UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Trabajo de Final de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Volumen I: Memoria

Diseño e implementación de un inversor trifásico dual para tracción eléctrica

Autor:
David Redondo Viñolo

Director:
Prof. Alfonso Conesa Roca

Convocatoria: Junio 2024

Resumen

Este Trabajo de Fin de Grado se enfoca en el diseño y prototipado de un inversor trifásico dual de 80 kW ($2 \cdot 40$ kW) y 600 V con control vectorial (FOC) para motores síncronos de imanes permanentes (PMSMs). Este inversor bidireccional busca ser una solución compacta y de alto rendimiento para aplicaciones de tracción eléctrica en vehículos de Formula Student.

En primer lugar se estudia en detalle la máquina eléctrica, abordando su modelo matemático y realizando simulaciones implementando el control vectorial. Posteriormente, se diseña el *hardware* que permitirá integrar este control, y por último, se implementa gran parte del *firmware* necesario para ello. Se validan con rigurosidad los aspectos de la electrónica de potencia para asegurar su funcionamiento conforme a los requisitos establecidos.

El proyecto ha alcanzado una notable densidad de potencia mediante la implementación de tecnologías de vanguardia como los semiconductores de carburo de silicio (SiC). Se plantean líneas de trabajo futuras para mejorar el algoritmo de control, y optimizar el *hardware* y *firmware* del inversor.

Además, este proyecto se concibe como el punto de partida para el desarrollo continuo de un inversor de alto rendimiento. Se espera que los conocimientos adquiridos y los resultados obtenidos sirvan como base sólida para un diseño robusto y eficiente.

Resum

Aquest Treball de Fi de Grau s'enfoca en el disseny i prototipat d'un inversor trifàsic dual de 80 kW ($2 \cdot 40$ kW) i 600 V amb control vectorial (FOC) per a motors síncrons d'imants permanents (PMSMs). Aquest inversor bidireccional cerca ser una solució compacta i d'alt rendiment per a aplicacions de tracció elèctrica en vehicles de Formula Student.

En primer lloc s'estudia detalladament la màquina elèctrica, abordant el seu model matemàtic i realitzant simulacions implementant el control vectorial. Posteriorment, es dissenya el *hardware* que permetrà integrar aquest control, i finalment, s'implementa gran part del *firmware* necessari per a això. Es validen amb rigorositat els aspectes de l'electrònica de potència per a assegurar el seu funcionament conforme als requisits establerts.

El projecte ha aconseguit una notable densitat de potència de mitjançant la implementació de tecnologies d'avantguarda com els semiconductors de carbur de silici (SiC). Es plantegen línies de treball futures per a millorar l'algorisme de control, i optimitzar el *hardware* i *firmware* de l'inversor.

A més, aquest projecte es concep com el punt de partida per al desenvolupament continu d'un inversor d'alt rendiment. S'espera que els coneixements adquirits i els resultats obtinguts serveixin com a base sòlida per a un disseny robust i eficient.

Abstract

This Bachelor's Thesis focuses on the design and prototyping of a dual 80 kW (2 · 40 kW) three-phase inverter with field-oriented control (FOC) for permanent magnet synchronous motors (PMSMs). This bidirectional inverter aims to be a compact and high-performance solution for electric traction applications in Formula Student vehicles.

Firstly, the electric machine is studied in detail, addressing its mathematical model and performing simulations implementing field-oriented control. Subsequently, the hardware is designed to integrate this control, and finally, a significant portion of the firmware necessary for this integration is implemented. The aspects of power electronics are rigorously validated to ensure its operation according to the established requirements.

The project has achieved a notable power density through the implementation of cutting-edge technologies such as silicon carbide (SiC) semiconductors. Future work is proposed to improve the control algorithm and optimize the inverter hardware and firmware.

Furthermore, this project is conceived as the starting point for the continuous development of a high-performance inverter. It is expected that the knowledge acquired and the results obtained will serve as a solid foundation for a robust and efficient design.

Agradecimientos

A mi familia y amistades, cuyo apoyo incondicional ha sido fundamental en cada etapa de este proyecto. A mi tutor, Alfonso Conesa, por sus valiosos y necesarios consejos, que han guiado el trabajo. Al CITCEA-UPC, por el soporte técnico brindado, sin el cual este trabajo no habría sido posible. A e-Tech Racing y a todas las personas que lo han formado, forman o formarán, por convertir un taller en un hogar.

Glosario

3D	Three-Dimensionsal	Tridimensional
AC	Alternating Current	Corriente Alterna
ADC	Analog to Digital Converter	Conversor Analógico a Digital
BEMF	Back Electromotive Force	Fuerza Contra-Electromotriz
BOM	Bill Of Materials	Lista de Materiales
CAD	Computer-Aided Design	Diseño Asistido por Computadora
CAN	Controller Area Network	Red de Área Controladora
CLC	Current Limit Circle	Círculo de Límite de Corriente
CTC	Constant Torque Curve	Curva de Par Constante
CVL	Current and Voltage Limits	Límites de Corriente y Voltaje
DC	Direct Current	Corriente Continua
DMA	Direct Memory Access	Acceso Directo a la Memoria
DSP	Digital Signal Processor	Procesador de Señales Digitales
ECU	Electronic Control Unit	Unidad de Control Rlectrónico
EEPROM	Electrically Erasable Programmable Read-Only Memory	Memoria de Solo Lectura Programable y Borrable Eléctricamente
EM	Electric Machine	Máquina eléctrica
EMI	Electromagnetic Interference	Interferencia Electromagnética
EMR	Energetic Macroscopic Representation	Representación Macroscópica de la Energía
ENIG	Electroless Nickel Immersion Gold	Oro de Inmersión de Níquel Sin Electrodes
ESD	Electrostatic Discharge	Descarga Electrostática
FET	Field-Effect Transistor	Transistor de Efecto de Campo
FOC	Field Oriented Control	Control Orientado al Campo
FPU	Floating Point Unit	Unidad de Punto Flotante
FSM	Finite State Machine	Máquina de Estados Finitos
FW	Field Weakening	Debilitamiento de Campo
GaN	Gallium Nitride	Nitruro de Galio

GPIO	General Purpose Input/Output	Entrada/Salida de Propósito General
HiZ	High Impedance	Alta Impedancia
I	Integral	Integral
I^2C	Inter-Integrated Circuit	Circuito Inter-Integrado
IDE	Integrated Development Environment	Entorno de Desarrollo Integrado
IGBT	Insulated Gate Bipolar Transistor	Transistor Bipolar de Puerta Aislada
IPM	Interior Permanent Magnet	Imanes Permanentes Interiores
JTAG	Joint Test Action Group	Grupo de Acción de Pruebas Conjuntas
KISS	Keep It Simple, Stupid	Mantenlo Simple, Estúpido
LDO	Low-Dropout Regulator	Regulador de Baja Caída
LED	Light-Emitting Diode	Diodo Emisor de Luz
LUT	Look-Up Table	Tabla de Búsqueda
LV	Low Voltage	Baja Tensión
LVS	Low Voltage System	Sistema de Baja Tensión
MCU	Microcontroller Unit	Microcontrolador
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor	Transistor de Efecto de Campo Semiconductor de Óxido Metálico
MTPA	Maximum Torque Per Ampere	Máximo Par Por Amperio
MTPV	Maximum Torque Per Volt	Máximo Par Por Voltio
PCB	Printed Circuit Board	Placa de Circuito Impreso
PI	Proportional-Integral	Proporcional-Integral
PID	Proportional-Integral-Derivative	Proporcional-Integral-Derivativo
PMSM	Permanent Magnet Synchronous Machine	Máquina Síncrona de Imanes Permanentes
PWM	Pulse Width Modulation	Modulación de Ancho de Pulso
RMS	Root Mean Square	Media Cuadrática
RPM	Revolutions Per Minute	Revoluciones Por Minuto
Si	Silicon	Silicio
SiC	Silicon Carbide	Carburo de Silicio
SMD	Surface Mounted Device	Dipositivo Montado en Superficie
SPI	Serial Peripheral Interface	Interfaz de Periféricos Serie
SPM	Surface Permanent Magnet	Imanes Permanentes Superficiales
SPWM	Sinusoidal Pulse Width Modulation	Modulación de Ancho de Pulso Sinusoidal
SVPWM	Space Vector Pulse Width Modulation	Modulación de Ancho de Pulso por Espacio Vectorial

SWD	Serial Wire Debug	Depuración por Cable Serie
TH	Torque Hyperbolas	Hipérbolas de Par
THD	Total Harmonic Distortion	Distorsión Armónica Total
TS	Tractive System	Sistema de Tracción
TSAL	Tractive System Active Light	Luz de Sistema de Tracción Activo
UART	Universal Asynchronous Receiver-Transmitter	Transmisor-Receptor Asíncrono Universal
USART	Universal Synchronous-Asynchronous Receiver-Transmitter	Transmisor-Receptor Síncrono-Asíncrono Universal
USB	Universal Serial Bus	Bus Serie Universal
VLE	Voltage Limit Ellipse	Elipses de Límite de Voltaje
VSI	Voltage Source Inverter	Inversor de Fuente de Voltaje
WBG	Wide Bandgap	Banda Prohibida Ancha

Índice general

1. Introducción y objetivos **12**
1.1. Introducción . . . 12
1.2. Objetivos . . . 13
1.3. Contexto y justificación . . . 14
1.4. Aplicación práctica . . . 14

2. Estado del Arte **16**
2.1. Bamocar PG-D3-700-400 . . . 16
2.2. MOBILE DCU 60/60 . . . 17
2.3. AMKASYN KW26-S5-FSE-2Q . . . 18
2.4. CM200 . . . 19

3. Metodología **20**
3.1. Principios de diseño . . . 20
3.1.1. Calidad sobre objetivos . . . 20
3.1.2. KISS . . . 21
3.2. Modelo en V . . . 22
3.3. Gestión del proyecto con Git . . . 26
3.3.1. Introducción a Git . . . 26
3.3.2. Funcionamiento de Git . . . 26
3.3.3. Por qué utilizar un sistema de control de versiones . . . 26
3.3.4. Uso de Git en el proyecto . . . 27
3.3.5. GitHub . . . 27
3.3.6. Automatización de la documentación y *releases* . . . 27

4. Desarrollo **28**
4.1. Modelo matemático del PMSM . . . 28
4.1.1. Marco de referencia estático . . . 28
4.1.2. Representación en el espacio vectorial . . . 29
4.1.3. Transformadas . . . 30
Transformada de Clarke ortonormal . . . 30
Transformada de Clarke de amplitud constante . . . 32
Transformada de Park . . . 33
4.1.4. Marco de referencia rotatorio . . . 34
4.1.5. Curvas características del PMSM . . . 37
Curva de par-velocidad y potencia-velocidad . . . 37
CLC (Círculo de Límite de Corriente) . . . 38
TH (Hipérbolas de Par) . . . 39
VLE (Elipses de Límite de Voltaje) . . . 40
4.2. Control del PMSM en el espacio $d-q$ . . . 42
4.2.1. Trayectorias de control . . . 42
MTPA (Máximo Par Por Amperio) . . . 42
CTC (Curva de Par Constante) . . . 43

MTPV (Máximo Par Por Voltio) 44
CVL (Límites de Corriente y Voltaje) 45
4.2.2. Diseño y simulación del control 47
EMR (Representación macroscópica energética) 48
Parámetros y curvas del PMSM simulado 49
Lazos de corriente y modelo promediado del inversor 50
Implementación de las trayectorias de control 53
Modelo conmutado 55
4.3. *Hardware* 65
4.3.1. Requisitos y pre-concepto 65
Potencia 65
Velocidad 68
Normativa 68
Interfaz de comunicación 72
Resumen de requisitos de *hardware* 72
Boceto del empaquetado 72
4.3.2. Topología y concepto 74
4.3.3. Semiconductores 76
Tecnología de semiconductores 76
Módulos de potencia 76
Análisis de pérdidas 77
4.3.4. Sistema de refrigeración 83
4.3.5. *Gate drivers* 84
Funcionamiento genérico 84
Criterios de selección 85
Comparativa de alternativas 85
Cálculos del *gate driver* seleccionado 87
4.3.6. Bus de condensadores 88
Función del bus de condensadores 88
Dimensionamiento del bus de condensadores 89
Selección de condensadores 90
4.3.7. Conectores de potencia 91
4.3.8. Sensor de posición 93
4.3.9. Microcontrolador 95
4.3.10. *Software* de CAD electrónico 96
4.3.11. PCB de potencia 97
Concepto y *layout* 97
Restricciones y enrutado 98
Bloques funcionales 102
Circuitos importantes 103
Resultado final 110
4.3.12. PCB de control 111
Concepto y *layout* 111
Restricciones y enrutado 112
Bloques funcionales 113
Configuración de *hardware* del MCU 114
Circuitos importantes 115

Resultado final . . . 123
4.3.13. Ensamblaje del convertidor . . . 123
Diseño en CAD . . . 123
Ensamblaje real . . . 124
4.4. *Firmware* . . . 130
4.4.1. Desarrollo del *firmware* . . . 130
Herramientas utilizadas . . . 130
Plataforma de desarrollo . . . 132
Lenguaje de programación . . . 133
Estilo de programación . . . 133
4.4.2. Arquitectura del *firmware* . . . 135
`INVERTER.c/.h` . . . 135
`MOTOR.c/.h` . . . 136
`FSM.c/.h` . . . 136
`ERRORS.c/.h` . . . 137
`MEASUREMENTS.c/.h` . . . 138
`REFERENCE.c/.h` . . . 139
`PCB_IO.c/.h` . . . 140
`CONTROL.c/.h` . . . 140
`PWM.c/.h` . . . 141
`CAN_e-Tech.c/.h` . . . 141
4.4.3. Implementación del *firmware* . . . 141
Configuración del MCU . . . 141
Manejo de interrupciones . . . 146
Lazo de control . . . 149

5. Resultados y validación **151**
5.1. Introducción . . . 151
5.2. Validación de *hardware* . . . 151
5.2.1. Pre-inspección de la placa de potencia . . . 152
5.2.2. Primer contacto con la placa de potencia . . . 153
Alimentación del *gate driver* . . . 153
Funcionalidad del *gate driver* . . . 155
Sensado de corriente . . . 156
Descarga . . . 156
Sensado de voltaje . . . 158
Resumen de errores encontrados y revisión de la PCB . . . 159
5.2.3. Conmutación de una rama como DC-DC síncrono . . . 160
5.2.4. Pruebas térmicas y de alta tensión . . . 169
5.2.5. Validación de la placa de control . . . 172
Alimentación . . . 172
MCU . . . 172
Retroalimentación . . . 173
Front-end analógico de la placa de potencia . . . 173
5.3. Validación de *firmware* . . . 173
5.3.1. Conmutación de las tres ramas . . . 173
5.3.2. Lazo abierto de tensión con carga R-L . . . 174

5.3.3. Lazo abierto de tensión con un PMSM 177

6. Consideraciones finales **179**

7. Bibliografía **191**

1. Introducción y objetivos

1.1. Introducción

La Formula Student es una competición internacional que desafía a estudiantes de ingeniería de todo el mundo a diseñar, construir y competir con monoplazas diseñados, construidos y pilotados por los mismos estudiantes. Esta competición proporciona una plataforma excepcional para que los futuros ingenieros pongan en práctica sus conocimientos y adquieran experiencia práctica en todos los ámbitos de la ingeniería, desde la manufactura hasta la gestión de proyectos. El énfasis en la innovación, la eficiencia y la adaptabilidad aporta un valor incalculable a la formación de los jóvenes ingenieros.

Dentro de los puntos clave de la Formula Student en los últimos años, se encuentra el interés creciente en la tracción eléctrica. En un mundo cada vez más preocupado por la sostenibilidad y la eficiencia energética, los motores eléctricos han surgido como una opción atractiva para la propulsión de vehículos de potencias y autonomías cada vez más elevadas. Este cambio de paradigma plantea la cuestión fundamental de cómo diseñar y controlar eficazmente los sistemas eléctricos para lograr un alto rendimiento sin dejar de ser accesibles para el público general. La optimización de costes no suele ser un reto en los deportes de motor, incluida la Formula Student, pero la innovación que se lleva a cabo en estos contextos de libertad absoluta permite traer ideas de las competiciones a los vehículos de calle. En este contexto, se establece un puente fundamental entre el presente y el futuro de la movilidad sostenible, explorando los elementos técnicos que impulsan el rendimiento de los motores eléctricos y su control, contribuyendo a dar forma al panorama de la movilidad del mañana. Además, la tracción eléctrica pura no es la única rama de la industria beneficiada por ingenieros conocedores de estos sistemas. Los trenes de potencia híbridos, los generadores en centrales energéticas e incluso los sistemas de gestión de la energía en hogares pueden volverse mucho más eficientes gracias a la investigación y el conocimiento en baterías, motores eléctricos, electrónica de potencia e integración.

Este proyecto se sitúa en el corazón de esta revolución en los deportes de motor, donde la ingeniería se fusiona con la sostenibilidad y la competición para forjar una nueva generación de soluciones de tracción eléctrica. A través de un enfoque riguroso en el diseño y control de motores eléctricos, este proyecto busca avanzar en el conocimiento y la aplicación de tecnologías de vanguardia, contribuyendo así a la formación de ingenieros y al desarrollo de soluciones de movilidad más ecológicas.

1.2. Objetivos

Objetivo 1

Adquirir **conocimiento** sobre control de motores eléctricos y diseño de convertidores de potencia.

En este primer objetivo se busca aprender, ya que se requiere de familiaridad con la electrónica de potencia y la teoría de control para poder abordar el proyecto. Además, se deberá profundizar en el análisis específico de los motores síncronos de imanes permanentes, con tal de implementar un lazo de control adecuado.

Objetivo 2

Definir unos **requisitos** para el *hardware* del inversor de tracción ideal para el equipo e-Tech Racing de la UPC-EEBE.

La identificación clara y precisa de los requisitos del *hardware* del inversor es crucial para garantizar su adecuación a las necesidades del equipo e-Tech Racing y cumplir con la normativa de la Formula Student.

Objetivo 3

Diseñar el *hardware* del inversor basado en esos requisitos.

Se llevará a cabo un diseño detallado del *hardware* del inversor, seleccionando componentes apropiados y diseñando PCBs que los integren de manera compacta.

Objetivo 4

Validar el *hardware* del inversor.

Se realizará una evaluación exhaustiva de cada parte del *hardware* del inversor para asegurar su funcionalidad y cumplimiento de los requisitos.

Objetivo 5

Implementar un **control vectorial** que permita el control independiente de **dos motores** con un solo microcontrolador.

Se procederá a la implementación del *firmware* que controlará el *hardware* del inversor, integrando el control diseñado previamente.

1.3. Contexto y justificación

La electrónica de potencia está viviendo grandes avances para aplicaciones de todos los rangos de potencia gracias a los avances en semiconductores de banda prohibida ancha (WBG). Los dispositivos semiconductores de carburo de silicio (SiC) están permitiendo mayores densidades de potencia en convertidores que van desde los centenares de vatios hasta alcanzar el megavatio, debido a la reducción de pérdidas y la alta conductividad térmica del material en comparación con sus equivalentes de silicio tradicional. De forma similar, los dispositivos de nitruro de galio (GaN) consiguen miniaturizar convertidores desde unos pocos vatios hasta casi el kilovatio. Otros avances en componentes, como la tecnología de condensadores de película, aceleran el proceso de miniaturización, permitiendo conseguir densidades de potencia más elevadas. Junto a estos dispositivos de nueva generación, avances en las técnicas de control y modulación permiten reducir las pérdidas y aumentar la eficiencia, reduciendo los requisitos de gestión térmica y permitiendo diseñar convertidores aún más pequeños.

1.4. Aplicación práctica

En este trabajo se explorará el diseño y control de un inversor trifásico dual de alto rendimiento para motores PMSM, aplicado al entorno de la Formula Student. Se abordarán los aspectos técnicos y prácticos de este proyecto, desde la selección de componentes hasta la implementación de algoritmos de control avanzados. Para contextualizar este proyecto, se hace referencia directa a la investigación previa realizada por [37], reconociendo los esfuerzos previos.

En particular, el diseño de esta controladora tendrá en cuenta las necesidades específicas de e-Tech Racing, el equipo de Formula Student de la UPC-EEBE. Desde su fundación en 2013, este equipo ha construido monoplazas año tras año para competir en eventos en España, República Checa, Italia, Holanda y Alemania.

Figura 1.1: ETR-08, el monoplaza con el que e-Tech Racing participó en las competiciones del verano de 2023.

Tras cinco años de evolución de la misma plataforma, el equipo se encuentra diseñando y construyendo un nuevo concepto en el momento en que se redacta este trabajo. El cambio principal son los motores eléctricos, cuyo nuevo diseño permitirá integrarlos en las propias ruedas del monoplaza debido a su compacidad, liberando así espacio del chasis. Se usarán dos motores, uno para cada rueda trasera, aunque el plan a largo plazo es implementar otros dos motores en las ruedas delanteras. Para controlarlos, se utilizarán dos inversores Bamocar D3 700-400 de la empresa alemana Unitek. Estos inversores están sobredimensionados en potencia y ocupan un espacio considerable dentro del chasis. Además, existen limitaciones en los parámetros modificables, lo que impide programar un control óptimo para el motor.

Para que el inversor sea fácilmente integrable con los futuros monoplazas del equipo, necesita contar con algunos componentes usados actualmente por el equipo. Por ejemplo, se usará el mismo microcontrolador que para el resto de ECUs [6], los mismos conectores de potencia y comunicación, y PCBs de estilos similares al resto de circuitos del monoplaza, con tal de que los fabricantes que colaboran con el equipo puedan fabricar todos los circuitos impresos del inversor. Esto permitirá una integración fluida del inversor como una ECU más del monoplaza, con el añadido de la electrónica de potencia personalizada.

2. Estado del Arte

En este capítulo se revisan diseños de inversores de características similares a las que debería tener este proyecto, especialmente aquellos que se usan en vehículos de Formula Student. La mayoría de coches eléctricos que compiten en eventos de Formula Student utilizan inversores comerciales. A continuación se describen las opciones más usadas entre los diferentes equipos.

2.1. Bamocar PG-D3-700-400

Figura 2.1: Inversor comercial Bamocar PG-D3-700-400 de la empresa alemana Unitek. [32]

- Fabricante: UniTek GmbH
- Potencia máxima: 135 kW
- Tensión DC máxima: 700 V
- Semiconductores: Si IGBT, módulos *half-bridge*
- Capacidad del bus DC: 320 µF
- Frecuencia de conmutación: 16 kHz

- Sensor de posición: Encoder incremental, Encoder seno-coseno, Resolver
- Control: Control vectorial, consigna de corriente.

Esta opción es una de las más escogidas por equipos en sus primeros años, ya que el inversor no es extremadamente difícil de programar y las puestas en marcha suelen ser rápidas. La desventaja principal es su peso y volumen, ya que están pensados para aplicaciones menos exigentes en cuanto a *packaging* que un monoplaza de competición.

2.2. MOBILE DCU 60/60

Figura 2.2: Inversor comercial MOBILE DCU 60/60 de la marca suiza Lenze. [17]

- Fabricante: Lenze Shchmidhauser
- Potencia máxima: 2x60 kW
- Tensión DC máxima: 848 V
- Semiconductores: Si IGBT, módulos *six-pack*
- Capacidad del bus DC: 240 µF
- Frecuencia de conmutación: 16 kHz
- Sensor de posición: Encoder, Resolver
- Control: Control vectorial, consigna de par y debilitamiento de campo.

Este inversor de Lenze es otra de las opciones seleccionadas por algunos equipos

de Formula Student que montan múltiples motores. Tiene la característica de ser un inversor dual.

2.3. AMKASYN KW26-S5-FSE-2Q

Figura 2.3: Inversor comercial AMKASYN KW26-S5-FSE-2Q y motores de la marca alemana AMKmotion GmbH. [3]

- Fabricante: AMKmotion GmbH
- Potencia máxima: 2x26 kW
- Tensión DC máxima: 720 V
- Semiconductores: Si IGBT
- Capacidad del bus DC: 1500 µF
- Frecuencia de conmutación: 8 kHz
- Sensor de posición: Encoder
- Control: Control vectorial, consigna de par y debilitamiento de campo.

AMK no solo ofrece este inversor dual si no que también pueden fabricar unos motores eléctricos ideales para su uso junto a estos inversores. Además, tienen la opción de empaquetar dos inversores duales en una sola caja creando un inversor cuádruple, idóneo para monoplazas con tracción integral. Es de lejos la opción favorita para los equipos con motores embebidos en las ruedas, ya es una solución de casi todo el tren de potencia.

2.4. CM200

Figura 2.4: Inversor comercial CM200 de la marca americana Cascadia Motion. [4]

- Fabricante: Cascadia Motion
- Potencia máxima: 225 kW
- Tensión DC máxima: 848 V
- Semiconductores: Si IGBT
- Capacidad del bus DC: 255 µF
- Frecuencia de conmutación: 16 kHz
- Sensor de posición: Encoder, Resolver
- Control: Desconocido.

Esta empresa ha colaborado históricamente en el desarrollo de inversores para los sistemas híbridos en la Fórmula 1. No hay mucha información al respecto disponible de forma abierta pero se considera un producto interesante.

3. Metodología

En este capítulo se detalla la metodología seguida para el desarrollo del proyecto. Se incluye una explicación detallada de los principios que han fundamentado el desarrollo, el listado de las etapas del proyecto, y la descripción del uso de un repositorio de Git para gestionar el proyecto.

3.1. Principios de diseño

3.1.1. Calidad sobre objetivos

En la ejecución de este proyecto, se ha adoptado una postura estratégica que privilegia la calidad y precisión en cada fase del desarrollo por encima del cumplimiento de todos los objetivos planteados inicialmente. Esta decisión se fundamenta en la conciencia de que el trabajo actual es una parte de un esfuerzo continuo, y por tanto, es imperativo establecer una base sólida y bien definida que facilite futuras continuaciones y expansiones, promoviendo la visión a largo plazo del proyecto.

El enfoque en la calidad se justifica por varios motivos clave:

- **Sostenibilidad del proyecto:** Realizar las tareas con un enfoque riguroso y meticuloso asegura que el sistema resultante sea robusto y fiable. Un trabajo bien hecho evita la acumulación de deuda técnica, lo cual es esencial para la sostenibilidad a largo plazo del proyecto.
- **Facilidad de continuación:** Dejando una documentación clara se facilita significativamente el trabajo de futuros desarrolladores. Esto es especialmente crítico en proyectos extensos donde la continuidad del desarrollo depende de la facilidad con la que nuevos desarrolladores puedan entender y construir sobre el trabajo previo.
- **Reducción de riesgos:** Priorizar la calidad y la claridad en el trabajo reduce la probabilidad de errores y fallos que podrían surgir de soluciones apresuradas o mal implementadas. Este enfoque minimiza riesgos y garantiza que cualquier problema futuro pueda ser abordado de manera efectiva.
- **Fomento de buenas prácticas:** Al enfocarse en hacer las cosas bien desde el principio, se promueve una cultura de buenas prácticas de ingeniería y desarrollo.

A continuación, se detallan las decisiones específicas que se han tomado para garantizar que el trabajo realizado no solo cumpla con los objetivos inmediatos, sino que también sea de la más alta calidad y preparado para futuras expansiones:

- **Documentación exhaustiva:** Se ha invertido tiempo en crear una documenta-

ción detallada y clara, tanto para el *hardware* (esquemáticos con notas explicativas), como para el *firmware* (documentación autogenerada a partir de comentarios muy detallados).

- **Revisión y validación minuciosa:** Cada etapa del desarrollo ha sido sometida a rigurosas revisiones y procesos de validación. Esto incluye pruebas unitarias exhaustivas, revisiones de código con personas externas al proyecto y pruebas de integración para asegurar que todos los componentes funcionen correctamente en conjunto.
- **Diseño modular:** El diseño de cada parte ha sido cuidadosamente modularizado para permitir una fácil modificación y expansión. Cada módulo ha sido diseñado para ser independiente y cohesivo, facilitando su mantenimiento y actualización sin afectar al resto del proyecto.
- **Código limpio y legible:** Se ha priorizado la legibilidad y claridad del código por encima de la optimización extrema. Esto no solo facilita el entendimiento y la modificación del código por parte de nuevos desarrolladores, sino que también reduce la probabilidad de errores y simplifica la depuración.

3.1.2. KISS

La filosofía KISS (*Keep It Simple, Stupid!*) [5] ha sido un principio fundamental que ha guiado el diseño, la implementación y la validación del inversor. La simplicidad se ha considerado como una ventaja clave en todos los aspectos del proyecto, desde la elección de componentes hasta el desarrollo del *firmware*.

Las razones por la que se cree que es necesario simplificar tanto como sea posible son las siguientes:

- **Facilidad de comprensión:** Un diseño simple es más fácil de entender, lo que facilita la colaboración, el aprendizaje y el futuro desarrollo.
- **Mayor fiabilidad:** Los sistemas simples suelen ser más fiables que los complejos, ya que tienen menos puntos de fallo.
- **Facilidad de desarrollo:** Complicar un sistema que ya de por sí es complicado aumenta mucho el tiempo que se debe invertir para diseñarlo y hacerlo funcionar.

El trabajo consta de muchas partes relacionadas entre sí y se debe facilitar esta interacción entre las diferentes etapas del proyecto. Las decisiones que se han tomado para seguir esta filosofía son los siguientes:

- **Minimizar la complejidad del diseño:** Se ha buscado simplificar el diseño del *hardware* y del *firmware* evitando soluciones excesivamente complicadas. Se han priorizado diseños limpios y directos que cumplan con los requisitos del proyecto, mitigando posibles errores derivados de la complejidad.
- **Utilizar herramientas y técnicas conocidas:** Se ha optado por utilizar herra-

mientas y técnicas ampliamente conocidas y documentadas, como por ejemplo, la estrategia de modulación SVPWM o la topología VSI.

- **Simplificar los procesos de validación:** Se han diseñado procedimientos de prueba simples, efectivos y reproducibles. Interesa que esta etapa sea ágil, y que durante las pruebas se pueda mantener el foco en los problemas relacionados estrictamente con lo que se está ensayando.

3.2. Modelo en V

El modelo en V es un enfoque de desarrollo y validación que organiza las etapas del proyecto en forma de una 'V' invertida, donde cada etapa de desarrollo tiene una contraparte de validación. Esto asegura que la validación se considere desde el principio del proyecto y que cada fase de desarrollo tenga su correspondiente prueba o verificación asociada.

Figura 3.1: Modelo en V para el desarrollo y validación de sistemas.

En el modelo en V, las etapas de diseño se encuentran en el lado izquierdo de la 'V', comenzando desde los requisitos del sistema hasta la implementación y codificación. Por otro lado, las etapas de validación se encuentran en el lado derecho de la 'V', comenzando desde las pruebas unitarias y de integración hasta las pruebas de sistema y aceptación.

El modelo en V proporciona una estructura clara y sistemática para el desarrollo y validación del convertidor, asegurando que cada etapa tenga su correspondiente prueba de validación y que los resultados sean coherentes con los objetivos del proyecto. Sin embargo, es necesario ser pragmático y eficaz con la validación, puesto

que tiende a llevar más tiempo del necesario si se sigue una metodología de forma estricta. Por ello, a lo largo de todo el proceso de diseño y de verificación se ha tomado la libertad de usar el modelo en V para aquello para lo que es útil, y ser más ágil con aquellas partes que lo requieran.

A continuación se enumeran las etapas del proyecto, aplicando pragmáticamente el modelo en V a este trabajo en específico.

1. **Definición de requisitos**

   La primera etapa del proyecto se centra en la definición de los requisitos del inversor trifásico y su control. Aquí se establecen los objetivos, las especificaciones técnicas y los criterios que guiarán todo el desarrollo. Además, se identifican las necesidades y expectativas del equipo de Formula Student, asegurando que el proyecto cumpla con sus requerimientos específicos. La duración de esta etapa es excepcionalmente larga, pues requiere de mucha familiaridad con el entorno de la Formula Student y conocimiento sobre las necesidades reales del equipo. Se ha realizado a lo largo de los años, observando los problemas asociados a un inversor comercial que un inversor personalizado podría solucionar.

2. **Modelo continuo y simulación del control**

   La siguiente etapa es el diseño del modelo en continuo y el desarrollo del control en Simulink. Aquí, se crea un modelo matemático del inversor y del motor PMSM y se implementa el control vectorial (FOC). Este proceso implica una comprensión profunda de la teoría detrás de los motores eléctricos y el diseño del control. Se basará en la representación macroscópica de la energía (EMR) con el fin de ilustrar la aplicación final del motor. La duración estimada para esta fase es de aproximadamente 2 meses.

3. **Discretización del modelo y simulación de la conmutación**

   Al acabar la etapa anterior, se trabaja en la discretización del modelo y la simulación de la conmutación de los interruptores de potencia en PLECS. Aquí se tiene en cuenta la naturaleza discreta de la electrónica de potencia y se simula el comportamiento del inversor en el dominio del tiempo discreto. Además, esta simulación incorpora también el modelo térmico, con lo que se pueden extraer las pérdidas del inversor. Esta fase dura alrededor de 2 semanas, pues gran parte del modelo continuo se puede reutilizar para el discreto, centrándose más en los aspectos de la conmutación y la implementación en un sistema discreto como es un microcontrolador.

4. **Diseño del *hardware***

   Con el diseño del control y la simulación de la conmutación como base, se procede al diseño del *hardware*. Esto implica seleccionar componentes, diseñar esquemáticos y PCBs y planificar su validación. La duración estimada para esta etapa es de 3 a 4 meses, y se solapa parcialmente con el diseño del *firmware*.

5. **Diseño del *firmware***

   Simultáneamente con el diseño del *hardware*, se trabaja en el desarrollo del *firmware*. Esto incluye programar el microcontrolador que controlará el inversor y la implementación del algoritmo de control. Se utilizará una placa de evaluación antes de tener la placa de control propia con tal de acelerar el desarrollo. La duración estimada para esta fase es de aproximadamente 4 meses y se puede solapar con otras etapas.

6. **Validación del *hardware***

   Una vez que el *hardware* se está construyendo, se procede a la validación con las pruebas planeadas anteriormente. La duración estimada para esta etapa es de aproximadamente 3 a 4 meses y se superpone con el desarrollo del *firmware*.

7. **Validación del *firmware***

   La validación del *firmware* se realiza una vez que el *hardware* está validado, y mayoritariamente, se puede realizar durante el mismo desarrollo del código. Aquí, se llevan a cabo pruebas unitarias para garantizar que cada módulo funcione según lo previsto y ver que las simulaciones y cálculos previos se ajusten a la realidad. La duración estimada para esta etapa es de 2 meses.

8. **Documentación y preparación para la implementación**

   La fase final del proyecto se enfoca en la documentación y la preparación para su implementación en los monoplazas de e-Tech Racing. La duración estimada para esta etapa es de aproximadamente un par de semanas.

Figura 3.2: Diagrama de Gantt para el proyecto

Es importante destacar que las etapas de diseño del *hardware* y del *firmware* pueden superponerse y solaparse con otras etapas, lo que permite un desarrollo más ágil y eficiente del proyecto. La superposición de estas etapas es esencial para cumplir con los plazos y garantizar que el proyecto avance de manera constante. El trabajo se ha realizado en el marco de un año académico, de septiembre a junio, aunque la definición de requisitos y los primeros pasos del modelo continuo se realizaron antes del comienzo del trabajo.

3.3. Gestión del proyecto con Git

3.3.1. Introducción a Git

Git es un sistema de control de versiones distribuido, diseñado para rastrear cambios en archivos y coordinar el trabajo entre múltiples personas en proyectos de desarrollo de *software*. Utiliza un enfoque descentralizado, lo que significa que cada desarrollador tiene una copia completa del historial de cambios del proyecto. Git facilita el trabajo colaborativo, la gestión de cambios y la integración continua en proyectos de cualquier tamaño. Existen otros sistemas de control de versiones como SVN, pero Git es el más popular.

3.3.2. Funcionamiento de Git

Git trabaja mediante la creación de instantáneas (*commits*) de los archivos en un repositorio. Cada vez que se realiza un cambio significativo en los archivos del proyecto, el desarrollador crea un nuevo *commit* que contiene una instantánea del estado de esos archivos en ese momento. Estos *commits* se organizan en ramas (*branches*), que permiten trabajar en paralelo en diferentes partes sin afectarse mutuamente. Las ramas se fusionan (*merge*) cuando el trabajo está completo y se quiere incorporar al código principal. En este proyecto solo se ha usado una rama puesto que solo hay un desarrollador.

3.3.3. Por qué utilizar un sistema de control de versiones

El uso de un sistema de control de versiones como Git proporciona numerosos beneficios para el desarrollo de proyectos de ingeniería [38], incluyendo:

- **Historial de cambios:** Permite mantener un registro detallado de todas las modificaciones realizadas en el código y otros archivos del proyecto, lo que facilita la identificación de errores y el seguimiento del progreso del desarrollo.
- **Seguridad:** Proporciona una copia de seguridad del código en caso de pérdida o corrupción de los archivos locales.

- **Experimentación:** Permite probar nuevas ideas y características con la tranquilidad de que la versión anterior está guardada.

3.3.4. Uso de Git en el proyecto

En el desarrollo de este proyecto, Git se utilizó para gestionar no solo el *firmware*, sino también las simulaciones, la documentación y el diseño del *hardware*. Cada aspecto del proyecto se organizó en una carpeta del mismo repositorio de Git, lo que permitió un seguimiento preciso de los cambios.

3.3.5. GitHub

GitHub es una plataforma de alojamiento de código que utiliza Git como sistema de control de versiones. Ofrece funciones adicionales como seguimiento de problemas, gestión de proyectos y revisión de código. En este proyecto, se utilizó GitHub para alojar el repositorio de Git.

3.3.6. Automatización de la documentación y *releases*

Una de las ventajas de utilizar Git y GitHub es la capacidad de automatizar tareas como la generación de documentación y la creación de *releases*. En este proyecto, se generó automáticamente una *wiki* utilizando los contenidos de esta memoria, lo cual facilitó la creación y el mantenimiento de documentación actualizada [27].

Además, se realizaron *releases* de *hardware* mediante el etiquetado de versiones en Git. Cada vez que se solicitaban PCBs, se creaba una nueva *release* que incluía los archivos de fabricación de las PCBs, una lista detallada de los cambios realizados desde la última versión, y se iban actualizando las *release notes* con los errores que se iban encontrando.

4. Desarrollo

4.1. Modelo matemático del PMSM

En esta sección, se abordará de manera gradual la introducción al control vectorial de máquinas eléctricas. Es importante tener en cuenta que existen muchas simplificaciones matemáticas, suposiciones no justificadas y pasos omitidos ya que la extensión de la sección sería extremadamente larga en caso de razonar desde cero todos los resultados.

4.1.1. Marco de referencia estático

Los motores síncronos de imanes permanentes (PMSM) están hechos de hierro, cobre y imanes. Se limita el análisis a máquinas de tres fases, aunque el análisis para sistemas con más de tres fases es similar. El cobre se distribuye en devanados, que tienen una resistencia y una inductancia iguales entre ellas si la máquina está equilibrada. Además, cuando el rotor gira, los imanes permanentes inducen una tensión en los devanados, lo que se conoce como fuerza contraelectromotriz (BEMF).

Figura 4.1: Motor síncrono de imanes permanentes desmontado, de manera que se pueden ver los empilados del rotor y el estátor, los imanes y el bobinado.

Se trabajará con motores cuyos devanados estén conectados en configuración estrella, ya que se utiliza con mucha más frecuencia que la configuración delta. De todas maneras, es sencillo extender el modelo y el control a motores con conexión delta.

Figura 4.2: Circuito eléctrico equivalente de un PMSM trifásico en configuración estrella.

4.1.2. Representación en el espacio vectorial

Cuando el PMSM gira, las formas de onda de corriente y voltaje son aproximadamente senoidales. Se representan las formas de onda de la corriente en la figura 4.3.

Figura 4.3: Sistema trifásico representado en el tiempo.

Todas las ecuaciones se pueden definir a partir de este modelo, pero se vuelve muy poco práctico para análisis más complejos. Se introduce una forma de representar estas formas de onda en el espacio tridimensional $\mathbb{R}^3$, en el que se escogen los ejes (a, b, c). Lo que debería esperarse ver es una forma tridimensional en el espacio $\mathbb{R}^3$. Se comienza en $t = 0$ y se dibuja un punto en las coordenadas $[i_a(0), i_b(0), i_c(0)]$. Luego, se continúa trazando puntos mientras se avanza a lo largo del eje del tiempo. La forma se puede expresar como curva paramétrica $[i_a(t), i_b(t), i_c(t)]$. Cuando se representa la forma resultante, puede sorprender, ya que es un círculo perfectamente plano. Cabe destacar que el orden de los ejes depende del contexto, y en este trabajo se toma el sistema de referencia mostrado en la figura 4.4.

Figura 4.4: Sistema trifásico representado en el Espacio Vectorial.

Se puede observar que el sistema trifásico, cuando está equilibrado, se puede representar con solo dos variables, ya que la forma resultante es bidimensional. Se explorará cómo simplificar esto para facilitar el control del PMSM.

4.1.3. Transformadas

Se puede sospechar que lo que se necesita para representar el sistema trifásico con dos variables es un cambio de base. El enfoque general sería utilizar la transformada de Concordia. La transformada de Clarke es el caso particular para 3 dimensiones de la transformada de Concordia, que se utiliza para cualquier número de dimensiones.

Se pueden encontrar dos transformadas de Clarke: la transformada de Clarke ortonormal y la transformada de Clarke de amplitud constante o módulo invariante. Además, se pueden considerar el eje α avanzado $90^{\underline{o}}$ respecto al eje β, o viceversa. Para el análisis y control de una máquina eléctrica se suele utilizar la transformada de Clarke de amplitud constante con α avanzada.

Transformada de Clarke ortonormal

Esta transformada se utiliza cuando la potencia del sistema debe permanecer inalterada después de la transformación. Se aplican dos rotaciones al marco de referencia abc y se crea el marco de referencia $\alpha\beta\gamma$. En este nuevo marco de referencia, la trayectoria del vector espacial está completamente contenida en el plano $\alpha\beta$ cuando el sistema está equilibrado, y el eje γ se utiliza para explicar el componente homopolar del sistema (la suma de los tres componentes debe ser igual a 0 si el sistema está equilibrado, de lo contrario, aparece el componente homopolar).

Figura 4.5: Sistema trifásico representado en el Espacio Vectorial con la transformada de Clarke ortonormal superpuesta.

Representar esta transformada en un eje temporal refleja de forma más intuitiva el resultado.

Figura 4.6: Transformada de Clarke ortonormal representada en el tiempo.

No se derivará la transformada aquí, aunque es necesario comprender lo que hace y conocer la matriz de transformación.

Lo que hace la transformada es colocar dos de los ejes del sistema de referencia en el plano formado por el círculo generado por el vector espacial. El eje restante es perpendicular a ese plano y representa el componente homopolar.

La forma matricial de la transformada es

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} . \tag{4.1}$$

La forma matricial de la transformada inversa es

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} . \tag{4.2}$$

Esta transformada tiene la particularidad de mantener constante la potencia del sistema, de modo que se cumple que

$$p(t) = p_{abc} = p_{\alpha\beta\gamma,\text{ortonormal}} = v_a \cdot i_a + v_b \cdot i_b + v_c \cdot i_c = v_\alpha \cdot i_\alpha + v_\beta \cdot i_\beta + v_\gamma \cdot i_\gamma \ . \quad (4.3)$$

Transformada de Clarke de amplitud constante

Mantener constante la potencia a lo largo de las transformadas puede ser útil en algunos contextos, pero lo que se suele implementar es una variante de la transformada de Clarke que mantiene constante la amplitud de la magnitud.

La transformada no es ortonormal, ya que ajusta las magnitudes para que el módulo de las variables sea el adecuado, pero la rotación se mantiene igual.

Figura 4.7: Sistema trifásico representado en el Espacio Vectorial con la transformada de Clarke de amplitud constante superpuesta.

Figura 4.8: Transformada de Clarke de amplitud constante representada en el tiempo.

De esta manera, la forma matricial de la transformada de amplitud constante es

$$\begin{bmatrix} \alpha' \\ \beta' \\ \gamma' \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \ . \quad (4.4)$$

Y la de la transformada inversa es

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \begin{bmatrix} \alpha' \\ \beta' \\ \gamma' \end{bmatrix} . \tag{4.5}$$

Se puede derivar que

$$p_{abc} = \frac{3}{2} \cdot (v_{\alpha'} \cdot i_{\alpha'} + v_{\beta'} \cdot i_{\beta'} + v_{\gamma'} \cdot i_{\gamma'}) . \tag{4.6}$$

Transformada de Park

Después de aplicar la transformada de Clarke, todavía quedan dos variables sinusoidales ($\alpha\beta$ si se considera $\gamma = 0$), lo que dificulta el análisis para que el control sea sencillo. Por lo tanto, se aplica otra transformada para convertir estas cantidades sinusoidales en constantes.

Ahora, considerando que el vector espacial gira a una velocidad de $\omega = 2\pi f$, si se aplica continuamente una rotación alrededor del eje γ con un ángulo $\theta = \omega t$, se puede representar el vector espacial como una composición de dos variables continuas (en lugar de sinusoides). Además, si se sincroniza esta rotación con el ángulo del vector espacial (el eje d apuntando al vector espacial), se obtiene una variable continua (q) y una segunda variable de valor nulo (d).

Figura 4.9: Transformada de Park representada en el tiempo junto a la transformada de Clarke de amplitud constante.

Se puede ver que la transformada es simplemente una rotación a lo largo de uno de los ejes de la base, de manera que

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} . \tag{4.7}$$

4.1.4. Marco de referencia rotatorio

En esta sección, se convertirá el modelo trifásico inicial del PMSM en un modelo continuo en el marco de referencia $d-q$. Antes, se omitieron las ecuaciones del modelo trifásico, que se pueden encontrar en [19], pero utilizando estas y aplicando las transformadas de Clarke y Park, se obtienen las ecuaciones que se derivarán en esta sección.

El enfoque general es utilizar la transformada de amplitud constante, por lo que para las tensiones y corrientes del estátor se utiliza

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \tag{4.8}$$

y

$$\begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}. \tag{4.9}$$

El modelo de circuito equivalente se divide en los circuitos del eje d y el eje q.

Figura 4.10: Circuitos eléctricos equivalentes del modelo $d-q$ de un PMSM.

Se puede observar que

$$v_d = v_{R_s} - \omega_e \cdot L_q \cdot i_q + v_{L_d} \tag{4.10}$$

$$v_d = R_s \cdot i_d - \omega_e \cdot L_q \cdot i_q + L_d \cdot \frac{di_d}{dt} \tag{4.11}$$

y

$$v_q = v_{R_s} - \omega_e \cdot L_d \cdot i_d + \omega_e \cdot \lambda_m + v_{L_q} \tag{4.12}$$

$$v_q = R_s \cdot i_q - \omega_e \cdot L_d \cdot i_d + \omega_e \cdot \lambda_m + L_q \cdot \frac{di_q}{dt} \ . \tag{4.13}$$

En estado estacionario, es decir, sin cambios muy bruscos de la corriente, el término diferencial puede ser eliminado, de manera que

$$v_d = R_s \cdot i_d - \omega_e \cdot L_q \cdot i_q \tag{4.14}$$

$$v_q = R_s \cdot i_q - \omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m \ , \tag{4.15}$$

donde

- v_d y v_q son las tensiones en los ejes d y q respectivamente.
- i_d y i_q son las corrientes en los ejes d y q respectivamente.
- L_d y L_q son las inductancias en los ejes d y q respectivamente.
- ω_e es la velocidad eléctrica, que es la velocidad mecánica multiplicada por el número de pares de polos del PMSM ($\omega_e = \omega_m \cdot pp = \omega_m \cdot \frac{n}{2}$).
- λ_m es el flujo magnético generado por los imanes permanentes. La magnitud del flujo magnético generado afecta directamente la magnitud de la tensión inducida en las fases del estátor. Este parámetro se puede transformar fácilmente en k_E, que es la relación entre la velocidad mecánica del rotor y la tensión generada en las 3 fases.

Hay PMSM cuyos imanes están montados dentro del rotor (IPM) y otros cuyos imanes están en la superficie del rotor (SPM). Esta diferenciación juega un papel importante en el desarrollo del modelo y el control, porque en los SPM se cumple $L_d = L_q$, a menudo escrito solo como L. Además, si se trata de un IPM, la orientación de los imanes puede cambiar las trayectorias del control, de manera que si son imanes tangenciales se da $L_d > L_q$ y si son imanes radiales $L_d < L_q$. Se desarrollarán las ecuaciones para motores IPM con imanes radiales, pero se ha de tener en cuenta que se pueden realizar muchas simplificaciones si el motor es un SPM. Una situación que se dará es que algunas ecuaciones tienen alguna forma de $L_d - L_q$ como denominador, lo cual es bastante problemático si se está tratando de implementar la ecuación directamente. Por ese motivo, es mejor diferenciar el tipo de motor antes de implementar las ecuaciones. Una solución válida es tener en cuenta la anisotropía magnética que suelen presentar la mayoría de SPMs, con lo que $L_d < L_q$ y se pueden aplicar las ecuaciones del IPM a costa de potencia de cálculo, de ejecutarse el control en tiempo real.

Figura 4.11: Tipos de PMSMs según la disposición de los imanes. a) SPM, b) IPM radial, c) IPM tangencial. [29]

La potencia eléctrica se define como

$$p_e = \frac{3}{2} \cdot (v_d i_d + v_q i_q) \ . \tag{4.16}$$

Si se supone que la máquina es perfectamente eficiente, se puede decir que la potencia mecánica es igual a la potencia eléctrica, $p_e = p_m$. Sabiendo que $p_m = \omega_m T_{\text{em}}$, donde T_{em} es el par electromagnético producido, se puede derivar que

$$T_{\text{em}} = \frac{3}{2 \cdot \omega_m} \cdot (v_d i_d + v_q i_q) = \frac{3}{2 \cdot \omega_e} \frac{n}{2} \cdot (v_d i_d + v_q i_q) \tag{4.17}$$

$$\therefore T_{\text{em}} = \frac{3}{2} \frac{n}{2} \cdot ((L_d - L_q) i_q i_d + \lambda_m i_q) \ . \tag{4.18}$$

También se puede establecer un límite de voltaje, porque el PMSM generalmente se controla con un inversor de fuente de voltaje (VSI) modulado por SVPWM y la tensión de salida V_s está limitada a $\frac{V_{\text{DC}}}{\sqrt{3}}$. En esta ecuación, no se considera la caída de tensión por la resistencia del estátor R_s, ya que haría que las ecuaciones fueran muy densas y el efecto de esta caída de voltaje es despreciable a altas velocidades. Por ello,

$$v_s = \sqrt{v_d^2 + v_q^2} \leq \frac{V_{\text{DC}}}{\sqrt{3}} \tag{4.19}$$

$$\therefore \sqrt{(R_s \cdot i_d - \omega_e \cdot L_q \cdot i_q)^2 + (R_s \cdot i_q - \omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m)^2} \leq \frac{V_{\text{DC}}}{\sqrt{3}} \ . \tag{4.20}$$

Despreciando los términos con R_s,

$$\sqrt{(-\omega_e \cdot L_q \cdot i_q)^2 + (-\omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m)^2} \leq \frac{V_{\text{DC}}}{\sqrt{3}} \ . \tag{4.21}$$

Y reordenando,

$$\left(\frac{\frac{V_{\text{DC}}}{\sqrt{3}}}{\omega_e} \right)^2 \geq (\lambda_m + L_d \cdot i_d)^2 + (L_q \cdot i_q)^2 \ . \tag{4.22}$$

4.1.5. Curvas características del PMSM

Las ecuaciones del PMSM son muy útiles cuando se diseña el control y la implementación real, pero antes de eso, es muy recomendable verlas en un gráfico para comprender mejor algunas de las intuiciones detrás de ellas.

Curva de par-velocidad y potencia-velocidad

Las primeras curvas estudiadas son las de par-velocidad y potencia-velocidad. Definen la intención de diseño del PMSM, ilustrando el rendimiento deseado. El objetivo al diseñar el control es tratar de igualar o incluso superar estas curvas.

Figura 4.12: Curva de par-velocidad del PMSM.

La curva es una función por tramos, que toma $T_{\text{em,máx}}$, $\omega_{\text{m,máx}}$ y $P_{\text{m,máx}}$ como sus parámetros.

La curva es constante desde $\omega_{\text{m}} = 0$ hasta $\omega_{\text{m}} = \omega_{\text{m,nom}} = \frac{P_{\text{m,máx}}}{T_{\text{em,máx}}}$, donde su valor es $T_{\text{em,máx}}$. Esta porción es lo que se conoce como la zona de par constante. Desde $\omega_{\text{m}} = \omega_{\text{m,nom}}$ hasta $\omega_{\text{m}} = \omega_{\text{m,máx}}$, T_{em} se define como $T_{\text{em}} = \frac{P_{\text{m,máx}}}{\omega_{\text{m}}}$, lo que da una curva de tipo $y = \frac{a}{x}$. Esto se llama la zona de potencia constante.

$$T_{\text{em}} = \begin{cases} T_{\text{em,máx}} & -\omega_{\text{m,nom}} < \omega_{\text{m}} < \omega_{\text{m,nom}} \\ -T_{\text{em,máx}} & -\omega_{\text{m,nom}} < \omega_{\text{m}} < \omega_{\text{m,nom}} \\ \frac{P_{\text{m,máx}}}{\omega_{\text{m}}} & \omega_{\text{m,nom}} \leq \omega_{\text{m}} \leq \omega_{\text{m,máx}}(T_{\text{em,máx}}), -\omega_{\text{m,máx}} \leq \omega_{\text{m}} \leq -\omega_{\text{m,nom}}(-T_{\text{em,máx}}) \\ -\frac{P_{\text{m,máx}}}{\omega_{\text{m}}} & \omega_{\text{m,nom}} \leq \omega_{\text{m}} \leq \omega_{\text{m,máx}}(-T_{\text{em,máx}}), -\omega_{\text{m,máx}} \leq \omega_{\text{m}} \leq -\omega_{\text{m,nom}}(T_{\text{em,máx}}) \end{cases}$$

Además P_{m} aumentará linealmente con ω_{m} hasta $\omega_{\text{m,nom}}$, ya que $P_{\text{m}} = T_{\text{em}} \cdot \omega_{\text{m}}$. Desde $\omega_{\text{m,nom}}$ hasta $\omega_{\text{m,máx}}$, P_{m} es una recta de valor $P_{\text{m,máx}}$.

Figura 4.13: Curva de potencia-velocidad del PMSM.

CLC (Círculo de Límite de Corriente)

Es obvio que la corriente eléctrica suministrada al PMSM debe estar limitada. Por lo general, el fabricante del motor establecerá la corriente alterna máxima, lo cual se traduce en un límite para i_d y i_q.

A continuación, se presenta un gráfico muy útil para conocer los límites del motor. Se establecen los ejes como i_d e i_q. Por ejemplo, si un motor está funcionando con $i_d = -1$ A e $i_q = 5$ A, se dibuja un punto en $(-1, 5)$. Como se puede apreciar, este es un sistema de coordenadas cartesianas. También puede convertirse en un sistema de coordenadas polares, que utiliza una magnitud y un ángulo. La magnitud del vector será

$$i_s = \sqrt{i_d^2 + i_q^2}\ , \tag{4.23}$$

y el ángulo,

$$\gamma = \arctan\left(\frac{i_q}{i_d}\right)\ . \tag{4.24}$$

Se puede observar que la corriente máxima puede expresarse más fácilmente como i_s, independientemente del ángulo γ (no debe confundirse con el γ de la transformada de Clarke). Por lo tanto, se puede representar $i_s = I_{s,\text{máx}}, \forall\gamma \in [0, 2\pi]$.

Figura 4.14: Círculo de límite de corriente con la representación de un punto en coordenadas cartesianas (azul) y otro en polares (rojo).

En la figura 4.14 se observa como resulta ser un círculo, lo cual tiene sentido, ya que es un vector de magnitud constante. Además, se representan los puntos $i_s\angle\gamma = I_{s,\text{máx}}\angle\frac{3\pi}{4}$ y $(-1, 5)$.

TH (Hipérbolas de Par)

Si se estudia la ecuación del par 4.18, es evidente que T_{em} es una función de (i_d, i_q). El resto de parámetros son constantes, por lo que se puede establecer un valor fijo de par y deslizar alrededor de (i_d, i_q) para generar una curva. La forma de la curva resultante es una hipérbola, que en su forma polar se expresa

$$T_{\text{em}} = \frac{3}{2}pp \cdot ((L_d - L_q) \cdot i_s^2 \cdot \sin(\gamma)\cos(\gamma) + \lambda_m \cdot i_s \cdot \sin(\gamma)) \ . \tag{4.25}$$

Figura 4.15: Hipérbolas de par.

El gráfico se limita a los cuadrantes 2 y 3 por un motivo ilustrado con estas hipérbolas: solo los valores negativos de i_d contribuyen a la generación de par en un PMSM con imanes radiales. Cuando $i_d > 0$, se necesita más corriente para generar la misma cantidad de par. Cuanto más alejada está la hipérbola del eje i_d, más par representa en valor absoluto. Aquellas hipérbolas que quedan por encima del eje i_d, es decir, $i_q > 0$ son par positivo, mientras que si $i_q < 0$, el par es de sentido opuesto.

VLE (Elipses de Límite de Voltaje)

Tomando la ecuación de voltaje 4.22, se puede demostrar que es una elipse. Del mismo modo que con las hipérbolas de par, se pueden establecer una velocidad y una tensión, y deslizar valores de (i_d, i_q) para generar la curva.

$$1 \geq \frac{\left(\frac{\lambda_m}{L_d} + i_d\right)^2}{\left(\frac{\frac{V_{\mathrm{DC}}}{\sqrt{3}}}{L_d \omega_e}\right)^2} + \frac{i_q^2}{\left(\frac{\frac{V_{\mathrm{DC}}}{\sqrt{3}}}{L_q \omega_e}\right)^2} \tag{4.26}$$

Figura 4.16: Elipses de límite de voltaje con $I_{\mathrm{sc}} > I_{s,\mathrm{máx}}$.

Al representar estas elipses, normalmente se anotan los valores de velocidad en RPM mecánicas, ya que es mucho más fácil hacerse una idea de los límites del motor junto al resto de curvas ($\omega_m[\mathrm{RPM}] = \frac{1}{pp}\omega_e \left[\frac{\mathrm{rad}}{s}\right] \cdot \frac{60}{2\pi}$).

Las elipses se reducen a medida que la velocidad aumenta. El foco de las elipses está ubicado exactamente en $(i_d, i_q) = (-I_{\mathrm{sc}}, 0) = \left(-\frac{\lambda_m}{L_d}, 0\right)$. En el gráfico anterior, el foco está fuera del círculo de límite de corriente, pero no siempre es el caso. Si $I_{\mathrm{sc}} \leq I_{s,\mathrm{máx}}$, teóricamente el motor puede alcanzar una velocidad infinita, ya que las elipses colapsan en un solo punto.

Figura 4.17: Elipses de límite de voltaje con $I_{\text{sc}} \leq I_{s,\text{máx}}$.

4.2. Control del PMSM en el espacio $d-q$

4.2.1. Trayectorias de control

Después de conocer los límites de la máquina, se pueden establecer criterios para decidir cuánto i_d y i_q (o i_s y γ) se deben aplicar al PMSM para que se comporte mecánicamente como se desee. El conjunto de puntos de trabajo que definen un comportamiento se llama trayectoria y existen una multitud de ellas. Por ejemplo, se puede desear que el motor produzca la mayor cantidad de par posible con la mínima corriente. Pero también se podría querer que tenga un cierto factor de potencia o que mantenga el par constante subiendo la velocidad, etc.

En un monoplaza de Formula Student, se desea que la salida de par esté perfectamente controlada y conocida para que el algoritmo de dinámica vehicular pueda estimar correctamente las fuerzas en los neumáticos. También es deseable que el motor pueda girar más rápido cuando no se requiere más par, ya que no es necesaria mucha tracción a altas velocidades del vehículo. Además, es necesario que sea eficiente para aprovechar mejor la energía de la batería. Con estos requisitos en mente, se estudian 4 trayectorias de control adecuadas para esta aplicación.

MTPA (Máximo Par Por Amperio)

La trayectoria de control más utilizada es el MTPA, o Máximo Par Por Amperio. Como su nombre indica, minimiza la corriente para entregar un par determinado. La condición que se debe cumplir es

$$\frac{\partial T_{\text{em}}}{\partial \gamma} = 0 \ . \tag{4.27}$$

La expresión analítica se desarrolla partiendo de la ecuación de par en forma polar 4.25,

$$T_{\text{em}} = \frac{3}{2} pp \cdot ((L_d - L_q) \cdot i_s^2 \cdot \sin(\gamma)\cos(\gamma) + \lambda_m \cdot i_s \cdot \sin(\gamma))$$

$$\frac{\partial T_{\text{em}}}{\partial \gamma} = \frac{\partial}{\partial \gamma}\frac{3}{2} pp \cdot (i_s^2 \cos(\gamma)\sin(\gamma) \cdot ((L_d - L_q) + \lambda_m i_s \sin(\gamma)) = 0 \tag{4.28}$$

$$\therefore i_{s,\text{MTPA}} = -\frac{\lambda_m \cos(\gamma)}{(2 \cdot \cos(\gamma)^2 - 1) \cdot (L_d - L_q)} \ . \tag{4.29}$$

Para la aplicación de esta trayectoria en el control se busca el ángulo como función de la corriente, y para ello se debe despejar γ_{MTPA} de la expresión.

$$\gamma_{\text{MTPA}} = \frac{\pi}{2} + \arcsin\left(\frac{\lambda_m - \sqrt{8(L_d - L_q)^2 \cdot i_s^2 + \lambda_m^2}}{4 \cdot i_s(L_d - L_q)}\right) \tag{4.30}$$

El resultado de graficar esta expresión sobre el plano (i_d, i_q) deja a la vista que el módulo de corriente es mínimo para cada hipérbola de par.

Figura 4.18: Trayectoria MTPA.

CTC (Curva de Par Constante)

Como se puede observar, las hipérbolas de par definen una trayectoria la cual permite mantener un par constante. Recordando las elipses de tensión, para un mismo valor de V_s las elipses se contraen hacia el foco a medida que la velocidad aumenta. Esto significa que siguiendo la curva de par constante de derecha a izquierda se puede mantener el par aumentando la velocidad. Usando la expresión 4.25 se puede obtener directamente

$$T_{\text{em}} = \frac{3}{2} pp \cdot ((L_d - L_q) \cdot i_s^2 \cdot \sin(\gamma)\cos(\gamma) + \lambda_m \cdot i_s \cdot \sin(\gamma))$$

$$\therefore i_{s,\text{CTC}} = \frac{\lambda_m}{L_d} \cdot \frac{\sqrt{\sin(\gamma)^2 + \frac{4 \cdot \frac{L_d - L_q}{L_d} \cdot \sin(2\gamma) \cdot T_{\text{em}} \cdot L_d}{3 \cdot pp \cdot \lambda_m^2}} - \sin(\gamma)}{\sin(2\gamma) \cdot (\frac{L_d - L_q}{L_d})} \,. \tag{4.31}$$

Figura 4.19: Trayectoria CTC.

MTPV (Máximo Par Por Voltio)

Existe una trayectoria que permite maximizar el par entregado por el motor en rangos de velocidad muy altos donde el límite es la tensión que puede sintetizar la controladora. La condición que se debe cumplir es que

$$\frac{\partial T_{\text{em}}}{\partial \delta} = 0. \tag{4.32}$$

Donde δ es el ángulo del vector de tensión V_s, de la misma manera que γ es el ángulo del vector de corriente I_s. La expresión analítica se desarrolla a partir de la expresión de par en coordenadas cartesianas (4.18), de manera que

$$T_{\text{em}} = \frac{3}{2} pp \cdot ((L_d - L_q) i_q i_d + \lambda_m i_q) \ .$$

Se aíslan i_d e i_q de 4.14 y 4.15, negligiendo la caída de tensión resistiva del estátor,

$$i_d = \frac{v_q}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d}; i_q = -\frac{v_d}{\omega_e \cdot L_q} \tag{4.33}$$

$$T_{\text{em}} = \frac{3}{2} pp \cdot \left((L_d - L_q)(-\frac{v_d}{\omega_e \cdot L_q})(\frac{v_q}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d}) + \lambda_m(-\frac{v_d}{\omega_e \cdot L_q}) \right) \tag{4.34}$$

$$T_{\text{em}} = \frac{3}{2} pp \cdot \left((L_d - L_q)(-\frac{V_s \cdot \cos(\delta)}{\omega_e \cdot L_q})(\frac{V_s \cdot \sin(\delta)}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d}) + \lambda_m(-\frac{V_s \cdot \cos(\delta)}{\omega_e \cdot L_q}) \right) \tag{4.35}$$

$$\begin{aligned}\therefore \frac{\partial T_{\text{em}}}{\partial \delta} = \frac{3}{2}pp \cdot ((L_d - L_q) \cdot (\frac{V_s \cdot \sin(\delta)}{\omega_e \cdot L_q}) \cdot (\frac{V_s \cdot \sin(\delta)}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d}) \\ - \left(\frac{V_s \cdot \cos(\delta)}{\omega_e}\right)^2 \cdot \frac{L_d - L_q}{L_d \cdot L_q} \\ - \frac{\lambda_m \cdot V_s \cdot \sin(\delta)}{L_q \cdot \omega_e}) = 0 \ .\end{aligned} \tag{4.36}$$

Definiendo la saliencia

$$\xi = \frac{L_q}{L_d} \tag{4.37}$$

y aislando, se obtiene

$$\begin{aligned}i_{s,\text{MTPV}} = \frac{\lambda_m}{L_d}(\frac{-(2-\xi)\cos(\gamma)}{2(1-\xi)(1+(\xi)^2)\cos(\gamma)^2 - 2(1-\xi)(\xi)^2} \\ - \frac{\sqrt{(2-\xi)^2\cos(\gamma)^2 - 4(1-\xi)(1+(\xi)^2)\cos(\gamma)^2 - 4(1-\xi)(\xi)^2}}{2(1-\xi)(1+(\xi)^2)\cos(\gamma)^2 - 2(1-\xi)(\xi)^2}) \ .\end{aligned} \tag{4.38}$$

Cabe destacar que esta trayectoria solamente se puede ejecutar si se cumple la condición de que $I_{\text{sc}} \leq I_{s,\text{máx}}$.

Figura 4.20: Trayectoria MTPV.

CVL (Límites de Corriente y Voltaje)

Por último, se presentan los límites eléctricos del motor y del convertidor. El límite de corriente consiste simplemente en saturar la magnitud de la corriente de manera que no sobrepase el valor máximo establecido. La trayectoria sería sencillamente seguir el círculo de corriente anteriormente presentado (CLC), con la siguiente expresión:

$$i_{s,\text{CLC}} = I_{s,\text{máx}}, \forall\gamma \in [0, 2\pi]$$

Figura 4.21: Trayectoria CLC.

El límite de tensión del motor en realidad se puede entender como la velocidad máxima a la que se puede llegar con una determinada tensión. Por ello, se usa la forma polar de la expresión de las elipses de tensión (VLE),

$$1 \geq \frac{\left(\frac{\lambda_m}{L_d} + i_s \cdot \cos(\gamma)\right)^2}{\left(\frac{\frac{V_{\text{DC}}}{\sqrt{3}}}{L_d \cdot \omega_e}\right)^2} + \frac{(i_s \cdot \sin(\gamma))^2}{\left(\frac{\frac{V_{\text{DC}}}{\sqrt{3}}}{L_q \cdot \omega_e}\right)^2}. \tag{4.39}$$

Igual que para el resto de trayectorias, se debe obtener una expresión de la elipse en función de I_s y γ. Ya que no es trivial despejar estas variables de la expresión anterior, se manipula usando la ecuación polar de la elipse desplazada del origen, de manera que

$$\rho(\theta) = \frac{b^2 x \cos(\theta) + a^2 y \sin(\theta) \pm ab\sqrt{(a^2 - x^2)\sin^2(\theta) + (b^2 - y^2)\cos^2(\theta) + 2xy\sin(\theta)\cos(\theta)}}{a^2 \sin^2(\theta) + b^2 \cos^2(\theta)}. \tag{4.40}$$

Dado que estas elipses tan solo están desplazadas en el eje x, se pueden eliminar todos los términos referentes al desplazamiento en y.

$$\rho(\theta) = \frac{b^2 x \cos(\theta) \pm ab\sqrt{(a^2 - x^2)\sin^2(\theta) + (b^2)\cos^2(\theta)}}{a^2 \sin^2(\theta) + b^2 \cos^2(\theta)} \tag{4.41}$$

Sustituyendo por los términos conocidos y simplificando,

$$i_{s,\mathrm{VLE}} = \frac{\left(\frac{1}{L_q}\right)^2 (-I_{sc})\cos(\gamma) \pm \frac{1}{L_d \cdot L_q}\sqrt{\left(\left(\frac{V_s}{L_d \cdot \omega_e}\right)^2 - (-I_{sc})^2\right)\sin^2(\gamma) + \left(\frac{V_s}{L_q \cdot \omega_e}\right)^2 \cos^2(\gamma)}}{\left(\frac{1}{L_d}\right)^2 \sin^2(\gamma) + \left(\frac{1}{L_q}\right)^2 \cos^2(\gamma)}. \tag{4.42}$$

Figura 4.22: Trayectoria VLE.

Además, el inversor es capaz de sintetizar un máximo de $V_s = \frac{V_{\mathrm{DC}}}{\sqrt{3}}$ utilizando SVPWM, por lo tanto, se debe saturar la consigna de tensión a ese valor. Adicionalmente, por seguridad, se multiplica por un factor de seguridad $K_{\mathrm{FW}} \in (0, 1)$.

$$V_{s,\mathrm{máx}} \leq \frac{V_{\mathrm{DC}}}{\sqrt{3}} \cdot K_{\mathrm{FW}} \tag{4.43}$$

4.2.2. Diseño y simulación del control

En esta sección, se aborda la implementación del modelo matemático del PMSM en un entorno de simulación. Además, se detallan los pasos cruciales en el diseño del control, destacando la implementación del lazo de control de corriente, el modelo promediado y conmutado del inversor, o la integración de las trayectorias y la estrategia de debilitamiento de campo. Disponer de un modelo de simulación completo permite ganar mucha comprensión sobre el sistema estudiado, pero por el coste computacional suele ser inviable juntar muchos sistemas en una misma simulación.

EMR (Representación macroscópica energética)

En primer lugar, se creará un modelo que permita simular la dinámica electromecánica del PMSM, así como los entornos mecánico (vehículo) y eléctrico (batería) en los que se encuentra, para posteriormente integrar el control vectorial. Para ello se usará un estándar para modelizar sistemas de potencia, la representación macroscópica energética o EMR por sus siglas en inglés [2]. El concepto se basa en agrupar o dividir las diferentes etapas en las que la potencia se transforma, utilizando el principio de acción-reacción y el principio causal (el efecto causa-efecto causa efecto porque la causa del efecto causa-efecto es a su vez causa y efecto [13]).

Figura 4.23: Modelo EMR completo.

En primer lugar se modeliza la planta eléctrica del PMSM. Se utiliza el modelo con el marco de referencia rotativo $d-q$ por su sencillez. Para ello se implementan las diferentes ecuaciones del motor en bloques separados siguiendo el estándar de la EMR.

Figura 4.24: Bloques que representan el PMSM.

También se modela la planta mecánica del motor, así como la transmisión de la potencia mecánica a las ruedas y al vehículo.

Figura 4.25: Planta mecánica.

Parámetros y curvas del PMSM simulado

A fecha de redacción de este documento no se han obtenido los parámetros del motor utilizado por el equipo. Sin embargo, se han estimado usando un pequeño *script* de MATLAB de manera que se cumplan los requisitos de potencia del motor.

Parámetros del motor			
Parámetro	Valor	Unidades	Descripción
pp	3	ad	Número de pares de polos
λ_m	52,615	mWb	Flujo magnético de los imanes permanentes
L_d	188,7	µH	Inductancia en el eje d
L_q	283,1	µH	Inductancia en el eje q
R_s	150	mΩ	Resistencia de fase del estátor
$\omega_{\text{m,máx}}$	20000	RPM	Velocidad angular máxima del motor
$T_{\text{em,máx}}$	26	N·m	Par máximo del motor
$V_{\text{DC,máx}}$	600	V	Voltaje máximo DC
$I_{\text{s,máx}}$	108	A	Corriente máxima en los ejes d-q

Cuadro 4.1: Parámetros del PMSM simulado.

Como se puede ver, el motor presenta $I_{\text{sc}} = \frac{\lambda_m}{L_d} = \frac{52{,}615\ \text{mWb}}{188{,}7\ \mu\text{H}} = 278{,}82\ \text{A} > I_{s,\text{máx}}$, y por tanto, no se aplica la trayectoria MTPV. Esto se comprueba graficando las curvas del motor, mostradas en la figura 4.26.

Figura 4.26: Gráfico de las curvas características del PMSM simulado.

Lazos de corriente y modelo promediado del inversor

Como se ha visto hasta ahora, es práctico utilizar la corriente para controlar el motor. Por ello, se implementa un lazo de corriente utilizando controladores PI para el eje d y para el eje q por separado. Como el inversor se utiliza como fuente de tensión, la salida de estos PI es la consigna de tensión. Dado que el motor genera una fuerza contraelectromotriz, se añade como *feed-forward* a los controladores. La salida del controlador no se satura directamente, sino que se implementa una saturación posterior la cual se realimenta al controlador para usar una técnica de *anti-windup* propuesta en [25]. Las constantes de los controladores se ajustan de la siguiente manera:

$$M_p = 15\,\%$$

$$t_s = T_s \cdot 20$$

Donde M_p es el sobre-impulso deseado en la respuesta a una entrada de escalón, t_s es el tiempo de establecimiento deseado, y T_s es la inversa de la frecuencia de control.

$$\xi = \sqrt{\frac{\log(M_p)^2}{\pi^2 + \log(M_p)^2}} \tag{4.44}$$

$$\omega_n = \frac{3}{\xi \cdot t_s} \tag{4.45}$$

$$Kp_{id} = 2 \cdot \xi \cdot \omega_n \cdot L_d - R_s \tag{4.46}$$

$$Ki_{id} = \omega_n^2 \cdot L_d \tag{4.47}$$

$$Kp_{iq} = 2 \cdot \xi \cdot \omega_n \cdot L_q - R_s \tag{4.48}$$

$$Ki_{iq} = \omega_n^2 \cdot L_q \tag{4.49}$$

Figura 4.27: Bloque de los lazos de corriente.

Figura 4.28: Simulación de los lazos de corriente, con una consigna de $(i_d, i_q) = (-8, 30)$ A.

Tras obtener las consignas de tensión, se modela el inversor VSI con SVPWM con un modelo promediado, es decir, sin llegar a generar una señal conmutada por PWM. Se usan relaciones básicas para convertir las magnitudes eléctricas del espacio $d-q$ a DC. Además se incorpora la fuente de energía del sistema, la batería, con un simple modelo RC.

