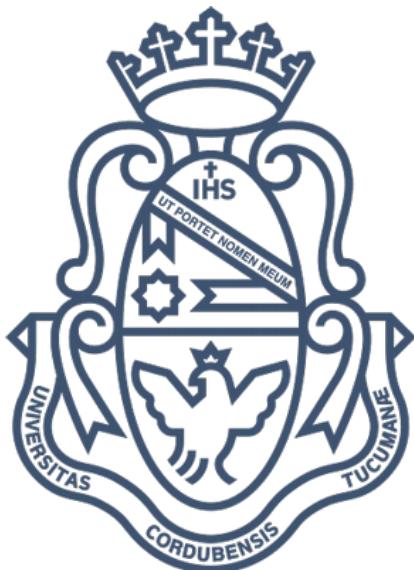


Proyecto Integrador



Autores

García Cabral Ana Belén - Sagripanti Sergio Javier

Tema

Diseño y desarrollo de robot omnidireccional

Director:

PhD. Orlando Micolini

Codirector:

Ing. Lichtensztein Leandro

CONTACTOS

García Cabral Ana Belén

Tel : 03543 15572123
Email : ab.garciacabral@gmail.com

Sagripanti Sergio Javier

Tel : 0351 153620621
Email : sergiosagripanti@gmail.com

Micolini Orlando

Tel : 0351 153731094
Email : omicolini@compuar.com

Lichtensztein Leandro

Tel : 0351 156828262
Email : leandrosnm@gmail.com

ÍNDICE GENERAL

1. Introducción	14
1.1. Motivación	14
1.2. Objetivo	14
1.2.1. Objetivos principales	15
1.2.2. Objetivos secundarios	15
1.3. Método de Desarrollo	15
1.4. Requerimientos	16
1.4.1. Requerimientos funcionales	16
1.4.2. Requerimientos no funcionales	16
1.5. Análisis de requerimientos funcionales	17
1.6. Análisis de riesgos	17
1.7. Arquitectura preliminar de alto nivel del sistema	18
1.8. Plan de trabajo	19
2. Marco Teórico	20
2.1. Modelo de desarrollo	20
2.1.1. Modelo incremental	20
2.1.2. Ingeniería de software orientada a la reutilización	21
2.2. Elección del modelo	21
2.3. Robots en la educación	22
2.4. Robots	22
2.4.1. Clasificación de robots	22
2.4.2. Sistemas mecánicos de locomoción	23
2.4.3. Restricciones holonómicas y no-holonómicas	26
2.4.4. Componentes	26

2.4.5. Sistema de control	30
2.4.6. Sistema de comunicación	31
3. Iteración 1	33
3.1. Introducción	33
3.2. Requerimientos	33
3.3. Desarrollo	34
3.3.1. Selección de la placa de desarrollo	34
3.3.2. Arquitectura de hardware	35
3.3.3. Selección del sistema de locomoción	36
3.3.4. Selección del sistema de alimentación	36
3.4. Resultados	39
3.5. Conclusiones	39
4. Iteración 2	40
4.1. Introducción	40
4.2. Requerimientos	40
4.3. Desarrollo	41
4.3.1. Selección de ruedas	41
4.3.2. Selección de motores	41
4.3.3. Diseño de nuevas bridas y adaptación de bujes	42
4.3.4. Armado del chasis	42
4.3.5. Comunicación Bluetooth con Arduino	43
4.3.6. Desarrollo de biblioteca para control de motores	43
4.3.7. Medición de velocidad de motores	44
4.3.8. Análisis de métodos de medición de RPM	47
4.3.9. Selección del método de medición de RPM	48
4.3.10. Sistema de control de velocidad	48
4.4. Pruebas	50
4.4.1. Comunicación Bluetooth	50

4.4.2. Biblioteca	51
4.4.3. Funcionamiento del robot	52
4.5. Resultados	53
4.6. Conclusiones	53
5. Iteración 3	54
5.1. Introducción	54
5.2. Requerimientos	54
5.3. Desarrollo	54
5.3.1. Análisis de motores	54
5.3.2. Selección de motor	56
5.3.3. Diseño de nuevas bridas	57
5.3.4. Rediseño de bujes	57
5.4. Pruebas	58
5.4.1. Medición de velocidad[RPM]	58
5.4.2. Medición del torque del motor	59
5.4.3. Prueba de motores incorporados en el robot	61
5.4.4. Desvío de trayectoria	62
5.5. Resultados	63
5.6. Conclusiones	63
6. Iteración 4	64
6.1. Introducción	64
6.2. Requerimientos	64
6.3. Desarrollo	64
6.3.1. Incorporación de Raspberry Pi B+	64
6.4. Pruebas	71
6.4.1. Comunicación Bluetooth en Raspberry Pi	71
6.4.2. Detección de dispositivos conectados al bus I2C	72
6.4.3. Comunicación I2C entre Raspberry Pi y Arduino	73

6.5. Resultados	74
6.6. Conclusiones	74
7. Iteración 5	75
7.1. Introducción	75
7.2. Requerimientos	75
7.3. Desarrollo	75
7.3.1. Rediseño de chasis para sensores	75
7.3.2. Arquitectura de software	75
7.3.3. Comunicación	76
7.3.4. Actuadores	78
7.3.5. Sensores	80
7.4. Pruebas	85
7.4.1. Actuador	85
7.4.2. Sensor de línea	86
7.4.3. Sensor de distancia	87
7.5. Resultados	87
7.6. Conclusión	88
8. Sistema Final	89
8.1. Introducción	89
8.2. Integración de componentes	89
8.3. Aplicación Android	92
8.4. Comunicación Smartphone-HermesII	93
8.5. Proceso de decodificación de los mensajes.	93
8.6. Casos de uso	94
8.6.1. Robot seguidor de línea	95
8.6.2. Movimiento mediante comandos	98
8.6.3. Movimiento con detección de obstáculos	100
8.7. Matriz de trazabilidad	102

I Anexos	108
A. Desarrollo de biblioteca para el control de los motores	109
A.1. DriverMotor.h	109
A.2. DriverMotor.cpp	110
B. Códigos de prueba	112
B.1. Comunicación Bluetooth con Arduino	112
B.2. Biblioteca	112
B.3. Funcionamiento del robot	113
B.4. Comunicación Bluetooth con Raspberry Pi	115
C. Manual de armado	117

ÍNDICE DE FIGURAS

1.1.	Desarrollo incremental	15
1.2.	Arquitectura preliminar del sistema	18
2.1.	Modelo de desarrollo incremental [3]	21
2.2.	Modelo orientado a la reutilización [3]	22
2.3.	Robot diferencial.	23
2.4.	Robot síncrono [2]	24
2.5.	Robot triciclo	24
2.6.	Robot Ackerman	25
2.7.	Robot omnidireccional.	25
2.8.	Ruedas omnidireccionales.	26
2.9.	Ruedas mecanum.	27
2.10.	Funcionamiento sensor de distancia por sonar	27
2.11.	Sensor óptico	28
2.12.	Esquema cinemático de robot omnidireccional de 4 ruedas.	31
2.13.	Comunicación I2C [8]	31
3.1.	Arquitectura de hardware del sistema	36
3.2.	Batería seleccionada	37
3.3.	Reductor de tensión	38
4.1.	Motor DC	41
4.2.	Piezas encastrables.	42
4.3.	Vista superior del chasis	42
4.4.	Conexión de módulo Bluetooth a Arduino	43
4.5.	Patrón impreso en la rueda	44
4.6.	Sensor infrarrojo [13]	44
4.7.	Primer método de medición de RPM	45

4.8.	Segundo método de medición de RPM	45
4.9.	Tercer método de medición de RPM	46
4.10.	Controlador proporcional	49
4.11.	Controlador proporcional, integral, derivativo	50
5.1.	Prototipo de prueba	55
5.2.	Prototipo de prueba	55
5.3.	Fuerza de rozamiento estático	56
5.4.	Nuevo motor	57
5.5.	Nuevo soporte de motor	57
5.6.	Buje de POM	58
5.7.	Velocidad del motor	59
5.8.	Sistema de medición de torque.	59
5.9.	Torque del motor	61
6.1.	Conexión de módulo Bluetooth a Raspberry Pi B+	68
6.2.	Conexión entre microcontroladores por protocolo I2C	70
6.3.	Prueba de conexión bajo protocolo I2C	72
7.1.	Base del robot	76
7.2.	Arquitectura de software	76
7.3.	Interfaz de comunicación	77
7.4.	Interfaz de actuadores	78
7.5.	Estructura del mensaje I2C	79
7.6.	Conexión de motores	80
7.7.	Interfaz de sensores	81
7.8.	Sensor de proximidad	81
7.9.	Diseño de circuito	82
7.10.	Diseño de pcb.	83
7.11.	Detector de línea	83
7.12.	Sensor de distancia HC-SR04	83
7.13.	Conexión sensor HC-SR04 a Raspberry Pi B+	84

7.14. Señales sensor HC-SR04 [19]	84
8.1. Sistema integrado	90
8.2. Diagrama de clases del sistema final	91
8.3. Aplicación android	92
8.4. Diagrama de secuencia Seguidor de Línea	97
8.5. Diagrama de secuencia manejado por comandos	99
8.6. Diagrama de secuencia de evasor de obstáculos	102

ÍNDICE DE TABLAS

1.1.	Análisis de riesgos de requerimientos funcionales.	17
1.2.	Análisis de riesgos de requerimientos no funcionales.	18
1.3.	Plan de trabajo	19
2.1.	Clases de dispositivos Bluetooth.	32
3.1.	Características de las placas de desarrollo	34
3.2.	Tipos y características de baterías	37
4.1.	Motores disponibles	41
4.2.	Características del motor seleccionado (figura 4.1)	41
4.3.	Caso de prueba 1	50
4.4.	Caso de prueba 2	51
4.5.	Caso de prueba 3	52
5.1.	Características del motor seleccionado	57
5.2.	Torque del motor	60
5.3.	Caso de prueba 4	62
6.1.	Pines de conexión entre Raspberry Pi B+ y módulo Bluetooth HC-05	68
6.2.	Pines de conexión entre Raspberry Pi B+ y Arduino ProMini	70
6.3.	Caso de prueba 5	71
6.4.	Prueba de comunicación I2C	73
7.1.	Características sensor TCRT5000	82
7.2.	Características del sensor de distancia HC-SR04	83
7.3.	Caso de prueba 7	85
7.4.	Caso de prueba 8	86
7.5.	Caso de prueba 9	87
8.1.	Caso de uso 1	95
8.2.	Caso de uso 2	96

8.3. Caso de uso 3	98
8.4. Caso de uso 4	100
8.5. Caso de uso 5	101
8.6. Caso de uso 6	101
8.7. Matriz de trazabilidad	102

RESUMEN

En el presente proyecto se rediseos sistemas mecco, de control y electro del robot omnidireccional Hermes, desarrollado por el Ingeniero Lichtenstein Leandro para la enseña de programaci la educacicundaria y universitaria.

Esta nueva versinominada Hermes II, cuenta con un sistema mecco de locomoci simple que su predecesora, debido a la disminuci sus partes mes para modificar su trayectoria. Esto se logra gracias a la incorporaci ruedas omnidireccionales, elemento disruptivo del proyecto.

El sistema de control del robot, que es momplejo, se encuentra embebido en una placa de desarrollo que cuenta con un sistema operativo libre, el cual permite la programacioncurrente haciendo uso de hilos en diferentes lenguajes de programaci

El software se desarrolla orientado a componentes con el fin de hacer un proyecto fí de mantener y ampliar mediante mos reutilizables.

En la ltima etapa del proyecto se realiza una integraci los mos desarrollados y se plantean diferentes escenarios de uso del robot a modo de ejemplo.

CAPÍTULO 1

INTRODUCCIÓN

1.1. Motivación

Académico:

- Realizar un proyecto que nos permita integrar los conocimientos adquiridos a lo largo de la carrera.
- Adquirir nuevos conocimientos mediante la resolución de un problema real.

Educativo:

- Diseñar un sistema robótico que lo pueda armar una persona sin conocimientos específicos del área.
- Diseñar un sistema para ser aplicado en un proceso de enseñanza de programación.
- Lograr un diseño modular con el fin de permitir la modificación del sistema conforme a requerimientos específicos.
- Desarrollar un sistema compuesto de módulos reutilizables.

Económico:

- Desarrollar un robot educativo competitivo en el mercado desde el punto de vista de la relación prestaciones/costos de desarrollo.
- Emplear componentes disponibles en el mercado.

Extensión:

- Desarrollar un sistema que pueda ser empleado para la educación en el nivel secundario.

1.2. Objetivo

Rediseñar el robot Hermes [1] con el fin de que pueda ser utilizado en la enseñanza secundaria y universitaria.

1.2.1. Objetivos principales

- Rediseñar el sistema de control del robot Hermes [1].
- Incorporar una arquitectura de hardware que permita la ejecución de un sistema operativo.

1.2.2. Objetivos secundarios

- Diseñar un sistema de ensamblaje simple.
- Implementar protocolos de comunicación estándar.
- Diseñar un sistema de percepción del entorno a través de sensores.
- Diseñar una arquitectura de software y hardware orientada a componentes.
- Construir un prototipo basado en la propuesta implementada en el presente trabajo.

1.3. Método de Desarrollo

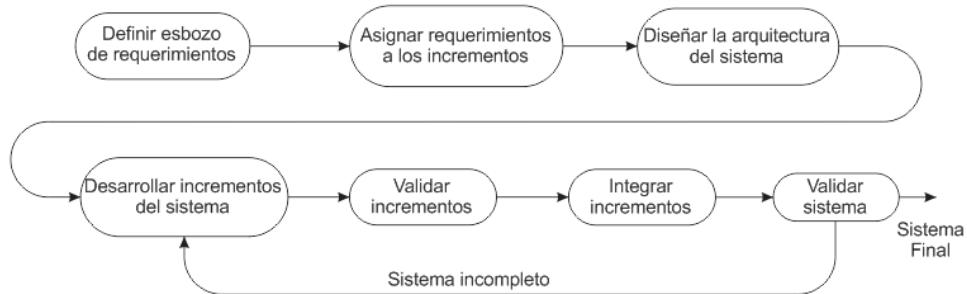


Figura 1.1: Desarrollo incremental

El método de desarrollo utilizado es el desarrollo iterativo con entrega incremental. Este modelo se ilustra en la figura 1.1. En esta metodología de desarrollo el trabajo se divide en iteraciones en las cuales el producto va evolucionando.

