



UNR Universidad
Nacional de Rosario



Escuela de Ingeniería Electrónica

PROYECTO FINAL DE INGENIERIA

“Desarrollo e implementación de funciones de navegación autónoma para un prototipo de robot pulverizador”

INTEGRANTES:

GIACOMINO, Gonzalo (G-4898/4)

IMSAND, Juan Facundo (I-0553/3)

MARTINEZ, Sebastián (M-6098/4)

DIRECTOR:

MIGONI, Gustavo

06/02/2020

Resumen

En este proyecto final de ingeniería se trabajó sobre un prototipo de robot pulverizador autónomo diseñado para el control de malezas en cultivos de soja.

Como objetivo principal se planteó el desarrollo de un algoritmo de seguimiento de trayectoria y su implementación sobre el robot. Para encarar este objetivo, previamente se debió resolver el sistema de tracción y dirección.

El proyecto transita desde el análisis teórico de funcionamientos y operación de equipos.

En el presente documento se detallan las tareas realizadas para lograr los objetivos propuestos, las problemáticas encontradas y finalmente los resultados obtenidos con sus respectivas conclusiones.

Índice:

1. Introducción	1
1.1. Tecnología agropecuaria	1
1.2. El prototipo.....	2
2. El Proyecto.....	4
3. Marco Teórico.....	6
3.1. Sistema electromecánico de tracción y dirección	6
3.1.1. Sistema de tracción.....	6
3.1.2. Sistema de dirección.....	14
3.2. Sistema de navegación autónoma	22
3.2.1. Modelo de Ackermann	24
3.2.2. Sistema de ubicación espacial basado en odometría	25
3.2.3. Control de velocidad y dirección del vehículo	27
3.2.4. Path Tracking.....	33
4. Equipamiento utilizado	38
4.1. Rueda y controlador	38
4.2. Microcontrolador.....	39
4.3. Mando a distancia.....	41
4.4. Sensores de Corriente.....	42
4.5. Stepper Motor.....	44
4.6. Driver Stepper Motor	44
4.7. Encoder Absoluto.....	47
4.8. Computadora utilizada	47
4.9. Contenedores y soportes.....	48
4.10. Gabinete	48
4.11. Fichas	48
5. Tareas desarrolladas.....	50
5.1. Relevamiento de la distribución física del sistema	50
5.2. Esquema de control	52
5.2.1. Motor	52
5.2.2. Drivers	53
5.2.3. Filtro de Señal HALL	55
5.2.4. Sensores de efecto HALL.....	57
5.2.5. Sensor de corriente	57

5.2.6. Filtros PWM	60
5.2.7. Estima de la velocidad tangencial y distancia recorrida	62
5.2.8. Controlador	64
5.2.9. Conexión de componentes del sistema de control	67
5.2.10. Elección del microprocesador	70
5.2.11. Interfaces de Adaptación	71
5.3. Diseño de placa electrónica	72
5.3.1. Diseño de placa electrónica universal	73
5.3.2. Diseño de placa electrónica PCB	78
5.4. Prestaciones del sistema	81
5.4.1. Relevamiento y actualización del software de control	81
5.4.2. Comunicación y Modos de funcionamiento	82
5.4.3. Relevamiento de las funciones del software	84
5.5. Navegación autónoma	94
5.5.1. Elección de parámetros de las leyes de control	95
5.5.2. Simulación	97
5.5.3. Implementación de software	104
6. Pruebas de Campo	119
6.1. Control de tracción	119
6.1.1. Controlador	119
6.1.2. Actuación de freno	120
6.1.3. Corriente consumida	121
6.1.4. Pruebas en terreno	122
6.2. Ensayos de navegación autónoma	124
Prueba 1 (Circunferencias)	125
Prueba 2 (Recta y curvas)	126
Prueba 3 (Símbolo ∞)	127
6.3. Fuentes de error en la implementación	128
7. Conclusiones	131
8. Bibliografía	132
9. Apéndice	133
9.1.1. Resultados finales	135

Figuras:

Figura 1.1: Foto del robot pulverizador.	3
Figura 3.1: Esquema de un motor brushless DC in-wheel. (a) Monofásico - (b) Trifásico.....	7
Figura 3.2: Fuerza contra electromotriz (Back EMF). (a) Trapezoidal - (b) Senoidal.	8
Figura 3.3: Tipos de Rotores. (a) Polos salientes - (b) Embebidos - (c) Insertados.	8
Figura 3.4: Disposición de los principales componentes de un BLDC.	9
Figura 3.5: (a) Diagrama temporal de una secuencia - (b) Secuencia en las bobinas.....	10
Figura 3.6: Inversores para controlar un BLDC. (a) Monofásico - (b) Trifásico.	11
Figura 3.7: Modelo eléctrico de un motor BLDC trifásico.....	12
Figura 3.8: Cuadrantes de operación de un BLDC.	13
Figura 3.9: Sistema de dirección del prototipo.	14
Figura 3.10: Esquema de un motor paso a paso (PaP).....	15
Figura 3.11: (a) PaP unipolar - (b) PaP bipolar.	15
Figura 3.12: Secuencia de pasos de PaP bipolar de 2 pares de polos.	16
Figura 3.13: Simple Voltage Driver de un motor paso a paso bipolar.	17
Figura 3.14: Driver L/R.	18
Figura 3.15: Tensión de bobinado igual a V_{CC} (Chopper Drivers).....	19
Figura 3.16: Recirculación de corriente (Chopper Drivers).	19
Figura 3.17: Forma de onda de corriente de fase característica (Chopper Drivers).	20
Figura 3.18: Corriente de fase método Full Step - Motor Paso a Paso.	21
Figura 3.19: Corriente de fase método Half Step - Motor Paso a Paso.	21
Figura 3.20: Formas de onda de corriente de fase - Microstepping.....	22
Figura 3.21: Esquema simplificado de la comunicación entre la NUC y el microcontrolador.	23
Figura 3.22: Modelo del biciclo de Ackermann.	24
Figura 3.23: Modelo del biciclo de Ackermann (rosa) impuesto sobre el cuatriciclo (gris).	24
Figura 3.24: Zona restringida del vehículo. (a) Posición (0,0,0) - (b) Posición (5,5, $\pi/6$).....	30
Figura 3.25: Robot orientado hacia el punto deseado.....	31
Figura 3.26: Robot orientado en dirección opuesta al punto deseado.	31
Figura 3.27: Dos trayectorias posibles para un mismo punto objetivo.....	32
Figura 3.28: Ejemplo de una trayectoria errónea en campo cultivado.	33
Figura 3.29: Secuencia de puntos objetivo en "Path Tracking" (Parte A).	34
Figura 3.30: Secuencia de puntos objetivo en "Path Tracking" (Parte B).	34
Figura 3.31: Ley de control brusca que produce oscilación.	36
Figura 3.32: Ley de control demasiado suave.	36
Figura 4.1: Kit e-bike adquirido.....	38
Figura 4.2: Principales componentes K64F.	39
Figura 4.3: Pines de Entrada/Salida de K64F.	40
Figura 4.4: Mando a distancia utilizado.....	41
Figura 4.5: Sensor ACS712 de 20A.....	42
Figura 4.6: Hoja de datos del Motor paso a paso comprado.....	44
Figura 4.7: Stepper Motor Driver M542H.	45
Figura 4.8: Esquema de conexión.	46
Figura 4.9: MINI-PC intel® NUC.	47

Figura 4.10: Gabinete GENROD serie S9000.	48
Figura 4.11: Ficha hembra de 8 vías.	49
Figura 4.12: Ficha macho de 8 vías.	49
Figura 5.1: Esquema del sistema de control.	50
Figura 5.2: Lazo de control detallado.	52
Figura 5.3: Motor BLDC.	53
Figura 5.4: Driver motor BLDC.	54
Figura 5.5: Filtro de la señal de un sensor de efecto HALL.	56
Figura 5.6: Gráfica temporal de la misma señal HALL antes y después del filtro (motor en vacío).	56
Figura 5.7: Señales de un sensor de efecto HALL con el motor en movimiento.	57
Figura 5.8: Esquema del montaje del sensor de corriente.	58
Figura 5.9: Circuito completo del sensor de corriente.	59
Figura 5.10: Tensión de salida del sensor (motor en vacío).	59
Figura 5.11: Tensión de salida vs Corriente sensada.	60
Figura 5.12: Filtro para las señales PWM.	61
Figura 5.13: Entrada (rojo) y salida (azul).	62
Figura 5.14: Entrada (rojo) y salida (azul).	62
Figura 5.15: Respuesta al escalón de un sistema de primer orden con retardo.	64
Figura 5.16: Tabla de constantes del controlador del método Ziegler - Nicholls.	64
Figura 5.17: Lazo de control implementado.	65
Figura 5.18: Velocidad [RPM] vs Tensión de manillar [V].	66
Figura 5.19: Respuesta del sistema frente a distintos escalones en la tensión de manillar.	67
Figura 5.20: Diagrama de conexión de componentes del sistema de control.	69
Figura 5.21: Interfaz de adaptación.	72
Figura 5.22: Vi (azul) y Vo (rojo).	72
Figura 5.23: Foto de la placa universal ya conectada y funcionando dentro del gabinete.	73
Figura 5.24: Esquema simplificado de placa electrónica universal (entradas y salidas) junto a receptor de radio frecuencia del mando a distancia.	74
Figura 5.25: Diseño final del PCB.	79
Figura 5.26: PCB visto de frente.	80
Figura 5.27: PCB visto desde arriba.	80
Figura 5.28: Módulo OpenSDA incluido en la placa.	82
Figura 5.29: Diagrama de flujo de la función main(void).	89
Figura 5.30: Diagrama de flujo de la función EMERGENCY.	90
Figura 5.31: Diagrama de flujo de la rutina de interrupción del ADC.	92
Figura 5.32: Diagrama de flujo de la función GetCorriente.	92
Figura 5.33: Rutina de la función CalibracionCorriente.	93
Figura 5.34: Trayectoria del robot al darle como objetivo el punto (0,4.722) con orientación π	96
Figura 5.35: Diagrama de bloques del sistema completo.	97
Figura 5.36: Esquema interno del bloque "Robot".	98
Figura 5.37: Cálculo de las coordenadas de waypoint.	99
Figura 5.38: Gráfica y vs x de la trayectoria 1.	100
Figura 5.39: v(t) para la trayectoria 1.	101
Figura 5.40: $\phi(t)$ para la trayectoria 1.	102

Figura 5.41: Gráfica y vs x para la trayectoria 2.....	103
Figura 5.42: $v(t)$ para la trayectoria 2.	104
Figura 5.43: $\phi(t)$ para la trayectoria 2.....	104
Figura 5.44: Diagrama de flujo principal.....	106
Figura 5.45: Diagrama de flujo de la función main (Parte 1).	107
Figura 5.46: Diagrama de flujo de la función main (Parte 2).	108
Figura 5.47: Diagrama de flujo de la función main (Parte 3).	109
Figura 5.48: Diagrama de flujo de la función calc_control (Parte 1).	110
Figura 5.49: Diagrama de flujo de la función calc_control (Parte 2).	111
Figura 5.50: Diagrama de flujo de la función calc_robot.	112
Figura 5.51: Diagrama de flujo de la función integral.....	113
Figura 5.52: Gráfica ejemplo del método de integración discreto utilizado.....	113
Figura 5.53: Diagrama de flujo de la función gen_trama.	114
Figura 5.54: Diagrama de flujo de la función archivo.	115
Figura 5.55: Diagrama de flujo de la función cx_rx.	116
Figura 5.56: Diagrama de flujo de la función cx_tx.	116
Figura 5.57: Diagrama de flujo de la función descifra_trama (Parte 1).	117
Figura 5.58: Diagrama de flujo de la función descifra_trama (Parte 2).	118
Figura 6.1: Respuesta de los motores frente a diferentes Set-Point de velocidad.	119
Figura 6.2: Comparación de actuación de freno a lazo abierto.....	120
Figura 6.3: Comparación de actuación de freno a lazo cerrado.....	121
Figura 6.4: Respuesta de los motores sin actuar el freno.....	122
Figura 6.5: Respuesta de los motores aplicando el freno.....	122
Figura 6.6: Respuesta de los motores en terreno regular.	123
Figura 6.7: Respuesta de los motores en un terreno irregular.....	123
Figura 6.8: Gráfica y vs x de la prueba 1.....	125
Figura 6.9: Gráfica y vs x de la prueba 2.....	127
Figura 6.10: Gráfica y vs x de la prueba 3.....	128
Figura 6.11: Trayectoria en línea recta.	129
Figura 9.1: Caja contenedora para dos drivers.....	133
Figura 9.2: Tapa para dos drivers.	133
Figura 9.3: Caja contenedora para un driver.....	134
Figura 9.4: Tapa para un driver.....	134
Figura 9.5: Soporte para sensor de corriente.	134
Figura 9.6: Foto de caja contenedora de dos drivers.	135
Figura 9.7: Foto caja contenedora de dos drivers vista desde arriba.	135
Figura 9.8: Foto de tapa de dos drivers.....	136
Figura 9.9: Foto caja contenedora de un driver.	136
Figura 9.10: Tapa de un driver.....	137
Figura 9.11: Cajas y drivers montados sobre el gabinete.	137

Tablas:

Tabla 3.1: Torque de sostén vs método Microsteps.....	22
Tabla 4.1: Características eléctricas del driver ($T_j=25^{\circ}\text{C}$).....	46
Tabla 4.2: Señales de control del driver.....	47
Tabla 5.1: Entradas/Salidas requeridas en el microprocesador.	70
Tabla 5.2: Descripción de entradas y salidas de la placa electrónica universal.....	75
Tabla 5.3: Descripción de pines utilizados del microprocesador.	77
Tabla 5.4: Rutinas principales del código.....	86
Tabla 5.5: Rutinas de Interrupciones del código.	87
Tabla 5.6: Otras rutinas de interés.	87
Tabla 5.7: Puntos objetivos de la trayectoria 1.	100
Tabla 5.8: Puntos objetivos de la trayectoria 2.	103
Tabla 5.9: Bibliotecas, defines, estructuras y funciones incluidos en el software de tracking.....	106
Tabla 6.1: Puntos objetivo de la prueba 1.....	125
Tabla 6.2: Puntos objetivo de la prueba 2.....	126
Tabla 6.3: Puntos objetivo de la prueba 3.....	127

1. Introducción

1.1. Tecnología agropecuaria

El concepto de tecnología agropecuaria, de este modo, hace mención a los conocimientos, las técnicas y los artefactos que permiten la utilización de elementos tecnológicos en las tareas agrícolas y ganaderas. La tecnología agropecuaria incluye desde distintos tipos de máquinas hasta trabajos de laboratorio que permiten incrementar la eficiencia de estas actividades.

El acelerado crecimiento de la población en el mundo exige una producción suficiente de alimentos en cantidad y calidad, a bajo costo y con el menor impacto ambiental posible.

Cuando de agricultura se trata, la forma convencional considera las condiciones de terreno como homogéneas y aplica la misma cantidad de insumos, por ejemplo, agua, fertilizantes y agroquímicos, a toda la superficie de siembra. Esto incrementa los costos de inversión y aumenta los riesgos de contaminación ambiental. Por estas razones, el mundo le ha dado lugar a la *Agricultura de Precisión*, que crece día a día.

El principio básico de la agricultura de precisión es aplicar la cantidad justa de insumos en el lugar exacto y en el momento adecuado. Con este objetivo se usan de sistemas captore y sensores capaces de detectar, tratar y almacenar información del cultivo con el fin de mantener un monitoreo constante del mismo. Además, se usan dispositivos de comunicación que permiten mantener el flujo de información constante de estos datos. En este tipo de agricultura se utilizan también dispositivos de procesamiento global, imágenes satelitales, herramientas de software y la red mundial (internet) para mejorar la producción, con el menor impacto ambiental y costo posible.

Este proyecto en particular se enmarca dentro de un proyecto más general que consiste en el desarrollo de un robot autónomo para tratamiento selectivo de maleza. Las malezas se abren paso en medio de los cultivos y compiten con ellos por el agua del suelo, los nutrientes, el espacio y la luz, reduciendo el rendimiento y en ocasiones portando enfermedades que afectan el cultivo. Además, suelen ser muy competitivas, con una gran capacidad de adaptación a condiciones adversas y un rápido crecimiento. La práctica más utilizada en la agricultura tradicional para el control de malezas es la de aplicar herbicida pulverizando de forma homogénea por todo el terreno. Las grandes dosis de productos químicos y la resistencia de las mismas constituyen un problema serio en la agricultura mundial, lo que provoca que el uso de agroquímicos sin control tenga efectos negativos, entre los que se destacan, el gasto innecesario de herbicida (pérdida económica) y el daño medioambiental (contaminación de suelo y aguas subterráneas). Consecuentemente se hace necesario un tratamiento específico por zonas, mejorando la eficiencia en el control de malezas y reduciendo la dosis de agroquímico utilizado. A su vez, aplicar el concepto de dosificación selectiva en un vehículo autónomo realza aún más los beneficios antes mencionados.

Finalmente, la virtud de autonomía del vehículo nos independiza de personal abocado a su conducción. Logrando así mayor precisión gracias a que el uso de la tecnología optimiza en gran medida el recorrido a lo largo del terreno cultivado y asegura que no habrá solapamiento entre pasada y pasada (nunca se recorre un mismo surco 2 veces).

1.2. El prototipo

Hace ya unos años, en el Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CIFASIS) se está llevando a cabo el desarrollo de un prototipo de robot pulverizador.

En él, se basaron varios proyectos anteriores de grupos de estudiantes de diferentes ramas de ingeniería de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA) perteneciente a la Universidad Nacional de Rosario.

El prototipo puede verse en la Figura 1.1 y tiene como principal objetivo la eliminación de malezas en cultivos mediante un adecuado tratamiento. El robot debe ser capaz de desplazarse entre los surcos del cultivo siguiendo una determinada trayectoria y diferenciar las malezas del propio cultivo. Una vez detectado un objetivo, debe aplicar una dosis del agroquímico correspondiente sobre la misma de manera local.

Para lograr este objetivo, el proyecto se puede desglosar de la siguiente manera:

- a) Diseño mecánico del robot.
- b) Dimensionamiento y control de los motores eléctricos de tracción y dirección; medición de las variables necesarias para poder conocer la dirección del vehículo y reconstruir, la posición y velocidad del mismo por odometría.
- c) Desarrollo de convertidores de potencia para administrar el sistema de energía y dimensionamiento de las celdas solares para disponer de la energía necesaria en todos los procesos del vehículo.
- d) Procesamiento de imágenes: Por un lado, identificar los surcos y el terreno para poder definir trayectorias. Y por el otro, diferenciar maleza de cultivo para su posterior tratamiento selectivo.
- e) Georreferenciación mediante técnicas de GPS de precisión para poder identificar la posición del vehículo dentro de un terreno agrícola, ayudar en la generación de trayectorias y realizar mapas con ubicación de maleza y cultivo.
- f) Desarrollo e implementación de algoritmos adecuados para el seguimiento de trayectoria.
- g) Diseño y control de sistema de fumigación selectiva inteligente.

Nuestro proyecto se centra en los puntos b) y f). El mismo se plantea como la continuación de un proyecto anterior, llamado “Desarrollo e implementación de sistema de tracción y dirección para prototipo de robot desmalezador”.



Figura 1.1: Foto del robot pulverizador.

2. El Proyecto

Como se hizo mención anteriormente, nuestro proyecto forma parte de un proyecto mucho mayor. Como todo gran proyecto en desarrollo, muchas partes deben ser rediseñadas debido a cambios de funcionalidad, limitaciones de funcionamiento o simplemente para agregar prestaciones. En particular, al momento de comenzar el proyecto se debían realizar cambios críticos en los diseños de tracción y dirección, estos cambios se manifiestan en:

- Limitaciones en la potencia de los motores (350W), evidenciándose en mayor medida en el arranque.
- Inversión de marcha no convenientemente implementada.
- Actualmente no cuenta con sistema de frenado.
- Problemas en sensado de corriente consumida por los motores.
- Problemas de interferencia electromagnética en el gabinete de control.
- Instalación y montaje no convenientemente implementados.

Una vez resueltas estas limitaciones de carácter electro-mecánicas, se implementará un algoritmo de control de trayectorias que permita seguir una trayectoria de referencia en base a las mediciones de odometría que realiza el prototipo.

Además, el sistema no contaba con algoritmos de navegación autónoma que le permitieran seguir trayectorias previamente establecidas.

Por estos motivos, en el transcurso de este proyecto se trabajó en solucionar todos estos puntos. Para poder rediseñar y reemplazar las partes necesarias se contó con el siguiente equipamiento:

- 2 Motores Brushless DC in-wheel de 750W.
- 4 drivers de 750W con inversión de marcha y freno regenerativo.
- 4 sensores de corriente.
- Freescale FRDM-K64F.
- 1 gabinete GENROD.
- Fichas y terminales de todo tipo para instalaciones eléctricas.
- Componentes varios (Resistencias, capacitores, transistores, operacionales, pcb experimental, rollos de plástico)
- Cables para instalación.

Para alcanzar los objetivos propuestos, deberemos realizar a grandes rasgos las siguientes tareas:

- Investigar a fondo las prestaciones del equipo disponible para llevar a cabo el proyecto.
- Conocer en detalle el robot: Ubicaciones físicas de los componentes, conexiones e interacción entre ellos.
- Ensayo exhaustivo de motores y drivers adquiridos para entender su funcionamiento y poder parametrizarlos.
- Diseño de circuitos electrónicos para la adaptación y medición de señales de interés.
- Diseño de placa de circuito impreso (PCB) para integrar los circuitos en una única placa.
- Relevamiento y reestructuración del software de control implementado e incluir nuevas prestaciones.
- Diseño de un sistema de interconexión modular para que sea sencillo el reemplazo de un equipo dañado.

-
- Diseño y elaboración de encapsulado para protección de sensores de corriente y conexionado de drivers.
 - Solución de problemas de ruido debido a interferencia electromagnética (EMI).
 - Medición en tiempo real de variables de interés y posterior transferencia a la PC (Velocidad, posición, dirección y corriente de cada motor). Comunicación en tiempo real entre la placa de control y la PC.
 - Montaje del gabinete de control y motores en el vehículo.
 - Desarrollo e implementación de sistema de navegación autónoma para que el mismo sea capaz de seguir trayectorias de referencia.
 - Pruebas de campo para verificar el correcto funcionamiento del sistema.

3. Marco Teórico

En esta sección se verán los conceptos teóricos y técnicos necesarios sobre los dispositivos y controles a utilizar en el desarrollo del proyecto.

3.1. Sistema electromecánico de tracción y dirección

Los motores **BLDC** (Brushless Direct Current), también llamados motores de corriente continua sin escobillas, son un tipo de motores que está ganando popularidad rápidamente debido a las ventajas que presentan frente a otros tipos de motores eléctricos, como los MCC (Motor de corriente continua) o los motores de inducción. Entre sus ventajas, se destaca:

- Mejor característica Par-Velocidad.
- Alta respuesta dinámica.
- Alta eficiencia.
- Larga vida útil.
- Bajo ruido.
- Rangos de velocidad más amplios.

Este tipo de motores tiene diversas aplicaciones tales como: electrodomésticos, automotriz, aeromodelismo, automatización industrial, entre otros.

Además, la relación entre torque y tamaño es más alta, por lo que los hace especialmente útil en aplicaciones en donde el espacio y peso son críticos.

3.1.1. Sistema de tracción

Los motores BLDC son un tipo de motor sincrónico. Esto significa que el campo magnético generado por el estator y el campo magnético generado por el rotor giran a la misma frecuencia, como en los motores de inducción. Gracias a sus características constructivas, no presentan el fenómeno de desplazamiento (que normalmente se observa en los motores de inducción). Al igual que los motores de alterna, estos motores vienen en configuraciones monofásicas, bifásicas y trifásicas; y la cantidad de devanados del estator depende de la configuración de los mismos. Los más populares son los trifásicos.

Para nuestro caso (Motores *Brushless DC in-wheel*), el estator conforma el eje del motor que permanece fijo y el rotor lo “encierra” y conforma la parte móvil. El hecho de que sean in-wheel significa que se montan en la propia rueda, vinculando el rotor a la misma mediante rayos y el estator a la suspensión.

Estator

Existen diferentes tipos de estatores para estos motores, los más comunes son monofásico y trifásico. La diferencia radica en la cantidad de bobinados que posee dicho estator. En la Figura 3.1 se muestran dos esquemas en donde se pueden apreciar las diferencias de estos dos tipos.

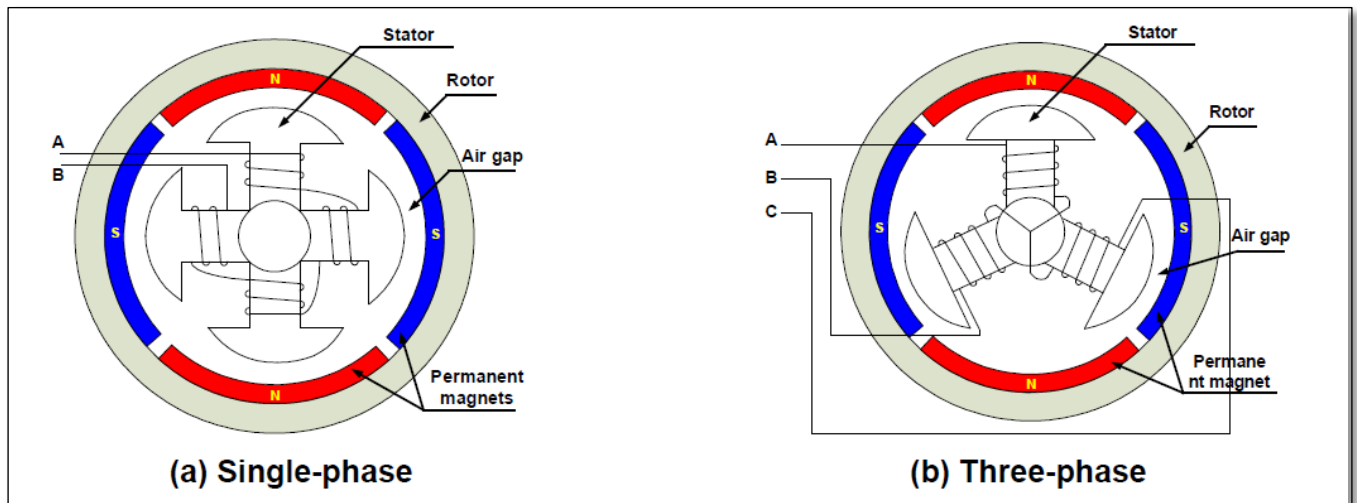


Figura 3.1: Esquema de un motor brushless DC in-wheel. (a) Monofásico - (b) Trifásico.

El estator de un BLDC, consiste de laminaciones de acero con los bobinados colocados entre las ranuras. Tradicionalmente, el estator es similar al de un motor de inducción; sin embargo, los devanados se distribuyen de una manera diferente. Cada uno de estos devanados, se distribuyen en la periferia del estator para formar un determinado número de par de polos.

Desde el punto de vista constructivo existen dos tipos de bobinados en el estator, senoidales y trapezoidales. Esta diferencia radica básicamente en la forma que toma la fuerza contraelectromotriz (Back EMF) producida en cada caso (Mostradas en la Figura 3.2). Además de la EMF, la corriente de fase también toma este tipo de formas (trapezoidales o senoidales) dependiendo del tipo de bobinado. Esto hace que el torque de salida de un motor senoidal sea más suave que el de uno trapezoidal. Sin embargo, esto viene con un costo extra, ya que los motores senoidales requieren más devanados en su interconexión debido a la distribución de bobinas en la periferia del estator aumentando la cantidad de cobre requerida por los devanados.

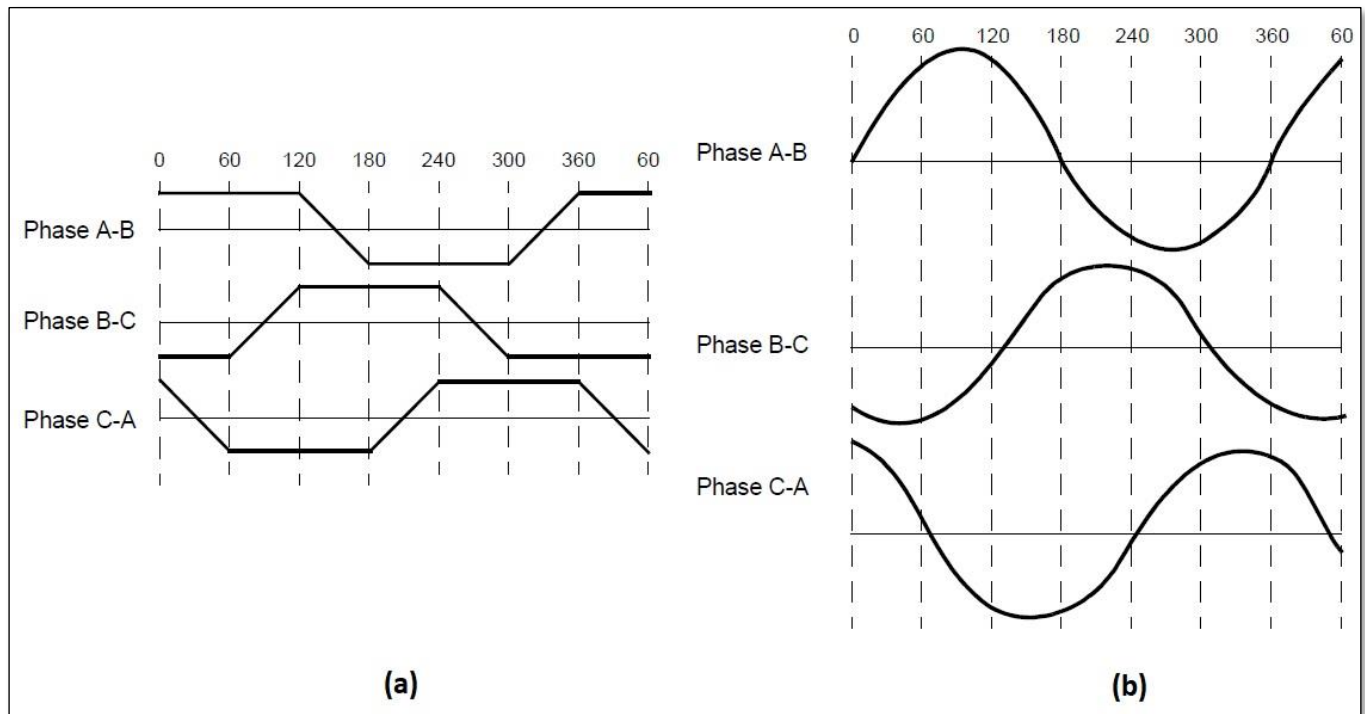


Figura 3.2: Fuerza contra electromotriz (Back EMF). (a) Trapezoidal - (b) Senoidal.

Rotor

El rotor está constituido por un imán permanente y la cantidad de pares de polos que posee es variada, dependiendo de la aplicación. Existen diferentes tipos de rotores, 3 de ellos se muestran en la Figura 3.3. Como se aprecia, hay de polos salientes, lo que significa que estos imanes están fuera del rotor en sí mismo. Por otro lado, hay dos tipos que tienen a los imanes dentro del rotor; están los embebidos, que se encuentran en la periferia del rotor y están los insertados que van desde la superficie hasta el interior del rotor.

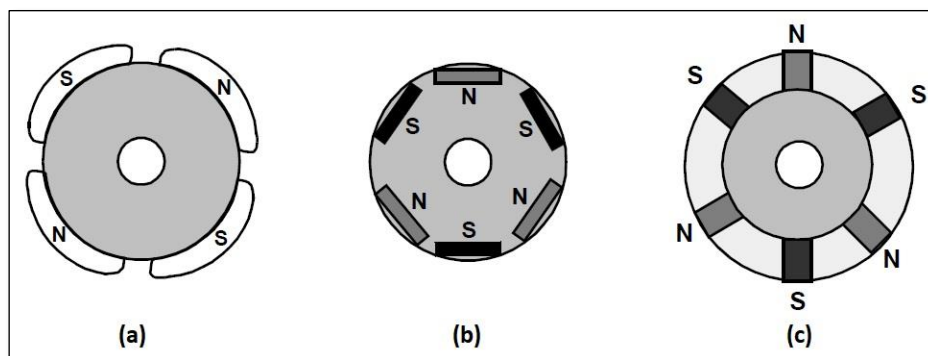


Figura 3.3: Tipos de Rotores. (a) Polos salientes - (b) Embebidos - (c) Insertados.

Sensores de efecto HALL

A diferencia de un motor de corriente continua con escobillas, la conmutación de un motor BLDC se controla electrónicamente. Para girar el motor BLDC, los devanados del estator deben energizarse en

una secuencia. Además, es importante conocer la posición del rotor para saber qué bobinado se activará después de la secuencia de energización. Con la finalidad de conocer la posición del rotor, estos motores poseen **sensores de efecto Hall** incrustados en el estator.

La mayoría de los motores BLDC tienen tres sensores Hall. Cada vez que los polos magnéticos del rotor pasan cerca de los sensores Hall, emiten una señal alta o baja, lo que indica que el polo N o S está pasando cerca de los sensores. En base a la combinación de estas tres señales de sensor Hall, se puede determinar la secuencia exacta de conmutación.

La Figura 3.4 muestra una sección transversal de un motor BLDC. En la misma puede verse un rotor con sus respectivos imanes permanentes (N y S alternativamente) y los sensores Hall incrustados en la parte estacionaria del motor.

Según la posición física de los sensores Hall, hay dos secuencias distintas de conmutación que se deben seguir para controlar el motor. Los sensores pueden estar a 60° o 120° de desplazamiento de fase entre sí. En base a esto, el fabricante del motor define la secuencia de conmutación, que se debe seguir cuando se controla el motor.

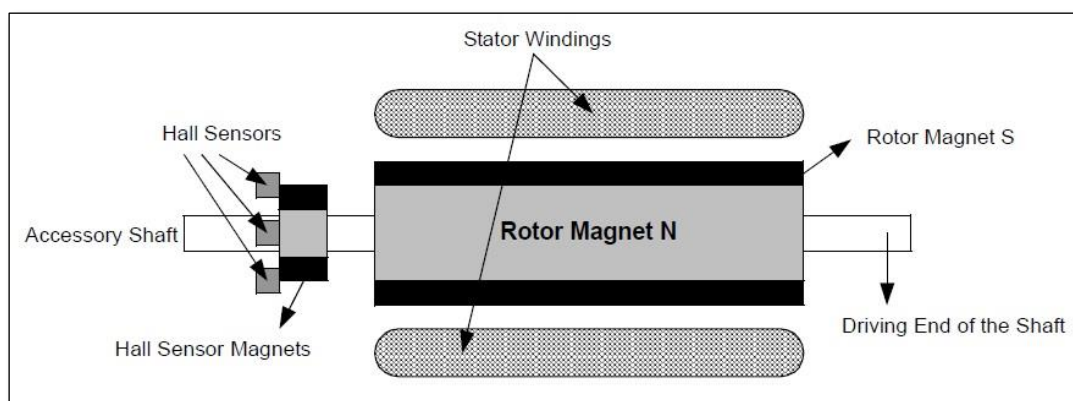


Figura 3.4: Disposición de los principales componentes de un BLDC.

Comando de los BLDC

Como se mencionó, para controlar estos motores se debe actuar sobre la secuencia de conmutación. Cada secuencia tiene uno de los devanados energizado a V_{CC} (la corriente entra en el devanado), el segundo devanado está a GND (la corriente sale del devanado) y el tercero no se encuentra energizado. El torque se produce debido a la interacción entre el campo magnético generado por las bobinas del estator y los imanes permanentes del rotor. Idealmente, el torque máximo ocurre cuando estos dos campos están a 90° entre sí y disminuye mientras dicho ángulo se achica. Para mantener el motor en funcionamiento, el campo magnético producido por los devanados debe cambiar de posición a medida que el rotor se mueve para alcanzar el campo del estator. Lo que se conoce como "*Six-Step Commutation*", define la secuencia de energización de los devanados.

Secuencia de Conmutación

La Figura 3.5(a) muestra un ejemplo de señales de sensor Hall, la Back EMF y la corriente por fase. La Figura 3.5(b) muestra la secuencia de conmutación que se debe seguir con respecto a los sensores Hall para obtener un correcto funcionamiento.

Cada 60 grados eléctricos de rotación, uno de los sensores Hall cambia de estado. Por este motivo, se requieren seis pasos para completar un ciclo eléctrico. En sincronismo, con cada 60 grados eléctricos, se debe conmutar las bobinas. Como un ciclo eléctrico en general no corresponde a una revolución mecánica del rotor, la cantidad de ciclos eléctricos que deben repetirse para completar una rotación mecánica está determinada por los pares de polos del rotor. Para cada par de polos del rotor, se completa un ciclo eléctrico. Entonces, la cantidad de ciclos eléctricos para dar una revolución mecánica en el motor, es igual a la cantidad de pares de polos que posee la máquina.

Por otra parte, para variar la velocidad de rotación, se utiliza una señal de modulación por ancho de pulso (PWM) en las llaves, a una frecuencia mucho mayor que la frecuencia de rotación del motor. Generalmente la frecuencia de la señal PWM debería ser por lo menos diez veces mayor que la máxima frecuencia de rotación del motor. Otra ventaja de utilizar la técnica PWM es que si la tensión *DC* del bus es mucho mayor que la tensión nominal del motor entonces se puede limitar el ciclo de trabajo o *duty cycle* del PWM para controlar la tensión media aplicada al mismo.

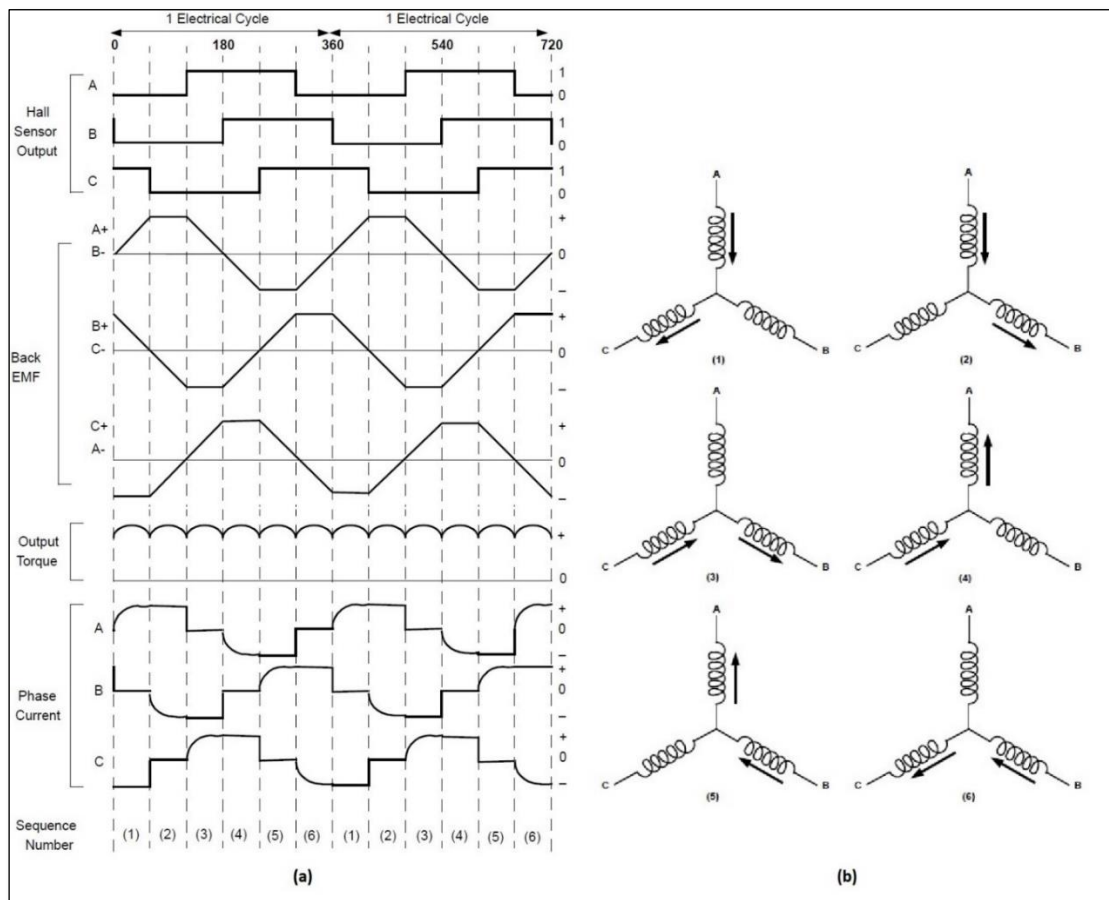


Figura 3.5: (a) Diagrama temporal de una secuencia - (b) Secuencia en las bobinas.

Las conmutaciones de las bobinas se llevan a cabo mediante llaves electrónicas (transistores MOSFET, IGBT, entre otros) conformando lo que se denomina *punte 'H'*. Existen dos tipos de puentes, *monofásicos* y *trifásicos*, dependiendo del tipo de motor a controlar (Figura 3.6).

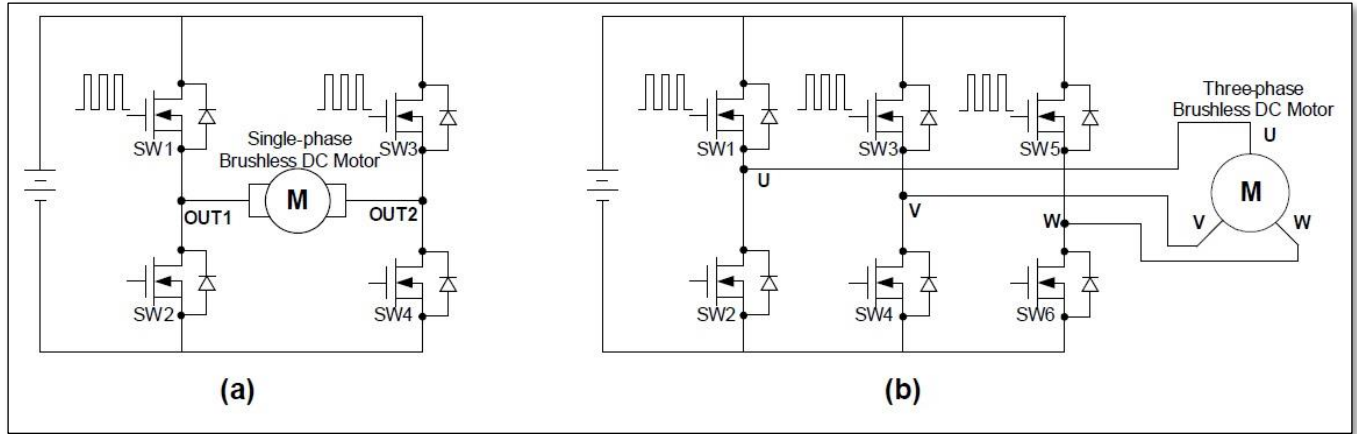


Figura 3.6: Inversores para controlar un BLDC. (a) Monofásico - (b) Trifásico.

Veamos ahora a las figuras, Figura 3.5 (b) y Figura 3.6 (b). Tomemos como equivalentes a las fases A con U, B con V y C con W. En la primera situación de la Figura 3.5 (b) conducen las fases U y W por lo que en la Figura 3.6 (b) las llaves SW1 y SW6 están cerradas (transistores en conducción). En la segunda situación cuando conducen las fases U y V, se encuentran cerradas SW1 y SW4. Siguiendo la secuencia de la Figura 3.5 (b), mencionamos los correspondientes casos de llaves cerradas; por lo tanto, a continuación, se encuentran cerradas SW5 y SW4. Luego lo hacen SW5 y SW2, seguido de SW2 y SW3, y finalmente cierran el ciclo SW3 y SW6.

Modelo eléctrico de un motor BLDC

La mayoría de los drivers de los motores BLDC son un puente H trifásico donde la conmutación se hace a partir de los sensores de efecto Hall. Una de las técnicas de control más simples y fáciles de implementar es la trapezoidal en la cual conducen sólo dos bobinados a la vez durante 1/3 del período eléctrico.

Basándonos en el modelo eléctrico del motor, mostrado en la Figura 3.7, podemos calcular la tensión del puente $V_{ab}(t)$ como:

$$V_{ab}(t) = 2Ri_a(t) + 2L \frac{di_a(t)}{dt} + E_a - E_b \quad (3.1)$$

$$\text{Donde: } E_a - E_b = EMF$$

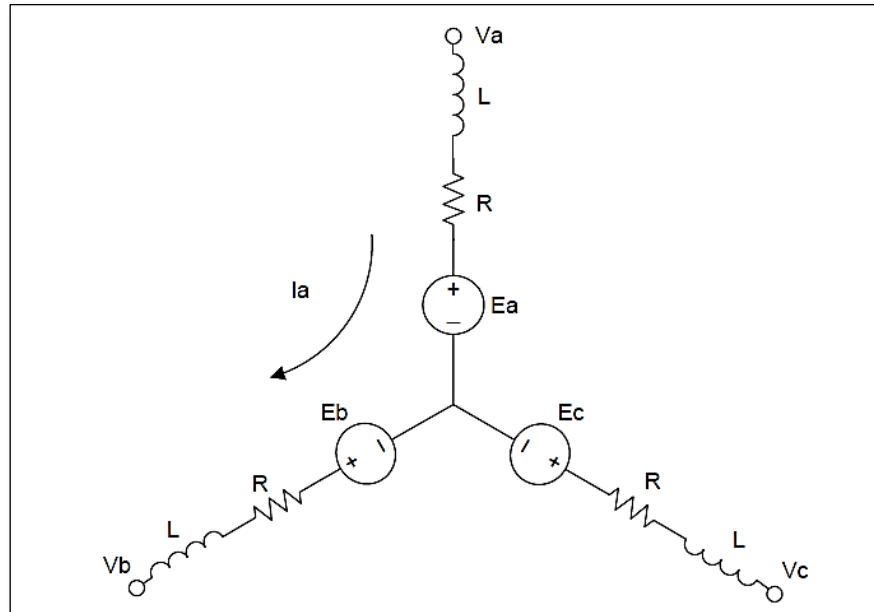


Figura 3.7: Modelo eléctrico de un motor BLDC trifásico.

De la ecuación, podemos concluir que la corriente $i_a(t)$ depende instante a instante de los siguientes parámetros:

- La tensión aplicada a través de dos de los bobinados V_{ab} .
- La resistencia e inductancia del bobinado R y L .
- La fuerza contra electromotriz EMF .

Además, como la EMF es proporcional a la velocidad angular del motor Ecuación (3.2), en condición de reposo o de bloqueo (velocidad nula), esta tensión será nula. Esto implica que cuando el motor se encuentra detenido o bloqueado la corriente en sus bobinados está limitada únicamente por sus resistencias R , por lo que debe evitarse que el mismo permanezca un tiempo prolongado en dicha condición, ya que puede provocarse sobrecalentamiento de sus bobinados.

$$EMF = K_e \omega = K_e \frac{v}{r} \quad (3.2)$$

Donde: K_e (constante de motor)

Driver de motor BLDC

El principio sobre el cual se diseñan estos drivers ya se explicó anteriormente, por lo que en esta sección se hará mayor hincapié en las características extras que tienen los drivers que se usarán, particularmente, la posibilidad de inversión de marcha y freno regenerativo.

El estado de funcionamiento de un motor BLDC se puede representar gráficamente como se muestra en la Figura 3.8, para ello se utiliza un sistema de coordenadas en el que el eje horizontal representa la EMF, y el vertical la corriente inducida. Como la EMF es proporcional a la velocidad, el eje horizontal también será de velocidades; del mismo modo, el torque se representa en el eje vertical ya que es proporcional a la corriente. Estos dos parámetros definen la zona de operación del motor en uno de los cuatro cuadrantes. Los signos de EMF e I dan una imagen del sentido de transferencia de la potencia del motor; así, en los cuadrantes 1 y 3 el producto $EMF \cdot I$ es positivo, esto equivale a decir que la potencia es positiva y que el motor está absorbiendo energía eléctrica para convertirla en energía mecánica; en otras palabras, se está comportando como motor. En los cuadrantes 2 y 4, el producto es negativo; en estas condiciones la máquina cede energía eléctrica, es decir, se comporta como generador.

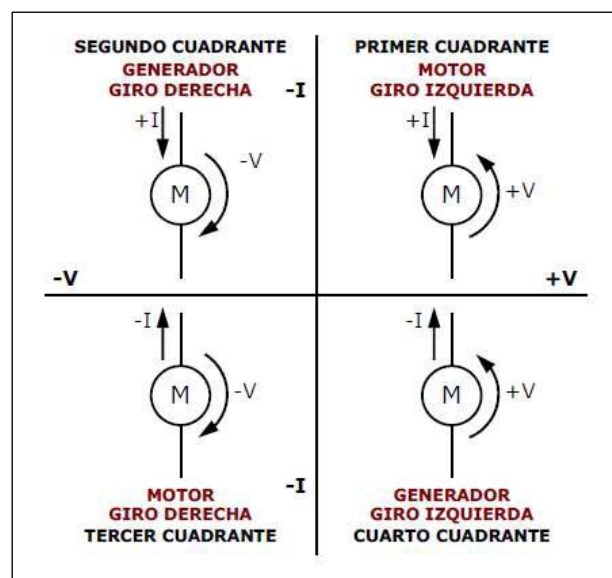


Figura 3.8: Cuadrantes de operación de un BLDC.

Los drivers que se implementarán están preparados para que los motores puedan trabajar en los cuatro cuadrantes. Esto permite que el motor funcione en algunos casos como generador, devolviendo energía a la fuente. Controlando la corriente que circula cuando el motor opera como generador, se logra aplicar una fuerza de frenado. Este fenómeno se llama, *frenado regenerativo*.

La inversión de marcha en estos motores se logra invirtiendo dos fases. En el caso de los drivers que se utilizarán, la inversión de marcha se controla con una señal digital.

Los drivers a usar tienen esta posibilidad resuelta en la electrónica que se encuentra en su interior y se comanda externamente con una señal digital.

Cuando se da la orden al motor para que disminuya su velocidad, éste responde disipando la energía que tiene acumulada por inercia, por lo que el sistema puede llegar a ser demasiado lento. Para lograr acelerar esta disminución de velocidad, se puede disipar la energía sobrante (freno mecánico) o se la puede convertir en energía eléctrica y devolverla a la red de alimentación para ser aprovechada más adelante. A este concepto se lo conoce como freno regenerativo.

3.1.2. Sistema de dirección

A continuación, en la Figura 3.9, se muestra la parte delantera del prototipo en donde se puede observar el sistema de dirección implementado en el mismo. En la misma puede verse que la dirección se comanda mediante un sistema de varillas cuyo movimiento es accionado mediante un motor. Para lograr las prestaciones necesarias se utilizó un motor paso a paso acoplado a una reducción mecánica 10 a 1. De este modo, se transforma el desplazamiento angular en un desplazamiento longitudinal de las varillas y posteriormente, este movimiento se traduce en un movimiento angular de las ruedas. Para la medición del ángulo de giro de las ruedas, variable de interés para el control del vehículo, se utilizó un encoder absoluto (OMRON E6CP-A) montado sobre el mismo motor paso a paso.

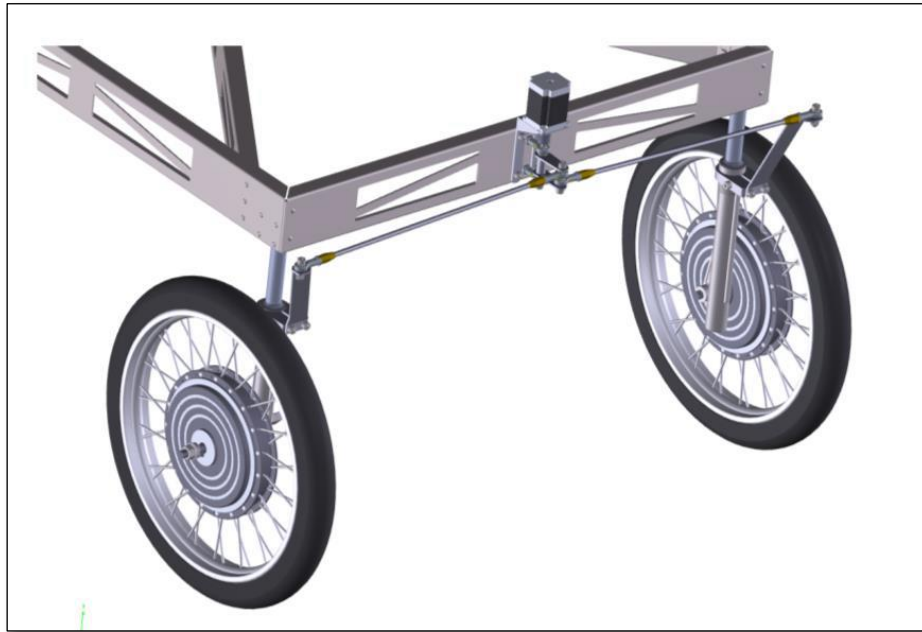


Figura 3.9: Sistema de dirección del prototipo.

Motor paso a paso (PaP)

Existe una gran diversidad de modelos de estos motores dependiendo del número de fases de su estator, la alimentación de estas (unipolar o bipolar), del número de pasos por vuelta y del tipo de rotor (reluctancia variable, imanes permanentes o híbridos). En cuanto al control, existen tres modos de operación, paso entero, medio paso y micropaso.

En el paso entero, cada vez que se modifica la alimentación de las fases del estator se avanza un paso disponiendo de torque nominal del rotor. En el medio paso se avanza sólo medio paso con lo que se dispone de mejor resolución, pero el torque desarrollado es menor. El funcionamiento en micropaso consiste en alimentar al mismo tiempo varias fases a la vez con corrientes medias distintas, de modo que la posición media del flujo en el entrehierro se puede fijar en cualquier posición. Con este funcionamiento se consigue una resolución muy buena sin sacrificar torque.

Un motor paso a paso (Figura 3.10) está compuesto principalmente por un rotor laminado, bobinados de fase y un sistema de control es el encargado de excitar las bobinas estáticas para que el motor gire un paso por cada pulso aplicado.

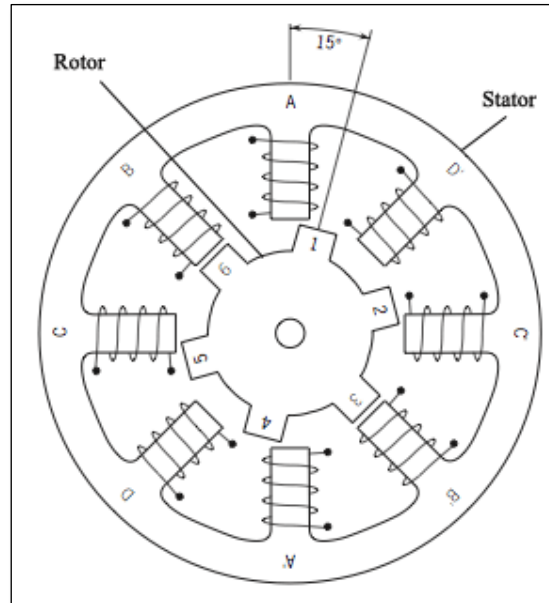


Figura 3.10: Esquema de un motor paso a paso (PaP).

El estator tiene ocho polos salientes, mientras que el rotor solamente dispone de seis. Ambos se suelen construir generalmente de acero blando. Los polos del estator se han bobinado para formar cuatro fases, cada una de las cuales consta de dos bobinas conectadas en serie y situadas físicamente en polos opuestos. La corriente se aplica desde una fuente de potencia DC a través de llaves electrónicas.

Los motores paso a paso se pueden dividir en dos tipos de configuraciones básicas: unipolar y bipolar. La diferencia radica en el modo en el que se realiza la inversión de este campo magnético. Como se puede ver en la Figura 3.11, los motores bipolares tienen una sola bobina por fase y necesitan dos llaves de conmutación o un puente-H por cada fase. Estas llaves son las encargadas de realizar la inversión de sentido de corriente de las bobinas, el cual es requerido para cambiar la polaridad del campo magnético inducido por dicha corriente y así lograr que el motor gire adecuadamente. En cambio, los motores unipolares poseen una sola bobina por cada fase y necesitan solo una llave de conmutación o dos transistores a masa.

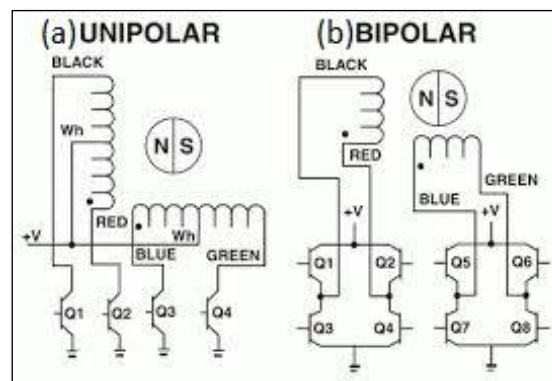


Figura 3.11: (a) PaP unipolar - (b) PaP bipolar.

