MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

DESIGN AND IMPLEMENTATION OF A 3-PHASE 2-LEVEL INVERTER WITH OUTPUT VOLTAGE REGULATION IMPROVEMENT

SEDAT ÇAĞLAR

MASTER THESIS

Department of Electrical and Electronics Engineering

Thesis Supervisor

Asst. Prof. Dr. Salih BAYAR

ISTANBUL, 2024

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

DESIGN AND IMPLEMENTATION OF A 3-PHASE 2-LEVEL INVERTER WITH OUTPUT VOLTAGE REGULATION IMPROVEMENT

SEDAT ÇAĞLAR
(525021004)

MASTER THESIS

Department of Electrical and Electronics Engineering

Thesis Supervisor

Asst. Prof. Dr. Salih BAYAR

ISTANBUL, 2024

ACKNOWLEDGEMENT

I am deeply grateful to my esteemed professor, Asst. Prof. Salih BAYAR., whose invaluable guidance and unwavering support were instrumental in shaping my master's thesis. His trust in my abilities and constant encouragement have been a source of inspiration throughout this journey.

I am filled with gratitude for my friend Burak GÜNAY, whose partnership I consider a stroke of luck. His expertise in embedded software design and his unwavering support during challenging times were crucial in my journey.

Lastly, I want to express my deepest gratitude to my beloved wife, Elif ÇAĞLAR. Her constant support and understanding have been my anchor throughout this thesis. I am also grateful to my dear son, Çınar ÇAĞLAR, whose infectious smile has always brightened even the most challenging days when I couldn't spend enough time with him due to my work schedule.

CONTENTS

ACKNOWLEDGEMENT i

ÖZET vii

ABSTRACT viii

SYMBOLS ix

ABBREVIATIONS xi

LIST OF FIGURES xiv

LIST OF TABLES xx

1. INTRODUCTION 1

2. RELATED WORK 4

3. MOTIVATION AND BACKGROUND 7

3.1. Classification of Inverters According to Current Source 7

3.1.1. Voltage source inverter 7

3.1.2. Current source inverter 8

3.2. Classification of Inverters According to Phase Structure 9

3.2.1. Single phase inverters 9

3.2.2. Three phase Inverters 12

3.3. Classification of Inverters According to Number of Levels 12

3.3.1. Two level inverters 13

3.3.2. Multilevel inverters 15

3.4. Three-Phase VSI Modulation Techniques 18

3.4.1. Sinusoidal pulse width modulation 19

3.4.2. Space vector pulse width modulation method 20

3.4.3. Third harmonic injection method 21

3.4.4. Reasons for preference of SPWM 22
3.5. Three-Phase VSI Inverter Control Techniques 23
3.5.1. PI control technique 23
3.5.2. Classical RMS voltage regulation with PI control 25
3.5.3. RMS voltage regulation with moving RMS filter 27
3.5.4. ADC oversampling method 29
3.5.5. Vectoral control with Park-Clarke transformations 31
4. METHODOLOGY 34
4.1. Hardware Design Steps 36
4.1.1. Determination of main system parameters 36
4.1.2. Output filter design 37
4.1.3. Power switch selection 42
4.1.4. DC link capacity selection 46
4.1.5. Gate driver design and DCDC selection for gate driver supplies 47
4.1.5.1. Bipolar totem pole gate drive 47
4.1.5.2. Bootstrap gate drive 48
4.1.5.3. Optocoupler gate drive 48
4.1.5.4 Capacitive gate drive 49
4.1.5.5 Transformer gate drive 50
4.1.6. Measurement circuit design 52
4.1.6.1. Voltage measurement circuit 53
4.1.6.2. Current measurement circuit 56
4.1.7. Auxiliary power supply design & selection 58
4.1.7.1. +25V to +5V converter 59

4.1.7.2. +5V to -5V converter 60
4.1.7.3. +5V to +15V isolated converter 60
4.1.7.4. +12V to +5V converter 62
4.1.7.5. +5V to +3.3V converter 62
4.1.8. Connector selection 63
4.2. Simulation Studies 64
4.2.1. LtSpice simulations 64
4.2.1.1. Main voltage source inverter simulation 64
4.2.1.2. Gate driver circuit simulation 66
4.2.1.3. Monte Carlo tolerance analysis 68
4.2.1.4. +5V to -5V converter simulation 70
4.2.2. Matlab simulations 72
4.2.3. Plecs simulations 73
4.3. Software Design Steps 78
4.3.1. STM32G431 at a glance 78
4.3.1.1. NucleoG431RB 81
4.3.2. Determination of necessary I/Os and I/O types 83
4.3.3. Calculation of ADC oversampling capacity 85
4.3.4. Writing bare-metal driver modules 88
4.3.4.1. Rationale behind choosing bare-metal programming 88
4.3.4.2. Embedded algorithm programming process 89
4.3.5. Developing embedded control algorithm 91
4.3.6. Programming embedded control algorithm 93
4.3.6.1. Configuration of Nucleo-G431RB 93

4.3.6.2. Writing the ADC oversampling code 99
4.3.6.3. Writing the moving RMS filter code 102
4.3.6.4. Writing the PI controller code 105
4.3.6.5. Combining everything and creating the algorithm 106
5. IMPLEMENTATION 110
5.1. Component Footprints & Stack-up 110
5.2. Schematic and PCB Layout 113
5.2.1. Input filter and protection circuits 114
5.2.2. Voltage measurement circuits 114
5.2.3. Current measurement circuits 115
5.2.4. Gate drive circuits 116
5.2.5. Output filter circuits 117
5.2.6. NucleoG431 Peripheral and Connection Circuits 118
5.2.7. Auxiliary power supply circuits 119
5.2.7.1. +25V to +5V converter circuit 119
5.2.7.2. +5V to -5V converter circuit 120
5.2.7.3. +12V to +5V converter circuit 120
5.2.7.4. +5V to +3.3V converter circuit 121
5.3. PCB Manufacturing and Assembly 121
5.4. Embedded Control Implementation on Designed Hardware 122
6. EXPERIMENTAL RESULTS 123
6.1. Sampling Accuracy 123
6.2. Static Response 125
6.3. Dynamic Response 127

7. CONCLUSION 136

REFERENCES 139

APPENDIX 146

ÖZET

ÇIKIŞ VOLTAJ REGÜLASYONU İYİLEŞTİRMESİ İLE 3-FAZ 2-SEVİYELİ İNVERTER TASARIMI VE İMPLEMENTASYONU

İnverter çeşitleri, Kesintisiz Güç Kaynakları (KGK), çeşitli motor sürücü uygulamaları ve biyomedikal cihazlar gibi birçok alanda kullanmaktadır. İnverter uygulamalarında en yaygın kullanım alanına sahip topoloji tipi ise Voltaj Kaynaklı İnverter (VKİ) diye de isimlendirebileceğimiz 2-seviyeli inverterlerdir. Birçok uygulamada inverter çıkışındaki voltaj regülasyonunu optimum düzeyde tutmak ve ani değişimlere hızlı tepki vermek hayati önem taşımaktadır. Bu tez çalışması değişken yük koşullarında dahi inverter çıkış voltaj regülasyonunda minimum bozulma ve hızlı toparlanma hedeflenerek yapılmıştır.

Bu tez çalışmasında 125 Watt gücünde, şebekeden bağımsız 3-faz bir inverter tasarlanmıştır. Kontrol biriminin tüm kaynak kodu, kaydedici tabanlı programlama ile özgün olarak yazılmıştır. Giriş gerilimi 52V DC, çıkış gerilimi ise 16V AC olarak belirlenmiştir. Sistem, inverter verimi %90 veya üzerinde olacak şekilde tasarlanmıştır.

Çalışmada geliştirilen kontrol metodu öncelikle Plecs ve Simulink programları kullanılarak simule edilmiştir. Ardından, algoritma STM32G431RB işlemcisi üzerinde kodlanarak tasarlanan donanım üzerinde gerçeklenmiştir. Çalışmada geliştirilen metot, her bir kesme periyodunda, ADC'nin aşırı örnekleme özelliği kullanılarak örneklemlerin elde edildiği bir sistemde, kayan RMS uygulanarak PI kontrol yapılmasına dayanır. Çalışmada yöntemin literatürde sıkça kullanılan bir diğer kontrol tekniği olan klasik RMS voltaj regülasyonu tekniğine göre avantajları da ortaya konulmuştur.

ABSTRACT

DESIGN AND IMPLEMENTATION OF A 3-PHASE 2-LEVEL INVERTER WITH OUTPUT VOLTAGE REGULATION IMPROVEMENT

We use inverter types in many areas such as Uninterruptible Power Supplies (UPS), various motor drive applications and biomedical devices. The most accepted topology in inverter applications is 2-level inverters, which is also called Voltage Source Inverter (VMI). In many applications, it is vital to keep the voltage regulation at the inverter output at an optimum level and to react quickly to sudden load changes. This thesis study has been carried out based on the principle of minimum distortion and fast recovery in the inverter output voltage under variable load conditions.

In this thesis, an off-grid 3-phase inverter prototype with a power level of 125 watts was designed. The entire source code of the control unit was written originally with register-based programming. The input voltage is determined as 52V DC, and the output voltage is determined as 16V AC. The system is designed to have an efficiency of 90% or above.

The control method developed in the study was first tested in simulation environment using Plecs and Simulink software. Then, the control algorithm was implemented on the hardware, using STM32G431RB microcontroller. The method developed in the study relies on applying sliding RMS to perform PI control in a system where all samples are obtained using the oversampling feature of the ADC at each interrupt period. In the study, the advantages of this method over another commonly used control technique in the literature, the classic RMS voltage regulation technique, have also been demonstrated.

SYMBOLS

ω	: Angular frequency
Φ	: Phase
τ	: Time constant
C_{filter}	: Output filter capacitance
f_{ADC}	: ADC modules frequency
f_c	: Cut-off frequency of LC filter
F_o	: Fundamental frequency
F_{sw}	: Switching frequency
$I_{in,RMS}$	: RMS input current
$I_{phase,peak}$	: Peak phase output current
I_{phase}	: RMS phase output current
Kd	: Derivative gain
Ki	: Integrator gain
Kp	: Proportional gain
L_{filter}	: Output filter inductance
M	: Modulation index
N	: Number of samples
P_{cond}	: Switching loss of the MOSFET
P_{dcdc}	: Output power of isolated DCDC converters
$P_{out,Phase}$	: Output power per phase
$P_{s,gate}$	: Gate charge loss of the MOSFET
$P_{s,oss}$	: Loss caused by the parasitic capacitances ($C_{iss} + C_{rss}$) of the MOSFET
P_{sw}	: Switching loss of the MOSFET
P_{total}	: Total loss of the MOSFET
$R_{ds,on}$	: MOSFET on resistance
R_{gate}	: Gate driving resistance
T	: Time
T_o	: Fundamental period

T_{off} : MOSFET closing duration

T_{on} : MOSFET opening duration

T_{sw} : Switching period

V_{actual} : Real measured voltage

V_{dc} : Busbar voltage

V_{dcdc} : Output voltage of isolated DCDC converters

V_{ds} : MOSFETs drain-source voltage

V_{meas} : Measured voltage at the MCU input terminals

$V_{ph,peak}$: Max phase output voltage

V_{phase} : RMS phase output voltage

ΔI_L : Ripple current

H : Efficiency

Θ : Rotation angle

ABBREVIATIONS

AC	: Alternating Current
ADC	: Analog-to-Digital Converter
A-NPC	: Active Neutral Point Clamped
API	: Application Programming Interface
BJT	: Bipolar Junction Transistor
BM	: Bare-Metal
BMP	: Bare-Metal Programming
CORDIC	: Coordinate Rotation Digital Computer
CPU	: Central Processing Unit
DAC	: Digital-to-Analog Converter
DC	: Direct Current
DSP	: Digital Signal Processing
EEPROM	: Electrically Erasable Programmable Read-Only Memory
ENOB	: Effective Number of Bits
FB:	: Full Bridge
FET	: Field Effect Transistor
FOC	: Field Oriented Control
FPU	: Floating-Point Unit
GaN	: Gallium Nitride
GTO	: Gate Turn-Off Thyristor
HAL	: Hardware Abstract Language
HB	: Half Bridge
HSE	: High-Speed External Clock
I/O	: Input/Outputs
IDE	: Integrated Development Environment
IGBT	: Insulated-Gate Bipolar Transistor
In-amp	: Instrumentation Amplifier
JTAG	: Joint Test Action Group

LDO : Linear Drop-Out Regulator

LED : Light Emitting Diode

LL : Low Level Language

LPF : Low-Pass Filter

LSE : Low-speed External Clock

MCU : Microcontroller Unit

MLI : Multi-level Inverter

MISO : Master In Slave Out

MKP : Metallized Polypropylene Film

MOSFET : Metal-Oxide-Semiconductor Field-Effect Transistors

MOSI : Master Out Slave In

MPU : Memory Protection Unit

MS : Mean Square

NPC : Neutral Point Clamped

NTC : Negative Temperature Coefficient Thermistors

Op-amp : Operational Amplifier

P : Proportional

PCB : Printed Circuit Board

PD : Proportional-Derivative

PI : Proportional-Integrator

PID : Proportional-Integrator-Derivative

PMSM : Permanent Magnet Synchronous Motor

PWM : Pulse Width Modulation

RCC : Reset and Clock Control

RMS : Root Mean Square

SGCT : Symmetrical Gate Commutated Thyristor

SiC : Silicon Carbide

SMD : Surface Mounted Device

SMPS : Switched Mode Power Supply

SNR	: Signal-to-Noise Ratio
SPI	: Serial Peripheral Interface
SPWM	: Sinusoidal Pulse Width Modulation
SRAM	: Static Random-Access Memory
SVPWM	: Space Vector Pulse Width Modulation
SWD	: Serial Wire Debug
THD	: Total Harmonic Distortion
T-NPC	: T-Type Neutral Point Clamped
TVS	: Transient Voltage Suppressor
UART	: Universal Asynchronous Receiver Transmitter
UPS	: Uninterruptible Power Supplies
USB	: Universal Serial Bus
VSI	: Voltage Source Inverter

LIST OF FIGURES

Figure 3.1. Simple Representation of VSI 7

Figure 3.2. Simple Representation of CSI 8

Figure 3.3. Half Bridge Topology 10

Figure 3.4. Half Bridge Output Voltage Before Output Filter 11

Figure 3.5. Full Bridge Topology 11

Figure 3.6. Three Phase Inverter 12

Figure 3.7. Three Phase Two Level Inverter 13

Figure 3.8. 3-Level Diode Clamped (Neutral Point Clamped) Inverter 15

Figure 3.9. (a) Neutral Point Clamped MLI (b) Active Neutral Point Clamped MLI 16

Figure 3.10. (a) 3-Phase VSI (b) SPWM Technique Implementation for 3-Phase VSI [24] 19

Figure 3.11. Clamping the Output Voltage in the Overmodulation Region [25] 20

Figure 3.12. (a) Reference Space Vector Generation (b) Space Vectors for a 2-Level Inverter ... 21

Figure 3.13. (a) Limitation of the Phase Output Voltage (b) Reduction of the Peak Phase Output Voltage with Third Harmonic Injection [25] 22

Figure 3.14. PID Control 23

Figure 3.15. Classical RMS Voltage Control of an Inverter 26

Figure 3.16. PI Controller Update Period for Classical RMS Voltage Control 27

Figure 3.17. RMS Voltage Regulation of an Inverter with Moving RMS Method 27

Figure 3.18. PI Controller Update Period RMS Voltage Control with Moving RMS Filter 28

Figure 3.19. ADC Oversampling Principle [28] 30

Figure 3.20. (a) ABC frame (b) αβ frame (c) dq Frame 32

Figure 3.21. A Common Vector Control Method for Motor Drive Applications 33

Figure 4.1. Overall Configuration of the Designed Three Phase Voltage Source Inverter 34

Figure 4.2. (a) L Type Line Filter (b) LC Type Line Filter (c) LCL Type Line Filter 38
Figure 4.3. LC Output Filter of a Three Phase Inverter 39
Figure 4.4. Designed Inductor for Output Filter 40
Figure 4.5. Phase Voltage Before and After Filtering 41
Figure 4.6. (a) Power Switch Selection Chart [37] (b) Power Switch Selection Trend Chart [38] 42
Figure 4.7. Excel Tool Developed for Power Switch Selection 43
Figure 4.8. Dead Time Setting for PWM Timer 44
Figure 4.9. Main Datasheet Parameters of IRFZ48N 45
Figure 4.10. Bipolar Totem Pole Gate Drive [42] 47
Figure 4.11. Bootstrap Gate Drive 48
Figure 4.12. Optocoupler Gate Drive [43] 49
Figure 4.13. Capacitive Gate Drive [42] 49
Figure 4.14. Transformer Gate Drive [42] 50
Figure 4.15. Gate Drive Technology of Selected IC: 1EDI40I12AF 51
Figure 4.16. Pin Layout and Pin Definitions of 1EDI40I12AF 51
Figure 4.17. Voltage Measurement Circuit Simulation 53
Figure 4.18. Power Tree of Designed Inverter Hardware 58
Figure 4.19. Stepdown Circuit Configuration of MP1584 59
Figure 4.20. (a) Inverting Circuit Configuration of NCP3063 (b) Equations for Inverting Operation 60
Figure 4.21. Recommended Circuit Configuration of B0515S-1W 61
Figure 4.22. (a) Circuit Configuration of LD117S50 (b) Typical Representation of an LDO 62
Figure 4.23. (a) Circuit Configuration of MIC5216 (b) Internal Block Diagram of MIC5216 63
Figure 4.24. Main Switching Behavior Simulation of the 3-Phase 2-Level Inverter 64

Figure 4.25. Three Phase Inverter Output Voltages 65

Figure 4.26. Gate Driver Simulation Model 66

Figure 4.27. Gate Driver Simulation Outputs 67

Figure 4.28. (a) Gate Driver Propagation Delay (b) Dead Time Sufficiency 67

Figure 4.29. (a) Group Delay of the Voltage Measurements (b) Exemplary Tolerance Analysis Result 69

Figure 4.30. Inverting Converter Spice Model 71

Figure 4.31. Inverting Converter Simulation Result 71

Figure 4.32. Inverter Control Simulation Matlab/Simulink Model 72

Figure 4.33. Matlab/Simulink Inverter Control Waveforms 72

Figure 4.34. Inverter Control Simulation Plecs Model 73

Figure 4.35. (a) Classical RMS Method PI Update Routine (b) Proposed RMS Method PI Update Routine 75

Figure 4.36. (a) Classical Method Recovery Time During Heavy to Light Load Transition (b) Proposed Method Recovery Time During Heavy to Light Load Transition (c) Classical Method Recovery Time During Light to Heavy Load Transition (d) Proposed Method Recovery Time During Light to Heavy Load Transition 76

Figure 4.37. (a) Classical Method Max Error During Heavy to Light Load Transition (b) Proposed Method Max Error During Heavy to Light Load Transition (c) Classical Method Max Error During Light to Heavy Load Transition (d) Proposed Method Max Error During Light to Heavy Load Transition 77

Figure 4.38. STM32G431RB Block Diagram 79

Figure 4.39. Physical Appearance of NUCLEOG431RB Development Board 81

Figure 4.40. Power Tree of NUCLEOG431RB 83

Figure 4.41. STM32G431RB Configuration for 3-Phase VSI Implementation 85

Figure 4.42. Sampling Time for ADC Modules of STM32G431RB 87

Figure 4.43. Printing ADC Read Value to Computer Screen 90

Figure 4.44. Flowchart of the Developed Novel Control Algorithm 92

Figure 4.45. GPIO Port Mode Register 95

Figure 4.46. Alternate Functions of GPIOA Module Pins 96

Figure 4.47. (a) GPIO Alternate Function Register (b) Instructions to Configure the AFRL[] Register 96

Figure 4.48. (a) TIMx_BDTR Register (b) Instructions to Configure the TIMx_BDTR Register 98

Figure 4.49. ADC Configuration Register-2 100

Figure 4.50. (a) Oversampling Shift Configuration Instructions (b) Oversampling Rate Configuration Instructions 101

Figure 4.51. ADC Regular Sequence Register-1 101

Figure 5.1. Schematic Symbol and Footprint Creation 110

Figure 5.2. PCB Rule Creation 111

Figure 5.3. Stack-up of the Designed PCB 111

Figure 5.4. (a) Top Layer (b) 2nd Layer (c) 3rd Layer (d) Bottom Layer 112

Figure 5.5. Three-Level Voltage Source Inverter Complete Schematic 113

Figure 5.6. 125W Three-Phase Two-Level Inverter Assembled PCB Hardware 113

Figure 5.7. (a) Schematic View of Input Filter Circuit (b) PCB View of Input Filter Circuit 114

Figure 5.8. (a) Voltage Measurement Circuit Power Side (b) Voltage Measurement Circuit MCU Side (c) PCB View of the Voltage Measurement Circuit 115

Figure 5.9. (a) Current Measurement Circuit Power Side (b) Voltage Measurement Circuit MCU Side (c) PCB View of the Current Measurement Circuit 116

Figure 5.10. Schematic View of Gate Driver Circuit 116

Figure 5.11. PCB View of Gate Driver Circuit 117

Figure 5.12. (a) Schematic View of Output Filter Circuit (b) PCB View of Output Filter Circuit 117

Figure 5.13. Schematic of Controller Board Connection Circuits.....118

Figure 5.14. (a) Main Board when Control Board Dissembled (b) Main Board when Control Board Assembled.....118

Figure 5.15. Schematic View of +25V to +5V Converter 119

Figure 5.16. (a) Physical View of MP1584 Module (b) PCB View of +25V to +5V Converter .119

Figure 5.17. (a) Schematic View of +5V to -5V Converter (b) PCB View of +5V to -5V Converter 120

Figure 5.18. (a) Schematic View of +12V to +5V Converter (b) PCB View of +12V to +5V Converter..... 120

Figure 5.19. (a) Schematic View of +5V to +3.3V Converter (b) PCB View of +5V to +3.3V Converter..... 121

Figure 5.20. (a) Front View of the Manufactured PCB (b) Rear View of the Manufactured PCB 121

Figure 6.1. Sampling Accuracy from STM32CubeIDE Interface 124

Figure 6.2. Output Voltage of Single Phase & Scaled RMS of Output Voltage 125

Figure 6.3. Voltage Outputs of Phase 1 and Phase 2..... 126

Figure 6.4 (a) Classical Method Heavy to Light Load Transition Max Error (b) Proposed Method Heavy to Light Load Transition Max Error (c) Classical Method Light to Heavy Load Transition Max Error (d) Proposed Method Light to Heavy Load Transition Max Error 129

Figure 6.5 (a) Classical Method Heavy to Light Load Transition Recovery Time (b) Proposed Method Heavy to Light Load Transition Recovery Time (c) Classical Method Light to Heavy Load Transition Recovery Time (d) Proposed Method Light to Heavy Load Transition Recovery Time 130

Figure 6.6 (a) Max Error with 8x Oversampling During Heavy to Light Load Transition (b) Recovery Time with 8x Oversampling During Heavy to Light Load Transition (c) Max Error with

8x Oversampling During Light to Heavy Load Transition (d) Recovery Time with 8x Oversampling During Light to Heavy Load Transition 132

Figure 6.7 Regulation Loss with Insufficient ADC Oversampling (2x) 133

LIST OF TABLES

Table 3.1. Half Bridge Topology Switching Sequences 10

Table 3.2. Full Bridge Topology Switching Sequences 12

Table 3.3. Switching States of 3-Phase 2-Level Inverter When Neutral is Grounded 14

Table 3.4. Switching States of 3-Phase 2-Level Inverter When Neutral is Left Float 14

Table 4.1. Electrical Characteristics of Designed Inverter 34

Table 4.2. Parameters of Designed Filter Inductor 41

Table 4.3. Electrical Parameters of 1EDI40I12AF 52

Table 4.4. Electrical Parameters of AD622 54

Table 4.5. Electrical Parameters of MCP1501-18E 55

Table 4.6. Electrical Parameters of SI8920 55

Table 4.7. Electrical Parameters of GO-20-SMS 57

Table 4.8. Electrical Parameters of LT1630 57

Table 4.9. Selected Connectors for Inverter Hardware 63

Table 4.10. Tolerance Analysis Results for Various Instantaneous Voltages 70

Table 4.11. Summary of the Plecs Simulations Dynamic Response Tests 78

Table 4.12. Pin Configurations of Main Functions 84

Table 6.1. Efficiency Table of the 3-Phase 2-Level Inverter 127

Table 6.2. Dynamic Response Comparison with Current Studies 134

1. INTRODUCTION

The most accepted and widely used circuit type in inverter applications, which we use in many areas from Uninterruptible Power Supplies (UPS) to biomedical devices in daily life, is 2-level inverters, which we can also call Voltage Source Inverter (VSI). Ideally, a good inverter system is expected to have excellent response in terms of both static and dynamic reaction. In other words, a good inverter system must both show excellent steady-state performance regardless of load change and react quickly during load changes. Together with that, it is also desirable that the instantaneous increases and decreases in the system loads or input voltage do not cause the system to collapse or lose the output regulation. Harmonic distortions in the output voltage can cause extra power loss and resonance. Of these two concepts, the focus of this thesis is on optimizing the dynamic response of the inverter.

The aim of this thesis is to develop a control method in order to minimize the deterioration in the output voltage regulations of 3-phase inverters under noisy environments and a working environment with sudden load changes and to provide rapid recovery during load change. By implementing this method on high-efficiency hardware designed with a power of 125 watts, it is also aimed to ensure that the thesis achieves its purpose. The static response of the inverter system can be possible by measuring the efficiency in the steady state, the amount of total harmonic distortion (THD), the noise and ripple on the output voltage.

On the other hand, it is possible to measure the quality of the dynamic response by observing two parameters. The first of these is the amount of overshoot or undershoot that occurs during the instantaneous output load change, and the other is the speed at which the system returns to the target RMS (Root Mean Square) value, in other words, the recovery time. Dynamic response is relatively difficult to measure and requires more capable measurement devices with higher bandwidth and sample rates, and more detailed study.

Within the scope of the thesis, necessary hardware design activities such as switch selection method, driver circuit design, voltage and current measurement circuit design, output filter designs, and PCB design and component soldering activities were carried out. In addition, the software that constitutes the control method is written in the STM32CubeIDE environment with bare-metal (BM) codes. It is aimed to compare the classical RMS control method in the literature

with the control method developed within the scope of the thesis on the designed and verified hardware and to reveal the contributions of the developed method. In the study, the traditional control method and the developed method in this thesis will be implemented using the same hardware and microprocessor, but only with different embedded software's.

In this study, first of all, sudden load changes were applied on the simulations selected in the simulation environment with the same component properties and system parameters as the real design, and the advantage of the developed method compared to the classical RMS method was tried to be observed. The developed method is based on the oversampling feature of Analog-to-Digital Converters (ADC) and the voltage measurements taken with high resolution and accuracy to an RMS filter that shifts with interrupt frequency.

The technique is also carried out by comparing the DC (Direct Current) value obtained from this filter with the target RMS value and applying correction through the PI controller. However, since it is not possible to reflect the oversampling properties of the ADCs in the simulation environment properly, the ADC oversampling contribution is explicitly shown during implementation. All the work done has been verified on the hardware prototype designed from scratch.

With the novel control method developed throughout this thesis, an instantaneous load change cannot disrupt the RMS output voltage regulation, even at the expense of some degradation of THD. In our thesis, it is aimed to increase the quality of the sample taken by applying the RMS control method in each sampling period for low-power and non-grid-connected systems and obtaining the samples with the ADC oversampling technique, thus ensuring the smoothness and continuity of the output voltage regulation in noisy environments. In the proposed method, instantaneous current information that creates RMS current is read in single conversion mode but with oversampling method, and this single measurement read again in each interrupt cycle acting as a continuous measurement with a robust low-pass filter (LPF) and increase the accuracy of the received data by the number of reading cycles. Since the proposed method requires calculation of RMS at each interrupt period, thus it needs a floating-point unit (FPU) controller to be implemented.

The simulation study of the proposed control method was conducted in the Plecs environment. The Plecs results will be analyzed in simulation section and later on the real implementation outputs

will be analyzed and compared, too. To summarize, the proposed novel control method offers benefits with a couple of aspects as below;

- Using the advantage of the ADC oversampling technique offered by the microprocessor to calculate more precise RMS
- Fast RMS calculation in each interrupt period with the sliding RMS method
- Applying RMS to the controller in each sampling (thus interrupt) period to achieve better output regulation
- Entirely bare-metal written codes to increase code quality and provide ease of debugging and decrease dependency to the microprocessor manufacturer.

To implement the proposed method, a comprehensive study has been carried out in terms of both hardware and software. Section 2 contains an overview of the literature. Briefly, various inverter types, studies on the control and regulation of these inverters, frequently used inverter modulation techniques, and articles with some comparative analyzes are mentioned. In Section 3, the concept of inverter in general, what the inverter types are, how their classification is, and frequently used modulation and control techniques are included.

Section 4 explains in detail all hardware, simulation and software development activities in this thesis. Section 4.1 describes in a comprehensive manner the design and selection of all hardware components, from determining basic hardware parameters, even up to connector selections. Section 4.2 shares information on Spice simulations used as a support for hardware design activities and Plecs and Simulink simulations that help develop the control technique that emphasizes the contribution of this thesis. Section 4.3 describes the development of novel control software with bare-metal coding. In this section, an overview of the control technique developed within the scope of the thesis is also made.

Section 5 describes the implementation phase in which the designed hardware and the embedded control software developed are combined. In Section 6, the experimental results are evaluated, and the contribution of the developed techniques is revealed with these results. Finally, Section 7 summarizes all the studies carried out within the scope of the thesis, discusses the results and offers suggestions for future improvements.

2. RELATED WORK

The basic theory of 2-level inverters was most comprehensively put forward by Mohan [1]. He classified 2-level inverters into single phase and 3-phase, and provided a comprehensive review of switching techniques, critical issues in inverter design such as blending, output ripple and switch utilization. Today, Mohan's equations and propositions are still used as the most reliable source for many power electronics designers.

An article [2] also examines various voltage source inverter types in terms of harmonics, operating modes, application areas and some other aspects. Among the topics of this article, the basics of DC to AC (Direct Current) conversion are also examined. The article discusses voltage source inverters in 3 stages as half-bridge and full-bridge single-phase inverters and three-phase inverters. In this article, a general evaluation of the techniques used in the modulation of inverter types has been made.

Many suggestions for control techniques have been put forward in various articles regarding single-phase and 3-phase inverter applications [3], [4], [5], [6], [7], [8].

This paper [3] presents a control method for the single-phase inverter that is looking to reduce the harmonic content of the output voltage, especially for the situation when the inverter feeds a crest-factor load. The method proposed uses a voltage controller followed by a current controller - both are implemented in the stationary reference frame. This approach regulates the fundamental harmonic of the voltage and is also used to attempt the elimination of some higher order harmonics.

In applications with no grid connection (stand-alone) like our thesis study, the method in which continuous RMS voltage control is applied at each fundamental frequency comes to the fore [4]. Most of the voltage source inverters use load voltage RMS controllers in order to regulate the amount of output voltage at load side. This paper presents a Simulink model of a single phase 1.5 KVA RMS controlled VSI inverter.

This article [5] has taken the regulation mitigation one step further by adding the effect of the output current. This article claims that, when combined with a Hall Effect sensor for current control, this technique keeps its parallel operation capability while increasing its noise immunity, which may lead to a reduction of the Total Harmonic Distortion (THD). Besides, as voltage regulation is designed in DC, a simple PI regulator can provide accurate voltage tracking.

Another commonly used control method in grid-connected inverters is vector control, inverter control using Park-Clarke transformations. This paper [6] presents Vector Control (Field Oriented Control (FOC)) simulation of a Permanent Magnet Synchronous Motor (PMSM). Mains, motor, and 2-level inverter were modeled in Matlab/Simulink environment. Gate signals for semiconductors were generated with the Sinusoidal Pulse Width Modulation (SPWM) method.

Our thesis includes the implementation of a 3-phase inverter with no grid connection. However, there are also grid-connected inverter applications in the literature. For instance, this paper [7] analyzes the stability problem of the grid-connected voltage-source inverter (VSI) with LC filters.

This research [8] also refers to these grid connected inverters. In this paper the issue of control strategies for single-stage photovoltaic (PV) inverter is addressed. A complete control structure for the single-phase PV system is also presented.

There are also some studies emphasizes the modulation techniques used with inverter designs. [9] and [10]. Amongst the various PWM (Pulse Width Modulation) techniques used in inverter models sinusoidal pulse width modulation technique is widely used because its circuit is simple and has rugged controlling schemes. By controlling width of the pulses of SPWM the desired output voltage and reduction of harmonics are achieved from the inverter [9]. Our proposed method in this study adopts the SPWM technique and can be used for single or 3-phase inverters.

SVPWM technique [10] is an alternative modulation technique over the SPVM technique. SVPWM is commonly used for obtaining sine waves with reduced THD in inverter applications. It is an advanced method for obtaining sine wave with reduced total harmonic distortion (THD). This method is increasing rapidly in recent years because it is easier to implement this method and it requires less computational time for calculation. However, in applications with half-bridge configuration where a neutral point exists, such as the application implemented herein, the SVPWM method cannot be applied.

With the method developed throughout our thesis, optimum output regulation by reducing the voltage error is aimed by reading the output voltage with ADC oversampling and applying the RMS of the read value to the controller in each interrupt cycle. Voltage error can be defined as the difference between the RMS value of the reference sine and the RMS value measured on the generated sine, and it is possible to reduce this error by adjusting the modulation index. In the classical approach, the RMS value of the samples taken during a fundamental period is given as

input to the system as a DC component, leading to some regulation problems where there is signal distortion on the measurements or unexpected changes in the load. Also, this method takes a long time for the output voltage to recover. A method in which the RMS value obtained during each interrupt period is given as input to the controller is examined in this article [11] to address a similar problem in UPS systems produced for grid-connected applications.

In paper [12], authors propose a zero-phase odd-harmonic repetitive control technique for conventional pulse-width modulation inverters and investigate the convergence of tracking errors. Paper [13] examines the stability of an LC-type voltage source inverter, highlighting the challenges in designing control parameters for ensuring system stability in traditional dual-loop control.

Article [14] introduces a single-loop Proportional (P) controller for a single-phase voltage source inverter circuit powering resistive, inductive, and nonlinear loads. The controller aims to produce a pure sinusoidal waveform and demonstrate superior dynamic response compared to double-loop voltage control. Furthermore, this article [15] explores classical RMS voltage regulation through simulation studies.

Apart from the above discussion of inverter drive and control technology, we have also researched how voltage and current measurement can affect inverter operation. Article [16] states that the accuracy of simple analog-to-digital conversion depends on both resolution of discretization in amplitude and resolution of discretization in time. For implementation convenience, high conversion accuracy is attained by refining the discretization in time using oversampling. This article mainly talks about how ADC oversampling increases the accuracy of ADC measurements in various applications. However, we could not find a study in the literature examining the effect of taking ADC readings using the oversampling feature on inverter circuit regulation. This led us to observe the ADC oversampling effect in the technique we developed. We examined the effect of ADC oversampling on inverter voltage regulation within the scope of this thesis.

In this section, we briefly mentioned the various sources that were taken as reference while developing the control technique that forms the basis of this thesis study and their contributions to the literature. In the next section (Section 3), we will comprehensively discuss the background of power electronics, which is the basis of this thesis.

3. MOTIVATION AND BACKGROUND

An inverter is a static device, and it can convert one form of electrical energy into other forms of electrical energy. The Inverter is a power electronics device which is used to convert fixed Direct Current (DC) into controlled Alternating Current (AC). Here, Controlled AC means controlling of two parameters of AC signal: frequency and amplitude.

In this section, various inverter structures in the literature are classified according to factors such as current source, number of phases and control topology, and a general evaluation is made.

3.1. Classification of Inverters According to Current Source

In general, inverters are divided into two according to direct current sources; current source inverters (CSI) and voltage source inverters [17].

3.1.1. Voltage source inverter

In a voltage source inverter, the input voltage remains constant throughout the operation. The VSI is connected to a DC voltage source with minimal or no impedance. To create a VSI, a DC source is paralleled with a large capacitor to maintain a constant input voltage. A simple representation for VSIs is given in Figure 3.1. In a VSI, the output voltage remains unaffected by the load, except from the waveform and magnitude of the load current dependency on the load impedance.

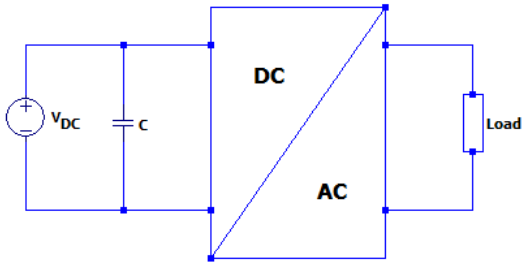


Figure 3.1. Simple Representation of VSI

Power semiconductor switching devices such as Power BJTs (Bipolar Junction Transistor), Power MOSFETs (Metal-oxide Semiconductor Field-effect Transistors), and IGBTS (Insulated-gate Bipolar Transistor) with self-commutation properties are used in the VSI circuitries.

VSI drives exhibit greater efficiency compared to CSI designs. This is attributed to the utilization of IGBT switching devices in VSI drives, which are more efficient than the Gate Turn-off Thyristor (GTO) or Silicon Gate Commutated Thyristor (SGCT) devices employed in CSI. The output current of the VSIs is regulated by altering the inverter's switching pattern or employing modulation techniques like PWM. To control the output voltage, the inverter adjusts the duty cycle of the switching signals.

VSIs can deliver varying output voltage while maintaining a constant frequency. By manipulating the pulse width, the RMS value of the output voltage can be managed. VSIs are commonly used for powering voltage-controlled loads such as motors, induction heaters, solar panel applications or specific types of AC drives. The input impedance of a VSI typically remains high, resulting in lower current draw from the input source. Additionally, VSIs might necessitate supplementary protection circuitry to avert damage from output short circuits such as usage of thermistor or similar components.

3.1.2. Current source inverter

A Current Source Inverter (CSI) operates by treating the input DC current as a fixed current source. It is supplied by an adjustable current source from a high-impedance DC voltage source. To convert a voltage source inverter into a CSI, a large series inductance is connected to maintain a constant input current as in Figure 3.2. In a CSI, the output current remains unaffected by the load, while the output voltage magnitude and waveform are influenced by the load impedance.

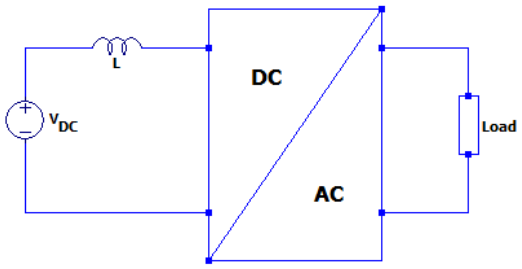


Figure 3.2. Simple Representation of CSI

In CSI, thyristors are commonly used as semiconductor switching devices. The output current of a CSI is regulated by adjusting the inverter's switching pattern or using modulation techniques. The inverter maintains a constant current output by adjusting the load impedance to regulate the

output voltage. The output voltage of a CSI depends on both the load impedance and the output current, varying accordingly with changes in load impedance. CSIs are typically employed for driving current-controlled loads such as electric arc furnaces or specific types of motor drives. Due to their low input impedance, CSIs draw more current from the input source. Additionally, CSIs possess inherent short circuit protection as they are regulated by current. The reason why CSIs are less preferred than VSIs is that they use inductors as storage elements and although inductors generally offer lower energy storage capacity than capacitors, they cause more conduction losses.

3.2. Classification of Inverters According to Phase Structure

Generally, inverters are divided into two as single-phase inverter and 3-phase inverter according to the number of phases.

3.2.1. Single phase inverters

The primary objective of a single-phase inverter is to produce an AC output waveform that closely resembles a sinusoidal waveform with minimal harmonic content. This is because the sinusoidal waveform is the typical form of AC electricity supplied by the utility grid. Single phase inverters are generally used by configuring them in two different ways as half bridge inverters and full bridge inverters, taking into account application requirements such as power, harmonic content, efficiency, etc.

Half bridge topology

In Half Bridge (HB) inverter topology, with a supply voltage of Vdc, two equal capacitors are linked in series across the DC input as it shown in Figure 3.3, positioned so that their junction rests at a mid-potential, holding a voltage approximately Vdc/2 across each capacitor. It's essential to use sufficiently large capacitances to maintain the potential at the mid-potential stable concerning the negative DC bus (the point at -Vdc/2 potential).

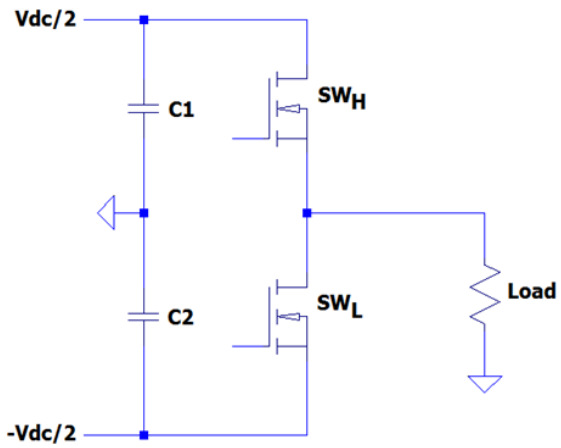


Figure 3.3. Half Bridge Topology

Table 3.1 shows a half bridge inverter switching sequence. While no output voltage can be produced when both switches are closed at the same time, the situation where both switches are open at the same time should be avoided due to the danger of short circuiting the source.

Table 3.1. Half Bridge Topology Switching Sequences

SWH	SWL	Vout
ON	ON	Failure
ON	OFF	+Vdc/2
OFF	ON	-Vdc/2
OFF	OFF	No Conduction

When SWH is activated, either SWH or its body diode conducts based on the output current direction, with the output current evenly distributed between the two capacitors. Similarly, when switch SWL is in its active state, either SWL or its body diode conducts based on the load current direction, again with the load current evenly distributed between the two capacitors. Consequently, capacitors C1 and C2 are effectively connected in parallel in the path of the output current. This also elucidates why the junction between the two capacitors stays at mid-potential. Since the output current must flow through the parallel combination of C1 and C2, the load current cannot possess a DC component in a steady state. Thus, these capacitors serve as DC blocking capacitors.

Assuming PWM switching, the output voltage waveform mirrors that shown in Figure 3.4 before filtering. Regardless of the switch states, the current between capacitors C1 and C2, both having equal and sizable values, divides evenly.

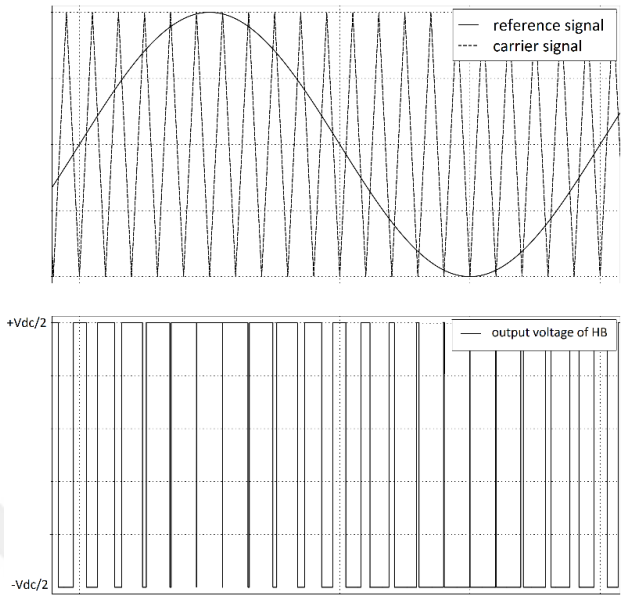


Figure 3.4. Half Bridge Output Voltage Before Output Filter

Full bridge topology

Full Bridge (FB) Inverter configuration comprises two one-leg inverters particularly in higher power ratings. Full Bridge configuration of a single-phase inverter is shown in Figure 3.5.

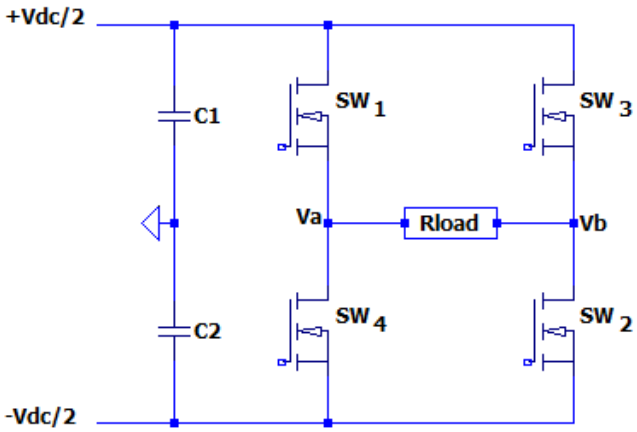


Figure 3.5. Full Bridge Topology

Switching states of a full bridge inverter is shown in Table 3.2. With the FB topology it is possible to utilize complete busbar voltage at an expense of using 2x power switches when compared with HB topology.

Table 3.2. Full Bridge Topology Switching Sequences

SW1	SW2	SW3	SW4	Va	Vb	Vab
ON	ON	OFF	OFF	+Vdc/2	-Vdc/2	+Vdc
ON	OFF	ON	OFF	+Vdc/2	+Vdc/2	0
OFF	OFF	ON	ON	-Vdc/2	+Vdc/2	-Vdc
OFF	ON	OFF	ON	-Vdc/2	-Vdc/2	0

In full bridge topology, when all switches are closed at the same time, no transmission occurs, while SW1-SW4 and SW2-SW3 switches being open at the same time is prohibited as it will cause the source to be short-circuited.

3.2.2. Three phase Inverters

In various applications like uninterruptible AC power supplies and AC motor drives, three-phase inverters are widely employed to power three-phase loads. The most utilized configuration for a three-phase inverter circuit comprises three legs, each dedicated to one phase, as depicted in Figure 3.6.

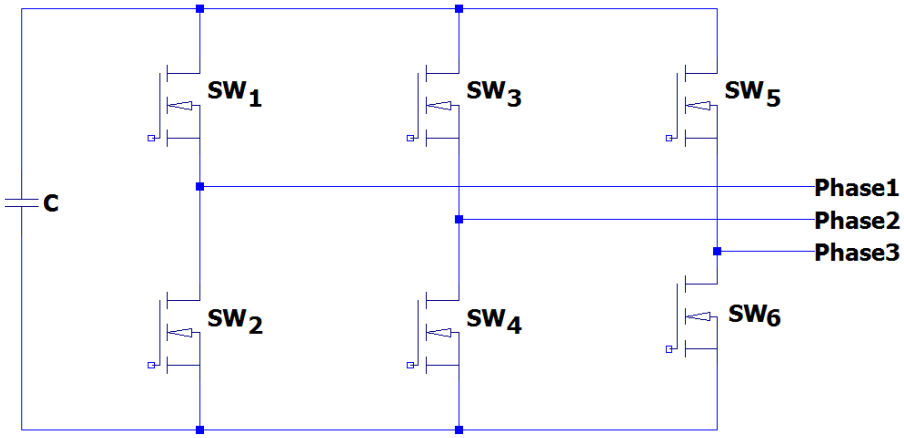


Figure 3.6. Three Phase Inverter

3.3. Classification of Inverters According to Number of Levels

A multilevel inverter offers advantages in system miniaturization and efficiency compared to a 2-level inverter. While a 2-level inverter controls the voltage waveform using 2 electric potentials, a 3-level inverter employs 3 electric potentials for control. When we have 8 switching states for a 2-

level 3-phase inverter, there are 27 switching states in a 3-level 3-phase inverter. A multilevel inverter produces an output waveform that more closely resembles a sinusoidal curve, leading to an anticipated reduction in the size of LC filters and switching losses. Although having more switching states improves the efficiency and the system miniaturization for multilevel inverters, it also brings with it an increase in complexity in terms of control as the number of levels increases.

3.3.1. Two level inverters

In this section, 3-phase 2-level inverters, which are the subject of this thesis, are examined. The diagram below shows a 3-phase 2-level inverter. 2-level inverters contain 6 switches and 8 switching states. Basically, they are formed by combining 3 half-bridge inverters as in Figure 3.7. Phase voltages are obtained by applying the PWM signals of the switches forming each phase, leaving a phase difference of 120°.

