Universidad Nacional del Comahue

Facultad de Ingeniería

Proyecto Integrador Profesional

Desarrollo e implementación del sistema de control de bajo nivel de un robot 4WD para ambientes frutícolas

FACUNDO ORTIZ – LEONEL WIRTH

Ing. EDGARDO BENITEZ PICCINI

Lic. RAFAEL IGNACIO ZURITA

Neuquén Argentina

2022

Prefacio

Esta tesis es presentada como parte de los requisitos finales para optar al grado académico de *Ingeniero Electrónico*, otorgado por la Universidad Nacional del Comahue, y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otras. La misma es el resultado de la investigación llevada a cabo en el Departamento de Electrotécnia de la Facultad de Ingeniería, en el período comprendido entre Julio-2021-15 y Noviembre-2022-08, bajo la dirección del Ing. Edgardo Benitez Piccini y la codirección del Lic. Rafael Ignacio Zurita.

Ortiz Facundo Gabriel y Wirth Leonel Emiliano
Facultad de Ingeniería
Universidad Nacional del Comahue
Neuquén, 08 de Noviembre de 2022.

UNIVERSIDAD NACIONAL DEL COMAHUE

Facultad de Ingeniería

La presente tesis ha sido aprobada el día ....., mereciendo la calificación de .....

Agradecimientos

En primer lugar, quiero comenzar agradeciendo a mis padres y hermana por siempre brindarme su apoyo incondicional y motivarme a seguir adelante durante todos estos años. Además, quiero agradecer especialmente a mi abuela Carmen por estar siempre presente, por su cariño y creer en mi. Sin ustedes, nada de todo esto hubiese sido posible.

En segundo lugar, quiero agradecer a Ianina por ser un pilar fundamental durante todos estos años. Gracias por brindarme tu apoyo en todo momento, por todas las horas de estudio llenas de risas y mates que hicieron mas llevaderas las cursadas y por estar siempre que te necesite.

En tercer lugar, agradezco a mis amigos y compañeros de estudio que me acompañaron durante el transcurso de la carrera, con los cuales compartí tantos buenos momentos. Particularmente quisiera agradecer a Leonel, por su constante dedicación, compromiso y compañerismo durante el proceso de realización de este trabajo.

Facundo Ortiz.

Quiero dar las gracias toda a mi familia, en particular a mis padres, hermana, primos y sobrinos que sin dudarlo estuvieron siempre para apoyar y acompañar. Además quiero agradecer los esfuerzos y sacrificios que hicieron mis padres para que pueda estudiar esta gran carrera. Ademas, me gustaría realizar un agradecimiento especial a mi novia Gisela, que Siempre tuvo fe en mi y me brindó un apoyo incondicional a lo largo de todos estos años.

Quiero agradecer a mi mentor "Guga"por potenciar mi desarrollo profesional en los años que trabajamos juntos.

Por último y no menos importante agradezco a mis amigos que me acompañaron durante todos este proceso, lleno de risas, mates y mucho estudio. En particular Facundo Ortiz, mi compañero de PIP por querer trabajar en conjunto.

Leonel Wirth.

Resumen

En el presente trabajo se muestra el proceso de diseño e implementación de un sistema de control de velocidad. Este se realizó sobre las cuatro ruedas de un vehiculo 4WD (Four Wheel Drive), proporcionado por el grupo de investigación de vehículos y sistemas inteligentes (GIVSI).

Para la realización de dicho sistema de control, se procedió en varias etapas. Inicialmente se identificó la planta (conformada por motor, engranaje, rueda y encoder), la cual luego se discretizó. A partir de la versión discreta, se diseñó el controlador. Luego se diseñó y fabricó una placa PCB, que contiene todos los componentes necesarios para implementar el controlador. Posteriormente se diseñó un software que se ejecuta en la misma. Dicho software fue realizado a partir de un RTOS, en el que se ejecutan todos los sistemas de control y además, implementa el protocolo de comunicación Modbus. Se plantea el sistema como un esclavo Modbus, al cual se le indican las señales de referencia y se le puede consultar el estado de las variables medidas. Seguido de esto, se realizó una estimación de la capacidad de las baterías con las que cuenta el vehículo, a modo de verificar el estado en el que encuentran. Además, se busca que el software sea robusto, por lo que se implementaron estrategias como control de errores, tiempos máximos de rutinas, WDT (Watch Dog Timer), tensión de bus mínima de funcionamiento, entre otras cosas. Por último se integraron todos los componentes del sistema en el vehículo y luego se realizaron pruebas experimentales, a modo de validación del trabajo.

Abstract

This work shows the design and implementation process of a speed control system. This is done on the four wheels of a 4WD vehicle (Four Wheel Drive), provided by the vehicles and intelligent systems (GIVSI) research group.

To create the control system, it proceeds in several stages. First, the plant (consisting of motor, gear, wheel and encoder) is identified, it is discretized and the controller is designed. A PCB board is then designed and manufactured, containing all the necessary components to implement the controller. Software is designed to run on it. This is done from an RTOS, in which all the control systems are executed and also implements the Modbus communication protocol. The system is considered as a Modbus slave, to which the setpoint signals are indicated and the measured variables can be consulted. An estimation of the batteries capacity that the vehicle has is made, in order to verify the state in which they are found. In addition, the software is intended to be robust, so strategies such as error control, maximum routine times, WDT (Watch Dog Timer), minimum operating bus voltage, among other things, are implemented. Finally, all the components of the system are integrated into the vehicle and then experimental tests are carried out, as a proyect validation.

Índice general

1. Introducción y objetivos **1**
1.1. Motivación . . . 1
1.2. Fundamentación . . . 1
1.3. Objetivos . . . 2
1.4. Estructura del trabajo . . . 3

2. Modelado y control del sistema **5**
2.1. Modelo matemático del sistema . . . 5
2.1.1. Principio de funcionamiento del motor . . . 5
2.1.2. Modelo del motor de CC . . . 7
2.1.3. Modelado del motor de CC con carga acoplada por reductor . . . 8
2.2. Análisis del comportamiento del sistema . . . 10
2.2.1. Verificación de la linealidad del sistema frente a señales continuas . . . 11
2.2.2. Verificación de la linealidad del sistema con ESC . . . 11
2.3. Control del sistema . . . 12
2.3.1. Control PID . . . 12

3. Diseño e implementación del Hardware **15**
3.1. Electrónica necesaria para el control . . . 16
3.1.1. Microcontrolador . . . 16
3.1.2. Sensores de velocidad . . . 16
3.1.3. Sensores de corriente . . . 16
3.1.4. Puerto de comunicación . . . 16
3.1.5. Actuador . . . 16
3.1.6. Etapa de alimentación . . . 16
3.2. Acondicionamiento mecánico . . . 17
3.2.1. Bujes . . . 17
3.2.2. Diseño de adaptadores . . . 18
3.3. Diseño del hardware . . . 19
3.3.1. Diseño del esquemático . . . 19
3.3.2. Consideraciones de diseño . . . 22
3.3.3. Diseño y fabricación del pcb . . . 24

4. Implementación de software de control en FreeRTOS **27**
4.1. Sistemas operativos de tiempo real (RTOS) . . . 27
4.1.1. ¿Qué es RTOS? . . . 27
4.1.2. Tareas . . . 28
4.1.3. Semáforos . . . 29
4.1.4. Colas . . . 30
4.1.5. Principio de funcionamiento del planificador de CPU en FreeRTOS . . . 31
4.2. Uso de memoria . . . 33

4.3. Arquitectura y diseño del software desarrollado . . . 34
4.3.1. Tarea Calculo_Velocidad . . . 35
4.3.2. Tarea Detección_Cero . . . 37
4.3.3. Tarea Corriente . . . 37
4.3.4. Tarea Modbus . . . 40
4.3.5. Tarea Control . . . 40
4.3.6. Tarea Ping . . . 42
4.3.7. Rutina de interrupción . . . 43

5. Protocolo de comunicación 45
5.1. Necesidades que se cubren . . . 45
5.2. ¿Qué es Modbus RTU? . . . 45
5.2.1. Modo de funcionamiento . . . 45
5.2.2. Utilización de variables flotantes . . . 46
5.3. Implementación de Modbus . . . 47
5.4. Estructura de lectura-escritura . . . 50
5.4.1. Lectura de holding registers . . . 50
5.4.2. Escritura de holding registers . . . 50
5.4.3. Excepciones . . . 51
5.5. Manual de uso del equipo visto desde un dispositivo maestro . . . 51
5.5.1. Capa física . . . 52
5.5.2. Escritura de Setpoints . . . 55
5.5.3. Lectura de datos . . . 55

6. Pruebas de integración de hardware y software 57
6.1. Linealización e identificación del sistema . . . 57
6.1.1. Diseño e implementación de bloque linealizador . . . 58
6.2. Identificación del sistema . . . 60
6.2.1. Obtención de R_a . . . 62
6.2.2. Obtención de L_a . . . 62
6.2.3. Obtención de k_d . . . 63
6.2.4. Obtención de k_f . . . 64
6.2.5. Obtención de B . . . 64
6.2.6. Obtención de J . . . 65
6.2.7. Resultados de la linealización e identificación . . . 66
6.3. Discretización del sistema identificado . . . 67
6.4. Validación del modelo . . . 67
6.5. Implementación y calibración del controlador . . . 68
6.5.1. Análisis de perturbaciones . . . 71
6.5.2. Pruebas de trayectoria a lazo abierto . . . 74
6.6. Esquema de consumo energético . . . 74
6.6.1. Ensayo de descarga . . . 77
6.6.2. Figura de consumo de cada subsistema electrónico . . . 77

7. Conclusiones y mejoras 79
7.1. Recapitulación y conclusión . . . 79
7.2. Mejoras y desarrollos a futuro . . . 81

A. Ensayos del sistema de control 85
A.1. Ensayos del sistema de control . . . 85

B. Diagramas de conexión **91**
B.1. Diagrama topográfico 91
B.2. Diagramas de conexionado 92

Índice de figuras

2.1. Esquema de motor de una sola espira, extraído de [11] 6
2.2. Modelo simplificado de un motor de CC. 7
2.3. Modelo simplificado de un motor de CC con carga. 9
2.4. Relación de velocidad vs tensión en motores al emplear fuente de CC. 11
2.5. Relación de velocidad vs tensión en motores al emplear un ESC. 12
2.6. Respuesta ideal y real obtenida del ESC para CCR=[300,3500,8000]. 13
2.7. Diagrama en bloques del sistema de control. 14

3.1. Estado inicial de recepción del vehículo. 15
3.2. Ejes sin buje de sujeción. 17
3.3. Eje de rueda con múltiples perforaciones. 18
3.4. Montaje empleando nuevo eje y buje. 18
3.5. Soportes diseñados para componentes. 19
3.6. Esquema de etapa de potencia. 19
3.7. Diagrama de conexionado del microcontrolador. 20
3.8. Esquema de conectores para enconders incrementales. 21
3.9. Esquema de conectores para sensores de corriente. 21
3.10. Esquema de conectores para ESC. 22
3.11. Diagrama en bloques de funcionamiento del sensor INA219, extraído de su hoja de datos. 23
3.12. 23
3.13. 24
3.14. Versión final de placa de control. 25

4.1. Ejecución de tareas con super loop (a) comparado con un sistema operativo (b). . 28
4.2. Estados posibles de una tarea en FreeRTOS. 29
4.3. Proceso de escritura y lectura de buffer utilizando un semáforo. En (a) se observa el momento previo a la escritura del buffer, donde el contador de semáforo esta en 0. En (b) se realiza la escritura y el contador de semáforo se incrementa en 1, habilitando la lectura del buffer, que se realiza en (c). En (d) se observa el momento posterior a la lectura, donde el buffer se limpia y el contador se actualiza a 0. . . 30
4.4. Proceso de escritura y lectura de memoria ejecutado por múltiples tareas, sin colas, ni protección de variables. 31
4.5. Buffer que brinda la cola para el manejo de variables globales. 31
4.6. Ejecución de tareas en función del tiempo en FreeRTOS. 32
4.7. Ejecución de tareas en función del tiempo en FreeRTOS. 32
4.8. Ejecución de tareas en función del tiempo en FreeRTOS. 33
4.9. Ejecución de tareas en función del tiempo en FreeRTOS. 33
4.10. Diagrama en bloques de la estructura de CMSIS, extraído de la página oficial [1]. 35
4.11. Diagrama en bloques de las tareas que se utilizan en el diseño e implementación del software y como dichas tareas se vinculan entre sí. 36

4.12. Período entre dos pulsos consecutivos de encoder, a máxima velocidad de rotación de un motor. 37
4.13. Diagrama de flujo de la tarea Calculo_Velocidad. 38
4.14. Diagrama de flujo de la tarea Deteccion_Cero 39
4.15. Diagrama de flujo de la tarea Corriente 40
4.16. Diagrama de flujo de la tarea Modbus 41
4.17. Diagrama de flujo de la tarea Control 42
4.18. Diagrama de flujo de la tarea Ping 43

5.1. Trama de comunicación con protocolo Modbus. 46
5.2. En esta imagen se muestran los distintos tipos de endianness. 47
5.3. En esta imagen se muestran el proceso de conversión de dos holding registers en el mapa Modbus uint16 en una variable flotante. 48
5.4. Trama de solicitud de lectura de Holding registers, interpretada por Rapid SCADA Parser. 50
5.5. Respuesta a la solicitud de lectura de HR's, interpretada por Rapid SCADA Parser. 51
5.6. Solicitud de escritura de multiples HR's, interpretada por Rapid SCADA Parser. 51
5.7. Respuesta a la solicitud de escritura de multiples HR's, interpretada por Rapid SCADA Parser. 52
5.8. Solicitud de escritura de HR en direcciones que no existen en el mapa Modbus (5.8a) y respuesta de la excepción correspondiente por parte del controlador (5.8b). 53
5.10. placa destinada a convertir UART a RS-485. 53
5.9. Solicitud de que el esclavo realice una función inexistente (5.8a) y respuesta de la excepción correspondiente por parte del controlador (5.8b). 54
5.11. Esquemático de la placa conversora de UART a RS-485 con auto-enable. 54
5.12. Dispositivo para aislar eléctricamente la comunicación USB. 55

6.1. Configuración del ensayo para medición de fuerza de tracción. 58
6.2. Esquema de diagrama en bloques que se sigue para la linealización del sistema . . 58
6.3. Rectas establecidas a partir de la nube de puntos (CCR,Velocidad) de la relación inversa a la obtenida en la figura 2.5. 59
6.4. Función linealizadora. 60
6.5. Implementación del bloque linealizador en Simulink. 60
6.6. En la figura se observan los pasos numerados para incluir el código que se genera a partir de la herramienta Embedded Coder dentro de path del programa. 61
6.7. Modelo motor de CC con rotor bloqueado, al cual se le coloca una resistencia externa conocida en serie. 62
6.8. Curva de tensión obtenida sobre la resistencia externa conocida, que se coloca en serie al motor. El rotor permanece bloqueado durante el ensayo y la señal PWM de entrada, posee un ciclo útil de 10 %. 63
6.9. Configuración que se emplea para el ensayo en el que se obtiene el parámetro k_d. 64
6.10. Esquema representativo de como se realiza el ensayo para obtener el parámetro k_f de un motor. 65
6.11. Curva de velocidad de la cual se obtiene t_m. 65
6.12. Respuesta de los motores ante la función linealizadora 66
6.13. Error relativo porcentual obtenido de la planta linealizada, con respecto a la respuesta ideal. 67
6.14. Esquema en bloques, con el que se realiza la simulación para validar comportamiento de la planta discreta teórica obtenida. 68
6.15. Esquema de bloques que se genera en Simulink para calibración virtual de PID . 68
6.16. Respuesta al escalón del sistema de control simulado, luego de realizar la calibración virtual del PID. 69

6.17. Vehículo, sin cobertor superior, con todos los componentes electrónicos integrados. Los diagramas topográfico y de conexionado, pueden observarse en el apéndice B. 70
6.18. Curvas de velocidades obtenidas para los setpoints 0,4 Rev/Seg, 0,8 Rev/Seg y 1 Rev/Seg, con constantes de PID obtenidas mediante la simualción, sobre una superficie plana) . . . 71
6.19. Curvas de velocidades obtenidas para los setpoints 0,4 Rev/Seg, 0,8 Rev/Seg y 1 Rev/Seg, con constantes de PID logradas experimentalmente, sobre una superficie plana) . . . 72
6.20. Error relativo porcentual existente entre los ensayos integrales con las constantes obtenidas a partir de la simulación y las generadas de manera experimental. . . . 72
6.21. Diagrama en bloque del sistema para el análisis de propagación de perturbación de tipo escalón. . . . 73
6.22. Curva de respuesta obtenida ante una perturbación de tipo escalón, en el sistema simulado. . . . 73
6.23. Representación gráfica de como se efectúa el ensayo en carga del sistema de control. 74
6.24. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira. . . . 75
6.25. Variación del ángulo Yaw que se obtiene cuando se establece un setpoint de 1 Rev/Seg sobre todos los motores y este se mueve en una superficie plana una distancia aproximada de 22 metros . . . 75
6.26. En la figura se puede observar el registrador con carga activa utilizado para realizar el ensayo de descarga de las baterías. . . . 76
6.27. En esta figura se muestra los datos de tensión y corriente registrados durante el ensayo de descarga de cada batería. Los gráficos del conjunto (a) corresponden a la batería N°1 y los del conjunto (b) a la N°2. . . . 77

A.1. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 1 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 0,5 Kg y en verde (FP) cuando se retira. . . . 85
A.2. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 1 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1 Kg y en verde (FP) cuando se retira. . . . 86
A.3. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 1 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira. . . . 86
A.4. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,8 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 0,5 Kg y en verde (FP) cuando se retira. . . . 87

A.5. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,8 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1 Kg y en verde (FP) cuando se retira. 87
A.6. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,8 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira. 88
A.7. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 0,5 Kg y en verde (FP) cuando se retira. 88
A.8. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1 Kg y en verde (FP) cuando se retira. 89
A.9. En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira. 89

B.1. En el diagrama topográfico se muestra la disposición de los componentes dentro del chasis de vehículo. 91
B.2. Diagrama de conexionado de placa de control. 92
B.3. Diagrama de conexionado de conversor RS-485, conversor dc-dc aislado y sensor de corriente para medir tensión en baterías. 92
B.4. Diagrama de conexionado de variadores de velocidad y sensores de corriente de los motores. 93
B.5. Diagrama de conexionado de encoders incrementales y motores. 93

Índice de tablas

5.1. Funciones de Modbus. 46
5.2. Tipos de datos utilizados en el protocolo Modbus. 47
5.3. Mapa Modbus implementado. 49
5.4. En la siguiente tabla se muestran los distintos códigos de excepción que posee el protocolo Modbus RTU y su respectiva descripción. 52

6.1. Valores promedio de fuerza que se obtienen para el ensayo tracción del vehículo. . 57
6.2. Parámetros que se obtienen del ensayo de los motores 66
6.3. Comparación entre las velocidades obtenidas teóricamente a partir la planta discreta y la velocidad real medida. 68
6.4. Torque de fricción que recibe la rueda a modo de perturbación, en función del peso de la carga agregada. 74
6.5. En la siguiente tabla se muestra la figura de consumo eléctrico de las partes individuales y luego del sistema en su conjunto en condiciones nominales. 78

Capítulo 1

Introducción y objetivos

En este capítulo se exponen los fundamentos y objetivos en los cuales se enmarca el desarrollo del presente. En la primera sección, se exponen los motivos por los cuales se decide realizar este tipo de trabajo. En la segunda sección se desarrollan los motivos y la necesidad que da origen a este proyecto. En la tercera sección se explican los objetivos a resolver. En la cuarta sección se presenta una explicación acotada de cómo se estructura el desarrollo del trabajo.

1.1. Motivación

La elección de este trabajo como proyecto integrador profesional de la carrera Ingeniería Electrónica se fundamenta en varios motivos. El primero se destaca en el cursado de la materia *Introducción a los Vehículos Inteligentes*, la cual brinda una noción de la existencia y diversidad de esta clase de vehículos y como pueden impactar en la sociedad. La segunda razón se centra en el interés individual de ambos en trabajar en un proyecto que ayude a consolidar habilidades relacionadas a la implementación de los sistemas de control sobre sistemas embebidos en conjunto con el desarrollo del Hardware requerido para esto. Por último, al formar parte del grupo de investigación de vehículos y sistemas inteligentes (GIVSI), surge la posibilidad de continuar con el desarrollo del vehículo autónomo, el cual involucraba temáticas de interés para ambos.

1.2. Fundamentación

Hace ya varios años, la fruticultura regional continua disminuyendo su actividad por diferentes motivos. La situación económica nacional y provincial no han contribuido en su crecimiento. La falta de una política económica estable y funcional, en conjunto con la disminución de presupuestos para la producción y mejora de la infraestructura, han incursionado en la desaparición de un enorme porcentaje de los productores independientes y generado una profunda crisis para los productores de mayor escala [12], [13], [25]. A esto se suma la dificultad para encontrar mano de obra calificada, elevados costos asociados a métodos productivos antiguos que no favorecen la competitividad económica y la complejidad asociada al tipo de cultivo.

En este contexto, la innovación tecnológica es uno de los pilares fundamentales para lograr que este rubro vuelva a ser competitivo dentro del mercado. En respuesta a esto, surge la agricultura de precisión [15] que consiste en la capacidad de recopilar, interpretar y aplicar información relativa de los cultivos y su ambiente, brindado conocimiento utilizable por los productores. Basado en esto, se implementan estrategias productivas y métodos de trabajos eficaces que logran maximizar su rentabilidad y calidad de producción disminuyendo el impacto ambiental generado por la actividad. Esto no solo es útil para los productores, también puede ser una fuente de información útil para proveedores de maquinaria o insumos, entre otros.

En el desarrollo de robótica y la automatización de sus funciones destinada a operar en estos ambientes se requiere información vinculada a la percepción del entorno y el comportamiento de

los cultivos. Para poder generar dicha percepción, es necesario poder medir múltiples variables a lo largo de toda la plantación. Para obtener esto, por ejemplo se podría realizar un recorrido por la plantación obteniendo las distintas variables requeridas o incluso esto podría lograrse a través de una red autónoma interconectada dentro de la plantación la cual diariamente mide las variables. Si bien esta última es una buena opción, en cultivos grandes, la complejidad y los costos de mantenimiento de la red se incrementa enormemente. Como alternativa surgen las plataformas móviles autónomas. De esta manera, los sensores involucrados en la adquisición de información pasan a estar sobre el vehículo, que recorre de manera autónoma la plantación [10], [24].

La variedad de vehículos autónomos asociados a la adquisición de datos dentro del mercado es amplia, como por ejemplo los terrestres (con sus distintos tipos de tracción) y los aéreos (aviones o drones). Particularmente resulta de interés el análisis de los vehículos terrestres, ya que son menos costosos y su mecanismo de movilidad es más simple. Dentro de esta categoría, los de tracción integral son los ampliamente utilizados, debido a las características asociadas al entorno sobre el que se va desarrollar su actividad. Los terrenos que debe atravesar este tipo de vehículos cuentan con hierbas altas, rocas, pozos, canales de riego, etc.

Este vehículo se desarrolla en el marco del trabajo conjunto entre la UNCo y el INTA. Su construcción mecánica se realizó en una primera fase de su desarrollo, y este proyecto integrador se centró en la tarea de dotarlo de control de bajo nivel. En trabajos futuros se podrá integrar con sensores a la plataforma para brindarle capacidades de percepción del ambiente y control de trayectorias.

Partiendo del trabajo mecánico inicial, este proyecto integrador profesional se centra el la etapa del control de bajo nivel.

1.3. Objetivos

El objetivo general del presente trabajo es el diseño y desarrollo de un sistema de control de velocidad para los cuatro motores de un robot 4WD (Four Wheel Drive) autónomo. Este debe contar con la capacidad de recibir señales de referencia de velocidad desde un dispositivo maestro externo y retornar información del sistema relativa a las condiciones de funcionamiento en caso de que se solicite.

A continuación se mencionan los objetivos específicos:

- Caracterizar y modelar los cuatro sistemas (uno por cada rueda), compuestos por: motor, engranaje, rueda y encoder óptico.
- Diseñar y sintonizar un lazo de control de velocidad para cada rueda del robot de manera independiente.
- Utilizar un sistema operativo de tiempo real (RTOS) para la implementación de los cuatro lazos de control en un microcontrolador.
- Establecer la estructura de los paquetes de datos y el protocolo de comunicación entre un dispositivo externo (maestro), que enviará las señales de referencia, y el sistema a diseñar (esclavo) que recibirá las órdenes.
- Realizar el control de versiones, y escribir la documentación del código fuente del software desarrollado durante el proyecto.
- Diseñar e implementar una placa (PCB) que integre todo el hardware utilizado para el funcionamiento del sistema de control.

- Integrar todos los componentes anteriores sobre la plataforma móvil. El sistema resultante compuesto por: baterías, motores, encoders, sistema embebido e interfaz de comunicación es el sistema de control resultante. Realizar pruebas funcionales de todo el sistema.
- Verificar, mediante pruebas con diferentes cargas, la respuesta del control de lazo cerrado de cada rueda.
- Estimar el consumo energético del conjunto, evaluar e implementar las estrategias que correspondan para el uso de baterías de plomo-ácido sobre el robot.
- Realizar pruebas en campo del sistema integrado con trayectorias predefinidas.

1.4. Estructura del trabajo

A continuación, se detalla brevemente como se estructura el desarrollo para este trabajo.

En el Capítulo 2 se explica y desarrolla el modelo matemático del sistema. Para esto se parte del análisis del motor de corriente continua de imanes permanentes. Luego se analiza su comportamiento cuando se lo utiliza en conjunto con un variador de velocidad. Por último, se explica brevemente el tipo de control que se desea implementar.

En el Capítulo 3 se realiza el diseño e implementación del hardware requerido para aplicar el sistema de control sobre el vehículo. Se parte de un análisis de las necesidades y en función de esto, se seleccionan los componentes que mejor se ajusten a la situación. Luego se implementa una placa de desarrollo que vincule físicamente todos estos.

En el Capítulo 4 se expone el principio de funcionamiento de los sistemas operativos de tiempo real, que ventajas representa frente a otras posibilidades y como este se implementa en base a las necesidades del proyecto.

En el Capítulo 5 se explica el modo de funcionamiento del protocolo de comunicación que se utiliza entre el vehículo y un dispositivo maestro, como se empaqueta la información dentro de su trama y como se utiliza dentro del trabajo.

En el Capítulo 6 se establece como se realiza la identificación del sistema y como se calibra el controlador. Para esto se realiza una simulación, mediante MATLAB y Simulink, en la cual se ingresan diferentes entradas de tipo escalón y se observa como responde el conjunto. También se realiza el ensayo de estado de las baterías y consumo del sistema. Por último, se lleva a cabo la prueba integral del sistema de control para distintas señales de referencia y perturbaciones.

En el Capítulo 7 se realiza una conclusión en base a los resultados obtenidos de las pruebas integrales y los objetivos iniciales. Por último, se mencionan posibles mejoras para este trabajo, con el objetivo de continuar con la línea de trabajo.

Capítulo 2

Modelado y control del sistema

El presente capítulo se dedica a describir el principio de funcionamiento general del motor de corriente continua (CC) de imán permanente, establecer su modelo matemático y como este actúa cuando se utiliza en conjunto con un variador de velocidad o ESC (por las siglas de su denominación en inglés, Electronic Speed Controller). Por último se describe el tipo de controlador que se implementa.

2.1. Modelo matemático del sistema

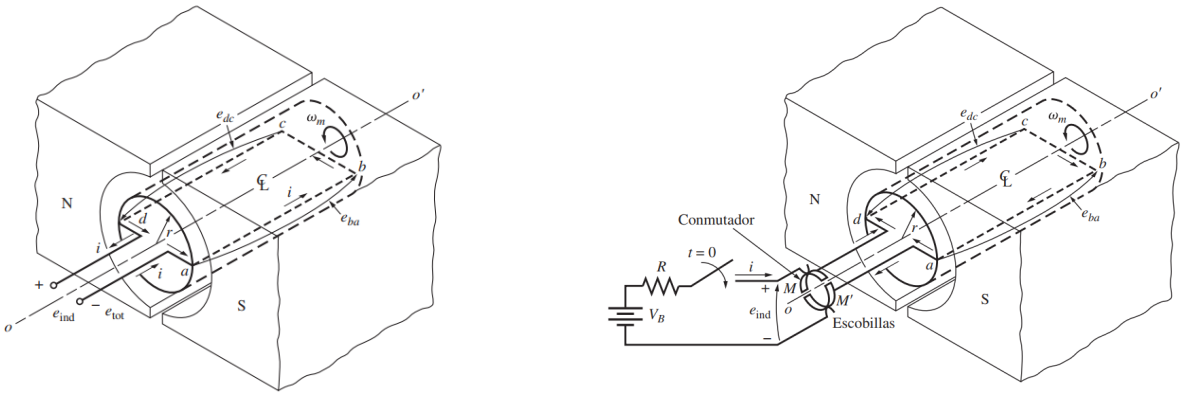
2.1.1. Principio de funcionamiento del motor

Los motores son dispositivos que se caracterizan por ser capaces de transformar energía eléctrica en energía mecánica. Cuando se aplica una diferencia de potencial en sus bornes, se obtiene un movimiento de rotación en el eje del motor, y la velocidad con la que se realiza dicho movimiento puede aumentar o disminuir manipulando la tensión presente en los bornes de entrada. A continuación se detalla la composición y su funcionamiento de una manera más profunda mediante el modelo de una sola espira [11] [14] [18].

Un motor se subdivide en dos grandes partes, el rotor y el estator. El rotor se compone de múltiples laminas delgadas de material ferromagnético, labradas y aisladas eléctricamente entre sí. Generalmente estas laminas son de hierro debido a la baja resistividad y se hallan unidas al eje del motor. El flujo magnético induce un voltaje dentro del núcleo ferromagnético al igual que en la espira. Esto genera corrientes parásitas cuya magnitud depende directamente del tamaño del material resistivo por el cual circulan. Por ese motivo, el núcleo se lamina para reducir la magnitud de las corrientes parásitas producidas y aumentar la eficiencia del motor reduciendo las perdidas por calor. Esta configuración mejora la interacción del flujo magnético que atraviesa el rotor. Dentro de dichos huecos labrados se encuentran los devanados de cobre por los cuales circula la corriente que se establece al colocar una señal en los bornes de entrada del motor. Mientras más devanados tiene la armadura, mas suave es la transición en las conmutaciones, logrando un mejor funcionamiento del motor.

Como se trata de un motor de CC de imán permanente, dentro del estator se encuentran dos imanes permanentes de polaridades opuestas que generan el campo magnético necesario para lograr el funcionamiento del motor. Debido a que el campo magnético está siempre activo, se evita tener que generarlo a través de un circuito de excitación independiente como es necesario con otro tipo de motores. Luego se encuentra el conmutador, que se encarga de cerrar el circuito entre las distintas espiras. La cantidad de separaciones que posee el conmutador va a depender directamente de la cantidad de espiras que posee la armadura.

A partir de las figuras 2.1a y 2.1b se explica el principio de funcionamiento del motor. Este modelo corresponde al de la maquina giratoria más sencilla posible y si bien no es un modelo representativo de la realidad, su modo de funcionamiento es generalizable para modelos más



(a) (b)

Figura 2.1: Esquema de motor de una sola espira, extraído de [11]

complejos. Considerando 2.1a, se establece que si el rotor gira a una velocidad v, debido a la densidad de flujo magnético B presente en el estator, se induce una diferencia de potencial en la espira denominada "tensión inducida". Esta tensión se describe por la ecuación 2.1.

$$e_{ind} = (v \times B)l \tag{2.1}$$

Debido a la velocidad de la espira, se produce una variación en la cantidad de lineas de campo magnético que la atraviesan originando una cambio de flujo magnético. A causa de la forma que posee la espira y de como se encuentra ubicada dentro del flujo magnético establecido, la tensión solo se induce en los segmentos cd y ab, siendo cero en los segmentos restantes. Cuando la espira se ubica de forma vertical o a 180°, la polaridad de la tensión inducida se invierte. Este modo de funcionamiento corresponde al generador, es decir cuando la energía mecánica se transforma en energía eléctrica.

Analizando 2.1b, mientras la llave permanece cerrada (t=0), la diferencia de potencial entre los bornes de la espira es cero y no se genera un movimiento de esta. Cuando el interruptor se cierra, comienza a circular una corriente sobre la espira debido a la diferencia de potencial aplicada. Ya que el conductor está dentro de lineas de campo magnético, se ejerce una fuerza sobre esta que se describe en la siguiente ecuación:

$$F = i(l \times B) \tag{2.2}$$

Siendo i la corriente generada por la diferencia de potencial y l la longitud de la espira. Esta fuerza inducida F posee una magnitud no nula en los segmentos ab y cd, siendo igual a cero en los segmentos restantes. A causa de esto, se genera un par inducido, descripto por la ecuación 2.3 y la espira comenzara a girar. La distancia que existe desde el centro de rotación a la espira se expresa mediante r y θ el ángulo que existe entre F y r.

$$\tau = rF\cos(\theta) \tag{2.3}$$

Al orientarse la espira verticalmente, gracias a los conmutadores, la polaridad en las espiras se invierte y la corriente mantiene su sentido de circulación. Esto asegura que el par inducido siempre tendrá el mismo sentido manteniendo la misma dirección de giro. Este modo de funcionamiento corresponde al motor. Cuando la maquina funciona en este modo, sobre la espira además se tiene un efecto similar al generador pero la diferencia de potencial generado se opone a la causa que lo genera. Este tensión se denomina contra fem inducida.

2.1.2. Modelo del motor de CC

Para el análisis, se consideró el modelo simplificado del motor propuesto en la figura 2.2. Este consiste de una resistencia y una bobina que representan la resistividad y la inductancia presente en los devanados del rotor. Además, se tiene una fuente electromotriz proporcional al flujo magnético y a la velocidad con la que gira el motor y un eje de masa despreciable al cual se acopla una carga. Esta representa al rotor, el cual tiene asociado un momento de inercia J_m y un coeficiente de fricción B_m [6] [38] [18].

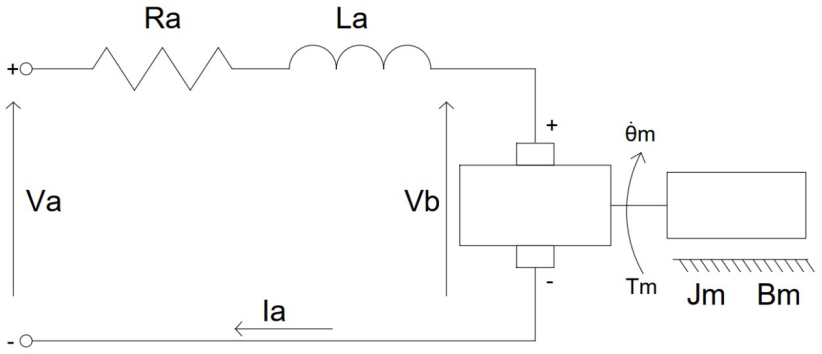


Figura 2.2: Modelo simplificado de un motor de CC.

Este esquema se subdivide en dos partes, la eléctrica que describe el comportamiento del circuito inducido y la mecánica la cual modela el movimiento del rotor. A partir de la Ley de Kirchoff y la segunda Ley de Newton para movimiento rotacional, de la anterior figura se obtiene:

$$v_a(t) = R_a i_a(t) + L_a \dot{i_a}(t) + v_b(t) \tag{2.4}$$

$$\tau_m(t) = J_m \ddot{\theta}_m(t) + \tau_{f,m}(t) \tag{2.5}$$

Donde:
τ_m: par desarrollado por el motor [Nm]
i_a: corriente de armadura [A]
R_a: resistencia de los devanados del rotor [Ω]
L_a: Inductancia de los devanados del rotor [mH]
v_a: tensión en bornes del motor [V]
v_b: tensión electromotriz inducida [V]
τ_{fm}: par de fricción del motor [Nm]

Teniendo en cuenta que el flujo magnético es constante, el par electromagnético generado es proporcional a la corriente de armadura presente en el rotor. Esta relación esta dada por 2.6. El movimiento generado por el par inducido genera una diferencia de potencial contra electromotriz la cual es proporcional a la velocidad angular a la que gira el rotor. Esta relación se muestra en 2.7.

$$\tau_m = k_d i_a(t) \tag{2.6}$$

$$v_b = k_f \dot{\theta}_m(t) \tag{2.7}$$

Reemplazando 2.6 y 2.7 en 2.4 y 2.5 y considerando solo la fricción viscosa como parte del par de fricción del motor se obtiene:

$$v_a(t) = R_a i_a(t) + L_a \dot{i_a}(t) + k_f \dot{\theta}_m(t) \tag{2.8}$$

$$k_d i_a(t) = J_m \ddot{\theta}_m(t) + B_m \dot{\theta}_m(t) \tag{2.9}$$

Donde:
k_d: constante de par del motor [Nm/A]
k_f: constante de contrafem [Vseg/rad]
J_m: momento de inercia del rotor [Kgm^2]
B_m: coeficiente de fricción dinámica del rotor [Nms/rad]

Aplicando la transformada de Laplace a las ecuaciones anteriores y considerando las condiciones iniciales nulas, se obtiene:

$$V_a(s) = R_a I_a(s) + sL_a I_a(s) + k_f \dot{\theta}_m(s) \tag{2.10}$$

$$k_d I_a(s) = sJ_m \dot{\theta}_m(s) + B_m \dot{\theta}_m(s) \tag{2.11}$$

De las ecuaciones 2.10 y 2.11 se obtiene la función de transferencia de tiempo continuo que describe la velocidad que genera el motor en función de la señal de tensión de entrada.

$$\frac{\dot{\theta}_m(s)}{V_a(s)} = \frac{k_d}{(L_a s + R_a)(J_m s + B_m) + k_f k_d} \tag{2.12}$$

Con las consideraciones realizadas, el modelo matemático que describe el comportamiento del motor es de segundo orden, como se observa en 2.12. Debido a su orden, tiene asociadas dos constantes temporales que afectan su dinámica. La primera de estas se asocia al tiempo de respuesta que posee el subsistema eléctrico para responder ante cambios producidos en la entrada, mientras que la segunda (constante mecánica) esta relacionada con el tiempo de respuesta asociado al subsistema mecánico ante las mismas variaciones de la entrada. En la gran mayoría de los motores de este tipo, la constante de tiempo eléctrico es despreciable predominando la constante mecánica.

2.1.3. Modelado del motor de CC con carga acoplada por reductor

Se consideró que el sistema se compone por el motor, un reductor que se encuentra conectado al motor, una rueda que se conecta en el otro extremo del reductor a través de un eje de masa despreciable y un enconder incremental cuya función es medir la velocidad de rotación. La función

del moto-reductor es reducir la velocidad a la que gira el eje del motor en un factor de N veces mediante un sistemas de engranajes. Esto facilita el uso de los motores en bajas velocidades obteniéndose un mayor par en la salida del reductor. Los reductores además poseen una eficiencia η. En la figura 2.3 se observa el esquema eléctrico/mecánico [23].

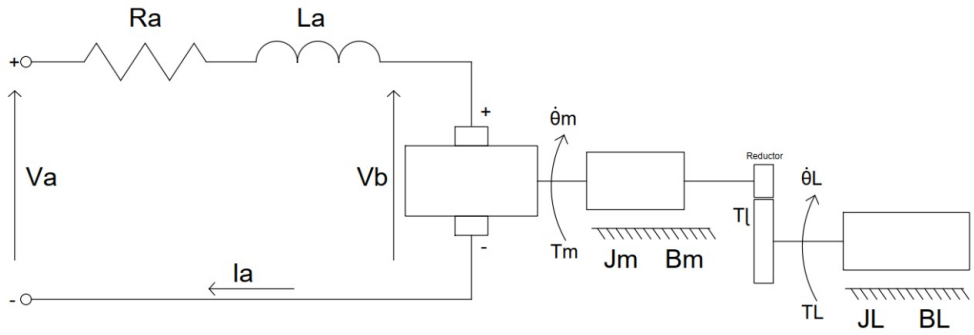


Figura 2.3: Modelo simplificado de un motor de CC con carga.

Debido al modo de funcionamiento del reductor se determina que:

$$r\dot{\theta}_l(t) = \dot{\theta}_L(t) \tag{2.13}$$

Donde $r = 1/N$.