Figura 4.29: Bloques que contienen el modelo promediado del VSI con SVPWM.

Figura 4.30: Detalle de los bloques del VSI.

Implementación de las trayectorias de control

Con lo anteriormente desarrollado solamente se pueden consignar las corrientes i_d e i_q manualmente, pero el objetivo es consignar el par y que el propio control sea capaz de gestionar el debilitamiento de campo. Por ello, se implementan las ecuaciones presentadas en el apartado anterior en bloques de código. El control se ha basado en la propuesta de [21].

La estrategia es la siguiente: Se implementan las ecuaciones de las trayectorias cuya salida es una corriente (CLC, CTC y MTPV) y se selecciona la mínima. En paralelo, se calcula el ángulo que correspondería a la trayectoria del MTPA, y se añade un control integral que aumenta el valor del ángulo controlando la tensión para poder entrar en el resto de trayectorias. Este integrador sería justamente el controlador de debilitamiento de campo y se encarga de que la consigna de ángulo no haga sobrepasar el límite de tensión establecido por el bus DC con un cierto factor de seguridad. Se trabaja con módulos de corriente siempre positivos, y ángulos comprendidos entre $\gamma \in [\frac{\pi}{2}, \pi]$. Para obtener par negativo (regeneración), simplemente se multiplica el ángulo γ por el signo de la consigna de par, atendiendo específicamente al caso de par igual a cero.

Figura 4.31: Detalle del bloque del lazo de control vectorial.

Para comprobar el funcionamiento y la estabilidad del control, se realiza una simulación en la que la consigna de par está extraída de un perfil de conducción real.

(a) Par electromagnético generado por el motor (T_{em}).

(b) Corrientes ($I_d - I_q$).

(c) Corriente (I_d) vs Corriente (I_q).

(d) Velocidad mecánica (ω_m) vs Par electromagnético (T_{em}).

(e) Velocidad mecánica (ω_m).

(f) Voltajes ($V_d - V_q$).

Figura 4.32: Resultados de la simulación.

Se puede observar que en esta simulación se ha limitado el comportamiento de la frenada regenerativa a la trayectoria del MTPA. Además, se pueden observar ciertos problemas en la implementación del lazo de tensión. Resulta que el control propuesto por [21] no considera la regeneración, ni mucho menos en debilitamiento de campo. Más adelante se realiza una simulación más enfocada en el control que en la aplicación, y en ella se revisan estas situaciones.

Modelo conmutado

Dado que el inversor realmente es una fuente conmutada, se debe modelar utilizando una herramienta que lo permita. Lo único que es necesario discretizar realmente es el control y la generación de tensiones, ya que la planta es continua. Por ello, el primer paso es recalcular las constantes de los lazos de control. Hasta ahora, se habían usado PIs continuos, descritos con la expresión

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau. \tag{4.50}$$

Cuando se discretiza esta expresión se obtiene

$$u[k] = K_p \cdot e[k] + K_i \cdot I[k], \tag{4.51}$$

donde la integral $I[k]$ se calcula con una aproximación trapezoidal como

$$I[k] = I[k-1] + \frac{(e[k] + e[k-1]) \cdot \Delta T}{2}, \tag{4.52}$$

donde ΔT es el tiempo de ejecución del PI. Para simplificar el cálculo, se introducen las constantes K_0 y K_1, que están relacionadas con K_p y K_i.

$$K_0 = K_p + K_i \cdot \frac{\Delta T}{2} \tag{4.53}$$

$$K_1 = K_i \cdot \frac{\Delta T}{2} \tag{4.54}$$

Así, la ecuación discreta del controlador PI trapezoidal se expresa como

$$u[k] = u[k-1] + K_0 \cdot e[k] + K_1 \cdot e[k-1]. \tag{4.55}$$

Utilizando el modelo EMR generado en Simulink se ha implementado la conmutación, pero el tiempo de simulación es demasiado grande como para que sea una herramienta práctica para el desarrollo.

Figura 4.33: Bloques que contienen el modelo conmutado del VSI con SVPWM.

Figura 4.34: Detalle de los bloques del VSI conmutado.

Figura 4.35: *Duty cycles* y tensiones alternas.

Por ello se desarrolla un modelo en PLECS que incorpora el lazo de control mejorado, el inversor con MOSFETs, la planta mecánica simplificada, y a la cual se le discretiza la adquisición y el control, de manera que es una aproximación muy realista de la posterior implementación en un microcontrolador. Además se incorporan cálculos térmicos de pérdidas y eficiencia.

Figura 4.36: Modelo de simulación en PLECS.

Se puede observar que la representación de los bloques en este modelo no sigue el EMR, ya que se elige una implementación más práctica y realista. Destaca la implementación de las funciones de control utilizando la librería PERGAMON, desarrollada por el CITCEA-UPC. Esta librería reproduce completamente el código de las funciones matemáticas implementadas en el control, como los PIs, el SVPWM..., en el *software* PLECS, lo que facilita la correlación de cualquier modelo con la implementación final en un sistema discreto.

A continuación se presentan los diferentes bloques que forman el modelo.

Figura 4.37: Planta electromecánica del conjunto fuente-inversor-motor-carga.

Figura 4.38: Detalle del subbloque VSI, donde se realiza la conexión electrotérmica de los módulos de potencia a la fuente DC, al motor y a la *coldplate*.

Figura 4.39: Tanto los MOSFETs como los diodos integrados están modelados térmicamente de mano del fabricante, lo que facilita mucho la estimación de pérdidas y la simulación térmica.

Figura 4.40: Se modela el ADC mediante el uso de *zero order holds* y cuantizando la adquisición y aplicando las ganancias adecuadas.

Figura 4.41: Se implementa la transformada de las corrientes, tanto de a, b, c al espacio $d-q$ como de coordenadas cartesianas a coordenadas polares.

Figura 4.42: Saturación y síntesis de las tensiones V_d y V_q mediante la transformada de Clarke inversa y la modulación SVPWM de la librería PERGAMON. Se modelan también los tiempos muertos de la modulación.

Figura 4.43: Los lazos de corriente están implementados con PIs discretos de la librería PERGAMON y afinados analíticamente con el procedimiento mostrado anteriormente.

Figura 4.44: Consigna de corriente con debilitamiento de campo.

Figura 4.45: Detalle del bloque de cálculo de corriente, donde se implementan las trayectorias de control del PMSM como ecuaciones analíticas resueltas.

En esta simulación se ha optado por una estrategia un tanto diferente a la de [21]. En vez de consignar la corriente mínima de entre todas las trayectorias, se ha implementado una selección más compleja que incorpora la trayectoria VLE y depende de la velocidad eléctrica del motor. Usando esta estrategia más elaborada se consigue un control de la frenada regenerativa más preciso, aunque todavía está en etapa de desarrollo. Hay ciertas situaciones (dependientes de la planta mecánica principalmente) en las que el lazo de tensión no es capaz de ajustar bien el ángulo de corriente y eso lleva a descontroles momentáneos.

Figura 4.46: Consigna de par y lazo de velocidad como saturación de la consigna de par.

Figura 4.47: Cálculo de pérdidas, eficiencia y factor de potencia.

Figura 4.48: Visualización y registro de las variables de interés para el análisis y el procesamiento de datos.

(a) Par electromagnético (T_{em}) y velocidad mecánica (ω_m).

(b) Corrientes ($I_d - I_q$).

(c) Corriente (I_d) vs corriente (I_q).

(d) Ángulo de corriente (γ).

(e) Temperatura y pérdidas del inversor.

(f) Voltajes ($V_d - V_q$).

Figura 4.49: Resultados de la simulación.

Como se puede observar, se ha optado por consignar un escalón de par del 90 % del par máximo hasta llegar a la velocidad máxima, donde el mismo control es capaz de limitarla. Posteriormente, se consigna un escalón de par igual pero de signo opuesto, para generar una situación extrema y verificar la robustez del control. De esta manera, se evalúa el comportamiento del control vectorial en las circunstancias más adversas, así como el rendimiento teórico del inversor a potencia máxima en los cuatro cuadrantes de operación del motor eléctrico.

Destaca la claridad con la que esta simulación permite apreciar las trayectorias de control en la figura (c), que muestra la consigna y el valor medido de i_d e i_q. Como ya se ha comentado, la simulación no refleja un perfil de conducción realista, ya que

la frenada regenerativa en debilitamiento de campo no está del todo solucionada para algunas situaciones de carga mecánica.

4.3. *Hardware*

4.3.1. Requisitos y pre-concepto

Para definir los requisitos del inversor de tracción, es crucial considerar las restricciones y exigencias del tren de potencia que impone la normativa de la competición [8], así como los parámetros eléctricos del vehículo en particular en el cual se va a implementar.

Potencia

La norma ***EV2.2.1*** [8] dicta que la potencia en la salida de la batería no debe exceder los 80 kW. Esta restricción es crucial en el diseño del inversor, ya que se prevé que el vehículo sea impulsado por un solo convertidor con esta potencia máxima. Por ende, el inversor debe estar dimensionado para manejar máximos de potencia de hasta 80 kW. Dado que se trata de un inversor doble, la asignación de potencia se divide en 40 kW por motor, aunque se podría necesitar más potencia en uno de los motores durante la aceleración en una curva. Esta decisión se toma pensando en el futuro del equipo, cuando implemente 4 motores, lo que dotaría al vehículo de hasta 160 kW de máximo. Sin embargo, en ese escenario, solo se podrían utilizar 80 kW repartidos entre las 4 ruedas, con un máximo de 40 kW por rueda.

La prueba donde se espera utilizar esta potencia máxima es la *Acceleration*, donde se podría requerir de los 80 kW durante un máximo de 10 segundos, siendo esta una aproximación muy conservadora. Por lo tanto, la potencia máxima queda como

$$P_{\text{out, máx, tot}} = 80\ \text{kW}(2 \cdot 40\ \text{kW})(10\ \text{s máx.})$$

Por otro lado, el requisito de potencia media es distinto. Sería un error dimensionar la potencia media del inversor como el valor máximo, ya que esto conduciría a un sobredimensionamiento excesivo de la refrigeración, los conductores, los conectores y los dispositivos de potencia. En cambio, se opta por determinar el valor medio de potencia requerido durante la prueba más exigente, la *Endurance*. En esta prueba, el vehículo debe recorrer aproximadamente 22 km, lo que equivale a alrededor de media hora de uso continuo. En la *Endurance*, considerar la frenada regenerativa es esencial, ya que aumenta el valor medio de la potencia total, sumándose a la potencia de tracción. Se ha calculado el valor medio de potencia a partir de una simulación de la *Endurance* que llega a máximos de 80 kW.

Figura 4.50: Potencia instantánea que sale de la batería en un evento de *Endurance* (Tiempo completo y detalle de un fragmento).

A continuación se muestra el cálculo del valor RMS de la potencia requerida, basado en los datos de potencia:

```
% Calcular el cuadrado de los valores de potencia
power_squared = power_time.Data.^2;

% Calcular el promedio de los valores cuadrados
mean_power_squared = mean(power_squared);

% Tomar la raíz cuadrada del promedio de los valores cuadrados para obtener
% el valor RMS
power_rms = sqrt(mean_power_squared);

power_rms =

2.3193e+04
```

Esto demuestra que el valor medio de potencia requerido es considerablemente menor que el pico máximo, en concreto un $\frac{23{,}2\ \mathrm{kW}}{80\ \mathrm{kW}} = 29\,\%$. Por lo tanto, se dimensiona el inversor considerando una potencia media de 35 kW, es decir, 17,5 kW por motor. Esta cifra proporciona un margen de seguridad de 11,8 kW respecto a la simulación, lo que permite algo de juego en el reparto de potencia entre ambos motores. De esta manera, la potencia constante queda como

$$P_{\text{out, const, tot}} = 35\ \mathrm{kW}(2 \cdot 17{,}5\ \mathrm{kW})$$

Cabe destacar que se está calculando la potencia aparente del convertidor, ya que la potencia activa y la reactiva se reparten en función del factor de potencia, que no es constante y depende de la planta mecánica y la situación del motor. La potencia activa es la que usa el motor para acelerar o frenar, y la reactiva la consume o suministra el bus de condensadores. Otra nota es que la simulación es poco realista puesto que utiliza picos de potencia máxima, cuando en realidad, en la prueba de la *Endurance* se suele limitar el valor de estos picos con tal de extender la autonomía.

Además, existe la norma ***EV4.1.1*** *The maximum permitted voltage that may occur between any two electrical connections is 600 V,DC and for motor controller/inverters internal low power control signals 630 V,DC.*, que impone el límite de 600 V para la tensión de bus máxima. Es de interés que el inversor esté dimensionado a esta tensión de funcionamiento, **600 V,DC**, para permitir máxima flexibilidad con el diseño de la batería, por ejemplo. Al usar la tensión más alta posible, se obtiene un beneficio en la reducción de la corriente, que implica menos pérdidas en los conductores y permite reducir la sección de cables, pletinas, conectores y otros conductores.

Ya que la estrategia de modulación es SVPWM y la tensión máxima es de 600 V, la tensión alterna fase-neutro de un motor se expresa como

$$V_{\text{fase-neutro, pico, 600 V,DC}} = \frac{V_{\text{DC}}}{\sqrt{3}} = \frac{600\text{ V}}{\sqrt{3}} = 346\text{ V}$$

$$V_{\text{fase-neutro, RMS, 600 V,DC}} = \frac{V_{\text{fase-neutro, pico}}}{\sqrt{2}} = \frac{346\text{ V}}{\sqrt{2}} = 245\text{ V}$$

Sin embargo, la batería no estará constantemente a 600 V,DC, por lo que para poder entregar la potencia de 40 kW pico por inversor en un rango de tensiones adecuado se debería calcular la corriente con una tensión menor. Se escoge 450 V,DC como tensión mínima a partir de la cual se puede entregar la potencia máxima de 40 kW por inversor. Entonces, la corriente de fase de un motor queda

$$V_{\text{fase-neutro, pico, 450 V,DC}} = \frac{V_{\text{DC}}}{\sqrt{3}} = \frac{450\text{ V}}{\sqrt{3}} = 261\text{ V}$$

$$V_{\text{fase-neutro, RMS, 450 V,DC}} = \frac{V_{\text{fase-neutro, pico}}}{\sqrt{2}} = \frac{261\text{ V}}{\sqrt{2}} = 184\text{ V}$$

$$I_{\text{fase, RMS, máx}} = \frac{P_{\text{out, máx}}}{3 \cdot V_{\text{fase-neutro, RMS, 450 V,DC}}} = \frac{40\text{ kW}}{3 \cdot 184\text{ V}} = 72\text{ A}$$

$$I_{\text{fase, pico, máx}} = I_{\text{fase, RMS, máx}} \cdot \sqrt{2} = 72\text{A} \cdot \sqrt{2} = 102\text{ A}$$

De la misma manera que con la potencia, se puede obtener el valor de corriente constante a una tensión menor, ya que el resultado será más restrictivo.

$$I_{\text{fase, RMS, const}} = \frac{P_{\text{out, const}}}{3 \cdot V_{\text{fase-neutro, RMS, 450 V,DC}}} = \frac{17{,}5\text{ kW}}{3 \cdot 184\text{ V}} = 32\text{ A}$$

$$I_{\text{fase, pico, const}} = I_{\text{fase, RMS, const}} \cdot \sqrt{2} = 32\text{ A} \cdot \sqrt{2} = 45\text{ A}$$

Iterando el cálculo de potencia, se puede ver que si se dimensiona a estas corrientes, las potencias máxima y constante calculadas a 600 V,DC son considerablemente mayores.

$$P_{\text{out, máx}} = 3 \cdot V_{\text{fase-neutro, RMS, 600 V,DC}} \cdot I_{\text{fase, RMS, máx}} = 3 \cdot 245\text{ V} \cdot 72\text{ A} = 53\text{ kW}$$

$$P_{\text{out, const}} = 3 \cdot V_{\text{fase-neutro, RMS, 600 V,DC}} \cdot I_{\text{fase, RMS, const}} = 3 \cdot 245\text{ V} \cdot 32\text{ A} = 23{,}5\text{ kW}$$

$$P_{\text{out, máx, tot}} = 2 \cdot P_{\text{out, máx}} = 2 \cdot 53\text{ kW} = 106\text{ kW}$$

$$P_{\text{out, const, tot}} = 2 \cdot P_{\text{out, const}} = 2 \cdot 23{,}5\text{ kW} = 47\text{ kW}$$

Velocidad

Puesto que se va a controlar un motor con posibilidad de operación en debilitamiento de campo, es necesario conocer la frecuencia eléctrica máxima que se va a tener que poder sintetizar. Este requisito va a dimensionar la velocidad de los lazos de control, y en última instancia, la frecuencia de conmutación. El motor en estudio tiene 3 pares de polos y una velocidad angular mecánica máxima de 20000 RPM. Por lo tanto, la frecuencia eléctrica máxima que se debe sintetizar se calcula

$f_{\text{e,máx}} = \frac{pp \cdot \omega_{\text{m, máx, RPM}}}{60} = \frac{3 \cdot 20000}{60} = 1000 \text{ Hz}$.

Dejando un poco de margen para la compatibilidad con motores de más polos, se escoge una frecuencia máxima de 1200 Hz. Para sintetizar esta frecuencia eléctrica con poca distorsión armónica, se debe escoger una frecuencia de conmutación al menos 30 veces superior, para tener como mínimo 30 pulsos por periodo a estas frecuencias.

$$f_{\text{conm}} > 30 \cdot f_{\text{e,máx}} = 30 \cdot 1200 \text{ Hz} = 36 \text{ kHz} \rightarrow f_{\text{conm}} = 40 \text{ kHz}$$

Se debe considerar también que con una frecuencia de conmutación mayor, las pérdidas de conmutación serán más grandes, pero el tamaño del bus de condensadores será más pequeño. Además, el lazo de control debe ir a 40 kHz, con lo que el microcontrolador que lo implemente debe ser capaz de ejecutar todos los cálculos en menos de $\frac{1}{40 \text{ kHz}} = 25$ µs.

Es importante que el control sea mucho más rápido que la planta eléctrica, cuya constante de tiempo es de

$\min(\frac{L_d}{R_s}, \frac{L_q}{R_s}) = \min(\frac{188{,}7 \text{ µH}}{0{,}15 \ \Omega}, \frac{283{,}1 \text{ µH}}{0{,}15 \ \Omega}) = \min(1{,}258 \text{ ms}, 1{,}887 \text{ ms}) = 1{,}258 \text{ ms}$

$1{,}258 \text{ ms} >> 25 \text{ µs}$.

Normativa

La normativa de la Formula Student no impone normativas muy restrictivas, ya que la mayoría están enfocadas a la seguridad. Estas son las normas que afectarán al diseño del inversor:

- ***EV2.2.2*** *Regenerating energy is allowed and unrestricted.* La posibilidad de regenerar energía implica que el inversor debe ser capaz de consignar par en el sentido opuesto a la velocidad de los motores para frenar, lo que tendrá un impacto fundamental en el diseño del control. La topología de VSI es bidireccional en sí misma.

- ***EV2.2.3*** *Wheels must not be spun in reverse.* Podría impactar la lógica de control del inversor y en la adquisición de los sensores de posición para evitar la marcha atrás. Se pueden implementar algoritmos de dependencia de dirección para que ambos motores traccionen el vehículo en la misma dirección.

- ***EV3.1.1*** *TS enclosures, see EV1.1.2, must consist of either*

  - *a grounded solid layer made of at least 0.5 mm thick electrically conductive material, aluminium or better, having a resistance below 300 mΩ, measured with a current of 1 A, to LVS ground and able to continuously carry at least 10 % of the TS accumulator main fuse current rating or*

  - *be fully made out of electrically insulating materials having an isolation resistance of at least 2 MΩ, measured with a voltage of 500 V. The enclosure must be rigid and must prevent possible mechanical penetrations. Protruding electrically conductive parts, such as fasteners or connectors, must follow EV3.1.2*

  The TSAC might use at least 0.9 mm thick steel layer as the grounded layer.

  Tendrá efectos en el diseño del empaquetado del convertidor y en la selección de materiales para garantizar la seguridad eléctrica y mecánica del convertidor.

- ***EV3.2.5*** *All overcurrent protection devices that are part of the TS must not rely on programmable logic. The overcurrent protection function of motor controllers/inverters for the motor outputs may rely on programmable logic.* Los dispositivos de protección contra sobrecorriente en el sistema de tracción para el motor pueden depender de lógica programable, lo que permite utilizar el sensado de corriente para garantizar esta protección.

- ***EV4.1.2*** *All components in the TS must be rated for the maximum TS voltage. The TS area of a PCB, see EV4.3.6, is considered as one component. Every input connected to the TS must be rated to the maximum TS voltage.* Tendrá efectos en la selección de los componentes del circuito de potencia del inversor.

- ***EV4.1.3*** *All components must be rated for the maximum possible temperature that may occur during usage.* Impactará en la selección de componentes que puedan manejar las temperaturas esperadas.

- ***EV4.2.1*** *TS enclosures, see EV1.1.2, must be labeled with (a) reasonably sized sticker(s) according to "ISO 7010-W012" (triangle with a black lightning bolt on a yellow background). The sticker must also contain the text "High Voltage" if the voltage is more than 60 V,DC or 50 VACRMS.* Impacta en los requisitos de señalización y seguridad del empaquetado.

- ***EV4.3.1*** *The entire TS and LVS must be galvanically isolated, see EV1.2.1 and IN4.1.1.* Afecta el diseño de la separación y el aislamiento entre ambos sistemas.

- ***EV4.3.4*** *Where both TS and LVS are present within an enclosure, they must be separated by barriers made of moisture-resistant insulating materials or maintain the following spacing through the air, or over a surface:*

Cuadro 4.2: Voltage Spacing Requirements

Voltage	Spacing
$U < 100$V,DC	10 mm
100V,DC $< U <$ 200V,DC	20 mm
$U > 200$V,DC	30 mm

Impacta en el diseño del empaquetado del inversor. Se usarán aislantes como el Nomex en caso de que no se pueda garantizar el espaciado por aire.

- ***EV4.3.6*** *If TS and LVS are on the same PCB, they must be on separate well-defined areas of the board, meeting the spacing requirements in table 5, each area clearly marked with "TS" or "LV". The outline of the area required for spacing must be marked. "Conformal coating" refers to a coating insulator, solder resist is not a coating.*

Cuadro 4.3: Voltage Spacing Requirements

Voltage	Over Surface	Through Air (Cut in board)	Conformal Coating
0 to 50	1.6 mm	1.6 mm	1.0 mm
50 to 150	6.4 mm	3.2 mm	2.0 mm
150 to 300	9.5 mm	6.4 mm	3.0 mm
300 to 600	12.7 mm	9.5 mm	4.0 mm

Tendrá efectos en el diseño de la PCB de potencia.

- ***EV4.5.3*** *The temperature rating for TS wiring, connections, and insulation must be appropriate for the expected surrounding temperatures but at least 85* °C. Influirá en la selección de cables y materiales aislantes.

- ***EV4.5.4*** *TS components and containers must be protected from moisture in the form of rain or puddles, see IN9.* Impactará en el diseño de la caja.

- ***EV4.5.6*** *All TS wiring must be completed to professional standards with appropriately sized conductors and terminals and with adequate strain relief and protection from loosening due to vibration etc.* Tendrá efectos en el diseño de las conexiones y terminales.

- ***EV4.5.10*** *Every TS connector outside of a housing must include a pilot contact/interlock line which is part of the SDC. Housings only used to avoid interlocks are prohibited.* Impactará en el diseño de los conectores y sistemas de interconexión.

- ***EV4.5.11*** *All TS connections must be designed so that they use intentional current paths through conductors such as copper or aluminium and must not rely on steel bolts to be the primary conductor.* Tendrá efectos en el diseño de las conexiones y la selección de materiales adecuados.

- ***EV4.5.12*** *All TS connections must not include compressible material such as*

plastic in the stack-up or as a fastener. FR-4 is allowed. Impactará en el diseño de las conexiones y la selección de materiales adecuados.

- ***EV4.5.13*** *TS connectors outside of TS enclosures must be designed in a way, that the TS cannot be activated, see EV4.11, if connected in any way other than the design intent configuration.* Tendrá efectos en el diseño de los conectores y la seguridad.

- ***EV4.5.14*** *All electrical connections, including bolts, nuts, and other fasteners, in the high current path of the TS must be secured from unintentional loosening. Fasteners must use positive locking mechanisms, see T10.2, that are suitable for high temperatures, see EV4.5.3.* Impactará en el diseño de los sistemas de fijación y bloqueo, especialmente en las conexiones de potencia.

- ***EV4.5.16*** *Soldered connections in the high current path are only allowed if all of the following are true:*

  - *connections on PCBs*
  - *the connected devices are not cells or wires*
  - *the devices are additionally mechanically secured against loosening*

  Tendrá efectos en el diseño de las conexiones y el uso de soldaduras. El uso de sistemas alternativos de conexión en placa como el *press-fit*, facilita el cumplimiento de esta norma.

- ***EV4.9.1*** *If a discharge circuit is required to meet EV6.1.5, it must be designed to handle the maximum TS voltage permanently. After three subsequent discharges within 15 s in total, the discharge time specified in EV6.1.5 may be exceeded. Full discharging functionality must be given after a reasonable time with a deactivated discharge circuit.* Se deberá integrar un circuito de descarga en el inversor que cumpla con estos requisitos.

- ***EV4.9.2*** *The discharge circuit must be wired in a way that it is always active whenever the SDC is open. Furthermore, the discharge circuit must be fail-safe such that it still discharges the intermediate circuit capacitors if the HVD has been opened or the TS accumulator is disconnected.* Tendrá efectos en la implementación del circuito de descarga.

- ***EV2.2.1*** *EV4.10.2 The TSAL itself must have a red light, flashing continuously with a frequency between 2 Hz and 5 Hz and a duty cycle of 50 %, active if and only if the LVS is active and the voltage across any DC-link capacitor exceeds*

  - *60 V,DC or 50 VACRMS*
  - *Half the nominal TS voltage*

  whichever is lower

  Se deberá implementar esta detección.

- ***EV4.10.4*** *The mentioned voltage detection must be performed inside the respective TS enclosure.* Obliga a situar la detección de alta tensión en el propio bus de condensadores.

Estas restricciones no solo definen los límites de diseño del inversor, sino que también influyen en la selección de componentes y la estrategia de control.

Interfaz de comunicación

Para que el vehículo pueda controlar los motores mediante consignas de par, el convertidor debe incorporar un protocolo adecuado para ello. El escogido es el **bus CAN**, puesto que es un estándar en la industria de la automoción y todas las otras unidades electrónicas del monoplaza integran este protocolo para intercomunicarse. Realmente, lo único estrictamente necesario es recibir las consignas de par y una señal para habilitar o deshabilitar el funcionamiento. Además, el inversor deberá enviar todos los datos relevantes para su posterior análisis.

Resumen de requisitos de *hardware*

Con toda la información que se ha recopilado se pueden listar los requisitos más importantes del convertidor.

- **Topología del convertidor:** VSI doble.
- **Potencia pico:** 80 kW ($2 \cdot 40$ kW).
- **Potencia constante:** 35 kW ($2 \cdot 17{,}5$ kW).
- **Tensión DC máxima:** 600 V.
- **Corriente de fase máxima:** 80 A,RMS.
- **Aislamiento galvánico entre TS y LVS**.
- **Detección de tensión**.
- **Circuito de descarga integrado**.
- **Interfaz por comunicación CAN**.

Boceto del empaquetado

Antes de entrar en el diseño detallado del convertidor, se realizaron algunos bocetos preliminares para visualizar la disposición de los componentes y planificar la distribución espacial. Estos bocetos iniciales sirven como punto de partida para el diseño final del empaquetado del inversor.

Figura 4.51: Bocetos preliminares del empaquetado del inversor.

En estos dos primeros conceptos se explora el uso de una sola *coldplate* para refrigerar ambos semiconductores. Adicionalmente se propone refrigerar el bus de condensadores, aunque posteriormente se encuentra que no es necesario. Se ha optado por una configuración que separa claramente los módulos de potencia de la placa de control, facilitando así el mantenimiento y la disipación de calor.

Figura 4.52: Evolución de los bocetos del empaquetado del inversor.

En los bocetos posteriores, se ha refinado la disposición de los componentes para mejorar la accesibilidad y la eficiencia de refrigeración. La disposición de los

conectores y la distribución de los conductores también se ha ajustado para optimizar la densidad de potencia y la eficiencia del diseño. Estos cambios reflejan un enfoque hacia un empaquetado más eficiente y funcional del inversor. De todas maneras, todavía presentan muchas complicaciones en el ensamblaje y fabricación de algunos elementos como la *coldplate*.

Estos bocetos iniciales proporcionan una idea visual de cómo se podría organizar el empaquetado del inversor. A medida que avanza el diseño, se realizarán ajustes a la vez que se encuentren limitaciones con los componentes específicos seleccionados y se hagan consideraciones eléctricas y térmicas que en este punto no se han tenido.

4.3.2. Topología y concepto

Con el fin de producir las tensiones trifásicas que calcula el control para los dos motores se implementa un ondulador trifásico fuente de voltaje doble (*dual VSI*). Poniendo el foco en uno solo de los convertidores, existen varias topologías que permiten hacer un ondulador trifásico, pero la más utilizada es el VSI de 2 niveles, formado por tres ramas de medio puente. Otras topologías de más niveles logran sintetizar las tensiones con menos distorsión armónica, pero dado que el motor eléctrico es una carga inductiva, y que el control está basado en consignas de corriente, la distorsión armónica del convertidor tiene un impacto muy poco significativo en el rendimiento global. De esta manera, se justifica la implementación del VSI dual.

Figura 4.53: VSI con MOSFETs.

Ya que el convertidor implementa código propio, se decide implementar dos VSI en el mismo convertidor, de modo que se opta por unificar algunos elementos de ambos como la refrigeración con el fin de optimizar el peso y el espacio. Esta decisión evita duplicados de medidas, componentes y código, sin embargo, conlleva otros retos, como por ejemplo, que el microcontrolador deberá ser capaz de ejecutar los dos lazos de control al mismo tiempo.

Para adquirir las variables necesarias para el control y la interfaz con el vehículo y sus periféricos, se requieren los submódulos que se presentan en el diagrama de bloques mostrado en la figura 4.54.

Figura 4.54: Diagrama de bloques del sistema completo.

El convertidor se divide en 3 PCBs, *Inverter Power*, que se instala por duplicado (una por motor), e *Inverter Control*. Como su nombre indica, *Inverter Power* alojará todos los componentes de potencia, incluyendo los *gate drivers*, los sensores de corriente, los semiconductores, el bus de condensadores y los conectores de potencia. Además, en esta placa se puede encontrar el sensado de la tensión de los buses, los circuitos de descarga, y la detección de alta tensión para la *TSAL*. El sensado de tensión sería el único subcircuito repetido, pero esto solamente aporta redundancia y tolerancia al fallo.

Por su parte, *Inverter Control* contiene el microcontrolador de la familia STM32F7 para consignar la conmutación a los *gate drivers*, así como para realizar las lecturas de corriente, tensión y temperatura, las interfaces con los sensores de posición de los motores o los protocolos comunicación CAN y USB.

El montaje de estas placas se realiza de forma compacta y eficiente, de manera que cabe una *coldplate* entre medias de las dos de potencia para refrigerar los semiconductores de ambos inversores. Se encajan de manera que la placa de control se puede conectar a las placas de potencia mediante conectores placa a placa. Las conexiones de potencia facilitan la integración con el cableado del monoplaza debido a su posicionamiento.

4.3.3. Semiconductores

Tecnología de semiconductores

La decisión de emplear MOSFETs de carburo de silicio (SiC) como interruptores se fundamenta en consideraciones específicas de la aplicación y en los requisitos de la competición. En un entorno donde la reducción de peso y volumen es crucial, el SiC emerge como una opción destacada frente a alternativas como el IGBT de silicio, el FET de GaN o el MOSFET de silicio. Aunque el precio no es una limitación primordial en este caso, el proyecto ha atraído el interés de empresas que han ofrecido muestras de semiconductores de forma gratuita para el desarrollo del convertidor.

Las ventajas inherentes del SiC, como su rendimiento superior o su resistencia a temperaturas más altas, permiten un diseño más compacto y robusto del inversor de tracción. Estas características son esenciales para cumplir con los requisitos de un monoplaza de competición, donde lograr altas densidades de potencia es muy beneficioso para la integración del resto de componentes del vehículo y aligeramiento del mismo.

Módulos de potencia

En el diseño del inversor de tracción, se optó por módulos *half-bridge* debido a su idoneidad para el rango de potencias y tensiones del convertidor. Dos modelos de semiconductores se consideraron para su integración: el DFS05HF12EYR1 de Leapers Semiconductor y el CAB016M12FM3 de Wolfspeed.

Ambos modelos cumplen estrictamente con los requisitos de la aplicación, con un voltaje de ruptura ($V_{\mathrm{DS,breakdown}}$) de 1200 V y una corriente máxima ($I_{\mathrm{DS,máx}}$) que excede los 80 A,RMS. La elección de dos modelos distintos se basa en la intención de realizar pruebas comparativas para verificar las diferencias entre ambos.

El DFS05HF12EYR1 ofrece especificaciones muy buenas en su datasheet, aunque Leapers Semiconductors no lleva muchos años en la industria y no han logrado crear la confianza que otras empresas han conseguido con su experiencia. Por otro lado, el CAB016M12FM3, de la reconocida marca Wolfspeed (antiguamente CREE), aporta la confiabilidad asociada a una empresa con amplia experiencia en el campo.

Según sus respectivos datasheets, ambos modelos permiten alcanzar sin mucho esfuerzo una frecuencia de conmutación de 40 kHz, lo que contribuye significativamente a la reducción del tamaño del bus de continua y optimiza el empaquetado del inversor. La placa de potencia se diseñará para permitir la prueba de ambos modelos, ya que comparten *footprint*, facilitando la adaptabilidad y la evaluación comparativa.

Parámetro	**DFS05HF12EYR1**	**CAB016M12FM3**
$V_{\text{DS,breakdown}}$ [V]	1200	1200
R_{on} [mΩ]	5.5 - 13	16.0 - 28.8
$V_{\text{f,D}}$ [V]	3.3 - 4	4.9 - 5.5
T_{rr} [ns]	41.5 - 45	20.0
Q_{rr} [µC]	2.19 - 3.94	1.30
$R_{\text{th,jc}}$ [K/W]	0.12 - 0.15	0.543
Q_{G} [nC]	520	236
C_{in} [nF]	14.5	6.6
$R_{\text{G,int}}$ [Ω]	1.9	2.4
$V_{\text{GS,th}}$ [V]	2.8 - 4.8	1.8 - 3.6
$I_{\text{DS,máx}}$ [A]	150	89

Cuadro 4.4: Comparación de parámetros entre DFS05HF12EYR1 y CAB016M12FM3.

Análisis de pérdidas

Los dispositivos semiconductores experimentan pérdidas de energía que influyen significativamente en su eficiencia y desempeño, y se transforman en calor, limitando así la potencia de salida. Estas pérdidas pueden clasificarse en dos categorías fundamentales: las pérdidas de conducción y las pérdidas de conmutación. Las **pérdidas de conducción** surgen cuando el dispositivo se encuentra en estado activo, conduciendo corriente a través de él. Esto genera una caída de voltaje y una disipación de potencia asociada a la resistencia interna del dispositivo. Por otro lado, las **pérdidas de conmutación** se manifiestan durante los ciclos de activación y desactivación del dispositivo, cuando la energía almacenada en la capacitancia del mismo se disipa durante la transición entre los estados de conducción y corte.

En el análisis de la eficiencia de los semiconductores, es imperativo considerar las pérdidas totales, las cuales se definen como la suma de las pérdidas de conducción y las de conmutación:

$$P_{\text{tot}} = P_{\text{cond}} + P_{\text{conm}} \tag{4.56}$$

A continuación, se examinarán detalladamente las pérdidas de conducción, seguidas por un análisis exhaustivo de las pérdidas de conmutación en los módulos seleccionados.

Pérdidas de conducción

Las pérdidas de conducción tienen su origen en la resistencia entre drenador y fuente cuando el MOSFET está en estado de conducción, o en la caída de tensión

del diodo cuando es el diodo quien está conduciendo. Según la ley de Ohm

$$P_{\text{cond,MOSFET}} = I_{\text{DS}}^2 \cdot R_{\text{DS, on}}, \tag{4.57}$$

y la definición de potencia

$$P_{\text{cond},D} = I_{\text{SD}} \cdot V_{\text{f}}. \tag{4.58}$$

Sin embargo, estas expresiones se corresponden con las pérdidas instantáneas y no son útiles a la hora de dimensionar el convertidor. Para ello son necesarias las pérdidas promediadas, pero tienen una expresión analítica muy compleja, ya que dependen de la estrategia de modulación, el índice de modulación instantáneo, el factor de potencia instantáneo, etc. Para el inversor se utiliza SVPWM, pero el cálculo de pérdidas tiene demasiados parámetros instantáneos. Muchas referencias asemejan las pérdidas en SVPWM a las de una modulación PWM sinusoidal (SPWM), con lo que las pérdidas se simplifican [10].

$$P_{\text{cond,MOSFET}} \approx I_{\text{RMS}} V_{\text{Q0}} \left(\frac{1}{\pi\sqrt{2}} + \frac{m \cdot cos(\phi)}{2\sqrt{8}} \right) + I_{\text{RMS}}^2 R_{\text{Q}} \left(\frac{1}{4} + \frac{2 \cdot m}{3\pi \cdot cos(\phi)} \right) \tag{4.59}$$

$$P_{\text{cond,D}} \approx I_{\text{RMS}} V_{\text{D0}} \left(\frac{1}{\pi\sqrt{2}} - \frac{m \cdot cos(\phi)}{2\sqrt{8}} \right) + I_{\text{RMS}}^2 R_{\text{D}} \left(\frac{1}{4} - \frac{2 \cdot m}{3\pi \cdot cos(\phi)} \right) \tag{4.60}$$

Como se puede apreciar, aunque son más compactas que otras en la bibliografía, estas expresiones son difíciles de abordar, puesto que casi todos los parámetros son instantáneos. Por ello, se ha usado PLECS para estimar las pérdidas de conducción debido a que facilita la simulación detallada de los semiconductores y permite obtener las pérdidas de forma diseccionada por cada dispositivo y tipo de pérdida. PLECS utiliza modelos detallados de los dispositivos, considerando sus características de conmutación y conducción reales.

Con los parámetros de estos modelos, PLECS calcula las pérdidas de conmutación y conducción de manera precisa. Sin embargo, como no tiene en cuenta el circuito completo de los *gate drivers*, no se pueden tomar directamente las pérdidas de conmutación. Se usa un modelo de MOSFET y diodo de Wolfspeed proporcionado directamente por el fabricante. Este modelo cuenta con una parametrización de las pérdidas y los efectos térmicos mucho más detallada que la que se puede obtener a partir de la hoja de datos. El fabricante utiliza tablas de búsqueda en función de la tensión, corriente y temperatura, e incorpora una relación matemática con las resistencias de puerta. Lamentablemente, Leapers no ofrece esta parametrización con sus semiconductores, por lo que el análisis en PLECS se realiza únicamente con el modelo de Wolfspeed, siendo este el más restrictivo en cuanto a las pérdidas de conducción.

Para sacarle el máximo partido, se ha obtenido un perfil de pérdidas a máxima potencia para el inversor en diferentes zonas de control del PMSM, que se puede observar a continuación.

Figura 4.55: Descomposición de las pérdidas de los semiconductores de una simulación en PLECS.

Se puede ver como las pérdidas de conducción son más significativas que las de conmutación, en particular, las de los MOSFETs. Las zonas de par constante (de 1,05 s a 1,4 s por ejemplo), muestran unas pérdidas casi constantes con la corriente, pero crecen debido a la temperatura. Al entrar en la zona de límite de tensión (de 1,7 s a 1,8 s), se observa un pico que se corresponde con la zona de potencia constante, que en esta simulación es la potencia máxima. Para obtener un valor constante de pérdidas con el que diseñar la refrigeración, se parte de este valor de pico (450 W aproximadamente) y se trata de la misma manera que la relación de potencia media respecto a la potencia pico calculada en la sección de requisitos.

$$\frac{P_{\text{RMS}}}{P_{\text{pico}}} = \frac{35\ \text{kW}}{80\ \text{kW}} = 0{,}4375 \approx 0{,}5$$

Con este razonamiento, se da un valor de pérdidas de conducción máximas de un inversor de $450\text{W} \cdot 0{,}5 = 225$ W. Como hay dos inversores anclados a la misma *coldplate*, las pérdidas de conducción que debe disipar es de $225\ \text{W} \cdot 2 = 450$ W. Es importante recordar que estas pérdidas son para el semiconductor de Wolfspeed. Dado que las pérdidas de conducción son proporcionales a $R_{\text{DS, on}}$, se pueden aproximar las pérdidas de conducción del semiconductor de Leapers.

$$P_{\text{cond,Wolfspeed}} = 450\ \text{W}$$

$$P_{\text{cond,Leapers}} \approx P_{\text{cond,Wolfspeed}} \cdot \frac{R_{\text{DS, on, Leapers}}}{R_{\text{DS, on, Wolfspeed}}} = 450\ \text{W} \cdot \frac{5{,}5\ \text{m}\Omega}{16\ \text{m}\Omega} \approx 155\ \text{W}$$

Pérdidas de conmutación

El cálculo de las pérdidas de conmutación es más fácil de determinar de forma analítica, aunque depende del circuito de *gate driver* y de sus parásitos. Si no se tienen en cuenta estos dos factores, se pueden calcular utilizando los valores

de E_{on} y E_{off}, que representan la energía necesaria para encender o apagar el dispositivo respectivamente. Cuando las hojas de datos de los semiconductores proporcionan los valores E_{on} y E_{off}, las pérdidas de conmutación pueden calcularse de forma directa. Estas pérdidas están influenciadas por la tensión de conmutación, la corriente de conmutación, la temperatura de la unión y la resistencia de puerta. La fórmula comúnmente utilizada para calcular las pérdidas es

$$E_{\text{conm}}(t) = (E_{\text{on}} + E_{\text{off}}) \left(\frac{I_{\text{Q,env}}(t)}{I_{\text{test}}} \right)^{K_i} \left(\frac{V_{\text{DC}}}{V_{\text{test}}} \right)^{K_v}, \tag{4.61}$$

donde $I_{\text{Q,env}}(t)$ es la corriente de conmutación en cada instante de tiempo, I_{test} y V_{test} son los valores de prueba, y K_i y K_v son coeficientes de ajuste que normalmente son igual a 1. La justificación se puede encontrar en los gráficos proporcionados en las hojas de datos de ambos semiconductores, donde la relación de las energías de encendido y apagado con la corriente es aproximadamente lineal.

Figure 10. E_{on}, E_{off}, E_{rr} vs I_{DS}
T_j =25°C, R_G =3.3Ω, V_{GS} =+18V/0V

(a) Leapers

Figure 11. Switching Energy vs. Drain Current (V_{DD} = 600 V)

(b) Wolfspeed

Figura 4.56: Energías de encendido y apagado en función de la corriente para Leapers [15] y Wolfspeed [34].

Si la corriente es continua,

$$P_{\text{conm}} \approx (E_{\text{on}} + E_{\text{off}}) f_{\text{conm}} \frac{I_{\text{out}}}{I_{\text{test}}} \frac{V_{\text{DC}}}{V_{\text{test}}}. \tag{4.62}$$

Y si la corriente es una sinusoide semi-rectificada con un valor de pico de $\sqrt{2} I_{\text{RMS}}$ (como en el caso del SVPWM),

$$P_{\text{conm}} \approx (E_{\text{on}} + E_{\text{off}}) f_{\text{conm}} \frac{\sqrt{2} I_{\text{RMS}}}{\pi I_{\text{test}}} \frac{V_{\text{DC}}}{V_{\text{test}}}. \tag{4.63}$$

Para obtener los valores de E_{on} y E_{off} se pueden usar los gráficos que aparecen en las hojas de datos, ya que principalmente dependen de la resistencia de puerta que se escoja.

(a) Leapers

(b) Wolfspeed

Figura 4.57: Energías de encendido y apagado en función de la resistencia de puerta externa para Leapers [15] y Wolfspeed [34].

Se escogerán los valores más grandes de resistencia de puerta para hacer este cálculo, pues un valor más grande produce menos sobrepico de tensión en la conmutación, pero aumenta las pérdidas. Sustituyendo por los valores del convertidor en diseño,

- Generales: $f_{\text{conm}} = 40$ kHz, $I_{\text{RMS}} = 80$ A, $V_{\text{DC}} = 600$ V
- Leapers: $E_{\text{on}} = 5{,}8$ mJ, $E_{\text{off}} = 3{,}7$ mJ, $V_{\text{test}} = 600$ V, $I_{\text{test}} = 150$ A
- Wolfspeed: $E_{\text{on}} = 1{,}8$ mJ, $E_{\text{off}} = 0{,}9$ mJ, $V_{\text{test}} = 600$ V, $I_{\text{test}} = 80$ A

Sustituyendo para los dispositivos Leapers,

$$P_{\text{conm, Leapers}} \approx (5{,}8 \text{ mJ} + 3{,}7 \text{ mJ}) \cdot 40 \text{ kHz} \cdot \frac{80 \text{ A}}{150 \text{ A}} \cdot \frac{600 \text{ V}}{600 \text{ V}} \approx 91{,}23 \text{ W},$$

y para los dispositivos Wolfspeed,

$$P_{\text{conm, Wolfspeed}} \approx (1{,}8 \text{ mJ} + 0{,}9 \text{ mJ}) \cdot 40 \text{ kHz} \cdot \frac{80 \text{ A}}{80 \text{ A}} \cdot \frac{600 \text{ V}}{600 \text{ V}} \approx 48{,}61 \text{ W}.$$

Dado que estas pérdidas son para un solo MOSFET, se deben multiplicar por 12 para obtener las pérdidas de conmutación totales, dado que hay 6 dispositivos de potencia en la topología VSI y hay dos VSIs en el convertidor.

$$P_{\text{conm, Leapers, tot}} = 12 \cdot P_{\text{conm, Leapers}} \approx 1095 \text{ W}$$

$$P_{\text{conm, Wolfspeed, tot}} = 12 \cdot P_{\text{conm, Wolfspeed}} \approx 583 \text{ W}$$

Resumen de pérdidas

El siguiente gráfico presenta una comparación detallada de las pérdidas de conducción y conmutación para los semiconductores de Leapers y Wolfspeed.

Figura 4.58: Comparación de pérdidas de conducción y conmutación.

Fabricante	**Pérdidas de conducción (W)**	**Pérdidas de conmutación (W)**	**Pérdidas totales (W)**
Leapers	155	1095	1250
Wolfspeed	450	583	1033

Cuadro 4.5: Comparación de pérdidas entre semiconductores.

Se observa que en el caso de Leapers, las pérdidas de conducción exhiben una significativa disminución en comparación con las de Wolfspeed, mientras que las pérdidas de conmutación muestran un incremento dramático para Leapers en relación con Wolfspeed. Este fenómeno sugiere que, en líneas generales, los dispositivos desarrollados por Leapers podrían ofrecer un camino de corriente más favorable en el semiconductor, y los dispositivos de Wolfspeed son capaces de conmutar más rápido debido a que su carga Q_G es mucho menor, reduciendo así E_{on} y E_{off}. Para el uso de los semiconductores de Leapers se recomienda bajar la frecuencia de conmutación, prestando atención a las afectaciones que pudiera tener en el bus de continua y en el control. El desarrollo práctico del trabajo se enfocará en los

semiconductores de Wolfspeed por parecer una opción más equilibrada para la aplicación.

La suma total de las pérdidas de ambos inversores, independientemente del semiconductor, es como máximo de 1300 W, valor con el cual se puede diseñar el sistema de refrigeración.

4.3.4. Sistema de refrigeración

A continuación se presenta una comparación entre las opciones comunes para refrigerar el inversor:

Opción de refrigeración	**Ventajas**	**Desventajas**
Convección natural	Económica, sin partes móviles	Limitada disipación de calor, depende del ambiente
Disipador de calor	Económico, fácil de instalar, no requiere energía adicional	Limitado en aplicaciones de alta potencia, depende del ambiente
Ventilador	Buena disipación de calor, control de temperatura	Limitado en aplicaciones de alta potencia, ruido
Coldplate de agua	Excelente capacidad de disipación, buen control de temperatura	Coste, mantenimiento, posible riesgo de fugas, instalación complicada
Coldplate de aceite	Buena capacidad de disipación, no corrosivo, alto rango de temperaturas	coste, mantenimiento, posible riesgo de fugas, instalación complicada

Cuadro 4.6: Comparación de opciones de refrigeración.

Para el sistema de refrigeración de los semiconductores se ha optado por un sistema de refrigeración líquida consistente en una *coldplate*, una bomba y un radiador. Esta elección se basa en varios factores clave que se detallan a continuación:

- **Mantenimiento de la temperatura:** A diferencia de una refrigeración por aire, una *coldplate* permite mantener una interfaz de temperatura fija para los semiconductores. Esto es crucial para garantizar un funcionamiento estable y confiable de los componentes, especialmente en aplicaciones de alta potencia como la que se está diseñando.

- **Experiencia previa:** El equipo tiene experiencia previa en sistemas de refrigeración líquida, lo que facilita la implementación y mantenimiento de este tipo de sistemas.

- **Flexibilidad del *packaging*:** La refrigeración líquida, especialmente con agua, permite un diseño más compacto en comparación con otras opciones. Un disipador

haría mucho más voluminoso el convertidor, mientras que la refrigeración líquida permite una distribución más libre de los componentes por el monoplaza. Por ejemplo, se puede situar el radiador en una zona donde tenga mucho contacto con el aire exterior, y conectarlo mediante mangueras a la bomba y a la *coldplate*.

Aunque el carburo de silicio tiene una temperatura máxima de la unión notablemente alta que el silicio, se ha establecido un objetivo de mantener la interfaz térmica del semiconductor a 80 °C como máximo. Este enfoque busca garantizar una operación óptima y una vida útil prolongada de los semiconductores, además de mitigar los posibles efectos negativos derivados del calor.

Se ha elegido agua como el fluido de refrigeración por varias razones:

- **Norma T7.2.2:** Según la norma ***T7.2.2*** [8], los sistemas de refrigeración solo pueden utilizar agua, aire u aceite como refrigerante. Con lo cual, no es posible utilizar líquidos refrigerantes específicos.
- **Propiedades térmicas:** El agua tiene una alta capacidad calorífica y conductividad térmica, lo que la hace eficaz para disipar el calor generado por los semiconductores. El aceite tiene peor conductividad térmica, y su amplio rango de temperaturas no es útil para la aplicación.
- **Seguridad:** El agua es un fluido seguro y no inflamable, lo que reduce los riesgos de seguridad en caso de fugas o accidentes. Además, es un fluido fácilmente disponible y de bajo costo en comparación con otros refrigerantes. Si se usa agua destilada, el riesgo de corrosión es más bajo, y los sistemas electrónicos no serán tan susceptibles a posibles fugas.

En el presente trabajo no se desarrolla más acerca el sistema de refrigeración puesto que la complejidad de diseñarlo excede el alcance del proyecto. Sin embargo, se reconoce su importancia, ya que al no disponer de este no se pueden realizar pruebas de potencia constante.

4.3.5. *Gate drivers*

Los gate drivers son componentes críticos en un convertidor de potencia, puesto que son los encargados de conectar las señales débiles de un microcontrolador con los semiconductores. Además, suelen llevar funcionalidades adicionales como la detección de cortocircuitos en el semiconductor, la lectura de señales analógicas de forma aislada, o mecanismos de protección avanzados.

Funcionamiento genérico

Los *gate drivers* desempeñan un papel esencial en la conmutación eficiente de los MOSFETs, proporcionando los niveles de tensión y corriente adecuados para activar y desactivar rápidamente los transistores. En esencia, es una etapa transistorizada que .ªmplifica”la fuerza de la señal del microcontrolador, y a menudo lo

hace de forma aislada.

Criterios de selección

Al seleccionar un *gate driver*, varios factores deben tenerse en cuenta para garantizar un rendimiento óptimo del sistema:

- **Topología**: A pesar de que hay soluciones que incluyen 6 *drivers* en un solo encapsulado, las más comunes son los circuitos integrados diseñados para controlar un solo dispositivo o dos en configuración de medio puente. Sin embargo, encontrar esta última opción para el rango de tensiones de 600 V con características óptimas es difícil. Por lo tanto, la topología más adecuada suele ser la de un solo *driver* por encapsulado.
- **Tensión de aislamiento**: Dado que las señales de puerta vienen del sistema de baja tensión, deben aislarse antes de llegar al *gate*, con lo que será necesario buscar un componente aislado a una tensión adecuada de al menos tres veces superior a la tensión máxima del inversor, 600 V.
- **Corriente de salida**: Es crucial elegir un *gate driver* que pueda suministrar la corriente necesaria para cargar y descargar rápidamente las capacidades de compuerta de los MOSFETs. La corriente necesaria depende de la resistencia de puerta, pero se puede determinar una primera cota de $\frac{V_{\mathrm{GS,typ}}}{R_{\mathrm{G,int}}} = \frac{20\ \mathrm{V}}{2{,}4\ \Omega} = 8{,}3\ \mathrm{A}$
- **Protecciones integradas**: Funciones como la lectura de señales analógicas, la posibilidad de paralelizar salidas o la inclusión de protecciones son muy beneficiosas para la integración del convertidor.

Comparativa de alternativas

A continuación se presenta una comparativa entre varios modelos de *gate drivers*, considerando sus características principales:

Características	**ADUM4146**	**UCC21710**	**GD3160**
Voltaje de Operación	-15 V hasta 30 V	–17,5 V hasta 36 V	-12 V hasta 25V
Corriente de Salida	4,61 A	10 A	15 A
Protecciones	Desaturación, UVLO	Sobrecorriente, Desaturación, *Miller Clamp*	Desaturación, Sobrecorriente, Apagado suave
Tiempo de Retardo	75 ns	90 ns	-
Aislamiento	5 kV	5,7 kV	8 kV
Extras	-	Sensor analógico aislado	Sensor analógico aislado
CMTI	100 V/ns	150 V/ns	100 V/ns

Cuadro 4.7: Comparación de *gate drivers*.

Considerando los criterios de selección, se observa que el UCC21710 se presenta como una opción intermedia adecuada. En primer lugar, la corriente de salida que ofrece es de hasta 10 A, lo que lo hace adecuado para cargar y descargar rápidamente las capacidades de compuerta de los MOSFETs, cumpliendo así con uno de los requisitos fundamentales. Además, integra protecciones contra sobrecorriente y desaturación, así como la funcionalidad de *Miller Clamp*. Además, la inclusión de un sensor analógico aislado proporciona la capacidad de monitorear la temperatura de los semiconductores con sus sensores integrados sin ningún componente adicional.

Aunque las protecciones de sobrecorriente y desaturación son elementos fundamentales para garantizar la seguridad y el correcto funcionamiento de un inversor, en este caso se ha optado por no implementarlas debido a las complicaciones que pueden surgir en el diseño del *layout*. Estas protecciones requieren la inclusión de componentes adicionales con *footprints* grandes y un enrutamiento más complejo de las señales, lo cual puede resultar en problemas de interferencia y aumento de la complejidad del circuito.

Es importante destacar que este inversor se encuentra en una fase inicial de prototipado y desarrollo, donde se prioriza la funcionalidad básica y la viabilidad del diseño. En futuras revisiones y etapas de desarrollo, se considerará la inclusión de estas protecciones, ya que son cruciales para proteger tanto el inversor como los dispositivos conectados. La decisión actual de omitir estas protecciones se basa en la necesidad de simplificar el diseño y garantizar una implementación más eficiente y manejable en esta fase del proyecto.

Figura 4.59: Diagrama de bloques funcional del *gate driver* seleccionado [31].

Cálculos del *gate driver* seleccionado

- **Tensión *gate-source***: Uno de los valores más importantes a determinar es el nivel de tensión de puerta para encender y apagar el MOSFET. En una aplicación *hard-switched* como la del convertidor en diseño, es beneficioso usar una tensión de encendido tan alta como sea posible.

  Dada la naturaleza rápida del carburo de silicio, se prevé una dV/dt considerable, la cual puede causar varios problemas. Uno de ellos sería la activación accidental de un MOSFET, la cual causaría la condición de *shoot-through*. Una estrategia para mitigar parcialmente este problema es el uso de un *gate driver* con *Miller Clamp*. Además, se suele apagar el dispositivo utilizando una tensión negativa, que amortigua posibles interferencias en la puerta del dispositivo afectado. El valor de tensión negativo al cual se encuentre la puerta aumenta la tensión que debe inducirse en ese nodo para llegar a la tensión umbral $V_{\mathrm{G,treshold}}$ del MOSFET. Dado que el valor óptimo no es trivial, conviene implementar alguna manera de modificarlo, por ejemplo, mediante el uso de un regulador lineal.

  Los dispositivos seleccionados tienen un rango de tensiones V_{GS} un tanto distintos, siendo el de Leapers de +21 V a -2 V, y el de Wolfspeed de +15 V a -4 V.

- **Potencia de alimentación del *gate driver*:** Se estima en función de la frecuencia de conmutación y la carga de la compuerta de los MOSFETs, además de la tensión V_{GS}.

$$I_{\text{supply, min}} = f_{\text{conm}} \cdot Q_{\mathrm{G}} \tag{4.64}$$

  Donde
  - $I_{\text{supply, min}}$ es al corriente mínima que debe poder suministrar la fuente de alimentación del *gate driver*,
  - f_{conm} es la frecuencia de conmutación, y
  - Q_{G} es la carga en la puerta.

  Utilizando el valor más alto de Q_{G} entre los dos semiconductores,

$$I_{\text{supply, min}} = f_{\text{conm}} \cdot Q_{\mathrm{G}} = 40\ \text{kHz} \cdot 520\ \text{nC} = 20{,}8\ \text{mA}$$

  se puede calcular la potencia necesaria de la fuente.

$$P_{\text{supply, min}} = \Delta V_{\mathrm{GS}} \cdot I_{\text{supply, min}} \tag{4.65}$$

  Donde:
  - $P_{\text{supply, min}}$ es la potencia mínima que debe poder suministrar la fuente de alimentación del *gate driver* y
  - ΔV_{GS} es la diferencia entre la tensión de puerta de encendido y apagado.

$$P_{\text{supply, min}} = \Delta V_{\text{GS}} \cdot I_{\text{supply, min}} = 20\ \text{V} \cdot 20{,}8\ \text{mA} = 0{,}416\ \text{W}$$

Con esta estimación de potencia mínima requerida, es esencial seleccionar una fuente adecuada para los *gate drivers*. En este contexto, se considera la serie MGJ2 de Murata, que ofrece características como:

- Tensión de entrada de 5 V, 12 V, 15 V o 24 V, lo que permite adaptarse a diferentes fuentes de alimentación disponibles. En esta aplicación, la entrada de 5 V simplifica mucho la arquitectura de *hardware*.
- Rango de tensiones de salida muy variados, que permite intercambiar fuentes para probar distintos niveles de tensión.
- Aislamiento reforzado hasta 5,2 kV,DC, garantizando la seguridad y el cumplimiento de la normativa.
- Caracterización de CMTI mayor a 200 V/ns, lo que garantiza una respuesta rápida y efectiva ante transitorios de alta velocidad.

- **Valores de resistencias de puerta ($R_{\mathbf{G,on}}$, $R_{\mathbf{G,off}}$):** Reducir el valor de las resistencias de puerta conlleva la disminución de las pérdidas de conmutación, ya que los MOSFETs cambiarán más rápido de estado y, por lo tanto, pasarán menos tiempo en la etapa de conmutación. Esta rapidez en el cambio implica un mayor dV/dt, lo que puede ser responsable del aumento de la interferencia electromagnética (EMI) y de sobrepico de tensión entre el *drain* y el *source* del propio MOSFET. El valor de estas resistencias se estimará mejor con pruebas empíricas, pero se parte del valor más alto considerado de 12 Ω para Wolfspeed, y 10 Ω para Leapers.

4.3.6. Bus de condensadores

La tarea del bus de condensadores es equilibrar la potencia instantánea fluctuante en los nodos positivo y negativo de la batería, resultante de los semiconductores que conmutan esta tensión continua en la carga. Estos condensadores, a menudo referidos como bus o enlace de continua, mitigan el rizado creado por la conmutación de alta frecuencia, lo que garantiza un suministro de tensión más estable para el sistema eléctrico en su conjunto.

Función del bus de condensadores

En un inversor de fuente de voltaje (VSI), el bus de condensadores tiene dos funciones principales:

- **Minimizar el rizado de tensión:** El rizado de tensión en el bus de continua está causado por la conmutación de los dispositivos semiconductores. El bus de

condensadores ayuda a reducir este rizado de tensión, lo que es crucial para evitar fluctuaciones en la tensión máxima que se puede consignar al motor.

El rizado de tensión se calcula como:

$$V_{\text{ripple}} = \frac{I_{\text{C,RMS}}}{C \cdot f_{\text{conm}}} \ , \tag{4.66}$$

donde:
- V_{ripple} es el rizado de tensión producido,
- $I_{\text{C,RMS}}$ es la corriente alterna del bus de condensadores,
- C es la capacitancia del bus de condensadores, y
- f_{conm} es la frecuencia de conmutación.

- **Proporcionar la potencia reactiva:** Además de la reducción del rizado de tensión, los condensadores en el bus también proporcionan potencia reactiva necesaria para compensar la carga inductiva del motor.

Dimensionamiento del bus de condensadores

Para dimensionar adecuadamente el bus de condensadores en un inversor de fuente de voltaje, se deben considerar las siguientes condiciones:

- $V_C > 1{,}1 \cdot V_{\text{max}}$

  Donde:
  - V_C es el voltaje del bus de condensadores,
  - V_{max} es la tensión máxima DC.

- $I_{C,\text{RMS}} \approx 0{,}65 \cdot I_{\text{fase,RMS}}$

  Donde:
  - $I_{C,\text{RMS}}$ es la corriente efectiva del bus de condensadores,
  - $I_{\text{fase,RMS}}$ es la corriente efectiva de fase,
  - El factor 0.65 se utiliza para estimar la corriente RMS máxima del bus de condensadores en el peor caso, que con una modulación SVPWM se da cuando el índice de modulación es de aproximadamente 60 % [30].

- $C > \frac{I_{C,\text{RMS}}}{V_{\text{ripple}} \cdot f_{\text{conm}}}$

  Donde:
  - C es la capacitancia del bus de condensadores,
  - V_{ripple} es el rizado de tensión permitido,
  - f_{conm} es la frecuencia de conmutación.

Es fácil ver como reducir la frecuencia de conmutación aumentará proporcionalmente el rizado de tensión para la misma corriente de salida.

Ahora, aplicando estas consideraciones a los valores del diseño:

- $V_C > 1{,}1 \cdot V_{\text{max}} = 1{,}1 \cdot 600\ \text{V} = 660\ \text{V} \rightarrow 850\ \text{V}$
  Escoger condensadores con un límite de tensión más grande es más seguro para transitorios y el régimen de trabajo en debilitamiento de campo.

- $I_{C,\text{RMS}} \approx 0{,}65 \cdot I_{\text{fase,RMS}} = 0{,}65 \cdot 80\ \text{A,RMS} = 52\ \text{A,RMS}$
  Se debe asegurar que el condensador puede soportar esta corriente, idealmente sin refrigeración. En caso de usar múltiples condensadores para formar el bus, la corriente se reparte de forma proporcional entre cada uno de ellos.

- $C > \frac{I_{C,\text{RMS}}}{V_{\text{ripple}} \cdot f_{\text{conm}}} = \frac{52\ \text{A,RMS}}{15\ \text{V} \cdot 40\ \text{kHz}} \approx 87\ \mu\text{F} \rightarrow 100\ \mu\text{F}$
  Conviene sobredimensionar la capacitancia tanto como sea posible atendiendo a límites mecánicos y de ensamblaje. Para el valor de V_{ripple} se suele considerar un 5 % de la tensión mínima de funcionamiento, que en este caso es de entorno a los 350 V ($\frac{15\ \text{V}}{350\ \text{V}} = 4{,}3\ \ \%$). Además, se puede jugar con el margen proporcionado para probar frecuencias de conmutación más bajas, dado que los inversores comerciales analizados muestran relaciones de capacitancia - frecuencia mucho más altas.

Selección de condensadores

Se ha optado por un diseño con 10 condensadores del modelo FHA85Y106KS de la serie FH de Murata. A continuación se detallan las razones para esta elección:

- **Material:** La serie FH de Murata tiene condensadores de película, cuya resistencia interna es muy inferior a la de un condensador electrolítico, lo cual es crucial para minimizar las pérdidas en el condensador.

- **Densidad de capacitancia:** Esta serie de condensadores presenta una alta densidad de capacitancia de entorno a 0,6 $\frac{\mu\text{F}}{\text{cm}^3}$. Esto va a permitir compactificar el diseño bastante.

- **Tensión máxima:** Con una gama de 500 V y otra 850 V, estos últimos están dentro del rango necesario para operar de manera segura, incluso en debilitamiento de campo, donde en caso de error, la tensión de bus podría ser superior a 600 V.

- **Temperatura de operación:** Tienen un rango de temperatura de operación amplio, que va desde -40 °C hasta +125 °C. Dado que se prefiere evitar su refrigeración, el hecho de que aguanten altas temperaturas permite tener más confiabilidad en la corriente que pueden soportar y el calentamiento que esta va a provocar.

- **Frecuencia de operación:** Los condensadores FHA85Y106KS están diseñados para operar en frecuencias de hasta 200 kHz, cuatro veces superior a la frecuencia de conmutación escogida.

Figura 4.60: Apariencia del condensador seleccionado [23].

Se evalúa el comportamiento térmico utilizando la hoja de datos:

Figura 4.61: Calentamiento propio del condensador en función de la corriente [23].

Dado que hay 10 condensadores, la corriente máxima de cada uno es de $\frac{52 \text{ A,RMS}}{10} =$ 5,2 A,RMS. Por tanto, el incremento de temperatura es de 10 °C como máximo. No será necesaria ninguna consideración térmica ni de refrigeración para estos condensadores.

4.3.7. Conectores de potencia

Para asegurar una conexión segura del bus de continua con la batería y de los semiconductores con las fases de los motores, se requiere una cuidadosa selección

de conectores. Dada la alta corriente que deben soportar, las conexiones soldadas sin soportes adicionales no son adecuadas según la norma ***EV4.5.16*** [8]. Por ello, se opta por la tecnología de conectores *press-fit*.

Un modelo que destaca en esta aplicación es el 7461063 de Würth Elektronik, diseñado para conexiones de hasta 160 A. Está fabricado en latón con una superficie estañada, y su montaje *press-fit* asegura una conexión mecánica y eléctrica sólida.

La tecnología *press-fit* utilizada en estos conectores REDCUBE de Würth Elektronik proporciona una serie de ventajas significativas. Al presionar los pines en la PCB, se genera una soldadura fría homogénea entre el pin y el orificio metalizado, asegurando una resistencia de contacto inferior a 200 µΩ.

Figura 4.62: Conector REDCUBE PRESS-FIT - Modelo 7461063 [36].

Estos conectores permiten una flexibilidad de integración muy grande, puesto que la conexión a ellos se realiza mediante una unión roscada, un tornillo o perno, lo cual permite conectar cables con terminales de anilla, pletinas y otros elementos conductores. Gracias a esto es muy sencillo implementar mecanismos de bloqueo positivo para evitar la desconexión por vibraciones.

Figura 4.63: Ejemplo de conexión con bloqueo positivo utilizando una tuerca DIN 980V.

4.3.8. Sensor de posición

A fecha de redacción, el sensor de posición específico a utilizar en el inversor aún no ha sido determinado. Sin embargo, se diseñará el *hardware* para permitir el uso del *encoder* incremental, ya que ha sido utilizado hasta ahora en el proyecto.

El *encoder* incremental es un dispositivo que genera pulsos eléctricos a medida que un actuador magnético gira. Este actuador se sitúa en el eje del cual se desea conocer la posición.

Figura 4.64: Actuador magnético situado en el eje de un motor.

Este tipo de sensor generalmente consta de tres canales: A, B y Z. Los canales A y B son señales cuadradas desfasadas 90 grados, mientras que el canal Z proporciona una referencia de posición conocida como pulso de índice. El *encoder* incremental

genera un número determinado de pulsos por cada rotación del ángulo medido, lo que se conoce como pulsos por revolución o PPR.

Figura 4.65: Señales de un *encoder* incremental de 4 PPR junto al ángulo real (θ) girando en sentido antihorario.

Para estimar la velocidad, dirección y posición del motor a partir del *encoder* incremental, se utilizan los pulsos de los canales A y B. La dirección del movimiento se determina mirando si A está avanzada respecto a B (sentido antihorario) o viceversa (sentido horario). La velocidad se calcula midiendo el intervalo entre pulsos y la posición se determina contando (integrando) el número de pulsos desde una posición de referencia conocida.

Sin embargo, este tipo de sensor, al igual que otros, presenta un desafío: hasta que no se obtiene la señal de índice, la posición es desconocida. Una solución efectiva podría ser implementar un lazo abierto de tensión hasta obtener el pulso que marca el origen, ya que no requiere una lectura de posición previa ya que el ángulo se impone externamente. En el contexto de la tracción eléctrica, no es viable girar el motor libremente al iniciar el convertidor, ya que esto implicaría un desplazamiento involuntario del vehículo. El lazo abierto de tensión sería una solución adecuada para la aplicación específica de la Formula Student, dado que los motores están acoplados a las ruedas mediante una reductora de velocidad, con una relación de transmisión típica de alrededor de 10. Esto implica que, en el peor escenario, el motor deberá dar una vuelta mecánica completa antes de obtener un pulso de índice, lo que resulta en que la rueda gire aproximadamente una décima parte de una revolución mediante el control de lazo abierto. Tras obtener el pulso de índice, se puede operar en lazo cerrado sin mucha dificultad.

Otra posibilidad es inicializar la variable de la posición con un valor arbitrario

hasta obtener el pulso de índice, aunque se desconoce si este método podría causar problemas más allá de la falta de rendimiento durante este periodo inicial.

El *hardware* necesario para leer la señal del *encoder* incremental puede consistir únicamente en el uso de tres comparadores analógicos. Estos comparadores se encargan de convertir las señales comúnmente diferenciales de los canales A, B y Z en señales digitales que pueden ser procesadas por el MCU del inversor. Esta interfaz también sería compatible con sensores de posición Hall, lo que brinda flexibilidad en la elección del sensor de posición para el inversor.

4.3.9. Microcontrolador

La unidad de procesamiento del convertidor es una de las piezas clave, pues será la encargada de gestionar lecturas analógicas, alarmas, comunicaciones, y por supuesto, ejecutar el control. Por ello su selección debe ser meticulosamente estudiada. En primer lugar se seleccionan unos requisitos:

- **Velocidad de reloj mayor a 200 MHz**: En base a la experiencia del autor con otros inversores, el cálculo de todos los PIs, filtros, transformadas y trayectorias de control puede tomar hasta 2500 ciclos de reloj. Por lo tanto, dado que hay que ejecutar dos lazos de control para los dos motores, $\frac{2500 \cdot 2}{200\ \text{MHz}} = 25\ \mu\text{s} \leftrightarrow 40\ \text{kHz}$, 200 MHz es la mínima velocidad para poder conmutar a 40 kHz.

- **Unidad de punto flotante**: Para evitar trabajar en coma fija, lo cual puede causar muchos problemas de integración y añade una capa de complejidad innecesaria, se pone como requisito que el MCU soporte el uso de *float* de forma nativa.

- **Dos *timers* avanzados**: Son necesarios dos *timers*, uno para cada inversor, con características como la salida PWM bipolar con varios canales, configuración sencilla del tiempo muerto, etc.

- **Lecturas analógicas**: Se necesitan al menos un par de ADCs de 12 bits suficientemente rápidos con múltiples canales cada uno, con tal de poder capturar las corrientes de fase y las tensiones de bus adecuadamente.

- **Interfaces de comunicación CAN, SPI, I^{2}C y USB**: Dado que se requerirá la interacción con otros dispositivos, es esencial contar con varias interfaces de comunicación. Obviamente CAN para la comunicación con el vehículo, pero también SPI/I^{2}C para utilizar memorias externas, o UART/USB como manera de conectar el inversor de forma sencilla a un ordenador.

- **Capacidades de depuración avanzadas**: Se requerirá de herramientas de depuración que permitan detener el código en puntos específicos, monitorizar en tiempo real el estado de las variables, etc.

- **Memoria *flash* superior a 500 kB**: El binario de un programa muy extenso no debería pesar mucho más de 100 kB, sin embargo, si en el futuro se programa

un *bootloader* u otras implementaciones como tablas de búsqueda grandes, la memoria *flash* debería ser el último de los problemas.

Teniendo en cuenta estos requisitos, se ha evaluado una variedad de familias y marcas de MCUs y DSPs, y se ha llegado a la conclusión de que el MCU STM32F777VI de la familia F7 de STMicroelectronics cumple con todos los criterios establecidos. Sus características incluyen:

- Procesador ARM Cortex-M7 con FPU, funcionando a una velocidad de hasta 216 MHz.
- Amplio conjunto de interfaces de comunicación, incluyendo I^2C, USART, SPI, CAN, USB y Ethernet.
- 3 ADCs (24 canales máximo) de 12 bits y hasta 2.4 MSPS.
- 18 *timers*, incluyendo *timers* de 16 y 32 bits, dos *timers* con PWM avanzado.
- Soporte para depuración mediante interfaces SWD y JTAG, incluyendo el modo de *Trace Asynchronous Serial Wire*, que incorpora un pin de comunicación serie independiente de las capacidades de *debug* de SWD.
- Hasta 2 MB de memoria *flash*, SRAM de 512 Kbytes y 16 Kbytes de RAM TCM para datos críticos en tiempo real.

El STM32F777VI cumple con todos los requisitos establecidos y proporciona capacidades adicionales que pueden ser beneficiosas para el desarrollo y la operación del convertidor. Además, es el mismo microcontrolador que se usa en el resto de ECUs del monoplaza, conviertiéndose en la elección más cómoda para el futuro desarrollo.

4.3.10. *Software* de CAD electrónico

El diseño de las PCBs se realizó utilizando el *software* Altium Designer, una herramienta ampliamente utilizada en la industria para diseños electrónicos. El programa ofrece una amplia gama de funcionalidades que facilitan el proceso de diseño, desde la creación de esquemáticos hasta la disposición de componentes y el enrutado de pistas.

- **Creación de esquemáticos:** Altium Designer proporciona un entorno intuitivo para crear el esquemático del circuito. Primero se crean los componentes usando la información de la hoja de datos para hacer un buen *footprint* y símbolo, y después se incluyen los símbolos de los componentes en el esquemático y se realizan las conexiones eléctricas.
- **Diseño de PCB:** Una vez completado el esquemático, Altium Designer facilita la transferencia de diseño a la PCB. Los componentes del esquemático quedan vinculados y de forma muy sencilla se importan los cambios del esquemático a la PCB. Posteriormente se realizan las conexiones eléctricas en la PCB en un

proceso conocido como enrutado.

