Icont, M_Ipeak;
//Inverter
float I_freq, I_Vdc;
float I_percent, I_Tout, I_Imag;
//Control
float C_setpoint_d, C_setpoint_q;
float C_bw, C_dt, C_delay;
float C_error_d, C_error_q;
float C_prevError_d, C_prevError_q;
float C_out_d, C_out_q;
```

```
float C_Kp_d, C_Ki_d, C_Kb_d;
float C_Kp_q, C_Ki_q, C_Kb_q;
float C_proportional_d, C_proportional_q;
float C_integral_d, C_integral_q;
float C_satL, C_satH;
//Output
float Out_Vd, Out_Vq;
//Space Vector
float SV_alpha, SV_beta;
float SV_Vmag, SV_ma, SV_Vlim;
float SV_ARR, SV_pwmU, SV_pwmV, SV_pwmW;
uint8_t SV_sector;
float maDetect, t1, t2, t0;

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_DAC_Init(void);
static void MX_TIM8_Init(void);
/* USER CODE BEGIN PFP */

void ClarkePark(float a, float b, float* d, float* q){
    //Called each current measurement
        //Clarke Transformation
    float Beta = a/M_SQRT3 + b*2.0/M_SQRT3;
        //Park Transformation
    *d = theta_cos*a + theta_sin*Beta;
    *q = -theta_sin*a + theta_cos*Beta;
}

void PI_Tustin(float error_d, float error_q, float* out_d,
   float* out_q){
    //Tustin Trapezoidal Bilinear Transform PI Controller
        //Proportional
    C_proportional_d = C_Kp_d * error_d;
    C_proportional_q = C_Kp_q * error_q;
        //Integral
```

```
112     C_integral_d += 0.5f * C_Ki_d * C_dt *
    (error_d+C_prevError_d);
113     C_integral_q += 0.5f * C_Ki_q * C_dt *
    (error_q+C_prevError_q);
114         //Integral Error Update
115     C_prevError_d = error_d;
116     C_prevError_q = error_q;
117         //Initial Output
118     float initial_d = C_proportional_d + C_integral_d;
119     float initial_q = C_proportional_q + C_integral_q;
120         //Output Clamping
121     if(initial_d < C_satL) *out_d = C_satL;
122     else if(initial_d > C_satH) *out_d = C_satH;
123     else *out_d = initial_d;
124     if(initial_q < C_satL) *out_q = C_satL;
125     else if(initial_q > C_satH) *out_q = C_satH;
126     else *out_q = initial_q;
127         //Back-Calculation
128     C_integral_d += C_Kb_d * (*out_d - initial_d);
129     C_integral_q += C_Kb_q * (*out_q - initial_q);
130 }
131
132 void Feedforward(float in_d, float in_q, float* out_d,
    float* out_q){
133     //Coupling Term Compensation via Feedforward Control
134     *out_d = in_d - (FB_omega * M_Lq * FB_Iq);
135     *out_q = in_q + (FB_omega * M_Ld * FB_Id) + (FB_omega *
    M_Psir);
136 }
137
138 void OpenLoop(float d, float q, float* out_d, float* out_q){
139     //Open-Loop Steady-State Output
140     *out_d = (d*M_Rs) - (FB_omega * M_Lq * q);
141     *out_q = (q*M_Rs) + (FB_omega * M_Ld * d) + (FB_omega *
    M_Psir);
142 }
143
144 void SVPWM(float d, float q, float theta, float pwm_period,
    float *pwm_u, float *pwm_v, float *pwm_w){
145     //Space Vector Pulse Width Modulation
146         //Vector Magnitude
```

```
147      SV_Vmag      = hypotf(d,q);
148      float scale = (SV_Vmag > SV_Vlim) ? SV_Vlim/SV_Vmag :
     1.0f;
149      d *= scale, q *= scale, SV_Vmag *= scale;
150  //  SV_Vmag = SV_Vlim;
151
152          //Inverse Park Transformation
153      SV_alpha     = ((d * theta_cos) - (q * theta_sin)) *
     M_SQRT3 / I_Vdc * pwm_period;
154      SV_beta      = ((d * theta_sin) + (q * theta_cos)) *
     M_SQRT3 / I_Vdc * pwm_period;
155
156          //Sector Detection
157      if (SV_beta >= 0.0f) {
158          if (SV_alpha >= 0.0f)    SV_sector = (SV_beta   /
     M_SQRT3 > SV_alpha) ? 2 : 1;
159          else                     SV_sector = (-SV_beta /
     M_SQRT3 > SV_alpha) ? 3 : 2;
160      }
161      else {
162          if (SV_alpha >= 0.0f)    SV_sector = (-SV_beta /
     M_SQRT3 > SV_alpha) ? 5 : 6;
163          else                     SV_sector = (SV_beta   /
     M_SQRT3 > SV_alpha) ? 4 : 5;
164      }
165
166          //Active Vector Times
167      switch(SV_sector) {
168          case 1:
169              t1 = (M_SQRT3 * SV_alpha - SV_beta)/2.0;
170              t2 = SV_beta;
171
172              *pwm_u = (pwm_period + t1 + t2) / 2.0;
173              *pwm_v = *pwm_u - t1;
174              *pwm_w = *pwm_v - t2;
175              break;
176
177          case 2:
178              t1 = (M_SQRT3 * SV_alpha + SV_beta)/2.0;
179              t2 = (-M_SQRT3 * SV_alpha + SV_beta)/2.0;
180
```