Estos motores necesitan la inversión de la corriente que circula en sus bobinas para dar un paso. Aplicando una correcta secuencia de inversiones de corriente se logra que el motor gire. La cantidad de pasos que puede dar un motor está determinada por la cantidad de pares de polos que posee, en un motor con 2 pares de polos se corresponden 8 pasos (Figura 3.12).

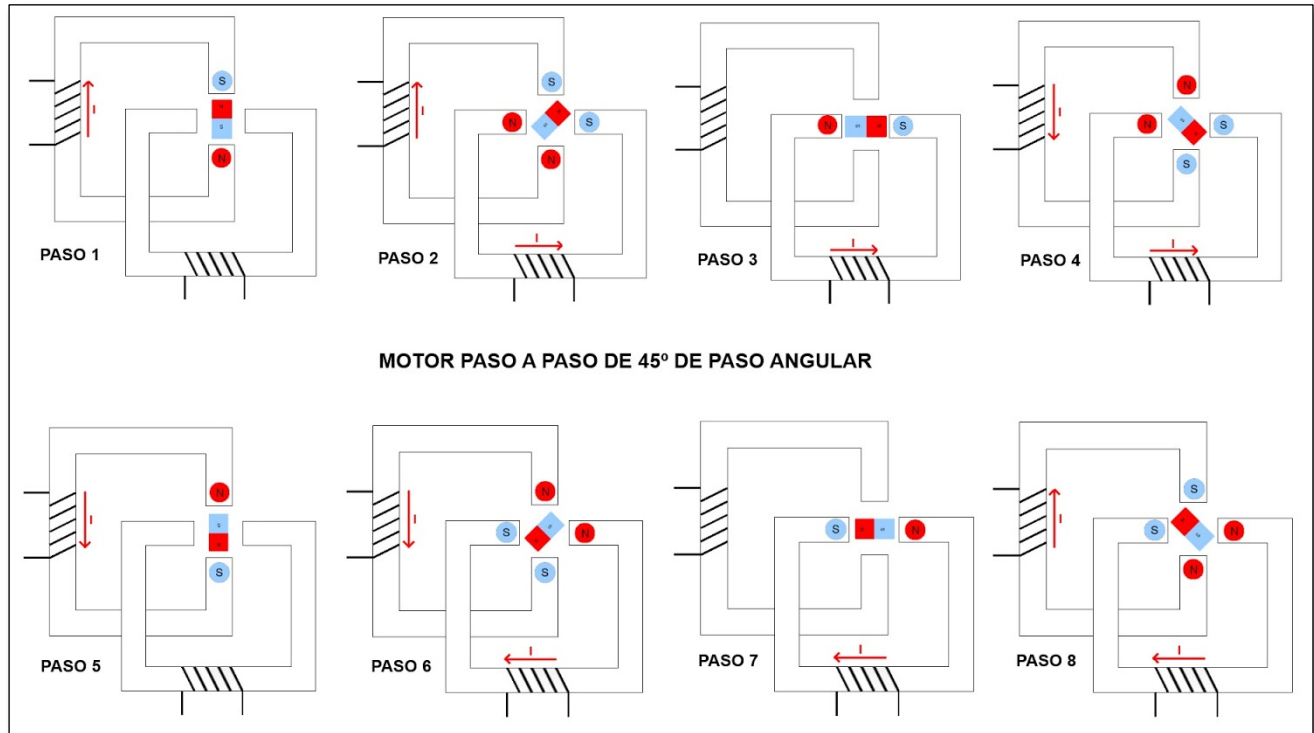


Figura 3.12: Secuencia de pasos de PaP bipolar de 2 pares de polos.

Siguiendo una secuencia correcta de control de la apertura y cierre de los interruptores, podremos girar el motor en el sentido y a la velocidad que deseemos con la ventaja de no tener que utilizar ningún tipo de realimentación. Además el error de posición que puede tener este tipo de motores no es acumulativo y tiende a cero en cuatro pasos, es decir cada 360° eléctricos. Cada cuatro pasos el rotor vuelve a la misma posición con respecto a la polaridad magnética y a la trayectoria del flujo. La precisión en el posicionado es un factor que mide la calidad de estos motores. Se diseñan de modo que tras recibir una señal eléctrica pasen de una situación de equilibrio a otra posición de equilibrio diferente separada de la anterior un determinado ángulo. Esta precisión depende en gran manera del mecanizado del rotor y estator con lo que su fabricación es delicada. Cuando una carga se aplica sobre el eje, se produce un par elevado que trata de posicionar el rotor en su posición natural de equilibrio. La responsabilidad de que este par sea mayor o menor recae en el entrehierro. Cuanto más pequeño sea, y esto depende de la calidad de la fabricación, el par que presente el motor a la carga y su precisión serán mayores.

Tipos de drivers de motores PaP

Simple Voltage Drivers

Los drivers tradicionales utilizan llaves electrónicas de potencia en configuración de puente-H para manejar las corrientes bipolares de los devanados de fase. En cada amplificador, al cerrarse $S1$ y $S4$ se aplica la tensión nominal en los devanados y la corriente fluye desde la alimentación hasta masa por el camino formado por $S1$, el devanado y $S4$. Luego al abrirse $S1$ y $S4$ y cerrarse $S2$ y $S3$ se invierte la dirección de las corrientes en los devanados.

En la Figura 3.13 se puede observar el esquema asociado a este tipo de drivers en donde solo las resistencias de devanado limitan las corrientes y V_{CC} no puede sobrepasar la tensión nominal V_{rated} .

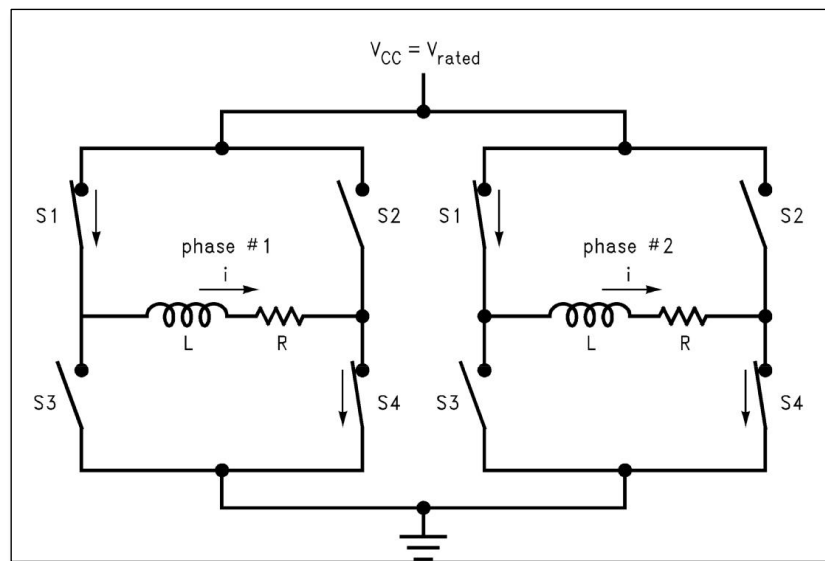


Figura 3.13: Simple Voltage Driver de un motor paso a paso bipolar.

L/R Drivers

Los drivers L/R utilizan dos resistencias de potencia en serie para aumentar la resistencia total de devanado R y de esta manera disminuir la constante de tiempo $\tau = L/R$. Por ejemplo, una resistencia de 45Ω en serie con cada una de las resistencias de devanado de 15Ω logra una constante de tiempo cuatro veces menor que la original. De esta manera no solo se logra una mejor evolución de la corriente, sino que también un mayor torque del motor para altas velocidades.

Cuando el motor se encuentra frenado realizando torque de sostén (*holding torque*) o gira a bajas velocidades, la serie de resistencias de potencia protegen al motor limitando las corrientes de devanado por debajo de su valor nominal.

Si bien existe una tensión máxima de alimentación para que no se exceda la tensión nominal de los devanados, la verdadera limitación de potencia en este tipo de drivers se encuentra en la potencia disipada en las resistencias conectadas en serie. Por ejemplo, si la tensión de alimentación es de $60V$, la corriente nominal de devanado es de $0,5A$ y la resistencia de devanado de 15Ω , se necesitan dos resistencias de 105Ω en serie para limitar las corrientes debajo del límite de $0,5A$ por fase. La potencia disipada en las resistencias cuando el motor se encuentra en posición de sostén es de $2 \times 105 \times 0,5^2 = 52,5W$, mientras

que la potencia total disipada en el driver $2 \times 60 \times 0.5 = 60W$, por lo que la eficiencia del driver es aproximadamente de 12.5%. Esta última es la principal desventaja de este tipo de drivers donde la eficiencia siempre se encuentra por debajo del 25%.

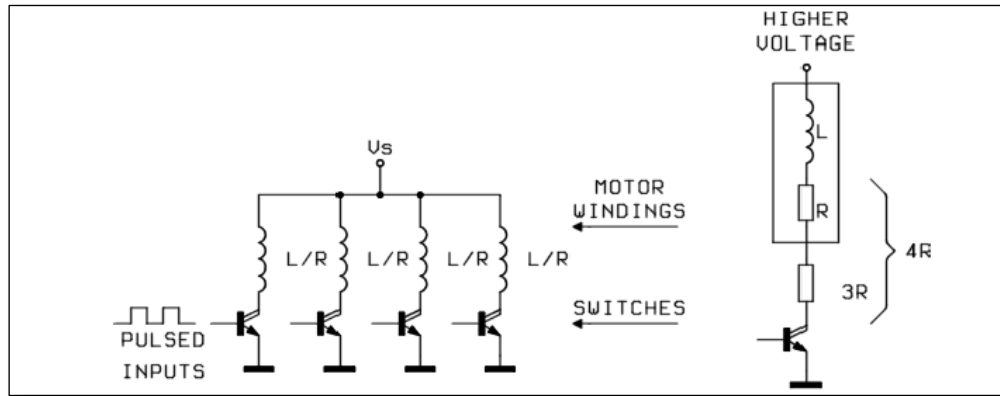


Figura 3.14: Driver L/R.

Chopper drivers

Los Chopper Drivers incrementan el *slew rate* de las corrientes de devanado al aplicar una tensión $V_{cc} \gg V_{rated}$. Las realimentaciones en las conmutaciones de los puentes-H son las encargadas de mantener la corriente de devanado por debajo de su límite nominal. Las ilustraciones, Figura 3.15 y Figura 3.16, muestran los diferentes estados de un puente H mediante la aplicación de la técnica chopper. La resistencia de sensado R_s entre el puente y tierra convierte el valor de la corriente de devanado en una tensión proporcional a dicha corriente para ser utilizado como realimentación, luego dicho valor de tensión es comparado con una tensión de referencia.

Cuando la tensión de realimentación V_{RS} se mantenga por debajo de la referencia, las llaves $S1$ y $S4$ aplican al devanado el total de la tensión de alimentación y las corrientes de devanado aumentan rápidamente. Luego cuando la tensión V_{RS} es igual a la tensión de referencia, es cuando el valor de corriente es el deseado y por lo tanto $S1$ y $S2$ le quitan la alimentación devanado por un periodo de tiempo fijo u off-time. Durante este tiempo, las corrientes de devanado decaen lentamente y luego al finalizar dicho periodo de tiempo $S1$ y $S4$ vuelven a energizar al devanado con el total de la tensión de alimentación y las corrientes vuelven a incrementarse.

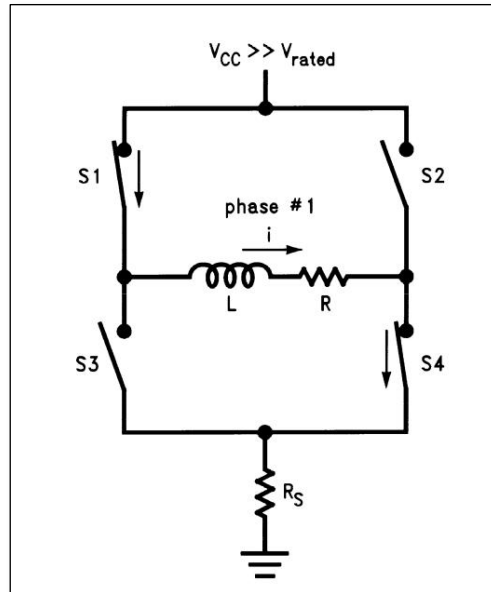


Figura 3.15: Tensión de bobinado igual a V_{CC} (Chopper Drivers).

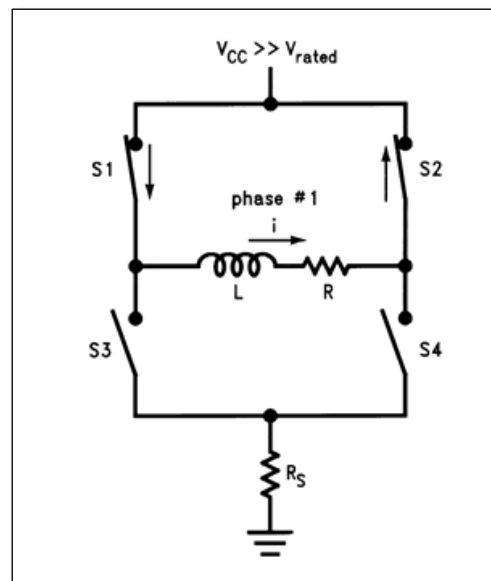


Figura 3.16: Recirculación de corriente (Chopper Drivers).

La repetición de esta secuencia resulta en una forma de corriente que posee un *ripple* similar a un diente de sierra con un periodo t_{off} , como se puede observar en la Figura 3.17. Esta acción de ‘choppeado’ de la corriente limita su valor máximo en un nivel determinado por la tensión de referencia y la resistencia de sensado siendo $I_{limite} = V_{ref}/R_s$.

El corte o *chopping* de la corriente solo ocurre cuando esta última alcanza el valor límite deseado (usualmente el valor nominal del devanado). Cuando la corriente de devanado cambia de sentido para realizar un paso en el motor, el comportamiento es el explicado anteriormente, siendo las llaves S1 y S3 las que conmutan para limitar el valor de corriente.

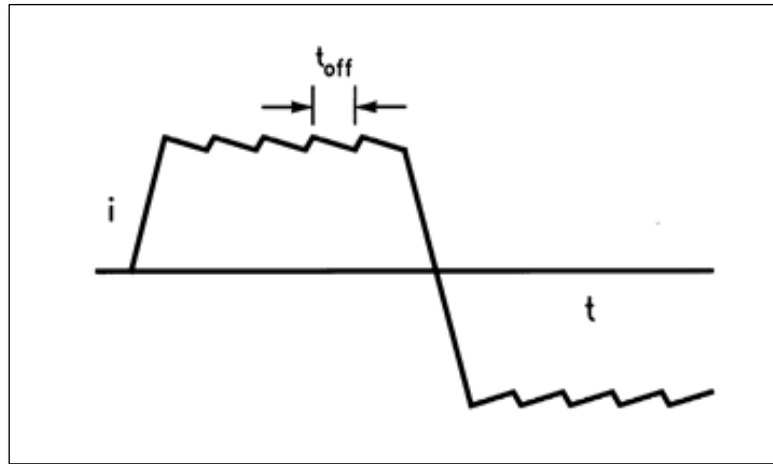


Figura 3.17: Forma de onda de corriente de fase característica (Chopper Drivers).

Debido a que el *punte-H* desenergiza a los devanados por un periodo de tiempo fijo, este tipo de driver es comúnmente llamado *fixed off-time drive*. A comparación de los drivers *L/R*, al no utilizar las resistencias en serie, se incrementa de gran manera su eficiencia teniendo comúnmente una eficiencia mayor al 75%.

Microstepping

En muy pocos casos los motores paso a paso tienen una curva torque vs posición del eje puramente senoidal, la mayoría de las veces poseen altos componentes armónicos que distorsionan esta curva y afectan la eficiencia del motor.

Este problema se puede solucionar, en parte, utilizando el método *microstepping* ya que la onda posee menor cantidad de armónicos y se aproxima a una senoidal, pero como veremos a continuación este método tiene la desventaja de reducir el torque en el eje.

Como se mencionó anteriormente, existen tres métodos básicos para el manejo de posición de motores paso a paso:

- Full step
- Half step
- Microstepping

El primero de los métodos, el *Full Step* (Figura 3.18) posee la ventaja de brindar el mayor torque, pero a su vez es el método de menor eficiencia debido a que la corriente de fase es una onda cuadrada provocando mayor consumo de las baterías.

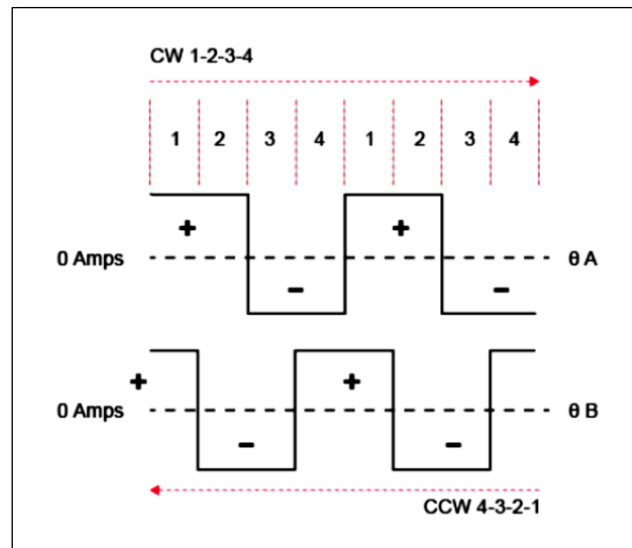


Figura 3.18: Corriente de fase método Full Step - Motor Paso a Paso.

El método de *Half Step* (Figura 3.19) a comparación del método anterior brinda torque de motor levemente inferior, pero a la vez mejora su eficiencia debido a que la onda de corriente posee conmutaciones de menor amplitud.

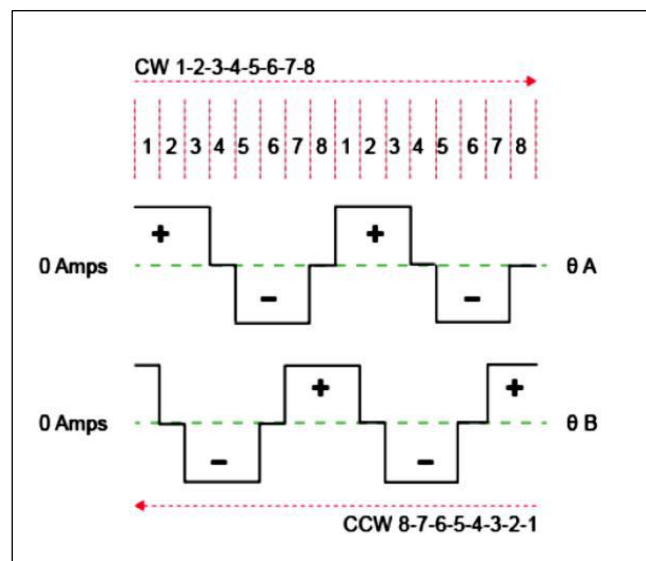


Figura 3.19: Corriente de fase método Half Step - Motor Paso a Paso.

Por otro lado, si aumentamos la división del paso utilizamos el método *Microstepping* (Figura 3.20) que posee ventajas y desventajas opuestas al *Full Step* siendo este el método que brinda el menor torque de motor, pero a su vez el de mayor eficiencia energética, ya que su forma de onda de corriente de fase se va aproximando cada vez a una senoidal a medida que la cantidad de micropasos a realizar sea mayor.

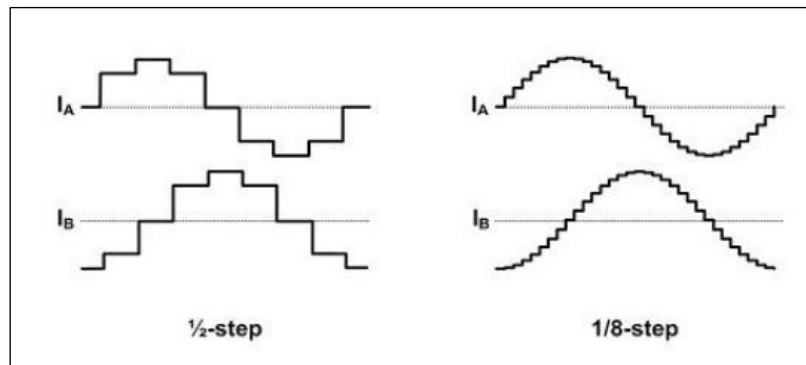


Figura 3.20: Formas de onda de corriente de fase - Microstepping.

Como se notó anteriormente, el compromiso es que a medida que se incrementa el número de *microsteps* por pasos, el torque incremental disminuye drásticamente como se puede observar en la siguiente tabla:

<i>Microsteps</i> utilizado	<i>Holding torque incremental</i>
1	100 %
2	70.71 %
4	38.27 %
8	19.51 %
16	9.80 %
32	4.91 %
64	2.45 %
128	1.23 %
256	0.61 %

Tabla 3.1: Torque de sostén vs método *Microsteps*

3.2. Sistema de navegación autónoma

Esta sección se dedica a comentar los temas relacionados con la navegación autónoma de un vehículo. Esta herramienta, es muy utilizada en distintas aplicaciones de robots como, por ejemplo, en exploraciones de terreno de otros planetas, en misiones militares, en seguridad, en trabajos de agricultura, etc.

Por este motivo, y para poder evaluar el desempeño de los dispositivos de hardware y software presentados en las secciones anteriores, acompañado además de que es una función necesaria dentro del desarrollo del robot pulverizador, en este proyecto se desarrolló un algoritmo de seguimiento de trayectoria.

Cuando se habla de navegación autónoma, área en constante crecimiento en la actualidad, el primer problema a tratar es el seguimiento de trayectorias. Para poder resolver este problema investigamos sobre distintos artículos relacionados a este tema, centrándonos en uno en particular ^[7].

El objetivo propuesto es encontrar una lógica de control para la velocidad y otra para la dirección de giro del robot que le permita al mismo alcanzar un punto objetivo en el plano. Entonces, para seguir una determinada trayectoria, le brindaremos en forma ordenada distintos puntos, llamados waypoints, que

pertenezcan a la misma. Aclaramos que estos puntos serán setpoints determinados previamente, dependiendo de la trayectoria particular que se quiera seguir.

Cabe destacar que nuestra solución final se desarrollará por software, algoritmo que será programado en lenguaje C y se ejecutará sobre un sistema operativo Linux, que se encuentra en el corazón del robot, la NUC (Next Unit of Computing). Estamos ahora en condiciones de mencionar las herramientas que se disponen para la solución del problema y comentar como se llevará a cabo la misma.

Para poder ejecutar la lógica de control dentro de la NUC, es necesario establecer una comunicación entre ella y el gabinete de control, más precisamente, con el microcontrolador que comanda la tracción y dirección del vehículo. En dicha comunicación se intercambiarán datos de vital importancia como la velocidad y la dirección, junto a otros más que no son de interés para esta sección. En la Figura 3.21 se puede ver un esquema simplificado de la comunicación. Ésta comunicación debe además de transmitir los datos, mantener un sincronismo temporal de los mismos.

El procedimiento sería el siguiente: el microcontrolador le proporciona periódicamente la velocidad y dirección del robot a la NUC, ésta los recibe y ejecuta el algoritmo de control basándose en dichos datos y en la trayectoria a seguir, respondiendo con consignas de velocidad y dirección.

Respecto a odometría, el algoritmo debe conocer el punto geográfico al que se desea llevar al robot. Tomando una coordenada como punto inicial (origen (0,0) más orientación (robot apuntando hacia el eje x)) se calcula la velocidad y la dirección que debe darle al microcontrolador como consigna. Con estas consignas, el microcontrolador mueve al robot y, a su vez, se comunica con la NUC periódicamente brindándole la velocidad y la dirección real del robot. Con el tiempo que transcurrió (tiempo de comunicación de 100 ms) y estos datos reales del robot, el algoritmo estima la posición y orientación del mismo. Finalmente, el algoritmo vuelve a calcular consignas de velocidad y dirección. Esto se realiza hasta que la posición estimada del robot es la deseada, momento en el que se envían consignas de velocidad y dirección nulas para la detección del robot y se da por finalizado el algoritmo.

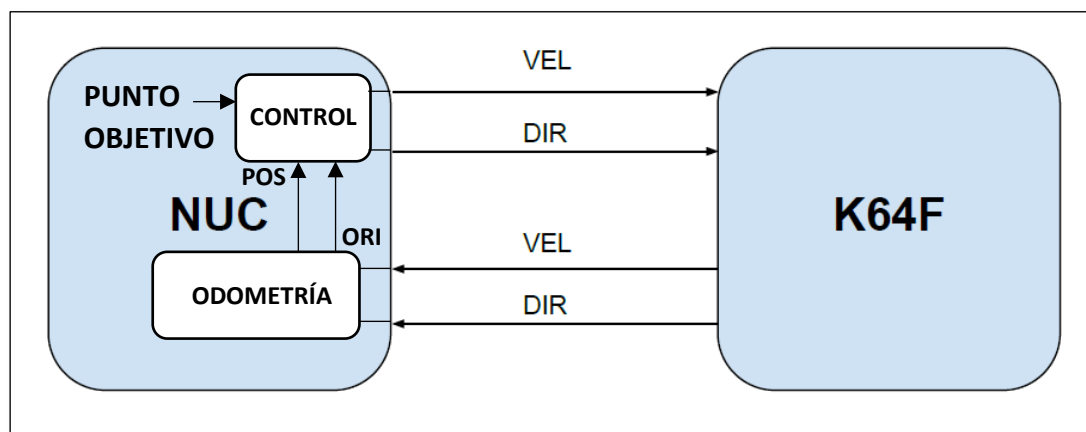


Figura 3.21: Esquema simplificado de la comunicación entre la NUC y el microcontrolador.

Para poder realizar una estimación de la posición del robot, nos basamos en un modelo del mismo. Este modelo, es conocido como modelo de Ackermann.

3.2.1. Modelo de Ackermann

Un modelo muy difundido en el área de control de vehículos tipo cuatriciclos es el **modelo de Ackermann** (Figura 3.22). Si bien este modelo está pensado para bicicletas, puede ser utilizado para cuatriciclos y permite generar leyes de control de manera sencilla.

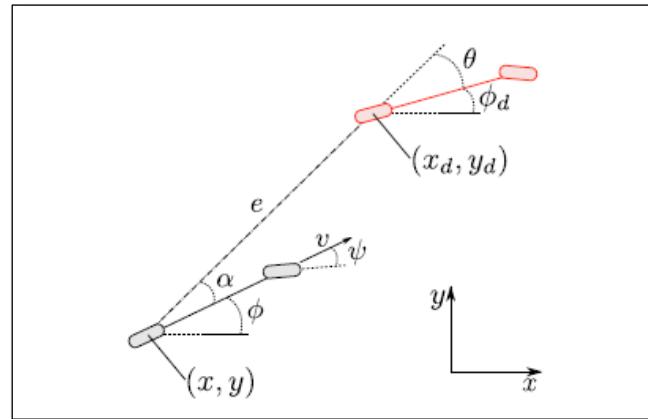


Figura 3.22: Modelo del bicicleta de Ackermann.

A continuación, Figura 3.23, ejemplificamos la situación propia de poseer un vehículo tipo cuatriciclo, y en que lugar del mismo ubicamos al bicicleta del modelo de Ackermann.

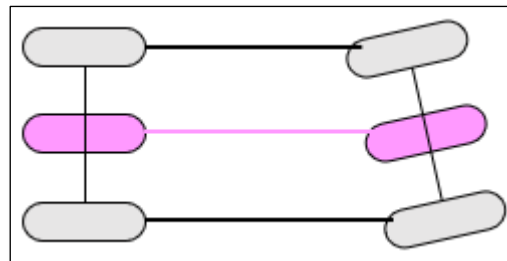


Figura 3.23: Modelo del bicicleta de Ackermann (rosa) impuesto sobre el cuatriciclo (gris).

Como se puede observar en el modelo, Figura 3.22, se tienen sólo dos ruedas donde la trasera representa al eje trasero y se ubica en el centro. De manera análoga ocurre con la rueda delantera que, a su vez, representa la dirección de giro del vehículo.

Los datos relevantes son, la posición del vehículo (x, y) , la orientación del mismo (ϕ) , la velocidad (v) , la dirección de giro (ψ) y la distancia al punto deseado (e) . Por otro lado, están α y θ que representan ángulos de interés, α es la diferencia entre la orientación ϕ y el ángulo que forma la recta que une la posición actual con la posición deseada del robot, mientras que θ es lo mismo pero respecto a la orientación deseada ϕ_d . Los datos de la posición deseada del robot (indicada en rojo en la imagen), se representan con el subíndice d. Notemos que, además de una posición deseada (x_d, y_d) , también se pretende llegar a destino con una cierta orientación definida (ϕ_d) . Esta última condición sobrepasa nuestro problema a resolver en un principio ya que nuestro objetivo es que el robot llegue a una posición deseada en el plano sin importar su orientación, pero poder contar con esto nos ayudará con el seguimiento de trayectoria, sin que la complejidad de la solución aumente demasiado, ya que las trayectorias se generan con secuencias de puntos que debe alcanzar el robot. Así, por ejemplo, si

quisiéramos que el robot realice una circunferencia podemos ordenarle algunos puntos de la misma y junto a la orientación que el robot debe tener en estos puntos, nos aseguramos de que no realice otro tipo de trayectoria, ya que podría cumplir con alcanzar a estos puntos pero con cualquier orientación, lo que se vería reflejado en la realización de otra trayectoria distinta a la circunferencia que se deseaba.

Aplicando trigonometría al modelo de la Figura 3.22, podemos deducir las siguientes ecuaciones:

$$\begin{cases} \dot{x} = v \cos \phi \\ \dot{y} = v \sin \phi \\ \dot{\phi} = v \frac{\tan \Psi}{l} \end{cases} \quad (3.3)$$

Dónde l es la distancia entre ejes del vehículo. Luego, pasando a coordenadas polares nos queda:

$$\begin{cases} \dot{e} = -v \cos \alpha \\ \dot{\alpha} = v \frac{\sin \phi}{e} - v \frac{\tan \psi}{l} \\ \dot{\theta} = v \frac{\sin \alpha}{e} \end{cases} \quad (3.4)$$

Estas últimas ecuaciones, representan las ecuaciones de estado del robot, $\{e, \theta, \alpha\}$.

Por otro lado, a partir de las coordenadas del punto de referencia (x_d, y_d, ϕ_d) , aplicando trigonometría y reagrupando se obtiene:

$$\begin{cases} e = \sqrt{(x_d - x)^2 + (y_d - y)^2} \\ \theta = \text{atan2}(y_d - y, x_d - x) - \theta_d \\ \alpha = \theta - \phi + \phi_d \end{cases} \quad (3.5)$$

Estas últimas ecuaciones, nos indican como se obtiene la distancia e entre la posición actual del robot y la deseada, y los ángulos α y θ por trigonometría. Destacamos especialmente la obtención de estas tres variables, ya que serán de vital importancia para el control de trayectoria, cuestión que se verá en detalle más adelante.

3.2.2. Sistema de ubicación espacial basado en odometría

Se define odometría como el estudio de la estimación de la posición de vehículos, que poseen ruedas, durante la navegación. Ésta, es necesaria cuando no se poseen datos específicos de la posición del vehículo, ya sean brindados por sensores que no se poseen, por GPS (Global Positioning System), etc.

En este caso en particular, la posición del robot no puede ser conocida si no es por odometría. Debido a esto, conocer las ecuaciones de la sección anterior es fundamental para poder realizar un control de trayectoria basado en odometría. Adicionamos a estas ecuaciones, la obtención de la posición en (x,y) del robot.

$$\begin{aligned} x &= \int v \cos \left(\int \frac{v}{l} \tan \psi \, dt \right) dt \\ y &= \int v \sin \left(\int \frac{v}{l} \tan \psi \, dt \right) dt \end{aligned} \quad (3.6)$$

Se puede observar que la posición del robot, dada por estas ecuaciones, se deduce de la velocidad v y de la dirección ψ (datos que serán brindados por el microcontrolador) del mismo. Por el momento, dejaremos de lado cómo se resolverán las integrales, tema que será explicado más adelante. Lo importante de estas ecuaciones, es ver que se puede obtener la posición del robot procesando datos reales del mismo a través de ecuaciones matemáticas.

Como se comentó, la odometría proporciona una estimación muy próxima a la ubicación real del robot a cortas distancias, además es barato y permite una frecuencia de muestreo muy alta. Sin embargo, la idea fundamental de la odometría es la integración de información incremental del movimiento de las ruedas a lo largo del tiempo, lo que conduce inevitablemente a la acumulación de errores.

En particular, la acumulación de errores de orientación causa grandes errores de posición que aumentan con la distancia recorrida por el robot. A pesar de estas limitaciones, la mayoría de los investigadores coinciden en que la odometría es una parte importante del sistema de navegación del robot. Es utilizado en casi todos los robots móviles por diversas razones:

- En algunas ocasiones, la odometría es la única información disponible para navegar, por ejemplo, cuando no hay referencia externas o balizas que puedan ser detectadas por el robot o cuando otros sensores no puedan proporcionar datos utilizables.
- La información puede combinarse con la estimación de la ubicación absoluta del robot para obtener una predicción de la siguiente ubicación global que ocupará éste. Si el algoritmo de localización global deja de localizar temporalmente al robot, por ejemplo, al no lograr su convergencia, se puede usar la odometría como mecanismo de localización absoluta. Obviamente este método introduce errores en la ubicación del robot con el paso del tiempo. Estos errores serán subsanados en cuanto el algoritmo de localización global recupere la convergencia.

La odometría se basa en sensores de bajo coste y ecuaciones simples que son fáciles de implementar en software. Sin embargo, también se basa en el supuesto de que las ruedas rotan, lo que se transforma en una traslación del robot. Esta suposición tiene una validez limitada. Un ejemplo es el deslizamiento de las ruedas sobre algunas superficies que ofrecen una baja adherencia de éstas. El resultado de este fenómeno es una estimación de la ubicación del robot errónea tras realizar los cálculos vistos anteriormente.

Los errores que afectan a la odometría suelen ser de muy diversa naturaleza y se clasifican en dos categorías, los errores sistemáticos y los errores no sistemáticos. El error sistemático, como su nombre indica, es aquel que se repite constantemente, es propio del sistema y se produce de igual modo en todas las medidas que se realizan de una magnitud. Puede estar originado por un defecto en cualquiera de los componentes que constituyen el robot. Entre los errores sistemáticos más habituales encontramos:

- Utilizar ruedas con diámetro diferente.
- Mala alineación de las ruedas.
- Incertidumbre del punto de contacto efectivo de las ruedas con el suelo
- Resolución finita del encoder. Cuanto mayor sea la resolución menor es el error introducido por este fenómeno.

Los errores sistemáticos son graves porque se acumulan constantemente. Los errores no sistemáticos son aquellos que tienen una naturaleza aleatoria, impredecible y por tanto pueden afectar en cualquier lugar y momento. En muchas ocasiones estos errores están provocados por el deslizamiento de las ruedas debido a:

- Suelos resbaladizos.
- Sobre-aceleración.
- Giros rápidos.
- Mal contacto de las ruedas en el suelo.
- Interacción con objetos externos.

En la mayoría de las superficies lisas los errores sistemáticos son más peligrosos que los no sistemáticos, por contrario, en las superficies rugosas, los errores sistemáticos son escasos y predominan los errores no sistemáticos. Debido a estos errores, a medida que el robot se desplaza, aumenta la región de incertidumbre que rodea a las sucesivas estimaciones de su ubicación que se calculan haciendo uso de la odometría. Esta región crece hasta que se obtiene la ubicación absoluta del robot, haciendo uso de algún algoritmo de localización global.

En resumen, la odometría no es una forma fiable de localizar a un robot móvil. Proporciona una forma sencilla de estimar su ubicación, pero esa estimación va acumulando errores, tanto sistemáticos como no sistemáticos. Eso quiere decir que, en trayectorias largas, la estimación de la ubicación por odometría deja de ser fiable. Como consecuencia, se debe corregir la estimación odométrica periódicamente con algún algoritmo de localización global para realizar un seguimiento adecuado del robot en su desplazamiento por el entorno.

Más allá de lo mencionado sobre las limitaciones de odometría, en este proyecto se debía resolver el problema de seguimiento de trayectoria de esta manera. Uno de los algoritmos más implementados es el denominado “*Follow the carot*”. La explicación de este algoritmo se hará en una sección posterior, pero hacemos mención ya que hablaremos del control que necesitamos para lograr implementar el algoritmo en cuestión.

3.2.3. Control de velocidad y dirección del vehículo

Para poder tener éxito en que el robot se desplace hacia un determinado lugar debemos desarrollar un control. Por lo que, conociendo el modelo del robot, la posición y el punto geográfico que se lo desea llevar, se debe generar alguna ley de control que nos permita alcanzarlo.

Comenzamos entonces a elaborar alguna ley de control que cumpla con las especificaciones necesarias. Antes que nada, debe asegurar la estabilidad del sistema a lazo cerrado sin importar cuáles son las posiciones actual y deseada del vehículo, no puede haber variaciones bruscas en el recorrido a realizar por el robot, ya que podría provocar el vuelco del mismo.

Así, se llevará al vehículo a la posición deseada de manera que la convergencia sea asintótica, por lo que el problema a resolver consiste en encontrar funciones suaves de la velocidad (v) y curvatura del robot (c). Para simplificar el problema, supondremos que nuestro vehículo sólo se puede mover hacia adelante, lo que quitará gran complejidad al problema de tracking y las prestaciones que se pierden no son un requisito en el problema requerido.

Se utilizará un control proporcional de velocidad definido según la distancia entre la posición actual del robot y la pretendida:

$$v = \gamma e, \quad \gamma > 0 \quad (3.7)$$

Donde e representa la distancia entre la posición actual del robot y el punto objetivo a alcanzar, mientras que γ es la constante de proporcionalidad, y debe ser mayor a 0 para satisfacer la primera condición impuesta de que el robot sólo se desplaza hacia adelante. Reemplazando (3.7) en (3.5), obtenemos:

$$\begin{aligned} \dot{e} &= -\gamma e \cos \alpha \\ \dot{\alpha} &= \gamma e \left(-c + \frac{\sin \alpha}{e} \right) \\ \dot{\theta} &= \gamma \sin \alpha \end{aligned} \quad (3.8)$$

Donde c es la curvatura de la circunferencia que realiza el robot. Esta se define como la inversa del radio de la circunferencia que está realizando el vehículo en un determinado momento y se calcula de la siguiente manera:

$$c = \frac{\tan \psi}{l} \quad (3.9)$$

Para poder asegurar convergencia, la ley de control se deduce a partir de un estudio energético a partir de Lyapunov. Proponiendo la función candidata de Lyapunov:

$$V = \frac{1}{2}(\alpha^2 + h\theta^2), \quad h > 0 \quad (3.10)$$

Y derivando en función del tiempo se tiene:

$$\dot{V} = \alpha \dot{\alpha} + h\theta \dot{\theta} = \gamma(\alpha \sin \alpha + h\theta \sin \alpha - \alpha e c) \quad (3.11)$$

Formalmente, para asegurar convergencia, $\dot{V}$ debe ser menor que 0. Esta última ecuación, sugiere una elección de c de la siguiente manera:

$$c = \frac{\sin \alpha}{e} + h \frac{\theta \sin \alpha}{e \alpha} + \beta \frac{\alpha}{e}, \quad \beta > 0 \quad (3.12)$$

La elección de c se basa en anular los términos que pueden ser positivos en (3.11). Los 2 primeros términos de (3.12), anulan a los 2 primeros términos de (3.11). Pero con esto no basta, ya que $\dot{V}$ sería nulo

y no aseguraría la convergencia asintótica. Por lo que, además, se debe agregar un término que haga que $\dot{V}$ sea menor que 0. Así, agregamos el tercer término de (3.12) donde se incorpora un nuevo parámetro β .

Reemplazando (3.12) en (3.11), obtenemos:

$$\dot{V} = -\gamma\beta\alpha^2 \leq 0 \quad (3.13)$$

Lo que nos condiciona a que β debe ser mayor que cero para que el sistema tenga convergencia asintótica.

Finalmente, el comportamiento de nuestro sistema a lazo cerrado, depende de los parámetros de control γ , h y β . Así, γ es fácil de elegir de acuerdo a la saturación de velocidad que se desee, mientras que h y β se eligen de manera que:

$$h > 1, \quad 2 < \beta < h + 1 \quad (3.14)$$

Estas condiciones, las obtuvimos de un artículo ^[8] y, al cumplirse, se determina que el vehículo se aproximará al punto deseado en línea recta.

Esta lógica de control nos asegura que nuestro robot podrá alcanzar asintóticamente cualquier posición deseada partiendo de cualquier condición inicial. Se sabe que en la práctica esto no es totalmente cierto debido a limitaciones físicas en el vehículo y a las simplificaciones en su modelado.

Limitaciones del vehículo

La primera limitación práctica se debe a que el robot tiene una longitud no nula, en otras palabras, un largo determinado, que lo denominamos con la letra l y un ángulo máximo de giro, simbolizado con ψ . Debido a esto, el robot tendrá una zona restringida, y cualquier punto dentro de la misma no será alcanzable (Figura 3.24). Dicha zona, está compuesta por 2 circunferencias de radio r , el cual es inversamente proporcional a la curvatura máxima que puede tomar el robot y viene dada por:

$$c_{max} = \frac{\tan \psi_{max}}{l} \quad (3.15)$$

Por lo tanto:

$$r = \frac{1}{c_{max}} = \frac{l}{\tan \psi_{max}} \quad (3.16)$$

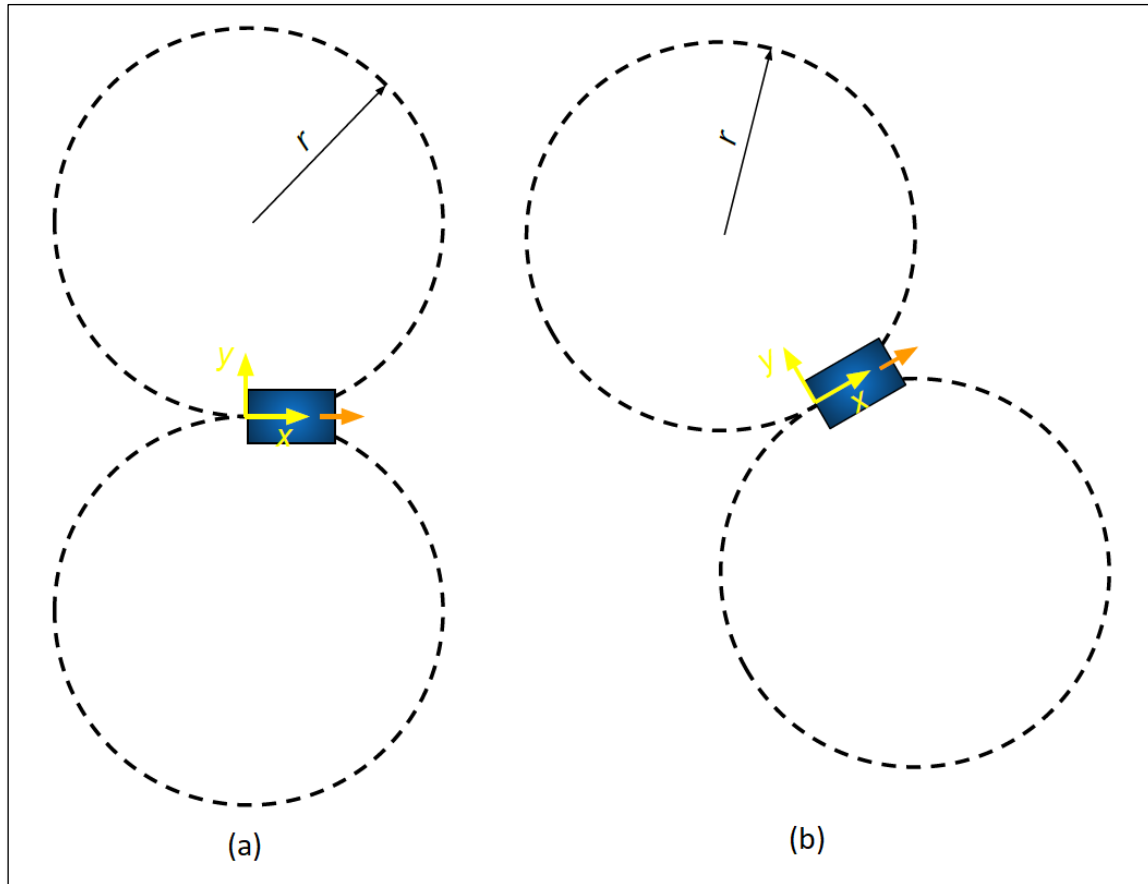


Figura 3.24: Zona restringida del vehículo. (a) Posición (0,0,0) - (b) Posición (5,5,π/6).

En nuestro caso, la longitud del robot es: $\ell = 1.65$ m y el ángulo máximo que puede girar es de $\psi = 35^\circ = 0.61$ rad, de (3.16) $r = 2.356$ m. Así, por ejemplo, nuestro robot en particular no podría llegar al punto (2.35,0) en la Figura 3.24 (a), como tampoco al (7,2) en la Figura 3.24 (b). Vale destacar que aquí no tuvimos en cuenta el ángulo de orientación deseado, sino solo las coordenadas en (x,y). Un ángulo de orientación deseado podría complicar un poco la solución, ya que, por ejemplo, en (a) no podría llegar al punto (1,0) (el cual en principio es alcanzable) con un ángulo de 90° directamente porque el ángulo de giro de la dirección y la longitud del robot no lo permiten.

Como el punto objetivo es elegido por nosotros, se tendrá especial cuidado en la designación de los mismos para no elegir algún punto no alcanzable.

¿Qué pasa entonces, si queremos que el robot alcance un punto que en principio no podría? La respuesta es simple, le brindaremos puntos intermedios que si sean alcanzables hasta transformar al punto no alcanzable en uno alcanzable. Para lograr seguir una trayectoria definida, debemos usar un algoritmo preparado para tal fin. En este caso, utilizaremos uno llamado **“path tracking”**. Este algoritmo nos brindará un camino (*path*) que el robot deberá seguir hasta alcanzar el punto final o deseado.

Ahora bien, observando la ley de control de la curvatura, según la ecuación (3.12), se puede notar que c es inversamente proporcional a la distancia e que hay entre la posición actual del robot y la deseada, así al darle un punto muy lejano, e es muy grande, c será pequeña y lo mismo ocurrirá con ψ , de

acuerdo a (3.9). Pero, ¿Qué problema hay con esto? Si el robot está orientado hacia el punto deseado, no tendremos problemas (Figura 3.25). Pero, si se encuentra en dirección opuesta (el peor caso sería con un ángulo de diferencia de π y una distancia e infinita), el robot comenzará su recorrido con poca curvatura debido al gran valor de e y tenderá a alejarse cada vez más, lo que produce un incremento de e y una disminución de c y, por ende, esto hará que el robot tienda a enderezar su dirección. Si no fue suficiente para que quede orientado hacia el punto deseado, ocurrirá que el robot se alejará más y más, sin poder alcanzar el objetivo. Esto se ve ilustrado en la Figura 3.26.

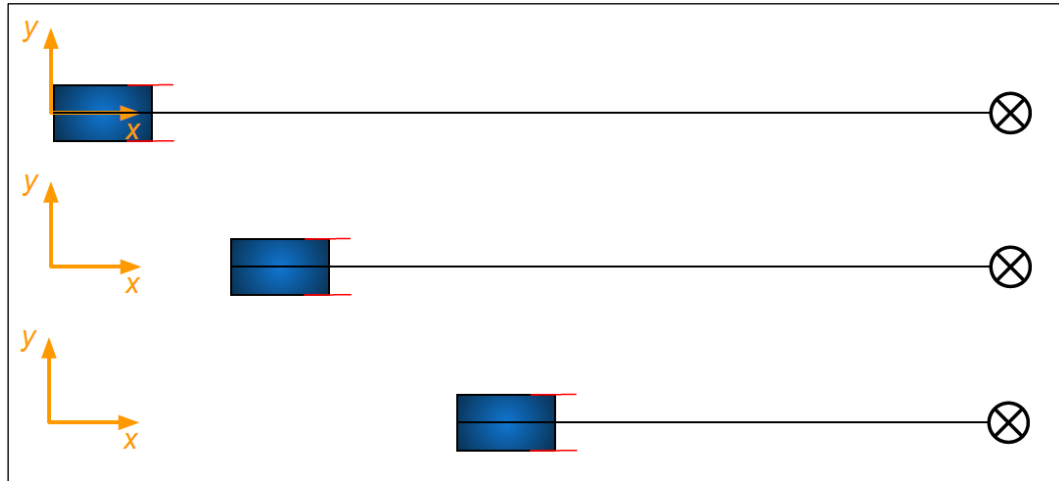


Figura 3.25: Robot orientado hacia el punto deseado.

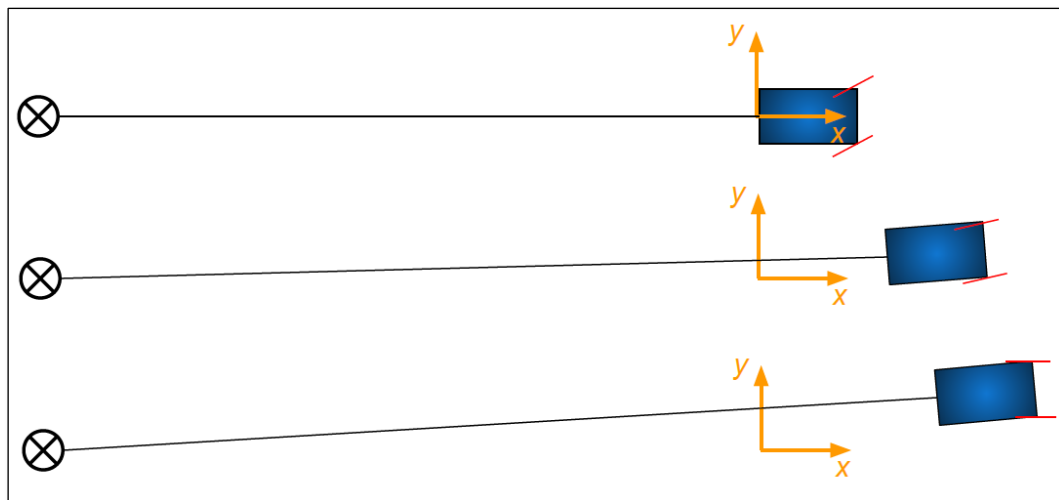


Figura 3.26: Robot orientado en dirección opuesta al punto deseado.

En las imágenes, se puede observar el comportamiento del robot frente a las situaciones antes mencionadas. En la primera secuencia (Figura 3.25), alcanza al punto satisfactoriamente ya que está orientado hacia el punto objetivo. En la segunda (Figura 3.26), el punto objetivo está en la misma dirección pero el robot no está “mirando” hacia él, en otras palabras, está en sentido totalmente opuesto al que debería. Entonces, es muy probable que, si el punto objetivo está muy alejado, el robot no pueda girar ya que la curvatura c no es lo suficientemente grande, provocando que el robot se aleje del punto objetivo incrementando su distancia con éste.

El problema anterior está asociado a un juego de parámetros y su relación, estos son β , e , y h , ya que α y θ se encuentran entre $(-\pi, \pi]$ y las funciones trigonométricas coseno y seno entre $[-1, 1]$. Así, si se eligen h y β lo suficientemente grandes con respecto a e , el inconveniente se solucionaría, pues c sería grande y en algún momento el robot quedaría orientado hacia el punto deseado hasta alcanzarlo eventualmente. La cuestión ahora es, ¿Cuánto más grande habría que elegir estos parámetros? Y peor aún, ¿Qué ocurre cuando un nuevo punto deseado está mucho más lejos que el anterior? Pues se deberán modificar los parámetros del control en función de la distancia e .

Además, si se tienen puntos lejanos, existen infinitas posibles trayectorias para alcanzar al punto objetivo y no se tiene control del trayecto a seguir. En el caso de querer ir entre surcos, podría ser que el robot pise a los mismos arruinando el cultivo. Estas dos circunstancias se ven reflejadas en las figuras Figura 3.27 y Figura 3.28 respectivamente. En la Figura 3.27 solo mostramos dos caminos posibles (de los infinitos) que podría realizar el robot, en principio la trayectoria en azul es más coherente que la trayectoria en rojo pero no podemos asegurar que el robot hará la azul y no la roja.

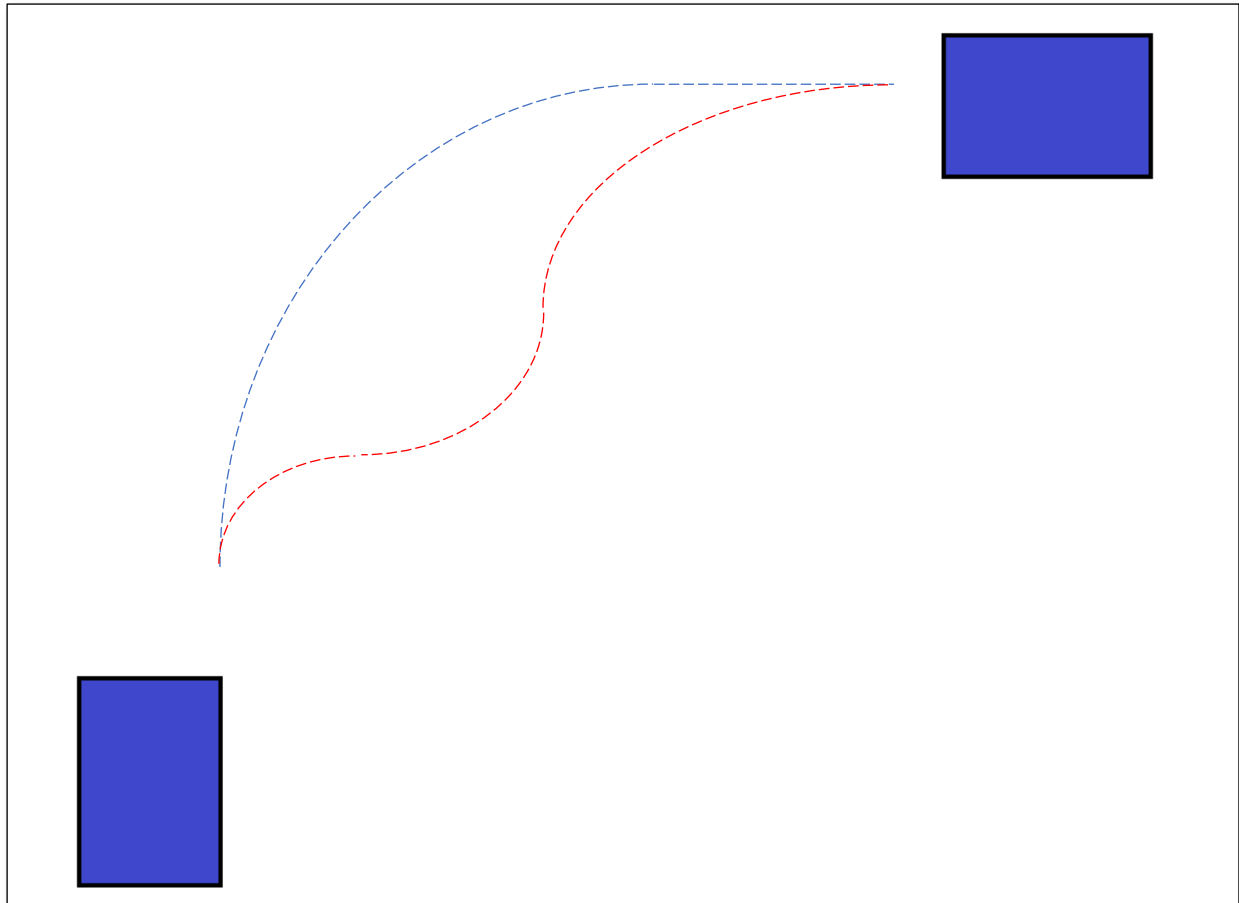


Figura 3.27: Dos trayectorias posibles para un mismo punto objetivo.

