Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN SISTEMAS ELECTRÓNICOS AVANZADOS

TRABAJO FIN DE MASTER

DEMOSTRADOR DE CONTROL DE MOTORES BASADO EN UN MICROPROCESADOR RISC-V

Estudiante: Zaballa Orduna, Jaime

Director/Directora: Astarloa Cuéllar, Armando

Curso: 2022-2023

Fecha: Bilbao, 13, julio, 2023

Resumen Laburpena Abstract

RISC-V es una ISA nacida en el entorno académico que ha ido ganando relevancia con el tiempo hasta convertirse en la arquitectura de microcontroladores abierta más consolidada del mercado. Aunque se trata de una tecnología novel, su uso y desarrollo se han disparado en los últimos años debido a que no requiere el pago de licencias para su uso, una gran barrera de entrada para desarrolladores. En este trabajo se estudia el uso de RISC-V multi-core para el control de motores PMSM mediante FOC. Para estudiar esta aplicación, se adapta un demostrador de control de motores existente y se porta a una plataforma multi-core basada en RISC-V.

Palabras Clave: ISA, RISC-V, control de motores.

RISC-V inguru akademikoan sortutako ISA ireki bat da. Denboran zehar garrantzia irabarzi du merkatuko mikrokontroladore arkitektura ireki garrantzitzuena bihurtzeraino. Tenkologia berria bada ere, haren erabilera eta garapena azken urteetan asko hazi da, hain zuzen ere, diseniatzaileentzako lizentzia kosterik ez duelako, enpresa txikientzako oztopo handia izaten dena. Lan honek RISC-V multi-coreren erabilera frogatzen du FOC bidezko PMSM motoreen kontrolerako. Aplikazio hau frogatzeko, motoreak kontrolatzeko erabiltzen den demostradore bat RISC-V multi-core plataforma batera egokituko da.

Gako-hitzak: ISA, RISC-V, motore kontrola.

RISC-V in an ISA born in academia which has gained traction over time to the point of becoming the most consolidated open-source microcontroller architecture in the market. Although it is a novel technology, its use and development have skyrocketed in the last few years due to its lack of licensing fees, a major barrier of entry for developers. Throughout this project, the use of multi-core RISC-V for the control of PMSM motors using FOC is studied. In order to study this aplication, an existing motor control demonstrator is adapted and ported to a multi-core RISC-V based platform.

Keywords: ISA, RISC-V, motor control.

Índice

Lista de figuras

Lista de tablas

Lista de acrónimos

ADC Analog to Digital Converter.

ASIC Application Specific Integrated Circuit.

B2B Back 2 Back demonstrator.

BLDC Brushless Direct Current.

CAN Controller Area Network.

CISC Complex Instruction Set Computing.

CPU Central Processing Unit.

CSI Camera Serial Interface.

CSV Comma Separated Values.

DSI Display Serial Interface.

DTC Direct Torque Control.

FOC Field Oriented Control.

FPGA Field Programmable Gate Array.

FPIOA Field Programmable Input Output Array.

GPIO General Purpose Input Output.

GPR General Purpose Register.

GPU Graphics Processing Unit.

HPC High Performance Computing.

I2C Inter Integrated Circuit.

I2S Inter Integrated Sound.

IoT Internet of Things.

IPC Industrial PC.

ISA Instruction Set Architecture.

ISR Interrupt Service Rutine.

LSB Least Significant Byte.

MBD Model Based Design.

MOSFET Metal-Oxide Semiconductor Field-Effect Transistor.

NMI Non-Maskable Interrupt.

PC Program Counter.

PCB Printed Circuit Board.

PLC Programmable Logic Controller.

PMSM Permanent Magnet Synchonous Motor.

PWM Pulse Width Modulation.

QSPI Quad Serial Peripheral Interface.

RISC Reduced Instruction Set Computing.

RTOS Real-Time Operating System.

SBC Single Board Computer.

SDK Software Development Kit.

SoC System on Chip.

SoM System on Module.

SPI Serial Peripheral Interface.

UART Universal Asynchronous Receiver-Transmitter.

VFD Variable Frequency Drive.

1. Introducción

Este trabajo se realiza en colaboración con IKERLAN S. COOP. como continuación de sus líneas de investigación en plataformas hardware de alto rendimiento. Partiendo de un banco de pruebas de control de motores PMSM implementado previamente por la empresa, se adapta el diseño con el fin de probar la ejecución de un control paralelo de dos motores sobre una plataforma RISC-V multi-core.

Para alcanzar este objetivo, se analiza el estado del arte de plataformas de control de motores basadas en microcontroladores y se realiza una búsqueda de plataformas compatibles con la aplicación que a su vez, cumpla con los requisitos establecidos para el proyecto. Tras escoger la plataforma, se plantea el escenario final sobre el que se realizará la evaluación.

Para implementar el escenario propuesto, en primer lugar se modifica el código existente para adaptarlo a la plataforma escogida. De esta forma, se logra poner en marcha el control del primer motor sobre un único core. Una vez implementado y verificado el funcionamiento del primer motor, se adapta el código para añadir un segundo motor y conseguir la ejecución paralela del control de dos motores. Mediante esta implementación, se evaluará el uso de la arquitectura RISC-V para el control de motores y se valorará el estado de la ISA.

El documento se estructura de la siguiente manera. Esta sección presenta el contexto, los objetivos y el alcance de este proyecto. En el Capítulo 2 se analizan las principales arquitecturas de microcontroladores del mercado actual y su uso en el ámbito del control de motores. El demostrador que consta de un control de motores y se utilizará para evaluar la arquitectura seleccionada, se presenta en el Capítulo 3. Finalmente, en el Capítulo 4 se valida la solución propuesta y el Capítulo 5 concluye este trabajo.

1.1. Contexto

A lo largo de los últimos años, distintos acontecimientos han hecho patente la necesidad de generar competición en el espacio de las arquitecturas de microcontroladores. Desde hace años, la arquitectura indisputada en PC ha sido x86, con leves incursiones por parte de arquitecturas basadas en ARM. Esta arquitectura, propiedad de Intel, únicamente se licencia a AMD, generando un oligopolio en el espacio de la computación personal.

En el espacio embebido, el líder indiscutible es ARM, quien licencia sus arquitecturas a todos los principales fabricantes de microcontroladores. Esta dependencia de licencias de una única empresa, sumado a la creciente necesidad de microcontroladores en sectores estratégicos para la defensa y la economía, ha hecho que ARM se convierta en un jugador clave en varios conflictos geopolíticos.

En la guerra comercial entre China y Estados Unidos, ante la amenaza que supone esta dependencia en una empresa extranjera, y movido por las sanciones comerciales impuestas por Estados Unidos [1], China ha promovido el desarrollo de microcontroladores basados en arquitecturas libres de licencias, como RISC-V [2]. De este modo, diversas empresas como Alibaba han comenzado a desarrollar diseños propios basados en esta arquitectura [3][4].

En un intento similar nace el European Chips Act, una propuesta que quiere aumentar la independencia de Europa de otras regiones en el mercado de los semiconductores. De igual manera, una de las bases de este proyecto europeo es el diseño de procesadores basados en la Instruction Set Architecture (ISA) libre RISC-V [5], con el fin de integrar más partes de la cadena de producción de chips en la región.

Estos eventos, combinados con los elevados costes de licencia que dificultan la introducción de nuevos jugadores en el espacio del diseño de microcontroladores, han proliferado el crecimiento de la ISA RISC-V. Es debido a la emergencia de esta que el estudio de nuevas aplicaciones embebidas basadas en RISC-V es pertinente.

1.2. Objetivos y alcance

1.2.1. Objetivos

En este proyecto se quiere desarrollar un demostrador de control de dos motores sobre una plataforma basada en la ISA RISC-V. Para ello, se parte de una implementación previa de un control FOC de un único motor PMSM sobre una plataforma RISC-V single-core. Los objetivos principales del proyecto son:

1. Portar el control actual a una plataforma multi-core basada en RISC-V y añadir un segundo motor para controlarlos en paralelo.
2. Comparar la implementación desplegada frente a la implementación previa.

1.2.2. Alcance

El alcance del proyecto consiste en la portabilidad del control existente a una plataforma multi-core basada en RISC-V. La plataforma que se selecciona en este proyecto es una placa de desarrollo comercial que cuenta con un microcontrolador "hard", no se contempla diseñar una PCB ni realizar un diseño basado en FPGA.

El objetivo principal del proyecto es lograr que el control simultáneo de ambos motores pueda ejecutarse en la nueva plataforma multi-core seleccionada, manteniendo todas sus funcionalidades y características. Para ello, se deberá realizar un proceso de adaptación del software y de la configuración de la plataforma. Adicionalmente, se quiere analizar la viabilidad de un control de motores sobre una plataforma RISC-V multi-core mediante esta implementación.

El proceso de portabilidad incluirá las siguientes tareas principales:

- Estudio de la implementación actual y de su funcionamiento en la plataforma original.
- Análisis del estado del arte de plataformas de control de motores y plataformas RISC-V.
- Identificación de los requisitos de la nueva plataforma multi-core basada en RISC-V, y selección de la placa de desarrollo adecuada.
- Adaptación del software del demostrador implementado actualmente para que pueda ejecutarse en la nueva plataforma, teniendo en cuenta las especificaciones del microcontrolador escogido.
- Verificación del correcto funcionamiento del nuevo demostrador en la plataforma escogida.

Es importante tener en cuenta que el proyecto no contempla la modificación del control implementado en sí mismo, sino únicamente su portabilidad a la nueva plataforma multi-core sobre la que se añadirá el segundo motor, duplicando el control. Por tanto, se considera que el diseño original ya ha sido validado y cumple con los requisitos de funcionamiento previstos.

2. Estado del arte

En este capítulo se analizan las principales arquitecturas de microcontroladores del mercado en la actualidad y su aplicación en el ámbito del control de motores. Tras realizar dicho análisis, se justifica la elección de arquitectura para la aplicación desplegada.

2.1. Arquitecturas de microcontroladores utilizadas en aplicaciones de control de motores

2.1.1. x86

x86 es una ISA desarrollada por Intel que sigue el modelo Complex Instruction Set Computing (CISC). Esta arquitectura se lanzó por primera vez en 1978 y ha ido evolucionando hasta hoy, siendo la arquitectura dominante en computación personal e industrial desde hace décadas y uno de los mayores jugadores en el segmento workstation/cloud.

Al tratarse de una ISA tipo CISC, x86 tiene un gran número de instrucciones para realizar funciones muy específicas, lo que permite realizar operaciones complejas en pocos ciclos de reloj o incluso de manera atómica. Por el contrario, la necesidad de soportar todas estas instrucciones para realizar una implementación completa de x86 provoca que el diseño de un procesador x86 sea muy complejo y que el área dedicada a la unidad de decodificación de instrucciones en silicio sea considerable, lo que impide utilizar ese espacio tan cotizado para otras funcionalidades. Además, debido a que se trata de una ISA que lleva muchos años en activo y que tiene gran soporte para retrocompatibilidad, ha pasado por muchas versiones y añadidos que se han ido sumando al conjunto de instrucciones. Esta complejidad en la unidad de decodificación derivada del soporte de instrucciones *legacy* ha reavivado en numerosas ocasiones el debate CISC vs. RISC, no sólo desde la perspectiva del área o del rendimiento, sino de la eficiencia energética [6][7][8][9].

Originalmente, la arquitectura tenía una anchura de palabra de 16-bit, pero con el tiempo se desarrolló la versión de 32-bit y AMD la extendió a 64-bit. En la actualidad las versiones de 32 y 64-bit coexisten, pero en la mayoría del software desarrollado para esta ISA el soporte para 32-bit está disminuyendo y terminará por desaparecer. Tanto es así que Intel ha propuesto una modificación a la arquitectura, denominada x86S, que deje de soportar los modos de 16 y 32-bits [10].

A pesar de que en el pasado ha habido otros fabricantes de procesadores para esta arquitectura, actualmente los dos únicos diseñadores de chips basados en x86 son AMD e Intel, quien no licencia el uso de la arquitectura a otros diseñadores.

2.1.1.1. Registros

x86 es una ISA little-endian, lo que implica que el guardado de los datos en memoria empieza por el LSB. En su implementación general para 32-bit, x86 tiene ocho registros de propósito general.

Tabla 1: Registros de propósito general en x86

31 Registros de propósito general 0	
	EAX
	EBX
	ECX
	EDX
	ESI
	EDI
	EBP
	ESP

En las versiones de 64-bit de la arquitectura, los registros de la Tabla 1 tienen un tamaño de palabra de 64-bit, y existen ocho registros adicionales **R8...R15** disponibles. Por motivos de compatibilidad con la arquitectura original de 16-bit, se puede acceder a los 16-bit menos significativos de todos los registros de propósito general.

A pesar de que estos registros son de propósito general, tienen funciones asignadas por defecto, por lo que hay que extremar el cuidado a la hora de utilizarlos para evitar la corrupción de datos. Las funciones para las que se suelen utilizar son:

- **EAX** Acumulador y resultado de operaciones.
- **EBX** Registro base, se utiliza para guardar el offset.
- **ECX** Registro del contador.
- **EDX** Acumulador secundario.
- **ESI** Puntero al origen de la instrucción.
- **EDI** Puntero al destino de la instrucción.
- **ESP** Stack pointer.
- **EBP** Puntero a datos en la pila, indica el offset con respecto al stack pointer del dato al que se quiere acceder.

Además de estos registros, la arquitectura de x86 cuenta con muchas revisiones en las que se han añadido registros e instrucciones, algunos notables, como x87, añaden funcionalidad de aritmética en coma flotante y, aunque actualmente se incluyen en cualquier implementación moderna de x86, originalmente formaban parte de un coprocesador adyacente al principal en el ASIC.

2.1.1.2. x86 para control de motores

Como se puede observar en el apartado anterior, x86 es una ISA compleja, por lo que no se suele emplear en aplicaciones *baremetal*, sino sobre un sistema operativo.

Debido a que se trata de una arquitectura generalmente utilizada en aplicaciones de alto peso computacional pero bajo determinismo (con notables excepciones como RTOSs basados en Linux), no suele utilizarse para el control de motores en entornos industriales, sino como parte del sistema de automatización que orquesta el proceso en el que se integra dicho control de motores.

Estos sistemas, denominados Industrial PC (IPC), se utilizan en aplicaciones de más alto nivel que los PLCs, en las que los requisitos temporales no son tan exigentes, pero se requiere una mayor capacidad de procesamiento o compatibilidad con sistemas PC, como la monitorización de procesos, por ejemplo. En la mayoría de los casos, estos sistemas utilizan el mismo hardware que una computadora personal, pero están diseñados para funcionar en entornos con condiciones adversas, como temperaturas extremas, vibraciones, ruido electromagnético, polvo y partículas en suspensión.

Algunos de los fabricantes que integran CPUs x86 en sus diseños de IPC son:

- Advantech
- Beckhoff
- Siemens

Sin embargo, en un sistema embebido monolítico de control de motores, donde se busque el sistema energética y computacionalmente más óptimo, x86 carece de sentido como arquitectura. Es en aplicaciones complejas con requisitos de comunicaciones de gran ancho de banda entre muchos dispositivos o en aplicaciones que requieran de gran capacidad de computación donde x86 tiene sentido como arquitectura seleccionada. En aplicaciones de control de motores que empleen la metodología de diseño basado en modelos (MBD) y lenguajes de programación de alto nivel, donde la versatilidad del sistema y la facilidad de programación sean más importantes que la optimización energética y de recursos, x86 puede ser la plataforma óptima sobre la que implementar ese sistema.

2.1.2. ARM

ARM es un conjunto de ISAs tipo RISC diseñadas para su uso en distintos entornos. Debido a la versatilidad de sus diseños, a su bajo consumo energético y al hecho de que licencian su propiedad intelectual a muchos fabricantes de microcontroladores, ARM se ha convertido con el tiempo en el líder de mercado del sector embebido. Dentro de su portfolio, ARM dispone de varias familias de arquitecturas, cada una de ellas optimizada para un tipo de aplicación. Las principales familias de productos actualmente son:

Cortex-A: Es la familia de microcontroladores de alto rendimiento de ARM. Esta diseñada para soportar la ejecución de sistemas operativos como Android o Linux, y es la arquitectura dominante bajo la amplia mayoría de los microcontroladores implementados en smartphones.

Cortex-R: Es la familia de microcontroladores diseñada para sistemas de tiempo real. Dentro de esta gama, existen distintas opciones dependiendo de los requisitos energéticos y de procesamiento de la aplicación, desde procesadores pequeños para procesos embebidos con requisitos de tiempo real hasta procesadores con aceleradores hardware para aplicaciones de machine learning. Los casos de uso

para los que ARM recomienda algunos de estos diseños son redes y soluciones de almacenamiento como controladores de SSD o clusters de discos duros.

Dentro de esta familia Cortex existen también varios diseños que se especializan en aplicaciones con requisitos de seguridad funcional, en los que los requisitos temporales pueden ser de importancia vital o de obligado cumplimiento por normativa, como el la industria de automoción entre otras.

Cortex-M: Es la familia con mayor penetración en el espacio embebido, esto se debe entre otras cosas a la amplia gama de soluciones que se ofrecen en esta familia, tanto en términos de funcionalidades como de poder de computación. Partiendo desde su versión más reducida, el Cortex-M0, pasando por el Cortex-M4, que implementa funciones de DSP hardware que aceleran sustancialmente los cálculos en coma flotante hasta el Cortex-M85 que ofrece la opción de añadir una extensión vectorial al procesador, Cortex-M cubre muchos de los casos de uso de aplicaciones embarcadas de la industria.

Dentro de estas familias de microcontroladores, conviven varias ISAs, centrando el foco en la familia Cortex-M, hay cuatro ISAs distintas que conviven dentro de la gama de productos.

Armv6-M: Utilizada en Cortex-M0, M0+ y M1. Esta ISA es la más compacta de las utilizadas en la familia Cortex-M, y la más indicada para aplicaciones que requieran de bajo consumo energético y que no tengan grandes requisitos de rendimiento.

Armv7-M: Utilizada en Cortex-M3, M4 y M7. Añade el conjunto de instrucciones Thumb-2, que combina instrucciones de 16 y 32-bits para incrementar la densidad de código y aumentar el rendimiento.

Armv8-M: Utilizada en Cortex-M23, M33 y M35P. Añade la zona de ejecución segura de código ARM TrustZone, que permite aislar la ejecución del código crítico del resto del código para aumentar la seguridad del producto.

ARMv8.1-M: Utilizada en Cortex-M55 y M85. Esta ISA añade una extensión vectorial denominada Helium, que permite acelerar el procesamiento en aplicaciones que se benefician de instrucciones vectoriales. Esta extensión añade más de 150 instrucciones escalares y vectoriales.

La gama de productos de ARM tiene un modelo de conjuntos, de modo que las gamas superiores incluyen las funcionalidades de las gamas inferiores. Existen productos que incluyen funcionalidades específicas que se encuentran ausentes en otros de gama igual o superior, pero, por norma general, es sencillo portar un desarrollo a una gama superior sin perder compatibilidad de instrucciones.

2.1.2.1. Registros

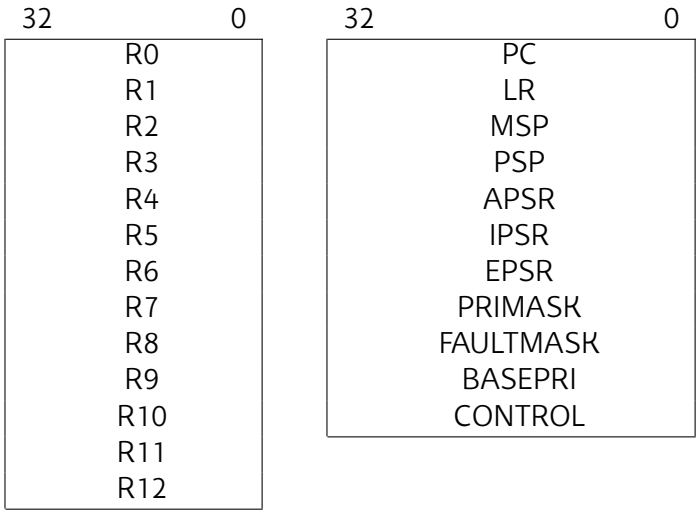
Tomando como ejemplo Armv6-M, los registros de la arquitectura son los siguientes, tal y como se observa en la Tabla 2 [11]:

R0-R12 son registros de propósito general o GPRs, mientras que el resto de los registros tienen un propósito específico:

- **PC:** Program Counter.

- **LR:** Link Register o registro de enlace. Guarda la dirección de regreso cuando se entra en una subrutina.
- **MSP:** Main Stack Pointer, es responsable de guardar primera dirección disponible en la pila.
- **PSP:** Process Stack Pointer, Actúa en el contexto de un proceso, su función es la misma pero es local al proceso que la contiene.
- **APSR:** Application Program Status Register, Almacena los flags de las operaciones realizadas (*carry, overflow, zero...*).
- **IPSR:** Interrupt Program Status Register, indica la fuente de interrupción de la ISR actual.
- **EPSR:** Execution Program Status Register, indica si la instrucción que está siendo ejecutada pertenece a la ampliación *Thumb* del conjunto de instrucciones.
- **PRIMASK:** Priority Mask Register, habilitación global de las fuentes de interrupción.
- **FAULTMASK:** Fault Mask Register, máscara global de interrupciones, si está a '1', enmascara todas las interrupciones salvo las NMI.
- **BASEPRI:** Base Priority Mask Register, define la prioridad mínima que una fuente de interrupción debe tener para que se llame a la ISR correspondiente.
- **CONTROL:** Selecciona la pila activa entre MSP y PSP. Además, permite definir el nivel de privilegio del código ejecutado cuando se está en modo Thread.

Tabla 2: Registros en Armv6-M



2.1.2.2. ARM para control de motores

Los microcontroladores basados en las ISAs de ARM han demostrado ser una elección eficiente y versátil para el control de motores en aplicaciones industriales y comerciales, ofreciendo un rendimiento excepcional y un consumo de energía reducido.