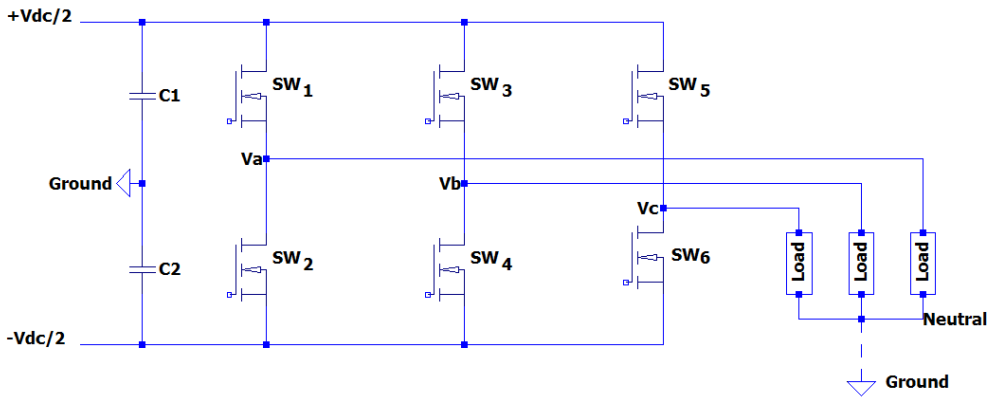


Figure 3.7. Three Phase Two Level Inverter

Switching states when neutral point is grounded

Two level inverters can be implemented in two ways by short-circuiting the neutral points with ground or leaving the neutral point as floating. Table 3.3 gives the phase-to-phase and phase-to-neutral voltages that occur when the neutral is shorted to ground in all possible switching states. Va, Vb and Vc nodes are the midpoints of each half bridge leg. In the table, Vab, Vbc, Vca points show phase-phase voltages of the circuit, and Va, Vb and Vc show phase-neutral (phase-ground) voltages. Note that these values are raw values before any filter is applied to the output.

Table 3.3. Switching States of 3-Phase 2-Level Inverter When Neutral is Grounded

SW1	SW2	SW3	Vab	Vbc	Vca	Va	Vb	Vc
ON	ON	ON	0	0	0	+Vdc/2	+Vdc/2	+Vdc/2
OFF	ON	ON	-Vdc	0	+Vdc	-Vdc/2	+Vdc/2	+Vdc/2
ON	OFF	ON	+Vdc	-Vdc	0	+Vdc/2	-Vdc/2	+Vdc/2
ON	ON	OFF	0	+Vdc	-Vdc	+Vdc/2	+Vdc/2	-Vdc/2
OFF	OFF	ON	0	-Vdc	+Vdc	-Vdc/2	-Vdc/2	+Vdc/2
ON	OFF	OFF	+Vdc	0	-Vdc	+Vdc/2	-Vdc/2	+Vdc/2
OFF	ON	OFF	-Vdc	+Vdc	0	-Vdc/2	+Vdc/2	-Vdc/2
OFF	OFF	OFF	0	0	0	-Vdc/2	-Vdc/2	-Vdc/2

Switching states when neutral point is left float

Table 3.4 gives the phase-to-phase and phase-to-neutral electrical potentials that occur when the neutral is left floating in all possible switching states.

Table 3.4. Switching States of 3-Phase 2-Level Inverter When Neutral is Left Float

SW1	SW2	SW3	Vab	Vbc	Vca	Van	Vbn	Vcn
ON	ON	ON	0	0	0	0	0	0
OFF	ON	ON	-Vdc	0	+Vdc	-2Vdc/3	+Vdc/3	+Vdc/3
ON	OFF	ON	+Vdc	-Vdc	0	+Vdc/3	-2Vdc/3	+Vdc/3
ON	ON	OFF	0	+Vdc	-Vdc	+Vdc/3	+Vdc/3	-2Vdc/3
OFF	OFF	ON	0	-Vdc	+Vdc	-Vdc/3	-Vdc/3	+2Vdc/3
ON	OFF	OFF	+Vdc	0	-Vdc	+2Vdc/3	-Vdc/3	-Vdc/3
OFF	ON	OFF	-Vdc	+Vdc	0	-Vdc/3	+2Vdc/3	-Vdc/3
OFF	OFF	OFF	0	0	0	0	0	0

Va, Vb and Vc nodes are the midpoints of each half bridge leg. In this table, Vab, Vbc, Vca points show phase-phase voltages of the circuit. Van, Vbn and Vcn show phase-neutral voltages. In this configuration the ground and the neutral are at different electrical potentials. These values are raw values before any filter is applied to the output.

3.3.2. Multilevel inverters

The multi-level inverter (MLI) is typically used as an alternative to traditional 2-level inverters in situations demanding high power and medium voltage. Generally, multilevel converters are aimed at generating substantial output power from medium voltage sources such as batteries, supercapacitors, and solar panels. Comprising numerous switches, the multi-level inverter places significant emphasis on the arrangement and angles of these switches.

Three three-types of multilevel inverters in general as; neutral point clamped multilevel inverter, flying capacitors multilevel inverter and cascaded h-bridge multilevel inverter [18]. The other topologies are the variations of those mentioned before.

The topology commonly used in multilevel inverters, the diode-clamped multilevel inverter, is illustrated in Figure 3.8.

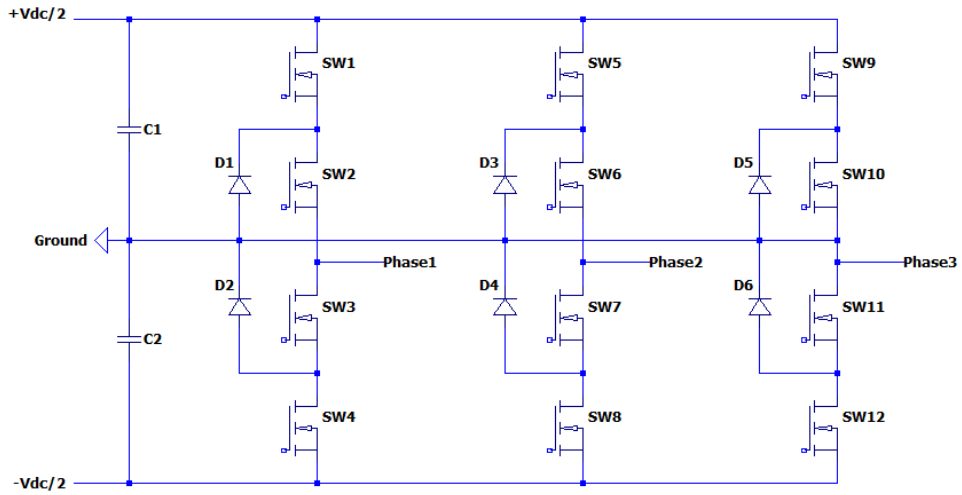


Figure 3.8. 3-Level Diode Clamped (Neutral Point Clamped) Inverter

Other common 3-level inverter types are shown in Figure 1 as in single phase representation for simplicity [19]. Since this thesis focuses on 2-level inverters, information about multilevel inverters will be kept brief.

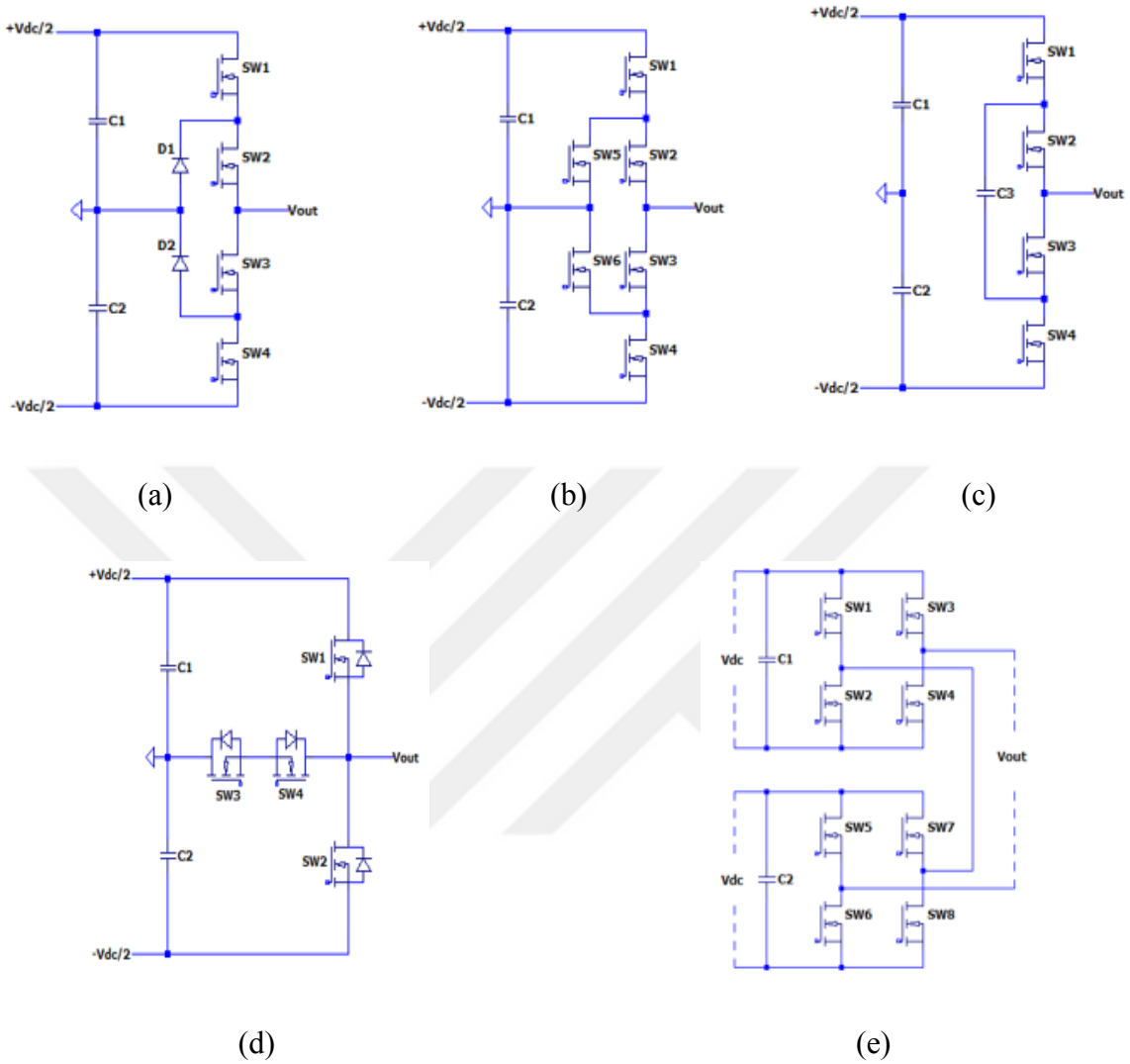


Figure 3.9. (a) Neutral Point Clamped MLI (b) Active Neutral Point Clamped MLI (c) Flying Capacitor MLI (d) T-Type Neutral Point Clamped MLI (e) Cascaded H-Bridge MLI

Diode clamped multilevel inverter

Fig 3.9 (a) shows a diode clamped inverter, which is also known as Neutral Point Clamped (NPC) inverter. Main idea behind this inverter is to utilize diodes to deliver multiple voltage levels across various phases to capacitor banks connected in series. Diodes serve to limit the voltage, thereby alleviating stress on other electrical components. The maximum output voltage equals half of the input DC voltage. Such inverters offer high efficiency due to the uniform employment of

fundamental frequency across all switching devices, presenting a straightforward approach to back-to-back power transfer systems.

Active NPC multilevel inverter

Fig 3.9 (b) shows an active neutral point clamped (A-NPC) inverter. The NPC inverter has the problem of uneven power loss distribution over the power devices. Active NPC topology is an improvement over diode clamped multilevel inverters at the expense of cost, adding power switches instead of clamp diodes. ANPC provides higher efficiency compared to NPC by reducing switching losses and offering better power balance. In addition, ANPC has a wider modulation range, allowing for better control of the output waveform compared to NPC. Also, it delivers lower harmonic distortion resulting in a cleaner and more stable output signal [20].

Flying capacitors multilevel inverter

Fig 3.9 (c) shows a flying capacitor inverter. The main idea behind this inverter is to utilize a series connection of capacitor-clamped switching cells, resembling the switching states found in the diode-clamped inverter. Capacitors transmit a limited voltage to electrical devices, resulting in an output voltage half that of the input DC voltage. Furthermore, it integrates switching redundancy within each phase to balance the flying capacitors.

T-Type NPC multilevel inverter

Fig 3.9 (d) shows a T-type neutral point clamped (T-NPC) inverter. Each leg of the T-NPC consists of four power devices: the T1 and T4 are connected positive and negative pole of DC power supply, respectively and one bidirectional switch (T2 and T3) which connect the middle point of dc-link to the load. The T-NPC topology is simpler than the conventional NPC. The T-NPC provides some advantages that are not available in conventional NPC such as: low conduction losses, fewer numbers of power switches and small size. The output voltage form is the same as diode clamped or capacitor clamped alternatives [21].

Cascaded H-Bridge multilevel inverter

Fig 3.9 (e) shows a cascaded H-bridge inverter. This topology consists of a series of power conversion cells and power can be easily scaled. It consists of H-bridge cells and each cell can

provide three different voltages like zero, positive DC, and negative DC voltages. Soft switching is possible by some of the new switching methods. The structure is simpler and cheaper than the flying capacitor and diode clamped versions of multi-level inverters. But these require a large number of isolated voltages to supply each cell. The output voltage is the sum of the voltages of each full bridge module.

In addition to 3-level inverters, 5,7 and more level inverter designs are also possible. Since it is not the subject of this thesis. For more information [22] and [23] can be referred.

3.4. Three-Phase VSI Modulation Techniques

Various techniques are employed to generate switching signals for driving the inverter power switches. The most common ones among these are Sinusoidal Pulse Width Modulation SPWM and Space Vector Pulse Width Modulation (SVPWM) techniques. In this section, detailed information will be provided about the SPWM technique applied in the modulation of the 3-phase 2-level inverter designed in this thesis. Additionally, a brief evaluation will be made for the other popular modulation technique, SVPWM. Before examining these two modulation techniques, it is necessary to address the concept of modulation index.

The modulation index

In a voltage source inverter the equation that gives the output voltage (V_{phase}) is given in Equation (3.1), '$V_{ph,peak}$' is the instantaneous maximum value of the output voltage and 'ω' is the angular frequency of the system, 't' is the time and 'ϕ' is the phase and can be neglected when considering single phase operations.

$$V_{phase} = V_{ph,peak} * \sin(\omega t + \phi) \tag{3.1}$$

Modulation index symbolized as 'm', characterizes the relative amplitude of the output voltage, with a maximum amplitude occurring at m=1, which corresponds to the peak of the sinusoidal voltage. If the modulation index is selected as 1, the peak value of the output sinusoidal voltage becomes half of the busbar voltage. Accordingly, when the equation given above is rearranged, Equation (3.2) is obtained. 'V_{dc}' stands for the busbar voltage.

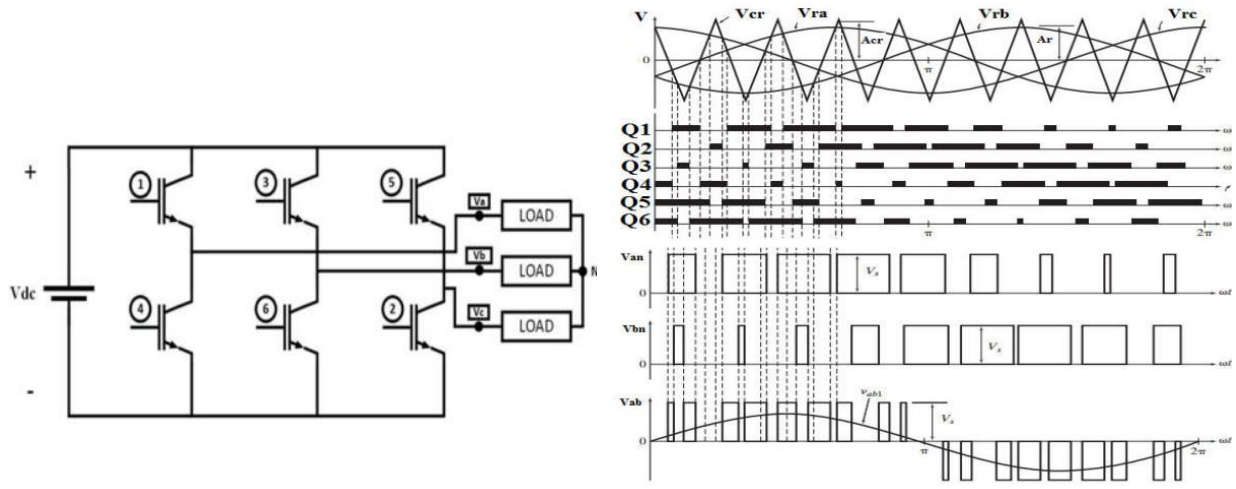
$$V_{phase} = m\left(\frac{V_{dc}}{2}\right)\sin(\omega t + \phi) \tag{3.2}$$

Finally, Equation (3.3) is used to express the modulation index in terms of the busbar voltage.

$$m = \left(\frac{V_{ph,peak}}{\frac{V_{dc}}{2}}\right) \tag{3.3}$$

3.4.1. Sinusoidal pulse width modulation

SPWM (Sinusoidal Pulse Width Modulation) is a commonly used modulation technique for generating waveforms. Using this technique, a sinusoidal AC signal can be generated using a DC source. SPWM achieves this control by modulating the pulse width, but unlike PWM, it modulates the pulse widths with a sinusoidal waveform. The image showing the creation of the PWM signal of each switch, and therefore each phase voltage, with the SPWM method in a 3-phase system is in Figure 3.10. Figure 3.10 (a) shows the switch positions on which these waveforms are based. Figure 3.10 (b) represents the waveforms generated by the SPWM technique. It is noteworthy that the reference signal must be in the same waveform as sine to be generated.



(a) (b)

Figure 3.10. (a) 3-Phase VSI (b) SPWM Technique Implementation for 3-Phase VSI [24]

Limitation of the DC bus utilization

We shared the equation in Equation (3.2) which shows how the phase voltage is formed according to the modulation index. The maximum amplitude of the sinusoidal voltage is obtained when m=1, and this amplitude is equal to $V_{dc}/2$. This situation indicates that in the SPWM technique, the maximum output voltage is limited to half of the busbar voltage even if the modulation index is selected as bigger than '1'. Consequently, this restriction leads to the introduction of low-order harmonics, which can pose challenges in filtration. When the voltage reference surpasses the capabilities of the DC bus, the converter enters the overmodulation region as illustrated in Figure 3.11.

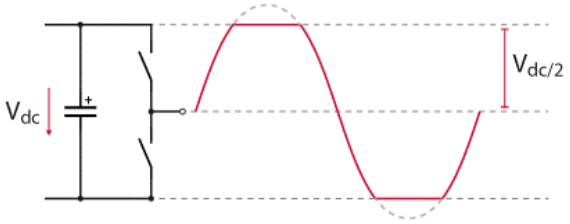


Figure 3.11. Clamping the Output Voltage in the Overmodulation Region [25]

Although the SPWM technique was applied in this thesis study, we will briefly touch on the SVPWM method in the next section, as it is another method frequently applied in various applications and in order to better understand our SPWM preference.

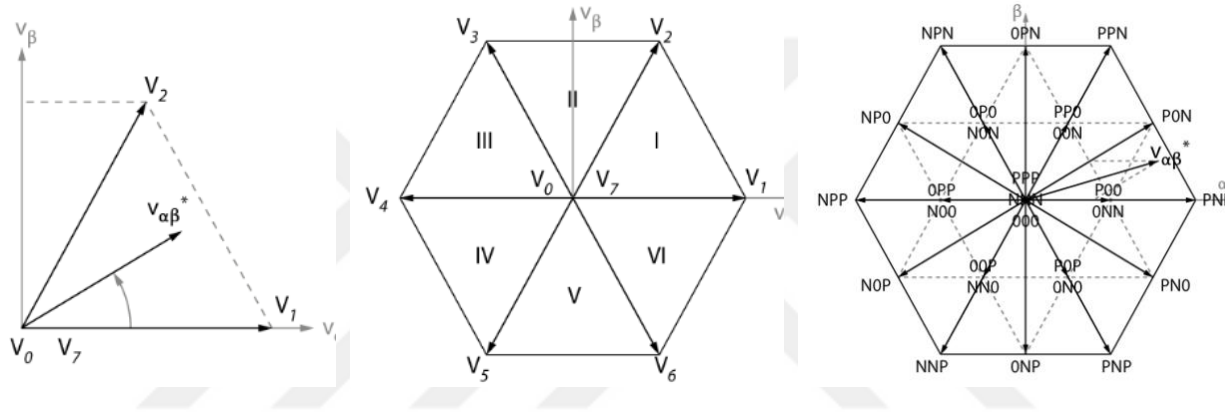
3.4.2. Space vector pulse width modulation method

SPWM and SVPWM both produce sinusoidal currents and compare a reference voltage signal to a triangular carrier for PWM generation [25]. The key difference is: SVPWM works on all three phases at once and therefore it is not possible to use it with single phase applications, whereas SPWM operates on a per-phase basis independently which makes it possible to operate either for single or 3-phase applications. SVPWM serves as a substitute for Carrier-Based modulation methodology. SVPWM operates within the Clarke reference frame (αβ), rather than the ABC framework utilized in carrier-based modulation [26].

Within the αβ frame, every inverter switching state is depicted by a space vector. Consequently, to prevent short-circuiting of the DC bus, the upper and lower switches within each leg must operate

in a complementary manner. This results in a total of eight potential switching states for the inverter.

Figure 3.12 (a) illustrates the generation of a reference space vector using active vectors V1 and V2, along with zero vectors V0 and V7. Additionally, Figure 3.12 (b) depicts the active and zero space vectors on the αβ plane for a 2-level inverter, divided into six sectors. Finally, for 3-level inverters the number of space vectors will become 27 making it even more complex as it is shown in Figure 3.12 (c).



(a) (b) (c)

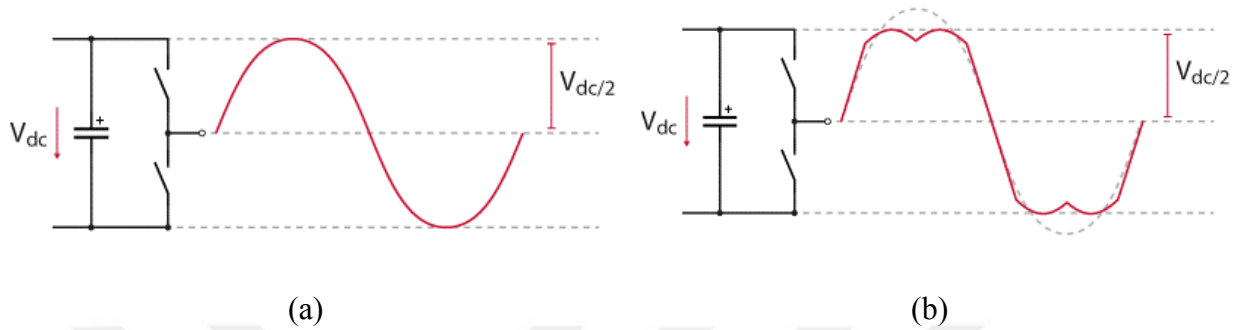
Figure 3.12. (a) Reference Space Vector Generation (b) Space Vectors for a 2-Level Inverter (c) Space Vectors for a 3-Level Inverter [26]

3.4.3. Third harmonic injection method

Using the SPWM technique results in the output voltage being limited to half of the DC bus voltage ($V_{dc}/2$). Consequently, this restriction leads to the introduction of low-order harmonics, which can pose challenges in filtration. When the voltage reference surpasses the capabilities of the DC bus, the converter enters the overmodulation region.

It can be shown that SVPWM is analogous to SPWM with minimum/maximum injection. The concept behind the minimum/maximum injection approach involves introducing a triangular zero-sequence component to the sinusoidal reference, thereby distorting the phase voltage references. This adjustment results in a reduction of the peak voltage compared to a purely sinusoidal reference, enabling the modulation index to reach up to $m=2/\sqrt{3} \approx 1.15$ before reaching the

constraints of the DC bus voltage. Figure 3.13 below is useful to understand how the benefit from the busbar is increased with this technique.



(a) (b)

Figure 3.13. (a) Limitation of the Phase Output Voltage (b) Reduction of the Peak Phase Output Voltage with Third Harmonic Injection [25]

3.4.4. Reasons for preference of SPWM

SVPWM is not a technique suitable for use in every circuit configuration in terms of harmonic content. The use of the SVPWM method with 2-level inverters is only possible under certain conditions. The first of these is that, due to the nature of SVPWM, the inverter must be 3-phase. Another important factor is the inverter circuit structure. In order for the circulating currents to cancel each other, the circuit structure must be a full bridge, or the neutral point of the 3-phase inverter must be left floating. Since the method developed and applied in this thesis study is suitable for both single phase and 3-phase inverter circuits, SPWM modulation technique was applied.

It is not recommended to use SVPWM in a system where the DC source is divided into two equal voltage potentials at its midpoint. In other words, while it is not practical to apply SVPWM in a system with a neutral connection in the form of -Vdc/2 0 +Vdc/2, SVPWM can be applied in a system with a busbar configured as 0 +Vdc without a neutral point.

In contrast to SPWM, SVPWM exhibits several drawbacks, such as increased complexity, elevated switching losses, and increased susceptibility to errors. These drawbacks stem from the need for more intricate calculations and algorithms to generate the switching signals, as well as the increased frequency of switching between sectors.

3.5. Three-Phase VSI Inverter Control Techniques

Different control techniques can be applied for inverters with different structures. In this section, we will mainly talk about the PI control technique, which is the method we use in this thesis, the classical RMS control technique and the innovative approach we brought to it, and the benefits of the ADC oversampling technique to the innovative approach. Finally, we will talk about vector control, which is another technique widely used in single phase and three phase inverter applications, and why vector control is not preferred in this thesis. It is a technical scalar control method based on RMS voltage control. We will refer to the technique applied based on Park and Clarke transforms as vector control. There are several aspects in which both techniques are advantageous. Both control techniques can be implemented with Proportional-Integrator (PI) and Proportional-Integrator-Derivative (PID) control structures, which are widely used in the literature.

3.5.1. PI control technique

In general, if we want the output of a system to follow a reference value through a feedback signal, we use PID control and its derivatives. In its simplest form, PID control takes two inputs, one measurement signal and one setpoint, and produces a control signal from them to regulate the system. Although the system where control is applied is mostly continuous time, PID control is applied digitally. In the Figure 3.14, PID control is modeled with its general structure. PID control application to the plant, that is, the reference system, is shown in the "z" domain.

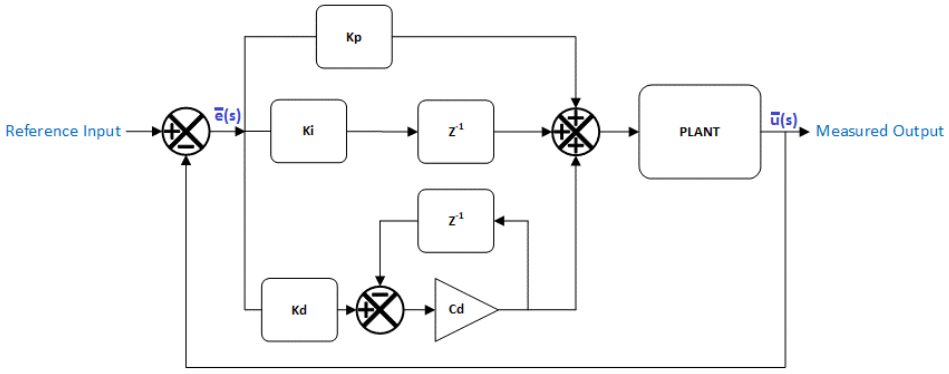


Figure 3.14. PID Control

Continuous time representation of PID in "S" domain is given in Equation (3.4).

$$G(s) = \frac{\hat{u}(s)}{\hat{e}(s)} = Kp + Ki\frac{1}{s} + Kd\frac{s}{s\tau + 1} \tag{3.4}$$

In this expression 'ê(s)' is the error term and 'û(s)' denotes the control term. 'Kp' is the proportional gain of the system, whereas 'Ki' is the integrator gain and 'Kd' is the derivative gain. Finally, 'τ' is the constant tau value. In order to apply the control method digitally, it is necessary to switch from continuous time to discrete time domain, that is, a difference equation must be obtained. There are various methods in the literature to obtain the difference equation, the most common of which is the Tustin transform (aka Bilinear Transform) [27].

We will not examine the Tustin transform or other transformation techniques from s-plane to z-plane in detail here. Since obtaining the difference equation from the 's' domain representation requires tons of substituting and rearranging, we instead give the difference equation that we calculated directly in Equation (3.5).

$$u[n] = p[n] + i[n] + d[n] \tag{3.5}$$

$$p[n] = Kp * e[n] \tag{3.6}$$

$$i[n] = Ki\frac{T}{2}(e[n] + e[n-1]) + i[n-1] \tag{3.7}$$

$$d[n] = 2\frac{Kd}{2\tau + T}(e[n] + e[n-1]) + \frac{2\tau - T}{2\tau + T}d[n-1] \tag{3.8}$$

In Equations (3.6), (3.7) and (3.8), for simplicity, the discrete time terms of proportional (p[n]), integrator (i[n]) and derivative (d[n]) are given separately in the discrete time domain, whereas the control term and error term are represented as u[n] and e[n]. At this stage, it is sufficient to know that this equation is obtained as a result of a series of mathematical operations after converting the PID controller equation from S domain to Z domain.

Different control techniques emerge by appropriately using one or more of the basic control effects (proportional, integrator and derivative). These control techniques are as follows.

- Proportional control (P)

- Proportional-integral control (PI)
- Proportional-derivative control (PD)
- Proportional-integral-derivative control (PID)

Proportional effect aims to correct a target proportional to the difference. While the difference gets near to zero, the applied correction approaches zero. Thus, the target value is never achieved. This is exactly where the integrator is effective. Integral effect attempts to mitigate this by effectively cumulating the error result from the proportional action to increase the correction factor. One trade of with the integrator term is, rather than stop acting when the target is reached, it attempts to drive the cumulative error to zero, resulting in an overshoot. However, considering the overall effect, a PI controller is much more effective and efficient than a P controller and is preferred in many applications.

On the other hand, derivative tuning attempts to minimize this overshoot by slowing the correction factor applied as the target is approached. Using derivative in the control might be dangerous due to it strongly amplifies the high frequency noise. Since it is not always possible to filter high-frequency noise perfectly in low-budget applications or to design a control block that is completely isolated from noise, we did not use PD or PID control in this thesis, instead we use PI control both with classical RMS method and the method we put forward in this thesis.

3.5.2. Classical RMS voltage regulation with PI control

Obtaining the RMS value in the classical RMS voltage control method is formulated [11] in Equation (3.9), where ‘i’ is the sequence index, ‘x[i]’ is the single measurement taken each interrupt period.

$$V_{out,RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{n} x^2[i]} \tag{3.9}$$

In this expression ‘N’ is the number of samples taken within the mains period which is equal to ‘To/Ts’. And here, ‘To’ is the fundamental period and ‘Ts’ is the sampling, that is the interrupt period.

According to this, the operation of the system is simply as follows; an initial modulation index value is given to the system and the RMS value of the output voltage is calculated from the samples collected at the end of each fundamental cycle. calculated value is a DC value and is given as input to the PI controller. Another PI controller input is the targeted RMS value. The difference between these two values is tuned according to the Kp and Ki values to produce a new reference amplitude value and this value is used to increase or decrease the modulation index. Accordingly, the PWM signal produced for the power switch is generated. A flow diagram of the classical RMS voltage control method is given in Figure 3.15.

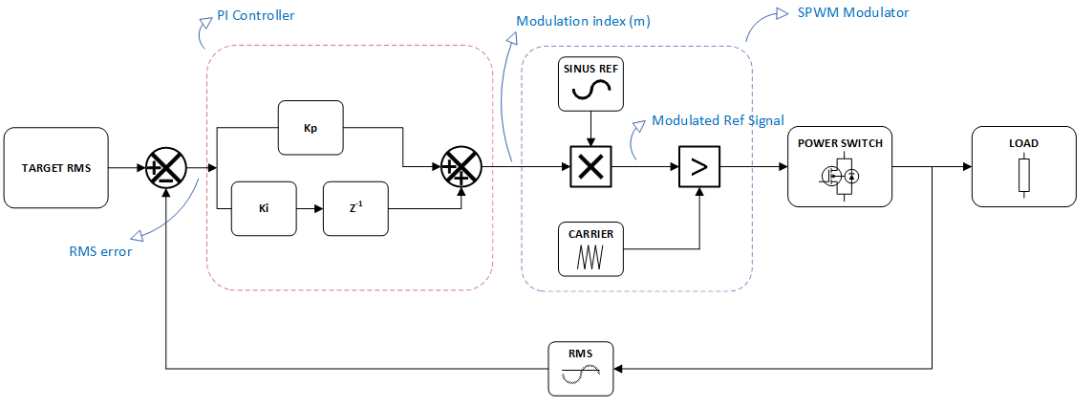


Figure 3.15. Classical RMS Voltage Control of an Inverter

If the output value is lower than the target value, the PI controller will try to increase the output by increasing the modulation index. If it is high, the modulation index will be reduced on the contrary. We have already observed in equation 5 how the modulation index is adjusted with the reference voltage. The success of this classical method depends on many factors such as voltage measurement quality, switching frequency and the degree of noise protection of the control circuit. Its biggest disadvantage is the slowness of the RMS measurement frequency and the update period of the PI controller output as it is shown in Figure 3.16.

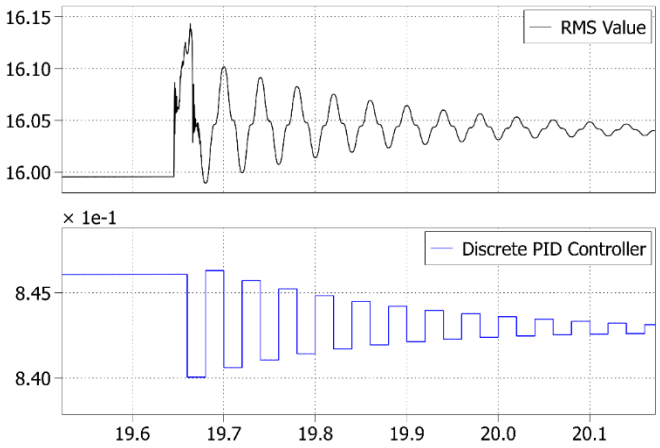


Figure 3.16. PI Controller Update Period for Classical RMS Voltage Control

Note that the PID controller is updated every 20ms, which is selected as the fundamental period of this thesis. We examine the technique we put forward to solve these problems in the next section.

3.5.3. RMS voltage regulation with moving RMS filter

Unlike the classical method, the control method applied in this study is a system with memory. The visually representative flow diagram of the technique we developed to solve this problem is shared in Figure 3.17.

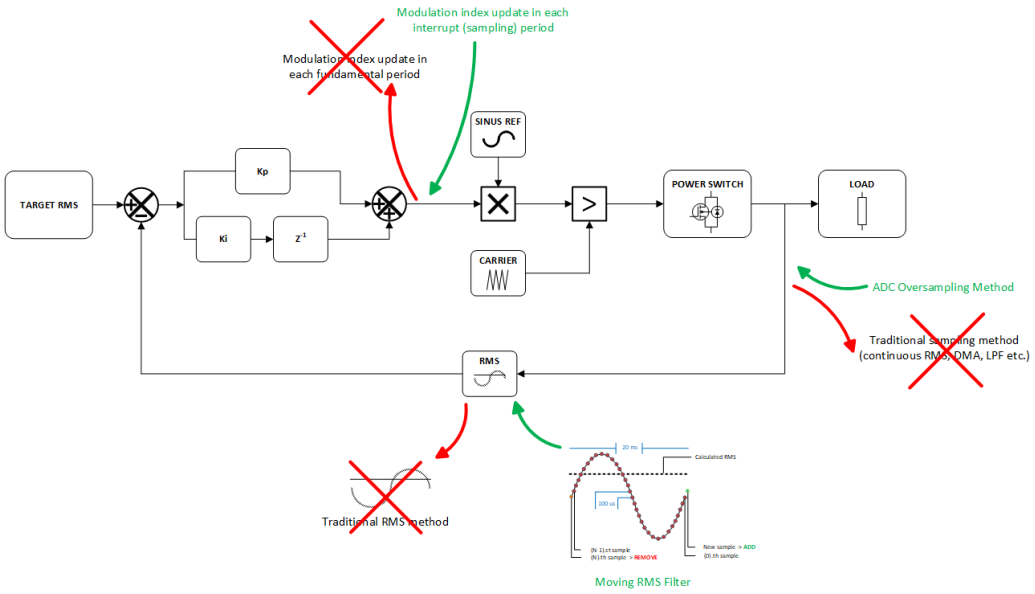


Figure 3.17. RMS Voltage Regulation of an Inverter with Moving RMS Method

When we call the current sample as 'p' and the last sample within a fundamental period as '(p+N)', the classical method adds n samples from $(p)^{th}$ sample up to $(p+N)^{th}$ sample into the RMS calculation. The $(p+N-1)^{th}$ sample has no effect on the calculated RMS.

In the novel method applied in this study, while including each new sample to the system with the sampling frequency in the RMS measurement, the effect of the $(p-n-1)^{th}$ sample or any sample that can be expressed as $(p-n-x)^{th}$ sample on the RMS voltage is still cumulatively carried on with the RMS value. Because the RMS value calculated with N samples is updated not in each fundamental period, but in each sampling period with the help of a sliding RMS filter. Accordingly, the RMS measurement input applied to the PI controller and therefore the update frequency of the PI is increased from the fundamental frequency to the interrupt frequency, providing a much more effective control. It is shown in Figure 3.18 how fast the PID output is updated and how the RMS value is updated accordingly in the method used in the study.

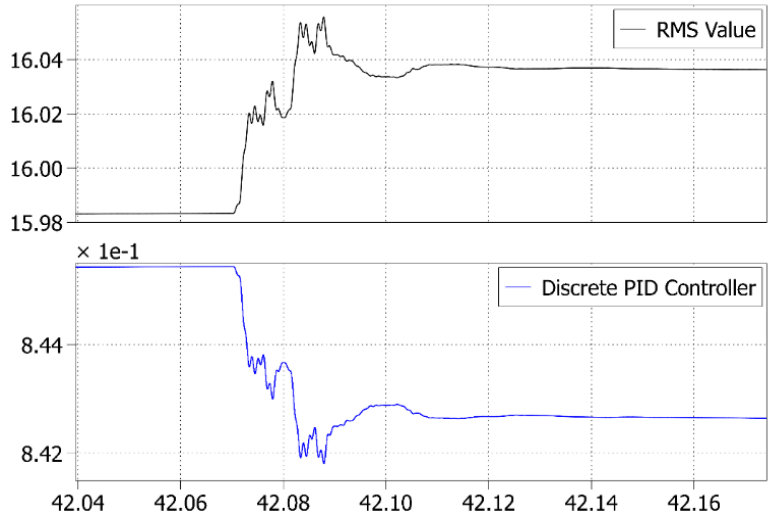


Figure 3.18. PI Controller Update Period RMS Voltage Control with Moving RMS Filter

Equation (3.10) is the formulization of the moving RMS filter structure applied in this thesis [11]

$$V_{out,RMS}[k] = \sqrt{\frac{1}{N}\sum_{i=k-N+1}^{k} x^2[i]} \tag{3.10}$$

Here, unlike 'i', which is the sequence index, 'k' refers to the sampling index. It will be even more clear when the flowchart in Section 4.3.5 examines how the sliding RMS structure is achieved by

using the RMS voltage values indexed for each 'k' value. Additionally, how this algorithm is implemented will be examined in detail in 4.3.6.3.

The improving effect of this on the dynamic response of the system is examined in detail in the implementation and results section. Figure 3.17 shows that the success of this method depends on the quality of the instantaneous voltage signal measured in each interrupt period, which we ensure by applying ADC oversampling. The ADC oversampling effect is briefly touched upon in the next section.

3.5.4. ADC oversampling method

Analog-to-digital converters (ADCs) provide the conversion of continuous analog signals into discrete digital sample values. In practical applications, ADC oversampling is employed to mitigate costs and enhance the performance characteristics of an ADC. The ADC oversampling functionality enables the hardware execution of the following functions: averaging, reduction of data rate, improvement of Signal-to-noise Ratio (SNR), and basic filtering.

By oversampling the signal at a factor of N, the dynamic range is proportionately augmented, given the multiplication of available values by N. The quantization of analog signal amplitudes into digital code words of finite length engenders quantization noise within the signal. This noise amplifies with decreasing word length, underscoring the direct relationship between quantization noise and word length. The mitigation of quantization noise requires augmenting the bit-depth within the ADC hardware architecture. Alternatively, software approaches can diminish quantization noise by oversampling the ADC and subsequently processing the digital signal.

The hardware implementation of ADC oversampling methodologies yields reductions in both Central Processing Unit (CPU) processing time and energy consumption, thereby contributing to an overall decrease in power utilization. This technique operates on a principle of trading bandwidth for precision, wherein components of lower resolution, and thus lower energy consumption, are subjected to oversampling to achieve enhanced effective number of bits (ENOB).

The principle of ADC oversampling lies upon operating on the premise that quantization noise is dispersed across a broader frequency spectrum, as illustrated in Figure 3.19.

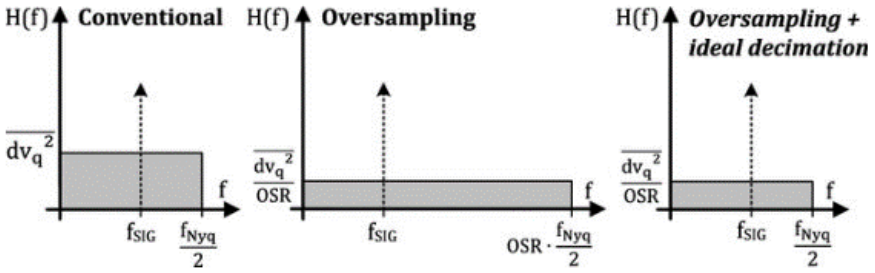


Figure 3.19. ADC Oversampling Principle [28]

The Nyquist theorem [29] states that for accurate reconstruction of an analog input signal, the signal must be sampled at a frequency which is greater than twice the maximum frequency component inherent in the input signal. Oversampling, as a strategy, involves sampling the analog input signal at rates surpassing the Nyquist frequency limit, subsequently filtering the samples, and thereafter reducing the sample rate through decimation. Employing this approach alleviates the stringent requirements imposed by anti-aliasing low-pass filters. When the SNR increases, the ADC effective number of bits increases. Sampling the input signal rates higher than the Nyquist frequency improves the SNR.

Theoretically we can reach up to 98dB of SNR with STM32G4 series microprocessors if all the signal quality conditions are met. The STM32G4 series CPUs offer a built-in oversampling unit to enhance the SNR. This integrated hardware oversampler enables improved analog performance while simultaneously reducing the associated computational workload from the CPU. The features of the hardware oversampler are outlined as follows:

- 16-bit data register
- Oversampling ratio adjustable from 2 to 256
- Programmable data shift up to 8-bit

The oversampling unit conducts data pre-processing to relieve the CPU of its workload. Capable of managing multiple conversions, it aggregates them into a single data with an expanded data width of up to 16 bits. It delivers a result in the following format (Equation (3.11), allowing for adjustment of parameters ‘R’ and ‘M’:

$$Result = \frac{1}{M} \sum_{n=0}^{n=R-1} Conversion(tn) \tag{3.11}$$

Here ‘R’ is the oversampling ratio and ‘M’ is the division coefficient consisting of a right bit shift up to 8 bits. The oversampler embedded in STM32G4 which we used in this thesis study, initiates by accumulating N samples. Subsequently, the sum undergoes a right shift by M bits. Post-shift, the engine retains the 16 least significant bits and rounds the resultant value to the nearest value based on the bits displaced during the shifting process before being finally transferred into the ADCs data register.

We will discuss in detail how we made these adjustments for the STM32G431RB, how we determined the system's ADC oversampling rate and shift amount, and all other implementation details of the ADC oversampling method in section 4.3.6.2.

3.5.5. Vectoral control with Park-Clarke transformations

We wanted to include vector control in our thesis as it is a control technique frequently applied in inverter applications. However, note that vector control was not applied in this thesis study. The applied control technique is to reinterpret the classical RMS voltage control and increase its dynamic benefit by combining an innovative measurement method, an innovative RMS acquisition method and a new error signal application method. The method developed in this thesis is not an alternative to vector control, that is, the control provided by Park-Clarke transformations [30], but a robustness-enhancing plugin that can be applied together with this method, as well.

Essentially, there are three reference frames considered in the implementation of vectoral control:

- The three-phase reference frame, known as the "ABC" frame, features 'Ia', 'Ib', and 'Ic', representing coplanar three-phase quantities situated at 120-degree angles to each other.
- The orthogonal stationary reference frame, referred to as the "αβ" frame, encompasses ‘Iα’ (aligned with the α axis) and ‘Iβ’ (aligned with the β axis), which are orthogonal and situated within the same plane as the three-phase reference frame.
- The orthogonal rotating reference frame (namely “dq” frame), where ‘Id’ is positioned at ‘θ’ angle (rotation angle) to the ‘α’ axis, and ‘Iq’ is perpendicular to ‘Id’ along the ‘q’ axis.

The graphical representation of these three frames is as follows in Figure 3.20.

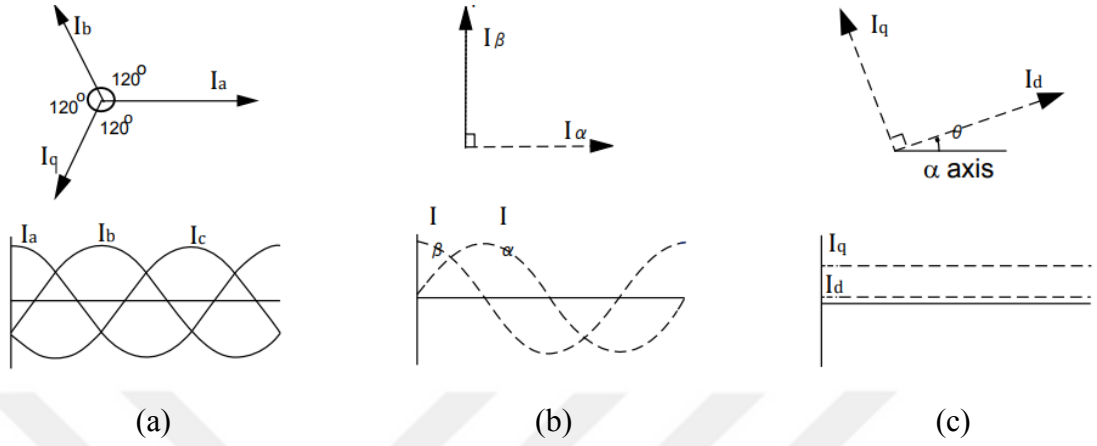


Figure 3.20. (a) ABC frame (b) αβ frame (c) dq Frame

Park-Clarke transformations are used to convert AC expressions in 3-phase systems into 2-phase AC expressions or DC and a relative component vector. Park-Clarke equations express the effective DC expression of a rotational vector in system equilibrium, that is, its value depending on the theta angle to the peak value of the system. Maximization of this value is used for torque control in motor drive systems.

In practice, it is possible to control 3-phase half bridge inverter systems with the Park-Clarke transformations method. However, Park-Clarke transformations may not be performed correctly in sinusoidal harmonic components. Park-Clarke transformations are used in applications where phase sequence accuracy needs to be determined, that is, for phase tracking.

In other words, the aim is to obtain maximum efficiency from torque. The most optimum angular alignment of the rotating field and static field vectors can be found with these transformations. And no alternative method has been described in the literature, so far. The general practice in motor drives is that the busbar is in the form of 0 to +Vdc. In other words, the system here is obtained by producing a phase-to-phase voltage close to 1.15 times the busbar, where two phases produce a voltage that is not related to sine relative to each other, but ultimately produces pure sine.

Figure 3.21 shows control of an AC 3-phase motor, that is a 3-phase inverter is controlled with vector control. As can be seen, each phase current of the 3-phase system is first taken from the ABC frame, Id and Iq components are obtained by applying first the Clarke and then the Park

transformations, respectively. These components are usually given as input to a PI controller, and the signals in the dq reference frame obtained from this PI controller are first converted to the αβ frame by applying the Inverse Park Transform, and then the PWM signals in the ABC frame are generated, usually with the SVPWM method.