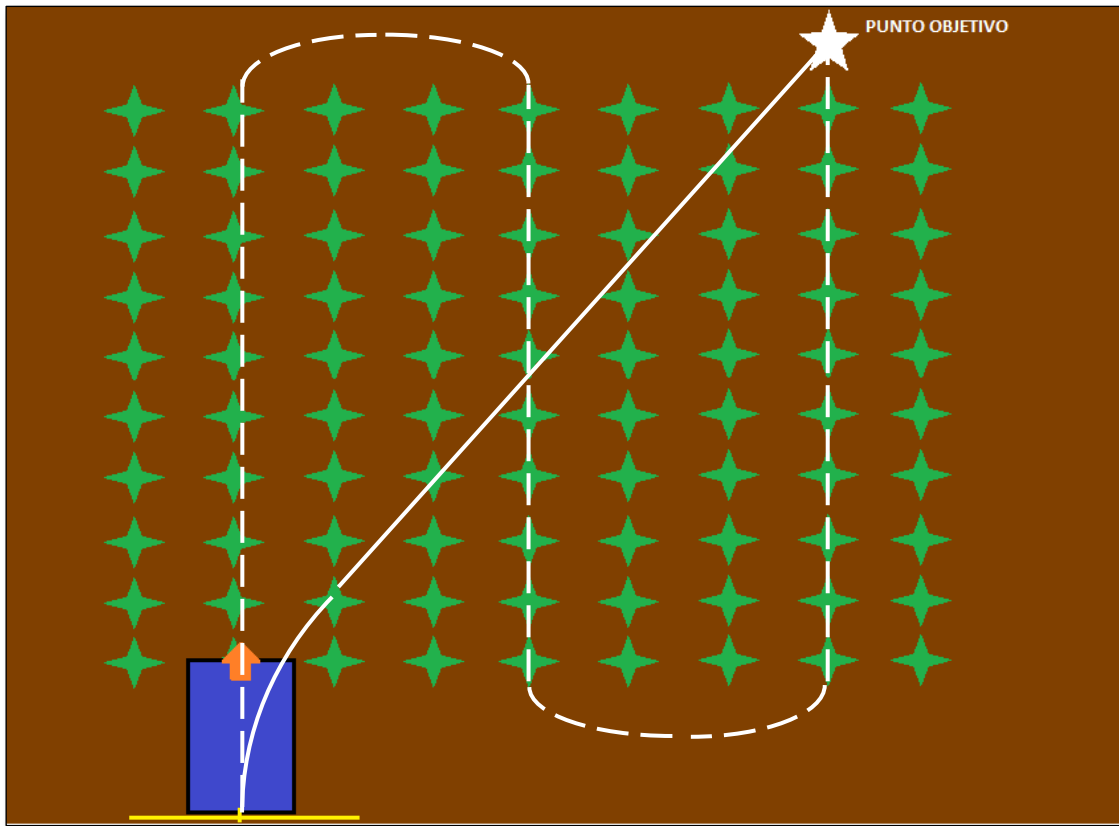


Figura 3.28: Ejemplo de una trayectoria errónea en campo cultivado.

En la Figura 3.28, se observa al robot dentro de su aplicación, o sea, en un campo cultivado por soja. El mismo deberá recorrerlo con la trayectoria en línea de puntos cubriendo así cada surco, pero si no existe un algoritmo “*Path tracking*” no podemos asegurar que el robot siga esta trayectoria hasta alcanzar el punto objetivo. Una posible trayectoria a realizar es la marcada en línea continua, aunque destacamos que no sabemos si hará esta u otra por lo visto anteriormente en la Figura 3.27 (infinitas soluciones).

Este último hecho es suficiente para justificar la necesidad de un algoritmo “*Path tracking*”, ya que es notable el hecho de que el robot no cumple con su objetivo principal de eliminar malezas, sino que además, arruina el cultivo.

3.2.4. Path Tracking

Uno de los algoritmos de trayectoria más sencillos y conocidos es el denominado “*Follow the Carrot*” (seguir la zanahoria), éste se basa en darle puntos objetivos intermedios (*waypoints*). La idea es mantener un punto objetivo (*waypoint*) que se desplaza sobre la trayectoria deseada a una distancia fija respecto al vehículo, equivalente a lo que se hace en las carreras de galgos.

Así, el robot recibirá un nuevo punto objetivo, calculado por nosotros, que esté a una distancia fija del robot sobre la recta que une al robot con el punto objetivo verdadero.

Con esta implementación, se logra tener siempre una distancia fija y, por la ley de control de velocidad (3.7), esta velocidad será fija también hasta que el último punto objetivo coincida con el final,

ya que γ es un parámetro fijo. Con este algoritmo de **path tracking**, tendríamos entonces una solución que nos permite seguir una trayectoria previamente establecida para ir de un punto inicial a uno final. En otras palabras, independizamos en parte a la curvatura de e y le damos más peso a los parámetros y ángulos en juego en esta ley de control.

A continuación, ilustramos un ejemplo (Figura 3.29).

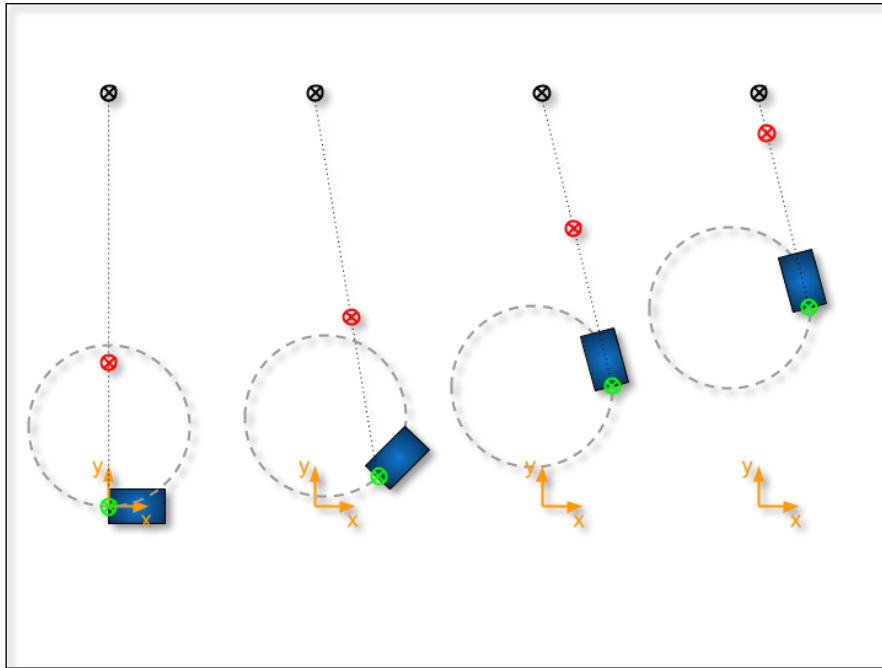


Figura 3.29: Secuencia de puntos objetivo en "Path Tracking" (Parte A).

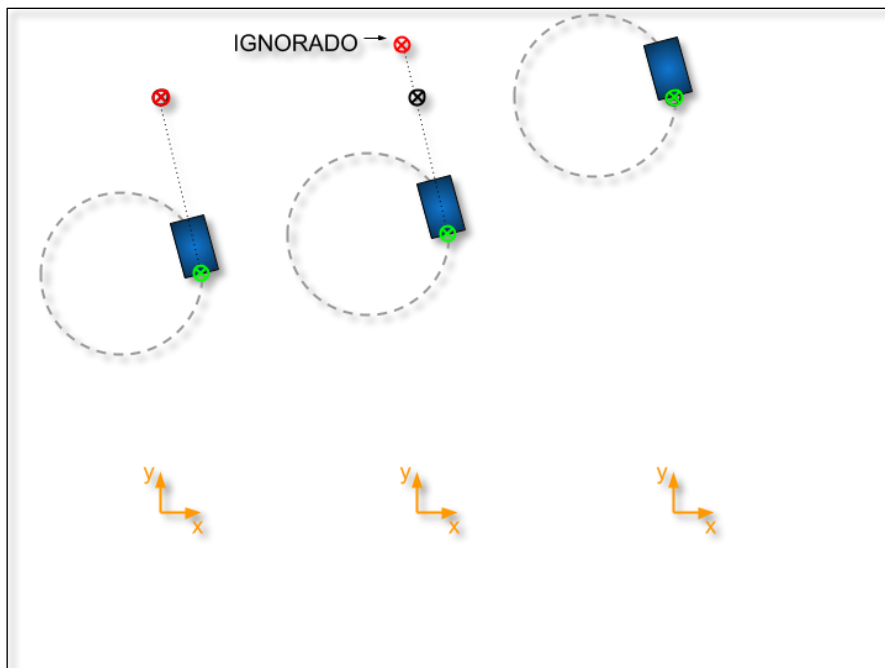


Figura 3.30: Secuencia de puntos objetivo en "Path Tracking" (Parte B).

El punto objetivo que el robot debe alcanzar se distingue con una cruz negra, el robot inicia desde el punto (0,0) con un ángulo de orientación $\phi = 0$ rad. A continuación, se observa al punto objetivo y se genera un punto objetivo intermedio (*carrot*) marcado con una cruz roja, el cual se encuentra a una distancia fija de las coordenadas del robot marcadas con cruz verde. Hacemos un paréntesis en esta situación, pues, vemos que el punto objetivo intermedio se encuentra dentro de la zona restringida por las características físicas del robot. Si bien el sistema de control lo ve como el objetivo final, se sabe que es un punto intermedio por lo que no es importante que esté en la zona restringida; y, aunque de antemano sabemos que el robot no podrá alcanzar este punto, lo intentará. Este intento obliga al robot a girar su dirección completamente hacia la izquierda recorriendo la circunferencia de radio mínimo (marcada en líneas de puntos en color gris).

En el siguiente momento vemos que el robot se encuentra sobre la circunferencia, pero se ha movido y sus coordenadas ya no son (0,0) ni su ángulo de orientación es $\phi = 0$ rad. Ahora, recalculamos el punto objetivo intermedio (con las mismas condiciones antes mencionadas). Así, el robot intentará alcanzar a este nuevo punto intermedio.

Pasemos al momento posterior, se observa que el robot continuó sobre la circunferencia de radio mínimo, pero ahora se encuentra orientado hacia el punto objetivo final. El nuevo punto objetivo intermedio se modificó y estará en esta dirección, por lo que el robot debe anular su dirección de giro para lograr ir en la misma dirección. El robot intentará alcanzar al nuevo punto objetivo intermedio.

Siguiendo este procedimiento, al acercarse al punto objetivo final, llegará un momento en el que el punto objetivo intermedio coincida con éste o que la distancia al punto objetivo final sea menor que la distancia al punto intermedio. Es aquí, donde no se le proporcionarán más consignas de puntos objetivos intermedios, sino el punto objetivo final. Así, la velocidad que responde proporcionalmente a la distancia entre el robot y el punto al que desea ir, irá disminuyendo y el robot se detendrá al alcanzar el punto objetivo, es decir, cuando la distancia sea nula. Esto se contempla mejor en la Figura 3.30, donde el punto objetivo intermedio coincide con el final. Luego, la distancia al punto objetivo intermedio es mayor que la distancia al punto objetivo final, por lo que es ignorado. Finalmente, el robot llega al punto objetivo final.

En lo que respecta a la elección de la distancia a la que estarán los puntos intermedios, debe tenerse en cuenta que la misma no puede ser ni muy grande ni muy pequeña. Si esta distancia es muy chica, el robot realiza trayectorias oscilantes hasta poder estar sobre la trayectoria establecida con la dirección correcta. Si tenemos presente la ley de control de la curvatura c se observa que a menor e mayor será c , lo que se traduce en una ley de control muy brusca que haría oscilar la dirección (Figura 3.31), hecho que no es para nada deseable. Por otro lado, una distancia muy grande provocará la situación opuesta, teniendo una ley de control muy suave por lo que se tardaría mucho en alcanzar la trayectoria correcta (Figura 3.32).

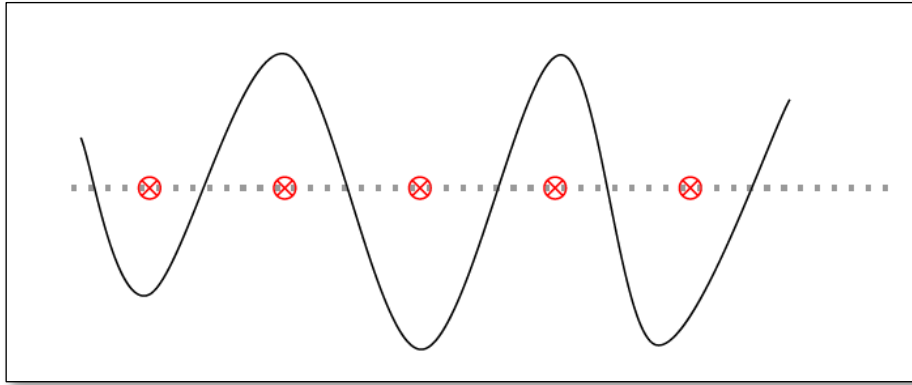


Figura 3.31: Ley de control brusca que produce oscilación.

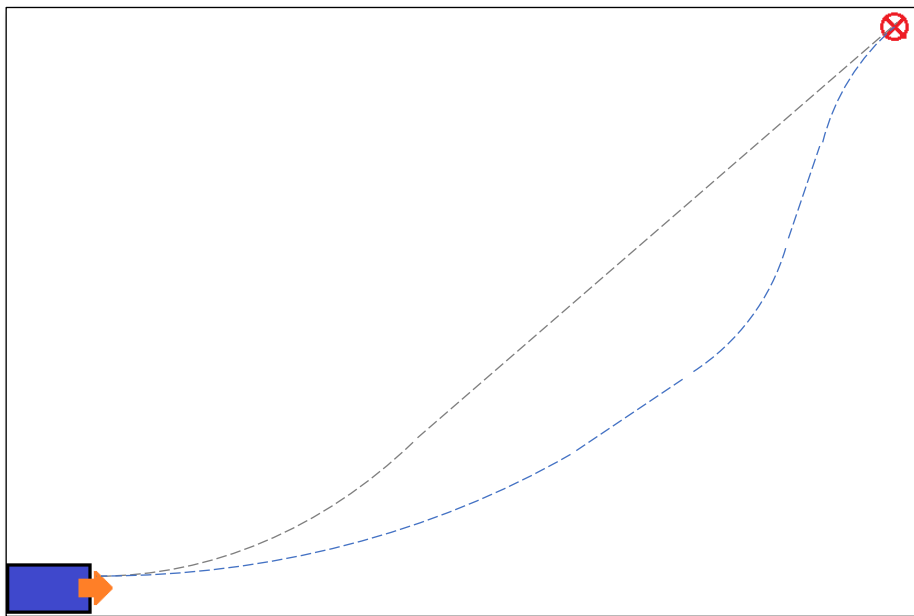


Figura 3.32: Ley de control demasiado suave.

Esta situación, en la que la distancia entre el robot y el punto intermedio es muy grande, queda ilustrada en la Figura 3.32. En gris, se observa un recorrido ideal que debería realizar el robot, mientras que en azul observamos una posibilidad de recorrido del mismo. Si bien no es seguro que realice exactamente esa forma de trayectoria (azul), lo que se quiere destacar es que la ley de control será lenta.

Esto puede justificarse recordando la ecuación de la curvatura c (3.12) que incide directamente en el ángulo de giro de dirección del robot. Como esta ley de control depende de una manera inversamente proporcional a la distancia entre robot y punto intermedio, al ser muy grande esta, c será pequeña y por lo tanto lo será la dirección de giro del robot, haciendo cada vez más lento el control a medida que sea mayor la distancia en cuestión.

Además, hasta ahora, analizamos trayectorias con un único punto objetivo final, pero puede haber más de uno y, al adoptar una distancia muy grande de e , podríamos eludir algunos de ellos, lo que no resolvería el problema principal de tracking.

Otro factor que hay que tener en cuenta es la velocidad del robot. Si bien dijimos que es fija, ésta no puede ser demasiado grande debido a que a mayor velocidad se acentúan los problemas de este ***path tracking***. Esto nos determinará el parámetro γ .

En la sección 5.5 se muestra el desarrollo en nuestro proyecto particularmente (elección de parámetros, simulación e implementación en código C).

Introducido el objetivo de este proyecto, la teoría de los problemas que se deberán resolver y las limitaciones presentes, pasaremos a hablar sobre las cuestiones prácticas. Comenzaremos presentando el equipamiento que utilizamos.

4. Equipamiento utilizado

En este proyecto se trabaja sobre el prototipo de un robot previamente desarrollado al cual se le realizarán cambios significativos. Por este motivo, cada una de las partes actuales del prototipo será reutilizada o reemplazada. A continuación, se detalla por un lado los componentes adquiridos para sustituir a aquellos que corresponde para lograr los objetivos propuestos. Por otro lado, se hace mención de aquellos que son de interés relevantes en el transcurso del desarrollo.

4.1. Rueda y controlador

Debido a los problemas mencionados en la sección 2 y continuando con la metodología utilizada en el proyecto global, donde se implementaron kits de bicicleta eléctrica para la tracción, se adquirieron dos nuevos kits completos con el agregado de 2 drivers extras. En la arquitectura final, quedaron dos motores de 750W (traseros), dos motores de 350W (delanteros) heredados del proyecto anterior y los 4 drivers adquiridos.

El kit, mostrado en la Figura 4.1, cuenta con motor un BLDC de 750W, un driver de los anteriormente mencionados y algunos elementos extras que no serán de utilidad en el presente proyecto (manillar acelerador, palanca de freno, sensor de pedaleo, accesorios para montaje, etc).



Figura 4.1: Kit e-bike adquirido.

4.2. Microcontrolador

Como unidad de procesamiento se utilizó una placa de desarrollo “Freescall FRDM-K64F”, siendo esta una plataforma constituida bajo la arquitectura ARM Cortex-M4 con un microcontrolador MK64FN1M0VLL12 de la serie Kinetis K. Este microcontrolador posee una frecuencia máxima de funcionamiento de 120 MHz, 1 MB de memoria flash, 256 kB de RAM, un controlador USB de alta velocidad, un controlador Ethernet, un controlador host digital seguro y algunas entradas/salidas analógicas y digitales.

En la Figura 4.2 se muestra un diagrama con los principales componentes.

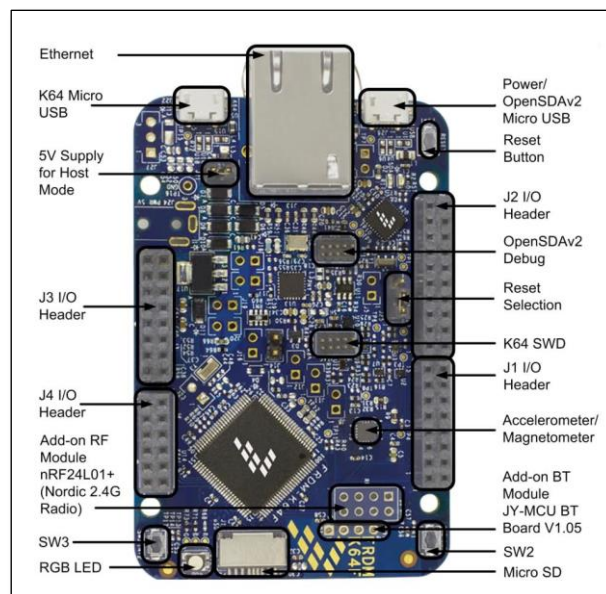


Figura 4.2: Principales componentes K64F.

Principales características

Rendimiento

- 120 MHz ARM® Cortex®-M4 con instrucciones DSP y unidad de punto flotante.

Memorias e Interfaces de Memoria

- Hasta 1 MB memoria programable flash y 256 KB de RAM.
- Hasta 28 KB FlexNVM y 4 KB FlexRAM en dispositivos con FlexMemory.
- Interfaz de Bus Externa FlexBus.

Periféricos del Sistema

- Múltiples modos bajo consumo. Unidad de reactivación de bajo consumo.
- Unidad de Protección de Memoria.
- Controlador DMA 16 Canales.

Módulos de Seguridad e Integridad

- Hardware Módulo CRC.
- Hardware Generador de Números Aleatorios.
- Hardware de Encriptación Compatible con DES, 3DES, AES.

Algoritmos MD5, SHA-1 y SHA-256

- Número de Identificación Único (ID) de 128 bits por chip.

Módulos Analógicos

- 2 ADC 16-bit SAR.
- 2 DAC 12-bit.
- 3 Comparadores Analógicos (CMP).

- Monitor de Vigilancia Externo (Watchdog).
- Voltaje de Referencia.

Controlador Ethernet con interfaz MII y RMII

- Controlador On-the-Go USB.
- Módulo de Red de Área de Controlador (CAN).
- 3 Módulos SPI.
- 3 Módulos I2C.
- 6 Módulos UART.
- Controlador de host digital seguro (SDHC).
- Módulo I2S.

Temporizadores

- 2 Temporizadores de 8 Canales (PWM/Motor).
- 2 Temporizadores de 2 Canales (PWM/Quad decoder).
- Temporizadores IEEE 158.

- PIT de 32 bits y Temporizadores de Bajo Consumo de 16 bits.
- Reloj de Tiempo Real (RTC).
- Bloque de Retardo Programable.

Relojes

- Oscilador de Cristal de 3 a 32 MHz y 32 kHz.
- PLL, FLL y Múltiples Osciladores Internos.
- Reloj de Referencia Interno de 48 MHz (IRC48M).

Características de Funcionamiento

- Rango de Voltaje: 1.71 a 3.6V.
- Rango de Voltaje de Escritura del Flash: 1.71 a 3.6V.
- Rango de Temperatura: -40 a 105°C.

Diagrama Entrada/Salida

En la Figura 4.3 se muestra el diagrama de pines de Entrada/Salida (Pinout).

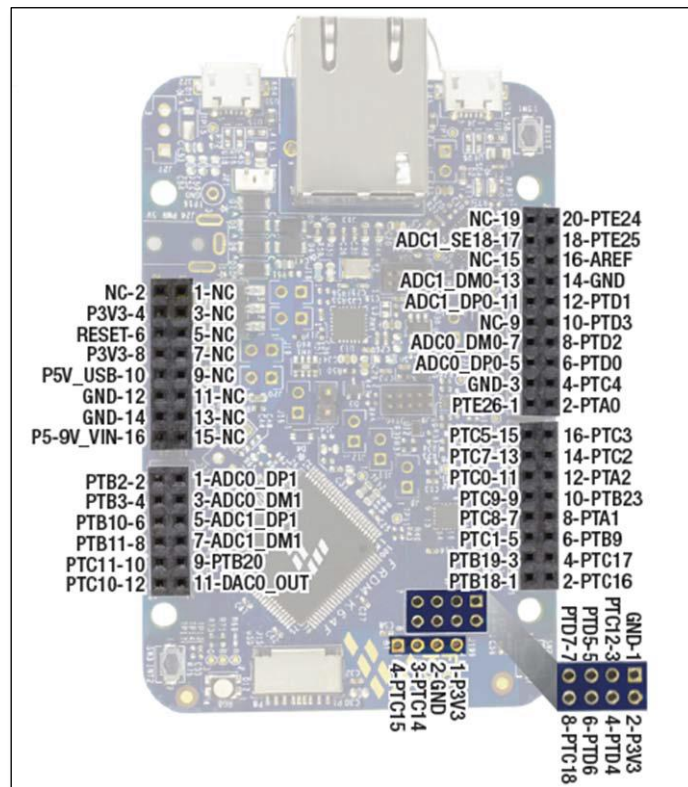


Figura 4.3: Pines de Entrada/Salida de K64F.

Compilador

Siguiendo con la línea de trabajo del proyecto anterior el software será programado en lenguaje C y el compilador a utilizar será *CodeWarrior Development V11* de la firma **Freescall**. Este compilador es muy útil ya que cuenta con una herramienta llamada “*Processor Expert*” que facilita en gran medida el proceso de programación de microcontroladores de esta firma.

También cuenta con una herramienta de depuración (*Debugger*), que nos permitirá depurar el código en tiempo real y frenarlo cuando se desee mediante *breakpoints* estratégicamente ubicados para solucionar posibles errores en la etapa de desarrollo del programa.

4.3. Mando a distancia

El mando a distancia (control remoto), genera una señal PWM tanto para la señal de dirección como para la señal de velocidad a través de un potenciómetro y un gatillo respectivamente. A continuación (Figura 4.4), se muestra una ilustración del mando en cuestión.

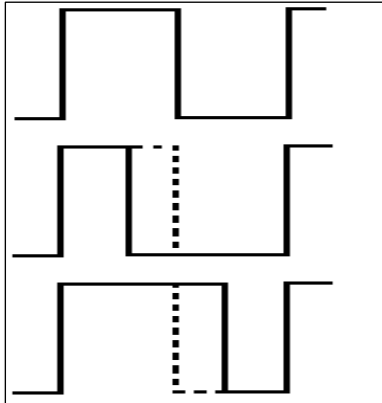


Figura 4.4: Mando a distancia utilizado.

Según sea la actuación sobre el potenciómetro o el gatillo, la señal cuadrada generada varía el ancho del pulso a partir de un punto medio. Por ejemplo, si se gira el potenciómetro a la derecha, el ciclo de la señal aumenta su tiempo en alto; por el contrario, si se gira a la izquierda el ciclo de la señal reduce el tiempo en alto.

Lo mismo sucede cuando se acciona el gatillo hacia atrás o hacia adelante.

A modo de ejemplo:



Actuador normal. Sin estimulación.

Actuador presionado. Por ejemplo: giro a la izquierda del potenciómetro.

Actuador presionado. Por ejemplo: giro a la derecha del potenciómetro.

El microcontrolador debe ser capaz de procesar esta señal PWM (0v ~ 3.3v) cuyo periodo se aproxima al milisegundo y en base al ancho del pulso proveniente de control remoto realizar la acción correspondiente.

4.4. Sensores de Corriente

Unos de los parámetros que debemos obtener del prototipo, es la corriente consumida por cada motor para luego enviar esta información a la NUC. Se eligió utilizar módulos sensores de corriente basados en el efecto Hall para llevar a cabo esta tarea.

Este módulo basado en el circuito integrado ACS712 de *Allegro MicroSystems* (Figura 4.5) permite medir la cantidad de corriente que fluye a través de un circuito de corriente alterna (AC) o corriente continua (DC). El método de medición es a través de un sensor de efecto hall que provee un voltaje de salida proporcional a la corriente que fluye en el circuito. El trayecto para la medida de corriente es por el interior del circuito integrado y se encuentra aislado del circuito de procesamiento.

El módulo viene en 3 versiones (5A, 20A y 30A), en nuestro caso se trata de el de 20A, ya que es el que mejor se adecuaba a las necesidades.

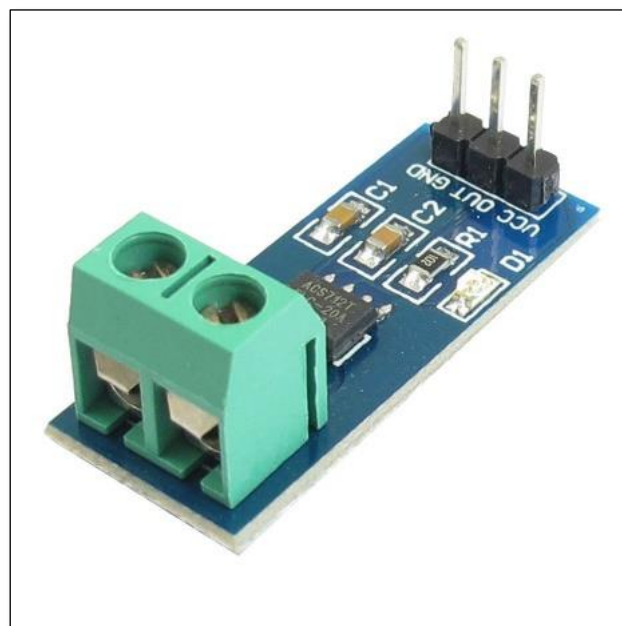


Figura 4.5: Sensor ACS712 de 20A.

Características Técnicas

- Sensor lineal de efecto hall de bajo offset.
- Alta precisión de medida debido a la cercanía del sensor de efecto hall al elemento de paso (ambos se encuentran dentro del IC).
- Baja resistencia del elemento de paso para una baja disipación de potencia (1.2 mΩ típico)
- Capacidad de sobrevivir a picos de corriente de hasta 5 veces la corriente nominal de operación.
- Las terminales del elemento conductor se encuentran aisladas electricamente (Hasta 2.1KV) por lo que evita la necesidad de aislamiento externo (optoacopladores).
- Voltaje de salida: 66mV / A.
- Voltaje de operación: 4.5V ~ 5.5V.
- Salida de tensión sin corriente: VCC / 2.
- Dimensiones PCB: 31 (mm) x14 (mm).
- 5 μs de tiempo de subida a la salida frente a un escalón de corriente.
- Ancho de banda 80 kHz.
- Error Total Salida: 1.5% at TA = 25°C.
- Resistencia interna: 1.2 mΩ.
- Mínimo voltaje de aislamiento entre pines 1-4 a pines 5-8: 2.1 kVRMS.
- Sensibilidad de salida: 66 a 185 mV/A. 100 mV/A típicos para el de 20A.

4.5. Stepper Motor

Como se mencionó con anterioridad, el motor paso a paso usado es “*Stepper Motor MACHI-NET 24HS3021*”, en la Figura 4.6 se muestra la hoja de datos con los parámetros principales del mismo.

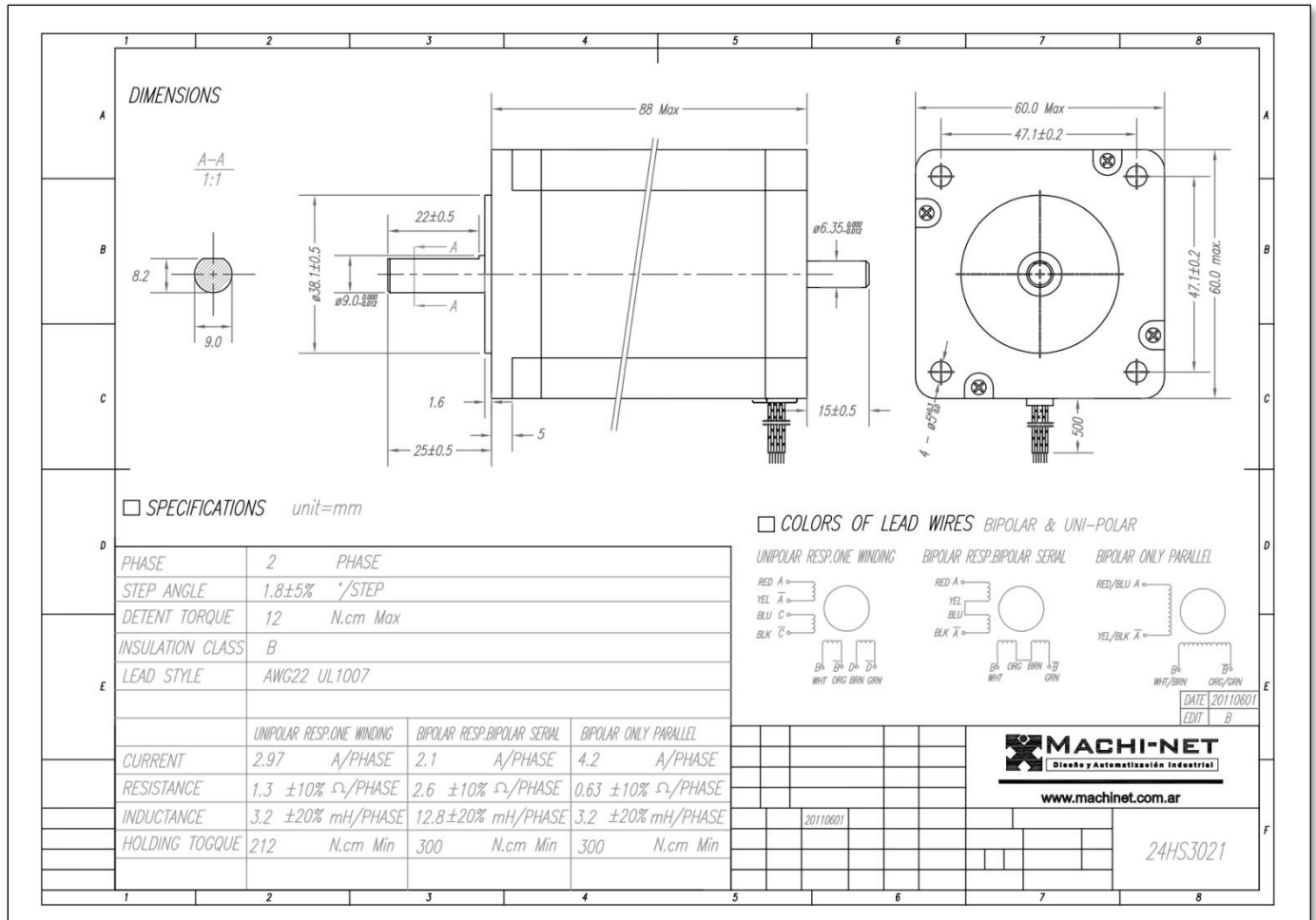


Figura 4.6: Hoja de datos del Motor paso a paso comprado.

4.6. Driver Stepper Motor

Para el correcto control del motor paso a paso se utilizará un *Stepper Motor Driver M542H*, el cual es un driver basado en *Microstepping*. El mismo resulta apropiado para controlar motores paso a paso híbridos tanto de 2 como de 4 fases. Se escogió este driver ya que el mismo implementa la técnica *bipolar constant-current chopping*, la cual, como se menciona en la introducción, posee la ventaja de lograr mayor torque a altas velocidades, en comparación con otros drivers como por ejemplo los *L/R*.

Principales características

A continuación, se muestra una foto del driver (Figura 4.7) y sus principales características.

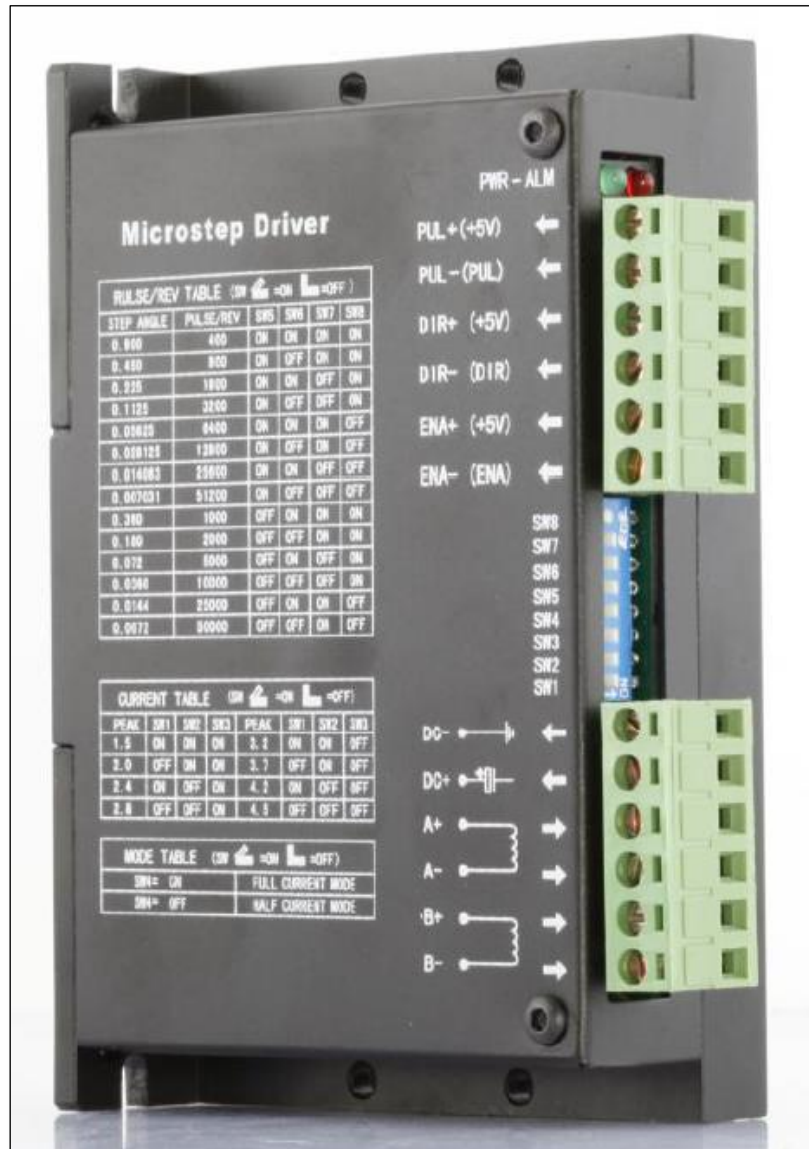


Figura 4.7: Stepper Motor Driver M542H.

- Bajo costo y buen torque a altas velocidades.
- Soporta sobretensiones de alimentación mayores a 100 VDC.
- Corriente de salida mayor a 4,5 A.
- Frecuencia de pulsos mayor a 300 KHz.
- Tecnología de control de corriente de tres estados.
- Adecuado tanto para motores de 2 fases como de 4 fases.
- Protecciones contra sobre voltajes y cortocircuitos.
- 16 resoluciones de microstepping seleccionables.
- DIP switch para setear entre 8 diferentes valores de corriente de salida.
- Entradas aisladas mediante octoacopladores.
- Temperatura de funcionamiento: -20°C a 65°C .
- Máxima temperatura de operación: 70°C .

Especificaciones eléctricas

Parámetros	DM542H			
	Mínimo	Típico	Máximo	Unidad
Corriente de Salida	1.00	-	5.00 (3.0 RMS)	A
Tensión de Entrada	+20	+50	+100	VDC
Corriente Señal Lógica	7.00	10	16	mA
Frecuencia Pulso Entrada	0.00	-	300	KHz
Ancho Pulso	2.50	-	-	μs
Voltaje del Pulso	5.00	-	24	VDC
Resistencia de aislamiento		500.00		MΩ

Tabla 4.1: Características eléctricas del driver ($T_j=25^{\circ}\text{C}$)

Esquema de conexión

Como se puede observar en la Figura 4.8 el driver *M542H* posee dos secciones de pines de entrada. La primera de ellas para conexión de las señales de control (*Pulse*, *Enable* y *Direction*) y la segunda para conexión de las señales de potencia, como lo son la alimentación y las fases del motor paso a paso.

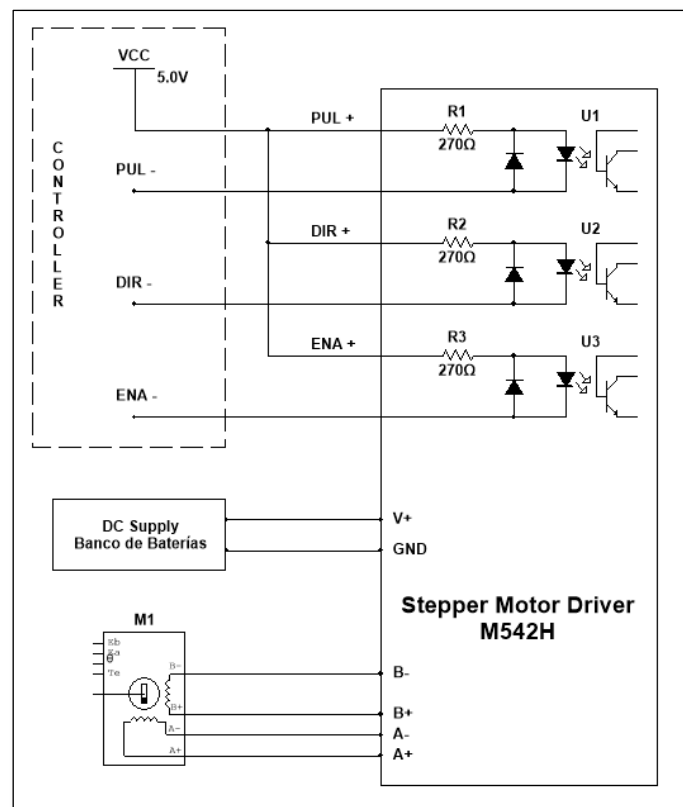


Figura 4.8: Esquema de conexión.

Señales de control

En la Tabla 4.2 se muestran las señales que se utilizan para comandar el driver junto a una breve descripción de cada una.

Señal	Detalle
PUL+ (5V)	Señal de pulsos: por cada pulso se avanza un paso en el sentido de rotación seteado.
PUL- (PUL)	
DIR+ (5V)	Señal de dirección: señal lógica que determina el sentido de giro del motor.
DIR- (DIR)	
ENA+ (5V)	Señal de habilitación: Esta señal se usa para habilitar o deshabilitar el driver.
ENA- (ENA)	

Tabla 4.2: Señales de control del driver.

4.7. Encoder Absoluto

Si bien, la posición angular del motor paso a paso encargado de la dirección del vehículo se podría estimar contando la cantidad de pulsos de la señal PULSE y luego multiplicarla por el ángulo girado por cada pulso. Esto sería una solución errónea, ya que el prototipo debe estar diseñado para trabajar en terrenos hostiles, es decir, terrenos que provoquen vibraciones que modifiquen la posición angular del motor sin haberle enviado un pulso.

Por este motivo, se decidió utilizar un encoder para tener más precisión sobre la posición angular del motor. Se utilizó un encoder Absoluto (*OMRON E6CP-A*) de 8 bits. Este permite codificar la posición angular del eje en una variable de 8 bits. Esta variable deberá ingresar al microcontrolador para procesarla por lo que, como se detalló anteriormente, se destinarán 8 pines del micro para tal fin.

4.8. Computadora utilizada

Para poder procesar la información se utiliza una MINI-PC intel® NUC kit NUC6CAYH (Intel Celeron J3455 CPU, quad-core 2,3 GHz and 8 GB DDR3 RAM Memory) con Ubuntu 16.04. Con la inclusión de un disco de estado sólido con el objetivo de minimizar los tiempos de escritura que pudieran dificultar la obtención de datos. Su tamaño es reducido, aproximadamente 11x11x6 cm. Dicha PC se muestra en la Figura 4.9.



Figura 4.9: MINI-PC intel® NUC.

4.9. Contenedores y soportes

Para lograr una mejor presentación y protección se decidió diseñar contenedores y soportes para los distintos elementos que no se montan dentro del gabinete de control, a su vez, estos diseños contribuyen con la idea de hacer una solución modular (ya que son fácilmente desmontables).

Más adelante, en la sección 9.1.1, se muestra en mejor detalle el diseño y los resultados finales.

4.10. Gabinete

Para montar todo el sistema de control se utilizó un gabinete estanco marca “GENROD” de la serie S9000 (Figura 4.10). Las dimensiones son 300mm de ancho, 450mm de alto y 150mm de profundidad.

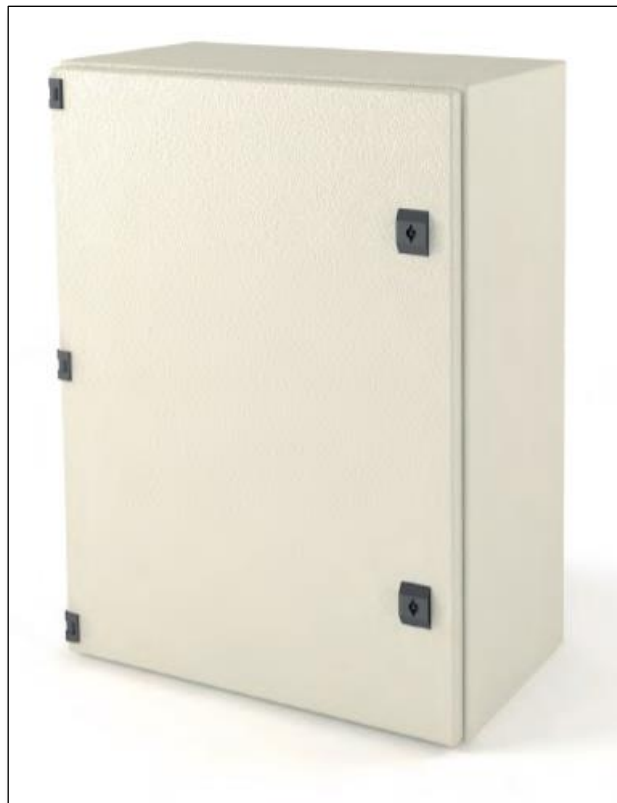


Figura 4.10: Gabinete GENROD serie S9000.

4.11. Fichas

Para lograr un sistema modular, es decir, que sea fácilmente desmontable, se adquirieron unas fichas de 8 vías marca “Portafich” usadas en la industria automotriz (Figura 4.11 y Figura 4.12). Tienen la particularidad de ser muy robustas y estancas, características muy favorables teniendo en cuenta el ambiente en el que trabaja el robot.

Estas fichas se utilizaron para alimentar los motores (bobinados y sensores de efecto Hall). Se colocó un juego en cada rueda para que, en caso de algún problema con alguna de ellas, se pueda reemplazar con facilidad. También se colocó un par a la altura de cada driver por el mismo motivo (facilidad de reemplazo en caso de que falle alguno).



Figura 4.11: Ficha hembra de 8 vías.



Figura 4.12: Ficha macho de 8 vías.

5. Tareas desarrolladas

Encaramos las tareas a partir del reconocimiento de las principales partes del robot y el estudio de los nuevos componentes adquiridos de manera de entender su funcionamiento y características principales.

5.1. Relevamiento de la distribución física del sistema

En la Figura 5.1 se muestra un esquema con las ubicaciones físicas de los distintos gabinetes, componentes de interés y sus interacciones; junto a una descripción de las distintas partes.

Cabe aclarar que este no es un esquema completo del prototipo en sí, sólo se visualizan los componentes que consideramos más relevantes para el control de tracción y dirección.

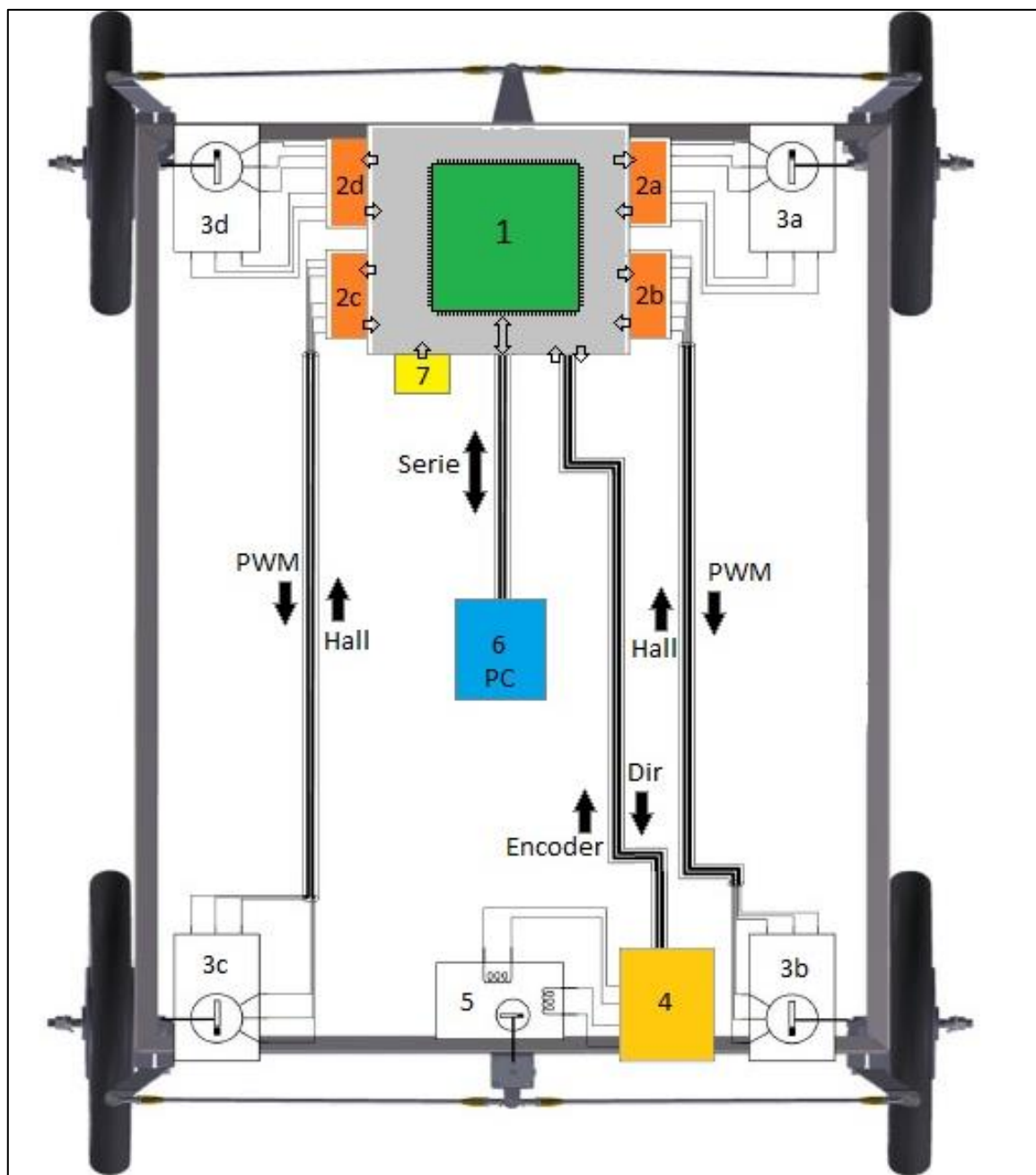


Figura 5.1: Esquema del sistema de control.

1. *Gabinete de control*: Contiene la placa de desarrollo Freescale K64F, la cual posee un microcontrolador MK64FN1M0VLL12 y una placa de adaptación de señal que actúa como interfaz entre el microcontrolador y los drivers. En este gabinete se encuentra la lógica encargada principalmente de controlar la tracción y dirección del robot en base a las referencias provenientes desde el control remoto o desde la PC.

2. *Drivers*: Varían la tensión de las fases del motor según la información recibida desde el gabinete de control, estableciendo la velocidad y el sentido de giro de los motores.

- a. *Driver Brushless DC motor rueda delantera derecha.*
- b. *Driver Brushless DC motor rueda trasera derecha.*
- c. *Driver Brushless DC motor rueda trasera izquierda.*
- d. *Driver Brushless DC motor rueda delantera izquierda.*

3. *Motores*: Proporcionan la tracción del robot, controlados a través sus respectivos drivers según las referencias establecidas desde el gabinete de control.

- a. *Brushless DC Motor rueda delantera derecha.*
- b. *Brushless DC Motor rueda trasera derecha.*
- c. *Brushless DC Motor rueda trasera izquierda.*
- d. *Brushless DC Motor rueda delantera izquierda.*

4. *Driver Stepper Motor*: Es el encargado de alimentar las bobinas del motor paso a paso (5) según se lo indiquen las señales Pulse, Enable y Direction provenientes del gabinete de control.

5. *Stepper Motor MACHI-NET 24HS3021 y caja reductora(10/1)*: La rotación de su eje establece la dirección del robot por medio de un sistema de caja reductora y varillas vinculadas a las columnas de dirección en las ruedas delanteras.

6. *Gabinete de PC*: En este gabinete encontramos la MINI-PC intel® NUC, la cual interactúa a través de una comunicación serie USB con la placa *freescale K64F* y es allí donde se llevará a cabo el desarrollo del software abocado al seguimiento de trayectorias.

7. *Emisor-Receptor de radiofrecuencia (2.4 GHz)*: Recibe señales de radio emitidas por el control remoto, correspondientes a consignas de dirección y velocidad de avance del robot. Luego de recibirlas las pone a disposición en pines de entrada de la placa de desarrollo Freescale K64F.

5.2. Esquema de control

Para poder lograr que el robot alcance una velocidad deseada y la mantenga, se debió realizar un control de velocidad del mismo.

Por lo tanto, con el objetivo de controlar el sistema, se aplicó un esquema de control muy conocido, de realimentación negativa, el cual puede observarse en la Figura 5.2. En esta vemos que hay una condición de velocidad deseada (*Set point de velocidad*), un controlador que, en base al error de velocidad manipula la tensión que se le aplica al driver, el sistema (*driver y motor*) y un bloque del cálculo de velocidad que se realiza gracias a las señales HALL. Finalmente, se cierra el lazo realimentándolo con el cálculo del error de la velocidad mencionada.

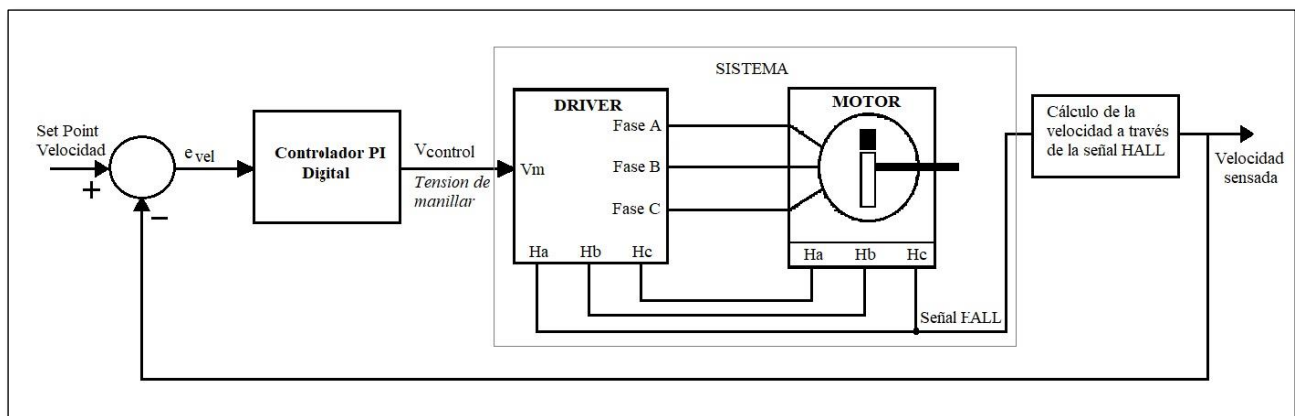


Figura 5.2: Lazo de control detallado.

Conociendo entonces los distintos sistemas, sensores, y el controlador que componen a todo el sistema de control, se comentará sobre cada uno de ellos.

5.2.1. Motor

Con el objetivo de sortear el problema de falta de tracción del robot, se utilizarán 2 motores traseros Brushless DC de 750W reemplazando a los actuales de 350W, mientras que para los motores delanteros se mantendrán los Brushless DC de 350W. Se pudo observar, tanto en los nuevos equipos como en los actuales, que cuentan con terminales para el conexionado de las 3 fases de potencia y un terminal con 5 señales vinculado a los sensores hall que permiten operar el equipo según lo visto en la sección 3.1.1. Como mencionamos anteriormente, se unificará el cableado a través de un único conector para facilitar el montaje.

Otro detalle a tener en cuenta es el número de polos de estos motores. Puede verse fácilmente, en la Figura 5.3, que cuentan con 23 pares de polos.

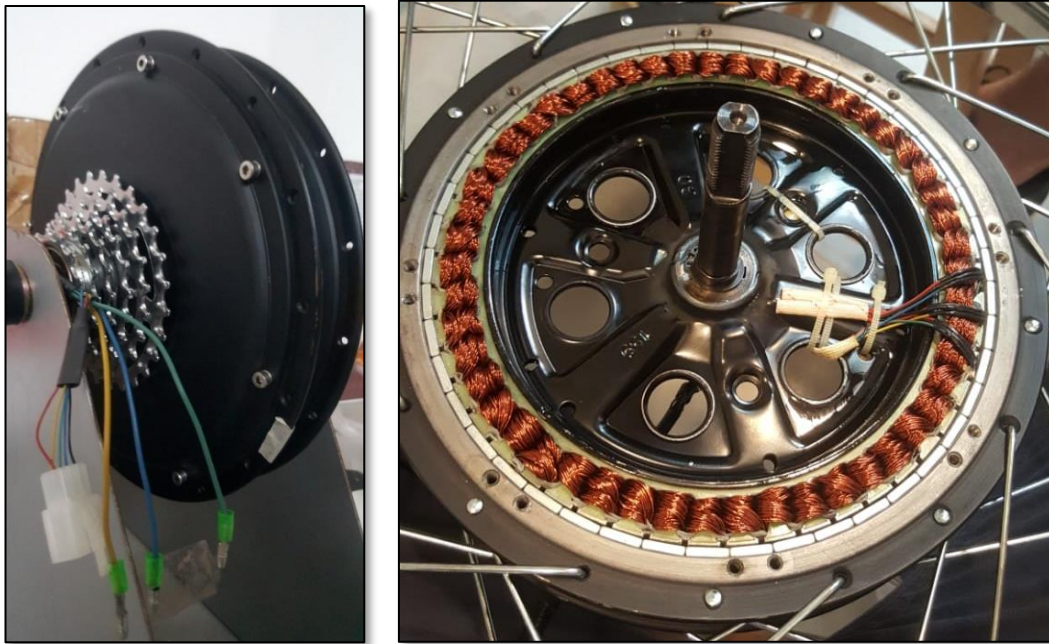


Figura 5.3: Motor BLDC.

5.2.2. Drivers

La elección de los drivers con la virtud de frenado regenerativo e inversión de marcha trae una ventaja a destacar. Una ventaja, es la posibilidad de giro de los motores en ambos sentidos permitiendo que el robot se desplace hacia adelante o hacia atrás, y otra, es el aprovechamiento energético del frenado regenerativo.

La primera nos permite sustituir el actual sistema de reversa que consiste en conmutar dos fases de cada motor a partir de 2 módulos de 8 relés optoacoplados. Esta solución era poco conveniente y problemática, ya que los relés tienen contactos y piezas móviles que se desgastan con el uso y el tiempo.