Una de las ventajas de los microcontroladores ARM es que posibilitan integrar una amplia gama de periféricos en un solo chip, lo que simplifica el diseño y reduce los costes de producción. Esta variedad de periféricos permite además encontrar el microcontrolador adecuado para cada aplicación, eliminando la necesidad de pagar por periféricos que no se van a utilizar en el producto final así como la necesidad de complementar las funcionalidades de un microcontrolador con otro que complete los requisitos de la aplicación.

La arquitectura ARM destaca por su eficiencia en el procesamiento de señales digitales y cálculos matemáticos, lo que permite la implementación de algoritmos de control complejos. Esto posibilita a los diseñadores optimizar el rendimiento y la eficiencia de los motores, así como implementar características avanzadas, como el control FOC.

Por otro lado, la arquitectura ARM cuenta con un ecosistema bien establecido y una amplia gama de herramientas de software y compiladores. Esto facilita el desarrollo de aplicaciones de control de motores eléctricos, ya que los diseñadores pueden aprovechar recursos como bibliotecas de controladores, depuradores y simuladores para acelerar el proceso de desarrollo.

Adicionalmente, al existir multitud de fabricantes que utilizan estas ISAs, con distintas especializaciones tecnológicas, es posible encontrar microcontroladores diseñados para aplicaciones muy específicas, en los que el rendimiento puede ser exhaustivamente optimizado.

En conclusión, los microcontroladores ARM ofrecen una solución solvente para el control de motores eléctricos, por lo que su presencia está muy extendida en la industria.

2.1.3. RISC-V

RISC-V (por sus siglas Reduced Instruction Set Architecture - V (five)) es una arquitectura de conjunto de instrucciones (ISA) libre y abierta que cualquier fabricante de microcontroladores puede emplear libremente en sus diseños.

2.1.3.1. Origen

RISC-V nació en 2010 como parte de un proyecto educativo en la universidad de Berkeley cuyo interés primordial era el desarrollo de sistemas de procesamiento paralelo. En su primera versión [12], RISC-V se define como una ISA abierta y adecuada para su implementación directa en hardware, independientemente de la tecnología de implementación (ASIC, FPGA).

Esta primera versión de RISC-V es heredera de proyectos anteriores desarrollados en la propia universidad de Berkeley que definieron otros conjuntos de instrucciones siguiendo el modelo Reduced Instruction Set Computing (RISC), consiguiendo reducir el área de la unidad de decodificación en silicio en comparación con procesadores Complex

Instruction Set Computing (CISC) construidos en procesos equivalentes. Estos cuatro proyectos que la preceden son RISC-I, RISC-II, SOAR y SPUR.

Desde 2015, la especificación de RISC-V es gestionada por la RISC-V foundation, una organización sin ánimo de lucro con sede en Suiza que cuenta con miembros como Google, AMD Xilinx, Qualcomm o Intel entre otros líderes de la industria.

En la actualidad, RISC-V se está empezando a utilizar en muchos productos como los controladores de discos duros Seagate o los nuevos microcontroladores de EspressIf. En el futuro, la consultora Semico Research Corp. proyecta que el crecimiento de RISC-V progrese hasta los 62.4 mil millones de cores RISC-V desplegados en el mercado en 2025 [13].

2.1.3.2. Conjuntos de instrucciones

Uno de los focos de RISC-V es la escalabilidad, ya que es una ISA que destaca por ser modular y adaptable a distintos casos de uso y aplicaciones, desde equipos de bajo consumo hasta sistemas pensados para High Performance Computing (HPC). La especificación de RISC-V define varios conjuntos de instrucciones base y diversas extensiones aplicables a estos para cubrir así el mayor número de aplicaciones posible. A diferencia de otras ISAs tradicionales, RISC-V no se concibe como un conjunto de instrucciones monolítico que va cambiando a lo largo de las versiones según se añaden instrucciones, sino como un conjunto de módulos inmutables que permiten extender la funcionalidad de la arquitectura a medida que se crean nuevos módulos, manteniendo compatibilidad con los anteriores. Cuando se habla de RISC-V como una especificación, se tiene que entender que esta no es única, sino que está, a la fecha de redacción de este documento, compuesta por lo que efectivamente son cuatro ISAs básicas diferentes.

- **RV32I**: Set de instrucciones para números enteros con anchura de 32-bits.
- **RV64I**: Set de instrucciones para números enteros con anchura de 64-bits.
- **RV128I**: Set de instrucciones para números enteros con anchura de 128-bits.
- **RV32E**: Set de instrucciones para números enteros con anchura de 32-bits. Se diferencia de RV32I en que el número de registros es 16, la mitad que RV32I, está pensado para sistemas embebidos de menor tamaño y capacidad.

Estos conjuntos de instrucciones básicos incluyen operaciones aritméticas entre enteros, pero no incluyen operaciones de multiplicación o división entre enteros. Para añadir funcionalidades extra, existen las extensiones. Las principales extensiones contempladas en la especificación de RISC-V son:

- **M:** Añade soporte para operaciones de multiplicación y división con enteros.
- **A:** Añade soporte para operaciones atómicas, es decir, operaciones que se ejecutan en un único ciclo de reloj.
- **F:** Añade soporte para números en coma flotante de precisión simple, además de registros de coma flotante.
- **D:** Añade soporte para números en coma flotante de precisión doble.
- **C:** Añade operaciones comprimidas, es decir, versiones de las operaciones habituales en tamaños de palabra de 16-bits.

2.1.3.3. Registros

A diferencia de otras ISAs como las de ARM, RISC-V no dispone de registros dedicados a una función específica, sino de registros de propósito general. Los únicos registros que tienen un propósito específico son el contador de programa PC y el registro *x0*, que siempre está a cero (Tabla 3). Este último registro se utiliza, entre otras cosas, para implementar la instrucción **NOP**, la cual se implementaría a bajo nivel de la siguiente manera:

```
ADDI x0, x0, 0 ;Sumar al registro x0 el valor inmediato 0
               ;y guardar el resultado en x0
```

Esta instrucción solamente incrementa el PC sin hacer nada más, se podría haber implementado de diversas maneras, pero RISC-V eligió esta debido a que es compatible con cualquier ISA base. A pesar de no tener registros de propósito específico, RISC-V sí que contempla una convención a la hora de seleccionar registros. Esta convención se puede emplear a la hora de diseñar el hardware para optimizar estos registros para la función que desempeñarán con mayor frecuencia. La convención utilizada en **RV32I** es la siguiente:

- **x1:** Registro que contiene la dirección de regreso (Link Register).
- **x5:** Link Register alternativo.
- **x2:** Posición en la pila (Stack Pointer)

2.1.3.4. Modos de privilegio

RISC-V tiene tres modos de ejecución distintos:

1. Modo usuario (User Mode): Es el modo de menor privilegio y está destinado a la ejecución de aplicaciones de usuario estándar. En este modo, el conjunto de instrucciones y las capacidades de acceso a recursos están restringidas para garantizar la seguridad y la protección del sistema.
2. Modo supervisor (Supervisor Mode): El modo de supervisor tiene un nivel más alto de privilegio que el modo de usuario. Permite el acceso a instrucciones y recursos adicionales, como registros de control del sistema, interrupciones y excepciones. Este modo se utiliza principalmente para el sistema operativo y las tareas de administración del sistema.
3. Modo máquina (Machine Mode): Es el modo de privilegio más alto en RISC-V y proporciona acceso completo a todas las instrucciones y recursos del procesador. Este modo está destinado al firmware y al sistema de tiempo de ejecución del hardware. El modo de máquina tiene el control total sobre el sistema y puede acceder a configuraciones y registros especiales del procesador.

2.1.3.5. IP cores

Uno de los motivos por los que RISC-V se ha extendido en los últimos años es la variedad de IP cores para implementaciones soft disponibles para FPGA que diversos

Tabla 3: Registros de RV32I

31 0
x0/zero
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x12
x13
x14
x15
x16
x17
x18
x19
x20
x21
x22
x23
x24
x25
x26
x27
x28
x29
x30
x31
pc

fabricantes han diseñado. Al no requerir pago de licencias para el uso de la ISA, sumado al coste significativamente menor del diseño de cores IP en comparación con el flujo de diseño, fabricación y verificación que supone un ASIC, se ha generado un amplio ecosistema alrededor de RISC-V como IP core en competición con su principal rival en este espacio, ARM.

Dentro de los fabricantes que venden diseños de cores IP de soft CPUs para FPGAs se destacan cuatro debido a su amplia oferta y a su relevancia en el espacio:

- **SiFive:** Es una compañía fundada por los creadores de RISC-V que diseña y comercializa cores IP a medida para distintas aplicaciones, la Tabla 4 resume sus productos al momento de redacción de este documento [14].

Tabla 4: Línea de productos de SiFive

Familia	Nombre del core	ISA	Escalar/ Vectorial	Características
Performance Family	P650/P670	64-bit	Vectorial	four issue, out of order
	P550	RV64GBC	Vectorial	three issue, out of order
	P450/P470	64-bit	Vectorial	out of order
	P270	RV64GBCV	Vectorial	
Automotive Family	E6-A	32-bit	Escalar	ASIL B, ASIL D, eight stage , single issue, superscalar.
	S7-AD	64-bit	Escalar	ASIL D, eight stage, dual issue, superscalar.
	X200-A	RV64GCV	Vectorial	ASIL B, ASIL D, eight stage, dual issue, in-order.
Intelligence Family	X280	RV64GCV	Vectorial	eight stage, dual issue, in-order.
Essential Family	2-Series	32 y 64-bit	Escalar	2-4 stage pipeline, bajo consumo.
	6-Series	32 y 64-bit	Escalar	bajo consumo.
	7-Series	32 y 64-bit	Escalar	eight stage, dual issue, in order.

- **OpenHW group:** Es una organización sin ánimo de lucro que diseña y verifica cores IP basados en RISC-V de código abierto. Actualmente cuenta con nueve cores distintos enfocados a diversas aplicaciones. la Tabla 5 resume sus productos en el momento de redacción de este documento [15].

Tabla 5: Línea de productos de OpenHW group

Clase	Nombre del core	ISA/ISAs	Número de stages del pipeline
Aplicación (Sistema operativo)	CV32A60X	RV32IMCA	5
	CVA5	RV32IMA	5
	CVW	RV32I, RV32E, RV64I +ACDFM	5
Embebido (Bare metal)	CV32E40P	RV32IMFC	4
	CV32E40X	RV32I, RV32E +M	4
	CV32E40S	RV32I, RV32E +M	4
	CV32E41P	RV32IMFC	4
	CVE2	RV32E, RV32I + MC	2

- **Andes Technology:** Es una organización especializada en RISC-V que, además de cores IP, desarrolla stacks software y herramientas de desarrollo para la arquitectura. En términos de soft cores basados en RISC-V, en la Tabla 6 se resumen los productos ofrecidos por la empresa [16].

Tabla 6: Familia de productos de Andes Technology

Familia	Core	ISA	Single-core/ Multi-core	Pipeline
A - Soporte para sistemas operativos	A25	RV32IMAFDC	SC	five stage
	A25MP	RV32IMAFDC	MC	five stage
	AX25	RV64IMAFDC	SC	five stage
	A27	RV32IMAFDC	SC	five stage
	A27L2	RV32IMAFDC	SC	five stage
	AX27	RV64IMAFDC	SC	five stage
	A45	RV32IMAFDC	SC	eight-stage
	A45MP	RV32IMAFDC	MC	eight-stage
	AX45MP	RV64IMAFDC	MC	eight-stage
	AX45MP	RV64IMAFDC	MC	eight-stage
	AX45MPV	RV64IMAFDC	MC	eight-stage
N - Soporte RTOS	N22	RV32IMAFDC	SC	two stage
	N25F	RV32IMAFDC	SC	five stage
	N25F-SE	RV32IMAFDC	SC	five stage
	NX25F	RV64IMAFDC	SC	five stage
	N45	RV32IMAFDC	SC	eight-stage
	NX45	RV64IMAFDC	SC	eight-stage
D - Soporte SIMD	D25F	RV32IMAFDC	SC	five stage
	D23	RV32IMAFDC	SC	three stage
	D45	RV32IMAFDC	SC	eight-stage

- **Intel:** NIOS-V es un soft core para FPGA basado en la ISA RV32IMA, con soporte para RTOSs como FreeRTOS y ZephyrRTOS.

2.1.3.6. RISC-V para control de motores

Las aplicaciones de control de motores son un campo en el que la incursión de RISC-V hasta el momento es mínima. El foco principal de desarrollo de productos basados en RISC-V en los últimos años se ha centrado más en el desarrollo de CPUs de propósito general, en la industria de la automoción, las aplicaciones IoT y HPC.

Sin embargo, recientemente ha surgido un microcontrolador de Renesas basado en el core N22 de Andes technology que se especializa en control de motores. Este microcontrolador, el R9A02G020 (Figura 1), está diseñado para operar con un bajo consumo, por lo que utiliza uno de los cores más pequeños que oferta Andes a una frecuencia máxima de operación de 32MHz y un consumo máximo en operación de 11.82mA. La aplicación de control de motores en la que se especializa es el control, tanto sensorizado como sensorless, de motores BLDC mediante control de campo orientado (FOC).

Para facilitar la integración, Renesas también comercializa una placa de desarrollo basada en este microcontrolador para el control de motores PMSM, BLDC mediante sensorless FOC. Esta placa integra también el módulo de potencia para los motores y una interfaz de conexión para computadores que permite visualizar el comportamiento del motor gráficamente.

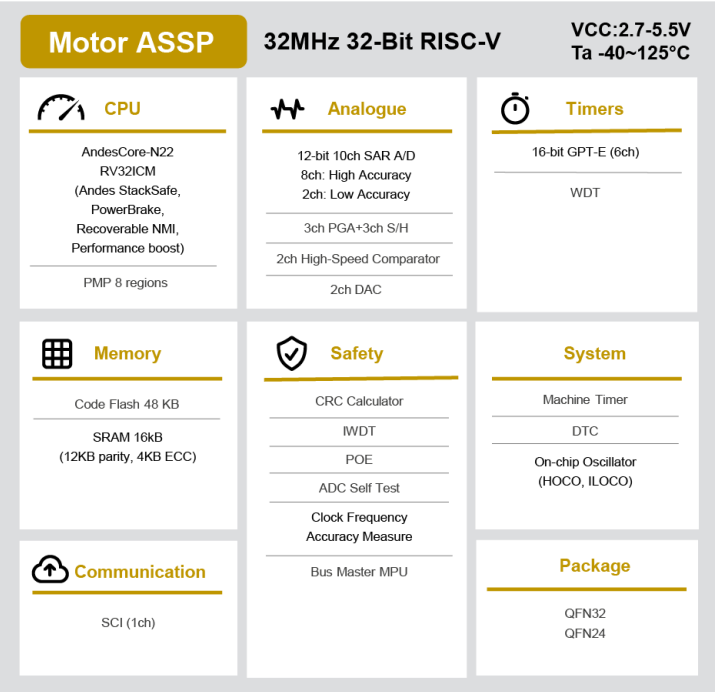


Figura 1: Diagrama conceptual del R9A02G020, obtenido de la página web de Renesas

2.2. Selección de la arquitectura para la aplicación

Tras analizar las distintas arquitecturas de microprocesadores y microcontroladores que se pueden utilizar en el control de motores, se selecciona RISC-V por las siguientes razones:

- **Alineación con las líneas de investigación de IKERLAN:** Este trabajo se realiza en el contexto de un centro de investigación. Dentro de las líneas de investigación que se están trabajando en el momento en el que este trabajo se realiza, RISC-V se plantea como una alternativa o un complemento a plataformas de alto rendimiento para aplicaciones industriales como GPUs y FPGAs. Algún ejemplo de proyectos trabajados en IKERLAN en los que se utiliza RISC-V:

  METASAT Es un proyecto europeo que nace de la necesidad de facilitar el diseño y testeo de módulos software para el sector aeroespacial cumpliendo requisitos de seguridad funcional. Para lograr este objetivo, el proyecto plantea el diseño de un toolchain siguiendo la metodología de diseño basado en modelos (MBD) sobre plataformas informáticas de alto rendimiento basadas en la ISA RISC-V. Sobre este hardware se implementará un hypervisor que permita aislar los distintos módulos software mediante entornos virtualizados.

  SELENE Es un proyecto europeo cuyo objetivo es proponer plataformas de computación safety-critical construidas sobre herramientas de código abierto como RISC-V y Linux. Estos sistemas deberán adaptarse a la aplicación en la que se despliegen mediante el uso de inteligencia artificial, ofreciendo una mayor flexibilidad que alternativas comerciales *off-the-shelf*.

- **RISC-V no se ha explorado en aplicaciones de control de motores:** Como se detalla en la Subsubsección 2.1.3.6, RISC-V apenas ha sido explorada como ISA en sistemas de control de motores eléctricos. Es por ello que, dado que se trata de una

ISA cuya relevancia está en auge, es pertinente estudiar si es apta para este caso de uso tan habitual en la industria, además de comprobar cómo rinde en comparación con otras arquitecturas cuyo uso está más extendido en aplicaciones de control de motores.

- **Relevancia de RISC-V en el ámbito de la investigación:** Existen diversos proyectos de investigación a nivel europeo que estudian y desarrollan aplicaciones basadas en la ISA RISC-V, entre ellos:

  De-RISC Es un proyecto cuyo objetivo es la creación de una plataforma multi-core basada en el estándar RISC-V para aplicaciones del sector aeroespacial con requisitos de seguridad funcional (safety-critical) y de tiempo real mediante el uso de virtualización con hypervisores. Este proyecto pretende ofrecer una alternativa moderna a los sistemas basados en PowerPC o SPARC que tradicionalmente han dominado el sector.

  DRAC Es un proyecto del BSC (Barcelona Supercomputing Center) cuyo objetivo es desarrollar un ASIC de alto rendimiento basado en RISC-V. Las aplicaciones planteadas para este ASIC son la seguridad post-cuántica, la medicina de precisión y la conducción autónoma.

3. Desarrollo del demostrador

Para estudiar la aplicación de control de motores, se parte de un banco de pruebas de control de motores denominado Back to back (B2B). Este demostrador consiste en dos motores enfrentados entre sí y unidos por el eje. Uno de estos motores está controlado para girar a una velocidad fija mientras que el otro ejerce un par fijo, actuando como una carga activa.

Para generar las tensiones de alimentación de los motores se utilizan dos drivers de potencia TI DRV8305EVM; estos drivers realizan la conversión DC/AC de la tensión de alimentación a las tensiones de fase de los motores. Los drivers cuentan además con sensores shunt amplificados en cada fase de corriente y tensión; estas señales se emplean como parámetros de entrada en el control que se explicará más adelante.

En el B2B original (Figura 2), el control se realiza en una placa de desarrollo TI DELFINO LAUNCHPAD, que está basada en un microcontrolador de doble núcleo de Texas Instruments (TMS320F28379D).

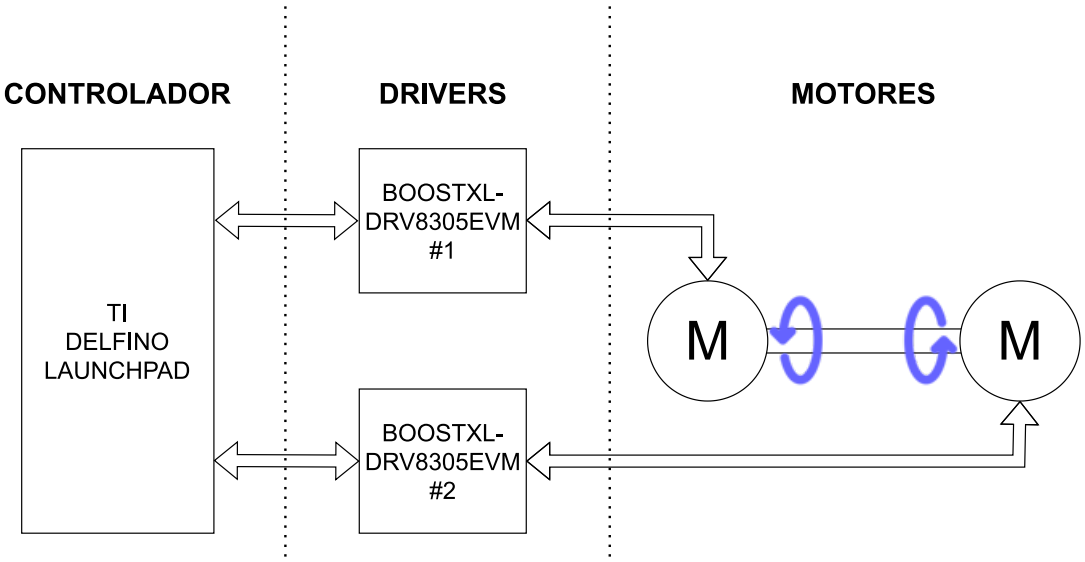


Figura 2: Diagrama conceptual del demostrador B2B

En la implementación anterior a este proyecto, se realiza un B2B parcial basado en RISC-V single-core. En esta implementación parcial se utiliza un único motor controlado en velocidad y un único driver de potencia. El control en este caso se ejecuta en una placa de desarrollo Nuclei RV-STAR, que emplea el microcontrolador GD32VF103VBT6 (Figura 3).

En este proyecto se quiere sustituir la placa de desarrollo single-core utilizada en la implementación anterior por otra multi-core, también basada en RISC-V, en la que se

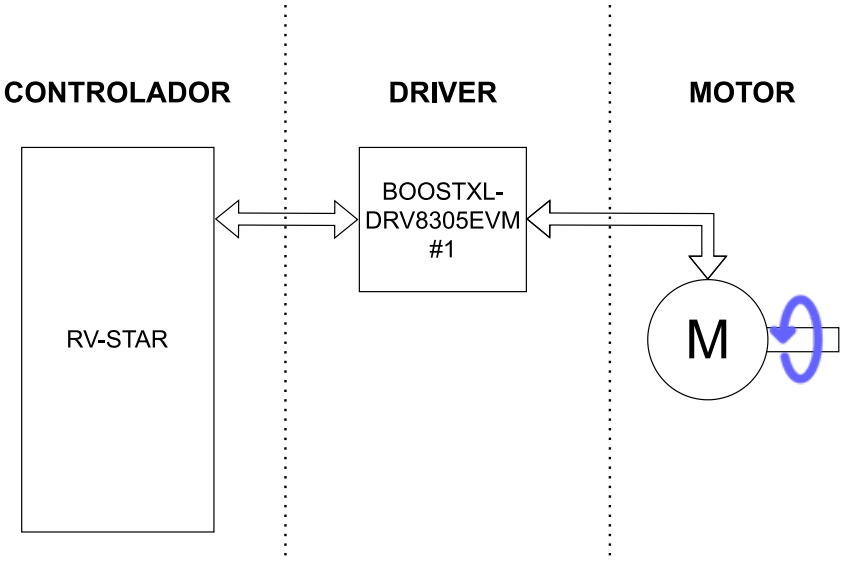


Figura 3: Diagrama conceptual del demostrador B2B parcial

pueda ejecutar simultáneamente el control de dos motores. De esta forma, se quiere evaluar el desempeño de la ISA en aplicaciones multi-core de control de motores. A diferencia del B2B, en la aplicación desplegada en este proyecto (Figura 4), los motores no estarán unidos por el eje, sino que estarán controlados en velocidad por separado pero simultáneamente.