Despreciando las perdidas que se producen en el reductor, se establece que la potencia que se genera en el rotor es la misma que se transmite a la carga, por lo que se obtiene la siguiente relación:

$$\tau_l\dot{\theta}_l(t) = \tau_L\dot{\theta}_L(t) \tag{2.14}$$

Donde
$\tau_l(t)$: par de la carga visto desde el eje del motor
$\tau_L(t)$: par de la carga
$\dot{\theta}_l(t)$: velocidad angular de la carga visto desde el eje del motor
$\dot{\theta}_L(t)$: velocidad de la carga

Utilizando la relación de 2.13 en 2.14 se tiene:

$$\tau_l(t) = \tau_L(t)r \tag{2.15}$$

A partir de la relación obtenida en 2.15 y considerando que existe un reductor junto con una rueda acoplado al eje del motor, la ecuación 2.9 se reescribe de la siguiente manera

$$\tau_m(t) = J_m\ddot{\theta}_m(t) + B_m\dot{\theta}_m + \tau_l(t) \tag{2.16}$$

Para la expresión del par generado en la carga se considera solo la fricción viscosa.

$$\tau_L(t) = J_L\ddot{\theta}_L(t) + B_L\dot{\theta}_L(t) \tag{2.17}$$

Reemplazando 2.17 y 2.13 en 2.16 y reagrupando se obtiene:

$$\tau_m(t) = \frac{J_m + J_l r^2}{r}\ddot{\theta}_L(t) + \frac{B_m + B_l r^2}{r}\dot{\theta}_L(t) \tag{2.18}$$

Considerando 2.6 y aplicando la transformada de Laplace a la ecuación 2.18 se tiene:

$$k_f I_a(s) = \frac{J_o s + B_o}{r}\dot{\theta}_L(s) \tag{2.19}$$

Donde
$J_o : J_m + r^2 J_L$ representa el momento de inercia equivalente del sistema
$B_o : B_m + r^2 B_L$ representa el coeficiente de fricción equivalente del sistema

De 2.11 se conoce $I_a(s)$. Luego se reemplaza en 2.19. Distribuyendo y reagrupando se tiene:

$$\frac{\dot{\theta}_L(s)}{V_a(s)} = \frac{k_d r}{(L_a s + R_a)(J_o s + B_o) + k_d k_f} \tag{2.20}$$

Como se observa de 2.20, la función de transferencia es del mismo orden que 2.12. El hecho de agregar una carga y vincularla al motor a través de un reductor, no afecta conceptualmente el significado de cada uno de los parámetro del modelo. Cuando se considera el reductor, aparece una constante que afecta la velocidad de estado estacionario. En el caso de la carga, se afecta la dinámica del motor. Mencionado esto, se decidió que la futura identificación se realice a partir de un motor equivalente, el cual se compone por el motor, el reductor y la rueda. En la ecuación 2.21 se observa la función de transferencia mencionada.

$$G_E(s) = \frac{k_d}{(L_a s + R_a)(J s + B) + k_d k_f} \tag{2.21}$$

2.2. Análisis del comportamiento del sistema

En esta sección se hizo un análisis comparativo entre el funcionamiento que presenta el sistema cuando se alimenta con una señal continua frente al comportamiento que se obtuvo cuando este es alimentado con un ESC.

2.2.1. Verificación de la linealidad del sistema frente a señales continuas

Inicialmente se inyectaron señales de entrada continuas al sistema y se esperó a que este alcance su estado estacionario. Cuando esto ocurrió, se comenzó a almacenar muestras de velocidad por un periodo de treinta segundos para luego promediarlas. Las señales de entrada se generaron con una fuente de tensión continua, las cuales se aplicaron directamente en los bornes de entrada del sistema. En la figura 2.4 se observan los resultados que se obtienen.

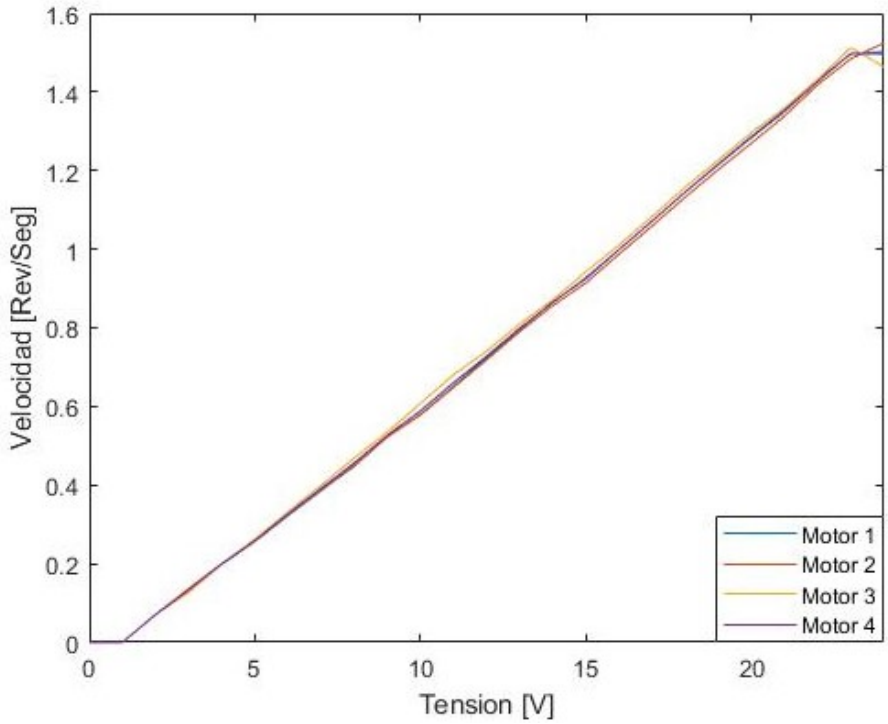


Figura 2.4: Relación de velocidad vs tensión en motores al emplear fuente de CC.

Como se observa en la figura 2.4 se tiene una relación lineal en casi todo el rango operativo del sistema. Para niveles de tensión inferiores a 2 V no se tiene respuesta en la salida. Esto genera una zona muerta en la que el motor no es operable. A continuación se realizó el mismo ensayo, con la diferencia de el sistema se alimentó mediante un ESC.

2.2.2. Verificación de la linealidad del sistema con ESC

Se procedió a verificar la respuesta del sistema en estado estacionario cuando la señal de entrada es de tipo PWM, la cual se obtuvo desde el variador de velocidad. Este tipo de señales se utiliza cuando se requiere variar la velocidad de los motores empleando microcontroladores. El ESC se encarga de adaptar señales de control a señales de potencia operables para el motor[26]. En la figura 2.5 se observa el resultado de la prueba. Dentro del microcontrolador existe un registro que se denomina CCR, el cual indica el ciclo útil de trabajo de la señal PWM entre cero y un valor limite, que en este caso es 10000 (siendo 0 el 0 % y 10000 el 100 % del tiempo en nivel alto de señal). Para el ensayo se incrementó el CCR gradualmente, tomando pasos de 100 desde 0 a 4000 y luego de 500 hasta 10000, obteniéndose la velocidad en estado estacionario en cada valor CCR. Cuando se alcanzó el estado estacionario se comienzó a almacenar las muestras por treinta segundos para luego promediarlas.

A diferencia del caso que se muestra en la figura 2.4, cuando la alimentación del motor proviene del ESC no se obtiene una relación lineal, como se observa en la figura 2.5. Al igual que en la figura 2.4, para valores bajos de CCR se tiene una zona muerta y a medida que aumenta este parámetro, aumentan las alinealidades.

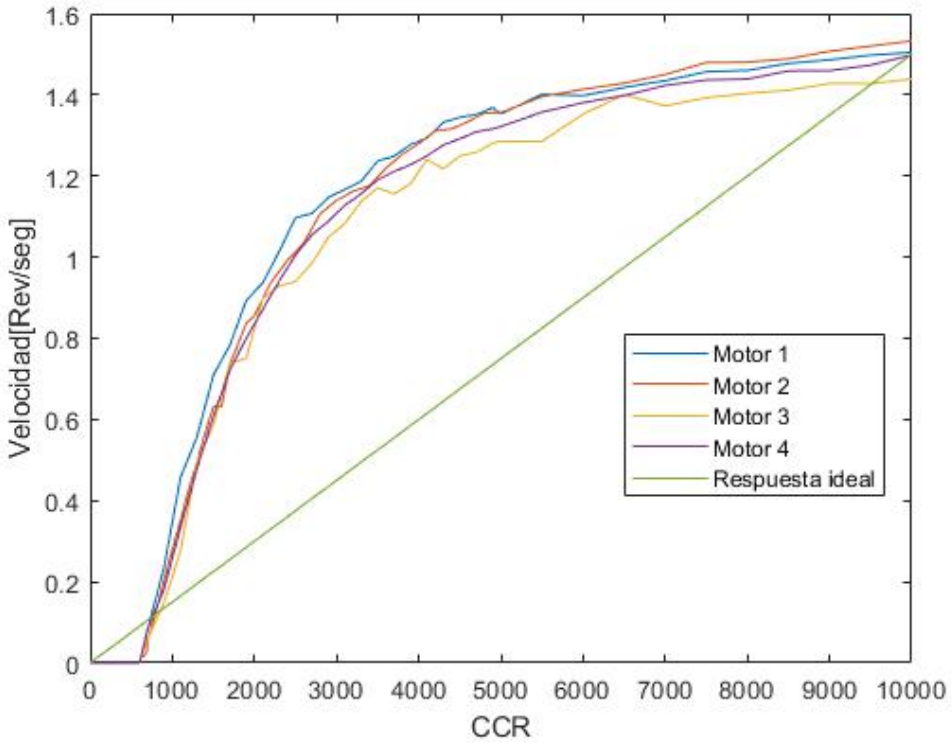


Figura 2.5: Relación de velocidad vs tensión en motores al emplear un ESC.

Este comportamiento no deseado tiene origen en el modo de funcionamiento del ESC particular que se utilizó. Como se observa de 2.6a, este sería el comportamiento ideal. Al colocar las puntas del osciloscopio en la salida del variador de velocidad y establecer distintos valores de CCR, se obtuvo un comportamiento completamente distinto. En las figuras 2.6b, 2.6c y 2.6d se observa esto. En los semiciclos positivos, la señal aplicada y la obtenida coinciden, difiriendo solo en su escala. En los semiciclos negativos, este tipo de ESC no pone en cero su salida. Esto genera una tensión promedio en bornes del motor distinta de cero.

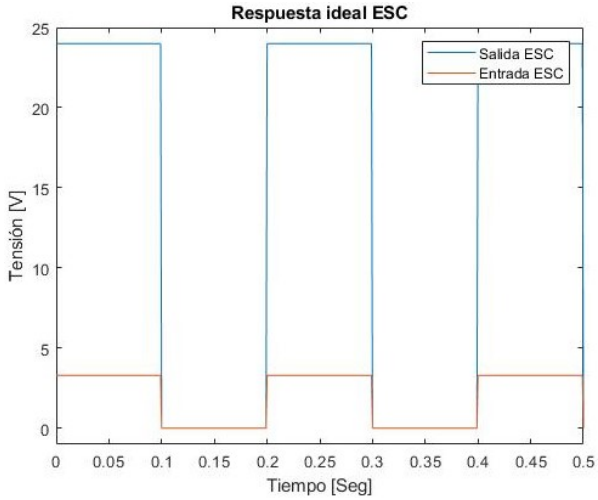
Inicialmente este ensayo, se planteó bajo la suposición de que el promedio de la PWM vista en bornes equivale a la tensión continua del gráfico 2.4. Al notar que la tensión del tiempo de apagado es distinta de cero, lo mencionado no se cumple y la relación deja de ser lineal. Para reducir este comportamiento no deseado, se diseñó un algoritmo que reduce la alinealidad existente y se trata en el capitulo 6.

2.3. Control del sistema

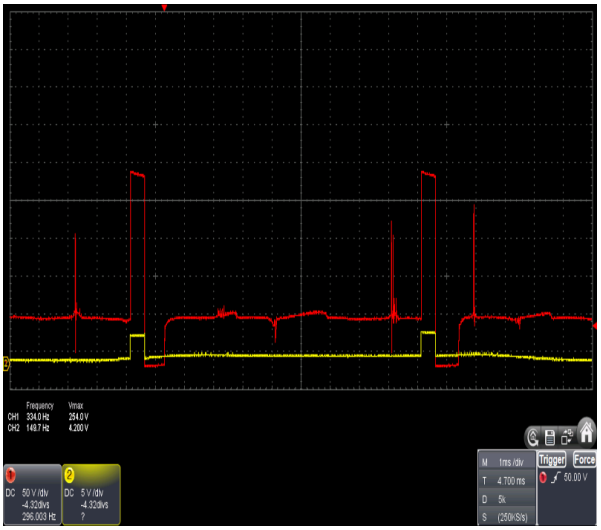
Existen numerosas técnicas de control y cada una presenta distintas ventajas. En este trabajo se desarrolló la que involucra la implementación de un controlador PID enfocado a la velocidad de giro del motor.

2.3.1. Control PID

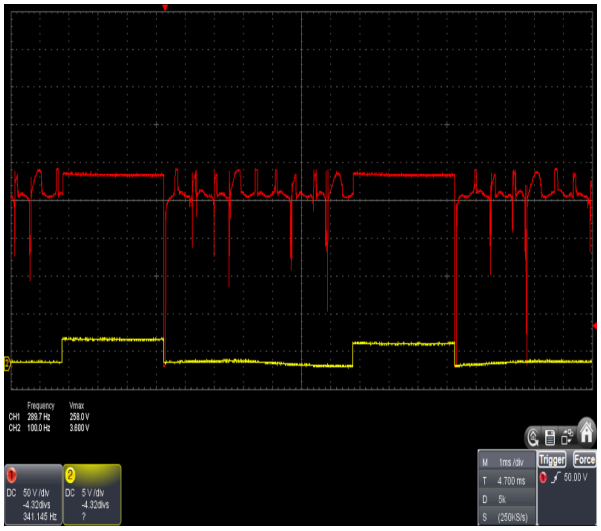
Uno de los controladores más utilizados en el control de procesos es el PID [27]. Esto se debe en gran parte a su sencilla configuración para ser utilizado. En la figura 2.7 se observa un diagrama el bloques de un sistema de lazo cerrado que implementa este controlador. Esta configuración, que emplea realimentación negativa, permite que la variable de interés permanezca cerca del valor de referencia establecido disminuyendo y/o eliminando variaciones que pueden generarse



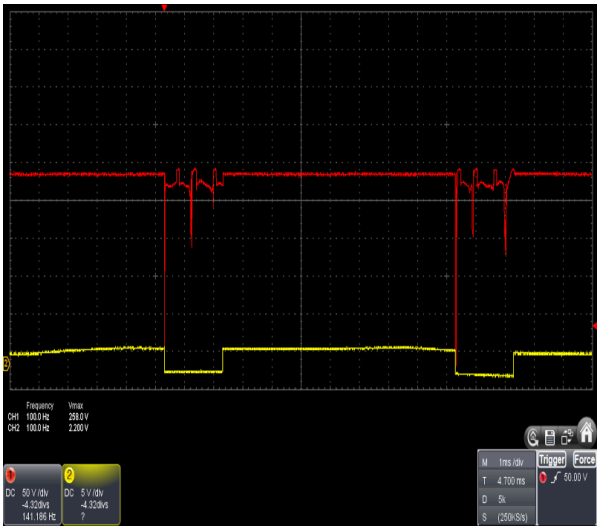
(a) Respuesta ideal del ESC.



(b) Respuesta para CCR = 500. Curva roja: salida ESC. Curva amarilla: salida microcontrolador.



(c) Respuesta para CCR = 3500. Curva roja: salida ESC. Curva amarilla: salida microcontrolador.



(d) Respuesta para CCR = 8000. Curva roja: salida ESC. Curva amarilla: salida microcontrolador.

Figura 2.6: Respuesta ideal y real obtenida del ESC para CCR=[300,3500,8000].

por perturbaciones ocasionadas en la variable controlada o cambios en la función de transferencia de la planta. Se denomina realimentación negativa debido a que la acción de control actúa sobre la entrada en sentido opuesto al cambio que se produce en la salida. Es decir que si por ejemplo el valor obtenido en la salida del sistema está por encima del valor de referencia, el control disminuye el valor de la entrada para lograr aproximar a la salida el valor que se busca. Este circuito de control no es capaz de detectar qué tipo de perturbación se está dando, sino que solo trata de mantener la variable de interés controlada y compensar cualquier perturbación. La precisión con la que se realiza esto último se denomina precisión del lazo de control. Otra característica del mismo, es el rechazo de perturbaciones, que representa la capacidad del lazo para no verse afectado por estas. El error de actuación que ingresa al controlador, se compone por la diferencia entre la señal de referencia deseada y la señal que se realimenta de la salida del sistema. Esta señal realimentada puede ser la propia salida, en el caso de que sea comparable con la señal de referencia, o una señal equivalente que se obtiene a partir de un sensor y que hace posible la comparación de señales. Como su nombre lo indica, el bloque actuador es el encargado de aplicar los cambios obtenidos del controlador una vez que estos pasan por el bloque linealizador

mencionado en 2.2.2.

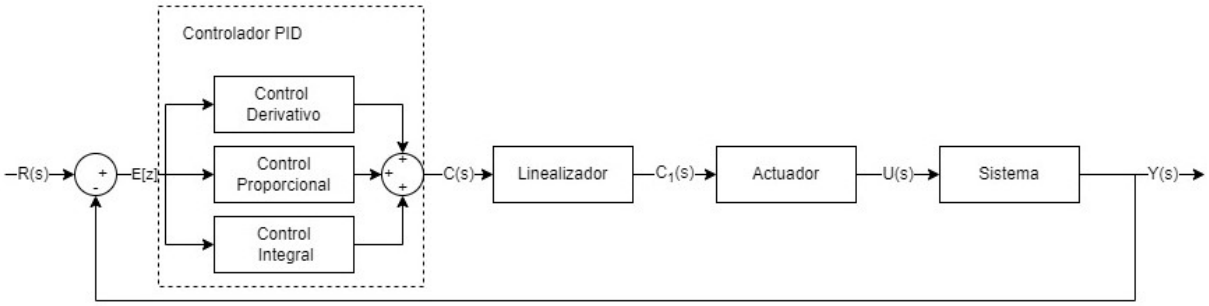


Figura 2.7: Diagrama en bloques del sistema de control.

De la figura 2.7, se tiene que $R(s)$ representa la referencia establecida, que en este caso es la velocidad de giro del rotor deseada. $E(s)$ es la señal de error que se obtiene de la comparación entre la velocidad referencia y la señal que se mide en la salida del sistema. Luego se tiene $C(s)$ que es la señal de control, esta se mapea por el bloque linealizador obteniéndose la señal $C_1(s)$, con la cual se comanda el ESC. Por último, $U(s)$ e $Y(s)$ son la entrada y la salida del sistema respectivamente y físicamente representan la tensión en bornes del motor y la velocidad del rotor. Respecto a los bloques que conforman el lazo, el actuador equivale al variador de velocidad y el bloque sistema se compone por el motor, el reductor y la rueda. El bloque controlador se conforma de tres etapas, proporcional, derivativo e integral. Cada una de las etapas posee una constante de proporcionalidad ajustable con la que se puede modificar el comportamiento individual de cada etapa y de esta manera lograr un desempeño distinto del controlador.

El control proporcional busca que el error en estado estacionario tienda a cero variando el valor de su constante de proporcionalidad. En la mayoría de los casos en donde se aplica solo este control, se tiene como resultado un error en estado estacionario, que para ser eliminado requiere una constante de proporcionalidad muy elevada. Hacer uso de valores elevados de dicha constante, brinda una respuesta más rápida del sistema pero compromete su estabilidad, ya que pueden presentarse oscilaciones y el sistema volverse inestable. Luego está el control integral, que tiene el propósito de disminuir y/o eliminar el error de estado estacionario debido a perturbaciones externas que el control proporcional es incapaz de eliminar. A medida que el tiempo transcurre, el error se va promediando y acumulando, se multiplica por una constante de proporcionalidad y se realimenta siendo cada vez mayor su efecto. En el control derivativo, tal como su nombre lo indica, el error se deriva y luego se multiplica por una constante de proporcionalidad. Este control actúa sobre la dinámica del error, permitiendo variar la velocidad con la que se ejerce la acción de control. De esta manera se evitan oscilaciones en el sistema, ya que el controlador puede reconocer cuando el error disminuye por lo que su acción de control disminuye. En la ecuación 2.22 se observa la función de transferencia para el PID. En la sección 6.5 se procede a la calibración virtual del controlador y su comparación con el desempeño real.

$$\frac{C(s)}{E(s)} = K_p + K_i/s + K_d s \tag{2.22}$$

Capítulo 3

Diseño e implementación del Hardware

En este capitulo se explica el proceso de diseño, fabricación e integración de todo el hardware necesario para que el vehículo quede en condiciones de funcionamiento.

En la figura 3.1 se muestra el vehículo sobre el cual se realizó el presente trabajo.

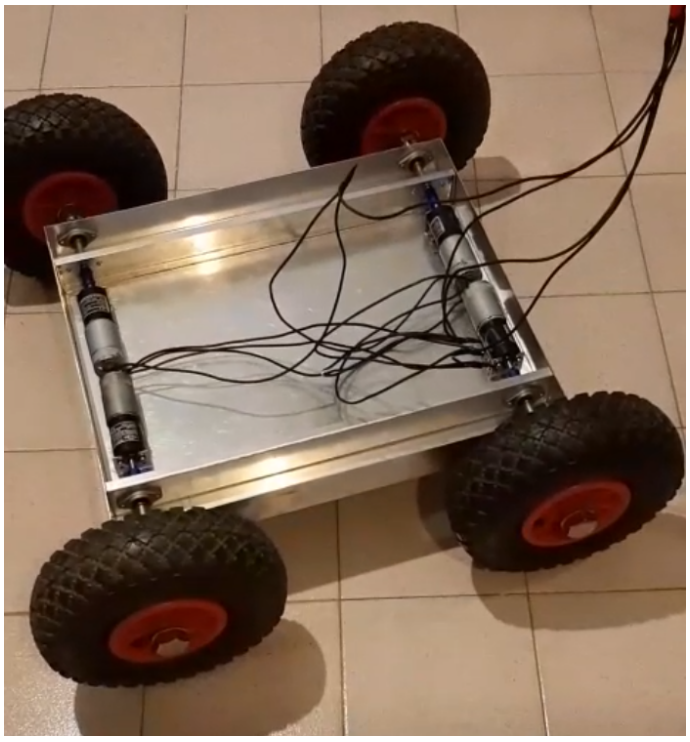


Figura 3.1: Estado inicial de recepción del vehículo.

El mismo cuenta con los siguientes componentes:

- Chasis de aluminio.
- Cuatro motor-reductores MR08D-024022-66 amurados al chasis.
- Cuatro ruedas con eje acoplado al motor-reductor.
- Cuatro encoders ópticos incrementales (amurados a chasis) con sus respectivos discos de 50 ranuras.
- Dos drivers PWM de velocidad (puente H doble) CC7A-160W [36] .
- Dos baterías[29] de 12 V con sus respectivos cargadores.

3.1. Electrónica necesaria para el control

3.1.1. Microcontrolador

Debe ser de 16 bits o superior, poseer memoria suficiente para poder implementar un sistema operativo de tiempo real, debe contar con capacidad de procesamiento necesaria para poder ejecutar todas las tareas de medición y control de los motores. También, debe disponer de periféricos adecuados para la comunicación con distintos sensores. Por lo mencionado, se optó por el STM32F103C8T6 [32] en su formato de placa de desarrollo conocida como "Blue Pill". Este microcontrolador de 32 bits cumple con los requisitos mínimos de procesamiento y memoria flash para implementar FreeRTOS y además cuenta con soporte de librerías HAL (Hardware Abstraction Layer) brindando fidelidad y control de errores a la hora de utilizar sus periféricos.

3.1.2. Sensores de velocidad

Los cuatro sensores de velocidad, uno por cada rueda, deben poder medir la velocidad en módulo/valor absoluto que desarrolla cada motor. Se decidió implementar los encoders incrementales, ya que cumplen con lo mencionado. Se descartó la opción de reemplazar los sensores existentes por encoders en cuadratura, luego de evaluar los cambios necesarios a nivel de hardware existente, tanto mecánico como electrónico.

3.1.3. Sensores de corriente

Los sensores deben medir la corriente que consume cada motor en ambos sentidos de giro y la corriente que circula sobre una resistencia conocida para indirectamente obtener el nivel de tensión de las baterías. Para realizar lo mencionado, se seleccionó el circuito integrado (CI) INA219 [17], el cual posee las ventajas de medir corriente de manera indirecta a través de la lectura de tensión sobre una resistencia shunt. Este sensor realiza su medición de manera diferencial, brindando la posibilidad de no asociar las masas de las etapas de potencia y control.

3.1.4. Puerto de comunicación

El puerto de comunicación, es la interfaz entre la placa de control y un dispositivo maestro, como se explica más adelante en la sección 5.1. Debe poseer los niveles lógicos de tensión y alimentación adecuados y tener en cuenta las velocidades de transmisión del puerto, utilizando buenas practicas de diseño. Por estos motivos y contemplando el contexto en el que se desarrolla el sistema, se implementó la comunicación sobre el periférico UART del microcontrolador, como se explica en la sección 5.3.

3.1.5. Actuador

Como se menciona en la sección 2.2.2 este se encarga de adaptar señales de control en otras de potencia operables para el motor. Cuenta con una interfaz, la cual posee alimentación, un pin para el control del módulo de la velocidad y dos pines adicionales. En el pin de control se emplea una señal de tipo PWM y los demás se encargan del control de sentido de giro en función de su nivel lógico. Lo mencionado es necesario para el manejo individual de cada uno de los motores. El actuador debe poder entregar la potencia necesaria para manejar los motores, por lo que se utilizó el que se dispone inicialmente (puente H doble CC7A-160W) debido a que cuenta con las prestaciones requeridas.

3.1.6. Etapa de alimentación

Esta etapa se encarga de convertir el nivel de tensión de la batería en los distintos niveles requeridos por las etapas mencionadas en esta sección. Debe ser capaz de suministrar la potencia

necesaria para cada una de las etapas. Se optó por utilizar conversores DC-DC reductores debido a su eficiencia para una implementación sobre un sistema con baterías. Uno de esos es un Buck (conversor reductor), destinado a regular el nivel de tensión de salida en 12 V, sin importar las variaciones de tensión en las baterías. El otro conversor utilizado, se conecta en cascada con el primero y se encarga de regular la tensión en 5 V necesaria para la etapa de control. Es necesario destacar que este último, cuenta con aislación eléctrica entre la tensión de entrada y la de salida.

3.2. Acondicionamiento mecánico

Previo a comenzar a trabajar sobre el vehículo, fue necesario realizar una de serie de modificaciones asociadas a su etapa mecánica, las cuales se mencionan a continuación.

3.2.1. Bujes

En las condiciones iniciales del robot, se notó que existe una gran rotación libre relativa entre la rueda y el eje del motor. Es decir que a pesar de que el eje del motor se encuentra en estado estacionario, la rueda puede rotar libremente dentro de cierto rango. Se llegó a la conclusión de que esto se debe a la diferencia existente entre los diámetros de las perforaciones en los bujes utilizados para el anclaje y el de los bulones, siendo esta diferencia mayor a 1 mm. Esto causa alinealidades, debido a que cuando se varía la velocidad del motor, existe un periodo de tiempo en el que éste gira a una velocidad distinta a la de la rueda. Por esto, se buscó la manera de minimizar este efecto.

Inicialmente se evaluó aumentar el diámetro de las perforaciones de los bujes, para luego roscar los ejes con una medida mayor a las perforaciones existentes. A modo de referencia, realizar esta reparación implica aumentar el diámetro de las perforaciones que se observan en la figura 3.3b. Esto no fue viable debido a que se compromete estructuralmente la integridad de ambos ejes. Por otro lado, se evaluó comprar ejes y tornillos pasantes. Luego de múltiples búsquedas y debido a que las medidas buscadas no eran tamaños estándar, no se consiguió un repuesto que se pueda comprar y resuelva directamente el problema encontrado. Por las razones mencionadas, se decidó fabricar los repuestos necesarios.

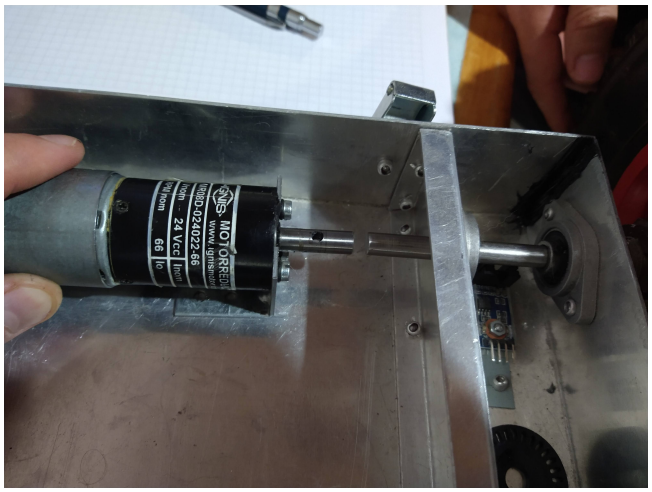
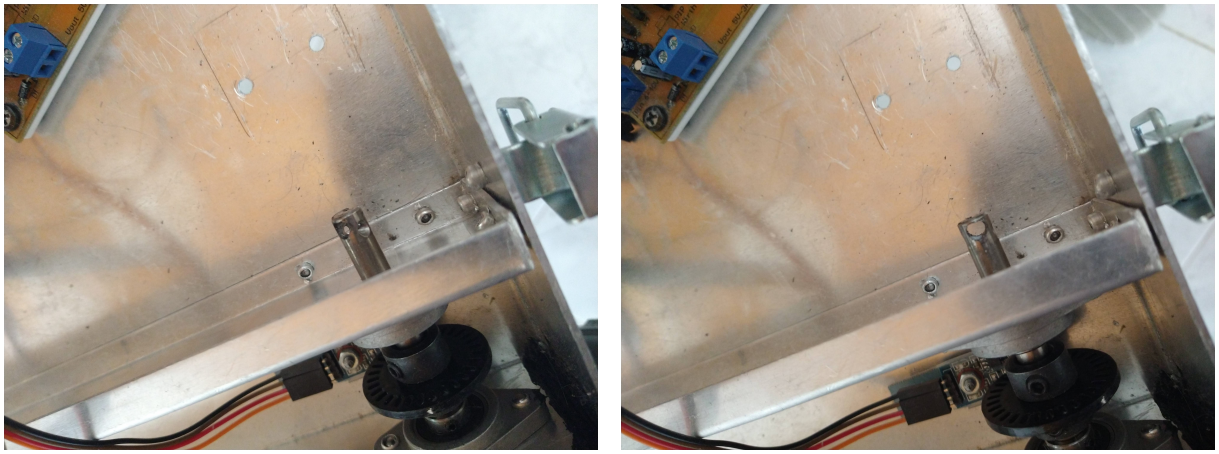


Figura 3.2: Ejes sin buje de sujeción.

Para iniciar la reparación, se retiró el buje existente del vehículo como se muestra en la figura 3.2 y luego se retiró el motor. Se fabricó el repuesto y luego se procedió al montaje de este, como se muestra en la figura 3.4.



(a) (b)

Figura 3.3: Eje de rueda con múltiples perforaciones.

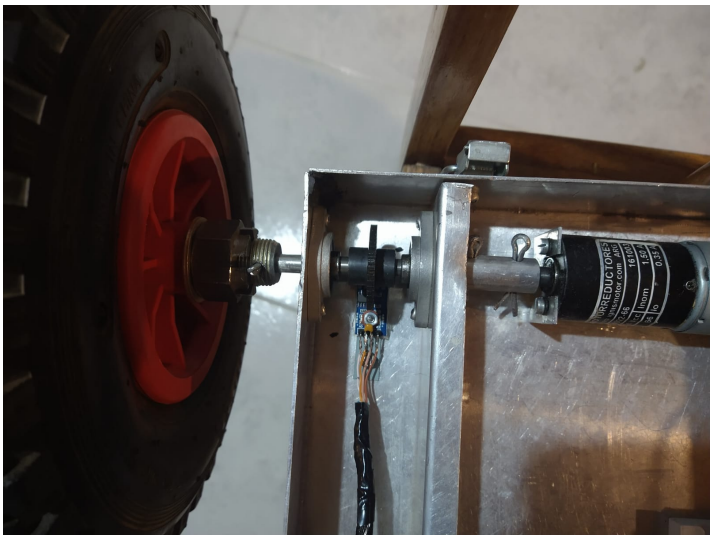


Figura 3.4: Montaje empleando nuevo eje y buje.

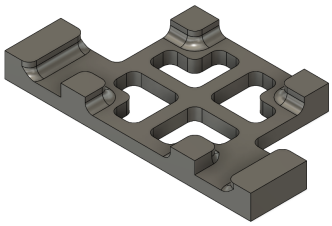
3.2.2. Diseño de adaptadores

Para la integración final en el vehículo, fue necesario posicionar de manera segura todas las placas electrónicas y sensores en el vehículo para minimizar la cantidad de partes móviles dentro de él. Para esto se diseñaron y fabricaron adaptadores.

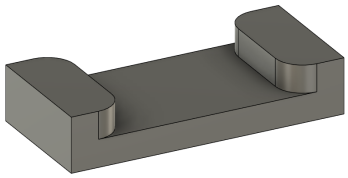
El hardware que necesita adaptador al chasis es el siguiente:

- Placa de control (1).
- Variadores de velocidad (2).
- Sensores de corriente (6).

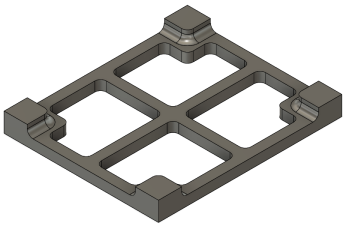
Se diseñó un adaptador para cada variador de velocidad, que cuenta con espacio para colocar un sensor de corriente a cada lado (el de cada rueda) como se muestra en la figura 3.5a. Luego el adaptador de la placa de control y por último otro adaptador separado para los dos sensores de corriente restantes. Estos se pueden ver en las figuras 3.5c y 3.5b respectivamente.



(a) Soporte para ESC y sensores INA219.



(b) Soporte para sensores INA219.



(c) Soporte para placa de control.

Figura 3.5: Soportes diseñados para componentes.

3.3. Diseño del hardware

En esta sección se detalla el proceso de diseño y fabricación de una placa PCB sobre la que se integra la gran mayoría de los componentes electrónicos a emplear.

3.3.1. Diseño del esquemático

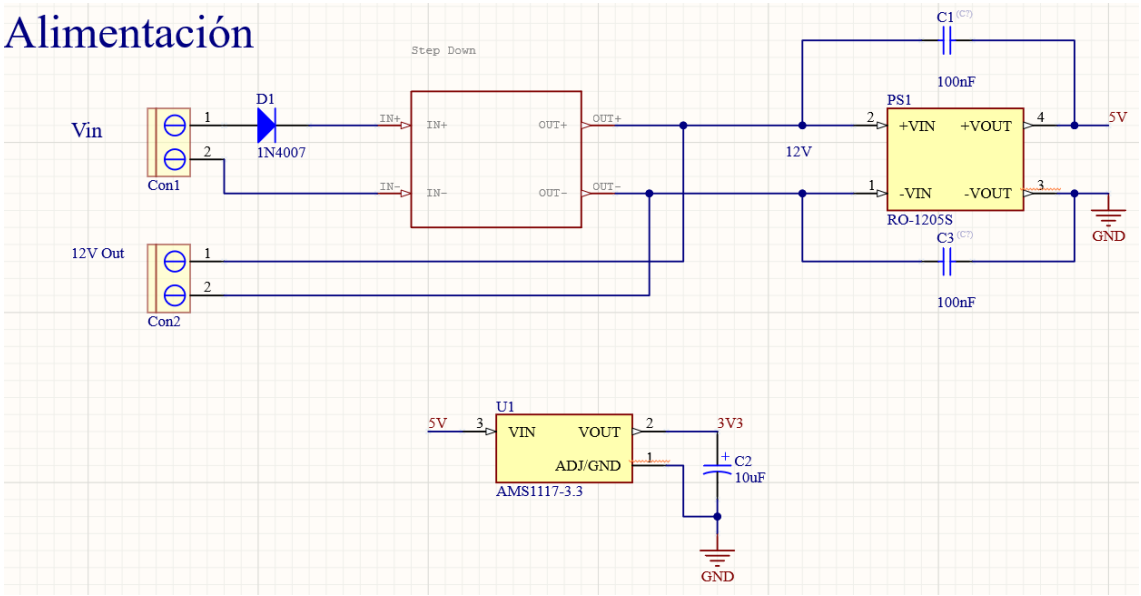


Figura 3.6: Esquema de etapa de potencia.

La etapa de potencia se muestra en la figura 3.6. A esta se ingresa por la bornera Con1 con el nivel de tensión de las baterías, luego de pasar por un diodo rectificador utilizado como protección contra polaridad inversa. La entrada rectificada ingresa a un conversor DC-DC reductor basado en el CI LM2596 [30], que fija en su salida 12 V y puede entregar una corriente de hasta 3 A. Como el conversor puede entregar una potencia mucho mayor a la necesaria, se coloca una bornera Con2 con 12 V de salida para que pueda ser usada por otro dispositivo del vehículo. Toda la

lógica y control, como requisito, debe estar aislada eléctricamente de la etapa de potencia, por lo que se implementó el conversor DC-DC reductor RO-1205S. Este regula 12 V de entrada en 5 V de salida completamente aislados eléctricamente. Con este conversor, se energiza los circuitos de control que se encuentra en los ESC's. Finalmente para alimentar el microcontrolador y todos los sensores se optó por el regulador lineal AMS11173.3 [34] implementado sobre el bus de 5 V.

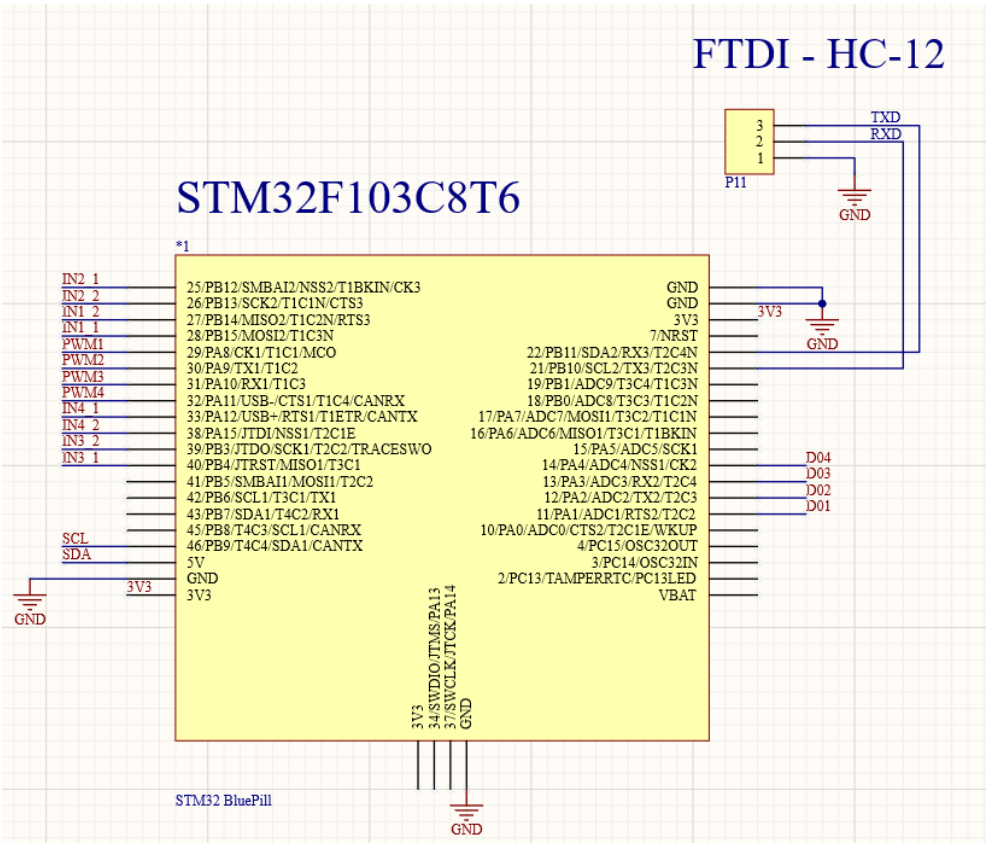


Figura 3.7: Diagrama de conexionado del microcontrolador.