Por lo tanto, podremos deshacernos de estos módulos, así como también de los circuitos adaptadores de señal de disparo de los Relés. Esta solución simple y sencilla es notablemente favorable, reduciendo la electrónica implementada ya que en principio solo basta una señal del microcontrolador para invertir la marcha, liberando espacio físico en el gabinete de control y fundamentalmente eliminando por completo la interferencia electromagnética que genera el antiguo método.

La segunda es sumamente importante ya que el prototipo no cuenta con ningún sistema de frenado y, además de usarse como sistema de frenado, podemos almacenar la energía eléctrica generada para su reutilización.

Hemos ensayado los drivers para caracterizarlos y comprender en mayor medida su funcionamiento. En la Figura 5.4 se muestra una foto junto a una descripción de las entradas y salidas.

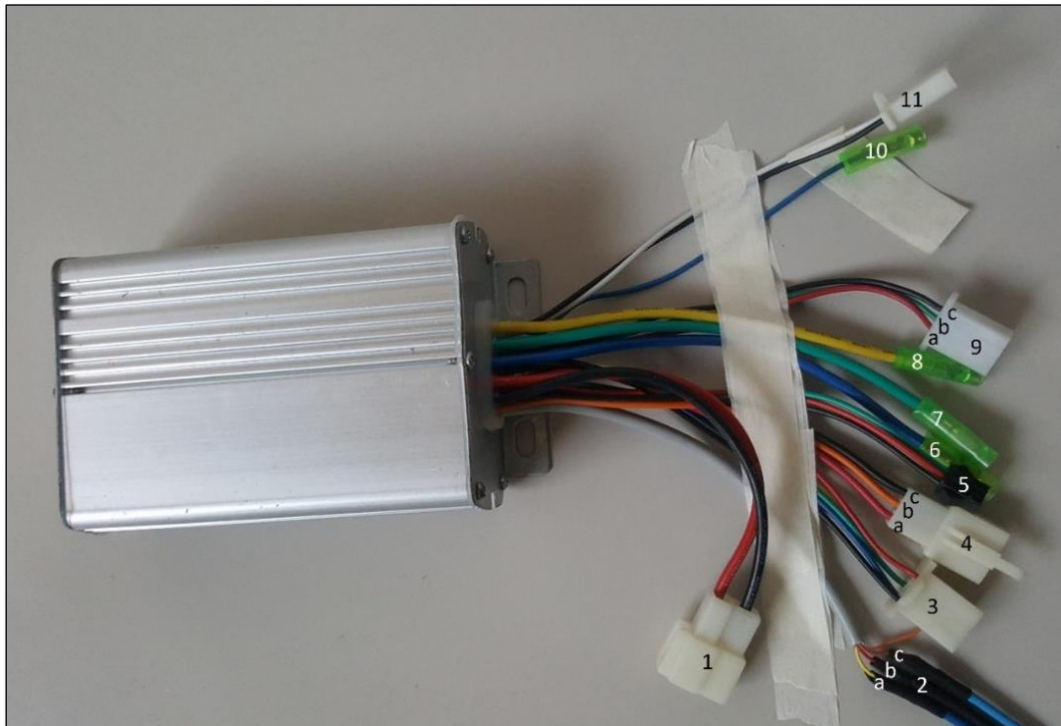


Figura 5.4: Driver motor BLDC.

1. *Alimentación (36V)*: Conectamos directamente a la batería para energizar el driver.

2. *Señales de habilitación de freno regenerativo e inversión de marcha*:

Pin a. *Freno Regenerativo*: Corresponde a la habilitación del freno regenerativo permitiendo su posterior actuación.

- Habilitado: Nivel bajo (GND)
- Deshabilitado: Nivel alto (Vcc)

Decidimos realizar la conexión permanente a GND de manera de tener habilitada la función de frenado regenerativo todo el tiempo.

Pin b. *Inversión de marcha*: Análogamente al pin anterior, por defecto se encuentra en estado alto (Vcc) y para invertir la marcha se debe colocar el pin 'b' en estado bajo (GND). En nuestro caso, se manejará el nivel de este pin a través de una de las salidas del microprocesador colocándolo en alto o en bajo según se requiera.

Pin c. *GND*.

3. *Sensores de efecto HALL*: Esta ficha contiene 5 conectores de los cuales, 3 representan las señales correspondientes a los 3 sensores de efecto HALL que posee el motor (espaciados 120° físicos) y los otros dos conectores corresponden a Vcc y GND.

Una de las 3 señales de los sensores HALL será utilizada para medir la velocidad de giro de los motores. En una sección posterior se explicará con mayor detalle cómo es el cálculo de la velocidad.

4. *Interruptor de encendido*: Esta ficha corresponde a la llave para energizar el Driver. Puenteando de manera permanente los pines 'a' y 'b' logramos energizar los drivers en el momento que se energiza el robot ya que no consideramos necesario un interruptor de encendido para cada driver. El pin 'c' no se utiliza.

5. *Sensor de pedaleo* (No utilizado).

6. Alimentación Fase A (PWM).

7. Alimentación Fase B (PWM).

8. Alimentación Fase C (PWM).

9. Tensión del Manillar (variable manipulada): Al variar la tensión del pin 'b' entre 1.23v y 3.51v, varia la tensión aplicada al motor lo que se traduce en disminución o aumento de su velocidad.

10. Señal PWM cuyo *duty cycle* es proporcional a la velocidad de giro.

11. Actuator de freno regenerativo: Con este pin se acciona el freno según se desee, su estado por defecto es alto (Vcc), es decir, si se encuentra en estado alto el freno no está actuado. Por el contrario, se frenará cuando el pin en cuestión esté en estado bajo (GND).

Hasta aquí, hemos explicado los componentes del sistema (*motor y driver*). A continuación, explicaremos brevemente los sensores de efecto HALL, quienes nos permitirán medir la velocidad del robot, para poder realizar posteriormente el control de la misma.

5.2.3. Filtro de Señal HALL

Como se mencionó anteriormente, se deben filtrar las señales de los sensores de efecto HALL para evitar una posible mala lectura de la velocidad, quien es fundamental en el control de velocidad del robot, y además hay que adecuarlas para su procesamiento dentro del microcontrolador, ya que un pico podría dañar al mismo.

El circuito más simple que proponemos para cumplir con dicha labor es el que se muestra en la Figura 5.5. Como se puede observar, consta de un filtro pasabajos conformado por R1 y C1 seguido de un comparador a lazo abierto.

Eligiendo de manera conveniente R1 y C1, podemos ajustar la frecuencia de corte del filtro a un valor conveniente de tal forma que deje pasar las componentes de frecuencia de las señales de HALL (onda cuadrada) y elimine las componentes de alta frecuencia debidas a ruido. Luego, la señal filtrada se compara con 2.5V (R2=R3) y a la salida del operacional se obtiene una onda cuadrada entre 0V y 5V lista para ser procesada dentro del microcontrolador.

La frecuencia de corte f_c de este filtro pasabajos RC es:

$$f_c = \frac{1}{2\pi R_1 C_1} = \frac{1}{2\pi \cdot 2.7 \text{ nF} \cdot 1 \text{ K}\Omega} \cong 59 \text{ KHz} \quad (5.1)$$

Cabe aclarar que necesitaremos 4 filtros de estos, uno por cada motor. Si bien podría presumirse que las 4 ruedas poseen la misma velocidad, se propone medir la velocidad de las 4 ruedas por separado en caso de algún eventual imprevisto, como por ejemplo podría ser el deslizamiento de alguna rueda.

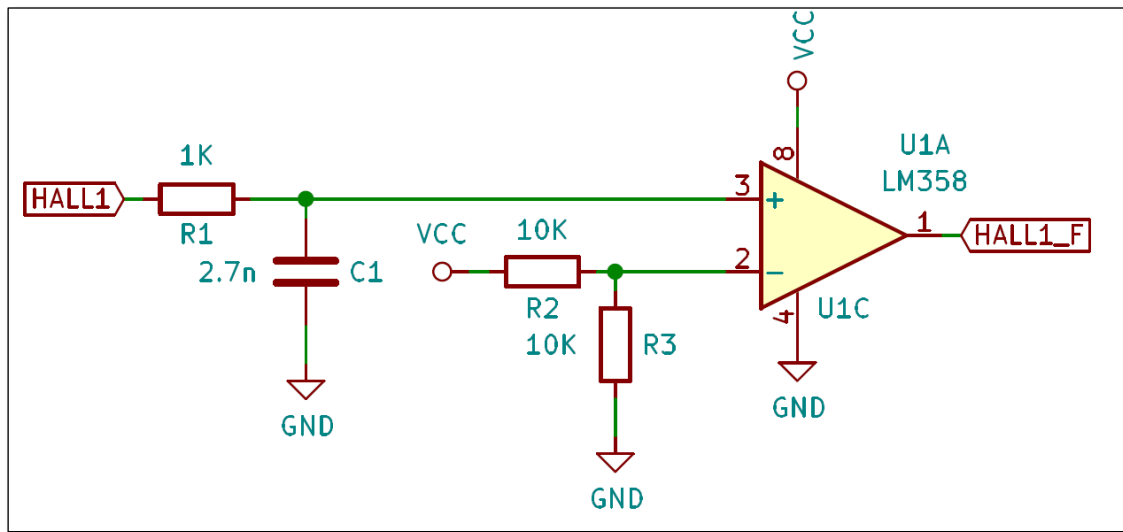


Figura 5.5: Filtro de la señal de un sensor de efecto HALL.

Una aclaración importante, es que la tensión de salida (V_o) de esta etapa no llega a los 5V, esto se debe principalmente al operacional utilizado que no es de tipo *rail to rail*¹ por lo que su excursión a la salida no es sobre todo el rango de operación. Dicho operacional (LM358) no es de este tipo, pero sin embargo posee una amplia excursión de salida y, además, puede lograr una saturación negativa muy próxima a los 0V, hecho de vital importancia.

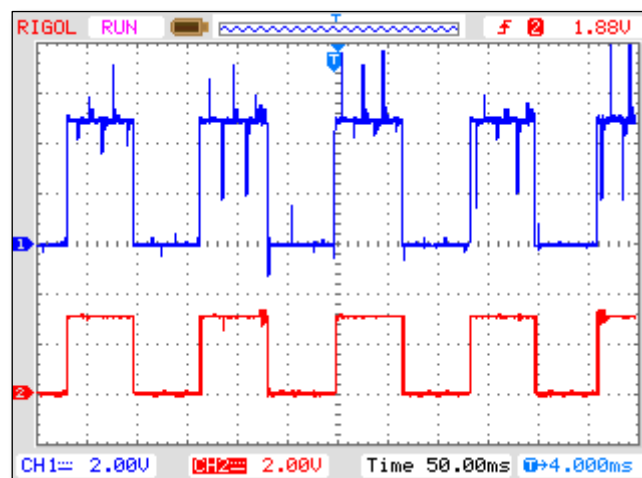


Figura 5.6: Gráfica temporal de la misma señal HALL antes y después del filtro (motor en vacío).

La Figura 5.6 nos muestra la misma señal de un sensor de efecto HALL sin filtrar y filtrada, en azul y rojo, respectivamente. Podemos concluir que el diseño y la implementación de este circuito eléctrico es satisfactoria, ya que se eliminaron en gran medida los problemas de ruido eléctrico.

Además, es importante destacar que esta frecuencia de corte es mucho mayor a la máxima frecuencia que puede tener la señal HALL. Para justificar este hecho pensemos en la aplicación del robot, supongamos que este recorre un campo cultivado a una velocidad de 5 m/s pero que alguna vez para trasladarse alcanza una velocidad mayor, y exagerada, de 15 m/s, esto nos daría una señal HALL con un período de, aproximadamente, 5 ms (si bien no hablamos hasta ahora de los cálculos para la estima de

¹ Los operacionales “Rail to Rail” logran que su salida excursione en todo el rango de la tensión de alimentación.

velocidad y la relación entre esta y la señal HALL, se pueden encontrar más adelante en la sección 5.2.7). Con un período de señal HALL de 5 ms, o bien, con una frecuencia de 200 Hz, vemos que la frecuencia de corte del filtro es mucho mayor a esta (alrededor de 300 veces más grande), lo que no traería problemas en eliminar frecuencias fundamentales significativas de la señal.

El valor de velocidad adoptado, de 15 m/s (aproximadamente 520 rpm), en el ejemplo anterior supera ampliamente al valor de velocidad máximo al que podrá avanzar el robot, ya que por software está limitado a 150 rpm (aproximadamente 4.32 m/s), por lo que la frecuencia máxima de la señal HALL será mucho menor que 200 Hz.

5.2.4. Sensores de efecto HALL

Este tipo de sensor se basa en la detección de campo electromagnético. Los sensores se encuentran en el estator del motor y detectan cuando un imán, que se encuentra en el rotor, pasa frente a él. Cuando esto sucede, el mismo tiene una diferencia de potencial con respecto a masa (cuando el imán tiene una polarización norte), o no (polarización sur).

Así, para este motor que tiene 23 pares de polos, cada sensor de efecto HALL detectará 23 veces el pasaje de un imán con polarización norte y 23 veces un imán con polarización sur, por lo que la señal de estos sensores conmutará entre un valor de tensión (imán norte) y tensión nula (imán sur) 46 veces en una vuelta del motor.

A continuación, en la Figura 5.7, mostramos algunas conmutaciones de los mismos. El ensayo se realizó con el motor en vacío, a la velocidad no se le dio importancia ya que solo se quiere mostrar la conmutación de los sensores.

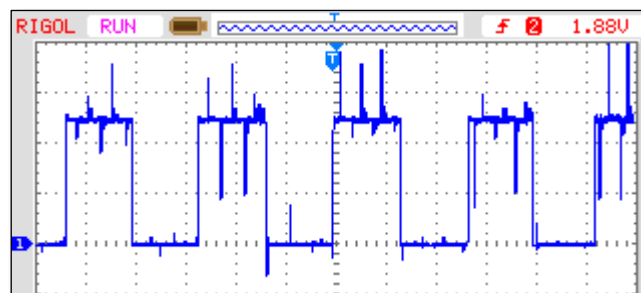


Figura 5.7: Señales de un sensor de efecto HALL con el motor en movimiento.

Con el conocimiento de la cantidad de polos que contiene el motor y el tiempo de conmutación de los sensores de efecto HALL, podremos saber la velocidad angular del motor, y luego, la velocidad traslacional del robot. Estos cálculos, se verán más adelante.

En la Figura 5.7 se puede observar la señal de un sensor de efecto HALL. Una característica que se nota a simple vista, es la presencia de ruido en esta. Este ruido eléctrico, podría ser perjudicial para la lectura de un cambio en esta señal lo que influiría en un mal cálculo de la velocidad. Debido a esto, se implementarán filtros, que serán los encargados de eliminar la presencia del ruido eléctrico mencionado.

5.2.5. Sensor de corriente

Anteriormente vimos cómo, a través de los sensores de efecto HALL, podremos calcular la velocidad del robot, la cual necesitamos para implementar el control.

Otro dato de interés, es la corriente que circula por el motor. Una de las ventajas podría ser, a futuro, la implementación de un lazo de control de corriente, otra el conocimiento de la misma para tomar

medidas de seguridad. Debido a esto, implementamos unos sensores de corriente (Módulo ACS712), los cuales trabajan en un rango de corriente entre -20 A y 20 A. Si bien, a potencia máxima de los motores (750 W) y alimentados con 36 V se tendría una corriente de 20,833 A, (5.2), nunca se trabajará a potencia máxima por lo que no tendremos inconvenientes al utilizar este tipo de sensor.

Cabe aclarar que existe otro sensor de la misma familia que tiene un rango entre -30 A y 30 A, y fue pensado como una posibilidad. Pero, ya que el motor no trabajará cercano a la corriente máxima, era innecesario disponer de mayor excursión, y a su vez, con este sensor se perdía sensibilidad en la lectura de corriente.

$$P_{MAX} = V_{MAX} I_{MAX} \gg I_{MAX} = \frac{P_{MAX}}{V_{MAX}}$$

$$I_{MAX} = \frac{750 \text{ W}}{36 \text{ V}} = 20,833 \text{ A} \quad (5.2)$$

Estos datos se pueden verificar en la hoja de datos de los módulos [9]. Una vez que tomamos la decisión sobre que sensores se utilizarán, estos fueron adquiridos y luego procedimos con el montaje de los mismos (Figura 5.8).

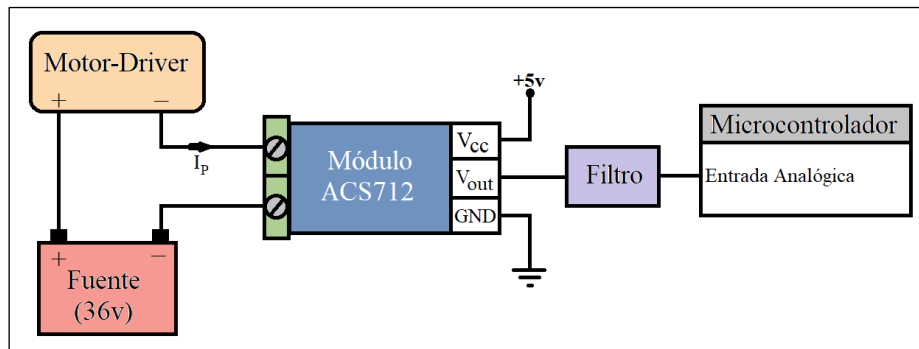


Figura 5.8: Esquema del montaje del sensor de corriente.

Al montarlos sobre el circuito de alimentación de un conjunto motor-driver y graficando la salida en un osciloscopio, nos encontramos con que la señal estaba inmersa en ruido (en rojo en la Figura 5.10) y que el nivel de tensión de continua es mayor a lo especificado en la hoja de datos la placa de desarrollo *Freescale FRDM-K64F*. El módulo ACS712 tiene una excursión de salida V_{out} de 0.5V ~ 4.5V; mientras que los valores típicos de tensión de referencia asumidos por el ADC son, $V_{DDA} = V_{REFH} = 3V$ y $V_{SSA} = V_{REFL} = GND$. Por esto, se diseñó un filtro pasabajos RC de primer orden y ganancia 0,6.

En la Figura 5.9, se muestra el circuito interno del módulo sensor de corriente (rojo) con el agregado del filtro pasabajos a la salida (azul). Podemos ver claramente que el pin de salida V_{IOUT} del módulo ACS712 está afectado por el divisor resistivo cuyos valores son 12K Ω y 18K Ω obteniéndose la siguiente tensión de entrada al conversor.

$$V_{I_{M1}} = \frac{18K\Omega}{12K\Omega + 18K\Omega} \times V_{IOUT} = 0,6 \times V_{IOUT} \quad (5.3)$$

Siendo 4,5V el máximo valor de tensión de salida del módulo ACS712, V_{LM1} nunca superará los 3V (máximo 2,7V) manteniéndose dentro de los valores posibles de V_{DDA} requeridos por el ADC de la placa de desarrollo. La excursión a la salida del sensor de corriente es 0,5V~ 4,5V y a la salida del filtro será de 0,3V~ 2,7V.

De esta manera logramos adecuar la salida del módulo ACS712 a la entrada del conversor.

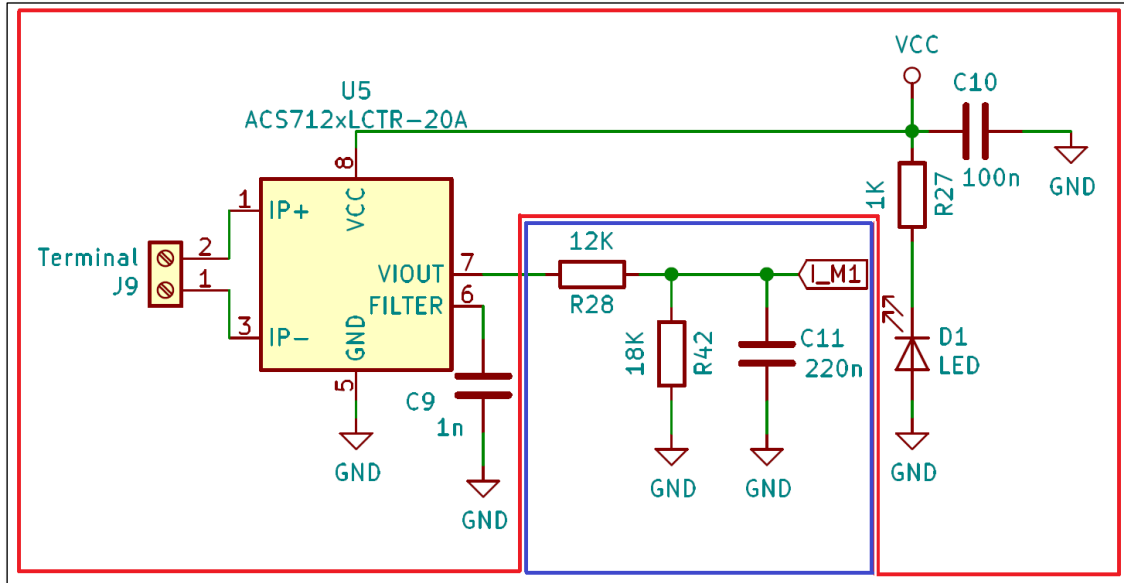


Figura 5.9: Circuito completo del sensor de corriente.

La frecuencia de corte f_c de este filtro pasabajos RC es:

$$f_c = \frac{1}{2\pi R_{eq} C} = \frac{1}{2\pi \cdot 220 \text{ nF} \cdot \frac{18K\Omega \cdot 12K\Omega}{18K\Omega + 12K\Omega}} \cong 100 \text{ Hz} \quad (5.4)$$

El resultado se muestra en azul en la Figura 5.10.

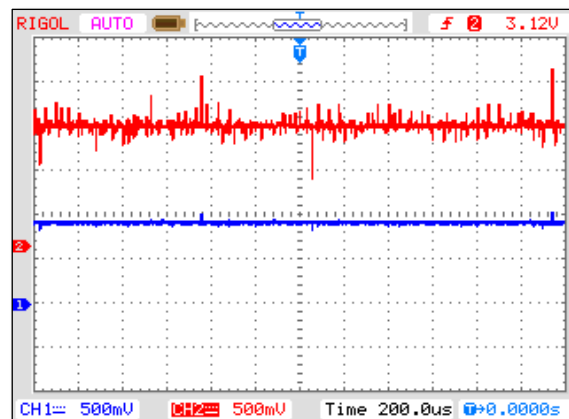


Figura 5.10: Tensión de salida del sensor (motor en vacío).

La intervención del filtro pasabajos afecta a la sensibilidad de la siguiente manera,

$$\frac{4,5V - 0,5V}{2,7V - 0,3V} = \frac{Sensibilidad_{exc-4V}}{Sensibilidad_{exc-2,4V}} \quad (5.5)$$

$$Sensibilidad_{exc-2,4V} = \frac{2,7V - 0,3V}{4,5V - 0,5V} \times 100mV/A \quad (5.6)$$

$$Sensibilidad_{exc-2,4V} = 60mV/A \quad (5.7)$$

En la sección 5.4.3.5 se detalla el procesamiento de la información obtenida desde la tensión de salida del módulo (V_{out}) para lograr la correcta interpretación de la corriente. Pero antes, se debía comprobar que el módulo funcione como se esperaba.

Así, una vez que tenemos la señal de corriente adaptada y lista para ser procesada por el microcontrolador, debemos verificar el funcionamiento del sensor en su zona de trabajo. Se procedió a medir la corriente para diferentes situaciones, y se comprobó que efectivamente el módulo se comporta según lo indicado en la hoja de datos del mismo, respondiendo perfectamente según la característica mostrada en la Figura 5.11 con una sensibilidad de $100mV/A$.

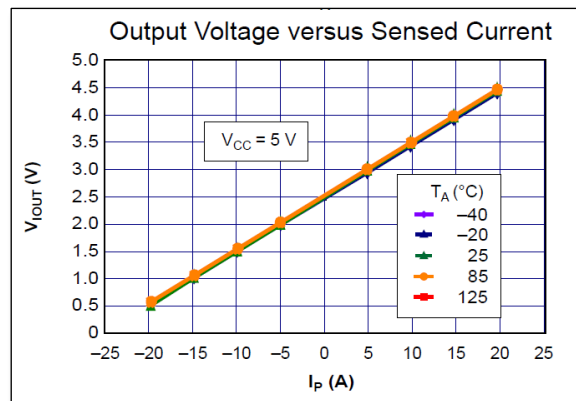


Figura 5.11: Tensión de salida vs Corriente sensada.

Además, la relación lineal que existe entre tensión y corriente (V-I), simplificará el posterior diseño de control.

5.2.6. Filtros PWM

El torque del motor es proporcional a la corriente, y como esta depende de la tensión aplicada, el control de velocidad se va a hacer controlando la tensión.

Para regular esta tensión a través de las señales del microprocesador, se propone implementar una modulación PWM, la cual nos permite variar el valor medio de una señal cuadrada ajustando su ciclo de trabajo (*Duty Cycle*). Luego, mediante un correcto filtrado de esta señal y una etapa de potencia se obtiene el valor de tensión pretendido que debe aplicarse al driver.

En la Figura 5.12, se ilustra el circuito implementado. Este circuito consta de un filtro pasabajos de primer orden compuesto por R1 y C1 seguido por un “buffer”. Esta última etapa es la encargada de evitar que se sobrecargue la salida del microcontrolador, proveyendo al circuito de una alta impedancia y permite entregar los niveles de potencia necesarios.

Análogamente al circuito del filtro hall, se puede ajustar los valores de R1 y C1 de manera conveniente para lograr el filtrado pretendido.

La frecuencia de corte f_c de este filtro pasabajos RC es:

$$f_c = \frac{1}{2\pi R_1 C_1} = \frac{1}{2\pi \cdot 3.3 \mu F \cdot 15 K\Omega} \cong 3.215 \text{ Hz} \quad (5.8)$$

Esta frecuencia de corte del filtro es muy pequeña y nos permite quedarnos con la componente de continua a la salida, que es nuestro objetivo. A la vez, se corrobora que la frecuencia de la señal PWM (1 KHz, Figura 5.13 y Figura 5.14) es mayor que la frecuencia de corte del filtro.

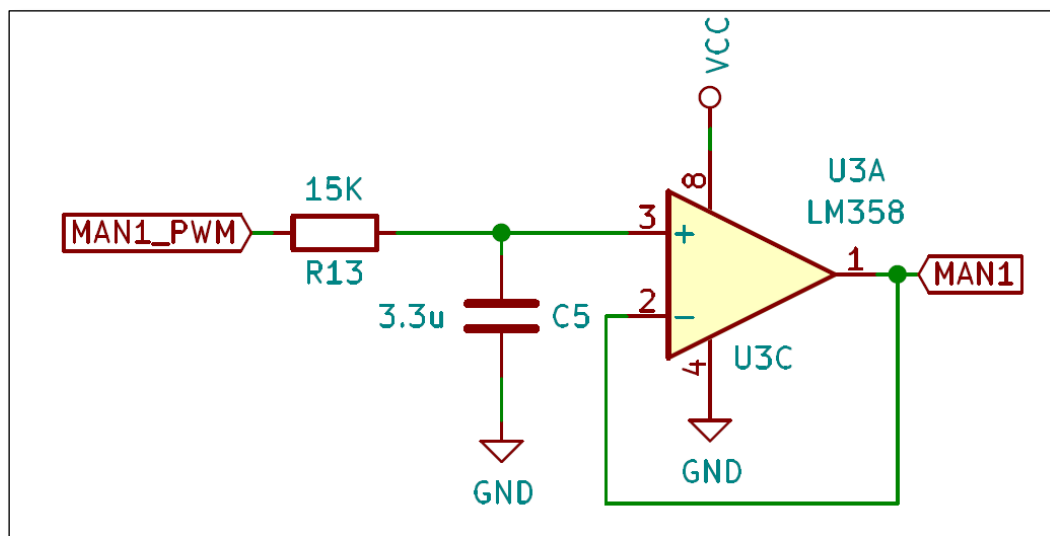


Figura 5.12: Filtro para las señales PWM.

Cabe aclarar que, como es de esperar, necesitaremos 4 circuitos como el de la Figura 5.12 (uno por cada motor).

La salida de este circuito (MAN1), es la que se conecta al driver del motor. Un problema que se podría haber presentado, era que el driver demande corriente de la etapa anterior (circuito de Figura 5.12) y, como se puede ver, esa corriente debería ser proveída por el AO (LM358). Por esto, se midió a través de una pinza amperométrica si el driver demandaba cierta corriente y notamos que no lo hacía, lo que nos dice que la primera etapa del driver es un buffer. Así, no hay inconvenientes de utilizar este circuito propuesto.

En las Figura 5.13 y Figura 5.14, se muestra el comportamiento del circuito para dos situaciones en particular, ambas con condición del motor en vacío. En la primera situación se brinda una señal de PWM con el 70% del ciclo en alto, mientras que en la segunda la señal tiene el 90% del ciclo en alto. En rojo se

muestra la señal PWM proveniente del microcontrolador, que conmuta entre 0V y 3.3V, y en azul la salida del filtro, o sea, la componente de continua de la señal PWM. Como es de esperar, en la segunda situación se obtiene una componente de continua mayor a la primera, ya que el tiempo de la señal en alto (3.3V) es mayor al de la primera situación.

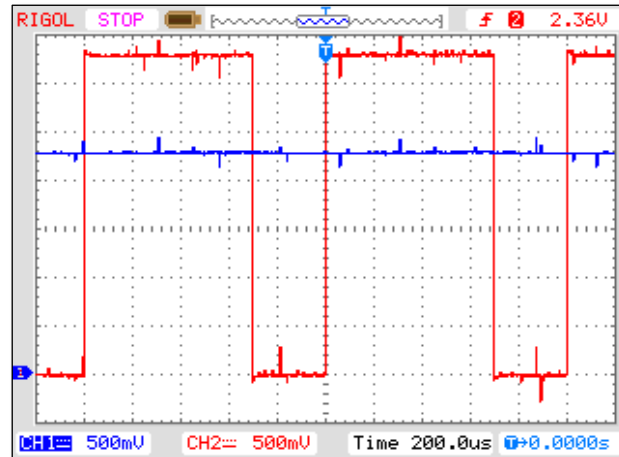


Figura 5.13: Entrada (rojo) y salida (azul).

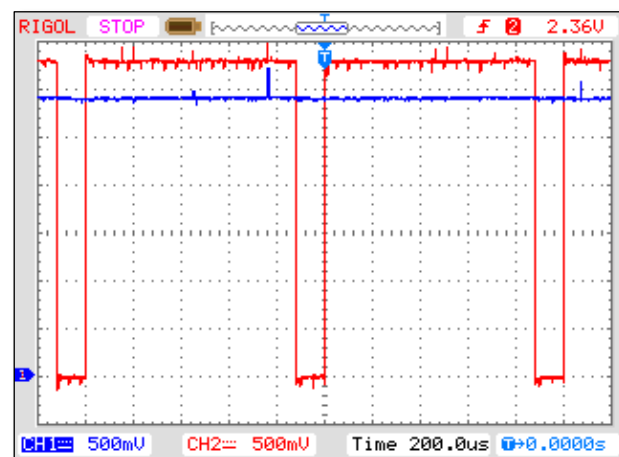


Figura 5.14: Entrada (rojo) y salida (azul).

5.2.7. Estima de la velocidad tangencial y distancia recorrida

La información de la señal filtrada de cualquiera de los tres sensores hall, ingresa al microcontrolador con el objetivo de ser usada para calcular la velocidad de giro instantánea del motor.

Teniendo en cuenta el número de pares de polos (p) del motor, por cada giro completo del mismo, el sensor de efecto hall emitirá p pulsos; como mencionamos en la sección 5.2.1 los motores cuentan con 23 pares de polos. Por lo tanto, el ángulo recorrido por el motor entre 2 pulsos consecutivos del sensor de efecto HALL es:

$$\theta_r = \frac{2\pi}{p} \text{ rad} = \frac{2\pi}{23} \text{ rad} = 15.652^\circ \quad (5.9)$$

Si se conoce el tiempo T_{Hall} entre dos pulsos consecutivos del sensor, entonces se puede calcular la velocidad media de giro del motor entre dos pulsos según:

$$\omega_r = \frac{\theta_r}{T_{Hall}} = \frac{2\pi}{\rho T_{Hall}} = \frac{2\pi}{23 T_{Hall}} \text{ rad/seg} \quad (5.10)$$

Luego, sabiendo que $\omega_{motor} = 2\pi f_{motor}$,

$$f_{motor} = \frac{1}{\rho T_{Hall}} \text{ 1/seg} \quad (5.11)$$

Finalmente, la velocidad en RPM puede calcularse de la siguiente manera:

$$RPM_{motor} = 60f_{motor} = \frac{60}{\rho T_{Hall}} \quad (5.12)$$

Una vez conocida la ω_r , se puede calcular la velocidad tangencial y la distancia recorrida por la rueda si se conoce el radio de la misma.

Midiendo el radio de la rueda montada al motor, se calcula el perímetro de la misma a través de (5.13), y luego conociendo la cantidad de vueltas por (5.14) podremos deducir la distancia recorrida por cada vuelta.

$$Perimetro = 2\pi radio = 2\pi 0,2752 = 1,729 \quad (5.13)$$

Como ya sabemos, la señal hall emite ρ ciclos por cada giro completo, almacenado la cantidad de ciclos emitidos en todo un recorrido y dividiéndolos entre ρ tendremos la cantidad de vueltas realizadas.

$$Cant_{vueltas} = \frac{Cant_{ciclos}}{\rho} \quad (5.14)$$

Por lo tanto, la distancia recorrida es:

$$Dist_{Recorrida} = Cant_{vueltas} \cdot Perimetro \quad (5.15)$$

Finalmente podemos calcular la velocidad del motor en m/s o km/h con:

$$Velocidad_{m/s} = \frac{Perimetro \cdot RPM_{motor}}{60} \quad (5.16)$$

$$Velocidad_{km/h} = 3,6 \cdot Velocidad_{m/s} \quad (5.17)$$

Como el perímetro es 1.729m, tenemos la siguiente relación:

$$Velocidad_{km/h} = 0,10374 \cdot RPM_{motor} \quad (5.18)$$

5.2.8. Controlador

Con el principal objetivo de poder controlar la velocidad de giro de los motores variando la tensión de entrada del driver, debemos conocer la respuesta de velocidad ante una excitación para obtener el modelo del sistema y proceder con el cálculo del controlador correspondiente.

A priori se sabe la respuesta de velocidad del motor ante una excitación se podrá aproximar a un primer orden con retardo, del cual se pueden extraer fácilmente los parámetros y construir el modelo, como se muestra en la Figura 5.15.

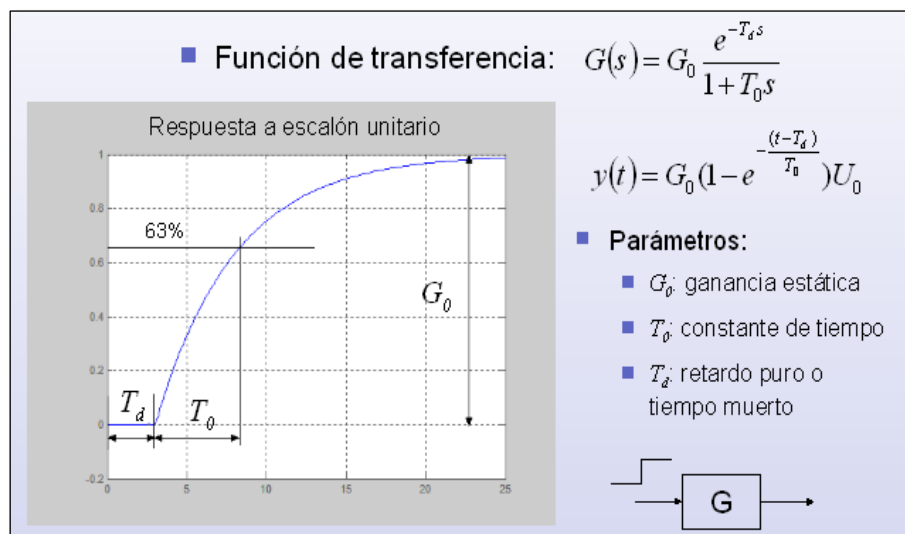


Figura 5.15: Respuesta al escalón de un sistema de primer orden con retardo.

Una vez que tengamos el sistema identificado utilizamos el método de Ziegler - Nicholls para determinar cada uno de los parámetros del controlador PI en el dominio continuo, trazando la tangente de la respuesta del sistema en el punto de inflexión, y usando la tabla correspondiente al método mencionado, tal como se muestra en la Figura 5.16.

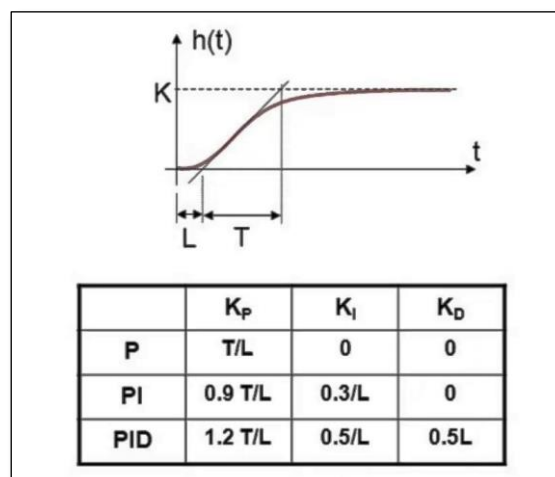


Figura 5.16: Tabla de constantes del controlador del método Ziegler - Nicholls.

Este controlador se caracteriza por eliminar el error en estado estacionario (offset) de la variable controlada que en este caso es la velocidad de giro del motor.

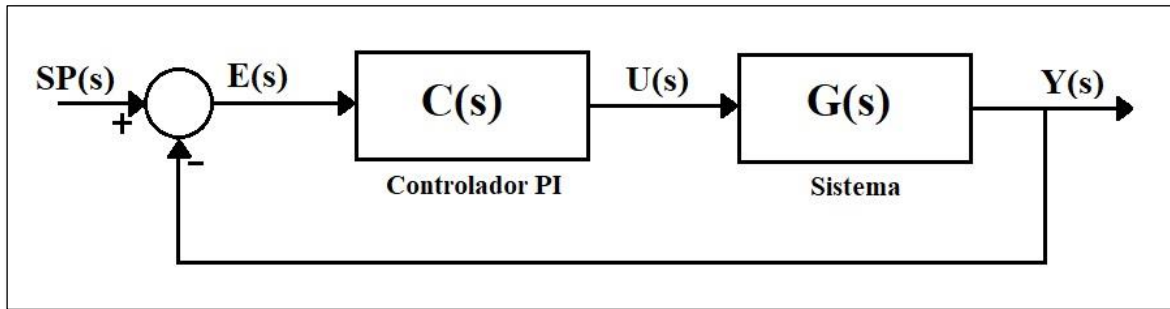


Figura 5.17: Lazo de control implementado.

La ecuación que describe al controlador PI en el dominio continuo, que relaciona la tensión de entrada al driver $U(s)$ con la velocidad $Y(s)$, queda expresada de la siguiente manera:

$$PI(s) = C(s) = K_p + \frac{K_I}{s} = K_p + \frac{K_p}{T_i s} \quad (5.19)$$

Donde:

K_p : Constante proporcional

K_I : Constante integral

T_i : Tiempo integrativo

Dado que trabajamos con un procesador digital para realizar el control a lazo cerrado es necesario encontrar el equivalente discreto de la función transferencia del controlador a través de, por ejemplo, el *método de Euler*. Este método relaciona la variable continua s con la variable discreta z de la siguiente manera:

$$z = e^{sT_s} \approx 1 + sT_s \quad (5.20)$$

Sustituyendo (5.20) en la ecuación (5.19) se tiene la función transferencia discreta del controlador:

$$PI(z) = C_d(z) = K_p + \frac{K_p}{T_i} \frac{T_s}{(z - 1)} \quad (5.21)$$

Donde:

$$K1 = K_p$$

$$K2 = K_p \left(\frac{T_s}{T_i} - 1 \right)$$

T_s : Periodo de muestreo

Una vez que definimos $K1$ y $K2$ en función de las constantes K_p , T_s y T_i , y junto con la función transferencia del controlador $PI(z)$, escribimos (5.22). Esta ecuación representa la función de salida del controlador $U(z)$ para una función de entrada $E(z)$ al mismo.

$$U(z) = PI(z) E(z) = \left(K_1 + \frac{K_1 + K_2}{(z - 1)} \right) E(z) \quad (5.22)$$

Luego, realizando la *anti-transformada de Z*, se tiene la ecuación en diferencias que permite calcular la variable a manipular. Esta variable, será utilizada en el correspondiente bloque *PI* del microcontrolador y su cálculo tendrá la siguiente forma:

$$u(k + 1) = u(k) + K_1 e(k + 1) + K_2 e(k) \quad (5.23)$$

Con el objetivo de comparar las respuestas de velocidad de los dos nuevos motores y de los dos reutilizados, se realizaron distintos ensayos. Con los motores en vacío, se les aplicó a ambos la misma tensión y se midió su velocidad.

En primer lugar, observamos en la Figura 5.18 la velocidad alcanzada en RPM de los motores traseros y delanteros ante variaciones de tensión aplicada.

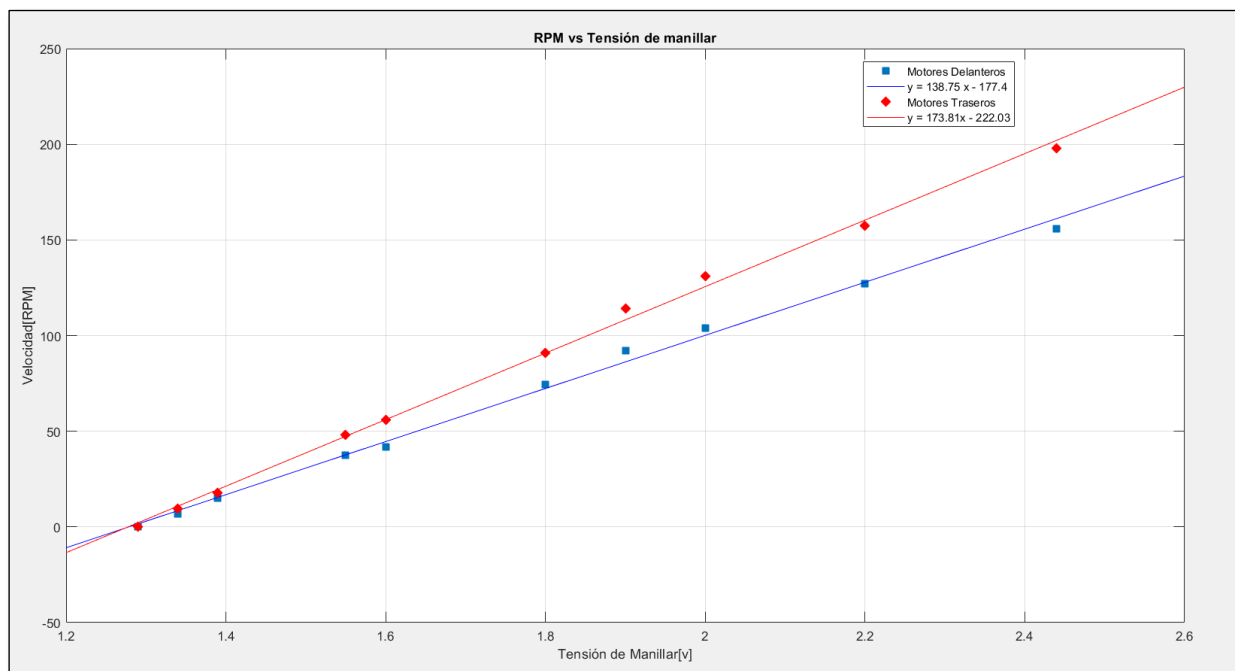


Figura 5.18: Velocidad [RPM] vs Tensión de manillar [V].

Pudo verse, en los ensayos en vacío de los cuatro motores, que estos poseen característica prácticamente lineal dentro del rango de operación normal, y, como era de esperarse, dado que los motores traseros (rojo) tienen mayor potencia que los motores delanteros (azul), ante igualdad de tensión aplicada a la entrada se obtiene mayor velocidad a la salida. Otro comportamiento a destacar es que el arranque de los cuatro motores se produce al aplicar una tensión de 1.29V.

En segundo lugar, en la Figura 5.19, se muestra la respuesta de velocidad de los cuatro motores ante dos casos particulares de tensión de manillar, a la izquierda (arriba y abajo) se presenta una situación, mientras que a la derecha (arriba y abajo) la restante.

La diferencia se encuentra únicamente en la tensión aplicada a los motores, pero en ambas situaciones estos se encuentran en vacío y a la hora de frenar, no se aplica el frenado regenerativo sino que se los deja frenar por inercia.

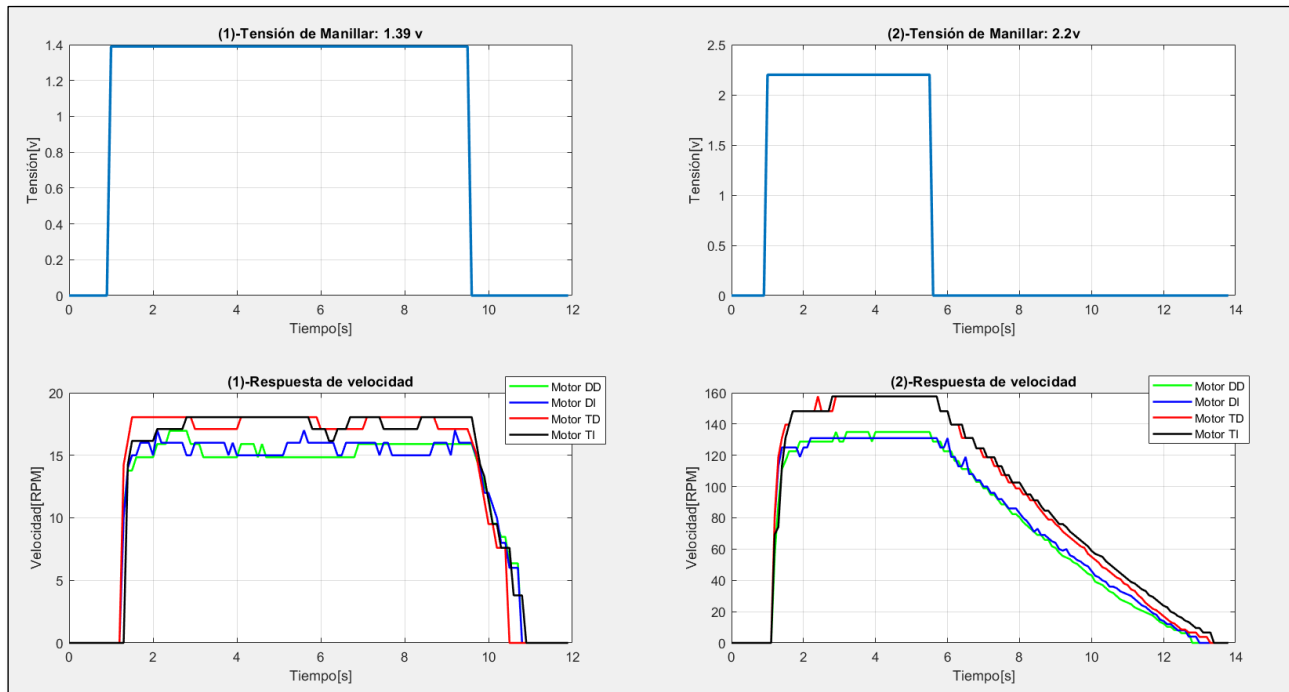


Figura 5.19: Respuesta del sistema frente a distintos escalones en la tensión de manillar.

En base a la observación de las respuestas, si bien la velocidad final alcanzada por cada uno de ellos varía correspondiéndose con lo visto en la Figura 5.18, tanto el retardo puro o tiempo muerto T_d como la constante de tiempo T_o (Figura 5.15) de los 4 motores, son similares.

Haciendo otra comparación entre las respuestas de ambas situaciones, se destaca el tiempo que demoran los motores en frenar. Este tiempo es mayor en la situación de la derecha, pues la tensión aplicada es mayor y, por ende, los motores giran a mayor velocidad, lo que les demanda más tiempo para frenar. Recordamos que no se aplicó el frenado regenerativo, y cuando nos referimos a frenar quiere decir que se dejó de aplicarle un valor de tensión a los motores.

A partir de esto, proponemos realizar el control de velocidad de los motores con el controlador PI con las siguientes constantes:

$$K_p = 15$$

$$T_i = 0.25$$

$$T_s = 100ms$$

Conociendo en detalle todos los componentes de nuestro sistema de control (sistema, controlador, sensores y filtros), ahora se presenta la interconexión de los mismos.

5.2.9. Conexión de componentes del sistema de control

En el diagrama de la Figura 5.20, mostramos el relevamiento de las señales principales del sistema.

Como unidad central de control tenemos la placa de desarrollo freescale K64F (bloque K64F) que regula el comportamiento global del robot a través de la información obtenida desde:

- *Filtros de Señal Hall:* Formado por una configuración pasabajos seguida de un comparador a lazo abierto, cuyo objetivo es adecuar la señal proveniente de los sensores hall de los motores brushless, para luego, a partir de ellas realizar el cálculo de la velocidad de giro del motor.
- *Sensores de corriente:* Esta señal proviene del filtro pasabajos en la salida VIOUT de los sensores ACS712. Permite conocer la corriente consumida por cada uno de los motores.
- *Parada de emergencia:* Señal asociada al estado del pulsador de parada de emergencia (PE).
- *Encoder:* Corresponde a los 8 bits del encoder absoluto que el procesador debe utilizar para calcular la posición angular del motor paso a paso y así conocer la dirección del robot.
- *Control Remoto:* Información proveniente desde el control remoto. Establece las consignas de dirección y velocidad que deberán alcanzarse en modo REMOTO.

Luego, la unidad de control, debe procesar esta información y poner a disposición:

- *Consigna de velocidad de cada motor:* Por medio de una señal PWM que se adecua mediante un filtro pasabajos seguido por un buffer logrando una referencia de tensión continua que ingresa al driver del motor brushless simulando la tensión de manillar.
- *Freno y Reversa:* Señales de salida alto o bajo que comandan el freno regenerativo y la marcha atrás respectivamente. Estas se adecuan mediante una interfaz de adaptación formada por un transistor NPN en configuración de llave inversora y luego ingresan al driver del motor brushless para establecer el cuadrante de trabajo.
- *Consigna de dirección:* Señales Pulse, Enable y Direction (PULL, ENA, DIR respectivamente) que se adaptan mediante una llave inversora implementada con un transistor NPN e ingresan en el driver que controla al motor paso a paso y establece la dirección del robot.

Observamos la transmisión (Tx) y recepción (Rx) de datos serie entre el bloque K64F (microcontrolador) y el bloque PC que representa la MINI-PC intel® NUC. La comunicación se logra mediante USB.

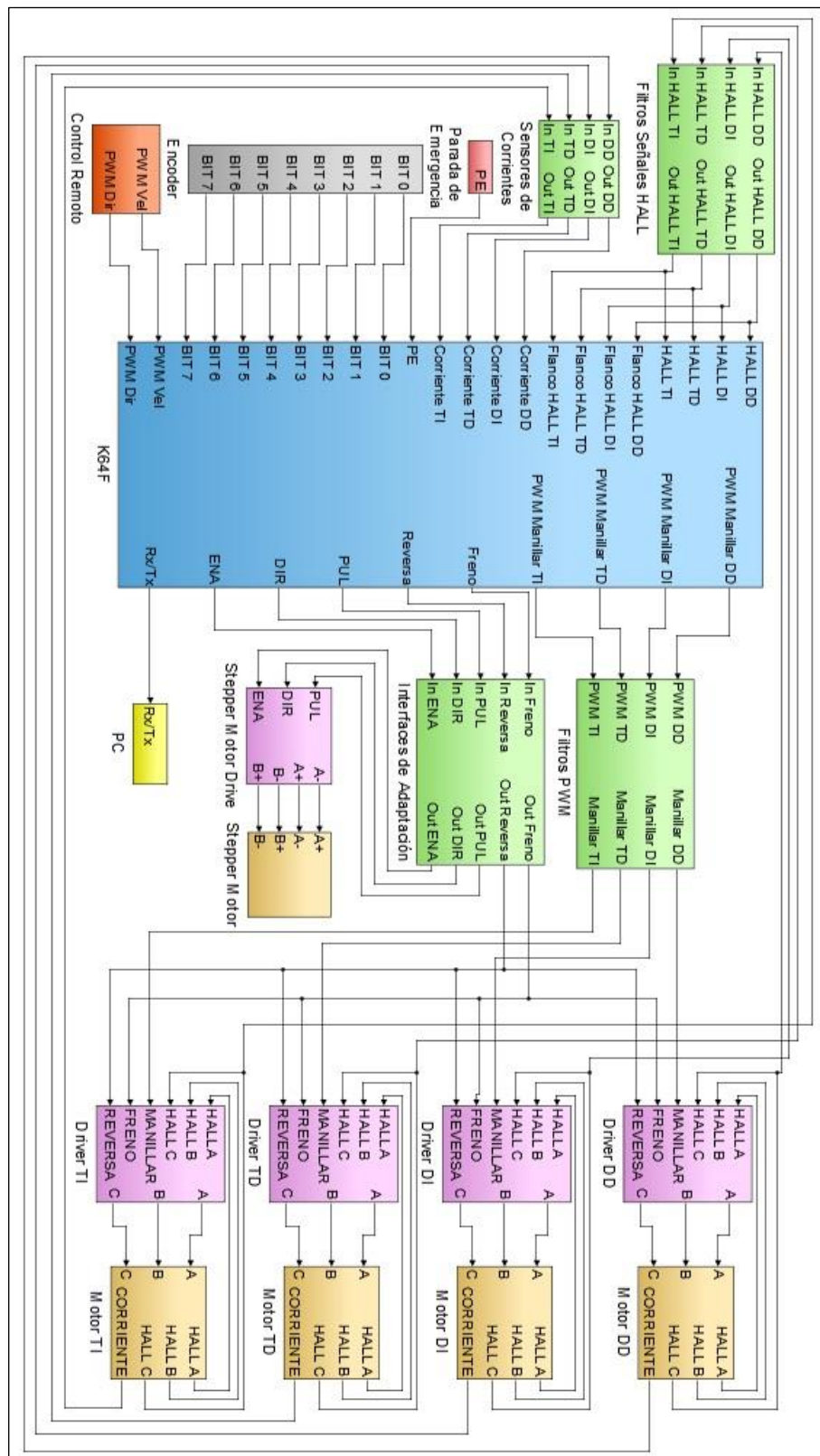


Figura 5.20: Diagrama de conexión de componentes del sistema de control.

5.2.10. Elección del microprocesador

Como se supo hasta ahora, se necesita un microprocesador para poder llevar a cabo, entre otras cosas, el controlador de dirección y el de tracción. La elección del microprocesador K64F, se realizó debido a que esta cumplía con los requerimientos necesarios, pues contaba con las entradas y salidas suficientes. A continuación, mostramos una tabla que explicita las características de lo utilizado.

La Tabla 5.1, muestra las Entradas/Salidas del microprocesador necesarias para solventar, en principio, todos los requerimientos que nos impone la solución completa del sistema de control de tracción y dirección.

	E/S		Cant.	Tipo	Descripción
MOTOR	8 ENTRADAS	(1xMotor)	4	Interrupción	Captura de tiempo
			4	Binaria	Validación
	6 SALIDAS	(1xMotor)	4	PWM	Tensión Manillar
				1	Binaria
			1	Binaria	Fren0
DIRECCION	8 ENTRADAS - (8 bits)		8	Binaria	Encoder
	3 SALIDAS - Motor PaP		1	Binaria	Habilitación - ENA
			1	Binaria	Pulsos - PULL
			1	Binaria	Dirección - DIR
CORRIENTE	4 ENTRADAS - (1xMotor)		4	ADC	Medición de corriente
CONTROL REMOTO	2 ENTRADAS		1	PWM	Ref dirección
			1	PWM	Ref velocidad
PARADA DE EMERGENCIA	1 ENTRADA		1	Binaria	Pulsador
LED	1 SALIDA		1	Binaria	Led de funcionamiento
PC	ENTRADA/SALIDA		1	Serial	Comunicación con la PC
Total	23 ENTRADAS				
	10 SALIDAS				
	1 Entrada/Salida				

Tabla 5.1: Entradas/Salidas requeridas en el microprocesador.

Comparando la Tabla 5.1 con las especificaciones de la placa K64F mencionadas en la sección 4.2, podemos observar que cumple sin mayores inconvenientes todos los requerimientos impuestos.

Por último, se tuvo que tener presente el hecho de que esta placa trabaja con niveles de voltajes distintos a los niveles de los drivers de los motores. El estado bajo se representa con 0 V y el alto con 3.3

V, mientras que los drivers lo hacen con 0 V y 5 V, respectivamente. Debido a esto se realizó una adaptación de estos niveles de tensión.