- **Verificación de reglas:** Altium Designer permite definir y verificar reglas de diseño para garantizar que la PCB cumpla con las restricciones de los fabricantes. Esto incluye reglas de espaciado, de colisiones, etc.
- **Generación de archivos de fabricación:** Una vez completado el diseño de la PCB, Altium Designer facilita la generación de archivos necesarios para la fabricación, como los archivos *Gerber*, la lista de materiales (BOM) o el archivo de taladros. Además, permite exportar un modelo 3D de la PCB con todos los componentes, lo cual facilita mucho la integración mecánica.

4.3.11. PCB de potencia

Esta PCB es una placa de 4 capas que integra componentes de potencia y circuitos de adquisición para el control de un solo motor eléctrico. Se requieren de dos de estas PCBs para montar el convertidor al completo.

Concepto y *layout*

La PCB de potencia es el sustrato físico sobre el cual se montan y conectan los componentes de potencia. En ella están situados los semiconductores, el bus de condensadores, las medidas de tensión, corriente y temperatura, los *gate drivers*, y algunos circuitos de seguridad. En la figura 4.66 se muestran unos bocetos iniciales que se usaron para tantear el emplazamiento de los componentes y conexiones de potencia.

Figura 4.66: Bocetos del *layout* del inversor.

Figura 4.67: *Layout* de la PCB de potencia, con las zonas más importantes marcadas.

En el *layout* se distinguen varias zonas.

- **Zona 1:** Conector a la PCB de control y protección de la alimentación de baja tensión.
- **Zona 2:** Conectores de las fases del motor y montajes mecánicos.
- **Zonas 3, 4 y 5:** Semiconductores, sensores de corriente, *gate drivers* y *snubbers* para cada fase.
- **Zona 6:** Bus de condensadores, conectores de alta tensión DC, circuito de descarga y sensado de tensión.

Restricciones y enrutado

Para dotar al convertidor de una alta densidad de potencia se decidió un formato de 150 mm por 200 mm, dejando espacio más que suficiente para distribuir todos los componentes de forma más o menos holgada.

Dado que esta PCB implementa partes del circuito del sistema de tracción y partes del sistema de baja tensión, se debe crear una separación física de 4 mm ya que estará acabada con un revestimiento conformal acrílico (*conformal coating*), acorde con la norma ***EV4.3.6*** [8].

Los espaciados entre conductores de la parte de alta tensión son de 3 mm, ya que la tensión que deben aislar es de 600 V. Este valor se obtiene de la norma genérica IPC-2221B [12], que especifica una separación de $2{,}5+(V-500)\cdot 0{,}005$ mm $= 3$ mm. De todas maneras, se usará un revestimiento conformal acrílico para asegurar una mejor característica dieléctrica y prevenir la aparición de arcos eléctricos.

El dimensionamiento de las pistas para la corriente que debe pasar es algo complicado, y de hecho condiciona el apilado de la PCB. Por ello, se selecciona un apilado de 4 capas de 70 micras (2 onzas por pulgada cuadrada).

Figura 4.68: Apilado de la PCB de potencia.

Para dimensionar el ancho de estas pistas se ha considerado la corriente RMS de fase, ya que es la corriente máxima en toda la PCB. Después, se ha usado este ancho para todas las conexiones de potencia, aunque la corriente que vean sea menor. El cálculo se ha realizado usando la norma IPC-2152 [11], que especifica unas relaciones de capacidad de corriente respecto al incremento de temperatura y el grosor de capa. Ya que el cálculo es algo tedioso, se opta por el uso de una calculadora específica para este propósito:

Figura 4.69: Cálculo del ancho de pista necesario.

El resultado es de 10 mm, lo cual es exagerado, pero sin embargo, solamente se está teniendo en cuenta una capa. Por ello, se decide enrutar las conexiones de potencia por un mínimo de 2 capas con un ancho mínimo de 5 mm. A continuación se detalla el enrutado de la conexión DC positiva de uno de los 3 módulos de medio puente.

(a) Vista del enrutado de baja corriente (azul), la conexión positiva del bus de continua (naranja) y de la fase A (rosado).

(b) Detalle de una sección más estrecha.

Figura 4.70: Ejemplo de enrutado de la conexión DC positiva del medio puente de la fase A.

Como se puede apreciar, el espaciado que hay que dejar por aislamiento compite por espacio con el ancho de las conexiones. En la mayoría de casos se ha podido respetar el ancho mínimo de 5 mm, sin embargo, en algunas situaciones esto no es posible. Por ejemplo, lo ilustrado en la subfigura (b) muestra una estrechez de 1,5 mm, pero como esa conexión está repetida en dos capas, el ancho total es de 3 mm. Además, la conexión positiva en el semiconductor se realiza a través de dos grupos de pines, y como antes de realizar esa estrechez ya se ha conectado un grupo de pines, se puede asumir que la corriente que circula por ese trozo de cobre es la mitad que la corriente continua entera que va a recibir ese módulo, que a su vez, es aproximadamente un tercio de la corriente continua total ($\frac{1}{3}\frac{17{,}5\ \mathrm{kW}}{450\ \mathrm{V}} = 13\ \mathrm{A}$), pasado por la calculadora, tan solo son necesarios 2,8 mm de cobre para esa corriente, lejos de los 10 mm que se han intentado mantener.

Esto es tan solo un ejemplo del análisis que se ha realizado con cualquier mínima sospecha de falta de cobre en todo el enrutado de potencia. Un camino de corriente muy estrecho podría presentar sobrecalentamiento y en casos extremos, levantamiento de la pista o plano.

Además, se han especificado el acabado y el espesor de cobre en las vías para asegurar un buen montaje de los componentes *press-fit* (los conectores de potencia y los semiconductores). Se ha optado por un acabado ENIG (oro electrolítico) por ser una opción con la que se pueden hacer este tipo de conexiones, además de aguantar mucho tiempo sin oxidarse. El ancho de cobre en las vías es algo que

se debe consultar directamente con el fabricante porque no todos son capaces de asegurar una tolerancia. Para saber cuánto cobre es necesario, se han consultado los documentos explicativos de Würth Elektronik (para los conectores de potencia) y de Wolfspeed (para los semiconductores).

	Minimum	Typical	Maximum
Hole drill diameter	1.12 mm	1.15 mm	
Copper thickness in hole	25 µm		50 µm
Metallization in hole			15 µm
End hole diameter	1.00 mm	1.05 mm	1.18 mm
Copper thickness of conductors	35 µm	70 µm – 105 µm	400 µm
Metallization of circuit board		Tin (chemical) recommended	
Metallization of pin		Tin (galvanic)	

Figura 4.71: Requisitos para la PCB según Wolfspeed [35].

Materials and tolerances

REDCUBE PRESS-FIT from Würth Elektronik are manufactured from the material CuZn39Pb3 and are therefore RoHS-compliant according to the RoHS stipulation concerning copper alloys.

The circuit board thickness should ideally be between 1.6 and 3.2 mm. Tested surfaces are chemical tin, HAL and ENIG. The **immersion tin** coating process is recommended. Using this process usually guarantees that the tin is evenly distributed in the case whereby the tolerances can be complied with more easily and thus chip formation can be prevented. Due to the uneven distribution of the tin in the case for the HAL process, we recommend the immersion tin process for circuit board thickness of 2.4 mm and greater. ENIG can be used but not recommended for Press-Fit technology.

Unless otherwise noted in the corresponding drawing, Würth Elektronik **REDCUBE** PRESS-FIT have quadratically designed press-fit pins. The through-hole plating in the PCB must therefore have the following characteristics:

Figura 4.72: Requisitos para la PCB según Würth Elektronik [36].

Ambos documentos apuntan que un acabado de estaño químico es lo ideal para realizar esta conexión, sin embargo, no se pudo conseguir un fabricante que lo

pudiera realizar a un precio razonable. El grosor de cobre que se marcó como requisito al fabricante fue de 25 µm a 50 µm, por ser el más restrictivo de ambos.

Bloques funcionales

La PCB de potencia se divide en varios bloques funcionales, unidos en un esquemático jerárquico que además incluye la conexión entre estos bloques, algunas notas indicativas y componentes que no pertenecen a ningún bloque en específico.

Figura 4.73: Esquemático jerárquico de la PCB de potencia.

En el esquemático se pueden distinguir el conector a la PCB de control, la protección de alimentación de baja tensión y unas notas indicativas, así como una leyenda de colores. Se anotan también los cambios y la fecha en la que se envía a producción. Los bloques que se pueden ver incluyen:

- **Etapa de potencia:** En este bloque aparecen los semiconductores, el bus de condensadores y una descarga pasiva.
- **Sensado de corriente:** Se incluyen los sensores de corriente y su interfaz analógica con la PCB de control.
- ***Gate drivers*:** En este bloque se recogen las señales necesarias para la conmutación, así como la configuración de los *gate drivers* en su conjunto. Este bloque contiene 3 subcircuitos correspondientes con cada fase. En ellos se encuentran los *gate drivers* en sí, junto a su alimentación aislada, los *snubbers* y la medida de temperatura.
- **DC:** Por último, hay un bloque dedicado al bus de continua, que incluye el circuito de descarga, la medida de tensión y la detección de 60 V requerida por

la normativa.

Circuitos importantes

VSI

Ya que los módulos son estructuras *half-bridge*, se deben conectar entre sí para formar la topología VSI.

Figura 4.74: Conexión de los módulos configurando un VSI.

Protección de alimentación

Aunque en principio es regulada, la entrada de 5 V necesita de un mínimo de protecciones sencillas para evitar retrasos ocasionados por fallos mientras se realizan pruebas. En este caso se ha usado un fusible como protección contra sobrecorriente, un diodo *zener* a modo de protección contra sobretensión (hará saltar el fusible cuando la tensión de entrada supere la tensión umbral), y un diodo *schottky* como protección frente a tensión inversa, ya que tiene poca caída de tensión, actúa bastante rápido y es barato.

Figura 4.75: Circuito de protección de alimentación.

Alimentación del *gate driver*

Figura 4.76: Alimentación bipolar aislada del *gate driver*.

En función del modelo de DC-DC específico escogido de la serie MGJ2 de Murata, se tendrá una tensión positiva y negativa diferente. La tensión positiva no es muy problemática puesto que se pueden seleccionar DC-DCs con salidas de +18 V o de +15 V, valores que encajan con los recomendados por los fabricantes de los semiconductores.

Sin embargo, ese no es el caso para las tensiones negativas, ya que el valor debería poder ajustarse para ser más versátil en el desarrollo. Por ello, se implementan reguladores lineales para ajustar $V_{\mathrm{EE,HS}}$ y $V_{\mathrm{EE,LS}}$ durante las pruebas para afinar el voltaje negativo necesario.

V_{EE} se calcula

$$V_{\mathrm{EE,LS}} = -1{,}186 \cdot \left(1 + \frac{R1}{R2}\right) \ ,$$

donde $R1 + R2 \approx 100$ kΩ según la hoja de datos del LDO. Para las configuraciones específicas de Leapers y Wolfspeed, los valores de las resistencias son

- Leapers: $R1 = 36$ kΩ, $R2 = 56$ kΩ
- Wolfspeed: $R1 = 68$ kΩ, $R2 = 30$ kΩ

Este proceso permite ajustar de forma precisa los voltajes negativos necesarios para el correcto funcionamiento del circuito de conmutación.

Gate driver

Figura 4.77: *Gate driver* con medida de temperatura y *snubbers*.

Uno de los componentes más importantes de todo el diseño tiene una de las implementaciones más sencillas, ya que la hoja de datos del UCC21710 proporciona mucha información útil para la aplicación final.

El *gate driver* UCC21710 se implementa con varias características que lo hacen adecuado para su uso en aplicaciones de potencia. Las señales *TRIP* y *OK* están configuradas en *open drain*, lo que permite que se paralelicen con los otros *gate drivers* para facilitar su integración. La entrada *IN-* no se utiliza y está conectada a tierra. El pin *ENABLE* se controla desde el MCU, y cuando se fuerza un estado bajo durante más de 1 µs, se resetea la señal *TRIP*. La señal *TRIP* se lee como una interrupción externa desde el MCU, y sirve para detectar un error en los *gate drivers* y actuar rápidamente.

Para la medición de temperatura, se utilizan los *drivers* de los MOSFETS del *low-side*. Estos proporcionan una corriente de salida de 200 µA en su salida, a través de la cual se puede interpretar la medida. Esta señal se convierte a PWM y luego a analógica mediante un filtro RC para ser recibida directamente por el ADC del MCU.

El circuito de medición de temperatura utiliza el sensor NTC del semiconductor para obtener una lectura de tensión, la cual se convierte en una lectura de bits mediante el ADC del MCU. La relación entre la resistencia del NTC y la temperatura se modela utilizando la fórmula de Steinhart-Hart, que toma en cuenta el valor β del NTC, la resistencia a una temperatura de referencia, y la temperatura ambiente. La fórmula utilizada es

$$NTC = R_0 \cdot e^{\left[-\beta \cdot \left(\frac{1}{T_0} - \frac{1}{\mathrm{T}}\right)\right]} .$$

El voltaje leído por el ADC se calcula

$$V_{\mathrm{ADC}} = V_{\mathrm{CC,GD}} \cdot \left(\frac{-20 \cdot I_{\mathrm{AIN}} \cdot (R_{\mathrm{filt}} + NTC)}{100} \right) = 5\ \mathrm{V} \cdot \left(\frac{-20 \cdot 200\ \mu\mathrm{A} \cdot (10\ \mathrm{k}\Omega + NTC)}{100} \right) .$$

El circuito también incluye protección de *Miller clamp* para evitar activaciones accidentales de los MOSFETs. Aunque los *gate drivers* tienen un *pull-down* activo, se ha implementado un *pull-down* externo con $R_{\mathrm{GS,HS}}$ y $R_{\mathrm{GS,LS}}$. Sin embargo, la detección de sobrecorriente no está implementada en este diseño porque es posible que su implementación traiga más problemas que los que pretende solucionar.

Adicionalmente, se incluyen unos *snubbers* consistentes en un RC entre *drain* y *source*, un condensador entre *gate* y *drain* y un condensador entre *gate* y *source*. En caso de encontrar problemas excesivos con la conmutación, pueden ayudar a amortiguar transitorios.

Cabe destacar que el enrutado de este subcircuito es especialmente crítico por los parásitos inductivos que se pueden ocasionar. En particular, la inductancia en el camino de retorno de *gate-source* puede ser fatal para el comportamiento de la conmutación, razón por la cual se ha tirado un plano conectado al *source* de

cada MOSFET en el área donde se enruta su *driver*, para minimizar el camino de retorno. En estas líneas, se ha enrutado en orden de prioridad, empezando por la conexión de las puertas de los MOSFETs a los *drivers* y la alimentación aislada, y acabando con los *snubbers*.

Figura 4.78: Enrutado en la capa superior de un *half-bridge*.

Debido a los parásitos que puede presentar el *layout*, los MOSFETs pueden presentar un sobrepico de tensión excesivo al conmutar. La figura 4.79 explica el circuito que pueden formar. Si la capacidad parásita es insuficiente, la inductancia parásita provocará una subamortiguación. Por ello, se ha minimizado el área que dejan estas conexiones para minimizar la inductancia parásita. Además, es importante añadir un condensador de desacoplo entre los terminales positivo y negativo de cada *half-bridge*.

Figura 4.79: Circuito de parásitos en un convertidor *buck* [7]. Cada medio puente de un VSI actúa como un convertidor *buck* cuando la corriente es positiva.

Descarga

Figura 4.80: Circuito de descarga.

Una de las partes más importantes en cuanto a la seguridad eléctrica es la implementación del circuito de descarga. Este circuito se utiliza para descargar la energía almacenada en el bus de condensadores cuando el convertidor no está operativo. Los requisitos de la normativa son que el circuito debe ser capaz de descargar el bus de la tensión máxima hasta 60 V en menos de 5 segundos. Adicionalmente, el circuito debe ser capaz de aguantar de forma constante la tensión máxima.

Con tal de cumplir estos requisitos se ha optado por un circuito basado en un NMOS, cuya puerta se conecta al terminal positivo del bus a través de una resistencia de valor elevado y se limita la tensión con un diodo *zener*. La puerta del MOSFET está controlada por un opto-acoplador de forma aislada desde el circuito de baja tensión. Mientras el bus tenga una tensión superior a 10 V (la tensión del *zener*) y el opto-acoplador esté desactivado, el MOSFET quedará cerrado, conectando el banco de resistencias a los condensadores, descargándolos.

La selección de la resistencia se ha basado en minimizar el tamaño de la misma. La solución óptima se ha encontrado en utilizar un banco de 24 resistencias SMD de 470 kΩ, en encapsulado 2512, que es capaz de disipar 1 W. Se conectan en paralelo

para llegar al valor necesario para cumplir con el requisito de tiempo de descarga. Para el cálculo del tiempo de descarga se utiliza la siguiente expresión

$$t_{\text{dis}} = R_{\text{dis}} \cdot C \cdot \ln\left(\frac{V_{\text{inicial}}}{V_{\text{final}}}\right) \ . \tag{4.67}$$

$$t_{\text{dis}} = R_{\text{dis}} \cdot C \cdot \ln\left(\frac{V_{\text{inicial}}}{V_{\text{final}}}\right) = \left(\frac{470\ \text{k}\Omega}{24}\right) \cdot (100\ \mu\text{F}) \cdot \ln\left(\frac{600\ \text{V}}{60\ \text{V}}\right) = 4{,}509\ \text{s}$$

La potencia disipada en cada resistencia de descarga se calcula como

$$P(R_{\text{dis}}, \text{máx}) = \frac{V_{\text{máx}}^2}{R_{\text{dis}}} \ . \tag{4.68}$$

$$P(R_{\text{dis}}, \text{máx}) = \frac{V_{\text{máx}}^2}{R_{\text{dis}}} = \frac{(600\ \text{V})^2}{470\ \text{k}\Omega} = 0{,}766\ \text{W} < 1\ \text{W}$$

Medida de corriente

Figura 4.81: Sensor de corriente de fase.

Uno de los componentes más críticos para asegurar el funcionamiento óptimo del inversor es la medición precisa de corriente. En inversores de bajo coste, comúnmente se emplean resistencias *shunt* junto con amplificadores aislados para esta tarea. Sin embargo, esta solución puede presentar diversos desafíos de integración.

Una opción destacada para abordar esta necesidad de medición de corriente es la gama de sensores CKSR-NP de LEM. Este sensor utiliza la tecnología de transductores de corriente basados en *fluxgate*. Sus características incluyen un rango de medición amplio de hasta ±150 A para modelo seleccionado, junto con una alta precisión del 0.8 %. Además, su diseño compacto y montaje en PCB lo hacen ideal para aplicaciones donde se requiere una medición precisa y confiable de corriente en un espacio limitado.

Dado que el sensor tiene un rango de medida de ±150 A y está alimentado a 5 V, se puede pasar por un divisor resistivo con tal de adecuar la señal de salida al rango

de entrada del ADC del MCU. Adicionalmente, se puede modificar la tensión de referencia (salida a 0 A) para ajustar todavía más la medida.

Figura 4.82: Sensor de corriente LEM CKSR 50-NP [16].

Cabe destacar que la serie CKSR-NP de LEM es compatible en especificaciones y *footprint* con la serie LKSR-NP, con lo que se pueden usar indistintamente. Para el proyecto se han conseguido muestras de LKSR-NP y se compraron unos pocos CKSR-NP.

Medida de tensión

Figura 4.83: Circuito de medida y detección de tensión DC.

El sensado de la tensión es una tarea crítica en el sistema, ya que no solo es esencial para el control del motor, sino también para la detección y alerta de niveles altos de tensión. Para ambos propósitos, se emplea un divisor de tensión utilizando una serie de resistencias.

$$V_{\text{DC,div}} = \left(TS^{+} - TS^{-}\right) \cdot \left(\frac{4{,}7\ \text{k}\Omega}{4{,}7\ \text{k}\Omega + 6 \cdot 68\ \text{k}\Omega}\right)$$

Sustituyendo los valores de 600 V y 60 V para TS+ y TS-, respectivamente,

$$V_{\text{DC,div}} = \frac{600\ \text{V} \cdot 4{,}7\ \text{k}\Omega}{4{,}7\ \text{k}\Omega + 6 \cdot 68\ \text{k}\Omega} = 6{,}833\ \text{V}$$

$$V_{\text{DC,div}} = \frac{60\ \text{V} \cdot 4{,}7\ \text{k}\Omega}{4{,}7\ \text{k}\Omega + 6 \cdot 68\ \text{k}\Omega} = 683\ \text{mV}\ .$$

El valor de potencia disipada en las resistencias de 68 kΩ se calcula

$$P_{\text{R68k}} = I^2_{\text{R4k7}} \cdot 68\ \text{k}\Omega = \left(\frac{V_{\text{DC,div,máx}}}{4{,}7\ \text{k}\Omega}\right)^2 \cdot 68\ \text{k}\Omega = \left(\frac{6{,}833\ \text{V}}{4{,}7\ \text{k}\Omega}\right)^2 \cdot 68\ \text{k}\Omega = 144\ \text{mW}.$$

Se elige un encapsulado 1206 para las resistencias de 68 kΩ, que es capaz de disipar por lo menos 250 mW.

El amplificador aislado ISO224 se utiliza para medir la tensión en el bus de continua. Según la hoja de datos, la salida se calcula como un tercio del voltaje de entrada del divisor de tensión.

$$(V_{\text{DC,sns+}} - V_{\text{DC,sns-}}) = \frac{1}{3} \cdot V_{\text{DC,div}} = \frac{1}{3} \cdot \left((TS^{+} - TS^{-}) \cdot \frac{4{,}7\ \text{k}\Omega}{4{,}7\ \text{k}\Omega + 6 \cdot 68\ \text{k}\Omega}\right)$$

$$(V_{\text{DC,sns+}} - V_{\text{DC,sns-}}) = \frac{1}{3} \cdot 0{,}011388 \cdot (TS^{+} - TS^{-})$$

$$(V_{\text{DC,sns+}} - V_{\text{DC,sns-}}) = \frac{1}{3} \cdot 0{,}011388 \cdot 600\ \text{V} = 2{,}278\ \text{V}$$

El amplificador aislado ISO224 es una elección eficaz para esta aplicación, ya que permite medir de manera precisa la diferencia de potencial entre los terminales, proporcionando un voltaje de salida proporcional a la tensión medida y cuadrando con el rango de adquisición del ADC. Adicionalmente, la salida es diferencial, lo que permite mayor inmunidad frente a interferencias al ser enrutada hasta la placa de control. En ella se deberá incluir un amplificador diferencial o similar para llevar la lectura al ADC.

Resultado final

Tras haber creado todos los subcircuitos, haber emplazado cada componente y enrutado cada nodo, se ha completado el diseño de la placa de potencia. La distribución de componentes permite empaquetar dos de estas placas enfrentadas de forma muy compacta, dejando el espacio justo entre medias para la inserción de la *coldplate*. La placa de control se podrá conectar a las dos de potencia mediante

conectores placa a placa, quedando a uno de las caras libres. Por la otra cara se deben incluir las conexiones de potencia DC y la entrada y salida de agua.

(a) Vista superior. (b) Vista inferior.

Figura 4.84: Vistas 3D de la PCB de potencia.

(a) Vista frontal, conexión del bus DC y *coldplate*. (b) Vista trasera, conexión de la placa de control.

Figura 4.85: Vistas 3D del ensamblaje de dos PCBs de potencia.

4.3.12. PCB de control

Concepto y *layout*

En esta placa se alojará el microcontrolador con todos los componentes necesarios para interactuar con las placas de potencia y el exterior. Dado que el esfuerzo de integración es mucho menor, las restricciones mecánicas y la facilidad de montaje son las que rigen el concepto. Por ello no se han tenido muchos miramientos para la disposición de los componentes, y simplemente se ha puesto atención en utilizar muchas simetrías para hacer que los circuitos repetidos para los dos motores/inversores sean exactamente iguales.

Figura 4.86: *Layout* de la PCB de control.

Como se puede observar, el microcontrolador es la pieza central y está orientado a 45 grados para que los pines se puedan enrutar de forma óptima hacia los extremos de la placa. Ambos lados de la placa son simétricos a excepción de los componentes que no están duplicados por el control dual.

Restricciones y enrutado

Se ha mencionado ya que esta placa no tiene muchas restricciones más allá de las mecánicas, lo cual ha permitido crearla en un formato de 150 mm por 40 mm, encajando perfectamente en ángulo recto con las placas de potencia en el espacio que dejan entre medias.

El apilado escogido es un estándar de 4 capas, con 35 micras de espesor en todas las capas (1 onza por pulgada cuadrada). Para hacer la placa incluso más barata, se puede fabricar con 17 micras en las capas internas (0.5 onzas por pulgada cuadrada). Esto asegura una producción económica, y la posibilidad de crear planos de referencia y alimentación en las capas internas.

Figura 4.87: Apilado de la PCB de control.

Dado que no hay grandes magnitudes eléctricas, el enrutado se ha centrado en minimizar el área de los caminos de corriente entre cada señal y la masa de la placa. Utilizando polígonos conectados a la masa en múltiples capas (tanto la de la propia señal como las que queden por encima o por debajo) se minimiza muchísimo este área.

También se ha procurado mantener todas las señales rápidas lo más separadas posibles para evitar el *cross-talk* entre ellas, causado por el acople capacitivo que presentan dos conductores paralelos. En ocasiones no se ha podido evitar y tan solo queda confiar en el resto de buenas prácticas de enrutado.

Figura 4.88: Ejemplo de grupo de señales que podrían presentar *cross-talk*.

Bloques funcionales

La PCB de control se divide en varios bloques funcionales, que igual que la PCB de potencia, se juntan en un esquemático jerárquico.

Figura 4.89: Esquemático jerárquico de la PCB de control.

En el esquemático se pueden apreciar el conector al vehículo, las monturas de la placa, unas notas indicativas con los cambios y una leyenda de colores. Los bloques

son los siguientes:

- **Alimentación**: Dado que esta placa no tiene conexión al sistema de alta tensión, se alimenta exclusivamente del sistema de baja tensión del monoplaza, cuya tensión es de entre 20 V y 30 V. Por ello se implementan distintas protecciones, un convertidor para obtener un bus de 5 V y un regulador lineal para obtener 3.3 V estables para el MCU.
- **MCU**: Es el bloque central, que implementa el STM32F777VI, un puerto USB, una memoria externa EEPROM y el conector de programación y depuración.
- **CAN**: Incluye un transceptor CAN para habilitar la comunicación con el vehículo.
- **Retroalimentación**: En estos bloques se integra la conexión del *encoder* incremental y el *front-end* analógico de la lectura de temperatura del motor.
- ***Front-end* analógico de la placa de potencia**: Se tratan las distintas señales que provienen de las placas de potencia, adaptándolas a los rangos de tensión que admiten los ADCs del MCU.
- **Extras**: Tan solo se colocan unos LEDs y un par de entradas digitales para el MCU.

Configuración de *hardware* del MCU

Con tal de asignar de forma adecuada la función de cada pin, se ha hecho uso de una herramienta proporcionada por STMicroelectronics llamada CubeMX. Este programa permite generar un código base con todos los periféricos del microcontrolador configurados según se escoja. Tiene una interfaz muy intuitiva que hace muy sencilla la implementación de *hardware*, permitiendo un desarrollo muy ágil.

De esta manera, se han decidido las conexiones de la placa de control con base en las siguientes consideraciones:

- **Funcionalidad del pin:** Cada pin del microcontrolador se ha configurado para cumplir una función específica de acuerdo al mapeado que tenga cada periférico, ya que normalmente un canal específico de un periférico específico solamente se puede conectar a uno o dos pines.
- **Compatibilidad con periféricos:** Se ha tenido en cuenta la compatibilidad entre los periféricos del microcontrolador y los dispositivos externos conectados a la placa de control.
- **Optimización del enrutado:** Se han escogido algunos pines a medida que se ha ido enrutando la placa, por facilitar algunas conexiones.

Finalmente, se logra que el esquemático y el programa CubeMX muestren la misma distribución de pines:

Figura 4.90: Pantalla principal de CubeMX con todos los pines usados configurados con los periféricos adecuados.

Figura 4.91: Símbolo esquemático del MCU con todas las conexiones realizadas.

Circuitos importantes

Alimentación

Ya que se propone alimentar la placa directamente desde el sistema de baja tensión del vehículo, se debe implementar un tratamiento de esta alimentación. Se incluye protección contra descargas electrostáticas, polaridad inversa y sobrecorriente. Se implementa un DC-DC de Recom de 15 W para generar el bus principal de 5 V, sin embargo, dado a problemas de disponibilidad y ensamblaje no se ha podido probar. Además, se incluye alimentación del conector USB, y ambos buses de 5 V se pasan por un filtro Pi con frecuencia de corte $f_c = \frac{1}{2\pi\sqrt{C \cdot L}} = \frac{1}{2\pi\sqrt{10\ \mu F \cdot 47\ \mu H}} = 7{,}34$ kHz para evitar acoples de ruido en la alimentación. También se puede observar un regulador lineal fijo de 3.3 V para proporcionar una tensión estable al MCU, y un pequeño LED indicativo de que la alimentación está activa.

Figura 4.92: Esquemático de los circuitos de alimentación de la placa de control.

MCU

Aunque ya se ha visto la implementación del MCU en sí, su esquemático contiene otras partes como una memoria externa, el conector USB, notas sobre qué periféricos están mapeados a qué pines y el conector de programación y depuración.

Figura 4.93: Esquemático de los circuitos relacionados con el MCU.

CAN

Para implementar comunicación CAN es necesario incorporar un *transceiver* que pueda comunicar el MCU con un bus de CAN real. Se escoge el MCP2551 por su coste, simplicidad y robustez. Se añade también un filtro de línea consistente

en un *choke* para las interferencias en modo común, y algunos condensadores. Se incluye también un final de línea por si fuera necesario, y protección contra ESD. Adicionalmente, se han implementado un par de luces LED para indicar el correcto funcionamiento del envío y recepción de datos.

Figura 4.94: Esquemático referente a la comunicación CAN.

Retroalimentación

Este bloque se implementa por duplicado, un bloque por cada motor que se controla. Contiene los componentes necesarios para leer su posición y temperatura.

Ya que se requiere de leer la posición con un *encoder* incremental, se equipan los componentes necesarios para ello. Se usa el LM339, un comparador cuádruple con salida HiZ/GND para obtener la lectura desde una interrupción externa en el MCU. Cabe destacar que esta configuración de *hardware* hace compatible este inversor con sensores de efecto Hall para la lectura de posición, aunque no se desarrollará el código que los interprete.

Figura 4.95: Esquemático del *front end* del *encoder* incremental.

A fecha de la redacción de este trabajo todavía no se conoce el sensor de temperatura que montarán los motores, de modo que se implementa un circuito modificable para la lectura de cualquier tipo de sensor de temperatura resistivo. Aprovechando el comparador restante del LM339 se añade también una alarma configurable por *hardware* para cualquier tipo de sensor.

Figura 4.96: Esquemático de la lectura de temperatura del motor y alarma arbitraria.

Por último, se añade un conector para el sensor que incluye también una alimentación de 5 V, el sensor de temperatura y el sensor arbitrario junto con unas notas sobre la implementación.

Figura 4.97: Esquemático del conector del sensor de posición y notas.

Front end de la placa de potencia

Este bloque se implementa por duplicado, un bloque por cada placa de potencia que tiene el inversor.

Figura 4.98: Esquemático del *front end* de la placa de potencia.

En primer lugar, *ENABLE* es una salida directa del MCU, y se ha comprobado que el UCC21710 es capaz de detectarla a 3.3V, igual que los PWMs. De manera similar, *TRIP* se lee a 5 V y utiliza un GPIO tolerante a 5 V en el MCU.

Para que el ADC del MCU reciba correctamente todas las señales analógicas, se deben tratar adecuadamente. Hay tres grupos de señales analógicas en cada placa de potencia: las corrientes de fase, la tensión DC y la temperatura de los semiconductores.

Corrientes de Fase

Como se ha visto en el apartado de la placa de potencia, se utilizan sensores referenciados a 5 V, y por tanto, sus señales podrían exceder el rango de 3.3 V del ADC. Por ello se implementa un simple divisor de tensión. La combinación de resistencias se puede ajustar para aumentar el rango de medición a costa de una menor resolución.

El *offset* de la corriente en bits se calcula

$$\mathit{offset}_i = \left(\frac{10\ \mathrm{k\Omega}}{3{,}3\ \mathrm{k\Omega} + 10\ \mathrm{k\Omega}}\right) \cdot 2{,}5\ \mathrm{V} \cdot \frac{2^{12}\ \mathrm{bits}}{3{,}3\ \mathrm{V}} = 2333\ \mathrm{bits}\ ,$$

y la ganancia de la medida de corriente

$$\mathit{gain}_i = \frac{\left(\frac{10\ \mathrm{k\Omega}}{3{,}3\ \mathrm{k\Omega}+10\ \mathrm{k\Omega}}\right)}{12{,}5\ \mathrm{mV/A} \cdot \left(\frac{2^{12}\ \mathrm{bits}}{3{,}3\ \mathrm{V}}\right)} = 0{,}0484609962\ \mathrm{A/bit}\ .$$

La corriente máxima que se puede medir es

$$\pm 0{,}0484609962\ \mathrm{A/bit} \cdot \frac{2^{12}\ \mathrm{bits}}{2} = \pm 99\ \mathrm{A}\ .$$

Para los ensayos se instalaron resistencias de 4,7 kΩ en vez de 3,3 kΩ y se calcularon de nuevo los parámetros para la lectura correcta.

Tensión de Bus

Dado que el amplificador aislado utilizado saca una señal diferencial, se debe convertir a *single-ended* usando un amplificador diferencial integrado.

Puesto que el modelo escogido presentaría una pequeña zona muerta, se debe añadir un *offset* de muy pocos milivoltios.

No existen referencias de tensión de tan poco nivel, así que se usa un divisor resistivo de valores bajos, a sabiendas de que esta decisión causa error en la medida.

El *offset* de la medida de tensión se calcula

$$\mathit{offset}_v = V_{ref} \cdot \frac{2^{12}\ \mathrm{bits}}{3{,}3\ \mathrm{V}} = 129\ \mathrm{bits},$$

y la ganancia de la medida de tensión en voltios por bit se calcula

$$gain_v = \frac{1}{\left(\frac{1}{3} \cdot 0{,}011388 \text{ V/V} \cdot \left(\frac{2^{12} \text{ bits}}{3{,}3 \text{ V}}\right)\right)} = 0{,}212240269 \text{ V/bit}.$$

La tensión máxima que se puede medir es

$$0{,}212240269 \text{ V/bit} \cdot 2^{12} \text{ bits} = 869{,}34 \text{ V} .$$

Temperaturas de los semiconductores

Sería poco práctico leer las tres temperaturas de cada inversor ya que ocuparían muchos pines del MCU. Por ello, se añaden unas pequeñas resistencias, dos de las cuales no se montan, lo cual permite escoger una de las tres medidas de temperatura. Las temperaturas de los inversores deben calcularse con una *lookup table* para ahorrar tiempo de computación. Como se ha visto, la señal que sale del UCC21710 es una señal pulsada que puede leerse directamente como una entrada PWM o pasar a través de un filtro RC en la placa de potencia para convertirla en una señal analógica. Esta placa pretende leerla con el ADC.

La lectura en sí misma está en la parte de alta tensión de la placa de potencia y se conecta al pin AIN de UCC21710. Basándose en el voltaje leído, se calcula el ciclo de trabajo (D) de la salida analógica aislada de UCC21710 utilizando la relación proporcionada por el fabricante [31].

$$D = -20 \cdot V_{AIN} + 100.$$

Si se filtra, el voltaje en leído por el ADC del MCU se calcula

$$V_{Tinv} = VCC_{GD} \cdot \frac{D}{100} = 5V \cdot (-20 \cdot V_{AIN} + 100) /100.$$

Extras

En este bloque aparecen tres LEDs informativos controlados por el MCU, un interruptor para cambiar la dirección de giro de los motores, y una lectura de la cadena de seguridad.

Figura 4.99: Esquemático de extras.

Resultado final

Después de diseñar todos los circuitos, emplazado todos los componentes y enrutado todos los nodos, se completa el diseño de la placa de control.

(a) Vista superior.

(b) Vista inferior.

Figura 4.100: Vistas 3D de la PCB de control.

4.3.13. Ensamblaje del convertidor

El ensamblaje del convertidor se realizó siguiendo un proceso meticuloso para garantizar la correcta integración de todos los componentes del mismo. Se utilizó un enfoque basado en diseño asistido por computadora (CAD) para planificar y visualizar el montaje antes de la construcción física. Durante el diseño de las PCBs se fue comprobando mediante CAD que no existían colisiones y que el convertidor se podía montar, atendiendo a razones como el acceso de herramientas.

Diseño en CAD

Se utilizó Solidworks para crear un modelo tridimensional detallado del convertidor. Se empezó por importar los archivos de ambas PCBs generados por Altium.

Figura 4.101: Ensamblaje de la placa de control con las dos placas de potencia.

A este ensamblaje se le añadieron unas pletinas para conectar entre sí los buses de continua, ya que van conectados a la misma batería. Además, se planteó un concepto de *coldplate* atendiendo únicamente a las restricciones mecánicas. Este pre-diseño no fue estudiado ni simulado, simplemente marca una primera referencia para desarrollar la *coldplate* en el futuro. También se incluyó la tornillería necesaria para el montaje.

Figura 4.102: Ensamblaje con las pletinas, concepto de *coldplate* y tornillería.

Ensamblaje real

Como se discutirá en el capítulo de resultados, el convertidor fue montado poco a poco para validar sección a sección. Se soldaron los componentes manualmente con un lápiz de soldadura e hilo de estaño. Ocasionalmente se usó una pistola de calor para extraer componentes con conexiones con mucha masa térmica.

Figura 4.103: Soldadura manual con hilo de estaño.

Se montó una PCB de control y dos de potencia. Aprovechando que se dispone de los dos semiconductores seleccionados, se montó una placa de potencia con cada uno. A parte de los semiconductores, a penas hay que cambiar el valor de un par de resistencias.

Para montar los componentes *press-fit* se usó un tornillo de banco y un utillaje que permita apretar los componentes. Cabe destacar que se priorizó el montaje de los componentes con este tipo de conexión puesto que al realizar el proceso se generan flexiones en la placa que podrían dañar componentes como los condensadores cerámicos.

El caso de los conectores de potencia fue sencillo puesto que no son muchos pines y se pudo insertar fácilmente usando herramientas comunes. Para los semiconductores sin embargo, se tuvo que imprimir un utillaje personalizado para evitar colisiones con otros componentes.

Figura 4.104: Placa lista para el montaje de un componente *press-fit* con un utillaje personalizado.

Figura 4.105: Prensado del componente usando un tornillo de banco.

Figura 4.106: Resultado de la inserción del componente.

Figura 4.107: Cara trasera de la PCB de potencia (variante Wolfspeed) montada.

Figura 4.108: PCB de control montada.

Tras montar todos los componentes de todas las placas se procedió a ensamblarlas. A fecha de cuando se tomaron las siguientes imágenes no se dispuso de las pletinas.

Figura 4.109: Placa de control y placa de potencia (variante Wolfspeed).

Figura 4.110: Vista en la que se aprecian ambas placas de potencia con sus respectivos semiconductores, dejando el espacio para una *coldplate*.

4.4. *Firmware*

4.4.1. Desarrollo del *firmware*

El *firmware* de un proyecto como este no se puede desarrollar al estilo *maker*, de manera informal o improvisada. Requiere un enfoque metodológico y estructurado para garantizar la estabilidad, la eficiencia y la continuidad del proyecto. Es fundamental seguir prácticas de desarrollo de *software* bien establecidas, como el diseño modular, la documentación detallada del código y la gestión de versiones. Además, el cumplimiento de estándares de codificación y la realización de análisis estáticos son pasos críticos para garantizar la calidad y la seguridad del *firmware*. Este enfoque profesional y riguroso es esencial para crear una base de código ampliable, portable y eficiente.

Herramientas utilizadas

Análisis estático: Cppcheck

Cppcheck es una herramienta de análisis estático de código C/C++ que se utiliza para detectar errores y problemas potenciales en el código. Realiza un escaneo exhaustivo del código para identificar posibles problemas como fugas de memoria, uso incorrecto de punteros, variables no inicializadas y otros errores comunes de programación. La integración de Cppcheck en el flujo de trabajo de desarrollo ayuda a identificar y corregir estos problemas en una etapa temprana del proceso de desarrollo, lo que contribuye a mejorar la calidad y fiabilidad del *firmware*.

Figura 4.111: Cppcheck detectando un potencial problema.

Documentación del código: Doxygen

Doxygen es una herramienta de generación de documentación que se utiliza para crear documentación automática a partir del mismo código. Permite documentar el código usando comentarios especiales incrustados en el código mismo, utilizando una sintaxis sencilla y clara. Doxygen procesa estos comentarios para generar documentación en varios formatos, incluyendo HTML (como una pequeña web), y PDF (para una documentación más tradicional). Esta documentación incluye detalles sobre la estructura del código, la descripción de las funciones y variables, así como relaciones y dependencias entre diferentes partes del código. La generación automática de documentación con Doxygen facilita la comprensión y el mantenimiento del código, y proporciona una referencia útil para futuros desarrolladores.

Figura 4.112: Documentación del código en formato HTML generada automáticamente por Doxygen.

Plataforma de desarrollo

Para el desarrollo del *firmware*, se emplea STM32CubeIDE de STMicroelectronics, una plataforma de desarrollo integrado (IDE) basada en Eclipse y diseñada específicamente para trabajar con microcontroladores STM32. CubeIDE proporciona un entorno de desarrollo completo que incluye un editor de código, un compilador, un depurador y otras herramientas útiles para la programación de microcontroladores STM32. Además, integra STM32CubeMX, que facilita la configuración de periféricos y la generación de código inicial mediante una interfaz gráfica intuitiva.

Figura 4.113: STM32CubeIDE en la perspectiva de código.

El uso de CubeIDE simplifica el proceso de desarrollo de *firmware* para microcontroladores STM32, ofreciendo características como autocompletado de código, resaltado de sintaxis, depuración paso a paso y visualización de variables en tiempo real.

Lenguaje de programación

El lenguaje de programación seleccionado para desarrollar el *firmware* de este proyecto es C. Esta elección se basa en varias consideraciones relacionadas con la naturaleza de la aplicación embebida en un microcontrolador STM32.

En primer lugar, el lenguaje C es ampliamente compatible y soportado por la mayoría de los microcontroladores, incluidos los STM32. La vasta cantidad de recursos, bibliotecas y herramientas disponibles para el desarrollo en C facilita la implementación de funcionalidades complejas y la integración con el *hardware* específico del microcontrolador.

Además, el lenguaje C es altamente eficiente en términos de uso de recursos de memoria y velocidad de ejecución, lo cual es crucial en aplicaciones embebidas donde se deben cumplir estrictas restricciones de recursos.

Se reconoce que usar C++ podría tener beneficios en la legibilidad y la modularidad por tratarse de un lenguaje orientado a objetos. Sin embargo, las ventajas de usar C son demasiadas como para considerar escribir el código en C++.

Estilo de programación

Para mantener un código claro y coherente, se seguirán las siguientes convenciones de nomenclatura y estilo de programación:

1. **Convenciones generales:**
   - Se usará el inglés como idioma a la hora de programar para facilitar el entendimiento.
2. **Convenciones de nomenclatura para variables/estructuras/enumeraciones:**
   - Se utilizará `camelCase` para los nombres de variables y la declaración de estructuras/enumeraciones.
   - Se utilizará `PascalCase` para la definición de estructuras/enumeraciones.
   - Se emplearán nombres descriptivos que indiquen claramente el propósito de la variable.
   - Se agregarán extensiones `_left` y `_right` a las variables que representen componentes del inversor izquierdo y derecho, respectivamente.
   - Se evitará añadir más extensiones con guión bajo.
   - Las constantes se nombrarán en MAYÚSCULAS.

   Ejemplo:

- Para el *offset* y el *slope* del ADC:

```
#define CURRENT_SLOPE  117.57704f  /**< [A/V] ((4.7+10)
   /10) * (1 / (12.5 mV / A)) */
#define CURRENT_OFFSET 1.70068027211f /**< [V]
   (10/(4.7+10))* 2.5 V */
```

- Para la referencia de par:

```
torqueRef_left
torqueRef_right
```

- Para la definición y declaración de la estructura `Encoder`:

```
typedef struct {} Encoder; // Encoder struct definition
Encoder encoder_left; // Declaration of encoder_left
```

3. **Convenciones de nomenclatura para funciones:**

   - Se utilizará `snake_case` para los nombres de funciones.
   - Se emplearán nombres descriptivos que indiquen claramente el propósito de la función, incluso si parecen demasiado largos.
   - El nombre de todas las funciones debe empezar por un verbo.

   Ejemplo:

   - `initialize_inverter()`
   - `limit_torque_to_prevent_overspeed()`
   - `handle_direction()`

4. **Convenciones de nomenclatura para archivos:**

   - Los pares de archivos .c/.h desarrollados se nombrarán en MAYÚSCULAS, facilitando su distinción de los archivos generados automáticamente.

   Ejemplo:

   - `MEASUREMENTS.c/.h` → Desarrollado
   - `adc.c/.h` → Generado automáticamente

5. **Convenciones para abreviaturas:**

   - Se utilizarán abreviaciones comunes solo cuando sean ampliamente entendidas dentro del contexto del proyecto.
   - Se evitarán abreviaciones ambiguas o poco claras.

   Ejemplo:

   - `ADC` (Convertidor Analógico-Digital)

- `FSM` (Máquina de Estados Finitos)
- `LUT` (Tabla de Búsqueda)

6. **Comentarios:**
   - Cada archivo y función debe ir precedido por un comentario estilo Doxygen que explique claramente su propósito.

   Ejemplo:

```
/**
* @brief Computes d-q currents from current measurements and
   electrical angle.
*
* This function computes the d-q currents from phase currents
   (ABC), theta_e, and stores
* the results in the provided pointers.
*
* @param[in] ia Phase A current in A.
* @param[in] ib Phase B current in A.
* @param[in] ic Phase C current in A.
* @param[in] sinTheta_e Electrical angle sine (-1..1)
* @param[in] cosTheta_e Electrical angle cosine (-1..1)
* @param[out] idMeas Pointer to store the d-axis current.
* @param[out] iqMeas Pointer to store the q-axis current.
*/
void get_idiq(float ia, float ib, float ic, float sinTheta_e,
   float cosTheta_e, float *idMeas, float *iqMeas);
```

4.4.2. Arquitectura del *firmware*

Para abordar fácilmente el manejo de los dos inversores de forma independiente, se han escrito varios módulos de código que se van reutilizando para evitar repetir código.

`INVERTER.c/.h`

El archivo `INVERTER.c` y su correspondiente encabezado `INVERTER.h` constituyen la implementación y la interfaz de la estructura de datos y las funciones asociadas a un inversor.

La principal funcionalidad de estos archivos es proporcionar una interfaz cohesiva y modular para la gestión de los inversores. La estructura `InverterStruct` definida en `INVERTER.h` encapsula todos los elementos necesarios para el funcionamiento de un inversor, incluyendo periféricos, estados de operación, estructuras de datos para mediciones y referencias y lazos de control, entre otros. Esta encapsulación permite una gestión eficiente y organizada de la complejidad asociada al control de los dos inversores. Si el código estuviera escrito en un lenguaje como C++ se podría crear una clase en vez de una estructura, pero al estar programado en C, solamente se han podido mantener las buenas prácticas de la programación orientada a objetos.

Un aspecto destacado de la implementación es la escritura de código modular y reutilizable, por ejemplo, las funciones se implementan de manera independiente

de la configuración específica del *hardware* del inversor. Se generan dos estructuras globales `InverterStruct` independientes, `inverter_left` e `inverter_right`, para controlar cada uno de los inversores de forma independiente. Esta estrategia permite un control simultáneo de ambos inversores sin repetir ni una línea de código.

`MOTOR.c/.h`

El archivo `MOTOR.c` y su correspondiente encabezado `MOTOR.h` constituyen la implementación y la interfaz de las funciones y estructuras asociadas a los parámetros y constantes del motor.

La funcionalidad principal de estos archivos es proporcionar una interfaz modular para la gestión de los parámetros del motor. La estructura `MotorParameters`, definida en `MOTOR.h`, encapsula todos los parámetros de un motor eléctrico.

Un aspecto importante de la implementación es la inclusión de una estructura adicional, `MotorConstants`, dentro de `MotorParameters`, que almacena las constantes precalculadas relacionadas con el motor. Estas constantes se calculan en la función `precalculate_motor_constants()` y se utilizan en el cálculo de trayectorias para ahorrar algo de tiempo de ejecución.

Además, se proporciona una función de verificación de parámetros, `check_motor_parameters()`, que evalúa los valores de los parámetros del motor y realiza ajustes si se detectan errores o valores fuera de los límites especificados. Esta función garantiza que los parámetros del motor estén dentro de rangos seguros.

`FSM.c/.h`

El archivo `FSM.c` y su encabezado correspondiente `FSM.h` contienen la implementación y la interfaz de la Máquina de Estados Finitos (FSM) para controlar el inversor. La FSM gestiona el funcionamiento de un inversor a través de la ejecución de diferentes estados. Cada estado representa una etapa específica en el ciclo de funcionamiento del inversor y está asociado con un conjunto definido de acciones y condiciones de transición. El archivo `FSM.c` implementa funciones para gestionar cada estado.

Los cuatro estados implementados son:

- **Inicio (`Startup`)**: En este estado, se realizan las acciones de inicialización del inversor, como la configuración de periféricos y la detección de errores durante el arranque.
- **Reposo (`Idle`)**: El inversor está inactivo y espera órdenes para iniciar o detecta condiciones de error.
- **Funcionamiento (`Running`)**: El inversor está en pleno funcionamiento y se eje-

cuta el bucle de control principal.

- **Error (`Fault`)**: Se gestionan las acciones para detener el inversor y manejar las condiciones de error.

En cada estado, se describen las acciones realizadas y las condiciones que pueden provocar una transición a otro estado, como la detección de errores o cambios en las condiciones de funcionamiento del inversor.

Figura 4.114: Diagrama de estados de la FSM para el control de un inversor.

`ERRORS.c/.h`

Los archivos `ERRORS.c` y su correspondiente encabezado `ERRORS.h` proporcionan las funciones necesarias para establecer, leer y eliminar errores del sistema.

En este archivo, se define una enumeración llamada `InverterError` que representa los diferentes estados de error del inversor. Cada error se representa como un bit en un número entero sin signo de 32 bits (`uint32_t`).

Posición	0	1	2	3	4	5	6	7	8	9
Código	POWER_FAULT	OVERTEMPERATURE_INV	OVERVOLTAGE	OVERCURRENT	OVERSPEED	UNDERVOLTAGE	CONTROL_FAULT	WARNING	OVERTEMPERATURE_MOT	FEEDBACK_FAULT
Ejemplo	0	0	0	0	0	0	1	0	0	1

Cuadro 4.8: Mapa de errores.

Por ejemplo, si los bits 0 y 3 están establecidos (valor binario 00 0000 1001 o valor decimal 9), significa que se ha producido un error en la etapa de potencia y otro de sobrecorriente.

En `ERRORS.c`, se implementan las siguientes funciones:

- `set_error`: Esta función establece un error específico en el campo de errores de una estructura de datos. Toma como argumentos un puntero a la estructura de datos y el error a establecer.
- `clear_error`: Esta función elimina un error específico del campo de errores de una estructura de datos. Toma como argumentos un puntero a la estructura de datos y el error a eliminar.
- `is_error_set`: Esta función verifica si un error específico está establecido en el campo de errores de una estructura de datos. Toma como argumentos un puntero a la estructura de datos y el error a verificar.

Estas funciones proporcionan una interfaz para manejar los errores del sistema de manera eficiente y modular. Al representar los errores como bits en un entero sin signo, se optimiza el uso de memoria y se facilita la manipulación y el análisis de los errores individuales.

`MEASUREMENTS.c/.h`

Los archivos `MEASUREMENTS.c` y su encabezado correspondiente `MEASUREMENTS.h` contienen la implementación y la interfaz para manejar las mediciones en el sistema.

En `MEASUREMENTS.h`, se definen algunas constantes relacionadas con ganancias y desplazamientos para corriente y voltaje, así como umbrales de falla para diversas condiciones del sistema. Además, se declaran las estructuras de datos necesarias

para almacenar las mediciones de los sensores y los valores de retroalimentación.

Las estructuras definidas son:

- `Encoder`: Estructura para la lectura del *encoder*, que almacena los valores de los canales A, B y Z, así como la velocidad angular eléctrica y la posición del rotor eléctrico.
- `Analog`: Estructura para las mediciones de los sensores analógicos de cada inversor, es decir, las corrientes de fase y la tensión del bus DC.
- `Feedback`: Estructura para los valores de retroalimentación, que contiene corrientes $d - q$, par calculado y velocidad medida.

En `MEASUREMENTS.c`, se implementan las funciones para manejar las mediciones, como la conversión de lecturas de ADC a medidas físicas, el cálculo de corrientes $d - q$ a partir de las mediciones de corriente de fase y el ángulo eléctrico, y la calibración de los *offsets* de los sensores de corriente.

`REFERENCE.c/.h`

Los archivos `REFERENCE.c` y su encabezado correspondiente `REFERENCE.h` proporcionan funciones para manejar la consigna de par de cada motor. En `REFERENCE.h`, se define una estructura de datos llamada `Reference`, que contiene los valores de consigna para las corrientes $d - q$ y el par.

Las funciones principales implementadas en `REFERENCE.c` son:

- `set_torque_direction`: Esta función ajusta la dirección del par en función de la dirección deseada del motor.
- `saturate_symmetric`: Esta función satura simétricamente un valor de referencia en función del valor máximo permitido.
- `limit_torque_to_prevent_overspeed`: Esta función actúa como una saturación de par, reduciendo el par para limitar la velocidad máxima del motor.
- `handle_torqueRef`: Esta función llama a las anteriores y las encadena para gestionar adecuadamente la consigna de par.

Además, se crea un *derating* lineal para la corriente, que limita el valor máximo de la consigna:

- `calculate_derated_current`: Calcula la corriente máxima basada en umbrales de temperatura.
- `derate_current_reference`: Limita la referencia de corriente basándose en las temperaturas del motor y del inversor.

Figura 4.115: *Derating* lineal de corriente en función de la temperatura, que entra en funcionamiento a partir de T_1 y si se llega a T_2, no se entrega corriente.

PCB_IO.c/.h

Los archivos `PCB_IO.c` y su encabezado correspondiente `PCB_IO.h` actúan como capa de abstracción del *hardware*. En `PCB_IO.h`, se definen macros para leer y controlar algunos GPIOs, así como enumeraciones y estructuras para manejar LEDs y direcciones de motores.

Las funciones principales implementadas en `PCB_IO.c` son:

- `handle_LED`: Esta función maneja los modos de parpadeo de los LEDs basándose en el modo actual del LED y un contador de milisegundos.
- `handle_direction`: Esta función lee el estado del interruptor de dirección (`DIR`) y actualiza las direcciones de los motores izquierdo y derecho en función de este estado.
- `enable_inverters`: Esta función lee el estado de la cadena de *shutdown* del vehículo y habilita o deshabilita los inversores en función de este estado y de una habilitación externa por *software*.

CONTROL.c/.h

Los archivos `CONTROL.c` y su encabezado correspondiente `CONTROL.h` proporcionan funciones para el lazo de control de cada inversor. Las funciones principales implementadas en `CONTROL.c` son:

- `calc_current_reference`: Esta función calcula las consignas de corriente $d - q$ de la misma manera que en las simulaciones de control. Se implementan las

trayectorias utilizando las ecuaciones analíticas de las mismas.

- `calc_current_loop`: Esta función ejecuta los lazos de corriente para los ejes d y q.
- `saturate_voltage`: Esta función satura la salida de los lazos de corriente para no exceder el voltaje máximo sintetizable.
- `calc_duties`: Esta función calcula la transformada inversa de Park y los ciclos de trabajo utilizando SVPWM.

Cabe destacar que este archivo hace uso de la librería PERGAMON del CITCEA-UPC, relacionando las diferentes partes del código con la simulación realizada anteriormente. Los bloques usados en la simulación contienen el mismo código que está escrito en estas librerías, agilizando mucho la integración.

`PWM.c/.h`

Los archivos `PWM.c` y su encabezado correspondiente `PWM.h` proporcionan funciones para controlar la salida PWM. Hace de capa de abstracción de los *timers*, ya que contiene funciones para habilitar y deshabilitar la salida PWM y para actualizar los ciclos de trabajo.

`CAN_e-Tech.c/.h`

Los archivos `CAN_e-Tech.c` y su encabezado correspondiente `CAN_e-Tech.h` proporcionan funciones para manejar la comunicación CAN con el vehículo. Las funciones principales implementadas en `CAN_e-Tech.c` son:

- `handle_CAN`: Esta función implementa la lógica para manejar mensajes CAN recibidos.
- `send_CAN_message`: Esta función prepara y envía un mensaje CAN utilizando información de `CAN1db.h`, que recoge la estructura de todos los mensajes del bus de CAN al cual estará conectado el inversor.

4.4.3. Implementación del *firmware*

Configuración del MCU

Como ya se ha mencionado, se usará CubeMX para configurar los periféricos del microcontrolador. Las configuraciones más importantes se detallan a continuación.

Reloj

Es crucial configurar adecuadamente el reloj del microcontrolador para que este

pueda medir tiempos de forma precisa. En este caso se usa un cristal de cuarzo externo de 20 MHz como referencia, y se trata internamente para lograr llegar a los 216 MHz máximos del microcontrolador.

Figura 4.116: Configuración del reloj del sistema.

GPIOs

Se configuran como entradas / salidas los pines individuales de cada señal, asignándoles un nombre descriptivo acorde con el esquemático de la placa de control.

Pin Name	GPIO mode	User Label
PA8	External Interrupt Mode with Rising edge trigger det...	TRIP_L
PB2	Output Push Pull	ENABLE_R
PB11	External Interrupt Mode with Rising edge trigger det...	Z_R
PC4	Input mode	SC_det
PC9	External Interrupt Mode with Rising edge trigger det...	TRIP_R
PD3	Input mode	DIR
PD4	Output Push Pull	LED_LEFT
PD5	Output Push Pull	LED_RIGHT
PD6	Output Push Pull	LED_ERR
PD15	External Interrupt Mode with Rising edge trigger det...	Z_L
PE7	Output Push Pull	ENABLE_L
PE14	Input mode	WRN_L
PE15	Input mode	WRN_R

Figura 4.117: Configuración de GPIOs.

ADCs

En el control del convertidor es importante que las lecturas analógicas sean capturadas en una ventana de tiempo específica, generalmente sincrónica con la frecuencia de conmutación. En este caso, se han configurado dos ADCs (uno por cada inversor) con la adquisición disparada con el desbordamiento de su respectivo *timer*. Se capturan cuatro canales por inversor, los tres primeros se corresponden con las corrientes de fase, y el último, con la tensión del bus DC. Un punto importante es que el ADC usa un *stream* de DMA para volcar la información directamente en la memoria en un *buffer*, aliviando algo de carga al MCU. Cuando se necesitan

las medidas, se leen directamente del *buffer*, que estará preparado con las últimas medidas tomadas.

ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of all conversions
ADC_Regular_ConversionMode	
Number Of Conversion	4
External Trigger Conversion Source	Timer 1 Trigger Out event
External Trigger Conversion Edge	Trigger detection on the rising edge
Rank	1
Channel	Channel 6
Sampling Time	3 Cycles
Rank	2
Channel	Channel 7
Sampling Time	3 Cycles
Rank	3
Channel	Channel 8
Sampling Time	3 Cycles
Rank	4
Channel	Channel 9
Sampling Time	3 Cycles

Figura 4.118: Configuración del ADC 2 por DMA, correspondiente al inversor izquierdo.

Dado que la temperatura tiene una dinámica muy lenta, no es necesario tomar medidas a la frecuencia de control. Por ello se enrutan las medidas de temperatura a un ADC separado y se leen de forma más lenta.

ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of all conversions
ADC_Regular_ConversionMode	
Number Of Conversion	4
External Trigger Conversion Source	Timer 6 Trigger Out event
External Trigger Conversion Edge	Trigger detection on the rising edge
Rank	1
Channel	Channel 10
Sampling Time	56 Cycles
Rank	2
Channel	Channel 11
Sampling Time	56 Cycles
Rank	3
Channel	Channel 12
Sampling Time	56 Cycles
Rank	4
Channel	Channel 13
Sampling Time	56 Cycles

Figura 4.119: Configuración del ADC para la adquisición de temperaturas.

Timers

Los periféricos más importantes son los *timers*, ya que el control es síncrono. Por ello se han configurado dos *timers*, uno para cada inversor. Las salidas PWM se controlan directamente con los *timers*, que a su vez, disparan la lectura de los ADCs. Los *timers* 1 y 8, encargados de los lazos de control izquierdo y derecho respectivamente, tienen una peculiaridad: pueden generar tres canales duplicados, donde tres de ellos mantienen su polaridad positiva y los otros tres son réplicas con polaridad invertida de los primeros. Además, estos periféricos facilitan la integración de características propias de una salida complementaria como la definición del tiempo muerto. La gestión se realiza de forma automática por parte del mismo *timer*.

Se ajustan las constantes correspondientes al periodo de conmutación y al tiempo muerto utilizando `#define TS` y `#define DT`, de manera que se puede modificar la frecuencia de conmutación muy fácilmente. Se configuran los *timers* en modo *up and down*, simulando una señal portadora triangular, de la misma manera que está configurada la conmutación en la simulación de PLECS. Si contara solamente hacia arriba o hacia abajo, la portadora sería una señal diente de sierra.

Figura 4.120: Comparación de las señales portadoras equivalentes. De arriba a abajo, triangular, diente de sierra y diente de sierra invertida.

Counter Settings	
Prescaler (PSC - 16 bits value)	0
Counter Mode	Center Aligned mode1
Counter Period (AutoReload Register - 16 b...	(216000000*TS)/2
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 16 bits value)	1
auto-reload preload	Enable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO	Update Event
Trigger Event Selection TRGO2	Reset (UG bit from TIMx_EGR)
Break And Dead Time management - BRK Configu...	
Break And Dead Time management - BRK2 Config...	
Break And Dead Time management - Output Confi...	
Automatic Output State	Disable
Off State Selection for Run Mode (OSSR)	Enable
Off State Selection for Idle Mode (OSSI)	Enable
Lock Configuration	Lock Level 1
Dead Time	DT*216000000

Figura 4.121: Configuración del *timer* 1, correspondiente al inversor izquierdo.

CAN

Para poder comunicarse externamente se usa el protocolo CAN. Por suerte, la configuración es muy sencilla. Se deben ajustar los parámetros para que la frecuencia de transmisión de datos (*baud rate*) sea de 500 kB/s exactamente, ya que el resto de ECUs en el monoplaza tienen el periférico configurado a esta frecuencia.

Bit Timings Parameters	
Prescaler (for Time Quantum)	27
Time Quantum	500.0 ns
Time Quanta in Bit Segment 1	2 Times
Time Quanta in Bit Segment 2	1 Time
Time for one Bit	2000 ns
Baud Rate	500000 bit/s
ReSynchronization Jump Width	1 Time
Basic Parameters	
Time Triggered Communication Mode	Disable
Automatic Bus-Off Management	Disable
Automatic Wake-Up Mode	Disable
Automatic Retransmission	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
Advanced Parameters	
Operating Mode	Normal

Figura 4.122: Configuración del periférico CAN.

Manejo de interrupciones

En un convertidor se debe tener control absoluto sobre *cuándo* se ejecuta *qué*. Esto quiere decir que el lazo de control debe seguir un orden de ejecución perfectamente definido. El orden se basa en las simulaciones realizadas en PLECS, que como ya se explicó, tienen una relación muy estrecha con la implementación del *firmware*. Estudiando [39] y [21], se decide seguir el siguiente orden en las interrupciones de control:

1. Se inicia la conversión de las lecturas analógicas de las corrientes y la tensión.
2. Cuando acaba la conversión, se evalúan los valores obtenidos para hacer saltar errores si estuvieran fuera de los límites. Se establece la tensión máxima sintetizable en el periodo.
3. Se calcula la consigna de corriente.
4. Se ejecutan los lazos de corriente.
5. Se satura la tensión de salida.
6. Se calculan los ciclos de trabajo y se actualiza el registro del *timer* de la salida PWM.

La conversión del ADC se inicia periódicamente con el *timer* correspondiente. El resto de tareas se implementan en una función vacía. Lo que se desea lograr se muestra en la figura 4.123.

Figura 4.123: Señal portadora (registro CNT del *timer*) durante 3 periodos de conmutación y los eventos asociados.

La implementación de código se maneja con la interrupción generada al final de las conversiones analógicas.

```
/**
* @brief Function to be executed every TS.
*
* This function is called by the ADC2 DMA handler every TS (
   ADC is triggered with TIM1).
*/
void tasks_critical_left(void){
        start_ticks = SysTick->VAL;

        // ADC
        get_currents_voltage(rawADC_left, &inverter_left.
           analog, &inverter_left.feedback, &inverter_left.
           errors, inverter_right.encoder.sinTheta_e,
           inverter_left.encoder.cosTheta_e);
        inverter_left.vsMax = 0.9F * inverter_left.analog.vDC
            * ISQ3; // Calculate max Vs voltage with a safety
            margin

        // FOC
        calc_current_reference(inverter_left.motor, &
           inverter_left.reference);

        // PIs and duty calc
        calc_current_loop(&inverter_left);
        saturate_voltage(&inverter_left);
        calc_duties(inverter_left.vd, inverter_left.vq,
           inverter_left.analog.vDC, inverter_left.encoder.
           sinTheta_e, inverter_left.encoder.cosTheta_e, &
           inverter_left.duties);

        update_PWM(inverter_left.htim, inverter_left.duties);


        elapsed_ticks = start_ticks - SysTick->VAL;

}
```

Esta función vacía se llama dentro de la interrupción del ADC, que salta cuando el DMA llena el *buffer* de datos, indicando que se ha completado la conversión.

```
/**
* @brief This function handles DMA2 stream2 global interrupt.
*/
void DMA2_Stream2_IRQHandler(void)
{
        /* USER CODE BEGIN DMA2_Stream2_IRQn 0 */

        /* USER CODE END DMA2_Stream2_IRQn 0 */
        HAL_DMA_IRQHandler(&hdma_adc2);
        /* USER CODE BEGIN DMA2_Stream2_IRQn 1 */
        tasks_critical_left();

        /* USER CODE END DMA2_Stream2_IRQn 1 */
}
```

Lazo de control

El algoritmo `calc_current_reference` calcula las referencias de corriente a partir de la consigna de par. Este algoritmo se encarga de calcular la trayectoria MTPA y limita la consigna de corriente a `isMaxRef`, que se satura usando el *derating* implementado anteriormente. La trayectoria MTPV no está implementada para optimizar el tiempo de cálculo debido a las características esperadas de los motores que se controlarán. Finalmente, se convierten las coordenadas polares de las referencias de corriente (`isRef` y `gammaRef`) a coordenadas cartesianas (`idRef` e `iqRef`).

También se deja un espacio para la implementación de debilitamiento de campo. Para implementarlo, se debe añadir el lazo de tensión para modificar `gammaRef` y la trayectoria VLE. Además se debería implementar una selección de trayectorias adecuada que permita trabajar en todos los puntos de operación.

Para conocer el tiempo de ejecución del algoritmo, y de cualquier parte del código, se puede utilizar el *timer* interno `SysTick` para contar los ciclos de reloj entre dos puntos del código. Se observó que es necesario activar la optimización de velocidad en las opciones del compilador para que los cálculos entren dentro del tiempo de conmutación de 25 µs. Se estaban obteniendo valores de unos 3000 ciclos antes de activar la optimización, y de 1800 después de activarla. Aún así, sería buena idea dejar más margen mediante un estudio de optimización del propio código, evitando cálculos innecesarios o proponiendo nuevas maneras de calcular funciones trigonométricas, que son las que más tiempo llevan.

Figura 4.124: Optimización de velocidad en el compilador.

5. Resultados y validación

5.1. Introducción

La validación es una etapa crítica en el proceso de desarrollo de cualquier proyecto, especialmente en un proyecto de la magnitud y complejidad del convertidor diseñado. La validación se encarga de verificar que todos los componentes y subsistemas del convertidor funcionen correctamente y cumplan con los requisitos y especificaciones establecidos previamente.

Dada la naturaleza multifacética del convertidor, que abarca desde aspectos eléctricos y electrónicos hasta aspectos de control y *firmware*, es esencial planificar una estrategia de validación exhaustiva y efectiva. En este sentido, se ha adoptado un enfoque metodológico basado en el modelo en V, que permite validar los subsistemas desde los más específicos hasta los más generales, siguiendo un flujo lógico y sistemático.

En este capítulo se desarrolla la parte derecha de la 'V', diseñando y ejecutando las pruebas a varios niveles. La estrategia de validación comenzará validando el *hardware*, desde los componentes individuales hasta las placas enteras, y continuará con una etapa de integración en la que se desarrolla el *firmware*. Cada bloque tiene sus propios desafíos y requisitos específicos, pero todos contribuyen al objetivo final de asegurar el funcionamiento correcto y robusto del convertidor. Las diferentes partes de la validación que se han encarado de formas también diferentes, atendiendo a las circunstancias y priorizando las buenas prácticas y la identificación de problemas a los resultados exitosos.

5.2. Validación de *hardware*

La parte más crítica en la validación del convertidor es la de *hardware* puesto que cada iteración cuesta tiempo y dinero. Por ello, son las pruebas de *hardware* las que se diseñan y ejecutan más meticulosamente. En particular, se prestó especial atención a la placa de potencia, puesto que es el diseño más complicado de los dos.

Inicialmente, se elaboraron hojas de cálculo detalladas para cada uno de los ensayos, abarcando desde la inspección visual y pruebas de subcircuitos, hasta la evaluación de sistemas algo más complejos. Sin embargo, a medida que avanzaba el proceso de validación y se integraban los subcircuitos unos con otros, se hizo evidente que era necesario un enfoque más dinámico y adaptativo. Esto se debió a que los problemas encontrados y las soluciones implementadas requerían una validación continua y menos estructurada, lo que permitía realizar ajustes y mejoras tiempo real.

5.2.1. Pre-inspección de la placa de potencia

Antes de realizar cualquier prueba funcional, se llevó a cabo una inspección visual exhaustiva de la PCB de potencia para identificar cualquier defecto físico o de manufactura. Esta inspección incluyó la verificación del acabado superficial, la serigrafía, el grosor del cobre en las capas y las dimensiones de la placa. En la primera iteración se encontraron algunos problemas menores, como una serigrafía invertida en los conectores del bus de continua, pero no afectaron las pruebas iniciales.

Se validó el grosor de cobre en las vías para aportar confianza a la hora de montar los componentes *press-fit*. Para ello, se extrajo una sección de la PCB que cortaba transversalmente una vía y se pulió. Posteriormente, se tomó una fotografía con un microscopio y se hicieron medidas digitales.

Figura 5.1: Extracción de una muestra de la PCB.

Figura 5.2: Micrografía de una vía de la placa de potencia. Se pueden apreciar el grosor de cobre de las capas y de la vía.

Se pudo comprobar que el grosor de las capas era correcto (70 µm), y que el de las vías estaba dentro de la tolerancia establecida (de 25 a 50 µm).

5.2.2. Primer contacto con la placa de potencia

Durante la primera interacción con la placa de potencia, se realizaron diversas pruebas para validar el funcionamiento de los distintos componentes y subcircuitos de forma aislada. Principalmente se busca validar una funcionalidad básica de cada uno de forma separada, de manera que si se encuentran problemas de integración se sepa que son de integración y no de los componentes o circuitos individuales. Se ordenaron estas pruebas de "dentro hacia afuera", empezando por el circuito de *driving* y acabando con la interacción externa del mismo. También se validó la adquisición de variables analógicas y se probó la funcionalidad básica del circuito de descarga.

Alimentación del *gate driver*

Para validar la alimentación del *gate driver* se separó en una verificación del LDO para la tensión negativa por separado, y en otra para validar el DC-DC.

En primer lugar se pobló únicamente el propio LDO con los componentes necesarios para su funcionamiento. Se anotó el primer error de la placa, que consiste en una equivocación en el mapeado de los pines del esquemático al *footprint*. De todas formas, se realizó un *dead-bugging* para conectar el componente de forma correcta y poder verificar el circuito de forma exitosa.

Figura 5.3: LDO *dead-bugged* y con resina para evitar daños con las siguientes pruebas.

- **Valor esperado**: -4 V $\leq$ V $\leq$ -3,5 V
- **Resultado**: -3,78 V

La dificultad de realizar este *rework* fue motivo suficiente para corregir el diseño, pero se esperó a finalizar la mayoría de pruebas básicas antes de pedir placas nuevas.

Después de verificar el funcionamiento del LDO se probaron los DC-DCs aislados. Funcionaron según lo previsto, aunque se notó que se calentaban ligeramente, incluso en vacío (sin carga). Esto se debe a su curva de rendimiento, que con cargas bajas es muy poca. Además se anotaron variaciones en las tensiones de salida, pero resultó que eran función de la carga, y está explicado en la hoja de datos.

Figura 5.4: Curva de rendimiento del DC-DC aislado [24].

Figura 5.5: Variación de la tensión de salida en función de la carga [24].

- **Valor esperado**: 15,5 V $\leq$ VDD $\leq$ 14,5 V ; -4 V $\leq$ VEE $\leq$ -3,5 V
- **Resultado**: 15,94 V ; -3,78 V

Funcionalidad del *gate driver*

Para validar la funcionalidad básica del *gate driver*, se montó un módulo de potencia y los dos *gate drivers* necesarios para controlar los dos MOSFETs.

- ***OK* y *TRIP***: Se poblaron los componentes necesarios y se alimentó la placa desde el lado de *LV*. Se midieron los nodos *TRIP* y *OK*. Ambas señales tenían un valor de 5 V, confirmando su correcto funcionamiento sin errores. Posteriormente se forzaron fallas en el lado del semiconductor y ambas bajaron a 0 V.

- **MOSFETs apagados en reposo**: Se montaron las resistencias de puerta y se verificó el estado de ambos MOSFETs midiendo la resistencia entre *drain* y *source*. Ambos dispositivos se encontraban en estado *OFF*, confirmando que no se encendían.

- ***ENABLE* no enciende los dispositivos**: Se introdujo una señal de 3,3 V el nodo *ENABLE*, simulando la acción del microcontrolador. De nuevo, ambos dispositivos se encontraban en estado *OFF*, lo cual es el resultado esperado, ya que ambas señales PWM estaban en estado bajo, es decir, consignando que los MOSFETs estuvieran abiertos.

- **Medición de temperatura**: Se verificó la capacidad del *gate driver* de tomar una medida analógica, en concreto, de la NTC interna del semiconductor. Se contrastaron las medidas obtenidas con los cálculos previos, arrojando resultados satisfactorios.

Figura 5.6: Medición de temperatura ambiente. Se puede ver la señal de salida sin filtrar (PWM) y filtrada.