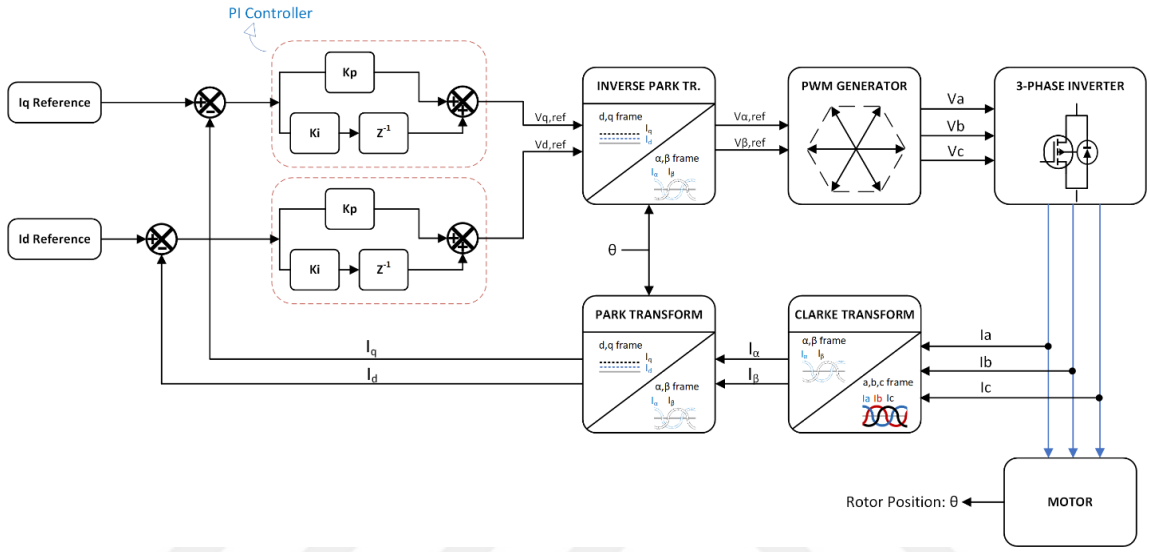


Figure 3.21. A Common Vector Control Method for Motor Drive Applications

Since it is beyond the scope of our thesis, more detailed information on the subject can be found in articles [30] and [31].

4. METHODOLOGY

In this section, the design and development phases of the hardware and the software in which the proposed method that is tested will be mentioned, the open and closed loop models produced for the mentioned topology and the theoretical and simulation studies carried out for implementation will be included, as well. The circuit configuration to implement the proposed control method is shown below Figure 4.1.

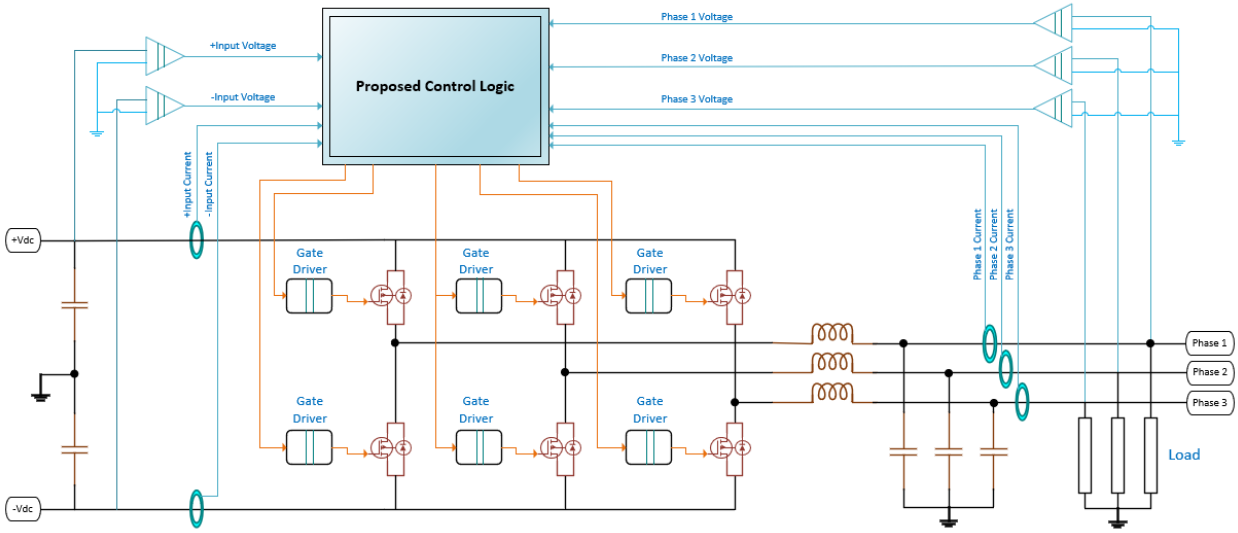


Figure 4.1. Overall Configuration of the Designed Three Phase Voltage Source Inverter

The electrical system parameters of this 3-phase voltage source inverter are given in Table 4.1.

Table 4.1. Electrical Characteristics of Designed Inverter

Input Power (P_{in})	125W
Input Voltage (V_{dc})	52VDC
Phase Output Voltage (V_{phase})	16VAC
Phase Output Current (I_{phase})	2,35A RMS
Target Efficiency (η)	>%90
Fundamental Frequency (F_o)	50Hz
Switching Frequency (F_{sw})	10KHz

Main components of the system that make up the inverter circuit can be summarized as follows;

- 2x TVS (Transient Voltage Suppressor), 2x NTC (Negative Temperature Coefficient) thermistors, 2x 5A fuses and 6x 220uF electrolytic DC link capacities for input filter and protection
- 5x instrumentation amplifiers (In-amp), 5x isolated amplifiers and 3x double channel general-purpose amplifiers (Op-amp) for positive and negative input voltage measurement circuits and phase voltage measurement circuits
- 5x current sensors and 3x double channel general-purpose amplifiers for positive and negative input current measurement circuits and phase current measurement circuits
- 3x 1.8V reference chip for various measurement circuitries
- 6x MOSFETs as switching elements
- 6x gate driver chips to drive power switches and 6x isolated power supplies to power-up the gate drivers
- 1x 12V to 5V LDO (Linear Drop-Out) regulator and 1x 5V to 3.3V LDO for MCU (microcontroller unit) power supplies
- 1x 24V to 5V DCDC converter and 1x 5V to -5V DCDC converter for measurement circuit op-amp supply voltages
- 3x toroidal inductance and 6x 25uF MKP capacitors for output LC filter
- 1x microprocessor board (NucleoG431RB) with necessary peripherals and I/Os (input/outputs) on which the control algorithm runs
- 1x isolator for UART (Universal Asynchronous Receiver Transmitter) communication
- 1x 3-channel industrial input connector, 3x 2-channel industrial output connectors, 1x 2-channel industrial microprocessor board power supply connector, 1x 4-channel male header for external UART communication and 2x 38-channel SMD (Surface Mounted Device) for controller board connection

Although it is not a must for proposed control techniques, in addition to the voltage sensors connected to each of the three phases, some current sensors are added to the each of the phases together with the current and voltages of the positive and negative busbars, in case of any future development or comparison necessity. (e.g. double loop control which is a method that also applies a fast current read value contribution to the controller to provide precise and low

THD sinusoidal output voltage). In any case, these sensors are being used to provide more precise electrical data such as efficiency of the hardware.

In the next section, the step-by-step design of the hardware that forms the basis of the thesis is explained. A table comparing the implementation results and theoretical results will also be given in the results section.

4.1. Hardware Design Steps

In the previous chapter, the electrical parameters of the hardware in which the problem to be solved within the scope of this thesis will be experimentally tested were determined. In this section, the design process of the mentioned hardware is explained step by step.

4.1.1. Determination of main system parameters

The first step for hardware design is theoretical calculations and verification of these theoretical calculations before moving on to the implementation phase. For this reason, first of all, an open-loop three-phase voltage source inverter model was established with LtSpice [32], which is thought to be more suitable for determining hardware needs than other software and whose libraries of ICs and other chips from many manufacturers can be easily found on the internet. The established LtSpice model is discussed in detail in Section 4.2.1.

Creating a proper working LTSpice model is the first step of the hardware implementation, thus the open loop control. Together with theoretical calculations by using the benefit of Excel [33], and with the help of LTSpice models and some of the online design tools the complete hardware design was made succesfully. All the open loop analysis within the project was made by using those materials. Theorical calculations conducted for determining the circuit components were supported via this LTSpice model throughout the hardware design process.

Based on the system electrical characteristics that we shared in Table 4.1; we determine the power circuit elements. To do that, first the expected output power per phase ($P_{out,Phase}$) is first calculated as per Equation (4.1). Accordingly, the output power for each phase is calculated as approximately 37,5W.

$$P_{out,Phase} = \left(\frac{Pin * \eta}{3}\right) \tag{4.1}$$

Calculation of RMS current for a phase with known output power is done as in Equation (4.2). The result of this calculation, 2.344A, is also approximately equal to the filter current.

$$I_{phase} = \left(\frac{P_{out,Phase}}{V_{phase}}\right) \tag{4.2}$$

In this design, the maximum ripple current (ΔI_L) to be allowed for the output filter design is determined as 10%. Accordingly, the maximum value of the output current ($I_{phase,peak}$) and therefore the filter current is calculated according to Equation (4.3).

$$I_{phase,peak} = \left(I_{phase} * \sqrt{2}\right) + \left(\frac{I_{phase} * 0.1}{2}\right) \tag{4.3}$$

In the next step, we calculate the maximum instantaneous switch current, which is an important parameter for switch selection. To do this, first, the maximum value of the DC voltage must be known, which in the SPWM method is equal to twice the value of the split busbar for each switch, that is, 52V. Accordingly, the maximum value of DC current is calculated as 3.29A with Equation (4.4).

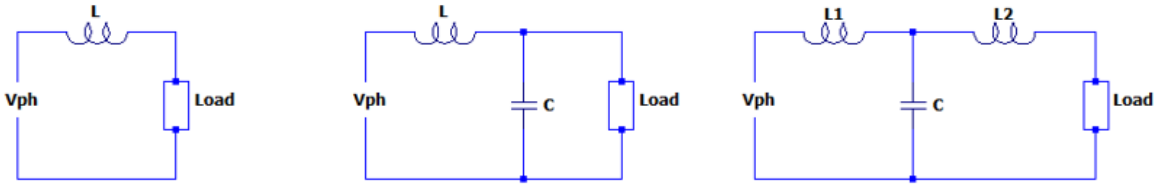
$$I_{dc,peak} = \left(\frac{3 * Vo * I_{phase,peak} * \cos(\phi)}{V_{dc}}\right) \tag{4.4}$$

For each phase, it is assumed that the currents lag behind the phase voltages by an angle 'ϕ'. In Equation (4.4), 'cos(Φ)' is taken as "1" to calculate the peak current. Switch current and maximum DC voltage are parameters in determining power switches, but these values are not sufficient for loss calculation. For this, RMS and average current values must also be known. In this thesis study, instead of calculating the values mathematically, RMS and average current values were taken from the LtSpice analysis whose other electrical results also consistent with our hand-made calculations. Power switch selection is addressed in more detail in section 4.1.3. The next section describes the output filter design.

4.1.2. Output filter design

The output filter reduces the current harmonics produced by the semiconductor elements used as power elements in the inverter circuit and is one of the most critical elements of the inverter circuit

design. In filters, the points where the power drops by half are called cut-off frequency (f_c). Low pass filters pass frequencies lower than the cutoff frequency. These are filters that prevent transmission by attenuating frequencies higher than the cut-off frequency. The most commonly used low-pass filter types in inverter designs, are shown in figure 4.2.



(a) (b) (c)

Figure 4.2. (a) L Type Line Filter (b) LC Type Line Filter (c) LCL Type Line Filter

The filter designed in this thesis study is an LC type. The following advantages of the LC filter can be considered as the reason for this decision.

- LC filter can be implemented with smaller inductance value than L filter
- Suppression ratio is higher than L filter
- Since the inductance is not very large, the voltage drop across it is small and the dynamic response time is relatively fast.
- Eliminates the inability of the L filter to suppress harmonics at the inverter's switching frequency
- Compared to the LCL filter, its complexity is lower, and it is more cost-effective because of consisting of fewer components.

Which one to choose between LC and LCL filters should be evaluated in different projects according to design requirements, cost factors and system performance criteria. The transfer function of the LC filter, which is determined as the output filter in this thesis, is given in Equation (4.5).

$$F(s) = \left(\frac{1}{1 + sL + s^2LC}\right) \tag{4.5}$$

The LC filter that is designed for a typical 3-phase inverter circuit is depicted in Figure 4.3.

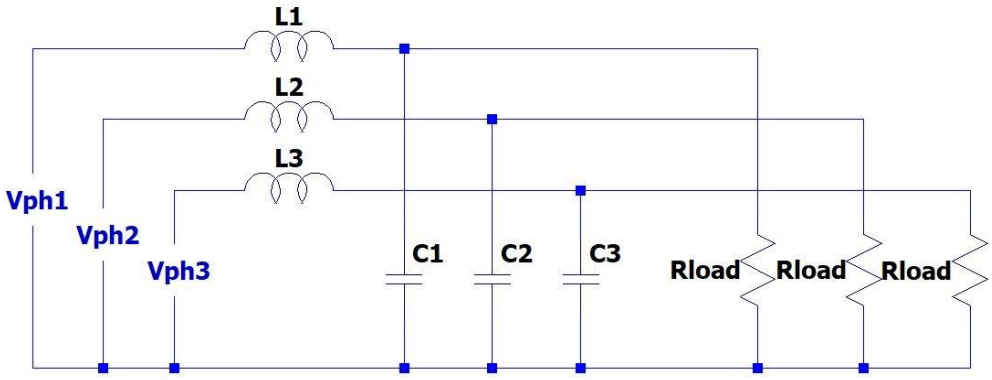


Figure 4.3. LC Output Filter of a Three Phase Inverter

Correctly determining the filter components and therefore the filter cut-off frequency (f_c) is an important design step. For LC filter design, it is first necessary to determine the resonance frequency (aka cutoff or corner frequency) of the filter. When determining the corner frequency, it is important to avoid resonance problems and achieve stable and trouble-free system dynamics. As stated in Equation (4.6) [34], general practice is to choose a value that is greater than 5 times the fundamental frequency (f_o) and less than one-fifth of the sampling (or switching) frequency (f_{sw}).

$$5fo < fc < 0.2fsw \tag{4.6}$$

Switching frequency is an important parameter in sizing filter elements. As the frequency increases, the same filter effect can be created with a smaller inductor and capacitance. However, in power electronics applications, switching losses are an important design criterion and these losses increase as the switching frequency increases. It is important to make a reasonable choice between the efficiency of the design and the sizing of the filter elements.

In this thesis study, the sampling frequency was determined as 10KHz and the fundamental frequency, which is the frequency of the output sinus voltage, was determined as 50Hz. In other words, according to the design criteria mentioned above, the cut-off frequency should be selected between 250Hz and 2KHz. In our design, the resonance frequency is determined as 1KHz, which is approximately the midpoint of these two values. The relationship between the cut-off frequency of the low-pass LC filter and the filter capacity and filter inductance is specified in Equation (4.7).

$$fc = \frac{1}{2 * \pi * \sqrt{\mathrm{LC}}} \tag{4.7}$$

An important parameter when choosing the filter inductor is the amount of ripple current (ΔI_L) that will occur on the inductor. In this study, the maximum ripple current allowed is limited to 10% of the maximum value of the load current. Since the positive and negative busbar voltage values are known, the minimum inductance value (L_{filter}) required to obtain a maximum 10% output ripple is calculated as 541uH according to Equation (4.8) [35].

$$L_{filter} = \frac{1}{8} * \frac{\frac{V_{dc}}{3}}{f_{sw} * \Delta I_L} \tag{4.8}$$

In order to eliminate possible calculation errors, the system input power was determined as 150W with margin, in the simulation studies. Accordingly, the maximum value of the load current was seen as approximately 3.9A in the simulation studies. Note that, in the previous section, the maximum value of the load current was calculated as 3.43A at 125W input power. Accordingly, the maximum allowed ripple current is determined as approximately 400mA.

The minimum required capacity (C_{filter}) at 1KHz cut-off frequency is calculated as 46.8uF in Equation (4.9), according to the minimum filter inductance value obtained from Equation (4.8).

$$C_{filter} = \frac{1}{L * (2 * \pi * fc)^2} \tag{4.9}$$

The filter inductance is designed to be greater than the minimum required inductance value by considering the inductance collapse factor under the load current. The filter inductor was designed with Magnetics' online design tool [36], taking into account the core availability factor. The physical appearance of the finalized filter inductance is shown in Fig 4.4.

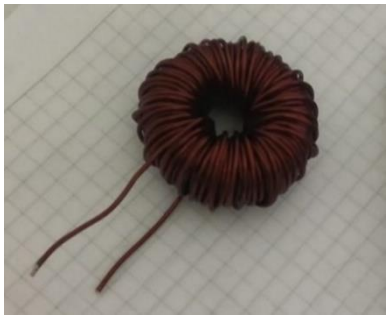


Figure 4.4. Designed Inductor for Output Filter

In addition, Table 4.2 contains the electrical and physical parameters of this designed inductor.

Table 4.2. Parameters of Designed Filter Inductor

Material	Kool Mu Max 79071
No load inductance	835uH
Full load inductance	670uH
Rms Current	2.8A
Peak Current	3.9A
Number of Turns	117
Wire Size	18 AWG (1mm)
Fill Factor	29.50%
Core Loss	2.07W
Copper Loss	1.14W

As it is known, the output LC filter increases the THD performance of the system by eliminating harmonics in the system. In other words, it ensures that the system output is closest to perfect sinus. To better explain this situation, it is sufficient to examine Figure 4.5. The upper waveform shows the phase voltage before the filter, that is the system output voltage without adding any LC filter. The lower waveform shows the phase voltage obtained after adding the output LC filter. It can be realized that the shape of the below waveform looks like almost a perfect sinusoidal.

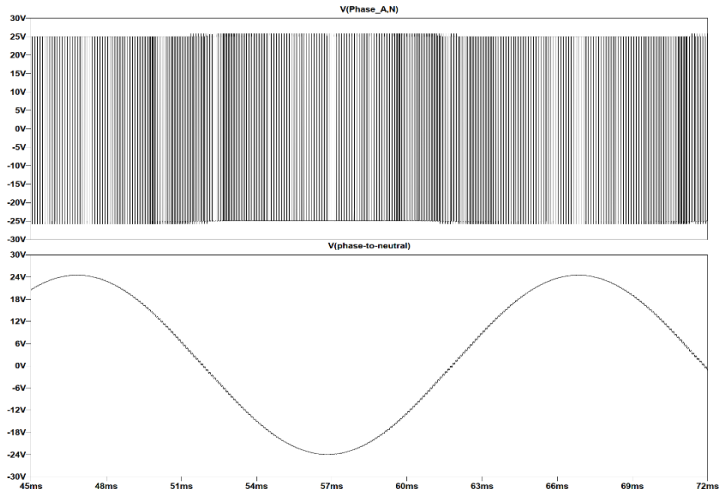
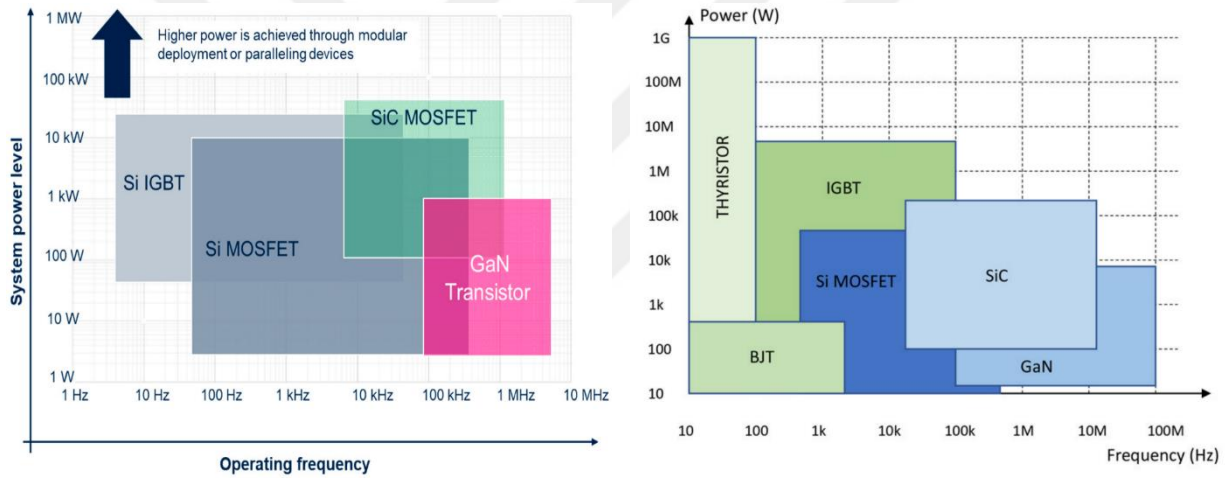


Figure 4.5. Phase Voltage Before and After Filtering

4.1.3. Power switch selection

One of the other most critical circuit components is undoubtedly power switches. There are many alternative materials on the market that can be used in power electronics projects, and each has different power and frequency ranges. Silicon MOSFETs, Silicon IGBTs, thyristors, SIC (Silicon Carbide) FETs (Field Effect Transistor) and Gallium Nitride (GaN) FETs can be listed among the most well-known power switches. The chart in Figure 4.6 (a) is important as it gives a general idea of how power switch selection is made today according to applications' power and frequency range [37]. This table is an important reference as it saves designers from the trouble of scanning all alternatives in detail for each project when choosing the required power switch.



(a) (b)

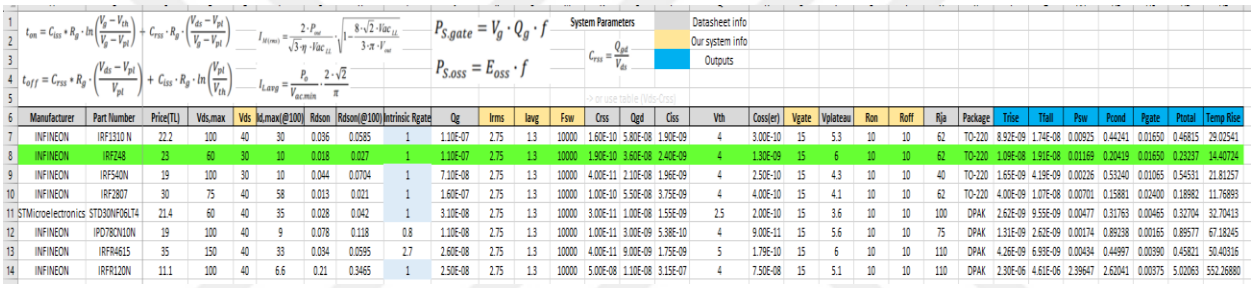
Figure 4.6. (a) Power Switch Selection Chart [37] (b) Power Switch Selection Trend Chart [38]

The other figure (Figure 4.6 (b)) suggests a rational forecast as to how switch choices might be shaped in the future based on power and switching frequency requirements [38].

The purpose of the hardware to be designed within the scope of this thesis is to create the necessary ecosystem to apply the developed control techniques rather than to apply high power. For this reason, taking benefit from the power switch comparison tables given, MOSFET switches, which are more cost-effective and meet the necessary electrical requirements since their power requirement is not high, were preferred over switches suitable for higher power or frequency level designs such as IGBT or SIC. However, in order to choose the optimum component in MOSFET

selection, technical loss and temperature rise calculations along with LtSpice analyzes were made in the Excel environment. Although theoretical calculations support that a DPAK (TO-252) package MOSFET can be a suitable option for the project if sufficient cooling is provided, TO-220 packaged MOSFET switches were preferred due to their ease of assembly and cooling and ease of measurement.

The equations for the calculations used when selecting the power switches are given below. We are not sharing the theoretical results after each equation for switch losses. Instead, below Figure 4.7 shows mentioned theoretical results automatically calculated by the Excel tool that is specifically developed within this thesis.



Manufacturer	Part Number	Price(TL)	Vds,max	Vds	Id,max(@100)	Rdson	Rdson(@100)	Intrinsic Rgate	Qg	Irms	Iavg	Fsw	Crss	Qgd	Ciss	Vth	Coss(er)	Vgate	Vplateau	Ron	Roff	Rja	Package	Trise	Tfall	Psw	Pcond	Pgate	Ptotal	Temp Rise
INFINEON	IRF1310 N	22.2	100	40	30	0.036	0.0585	1	1.10E-07	2.75	1.3	10000	1.60E-10	5.80E-08	1.90E-09	4	3.00E-10	15	5.3	10	10	62	TO-220	8.92E-09	1.74E-08	0.00925	0.44241	0.01650	0.46815	29.02541
INFINEON	IRF248	23	60	30	10	0.018	0.027	1	1.10E-07	2.75	1.3	10000	1.90E-10	3.60E-08	2.40E-09	4	1.30E-09	15	6	10	10	62	TO-220	1.09E-08	1.91E-08	0.01169	0.20419	0.01650	0.23237	14.40724
INFINEON	IRF540N	19	100	30	10	0.044	0.0704	1	7.10E-08	2.75	1.3	10000	4.00E-11	2.10E-08	1.96E-09	4	2.50E-10	15	4.3	10	10	40	TO-220	1.65E-09	4.19E-09	0.00226	0.53240	0.01065	0.54531	21.81257
INFINEON	IRF2807	30	75	40	58	0.013	0.021	1	1.60E-07	2.75	1.3	10000	1.00E-10	5.50E-08	3.75E-09	4	4.00E-10	15	4.1	10	10	62	TO-220	4.00E-09	1.07E-08	0.00701	0.15881	0.02400	0.18982	11.76893
STMicroelectronics	STD30NF06LT4	21.4	60	40	35	0.028	0.042	1	3.10E-08	2.75	1.3	10000	3.00E-11	1.00E-08	1.55E-09	2.5	2.00E-10	15	3.6	10	10	100	DPAK	2.62E-09	9.55E-09	0.00477	0.31763	0.00465	0.32704	32.70413
INFINEON	IPD78CN10N	19	100	40	9	0.078	0.118	0.8	1.10E-08	2.75	1.3	10000	1.00E-11	3.00E-09	5.38E-10	4	9.00E-11	15	5.6	10	10	75	DPAK	1.31E-09	2.62E-09	0.00174	0.89238	0.00165	0.89577	67.18245
INFINEON	IRFR4615	35	150	40	33	0.034	0.0595	2.7	2.60E-08	2.75	1.3	10000	4.00E-11	9.00E-09	1.75E-09	5	1.79E-10	15	6	10	10	110	DPAK	4.26E-09	6.93E-09	0.00434	0.44997	0.00390	0.45821	50.40316
INFINEON	IRFR120N	11.1	100	40	6.6	0.21	0.3465	1	2.50E-08	2.75	1.3	10000	5.00E-08	1.10E-08	3.15E-07	4	7.50E-08	15	5.1	10	10	110	DPAK	2.30E-06	4.61E-06	2.39647	2.62041	0.00575	5.02063	552.26880

Figure 4.7. Excel Tool Developed for Power Switch Selection

As it is known, switching losses occur when current flows between the drain and source ends of the MOSFET before the gate-source voltage completely decreases to zero during the opening and closing moments of the MOSFET. For this reason, the opening and closing durations of MOSFETs are important parameters in MOSFET selection. The formulas used to calculate the MOSFET opening and closing durations ('T_{on}' and 'T_{off}') are given in Equation (4.11) and (4.12).

$$T_{on} = C_{iss} * R_{gate} * ln\left(\frac{V_{gate}-V_{threshold}}{V_{gate}-V_{plateau}}\right) + C_{rss} * R_{gate} * \left(\frac{V_{ds}-V_{plateau}}{V_{gate}-V_{plateau}}\right) \quad (4.10)$$

$$T_{off} = C_{rss} * R_{gate} * \left(\frac{V_{ds}-V_{pl}}{V_{pl}}\right) + C_{iss} * R_{gate} * ln\left(\left(\frac{V_{pl}}{V_{th}}\right)\right) \quad (4.11)$$

In these equations, 'C_{iss}', 'C_{rss}', '$V_{threshold}$' '$V_{plateau}$' ara datasheet parameters that each MOSFET switch has. 'R_{gate}' is the gate driving resistance and 'V_{ds}' is the max voltage level that switch will see across its drain-source terminals during intended operation.

Along with the MOSFET opening and closing times, another important parameter is dead time. While the dead time should be long enough to prevent the switches in the same leg from being open at the same time and to prevent the risk of short circuit that may occur, a dead time chosen longer than necessary causes the busbar not to be utilized sufficiently and the phase voltage to be produced less than expected. Therefore, dead time is a critical parameter for ideally driving the switches and maximize the system efficiency. Simulation studies have shown that dead-times of 200ns and above are sufficient to prevent the switches from shoot through under various load conditions. However, the real operation does not always match the simulations exactly. The reason for this is parasitic effects caused by PCB (printed circuit board) layout and component tolerances. During the circuit implementation, dead-time was initialized with a relatively larger value such as 1us, and during the debug process, it was optimized as 320ns in the software environment, as seen in the code snippet in Figure 4.8.

```
420   TIM1->DIER=0;
421   SET_BIT(TIM1->DIER,TIM_DIER_UIE);     //  Update interrupt enable
422   //  SET_BIT(TIM1->DIER,TIM_DIER_BIE);   //  Break interrupt enable
423   SET_BIT(TIM1->CR1,TIM_CR1_ARPE);
424   //TIM1->BDTR |= 0x64;                   // Dead time = 1/164M*100 = 610ns.
425   //TIM1->BDTR |= 0x92;                   // Dead time = 2*(1/164M)*82 = 1us.
426   TIM1->BDTR |= 0x32;                   // Dead time = 1/164M*50 = 305ns.
```

Figure 4.8. Dead Time Setting for PWM Timer

The bare-metal programming (BMP) section will be discussed in more detail in Section 4.3.6.

For each MOSFET considered to be selected as a switch for the designed circuit, the MOSFET on-off times are calculated as previously explained. Then, knowing that the dead time is set to 305ns and using the electrical parameters specific to the inverter hardware and the datasheet information of the candidate MOSFETs, the switching losses are calculated as per following equations; Equation (4.12), (4.13) and (4.14).

$$P_{s,oss} = E_{oss} * F_{sw} \tag{4.12}$$

$$P_{sw} = \left(\frac{V_{ds} * I_{avg}}{T_{on} + T_{off}}\right) * P_{s,oss} \tag{4.13}$$

$$P_{s,gate} = V_{gate} * Q_g * F_{sw} \tag{4.14}$$

Here, '$P_{s,oss}$' is the loss caused by the paracitic capacitance ($C_{iss} + C_{rss}$) of the MOSFET, 'P_{sw}' is the total switching loss and '$P_{s,gate}$' is the loss caused by the gate charge.

Conduction loss 'P_{cond}' is calculated based on the maximum RMS current that is expected to flow continuously through the MOSFET and the worst-case MOSFET internal on resistance ($R_{ds,on}$). MOSFET conduction resistance is a dynamic parameter and is a function of temperature. Therefore, it would be useful to base the 'Rds,on' value at 100˚C as a general approach in order to make a safe design. The equation giving the conduction loss of the MOSFET is as Equation (4.15).

$$P_{cond} = I_{rms}^2 * R_{ds,on} \tag{4.15}$$

In the light of all the calculations detailed above, as the last step, the total MOSFET loss 'P_{total}' and the maximum heat increase ('Temp_rise') caused by this loss were calculated. With heat loss and natural cooling (without heatsink), the maximum heat increase expected to occur due to loss on the switch is calculated with the following equations (Equation (4.16) and (4.17)).

$$P_{total} = P_{sw} + P_{s,gate} + P_{cond} \tag{4.16}$$

$$Temp_rise = P_{total} * R_{th,ja} \tag{4.17}$$

Considering the availability of materials during the period in which the study was carried out, the component with part number IRFZ48N [54] from the Infineon manufacturer whose main datasheet parameters given in Figure 4.9 was chosen as the switching element of the inverter circuit.

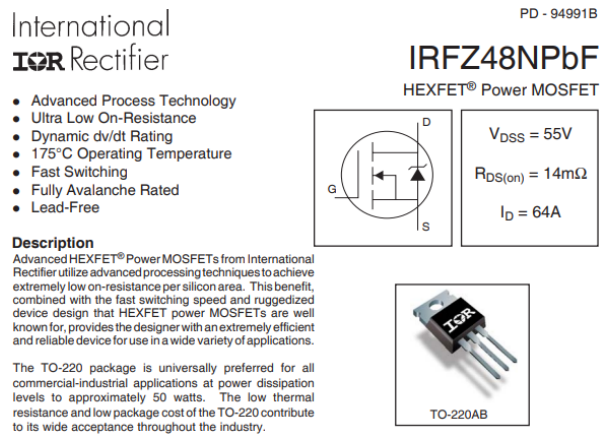


Figure 4.9. Main Datasheet Parameters of IRFZ48N

According to these calculations, the resulting maximum power dissipation of the IRFZ48N is calculated as 232mW and the temperature rise is 14.4 ˚C without the heatsink and only 2.3 ˚C with the heatsink as it shown in Figure 4.7, above. All the calculated loss values by this tool also coincide with the simulation results.

4.1.4. DC link capacity selection

DC link capacity is vital for the control technique implemented in this study to work efficiently. Although the DC bus inherently produces a stable and ripple-free voltage, noise will be reflected backwards to the source due to the switching of the power switches to produce the output sinusoidal voltage. The maximum allowable value for the amount of fluctuation that will be reflected back under full load and no-load output conditions will also determine the number and amount of DC link capacities. In 3-phase operation, since the return currents cancel each other in balanced load, there will be a significant decrease in the amount of ripple reflected on the busbar. As the load imbalance increases, the amount of ripple reflected at the input will also increase.

There are various studies for DC link capacity calculation. Some of these do not take into account unbalanced load conditions [39] or only calculate DC link capacity for single phase applications [40]. Therefore, calculations made according to different publications give different results. In this design, the following calculation method was adopted for the previously mentioned reasons. Accordingly, when the busbar ripple voltage ($\Delta V_{dc,max}$) is limited to 200mV, the minimum required busbar capacitor (C_{dclink}) for each busbar (positive and negative) is calculated according to Equation (4.17) [41].

$$C_{dclink} = \left(\frac{I_{in,RMS} * duty * (1 - duty)}{f_{sw} * \Delta V_{dc,max}}\right) \tag{4.17}$$

Here, '$I_{in,RMS}$' is the RMS current for each side of the busbar. We obtained this value as 3.29A from our LtSpice simulations. The worst situation in terms of ripple reflected at the input occurs when switching with 50% duty [39], [41]. That's why the duty is selected as 0.5 in above equation, which comes out a minimum DC link capacitor value of 410uF for each busbar.

In the light of these calculations, in order to leave margin for a safe design and to meet PCB design requirements for this inverter study, the DC link capacity was determined as 660uF (3x220uF) separately for the positive and negative buses.

4.1.5. Gate driver design and DCDC selection for gate driver supplies

In applications where semiconductor switching elements are operated at high frequency, switching operations must be carried out quickly and the selection of the driving technique applied to perform these operations is important for optimum switching. Since the switch type chosen in the thesis is MOSFET, information is given about the most common techniques applied with MOSFETs. These techniques can be summarized as bootstrap, optocoupler drive, transformer drive, totem pole drive, and finally drive using capacitive gate drivers. While bootstrap and totem pole techniques are non-insulated driving techniques, the other techniques we have mentioned are isolated driving techniques.

4.1.5.1. Bipolar totem pole gate drive

The bipolar totem pole technique which is shown in Figure 4.10 is one of the most popular and cost-effective driver circuits for driving MOSFETs. In a totem pole configuration, two drivers are used: a high-side driver and a low-side driver. The high-side driver is typically responsible for pulling the gate voltage high, while the low-side driver is responsible for pulling it low. They are formed by using one n-channel MOSFET on the low side and one p-channel MOSFET on the high side.

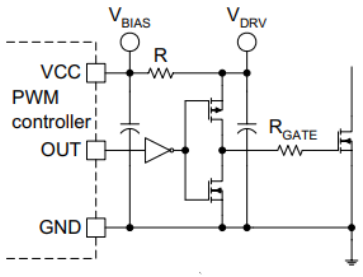


Figure 4.10. Bipolar Totem Pole Gate Drive [42]

The most important advantages of the totem pole driver structure are as follows: It can be used as a current amplifier in cases where the gate driver IC (integrated circuit) cannot provide sufficient

gate current, has a simple structure and is cheap in cost. A disadvantage of this structure is that the PWM signal must be inverted before reaching the switches.

4.1.5.2. Bootstrap gate drive

Bootstrap gate drive which is illustrated in Figure 4.11 is a technique used to efficiently drive the high-side MOSFET in inverter applications, such as in half-bridge or full-bridge configurations. It is particularly useful when the high-side switch needs to be driven by a voltage higher than the available power supply voltage.

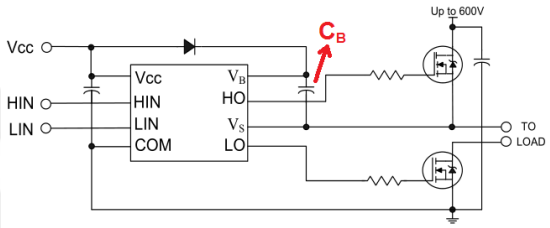


Figure 4.11. Bootstrap Gate Drive

In a bootstrap gate drive circuit, a bootstrap capacitor is charged to a voltage higher than the power supply voltage with the help of the bootstrap diode during the low side switch's “ON” state. Then, during the high-side switch's “ON” state, the voltage across this capacitor is used to drive the gate of the high-side MOSFET. The advantages of bootstrap technique can be listed as follows; eliminating the need for an additional isolated power supply for the high-side gate driver and simplifying the gate drive circuitry compared to other methods requiring isolated power supplies. However, one of the main disadvantages of this technique is the voltage droop over time due to charge depletion in the bootstrap capacitor, which can affect the reliability of the gate drive circuit in certain applications.

4.1.5.3. Optocoupler gate drive

Gate driving with optocouplers which is illustrated in Figure 4.12 involves using optocouplers, also known as opto-isolators, to provide electrical isolation between the control circuitry and the power devices. In a gate driving application, the PWM signal generated by the MCU is connected to the LED side of the optocoupler. When the control circuitry applies a PWM signal to the LED, it emits light, which is transmitted across the optical barrier and activates the photodetector on the

output side. Because there is no direct electrical connection between the input and output sides of the optocoupler, any voltage or current differences between them are isolated.

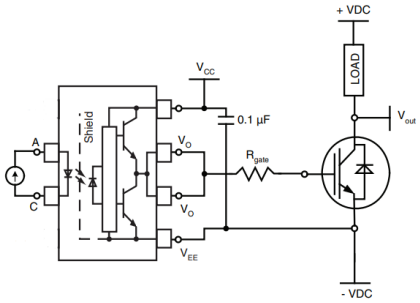


Figure 4.12. Optocoupler Gate Drive [43]

Gate driving with optocouplers offers several advantages such as electrical isolation and noise immunity. Although they can operate in some frequencies from low to a few KHz, gate driving with optocouplers also has limitations, such as compatibility with limited switching frequencies and higher costs compared to direct gate driving methods. Also, other disadvantages of these techniques can be mentioned as propagation delays and need of an isolated power supply.

4.1.5.4 Capacitive gate drive

A capacitive gate driver which is illustrated in Figure 4.13 is a type of gate driver circuit that uses capacitive coupling to achieve galvanic isolation and used to drive the gates of power semiconductor devices in various electronic applications, particularly in high-frequency power converters. Capacitive gate drivers utilize capacitors to transfer control signals from the input stage to the output stage. Capacitive gate drivers may include various protection and control features such as overcurrent protection, overvoltage protection, and temperature monitoring.

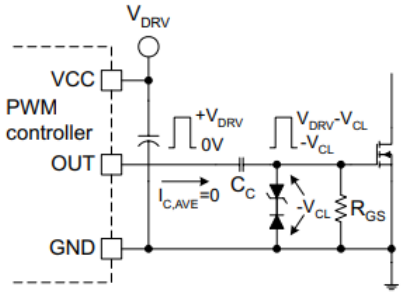


Figure 4.13. Capacitive Gate Drive [42]

Capacitive gate drivers typically offer lower isolation voltages compared to gate drivers with transformers, which may limit their use in applications requiring high isolation voltages. Their main benefits can be summarized as their compactness and suitability for high-frequency applications.

4.1.5.5 Transformer gate drive

The basic operation of the gate drive with a transformer involves using the primary winding of the transformer to receive control signals from the low-voltage circuitry, and then these signals are transformed to the required level on the secondary winding to drive the gate of the semiconductor device as it is illustrated in Figure 4.14. This allows for efficient switching while maintaining electrical isolation between the control and power circuits, enhancing safety and performance in many applications, particularly in high-power and high-voltage environments.

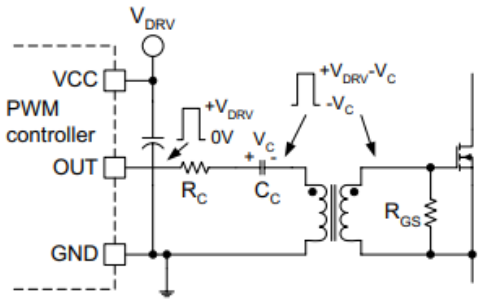


Figure 4.14. Transformer Gate Drive [42]

Depending on the frequency range and power rating considered, transformers can be designed to be highly efficient. The gate drive transformer carries a very small average power. In this technique, high peak currents enable the MOSFET to turn on and off. Together with it has the isolation and noise immunity advantages like the optocoupler drive technique, other advantages of using a transformer for the gate driver are as follows: no isolated DCDC converter necessity, no propagation delay time in the transformer to move signals from the primary side to the secondary side, allowing voltage level shifting by its nature. However, its disadvantages are too many to deny as it is an old technique compared to the other techniques. These are compatibility with only AC signals, design complexity and cost, size and weight drawback, efficiency degradation due to transformer losses and limitation of operating frequency.

Coreless Transformer Gate Drive

Coreless transformer gate drivers are a type of gate driver circuit commonly used in high-frequency power converters. They utilize a coreless transformer as the main component for galvanic isolation between the control circuitry and the MOSFETs. Unlike traditional transformers, which have a magnetic core, coreless transformers have windings without a magnetic core which allows them for high-frequency operation and reduces the size and weight of the transformer.

In this thesis study, we used the 1EDI40I12AF [44] as gate driver, which is an integrated circuit that utilizes the following advantages of the coreless transformer structures with a SOIC-8 package, whose internal structure is shown as in Figure 4.15 high efficiency, compact size, and improved reliability compared to traditional gate driver circuits with magnetic core transformers.

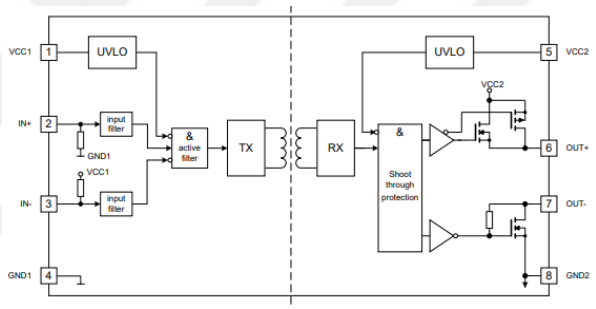
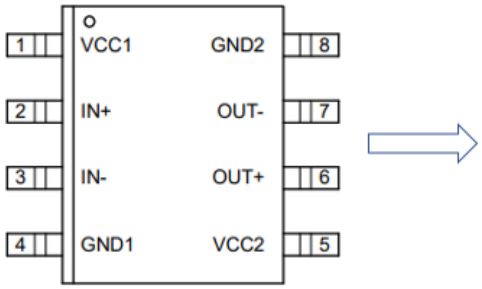


Figure 4.15. Gate Drive Technology of Selected IC: 1EDI40I12AF

Pin layout and definitions of the gate driver IC used in the designed inverter hardware are given in Figure 4.16.



Pin No.	Name	Function
1	*VCC1*	Positive logic supply
2	*IN+*	Non-inverted driver input (active high)
3	*IN-*	Inverted driver input (active low)
4	*GND1*	Logic ground
5	*VCC2*	Positive power supply output side
6	*OUT+*	Driver source output
7	*OUT-*	Driver sink output
8	*GND2*	Power ground

Figure 4.16. Pin Layout and Pin Definitions of 1EDI40I12AF

Main technical parameters for 1EDI40I12AF can be seen in Table 4.3.

Table 4.3. Electrical Parameters of 1EDI40I12AF

Parameter	Symbol	Values		Unit	Note/Test Condition
		Min	Max		
Sink / Source Current	Io+/Io-	-	7.5	A	
Propagation Delay	t_{PD}	270	330	ns	
Power supply output side	V_{VCC2}	13	35	V	With respect to GND2
Power supply input side	V_{VCC1}	3.1	17	V	-
Logic input voltages (IN+,IN-)	$V_{LogicIN}$	-0.3	17	V	-
Switching frequency	f_{sw}	-	1.0	MHz	Do not exceed max power dissipation
Ambient temperature	T_A	-40	125	°C	-

As can be seen, the gate driver integrated circuit we have chosen not only offers a compact solution with its coreless transformers structure, but also allows sink and source current up to 7.5A, and under these conditions, the maximum propagation delay is given as 300ns, typically. In addition, 1EDI40I12AF not only offers good thermal behavior for the SOIC-8 package, but also has electrically satisfactory properties with its wide input and output voltage range and wide bandwidth up to 1MHz. In addition, it is an important selection criterion that it offers different drive paths for MOSFET opening and closing. Finally, it was preferred to offer UVLO (Under voltage lock-out), short circuit clamping and active shut-down features together in this cost-effective product.

We created a simple spice model to observe how consistent this gate driver integration is with the datasheet data. The visual about this is given in section 4.2.1, in Figure 4.28.

4.1.6. Measurement circuit design

Measurement circuit design is one of the most important building blocks in the thesis study. In order to minimize the measurement tolerance and linearity error caused by the design, each component contributing to the measurement circuit was subjected to a detailed screening in terms of electrical parameters and price effectiveness. In this section, the electrical characteristics of measurement circuits are discussed theoretically and with hand calculations, and studies requiring Spice modeling such as tolerance analysis are used where necessary. Detailed schematic design and PCB layout studies will also be included in the implementation section.

It is unthinkable to correct the output regulation based on an inaccurate and noisy measurement. Therefore, in the next section, the components selected for the voltage measurement circuit and the measurement tolerance of the circuit will be discussed first.

4.1.6.1. Voltage measurement circuit

Following the research for the voltage measurement circuit and comparing the advantages and disadvantages of the alternatives, AD622ARZ [45] was chosen as the instrumentation amplifier, SI8920 [46] as the isolated differential amplifier and LT1630 [47] as the general-purpose amplifier. As the reference IC, the product with part number MCP1501-18E [48] was preferred. The simulation model of the designed measurement circuit is shared in Figure 4.17. The simulation model will be further examined in Section 4.2.1.3.

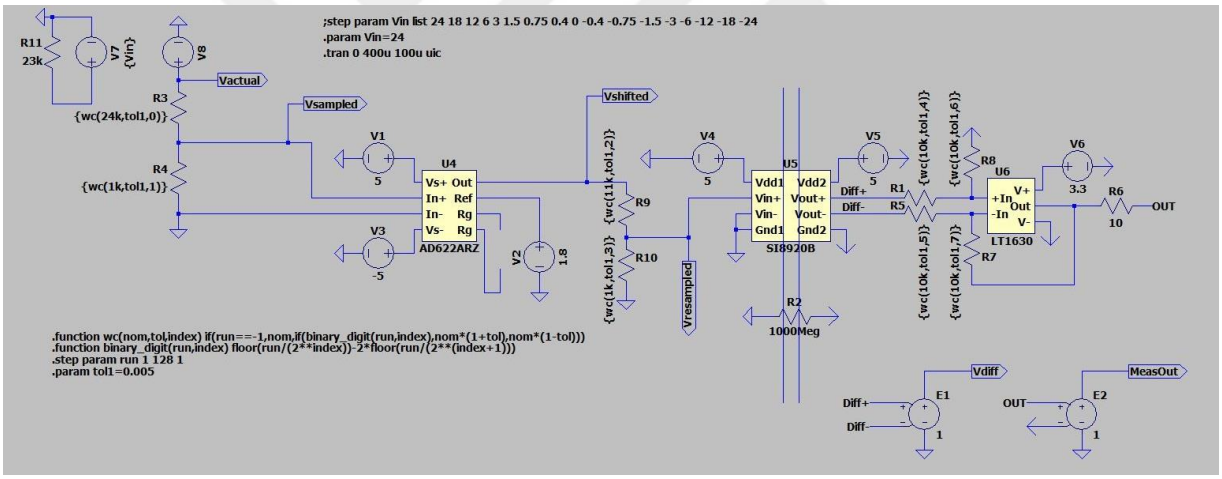


Figure 4.17. Voltage Measurement Circuit Simulation

The same circuit structure was used for DC+, DC- voltage measurements and AC voltage measurement of each phase.