5.2.11. Interfaces de Adaptación

Como se mencionó, las salidas digitales del microcontrolador son señales que en su estado alto presentan una tensión de 3.3V y en su estado bajo 0V. Dado que los niveles de alto y bajo del driver son 0V y 5V respectivamente, se precisa un circuito capaz de adaptar los niveles de salida del microprocesador.

Esto se puede lograr polarizando un transistor tipo NPN trabajando como llave inversora, es decir, conmutando entre corte y saturación (Figura 5.21), en una configuración emisor común. Así, al tener 0 V en PULin, el transistor estará cortado y la salida PUL será Vcc (5 V). Por otro lado, al ser PULin 3.3 V el transistor conducirá en estado de saturación y PUL, que es la tensión colector-emisor del mismo, será aproximadamente 0.2 V.

A pesar de que existan aproximadamente 0.2 V en PUL (cuando idealmente debería ser 0 V), estos son interpretados por el driver como estado bajo, lo que no trae problemas de mala interpretación. Otro problema, podría presentarse cuando PUL debe estar en estado alto (transistor cortado). Si bien en este circuito es lógico pensar que PUL será 5 V, hay que tener presente que esta salida se conectará al driver por lo que estará cargada por la primera etapa del mismo. Se comprobó, a través de mediciones por pinza amperométrica, que la corriente demandada por el driver era nula, lo que nos da la pauta de que la etapa que sigue es un buffer (seguidor de tensión) que, prácticamente, no consume corriente asegurándonos que PUL será 5 V.

Como se mencionó, este circuito funciona como llave inversora por lo que un estado bajo a la salida del microcontrolador (PULin), se vería como estado alto en la entrada al driver (PUL). Este problema se podría solucionar agregando otra etapa similar a continuación, pero esto demanda más circuitos. Una solución mucho más simple fue cambiar la lógica en el software del microcontrolador. Así, un nivel bajo (0) es cambiado a un nivel alto (1), que, con la llave inversora, representa un 0 a la entrada del driver, y viceversa.

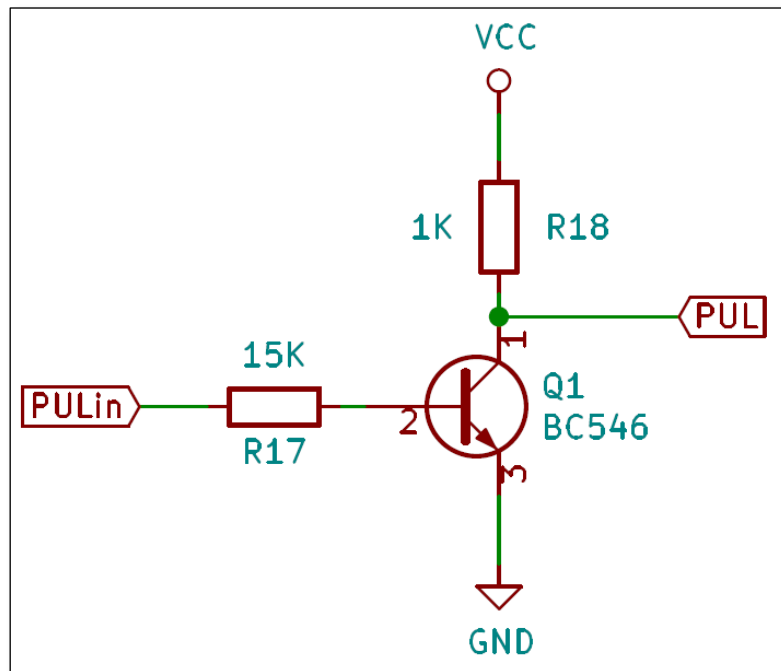


Figura 5.21: Interfaz de adaptación.

Como se ilustra en el diagrama de bloques de la Figura 5.20, necesitaremos 5 de estos circuitos para las señales de “Reversa”, “Freno”, “PUL”, “DIR” y “ENA”.

En la Figura 5.22, se puede apreciar el circuito funcionando correctamente.

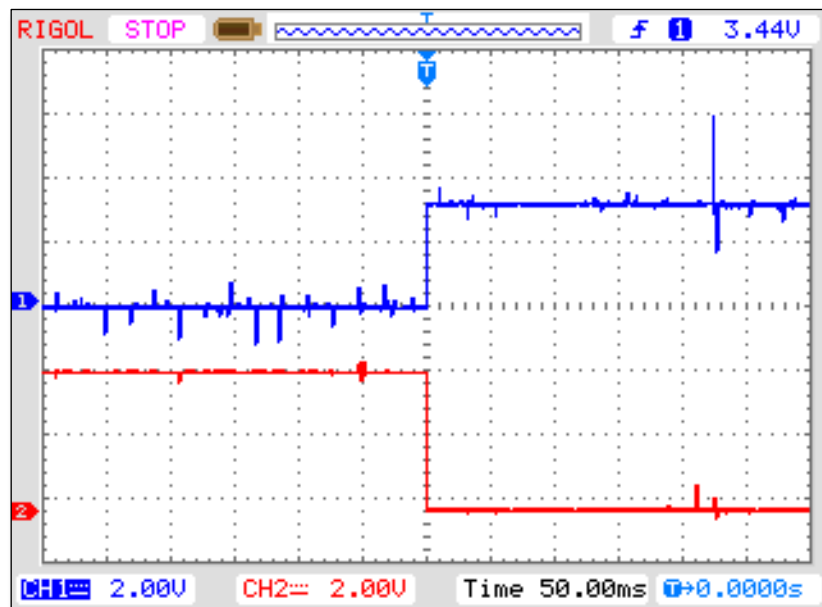


Figura 5.22: Vi (azul) y Vo (rojo).

5.3. Diseño de placa electrónica

Una vez que se obtuvo el satisfactorio comportamiento de estos circuitos electrónicos, se debió realizar el montaje de los mismos sobre una placa electrónica, la cual se encuentra dentro del gabinete (Sección 4.10). Para poder realizar esta tarea, se utilizó una placa universal en conjunto con los componentes electrónicos necesarios.

5.3.1. Diseño de placa electrónica universal

Como se comentó, la placa junto a los componentes nos permitirá realizar todos los circuitos necesarios. Una vez que se supo que circuitos se utilizarían se procedió a adquirir los componentes electrónicos y la placa universal, para luego, a través de soldadura con estaño, crear estos circuitos sobre la placa universal y unificar todos estos dentro de la misma. Además, se dejó espacio físico para poder montar la placa de desarrollo K64F. Se muestra en la Figura 5.23, una foto de la placa real dentro del gabinete, ya funcionando.

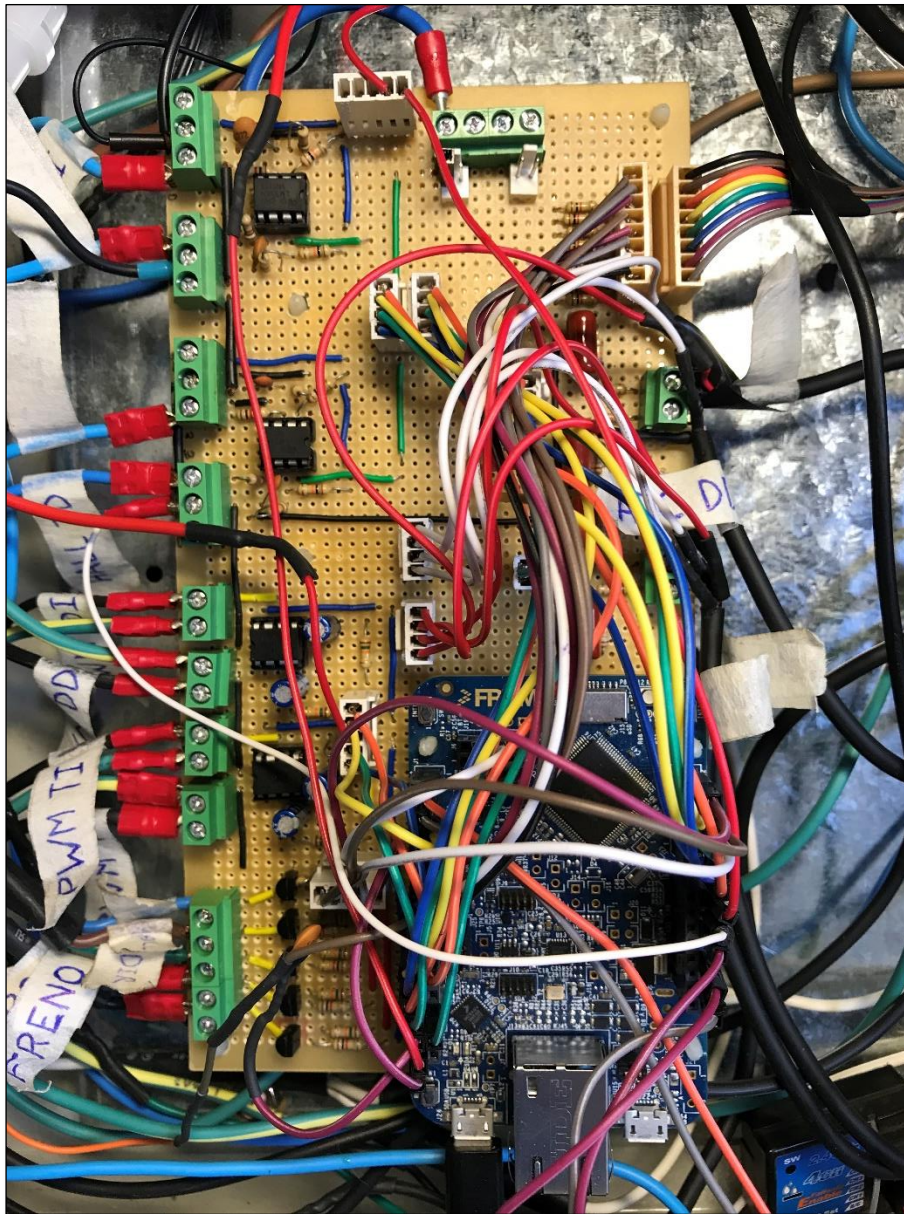


Figura 5.23: Foto de la placa universal ya conectada y funcionando dentro del gabinete.

Como se puede observar, la conexión entre esta placa y la K64F es desprolija, esto se debe a que los pines de la K64F no están ordenados de acuerdo a la configuración de nuestra placa electrónica por ende hay conexiones que se cruzan y no es para nada fácil poder seguir a las mismas. Por este motivo, se realizó una vista superior de la placa electrónica.

En la Figura 5.24, mostramos las entradas y salidas de esta placa electrónica y la ubicación de la K64F. Indicar el lugar a donde se encuentra cada circuito es innecesario y dificulta la visión de lo que realmente es de interés, pero conocer la ubicación física de entradas y salidas es fundamental, ya que, si una persona ajena al proyecto quiere manipular esta placa o debe retirarla del gabinete por alguna necesidad, puede desconectarla y volver a conectarla sin problemas.

Como se puede ver en los rectángulos verdes, que simulan ser borneras en vista superior, hay algunas entradas de estas sin especificar, esto se debe a que no fue necesario la utilización de las mismas y, aunque podrían no estar presentes en el esquema, se dejaron de todas formas para una mejor interpretación debido a que es así como se encuentran en la placa real.

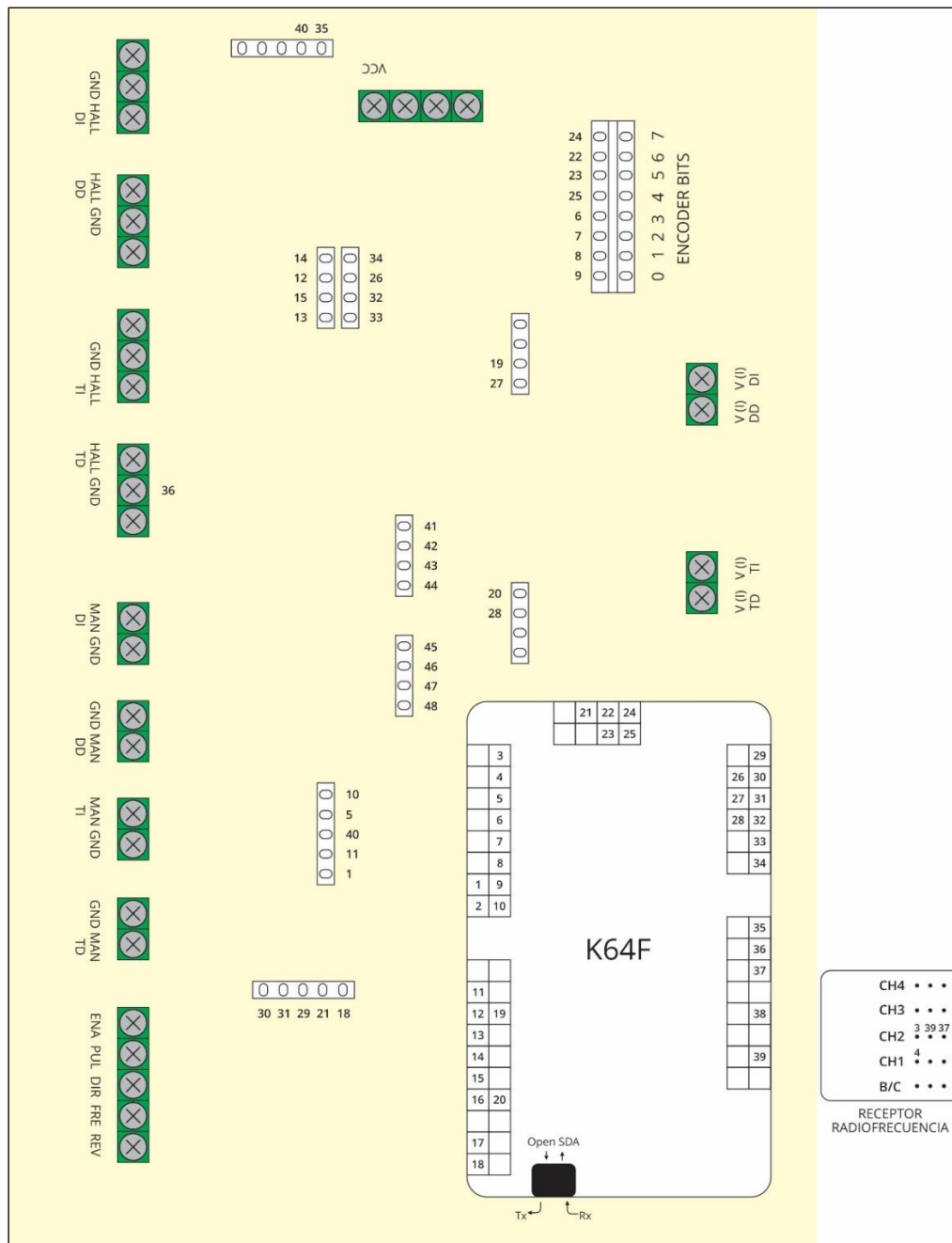


Figura 5.24: Esquema simplificado de placa electrónica universal (entradas y salidas) junto a receptor de radio frecuencia del mando a distancia.

Seguido, en la Tabla 5.2, detallamos las entradas y salidas de la placa electrónica para una mayor comprensión.

Subsistema	Nombre	Tipo	Descripción
ALIMENTACIÓN	VCC	Entrada	Tensión de alimentación de la placa proveniente de un regulador de tensión (5V)
	GND	Entrada	Masa
CORRIENTE	V(I) DI	Entrada	Tensión proveniente del módulo de corriente del motor delantero izquierdo
	V(I) DD	Entrada	Tensión proveniente del módulo de corriente del motor delantero derecho
	V(I) TI	Entrada	Tensión proveniente del módulo de corriente del motor trasero izquierdo
	V(I) TD	Entrada	Tensión proveniente del módulo de corriente del motor trasero derecho
DIRECCIÓN	ENCODER BITS (bit0)	Entrada	Bit 0 del encoder del motor paso a paso
	ENCODER BITS (bit1)	Entrada	Bit 1 del encoder del motor paso a paso
	ENCODER BITS (bit2)	Entrada	Bit 2 del encoder del motor paso a paso
	ENCODER BITS (bit3)	Entrada	Bit 3 del encoder del motor paso a paso
	ENCODER BITS (bit4)	Entrada	Bit 4 del encoder del motor paso a paso
	ENCODER BITS (bit5)	Entrada	Bit 5 del encoder del motor paso a paso
	ENCODER BITS (bit6)	Entrada	Bit 6 del encoder del motor paso a paso
	ENCODER BITS (bit7)	Entrada	Bit 7 del encoder del motor paso a paso
MOTORES	MAN TD	Salida	Tensión de alimentación del motor trasero derecho
	MAN DI	Salida	Tensión de alimentación del motor delantero izquierdo
	MAN DD	Salida	Tensión de alimentación del motor delantero derecho
	MAN TI	Salida	Tensión de alimentación del motor trasero izquierdo
	HALL DD	Entrada	Señal HALL del motor delantero derecho
	HALL TD	Entrada	Señal HALL del motor trasero derecho
	HALL TI	Entrada	Señal HALL del motor trasero izquierdo
	HALL DI	Entrada	Señal HALL del motor delantero izquierdo
HABILITACIÓN	ENA	Salida	Habilita el uso del motor paso a paso
	PUL	Salida	Indica la cantidad de pulsos que debe girar el motor paso a paso
	DIR	Salida	Indica la dirección de giro del motor paso a paso (derecha o izquierda)
	FRE	Salida	Habilitación del freno de los motores BLDC
	REV	Salida	Habilitación de reversa en los motores BLDC

Tabla 5.2: Descripción de entradas y salidas de la placa electrónica universal.

En la Figura 5.24, además de las entradas y salidas de la placa electrónica, se realizó una referencia numérica de los pines utilizados en la placa K64F (1 a 39). A su vez, se enumeraron los pines de la placa electrónica en donde se encuentran las salidas o entradas a circuitos electrónicos necesarios (filtros, adaptación de nivel, etc.) en relación a los números sobre la K64F, como así también los pines del receptor de radiofrecuencia del mando a distancia, que se encuentra en el gabinete, pero no dentro de la placa electrónica. Esto permite que una persona ajena al proyecto pueda realizar la conexión entre pines con el mismo número sin previo análisis de circuitos. Hay que tener presente que los números que figuran junto a la palabra ENCODER BITS, son los números respectivos de cada bit del encoder (sistema de dirección) y no referencias numéricas dentro de la placa electrónica.

Un caso especial es el del número 40, ya que ambos números 40 se encuentran sobre pines de la placa electrónica. Este conexionado entre pines propios de la placa electrónica se debe a que se necesita la tensión de referencia Vcc (5V) en distintas partes físicas de la placa y resultó más conveniente realizar la conexión mediante un cable, en lugar de soldar una pista de estaño, ya que esta debía ser muy larga.

Por otro lado, algunos pines de la K64F no tienen su número equivalente en la placa electrónica, estos números son 2, 16, 17 y 38. Esto se debe a que las señales que se utilizan en estos pines no se encuentran en la placa electrónica sino en otro lugar ajeno a la misma. Lo mismo ocurre con algunos números de la placa electrónica, del 41 al 48 inclusive.

Se detalla a continuación, en la Tabla 5.3, información acerca de cada pin utilizado estando enumerado del 1 al 39. Se hace mención al subsistema al que pertenecen, el nombre de la señal, el tipo, el nombre del pin en la placa y la referencia, que junto a la Figura 5.24, permite conocer el lugar físico del pin en la placa.

Subsistema	Nombre	Tipo	Pin	Referencia
CONTROL REMOTO	RECEPTORVELOCIDAD	Entrada (Capture)	PTB18	3
	RECEPTORDIRECCION	Entrada (Capture)	PTB19	4
CORRIENTE	MOTORDI	Entrada (ADC)	ADC1_DP3	19
	MOTORDD	Entrada (ADC)	ADC1_DM1	27
	MOTORTI	Entrada (ADC)	ADC1_DM0	20
	MOTORTD	Entrada (ADC)	ADC1_DP1	28
DIRECCIÓN	ENCODER (bit0)	Entrada (binaria)	PTC7	9
	ENCODER (bit1)	Entrada (binaria)	PTC0	8
	ENCODER (bit2)	Entrada (binaria)	PTC9	7
	ENCODER (bit3)	Entrada (binaria)	PTC8	6
	ENCODER (bit4)	Entrada (binaria)	PTD7	25
	ENCODER (bit5)	Entrada (binaria)	PTD5	23
	ENCODER (bit6)	Entrada (binaria)	PTD6	22
	ENCODER (bit7)	Entrada (binaria)	PTC18	24
	DIR_SENT	Salida (binaria)	PTC10	29
	DIR_PWM	Salida (binaria)	PTB11	31
	DIR_EN	Salida (binaria)	PTC11	30
MOTORES	PWM_TD	Salida (PWM)	PTC2	1
	PWM_DI	Salida (PWM)	PTC5	10
	PWM_DD	Salida (PWM)	PTC1	5

	PWM_TI	Salida (PWM)	PTC4	11
	Hall_DD	Entrada (Capture)	PTD0	12
	Hall_TD	Entrada (Capture)	PTD2	13
	Hall_TI	Entrada (Capture)	PTD1	15
	Hall_DI	Entrada (Capture)	PTD3	14
	Hall_TD	Entrada (binaria)	PTB3	33
	Hall_DD	Entrada (binaria)	PTB20	26
	Hall_DI	Entrada (binaria)	PTB2	34
	Hall_TI	Entrada (binaria)	PTB10	32
	OUT_REVERSA	Salida (binaria)	PTE24	18
	FRENO	Salida (binaria)	PTD4	21
PARADA EMERGENCIA	EMERGENCIA	Entrada (binaria)	PTC3	2
ESTADO	Status_LED	Salida (binaria)	PTE25	17
COMUNICACIÓN	Rx	Entrada (serial)	PTB16	OpenSDA
	Tx	Salida (serial)	PTB17	OpenSDA
ALIMENTACIÓN	VCC	Entrada (analógica)	Vin	35
	GND	Entrada (analógica)	GND	36
	GND	Entrada (analógica)	GND	37
	GND	Entrada (analógica)	GND	16
	3V3	Salida (analógica)	3V3	38
	3V3	Salida (analógica)	3V3	39

Tabla 5.3: Descripción de pines utilizados del microprocesador.

Con respecto a los pines de la K64F, el pin 2 y el pin 16 se conectan a los bornes de un capacitor (de 100 nF), esto se realizó para evitar que una posible presencia de ruido en la señal de parada de emergencia no haga cambiar de estado, ya que la tensión en los bornes de un capacitor no varía instantáneamente, a la señal que ingresa a la K64F produciéndose así una mala lectura. El pin 17, se deriva al circuito de LED indicadores que se encuentra en otro gabinete junto a los reguladores de tensión del robot. Por último, el pin 38 es un voltaje de 3V3 que se necesita para el circuito de parada de emergencia, el cual se encuentra en otro gabinete.

Además, vamos a comentar el resto de los pines que faltan. Como se dijo anteriormente el número 40 es una conexión entre distintos lugares de la placa que contiene Vcc (5V). Los pines del 41 al 44, son 4 pines con GND, mientras que los enumerados desde 45 a 48 son pines que tienen Vcc (5V). Estos 8 pines se necesitan para los circuitos de medición de corriente (1 par de GND y Vcc por circuito), los cuales están fuera del gabinete.

Esta placa electrónica universal fue realizada manualmente a través de soldadura con estaño. Esto es sencillo y rápido de hacer, pero trae consigo desventajas. Como, por ejemplo, que se desuelda algún componente y deje de funcionar el circuito.

Debido a esto, se decidió realizar un diseño sobre PCB que conste de estos circuitos y cumpla el mismo objetivo que la placa universal.

5.3.2. Diseño de placa electrónica PCB

Como se mencionó, uno de los objetivos que se propuso fue realizar el diseño de un PCB para implementar todos los circuitos involucrados y queden todos agrupados en una única placa electrónica. Además de los circuitos que están sobre la placa universal, se decidió integrar los circuitos de LED indicador, de sensores de corriente y de parada de emergencia.

Esta tarea se ejecutó sobre un software libre dedicado a diseño de circuitos impresos, llamado **KiCad**. En las secciones siguientes se explica con mayor detalle el procedimiento utilizado para el diseño.

Diseño de esquemático

La primera tarea fue realizar un esquemático de todos los circuitos implementados y sus interconexiones entre ellos. Los circuitos tenidos en cuenta fueron:

- Filtro de señales Hall (4 unidades).
- Filtro de señales PWM del manillar (4 unidades).
- Sensores de corriente (4 unidades).
- Led indicador (4 unidades).
- Adaptadores de señales del driver del motor paso a paso, reversa y freno (5 unidades).
- Circuito de parada de emergencia (1 unidad).

Al final del informe se adjuntará la impresión del mismo en tamaño A3, para que se pueda apreciar con mayor detalle.

Diseño de la placa

Luego de diseñar el esquemático, se procedió a diseñar la placa propiamente dicha, se acomodaron las pistas, los componentes y se previó un espacio para montar el microcontrolador (FRDM-K64F). El resultado final se muestra en la Figura 5.25.

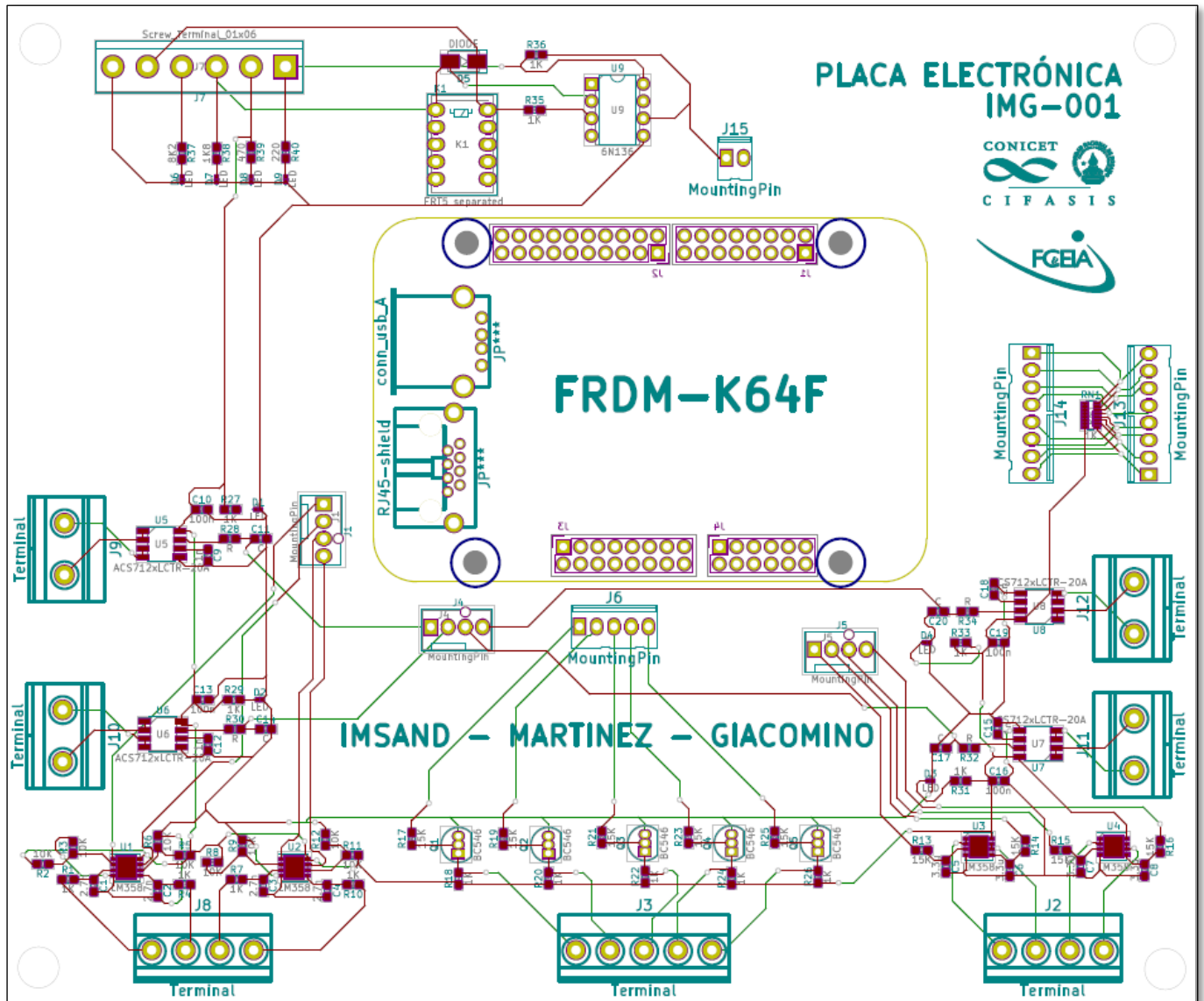


Figura 5.25: Diseño final del PCB.

Representación en 3D

Por último, **KiCad** posee una herramienta muy útil para apreciar cómo quedaría la placa impresa, mostrando una imagen en 3D de la misma en dónde se pueden sacar más conclusiones y se puede tener mayor certeza de cómo quedaría la placa impresa con todos sus componentes montados. En las siguientes figuras se muestran distintas vistas de este resultado.

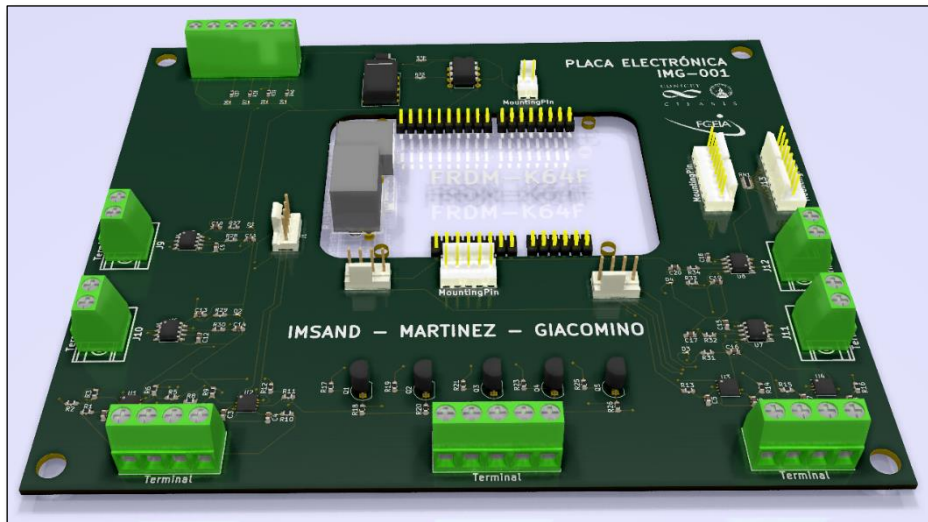


Figura 5.26: PCB visto de frente.

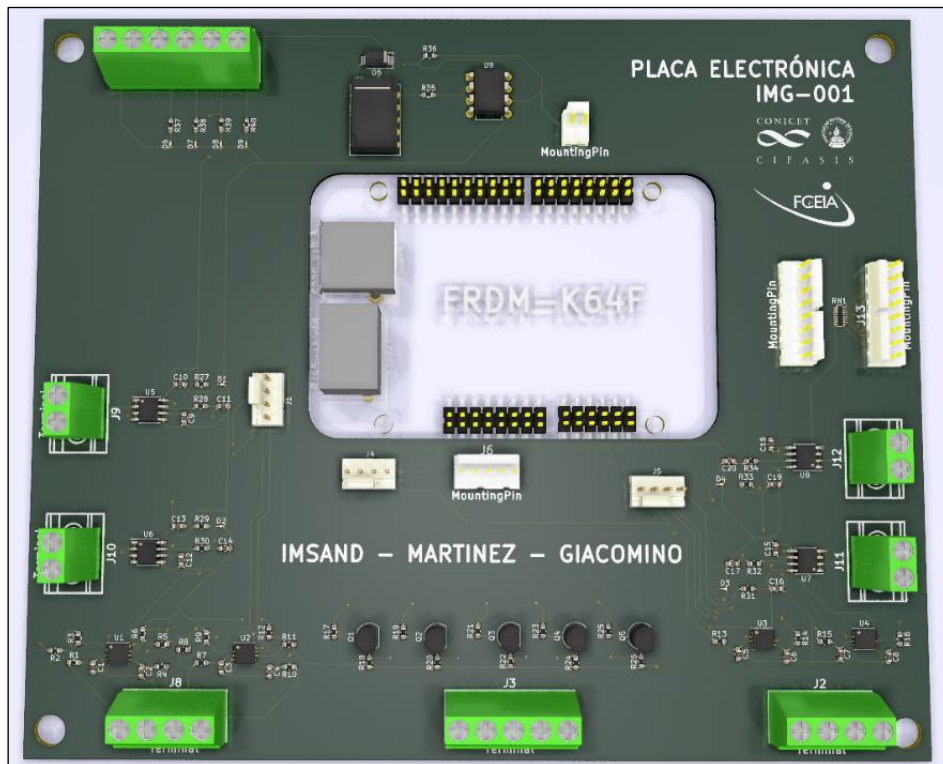


Figura 5.27: PCB visto desde arriba.

5.4. Prestaciones del sistema

Para poder llevar a cabo todas las tareas que demanda el correcto funcionamiento del robot, se debió tener presente las siguientes características:

- Medición de dirección de giro: Necesario para poder implementar un control de dirección y seguimiento de trayectoria.
- Estimación de velocidad: Necesario para poder implementar un control de velocidad y seguimiento de trayectoria.
- Parada de emergencia: Dispositivo necesario en todo sistema para reducir los riesgos que pueden perjudicar a personas, a la máquina y/o a la labor en marcha.
- Inversión de marcha: Necesario para poder esquivar obstáculos con mayor facilidad.
- Freno regenerativo: Debido a la fuente de energía limitada, es conveniente un sistema de frenado que almacena energía para su reutilización.
- Control por mando a distancia: Permite comandar y posicionar al robot por medio de un radio control.
- Control por PC: Permite comandar al robot desde la NUC.
- Comunicación: Se necesitan implementar protocolos para establecer la comunicación entre la NUC y el microcontrolador.
- Controladores de velocidad y posición: Necesarios para lograr la dirección y velocidad deseada en todo momento.
- Medición de corriente: Planteado y puesto a disposición para futuras implementaciones (por ejemplo: historización de energía consumida o detección de fallas).

Debido a esto, se necesitó relevar y modificar el software de control existente con el objetivo de saber qué características ya se encontraban cubiertas y cuales necesitaban una nueva implementación de software. A su vez, algunas de estas características, se necesitarán para la implementación de seguimiento de trayectoria.

5.4.1. Relevamiento y actualización del software de control

En lo que respecta al software, se tratará de reutilizar la mayor parte posible de código preexistente por lo que es esencial analizar exhaustivamente el mismo. Una vez culminada la etapa de relevamiento, se corregirán las fallas detectadas y se incorporarán las nuevas prestaciones.

En principio, el mayor cambio está en el sistema de tracción, mientras que el sistema de dirección no presenta cambios significativos. La parte de código abocada al control de dirección se reutilizará completamente, con la posibilidad de ordenar un cambio de dirección solo cuando el vehículo está en movimiento como medida de seguridad, ya que, al estar detenido el robot, se fuerza al motor paso a paso el cual es el encargado de comandar la dirección. También se realizará la calibración de *dirección cero* y *ángulo máximo de giro* con el objetivo de obtener resultados convenientes a la hora de realizar pruebas de campo y evitar daños sobre este sistema.

En cuanto al control de tracción, el mayor cambio radica en la potencia del sistema de tracción nuevo y, en la inclusión de inversión de marcha y freno regenerativo. Esto resuelve gran parte del control, por lo que solo nos interesa desarrollar lo correspondiente a la intervención del freno regenerativo dado que para la inversión de marcha puede reutilizarse el código existente. Por este motivo, el mayor desafío desde el punto de vista del código radica en tener en cuenta la inversión de marcha y el frenado regenerativo.

En lo que respecta a medición de velocidad y distancia recorrida, se realiza a partir de las señales de los sensores de efecto HALL.

Otro punto a considerar es la medición de corriente consumida por cada motor. Al cambiar el método de medición (reemplazo del método de resistencia SHUNT por módulos ACS712), se debe desarrollar el código correspondiente a la calibración del método y posterior medición de corriente.

Por último, la mayor parte del tiempo, el robot se encontrará en terreno rural por lo que se propone utilizar tracción en las 4 ruedas en todo momento, ya que se desconoce cuándo se puede dar una situación en la que se necesite afrontar irregularidades del terreno. En conclusión, se decidió eliminar la parte de código encargada del funcionamiento de tracción simple (solo ruedas traseras).

5.4.2. Comunicación y Modos de funcionamiento

El vehículo puede comandarse por medio del control remoto o de una computadora. Cada uno de estos modos abre dos posibilidades de trabajo, por un lado, podemos trabajar a *lazo abierto* enviando consignas de tensión para producir el avance o retroceso del vehículo y la segunda opción es a *lazo cerrado* enviando referencias de velocidad para el avance o retroceso del robot; por lo tanto, puede operar en 4 modos diferentes.

Dado que se trata de un vehículo autónomo la prioridad de control la tiene la PC, la cual, mediante comunicación serie envía tramas de mensajes al microcontrolador, dónde le indica el modo de funcionamiento que se utilizará y otros parámetros relevantes. Por su parte, el microcontrolador responde la comunicación enviando información de interés hacia la PC. El formato del mensaje se mantiene tanto en la recepción como en la transmisión y debe encontrarse dentro de paréntesis [(mensaje)] y finalizar con los caracteres de **retorno de carro** y **salto de línea** (\r \n).

La comunicación serie se establece a partir del módulo de comunicación UART (“*Universal Asynchronous Receiver/Transmitter*”), en particular, hace uso del módulo UART3 con los pines **PTC16** y **PTC17** (Figura 4.3) para la recepción y transmisión de datos, luego se adapta para recibir la información en la PC a través del puerto USB. Nos pareció conveniente cambiar el conexionado y hacer uso del adaptador **OpenSDA** incorporado en la placa, que proporciona un puente entre la PC y el procesador incorporado. Esta se puede utilizar para depuración, programación flash y comunicación serie, todo a través de un simple cable USB. En la Figura 5.28 podemos ver el esquema correspondiente.

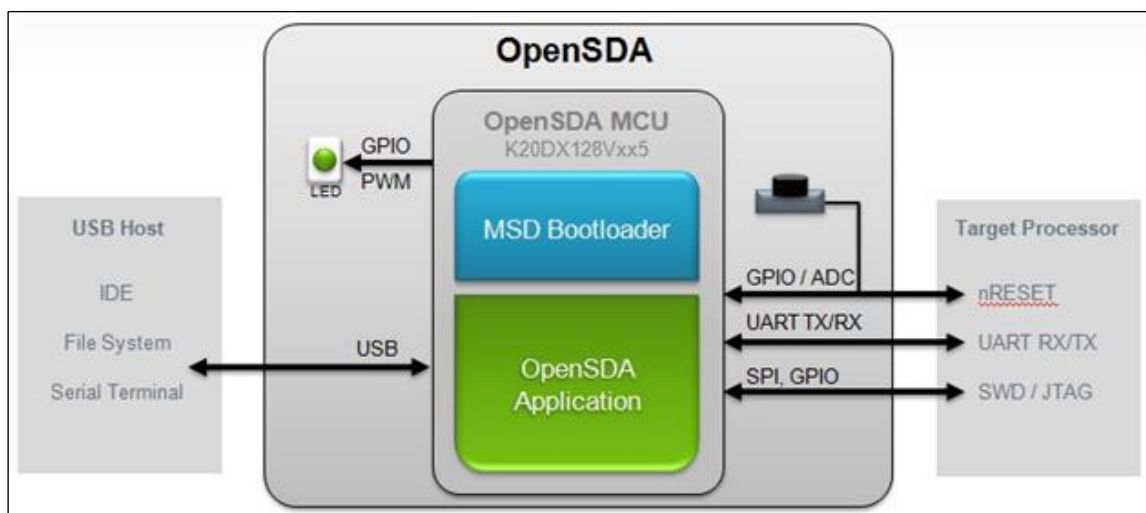


Figura 5.28: Módulo OpenSDA incluido en la placa.

Luego, la comunicación serie queda establecida con la siguiente configuración:

- Canal: UART0
- Interrupción habilitada
- Tamaño del buffer de entrada: 256 bytes
- Tamaño del buffer de salida: 256 bytes
- Paridad: Ninguno
- 8 bits de datos
- 1 bit de stop
- Recepción: PTB16/SPI1_SOUT/UART0_RX/FTM_CLKIN0/FB_AD17/EWM_IN
- Transmisión: PTB17/SPI1_SIN/UART0_TX/FTM_CLKIN1/FB_AD16/EWM_OUT_b
- Velocidad: 115200

Cabe destacar que tanto la parte de código dedicada a generar la trama de transmisión como la encargada de descifrar la trama recibida no sufrirá modificaciones, solo se pretende incorporar a la trama recibida, la señal de freno cuando sea necesaria dado que en el desarrollo anterior la misma no existía.

Mostramos a continuación la trama recibida por el microcontrolador proveniente desde la PC:

(	ESTADO	:	PARAMETROS	)	\r	\n
---	--------	---	------------	---	----	----

ESTADO 1 => Duty_REMOTO:

En este modo el sistema trabaja a *lazo abierto*, es decir, el *setpoint* que recibirá el microcontrolador será proporcional a la tensión a aplicar sobre los motores, proveniente del control remoto en forma de un porcentaje de la misma. Este modo de funcionamiento recibe las órdenes desde el control remoto una vez inicializado desde la PC con la siguiente trama, donde *dutyMáximo* corresponde al valor de duty aplicado cuando el potenciómetro de avance del control remoto se encuentra actuado al máximo posible.

(	1	:	dutyMáximo	)	\r	\n
---	---	---	------------	---	----	----

ESTADO 2 => RPM_REMOTO:

El sistema trabaja a *lazo cerrado*, recibiendo la referencia de velocidad deseada. Las órdenes son recibidas desde la señal proveniente del control remoto. Para inicializar este modo se debe enviar desde la PC la siguiente trama, indicando en *rpmMáximo* el valor de velocidad que se corresponde con el gatillo totalmente presionado.

(	2	:	rpmMáximo	)	\r	\n
---	---	---	-----------	---	----	----

ESTADO 3 => Duty_PC:

Aquí se trabaja nuevamente a *lazo abierto*, recibiendo un porcentaje de la tensión que se aplicará a los motores, pero las órdenes serán recibidas desde la PC a través de comunicación serie. Para operar este modo se debe enviar desde la PC una trama como la siguiente:

(	3	:	dutyDI	,	dutyDD	,	dutyTI	,	dutyTD	,	Reversa	,	Freno	,	Dirección	)	\r	\n
---	---	---	--------	---	--------	---	--------	---	--------	---	---------	---	-------	---	-----------	---	----	----

- **dutyXX**: valor porcentual que corresponde a la tensión a aplicar en el motor XX.
- **Reversa**: tomando valor '0' para marcha adelante y '1' para marcha hacia atrás.

- **Freno:** valor '0' indica freno liberado y '1' corresponde a la actuación del freno.
- **Dirección:** valor entre -100 y 100 desde izquierda a derecha.

ESTADO 4 => RPM_PC:

Sistema a lazo cerrado, recibiendo la referencia de velocidad deseada en RPM. Las órdenes son recibidas a través de comunicación serie desde la PC con la trama de mensaje que se muestra a continuación.

(	4	:	rpmDI	,	rpmDD	,	rpmTI	,	rpmTD	,	Reversa	,	Freno	,	Dirección	)	\r	\n
---	---	---	-------	---	-------	---	-------	---	-------	---	---------	---	-------	---	-----------	---	----	----

- **rpmXX:** referencia de velocidad para el motor XX.
- **Reversa:** tomando valor '0' para marcha adelante y '1' para marcha hacia atrás.
- **Freno:** valor '0' indica freno liberado y '1' corresponde a la actuación del freno.
- **Dirección:** valor entre -100 y 100 desde izquierda a derecha.

La trama a enviar desde el microcontrolador hacia la PC es igual para cualquier modo de funcionamiento, de la siguiente forma:

(	ESTADO	:	PARAMETROS	Dirección	,	Sentido	)	\r	\n
---	--------	---	------------	-----------	---	---------	---	----	----

ESTADO:

- 1: Duty_REMOTO.
- 2: RPM_REMOTO.
- 3: Duty_PC.
- 4: RPM_PC.

PARAMETROS:

- **Nº Motor:** 1 (rueda DD), 2 (rueda DI), 3 (rueda TD), 4 (rueda TI).
- **rpm:** Velocidad angular instantánea del motor.
- **pulso posición:** Cantidad de pulsos generados por los sensores hall de cada rueda al girar.
- **duty:** valor de duty aplicado a cada motor (0 ~ 65535)
- **corriente:** corriente consumida por el motor.

5.4.3. Relevamiento de las funciones del software

A partir del análisis mencionado anteriormente, se pudo trabajar sobre el software para lograr su actualización de manera satisfactoria. Con el objetivo de mejorar la comprensión, las tablas mostradas a continuación dan a conocer las principales funciones y rutinas de interrupciones heredadas y/o modificadas que se incluyen en el código actualizado.

Funciones	
Nombre de la función	Características Principales
int main (void)	Rutina principal. Inicialización de variables y periféricos de interés. Lazo principal, lectura y seteo de variables, manejo del modo de funcionamiento, envío y recepción de información por puerto serie.
void duty_remoto (void);	Modo duty remoto (control de tracción a lazo abierto por mando a distancia).
void rpm_remoto (void);	Modo rpm remoto (control a lazo cerrado de velocidad por mando a distancia).
void duty_pc (void);	Modo duty PC (control de tracción a lazo abierto por comunicación serie).
void rpm_pc (void);	Modo rpm remoto (control a lazo cerrado de velocidad por comunicación serie).
void calibracion (void);	Calibración del control remoto. Toma varias lecturas de la señal de velocidad y dirección enviadas por el mando a distancia y verifica que sean iguales una cantidad preestablecida de veces.
void perdida_senal (void);	Estado al que se ingresa ante la pérdida de señal del control remoto o una demora en recibir datos por puerto serie. Se resetean todos los valores de CALIBRACION, datos del controlador, setpoint de velocidad y Tensiones en los drivers.
void EMERGENCY (void);	Estado de emergencia. Seteo de la tensión suministrada a los motores a CERO logrando así detener el sistema.
void CalibrarCorriente (void);	Calibración del 'cero' de los sensores de corriente.
void init (void);	Inicialización de variables de interés.
void GetDireccion (PAP *pap_x);	Determina la dirección del robot, tomando lectura de la posición del encoder.
void GetHall (MOTOR *motor_x)	Lee sensor de efecto hall del motor_x, pone a disposición el período de la señal.
void GetVelocidad (MOTOR *motor_x)	Cálculo de la velocidad del motor_x a partir de la lectura del sensor hall.
void GetCorriente (void);	Lectura de corriente de los 4 motores.
void SetDuty (MOTOR motor_x);	Setear DUTY CYCLE del PWM correspondiente al driver de cada motor.
void set_direccion (void);	Control de la dirección.

void RX (void);	Recibe y decodifica la información que ingresa por puerto SERIE.
void TX (SERIE *serie_x);	Transmite información por puerto SERIE.
void StrcatMotor (MOTOR motor_x);	Pasa los datos del motor a un String.
void led_aux (void);	Control del led indicador de estado.
void Get_Remoto (REMOTO *remoto_x)	Adquirir datos provenientes de las señales del mando a distancia. Determinar el ancho de pulso de la señal de velocidad y dirección.
void Duty2Motor (PC *pc, MOTOR *motor)	Pasa de la variable Duty de la Comunicación SERIE a la variable Duty de cada Motor.
void Tension2Duty (MOTOR *motor_x)	Convierte variable de tensión a duty.
void NumeroFin (SERIE *serie_x)	Calcula la Cantidad de Caracteres del Buffer de Trasmisión.
long Mapeo (long x, long in_min, long in_max, long out_min, long out_max)	Mapeo de variables.
unsigned short GrayToBin (unsigned short N)	Convierte de código Gray a código Binario para la decodificación de información proveniente del encoder de dirección.
unsigned char Tension_Cero (MOTOR motor_1, MOTOR motor_2, MOTOR motor_3, MOTOR motor_4)	Retorna '1' si la suma de las tensiones de los cuatro motores es aproximadamente cero, de lo contrario devuelve '0'.
unsigned char Vel_Cero (MOTOR motor_1, MOTOR motor_2, MOTOR motor_3, MOTOR motor_4)	Retorna '1' si la suma de las velocidades de los cuatro motores es aproximadamente cero, de lo contrario devuelve '0'.
void Reset_PIDs (MOTOR motor_x)	Resetea el controlador PID.
void CtrlPID_SetK (MOTOR motor_x)	Setea el valor constante K del controlador PID.
void Error_PID (MOTOR *motor_x)	Calcula del ERROR para la entrada del controlador PID, con una saturación de ERROR máximo de 50rpm de manera de impedir inestabilidad.
void CtrlPID_Control (MOTOR *motor_x)	Llamado al controlador PID.

Tabla 5.4: Rutinas principales del código.

Interrupciones		
Nombre del evento	Características Principales	Prioridad
void Input_Hall_XX_OnCapture (void)	Captura el tiempo en el que ocurre el flanco de la señal del sensor HALL del motor XX. Primer paso en el cálculo de la velocidad del motor.	ALTA
void IntTiempo_OnInterrupt (void)	Temporizador. Periodo de interrupción: 1 ms.	MEDIA
void ADC_I_OnEnd (void)	Indica conversión completa de ADC.	MEDIA
void ReceptorVelocidad_OnCapture (void)	Interrumpe con cada flanco de la señal de velocidad del control remoto y determina a su vez el periodo de la misma.	MEDIA

void ReceptorDireccion_OnCapture (void)	Interrumpe con cada flanco de la señal de dirección proveniente del control remoto y determina a su vez el periodo de la misma.	MEDIA
void IntDireccion_OnInterrupt (void)	Control del motor Paso a Paso. Período de interrupción: 1.5ms.	MEDIA
void UART_OnRxChar (void)	Recepción correcta de un carácter y almacenado en un buffer, hasta la detección de '\r\n' que indica la finalización de la cadena a recibir.	MEDIA

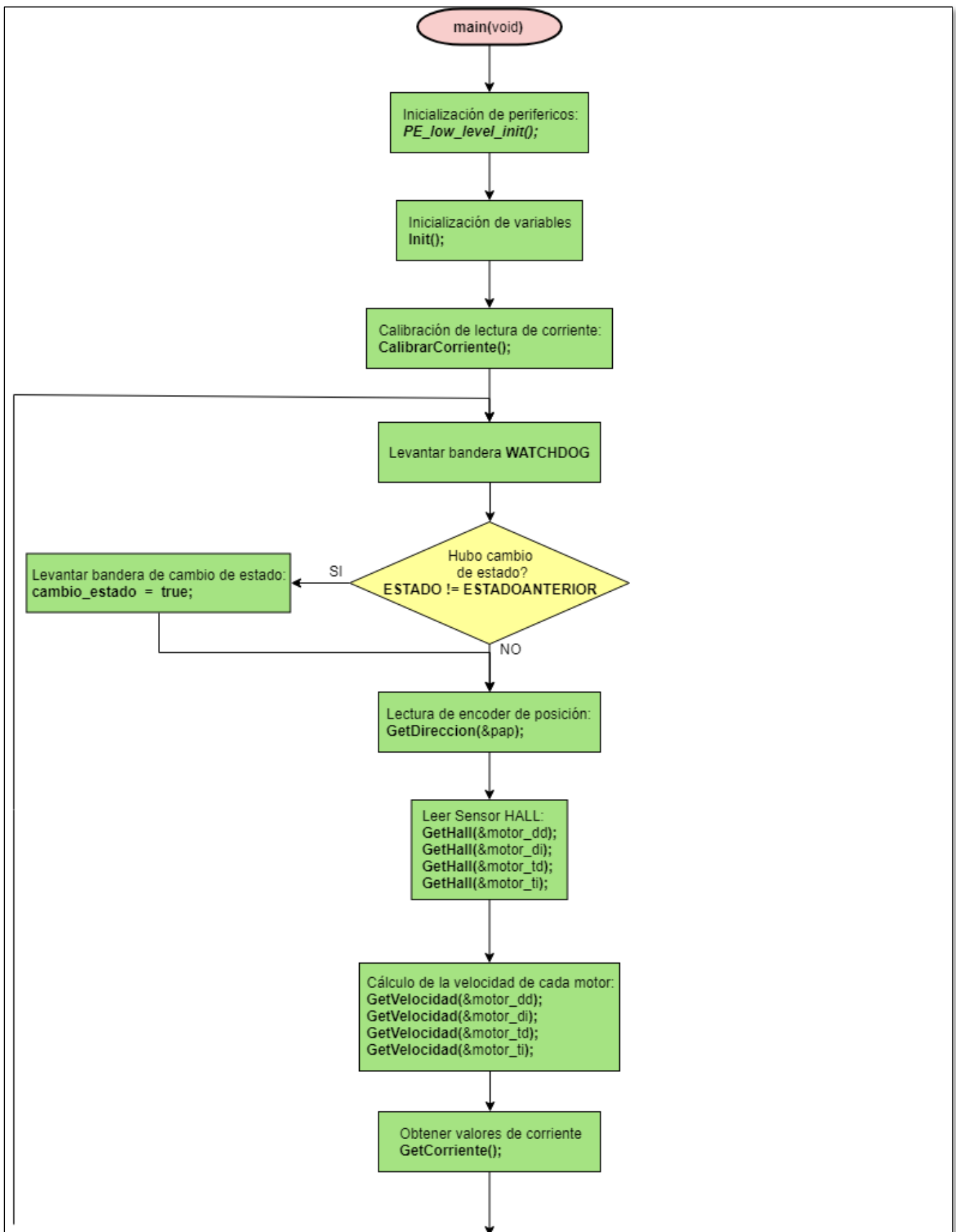
Tabla 5.5: Rutinas de Interrupciones del código.

Funciones creadas por Processor Expert	
Nombre de la función	Características Principales
void PE_low_level_init ();	Inicialización de los registros del microcontrolador
byte Input_Hall_XX_GetCaptureValue (&motor_XX.Input.aux);	Pone a disposición el valor de la captura realizada en la interrupción correspondiente, ya sea que se trate de la señal PWM de dirección o velocidad proveniente del mando a distancia, o de la señal cuadrada generada por los sensores de efecto HALL de cada motor. Esta información es relevante para el cálculo del ancho de pulso de cada señal.
Byte ReceptorVelocidad_GetCaptureValue (&velocidad.Input.datos[velocidad.Input.indices]);	
byte ReceptorDireccion_GetCaptureValue (&direccion.Input.datos[direccion.Input.indices]);	
byte UART_RecvChar (&serie.rx_buf[serie.rx_next]);	Buffer de recepción de la comunicación UART. Ubica el caracter recibido en el buffer.
void CtrlPID_XX_Control (motor_x->error_RPM,&motor_x->control);	Control PID para el motor X.
void CtrlPID_XX_Reset ();	Resetea errores acumulados del PID.
byte ADC_I_Measure (bool WaitForResult);	Comenzar lectura de ADC.
byte ADC_I_GetChanValue16 (MOTOR_XX, &motor_xx.adc);	Valor leído en el canal XX de ADC.
byte Out_PWM_XX_SetRatio16 (motor_x.duty);	Setea Duty a PWM para el motor XX.

Tabla 5.6: Otras rutinas de interés.

5.4.3.1. Rutina principal

El siguiente diagrama de flujo corresponde a la rutina principal del programa. El mismo consiste en una etapa previa para la calibración e inicialización de periféricos y variables de interés que se ejecutará únicamente al energizar el microcontrolador. Luego ingresa en el bucle principal donde se produce la lectura de los sensores para lograr la estimación de dirección, velocidad y corriente consumida, a continuación, acude al modo de funcionamiento en el que se está trabajando para calcular y setear los valores que deben tomar las variables manipuladas a partir de las lecturas realizadas. Posteriormente se realiza el chequeo del estado de la parada de emergencia y para finalizar llama a las rutinas de recepción y transmisión de datos por puerto serie.