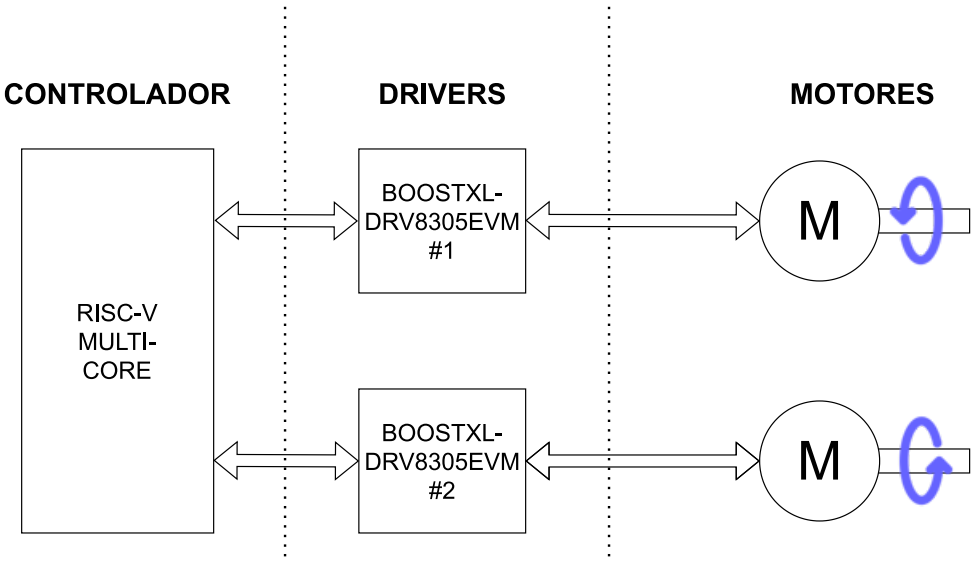


Figura 4: Diagrama conceptual del demostrador implementado en este proyecto

3.1. Motores PMSM

El principio de funcionamiento de cualquier motor eléctrico es la generación de movimiento mediante la interacción entre los campos magnéticos del rotor (parte móvil del motor) y el estator (parte fija del motor). Sin embargo, partiendo de los mismos principios electromagnéticos, existen infinidad de topologías de motor eléctrico.

El demostrador B2B utiliza motores PMSM trifásicos, es decir, motores síncronos de imanes permanentes. Son motores síncronos de corriente alterna (AC) cuyo rotor

está compuesto por imanes, al contrario que en otras topologías de motores AC, como los de inducción, donde los campos magnéticos que generan el movimiento se deben a corrientes inducidas en los devanados del rotor por las corrientes generadas en los devanados del estator (Figura 5).

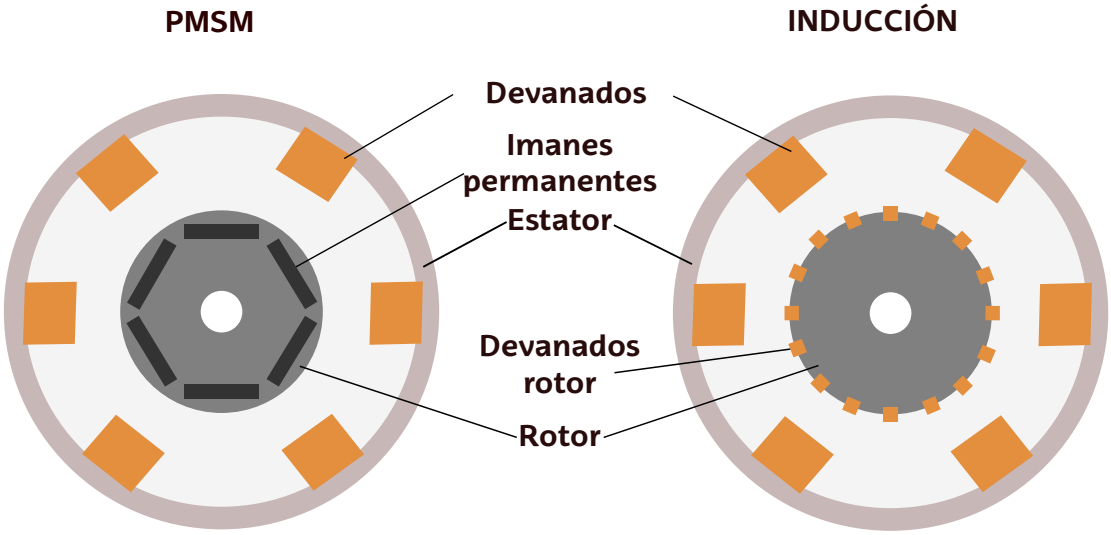


Figura 5: Diferencias físicas entre un motor PMSM y uno de inducción

En el caso de los motores PMSM, al tener un rotor compuesto por imanes permanentes, el campo magnético del rotor es constante; por tanto, para provocar el movimiento, se genera un campo magnético rotativo en los devanados del estator. La velocidad de rotación del motor será, por tanto, síncrona con la frecuencia de giro de este campo magnético.

El motor generará un par que depende del ángulo entre el campo magnético del rotor y el campo magnético del estator. El par será máximo cuando los campos sean ortogonales entre sí ($\vec{B}_r \perp \vec{B}_s$).

Una de las particularidades de estos motores es que, al tener un imán permanente en el rotor en lugar de devanados, se puede generar par a velocidad cero, lo que permite mantener un par resistivo en el motor con un control sencillo. Además, la potencia y el par de estos motores en relación a su tamaño es superior a los motores paso a paso y BLDC; sin embargo, tienen el inconveniente de ser más costosos que estos [17].

Otra característica de estos motores es que requieren de un controlador para funcionar. Al contrario de otras topologías de motor que se benefician de un controlador pero no lo requieren, la naturaleza de los motores PMSM conlleva que un controlador sea necesario para su funcionamiento. Existen distintas estrategias de control para motores PMSM dependiendo de los requisitos de la aplicación en la que se vayan a utilizar, estas son algunas de estas estrategias [18] (Tabla 7):

3.2. Algoritmo de control FOC

El control FOC nace de la necesidad de controlar el torque en motores de corriente alterna. Gracias a los variadores de frecuencia (VFD), es sencillo controlar en velocidad un motor de inducción. Sin embargo, debido a que el torque es dependiente del flujo magnético del rotor, en motores de inducción AC tradicionales es complicado controlarlo (Figura 5). Por ello, en el momento en el que surgen motores PMSM, que al tener un rotor

Tabla 7: Estrategias de control para motores PMSM

<table>
<tr><th colspan="4">Tipo de control</th><th>Ventajas</th><th>Desventajas</th></tr>
<tr><td rowspan="4">Lazo cerrado</td><td colspan="3">Escalar</td><td>- Control sencillo</td><td>- En aplicaciones con carga variable se puede producir pérdida de control.</td></tr>
<tr><td rowspan="3">Vectorial</td><td rowspan="2">Field Oriented Control (FOC)</td><td>Sensorizado</td><td>- Precisión
-Control suave
- Amplio rango de control</td><td>- Requiere un controlador con gran capacidad de procesamiento.
- Requiere sensores de posición del motor.</td></tr>
<tr><td>Sensorless</td><td>- Precisión
-Control suave
- Amplio rango de control</td><td>- Requiere un controlador con gran capacidad de procesamiento.
- Rango de velocidad más reducido que en el caso sensorizado.</td></tr>
<tr><td colspan="2">Control directo del torque (DTC)</td><td>- Buena respuesta dinámica
- Robustez
- Control sencillo</td><td>- Alto rizado de corriente y par.</td></tr>
<tr><td colspan="4">Lazo abierto</td><td>- Control sencillo</td><td>- En aplicaciones con carga variable se puede producir pérdida de control.</td></tr>
</table>

formado por imanes permanentes tienen un campo magnético constante en el rotor, surge este control que permite aprovechar esta característica para realizar un control de torque.

Para el control de los motores que componen el demostrador se utiliza el algoritmo de control FOC sensorizado. Este tipo de control lineal destaca porque permite controlar motores PMSM como si se tratase de motores DC. Como el rotor es un imán permanente, su campo magnético es constante. Teniendo esto en cuenta, se divide el campo magnético del rotor en dos componentes d (flujo) y q (torque) ortogonales entre sí y se transforma el sistema de coordenadas del motor a un marco de referencia invariante en el tiempo que utilice como ejes estas componentes [19]. Al cambiar el marco de referencia, se pueden desacoplar las componentes de flujo y torque para controlarlas por separado, de manera que se puede anular la componente de flujo magnético (d) y controlar el torque (q) directamente [20].

Este cambio de referencia se realiza en varios pasos, se parte considerando las tres corrientes de fase del motor, que están desfasadas $2\pi/3$ rad entre sí. Este sistema de referencia se denomina abc:

$$
\begin{aligned}
I_a &= I_m \cdot cos(\omega \cdot t), \\
I_b &= I_m \cdot cos(\omega \cdot t + 2\pi/3), \\
I_c &= I_m \cdot cos(\omega \cdot t - 2\pi/3)
\end{aligned}
$$

A partir de este sistema de referencia estacionario de tres ejes, se realiza la transformación de Clarke (Figura 7) para convertir el sistema abc en un sistema estacionario de dos ejes $\alpha\beta$ ortogonales entre sí.

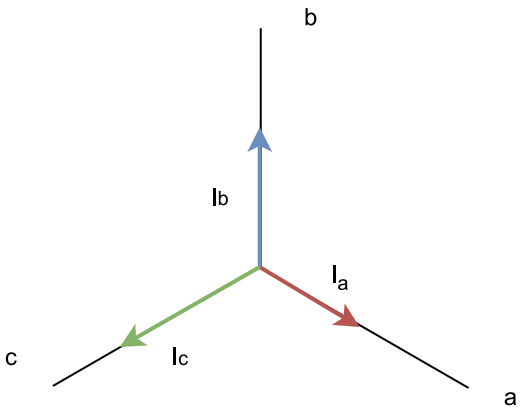


Figura 6: Sistema de referencia estacionario abc

$$I_\alpha = I_a,$$
$$I_\beta = (I_a + 2 \cdot I_b)/\sqrt{3}$$

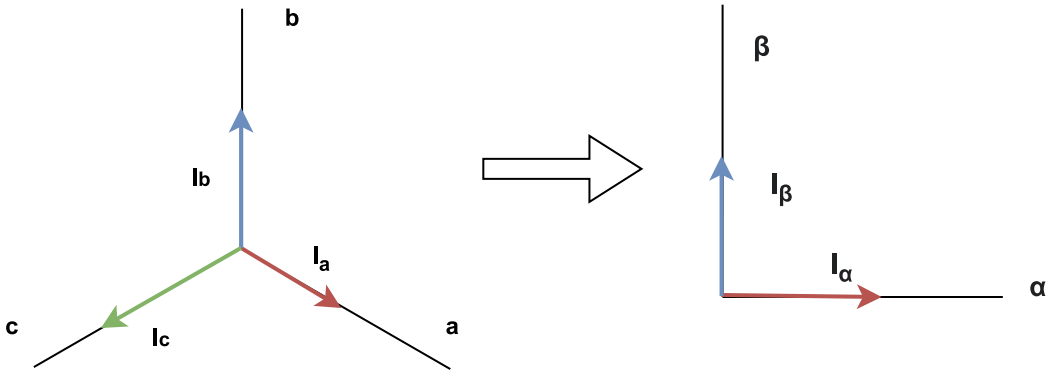


Figura 7: Transformada de Clarke

Esta forma reducida de la transformada de Clarke se puede utilizar cuando se cumpla que: $I_a + I_b + I_c = 0$ [21], es decir, cuando el sistema trifásico sea equilibrado.

A partir de este sistema de referencia estacionario de dos ejes $\alpha\beta$ se emplea la transformación de Park (Figura 8) para convertir el sistema $\alpha\beta$ en un sistema de referencia rotativo de dos ejes dq que rotarán con la posición del rotor (ϕ). Para poder realizar esta transformación, es necesario conocer la posición instantánea del rotor. Aquí es donde se diferencia el algoritmo de control FOC sensorizado del FOC sin sensorizar. En el caso sensorizado, la posición instantánea del rotor se conoce mediante la lectura de un sensor de posición (generalmente un encoder rotativo), mientras que en el caso sin sensorizar, la posición del rotor se estima a partir de las demás variables del control. Esto resulta en un control computacionalmente más complejo, pero también incurre en un menor coste del motor.

$$I_d = I_\alpha \cdot cos(\phi) + I_\beta \cdot sin(\phi),$$
$$I_q = -I_\alpha \cdot sin(\phi) + I_\beta \cdot cos(\phi)$$

Una vez realizada la transformada de Park y obtenidas las componentes I_d, I_q, se aplica un control PI a las corrientes. Para maximizar el giro, se busca que la componente del flujo (I_d) sea nula, de forma que el campo magnético del rotor esté exclusivamente compuesto por la componente del torque. Para obtener la referencia de la corriente del torque (I_{qref}) con la que se alimenta el PI de la corriente I_q, se realiza un PI de velocidad.

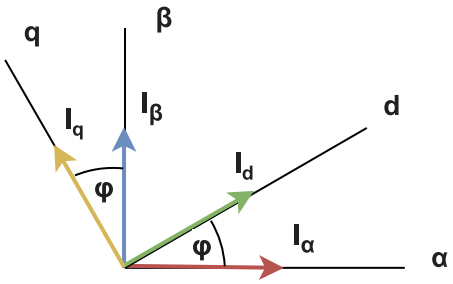


Figura 8: Transformada de Park

El PI de velocidad toma como entradas la velocidad actual del motor y la consigna de velocidad establecida; y da como resultado la referencia de corriente para la componente del torque del motor (I_{qref}). El lazo de velocidad se ejecuta a una frecuencia menor que los de corriente.

Como resultado de los PI de corriente, se obtienen valores de tensión en el plano dq, por lo que hay que realizar las transformaciones inversas a las explicadas previamente para obtener valores de tensión aplicables a las fases del motor. La Figura 9 resume el funcionamiento del control completo.

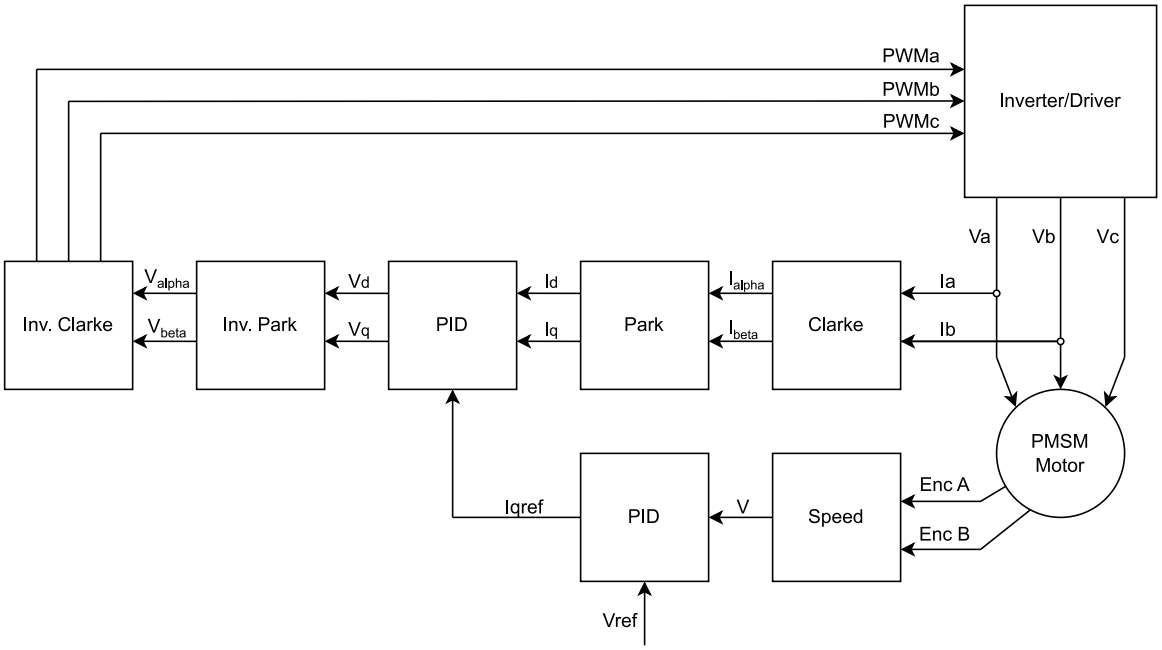


Figura 9: Diagrama esquemático del control FOC utilizado

El paso previo a establecer las tensiones de fase de los motores se realiza en el driver, que realiza la conversión DC/AC a partir de las señales PWM generadas por el control. Los driver utilizados en el demostrador desarrollado cuentan con protecciones contra sobrecorriente y sobretensión, además de sensores de corriente shunt amplificados en cada fase, los cuales se emplean para leer las corrientes de fase que se utilizan en el algoritmo de control FOC. Estos sensores están situados en la parte baja de cada semi-puente de MOSFET como se puede observar en la Figura 10.

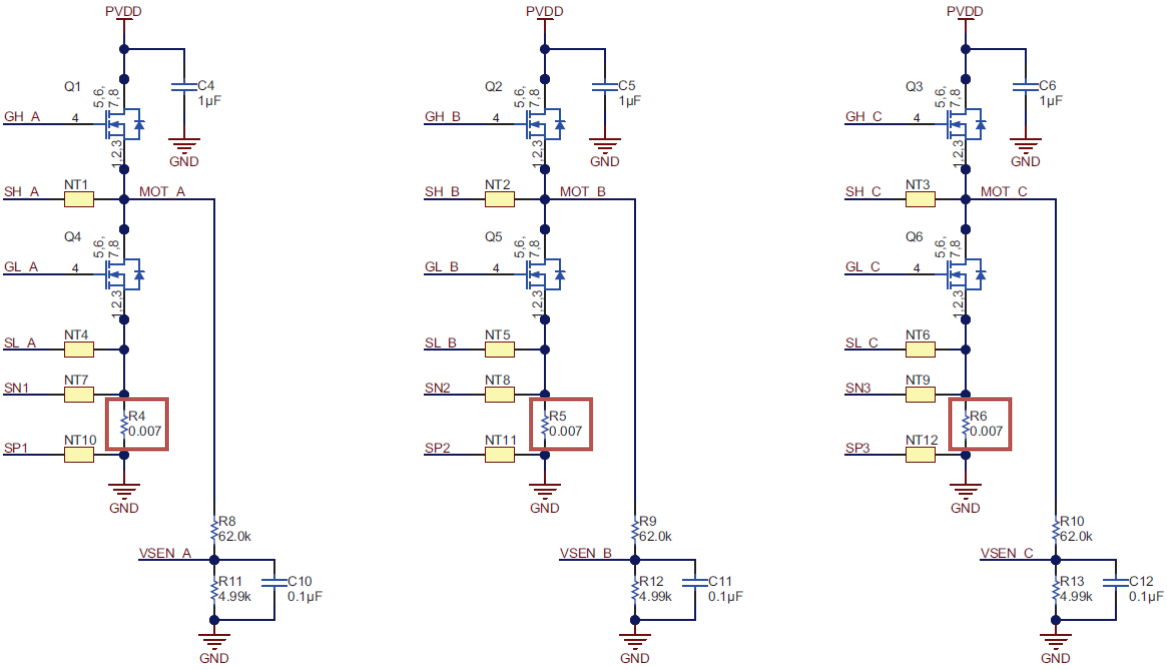
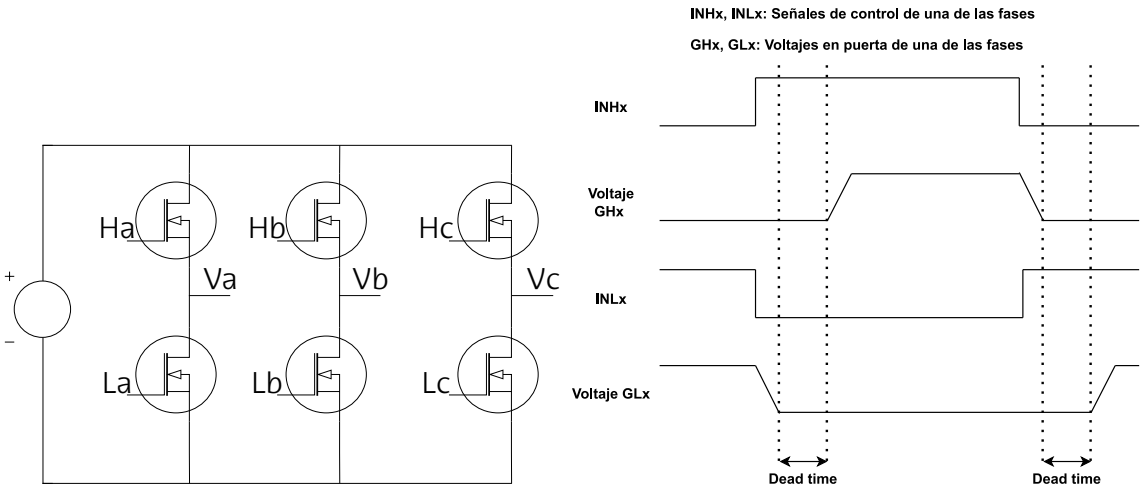


Figura 10: Detalle de los semipuentes MOSFET del driver de potencia

3.3. Plataformas embebidas RISC-V para el caso de estudio

Analizando la implementación anterior del B2B parcial sobre RISC-V single-core, se establecen los requisitos de la plataforma RISC-V óptima para el proyecto. Teniendo en cuenta el setup, los periféricos necesarios son:

- 6xPWM: Cada uno de los drivers de motor emplea como entrada tres señales PWM (high side del puente trifásico (Figura 11a)) y genera las otras tres (low side) con opción a configurar un dead time (Figura 11b) entre conmutaciones que es gestionado por el propio driver.
- 2xSPI: La configuración inicial de los registros de los drivers se realiza mediante SPI, por lo que se necesita tener sendas interfaces SPI para realizar dicha configuración inicial en ambos drivers. Además, los registros que contienen el diagnóstico cuando se produce un error se leen a través de SPI.
- 4xADC (4 canales de ADC): En el control FOC se monitorizan las corrientes de fase, el driver de potencia dispone de un shunt amplificado por un amplificador operacional por cada una de las fases, por lo que se requiere un canal de ADC para cada una de las corrientes medidas. Tres corrientes por motor (una por fase, de las cuales sólo se leen dos), cuatro corrientes en total.
- 2xSalidas digitales: Cada uno de los drivers dispone de un puerto enable que hay que activar o desactivar para habilitar o deshabilitar el funcionamiento del driver.
- 2xEntradas digitales: Cada uno de los drivers dispone de un pin indicador de fallo. Un nivel bajo en estos pines activaría una secuencia de lectura de los registros de fallo.
- 2xTimers: Para decodificar los encoder de cuadratura de cada uno de los motores.