Sensado de corriente

Para validar el funcionamiento del sensado de corriente, se realizaron las siguientes pruebas:

- **Standby**: Se esperaba una salida de 2,5 V con un margen de error de ± 5 mV. La salida medida fue de 2,499 V, lo que se considera dentro del rango aceptable.

- **Operación básica**: Se esperaba una salida de 2,5625V para una corriente de +5A y 2,4375V para una corriente de -5 A, con un margen de error de ± 5 mV. Las mediciones obtenidas fueron de 2,557 V para +5 A y 2,436 V para -5 A, ambas dentro del rango esperado.

Descarga

Para verificar la funcionalidad de la descarga, se realizaron las siguientes pruebas:

- **Descarga pasiva**: Se precalculó una descarga de 24 V a 10 V en 35 segundos usando un 20 % de la capacidad del bus. La medición obtenida fue de 5,5 segundos,

debido a que no se tuvo en cuenta el resto de conexiones entre los terminales positivo y negativo. La resistencia aparente entre estos terminales era más baja, principalmente por el divisor de tensión usado para tomar la medida de voltaje del bus. El divisor de voltaje para la adquisición ya es de 6·68 kΩ+4,7 kΩ = 412,7 kΩ, y está en paralelo con la resistencia de 2 MΩ, lo que resulta en una resistencia equivalente de

$$\frac{1}{\frac{1}{2\ \mathrm{M\Omega}} + \frac{1}{412{,}7\ \mathrm{k\Omega}}} = 342\ \mathrm{k\Omega},$$

lo que equivale a aproximadamente 6 segundos de descarga. Al agregar cualquier otra resistencia en paralelo, como la de los propios semiconductores u otros componentes, los cálculos coinciden.

Figura 5.7: Descarga pasiva a 24 V con 20 µF.

- **Control de la descarga**: Se aplicó un voltaje de 20 V en el nodo *SC_END* y se suministraron 24 V entre *TS+* y *TS-*. Se esperaba que la diferencia de potencial entre *TS+* y *TS-* fuera de 10 V 0,35 segundos después de retirar el voltaje, que coincidió perfectamente con el cálculo previo.

Figura 5.8: Descarga activa a 24 V con 20 µF. Se puede ver como a 10 V (la tensión del zener) se desactiva la descarga activa.

Sensado de voltaje

Para verificar el sensado de voltaje, se llevaron a cabo las siguientes pruebas:

- **Divisor de voltaje**: Se aplicaron 24 V entre *TS+* y *TS-*. Se capturó el voltaje en el divisor de tensión. La tensión esperada era de $0{,}011388 \cdot 24$ V $= 0{,}2733$ V con un margen de error de ± 1 mV. La medición obtenida fue de 0,272 V, dentro del rango aceptable.

- **Alimentación aislada**: Se montó el convertidor DCDC501 y se alimentó la placa desde el lado LV para medir el voltaje entre *+5_TS* y *TS-*. Se esperaba una lectura de 5V con un margen de error de $\pm 0{,}5$ V. Sin embargo, la medición fue de 5,5 V, indicando que se deberá ajustar el divisor de voltaje para el umbral de detección debido a la inexactitud.

- **Detección de voltaje**: Se aplicaron 50 V entre *TS+* y *TS-*, y se midió *TSAL_ON*. Se repitió la prueba con 60 V. Se esperaba que *V_TSAL* fuera de 677 mV y que *TSAL_ON* fuera 5 V con 50 V de tensión de bus y 0 V con 60 V. Sin embargo, la medición fue incorrecta, ya que el umbral aproximado para la transición fue de 63 V. Se encontró que la causa podría estar la tolerancia del divisor de resistencia y el hecho de que la salida del convertidor DC-DC era medio voltio superior.

- **Sensado de voltaje**: Se aplicaron 13 V entre *TS+* y *TS-*. Se capturó el valor entre *VDC_sns+* y *VDC_sns-*. Se esperaba que la diferencia de potencial fuera de 0,0495 V. La medición obtenida fue de 0,05 V, dentro del rango aceptable.

Figura 5.9: Circuitos evaluados.

Resumen de errores encontrados y revisión de la PCB

Se encontraron los siguientes errores durante el proceso de prueba y montaje:

1. **Problema:** El serigrafiado de los conectores de alimentación (+ y -) está invertido. Esto no afecta las pruebas.
   Solución propuesta: Nombrar los conectores según corresponda y cambiar los textos del serigrafiado. Pedir nuevas PCBs.

2. **Problema:** Se conectó accidentalmente el nodo de 5 V de alimentación a *SC_-END* (20 V), lo que hizo que todos los componentes alimentados a 5 V murieran.
   Solución propuesta: Añadir protecciones a la alimentación de 5 V.

3. **Problema:** El *footprint* del LDO tiene los pines 4 y 5 intercambiados (-*VEE*, *FB*).
   Solución propuesta: Montar el LDO verticalmente con conexiones flotantes cruzadas.

Para realizar el resto de pruebas se pidieron nuevas placas de potencia, con las siguientes modificaciones:

- Protecciones para la alimentación de 5 V.
- Se intercambiaron los pines 4 y 5 en los LDO de los *gate drivers*.
- Se intercambiaron *MP+* y *MP-* junto con su serigrafiado.
- Se añadieron *testpoints* para la referencia de los sensores de corriente.
- Se añadieron *testpoints* para *VDC_sns+* y *VDC_sns-*.
- Se renombraron los *testpoints* en el esquemático de los *gate drivers* para agilizar

las pruebas.

- Se añadieron varios textos e indicaciones en el serigrafiado.

5.2.3. Conmutación de una rama como DC-DC síncrono

Con todos los subcircuitos validados, el siguiente paso fue analizar la conmutación. Se utilizó una sola rama de las tres, creando una topología de *buck* síncrono. Se ensayó con una carga R-L de 7 Ω y 1 mH. Lo que se pretende evaluar con estas pruebas es el correcto funcionamiento del circuito de *driving* y el *overshoot* entre *drain* y *source* de los MOSFETs.

Figura 5.10: Topología de *buck* síncrono.

Figura 5.11: Carga R-L, fuente de alimentación y osciloscopio.

Se conectó la placa de potencia con una placa de evaluación ya que en este punto todavía no se había fabricado la placa de control. De esta manera, se generó una pareja de señales PWM complementarias a 50 kHz y 50 % de ciclo de trabajo con 150 ns de tiempo muerto.

Figura 5.12: Placa de evaluación Nucleo conectada a la placa de potencia.

Usando una fuente de tensión regulable, se pudo controlar la tensión de entrada, y dado que se fija el ciclo de trabajo al 50 %, y que la carga es fija, se pudo variar la tensión de salida modificando la tensión de entrada. Conectando y desconectando la carga, se pudo probar la conmutación tanto en vacío como con algo de corriente.

En primer lugar, sin tensión en el bus de continua, se verificó que tanto las señales PWM entrantes a los *gate drivers* como las señales en la puerta de los transistores eran adecuadas en cuanto a niveles de tensión, frecuencia y polaridad.

Figura 5.13: Señales PWM a la entrada de los *gate drivers*, producidas por la placa de evaluación.

Figura 5.14: Tensión *gate-source* de uno de los dos MOSFETS.

El siguiente paso fue verificar la tensión entre *drain* y *source*, y dado que se trata de una medida que tiene implicaciones con los parásitos del circuito, se usó una sonda con conexión *pigtail* en la masa. El objetivo es minimizar el área entre la conexión a masa y la señal medida, ya que de esa forma se minimiza la inductancia equivalente total. Una medida con una inductancia alta falsearía el valor de *overshoot*.

Figura 5.15: Sonda con *pigtail*.

Figura 5.16: Medición de la tensión *drain - source* (fotografía tomada posteriormente al ensayo).

Para poder observar el rizado provocado por la carga inductiva se decidió bajar la frecuencia de conmutación a 10 kHz. Posteriormente, se hizo una primera prueba con 10 V de entrada, y se observó un *overshoot* muy grande.

Figura 5.17: *Overshoot* observado con 10 V de entrada y 0 A de salida (en vacío).

Figura 5.18: *Overshoot* observado con 10 V de entrada y 0,75 A de salida (con la carga R-L).

Dado que este *overshoot* es aproximadamente lineal con la corriente de salida y la tensión de entrada, se creó un modelo lineal para predecir el *overshoot* a las especificaciones máximas del convertidor. Para ello se tomó un dato más a 20 V de entrada y sin carga. El resultado del modelo lineal se muestra a continuación.

Modelo		$Overshoot = a \cdot V_{\mathrm{DC,in}} + b \cdot I_{\mathrm{out}} + c$	
Coeficientes			
a		1,5	
b		6,6667	
c		$3{,}9721 \cdot 10^{-15}$	
Datos medidos y predicción			
$V_{\mathrm{DC,in}}$	I_{out}	*Overshoot* medido	Predicción de *overshoot*
10 V	0 A	15 V_p	15 V_p
20 V	0 A	30 V_p	30 V_p
10 V	0,75 A	20 V_p	20 V_p
Predicción de *overshoot* a 600 V y 80 A: 1433 $\mathbf{V}_p$			

Cuadro 5.1: Resumen de resultados del modelo de *overshoot*.

Figura 5.19: Modelo lineal (plano) junto a los datos tomados (puntos azules).

La predicción de 1433 V_p es muy preocupante, ya que en primer lugar, los semiconductores serían los primeros en dañarse ya que solo aguantan 1200 V, y además, tensiones tan altas podrían provocar rupturas dieléctricas. Por tanto, es necesario reducir este *overshoot*. La solución consiste en incorporar desacoplos de la conmutación entre los terminales positivo y negativo, lo más cerca posible de cada módulo *half-bridge*. Interesa tener el máximo de capacidad con una resistencia serie equivalente muy baja, con lo que se descartan los condensadores electrolíticos. Se escogieron condensadores cerámicos puesto que tienen una densidad de capacidad bastante grande, mucho más alta que los condensadores de película. Sin embargo, los cerámicos tienen una pérdida de capacidad por *DC bias* muy agresiva, es decir, la capacidad real es mucho baja cuando se sube la tensión a la que se somete el condensador. Por suerte, existen gamas de condensadores pensados para estas aplicaciones que ofrecen un *derating* por *DC bias* bastante bajo. En este caso se escogieron los 2220Y1K00104KZT de la gama Hiteca de Knowles Syfer.

Figura 5.20: Comparativa de la pérdida de capacidad de un condensador con aislante X7R estándar vs. el aislante usado en la gama Hiteca de Knowles Syfer [14].

Estos condensadores son muy caros, ya que no solo el aislante es mucho mejor, sino que pueden aguantar hasta 1 kV e integran una capacidad de 100 nF en un encapsulado 2220. Se instalaron dos condensadores en cada módulo de potencia. Para esta aplicación, tan solo se perdería un 25 % de la capacidad, es decir, a 600 V no se tendrán 200 nF si no 150 nF, que ya es bastante. El *rework* quedó de la siguiente manera:

Figura 5.21: Instalación de los condensadores de desacoplo usando los pines de los módulos de potencia.

Se volvieron a realizar las pruebas con 10 V de entrada, y se observó una reducción muy drástica del *overshoot*, verificando que es una solución adecuada.

Figura 5.22: *Overshoot* observado con 10 V de entrada y 0 A de salida (en vacío).

Figura 5.23: *Overshoot* observado con 10 V de entrada y 0,75 A de salida (con la carga R-L).

Se creó un nuevo modelo lineal para realizar la predicción de *overshoot* a las especificaciones máximas, y en este caso, se tomaron muchos más puntos, siendo la fuente de alimentación con la que se ensayó el factor limitante para no poder tomar más.

Figura 5.24: *Overshoot* observado con 90 V de entrada y 6,2 A de salida (con la carga R-L).

Modelo	$Overshoot = a \cdot V_{\mathrm{DC,in}} + b \cdot I_{\mathrm{out}} + c$		
Coeficientes			
a	1,1463		
b	0,42886		
c	−1,3001		
Datos medidos y predicción			
$V_{\mathrm{DC,in}}$	I_{out}	*Overshoot* medido	Predicción de *overshoot*
10 V	0 A	10,8 V_p	10,1626 V_p
10 V	0,75 A	11,1 V_p	10,4843 V_p
20 V	0 A	21,3 V_p	21,6254 V_p
20 V	1,5 A	21,5 V_p	22,2687 V_p
30 V	0 A	31,7 V_p	33,0882 V_p
30 V	2,25 A	32,3 V_p	34,0531 V_p
40 V	0 A	45,9 V_p	44,551 V_p
40 V	3 A	46,9 V_p	45,8375 V_p
80 V	5,5 A	92,6 V_p	92,7608 V_p
90 V	0 A	102,5 V_p	101,8648 V_p
90 V	6,2 A	104,9 V_p	104,5238 V_p
120 V	0 A	135,6 V_p	136,2532 V_p
150 V	0 A	172 V_p	170,6415 V_p
180 V	0 A	205 V_p	205,0298 V_p
200 V	0 A	227 V_p	227,9553 V_p
Predicción de *overshoot* a 600 V y 80 A: 720,775 V_p			

Cuadro 5.2: Resumen de resultados del modelo de *overshoot*.

Figura 5.25: Modelo lineal (plano) junto a los datos tomados (puntos azules).

Los resultados fueron muy positivos porque el *overshoot* de 720,775 V_p es totalmente asumible. A estas alturas no se pudo pedir otra iteración de la PCB puesto que hubiese sido muy costoso, pero igualmente se actualizaron los esquemáticos con los desacoplos en los terminales de los semiconductores.

Figura 5.26: Esquemático de la etapa de potencia actualizado.

5.2.4. Pruebas térmicas y de alta tensión

La última prueba que se hizo a la placa de potencia fue una serie de ensayos de descarga, a cada vez más tensión. El objetivo de este test es asegurar que no se produce ninguna ruptura dieléctrica y que las resistencias de descarga son capaces de aguantar la tensión máxima permanentemente.

Para realizar estos ensayos fue necesario usar dos autotransformadores, uno de ellos aislado. El primero, monofásico, se conecta a la red normal, y el otro, a la red trifásica. Se rectificó la salida de cada uno con un puente de diodos (uno monofásico y uno trifásico), y se conectaron los dos rectificadores en serie para obtener una

tensión continua de hasta 800 V. Se usaron un par de contactores, uno para realizar la precarga del bus de continua y otro para conectarlo directamente al rectificador.

Para evaluar las temperaturas se dispuso de una cámara térmica, que permitió distinguir con claridad qué partes de la PCB se calentaban más. Para capturar la tensión se usó una sonda diferencial de 1400 V.

Figura 5.27: Vistas del *setup* para las pruebas de descarga.

Los ensayos se realizaron de la siguiente forma:

1. Se precarga el bus de continua con una resistencia de externa.
2. Se mantiene hasta lograr el estado estacionario térmico, evaluado con una cámara térmica. En este punto, la descarga activa está desactivada y se evalúa la descarga pasiva.
3. Se conecta la descarga activa pero se deja el contactor conectado de manera que el bus no se descarga. Se espera hasta alcanzar el estado estacionario térmico.
4. Se abren los contactores y se descarga el bus.

A continuación se muestra el ensayo más exigente, una descarga a 600 V.

Figura 5.28: Descarga de 600 V a 60 V con 100 µF de capacidad en el bus de continua, en un tiempo de 4,5 segundos.

Figura 5.29: Captura de la cámara térmica tras alcanzar el estado estacionario térmico a 600 V.

Se puede observar un pico de temperatura de 100 °C, y la temperatura ambiente era de 25 °C aproximadamente. Cada resistencia disipa $\frac{(600\ \text{V})^2}{470\ \text{k}\Omega} = 0{,}766$ W, lo cual se correspondería con una resistencia térmica de aproximadamente $\frac{100\ °\text{C} - 25\ °\text{C}}{0{,}766\ \text{W}} \approx$ 100 K/W. Este valor es muy razonable para una resistencia SMD sin *pad* térmico ni mucha consideración en el *layout*, de acuerdo con [33].

5.2.5. Validación de la placa de control

La validación de la placa de control se llevó a cabo tras la instalación inicial del *hardware*, que incluyó el microcontrolador y los componentes esenciales para su funcionamiento básico. El *hardware* de esta placa es relativamente sencillo en comparación con la placa de potencia, así que se destinó más tiempo a programar que a mirar detenidamente los subcircuitos de esta placa. Aunque el proceso de validación se realizó de manera preliminar y parcial, se abordaron varias áreas clave:

Alimentación

La placa se alimenta exclusivamente del sistema de baja tensión del monoplaza, con una tensión nominal entre 20 V y 30 V. Inicialmente, se tenía previsto implementar un convertidor para obtener un bus de 5 V, sin embargo, debido a la complejidad y dificultades de soldadura del convertidor, se optó por utilizar una fuente de alimentación externa de 5 V para realizar las pruebas iniciales. De todas maneras, el convertidor es de Recom, una marca con la que el equipo ha trabajado varios años y se confía en la calidad de sus convertidores. Sin embargo, se vio que la bobina del filtro Pi (L101) no tiene el *rating* de corriente suficiente. Se deberá escoger una bobina que pueda soportar más corriente en una siguiente versión de la placa.

MCU

El MCU seleccionado fue el STM32F777VI, que se integra como el bloque central de la placa. La prioridad fue lograr programarlo por primera vez, que se logró con éxito no mucho después de montarlo. Por suerte, antes de montarlo por primera vez, se detectaron algunos errores en el diseño inicial de la PCB, con lo que se evitó soldarlo en esta primera versión. Estos errores son:

- El pin *BOOT0* (pin 94 del MCU) debe estar conectado a GND utilizando una resistencia de 10 kΩ o un valor similar.
- Las pistas del bus I^2C de la EEPROM deben tener resistencias de *pull-up* cuyo valor debe determinarse según la velocidad del bus.
- El filtro de alimentación debería estar en el bus de 5 V en vez de en el de 20-30 V.
- El pin *Vbat* debe estar conectado a 3.3 V.

Para abordar estos errores, se realizaron las siguientes modificaciones en *Inverter_-Control*:

- Se agregaron resistencias de *pull-up* y *pull-down* para controlar *BOOT0*.

- Se añadieron resistencias de *pull-up* para las líneas I^2C.
- Se eliminó la conexión *LV+_sns* y se conectó *Vbat* a 3.3 V.
- Se realizaron cambios en el color y nombres de los LEDs para mayor claridad.
- Se trasladó el filtro de alimentación al bus de 5 V.
- Se realizaron ajustes en el serigrafiado.

Estas modificaciones corrigieron los errores detectados en *Inverter_Control* y permitieron un desarrollo adecuado.

Retroalimentación

Aunque se han integrado conexiones para el *encoder* incremental y el *front-end* analógico de la lectura de temperatura del motor, las pruebas se han limitado principalmente a las lecturas en el MCU. Esto se debe a que no se ha dispuesto del motor con el que se usará este inversor, pero por suerte los subcircuitos son muy simples y se ha podido verificar la lectura desde el MCU con señales externas.

Front-end **analógico de la placa de potencia**

El *front-end* analógico, que trata las señales de las placas de potencia para adaptarlas a los rangos de tensión admitidos por los ADCs del MCU, presenta algunos desafíos. Mientras que las lecturas de corriente y temperatura se han validado con éxito, la lectura de tensión mediante un amplificador diferencial no funcionó como se esperaba. La integración del circuito integrado no fue buena, y, aunque fue simulado mediante SPICE adecuadamente, los resultados no coincidieron con lo esperado. Se están considerando otras posibilidades, como obtener la lectura de tensión a través del bus CAN, que podría ser proporcionada directamente por la batería del vehículo. Esto sería viable ya que realmente esta medida sirve como límite para evitar sobremodular, y la dinámica eléctrica de una batería es muy lenta. De todas maneras, se recomienda efusivamente revisar la implementación para futuras iteraciones del convertidor y aplicar las correcciones necesarias.

5.3. Validación de *firmware*

5.3.1. Conmutación de las tres ramas

Tras configurar correctamente los periféricos necesarios, el siguiente paso es la conmutación de las tres ramas simultáneamente, siendo el siguiente paso desde los ensayos como *buck* síncrono con una sola rama. Realmente lo único necesario fue

portar el *firmware* de la placa de evaluación a la placa de control y activar la salida de los otros dos canales (y sus respectivos negados).

5.3.2. Lazo abierto de tensión con carga R-L

Posteriormente se implementó la modulación SVPWM y las transformadas inversas para poder consignar una tensión alterna. Usando una fuente de tensión limitada por corriente como bus de continua, se conectó una carga R-L trifásica estática a los terminales de las fases del convertidor. El valor de las inductancias utilizadas es de unos 500 µH, y el de las resistencias de 0,5 Ω. El valor real de la carga resistiva, sin embargo, se sitúa entorno a los 0,8 Ω si se tienen en cuenta las resistencias de los cables y conexiones.

Figura 5.30: Carga R-L trifásica utilizada.

Figura 5.31: *Setup* del ensayo con carga R-L.

En primer lugar, y con una frecuencia de conmutación muy baja (5 kHz), se sintetizó una frecuencia de 500 Hz (relación de portadora a modulada de 10). Con esto se puede verificar el ensanchamiento y estrechamiento de los tres canales PWM reflejando las tres señales senoidales desfasadas 120 grados, generadas por la modulación SVPWM. Además, al ser una frecuencia de conmutación tan baja, se puede apreciar el rizado en la corriente de salida.

Figura 5.32: Lazo abierto de tensión. Señales PWM a la entrada del *driver* y corriente de una fase.

Se puede validar si la tensión sintetizada es correcta según la corriente de la carga.

Se cambió la frecuencia sintetizada a 100 Hz, se ajustó la fuente de tensión a 5 V y se consignó una en el eje q de dos tercios de la máxima disponible, es decir, $(v_d, v_q) = (0, \frac{2}{3}\frac{V_{\text{DC}}}{\sqrt{3}}) = (0, \frac{2}{3}\frac{5\text{ V,DC}}{\sqrt{3}})$. Se midieron las corrientes de fase para verificar que se estaba modulando correctamente, y se volvió a subir la frecuencia de conmutación para evitar ver el rizado en la carga.

Sabiendo que la tensión en la carga es

$$v_{\text{fase-neutro, pico}} = i_{\text{fase, pico}} \cdot (R + L \cdot \omega)$$

y que la tensión modulada es de

$$(v_d, v_q) = (0, \frac{2}{3}\frac{5\text{ V,DC}}{\sqrt{3}})\text{ V}$$

$$\therefore v_s = v_q = v_{\text{fase-neutro, pico}} = \frac{2}{3}\frac{5\text{ V,DC}}{\sqrt{3}}\text{ V ,}$$

se puede calcular que la corriente esperada es de

$$i_{\text{fase, pico}} = \frac{v_{\text{fase-neutro, pico}}}{(R + L \cdot \omega)} = \frac{\frac{2}{3}\frac{5\text{ V,DC}}{\sqrt{3}}\text{ V}}{0{,}8\ \Omega + 500\mu\text{H} \cdot 2\pi 100\text{ Hz}} = 1{,}73\text{ A ,}$$

que encaja con el valor medido de aproximadamente 1,7 A.

Figura 5.33: Corrientes de las tres fases de la carga R-L. El canal CH3 muestra un escalado incorrecto, que debería ser igual al de los otros dos canales visibles.

5.3.3. Lazo abierto de tensión con un PMSM

Con las tensiones trifásicas correctamente sintetizadas, se probó un lazo abierto de tensión con un PMSM.

Figura 5.34: Motor utilizado en los ensayos [20].

Parámetros del motor			
Parámetro	Valor	Unidades	Descripción
pp	4	ad	Número de pares de polos
λ_m	0,13391	mWb	Flujo magnético de los imanes permanentes
L_d	2,91	mH	Inductancia en el eje d
L_q	2,91	mH	Inductancia en el eje q
R_s	1,95	Ω	Resistencia de fase del estátor
$\omega_{\text{m,máx}}$	8500	RPM	Velocidad angular máxima del motor
$T_{\text{em,máx}}$	10	$\text{N} \cdot \text{m}$	Par máximo del motor
$V_{\text{DC,máx}}$	450	V	Voltaje máximo DC
$I_{\text{s,máx}}$	60	A	Corriente máxima en los ejes d-q

Cuadro 5.3: Parámetros del motor utilizado en los ensayos.

Figura 5.35: *Setup* del ensayo con el motor.

Se procedió de manera muy similar al ensayo de la carga estática, aunque en este caso fue necesario implementar una rampa de frecuencia para lograr que el motor girara. El resto fue idéntico, y se tomó una captura de las corrientes de fase.

Figura 5.36: Corrientes de fase del motor.

Como se puede observar, las formas de onda no son perfectamente senoidales, y este hecho se asocia a los armónicos que pueda presentar el bobinado del estátor. La frecuencia sintetizada fue de tan solo 5 Hz, atendiendo a un análisis de estabilidad [22].

6. Consideraciones finales

Impacto ambiental

El presente proyecto tiene un impacto ambiental que se concentra principalmente en la fabricación del *hardware*. El control y *firmware* del inversor tienen un impacto ambiental prácticamente nulo, la producción y ensamblaje de los componentes electrónicos son aspectos críticos a considerar.

Fabricación de PCBs

La fabricación de placas de circuito impreso es un proceso que requiere el uso de materiales como fibra de vidrio, cobre y resinas, cuya extracción y procesamiento pueden generar impactos ambientales significativos, especialmente si no se gestionan de manera adecuada los residuos. La generación de desechos y la emisión de productos químicos durante la fabricación de PCBs pueden contribuir a la contaminación del suelo, agua y aire. En este caso, las PCBs presentaban apilados bastante convencionales, con lo que la producción está más optimizada que para apilados más complejos con mayor número de capas o un laminado de cobre más gordo.

Montaje y selección de materiales

El proceso de montaje de los componentes electrónicos en las PCBs implica operaciones como la soldadura, que utiliza materiales como el estaño. Los estaños que se han usado contenían plomo, puesto que ayuda a bajar la temperatura de fusión del estaño para facilitar la soldadura. Además, la selección de materiales para los componentes electrónicos también puede influir en el impacto ambiental del proyecto. Es importante considerar la procedencia de los materiales y su impacto ambiental durante su ciclo de vida. De todas maneras, el proyecto es un prototipo y no existe ninguna intención de comercialización, con lo que el impacto global es muy bajo. Si se tratara de industrializar el diseño para una producción en serie, deberían considerarse las normativas de allá donde se quisiera distribuir. Para comercializar un equipo electrónico en Europa, todos los componentes electrónicos que lo forman deben cumplir la directiva RoHS.

Gestión de residuos

Si se tratara de escalar el proyecto a una producción, la disposición de los residuos generados durante el proceso de fabricación y ensamblaje de los inversores es un

aspecto crucial a tener en cuenta para mitigar el impacto ambiental. La correcta gestión de residuos electrónicos es fundamental para prevenir la contaminación, así como para evitar la liberación de sustancias tóxicas en el medio ambiente. La implementación de sistemas de reciclaje y la adopción de prácticas de disposición adecuadas pueden ayudar a reducir el impacto ambiental de esta etapa, y esto se debería consultar con los fabricantes y ensambladores que se encarguen de la producción. A día de hoy, la mayoría de empresas del sector cumplen con estándares altos en cuanto a la gestión de los residuos, especialmente en Europa.

Transporte

El transporte de los materiales y componentes utilizados en la fabricación y ensamblaje de los inversores también puede tener un impacto ambiental significativo. Las emisiones de gases de efecto invernadero generadas por los vehículos de transporte son quizás la mayor parte del impacto ambiental generado por este proyecto. Ya que la mayoría de componentes se han pedido a modo de muestras y se han realizado varios pedidos extras, el transporte de los componentes y las placas seguramente ha supuesto una cantidad de emisiones descomunal. Además, la mayoría de componentes y todas las placas vinieron de China, siendo una ruta muy larga que contribuye a la huella de carbono que este proyecto ha dejado.

Consumo energético

Al hablar de consumo energético en este apartado se hace referencia a la electricidad consumida para el diseño y fabricación del prototipo. Es muy difícil cuantificar la energía utilizada ya que no solamente se tienen que tener en cuenta los equipos utilizados de primera mano, si no que sería necesario tener en cuenta el consumo de energía asociado con la fabricación de los componentes electrónicos, la operación de maquinaria y equipos, así como el uso de sistemas de climatización e iluminación en las instalaciones de producción. El proceso de diseño y desarrollo del prototipo también puede requerir el uso de herramientas y *software* que consumen energía eléctrica, como ordenadores, estaciones de trabajo y equipos de prueba.

Presupuesto

Costes de ingeniería

Para estimar los costes de ingeniería, se consulta la tabla salarial [28], donde un graduado en ingeniería técnica debería cobrar al menos **9.58 € la hora** por ser del segundo nivel salarial. El tiempo total invertido en ingeniería se puede estimar sabiendo que se trabajó una media de 30 horas a la semana desde septiembre hasta la fecha de entrega, lo que equivale a unas **1140 horas** de diseño, desarrollo y documentación.

Desarrollo del control

De todas las horas invertidas, se estima que un 25 % de las mismas fueron destinadas al estudio y diseño del control. Por lo tanto, se calculan 2730,3 € para los costes humanos en este aspecto. Además, se usaron MATLAB, Simulink y PLECS, cuyas licencias cuestan 119 € [18], 35 € [18] y 1400 € [26] respectivamente. En total, los costes para desarrollar el control ascienden a (2730,3 + 119 + 35 + 1400) € = **4284,3 €**.

Desarrollo del *hardware*

Se estima que se invirtió un 40 % del tiempo total en el diseño, prototipado y validación del *hardware*, lo que equivale a 4368,48 € . Para diseñar las PCBs se usó Altium Designer, y una licencia anual para el mismo está situada en los 4425 € [1]. Se usó también Solidworks, cuya licencia cuesta unos 2600 € [9]. Los costes de desarrollo de *hardware* suman **11393,48 €**.

Desarrollo del *firmware*

Para desarrollar el *firmware* se dedicó un 20 % del tiempo aproximadamente, y eso se traduce a **2184,24 €**. No se usó ningún programa de pago por lo que ese precio es todo lo que costaría esta parte del desarrollo.

Documentación

La redacción de este trabajo junto al resto de documentación generada llevó el 15 % de tiempo restante, es decir, **1638,18 €**. Se usaron Sublime Text y TeXstudio como editores de texto, y MiKTeX como compilador de LaTeX, que son herramientas *open source* y por tanto gratuitas.

Costes materiales

Placas de circuito impreso

En el proceso de fabricación de las PCBs para el inversor, se realizaron dos pedidos separados con el proveedor JLCPCB.

Primer pedido

- Fecha de pedido: 2024-02-15
- Método de envío: FedEx International Packet
- Coste total: 156,26 € (PCBs de potencia) + 39,98 € (PCBs de control) + 19,24 € (envío)
- Cantidad: 10 unidades (5 unidades de la PCB de control y 5 unidades de la PCB de potencia)
- Coste final: **215,48 €**

Segundo pedido

- Fecha de pedido: 2024-03-28
- Método de envío: DHL Express Worldwide
- Coste total: 119,99 € (PCBs de potencia) + 39,98 € (PCBs de control) + 26,58 € (envío)
- Cantidad: 10 unidades (5 unidades de la PCB de control y 5 unidades de la PCB de potencia)
- Coste final: **186,55 €**

Total

- En total, se gastaron 215,48 + 186,55 = **402,03 €** en placas. Por suerte, se obtuvieron algunos descuentos por parte del fabricante que aliviaron la carga económica.

Componentes electrónicos

Para obtener el precio de los componentes electrónicos se extraen los BOMs de los proyectos generados en Altium Designer. El programa es capaz de consultar los distribuidores más conocidos y obtener automáticamente el mejor precio.

Placa de potencia

Componente	Cantidad para una PCB	Precio para dos PCBs (€)
0437001.WRA	1	2.8
10uF 0805	12	4.08
10uF 850V	10	Desconocido
150080GS75000	1	0.328
1779205141	1	7.4
2220Y1K00104KZT	6	61.56
4N35	2	0.924
613026243121	1	4.68
885012207098	15	1.98
885012207103	9	4.302
885012208058	1	0.33
CR0805-FX-1000ELF	3	0.023
CR0805-JW-102ELF	1	0.061
CR0805-JW-103ELF	12	0.79
CR1206AFX-6802EAS	6	0.12
CR1206-FX-2201ELF	2	0.37
CR1206-JW-363ELF	6	1.12
CR1206-JW-563ELF	6	1.12
CRCW120610K0FKEA	1	0.19
CRCW120630K0FKEA	1	0.06
CRG1206F100R	12	2.21
CRS1206-FX-4701ELF	2	1.14
CAB016M12FM3	3	778.68
ISO224	1	24.56
LEM CKSR 50-NP	3	94.8
LM311DR2G	1	0.82
MBR0530	1	0.8
MGJ6-series	6	65.04
R2M-2512FTK	2	2.99
RCV2512470KFKEG	24	32.69
SIHB150N60E-GE3	1	6.76
TPS72301	6	44.64
UCC21710	6	101.28
Total	175	**1248.64**

Cuadro 6.1: Lista de materiales de *Inverter_Power*, variante Wolfspeed.

Placa de control

Componente	Cantidad para una PCB	Precio para una PCB (€)
0437001.WRA	2	2.8
105310-1108	1	1.97
1053101112	2	3.48
10uF	6	2.04
150080GS75000	2	0.33
150080RS75000	1	0.11
150080YS75000	1	0.16
20MHz	1	0.32
24AA024H-I/SN	1	0.37
450405020524	1	1.01
47uH	2	1.44
61300611121	1	0.35
61302621121	2	2.52
629105136821	1	1.25
744764147	1	0.63
824501261	1	0.29
885012007052	2	0.096
885012107011	4	2.24
885012107014	2	0.50
885012207092	13	0.29
885012207098	17	1.12
885012207103	10	2.39
AD8479ARZ-RL	2	16.92
CPF0805B15RE	6	2.72
CPH3455-TL-H	3	1.38
CR0805-FX-1000ELF	9	0.03
CR0805-JW-102ELF	9	0.27
CR0805-JW-103ELF	19	0.63
CR0805-JW-472ELF	8	0.02
Diode 4D	1	0.48
DLW32SH510XF2	1	1.37
LM1117IMP-3.3/NOPB	1	0.32
LM339D	2	1.88
MCP2551-I/SN	1	1.24
RPMB5.0-3.0	1	8.1
STM32F777VIT6	1	19.27
USBLC6-2P6	1	0.48
Total	148	**80.82**

Cuadro 6.2: Lista de materiales de *Inverter_Control*.

Total

En total, el precio de los componentes es de 80,82 + 1248,64 = **1329,46 €**. Afortunadamente, la mayoría de componentes se obtuvieron como muestras, preguntando

a los fabricantes uno por uno, lo cual hizo viable el proyecto.

Herramientas

En este apartado se listan y se hace una valoración económica aproximada de todas las herramientas físicas utilizadas para el desarrollo del proyecto.

Herramienta	**Precio (€)**
Ordenador portátil	1200
Soladador Pinecil v2	25
Multímetro AN8009	25
Osciloscopio Yokogawa	Cesión temporal
Fuente de alimentación	Cesión temporal
Herramientas manuales	100
Total	1450

Cuadro 6.3: Lista de herramientas utilizadas en el proyecto y su coste.

El total de los costes asociados a las herramientas físicas asciende a **1450 €**.

Costes totales

A continuación se presenta un resumen de los costes totales del proyecto, incluyendo tanto los costes de ingeniería como los costes de fabricación.

Concepto	**Coste (€)**
Costes de ingeniería	
Desarrollo del control	4284,3
Desarrollo del *hardware*	11393,48
Desarrollo del *firmware*	2184,24
Documentación	1638,18
Total costes de ingeniería	**19500,2**
Costes de fabricación	
Placas de circuito impreso	402,03
Componentes electrónicos	1329,46
Herramientas	1450
Total costes de fabricación	**3181,49**
Coste total del proyecto	**22681,69**

Cuadro 6.4: Resumen de los costes totales del proyecto.

Conclusiones

En esta sección se rescatan los objetivos propuestos al inicio del trabajo y se comenta en qué medida se han logrado alcanzar. Además, se plantean los siguientes pasos en el desarrollo e implementación de este inversor para un vehículo de Formula Student. Estas partes concluyen el trabajo realizado.

Cumplimiento de los objetivos

Objetivo 1

> Adquirir **conocimiento** sobre control de motores eléctricos y diseño de convertidores de potencia.

En el desarrollo de este trabajo se ha abordado rigurosamente el modelo matemático del PMSM, describiendo analíticamente su comportamiento y diseñando un control analítico basado en estas ecuaciones. Se han tomado referencias de alta calidad para diseñar el control vectorial y se considera que el algoritmo implementado está a la orden del día. Cabe destacar que se han obviado algunos razonamientos matemáticos y formalismos para no sobrecomplicar el análisis, permitiendo una mejor comprensión de lo realmente útil para el control de máquinas eléctricas.

En cuanto al diseño de la electrónica de potencia, se ha profundizado considerablemente en los aspectos tanto de *hardware* como de la implementación del control en tiempo real. Se ha adquirido un conocimiento amplio y detallado sobre la selección y dimensionamiento de componentes electrónicos en convertidores de potencia, así como sobre las técnicas de diseño de PCBs y la integración de los diferentes módulos que conforman un accionamiento eléctrico. Sin embargo, no se han abordado aspectos importantes como los ensayos de doble pulso, la descomposición espectral de frecuencias o el análisis de impacto de EMIs debido a la extensión limitada del trabajo.

Se puede afirmar que este objetivo ha sido cumplido prácticamente en su totalidad.

Objetivo 2

> Definir unos **requisitos** para el *hardware* del inversor de tracción ideal para el equipo e-Tech Racing de la UPC-EEBE.

Atendiendo al apartado que define los requisitos, se consideran establecidos los requisitos del *hardware* del inversor de tracción ideal para e-Tech Racing. Se tomaron en cuenta todos los aspectos relevantes tanto por la normativa como internos del equipo, y se fusionaron en una lista de requisitos. Cumplir este objetivo no solo es

útil para dimensionar el convertidor de este trabajo, sino que también sirve como metodología de análisis para otros aspectos del tren de potencia del monoplaza.

Se considera más que cumplido este objetivo.

Objetivo 3

> **Diseñar** el *hardware* del inversor basado en esos requisitos.

El diseño del *hardware* del inversor está completado y se ha verificado en gran medida mediante simulaciones y pruebas de laboratorio. Se han considerado cada uno de los requisitos establecidos, y se ha procedido con la selección de componentes adecuados, la creación de esquemas eléctricos y el diseño de las PCBs. Se han realizado dos iteraciones de las placas para corregir los errores iniciales.

Este objetivo se ha alcanzado de manera satisfactoria.

Objetivo 4

> Evaluar y **validar** el *hardware* del inversor.

Se ha realizado una evaluación exhaustiva de cada parte del *hardware* del inversor para asegurar su funcionalidad y cumplimiento de los requisitos. Las pruebas realizadas han verificado que el diseño cumple con las especificaciones y que el *hardware* funciona correctamente en las condiciones esperadas. Sin embargo, debido a que no se ha dispuesto de una batería de 600 V, ni un motor adecuado, ni de una refrigeración líquida, el convertidor no se pudo probar en sus máximas especificaciones. Para algún parámetro crítico se han hecho predicciones matemáticas en base a algunos ensayos a menos potencia, pero esto no es suficiente para cerciorarse de que el convertidor realmente alcanza empíricamente el valor de potenia con el que se ha diseñado.

Este objetivo ha sido cumplido parcialmente, y sería necesaria una validación más extensa a potencias más elevadas.

Objetivo 5

> Implementar un **control vectorial** que permita el control independiente de **dos motores** con un solo microcontrolador.

Se ha procedido a la implementación del *firmware* que controlará el *hardware* del inversor, integrando gran parte del control diseñado previamente. Sin embargo, no se ha podido implementar la lectura del sensor de posición, motivo por el cual no

se ha podido verificar el lazo de corriente con un motor. Por esa misma razón, no se ha podido verificar el control dual, aunque en todo momento del desarrollo se han estado ejecutando ambos lazos de control simultáneamente, con lo que no debería ser un problema. Se han integrado las trayectorias que permiten un control de par de la misma manera que se ha hecho en las simulaciones. El lazo de control con debilitamiento de campo no se ha integrado, pero se considera sencillo implementarlo cogiendo como base las simulaciones realizadas.

Se ha cumplido en gran medida este objetivo, aunque queda lejos de ser una parte finalizada, ya que no se han validado algunos aspectos.

Futuras líneas de trabajo

Control

Como se ha mencionado anteriormente, se es consciente de que el algoritmo propuesto no es capaz de controlar la corriente en algunas situaciones como por ejemplo la regeneración en debilitamiento de campo. Se propone esperar una solución por parte de la comunidad científico-técnica, y hasta entonces, limitar la frenada regenerativa a la zona de baja BEMF. Otra estrategia sería explorar el control tabulado, con los puntos de operación pre-calculados.

Hardware

Líneas generales

- Diseñar e integrar una *coldplate*.
- Integrar el convertidor en una caja con conectores adecuados para el monoplaza.
- Evaluar la compatibilidad electromagnética con el resto de sistemas eléctricos del monoplaza.

Placa de potencia

- D1 tiene una especificación de corriente por debajo del valor medido. Debe ser reemplazado por una protección basada en PMOS o por otro *schottky* más adecuado.
- Rdis prácticamente no tiene efecto en comparación con R504...R510 y podría eliminarse.
- Implementar los condensadores de desacoplo de la conmutación en el *layout*.
- R301 puede tener un valor más bajo.
- La mayoría de los condensadores DNP son realmente necesarios.

- D501 falló una vez, aunque la causa sigue siendo desconocida.
- R501 debería tener un valor más grande.
- R511 y R512 deben tener una tolerancia pequeña, y esto debe especificarse en el esquemático.
- Los condensadores de 10 µF 50 V 0805 son caros y deberían reemplazarse por 10 µF 50 V 1206 o 10 µF 50 V 1210.
- Implementar la protección de desaturación aprovechando que el *gate driver* escogido está habilitado para ello.
- Tratar de fabricar las placas con estaño químico para asegurar una mejor conexión de los componentes *press-fit*.
- Usar un revestimiento conformal para cumplir con la normativa y garantizar el aislamiento.

Placa de control

- Revisar la lectura diferencial de la tensión.
- DCDC101 no se ha probado todavía.
- L101 tiene una especificación de corriente muy baja. Debería aguantar al menos 3 A.
- Q101 y el USB en general no se han probado aún.
- El punto del MCU en la serigrafía podría ser confuso. Verificar la orientación con el montaje, e indicar mejor en la serigrafía.
- Se debería añadir protección contra sobretensión de 3,3 V a todos los pines del MCU que no sean tolerantes a 5 V.
- La mayoría, si no todas, las resistencias de 15 Ω podrían ser de 0 Ω o incluso puentes de soldadura.
- Los condensadores de 10 µF 50 V 0805 son caros y deberían reemplazarse por 10 µF 50 V 1206 o 10 µF 50 V 1210.

Firmware

- Integrar la lectura del *encoder* incremental.
- Verificar el lazo de corriente.
- Optimizar el lazo de control para ocupar menos tiempo de ejecución.
- Verificar el control simultáneo de dos motores.

- Integrar el control con debilitamiento de campo (lazo de tensión) junto con una buena gestión de la frenada regenerativa en debilitamiento de campo.
- Verificar la comunicación CAN con el vehículo.
- Dejar preparada la posibilidad de tener dos inversores duales en el vehículo (para cuando se integren cuatro motores), prestando especial atención a los IDs de la comunicación CAN.
- Implementar un sistema operativo en tiempo real (RTOS) que gestione todas las tareas no síncronas con el control, es decir, comunicaciones, LEDs, y otros procesos poco prioritarios.
- Externalizar los parámetros constantes (los umbrales de error, los parámetros del motor, etc.) para evitar tener que cargar un binario nuevo cada vez que se modifican.
- Integrar los parámetros y otras configuraciones estáticas en una memoria EEPROM para tener que cargarlos solamente una vez.
- Crear una interfaz por USB para poder cargar parámetros y hacer ensayos externamente, registrando datos a una velocidad adecuada para su análisis.
- Crear un algoritmo que identifique los parámetros del motor automáticamente.
- Diseñar e implementar un modo de pruebas que permita encarar dos motores para ensayarlos de forma automática, uno actuando como motor y el otro como carga o generador.

Trabajos de Final de Estudios propuestos

Como manera de documentar detalladamente los avances en el diseño, se proponen tres trabajos a modo de continuación de este:

1. **Integración mecánica y gestión térmica de un inversor trifásico dual para tracción eléctrica**
2. **Optimización del diseño del *hardware* de un inversor trifásico dual para tracción eléctrica**
3. **Desarrollo del *firmware* de un inversor trifásico dual para tracción eléctrica**

7. Bibliografía

[1] Altium. Altium Designer Licensing Options. https://www.altium.com/altium-designer/licensing?TrackingId=ALCOM.ADPP.LP, 2024.

[2] D. Bienaimé, N. Devillers, M.C. Péra, F. Gustin, A. Berthon, and M.L. Grojo. Energetic macroscopic representation as an efficient tool for energy management in a hybrid electrical system embedded in a helicopter. In *2012 Electrical Systems for Aircraft, Railway and Ship Propulsion*, pages 1–6, 2012. doi: 10.1109/ESARS.2012.6387385.

[3] Campus Tirol Motorsport. AMK Motors. https://www.facebook.com/campustirolmotorsport/photos/a.328788424164236/439929373050140/?type=3, 2017.

[4] Cascadia Motion. CM200 Inverter. https://www.cascadiamotion.com/productlist/14-inverters/26-cm-inverters/49-cm200, 2024.

[5] Contribuyentes de Wikipedia. KISS Principle. https://en.wikipedia.org/wiki/KISS_principle, 2024.

[6] Bernat Costa. Disseny i validació d'una PCB tipus ECU per un cotxe de Formula Student. Bachelor's thesis, UPC-EEBE, 2024.

[7] David Costine. Power Electronic Circuits: Experiment 3 Introduction, Power Converter Layout, Loss Analysis and Design. https://web.eecs.utk.edu/~dcostine/ECE482/Spring2018/lectures/L5_out.pdf, 2018. ECE 482: Spring 2018.

[8] FSG. Formula Student Rules 2024. https://www.formulastudent.de/fileadmin/user_upload/all/2024/rules/FS-Rules_2024_v1.1.pdf, 2024.

[9] GoEngineer. Pricing Guide for SOLIDWORKS 3D CAD Software. https://www.goengineer.com/guide-to-buying-solidworks, 2024.

[10] Gabriel Gross. Cálculo de pérdidas en semiconductores. Documento interno, CITCEA-UPC, 2018.

[11] IPC. *Standard for Determining Current Carrying Capacity in Printed Board Design*. IPC – Association Connecting Electronics Industries, 2009. IPC-2152.

[12] IPC. *Generic Standard on Printed Board Design*. IPC – Association Connecting Electronics Industries, b edition, 2012. IPC-2221B.

[13] M. Jägger. Análisis del Efecto Causal en Sistemas Dinámicos. https://youtu.be/6peMik4tpLA?si=8CHV2GdOyZnhgKPx&t=12954, 2019.

[14] Knowles Capacitors. Hiteca™ Multilayer Ceramic Capacitors Datasheet. https://www.knowlescapacitors.com/getattachment/9e0201e7-ec9b-4362-8da3-5643cf1d64c8/Hiteca%E2%84%A2-Multilayer-Ceramic-Capacitors, 2024.

[15] Leapers Power. DFS05HF12EYR1 1200V/5.5mR Half Bridge SiC MOSFET Module Datasheet. https://www.leapers-power.com/action/download?file=/web/uploads/file/20230309/E66B0Y1w3x716fyKzgl84X9Kg38sCV40.pdf, 2023.

[16] LEM. CKSR_XX-NP Current Transducer Datasheet. https://www.lem.com/sites/default/files/products_datasheets/cksr_xx-np_v14.pdf, 2024.

[17] Lenze Bucher Hydraulics. MOBILE DCU. https://www.bucherdrives.com/64827/Products/MOBILE-DCU/index.aspx, 2024.

[18] Mathworks. MATLAB and Simulink Home Use. https://es.mathworks.com/store/link/products/home/ML, 2024.

[19] Mathworks. Permanent Magnet Synchronous Motor with Sinusoidal Flux Distribution. https://es.mathworks.com/help/sps/ref/pmsm.html, 2024.

[20] Mavilor. AC Servo Motors BL 110/140/190 Series. https://mavilor.es/wp-content/uploads/2018/06/bl100_series_sc.pdf, 2018.

[21] Carlos Miguel-Espinar. *Enhanced Flux-Weakening Control Algorithm for Permanent Magnet Synchronous Machines*. PhD thesis, CITCEA, UPC, 2023.

[22] Montesinos Miracle, Daniel. *Modelització i control d'accionaments elèctrics*. PhD thesis, UPC, Departament d'Enginyeria Elèctrica, Jun 2008.

[23] Murata. High Temperature Film Capacitor FH Series. https://www.murata.com/en-global/products/capacitor/htfc/overview/lineup/fh, 2023.

[24] Murata. KDC-MGJ2 DC-DC Converter Datasheet. https://www.murata.com/products/productdata/8807029997598/kdc-mgj2.pdf, 2024.

[25] P. D. Chandana Perera. *Sensorless Control of Permanent-Magnet Synchronous Motor Drives*. PhD thesis, Institut for Energiteknik, Aalborg Universitet, 2002.

[26] Plexim. Academic Price List. https://www.plexim.com/store/academic?field_product_category_tid_3=5&sort_by=title&sort_order=ASC, 2024.

[27] D. Redondo. Inverter Wiki. https://github.com/dweggg/Inverter/wiki, 2024.

[28] Seguridad Social. Cotización y Recaudación para Trabajadores. https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538, 2024.

[29] Grzegorz Sieklucki and Dawid Kara. Design and Modelling of Energy Conversion with the Two-Region Torque Control of a PMSM in an EV Powertrain. *Energies*, 15(13):4887, 2022. doi: 10.3390/en15134887. URL https://doi.org/10.3390/en15134887.

[30] Jason Sylvestre. Inverter DC Link Capacitor Selection. https://www.specterengineering.com/blog/2019/9/7/dc-link-capacitor-selection-for-your-inverter, Nov 2020.

[31] Texas Instruments. UCC21710-Q1 10-A Source/Sink Reinforced Isolated Single Channel Gate Driver for SiC/IGBT with Active Protection, Isolated Analog Sensing and High-CMTI Datasheet. https://www.ti.com/lit/ds/symlink/ucc21710-q1.pdf?ts=1705307191360&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fes-mx%252FUCC21710-Q1, 2023.

[32] UNITEK Industrie-Elektronik. Produkte. https://www.unitek-industrie-elektronik.de/produkte/, 2024.

[33] Vishay Sfernice. Power Dissipation in High Precision Vishay Sfernice Chip Resistors and Arrays. Technical Report 53048, Vishay Sfernice, October 2009.

[34] Wolfspeed. CAB016M12FM3 1200 V, 16 mR, Silicon Carbide, Half-Bridge Module Datasheet. https://assets.wolfspeed.com/uploads/2023/05/Wolfspeed_CAB016M12FM3_data_sheet.pdf, 2023.

[35] Wolfspeed. Wolfpack Mounting Instructions User Guide. https://assets.wolfspeed.com/uploads/2023/10/Wolfspeed_PRD-02302_Wolfpack_Mounting_Instructions_User_Guide.pdf, 2023.

[36] Würth Elektronik. REDCUBE PRESS-FIT Design Guide. https://www.we-online.com/components/media/o120138v410%20BCF_REDCUBE-PRESS-FIT-Design%20Guide_EN.pdf, 2023.

[37] Pau Arranz Pérez y Jordi Clos Garrido. Diseño e Implementación de un Inversor Trifásico para Controlar un Motor PMSM. Bachelor's thesis, UPC-EEBE, 2018.

[38] D. Redondo y N. Murguizur. Gestión de Proyectos. Documento interno Versión v0.2, e-Tech Racing, 2021.

[39] Simone Buso y Paolo Mattavelli. *Digital Control in Power Electronics*. Morgan & Claypool, 2006. doi: 10.2200/S00047ED1V01Y200609PEL002.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Trabajo de Final de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Volumen II: Apéndices

Diseño e implementación de un inversor trifásico dual para tracción eléctrica

Autor:
David Redondo Viñolo

Director:
Prof. Alfonso Conesa Roca

Convocatoria: Junio 2024

A. Documentación de control

A.1. *Script* para estimar los parámetros de un PMSM

```
% Clear workspace and command window
clc;
clear;

% Constants and motor specifications
P_max = 35e3;                        % Maximum Power (W)
Te_max = 26;                         % Maximum
   Electromagnetic Torque (Nm)
wm_max_rpm = 20e3;                   % Maximum Speed (RPM)
wm_max = wm_max_rpm * 2 * pi / 60;   % Maximum Speed (rad/s)

Vbat_min = 350;                      % Minimum Battery
   Voltage (V)
Vbat_max = 600;                      % Maximum Battery
   Voltage (V)
Vbat_nom = 540;                      % Nominal Battery
   Voltage (V)

Vs_min = Vbat_min / sqrt(3);         % Minimum motor Voltage
    (V)
Vs_max = Vbat_max / sqrt(3);         % Maximum motor Voltage
    (V)
Vs_nom = Vbat_nom / sqrt(3);         % Nominal motor Voltage
    (V)

fsw = 50e3;                          % Inverter maximum
   switching Frequency (Hz)

K_FW = 0.80;                         % Field Weakening
   Factor [0.7 .. 0.95]

% Pole number calculation
f_max = fsw / 40;
pp = floor(60 * f_max / wm_max_rpm);
n = 2 * pp;

% Constant power region
wm_nom = P_max / Te_max;
wm_nom_rpm = wm_nom * 60 / (2 * pi);
Te_wm_max = P_max / wm_max;
```

```
epsilon = 1.5; % Should be an output, but we lack one
   equation and we can take similar motors' saliency ratios

% Initial guesses for optimization
lambda_guess = 0.05;
Ld_guess = 0.0001;
Lq_guess = 0.0001;
Is_max_guess = 100;
gamma_MTPA_guess = pi/2;
gamma_FW_guess = pi;

%{
        Equations:
        eqn 1 --> Te_max = (3/2)*(n/2)*(lambda*Is_max*sin(
           gamma_MTPA)+(Ld-Lq)*Is_max^2*sin(gamma_MTPA)*cos
           (gamma_MTPA))
        We look for the MTPA working point, where Is_max at
            gamma_MTPA gives exactly Te_max

        eqn 2 --> 0 = lambda*Is_max*cos(gamma_MTPA)+(Ld-Lq)
           *Is_max^2*(cos(gamma_MTPA)^2-sin(gamma_MTPA)^2)
        In order to find a relationship between Is_max and
           gamma_MTPA we write the condition for MTPA: dTe/
           dgamma = 0

        eqn 3 --> Te_wm_max = (3/2)*(n/2)*(lambda*Is_max*
           sin(gamma_FW)+(Ld-Lq)*Is_max^2*sin(gamma_FW)*cos
           (gamma_FW))
        In order to aim for the constant power region, we
           find a working point in which Is_max at gamma_FW
            gives Te_wm_max

        eqn 4 --> (Vs_max/((n/2)*wm_max))^2 >= (lambda + Ld
           *Is_max*cos(gamma_FW))^2 + (Lq*Is_max*sin(
           gamma_FW))^2
        The FW working point (Is_max at gamma_FW) should be
            inside (>=) the voltage ellipse at wm_max

        eqn 5 --> (Vs_max/((n/2)*wm_nom))^2 >= (lambda + Ld
           *Is_max*cos(gamma_MTPA))^2 + (Lq*Is_max*sin(
           gamma_MTPA))^2
        The MTPA working point (Is_max at gamma_MTPA)
           should be inside (>=) the voltage ellipse at
           wm_nom

        %}

% Initial guess for optimization
x0 = [
lambda_guess;
```

```
Ld_guess;
Lq_guess;
Is_max_guess;
gamma_MTPA_guess;
gamma_FW_guess
];

% Define the equations
F = @(x) [
-Te_max + (3/2) * (n/2) * (x(1) * x(4) * sin(x(5)) + (x(2)
   - x(3)) * x(4)^2 * sin(x(5)) * cos(x(5)));
-0 + x(1) * x(4) * cos(x(5)) + (x(2) - x(3)) * x(4)^2 * (
   cos(x(5))^2 - sin(x(5))^2);
-Te_wm_max + (3/2) * (n/2) * (x(1) * x(4) * sin(x(6)) + (x
   (2) - x(3)) * x(4)^2 * sin(x(6)) * cos(x(6)));
-(Vs_nom * K_FW / ((n/2) * wm_max))^2 + (x(1) + x(2) * x(4)
    * cos(x(6)))^2 + (x(3) * x(4) * sin(x(6)))^2;
-(Vs_nom * K_FW / ((n/2) * wm_nom))^2 + (x(1) + x(2) * x(4)
    * cos(x(5)))^2 + (x(3) * x(4) * sin(x(5)))^2;
-epsilon + x(3) / x(2)
];

options = optimoptions('fsolve', 'MaxFunctionEvaluations',
   10000);

% Solve the equations
[x_sol, ~, exitflag] = fsolve(F, x0, options);

% Check if the optimization was successful
if exitflag <= 0
error('Optimization did not converge to a solution.');
end

% Extract optimized values
lambda_sol = x_sol(1);
Ld_sol = x_sol(2);
Lq_sol = x_sol(3);
Is_max_sol = x_sol(4);
gamma_MTPA_sol = x_sol(5);
gamma_FW_sol = x_sol(6);

% Create a symbolic variable for speed (wm)
syms wm

% Define the electromagnetic torque equation
Te_plot = piecewise(wm < wm_nom_rpm, Te_max, wm >
   wm_nom_rpm, P_max / (wm * 2 * pi / 60));

% Plot the electromagnetic torque
fplot(Te_plot, 'LineWidth', 2, 'Color','r');
```

```
        title('Electromagnetic␣Torque␣vs.␣Speed', 'FontSize', 14);
        xlabel('Speed␣(RPM)', 'FontSize', 12);
        ylabel('Electromagnetic␣Torque␣(Nm)', 'FontSize', 12);

        % Customize plot appearance
        xlim([0, wm_max_rpm]);
        ylim([0, Te_max * 1.5]);
        grid on;
        grid minor;

        % Display motor parameters
        fprintf('MOTOR␣PARAMETERS\n');
        fprintf('n␣=␣%.0f,␣pp␣=␣%.0f\n', [n, pp]);
        fprintf('lambda_sol␣=␣%.6f␣Wb\n', lambda_sol);
        fprintf('Ld_sol␣=␣%.4fe-3␣H\n', Ld_sol * 1e3);
        fprintf('Lq_sol␣=␣%.4fe-3␣H\n', Lq_sol * 1e3);
        fprintf('Is_max_sol␣=␣%.4f␣A\n', Is_max_sol);
        fprintf('gamma_MTPA_sol␣=␣%.4f␣rad\n', gamma_MTPA_sol);
        fprintf('gamma_FW_sol␣=␣%.4f␣rad\n', gamma_FW_sol);
```

A.2. *Script* para graficar las curvas de un PMSM

```
clc, clear, close all

% Permanent magnet synchronous machine constant parameters
motor = 'e-Tech␣2024';
FWF = 0.85;
switch motor
case 'e-Tech␣2024'
Ke = 0.13;                              % [V/(rad/s)] Mechanical
   speed constant
n = 6;                                  % [ad] Number of poles
pp = n/2;                               % [ad] Number of pole pairs
lambda_b = Ke / sqrt(3);                % [Wb] Base flux linkage
lambda_b = 0.052615;                    % [Wb] PM flux linkage
Ld =  0.1887e-3;                        % [H] d-axis inductance
Lq =  0.2831e-3;                        % [H] q-axis inductance
xi = Lq/Ld;                             % [ad] Saliency ratio
Rs = 0.0201;                            % [Ohm] Stator phase
   resistance (phase-to-phase/2)
P_max = 35e3;                           % [W] Maximum output power
SpeedMax = 20000;                       % [rpm] Motor maximum
   angular speed
Te_max = 26;                            % [Nm] Motor maximum
   angular torque
Vdc = 540;                              % [V] Battery DC voltage
V_base = FWF * Vdc / sqrt(3);           % [V] Maximum d-q voltage
```

```
I_max = 107;                            % [A] Maximum d-q current (
   sqrt(i_d^2+i_q^2))
I_base = lambda_b / Ld;                 % [A] Base current
T_base = (3/2)*pp*I_base*lambda_b;      % [Nm] Base torque
w_base = V_base/lambda_b;               % [rad/s] Base speed
Te_command = 26;
SpeedRef = 20000*2*pi/60;
case 'e-Tech␣2017'
Ke = 49.7e-3*60/(2*pi);                 % [Vrms,phph/(rad/s)] Speed
    constant, Vrms,phph/wm
n = 8;                                  % [ad] Number of poles
pp = n/2;                               % [ad] Number of pole pairs
lambda_b = Ke / (sqrt(3) * (n/2));      % [Wb] PM flux linkage,
   Vrms,phn/we
Ld =  0.520e-3;                         % [H] d-axis inductance
Lq =  1.265e-3;                         % [H] q-axis inductance
xi = Lq/Ld;                             % [ad] Saliency ratio
Rs = 0.104/2;                           % [Ohm] Stator phase
   resistance (phase-to-phase/2)
P_max = 60e3;                           % [W] Maximum output power
SpeedMax = 6000;                        % [rpm] Motor maximum
   angular speed
Te_max = 150;                           % [Nm] Motor maximum
   angular torque
Vdc = 580;                              % [V] Battery DC voltage
V_base = FWF * Vdc / sqrt(3);           % [V] Maximum d-q voltage (
   Maximum Torque per Voltage Flux-Weakening strategy with speed
   limiter for PMSM drives, 2020)
I_max = 200;                            % [A] Maximum d-q current (
   sqrt(i_d^2+i_q^2))
I_base = lambda_b / Ld;                 % [A] Base current
T_base = (3/2)*pp*I_base*lambda_b;      % [Nm] Base torque
w_base = V_base/lambda_b;               % [rad/s] Base speed
Te_command = 80;
SpeedRef = 7500*2*pi/60;
case 'Silence'
n = 40;                                 % [ad] Number of poles
pp = n/2;                               % [ad] Number of pole pairs
lambda_b = 0.02282824;                  % [Wb] Base flux linkage
Ld =  70e-6;                            % [H] d-axis inductance
Lq =  79e-6;                            % [H] q-axis inductance
xi = Lq/Ld;                             % [ad] Saliency ratio
Rs = 0.017;                             % [Ohm] Stator phase
   resistance (phase-to-phase/2)
SpeedMax = 1000;                        % [rpm] Motor maximum
   angular speed
Te_max = 80;                            % [Nm] Motor maximum
   angular torque
Vdc = 48;                               % [V] Battery DC voltage
```

```
V_base = FWF * Vdc / sqrt(3);           % [V] Maximum d-q voltage (
   Maximum Torque per Voltage Flux-Weakening strategy with speed
   limiter for PMSM drives, 2020)
I_max = 156;                            % [A] Maximum d-q current (
   sqrt(i_d^2+i_q^2))
I_base = lambda_b / Ld;                 % [A] Base current
T_base = (3/2)*pp*I_base*lambda_b;      % [Nm] Base torque
w_base = V_base/lambda_b;               % [rad/s] Base speed
Te_command = 66;
SpeedRef = 500*2*pi/60;
case 'Caruso␣2019'
n = 6;                                  % [ad] Number of poles
pp = n/2;                               % [ad] Number of pole pairs
lambda_b = 0.084;                       % [Wb] Base flux linkage
Ld =  9.77E-3;                          % [H] d-axis inductance
Lq =  14.94E-3;                         % [H] q-axis inductance
xi = Lq/Ld;                             % [ad] Saliency ratio
Rs = 2.21;                              % [Ohm] Stator phase
   resistance (phase-to-phase/2)
SpeedMax = 4300;                        % [rpm] Motor maximum
   angular speed
Te_max = 2;                             % [Nm] Motor maximum
   angular torque
Vdc = 310;                              % [V] Battery DC voltage
V_base = FWF * Vdc / sqrt(3);           % [V] Maximum d-q voltage (
   Maximum Torque per Voltage Flux-Weakening strategy with speed
   limiter for PMSM drives, 2020)
I_max = 8.5;                            % [A] Maximum d-q current (
   sqrt(i_d^2+i_q^2))
I_base = lambda_b / Ld;                 % [A] Base current
T_base = (3/2)*pp*I_base*lambda_b;      % [Nm] Base torque
w_base = V_base/lambda_b;               % [rad/s] Base speed
Te_command = 1.8;
SpeedRef = 380;

case 'AMK'
n = 10;                                 % [ad] Number of poles
pp = n/2;                               % [ad] Number of pole pairs
kE = 18.8;                              % [Vrmsphn/krpm(wm)] Speed
   constant
lambda_b = kE*(60/(2*pi))/(1000*(n/2)); % [Wb] Base flux linkage
Ld =  0.12e-3;                          % [H] d-axis inductance
Lq =  0.24e-3;                          % [H] q-axis inductance
xi = Lq/Ld;                             % [ad] Saliency ratio
Rs = 0.135;                             % [Ohm] Stator phase
   resistance (phase-to-phase/2)
SpeedMax = 20000;                       % [rpm] Motor maximum
   angular speed
Te_max = 30;                            % [Nm] Motor maximum
   angular torque
```

```
Vdc = 560;                                % [V] Battery DC voltage
V_base = FWF * Vdc / sqrt(3);             % [V] Maximum d-q voltage (
   Maximum Torque per Voltage Flux-Weakening strategy with speed
   limiter for PMSM drives, 2020)
I_max = 75;                               % [A] Maximum d-q current (
   sqrt(i_d^2+i_q^2))
I_base = lambda_b / Ld;                   % [A] Base current
T_base = (3/2)*pp*I_base*lambda_b;        % [Nm] Base torque
w_base = V_base/lambda_b;                 % [rad/s] Base speed
Te_command = 21;
SpeedRef = 25000*2*pi/60;
otherwise
end

%% dq plot
idiq = figure;

axis([[-I_max*1.2, I_max*1.2], [-I_max*1.2, I_max*1.2]])
               % [A] Current maximum values
xlabel('i_d␣[A],␣p.u.')
ylabel('i_q␣[A],␣p.u.')
grid on
ax = gca;
ax.DataAspectRatio = [1 1 1];
ax.GridLineStyle = '--';
ax.GridAlpha = 0.5;
ax.XAxisLocation="origin";
ax.YAxisLocation="origin";

%% Torque curves

hold on

tic
Te_vals = linspace(-Te_max, Te_max, 8); % [Nm] Torque values

syms id

for Te_val = 1:length(Te_vals)
if Te_vals(Te_val) > 0
tq_plot_pos = fplot(4*Te_vals(Te_val)/(3*n*(lambda_b+lambda_b/
   I_base*(1-xi)*id)), 'm', 'LineWidth', 2); % IPMSM torque
   equation, solved for iq
text(-0.5, 4*Te_vals(Te_val)/(3*n*(lambda_b+lambda_b/I_base*(1-xi)
   *-0.5)), sprintf('%.1f␣Nm',Te_vals(Te_val)),'Color','magenta','
   FontSize',12)
else
tq_plot_neg = fplot(4*Te_vals(Te_val)/(3*n*(lambda_b+lambda_b/
   I_base*(1-xi)*id)), 'r', 'LineWidth', 2); % IPMSM torque
   equation, solved for iq
```

```
text(-0.5, 4*Te_vals(Te_val)/(3*n*(lambda_b+lambda_b/I_base*(1-xi)
   *-0.5)), sprintf('%.1f␣Nm',Te_vals(Te_val)),'Color','red','
   FontSize',12)
end
end

clear id iq
toc
%% Current limit circle

alpha = linspace(0,2*pi);
i_lim_plot = plot(I_max*cos(alpha), I_max*sin(alpha), 'k', '
   LineWidth',2);

clear alpha


hold on
%% MTPA curve



if xi > 1
id_ref_MTPA = linspace(-I_max, 0, 1000);
iq_ref_MTPA_pos = sqrt((lambda_b*id_ref_MTPA+lambda_b/I_base*(1-xi)
   *id_ref_MTPA.^2)/(lambda_b/I_base*(1-xi)));
iq_ref_MTPA_pos(iq_ref_MTPA_pos > I_max | iq_ref_MTPA_pos < -I_max)
    = NaN;

iq_ref_MTPA_neg = -sqrt((lambda_b*id_ref_MTPA+lambda_b/I_base*(1-xi
   )*id_ref_MTPA.^2)/(lambda_b/I_base*(1-xi)));
iq_ref_MTPA_neg(iq_ref_MTPA_neg > I_max | iq_ref_MTPA_neg < -I_max)
    = NaN;

% Eliminar NaN de ambos vectores
valid_indices = ~isnan(iq_ref_MTPA_pos) & ~isnan(iq_ref_MTPA_neg);
id_ref_MTPA = id_ref_MTPA(valid_indices);
iq_ref_MTPA_pos = iq_ref_MTPA_pos(valid_indices);
iq_ref_MTPA_neg = iq_ref_MTPA_neg(valid_indices);
elseif xi < 1
id_ref_MTPA = linspace(I_max, 0, 1000);
iq_ref_MTPA_pos = sqrt((lambda_b*id_ref_MTPA-lambda_b/I_base*(1-xi)
   *id_ref_MTPA.^2)/(lambda_b/I_base*(1-xi)));
iq_ref_MTPA_pos(iq_ref_MTPA_pos > I_max | iq_ref_MTPA_pos < -I_max)
    = NaN;

iq_ref_MTPA_neg = -sqrt((lambda_b*id_ref_MTPA-lambda_b/I_base*(1-xi
   )*id_ref_MTPA.^2)/(lambda_b/I_base*(1-xi)));
iq_ref_MTPA_neg(iq_ref_MTPA_neg > I_max | iq_ref_MTPA_neg < -I_max)
    = NaN;
```

```
% Eliminar NaN de ambos vectores
valid_indices = ~isnan(iq_ref_MTPA_pos) & ~isnan(iq_ref_MTPA_neg);
id_ref_MTPA = id_ref_MTPA(valid_indices);
iq_ref_MTPA_pos = iq_ref_MTPA_pos(valid_indices);
iq_ref_MTPA_neg = iq_ref_MTPA_neg(valid_indices);
elseif xi == 1
id_ref_MTPA = [0, 0];
iq_ref_MTPA_pos = [0, I_max];
iq_ref_MTPA_neg = [-I_max, 0];

end

for i = 1:length(id_ref_MTPA)
if sqrt(id_ref_MTPA(i)^2 + iq_ref_MTPA_pos(i)^2) >= I_max
gamma_MTPA = abs(atan(iq_ref_MTPA_pos(i)/id_ref_MTPA(i))) + pi;
end
end

MTPA_plot_pos = plot(id_ref_MTPA, iq_ref_MTPA_pos, 'g', 'LineWidth'
   , 2);
MTPA_plot_neg = plot(id_ref_MTPA, iq_ref_MTPA_neg, 'c', 'LineWidth'
   , 2);

clear id_ref_MTPA iq_ref_MTPA_pos iq_ref_MTPA_neg valid_indices id
   iq
%% Voltage ellipses


syms w_MTPA

eqn_w_MTPA = (I_base + I_max*cos(gamma_MTPA))^2 / (V_base / (w_MTPA
    * lambda_b / I_base))^2 + (I_max*sin(gamma_MTPA))^2 / (V_base /
    (xi * w_MTPA * lambda_b / I_base))^2 - 1;

w_MTPA = double(solve(eqn_w_MTPA, w_MTPA));

speed_vals = [linspace(pp*SpeedMax*2*pi/60/5, pp*SpeedMax*2*pi/60,
   5), w_MTPA(w_MTPA>0), w_base]; % speed values

for i = 1:length(speed_vals)
h_ellipse = -I_base;
a_ellipse = V_base./(lambda_b/I_base*speed_vals(i));
b_ellipse = V_base./(xi*lambda_b/I_base*speed_vals(i));

t = linspace(0, 2*pi, 100);

id = a_ellipse * cos(t) + h_ellipse;
iq = b_ellipse * sin(t);
```

```
if i < length(speed_vals) - 1
voltageEllipse_plot = plot(id, iq, '--b','LineWidth',1);
text(h_ellipse, min(iq), sprintf('%.f␣rpm',speed_vals(i)*60/(2*pi)/
   pp),'Color','blue','FontSize',12);
elseif i == length(speed_vals) - 1
MTPAEllipse_plot = plot(id, iq, '--g','LineWidth',1);
text(h_ellipse, min(iq), sprintf('%.f␣rpm',speed_vals(i)*60/(2*pi)/
   pp),'Color','green','FontSize',12);
else
w0Ellipse_plot = plot(id, iq, '--c','LineWidth',1);
text(h_ellipse, min(iq), sprintf('%.f␣rpm',speed_vals(i)*60/(2*pi)/
   pp),'Color','cyan','FontSize',12);

end

end

clear h_ellipse a_ellipse b_ellipse id iq t
%% Plot legend

legend([i_lim_plot(1), tq_plot_pos(1), tq_plot_neg(1),
   MTPA_plot_pos(1), MTPA_plot_neg(1), voltageEllipse_plot(1),
   MTPAEllipse_plot(1), w0Ellipse_plot(1)], 'Current␣limit␣[A]', '
   Torque␣curves␣(traction)␣[Nm]', 'Torque␣curves␣(regen)␣[Nm]', '
   MTPA␣hyperbola␣(traction)', 'MTPA␣hyperbola␣(regen)', 'Voltage␣
   limit␣ellipses␣[rpm,␣mechanical]', 'MTPA␣speed␣point␣ellipse␣[
   rpm,␣mechanical]', 'MTPA␣speed␣limit␣ellipse␣[rpm,␣mechanical]')
```

B. Documentación del *hardware*

B.1. Generación de LUTs para NTCs

```
%% Temperature sensing LUT calculation for DFS05HF12EYR1
clc, clear
%% Initial variables
temperatures = 0:10:120; % Temperature array, 0.01degC resolution [
   degC]

%% NTC Parameters

% DFS05HF12EYR1 internal NTC
% Beta is a function of temperature

%beta_values = [3375, 3411, 3433]; % Beta values for different
   temperatures [K]
%beta_temps = [50, 80, 100]; % Temperatures for the different beta
   values [degC]

% CAB016M12FM3 internal NTC
% Beta is a function of temperature
beta_values = [3380, 3468, 3523]; % Beta values for different
   temperatures [K]
beta_temps = [50, 80, 100]; % Temperatures for the different beta
   values [degC]

% Both very similar, interchangable

beta_coeffs = polyfit(beta_temps, beta_values, 1); % Fit the beta
   deviation with linear regression

beta_temp = polyval(beta_coeffs,log(temperatures+1e-9)); % Beta is
   evaluated for all temperatures, avoiding 0degC [K]

T_0 = 25; % T at which NTC = R0 [degC]

R_0 = 5e3; % NTC resistance value at T_0 [R]

%% Calculations

% NTC resistance value for all temperatures, with varying beta
NTC = R_0*exp(-beta_temp.*(1./(273.15+25)-1./(273.15+temperatures))
   ); % NTC resistance with no errors [R]

% Reading using UCC21732 isolated analog reading. 200uA current
   source
```

```
R_filt = 10e3; % Filter resistance, in series with the NTC [R]

I_AIN = 200e-6; % Internal current source [A]
V_AIN = I_AIN * (R_filt + NTC); % Sensed voltage [V]

D = -20 * V_AIN + 100; % Duty cycle out [%]

VCC_GD = 5; % GD supply voltage [V]
V_read = VCC_GD * D/100; % Voltage read by ADC [V] (filtered with
   ideal RC)

VCC_ADC = 3.3; % MCU/ADC supply voltage [V]
bits = 12; % ADC bits [b]

bits_read = ceil(V_read * (2^bits) / VCC_ADC); % MCU/ADC read bits
   [b]
bits_read(bits_read>2^bits)=2^bits; % Saturation to 2^bits
bits_read(bits_read<0)=0; % Saturation to 0

% Create the plot
figure;
plot(temperatures, bits_read, 'LineWidth', 4);

% Add labels and title
xlabel('Temperature␣(degC)', 'FontSize', 12);
ylabel('Bits␣Read', 'FontSize', 12);
title('Bits␣Read␣vs.␣Temperature', 'FontSize', 14);

% Add grid and adjust limits
grid on;
xlim([min(temperatures)-5, max(temperatures)+5]);
ylim([min(bits_read)-50, max(bits_read)+50]);

% Customize the appearance
set(gca, 'FontSize', 10); % Adjust font size for axis labels
set(gca, 'LineWidth', 1.5); % Adjust axis line width

OUTPUT_LUT = [temperatures; bits_read];
```

B.2. Dimensionado de las resistencias de descarga

```
import itertools
import math

# Given data
target_resistance = 20000  # 18kR
tolerance = 1000  # +-1kR
power_rating = 20  # 20W
```

```
voltage = 600  # 600V
capacitance = 100e-6  # 100uF
Vf = 60  # 60V

# E-12 resistor values in R
e12_values = [1.0, 1.2, 1.5, 1.8, 2.2, 2.7, 3.3, 3.9, 4.7, 5.6,
   6.8, 8.2]
multipliers = [10 ** x for x in range(6)]

# SMD packages with their corresponding wattage
smd_packages = {
        "1206": 0.25,
        "1210": 0.5,
        "2010": 0.75,
        "2512": 1
}


# combination = [value, number of parallel resistors]

def calculate_resistance(combination):
return combination[0] / combination[1]


def calculate_power_dissipation(combination):
power_dissipation = voltage ** 2 / combination[0]
return power_dissipation


def calculate_discharge_time(combination):
resistance = calculate_resistance(combination)
discharge_time = resistance * capacitance * math.log(voltage / Vf)
return discharge_time


def find_parallel_combinations(target_resistance, tolerance,
   power_rating):
all_combinations = []
for r_value in e12_values:
for multiplier in multipliers:
base_resistance = r_value * multiplier
for num_parallel in range(1, 101):  # Assuming a maximum of 100
   parallel resistors
current_resistance = base_resistance / num_parallel
if target_resistance - tolerance <= current_resistance <=
   target_resistance + tolerance:
for package, wattage in smd_packages.items():
power_dissipation = calculate_power_dissipation([base_resistance,
   num_parallel])
if power_dissipation <= wattage:
```

```
discharge_time = calculate_discharge_time([base_resistance,
   num_parallel])
power_percentage = (power_dissipation / wattage) * 100
all_combinations.append((base_resistance, num_parallel, package,
   power_dissipation, discharge_time, power_percentage))
return all_combinations


if __name__ == "__main__":
combinations = find_parallel_combinations(target_resistance,
   tolerance, power_rating)
print("Possible␣combinations:")
for combo in combinations:
print(f"Resistance:␣{round(combo[0]/1000)}kR,␣n:␣{combo[1]},␣SMD␣
   Package:␣{combo[2]},␣"
f"P_diss:␣{round(combo[3],␣3)}␣W,␣t_dis:␣{round(combo[4],␣3)}␣s,␣"
f"Power:␣{round(combo[5],␣2)}%␣of␣package␣maximum")
```

B.3. Cálculo de divisores de tensión estandarizados

```
import itertools

# Define resistor series and their respective multipliers
resistor_series = {
        'E6': [1.0, 1.5, 2.2, 3.3, 4.7, 6.8],
        'E12': [1.0, 1.2, 1.5, 1.8, 2.2, 2.7, 3.3, 3.9, 4.7, 5.6,
           6.8, 8.2],
        'E24': [1.0, 1.1, 1.2, 1.3, 1.5, 1.6, 1.8, 2.0, 2.2, 2.4,
           2.7, 3.0, 3.3, 3.6, 3.9, 4.3, 4.7, 5.1, 5.6, 6.2, 6.8,
           7.5, 8.2, 9.1],
        'E48': [1.00, 1.05, 1.10, 1.15, 1.21, 1.27, 1.33, 1.40,
           1.47, 1.54, 1.62, 1.69, 1.78, 1.87, 1.96, 2.05, 2.15,
           2.26, 2.37, 2.49, 2.61, 2.74, 2.87, 3.01, 3.16, 3.32,
           3.48, 3.65, 3.83, 4.02, 4.22, 4.42, 4.64, 4.87, 5.11,
           5.36, 5.62, 5.90, 6.19, 6.49, 6.81, 7.15, 7.50, 7.87,
           8.25, 8.66, 9.09],
}

# Define tolerance values
sum_tolerance = 10000  # 10k ohms
# Target values
target_sum = 100000  # 100k ohms

Vout = 4
Vout_error = 0.15

# Function to generate resistor values for a given series
def generate_resistor_values(series):
```

```
return resistor_series[series]
# Function to calculate the closest combination
def find_closest_combination(Vout, Vout_error):
closest_sum = None
closest_combination = None
lowest_series_combination = None
comb_number = 0
for series, values in resistor_series.items():
for multiplier in values:
resistor_values = generate_resistor_values(series)
for combo in itertools.combinations(resistor_values, 2):
R1, R2 = combo
comb_number += 1

sum_resistors = (R1 + R2) * 10000  # in ohms
if Vout >  1.186 * 2:
actual_Vout = 1.186 * (1 + R2 / R1)
else:
actual_Vout = 1.186 * (1 + R1 / R2)
Vout_diff = Vout - actual_Vout

# Check if the sum is within +/- 10kR and the Vout error is within
   the specified tolerance
if ((target_sum - sum_tolerance) <= sum_resistors <= (target_sum +
   sum_tolerance)) and (Vout_diff <= Vout_error and Vout_diff > 0):
if closest_sum is None or abs(sum_resistors - target_sum) < abs(
   closest_sum - target_sum):
closest_sum = sum_resistors
if Vout >  1.186 * 2:
closest_combination = (series, R1, R2, actual_Vout)
else:
closest_combination = (series, R2, R1, actual_Vout)

if lowest_series_combination is None or series <
   lowest_series_combination[0]:
if Vout >  1.186 * 2:
lowest_series_combination = (series, R1, R2, actual_Vout)
else:
lowest_series_combination = (series, R2, R1, actual_Vout)

return closest_combination, lowest_series_combination, comb_number



# Find the closest combination
closest_combination, lowest_series_combination, comb_number =
   find_closest_combination(Vout, Vout_error)

# Print the result
if closest_combination:
```

```
series, R1, R2, actual_Vout = closest_combination
print("Closest combination found:")
print("Series:", series)
print("R1 =", R1*10, "kOhms")
print("R2 =", R2*10, "kOhms")
print("Vout =", actual_Vout, "V")
else:
print("No combination found within the specified Vout error.")

if lowest_series_combination:
series, R1, R2, actual_Vout = lowest_series_combination
print("Lowest series combination found:")
print("Series:", series)
print("R1 =", R1*10, "kOhms")
print("R2 =", R2*10, "kOhms")
print("Vout =", actual_Vout, "V")
else:
print("No combination found using the lowest series.")

print("A total of ", comb_number, " combinations were checked")
```

B.4. Documentación de *Inverter_Power* (Leapers)

ⓘ Dashed red lines indicate isolation barriers between TS and LV.