```
            *pwm_v = (pwm_period + t1 + t2) / 2.0;
            *pwm_u = *pwm_v - t2;
            *pwm_w = *pwm_u - t1;
            break;

        case 3:
            t1 = SV_beta;
            t2 = (-M_SQRT3 * SV_alpha - SV_beta)/2.0;

            *pwm_v = (pwm_period + t1 + t2) / 2.0;
            *pwm_w = *pwm_v - t1;
            *pwm_u = *pwm_w - t2;
            break;

        case 4:
            t1 = (-M_SQRT3 * SV_alpha + SV_beta)/2.0;
            t2 = -SV_beta;

            *pwm_w = (pwm_period + t1 + t2) / 2.0;
            *pwm_v = *pwm_w - t2;
            *pwm_u = *pwm_v - t1;
            break;

        case 5:
            t1 = (-M_SQRT3 * SV_alpha - SV_beta)/2.0;
            t2 = (M_SQRT3 * SV_alpha - SV_beta)/2.0;

            *pwm_w = (pwm_period + t1 + t2) / 2.0;
            *pwm_u = *pwm_w - t1;
            *pwm_v = *pwm_u - t2;
            break;

        case 6:
            t1 = -SV_beta;
            t2 = (M_SQRT3 * SV_alpha + SV_beta)/2.0;

            *pwm_u = (pwm_period + t1 + t2) / 2.0;
            *pwm_w = *pwm_u - t2;
            *pwm_v = *pwm_w - t1;
            break;
    }
```

```
222
223         //Output Period Clamping
224     if(*pwm_u > pwm_period) *pwm_u = pwm_period;
225     if(*pwm_v > pwm_period) *pwm_v = pwm_period;
226     if(*pwm_w > pwm_period) *pwm_w = pwm_period;
227 }
228
229 /* USER CODE END PFP */
230
231 /* Private user code -----*/
232 /* USER CODE BEGIN 0 */
233
234 void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef
    *hadc){
235         //ADC Preprocessing
236     adc_raw[0] = HAL_ADCEx_InjectedGetValue(&hadc1,
    ADC_INJECTED_RANK_1);
237     if(adc_raw[0] > RAW_MAX) adc_raw[0] = RAW_MAX;
238     adc_raw[1] = HAL_ADCEx_InjectedGetValue(&hadc1,
    ADC_INJECTED_RANK_2);
239     if(adc_raw[1] > RAW_MAX) adc_raw[1] = RAW_MAX;
240     adc_raw[2] = HAL_ADCEx_InjectedGetValue(&hadc1,
    ADC_INJECTED_RANK_3);
241     if(adc_raw[2] > RAW_MAX) adc_raw[2] = RAW_MAX;
242     adc_raw[3] = HAL_ADCEx_InjectedGetValue(&hadc1,
    ADC_INJECTED_RANK_4);
243     if(adc_raw[3] > RAW_MAX) adc_raw[3] = RAW_MAX;
244
245         //Currents
246     FB_Ia       = (float)adc_raw[0] * RAW_TO_CURR - I_MAX;
247     FB_Ib       = (float)adc_raw[1] * RAW_TO_CURR - I_MAX;
248         //Theta
249     FB_theta = (float)adc_raw[2] * RAW_TO_ANGLE;
250     theta_cos = cosf(FB_theta), theta_sin = sinf(FB_theta);
251
252         //Omega
253     FB_omega    = (float)adc_raw[3] * RAW_TO_OMEGA -
    OMEGA_MAX;
254
255     //SETPOINT SETTER
256     if(step == 0){
```

```
257         //External Trigger
258         HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1,
    DAC_ALIGN_12B_R, 4095);
259         step = 1;
260         C_setpoint_q    = 0;
261         }
262 
263     if(step && setpoint){
264         itr++;
265         if(itr <= pdelay) C_setpoint_q = 0.0;
266         if(itr > pdelay && itr <= p1) C_setpoint_q =
    M_Icont;
267         if(itr > p1 && itr <= p2) C_setpoint_q =
    2.0*M_Icont;
268         if(itr > p2 && itr <= p3) C_setpoint_q =
    0.5*M_Icont;
269         if(itr > p3 && itr <= p4) C_setpoint_q = M_Ipeak;
270         if(itr > p4) {
271             C_setpoint_q = 0.0;
272             itr = 0;
273         }
274     }
275     else if(step && !setpoint){
276         itr++;
277         if(itr > pdelay) C_setpoint_q = I_Imag;
278     }
279 
280     //CONTROL
281     if(!loop){
282         //OPEN-LOOP CONTROL
283             //Open-Loop Calculation
284 //      void OpenLoop(float d, float q, float* out_d,
    float* out_q)
285         OpenLoop(C_setpoint_d, C_setpoint_q, &Out_Vd,
    &Out_Vq);
286     }
287     else if(loop){
288         //CLOSED-LOOP CONTROL
289             //Reference Frame Transformation
290 //      void ClarkePark(float a, float b, float* d, float*
    q)
```