Un aspecto fundamental para guiar el desarrollo incremental es la priorización de los requerimientos/objetivos en función del valor que aportan al cliente. De esta manera se van añadiendo nuevos requerimientos o mejorando los que ya se completaron. Al finalizar cada iteración se obtiene un prototipo funcional.

1.4. Requerimientos

Los requerimientos funcionales son aquellas declaraciones de los servicios que debe proporcionar el sistema, cómo debe reaccionar ante determinados eventos y cómo debe comportarse en situaciones particulares.

Los requerimientos no funcionales son aquellos que no se refieren directamente a la funcionalidad del sistema sino a aquellas propiedades que emergen de él como la fiabilidad, la capacidad de almacenamiento, disponibilidad, etc. También definen la restricciones del sistema como los dispositivos de entrada/salida y las interfaces de comunicación del sistema.

1.4.1. Requerimientos funcionales

1. El robot debe poder moverse de forma holonómica en todas las direcciones sobre el plano.
2. El usuario debe poder comunicarse con el robot para configurarlo y controlarlo.
3. El robot debe poder reconocer un ambiente mediante el uso de sensores.
4. El robot debe soportar la programación de hilos para ser utilizado en la enseñanza de programación concurrente.

1.4.2. Requerimientos no funcionales

1.4.2.1. Requerimientos generales

- Fácil armado siguiendo un instructivo (seis horas).
- El máximo desvío de su trayectoria no debe superar los 2cm en 1m recorrido.
- El costo del robot debe ser inferior a \$3000 para ser competitivo en el mercado.
- Permitir su manejo desde múltiples plataformas (PC, Smartphone, Tablet, joystik)
- Diseño estructural robusto.
- Que sea capaz de resolver problemas de concurrencia

1.4.2.2. Requerimientos de software

- Programable en Java, C, C++ y Python.
- Implementado con bibliotecas abiertas.
- Uso de Sistema Operativo.

1.4.2.3. Requerimientos de hardware

- Hardware libre.
- Componentes con abundante documentación.
- Microcontrolador de simple programación.
- Uso de múltiples sensores.
- Múltiples entradas para sensores.
- Múltiples salidas para actuadores (4 salidas PWM).

1.5. Análisis de requerimientos funcionales

En la tabla 1.1 se realiza un análisis de riesgos de los requerimientos funcionales y se propone una prueba para cada uno de ellos.

ID	Nombre	Riesgo	Prueba
1	El robot debe poder moverse de forma holónómica en todas las direcciones	Alto	Se programa una rutina para que el robot se mueva en distintas direcciones (avanzar, retroceder, girar izquierda, girar derecha, etc)
2	Establecer una comunicación entre el usuario y el robot	Alto	Se envía un comando al robot y éste responde al usuario.
3	El robot debe poder reconocer un ambiente mediante el uso de sensores	Media	Se programa una rutina de lectura de los sensores y el sistema envía estas lecturas al usuario
4	Debe admitir programación de hilos	Media	Se programa una rutina en la que se ejecutan actividades de forma paralela a nivel de hilos

Tabla 1.1: Análisis de riesgos de requerimientos funcionales.

1.6. Análisis de riesgos

La gestión de riesgos permite anticiparse a eventualidades que puedan afectar al desarrollo y calidad del producto tomando decisiones para mitigarlas. En la tabla 1.2 se analizan los riesgos más relevantes.

Riesgo	Probabilidad	Impacto	Acciones
Falla en los motores	Media	Alto	Reemplazar los motores elegidos.
Mala elección de las ruedas	Media	Alto	Cambiar las ruedas elegidas
Variación de los costos	Baja	Bajo	Sustituir por componentes equivalentes y de menor costo
Mala estimación de tiempos	Media	Bajo	Reducir el alcance del proyecto
Imprecisión del sistema de medición de velocidad	Alta	Medio	Emplear un nuevo método de medición
Mala elección de batería	Baja	Bajo	Usar otro tipo de batería y conversores de nivel de tensión

Tabla 1.2: Análisis de riesgos de requerimientos no funcionales.

1.7. Arquitectura preliminar de alto nivel del sistema

Luego de estudiar los requerimientos y los objetivos del proyecto se diseña la arquitectura preliminar del sistema que se observa en la figura 1.2.

La arquitectura propuesta consta de diferentes subsistemas, los cuales permiten realizar un diseño modular. De esta forma se facilita la incorporación y eliminación de subsistemas sin la necesidad de modificar el sistema completo.

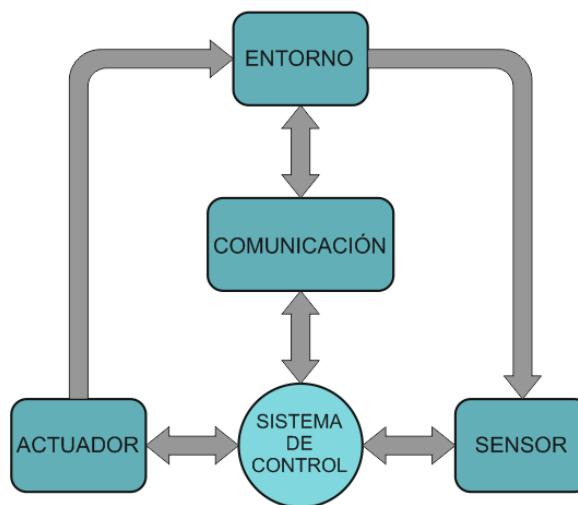


Figura 1.2: Arquitectura preliminar del sistema

1.8. Plan de trabajo

En cada iteración se busca satisfacer aquellos requerimientos que son más críticos para el sistema, agregando aquellos que incluyan una nueva funcionalidad al prototipo. Se tiene en cuenta que al momento de realizar las pruebas correspondientes a dicha iteración se pueden detectar fallas. En ese caso se agrega una nueva iteración en la que se busca la corrección de los problemas detectados.

A continuación se presenta la tabla 1.3 en la cual se especifican las tareas a desarrollar en cada iteración.

Iteración	Desarrollo
1	<ul style="list-style-type: none">■ Estudio y diseño de arquitectura de hardware■ Estudio y selección de placa de desarrollo■ Seleccionar el sistema de locomoción.■ Selección del sistema de alimentación
2	<ul style="list-style-type: none">■ Cambio de ruedas.■ Desarrollo de biblioteca para control de motores.■ Comunicación Bluetooth.■ Sistema de control de velocidad.
3	<ul style="list-style-type: none">■ Incorporación de arquitectura que soporte sistema operativo.■ Comunicación I2C.
4	<ul style="list-style-type: none">■ Incorporación de sensores.
5	<ul style="list-style-type: none">■ Diseño y desarrollo de una placa con todos los componentes.■ Desarrollar un manual de armado.

Tabla 1.3: Plan de trabajo

CAPÍTULO 2

MARCO TEÓRICO

2.1. Modelo de desarrollo

Es una estructura aplicada al desarrollo de un producto de software. Cada modelo describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso. Los modelos que se mencionan son los siguientes:

“Modelo en cascada (waterfall): Éste toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y, luego, los representa como fases separadas del proceso, tal como especificación de requerimientos, diseño de software, implementación, pruebas, etcétera.

Desarrollo incremental: Este enfoque vincula las actividades de especificación, desarrollo y validación. El sistema de desarrollo como una serie de versiones (incrementos), y cada versión añade funcionalidad a la versión anterior.

Ingeniería de software orientada a la reutilización: Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en la integración de estos componentes en un sistema, en vez de desarrollo desde cero” [3]

2.1.1. Modelo incremental

Se basa en la idea de diseñar una implementación inicial, exponer ésta al comentario del usuario, y luego desarrollarla en sus diversas versiones hasta producir un sistema adecuado.

Las actividades de especificación, desarrollo y validación están entrelazadas en vez de separadas como se observa en la figura 2.1, y poseen una retroalimentación rápida a través de las actividades.

El desarrollo incremental refleja la forma en que se resuelven los problemas. No se trabaja por adelantado una solución completa, se avanza en una serie de pasos hacia una solución y se retrocede cuando se detecta que se cometieron errores.

Al desarrollar el software de manera incremental, resulta más barato y fácil realizar cambios en él conforme éste se diseña. Cada incremento o versión del sistema incorpora alguna de las funciones que necesita el cliente.

Por lo general, los primeros incrementos del sistema incluyen la función más importante o la más urgente. Esto significa que el cliente puede evaluar el desarrollo del sistema en una etapa relativamente temprana, para constatar si se entrega lo que se requiere. En caso contrario, sólo el incremento actual debe cambiarse y, posiblemente, definir una nueva función para incrementos posteriores.

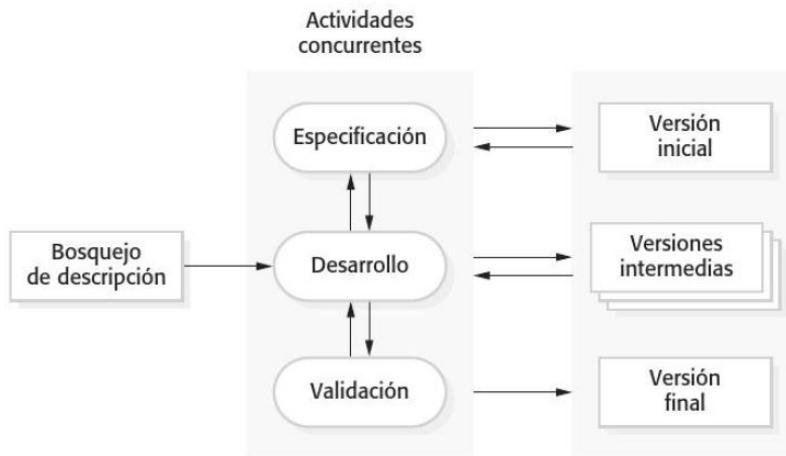


Figura 2.1: Modelo de desarrollo incremental [3]

2.1.2. Ingeniería de software orientada a la reutilización

En la mayoría de los proyectos se reutilizan partes que son iguales o similares de otros proyectos. Esta reutilización informal es independiente del proceso de desarrollo que emplee.

Los enfoques orientados en la reutilización se apoyan en una gran base de componentes de software reutilizable y en la integración de marcos para la composición de dichos componentes.

La principal ventaja de la reutilización de software es que reduce la cantidad de software a desarrollar en un proyecto haciendo entregas más rápidas y al mismo tiempo se reducen costos y riesgos.

Las etapas de desarrollo se observan en la figura 2.2.

2.2. Elección del modelo

Los modelos de desarrollo no son mutuamente excluyentes y con frecuencia se usan en conjunto, sobre todo para el desarrollo de grandes sistemas. Por este motivo se decide trabajar con el modelo de desarrollo incremental y el modelo de ingeniería de software orientada a la reutilización. Lo que se busca es combinar algunas de las mejores características de ambos modelos de desarrollo.



Figura 2.2: Modelo orientado a la reutilización [3]

El modelo orientado en la reutilización permite desarrollar módulos con funcionalidades bien definidas que puedan ser usados en otros proyectos.

Por otro lado al trabajar con el modelo incremental se puede ir cumpliendo con los requerimientos en orden de riesgo y prioridad. A su vez cada incremento finaliza con la incorporación de una o más funcionalidades.

2.3. Robots en la educación

A principios de los 80's, muchos países comenzaron a usar la robótica como una forma alternativa de fomentar la ciencia y la tecnología. Actualmente en nuestro país, muchas provincias han incluido esta metodología de enseñanza.

Está demostrado que este tipo de actividades promueve el trabajo en equipo, la creatividad, motivación e incluso ayuda a mejorar las habilidades para la resolución de problemas.

2.4. Robots

Un robot es un dispositivo capaz de interactuar con su entorno haciendo uso de sensores para obtener datos, procesar la información mediante una unidad de procesamiento y ejecutar acciones mediante actuadores.

A. Ollero Baturone define los robots como “máquinas en las que se integran componentes mecánicos, eléctricos, electrónicos y de comunicaciones, y dotadas de un sistema informático para su control en tiempo real, percepción del entorno y programación” [2]

2.4.1. Clasificación de robots

Se pueden clasificar de la siguiente manera:

- Terrestres: se desplazan sobre una superficie y lo pueden realizar mediante el uso de ruedas, orugas o patas.

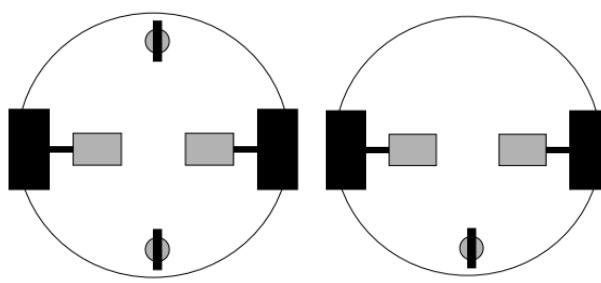
-
- Acuáticos: operan en la superficie del agua o por debajo de ella.
 - Aéreos: son conocidos como vehículos aéreos no tripulados y controlados por radiocontrol o de forma autónoma mediante un programa previamente cargado.

2.4.2. Sistemas mecánicos de locomoción

El sistema de locomoción es el responsable de la traslación del robot. Las configuraciones más comunes son las siguientes: Diferencial, síncrono, triciclo, Ackerman y omnidireccional. [2]

2.4.2.1. Diferencial

Es uno de los esquemas más sencillos. El mismo consta de dos ruedas en un eje común y se controlan de forma independiente. A este tipo de configuración se le agrega una o dos ruedas castores encargadas del balance del robot, como se observa en la imagen 2.3. Los robots que usan este sistema de locomoción pueden desplazarse en forma recta, en arco o rotar sobre su propio eje. La dirección en la que se desplace el robot está relacionada con la velocidad y sentido de giro de cada motor. [2]



(a) Dos ruedas de apoyo (b) Una rueda de apoyo

Figura 2.3: Robot diferencial.