ⓘ Cyan areas indicate GLVS and cyan nets indicate signals from/to GLVS.

ⓘ Orange areas indicate TS and orange nets indicate high power TS.

Specifications:

f_sw = 40 kHz

V_in, max = 600 VDC
V_out, max = 245 V,RMS,ph-n (SVPWM)
I_out, max = 80 A,RMS
I_out, cont = 32 A,RMS

P_out, max = 53 kW
P_out, cont = 23.5 kW

Liquid cooled with water at 50ºC max

Changelog:

Version 1.0: (sent to production 15-02-2024)
- Base version

Version 1.1: (sent to production 28-03-2024)
- Added 5V supply protections
- Swapped pins 4 and 5 in gate drivers' LDOs
- Swapped MP+ and MP-, and their silkscreen
- Added testpoints for current sensors' reference
- Added testpoints for VDC_sns+ and VDC_sns-
- Renamed testpoints in [3]
- Added various silkscreen texts and indications
- Added layer physical logo

Changes in SCH but not in PCB:

19-04-2024: Added decoupling capacitors to semiconductor terminals (TS+ / TS-).

Known issues:

- D1 is not correctly rated. Replace with appropiate PMOS or beefier Schottky (beware the voltage drop)
- Rdis does practically nothing compared to R504 .. R510 and could be eliminated.
- R301 can be of lower value.
- Most DNP caps are actually needed.
- D501 failed once, but cause remains unknown.
- R501 should have a bigger value.
- R511 and R512 should have a small tolerance, and it should be specified in the schematic.
- 10uF 50V 0805 caps are expensive and should be replaced with 10uF 50V 1206 or 10uF 50V 1210.

LV Connector

OCP, OVP, reverse

Simple crowbar style protection. D2 will short when +5_pre > 5.1V, and it will cause the fuse to blow.

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power	Variant: Leapers		
Size: -	Page Contents: Inverter_Power.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet 1 of 5	
Checked by: _		Date: 29/05/2024	

Passive discharge

DC Bus capacitors, 100uF, Murata FHA85Y106KS

DC Link design considerations:

V_C > 1.1 · Vmax = 1.1 · 600 V = 660 V → 850 V

I_C,RMS ≈ 0.65 · I_phase,RMS = 0.65 · 80 A,RMS = 52 A,RMS → 10 x 5 A,RMS (ΔT = 10 ºC)
C > I_C,RMS / (V_ripple · f_sw) = 52 A,RMS / (15V · 40 kHz) ≈ 87 µF → 10 x 10 µF

Check:
https://www.specterengineering.com/blog/2019/9/7/dc-link-capacitor-selection-for-y

INPUTS/OUTPUTS

SiC Half-Bridges

Semiconductor details:

V_DSS(breakdown) = 1200 V // 1200 V
Ron = 5.5 .. 13 mΩ // 16.0 .. 28.8mΩ
Vf_D = 3.3 .. 4 V // 4.9 .. 5.5 V
T_rr = 41.5 .. 45 ns // 20.0 ns
Q_rr = 2.19 .. 3.94 µC // 1.30 µC
R_th_jc = 0.12 .. 0.15 K/W // 0.543 K/W
Q_G = 520 nC / 236 nC
C_in = 14.5 nF // 6.6 nF
R_G(int) = 1.9 Ω // 2.4 Ω
V_GS(th) = 2.8 .. 4.8 V // 1.8 .. 3.6 V

CA1, CA2, CB1, CB2, CC1, CC2

Current loop between top and bottom MOSFETs will cause excessive overshoot due to parasitic inductance and low parasitic capacitance. Increasing capacitance to hundreds of nF mitigates the effect. The capacitors essentially work as a switching decoupling. MLCCs with low DC bias or film, but MLCCs are way more compact. Place as close as possible to semiconductor terminals.

Overshoot analysis can be performed using a linear model proportional to DC bus voltage and output current. Easiest way to do it is using only one half bridge as a synchronous buck and varying input voltage with a fixed duty cycle and a R or RL load.

No C leads to 1430 Vp at max. specs.

C brings it down to 720 Vp at max. specs.

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Leapers	
Size: -	Page Contents: [1] Power_Stage.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 2 of 5
Checked by: _			Date: 29/05/2024

CSA , CSB , CSC

CKSR 50-NP/SP1 configured with Number of primary turns = 1 (R_phase-connector = 0.18 mΩ)

CSA , CSB , CSC

CKSR 50-NP/SP1 2.5V internal reference is used in order to have equal measuring range for positive and negative values. Voltage divider implemented just in case.

I_A, I_B, I_C

U_meas = (12.5mV/A · I_meas + Uref)

For ±150Apk:
V_meas_pk+ = 4.375V
V_meas_pk- = 0.625V

C201, C202, C203

The fluxgate oscillator draws current pulses of up to 30 mA at a rate of ca. 900 kHz. In the case of a power supply with high impedance, it is advised to provide local decoupling (100 nF or more, located close to the transducer).

CSA , CSB , CSC

AC insulation test RMS voltage, 50 Hz, 1 min:

Ud = 4.3 kV > 3*V_max = 1.8 kV

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power	Variant: Leapers		
Size: -	Page Contents: [2] Current_Sense.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet 3 of 5	
Checked by: _		Date: 29/05/2024	

INPUTS/OUTPUTS

T_A, T_B, T_C

Look-up table obtained with MATLAB script which can be found in the simulations folder.

For different temperatures:
V_meas(0ºC) = 0.246V
V_meas(25ºC) = 2V
V_meas(50ºC) = 2.578V
V_meas(90ºC) = 2.864V

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power	Variant: Leapers		
Size: -	Page Contents: [3] Gate_Drivers.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet 4 of 5	
Checked by: _		Date: 29/05/2024	

U_HS, U_LS

1. TRIP and OK signals are in open drain configuration, so they can be paralleled.
2. IN- is not used and tied to GND.
3. ENABLE to be given by MCU in active-high mode. When set to low for more than 1 µs, TRIP is reset.
4. Temperature sensing using low-side drivers. Ain outputs a current of 200 µA. PWM to analog using a RC filter, to be fed directly to MCU ADC. R405, R406 and C411 from SPICE simulation.
5. Miller clamp protection is used.
6. RGS_HS, RGS_LS: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
7. Overcurrent detection is not implemented.

LDO_HS, LDO_LS

An LDO is implemented to trim VEE_HS_A and VEE_LS_A during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

VEE = - 1.186 · (1 + R1/R2)
R1 + R2 ≈ 100 kΩ
Leapers → R1 = 36 kΩ, R2 = 56 kΩ
Wolfspeed → R1 = 68 kΩ, R2 = 30 kΩ

DCDC_HS, DCDC_LS
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM (t = 60 s (qualification test)): 8000 VPK

V_GS values:

The values can be modified by replacing DCDC_HS and DCDC_LS with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
I_GD(min) = f_sw · Q_G = 40 kHz · 520 nC = 20.8 mA
P_min = ΔV_GS · I_GD(min) = 20 V · 20.8 mA = 0.416 W → 2 W

RG_ON_HS, RG_OFF_HS, RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

CGS_HS, CGS_LS, CGD_HS, CGD_LS, Csn_HS, Csn_LS, Rsn_HS, Rsn_LS

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for DCDC_HS and DCDC_LS, as the gate charge would increase significantly. The maximum allowed capacitance would be:

CGS_max = 2 · P_DCDC / (ΔV_GS ^2 · f_sw)
= 2 · 2 W / ((20 V)^2) · 40 kHz) = 250 nF

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Leapers	
Size: -	Page Contents: [4] Half_Bridge.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 5 of 5
Checked by: _			Date: 29/05/2024

U_HS, U_LS

1. TRIP and OK signals are in open drain configuration, so they can be paralleled.
2. IN- is not used and tied to GND.
3. ENABLE to be given by MCU in active-high mode. When set to low for more than 1 µs, TRIP is reset.
4. Temperature sensing using low-side drivers. Ain outputs a current of 200 µA. PWM to analog using a RC filter, to be fed directly to MCU ADC. R405, R406 and C411 from SPICE simulation.
5. Miller clamp protection is used.
6. RGS_HS, RGS_LS: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
7. Overcurrent detection is not implemented.

LDO_HS, LDO_LS

An LDO is implemented to trim VEE_HS_A and VEE_LS_A during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

VEE = - 1.186 · (1 + R1/R2)
R1 + R2 ≈ 100 kΩ
Leapers → R1 = 36 kΩ, R2 = 56 kΩ
Wolfspeed → R1 = 68 kΩ, R2 = 30 kΩ

DCDC_HS, DCDC_LS
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM (t = 60 s (qualification test)): 8000 VPK

V_GS values:

The values can be modified by replacing DCDC_HS and DCDC_LS with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
I_GD(min) = f_sw · Q_G = 40 kHz · 520 nC = 20.8 mA
P_min = ΔV_GS · I_GD(min) = 20 V · 20.8 mA = 0.416 W → 2 W

RG_ON_HS, RG_OFF_HS, RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

CGS_HS, CGS_LS, CGD_HS, CGD_LS, Csn_HS, Csn_LS, Rsn_HS, Rsn_LS

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for DCDC_HS and DCDC_LS, as the gate charge would increase significantly. The maximum allowed capacitance would be:

CGS_max = 2 · P_DCDC / (ΔV_GS ^2 · f_sw)
= 2 · 2 W / ((20 V)^2) · 40 kHz) = 250 nF

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Leapers	
Size: -	Page Contents: [4] Half_Bridge.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 5 of 5
Checked by: _			Date: 29/05/2024

U_HS, U_LS

1. TRIP and OK signals are in open drain configuration, so they can be paralleled.
2. IN- is not used and tied to GND.
3. ENABLE to be given by MCU in active-high mode. When set to low for more than 1 µs, TRIP is reset.
4. Temperature sensing using low-side drivers. Ain outputs a current of 200 µA. PWM to analog using a RC filter, to be fed directly to MCU ADC. R405, R406 and C411 from SPICE simulation.
5. Miller clamp protection is used.
6. RGS_HS, RGS_LS: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
7. Overcurrent detection is not implemented.

LDO_HS, LDO_LS

An LDO is implemented to trim VEE_HS_A and VEE_LS_A during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

VEE = - 1.186 · (1 + R1/R2)
R1 + R2 ≈ 100 kΩ
Leapers → R1 = 36 kΩ, R2 = 56 kΩ
Wolfspeed → R1 = 68 kΩ, R2 = 30 kΩ

DCDC_HS, DCDC_LS
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM (t = 60 s (qualification test)): 8000 VPK

V_GS values:

The values can be modified by replacing DCDC_HS and DCDC_LS with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
I_GD(min) = f_sw · Q_G = 40 kHz · 520 nC = 20.8 mA
P_min = ΔV_GS · I_GD(min) = 20 V · 20.8 mA = 0.416 W → 2 W

RG_ON_HS, RG_OFF_HS, RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

CGS_HS, CGS_LS, CGD_HS, CGD_LS, Csn_HS, Csn_LS, Rsn_HS, Rsn_LS

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for DCDC_HS and DCDC_LS, as the gate charge would increase significantly. The maximum allowed capacitance would be:

CGS_max = 2 · P_DCDC / (ΔV_GS ^2 · f_sw)
= 2 · 2 W / ((20 V)^2) · 40 kHz) = 250 nF

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Leapers	
Size: -	Page Contents: [4] Half_Bridge.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 5 of 5
Checked by: _			Date: 29/05/2024

Discharge resistors:

t_dis = R_dis · C · ln(V_initial/V_final) = (470 kΩ / 24) · (100 µF) · ln(600 V / 60 V) = 4.509 s
P(R_dis, max) = V_max^2 / R_dis = 600 V^2 / 470 kΩ = 0.766 W < 1 W
ΔT ~ 110ºC/W · 0.766 W = 85 ºC
I_dis, max = 600 V / (470 kΩ / 24) = 30.64 mA

U503

Single supply configuration as per datasheet.

Maximum differential input voltage = 6.833 V - 677 mV = 6.156 V < 30 V

VDC_div = (TS+ - TS-) · 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ)

600 V · 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ) = 6.833 V
60 V · 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ) = 683 mV

P_R4 = I_R4^2 · R4 = ((600 V / (4.7 kΩ + 6 · 68 kΩ)) / 68 kΩ)^2 · 68 kΩ = 144 mW → 1206 package

V_TSAL = 5 V · 4.7 kΩ / (4.7 kΩ + 30 kΩ) = 677 mV ≡ 59.46 V in TS+ - TS-

R501

For U501 , the recommended diode forward current is 10 mA. SC_END can be between 20 V and 30 V.

I_min = 20 V / 2.2 kΩ= 9.1 mA
I_max = 30 V / 2.2 kΩ = 13.6 mA

U501

Maximum current across fototransistor

I_t(max) = 600 V / 2 MΩ = 0.3 mA < 150 mA

V_ce(max) = 10 V < 70 V

INPUTS/OUTPUTS

(TS+ - TS-) > 60V → TSAL_ON = 0V

(TS+ - TS-) < 60V → TSAL_ON = 5V

U504

(VDC_sns+ - VDC_sns-) = 1/3 · VDC_div = 1/3 · ((TS+ - TS-) 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ)) = 1/3 · 0.011388 · (TS+ - TS-)

(VDC_sns+ - VDC_sns-) = 1/3 · 0.011388 · 600 V = 2.278 V

U501, U502

Isolation Voltage: AC For 1 Minute, R.H. = 40 ~ 60% Viso = 5000 Vrms

U504

Maximum transient isolation voltage: VTEST = VIOTM, t = 60 s (qualification test) VIOTM = 7071 Vpk

DCDC501

Isolation voltage input to output, tested 100% for 60s(2)
VISO = 3000 V

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Leapers	
Size: -	Page Contents: [5]DC.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet * of *
Checked by: _			Date: 29/05/2024

A
B
C
ID:
Variant:
REV 1.1 - MARCH 2024
LV
TS
Warning
High voltage
Warning
High temperatures
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
GND
GND
+5V
+5V
TSAL
VDC+
SC
VDC-
AUX1
AUX2
PWM6
PWM5
PWM4
PWM3
PWM2
PWM1
I_C
TRIP
I_A
I_B
T_C
ENA
T_A
T_B
GND
GND
MPA
MPB
MPC
CSA
CSB
CSC
HB_A
HB_B
HB_C
David Redondo

A
B
C
ID:
Variant:
LV
TS
Warning
High voltage
Warning
High temperatures
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
David Redondo
GND
GND
+5V
+5V
TSAL
VDC+
SC
VDC-
AUX1
AUX2
PWM6
PWM5
PWM4
PWM3
PWM2
PWM1
I_C
TRIP
I_A
I_B
T_C
ENA
T_A
T_B
GND
GND
TRIP
ENA
OK
OK
MPA
MPB
MPC
CSA
CSB
CSC
HB_A
HB_B
HB_C
VDD
VEE1
VEE2
S1
S2
S3
S4
S5
S6
G1
G2
G3
G4
G5
G6
DC501

ID:
Variant:
A
B
C
LV
TS
GND GND
+5V +5V
TSAL VDC+
SC VDC-
AUX1 AUX2
PWM6 PWM5
PWM4 PWM3
PWM2 PWM1
I_C TRIP
I_A I_B
T_C ENA
T_A T_B
GND GND
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
Warning
High voltage
Warning
High temperatures
DC501
Layers

A
B
C
ID:
Variant:
LV
TS
GND GND
+5V +5V
TSAL VDC+
SC VDC-
AUX1 AUX2
PWM6 PWM5
PWM4 PWM3
PWM2 PWM1
I_C TRIP
I_A I_B
T_C ENA
T_A T_B
GND GND
TRIP
ENA
OK
MPA
MPB
MPC
CSA
CSB
CSC
HB_A
HB_B
HB_C
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
Warning
High voltage
Warning
High temperatures
DC501

Inverter Power

View from Top side (Scale 1:2)

View from Bottom side (Scale 1:2)

View from Back side (Scale 1:2)

Bill Of Materials

Designator	Name	Quantity
C405_A, C405_B, C405_C, C406_A, C406_B, C406_C, C408_A, C408_B, C408_C, C409_A, C409_B, C409_C	10uF	12
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10	10uF 850V	10
CA1, CA2, CB1, CB2, CC1, CC2	2220Y1K00104KZT	6
F1	0437001.WRA	1
DCDC501	1779205141	1
J1	613026243121	1
D2	BZG05C5V1-E3-TR	1
R401_A, R401_B, R401_C, R403_A, R403_B, R403_C	CR1206-JW-363ELF	6
R402_A, R402_B, R402_C, R404_A, R404_B, R404_C	CR1206-JW-563ELF	6
R503	CRCW120610K0FKEA	1
R511	CRCW120630K0FKEA	1
HB_A, HB_B, HB_C	DFS05HF12EYR1	3
HW1	LOGO CAPAS (4)	1
MP-, MP+, MPA, MPB, MPC	M4	5
D1	MBR0530	1
MH1, MH2	Mounting_Hole_M4	2
RD1, RD2, RD3, RD4, RD5, RD6, RD7, RD8, RD9, RD10, RD11, RD12, RD13, RD14, RD15, RD16, RD17, RD18, RD19, RD20, RD21, RD22, RD23, RD24	RCV2512470KFKEG	24
R406_A, R406_B, R406_C	CR0805-FX-1000ELF	3
R301	CR0805-JW-102ELF	1
R302, R305, R306, R405_A, R405_B, R405_C, RGS_HS_A, RGS_HS_B, RGS_HS_C, RGS_LS_A, RGS_LS_B, RGS_LS_C	CR0805-JW-103ELF	12
R504, R505, R506, R507, R508, R509	CR1206AFX-6802EAS	6
R501, R513	CR1206-FX-2201ELF	2
R510, R512	CRS1206-FX-4701ELF	2
CSA, CSB, CSC	LEM CKSR 50-NP	3
DCDC_HS_A, DCDC_HS_B, DCDC_HS_C, DCDC_LS_A, DCDC_LS_B, DCDC_LS_C	MGJ6-series	6
U503	LM311DR2G	1
C501	885012208058	1
RG_OFF_HS_A, RG_OFF_HS_B, RG_OFF_HS_C, RG_OFF_LS_A, RG_OFF_LS_B, RG_OFF_LS_C, RG_ON_HS_A, RG_ON_HS_B, RG_ON_HS_C, RG_ON_LS_A, RG_ON_LS_B, RG_ON_LS_C	CRG1206F100R	12
U504	ISO224	1
LDO_HS_A, LDO_HS_B, LDO_HS_C, LDO_LS_A, LDO_LS_B, LDO_LS_C	TPS72301	6
U_HS_A, U_HS_B, U_HS_C, U_LS_A, U_LS_B, U_LS_C	UCC21710	6
Q501	SIHB150N60E-GE3	1
R502, Rdis	R2M-2512FTK	2
U501, U502	4N35	2
D501	BZG05C6V2-E3-TR	1
D301	150080GS75000	1
C201, C202, C203, C402_A, C402_B, C402_C, C404_A, C404_B, C404_C, C411_A, C411_B, C411_C, C502, C504, C506	885012207098	15
C401_A, C401_B, C401_C, C403_A, C403_B, C403_C, C503, C505, C507	885012207103	9

Copper thickness in hole 25-50um, watch out for 1.10mm holes
Chemical tin 1-15um

Layer Stack Legend

Material	Layer	Thickness	Dielectric Material	Type	Gerber
	Top Overlay			Legend	GTO
Surface Material	Top Solder	0.01mm	Solder Resist	Solder Mask	GTS
CF-004	**TOP**	**0.07mm**		**Signal**	**GTL**
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
Copper	**GND**	**0.07mm**		**Signal**	**G1**
		0.90mm	*FR-4*	*Dielectric*	
Copper	**PWR**	**0.07mm**		**Signal**	**G2**
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
CF-004	**BOT**	**0.07mm**		**Signal**	**GBL**
Surface Material	Bottom Solder	0.01mm	Solder Resist	Solder Mask	GBS
	Bottom Overlay			Legend	GBO

Total thickness: 1.60mm

B.5. Documentación de *Inverter_Power* (Wolfspeed)

ⓘ Dashed red lines indicate isolation barriers between TS and LV.

ⓘ Cyan areas indicate GLVS and cyan nets indicate signals from/to GLVS.

ⓘ Orange areas indicate TS and orange nets indicate high power TS.

Specifications:

f_sw = 40 kHz

V_in, max = 600 VDC
V_out, max = 245 V,RMS,ph-n (SVPWM)
I_out, max = 80 A,RMS
I_out, cont = 32 A,RMS

P_out, max = 53 kW
P_out, cont = 23.5 kW

Liquid cooled with water at 50ºC max

Changelog:

Version 1.0: (sent to production 15-02-2024)
- Base version

Version 1.1: (sent to production 28-03-2024)
- Added 5V supply protections
- Swapped pins 4 and 5 in gate drivers' LDOs
- Swapped MP+ and MP-, and their silkscreen
- Added testpoints for current sensors' reference
- Added testpoints for VDC_sns+ and VDC_sns-
- Renamed testpoints in [3]
- Added various silkscreen texts and indications
- Added layer physical logo

Changes in SCH but not in PCB:

19-04-2024: Added decoupling capacitors to semiconductor terminals (TS+ / TS-).

Known issues:

- D1 is not correctly rated. Replace with appropiate PMOS or beefier Schottky (beware the voltage drop)
- Rdis does practically nothing compared to R504 .. R510 and could be eliminated.
- R301 can be of lower value.
- Most DNP caps are actually needed.
- D501 failed once, but cause remains unknown.
- R501 should have a bigger value.
- R511 and R512 should have a small tolerance, and it should be specified in the schematic.
- 10uF 50V 0805 caps are expensive and should be replaced with 10uF 50V 1206 or 10uF 50V 1210.

LV Connector

OCP, OVP, reverse

Simple crowbar style protection. D2 will short when +5_pre > 5.1V, and it will cause the fuse to blow.

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power	Variant: Wolfspeed		
Size: -	Page Contents: Inverter_Power.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet 1 of 5	
Checked by: _		Date: 29/05/2024	

Passive discharge

DC Bus capacitors, 100uF, Murata FHA85Y106KS

DC Link design considerations:

V_C > 1.1 · Vmax = 1.1 · 600 V = 660 V → 850 V

I_C,RMS ≈ 0.65 · I_phase,RMS = 0.65 · 80 A,RMS = 52 A,RMS → 10 x 5 A,RMS (ΔT = 10 ºC)
C > I_C,RMS / (V_ripple · f_sw) = 52 A,RMS / (15V · 40 kHz) ≈ 87 µF → 10 x 10 µF

Check:
https://www.specterengineering.com/blog/2019/9/7/dc-link-capacitor-selection-for-y

INPUTS/OUTPUTS

SiC Half-Bridges

Semiconductor details:

V_DSS(breakdown) = 1200 V // 1200 V
Ron = 5.5 .. 13 mΩ // 16.0 .. 28.8mΩ
Vf_D = 3.3 .. 4 V // 4.9 .. 5.5 V
T_rr = 41.5 .. 45 ns // 20.0 ns
Q_rr = 2.19 .. 3.94 µC // 1.30 µC
R_th_jc = 0.12 .. 0.15 K/W // 0.543 K/W
Q_G = 520 nC / 236 nC
C_in = 14.5 nF // 6.6 nF
R_G(int) = 1.9 Ω // 2.4 Ω
V_GS(th) = 2.8 .. 4.8 V // 1.8 .. 3.6 V

CA1, CA2, CB1, CB2, CC1, CC2

Current loop between top and bottom MOSFETs will cause excessive overshoot due to parasitic inductance and low parasitic capacitance. Increasing capacitance to hundreds of nF mitigates the effect. The capacitors essentially work as a switching decoupling. MLCCs with low DC bias or film, but MLCCs are way more compact. Place as close as possible to semiconductor terminals.

Overshoot analysis can be performed using a linear model proportional to DC bus voltage and output current. Easiest way to do it is using only one half bridge as a synchronous buck and varying input voltage with a fixed duty cycle and a R or RL load.

No C leads to 1430 Vp at max. specs.

C brings it down to 720 Vp at max. specs.

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Wolfspeed	
Size: -	Page Contents: [1] Power_Stage.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 2 of 5
Checked by: _			Date: 29/05/2024

CSA , CSB , CSC

CKSR 50-NP/SP1 configured with Number of primary turns = 1 (R_phase-connector = 0.18 mΩ)

CSA , CSB , CSC

AC insulation test RMS voltage, 50 Hz, 1 min:

Ud = 4.3 kV > 3*V_max = 1.8 kV

CSA , CSB , CSC

CKSR 50-NP/SP1 2.5V internal reference is used in order to have equal measuring range for positive and negative values. Voltage divider implemented just in case.

I_A, I_B, I_C

U_meas = (12.5mV/A · I_meas + Uref)

For ±150Apk:
V_meas_pk+ = 4.375V
V_meas_pk- = 0.625V

C201, C202, C203

The fluxgate oscillator draws current pulses of up to 30 mA at a rate of ca. 900 kHz. In the case of a power supply with high impedance, it is advised to provide local decoupling (100 nF or more, located close to the transducer).

Company: e-Tech Racing	e-techracing.es	
Project: Inverter Power	Variant: Wolfspeed	
Size: -	Page Contents: [2] Current_Sense.SchDoc	Version: 1.1
		Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com	Sheet 3 of 5
Checked by: _		Date: 29/05/2024

INPUTS/OUTPUTS

T_A, T_B, T_C

Look-up table obtained with MATLAB script which can be found in the simulations folder.

For different temperatures:
V_meas(0ºC) = 0.246V
V_meas(25ºC) = 2V
V_meas(50ºC) = 2.578V
V_meas(90ºC) = 2.864V

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power	Variant: Wolfspeed		
Size: -	Page Contents: [3] Gate_Drivers.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet 4 of 5	
Checked by: _		Date: 29/05/2024	

U_HS, U_LS

1. TRIP and OK signals are in open drain configuration, so they can be paralleled.
2. IN- is not used and tied to GND.
3. ENABLE to be given by MCU in active-high mode. When set to low for more than 1 µs, TRIP is reset.
4. Temperature sensing using low-side drivers. Ain outputs a current of 200 µA. PWM to analog using a RC filter, to be fed directly to MCU ADC. R405, R406 and C411 from SPICE simulation.
5. Miller clamp protection is used.
6. RGS_HS, RGS_LS: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
7. Overcurrent detection is not implemented.

LDO_HS, LDO_LS

An LDO is implemented to trim VEE_HS_A and VEE_LS_A during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

VEE = - 1.186 · (1 + R1/R2)
R1 + R2 ≈ 100 kΩ
Leapers → R1 = 36 kΩ, R2 = 56 kΩ
Wolfspeed → R1 = 68 kΩ, R2 = 30 kΩ

DCDC_HS, DCDC_LS
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM (t = 60 s (qualification test)): 8000 VPK

V_GS values:

The values can be modified by replacing DCDC_HS and DCDC_LS with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
I_GD(min) = f_sw · Q_G = 40 kHz · 520 nC = 20.8 mA
P_min = ΔV_GS · I_GD(min) = 20 V · 20.8 mA = 0.416 W → 2 W

RG_ON_HS, RG_OFF_HS, RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

CGS_HS, CGS_LS, CGD_HS, CGD_LS, Csn_HS, Csn_LS, Rsn_HS, Rsn_LS

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for DCDC_HS and DCDC_LS, as the gate charge would increase significantly. The maximum allowed capacitance would be:

CGS_max = 2 · P_DCDC / (ΔV_GS ^2 · f_sw)
= 2 · 2 W / ((20 V)^2) · 40 kHz) = 250 nF

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Wolfspeed	
Size: -	Page Contents: [4] Half_Bridge.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 5 of 5
Checked by: _			Date: 29/05/2024

U_HS, U_LS

1. TRIP and OK signals are in open drain configuration, so they can be paralleled.
2. IN- is not used and tied to GND.
3. ENABLE to be given by MCU in active-high mode. When set to low for more than 1 µs, TRIP is reset.
4. Temperature sensing using low-side drivers. Ain outputs a current of 200 µA. PWM to analog using a RC filter, to be fed directly to MCU ADC. R405, R406 and C411 from SPICE simulation.
5. Miller clamp protection is used.
6. RGS_HS, RGS_LS: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
7. Overcurrent detection is not implemented.

LDO_HS, LDO_LS

An LDO is implemented to trim VEE_HS_A and VEE_LS_A during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

VEE = - 1.186 · (1 + R1/R2)
R1 + R2 ≈ 100 kΩ
Leapers → R1 = 36 kΩ, R2 = 56 kΩ
Wolfspeed → R1 = 68 kΩ, R2 = 30 kΩ

DCDC_HS, DCDC_LS
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM (t = 60 s (qualification test)): 8000 VPK

V_GS values:

The values can be modified by replacing DCDC_HS and DCDC_LS with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
I_GD(min) = f_sw · Q_G = 40 kHz · 520 nC = 20.8 mA
P_min = ΔV_GS · I_GD(min) = 20 V · 20.8 mA = 0.416 W → 2 W

RG_ON_HS, RG_OFF_HS, RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

CGS_HS, CGS_LS, CGD_HS, CGD_LS, Csn_HS, Csn_LS, Rsn_HS, Rsn_LS

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for DCDC_HS and DCDC_LS, as the gate charge would increase significantly. The maximum allowed capacitance would be:

CGS_max = 2 · P_DCDC / (ΔV_GS ^2 · f_sw)
= 2 · 2 W / ((20 V)^2) · 40 kHz) = 250 nF

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Wolfspeed	
Size: -	Page Contents: [4] Half_Bridge.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 5 of 5
Checked by: _			Date: 29/05/2024

U_HS, U_LS

1. TRIP and OK signals are in open drain configuration, so they can be paralleled.
2. IN- is not used and tied to GND.
3. ENABLE to be given by MCU in active-high mode. When set to low for more than 1 µs, TRIP is reset.
4. Temperature sensing using low-side drivers. Ain outputs a current of 200 µA. PWM to analog using a RC filter, to be fed directly to MCU ADC. R405, R406 and C411 from SPICE simulation.
5. Miller clamp protection is used.
6. RGS_HS, RGS_LS: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
7. Overcurrent detection is not implemented.

LDO_HS, LDO_LS

An LDO is implemented to trim VEE_HS_A and VEE_LS_A during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

VEE = - 1.186 · (1 + R1/R2)
R1 + R2 ≈ 100 kΩ
Leapers → R1 = 36 kΩ, R2 = 56 kΩ
Wolfspeed → R1 = 68 kΩ, R2 = 30 kΩ

DCDC_HS, DCDC_LS
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM (t = 60 s (qualification test)): 8000 VPK

V_GS values:

The values can be modified by replacing DCDC_HS and DCDC_LS with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
I_GD(min) = f_sw · Q_G = 40 kHz · 520 nC = 20.8 mA
P_min = ΔV_GS · I_GD(min) = 20 V · 20.8 mA = 0.416 W → 2 W

RG_ON_HS, RG_OFF_HS, RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

CGS_HS, CGS_LS, CGD_HS, CGD_LS, Csn_HS, Csn_LS, Rsn_HS, Rsn_LS

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for DCDC_HS and DCDC_LS, as the gate charge would increase significantly. The maximum allowed capacitance would be:

CGS_max = 2 · P_DCDC / (ΔV_GS ^2 · f_sw)
= 2 · 2 W / ((20 V)^2) · 40 kHz) = 250 nF

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Wolfspeed	
Size: -	Page Contents: [4] Half_Bridge.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet 5 of 5
Checked by: _			Date: 29/05/2024

Discharge resistors:

t_dis = R_dis · C · ln(V_initial/V_final) = (470 kΩ / 24) · (100 µF) · ln(600 V / 60 V) = 4.509 s
P(R_dis, max) = V_max^2 / R_dis = 600 V^2 / 470 kΩ = 0.766 W < 1 W
ΔT ~ 110ºC/W · 0.766 W = 85 ºC
I_dis, max = 600 V / (470 kΩ / 24) = 30.64 mA

U503

Single supply configuration as per datasheet.

Maximum differential input voltage = 6.833 V - 677 mV = 6.156 V < 30 V

VDC_div = (TS+ - TS-) · 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ)

600 V · 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ) = 6.833 V
60 V · 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ) = 683 mV

P_R4 = I_R4^2 · R4 = ((600 V / (4.7 kΩ + 6 · 68 kΩ)) / 68 kΩ)^2 · 68 kΩ = 144 mW → 1206 package

V_TSAL = 5 V · 4.7 kΩ / (4.7 kΩ + 30 kΩ) = 677 mV ≡ 59.46 V in TS+ - TS-

R501

For U501 , the recommended diode forward current is 10 mA. SC_END can be between 20 V and 30 V.

I_min = 20 V / 2.2 kΩ= 9.1 mA
I_max = 30 V / 2.2 kΩ = 13.6 mA

U501

Maximum current across fototransistor

I_t(max) = 600 V / 2 MΩ = 0.3 mA < 150 mA

V_ce(max) = 10 V < 70 V

INPUTS/OUTPUTS

(TS+ - TS-) > 60V → TSAL_ON = 0V

(TS+ - TS-) < 60V → TSAL_ON = 5V

U504

(VDC_sns+ - VDC_sns-) = 1/3 · VDC_div = 1/3 · ((TS+ - TS-) 4.7 kΩ / (4.7 kΩ + 6 · 68 kΩ)) = 1/3 · 0.011388 · (TS+ - TS-)

(VDC_sns+ - VDC_sns-) = 1/3 · 0.011388 · 600 V = 2.278 V

U501, U502

Isolation Voltage: AC For 1 Minute, R.H. = 40 ~ 60% Viso = 5000 Vrms

U504

Maximum transient isolation voltage: VTEST = VIOTM, t = 60 s (qualification test) VIOTM = 7071 Vpk

DCDC501

Isolation voltage input to output, tested 100% for 60s(2)
VISO = 3000 V

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Power		Variant: Wolfspeed	
Size: -	Page Contents: [5]DC.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet * of *
Checked by: _			Date: 29/05/2024

A
B
C
ID:
Variant:
REV 1.1 - MARCH 2024
LV
TS
Warning
High voltage
Warning
High temperatures
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
GND
GND
+5V
+5V
TSAL
VDC+
SC
VDC-
AUX1
AUX2
PWM6
PWM5
PWM4
PWM3
PWM2
PWM1
I_C
TRIP
I_A
I_B
T_C
ENA
T_A
T_B
GND
GND
MPA
MPB
MPC
CSA
CSB
CSC
HB_A
HB_B
HB_C
VDD
VEE1
VEE2
RG_OFF_HS_A
RG_ON_HS_A
RG_ON_LS_A
RG_OFF_LS_A
RGS_LS_A
RG_OFF_HS_B
RG_ON_HS_B
RG_ON_LS_B
RG_OFF_LS_B
RGS_LS_B
RG_OFF_HS_C
RG_ON_HS_C
RG_ON_LS_C
RG_OFF_LS_C
RGS_LS_C
RGS
G1
G2
G3
G4
G5
G6
S1
S2
S3
S4
S5
S6
Q501
D501
TP501
TP502
TP503
TP504
TP505
TP104
TP105
R501
R502
R503
R504
R505
R506
R507
R508
R509
R510
R511
R512
R513
C501
C502
C503
C504
C505
C506
C507
C508
C509
C510
C511
C512
C513
C514
C515
C516
C517
C518
C519
C520
C521
C522
C523
C524
C525
C526
C527
C528
C529
C530
C531
C532
C533
C534
C535
C536
C537
C538
C539
C540

A
B
C
ID:
Variant:
LV
TS
Warning
High voltage
Warning
High temperatures
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
David Redondo
GND
GND
+5V
+5V
TSAL
VDC+
SC
VDC-
AUX1
AUX2
PWM6
PWM5
PWM4
PWM3
PWM2
PWM1
I_C
TRIP
I_A
I_B
T_C
ENA
T_A
T_B
GND
GND
TRIP
ENA
OK
MPA
MPB
MPC
CSA
CSB
CSC
HB_A
HB_B
HB_C
VDD
VEE1
VEE2
S1
S2
S3
S4
S5
S6
G1
G2
G3
G4
G5
G6
DC501

ID:
Variant:
A
B
C
LV
TS
GND GND
+5V +5V
TSAL VDC+
SC VDC-
AUX1 AUX2
PWM6 PWM5
PWM4 PWM3
PWM2 PWM1
I_C TRIP
I_A I_B
T_C ENA
T_A T_B
GND GND
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
Warning
High voltage
Warning
High temperatures
DC501
Layers

A
B
C
ID:
Variant:
LV
TS
GND GND
+5V +5V
TSAL VDC+
SC VDC-
AUX1 AUX2
PWM6 PWM5
PWM4 PWM3
PWM2 PWM1
I_C TRIP
I_A I_B
T_C ENA
T_A T_B
GND GND
TRIP
ENA
OK
MPA
MPB
MPC
CSA
CSB
CSC
HB_A
HB_B
HB_C
VDD VEE1
VEE2
RG_OFF_HS_A
RG_ON_HS_A
RG_ON_LS_A
RG_OFF_LS_A
RGS_LS_A
RGS_HS_A
RG_OFF_HS_B
RG_ON_HS_B
RG_ON_LS_B
RG_OFF_LS_B
RGS_LS_B
RGS_HS_B
RG_OFF_HS_C
RG_ON_HS_C
RG_ON_LS_C
RG_OFF_LS_C
RGS_LS_C
RGS_HS_C
S1 S2 S3 S4 S5 S6
G1 G2 G3 G4 G5 G6
R405_A
R405_B
R405_C
TP101
TP102
TP103
DIN912
M2.5x08
0.4 N·m
DIN912
M4x06
2 N·m
DIN912 M4x16
DIN980 M4
DIN125 M4
Warning
High voltage
Warning
High temperatures
DC501
David Redondo

Inverter Power

View from Top side (Scale 1:2)

View from Bottom side (Scale 1:2)

View from Back side (Scale 1:2)

Bill Of Materials

Designator	Name	Quantity
C405_A, C405_B, C405_C, C406_A, C406_B, C406_C, C408_A, C408_B, C408_C, C409_A, C409_B, C409_C	10uF	12
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10	10uF 850V	10
CA1, CA2, CB1, CB2, CC1, CC2	2220Y1K00104KZT	6
F1	0437001.WRA	1
DCDC501	1779205141	1
J1	613026243121	1
D2	BZG05C5V1-E3-TR	1
HB_A, HB_B, HB_C	CAB016M12FM3	3
R402_A, R402_B, R402_C, R404_A, R404_B, R404_C	CR1206-JW-303ELF	6
R401_A, R401_B, R401_C, R403_A, R403_B, R403_C	CR1206-JW-683ELF	6
R503	CRCW120610K0FKEA	1
R511	CRCW120630K0FKEA	1
HW1	LOGO CAPAS (4)	1
MP-, MP+, MPA, MPB, MPC	M4	5
D1	MBR0530	1
DCDC_HS_A, DCDC_HS_B, DCDC_HS_C, DCDC_LS_A, DCDC_LS_B, DCDC_LS_C	MGJ6-series	6
MH1, MH2	Mounting_Hole_M4	2
RD1, RD2, RD3, RD4, RD5, RD6, RD7, RD8, RD9, RD10, RD11, RD12, RD13, RD14, RD15, RD16, RD17, RD18, RD19, RD20, RD21, RD22, RD23, RD24	RCV2512470KFKEG	24
R406_A, R406_B, R406_C	CR0805-FX-1000ELF	3
R301	CR0805-JW-102ELF	1
R302, R305, R306, R405_A, R405_B, R405_C, RGS_HS_A, RGS_HS_B, RGS_HS_C, RGS_LS_A, RGS_LS_B, RGS_LS_C	CR0805-JW-103ELF	12
R504, R505, R506, R507, R508, R509	CR1206AFX-6802EAS	6
R501, R513	CR1206-FX-2201ELF	2
R510, R512	CRS1206-FX-4701ELF	2
CSA, CSB, CSC	LEM CKSR 50-NP	3
U503	LM311DR2G	1
C501	885012208058	1
RG_OFF_HS_A, RG_OFF_HS_B, RG_OFF_HS_C, RG_OFF_LS_A, RG_OFF_LS_B, RG_OFF_LS_C, RG_ON_HS_A, RG_ON_HS_B, RG_ON_HS_C, RG_ON_LS_A, RG_ON_LS_B, RG_ON_LS_C	CRG1206F100R	12
U504	ISO224	1
LDO_HS_A, LDO_HS_B, LDO_HS_C, LDO_LS_A, LDO_LS_B, LDO_LS_C	TPS72301	6
U_HS_A, U_HS_B, U_HS_C, U_LS_A, U_LS_B, U_LS_C	UCC21710	6
Q501	SIHB150N60E-GE3	1
R502, Rdis	R2M-2512FTK	2
U501, U502	4N35	2
D501	BZG05C6V2-E3-TR	1
D301	150080GS75000	1
C201, C202, C203, C402_A, C402_B, C402_C, C404_A, C404_B, C404_C, C411_A, C411_B, C411_C, C502, C504, C506	885012207098	15
C401_A, C401_B, C401_C, C403_A, C403_B, C403_C, C503, C505, C507	885012207103	9

Copper thickness in hole 25-50um, watch out for 1.10mm holes
Chemical tin 1-15um

Layer Stack Legend

Material	Layer	Thickness	Dielectric Material	Type	Gerber
	Top Overlay			Legend	GTO
Surface Material	Top Solder	0.01mm	Solder Resist	Solder Mask	GTS
CF-004	**TOP**	**0.07mm**		**Signal**	**GTL**
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
Copper	**GND**	**0.07mm**		**Signal**	**G1**
		0.90mm	*FR-4*	*Dielectric*	
Copper	**PWR**	**0.07mm**		**Signal**	**G2**
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
Prepreg		*0.10mm*	*PP-006*	*Dielectric*	
CF-004	**BOT**	**0.07mm**		**Signal**	**GBL**
Surface Material	Bottom Solder	0.01mm	Solder Resist	Solder Mask	GBS
	Bottom Overlay			Legend	GBO

Total thickness: 1.60mm

B.6. Documentación de *Inverter_Control*

ⓘ Cyan nets indicate external signals.

ⓘ Purple nets indicate internal 3.3V signals.

ⓘ Blue nets indicate analog signals read by the ADC.

ⓘ Red nets indicate 20-30V.

ⓘ Pink nets indicate treated supply.

ⓘ Light green nets indicate CAN.

ⓘ Yellow nets indicate serial communication.

ⓘ Dark green nets indicate input capture.

Known issues:

- DCDC101 not tested yet.
- L101 has a very low current rating. Choose new part with a rating of at least 3A.
- Q101 and USB in general not tested yet.
- U501 not working as expected.
- U201 dot in silkscreen might be misleading. Check the orientation very carefully.
- 3.3V OVP should be added to all non-5V-tolerant MCU pins.
- Most if not all 15R resistors could very well be 0R or solder bridges.
- 10uF 50V 0805 caps are expensive and should be replaced with 10uF 50V 1206 or 10uF 50V 1210.

Changelog:

Version 1.0:
- Base version, sent to production 21-02-2024

Version 1.1:
- Pull-up and pull-down added for BOOT0 control
- Added I2C pull-ups
- LV+_sns deleted, VBAT connected to 3.3V
- Extras LED color and names changed
- Added layers physical logo
- Moved supply filter to +5V
- Some silkscreen

Company: e-Tech Racing	e-techracing.es	
Project: Inverter Control	Variant: [No Variations]	
Size: -	Page Contents: Inverter_Control.SchDoc	Version: 1.1
		Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com	Sheet 1 of 1
Checked by: _		Date: 20/05/2024

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Control	Variant: [No Variations]		
Size: -	Page Contents: [1] Supply.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet * of *	
Checked by: _		Date: 20/05/2024	

INPUTS/OUTPUTS
USB_VBUS
PWM
PWM1_L
PWM2_L
PWM3_L
PWM4_L
PWM5_L
PWM6_L
PWM_L
ia_L
ib_L
ic_L
VDC_L
Tinv_L
TRIP_L
ENABLE_L
Tmot_L
WRN_L
Z_L
B_L
A_L
ia_R
ib_R
ic_R
VDC_R
Tinv_R
TRIP_R
ENABLE_R
Tmot_R
WRN_R
Z_R
B_R
A_R
CAN1_RX
CAN1_TX
SC_det
DIR
PWM1_R
PWM2_R
PWM3_R
PWM4_R
PWM5_R
PWM6_R
PWM_R
ERR
LED_R
LED_L
Timers
PWM L → TIM1
PWM R → TIM8
- PWM1 → CH1N
- PWM2 → CH1
- PWM3 → CH2N
- PWM4 → CH2
- PWM5 → CH3N
- PWM6 → CH3
Enc. L → TIM4
- A → CH1
- B → CH3
- Z → CH4
Enc. R → TIM2
- A → CH1
- B → CH3
- Z → CH4
ADC
ADC L → ADC2:
- iaMeas_L (IN6)
- ibMeas_L (IN7)
- icMeas_L (IN8)
- VDC_L (IN9)
ADC R → ADC1:
- iaMeas_R (IN0)
- ibMeas_R (IN1)
- icMeas_R (IN2)
- VDC_R (IN3)
ADC3:
- Tinv_L (IN10)
- Tinv_R (IN11)
- Tmot_L (IN12)
- Tmot_R (IN13)
ST-link
+3.3
SYS_JTMS-SWCLK
SYS_JTMS-SWDIO
NRST
SYS_JTMS-SWO
GND
ST-LINK
Serial Wire Debug
STM32F777VIT
RST protection
NRST
R203
10kR
NRST_MCU
C221
1uF
VDDA
L201
47uH
C206
C207
10nF
1uF
C208
C209
C210
100nF
22uF
BOOT select
R204
R205
BOOT0
U201
STM32F777VIT6
C213
C212
C211
C216
C215
C214
C220
2.2uF
C203
C204
C205
C217
C218
C219
EXTI9
EXTI8
EEPROM (I2C1)
U202
WP
SCL
SDA
VSS
VCC
A0
A1
A2
24AA024H-I/SN
I2C_SDA
I2C_SCL
I2C pull-ups
R206
R207
10kR
Oscillator
C201
C202
15pF
OSC_IN
OSC_OUT
XTAL201
Connect to cap before GND
USB connector
USB_VBUS
R201
R202
USB_OTG_ID
USB_OTG_VBUS
USB_OTG_H
USB_OTG_L
TVS1
USBLC6-2P6
VBUS
D-
D+
ID
GND
Shell
USB
Configured as USB OTG / Dual Role Device
Company: e-Tech Racing
e-techracing.es
Project: Inverter Control
Variant: [No Variations]
Size: -
Page Contents: [2]MCU.SchDoc
Version: 1.1
Department: Powertrain
Author: David Redondo
dredondovinolo@gmail.com
Sheet * of *
Checked by: _
Date: 20/05/2024

Company: e-Tech Racing	e-techracing.es	
Project: Inverter Control	Variant: [No Variations]	
Size: -	Page Contents: [3] CAN.SchDoc	Version: 1.1
		Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com	Sheet * of *
Checked by: _		Date: 20/05/2024

Feedback connector

Auxiliary warning (WRN)

Incremental encoder

As the motors' temperature sensors are not specified, the user may modify the resistor combination to find a suitable input for the ADC, then load a custom lookup table to have an appropiate reading.

Example

The WRN circuit can be used so that the MCU can detect a specified alarm. A resistive sensor can be used to detect any physical signal, such as overtemperature in any component (e.g. water outlet, gearbox, ...), underpressure of the cooling system, etc. Similar to the motor temperature sensors, the user may modify the resistor combination to have a suitable reading and adjust the voltage divider in order to set the treshold. Other types of sensors can be used, given a previous study and correct implementation.

INPUTS/OUTPUTS

Comparator supply

Motor thermistor

Only compatible with resistive temperature sensors.

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Control		Variant: [No Variations]	
Size: -	Page Contents: [4] Feedback.SchDoc		Version: 1.1
			Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com		Sheet * of *
Checked by: _			Date: 20/05/2024

Power PCB Connector

VDC sense AFE

Difference amplifier

LT SPICE simulation for reference.

VDC = Vref + (VDC_sns+ - VDC_sns-) = Vref + 1/3 · 0.011388 · (TS+ - TS-)

Maximum error due to using a voltage divider as a voltage reference is 1.79mV in ADC, which translates to 0.47V in (TS+ - TS-). Lowering the resistance values (while keeping proportionality) will reduce the error, but more current will be drawn.

INPUTS/OUTPUTS

Current sense

Resistor combination can be adjusted for increased measuring range at the cost of lower resolution.

Temperature selection

ENABLE is output directly from the MCU, it has been checked that UC21732 is able to detect it at 3.3V. Similarly, TRIP comes at 5V, and uses a 5V tolerant GPIO in the MCU.

VDC_offset = Vref · 2^12 bits / (3.3V) = 0.02083 · 2^12 bits / (3.3V) = 129 bits
VDC_gain = 1/((1/3 · 0.011388 V/V) · (2^12 bits / 3.3 V)) = 0.212240269 V / bit
VDC_max = 0.212240269 V / bit · 2^12 bits = 869.34V

ix_offset = (10k/(3.3k+10k)) · 2.5V · 2^12 bits / (3.3V) = 2333 bits
ix_gain = (10k/(3.3k+10k)) / (12.5 mV/A · (2^12 bits / 3.3 V))) = 0.0484609962 A / bit
ix_max(+/-) = +/- 0.0484609962 A / bit · 2^12 bits / 2 = +/- 99 A

Inverters temperature should be calculated with a lookup table according to this graph. The lookup table and graph is generated with a MATLAB script which can be found in the simulations folder.

Tinv is a pulsated signal that can read directly as a PWM input or be passed through an RC filter (in Inverter_Power) to convert it into an analog signal. This board intends to read it with the ADC. The reading itself is in the TS part of the power board and connected to the AIN pin of UCC21732.

Based on the sensed voltage, the duty cycle (D) of the UCC21732 isolated output signal is calculated using the following relationship: D = -20 · V_AIN + 100

If filtered, the voltage at Tinv is calculated as: V_Tinv = VCC_GD · D/100 = 5V · (-20 · V_AIN + 100)/100

Company: e-Tech Racing	e-techracing.es	
Project: Inverter Control	Variant: [No Variations]	
Size: -	Page Contents: [5] Power_AFE.SchDoc	Version: 1.1
		Department: Powertrain
Author: David Redondo	dredondovinolo@gmail.com	Sheet * of *
Checked by: _		Date: 20/05/2024

Reverse direction

SC detection

Status LEDs

INPUTS/OUTPUTS

Company: e-Tech Racing	e-techracing.es		
Project: Inverter Control	Variant: [No Variations]		
Size: -	Page Contents: [6] Extras.SchDoc	Version: 1.1	
		Department: Powertrain	
Author: David Redondo	dredondovinolo@gmail.com	Sheet * of *	
Checked by: _		Date: 20/05/2024	

ID:
REV 1.1 - MARCH 202
Left
Right
STM32F7/VI
DCDC101
R507_L
R508_L
C503_L
R506_L
C502_L
C403_L
R403_L
R504_L
R505_L
R504_R
R505_R
RR403_R
3V3
CLK
GND
DIO
RST
SWO
Layers
1

ID:
Left
Right
STM32F777VI
3V3
CLK
GND
DIO
RST
SWO
XT
RX
PWR
L
R
FLT
Layers
2

ID:
Left
Right
STM32F777VI
Feedback_L
Feedback_R
PWR_L
PWR_R
ST-LINK
3V3
CLK
GND
DIO
RST
SWO
USB
DIR
FLT
XT
XR
Layers
3

ID:
Left
Right
STM32F777VI
XT
RX
3V3
CLK
GND
DIO
RST
SWD
USB
PWR
L
R
FLT
Layers
4

Inverter Control

View from Top side (Scale 1:1)

View from Bottom side (Scale 1:1)

View from Front side (Scale 1:1)

Bill Of Materials

Designator	Name	Quantity
C101, C105, C204, C207, C209, C212, C215, C218, C221, C307	885012207103	10
C102, C103, C106, C107, C409_L, C409_R	10uF	6
C104, C203, C208, C211, C214, C217, C306, C407_L, C407_R, C410_L, C410_R, C501_L, C501_R, C502_L, C502_R, C503_L, C503_R	885012207098	17
C108, C109	885012107014	2
C201, C202	885012007052	2
C205, C210, C213, C216	885012107011	4
C206, C401_L, C401_R, C402_L, C402_R, C403_L, C403_R, C404_L, C404_R, C405_L, C405_R, C406_L, C406_R	885012207092	13
C219, C220	2.2uF	2
D101	824501261	1
D102	MBR0530	1
D104	150080YS75000	1
D601	150080RS75000	1
D602, D603	150080GS75000	2
DCDC101	RPMB5.0-3.0	1
DIR	450405020524	1
F101, F102	0437001.WRA	2
Feedback_L, Feedback_R	1053101112	2
HW1	LOGO CAPAS (4)	1
J1	105310-1108	1
L101, L102	47uH	2
L201	744764147	1
L301	DLW32SH510XF2	1
MP_L, MP_R	M3	2
PWR_L, PWR_R	61302621121	2
Q601, Q602, Q603	CPH3455-TL-H	3
R101, R401_L, R401_R, R403_L, R403_R, R405_L, R405_R, R502_L, R502_R	CR0805-FX-1000ELF	9
R201, R202, R203, R205, R206, R207, R402_L, R402_R, R404_L, R404_R, R406_L, R406_R, R504_L, R504_R, R506_L, R506_R, R508_L, R508_R, R610	CR0805-JW-103ELF	19
R407_L, R407_R, R601, R602, R603, R604, R605, R606, R611	CR0805-JW-102ELF	9
R408_L, R408_R, R409_L, R409_R, R509_L, R509_R	CPF0805B15RE	6
R501_L, R501_R, R503_L, R503_R, R505_L, R505_R, R507_L, R507_R	CR0805-JW-472ELF	8
R607, R608, R609	47kR	3
ST-LINK	61300611121	1
TVS1	USBLC6-2P6	1
TVS301	Diode 4D	1
U101	LM1117IMP-3.3/NOPB	1
U201	STM32F777VIT6	1
U202	24AA024H-I/SN	1
U301	MCP2551-I/SN	1
U401_L, U401_R	LM339D	2
U501_L, U501_R	AD8479ARZ-RL	2
USB	629105136821	1
XTAL201	20MHz	1

Layer Stack Legend

Material	Layer	Thickness	Dielectric Material	Type	Gerber
	Top Overlay			Legend	GTO
Surface Material	Top Solder	0.010mm	Solder Resist	Solder Mask	GTS
CF-004	**TOP**	**0.035mm**		**Signal**	**GTL**
Prepreg		*0.100mm*	*PP-006*	*Dielectric*	
Prepreg		*0.100mm*	*PP-006*	*Dielectric*	
Copper	**GND**	**0.035mm**		**Signal**	**G1**
		1.040mm	*FR-4*	*Dielectric*	
Copper	**PWR**	**0.035mm**		**Signal**	**G2**
Prepreg		*0.100mm*	*PP-006*	*Dielectric*	
Prepreg		*0.100mm*	*PP-006*	*Dielectric*	
CF-004	**BOT**	**0.035mm**		**Signal**	**GBL**
Surface Material	Bottom Solder	0.010mm	Solder Resist	Solder Mask	GBS
	Bottom Overlay			Legend	GBO

Total thickness: 1.600mm

C. Documentación del *firmware*

e-Tech Racing's Inverter Firmware

v0

Generated by Doxygen 1.10.0

1 Data Structure Index 1
1.1 Data Structures . . . 1

2 File Index 3
2.1 File List . . . 3

3 Data Structure Documentation 5
3.1 Analog Struct Reference . . . 5
3.1.1 Detailed Description . . . 5
3.1.2 Field Documentation . . . 5
3.1.2.1 currentOffsets . . . 5
3.1.2.2 ia . . . 5
3.1.2.3 ib . . . 6
3.1.2.4 ic . . . 6
3.1.2.5 vDC . . . 6
3.2 CANMessageInfo Struct Reference . . . 6
3.2.1 Field Documentation . . . 6
3.2.1.1 DLC . . . 6
3.2.1.2 getSig . . . 6
3.2.1.3 ID . . . 7
3.2.1.4 IDE . . . 7
3.3 Duties Struct Reference . . . 7
3.3.1 Detailed Description . . . 7
3.3.2 Field Documentation . . . 7
3.3.2.1 Da . . . 7
3.3.2.2 Db . . . 7
3.3.2.3 Dc . . . 8
3.4 Encoder Struct Reference . . . 8
3.4.1 Detailed Description . . . 8
3.4.2 Field Documentation . . . 8
3.4.2.1 A . . . 8
3.4.2.2 B . . . 8
3.4.2.3 cosTheta_e . . . 9
3.4.2.4 directionMeas . . . 9
3.4.2.5 sinTheta_e . . . 9
3.4.2.6 theta_e . . . 9
3.4.2.7 we . . . 9
3.4.2.8 Z . . . 9
3.5 Feedback Struct Reference . . . 9
3.5.1 Detailed Description . . . 10
3.5.2 Field Documentation . . . 10
3.5.2.1 idMeas . . . 10
3.5.2.2 iqMeas . . . 10

3.5.2.3 speedMeas . . . 10
3.5.2.4 torqueCalc . . . 10
3.6 InverterStruct Struct Reference . . . 11
3.6.1 Detailed Description . . . 11
3.6.2 Field Documentation . . . 12
3.6.2.1 analog . . . 12
3.6.2.2 direction . . . 12
3.6.2.3 duties . . . 12
3.6.2.4 enable . . . 12
3.6.2.5 enable_pin . . . 12
3.6.2.6 enable_port . . . 12
3.6.2.7 enableSW . . . 12
3.6.2.8 encoder . . . 13
3.6.2.9 errors . . . 13
3.6.2.10 feedback . . . 13
3.6.2.11 hadc . . . 13
3.6.2.12 htim . . . 13
3.6.2.13 idLoop . . . 13
3.6.2.14 iqLoop . . . 13
3.6.2.15 led . . . 13
3.6.2.16 motor . . . 14
3.6.2.17 reference . . . 14
3.6.2.18 speedLoop . . . 14
3.6.2.19 state . . . 14
3.6.2.20 tempInverter . . . 14
3.6.2.21 tempMotor . . . 14
3.6.2.22 vd . . . 14
3.6.2.23 vq . . . 14
3.6.2.24 vsMax . . . 15
3.7 LED Struct Reference . . . 15
3.7.1 Detailed Description . . . 15
3.7.2 Field Documentation . . . 15
3.7.2.1 mode . . . 15
3.7.2.2 pin . . . 15
3.7.2.3 port . . . 15
3.8 MotorConstants Struct Reference . . . 16
3.8.1 Detailed Description . . . 16
3.8.2 Field Documentation . . . 16
3.8.2.1 betaMinusIsc . . . 16
3.8.2.2 eightTimesOneMinusXiSquared . . . 16
3.8.2.3 fourTimesOneMinusXi . . . 17
3.8.2.4 fourTimesOneMinusXiOnePlusXiSquared . . . 17

3.8.2.5 fourTimesOneMinusXiXiSquared . . . 17
3.8.2.6 invThreePpLambda . . . 17
3.8.2.7 invTorqueBase . . . 17
3.8.2.8 isc . . . 17
3.8.2.9 lambdaDivLqMinusLd . . . 17
3.8.2.10 oneMinusXi . . . 17
3.8.2.11 threePpLambda . . . 18
3.8.2.12 threePpLdMinusLq . . . 18
3.8.2.13 torqueBase . . . 18
3.8.2.14 twoMinusXi . . . 18
3.8.2.15 twoMinusXiSquared . . . 18
3.8.2.16 twoTimesOneMinusXiOnePlusXiSquared . . . 18
3.8.2.17 twoTimesOneMinusXiXiSquared . . . 18
3.8.2.18 xi . . . 18
3.8.2.19 xiSquared . . . 19
3.9 MotorParameters Struct Reference . . . 19
3.9.1 Detailed Description . . . 20
3.9.2 Field Documentation . . . 20
3.9.2.1 b . . . 20
3.9.2.2 constants . . . 20
3.9.2.3 dTorqueMax . . . 20
3.9.2.4 iMax . . . 20
3.9.2.5 J . . . 20
3.9.2.6 lambda . . . 20
3.9.2.7 Ld . . . 21
3.9.2.8 Lq . . . 21
3.9.2.9 pp . . . 21
3.9.2.10 Rs . . . 21
3.9.2.11 speedMax_RPM . . . 21
3.9.2.12 torqueMax . . . 21
3.9.2.13 vDCMax . . . 21
3.10 Reference Struct Reference . . . 22
3.10.1 Detailed Description . . . 22
3.10.2 Field Documentation . . . 22
3.10.2.1 idRef . . . 22
3.10.2.2 iqRef . . . 22
3.10.2.3 isMaxRef . . . 22
3.10.2.4 torqueRef . . . 22

4 File Documentation **23**
4.1 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h File Reference . . . 23
4.1.1 Detailed Description . . . 24

4.1.2 Function Documentation . . . 25
4.1.2.1 handle_CAN() . . . 25
4.1.2.2 send_CAN_message() . . . 26
4.1.3 Variable Documentation . . . 27
4.1.3.1 enableCAN . . . 27
4.2 CAN_e-Tech.h . . . 27
4.3 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h File Reference . . . 27
4.3.1 Detailed Description . . . 29
4.3.2 Function Documentation . . . 29
4.3.2.1 calc_current_loop() . . . 29
4.3.2.2 calc_current_reference() . . . 29
4.3.2.3 calc_duties() . . . 30
4.3.2.4 saturate_voltage() . . . 31
4.4 CONTROL.h . . . 31
4.5 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/ERRORS.h File Reference . . . 32
4.5.1 Macro Definition Documentation . . . 33
4.5.1.1 OVERCURRENT_TH . . . 33
4.5.1.2 OVERSPEED_TH . . . 33
4.5.1.3 OVERTEMPERATURE_INVERTER_TH . . . 33
4.5.1.4 OVERTEMPERATURE_MOTOR_TH . . . 33
4.5.1.5 OVERVOLTAGE_TH . . . 33
4.5.1.6 UNDERVOLTAGE_TH . . . 33
4.5.2 Enumeration Type Documentation . . . 33
4.5.2.1 InverterError . . . 33
4.5.3 Function Documentation . . . 34
4.5.3.1 clear_error() . . . 34
4.5.3.2 is_error_set() . . . 34
4.5.3.3 set_error() . . . 35
4.6 ERRORS.h . . . 35
4.7 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h File Reference . . . 36
4.7.1 Detailed Description . . . 37
4.7.2 Function Documentation . . . 37
4.7.2.1 eval_inv_FSM() . . . 37
4.8 FSM.h . . . 38
4.9 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h File Reference . . . 38
4.9.1 Detailed Description . . . 40
4.9.2 Macro Definition Documentation . . . 40
4.9.2.1 DT . . . 40
4.9.2.2 TS . . . 40
4.9.3 Enumeration Type Documentation . . . 40
4.9.3.1 InverterState . . . 40
4.9.4 Function Documentation . . . 41

4.9.4.1 disable_control_loops() . . . 41
4.9.4.2 enable_control_loops() . . . 41
4.9.4.3 init_control_loops() . . . 41
4.9.4.4 initialize_inverter() . . . 42
4.9.5 Variable Documentation . . . 43
4.9.5.1 inverter_left . . . 43
4.9.5.2 inverter_right . . . 43
4.10 INVERTER.h . . . 44
4.11 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h File Reference . . . 44
4.11.1 Detailed Description . . . 47
4.11.2 Macro Definition Documentation . . . 47
4.11.2.1 A_L_GPIO_Port . . . 47
4.11.2.2 A_L_Pin . . . 47
4.11.2.3 A_R_GPIO_Port . . . 47
4.11.2.4 A_R_Pin . . . 48
4.11.2.5 B_L_GPIO_Port . . . 48
4.11.2.6 B_L_Pin . . . 48
4.11.2.7 B_R_GPIO_Port . . . 48
4.11.2.8 B_R_Pin . . . 48
4.11.2.9 DAC_GPIO_Port . . . 48
4.11.2.10 DAC_Pin . . . 48
4.11.2.11 DIR_GPIO_Port . . . 48
4.11.2.12 DIR_Pin . . . 48
4.11.2.13 ENABLE_L_GPIO_Port . . . 48
4.11.2.14 ENABLE_L_Pin . . . 49
4.11.2.15 ENABLE_R_GPIO_Port . . . 49
4.11.2.16 ENABLE_R_Pin . . . 49
4.11.2.17 ia_L_GPIO_Port . . . 49
4.11.2.18 ia_L_Pin . . . 49
4.11.2.19 ia_R_GPIO_Port . . . 49
4.11.2.20 ia_R_Pin . . . 49
4.11.2.21 ib_L_GPIO_Port . . . 49
4.11.2.22 ib_L_Pin . . . 49
4.11.2.23 ib_R_GPIO_Port . . . 49
4.11.2.24 ib_R_Pin . . . 50
4.11.2.25 ic_L_GPIO_Port . . . 50
4.11.2.26 ic_L_Pin . . . 50
4.11.2.27 ic_R_GPIO_Port . . . 50
4.11.2.28 ic_R_Pin . . . 50
4.11.2.29 LED_ERR_GPIO_Port . . . 50
4.11.2.30 LED_ERR_Pin . . . 50
4.11.2.31 LED_LEFT_GPIO_Port . . . 50

4.11.2.32 LED_LEFT_Pin . . . 50
4.11.2.33 LED_RIGHT_GPIO_Port . . . 50
4.11.2.34 LED_RIGHT_Pin . . . 51
4.11.2.35 PWM1_L_GPIO_Port . . . 51
4.11.2.36 PWM1_L_Pin . . . 51
4.11.2.37 PWM1_R_GPIO_Port . . . 51
4.11.2.38 PWM1_R_Pin . . . 51
4.11.2.39 PWM2_L_GPIO_Port . . . 51
4.11.2.40 PWM2_L_Pin . . . 51
4.11.2.41 PWM2_R_GPIO_Port . . . 51
4.11.2.42 PWM2_R_Pin . . . 51
4.11.2.43 PWM3_L_GPIO_Port . . . 51
4.11.2.44 PWM3_L_Pin . . . 52
4.11.2.45 PWM3_R_GPIO_Port . . . 52
4.11.2.46 PWM3_R_Pin . . . 52
4.11.2.47 PWM4_L_GPIO_Port . . . 52
4.11.2.48 PWM4_L_Pin . . . 52
4.11.2.49 PWM4_R_GPIO_Port . . . 52
4.11.2.50 PWM4_R_Pin . . . 52
4.11.2.51 PWM5_L_GPIO_Port . . . 52
4.11.2.52 PWM5_L_Pin . . . 52
4.11.2.53 PWM5_R_GPIO_Port . . . 52
4.11.2.54 PWM5_R_Pin . . . 53
4.11.2.55 PWM6_L_GPIO_Port . . . 53
4.11.2.56 PWM6_L_Pin . . . 53
4.11.2.57 PWM6_R_GPIO_Port . . . 53
4.11.2.58 PWM6_R_Pin . . . 53
4.11.2.59 SC_det_GPIO_Port . . . 53
4.11.2.60 SC_det_Pin . . . 53
4.11.2.61 Tinv_L_GPIO_Port . . . 53
4.11.2.62 Tinv_L_Pin . . . 53
4.11.2.63 Tinv_R_GPIO_Port . . . 53
4.11.2.64 Tinv_R_Pin . . . 54
4.11.2.65 Tmot_L_GPIO_Port . . . 54
4.11.2.66 Tmot_L_Pin . . . 54
4.11.2.67 Tmot_R_GPIO_Port . . . 54
4.11.2.68 Tmot_R_Pin . . . 54
4.11.2.69 TRIP_L_GPIO_Port . . . 54
4.11.2.70 TRIP_L_Pin . . . 54
4.11.2.71 TRIP_R_GPIO_Port . . . 54
4.11.2.72 TRIP_R_Pin . . . 54
4.11.2.73 VDC_L_GPIO_Port . . . 54