In order to measure negative alternance, instrumentation amplifiers (in-amp) were fed with positive and negative 5V voltages. In addition, to bring the negative voltage to the range where isolated op-amps can make meaningful measurements and to minimize noise-related error, a fixed reference voltage of 1.8V is provided to the reference voltage input of the instrumentation amplifier by MCP1501-18E. The output voltage error tolerance of this integrated circuit is only 0.1%. In order to reduce the bus voltage or phase voltage to the measurement voltage range that AD622ARZ can accept, a voltage divider has been applied with very low tolerance (0.5%)

resistors. AD622 is the in-amp used in the first stage of the measurement circuit and its task is to collect the sensitive measurement value proportioned at the rate of 1/25 with precise 1.8V and transmit it to the output to avoid attenuation. AD622 is configured to have a gain of 1 in this design. A sum of the main electrical parameters of the AD622 under this condition (Gain = 1) are shown in Table 4.4.

Table 4.4. Electrical Parameters of AD622

Parameter	Symbol	Values			Unit	Note/Test Condition
		Min	Typ	Max		
Supply Voltage Range	Vs	+-2.6	-	+-18	V	
Gain Error	e_G	-	0.05	0.15	%	G=1, Vout = +-10V,
Input Offset	V_{OSI}	-	60	125	uV	
Input Voltage Noise	e_{ni}	-	12	-	$nV/\sqrt{Hz}$	
Output Voltage Noise	e_{no}	-	72	-	$nV/\sqrt{Hz}$	
Slew Rate	SR	-	1.2	-	V/µs	
Output Swing	$Vout_{SW}$	-Vs + 1.1	-	+Vs – 1.2	V	Vs=+-2.6V to +-5V, R_{load}=10kΩ
Small Signal -3dB Bandwidth	fsw	-	-	1.0	MHz	G=1
Ambient temperature	T_A	-40	-	85	°C	

The important parameters that enable the selection of this component were its wide voltage supply range, low gain error and offset voltage, high slew rate and wide bandwidth at gain 1. Since we added 1.8V to the output voltage, another important factor is that the output swing voltage is much lower than its counterparts.

MCP1501-18E is one of the most cost-effective solutions that offers the 1.8V reference voltage we need with 0.1% error. The power supply voltage range it offers and the low amount of noise in the output voltage are also compatible with the project requirements. The most important features of the MCP1501-18E, the component responsible for generating the reference voltage, are listed in Table 4.5.

Table 4.5. Electrical Parameters of MCP1501-18E

Parameter	Symbol	Values			Unit	Note/Test Condition
		Min	Typ	Max		
Supply Voltage Range	Vs	2		5.5	V	
Output Voltage	Vo	1.7982	1.8	1.8018	V	at 25°C
Output Voltage Accuracy	Vo_{acc}	-	0.1	-	%	
Line Regulation	$\Delta V_{OUT} / \Delta I_{OUT}$	-	5	50	ppm/V	
Load Regulation	$\Delta V_{OUT} / \Delta I_{OUT}$	-	10	40	ppm/mA	-5 ma < I_{LOAD}
Output Voltage Noise	e_{no}	-	18	-	uV_{p-p}	0.1 Hz to 10 Hz, T_A = +25°C
Power Supply Rejection Ratio	PSRR	-	94	-	dB	Vin = 5.5V, 60 Hz, 100 mV_{PP}
Ambient temperature	T_A	-40	-	125	°C	

In order to bring the voltage at the AD622 output to the -200mV ~ +200mV range, which is the range where the SI8920 can make meaningful measurements, a voltage divider was applied with resistors with the same tolerance error (0.5%). The SI8920 IC used afterwards has the task of both carrying the measurement signal to the microprocessor side in isolation and converting the single-ended signal into a differential signal to protect it from noise on the way to the microprocessor input. The electrical parameters of SI8920 are given in Table 4.6.

Table 4.6. Electrical Parameters of SI8920

Parameter	Symbol	Values			Unit	Note/Test Condition
		Min	Typ	Max		
Input Supply Voltage Range	VDDA	3	-	5.5	V	
Output Supply Voltage Range	VDDB	3	-	5.5	V	
Specified FullScale Input	$V_{AIP} – V_{AIN}$	-200	-	200	mV	
Specified FullScale Output	$V_{AOP} - V_{AON}$	1.58	-	1.65	V	
Output Voltage Noise	e_{no}	-	0.14	0.28	mVrms	100 KHz bandwidth
Nonlinearity	nl	-	0.04	0.15	%	
Gain	G	-	8.1	-	-	T_A = 25 °C
Gain Error	e_G	-0.5	-	0.5	%	G=8.1 (fixed)
Amplifier Bandwidth	fsw		950		KHz	
Ambient Temperature	T_A	-40	-	125	°C	

The last element of the voltage measurement circuit is the LT1630, which is a rail-to-rail general purpose amplifier. The task of this general-purpose amplifier is to convert the signal coming differentially close to the microprocessor ADC pin back to single-ended and transmit it to the ADC input, acting as a buffer.

Equation (4.18), which shows how a step signal applied to the measurement circuit should ideally be monitored at the circuit output, is given below. Here, 'V_{meas}' corresponds to the value at the MCU ADC input terminals, whereas 'V_{actual}' is the real measured value.

$$V_{meas} = \left(\frac{\left(\left(\frac{V_{actual}}{25} \right) + 1.8 \right)}{12} \right) * 8 \tag{4.18}$$

As an example, according to this equation, when a 12V step test signal is applied, the voltage to be measured at the ADC input of the microprocessor should be equal to 1.52V. Included in section 4.2.1, we present the tolerance analysis we conducted on this matter as well as the analysis of the total delay of the measurement circuit.

4.1.6.2. Current measurement circuit

For current measurements, the component with part number GO-20-SMS [49], which works directly with the hall-effect principle and takes measurements in isolation, was used in cascade with LT1630. Circuit details will be covered in the schematic design section (Section 5.2). According to the datasheet information provided, the sensitivity of this product is 40mV/A and the maximum measurement error is 1% even under the worst condition. In addition, the supply voltage requirement and other electrical parameters make this component stand out among current sensors. Although GO-10-SMS with lower current measurement capacity (10A RMS) is a more cost-effective option in terms of not over-designing, GO-20-SMS stood out as a suitable option in terms of material availability as it could not be supplied during the study period.

Main electrical parameters of GO-20-SMS are given in Table 4.7.

Table 4.7. Electrical Parameters of GO-20-SMS

Parameter	Symbol	Values			Unit	Note/Test Condition
		Min	Typ	Max		
Input Supply Voltage Range	Uc	4.5	5	5.5	V	
RMS Current Measurement Range	I_{PM}	-50	20 (I_{PN})	50	A	
Output Voltage Range	$U_{OUT} - U_{REF}$	-2	-	2	V	At I_{PM}
Electrical Offset Voltage	U_{OE}	-5	-	5	mV	$U_{OUT} - U_{REF}$ @ U_{REF} = 2.5 V
Noise voltage spectral density	u_{no}	-	7	-	$nV/\sqrt{Hz}$	100 Hz to 100 KHz
Nominal Sensitivity	S_N	-	40	-	mV/A	800 mV @ I_{PN}
Total error over temperature	E_{TOT}	1	-	1	% (I_{PM})	@ I_{PN}
Frequency Bandwidth −3 dB	fsw	-	300	-	KHz	T_A = 25 °C
Ambient Temperature	T_A	-40	-	125	°C	

Although it is not a must for proposed control technique these current sensors are added to the design for the hardware validation task such as efficiency calculations.

Finally, some electrical properties of the LT1630, which is used as the last element of both the voltage and current measurement circuit, are given in Table 4.8. The main factors for choosing this component are that it has a wide supply voltage range, low offset voltage, high slew rate and low output swing voltage, along with reasons related to material supply.

Table 4.8. Electrical Parameters of LT1630

Parameter	Symbol	Values			Unit	Note/Test Condition
		Min	Typ	Max		
Supply Voltage Range	Vs	+-2.7	-	+-15	V	
Gain Error	e_G	-	-	-	%	
Input Offset Voltage	V_{OSI}	-	150	525	uV	V_{CM} = V+, V_{CM} = V-
Input Voltage Noise	e_{ni}	-	6	-	$nV/\sqrt{Hz}$	f = 1KHz
Output Voltage Noise	e_{no}	-	200	-	$nV/\sqrt{Hz}$	
Slew Rate	SR	-	10	-	V/us	
Output Swing	$Vout_{SW}$	-Vs + 1	-	+Vs – 1.4	V	I_{SINK} = 20mA, Vs = 3V
Gain-Bandwidth Product	GBP	15	30	-	MHz	f = 100KHz
Ambient temperature	T_A	-40	-	85	°C	

4.1.7. Auxiliary power supply design & selection

Throughout the thesis study, due to various requirements, switched mode power supply (SMPS) or LDO (Linear Drop-out Regulator) based DCDC converter designs with different power, input and voltage ranges were needed. The block diagram of the voltage converters designed within the framework of the developed inverter hardware or purchased ready-made as modules is given in Figure 4.18.

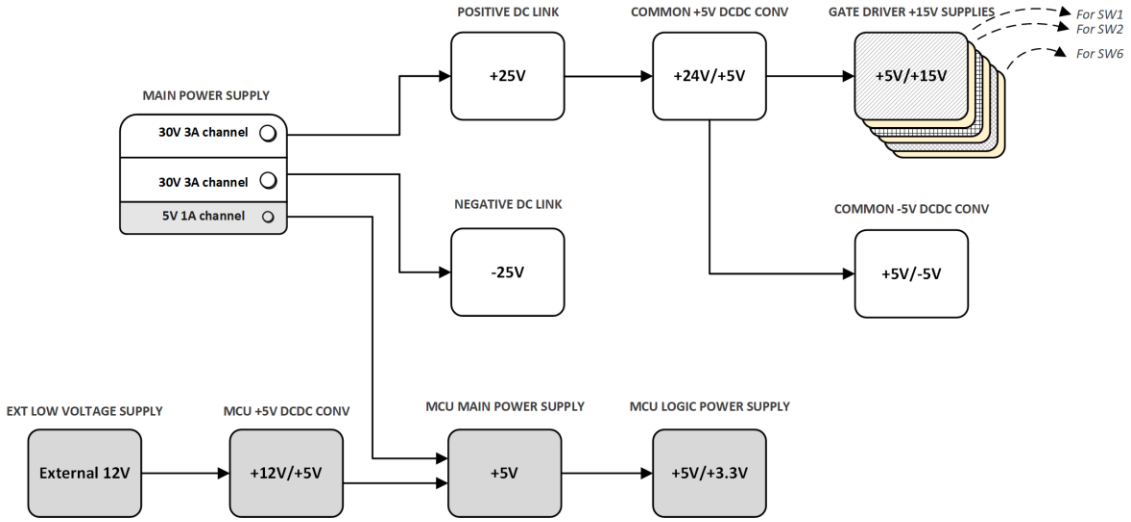


Figure 4.18. Power Tree of Designed Inverter Hardware

In the project, the product with part number MT-305-DII from Mervesan, which has two 30V 3A and one 5V 1A output, was used as the main power supply. This power supply also provides the split busbar structure required for the applied voltage source inverter architecture and SPWM method. The split busbar structure (+26V/-26V busbar) is set in the MT-305-DII serial configuration, and the middle point of the 2 high current channels, which is at the same electrical potential, is used as the system ground. The 5V/1A channel is an isolated channel. Therefore, it can directly power the microprocessor when there is no other external power source. As shown above, the converters designed and selected within the framework of the thesis study are as follows:

- 1x +25V to +5V Converter (3A output current)
- 1x +5V to -5V Converter (100mA output current)
- 6x +5V to +15V Isolated Converter (67mA output current)

- 1x +12V to +5V Converter (800mA output current)
- 1x +5V to +3.3V Converter (300mA output current)

More detailed information about these converters, which are designed or selected from among the products available on the market as modules, will be shared in the following sections. The power tree of the control hardware card, NucleoG431RB, will also be mentioned in Section 4.3.1.

4.1.7.1. +25V to +5V converter

+5V is obtained from +26V, which is the positive DC bus voltage, with a Buck converter working with the SMPS method. The regulated +5V voltage is used in the circuit both for the positive supplies of the integrated circuits used for voltage measurements and serves as a source to produce -5V and isolated +15V voltages in the circuit. Therefore, it is critical that it is robust and stable.

To produce +5V, Monolithic Power Systems' MP1584 [50] based DCDC converter product [51], which is available in the market as a ready-made printed circuit board, was used and we can set the input voltage in the range of 4.5V- 28V to the range of 0.8V- 20V with the help of the potentiometer on its output. The compatible price of the product, the fact that it is available in a ready-made PCB form, its wide current range up to 3A and the wide operating temperature range from -40°C to +85°C are among the factors that were effective in choosing this product. The circuit configuration of MP1584 to obtain 5V output is given in Figure 4.19.

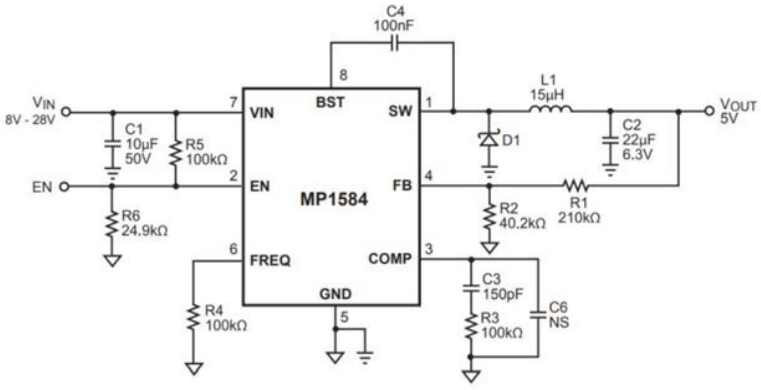
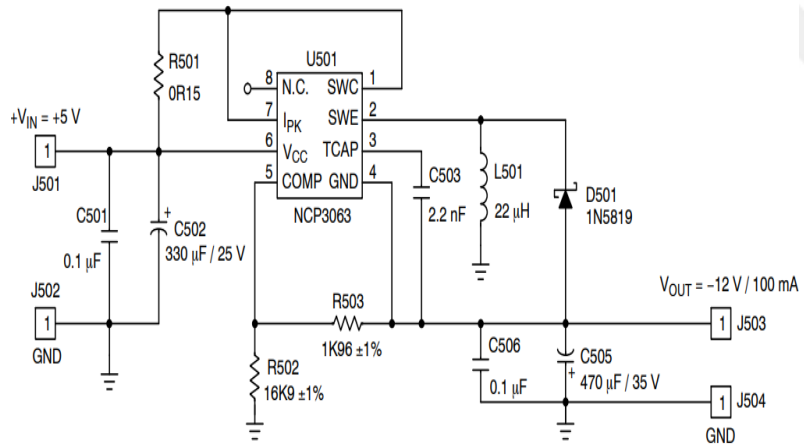


Figure 4.19. Stepdown Circuit Configuration of MP1584

We will discuss how the MP1584 based DCDC converter is implemented in our design, its schematic and PCB image in Section 5.2.7.

4.1.7.2. +5V to -5V converter

Generating negative 5V voltage is one of the most challenging issues within the scope of the project. Because high-accuracy negative voltage generation techniques are limited. -5V voltage is required to prevent the integrated circuits we use when making voltage measurements from clamping with negative alternance due to swing voltage. To produce -5V voltage, we used +5V as the source, which was produced as explained in the previous section. Onsemi's IC with part number NCP3063 [52] and with a SOIC-8 package was set in inverting configuration and -5V was obtained with an inverting SMPS application. For this, the following changes were made to the circuit configuration given in Figure 4.20 (a). R503 was set as 1KΩ and R502 was set as 3.1KΩ. The inductance value seen as L501 was set as 390uH, and the timer capacity defined as C503 was set as 10pF. The equations from the datasheet which are used to calculate the circuit components are given in Figure 4.20 (b).



(See Notes 9, 10, 11)	Voltage−Inverting
$\frac{t_{on}}{t_{off}}$	$\frac{\lvert V_{out}\rvert + V_F}{V_{in} - V_{SWCE}}$
t_{on}	$\frac{\frac{t_{on}}{t_{off}}}{f\left(\frac{t_{on}}{t_{off}} + 1\right)}$
C_T	$C_T = \frac{381.6 \cdot 10^{-6}}{f_{osc}} - 343 \cdot 10^{-12}$
$I_{L(avg)}$	$I_{out}\left(\frac{t_{on}}{t_{off}} + 1\right)$
$I_{pk\ (Switch)}$	$I_{L(avg)} + \frac{\Delta I_L}{2}$
R_{SC}	$\frac{0.20}{I_{pk\ (Switch)}}$
L	$\left(\frac{V_{in} - V_{SWCE}}{\Delta I_L}\right) t_{on}$
$V_{ripple(pp)}$	$\approx \frac{t_{on}\, I_{out}}{C_O} + \Delta I_L \cdot ESR$
V_{out}	$V_{TH}\left(\frac{R_2}{R_1} + 1\right)$

(a) (b)

Figure 4.20. (a) Inverting Circuit Configuration of NCP3063 (b) Equations for Inverting Operation

4.1.7.3. +5V to +15V isolated converter

As it is known, we used six power switches in the voltage source inverter circuit and six isolated gate drive integrated circuits to drive these switches. This situation created the need for isolated voltage to power the MOSFET side of each gate driver. The most cost-effective solution found

was to produce these voltages from +5V voltage with six DCDC modules. First of all, the minimum electrical requirements of the gate driver had to be determined. The average required DCDC output current was calculated as in Equation (4.19).

$$Q_g = I_{avg} * t_{sw} \tag{4.19}$$

Where, 'Q_q' is total gate charge of IRFZ48N, 't_{sw}' is the switching period of the MOSFET which is 100µs. So, 'I_{avg}', that is the required average current for the MOSFET, is calculated as 1.1mA. The minimum input power required for DCDC was calculated according to Equation (4.20).

$$P_{dcdc} = \frac{I_{avg} * V_{dcdc}}{eff_{dcdc}} * 1.5 \tag{4.20}$$

Here, 'V_{dcdc}' is the output voltage of the isolated DCDC converter. The 50% margin was introduced as a precaution against the possibility of MOSFET capacities drawing current from DCDC much higher than expected in cases where high di/dt or dv/dt may occur. Even under such a conservative estimation, the amount of power required by isolated DCDC was calculated as 35mW.

Under these conditions, the 1W B0515S-1W [53] DCDC SMPS module of Mornsun appeared as the most suitable solution. The product isolatedly converts voltage levels between 4.5V and 5.5V to 15V, the voltage value at which we drive fixed MOSFET gates, with typically 80% efficiency. The output current of the product is 67mA and easily meets our needs. The line regulation value is given as 1.2% and the load regulation value is typically given as 6.3% for 15V output, which is within the safe voltage range required to drive MOSFETs. In addition, it meets the requirements of our project with its compact size and operating temperature range of -40°C - +85°C. The circuit structure is simply depicted in Figure 4.21.

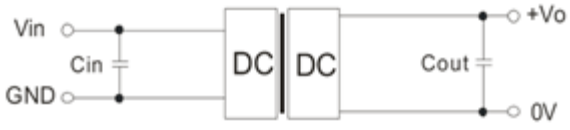
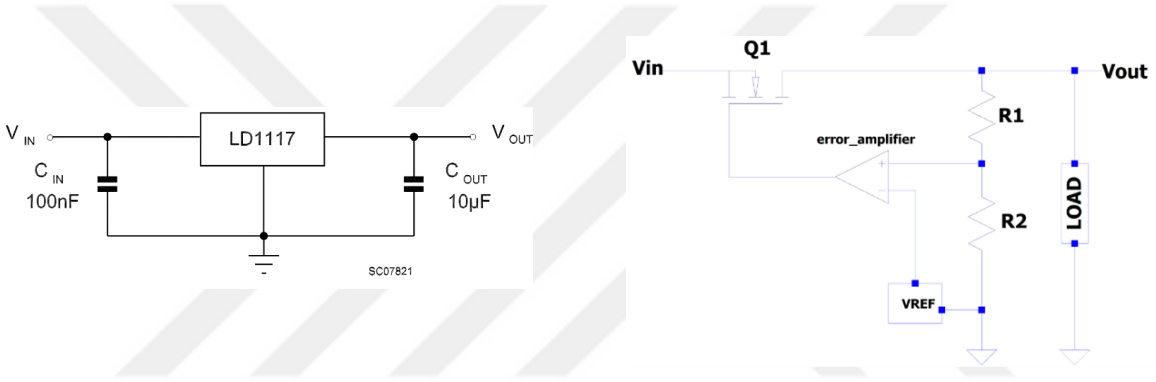


Figure 4.21. Recommended Circuit Configuration of B0515S-1W

4.1.7.4. +12V to +5V converter

Conversion from 12V to 5V is required to produce the microprocessor supply voltage when the microprocessor board does not have a computer connection and is not supplied with an external isolated 5V. For the microprocessor power to be supplied properly, the 12V supplied to the system must have a reference ground isolated from the power circuit. It is important that there is no ripple and overshoot/undershoot on the 5V to be used for microprocessor supply. Therefore, an LDO type converter was used for 12V to 5V conversion. Figure 4.22 (a) shows the typical connection circuit of the LD1117S50TR [55] chosen for this design.



(a) (b)

Figure 4.22. (a) Circuit Configuration of LD117S50 (b) Typical Representation of an LDO

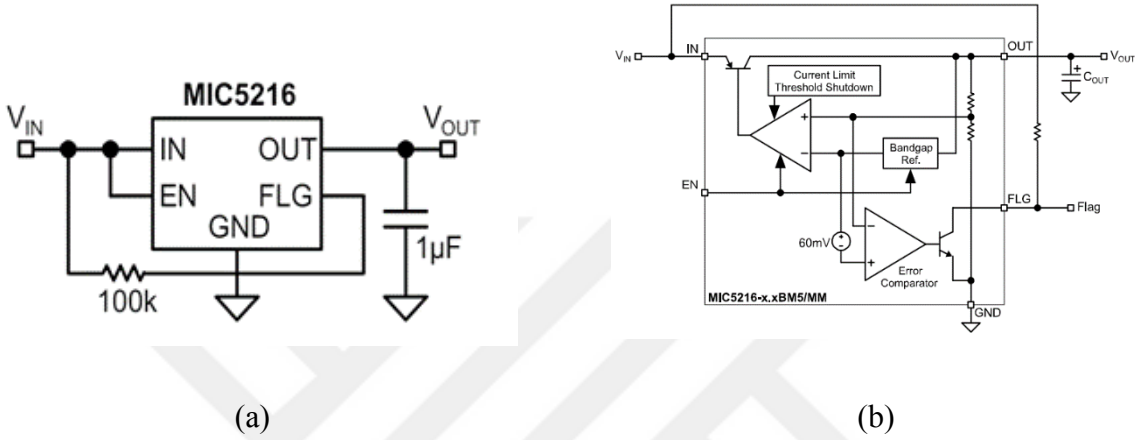
ST Microelectronics' LD1117S50TR product is also on the Nucleo board, which has been validated many times before. In our project, the SOT-223 package version of this integrated circuit was preferred as it is in Nucleo board. This product can easily flow 800mA current while converting from 12V to 5V, when sufficient cooling is provided.

The regulators operating temperature range is -40°C - +125°C, which is a preferred range amongst linear regulators. Because linear regulators dissipate the energy loss resulting from the voltage drop between input and output as heat. Figure 4.22 (b) highlights this situation caused by the circuit structure of LDOs.

4.1.7.5. +5V to +3.3V converter

The conversion of isolated 5V to 3.3V is necessary to make measurements that require isolated 3.3V on the power card, and at the same time, for this application that requires precise

measurement, it is very important that the logic signals between the multi-power card and the microprocessor card have the same 3.3V reference voltage. As a result, MIC5216-3.3YM5 was preferred with the SOT-23 package for this precise conversion. Images showing the typical circuit structure and internal circuit structure of MIC5216-3.3 are given in Figure 4.23.



(a) (b)

Figure 4.23. (a) Circuit Configuration of MIC5216 (b) Internal Block Diagram of MIC5216

MIC5216-3.3 -40 125 stands out with its wide temperature range and max 0.05% of line regulation and max 0.5% load regulation features. Also, its output voltage accuracy tolerance is less than 1%.

4.1.8. Connector selection

The connectors selected for the inverter hardware design and development are given in Table 4.9.

Table 4.9. Selected Connectors for Inverter Hardware

Purpose of Usage	Part Number	Description	Amount
Input Power Connections	ZB128-5.0-03P+	3-pin 10A 300V screw terminal (5.00mm)	1x
Phase to Load Connections	ZB128-5.0-02P+	2-pin 10A 300V screw terminal (5.00mm)	3x
Power Board to Nucleo Signal Connections	SSW-119-02-G-D	38 Pos TH receptacle connector (2.54mm)	2x
Power Board to Nucleo GND Connections	SSW-102-01-G-S	2 Pos TH receptacle connector (2.54mm)	2x
External Supply to Iso 5V Conn.	ZB128-5.0-02P+	2-pin 10A 300V screw terminal (5.00mm)	1x
Power Board UART Connections	87438-0443	4-pin 2.5A right angle header (2.54mm)	1x

4.2. Simulation Studies

In this section, the simulation studies are carried out within the scope of the thesis and the findings obtained are examined. While the first part examines the LtSpice models and outputs created for hardware design activities, in the second part we evaluate the simulation activities and outputs focused on control using Plecs and Simulink.

4.2.1. LtSpice simulations

While conducting hardware design activities, we utilized LtSpice simulations in many stages. Since the design steps are elaborately explained in Section 4.1, this section will only include the generated models and the obtained outputs for each spice simulation, along with a brief mention of the application purpose.

4.2.1.1. Main voltage source inverter simulation

The main purpose of using LtSpice simulations in the thesis is to support hardware design activities and verify the consistency of hand calculations regarding the selected components. The spice model for the open loop operation of the 3-phase voltage source inverter is shown in Figure 4.24.

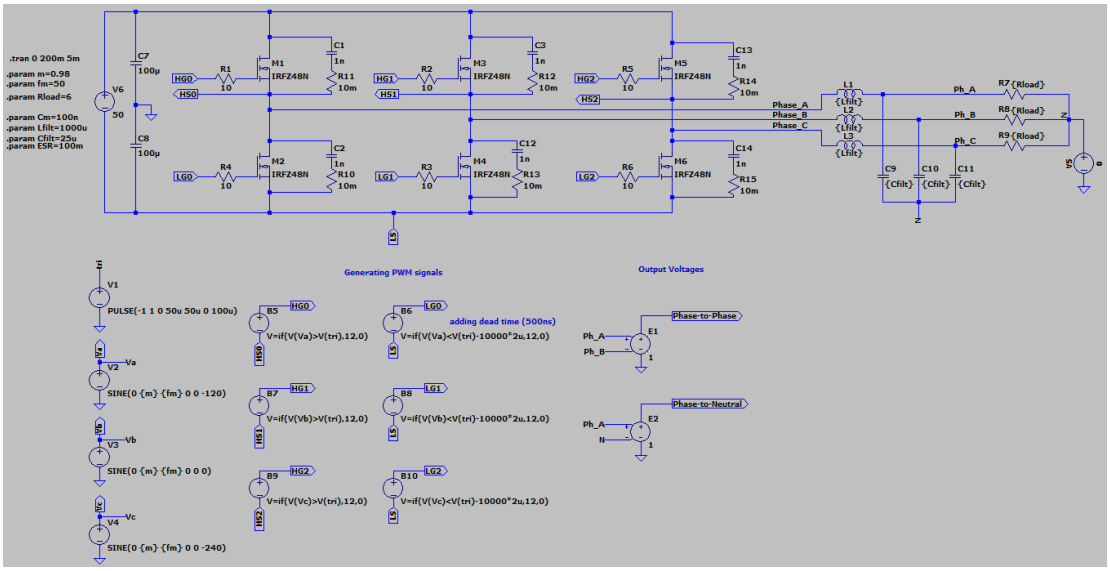


Figure 4.24. Main Switching Behavior Simulation of the 3-Phase 2-Level Inverter

This model has formed the basis for many studies within the scope of the thesis, such as determining power switches, designing the output LC filter, determining dead time, controlling busbar stabilization and testing different switching topologies. According to this LtSpice model, the output phase-neutral voltages for 3 phases are given in Figure 4.25.

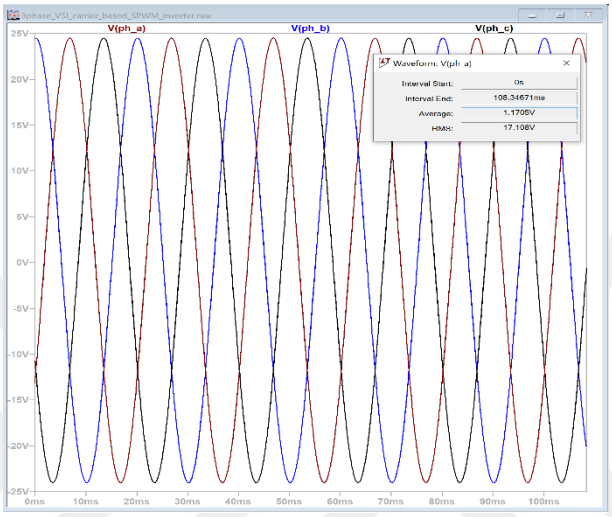


Figure 4.25. Three Phase Inverter Output Voltages

According to the simulation result, the RMS value of the output voltage is measured to be approximately 17.1V which is slightly higher than the targeted value of 16V and a little bit lower than the theoretical maximum value which is calculated in Equation (4.21).

$$V_{phase} = \frac{V_{dc}}{2} * \sqrt{2} \tag{4.21}$$

The fact that the phase voltage value measured in the simulation is very close to the theoretical maximum degree can be explained by many reasons, such as not adding the noise, busbar imbalance and parasitic elements, neglecting the microprocessor delays and consideration of the gate-drive circuit as ideal. These assumptions and idealizations applied during the simulation were made in order not to increase the simulation complexity. However, it is expected that the theoretical maximum value will not be reached since the switching and connection loss of the switches and the voltage drops on the output inductances are reflected in the simulation.

When deemed necessary, component-specific LtSpice simulations were not avoided. These simulations are also mentioned in the next section.

4.2.1.2. Gate driver circuit simulation

A robust hardware design is required to implement the control algorithm in the most optimal way. One of the most important features of a good switching circuit is that the switching element is selected well, and another is that the driving technique is strong in accordance with the system requirements. For this reason, the simulation model in Figure 4.26 was created to test whether the IRFZ48N MOSFETs we chose as the power switch on a single-phase inverter circuit and the gate driver integrated 1EDI40I12AF selected for the switching circuit work in accordance with the hand calculations and datasheet data.

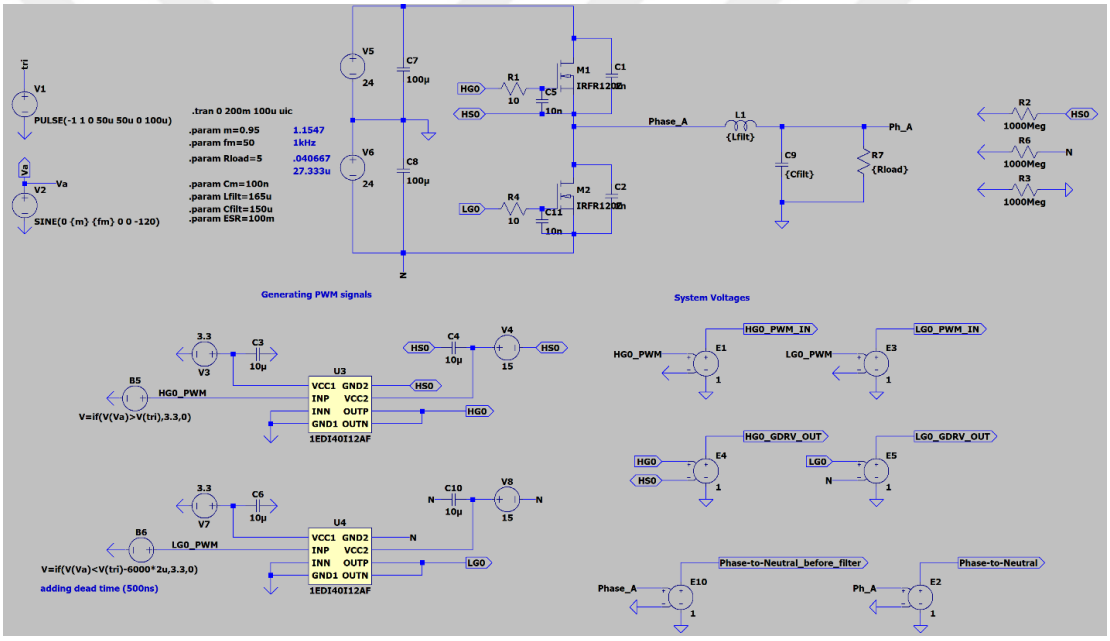


Figure 4.26. Gate Driver Simulation Model

In this simulation, the overall operation of the gate driver is examined. The waveforms obtained when the PWM pulses produced with the SPWM method are given as input to the gate driver integrated are shown in Figure 4.27. In the top waveform, the blue line shows the sine reference, while the triangular shaped wave represents the carrier signal. The middle waveforms show high gate drivers output signals, where the bottom one shows the same for low gate driver.

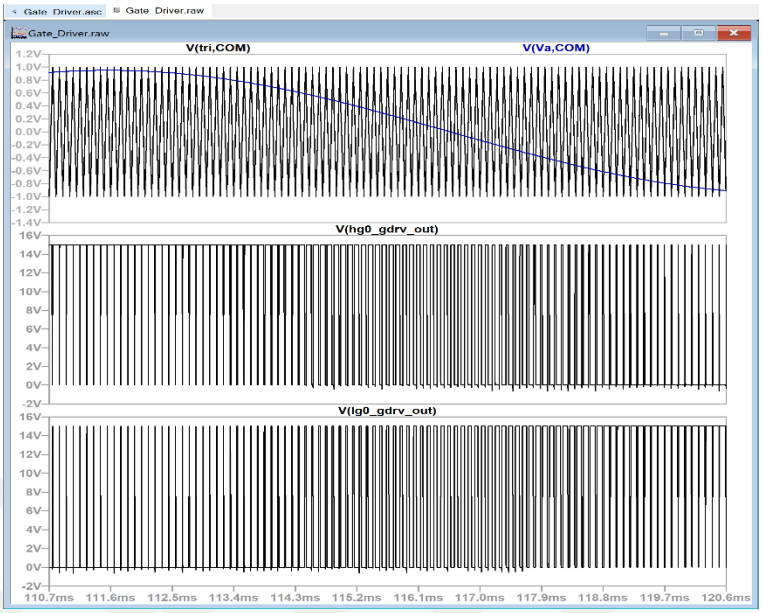
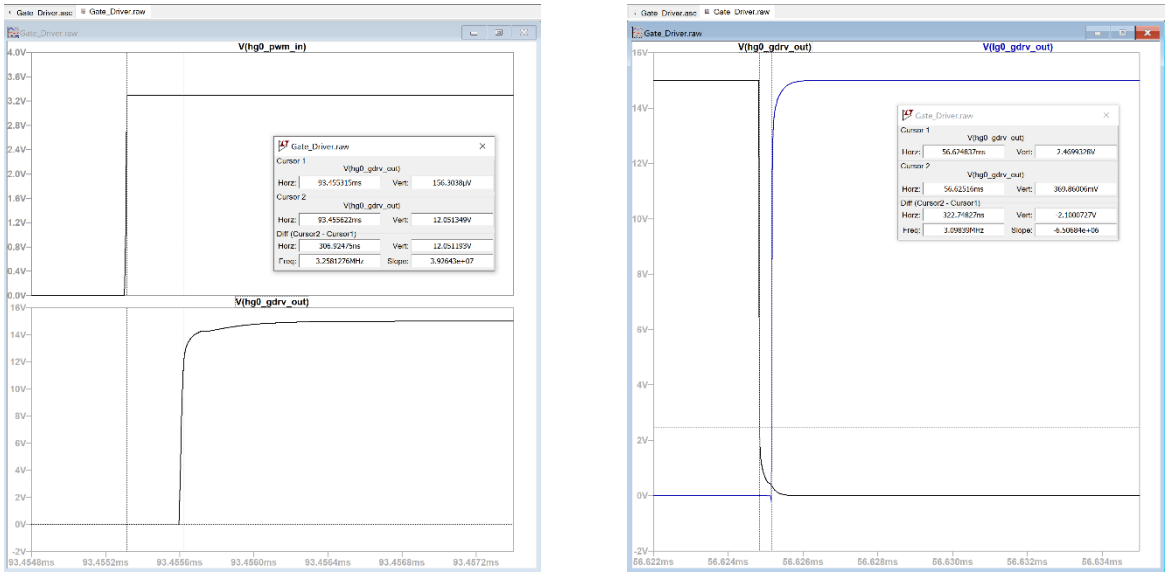


Figure 4.27. Gate Driver Simulation Outputs

The propagation delay parameter of the 1EDI40I12AF is given as 300ns in the datasheet. In Figure 4.28 (a), it is observed how long after the step input given to the integrated input turns out to be step output. The result shows that the simulation study is consistent with the datasheet data.



(a) (b)

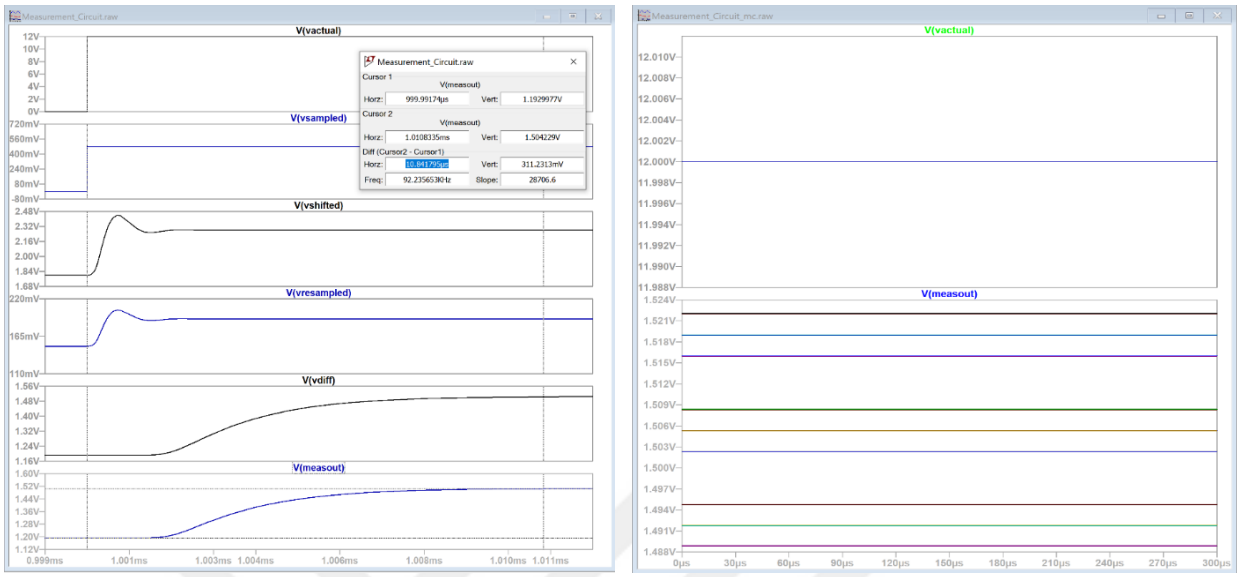
Figure 4.28. (a) Gate Driver Propagation Delay (b) Dead Time Sufficiency

One of the most important criteria to consider in a half bridge structure is that complementary switches should not be in transmission at the same time. Two MOSFETs remaining in conduction at the same time due to the delays experienced during the opening and closing moments may cause shoot through. In order to prevent this situation from occurring, it is essential to allocate a dead-time period while generating switching signals, that is, both switches remain in the "OFF" position for a short duration at the same time. In Figure 4.28 (b), it is examined whether the dead-time period, which we previously determined as 300ns, is sufficient for the switches to open and close correctly and whether any short circuits occur. The simulation study shows that the 300ns period determined for dead-time is suitable for the proper operation of the system.

4.2.1.3. Monte Carlo tolerance analysis

The success of the control techniques developed and applied in this thesis is largely depending to the measurement sensitivity. It is unthinkable that an RMS value calculated based on an incorrect measurement completely coincides with the target value. Within the scope of this design, we determined our measurement error to be less than 5% even under the worst case.

In the simulation model we shared earlier (Figure 4.10), a step signal was applied to AD622, the component at the input of the measurement circuit, and the output signal produced from the LT1630 output, the last component in the circuit, was measured. The total time until the output signal was obtained, that is, the group delay of the circuit, was measured as 10.7 µs in Figure 4.29 (a). Since the sampling period is 100 µs, the measurement delay of 10.7 µs is quite satisfactory and coincides with the software requirements of the design. The selected sampling and therefore interrupt frequency is 10 KHz, leaving us with a window of 83.3 µs to perform the necessary calculations for control. This duration is sufficient to implement the developed control algorithm.



(a) (b)

Figure 4.29. (a) Group Delay of the Voltage Measurements (b) Exemplary Tolerance Analysis Result

As an example, when tolerance analysis is applied for the step input signal with 12Vdc amplitude, the voltage that the microprocessor ADC will read is expected to be 1.52V according to Equation (4.18). According to the Monte Carlo analysis applied by entering the tolerances of the passive materials selected in the circuit design, the highest value that could be obtained in the worst case was 1.522V, while the lowest value was 1.488V. LtSpice outputs showing Monte Carlo worst-case analysis results for 12V step input are shared in Figure 4.29 (b).

In this way, various values were given to the measurement circuit as step input and Monte Carlo analysis was applied for each separately. Analysis results are presented in Table 4.10. Throughout the analysis, the reference voltage was applied as 1.8V and the measurement resistance tolerances were applied as 0.5%.

Table 4.10. Tolerance Analysis Results for Various Instantaneous Voltages

Vsimultaneous (V)	Vexpected (V)	Vmeasured,max (V)	Vmeasured,min (V)	Error (%)
25	1.866	1.870	1.822	2.35
12	1.520	1.522	1.488	2.11
6	1.360	1.362	1.335	1.84
0.75	1.220	1.222	1.199	1.72
0	1.200	1.202	1.18	1.67
-12	0.880	0.885	0.863	1.93
-25	0.533	0.539	0.517	3.00

Typically, due to factors such as noise on the measurement signal and IC offset voltages, the situations where measurement distortion is most common are measurements with negative alternance. Thus, the worst-case measurement in this design is at -25V. As we mentioned in Section 4.1.6, the real voltage values equivalent in the microprocessor is calculated according to the Equation (4.18). Theoretically, an output voltage of 1.866V should be obtained for +25V and 533mV for -25V. Indeed, during the simulation run, the worst-case maximum measurement error was determined when reading the -25V voltage and even in that case the error rate was limited to 3%. The fact that the error measured at the +25V input is higher than at lower positive input voltages can be explained by the fact that the SI8920 is approaching the limits of its linear measurement range at this point.

4.2.1.4. +5V to -5V converter simulation

Unlike all other converters used as auxiliary power sources in the circuit, the conversion from +5V to -5V is made as a discrete design. Therefore, there was a need to verify the electrical outputs in a simulation environment. Although the LtSpice library of NCP3063 is not available, a simulation study was carried out with MC34063, an old IC model that works with a similar topology. Manufacturer approval has also been received from Onsemi's that NCP3063 is a replacement of MC34063. The simulation model of the inverting converter applied in our design is given in Figure 4.30.

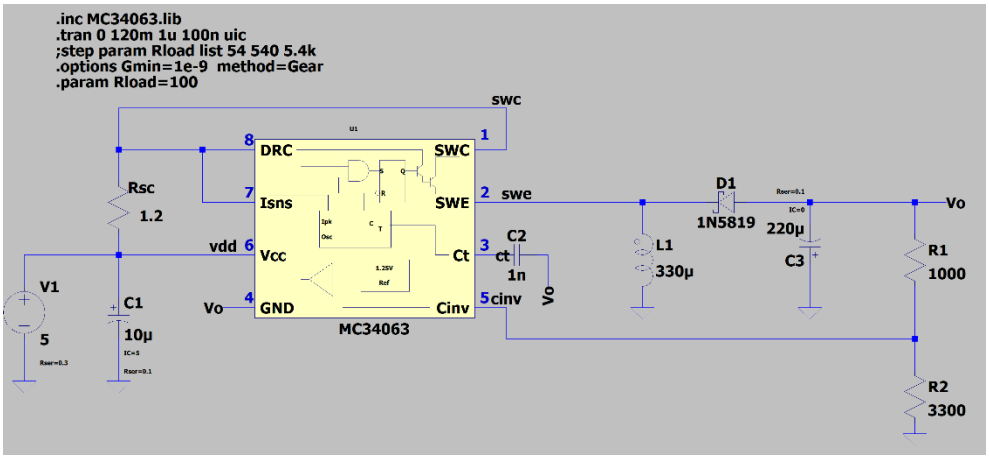


Figure 4.30. Inverting Converter Spice Model

In Figure 4.31, it is seen that the output voltage is obtained as -5.35V with a very low ripple voltage ($30mV_{PP}$). This value is within the voltage range that can be considered as safe for the other circuit components that are supplied by this converter.



Figure 4.31. Inverting Converter Simulation Result

In this section, we have included the most remarkable simulation studies on hardware design. In the next section, we will discuss simulation studies that allow the development of the control algorithm.

4.2.2. Matlab simulations

The Simulink [56] model created for the control algorithm is as follows. (Figure 4.32)

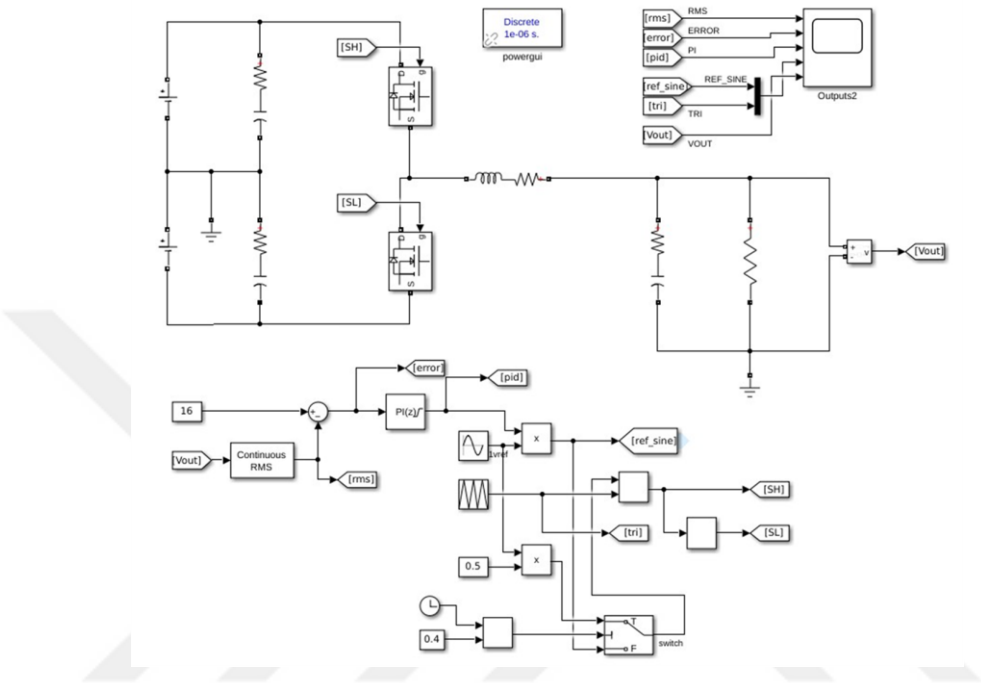


Figure 4.32. Inverter Control Simulation Matlab/Simulink Model

Based on the model above, we see the oscilloscope outputs of the output current (IOUT), the output voltage (VOUT) at the moment of load change, the RMS value of this voltage (RMS), the error information coming to the PI controller (ERROR), the PI controller output (PI) and the reference sine modulated by the PI output (REF_SINE) for the developed control algorithm in Figure 4.33.