```
291         ClarkePark(FB_Ia, FB_Ib, &FB_Id, &FB_Iq);
292             //PI Controller
293         C_error_d = C_setpoint_d - FB_Id;
294         C_error_q = C_setpoint_q - FB_Iq;
295 //      void PI_Tustin(float error_d, float error_q, float*
    out_d, float* out_q)
296         PI_Tustin(C_error_d, C_error_q, &C_out_d, &C_out_q);
297             //Feedforward Control
298 //      void Feedforward(float in_d, float in_q, float*
    out_d, float* out_q)
299         Feedforward(C_out_d, C_out_q, &Out_Vd, &Out_Vq);
300      }
301
302      //SVPWM OUTPUT
303         //Space Vector Calculations
304 //  void SVPWM(float alpha, float beta, float theta,
    int32_t pwm_period, int32_t *pwm_u, int32_t *pwm_v,
    int32_t *pwm_w)
305      SVPWM(Out_Vd, Out_Vq, FB_theta, SV_ARR, &SV_pwmU,
    &SV_pwmV, &SV_pwmW);
306         //Duty Cycle Output
307      TIM8->CCR1 = (SV_pwmU);
308      TIM8->CCR2 = (SV_pwmV);
309      TIM8->CCR3 = (SV_pwmW);
310
311      //MONITORING VARIABLES
312      theta_deg   = FB_theta / DEG2RAD;
313      omega_rpm   = FB_omega * RADSEC2RPM;
314      maDetect    = SV_Vmag / (I_Vdc / M_SQRT3);
315      Torque_e    = (1.5) * M_PP * FB_Iq * (M_Psir +
    (M_Ld-M_Lq)*FB_Id);
316 }
317
318 /* USER CODE END 0 */
319
320 /**
321   * @brief  The application entry point.
322   * @retval int
323   */
324 int main(void)
325 {
```

```
326   /* USER CODE BEGIN 1 */
327
328     RAW_TO_CURR     = (2*I_MAX)/RAW_MAX;
329     VOLT_TO_RAW     = RAW_MAX/(2*V_MAX);
330     RAW_TO_ANGLE    = M_TWOPI/RAW_MAX;
331     RAW_TO_OMEGA    = (2*OMEGA_MAX)/RAW_MAX;
332
333     //MODE SELECTION
334 //  loop    = 0;    //OPEN-LOOP
335     loop    = 1;    //CLOSED-LOOP
336
337     setpoint    = 0;    //TRANSIENT TO STEADY
338 //  setpoint    = 1;    //VARIABLE
339
340     step        = 0;
341     itr         = 0;
342
343     //PMSM
344         //BSM90N-175
345     M_Rs        = 1.24;         //Ohm
346     M_Ld        = 0.00415;      //Henry
347     M_Lq        = 0.00415;      //Henry
348     M_PP        = 4;            //Pairs
349     M_Psir      = 0.173797198;  //Wb
350     M_Icont     = 7.8;          //A
351     M_Ipeak     = 28.1;         //A
352     M_Te        = 1.5 * M_PP * M_Icont * M_Psir;
353
354     //Inverter
355     I_Vdc       = 300.0;        //V
356     I_freq      = 8000.0;       //Hertz
357     I_percent   = 100.0;
358 //  I_Imag      = I_percent/100.0 * M_Icont;
359     I_Imag      = M_Ipeak;
360     I_Tout      = M_Te * (I_percent/100.0);
361
362     //Control
363     C_setpoint_d = 0.0;
364     C_setpoint_q = 0.0;
365
366     C_bw        = 400.0 * M_TWOPI;
```

```
367     C_dt         = 1.0/I_freq;
368     C_delay      = 0.010;
369
370     C_satL       = -V_MAX;
371     C_satH       = V_MAX;
372
373     C_Kp_d       = C_bw * M_Ld;
374     C_Ki_d       = C_bw * M_Rs;
375     C_Kb_d       = 1/C_Kp_d;
376
377     C_Kp_q       = C_bw * M_Lq;
378     C_Ki_q       = C_bw * M_Rs;
379     C_Kb_q       = 1/C_Kp_q;
380
381     //Space Vector
382     SV_ARR = 10000;
383     SV_ma = 99.9/100.0;
384     SV_Vlim = SV_ma * I_Vdc / M_SQRT3;
385
386     //Variable Setpoint Mode
387         //Trigger Delay 10ms
388     pdelay  = (int)((C_delay) / C_dt);
389         //Step Input 150ms
390     ptrans  = (int)((0.15 / C_dt) + pdelay);
391         //Variable Setpoint Durations
392     p1       = (int)(300 + pdelay);  //1st Period 10ms
393     p2       = (int)(300 + p1);      //2nd Period 30ms
394     p3       = (int)(300 + p2);      //3rd Period 30ms
395     p4       = (int)(300 + p3);      //4th Period 30ms
396
397   /* USER CODE END 1 */
398
399   /* MCU Configuration-----*/
400
401   /* Reset of all peripherals, Initializes the Flash
       interface and the Systick. */
402   HAL_Init();
403
404   /* USER CODE BEGIN Init */
405
406   /* USER CODE END Init */
```