(a) Puente de MOSFET trifásico **(b)** Tiempo muerto entre conmutaciones

3.3.1. Placas de desarrollo valoradas

Una vez definidos los requisitos de la aplicación, se realiza una búsqueda de plataformas y se estudia su aptitud para el proyecto. A lo largo de esta búsqueda se encuentran distintas dificultades, como la falta de stock disponible, la escasez de documentación de algunos fabricantes que dificulta la determinación de las funcionalidades del producto o la falta periféricos necesarios en la aplicación.

Como se observará más adelante, las placas de desarrollo que disponen de microcontroladores RISC-V multi-core están principalmente enfocadas a aplicaciones de computación más exigente, como alternativa a sistemas basados en x86. Por ende, a la hora de incluir periféricos y puertos de expansión, el foco está puesto en interfaces de gran ancho de banda como puede ser PCIe en lugar de otros periféricos más adaptados a la aplicación embebida estudiada como pueden ser PWM o SPI.

3.3.1.1. SiFive HiFive Unmatched

Es una placa de desarrollo que incluye el SoC Freedom U740 de SiFive, un microcontrolador de 5 núcleos basado en la arquitectura RV64IMAFDC con soporte para PCIe 3.0x8 y 16GB de RAM DDR4. Como se puede observar por las especificaciones anteriores, sumado a que el formato de la placa es Mini-ITX, este sistema está pensado para competir con sistemas basados en x86. En lo referente a puertos de expansión, la placa dispone de un cabezal de 24 pines compuesto por:

- 2xI2C
- 4xGPIO
- 2xUART
- 2xQSPI
- 1xPWM

Los periféricos accesibles mediante puertos de expansión no cubren las necesidades del proyecto, dado que dispone de una única salida PWM y cuatro GPIOs, lo que impide

manejar el driver de potencia empleado en el proyecto. Además, debido a la insuficiente capacidad de producción de ASICs que el mercado ha sufrido estos últimos años, SiFive ha descontinuado este modelo para centrar su producción en otros modelos de microcontrolador, por lo que no se ha encontrado stock disponible de esta placa de desarrollo.

3.3.1.2. Kendryte KD233

Es una placa de desarrollo basada en el microcontrolador dual-core Kendryte K210 de Canaan. Esta placa de desarrollo cuenta con una pantalla y una cámara integradas y está diseñada para aplicaciones de machine vision.

Los periféricos disponibles en esta tarjeta de desarrollo son:

- 3xI2C
- 3xSPI
- 3xUART
- 32xGPIO
- 1xCSI
- 1xDSI
- 3xTimer
- 6xPWM

La tarjeta también dispone de otros conectores como una expansión de almacenamiento mediante tarjetas TF, una interfaz para conectar un módulo WiFi comunicado mediante UART o una interfaz de audio. En el momento en el que se realiza la búsqueda de plataforma la tarjeta de desarrollo no se encuentra disponible para su compra, por lo que se descarta para el proyecto.

3.3.1.3. Aries Embedded MP100PFS

Es un SoM basado en la FPGA MPFS250T de microchip que combina cinco cores RISC-V de 64-bit (cuatro de ellos con la ISA RV64GC y un quinto core monitor con la ISA RV64IMAC) con las capacidades de una FPGA. Los soft cores que contiene utilizan un pipeline de cinco fases in-order. La CPU soft dispone de los siguientes periféricos:

- 2x Gigabit Ethernet
- 1x USB 2.0 OTG
- 2x CAN
- 2x SPI
- 2x I2C
- 5x UART

También dispone de las siguientes interfaces de conexión CPU-FPGA:

- 2xAXI4 de 64-bits (processor to fabric)
- 3xAXI4 de 64-bits (fabric to processor)
- 1xAPB de 32-bits (processor to fabric)

Debido a la complejidad añadida del diseño con FPGA, el elevado precio de la placa de desarrollo y a que la aplicación desarrollada en el proyecto, que no se beneficia de las características de una FPGA, se descarta este módulo.

3.3.1.4. StarFive VisionFive 2

Es un SBC con una GPU integrada cuyo SoC es un JH7110 de StarFive. Este microcontrolador, cuenta con cuatro cores U74 de SiFive basados en la ISA RV64GC, un core de monitorización S7, basado en la ISA RV64IMAC y un core de bajo consumo E24, basado en la ISA RV32IMFC, también de SiFive. La placa de desarrollo dispone de un puerto de expansión GPIO de 40 pines que cuenta con los siguientes periféricos:

- 1xUART
- 1xSPI
- 1xI2C
- 28xGPIO
- 2xPWM

La placa de desarrollo está disponible en versiones de 2/4/8 GB de RAM LPDDR4 y dispone también de los siguientes puertos:

- 2x USB 2.0
- 2x USB 3.0
- 2x RJ-45 Gigabit Ethernet
- 1x M.2 M-Key
- 2x HDMI

Debido a que esta placa de desarrollo forma parte de un proyecto kickstarter, su disponibilidad está sujeta a la obtención de financiación por parte de los fabricantes, lo que dificulta la compra a tiempo para la realización del proyecto. Además, no dispone de salidas PWM suficientes, ni de convertidores analógico-digital, por lo que no cumple con las necesidades del demostrador.

3.3.1.5. Kendryte K510 CRB-KIT

Es una placa de desarrollo basada en el SoM K510-CORE de Canaan, que dispone de un microcontrolador Kendryte K510. Este microcontrolador, basado en el core RISC-V AX25MP de Andes Technology, es un dual-core in-order de 5 fases. La placa de desarrollo soporta los siguientes periféricos:

- 3xSPI
- 3xI2C
- 3xI2S
- 3xUART
- 8xPWM
- 6xTimers

En el momento en el que se realiza la búsqueda esta tarjeta de desarrollo no se encuentra disponible para su compra, por lo que se descarta para el proyecto.

3.3.1.6. Sipeed Maixduino

Es una placa de desarrollo compuesta por dos microcontroladores conectados entre sí mediante una interfaz SPI. Uno de estos microcontroladores es un ESP32-WROOM-32 de EspressIf, mientras que el otro es un K210 de Canaan. Este último es un microcontrolador dual-core basado en la arquitectura RV64IMAFDC que cuenta con unidades de coma flotante independientes para cada core y un procesador para redes neuronales al que denominan KPU. Los periféricos con los que cuenta esta placa de desarrollo son:

- 3xI2C
- 3xSPI
- 3xUART
- 32xGPIO
- 1xCSI
- 1xDSI
- 3xTimer
- 6xPWM

Es importante señalar que, aunque todos estos periféricos están disponibles en el K210, no es posible emplear todos a la vez en la placa. Se pueden seleccionarla funcionalidad de cada pin del microcontrolador mediante el módulo FPIOA, con ciertas limitaciones.

En la Tabla 8 se resumen las placas consideradas con los criterios de selección. Como se puede observar, ninguna de las placas de desarrollo cumple con la totalidad de los requisitos. A esto le debemos sumar que la disponibilidad en el momento de la compra

es baja debido a que muchos de los fabricantes no consiguieron comprar capacidad de fabricación de ASICs durante el *silicon shortage* que se produjo en 2020, lo que ha propiciado que los productos menos exitosos se conviertan en *end-of-life* de manera prematura. Lo que conlleva a que la selección de un modelo apto para el caso de uso desarrollado en este trabajo se limita significativamente. Finalmente se opta por una solución híbrida basada en la placa de desarrollo **Sipeed Maixduino** de Canaan.

Tabla 8: Tabla resumen de las placas de desarrollo consideradas

Placa de desarrollo	PWM	SPI	ADC	GPIO	Entradas Encoder	Disponibilidad	Precio
SiFive HiFive Unmatched	1	0	0	4	0	No	-
Kendryte KD233	6	3	0	32	0	No	-
Aries Embedded MP100PFS	0	2	0	0	0	Sí	250$
StarFive VisionFive 2	2	1	0	28	0	No	111€
Kendryte K510 CRB-KIT	8	3	0	-	0	No	-
Sipeed Maixduino	**6**	**3**	**6***	**32**	**0**	**Sí**	**32€**

*Las entradas analógicas pertenecen al coprocesador ESP32 incluido a la placa, no al microcontrolador RISC-V principal (K210).

3.3.2. Solución propuesta

Una vez analizadas las placas de desarrollo basadas en microcontroladores RISC-V multi-core disponibles en el mercado y descartadas aquellas que no cumplen con los requisitos de la aplicación, se opta por un setup compuesto por dos placas de desarrollo comunicadas entre sí a través de un bus I2C.

1. Sipeed Maixduino
2. Teensy 3.6

De este modo, se quiere complementar la funcionalidad del microcontrolador dual-core Kendryte K210 de la Sipeed Maixduino, que está basado en RISC-V con un microcontrolador single-core basado en la arquitectura Cortex-M4. Se ha optado por esta solución porque, a pesar de que la placa de desarrollo Sipeed Maixduino dispone sobre el papel de la mayoría de los periféricos necesarios para implementar el demostrador, en la práctica el SDK del microcontrolador carece de la configurabilidad necesaria para implementar algunos de los periféricos, las carencias principales que se han encontrado son:

1. PWM alineadas en el centro: El driver que se utiliza para esta aplicación requiere que las entradas PWM de las fases del motor estén alineadas en el centro, la implementación de timers en modo PWM del SDK es bastante básica y sólo admite alineación a extremo, por lo que no es compatible con el driver.

2. Entradas de encoder de cuadratura: En el algoritmo de FOC sensorizado se utilizan encoders de cuadratura para contabilizar la posición del motor. Habitualmente esta lógica se implementa con un contador, pero en el caso del K210 esta opción no está implementada, por lo que habría que realizar esta funcionalidad por software, incurriendo en un uso de tiempo de computación adicional.

Al encontrar estas carencias en la placa de desarrollo seleccionada, se ha optado por complementar las funcionalidades de la Sipeed Maixduino con una Teensy 3.6, que dispone de un conjunto de periféricos más completo para hacer de interfaz con los drivers del motor.

De este modo, el planteamiento elegido es una placa de desarrollo (Sipeed Maixduino) para ejecutar el control en paralelo (cada motor se controla en un core) y una segunda placa de desarrollo (Teensy 3.6) para llevar a cabo la lectura de los datos y la transmisión de las señales de control procedentes de la Sipeed Maixduino. En la Figura 12 se muestra un diagrama conceptual del sistema completo.

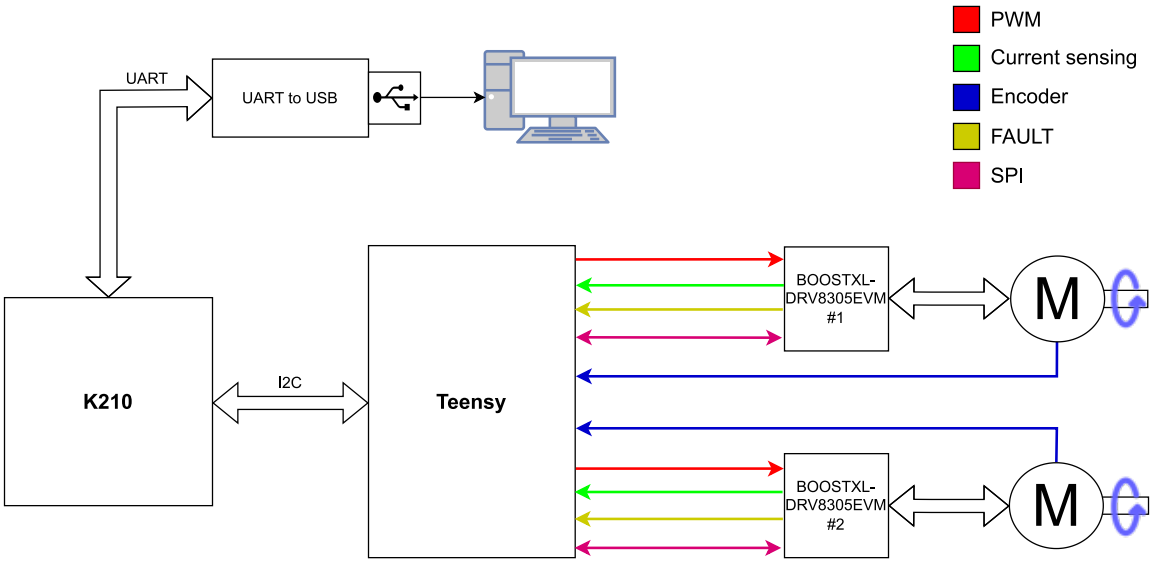


Figura 12: Diagrama conceptual del setup completo

El control se ejecuta cada $400\mu s$, generando una interrupción con un temporizador en la K210 que realiza una solicitud de datos a la Teensy mediante I2C, espera el resultado y activa ambos controles, cada uno de los cuales se ejecuta en un core. Una vez que ambos cores han terminado de realizar los cálculos, se realiza un envío de las consignas PWM mediante I2C.

K210 Core 0:

```
//La ISR del timer se ejecuta cada 400us
TIMER_ISR(){
    //Mensaje I2C solicitando los datos de los motores
    REQUEST_DATA(*DataMotor1, *DataMotor2);

    //Cálculo de consignas PWM para el motor #2 (ejecutado en el Core 1)
    TriggerControl2 = 1;

    //Cálculo de consignas PWM para el motor #1 (ejecutado en el Core 0)
    CONTROL#1(*DataMotor1);
```

```
    while (!Control2Terminado){
        ;            //Bucle para asegurar que el cálculo
                     //en el segundo core ha terminado.
    }
    //Mensaje I2C enviando las consignas PWM calculadas
    SEND_DATA(*PWM1Values, *PWM2Values);
}
```

K210 Core 1:

```
if(TriggerControl2 == 1)
    Control2Terminado = 0;

    //Cálculo de consignas PWM para el motor #2
    CONTROL#2(*DataMotor2);

    Control2Terminado = 1;
    TriggerControl2 =0;

}
```

Por parte de la Teensy, tanto la solicitud I2C como el envío generan interrupciones, de modo que al recibir una solicitud, se envían los datos más recientemente adquiridos y, al recibir un envío, se actualizan las consignas PWM de las salidas.

Teensy:

```
REQUEST_ISR(){
    //Envío de datos de los motores por I2C
    SEND_DATA(*DataMotor1, *DataMotor2);
}
RECEIVE_ISR(){
    //Modificar los ciclos de trabajo para cada fase
    SET_PWM_VALUES_MOTOR#1(*PWM1Values);
    SET_PWM_VALUES_MOTOR#2(*PWM2Values);
}
```

La Figura 13 muestra un diagrama temporal de la ejecución. Los tiempos de ejecución para la Teensy que se pueden ver en la figura son orientativos, mientras que los tiempos de ejecución descritos para la K210 están basados en la visualización de las tramas I2C en un analizador lógico.

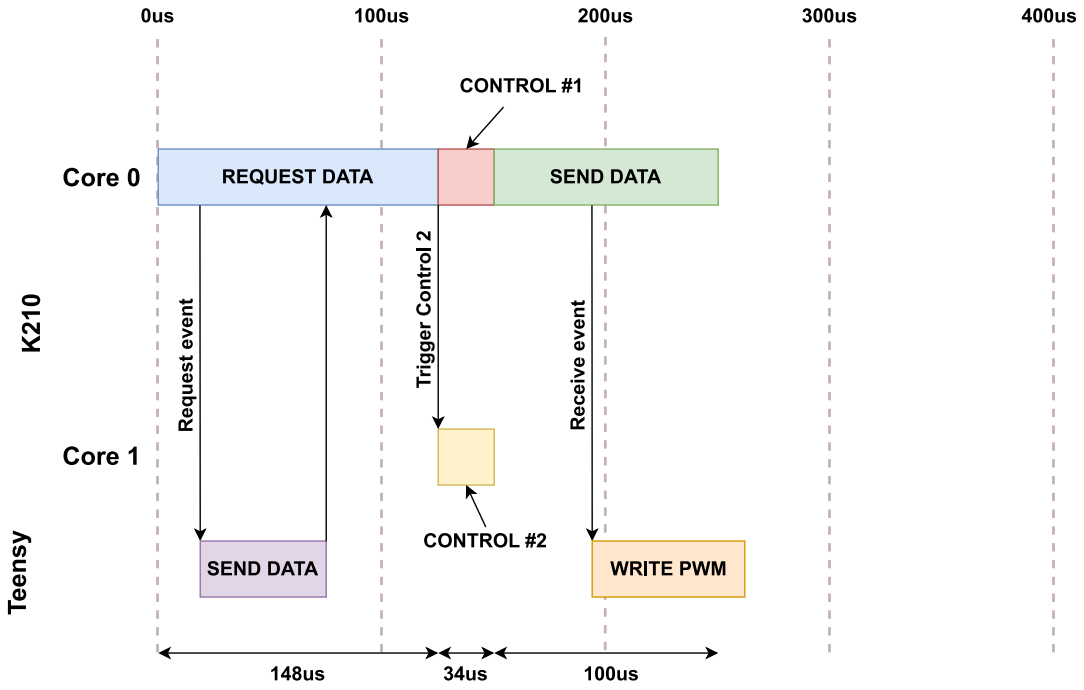


Figura 13: Desarrollo temporal del control por core

4. Validación del demostrador

Para asegurar que el funcionamiento del sistema es el esperado, se han diseñado varias pruebas. En primer lugar se comprueba durante su implementación que el funcionamiento de los periféricos individuales es correcto; posteriormente se verifica que el sistema de control en su conjunto es funcional y responde a las consignas fijadas. Estas pruebas se realizan en primer lugar para un único motor en single-core, comparando los resultados con la implementación de control de motores previa de realizada sobre la RV-STAR (Figura 3). Posteriormente se realizan las pruebas para ambos motores (Figura 4), en primer lugar con ambos controles ejecutándose en el mismo core, y, finalmente, con cada control ejecutándose en un core, tal y como se muestra en la Figura 13.

4.1. Validación de periféricos

4.1.1. Interrupción del timer

Para el funcionamiento del lazo de control, se requiere una señal periódica que sincronice la ejecución. Esta señal se obtiene mediante una interrupción generada por un contador cada $400\mu s$. Para comprobar si la señal se produce a tiempo con la periodicidad requerida, se activa un GPIO del microcontrolador en la ISR del contador y se registra la forma de onda resultante en un analizador lógico. En la Figura 14 se puede ver una captura de la trama en el analizador lógico.