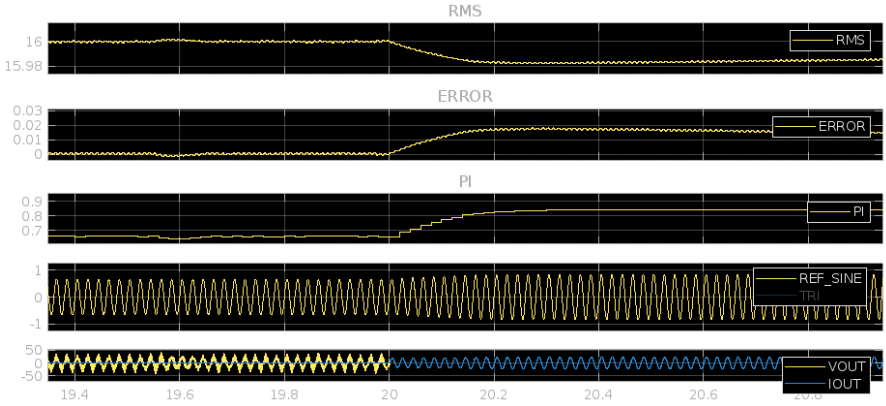


Figure 4.33. Matlab/Simulink Inverter Control Waveforms

The same model works properly with both the classical control method and the developed control method, with a few changes. In the initial phase of control-oriented simulation studies, it was observed that the results obtained from the simulation studies carried out in the Matlab environment were parallel to Plecs. In addition, some simulation blocks in the Plecs environment were evaluated to be more advantageous than Simulink in terms of our control methods under test. For this reason, Plecs simulations were used as reference in the rest of the study in order not to increase the workload. In the next section, the simulations performed in the Plecs domain, and the results of these simulation studies will be examined in detail and compared according to various criteria.

4.2.3. Plecs simulations

The simplified Plecs [57] model as single phase which is created to validate the proposed control algorithm is shown in Figure 4.34.

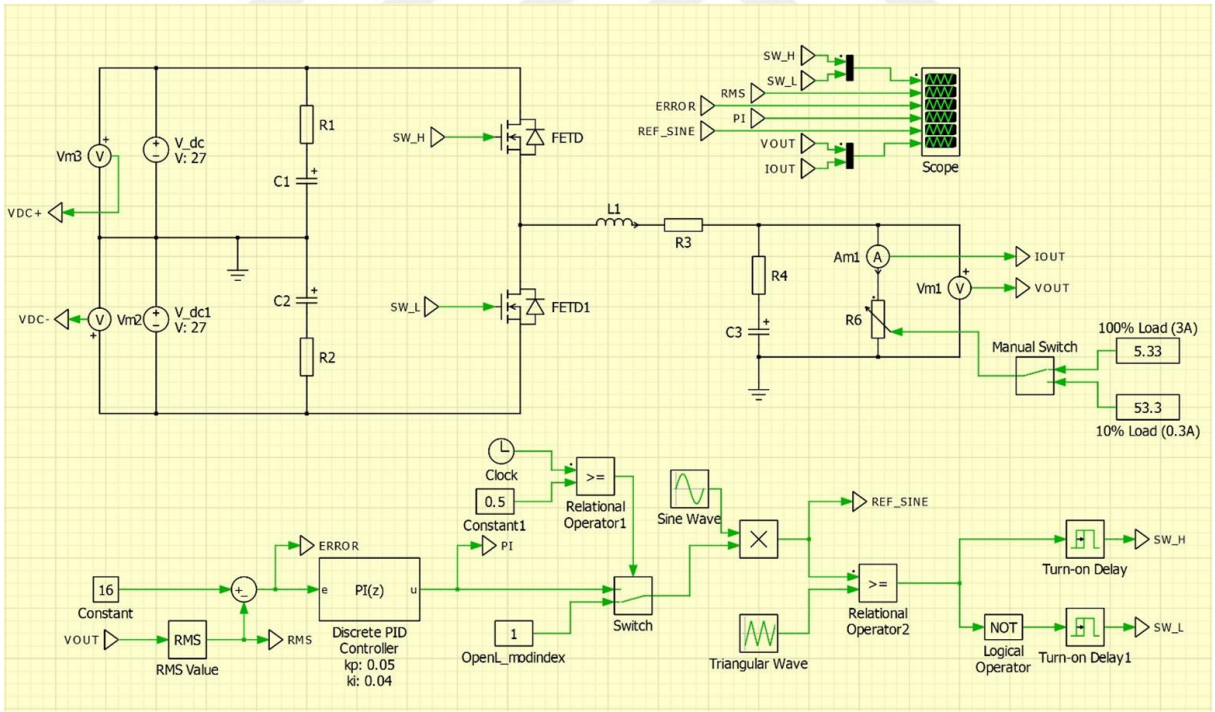


Figure 4.34. Inverter Control Simulation Plecs Model

In this Plecs model, it is shown that a sine with a peak-to-peak value of 2V (-1V +1V) is multiplied by the modulation index and given to the SPWM generator as reference. The maximum amplitude of the sinusoidal voltage corresponds to m = 1. If m exceeds '1', the instantaneous output voltage

will be clipped to Vdc/2, introducing low-order harmonics that can be challenging to filter out. Therefore, to prevent overmodulation resulting from this effect, m should be chosen as less than or equal to '1'."

The established Plecs model allows this modulation index to be updated both in the fundamental and in the sampling frequency. In this way, it is possible to compare the classical RMS approach and the proposed method on the same model. However, since it is not possible to present the ADC oversampling contribution in the Plecs, this part is only included in the implementation phase.

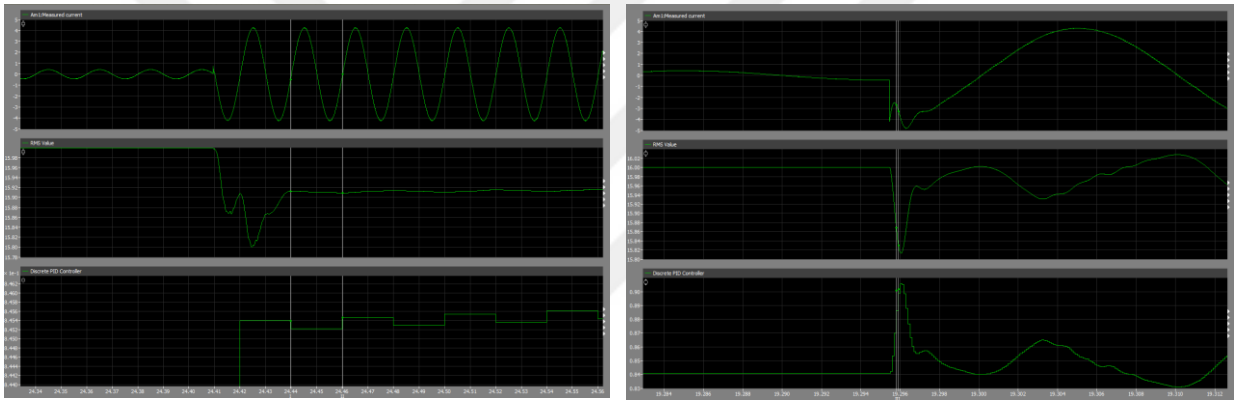
There are two control techniques applied in the model; One is the classical RMS voltage control technique, and the other is the control technique based on the developed sliding RMS method. The working principles of the models that simulate these two techniques are as follows. The output voltage is read with 100µs samples for both control techniques. This read value is compared with the 16Vdc value, which is the reference RMS voltage, and the resulting difference, that is, the error, is sent to the PI controller. This is one of the points where control techniques begin to differ. When applying the classical method, 200 samples are collected within 20ms and RMS voltage occurs only after the first 200 samples. 200 more samples are needed to form the next RMS value. With the developed control technique, RMS value starts to form starting from the 1st sample. After 200 samples are collected, the RMS block begins to move this 200-sample window by shifting it with each new sample. A new RMS calculation does not require collecting another 200 new samples unlike the classical approach.

Another difference is related to the frequency with which the PI controller output is applied to the system. While in the classical method, the intervention to the system is made every 20ms, in the developed technique, we apply the new correction signal to the system in each 100µs without wasting any time. Note that this signal is updated every 100µs in this novel method. It should be noted at this point that we optimized the PI controller parameters for both methods within the simulation models, in their respective ecosystems.

The method of changing the modulation index is again the same for both techniques. We create a new peak value sine reference by multiplying the sine reference we produced with a peak value of +-1V by the value obtained from the PI controller output with a limited value between 0.5 ~ 0.95. In the remaining part, we produce switching signals by comparing our reference sine with a

triangular wave with +-1V peak and 100µs period, as in the classical SPWM method. In the last part of the block diagram, we add our PWM signals, propagation delay and dead-time values.

Figure 4.35 (a) shows, from top to bottom, the output phase voltage occurring in the classical control technique, the RMS value of the output voltage and the update period of the control algorithm (20ms). These signals were captured with the Plecs' oscilloscope during the step load change. In Figure 4.35 (b), the same situation is examined in terms of the developed control technique. It should be noted that the application period of the control signal is 100µs in this waveform. This provides a much faster dynamic response in the developed technique compared to the classical technique.



(a) (b)

Figure 4.35. (a) Classical RMS Method PI Update Routine (b) Proposed RMS Method PI Update Routine

The dynamic response of both control techniques was examined in terms of RMS voltage recovery time and maximum error during various load transitions. The load transitions subject to examination are the transition from light load to heavy load and the transition from heavy load to light load. These load transitions were carried out in the simulation environment through manual and timer switches. The light load condition was created by connecting a 53.3Ω resistive load to the output, and the heavy load condition was created by reducing the resistive load to 5.33Ω. Under these loads, phase currents are 300mA and 3A, respectively.

Figure 4.36 shows the recovery time for both techniques in terms of different step load changes.

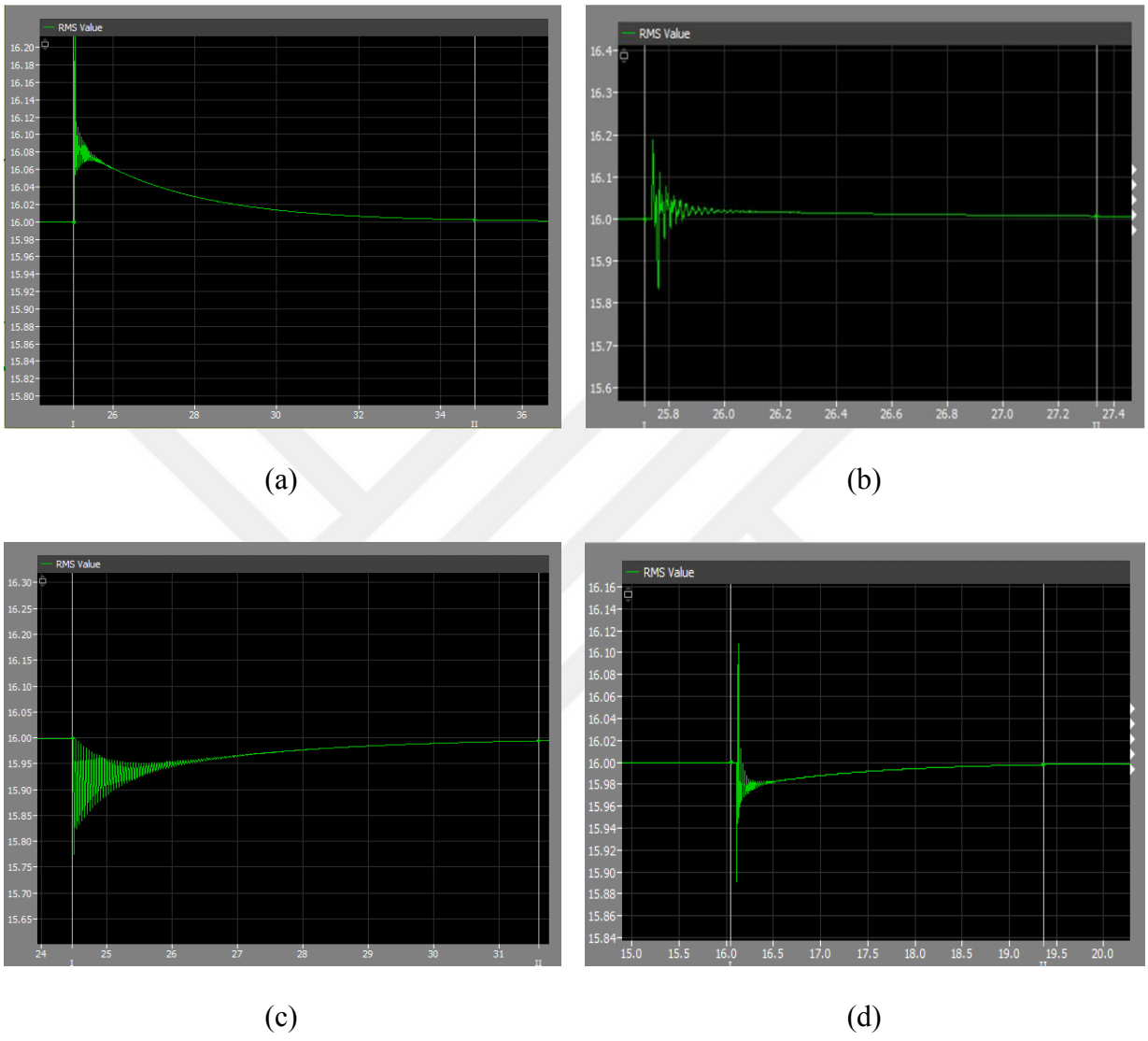


Figure 4.36. (a) Classical Method Recovery Time During Heavy to Light Load Transition (b) Proposed Method Recovery Time During Heavy to Light Load Transition (c) Classical Method Recovery Time During Light to Heavy Load Transition (d) Proposed Method Recovery Time During Light to Heavy Load Transition

The classic method required approximately 9.79 seconds (Figure 4.36 (a)) for the output RMS voltage to settle back to the 16V level during transition from heavy load to light load. In the proposed method, this duration was only 1.62 seconds (Fig 4.36 (b)). Additionally, we conducted load drop tests from light load to full load. While it took 7.12 seconds for output RMS value to

recover for classical method during this load transition (Figure 4.36 (c)), it took only 3.31 seconds for the developed control technique. (Figure 4.36 (d)).

Furthermore, the comparison between the classical and proposed methods is notably significant in terms of error amounts. Figure 4.37 shows the maximum errors for both techniques.

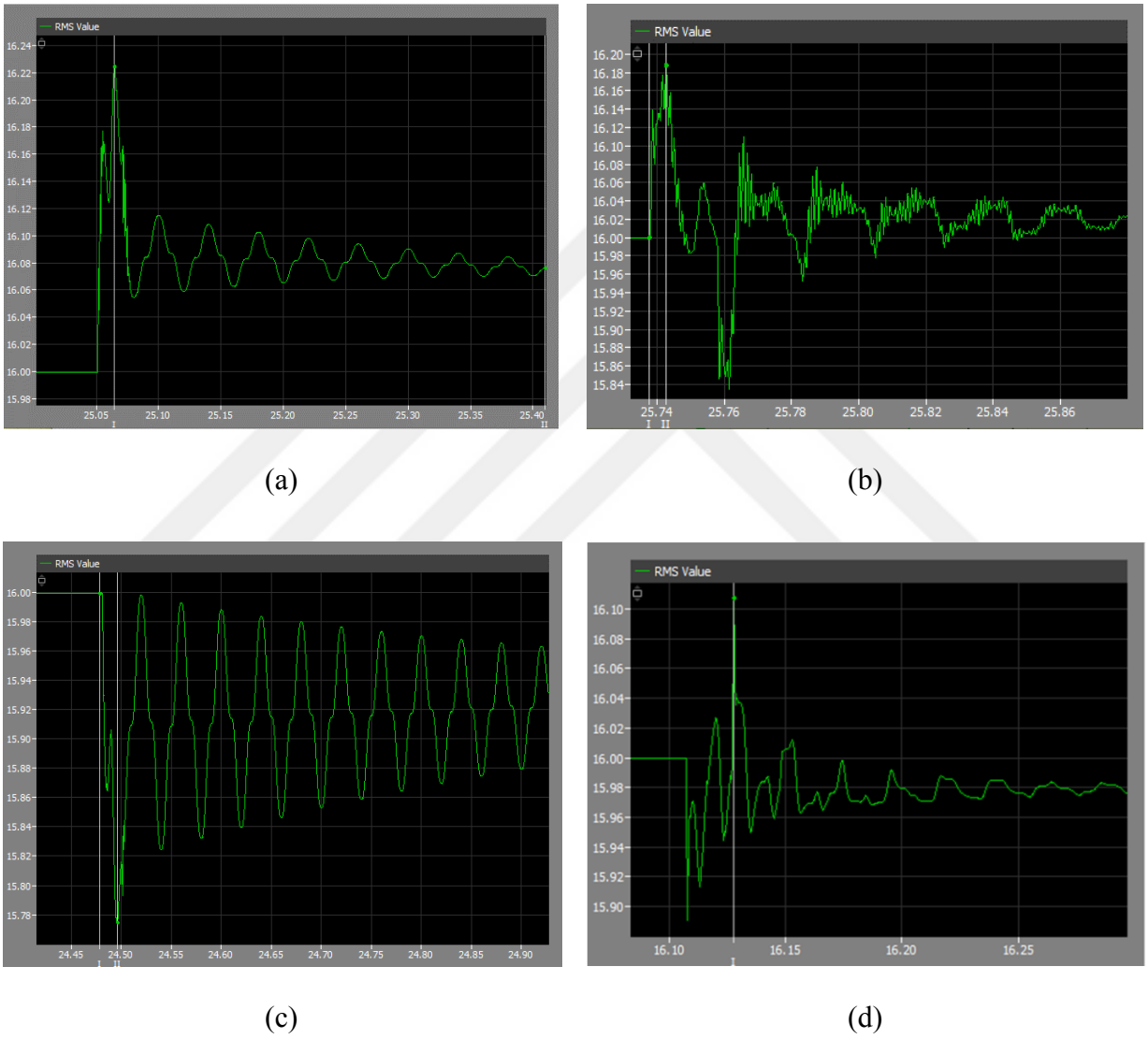


Figure 4.37. (a) Classical Method Max Error During Heavy to Light Load Transition (b) Proposed Method Max Error During Heavy to Light Load Transition (c) Classical Method Max Error During Light to Heavy Load Transition (d) Proposed Method Max Error During Light to Heavy Load Transition

During the same heavy load to light load transition, in the classical method, we observed a maximum error of 220 mV immediately after the transient load change (Figure 4.37 (a)), whereas in the proposed control method, this error was limited to 187 mV (Figure 4.37 (b)). During the light to full transition this is around 224 mV for the classical method (Figure 4.37 (c)), while its measured only 127 mV for the developed technique (Figure 4.37 (d)).

For simplicity all the data collected from these simulation waveforms are summarized in below Table 4.11.

Table 4.11. Summary of the Plecs Simulations Dynamic Response Tests

Transition	Heavy Load → Light Load		Light Load → Heavy Load	
Technique	Classical	Developed	Classical	Developed
Max Error (mV)	220	187	224	127
Recovery Duration (sec)	9.79	1.62	7.12	3.31

Based on these results, the success of the proposed control technique is satisfactory, even though the impact of ADC oversampling cannot be reflected in these simulation results.

4.3. Software Design Steps

In the previous chapters, we have examined in detail the hardware design steps of the 3-phase voltage source inverter designed to be used in this thesis, the simulation models we developed within the scope of the thesis, and the outputs obtained from these studies. In the next section, we will discuss the software design steps applied to implement the control models validated in the simulation environment into the designed hardware. In this section, the design process of the software is explained step by step.

4.3.1. STM32G431 at a glance

The STM32G431RB [58] device is powered by the high-performance Arm Cortex-M4 32-bit RISC core operating at a frequency of up to 170 MHz. In our application, the system frequency is set at 164 MHz using PLL for safe operation.

The Cortex-M4 core incorporates a single-precision FPU, essential for our innovative control method, supporting all Arm single-precision data-processing instructions and implementing a full

set of DSP (Digital Signal Processing) instructions along with a memory protection unit (MPU) to enhance application security. This device integrates high-speed memories, including up to 128 Kbytes of Flash memory and 32 Kbytes of SRAM (Static Random Access Memory), along with an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses, and a 32-bit multi-AHB bus matrix. A detailed functional block scheme of STM32G431RB is given in Figure 4.38.

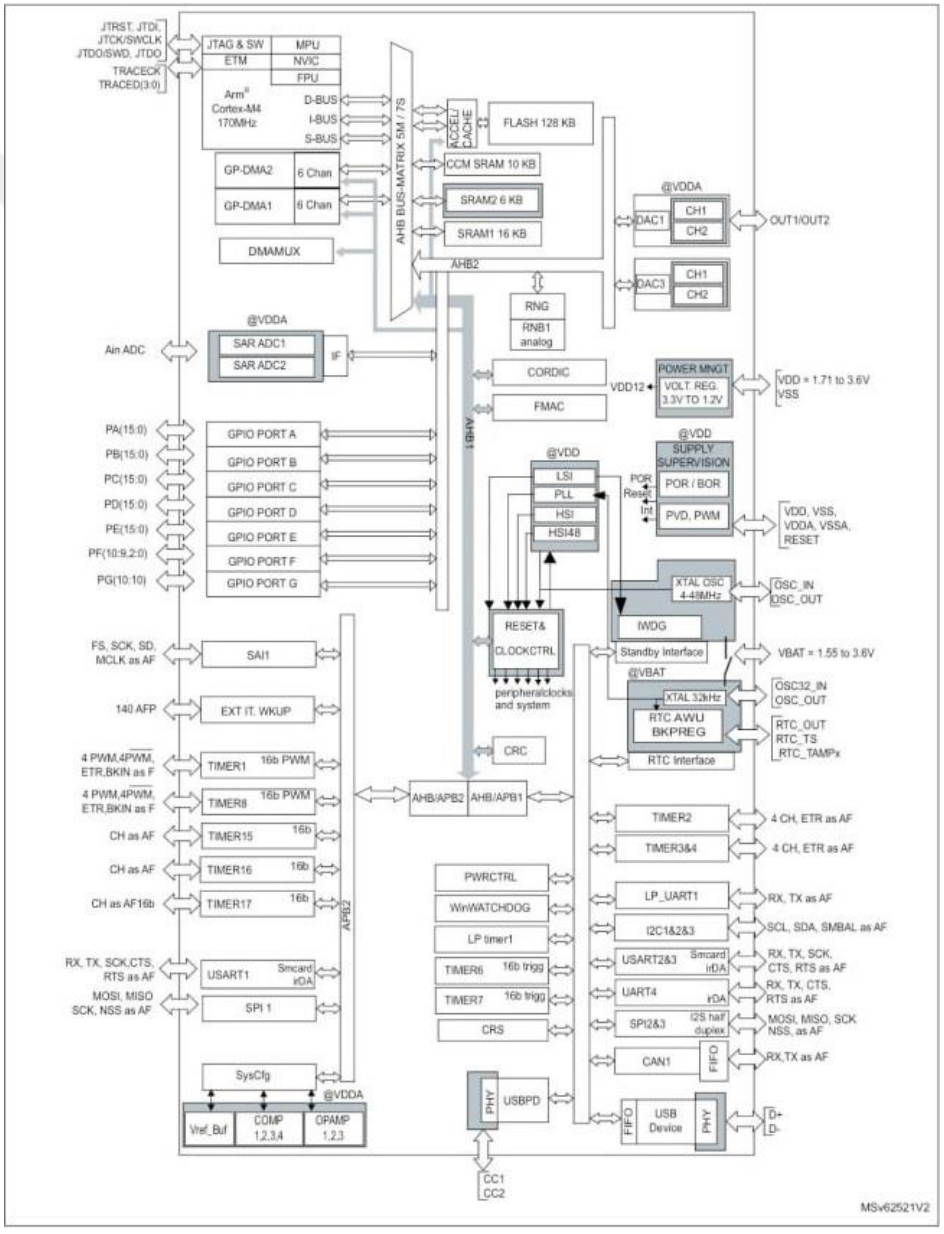


Figure 4.38. STM32G431RB Block Diagram

Furthermore, the STM32G431RB incorporates various protection mechanisms for embedded Flash memory and SRAM, such as readout protection, write protection, securable memory area, and proprietary code readout protection.

The device features peripherals for accelerating mathematical and arithmetic functions, including CORDIC (Coordinate Rotation Digital Computer) for trigonometric functions. Particularly, CORDIC provided precise results for us during sinusoidal reference voltage generation for power switches.

STM32G431RB provides two fast 12-bit ADCs (4 Msps), four comparators, three operational amplifiers, four DAC (Digital-to-analog Converter) channels, an internal voltage reference buffer, a low-power RTC, general-purpose timers, PWM timers dedicated to motor control, and various communication interfaces including I2C, SPI, UART, USB (universal serial bus), and more. Additionally, it supports independent power supplies, including an analog independent supply input for ADC, DAC, OPAMPs, and comparators.

The control technique we have developed requires multiple sensitive and high-quality ADC blocks for high-accuracy measurements, precise timer blocks to implement the hardware and generate switch signals at high speed, reliable DAC blocks for verification tests, a high number of GPIOs and various communication interfaces for communication with the external world. With all its features, interfaces and functional blocks and relatively cheaper price, STM32G431RB stood out as the most suitable candidate in terms of price-performance to implement our design.

In addition, the STM32G431RB incorporates ST state-of-the-art patented technology, including a Cortex-M4 CPU with FPU, ART Accelerator for zero-wait-state execution from Flash memory, mathematical hardware accelerators, memories with ECC support, reset and supply management, clock management, interconnect matrix, DMA controller, and various communication interfaces.

Development support includes serial wire debug (SWD) and Join Test Action Group (JTAG). In this thesis study, we programmed STM32G431RB via SWD feature using mini-USB connector of the NucleoG31RB board that we employ as control board.

4.3.1.1. NucleoG431RB

Since our hardware development and embedded software development activities were quite intense in our thesis study, we preferred to use a ready-made solution for microprocessor and microprocessor peripherals design. For this purpose, we decided to configure and program the NucleoG431[59] board based on the STM32G431 microprocessor as the main control board. We progressed our hardware design activities by taking this fact into consideration. For example, using the Nucleo board as a control card required having the mating connectors of the 38-pin headers on the Nucleo as the microprocessor and power electronics connection connector on the power board. The STM32G431 Nucleo-64 board is built on the MB1367 reference board. This board offers an accessible and adaptable platform for designers to experiment with new ideas and create prototypes using the STM32G4 series microcontrollers. Figure 4.39 shows the front and back views of the NucleoG431RB board, which is based on the STM32G431RB microprocessor and used as the main control unit in this thesis.

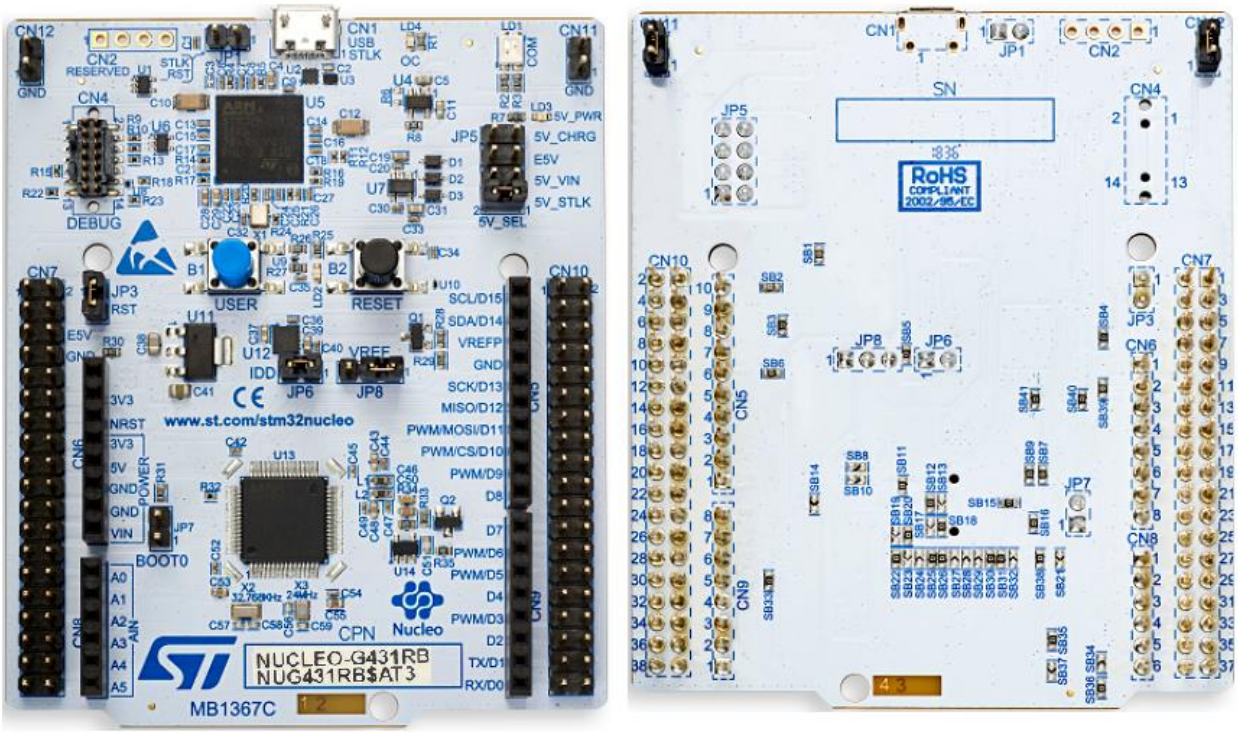


Figure 4.39. Physical Appearance of NUCLEOG431RB Development Board

With ST morpho headers, this Nucleo board facilitates easy expansion of functionality. This open development platform supports a wide range of specialized shields and features, enabling designers

to explore various applications. One notable feature of the STM32G4 Nucleo-64 boards is the integration of the STLINK-V3E debugger/programmer, eliminating the need for a separate probe. This streamlines the development process and enhances convenience for developers.

Here are some of the most important features of the NucleoG431RB that facilitate programming and debugging:

- 1 user programmable LED
- 1 user programmable and 1 reset push-buttons
- 32.768 KHz LSE (Low-speed External Clock) crystal oscillator
- 24 MHz HSE (High-speed External Clock) on-board oscillator
- USB connector with Micro-AB
- ST morpho extension pin headers for full access to all STM32G4 I/Os
- Flexible power-supply options: ST-LINK, USB VBUS, or external sources

24Mhz HSE on-board oscillator is used as the main source to obtain 168Mhz of system frequency. And we use the micro-USB connector to obtain computer connection to program and debug the embedded software. Finally, we used ST morpho extension pin headers as an easy and noise-free connection to our power electronics circuit and all other hardware on the power card.

We previously shared the power tree on the power card in Section 4.1.7. The power tree of NucleoG431 is also given in Figure 4.40 below. According to this scheme, it is possible to power the controls board both via USB and the power board. We designed the hardware of our system so that the control card is powered by USB via the computer during debugging and programming processes, and during continuous operation, it is powered by the 5V isolated source produced on the power card.

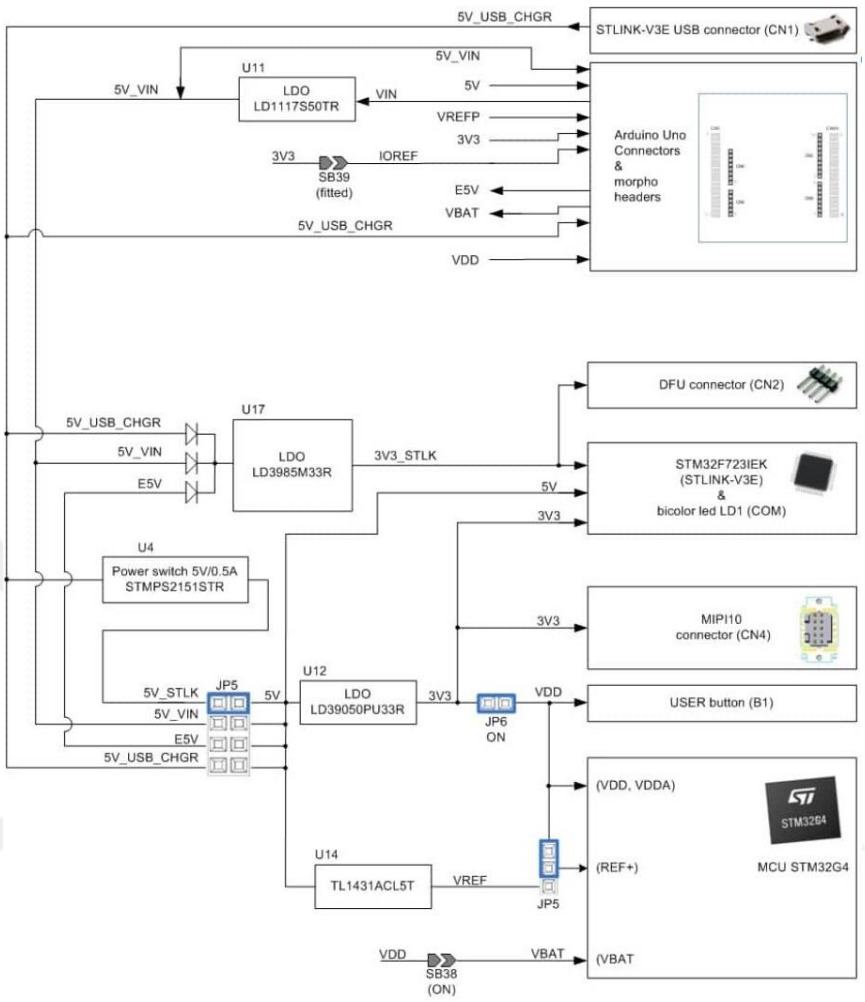


Figure 4.40. Power Tree of NUCLEOG431RB

The Nucleo board supports a wide range of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE. We utilized the STM32CubeIDE [60] software, during the development of our application.

4.3.2. Determination of necessary I/Os and I/O types

The first step in programming the control algorithm was to determine the modules and the number of these modules that would be necessary for the 3-phase inverter hardware to function properly. For this purpose, the adequacy of the target microprocessor in terms of I/O number and peripheral diversity was verified for our design needs. The I/Os distributed from the STM32G431 processor to various parts on the NucleoG431 are as follows: 2x 32KHz on-board oscillator pins, 2x RCC (Reset and Clock Control) oscillator pins, 3x ST-Link connection pins, 1x push button pin and

necessary supply and reference voltage pins. For simplicity, internal connections of the NucleoG431RB are not further explained in this section.

Table 4.12 below shows the configuration and functions of some important pins in terms of hardware design.

Table 4.12. Pin Configurations of Main Functions

Function Name	I/O Type	MCU Module	Module Channel	MCU Pin Name
High Switch 1 PWM Signal	O	TIMER1	CH1	PA8
Low Switch 1 PWM Signal	O	TIMER1	CH1N	PB13
High Switch 2 PWM Signal	O	TIMER1	CH2	PA9
Low Switch 2 PWM Signal	O	TIMER1	CH2N	PB14
High Switch 3 PWM Signal	O	TIMER1	CH3	PA10
Low Switch 3 PWM Signal	O	TIMER1	CH3N	PB15
Phase 1 Output Voltage	I	ADC1	CH6	PC0
Phase 2 Output Voltage	I	ADC1	CH7	PC1
Phase 3 Output Voltage	I	ADC1	CH8	PC2
DC Busbar + Voltage	I	ADC1	CH3	PA2
DC Busbar - Voltage	I	ADC1	CH4	PA3
Phase 1 Output Current	I	ADC2	CH9	PC3
Phase 2 Output Current	I	ADC2	CH1	PA0
Phase 3 Output Current	I	ADC2	CH2	PA1
DC Busbar + Current	I	ADC2	CH3	PA6
DC Busbar - Current	I	ADC2	CH4	PA7
External Communication Transmit	O	USART1	TX	PC4
External Communication Receive	I	USART1	RX	PC5
EEPROM Connection (Optional)	O	SPI	SCK	PC10
EEPROM Connection (Optional)	I	SPI	MISO	PC11
EEPROM Connection (Optional)	O	SPI	MOSI	PC12
Validation Signal 1	O	DAC1	CH1	PA4
Validation Signal 2	O	DAC1	CH2	PA5

The I/Os distributed from NucleoG431 to the power card via ST Morpho connectors are as follows; 6x timer pin, 12x ADC pin, 2x UART pin (1 transmit, 1 receive), 3x SPI (Serial Peripheral Interface) pin (MOSI, MISO and CLK pins), several general-purpose inputs and outputs to provide miscellaneous functions (e.g. enable/disable signal or fault flags).

All I/O configurations made in the STMCubeIDE environment, according to the needs of our power electronics circuit on the STM32G431RB processor, can be accessed in Figure 4.41.

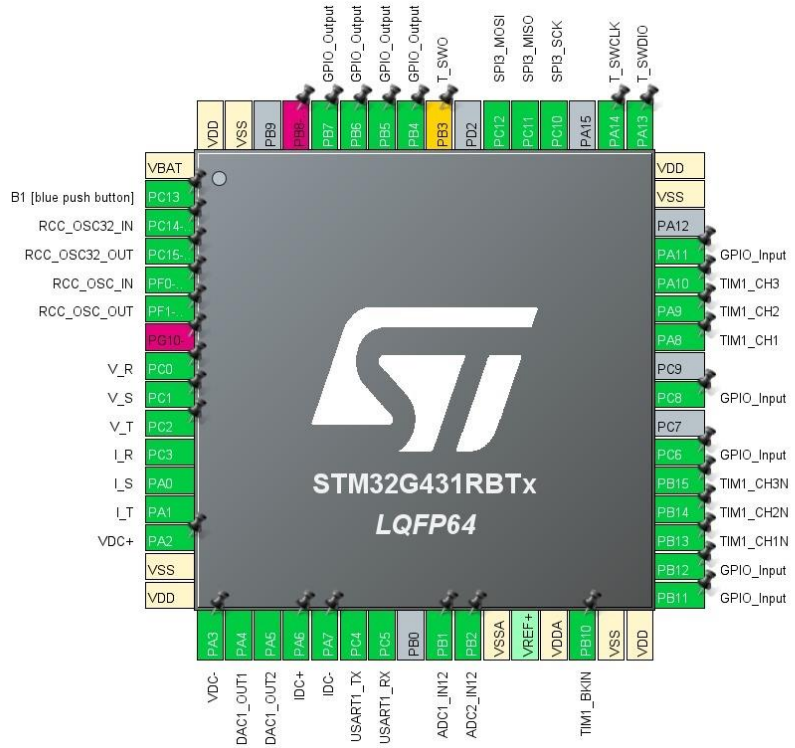


Figure 4.41. STM32G431RB Configuration for 3-Phase VSI Implementation

Once all I/O types, their amounts and placements have been clarified, the next step is to determine the maximum oversampling capacity that can be achieved with this configuration. We include this part in the next section.

4.3.3. Calculation of ADC oversampling capacity

This thesis stands out from alternative studies due to its approach to embedded control software development. As we mentioned in the previous sections, the control technique we have developed has three basic pillars. Updating the RMS value by applying the sliding RMS method with the

interrupt frequency, improving regulation by applying PI control with the interrupt frequency, and eliminating noise by obtaining precise measurement with ADC oversampling. The success of this technique is directly proportional to the ADC oversampling capacity. For this, before starting the programming phase, the oversampling capacity, which is a function of the system frequency, fundamental frequency and the number and channel of ADC modules used, must be calculated accurately. The basic parameters that need to be known to determine the ADC oversampling rate are as follows;

- Interrupt (sampling) frequency (f_{sw}): 100KHz
- Microprocessor system frequency (f_{MCU}): 164MHz
- ADC module frequency (f_{ADC}): 41MHz
- Number of ADC modules within the MCU: 2
- Number of channels within each ADC module: 5
- Duration for 1 complete ADC measurement: 19 cycles

Among these parameters, we set the interrupt frequency as 10KHz in line with our hardware design goals. We set the microprocessor frequency to 164MHz, which is the most stable and highest frequency possible within the microprocessor limits. The speed of the ADC modules was set to 41MHz, which is one fourth of the system frequency, again within the MCU limits.

STM32G4 offers us a total of 12 12-bit ADC channels, 6 in one module and 6 in the other. In line with our design, it was the most logical design decision to distribute the ADCs we needed equally to both modules as 5 channels each. We obtained the time taken for a complete ADC measurement from the reference manual of the microprocessor we used. According to this;

$$T_{CONV} = Sampling\ time + 12.5\ ADC\ clock\ cycles \tag{4.22}$$

We obtained the sampling time from the MCU's datasheet (Figure 4.42). Although this period is 2.5 cycles for fast channels, we used the value 6.5 in the calculation because not all of the ADC channels we used are fast channels. According to Equation (4.22), the conversion time was calculated as 19 per cycle.

Resolution	Sampling cycle @60 MHz	Sampling time [ns]	R_{AIN} max (Ω)	
			Fast channels(3)	Slow channels(4)
12 bits	2.5	41.67	100	N/A
	6.5	108.33	330	100
	12.5	208.33	680	470
	24.5	408.33	1500	1200
	47.5	791.67	2200	1800
	92.5	1541.67	4700	3900
	247.5	4125	12000	10000
	640.5	10675	39000	33000

Figure 4.42. Sampling Time for ADC Modules of STM32G431RB

The time equivalent of an ADC conversion is calculated as 463ns as per Equation (4.23).

$$T_{CONV(\text{sec})} = T_{CONV} * \left(\frac{1}{f_{ADC}}\right) \tag{4.23}$$

Since we use 5 channels in each ADC module, the total time is calculated as 2.317us according to Equation (4.24).

$$T_{CONV_{total(\text{sec})}} = 5 * T_{CONV(\text{sec})} \tag{4.24}$$

Each interrupt period lasts 100µs. Accordingly, the number of times each channel can be oversampled is calculated by rounding the value obtained in equation 4 down to the power of 2.

$$max_{OVS} = ROUNDDOWN\left(\frac{t_{SW}}{T_{CONV_{total(\text{sec})}}}\right) \tag{4.25}$$

Hence, the maximum oversampling rate that can be achieved per ADC channel in this system is calculated as 32 (2^5).

$$t_{ADC_{total}} = T_{CONV_{total(\text{sec})}} * max_{OVS} \tag{4.26}$$

The total time required for 32x oversampling of 5 channels separately for each ADC module is calculated as 74.08us according to equation 5. Since an interrupt period is 100µs, this result leaves enough space before the current interrupt period completely elapses. So, the RMS measurement and PI controller algorithm can be implemented after all the samples obtained successfully before the next interrupt period, which is sufficient to implement the proposed control technique.

So, we validated that STM32G4 can provide 32x oversampled ADC readings for each of the five channels of each particular ADC module which can drastically improve the sample accuracy.

4.3.4. Writing bare-metal driver modules

4.3.4.1. Rationale behind choosing bare-metal programming

The HAL (Hardware Abstract Language) or LL (Low Level) libraries, which are offered free of charge by ST company and are readily available in many modules and drivers, could be chosen for embedded programming, as they provide ease of programming while carrying out the thesis work. The HAL provides high-level and feature-focused APIs, ensuring a high level of portability. It abstracts away the intricacies of the microcontroller unit and peripherals from end-users. The LL interface offers APIs (Application Programming Interface) at the register level, prioritizing optimization over portability. On the other hand, even LL could not provide us with the deeper hardware understanding at the register level that bare-metal programming provides. Below we have briefly listed the advantages of bare-metal programming over using ready-made libraries;

- Enhanced control: Writing code at the register level provides direct access to hardware, enabling finer control and optimized performance. This allows developers to finely tune the behavior of the system for specific requirements, resulting in optimal performance.
- Reduced memory usage: By writing register-based code, designers can minimize memory consumption by only including necessary features, thus utilizing system resources more efficiently. This leads to more efficient memory usage, which is particularly important in embedded systems with limited memory resources.
- Reduced dependencies: Register-based coding reduces dependencies on library versions or updates, allowing for better adaptation to hardware changes or updates. This enhances the flexibility and maintainability of the codebase, as it is less reliant on external dependencies.
- Deeper hardware understanding: Writing code directly at the register level helps developers gain a deeper understanding of how hardware units operate. This promotes a more thorough understanding of the underlying hardware architecture, which can be beneficial for debugging, optimization, and future development efforts.

With all these facts, bare-metal programming is not without some risks and disadvantages;

- Increased development time: Writing register-based code typically requires more development time as developers need to create everything from scratch.
- Limited portability: Register-based code is often tied to specific hardware architectures or microcontrollers, requiring more adaptation when porting to different platforms.

However, the mentioned risks and disadvantages are not considered to be at a level that would prevent embedded control with bare-metal programming in this thesis study. The only two sources taken as reference for all bare-metal source codes written within the scope of this thesis are the datasheet document of the STM32G431RB microprocessor and its reference manual.

4.3.4.2. Embedded algorithm programming process

We programmed the proposed control algorithm using bare-metal written driver modules using STM32CubeIDE without any third-party library. Doing this required writing and verifying each module to be used in programming separately. Throughout the entire bare-metal programming process, our sole supportive resources were the STM32G431 datasheet and reference manual [61]. During the development of the embedded software, modules such as RCC, SYSTICK timer, regular timer, ADC, DAC, UART and more were written and verified separately.

The below code snippet is a small example of how ADC sequence length and order are established in bare-metal programming

```
//7- Set the conversion sequence length
ADC2->SQR1 = ADC_SEQ_LEN_1;        // We only have single channel as Ch.4
//8- Set the conversion sequence start channel
ADC2->SQR1 |= CH4_RANK1;           // Set CH4 as first sequence
```

Complete bare-metal written ADC reading code example is given in the appendix. Based on this example module driver, in the *"main.c"* section, the headers of the processor used in the first stage and all the drivers used were included. Then, the necessary variable assignments were made, and finally, the *"uart1_tx_init()", "adc_init()", "activate_adc()"* and *"start_conversion"* functions were called within the *"int main()"* function, respectively. The main aim of the function is to ensure that the value read from the sensor with the ADC module is printed on the computer screen via UART.

It is useful to summarize the operation of the *"adc.c"* code fragment in general terms. At the beginning of the code, macros for the registers to be used were created for ease of use. Then, the ADC reading task is divided into different tasks, that is, functions. In order to use a microprocessor module, the relevant pin and the relevant peripheral need to be configured first. The first part of the code consists of this configuration.

After the ADC is configured ready for operation, it must first be activated and then the conversion must be started. In the last section, a code section has been added to clear the necessary flags and complete the reading.

Finally, we emphasize the code part of the "*adc.h"* file, which carries the headers of the ADC module. Each *"C"* code is used with a header file that carries the functions inside of it. This does not only increase code readability, but also facilitates modular code writing and allows a driver and program created once to be used many times.

Figure 4.43 shows an image of the continuous ADC readings taken on PA7 pin on the Nucleo board with this bare-metal code, when the pin is left floating. The ADC values in the image are obtained by printing the ADC values data transferred via UART connection to the console screen of Realterm [62] serial capture software.

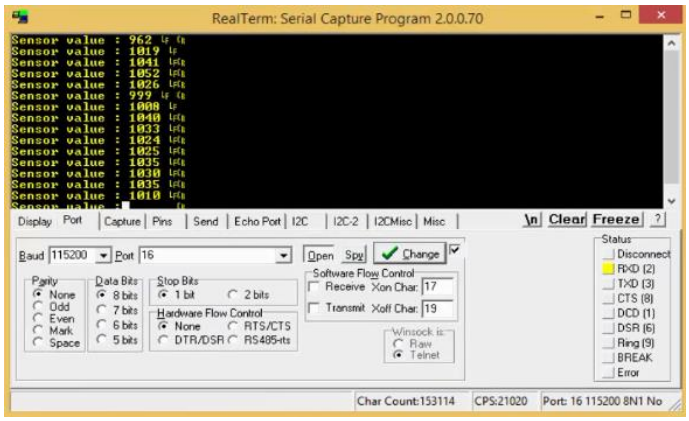


Figure 4.43. Printing ADC Read Value to Computer Screen

A floating ADC pin printing random values to the screen is a situation we expect in this test. As understood from this section, what is meant by bare-metal programming is coding at the register level. In the following sections, we discuss step by step how to program the control algorithm applied in the thesis.