El esquemático que contempla la implementación del microcontrolador, se observa en la figura 3.7. En este se puede ver las conexiones al microcontrolador, como lo son las entradas, salidas, alimentación y puerto dedicado a la comunicación. Los encoders incrementales utilizados para medir la velocidad se alimentaron con 3,3 V y se conectaron en entradas del microcontrolador capaces de generar interrupciones por cambio de flanco ascendente. El objetivo es detectar las interrupciones del haz de luz ocasionadas por el disco ranurado. Además, a modo de filtro pasa bajos para evitar rebotes en la medición, se colocó un capacitor de 100 nF entre GND y la salida digital del sensor. Esta última modificación se implementó sobre el propio encoder incremental. Finalmente, para su fácil conexión, se disponen en conectores los 4 pines necesarios por cada encoder como se muestra en la figura 3.8.

Los sensores de corriente se alimentaron con 3,3 V y se conectaron las líneas de comunicación I2C al microcontrolador en los pines correspondientes al periférico. La interfaz entre el sensor y la placa se realizó mediante un conector de cuatro pines como se muestra en la figura 3.9 y se conectaron todos en paralelo, ya que se comparte el bus de datos.

Los actuadores de los motores (puentes H dobles) poseen alimentación de 0 a 5 V y cuentan con 3 pines. Su principio de funcionamiento es el siguiente, en el primero se aplica una señal PWM que es la que se ve en bornes del motor y con los otros 2 pines restantes, se determina el sentido de giro según su estado lógico.

El fabricante define que la interfaz funciona con niveles de tensión de 3 a 6 V, pero luego de comprobar que con un nivel lógico de 3,3 V el modulo no funciona, se decidió colocar un

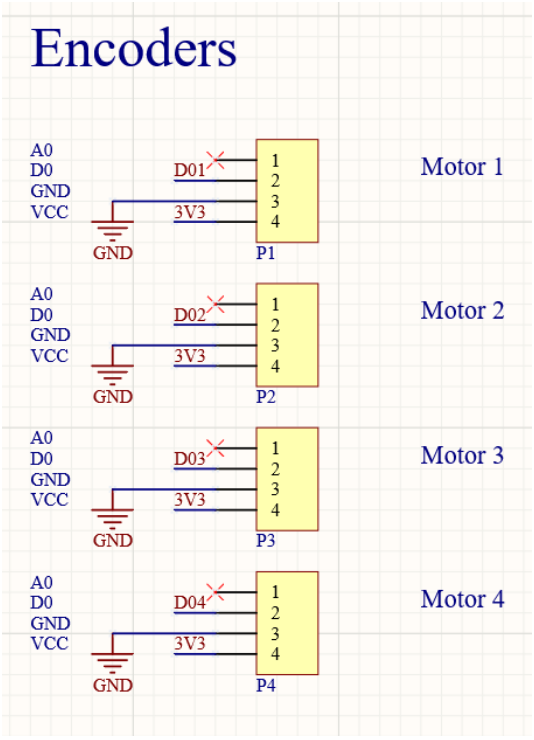


Figura 3.8: Esquema de conectores para enconders incrementales.

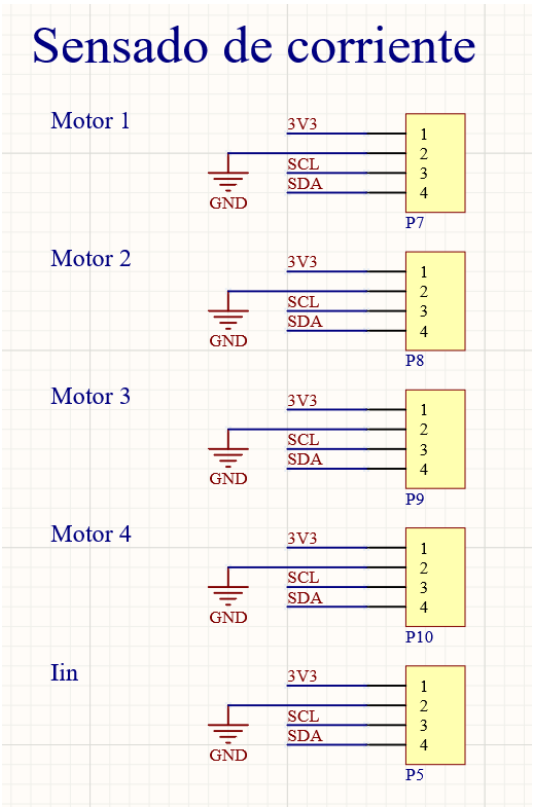


Figura 3.9: Esquema de conectores para sensores de corriente.

conversor de nivel lógico de 3,3 V a 5 V para lograr manejar los motores con el microcontrolador como se muestra en la figura 3.10. Para evitar errores de conexionado, se colocó un zócalo IDC de 2x5 en la placa.

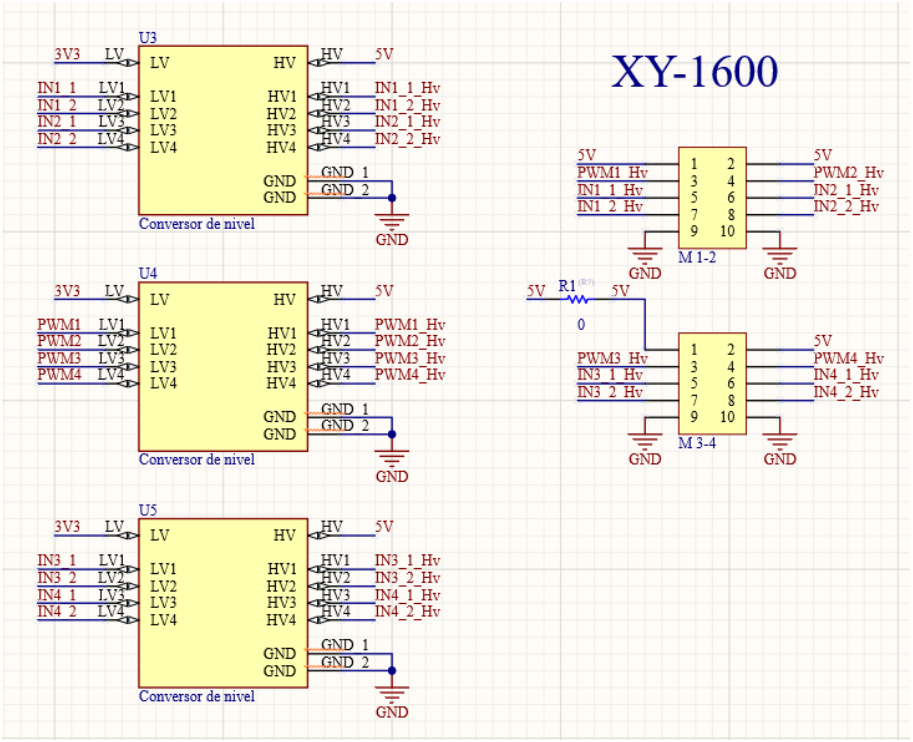


Figura 3.10: Esquema de conectores para ESC.

3.3.2. Consideraciones de diseño

En esta sección se describen las consideraciones necesarias que se tuvieron en cuenta a la hora de realizar el diseño del hardware definitivo que se implementa sobre el vehículo.

En una primera instancia, se realizó una placa PCB que cumplía con las funcionalidades mencionadas en la sección 3.1 y que contaba con las siguientes diferencias:

- Los sensores de corrientes utilizados son los ACS712 [7].
- La etapa de alimentación se implementa únicamente con un conversor DC-DC (no aislado).

En un principio se optó por utilizar el sensor de corriente ACS712, el cual posee una interfaz analógica. A la hora de adquirirlos, se lo ensayó para contrastar su funcionamiento. Como resultado se obtuvo que la pendiente que relaciona la tensión del sensor contra la corriente medida poseía un error superior al 50 % respecto a la que propone el fabricante en la hoja de datos en cada uno de los sensores. Una de las opciones luego de ensayarlos fue realizar una corrección por software, pero como resulta muy especifico a cada sensor, en caso de requerir reemplazarlo se vuelve necesario cambiar el software. Como esto último no cumple con la robustez y facilidad de reparación buscada en el diseño del robot, se decidió cambiar el sensor por el INA219. Esta modificación de sensores, involucró un cambio en la interfaz con la que se realiza la medición, debido a que el INA219 se comunica a través de I2C.

Por otro lado, al realizar mediciones de velocidad, se detectaron grandes oscilaciones e inconsistencias respecto a los valores reales. Además, se detectaban velocidades distintas de cero con sensores cuyo motor asociado se encuentra detenido. Tras examinar las posibles causas, se encontró que esto se debía a ruido electromagnético generado por la conmutación en los colectores de los motores cuando estos se encuentran en funcionamiento. En la figura 3.12a se registraron grandes variaciones de tensión en uno de los pines de interrupción por flanco asociado a un encoder. Estas variaciones no eran generadas por movimiento en la rueda, ya que, esta se encontraba detenida. A modo de ensayo, se energizaron con distintas fuentes de energía ambas etapas y se

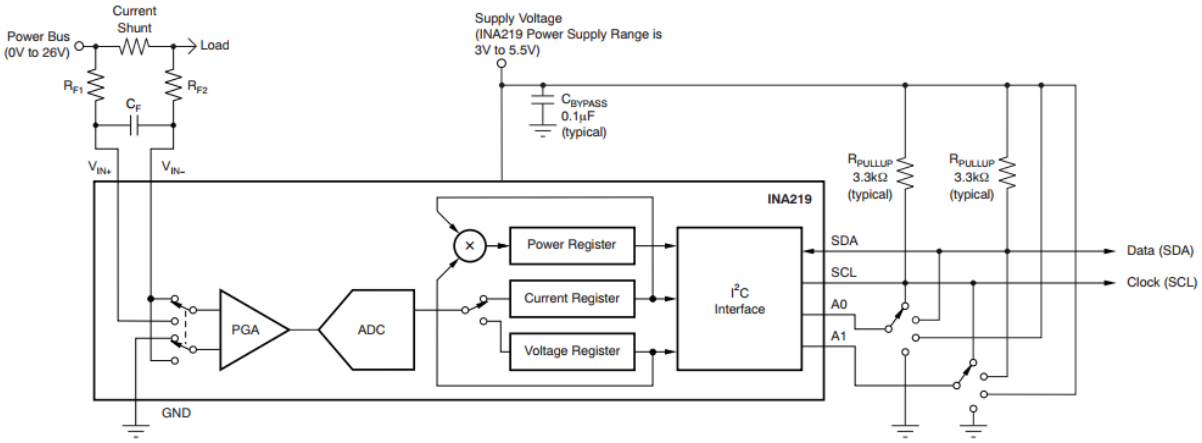
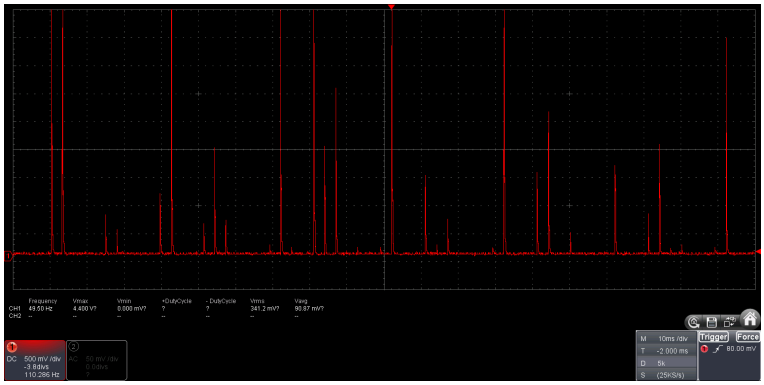
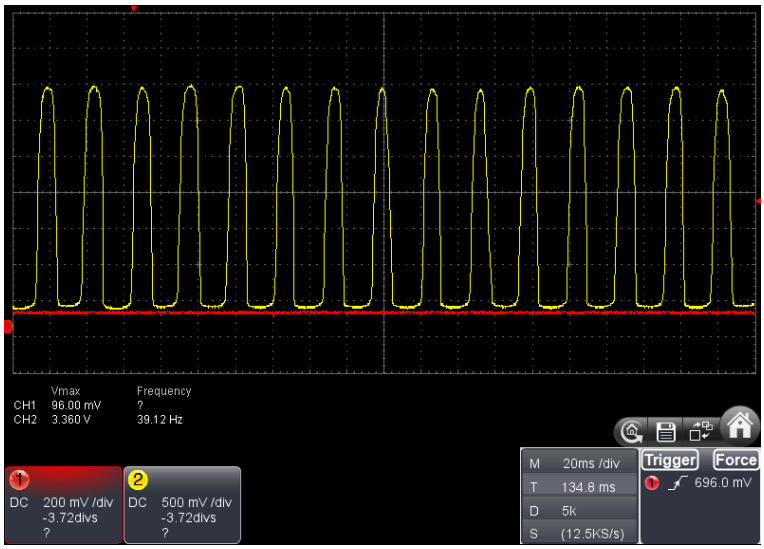


Figura 3.11: Diagrama en bloques de funcionamiento del sensor INA219, extraído de su hoja de datos.

apreció que este fenómeno no ocurría. Por esto, surge la necesidad de desvincular eléctricamente dichas etapas.



(a) Ruido inducido sobre el pin de señal de un encoder incremental, cuyo motor se encuentra detenido y otro de los motores gira a velocidad máxima.



(b) En esta figura se muestran las señales de dos motores, uno girando a velocidad nominal (amarillo) y otro estático (rojo). Luego de aplicar el filtro DC-DC, el ruido inducido desaparece completamente.

Figura 3.12

Por todo lo mencionado, se encontraron motivos suficientes para un rediseño de la versión inicial de la placa de control. En la nueva versión, se implementó la nueva interfaz I2C para los sensores de corriente y un conversor reductor R0-1205S [28]. Este posibilitó la aislación eléctrica entre la etapa de potencia y la de control empleando una única fuente de alimentación.

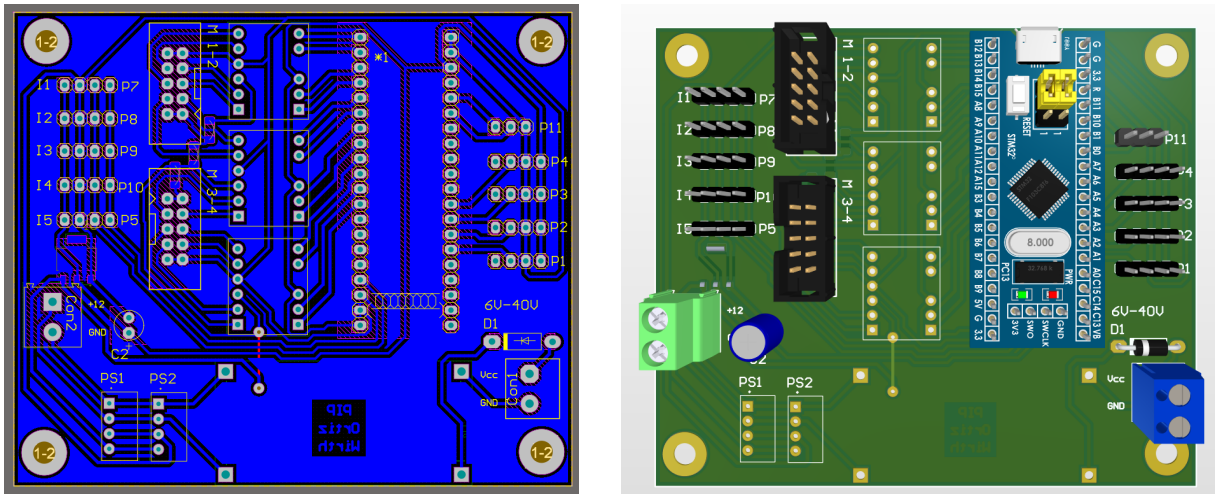
3.3.3. Diseño y fabricación del pcb

Una vez diseñado el esquemático se procedió al diseño de layout de componentes y ruteo de pistas. Se tuvo en consideración las capacidades de fabricación a la hora del diseño, las cuales son:

- Fabricación a mano, método de planchado para transferir tóner desde una imagen negativa del circuito en una placa con lámina de cobre.
- Utilización de placa de fibra con lámina de cobre en un solo lado.
- Perforación y soldadura a mano.
- Eliminación de exceso de cobre mediante un baño de cloruro férrico.

Teniendo en cuenta lo mencionado anteriormente, los aspectos a considerar fueron el espaciado entre pistas y su grosor, uso de vías, tamaño de los componentes, etc.

Se procedió a posicionar los componentes de manera estratégica en la placa con el objetivo de facilitar el conexionado entre ellos y luego rutearlos entre si, como se muestra en la figura 3.13a. Se obtiene el modelo que se muestra en la figura 3.13b luego del diseño.



(a) Capa inferior de placa de control.

(b) Esquema 3D de placa de control desarrollada.

Figura 3.13

Para la elaboración del PCB, se recurrió al método de fabricación manual. Este consta de imprimir una imagen monocromática de la figura 3.13a en papel fotográfico, para luego transferirla sobre una placa de cobre utilizando calor. Una vez que el tóner se transfiere, se sumerge la placa en cloruro férrico con el objetivo de eliminar el cobre expuesto sobre esta, dejando solo el cobre utilizado para las pistas. Luego, se procedió a la perforación de la placa para colocar los elementos con formato THT (Through Hole Technology). Se posicionaron y soldaron los componentes en la placa, empezando por los que son mas pequeños (SMD y puentes), hasta los más voluminosos (conectores y demás). Luego, se realizaron mediciones de continuidad con multímetro previo a energizar la placa para el chequeo de las conexiones. En este punto ya se contaba con una placa fabricada y con niveles de tensión adecuados para utilizar todos los módulos. Se

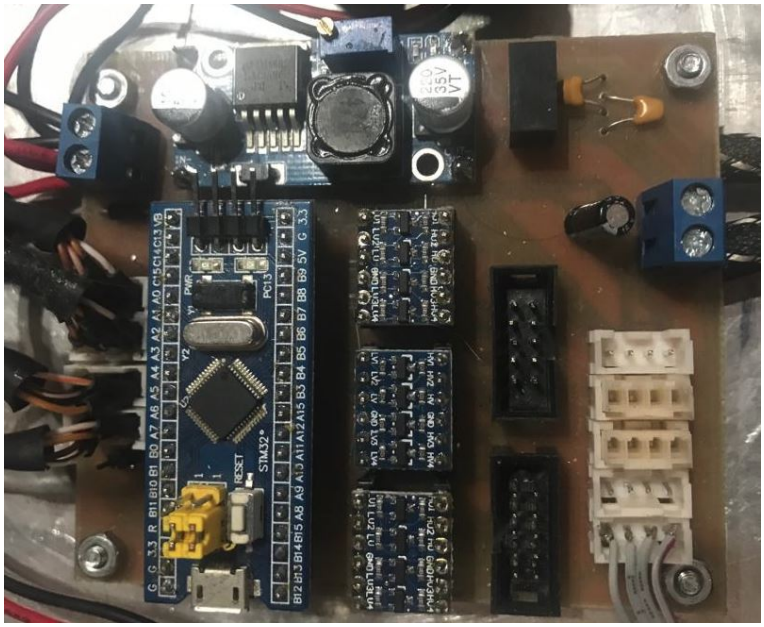


Figura 3.14: Versión final de placa de control.

procedió a ensayar las demás partes de la placa, funcionamiento de microcontrolador, encoders, sensores de corriente, puerto de comunicación e interfaz con variador de velocidad.

Finalmente se obtiene el producto terminado como se muestra en la figura 3.14. Luego se procedió a chequear con el osciloscopio los pines de salida digitales de los enconders de dos motores. Uno de ellos gira a velocidad nominal (amarillo), mientras que otro se encuentra estático (rojo). El resultado de esto se observa en la figura 3.12b. De la figura se ve que el motor en movimiento no induce ruido sobre el que permanece estático.

Capítulo 4

Implementación de software de control en FreeRTOS

En este capítulo se explica qué es RTOS y su principio de funcionamiento. Se aborda la arquitectura del software utilizado y las funcionalidades de cada una de sus tareas.

4.1. Sistemas operativos de tiempo real (RTOS)

4.1.1. ¿Qué es RTOS?

Existen al menos dos formas de diseñar un sistema embebido. Un primer modelo sencillo y muy utilizado, es cuando todas las funciones del programa están en un bucle infinito. En este modelo, las acciones en el bucle se ejecutan de manera secuencial, una después de otra. Este principio de funcionamiento es adecuado cuando no es necesario ejecutar múltiples tareas en concurrencia o no es prioritario ejecutar una función por sobre otra. A este concepto de diseño se conoce como superloop. Una segunda posibilidad, es modelar el sistema en tareas independientes, que se ejecutarán de manera concurrente. En este modelo, resulta útil el uso de un sistema operativo (SO), ya que es una herramienta de software que provee servicios de ejecución concurrente de tareas, y recursos de comunicación y sincronización entre las mismas. Dentro de esta categoría, existen los RTOS. Estos últimos se destacan por ser más simples y predecibles que los SO, motivo por el cual, se utilizó en este proyecto. Antes de pasar a los RTOS, se analizan los SO y sus diferencias con los de tiempo real.

Un sistema operativo [35] [31] es el software que gestiona los recursos de hardware, oculta los detalles de bajo nivel, y presenta a las aplicaciones un conjunto de servicios de alto nivel. Es decir, es el software que se encuentra entre el hardware y los programas de los usuarios del sistema. Su función principal es ejecutar programas y asignar recursos de acuerdo a ciertos criterios. Sus componentes principales son: un gestor de procesos, un gestor de memoria, un sistemas de archivos, drivers de E/S, un subsistema de comunicaciones (red) y protección.

Dentro de las funcionalidades que ofrece, se puede mencionar que un SO permite la ejecución de múltiples funciones en simultáneo y la posibilidad de establecer un orden de ejecución. Para esto se utiliza un concepto denominado proceso que permite ejecutar lógica de código que se ubica dentro de su dominio de manera independiente y en concurrencia con otros procesos. En la Figura 4.1.(a) se observa el comportamiento explicado para el super loop, mientras que en 4.1.(b) se puede ver el modo de proceder de los SO.

Si se trata de funcionalidades, un SO ofrece muchas más posibilidades que un RTOS. Los SO que se utilizan en PCs o servidores disponen de planificadores de CPU complejos, orientados a ser altamente justos en la repartición de recursos, empleando criterios complejos y en algunos casos dinámicos. Esto implica que sus criterios dependen de la demanda que soporta el sistema o la PC, y demuestra la mayor complejidad que manejan los SO frente a los RTOS.

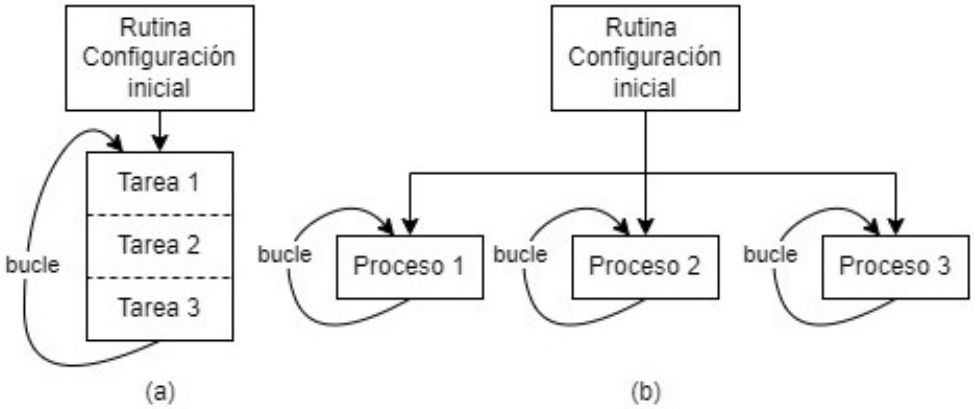


Figura 4.1: Ejecución de tareas con super loop (a) comparado con un sistema operativo (b).

Los RTOS [21] [5] tienen como principal característica permitir la ejecución concurrente de tareas de manera predecible. Esto simplifica su composición a un gestor de tareas apropiativo que funciona en base a prioridades y mecanismos para la sincronización y comunicación entre las mismas. Cabe aclarar, que se denomina tarea a los procesos dentro de los RTOS. El planificador de CPU se diseña para garantizar la ejecución de tareas, teniendo como referencia la prioridad y los plazos definidos para estas. Esto no implica necesariamente que las tareas se ejecutaran con mayor rapidez, sino que las tareas se gestionan de una manera adecuada reduciendo los retardos al mínimo. Esto proporciona un comportamiento predecible y estable de la ejecución del código. Asimismo, RTOS brinda funcionalidades que pueden ayudar a la gestión de la ejecución y las cuales se explican a continuación.

En la industria existen muchas posibilidades de sistemas operativos de tiempo real que brindan distintas posibilidades de desarrollo. Dentro de todo este espectro de posibilidades y por las características asociadas al proyecto se utilizó FreeRTOS. Este es uno de los RTOS de código abierto más utilizado en la industria debido a que no posee una gran curva de aprendizaje, cuenta con amplia documentación de consulta y puede ser implementado en el microcontrolador empleado [9].

4.1.2. Tareas

FreeRTOS gestiona la ejecución de varias tareas independientes de manera concurrente a través del planificador de CPU, en base a prioridades asignadas. Cada tarea es una subrutina en lenguaje C al que FreeRTOS le asigna su propia pila de memoria, y que compite por el procesador, en simultáneo, con el resto de las tareas. Generalmente, si el sistema final debe realizar dos o tres funciones diferentes, el equipo de desarrollo implementará cada funcionalidad del sistema dentro de su propia tarea. Por ejemplo, una tarea puede consistir en la lectura, almacenamiento y conversión de información obtenida a través de un conversor analógico-digital y otra en transmitir dicha información a través de un módulo UART.

En general las tareas pueden adquirir tres estados a lo largo del tiempo y estos son: listo para ejecutar, ejecutándose y bloqueada. Cuando la tarea está lista para ejecutarse, el planificador de CPU la coloca en estado de listo para ejecutarse, para luego chequear la prioridad que tiene asignada. En caso de que su prioridad sea mayor que la de la tarea que se encuentra ejecutando el CPU, ocurre un cambio de contexto y comienza la ejecución de la nueva tarea que estaba por entrar en estado de listo, adquiriendo el estado de ejecutándose. Si el procesador posee más de un núcleo, es posible ejecutar múltiples tareas al mismo tiempo, siendo el planificador de CPU el encargado de la selección.

Ejecutada la totalidad de la función, el planificador de CPU comienza la ejecución de otra

tarea o queda a la espera de hacerlo y la función adquiere el estado de bloqueo. En este estado, las tareas no pueden ser ejecutadas por el CPU hasta que un determinado evento se presente, como puede ser el vencimiento de un plazo de tiempo. Cuando esto sucede, la tarea adquiere nuevamente el estado de lista, esperando al planificador de CPU para volver a ser ejecutada.

Adicionalmente FreeRTOS ofrece una serie de funciones que permiten establecer un estado de suspensión en una tarea. Esto se diferencia del estado de bloqueo en que se puede pasar entre estados según lo requiera el usuario y no hay que esperar un evento. En la figura 4.2 se observan los estados mencionados.

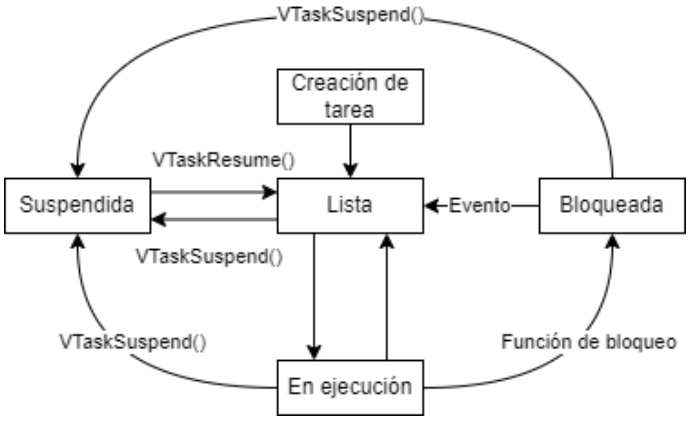


Figura 4.2: Estados posibles de una tarea en FreeRTOS.

4.1.3. Semáforos

Un semáforo es un recurso provisto por el RTOS que puede utilizarse para la sincronización de tareas, y tambien, para coordinar el acceso a recursos compartidos.

El sincronismo de tareas puede darse, por ejemplo, al momento de acceder a un buffer, en el que algunas tareas depositan información, mientras que otros la leen y posteriormente la eliminan. Cuando el buffer está vacío, el contador de semáforo es igual a cero. Esto indica que las tareas encargadas de leer la información deben esperar a adquirir un semáforo para acceder al recurso. Mientras el buffer permanezca vacío, el valor del contador será cero. Una vez que una tarea deposita información en el buffer, un semáforo se cede y el contador incrementa su valor en uno. Así, el buffer podrá ser leído por una tarea destinada para esto, lo cual decrementará su valor. De este modo se logra sincronizar las tareas para un mejor funcionamiento.

Cada vez que una tarea de lectura quiere acceder al recurso, primero chequea el valor del contador. En caso de que dicho valor sea cero, queda a la espera del semáforo. Para las tareas de escritura ocurre algo similar, si el valor del contador es máximo dicha tarea permanece a la espera de un decremento del contador para ejecutarse. En la figura 4.3 se observa lo mencionado.

Un caso particular de los semáforos es el binario, el cual cumple con el mismo modo de funcionamiento con la diferencia de que su contador solo puede adquirir los valores de uno o cero. Esto quiere decir que una sola tarea puede acceder al recurso protegido por esta tarea al mismo tiempo o para sincronizar el funcionamiento de dos tareas.

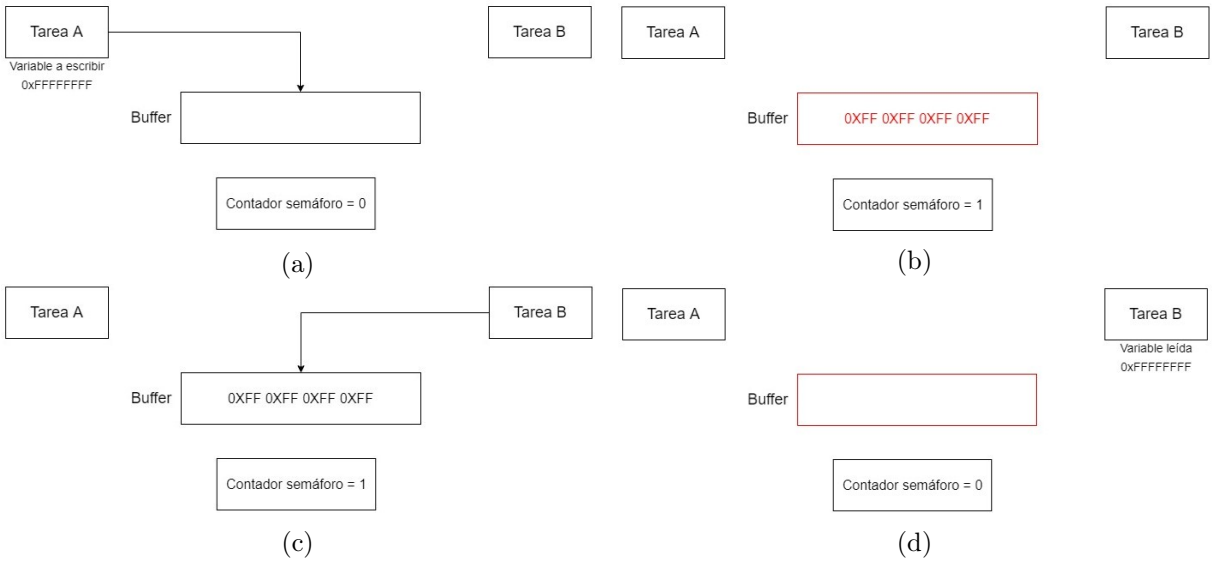


Figura 4.3: Proceso de escritura y lectura de buffer utilizando un semáforo. En (a) se observa el momento previo a la escritura del buffer, donde el contador de semáforo esta en 0. En (b) se realiza la escritura y el contador de semáforo se incrementa en 1, habilitando la lectura del buffer, que se realiza en (c). En (d) se observa el momento posterior a la lectura, donde el buffer se limpia y el contador se actualiza a 0.

4.1.4. Colas

Cuando dos tareas deben comunicarse, existe la posibilidad de utilizar una variable global para la comunicación. Desafortunadamente, acceder a una variable global desde dos tareas concurrentes puede ocasionar que el valor almacenado en la misma, en cierto momento, no sea el esperado. Para clarificar el problema, suponga el siguiente ejemplo. Tres tareas acceden a la misma variable global, de 64 bits. En un primer momento, la tarea A desea modificar la variable. Primero, modifica la parte alta de la variable. Antes de poder escribir la parte baja de la variable, se interrumpe la operación por una tarea de mayor prioridad denominada B, lo cual se observa en la figura 4.4a. Esta tarea actualiza la parte baja y alta de la variable global y luego este valor se comparte con otra tarea que se denomina C que se encargada de leer la información, como se ve en 4.4b. Una vez que C finaliza, A retoma la operación y escribe la parte baja del dato, generándose un dato erróneo, como se detalla en 4.4c. Este dato se captura por la tarea C.

Por lo mencionado anteriormente, es necesario proteger las variables globales y acceso a memoria. Una forma de hacer esto es desactivar las interrupciones por prioridad de tareas por un corto lapso de tiempo y en este periodo temporal convertir las variables globales de interés en variables locales. De este modo, la tarea opera haciendo uso de estas variables locales sin correr el riesgo de que surjan problemas similares a los mencionados. Por último, cuando la tarea finaliza la ejecución, se realiza el proceso inverso de asignación para retornar la variable global. El problema de esto es que cada vez que se desea trabajar con una variable global compartida dentro de una tarea, es necesario repetir este proceso generando mucho código repetitivo lo cual puede resultar engorroso.

Otro mecanismo, además de la forma básica son las colas. Una cola es un recurso provisto por el RTOS, que consiste básicamente en un buffer FIFO (first in first out), en el cual se escribe y lee la información por las tareas correspondientes. La cola garantiza que el proceso de escritura no va a poder ser interrumpido por otra tarea, aunque esta sea de mayor prioridad. Volviendo al caso anterior, las tareas A y B van a depositar la información en el buffer y la tarea C podrá leer la información según el orden en el que fue colocada. En la figura 4.5 se observa esto.

Una vez que el dato se lee por la tarea, se elimina y los datos anteriores se desplazan un lugar

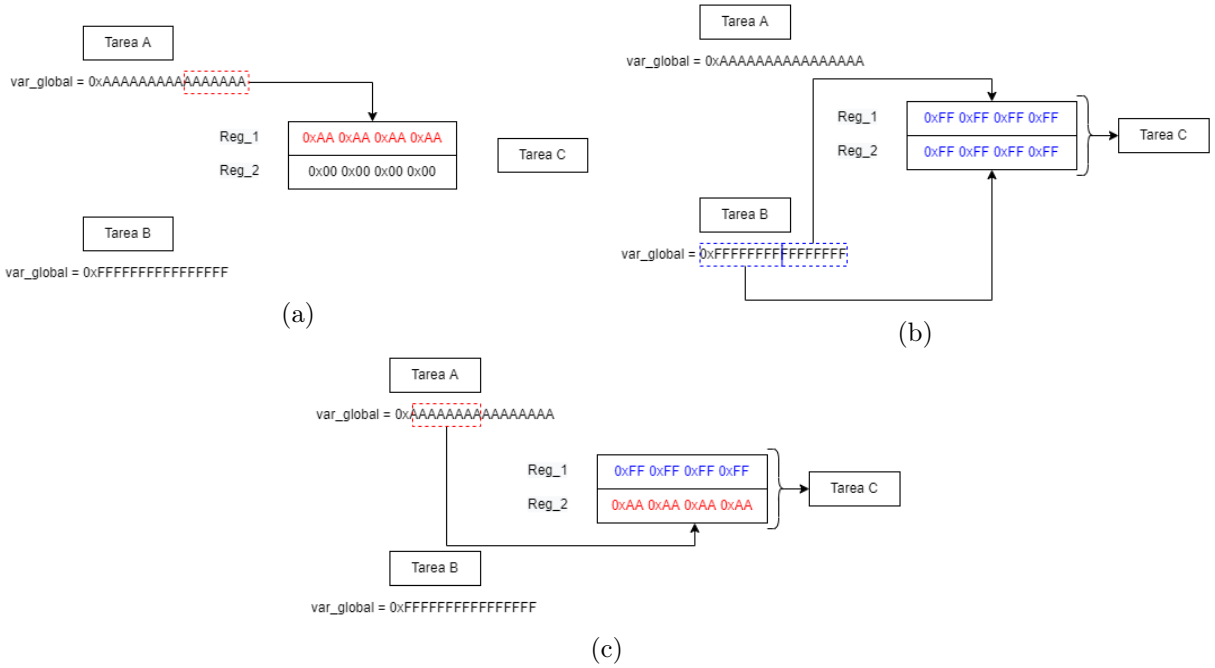


Figura 4.4: Proceso de escritura y lectura de memoria ejecutado por múltiples tareas, sin colas, ni protección de variables.

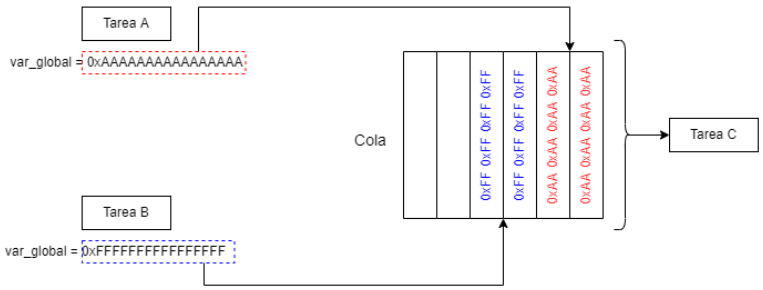


Figura 4.5: Buffer que brinda la cola para el manejo de variables globales.

a la derecha para continuar con la lectura. La lectura se realiza de manera asíncrona. También es posible definir un tiempo de espera máximo para cuando la cola no admite más valores debido a que se encuentra completo el buffer y superado este valor, el dato se descarta. Lo mismo ocurre cuando el buffer está vacío y la tarea de lectura se ejecuta esperando leer.

4.1.5. Principio de funcionamiento del planificador de CPU en FreeRTOS

Para detallar el modo de funcionamiento de un programa que implementa FreeRTOS, se propone establecer cuatro tareas de distintas prioridades. Se comienza por la rutina de configuración inicial y luego se crea cada una de las tareas. En los microcontroladores multi-núcleo es posible ejecutar más de una tarea al mismo tiempo, pero en los microcontroladores más simples que solo disponen de un núcleo, solo es posible ejecutar una tarea a la vez. Por esto, es necesario dividir el tiempo de procesamiento del núcleo en las distintas tareas. Para esto se destina un timer por hardware que se encarga de interrumpir el procesador en intervalos de tiempo regulares denominados ticks. Este periodo de tiempo puede variar dependiendo de las características del núcleo, pero generalmente es 1 ms. Por lo tanto en cada tick, se ejecuta el planificador de CPU para determinar cuál es la tarea que debe continuar usando la CPU dependiendo de su prioridad. En la figura 4.6 se observa la actividad del RTOS utilizando tiempo de CPU (barra azul).

Suponga que existen 4 tareas y que poseen las siguientes características:

- Tarea A, de prioridad 1, la cual comienza su ejecución en el momento 0. Utilizará 4 slices de tiempo.
- Tarea B, de prioridad 2, la cual comienza su ejecución en el momento 2. Utilizará 2 slices de tiempo.
- Tarea C, de prioridad 3, la cual comienza su ejecución en el momento 3. Utilizará 2 slices de tiempo.
- Tarea D, de prioridad 1, la cual comienza su ejecución en el momento 9. Utilizará 2 slices de tiempo.

Para comenzar, el planificador de CPU verifica que tareas se encuentran listas para ser ejecutadas. Suponiendo que solo la tarea A esta disponible comienza su ejecución.

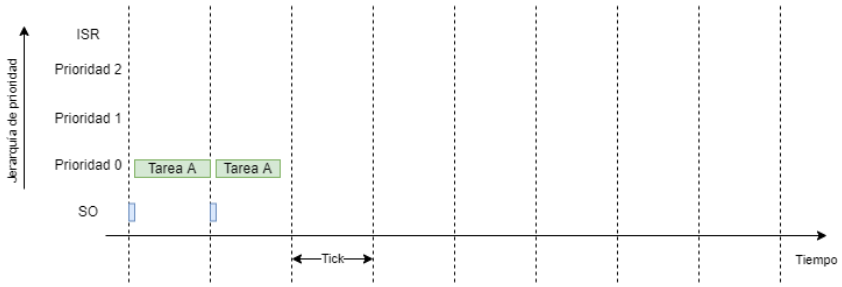


Figura 4.6: Ejecución de tareas en función del tiempo en FreeRTOS.