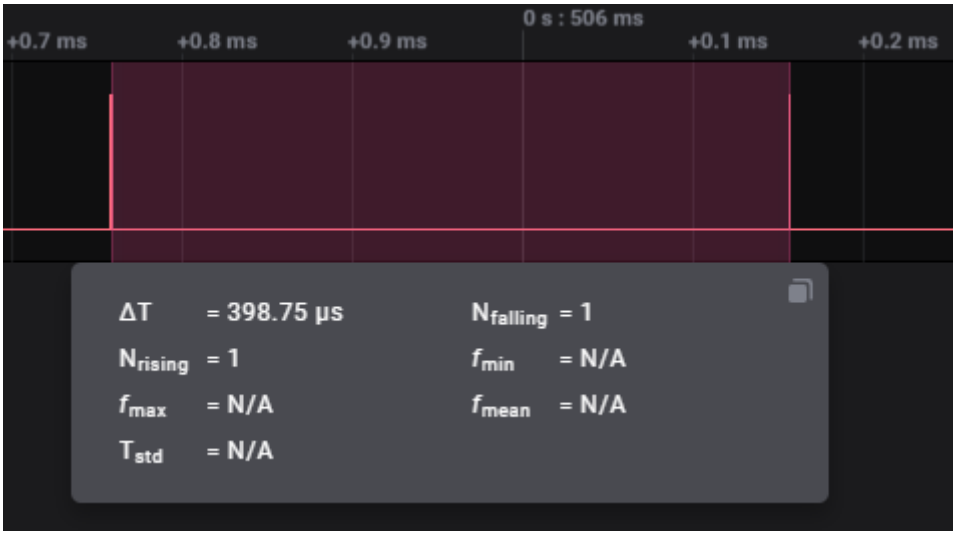


Figura 14: Captura de la línea de GPIO activada en la ISR del timer

Para comprobar la dispersión de esta medida durante la ejecución, se han tomado mil muestras y se ha medido el lapso de tiempo entre flancos de subida. Los resultados de la prueba son:

Tabla 9: Análisis de tiempos de la ISR del timer

Valor máximo (μs)	399.875
Valor mínimo (μs)	398.75
Moda (μs)	399.75
Dispersión (μs)	1.125

4.1.2. Salidas PWM

Para comprobar el funcionamiento de las salidas PWM se han configurado distintos ciclos de trabajo y se ha comprobado que se producen las formas de onda esperadas mediante el analizador lógico. La señal resultante ha de ser una PWM de **10kHz** alineada en el centro, es decir, ha de tener un periodo de $100\mu s$. En esta prueba se han definido los ciclos de trabajo reflejados en la Tabla 10:

Tabla 10: Ciclos de trabajo empleados en la prueba

	Fase	**Duty cycle #1**	**Duty cycle #2**
Motor 1	A	10 %	60 %
	B	20 %	50 %
	C	30 %	40 %
Motor 2	A	40 %	30 %
	B	50 %	20 %
	C	60 %	10 %

Las formas de onda resultantes de la prueba se pueden observar en la Figura 15

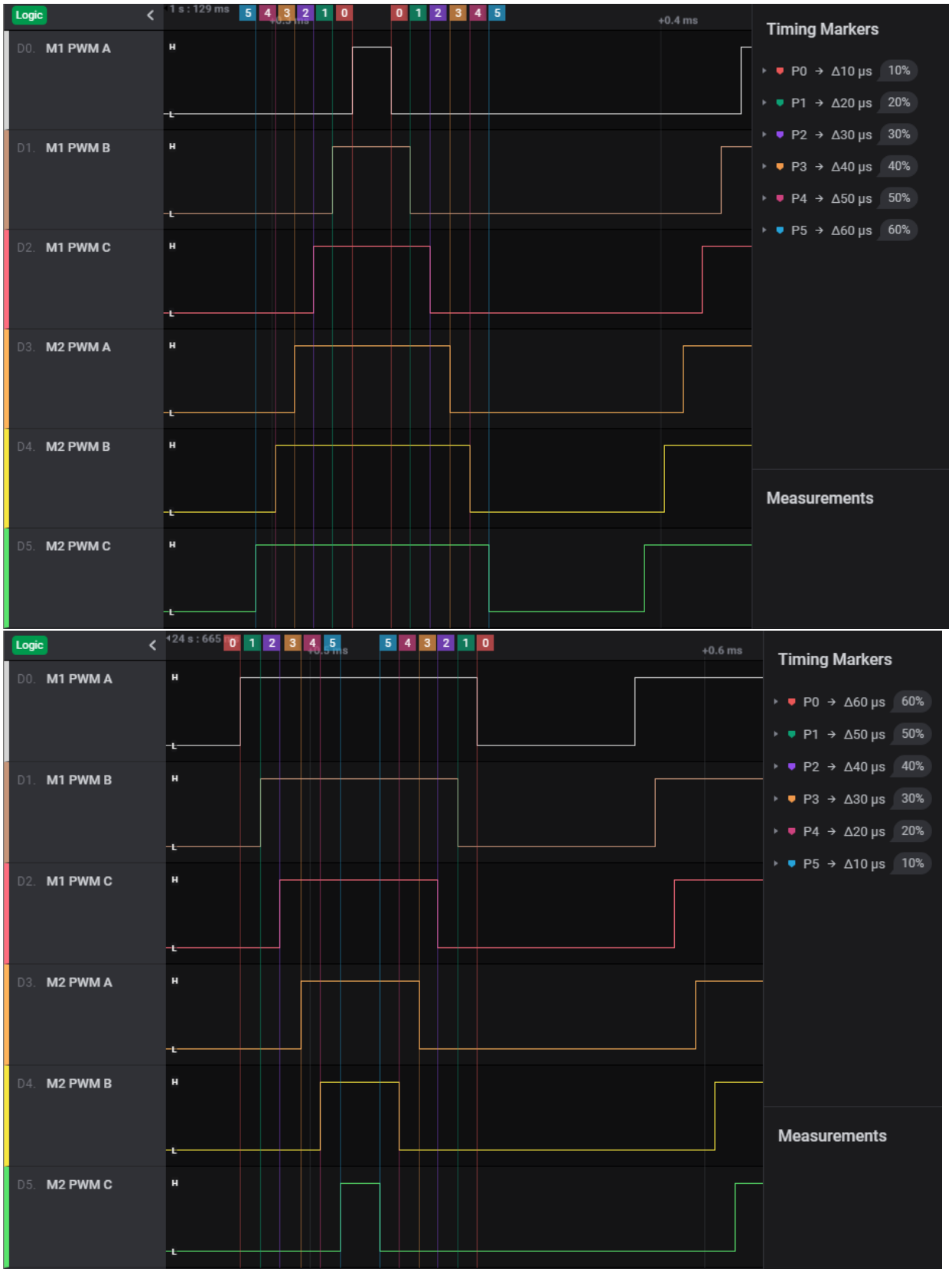


Figura 15: Verificación de las salidas PWM

4.1.3. Lectura del encoder

Para comprobar la lectura del encoder se realiza un giro a mano del rotor de cada motor y se imprimen mediante UART los valores de posición a lo largo de una vuelta. Los valores impresos se reflejan en la Tabla 11.

Tabla 11: Transcripción de los mensajes de posición recibidos por UART

Posicion motor 2: 3920	Posicion motor 1: 3994
Posicion motor 2: 3904	Posicion motor 1: 3988
Posicion motor 2: 3894	Posicion motor 1: 3983
Posicion motor 2: 3771	Posicion motor 1: 3973
Posicion motor 2: 3564	Posicion motor 1: 3971
Posicion motor 2: 3423	Posicion motor 1: 3919
Posicion motor 2: 3301	Posicion motor 1: 3805
Posicion motor 2: 3201	Posicion motor 1: 3754
Posicion motor 2: 2990	Posicion motor 1: 3581
Posicion motor 2: 2888	Posicion motor 1: 3539
Posicion motor 2: 2841	Posicion motor 1: 3519
Posicion motor 2: 2786	Posicion motor 1: 3472
Posicion motor 2: 2636	Posicion motor 1: 3388
Posicion motor 2: 2347	Posicion motor 1: 3348
Posicion motor 2: 2148	Posicion motor 1: 3322
Posicion motor 2: 1795	Posicion motor 1: 3310
Posicion motor 2: 1621	Posicion motor 1: 3233
Posicion motor 2: 1253	Posicion motor 1: 3094
Posicion motor 2: 983	Posicion motor 1: 3054
Posicion motor 2: 953	Posicion motor 1: 2959
Posicion motor 2: 952	Posicion motor 1: 2885
Posicion motor 2: 951	Posicion motor 1: 2878
Posicion motor 2: 949	Posicion motor 1: 2871
Posicion motor 2: 909	Posicion motor 1: 2839
Posicion motor 2: 882	Posicion motor 1: 2810
Posicion motor 2: 844	Posicion motor 1: 2670
Posicion motor 2: 831	Posicion motor 1: 2350
Posicion motor 2: 803	Posicion motor 1: 2304
Posicion motor 2: 701	Posicion motor 1: 2303
Posicion motor 2: 476	Posicion motor 1: 2270
Posicion motor 2: 432	Posicion motor 1: 2256
Posicion motor 2: 426	Posicion motor 1: 2238
Posicion motor 2: 416	Posicion motor 1: 1935
Posicion motor 2: 395	Posicion motor 1: 1927
Posicion motor 2: 370	Posicion motor 1: 1868
Posicion motor 2: 328	Posicion motor 1: 1756
Posicion motor 2: 240	Posicion motor 1: 1678
Posicion motor 2: 131	Posicion motor 1: 1394
	Posicion motor 1: 1313
	Posicion motor 1: 1272
	Posicion motor 1: 1062
	Posicion motor 1: 897
	Posicion motor 1: 874
	Posicion motor 1: 833
	Posicion motor 1: 749
	Posicion motor 1: 625
	Posicion motor 1: 462
	Posicion motor 1: 386
	Posicion motor 1: 275
	Posicion motor 1: 162

4.1.4. Comunicación I2C

Se utiliza I2C para comunicar las dos placas de desarrollo entre sí. A través de esta interfaz se transmiten los datos leídos desde el motor y los drivers y se devuelven los valores calculados para los ciclos de trabajo de las señales PWM. La interfaz I2C de esta implementación tiene una frecuencia de funcionamiento de **2 MHz**.

Para validar su funcionamiento, se transmite una trama conocida y se comprueba que:

1. La transmisión es legible por un analizador lógico.
2. Los valores recibidos son iguales a los enviados. Para ello se transmitirá la trama recibida por UART (ver Figura 17).

La trama transmitida por I2C consiste en tres bytes (Tabla 12):

Tabla 12: Trama I2C

0x55	0x56	0x57

En la Figura 16 se puede observar la forma de onda que genera la trama I2C capturada por el analizador lógico, y, en la Figura 17, se pueden ver los valores recibidos impresos por la Teensy.

Figura 16: Trama I2C visualizada en el analizador lógico

Figura 17: Valores recibidos por I2C impresos por UART

4.1.5. Programación de los drivers mediante SPI

La programación de los driver de los motores se realiza mediante SPI. Para comprobar que la programación es correcta, se escribe en primer lugar a los registros del driver la configuración deseada; seguidamente, el driver responde con el contenido del registro, de manera que, transmitiendo por UART el mensaje de vuelta recibido desde el driver, se comprueba que los datos escritos se corresponden con los que se quería escribir. Los valores de configuración que se quieren escribir en los registros de los driver se pueden observar en la Tabla 13.

Tabla 13: Valores a escribir via SPI

Número de registro (BIN)	Número de registro (HEX)	Valor (BIN)	Valor (HEX)	Trama enviada
0101	5	00001000100	44	0x2844
0110	6	00001000100	44	0x3044
0111	7	00000010110	16	0x3816
1001	9	00000100000	20	0x4820
1010	10	00000101010	2A	0x502A
1011	11	00100001010	10A	0x590A
1100	12	00011001000	C8	0x60C8

Los valores recibidos mediante la UART (Figura 18) muestran que los registros han sido escritos correctamente.

Los envíos realizados son tramas de 16-bit, en las cuales el bit más significativo indica un comando de lectura ('1') o escritura ('0'). Los siguientes 4 bits indican la dirección a la que se escribe o de la que se lee, finalmente, los once bits siguientes indican los datos a escribir en el caso de que sea un comando tipo "write". Ambos drivers se programan con la misma configuración, por lo que se reciben las tramas de datos de vuelta por duplicado:

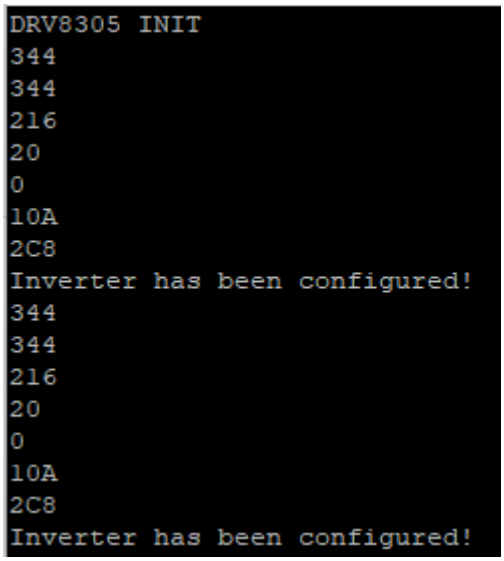


Figura 18: Captura de escritura a los registros SPI

4.1.6. Sincronización de la posición en lazo abierto

Debido a que el control FOC que se realiza en el proyecto es sensorizado (es decir, no es sensorless FOC), es un requisito imprescindible conocer la posición del rotor en todo momento para orientar el campo. Por ello, dado que la posición inicial del rotor nos es desconocida, la primera fase del control es un posicionamiento en lazo abierto del rotor. Esta posición inicial se sincroniza mediante la señal Index del encoder de cuadratura. Para comprobar que el posicionamiento es correcto, se realiza esta fase inicial del programa partiendo de varias posiciones aleatorias (girando el eje a mano) y se comprueba que la posición final es la misma en todos los casos.

Esta comprobación se hace enviando por UART el valor de contaje de los contadores que registran la posición del encoder. Se han realizado cuatro iteraciones de esta prueba:

```
Posicion motor 1: 338
Posicion motor 2: 329
```

(a) Prueba en lazo abierto 1

```
Posicion motor 1: 339
Posicion motor 2: 329
```

(b) Prueba en lazo abierto 2

```
Posicion motor 1: 339
Posicion motor 2: 329
```

(c) Prueba en lazo abierto 3

```
Posicion motor 1: 339
Posicion motor 2: 329
```

(d) Prueba en lazo abierto 4

4.2. Validación del sistema con un motor

Para comprobar el funcionamiento del control de un motor, se realizan varias pruebas de escalón a distintas consignas de velocidad. Durante estas pruebas, se recogen los datos percibidos mediante I2C a lo largo de mil muestras (0.4s) y, tras finalizar la prueba, se transmiten al PC mediante UART, se guardan los datos en ficheros CSV y se grafica el resultado mediante un script Python.

Estas pruebas se realizan también para la plataforma de control de motores implementada en la tarjeta de desarrollo RV-STAR con la que se quiere comparar la implementación. En el caso de la RV-STAR, debido a que no se dispone de RAM suficiente para adquirir mil muestras, se ha reducido la prueba a ochocientas muestras (0.32s).

Se realiza la prueba para los siguientes valores de consigna de velocidad:

- Implementación Actual
  - 20 rad/s →Figura 20a
  - 50 rad/s →Figura 21a
  - 80 rad/s →Figura 22a
  - 100 rad/s →Figura 23a
  - 120 rad/s →Figura 24a
  - 150 rad/s →Figura 25a
- Implementación anterior (RV-STAR)
  - 20 rad/s →Figura 20b
  - 50 rad/s →Figura 21b

- 80 rad/s →Figura 22b
- 100 rad/s →Figura 23b
- 120 rad/s →Figura 24b
- 150 rad/s →Figura 25b

En cada gráfica se muestran cuatro parámetros distintos medidos a lo largo de la prueba:

1. Velocidad del motor vs. velocidad de referencia.
2. Posición del motor (ángulo mecánico).
3. Corriente de fase A.
4. Corriente de fase B.

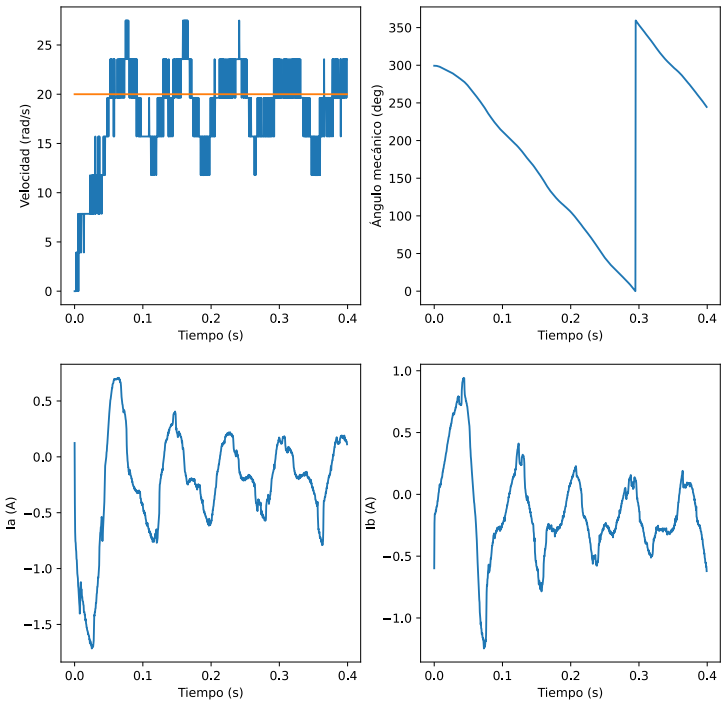
A simple vista, el rizado en la velocidad es mayor en la implementación realizada en este proyecto que en la implementación previa realizada sobre la RV-STAR. Si se analizan los datos de velocidad en régimen estacionario, los resultados son los siguientes:

Tabla 14: Resultados de velocidad para un motor

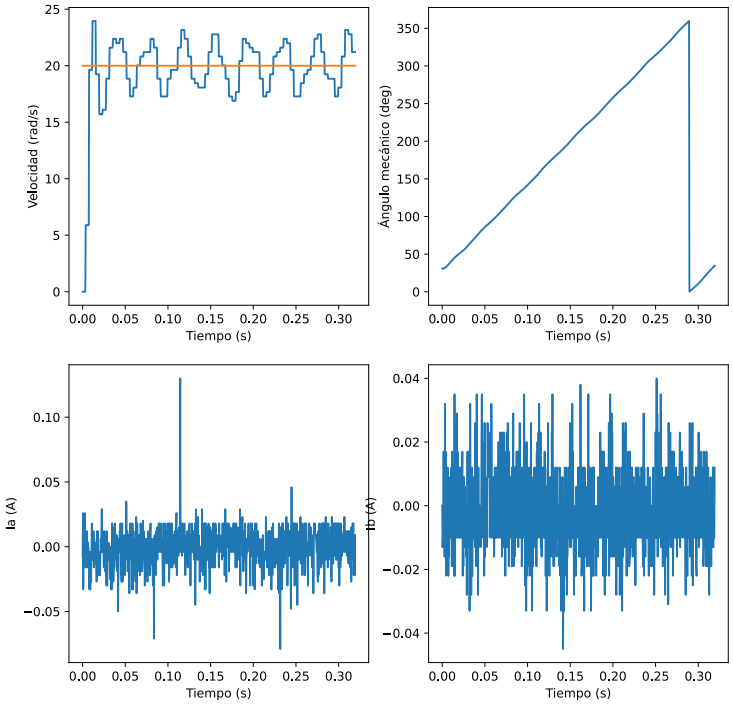
	RV-STAR			Implementación actual		
Vref	**Max**	**Min**	**Delta**	**Max**	**Min**	**Delta**
20	23,169	16,886	6,283	27,489	11,781	15,708
50	51,836	47,909	3,927	54,978	43,197	11,781
80	81,288	79,325	1,963	86,394	74,613	11,781
100	100,530	98,960	1,570	106,029	94,248	11,781
120	120,558	119,380	1,178	125,664	113,883	11,781
150	150,796	149,225	1,571	157,080	141,372	15,708

Esta diferencia con el sistema de referencia se puede deber a la mayor latencia que existe en la adquisición de los datos. Al no disponer de los periféricos necesarios en el propio microcontrolador de la tarjeta de desarrollo Sipeed Maixduino, se ha tenido que añadir una comunicación I2C en la cadena.

Esto implica necesariamente que hay un retardo entre la lectura de los datos y el cálculo de la señal de control (Figura 13). Este desfase entre el estado de la planta en el instante de medición de los datos y el estado de la planta cuando se establece la señal de control hace que el comportamiento de la planta sea peor que en el caso de la RV-STAR, donde todo el sistema estaba integrado en un único microcontrolador y la latencia era significativamente menor.