4.3.5. Developing embedded control algorithm

The proposed control technique is based on oversampling on ADC readings, calculating the RMS value after each interrupt cycle and updating the modulation index in each interrupt period. In Section 3.5, we examined these three main pillars that form our algorithm, in detail. Here, we explain how they are brought together and how they are reflected in the software domain.

In general, when the output loads of power supplies are suddenly increased, they tend to experience an instant undershoot in voltage amplitudes and then a permanent decrease in the steady state voltage. Similarly, they experience a sudden overshoot in the output voltage in case of sudden load drops and a permanent increase in the voltage value in the steady state. The embedded control method we developed reduces these overshoot and undershoot amplitudes, provides a quick recovery to the target RMS value, and ensures that the target RMS value is achieved at least as efficiently as the classical method without sacrificing THD in steady state.

In closed loop, voltage regulation is tried to be achieved by simply increasing the modulation index when the modulation load current increases. Similarly, when the load decreases, the target RMS value is tried to be achieved by decreasing the modulation index. Effectiveness of the closed loop control is the main subject of the control technique proposed in this thesis.

In each control method, activating closed loop control immediately may cause stability problems. The first phase of our control method shown in Figure 4.44 starts by allowing the system to operate open loop for a certain period. Although the open-loop system is not optimal in terms of efficiency and regulation, it is stable as long as there is no change in the load and input voltage.

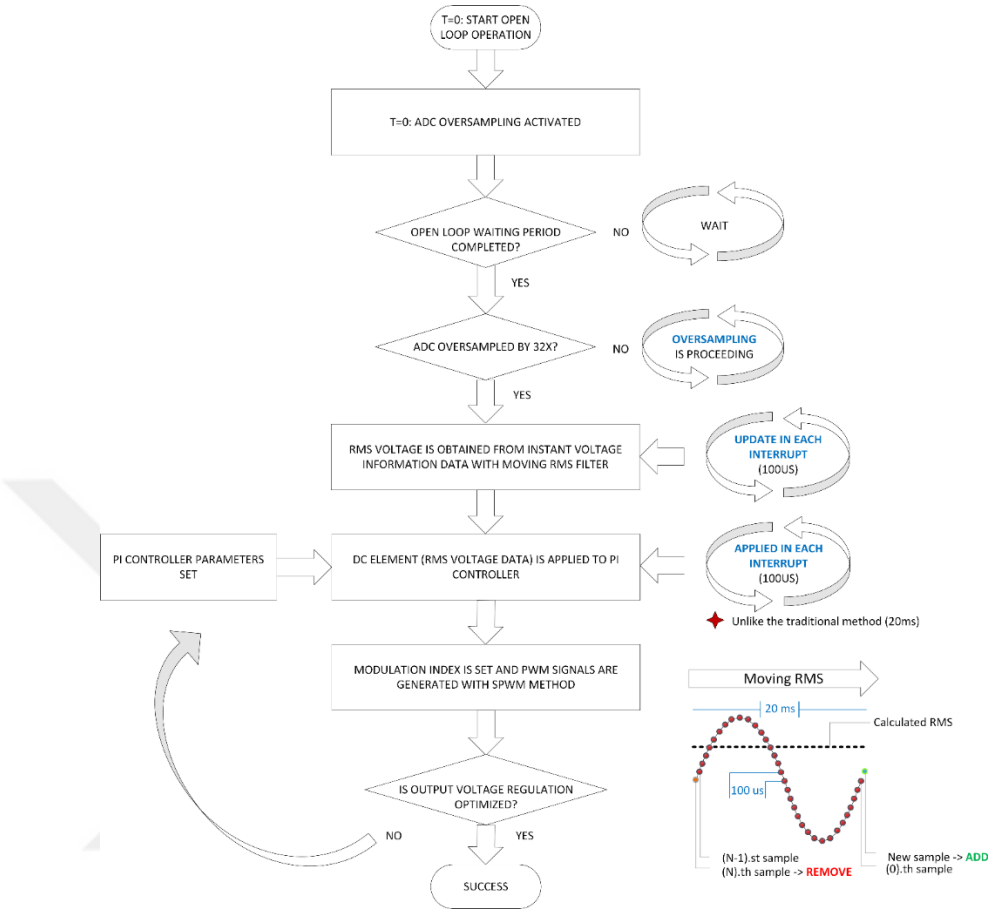


Figure 4.44. Flowchart of the Developed Novel Control Algorithm

In this technique, ADC oversampling is activated even during the system is running open loop. Each sample is processed in the microprocessor by oversampling 32x, which is the maximum rate we calculated according to our system parameters and microprocessor capability. In other words, a noise-free and pure measurement value is obtained without any time shift and without applying any low pass filter. Oversampling is given to the system as a configuration from the startup and is never turned off.

The sliding RMS calculation function starts by giving the very first sample (t=100 µs) read and oversampled from each ADC channel as input to the RMS calculation string, and the first RMS data is created after the first 20 milliseconds regardless of the system operation (open-loop or closed-loop). Another contribution of our proposed method begins at this point. The RMS calculated in the first 20 ms is updated with moving RMS function as soon as the next 100 µs passes, which is t=20.1 ms. Meanwhile, the sample at t=0.1 ms was removed from the RMS

calculation. The calculated RMS value at 20.2 ms does not include the 2 sample values from t=0s to t=200 µs. We explained this situation in section 3.5.3, in detail. The implementation of this algorithm is achieved by reflecting Equation (3.10) into the software environment.

After a certain time of open-loop operation and the output voltage is stabilized, the closed loop operation is being started. In each interrupt period, the updated RMS value is compared with the reference RMS value and the PI corrective output is created and applied to the system every 100µs to create a new PWM signal. Basically, this corrective output arranges the modulation index which is the amplitude of the reference sinusoidal. In this way, the third contribution of the proposed method is realized.

With this method, the details of which are illustrated in the flowchart above, we improved the dynamic response significantly by giving the controller time to start correcting an error as soon as possible; in addition, if an error occurs as a transient and disappears on its own, this method prevents a modulation index from being sent to the controller, which would disrupt the regulation. In other words, this control method was appropriate for situations where abrupt load changes could disrupt voltage regulation, and where precision in sampling is crucial for maintaining stable voltage regulation.

In this section, we talked about how the control algorithm works in general. In the following sections, we will try to explain how the technique developed on the microprocessor is implemented through code snippets.

4.3.6. Programming embedded control algorithm

It was not deemed appropriate to share all module, sub-module, driver and header codes used in the designed 3-phase inverter control, as it is both impractical and may reduce understandability. Instead, we will include 4 main headings, which are the building blocks of the project, at the code detail level. These are the configuration of the NucleoG431RB board, implementation of the sliding RMS filter, application of oversampling and finally writing the PI control code.

4.3.6.1. Configuration of Nucleo-G431RB

In this section, we see how the I/Os specified in section 4.3.2 are configured with register level codes, and how the pin functions and operating modes are set.

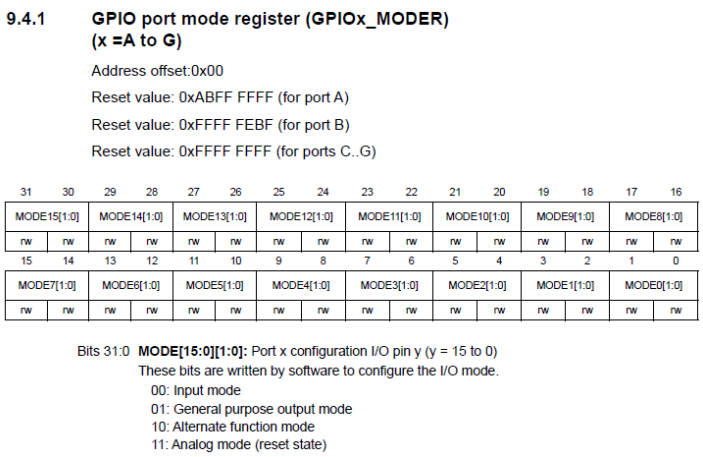
In the first stage, we activate the GPIOs to which all the modules to be used are connected to the relevant peripheral bus and the clocks of all the modules we will use during programming. For example, according to the processor block diagram in Figure 4.41, while the GPIOA module and ADC module are connected to the AHB2 bus, the TIM1 module is connected to the APB2 bus and the CORDIC module is connected to the AHB1 bus. These connections not only vary between different microprocessor manufacturers but can even differ between ST-manufactured processors. The relevant pseudo code is shared below.

```
SET_BIT(RCC->AHB1ENR, RCC_AHB1ENR_DMA1EN);

SET_BIT(RCC->AHB2ENR, RCC_AHB2ENR_GPIOAEN);

SET_BIT(RCC->AHB2ENR, RCC_AHB2ENR_ADC12EN);

SET_BIT(RCC->AHB2ENR, RCC_AHB2ENR_DAC1EN);

SET_BIT(RCC->APB2ENR, RCC_APB2ENR_TIM1EN);
```

In the next stage, we set the mode in which each I/O will be used for all GPIO ports (GPIOA, GPIOB and GPIOC) to be used. The relevant pseudo code part is as follows.

```
SET_BIT(GPIOA->MODER, GPIO_MODER_MODE0_1);

SET_BIT(GPIOA->MODER, GPIO_MODER_MODE0_0); //PA0 Analog Mode - ADC2_IN1

SET_BIT(GPIOA->MODER, GPIO_MODER_MODE8_1);

CLEAR_BIT(GPIOA->MODER, GPIO_MODER_MODE8_0); //PA8 Altr. Func. - TIM1_CH1

CLEAR_BIT(GPIOB->MODER, GPIO_MODER_MODE4_1);

SET_BIT(GPIOB->MODER, GPIO_MODER_MODE4_0); //PB4 - Gen Purp Output Mode
```

For pins to be used for general purposes, the port mode register of the relevant GPIO is set according to whether it is input or output. Since our ADC and DAC ports are analog pins, they are set as analog mode in the port mode register. In Figure 4.45, reference manual information on how to set each port mode in the GPIO port mode register is shared.



9.4.1 **GPIO port mode register (GPIOx_MODER) (x =A to G)**

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF (for ports C..G)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODE[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Figure 4.45. GPIO Port Mode Register

Alternate function registers are located in the STM32G431 for the ports used for timer ports and communication ports like SPI, UART etc. While these pins are set as alternative functions in the port mode register, in the next step, the alternative function type is set in the “AFRL” and “AFRH” registers. The relevant pseudo code is shared below.

```
// PA8 -->     TIM1_CH1 (AF6) - 0b0110

CLEAR_BIT(GPIOA->AFRH, GPIO_AFRH_AFSEL8_3);

SET_BIT(GPIOA->AFRH, GPIO_AFRH_AFSEL8_2);

SET_BIT(GPIOA->AFRH, GPIO_AFRH_AFSEL8_1);

CLEAR_BIT(GPIOA->AFRH, GPIO_AFRH_AFSEL8_0);
```

We find information on how to set the alternate function type from the alternate function table in the processor datasheet. How these selections can be made for GPIOA is shown in Figure 4.46.

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	LPTIM1/TIM2/6/16/17	I2C3/TIM1/2/3/4/8/16/GPCOMP1	I2C3/SAI1/USB/TIM8/16/GPCOMP3	I2C1/2/3/TIM1/8/16/17	SPI1/2/3/I2S2/3/UART4/TIM8/Infrared	SPI2/3/I2S2/3/TIM1/8/Infrared	USART1/2/3	I2C3/4/UART4/LPUART1/GPCOMP1/2/3	TIM1/8/16/FDCAN1	TIM2/3/4/8/17	LPTIM1/TIM1/8/FDCAN1	LPUART1/SAI1/TIM1	SAI1/OPAMP2	UART4/SAI1/TIM2/16/UCPD1	EVENT
PortA	PA0	-	TIM2_CH1	-	-	-	-	-	USART2_CTS	COMP1_OUT	TIM8_BKIN	TIM8_ETR	-	-	-	TIM2_ETR	EVENT OUT
	PA1	RTC_REFIN	TIM2_CH2	-	-	-	-	-	USART2_RTS_DE	-	TIM15_CH1N	-	-	-	-	-	EVENT OUT
	PA2	-	TIM2_CH3	-	-	-	-	-	USART2_TX	COMP2_OUT	TIM15_CH1	-	-	LPUART1_TX	-	UCPD1_FRSTX	EVENT OUT
	PA3	-	TIM2_CH4	-	SAI1_CK1	-	-	-	USART2_RX	-	TIM15_CH2	-	-	LPUART1_RX	SAI1_MCLK_A	-	EVENT OUT
	PA4	-	-	TIM3_CH2	-	-	SPI1_NSS	SPI3_NSS/I2S3_WS	USART2_CK	-	-	-	-	-	SAI1_FS_B	-	EVENT OUT
	PA5	-	TIM2_CH1	TIM2_ETR	-	-	SPI1_SCK	-	-	-	-	-	-	-	-	UCPD1_FRSTX	EVENT OUT
	PA6	-	TIM16_CH1	TIM3_CH1	-	TIM8_BKIN	SPI1_MISO	TIM1_BKIN	-	COMP1_OUT	-	-	-	LPUART1_CTS	-	-	EVENT OUT
	PA7	-	TIM17_CH1	TIM3_CH2	-	TIM8_CH1N	SPI1_MOSI	TIM1_CH1N	-	COMP2_OUT	-	-	-	-	-	UCPD1_FRSTX	EVENT OUT
	PA8	MCO	-	I2C3_SCL	-	I2C2_SDA	I2S2_MCK	TIM1_CH1	USART1_CK	-	-	TIM4_ETR	-	SAI1_CK2	-	SAI1_SCK_A	EVENT OUT
	PA9	-	-	I2C3_SMBA	-	I2C2_SCL	I2S3_MCK	TIM1_CH2	USART1_TX	-	TIM15_BKIN	TIM2_CH3	-	-	-	SAI1_FS_A	EVENT OUT
	PA10	-	TIM17_BKIN	-	USB_CRS_SYNC	I2C2_SMBA	SPI2_MISO	TIM1_CH3	USART1_RX	-	-	TIM2_CH4	TIM8_BKIN	SAI1_D1	-	SAI1_SD_A	EVENT OUT
	PA11	-	-	-	-	-	SPI2_MOSI/I2S2_SD	TIM1_CH1N	USART1_CTS	COMP1_OUT	FDCAN1_RX	TIM4_CH1	TIM1_CH4	TIM1_BKIN2	-	-	EVENT OUT
	PA12	-	TIM16_CH1	-	-	-	I2SCKIN	TIM1_CH2N	USART1_RTS_DE	COMP2_OUT	FDCAN1_TX	TIM4_CH2	TIM1_ETR	-	-	-	EVENT OUT
	PA13	SWDIO-JTMS	TIM16_CH1N	-	-	I2C1_SCL	IR_OUT	-	USART3_CTS	-	-	TIM4_CH3	-	-	SAI1_SD_B	-	EVENT OUT
	PA14	SWCLK-JTCK	LPTIM1_OUT	-	-	I2C1_SDA	TIM8_CH2	TIM1_BKIN	USART2_TX	-	-	-	-	-	SAI1_FS_B	-	EVENT OUT
	PA15	JTDI	TIM2_CH1	TIM8_CH1	-	I2C1_SCL	SPI1_NSS	SPI3_NSS/I2S3_WS	USART2_RX	UART4_RTS_DE	TIM1_BKIN	-	-	-	-	TIM2_ETR	EVENT OUT

Figure 4.46. Alternate Functions of GPIOA Module Pins

For example, according to table given in Figure 4.46, in order to use the PA8 port with the TIM1_CH1 function, "0b0110" value must be set in the AF6 register. How to set a value to the AFRL register is shown in Figure 4.47. The only difference between AFRL and AFRH registers is pin numbers. GPIO pins from 0 to 7 are set using the AFRL register, while pins from 8 to 15 are set using the ARFH register. This data is also available in the reference manual as in Figure 4.47.

9.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to G)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

(a)

Bits 31:0 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

(b)

Figure 4.47. (a) GPIO Alternate Function Register (b) Instructions to Configure the AFRL[] Register

The next step is to configure all modules to be used in the *"main.c"* file separately. In *"main.c"*, functions are implemented through the configuration of the system clock, the configuration of the ADC modules, the configuration of the UART module, inclusion of the header files, variable assignments and many other module configurations. In order to keep the thesis content short and understandable, it is sufficient to include only the configuration of timer number 1 in this section, which will give an idea of how a microprocessor module is configured. The pseudo code containing Timer 1 configuration is given below.

```
void TIM1_Config(void)
{
 TIM1->SR=0;
 TIM1->ARR=8200;
 TIM1->PSC=0;                    //  PWM timer is set to 10 KHz
 TIM1->CNT=0;
 TIM1->RCR=1;
 TIM1->CCER=0x00000555;     //  TIM1 [CH1 : CH3] PWM outputs are enabled
 …
}
```

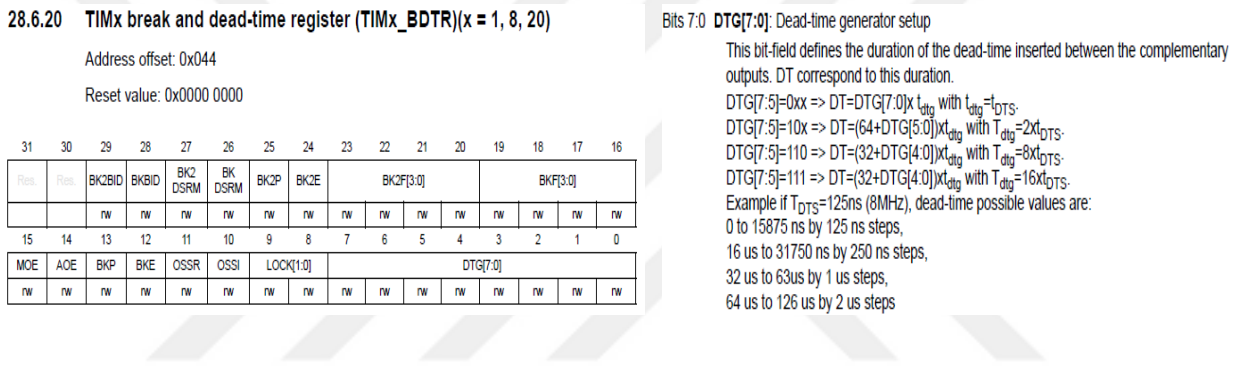
Each line of the above pseudo code is important to produce the PWM signal at the correct frequency and without any frequency shift. Here we will touch on the most important parts of the TIMER configuration. One of these is setting the TIMER frequency, the other is setting the dead-time and the last one is activating the timer interrupt. PWM frequency (f_{PWM}) is set as follows according to the Equation (4.27).

$$f_{PWM} = \left(\frac{f_{MCU}}{\left(\frac{1}{TIMx_ARR + 1}\right) * \left(\frac{1}{TIMx_PSC + 1}\right)} \right) \tag{4.27}$$

We determined the system frequency as 164MHz in RCC configuration step. Since we selected the "TIMx_ARR" register value as '0' and the "TIMx_PSC" register as '8100', the PWM

frequency is obtained as 20 KHz from the formula above. Since we configured the PWM timer as high side and low side switches operate complementary, the switching frequency is set as half of the PWM frequency, that is, 10 KHz.

We have previously mentioned the effect of setting the dead-time correctly on the system efficiency and the success of the control algorithm. Although finding the correct hardware duration is based on a theoretical calculation, it is an iterative and quite complex problem. However, deadtime is relatively simple to implement in software. The "TIMx_BDTR" register of the relevant timer is a register specifically designated for deadtime adjustments (Figure 4.48).



28.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8, 20)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with t_{dtg}=t_{DTS}.

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with T_{dtg}=2xt_{DTS}.

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=8xt_{DTS}.

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with T_{dtg}=16xt_{DTS}.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

(a) (b)

Figure 4.48. (a) TIMx_BDTR Register (b) Instructions to Configure the TIMx_BDTR Register

The equation given for dead time calculation (Equation (4.28)) in the reference manual of the microprocessor used is as follows.

$$DTG[7:5] = 0xx \Rightarrow DT = DTG[7:0] * tDTG \tag{4.28}$$

Accordingly, setting the first 8 bits of the relevant register, that is, the "DTG[7:0]" bits, as '0x32' (decimal equivalent is '50') leaves us 305ns of dead time while generating PWM signals for the power switches. In this expression 't_{DTG}' is equal to 't_{DTS}', and 't_{DTS}' equals to '$1/f_{MCU}$'.

Although all configurations are made within the main function, the body of the control algorithm runs within the timer interrupt. Therefore, it is important to set and prioritize this interrupt properly. The order of prioritization is as follows; ADC readings are at the highest priority level, followed by timer interrupt.

The *"NVIC_SetPriority()"* function is a special function for the STM32G431, a processor of the Cortex M4 family, and is used to set interrupt priorities. "TIM1_UP_TIM16_IRQn" register is a special register determined by the core manufacturer to enable the interrupt setting of each timer from Timer 1 to Timer 6. In the last part, the priority of this interrupt is set to 2. This means that the timer interrupt is set to a lower priority level than the interrupt with priority number 1 (ADC interrupt). Interrupt is also enabled with the *"NVIC_EnableIRQn()"* function, which is another special function for Cortex M4 cores.

As we mentioned at the beginning of this section, what is done in the *"main.c"* section includes, but is not limited to, the configurations described above. However, this detailed information can only be implemented if the microprocessor datasheet and reference manual are carefully followed.

4.3.6.2. Writing the ADC oversampling code

To better understand the issue of ADC oversampling, it should be discussed together with the ADC configuration. For this reason, we explain this section through the relevant code snippet in *"main.c*" where the ADC configuration is made. For the STM32G4 processor family, ADC1 and ADC2 modules can be used together (dual-operation) or separately (single-operation). For this reason, there is a section where common configurations of these two ADCs are made, but we will skip this section in order to focus on oversampling.

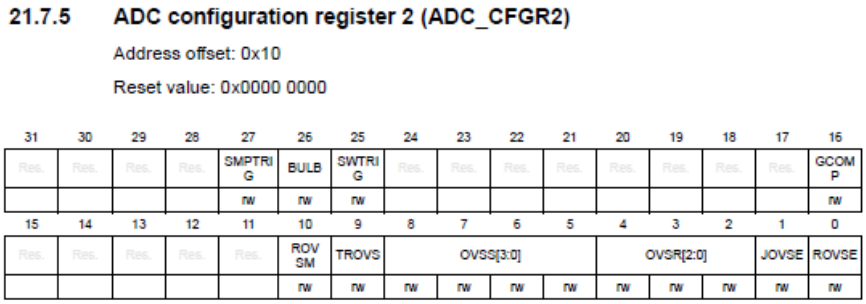
In the previous sections, we stated that we divided five channels of each of two ADC modules in the system between different measurements. The two ADC module configurations are exactly the same except for the selected processor pins. Therefore, in this section, it is sufficient to examine the configuration of the ADC1 module, only. The complete ADC1 module configuration is given in the appendix.

In the configuration, initially, the relevant ADC module is separated from the *"deep power down"* mode by clearing the corresponding register bit "ADC_CR_DEEPPWD", allowing it to be powered from the voltage regulator output by setting the "ADC_CR_ADVREGEN" bit. Subsequently, clearing (setting the bit as '1') all flags associated with the ADC module is a good common practice in bare-metal programming and is crucial for ensuring accurate ADC readings.

To properly configure ADC oversampling, two parameters need to be set. One is the oversampling ratio, and the other is the number of bits the read value should be shifted by. The shift amount must be compatible with the oversampling ratio for the microcontroller to accurately capture the decimal value it reads. For an oversampling ratio of '2^5', it's important for the shift ratio to be '5' to ensure the proper reading of the instantaneous voltage value in decimal. Associated pseudo code is given below;

```
/* Oversampling shift: 5-bits */
SET_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_0);
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_1);
SET_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_2);
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_3);
// Oversampling ratio: 32x
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSR_0);
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSR_1);
SET_BIT(ADC1->CFGR2, ADC_CFGR2_OVSR_2);
SET_BIT(ADC1->CFGR2, ADC_CFGR2_ROVSE);
```

The oversampling configuration is performed through the "ADC_CFGR2" register of each ADC (Figure 4.49).



21.7.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SMPTRIG	BULB	SWTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMP
				rw	rw	rw									rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 4.49. ADC Configuration Register-**2**

Bits [8:5] of this register are used to set the oversampling shift, while bits [4:2] are used to configure the oversampling ratio (Figure 4.50).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the result.

0000: No shift
0001: Shift 1-bit
0010: Shift 2-bits
0011: Shift 3-bits
0100: Shift 4-bits
0101: Shift 5-bits
0110: Shift 6-bits
0111: Shift 7-bits
1000: Shift 8-bits
Other codes reserved

(a)

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

000: 2x
001: 4x
010: 8x
011: 16x
100: 32x
101: 64x
110: 128x
111: 256x

(b)

Figure 4.50. (a) Oversampling Shift Configuration Instructions (b) Oversampling Rate Configuration Instructions

To achieve 32x (or 2^5x), oversampling the raw oversampling value should be shifted right by '5' bits. To accomplish this, according to the reference manual, the value "0101" should be set in bits [8:5] of the "CFGR2" register. The next step is to set the oversampling ratio in the microcontroller. For this purpose, setting "100" in bits [4:2] of this register is sufficient.

Following the necessary adjustments for calibrating the ADC module, the next step involves setting the number of channels to be converted and the sequence in which these channels will be converted. The number of conversion channels is set by configuring the first four bits [3:0] of the "ADC_SQR1" register, as indicated in Figure 4.51. The channel length is set to '5' channels in binary format by setting the relevant register to "0101".

21.7.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Figure 4.51. ADC Regular Sequence Register-1

Then, we use the same register to determine the sequence in which pins will be converted. While "ADC_SQR1" is used to specify conversions from the 1st channel to the 4th channel,

“ADC_SQR2” is used to determine the sequence of conversions from the 5th channel to the 9th channel. For example, configuring pin PC0 (ADC1_IN6) as the 1st sequence is achieved by setting the bitwise value "00110" corresponding to the relevant channel "6" in the “SQR1” section of the ADC module.

Finally, by enabling the relevant ADC module, the ADC oversampling feature is activated, and conversions with oversampling enabled are initiated. As described in section 3.1.5.4 of the ADC oversampling section, it acts as a high-frequency filter, akin to a low-pass filter (LPF), without causing any delay in the system. Therefore, no LPF is applied during ADC readings. There are no changes in conversion timings in oversampled mode; the sample time remains constant throughout the entire oversampling sequence.

4.3.6.3. Writing the moving RMS filter code

Complete moving RMS filter code is given in appendix. The implementation of the developed control algorithm on the Nucleo board is done gradually. The first stage involves defining the variables and array elements related to the RMS filter parameter, along with their variable type and dimensions. In the second stage, the defined variables and arrays are initialized. The body of the function is implemented within *“main.c”* file. However, the function is called separately for each phase in the *“interrupt.c”* file during each interrupt cycle.

In *“main.c”*, variable declarations and initializations are done. Variable declarations are made within a variable of type struct named *“MOV_StructType”* at the beginning of the *“main.c”* The initialization process is performed within a function defined inside the “int main()” function called *“moving_rms_init()”*.

The initialization function mentioned above is called only once for each inverter operation. The main function responsible for the actual moving RMS process is the *“moving_rms_update()”* function, the pseudo code of which we are sharing below. The return type of this function is float, so its proper implementation is directly related to whether the microcontroller has a floating-point unit.

```
float moving_rms_update (MOV_StructType *mRMS, int32_t in) {
        uint32_t in_sq = in*in;
        mRMS->ACC+= in_sq - mRMS->in_sq_L[mRMS->count];
        mRMS->avg_sq= mRMS->ACC*mRMS->invL;
        mRMS->in_sq_L[mRMS->count] = in_sq;
        if (mRMS->count == (mRMS->L-1)) mRMS->count = 0;
        else mRMS->count++;
        return mRMS->avg_sq;
  }
```

The moving RMS function operates as follows: The square of the instantaneous input from the ADC is taken directly. Then, the accumulator parameter is updated by adding the difference between the square of each new input and the square of the previous input corresponding to the current position in the circular buffer. This means that the difference between the square of the current input and the square of the previous input at the corresponding position in the buffer is accumulated into the accumulation value. This operation is used to update the accumulation process each time a new input data arrives.

The next step computes the updated average squared RMS value using a recursive equation. The accumulation value *("mRMS->ACC")* is multiplied by the inverse of the window length *("mRMS->invL").* This calculation results in the average squared RMS value. The accumulation value represents the sum of the squares of the input samples within the specified window length, and dividing this sum by the window length gives the average squared RMS value. This step is essential in the calculation of the moving RMS, providing the updated RMS value based on the accumulated input samples.

In this step, the square of the current input *("in_sq")* is stored in the circular buffer at the current position *("mRMS->count").* This means that the square of the current input value is recorded in the circular buffer at the index determined by the current position counter. This updating of the circular buffer allows the algorithm to keep track of the squared values of the input samples within the specified window length, facilitating the calculation of the moving RMS. Each time this

function is called with a new input sample, the square of that sample is stored in the circular buffer at the appropriate position, replacing the previous value stored at that position. This ensures that the circular buffer always contains the most recent squared values of the input samples within the window length.

If the circular buffer pointer reaches the end of the buffer, it means that we have filled the buffer and need to wrap around to the beginning of the buffer. In the case where *"mRMS->count"* equals to '0', The circular buffer pointer is reset to '0', indicating the start of the buffer. Otherwise, the circular buffer pointer is incremented by '1', moving to the next position in the buffer.

These steps control the movement of the circular buffer pointer, ensuring that it correctly loops back to the beginning of the buffer when it reaches the end. This allows the circular buffer to continuously store and update squared input values within the specified window length without overflowing.

Finally, the function returns the updated value of the average squared of phase output voltage *("mRMS->avg_sq")*. This value represents the current state of the moving RMS algorithm after processing the latest input sample. By returning this value, the function provides the updated output of the moving mean squared (MS) value of phase voltage, but not the root of it. which represents the current MS value based on the accumulated input samples within the specified window length.

In the *"interrupt.c"* file, the first function we call provides us with the MS value for the respective phase. In the subsequent step, we obtain the RMS value by taking the square root of this value as below;

```
mRMS_SqEstimate = moving_rms_update(&mRMS, ROUNDNUM(PH1_VLT));

mRMS_SqRtEstimate = ((sqrtf(mRMS_SqEstimate)));
```

As a result, since these operations are performed within the interrupt, we obtain an updated RMS value every 100 microseconds. Next, we will address coding of the PI controller algorithm.

4.3.6.4. Writing the PI controller code

One of the distinguishing features of this thesis work from other studies is not the way the PI controller is written, but rather the way the PI controller is implemented in the algorithm. While this section of the thesis is not entirely original, it has emerged through the modification of Phil's work [63] to meet the project's requirements.

The PI controller was optimized separately for both the classical control technique and the developed control technique. Therefore, variables were parameterized to enable a quick comparison between the two techniques. Below are the macros defining the PI parameters;

```
/* PID Parameters */

#define PID_KP  0.8f

#define PID_KI  0.4f

#define PID_KD  0

#define PID_LIM_MIN 0.5 f

#define PID_LIM_MAX 0.9f

#define PID_LIM_MIN_INT   0.5f

#define PID_LIM_MAX_INT   0.95f

#define SAMPLE_TIME_S 0.0001f
```

The struct type variable that contains these parameters in it, is shared below. The very first step of the PI control implementation is to initialize these parameters with a function named *"PIDController_Init(&pid)"* which takes the *"pid"* struct as an argument.

```
PIDController pid =   {  PID_KP, PID_KI, PID_KD,
                         PID_TAU, PID_LIM_MIN, PID_LIM_MAX,
                         PID_LIM_MIN_INT, PID_LIM_MAX_INT,
                         SAMPLE_TIME_S
                       };
```

Since Kp, Ki, and Kd are given through macros before the code is executed, there is no need to initialize them. However, the set value is also assigned as 16V within the main function.

The originality of this thesis stems not from the technique of writing the PID control algorithm, but from the frequency of applying the PI control output to the system. Since there is no significant modification on Phil's work on the PID code part, we will not investigate the PID implementation any further. As previously mentioned, the novelty we brought to the PI control technique is our implementation of the PI control correction signal not every 20 ms as in conventional techniques, but every 100 μs.

In the next section, we examine how all the techniques utilized in this study are brought together.

4.3.6.5. Combining everything and creating the algorithm

In this section, the implementation of the novel control technique developed in this study is examined. The classical RMS voltage control widely applied for three-phase voltage source inverter control relies on updating the RMS value and applying the corrective effect of the PI controller at the frequency of each fundamental period. This means that both the RMS value and the modulation index are updated every 20 milliseconds.

In the proposed technique, we update both the RMS value and the PI controller output, that is the modulation index, every 100 microseconds. This requires a different development approach compared to the classical method in terms of embedded software.

The body of main bare-metal embedded control algorithm is given in appendix. For a better understanding of the method, in this section, we will address each stage of the algorithm separately. Note that all the pseudo code we examine in this section is located under the *"interrupt.c"* file within a specific function of the timer 1 interrupt called *"TIM1_UP_TIM16_IRQHandler()"*. This function interrupts the main microprocessor flow every 100 microseconds, meaning every interrupt period, and executes the code lines within it. Below is the conversion of the developed technique into code using the bare-metal programming method.

To generate a sinusoidal PWM, first, we create an array of values for each phase sinusoid, each containing 12,000 elements *("string_unit_number")*. According to Equation (4.29), within each fundamental period (20ms), there are 200 samples of 100 microseconds each.

$$sample_number = \frac{T_{fund}}{T_{sw}} \tag{4.29}$$

Therefore, the variation of each element within the sinusoid *("CordicAngle")* corresponds to a change of 60 unit-angle according to Equation (4.30).

$$CordicAngle = \frac{string_unit_number}{sample_number} \tag{4.30}$$

The *"CordicAngle"* variable is initialized in the *"main.c()"* function as follows: '0' for phase 1, '4000' for phase 2, and '8000' for phase 3. This means there is a phase difference of '120' degrees between the reference sinusoids of each phase.

With each activation of the interrupt, the *"CordicAngle"* variable is incremented by 1 unit, corresponding to an angle increment of '60' degrees. Since a complete period corresponds to 12,000 array elements, arrays reaching this value are reset to zero. In the next step, modulation index is applied to each of the three phases. The system will be started in an open-loop manner, and for this, any value within a safe range (0.5-0.95) can be chosen as the initial modulation index. We initialize the modulation index at "0.9" in our experiment. In the following pseudo code, we generate sinusoidal reference for single phase.

```
CordicAngle+=60;

if(CordicAngle>11999) CordicAngle=CordicAngle-12000;

CordicSine=Sine[(uint16_t)CordicAngle]* 0.0005 *mod_ind;
```

The next step is the generation of PWM signals for the three phases. TIM1's "CCR1" register is used for phase 1, "CCR2" register for phase 2, and "CCR3" register for phase 3 in generating the PWM signals. As we recall, we set the dead-time to 305 ns during the timer configuration stage. In this code snippet, when *"CordicSine"* is '1', it means that the high-side switch is fully conducting, leaving only 305ns dead time for proper operation, while the complementary low-side switch operates in full cutoff mode with the same dead-time.

Pseudo code for this part is as below;

```
static uint16_t DCOFFSET=4100;
```

```
TIM1->CCR1= (uint16_t)(CordicSine*4100+DCOFFSET);          // PA8  &  PB13
TIM1->CCR2= (uint16_t)(CordicSine2*4100+DCOFFSET);         // PA9  &  PB14
TIM1->CCR3= (uint16_t)(CordicSine3*4100+DCOFFSET);         // PA10 &  PB15
```

The next pseudo code performs the conversion of ADC voltage measurements into real values.

```
PH1_VLT = (1.4814815 * (PC0_SEQ1 * 0.00080830281f) - 1.8f) * 25;
PH2_VLT = (1.4814815 * (PC1_SEQ2 * 0.00080830281f) - 1.8f) * 25;
PH3_VLT = (1.4814815 * (PC2_SEQ3 * 0.00080830281f) - 1.8f) * 25;
```

The real value of the instantaneous voltage measurement for any ADC channel is determined according to Equation (4.31).

$$V_{actual} = \left\{\left(\frac{12}{8.1}\right) * ADC_{val} * \left(\left(\frac{3.3}{4095}\right) - 1.8\right)\right\} * 25 \tag{4.31}$$

This equation is tailored to our hardware design and the reference voltage of the microcontroller. The constants '12' and '8.1' represent values obtained from the op-amps and voltage dividers used in the analog voltage measurement circuit, while the subtraction of '1.8' represents the output voltage of the reference integrated circuit in the measurement circuit, which is precisely 1.8V. The value 3.3 represents the reference voltage of the microcontroller (3.3V), and '4095' corresponds to the largest decimal value read by the ADC against 3.3V.

Since current measurements are obtained from sensors based on the Hall effect principle, they do not contain any mathematical contributions from the design. The reference voltage of the current sensor is 2.5V. Therefore, according to Equation (4.32), when no current flows (0A) from any channel, the expected value of 'I_{actual}' is measured as 2.5V.

$$I_{actual} = ADC_{val} * \left(\frac{3.3}{4095}\right) \tag{4.32}$$

The next section activates the moving RMS filter for each phase, obtaining instantaneous RMS values for the three phases. The first part takes the MS value and the second part simply calculates the RMS value using *"sqrtf()"* function. The pseudo code is as follows:

```
mRMS_SqEstimate = moving_rms_update(&mRMS, ROUNDNUM(PH1_VLT));
mRMS_SqRtEstimate = ((sqrtf(mRMS_SqEstimate)));
measurement = mRMS_SqRtEstimate;
```

The final stage of the embedded control algorithm is where the PI controller updates the modulation index based on the incoming RMS feedback. Here is how this part works: Firstly, the RMS values are stored in variables named “measurement” to be passed as input arguments to the *“PIDController_Update()”* function. Then, the system operation is initiated in an open-loop manner to be active only for the first 10 seconds. After the first 10 seconds, the *“PIDController_Update()”* functions, which update the modulation index according to the system's needs, are called every 100 microseconds.

```
if(Power_start) {
  A=4100;
  if (delay_counter < 100000.0f) {
    mod_ind = 0.9f;
    delay_counter++;
   }
  else mod_ind = PIDController_Update(&pid, setpoint, measurement);
 }
else {
  A=0;
  delay_counter=0;
  }
```

The entire process of the proposed technique has been programmed into the microcontroller with unique *‘C’* codes. With the developed technique, a control system has been achieved that provides much faster and more accurate dynamic response compared to the classical method, aided by samples obtained using ADC oversampling technique.

5. IMPLEMENTATION

In this section, we examine the implementation stages of a 125W highly efficient 3-phase voltage source inverter design, which operates under the novel control developed within the scope of this thesis. This section provides an overall perspective on the entire process, from creating component footprints up to the integration of the designed hardware with the developed embedded software.

5.1. Component Footprints & Stack-up

During our thesis work, Altium Designer [64] software was used for the schematic and PCB design of the necessary hardware. All the material footprints required were also created in the Altium environment. As an example, Figure 5.1 illustrates the creation of the schematic symbol and footprint of the SI8920, which we selected as an isolated single-ended to differential op-amp, in the Altium Designer environment, and their packaging together as an integrated library.

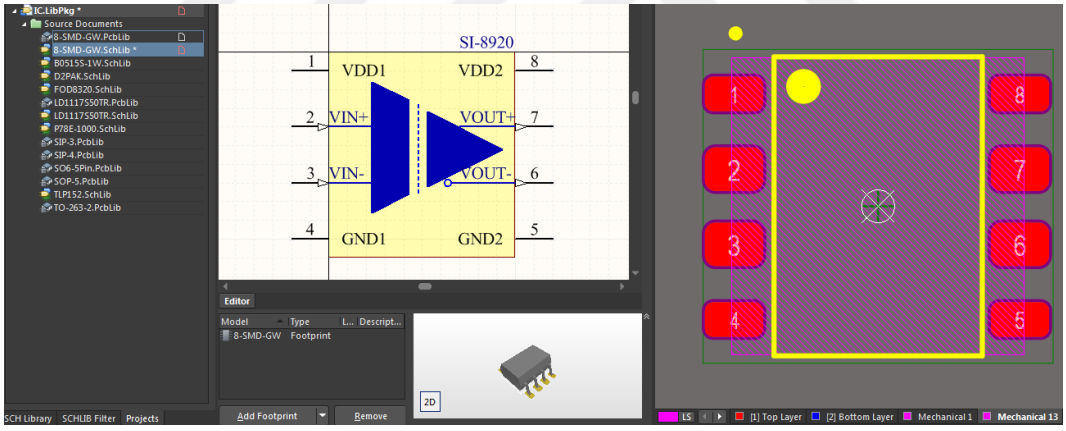


Figure 5.1. Schematic Symbol and Footprint Creation

Similarly, all the materials used in the project were created using Altium Designer's various component creation options, such as the "Footprint Wizard" and "IPC Compliant Footprint Wizard," and integrated into the project. The process of creating Altium libraries for the components was continued as the thesis matured.

For a good PCB design, the first steps include designing the schematic in a clear and understandable manner, defining the PCB rules properly, and creating the correct stack-up. We will address the schematic part in the next section. In Figure 5.2, a brief excerpt from the PCB rules specifically created for this thesis work is presented.

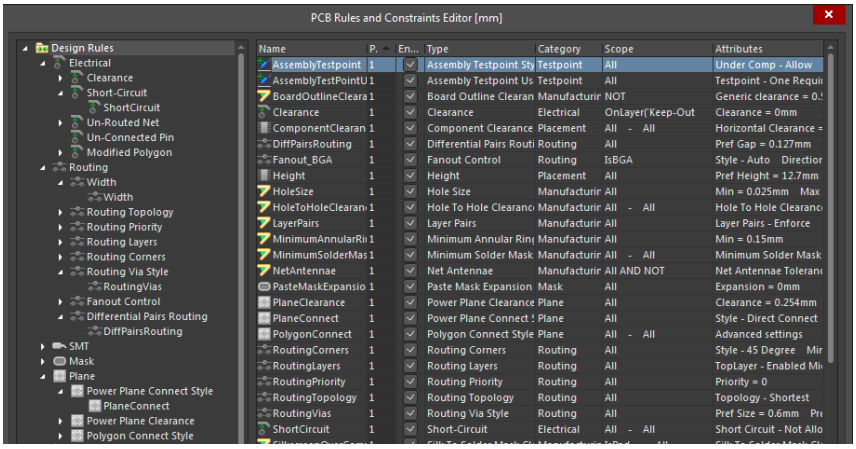


Figure 5.2. PCB Rule Creation

These rules are established based on manufacturability (capabilities of manufacturing companies), electrical characteristics derived from our design, considerations for signal integrity, past experiences, and globally accepted PCB design recommendations. Parameters such as trace widths, power plane and via current carrying capacities, copper and prepreg thicknesses, and impedance of differential pairs were determined using the Saturn PCB Toolkit [65] program.

Since the hardware is to be designed as a prototype, the size parameter has been disregarded, and all components have been placed on the top surface to facilitate debugging. This approach allows us to intervene on the inverter hardware without the need to flip the PCB's while operating under load. Additionally, to facilitate embedded software development, the NucleoG431 board is designed to be mounted in the middle section of the inverter PCB using SMD connectors. Four layers were deemed sufficient for the stack-up design. The stack-up design is shown in Figure 5.3.

#	Name	Material	Type	Weight	Thickness	Dk	Df
	Top Overlay		Overlay				
	Top Solder	SM-001	Solder Mask		0,025mm	4	0,03
	Top Copper Plating	PbSn	Copper plating		0,02mm		
1	Top Layer	CF-004	Signal	1oz	0,035mm		
	Dielectric 1	PP-017	Prepreg		0,203mm	4,3	0,02
	Dielectric 2	PP-017	Prepreg		0,203mm	4,3	0,02
2	Int1 (PWR)	CF-004	Signal	1/2oz	0,018mm		
	Dielectric 3	Core-039	Core		0,61mm	4,8	0,02
3	Int2 (Signal)	CF-004	Signal	1/2oz	0,018mm		
	Dielectric 4	PP-017	Prepreg		0,203mm	4,3	0,02
	Dielectric 5	PP-017	Prepreg		0,203mm	4,3	0,02
4	Bottom Layer	CF-004	Signal	1oz	0,035mm		
	Bottom Copper Plati...	PbSn	Copper plating		0,02mm		
	Bottom Solder	SM-001	Solder Mask		0,025mm	4	0,03
	Bottom Overlay		Overlay				

Figure 5.3. Stack-up of the Designed PCB

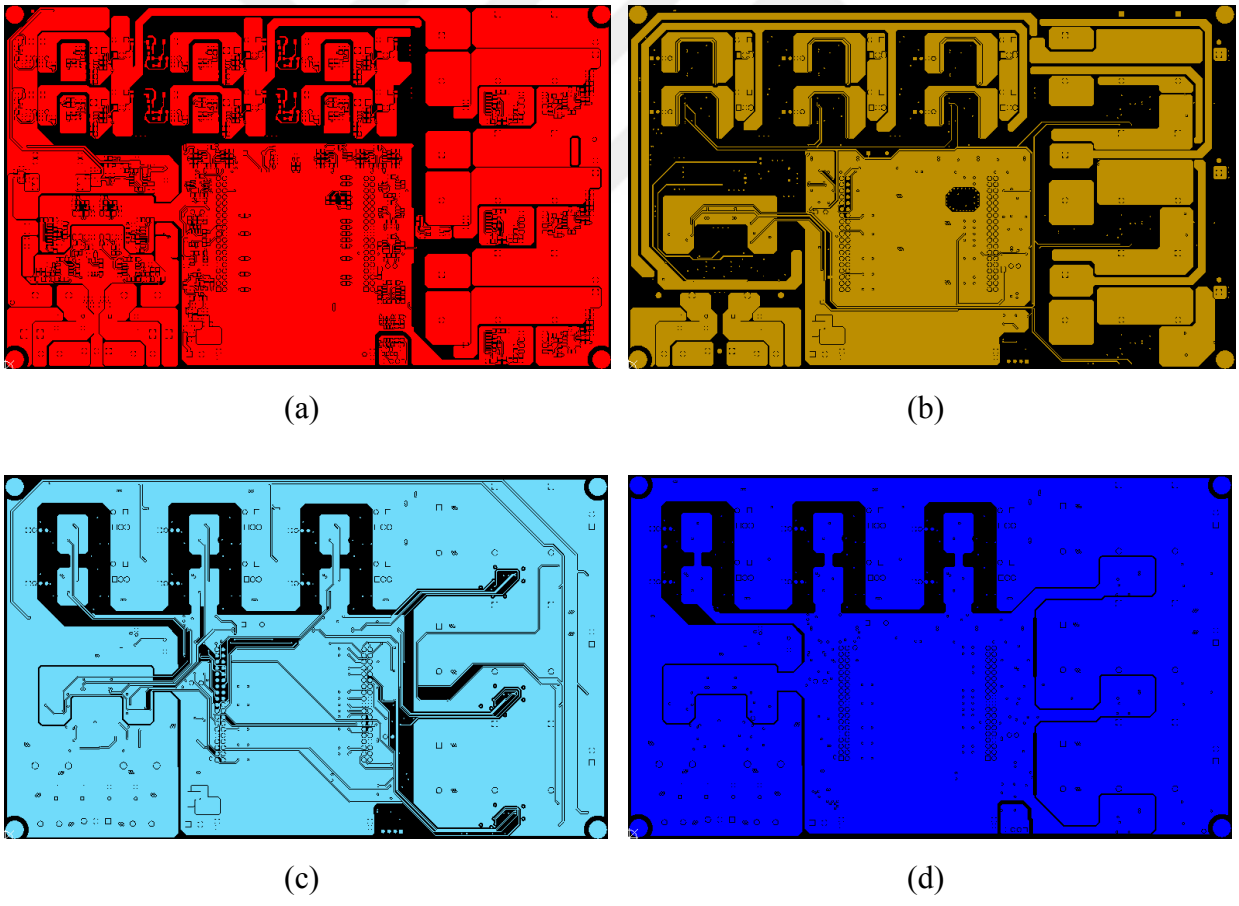
The 1st layer (top-layer) is the area where components are placed. Efforts have been made to minimize the routing of signals on this layer as much as possible to create noise immunity. This layer consists of components along with power and ground lines.

The 2nd layer is considered as the power layer. Care has been taken not to carry sensitive signals on this layer unless necessary.

The 3rd layer is the signal layer. Almost all signals, including switching signals, measurement signals, and various enable/disable functions, are routed on this layer.

The 4th layer is entirely dedicated to ground. The purpose is to provide isolation for sensitive signals from noisy power circuits within a Faraday cage between the 2nd and 4th layers.

In Figure 5.4, the Gerber outputs of each layer are shared.