```
407
408     /* Configure the system clock */
409     SystemClock_Config();
410
411     /* USER CODE BEGIN SysInit */
412
413     /* USER CODE END SysInit */
414
415     /* Initialize all configured peripherals */
416     MX_GPIO_Init();
417     MX_ADC1_Init();
418     MX_DAC_Init();
419     MX_TIM8_Init();
420     /* USER CODE BEGIN 2 */
421
422     HAL_TIM_Base_Start(&htim8);
423
424     HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
425     HAL_TIMEx_PWMN_Start(&htim8, TIM_CHANNEL_1);
426     HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_2);
427     HAL_TIMEx_PWMN_Start(&htim8, TIM_CHANNEL_2);
428     HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_3);
429     HAL_TIMEx_PWMN_Start(&htim8, TIM_CHANNEL_3);
430     HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_4);
431
432     HAL_ADCEx_InjectedStart_IT(&hadc1);
433
434     HAL_DAC_Start(&hdac, DAC1_CHANNEL_1);
435     HAL_DAC_Start(&hdac, DAC1_CHANNEL_2);
436
437     /* USER CODE END 2 */
438
439 }
```

Listing 1. STM32 Source Code in C for FOC

## L.1.2 Configurations via STM32CubeMX

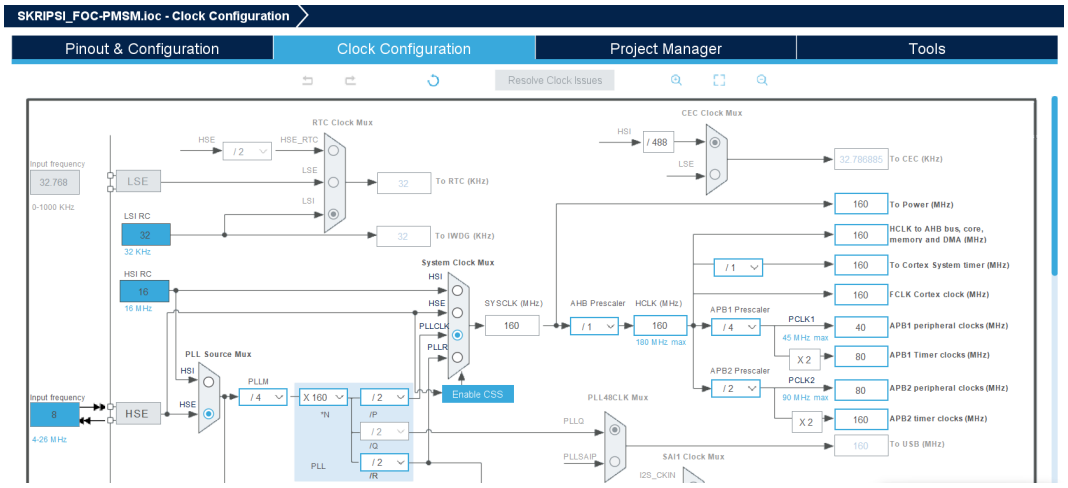


Figure 1. Clock Configuration.

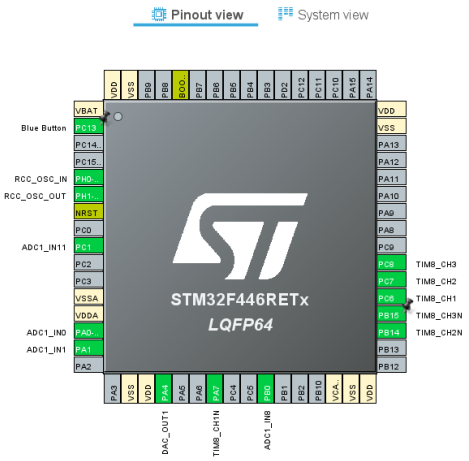


Figure 2. Pinout Configuration.

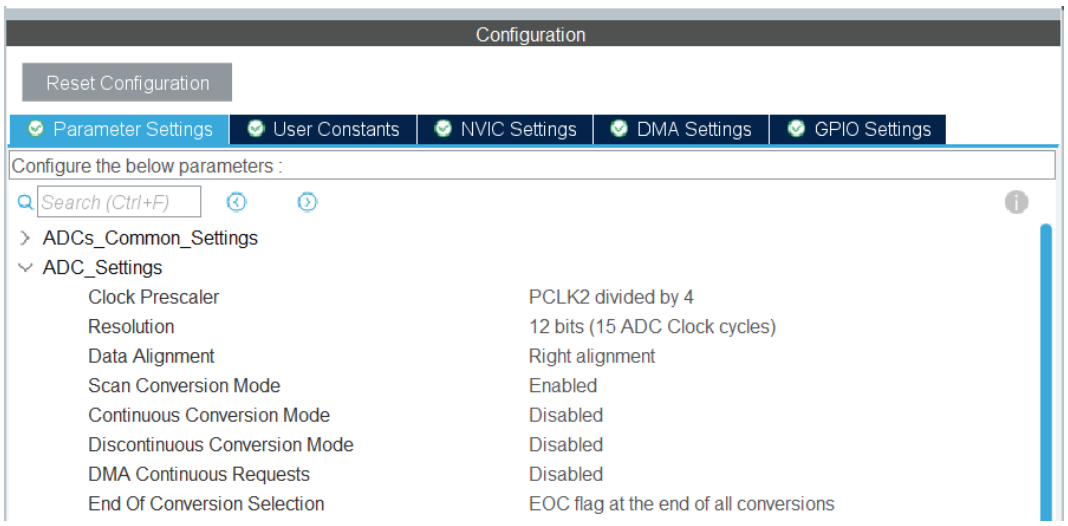


Figure 3. ADC1 Configuration.

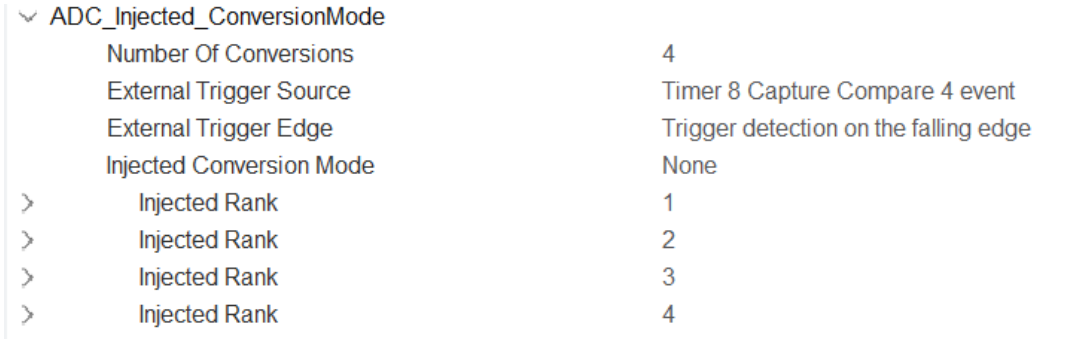


Figure 4. ADC1 Injected Mode.

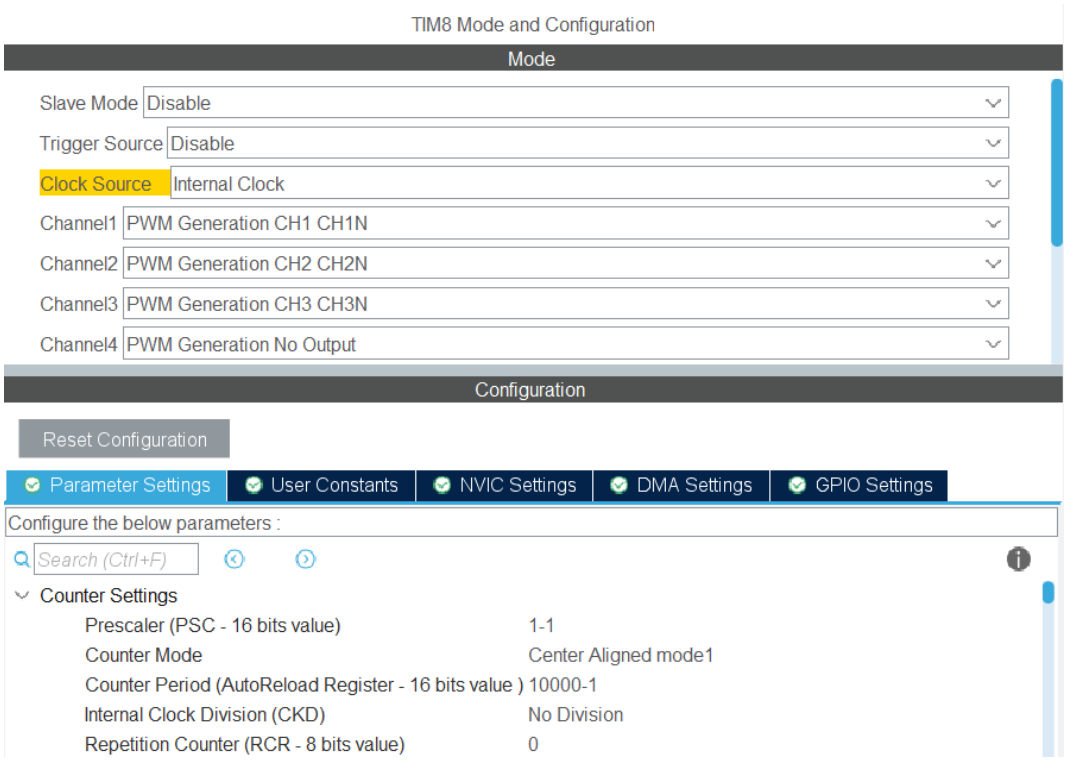


Figure 5. TIM8 Configuration.

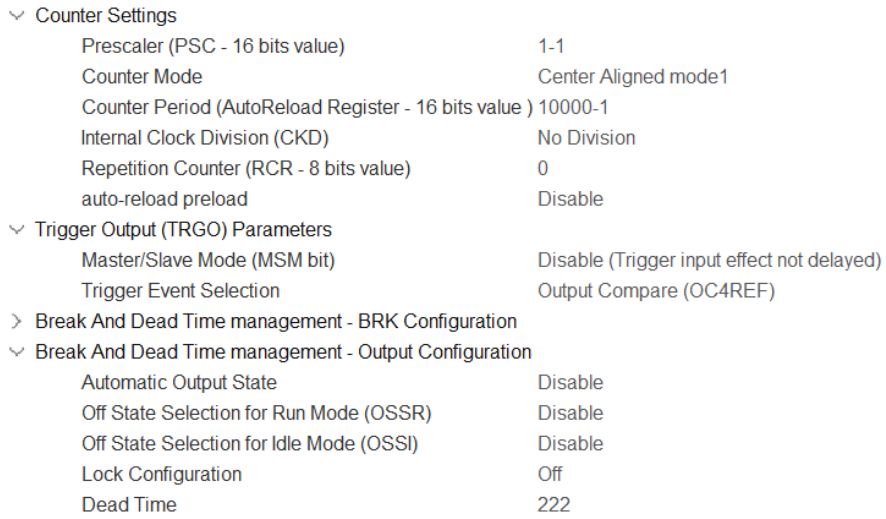


Figure 6. TIM8 Configuration for PWM.

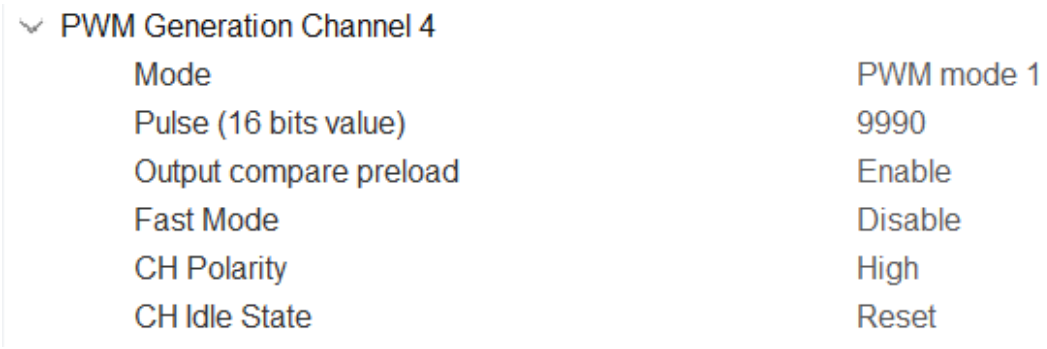


Figure 7. Injected Trigger via PWM4.

## L.2 PLECS HIL Parameter Code