4.11.2.74 VDC_L_Pin . . . 55
4.11.2.75 VDC_R_GPIO_Port . . . 55
4.11.2.76 VDC_R_Pin . . . 55
4.11.2.77 WRN_L_GPIO_Port . . . 55
4.11.2.78 WRN_L_Pin . . . 55
4.11.2.79 WRN_R_GPIO_Port . . . 55
4.11.2.80 WRN_R_Pin . . . 55
4.11.2.81 Z_L_GPIO_Port . . . 55
4.11.2.82 Z_L_Pin . . . 55
4.11.2.83 Z_R_GPIO_Port . . . 55
4.11.2.84 Z_R_Pin . . . 55
4.11.3 Function Documentation . . . 55
4.11.3.1 Error_Handler() . . . 55
4.12 main.h . . . 56
4.13 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h File Reference . . . 58
4.13.1 Detailed Description . . . 59
4.13.2 Macro Definition Documentation . . . 60
4.13.2.1 CURRENT_OFFSET . . . 60
4.13.2.2 CURRENT_SLOPE . . . 60
4.13.2.3 VOLTAGE_OFFSET . . . 60
4.13.2.4 VOLTAGE_SLOPE . . . 60
4.13.3 Function Documentation . . . 60
4.13.3.1 calibrate_offsets() . . . 60
4.13.3.2 get_currents_voltage() . . . 61
4.13.3.3 get_idiq() . . . 62
4.13.3.4 get_linear() . . . 63
4.13.3.5 get_temperature() . . . 64
4.13.4 Variable Documentation . . . 65
4.13.4.1 rawADC_left . . . 65
4.13.4.2 rawADC_right . . . 65
4.13.4.3 rawADC_temp . . . 65
4.13.4.4 tempInverterLUT . . . 65
4.13.4.5 tempMotorLUT . . . 65
4.14 MEASUREMENTS.h . . . 66
4.15 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h File Reference . . . 66
4.15.1 Detailed Description . . . 68
4.15.2 Function Documentation . . . 68
4.15.2.1 check_motor_parameters() . . . 68
4.15.2.2 precalculate_motor_constants() . . . 68
4.15.3 Variable Documentation . . . 69
4.15.3.1 motor_left . . . 69
4.15.3.2 motor_right . . . 69

4.16 MOTOR.h . . . 69
4.17 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h File Reference . . . 70
4.17.1 Detailed Description . . . 72
4.17.2 Macro Definition Documentation . . . 72
4.17.2.1 DIR_STATE . . . 72
4.17.2.2 DISABLE . . . 72
4.17.2.3 ENABLE . . . 72
4.17.2.4 SC_DET_STATE . . . 72
4.17.2.5 WRN_STATE . . . 72
4.17.3 Enumeration Type Documentation . . . 72
4.17.3.1 LEDMode . . . 72
4.17.4 Function Documentation . . . 73
4.17.4.1 enable_inverters() . . . 73
4.17.4.2 handle_direction() . . . 73
4.17.4.3 handle_LED() . . . 74
4.17.5 Variable Documentation . . . 75
4.17.5.1 led_left . . . 75
4.17.5.2 led_right . . . 75
4.17.5.3 ledError . . . 75
4.18 PCB_IO.h . . . 75
4.19 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h File Reference . . . 76
4.19.1 Detailed Description . . . 77
4.19.2 Function Documentation . . . 77
4.19.2.1 disable_PWM() . . . 77
4.19.2.2 enable_PWM() . . . 77
4.19.2.3 update_PWM() . . . 77
4.20 PWM.h . . . 78
4.21 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h File Reference . . . 78
4.21.1 Detailed Description . . . 80
4.21.2 Macro Definition Documentation . . . 80
4.21.2.1 TEMP_INVERTER_DERATING . . . 80
4.21.2.2 TEMP_INVERTER_MAX . . . 80
4.21.2.3 TEMP_MOTOR_DERATING . . . 81
4.21.2.4 TEMP_MOTOR_MAX . . . 81
4.21.3 Function Documentation . . . 81
4.21.3.1 calculate_derated_current() . . . 81
4.21.3.2 derate_current_reference() . . . 81
4.21.3.3 handle_torqueRef() . . . 82
4.21.3.4 limit_torque_to_prevent_overspeed() . . . 83
4.21.3.5 saturate_symmetric() . . . 84
4.21.3.6 set_torque_direction() . . . 85
4.21.4 Variable Documentation . . . 86

4.21.4.1 torqueRefIn_left . . . 86
4.21.4.2 torqueRefIn_right . . . 86
4.22 REFERENCE.h . . . 86
4.23 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h File Reference . . . 87
4.23.1 Detailed Description . . . 88
4.23.2 Function Documentation . . . 88
4.23.2.1 handle_overtemperature_faults() . . . 88
4.23.2.2 read_temperatures() . . . 89
4.23.2.3 tasks_1ms() . . . 89
4.24 TASKS_1ms.h . . . 90
4.25 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_CRITICAL.h File Reference . . . 91
4.25.1 Detailed Description . . . 91
4.25.2 Function Documentation . . . 92
4.25.2.1 tasks_critical_left() . . . 92
4.25.2.2 tasks_critical_right() . . . 92
4.26 TASKS_CRITICAL.h . . . 93
4.27 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c File Reference . . . 93
4.27.1 Detailed Description . . . 94
4.27.2 Function Documentation . . . 94
4.27.2.1 handle_CAN() . . . 94
4.27.2.2 send_CAN_message() . . . 95
4.27.3 Variable Documentation . . . 96
4.27.3.1 keepAlive . . . 96
4.28 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c File Reference . . . 96
4.28.1 Detailed Description . . . 97
4.28.2 Function Documentation . . . 97
4.28.2.1 calc_current_loop() . . . 97
4.28.2.2 calc_current_reference() . . . 97
4.28.2.3 calc_duties() . . . 98
4.28.2.4 saturate_voltage() . . . 99
4.29 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ERRORS.c File Reference . . . 99
4.29.1 Detailed Description . . . 100
4.29.2 Function Documentation . . . 100
4.29.2.1 clear_error() . . . 100
4.29.2.2 is_error_set() . . . 101
4.29.2.3 set_error() . . . 101
4.30 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c File Reference . . . 101
4.30.1 Detailed Description . . . 102
4.30.2 Function Documentation . . . 103
4.30.2.1 eval_inv_FSM() . . . 103
4.31 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c File Reference . . . 103
4.31.1 Detailed Description . . . 104

4.31.2 Function Documentation . . . 105
4.31.2.1 disable_control_loops() . . . 105
4.31.2.2 enable_control_loops() . . . 105
4.31.2.3 init_control_loops() . . . 105
4.31.2.4 initialize_inverter() . . . 106
4.31.3 Variable Documentation . . . 107
4.31.3.1 inverter_left . . . 107
4.31.3.2 inverter_right . . . 107
4.32 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c File Reference . . . 107
4.32.1 Detailed Description . . . 108
4.32.2 Function Documentation . . . 108
4.32.2.1 Error_Handler() . . . 108
4.32.2.2 main() . . . 109
4.32.2.3 SystemClock_Config() . . . 109
4.33 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MEASUREMENTS.c File Reference . . . 110
4.33.1 Detailed Description . . . 120
4.33.2 Function Documentation . . . 121
4.33.2.1 calibrate_offsets() . . . 121
4.33.2.2 get_currents_voltage() . . . 121
4.33.2.3 get_idiq() . . . 122
4.33.2.4 get_linear() . . . 123
4.33.2.5 get_temperature() . . . 123
4.33.3 Variable Documentation . . . 124
4.33.3.1 rawADC_left . . . 124
4.33.3.2 rawADC_right . . . 124
4.33.3.3 rawADC_temp . . . 124
4.33.3.4 tempInverterLUT . . . 125
4.33.3.5 tempMotorLUT . . . 130
4.34 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTOR.c File Reference . . . 135
4.34.1 Detailed Description . . . 136
4.34.2 Function Documentation . . . 137
4.34.2.1 check_motor_parameters() . . . 137
4.34.2.2 precalculate_motor_constants() . . . 137
4.34.3 Variable Documentation . . . 138
4.34.3.1 motor_left . . . 138
4.34.3.2 motor_right . . . 138
4.35 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB_IO.c File Reference . . . 139
4.35.1 Detailed Description . . . 140
4.35.2 Function Documentation . . . 140
4.35.2.1 enable_inverters() . . . 140
4.35.2.2 handle_direction() . . . 140
4.35.2.3 handle_LED() . . . 141

4.35.3 Variable Documentation . . . 141
4.35.3.1 led_left . . . 141
4.35.3.2 led_right . . . 142
4.35.3.3 ledError . . . 142
4.36 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c File Reference . . . 142
4.36.1 Detailed Description . . . 143
4.36.2 Function Documentation . . . 143
4.36.2.1 disable_PWM() . . . 143
4.36.2.2 enable_PWM() . . . 143
4.36.2.3 update_PWM() . . . 143
4.37 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/REFERENCE.c File Reference . . . 144
4.37.1 Detailed Description . . . 146
4.37.2 Function Documentation . . . 146
4.37.2.1 calculate_derated_current() . . . 146
4.37.2.2 derate_current_reference() . . . 146
4.37.2.3 handle_torqueRef() . . . 147
4.37.2.4 limit_torque_to_prevent_overspeed() . . . 148
4.37.2.5 saturate_symmetric() . . . 149
4.37.2.6 set_torque_direction() . . . 150
4.37.3 Variable Documentation . . . 150
4.37.3.1 torqueRefIn_left . . . 150
4.37.3.2 torqueRefIn_right . . . 150
4.38 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx_it.c File Reference . . . 150
4.38.1 Detailed Description . . . 152
4.38.2 Function Documentation . . . 152
4.38.2.1 BusFault_Handler() . . . 152
4.38.2.2 CAN1_RX0_IRQHandler() . . . 152
4.38.2.3 DebugMon_Handler() . . . 153
4.38.2.4 DMA2_Stream0_IRQHandler() . . . 153
4.38.2.5 DMA2_Stream1_IRQHandler() . . . 153
4.38.2.6 DMA2_Stream2_IRQHandler() . . . 154
4.38.2.7 HAL_ADC_ConvCpltCallback() . . . 154
4.38.2.8 HardFault_Handler() . . . 154
4.38.2.9 MemManage_Handler() . . . 154
4.38.2.10 NMI_Handler() . . . 154
4.38.2.11 PendSV_Handler() . . . 155
4.38.2.12 SVC_Handler() . . . 155
4.38.2.13 SysTick_Handler() . . . 155
4.38.2.14 TIM1_UP_TIM10_IRQHandler() . . . 155
4.38.2.15 TIM6_DAC_IRQHandler() . . . 155
4.38.2.16 UsageFault_Handler() . . . 155
4.38.3 Variable Documentation . . . 155

4.38.3.1 hcan1 155
4.38.3.2 hdac 156
4.38.3.3 hdma_adc1 156
4.38.3.4 hdma_adc2 156
4.38.3.5 hdma_adc3 156
4.38.3.6 htim1 156
4.38.3.7 htim6 156
4.39 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_1ms.c File Reference 156
4.39.1 Detailed Description 157
4.39.2 Function Documentation 158
4.39.2.1 handle_overtemperature_faults() 158
4.39.2.2 read_temperatures() 158
4.39.2.3 tasks_1ms() 159
4.40 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_CRITICAL.c File Reference 160
4.40.1 Detailed Description 161
4.40.2 Function Documentation 161
4.40.2.1 tasks_critical_left() 161
4.40.2.2 tasks_critical_right() 162
4.40.3 Variable Documentation 162
4.40.3.1 angle_left 162
4.40.3.2 elapsed_ticks 162
4.40.3.3 freqRamp_left 162
4.40.3.4 start_ticks 162

Index **163**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Analog
Structure for ADC measurements in units 5
CANMessageInfo 6
Duties
Structure to hold PWM configuration parameters 7
Encoder
Structure for encoder reading 8
Feedback
Structure for feedback values 9
InverterStruct
Inverter structure 11
LED
LED structure 15
MotorConstants
Structure to hold precomputed motor constants 16
MotorParameters
Structure to hold motor parameters 19
Reference
Structure for reference values 22

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h
Header file for handling CAN communication with the car 23
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h
Header file for the control loop 27
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/ERRORS.h 32
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h
Header for the inverter Finite State Machine 36
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h
Header file for the inverter struct and extern variables 38
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h
: Header for main.c file. This file contains the common defines of the application 44
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h
Header file for handling measurements 58
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h
Header file for motor parameters 66
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h
Header file for handling GPIOs 70
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h
Header file for controlling PWM output 76
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h
Header file for torque reference handling 78
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h
Header file for functions related to tasks executed every 1ms 87
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_CRITICAL.h
Header file for functions related to tasks executed in each PWM timer interruption 91
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c
This file contains functions to handle CAN communication with the car 93
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c
This file provides code for the control loop 96
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ERRORS.c
Header file for the necessary components to set, read and clear ERRORS 99
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c
This file provides code for the inverter Finite State Machine 101
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c
This file provides code for the inverter struct 103

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c
: Main program body 107
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MEASUREMENTS.c
This file provides functions for handling measurements 110
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTOR.c
Source file for motor parameters 135
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB_IO.c
This file provides functions for handling GPIOs 139
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c
This file provides functions for controlling PWM output 142
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/REFERENCE.c
Source file for torque reference handling 144
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx_it.c
Interrupt Service Routines 150
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_1ms.c
This file contains functions to execute tasks every 1ms 156
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_CRITICAL.c
This file contains functions executed in each PWM timer interruption 160

Chapter 3

Data Structure Documentation

3.1 Analog Struct Reference

Structure for ADC measurements in units.

```
#include <MEASUREMENTS.h>
```

Data Fields

- float ia
- float ib
- float ic
- float vDC
- float currentOffsets [3]

3.1.1 Detailed Description

Structure for ADC measurements in units.

3.1.2 Field Documentation

3.1.2.1 currentOffsets

```
float currentOffsets[3]
```

Offsets for the current measurements

3.1.2.2 ia

```
float ia
```

Phase A current in A

3.1.2.3 ib

```
float ib
```

Phase B current in A

3.1.2.4 ic

```
float ic
```

Phase C current in A

3.1.2.5 vDC

```
float vDC
```

DC link voltage in V

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h

3.2 CANMessageInfo Struct Reference

```
#include <CAN_e-Tech.h>
```

Data Fields

- const uint32_t ID
- const uint8_t IDE
- const uint8_t DLC
- const signal_positioned * getSig

3.2.1 Field Documentation

3.2.1.1 DLC

```
const uint8_t DLC
```

3.2.1.2 getSig

```
const signal_positioned* getSig
```

3.2.1.3 ID

```
const uint32_t ID
```

3.2.1.4 IDE

```
const uint8_t IDE
```

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h

3.3 Duties Struct Reference

Structure to hold PWM configuration parameters.

```
#include <PWM.h>
```

Data Fields

- float Da
- float Db
- float Dc

3.3.1 Detailed Description

Structure to hold PWM configuration parameters.

3.3.2 Field Documentation

3.3.2.1 Da

```
float Da
```

Duty cycle for channel 1

3.3.2.2 Db

```
float Db
```

Duty cycle for channel 2

3.3.2.3 Dc

```
float Dc
```

Duty cycle for channel 3

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h

3.4 Encoder Struct Reference

Structure for encoder reading.

```
#include <MEASUREMENTS.h>
```

Data Fields

- uint16_t A
- uint16_t B
- uint16_t Z
- float we
- float theta_e
- float sinTheta_e
- float cosTheta_e
- uint8_t directionMeas

3.4.1 Detailed Description

Structure for encoder reading.

3.4.2 Field Documentation

3.4.2.1 A

```
uint16_t A
```

Encoder channel A value

3.4.2.2 B

```
uint16_t B
```

Encoder channel B value

3.4.2.3 cosTheta_e

```
float cosTheta_e
```

Electrical rotor position cosine

3.4.2.4 directionMeas

```
uint8_t directionMeas
```

Measured direction

3.4.2.5 sinTheta_e

```
float sinTheta_e
```

Electrical rotor position sine

3.4.2.6 theta_e

```
float theta_e
```

Electrical rotor position

3.4.2.7 we

```
float we
```

Electrical angular velocity

3.4.2.8 Z

```
uint16_t Z
```

Encoder channel Z value

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h

3.5 Feedback Struct Reference

Structure for feedback values.

```
#include <MEASUREMENTS.h>
```

Data Fields

- float idMeas
- float iqMeas
- float torqueCalc
- float speedMeas

3.5.1 Detailed Description

Structure for feedback values.

3.5.2 Field Documentation

3.5.2.1 idMeas

```
float idMeas
```

Measured d-axis current in A

3.5.2.2 iqMeas

```
float iqMeas
```

Measured q-axis current in A

3.5.2.3 speedMeas

```
float speedMeas
```

Measured speed in RPM

3.5.2.4 torqueCalc

```
float torqueCalc
```

Calculated torque in N·m

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h

3.6 InverterStruct Struct Reference

Inverter structure.

```
#include <INVERTER.h>
```

Collaboration diagram for InverterStruct:

Data Fields

- LED ∗ led
- GPIO_TypeDef ∗ enable_port
- uint16_t enable_pin
- TIM_HandleTypeDef ∗ htim
- ADC_HandleTypeDef ∗ hadc
- InverterState state
- Analog analog
- Encoder encoder
- Feedback feedback
- Reference reference
- Duties duties
- int8_t direction
- float tempInverter
- float tempMotor
- MotorParameters ∗ motor
- pi_struct idLoop
- pi_struct iqLoop
- float vsMax
- float vd
- float vq
- pi_struct speedLoop
- InverterError errors
- bool enable
- bool enableSW

3.6.1 Detailed Description

Inverter structure.

3.6.2 Field Documentation

3.6.2.1 analog

`Analog analog`

Structure for phase currents and DC voltage measurements

3.6.2.2 direction

`int8_t direction`

Motor direction: 1 CW, -1 CCW, 0 stopped

3.6.2.3 duties

`Duties duties`

Structure for duty cycles for phases A, B, and C

3.6.2.4 enable

`bool enable`

Enable bit for transitioning states

3.6.2.5 enable_pin

`uint16_t enable_pin`

Pin number for enabling/disabling the inverter

3.6.2.6 enable_port

`GPIO_TypeDef* enable_port`

Pointer to GPIO port for enabling/disabling the inverter

3.6.2.7 enableSW

`bool enableSW`

External enable order (needs HW enable to set inv.enable to 1, and if the FAULT state is entered, enableSW must be set to 0 to transition to the IDLE state)

3.6.2.8 encoder

`Encoder encoder`

Structure for encoder input

3.6.2.9 errors

`InverterError errors`

Error field storing error bits, using InverterError enum

3.6.2.10 feedback

`Feedback feedback`

Structure for measured currents and calculated mechanical torque and speed

3.6.2.11 hadc

`ADC_HandleTypeDef* hadc`

Handle of the ADC peripheral for current phase currents and DC voltage sensing

3.6.2.12 htim

`TIM_HandleTypeDef* htim`

Handle of the timer peripheral for PWM output

3.6.2.13 idLoop

`pi_struct idLoop`

PI controller for d-axis current

3.6.2.14 iqLoop

`pi_struct iqLoop`

PI controller for q-axis current

3.6.2.15 led

`LED* led`

Pointer to LED control structure

3.6.2.16 motor

`MotorParameters* motor`

Motor parameters struct

3.6.2.17 reference

`Reference reference`

Structure for referece currents and torque

3.6.2.18 speedLoop

`pi_struct speedLoop`

PI controller for motor speed

3.6.2.19 state

`InverterState state`

Current state of inverter operation

3.6.2.20 tempInverter

`float tempInverter`

Semiconductor temperature in degC

3.6.2.21 tempMotor

`float tempMotor`

Motor temperature in degC

3.6.2.22 vd

`float vd`

d-axis voltage

3.6.2.23 vq

`float vq`

q-axis voltage

3.6.2.24 vsMax

```
float vsMax
```

Maximum output voltage, should be calculated as vDC / sqrt3 in volts

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h

3.7 LED Struct Reference

LED structure.

```
#include <PCB_IO.h>
```

Data Fields

- GPIO_TypeDef * port
- uint16_t pin
- LEDMode mode

3.7.1 Detailed Description

LED structure.

3.7.2 Field Documentation

3.7.2.1 mode

```
LEDMode mode
```

Current LED mode

3.7.2.2 pin

```
uint16_t pin
```

Pin number for controlling the LED

3.7.2.3 port

```
GPIO_TypeDef* port
```

GPIO port for controlling the LED

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h

3.8 MotorConstants Struct Reference

Structure to hold precomputed motor constants.

```
#include <MOTOR.h>
```

Data Fields

- float threePpLambda
- float threePpLdMinusLq
- float invThreePpLambda
- float isc
- float torqueBase
- float invTorqueBase
- float xi
- float xiSquared
- float oneMinusXi
- float twoMinusXi
- float fourTimesOneMinusXi
- float eightTimesOneMinusXiSquared
- float twoMinusXiSquared
- float twoTimesOneMinusXiOnePlusXiSquared
- float twoTimesOneMinusXiXiSquared
- float fourTimesOneMinusXiOnePlusXiSquared
- float fourTimesOneMinusXiXiSquared
- float lambdaDivLqMinusLd
- float betaMinusIsc

3.8.1 Detailed Description

Structure to hold precomputed motor constants.

3.8.2 Field Documentation

3.8.2.1 betaMinusIsc

```
float betaMinusIsc
```

lambda / (Lq - Ld) - lambda / Ld

3.8.2.2 eightTimesOneMinusXiSquared

```
float eightTimesOneMinusXiSquared
```

8 $*$ (1 - Lq / Ld)^2

3.8.2.3 fourTimesOneMinusXi

```
float fourTimesOneMinusXi
```

4 ∗ (1 - Lq / Ld)

3.8.2.4 fourTimesOneMinusXiOnePlusXiSquared

```
float fourTimesOneMinusXiOnePlusXiSquared
```

4 ∗ (1 - Lq / Ld) ∗ (1 + (Lq / Ld)^2)

3.8.2.5 fourTimesOneMinusXiXiSquared

```
float fourTimesOneMinusXiXiSquared
```

4 ∗ (1 - Lq / Ld) ∗ (Lq / Ld)^2

3.8.2.6 invThreePpLambda

```
float invThreePpLambda
```

1 / (3 ∗ pp ∗ lambda)

3.8.2.7 invTorqueBase

```
float invTorqueBase
```

1 / (3 ∗ pp ∗ lambda^2 / Ld)

3.8.2.8 isc

```
float isc
```

-lambda / Ld

3.8.2.9 lambdaDivLqMinusLd

```
float lambdaDivLqMinusLd
```

lambda / (Lq - Ld)

3.8.2.10 oneMinusXi

```
float oneMinusXi
```

1 - Lq / Ld

3.8.2.11 threePpLambda

```
float threePpLambda
```

3 ∗ pp ∗ lambda

3.8.2.12 threePpLdMinusLq

```
float threePpLdMinusLq
```

3 ∗ pp ∗ (Ld - Lq)

3.8.2.13 torqueBase

```
float torqueBase
```

3 ∗ pp ∗ lambda^2 / Ld

3.8.2.14 twoMinusXi

```
float twoMinusXi
```

2 - Lq / Ld

3.8.2.15 twoMinusXiSquared

```
float twoMinusXiSquared
```

(2 - Lq / Ld)^2

3.8.2.16 twoTimesOneMinusXiOnePlusXiSquared

```
float twoTimesOneMinusXiOnePlusXiSquared
```

2 ∗ (1 - Lq / Ld) ∗ (1 + (Lq / Ld)^2)

3.8.2.17 twoTimesOneMinusXiXiSquared

```
float twoTimesOneMinusXiXiSquared
```

2 ∗ (1 - Lq / Ld) ∗ (Lq / Ld)^2

3.8.2.18 xi

```
float xi
```

Lq / Ld

3.8.2.19 xiSquared

```
float xiSquared
```

(Lq / Ld)^2

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h

3.9 MotorParameters Struct Reference

Structure to hold motor parameters.

```
#include <MOTOR.h>
```

Collaboration diagram for MotorParameters:

Data Fields

- float Ld
- float Lq
- float Rs
- float lambda
- uint8_t pp
- float J
- float b
- float torqueMax
- float dTorqueMax
- float speedMax_RPM
- float iMax
- float vDCMax
- MotorConstants constants

3.9.1 Detailed Description

Structure to hold motor parameters.

3.9.2 Field Documentation

3.9.2.1 b

```
float b
```

Viscous friction in N·m·s

3.9.2.2 constants

```
MotorConstants constants
```

Precomputed motor constants

3.9.2.3 dTorqueMax

```
float dTorqueMax
```

Maximum torque increment in N·m/s

3.9.2.4 iMax

```
float iMax
```

Maximum phase current (peak value, or RMS∗sqrt2)

3.9.2.5 J

```
float J
```

Rotational inertia in N·m·s^2

3.9.2.6 lambda

```
float lambda
```

Magnet flux linkage measured V_pk_ph-n · s (phase-neutral peak voltage divided by electrical speed in rad/s)

3.9.2.7 Ld

```
float Ld
```

D-axis inductance in Henries

3.9.2.8 Lq

```
float Lq
```

Q-axis inductance in Henries

3.9.2.9 pp

```
uint8_t pp
```

Pole pairs (total number of poles divided by 2)

3.9.2.10 Rs

```
float Rs
```

Stator resistance in Ohms

3.9.2.11 speedMax_RPM

```
float speedMax_RPM
```

Maximum speed in RPM

3.9.2.12 torqueMax

```
float torqueMax
```

Maximum torque in N·m

3.9.2.13 vDCMax

```
float vDCMax
```

Maximum DC bus voltage in volts

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h

3.10 Reference Struct Reference

Structure for reference values.

```
#include <REFERENCE.h>
```

Data Fields

- float idRef
- float iqRef
- float isMaxRef
- float torqueRef

3.10.1 Detailed Description

Structure for reference values.

3.10.2 Field Documentation

3.10.2.1 idRef

```
float idRef
```

Reference d-axis current in A

3.10.2.2 iqRef

```
float iqRef
```

Reference q-axis current in A

3.10.2.3 isMaxRef

```
float isMaxRef
```

Maximum reference current in A

3.10.2.4 torqueRef

```
float torqueRef
```

Reference torque in N·m

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h

Chapter 4

File Documentation

4.1 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h File Reference

Header file for handling CAN communication with the car.

```
#include "CAN1db.h"
```

Include dependency graph for CAN_e-Tech.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct CANMessageInfo

Functions

- void handle_CAN (CAN_HandleTypeDef ∗hcan)

  Handle CAN messages.
- void send_CAN_message (CAN_HandleTypeDef ∗hcan, void ∗dbc_msg, const float ∗data)

  Send a CAN message using CAN1db.h information.

Variables

- uint8_t enableCAN

4.1.1 Detailed Description

Header file for handling CAN communication with the car.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.1.2 Function Documentation

4.1.2.1 handle_CAN()

```
void handle_CAN (
            CAN_HandleTypeDef * hcan )
```

Handle CAN messages.

This function implements the logic to handle received CAN messages.

Parameters

hcan	Pointer to the CAN handle structure.

Here is the call graph for this function:

Here is the caller graph for this function:

4.1.2.2 send_CAN_message()

```
void send_CAN_message (
            CAN_HandleTypeDef * hcan,
            void * dbc_msg,
            const float * data )
```

Send a CAN message using CAN1db.h information.

This function prepares and sends a CAN message using information from CAN1db.h.

Parameters

hcan	Pointer to the CAN handle structure.
dbc_msg	Pointer to the structure containing CAN message information from CAN1db.h.
data	Pointer to the array of float data to be sent.

Here is the caller graph for this function:

4.1.3 Variable Documentation

4.1.3.1 enableCAN

```
uint8_t enableCAN  [extern]
```

4.2 CAN_e-Tech.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef CAN_E_TECH_H
00021 #define CAN_E_TECH_H
00022
00023 #include "CAN1db.h" // needs the CAN1db and its types
00024
00025 extern uint8_t enableCAN;
00026
00027 typedef struct {
00028     const uint32_t ID;
00029     const uint8_t IDE;
00030     const uint8_t DLC;
00031     const signal_positioned *getSig;
00032 } CANMessageInfo;
00033
00041 void handle_CAN(CAN_HandleTypeDef *hcan);
00042
00052 void send_CAN_message(CAN_HandleTypeDef *hcan, void *dbc_msg, const float *data);
00053
00054 #endif /* CAN_E_TECH_H */
```

4.3 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h File Reference

Header file for the control loop.

```
#include "PWM.h"
#include "INVERTER.h"
```

Include dependency graph for CONTROL.h:

This graph shows which files directly or indirectly include this file:

Functions

- void calc_current_reference (MotorParameters ∗motor, volatile Reference ∗reference)

  Calculates the current references using a FOC algorithm. It computes the current vector for the MTPA trajectory and limits the current reference to isMaxRef (calculated by derating, starting from the motor's maximum current). The MTPV trajectory is not implemented to save some computation time due to the nature of the motors expected. In order to implement field weakening, an external voltage loop modifying gammaRef is needed and should be called inside here. When implementing field weakening, special attention must be put to the torque reference being near 0 or differing from the speed sign (regeneration). A minimum id current must be set for speeds higher than Vs/lambda. Study thoroughly, simulate first.
- void calc_current_loop (volatile InverterStruct ∗inv)

Calculates the id-iq loops.

- void saturate_voltage (volatile InverterStruct *inv)

  Saturates PI output to not surpass DC voltage.

- void calc_duties (float vd, float vq, float vDC, float sinTheta_e, float cosTheta_e, volatile Duties *duties)

  function.

4.3.1 Detailed Description

Header file for the control loop.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.3.2 Function Documentation

4.3.2.1 calc_current_loop()

```
void calc_current_loop (
            volatile InverterStruct * inv )
```

Calculates the id-iq loops.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.3.2.2 calc_current_reference()

```
void calc_current_reference (
            MotorParameters * motor,
            volatile Reference * reference )
```

Calculates the current references using a FOC algorithm. It computes the current vector for the MTPA trajectory and limits the current reference to isMaxRef (calculated by derating, starting from the motor's maximum current). The MTPV trajectory is not implemented to save some computation time due to the nature of the motors expected. In order to implement field weakening, an external voltage loop modifying gammaRef is needed and should be called inside here. When implementing field weakening, special attention must be put to the torque reference being near 0 or differing from the speed sign (regeneration). A minimum id current must be set for speeds higher than Vs/lambda. Study thoroughly, simulate first.

Parameters

in	*motor*	Pointer to the motor parameters structure.
in,out	*reference*	Pointer to the reference struct.

Here is the caller graph for this function:

4.3.2.3 calc_duties()

```
void calc_duties (
            float vd,
            float vq,
            float vDC,
            float sinTheta_e,
            float cosTheta_e,
            volatile Duties * duties )
```

function.

This function calculates the inverse Park transform and the duty cycles using SVPWM

Parameters

in	*vd*	Voltage in the d-axis.
in	*vq*	Voltage in the q-axis.
in	*vDC*	DC voltage.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)
in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)
out	*duties*	Pointer to the duties structure.

Here is the caller graph for this function:

4.3.2.4 saturate_voltage()

```
void saturate_voltage (
            volatile InverterStruct * inv )
```

Saturates PI output to not surpass DC voltage.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.4 CONTROL.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

#ifndef CONTROL_H
#define CONTROL_H

#include "PWM.h" // duties struct
#include "INVERTER.h" // TS & Inverter struct

void calc_current_reference(MotorParameters * motor, volatile Reference * reference);

void calc_current_loop(volatile InverterStruct *inv);

void saturate_voltage(volatile InverterStruct *inv);

void calc_duties(float vd, float vq, float vDC, float sinTheta_e, float cosTheta_e, volatile Duties
*duties);

#endif /* CONTROL_H */
```

4.5 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/ERRORS.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
```

Include dependency graph for ERRORS.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define OVERTEMPERATURE_INVERTER_TH 60.0f
- #define OVERVOLTAGE_TH 600.0f
- #define OVERCURRENT_TH 100.0f
- #define OVERSPEED_TH 20000.0f
- #define UNDERVOLTAGE_TH 10.0f
- #define OVERTEMPERATURE_MOTOR_TH 90.0f

Enumerations

- enum InverterError {
  NONE = 0 , POWER_FAULT = (1 << 0) , OVERTEMPERATURE_INV = (1 << 1) , OVERVOLTAGE = (1 << 2) ,
  OVERCURRENT = (1 << 3) , OVERSPEED = (1 << 4) , UNDERVOLTAGE = (1 << 5) , CONTROL_FAULT = (1 << 6) ,
  WARNING = (1 << 7) , OVERTEMPERATURE_MOT = (1 << 8) , FEEDBACK_FAULT = (1 << 9) }

  Enumeration of inverter error states.

Functions

- void set_error (volatile void *data, InverterError error)
  Sets an error in the error field of a data structure.
- void clear_error (volatile void *data, InverterError error)
  Clears an error in the error field of a data structure.
- bool is_error_set (volatile void *data, InverterError error)
  Checks if an error is set in the error field of a data structure.

4.5.1 Macro Definition Documentation

4.5.1.1 OVERCURRENT_TH

```
#define OVERCURRENT_TH 100.0f
```

[A] Threshold for instantaneous overcurrent fault

4.5.1.2 OVERSPEED_TH

```
#define OVERSPEED_TH 20000.0f
```

[RPM] Threshold for motor overspeed fault

4.5.1.3 OVERTEMPERATURE_INVERTER_TH

```
#define OVERTEMPERATURE_INVERTER_TH 60.0f
```

[degC] Threshold for inverter overtemperature fault

4.5.1.4 OVERTEMPERATURE_MOTOR_TH

```
#define OVERTEMPERATURE_MOTOR_TH 90.0f
```

[degC] Threshold for motor overtemperature fault

4.5.1.5 OVERVOLTAGE_TH

```
#define OVERVOLTAGE_TH 600.0f
```

[V] Threshold for overvoltage fault

4.5.1.6 UNDERVOLTAGE_TH

```
#define UNDERVOLTAGE_TH 10.0f
```

[V] Threshold for undervoltage fault

4.5.2 Enumeration Type Documentation

4.5.2.1 InverterError

```
enum InverterError
```

Enumeration of inverter error states.

Enumerator

NONE	
POWER_FAULT	Power fault error bit
OVERTEMPERATURE_INV	Inverter overtemperature error bit
OVERVOLTAGE	Overvoltage error bit
OVERCURRENT	Overcurrent error bit
OVERSPEED	Overspeed error bit
UNDERVOLTAGE	Undervoltage error bit
CONTROL_FAULT	Control fault error bit
WARNING	Warning error bit
OVERTEMPERATURE_MOT	Motor overtemperature error bit
FEEDBACK_FAULT	Feedback fault error bit

4.5.3 Function Documentation

4.5.3.1 clear_error()

```
void clear_error (
            volatile void * data,
            InverterError error )
```

Clears an error in the error field of a data structure.

This function clears the specified error bit in the error field of a data structure.

Parameters

out	*data*	Pointer to the data structure containing the error field.
in	*error*	The error to be cleared. This should be one of the values from the InverterError enumeration.

Here is the caller graph for this function:

4.5.3.2 is_error_set()

```
bool is_error_set (
            volatile void * data,
            InverterError error )
```

Checks if an error is set in the error field of a data structure.

This function checks if the specified error bit is set in the error field of a data structure.

Parameters

in	*data*	Pointer to the data structure containing the error field.
in	*error*	The error to be checked. This should be one of the values from the InverterError enumeration.

Returns

true if the specified error is set, false otherwise.

4.5.3.3 set_error()

```
void set_error (
            volatile void * data,
            InverterError error )
```

Sets an error in the error field of a data structure.

This function sets the specified error bit in the error field of a data structure.

Parameters

out	*data*	Pointer to the data structure containing the error field.
in	*error*	The error to be set. This should be one of the values from the InverterError enumeration.

Here is the caller graph for this function:

4.6 ERRORS.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019 #include <stdbool.h>
00020 #include <stdint.h>
00021
00025 typedef enum {
00026     NONE = 0,
00027     POWER_FAULT = (1 « 0),
00028     OVERTEMPERATURE_INV = (1 « 1),
00029     OVERVOLTAGE = (1 « 2),
00030     OVERCURRENT = (1 « 3),
00031     OVERSPEED = (1 « 4),
00032     UNDERVOLTAGE = (1 « 5),
00033     CONTROL_FAULT = (1 « 6),
00034     WARNING = (1 « 7),
00035     OVERTEMPERATURE_MOT = (1 « 8),
00036     FEEDBACK_FAULT = (1 « 9)
00037 } InverterError;
00038
00039 /* Define fault thresholds */
```

```
#define OVERTEMPERATURE_INVERTER_TH 60.0f
#define OVERVOLTAGE_TH 600.0f
#define OVERCURRENT_TH 100.0f
#define OVERSPEED_TH 20000.0f
#define UNDERVOLTAGE_TH 10.0f
#define OVERTEMPERATURE_MOTOR_TH 90.0f
void set_error(volatile void *data, InverterError error);

void clear_error(volatile void *data, InverterError error);

bool is_error_set(volatile void *data, InverterError error);
```

4.7 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h File Reference

Header for the inverter Finite State Machine.

`#include "INVERTER.h"`

Include dependency graph for FSM.h:

This graph shows which files directly or indirectly include this file:

Functions

- void eval_inv_FSM (volatile InverterStruct *inv)

  Run the Finite State Machine (FSM) for inverter operation control.

4.7.1 Detailed Description

Header for the inverter Finite State Machine.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

4.7.2 Function Documentation

4.7.2.1 eval_inv_FSM()

```
void eval_inv_FSM (
            volatile InverterStruct * inv )
```

Run the Finite State Machine (FSM) for inverter operation control.

Parameters

inv	Pointer to the inverter structure.

Run the Finite State Machine (FSM) for inverter operation control.

This function executes the finite state machine to control the inverter based on its current state.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.8 FSM.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef FSM_H
00021 #define FSM_H
00022 #include "INVERTER.h" // inverter struct
00023
00024
00030 void eval_inv_FSM(volatile InverterStruct *inv);
00031
00032 #endif /* FSM_H */
```

4.9 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h File Reference

Header file for the inverter struct and extern variables.

```
#include "PCB_IO.h"
#include "MEASUREMENTS.h"
#include "REFERENCE.h"
#include "MOTOR.h"
```

```
#include "PWM.h"
```

Include dependency graph for INVERTER.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct InverterStruct

  Inverter structure.

Macros

- #define TS 0.000025
- #define DT 0.0000005

Enumerations

- enum InverterState { INV_STATE_IDLE , INV_STATE_STARTUP , INV_STATE_RUNNING , INV_STATE_FAULT }

  Enumeration of inverter operation states.

Functions

- void initialize_inverter (volatile InverterStruct ∗inv, LED ∗led, GPIO_TypeDef ∗enable_port, uint16_t enable↩_pin, TIM_HandleTypeDef ∗htim, ADC_HandleTypeDef ∗hadc, MotorParameters ∗motor, volatile uint16_t ∗rawADC)

  Initialize the inverter.
- void init_control_loops (volatile InverterStruct ∗inv, MotorParameters ∗motor)

  Initializes the id-iq current control PI controllers.
- void enable_control_loops (volatile InverterStruct ∗inv)

  Enables the PI controllers.
- void disable_control_loops (volatile InverterStruct ∗inv)

  Disables the PI controllers.

Variables

- volatile InverterStruct inverter_left

  Left inverter structure.
- volatile InverterStruct inverter_right

  Right inverter structure.

4.9.1 Detailed Description

Header file for the inverter struct and extern variables.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.9.2 Macro Definition Documentation

4.9.2.1 DT

```
#define DT 0.0000005
```

Dead time in seconds (500 ns), time in which both top and bottom transistors are open

4.9.2.2 TS

```
#define TS 0.000025
```

Switching time in seconds (25 us), inverse of the switching frequency of 40 kHz

4.9.3 Enumeration Type Documentation

4.9.3.1 InverterState

```
enum InverterState
```

Enumeration of inverter operation states.

Enumerator

INV_STATE_IDLE	Inverter idle state
INV_STATE_STARTUP	Inverter startup state
INV_STATE_RUNNING	Inverter running state
INV_STATE_FAULT	Inverter fault state

4.9.4 Function Documentation

4.9.4.1 disable_control_loops()

```
void disable_control_loops (
            volatile InverterStruct * inv )
```

Disables the PI controllers.

Parameters

inv	Pointer to the inverter structure.

4.9.4.2 enable_control_loops()

```
void enable_control_loops (
            volatile InverterStruct * inv )
```

Enables the PI controllers.

Parameters

inv	Pointer to the inverter structure.

4.9.4.3 init_control_loops()

```
void init_control_loops (
            volatile InverterStruct * inv,
            MotorParameters * motor )
```

Initializes the id-iq current control PI controllers.

Parameters

inv	Pointer to the inverter structure.

Initializes the id-iq current control PI controllers.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.9.4.4 initialize_inverter()

```
void initialize_inverter (
            volatile InverterStruct * inv,
            LED * led,
            GPIO_TypeDef * enable_port,
            uint16_t enable_pin,
            TIM_HandleTypeDef * htim,
            ADC_HandleTypeDef * hadc,
            MotorParameters * motor,
            volatile uint16_t * rawADC )
```

Initialize the inverter.

This function initializes the inverter structure with the specified LED, GPIO port, and pin.

Parameters

out	*inv*	Pointer to the inverter structure.
in	*led*	Pointer to the LED structure.
in	*enable_port*	Pointer to the GPIO port for enabling/disabling the inverter.
in	*enable_pin*	Pin number for enabling/disabling the inverter.
in	*htim*	Timer peripheral for the PWM output.
in	*hadc*	ADC peripheral for the current phase current and DC voltage sensing.
in	*motor*	MotorParameters struct.

Here is the call graph for this function:

Here is the caller graph for this function:

4.9.5 Variable Documentation

4.9.5.1 inverter_left

```
volatile InverterStruct inverter_left  [extern]
```

Left inverter structure.

External declaration of the left inverter structure

External declaration of the left inverter structure.

4.9.5.2 inverter_right

```
volatile InverterStruct inverter_right  [extern]
```

Right inverter structure.

External declaration of the right inverter structure

External declaration of the right inverter structure.

4.10 INVERTER.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

#ifndef INVERTER_H
#define INVERTER_H

#include "PCB_IO.h" // peripheral types
#include "MEASUREMENTS.h" // a few structs
#include "REFERENCE.h" // reference struct
#include "MOTOR.h" // motor struct
#include "PWM.h" // duties struct


#define TS 0.000025
#define DT 0.0000005
typedef enum {
    INV_STATE_IDLE,
    INV_STATE_STARTUP,
    INV_STATE_RUNNING,
    INV_STATE_FAULT
} InverterState;

typedef struct {
    LED *led;
    GPIO_TypeDef *enable_port;
    uint16_t enable_pin;
    TIM_HandleTypeDef *htim;
    ADC_HandleTypeDef *hadc;
    InverterState state;
    Analog analog;
    Encoder encoder;
    Feedback feedback;
    Reference reference;
    Duties duties;
    int8_t direction;
    float tempInverter;
    float tempMotor;
    MotorParameters *motor;
    pi_struct idLoop;
    pi_struct iqLoop;
    float vsMax;
    float vd;
    float vq;
    pi_struct speedLoop;
    InverterError errors;
    bool enable;
    bool enableSW;
} InverterStruct;

extern volatile InverterStruct inverter_left;
extern volatile InverterStruct inverter_right;
void initialize_inverter(volatile InverterStruct *inv, LED *led, GPIO_TypeDef *enable_port, uint16_t
enable_pin, TIM_HandleTypeDef *htim, ADC_HandleTypeDef *hadc, MotorParameters *motor, volatile
uint16_t *rawADC);

void init_control_loops(volatile InverterStruct *inv, MotorParameters *motor);

void enable_control_loops(volatile InverterStruct *inv);

void disable_control_loops(volatile InverterStruct *inv);

#endif /* INVERTER_H */
```

4.11 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f7xx_hal.h"
```

Include dependency graph for main.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define Tinv_L_Pin GPIO_PIN_0
- #define Tinv_L_GPIO_Port GPIOC
- #define Tinv_R_Pin GPIO_PIN_1
- #define Tinv_R_GPIO_Port GPIOC
- #define Tmot_L_Pin GPIO_PIN_2
- #define Tmot_L_GPIO_Port GPIOC
- #define Tmot_R_Pin GPIO_PIN_3
- #define Tmot_R_GPIO_Port GPIOC
- #define ia_L_Pin GPIO_PIN_0
- #define ia_L_GPIO_Port GPIOA
- #define ib_L_Pin GPIO_PIN_1
- #define ib_L_GPIO_Port GPIOA
- #define ic_L_Pin GPIO_PIN_2
- #define ic_L_GPIO_Port GPIOA
- #define VDC_L_Pin GPIO_PIN_3
- #define VDC_L_GPIO_Port GPIOA
- #define DAC_Pin GPIO_PIN_4

- #define DAC_GPIO_Port GPIOA
- #define PWM1_R_Pin GPIO_PIN_5
- #define PWM1_R_GPIO_Port GPIOA
- #define ia_R_Pin GPIO_PIN_6
- #define ia_R_GPIO_Port GPIOA
- #define ib_R_Pin GPIO_PIN_7
- #define ib_R_GPIO_Port GPIOA
- #define SC_det_Pin GPIO_PIN_4
- #define SC_det_GPIO_Port GPIOC
- #define ic_R_Pin GPIO_PIN_0
- #define ic_R_GPIO_Port GPIOB
- #define VDC_R_Pin GPIO_PIN_1
- #define VDC_R_GPIO_Port GPIOB
- #define ENABLE_R_Pin GPIO_PIN_2
- #define ENABLE_R_GPIO_Port GPIOB
- #define ENABLE_L_Pin GPIO_PIN_7
- #define ENABLE_L_GPIO_Port GPIOE
- #define PWM1_L_Pin GPIO_PIN_8
- #define PWM1_L_GPIO_Port GPIOE
- #define PWM2_L_Pin GPIO_PIN_9
- #define PWM2_L_GPIO_Port GPIOE
- #define PWM3_L_Pin GPIO_PIN_10
- #define PWM3_L_GPIO_Port GPIOE
- #define PWM4_L_Pin GPIO_PIN_11
- #define PWM4_L_GPIO_Port GPIOE
- #define PWM5_L_Pin GPIO_PIN_12
- #define PWM5_L_GPIO_Port GPIOE
- #define PWM6_L_Pin GPIO_PIN_13
- #define PWM6_L_GPIO_Port GPIOE
- #define WRN_L_Pin GPIO_PIN_14
- #define WRN_L_GPIO_Port GPIOE
- #define WRN_R_Pin GPIO_PIN_15
- #define WRN_R_GPIO_Port GPIOE
- #define B_R_Pin GPIO_PIN_10
- #define B_R_GPIO_Port GPIOB
- #define Z_R_Pin GPIO_PIN_11
- #define Z_R_GPIO_Port GPIOB
- #define PWM3_R_Pin GPIO_PIN_14
- #define PWM3_R_GPIO_Port GPIOB
- #define PWM5_R_Pin GPIO_PIN_15
- #define PWM5_R_GPIO_Port GPIOB
- #define A_L_Pin GPIO_PIN_12
- #define A_L_GPIO_Port GPIOD
- #define B_L_Pin GPIO_PIN_14
- #define B_L_GPIO_Port GPIOD
- #define Z_L_Pin GPIO_PIN_15
- #define Z_L_GPIO_Port GPIOD
- #define PWM2_R_Pin GPIO_PIN_6
- #define PWM2_R_GPIO_Port GPIOC
- #define PWM4_R_Pin GPIO_PIN_7
- #define PWM4_R_GPIO_Port GPIOC
- #define PWM6_R_Pin GPIO_PIN_8
- #define PWM6_R_GPIO_Port GPIOC
- #define TRIP_R_Pin GPIO_PIN_9
- #define TRIP_R_GPIO_Port GPIOC

- #define TRIP_L_Pin GPIO_PIN_8
- #define TRIP_L_GPIO_Port GPIOA
- #define A_R_Pin GPIO_PIN_15
- #define A_R_GPIO_Port GPIOA
- #define DIR_Pin GPIO_PIN_3
- #define DIR_GPIO_Port GPIOD
- #define LED_LEFT_Pin GPIO_PIN_4
- #define LED_LEFT_GPIO_Port GPIOD
- #define LED_RIGHT_Pin GPIO_PIN_5
- #define LED_RIGHT_GPIO_Port GPIOD
- #define LED_ERR_Pin GPIO_PIN_6
- #define LED_ERR_GPIO_Port GPIOD

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.11.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

4.11.2 Macro Definition Documentation

4.11.2.1 A_L_GPIO_Port

```
#define A_L_GPIO_Port GPIOD
```

4.11.2.2 A_L_Pin

```
#define A_L_Pin GPIO_PIN_12
```

4.11.2.3 A_R_GPIO_Port

```
#define A_R_GPIO_Port GPIOA
```

4.11.2.4 A_R_Pin

```
#define A_R_Pin GPIO_PIN_15
```

4.11.2.5 B_L_GPIO_Port

```
#define B_L_GPIO_Port GPIOD
```

4.11.2.6 B_L_Pin

```
#define B_L_Pin GPIO_PIN_14
```

4.11.2.7 B_R_GPIO_Port

```
#define B_R_GPIO_Port GPIOB
```

4.11.2.8 B_R_Pin

```
#define B_R_Pin GPIO_PIN_10
```

4.11.2.9 DAC_GPIO_Port

```
#define DAC_GPIO_Port GPIOA
```

4.11.2.10 DAC_Pin

```
#define DAC_Pin GPIO_PIN_4
```

4.11.2.11 DIR_GPIO_Port

```
#define DIR_GPIO_Port GPIOD
```

4.11.2.12 DIR_Pin

```
#define DIR_Pin GPIO_PIN_3
```

4.11.2.13 ENABLE_L_GPIO_Port

```
#define ENABLE_L_GPIO_Port GPIOE
```

4.11.2.14 ENABLE_L_Pin

```
#define ENABLE_L_Pin GPIO_PIN_7
```

4.11.2.15 ENABLE_R_GPIO_Port

```
#define ENABLE_R_GPIO_Port GPIOB
```

4.11.2.16 ENABLE_R_Pin

```
#define ENABLE_R_Pin GPIO_PIN_2
```

4.11.2.17 ia_L_GPIO_Port

```
#define ia_L_GPIO_Port GPIOA
```

4.11.2.18 ia_L_Pin

```
#define ia_L_Pin GPIO_PIN_0
```

4.11.2.19 ia_R_GPIO_Port

```
#define ia_R_GPIO_Port GPIOA
```

4.11.2.20 ia_R_Pin

```
#define ia_R_Pin GPIO_PIN_6
```

4.11.2.21 ib_L_GPIO_Port

```
#define ib_L_GPIO_Port GPIOA
```

4.11.2.22 ib_L_Pin

```
#define ib_L_Pin GPIO_PIN_1
```

4.11.2.23 ib_R_GPIO_Port

```
#define ib_R_GPIO_Port GPIOA
```

4.11.2.24 ib_R_Pin

```
#define ib_R_Pin GPIO_PIN_7
```

4.11.2.25 ic_L_GPIO_Port

```
#define ic_L_GPIO_Port GPIOA
```

4.11.2.26 ic_L_Pin

```
#define ic_L_Pin GPIO_PIN_2
```

4.11.2.27 ic_R_GPIO_Port

```
#define ic_R_GPIO_Port GPIOB
```

4.11.2.28 ic_R_Pin

```
#define ic_R_Pin GPIO_PIN_0
```

4.11.2.29 LED_ERR_GPIO_Port

```
#define LED_ERR_GPIO_Port GPIOD
```

4.11.2.30 LED_ERR_Pin

```
#define LED_ERR_Pin GPIO_PIN_6
```

4.11.2.31 LED_LEFT_GPIO_Port

```
#define LED_LEFT_GPIO_Port GPIOD
```

4.11.2.32 LED_LEFT_Pin

```
#define LED_LEFT_Pin GPIO_PIN_4
```

4.11.2.33 LED_RIGHT_GPIO_Port

```
#define LED_RIGHT_GPIO_Port GPIOD
```

4.11.2.34 LED_RIGHT_Pin

```
#define LED_RIGHT_Pin GPIO_PIN_5
```

4.11.2.35 PWM1_L_GPIO_Port

```
#define PWM1_L_GPIO_Port GPIOE
```

4.11.2.36 PWM1_L_Pin

```
#define PWM1_L_Pin GPIO_PIN_8
```

4.11.2.37 PWM1_R_GPIO_Port

```
#define PWM1_R_GPIO_Port GPIOA
```

4.11.2.38 PWM1_R_Pin

```
#define PWM1_R_Pin GPIO_PIN_5
```

4.11.2.39 PWM2_L_GPIO_Port

```
#define PWM2_L_GPIO_Port GPIOE
```

4.11.2.40 PWM2_L_Pin

```
#define PWM2_L_Pin GPIO_PIN_9
```

4.11.2.41 PWM2_R_GPIO_Port

```
#define PWM2_R_GPIO_Port GPIOC
```

4.11.2.42 PWM2_R_Pin

```
#define PWM2_R_Pin GPIO_PIN_6
```

4.11.2.43 PWM3_L_GPIO_Port

```
#define PWM3_L_GPIO_Port GPIOE
```

4.11.2.44 PWM3_L_Pin

```
#define PWM3_L_Pin GPIO_PIN_10
```

4.11.2.45 PWM3_R_GPIO_Port

```
#define PWM3_R_GPIO_Port GPIOB
```

4.11.2.46 PWM3_R_Pin

```
#define PWM3_R_Pin GPIO_PIN_14
```

4.11.2.47 PWM4_L_GPIO_Port

```
#define PWM4_L_GPIO_Port GPIOE
```

4.11.2.48 PWM4_L_Pin

```
#define PWM4_L_Pin GPIO_PIN_11
```

4.11.2.49 PWM4_R_GPIO_Port

```
#define PWM4_R_GPIO_Port GPIOC
```

4.11.2.50 PWM4_R_Pin

```
#define PWM4_R_Pin GPIO_PIN_7
```

4.11.2.51 PWM5_L_GPIO_Port

```
#define PWM5_L_GPIO_Port GPIOE
```

4.11.2.52 PWM5_L_Pin

```
#define PWM5_L_Pin GPIO_PIN_12
```

4.11.2.53 PWM5_R_GPIO_Port

```
#define PWM5_R_GPIO_Port GPIOB
```

4.11.2.54 PWM5_R_Pin

```
#define PWM5_R_Pin GPIO_PIN_15
```

4.11.2.55 PWM6_L_GPIO_Port

```
#define PWM6_L_GPIO_Port GPIOE
```

4.11.2.56 PWM6_L_Pin

```
#define PWM6_L_Pin GPIO_PIN_13
```

4.11.2.57 PWM6_R_GPIO_Port

```
#define PWM6_R_GPIO_Port GPIOC
```

4.11.2.58 PWM6_R_Pin

```
#define PWM6_R_Pin GPIO_PIN_8
```

4.11.2.59 SC_det_GPIO_Port

```
#define SC_det_GPIO_Port GPIOC
```

4.11.2.60 SC_det_Pin

```
#define SC_det_Pin GPIO_PIN_4
```

4.11.2.61 Tinv_L_GPIO_Port

```
#define Tinv_L_GPIO_Port GPIOC
```

4.11.2.62 Tinv_L_Pin

```
#define Tinv_L_Pin GPIO_PIN_0
```

4.11.2.63 Tinv_R_GPIO_Port

```
#define Tinv_R_GPIO_Port GPIOC
```

4.11.2.64 Tinv_R_Pin

```
#define Tinv_R_Pin GPIO_PIN_1
```

4.11.2.65 Tmot_L_GPIO_Port

```
#define Tmot_L_GPIO_Port GPIOC
```

4.11.2.66 Tmot_L_Pin

```
#define Tmot_L_Pin GPIO_PIN_2
```

4.11.2.67 Tmot_R_GPIO_Port

```
#define Tmot_R_GPIO_Port GPIOC
```

4.11.2.68 Tmot_R_Pin

```
#define Tmot_R_Pin GPIO_PIN_3
```

4.11.2.69 TRIP_L_GPIO_Port

```
#define TRIP_L_GPIO_Port GPIOA
```

4.11.2.70 TRIP_L_Pin

```
#define TRIP_L_Pin GPIO_PIN_8
```

4.11.2.71 TRIP_R_GPIO_Port

```
#define TRIP_R_GPIO_Port GPIOC
```

4.11.2.72 TRIP_R_Pin

```
#define TRIP_R_Pin GPIO_PIN_9
```

4.11.2.73 VDC_L_GPIO_Port

```
#define VDC_L_GPIO_Port GPIOA
```

4.11.2.74 VDC_L_Pin

```
#define VDC_L_Pin GPIO_PIN_3
```

4.11.2.75 VDC_R_GPIO_Port

```
#define VDC_R_GPIO_Port GPIOB
```

4.11.2.76 VDC_R_Pin

```
#define VDC_R_Pin GPIO_PIN_1
```

4.11.2.77 WRN_L_GPIO_Port

```
#define WRN_L_GPIO_Port GPIOE
```

4.11.2.78 WRN_L_Pin

```
#define WRN_L_Pin GPIO_PIN_14
```

4.11.2.79 WRN_R_GPIO_Port

```
#define WRN_R_GPIO_Port GPIOE
```

4.11.2.80 WRN_R_Pin

```
#define WRN_R_Pin GPIO_PIN_15
```

4.11.2.81 Z_L_GPIO_Port

```
#define Z_L_GPIO_Port GPIOD
```

4.11.2.82 Z_L_Pin

```
#define Z_L_Pin GPIO_PIN_15
```

4.11.2.83 Z_R_GPIO_Port

```
#define Z_R_GPIO_Port GPIOB
```

4.11.2.84 Z_R_Pin

```
#define Z_R_Pin GPIO_PIN_11
```

4.11.3 Function Documentation

4.11.3.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

Here is the caller graph for this function:

4.12 main.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f7xx_hal.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define Tinv_L_Pin GPIO_PIN_0
#define Tinv_L_GPIO_Port GPIOC
#define Tinv_R_Pin GPIO_PIN_1
#define Tinv_R_GPIO_Port GPIOC
#define Tmot_L_Pin GPIO_PIN_2
#define Tmot_L_GPIO_Port GPIOC
#define Tmot_R_Pin GPIO_PIN_3
#define Tmot_R_GPIO_Port GPIOC
#define ia_L_Pin GPIO_PIN_0
#define ia_L_GPIO_Port GPIOA
#define ib_L_Pin GPIO_PIN_1
```

```
#define ib_L_GPIO_Port GPIOA
#define ic_L_Pin GPIO_PIN_2
#define ic_L_GPIO_Port GPIOA
#define VDC_L_Pin GPIO_PIN_3
#define VDC_L_GPIO_Port GPIOA
#define DAC_Pin GPIO_PIN_4
#define DAC_GPIO_Port GPIOA
#define PWM1_R_Pin GPIO_PIN_5
#define PWM1_R_GPIO_Port GPIOA
#define ia_R_Pin GPIO_PIN_6
#define ia_R_GPIO_Port GPIOA
#define ib_R_Pin GPIO_PIN_7
#define ib_R_GPIO_Port GPIOA
#define SC_det_Pin GPIO_PIN_4
#define SC_det_GPIO_Port GPIOC
#define ic_R_Pin GPIO_PIN_0
#define ic_R_GPIO_Port GPIOB
#define VDC_R_Pin GPIO_PIN_1
#define VDC_R_GPIO_Port GPIOB
#define ENABLE_R_Pin GPIO_PIN_2
#define ENABLE_R_GPIO_Port GPIOB
#define ENABLE_L_Pin GPIO_PIN_7
#define ENABLE_L_GPIO_Port GPIOE
#define PWM1_L_Pin GPIO_PIN_8
#define PWM1_L_GPIO_Port GPIOE
#define PWM2_L_Pin GPIO_PIN_9
#define PWM2_L_GPIO_Port GPIOE
#define PWM3_L_Pin GPIO_PIN_10
#define PWM3_L_GPIO_Port GPIOE
#define PWM4_L_Pin GPIO_PIN_11
#define PWM4_L_GPIO_Port GPIOE
#define PWM5_L_Pin GPIO_PIN_12
#define PWM5_L_GPIO_Port GPIOE
#define PWM6_L_Pin GPIO_PIN_13
#define PWM6_L_GPIO_Port GPIOE
#define WRN_L_Pin GPIO_PIN_14
#define WRN_L_GPIO_Port GPIOE
#define WRN_R_Pin GPIO_PIN_15
#define WRN_R_GPIO_Port GPIOE
#define B_R_Pin GPIO_PIN_10
#define B_R_GPIO_Port GPIOB
#define Z_R_Pin GPIO_PIN_11
#define Z_R_GPIO_Port GPIOB
#define PWM3_R_Pin GPIO_PIN_14
#define PWM3_R_GPIO_Port GPIOB
#define PWM5_R_Pin GPIO_PIN_15
#define PWM5_R_GPIO_Port GPIOB
#define A_L_Pin GPIO_PIN_12
#define A_L_GPIO_Port GPIOD
#define B_L_Pin GPIO_PIN_14
#define B_L_GPIO_Port GPIOD
#define Z_L_Pin GPIO_PIN_15
#define Z_L_GPIO_Port GPIOD
#define PWM2_R_Pin GPIO_PIN_6
#define PWM2_R_GPIO_Port GPIOC
#define PWM4_R_Pin GPIO_PIN_7
#define PWM4_R_GPIO_Port GPIOC
#define PWM6_R_Pin GPIO_PIN_8
#define PWM6_R_GPIO_Port GPIOC
#define TRIP_R_Pin GPIO_PIN_9
#define TRIP_R_GPIO_Port GPIOC
#define TRIP_L_Pin GPIO_PIN_8
#define TRIP_L_GPIO_Port GPIOA
#define A_R_Pin GPIO_PIN_15
#define A_R_GPIO_Port GPIOA
#define DIR_Pin GPIO_PIN_3
#define DIR_GPIO_Port GPIOD
#define LED_LEFT_Pin GPIO_PIN_4
#define LED_LEFT_GPIO_Port GPIOD
#define LED_RIGHT_Pin GPIO_PIN_5
#define LED_RIGHT_GPIO_Port GPIOD
#define LED_ERR_Pin GPIO_PIN_6
#define LED_ERR_GPIO_Port GPIOD

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

4.13 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/↩ MEASUREMENTS.h File Reference

Header file for handling measurements.

```
#include <stdint.h>
#include "ERRORS.h"
```

Include dependency graph for MEASUREMENTS.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct Encoder

  Structure for encoder reading.
- struct Analog

  Structure for ADC measurements in units.
- struct Feedback

  Structure for feedback values.

Macros

- #define CURRENT_SLOPE 117.57704f
- #define CURRENT_OFFSET 1.70068027211f
- #define VOLTAGE_SLOPE 263.435f
- #define VOLTAGE_OFFSET 0.02083f

Functions

- void get_currents_voltage (volatile uint16_t rawADC[ ], volatile Analog ∗analog, volatile Feedback ∗feedback, volatile InverterError ∗errors, float sinTheta_e, float cosTheta_e)

  Get electrical ADC measurements.
- float get_linear (uint32_t bits, float slope, float offset)

  Convert ADC reading to physical measurement with linear response.
- void get_idiq (float ia, float ib, float ic, float sinTheta_e, float cosTheta_e, float ∗idMeas, float ∗iqMeas)

  Computes d-q currents from current measurements and electrical angle.
- float get_temperature (uint32_t bits, const float tempLUT[ ])

  Retrieves temperature from a lookup table based on ADC bits.
- void calibrate_offsets (volatile uint16_t rawADC[ ], volatile float currentOffsets[ ], uint32_t numSamples)

  Calibrate the current sensor offsets.

Variables

- const float tempInverterLUT [ ]
- const float tempMotorLUT [ ]
- volatile uint16_t rawADC_left [4]

  Raw ADC data for the left inverter.
- volatile uint16_t rawADC_right [4]

  Raw ADC data for the right inverter.
- volatile uint16_t rawADC_temp [4]

  Raw ADC data for the temperatures.

4.13.1 Detailed Description

Header file for handling measurements.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.13.2 Macro Definition Documentation

4.13.2.1 CURRENT_OFFSET

```
#define CURRENT_OFFSET 1.70068027211f
```

[V] (10/(4.7+10))∗ 2.5 V (not actually used, self calibration at start)

4.13.2.2 CURRENT_SLOPE

```
#define CURRENT_SLOPE 117.57704f
```

[A/V] ((4.7+10)/10) ∗ (1 / (12.5 mV / A))

4.13.2.3 VOLTAGE_OFFSET

```
#define VOLTAGE_OFFSET 0.02083f
```

[V] (100/(4700+100) ∗ 5 V

4.13.2.4 VOLTAGE_SLOPE

```
#define VOLTAGE_SLOPE 263.435f
```

[V/V] 1/(1/3 ∗ 0.011388) V

4.13.3 Function Documentation

4.13.3.1 calibrate_offsets()

```
void calibrate_offsets (
            volatile uint16_t rawADC[],
            volatile float currentOffsets[],
            uint32_t numSamples )
```

Calibrate the current sensor offsets.

This function calculates the average offset for each current sensor channel by reading the ADC values when no current is flowing. The calculated offsets are used to correct the sensor readings.

Parameters

in	*rawADC*	Buffer containing the raw ADC values for the channels.
out	*currentOffsets*	Array to store the calculated offsets for each current channel.
in	*numSamples*	Number of samples to average for the offset calculation.

Here is the caller graph for this function:

4.13.3.2 get_currents_voltage()

```
void get_currents_voltage (
            volatile uint16_t rawADC[],
            volatile Analog * analog,
            volatile Feedback * feedback,
            volatile InverterError * errors,
            float sinTheta_e,
            float cosTheta_e )
```

Get electrical ADC measurements.

Parameters

in	*rawADC*	Pointer to the raw ADC values array.
out	*analog*	Pointer to the ADC struct to store the results.
out	*feedback*	Pointer to the Feedback struct to store id and iq.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)
in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)

Here is the call graph for this function:

Here is the caller graph for this function:

4.13.3.3 get_idiq()

```
void get_idiq (
            float ia,
            float ib,
            float ic,
            float sinTheta_e,
            float cosTheta_e,
            float * idMeas,
            float * iqMeas )
```

Computes d-q currents from current measurements and electrical angle.

This function computes the d-q currents from phase currents (ABC), theta_e, and stores the results in the provided pointers.

Parameters

in	*ia*	Phase A current in A.
in	*ib*	Phase B current in A.
in	*ic*	Phase C current in A.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)

Parameters

in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)
out	*idMeas*	Pointer to store the d-axis current.
out	*iqMeas*	Pointer to store the q-axis current.

This function computes the d-q currents from phase currents (ABC), theta_e, and stores the results in the provided pointers.

Parameters

in	*ia*	Phase A current in A.
in	*ib*	Phase B current in A.
in	*ic*	Phase C current in A.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)
in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)
out	*idMeas*	Pointer to store the D-axis current.
out	*iqMeas*	Pointer to store the Q-axis current.

Here is the caller graph for this function:

4.13.3.4 get_linear()

```
float get_linear (
            uint32_t bits,
            float slope,
            float offset )
```

Convert ADC reading to physical measurement with linear response.

Parameters

in	*bits*	The ADC reading.
in	*slope*	The slope (volts per unit).
in	*offset*	The offset (volts at zero).

Return values

measurement	The physical measurement.

Parameters

in	*bits*	The ADC reading.
in	*slope*	The slope (units per volt).
in	*offset*	The offset (volts at zero).

Return values

measurement	The physical measurement.

Here is the caller graph for this function:

4.13.3.5 get_temperature()

```
float get_temperature (
            uint32_t bits,
            const float tempLUT[] )
```

Retrieves temperature from a lookup table based on ADC bits.

This function retrieves temperature from a lookup table based on the ADC bits. The lookup table (LUT) must have a value for each possible ADC bit combination.

Parameters

in	*bits*	ADC reading converted to bits.
in	*tempLUT*	Lookup table containing temperature values.

Returns

Temperature corresponding to the provided ADC bits.

Here is the caller graph for this function:

4.13.4 Variable Documentation

4.13.4.1 rawADC_left

```
volatile uint16_t rawADC_left[4]  [extern]
```

Raw ADC data for the left inverter.

External declaration of raw ADC data for the left inverter

External declaration of raw ADC data for the left inverter.

4.13.4.2 rawADC_right

```
volatile uint16_t rawADC_right[4]  [extern]
```

Raw ADC data for the right inverter.

External declaration of raw ADC data for the right inverter

External declaration of raw ADC data for the right inverter.

4.13.4.3 rawADC_temp

```
volatile uint16_t rawADC_temp[4]  [extern]
```

Raw ADC data for the temperatures.

External declaration of raw ADC data for the temperatures

External declaration of raw ADC data for the temperature readings.

4.13.4.4 tempInverterLUT

```
const float tempInverterLUT[]  [extern]
```

4.13.4.5 tempMotorLUT

```
const float tempMotorLUT[]  [extern]
```

4.14 MEASUREMENTS.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */


#ifndef MEASUREMENTS_H
#define MEASUREMENTS_H

#include <stdint.h>
#include "ERRORS.h"

/* Define current and voltage gains/offsets */
#define CURRENT_SLOPE  117.57704f
#define CURRENT_OFFSET 1.70068027211f
#define VOLTAGE_SLOPE  263.435f
#define VOLTAGE_OFFSET 0.02083f
extern const float tempInverterLUT[];
extern const float tempMotorLUT[];

extern volatile uint16_t rawADC_left[4];
extern volatile uint16_t rawADC_right[4];
extern volatile uint16_t rawADC_temp[4];
typedef struct {
    uint16_t A;
    uint16_t B;
    uint16_t Z;
    float we;
    float theta_e;
    float sinTheta_e;
      float cosTheta_e;
    uint8_t directionMeas;
} Encoder;


typedef struct {
    float ia;
    float ib;
    float ic;
    float vDC;
    float currentOffsets[3];
} Analog;


typedef struct {
    float idMeas;
    float iqMeas;
    float torqueCalc;
    float speedMeas;
} Feedback;



void get_currents_voltage(volatile uint16_t rawADC[], volatile Analog* analog, volatile Feedback*
feedback, volatile InverterError *errors, float sinTheta_e, float cosTheta_e);

float get_linear(uint32_t bits, float slope, float offset);


void get_idiq(float ia, float ib, float ic, float sinTheta_e, float cosTheta_e, float *idMeas, float
*iqMeas);


float get_temperature(uint32_t bits, const float tempLUT[]);


void calibrate_offsets(volatile uint16_t rawADC[], volatile float currentOffsets[], uint32_t
numSamples);


#endif /* MEASUREMENTS_H */
```

4.15 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h File Reference

Header file for motor parameters.

```
#include <stdint.h>
```

Include dependency graph for MOTOR.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct MotorConstants

  Structure to hold precomputed motor constants.
- struct MotorParameters

  Structure to hold motor parameters.

Functions

- void precalculate_motor_constants (MotorParameters ∗motor)

  Precomputes the constants for a motor and updates the MotorParameters structure.
- int check_motor_parameters (MotorParameters ∗motor, float Ts)

  Perform a parameter check and correct possible errors.

Variables

- MotorParameters motor_left

  Left motor parameters.
- MotorParameters motor_right

  Right motor parameters.

4.15.1 Detailed Description

Header file for motor parameters.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.15.2 Function Documentation

4.15.2.1 check_motor_parameters()

```
int check_motor_parameters (
            MotorParameters * motor,
            float Ts )
```

Perform a parameter check and correct possible errors.

Parameters

in	*motor*	Pointer to the MotorParameters struct.

Return values

OK	0 if an error occurred, 1 if successful.

Here is the caller graph for this function:

4.15.2.2 precalculate_motor_constants()

```
void precalculate_motor_constants (
            MotorParameters * motor )
```

Precomputes the constants for a motor and updates the MotorParameters structure.

Parameters

motor	[in, out] Pointer to the motor parameters structure

Here is the caller graph for this function:

4.15.3 Variable Documentation

4.15.3.1 motor_left

```
MotorParameters motor_left  [extern]
```

Left motor parameters.

4.15.3.2 motor_right

```
MotorParameters motor_right  [extern]
```

Right motor parameters.

4.16 MOTOR.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

#ifndef MOTOR_H
#define MOTOR_H

#include <stdint.h>

typedef struct {
    float threePpLambda;
    float threePpLdMinusLq;
    float invThreePpLambda;
    float isc;
    float torqueBase;
    float invTorqueBase;
    float xi;
    float xiSquared;
    float oneMinusXi;
    float twoMinusXi;
    float fourTimesOneMinusXi;
    float eightTimesOneMinusXiSquared;
    float twoMinusXiSquared;
    float twoTimesOneMinusXiOnePlusXiSquared;
    float twoTimesOneMinusXiXiSquared;
    float fourTimesOneMinusXiOnePlusXiSquared;
```

```
    float fourTimesOneMinusXiXiSquared;
    float lambdaDivLqMinusLd;
    float betaMinusIsc;
} MotorConstants;

typedef struct {
    float Ld;
    float Lq;
    float Rs;
    float lambda;
    uint8_t pp;
    float J;
    float b;
    float torqueMax;
    float dTorqueMax;
    float speedMax_RPM;
    float iMax;
    float vDCMax;
    MotorConstants constants;
} MotorParameters;

extern MotorParameters motor_left;
extern MotorParameters motor_right;

void precalculate_motor_constants(MotorParameters* motor);

int check_motor_parameters(MotorParameters *motor, float Ts);
#endif /* MOTOR_H */
```

4.17 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h File Reference

Header file for handling GPIOs.