2.4.2.2. Síncrono

En este tipo de configuración todas las ruedas, usualmente tres, se mueven en forma síncrona para dar vuelta y avanzar. Las mismas están unidas de forma tal que siempre están orientadas en la misma dirección como se observa en la figura 2.4. Para girar se mueven las ruedas sobre el eje vertical, por lo que la dirección del chasis se mantiene. La distribución de las ruedas elimina el problema de inestabilidad que tiene la configuración diferencial y de pérdida de contacto de las ruedas. La mayor desventaja de este sistema es la complejidad mecánica [2]

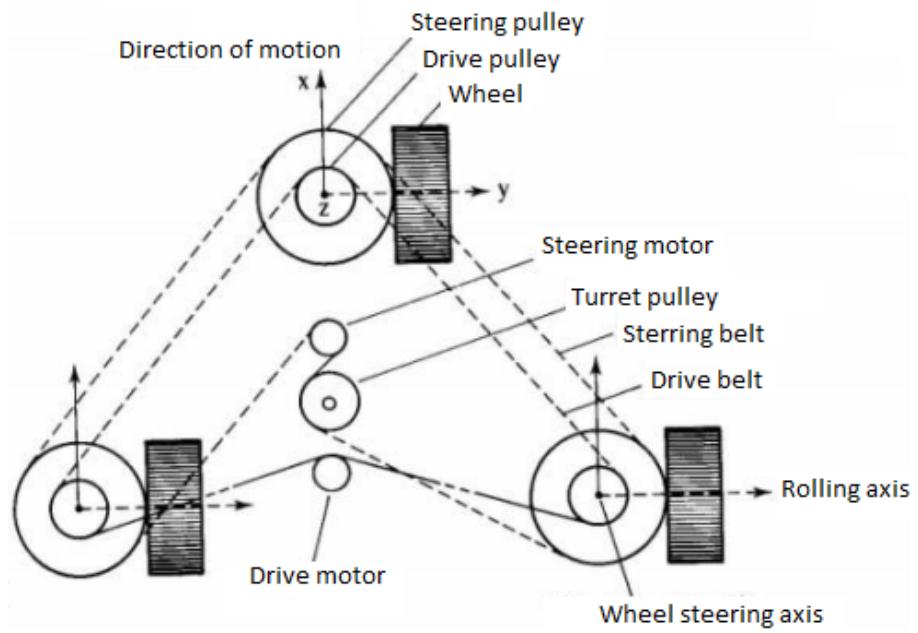


Figura 2.4: Robot síncrono [2]

2.4.2.3. Triciclo

Se basa en dos ruedas traseras que giran libremente y una tercera rueda delantera que se encarga de la tracción y dirección del vehículo. En la figura 2.5 se puede ver que tiene una mecánica simple, pero una cinemática más compleja. El centro de gravedad tiende a desplazarse cuando se mueve sobre una pendiente, causando una pérdida de tracción. [2]

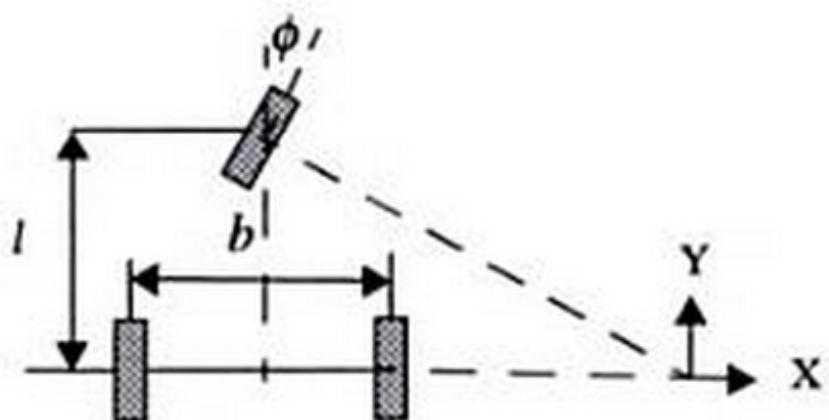


Figura 2.5: Robot triciclo

2.4.2.4. Ackerman

Es el modelo más usado en los vehículos de cuatro ruedas convencionales. Este sistema tiene más complejidad mecánica que el triciclo ya que utiliza las dos ruedas delanteras para controlar la dirección. La mayor ventaja de este sistema es que tiene buena estabilidad y su control en trayectorias rectas es simple. En la figura 2.6 se observa la estructura de un vehículo del tipo Ackerman.

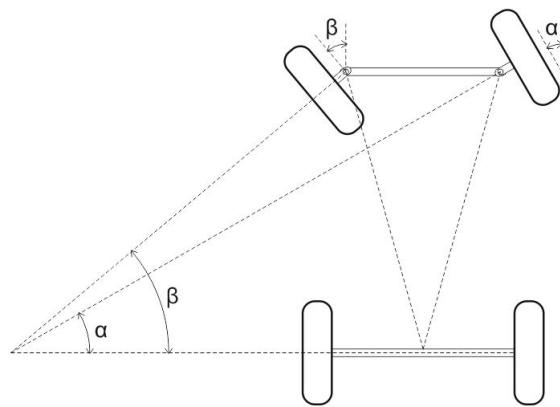
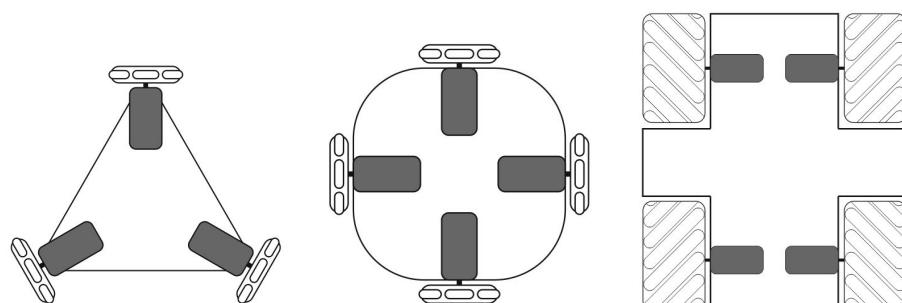


Figura 2.6: Robot Ackerman

2.4.2.5. Omnidireccional

El sistema de locomoción omnidireccional permite mayor libertad de movimiento que los sistemas de ruedas clásicos. Los robots que implementan este sistema pueden moverse en cualquier dirección sobre el plano y en cualquier momento sin tener que hacer movimientos previos para modificar su trayectoria. Requiere ruedas que permitan movimiento en más de una dirección, como es el caso de las omnidireccionales y las mecanum. Se requiere de un sistema de control para garantizar los movimientos en línea recta. Este sistema puede ser implementado con tres o cuatro ruedas como se muestra en la imagen 2.7



(a) Tres ruedas omnidireccionales (b) Cuatro ruedas omnidireccionales (c) Cuatro ruedas mecanum

Figura 2.7: Robot omnidireccional.

2.4.3. Restricciones holonómicas y no-holonómicas

Se refiere a la forma en que se mueve el robot o vehículo. Los distintos tipos de ruedas tienen diferentes propiedades cinemáticas. Restricciones cinemáticas [5]:

- Holonómica: los diferentes grados de libertad están desacoplados permitiendo que el robot pueda cambiar su dirección sin rotar previamente. Esto se puede realizar con las configuraciones diferencial, síncrona y omnidireccional.
- No holonómica: los grados de libertad están acoplados, de modo que para dar vuelta debe moverse primero hacia el frente y luego hacia atrás. Los sistemas que están contemplados en este caso son triciclo y Ackerman.

2.4.4. Componentes

2.4.4.1. Ruedas

Las ruedas omnidireccionales y las mecanum permiten la movilidad del robot en cualquier sentido de forma holonómica sin necesidad de usar actuadores encargados de la orientación.

"Las ruedas omnidireccionales poseen pequeños discos alrededor de la circunferencia que son perpendiculares a la dirección de laminación, como se observa en la figura 2.8. El efecto es que la rueda rodará en sentido de avance, pero también se desplazará lateralmente con gran facilidad. Estas ruedas son a menudo empleadas en los sistemas de accionamiento holonómicos ya que permiten más movimientos que una rueda fija." [4]

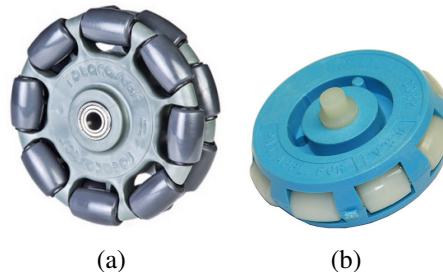


Figura 2.8: Ruedas omnidireccionales.

"Las ruedas mecanum permiten mover al vehículo en cualquier dirección. Es una rueda convencional con una serie de rodillos unidos a su circunferencia. Estos rodillos típicamente tienen cada uno un eje de rotación a 45° con respecto al plano de la rueda y en 45° a una línea a través del centro del rodillo paralelo al eje de rotación de la rueda (Figura 2.9)." [4]

2.4.4.2. Sensores

Un sensor es un dispositivo eléctrico y/o mecánico capaz de convertir magnitudes físicas, como la luz, velocidad, aceleración, presión, temperatura, etc, en otra magnitud, normalmente eléctrica, que sea posible manipular y cuantificar.

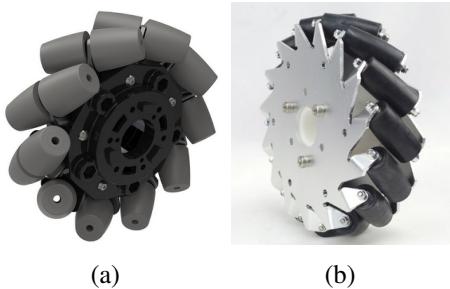


Figura 2.9: Ruedas mecanum.

- Sensor de distancia por sonar

“Estos sensores funcionan mediante el principio del tiempo de vuelo, emitiendo pulsos de sonido y determinando el tiempo hasta que se detecta una vez que ha sido reflejado por el objeto. De esta forma, teniendo en cuenta la velocidad de propagación del sonido, puede llegar a determinarse una distancia...”

... Sin embargo se cuenta también con sensores de proximidad que indican si existe o no un objeto a una distancia menor que una dada, la cual puede programarse en el sensor. Conviene poner de manifiesto que las características de la superficie que refleja la onda y el ángulo de incidencia tienen una notable influencia en la eficacia de estos sensores. En efecto si el ángulo de incidencia excede un cierto valor crítico, la energía reflejada no entrará en la zona de detección. Pueden recibirse también reflexiones desperdigadas de otros objetos generando señales falsas tal como se pone en manifiesto en la imagen 2.10. Nótese como en este caso pudiera no detectarse que el objeto O está muy próximo debido a que se recibe el rebote de O’.” [1]

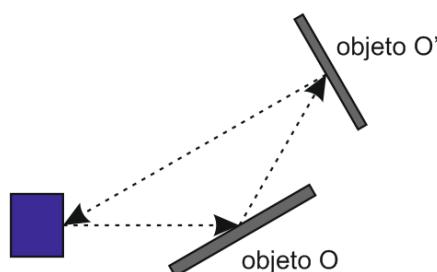


Figura 2.10: Funcionamiento sensor de distancia por sonar

- Sensor óptico de proximidad:

Es un detector óptico, en el cual se emplean diodos emisores de luz y fotodetectores tales como fotorresistores, fototransistores o fotodiódos. “Los fotorresistores, o fotocélulas, son resistencias cuyo valor cambia con la intensidad de luz recibida... Los fotodiódos poseen una mayor sensibilidad y producen una señal lineal en un rango muy amplio de niveles de intensidad. Sin embargo, su salida necesita ser amplificada.” [2] Se emplean también sensores ópticos en los que el emisor y el receptor se montan sobre un

mismo encapsulado, utilizados como detector de presencia por reflexión como muestra la figura 2.11. En este caso particular, se utilizan sensores infrarrojo que son sensibles a la longitud de onda, tales como 880nm inmediatamente inferiores a la visible. “Sin embargo, conviene poner de manifiesto que, en principio, no suministran ninguna medida de distancia, sino tan solo una señal binaria indicando si existe o no un objeto próximo en un rango de distancia característico del sensor” [2]

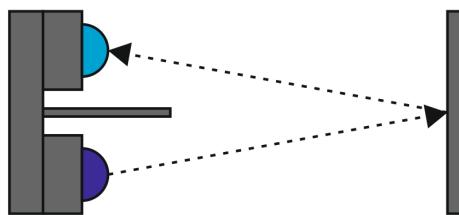


Figura 2.11: Sensor óptico

- Sensor de color:

“Un sensor de color está compuesto por un arreglo de fotodiodos con filtros. Estos filtros hacen a la matriz sensible a sólo una longitud de onda, roja, verde o azul. De esta manera la intensidad a la salida del sensor se corresponde con la intensidad de luz de la frecuencia filtrada.

La forma de uso imita al funcionamiento del ojo humano, por ejemplo, cuando este percibe el color rojo en un objeto, es porque la superficie absorbe todas las frecuencias de la luz exceptuando la blanca, de manera que una alta intensidad de una determinada frecuencia indica el mismo color.

La salida del sensor de color es una señal cuadrada cuyo ciclo de trabajo es proporcional a la intensidad de la luz detectada. De manera que se puede mediante cálculos matemáticos obtener el componente RGB del color percibido y luego obtener el color real.” [1]

- Acelerómetro: “Un acelerómetro es un dispositivo electro mecánico que mide aceleraciones, ya sean estáticas (gravedad) o dinámicas (producidas por movimientos). Por ello pueden utilizarse para medir variaciones del movimiento o la inclinación con respecto a la Tierra. Existen diferentes tipos de acelerómetros, algunos utilizan el efecto piezoelectrónico, esto quiere decir que contienen estructuras cristalinas microscópicas que son sensibles a las aceleraciones. Otra tipo mide cambios en la capacitancia, si hay dos microestructuras cercanas entre sí existe cierta capacidad entre ellas, si una fuerza mueve una de esas estructuras la capacidad varía. También existen algunos que utilizan el efecto piezoresistivo, burbujas de aire, luz, etc. Finalmente cabe aclarar que existen acelerómetros con salidas digitales o analógicas, de uno a tres ejes de medición, diferentes sensibilidades, velocidades máximas, etc.” [1]

- Magnetómetro:

“Se emplean para medir la orientación de un vehículo. Los compases son magnetómetros, es decir, sensores de medida de campo magnético, que se emplean para medir el campo

de la Tierra. Como se sabe, el campo magnético terrestre hace que una barra imantada se sitúe paralela con respecto a sus líneas de fuerza. Este principio se ha utilizado desde épocas remotas en las brújulas, en las cuales una aguja imanada colocada sobre un soporte vertical, que le sirve de apoyo y le permite girar libremente, apunta hacia el Norte. Si la brújula se coloca en un plano horizontal, puede emplearse para determinar orientaciones de dicho plano sobre una escala graduada. . . . Un aspecto muy importante es tratar de eliminar las perturbaciones del campo magnético terrestre, debidas a los materiales metálicos existentes en las proximidades, que generan una desviación en la medida de la orientación. Para ello se utilizan esferas de hierro con imanes permanentes ajustables en la base, mediante los cuales se compensan las desviaciones de las líneas de flujo geomagnéticas. Otro aspecto que es necesario tener en cuenta es la eliminación de las declinaciones magnéticas que hace que el ángulo marcado varíe en función del lugar y con el tiempo. Para ello se emplean cartas de declinaciones.” [2]

“Existen diferentes tipos de magnetómetros basados en diferentes principios de funcionamiento. La mayoría contiene un dispositivo sensible al campo magnético externo, algunos utilizan imanes permanentes y otros electroimanes, incluso existen aquellos que utilizan propiedades magnéticas del material con el cual están construidos.” [1]

- Giróscopo:

“Existen diversos tipos de giróscopos, nos interesan los giróscopos electrónicos; Son normalmente sensores de velocidad angular que emplean el efecto de Coriolis”. “Para ello se realizan micromecanizados del silicio configurando un anillo que se hace vibrar a una frecuencia de resonancia. El movimiento de rotación produce fuerzas de Coriolis que dependen de la velocidad de giro. La medida de la velocidad se obtiene determinando la diferencia de las vibraciones a diferentes ángulos. Un sensor típico puede tener dimensiones entre 2 y 3 milímetros y permite medir hasta 100 grados por segundo.” [1]

Los giróscopos presentan la ventaja de su independencia ante anomalías magnéticas presentes en el entorno respecto de los magnetómetros.