En el siguiente tick, planificador de CPU revisa si hay alguna tarea de mayor prioridad disponible para ser ejecutada y como esto es negativo, procede con la finalización de la tarea A para luego asignarle un estado de bloqueo. Una vez que dicha tarea finaliza, la tarea B adquiere el estado de lista y como es la de mayor prioridad, el planificador comienza su ejecución. En algún momento de la ejecución de B, la tarea C está lista. Esta última, cuenta con mayor prioridad, pero debe esperar al siguiente tick para poder ser ejecutarse. Cuando se llega al siguiente tick, el planificador de CPU encuentra una tarea de mayor prioridad lista y la tarea B se interrumpe quedando bloqueada como se ve en 4.7.

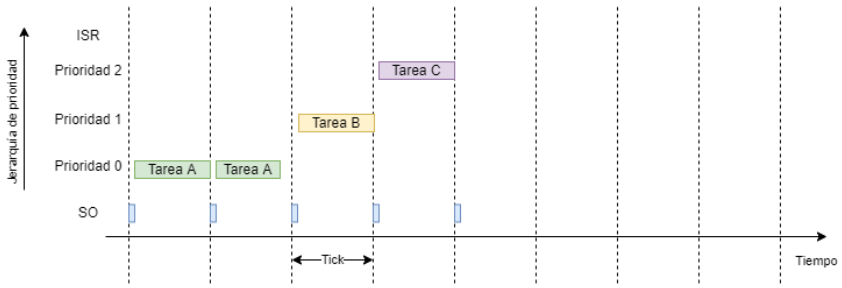


Figura 4.7: Ejecución de tareas en función del tiempo en FreeRTOS.

La ejecución de la tarea C comienza. En el próximo tick, el planificador verifica que C es la tarea con mayor prioridad y procede a seguir con su ejecución. En algún instante del intervalo de operación, una rutina de servicio de interrupción (ISR) asociada al hardware detiene la ejecución de la tarea actual. Este tipo de rutinas disponen de la prioridad suficiente para interrumpir cualquier tarea, salvo que sean deshabilitadas, y permiten detectar eventos de manera asíncrona como por ejemplo el cambio de estado lógico de un pin de entrada. Estas funciones deben mantener la menor cantidad de lineas de código posible para evitar un retraso en la ejecución

de las tareas. Finalizada la ISR, la ejecución retoma en la tarea C. Cuando la ejecución de C termina, el planificador verifica las prioridades nuevamente y retoma la ejecución de B. Cuando B concluye con su operación, por como fue diseñada esta tarea, entra en un estado de bloqueo por 2 ticks. El primer tick se resta antes de finalizar el intervalo temporal actual, luego resta contar un tick más y como no hay tareas listas el planificador de CPU queda en estado inactivo hasta el siguiente tick. Eso se ve en la figura 4.8.

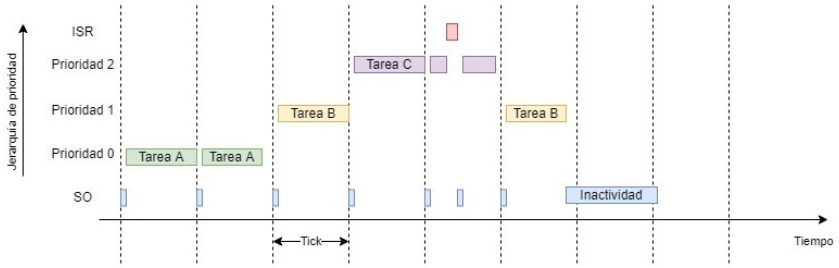


Figura 4.8: Ejecución de tareas en función del tiempo en FreeRTOS.

Al completar esto último, las tareas A y D se encuentran listas para ser ejecutadas. Asumiendo que ambas cuentan con la misma prioridad, el planificador alterna la ejecución entre ambas tareas (round-robin) como se ve en la figura 4.9. Esto continua mientras sea necesario hasta finalizar.

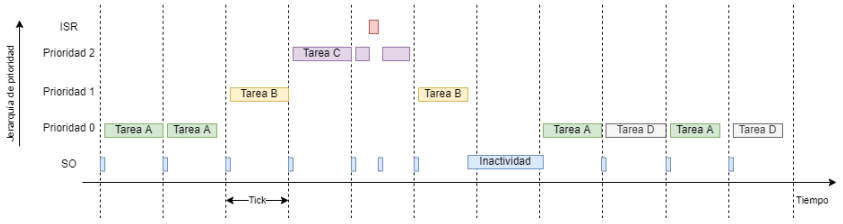


Figura 4.9: Ejecución de tareas en función del tiempo en FreeRTOS.

4.2. Uso de memoria

Un recurso sumamente importante a administrar para el buen funcionamiento de un programa que utiliza RTOS es la memoria del programa. Se puede establecer que esta se segmenta en tres áreas, la memoria estática, la memoria heap y la stack. Cada una de ellas se destina a contener información necesaria para el correcto funcionamiento del programa.

La memoria estática contiene cualquier variable global o estática que se declare en el programa. Luego de haber comenzado la ejecución, el compilador determina el espacio necesario para estas variables y las asigna en dicha región, esto se denomina almacenamiento estático. Las variables asignadas en la memoria estática no pueden cambiar su tamaño y su existencia durante toda la ejecución del programa.

La memoria stack entra juego cuando dentro del programa en ejecución, se efectúa un llamado a una subrutina. Al ejecutar una función, en esta sección de memoria se designa y almacena automáticamente información relacionada a esta, como las variables locales empleadas y datos requeridos para retornar al hilo de ejecución principal. Esta sección de memoria es del tipo LIFO (last in first out), por lo que a medida que ingresan información a la memoria stack, esta es empujada hacia abajo. Esto facilita el manejo y control de bloques de información referidos a distintas funciones y el orden en que estos deben ser liberados. Cuando la función termina, la sección de memoria stack reservada se libera, dejándola disponible a otras funciones.

Por último, la memoria heap, es la región utilizada por el usuario para almacenar dinámicamente datos generados durante la ejecución. El usuario destina explícitamente la cantidad de

memoria a una variable o arreglo y una vez que finaliza su uso se elimina, siendo también esto responsabilidad del usuario. En el lenguaje C, esto se realiza mediantes las funciones malloc() y free(). En caso de no producirse la liberación, la memoria podría crecer en tamaño y causar perdidas de información. Otro problema no tan extraño pero difícil de localizar es la colisión de las memorias heap y stack. Si ambas crecen mucho en tamaño puede ocurrir que estas se solapen y se reescriban entre sí.

En FreeRTOS, cada vez que se crea un objeto, como una tarea, una cola o semáforo, se realiza una asignación dinámica dentro del bloque de memoria heap y cuando dicho objeto se elimina, el bloque de memoria se libera. En el caso particular de la creación de una tarea, la porción de memoria reservada se subdivide en dos regiones, la stack de la tarea y un bloque de control de tareas (TCB). Dentro de TCB se guarda la información relacionada con la tarea como su prioridad y su dirección. La stack de la tarea funciona del mismo modo que la memoria stack del sistema, pero con la diferencia que el usuario es el responsable de asignar correctamente su tamaño. En caso de que la cantidad asignada por el programador, para la pila de una tarea, no sea suficiente, se corre el riesgo de que se solapen segmentos de memoria y se modifique información de otras tareas. En tal caso, la depuración del problema será compleja, porque el sistema se comportará de manera "extraña" y será difícil relacionar su comportamiento con una reserva de memoria insuficiente.

En muchos casos, los esquemas de asignación dinámica de propósitos generales no son adecuados para aplicaciones de tiempo real. Anteriormente se mencionó que en C es posible utilizar funciones como malloc() y free() para administrar la memoria heap. Si bien estas funciones puede ser útiles en muchos programas, hay que reconsiderar su uso en los de tipo RTOS, ya que puedan causar problemas. Dentro de todas las dificultades que puede acarrear su uso, se puede considerar que estas funciones no son determinísticas, por lo que su tiempo de ejecución puede variar y esto es poco deseable en programas de esta características. Tampoco están protegidos de subprocesos, por lo que su ejecución podría verse interrumpida. Por último, el uso de estas funciones puede causar la fragmentación de la memoria heap. Esto consiste en la subdivisión de la RAM libre dentro de la memoria heap en fragmentos pequeños. A causa de esto, si se quiere realizar una asignación de memoria dentro de la heap y el bloque a asignar es de mayor tamaño que cualquier fragmento, ocurrirá un error. Aunque la suma de todos los bloques libres separados sea más grande que el bloque que se desea asignar, esto no será posible. Para solucionar esto, FreeRTOS trata la asignación dinámica de memoria en su capa portátil, lo que posibilita al usuario proporcionar una implementación específica de la asignación de la heap.

4.3. Arquitectura y diseño del software desarrollado

Para el desarrollo del software necesario para la implementación del sistema de control, se hizo uso de CMSIS (Cortex Microcontroller Software Interface Standard) y HAL (Hardware Abstration Layer). El estandar CMSIS es una capa de abstracción de hardware, que permite lograr una interfaz de software simple y consistente con el procesador y los periféricos. Si bien los distintos fabricantes como FSL, ST, Energy Micro, etc, utilizan Cortex M, los periféricos son distintos lo que implica que su diseño, interfaz y registros son diferentes. El propósito de CMSIS es permitir que las MCU Cortex M de diferentes fabricantes logren un cierto grado de consistencia al menos en el nivel central, y mejorar la eficiencia de la migración de software. En la figura 4.10 se observa la estructura de CMSIS. Dentro de esta estructura, se centra la atención en el bloque CMSIS-RTOS, que es una interfaz de programación estandarizada para sistemas operativos en tiempo real para control de subprocesos, gestión de recursos y tiempo.

Por otro lado HAL, proporciona un conjunto sencillo de interfaces de programación de aplicaciones (APIs) para interactuar con capas superiores. Las APIs se dividen en dos categorías, las genéricas que proporcionan funciones comunes para toda la serie de STM32 y las API de extensión, que incluyen funciones especificas y modificables. HAL incluye un conjunto completo

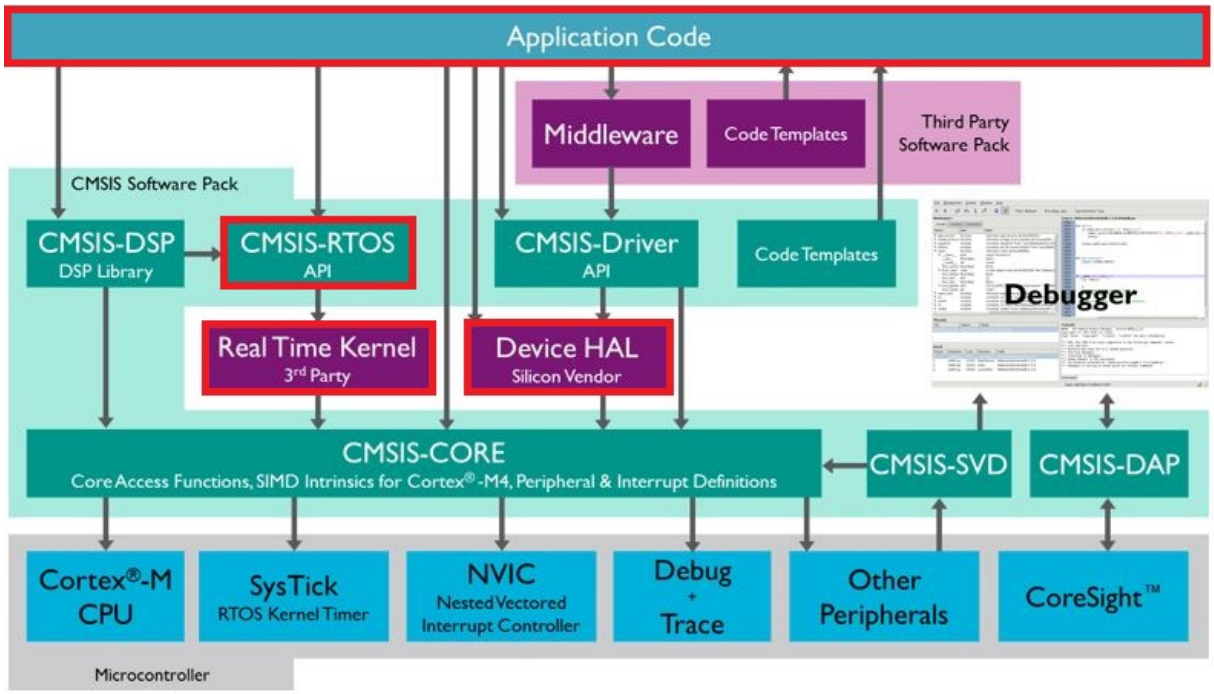


Figura 4.10: Diagrama en bloques de la estructura de CMSIS, extraído de la página oficial [1].

de APIs listas para ser usadas, que simplifican la aplicación para el usuario. También implementa la detección de fallas en los tiempos de ejecución comprobando los valores de entrada de todas las funciones, lo cual mejora la solidez del firmware[33].

El código desarrollado implementa 6 tareas de distintas prioridades y cada una de ellas desempeña una funcionalidad diferente. A continuación se describe el flujo que posee el programa y la función de las tareas. Cabe destacar que en la totalidad del código, la transmisión de información entre tareas se realizó con el método de conversión y reconversión de variables globales a locales como se explica en la sección 4.1.4. No se implementa una cola ya que el microcontrolador con el que se trabajó cuenta con una cantidad de RAM limitada y se decidió priorizar el uso de dicho recurso en las tareas.

Antes de abordar la explicación de la funcionalidad de cada tarea existente dentro del código, se presenta en la figura 4.11 un esquema que detalla la cantidad de tareas presentes en el código y cómo interactúan entre ellas. De dicha figura se observan 7 tareas distintas, donde cada una de ellas cuenta con una función particular. El texto que se visualiza sobre las flechas representa la información que comparte cada tarea y la flecha que se cierra sobre la misma tarea indica su recursividad.

4.3.1. Tarea Calculo_Velocidad

Esta tarea se encarga de realizar el cálculo de la velocidad de giro que desarrolla cada uno de los motores. Dentro de todas las tareas existentes en el programa, esta es la única que utiliza un semáforo para realizar la ejecución, el cual es cedido por una ISR, que se explicará en las próximas subsecciones. Por lo tanto, hasta que la tarea no adquiere el semáforo, esta se encuentra bloqueada y apenas lo reciba, debido a su prioridad, comienza la ejecución.

En una primera instancia, se planteó utilizar el método T [19] que consiste en registrar la cantidad de pulsos de encoder en un tiempo dado, el cual es fijo. Este método se utiliza cuando la relación pulsos por intervalo es alto. En el caso del vehículo sobre el cual se realizó el desarrollo, inicialmente se cuenta con un encoder que posee 30 ranuras, lo que genera 60 interrupciones por cambio de flanco por cada revolución. Esto generó una relación de compromiso a la hora de medir

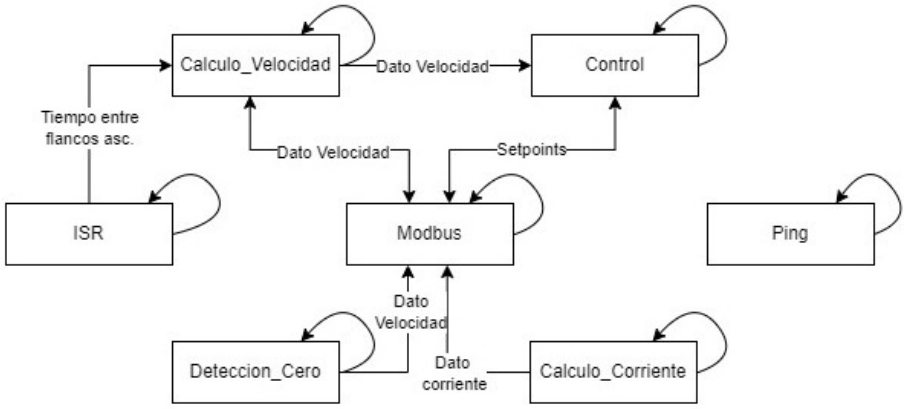


Figura 4.11: Diagrama en bloques de las tareas que se utilizan en el diseño e implementación del software y como dichas tareas se vinculan entre sí.

bajas velocidades, ya que para tener un buena medición de velocidad se requiere una ventana de tiempo muy amplía, lo que limita la frecuencia de muestreo de la velocidad. Ahora, si el intervalo es pequeño, se pierde resolución del valor medido.

Como se menciona en la sección 6.3, la frecuencia de muestreo del sistema es de 200 Hz. Por lo tanto, para obtener una resolución de medida de 0,1 Vueltas/Seg utilizando el método T, se necesita un disco que cuente con 1000 ranuras. Esto último no es realizable dentro del marco del proyecto, por lo cual se decidió implementar el método M. Este algoritmo consiste en determinar el tiempo existente entre dos pulsos consecutivos generados por el encoder óptico ubicado en el eje del motor. Este método está diseñado para medir bajas velocidades y su ecuación viene dada por 4.1.

$$v_L = \frac{1}{N}\frac{F_{timer}}{\Delta ticks} \tag{4.1}$$

Donde:
v_L: velocidad de rotación del eje en RPS.
N: numero de ranuras del disco con el cual se generan los flancos.
$\Delta ticks$: representa el número de ticks que se cuenta entre flanco y flanco por el timer.
F_{timer}: es la frecuencia a la que está configurado el timer en Hz.

Ya que se dispone de una tarea para realizar el cálculo de velocidad de todos los motores, fue necesario identificar que motor fue el que cedió el semáforo a la tarea para el cómputo. Esto se realizó mediante un buffer global de cuatro componentes donde cada componente está asociada a un motor y representa que si su respectiva componente es igual a uno, entonces ese motor cedió el semáforo. Lo mismo ocurre con el timer que se destina a la obtención de los ticks entre flanco y flanco. Solo se dispone de un timer para obtener el tiempo entre pulsos de cuatro motores, por lo que también se requierió un buffer para esto. Cuando se obtiene la velocidad, esta también es almacenada.

En la figura 4.12, se presenta la señal que genera el encoder óptico cuando el motor gira a velocidad nominal. De esta se establece que el tiempo promedio entre pulso y pulso ascendente es aproximadamente 13 milisegundos. Por lo tanto, este es en promedio el tiempo mínimo que se puede presentar entre pulsos. Un Δticks inferior a este implica un error en la medición. Por esto se implementó un filtro pasa bajos, antes de realizar el cálculo de la velocidad. En caso de que la diferencia temporal entre pulsos sea inferior a 12 milisegundos, se descarta el valor de Δticks obtenido y se mantiene el anterior. En caso de que el valor sea superior, la tarea procede a hacer el cálculo de la velocidad. Obtenida dicha magnitud, se aplica un filtro pasa bajos para reducir su variabilidad. En la ecuación 4.2 se observa su ecuación. El ajuste de sus parámetros se realizó

Figura 4.12: Período entre dos pulsos consecutivos de encoder, a máxima velocidad de rotación de un motor.

experimentalmente, teniéndose en cuenta que su acción no genere un retardo considerable en la medición de la velocidad, que luego se traslada a la acción de control.

$$v_{filtrada} = 0{,}85 v_{filtrada} + 0{,}15 v_L \tag{4.2}$$

La prioridad de ejecución de esta tarea es alta ya que proporciona información necesaria para la ejecución del sistema de control. En la figura 4.13 se observa el diagrama de flujo de la tarea.

4.3.2. Tarea Detección_Cero

Esta tarea se encarga de establecer en cero la velocidad de cualquiera de los motores en los que no se detecte movimiento después de un breve periodo de tiempo. Para lograr esto, se utilizó el mismo timer que se destinó para la medición de la velocidad. Cada vez que el timer se desborda, se ingresa a una rutina de interrupción en la cual se incrementa en una unidad todas las componentes de un buffer que contabiliza la cantidad de desbordes que se van produciendo. Hasta ahora la tarea de interés no ha entrado en juego. Cuando se ingresa en esta tarea, se toma el buffer mencionado y se analiza en cada una de las componentes si no se supera un valor umbral establecido. En caso de haberse superado, la respectiva componente de velocidad se iguala a cero, mientras que si esto no ocurre, la velocidad no sufre modificaciones por esta tarea.

La frecuencia de ejecución de la tarea es de 200 Hz, que se corresponde con la frecuencia de funcionamiento del sistema general. La prioridad de esta tarea es normal debido a que el método empleado para la detección de velocidad cero, maneja tiempos de desborde varios órdenes de magnitud superiores al periodo de ejecución de la tarea.

Un problema asociado a esta manera de detectar que la rueda no esta girando, es que si el valor umbral es muy pequeño, a velocidades bajas de rotación podría detectarse que es cero. Esto se debe a que el tiempo entre flanco y flanco podría ser mayor al valor umbral. Por otro lado, si el valor umbral es muy elevado, el tiempo requerido para detectar que el motor está quieto puede elevarse considerablemente. Por esto se debe encontrar un valor de equilibro. En la figura 4.14 se observa el diagrama de flujo de la tarea. Se fijó como tiempo umbral 0,8 segundos.

4.3.3. Tarea Corriente

Esta tarea se encarga de realizar múltiples mediciones con los sensores INA219. En ella, se mide la corriente de cada uno de los motores del vehículo y de manera indirecta se mide la tensión

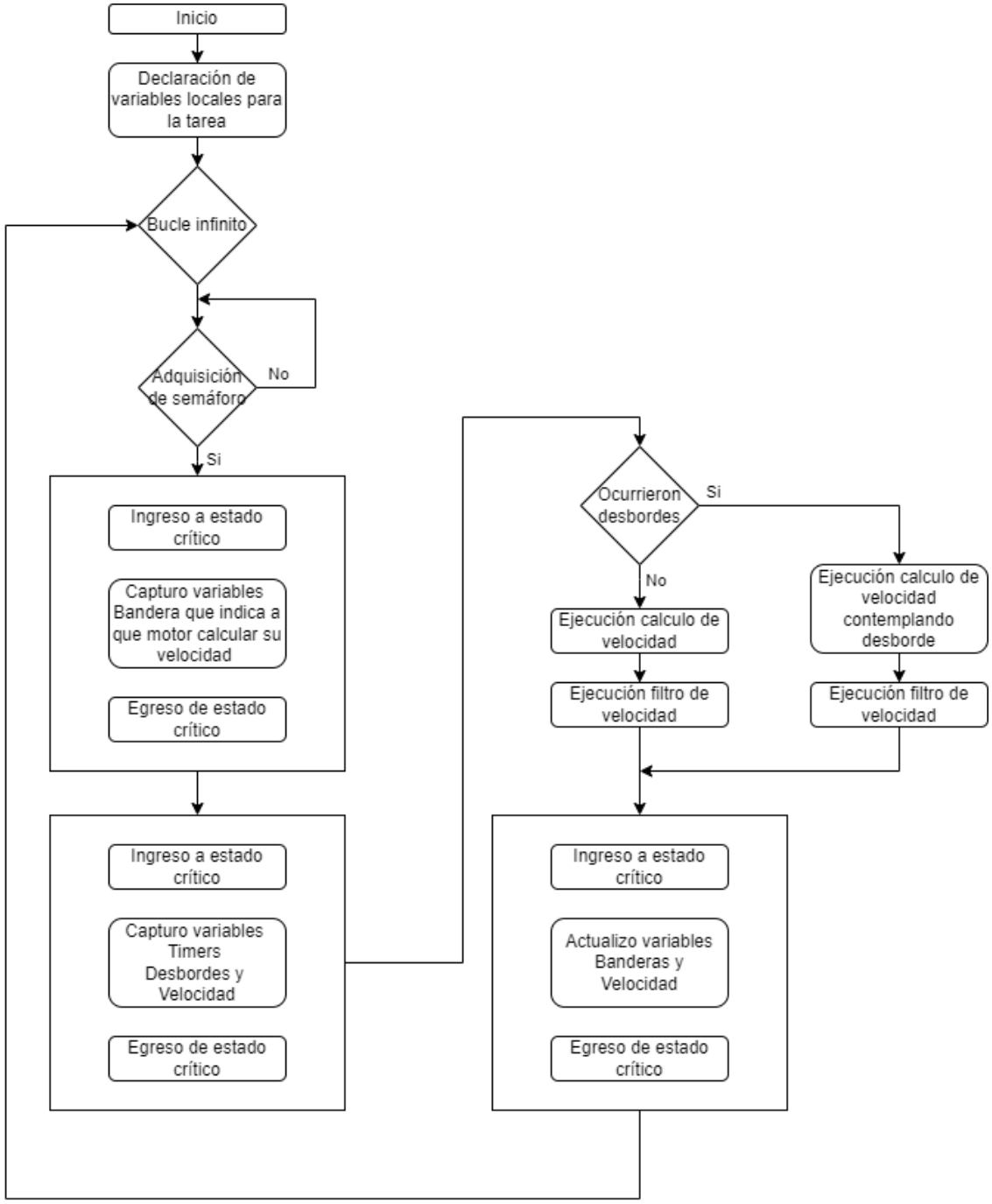


Figura 4.13: Diagrama de flujo de la tarea Calculo_Velocidad.

del bus de baterías. Se crea un arreglo de 5 componentes, el cual almacena todas las mediciones efectuadas con los módulos.

Se procedió inicializando y configurando cada uno de los módulos, definiendo de esta manera la tensión máxima que puede medir sobre la resistencia tipo shunt alojada en el módulo. Tambien otros parametros, como la ganancia de la medición, la resolución del conversor analógico digital empleado, el modo de operación del módulo y por último el tamaño de la ventana del filtrado de la medición. Finalmente obteniendo un rango de medición de $\pm$ 6400 mA, con una resolución de 12 bits y un filtro promediador con 128 muestras en una ventana temporal de 64 mSeg.

Una vez configurados los sensores, se realizó la medición de todos estos a una frecuencia de 200 Hz lo que se corresponderse con el tiempo de muestreo del sistema. Ya que esta tarea

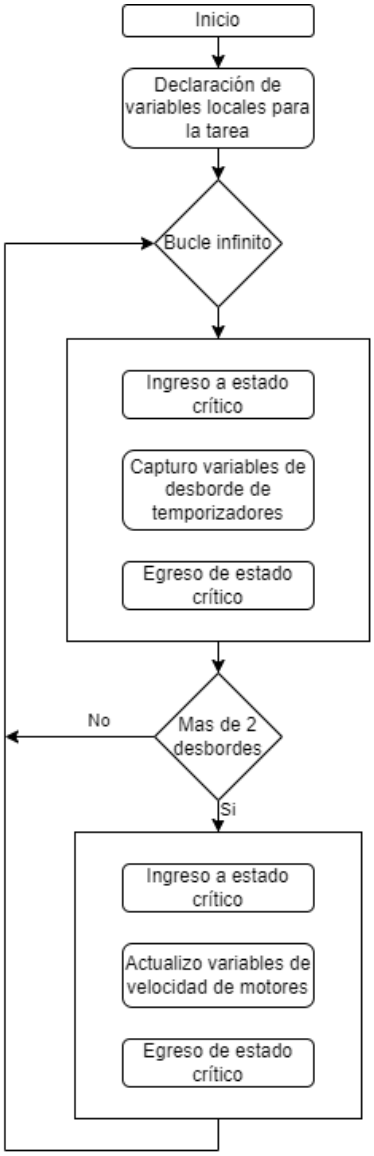


Figura 4.14: Diagrama de flujo de la tarea Deteccion_Cero

suministra información relevante a la tarea de control, su prioridad de ejecución es alta. Para realizar la medida, la tarea selecciona la dirección del módulo y el tiempo de expiración para ejecutar la función (este último es necesario para no bloquearla). Definido lo anterior, consulta el estado del bus de comunicaciones I2C y en caso de que esté libre, configura nuevamente el módulo (paso intermedio necesario debido a que si el módulo detecta grandes armónicos en la medición se reinicia) y luego solicita el valor de la corriente. En el caso de que el bus de comunicaciones se encuentre ocupado o la petición al módulo llegue a su tiempo de expiración, detecta que existe un problema. A partir de esto, reinicia el periférico de I2C del microcontrolador y además vuelve a configurar el sensor de corriente. Cuando un error en la comunicación ocurre, devuelve el valor de la medición previa junto con una bandera que indica la ocurrencia de un error.

Tras obtener el valor de corriente, la tarea almacena en el arreglo previamente mencionado, en la posición que corresponde al sensor encuestado. Luego de esto, a cada valor de corriente obtenido desde los motores le aplica un filtro pasabajos similar al de la tarea del cálculo de la velocidad, con los coeficientes 0,85 y 0,15. Finalmente, para que la información pueda ser utilizada en otras tareas, se entra en un estado crítico por software, el cual se desactivan las interrupciones, se actualizan las variables globales del sistema y se sale de dicho estado. En la

figura 4.15 se puede ver el diagrama de flujo de la tarea que se explica.

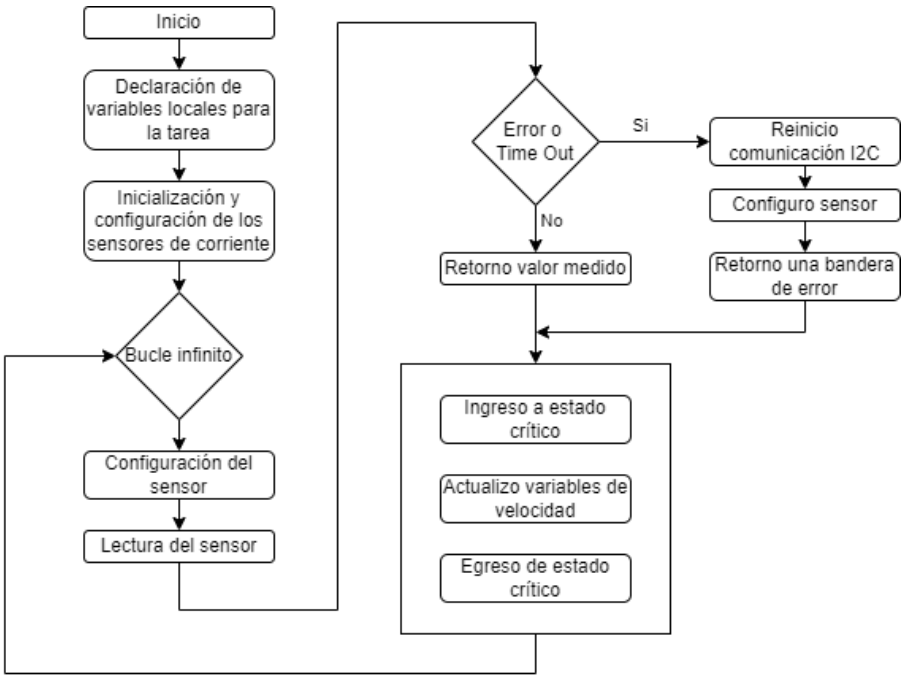


Figura 4.15: Diagrama de flujo de la tarea Corriente

4.3.4. Tarea Modbus

Esta tarea tiene la función de administrar la información que proviene de las demás tareas y representarla en el mapa Modbus. Dicha información corresponde a las variables de velocidad, corriente, sentido de giro, tensión de bus y bits de control del sistema. Los formatos que se utilizaron para representar la información son enteros de 16 bits y flotantes de 32 bits, según corresponda, como se muestra en 5.3

Inicialmente la tarea captura las variables globales mencionadas y copia en variables locales, todo dentro de una sección crítica donde se deshabilitan las interrupciones por prioridad. Luego procede a almacenar el contenido de las variables locales mencionadas (flotantes de 32 bits) dentro un arreglo de enteros sin signo de 16 bits de dos componentes a través de la función memcpy() propia del lenguaje C. Finalmente retorna a la sección crítica donde se actualiza el mapa Modbus con los valores expresados en un nuevo formato.

Debido a que la funcionalidad de la tarea está enfocada a la transmisión y visualización de información, la cual no es crítica para el funcionamiento del sistema, se asignó una prioridad normal. La frecuencia de ejecución de la tarea es 20 Hz. En la figura 4.16 se observa el diagrama de flujo de la tarea.

4.3.5. Tarea Control

Esta tarea se encarga de realizar actividades relevantes, como el control y linealización de la velocidad, la inversión de la polaridad del motor cuando se requiere un cambio de giro, corte de seguridad ante baja tensión en baterías y una parada general del sistema. Se comenzó por establecer los arreglos para almacenar la velocidad medida, la velocidad de referencia o setpoint y el sentido de giro actual de cada motor. Cada uno de esos arreglos consta de cuatro componentes, donde cada componente se asocia a cada uno de los motores. Por último, se designó una variable para la parada general del sistema.

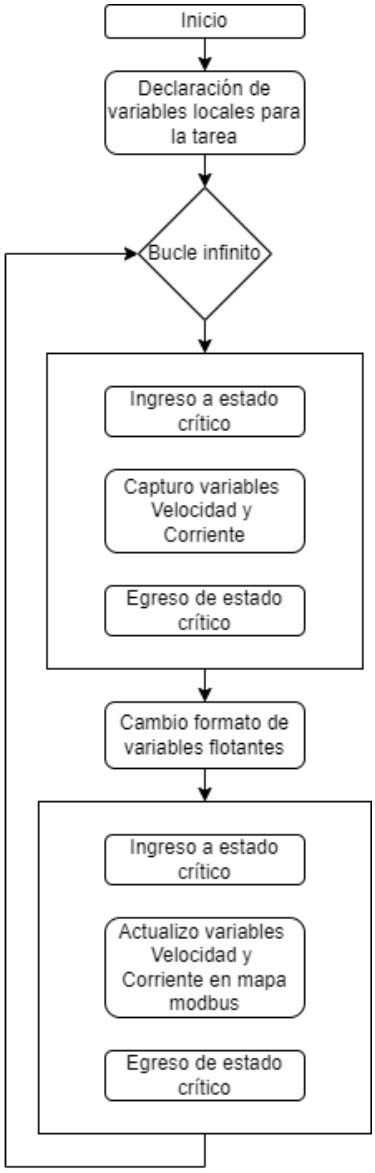


Figura 4.16: Diagrama de flujo de la tarea Modbus

La tarea comienza capturando los valores globales de la velocidad, setpoint, sentido de giro y parada general del sistema para trabajarlos dentro de ella de manera local. Esto lo realiza dentro de una sección crítica de código en la cual las interrupciones por prioridad, se desactivan. Luego verifica el estado de la variable local de parada general, en caso de que sea igual a cero, el arreglo local de setpoint se establece en cero y en caso de que sea uno no surgen modificaciones. Se procede a realizar un cambio de giro en caso de ser necesario.

Luego ejecuta las funciones de control y linealización para la velocidad de cada motor como se explica en la secciones 6.1.1 y 6.5. Verifica que las baterías no se encuentren en bajos niveles de tension. Luego la salida del linealizador se pasa como valor de entrada del registro que regula el ciclo activo de las señales PWM que fijan las velocidades de los motores.

Como se menciona previamente, el valor de la parada general puede modificar los valores de setpoint. De esta manera si la variable es cero, los valores de setpoint se vuelven cero generando que los motores disminuyan su velocidad hasta detenerse.

La frecuencia de ejecución es de 200 Hz, la cual se corresponde a la frecuencia de muestreo con la que se discretiza el sistema y se diseña el control previamente en Simulink. Debido a la necesidad de mantener una frecuencia estable para que el vehículo tenga un comportamiento

similar al planteado en el diseño, se asignó la prioridad más alta posible (tiempo real). En la figura 4.17 se observa el diagrama de flujo de la tarea.

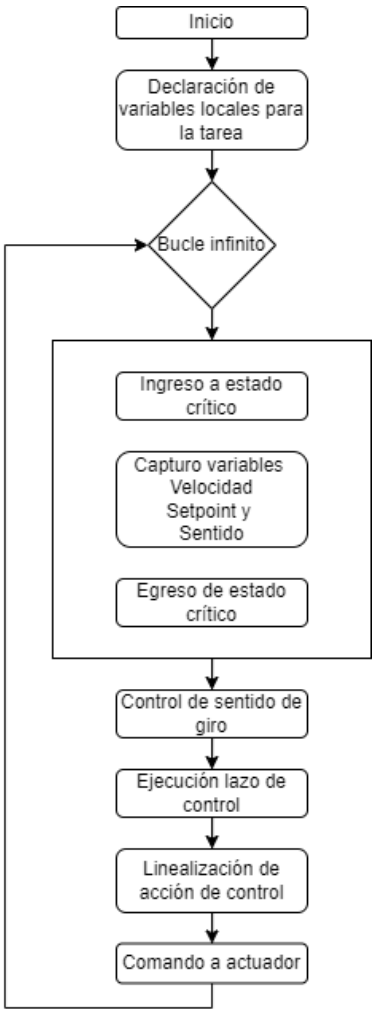


Figura 4.17: Diagrama de flujo de la tarea Control

4.3.6. Tarea Ping

Esta tarea tiene como función asegurar el reinicio del microcontrolador, en caso de que ocurra algún error grave durante el funcionamiento. De esta manera, se evita que ante una falla el sistema deje de funcionar hasta ser reiniciado manualmente. Para esto se implementó el watchdog del microcontrolador. Este es, básicamente, un timer que cuando llega a cero reinicia el microcontrolador, por lo cual activamente se busca reiniciarlo para que no suceda. En caso de que alguna tarea demore mucho más de lo planeado, o se corrompa el oscilador del microcontrolador, este timer llegará a cero y reiniciará el sistema.

Para implementar esta funcionalidad en un sistema operativo, en el cual todo el código es non-blocking, se eligió la estrategia de señalizar mediante banderas la ejecución de cada tarea y verificar cada cierto tiempo que todas las banderas se encuentren señalizadas correctamente. Que el código sea non-blocking quiere decir que el mal funcionamiento de una o varias tareas dentro del código no ralentizan o afectan el funcionamiento general del programa o las demás tareas.

Para esto la tarea procede a verificar que las componentes de un arreglo global que contiene en cada posición las banderas de cada tarea, se encuentren en nivel alto. En caso de éxito, procede a

restablecer el temporizador del watchdog, de lo contrario indica que existe un problema, causando un reinicio del sistema.

Es necesario mencionar que existe una tarea (Calculo_Velocidad) que para ejecutarse espera hasta que se le ceda un semáforo. Dicha tarea no se ejecuta hasta que el vehículo se encuentre en movimiento y en caso de comprobar el estado de su bandera, si el vehículo se encuentra detenido, nunca reportará un estado correcto. Debido a este comportamiento en el flujo de las tareas, se decide no incluirla en el chequeo de estado de las banderas.

Esta tarea se ejecuta con una frecuencia de 10 Hz y el timer del watchdog tarda 4096 mSeg en llegar a cero. Esto implica que en caso de tener que reiniciar el microcontrolador habrán ocurrido múltiples lecturas de las banderas mencionadas anteriormente en cero, indicando que alguna tarea no se reporta. Debido a las grandes diferencias de intervalos temporales entre el desborde del timer y el periodo de ejecución de la tarea, no es necesario cumplir con estampas temporales de manera precisa, sino que se necesita una ejecución relativamente periódica. Por lo tanto se le asignó una baja prioridad. Por último, en la figura 4.18 se observa el diagrama de flujo para la tarea.

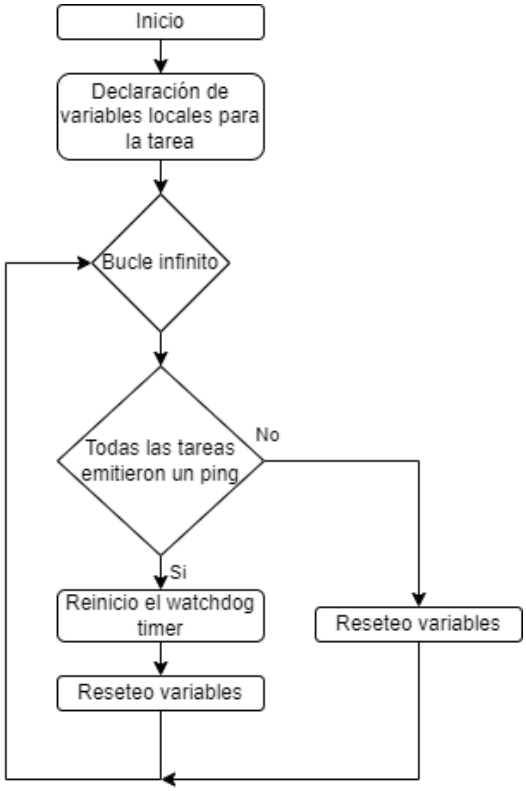


Figura 4.18: Diagrama de flujo de la tarea Ping

4.3.7. Rutina de interrupción

La rutina de interrupción se ejecuta cada vez que se detecta un flanco ascendente en cualquiera de los pines de entrada del microcontrolador que se destinan para esta función. Estos pines se encuentran conectados a los enconders ópticos colocados en los ejes de los motores y cuando se produce esta interrupción se llama a una función callback, la cual tiene como parámetro de entrada el pin que genera dicha interrupción. Dentro de esta función, se comienza chequeando que pin origina la interrupción y una vez que se determina esto, se almacena en variables globales el valor actual del timer y las dos muestras temporales anteriores. Luego, en un arreglo global de cuatro componentes identifica el encoder que genera la interrupción.