(a) (b)

(c) (d)

Figure 5.4. (a) Top Layer (b) 2nd Layer (c) 3rd Layer (d) Bottom Layer

5.2. Schematic and PCB Layout

In this section, we investigate the schematic and PCB layout design of the hardware where the developed control is embedded in. The overall schematic view of the design is given in Figure 5.5.

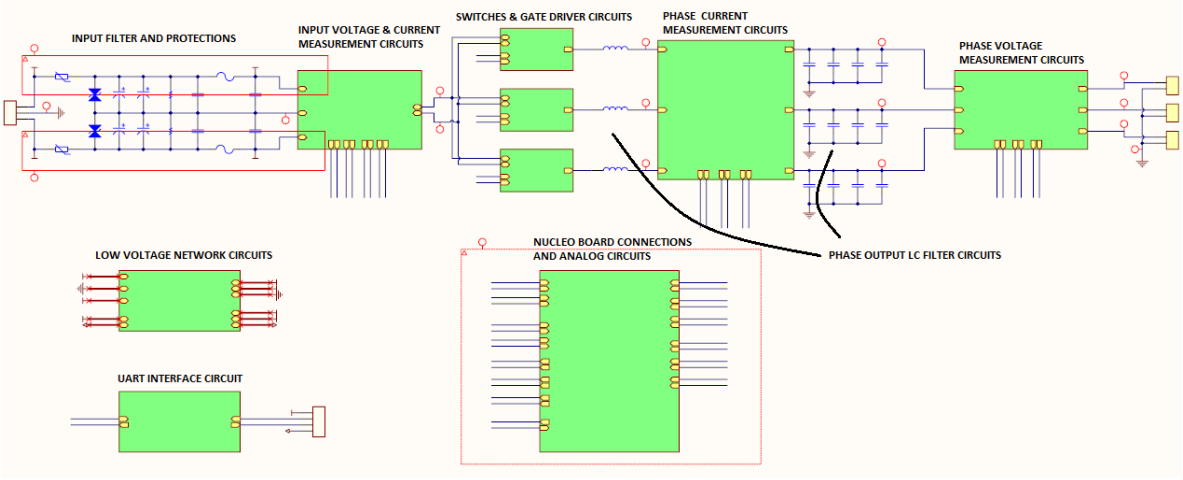


Figure 5.5. Three-Level Voltage Source Inverter Complete Schematic

The real designed and validated 125W inverter PCB is shown in Figure 5.6 below.

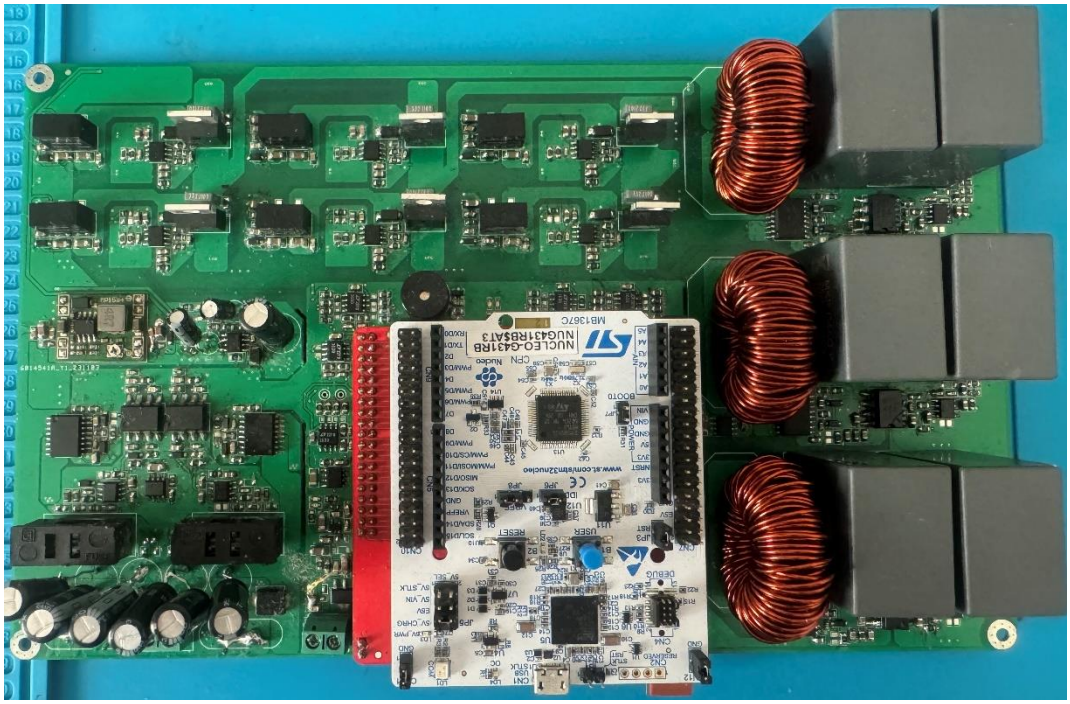


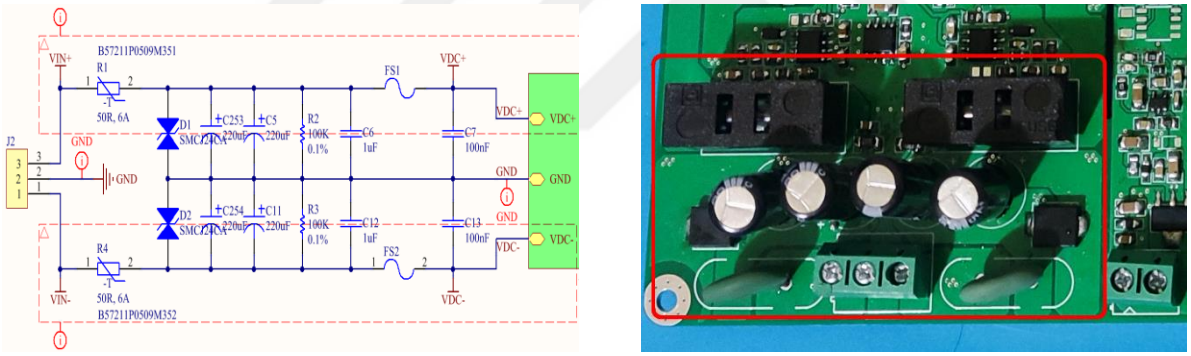
Figure 5.6. 125W Three-Phase Two-Level Inverter Assembled PCB Hardware

The main blocks of the three-phase voltage source inverter circuit include the input protection and filtering section, input-output voltage and current measurements, three-phase half-bridge switching circuits, output LC filter, low auxiliary voltage generation circuits, and finally the microcontroller connection blocks.

Since the hardware design for each circuit component is detailed in section 4.1, this section provides just a brief summary of the schematic and PCB design.

5.2.1. Input filter and protection circuits

The schematic design and PCB view of the input filter and protection circuit is shared in Figure 5.7. The input filter mainly consists of 6A current-rated NTC components, followed by the input busbar balancing capacitor block, and then voltage balancing resistors and fuses. Each of these components has been added separately for the positive and negative busbars.



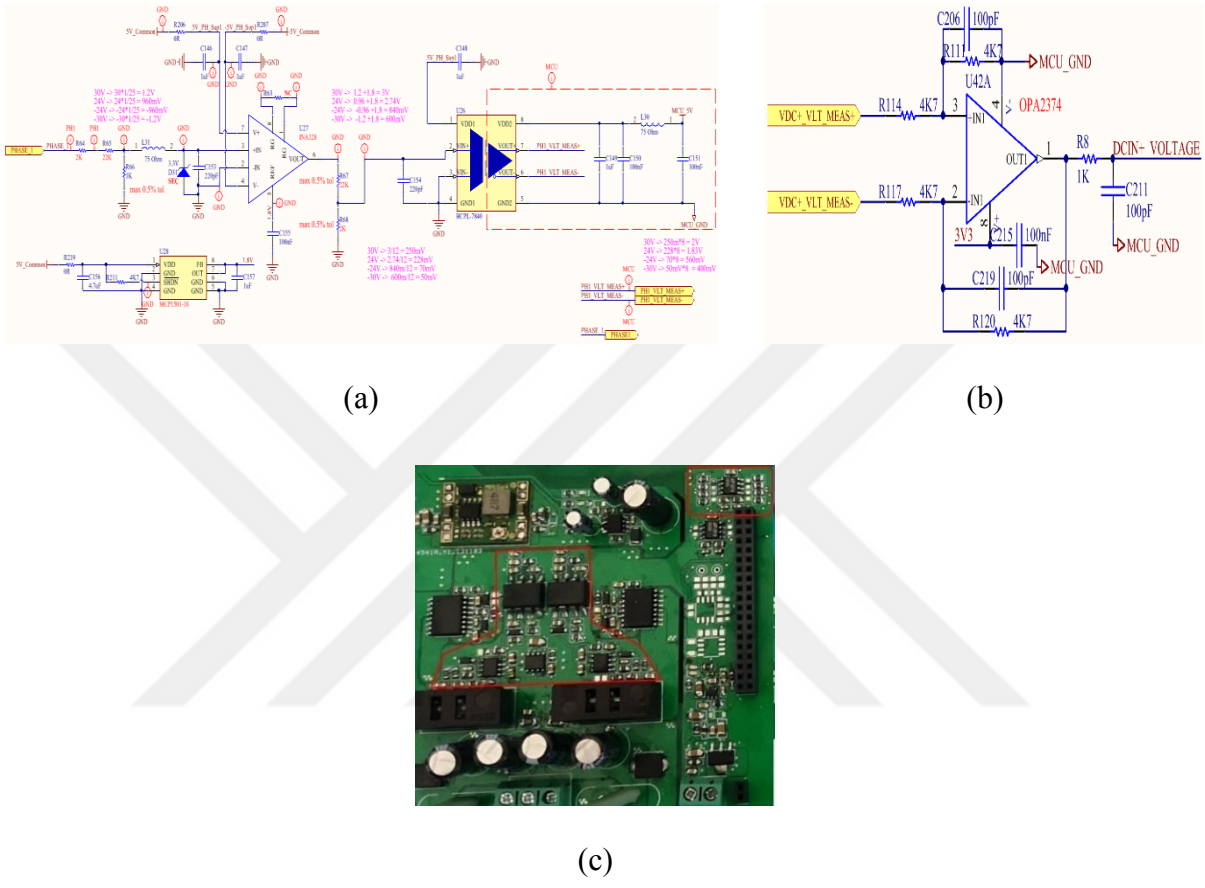
(a) (b)

Figure 5.7. (a) Schematic View of Input Filter Circuit (b) PCB View of Input Filter Circuit

5.2.2. Voltage measurement circuits

Voltage measurements for the positive and negative busbars and the voltages at the 3-phase outputs have the same measurement circuit structure. Therefore, sharing one measurement circuit is sufficient. Figure 5.8 shows the schematic design of the voltage measurement circuit and also its physical appearance on the PCB for input side of the design. As clearly seen from the PCB layout, measurements are taken as close to the source as possible, right next to the monitored source. The isolated signals are converted into differential signals and then routed to the microcontroller

connection points, where they are buffered and finally connected to the relevant pins of the microcontroller.



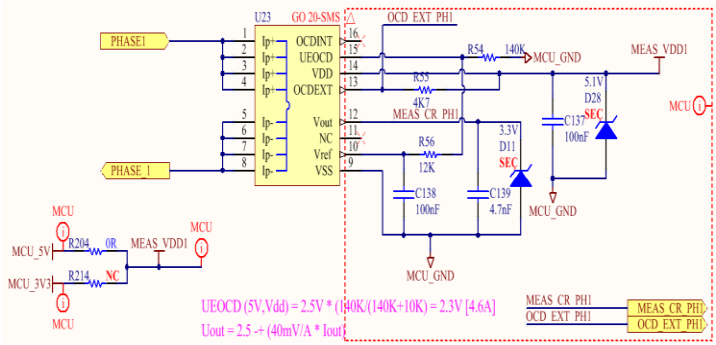
(a) (b)

(c)

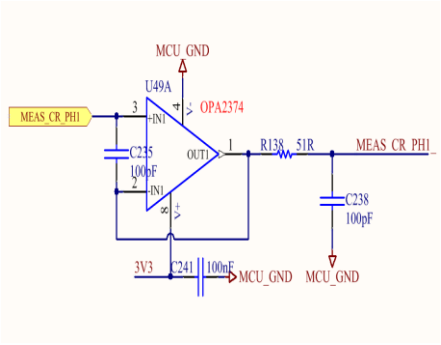
Figure 5.8. (a) Voltage Measurement Circuit Power Side (b) Voltage Measurement Circuit MCU Side (c) PCB View of the Voltage Measurement Circuit

5.2.3. Current measurement circuits

The method applied for all current measurements is common. The circuit structure is as shown in Figure 5.9. In this figure, the physical view of the input current measurement circuit on the PCB is also shared as an example. Since Hall effect sensors provide isolated measurement outputs, the measurement signal is directly routed to the vicinity of the Nucleo board without using any differential signal routing and then buffered before being transmitted to the relevant pin of the microcontroller.



(a)



(b)



(c)

Figure 5.9. (a) Current Measurement Circuit Power Side (b) Voltage Measurement Circuit MCU Side (c) PCB View of the Current Measurement Circuit

5.2.4. Gate drive circuits

Gate drive circuit for one switch is shared in Figure 5.10.

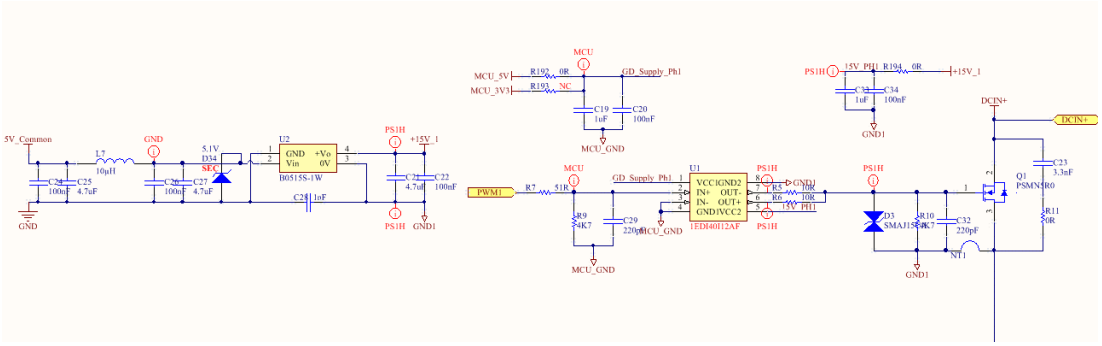


Figure 5.10. Schematic View of Gate Driver Circuit

The same driving technique is applied for each phase of the 3-phase inverter, comprising high-side and low-side switches. A separate isolated DC-DC converter produces an isolated 15V supply voltage for each gate driver. Since the circuit structures are identical, it is sufficient to examine the driving circuitry of a single MOSFET and its driving circuit.

Figure 5.11 shows the power switch, driving circuit, and isolated DC-DC converter for one phase, together on the PCB.

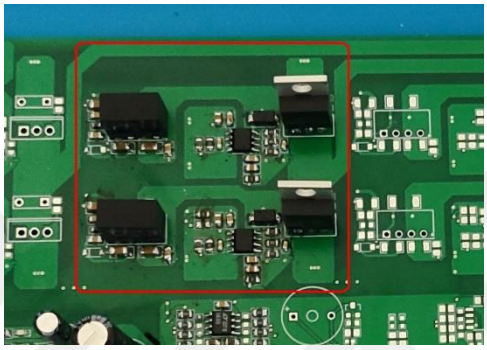
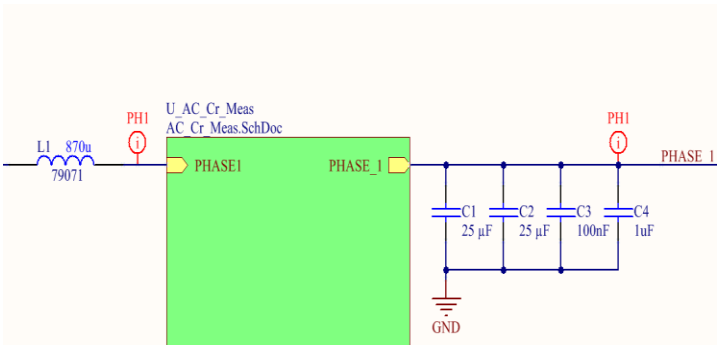
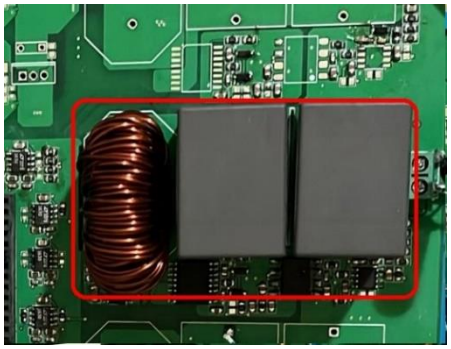


Figure 5.11. PCB View of Gate Driver Circuit

5.2.5. Output filter circuits

As explained in Section 4.1.2, the output filter is implemented with one 835uH 3.9Apeak custom-designed inductor and two 25uF MKP capacitors for each phase. The schematic design and PCB layout for the output filter of one phase are provided in Figure 5.12.





(a) (b)

Figure 5.12. (a) Schematic View of Output Filter Circuit (b) PCB View of Output Filter Circuit

5.2.6. NucleoG431 Peripheral and Connection Circuits

The power electronics circuit is connected to the controller board via SMD board-to-board connectors that mate with the connectors on the NucleoG431 board. There are buffer op-amp's for the voltage and current measurements, buffer op-amp's for DAC outputs used for debugging purposes, and other components such as LEDs and buzzers in this region. All of these circuits share the same reference ground as the microcontroller, and care has been taken to ensure that there is no interference between the power ground and this isolated ground even in the internal layers. The circuit diagram showing the connections to Nucleo board is given in Figure 5.13.

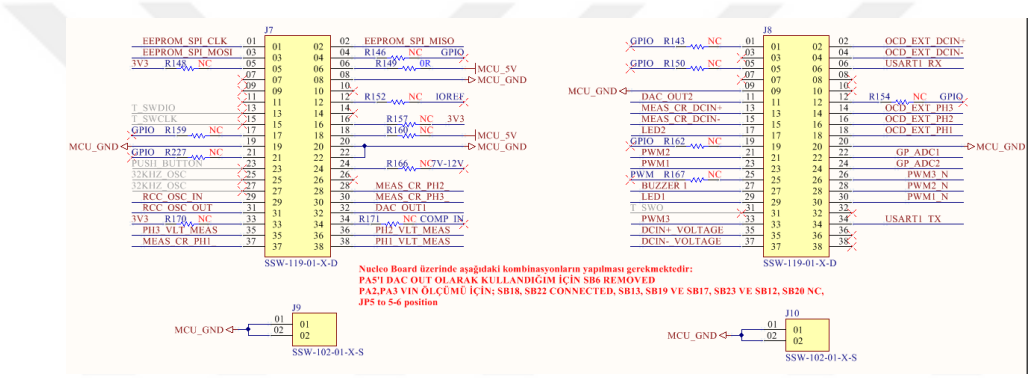
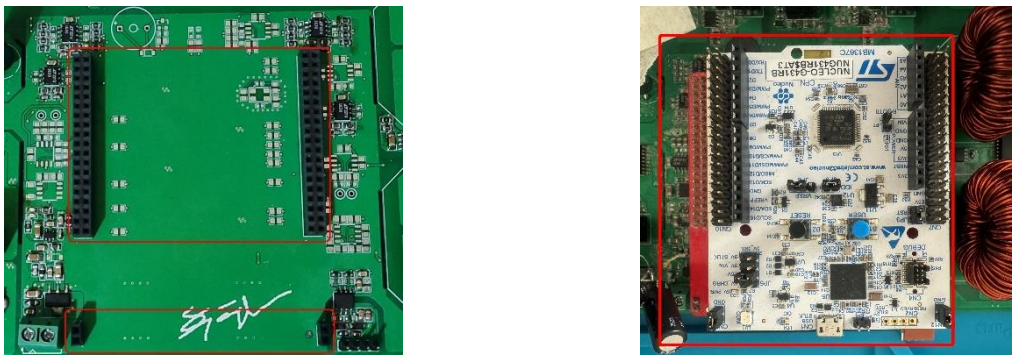


Figure 5.13. Schematic of Controller Board Connection Circuits

Figure 5.13 (a) shows the PCB view of the power board when Nucleo board is dissembled, and Fig 5.13 (b) shows the same when the NucleoG431 board is integrated onto the hardware board.



(a) (b)

Figure 5.14. (a) Main Board when Control Board Dissembled (b) Main Board when Control Board Assembled

5.2.7. Auxiliary power supply circuits

5.2.7.1. +25V to +5V converter circuit

Generation of 5V from the positive busbar voltage was made possible by using the MP1584 step-down converter module, which is available as a ready-made PCB module in the market. Additional filtering and protection elements were added for the safe integration of this product into the circuit. The circuit diagram for producing 5V with the MP1584 was shared in previous sections. The schematic showing its integration into our designed hardware is provided in Figure 5.15.

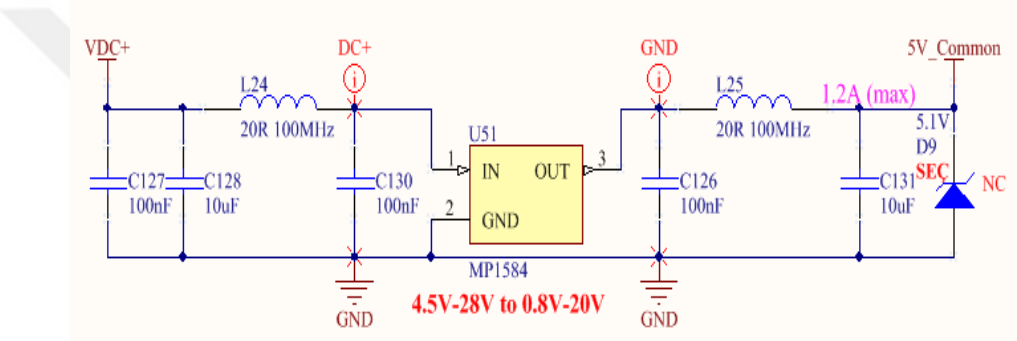
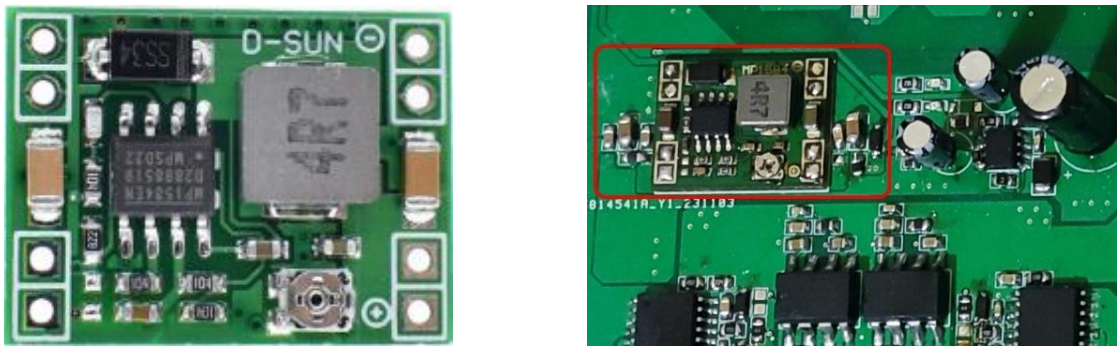


Figure 5.15. Schematic View of +25V to +5V Converter

The physical appearance of the MP1584 module and its assembled version on our PCB board are given in Figure 5.16, respectively.

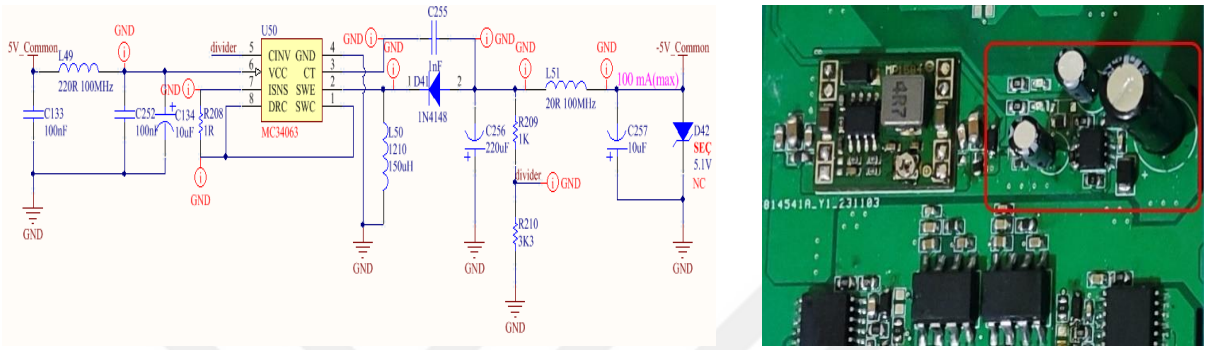


(a) (b)

Figure 5.16. (a) Physical View of MP1584 Module (b) PCB View of +25V to +5V Converter

5.2.7.2. +5V to -5V converter circuit

The generation of the negative 5V voltage was achieved using the NCP3063, a newer version of the MC34063 integrated circuit. The circuit diagram and PCB view are shared in Figure 5.17.

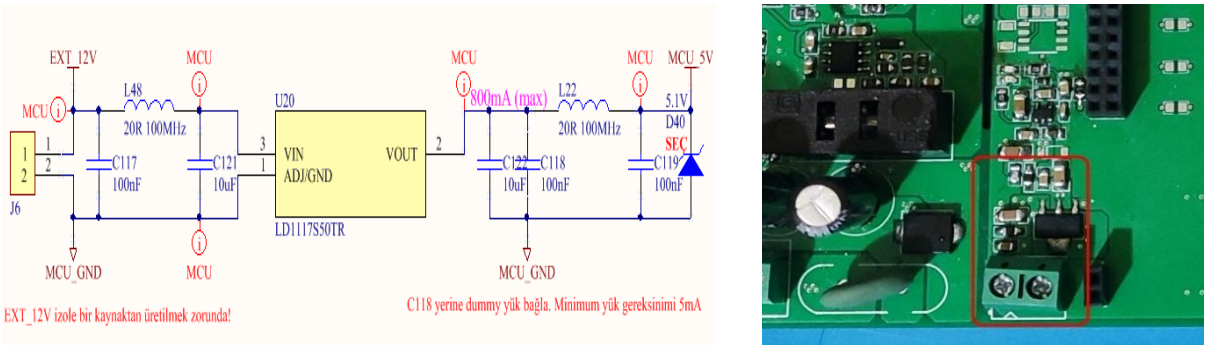


(a) (b)

Figure 5.17. (a) Schematic View of +5V to -5V Converter (b) PCB View of +5V to -5V Converter

5.2.7.3. +12V to +5V converter circuit

The circuit diagram of the LDO regulator used to generate the 5V needed by the microcontroller is shown in Figure 5.18 (a). This voltage is generated from external sources sharing the same reference ground with the MCU. Its physical appearance on the PCB is shared in Figure 5.18 (b).

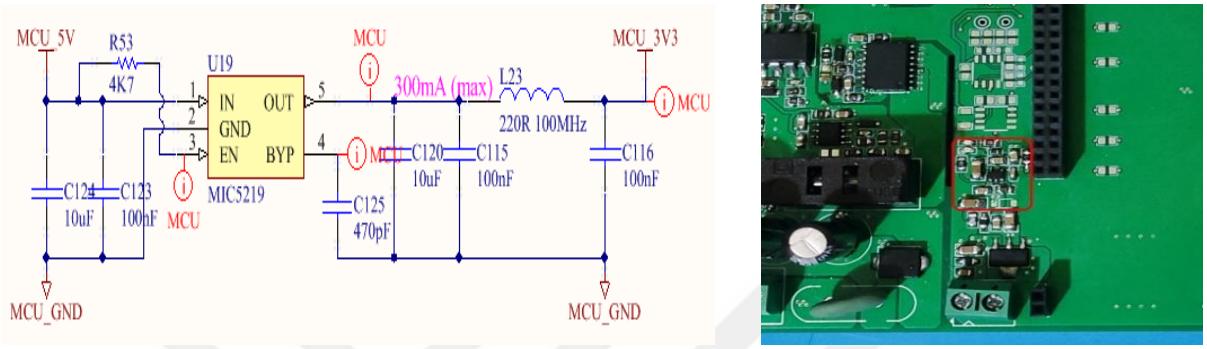


(a) (b)

Figure 5.18. (a) Schematic View of +12V to +5V Converter (b) PCB View of +12V to +5V Converter

5.2.7.4. +5V to +3.3V converter circuit

The circuit diagram of the LDO-based regulator used for converting isolated 5V to 3.3V is shared in Figure 5.19 (a). The physical appearance of the 3.3V regulator is shown in Figure 5.19 (b).

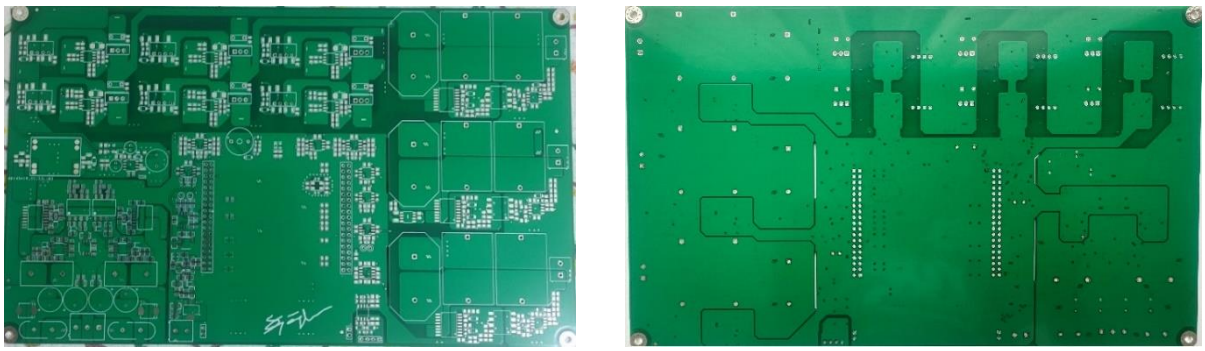


(a) (b)

Figure 5.19. (a) Schematic View of +5V to +3.3V Converter (b) PCB View of +5V to +3.3V Converter

5.3. PCB Manufacturing and Assembly

PCB production process includes creating quality schematics, establishing PCB rules, designing good stack-up, placing components, and routing. Designing a manufacturable PCB is as crucial as meeting the electrical expectations. The PCB production was carried out with the JLC PCB [66], and five PCBs were obtained. Top and bottom views of procured PCBs are shown in Figure 5.20.



(a) (b)

Figure 5.20. (a) Front View of the Manufactured PCB (b) Rear View of the Manufactured PCB

PCBs were visually inspected and underwent continuity tests using a multimeter. Afterwards, assembly, that is soldering of the components on the PCB, was conducted step by step, and each function was verified internally.

In the next section, we discuss the testing environment and how the embedded control software was implemented on the designed inverter hardware.

5.4. Embedded Control Implementation on Designed Hardware

After the production and procurement of PCBs, a comprehensive test environment was established for the development of embedded control software and for complete functional verification of the 3-phase voltage source inverter circuit. A visual representation of this test environment is shared in Figure 5.21.

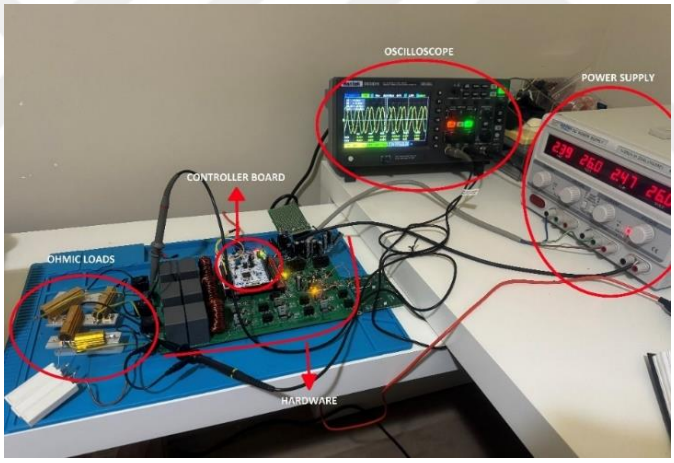


Figure 5.21. Test Environment of Three-Phase Voltage Source Inverter

Before implementing the control technique for 3-phase operation, it was ensured that all current-voltage measurements for single-phase operation were functioning correctly, that the switching circuit was applying the desired PWM signals, that the low voltage network was operating as expected, and essentially that all modules were functioning properly locally.

The embedded software was developed in the STMCubeIDE environment, and the firmware was uploaded from the computer to the control board via the mini-USB connector provided by the NucleoBoard's ST-Link connection.

6. EXPERIMENTAL RESULTS

In this section, we integrated the novel control technique with the hardware of the 125W 3-phase 2-level inverter that we designed. The waveforms shared in this section are real oscilloscope screenshots related to the implementation of the designed inverter.

The loads driven by inverter circuits can be divided into two main categories: linear loads and non-linear loads. Linear loads typically have electrical characteristics such as a fixed resistance or inductance, and they draw AC voltage generated by the inverter directly and steadily. They generally consume constant power and do not create harmonic distortion or fluctuations. Non-linear loads, on the other hand, are loads that operate using electronic switching devices or power electronics. These loads typically have varying resistance or capacitance and can deform the waveform generated by the inverter, potentially creating harmonic distortion. For example, a variable-speed motor exhibits non-linear load behavior, whereas a constant-speed motor is considered a linear load.

The success of the technique we developed relies on improving dynamic response across all linear loads, but we make no such claim for non-linear loads. Furthermore, since the developed technique does not promise to enhance circuit THD performance, meaning producing high-quality and low harmonic content, it was deemed sufficient to test the circuit with readily available resistive linear loads in the market. The type of linear load (whether inductance or a constant-speed motor load) has not influenced the testing method.

Similarly, we believe that changing the values of the components of the output filter, such as inductance and capacitance, will have a minor effect on the dynamic response as long as it does not alter the cutoff frequency of the filter.

6.1. Sampling Accuracy

Sampling accuracy is the cornerstone of the control algorithm developed in this thesis. Merely designing a well-functioning measurement circuit hardware is not sufficient to maintain voltage regulation optimally under all conditions; it is also crucial how the microcontroller's ADC module takes measurements and processes measurement results. Moreover, successful control based on an RMS calculated from incorrectly sampled voltage measurements cannot be expected.

In Section 4.1.6, we demonstrated in the simulation environment that the error in voltage measurements due to inverter hardware would not exceed 3%. To examine measurement accuracy, the busbar capacities have intentionally been undersized, allowing the busbar to swing. The input busbar voltage given as +-26V increases Vdc+ by 660mV, while decreasing Vdc- by the same amount. Even such fluctuating voltages with high ripple are accurately read by the ADCs and seen in STM32CubeIDE's debug screen as in Figure 6.1

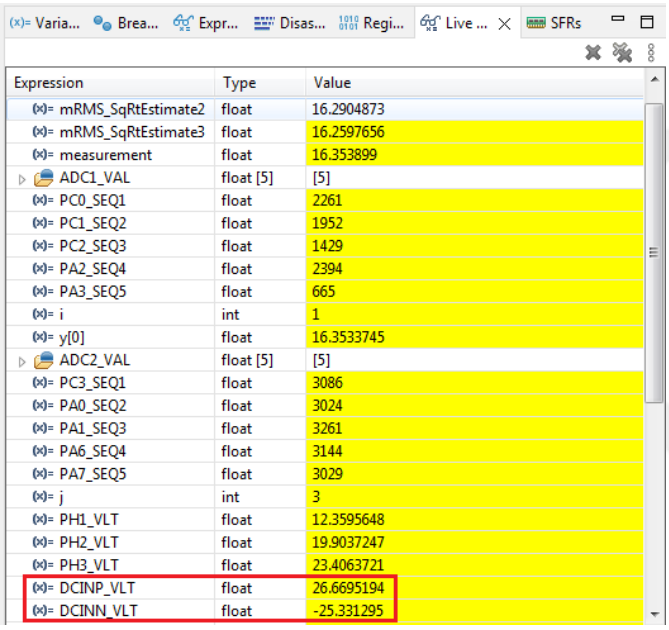


Figure 6.1. Sampling Accuracy from STM32CubeIDE Interface

To obtain a noise-free precise measurement signal, a commonly used method is to apply a low pass filter to the measurement signals, at the expanse of some delay in measurement to obtain clean readings. However, in applications where switching frequencies are high, it may not be feasible to effectively use LPFs. The delay introduced by LPFs may exceed the interrupt duration required for completing the operations. This is where the benefits of ADC oversampling technique come into play. Oversampling not only increases signal accuracy but also efficiently allocates time, allowing the system to perform other necessary arithmetic functions and various algorithms.

It is not possible to accurately test the error rate of instantaneous measurements against the sinusoidal output voltage. This aspect was covered during functional testing by applying DC voltages separately for each ADC module. However, the compatibility of the scaled RMS value shown by the DAC output with the real value, can be considered as another parameter indicating

measurement accuracy. In Figure 6.2, the output sinusoidal voltage (V_{REAL}) for phase 1 obtained during implementation and the DAC equivalent (V_{DAC}) of this voltage's RMS value are shown simultaneously.

A scaling factor of 16/3 was applied to convert the values seen on the oscilloscope in mV to their real voltage equivalents according to Equation (6.1).

$$V_{REAL} = V_{DAC} * \left(\frac{16}{3}\right) \tag{6.1}$$

In Figure 6.2, the measured DAC value of 3.02V corresponds to a real value of 16.106V according to equation 3. However, we observe this value as 16.02V on the oscilloscope. This indicates that the steady-state RMS voltage is calculated as 86mV with an error rate of less than 1%. This visualization is important as it demonstrates how accurately the actual generated phase voltage is reflected during the measurement process.

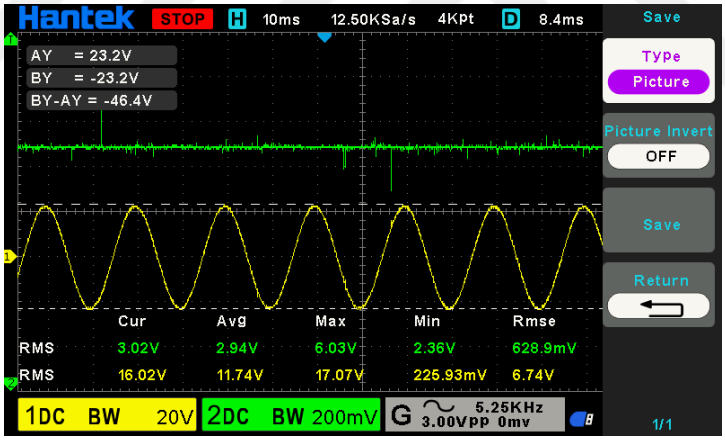


Figure 6.2. Output Voltage of Single Phase & Scaled RMS of Output Voltage

6.2. Static Response

In this section, the steady-state regulation, RMS value stability, and efficiency of the designed 3-phase voltage source inverter circuit were examined. It is observed that the recommended control method meets the electrical parameters promised in Table 4.1. The efficiency of the classical method is also touched in this section.

In Figure 6.3, the waveform shown in green corresponds to the output voltage of phase 1, and the yellow waveform corresponds to the output voltage of phase 2. Despite phase 1 being under heavy load and phase 2 under light load, it can be seen that the output sinusoids are identical.

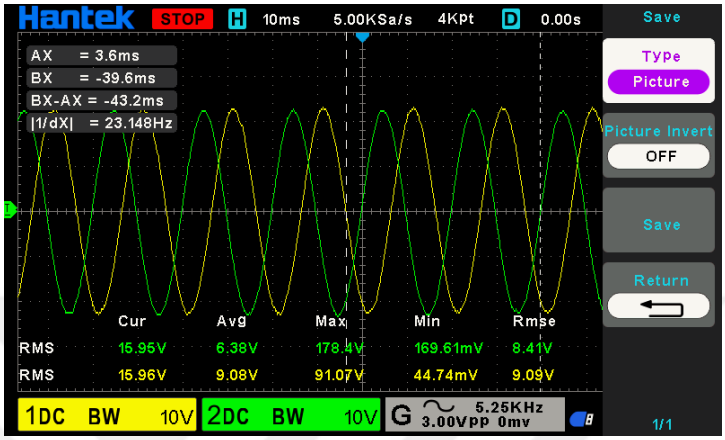


Figure 6.3. Voltage Outputs of Phase 1 and Phase 2

Another important parameter for the static response of the design is efficiency. It is as crucial for the system efficiency not to fall below the target value under both balanced and unbalanced load conditions as it is for the regulation not to be compromised. Balanced and unbalanced load conditions were created to measure system efficiency.

System efficiency is not a metric used to measure dynamic response. The primary advantage of this novel control technique lies in the dynamic response of the inverter, not in its static response. Therefore, since both control techniques are optimized within their own ecosystems, it is expected that their efficiencies would be similar.

In the balanced load scenario, approximately 2.35A of load is drawn from all phases, while in the unbalanced load scenario, 2.35A is drawn from two phases, and a lighter load of 485mA, referred to as the "light load," is drawn from the other phase.

Under these conditions, the system efficiencies, calculated based on the current and voltage values measured at the output and those obtained from the power supply, are shown in the table 6.1.

Table 6.1. Efficiency Table of the 3-Phase 2-Level Inverter

	Proposed Control Technique		**Classical Control Technique**	
	Balanced Load	**Unbalanced Load**	**Balanced Load**	**Unbalanced Load**
Vphase1 (V)	16.14	16.19	16.15	16.42
Vphase2 (V)	16.15	16.14	16.10	15.98
Vphase3 (V)	16.19	16.17	16.14	16.00
Rphase1 (Ω)	6.8	33	6.8	33
Rphase2 (Ω)	6.8	6.8	6.8	6.8
Rphase3 (Ω)	6.8	6.8	6.8	6.8
Pphase1 (W)	38.31	7.94	38.36	8.17
Pphase2 (W)	38.36	38.31	38.12	37.55
Pphase3 (W)	38.55	38.45	38.31	37.65
$Pout_{Total}$ (W)	115.21	84.70	114.78	83.37
Vsupply (V)	52	52	52	52
Idc+ (A)	2.45	1.83	2.44	1.84
Idc- (A)	2.37	1.75	2.36	1.76
Psupply (W)	125.32	93.08	124.8	93.6
Efficiency (%)	91.93	91	91.97	89.07

However, it can be observed that in terms of the steady-state efficiency achieved under unbalanced load, the developed control technique is slightly more successful. This table also indicates the importance of achieving the target efficiency of >90% under all load conditions, as aimed at the beginning of the design process.

6.3. Dynamic Response

The primary contribution of the developed control technique lies in improving the dynamic response. Therefore, in this section, we will discuss how the developed control technique, applied to real hardware, enhances the dynamic response of the inverter. The dynamic response of the three-phase voltage source inverter circuit designed using SPWM technique can be examined by observing two factors. The first is the error voltage amplitude during sudden load changes, and the

second is the time taken for the system to return to the 16V RMS voltage level after a sudden load change, known as the recovery time.

In this section, as in the simulation studies, tests were conducted in two ways: transition from light load to heavy load and transition from heavy load to light load. The implementation results based on real oscilloscope outputs are consistent with the findings obtained during simulation studies. Additionally, the impact of ADC oversampling on control, which was not possible to reflect during simulation, could also be examined during real inverter operation.

In the application developed within the scope of this thesis, we generated 16 V AC from a 52 V (±26V) DC link. To evaluate the effectiveness of the technique we developed for optimizing the dynamic response and output regulation of the designed hardware, we tested it in two groups. We conducted tests by applying heavy loads of 6.8 Ω 50 W and light loads of 33 Ω 50 W ohmic loads. For this test setup, according to Equation (6.2), the RMS value of the light load is 485mA, and that of the heavy load is 2.35A.

$$I_{phase_{RMS}} = \frac{V_{phase_{RMS}}}{R_{load}} \tag{6.2}$$

The implementation results could be observed by sending the measured RMS voltage from the DAC output to the oscilloscope. According to Equation (6.3), the real-time equivalent of the measured voltage from the DAC output corresponding to 16V RMS voltage is 3V. ADCs and DACs inherently clamp voltages above 3.3V to 3.3V due to the nature of the microcontroller. Therefore, the target RMS value's corresponding DAC output voltage is set to a safe value of 3V. The next scaling factor is required to convert the analog value into a digital degree that the DAC can accurately interpret.

$$V_{DAC} = V_{phase_{RMS}} * \left(\frac{3}{16}\right) * \left(\frac{4095}{3.3}\right) \tag{6.3}$$

In the first test, we observed the maximum error rate generated during the transition from light load to heavy load and vice versa for both techniques while keeping the ADC oversampling rate constant at 32x. Figure 6.4 shows the oscilloscope outputs for this test, both for classical approach and the developed method.

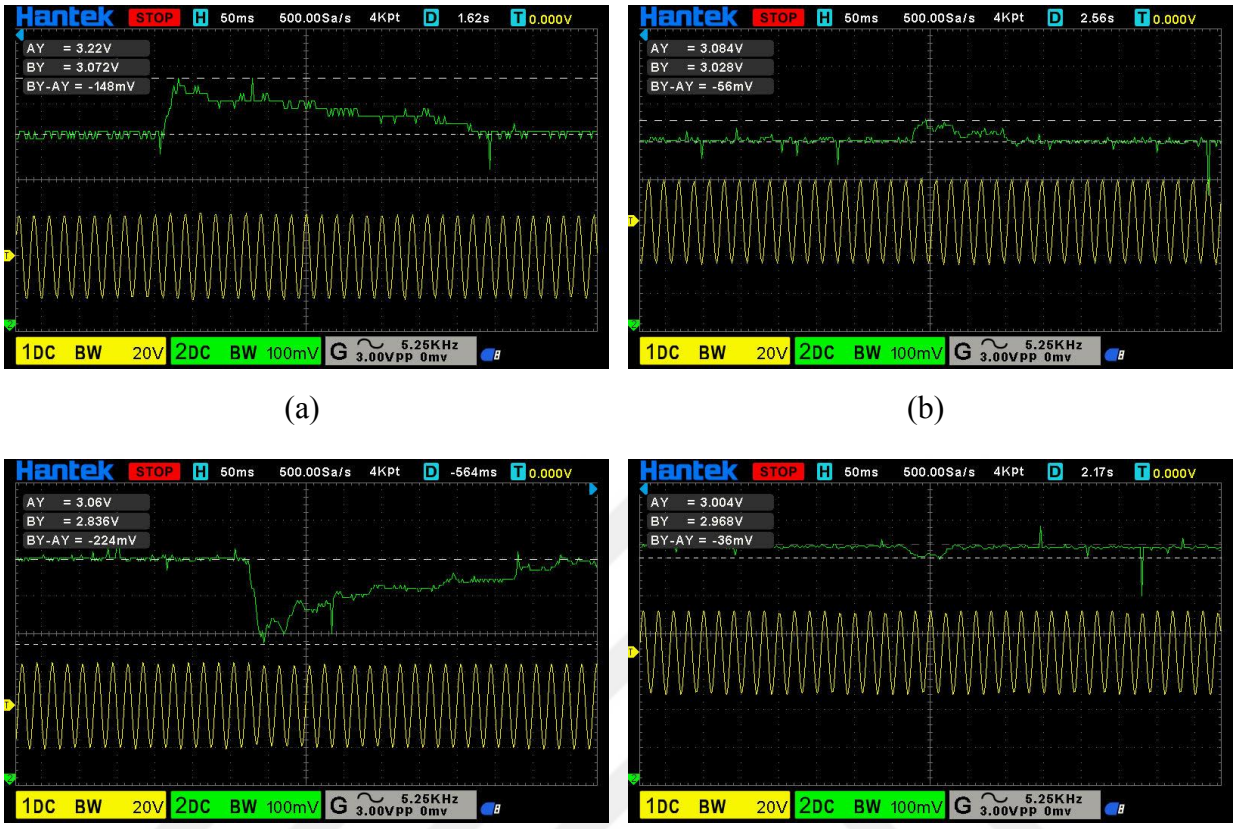


Figure 6.4 (a) Classical Method Heavy to Light Load Transition Max Error (b) Proposed Method Heavy to Light Load Transition Max Error (c) Classical Method Light to Heavy Load Transition Max Error (d) Proposed Method Light to Heavy Load Transition Max Error

During the transition from heavy load to light load, the maximum error rate in terms of DAC voltage decreased from 148 mV in the conventional method to 56 mV in the proposed method. The real voltage equivalents of these values are 789 mV for the conventional method and 298 mV for the proposed method. Thus, while the RMS value drops to 15.21 V in the conventional technique, with the developed technique, this decrease is limited to 15.7 V, indicating that the maximum undershoot amount during the step load transition is around 1.86% of the RMS voltage.