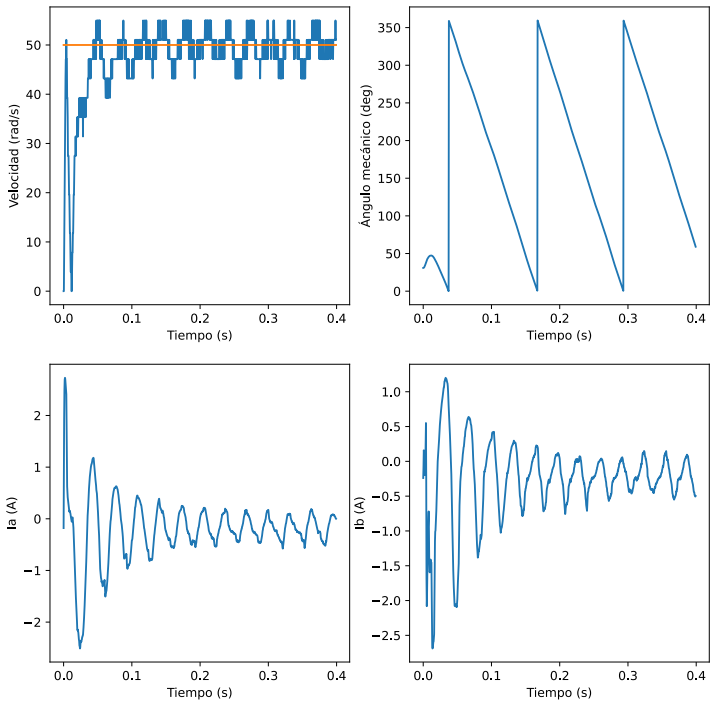


(a) Escalón a 20 rad/s con un motor en la implementación actual

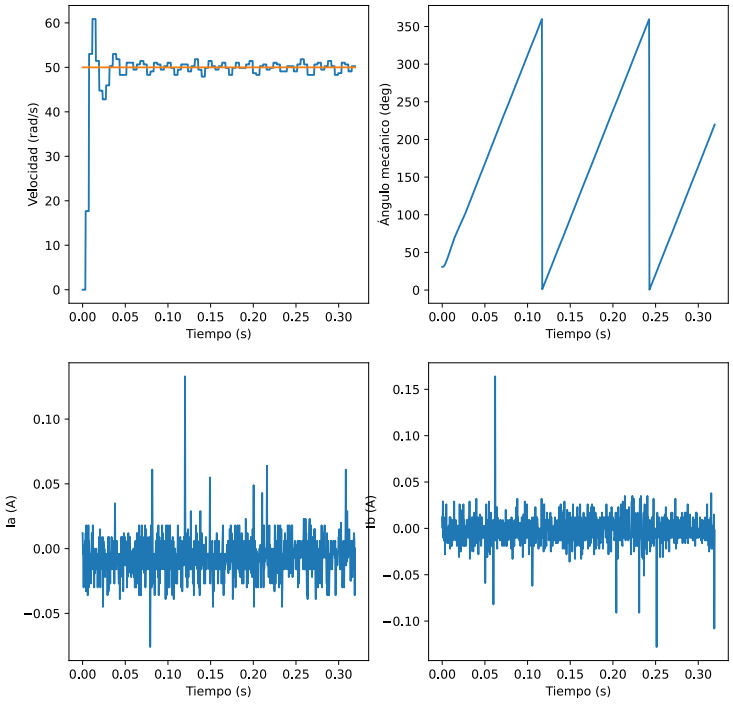


(b) Escalón a 20 rad/s con un motor en la RV-STAR

Figura 20: Escalón a 20 rad/s con un motor

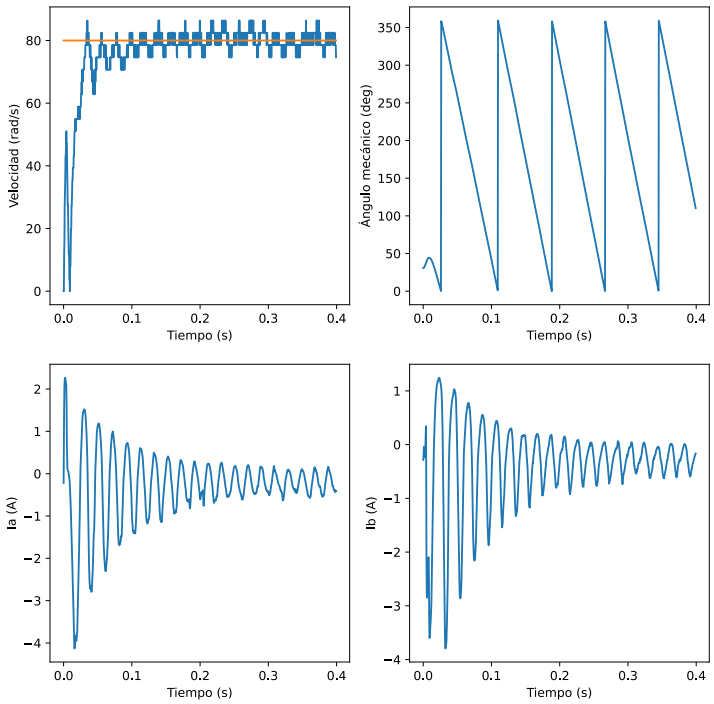


(a) Escalón a 50 rad/s con un motor en la implementación actual

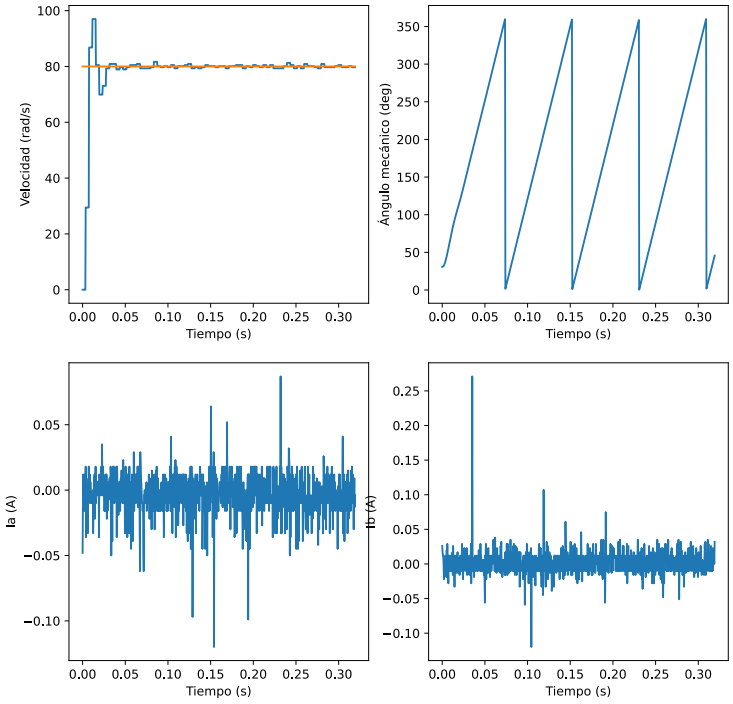


(b) Escalón a 50 rad/s con un motor en la RV-STAR

Figura 21: Escalón a 50 rad/s con un motor

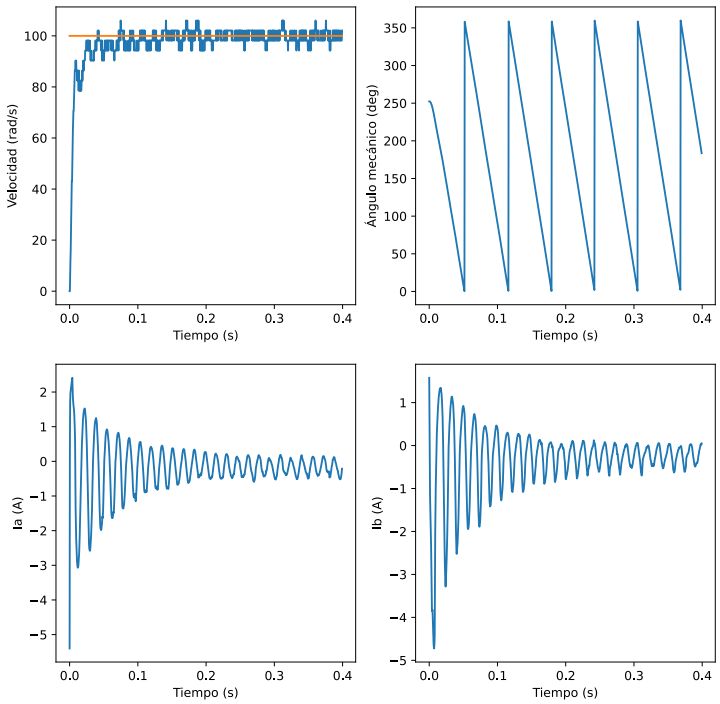


(a) Escalón a 80 rad/s con un motor en la implementación actual

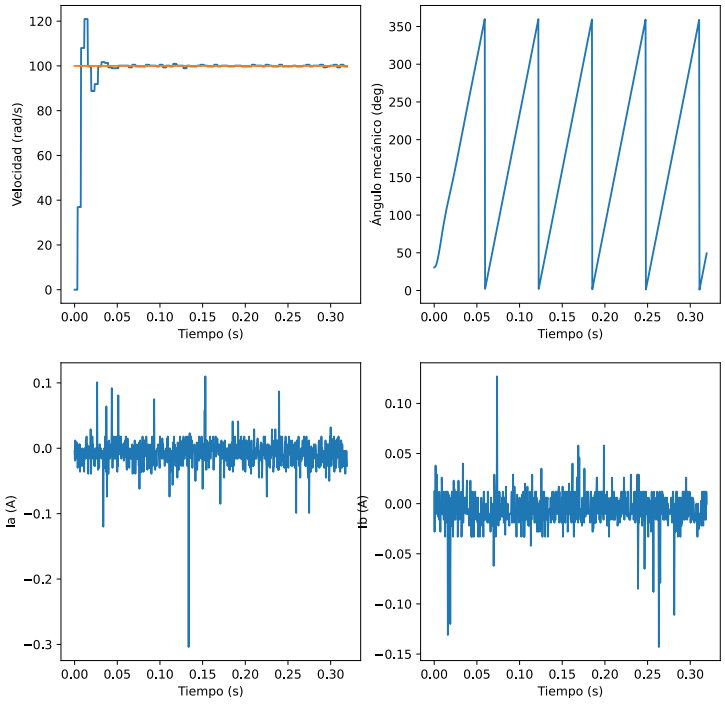


(b) Escalón a 80 rad/s con un motor en la RV-STAR

Figura 22: Escalón a 80 rad/s con un motor

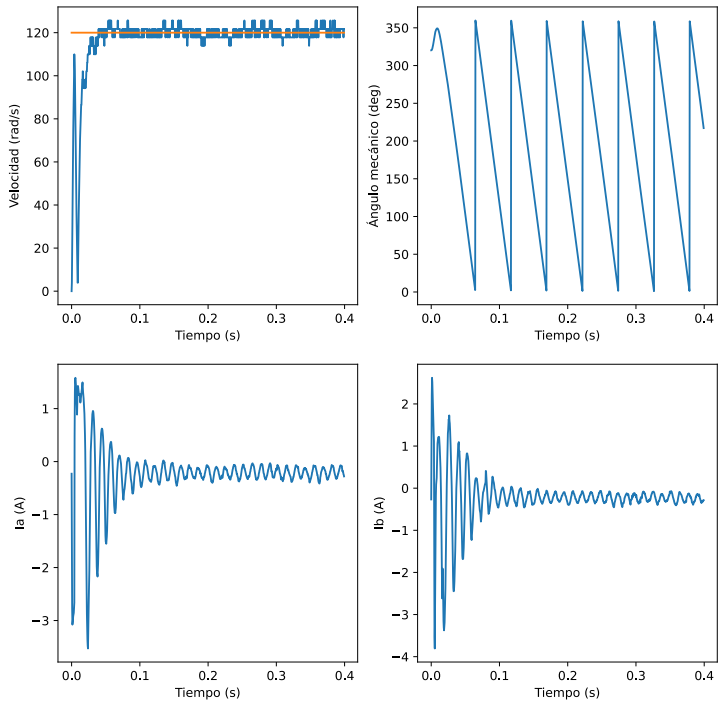


(a) Escalón a 100 rad/s con un motor en la implementación actual

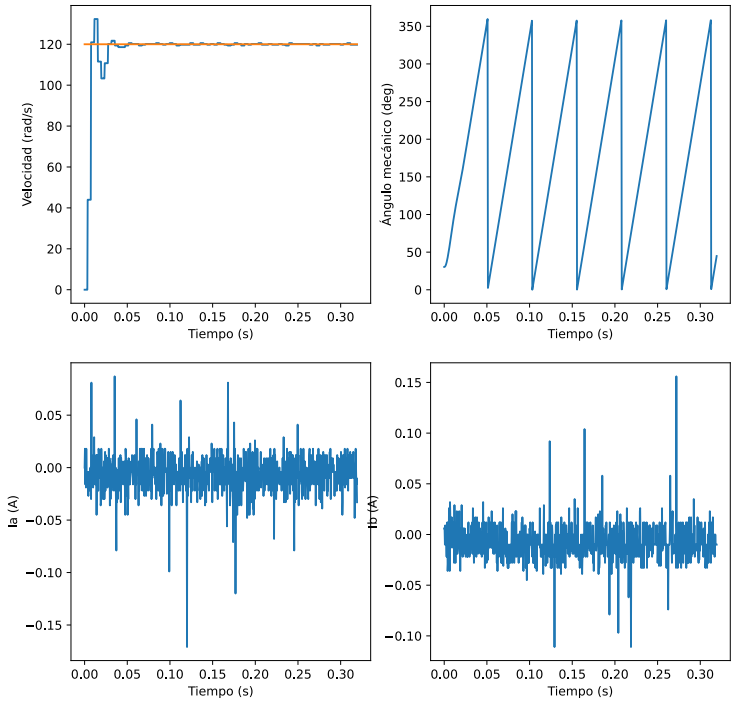


(b) Escalón a 100 rad/s con un motor en la RV-STAR

Figura 23: Escalón a 100 rad/s con un motor

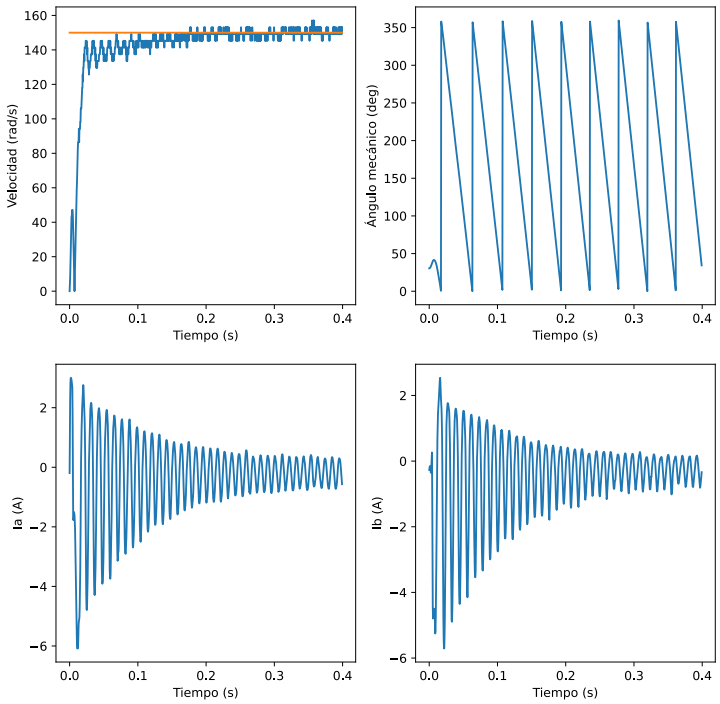


(a) Escalón a 120 rad/s con un motor en la implementación actual

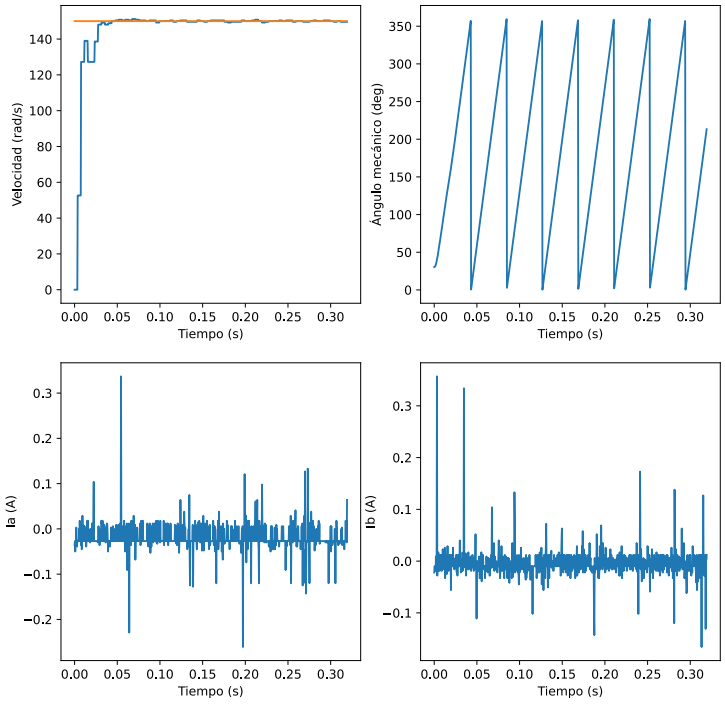


(b) Escalón a 120 rad/s con un motor en la RV-STAR

Figura 24: Escalón a 120 rad/s con un motor



(a) Escalón a 150 rad/s con un motor en la implementación actual

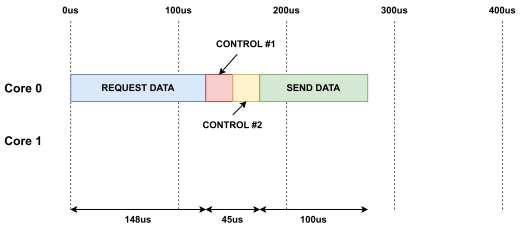


(b) Escalón a 150 rad/s con un motor en la RV-STAR

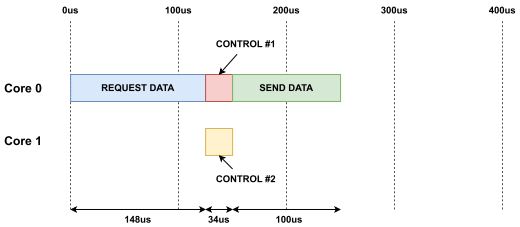
Figura 25: Escalón a 150 rad/s con un motor

4.3. Validación del sistema completo

Para comprobar el funcionamiento simultáneo de ambos motores, se repite la prueba anterior, en esta ocasión recogiendo datos de ambos motores. La prueba se realiza tanto para la ejecución en serie de los dos controles en un único core (Figura 26a) como para la ejecución del control en paralelo (Figura 26b).



(a) Cronograma de la ejecución en serie del control



(b) Cronograma de la ejecución paralela del control

Las gráficas en esta ocasión muestran una comparativa entre la respuesta en el primer motor y la respuesta del segundo. Los parámetros comparados son:

1. Velocidad del motor vs. velocidad de referencia de cada motor.
2. Corriente de la fase A en cada motor.
3. Corriente de la fase B en cada motor.
4. Posición del rotor de cada motor.

Se realizan las pruebas para los siguientes valores de consigna de velocidad:

1. Ejecución del control en serie
   - 80 rad/s →Figura 27
   - 100 rad/s →Figura 28
   - 120 rad/s →Figura 29
   - 150 rad/s →Figura 30
2. Ejecución del control en paralelo
   - 80 rad/s →Figura 31
   - 100 rad/s →Figura 32
   - 120 rad/s →Figura 33
   - 150 rad/s →Figura 34

Si se observan los datos de esta prueba se puede apreciar que la respuesta del control del segundo motor en régimen estacionario es significativamente más oscilante que la del primero. Se catalogan los valores máximos y mínimos del régimen estacionario (Tabla 15, Tabla 16). Adicionalmente, se puede observar que la ejecución en serie del control no afecta a la respuesta de los motores. Esto se debe a que, como se aprecia en la Figura 26a, para el periodo para el que se ha diseñado este control ($400\mu s$), hay un remanente de tiempo entre la recepción de las señales del control de una iteración y el comienzo de la siguiente.

Tabla 15: Resultados de velocidad para dos motores con el control ejecutado en un único core

	Motor 1			Motor 2		
Vref	**Max**	**Min**	**Delta**	**Max**	**Min**	**Delta**
80	84,467	78,540	3,927	117,810	58,905	58,905
100	102,102	98,175	3,927	121,737	82,467	39,270
120	121,737	117,810	3,927	137,445	102,102	35,343
150	153,153	145,299	7,854	164,934	137,445	27,489

Tabla 16: Resultados de velocidad para dos motores con el control ejecutado en paralelo

	Motor 1			Motor 2		
Vref	**Max**	**Min**	**Delta**	**Max**	**Min**	**Delta**
80	82,467	78,540	3,927	109,955	51,051	58,905
100	102,102	98,175	3,927	125,664	78,540	47,124
120	121,737	117,810	3,927	141,372	98,175	43,197
150	153,153	145,299	7,854	168,861	133,518	35,343

Los datos de la Tabla 15 y la Tabla 16 muestran que a velocidades mayores la amplitud de las oscilaciones observadas es menor, sin embargo, esta no es una tendencia que se pueda establecer con los datos recopilados, sino que requeriría de un estudio más exhaustivo en el que analizar el rango de operación del motor bajo este algoritmo de control. Como hasta este punto se ha utilizado el mismo motor para todas las pruebas, y es el motor para el que se modeló este control, se quiere descartar que la diferencia en la respuesta se deba a que los parámetros de los PI que componen el control no sean adecuados para el otro motor, por lo que se intercambian los motores y se repite la prueba; sin embargo, el resultado obtenido no varía, por lo que se descarta esta hipótesis.

En el momento en el que se redacta este documento no se ha concluido el porqué de este comportamiento asimétrico en el control del segundo motor, por falta de tiempo no se han podido realizar más pruebas, por lo que se atribuye a un error de programación que no se ha llegado a depurar.