```
1
2  %% ===== HIL Feedback Mode Selection ===== %%
3  %feedback    = 1 %ADC Testing
4  feedback     = 5 %PWM Input
5
6  %% ===== PMSM ===== %%
7  %% BSM90N-175 %%
8  M.Rs         = 1.24/2;            %Ohm
9  M.Ld         = 4.15e-3;       %Henry
10 M.Lq         = 4.15e-3;       %Henry
11 M.P          = 2*4;           %Poles
12 M.J      = 339e-6;        %Nms^2 or kgm^2
13 M.Psir       = 0.173797198;   %Wb
14 I.Vdc    = 300;           %V
15 M.Icont = 7.8;           %A
```

```
M.Ipeak = 28.1;          %A
M.Lss   = (M.Ld + M.Lq)/50;                          %Henry
M.Te        = 3/2 * M.P/2 * M.Icont * M.Psir;    %Nm

%% ===== Load ===== %
L.B = 0.75
L.J = 1e-3

%% ===== Output Setpoints ===== %%
control = 'Current'
%control = 'Torque'

Out.percent = 100
Out.Itheta  = 90

if (strcmp(control,'Current'))
    % --- Current Setpoint --- %
    Out.Imag    = M.Ipeak
    Out.Id      = Out.Imag * (cosd(Out.Itheta))
    Out.Iq      = Out.Imag * (sind(Out.Itheta))
    Out.Te      = (3/2) * (M.P/2) * Out.Iq * M.Psir
else
    % --- Torque Setpoint --- %
    Out.Te      = M.Te * (Out.percent/100)
    Out.Id      = 0
    Out.Iq      = (2/3) * Out.Te * (1/(M.P/2*M.Psir))
    Out.Imag        = Out.Iq
end

Out.omegaE  = M.P * Out.Te / L.B
Out.Vd      = (Out.Id*M.Rs)-(Out.omegaE*Out.Iq*M.Lq)
Out.Vq      =
   (Out.Iq*M.Rs)+(Out.omegaE*Out.Id*M.Ld)+(Out.omegaE*M.Psir)
Out.Vmag        = sqrt(Out.Vd*Out.Vd + Out.Vq*Out.Vq)