- Barómetro:

“Son dispositivos que permiten medir presión. Midiendo la presión atmosférica se puede medir la altitud y a partir de esta medida se pueden establecer ascensos y descensos del dispositivo. Estos sensores son en realidad transductores, generan una señal en función de la presión aplicada, en general esta señal es eléctrica. Existen diferentes tipos de sensores caracterizados por el rango de medición, la temperatura de operación y el tipo de presión que miden, ya sea atmosférica, absoluta, diferencias de presiones, etc. También pueden clasificarse según el método que utilizan para realizar las mediciones: piezoresistivos, capacitivos, electromagnéticos, piezoeléctricos, ópticos, etc.” [1]

2.4.4.3. Actuadores

Los actuadores tienen por misión generar el movimiento de los elementos del robot según las órdenes dadas por la unidad de control. De manera general, los actuadores utilizados en robótica pueden emplear energía neumática, hidráulica o eléctrica.

Actuadores eléctricos: Las características de control, sencillez y precisión de los accionamientos eléctricos ha hecho que sean los más usados. Dentro de los actuadores eléctricos pueden distinguirse tres tipos diferentes:

- Motores de corriente continua (DC):

- Controlados por inducción
- Controlados por excitación

- Motores de corriente alterna (AC):

- Síncronos
- Asíncronos

- Motores paso a paso [6]

Motor de corriente continua: “El motor de corriente continua (denominado también motor de corriente directa, motor CC o moto DC) es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción del campo magnético” [4] El torque generado es proporcional a la diferencia de potencial aplicado a los terminales de alimentación. El sentido de giro depende de la polaridad.

2.4.5. Sistema de control

“La cinemática es la ciencia que estudia el movimiento sin tomar en cuenta las fuerzas que lo causan. Dentro de la ciencia de la cinemática se estudia la posición, velocidad, aceleración y otras derivadas mayores de las variables de posición (con respecto al tiempo ó alguna(s) otra(s) variable(s)). Por lo tanto, el estudio de la cinemática de manipuladores refiere a todas las propiedades basadas en la geometría y el tiempo del movimiento. Las relaciones entre estos movimientos y las fuerzas y torques que los causan es un problema atacado por la dinámica.”

“Se representan las velocidades del robot deseadas como el vector $\{\dot{x}, \dot{y}, \dot{\theta}\}$ en coordenadas cartesianas y luego se describe como llegar a la configuración de las ruedas $\{\dot{\phi}_1, \dot{\phi}_2, \dots, \dot{\phi}_n\}$, determinando la velocidad angular de las ruedas.” [12]

Luego de un desarrollo matemático que se puede profundizar en el libro de John J. Craig [12], se obtiene la matriz de transformación 2.1 para el sistema de control de un robot omnidireccional de cuatro ruedas fijas.

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\sin \alpha_1 & \cos \alpha_1 & L \\ -\sin \alpha_2 & \cos \alpha_2 & L \\ -\sin \alpha_3 & \cos \alpha_3 & L \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \quad (2.1)$$

Por lo tanto, dada la velocidad lineal ($v = [\dot{x}, \dot{y}]$) y angular ($\dot{\theta}$) deseadas, las velocidades angulares de las ruedas requeridas $\dot{\phi}_i$ ($i=1,2,3,4$) pueden ser determinadas por el sistema de ecuaciones 2.1

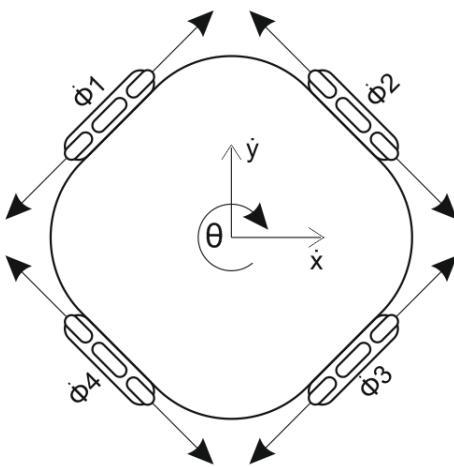


Figura 2.12: Esquema cinemático de robot omnidireccional de 4 ruedas.

2.4.6. Sistema de comunicación

2.4.6.1. Protocolo de comunicación I2C

I2C, abreviatura que deriva de Inter Integrated Circuits, es un protocolo de comunicación diseñado por Philips Semiconductors. Se usa para la comunicación entre microcontroladores, memorias y otros dispositivos. El bus I2C es bidireccional y utiliza dos líneas, una de datos (SDA) y una de reloj serie (SCL), que requiere resistencias de polarización a positivo (RPA). SCL es la línea de reloj y se usa para sincronizar todos los datos SDA de las transferencias durante I2C bus.

La arquitectura del bus I2C consta de un dispositivo maestro y uno o más dispositivos esclavos. El maestro es el encargado de manejar la línea de reloj SCL y los esclavos son los dispositivos que responden al maestro. Normalmente hay un solo dispositivo maestro y es él el que puede iniciar una transferencia a través del bus I2C. Tanto maestro como esclavos pueden transferir datos a través del bus I2C, pero es el maestro el que controla la transferencia.

Los dispositivos se conectan a las líneas SDA y SCL del bus I2C y se reconocen por su dirección. Al ser direcciones de 7bits se puede tener hasta 128 dispositivos conectados en el bus.

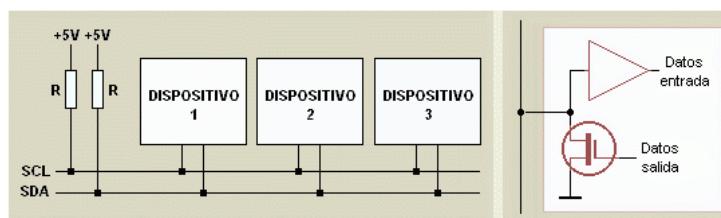


Figura 2.13: Comunicación I2C [8]

2.4.6.2. Bluetooth

La comunicación se establece por radiofrecuencia de forma que los dos dispositivos no tiene que estar alineados y pueden estar en distintos ambientes siempre que la potencia de transmisión sea suficiente. Los dispositivos Bluetooth se clasifican según su potencia de transmisión en Clase 1, Clase 2 y Clase 3. Se mantiene una compatibilidad entre dispositivos que sean de distintas clases. Las características principales son:

Clase	Potencia máxima permitida [mW]	Potencia máxima permitida [dBm]	Alcance aproximado [m]
Clase 1	100mW	20dBm	100m
Clase 2	2.5mW	4dBm	5 - 10m
Clase 3	1mW	0dBm	1m

Tabla 2.1: Clases de dispositivos Bluetooth.

“Durante el uso normal, un dispositivo funciona en modo pasivo, es decir, que está escuchando la red. El establecimiento de una conexión comienza con una fase denominada “solicitud”, durante la cual el dispositivo maestro envía una solicitud a todos los dispositivos que encuentran dentro de su rango, denominados “puntos de acceso”. Todos los dispositivos que reciben la solicitud responden con su dirección. El dispositivo maestro elige una dirección y se sincroniza con el punto de acceso mediante una técnica denominada “paginación”, que principalmente consiste en la sincronización de su reloj y frecuencia con el punto de acceso. De esta manera se establece un enlace con el punto de acceso que le permite al dispositivo maestro ingresar a una fase de “descubrimiento del servicio” del punto de acceso, mediante un protocolo denominado SDC (Service Discovery Protocol). Cuando esta fase de descubrimiento del servicio finaliza, el dispositivo maestro está preparado para crear un canal de comunicación con el punto de acceso, mediante el protocolo L2CAP. El punto de acceso puede incluir un mecanismo de seguridad denominado emparejamiento, que restringe el acceso sólo a los usuarios autorizados para brindarle a la piconet cierto grado de protección. El emparejamiento se realiza con una clave cifrada comúnmente conocida como “PIN” (Personal Information Number). Para esto, el punto de acceso le envía una solicitud de emparejamiento al dispositivo maestro. La mayoría de las veces se le solicitará al usuario que ingrese el PIN del punto de acceso. Si el PIN recibido es correcto, se lleva a cabo la conexión. En el modo seguro, el PIN se enviará cifrado con una segunda clave para evitar poner en riesgo la señal. Cuando el emparejamiento se activa, el dispositivo maestro puede utilizar libremente el canal de comunicación establecido.” [7]

CAPÍTULO 3

ITERACIÓN 1

3.1. Introducción

Para el desarrollo del proyecto se comienza con la etapa de diseño, por lo que en esta iteración se estudia y determina la arquitectura de hardware a usar.

Al mismo tiempo se investigan las características de las diferentes placas de desarrollo disponibles en el mercado y se selecciona aquella que más se ajuste a los requerimientos.

El próximo paso es estudiar los distintos sistemas de locomoción, y seleccionar aquel que permita movimientos de forma holonómica, éste será el que determine el sistema mecánico.

Por último se desarrolla un sistema de alimentación integral y estándar para alimentar los distintos componentes que se incorporarán a lo largo del desarrollo.

3.2. Requerimientos

Los requerimientos mencionados a continuación emergen de la experiencia obtenida durante el desarrollo del proyecto Hermes [1] con el fin de realizar una actualización tecnológica del sistema.

- La placa de desarrollo debe contar con:
 - 4 puertos de pwm para control de velocidad de los motores
 - 4 puertos de interrupción para la medición de rpm
 - Comunicación I2C
 - Comunicación serial
 - 10 o más puertos I/O digitales
 - Debe poder soportar un Sistema Operativo (memoria y capacidad de procesamiento) para permitir la programación en distintos lenguajes
 - El costo no debe ser superior a los \$1000
- Arquitectura de hardware modular.
- El sistema de alimentación debe estar unificado y ser estándar.

3.3. Desarrollo

3.3.1. Selección de la placa de desarrollo

Para el estudio y selección de la placa de desarrollo se tienen en cuenta los requerimientos de software y hardware planteados. En la tabla 3.1 se realiza un análisis de las siguientes placas que se encuentran disponibles actualmente en el mercado argentino: Arduino Mega 2560, Arduino Due, Raspberry Pi B+, Freescale FRDM-K64, CubieBoard.

	Arduino Mega 2560	Arduino Due	Raspberry	FRDM- K64	CubieBoard
Precio	32(480)	40(600)	60(900)	66(990)	100(1500)
Lenguaje de programación	C	C	C, Java, Python	C, Java	C, Java, Python
Memoria Flash	256Kb	512Kb	micro sd	1Mb	4Gb
SO	No tiene	No tiene	Linux, externo en la memoria sd	RTOS	Linux, Android
Puertos de interrupciones externas	6	Todos los pines GPIO	GPIO configurables	Todos los pines GPIO	Todos los pines GPIO
Puertos GPIO	54	54	26		96
Puertos PWM	15	12	1	12	2
Procesador	ATmega 2560	ARM cor- tex M3	ARM11	ARM Cor- tex M4	ARM cor- tex A8
Velocidad del procesador	16MHz	85MHz	700MHz	120MHz	1GHz
RAM	8Kb	96Kb	512Mb	256Kb	1Gb

Tabla 3.1: Características de las placas de desarrollo

Analizando las principales características de cada placa se puede ver que todas cumplen parcialmente con los requerimientos de software y hardware.

Puesto que consideramos más importante el sistema operativo procedemos a analizar las ventajas y desventajas de las tres placas de desarrollo que trabajan bajo un sistema operativo: Raspberry Pi, FRDM-K64 y CubieBoard.

Se descarta CubieBoard dado que posee prestaciones similares a las otras dos y su costo es superior al establecido en los requerimientos.

Analizamos las placas de desarrollo Raspberry Pi y FRDM-K64 a nivel de sistema operativo. La primera trabaja con diferentes distribuciones de Linux optimizadas para esta arquitectura, de las cuales se dispone de abundante documentación.

Por otro lado, la segunda placa opera con el sistema operativo MQX RTOS (Real Time Operating System), el cual es software propietario del fabricante.

Como el sistema está destinado principalmente a ser usado en materias de la carrera Ingeniería en computación y, dado que los alumnos poseen conocimiento sobre sistemas operativos Linux, se considera como punto a favor el uso de Raspberry Pi.

Para la programación de la placa FRDM-K64, es necesario tener conocimientos del entorno de desarrollo CodeWarrior. En cambio la programación de Raspberry Pi, puede ser realizada en cualquier entorno de programación.

Analizando las ventajas y desventajas de cada placa, se decide usar la placa de desarrollo Raspberry Pi B+ la cual nos da la posibilidad de trabajar con un Sistema Operativo open source, admite comunicación I2C, comunicación serial, dispone de numerosos puertos GPIO para el uso de sensores y puede ser programada en diferentes lenguajes y entornos de programación.

El inconveniente de esta placa es que no dispone de la cantidad suficiente de puertos PWM, por lo que decidimos incorporar un microcontrolador que disponga de cuatro o más puertos PWM.