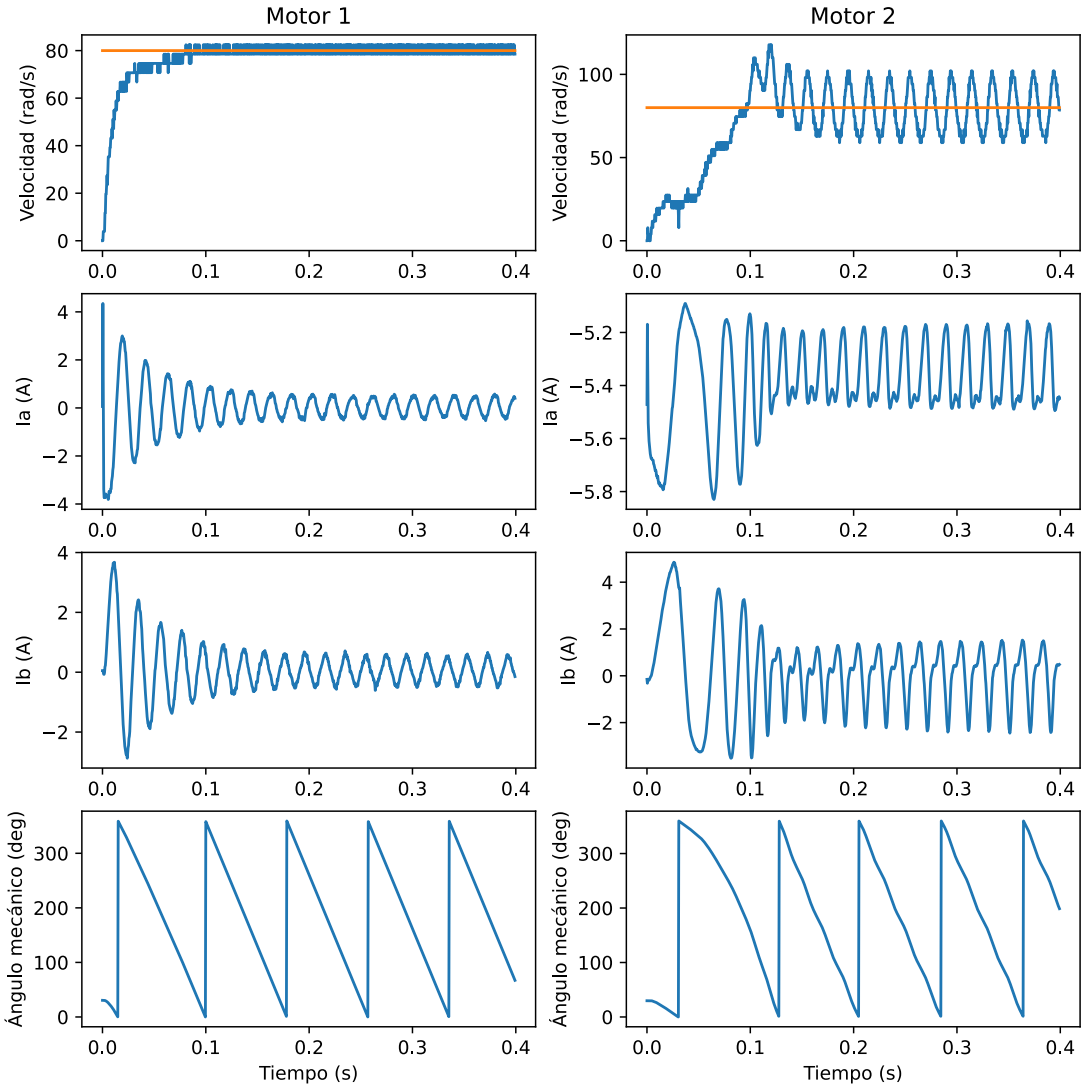


Figura 27: Escalón a 80 rad/s con dos motores controlados en serie

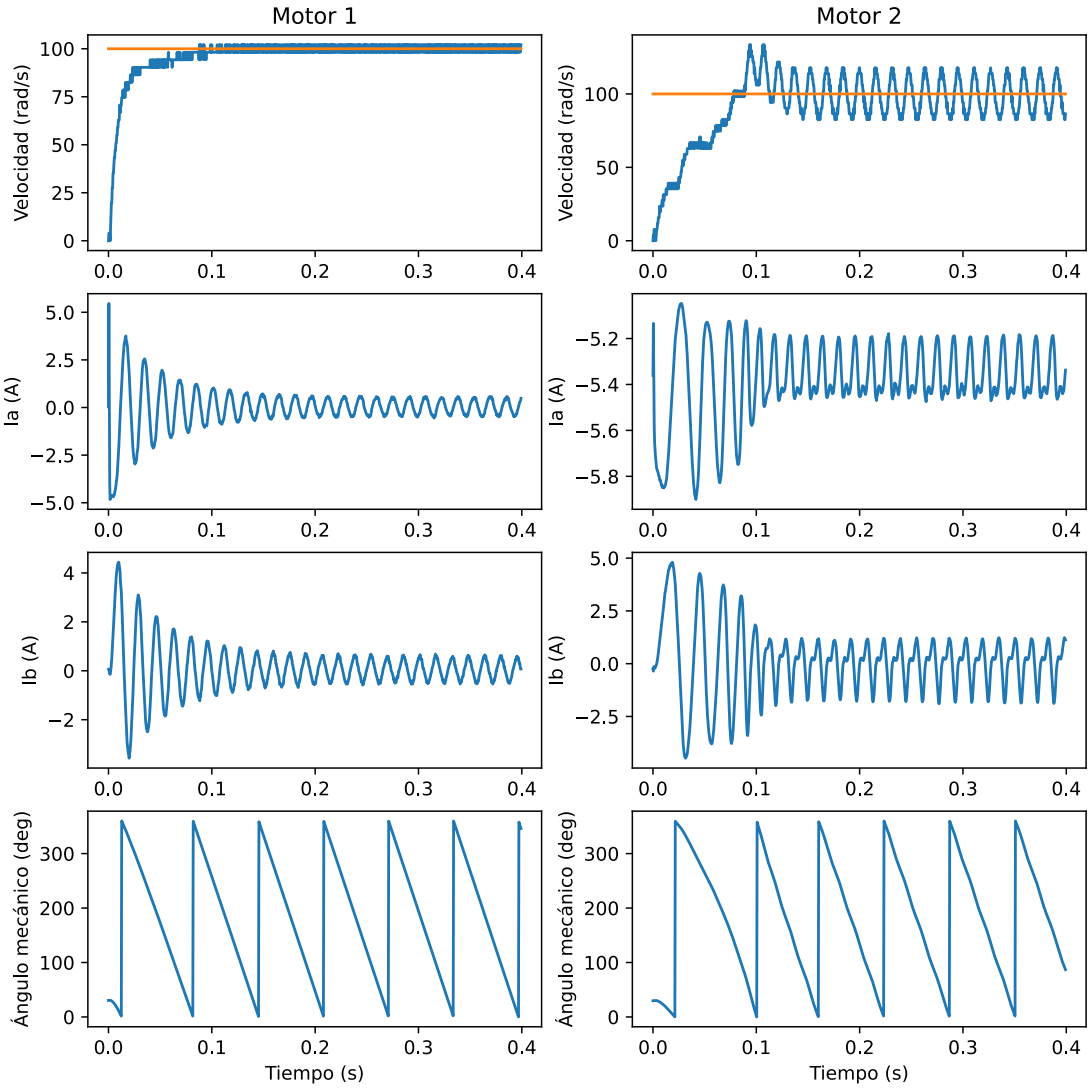


Figura 28: Escalón a 100 rad/s con dos motores controlados en serie

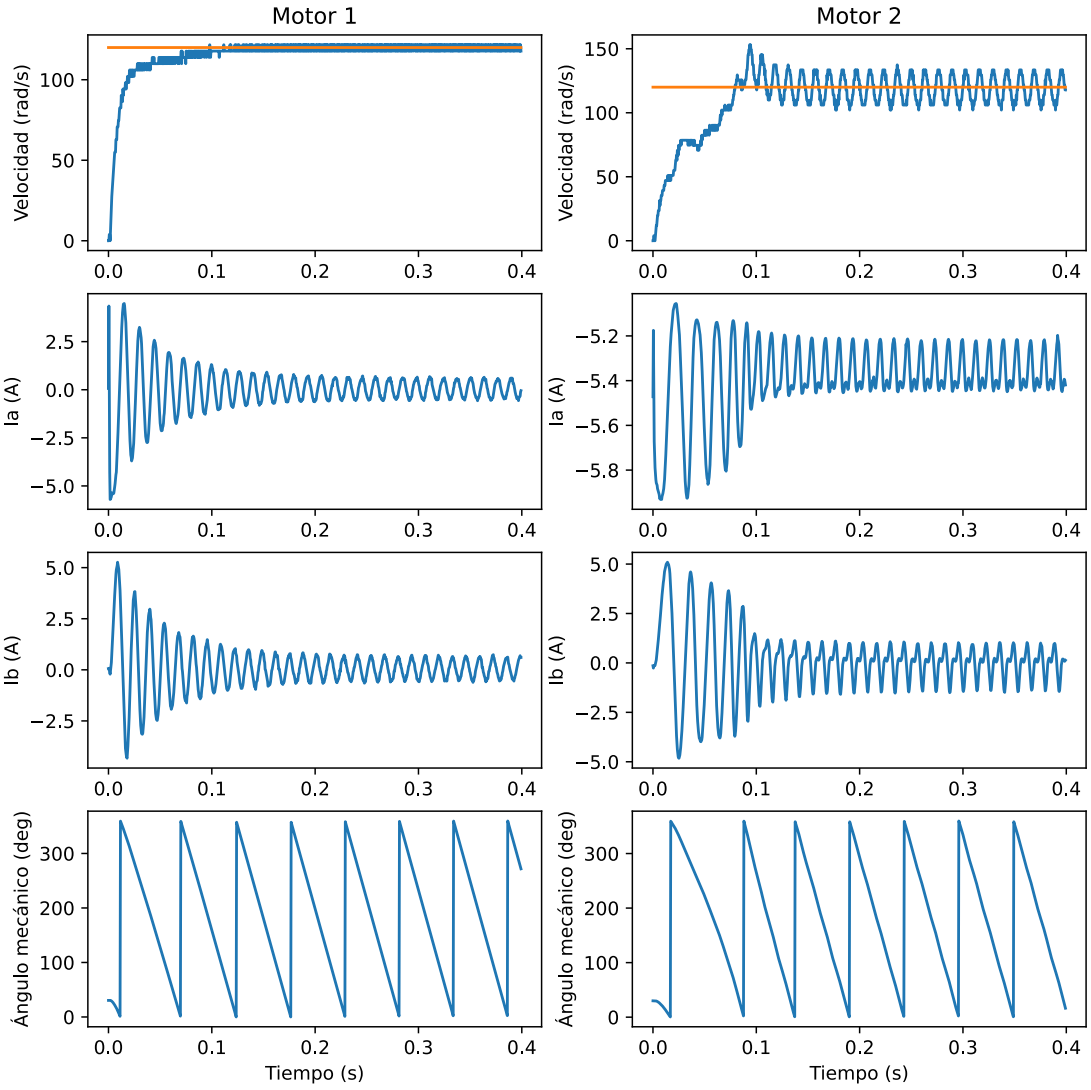


Figura 29: Escalón a 120 rad/s con dos motores controlados en serie

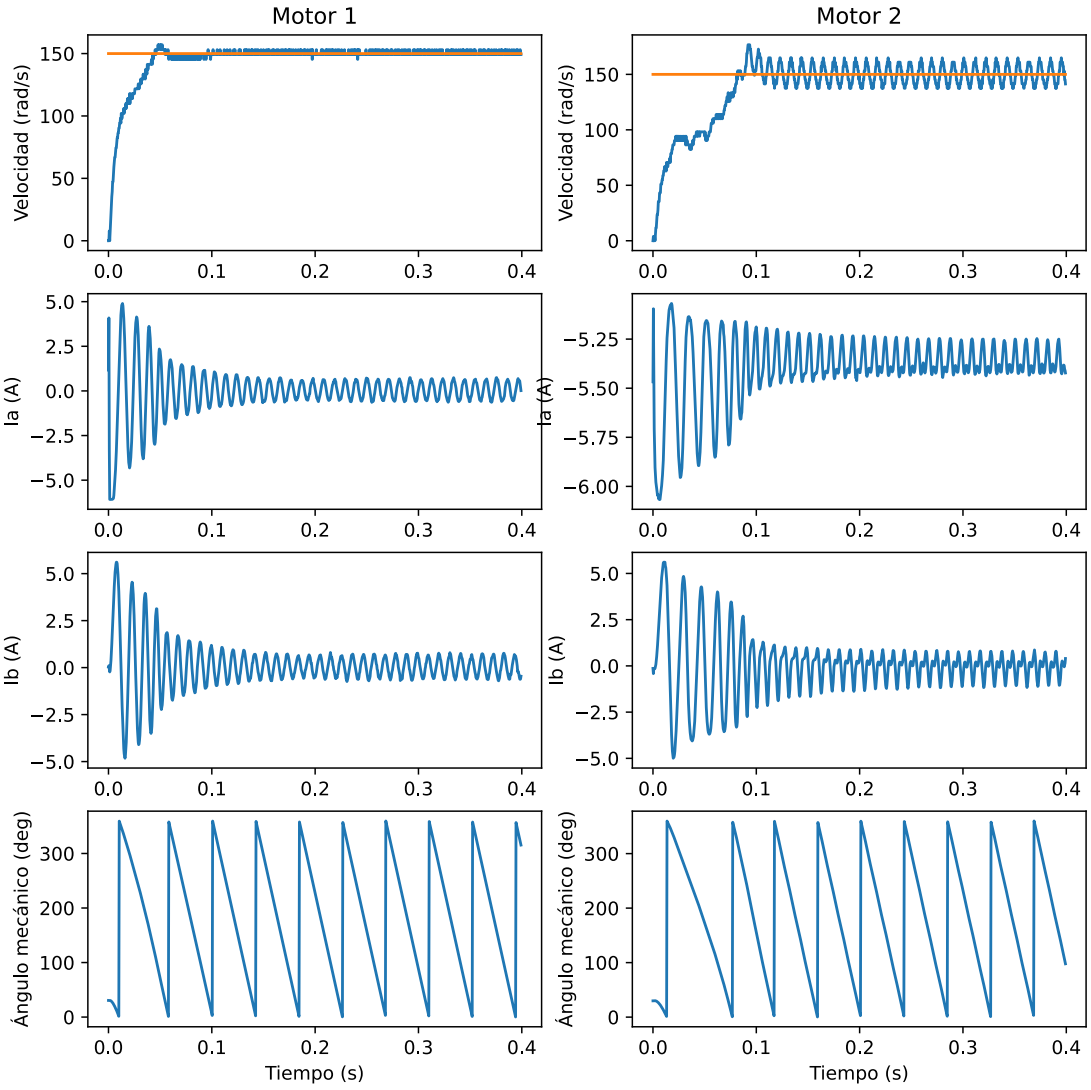


Figura 30: Escalón a 150 rad/s con dos motores controlados en serie

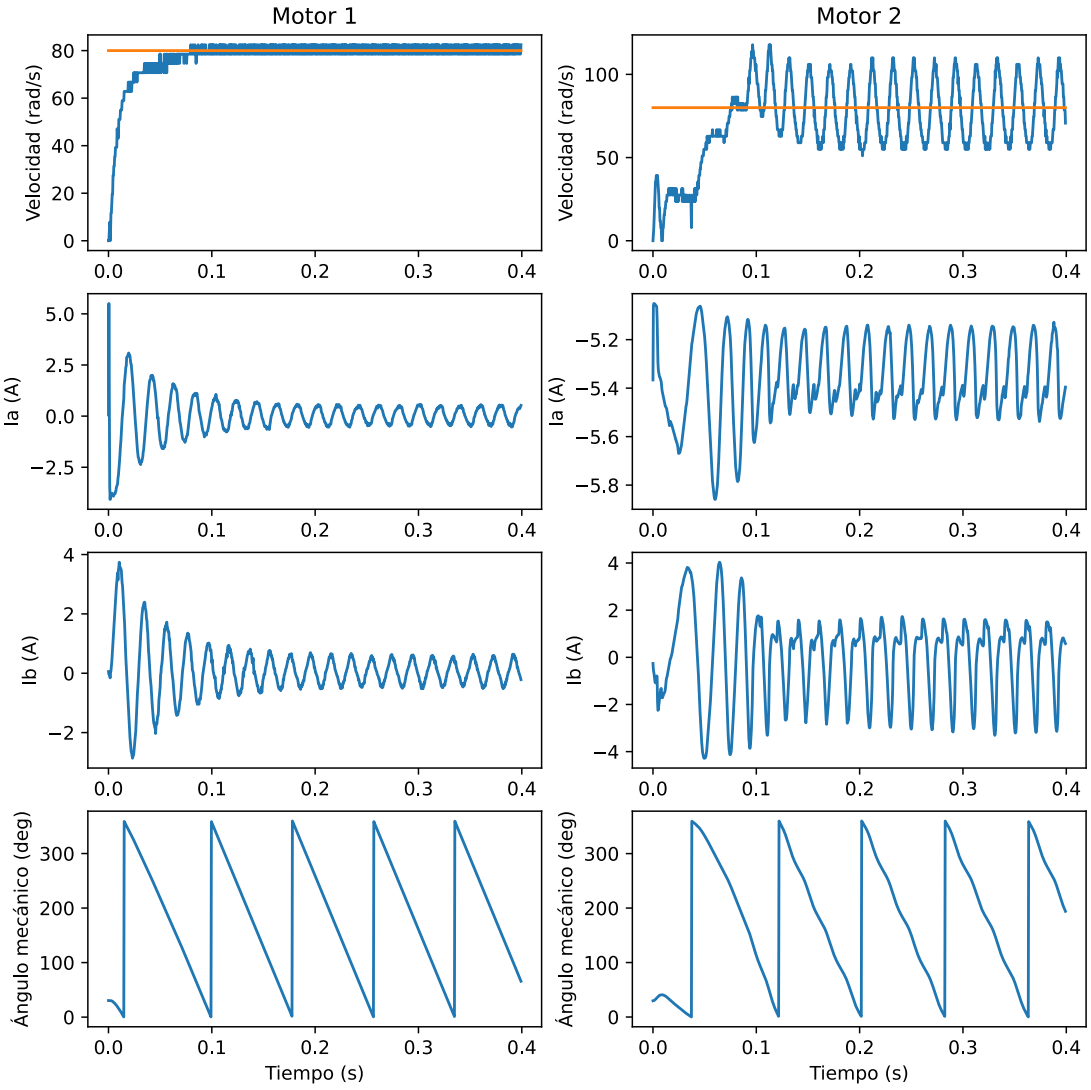


Figura 31: Escalón a 80 rad/s con dos motores

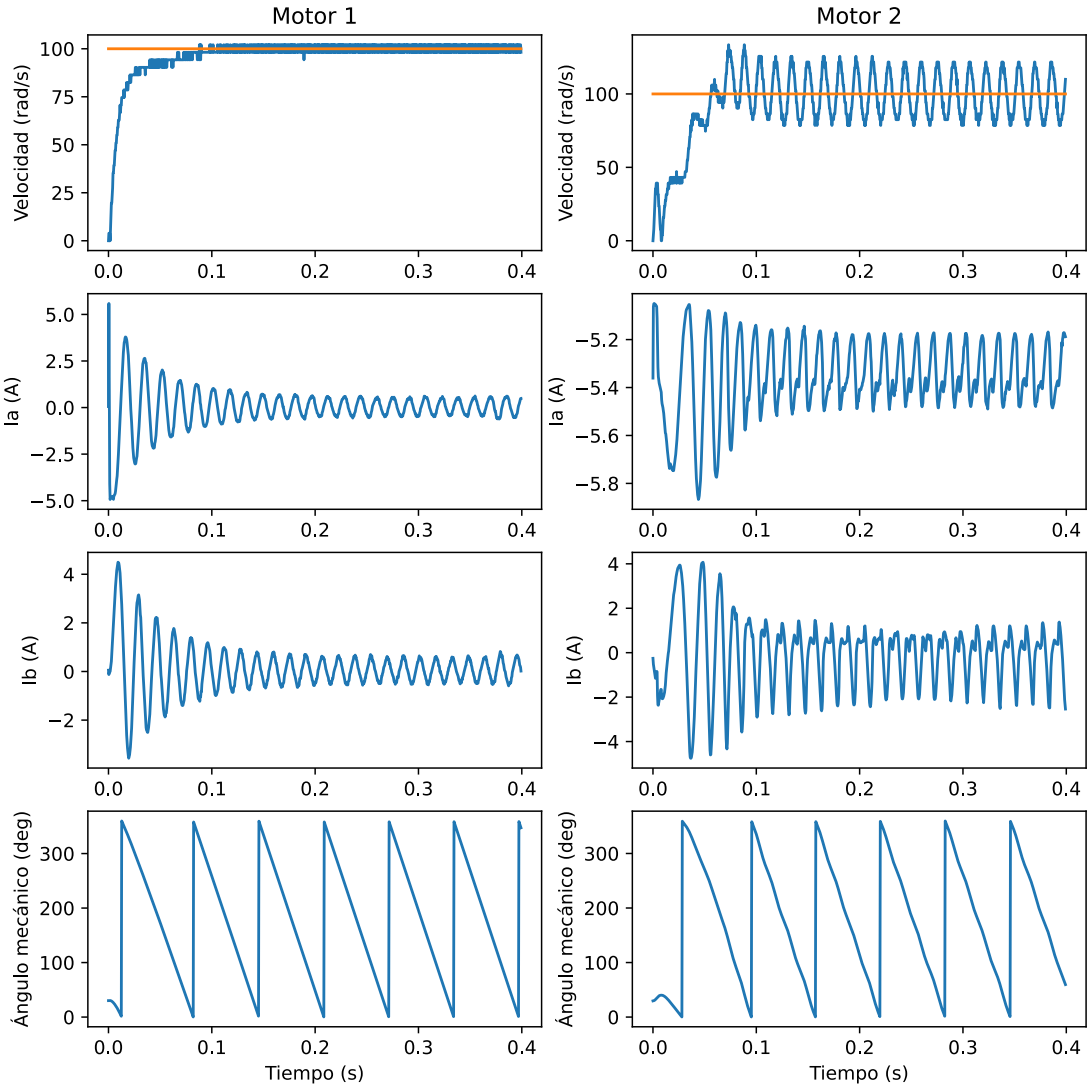


Figura 32: Escalón a 100 rad/s con dos motores

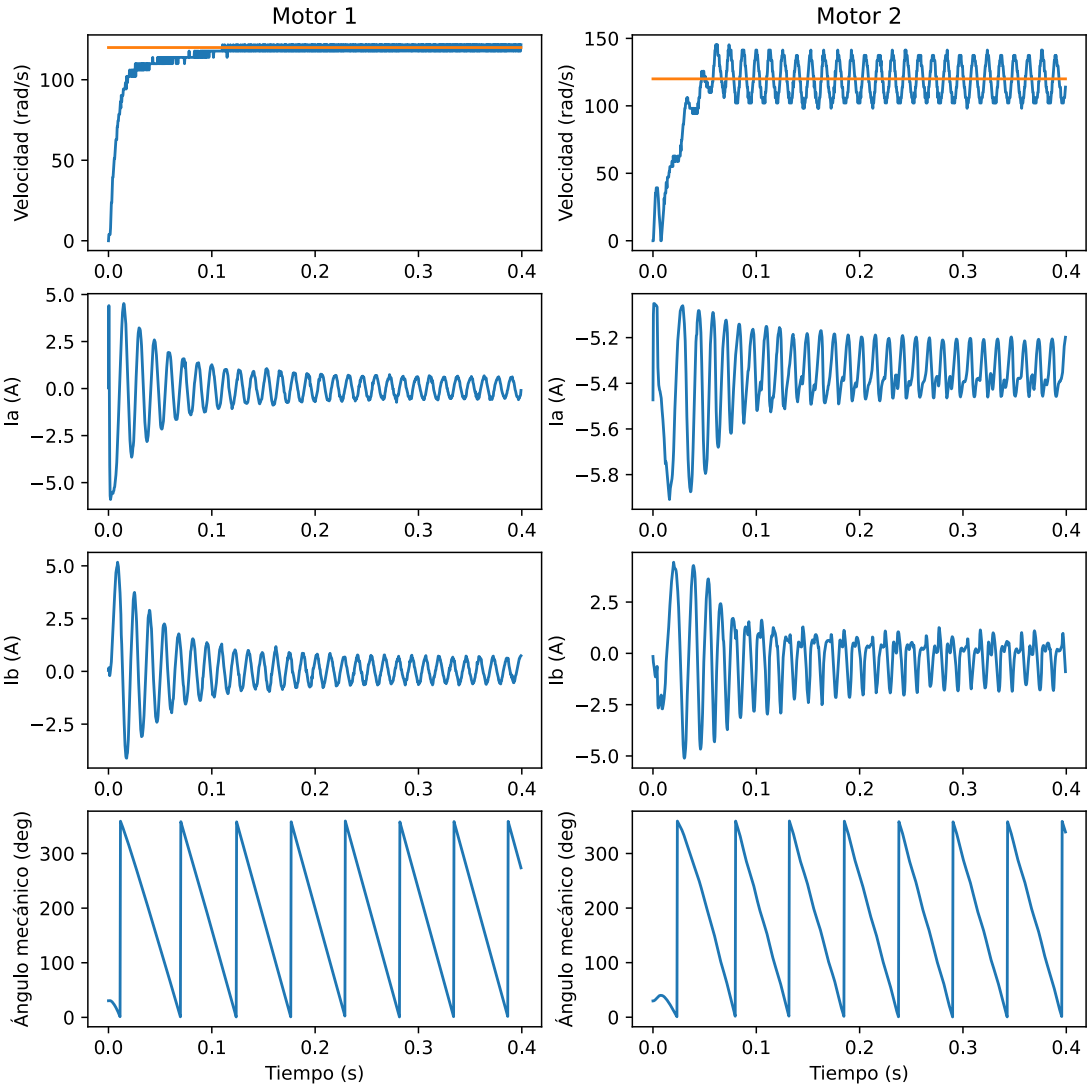


Figura 33: Escalón a 120 rad/s con dos motores

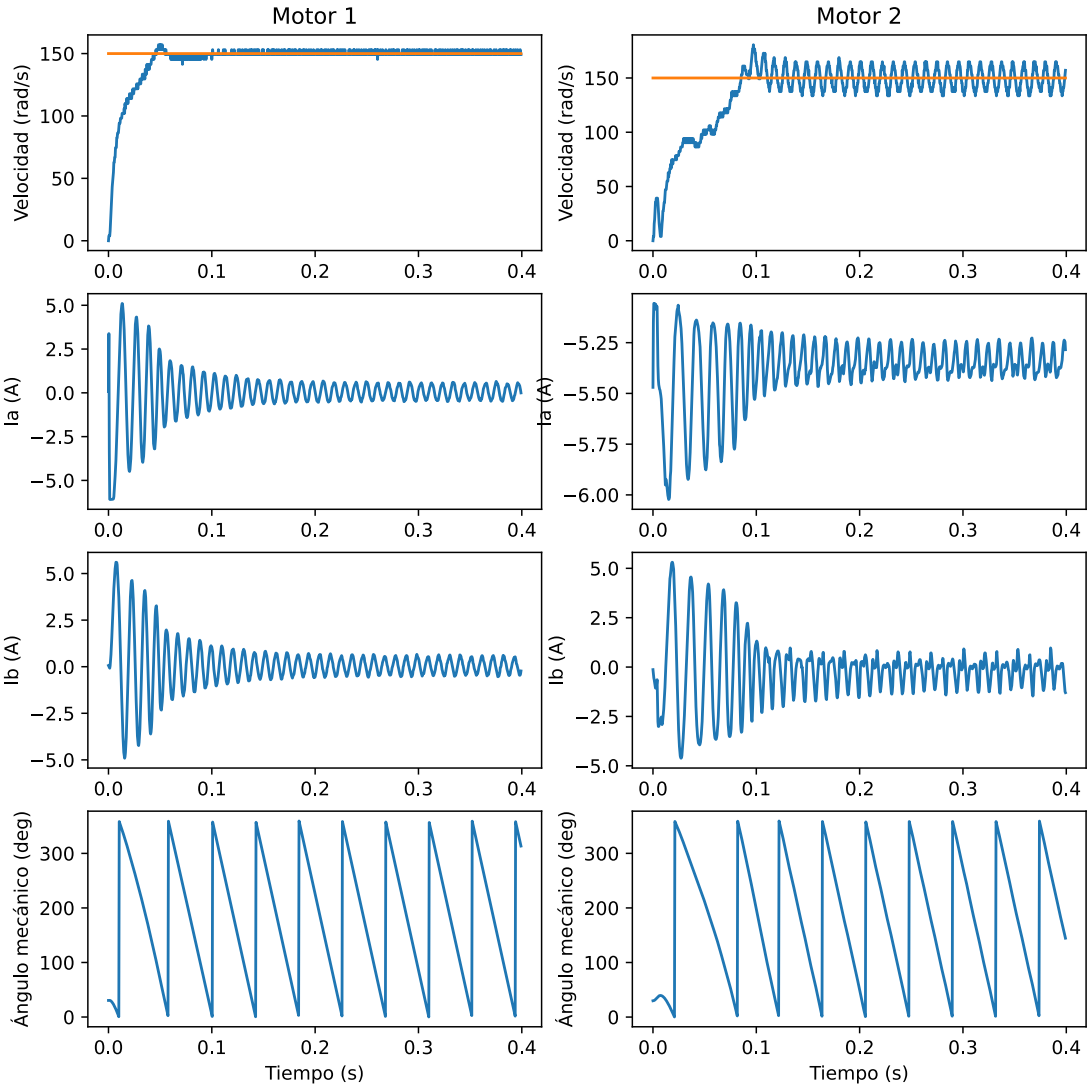


Figura 34: Escalón a 150 rad/s con dos motores

5. Conclusiones

Una vez analizados los resultados de validación, se observa que el comportamiento del demostrador descrito en la Figura 4 difiere de la respuesta obtenida en la implementación previa. Sin embargo, cabe destacar que, tanto la implementación realizada para un motor como el control paralelo de los dos motores cumplen con los tiempos de ejecución requeridos y son implementaciones funcionales de un control de velocidad.

En el caso del control del segundo motor, la implementación requiere todavía trabajo para diagnosticar la causa del desvío y corregirlo hasta el punto de obtener un control robusto, el control del primer motor, sin embargo, ofrece una respuesta razonable.

En lo referente a RISC-V como arquitectura de control de motores, se puede afirmar que es una alternativa viable, si bien se requiere de un ecosistema más robusto a su alrededor que facilite este tipo de implementaciones. Como se alude en Subsección 3.3.1, la disponibilidad de ASICs RISC-V es todavía escasa, y, entre la oferta multi-core disponible, el tipo de periféricos que implementan estos microcontroladores no se adecúa a este tipo de aplicaciones.

En una nota más general, en el momento de seleccionar un microcontrolador para una aplicación industrial, las funcionalidades no son el único criterio a tener en cuenta. Es imprescindible contar con:

1. **Disponibilidad a largo plazo:** Uno de los problemas que se han afrontado en este trabajo es la falta de disponibilidad de algunas de las tarjetas de desarrollo consideradas. A pesar de que las circunstancias particulares de los últimos años hayan provocado que muchas compañías de diseño de microcontroladores reduzcan su oferta a las gamas de productos más vendidas o más lucrativas, no se puede ignorar que para optar por un producto en un diseño que potencialmente pueda estar años en el mercado, hay que tener en cuenta el historial de los fabricantes a la hora de mantener el soporte de sus productos.

2. **Ecosistema de desarrollo:** Más allá de los propios microcontroladores, la existencia de entornos de desarrollo, depuración y emulación adaptados a la plataforma, así como librerías optimizadas y capas de abstracción del hardware son elementos intrínsecos al desarrollo que posibilitan el diagnóstico y la predicción del comportamiento de los sistemas creados en dicha plataforma. En ausencia de un ecosistema de desarrollo maduro la labor de diseño y programación se puede dificultar hasta el punto de llegar a ser inviable.

3. **Documentación:** A la hora de seleccionar un microcontrolador es necesario poder conocer su funcionamiento a tan bajo nivel como la aplicación a la que está destinado requiera. Esto impone la necesidad por parte de los fabricantes de ofrecer documentación detallada respecto a sus productos. A lo largo de la fase de

búsqueda de este proyecto, la ausencia de documentación accesible, actualizada o en inglés han sido contratiempos que han dificultado enormemente el poder determinar si un producto era adecuado para las necesidades de la aplicación.

Es en estos aspectos donde el ecosistema de RISC-V tiene todavía mucho desarrollo por delante antes de convertirse en un competidor real para ARM. Sin embargo, muchos de estos factores vienen con el tiempo. La presencia de firmas consolidadas en el mercado de los microcontroladores entre los miembros del consorcio RISC-V parecen indicar un respaldo de la industria que podría fomentar que este tipo de prácticas se vuelvan más comunes en el futuro; en cualquier caso, para que RISC-V evolucione del contexto académico e investigador y compita al mismo nivel que ARM, es necesario que los fabricantes de microcontroladores RISC-V adopten estas prácticas de la industria.

Bibliografía

[1] Michael Bluhm. *Biden's hugely consequential high-tech export ban on China, explained by an expert*. Nov. de 2022. URL: https://www.vox.com/world/2022/11/5/23440525/biden-administration-semiconductor-export-ban-china.

[2] Matthew Humphries. *Following US Sanctions, China Decides Its Future Lies With RISC Chips*. Dic. de 2022. URL: https://www.pcmag.com/news/following-us-sanctions-china-decides-its-future-lies-with-risc-chips.

[3] Jake Hertz. *Alibaba RISC-V SoC Revealed as Processor for First RISC-V Laptop*. Oct. de 2022. URL: https://www.allaboutcircuits.com/news/alibaba-risc-v-soc-revealed-as-processor-for-first-risc-v-laptop/.

[4] Dylan Martin. *Alibaba, Tencent enlisted to help sanction-weary China build RISC-V chips*. 2022. URL: https://www.theregister.com/2022/12/01/alibaba_tencent_china_riscv/.

[5] Agam Shah. *Europe to Dish out €270 Million to Build RISC-V Hardware and Software*. 2022. URL: https://www.hpcwire.com/2022/12/16/europe-to-dish-out-e270-million-to-build-risc-v-hardware-and-software/.

[6] Zohaib Ali et al. «Reassessing the Performance of ARM vs x86 with Recent Technological Shift of Apple». En: *2022 International Conference on IT and Industrial Technologies (ICIT)*. 2022, págs. 01-06. DOI: 10.1109/ICIT56493.2022.9988933.