During light load to heavy load transition test, while a drop of 224 mV is observed with the conventional method, the drop remains limited to 36 mV with the developed technique. These values correspond to 1.19 V for the conventional method and only 192 mV for the developed technique in real terms. So, while in the traditional method the voltage drops even below 15 V, in

our developed innovative technique, the RMS value decreases only to 15.81 V level, presenting the disturbance in RMS regulation remains at a satisfactory rate of 1.2%.

Next, we compared the two methods in terms of the recovery time to reach RMS voltage of 16 V during sudden load changes. Figure 6.5 shows the scope outputs for this test for each method.

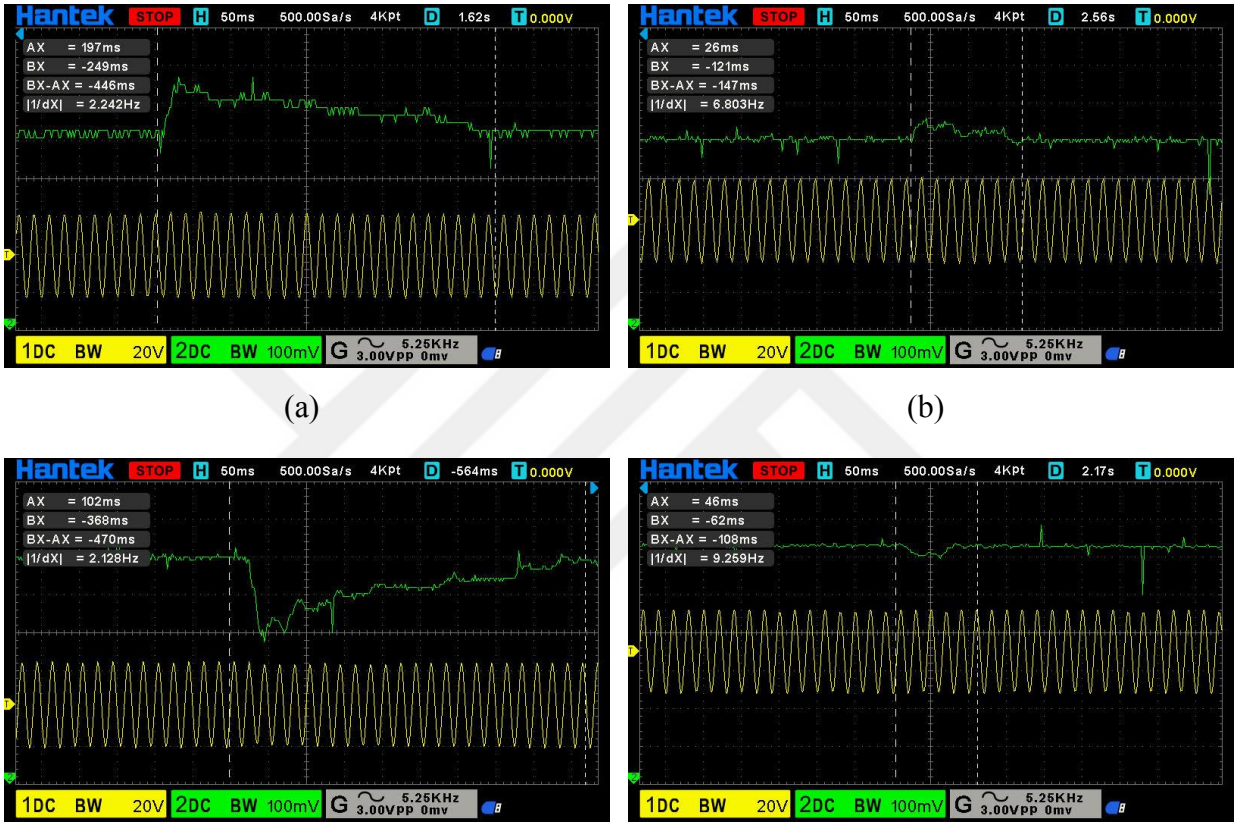


Figure 6.5 (a) Classical Method Heavy to Light Load Transition Recovery Time (b) Proposed Method Heavy to Light Load Transition Recovery Time (c) Classical Method Light to Heavy Load Transition Recovery Time (d) Proposed Method Light to Heavy Load Transition Recovery Time

During heavy load to light load transition, in the conventional technique, the recovery time was measured as 446 ms, while for the proposed method, this time was measured as 147 ms. These values were measured as 470 ms and 108 ms for the conventional and the proposed techniques, respectively.

The extended duration observed in the conventional technique, when considered in conjunction with the level at which the RMS voltage drops, may result in drawing significantly more current from the output than under steady-state conditions. During constant power operation, this could potentially lead the system to enter an overcurrent fault state or, in systems without current protection, even result in a fatal crash. In contrast, the developed technique optimizes the overall dynamic response to the maximum extent by providing both minimal voltage drop and minimum recovery time.

The results shared above represent the optimal outcomes achievable with both the conventional and the novel technique developed with our designed hardware. This is because the ADC oversampling rate was set to the highest possible value of 32x when considering the electrical characteristics of both the microcontroller and our system.

To observe the contribution of ADC oversampling to this technique, in the next section, we reduced the oversampling rate from 32x ('2^5) to 8x (2^3), and then to 2x (2^1) while basing the proposed technique. With 32x ADC oversampling, the obtained real values were already shared as 298 mV (real equivalent) and 147 ms for heavy load to light load transition (Figure 6.4 (b) and 6.5(b)). And these values were measured as 192 mV (real equivalent) and 108 ms for light load to heavy load transition (Figure 6.4 (d) and 6.5(d)).

For 8x ADC oversampling rate the observed decrease in dynamic performance shown in Figure 6.6, was notable in terms of both RMS error rate and recovery time.

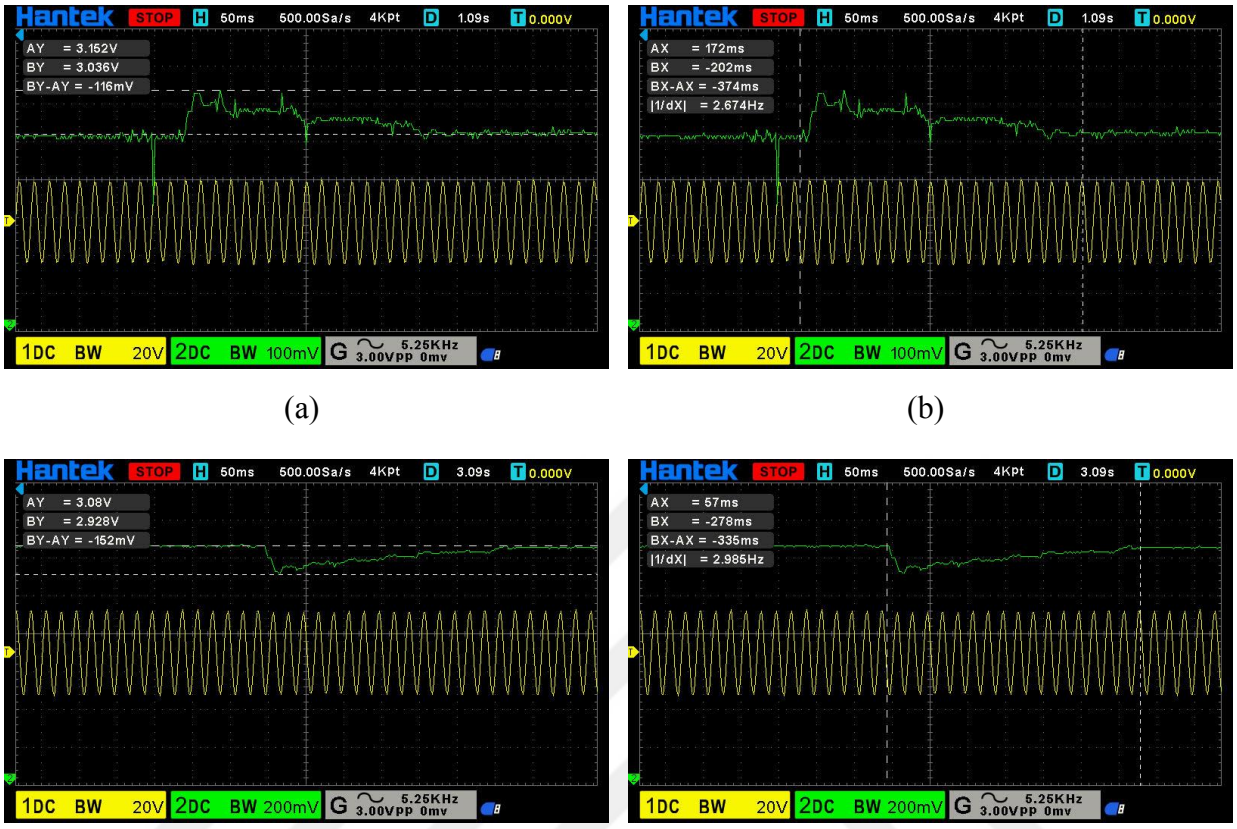


Figure 6.6 (a) Max Error with 8x Oversampling During Heavy to Light Load Transition (b) Recovery Time with 8x Oversampling During Heavy to Light Load Transition (c) Max Error with 8x Oversampling During Light to Heavy Load Transition (d) Recovery Time with 8x Oversampling During Light to Heavy Load Transition

When 8x ADC oversampling was applied, for heavy load to light load transition, the maximum voltage error measured from the DAC output was 116 mV which corresponds to 619 mV in real value terms. And we recorded the recovery time of the RMS voltage as 374 ms. Thus, transitioning from a 32x ADC oversampling rate to an 8x oversampling rate resulted in a much more undershoot much longer recovery time for the system. Significant degradation occurred in regulation, and the recovery time increased.

The performance degradation for light load to heavy load transition is as follows; the maximum voltage error measured as 152 mV, which corresponds to 810 mV as real term. And recovery time increased to 335 ms. So, the error amount increased roughly fourfold, and the recovery time also doubled.

When we reduced the ADC oversampling rate to 2x (2^1), while keeping all other parameters of the system constant and the control technique as our method, the performance drop we observed was staggering. With oversampling reduced to such an extent, it became impossible for our designed inverter to control the output voltage regulation.

Consequently, capturing the maximum error and recovery time became impossible as well. The most remarkable visual representation of this situation was seen in Figure 6.7, which shows how fast the regulation is lost when transitioning from open-loop operation to closed-loop operation under this oversampling configuration.

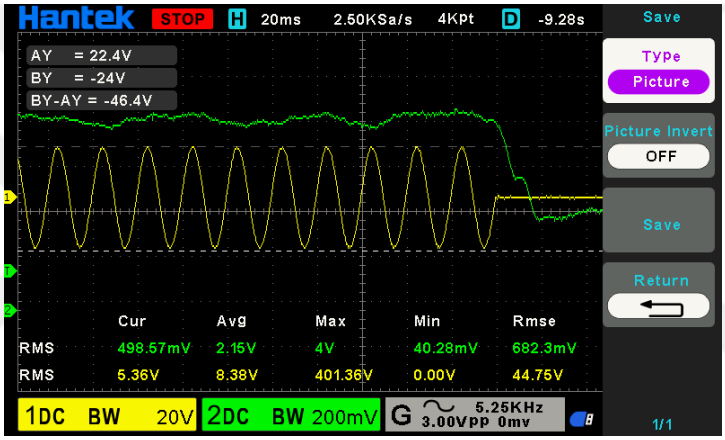


Figure 6.7 Regulation Loss with Insufficient ADC Oversampling (2x)

These striking examples have illustrated how much the success of the developed control technique relies on precise voltage measurement and, consequently, on the high rate and successful implementation of ADC oversampling. While it is possible to improve ADC readings with the help of digital LPFs in situations where oversampling is not used, regulating the output with LPF without oversampling is not an effective method in terms of measurement delay and accuracy. Therefore, we did not adopt any digital LPF throughout the development of this control technique.

To emphasize the impact of this thesis compared to similar studies in the literature, numerous studies have been reviewed. No study highlighting the effect of ADC oversampling on inverter output regulation could be found. However, there are various studies employing the classical RMS voltage regulation method or double loop control method that includes both conventional RMS voltage regulation and effect of output current on RMS voltage regulation. Each study targets different application purposes, making it unlikely to find identical test conditions as in this study.

With that purpose, we shared the particular circuit parameters and test conditions for each subject. Nonetheless, Table 6.2 is crucial for demonstrating how the dynamic response of our study compares favorably to similar studies in the literature.

Table 6.2. Dynamic Response Comparison with Current Studies

Subject Study	Test Condition	Load Transition	Maximum Error (Amplitude)	Maximum Error (Percentage)	Recovery Time (ms)
Proposed Study	Three-Phase Application Pin=125W, Vph=16VAC Fo=50Hz, Fsw=10KHz (100% ↔ 20% transition)	Light → Heavy	298mV	1,86%	147
		Heavy → Light	192mV	1,20%	108
[11]	Three-Phase Application Pin=160kVA, Vph=220VAC Fo=50Hz, Fsw=10KHz (80% ↔ 0% transition)	Light → Heavy	15V	6,82%	65
		Heavy → Light	18V	8,18%	65
[12]	Single-Phase Application Pin=60W, Vph= 35.35VAC Fo=50Hz, Fsw=10KHz, (100% ↔ 0% transition)	Light → Heavy	4.2V	11,90%	250
		Heavy → Light	4.2V	11,90%	250
[13]	Three-Phase Application Pin=10KW, Vph=110VAC Fo=50Hz, Fsw=2.5KHz (100% ↔ 0% transition)	Light → Heavy	18.1V	16,40%	100
		Heavy → Light	23.7V	21,54%	140
[14]	Single Phase Application Pin=16KW, Vph=230VAC Fo=50Hz, Fsw=5KHz (100% ↔ 67% transition)	Light → Heavy	N/A	1,07%	200
		Heavy → Light	N/A	N/A	N/A
[15]	Single Phase Application Pin=150W, Vph=230VAC Fo=50Hz, Fsw=20KHz (100% ↔ 67% transition)	Light → Heavy	120V	52,17%	600
		Heavy → Light	N/A	N/A	N/A

The first four examples in the table present real implementation results, while the last example only provides simulation results.

Some of the studies mentioned above do not include the RMS distortion amount in their articles. In such cases, the RMS error is calculated based on the information provided in these articles, such as determining the RMS error from given sinusoidal voltage errors or extracting it from provided waveforms. For example, while we obtain exact values from the article for the first study, the results presented in this table for the 2nd and 3rd studies are approximate values calculated by us.

In terms of inverter control, there is always a trade-off between RMS error and recovery time. This means that a system can be optimized in two different ways: to minimize RMS error or to minimize recovery time. Compared to other studies, the developed technique offers impressive RMS error results while not significantly sacrificing recovery time. While the closest conventional technique yields an error rate of 6.82%, our study achieves a rate of only 1.86% under similar test conditions. Moreover, the recovery time is only extended by the equivalent of 2 fundamental periods.

In the next part (Section 7), we will discuss the results, summarize all the work done within the scope of the thesis, and evaluate potential improvements. Finally, we will conclude our thesis report by providing assessments on possible future studies.

7. CONCLUSION

In this thesis, a 3-phase 2-level inverter prototype was designed. The designed prototype was controlled using an embedded novel control method developed with original code through bare-metal programming. Initially, the theoretical design of the hardware, which obtains three-phase outputs of 16V from a 52V split DC busbar, was conducted. The hardware design of the three-phase inverter aimed at achieving >90% efficiency at 125W input power was developed, and the electrical design objectives were achieved with the developed control technique. In this study, simulations of various designs such as the half-bridge structure power switches, their driving circuits, output LC filters, and precision measurement circuits were validated in the LtSpice environment.

In this study, SPWM was adopted as the modulation technique, while PI control was chosen as the main control technique. Unlike the commonly used SVPWM technique, which is often applied with vector control, this technique can be easily adapted to both single and three-phase applications. The voltage regulation method developed within the scope of this thesis, which can be implemented with PI control, is an improved version of the classical RMS voltage technique.

Within the scope of this thesis, the technique we have developed by further mitigating the classical RMS method offers significantly better dynamic response. Unlike the classical method, this developed technique relies on updating the measured RMS value in each interrupt cycle. Another advantage of the control technique is that it shifts the application frequency of the corrective effect of the PI controller from the fundamental frequency to the switching frequency level. In this way, much more effective control and output regulation are achieved, especially during instantaneous load and input voltage variations.

Simulation studies of the developed control technique were also conducted using Plecs and Matlab/Simulink environments, and the success of the developed technique was demonstrated based on the results obtained from these studies. Since it was not possible to reflect the quality of ADC measurements in the simulation studies, this aspect was covered during the implementation phase.

The success of the developed technique relies on the quality of ADC voltage measurements. In this study, precise and almost delay-free measurements were obtained using the oversampling

feature of the ADC modules of the STM32G431RB microcontroller. In other words, the developed technique has provided an optimal solution by combining the precise measurement advantage of ADC oversampling with the dynamism of the sliding RMS method based on the interrupt period. It has been proven that the dynamic performance of the classical control technique is much less satisfactory. To achieve similar dynamic performance with the classical method, a double loop control structure can be used.

In the double loop control technique, the internal current loop regulates the output current to maintain the inverter's stability by a fast-internal current loop. Meanwhile, the external voltage loop functions at a slower pace compared to the current loop and generates the current setpoint. In such a system, both the difficulty of accurately measuring and managing the load current over a wide spectrum and the addition of an extra current sensor for the dual control loop increases costs. Therefore, although similar performances can be achieved, the control technique we have developed is much lower cost and only requires maximum use of the microprocessor's capabilities.

Our study differs from similar works in terms of the method of developing the control technique. The original aspect of the study in this context is that all control software has been programmed using bare-metal programming, that is register-based codes.

The focal point of this thesis is to ensure fast regulation and minimum error control in the system during noisy environments, critical load changes, or sudden load variations such as pulse loads. Within the scope of the thesis, the dynamic response of the system has been observed by monitoring both maximum RMS voltage error (overshoot or undershoot) and minimum RMS voltage recovery time through oscilloscope outputs. The developed method has been compared with the classical method in terms of these two metrics, and its advantages have been demonstrated. Additionally, to assess the contribution of ADC oversampling to this technique, oversampling has been adjusted at different rates, and the degradation in dynamic response when reducing the oversampling rate has been proven with real oscilloscope outputs.

The primary focus of the proposed method was to ensure the stability of RMS voltage during transient loads, albeit with some trade-off in terms of THD. Although THD improvement is not the main focus of this study, we anticipate that THD degradation during load changes would be even worse in the classical method. On the other hand, efficiency results were similar for both techniques. In fact, the success of the proposed technique in terms of efficiency was observed to

be slightly better under unbalanced load conditions. The main contribution of the developed technique is to improve the dynamic response of the inverter, and we do not believe it has a major impact on steady-state efficiency. Therefore, the similarity in efficiency for both control techniques is considered a normal outcome.

The success of the developed technique has been demonstrated through linear-ohmic loads. A constant-value inductance or a constant-speed motor does not differ from ohmic loads in terms of linearity. Therefore, in future studies, testing the circuit with non-linear loads rather than different types of linear loads will open the door to more beneficial research.

Although SPWM technique was implemented in this study, the possibility of applying this technique with SVPWM method in the future is also considered. Another potential area for future work could be the investigation of the effect of the improvement in output regulation on the lifetimes of filter inductors and power switches. Another potential area for future research could be testing the developed control technique under variable busbar voltage conditions.

In conclusion, this innovative control technique programmed with bare-metal code on a 125W prototype 3-phase inverter design is an efficient method that can be applied effectively for both single-phase and 3-phase voltage source inverter applications.

REFERENCES

[1] Mohan, N. Undeland T.M. Robbins W.P. (1995) Power Electronics: Converters, Applications, and Design, 3rd Edition

[2] S. Kharjule," Voltage source inverter," 2015 International Conference on Energy Systems and Applications, Pune, India, 2015, pp. 537-542, doi: 10.1109/ICESA.2015.7503407.

[3] B. Proca and M. Comanescu," Design and Implementation of a Voltage Harmonic Elimination Control Method for the Single-Phase Inverter," 2020 IEEE 11th International Symposium on Power Electronics for Distributed Generation Systems (PEDG), Dubrovnik, Croatia, 2020, pp. 13-17, doi: 10.1109/PEDG48541.2020.9244362.

[4] V. K. Chandna, Z. A. Jaffery and S. K. Chaudhry, "Simulation of single phase RMS controlled Voltage Source Inverter (VSI) for variation in Cdc, Cs and Rs," 2010 Joint International Conference on Power Electronics, Drives and Energy Systems & 2010 Power India, New Delhi, India, 2010, pp. 1-7, doi: 10.1109/PEDES.2010.5712403.

[5] P. Varela, D. Meneses, O. Garcia, J. A. Oliver, P. Alou and J.A. Cobos, "Current mode with RMS voltage and offset control loops for a single-phase aircraft inverter suitable for parallel and 3-phase operation modes," 2011 IEEE Energy Conversion Congress and Exposition, Phoenix, AZ, USA, 2011, pp. 2562-2567, doi:10.1109/ECCE.2011.6064110.

[6] O. Tahmaz, M. N. Ekim and A. B. Yildiz, "Vector Control of Permanent Magnet Synchronous Motor by a Two-Level SPWM Inverter," 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 2020, pp. 1-7, doi: 10.1109/ISMSIT50672.2020.9254617.

[7] S. Yang, Q. Lei, F. Z. Peng and Z. Qian," A Robust Control Scheme for Grid-Connected Voltage-Source Inverters," in IEEE Transactions on Industrial Electronics, vol. 58, no. 1, pp. 202-212, Jan. 2011, doi:10.1109/TIE.2010.2045998.

[8] M. Ciobotaru, R. Teodorescu and F. Blaabjerg," Control of single-stage single-phase PV inverter," 2005 European Conference on Power Electronics and Applications, Dresden, Germany, 2005, pp. 10 pp.-P.10, doi: 10.1109/EPE.2005.219501.

[9] T. Bhattacharjee, M. Jamil and A. Jana, "Design of SPWM based three phase inverter model," 2018 Technologies for Smart-City Energy Security and Power (ICSESP), Bhubaneswar, India, 2018, pp. 1-6, doi: 10.1109/ICSESP.2018.8376696.

[10] S. Morey, R. Gurav and P. M. Joshi," Space Vector Modulation Topology for Two Level Three Phase Voltage Source Inverter," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 2018, pp. 1-5, doi: 10.1109/I2CT.2018.8529477.

[11] F. Zengin, B. G¨unay and A. Faruk Bakan," A Novel RMS Voltage Control Method for UPS Applications," 2023 14th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkiye, 2023, pp. 1-5, doi: 10.1109/ELECO60389.2023.10415997

[12] Keliang Zhou, Kay-Soon Low, D. Wang, Fang-Lin Luo, Bin Zhang and Yigang Wang, "Zero-phase odd-harmonic repetitive controller for a single-phase PWM inverter," in IEEE Transactions on Power Electronics, vol. 21, no. 1, pp. 193-201, Jan. 2006, doi: 10.1109/TPEL.2005.861190.

[13] Y. Geng, Y. Yun, R. Chen, K. Wang, H. Bai and X. Wu, "Parameters Design and Optimization for LC-Type Off-Grid Inverters With Inductor-Current Feedback Active Damping," in IEEE Transactions on Power Electronics, vol. 33, no. 1, pp. 703-715, Jan. 2018, doi: 10.1109/TPEL.2017.2664812.

[14] M. Gattamaneni and S. K. Gudey, "Design and Stability Analysis of Single-Loop Voltage Control of Voltage Source Inverter," 2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA), Arad, Romania, 2021, pp. 249-253, doi: 10.1109/ICCCA52192.2021.9666255.

[15] Diouri, Omar & Errahimi, Fatima & Es-Sbai, Najia. (2018). Regulation of the Output Voltage of an Inverter in Case of Load Variation. IOP Conference Series: Materials Science and Engineering. 353. 012021. 10.1088/1757-899X/353/1/012021.

[16] Z. Cvetkovic and M. Vetterli," Error-rate characteristics of over-sampled analog-to-digital conversion," in IEEE Transactions on Information Theory, vol. 44, no. 5, pp. 1961-1964, Sept. 1998, doi:10.1109/18.705574.

[17] Gupta, R. (2021), Difference between Inverters VSI vs CSI,

https://quick-learn.in/difference-between-inverters-vsi-vs-csi/ (accessed Jan 15, 2023).

[18] A. Krishna R and L. P. Suresh, "A brief review on multi level inverter topologies," 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 2016, pp. 1-6, doi: 10.1109/ICCPCT.2016.7530373.

[19] Schweizer, Mario & Friedli, Thomas & Kolar, Johann. (2013). Comparative Evaluation of Advanced Three-Phase Three-Level Inverter/Converter Topologies Against Two-Level Systems. IEEE Transactions on Industrial Electronics. 60. 5515-5527. 10.1109/TIE.2012.2233698.

[20] Feng, Zhijian & Zhang, Xing & Wang, Jianing & Yu, Shaolin. (2020). A High-Efficiency Three-Level ANPC Inverter Based on Hybrid SiC and Si Devices. Energies. 13. 1159. 10.3390/en13051159.

[21] Nguyen, Tuyen & Phan Quoc, Dung & Dao, Ngoc Dat & Nguyen, Nhan. (2014). The carrier - based PWM method to reduce common-mode voltage for three - level T - type neutral point clamp inverter. 1549-1554. 10.1109/ICIEA.2014.6931415.

[22] Elprocus, Multilevel Inverter – Types & Advantages, https://www.elprocus.com/multilevel-inverter-types-advantages/ (accessed April 20, 2023)

[23] R. M. Patil, V. P. Dhote and A. Thosar, "Comparative Analysis of Three Phase 5,7 9 Level Inverter Using PDPWM Technique," 2018 International Conference on Smart Electric Drives and Power System (ICSEDPS), Nagpur, India, 2018, pp. 323-328, doi: 10.1109/ICSEDPS.2018.8536031.

[24] Rashid, M. H. (2013). Power Electronics Devices, Circuits and Applications, 4th edition.

[25] Strobl, S. (2021), SVPWM vs SPWM modulation techniques, https://imperix.com/doc/implementation/svpwm-vs-spwm-modulation-techniques (accessed May 1, 2023)

[26] Strobl, S. (2021), Space Vector Modulation (SVM), https://imperix.com/doc/implementation/space-vector-modulation (accessed May 1, 2023)

[27] Control Tutorials for Matlab & Simulink, PID Compensator with Bilinear Approximation, https://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_PIDbilin (accessed May 5, 2023)

[28] Bos, Thomas & Badami, Komail & Dehaene, Wim & Verhelst, Marian. (2018). Architecture optimization for energy-efficient resolution-scalable 8–12-bit SAR ADCs. Analog Integrated Circuits and Signal Processing. 97. 10.1007/s10470-018-1235-0.

[29] The Science of Color (Second Edition), 2003, Nyguist - Shannon Sampling Theorem, https://www.sciencedirect.com/topics/computer-science/shannon-sampling-theorem (accessed June 21, 2023)

[30] Microsemi, (2013), Park – Clarke Transformations, Rev 50200359-0/11.13

https://www.microsemi.com/document-portal/doc_view/132799-park-inverse-park-and-clarke-inverse-clarke-transformations-mss-software-implementation-user-guide (accessed June 23, 2023)

[31] Vatansever, F. (2019). Clarke ve Park Dönüşümlerinin Görselleştirilmesi. Academic Perspective Procedia, 2 (3), 502-507. DOI: 10.33793/acperpro.02.03.40

[32] Analog Devices, (2023), Ltspice Simulator, Ver. 17.1.6, https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html (accessed March 29, 2024)

[33] Microsoft Corporation, (2024), Microsoft Excel, https://www.microsoft.com/tr-tr/microsoft-365/excel, (accessed May 27, 2024)

[34] Sujan, G.P., Kishore, L., 2018, Power Quality Improvement in Aerospace Applications Using Active Filter, International Journal of Scientific Engineering and Technology Research, Vol. 07, Issue 10. October 2018, p. 1788-1791.

[35] Safak B. (2019), Design and implementation of a 115 V 400 Hz inverter for aircraft systems, https://tez.yok.gov.tr/UlusalTezMerkezi/tezDetay.jsp?id=utLjIexQeeVWjxAtvrEfYw&no=WVMv1axiZS5HtiFtlDCNEg

[36] Magnetics Inc, (2023), Inductor Design Tool,

https://www.mag-inc.com/Design/Design-Tools/Inductor-Design, (accessed March 29, 2024).

[37] Di Giovanni, F. Silicon Carbide: Physics, Manufacturing, and Its Role in Large-Scale Vehicle Electrification. Chips 2023, 2, 209-222. https://doi.org/10.3390/chips2030013

[38] Bhalla, A. Considerations when comparing SiC and GaN in power applications, https://www.electronicspecifier.com/products/power/considerations-when-comparing-sic-and-gan-in-power-applications, (accessed June 17, 2023)

[39] M. Salcone and J. Bond, “Selecting film bus link capacitors for high performance inverter applications,” 2009 IEEE International Electric Machines and Drives Conference, Miami, FL, USA, 2009, pp. 1692-1699, doi: 10.1109/IEMDC.2009.5075431.

[40] Infineon Technologies, (2023), Input Capacitor (DC Link) Calculation For Single Phase Motor Bridge, Revision 1.00 2023-08-01, https://www.infineon.com/dgdl/Infineon-Input_capacitor_calculation_for_single_motor_bridge-AN-ApplicationNotes-v01_00-EN.pdf?fileId=8ac78c8c8929aa4d0189d5bbe46d21f8 (accessed April 15, 2023)

[41] Kemet, (2019), DC-Link Design Tips, https://www.kemet.com/en/us/technical-resources/dc-link-design-tips.html (accessed April 15, 2023)

[42] Texas Instruments, (2018), Fundamentals of MOSFET and IGBT Gate Driver Circuits, Rev. A, https://www.ti.com/lit/ml/slua618a/slua618a.pdf?ts=1714209573738&ref_url=https%253A%252F%252Fwww.google.com%252F (accessed May 21, 2023)

[43] Vishay, (2024), IGBT/MOSFET Gate Drive Optocoupler, Rev. 1.3, 24-Oct-11 https://www.vishay.com/docs/81227/81227.pdf (accessed Feb 28, 2024)

[44] Infineon Technologies, (2023), Datasheet: 1EDI40I12AF Rev 1.10, https://www.infineon.com/dgdl/Infineon-1EDIXXI12AF-DataSheet-v01_10-EN.pdf?fileId=db3a3043427ac3e201428e648a333734

[45] Analog Devices, (2022), Datasheet: AD622ARZ Rev. E, https://www.analog.com/media/en/technical-documentation/data-sheets/AD622.pdf

[46] Skyworks Inc, (2022), Datasheet: Si8920BC-IP Rev. A, https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/data-sheets/si8920-datasheet.pdf

[47] Analog Devices, (2010), Datasheet: LT1630CS8#PBF Rev. A, https://www.analog.com/media/en/technical-documentation/data-sheets/16301fa.pdf

[48] Microchip Technology, (2024), Datasheet: MCP1501-18E Rev. H

https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP1501-Data-Sheet-DS20005474.pdf

[49] LEM International, (2017), Datasheet: GO-20-SMS/SP3 Rev. 1

https://www.lem.com/sites/default/files/products_datasheets/go10-20-30-sms_sp3_series.pdf

[50] Monolithic Power Systems, (2011), Datasheet: MP1584 Rev 10. (NRFND)

https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP1584EN-LF-Z/document_id/204/

[51] Ozdisan, (2022), Mini Mp1584en Dc-Dc Buck Converter Adjustable Power Step Down Module, https://cdn.ozdisan.com/ETicaret_Dosya/626534_924471.pdf (accessed Mar 5, 2023)

[52] On Semiconductor, (2023), Datasheet NCP3063 Rev 11,

https://www.onsemi.com/pdf/datasheet/ncp3063-d.pdf

[53] Mornsun Power, (2012), Datasheet: B0515S-1W Rev. B/2, https://www.e-kom.com/Datasheet/B0515S-1W.pdf

[54] Infineon Technologies, (2004), Datasheet: IRFZ48NPbF Rev 01_01, https://www.infineon.com/dgdl/Infineon-IRFZ48N-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a40153563ea972222c

[55] ST Microelectronics, (2020), Datasheet: LD1117S50TR Rev. 37

https://www.st.com/resource/en/datasheet/ld1117.pdf

[56] Mathworks, (2023), “Matlab/Simulink”, Ver. R2023a,

https://www.mathworks.com/products/simulink.html (accessed April 3, 2024)

[57] Plexim GmbH, (2024), Plecsim Electrical Engineering Software, Ver. 4.8, https://www.plexim.com/download/standalone (accessed March 29, 2024)

[58] St Microelectronics, (2021), Datasheet: STM32G431xB, Rev. 6, https://www.st.com/en/microcontrollers-microprocessors/stm32g431rb.html

[59] St Microelectronics, (2023), NUCLEOG431RB Development Board,

https://www.st.com/en/evaluation-tools/nucleo-g431rb.html

[60] St Microelectronics, (2023), Stm32CubeIDE, Ver. 1.13.2, https://www.st.com/en/development-tools/stm32cubeide.html#get-software

[61] St Microelectronics, (2024), RM0440 Reference Manual, Rev. 8, https://www.st.com/en/microcontrollers-microprocessors/stm32g431rb.html#documentation

[62] Realterm Global, (2017), Realterm, Ver. 2.0.0.70, https://sourceforge.net/projects/realterm/

[63] Salmony P., (2020), PID, https://github.com/pms67/PID (Accessed Aug 10, 2023)

[64] Altium Limited, (2019), Altium Designer, Ver. 19.0.15, https://www.altium.com/products/downloads, (Accessed Oct 13, 2023)

[65] Saturn PCB Design, Inc. (2019), Saturn PCB Toolkit. Ver. 8.33, https://saturnpcb.com/saturn-pcb-toolkit/ (Accessed Oct 13, 2023)

[66] JLCPCB, https://jlcpcb.com/, (Accessed Dec 14, 2023)

APPENDIX

Appendix A.1 Example ADC Read Bare-Metal Code “main.c” Part

```
#include "stm32g4xx.h"
#include <stdint.h>
#include <stdio.h>
#include "uart.h"
#include "adc.h"
#include "system_clk_config.h"
volatile uint32_t sensor_value;
int main(void) {
  // SystemClock_Config();
  uart1_tx_init();                    // Just to print the data
  adc_init();
  activate_adc();
  start_conversion();
  while(1) {
    sensor_value = adc_read();
    printf("Sensor value: %d \n\r",(int) sensor_value);
    }
}
```

Appendix A.2 Example ADC Read Bare-Metal Code “adc.c” Part

```
#include "stm32g4xx.h"
#include "adc.h"
#include <stdint.h>
#include <SYSTICK.h>
#define LED_PIN                 (1U<<5)
#define GPIOAEN                 (1U<<0)
#define ADC12EN                 (1U<<13)
#define CH4_RANK1          (1U<<8)
#define CFGR_CNT_CONV           (1U<<13)
#define ADC_SEQ_LEN_1           0x00
#define CR_ADC_EN               (1U<<0)
#define CR_ADC_RDY              (1U<<0)
#define CR_ADSTART              (1U<<2)
#define ISR_EOC                 (1U<<2)
#define ISR_EOS                 (1U<<3)
#define CR_ADC_VREG_EN          (1U<<28)
#define DIS_DEEPPWD             (1U<<29)
#define CFGR_AUTODLY            (1U<<14)
void pa7_adc_init(void) {
  /*CONFIGURE GPIO PORT AS ADC*/
  //1- Enable clock access to adc pin's GPIO port (GPIOA)
  RCC->AHB2ENR |= GPIOAEN;
  //2- Conf. the mode of PA7 as analog (ADC2_IN4 -> Channel 4)
```

```
GPIOA->MODER |= (1U<<14);

GPIOA->MODER |= (1U<<15);

GPIOA->MODER |= (1U<<10);

GPIOA->MODER &=~(1U<<11);

/*CONFIGURE ADC PERIPHERAL FOR DATA CONVERSION*/

//3- Enable clock access to ADC

RCC->AHB2ENR |= ADC12EN;

//4- Set ADC common parameters: clock source & prescaler

//Clear bits

ADC12_COMMON->CCR &=~ (1U<<16);

ADC12_COMMON->CCR &=~ (1U<<17);

//Set 1 0 to bits 17 16 ->  adc_hclk/4

ADC12_COMMON->CCR |= (1U<<16);

ADC12_COMMON->CCR |= (1U<<17);

//5- Set ADC trigger source (//Clear bits for SW trigger)

ADC2->CFGR &=~ (1U<<10);

ADC2->CFGR &=~ (1U<<11);

//6- Set conversion type: continuous / single

ADC2->CFGR |= CFGR_CNT_CONV;

//Select what happens when overrun occurs

ADC2->CFGR |= CFGR_AUTODLY;

//7- Set the conversion sequence length

ADC2->SQR1 = ADC_SEQ_LEN_1;       //We only have single channel as Ch.4

//8- Set the conversion sequence start channel
```

```
  ADC2->SQR1 |= CH4_RANK1;              // Set CH4 as first sequence
}
void activate_adc(void) {
  //1- Enable ADC internal voltage regulator
  ADC2->CR &=~ DIS_DEEPPWD;             // Clear DEEPPWD
  ADC2->CR |= CR_ADC_VREG_EN;
  SYSTICKDelayMs(1);                    // Wait for 20us start-up duration
  //2- Enable ADC module
  ADC2->ISR |= CR_ADC_RDY;              // Clear ADC READY
  ADC2->CR |= CR_ADC_EN;
}
void start_conversion(void) {
  // Start conversion
  while(!(ADC2->ISR & CR_ADC_RDY)) {}      // If ADRDY is set initialize the conversion
  ADC2->CR |= CR_ADSTART;
}
uint32_t adc_read(void) {
  //1- Wait for conversion to be completed
  if(!(ADC2->ISR & ISR_EOC)) {
    GPIOA->BSRR |= LED_PIN;
  }
  //2- Read converted value
  return (ADC2->DR);
}
```

Appendix A.3 Example ADC Read Bare-Metal Code “adc.h” Part

```
#ifndef ADC_H_

#define ADC_H_

#include <stdint.h>

void pa7_adc_init(void);

void activate_adc(void);

void start_conversion(void);

uint32_t adc_read(void);

#endif /* ADC_H_ */
```

Appendix B Timer Configuration Bare-Metal Pseudo Code

```
void TIM1_Config(void) {
  TIM1->SR=0;
  TIM1->ARR=8200;
  TIM1->PSC=0;                          //  PWM timer is set to 10 KHz
  TIM1->CNT=0;
  TIM1->CCMR1=0x00006868;          //  Preload Enable , PWM Mode:1 Selected
  TIM1->CCMR2=0x00000068;
  TIM1->CCER=0x00000555;               //  PWMs and their dedicated compl. outputs are enabled
  SET_BIT(TIM1->BDTR,TIM_BDTR_OSSI);
  SET_BIT(TIM1->BDTR,TIM_BDTR_OSSR);
  TIM1->DIER=0;
  SET_BIT(TIM1->DIER,TIM_DIER_UIE);                    //  Update interrupt enable
  TIM1->BDTR |= 0x32;                                          //  Dead time = 1/164M*50 = 305ns.
#ifdef TIM1UEVHWTRIG
  CLEAR_BIT(TIM1->CR2,TIM_CR2_MMS2_3);
  CLEAR_BIT(TIM1->CR2,TIM_CR2_MMS2_2);
  SET_BIT(TIM1->CR2,TIM_CR2_MMS2_1);
  CLEAR_BIT(TIM1->CR2,TIM_CR2_MMS2_0);
#endif
  NVIC_SetPriority(TIM1_UP_TIM16_IRQn,2);              // Update interrupt
  NVIC_EnableIRQ(TIM1_UP_TIM16_IRQn);
return;
}
```

Appendix C ADC Configuration Bare-Metal Pseudo Code

```
/*    ADC1 CONFIG     */
CLEAR_BIT(ADC1->CR, ADC_CR_DEEPPWD);
SET_BIT(ADC1->CR, ADC_CR_ADVREGEN);
For (uint8_t i=0;i<255;i++) {__NOP();}
/*  Clear all flags */
ADC1->ISR |= 0x7FF
/* Oversampling shift: 5-bits */
SET_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_0);
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_1);
SET_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_2);
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSS_3);
// Oversampling ratio: 32x
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSR_0);
CLEAR_BIT(ADC1->CFGR2, ADC_CFGR2_OVSR_1);
SET_BIT(ADC1->CFGR2, ADC_CFGR2_OVSR_2);
SET_BIT(ADC1->CFGR2, ADC_CFGR2_ROVSE);
SET_BIT(ADC1->CR, ADC_CR_ADCAL);
while (READ_BIT(ADC1->CR,ADC_CR_ADCAL)) __NOP();
/* Channel Length to be converted is set to "5" */
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_L_0);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_L_1);
SET_BIT(ADC1->SQR1, ADC_SQR1_L_2);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_L_3);
```

```
/* SQ1=ADC1_IN6 -> PC0 (PH1_VLT) */
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ1_0);
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ1_1);
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ1_2);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ1_3);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ1_4);
/* SQ2=ADC1_IN7 -> PC1 (PH2_VLT) */
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ2_0);
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ2_1);
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ2_2);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ2_3);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ2_4);
/* SQ3=ADC1_IN8 -> PC2 (PH3_VLT) */
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ3_0);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ3_1);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ3_2);
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ3_3);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ3_4);
/* SQ4=ADC1_IN3 -> PA2 (DCIN+_VLT) */
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ4_0);
SET_BIT(ADC1->SQR1, ADC_SQR1_SQ4_1);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ4_2);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ4_3);
CLEAR_BIT(ADC1->SQR1, ADC_SQR1_SQ4_4);
```

```
/* SQ5=ADC1_IN4 -> PA3 (DCIN-_VLT)   */
CLEAR_BIT(ADC1->SQR2, ADC_SQR2_SQ5_0);
CLEAR_BIT(ADC1->SQR2, ADC_SQR2_SQ5_1);
SET_BIT(ADC1->SQR2, ADC_SQR2_SQ5_2);
CLEAR_BIT(ADC1->SQR2, ADC_SQR2_SQ5_3);
CLEAR_BIT(ADC1->SQR2, ADC_SQR2_SQ5_4);
SET_BIT(ADC1->CR, ADC_CR_ADEN);
while(!READ_BIT(ADC1->ISR, ADC_ISR_ADRDY)) __NOP();
SET_BIT(ADC1->ISR, ADC_ISR_ADRDY);
```

Appendix D Moving RMS Filter Bare-metal Code “main.c” Part

```
typedef struct {
  uint16_t L;                    // Window length
  float invL;
  uint16_t count;                // Current position in the circular buffer
  uint32_t in_sq_L[300];         // Circular buffer consists of 300 element
  float avg_sq;          // Estimated average squared RMS (not root!)
  uint32_t ACC;                  // Accumulator
  } MOV_StructType;
void moving_rms_init (MOV_StructType *mRMS_x, uint16_t L) {
  mRMS_x->L = mRMS_WindowLength;             // Get the RMS window length to store
  mRMS_x->invL = 1.0f / mRMS_x->L;           // Get 1/L
  mRMS_x->count = 0;                         // Reset the point number of circ. buffer
  /* Clear circular buffer */
  for (uint16_t n=0; n<mRMS_x->L; n++) {
    mRMS_x->in_sq_L[n] = 0;
  }
  /* Clear output */
  mRMS_x->ACC = 0;
  mRMS_x->avg_sq = 0.0f;
}
float moving_rms_update (MOV_StructType *mRMS, int32_t in)  {
  // "in" is the n-th sample of my sinusoid (cordic sine itself)
  // Compute input-squared
```

```
  uint32_t in_sq = in*in;
  // Store value in circular buffer (to use in M samples)
  mRMS->ACC+= in_sq - mRMS->in_sq_L[mRMS->count];
  // Compute latest output squared using recursive equation
  mRMS->avg_sq= mRMS->ACC*mRMS->invL;
  mRMS->in_sq_L[mRMS->count] = in_sq;
  // Update circular buffer pointer
  if ( mRMS->count == (mRMS->L-1) )  mRMS->count = 0;
  else { mRMS->count++;              /* The next in2 will be stored in indx+1 */  }
  return mRMS->avg_sq;               // Returns squared value
                                     // We get the root of this in interrupt
}
```

Appendix E Body of Main Bare-metal Embedded Control Algorithm

```
void TIM1_UP_TIM16_IRQHandler(void) {
   /* SINUSOIDAL REFERENCE GENERATION */
  CordicAngle+=60;
  CordicAngle2+=60;
  CordicAngle3+=60;
  if(CordicAngle>11999) CordicAngle=CordicAngle-12000;
  if(CordicAngle2>11999) CordicAngle2=CordicAngle2-12000;
  if(CordicAngle3>11999) CordicAngle3=CordicAngle3-12000;
  CordicSine=Sine[(uint16_t)CordicAngle]* 0.0005 *mod_ind;
  CordicSine2=Sine[(uint16_t)CordicAngle2]* 0.0005 *mod_ind2;
  CordicSine3=Sine[(uint16_t)CordicAngle3]* 0.0005 *mod_ind3;
  /* COMPLEMENTARY PWM GENERATION */
 static uint16_t DCOFFSET=4100;
 TIM1->CCR1= (uint16_t)(CordicSine*4100+DCOFFSET);              // PA8  &  PB13
 TIM1->CCR2= (uint16_t)(CordicSine2*4100+DCOFFSET);             // PA9  &  PB14
 TIM1->CCR3= (uint16_t)(CordicSine3*4100+DCOFFSET);             // PA10 &  PB15
 /* VOLTAGE MEASUREMENTS */
 PH1_VLT = (1.4814815 * (PC0_SEQ1 * 0.00080830281f) - 1.8f) * 25;
 PH2_VLT = (1.4814815 * (PC1_SEQ2 * 0.00080830281f) - 1.8f) * 25;
 PH3_VLT = (1.4814815 * (PC2_SEQ3 * 0.00080830281f) - 1.8f) * 25;
 DCINP_VLT = (1.4814815 * (PA2_SEQ4 * 0.00080830281f) - 1.8f) * 25;
 DCINN_VLT = (1.4814815 * (PA3_SEQ5 * 0.00080830281f) - 1.8f) * 25;
 /* CURRENT MEASUREMENTS */
```

```
PH1_CR = (PC3_SEQ1 * 0.00080830281f);
PH2_CR = (PA0_SEQ2 * 0.00080830281f);
PH3_CR = (PA1_SEQ3 * 0.00080830281f);
DCINP_CR = (PA6_SEQ4 * 0.00080830281f);
DCINN_CR = (PA7_SEQ5 * 0.00080830281f);
/* RMS CODE */
mRMS_SqEstimate = moving_rms_update(&mRMS, ROUNDNUM(PH1_VLT));
mRMS_SqEstimate2 = moving_rms_update2(&mRMS2, ROUNDNUM(PH2_VLT));
mRMS_SqEstimate3 = moving_rms_update3(&mRMS3, ROUNDNUM(PH3_VLT));
mRMS_SqRtEstimate = ((sqrtf(mRMS_SqEstimate)));
mRMS_SqRtEstimate2 = ((sqrtf(mRMS_SqEstimate2)));
mRMS_SqRtEstimate3 = ((sqrtf(mRMS_SqEstimate3)));
/* PID CODE */
measurement = mRMS_SqRtEstimate;
measurement2 = mRMS_SqRtEstimate2;
measurement3 = mRMS_SqRtEstimate3;
/* START CLOSED LOOP CONTROL AFTER 10SEC OF OPEN LOOP OPERATION */
if(Power_start) {
  A=4100;
  if (delay_counter < 100000.0f) {
    mod_ind = 0.9f;
    mod_ind2 = 0.9f;
    mod_ind3 = 0.9f;
    delay_counter++;
```

```
      }
      else {
        mod_ind = PIDController_Update(&pid, setpoint, measurement);
        mod_ind2 = PIDController_Update(&pid, setpoint, measurement2);
        mod_ind3 = PIDController_Update(&pid, setpoint, measurement3);
        }
  }
  else {
    A=0;
    delay_counter=0;
   }
return;
}
```