Capítulo 5

Protocolo de comunicación

En el presente capítulo se trata el protocolo de comunicación utilizado para realizar la comunicación entre el vehículo y un dispositivo maestro, como puede ser una PC. Además se explica como funciona el protocolo, sus características y como se aplicó al proyecto.

5.1. Necesidades que se cubren

Uno de los problemas que se resolvió, es la comunicación entre el robot y un dispositivo maestro. Esta permitió la vinculación del sistema de control de movimiento, como esclavo, con un dispositivo externo que actúa como maestro. El dispositivo maestro debe poder enviarle los distintos setpoints de velocidad de giro de cada motor y a su vez el robot, como esclavo, debe poder enviarle información del sistema si el maestro la solicita, como por ejemplo velocidad, corriente, setpoint, etc. Por último, es un protocolo robusto y capaz de recuperarse e informar en caso de fallos en la comunicación.

Evaluando los requerimientos planteados, se implementó el protocolo Modbus RTU para la comunicación.

5.2. ¿Qué es Modbus RTU?

Modbus RTU (Remote Terminal Unit) es un protocolo con formato solicitud - respuesta que trabaja con jerarquías de maestro - esclavo. La comunicación ocurre entre dos dispositivos, el maestro debe iniciar la comunicación con una solicitud de algún tipo y esperar la respuesta del esclavo que corresponda. Un dispositivo maestro puede contar con un máximo de 247 esclavos conectados en un mismo bus de datos [2][4].

5.2.1. Modo de funcionamiento

La trama del protocolo se muestra en la figura 5.1. El maestro, para poder realizar una solicitud Modbus, debe enviar una trama que cumpla con la que el protocolo establece. Debe contar con una condición de inicio conformada por un tiempo de silencio equivalente al de la transmisión de cuatro caracteres, dirección de esclavo, una función que se desea ejecutar (como las que se muestran en la tabla 5.1), la información que se desea transmitir, un CRC (cyclic redundancy check) para control de errores de la trama y por último una condición de finalización. Cuando el dispositivo esclavo recibe una trama Modbus debe procesarla. Este tiene que interpretar la trama recibida, proceder a identificar si el CRC es correcto y luego verificar si la dirección a la cual esta destinado el mensaje coincide con la que este posee. En caso contrario debe descartar la trama. Si el mensaje es válido, se identifica el tipo de función que se solicita y se realiza la ejecución. Se procede a armar una respuesta válida para el dispositivo maestro indicando una confirmación de la ejecución o devolviendo la información solicitada según corresponda (siempre

que no se trate de comunicación Broadcast, dado el caso no debe responder). Se calcula y anexa sobre el final de la respuesta un CRC. Por último se procede a enviar la trama.

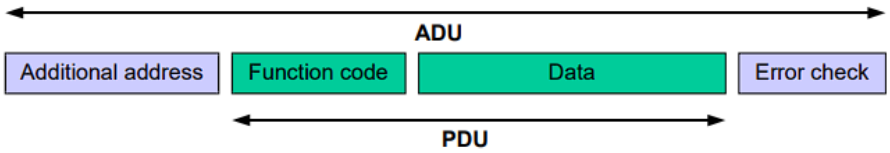


Figura 5.1: Trama de comunicación con protocolo Modbus.

Los mensajes que puede enviar un dispositivo maestro a uno esclavo pueden ser los siguientes:

- Lectura (códigos de función 01, 02, 03 o 04) el dispositivo maestro solicita al esclavo que este le envíe cierta cantidad de información.
- Escritura (códigos de función 05, 06, 15 o 16) el dispositivo maestro escribe distintos tipos de datos en el esclavo según la función.
- Broadcast el dispositivo maestro se comunica con todos los esclavos que se encuentren en el bus de datos (el dispositivo no espera ningún tipo de respuesta).

En la tabla 5.1 se muestra el significado de los códigos de función.

Tabla 5.1: Funciones de Modbus.

Function Code	Action	Table Name
01	Read	Coils
05	Write single	Coil
15	Write multiple	Coils
02	Read	Discrete Input
04	Read	Input Registers
03	Read	Holding Registers
06	Write single	Register
16	Write multiple	Holding Registers

La información que maneja un esclavo Modbus se puede almacenar de distintas maneras según corresponda. Existen dos tipos de unidades de almacenamiento establecidas por el protocolo, los Holding Registers (HR) e Input Registers (IR) y por otro lado, los Coils (o bobinas) y Discrete Inputs. Los HR e IR son registros de números enteros de 16 bits mientras que los Coils y los Discrete Inputs son registro booleanos (de 1 bit). Los distintos tipos de datos pueden ser accedidos por el maestro, como se muestra en la tabla 5.2. Además, en función del tipo de dato será la función necesaria para que el maestro pueda leer o escribir el registro como se indica en la tabla 5.1.

Tanto los HR como los Coils son unidades de información orientada a una utilización de propósito general, en cambio los IR son pensados para capturar valores de conversores analogicos digitales y las Discrete Inputs son pensadas para capturar valores de entradas digitales del dispositivo esclavo.

5.2.2. Utilización de variables flotantes

Una variable de tipo flotante posee 32 bits y es utilizada para representar números con su parte decimal, obteniendo de esta manera más precisión en la representación del número. Este tipo de variables no pueden ser utilizadas de manera directa en el protocolo Modbus debido

Tabla 5.2: Tipos de datos utilizados en el protocolo Modbus.

Tipo de dato	Tipo de variable	Acceso del maestro	Acceso del esclavo
Coils	Boolean	Read and write	Read and write
Discrete inputs	Boolean	Read only	Read and write
Holding registers	16 bit unsigned integer	Read and write	Read and write
Input registers	16 bit unsigned integer	Read only	Read and write

a que las variables con mayor tamaño de memoria con las que trabaja son los HR, los cuales son enteros sin signo de 16 bits. De todas maneras existe un modo de implementar variables de tipo flotante con el protocolo. Para ello, se separan los 32 bits de información en dos partes de 16 bits, el extremo superior y el inferior. Se procede a designar un HR para la parte inferior y otro distinto para la parte superior. Notar que estos registros suelen ser contiguos uno del otro dentro del mapa Modbus. Se asignan los valores de ambas partes del numero flotante a los HR mencionados y se los utiliza como cualquier otro registro.

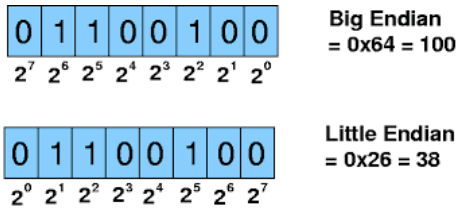


Figura 5.2: En esta imagen se muestran los distintos tipos de endianness.

Es necesario saber como interpretar los HR mencionados, ya sea desde el lado del maestro, como del esclavo. Para esto es necesario conocer el significado de endianness (o extremidad) cuando se habla de almacenar variables en direcciones de memoria. Endianness es el término utilizado para indicar el formato de como se almacenan datos de más de un byte en espacios de memoria. Existen dos tipos de endianness como se muestra en la figura 5.2. Dicho esto, es necesario conocer el tipo de endianness que posee el numero flotante antes de ser descompuesto en 2 registros de 16 bits. Una vez conocido esto se procede a unirlos nuevamente en un espacio de memoria de 32 bits y luego se pueden interpretar correctamente como variables flotantes. En la figura 5.3 se muestra un ejemplo del proceso de interpretación de una variable flotante con la herramienta Modbus Poll.

5.3. Implementación de Modbus

Se buscó que el sistema, en cuanto a las comunicaciones refiere, sea un dispositivo esclavo Modbus RTU. Este debe contar con registros suficientes para poder recibir las señales de control de los cuatro sistemas en su mapa Modbus. Además, debe guardar las señales medidas durante el proceso en el mapa para que puedan ser accedidas por el dispositivo maestro, si así se requiere. La implementación del protocolo, como mínimo debe poder responder a las funciones 16, 06 y 03 de la tabla 5.1.

Se utilizó una biblioteca desarrollada por Alejandro Mera, la cual permite implementar Modbus slave sobre microcontroladores STM32F103xx y además es compatible con el sistema operativo FreeRTOS. Al considerar que en un futuro se proyecta integrar el dispositivo maestro dentro del robot y que este probablemente sea una micro computadora (como lo son raspberry PI, nvidia Jetson, etc), se decidió implementar el protocolo con una interfaz UART. De esta manera simplemente se necesita un conversor USB a UART para implementar la capa física de la comunicación.

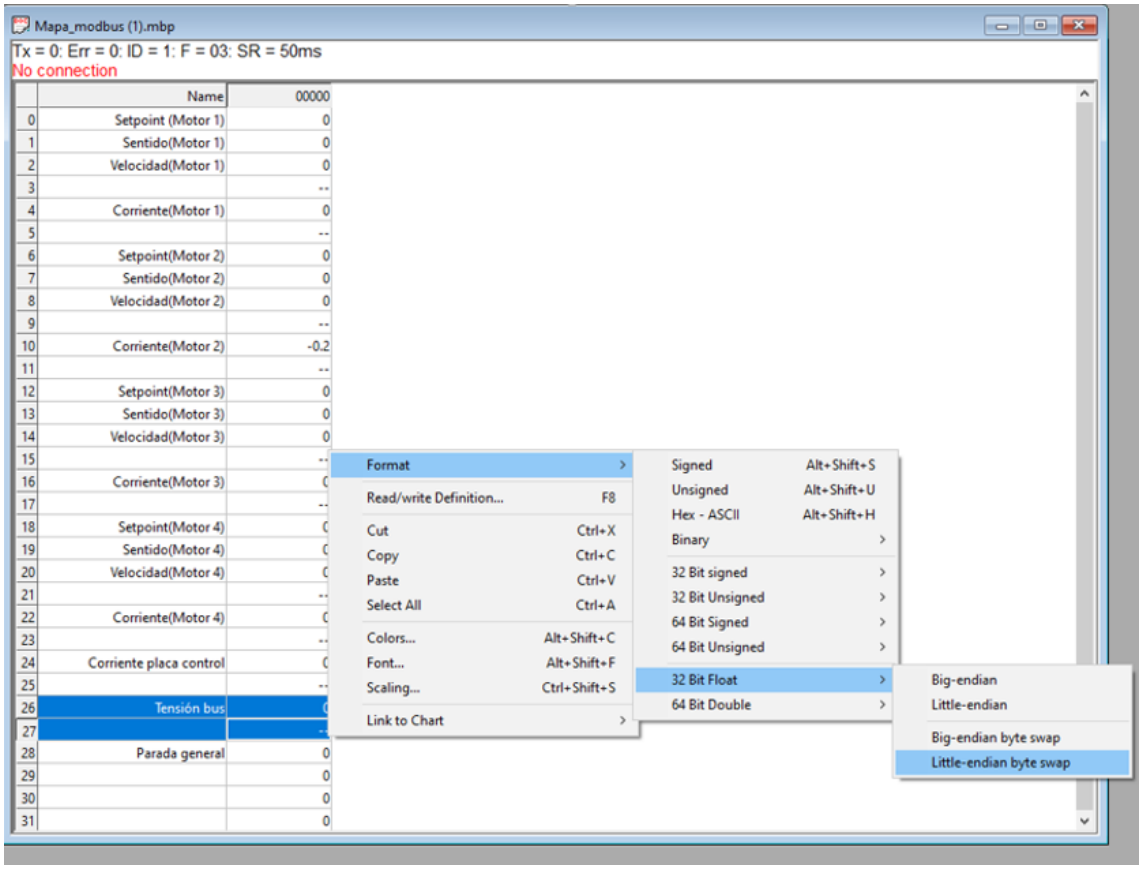


Figura 5.3: En esta imagen se muestran el proceso de conversión de dos holding registers en el mapa Modbus uint16 en una variable flotante.

Se procedió a configurar los parámetros del periférico UART en el microcontrolador:

- N°de UART: 3
- BaudRate: 115200 bit/Seg
- Longitud de palabra: 8 bit
- Bits de parada: 1 bit
- Paridad: Ninguna

En software se implementó el protocolo con las funciones necesarias para la comunicación extraídas de la biblioteca mencionada anteriormente. Se configuraron los parámetros del esclavo Modbus, siendo estos los siguientes:

- Tipo de dispositivo modbus: Esclavo
- Puerto: UART3
- ID: 1
- TimeOut: 1000 mS
- Mapa modbus: ubicado en el arreglo ModbusDATA
- Tamaño del mapa: 32

Se siguió a corroborar el desempeño de las funciones mencionadas de la biblioteca. Para esto se utilizó Modbus Poll, un software que permite emular a un dispositivo maestro Modbus. En dicho software se puede seleccionar una frecuencia, función, ID de esclavo y enviar la petición correspondiente al esclavo. Luego, mediante el proceso de envío de solicitudes con las funciones requeridas (16, 06 y 03) y posterior verificación de las respuestas exitosas obtenidas del dispositivo esclavo, se concluye que la biblioteca elegida es adecuada para la aplicación.

Para organizar la información que se puede leer o escribir, se creó un mapa Modbus, que es básicamente una tabla qué indica que significa la información que tiene cada HR ubicado en el mapa. El mapa del vehículo se muestra en la tabla 5.3. Este cuenta con mediciones de corriente, tensión y velocidad. También cuenta con los setpoints de cada motor, los cuales se encuentran en [Rev/seg]/1000 (con el fin de enviar en un solo HR la señal de referencia con 3 decimales de precisión) y un HR que indica el sentido de giro, siendo 0 sentido horario y 1 antihorario. Asimismo, se cuenta con un HR que es encargado de armar y desarmar el sistema.

Tabla 5.3: Mapa Modbus implementado.

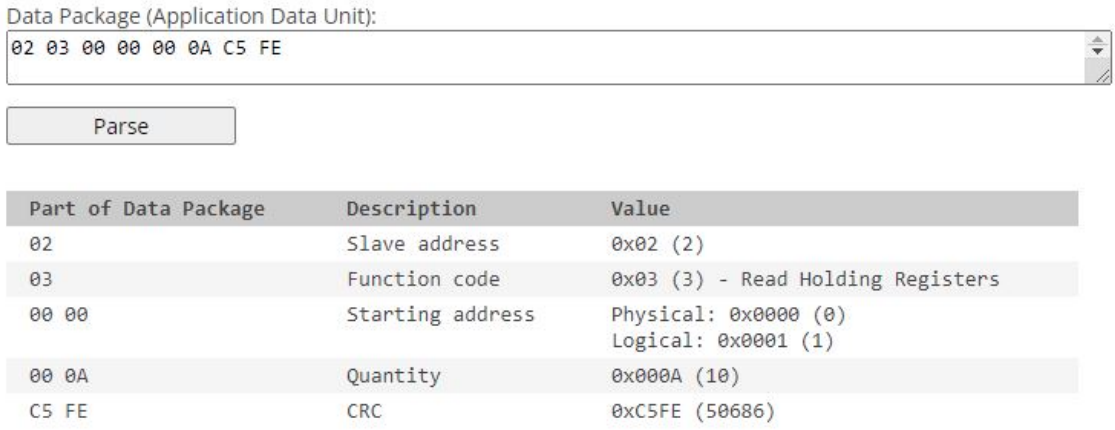
Dirección	Función
0	Setpoint (Motor 1)
1	Sentido (Motor 1)
2	Velocidad (Motor 1) H
3	Velocidad (Motor 1) L
4	Corriente (Motor 1) H
5	Corriente (Motor 1) L
6	Setpoint (Motor 2)
7	Sentido (Motor 2)
8	Velocidad (Motor 2) H
9	Velocidad (Motor 2) L
10	Corriente (Motor 2) H
11	Corriente (Motor 2) L
12	Setpoint (Motor 3)
13	Sentido (Motor 3)
14	Velocidad (Motor 3) H
15	Velocidad (Motor 3) L
16	Corriente (Motor 3) H
17	Corriente (Motor 3) L
18	Setpoint (Motor 4)
19	Sentido (Motor 4)
20	Velocidad (Motor 4) H
21	Velocidad (Motor 4) L
22	Corriente (Motor 4) H
23	Corriente (Motor 4) L
24	Reservado
25	Reservado
26	Tensión de Baterías H
27	Tensión de Baterías L
28	Armado de sistema
29	Reservado
30	Reservado
31	Reservado
32	Reservado

5.4. Estructura de lectura-escritura

A continuación se muestra, a modo de ejemplo, la comunicación Modbus entre dos dispositivos tanto lectura como escritura de HR. El ejemplo se realizó con el programa Modbus poll a modo de dispositivo maestro y con un hardware esclavo Modbus dedicado para la respuesta. Al utilizar dicho programa se observaron los bytes de comunicación, que luego se procesaron con la herramienta online denominada rapid SCADA Modbus Parser.

5.4.1. Lectura de holding registers

Para realizar la lectura de HR's, se armó la trama de solicitud al dispositivo esclavo deseado.


Data Package (Application Data Unit):

02 03 00 00 00 0A C5 FE

Parse

Part of Data Package	Description	Value
02	Slave address	0x02 (2)
03	Function code	0x03 (3) - Read Holding Registers
00 00	Starting address	Physical: 0x0000 (0) Logical: 0x0001 (1)
00 0A	Quantity	0x000A (10)
C5 FE	CRC	0xC5FE (50686)

Figura 5.4: Trama de solicitud de lectura de Holding registers, interpretada por Rapid SCADA Parser.

En la figura 5.4 se muestra como quedó armado el paquete de datos. Se puede observar el significado de cada segmento de la trama en la parte inferior de la figura. En este caso, la solicitud indica que se busca al esclavo con ID 2, a este se le envía la función 03, luego se le indica a partir de qué dirección se requiere la lectura de datos, cuántos registros se buscan leer y por último se envía el CRC calculado.

El dispositivo esclavo recibe esta trama, la procesa y procede a preparar la respuesta para el maestro. En ella debe indicar su ID, la función a la cual está respondiendo, la cantidad de bytes de información, la información y por último el CRC.

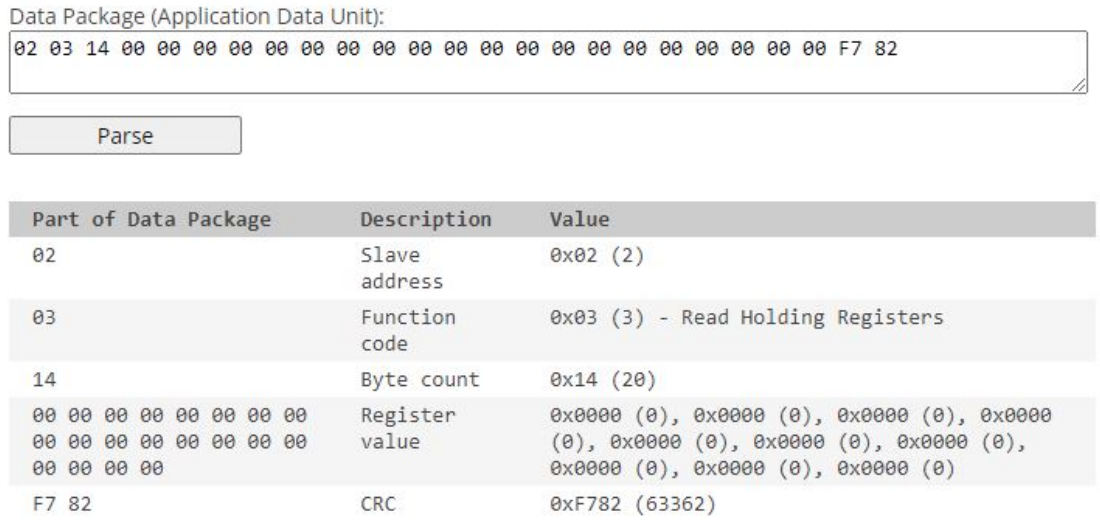
En la figura 5.5 se muestra la respuesta a la solicitud enviada por el maestro inicialmente. El dispositivo esclavo inicia indicando su ID, la cual es 2, indica que va a responder a la función 03 solicitada por el maestro. Procede a informar la cantidad de bytes de información que siguen en la respuesta que en este caso son 20, seguido de dichos bytes (en estos se muestra que el mapa modbus del dispositivo se encuentra con todos sus valores en cero) y finalmente anexa el CRC.

5.4.2. Escritura de holding registers

Para realizar la escritura de HR's, se armó la trama de solicitud al dispositivo esclavo deseado.

Como se muestra en la figura 5.6 se envía la solicitud al esclavo con ID 1, se utiliza la función de código 16 y se indica que se busca escribir 2 HR's a partir del HR número 28. El paquete de datos cuenta con 4 bytes de información y como estos se mandan en orden se observa que se escribe el valor 1 en el HR 28 y 1234 en el HR 29. Finalmente se envía un CRC.

En la figura 5.7 se muestra la respuesta del esclavo. En esta se observa que el esclavo con ID 1 responde a la función 16, modifica a partir del HR 28 una cantidad de 2 HR's y finalmente



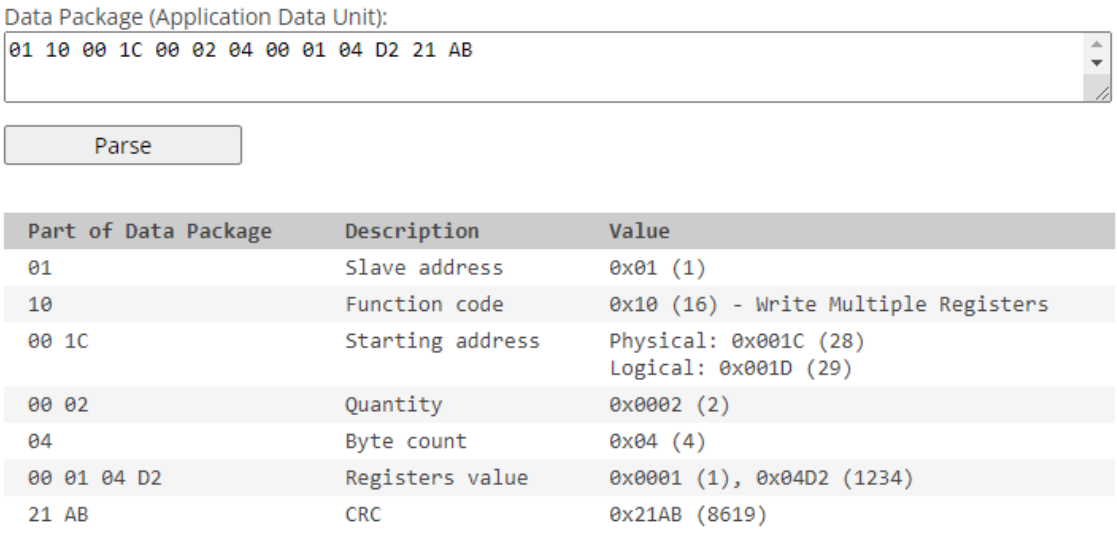
Data Package (Application Data Unit):

02 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F7 82

Parse

Part of Data Package	Description	Value
02	Slave address	0x02 (2)
03	Function code	0x03 (3) - Read Holding Registers
14	Byte count	0x14 (20)
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Register value	0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0)
F7 82	CRC	0xF782 (63362)

Figura 5.5: Respuesta a la solicitud de lectura de HR's, interpretada por Rapid SCADA Parser.



Data Package (Application Data Unit):

01 10 00 1C 00 02 04 00 01 04 D2 21 AB

Parse

Part of Data Package	Description	Value
01	Slave address	0x01 (1)
10	Function code	0x10 (16) - Write Multiple Registers
00 1C	Starting address	Physical: 0x001C (28) Logical: 0x001D (29)
00 02	Quantity	0x0002 (2)
04	Byte count	0x04 (4)
00 01 04 D2	Registers value	0x0001 (1), 0x04D2 (1234)
21 AB	CRC	0x21AB (8619)

Figura 5.6: Solicitud de escritura de multiples HR's, interpretada por Rapid SCADA Parser.

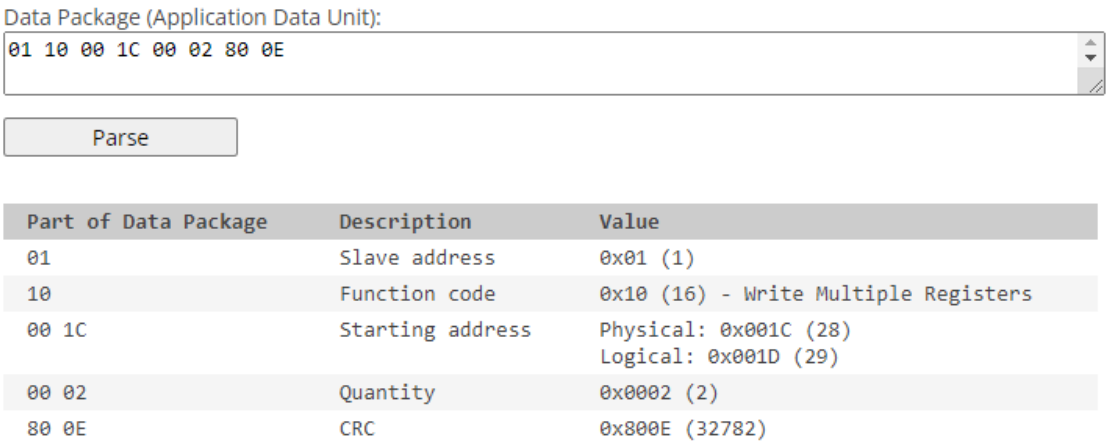
anexa un CRC. El dispositivo debe responder con esta trama a modo de confirmarle al maestro la recepción del mensaje y correcta ejecución de la función.

5.4.3. Excepciones

El sistema es capaz de responder con las excepciones correspondientes que existen en el protocolo, si ocurre algún tipo de error. Las excepciones con las que cuenta el protocolo Modbus RTU son las que se muestran en la tabla 5.4. A modo de ejemplo, en las figuras 5.8 y 5.9 se muestran dos solicitudes con sus correspondientes respuestas, en las cuales se generan excepciones en la comunicación.

5.5. Manual de uso del equipo visto desde un dispositivo maestro

Para la correcta comunicación de un dispositivo maestro con el robot, se debe proceder como se explica a continuación.


Data Package (Application Data Unit):

01 10 00 1C 00 02 80 0E

Parse

Part of Data Package	Description	Value
01	Slave address	0x01 (1)
10	Function code	0x10 (16) - Write Multiple Registers
00 1C	Starting address	Physical: 0x001C (28) Logical: 0x001D (29)
00 02	Quantity	0x0002 (2)
80 0E	CRC	0x800E (32782)

Figura 5.7: Respuesta a la solicitud de escritura de multiples HR's, interpretada por Rapid SCADA Parser.

Tabla 5.4: En la siguiente tabla se muestran los distintos códigos de excepción que posee el protocolo Modbus RTU y su respectiva descripción.

Código de excepción	Nombre	Descripción
01 (0x01)	"Illegal function"	El código de función recibido, no es una acción autorizada por el esclavo.
02 (0x02)	"Illegal data addres"	La dirección de datos recibida por el esclavo, no es una dirección autorizada.
03 (0x03)	"Illegal data value"	El valor en el campo de datos de la solicitud no es un valor autorizado para el esclavo.
04 (0x04)	"Slave device failure"	El esclavo no puede realizar una acción solicitada debido a un error irrecuperable.
05 (0x05)	"Acknowledge"	El esclavo acepta la solicitud pero necesita mucho tiempo para procesarla.
06 (0x06)	"Slave device busy"	El esclavo está ocupado procesando otro comando. El maestro debe enviar la solicitud una vez que el esclavo esté disponible.
07 (0x07)	"Negative acknowledgment"	El esclavo no puede realizar la solicitud de programación enviada por el maestro.
08 (0x08)	"Memory parity error"	El esclavo detecta un error de paridad en la memoria cuando intenta leer la memoria extendida.

5.5.1. Capa física

Para establecer la conexión con la capa física hay que conectar un conversor USB-UART. Es necesario notar que la tierra de la placa de control no es la misma a la que se encuentra en el terminal negativo de las baterías, la alimentación fue intencionalmente aislada por diseño para minimizar ruido eléctrico en el sistema, por lo que se debe tener la precaución de no asociar dichas masas. Para solucionar esto, se diseñó y fabricó un conversor UART a RS-485 como se observa en las figuras 5.10 y 5.11, ya que, este último tiene comunicación diferencial y posee la ventaja de que el bus de datos puede ser utilizado por múltiples dispositivos.

Data Package (Application Data Unit):

```
01 10 00 28 00 0A 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0E 7F
```

Parse

Part of Data Package	Description	Value
01	Slave address	0x01 (1)
10	Function code	0x10 (16) - Write Multiple Registers
00 28	Starting address	Physical: 0x0028 (40) Logical: 0x0029 (41)
00 0A	Quantity	0x000A (10)
14	Byte count	0x14 (20)
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Registers value	0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0), 0x0000 (0)
0E 7F	CRC	0x0E7F (3711)

(a)

Data Package (Application Data Unit):

```
01 90 02 CD C1
```

Parse

Part of Data Package	Description	Value
01	Slave address	0x01 (1)
90	Error code	0x80 + 0x10 (16) - Write Multiple Registers
02	Exception code	[02] ILLEGAL DATA ADDRESS
CD C1	CRC	0xCDC1 (52673)

(b)

Figura 5.8: Solicitud de escritura de HR en direcciones que no existen en el mapa Modbus (5.8a) y respuesta de la excepción correspondiente por parte del controlador (5.8b).

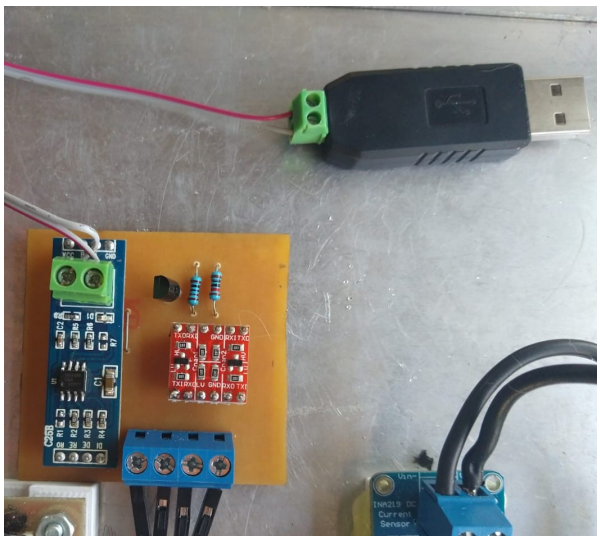


Figura 5.10: placa destinada a convertir UART a RS-485.

Data Package (Application Data Unit):

```
01 33 00 00 00 0A 85 C9
```

Parse

Part of Data Package	Description	Value
01	Slave address	0x01 (1)
33	Function code	0x33 (51)
00 00 00 0A	Data	
85 C9	CRC	0x85C9 (34249)

(a)

Data Package (Application Data Unit):

```
01 33 00 00 00 0A 85 C9
```

Parse

Part of Data Package	Description	Value
01	Slave address	0x01 (1)
B3	Error code	0x80 + 0x33 (51)
01	Exception code	[01] ILLEGAL FUNCTION
94 F0	CRC	0x94F0 (38128)

(b)

Figura 5.9: Solicitud de que el esclavo realice una función inexistente (5.8a) y respuesta de la excepción correspondiente por parte del controlador (5.8b).

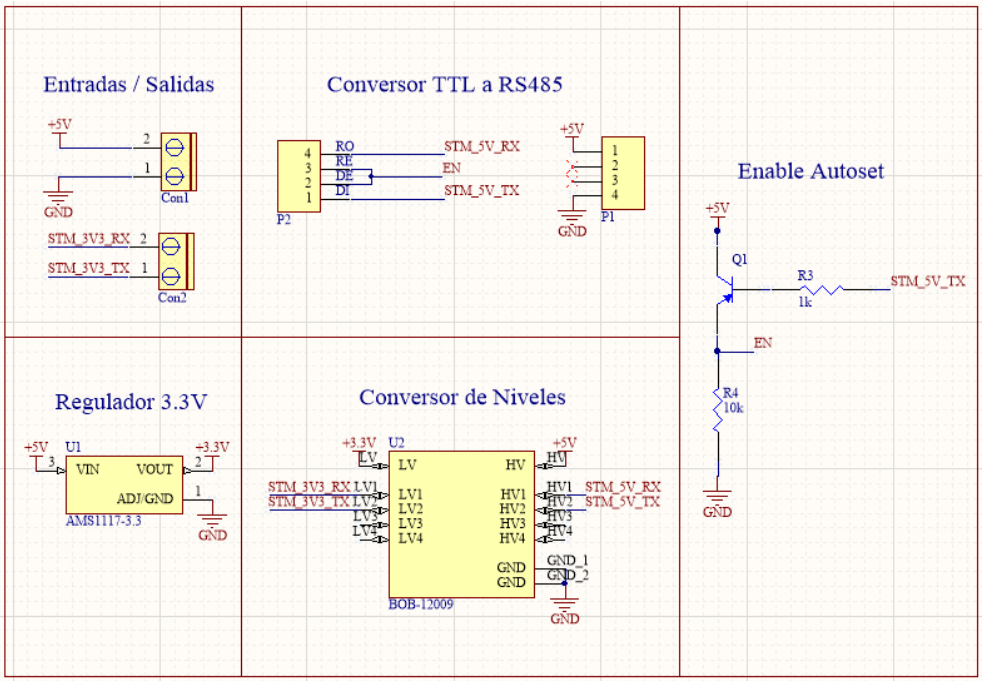


Figura 5.11: Esquemático de la placa conversora de UART a RS-485 con auto-enable.

Otra opción posible para solucionar este problema, es conectar el conversor a través de un

aislador de USB como el de la figura 5.12. El periférico UART del dispositivo maestro se debe configurar de igual manera que el esclavo, como se muestra en la sección 5.3.

Figura 5.12: Dispositivo para aislar eléctricamente la comunicación USB.

5.5.2. Escritura de Setpoints

El vehículo cuenta con un sistema de armado y desarmado de sistema. En caso de que el sistema se encuentre desarmado (HR 28 = 0) no ejecutará los sistemas de control con las señales de referencia que se encuentren en el mapa Modbus. Esto se hizo con el fin de poder frenar el vehículo sin tener que hacer múltiples escrituras y de igual manera poder aplicar todos los setpoints en simultáneo. En caso de que el sistema se encuentre armado (HR 28 = 1) todos los sistemas de control se ejecutaran con las señales de referencia que se encuentren en el mapa.

Modbus permite realizar la modificación de HR en el mapa Modbus pero únicamente de manera consecutiva, por lo que para enviar la escritura de todo el mapa, para modificar los setpoints se generaron compromisos en el sistema, debido a que se produce una reescritura en las mediciones de velocidad y corriente que se encuentran entre estos HR. Por estos motivos se recomienda hacer una escritura a cada par de HR, sentido y módulo de la velocidad por cada motor.

En caso de querer iniciar el movimiento del vehículo, se recomienda enviar cuatro solicitudes de escritura múltiple para especificar cada señal de referencia y luego armar el sistema. En el caso de encontrarse en movimiento continuo del vehículo se deben enviar las señales de referencia deseadas y el sistema debe permanecer armado. En caso de querer detener el vehículo, se recomienda desarmar el sistema y luego enviar a cero todas las señales de referencia.

5.5.3. Lectura de datos

Las lecturas de las mediciones efectuadas por el vehículo se pueden obtener de muchas maneras, pueden realizarse lecturas individuales o multiples de HR. En caso de querer adquirir la información de todo el vehículo, se recomienda enviar una única solicitud de lectura múltiple solicitando la información del mapa Modbus para evitar peticiones innecesarias.

Es necesario destacar que la comunicación funcionará sin importar la lógica de comunicación del dispositivo maestro siempre que cumpla con el protocolo Modbus RTU. Las presentes únicamente se encuentran a modo de recomendación de uso.

Capítulo 6

Pruebas de integración de hardware y software

En este capítulo se explican los procesos experimentales y se presentan los resultados obtenidos. Al inicio se explica como se realizó el proceso de linealización e identificación del sistema, de donde se obtiene el algoritmo necesario para lograr el funcionamiento lineal requerido para el sistema. Seguido a esto, se trata el proceso de discretización, donde se presenta la versión discreta de la función de transferencia. Luego, se realizó la identificación de cada sistema, a partir de ensayos experimentales, donde se determinó cada parámetro que compone a la función de transferencia. Identificado el sistema, se procedió al diseño del controlador. Obtenido el sistema de control, se procedió a integrar todas las partes sobre la plataforma móvil, y a realizar pruebas de funcionalidad. Por último, se determinó el consumo eléctrico de las distintas partes del sistema y además, se estimó la capacidad de las baterías con las que cuenta el vehículo.

6.1. Linealización e identificación del sistema

La identificación y linealización de cada uno de los sistemas se realizó colocando una carga equivalente a la fuerza necesaria que se debe desarrollar para mover el vehículo a una velocidad constante. El objetivo es establecer una dinámica similar a la obtenida cuando el robot opera en condiciones normales, sobre un terreno plano. Para determinar su magnitud, se colocó en uno de los extremos del vehículo un dinamómetro y este se vinculó a un taladro por medio de un hilo que considerado inextensible y de masa despreciable. Luego, se enrolló el hilo logrando que el vehículo se desplace a una velocidad constante y al mismo tiempo se registró la fuerza requerida con el dinamómetro. En la figura 6.1 se observa la configuración que se utilizó en el ensayo.

Posterior a esto, se calculó la fuerza promedio. Considerando que la fuerza necesaria para mover al vehículo se distribuye equitativamente sobre cada uno de los motores, la fuerza que cada uno debe generar en condiciones ideales y estado estacionario es un cuarto de la obtenida del promedio. En la tabla 6.1 se visualizan los resultados que se obtienen del ensayo.

Tabla 6.1: Valores promedio de fuerza que se obtienen para el ensayo tracción del vehículo.

Numero de ensayo	Velocidad promedio rueda [Rev/Seg]	Fuerza promedio[KgF]
1	0,15	2,09
2	0,22	2,14
3	0,14	1,99

A partir de los resultados obtenidos, la fuerza promedio que cada motor debe desarrollar en las condiciones mencionadas es aproximadamente 0,5 KgF. La carga que se vinculó a la rueda a través de un hilo de las mismas condiciones mencionadas con anterioridad.

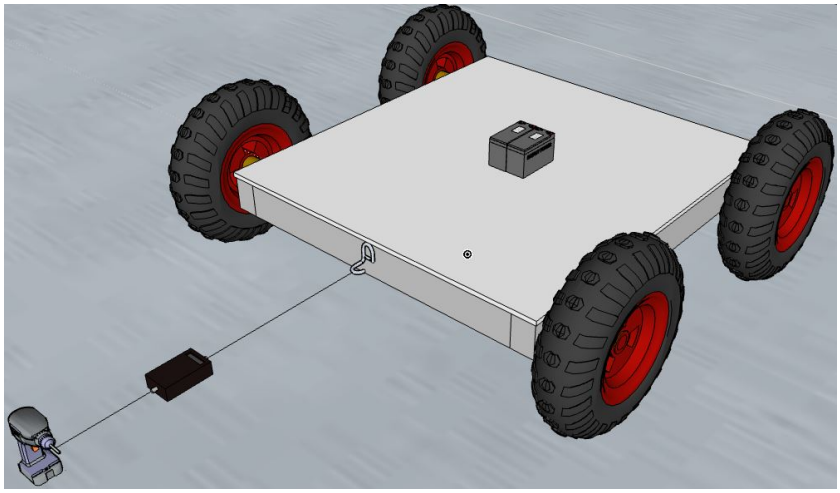


Figura 6.1: Configuración del ensayo para medición de fuerza de tracción.

6.1.1. Diseño e implementación de bloque linealizador

Como se mencionó en la sección 2.2.2, debido a que la respuesta del sistema no es lineal cuando se emplea en conjunto con el variador de velocidad, se creó un algoritmo encargado de linealizarla [6]. Esto se vuelve un requerimiento necesario al aplicar controles de tipo lineal, como el mencionado en la sección 2.3.1. En la figura 6.2 se observa el diagrama en bloques del conjunto.