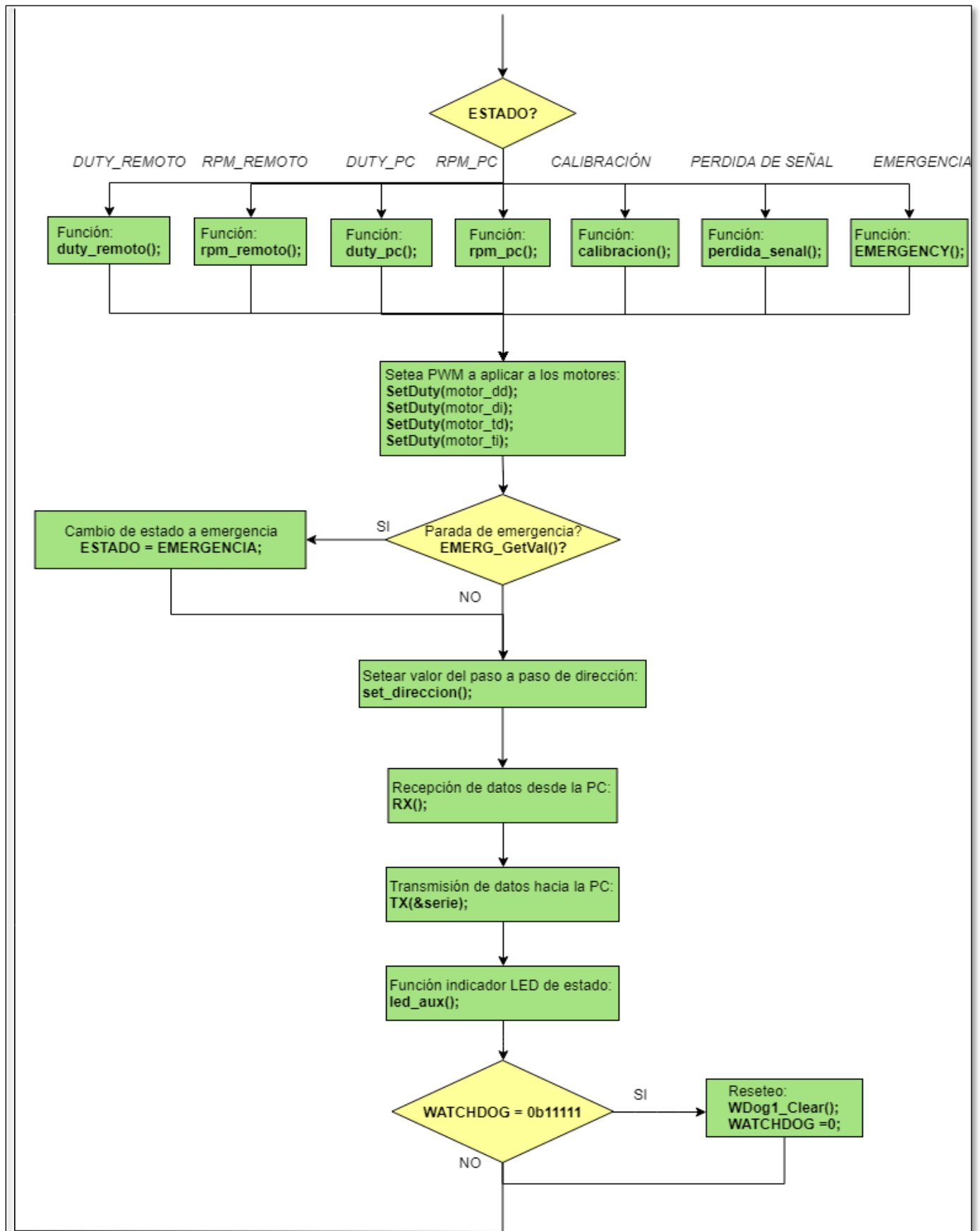


Figura 5.29: Diagrama de flujo de la función main(void).

5.4.3.2. Parada de emergencia

El vehículo cuenta con un pulsador de emergencia en la parte trasera del chasis, es un pulsador de golpe de puño *PE* con retención por giro, el cual interrumpe la alimentación de los 4 drivers de los motores de tracción y el driver del motor PaP y, además, emite una señal al microcontrolador dando aviso de que el pulsador ha sido presionado. Por otra parte, la activación de la parada de emergencia no afecta al suministro energético del gabinete de control lo que permite que el microcontrolador siga funcionando y pueda dar las ordenes determinadas cuando se encuentra en parada de emergencia.

A partir del análisis y las pruebas realizadas sobre el robot, se detectó una falla en la sección de código abocada a la parada de emergencia.

En el microcontrolador, el pin asociado a la parada de emergencia está configurado para generar una interrupción de alta prioridad, la cual resetea todas las variables de importancia.

El problema consiste en que si eventualmente se pulsa el interruptor de emergencia sin lograr que este quede actuado, no se interrumpe la alimentación de potencia del robot, pero el microcontrolador logra interpretarla e interrumpir el programa y se reestablecen las variables de importancia, entre ellas, la variable *DUTY_CERO* (asociada a la tensión suministrada a los motores) a la cual se le asigna el valor 0, esto se traduce en tensión máxima lo que genera máxima velocidad de giro de los motores ya que el circuito de potencia no se interrumpió pero la señal fue interpretada como emergencia.

Para que esto no suceda, una de las acciones a tomar fue modificar el seteo del *DUTY_CERO* a 65535, el cual da como resultado tensión nula suministrada a los motores. Por otra parte, dejamos de interpretar la señal de emergencia a través de interrupción externa de máxima prioridad y trabajamos por *polling* (puede verse en el diagrama de flujo de la rutina principal), que consiste en chequear continuamente el estado de la parada de emergencia, cuando su valor sea verdadero, el software ingresa en un nuevo estado “EMERGENCY” y se queda allí hasta que la señal de emergencia resulte falsa. De esta manera logramos mitigar el problema mencionado anteriormente. A continuación, vemos el diagrama de flujo correspondiente al estado EMERGENCY.

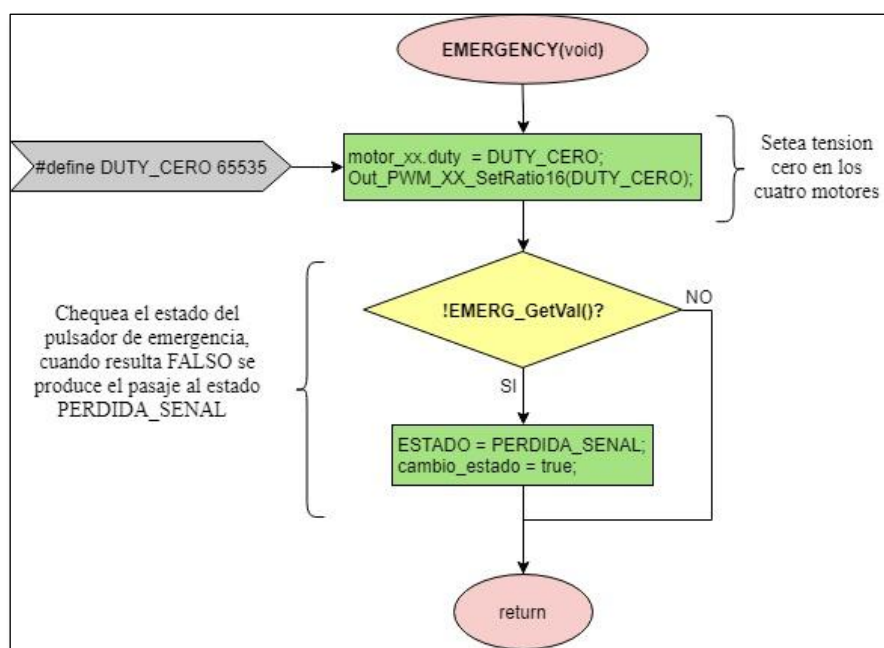


Figura 5.30: Diagrama de flujo de la función EMERGENCY.

5.4.3.3. Freno regenerativo

Para el freno regenerativo se necesita la inclusión de un pin de salida para actuar sobre el freno del vehículo, a través de una señal en alto o bajo según corresponda. Para esto usamos el pin **PTD4** (renombrado como **Freno** y configurado como output), con su respectivo circuito para la adaptación de la señal (ver sección 5.2.11).

Para implementar esta función en primer lugar se debía definir que eventos o sucesos indican que se está frenando, en los modos de funcionamientos en los que se comanda el vehículo desde la PC, la cuestión es fácil de resolver, ya que se agrega en la trama la información correspondiente a la actuación o no del freno.

En las funciones **DUTY_PC** y **RPM_PC** se agregó una condición, el pin de salida asociado al freno permanecerá en nivel bajo (**Freno_ClrVal()**) hasta recibir la indicación de freno por puerto serie, seteando el pin en alto a través de la función **Freno_SetVal()** y a su vez reinicia las variables correspondientes a cada control.

En los modos correspondientes al funcionamiento por mando a distancia, dado que el mismo solo cuenta con un gatillo para el avance o retroceso del vehículo, en las funciones **DUTY_REMOTO** y **RPM_REMOTO** implementamos el freno en el momento en que el vehículo avance en un sentido y se presione el gatillo bruscamente en sentido opuesto manteniéndolo en esta última condición. En el eventual cambio de sentido de la señal proveniente del control remoto, si la velocidad de avance de los 4 motores es distinta de cero, se pone en alto el pin de freno (**Freno_SetVal()**) y al mismo tiempo se indica tensión o velocidad cero según corresponda. Cuando el gatillo retoma su posición de reposo, el freno se suelta (**Freno_ClrVal()**).

5.4.3.4. Inversión de marcha

Para la inversión de marcha, como se explica en la sección 5.2.2, el sistema de reversa, desde el punto de vista hardware, se sustituye gracias a que los drivers adquiridos permiten realizar esta tarea mediante una entrada binaria.

En cuanto a software, solo se necesita accionar un pin de salida del microcontrolador para efectuar la tarea mencionada. Así, si se quiere invertir la marcha se deberá setear un nivel de tensión bajo, y viceversa.

5.4.3.5. Medición de corriente

Para la medición de corriente, los cuatro módulos ACS712 (uno para cada motor) devuelven una tensión analógica proporcional a la corriente que circule por la línea de potencia a medir (ver sección 5.2.5), por lo que debemos realizar una conversión analógica/digital (ADC) de cada una de las señales.

Usamos el ADC incorporado en la placa K64F, este conversor es del tipo de aproximaciones sucesivas. Las salidas de los cuatro módulos ACS712 se conectan a entradas analógicas del microcontrolador (ver Figura 4.3) y se les asigna un canal ADC configurado con una resolución de 16 bits, tiempo de conversión de 9.61 μ s e interrupción (prioridad media) al completar una lectura.

Canal 0: MotorDI => ADC0_DP0/ADC1_DP3

Canal 1: MotorDD => ADC1_DM1

Canal 2: MotorTI => ADC1_DM0/ADC0_DM3

Canal 3: MotorTD => ADC1_DP1

Dicho esto, cada vez que se complete una lectura de ADC, el programa interrumpirá y levantará una bandera indicando conversión completa que nos habilita a leer los cuatro canales mediante la función **ADC_I_GetChanValue16(Canal_correspondiente, &motor_xx.adc)**. En la Figura 5.31, se muestra el diagrama de flujo de la rutina de interrupción del ADC y, en la Figura 5.32, el diagrama correspondiente a la lectura de los canales de los ADC.

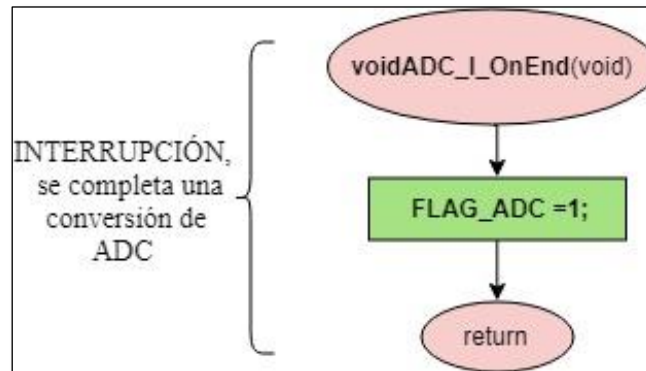


Figura 5.31: Diagrama de flujo de la rutina de interrupción del ADC.

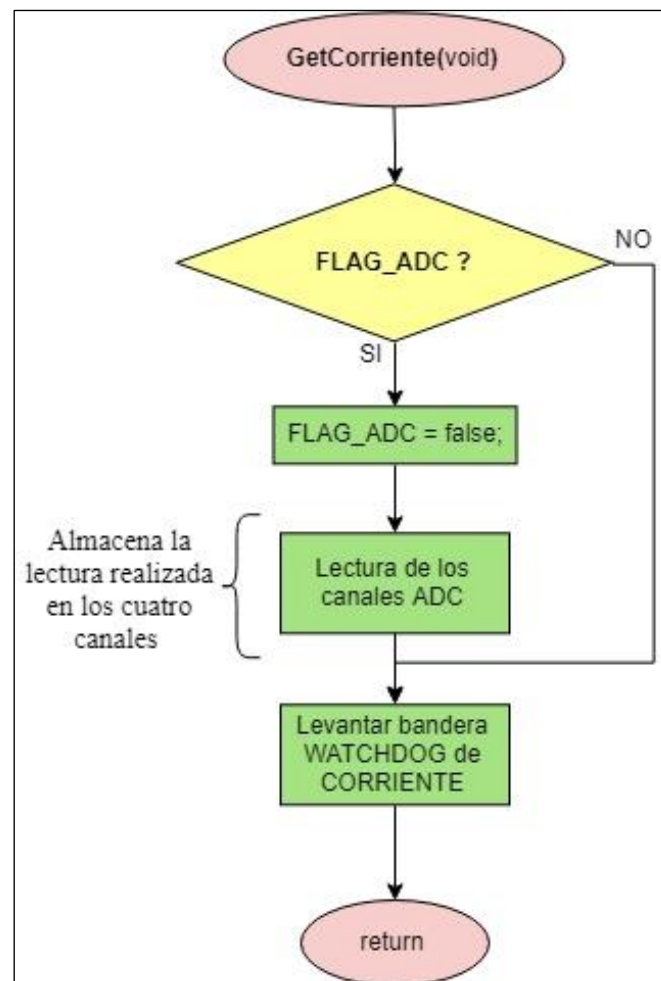


Figura 5.32: Diagrama de flujo de la función GetCorriente.

Como puede observarse en la Figura 5.11: Tensión de salida vs Corriente sensada., cuando la corriente a medir es 0A, el conjunto sensor-filtro pone a la entrada del ADC una tensión cercana a los

1.5V. Por lo tanto, lectura realizada por el ADC, corresponde a la suma de la “corriente cero” (I_o) y el valor que se desea medir (I_p).

$$I_T = I_p + I_o \quad (5.24)$$

Existe una discrepancia entre las mediciones de corriente en los cuatro motores estando el robot detenido. Esto se debe a que el ADC de aproximaciones sucesivas es de 16 bits, tendremos 65535 cuentas para un rango de tensión de entrada de 0V a 3V, lo que se traduce en 46 μ V por cuenta. Por lo tanto, pequeñas variaciones de tensión originadas por ruido o diferencias constructivas de los componentes, harán que las lecturas de “corriente cero” o valor inicial sean diferentes para cada motor y a su vez, serán diferente cada vez que encendemos el equipo. por lo que debe calibrarse la lectura de “cero” al energizar el vehículo realizando sucesivas mediciones de corriente y almacenarlas siempre que estén dentro de un umbral permitido, luego tomamos el promedio de las lecturas realizadas con el objetivo de acercarnos lo más posible al valor real correspondiente a la “corriente cero” para cada sensor. El diagrama de la Figura 5.33 lo muestra con más detalle.

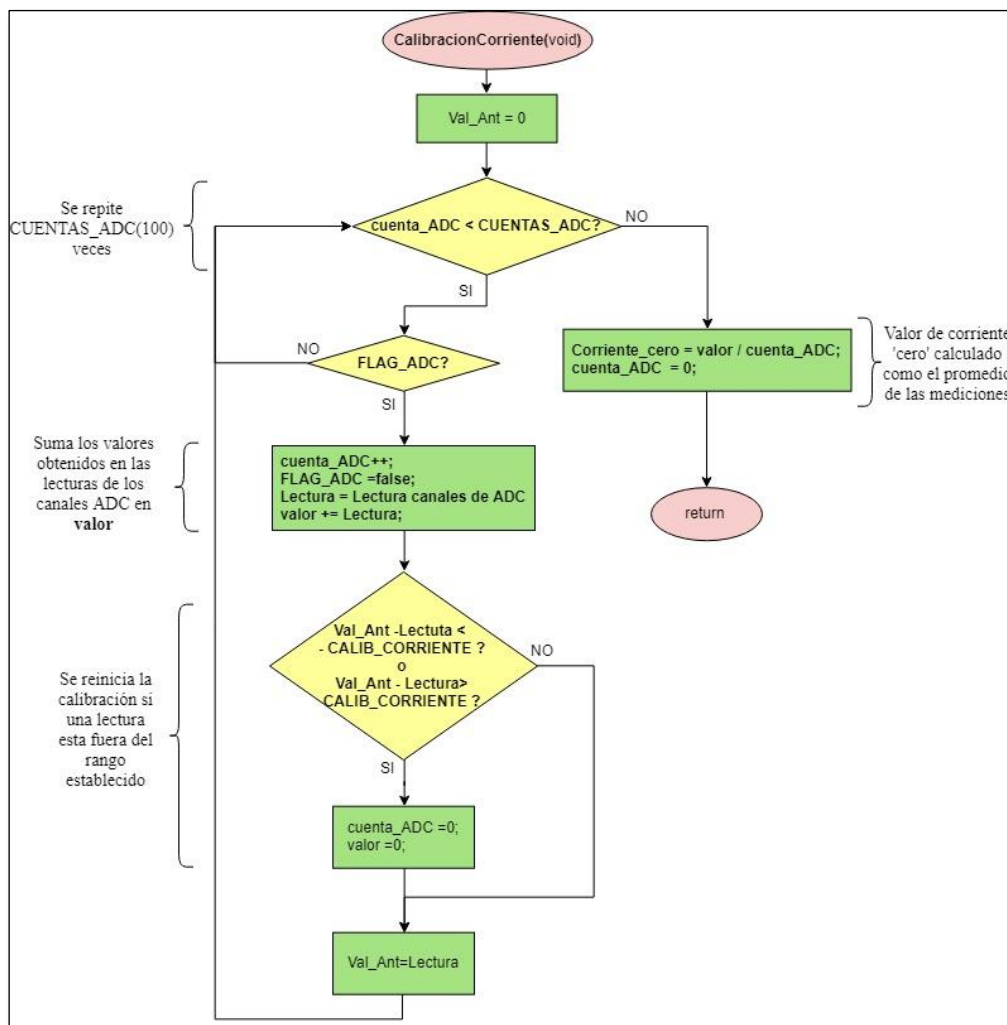


Figura 5.33: Rutina de la función CalibracionCorriente.

Dado que la lectura de corriente en el marco de este proyecto solo cumple la función de información adicional, y con la finalidad de no perder precisión en los cálculos, los valores obtenidos se envían en formato binario de 16 bits como se obtuvieron desde el ADC.

De todos modos, una vez adquirido el valor en el ADC, para obtener el valor de corriente debemos proceder de la siguiente manera:

Por un lado, trabajamos con formato binario de 16 bits, por lo tanto, las lecturas de ADC están entre 0 y 65535, y según lo especificado en la sección 5.2.5, $V_{DDA} = 3V$. Así, el ADC transforma una tensión de entrada de $(0 \sim 3)V$ en $(0 \sim 65535)$ cuentas respectivamente.

En otras palabras, y siendo que V_{out} corresponde a la tensión que ingresa al ADC y es nuestro valor de interés a medir, tenemos:

$$\frac{V_{out}}{V_{DDA}} = \frac{Lectura_{ADC}}{65535} \quad (5.25)$$

Como conocemos que la sensibilidad del sensor afectada por la ganancia del filtro es $0.06V/A$ (remóntese a la sección 5.2.5 para más detalle), podemos expresar la tensión de interés que ingresa al ADC de la siguiente manera:

$$V_{out} = I_T \cdot 0,06 \frac{V}{A} \quad (5.26)$$

Donde I_T corresponde a la suma de I_P (corriente que se desea medir) e I_0 (corriente “cero”). Por último, sustituyendo y agrupando las ecuaciones (5.24), (5.25) y (5.26) tendremos el valor de corriente deseado:

$$I_{P[A]} = \left(\frac{Lectura_{ADC[Cuenta]}}{65535_{[Cuenta]}} \times \frac{1}{Sensibilidad_{[V/A]}} \times V_{DDA[V]} \right) - I_{0[A]} \quad (5.27)$$

$$I_P[A] = \frac{Lectura_{ADC[Cuenta]}}{65535_{[Cuenta]}} \frac{3V}{0,06 \frac{V}{A}} - \frac{Corriente_cero[Cuenta]}{65535_{[Cuenta]}} \frac{3V}{0,06 \frac{V}{A}} \quad (5.28)$$

5.5. Navegación autónoma

Una de las prestaciones con las que el robot pulverizador cuenta, es la navegación autónoma, o sea, es conducido por sí mismo en funcionamiento normal, aunque se sabe también que brinda la herramienta de conducción por mando a distancia. Para que el robot sea autónomo, se debió contemplar cómo iba a tomar las decisiones de conducción en base a donde se encontraba y hacia donde debía ir. Para lograr esto, se aplicó un control de velocidad y de dirección. El control de ambas variables de interés se computa en la NUC del robot, y para eso, se realizó un software que ejecuta las leyes de control y, como se mencionó, en base a estas, su ubicación actual y la ubicación objetivo a la que desea llegar, se manipula la dirección y la velocidad del vehículo. Finalmente, para lograr llevar a cabo lo mencionado, se realiza una comunicación entre la NUC y el microcontrolador quien se encarga de brindar las consignas a los motores y de devolverle a la NUC la velocidad y la dirección real del robot.

Las leyes de control a utilizar, determinadas en la Sección 3.2.3, requieren parámetros a determinar que dependen de la aplicación y características físicas del vehículo, entre otras cosas. Control de velocidad y dirección del vehículo

5.5.1. Elección de parámetros de las leyes de control

Luego del análisis efectuado en la sección 3.2, y teniendo las leyes de control y el algoritmo a utilizar, se procedió a determinar los parámetros relevantes para la solución del problema.

Al aplicar el algoritmo de *path tracking* “**Follow the carrot**” se decidió que los puntos intermedios (waypoints) se encuentren a una distancia fija del robot. Esto, en conjunto con la ley de control de velocidad, nos hace notar que esta última debe ser constante, pues $v = \gamma e$, donde e es la distancia fija entre el robot y el waypoint. Entonces, la primera decisión que se tomó fue fijar la velocidad a la que se movería el robot, se optó por: $v = 0.4$ m/s, ya que era una velocidad aceptable para el robot teniendo en cuenta la aplicación del mismo. Seguido a esto, procedimos a la elección del parámetro γ , decidimos que sea: $\gamma = 0.4$. En principio esta elección no se basó en nada, por lo que restaba ensayar el comportamiento del robot para tener una orientación de la decisión tomada.

Elegidos v y γ , la distancia e que se le dió a los puntos objetivos intermedios que se calcularán, debe ser 1 m, pues queremos que la velocidad sea fija. Como esta distancia depende de la velocidad a la que se desea que se mueva el robot y del parámetro adoptado, el cual no tuvo criterio alguno, se debió ensayar el comportamiento del vehículo y poder sacar conclusiones por prueba y error. Tanto este parámetro como los de la ley de control, que se explicarán posteriormente, debieron adoptarse sin ningún criterio ya que no se contaba con un sistema exacto del robot para poder simular el comportamiento del mismo y tener una idea mas acertada de la adopción de parámetros.

Comenzamos con la elección de valores pequeños de e y lo ajustamos hasta lograr un valor satisfactorio. Con $e = 1$ m detectamos que el robot tenía problemas de oscilaciones mencionados en la Sección 3.2.4. Así, fuimos aumentando su valor, pero al mismo tiempo debíamos disminuir, en la misma magnitud, el parámetro γ para mantener la velocidad igual. Así, tras varias pruebas se decidió por elegir los siguientes valores: $e = 4$ m y $\gamma = 0.1$. Cabe destacar que estas pruebas se realizaron con las ruedas del robot suspendidas en el aire, sacando conclusiones del comportamiento del mismo con los distintos parámetros ensayados.

El próximo paso era la elección de los parámetros que influyen sobre la ley de control de la curvatura, es decir, la dirección de giro del robot. Estos son h y β .

De la ecuación (3.14) se sabe que $h > 1$, así que adoptamos $h = 1.2$. Por otro lado, se tiene que cumplir que $2 < \beta < h + 1$, por lo que adoptamos $\beta = 2.1$. Estos parámetros fueron elegidos con criterio de que cumplan las condiciones pero que no estén muy lejanos a sus límites inferiores para poder realizar ensayos de prueba y error, ya que no teníamos noción alguna de que valores eran aproximados a los que el robot necesitaba debido a la faltante de un sistema completo del robot para poder simular el control.

Lo que se debió asegurar fue que, con esos parámetros elegidos, el robot iba a girar su dirección lo suficiente como para alcanzar al objetivo y, si esto no ocurría, debíamos corregir los parámetros asignados.

Una de las formas de ensayar esto fue enviarle un punto conocido sobre la circunferencia de menor radio que podía realizar el robot y corroborar que realmente lograba llegar a destino, en otras palabras, así logramos determinar si la curvatura que adopta bajo la ley de control es suficiente. Si se logra, por lógica,

para circunferencias de radio mayor, también se tendrá éxito. Una de las preguntas que se podrían hacer es, ¿Qué pasa si, con la elección de estos parámetros, la ley de curvatura nos da un valor de la misma, mayor al máximo? No habrá problemas, ya que esta elección de la curvatura c estará limitada por software y no podrá ser mayor que un cierto valor determinado. Así, le dimos de objetivo el punto con coordenadas cartesianas (0,4.722) y con una orientación del robot de $\phi_d = \pi$; el cual exigía al robot a hacer la mitad de la circunferencia de menor radio posible, ya que para el valor máximo de ángulo de giro de la dirección y la longitud, y siguiendo la ecuación (3.16) podemos saber que el radio mínimo es 2.361.

La prueba se llevó a cabo con éxito por lo que no se modificaron los parámetros elegidos, en la Figura 5.34 se muestra la trayectoria. A la izquierda, se observan 2 circunferencias (de radio mínimo) en línea de puntos, las cuales encierran los puntos que el robot no puede alcanzar, con el robot en el origen, representando el momento de salida del robot, o sea, el inicio del recorrido. A la derecha, se observan las mismas 2 circunferencias y en la superior, en línea continua, se muestra el recorrido realizado por el robot, el cual se encuentra en el punto de coordenadas (0,4.722) con dirección $\phi = \pi$, o sea, el momento final del recorrido.

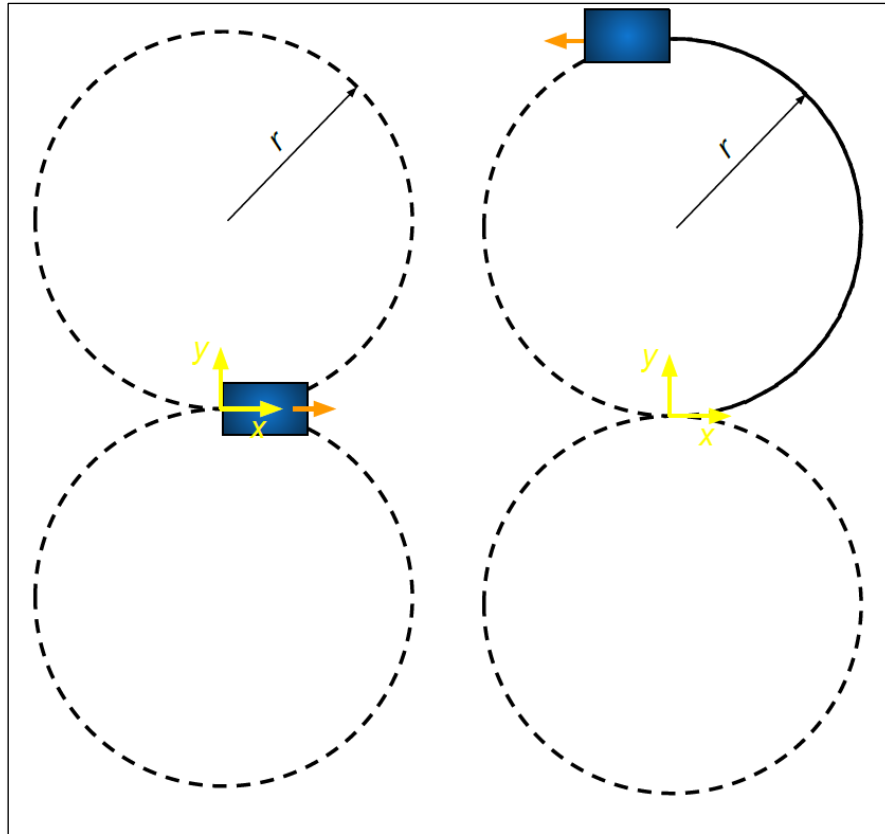


Figura 5.34: Trayectoria del robot al darle como objetivo el punto (0,4.722) con orientación π .

Teóricamente, esto se puede demostrar, con la ecuación (3.12), que representa la ley de control de la curvatura.

$$c = \frac{\sin \alpha}{e} + h \frac{\theta \sin \alpha}{e \alpha} + \beta \frac{\alpha}{e} \quad (5.29)$$

Donde: $e = 4$; $h = 1.2$; $\beta = 2.1$; $\alpha = \pi/2$; $\theta = -\pi/2$. Con estos parámetros la curvatura resultante es la siguiente.

$$c = \frac{\sin \pi/2}{4} + 1.2 \frac{-\pi/2 \sin \pi/2}{4 \pi/2} + 2.1 \frac{\pi/2}{4} = 0.25 - 0.3 + 0.824 = 0.774 \quad (5.30)$$

Por otro lado, se sabe que la curvatura máxima se muestra en la ecuación (3.15). Reemplazando por los valores reales, obtenemos:

$$c_{max} = \frac{\tan \psi_{max}}{l} = \frac{\tan 35^\circ}{1.65} = 0.424 \quad (5.31)$$

Se concluye entonces que la curvatura que tomará el vehículo será la máxima ya que, aunque la resultante es mayor, se encuentra limitada por ésta. Podemos concluir entonces que los parámetros elegidos para la ley de curvatura, son suficientes para alcanzar los puntos de la circunferencia de menor radio.

5.5.2. Simulación

Con las leyes de control vistas anteriormente, los parámetros adoptados para las mismas, las ecuaciones que componen al modelo de Ackermann y los límites que posee el vehículo en particular, se procedió a realizar una simulación del comportamiento del sistema para distintos objetivos. Las simulaciones fueron hechas con la herramienta *Simulink* que nos provee el software *Matlab*.

Las trayectorias que le indicaremos como consigna estarán compuestas por un conjunto de puntos objetivos, los que constarán de sus coordenadas en el plano (x_d, y_d) con el agregado de la orientación deseada ϕ_d .

En la Figura 5.35, se muestra un diagrama de bloques de todo el sistema interconectado, los bloques más relevantes, a los cuales explicaremos con mayor detalle, son: “**Determinación de puntos**”, “**Control**” y “**Robot**”.

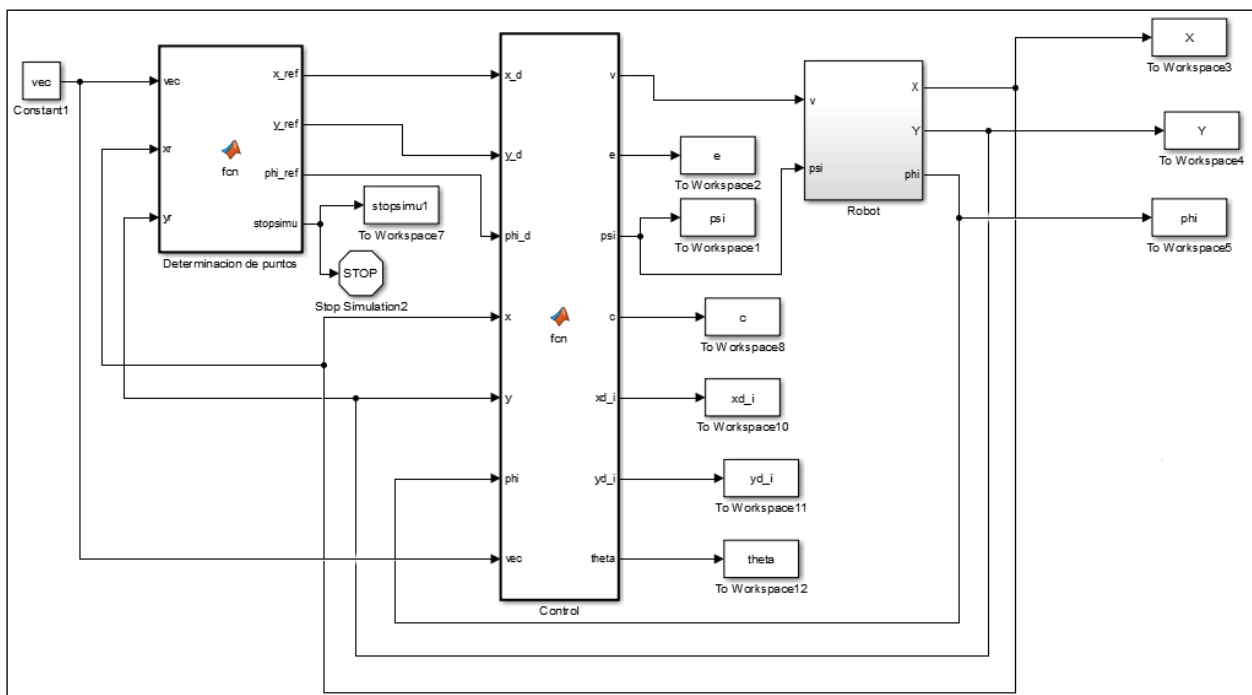


Figura 5.35: Diagrama de bloques del sistema completo.

- **Determinación de puntos:** Es el más sencillo de los tres. Este bloque tiene la función de, a partir de un vector con la serie de puntos objetivos a seguir junto a la posición en x e y actuales del robot realimentadas desde el bloque **“Robot”**, su función es determinar cuál es el próximo punto objetivo y seleccionarlo como punto objetivo de salida, el cual será brindado como dato al bloque **“Control”**. Además, cuenta con una salida (*stopsimu*) que se encarga de detener la simulación cuando el robot alcanza al último punto objetivo, es decir, al último punto del vector de entrada y de la trayectoria.
- **Control:** Este es el encargado de ejecutar el algoritmo de control de dirección y de velocidad. Por eso, este debe ordenarle al robot la velocidad y la dirección que debe tener (v y ψ respectivamente) como referencia, se calculan bajo las leyes de control mencionadas en la Sección 3.2.3. Las demás salidas, son datos que se almacenan para tener referencia de los puntos objetivos intermedios. Las entradas que necesita este bloque son: el punto objetivo (referencia) compuesto por sus coordenadas x_d , y_d , y ϕ_d y el punto actual con las coordenadas x , y , y ϕ . Como es de esperar, las variables con subíndice d provienen de **“Determinación de puntos”** ya que son las referencias, y las variables sin subíndice provienen de **“Robot”** y representan la posición actual del mismo. La entrada restante es el vector de puntos objetivos, necesaria para mantener la velocidad constante durante todo el recorrido hasta el momento en el que se acerca al último punto objetivo del vector, momento en que la velocidad comienza a disminuir de forma proporcional con la distancia al punto hasta hacerse nula al llegar al objetivo.
- **Robot:** Este bloque representa un modelo del robot y es el encargado de calcular la posición del robot en todo momento (x , y , y ϕ). Necesita como parámetros de entrada la velocidad y la dirección de giro provistas por el bloque **“Control”**. La lógica interna del bloque se muestra en la Figura 5.36.

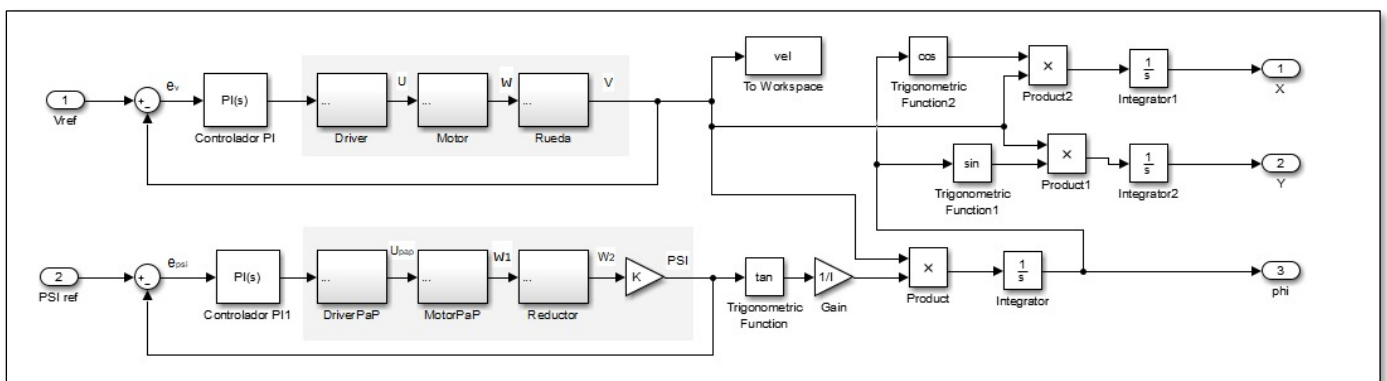


Figura 5.36: Esquema interno del bloque **“Robot”**.

En la Figura 5.36, se observan los bloques que corresponden al robot más el cálculo de la posición por odometría. En el funcionamiento de los motores, además del propio motor, el correspondiente driver y demás componentes, se encuentran los controles PI realizados para los mismos.

Despreciando la dinámica del BLDC y del PAP, y considerando que los controles PI convergen rápidamente, se tiene que este bloque solo tiene ecuaciones matemáticas referidas a parámetros del robot, como ser la longitud del mismo, expresadas en la Sección (3.3).

A continuación, se mostrarán algunos ejemplos de seguimiento de trayectoria realizados. Con cruz negra representaremos los distintos puntos objetivos de la trayectoria provenientes del bloque vector que los contiene, con cruz roja los puntos objetivos intermedios (*waypoints*) generados, calculados en el bloque que realiza el control, y con línea recta azul el recorrido realizado por el robot, la velocidad y la orientación.

Los *waypoints* se generan dentro del bloque de control y la manera en que se hace esto es conociendo la posición actual del robot y la del punto objetivo que se quiere alcanzar. La forma de realizar esto, es calculando el ángulo de la recta que une a los puntos de la posición del robot y del punto objetivo con respecto a la coordenada y del robot. Esto se realiza con una función denominada *atan2* la cual tiene este propósito. Una vez que tenemos el ángulo, se calcula la posición del *waypoint* en ambas coordenadas, esto se realiza de manera sencilla, pues este punto se encontrará siempre a 4 metros del robot por lo que la posición en x será la distancia de 4 metros multiplicada por el coseno del ángulo calculado (por *atan2*) sumada a que tiene el robot actualmente. Para la coordenada y es similar, excepto que se la multiplicación se realiza por el seno del ángulo calculado. En la Figura 5.37 se ilustra un ejemplo de cómo se realiza el cálculo, donde el subíndice a corresponde a la posición actual del robot, o a la posición del punto objetivo, d a la diferencia que hay entre las variables de subíndice o con a y w a la posición del punto intermedio *waypoint*.

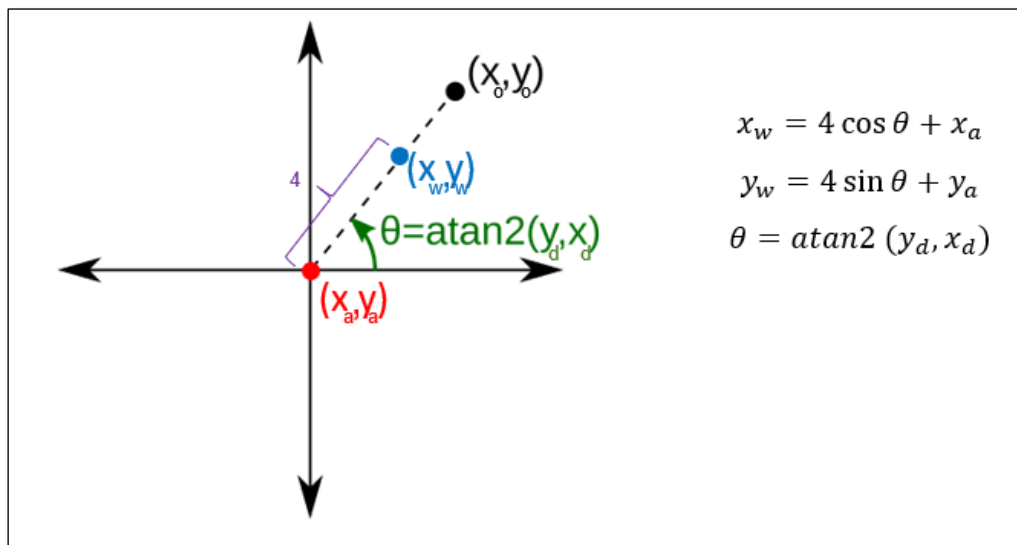


Figura 5.37: Cálculo de las coordenadas de *waypoint*.

Primero se mostrará una tabla con los puntos objetivos que contiene el vector, y luego algunas gráficas de interés como la velocidad v en función del tiempo, la orientación ϕ en función del tiempo y la posición en el plano (x,y) .

Trayectoria 1 (Recta y curvas)

Como primera trayectoria a realizar, se optó por una serie de rectas, y circunferencias con sentido de giro hacia ambos lados. El robot comienza desde una posición inicial que será el origen en el marco de referencia (x,y) , apuntando el frente del mismo hacia el eje x positivo. Realizará una recta sobre el eje x , una semicircunferencia, otra recta, un cuarto de una circunferencia y otro cuarto cambiando el sentido de giro, alcanzado aquí el origen de coordenadas, pero su frente queda apuntando hacia el eje x negativo. Los puntos objetivos que se le brindan como referencia, son los siguientes.

Nro.	x	y	ϕ
Inicio	0	0	0
1	15	0	0
2	20	5	$\pi/2$
3	15	10	π
4	10	10	π
5	5	5	$-\pi/2$
6	0	0	$-\pi$

Tabla 5.7: Puntos objetivos de la trayectoria 1.

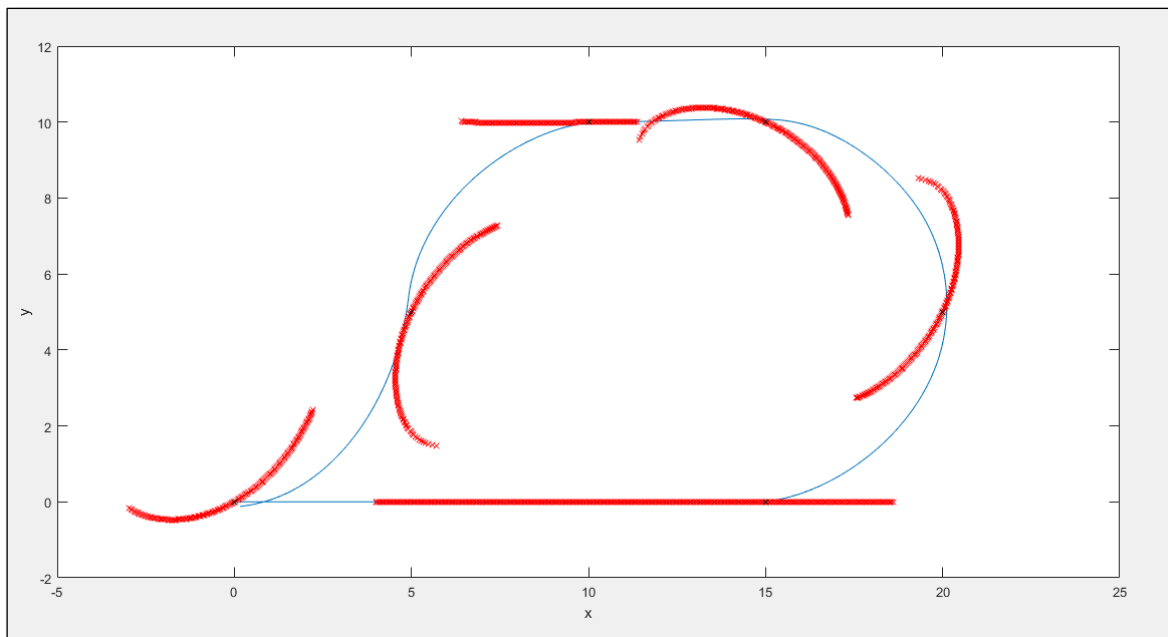


Figura 5.38: Gráfica y vs x de la trayectoria 1.

La Figura 5.38 es la más ilustrativa para conocer la posición en (x,y) del robot, y el recorrido que este realizó. La orientación que el robot posee no puede ser apreciada, pero es imaginable sabiendo que inicia con orientación nula (frente hacia el eje x positivo) y conociendo los puntos objetivos de la trayectoria. También podemos apreciar los waypoints en rojo, y los puntos objetivos en negro, a los cuales el robot llega satisfactoriamente, pues la trayectoria del robot pasa por estos puntos. En los waypoints, puntos rojos, se aprecian discontinuidades o saltos, esto se debe a que cuando el robot alcanza un punto objetivo de la trayectoria, se actualiza con el punto objetivo siguiente y, si no se encuentra en la misma dirección que el punto objetivo anterior, el robot deberá doblar, ya que el próximo waypoint estará en la dirección de la recta que une al robot con el nuevo punto objetivo.

Otra cosa que se puede apreciar, es que hay waypoints en la zona cercana a $y = 0$, con $x < 0$, estos se generan en el recorrido del robot hacia el último punto objetivo pero no deben ser alcanzados, por eso, en un momento pasan a ser ignorados y el punto a alcanzar queda fijo, siendo el último punto objetivo de la trayectoria, el número 6 de la tabla. La ignoración de estos waypoints se debe a que el robot está finalizando la trayectoria, y debe detenerse en el último punto objetivo, en este momento la distancia e entre el robot y el punto que debe alcanzar, deja de ser fija debido a que el punto a alcanzar es fijo (punto final de trayectoria) y que el robot se aproxima al mismo.

A continuación, podemos observar la evolución de la velocidad que, al ser proporcional a la distancia, nos permite tener una idea de la evolución de la misma, que será similar.

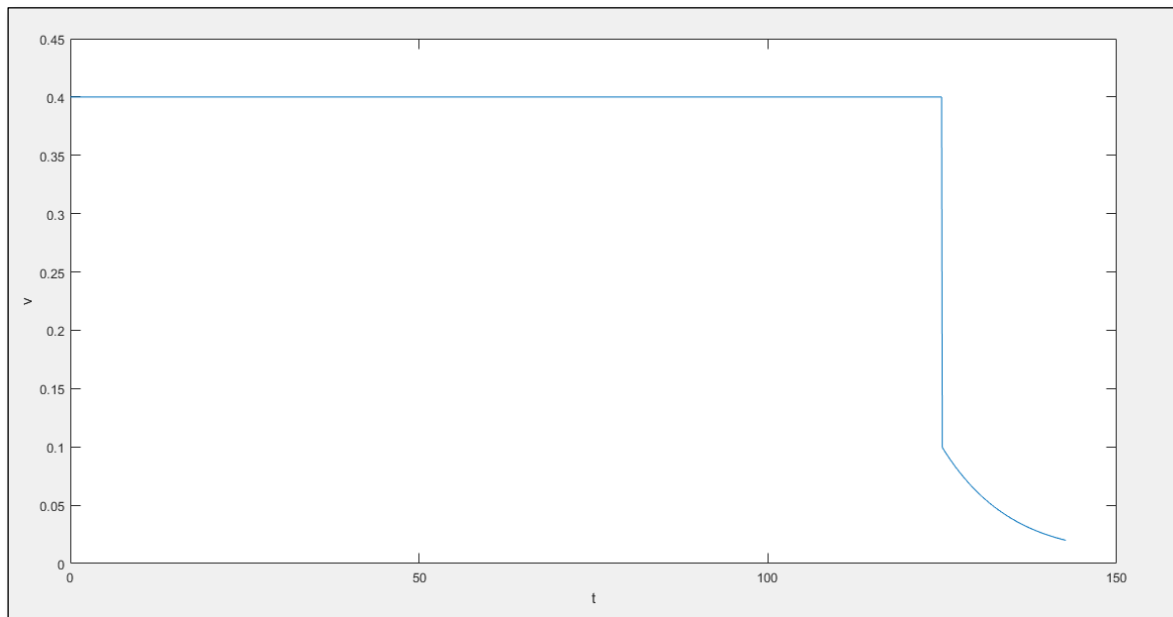


Figura 5.39: $v(t)$ para la trayectoria 1.

El análisis que requiere no es muy complejo, al comienzo arranca a una velocidad constante de 0.4 m/s, y se mantiene así hasta que comenzamos a ignorar los *waypoints* y fijamos nuestro punto objetivo como el último de la trayectoria. Es aquí, donde la velocidad comienza a disminuir proporcionalmente con la distancia, la cual disminuye a medida que el robot se acerca al punto objetivo final. Esta situación se resume en la Figura 5.39.

Como vemos, la velocidad no alcanza a ser nula. Es aquí cuando el bloque **“Determinación de puntos”** detiene la simulación. En la práctica esto será equivalente a darle al robot como consigna una velocidad nula, provocando su detención.

El motivo por el que la velocidad no es nula se debe a que, en la simulación, hay un margen de error de posición que le brindamos intencionalmente, el mismo es de 0.2 m. Esta decisión se tomó porque, si el robot no alcanza exactamente la posición deseada, el mismo intentará volver a esta, ya que la distancia entre el robot y la posición a alcanzar no es nula, por lo que tampoco es nula la curvatura ni la velocidad debido a las leyes de control, y el robot nunca se detendrá.

Otro dato relevante, es conocer la evolución de la orientación del robot por lo que a continuación se ilustra la misma.

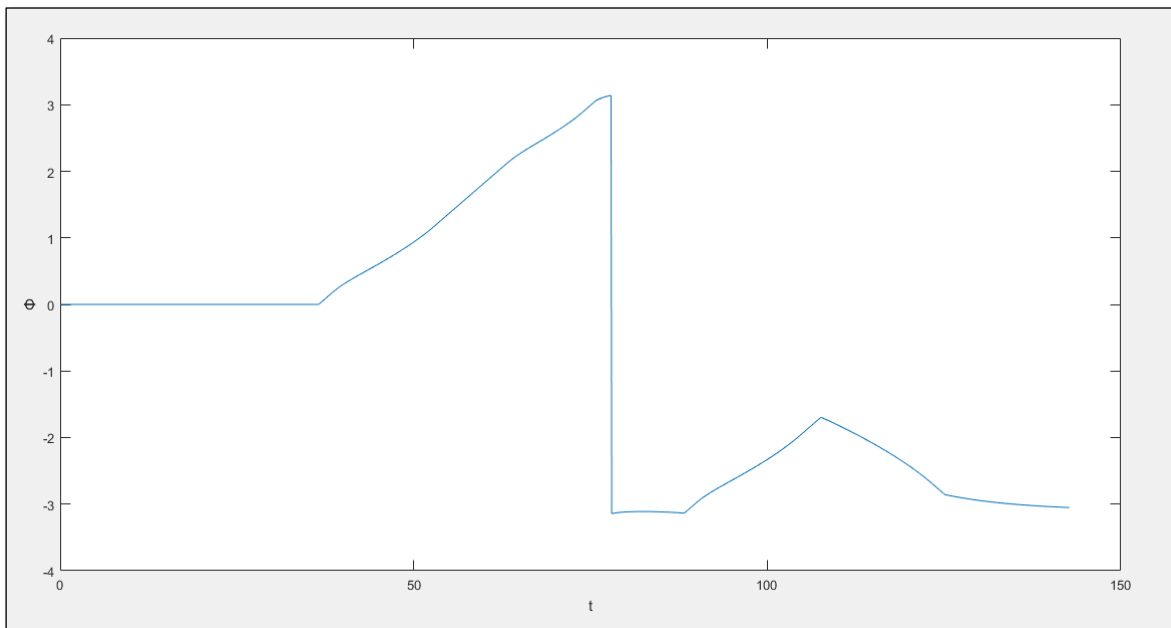


Figura 5.40: $\phi(t)$ para la trayectoria 1.

En la Figura 5.40, se muestra la orientación ϕ del robot. Aquí vemos que comienza en 0 rad, y se mantiene así hasta alcanzar el punto objetivo 1. Luego comienza a aumentar, lo cual es lógico, pasando por $\pi/2$ rad (punto objetivo 2) y llegando a π rad (punto objetivo 3). Es importante aclarar la situación que se da en estos momentos. Antes que nada, hay que decir que todos los ángulos con los que trabajamos, están normalizados en el intervalo $[\pi, -\pi]$ y, además, como había un margen de error en la posición (x, y) , también, por el mismo motivo, lo hay en el ángulo del robot ϕ y es de 0.175 rad, equivalente a 10° . Si bien consideramos que podría ser menor, recordamos que a nosotros no nos interesa demasiado que se cumpla con el ángulo de orientación del robot a la hora de recorrer la trayectoria planeada, aunque nos permite que el robot pueda seguir una trayectoria a través de puntos objetivos.

Entonces, en un momento el robot alcanza al punto objetivo 3. Este punto se alcanza con un error en el ángulo de orientación, o sea, no es π como se espera, sino un poco menos. Esto en conjunto con que el robot ya busca alcanzar al próximo punto objetivo, y su velocidad es 0.4 m/s, hace que el robot se pase en la coordenada y, o sea, la posición actual en y del robot es mayor a la que debe lograr tener en el punto objetivo 4. Por esto, la ley de control hace girar al robot logrando un ángulo de orientación mayor que π , que al normalizarlo en el intervalo $[\pi, -\pi]$, queda negativo y levemente más grande que $-\pi$. Esto se aprecia en la figura con el salto significativo que se da aproximadamente en $t = 77$. Luego, alcanzado el punto objetivo 4, el robot sigue girando hacia la izquierda, pues el punto objetivo 5 tiene un ángulo ϕ_d de $-\pi/2$ rad. Finalmente, para alcanzar al último punto el robot debe tener una orientación de $-\pi$, por lo que necesita cambiar su giro hacia la derecha.

Trayectoria 2 (Símbolo ∞)

Nro.	x	y	ϕ
Inicio	0	0	0
1	5	5	$\pi/2$
2	0	10	π
3	-8	5	$-3\pi/4$
4	-16	0	$-\pi$
5	-21	5	$\pi/2$
6	-16	10	0
7	-8	5	$\pi/4$
8	0	0	0

Tabla 5.8: Puntos objetivos de la trayectoria 2.

Esta trayectoria, es similar a un número 8 horizontal, o al símbolo infinito. Se le pasarán 8 puntos objetivos, y veremos cómo responde el modelo. Cabe aclarar que cuanto mayor sea la cantidad de puntos ofrecidos como datos, la silueta dibujada será más precisa. Un claro ejemplo sería el deseo de que el robot realice una circunferencia. Supongamos que es de radio muy grande en relación a las condiciones físicas del robot (dirección y longitud), por ejemplo 200 m, y que solo le pasamos 8 puntos distribuidos de igual manera; las probabilidades de que se cumpla con el objetivo de orientación antes que el de posición en coordenadas (x, y) son muy altas, produciendo la llegada del robot a los puntos objetivos en línea recta. Esto hay que tenerlo presente a la hora de brindarle datos a nuestra trayectoria, ya que dependerá de las distancias que tenga la misma.

Volviendo a la trayectoria realizada, mostraremos gráficas relevantes, las mismas que mostramos en el ejemplo de Trayectoria 1.

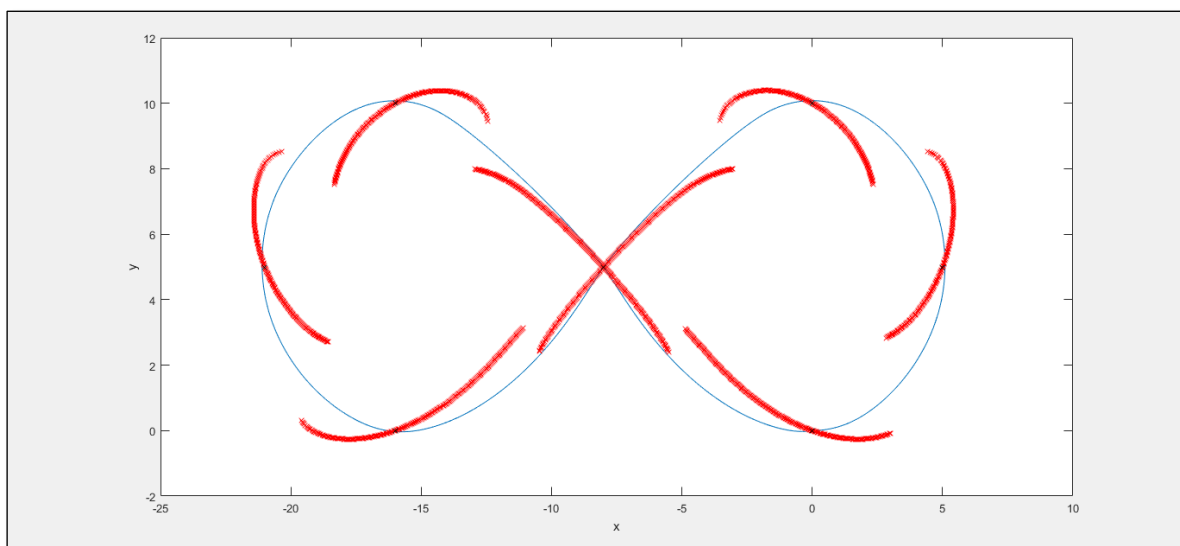


Figura 5.41: Gráfica y vs x para la trayectoria 2.