Para el control de los motores se decidió anexar una placa de desarrollo Arduino Pro Mini, con un microcontrolador ATMEGA328. Esta placa dispone de 6 puertos de PWM y comunicación I2C.

3.3.2. Arquitectura de hardware

La arquitectura de hardware se plantea de forma general en el gráfico 3.1

Se diseñó una arquitectura modular con cuatro subsistemas: comunicación, sensores, actuadores y placa de desarrollo.

La placa de desarrollo, subsistema principal (Raspberry Pi), interactúa con los otros subsistemas a través de sus correspondientes interfaces.

El subsistema comunicación es la interfaz entre el usuario y la placa de desarrollo.

El subsistema sensores está formado por los sensores que son los encargados de captar las condiciones del entorno y enviar datos a la placa principal para su proceso.

El subsistema actuadores recibe información de la placa de desarrollo para ejecutar una acción. En este caso los actuadores son cuatro motores que reciben de la placa de desarrollo información de velocidad y sentido de giro. Además implementa un control PID para mantener la velocidad de giro de dichos motores.

A lo largo del informe se irá explicando el desarrollo de cada subsistema.

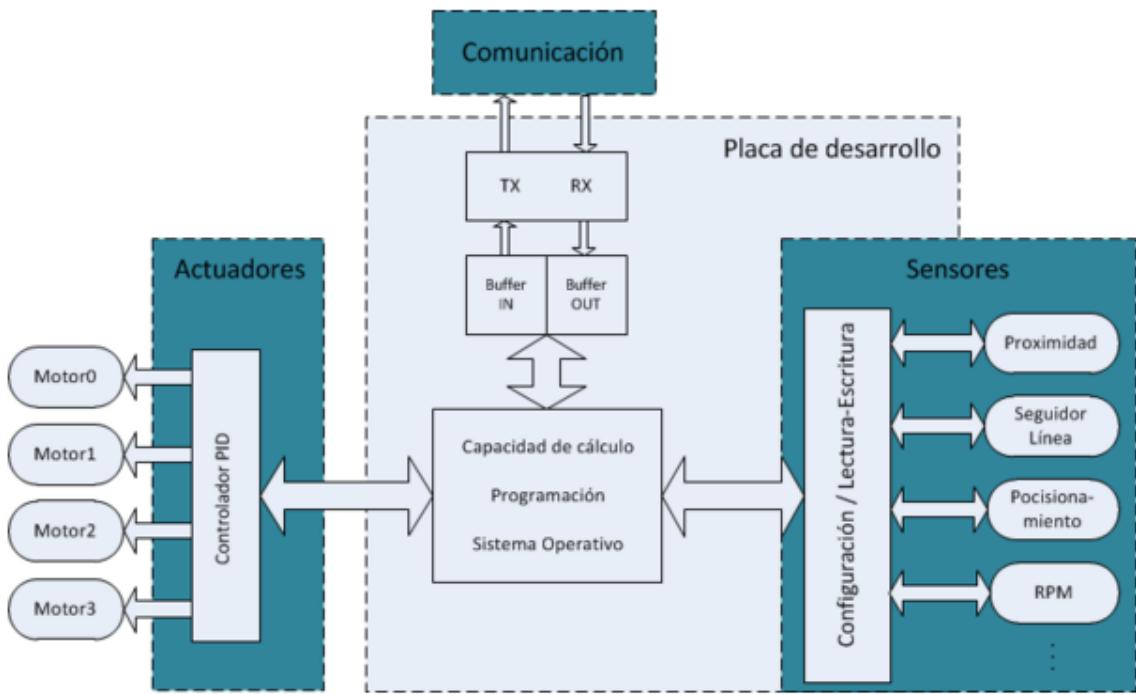


Figura 3.1: Arquitectura de hardware del sistema

3.3.3. Selección del sistema de locomoción

Luego del desarrollo realizado durante la práctica profesional [9] decidimos mantener el sistema mecánico omnidireccional ya que le permite al robot moverse de forma holonómica. Al realizar esta nueva versión buscamos diseñar el control del robot de tal forma que éste pueda desplazarse en cualquier dirección del plano, es decir que se desplace en una dirección y sentido especificada por el usuario.

3.3.4. Selección del sistema de alimentación

Una de las mejoras para la segunda versión de Hermes es usar un sistema de alimentación unificado y estandarizado para todos los componentes del robot (microcontroladores, actuadores y sensores). La tensión de la batería debe ser de 12V para alimentar los drivers de los motores, y la alimentación de la electrónica se obtienen mediante reguladores de tensión a 5V.

Para la selección del sistema de alimentación, se analizan tres tipos de baterías recargables. Litio-Ion, Litio-Polímero y NiMH (Níquel Metal Hidruro)

Luego de realizar un análisis entre los tres tipos de baterías seleccionadas en la tabla 3.2, se descartan las de tipo NiMH, debido a la alta tasa de autodescarga y la baja tensión por celda, ya que se necesitarán diez celdas para alcanzar la tensión deseada aumentando el peso final de la batería.

	Litio-Ión	Litio-Polímero	NiMH
Energía Precio [Wh/U\$S]	2,5	2,2	2,75
Voltaje por célula [V]	3,7	3,7	1,2
Capacidad [mAh]	1000-1800	230-4500	500-3800
Energía específica [Wh/kg]	100-265	130-200	60-120
N de recargas	400-1200 aprox	500 aprox	500-200aprox
Tasa de autodescarga [%/mes]	5	5	20
Máxima corriente de descarga	1C	Hasta 20C	1C

Tabla 3.2: Tipos y características de baterías

En el caso de usar celdas de Litio-Ion o Litio-Polímero basta con usar un pack armado con tres celdas. Se puede observar que la energía específica y precio por vatio de las celdas de Li-Ion y Li-Po son similares, de modo que se opta por las celdas de Li-Ion ya que tolera mayor cantidad de recargas, prolongando su vida útil.

El proveedor de baterías Probattery [11] confeccionó la batería que se observa en la figura 3.2 con las siguientes características:

- Tipo: Litio Ión recargable
- Tensión: Tres celdas (11.1V)
- Capacidad: 4100mA/h.
- Peso: 240g



Figura 3.2: Batería seleccionada

En base al consumo estimado del robot (1500mAh) y dado que la capacidad de la batería es de 4100 mAh, la autonomía estimada del robot en funcionamiento constante es de 2 horas 45 minutos.

3.3.4.1. Reductor de tensión de alimentación

Como el sistema de alimentación seleccionado es de 11,1V y la electrónica de Hermes II se alimenta con 5V, es necesario incorporar un reductor de tensión, el cual consta además de un limitador de corriente. (figura 3.3)

Las características del módulo seleccionado son:

- Tensión de entrada: DC 7- 24V
- Tensión de salida: DC 5V
- Corriente de salida: 3A (max)
- Eficiencia de conversión: 96 % (MAX)
- Frecuencia de switcheo: 340KHz
- Ripple de salida: 30mV (max)
- Regulación de carga: 0.5 %
- Regulación de tensión: 2.5 %
- Temperatura de trabajo: -40°C to +85°C
- Con limitador de corriente
- Dimensiones: 6.2 x 2.1 x 1 cm
- Peso: 14 g

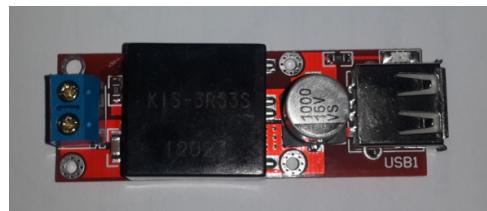


Figura 3.3: Reductor de tensión

3.4. Resultados

Se diseñó una arquitectura de hardware de forma modular.

La placa de desarrollo seleccionada (Raspberry Pi B+) cubre parcialmente todos los requerimientos de hardware, salvo los puertos PWM necesarios para el control de motores. Por esto se incorpora a la arquitectura de hardware una placa de desarrollo arduino con comunicación I2C, para expansión de puertos PWM.

Con respecto al sistema de alimentación, se escoge una batería de 11,1V para alimentación de motores, y se incluye un regulador de tensión de 5V para alimentación de la parte electrónica

3.5. Conclusiones

La arquitectura diseñada en esta iteración permite plantear el desarrollo del proyecto por componentes.

Se decide utilizar como unidad de procesamiento principal la placa de desarrollo Raspberry Pi B+, la cual soporta diferentes sistemas operativos libres. En una iteración posterior se analizarán los sistemas operativos disponibles y se seleccionará el más conveniente.

El sistema de locomoción no se modifica, por lo que se seguirá trabajando con una arquitectura omnidireccional de cuatro ruedas.

Se debe tener en cuenta el peso y tamaño de la batería para el diseño del chasis y la elección de los motores en posteriores iteraciones, como así también la distribución de masas generada por la localización de diversos componentes.

CAPÍTULO 4

ITERACIÓN 2

4.1. Introducción

Durante el desarrollo de la práctica profesional [9] se detectaron problemas de desplazamiento derivados del uso de ruedas inadecuadas. Además, el costo de los motores utilizados anteriormente aumentó significamente.

En esta iteración se investiga cuales son los componentes disponibles en el mercado y se selecciona aquellos que permitan corregir las desviaciones en el desplazamiento.

Una vez seleccionados los motores se rediseña el chasis y la distribución de los componentes para aprovechar el espacio y distribuir correctamente el peso sobre la superficie. Se adaptan los soportes para los motores y los bujes para las ruedas.

Para el control de los motores se desarrolla una biblioteca en lenguaje C. En ella se especifican los puertos de conexión de salidas de modulación por ancho de pulso (Pulse-Width-Modulation) (PWM) y los usados para la selección del sentido de giro de los motores.

Se desarrolla el código necesario para establecer la comunicación Bluetooth entre un dispositivo y Arduino para hacer las pruebas de movimiento de forma inalámbrica.

Por último se agrega el hardware correspondiente al sistema para medir la velocidad de los motores e implementar un sistema de control de velocidad Proporcional Integral Derivativo (PID) de los mismos.

4.2. Requerimientos

- El robot tiene que ser de tamaño inferior a 30cm.
- El robot tiene que recibir comandos bajo el protocolo de comunicación Bluetooth.
- El robot debe ser capaz de interpretar comandos para ejecutar acciones.
- El robot debe poder cambiar la velocidad y sentido de giro de los motores para su control.
- Se debe poder medir la velocidad de cada motor.
- Cada motor debe implementar un sistema de control de velocidad.

4.3. Desarrollo

4.3.1. Selección de ruedas

Para el prototipo original diseñado en la práctica profesional [9] se adquirieron ruedas con rodillos de plástico. Al momento de realizar pruebas se percibió que el material usado para éstos es sumamente importante, ya que se necesita que las ruedas traccionen para que el robot no se desvíe de su trayectoria.

Se decide reemplazar las ruedas por un modelo similar que cuenta con rodillos de goma.

4.3.2. Selección de motores

Los motores utilizados en la primer versión de Hermes II [9] incrementaron su costo en un 50 % de modo que se investiga la existencia de otras opciones disponibles en el mercado.

De la tabla 4.1 se selecciona el motor que más se asemeja al utilizado en la primer versión de Hermes II [9].

RPM	Torque	Relación de caja
2200RPM	3 oz-in	10:1 MP
730RPM	8 oz-in	30:1 MP
420RPM	13 oz-in	50:1 MP
290RPM	17 oz-in	75:1 MP
150RPM	24 oz-in	150:1 MP
75RPM	46 oz-in	298:1 MP

Tabla 4.1: Motores disponibles

El motor seleccionado presenta las características expuestas en la tabla 4.2

RPM	Torque	Relación de caja
420RPM	13 oz-in	50:1 MP

Tabla 4.2: Características del motor seleccionado (figura 4.1)



Figura 4.1: Motor DC

4.3.3. Diseño de nuevas bridas y adaptación de bujes

A diferencia de los motores con los que contaba la versión anterior del robot Hermes II [9], los motores seleccionados en el punto 4.3.2 poseen el eje en uno de sus extremos, lo que lleva a una modificación del soporte del sensor de medición de velocidad.

Además se rediseñan los soportes para los motores y los bujes de adaptación de los ejes. Las piezas de la figura 4.2 se modelan en un programa de diseño y se cortan con una máquina de corte láser en mdf.

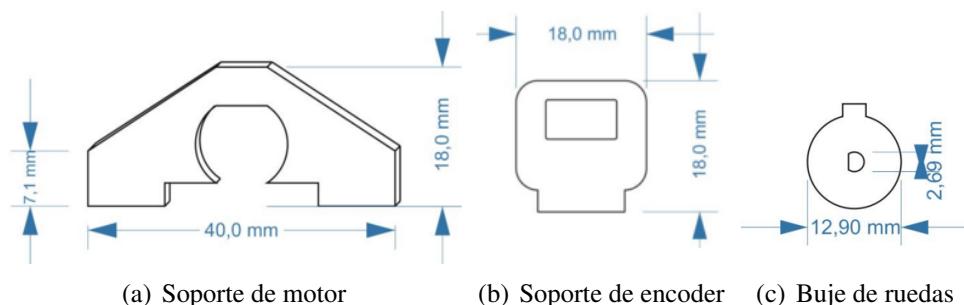


Figura 4.2: Piezas encastrables.

4.3.4. Armado del chasis

Para que el robot pueda ser armado por estudiantes de la carrera, se decide continuar con el diseño de un chasis con piezas encastrables. En la imagen 4.3 se puede observar el chasis al cual se le incorporaron las nuevas bridas y motores.

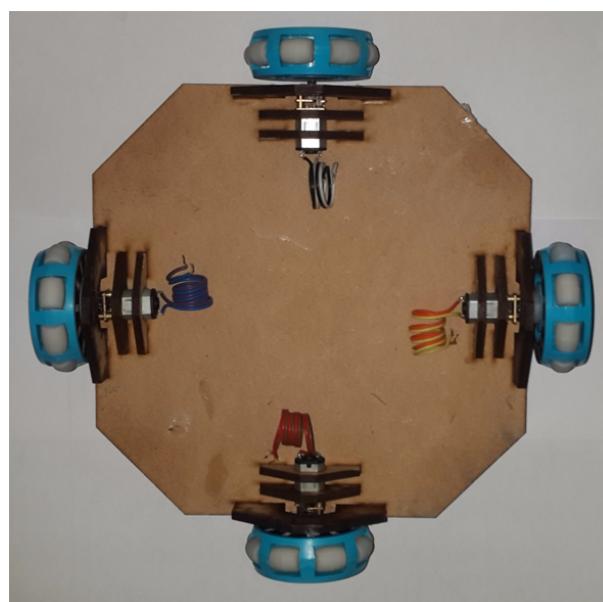


Figura 4.3: Vista superior del chasis

4.3.5. Comunicación Bluetooth con Arduino

Es necesario que el robot pueda recibir una orden para ejecutar una acción. Para ello se necesita establecer una comunicación entre el robot y el dispositivo que enviará las órdenes. Para establecer dicha comunicación se usa el módulo HC-05 y se realiza una conexión como se muestra en la figura 4.4.