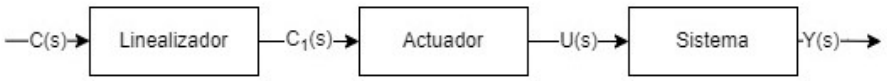


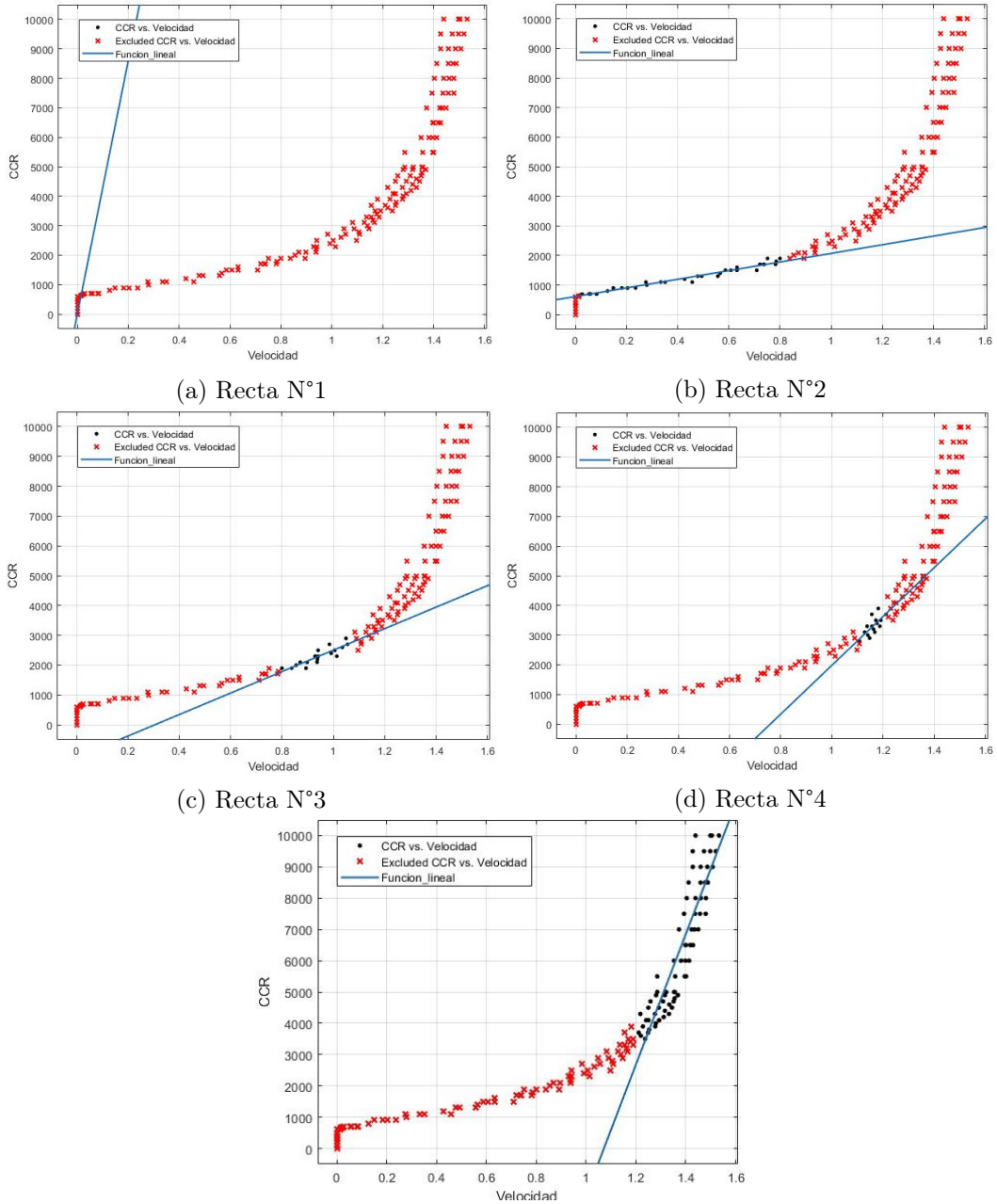
Figura 6.2: Esquema de diagrama en bloques que se sigue para la linealización del sistema

La función del bloque linealizador, es mapear la velocidad deseada en un nivel de tensión especifico definido por la relación inversa entre velocidad y tensión de la figura 2.5. Ya que dicha función debe ser de tipo lineal, fue necesario aplicar una linealización por partes.

Para analizar la conveniencia de los entornos de validez de cada función lineal, se recurrió a la herramienta Curve Fitting de MATLAB. Esta aplicación permite establecer a partir de una nube de puntos correspondiente al conjunto (CCR,Velocidad), la recta que mejor se ajuste aumentando R^2 dentro del rango de conveniencia. Este parámetro cuantifica, dentro de un rango que va de 0 a 1, que tan bien la curva se ajusta a la nube de puntos, siendo el mejor de los casos cuando R^2 es igual a 1. Siguiendo esta idea, se establecieron cinco rectas de distintas pendientes que se observan en la figura 6.3.

Debido a especificaciones detalladas por el fabricante, el reductor se diseña para funcionar a una velocidad nominal de 1,2 Rev/Seg. A su vez, a partir de la función mostrada en la figura 6.4 se observa que a medida que aumenta el valor del CCR, las velocidades de estado estacionario logradas por los motores difieren cada vez más entre si. Esto causa errores cada vez mayores entre la planta promedio, mencionada sección 6.2 y las plantas individuales. Por los motivos mencionados, la velocidad del sistema se limitó a 1,2 Rev/Seg. Esto implica que si el setpoint de velocidad se establece en un valor superior al límite, por software se modifica dicho valor y se establece en el valor especificado. En la figura 6.4 se observa la función linealizadora que se obtuvo.

Una vez definida la función linealizadora, se procedió al diseño del código que la implementa sobre el microcontrolador mediante Simulink. En este entorno de programación visual, se diseñó



(a) Recta N°1

(b) Recta N°2

(c) Recta N°3

(d) Recta N°4

(e) Recta N°5

Figura 6.3: Rectas establecidas a partir de la nube de puntos (CCR,Velocidad) de la relación inversa a la obtenida en la figura 2.5.

un esquema de bloques que luego se exportó como código C hacia el microcontrolador, como se muestra en la figura 6.5. Su modo de funcionamiento consiste en evaluar el valor de velocidad que se tiene en la entrada y a partir de este, generar un determinado valor en la salida. Dicha evaluación se lleva a cabo en el bloque subsecuente a u1, que corresponde a la entrada.

Determinado que bloque condicional actúa, se envía a este el valor de la entrada para determinar el valor de salida a través de la relación lineal que corresponda según la figura 6.4. Dentro de los bloques condición 2 al 7 se encuentran las funciones lineales correspondiente para cada rango y el bloque condición 1 se encuentra la condición de velocidad 0.

Descrito el funcionamiento del esquema en bloques, se procedió a exportar el código hacia el microcontrolador mediante la herramienta Embedded Coder. Dicha herramienta permite generar código C y C++ para procesadores embebidos de manera rápida, permitiendo optimizar el

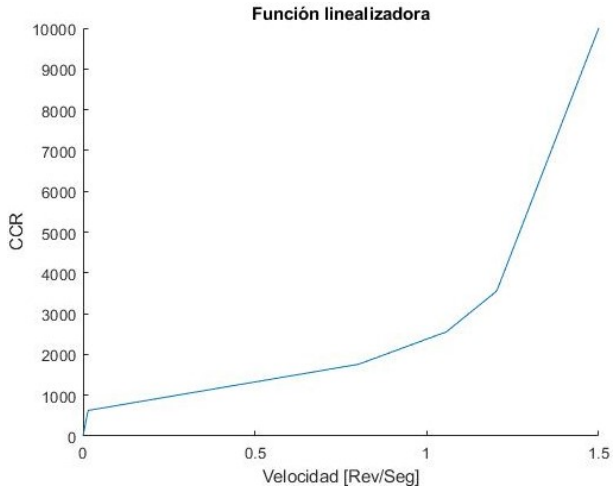


Figura 6.4: Función linealizadora.

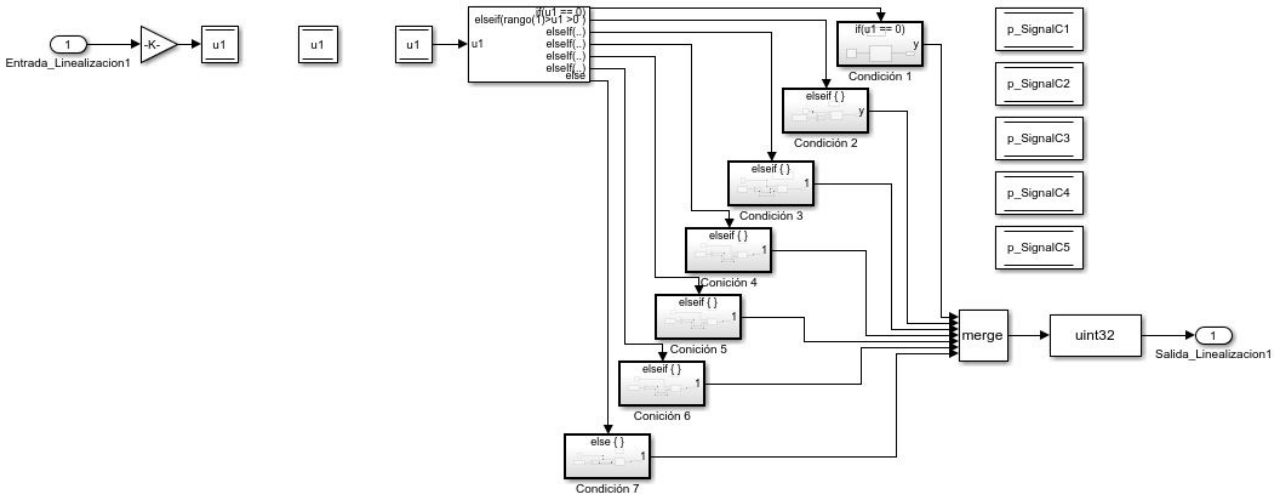


Figura 6.5: Implementación del bloque linealizador en Simulink.

funcionamiento del microcontrolador. Una vez que se selecciona la aplicación mencionada, en el entorno de Simulink aparece una pestaña denominada C Code, dentro de la cual se selecciona Quick Start. Al hacerlo, se abre una ventana en la cual es posible configurar el lenguaje en el que se genera el código, el proveedor del dispositivo (ARM Compatible) y el tipo de procesador (ARM Cortex-M). El código que se generó, se almacenó en dos archivos de extensión .c y .h en una ubicación definida, la cual luego se incluyó en el programa del microcontrolador. En la figura 6.6 se presenta los pasos a seguir (1 al 6) para incluir la carpeta contenedora de los archivos .h y .c, en el path del programa que se diseña y ejecuta en el microcontrolador.

6.2. Identificación del sistema

Para la identificación, se procedió a determinar el valor numérico de cada uno de los parámetros que se involucra en la función de transferencia de los cuatro sistemas de manera experimental. De esta manera se busca estimar una planta promedio del sistema lo suficientemente robusta como para representar las plantas individuales, lo que a su vez va a permitir reemplazar cualquier parte involucrada en el sistema por otro similar sin tener que volver a identificarlo. A continuación se reitera la ecuación 2.21 de la función de transferencia hallada en la sección 2.1.2 y las ecuaciones correspondientes, a modo de facilitar el subsecuente análisis [16] [37] [8] [22].

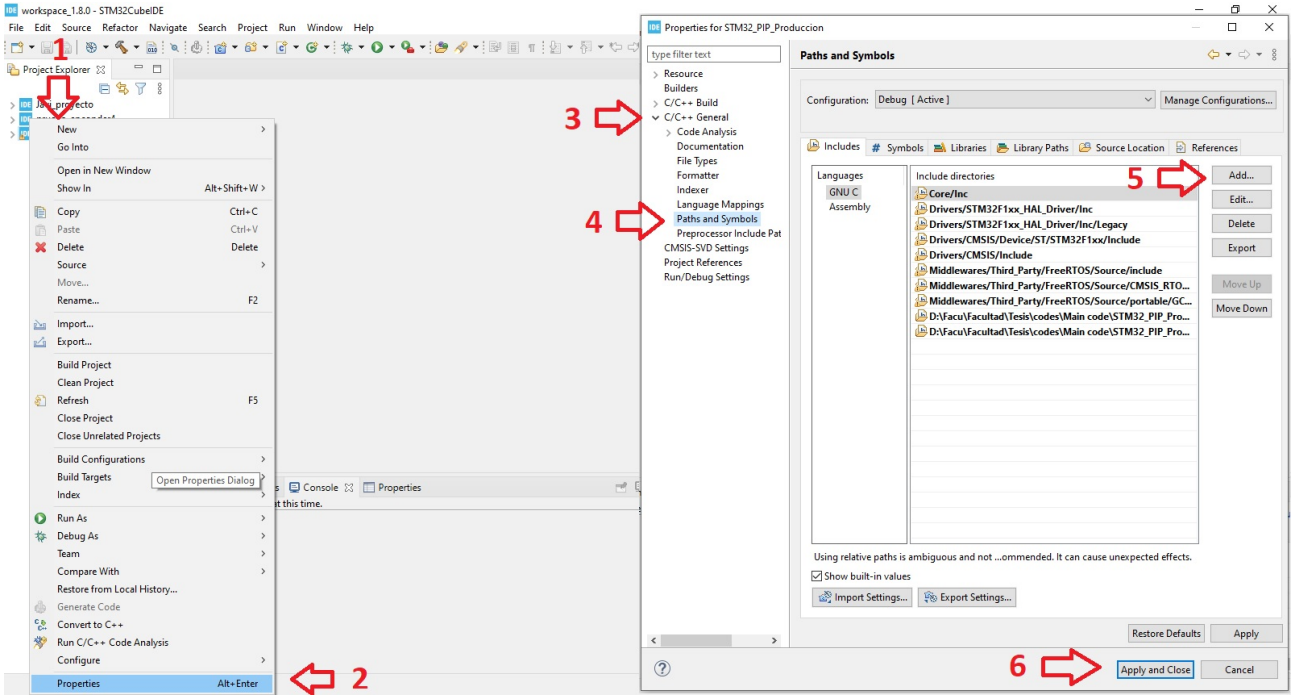


Figura 6.6: En la figura se observan los pasos numerados para incluir el código que se genera a partir de la herramienta Embedded Coder dentro de path del programa.

$$G_E(s) = \frac{k_d}{(L_a s + R_a)(Js + B) + k_d k_f} \tag{6.1}$$

$$v_a(t) = R_a i_a(t) + L_a \dot{i_a}(t) + v_b(t) \tag{6.2}$$

$$\tau_m(t) = k_d i_a(t) \tag{6.3}$$

$$\tau_m(t) = J\ddot{\theta}_L(t) + B\dot{\theta}_L(t) \tag{6.4}$$

$$v_b(t) = k_f \dot{\theta}_L(t) \tag{6.5}$$

Siendo:
R_a: resistencia de los devanados del rotor [Ω]
L_a: Inductancia de los devanados del rotor [mH]
v_a: tensión en bornes del motor [V]
v_b: tensión electromotriz inducida [V]
k_d: constante de par del motor [Nm/A]
k_f: constante de contrafem [Vseg/rad]
J: momento de inercia equivalente del rotor [Kgm^2]
B: coeficiente de fricción dinámica equivalente del rotor [Nms/rad]
τ_m: par desarrollado por el motor [Nm]
i_a: corriente de armadura [A]

6.2.1. Obtención de R_a

Para la obtención de la resistencia que se presenta en los devanados del estator se utiliza el multímetro UNI-T UT890D+. La escala del dispositivo se ajustó en la más baja posible debido a la baja resistividad que presentan los devanados. Se procedió a medir la resistencia que se presenta en bornes del motor. El proceso se repitió múltiples veces variando la posición del rotor y luego promediando todas las mediciones.

6.2.2. Obtención de L_a

Para obtener el valor de inductancia del estator, se observó el tiempo de respuesta de un sistema de primer orden ante una señal conocida y en conjunto con parámetros del sistema definidos específicamente para este ensayo, se halló L_a. Si el rotor se bloquea, la velocidad que desarrolla ante una señal de tensión en la entrada es cero. Debido a esto, la diferencia de potencial contraelectromotriz $v_b(t)$ es igual a cero, por lo que 6.2 está dada por:

$$v_a(t) = R_a i_a(t) + L_a \dot{i_a}(t) \tag{6.6}$$

Luego, sobre uno de los bornes del motor se conectó en serie una resistencia de potencia de valor conocido (4 Ω). En la figura 6.7 se observa el esquema de conexión.

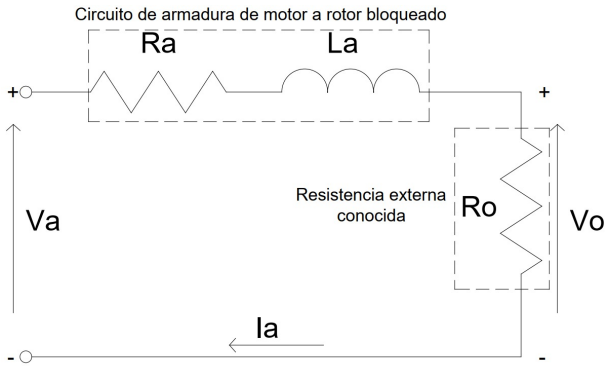


Figura 6.7: Modelo motor de CC con rotor bloqueado, al cual se le coloca una resistencia externa conocida en serie.

Analizando la malla de dicha figura se obtienen las siguientes relaciones:

$$v_a(t) = R_a i_a(t) + L_a \dot{i_a}(t) + v_o(t) \tag{6.7}$$

$$v_o(t) = R_o i_a(t) \tag{6.8}$$

Reemplazando 6.8 en 6.7 y aplicando la transformada de Laplace se obtiene:

$$\frac{V_o(s)}{V_a(s)} = \frac{R_o/L_a}{s + (R_a + R_o)/L_a} \tag{6.9}$$

La ecuación 6.9 corresponde a un sistema de primer orden con un polo en -τ, siendo $\tau = L_a/(R_a + R_o)$. Si la entrada del sistema es una función escalón de magnitud A entonces se tiene que $V_a(s) = A/s$.

$$V_o(s) = \frac{A}{s}\frac{R_o/L_a}{s + 1/\tau} \tag{6.10}$$

Aplicando fracciones parciales 6.10 se expresa como:

$$V_o(s) = \frac{AR_o}{sL_a}(\frac{\tau}{s} + \frac{\tau}{s + 1/\tau}) \tag{6.11}$$

Aplicando la transformada inversa de Laplace a la ecuación 6.11 se obtiene:

$$v_o(t) = \frac{AR_o}{R_a + R_o}(1 - e^{-t/\tau}) \tag{6.12}$$

Todos los parámetros de 6.12 son conocidos a excepción de τ. Por ser un sistema de primer orden, este parámetro corresponde al 63,2 % del valor en estado estacionario para una entrada de tipo escalón. Obtenido este valor a partir de observarlo en el osciloscopio, el valor de L_a está dado por:

$$L_a = \tau(R_o + R_a) \tag{6.13}$$

Para generar una curva y obtener τ, se bloqueó el rotor, se colocó al osciloscopio en modo disparo con las puntas de medición sobre R_a y se ingresó una señal PWM con un ciclo útil de 10 % para evitar dañar los devanados por la elevada corriente debido al rotor bloqueado. Este proceso se repitió múltiples veces y luego se promediaron los valores obtenidos. En la figura 6.8 se observa la respuesta que se obtiene con las condiciones mencionadas para uno de los cuatro sistemas.

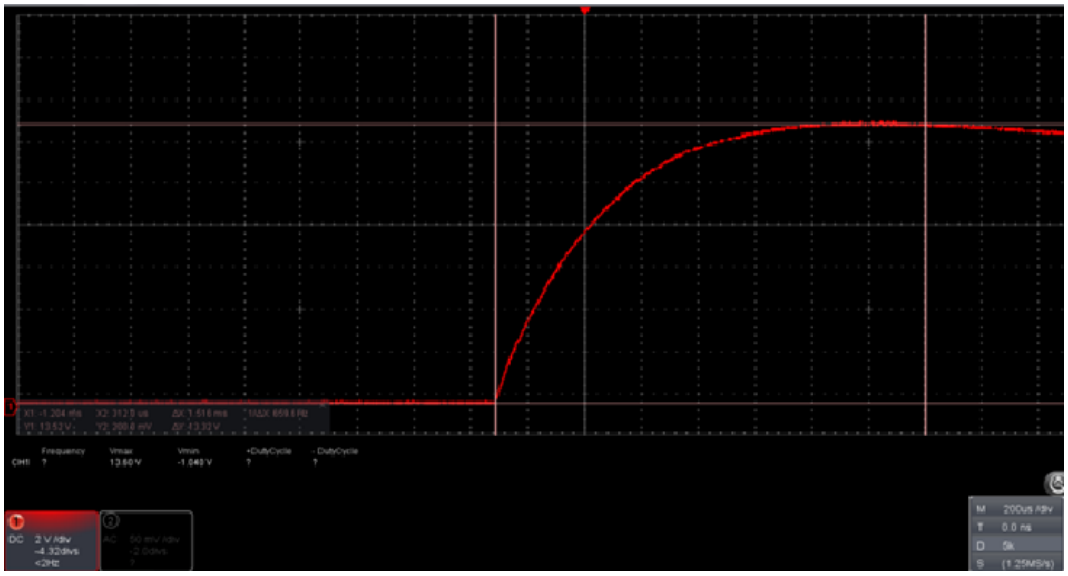


Figura 6.8: Curva de tensión obtenida sobre la resistencia externa conocida, que se coloca en serie al motor. El rotor permanece bloqueado durante el ensayo y la señal PWM de entrada, posee un ciclo útil de 10 %.

6.2.3. Obtención de k_d

Este ensayo consiste en el registro de la corriente de armadura y la fuerza que genera el rotor en condición de bloqueo, empleando la tensión nominal en bornes del motor. La fuerza se registró a través de un dinamómetro y el vínculo entre la periferia de la rueda y el dispositivo mencionado se realizó a través de un hilo (considerado inextensible y de masa despreciable). En la figura 6.9 se observa la configuración que se menciona. De 6.3, k_d se puede representar de la siguiente manera:

$$k_d = \frac{\tau_m}{i_a(t)} \tag{6.14}$$

Conociendo la distancia a la que se vinculó el hilo respecto al eje del motor, el par que desarrolla el motor se puede expresar como la fuerza por la distancia, que en este caso corresponde al radio de la rueda R, obteniéndose:

$$k_d = \frac{FR}{i_a(t)} \tag{6.15}$$

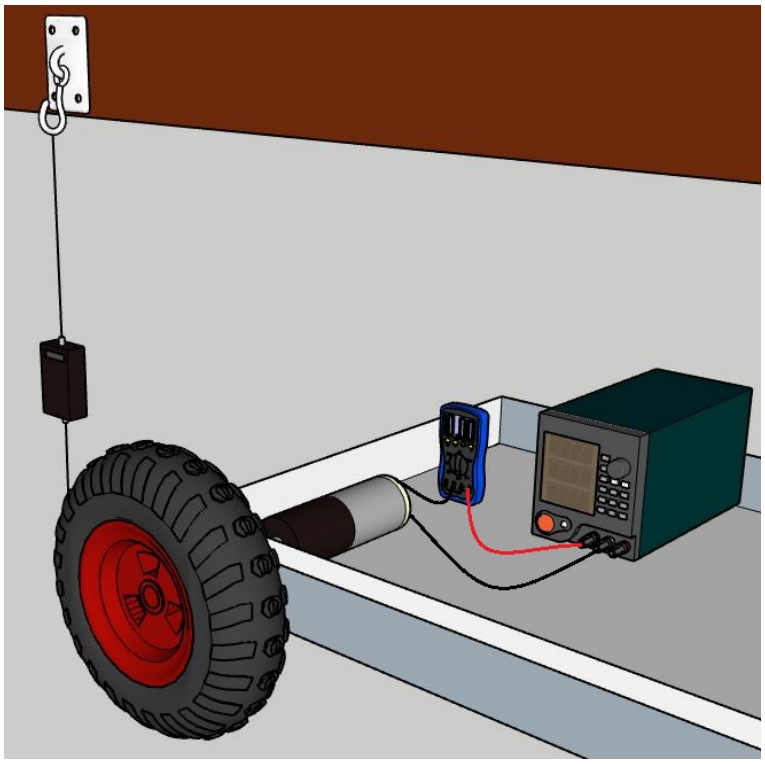


Figura 6.9: Configuración que se emplea para el ensayo en el que se obtiene el parámetro k_d.

Cabe aclarar que este ensayo se realizó en intervalos de tiempo muy cortos para evitar dañar los devanados. El proceso se repitío múltiples veces y luego se promediaron los valores obtenidos.

6.2.4. Obtención de k_f

Para obtener este parámetro, se realizaron múltiples mediciones de la corriente y velocidad sobre el motor ante distintas señales de tipo escalón. El motor pasa de un estado transitorio inicial en el que se acelera, hasta alcanzar un estado estacionario donde la velocidad y la corriente son constantes. Considerando esto, se puede asumir de la ecuación 6.2 que $\dot{i}_a(t) = 0$ y aplicando 6.5 se obtiene:

$$k_f = \frac{v_a(t) - R_a i_a(t)}{\dot{\theta}_L(t)} \tag{6.16}$$

En la figura 6.10 se observa el esquema de conexión que se sigue para el ensayo. Obtenidos los múltiples valores del parámetro, se realizó un promedio con todas las mediciones.

6.2.5. Obtención de B

Para hallar B, fue necesario alcanzar el régimen estable cuando se aplicó una señal en la entrada de tipo escalón. Considerando la ecuación 6.4, ya que la velocidad en régimen permanente es constante, se tiene que la aceleración angular del rotor es igual a cero. Aplicando la ecuación 6.3 se obtiene:

$$K_d i_a(t) = B\dot{\theta}_L(t) \tag{6.17}$$

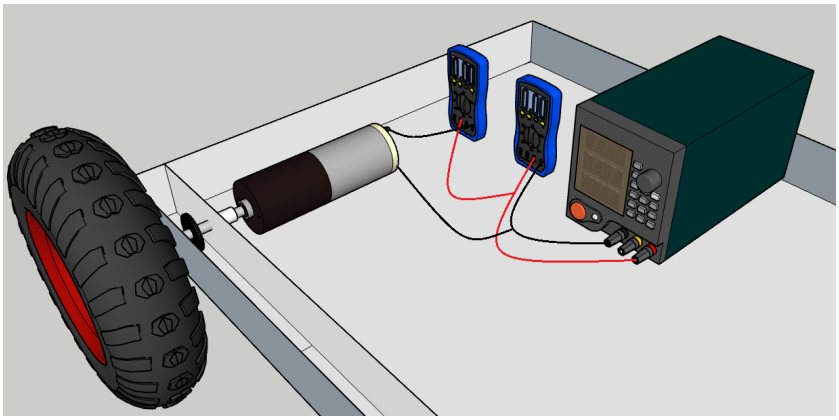


Figura 6.10: Esquema representativo de como se realiza el ensayo para obtener el parámetro k_f de un motor.

Conociendo el valor de la constante contraelectromotriz y registrando los valores de la corriente de armadura y velocidad en estado estacionario para distintos niveles de tensión, se obtiene un conjunto de valores del parámetro a determinar para luego promediarlos.

6.2.6. Obtención de J

El momento de inercia de todo el conjunto se calcula la siguiente ecuación.

$$J = \frac{t_m k_d k_f}{R_a} \tag{6.18}$$

Para aplicar 6.18, es necesario conocer t_m que se denomina tiempo mecánico y establece el tiempo requerido para que al aplicar una señal de tipo escalón en la entrada del sistema, se alcance el 63.2 % de la velocidad del estado estacionario. Debido a esto, es necesario poder medir la velocidad del eje del reductor. En la figura 6.11 se observa la curva que se obtiene sobre uno de los sistemas de la cual se obtiene t_m.

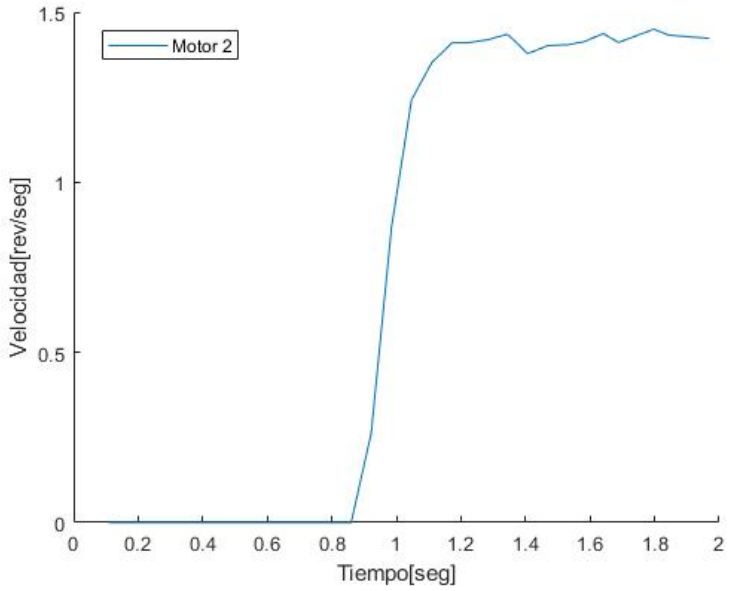


Figura 6.11: Curva de velocidad de la cual se obtiene t_m.

6.2.7. Resultados de la linealización e identificación

Se procedió a comprobar el desempeño del algoritmo diseñado para la linealización. En la figura 6.12 se observa el comportamiento que se obtuvo al implementar el bloque linealizador. Para generar dicha curva, se registraron valores de velocidad en estado estacionario para distintos valores de CCR. De la figura se observa que al aplicar el bloque linealizador, se obtiene un comportamiento similar en todos los motores y además dicho comportamiento se aproxima al lineal que se busca.

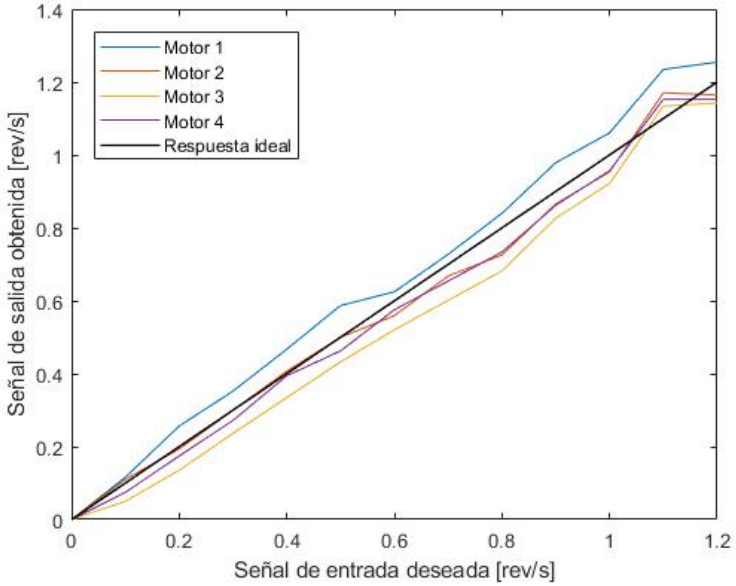


Figura 6.12: Respuesta de los motores ante la función linealizadora

En la figura 6.13 se observa el error relativo porcentual con respecto a la respuesta ideal. De esta última figura se observa un elevado error inicial que disminuye a medida que aumenta la velocidad. El factor que más contribuye a esto son las irregularidades de la fricción según el ángulo de rotación causada por la incorrecta alineación mecánica entre el eje y el rotor. A esto se suma el ángulo de rotación libre, mencionado en la sección 3.2.1, que existe entre el eje y el rotor de cada motor. Estos efectos a bajas velocidades, generan una variación en la velocidad promedio medida.

En la tabla 6.2, se presentan los resultados que se obtuvieron de la identificación de los parámetros que describen la función de transferencia de cada sistema. A su vez, se detalla un valor promedio de cada parámetro para establecer una planta promedio.

Tabla 6.2: Parámetros que se obtienen del ensayo de los motores

-	Motor	1	2	3	4	Promedio
Resistencia devanado	$R_a[\Omega]$	2,99	2,61	2,77	3,13	2,88
Inductancia devanado	$L_a[mH]$	2,28	2,21	2,33	2,45	2,32
Constante par motor	$k_d[Nm/A]$	2,37	2,58	2,56	2,44	2,49
Constante contrafem	$k_f[Vs/rad]$	2,54	2,48	2,60	2,51	2,53
Coeficiente de fricción dinámica	$B[Nms/rad]$	0,020	0,027	0,017	0,017	0,021
Momento de inercia	$J[Kgm^2]$	0,018	0,023	0,022	0,017	0,020
Tiempo de respuesta mecánico	$t_m[ms]$	9,40	9,60	9,20	8,80	9,25

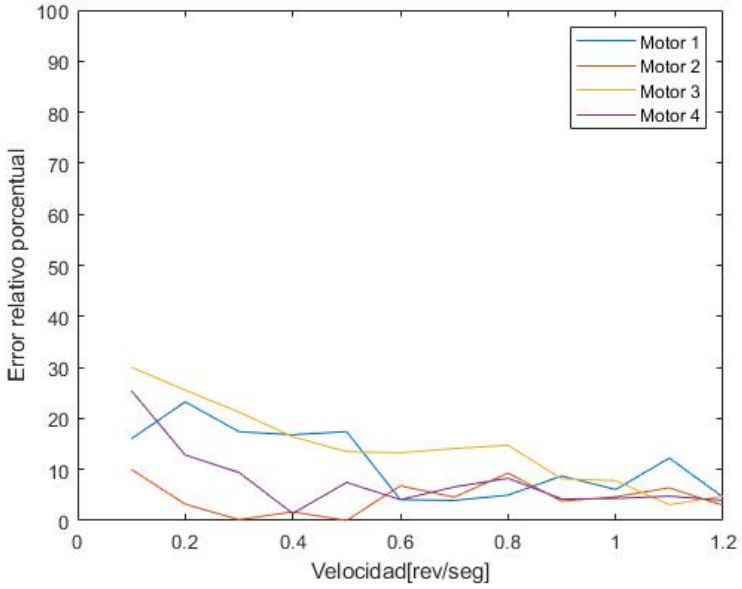


Figura 6.13: Error relativo porcentual obtenido de la planta linealizada, con respecto a la respuesta ideal.

6.3. Discretización del sistema identificado

Debido al modo en el que trabajan los microcontroladores, fue necesario discretizar el sistema. Esto consiste en convertir un sistema expresado en variables de tiempo continuo, en uno equivalente en tiempo discreto operable por el microcontrolador [20]. Para realizar esto último, se utilizó la función *c2d* de MATLAB. Este comando permite convertir un sistema de tiempo continuo, en uno de tiempo discreto partiendo de su función de transferencia o representación en espacio de estados. A su vez, permite establecer un método y periodo de discretización específico. El método de discretización que se utilizó es el mantenedor de orden cero (ZOH), el cual se caracteriza por mantener la señal de entrada constante entre muestra y muestra, coincidiendo con la señal real en cada punto de muestreo.

La frecuencia de muestreo (F_s) es la misma con la que se ejecuta el lazo de control. Para la elección de esta, se tuvieron en cuenta las siguientes consideraciones. Si la F_s es excesivamente alta, aumenta el ancho de banda del conjunto y con esto la susceptibilidad ante el ruido en las mediciones. Esto último puede generar que el controlador aplique acciones de corrección sobre el ruido en si y no sobre variaciones reales en el sistema. Por otra parte si F_s es muy pequeña, el conjunto se ralentiza y se pierde información de las señales que se buscan medir. Esto provoca que el controlador actúe sobre mediciones que no son representativas del comportamiento del conjunto. Por lo mencionado anteriormente, se encontró adecuado que la frecuencia de muestreo del conjunto sea de 200 Hz.

En la ecuación 6.19 se observa la función de transferencia discreta que se obtiene a partir de MATLAB utilizando el periodo de muestreo mencionado y los valores promedios que se presentaron en la tabla 6.2.

$$G[z] = \frac{0{,}1514z + 0{,}02494}{z^2 - 0{,}5503z + 0{,}00201} \tag{6.19}$$

6.4. Validación del modelo

La validación del modelo consiste en determinar si el comportamiento que se obtiene a partir del modelo teórico discreto, obtenido en la ecuación 6.19, es representativo del comportamiento

experimental del sistema observado. Para esto se compararon las respuestas en estado estacionario de ambos ante distintas entradas de tipo escalón.

Para obtener los valores de velocidad en estado estacionario de manera teórica, en Simulink se realizó una simulación de la función de transferencia discreta ante distintas entradas de tipo escalón que representan señales de tensión. En la figura 6.14 se observa el esquema que se diseña en el entorno de desarrollo gráfico.

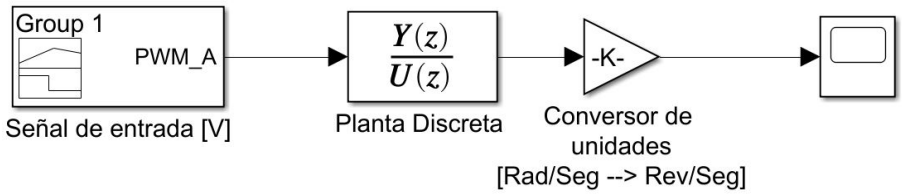


Figura 6.14: Esquema en bloques, con el que se realiza la simulación para validar comportamiento de la planta discreta teórica obtenida.

En la tabla 6.3 se observa el valor real medido para distintas entradas de tipo escalón y el error relativo porcentual existente. Estos valores, se obtuvieron a partir de la figura 2.4 de la sección 2.2.1. Se nota que a muy bajas velocidades existe un error no despreciable, pero este se corrige rápidamente a medida que la velocidad aumenta.

Tabla 6.3: Comparación entre las velocidades obtenidas teóricamente a partir la planta discreta y la velocidad real medida.

Tensión [V]	Velocidad teórica [Rev/Seg]	Velocidad real [Rev/Seg]	Error relativo porcentual [%]
5	0,25	0,31	20,4
10	0,58	0,62	5,8
15	0,92	0,93	0,8
20	1,28	1,24	2,9
24	1,49	1,49	0,3

6.5. Implementación y calibración del controlador

Conocida la función de transferencia discreta promedio, se procedió a la implementación y calibración del PID en Simulink. Partiendo del esquema generado en la figura 6.14, se agrega a este el bloque controlador denominado *controlador PID discreto*. En la figura 6.15 se observa el esquema en bloques.

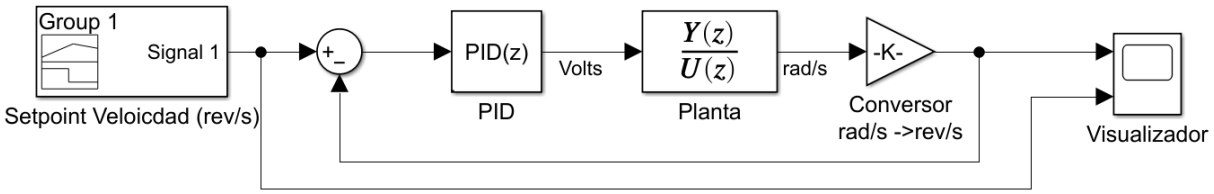


Figura 6.15: Esquema de bloques que se genera en Simulink para calibración virtual de PID

Para la calibración se procedió al uso de la herramienta *Sintonizador de PID*, la cual provee un entorno visual que permite variar el comportamiento transitorio y el tiempo de respuesta del

controlador y visualizar los cambios generados de manera rápida, facilitando su calibración. La variación de estos parámetros implícitamente varía las constantes de proporcionalidad del PID. Obtenido el comportamiento deseado para el controlador, se realizaron pruebas para distintos setpoints. En la figura 6.16 se observa el resultado para una señal de referencia de 1 Rev/Seg.

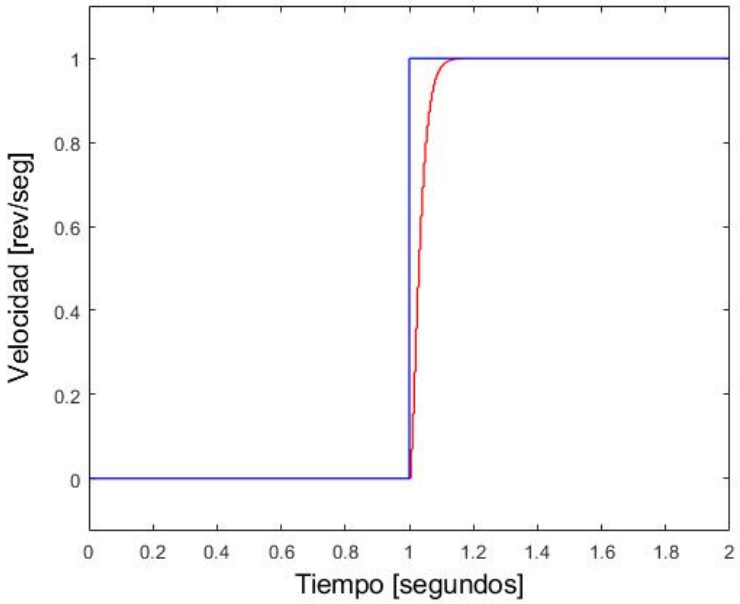


Figura 6.16: Respuesta al escalón del sistema de control simulado, luego de realizar la calibración virtual del PID.