[7] Ming Ling et al. «Does the ISA Really Matter? A Simulation Based Investigation». En: *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. 2019, págs. 1-6. DOI: 10.1109/PACRIM47961.2019.8985059.

[8] Emily Blem, Jaikrishnan Menon y Karthikeyan Sankaralingam. «Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures». En: *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 2013, págs. 1-12. DOI: 10.1109/HPCA.2013.6522302.

[9] A.D. George. «An overview of RISC vs. CISC». En: *[1990] Proceedings. The Twenty-Second Southeastern Symposium on System Theory*. 1990, págs. 436-438. DOI: 10.1109/SSST.1990.138185.

[10] Intel. *Envisioning a Simplified Intel Architecture*. Inf. téc. 2023. URL: https://www.intel.com/content/www/us/en/developer/articles/technical/envisioning-future-simplified-architecture.html.

[11] ARM. *Cortex-M3 Devices Generic User Guide*. URL: https://developer.arm.com/documentation/dui0552/a/the-cortex-m3-processor/programmers-model/core-registers.

[12] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA*. Inf. téc. UCB/EECS-2011-62. EECS Department, University of California, Berkeley, mayo de 2011. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-62.html.

[13] Semico Research. *RISC-V Market Analysis: The New Kid on the Block*. 2019. URL: https://semico.com/content/risc-v-market-analysis-new-kid-block.

[14] SiFive. *SiFive product lineup*. URL: https://www.sifive.com/risc-v-core-ip.

[15] OpenHW Group. *Open HW Group GitHub*. URL: https://github.com/openhwgroup.

[16] Andes Tech. *Andes Tech core lineup*. URL: http://www.andestech.com/en/products-solutions/andescore-processors/.

[17] Stijn Derammelaere et al. «A quantitative comparison between BLDC, PMSM, brushed DC and stepping motor technologies». En: *2016 19th International Conference on Electrical Machines and Systems (ICEMS)*. 2016, págs. 1-5. URL: https://ieeexplore.ieee.org/document/7837471.

[18] David Ocen. «Direct Torque Control of a Permanent Magnet Synchronous Motor». Tesis de mtría. KTH Royal Institute of Technology, 2005. URL: http://www.diva-portal.org/smash/get/diva2:582454/FULLTEXT01.pdf.

[19] Fengxiang Wang et al. «Advanced Control Strategies of Induction Machine: Field Oriented Control, Direct Torque Control and Model Predictive Control». En: *Energies* 11.1 (2018). ISSN: 1996-1073. DOI: 10.3390/en11010120. URL: https://www.mdpi.com/1996-1073/11/1/120.

[20] Foram Patel, Jignesha Ahir y Priyanka Patel. «Control Theory For Permanent Magnet Synchronous Motor – A Review». En: *Asian Journal For Convergence In Technology (AJCT) ISSN -2350-1146* (abr. de 2019). URL: https://asianssr.org/index.php/ajct/article/view/736.

[21] *Park, Inverse ParkandClarke, Inverse ClarkeTransformations MSS Software Implementation*. Microsemi Corporation. 2013. URL: https://www.microsemi.com/document-portal/doc_view/132799-park-inverse-park-and-clarke-inverse-clarke-transformations-mss-software-implementation-user-guide.

6. Descripción de tareas

Con el fin de afrontar el proyecto, se establecen las tareas descritas en la Tabla 17 y se planifican en el tiempo como se establece en la Figura 35. En el diagrama de Gantt se establece la planificación inicial y el desarrollo real. Las tareas que muestran un tiempo final pero no uno previsto son aquellas en las que la previsión inicial se ha cumplido.

Tabla 17: Descripción de tareas.

Clave	Nombre de la tarea	Descripción de la tarea
T1	Estudio del trabajo previo.	Estudio de la documentación previa, así como de los datasheets de los componentes que constituyen el demostrador.
T2	Estado del arte de plataformas RISC-V y de control de motores.	Análisis del estado del arte de las arquitecturas de microcontroladores más utilizadas en control de motores.
T3	Análisis del código desplegado.	Análisis del funcionamiento del código del proyecto previo y puesta en marcha de este.
T4	Búsqueda de plataforma RISC-V compatible.	Análisis de los requerimientos de la aplicación prevista y búsqueda de una plataforma (placa de desarrollo) basada en RISC-V que cumpla con ellos.
T5	Pruebas con la plataforma escogida.	Tras adquirir la plataforma, realización de pruebas de programación básica como toma de contacto con el SDK.
T6	Porting del código existente a la nueva plataforma.	Se parte de la implementación previa y se traslada la funcionalidad programada al nuevo sistema teniendo en cuenta las diferencias entre las plataformas.
T7	Diseño del sistema tras añadir el segundo motor.	Se modifica el nuevo sistema para introducir el control del segundo motor.
T8	Verificación del funcionamiento del sistema diseñado.	Se comprueba que la integración de los distintos componentes del sistema haya sido exitosa y que el funcionamiento es el esperado.
T9	Definir la metodología para evaluar el rendimiento.	Decidir cómo se va a evaluar el rendimiento de la plataforma y qué pruebas se van a realizar.
T10	Realización de pruebas.	Realización de las pruebas definidas y obtención de resultados.
T11	Análisis de los datos obtenidos.	Una vez se dispone de los resultados, realización de un análisis con el fin de obtener conclusiones.
T12	Redacción de la memoria.	Redacción del presente documento.

Siguiendo el esquema descrito en la Tabla 17, se plantea el siguiente cronograma para la consecución del proyecto:

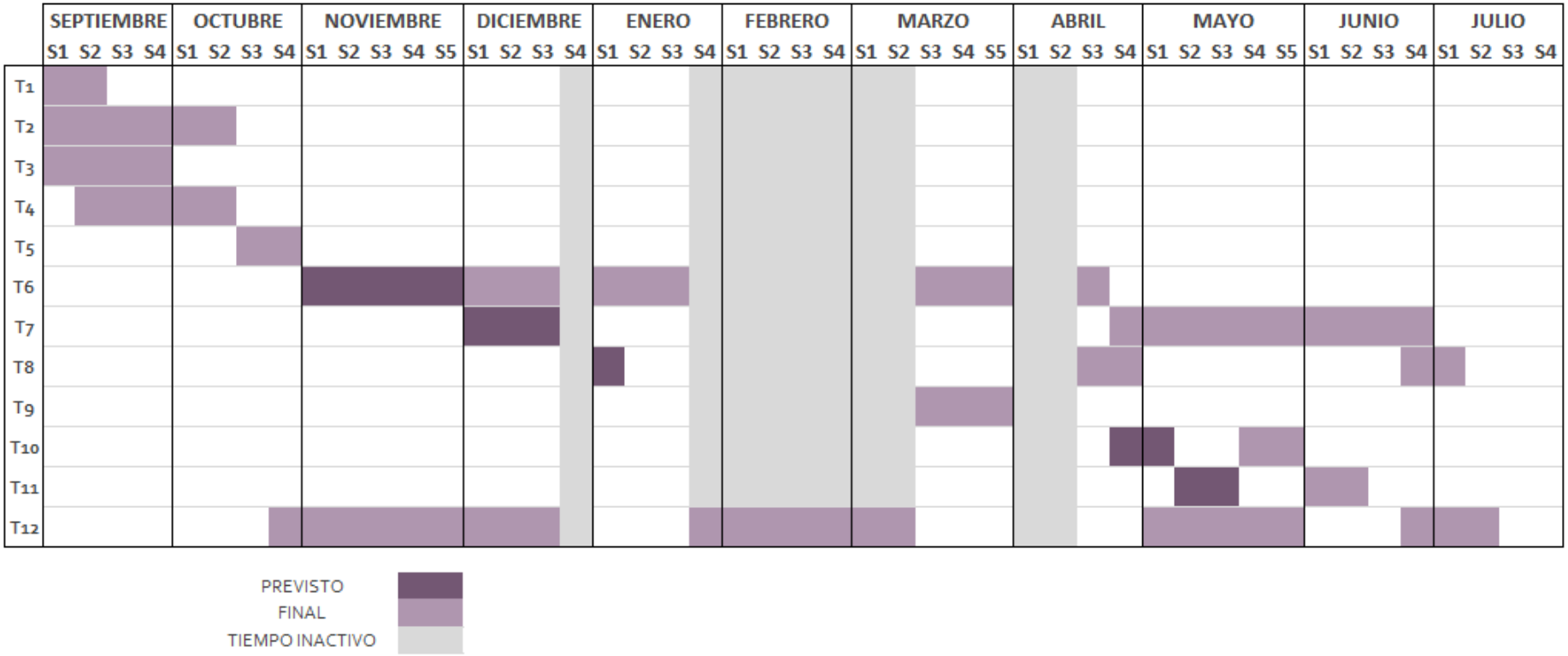
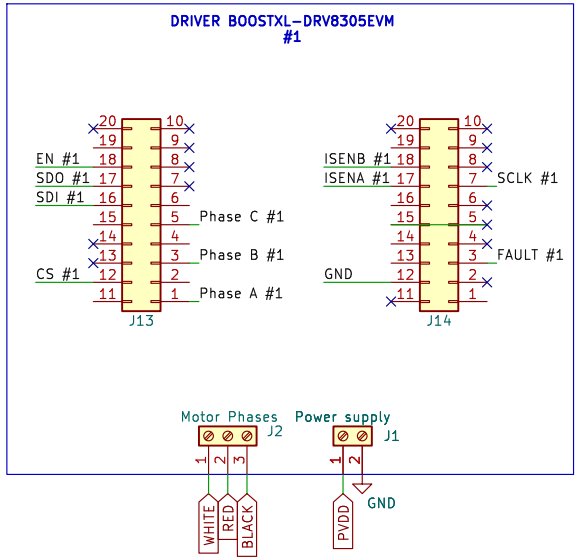
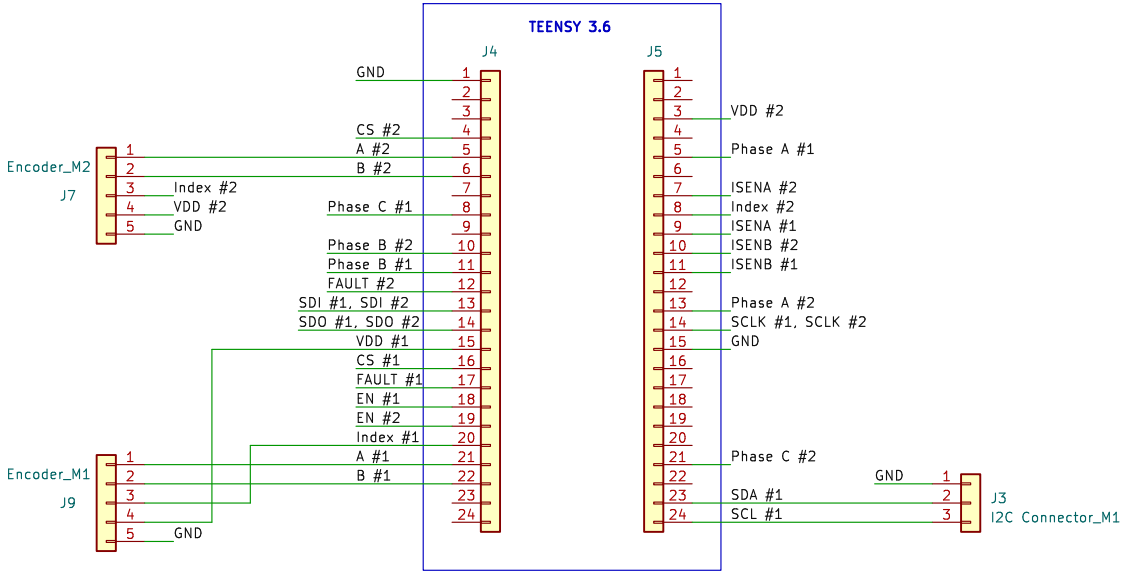
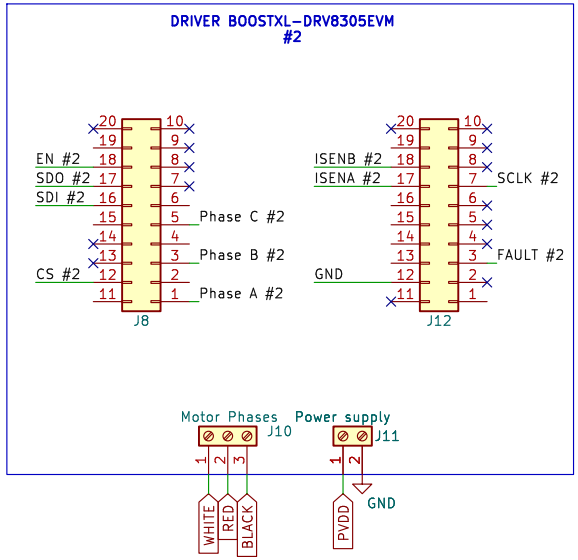


Figura 35: Cronograma de tareas

7. Anexo I: Esquemático del prototipo







Author: Jaime Zaballa Orduna
IKERLAN S. COOP.

Sheet: /
File: B2B_schematic_clean.kicad_sch

Title: Back2Back connector board

Size: A4	Date: 2023-07-02	**Rev: 5**
KiCad E.D.A. eeschema 7.0.2		Id: 1/1