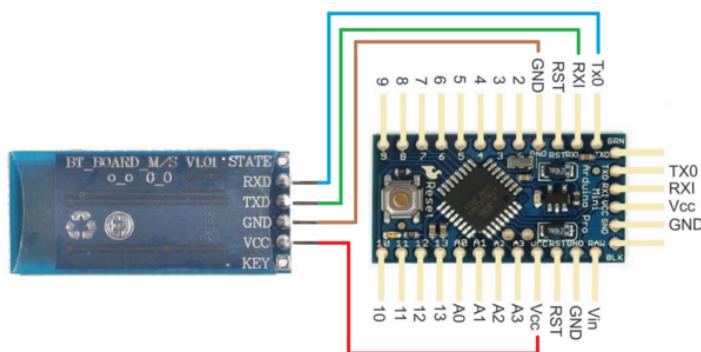


Figura 4.4: Conexión de módulo Bluetooth a Arduino

Cabe destacar que la comunicación Bluetooth con Arduino se realiza en una primera instancia para trabajar de forma inalámbrica con el robot y poder realizar las primeras pruebas. En una iteración posterior el encargado de la comunicación con el usuario será el microcontrolador principal, en este caso Raspberry.

La placa de desarrollo Arduino dispone de un puerto de comunicación serial. Las funciones utilizadas para la comunicación son las siguientes:

```
Serial.begin(baudrate);  
Serial.available();  
Serial.Read();  
Serial.println();
```

4.3.6. Desarrollo de biblioteca para control de motores

Para simplificar el control de los motores se desarrolla la biblioteca DriverMotor, la cual posee las siguientes funciones:

```
DriverMotor(uint8_t motornum); //Crea un motor  
setSpeed(uint8_t motor_speed); //Setea la velocidad del motor (PWM)  
motorStart(uint8_t direccion); //Setea el sentido de giro del motor
```

Esta biblioteca se encuentra disponible en el anexo A.

4.3.7. Medición de velocidad de motores

Para el control de velocidad de los motores es necesario obtener una medición de la misma. Para medir la velocidad en revoluciones por minuto se coloca un sensor infrarrojo (figura 4.6) y un patrón impreso en la rueda (figura 4.5), la cual consta de marcas blancas sobre un fondo negro que generan interrupciones en el microcontrolador cada vez que el sensor detecta un cambio de negro a blanco.

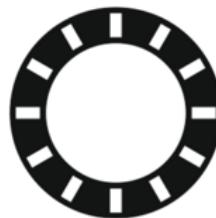


Figura 4.5: Patrón impreso en la rueda

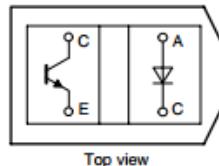


Figura 4.6: Sensor infrarrojo [13]

Se plantean tres métodos para calcular las RPM a través de la lectura de las interrupciones generadas por el sensor.

4.3.7.1. Primer método

Mediante este método se cuentan las interrupciones generadas por el sensor óptico en una base de tiempo fija controlada por timer. Este comportamiento se observa en la figura 4.7.

Para calcular la velocidad en RPM de los motores se necesita medir la cantidad de flancos, en este caso ascendentes, que se produjeron en un tiempo fijo. Para ello se establece una base de tiempo fija mediante una interrupción de timer y se cuenta la cantidad de interrupciones generadas. Si se aumenta la velocidad del motor, aumentará consecuentemente la cantidad de interrupciones y como la base de tiempo es fija, podemos calcular las RPM.

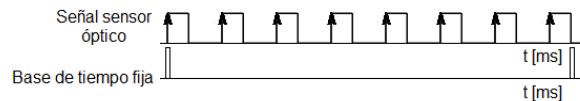


Figura 4.7: Primer método de medición de RPM

El patrón consta de 12 marcas blancas a lo largo de la circunferencia negra, el cual permite calcular las revoluciones por minutos (RPM) de la siguiente forma:

- Nran: cantidad de ranuras en una vuelta de la rueda (12 en este caso).
- BT (ms): Base de tiempo
- Nint: Número de interrupciones
- Kt (60000): Factor de conversión de tiempo

$$NúmeroDeVueltas = \frac{Nint}{Nran} \quad (4.1)$$

$$CantidadDeVueltasEnBT = \frac{\frac{Nint}{Nran}}{BT} \quad (4.2)$$

Multiplicando (4.2) por Kt obtenemos la cantidad de vueltas por minuto como se observa en 4.3

$$RPM = \frac{\frac{Nint}{Nran}}{BT} * Kt \quad (4.3)$$

Este cálculo se realiza para cada rueda, obteniéndose la velocidad en RPM correspondiente de forma independiente.

Este método debe esperar un tiempo BT fijo para obtener un valor de RPM.

4.3.7.2. Segundo método

Se mide el tiempo que transcurre entre dos interrupciones consecutivas generadas por una transición de blanco a negro, tomando el tiempo del procesador en cada interrupción y calculando la diferencia como se ilustra en la imagen

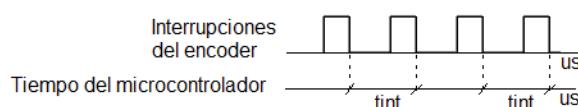


Figura 4.8: Segundo método de medición de RPM

Se utiliza la función micros() para tomar el tiempo entre interrupciones y se realiza el siguiente cálculo. Teniendo en cuenta que:

- Nran: cantidad de ranuras que posee el patrón de la rueda.
- tint (ms):tiempo entre dos interrupciones consecutivas tomado por la función micros.
- $tv = tint * Nran$: tiempo que tarda en dar una vuelta la rueda.
- Factor de conversión de tiempo $Kt = 60000$

$$RPM = \frac{Kt}{tv} \quad (4.4)$$

$$RPM = \frac{Kt}{tint * Nram} \quad (4.5)$$

Si suponemos que Nram es 12

$$RPM = \frac{60000}{tint * 12} \quad (4.6)$$

Este método permite calcular en forma instantánea las RPM por cada interrupción.

4.3.7.3. Tercer método

Se mide el tiempo que transcurre entre dos interrupciones consecutivas generadas por una transición de blanco a negro, contando la cantidad de interrupciones del timer que se generaron cada 0,1 ms. En la imagen 4.9 se observa el método de medición.

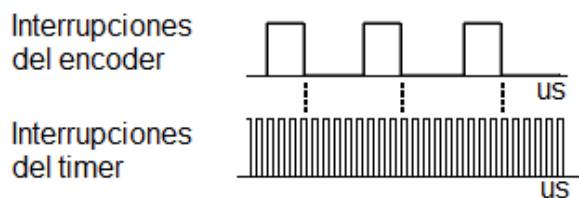


Figura 4.9: Tercer método de medición de RPM

Se mide la cantidad de interrupciones del timer que suceden entre dos interrupciones sucesivas generadas por el encoder. Teniendo en cuenta que:

- Nran: cantidad de ranuras que posee el patrón de la rueda
- tint (ms): período de interrupción del timer
- Nint: cantidad de interrupciones detectadas

-
- Kt: Factor de conversión de tiempo

$$RPM = \frac{Kt}{Nint * tint * Nran} \quad (4.7)$$

Suponiendo que Nran = 12

$$RPM = \frac{60000}{Nint * 0,1 * 12} \quad (4.8)$$

Este método al igual que el segundo, permite calcular las RPM de forma instantánea por cada interrupción de hardware generada cada encoder.

4.3.8. Análisis de métodos de medición de RPM

4.3.8.1. Primer Método

La cantidad de muestras indica la velocidad con la que gira la rueda, si la rueda se encuentra detenida, la cantidad de interrupciones es cero. El cálculo de RPM se realiza en un período fijo. Si este período es demasiado grande, se obtienen las muestras a una frecuencia muy baja y esto genera un retardo en la respuesta del sistema de control cuando se agrega el PID. Por el contrario, si este período es demasiado corto, la cantidad de muestras que se toman es muy pequeña y el error que se comete es mayor.

La incertidumbre que se tiene con este método es de ± 1 interrupción, pero este valor de ± 1 interrupción, no implica el mismo porcentaje de incertidumbre si la base de tiempo cambia. Suponiendo que en un período de 50 ms se miden 3 pulsos, el valor de revoluciones por minuto usando (4.3) es:

$$\begin{aligned} Nint = 2 &\Rightarrow RPM = 400 \\ Nint = 3 &\Rightarrow RPM = 600 \\ Nint = 4 &\Rightarrow RPM = 800 \\ \text{Incertidumbre} &: \pm 33,4\% \end{aligned}$$

Incrementando la base de tiempo a 100ms y asumiendo que la velocidad del motor permanece constante, la cantidad de muestras debería ser el doble. Tomando como incertidumbre ± 1 interrupción se obtiene:

$$\begin{aligned} Nint = 5 &\Rightarrow RPM = 500 \\ Nint = 6 &\Rightarrow RPM = 600 \\ Nint = 7 &\Rightarrow RPM = 700 \\ \text{Incertidumbre} &: \pm 16,7\% \end{aligned}$$

Si se aumenta aún más la base de tiempo (500ms), y suponiendo que la velocidad de los motores no varía, se puede ver como la incertidumbre disminuye.

$$N_{int} = 29 \Rightarrow RPM = 580$$

$$N_{int} = 30 \Rightarrow RPM = 600$$

$$N_{int} = 31 \Rightarrow RPM = 620$$

$$Incertidumbre : \pm 3,4\%$$

Como puede observarse, al aumentar la base de tiempo la incertidumbre disminuye pero también disminuye la frecuencia de muestreo. Por esto se deben tener en cuenta estas dos variables a la hora de seleccionar una base de tiempo.

4.3.8.2. Segundo Método

El problema de este método ocurre cuando los motores se encuentran detenidos ya que la base de tiempo no es fija y ésta tiende a un valor infinito. Además, la frecuencia con la que se obtiene un dato depende de la velocidad de giro del motor, así cuanto más rápido gire obtendremos datos con mayor frecuencia. Esto traerá problemas a la hora de realizar el control de velocidad, ya que éste necesita de una base de tiempo fija. Si los motores se encuentran detenidos el tiempo entre interrupciones tenderá a infinito, la fórmula 4.6 indica que, en teoría, la velocidad en RPM será 0 pero en la práctica, al no haber interrupción del encoder, el dato nunca será calculado.

4.3.8.3. Tercer Método

El problema que se detecta en este método es que a medida que se disminuye la velocidad, aumenta la cantidad pulsos que hay entre dos interrupciones externas del encoder, pudiendo generar un desbordamiento del contador y una lectura errónea del dato. Otro problema detectado en este método, es la gran carga sobre los recursos, ya que usar un timer para incrementar el contador cada 100 us implica usar ese timer solo para esta tarea y esto genera conflictos con el manejo de los PWM.

4.3.9. Selección del método de medición de RPM

Como se analizó en el apartado anterior sólo el primer método puede ser usado para un controlador PID ya que tiene una base de tiempo fija. La principal desventaja de este método es que genera una gran cantidad de interrupciones.

4.3.10. Sistema de control de velocidad

Luego de tener la medición de la velocidad, se desarrolla un sistema de control sobre un motor.

En una primera instancia se desarrolla un controlador proporcional como se muestra en la figura 4.10, en el cual la salida del mismo es el producto de la señal de error y la ganancia proporcional.

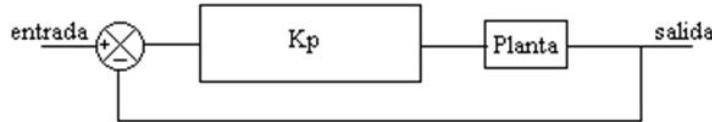


Figura 4.10: Controlador proporcional

Al probar el controlador se detecta que el sistema oscila debido a la incertidumbre de la medición de la velocidad del motor. Para reducir dicha incertidumbre se aumenta la precisión, incrementando la cantidad de pasos en el patrón del encoder impreso en la rueda. Esto genera una mayor cantidad de interrupciones lo que permite, manteniendo la base de tiempo, disminuir el porcentaje de incertidumbre calculado con la fórmula 4.3.

Luego de realizar esta modificación se prueba nuevamente el control proporcional, detectando que el microcontrolador es desbordado por la cantidad de interrupciones, produciendo mediciones incorrectas.

Como trabajo futuro se propone realizar las mediciones utilizando un contador externo, liberando al microcontrolador del tratamiento de interrupciones.

Un contador externo es un módulo de hardware independiente del microcontrolador que cuenta impulsos externos, en este caso del sensor infrarrojo.

El acceso al contador puede ser de dos formas:

- Por consultas desde el programa principal.
- Por activación de una interrupción al programa principal cuando el contador se desborda.

Además se propone utilizar el método Ziegler y Nichols para la sintonización del controlador proporcional, integral, derivativo.

El método “Ziegler y Nichols en lazo cerrado” o también llamado “de oscilaciones sostenidas” se profundiza en [14] .