Como se observa en la figura, el controlador lleva la salida del sistema al valor deseado en estado estacionario. Por lo obtenido, se decide realizar la implementación del controlador PID sobre el microcontrolador con los parámetros obtenidos de la simulación, los cuales se observan a continuación.

- $k_p = 2{,}28$
- $k_i = 4{,}5697\text{x}10^{2}$
- $k_d = 2{,}85\text{x}10^{-3}$

Para su implementación en hardware, se procedió a exportarlo en forma de código C como se explicó en la sección 6.1.1. Luego de importar los archivos generados por Simulink, se procedió a realizar la prueba experimental del control. Inicialmente se realizó la prueba individual de cada lazo de control. En esta prueba, se colocó un carga que emula una perturbación de tipo escalón sobre la rueda y se verificó el tiempo que conlleva alcanzar el valor de estado estacionario establecido como referencia. En cada uno de los ensayos con los motores ante distintas cargas, el sistema logró alcanzar el valor deseado en estado estacionario y en promedio, tardó mas de 10 segundos en hacerlo. Luego se procedió a integrar el sistema sobre el vehículo, como se muestra en la figura 6.17. Hecho esto, se realizó una prueba integral de los sistemas sobre una superficie plana con distintos setpoints, mientras se registraron los datos de velocidad. En la figura 6.18 puede observarse el resultado obtenido.

Como se ve en la figura 6.16, para los setpoints de 0,8 Rev/Seg y 1 Rev/Seg la velocidad de las ruedas se aproxima al valor deseado en estado estacionario, aunque se observan pequeñas variaciones. Por otro lado, para el setpoint de 0,4 Rev/Seg se observa que el controlador tiende a oscilar de manera sostenida. Este comportamiento no se asemeja al obtenido en las pruebas individuales de los sistemas. Dicho comportamiento puede deberse a perturbaciones que surgieron

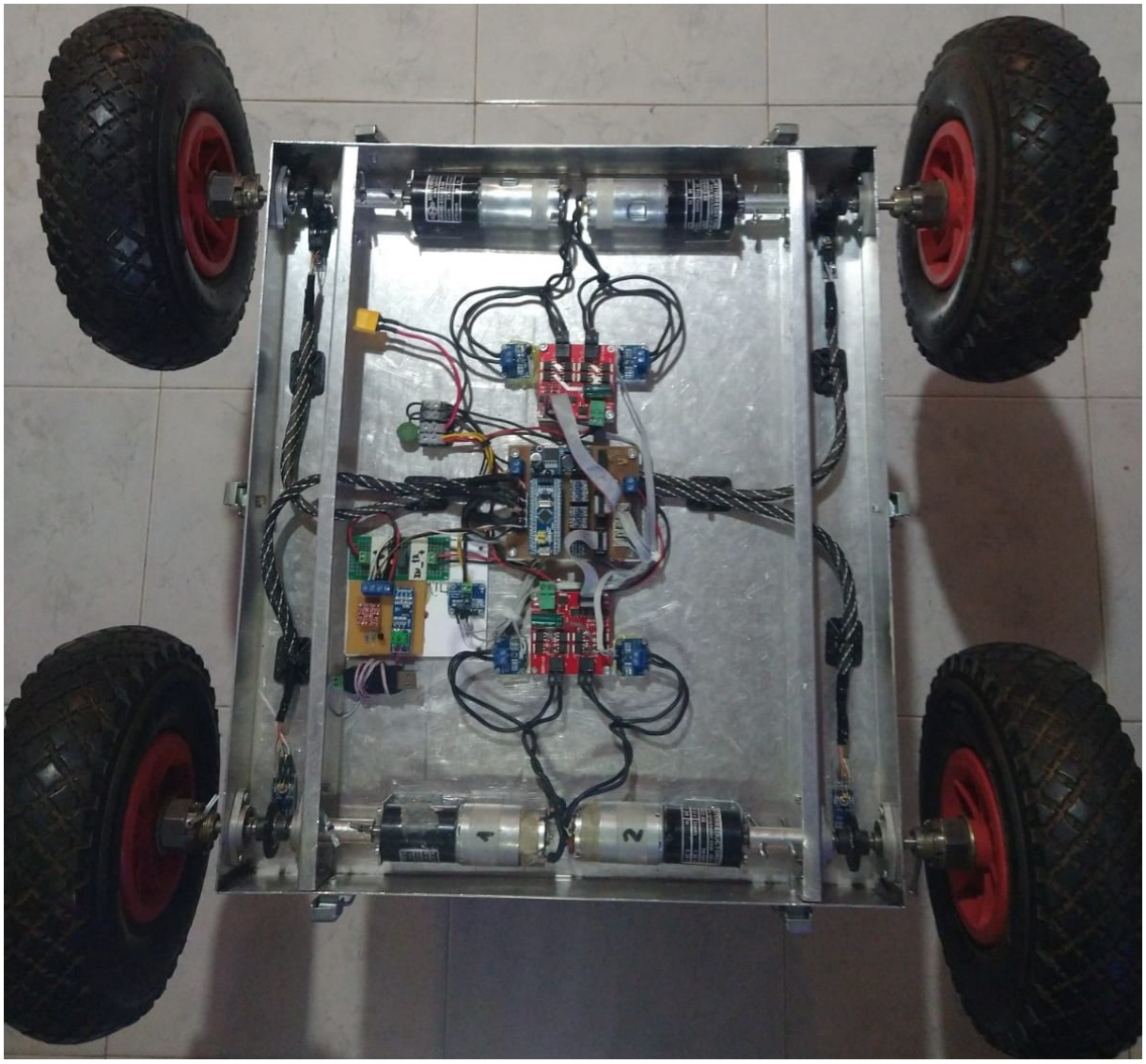


Figura 6.17: Vehículo, sin cobertor superior, con todos los componentes electrónicos integrados. Los diagramas topográfico y de conexionado, pueden observarse en el apéndice B.

al realizar la integración mecánica del vehículo, las cuales no fueron consideradas durante el diseño de la planta. Estas son producto de la relación compleja entre las imperfecciones constructivas del vehículo y como esto hace que las tracciones y fricciones de cada conjunto de rueda y motor actúe como perturbaciones sobre los otros.

Por lo mencionado y ya que se requiere un modo de funcionamiento robusto y operativo en un rango de velocidades amplio, se decidió realizar una calibración experimental del controlador para mejorar los tiempos de repuesta y evitar oscilaciones sostenidas a bajas velocidades.

Para el ajuste experimental primero se colocaron todas las constantes a cero. Luego se incrementó gradualmente la constante proporcional (k_p) hasta obtener una velocidad de respuesta lo más parecida a la deseada. Se observó que aparece un error en estado estacionario, el cual no puede ser eliminado únicamente con esta constante. Además, como es de esperar, si se continuaba incrementando dicha constante, aparecían oscilaciones. Esto mostró que este parámetro no podía seguir aumentando y para llegar al valor de referencia se procedió a modificar el valor de la constante de proporcionalidad integral (k_i). Esta constante se aumentó hasta lograr que el error en estado estacionario se elimine, teniendo en cuenta que si el valor de la constante es elevado, nuevamente aparecían oscilaciones. Por último, se procedió a incrementar la constante

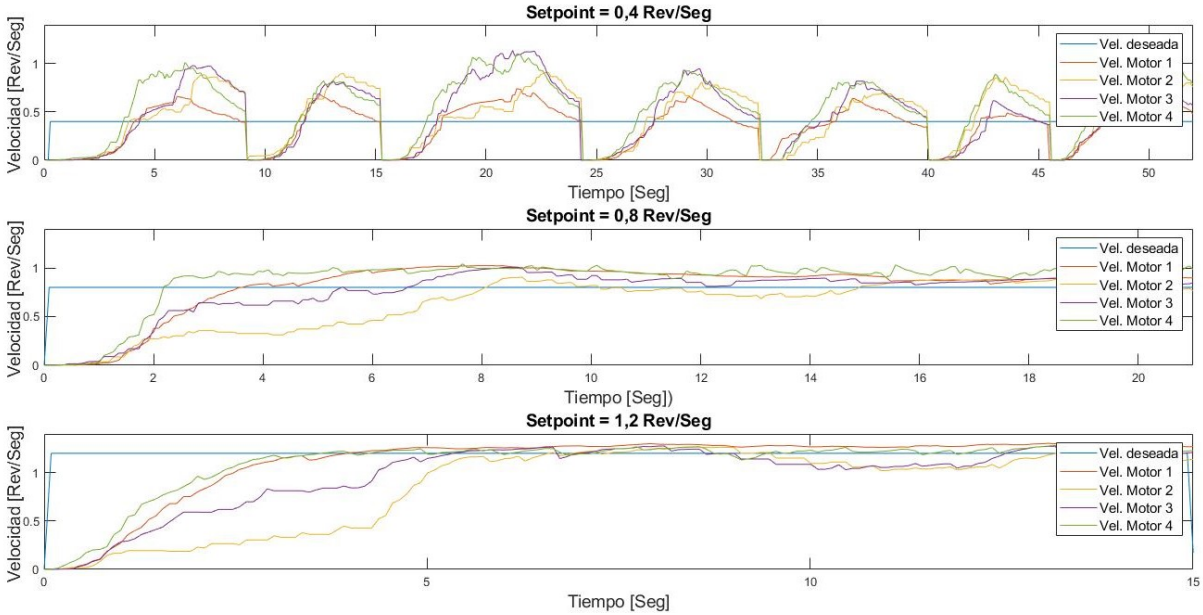


Figura 6.18: Curvas de velocidades obtenidas para los setpoints 0,4 Rev/Seg, 0,8 Rev/Seg y 1 Rev/Seg, con constantes de PID obtenidas mediante la simualción, sobre una superficie plana)

derivativa (k_d) hasta reducir los sobre-picos que se producían al variar el valor de referencia de la velocidad.

Luego de esto, las constantes para el controlador PID son:

- $k_p = 6\text{x}10^2$
- $k_i = 2{,}6\text{x}10^2$
- $k_d = 1{,}58\text{x}10^{-2}$

A continuación se realizó el mismo ensayo con las nuevas constantes obtenidas para el controlador. Como se aprecia en la figura 6.19, el controlador alcanza el equilibrio en menos tiempo, posee una menor variación en estado estacionario y se elimina la oscilación sostenida a bajas velocidades. A partir del resultado del ensayo, si bien las perturbaciones entre sistemas siguen existiendo, el controlador actúa de manera adecuada para reducirlas y llegar al valor deseado. Por lo mencionado, se optó por utilizar las constantes obtenidas experimentalmente.

En la figura 6.20 se observa el error relativo porcentual que existe entre ambos ensayos, cuando el setpoint se establece en 0.8 Rev/Seg. De este se aprecia que con los nuevos coeficientes del controlador, el error converge a cero con una velocidad mayor que en el caso anterior. Además, el error de estado estacionario es menor magnitud y presenta una menor variación respecto al anterior.

6.5.1. Análisis de perturbaciones

Cuando el sistema sufre una perturbación constante externa, esta actúa en forma de torque sobre el rotor. En la figura 6.21 se observa el esquema del sistema en conjunto con el controlador, el cual tiene una señal de referencia de 1 Rev/Seg. Luego de haberse alcanzado la velocidad deseada, el sistema se perturba con una señal de tipo escalón. En la figura 6.22 se observa lo mencionado. Utilizando álgebra de bloques y conociendo el comportamiento lineal que posee el sistema, es posible trasladar el comportamiento que genera la perturbación en la salida y a su vez se puede considerar como una perturbación de tipo escalón. Debido a que el controlador cuenta con acción de control integral, es capaz de eliminar la perturbación en estado estacionario.

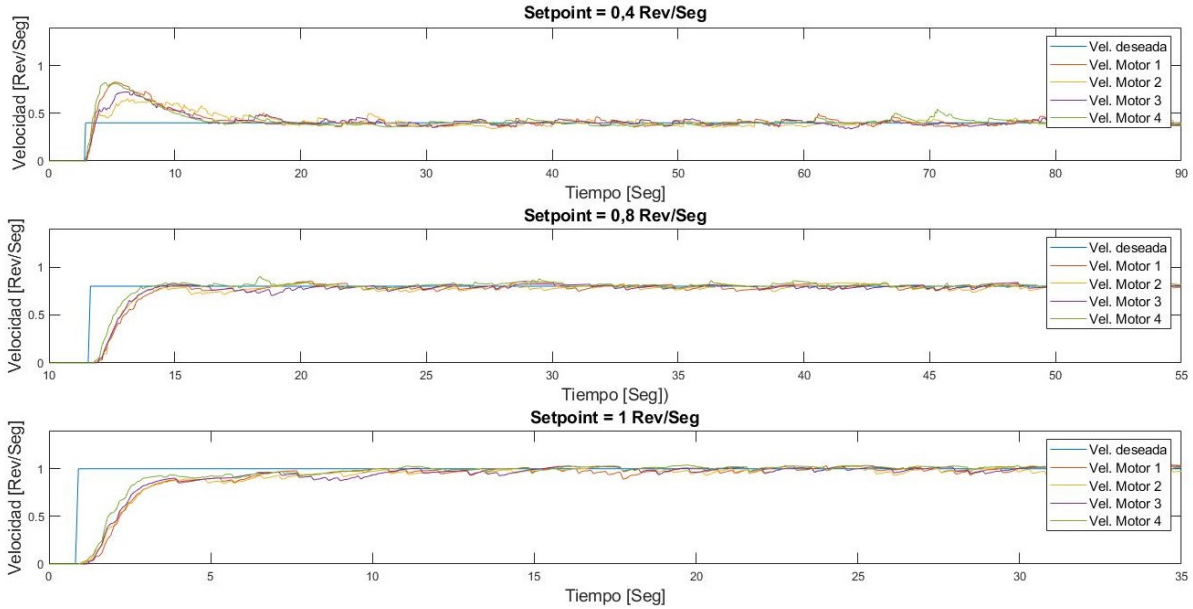


Figura 6.19: Curvas de velocidades obtenidas para los setpoints 0,4 Rev/Seg, 0,8 Rev/Seg y 1 Rev/Seg, con constantes de PID logradas experimentalmente, sobre una superficie plana)

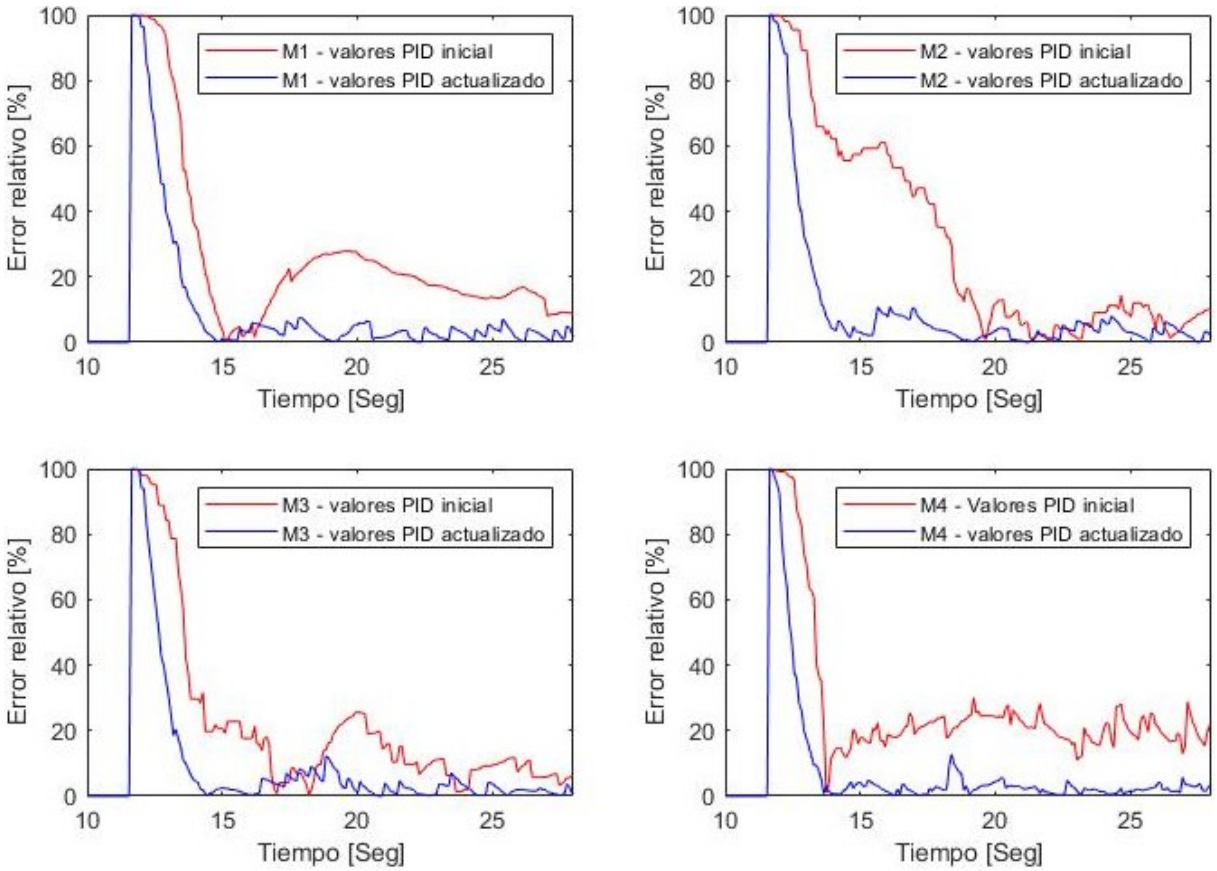


Figura 6.20: Error relativo porcentual existente entre los ensayos integrales con las constantes obtenidas a partir de la simulación y las generadas de manera experimental.

Como se mencionó en la sección 6.1, el algoritmo de linealización se realizó considerando una determinada carga equivalente a la inercia del vehículo. Si existe una perturbación distinta en los motores, la corrección efectuada por el actuador es ligeramente diferente a la planteada ini-

cialmente. Trasladando dicha perturbación al valor de salida del sistema, se puede modelar como una perturbación escalón y como se menciona en esta sección, el controlador puede eliminarla en estado estacionario.

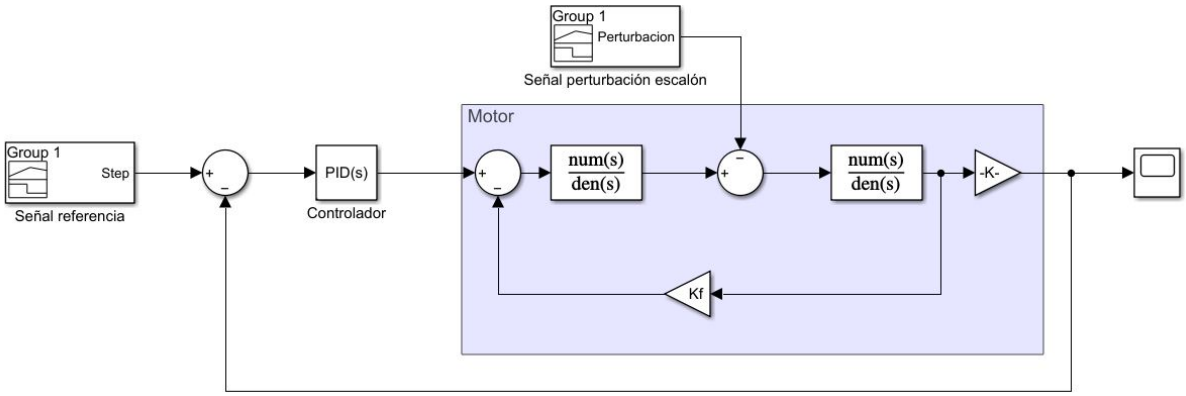


Figura 6.21: Diagrama en bloque del sistema para el análisis de propagación de perturbación de tipo escalón.

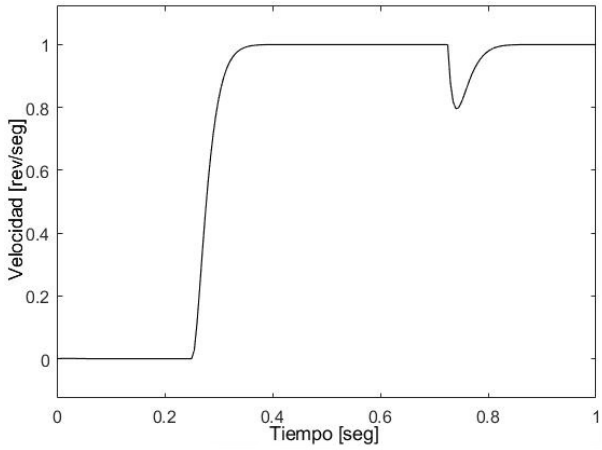


Figura 6.22: Curva de respuesta obtenida ante una perturbación de tipo escalón, en el sistema simulado.

Para validar experimentalmente lo mencionado, se procedió a realizar múltiples ensayos de la siguiente manera. Inicialmente se colocó un setpoint en cada sistema de control del vehículo y se generaron perturbaciones de tipo escalón de distintos valores, mientras se registraba la velocidad antes y después de haber aplicado la perturbación. Para generar una perturbación escalón en el sistema, se situó una lámina de aluminio sobre la rueda y se colocó peso sobre esta generando un torque de fricción. En la figura 6.23 se observa un esquema representativo de la configuración que se utiliza para el ensayo. Cada ensayo se realizó variando entre tres setpoints de velocidad distintos y tres magnitudes de carga distinta. En la tabla 6.4 se presenta el torque que se genera para cada una de las cargas, en las condiciones que el ensayo se llevó a cabo.

En la figura 6.24 se observa el resultado del ensayo cuando la referencia de velocidad establecida es de 0.4 Rev/Seg y posee una carga de 1.5 Kg. Se observa que cuando la carga afecta al sistema (*P o perturbación*, punto rojo del gráfico), la velocidad disminuye su magnitud. Luego comienza a actuar el sistema de control llevando a la velocidad nuevamente al valor de referencia en el que se encontraba inicialmente. Cuando la carga desaparece (*FP o Fin de perturbación*, punto verde del gráfico), la velocidad incrementa su magnitud y el sistema de control nuevamente actúa, llevándola al valor de referencia.

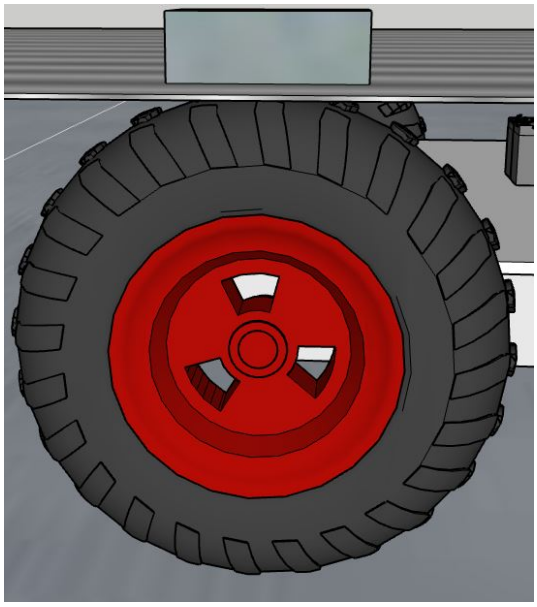


Figura 6.23: Representación gráfica de como se efectúa el ensayo en carga del sistema de control.

Tabla 6.4: Torque de fricción que recibe la rueda a modo de perturbación, en función del peso de la carga agregada.

Masa de la carga [Kg]	Torque de fricción [Nm]
0,5	0,205
1,0	0,411
1,5	0,617

6.5.2. Pruebas de trayectoria a lazo abierto

Con el fin de analizar el comportamiento del sistema de control integral de movimiento sobre el vehículo, se realizó un ensayo a lazo abierto de movimiento en linea recta sobre una superficie plana. Para esto se colocaron señales de referencia iguales en módulo en los cuatro sistemas buscando que el vehículo se traslade en linea recta a velocidad constante. El setpoint de velocidad para el ensayo se estableció en 1 Rev/Seg. Para registrar la información referida a la variación angular de desplazamiento del vehículo, se utilizó una IMU (Inertial Measurement Unit). Durante el ensayo, el vehículo se desplazó una distancia aproximada de 22 metros.

En figura 6.25 se muestra el ángulo yaw del ensayo para las condiciones mencionadas anteriormente. De esta se observa que a pesar de que las ruedas del vehículo se encuentra girando a la velocidad establecida, este presenta una desviación en su trayectoria de aproximadamente 6 grados. Se presume que esto se debe a parámetros constructivos del vehículo. Los datos que se generan en este ensayo, se corresponden con el obtenido en la figura 6.19.

6.6. Esquema de consumo energético

En esta sección se buscó cuantificar la energía que consumen las distintas partes del vehículo y a su vez analizar el estado en el que se encuentran las baterías.

Para obtener una estimación de la capacidad de las baterías se realizó un ensayo de descarga a corriente constante de ambas [3]. Inicialmente se diseñó un registrador capaz de medir corriente, tensión y regular el consumo de corriente. Con el objetivo de mantener la corriente constante mientras la tensión varía se implementó un controlador PI, el cual mantiene la corriente de

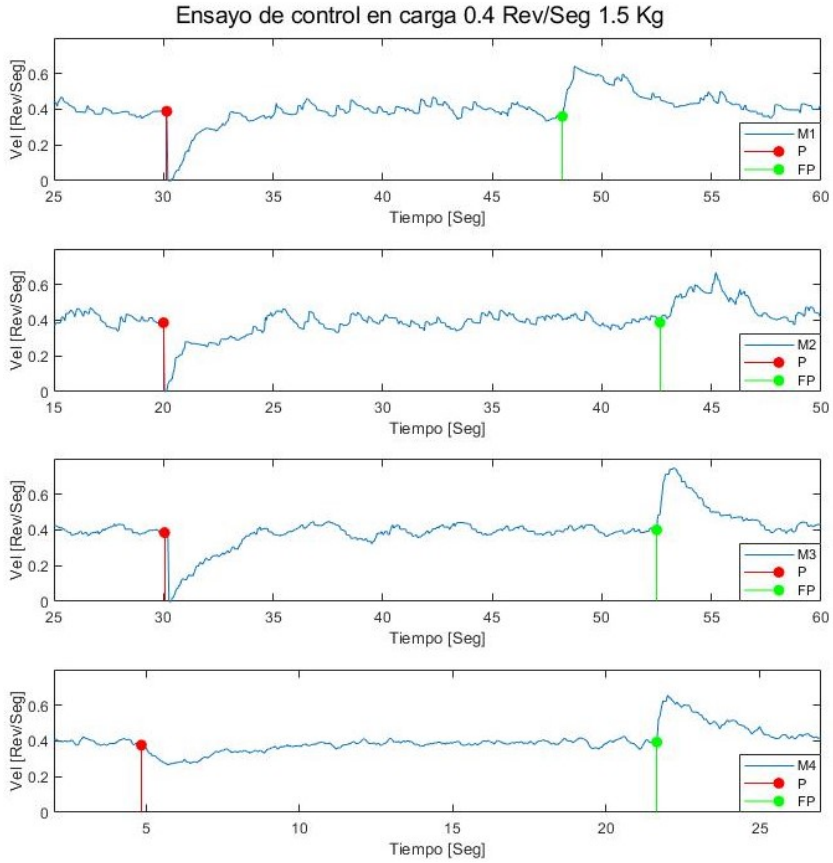


Figura 6.24: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira.

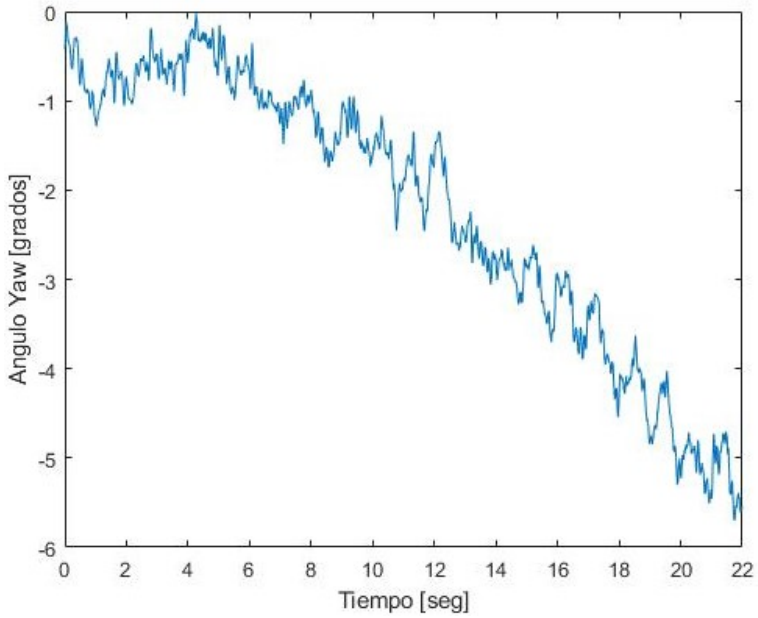
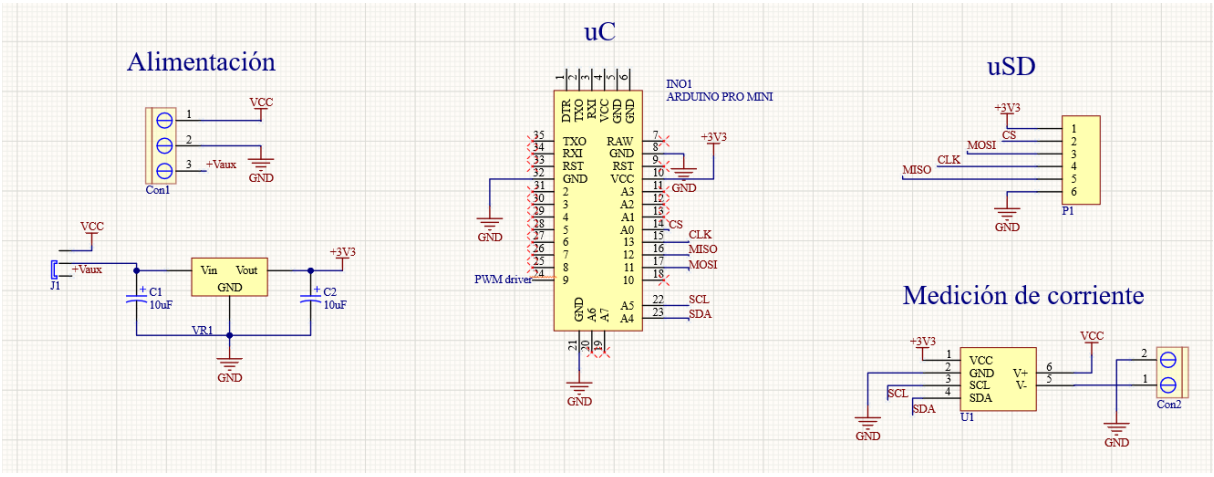
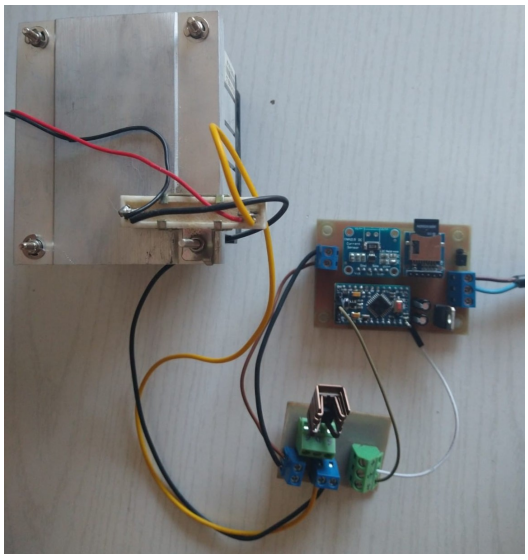


Figura 6.25: Variación del ángulo Yaw que se obtiene cuando se establece un setpoint de 1 Rev/Seg sobre todos los motores y este se mueve en una superficie plana una distancia aproximada de 22 metros

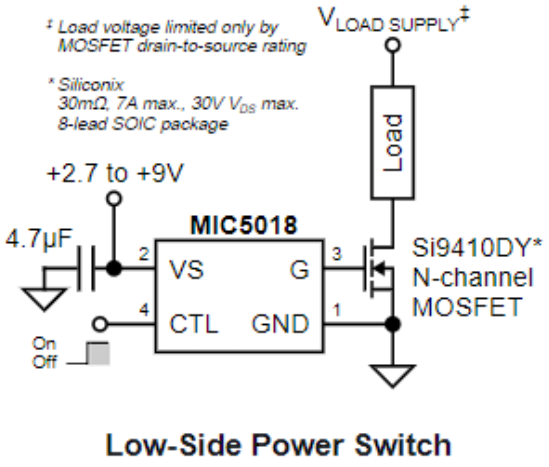
descarga de las baterías constante.



(a) Esquemático diseñado para fabricar el circuito registrador.



(b) Registrador con carga activa con el que se realiza el ensayo de descarga de las baterías.



(c) Circuito con el que se utiliza el CI MIC5018 como driver a drenador abierto, de la carga activa.

Figura 6.26: En la figura se puede observar el registrador con carga activa utilizado para realizar el ensayo de descarga de las baterías.

Para el registrador se utilizó un Arduino pro mini en conjunto con un INA219 y una microSD para poder almacenar los datos. Además, se debió regular la alimentación a 3.3 V. El esquema final del circuito se observa en la figura 6.26a.

Para la carga activa se utilizí un mosfet canal N (IRF540N) con el driver Mic5018, el cual queda a drenador abierto. Para conmutar la masa de la carga que se le coloque, se utilizó una señal de tipo PWM tal como se muestra en la figura 6.26c. De esta manera se logró mantener constante la corriente promedio de descarga. Como carga de la batería, se conectó una resistencia de potencia de 10 Ω a la cual se le colocó un cooler como disipador.

Una vez finalizado el diseño, se procedió a la elaboración de los dispositivos como se describió en la sección 3.3.3, obteniendo como resultado lo que se muestra en la figura 6.26b.

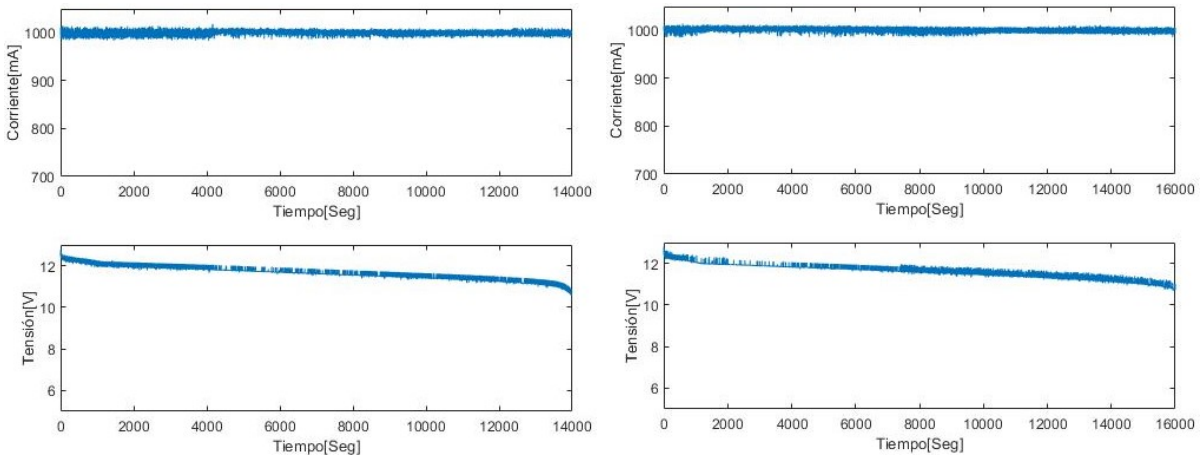
El código que se ejecuta sobre el Arduino se encarga de medir la tensión y corriente (ambos filtrados por software con una media móvil) y guarda los valores en una tarjeta microSD con una estampa de tiempo respecto a cuando se enciende el microcontrolador. También controla la señal PWM con la que conmuta la carga en la batería y ejecuta el lazo de control mencionado.

6.6.1. Ensayo de descarga

En la hoja de datos de la batería se indica que se espera que esta entregue una capacidad de 8.37 AH si se la descarga con una corriente constante a un 10 % de su capacidad nominal.

Para realizar el ensayo de descarga, primero se cargaron al máximo las baterías a una tensión de 14,4 V como propone su hoja de datos, luego se conectó el registrador y la carga activa en la batería dejando que esta se descargue hasta un valor mínimo de 10,5 V (1,75 V por celda).

En la figura 6.27 se observan los resultados obtenidos. Luego, se procede a integrar la corriente empleado la ecuación 6.20 para obtener la capacidad.



(a) (b)

Figura 6.27: En esta figura se muestra los datos de tensión y corriente registrados durante el ensayo de descarga de cada batería. Los gráficos del conjunto (a) corresponden a la batería N°1 y los del conjunto (b) a la N°2.

$$C = \int_{t_1}^{t_2} i(t)\, dt \tag{6.20}$$

Donde:
C : Capacidad de la batería
$i(t)$: Corriente de descarga
t_1 : Tiempo inicial del ensayo
t_2 : Tiempo final del ensayo

Finalmente se obtuvieron las capacidades estimadas de las baterías que brindan una idea del estado en el que se encuentran.

- Capacidad aproximada de Batería 1: 3.88 AH.
- Capacidad aproximada de Batería 2: 4.51 AH.

Al conectar el conjunto en serie, se obtiene un banco de baterías con el doble de tensión cuya capacidad es de 3.88 AH. Se puede observar que la capacidad de las baterías del vehículo se encuentra reducida hasta en un 54 % de lo esperado de un conjunto de baterías nuevas.

6.6.2. Figura de consumo de cada subsistema electrónico

El vehículo cuenta con dos grandes subsistemas, el primero es el sistema de control que incluye placa de control y todos los sensores del vehículo y el segundo es el de movimiento el cual incluye

los variadores de velocidad y los motores. Se procedió a medir el consumo eléctrico de cada uno como se muestra en la tabla 6.5.

Tabla 6.5: En la siguiente tabla se muestra la figura de consumo eléctrico de las partes individuales y luego del sistema en su conjunto en condiciones nominales.

Consumos	Medición [mA]	Cantidad	Total [mA]
INA 219	0,5	6	3
Placa de control	51,8	1	51,8
ESC	8,7	2	17,4
Encoder	3,7	4	14,8
Sistema de control	87	1	87
Motores (nominal)	1500	4	6000
Sistema completo (nominal)	6087	1	6087

De la tabla se observa que la etapa asociada al control tiene un consumo despreciable frente a la etapa de movimiento del vehículo.

Capítulo 7

Conclusiones y mejoras

Este capítulo se estructura en dos secciones. En la primera de ellas se realiza una recopilación de lo que se hizo, de los resultados obtenidos y en base a los objetivos planteados se establece una conclusión. En la segunda sección, se plantean los posibles desarrollos a futuro para continuar en el avance hacia el logro de un vehículo autónomo.

7.1. Recapitulación y conclusión

El presente proyecto explica cómo se llevó adelante el desarrollo e implementación de un sistema de control de velocidad para los motores de un vehículo 4WD destinado a navegar en un ambiente frutícola. El desarrollo partió por establecer cómo se compone el sistema, a partir del cual se obtuvo un modelo teórico que lo describe, para luego identificarlo. La identificación del sistema consistió en determinar cada uno de los parámetros físicos que lo compone a través de ensayos experimentales. El período de pandemia que se cursaba al momento de realizar dichos ensayos impidió acceder a mejores recursos, lo que implicó diseñar y realizar cada ensayo con los recursos que se contaba.

En paralelo a lo anterior, se diseñó y fabricó una placa PCB que integra todos los componentes electrónicos necesarios para lograr un sistema de control funcional. Durante su testeo, se detectó que el ruido electromagnético generado por la conmutación de los estatores se inducía en la etapa de control, lo que causaba mediciones erróneas de la velocidad. Esta fue una de las principales razones por las cuales se realizó un rediseño de la placa. El rediseño implementado tiene la particularidad de que aísla eléctricamente la etapa de potencia y la de control para eliminar el ruido inducido. Respecto a los componentes empleados para el desarrollo de la placa, se considera importante destacar la potencia de procesamiento y recursos que ofrece el microcontrolador STM32F103C8T6. Una de sus características más destacable es la posibilidad de utilizar FreeRTOS, un tipo de RTOS que brindó gran flexibilidad para atomizar tareas sin comprometer el flujo de código y facilidad para incorporar nuevas tareas. Previamente no se contaba con experiencia implementando algún tipo de RTOS, por lo que fue necesario atravesar una curva de aprendizaje.