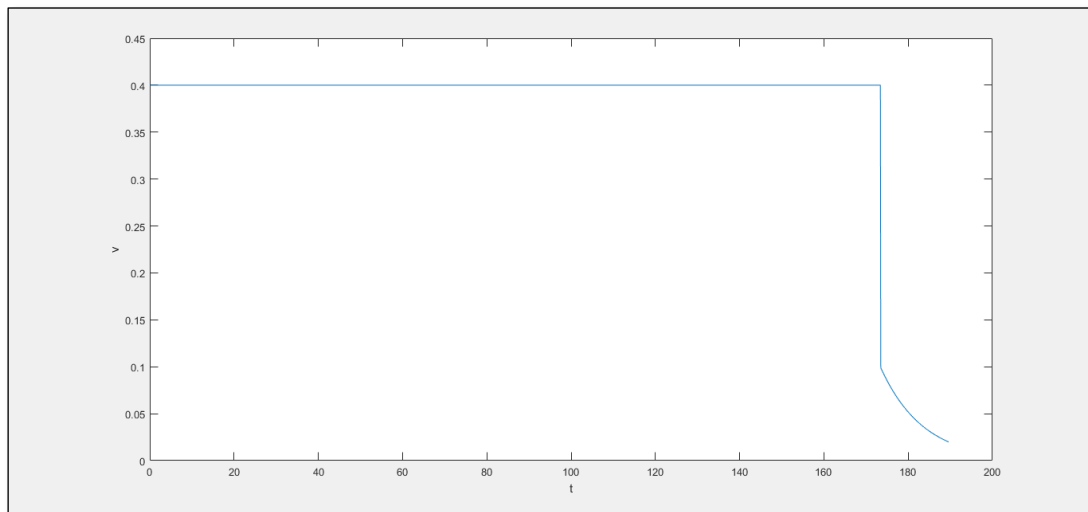


Figura 5.42: $v(t)$ para la trayectoria 2.

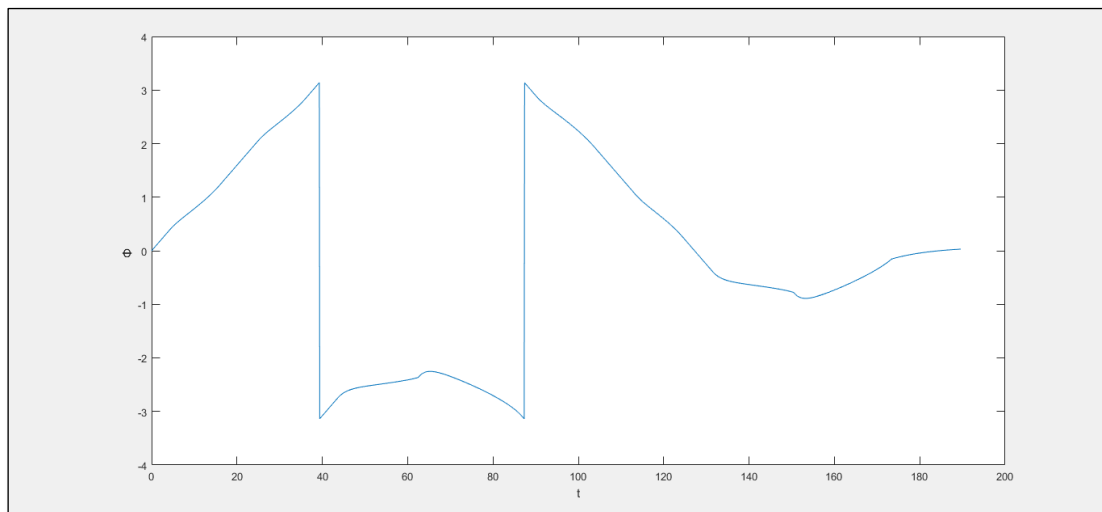


Figura 5.43: $\phi(t)$ para la trayectoria 2.

Se puede apreciar en todas las imágenes, que el control utilizado tuvo éxito también en esta nueva trayectoria.

No se profundiza sobre las conclusiones de cada figura, ya que para este caso valen las mismas consideraciones hechas para el análisis de la trayectoria anterior.

5.5.3. Implementación de software

Una vez que se verificó el correcto funcionamiento de tracking (modelo del robot + leyes de control), nos dedicamos a crear el software que se implementará. Como es sabido, el robot contará con una NUC, la cual tiene un sistema operativo Linux, y será aquí donde se ejecute el software mencionado. La comunicación entre la NUC y la placa de desarrollo K64F será sincronizada y se hará mediante USB con una frecuencia de 10 Hz, equivalente a un periodo de 100 ms.

A continuación, en la Tabla 5.9, presentamos una lista de las funciones implementadas para realizar el control de trayectoria. Cada una de estas funciones será explicada en mayor detalle.

Nombre	Tipo	Descripción
PI	Constante	Número π .
L	Constante	Largo entre 2 ejes del robot en mts.
BETA	Constante	Parámetro de control.
H	Constante	Parámetro de control.
GAMMA	Constante	Parámetro de control.
ESTADO_RPMPC	Constante	Estado de RPM y PC.
MARCHA_ADELANTE	Constante	El robot marcha hacia adelante.
MARCHA_ATRAS	Constante	El robot marcha hacia atrás.
FRENO	Constante	Señal del freno activa.
NO_FRENO	Constante	Señal del freno inactiva.
OPEN_PAR	Constante	Caracter '('.
CLOSE_PAR	Constante	Caracter ')'.
DOS_PUNTOS	Constante	Caracter ':'.
COMA	Constante	Caracter ','.
BARRA_R	Constante	Caracter '/'.
BARRA_N	Constante	Caracter '/'.
PERIM_RUEDA	Constante	Perímetro de la rueda en mts.
ANGULO_MAX	Constante	Ángulo de la dirección máximo ψ en grados.
LIMITE	Constante	Limitación de ángulo de la dirección ψ en grados.
robot	Estructura	Contiene la posición en x, y , la dirección ψ , la vel. angular ω , la velocidad del robot v y la velocidad descompuesta en x, y .
manip	Estructura	Contiene la velocidad v y la dirección ψ que se les darán como órdenes al robot.
datos_rec	Estructura	Contiene datos de interés (dirección, rpm, sentido, flag) de la trama recibida. Flag indica si la trama recibida fue correcta.
punto_ref	Estructura	Contiene las coordenadas x, y ; y la orientación deseadas del punto objetivo.
robot calc_robot(float, float, int, float, float, float, float, float, long)	Función	Es la encargada de calcular todas las variables de interés del robot (ver estructura).
manip calc_control(float, float, float, float, float, float, float, float)	Función	Es la que aplica las leyes de control, y devuelve las variables manipuladas (vel. y dir.).
float integral(float, float)	Función	Es la que se encarga de integrar una determinada variable.
char* gen_trama(float, float, char*)	Función	Tiene como objetivo generar un string que contenga la trama a ser enviada.
char* cx_rx(int)	Función	Es la que lee lo que hay en el puerto de comunicación.
void cx_tx(char*, int)	Función	Envía la trama generada por la función gen_trama .

<code>datos_rec</code> <code>descifra_trama(char*)</code>	Función	Esta función descifra los datos de interés que contiene una trama leída.
<code>void archivo(char*, robot, FILE*, datos_rec)</code>	Función	Es la que se encarga de guardar los datos de interés del robot en un archivo.

Tabla 5.9: Bibliotecas, defines, estructuras y funciones incluidos en el software de tracking.

A continuación, se da una explicación detallada de las principales funciones implementadas.

5.5.3.1. Función main e inicialización

Antes de la ejecución del bloque **main**, se deben definir todas las funciones y tipos de datos a utilizar, así como también incluir las bibliotecas que se utilizarán. La Figura 5.44 representa un diagrama de flujo de estos pasos.

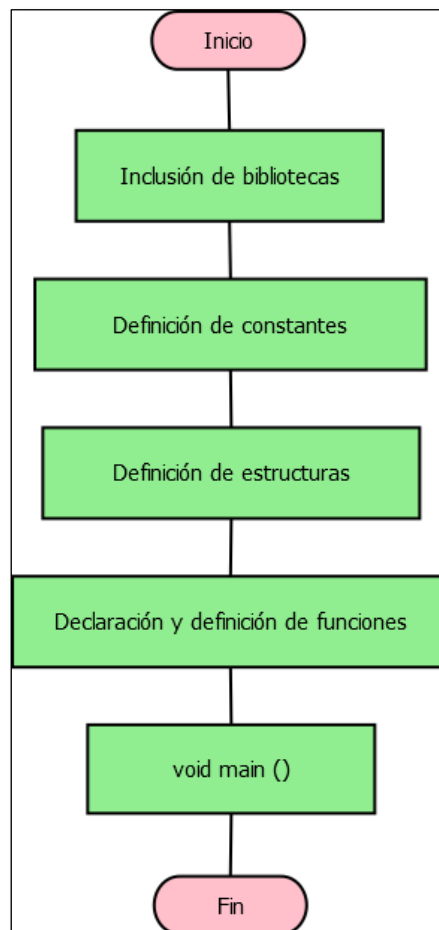


Figura 5.44: Diagrama de flujo principal.

Una vez culminado este proceso de inicialización, comienza la ejecución del bloque **main**. En la Figura 5.45 se muestra un diagrama de flujo que representa el comportamiento del mismo. En este procedimiento se encuentra la definición de todas las variables a utilizar, archivos, comunicación, funciones propias para el control y necesarias para un correcto funcionamiento.

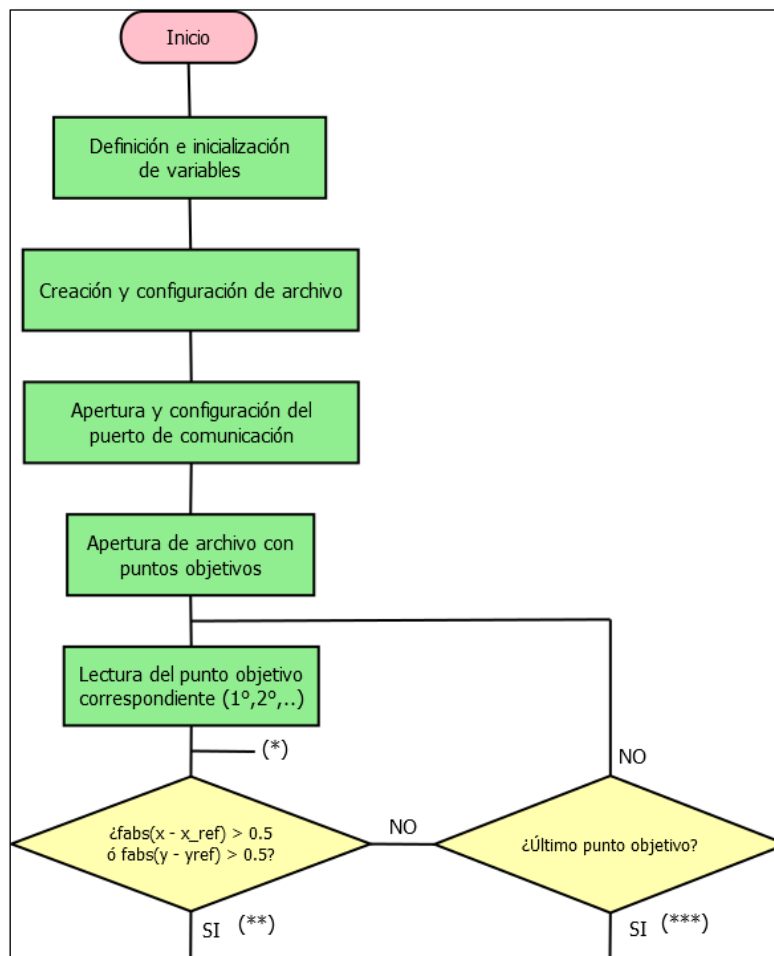


Figura 5.45: Diagrama de flujo de la función main (Parte 1).

Este diagrama muestra cómo está compuesta la función **main** del programa, la cual es, valga la redundancia, la función principal del código. Esta es una función de tipo **void**, lo que indica que no devuelve ningún valor, es decir, es lo que se conoce como procedimiento. En ella se definen variables necesarias para el desarrollo de software, seguido, se crea y se configura un archivo de tipo escritura para guardar datos de interés como la posición del robot, velocidad, dirección, ángulo de orientación, etc. Luego, se prepara el puerto de comunicación “ttyACM0” de la NUC y se lo configura para lograr una comunicación adecuada con la placa desarrollo K64F, la velocidad elegida es de 115200 bps.

A continuación, se abre un archivo, que debe contener los puntos objetivos de la trayectoria a realizar. Este archivo debe ser generado por las personas que requieran que el robot realice una determinada trayectoria, y debe ser de la siguiente forma, *coordenada “x;coordenada y;ángulo de orientación deseado”*. A estos puntos se los lee de a uno y se discrimina los diferentes datos del punto en particular gracias a la separación por el símbolo “;”, se envía el punto objetivo al cálculo de seguimiento de trayectoria y, cuando el robot está aproximándose al mismo, se conmuta al punto objetivo siguiente, excepto cuando este punto objetivo es el último de la trayectoria, o sea, el punto final de la misma.

En la primera decisión que se debe tomar (izquierda), se verifica si el robot está cerca del punto objetivo. La tolerancia que le dimos fue de 0.5 m en cada coordenada, así lo máximo que se acercará al punto será 0.5 m, y lo más lejos que estará será 0.707 m. En comparación con el modelado que se realizó en *Simulink* el margen de error fue aumentado, ya que esta distancia era de 0.2 m; esto se hizo solamente para compensar algunos errores que surgen en la práctica, sin ser determinante el margen de error para las distancias que el robot debe recorrer.

Si el robot no está lo suficientemente cerca del punto objetivo, se aplica el control a través de la función **calc_control** para luego generar la trama con la función **gen_trama**. Todo esto se muestra en el diagrama de la Figura 5.45.

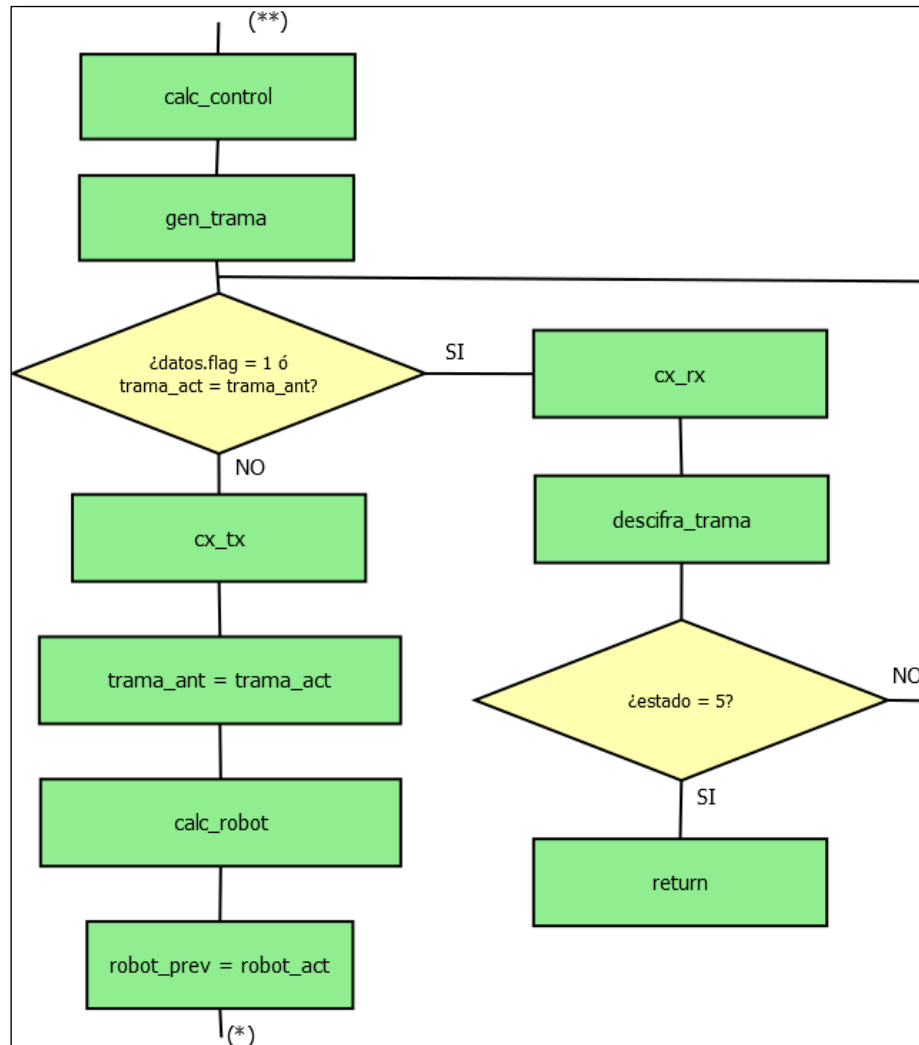


Figura 5.46: Diagrama de flujo de la función main (Parte 2).

El diagrama superior (Figura 5.46), es la continuación del anterior (Figura 5.45). En la decisión de la izquierda, si se produjo un error en la comunicación (*datos.flag* = 1) o si la trama recibida es igual a la anterior, se lee el puerto de comunicación y se descifra la trama con **cx_rx** y **descifra_trama**. Aquí se llega a una nueva toma de decisión, la cual es para brindar seguridad, si el estado de la trama es 5 indica un estado de emergencia (este estado se designa en la placa K64F), por lo que se procede a finalizar la ejecución del código a través de **return**. Si no es 5, vuelve a la toma de decisión comentada anteriormente.

Si no ocurre lo mencionado en esta primera toma de decisión, se procede a enviar la trama generada previamente, a través de **cx_tx**, y luego se le asigna la trama leída a la trama anterior. Esto se debe a que el código en la NUC se ejecuta en menos de 100 ms, entonces se va a leer la misma trama que envía la placa K64F a través de **cx_rx**, pues ésta la actualiza y envía cada 100 ms. Entonces, una manera fácil de saber que la trama leída es una nueva, es guardar la anterior y compararlas. Cabe aclarar, que, si bien el robot puede mantenerse con la misma velocidad y dirección, las tramas que envía no serán nunca iguales, debido a que en ellas se envía una señal con la cantidad de pulsos generados por los sensores de efecto

hall de los motores BLDC. Aunque puede ser que se envíe la misma trama más de una vez, momento en el que el robot está detenido y no genera nuevos pulsos de sensores hall, cuando este se encuentra en movimiento (lo está en todo momento de la solución de tracking) esto no sucede, por lo que no habrá inconvenientes.

Seguimos con la llamada a **calc_robot** y luego guardamos los datos del robot como anteriores, esto nos servirá para el cálculo de integrales, que explicaremos más adelante. Finalmente, volvemos a (*) mostrado en la Figura 5.45.

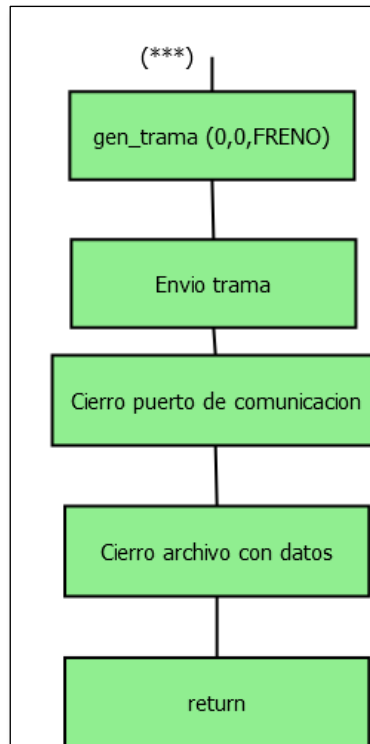


Figura 5.47: Diagrama de flujo de la función main (Parte 3).

Por último, analizamos (***) de la Figura 5.47, rama que nos lleva a la conclusión del programa.

Nos encontraremos en esta parte del programa, si el punto objetivo es el último. En este instante, se genera una trama con velocidad y dirección nulas, y con el freno activo, para luego ser enviada. Hay que tener en cuenta la inercia, la cual mantiene al robot en movimiento por unos instantes, lo que podría acercar o alejar más al robot hacia el punto objetivo, en el primer caso sería beneficioso y en el segundo perjudicial, aunque insignificante, porque la distancia es muy pequeña debido a que se aplica el freno y que entre esta trama y la anterior pasan solo 100 ms.

La misma se completa cerrando el puerto de comunicación, el archivo de escritura con los datos de interés y finalmente **return**.

5.5.3.2. Función calc_control

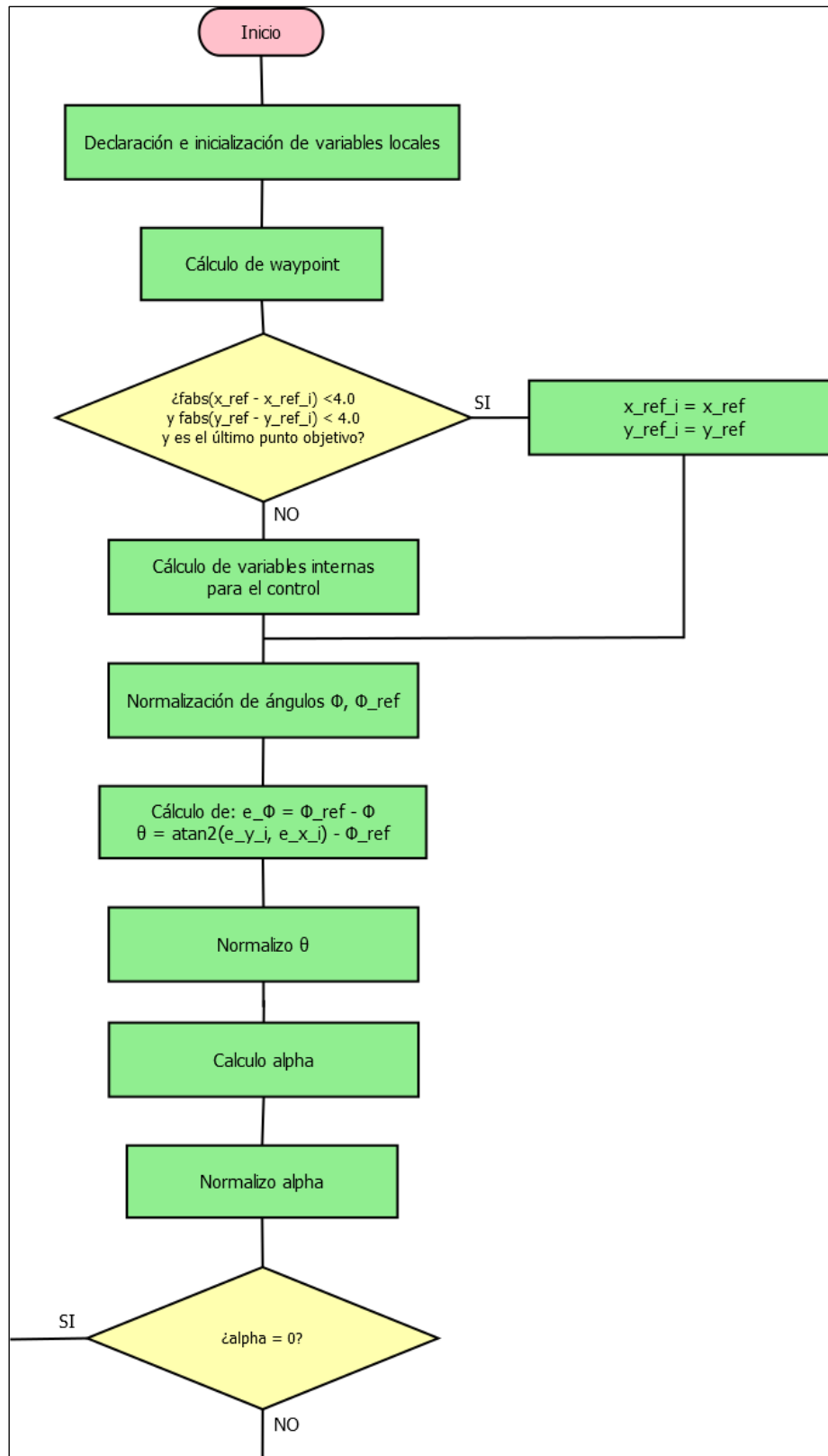


Figura 5.48: Diagrama de flujo de la función calc_control (Parte 1).

Esta función se dedica a calcular la velocidad y la dirección que debe tener el robot a través de las leyes de control.

Como se observa, se calcula el *waypoint*, se lo compara con el punto objetivo y se verifica si es el último o no, esto es para detectar si se debe descartar el *waypoint* o no, si se encuentra a menos de 4 metros de distancia, se toma el punto objetivo final y si no, se siguen calculando *waypoints*. Luego, se calculan variables que necesita el control con el *waypoint* o con el punto objetivo, según corresponda. La normalización de ángulos es comprender a los mismos entre $(-\pi, \pi]$ para el correcto funcionamiento del algoritmo de control.

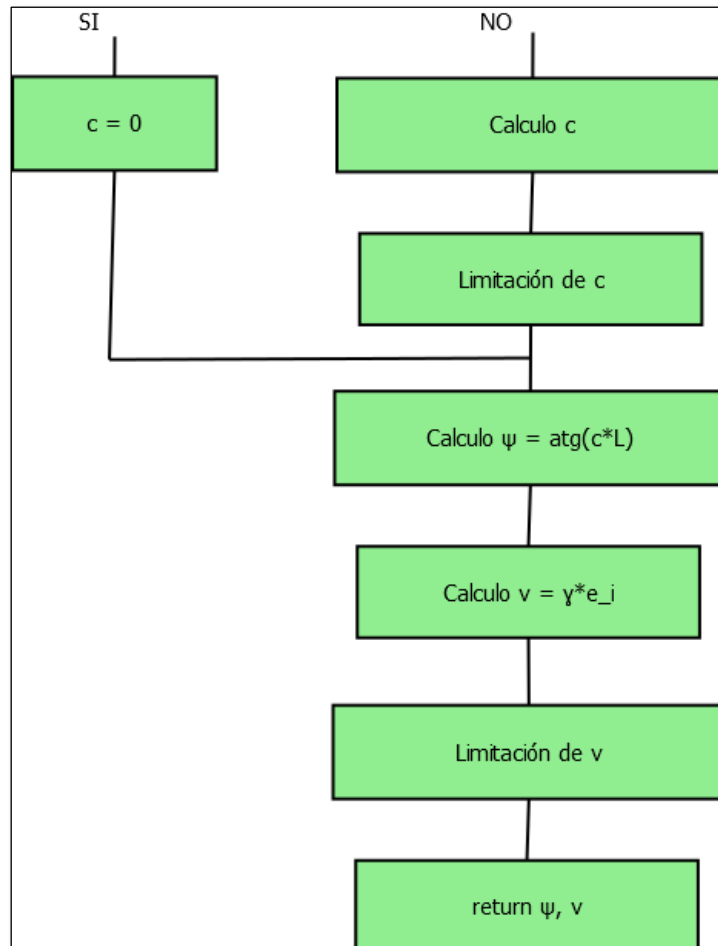


Figura 5.49: Diagrama de flujo de la función *calc_control* (Parte 2).

A continuación, seguimos con la toma de decisión donde nos preguntamos si es $\alpha = 0$, hay que recordar que este ángulo es necesario para la implementación de la solución al problema de tracking (Sección 3.2.3). Si lo es, decidimos que la curvatura sea 0, esto se debe a que hay un término en el cálculo de la curvatura c que es inversamente proporcional a α , por lo que daría ∞ en lugar de 0 que es lo correcto. Por lo que concluimos en que la ley de control de c , tiene una restricción para $\alpha = 0$. Si α no es 0, se calcula c bajo la ley de control y se limita a los valores máximos que tiene la dirección del robot. Luego se calculan velocidad y dirección, a esta primera la vamos a limitar por seguridad, aunque sabemos que nunca supera los 0.4 m/s. Se devuelven la velocidad y la dirección calculadas por las leyes de control.

5.5.3.3. Función calc_robot

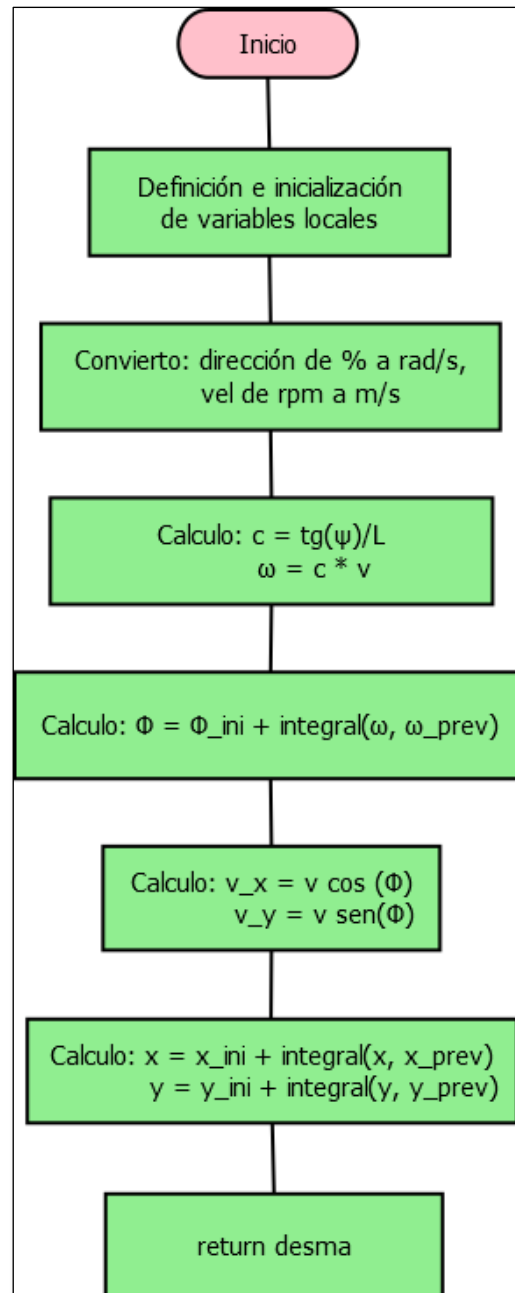


Figura 5.50: Diagrama de flujo de la función calc_robot.

Esta función es la encargada de realizar el cálculo de odometría del robot. Lo más importante del cálculo mencionado, es el que se hace aplicando la función **integral** entre un intervalo de 100 ms, y luego se le suma el valor que tenía previamente (valor inicial).

Finalmente, se devuelven estos datos, a través de “**return desma**”, *desma* hace alusión al robot pulverizador y es una variable de tipo robot que contiene los datos de éste.

5.5.3.4. Función integral

La función integral, es utilizada para calcular los datos de interés más relevantes del robot. La posición en coordenadas x e y , y la orientación del mismo.

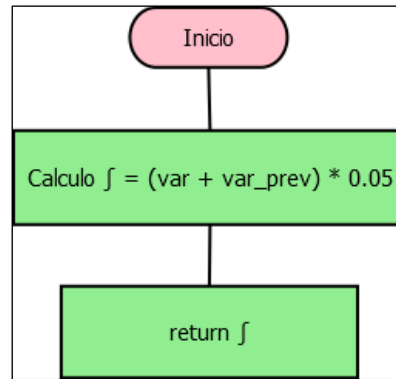


Figura 5.51: Diagrama de flujo de la función integral.

A continuación, además del diagrama, mostramos la parte de software que hace referencia a esta función:

```

float integral (float var, float var_prev) {
    float integ = (var_prev + var) * 0.05;
    return integ;
}
  
```

Esta función es muy sencilla, realiza el cálculo de una integral discreta, aplicando como integración numérica un método similar al del punto medio. Así, dado 2 valores consecutivos de una variable (muestras) se realiza un promedio de estos y se los multiplica por el tiempo transcurrido, que será el período de comunicación (100 ms), obteniendo como resultado el área debajo de la curva que representa la integral de una función. En el software o diagrama de bloques, multiplicar por 100 ms (0,1 s) y dividir la suma de los datos por 2, es equivalente a multiplicar esta suma por 0,05.

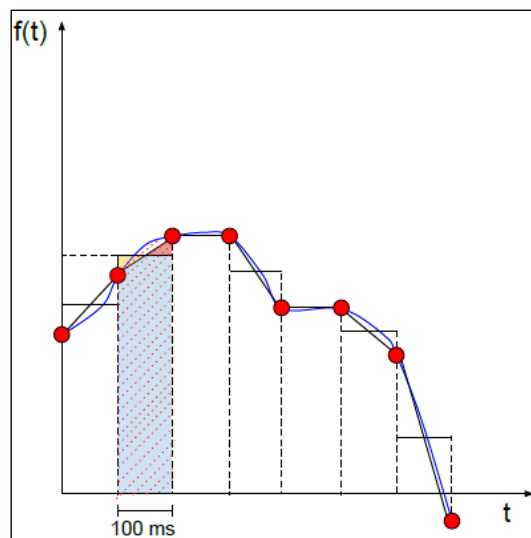


Figura 5.52: Gráfica ejemplo del método de integración discreto utilizado.

Observando la Figura 5.52, supongamos que la curva de línea azul es la función a integrar, los datos obtenidos de la comunicación son los marcados con puntos rojos. En el periodo marcado con 100 ms, analizamos lo realizado.

El error que se comete es la suposición de que la función entre 2 muestras evoluciona linealmente, hecho que no necesariamente es así. El área pintada de color celeste + amarillo, es el área que se calcula, la cual es equivalente, al área de color celeste + rosa, que es el área debajo de la curva linealizada.

Así, el error cometido es igual al área que está entre la curva de la función real, y la línea que une las 2 muestras. Se puede apreciar que, en este caso, el error será negativo, pues faltó tener en cuenta a la parte de área mencionada. En otros casos, como en el del primer período, vemos que la curva que representa a la función está por debajo de la línea que une las primeras 2 muestras, dándonos la pauta de que el error será positivo.

5.5.3.5. Función gen_trama

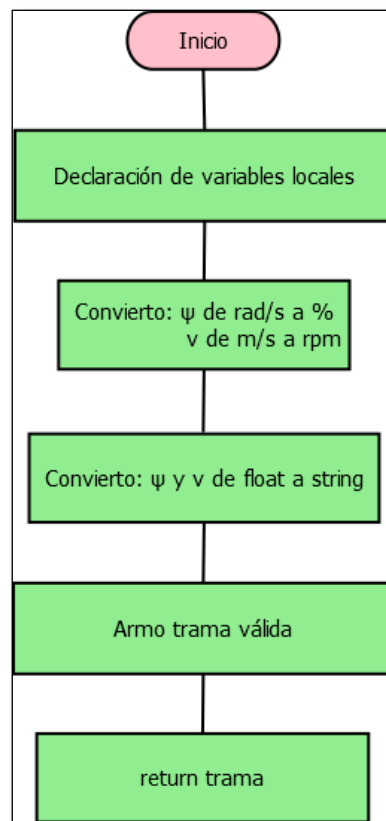


Figura 5.53: Diagrama de flujo de la función gen_trama.

En esta función nos encargamos de generar la trama que será enviada al microcontrolador. Se reciben como parámetros la velocidad y dirección calculadas bajo las leyes de control (función **calc_control**), luego, se hacen las correspondientes conversiones para generar una trama válida y legible. Como esta trama es generada realizando una concatenación de caracteres y datos, a estos últimos se los debe transformar en un tipo de dato string, por lo que se realiza la conversión de tipo de datos de float a string. Esto se basa en poder armar un tipo de dato string, que será la trama, con el formato de trama válida impuesto en la Sección 5.4.2. Finalmente, se devuelve la trama válida con **“return trama”**.

5.5.3.6. Función archivo

Esta función es la encargada de escribir los datos de interés en el archivo de escritura, por lo que, como se realizó con la trama a generar, hay que convertir estos datos en unos de tipo *string*, para luego concatenarlos y armar un *string* de mayor tamaño, el cual tendrá los datos tabulados para mayor comprensión visual. Los datos que se almacenarán en el archivo son, la posición en x e y, la velocidad, el ángulo de giro de la dirección del robot y la orientación del mismo. Finalmente escribimos este *string* en el archivo mencionado y volvemos con **return**. Este archivo generado es un documento de texto (.txt). En la Figura 5.54, observamos el diagrama de flujo de esta función que forma parte del software de tracking.

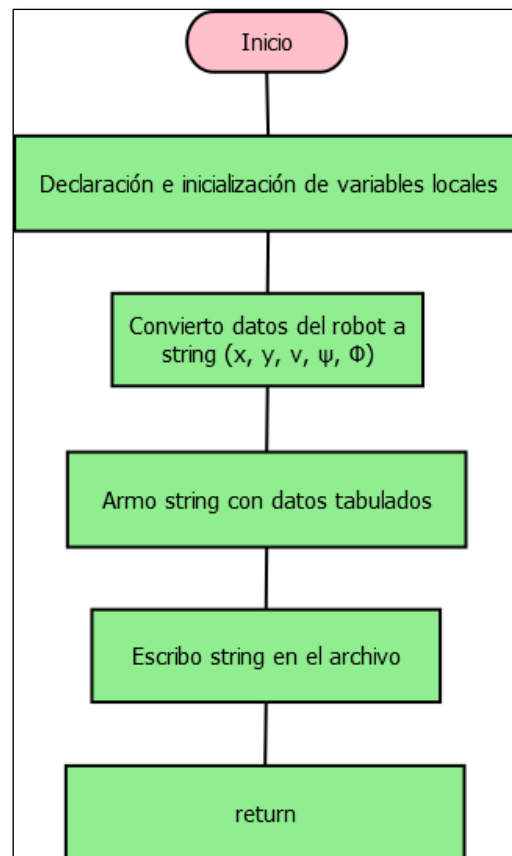


Figura 5.54: Diagrama de flujo de la función archivo.

5.5.3.7. Función cx_rx

Esta función es la encargada de realizar la lectura de la trama que nos envía el microcontrolador con los datos que tiene el robot. La lectura se hace carácter a carácter. Vemos que en un momento hay una toma de decisión, esto es si el carácter recibido es '*n*', el cual indica el fin de trama. Si no lo es, se almacena el carácter leído en un arreglo y se incrementa el índice del mismo para una nueva lectura. En cambio, si es '*n*', lo almaceno en el arreglo que contendrá los caracteres de la trama completa, y lo devuelvo con "**return array**". Esta secuencia se puede ver en el diagrama de flujos de la Figura 5.55.

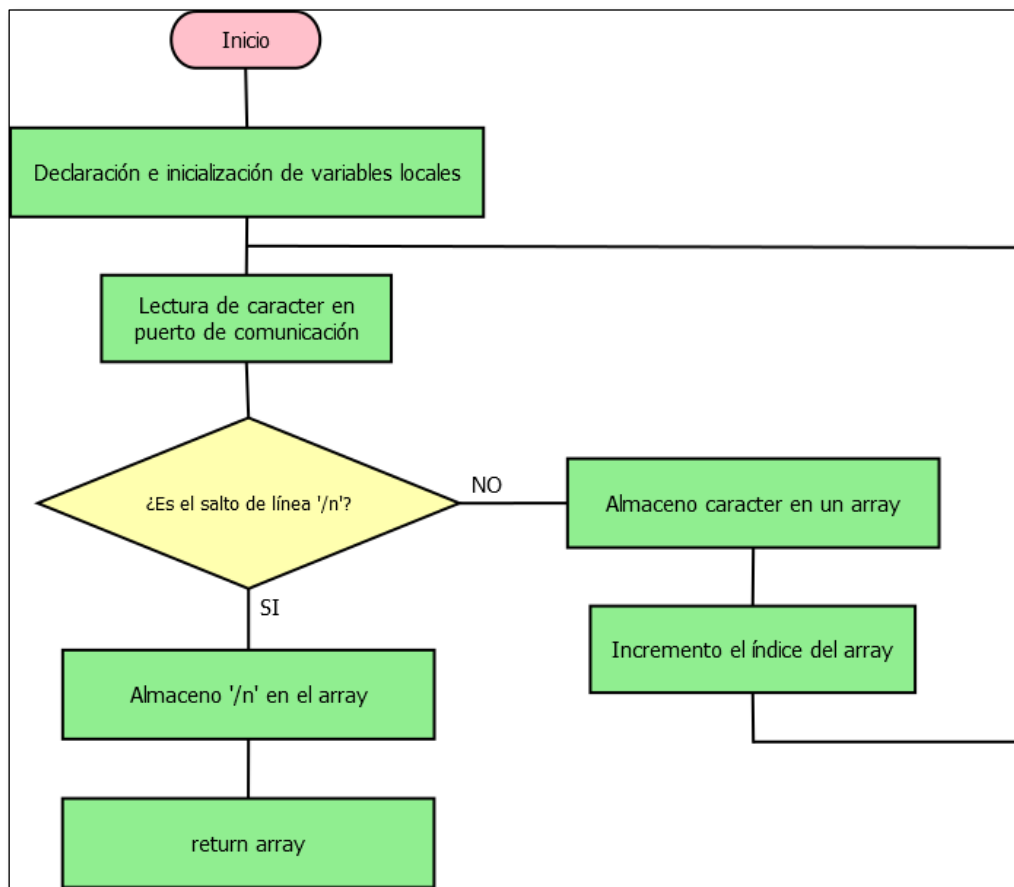


Figura 5.55: Diagrama de flujo de la función *cx_rx*.

5.5.3.8. Función *cx_tx*

Esta función se encarga de enviar, a través del puerto de comunicación, la trama generada por la función *gen_trama*. Luego, vuelve con **return**.

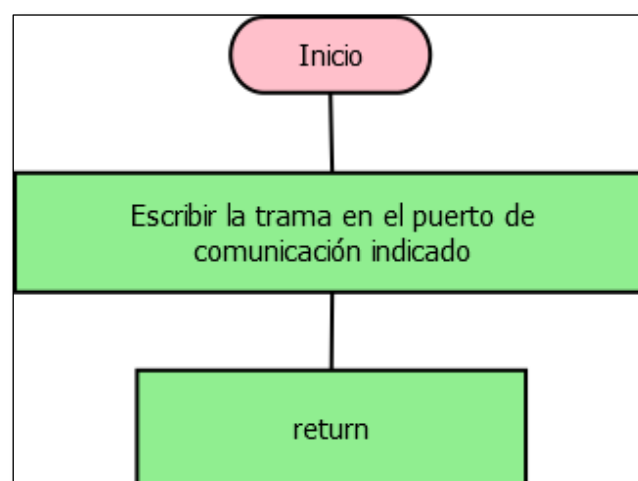


Figura 5.56: Diagrama de flujo de la función *cx_tx*.

5.5.3.9. Función descifra_trama

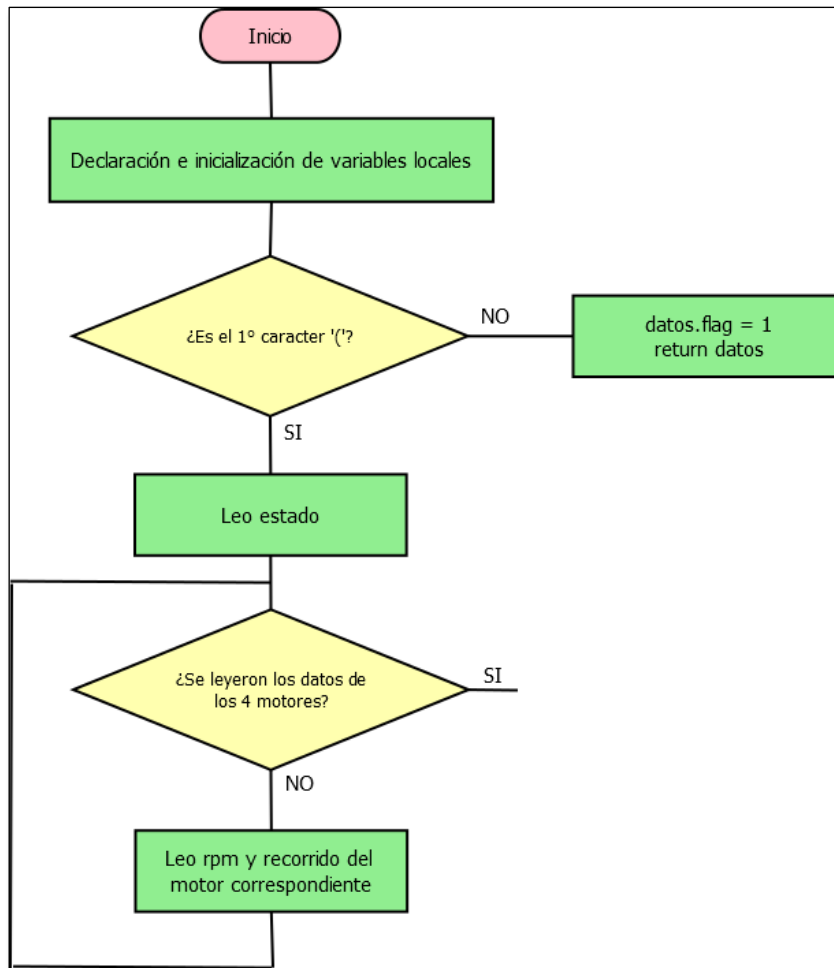


Figura 5.57: Diagrama de flujo de la función descifra_trama (Parte 1).

Esta función lee la trama recibida por **cx_rx**. La toma de decisión es si el primer caracter es '(', si no lo es, se levanta la bandera *datos.flag* y se devuelve datos con **“return datos”**. Si lo es, se sigue descifrando. Lo siguiente de interés es el estado en el que se encuentra y, a continuación, se leen los datos de los 4 motores (*nro. motor, velocidad, recorrido, corriente*), en este caso sólo nos interesan la velocidad y el recorrido realizado. Así, si se leyeron los datos correspondientes a los 4 motores, pasamos a realizar las siguientes tareas, expuestas en la Figura 5.58.

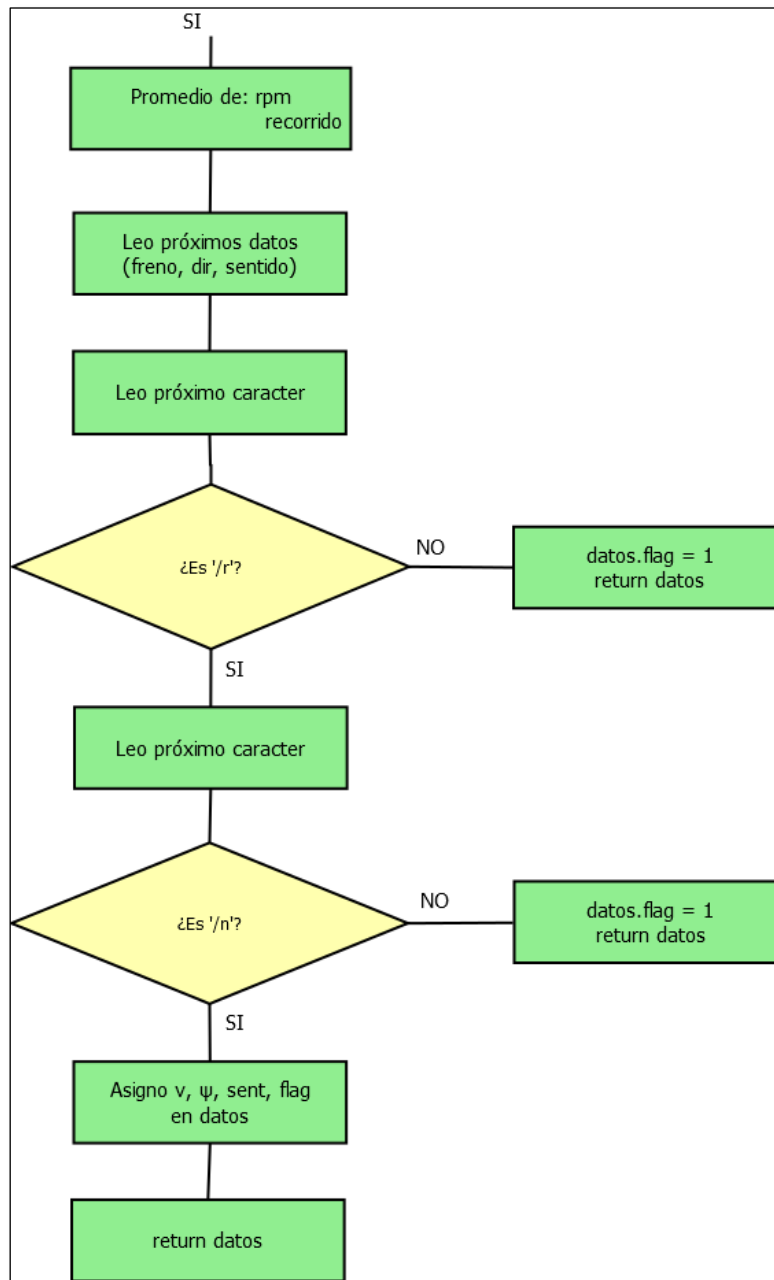


Figura 5.58: Diagrama de flujo de la función descifra_trama (Parte 2).

Entonces, una vez que se leyeron los datos de los 4 motores, se hace un promedio para tener la velocidad y recorrido del robot, seguido de la lectura de la dirección, sentido y freno. Finalmente se verifica que la trama finalice como corresponde, leyendo los últimos caracteres.

Se lee el caracter siguiente y se verifica si es *'r'*, si no lo es, se levanta la bandera *datos.flag* y se devuelve con ***"return datos"***, si es *'r'* se lee el próximo. Nuevamente se repite lo hecho anteriormente, con el caracter *'n'*, aunque esta vez, si es correcto, se terminó de leer una trama válida, por lo que se procede a asignar la velocidad, la dirección, y el sentido leídos; y la bandera *flag* se baja. Finalmente, devolvemos con ***"return datos"*** lo mencionado.

6. Pruebas de Campo

En esta sección se abordarán temas referidos a la operación del robot pulverizador en situaciones reales, es necesario chequear el correcto funcionamiento del controlador de tracción para luego ensayar y validar el algoritmo de navegación autónoma.

6.1. Control de tracción

Como primera tarea se monta el vehículo sobre caballetes para evaluar su comportamiento con el principal objetivo de validar el controlador PI, luego, una vez asegurado el correcto funcionamiento del sistema, el mismo se lleva a un terreno de pruebas para evaluarlo en condiciones reales.

6.1.1. Controlador

En base a lo mencionado en la sección 5.2.8, ensayamos el sistema a lazo cerrado con distintas referencias de velocidad y pudimos ver que los cuatro motores se comportan de la manera esperada (Figura 6.1).

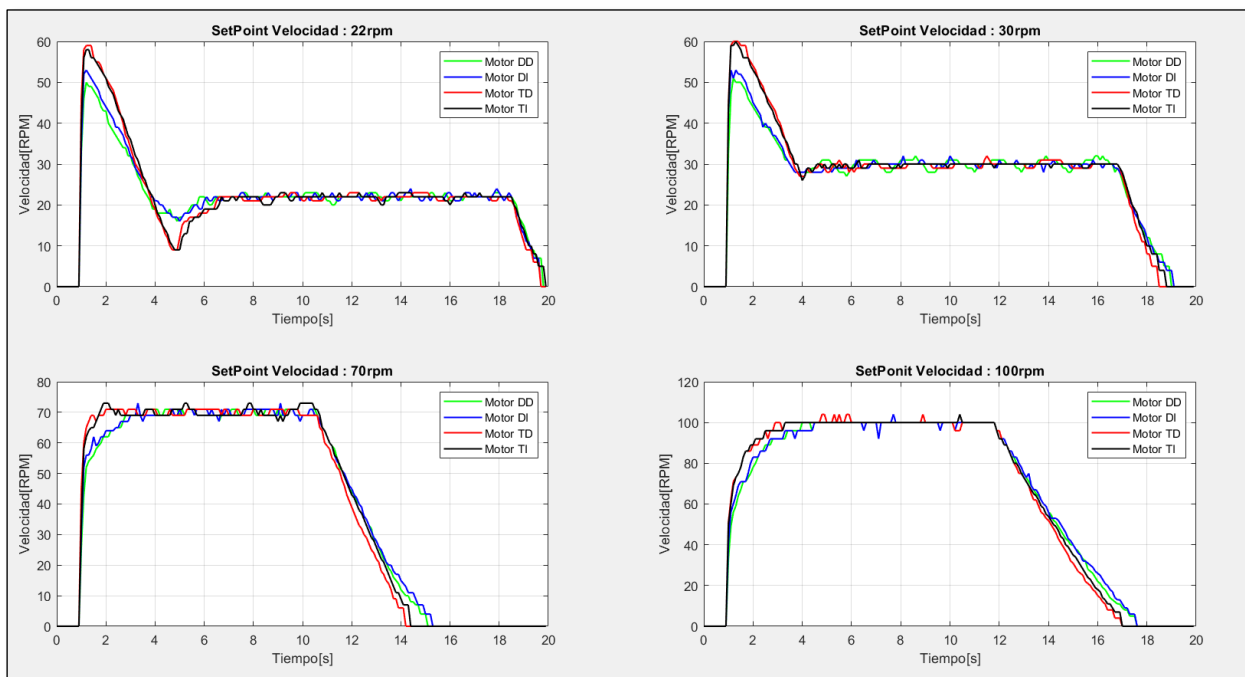


Figura 6.1: Respuesta de los motores frente a diferentes Set-Point de velocidad.

Esta prueba se realizó con el robot montado sobre caballetes impidiendo el apoyo de las ruedas en el piso por lo que puede considerarse como la respuesta de velocidad de los motores trabajando sin carga dado que solo tiene el agregado de inercia que provoca la rueda montada al motor.

Si bien los motores traseros tienen una respuesta más “brusca”, el regulador funciona correctamente logrando establecer la velocidad de referencia en los cuatro motores en un tiempo razonable.

El sobrevalor observado en las gráficas se debe a que, para mitigar el problema de falta de potencia en el arranque del vehículo, se agregó una condición en el firmware que referencia 25% de la máxima tensión en los motores cuando la velocidad del vehículo es menor que el 25% del setpoint establecido. Así, en el arranque siempre tendremos 25% de la máxima tensión aplicable a los motores.

6.1.2. Actuación de freno

Como vimos a lo largo del proyecto, los drivers cuentan con sistema de freno regenerativo, en la sección 5.4.3.3 hemos mencionado la manera de actuarlo, siendo el robot comandado por control remoto o mediante la PC. Aquí (Figura 6.2 y Figura 6.3), queremos mostrar la respuesta de los motores a lazo abierto y a lazo cerrado respectivamente, comparando la actuación del freno regenerativo contra el frenado por inercia de los motores.

En el primer caso, con el motor a lazo abierto, se le aplicó una tensión de manillar cercana a los 2V, allí podemos ver que la velocidad del motor se redujo más del 50% en alrededor de 0,3 segundos del valor inicial al aplicar el freno; mientras que al frenar el motor solo quitando el suministro de tensión, tuvo una demora de alrededor de 3 segundos para la misma reducción de velocidad. Luego, con la actuación del freno, logramos la detención completa del motor en aproximadamente 2,5 segundos contra los 6 segundos que demora sin la actuación del freno.

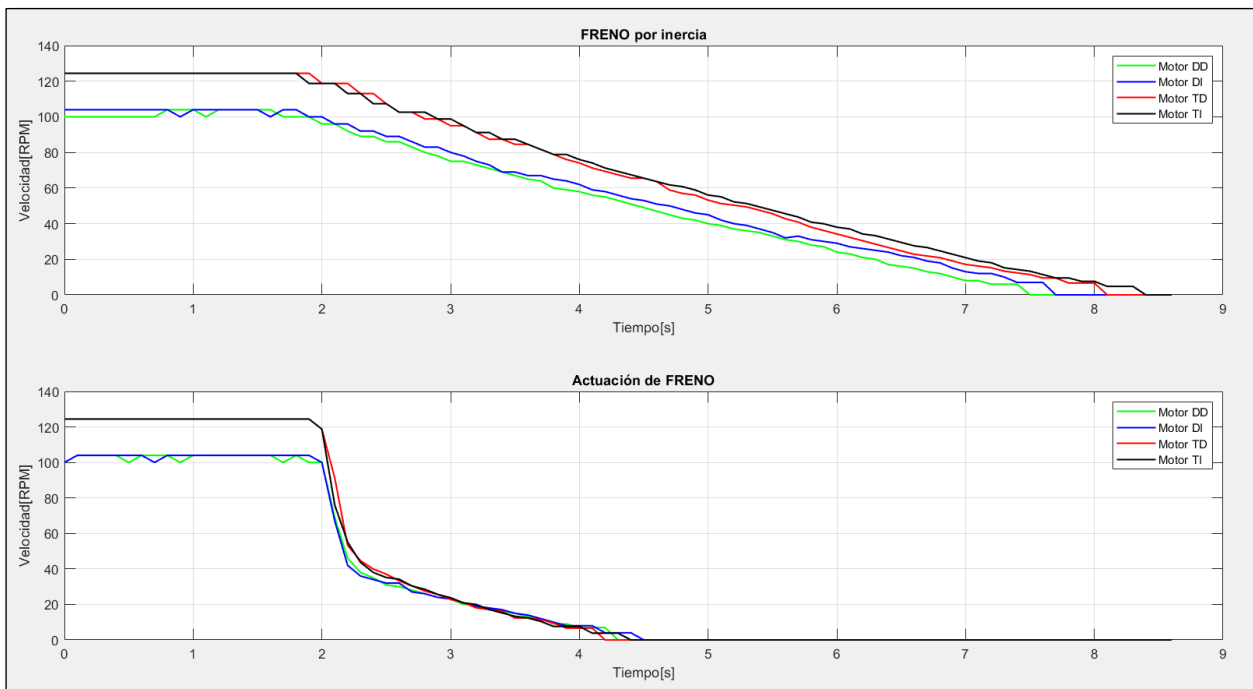


Figura 6.2: Comparación de actuación de freno a lazo abierto.