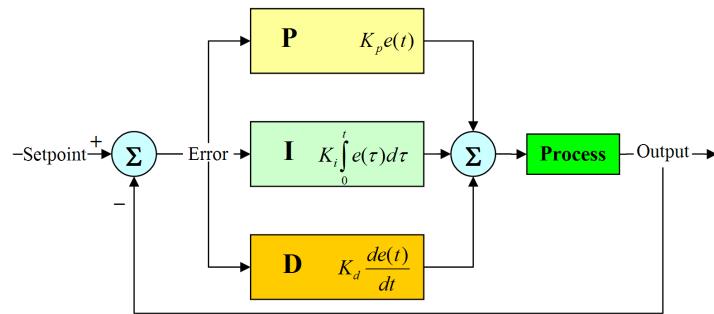


Figura 4.11: Controlador proporcional, integral, derivativo

4.4. Pruebas

4.4.1. Comunicación Bluetooth

Caso de prueba 1	Comunicación Bluetooth
Objetivo	Verificar que el robot pueda recibir un comando y responder.
Prerrequisitos	<ul style="list-style-type: none"> ■ Debe haber conectado un módulo Bluetooth a la placa de desarrollo. ■ Debe estar en ejecución una aplicación para la comunicación en la placa de desarrollo.
Pasos	<ul style="list-style-type: none"> ■ El usuario inicia la comunicación desde un dispositivo que disponga conectividad Bluetooth. ■ El usuario envía un comando.
Resultado esperado	<ul style="list-style-type: none"> ■ El usuario debe recibir en el dispositivo un eco del comando enviado.
Resultado obtenido	<ul style="list-style-type: none"> ■ El usuario recibe el mismo comando que envió.
Código de prueba	Anexo B.1

Tabla 4.3: Caso de prueba 1

4.4.2. Biblioteca

Caso de prueba 2	Prueba de la biblioteca
Objetivo	Verificar que la velocidad y sentido de giro del motor se puede cambiar.
Prerrequisitos	<ul style="list-style-type: none">■ La biblioteca desarrollada tiene que estar incorporada en el entorno de Arduino.■ Los motores que se desean probar deben estar conectados a los drivers que proveen la etapa de potencia.
Pasos	<ul style="list-style-type: none">■ Crear un proyecto nuevo■ Instanciar los motores que se desea probar.■ Indicar velocidad y sentido de giro de los motores.
Resultado esperado	<ul style="list-style-type: none">■ Que el sentido de giro de los motores y la velocidad de los motores cambie según se lo indique.
Resultado obtenido	<ul style="list-style-type: none">■ El sentido de giro de los motores cambia según se lo ordenó.■ La velocidad de los motores varía cuando se lo ordena.
Código de prueba	Anexo B.2

Tabla 4.4: Caso de prueba 2

4.4.3. Funcionamiento del robot

Caso de prueba 3	Funcionamiento del robot
Objetivo	Verificar que el robot es capaz de recibir un comando, procesarlo y ejecutar el movimiento correspondiente
Prerrequisitos	<ul style="list-style-type: none">■ El sistema debe estar alimentado.■ Debe estar corriendo la aplicación en la placa de desarrollo.
Pasos	<ul style="list-style-type: none">■ Establecer la comunicación entre el robot y el dispositivo que lo controlará.■ Enviar un comando.
Resultado esperado	<ul style="list-style-type: none">■ Que el robot sea capaz de procesar el comando recibido.■ Que se ejecute la acción correspondiente.
Resultado obtenido	<ul style="list-style-type: none">■ A baja velocidad el robot no es capaz de romper la inercia.■ A alta velocidad el robot es capaz de arrancar, pero no se pueden controlar los desvíos de su trayectoria.■ El robot no es capaz de realizar trayectorias rectas (debido a los motores).■ El robot responde al comando enviado por el usuario■ El robot responde al comando enviado por el usuario.
Código de prueba	Anexo B.3

Tabla 4.5: Caso de prueba 3

4.5. Resultados

Las pruebas realizadas mostraron que las ruedas seleccionadas mejoran la tracción del vehículo.

Se detectó que no es posible controlar el motor en rangos pequeños de velocidad debido al alto coeficiente de rozamiento de la caja de engranajes.

Se obtuvo un nuevo chasis de piezas encastrables que no requieren tornillos separadores. Se logró acercar el centro de gravedad al piso haciendo el sistema más estable. Se incorporaron los motores en el cuerpo del chasis, lo que trae como beneficio que las partes móviles de los motores se ensucien menos.

Se detectó que luego de 5 horas de uso del robot los bujes que adaptan las ruedas a los ejes del motor se deterioran.

La biblioteca desarrollada simplifica la configuración y el control de los motores.

No fue posible medir la velocidad de los motores en RPM con la precisión necesaria para implementar un sistema de control.

4.6. Conclusiones

Se debe incorporar un nuevo actuador que proporcione mayor torque para vencer el rozamiento estático. Para ello se modifica el plan de trabajo incorporando una nueva iteración.

Es necesario rediseñar y seleccionar otro material para la construcción de los bujes.

El método de medición de velocidad desarrollado no es eficiente, de modo que no se pudo implementar un sistema de control para el robot. La solución de este problema excede los límites de tiempo de este proyecto, por lo que se deriva el desarrollo de un nuevo hardware a otro equipo de trabajo.

En la próxima iteración se estudiarán las características más relevantes de los motores y se realizarán las pruebas correspondientes sobre los mismos para seleccionar el apropiado. Además se realizará un análisis de materiales para la mecanización y optimización de funcionamiento de los bujes necesarios en esta aplicación.

CAPÍTULO 5

ITERACIÓN 3

5.1. Introducción

Los motores seleccionados en la iteración 2 no tienen la fuerza suficiente para romper la inercia a bajas revoluciones, lo que dificulta el control del robot. Por este motivo se profundiza el estudio sobre las características más importantes de los mismos.

Se seleccionan nuevos motores y se rediseñan las piezas necesarias, como es el caso de las bridas y los bujes.

5.2. Requerimientos

De la iteración 2 derivan los siguientes requerimientos:

- Seleccionar un motor que tenga las siguientes características:
 - Revoluciones: menor a 200 RPM
 - Tensión de alimentación máxima: 12V
 - Torque: Fuerza necesaria para iniciar el movimiento del robot

5.3. Desarrollo

5.3.1. Análisis de motores

Los motores de corriente continua incorporados en el modelo actual de Hermes II, no son los indicados para realizar los movimientos del robot. Para transformar la energía motriz en mayor fuerza reduciendo la velocidad transmitida a las ruedas del robot se necesita usar una caja reductora.

La elección de la caja reductora es el aspecto más importante ya que, si es demasiado grande hará que el robot se mueva muy lentamente y si es demasiado pequeña no tendrá la fuerza suficiente para moverlo. Otro aspecto a tener en cuenta es el peso del robot, ya que de esto dependen las características anteriormente mencionadas.

5.3.1.1. Mediciones

Una de las características fundamentales para la selección del motor es el torque. En base al peso del robot se mide la fuerza máxima que debe aportar cada motor. Se analiza el caso en que el robot avanza hacia uno de sus laterales, caso en que solo actúan dos de los cuatro motores (máximo esfuerzo)

Para medir el torque necesario que debe aportar cada motor, se fijan las ruedas de un eje de tal forma que no giren, por ejemplo las ruedas 1 y 3 como en el caso planteado en el gráfico 5.1. De esta manera, se puede medir la fuerza que debe tener cada motor para vencer el rozamiento entre las ruedas y el plano donde el robot se desplaza, fuerza necesaria para sacar al robot del estado de reposo.

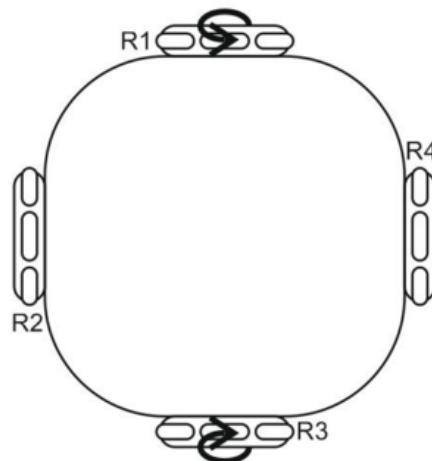


Figura 5.1: Prototipo de prueba

Luego se dispone el robot sobre un plano y se ejerce sobre él una fuerza \vec{F} de tracción hasta el instante en que éste comienza a desplazarse. Esta fuerza se mide con una celda de carga a una frecuencia determinada y se registran los datos con un datalogger. Esta prueba se ilustra en la figura 5.2 .

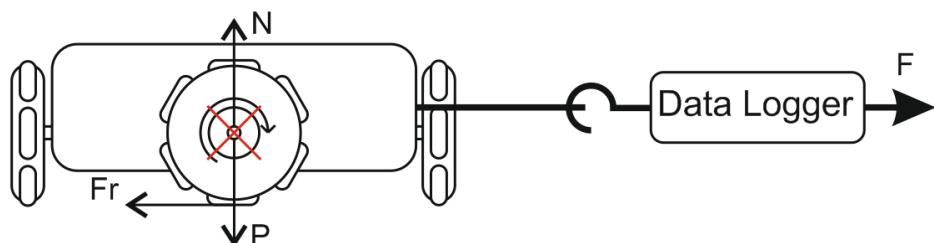


Figura 5.2: Prototipo de prueba

Condiciones iniciales:

- El peso del robot es 0,965Kg

- Ruedas R1 y R3 fijas.
- Plano de desplazamiento de madera
- Velocidad inicial del robot nula
- Masa del robot distribuida uniformemente

Un vez obtenido los datos, se realiza la gráfica 5.3, donde se puede ver que la máxima fuerza es de 4N, a los 9,3 segundos, momento en que el robot comienza a moverse.

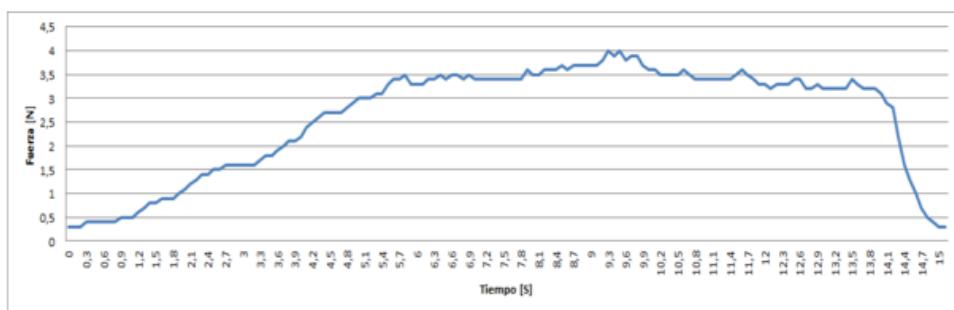


Figura 5.3: Fuerza de rozamiento estático

La fuerza de rozamiento medida, (4N) es la fuerza que se opone al movimiento de las dos ruedas fijas, por lo que sobre cada una de ellas actúa una fuerza de 2N. Sabiendo que la rueda tiene un radio de 2,5 cm, se puede calcular el torque mínimo necesario que debe tener el motor de la siguiente manera:

$$\begin{aligned} \tau &= F * r \\ \tau &= 2N * 2,5cm \\ \tau &= 5cmN \end{aligned} \tag{5.1}$$

Por lo tanto, el torque mínimo necesario por motor para iniciar el movimiento lateral del robot, es de 5cmN. Se debe tener en cuenta que si el torque de los motores es muy elevado, las ruedas perderán tracción debido a que se supera el coeficiente de rozamiento estático, produciéndose en consecuencia el deslizamiento en lugar de la rodadura (la rueda patina, y prevalece el coeficiente de rozamiento cinético que es menor al estático).

5.3.2. Selección de motor

Teniendo en cuenta las mediciones realizadas, se selecciona el motor de la imagen 5.4 que posee las características indicadas en la tabla 5.1 .



Figura 5.4: Nuevo motor

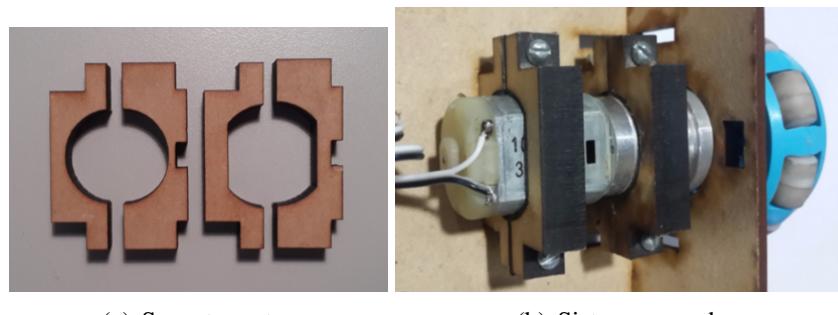
Tensión [V]	Velocidad [RPM]	Torque [Kg*cm]	Torque [N*cm]
3V	80RPM	0,3[Kg*cm]	2,94[N*cm]
6V	200RPM	1[Kg*cm]	9.8[N*cm]
12V	400RPM	2[Kg*cm]	19,6[N*cm]

Tabla 5.1: Características del motor seleccionado

5.3.3. Diseño de nuevas bridas

Debido a que se cambiaron los motores fue necesario realizar nuevas bridas que se adapten a la forma de los mismos. En el modelo anterior el motor se había sujetado con dos piezas rígidas que dificultan la colocación y extracción del mismo, por este motivo se aprovechó la oportunidad para hacer un nuevo diseño que tuviera en cuenta estas consideraciones. El nuevo diseño que se muestra en la imagen 5.5, fue diseñado en dos piezas. La pieza inferior queda fija al chasis y la superior se ajusta a la inferior usando tornillos, sujetando al motor entre ambas piezas.

Se recubrió el interior de las piezas con goma con el fin de reducir las vibraciones producidas por el motor.



(a) Soporte motor

(b) Sistema armado

Figura 5.5: Nuevo soporte de motor

5.3.4. Rediseño de bujes

En la iteración 2 se detectó que luego del uso prolongado del robot, los bujes de madera MDF presentan fallas por deformaciones permanentes e inadmisibles. Se busca un material alternativo

que permita una presión específica mayor sin pérdida de forma que conduzca a una falla por deformación y/o desgaste.

Por este motivo se busca un material que sea más resistente que el mdf y a su vez fácil de trabajar.

Algunos de los materiales que pueden ser usados son: duraluminio, teflón y polioximetileno (POM).

El duraluminio es un material fácil de maquinar, presenta resistencia a distintos agentes químicos y no necesita de mantenimiento. Su costo elevado.

El teflón es muy usado para realizar bujes y los mismos pueden ser mecanizados en un torno. Es un material costoso y no se dispone de muchos proveedores.

El POM, Polioximetileno o Polióxido de metileno, es un termoplástico cristalino de alta rigidez. Es muy usado para la fabricación de pequeñas piezas. Este material puede ser maquinado con un torno, permitiendo obtener precisión de décimas en su acabado. Es liviano, económico y de alta disponibilidad en el mercado.

Debido a que las cargas sobre el buje son muy bajas, es irrelevante un análisis en mayor profundidad de la elección del material.

Se decide realizar los bujes con POM por sus características ya mencionadas.

En la figura 5.6 puede observarse el buje diseñado.