La arquitectura del código se segmentó en distintas tareas, las cuales son ejecutadas en concurrencia por el planificador de tareas. Este las ejecuta en base a la prioridad que se le asignó, la cual depende de su importancia dentro del flujo general del código. Cada una de ellas tiene una funcionalidad particular dentro de la implementación del sistema de control. Para la comunicación entre el vehículo y un dispositivo maestro, se implementó el protocolo Modbus RTU sobre la capa fisica RS-485. De manera tal que el sistema de control es un esclavo Modbus y todos sus setpoints y variables medidas, se encuentran en el mapa Modbus del dispositivo. A modo de evitar que existan bloqueos de flujo de ejecución y no pueda retornar a su modo de operación normal, se utilizaron diversas técnicas. Entre estas, se encuentra el control de errores, tiempos de espera máximos (o Timeout) en las funciones HAL y el Watchdog timer. El diseño

del controlador ejecutado en la tarea control, se realizó en Simulink en conjunto con MATLAB, mediante una estrategia de control PID. Con dichas herramientas, se generó código C en formato del librería el cual se implementó sobre la tarea. Esto último trae grandes beneficios, ya que en caso de la planta con la se calibró el controlador varíe por algún motivo, solo es necesario generar nuevamente las librerías y reprogramar el microcontrolador. Finalmente, para llevar un control de la documentación y cambios en el software, el desarrollo fue llevado a cabo en la plataforma de Github, brindando control de versiones y facilidad para programar en un entorno colaborativo.

Definido todo lo anterior, se realizó una integración completa sobre la plataforma móvil, obteniéndose un vehículo funcional. Se realizaron mediciones de los consumos en condiciones nominales y se estimó la capacidad remanente en ambas baterías, obteniéndose que estas se encuentran deterioradas en un 50 % al compararlas con los valores nominales de fábrica. Por esto último, se recomienda el cambio de las mismas para prolongar la autonomía del vehículo. Con el fin de evitar el deterioro de las baterías, se implementó un corte de energía de motores por software, el cual impide drenar las baterías, si ya se encuentran descargadas.

Al implementar el lazo de control sobre el hardware, se realizaron pruebas individuales de cada lazo, donde se observó que el sistema de control elimina la perturbación, en un tiempo en el orden de los 10 segundos. Luego se procedió a realizar la prueba integral donde se colocó al vehículo sobre una superficie plana y se estableció un setpoint en los cuatro sistemas de control. Se detectó que para bajas velocidades, el sistema se vuelve oscilatorio y a velocidades medias y altas, el tiempo de respuesta es mayor al observado en los ensayos individuales. En estado estacionario, se ve una pequeña oscilación remanente. Al integrar el vehículo y los cuatro sistemas de control, se tiene que estos se encuentran vinculados mecánicamente entre sí, a través del chasis. Esto ocasiona que la acción del control sobre uno de los motores, pueda perturbar alguna de las velocidades de los demás sistemas. Por lo mencionado, se decidió realizar un ajuste experimental de los coeficientes de los controladores buscando reducir el tiempo de respuesta obtenido y una mayor robustez en las acciones de control de cada sistema. Con los coeficientes nuevos obtenidos, se eliminó la oscilación existente en estado estacionario, se redujo el tiempo de respuesta del controlador y se logró que funcione de manera correcta a baja velocidad. De esta manera, si bien las perturbaciones entre sistemas siguen existiendo, el controlador actúa de manera adecuada para reducirlas y llegar al valor deseado. Para validar los nuevos coeficientes, se repitieron las pruebas individuales, en las cuales se observó un mejor desempeño de los sistemas.

Durante la última prueba integral, se registraron los datos de una IMU con los cuales se mide la desviación del vehículo cuando todas las ruedas giran a la misma velocidad. Se obtuvo que a una velocidad de 1 Rev/Seg, el vehículo se desvía a una tasa de 6° cada 22 metros recorridos. Al observar que la velocidad se mantuvo relativamente constante durante la prueba, se considera que la causa de la variación en la dirección se debe a diversas características mecánicas del vehículo. Es necesario resaltar dos aspectos: que dichas características pueden ser mejoradas, resultando en una reducción de la desviación obtenida, y por otro lado que la prueba era de lazo abierto. El cierre del lazo, será implementado en futuros desarrollos, el cual eventualmente corregirá dichos desvíos actuando sobre la velocidad de cada motor.

En lo personal, realizar un proyecto integrador profesional que abarque la integración de hardware y software, representó un verdadero desafío. Durante el acondicionamiento mecánico, el desafío se centró en realizar las modificaciones mecánicas necesarias, para que el sistema real se asemeje los mas posible al sistema que logramos modelar matemáticamente. Luego, durante la implementación en PCB, la dificultad se centró en eliminar ruido generado por la conmutación de los motores. Al encontrarse el sistema de control, en un ambiente altamente ruidoso, fue necesario hallar el modo de que este último, no afecte a las mediciones ni al desempeño del sistema de control. En cuanto a la implementación sobre el microcontrolador, tanto hardware como software, fue necesario utilizar una correcta administración de recursos, ya que estos fueron empleados en su mayor parte. A esto se suma la implementación de nuevas tecnologías, como FreeRTOS y STM32. Por último, uno a uno fuimos encontrando las soluciones para cada uno de

los desafíos que fueron surgiendo y pudimos efectivamente brindarle movilidad y un sistema de control de velocidad 4WD al vehículo.

7.2. Mejoras y desarrollos a futuro

En el presente trabajo se diseñó e implementó un sistema de control de velocidad funcional, para los cuatro motores de un robot 4WD. El mismo es capaz de recibir setpoints de velocidad de un dispositivo maestro a través de una interfaz Modbus, establecerlos y mantenerlos. Luego se implementó sobre la plataforma móvil, brindada por el grupo de investigación de vehículos y sistemas inteligentes. Cabe destacar, que el presente trabajo va a actuar como punto de partida para múltiples desarrollos vinculados a la percepción del ambiente en entornos frutícolas (generación de datasets y procesamiento) y el manejo autónomo de maquinaria en la fruticultura.

Si bien se trabajó sobre la parte mecánica del robot, existen posibilidades de mejorarla aún más. Es necesario reforzar la estructura del chasis del vehículo, con el fin de prolongar la vida útil del mismo y eliminar flexiones que pueden causar un daño estructural. Además, es necesario disminuir aún más (en lo posible a cero), la rotación relativa libre, existente entre el eje del motor y el eje de la rueda. De esta manera se disminuyen las no linealidades ante variaciones de velocidad.

Para continuar con el desarrollo del vehículo autónomo, es necesario que adquiera la capacidad de trasladarse de un punto a otro. Si bien posee control de sus ruedas, debe poder controlar cuánto tiene girar cada una para desplazarse de un punto "A" a otro "B". Una opción para resolverlo es el diseño de un sistema, que represente la dinámica del vehículo ante una entrada de control acorde y que sus salidas sean los setpoints de velocidad de cada rueda. De este modo, los sistemas de control de cada rueda, representarían un subsistema del vehículo.

Por último, una vez que el vehículo cuente con la capacidad para trasladarse siguiendo trayectorias especificas, este debe poder identificar obstáculos para luego poder evitarlos. Se lo debe equipar de sensores adecuados como lo pueden ser IMU, GPS, camara stereo, LIDAR y sensores de proximidad. Toda la información proveniente de dichos sensores debe ser procesada de manera adecuada en tiempo real y utilizada por algoritmos, que son los que luego interactúan con la capa de motricidad del vehículo.

Bibliografía

[1] Cmsis components, https://www.keil.com/pack/doc/cmsis/general/html/index.html.

[2] Modbus over serial line specification and implementation guide. page 44, dec 2006.

[3] Ieee recommended practice for maintenance, testing, and replacement of vented lead-acid batteries for stationary applications. page 71, February 2011.

[4] Development of a data acquisition and monitoring system based on modbus rtu communication protocol. page 6, 2020.

[5] Andy Wellings Alan Burns. *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real*. Time POSIX, 4 edition, 2005.

[6] Sidenko Alexey. Diseño y control del péndulo invertido. January 2011.

[7] ALLEGRO. Fully integrated, hall effect-based linear current sensor with 2.1 kvrms voltage isolation and a low-resistance current conductor. Datasheet.

[8] Alfredo Carrasco Araoz. Cómo obtener los parámetros de un motor de corriente continua e iman permanente. May 2009.

[9] Richard Barry. *Mastering the FreeRtos Real Time Kernel (Vol. 10)*, chapter Preface, 2, 3, 4, 6. 2016.

[10] Marcel Bergerman. Robot farmers: Autonomous orchard vehicles help tree fruit production. *IEEE Robotics & Automation Magazine*, pages 54–63, 7 2015.

[11] S.J Chapman. *Máquinas Eléctricas*. McGraw Hill, 5 edition, 2004.

[12] Gerardo Mario de Jong. *Análisis regional, estructuras agrarias y estrategias de desarrollo regional en la fruticultura del Alto Valle de la Cuenca del Río Negro*. PhD thesis, Universidad Nacional de la Plata, 2008.

[13] Ricardo Epianio. La problematica regional de Río Negro y Neuquén. 2020.

[14] Umans Stephen D. Fritzgerald A.E, Kingsley Charles Jr. *Máquinas Eléctricas*. McGraw Hill, 5 edition, 1996.

[15] Flego Fernando García Emiliano. Agricultura de precisión.

[16] Oscar Alexander Bellón Henpandez. Descripción teórica de un procedimiento para determinar los parámetros de un motor de corriente continua. March 2014.

[17] Texas Instruments. Zero-drift bi-directional current/power monitor with i2c interface. SBOS448A, August 2008. Datasheet.

[18] Peter Joslin. *Modelo Matemático motor DC conexión*. PhD thesis, Universidad politécnica salesiana sede Guayaquil, April 2015.

[19] José Miguel Fonseca-Gómez José Danilo Rairan-Antolines. December 2013.

[20] Bjorn Wittenmark Karl J. Astrom. *Computer controlled systems, theory and design.* Prentice Hall, 3 edition, 1984.

[21] Phillip A. Laplante. *Real-Time Systems Design and Analysis.* Wiley-IEEE Press, 10 edition, 2004.

[22] Carlos Gerardo Hernandez Capacho Manuel Guillermo Quijano Ortega. Obtención experimental de los parámetros del motor que se utilizará en el sistema de locomoción de una esfera rodante. May 2009.

[23] Ángel Rodriguez-Liñán Melissa E. Pérez Morales, Manuel Zamora. Caracterización paramétrica para un modelo de segundo orden del servomotor rc. March 2019.

[24] Francesco Mondada Mordechai Ben-Ari. *Elements of Robotics.* Springer open, 1 edition, 2018.

[25] Poder Legilativo Nacional. Ley nacional 27354 - declárece emergencia económica, financiera y social. 2017.

[26] Tore E. Undeland Ned Mohan and William P. Robbins. *Electrónica de Potencia Convertidores, Aplicaciones y Diseño.* McGraw Hill, 3 edition, 2009.

[27] Katsuhiko Ogata. *Ingenier∫ia de control moderna.* Pearson, 5 edition, 2009.

[28] RECOM INTERNATIONAL POWER. Dc-dc converter, April 2015. Datasheet.

[29] Press. Pr1290 12volt 9ah. 7A/160W Dual H-Bridge Motor Controller. Datasheet.

[30] National Semiconductor. 3.0a, 150khz, step-down switching regulator, May 2002. Datasheet.

[31] Gagne Silberschatz, Galvin. *Operating Systems Concepts.* John Wiley & Sons, 10 edition, 2018.

[32] STMicroelectronics. Performance line, arm-based 32-bit mcu with flash, usb, can, seven 16-bit timers, two adcs and nine communication interfaces. Rev 2, July 2007. Datasheet.

[33] STMicroelectronics. Description of stm32f4 hal and low-layer drivers. UM1725 - Rev 7, June 2021. Datasheet.

[34] Advanced Monolithic Systems. 800ma low dropout voltage regulator. Datasheet.

[35] Bos Tanenbaum. *Modern Operating Systems.* Prentice Hall, 4 edition, 2014.

[36] Handson Technology. 7a/160w dual h-bridge motor controller. Datasheet.

[37] Cristian Julián Solarte Rosas y Jhon Edinson Muñoz Ordoñez. Guía para la parametrización de un motor dc de imán permanente. March 2015.

[38] Felix Monasterio-Huelin y Álvaro Gutiérrez. Modelado de un motor dc. January 2021.

Apéndice A

Ensayos del sistema de control

En el presente anexo, se muestran los resultados obtenidos al relizar ensayos en carga y perturbaciones del sistema de control.

A.1. Ensayos del sistema de control

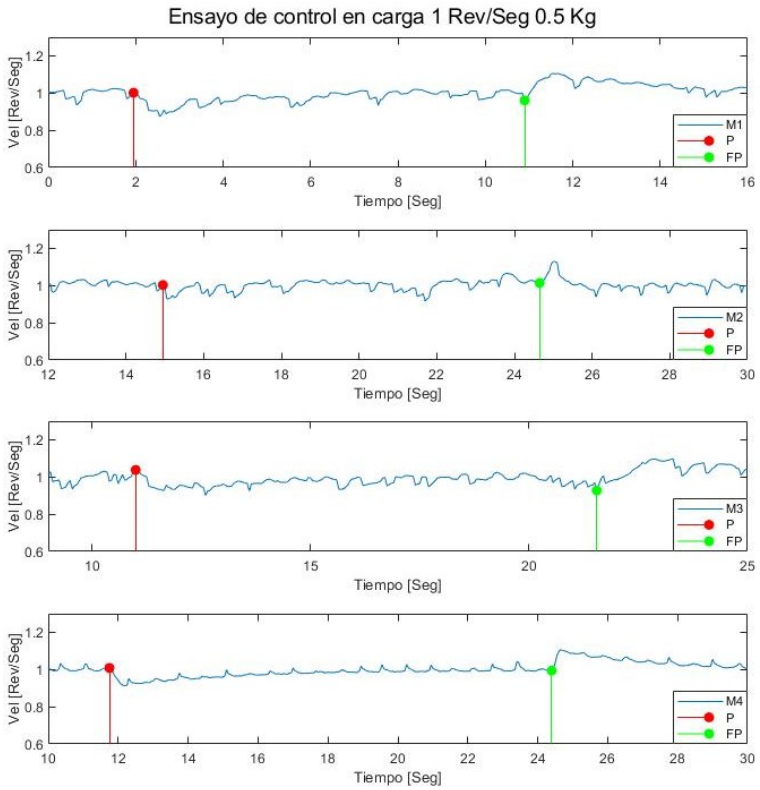


Figura A.1: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 1 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 0,5 Kg y en verde (FP) cuando se retira.

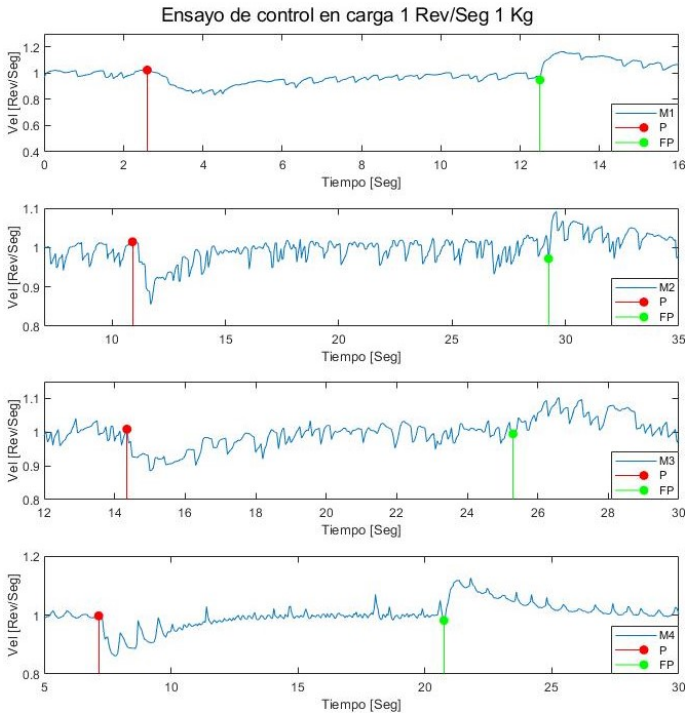


Figura A.2: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 1 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1 Kg y en verde (FP) cuando se retira.

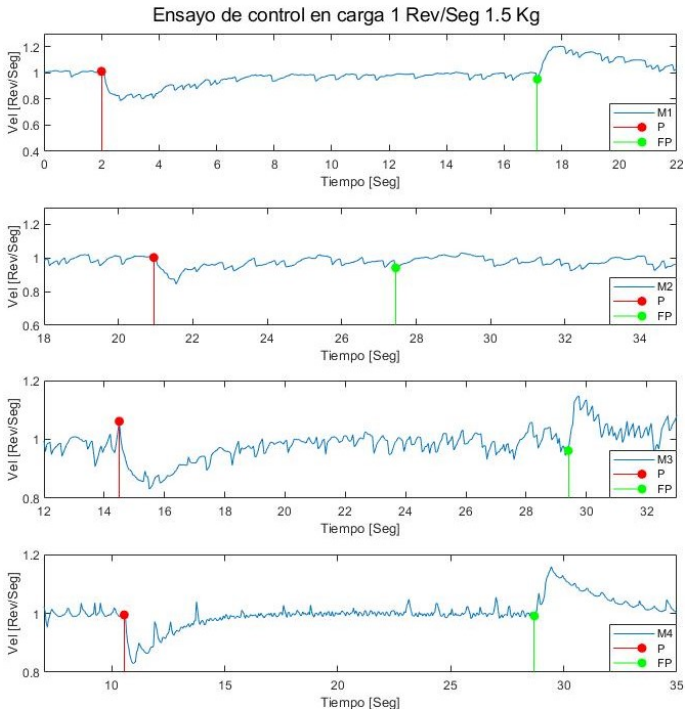


Figura A.3: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 1 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira.

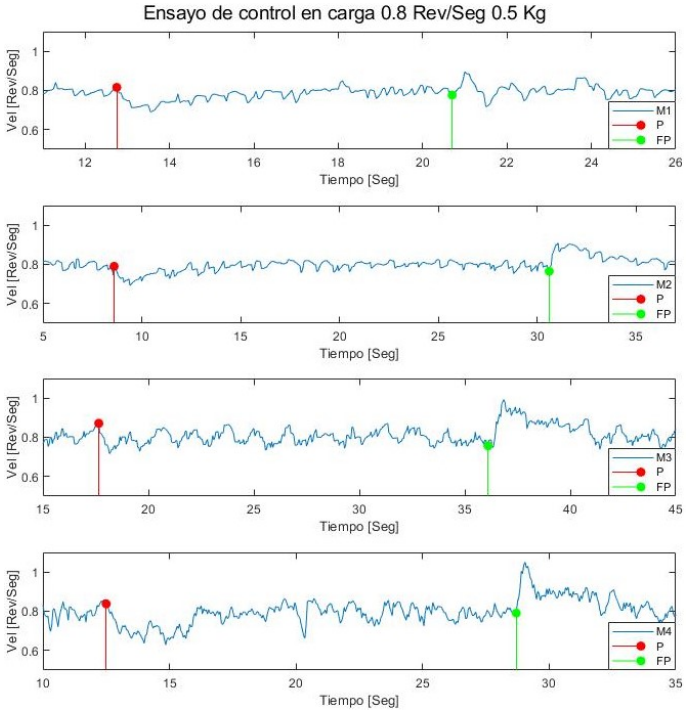


Figura A.4: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,8 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 0,5 Kg y en verde (FP) cuando se retira.

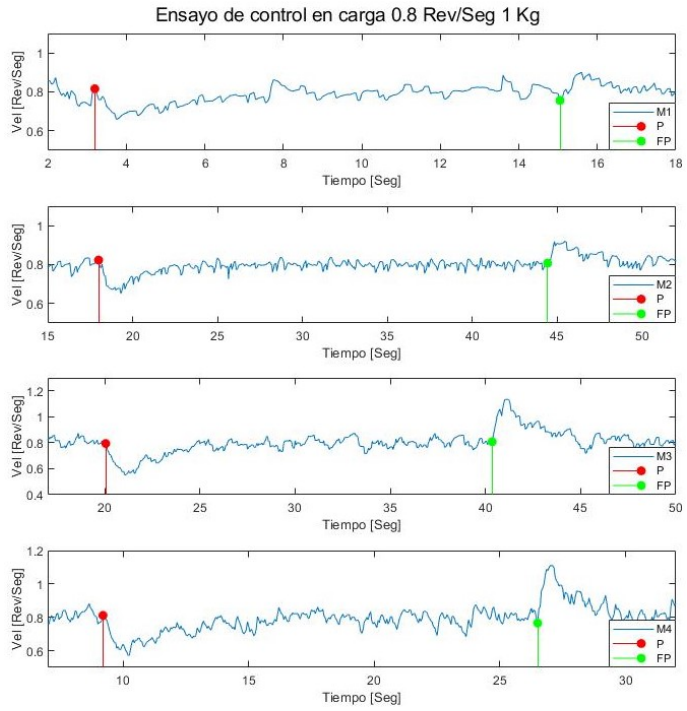


Figura A.5: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,8 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1 Kg y en verde (FP) cuando se retira.

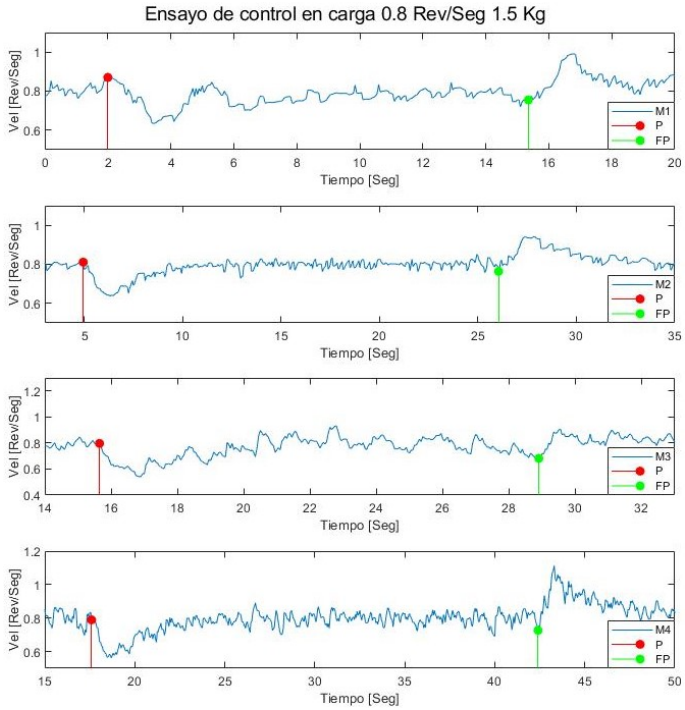


Figura A.6: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,8 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira.

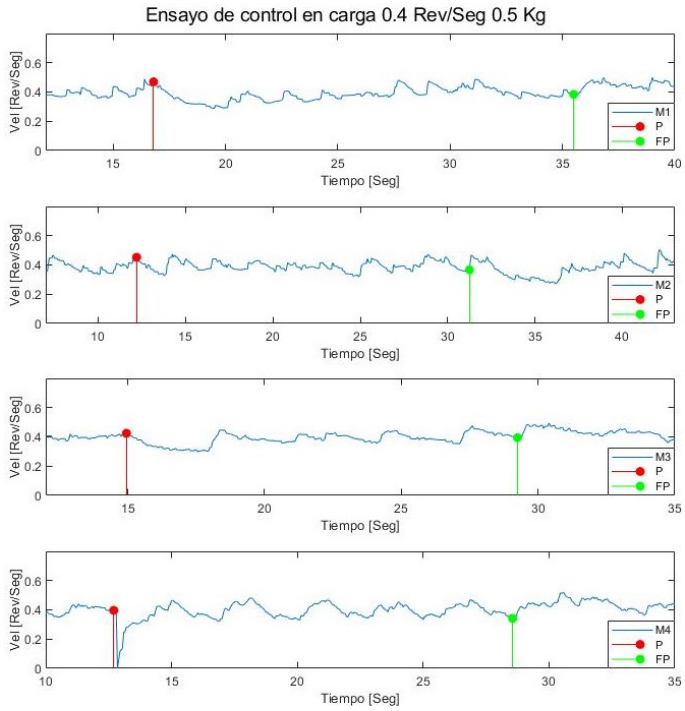


Figura A.7: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 0,5 Kg y en verde (FP) cuando se retira.

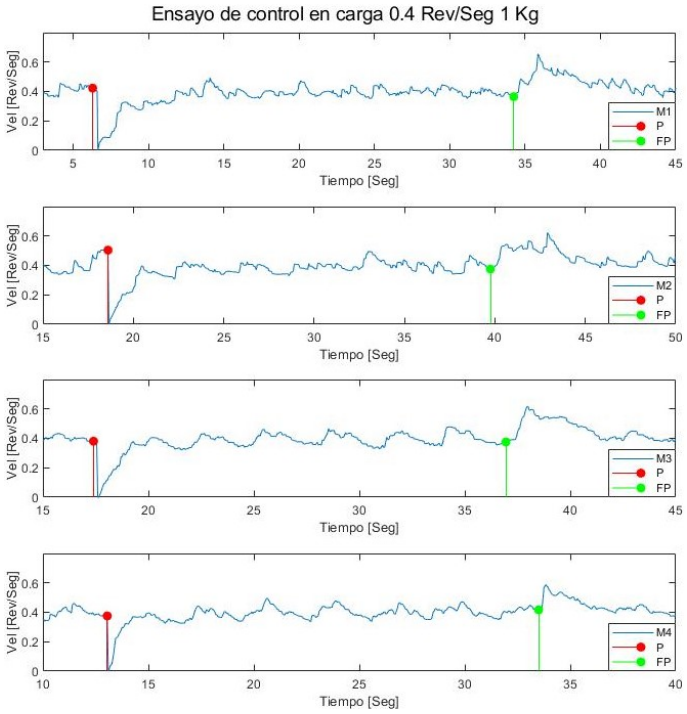


Figura A.8: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1 Kg y en verde (FP) cuando se retira.

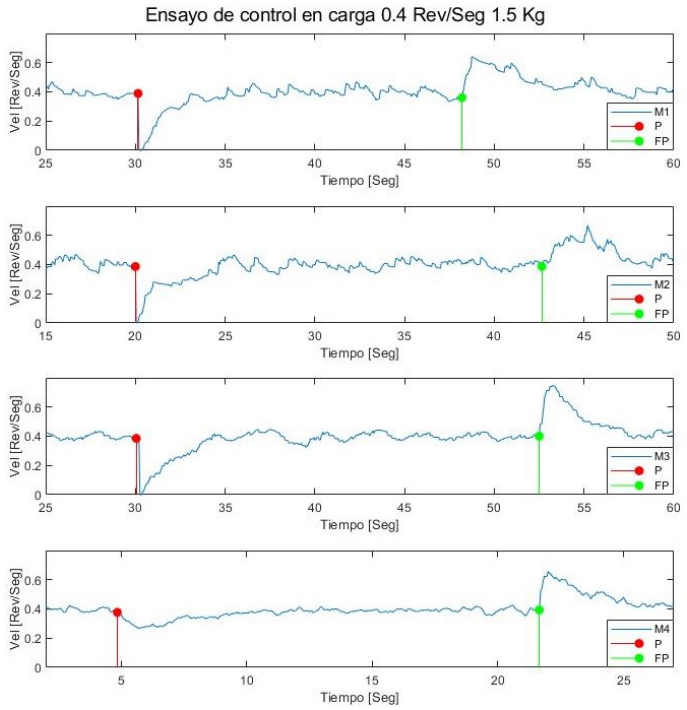


Figura A.9: En esta figura se muestra la respuesta del sistema de control en cada uno de los motores, cuando se les aplica una perturbación de tipo escalón generada por un torque de fricción como se explica en la sección 6.5.1. La señal de referencia es de 0,4 Rev/Seg, en rojo (P) se indica el momento en que se coloca la carga de 1,5 Kg y en verde (FP) cuando se retira.

Apéndice B

Diagramas de conexión

En el presente anexo, se detallan los diagramas topográfico y de de conexionado, del vehículo.

B.1. Diagrama topográfico

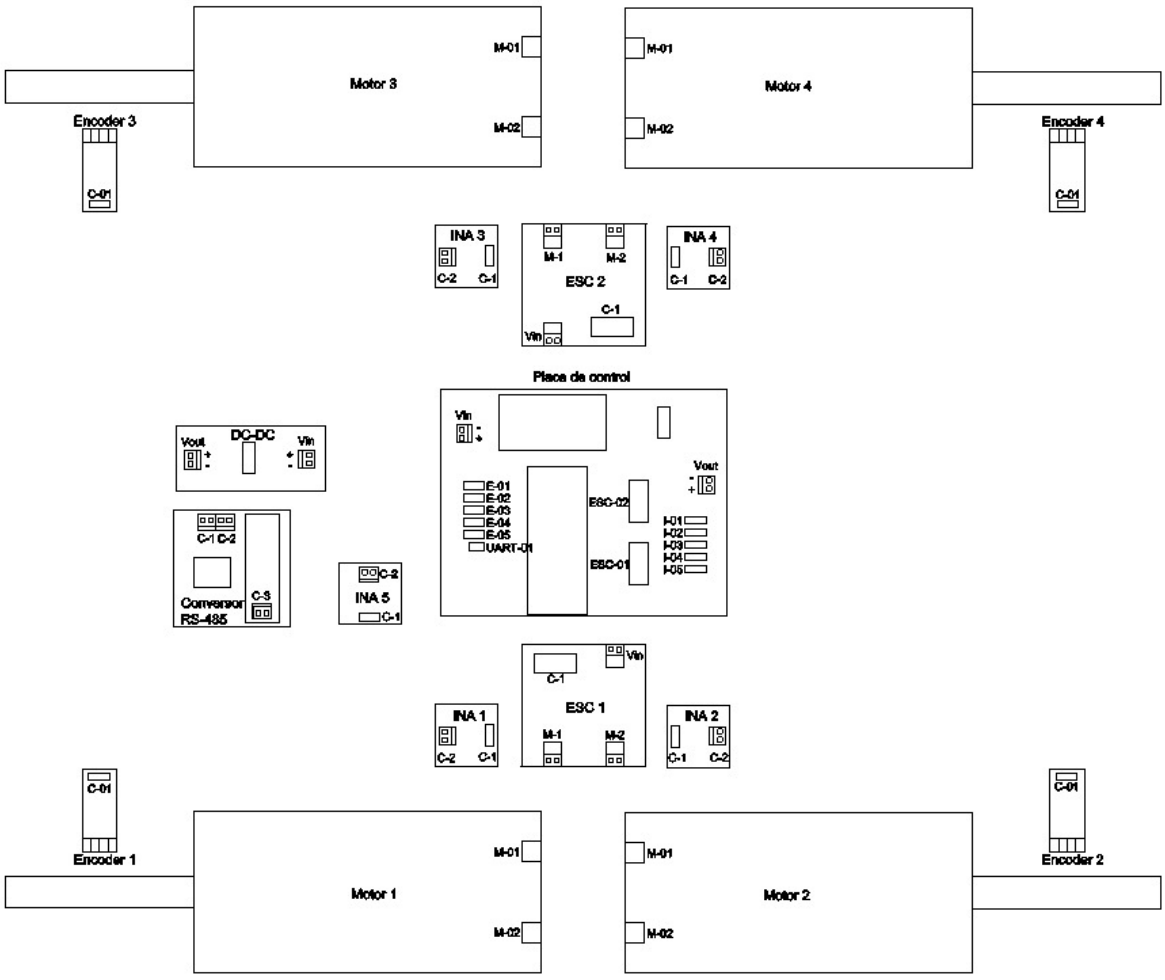


Figura B.1: En el diagrama topográfico se muestra la disposición de los componentes dentro del chasis de vehículo.

B.2. Diagramas de conexionado

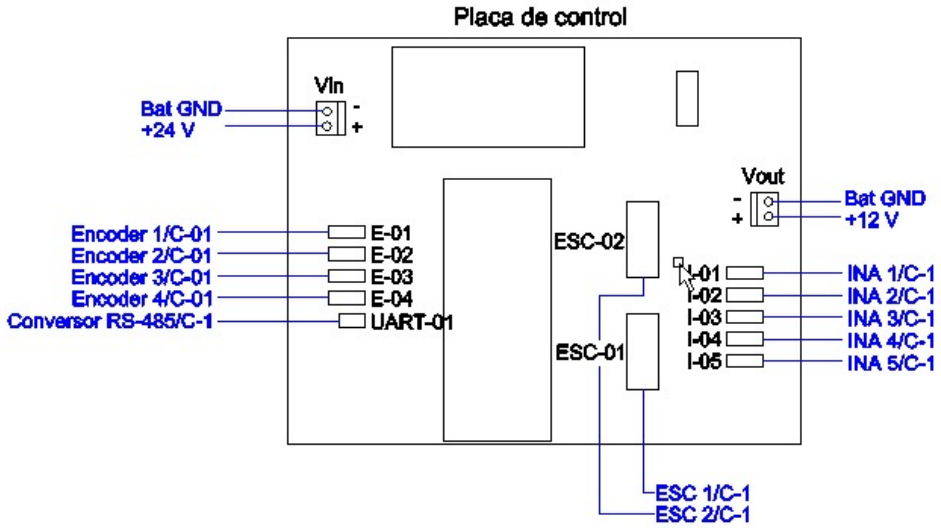


Figura B.2: Diagrama de conexionado de placa de control.

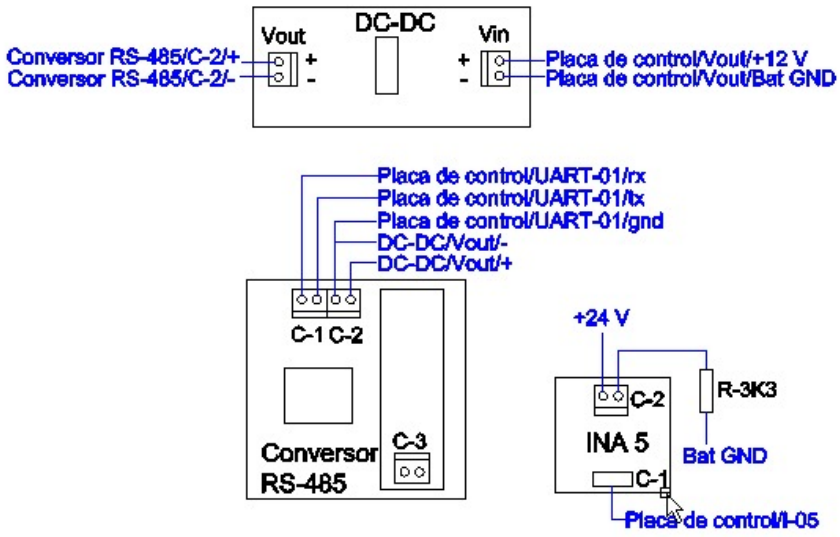


Figura B.3: Diagrama de conexionado de conversor RS-485, conversor dc-dc aislado y sensor de corriente para medir tensión en baterías.

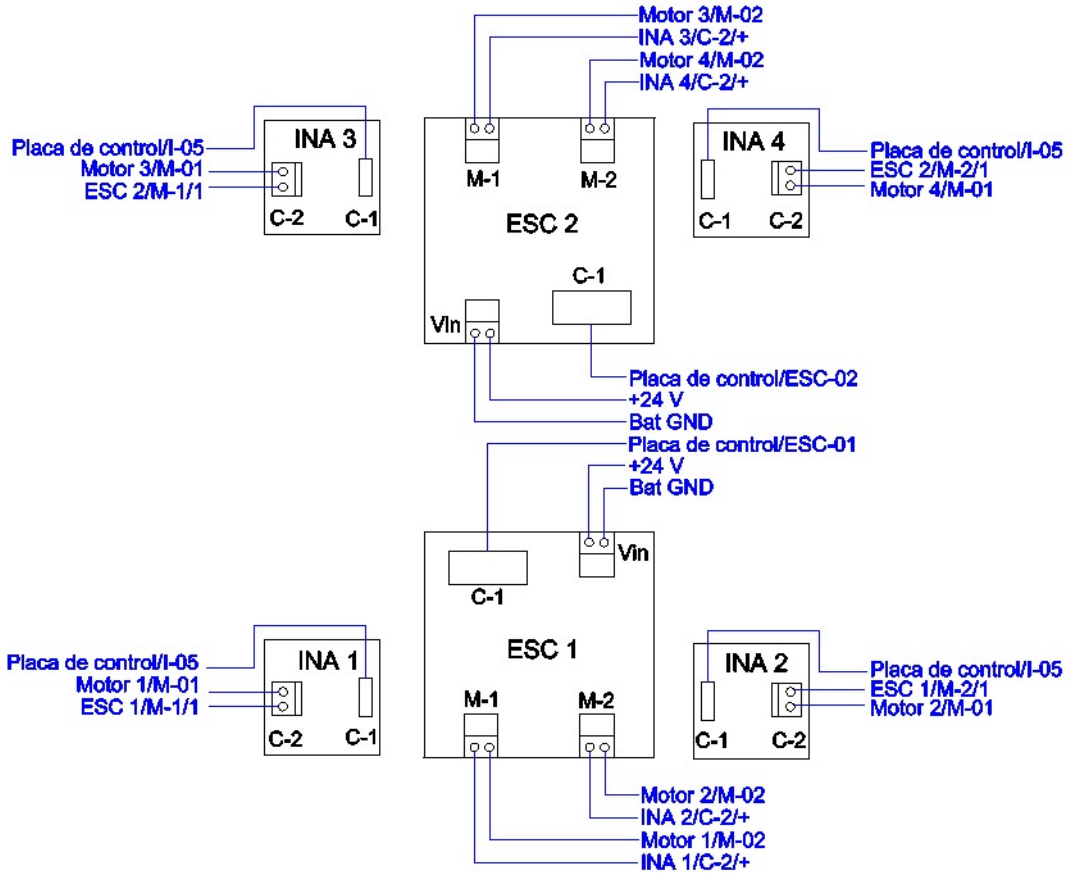


Figura B.4: Diagrama de conexionado de variadores de velocidad y sensores de corriente de los motores.

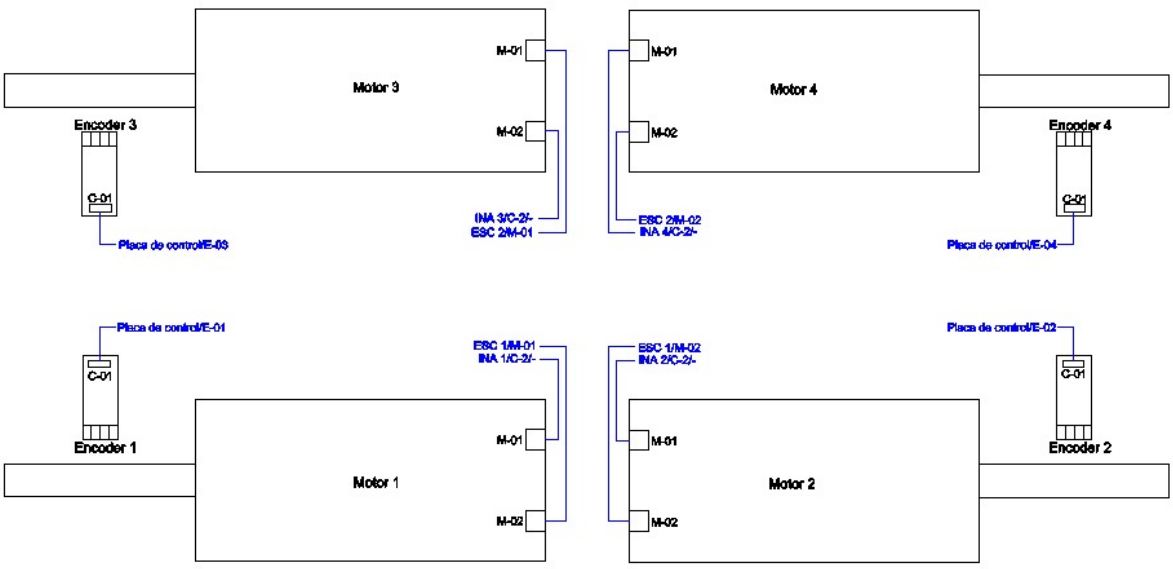


Figura B.5: Diagrama de conexionado de encoders incrementales y motores.