Out.Vdq     = [Out.Vd Out.Vq];
Out.Idq     = [Out.Id Out.Iq];

%% ===== Control Variables ===== %%
C.Fsw   = 8e3;
C.bwHz  = 250;
```

```
C.bw         = C.bwHz * 2*pi;
C.LPFnum     = [C.bw];
C.LPFden     = [1 C.bw];
C.delay = 10e-3;

%% ===== Feedback Setup ===== %%
% --- Pin Settings --- %
FB.Iab           = [15 14];
FB.Theta         = 13;
FB.Omega         = 12;
FB.PWM_A             = [16 17];
FB.PWM_B             = [18 3];
FB.PWM_C             = [20 5];
FB.Trig              = 15
%FB.PWM_STM32    = [16 17 18 3 20 5];
% --- Boundaries --- %
RT.Imax          = 100
RT.Vmax          = 150
RT.Omegamax = 250
RT.Vdig          = 2.9225
% --- Test Values --- %
Test.Ia          = 20
Test.Ib          = 10
Test.Theta   = 105 * (pi/180)
Test.Omega   = 85
```

Listing 2. HIL Simulation Parameter for FOC

### L.3 MATLAB Code

```

clear
clc

%Raditya Prajnabuwana | 21/479720/TK/52923

%% Low Pass Filter: RC Filter

LPF.wcc = 800 * 2 * pi;
LPF.RC = 1/LPF.wcc;

% Transfer function
```

```
s = tf('s');
LPF.TF = 1 / (1 + s*LPF.RC);

%% PI Controller

M.Rs = 1.24;
M.Ls = 4.15e-3;
M.TF = 1/(M.Rs + s*M.Ls);

C.bw = 400 * 2*pi;
C.Kp = C.bw * M.Ls;
C.Ki = C.bw * M.Rs;

% Transfer function
C.PITF = C.Kp + (C.Ki/s);

%% PMSM

M.PolePair  = 4;
M.Psir      = 0.174;
M.Ipeak     = 28.1;
M.Te_peak   = 3/2 * M.PolePair * M.Psir * M.Ipeak;

L.B         = 0.75;
M.Wm_peak   = M.Te_peak / L.B;
M.fe_peak   = M.Wm_peak * M.PolePair / (2*pi);

%% Full Transfer Function

G.Open = C.PITF * M.TF;
G.Closed = feedback(G.Open, LPF.TF);
G.Unfiltered = feedback(G.Open, 1); %Look for 200Hz


%% ============== PLOTS ==============

w = logspace(0, 5, 5000);        % rad/s
f = w / (2*pi);                  % Hz

% ==========================================
%  1. Bode Plot: G.Unfiltered vs G.Closed
```

```
% ==========================================
bodeopts = bodeoptions;
bodeopts.FreqUnits = 'Hz';
bodeopts.Grid = 'on';

fig1 = figure;
bode(LPF.TF,       w, bodeopts); hold on;
bode(G.Unfiltered, w, bodeopts); hold on;
bode(G.Closed,     w, bodeopts);

legend('LPF', 'Unfiltered', 'Filtered');

% Export
%exportgraphics(fig1, 'MATLAB-BodeFull.png', 'Resolution', 600);

% ==========================================
%  2. Bode Mag: G.Unfiltered vs G.Closed
% ==========================================
fig2 = figure;
bodemag(LPF.TF,       w, bodeopts); hold on;
bodemag(G.Unfiltered, w, bodeopts); hold on;
bodemag(G.Closed,     w, bodeopts);
title('Bode Plot: Unfiltered vs Filtered');

% --- Vertical dashed lines ---
xline(LPF.wcc/(2*pi), '--b');   % blue
xline(C.bw/(2*pi), '--r');   % red
xline(M.fe_peak, '--', 'Color', [1 0.5 0]);   % dashed orange
   line
legend('LPF', 'Unfiltered', 'Filtered', 'LPF Cut-Off', 'PI
   Bandwidth', 'Freq at Ipeak');

% Export
%exportgraphics(fig2, 'MATLAB-BodeMag.png', 'Resolution', 600);

% ==========================================
%  3. Root Locus of G.Open
% ==========================================
fig3 = figure;
rlocus(G.Open);
title('Root Locus of Pole-Zero Cancelled System');
```

```
grid on;

% --- Customize poles (x) and zeros (o) ---
h = findobj(gca, 'Type', 'line');

for k = 1:length(h)
    if strcmp(h(k).Marker, 'x')
        h(k).MarkerSize = 14;      % larger poles
        h(k).LineWidth = 1.8;
        h(k).Color = [1 0 0];      % red poles
    elseif strcmp(h(k).Marker, 'o')
        h(k).MarkerSize = 10;      % slightly smaller zeros
        h(k).LineWidth = 1.4;
        % keep default color
    end
end

% ==========================================
%  4. Step Response: G.Closed vs G.Unfiltered
% ==========================================
fig4 = figure;
step(G.Unfiltered); hold on;
step(G.Closed);

title('Filtered and Unfiltered Closed-Loop System');
legend('Unfiltered','Filtered');
grid on;