```
#include "stm32f7xx_hal.h"
#include "main.h"
#include <stdbool.h>
```

Include dependency graph for PCB_IO.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct LED

  LED structure.

Macros

- #define SC_DET_STATE() (HAL_GPIO_ReadPin(SC_det_GPIO_Port, SC_det_Pin))
- #define DIR_STATE() (HAL_GPIO_ReadPin(DIR_GPIO_Port, DIR_Pin))
- #define WRN_STATE(port, pin) (HAL_GPIO_ReadPin(port, pin))
- #define ENABLE(port, pin) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET); } while(0)
- #define DISABLE(port, pin) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET); } while(0)

Enumerations

- enum LEDMode { LED_MODE_BLINK_FAST , LED_MODE_BLINK_SLOW , LED_MODE_ON , LED_MODE_OFF }

Functions

- void handle_LED (LED ∗led, uint32_t ms_counter)

  LED handler function.
- void handle_direction (volatile int8_t ∗dir_left, volatile int8_t ∗dir_right)

  Handles the direction of the motors.
- void enable_inverters (volatile bool enableSW_left, volatile bool enableSW_right, volatile bool ∗enable_left, volatile bool ∗enable_right)

  Handles the direction of the motors and enables/disables the inverters.

Variables

- LED led_left
- LED led_right
- LED ledError

4.17.1 Detailed Description

Header file for handling GPIOs.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.17.2 Macro Definition Documentation

4.17.2.1 DIR_STATE

```
#define DIR_STATE( ) (HAL_GPIO_ReadPin(DIR_GPIO_Port, DIR_Pin))
```

4.17.2.2 DISABLE

```
#define DISABLE(
            port,
            pin ) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET); } while(0)
```

4.17.2.3 ENABLE

```
#define ENABLE(
            port,
            pin ) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET); } while(0)
```

4.17.2.4 SC_DET_STATE

```
#define SC_DET_STATE( ) (HAL_GPIO_ReadPin(SC_det_GPIO_Port, SC_det_Pin))
```

4.17.2.5 WRN_STATE

```
#define WRN_STATE(
            port,
            pin ) (HAL_GPIO_ReadPin(port, pin))
```

4.17.3 Enumeration Type Documentation

4.17.3.1 LEDMode

```
enum LEDMode
```

Enumerator

LED_MODE_BLINK_FAST	Fast blink mode
LED_MODE_BLINK_SLOW	Slow blink mode
LED_MODE_ON	LED on mode
LED_MODE_OFF	LED off mode

4.17.4 Function Documentation

4.17.4.1 enable_inverters()

```
void enable_inverters (
            volatile bool enableSW_left,
            volatile bool enableSW_right,
            volatile bool * enable_left,
            volatile bool * enable_right )
```

Handles the direction of the motors and enables/disables the inverters.

This function reads the state of the shutdown chain (SC or SDC) and enables/disables the inverters based on that and an external software enable bool.

Parameters

in	*enableSW_left*	The software enable state for the left inverter.
in	*enableSW_right*	The software enable state for the right inverter.
out	*enable_left*	Output parameter for the left inverter's enable state.
out	*enable_right*	Output parameter for the right inverter's enable state.

Here is the caller graph for this function:

4.17.4.2 handle_direction()

```
void handle_direction (
            volatile int8_t * dir_left,
            volatile int8_t * dir_right )
```

Handles the direction of the motors.

This function reads the state of the DIR switch and updates the directions of both the left and right motors. If one motor is set to rotate clockwise (CW), the other one is set to rotate counterclockwise (CCW), and vice versa.

Parameters

dir_left	Pointer to the direction parameter in the left inverter structure.
dir_right	Pointer to the direction parameter in the right inverter structure.

Here is the caller graph for this function:

4.17.4.3 handle_LED()

```
void handle_LED (
            LED * led,
            uint32_t ms_counter )
```

LED handler function.

This function handles the LED blinking modes based on the LED mode and current millisecond counter.

Parameters

led	Pointer to the LED structure.
ms_counter	Millisecond counter for timing.

This function handles the LED blinking modes based on the LED mode and current millisecond counter.

Parameters

led	Pointer to the LED structure.
ms_counter	Current millisecond counter.

Here is the caller graph for this function:

4.17.5 Variable Documentation

4.17.5.1 led_left

```
LED led_left [extern]
```

4.17.5.2 led_right

```
LED led_right [extern]
```

4.17.5.3 ledError

```
LED ledError [extern]
```

4.18 PCB_IO.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */


#ifndef PCB_IO_H
#define PCB_IO_H

#include "stm32f7xx_hal.h"
#include "main.h" // pin names/ports
#include <stdbool.h>

// Read SC_det and DIR GPIOs
#define SC_DET_STATE()              (HAL_GPIO_ReadPin(SC_det_GPIO_Port, SC_det_Pin))
#define DIR_STATE()                 (HAL_GPIO_ReadPin(DIR_GPIO_Port, DIR_Pin))

// Read WRN GPIOs
#define WRN_STATE(port, pin)        (HAL_GPIO_ReadPin(port, pin))

// Control ENABLE GPIOs
#define ENABLE(port, pin)           do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET); } while(0)
#define DISABLE(port, pin)          do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET); } while(0)

// Define LED modes
typedef enum {
    LED_MODE_BLINK_FAST,
    LED_MODE_BLINK_SLOW,
    LED_MODE_ON,
    LED_MODE_OFF
} LEDMode;

typedef struct {
    GPIO_TypeDef *port;
    uint16_t pin;
    LEDMode mode;
} LED;

// Declare LED variables as extern
extern LED led_left;
extern LED led_right;
extern LED ledError;

// Function prototypes
void handle_LED(LED *led, uint32_t ms_counter);

void handle_direction(volatile int8_t *dir_left, volatile int8_t *dir_right);

void enable_inverters(volatile bool enableSW_left, volatile bool enableSW_right, volatile bool
    *enable_left, volatile bool *enable_right);


#endif /* PCB_IO_H */
```

4.19 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h File Reference

Header file for controlling PWM output.

`#include "stm32f7xx_hal.h"`
Include dependency graph for PWM.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct Duties

  Structure to hold PWM configuration parameters.

Functions

- void enable_PWM (TIM_HandleTypeDef ∗htim)

  Enable PWM output.
- void disable_PWM (TIM_HandleTypeDef ∗htim)

  Disable PWM output.
- void update_PWM (TIM_HandleTypeDef ∗htim, Duties duties)

  Set PWM duty cycles.

4.19.1 Detailed Description

Header file for controlling PWM output.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.19.2 Function Documentation

4.19.2.1 disable_PWM()

```
void disable_PWM (
            TIM_HandleTypeDef * htim )
```

Disable PWM output.

This function disables PWM output for the specified timer.

Parameters

htim	Pointer to the TIM_HandleTypeDef structure.

4.19.2.2 enable_PWM()

```
void enable_PWM (
            TIM_HandleTypeDef * htim )
```

Enable PWM output.

This function enables PWM output for the specified timer.

Parameters

htim	Pointer to the TIM_HandleTypeDef structure.

4.19.2.3 update_PWM()

```
void update_PWM (
            TIM_HandleTypeDef * htim,
            Duties duties )
```

Set PWM duty cycles.

This function sets the duty cycles for the PWM channels.

Parameters

htim	Pointer to the TIM_HandleTypeDef structure.
duties	Duties structure containing duty cycle values.

Here is the caller graph for this function:

4.20 PWM.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef PWM_H
00021 #define PWM_H
00022
00023 #include "stm32f7xx_hal.h"
00024
00028 typedef struct {
00029     float Da;
00030     float Db;
00031     float Dc;
00032 } Duties;
00033
00041 void enable_PWM(TIM_HandleTypeDef *htim);
00042
00050 void disable_PWM(TIM_HandleTypeDef *htim);
00051
00052
00061 void update_PWM(TIM_HandleTypeDef *htim, Duties duties);
00062
00063 #endif /* PWM_H */
```

4.21 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/↩ REFERENCE.h File Reference

Header file for torque reference handling.

```
#include "PergaMOD.h"
#include "MOTOR.h"
#include "MEASUREMENTS.h"
```

`#include <stdint.h>`

Include dependency graph for REFERENCE.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct Reference

  Structure for reference values.

Macros

- #define TEMP_MOTOR_DERATING (OVERTEMPERATURE_MOTOR_TH - 20.0F)
- #define TEMP_INVERTER_DERATING (OVERTEMPERATURE_INVERTER_TH - 20.0F)
- #define TEMP_MOTOR_MAX (OVERTEMPERATURE_MOTOR_TH + 10.0F)
- #define TEMP_INVERTER_MAX (OVERTEMPERATURE_INVERTER_TH + 10.0F)

Functions

- float handle_torqueRef (float torqueRefIn, int8_t direction, float torqueMax, float speedMaxRPM, float speedMeas, volatile pi_struct *loopSpeed)

  Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.
- float set_torque_direction (float torqueRef, int8_t direction)

  Set torque direction based on inverter direction.
- float saturate_symmetric (float ref, float max)

  Symmetrically saturate a reference value.
- float limit_torque_to_prevent_overspeed (float speedMax, float speedMeas, float torqueRefIn, volatile pi_struct *loopSpeed)

  Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.
- float calculate_derated_current (float temperature, float tempStart, float tempMax, float iMax)

  Calculate derated current based on temperature thresholds. It implements a simple linear derating from tempStart to tempMax.
- float derate_current_reference (float tempMotor, float tempInverter, float iMax)

  Derate the current reference based on both motor and inverter temperatures.

Variables

- float torqueRefIn_left
- float torqueRefIn_right

4.21.1 Detailed Description

Header file for torque reference handling.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

4.21.2 Macro Definition Documentation

4.21.2.1 TEMP_INVERTER_DERATING

```
#define TEMP_INVERTER_DERATING (OVERTEMPERATURE_INVERTER_TH - 20.0F)
```

Temperature at which linear derating starts for the inverter (20 degC before the fault)

4.21.2.2 TEMP_INVERTER_MAX

```
#define TEMP_INVERTER_MAX (OVERTEMPERATURE_INVERTER_TH + 10.0F)
```

Temperature at which derating is 0 for the inverter (10 degC more than the fault)

4.21.2.3 TEMP_MOTOR_DERATING

```
#define TEMP_MOTOR_DERATING (OVERTEMPERATURE_MOTOR_TH - 20.0F)
```

Temperature at which linear derating starts for the motor (20 degC before the fault)

4.21.2.4 TEMP_MOTOR_MAX

```
#define TEMP_MOTOR_MAX (OVERTEMPERATURE_MOTOR_TH + 10.0F)
```

Temperature at which derating is 0 for the motor (10 degC more than the fault)

4.21.3 Function Documentation

4.21.3.1 calculate_derated_current()

```
float calculate_derated_current (
            float temperature,
            float tempStart,
            float tempMax,
            float iMax )
```

Calculate derated current based on temperature thresholds. It implements a simple linear derating from tempStart to tempMax.

Parameters

in	*temperature*	The current temperature.
in	*tempStart*	The temperature at which derating starts.
in	*tempMax*	The temperature at which the current is fully derated to 0.
in	*iMax*	The maximum current.

Returns

The derated current.

Here is the caller graph for this function:

4.21.3.2 derate_current_reference()

```
float derate_current_reference (
            float tempMotor,
```

```
            float tempInverter,
            float iMax )
```

Derate the current reference based on both motor and inverter temperatures.

Parameters

in	*tempMotor*	The motor temperature.
in	*tempInverter*	The inverter temperature.
in	*iMax*	The maximum current.

Returns

The derated current reference.

Here is the call graph for this function:

Here is the caller graph for this function:

4.21.3.3 handle_torqueRef()

```
float handle_torqueRef (
            float torqueRefIn,
            int8_t direction,
            float torqueMax,
            float speedMaxRPM,
            float speedMeas,
            volatile pi_struct * loopSpeed )
```

Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.

Parameters

torqueRefIn	Input reference torque.
direction	Direction of torque (1 for positive torque, -1 for negative torque).
torqueMax	Maximum allowable torque.
speedMaxRPM	Maximum allowable speed in RPM.
speedMeas	Measured speed.
loopSpeed	Speed control loop parameters.

Returns

The output torque after handling direction, saturation, and rate limiting.

Here is the call graph for this function:

Here is the caller graph for this function:

4.21.3.4 limit_torque_to_prevent_overspeed()

```
float limit_torque_to_prevent_overspeed (
            float speedMaxRPM,
            float speedMeas,
            float torqueRefIn,
            volatile pi_struct * loopSpeed )
```

Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.

Parameters

in	*speedMax*	The maximum speed value in RPM.
in	*speedMeas*	The measured speed value in RPM.
in	*torque↩ RefIn*	The torque reference value before this saturation.
in	*loopSpeed*	Pointer to the speed PI controller structure.

Returns

torqueRef_out The limited torque reference value after this saturation.

Parameters

in	*speedMaxRPM*	The maximum speed value in RPM.
in	*speedMeas*	The measured speed value in RPM.
in	*torqueRefIn*	The torque reference value before this saturation.
in	*loopSpeed*	Pointer to the speed PI controller structure.

Returns

torqueRefOut The limited torque reference value after this saturation.

Here is the caller graph for this function:

4.21.3.5 saturate_symmetric()

```
float saturate_symmetric (
            float ref,
            float max )
```

Symmetrically saturate a reference value.

This function symmetrically saturates a reference value based on the maximum allowed value. If the reference value exceeds the maximum allowed value, it is saturated to the maximum value. If the reference value is less than the negative of the maximum allowed value, it is saturated to the negative of the maximum value.

Parameters

in	*ref*	The reference value to saturate.
in	*max*	The maximum allowed value for saturation.

Returns

The saturated reference value.

Here is the caller graph for this function:

4.21.3.6 set_torque_direction()

```
float set_torque_direction (
            float torqueRefIn,
            int8_t direction )
```

Set torque direction based on inverter direction.

This function adjusts the torque reference based on the direction of the inverter. If the inverter is set to rotate counterclockwise (CCW), positive torque represents braking. If the inverter is set to rotate clockwise (CW), positive torque represents traction.

Parameters

in	*torqueRef*	The torque reference value to adjust.
in	*direction*	Pointer to the direction of the inverter (1 for CW, -1 for CCW).

Returns

The adjusted torque reference value.

This function adjusts the torque reference based on the desired direction. If the motor is set to rotate counterclockwise (CCW), positive torque represents traction, negative is braking. If the motor is set to rotate clockwise (CW), negative torque represents traction, positive is braking.

Parameters

in	*torque↩ RefIn*	The torque reference value to adjust.
in	*direction*	Pointer to the direction of the inverter (1 for CW, -1 for CCW).

Returns

torqueRefOut The adjusted torque reference value.

Here is the caller graph for this function:

4.21.4 Variable Documentation

4.21.4.1 torqueRefIn_left

```
float torqueRefIn_left  [extern]
```

4.21.4.2 torqueRefIn_right

```
float torqueRefIn_right  [extern]
```

4.22 REFERENCE.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef REFERENCE_H
00021 #define REFERENCE_H
00022
00023 #include "PergaMOD.h" // ramp, pi struct
00024 #include "MOTOR.h" // motor struct
00025 #include "MEASUREMENTS.h" // overtemperature defines
00026 #include <stdint.h>
00027
00028 // Define temperature derating tresholds
00029 #define TEMP_MOTOR_DERATING (OVERTEMPERATURE_MOTOR_TH - 20.0F)
00030 #define TEMP_INVERTER_DERATING (OVERTEMPERATURE_INVERTER_TH - 20.0F)
00032 #define TEMP_MOTOR_MAX (OVERTEMPERATURE_MOTOR_TH + 10.0F)
00033 #define TEMP_INVERTER_MAX (OVERTEMPERATURE_INVERTER_TH + 10.0F)
00036 // These variables should be updated via CAN
00037 extern float torqueRefIn_left;
00038 extern float torqueRefIn_right;
00039
00043 typedef struct {
00044     float idRef;
00045     float iqRef;
00046     float isMaxRef;
00047     float torqueRef;
00048 } Reference;
00049
00050
00064 float handle_torqueRef(float torqueRefIn, int8_t direction, float torqueMax, float speedMaxRPM, float
      speedMeas, volatile pi_struct *loopSpeed);
00065
00066
00078 float set_torque_direction(float torqueRef, int8_t direction);
00079
00091 float saturate_symmetric(float ref, float max);
00092
00101 float limit_torque_to_prevent_overspeed(float speedMax, float speedMeas, float torqueRefIn, volatile
      pi_struct *loopSpeed);
00102
00103
00115 float calculate_derated_current(float temperature, float tempStart, float tempMax, float iMax);
00116
00126 float derate_current_reference(float tempMotor, float tempInverter, float iMax);
00127 #endif /* REFERENCE_H */
```

4.23 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h File Reference

Header file for functions related to tasks executed every 1ms.

```
#include "INVERTER.h"
```

Include dependency graph for TASKS_1ms.h:

This graph shows which files directly or indirectly include this file:

Functions

- void tasks_1ms (void)

Function to be executed every 1ms.

- void read_temperatures (void)

  Function to read temperatures and handle overtemperature faults.

- void handle_overtemperature_faults (volatile InverterStruct *inv)

  Function to handle overtemperature faults.

4.23.1 Detailed Description

Header file for functions related to tasks executed every 1ms.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.23.2 Function Documentation

4.23.2.1 handle_overtemperature_faults()

```
void handle_overtemperature_faults (
            volatile InverterStruct * inv )
```

Function to handle overtemperature faults.

Parameters

in,out	*inv*	Pointer to the InverterStruct structure.

Here is the call graph for this function:

Here is the caller graph for this function:

4.23.2.2 read_temperatures()

```
void read_temperatures (
            void  )
```

Function to read temperatures and handle overtemperature faults.

Here is the call graph for this function:

Here is the caller graph for this function:

4.23.2.3 tasks_1ms()

```
void tasks_1ms (
            void  )
```

Function to be executed every 1ms.

This function is called by the TIM6 IRQ handler every millisecond.

This function is called by the TIM6 IRQ handler every millisecond. It increments the millisecond counter and executes all the low priority tasks. Here is the call graph for this function:

Here is the caller graph for this function:

4.24 TASKS_1ms.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */


#ifndef TASKS_1MS_H
#define TASKS_1MS_H

#include "INVERTER.h" // needs invLeft/invRight


void tasks_1ms(void);

void read_temperatures(void);

void handle_overtemperature_faults(volatile InverterStruct *inv);

#endif /* TASKS_1MS_H */
```

4.25 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_CRITICAL.h File Reference

Header file for functions related to tasks executed in each PWM timer interruption.

This graph shows which files directly or indirectly include this file:

Functions

- void tasks_critical_left ()

  Function to be executed every TS.
- void tasks_critical_right ()

  Function to be executed every TS.

4.25.1 Detailed Description

Header file for functions related to tasks executed in each PWM timer interruption.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

4.25.2 Function Documentation

4.25.2.1 tasks_critical_left()

```
void tasks_critical_left (
            void  )
```

Function to be executed every TS.

This function is called by the TIM1 trigger out handler every TS.

This function is called by the TIM1 trigger handler every TS. Here is the call graph for this function:

Here is the caller graph for this function:

4.25.2.2 tasks_critical_right()

```
void tasks_critical_right (
            void  )
```

Function to be executed every TS.

This function is called by the TIM8 trigger out handler every TS.

This function is called by the TIM8 trigger handler every TS. Here is the caller graph for this function:

4.26 TASKS_CRITICAL.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00025 void tasks_critical_left();
00026
00032 void tasks_critical_right();
```

4.27 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c File Reference

This file contains functions to handle CAN communication with the car.

```
#include "can.h"
#include "INVERTER.h"
#include "CAN_e-Tech.h"
```

Include dependency graph for CAN_e-Tech.c:

Functions

- void handle_CAN (CAN_HandleTypeDef ∗hcan)

  Handle CAN messages.
- void send_CAN_message (CAN_HandleTypeDef ∗hcan, void ∗dbc_msg, const float ∗data)

  Send a CAN message using CAN1db.h information.

Variables

- uint8_t keepAlive

4.27.1 Detailed Description

This file contains functions to handle CAN communication with the car.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.27.2 Function Documentation

4.27.2.1 handle_CAN()

```
void handle_CAN (
            CAN_HandleTypeDef * hcan )
```

Handle CAN messages.

This function implements the logic to handle received CAN messages.

Parameters

hcan	Pointer to the CAN handle structure.

Here is the call graph for this function:

Here is the caller graph for this function:

4.27.2.2 send_CAN_message()

```
void send_CAN_message (
            CAN_HandleTypeDef * hcan,
            void * dbc_msg,
            const float * data )
```

Send a CAN message using CAN1db.h information.

This function prepares and sends a CAN message using information from CAN1db.h.

Parameters

hcan	Pointer to the CAN handle structure.
dbc_msg	Pointer to the structure containing CAN message information from CAN1db.h.
data	Pointer to the array of float data to be sent.

Here is the caller graph for this function:

4.27.3 Variable Documentation

4.27.3.1 keepAlive

```
uint8_t keepAlive
```

4.28 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c File Reference

This file provides code for the control loop.

```
#include "CONTROL.h"
#include <math.h>
#include <PergaMOD.h>
```

Include dependency graph for CONTROL.c:

Functions

- void calc_current_reference (MotorParameters *motor, volatile Reference *reference)

  Calculates the current references using a FOC algorithm. It computes the current vector for the MTPA trajectory and limits the current reference to isMaxRef (calculated by derating, starting from the motor's maximum current). The MTPV trajectory is not implemented to save some computation time due to the nature of the motors expected. In order to implement field weakening, an external voltage loop modifying gammaRef is needed and should be called inside here. When implementing field weakening, special attention must be put to the torque reference being near 0 or differing from the speed sign (regeneration). A minimum id current must be set for speeds higher than Vs/lambda. Study thoroughly, simulate first.
- void calc_current_loop (volatile InverterStruct *inv)

  Calculates the id-iq loops.
- void saturate_voltage (volatile InverterStruct *inv)

  Saturates PI output to not surpass DC voltage.
- void calc_duties (float vd, float vq, float vDC, float sinTheta_e, float cosTheta_e, volatile Duties *duties)

  function.

4.28.1 Detailed Description

This file provides code for the control loop.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.28.2 Function Documentation

4.28.2.1 calc_current_loop()

```
void calc_current_loop (
            volatile InverterStruct * inv )
```

Calculates the id-iq loops.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.28.2.2 calc_current_reference()

```
void calc_current_reference (
            MotorParameters * motor,
            volatile Reference * reference )
```

Calculates the current references using a FOC algorithm. It computes the current vector for the MTPA trajectory and limits the current reference to isMaxRef (calculated by derating, starting from the motor's maximum current). The MTPV trajectory is not implemented to save some computation time due to the nature of the motors expected. In order to implement field weakening, an external voltage loop modifying gammaRef is needed and should be called inside here. When implementing field weakening, special attention must be put to the torque reference being near 0 or differing from the speed sign (regeneration). A minimum id current must be set for speeds higher than Vs/lambda. Study thoroughly, simulate first.

Parameters

in	*motor*	Pointer to the motor parameters structure.
in,out	*reference*	Pointer to the reference struct.

Here is the caller graph for this function:

4.28.2.3 calc_duties()

```
void calc_duties (
            float vd,
            float vq,
            float vDC,
            float sinTheta_e,
            float cosTheta_e,
            volatile Duties * duties )
```

function.

This function calculates the inverse Park transform and the duty cycles using SVPWM

Parameters

in	*vd*	Voltage in the d-axis.
in	*vq*	Voltage in the q-axis.
in	*vDC*	DC voltage.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)
in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)
out	*duties*	Pointer to the duties structure.

Here is the caller graph for this function:

4.28.2.4 saturate_voltage()

```
void saturate_voltage (
            volatile InverterStruct * inv )
```

Saturates PI output to not surpass DC voltage.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.29 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ERRORS.c File Reference

Header file for the necessary components to set, read and clear ERRORS.

`#include "ERRORS.h"`
Include dependency graph for ERRORS.c:

Functions

- void set_error (volatile void *data, InverterError error)

  Sets an error in the error field of a data structure.
- void clear_error (volatile void *data, InverterError error)

  Clears an error in the error field of a data structure.
- bool is_error_set (volatile void *data, InverterError error)

  Checks if an error is set in the error field of a data structure.

4.29.1 Detailed Description

Header file for the necessary components to set, read and clear ERRORS.

This file contains the necessary components to set, read and clear ERRORS.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.29.2 Function Documentation

4.29.2.1 clear_error()

```
void clear_error (
            volatile void * data,
            InverterError error )
```

Clears an error in the error field of a data structure.

This function clears the specified error bit in the error field of a data structure.

Parameters

out	*data*	Pointer to the data structure containing the error field.
in	*error*	The error to be cleared. This should be one of the values from the InverterError enumeration.

Here is the caller graph for this function:

4.29.2.2 is_error_set()

```
bool is_error_set (
            volatile void * data,
            InverterError error )
```

Checks if an error is set in the error field of a data structure.

This function checks if the specified error bit is set in the error field of a data structure.

Parameters

in	*data*	Pointer to the data structure containing the error field.
in	*error*	The error to be checked. This should be one of the values from the InverterError enumeration.

Returns

true if the specified error is set, false otherwise.

4.29.2.3 set_error()

```
void set_error (
            volatile void * data,
            InverterError error )
```

Sets an error in the error field of a data structure.

This function sets the specified error bit in the error field of a data structure.

Parameters

out	*data*	Pointer to the data structure containing the error field.
in	*error*	The error to be set. This should be one of the values from the InverterError enumeration.

Here is the caller graph for this function:

4.30 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c File Reference

This file provides code for the inverter Finite State Machine.

```
#include "FSM.h"
```

Include dependency graph for FSM.c:

Functions

- void eval_inv_FSM (volatile InverterStruct *inv)

  Execute the finite state machine for inverter.

4.30.1 Detailed Description

This file provides code for the inverter Finite State Machine.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.30.2 Function Documentation

4.30.2.1 eval_inv_FSM()

```
void eval_inv_FSM (
            volatile InverterStruct * inv )
```

Execute the finite state machine for inverter.

Run the Finite State Machine (FSM) for inverter operation control.

This function executes the finite state machine to control the inverter based on its current state.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.31 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c File Reference

This file provides code for the inverter struct.

```
#include "INVERTER.h"
#include "stm32f7xx_hal.h"
#include <math.h>
```

Include dependency graph for INVERTER.c:

Functions

- void initialize_inverter (volatile InverterStruct ∗inv, LED ∗led, GPIO_TypeDef ∗enable_port, uint16_t enable↩_pin, TIM_HandleTypeDef ∗htim, ADC_HandleTypeDef ∗hadc, MotorParameters ∗motor, volatile uint16_t ∗rawADC)

  Initialize the inverter.
- void init_control_loops (volatile InverterStruct ∗inv, MotorParameters ∗motor)

  Initializes the PI controllers.
- void enable_control_loops (volatile InverterStruct ∗inv)

  Enables the PI controllers.
- void disable_control_loops (volatile InverterStruct ∗inv)

  Disables the PI controllers.

Variables

- volatile InverterStruct inverter_left = {0}

  Left inverter structure.
- volatile InverterStruct inverter_right = {0}

  Right inverter structure.

4.31.1 Detailed Description

This file provides code for the inverter struct.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.31.2 Function Documentation

4.31.2.1 disable_control_loops()

```
void disable_control_loops (
            volatile InverterStruct * inv )
```

Disables the PI controllers.

Parameters

inv	Pointer to the inverter structure.

4.31.2.2 enable_control_loops()

```
void enable_control_loops (
            volatile InverterStruct * inv )
```

Enables the PI controllers.

Parameters

inv	Pointer to the inverter structure.

4.31.2.3 init_control_loops()

```
void init_control_loops (
            volatile InverterStruct * inv,
            MotorParameters * motor )
```

Initializes the PI controllers.

Initializes the id-iq current control PI controllers.

Parameters

inv	Pointer to the inverter structure.

Here is the caller graph for this function:

4.31.2.4 initialize_inverter()

```
void initialize_inverter (
            volatile InverterStruct * inv,
            LED * led,
            GPIO_TypeDef * enable_port,
            uint16_t enable_pin,
            TIM_HandleTypeDef * htim,
            ADC_HandleTypeDef * hadc,
            MotorParameters * motor,
            volatile uint16_t * rawADC )
```

Initialize the inverter.

This function initializes the inverter structure with the specified LED, GPIO port, and pin.

Parameters

out	*inv*	Pointer to the inverter structure.
in	*led*	Pointer to the LED structure.
in	*enable_port*	Pointer to the GPIO port for enabling/disabling the inverter.
in	*enable_pin*	Pin number for enabling/disabling the inverter.
in	*htim*	Timer peripheral for the PWM output.
in	*hadc*	ADC peripheral for the current phase current and DC voltage sensing.
in	*motor*	MotorParameters struct.

Here is the call graph for this function:

Here is the caller graph for this function:

4.31.3 Variable Documentation

4.31.3.1 inverter_left

```
volatile InverterStruct inverter_left = {0}
```

Left inverter structure.

External declaration of the left inverter structure.

4.31.3.2 inverter_right

```
volatile InverterStruct inverter_right = {0}
```

Right inverter structure.

External declaration of the right inverter structure.

4.32 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "adc.h"
#include "can.h"
#include "dac.h"
#include "dma.h"
#include "i2c.h"
#include "tim.h"
#include "usb_otg.h"
#include "gpio.h"
#include "FSM.h"
```

```
#include "INVERTER.h"
```

Include dependency graph for main.c:

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

  The application entry point.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.32.1 Detailed Description

: Main program body

Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

4.32.2 Function Documentation

4.32.2.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

Here is the caller graph for this function:

4.32.2.2 main()

```
int main (
            void  )
```

The application entry point.

Return values

int	

Here is the call graph for this function:

4.32.2.3 SystemClock_Config()

```
void SystemClock_Config (
```

```
void )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Activate the Over-Drive mode

Initializes the CPU, AHB and APB buses clocksHere is the call graph for this function:

Here is the caller graph for this function:

4.33 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/↩ MEASUREMENTS.c File Reference

This file provides functions for handling measurements.

```
#include "MEASUREMENTS.h"
#include <math.h>
```

```
#include <PergaMOD.h>
```

Include dependency graph for MEASUREMENTS.c:

Functions

- void get_currents_voltage (volatile uint16_t rawADC[ ], volatile Analog ∗analog, volatile Feedback ∗feedback, volatile InverterError ∗errors, float sinTheta_e, float cosTheta_e)

  Get electrical ADC measurements.
- float get_linear (uint32_t bits, float slope, float offset)

  Convert ADC reading to physical measurement with linear response.
- void get_idiq (float ia, float ib, float ic, float sinTheta_e, float cosTheta_e, float ∗idMeas, float ∗iqMeas)

  Computes d-q currents from current measurements and electrical angle.
- float get_temperature (uint32_t bits, const float tempLUT[ ])

  Retrieves temperature from a lookup table based on ADC bits.
- void calibrate_offsets (volatile uint16_t rawADC[ ], volatile float currentOffsets[ ], uint32_t numSamples)

  Calibrate the current sensor offsets.

Variables

- const float tempInverterLUT [ ] = {-2.45, -2.44, -2.44, -2.43, -2.42, -2.42, -2.41, -2.41, -2.40, -2.39, -2.39, -2.38, -2.37, -2.37, -2.36, -2.36, -2.35, -2.34, -2.34, -2.33, -2.32, -2.32, -2.31, -2.31, -2.30, -2.29, -2.29, -2.28, -2.27, -2.27, -2.26, -2.26, -2.25, -2.24, -2.24, -2.23, -2.22, -2.22, -2.21, -2.20, -2.20, -2.19, -2.19, -2.18, -2.17, -2.17, -2.16, -2.15, -2.15, -2.14, -2.14, -2.13, -2.12, -2.12, -2.11, -2.10, -2.10, -2.09, -2.08, -2.08, -2.07, -2.07, -2.06, -2.05, -2.05, -2.04, -2.03, -2.03, -2.02, -2.01, -2.01, -2.00, -2.00, -1.99, -1.98, -1.98, -1.97, -1.96, -1.96, -1.95, -1.94, -1.94, -1.93, -1.93, -1.92, -1.91, -1.91, -1.90, -1.89, -1.89, -1.88, -1.87, -1.87, -1.86, -1.86, -1.85, -1.84, -1.84, -1.83, -1.82, -1.82, -1.81, -1.80, -1.80, -1.79, -1.78, -1.78, -1.77, -1.77, -1.76, -1.75, -1.75, -1.74, -1.73, -1.73, -1.72, -1.71, -1.71, -1.70, -1.69, -1.69, -1.68, -1.67, -1.67, -1.66, -1.66, -1.65, -1.64, -1.64, -1.63, -1.62, -1.62, -1.61, -1.60, -1.60, -1.59, -1.58, -1.58, -1.57, -1.56, -1.56, -1.55, -1.54, -1.54, -1.53, -1.53, -1.52,

-1.51, -1.51, -1.50, -1.49, -1.49, -1.48, -1.47, -1.47, -1.46, -1.45, -1.45, -1.44, -1.43, -1.43, -1.42, -1.41, -1.41, -1.40, -1.39, -1.39, -1.38, -1.37, -1.37, -1.36, -1.36, -1.35, -1.34, -1.34, -1.33, -1.32, -1.32, -1.31, -1.30, -1.30, -1.29, -1.28, -1.28, -1.27, -1.26, -1.26, -1.25, -1.24, -1.24, -1.23, -1.22, -1.22, -1.21, -1.20, -1.20, -1.19, -1.18, -1.18, -1.17, -1.16, -1.16, -1.15, -1.14, -1.14, -1.13, -1.12, -1.12, -1.11, -1.10, -1.10, -1.09, -1.08, -1.08, -1.07, -1.06, -1.06, -1.05, -1.04, -1.04, -1.03, -1.02, -1.02, -1.01, -1.00, -1.00, -0.99, -0.98, -0.98, -0.97, -0.96, -0.96, -0.95, -0.94, -0.94, -0.93, -0.92, -0.92, -0.91, -0.90, -0.90, -0.89, -0.88, -0.88, -0.87, -0.86, -0.86, -0.85, -0.84, -0.84, -0.83, -0.82, -0.82, -0.81, -0.80, -0.80, -0.79, -0.78, -0.78, -0.77, -0.76, -0.76, -0.75, -0.74, -0.73, -0.73, -0.72, -0.71, -0.71, -0.70, -0.69, -0.69, -0.68, -0.67, -0.67, -0.66, -0.65, -0.65, -0.64, -0.63, -0.63, -0.62, -0.61, -0.61, -0.60, -0.59, -0.59, -0.58, -0.57, -0.56, -0.56, -0.55, -0.54, -0.54, -0.53, -0.52, -0.52, -0.51, -0.50, -0.50, -0.49, -0.48, -0.48, -0.47, -0.46, -0.46, -0.45, -0.44, -0.43, -0.43, -0.42, -0.41, -0.41, -0.40, -0.39, -0.39, -0.38, -0.37, -0.37, -0.36, -0.35, -0.35, -0.34, -0.33, -0.32, -0.32, -0.31, -0.30, -0.30, -0.29, -0.28, -0.28, -0.27, -0.26, -0.26, -0.25, -0.24, -0.23, -0.23, -0.22, -0.21, -0.21, -0.20, -0.19, -0.19, -0.18, -0.17, -0.17, -0.16, -0.15, -0.14, -0.14, -0.13, -0.12, -0.12, -0.11, -0.10, -0.10, -0.09, -0.08, -0.07, -0.07, -0.06, -0.05, -0.05, -0.04, -0.03, -0.03, -0.02, -0.01, -0.00, 0.00, 0.01, 0.02, 0.02, 0.03, 0.04, 0.04, 0.05, 0.06, 0.07, 0.07, 0.08, 0.09, 0.09, 0.10, 0.11, 0.12, 0.12, 0.13, 0.14, 0.14, 0.15, 0.16, 0.16, 0.17, 0.18, 0.19, 0.19, 0.20, 0.21, 0.21, 0.22, 0.23, 0.24, 0.24, 0.25, 0.26, 0.26, 0.27, 0.28, 0.29, 0.29, 0.30, 0.31, 0.31, 0.32, 0.33, 0.34, 0.34, 0.35, 0.36, 0.36, 0.37, 0.38, 0.39, 0.39, 0.40, 0.41, 0.41, 0.42, 0.43, 0.44, 0.44, 0.45, 0.46, 0.46, 0.47, 0.48, 0.49, 0.49, 0.50, 0.51, 0.51, 0.52, 0.53, 0.54, 0.54, 0.55, 0.56, 0.56, 0.57, 0.58, 0.59, 0.59, 0.60, 0.61, 0.61, 0.62, 0.63, 0.64, 0.64, 0.65, 0.66, 0.67, 0.67, 0.68, 0.69, 0.69, 0.70, 0.71, 0.72, 0.72, 0.73, 0.74, 0.75, 0.75, 0.76, 0.77, 0.77, 0.78, 0.79, 0.80, 0.80, 0.81, 0.82, 0.83, 0.83, 0.84, 0.85, 0.85, 0.86, 0.87, 0.88, 0.88, 0.89, 0.90, 0.91, 0.91, 0.92, 0.93, 0.94, 0.94, 0.95, 0.96, 0.96, 0.97, 0.98, 0.99, 0.99, 1.00, 1.01, 1.02, 1.02, 1.03, 1.04, 1.05, 1.05, 1.06, 1.07, 1.08, 1.08, 1.09, 1.10, 1.10, 1.11, 1.12, 1.13, 1.13, 1.14, 1.15, 1.16, 1.16, 1.17, 1.18, 1.19, 1.19, 1.20, 1.21, 1.22, 1.22, 1.23, 1.24, 1.25, 1.25, 1.26, 1.27, 1.28, 1.28, 1.29, 1.30, 1.31, 1.31, 1.32, 1.33, 1.34, 1.34, 1.35, 1.36, 1.37, 1.37, 1.38, 1.39, 1.40, 1.40, 1.41, 1.42, 1.43, 1.43, 1.44, 1.45, 1.46, 1.46, 1.47, 1.48, 1.49, 1.49, 1.50, 1.51, 1.52, 1.52, 1.53, 1.54, 1.55, 1.55, 1.56, 1.57, 1.58, 1.58, 1.59, 1.60, 1.61, 1.61, 1.62, 1.63, 1.64, 1.64, 1.65, 1.66, 1.67, 1.67, 1.68, 1.69, 1.70, 1.71, 1.71, 1.72, 1.73, 1.74, 1.74, 1.75, 1.76, 1.77, 1.77, 1.78, 1.79, 1.80, 1.80, 1.81, 1.82, 1.83, 1.84, 1.84, 1.85, 1.86, 1.87, 1.87, 1.88, 1.89, 1.90, 1.90, 1.91, 1.92, 1.93, 1.93, 1.94, 1.95, 1.96, 1.97, 1.97, 1.98, 1.99, 2.00, 2.00, 2.01, 2.02, 2.03, 2.04, 2.04, 2.05, 2.06, 2.07, 2.07, 2.08, 2.09, 2.10, 2.10, 2.11, 2.12, 2.13, 2.14, 2.14, 2.15, 2.16, 2.17, 2.17, 2.18, 2.19, 2.20, 2.21, 2.21, 2.22, 2.23, 2.24, 2.25, 2.25, 2.26, 2.27, 2.28, 2.28, 2.29, 2.30, 2.31, 2.32, 2.32, 2.33, 2.34, 2.35, 2.35, 2.36, 2.37, 2.38, 2.39, 2.39, 2.40, 2.41, 2.42, 2.43, 2.43, 2.44, 2.45, 2.46, 2.46, 2.47, 2.48, 2.49, 2.50, 2.50, 2.51, 2.52, 2.53, 2.54, 2.54, 2.55, 2.56, 2.57, 2.58, 2.58, 2.59, 2.60, 2.61, 2.62, 2.62, 2.63, 2.64, 2.65, 2.66, 2.66, 2.67, 2.68, 2.69, 2.70, 2.70, 2.71, 2.72, 2.73, 2.74, 2.74, 2.75, 2.76, 2.77, 2.78, 2.78, 2.79, 2.80, 2.81, 2.82, 2.82, 2.83, 2.84, 2.85, 2.86, 2.86, 2.87, 2.88, 2.89, 2.90, 2.90, 2.91, 2.92, 2.93, 2.94, 2.94, 2.95, 2.96, 2.97, 2.98, 2.98, 2.99, 3.00, 3.01, 3.02, 3.02, 3.03, 3.04, 3.05, 3.06, 3.07, 3.07, 3.08, 3.09, 3.10, 3.11, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.16, 3.17, 3.18, 3.19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31, 3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.56, 3.57, 3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83, 3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09, 4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35, 4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62, 4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.76, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.86, 4.87, 4.88, 4.89, 4.90, 4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.04, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45, 5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74, 5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6.00, 6.01, 6.01, 6.02, 6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32, 6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61, 6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91, 6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02,

7.03, 7.04, 7.05, 7.06, 7.07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52, 7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83, 7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47, 8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80, 8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47, 9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81, 9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.13, 10.14, 10.16, 10.17, 10.↩
18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.↩
53, 10.54, 10.55, 10.57, 10.58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.88, 10.↩
89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.↩
26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.↩
63, 11.64, 11.65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.97, 11.98, 11.99, 12.↩
00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.↩
39, 12.40, 12.41, 12.43, 12.44, 12.45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.↩
78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.11, 13.12, 13.14, 13.15, 13.16, 13.↩
18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.↩
58, 13.59, 13.61, 13.62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.97, 13.98, 13.↩
99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.↩
41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.↩
84, 14.86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.22, 15.24, 15.25, 15.26, 15.↩
28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.↩
72, 15.74, 15.75, 15.77, 15.78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.16, 16.↩
18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.↩
64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.↩
11, 17.13, 17.14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.55, 17.56, 17.58, 17.↩
60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.↩
09, 18.11, 18.12, 18.14, 18.15, 18.17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.↩
59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.02, 19.04, 19.06, 19.08, 19.09, 19.↩

11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.↩
64, 19.65, 19.67, 19.69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.14, 20.16, 20.↩
18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.↩
73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.↩
29, 21.31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.80, 21.82, 21.84, 21.86, 21.↩
87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.↩
47, 22.49, 22.51, 22.53, 22.55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.06, 23.↩
08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.↩
71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.↩
35, 24.37, 24.39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.95, 24.97, 24.99, 25.↩
01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.↩
69, 25.72, 25.74, 25.76, 25.78, 25.81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.↩
39, 26.42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.00, 27.02, 27.05, 27.07, 27.09, 27.↩
12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.↩
86, 27.89, 27.91, 27.93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.58, 28.60, 28.↩
63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.↩
42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.↩
24, 30.27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.98, 31.01, 31.04, 31.07, 31.↩
09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.↩
97, 32.00, 32.03, 32.06, 32.09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.86, 32.↩
89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.↩
84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.↩
82, 34.85, 34.89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.75, 35.78, 35.81, 35.↩
85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.↩
92, 36.95, 36.99, 37.02, 37.06, 37.09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.↩
04, 38.07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.01, 39.05, 39.09, 39.13, 39.17, 39.↩
21, 39.24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.↩
43, 40.47, 40.51, 40.55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.64, 41.68, 41.↩
72, 41.76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.↩

08, 43.12, 43.17, 43.21, 43.26, 43.30, 43.35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.↩
51, 44.56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.82, 45.87, 45.92, 45.97, 46.↩
02, 46.07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.↩
63, 47.69, 47.74, 47.79, 47.85, 47.90, 47.95, 48.01, 48.06, 48.12, 48.17, 48.23, 48.28, 48.34, 48.39, 48.45, 48.50, 48.56, 48.61, 48.67, 48.72, 48.78, 48.84, 48.89, 48.95, 49.00, 49.06, 49.12, 49.17, 49.23, 49.29, 49.↩
34, 49.40, 49.46, 49.52, 49.57, 49.63, 49.69, 49.75, 49.80, 49.86, 49.92, 49.98, 50.04, 50.10, 50.16, 50.21, 50.27, 50.33, 50.39, 50.45, 50.51, 50.57, 50.63, 50.69, 50.75, 50.81, 50.87, 50.93, 50.99, 51.05, 51.11, 51.↩
17, 51.24, 51.30, 51.36, 51.42, 51.48, 51.54, 51.61, 51.67, 51.73, 51.79, 51.86, 51.92, 51.98, 52.04, 52.11, 52.17, 52.23, 52.30, 52.36, 52.43, 52.49, 52.55, 52.62, 52.68, 52.75, 52.81, 52.88, 52.94, 53.01, 53.07, 53.↩
14, 53.21, 53.27, 53.34, 53.40, 53.47, 53.54, 53.60, 53.67, 53.74, 53.80, 53.87, 53.94, 54.01, 54.08, 54.14, 54.21, 54.28, 54.35, 54.42, 54.49, 54.56, 54.63, 54.70, 54.76, 54.83, 54.90, 54.98, 55.05, 55.12, 55.19, 55.↩
26, 55.33, 55.40, 55.47, 55.54, 55.62, 55.69, 55.76, 55.83, 55.91, 55.98, 56.05, 56.12, 56.20, 56.27, 56.35, 56.42, 56.49, 56.57, 56.64, 56.72, 56.79, 56.87, 56.94, 57.02, 57.10, 57.17, 57.25, 57.32, 57.40, 57.48, 57.↩
56, 57.63, 57.71, 57.79, 57.87, 57.94, 58.02, 58.10, 58.18, 58.26, 58.34, 58.42, 58.50, 58.58, 58.66, 58.74, 58.82, 58.90, 58.98, 59.06, 59.15, 59.23, 59.31, 59.39, 59.48, 59.56, 59.64, 59.72, 59.81, 59.89, 59.98, 60.↩
06, 60.15, 60.23, 60.32, 60.40, 60.49, 60.57, 60.66, 60.75, 60.83, 60.92, 61.01, 61.10, 61.18, 61.27, 61.36, 61.45, 61.54, 61.63, 61.72, 61.81, 61.90, 61.99, 62.08, 62.17, 62.26, 62.35, 62.44, 62.54, 62.63, 62.72, 62.↩
82, 62.91, 63.00, 63.10, 63.19, 63.29, 63.38, 63.48, 63.57, 63.67, 63.76, 63.86, 63.96, 64.06, 64.15, 64.25, 64.35, 64.45, 64.55, 64.65, 64.75, 64.85, 64.95, 65.05, 65.15, 65.25, 65.35, 65.46, 65.56, 65.66, 65.76, 65.↩
87, 65.97, 66.08, 66.18, 66.29, 66.39, 66.50, 66.61, 66.71, 66.82, 66.93, 67.03, 67.14, 67.25, 67.36, 67.47, 67.58, 67.69, 67.80, 67.91, 68.03, 68.14, 68.25, 68.36, 68.48, 68.59, 68.71, 68.82, 68.94, 69.05, 69.17, 69.↩
29, 69.40, 69.52, 69.64, 69.76, 69.88, 70.00, 70.12, 70.24, 70.36, 70.48, 70.60, 70.73, 70.85, 70.97, 71.10, 71.22, 71.35, 71.47, 71.60, 71.73, 71.86, 71.98, 72.11, 72.24, 72.37, 72.50, 72.63, 72.76, 72.90, 73.03, 73.↩
16, 73.30, 73.43, 73.57, 73.70, 73.84, 73.98, 74.11, 74.25, 74.39, 74.53, 74.67, 74.81, 74.95, 75.10, 75.24, 75.38, 75.53, 75.67, 75.82, 75.97, 76.11, 76.26, 76.41, 76.56, 76.71, 76.86, 77.01, 77.17, 77.32, 77.47, 77.↩
63, 77.78, 77.94, 78.10, 78.26, 78.42, 78.58, 78.74, 78.90, 79.06, 79.23, 79.39, 79.55, 79.72, 79.89, 80.06, 80.23, 80.40, 80.57, 80.74, 80.91, 81.08, 81.26, 81.44, 81.61, 81.79, 81.97, 82.15, 82.33, 82.51, 82.70, 82.↩
88, 83.07, 83.25, 83.44, 83.63, 83.82, 84.01, 84.20, 84.40, 84.59, 84.79, 84.98, 85.18, 85.38, 85.58, 85.79, 85.99, 86.20, 86.40, 86.61, 86.82, 87.03, 87.24, 87.46, 87.67, 87.89, 88.11, 88.33, 88.55, 88.77, 89.00, 89.↩
22, 89.45, 89.68, 89.91, 90.14, 90.38, 90.62, 90.85, 91.09, 91.34, 91.58, 91.82, 92.07, 92.32, 92.57, 92.83, 93.08, 93.34, 93.60, 93.86, 94.13, 94.39, 94.66, 94.93, 95.20, 95.48, 95.76, 96.04, 96.32, 96.61, 96.89, 97.↩
18, 97.48, 97.77, 98.07, 98.37, 98.68, 98.98, 99.29, 99.61, 99.92, 100.24, 100.56, 100.89, 101.22, 101.55, 101.88, 102.22, 102.56, 102.91, 103.26, 103.61, 103.97, 104.33, 104.70, 105.07, 105.44, 105.82, 106.20, 106.58, 106.98, 107.37, 107.77, 108.18, 108.59, 109.00, 109.42, 109.85, 110.28, 110.71, 111.16, 111.60, 112.06, 112.52, 112.99, 113.46, 113.94, 114.43, 114.92, 115.42, 115.93, 116.44, 116.97, 117.50, 118.04, 118.59, 119.14, 119.71, 120.29, 120.87, 121.47, 122.07, 122.69, 123.32, 123.95, 124.61, 125.27, 125.94, 126.63, 127.33, 128.05, 128.78, 129.52, 130.29, 131.06, 131.86, 132.67, 133.50, 134.35, 135.22, 136.12, 137.03, 137.97, 138.93, 139.92, 140.93, 141.97, 143.05, 144.15, 145.29, 146.46, 147.67, 148.92, 150.22, 151.55, 152.94, 154.38, 155.87, 157.42, 159.04, 160.72, 162.48, 164.33, 166.26, 168.29, 170.42, 172.68, 175.07, 177.60, 180.30, 183.19, 186.29, 189.63, 193.26, 197.23, 201.59, 206.43, 211.86, 218.04, 225.19, 233.64, 243.93, 257.02, 274.82, 302.24, 360.00, 421.98, 421.98, 421.98

421.98, 421.98, 421.98,

2.98, 2.99, 3.00, 3.01, 3.02, 3.02, 3.03, 3.04, 3.05, 3.06, 3.07, 3.07, 3.08, 3.09, 3.10, 3.11, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.16, 3.17, 3.18, 3.19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31, 3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.56, 3.57, 3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83, 3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09, 4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35, 4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62, 4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.76, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.86, 4.87, 4.88, 4.89, 4.90, 4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.04, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45, 5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74, 5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6.00, 6.01, 6.01, 6.02, 6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32, 6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61, 6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91, 6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52, 7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83, 7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47, 8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80, 8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47, 9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81, 9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.13, 10.14, 10.16, 10.17, 10.18, 10.↩
19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.↩
54, 10.55, 10.57, 10.58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.88, 10.89, 10.↩
90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.↩
27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.↩
64, 11.65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.97, 11.98, 11.99, 12.00, 12.↩
02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.↩
40, 12.41, 12.43, 12.44, 12.45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.78, 12.↩
79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.↩
19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.↩

59, 13.61, 13.62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.97, 13.98, 13.99, 14.↩
01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.↩
43, 14.44, 14.45, 14.47, 14.48, 14.50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.↩
86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.22, 15.24, 15.25, 15.26, 15.28, 15.↩
29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.↩
74, 15.75, 15.77, 15.78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.16, 16.18, 16.↩
19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64, 16.↩
66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.↩
13, 17.14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.55, 17.56, 17.58, 17.60, 17.↩
61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.↩
11, 18.12, 18.14, 18.15, 18.17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.59, 18.↩
61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.↩
13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.↩
65, 19.67, 19.69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.14, 20.16, 20.18, 20.↩
19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.↩
75, 20.76, 20.78, 20.80, 20.82, 20.84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.↩
31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.80, 21.82, 21.84, 21.86, 21.87, 21.↩
89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.↩
49, 22.51, 22.53, 22.55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.06, 23.08, 23.↩
10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.↩
73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.↩
37, 24.39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.95, 24.97, 24.99, 25.01, 25.↩
03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.↩
72, 25.74, 25.76, 25.78, 25.81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.39, 26.↩
42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.↩
14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.↩
89, 27.91, 27.93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.58, 28.60, 28.63, 28.↩
66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.↩
45, 29.48, 29.50, 29.53, 29.55, 29.58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.↩

27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.98, 31.01, 31.04, 31.07, 31.09, 31.↩
12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.↩
00, 32.03, 32.06, 32.09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.86, 32.89, 32.↩
92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.↩
87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.↩
85, 34.89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.75, 35.78, 35.81, 35.85, 35.↩
88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.92, 36.↩
95, 36.99, 37.02, 37.06, 37.09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.04, 38.↩
07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.01, 39.05, 39.09, 39.13, 39.17, 39.21, 39.↩
24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.43, 40.↩
47, 40.51, 40.55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.64, 41.68, 41.72, 41.↩
76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.08, 43.↩
12, 43.17, 43.21, 43.26, 43.30, 43.35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.51, 44.↩
56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.82, 45.87, 45.92, 45.97, 46.02, 46.↩
07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.63, 47.↩
69, 47.74, 47.79, 47.85, 47.90, 47.95, 48.01, 48.06, 48.12, 48.17, 48.23, 48.28, 48.34, 48.39, 48.45, 48.50, 48.56, 48.61, 48.67, 48.72, 48.78, 48.84, 48.89, 48.95, 49.00, 49.06, 49.12, 49.17, 49.23, 49.29, 49.34, 49.↩
40, 49.46, 49.52, 49.57, 49.63, 49.69, 49.75, 49.80, 49.86, 49.92, 49.98, 50.04, 50.10, 50.16, 50.21, 50.27, 50.33, 50.39, 50.45, 50.51, 50.57, 50.63, 50.69, 50.75, 50.81, 50.87, 50.93, 50.99, 51.05, 51.11, 51.17, 51.↩
24, 51.30, 51.36, 51.42, 51.48, 51.54, 51.61, 51.67, 51.73, 51.79, 51.86, 51.92, 51.98, 52.04, 52.11, 52.17, 52.23, 52.30, 52.36, 52.43, 52.49, 52.55, 52.62, 52.68, 52.75, 52.81, 52.88, 52.94, 53.01, 53.07, 53.14, 53.↩
21, 53.27, 53.34, 53.40, 53.47, 53.54, 53.60, 53.67, 53.74, 53.80, 53.87, 53.94, 54.01, 54.08, 54.14, 54.21, 54.28, 54.35, 54.42, 54.49, 54.56, 54.63, 54.70, 54.76, 54.83, 54.90, 54.98, 55.05, 55.12, 55.19, 55.26, 55.↩
33, 55.40, 55.47, 55.54, 55.62, 55.69, 55.76, 55.83, 55.91, 55.98, 56.05, 56.12, 56.20, 56.27, 56.35, 56.42, 56.49, 56.57, 56.64, 56.72, 56.79, 56.87, 56.94, 57.02, 57.10, 57.17, 57.25, 57.32, 57.40, 57.48, 57.56, 57.↩
63, 57.71, 57.79, 57.87, 57.94, 58.02, 58.10, 58.18, 58.26, 58.34, 58.42, 58.50, 58.58, 58.66, 58.74, 58.82, 58.90, 58.98, 59.06, 59.15, 59.23, 59.31, 59.39, 59.48, 59.56, 59.64, 59.72, 59.81, 59.89, 59.98, 60.06, 60.↩
15, 60.23, 60.32, 60.40, 60.49, 60.57, 60.66, 60.75, 60.83, 60.92, 61.01, 61.10, 61.18, 61.27, 61.36, 61.45, 61.54, 61.63, 61.72, 61.81, 61.90, 61.99, 62.08, 62.17, 62.26, 62.35, 62.44, 62.54, 62.63, 62.72, 62.82, 62.↩
91, 63.00, 63.10, 63.19, 63.29, 63.38, 63.48, 63.57, 63.67, 63.76, 63.86, 63.96, 64.06, 64.15, 64.25, 64.35, 64.45, 64.55, 64.65, 64.75, 64.85, 64.95, 65.05, 65.15, 65.25, 65.35, 65.46, 65.56, 65.66, 65.76, 65.87, 65.↩
97, 66.08, 66.18, 66.29, 66.39, 66.50, 66.61, 66.71, 66.82, 66.93, 67.03, 67.14, 67.25, 67.36, 67.47, 67.58, 67.69, 67.80, 67.91, 68.03, 68.14, 68.25, 68.36, 68.48, 68.59, 68.71, 68.82, 68.94, 69.05, 69.17, 69.29, 69.↩
40, 69.52, 69.64, 69.76, 69.88, 70.00, 70.12, 70.24, 70.36, 70.48, 70.60, 70.73, 70.85, 70.97, 71.10, 71.22, 71.35, 71.47, 71.60, 71.73, 71.86, 71.98, 72.11, 72.24, 72.37, 72.50, 72.63, 72.76, 72.90, 73.03, 73.16, 73.↩
30, 73.43, 73.57, 73.70, 73.84, 73.98, 74.11, 74.25, 74.39, 74.53, 74.67, 74.81, 74.95, 75.10, 75.24, 75.38, 75.53, 75.67, 75.82, 75.97, 76.11, 76.26, 76.41, 76.56, 76.71, 76.86, 77.01, 77.17, 77.32, 77.47, 77.63, 77.↩
78, 77.94, 78.10, 78.26, 78.42, 78.58, 78.74, 78.90, 79.06, 79.23, 79.39, 79.55, 79.72, 79.89, 80.06, 80.23, 80.40, 80.57, 80.74, 80.91, 81.08, 81.26, 81.44, 81.61, 81.79, 81.97, 82.15, 82.33, 82.51, 82.70, 82.88, 83.↩
07, 83.25, 83.44, 83.63, 83.82, 84.01, 84.20, 84.40, 84.59, 84.79, 84.98, 85.18, 85.38, 85.58, 85.79, 85.99, 86.20, 86.40, 86.61, 86.82, 87.03, 87.24, 87.46, 87.67, 87.89, 88.11, 88.33, 88.55, 88.77, 89.00, 89.22, 89.↩
45, 89.68, 89.91, 90.14, 90.38, 90.62, 90.85, 91.09, 91.34, 91.58, 91.82, 92.07, 92.32, 92.57, 92.83, 93.08, 93.34, 93.60, 93.86, 94.13, 94.39, 94.66, 94.93, 95.20, 95.48, 95.76, 96.04, 96.32, 96.61, 96.89, 97.18, 97.↩

48, 97.77, 98.07, 98.37, 98.68, 98.98, 99.29, 99.61, 99.92, 100.24, 100.56, 100.89, 101.22, 101.55, 101.88, 102.22, 102.56, 102.91, 103.26, 103.61, 103.97, 104.33, 104.70, 105.07, 105.44, 105.82, 106.20, 106.58, 106.98, 107.37, 107.77, 108.18, 108.59, 109.00, 109.42, 109.85, 110.28, 110.71, 111.16, 111.60, 112.06, 112.52, 112.99, 113.46, 113.94, 114.43, 114.92, 115.42, 115.93, 116.44, 116.97, 117.50, 118.04, 118.59, 119.14, 119.71, 120.29, 120.87, 121.47, 122.07, 122.69, 123.32, 123.95, 124.61, 125.27, 125.94, 126.63, 127.33, 128.05, 128.78, 129.52, 130.29, 131.06, 131.86, 132.67, 133.50, 134.35, 135.22, 136.12, 137.03, 137.97, 138.93, 139.92, 140.93, 141.97, 143.05, 144.15, 145.29, 146.46, 147.67, 148.92, 150.22, 151.55, 152.94, 154.38, 155.87, 157.42, 159.04, 160.72, 162.48, 164.33, 166.26, 168.29, 170.42, 172.68, 175.07, 177.60, 180.30, 183.19, 186.29, 189.63, 193.26, 197.23, 201.59, 206.43, 211.86, 218.04, 225.19, 233.64, 243.93, 257.02, 274.82, 302.24, 360.00, 421.98, 421.98, 421.98

4.33.2 Function Documentation

4.33.2.1 calibrate_offsets()

```
void calibrate_offsets (
            volatile uint16_t rawADC[],
            volatile float currentOffsets[],
            uint32_t numSamples )
```

Calibrate the current sensor offsets.

This function calculates the average offset for each current sensor channel by reading the ADC values when no current is flowing. The calculated offsets are used to correct the sensor readings.

Parameters

in	*rawADC*	Buffer containing the raw ADC values for the channels.
out	*currentOffsets*	Array to store the calculated offsets for each current channel.
in	*numSamples*	Number of samples to average for the offset calculation.

Here is the caller graph for this function:

4.33.2.2 get_currents_voltage()

```
void get_currents_voltage (
            volatile uint16_t rawADC[],
            volatile Analog * analog,
            volatile Feedback * feedback,
            volatile InverterError * errors,
            float sinTheta_e,
            float cosTheta_e )
```

Get electrical ADC measurements.

Parameters

in	*rawADC*	Pointer to the raw ADC values array.
out	*analog*	Pointer to the ADC struct to store the results.
out	*feedback*	Pointer to the Feedback struct to store id and iq.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)
in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)

Here is the call graph for this function:

Here is the caller graph for this function:

4.33.2.3 get_idiq()

```
void get_idiq (
            float ia,
            float ib,
            float ic,
            float sinTheta_e,
            float cosTheta_e,
            float * idMeas,
            float * iqMeas )
```

Computes d-q currents from current measurements and electrical angle.

This function computes the d-q currents from phase currents (ABC), theta_e, and stores the results in the provided pointers.

Parameters

in	*ia*	Phase A current in A.
in	*ib*	Phase B current in A.
in	*ic*	Phase C current in A.
in	*sinTheta↩ _e*	Electrical angle sine (-1..1)

Parameters

in	*cosTheta↩ _e*	Electrical angle cosine (-1..1)
out	*idMeas*	Pointer to store the D-axis current.
out	*iqMeas*	Pointer to store the Q-axis current.

Here is the caller graph for this function:

4.33.2.4 get_linear()

```
float get_linear (
            uint32_t bits,
            float slope,
            float offset )
```

Convert ADC reading to physical measurement with linear response.

Parameters

in	*bits*	The ADC reading.
in	*slope*	The slope (units per volt).
in	*offset*	The offset (volts at zero).

Return values

measurement	The physical measurement.

Here is the caller graph for this function:

4.33.2.5 get_temperature()

```
float get_temperature (
            uint32_t bits,
            const float tempLUT[] )
```

Retrieves temperature from a lookup table based on ADC bits.

This function retrieves temperature from a lookup table based on the ADC bits. The lookup table (LUT) must have a value for each possible ADC bit combination.

Parameters

in	*bits*	ADC reading converted to bits.
in	*tempLUT*	Lookup table containing temperature values.

Returns

Temperature corresponding to the provided ADC bits.

Here is the caller graph for this function:

4.33.3 Variable Documentation

4.33.3.1 rawADC_left

```
volatile uint16_t rawADC_left[4] = {0}
```

Raw ADC data for the left inverter.

External declaration of raw ADC data for the left inverter.

4.33.3.2 rawADC_right

```
volatile uint16_t rawADC_right[4] = {0}
```

Raw ADC data for the right inverter.

External declaration of raw ADC data for the right inverter.

4.33.3.3 rawADC_temp

```
volatile uint16_t rawADC_temp[4] = {0}
```

Raw ADC data for the temperatures.

External declaration of raw ADC data for the temperature readings.

4.33.3.4 tempInverterLUT

```
const float tempInverterLUT[] = {-2.45, -2.44, -2.44, -2.43, -2.42, -2.42, -2.41, -2.41, -2.↩
40, -2.39, -2.39, -2.38, -2.37, -2.37, -2.36, -2.36, -2.35, -2.34, -2.34, -2.33, -2.32, -2.↩
32, -2.31, -2.31, -2.30, -2.29, -2.29, -2.28, -2.27, -2.27, -2.26, -2.26, -2.25, -2.24, -2.↩
24, -2.23, -2.22, -2.22, -2.21, -2.20, -2.20, -2.19, -2.19, -2.18, -2.17, -2.17, -2.16, -2.↩
15, -2.15, -2.14, -2.14, -2.13, -2.12, -2.12, -2.11, -2.10, -2.10, -2.09, -2.08, -2.08, -2.↩
07, -2.07, -2.06, -2.05, -2.05, -2.04, -2.03, -2.03, -2.02, -2.01, -2.01, -2.00, -2.00, -1.↩
99, -1.98, -1.98, -1.97, -1.96, -1.96, -1.95, -1.94, -1.94, -1.93, -1.93, -1.92, -1.91, -1.↩
91, -1.90, -1.89, -1.89, -1.88, -1.87, -1.87, -1.86, -1.86, -1.85, -1.84, -1.84, -1.83, -1.↩
82, -1.82, -1.81, -1.80, -1.80, -1.79, -1.78, -1.78, -1.77, -1.77, -1.76, -1.75, -1.75, -1.↩
74, -1.73, -1.73, -1.72, -1.71, -1.71, -1.70, -1.69, -1.69, -1.68, -1.67, -1.67, -1.66, -1.↩
66, -1.65, -1.64, -1.64, -1.63, -1.62, -1.62, -1.61, -1.60, -1.60, -1.59, -1.58, -1.58, -1.↩
57, -1.56, -1.56, -1.55, -1.54, -1.54, -1.53, -1.53, -1.52, -1.51, -1.51, -1.50, -1.49, -1.↩
49, -1.48, -1.47, -1.47, -1.46, -1.45, -1.45, -1.44, -1.43, -1.43, -1.42, -1.41, -1.41, -1.↩
40, -1.39, -1.39, -1.38, -1.37, -1.37, -1.36, -1.36, -1.35, -1.34, -1.34, -1.33, -1.32, -1.↩
32, -1.31, -1.30, -1.30, -1.29, -1.28, -1.28, -1.27, -1.26, -1.26, -1.25, -1.24, -1.24, -1.↩
23, -1.22, -1.22, -1.21, -1.20, -1.20, -1.19, -1.18, -1.18, -1.17, -1.16, -1.16, -1.15, -1.↩
14, -1.14, -1.13, -1.12, -1.12, -1.11, -1.10, -1.10, -1.09, -1.08, -1.08, -1.07, -1.06, -1.↩
06, -1.05, -1.04, -1.04, -1.03, -1.02, -1.02, -1.01, -1.00, -1.00, -0.99, -0.98, -0.98, -0.↩
97, -0.96, -0.96, -0.95, -0.94, -0.94, -0.93, -0.92, -0.92, -0.91, -0.90, -0.90, -0.89, -0.↩
88, -0.88, -0.87, -0.86, -0.86, -0.85, -0.84, -0.84, -0.83, -0.82, -0.82, -0.81, -0.80, -0.↩
80, -0.79, -0.78, -0.78, -0.77, -0.76, -0.76, -0.75, -0.74, -0.73, -0.73, -0.72, -0.71, -0.↩
71, -0.70, -0.69, -0.69, -0.68, -0.67, -0.67, -0.66, -0.65, -0.65, -0.64, -0.63, -0.63, -0.↩
62, -0.61, -0.61, -0.60, -0.59, -0.59, -0.58, -0.57, -0.56, -0.56, -0.55, -0.54, -0.54, -0.↩
53, -0.52, -0.52, -0.51, -0.50, -0.50, -0.49, -0.48, -0.48, -0.47, -0.46, -0.46, -0.45, -0.↩
44, -0.43, -0.43, -0.42, -0.41, -0.41, -0.40, -0.39, -0.39, -0.38, -0.37, -0.37, -0.36, -0.↩
35, -0.35, -0.34, -0.33, -0.32, -0.32, -0.31, -0.30, -0.30, -0.29, -0.28, -0.28, -0.27, -0.↩
26, -0.26, -0.25, -0.24, -0.23, -0.23, -0.22, -0.21, -0.21, -0.20, -0.19, -0.19, -0.18, -0.↩
17, -0.17, -0.16, -0.15, -0.14, -0.14, -0.13, -0.12, -0.12, -0.11, -0.10, -0.10, -0.09, -0.08,
-0.07, -0.07, -0.06, -0.05, -0.05, -0.04, -0.03, -0.03, -0.02, -0.01, -0.00, 0.00, 0.01, 0.02,
0.02, 0.03, 0.04, 0.04, 0.05, 0.06, 0.07, 0.07, 0.08, 0.09, 0.09, 0.10, 0.11, 0.12, 0.12, 0.↩
13, 0.14, 0.14, 0.15, 0.16, 0.16, 0.17, 0.18, 0.19, 0.19, 0.20, 0.21, 0.21, 0.22, 0.23, 0.24,
0.24, 0.25, 0.26, 0.26, 0.27, 0.28, 0.29, 0.29, 0.30, 0.31, 0.31, 0.32, 0.33, 0.34, 0.34, 0.↩
35, 0.36, 0.36, 0.37, 0.38, 0.39, 0.39, 0.40, 0.41, 0.41, 0.42, 0.43, 0.44, 0.44, 0.45, 0.46,
0.46, 0.47, 0.48, 0.49, 0.49, 0.50, 0.51, 0.51, 0.52, 0.53, 0.54, 0.54, 0.55, 0.56, 0.56, 0.↩
57, 0.58, 0.59, 0.59, 0.60, 0.61, 0.61, 0.62, 0.63, 0.64, 0.64, 0.65, 0.66, 0.67, 0.67, 0.68,
0.69, 0.69, 0.70, 0.71, 0.72, 0.72, 0.73, 0.74, 0.75, 0.75, 0.76, 0.77, 0.77, 0.78, 0.79, 0.↩
80, 0.80, 0.81, 0.82, 0.83, 0.83, 0.84, 0.85, 0.85, 0.86, 0.87, 0.88, 0.88, 0.89, 0.90, 0.91,
0.91, 0.92, 0.93, 0.94, 0.94, 0.95, 0.96, 0.96, 0.97, 0.98, 0.99, 0.99, 1.00, 1.01, 1.02, 1.↩
02, 1.03, 1.04, 1.05, 1.05, 1.06, 1.07, 1.08, 1.08, 1.09, 1.10, 1.10, 1.11, 1.12, 1.13, 1.13,
1.14, 1.15, 1.16, 1.16, 1.17, 1.18, 1.19, 1.19, 1.20, 1.21, 1.22, 1.22, 1.23, 1.24, 1.25, 1.↩
25, 1.26, 1.27, 1.28, 1.28, 1.29, 1.30, 1.31, 1.31, 1.32, 1.33, 1.34, 1.34, 1.35, 1.36, 1.37,
1.37, 1.38, 1.39, 1.40, 1.40, 1.41, 1.42, 1.43, 1.43, 1.44, 1.45, 1.46, 1.46, 1.47, 1.48, 1.↩
49, 1.49, 1.50, 1.51, 1.52, 1.52, 1.53, 1.54, 1.55, 1.55, 1.56, 1.57, 1.58, 1.58, 1.59, 1.60,
1.61, 1.61, 1.62, 1.63, 1.64, 1.64, 1.65, 1.66, 1.67, 1.67, 1.68, 1.69, 1.70, 1.71, 1.71, 1.↩
72, 1.73, 1.74, 1.74, 1.75, 1.76, 1.77, 1.77, 1.78, 1.79, 1.80, 1.80, 1.81, 1.82, 1.83, 1.84,
1.84, 1.85, 1.86, 1.87, 1.87, 1.88, 1.89, 1.90, 1.90, 1.91, 1.92, 1.93, 1.93, 1.94, 1.95, 1.↩
96, 1.97, 1.97, 1.98, 1.99, 2.00, 2.00, 2.01, 2.02, 2.03, 2.04, 2.04, 2.05, 2.06, 2.07, 2.07,
2.08, 2.09, 2.10, 2.10, 2.11, 2.12, 2.13, 2.14, 2.14, 2.15, 2.16, 2.17, 2.17, 2.18, 2.19, 2.↩
20, 2.21, 2.21, 2.22, 2.23, 2.24, 2.25, 2.25, 2.26, 2.27, 2.28, 2.28, 2.29, 2.30, 2.31, 2.32,
2.32, 2.33, 2.34, 2.35, 2.35, 2.36, 2.37, 2.38, 2.39, 2.39, 2.40, 2.41, 2.42, 2.43, 2.43, 2.↩
44, 2.45, 2.46, 2.46, 2.47, 2.48, 2.49, 2.50, 2.50, 2.51, 2.52, 2.53, 2.54, 2.54, 2.55, 2.56,
2.57, 2.58, 2.58, 2.59, 2.60, 2.61, 2.62, 2.62, 2.63, 2.64, 2.65, 2.66, 2.66, 2.67, 2.68, 2.↩
69, 2.70, 2.70, 2.71, 2.72, 2.73, 2.74, 2.74, 2.75, 2.76, 2.77, 2.78, 2.78, 2.79, 2.80, 2.81,
2.82, 2.82, 2.83, 2.84, 2.85, 2.86, 2.86, 2.87, 2.88, 2.89, 2.90, 2.90, 2.91, 2.92, 2.93, 2.↩
94, 2.94, 2.95, 2.96, 2.97, 2.98, 2.98, 2.99, 3.00, 3.01, 3.02, 3.02, 3.03, 3.04, 3.05, 3.06,
3.07, 3.07, 3.08, 3.09, 3.10, 3.11, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.16, 3.17, 3.18, 3.↩
```

```
19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31,
3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.↩
44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.56, 3.57,
3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.↩
70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83,
3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.↩
96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09,
4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.↩
23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35,
4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.↩
49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62,
4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.↩
76, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.86, 4.87, 4.88, 4.89, 4.90,
4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.↩
04, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17,
5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.↩
32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45,
5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.↩
60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74,
5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.↩
88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6.00, 6.01, 6.01, 6.02,
6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.↩
17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32,
6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.↩
47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61,
6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.↩
76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91,
6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.↩
07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21,
7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.↩
37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52,
7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.↩
68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83,
7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.↩
00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15,
8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.↩
32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47,
8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.↩
64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80,
8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.↩
97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13,
9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.↩
30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47,
9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.↩
64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81,
9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.↩
99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.↩
13, 10.14, 10.16, 10.17, 10.18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.↩
28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.↩
43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.54, 10.55, 10.57, 10.↩
58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.↩
73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.↩
88, 10.89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.↩
03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.↩
18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.↩
34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.↩
49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.64, 11.↩
65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.↩
```

```
81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.↩
97, 11.98, 11.99, 12.00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.↩
13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.↩
29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.40, 12.41, 12.43, 12.44, 12.↩
45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.↩
61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.↩
78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.↩
94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.↩
11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.↩
28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.↩
45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.59, 13.61, 13.↩
62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.↩
79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.↩
97, 13.98, 13.99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.↩
14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.↩
32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.↩
50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.↩
67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.↩
86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.↩
04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.↩
22, 15.24, 15.25, 15.26, 15.28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.↩
41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.↩
59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.74, 15.75, 15.77, 15.↩
78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.↩
97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.↩
16, 16.18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.↩
35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.↩
55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.↩
75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.↩
94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.13, 17.↩
14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.↩
35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.↩
55, 17.56, 17.58, 17.60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.↩
75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.↩
96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.11, 18.12, 18.14, 18.15, 18.↩
17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.↩
38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.↩
59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.↩
81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.↩
02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.↩
24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.↩
46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.65, 19.67, 19.↩
69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.↩
91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.↩
14, 20.16, 20.18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.↩
37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.↩
60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.↩
84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.↩
07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.↩
31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.↩
55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.↩
80, 21.82, 21.84, 21.86, 21.87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.↩
05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.↩
30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.49, 22.51, 22.53, 22.↩
55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.↩
80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.↩
06, 23.08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.↩
32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.↩
```

```
58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.↩
85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.↩
12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.37, 24.↩
39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.↩
67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.↩
95, 24.97, 24.99, 25.01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.↩
23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.↩
52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.72, 25.74, 25.76, 25.78, 25.↩
81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.↩
10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.↩
39, 26.42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.↩
69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.↩
00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.↩
31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.↩
62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.89, 27.91, 27.↩
93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.↩
26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.↩
58, 28.60, 28.63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.↩
91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.↩
24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.↩
58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.↩
92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.↩
27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.↩
62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.↩
98, 31.01, 31.04, 31.07, 31.09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.↩
35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.↩
72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.00, 32.03, 32.06, 32.↩
09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.↩
47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.↩
86, 32.89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.↩
25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.↩
65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.↩
05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.↩
47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.85, 34.↩
89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.↩
31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.↩
75, 35.78, 35.81, 35.85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.↩
19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.↩
64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.92, 36.95, 36.99, 37.02, 37.06, 37.↩
09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.↩
56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.↩
04, 38.07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.↩
52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.↩
01, 39.05, 39.09, 39.13, 39.17, 39.21, 39.24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.↩
52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.↩
03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.43, 40.47, 40.51, 40.↩
55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.↩
09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.↩
64, 41.68, 41.72, 41.76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.↩
19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.↩
76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.08, 43.12, 43.17, 43.21, 43.26, 43.30, 43.↩
35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.↩
94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.51, 44.↩
56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.↩
18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.↩
82, 45.87, 45.92, 45.97, 46.02, 46.07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.↩
48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.↩
15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.63, 47.69, 47.74, 47.79, 47.↩
```

```
85, 47.90, 47.95, 48.01, 48.06, 48.12, 48.17, 48.23, 48.28, 48.34, 48.39, 48.45, 48.50, 48.↩
56, 48.61, 48.67, 48.72, 48.78, 48.84, 48.89, 48.95, 49.00, 49.06, 49.12, 49.17, 49.23, 49.↩
29, 49.34, 49.40, 49.46, 49.52, 49.57, 49.63, 49.69, 49.75, 49.80, 49.86, 49.92, 49.98, 50.↩
04, 50.10, 50.16, 50.21, 50.27, 50.33, 50.39, 50.45, 50.51, 50.57, 50.63, 50.69, 50.75, 50.↩
81, 50.87, 50.93, 50.99, 51.05, 51.11, 51.17, 51.24, 51.30, 51.36, 51.42, 51.48, 51.54, 51.↩
61, 51.67, 51.73, 51.79, 51.86, 51.92, 51.98, 52.04, 52.11, 52.17, 52.23, 52.30, 52.36, 52.↩
43, 52.49, 52.55, 52.62, 52.68, 52.75, 52.81, 52.88, 52.94, 53.01, 53.07, 53.14, 53.21, 53.↩
27, 53.34, 53.40, 53.47, 53.54, 53.60, 53.67, 53.74, 53.80, 53.87, 53.94, 54.01, 54.08, 54.↩
14, 54.21, 54.28, 54.35, 54.42, 54.49, 54.56, 54.63, 54.70, 54.76, 54.83, 54.90, 54.98, 55.↩
05, 55.12, 55.19, 55.26, 55.33, 55.40, 55.47, 55.54, 55.62, 55.69, 55.76, 55.83, 55.91, 55.↩
98, 56.05, 56.12, 56.20, 56.27, 56.35, 56.42, 56.49, 56.57, 56.64, 56.72, 56.79, 56.87, 56.↩
94, 57.02, 57.10, 57.17, 57.25, 57.32, 57.40, 57.48, 57.56, 57.63, 57.71, 57.79, 57.87, 57.↩
94, 58.02, 58.10, 58.18, 58.26, 58.34, 58.42, 58.50, 58.58, 58.66, 58.74, 58.82, 58.90, 58.↩
98, 59.06, 59.15, 59.23, 59.31, 59.39, 59.48, 59.56, 59.64, 59.72, 59.81, 59.89, 59.98, 60.↩
06, 60.15, 60.23, 60.32, 60.40, 60.49, 60.57, 60.66, 60.75, 60.83, 60.92, 61.01, 61.10, 61.↩
18, 61.27, 61.36, 61.45, 61.54, 61.63, 61.72, 61.81, 61.90, 61.99, 62.08, 62.17, 62.26, 62.↩
35, 62.44, 62.54, 62.63, 62.72, 62.82, 62.91, 63.00, 63.10, 63.19, 63.29, 63.38, 63.48, 63.↩
57, 63.67, 63.76, 63.86, 63.96, 64.06, 64.15, 64.25, 64.35, 64.45, 64.55, 64.65, 64.75, 64.↩
85, 64.95, 65.05, 65.15, 65.25, 65.35, 65.46, 65.56, 65.66, 65.76, 65.87, 65.97, 66.08, 66.↩
18, 66.29, 66.39, 66.50, 66.61, 66.71, 66.82, 66.93, 67.03, 67.14, 67.25, 67.36, 67.47, 67.↩
58, 67.69, 67.80, 67.91, 68.03, 68.14, 68.25, 68.36, 68.48, 68.59, 68.71, 68.82, 68.94, 69.↩
05, 69.17, 69.29, 69.40, 69.52, 69.64, 69.76, 69.88, 70.00, 70.12, 70.24, 70.36, 70.48, 70.↩
60, 70.73, 70.85, 70.97, 71.10, 71.22, 71.35, 71.47, 71.60, 71.73, 71.86, 71.98, 72.11, 72.↩
24, 72.37, 72.50, 72.63, 72.76, 72.90, 73.03, 73.16, 73.30, 73.43, 73.57, 73.70, 73.84, 73.↩
98, 74.11, 74.25, 74.39, 74.53, 74.67, 74.81, 74.95, 75.10, 75.24, 75.38, 75.53, 75.67, 75.↩
82, 75.97, 76.11, 76.26, 76.41, 76.56, 76.71, 76.86, 77.01, 77.17, 77.32, 77.47, 77.63, 77.↩
78, 77.94, 78.10, 78.26, 78.42, 78.58, 78.74, 78.90, 79.06, 79.23, 79.39, 79.55, 79.72, 79.↩
89, 80.06, 80.23, 80.40, 80.57, 80.74, 80.91, 81.08, 81.26, 81.44, 81.61, 81.79, 81.97, 82.↩
15, 82.33, 82.51, 82.70, 82.88, 83.07, 83.25, 83.44, 83.63, 83.82, 84.01, 84.20, 84.40, 84.↩
59, 84.79, 84.98, 85.18, 85.38, 85.58, 85.79, 85.99, 86.20, 86.40, 86.61, 86.82, 87.03, 87.↩
24, 87.46, 87.67, 87.89, 88.11, 88.33, 88.55, 88.77, 89.00, 89.22, 89.45, 89.68, 89.91, 90.↩
14, 90.38, 90.62, 90.85, 91.09, 91.34, 91.58, 91.82, 92.07, 92.32, 92.57, 92.83, 93.08, 93.↩
34, 93.60, 93.86, 94.13, 94.39, 94.66, 94.93, 95.20, 95.48, 95.76, 96.04, 96.32, 96.61, 96.89,
97.18, 97.48, 97.77, 98.07, 98.37, 98.68, 98.98, 99.29, 99.61, 99.92, 100.24, 100.56, 100.89,
101.22, 101.55, 101.88, 102.22, 102.56, 102.91, 103.26, 103.61, 103.97, 104.33, 104.70, 105.↩
07, 105.44, 105.82, 106.20, 106.58, 106.98, 107.37, 107.77, 108.18, 108.59, 109.00, 109.42,
109.85, 110.28, 110.71, 111.16, 111.60, 112.06, 112.52, 112.99, 113.46, 113.94, 114.43, 114.↩
92, 115.42, 115.93, 116.44, 116.97, 117.50, 118.04, 118.59, 119.14, 119.71, 120.29, 120.87,
121.47, 122.07, 122.69, 123.32, 123.95, 124.61, 125.27, 125.94, 126.63, 127.33, 128.05, 128.↩
78, 129.52, 130.29, 131.06, 131.86, 132.67, 133.50, 134.35, 135.22, 136.12, 137.03, 137.97,
138.93, 139.92, 140.93, 141.97, 143.05, 144.15, 145.29, 146.46, 147.67, 148.92, 150.22, 151.↩
55, 152.94, 154.38, 155.87, 157.42, 159.04, 160.72, 162.48, 164.33, 166.26, 168.29, 170.42,
172.68, 175.07, 177.60, 180.30, 183.19, 186.29, 189.63, 193.26, 197.23, 201.59, 206.43, 211.↩
86, 218.04, 225.19, 233.64, 243.93, 257.02, 274.82, 302.24, 360.00, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
```

```
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98}
```