En segundo lugar, con el sistema a lazo cerrado, seteamos la velocidad de giro de los motores a 80RPM y los resultados obtenidos son similares a los vistos para el sistema a lazo abierto.

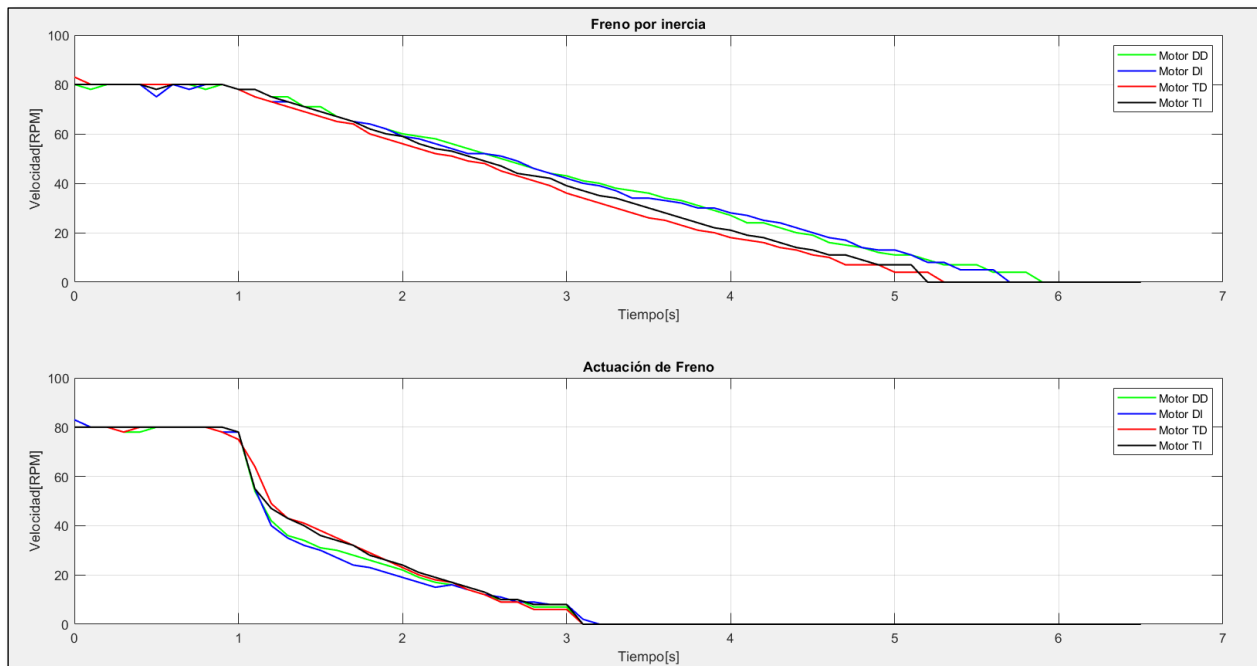


Figura 6.3: Comparación de actuación de freno a lazo cerrado.

6.1.3. Corriente consumida

Ya hemos mencionado que el estudio de la corriente excede los límites de este proyecto, pero nos pareció interesante mostrar el pico de corriente inversa generado en el momento en que se actúa el freno regenerativo, este pico de corriente es absorbido por una de las baterías montada en el vehículo capaz de soportar este pico de corriente inversa. Las características, dimensionado y protecciones de este sistema de recuperación y almacenamiento de energía proveniente del sistema de freno regenerativo no forma parte del presente desarrollo.

A continuación, podemos observar el comportamiento de la corriente en dos situaciones similares en las que se produce la detención de los motores. Primeramente, en la Figura 6.4, lo anterior sucede por el solo hecho del corte del suministro de energía haciendo que el motor reduzca su velocidad por inercia, y en el segundo lugar, Figura 6.5, se hace uso del freno regenerativo donde puede verse claramente el pico de corriente inversa generado.

Estos ensayos se realizaron con el robot montado sobre caballetes de modo de que los motores no tengan contacto con el piso. Es por esto que se observan bajos valores de corriente.

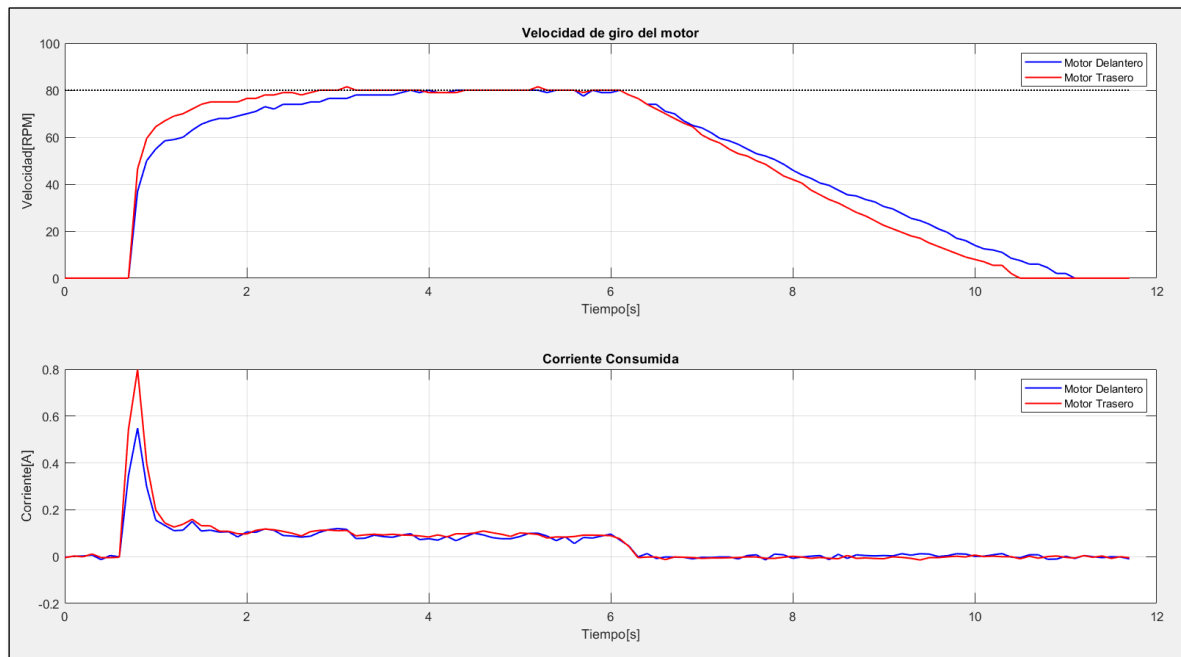


Figura 6.4: Respuesta de los motores sin actuar el freno.

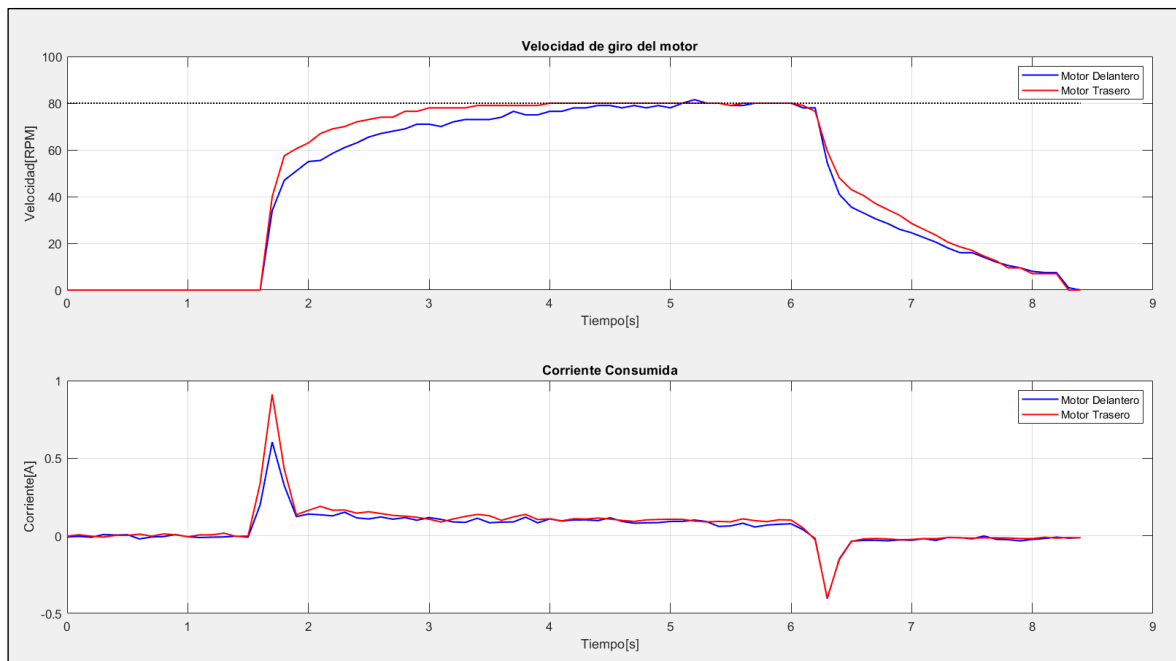


Figura 6.5: Respuesta de los motores aplicando el freno.

6.1.4. Pruebas en terreno

Ensayamos la respuesta del vehículo en el terreno real. El primer ensayo fue realizado en un terreno regular, exento de obstáculos o pendientes y fijamos una velocidad de avance de 2.5 km/h, comparando la respuesta en vacío respecto a la obtenida en el terreno, se puede notar a simple vista que la velocidad del vehículo converge al mismo valor en ambos casos, con la diferencia de que la segunda posee un tiempo de respuesta mayor y un sobrevalor menor debido al esfuerzo adicional provocado por la inercia total del robot.

Cabe destacar que, por cuestiones de seguridad, dado para las pruebas contamos con espacio físico reducido, optamos por realizarlas a una velocidad relativamente baja (2.5km/h), siendo la velocidad normal de trabajo cercana a los 5km/h.

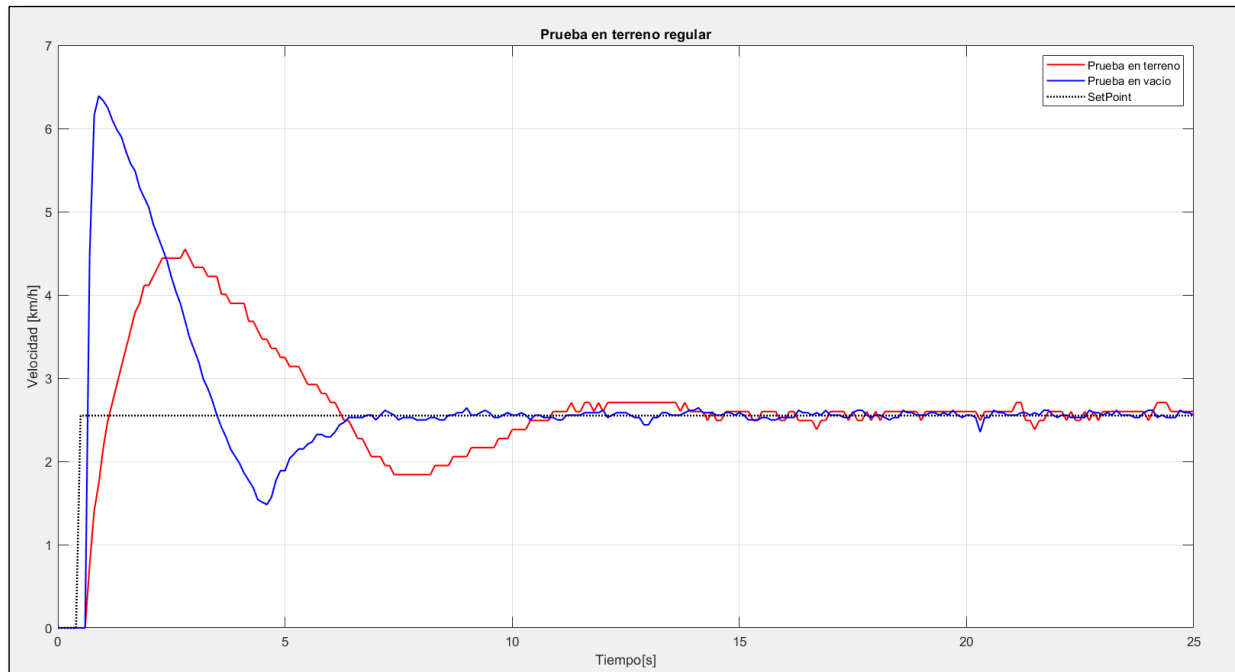


Figura 6.6: Respuesta de los motores en terreno regular.

Luego hicimos una prueba en un terreno irregular, referenciando una velocidad de avance de 30 RPM (aproximadamente 3.1 km/h). Estando el vehículo en velocidad de régimen se colocó un obstáculo que provocó disminución de la velocidad de avance, pero, como bien puede observarse en la Figura 6.7, el regulador pudo actuar de manera satisfactoria y lograr mantener la velocidad seteada.

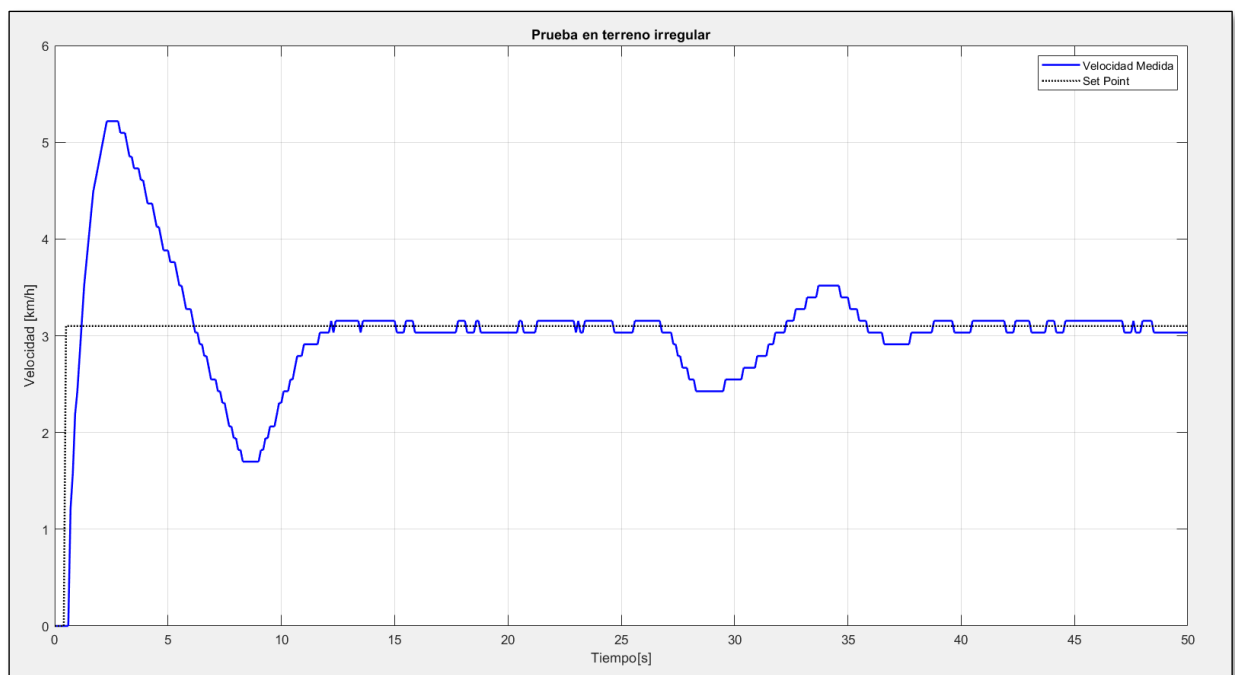


Figura 6.7: Respuesta de los motores en un terreno irregular.

6.2. Ensayos de navegación autónoma

Luego de la programación del algoritmo de control de trayectoria encargado de generar las referencias para el control de dirección y velocidad, y junto a sus respectivas pruebas en un entorno seguro, se procedió a realizar pruebas más realistas, es decir, se llevó al prototipo a un espacio abierto y se le fijaron trayectorias de referencia para corroborar que cumplía con su objetivo.

Ya hemos visto las complejidades que trae consigo realizar el cálculo de la ubicación del robot por odometría. Debido a esto, nos propusimos realizar diferentes trayectorias con el robot, y así conocer la magnitud del error cometido.

Para conocer con exactitud la posición del robot se procedió a utilizar un GPS (Global Positioning System) montado sobre el mismo. Entonces, calculamos con el GPS la posición del robot cada 100 mseg y almacenamos esos datos para compararlos con los datos recolectados por odometría.

Si bien se comentó que el GPS nos brinda mayor precisión, dentro de las pruebas de campo realizadas mayormente obtuvimos datos con la precisión conocida como Float, este tipo de precisión nos brinda datos que tienen un error que ronda los 2 mts por lo que una mala lectura del mismo influye en la creencia de que el robot está realizando una trayectoria indebida. De todas formas, comentaremos la presencia de esta situación con detalle en las distancias pruebas cuando sea necesario.

A continuación, se mencionan las pruebas efectuadas.

Prueba 1 (Circunferencias)

Como primera trayectoria optamos por realizar un mismo círculo en 2 oportunidades. Los datos de referencia se muestran en la Tabla 6.1.

Nro.	x	y	ϕ
Inicio	0	0	0
1	5	5	$\pi/2$
2	0	10	π
3	-5	5	$-\pi/2$
4	0	0	0
5	5	5	$\pi/2$
6	0	10	π
7	-5	5	$-\pi/2$
8	0	0	0

Tabla 6.1: Puntos objetivo de la prueba 1.

En la Figura 6.8, se muestran los círculos realizados por el robot (línea punteada naranja) y los datos de odometría obtenidos por el software que resuelve el control de tracking (línea continua azul).

Como resultado se nota que el recorrido realizado por el robot es correcto, teniendo en cuenta que puede haber errores de posición debido al error introducido por GPS ya mencionado.

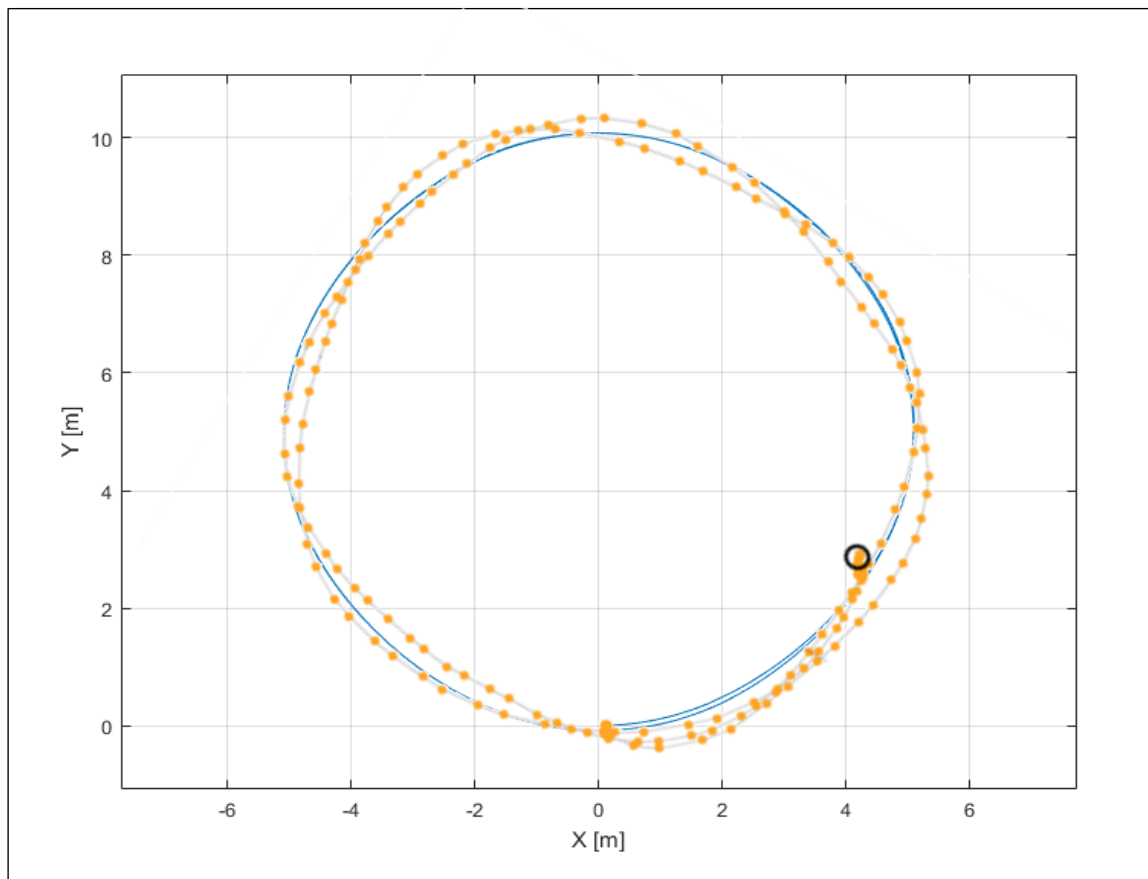


Figura 6.8: Gráfica y vs x de la prueba 1.

El robot comienza la trayectoria desde las coordenadas $(0,0)$ y realiza una circunferencia de radio 10 m, 2 veces consecutivas. El mayor error se ve en el final de la trayectoria, donde el robot finaliza en el punto marcado con un círculo negro. En la figura vemos que es aproximadamente de 4 m en la coordenada x y 3,5 m en la coordenada y , sin embargo, en la realidad el error no fue tan notorio. Esta diferencia se debe principalmente al error introducido por GPS. Para justificar esto, nos basamos pura y exclusivamente en nuestra visualización de la posición final del robot durante la prueba, la cual tuvo un error de 1,5 m, y la trayectoria realizada por el robot fue satisfactoria.

Mas allá de que el error no haya sido el mismo que el visualizado en la figura, este existió y nos da un indicio de los problemas al implementar la solución mediante odometría.

Prueba 2 (Recta y curvas)

Como segunda trayectoria elegimos una recta seguida de partes de circunferencias, la cual fue presentada en la Sección 5.5.2. Se detalla en la Tabla 6.2 los puntos objetivos que componen la trayectoria.

Nro.	x	y	ϕ
Inicio	0	0	0
1	15	0	0
2	20	5	$\pi/2$
3	15	10	π
4	10	10	π
5	5	5	$-\pi/2$
6	0	0	$-\pi$

Tabla 6.2: Puntos objetivo de la prueba 2.

Como ya se explicó, el robot recibirá como consigna estos puntos, en orden y a medida que los alcance, conmutará al siguiente automáticamente hasta llegar al final. También, se sabe que estos puntos son elegidos de manera que el robot pueda realizar una trayectoria como la esperada, por lo que un error en la consigna de los mismos o en el orden, provocaría un error en el recorrido que realice el robot posteriormente.

Esta trayectoria, fue considerada relevante para que el robot la realice, ya que contiene rectas, y curvas en ambos sentidos de giro.

Para este caso en particular, en la Figura 6.9, se pueden apreciar varios errores. Los mayores errores fueron introducidos por GPS, los cuales se potencian cuando el robot realiza giros. En la prueba se visualizó una trayectoria muy buena, de hecho, fue la mejor de todas, teniendo un error en la llegada de aproximadamente 0,5 m, el cual no se corresponde con los datos del GPS que nos muestra que es cercano a 4 m.

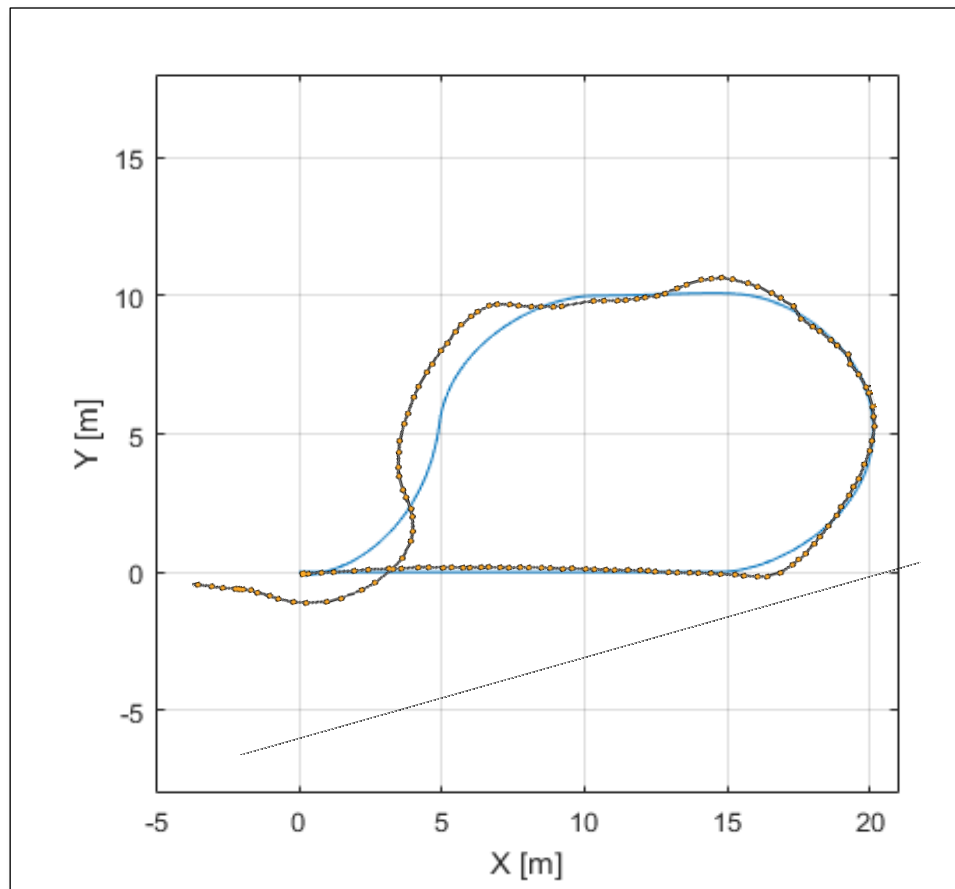


Figura 6.9: Gráfica y vs x de la prueba 2.

Prueba 3 (Símbolo ∞)

Por último, realizamos una trayectoria del símbolo “ ∞ ” repetido tres veces. Los datos de referencia fueron los que se muestran en la Tabla 6.3. Esta trayectoria es la mejor para visualizar con mayor detalle los errores que el seguimiento de trayectoria por odometría comete.

Nro.	x	y	ϕ
Inicio	0	0	0
1	13	0	0
2	15	0	0
3	20	5	$\pi/2$
4	15	10	π
5	13	10	π
6	6,5	5	$-3\pi/4$
7	0	0	$-\pi$
8	-2	0	$-\pi$
9	-7	5	$\pi/2$
10	-2	10	π
11	0	10	π
12	6,5	5	$-\pi/4$

Tabla 6.3: Puntos objetivo de la prueba 3.

Para que la Tabla 6.3 no sea demasiado extensa sólo se muestran los puntos referentes a un “ ∞ ”, al robot se le pasaron estos mismos datos tres veces consecutivas como es debido, respetando el orden.

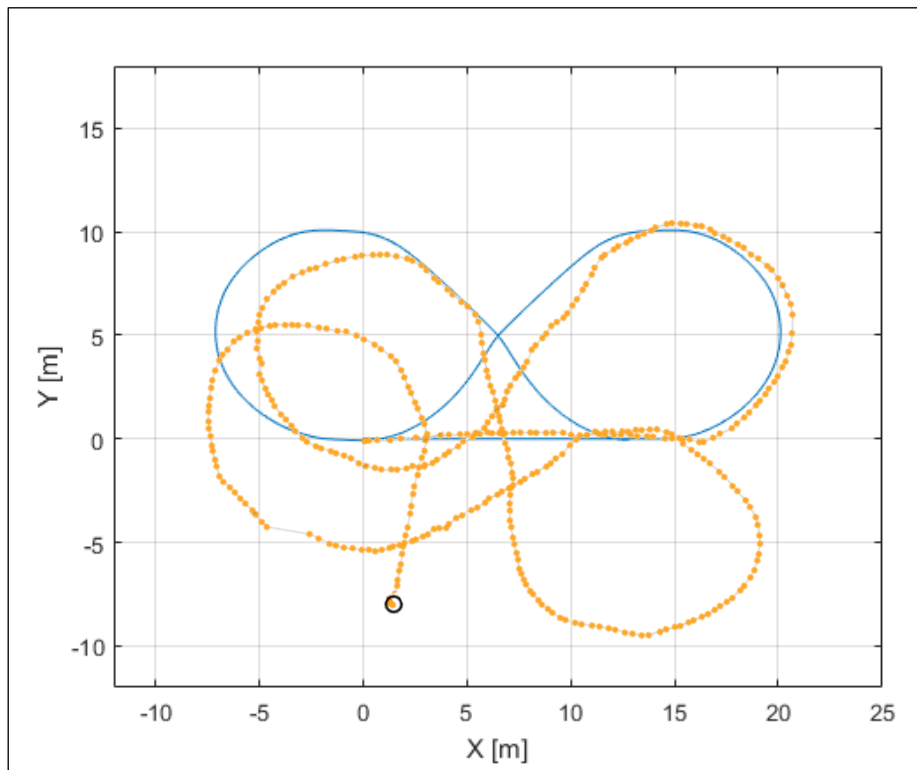


Figura 6.10: Gráfica y vs x de la prueba 3.

El robot comienza su trayectoria desde el origen de coordenadas, y notamos que inicia correctamente, pero al girar comienzan a notarse errores. Estos errores se ven traducidos en una rotación y en un desplazamiento de la trayectoria realizada (puntos amarillos) respecto a la teórica (línea continua azul). Esta incoherencia se visualizó durante la prueba. Aunque los motivos de estos errores podrían ser muchos, destacamos el que se pudo corroborar durante la prueba.

Cuando notamos este error, se decidió a medir el radio de las ruedas del robot, y nos encontramos con que diferían en algunos centímetros de los medidos anteriormente y, además, que eran distintos entre sí. La razón de esto es que cada cubierta puede desinflarse más o menos que las demás a medida que transcurre el tiempo, este error es imposible de corregir con el robot, pues no cuenta con sensores de presión en las cubiertas, por lo que la corroboración queda a cargo de la persona que utilice al mismo.

Las conclusiones principales que podemos sacar de esta prueba son que, principalmente el cálculo de la posición del robot mediante odometría no es viable, ya que el software estima una orientación y posición del robot anticipadas a las que este posee, debido a que el radio de las ruedas se redujo, dando como resultado los problemas de rotación de la trayectoria y desplazamiento de esta, ya mencionados.

6.3. Fuentes de error en la implementación

Cuando se vuelca todo lo que la teoría conlleva en la práctica, es casi imposible que no aparezcan problemas que vienen de la mano de imprevistos que se presentan en este momento. En esta sección comentaremos los problemas con los que nos encontramos al momento de realizar pruebas en el robot.

El principal inconveniente fue la inexactitud en el control de la dirección del robot. Esto, se notó observando que el giro del prototipo hacia la derecha no era el mismo con el giro en el otro sentido.

Una forma sencilla que propusimos para minimizar este error, fue que, con la dirección fijada al 100% en un sentido de giro, hicimos avanzar al robot para que este realice circunferencias, luego de que realice varias, se marcaron sobre el terreno 2 puntos de paso (asegurando que estén sobre una recta que cruce por el centro del robot), para así poder medir el diámetro de la circunferencia realizada. Este proceso se realizó muchas veces sin detener al robot, hasta comprobar que un momento el mismo pasaba por los puntos marcados. Lo mismo, se realizó con sentido de giro contrario, midiendo el diámetro de la circunferencia realizada en ambos sentidos de giro.

Entonces, conociendo el diámetro de las circunferencias, acudimos a modificar la consigna de dirección únicamente para el sentido en el que se obtuvo menor diámetro de circunferencia (para el cual la dirección gira más) y se escaló por K :

$$K = \frac{Diámetro_{Menor}}{Diámetro_{Mayor}}$$

Al hacerlo, notamos que los diámetros de las circunferencias seguían discrepando; uno de los motivos fue que la dirección traducida desde el motor paso a paso hacia la dirección de las ruedas se hace mediante una relación traslacional y dos rotacionales según la mecánica del robot, dando lugar a que esta traducción sea no lineal. Un dibujo en 3D de la mecánica de la dirección del robot se puede observar en la Figura 3.9.

Otros motivos que dieron consecuencia a este error fueron las imperfecciones mecánicas como, por ejemplo, mala alineación de la dirección o juegos en los vínculos mecánicos que forman parte del sistema de dirección. Los más significativos tuvieron lugar en la caja reductora del motor paso a paso dando lugar a una rotación del brazo a la salida de la caja cercana a los 2° y en la columna de la horquilla con respecto al brazo que la vincula con dicho motor, permitiendo una rotación libre de las ruedas cercana a los 4° . Estos, influyen en que un giro de la dirección puede no verse representado en un giro de las ruedas, o viceversa. Para entender mejor esta complicación, se le ordeno al robot alcanzar un punto situado 15m por delante en línea recta, en la Figura 6.11 podemos observar los resultados.

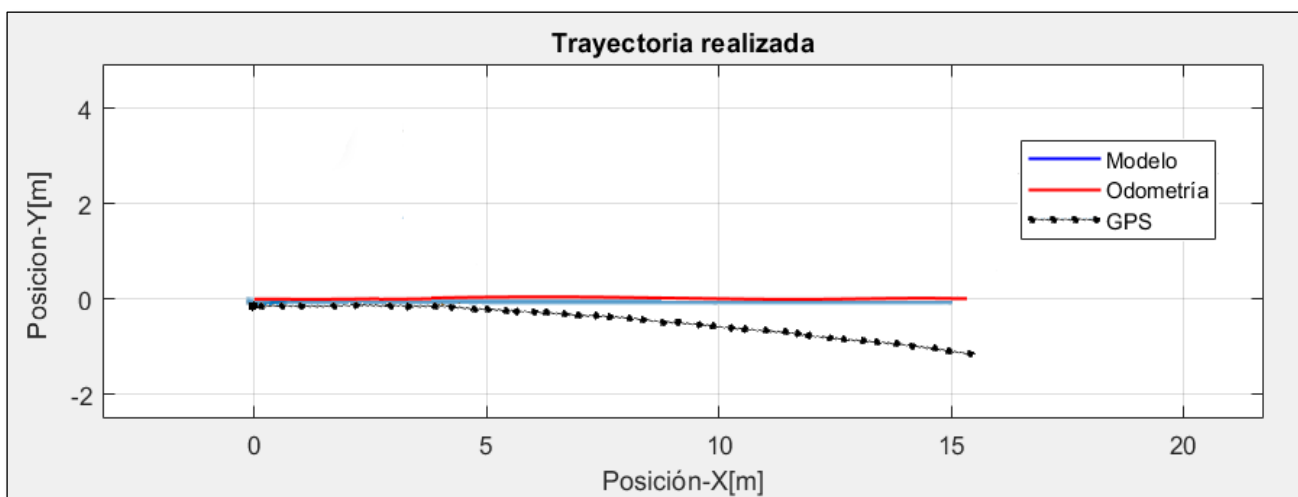


Figura 6.11: Trayectoria en línea recta.

En azul se graficó la trayectoria obtenida desde el modelo matemático del robot, viendo que se logra el objetivo perfectamente.

En rojo, observamos la trayectoria realizada según los datos obtenidos por la odometría, donde podemos ver que, por un lado, el recorrido no es perfectamente recto dado que el controlador fue corrigiendo la trayectoria para mantenerse en línea recta, pero la corrección no es instantánea como lo es en el modelo, dando como resultado leves oscilaciones. Por otro lado, notamos que el robot no se detuvo perfectamente en 15m debido a que el sensado de velocidad del vehículo no es perfecto. Las causas pueden abarcar desde una mala medición de la velocidad hasta una diferencia en el perímetro de las ruedas debido a que alguna se haya desinflado o por que la dirección del robot no se encuentra alineada.

En línea punteada observamos los datos obtenidos desde el GPS con solución FIX (este tipo de precisión en medición se da automáticamente, dependiendo la precisión en las señales satelitales del GPS), esto nos permite concluir que la lectura de la dirección no se corresponde con la real del prototipo. Se nota claramente una desviación en la trayectoria, aunque, si bien el robot mediante odometría cree estar posicionado correctamente (trayectoria roja) vemos que en un recorrido de 15 metros se desvió aproximadamente 1 metro de su posición final. Cabe destacar, que este metro de error en la posición final es según los datos del GPS, que no son exactos ni están muy cerca de serlo. Por este motivo se tomó la medición de la distancia errónea y dio cerca al medio metro, dando por aludido que, aunque el error existe, el error del GPS lo hace más grande, o bien, podría haber sucedido que el GPS de un error en la coordenada y positiva de medio metro, lo que eliminaría el error.

El análisis del comportamiento del GPS queda solo en eso para un posterior posible uso en el robot, ya que en este proyecto se computó la posición del robot bajo leyes de odometría. Lamentablemente, para este problema en particular no encontramos una solución viable y rápida que pudiéramos llevar a cabo para subsanarlo, ya que como mencionamos se debe a problemas de “juego” mecánico, si bien en esta prueba la desviación fue en un sentido, podría haber sido hacia el otro o no tenerla, esto variará según la posición inicial de la dirección, el terreno y/o la trayectoria recorrida.

7. Conclusiones

Antes de dar conclusiones acerca de lo realizado, queremos mencionar que nos pareció un proyecto muy interesante, y que a su vez incluye muchas de las incumbencias profesionales de un ingeniero electrónico. A grandes rasgos destacamos, la teoría de control utilizada en el control de tracción y dirección, el diseño electrónico aplicado en la placa de control, la medición de señales (analógicas y digitales) provenientes de dispositivos captadores, implementación de protocolos de comunicación y el desarrollo de software sobre un microcontrolador para englobar todas estas tareas. Por otro lado, trabajamos sobre la NUC, en la cual se implementó el software que procesa en tiempo real y está basado en la solución de control encargado del seguimiento de trayectoria.

En base a la selección del equipamiento adquirido (sección 4) y a la capacitación teórico/práctica realizada logramos obtener los conocimientos necesarios para afrontar el proyecto.

Pudimos resolver satisfactoriamente los inconvenientes previos que presentaba el vehículo mencionados en la sección 2. Gracias a la adquisición de motores y drivers de mayor porte, se resolvieron los inconvenientes ligados a la falta de potencia en el sistema de tracción, especialmente en terrenos irregulares. Al mismo tiempo se logró implementar el sistema de frenado y la eliminación del sistema de inversión de marcha (anteriormente implementado con relés), de manera satisfactoria.

Debido a que el montaje de los equipos se hizo en forma modular resulta simple el reemplazo de algún componente dañado, ganando tiempo y comodidad.

En cuanto a lo comentado sobre tracking, si bien el control tiene un comportamiento satisfactorio, en el marco de este proyecto el mismo se realizó solo mediante odometría, lo que trae consigo errores. Por lo que la posibilidad de tomar datos GPS para realizar el control brinda una disminución en los errores posibilitando mayor precisión en el seguimiento de una trayectoria ya que el controlador se realimentaría con la posición real del vehículo obtenida por GPS o por cortos periodos de tiempo mediante odometría en caso perdida de señal. Además, podría implementarse visión artificial para el reconocimiento de la posición real del robot, realizando detección de surcos mediante un algoritmo basado en el procesamiento de imágenes en tiempo real sobre el terreno rural, permitiendo recalcular instante a instante las ordenes de control según la desviación entre el vehículo y los surcos.

Finalmente, como mejoras a futuro podemos destacar la implementación de un control de corriente debido a la posibilidad de medir la misma con los sensores adquiridos. Para la navegación autónoma, el agregado de un giroscopio o de un sensor de campo magnético para mejorar la precisión en la posición del robot. Mejorar la dirección mecánicamente para disminuir los problemas en cuanto a la alineación y la precisión entre el vínculo que relaciona al encoder, que es quien realiza la medición, y a la dirección de giro que el robot posee verdaderamente. También, se podría implementar una nueva solución de tracking que permita alcanzar los puntos que ahora son inalcanzables por el robot de una manera más efectiva.

8. Bibliografía

- [1] **Zhao, Jian y Yu, Yangwei.** Brushless DC Motor Fundamentals - Application Note. Julio de 2011.

- [2] **Padmaraja Yademale (Microchip).** AN855 - Brushless DC (BLDC) Motor Fundamentals - Application Note. 2003.

- [3] **Copley Controls Corp.** What is 'Field Oriented Control' and what good is it? - Article.

- [4] **H. SAX - SGS-THOMSON Microelectronics.** Stepper Motor Driving - Application Note.

- [5] **Omron.** General-purpose Absolute Encoder with External Diameter of 50 mm (E6CP-A) - Data Sheet.

- [6] **Escobar García C. F., Martínez Peña J. A., Telléz González G. A.** CONTROL DE UN MOTOR BRUSHLESS DC CON FRENADO REGENERATIVO - Tesis. 2005, Bogotá.

- [7] **Bascetta Luca, Cucci Davide A., Matteucci Mateo.** Kinematic trajectory tracking controller for an all-terrain Ackermann steering vehicle - Paper. 2016.

- [8] **Giovanni Indiveri.** Kinematic Time-invariant Control of a 2D Nonholonomic Vehicle - Paper. 1999.

- [9] **Allegro Microsystem Inc.** Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor (ACS712) - Data Sheet.

- [10] **NXP Semiconductors.** FRDM-K64F Freedom Module User's Guide. Agosto de 2016.

- [11] **NXP Semiconductors.** Kinetis K64F Sub-Family Data Sheet. Noviembre de 2016.

- [12] **NXP Semiconductors.** FRDM-K64F Schematic. Abril de 2016.

9. Apéndice

En esta sección dejaremos ilustrado el resultado de cajas de seguridad que fueron montadas exteriormente sobre el gabinete, las mismas contienen cables que corresponden a los motores de tracción BLDC, y los sensores de corriente.

El software que se utilizó para hacer los diseños es SolidWorks 2018, una herramienta muy poderosa para hacer diseños 3D.

El listado de impresiones se detalla a continuación:

- 1 caja contenedora de los cables correspondiente a dos drivers y dos sensores de corriente (Figura 9.1).
- 1 tapa para la caja antes mencionada (Figura 9.2).
- 2 cajas contenedoras de cables correspondientes a un driver y un sensor de corriente (Figura 9.3).
- 2 tapas para dichas cajas (Figura 9.4).
- 4 soportes para los sensores de corriente (Figura 9.5).

A continuación, se muestran los diseños finales.

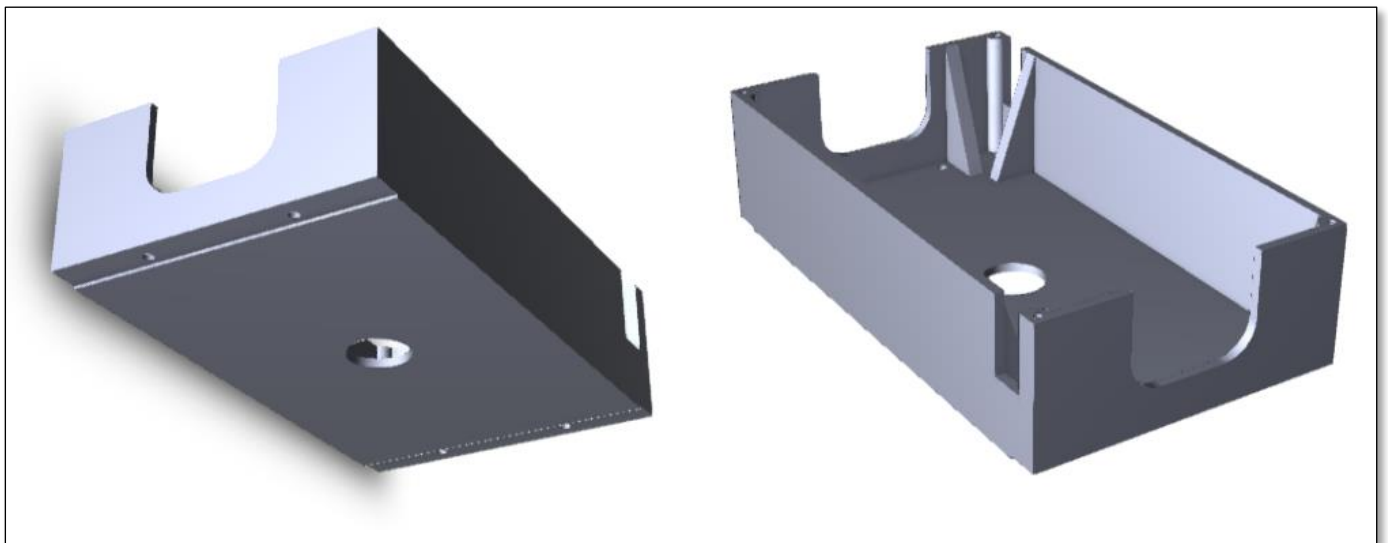


Figura 9.1: Caja contenedora para dos drivers.

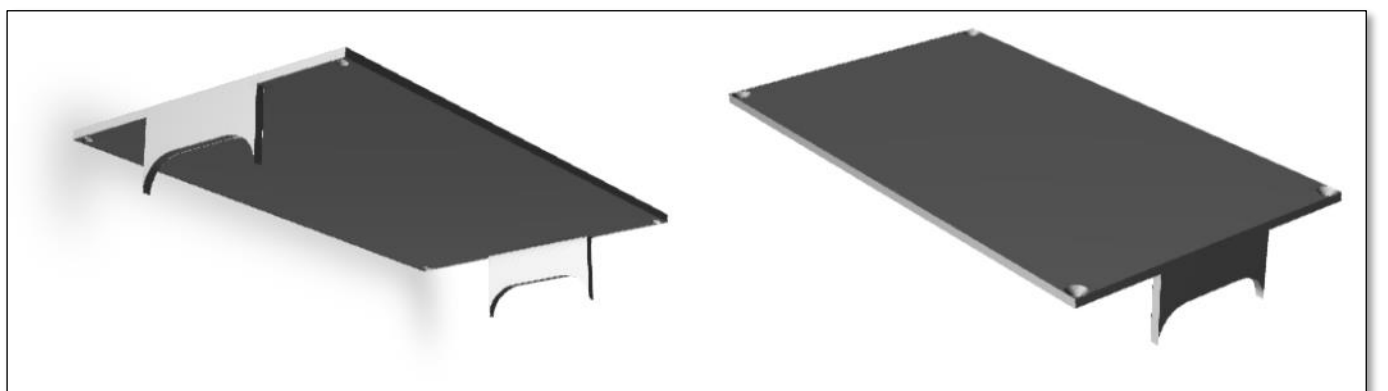


Figura 9.2: Tapa para dos drivers.

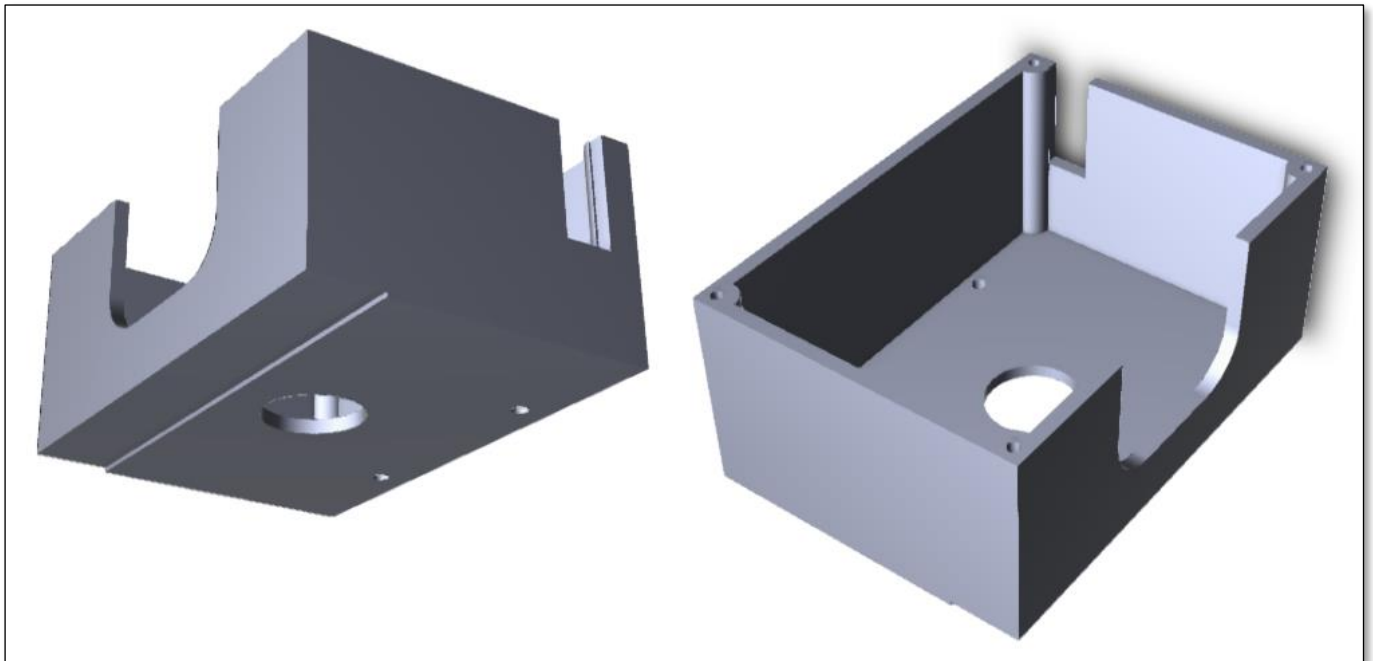


Figura 9.3: Caja contenedora para un driver.

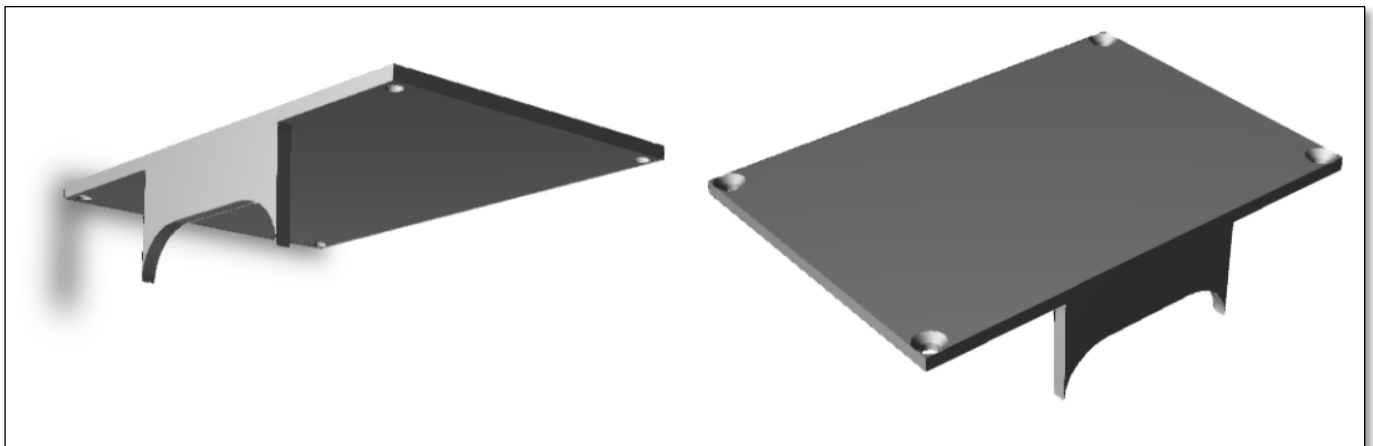


Figura 9.4: Tapa para un driver.

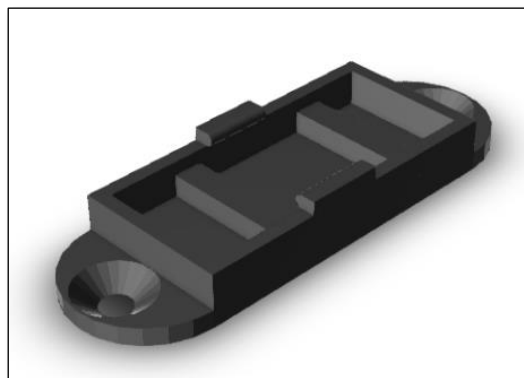


Figura 9.5: Soporte para sensor de corriente.

9.1.1. Resultados finales

En esta sección mostraremos las fotos de las impresiones finales que fueron montadas en el prototipo.



Figura 9.6: Foto de caja contenedora de dos drivers.



Figura 9.7: Foto caja contenedora de dos drivers vista desde arriba.



Figura 9.8: Foto de tapa de dos drivers.



Figura 9.9: Foto caja contenedora de un driver.

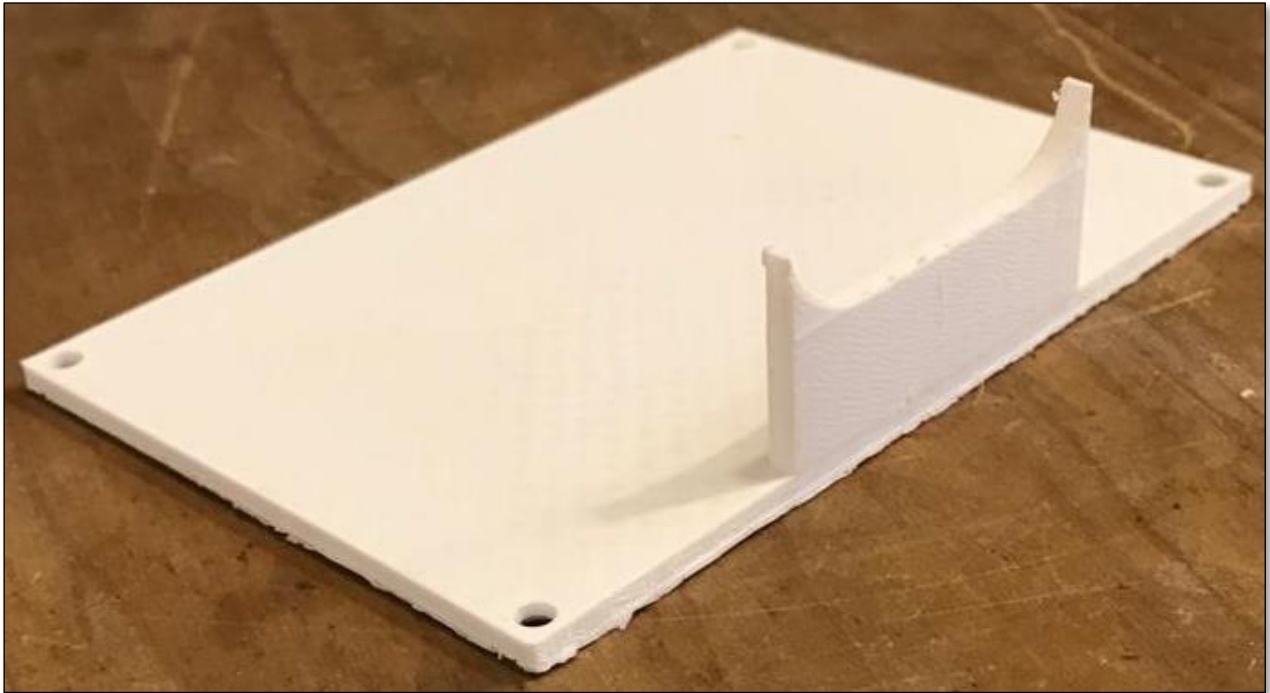


Figura 9.10: Tapa de un driver.

Por último, en la Figura 9.11, se puede observar una foto del gabinete de control ya montado y, a su alrededor, las cajas impresas y los drivers de los motores BLDC.



Figura 9.11: Cajas y drivers montados sobre el gabinete.