% Export
%exportgraphics(fig4, 'MATLAB-StepResponse.png', 'Resolution',
   600);
```

Listing 3. MATLAB Code for Bandwidth Evaluation

## L.4 Test Bench

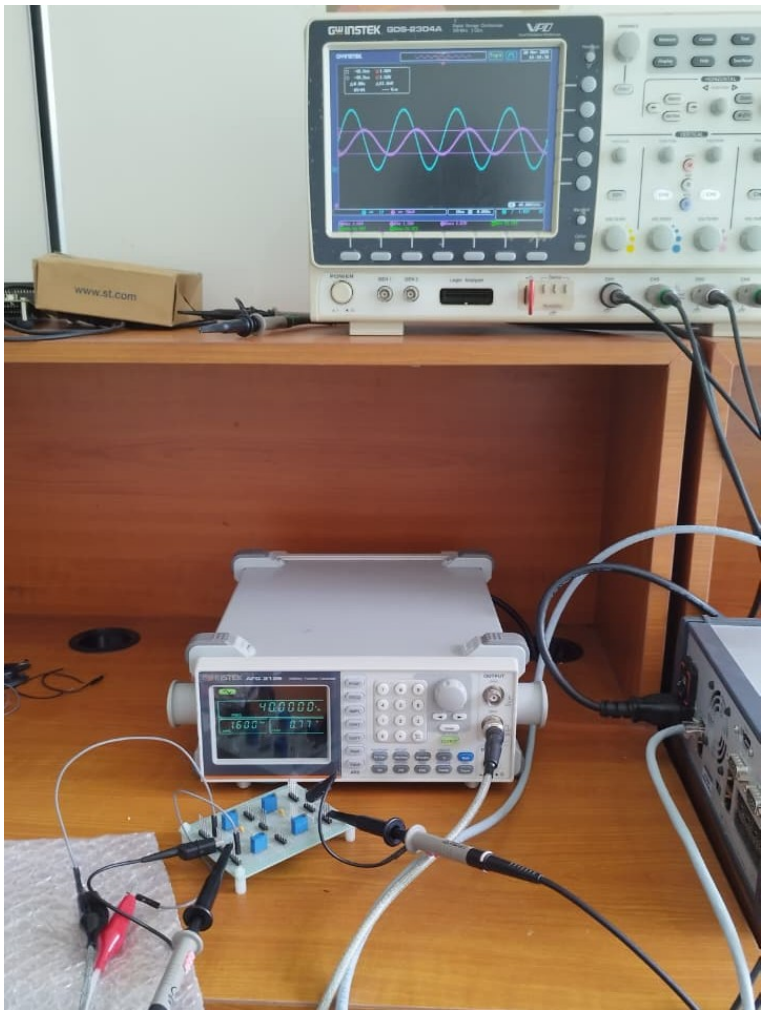


Figure 8. LPF Test Bench.

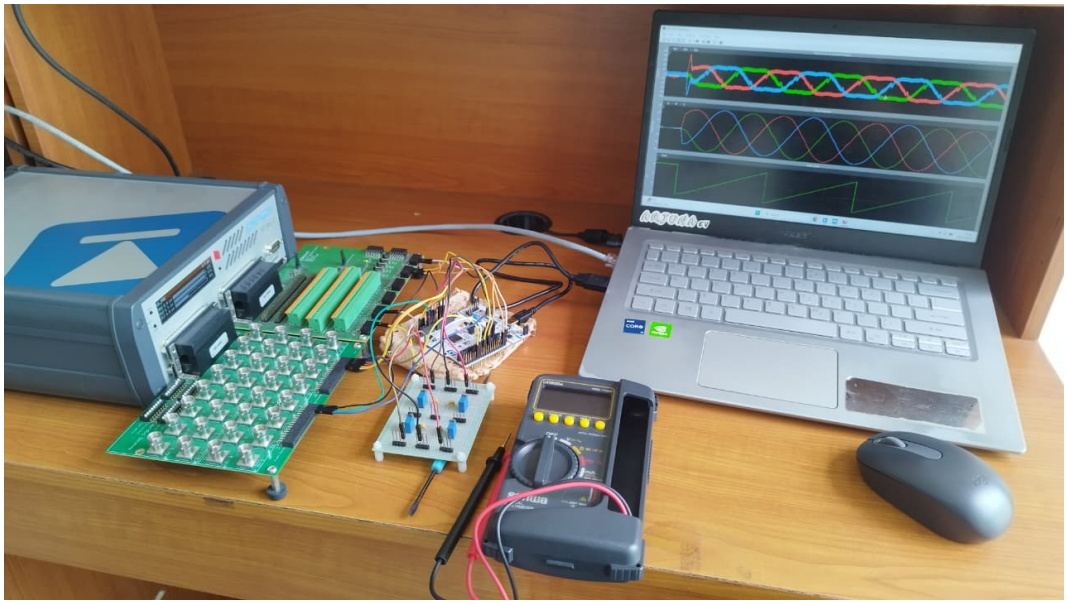

Figure 9. HIL Test Bench.