4.33.3.5 tempMotorLUT

```
const float tempMotorLUT[] = {-2.45, -2.44, -2.44, -2.43, -2.42, -2.42, -2.41, -2.41, -2.↩
40, -2.39, -2.39, -2.38, -2.37, -2.37, -2.36, -2.36, -2.35, -2.34, -2.34, -2.33, -2.32, -2.↩
32, -2.31, -2.31, -2.30, -2.29, -2.29, -2.28, -2.27, -2.27, -2.26, -2.26, -2.25, -2.24, -2.↩
24, -2.23, -2.22, -2.22, -2.21, -2.20, -2.20, -2.19, -2.19, -2.18, -2.17, -2.17, -2.16, -2.↩
15, -2.15, -2.14, -2.14, -2.13, -2.12, -2.12, -2.11, -2.10, -2.10, -2.09, -2.08, -2.08, -2.↩
07, -2.07, -2.06, -2.05, -2.05, -2.04, -2.03, -2.03, -2.02, -2.01, -2.01, -2.00, -2.00, -1.↩
99, -1.98, -1.98, -1.97, -1.96, -1.96, -1.95, -1.94, -1.94, -1.93, -1.93, -1.92, -1.91, -1.↩
91, -1.90, -1.89, -1.89, -1.88, -1.87, -1.87, -1.86, -1.86, -1.85, -1.84, -1.84, -1.83, -1.↩
82, -1.82, -1.81, -1.80, -1.80, -1.79, -1.78, -1.78, -1.77, -1.77, -1.76, -1.75, -1.75, -1.↩
74, -1.73, -1.73, -1.72, -1.71, -1.71, -1.70, -1.69, -1.69, -1.68, -1.67, -1.67, -1.66, -1.↩
66, -1.65, -1.64, -1.64, -1.63, -1.62, -1.62, -1.61, -1.60, -1.60, -1.59, -1.58, -1.58, -1.↩
57, -1.56, -1.56, -1.55, -1.54, -1.54, -1.53, -1.53, -1.52, -1.51, -1.51, -1.50, -1.49, -1.↩
49, -1.48, -1.47, -1.47, -1.46, -1.45, -1.45, -1.44, -1.43, -1.43, -1.42, -1.41, -1.41, -1.↩
40, -1.39, -1.39, -1.38, -1.37, -1.37, -1.36, -1.36, -1.35, -1.34, -1.34, -1.33, -1.32, -1.↩
32, -1.31, -1.30, -1.30, -1.29, -1.28, -1.28, -1.27, -1.26, -1.26, -1.25, -1.24, -1.24, -1.↩
23, -1.22, -1.22, -1.21, -1.20, -1.20, -1.19, -1.18, -1.18, -1.17, -1.16, -1.16, -1.15, -1.↩
14, -1.14, -1.13, -1.12, -1.12, -1.11, -1.10, -1.10, -1.09, -1.08, -1.08, -1.07, -1.06, -1.↩
06, -1.05, -1.04, -1.04, -1.03, -1.02, -1.02, -1.01, -1.00, -1.00, -0.99, -0.98, -0.98, -0.↩
97, -0.96, -0.96, -0.95, -0.94, -0.94, -0.93, -0.92, -0.92, -0.91, -0.90, -0.90, -0.89, -0.↩
88, -0.88, -0.87, -0.86, -0.86, -0.85, -0.84, -0.84, -0.83, -0.82, -0.82, -0.81, -0.80, -0.↩
80, -0.79, -0.78, -0.78, -0.77, -0.76, -0.76, -0.75, -0.74, -0.73, -0.73, -0.72, -0.71, -0.↩
71, -0.70, -0.69, -0.69, -0.68, -0.67, -0.67, -0.66, -0.65, -0.65, -0.64, -0.63, -0.63, -0.↩
62, -0.61, -0.61, -0.60, -0.59, -0.59, -0.58, -0.57, -0.56, -0.56, -0.55, -0.54, -0.54, -0.↩
53, -0.52, -0.52, -0.51, -0.50, -0.50, -0.49, -0.48, -0.48, -0.47, -0.46, -0.46, -0.45, -0.↩
44, -0.43, -0.43, -0.42, -0.41, -0.41, -0.40, -0.39, -0.39, -0.38, -0.37, -0.37, -0.36, -0.↩
35, -0.35, -0.34, -0.33, -0.32, -0.32, -0.31, -0.30, -0.30, -0.29, -0.28, -0.28, -0.27, -0.↩
26, -0.26, -0.25, -0.24, -0.23, -0.23, -0.22, -0.21, -0.21, -0.20, -0.19, -0.19, -0.18, -0.↩
17, -0.17, -0.16, -0.15, -0.14, -0.14, -0.13, -0.12, -0.12, -0.11, -0.10, -0.10, -0.09, -0.08,
-0.07, -0.07, -0.06, -0.05, -0.05, -0.04, -0.03, -0.03, -0.02, -0.01, -0.00, 0.00, 0.01, 0.02,
0.02, 0.03, 0.04, 0.04, 0.05, 0.06, 0.07, 0.07, 0.08, 0.09, 0.09, 0.10, 0.11, 0.12, 0.12, 0.↩
13, 0.14, 0.14, 0.15, 0.16, 0.16, 0.17, 0.18, 0.19, 0.19, 0.20, 0.21, 0.21, 0.22, 0.23, 0.24,
0.24, 0.25, 0.26, 0.26, 0.27, 0.28, 0.29, 0.29, 0.30, 0.31, 0.31, 0.32, 0.33, 0.34, 0.34, 0.↩
35, 0.36, 0.36, 0.37, 0.38, 0.39, 0.39, 0.40, 0.41, 0.41, 0.42, 0.43, 0.44, 0.44, 0.45, 0.46,
0.46, 0.47, 0.48, 0.49, 0.49, 0.50, 0.51, 0.51, 0.52, 0.53, 0.54, 0.54, 0.55, 0.56, 0.56, 0.↩
57, 0.58, 0.59, 0.59, 0.60, 0.61, 0.61, 0.62, 0.63, 0.64, 0.64, 0.65, 0.66, 0.67, 0.67, 0.68,
```

```
0.69, 0.69, 0.70, 0.71, 0.72, 0.72, 0.73, 0.74, 0.75, 0.75, 0.76, 0.77, 0.77, 0.78, 0.79, 0.↩
80, 0.80, 0.81, 0.82, 0.83, 0.83, 0.84, 0.85, 0.85, 0.86, 0.87, 0.88, 0.88, 0.89, 0.90, 0.91,
0.91, 0.92, 0.93, 0.94, 0.94, 0.95, 0.96, 0.96, 0.97, 0.98, 0.99, 0.99, 1.00, 1.01, 1.02, 1.↩
02, 1.03, 1.04, 1.05, 1.05, 1.06, 1.07, 1.08, 1.08, 1.09, 1.10, 1.10, 1.11, 1.12, 1.13, 1.13,
1.14, 1.15, 1.16, 1.16, 1.17, 1.18, 1.19, 1.19, 1.20, 1.21, 1.22, 1.22, 1.23, 1.24, 1.25, 1.↩
25, 1.26, 1.27, 1.28, 1.28, 1.29, 1.30, 1.31, 1.31, 1.32, 1.33, 1.34, 1.34, 1.35, 1.36, 1.37,
1.37, 1.38, 1.39, 1.40, 1.40, 1.41, 1.42, 1.43, 1.43, 1.44, 1.45, 1.46, 1.46, 1.47, 1.48, 1.↩
49, 1.49, 1.50, 1.51, 1.52, 1.52, 1.53, 1.54, 1.55, 1.55, 1.56, 1.57, 1.58, 1.58, 1.59, 1.60,
1.61, 1.61, 1.62, 1.63, 1.64, 1.64, 1.65, 1.66, 1.67, 1.67, 1.68, 1.69, 1.70, 1.71, 1.71, 1.↩
72, 1.73, 1.74, 1.74, 1.75, 1.76, 1.77, 1.77, 1.78, 1.79, 1.80, 1.80, 1.81, 1.82, 1.83, 1.84,
1.84, 1.85, 1.86, 1.87, 1.87, 1.88, 1.89, 1.90, 1.90, 1.91, 1.92, 1.93, 1.93, 1.94, 1.95, 1.↩
96, 1.97, 1.97, 1.98, 1.99, 2.00, 2.00, 2.01, 2.02, 2.03, 2.04, 2.04, 2.05, 2.06, 2.07, 2.07,
2.08, 2.09, 2.10, 2.10, 2.11, 2.12, 2.13, 2.14, 2.14, 2.15, 2.16, 2.17, 2.17, 2.18, 2.19, 2.↩
20, 2.21, 2.21, 2.22, 2.23, 2.24, 2.25, 2.25, 2.26, 2.27, 2.28, 2.28, 2.29, 2.30, 2.31, 2.32,
2.32, 2.33, 2.34, 2.35, 2.35, 2.36, 2.37, 2.38, 2.39, 2.39, 2.40, 2.41, 2.42, 2.43, 2.43, 2.↩
44, 2.45, 2.46, 2.46, 2.47, 2.48, 2.49, 2.50, 2.50, 2.51, 2.52, 2.53, 2.54, 2.54, 2.55, 2.56,
2.57, 2.58, 2.58, 2.59, 2.60, 2.61, 2.62, 2.62, 2.63, 2.64, 2.65, 2.66, 2.66, 2.67, 2.68, 2.↩
69, 2.70, 2.70, 2.71, 2.72, 2.73, 2.74, 2.74, 2.75, 2.76, 2.77, 2.78, 2.78, 2.79, 2.80, 2.81,
2.82, 2.82, 2.83, 2.84, 2.85, 2.86, 2.86, 2.87, 2.88, 2.89, 2.90, 2.90, 2.91, 2.92, 2.93, 2.↩
94, 2.94, 2.95, 2.96, 2.97, 2.98, 2.98, 2.99, 3.00, 3.01, 3.02, 3.02, 3.03, 3.04, 3.05, 3.06,
3.07, 3.07, 3.08, 3.09, 3.10, 3.11, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.16, 3.17, 3.18, 3.↩
19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31,
3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.↩
44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.56, 3.57,
3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.↩
70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83,
3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.↩
96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09,
4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.↩
23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35,
4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.↩
49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62,
4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.↩
76, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.86, 4.87, 4.88, 4.89, 4.90,
4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.↩
04, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17,
5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.↩
32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45,
5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.↩
60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74,
5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.↩
88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6.00, 6.01, 6.01, 6.02,
6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.↩
17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32,
6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.↩
47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61,
6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.↩
76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91,
6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.↩
07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21,
7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.↩
37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52,
7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.↩
68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83,
7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.↩
00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15,
8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.↩
32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47,
```

```
8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.↩
64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80,
8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.↩
97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13,
9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.↩
30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47,
9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.↩
64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81,
9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.↩
99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.↩
13, 10.14, 10.16, 10.17, 10.18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.↩
28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.↩
43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.54, 10.55, 10.57, 10.↩
58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.↩
73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.↩
88, 10.89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.↩
03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.↩
18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.↩
34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.↩
49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.64, 11.↩
65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.↩
81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.↩
97, 11.98, 11.99, 12.00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.↩
13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.↩
29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.40, 12.41, 12.43, 12.44, 12.↩
45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.↩
61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.↩
78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.↩
94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.↩
11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.↩
28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.↩
45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.59, 13.61, 13.↩
62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.↩
79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.↩
97, 13.98, 13.99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.↩
14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.↩
32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.↩
50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.↩
67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.↩
86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.↩
04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.↩
22, 15.24, 15.25, 15.26, 15.28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.↩
41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.↩
59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.74, 15.75, 15.77, 15.↩
78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.↩
97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.↩
16, 16.18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.↩
35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.↩
55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.↩
75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.↩
94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.13, 17.↩
14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.↩
35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.↩
55, 17.56, 17.58, 17.60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.↩
75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.↩
96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.11, 18.12, 18.14, 18.15, 18.↩
17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.↩
38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.↩
```

```
59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.↩
81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.↩
02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.↩
24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.↩
46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.65, 19.67, 19.↩
69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.↩
91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.↩
14, 20.16, 20.18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.↩
37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.↩
60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.↩
84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.↩
07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.↩
31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.↩
55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.↩
80, 21.82, 21.84, 21.86, 21.87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.↩
05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.↩
30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.49, 22.51, 22.53, 22.↩
55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.↩
80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.↩
06, 23.08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.↩
32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.↩
58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.↩
85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.↩
12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.37, 24.↩
39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.↩
67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.↩
95, 24.97, 24.99, 25.01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.↩
23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.↩
52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.72, 25.74, 25.76, 25.78, 25.↩
81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.↩
10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.↩
39, 26.42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.↩
69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.↩
00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.↩
31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.↩
62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.89, 27.91, 27.↩
93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.↩
26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.↩
58, 28.60, 28.63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.↩
91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.↩
24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.↩
58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.↩
92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.↩
27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.↩
62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.↩
98, 31.01, 31.04, 31.07, 31.09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.↩
35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.↩
72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.00, 32.03, 32.06, 32.↩
09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.↩
47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.↩
86, 32.89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.↩
25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.↩
65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.↩
05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.↩
47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.85, 34.↩
89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.↩
31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.↩
75, 35.78, 35.81, 35.85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.↩
```

```
19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.↩
64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.92, 36.95, 36.99, 37.02, 37.06, 37.↩
09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.↩
56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.↩
04, 38.07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.↩
52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.↩
01, 39.05, 39.09, 39.13, 39.17, 39.21, 39.24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.↩
52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.↩
03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.43, 40.47, 40.51, 40.↩
55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.↩
09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.↩
64, 41.68, 41.72, 41.76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.↩
19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.↩
76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.08, 43.12, 43.17, 43.21, 43.26, 43.30, 43.↩
35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.↩
94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.51, 44.↩
56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.↩
18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.↩
82, 45.87, 45.92, 45.97, 46.02, 46.07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.↩
48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.↩
15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.63, 47.69, 47.74, 47.79, 47.↩
85, 47.90, 47.95, 48.01, 48.06, 48.12, 48.17, 48.23, 48.28, 48.34, 48.39, 48.45, 48.50, 48.↩
56, 48.61, 48.67, 48.72, 48.78, 48.84, 48.89, 48.95, 49.00, 49.06, 49.12, 49.17, 49.23, 49.↩
29, 49.34, 49.40, 49.46, 49.52, 49.57, 49.63, 49.69, 49.75, 49.80, 49.86, 49.92, 49.98, 50.↩
04, 50.10, 50.16, 50.21, 50.27, 50.33, 50.39, 50.45, 50.51, 50.57, 50.63, 50.69, 50.75, 50.↩
81, 50.87, 50.93, 50.99, 51.05, 51.11, 51.17, 51.24, 51.30, 51.36, 51.42, 51.48, 51.54, 51.↩
61, 51.67, 51.73, 51.79, 51.86, 51.92, 51.98, 52.04, 52.11, 52.17, 52.23, 52.30, 52.36, 52.↩
43, 52.49, 52.55, 52.62, 52.68, 52.75, 52.81, 52.88, 52.94, 53.01, 53.07, 53.14, 53.21, 53.↩
27, 53.34, 53.40, 53.47, 53.54, 53.60, 53.67, 53.74, 53.80, 53.87, 53.94, 54.01, 54.08, 54.↩
14, 54.21, 54.28, 54.35, 54.42, 54.49, 54.56, 54.63, 54.70, 54.76, 54.83, 54.90, 54.98, 55.↩
05, 55.12, 55.19, 55.26, 55.33, 55.40, 55.47, 55.54, 55.62, 55.69, 55.76, 55.83, 55.91, 55.↩
98, 56.05, 56.12, 56.20, 56.27, 56.35, 56.42, 56.49, 56.57, 56.64, 56.72, 56.79, 56.87, 56.↩
94, 57.02, 57.10, 57.17, 57.25, 57.32, 57.40, 57.48, 57.56, 57.63, 57.71, 57.79, 57.87, 57.↩
94, 58.02, 58.10, 58.18, 58.26, 58.34, 58.42, 58.50, 58.58, 58.66, 58.74, 58.82, 58.90, 58.↩
98, 59.06, 59.15, 59.23, 59.31, 59.39, 59.48, 59.56, 59.64, 59.72, 59.81, 59.89, 59.98, 60.↩
06, 60.15, 60.23, 60.32, 60.40, 60.49, 60.57, 60.66, 60.75, 60.83, 60.92, 61.01, 61.10, 61.↩
18, 61.27, 61.36, 61.45, 61.54, 61.63, 61.72, 61.81, 61.90, 61.99, 62.08, 62.17, 62.26, 62.↩
35, 62.44, 62.54, 62.63, 62.72, 62.82, 62.91, 63.00, 63.10, 63.19, 63.29, 63.38, 63.48, 63.↩
57, 63.67, 63.76, 63.86, 63.96, 64.06, 64.15, 64.25, 64.35, 64.45, 64.55, 64.65, 64.75, 64.↩
85, 64.95, 65.05, 65.15, 65.25, 65.35, 65.46, 65.56, 65.66, 65.76, 65.87, 65.97, 66.08, 66.↩
18, 66.29, 66.39, 66.50, 66.61, 66.71, 66.82, 66.93, 67.03, 67.14, 67.25, 67.36, 67.47, 67.↩
58, 67.69, 67.80, 67.91, 68.03, 68.14, 68.25, 68.36, 68.48, 68.59, 68.71, 68.82, 68.94, 69.↩
05, 69.17, 69.29, 69.40, 69.52, 69.64, 69.76, 69.88, 70.00, 70.12, 70.24, 70.36, 70.48, 70.↩
60, 70.73, 70.85, 70.97, 71.10, 71.22, 71.35, 71.47, 71.60, 71.73, 71.86, 71.98, 72.11, 72.↩
24, 72.37, 72.50, 72.63, 72.76, 72.90, 73.03, 73.16, 73.30, 73.43, 73.57, 73.70, 73.84, 73.↩
98, 74.11, 74.25, 74.39, 74.53, 74.67, 74.81, 74.95, 75.10, 75.24, 75.38, 75.53, 75.67, 75.↩
82, 75.97, 76.11, 76.26, 76.41, 76.56, 76.71, 76.86, 77.01, 77.17, 77.32, 77.47, 77.63, 77.↩
78, 77.94, 78.10, 78.26, 78.42, 78.58, 78.74, 78.90, 79.06, 79.23, 79.39, 79.55, 79.72, 79.↩
89, 80.06, 80.23, 80.40, 80.57, 80.74, 80.91, 81.08, 81.26, 81.44, 81.61, 81.79, 81.97, 82.↩
15, 82.33, 82.51, 82.70, 82.88, 83.07, 83.25, 83.44, 83.63, 83.82, 84.01, 84.20, 84.40, 84.↩
59, 84.79, 84.98, 85.18, 85.38, 85.58, 85.79, 85.99, 86.20, 86.40, 86.61, 86.82, 87.03, 87.↩
24, 87.46, 87.67, 87.89, 88.11, 88.33, 88.55, 88.77, 89.00, 89.22, 89.45, 89.68, 89.91, 90.↩
14, 90.38, 90.62, 90.85, 91.09, 91.34, 91.58, 91.82, 92.07, 92.32, 92.57, 92.83, 93.08, 93.↩
34, 93.60, 93.86, 94.13, 94.39, 94.66, 94.93, 95.20, 95.48, 95.76, 96.04, 96.32, 96.61, 96.89,
97.18, 97.48, 97.77, 98.07, 98.37, 98.68, 98.98, 99.29, 99.61, 99.92, 100.24, 100.56, 100.89,
101.22, 101.55, 101.88, 102.22, 102.56, 102.91, 103.26, 103.61, 103.97, 104.33, 104.70, 105.↩
07, 105.44, 105.82, 106.20, 106.58, 106.98, 107.37, 107.77, 108.18, 108.59, 109.00, 109.42,
109.85, 110.28, 110.71, 111.16, 111.60, 112.06, 112.52, 112.99, 113.46, 113.94, 114.43, 114.↩
```

```
92, 115.42, 115.93, 116.44, 116.97, 117.50, 118.04, 118.59, 119.14, 119.71, 120.29, 120.87,
121.47, 122.07, 122.69, 123.32, 123.95, 124.61, 125.27, 125.94, 126.63, 127.33, 128.05, 128.↩
78, 129.52, 130.29, 131.06, 131.86, 132.67, 133.50, 134.35, 135.22, 136.12, 137.03, 137.97,
138.93, 139.92, 140.93, 141.97, 143.05, 144.15, 145.29, 146.46, 147.67, 148.92, 150.22, 151.↩
55, 152.94, 154.38, 155.87, 157.42, 159.04, 160.72, 162.48, 164.33, 166.26, 168.29, 170.42,
172.68, 175.07, 177.60, 180.30, 183.19, 186.29, 189.63, 193.26, 197.23, 201.59, 206.43, 211.↩
86, 218.04, 225.19, 233.64, 243.93, 257.02, 274.82, 302.24, 360.00, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.↩
98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98, 421.98,
421.98}
```

4.34 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTOR.c File Reference

Source file for motor parameters.

```
#include "MOTOR.h"
#include <math.h>
```

Include dependency graph for MOTOR.c:

Functions

- void precalculate_motor_constants (MotorParameters *motor)

  Precomputes the constants for a motor and updates the MotorParameters structure.
- int check_motor_parameters (MotorParameters *motor, float Ts)

  Perform a parameter check and correct possible errors.

Variables

- MotorParameters motor_left

  Left motor parameters.
- MotorParameters motor_right

  Right motor parameters.

4.34.1 Detailed Description

Source file for motor parameters.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.34.2 Function Documentation

4.34.2.1 check_motor_parameters()

```
int check_motor_parameters (
            MotorParameters * motor,
            float Ts )
```

Perform a parameter check and correct possible errors.

Parameters

in	*motor*	Pointer to the MotorParameters struct.

Return values

OK	0 if an error occurred, 1 if successful.

Here is the caller graph for this function:

4.34.2.2 precalculate_motor_constants()

```
void precalculate_motor_constants (
            MotorParameters * motor )
```

Precomputes the constants for a motor and updates the MotorParameters structure.

Parameters

motor	[in, out] Pointer to the motor parameters structure

Here is the caller graph for this function:

4.34.3 Variable Documentation

4.34.3.1 motor_left

```
MotorParameters motor_left
```

Initial value:

```
= {
    .Ld = 0.00291F,
    .Lq = 0.00291F,
    .Rs = 1.95F,
    .lambda = 0.13391F,
    .pp = 4,
    .J = 0.00093F,
    .b = 0.632653F,
    .torqueMax = 10.0F,
    .dTorqueMax = 1.0F,
    .speedMax_RPM = 8500.0F,
    .iMax = 60.0F,
    .vDCMax = 450.0F
}
```

Left motor parameters.

4.34.3.2 motor_right

```
MotorParameters motor_right
```

Initial value:

```
= {
    .Ld = 0.00291F,
    .Lq = 0.00291F,
    .Rs = 1.95F,
    .lambda = 0.13391F,
    .pp = 4,
    .J = 0.00093F,
    .b = 0.632653F,
    .torqueMax = 10.0F,
    .dTorqueMax = 1.0F,
    .speedMax_RPM = 8500.0F,
    .iMax = 60.0F,
    .vDCMax = 450.0F
}
```

Right motor parameters.

4.35 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB_IO.c File Reference

This file provides functions for handling GPIOs.

```
#include "PCB_IO.h"
#include "main.h"
```

Include dependency graph for PCB_IO.c:

Functions

- void handle_LED (LED ∗led, uint32_t ms_counter)

  LED handler function.
- void handle_direction (volatile int8_t ∗dir_left, volatile int8_t ∗dir_right)

  Handles the direction of the motors.
- void enable_inverters (volatile bool enableSW_left, volatile bool enableSW_right, volatile bool ∗enable_left, volatile bool ∗enable_right)

  Handles the direction of the motors and enables/disables the inverters.

Variables

- LED led_left = { .port = LED_LEFT_GPIO_Port, .pin = LED_LEFT_Pin, .mode = LED_MODE_OFF }
- LED led_right = { .port = LED_RIGHT_GPIO_Port, .pin = LED_RIGHT_Pin, .mode = LED_MODE_OFF }
- LED ledError = { .port = LED_ERR_GPIO_Port, .pin = LED_ERR_Pin, .mode = LED_MODE_OFF }

4.35.1 Detailed Description

This file provides functions for handling GPIOs.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.35.2 Function Documentation

4.35.2.1 enable_inverters()

```
void enable_inverters (
            volatile bool enableSW_left,
            volatile bool enableSW_right,
            volatile bool * enable_left,
            volatile bool * enable_right )
```

Handles the direction of the motors and enables/disables the inverters.

This function reads the state of the shutdown chain (SC or SDC) and enables/disables the inverters based on that and an external software enable bool.

Parameters

in	*enableSW_left*	The software enable state for the left inverter.
in	*enableSW_right*	The software enable state for the right inverter.
out	*enable_left*	Output parameter for the left inverter's enable state.
out	*enable_right*	Output parameter for the right inverter's enable state.

Here is the caller graph for this function:

4.35.2.2 handle_direction()

```
void handle_direction (
            volatile int8_t * dir_left,
            volatile int8_t * dir_right )
```

Handles the direction of the motors.

This function reads the state of the DIR switch and updates the directions of both the left and right motors. If one motor is set to rotate clockwise (CW), the other one is set to rotate counterclockwise (CCW), and vice versa.

Parameters

dir_left	Pointer to the direction parameter in the left inverter structure.
dir_right	Pointer to the direction parameter in the right inverter structure.

Here is the caller graph for this function:

4.35.2.3 handle_LED()

```
void handle_LED (
            LED * led,
            uint32_t ms_counter )
```

LED handler function.

This function handles the LED blinking modes based on the LED mode and current millisecond counter.

Parameters

led	Pointer to the LED structure.
ms_counter	Current millisecond counter.

Here is the caller graph for this function:

4.35.3 Variable Documentation

4.35.3.1 led_left

```
LED led_left = { .port = LED_LEFT_GPIO_Port, .pin = LED_LEFT_Pin, .mode = LED_MODE_OFF }
```

4.35.3.2 led_right

```
LED led_right = { .port = LED_RIGHT_GPIO_Port, .pin = LED_RIGHT_Pin, .mode = LED_MODE_OFF }
```

4.35.3.3 ledError

```
LED ledError = { .port = LED_ERR_GPIO_Port, .pin = LED_ERR_Pin, .mode = LED_MODE_OFF }
```

4.36 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c File Reference

This file provides functions for controlling PWM output.

`#include "PWM.h"`
Include dependency graph for PWM.c:

Functions

- void enable_PWM (TIM_HandleTypeDef *htim)

  Enable PWM output.
- void disable_PWM (TIM_HandleTypeDef *htim)

  Disable PWM output.
- void update_PWM (TIM_HandleTypeDef *htim, Duties duties)

  Set PWM duty cycles.

4.36.1 Detailed Description

This file provides functions for controlling PWM output.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.36.2 Function Documentation

4.36.2.1 disable_PWM()

```
void disable_PWM (
            TIM_HandleTypeDef * htim )
```

Disable PWM output.

This function disables PWM output for the specified timer.

Parameters

htim	Pointer to the TIM_HandleTypeDef structure.

4.36.2.2 enable_PWM()

```
void enable_PWM (
            TIM_HandleTypeDef * htim )
```

Enable PWM output.

This function enables PWM output for the specified timer.

Parameters

htim	Pointer to the TIM_HandleTypeDef structure.

4.36.2.3 update_PWM()

```
void update_PWM (
            TIM_HandleTypeDef * htim,
            Duties duties )
```

Set PWM duty cycles.

This function sets the duty cycles for the PWM channels.

Parameters

htim	Pointer to the TIM_HandleTypeDef structure.
duties	Duties structure containing duty cycle values.

Here is the caller graph for this function:

4.37 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/↩ REFERENCE.c File Reference

Source file for torque reference handling.

```
#include "REFERENCE.h"
#include <math.h>
```

Include dependency graph for REFERENCE.c:

Functions

- float handle_torqueRef (float torqueRefIn, int8_t direction, float torqueMax, float speedMaxRPM, float speedMeas, volatile pi_struct ∗loopSpeed)

  Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.
- float set_torque_direction (float torqueRefIn, int8_t direction)

  Set torque direction based on inverter direction.
- float saturate_symmetric (float ref, float max)

  Symmetrically saturate a reference value.
- float limit_torque_to_prevent_overspeed (float speedMaxRPM, float speedMeas, float torqueRefIn, volatile pi_struct ∗loopSpeed)

  Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.
- float calculate_derated_current (float temperature, float tempStart, float tempMax, float iMax)

  Calculate derated current based on temperature thresholds. It implements a simple linear derating from tempStart to tempMax.
- float derate_current_reference (float tempMotor, float tempInverter, float iMax)

  Derate the current reference based on both motor and inverter temperatures.

Variables

- float torqueRefIn_left = 0.0F
- float torqueRefIn_right = 0.0F

4.37.1 Detailed Description

Source file for torque reference handling.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.37.2 Function Documentation

4.37.2.1 calculate_derated_current()

```
float calculate_derated_current (
            float temperature,
            float tempStart,
            float tempMax,
            float iMax )
```

Calculate derated current based on temperature thresholds. It implements a simple linear derating from tempStart to tempMax.

Parameters

in	*temperature*	The current temperature.
in	*tempStart*	The temperature at which derating starts.
in	*tempMax*	The temperature at which the current is fully derated to 0.
in	*iMax*	The maximum current.

Returns

The derated current.

Here is the caller graph for this function:

4.37.2.2 derate_current_reference()

```
float derate_current_reference (
            float tempMotor,
```

```
            float tempInverter,
            float iMax )
```

Derate the current reference based on both motor and inverter temperatures.

Parameters

in	*tempMotor*	The motor temperature.
in	*tempInverter*	The inverter temperature.
in	*iMax*	The maximum current.

Returns

The derated current reference.

Here is the call graph for this function:

Here is the caller graph for this function:

4.37.2.3 handle_torqueRef()

```
float handle_torqueRef (
            float torqueRefIn,
            int8_t direction,
            float torqueMax,
            float speedMaxRPM,
            float speedMeas,
            volatile pi_struct * loopSpeed )
```

Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.

Parameters

torqueRefIn	Input reference torque.
direction	Direction of torque (1 for positive torque, -1 for negative torque).
torqueMax	Maximum allowable torque.
speedMaxRPM	Maximum allowable speed in RPM.
speedMeas	Measured speed.
loopSpeed	Speed control loop parameters.

Returns

The output torque after handling direction, saturation, and rate limiting.

Here is the call graph for this function:

Here is the caller graph for this function:

4.37.2.4 limit_torque_to_prevent_overspeed()

```
float limit_torque_to_prevent_overspeed (
            float speedMaxRPM,
            float speedMeas,
            float torqueRefIn,
            volatile pi_struct * loopSpeed )
```

Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.

Parameters

in	*speedMaxRPM*	The maximum speed value in RPM.
in	*speedMeas*	The measured speed value in RPM.
in	*torqueRefIn*	The torque reference value before this saturation.
in	*loopSpeed*	Pointer to the speed PI controller structure.

Returns

torqueRefOut The limited torque reference value after this saturation.

Here is the caller graph for this function:

4.37.2.5 saturate_symmetric()

```
float saturate_symmetric (
            float ref,
            float max )
```

Symmetrically saturate a reference value.

This function symmetrically saturates a reference value based on the maximum allowed value. If the reference value exceeds the maximum allowed value, it is saturated to the maximum value. If the reference value is less than the negative of the maximum allowed value, it is saturated to the negative of the maximum value.

Parameters

in	*ref*	The reference value to saturate.
in	*max*	The maximum allowed value for saturation.

Returns

The saturated reference value.

Here is the caller graph for this function:

4.37.2.6 set_torque_direction()

```
float set_torque_direction (
            float torqueRefIn,
            int8_t direction )
```

Set torque direction based on inverter direction.

This function adjusts the torque reference based on the desired direction. If the motor is set to rotate counterclockwise (CCW), positive torque represents traction, negative is braking. If the motor is set to rotate clockwise (CW), negative torque represents traction, positive is braking.

Parameters

in	*torque↩ RefIn*	The torque reference value to adjust.
in	*direction*	Pointer to the direction of the inverter (1 for CW, -1 for CCW).

Returns

torqueRefOut The adjusted torque reference value.

Here is the caller graph for this function:

4.37.3 Variable Documentation

4.37.3.1 torqueRefIn_left

```
float torqueRefIn_left = 0.0F
```

4.37.3.2 torqueRefIn_right

```
float torqueRefIn_right = 0.0F
```

4.38 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx↩ _it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f7xx_it.h"
```

```
#include "TASKS_1ms.h"
#include "TASKS_CRITICAL.h"
#include "CAN_e-Tech.h"
#include "adc.h"
#include "MEASUREMENTS.h"
```

Include dependency graph for stm32f7xx_it.c:

Functions

- void NMI_Handler (void)

  This function handles Non maskable interrupt.
- void HardFault_Handler (void)

  This function handles Hard fault interrupt.
- void MemManage_Handler (void)

  This function handles Memory management fault.
- void BusFault_Handler (void)

  This function handles Pre-fetch fault, memory access fault.
- void UsageFault_Handler (void)

  This function handles Undefined instruction or illegal state.
- void SVC_Handler (void)

  This function handles System service call via SWI instruction.
- void DebugMon_Handler (void)

  This function handles Debug monitor.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void CAN1_RX0_IRQHandler (void)

  This function handles CAN1 RX0 interrupts.
- void TIM1_UP_TIM10_IRQHandler (void)

  This function handles TIM1 update interrupt and TIM10 global interrupt.
- void TIM6_DAC_IRQHandler (void)

  This function handles TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts.
- void DMA2_Stream0_IRQHandler (void)

  This function handles DMA2 stream0 global interrupt.
- void DMA2_Stream1_IRQHandler (void)

  This function handles DMA2 stream1 global interrupt.
- void DMA2_Stream2_IRQHandler (void)

  This function handles DMA2 stream2 global interrupt.
- void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef *hadc)

Variables

- DMA_HandleTypeDef hdma_adc1
- DMA_HandleTypeDef hdma_adc2
- DMA_HandleTypeDef hdma_adc3
- CAN_HandleTypeDef hcan1
- DAC_HandleTypeDef hdac
- TIM_HandleTypeDef htim1
- TIM_HandleTypeDef htim6

4.38.1 Detailed Description

Interrupt Service Routines.

Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

4.38.2 Function Documentation

4.38.2.1 BusFault_Handler()

```
void BusFault_Handler (
            void  )
```

This function handles Pre-fetch fault, memory access fault.

4.38.2.2 CAN1_RX0_IRQHandler()

```
void CAN1_RX0_IRQHandler (
            void  )
```

This function handles CAN1 RX0 interrupts.

Here is the call graph for this function:

4.38.2.3 DebugMon_Handler()

```
void DebugMon_Handler (
            void  )
```

This function handles Debug monitor.

4.38.2.4 DMA2_Stream0_IRQHandler()

```
void DMA2_Stream0_IRQHandler (
            void  )
```

This function handles DMA2 stream0 global interrupt.

Here is the call graph for this function:

4.38.2.5 DMA2_Stream1_IRQHandler()

```
void DMA2_Stream1_IRQHandler (
            void  )
```

This function handles DMA2 stream1 global interrupt.

Here is the call graph for this function:

4.38.2.6 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
            void  )
```

This function handles DMA2 stream2 global interrupt.

4.38.2.7 HAL_ADC_ConvCpltCallback()

```
void HAL_ADC_ConvCpltCallback (
            ADC_HandleTypeDef * hadc )
```

Here is the call graph for this function:

4.38.2.8 HardFault_Handler()

```
void HardFault_Handler (
            void  )
```

This function handles Hard fault interrupt.

4.38.2.9 MemManage_Handler()

```
void MemManage_Handler (
            void  )
```

This function handles Memory management fault.

4.38.2.10 NMI_Handler()

```
void NMI_Handler (
            void  )
```

This function handles Non maskable interrupt.

4.38.2.11 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

4.38.2.12 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

4.38.2.13 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

4.38.2.14 TIM1_UP_TIM10_IRQHandler()

```
void TIM1_UP_TIM10_IRQHandler (
            void  )
```

This function handles TIM1 update interrupt and TIM10 global interrupt.

4.38.2.15 TIM6_DAC_IRQHandler()

```
void TIM6_DAC_IRQHandler (
            void  )
```

This function handles TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts.

4.38.2.16 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

4.38.3 Variable Documentation

4.38.3.1 hcan1

```
CAN_HandleTypeDef hcan1  [extern]
```

4.38.3.2 hdac

```
DAC_HandleTypeDef hdac  [extern]
```

4.38.3.3 hdma_adc1

```
DMA_HandleTypeDef hdma_adc1  [extern]
```

4.38.3.4 hdma_adc2

```
DMA_HandleTypeDef hdma_adc2  [extern]
```

4.38.3.5 hdma_adc3

```
DMA_HandleTypeDef hdma_adc3  [extern]
```

4.38.3.6 htim1

```
TIM_HandleTypeDef htim1  [extern]
```

4.38.3.7 htim6

```
TIM_HandleTypeDef htim6  [extern]
```

4.39 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_↩ 1ms.c File Reference

This file contains functions to execute tasks every 1ms.

```
#include "TASKS_1ms.h"
#include "PCB_IO.h"
#include "MEASUREMENTS.h"
```

```
#include "REFERENCE.h"
```

Include dependency graph for TASKS_1ms.c:

Functions

- void tasks_1ms (void)

  Function to be executed every 1ms.
- void read_temperatures (void)

  Function to read temperatures and handle overtemperature faults.
- void handle_overtemperature_faults (volatile InverterStruct *inv)

  Function to handle overtemperature faults.

4.39.1 Detailed Description

This file contains functions to execute tasks every 1ms.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

4.39.2 Function Documentation

4.39.2.1 handle_overtemperature_faults()

```
void handle_overtemperature_faults (
            volatile InverterStruct * inv )
```

Function to handle overtemperature faults.

Parameters

in,out	*inv*	Pointer to the InverterStruct structure.

Here is the call graph for this function:

Here is the caller graph for this function:

4.39.2.2 read_temperatures()

```
void read_temperatures (
            void  )
```

Function to read temperatures and handle overtemperature faults.

Here is the call graph for this function:

Here is the caller graph for this function:

4.39.2.3 tasks_1ms()

```
void tasks_1ms (
            void  )
```

Function to be executed every 1ms.

This function is called by the TIM6 IRQ handler every millisecond. It increments the millisecond counter and executes all the low priority tasks. Here is the call graph for this function:

Here is the caller graph for this function:

4.40 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_↩ CRITICAL.c File Reference

This file contains functions executed in each PWM timer interruption.

```
#include "CONTROL.h"
#include "INVERTER.h"
#include <math.h>
```

Include dependency graph for TASKS_CRITICAL.c:

Functions

- void tasks_critical_left (void)

  Function to be executed every TS.
- void tasks_critical_right (void)

  Function to be executed every TS.

Variables

- uint32_t start_ticks = 0
- uint32_t elapsed_ticks = 0
- angle_struct angle_left
- rampa_struct freqRamp_left

4.40.1 Detailed Description

This file contains functions executed in each PWM timer interruption.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: https://creativecommons.org/licenses/by-nc-sa/4.↩0/legalcode

4.40.2 Function Documentation

4.40.2.1 tasks_critical_left()

```
void tasks_critical_left (
            void  )
```

Function to be executed every TS.

This function is called by the TIM1 trigger handler every TS. Here is the call graph for this function:

Here is the caller graph for this function:

4.40.2.2 tasks_critical_right()

```
void tasks_critical_right (
            void  )
```

Function to be executed every TS.

This function is called by the TIM8 trigger handler every TS. Here is the caller graph for this function:

4.40.3 Variable Documentation

4.40.3.1 angle_left

```
angle_struct angle_left
```

Initial value:

```
= {
    .freq = 0.0F,
    .Ts = TS,
}
```

4.40.3.2 elapsed_ticks

```
uint32_t elapsed_ticks = 0
```

4.40.3.3 freqRamp_left

```
rampa_struct freqRamp_left
```

Initial value:

```
= {
        .in = 5.0F,
        .Incr = TS*1000,
}
```

4.40.3.4 start_ticks

```
uint32_t start_ticks = 0
```

Index

A
Encoder, 8
A_L_GPIO_Port
main.h, 47
A_L_Pin
main.h, 47
A_R_GPIO_Port
main.h, 47
A_R_Pin
main.h, 47
Analog, 5
currentOffsets, 5
ia, 5
ib, 5
ic, 6
vDC, 6
analog
InverterStruct, 12
angle_left
TASKS_CRITICAL.c, 162

B
Encoder, 8
b
MotorParameters, 20
B_L_GPIO_Port
main.h, 48
B_L_Pin
main.h, 48
B_R_GPIO_Port
main.h, 48
B_R_Pin
main.h, 48
betaMinusIsc
MotorConstants, 16
BusFault_Handler
stm32f7xx_it.c, 152

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h, 23, 27
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h, 27, 31
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/ERRORS.h, 32, 35
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h, 36, 38
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h, 38, 44
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h, 44, 56
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS, 58, 66
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h, 66, 69
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h, 70, 75
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h, 76, 78
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h, 78, 86
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h, 87, 90
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_CRITICAL, 91, 93
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c, 93
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c, 96
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ERRORS.c, 99
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c, 101
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c, 103
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c, 107
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MEASUREMENTS, 110
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTOR.c, 135
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB_IO.c, 139
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c, 142
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/REFERENCE.c, 144
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx_it.c, 150
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_1ms.c, 156
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_CRITICAL, 160
calc_current_loop
CONTROL.c, 97
CONTROL.h, 29
calc_current_reference
CONTROL.c, 97
CONTROL.h, 29
calc_duties

CONTROL.c, 98
CONTROL.h, 30
calculate_derated_current
REFERENCE.c, 146
REFERENCE.h, 81
calibrate_offsets
MEASUREMENTS.c, 121
MEASUREMENTS.h, 60
CAN1_RX0_IRQHandler
stm32f7xx_it.c, 152
CAN_e-Tech.c
handle_CAN, 94
keepAlive, 96
send_CAN_message, 95
CAN_e-Tech.h
enableCAN, 27
handle_CAN, 25
send_CAN_message, 26
CANMessageInfo, 6
DLC, 6
getSig, 6
ID, 6
IDE, 7
check_motor_parameters
MOTOR.c, 137
MOTOR.h, 68
clear_error
ERRORS.c, 100
ERRORS.h, 34
constants
MotorParameters, 20
CONTROL.c
calc_current_loop, 97
calc_current_reference, 97
calc_duties, 98
saturate_voltage, 98
CONTROL.h
calc_current_loop, 29
calc_current_reference, 29
calc_duties, 30
saturate_voltage, 31
CONTROL_FAULT
ERRORS.h, 34
cosTheta_e
Encoder, 8
CURRENT_OFFSET
MEASUREMENTS.h, 60
CURRENT_SLOPE
MEASUREMENTS.h, 60
currentOffsets
Analog, 5

Da
Duties, 7
DAC_GPIO_Port
main.h, 48
DAC_Pin
main.h, 48
Db
Duties, 7
Dc
Duties, 7
DebugMon_Handler
stm32f7xx_it.c, 152
derate_current_reference
REFERENCE.c, 146
REFERENCE.h, 81
DIR_GPIO_Port
main.h, 48
DIR_Pin
main.h, 48
DIR_STATE
PCB_IO.h, 72
direction
InverterStruct, 12
directionMeas
Encoder, 9
DISABLE
PCB_IO.h, 72
disable_control_loops
INVERTER.c, 105
INVERTER.h, 41
disable_PWM
PWM.c, 143
PWM.h, 77
DLC
CANMessageInfo, 6
DMA2_Stream0_IRQHandler
stm32f7xx_it.c, 153
DMA2_Stream1_IRQHandler
stm32f7xx_it.c, 153
DMA2_Stream2_IRQHandler
stm32f7xx_it.c, 153
DT
INVERTER.h, 40
dTorqueMax
MotorParameters, 20
Duties, 7
Da, 7
Db, 7
Dc, 7
duties
InverterStruct, 12

eightTimesOneMinusXiSquared
MotorConstants, 16
elapsed_ticks
TASKS_CRITICAL.c, 162
ENABLE
PCB_IO.h, 72
enable
InverterStruct, 12
enable_control_loops
INVERTER.c, 105
INVERTER.h, 41
enable_inverters
PCB_IO.c, 140
PCB_IO.h, 73

ENABLE_L_GPIO_Port
main.h, 48
ENABLE_L_Pin
main.h, 48
enable_pin
InverterStruct, 12
enable_port
InverterStruct, 12
enable_PWM
PWM.c, 143
PWM.h, 77
ENABLE_R_GPIO_Port
main.h, 49
ENABLE_R_Pin
main.h, 49
enableCAN
CAN_e-Tech.h, 27
enableSW
InverterStruct, 12
Encoder, 8
A, 8
B, 8
cosTheta_e, 8
directionMeas, 9
sinTheta_e, 9
theta_e, 9
we, 9
Z, 9
encoder
InverterStruct, 12
Error_Handler
main.c, 108
main.h, 55
errors
InverterStruct, 13
ERRORS.c
clear_error, 100
is_error_set, 101
set_error, 101
ERRORS.h
clear_error, 34
CONTROL_FAULT, 34
FEEDBACK_FAULT, 34
InverterError, 33
is_error_set, 34
NONE, 34
OVERCURRENT, 34
OVERCURRENT_TH, 33
OVERSPEED, 34
OVERSPEED_TH, 33
OVERTEMPERATURE_INV, 34
OVERTEMPERATURE_INVERTER_TH, 33
OVERTEMPERATURE_MOT, 34
OVERTEMPERATURE_MOTOR_TH, 33
OVERVOLTAGE, 34
OVERVOLTAGE_TH, 33
POWER_FAULT, 34
set_error, 35
UNDERVOLTAGE, 34
UNDERVOLTAGE_TH, 33
WARNING, 34
eval_inv_FSM
FSM.c, 103
FSM.h, 37

Feedback, 9
idMeas, 10
iqMeas, 10
speedMeas, 10
torqueCalc, 10
feedback
InverterStruct, 13
FEEDBACK_FAULT
ERRORS.h, 34
fourTimesOneMinusXi
MotorConstants, 16
fourTimesOneMinusXiOnePlusXiSquared
MotorConstants, 17
fourTimesOneMinusXiXiSquared
MotorConstants, 17
freqRamp_left
TASKS_CRITICAL.c, 162
FSM.c
eval_inv_FSM, 103
FSM.h
eval_inv_FSM, 37

get_currents_voltage
MEASUREMENTS.c, 121
MEASUREMENTS.h, 61
get_idiq
MEASUREMENTS.c, 122
MEASUREMENTS.h, 62
get_linear
MEASUREMENTS.c, 123
MEASUREMENTS.h, 63
get_temperature
MEASUREMENTS.c, 123
MEASUREMENTS.h, 64
getSig
CANMessageInfo, 6

hadc
InverterStruct, 13
HAL_ADC_ConvCpltCallback
stm32f7xx_it.c, 154
handle_CAN
CAN_e-Tech.c, 94
CAN_e-Tech.h, 25
handle_direction
PCB_IO.c, 140
PCB_IO.h, 73
handle_LED
PCB_IO.c, 141
PCB_IO.h, 74
handle_overtemperature_faults
TASKS_1ms.c, 158

TASKS_1ms.h, 88
handle_torqueRef
REFERENCE.c, 147
REFERENCE.h, 82
HardFault_Handler
stm32f7xx_it.c, 154
hcan1
stm32f7xx_it.c, 155
hdac
stm32f7xx_it.c, 155
hdma_adc1
stm32f7xx_it.c, 156
hdma_adc2
stm32f7xx_it.c, 156
hdma_adc3
stm32f7xx_it.c, 156
htim
InverterStruct, 13
htim1
stm32f7xx_it.c, 156
htim6
stm32f7xx_it.c, 156

ia
Analog, 5
ia_L_GPIO_Port
main.h, 49
ia_L_Pin
main.h, 49
ia_R_GPIO_Port
main.h, 49
ia_R_Pin
main.h, 49
ib
Analog, 5
ib_L_GPIO_Port
main.h, 49
ib_L_Pin
main.h, 49
ib_R_GPIO_Port
main.h, 49
ib_R_Pin
main.h, 49
ic
Analog, 6
ic_L_GPIO_Port
main.h, 50
ic_L_Pin
main.h, 50
ic_R_GPIO_Port
main.h, 50
ic_R_Pin
main.h, 50
ID
CANMessageInfo, 6
IDE
CANMessageInfo, 7
idLoop
InverterStruct, 13
idMeas
Feedback, 10
idRef
Reference, 22
iMax
MotorParameters, 20
init_control_loops
INVERTER.c, 105
INVERTER.h, 41
initialize_inverter
INVERTER.c, 105
INVERTER.h, 42
INV_STATE_FAULT
INVERTER.h, 41
INV_STATE_IDLE
INVERTER.h, 41
INV_STATE_RUNNING
INVERTER.h, 41
INV_STATE_STARTUP
INVERTER.h, 41
INVERTER.c
disable_control_loops, 105
enable_control_loops, 105
init_control_loops, 105
initialize_inverter, 105
inverter_left, 107
inverter_right, 107
INVERTER.h
disable_control_loops, 41
DT, 40
enable_control_loops, 41
init_control_loops, 41
initialize_inverter, 42
INV_STATE_FAULT, 41
INV_STATE_IDLE, 41
INV_STATE_RUNNING, 41
INV_STATE_STARTUP, 41
inverter_left, 43
inverter_right, 43
InverterState, 40
TS, 40
inverter_left
INVERTER.c, 107
INVERTER.h, 43
inverter_right
INVERTER.c, 107
INVERTER.h, 43
InverterError
ERRORS.h, 33
InverterState
INVERTER.h, 40
InverterStruct, 11
analog, 12
direction, 12
duties, 12
enable, 12
enable_pin, 12
enable_port, 12

enableSW, 12
encoder, 12
errors, 13
feedback, 13
hadc, 13
htim, 13
idLoop, 13
iqLoop, 13
led, 13
motor, 13
reference, 14
speedLoop, 14
state, 14
tempInverter, 14
tempMotor, 14
vd, 14
vq, 14
vsMax, 14
invThreePpLambda
MotorConstants, 17
invTorqueBase
MotorConstants, 17
iqLoop
InverterStruct, 13
iqMeas
Feedback, 10
iqRef
Reference, 22
is_error_set
ERRORS.c, 101
ERRORS.h, 34
isc
MotorConstants, 17
isMaxRef
Reference, 22

J
MotorParameters, 20

keepAlive
CAN_e-Tech.c, 96

lambda
MotorParameters, 20
lambdaDivLqMinusLd
MotorConstants, 17
Ld
MotorParameters, 20
LED, 15
mode, 15
pin, 15
port, 15
led
InverterStruct, 13
LED_ERR_GPIO_Port
main.h, 50
LED_ERR_Pin
main.h, 50
led_left
PCB_IO.c, 141
PCB_IO.h, 75
LED_LEFT_GPIO_Port
main.h, 50
LED_LEFT_Pin
main.h, 50
LED_MODE_BLINK_FAST
PCB_IO.h, 73
LED_MODE_BLINK_SLOW
PCB_IO.h, 73
LED_MODE_OFF
PCB_IO.h, 73
LED_MODE_ON
PCB_IO.h, 73
led_right
PCB_IO.c, 141
PCB_IO.h, 75
LED_RIGHT_GPIO_Port
main.h, 50
LED_RIGHT_Pin
main.h, 50
ledError
PCB_IO.c, 142
PCB_IO.h, 75
LEDMode
PCB_IO.h, 72
limit_torque_to_prevent_overspeed
REFERENCE.c, 148
REFERENCE.h, 83
Lq
MotorParameters, 21

main
main.c, 109
main.c
Error_Handler, 108
main, 109
SystemClock_Config, 109
main.h
A_L_GPIO_Port, 47
A_L_Pin, 47
A_R_GPIO_Port, 47
A_R_Pin, 47
B_L_GPIO_Port, 48
B_L_Pin, 48
B_R_GPIO_Port, 48
B_R_Pin, 48
DAC_GPIO_Port, 48
DAC_Pin, 48
DIR_GPIO_Port, 48
DIR_Pin, 48
ENABLE_L_GPIO_Port, 48
ENABLE_L_Pin, 48
ENABLE_R_GPIO_Port, 49
ENABLE_R_Pin, 49
Error_Handler, 55
ia_L_GPIO_Port, 49
ia_L_Pin, 49
ia_R_GPIO_Port, 49

ia_R_Pin, 49
ib_L_GPIO_Port, 49
ib_L_Pin, 49
ib_R_GPIO_Port, 49
ib_R_Pin, 49
ic_L_GPIO_Port, 50
ic_L_Pin, 50
ic_R_GPIO_Port, 50
ic_R_Pin, 50
LED_ERR_GPIO_Port, 50
LED_ERR_Pin, 50
LED_LEFT_GPIO_Port, 50
LED_LEFT_Pin, 50
LED_RIGHT_GPIO_Port, 50
LED_RIGHT_Pin, 50
PWM1_L_GPIO_Port, 51
PWM1_L_Pin, 51
PWM1_R_GPIO_Port, 51
PWM1_R_Pin, 51
PWM2_L_GPIO_Port, 51
PWM2_L_Pin, 51
PWM2_R_GPIO_Port, 51
PWM2_R_Pin, 51
PWM3_L_GPIO_Port, 51
PWM3_L_Pin, 51
PWM3_R_GPIO_Port, 52
PWM3_R_Pin, 52
PWM4_L_GPIO_Port, 52
PWM4_L_Pin, 52
PWM4_R_GPIO_Port, 52
PWM4_R_Pin, 52
PWM5_L_GPIO_Port, 52
PWM5_L_Pin, 52
PWM5_R_GPIO_Port, 52
PWM5_R_Pin, 52
PWM6_L_GPIO_Port, 53
PWM6_L_Pin, 53
PWM6_R_GPIO_Port, 53
PWM6_R_Pin, 53
SC_det_GPIO_Port, 53
SC_det_Pin, 53
Tinv_L_GPIO_Port, 53
Tinv_L_Pin, 53
Tinv_R_GPIO_Port, 53
Tinv_R_Pin, 53
Tmot_L_GPIO_Port, 54
Tmot_L_Pin, 54
Tmot_R_GPIO_Port, 54
Tmot_R_Pin, 54
TRIP_L_GPIO_Port, 54
TRIP_L_Pin, 54
TRIP_R_GPIO_Port, 54
TRIP_R_Pin, 54
VDC_L_GPIO_Port, 54
VDC_L_Pin, 54
VDC_R_GPIO_Port, 55
VDC_R_Pin, 55
WRN_L_GPIO_Port, 55
WRN_L_Pin, 55
WRN_R_GPIO_Port, 55
WRN_R_Pin, 55
Z_L_GPIO_Port, 55
Z_L_Pin, 55
Z_R_GPIO_Port, 55
Z_R_Pin, 55
MEASUREMENTS.c
calibrate_offsets, 121
get_currents_voltage, 121
get_idiq, 122
get_linear, 123
get_temperature, 123
rawADC_left, 124
rawADC_right, 124
rawADC_temp, 124
tempInverterLUT, 124
tempMotorLUT, 130
MEASUREMENTS.h
calibrate_offsets, 60
CURRENT_OFFSET, 60
CURRENT_SLOPE, 60
get_currents_voltage, 61
get_idiq, 62
get_linear, 63
get_temperature, 64
rawADC_left, 65
rawADC_right, 65
rawADC_temp, 65
tempInverterLUT, 65
tempMotorLUT, 65
VOLTAGE_OFFSET, 60
VOLTAGE_SLOPE, 60
MemManage_Handler
stm32f7xx_it.c, 154
mode
LED, 15
motor
InverterStruct, 13
MOTOR.c
check_motor_parameters, 137
motor_left, 138
motor_right, 138
precalculate_motor_constants, 137
MOTOR.h
check_motor_parameters, 68
motor_left, 69
motor_right, 69
precalculate_motor_constants, 68
motor_left
MOTOR.c, 138
MOTOR.h, 69
motor_right
MOTOR.c, 138
MOTOR.h, 69
MotorConstants, 16
betaMinusIsc, 16
eightTimesOneMinusXiSquared, 16

fourTimesOneMinusXi, 16
fourTimesOneMinusXiOnePlusXiSquared, 17
fourTimesOneMinusXiXiSquared, 17
invThreePpLambda, 17
invTorqueBase, 17
isc, 17
lambdaDivLqMinusLd, 17
oneMinusXi, 17
threePpLambda, 17
threePpLdMinusLq, 18
torqueBase, 18
twoMinusXi, 18
twoMinusXiSquared, 18
twoTimesOneMinusXiOnePlusXiSquared, 18
twoTimesOneMinusXiXiSquared, 18
xi, 18
xiSquared, 18
MotorParameters, 19
b, 20
constants, 20
dTorqueMax, 20
iMax, 20
J, 20
lambda, 20
Ld, 20
Lq, 21
pp, 21
Rs, 21
speedMax_RPM, 21
torqueMax, 21
vDCMax, 21

NMI_Handler
stm32f7xx_it.c, 154
NONE
ERRORS.h, 34

oneMinusXi
MotorConstants, 17
OVERCURRENT
ERRORS.h, 34
OVERCURRENT_TH
ERRORS.h, 33
OVERSPEED
ERRORS.h, 34
OVERSPEED_TH
ERRORS.h, 33
OVERTEMPERATURE_INV
ERRORS.h, 34
OVERTEMPERATURE_INVERTER_TH
ERRORS.h, 33
OVERTEMPERATURE_MOT
ERRORS.h, 34
OVERTEMPERATURE_MOTOR_TH
ERRORS.h, 33
OVERVOLTAGE
ERRORS.h, 34
OVERVOLTAGE_TH
ERRORS.h, 33

PCB_IO.c
enable_inverters, 140
handle_direction, 140
handle_LED, 141
led_left, 141
led_right, 141
ledError, 142
PCB_IO.h
DIR_STATE, 72
DISABLE, 72
ENABLE, 72
enable_inverters, 73
handle_direction, 73
handle_LED, 74
led_left, 75
LED_MODE_BLINK_FAST, 73
LED_MODE_BLINK_SLOW, 73
LED_MODE_OFF, 73
LED_MODE_ON, 73
led_right, 75
ledError, 75
LEDMode, 72
SC_DET_STATE, 72
WRN_STATE, 72
PendSV_Handler
stm32f7xx_it.c, 154
pin
LED, 15
port
LED, 15
POWER_FAULT
ERRORS.h, 34
pp
MotorParameters, 21
precalculate_motor_constants
MOTOR.c, 137
MOTOR.h, 68
PWM.c
disable_PWM, 143
enable_PWM, 143
update_PWM, 143
PWM.h
disable_PWM, 77
enable_PWM, 77
update_PWM, 77
PWM1_L_GPIO_Port
main.h, 51
PWM1_L_Pin
main.h, 51
PWM1_R_GPIO_Port
main.h, 51
PWM1_R_Pin
main.h, 51
PWM2_L_GPIO_Port
main.h, 51
PWM2_L_Pin
main.h, 51
PWM2_R_GPIO_Port

main.h, 51
PWM2_R_Pin
main.h, 51
PWM3_L_GPIO_Port
main.h, 51
PWM3_L_Pin
main.h, 51
PWM3_R_GPIO_Port
main.h, 52
PWM3_R_Pin
main.h, 52
PWM4_L_GPIO_Port
main.h, 52
PWM4_L_Pin
main.h, 52
PWM4_R_GPIO_Port
main.h, 52
PWM4_R_Pin
main.h, 52
PWM5_L_GPIO_Port
main.h, 52
PWM5_L_Pin
main.h, 52
PWM5_R_GPIO_Port
main.h, 52
PWM5_R_Pin
main.h, 52
PWM6_L_GPIO_Port
main.h, 53
PWM6_L_Pin
main.h, 53
PWM6_R_GPIO_Port
main.h, 53
PWM6_R_Pin
main.h, 53

rawADC_left
MEASUREMENTS.c, 124
MEASUREMENTS.h, 65
rawADC_right
MEASUREMENTS.c, 124
MEASUREMENTS.h, 65
rawADC_temp
MEASUREMENTS.c, 124
MEASUREMENTS.h, 65
read_temperatures
TASKS_1ms.c, 158
TASKS_1ms.h, 89
Reference, 22
idRef, 22
iqRef, 22
isMaxRef, 22
torqueRef, 22
reference
InverterStruct, 14
REFERENCE.c
calculate_derated_current, 146
derate_current_reference, 146
handle_torqueRef, 147
limit_torque_to_prevent_overspeed, 148
saturate_symmetric, 149
set_torque_direction, 149
torqueRefIn_left, 150
torqueRefIn_right, 150
REFERENCE.h
calculate_derated_current, 81
derate_current_reference, 81
handle_torqueRef, 82
limit_torque_to_prevent_overspeed, 83
saturate_symmetric, 84
set_torque_direction, 85
TEMP_INVERTER_DERATING, 80
TEMP_INVERTER_MAX, 80
TEMP_MOTOR_DERATING, 80
TEMP_MOTOR_MAX, 81
torqueRefIn_left, 86
torqueRefIn_right, 86
Rs
MotorParameters, 21

saturate_symmetric
REFERENCE.c, 149
REFERENCE.h, 84
saturate_voltage
CONTROL.c, 98
CONTROL.h, 31
SC_det_GPIO_Port
main.h, 53
SC_det_Pin
main.h, 53
SC_DET_STATE
PCB_IO.h, 72
send_CAN_message
CAN_e-Tech.c, 95
CAN_e-Tech.h, 26
set_error
ERRORS.c, 101
ERRORS.h, 35
set_torque_direction
REFERENCE.c, 149
REFERENCE.h, 85
sinTheta_e
Encoder, 9
speedLoop
InverterStruct, 14
speedMax_RPM
MotorParameters, 21
speedMeas
Feedback, 10
start_ticks
TASKS_CRITICAL.c, 162
state
InverterStruct, 14
stm32f7xx_it.c
BusFault_Handler, 152
CAN1_RX0_IRQHandler, 152
DebugMon_Handler, 152
DMA2_Stream0_IRQHandler, 153

DMA2_Stream1_IRQHandler, 153
DMA2_Stream2_IRQHandler, 153
HAL_ADC_ConvCpltCallback, 154
HardFault_Handler, 154
hcan1, 155
hdac, 155
hdma_adc1, 156
hdma_adc2, 156
hdma_adc3, 156
htim1, 156
htim6, 156
MemManage_Handler, 154
NMI_Handler, 154
PendSV_Handler, 154
SVC_Handler, 155
SysTick_Handler, 155
TIM1_UP_TIM10_IRQHandler, 155
TIM6_DAC_IRQHandler, 155
UsageFault_Handler, 155
SVC_Handler
stm32f7xx_it.c, 155
SystemClock_Config
main.c, 109
SysTick_Handler
stm32f7xx_it.c, 155

tasks_1ms
TASKS_1ms.c, 159
TASKS_1ms.h, 89
TASKS_1ms.c
handle_overtemperature_faults, 158
read_temperatures, 158
tasks_1ms, 159
TASKS_1ms.h
handle_overtemperature_faults, 88
read_temperatures, 89
tasks_1ms, 89
TASKS_CRITICAL.c
angle_left, 162
elapsed_ticks, 162
freqRamp_left, 162
start_ticks, 162
tasks_critical_left, 161
tasks_critical_right, 162
TASKS_CRITICAL.h
tasks_critical_left, 92
tasks_critical_right, 92
tasks_critical_left
TASKS_CRITICAL.c, 161
TASKS_CRITICAL.h, 92
tasks_critical_right
TASKS_CRITICAL.c, 162
TASKS_CRITICAL.h, 92
TEMP_INVERTER_DERATING
REFERENCE.h, 80
TEMP_INVERTER_MAX
REFERENCE.h, 80
TEMP_MOTOR_DERATING
REFERENCE.h, 80
TEMP_MOTOR_MAX
REFERENCE.h, 81
tempInverter
InverterStruct, 14
tempInverterLUT
MEASUREMENTS.c, 124
MEASUREMENTS.h, 65
tempMotor
InverterStruct, 14
tempMotorLUT
MEASUREMENTS.c, 130
MEASUREMENTS.h, 65
theta_e
Encoder, 9
threePpLambda
MotorConstants, 17
threePpLdMinusLq
MotorConstants, 18
TIM1_UP_TIM10_IRQHandler
stm32f7xx_it.c, 155
TIM6_DAC_IRQHandler
stm32f7xx_it.c, 155
Tinv_L_GPIO_Port
main.h, 53
Tinv_L_Pin
main.h, 53
Tinv_R_GPIO_Port
main.h, 53
Tinv_R_Pin
main.h, 53
Tmot_L_GPIO_Port
main.h, 54
Tmot_L_Pin
main.h, 54
Tmot_R_GPIO_Port
main.h, 54
Tmot_R_Pin
main.h, 54
torqueBase
MotorConstants, 18
torqueCalc
Feedback, 10
torqueMax
MotorParameters, 21
torqueRef
Reference, 22
torqueRefIn_left
REFERENCE.c, 150
REFERENCE.h, 86
torqueRefIn_right
REFERENCE.c, 150
REFERENCE.h, 86
TRIP_L_GPIO_Port
main.h, 54
TRIP_L_Pin
main.h, 54
TRIP_R_GPIO_Port
main.h, 54

TRIP_R_Pin
main.h, 54
TS
INVERTER.h, 40
twoMinusXi
MotorConstants, 18
twoMinusXiSquared
MotorConstants, 18
twoTimesOneMinusXiOnePlusXiSquared
MotorConstants, 18
twoTimesOneMinusXiXiSquared
MotorConstants, 18

UNDERVOLTAGE
ERRORS.h, 34
UNDERVOLTAGE_TH
ERRORS.h, 33
update_PWM
PWM.c, 143
PWM.h, 77
UsageFault_Handler
stm32f7xx_it.c, 155

vd
InverterStruct, 14
vDC
Analog, 6
VDC_L_GPIO_Port
main.h, 54
VDC_L_Pin
main.h, 54
VDC_R_GPIO_Port
main.h, 55
VDC_R_Pin
main.h, 55
vDCMax
MotorParameters, 21
VOLTAGE_OFFSET
MEASUREMENTS.h, 60
VOLTAGE_SLOPE
MEASUREMENTS.h, 60
vq
InverterStruct, 14
vsMax
InverterStruct, 14

WARNING
ERRORS.h, 34
we
Encoder, 9
WRN_L_GPIO_Port
main.h, 55
WRN_L_Pin
main.h, 55
WRN_R_GPIO_Port
main.h, 55
WRN_R_Pin
main.h, 55
WRN_STATE
PCB_IO.h, 72

xi
MotorConstants, 18
xiSquared
MotorConstants, 18

Z
Encoder, 9
Z_L_GPIO_Port
main.h, 55
Z_L_Pin
main.h, 55
Z_R_GPIO_Port
main.h, 55
Z_R_Pin
main.h, 55