Figura 5.6: Buje de POM

5.4. Pruebas

5.4.1. Medición de velocidad[RPM]

Se analiza si la velocidad de los motores se mantiene constante con un valor fijo de tensión. Alimentando el motor con un PWM equivalente al 39 % ($PWM=100$, $V_{cc}=4,7V$), se miden los siguientes valores de RPM (Figura 5.7)

Se considera que la velocidad se mantiene constante, y las oscilaciones que se observan en el gráfico 5.7 derivan de la incertidumbre del método de medición usado desarrollado en la iteración 4.

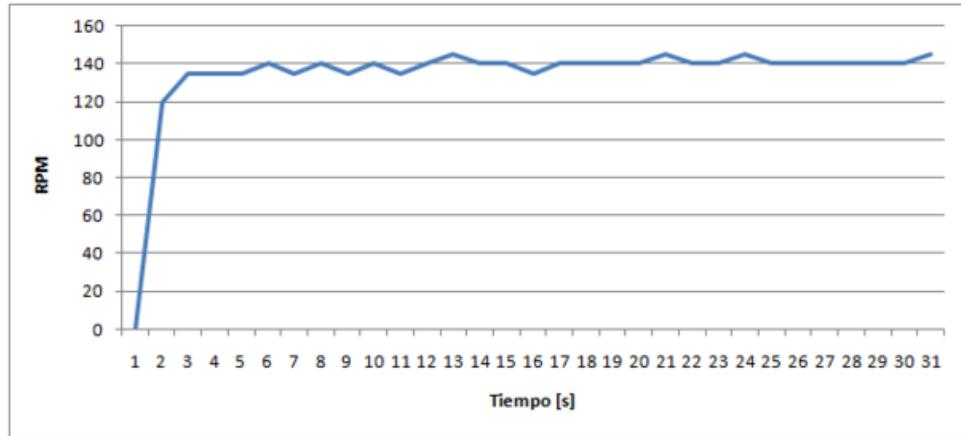


Figura 5.7: Velocidad del motor

5.4.2. Medición del torque del motor

Teniendo en cuenta los datos obtenidos y el motor seleccionado en el punto 5.3.2, se realizan una serie de mediciones sobre dicho motor.

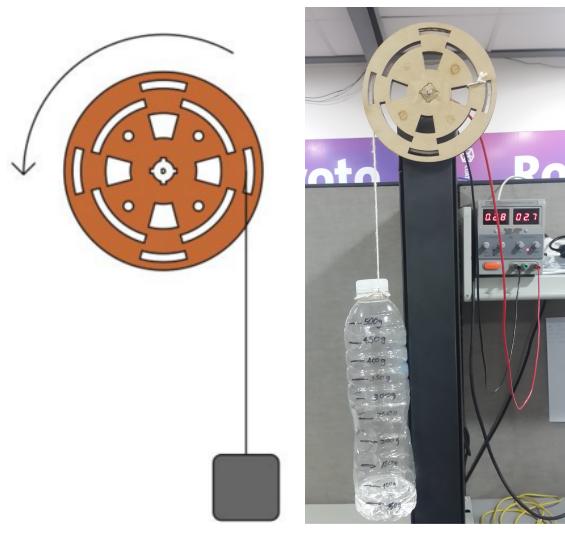


Figura 5.8: Sistema de medición de torque.

Se busca medir el torque estático del motor para distintos niveles de tensión de alimentación. Para ello se monta el escenario de la figura 5.8 en el que se conecta el motor a una polea. Sobre la polea se enrolla un hilo, al cual se le coloca en su extremo un soporte en el cual se irá agregando peso.

Considerando como tensión mínima 3V, se comienzan las mediciones con dicho valor. Se coloca un peso de 50g (40ml de agua más 10g de la botella) y se alimenta el sistema. Si el motor es capaz de elevar esa masa, se aumenta el peso de a 10g (10ml de agua medidos con una jeringa) y se repite este procedimiento hasta que el motor no pueda vencer la inercia inicial, teniendo en cuenta que la fricción inicial es muy baja ésta es despreciada. Llegado a este punto se registran el nivel de corriente, la tensión de alimentación y el último peso que fue capaz de elevar el motor para confeccionar la tabla 5.2.

Se incrementa el valor de tensión en 1V y se repite el procedimiento mencionado hasta alcanzar un valor de tensión de 12V.

$$F = P * g[N] \\ T = F * d[cmN]$$
 (5.2)

Siendo

- F = fuerza
- T = torque
- P = peso
- g = aceleración de la gravedad ($9,8m/s^2$)
- d = radio de la polea (5cm)

En la tabla 5.2 pueden observarse el torque generado por el motor para los distintos niveles de tensión.

Tensión [V]	Corriente [A]	Potencia [W]	Torque [cmN]
3	0,25	0,75	2,45
4	0,4	1,6	4,9
5	0,53	2,65	6,86
6	0,68	4,08	9,80
7	0,8	5,6	11,76
8	0,97	7,76	13,72
9	1	9	14,70
10	1,1	11	16,17
11	1,25	13,75	18,37
12	1,4	16,8	20,82

Tabla 5.2: Torque del motor

Como se puede observar en el gráfico 5.9, para una tensión entre 6V y 7V, se obtiene un torque estático entre 9,8 cmN y 11,76cmN, satisfaciendo el requerimiento de torque de arranque medido en el punto 5.3.1.1 .

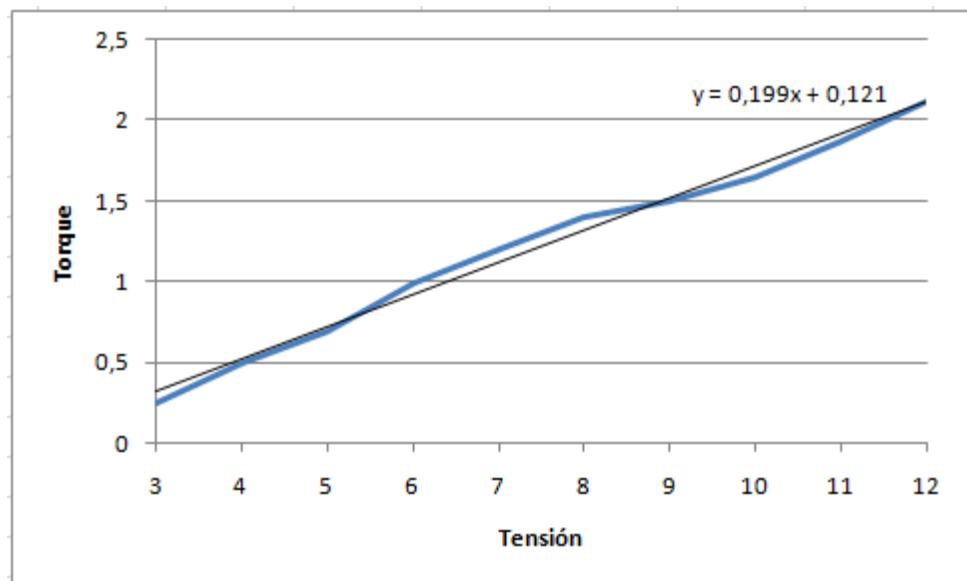


Figura 5.9: Torque del motor

Para poder validar los datos de torque dinámico del motor proporcionados por el proveedor, se debe montar un banco de prueba, dejando esta tarea como trabajo futuro.

5.4.3. Prueba de motores incorporados en el robot

Luego de armar el robot con los nuevos motores se realiza una prueba de movimiento lateral, donde el esfuerzo de los motores es el máximo.

Se alimentan los motores con un valor de PWM=70, 27 % de Vcc, siendo Vcc=12V. El robot no es capaz de iniciar su movimiento, por lo que se aumenta el valor de PWM progresivamente hasta lograr el objetivo.

Se corrobora que el robot es capaz de iniciar su movimiento con un PWM = 110, equivalente al 43 % de Vcc (5,1V).

De la ecuación 5.1 se llega a la conclusión que el torque estático necesario por motor es de 5N.cm. Del gráfico 5.9 se obtiene que para una tensión de alimentación de 5,1V el torque resultante es 6,9N.cm cumpliendo con el requerimiento de torque necesario.

5.4.4. Desvío de trayectoria

Caso de prueba 4	Error de trayectoria
Objetivo	Verificar que el robot sea capaz de realizar trayectorias en línea recta, con un error máximo de 2cm de desvío en 1m de avance.
Prerrequisitos	<ul style="list-style-type: none">■ El sistema debe estar alimentado.■ Debe estar corriendo la aplicación en la placa de desarrollo.
Pasos	<ul style="list-style-type: none">■ Establecer la comunicación entre el robot y el dispositivo que lo controlará.■ Enviar un comando.
Resultado esperado	<ul style="list-style-type: none">■ Que el robot sea capaz de desplazarse en línea recta con la dirección y sentido establecida por el comando, cometiendo un error máximo de 2cm de desvío en 1m recorrido.
Resultado obtenido	<ul style="list-style-type: none">■ El robot puede desplazarse en líneas rectas.
Código de prueba	Anexo B.3

Tabla 5.3: Caso de prueba 4

5.5. Resultados

Se logra constatar que los motores seleccionados poseen el torque necesario para el funcionamiento del robot.

El robot es capaz de desplazarse en trayectorias rectas.

Las bridas diseñadas permiten colocar y extraer el motor facilitando el ensamblaje del robot.

5.6. Conclusiones

Al seleccionar un motor para el desarrollo de un proyecto es necesario realizar pruebas sobre el sistema que lo va a incorporar. De esta manera se plantean los requerimientos que debe cumplir.

Habiendo efectuado los ensayos y pruebas pertinentes, y teniendo el sistema mecánico funcionando correctamente, se procede a la configuración y puesta a punto de la placa de desarrollo principal.

CAPÍTULO 6

ITERACIÓN 4

6.1. Introducción

En esta iteración se trabaja sobre la placa de desarrollo Raspberry Pi B+ seleccionada en la iteración 1. Se escoge y embebe un sistema operativo con el fin de que el robot pueda ser programado en los lenguajes de programación planteados en los requerimientos de software del proyecto.

Se configuran los parámetros para hacer uso de los distintos protocolos de comunicación de los que dispone Raspberry Pi y finalmente se implementa la comunicación I2C con los actuadores.

6.2. Requerimientos

- La placa de desarrollo principal debe tener un sistema operativo configurado para la programación en lenguajes C, C++, Python y Java.
- La placa de desarrollo principal debe recibir los comandos del usuario mediante Bluetooth.
- La placa de desarrollo principal debe procesar los comandos recibidos
- Se debe establecer una comunicación I2C entre Raspberry Pi B+ y Arduino.

6.3. Desarrollo

6.3.1. Incorporación de Raspberry Pi B+

Actualmente en Raspberry Pi puede ejecutar sistemas operativos Linux que soporte procesadores ARM.

Algunos de los sistemas operativos disponibles para Raspberry Pi que detalla Hypertextual [15] son:

-
- RISC OS: Es un sistema operativo desarrollado por Acorn Computers para computadoras basadas en chips ARM. Incluye algunas aplicaciones básicas, y se puede pagar para aumentar a una versión completa.
 - RASPBMC: Es un sistema operativo orientado al uso de Raspberry Pi como media center. Se apoya sobre Debian y XBMC y es una opción si se desea conectar a un televisor un dispositivo para visualizar contenidos multimedia.
 - OPENELEC: Es un sistema operativo orientado al uso de Raspberry Pi como media center. Se apoya sobre Debian y XBMC y es una opción si se desea conectar a un televisor un dispositivo para visualizar contenidos multimedia.
 - PIDORA: Es un sistema operativo orientado al uso de Raspberry Pi como media center. Se apoya sobre Debian y XBMC y es una opción si se desea conectar a un televisor un dispositivo para visualizar contenidos multimedia.
 - RASPBIAN: Es un sistema operativo basado en Debian que ofrece un entorno tanto en modo consola como en modo escritorio. Actualmente es el más conocido y estable para Raspberry Pi.

Para el desarrollo del presente proyecto se decide utilizar el Sistema Operativo Raspbian por ser el más versátil, el más optimizado para Raspberry y el que más documentación posee, sin dejar de mencionar que es el recomendado por el fabricante.

6.3.1.1. Instalación de sistema operativo

Se debe tener en cuenta que, para almacenar el sistema operativo y todos los programas se requiere una tarjeta de memoria microSD de al menos 8GB de capacidad, ya que Raspberry Pi no dispone de Memoria interna.

Para la instalación del sistema operativo Raspbian se procede de la siguiente manera:

1. Descargar la última versión de Raspbian que se encuentra disponible en la página oficial de Raspberry Pi [16]
2. Descomprimir la imagen descargada del fichero.
3. Insertar la tarjeta de memoria microSD en la computadora.
4. La imagen del sistema operativo puede ser cargada en la memoria de dos formas:

- Por consola: Determinar la localización del dispositivo. Para ello ejecutamos el comando

```
user@dist:~$ df -h
```

que muestra una lista de los dispositivos de almacenamiento actualmente conectados en el PC. En este caso se llama sdc y tiene dos particiones sdc1 y sdc2. El siguiente paso es desmontar la tarjeta SD del sistema de ficheros. Hay que desmontar tantos dispositivos como particiones tenga la tarjeta:

```
umount /dev/sdc1  
umount /dev/sdc1
```

El paso final es cargar la imagen de Raspbian en la tarjeta SD:

```
sudo dd bs=1M if=2014-06-20-wheezy-raspbian.img of=/dev/sdc
```

- Usando la aplicación Win32DiskImager: Se ejecuta el programa y luego se inserta la tarjeta de memoria microSD en la computadora. Se selecciona la imagen de Raspian luego de haber sido descomprimida y se procede al grabado de la misma.
5. Se debe hacer un update para sincronizar la base de datos de los paquetes de software y las versiones disponibles.
 6. Realizar un upgrade que hará que los paquetes con versiones más recientes disponibles se actualicen. [16]

6.3.1.2. Instalación de proyecto PI4J

El robot será utilizado principalmente para la realización de trabajos prácticos, de modo que uno de los requerimientos principales es la programación en Java.

Al instalar el sistema operativo se instala por defecto una máquina virtual de Java. Para ver la versión actualmente instalada se ejecuta el comando

```
java -version
```

Para acceder a los puertos de comunicación GPIO de Raspberry Pi se hará uso de Pi4J, proyecto que provee un puente entre librerías nativas y Java.

Para su instalación se debe ejecutar el comando

```
curl -s get.pi4j.com | sudo bash
```

Este comando descarga y lanza un script que realizará los siguientes pasos:

- agregar el repositorio Pi4J APT a los repositorios locales APT.
- descargar e instalar la validación de la firma de clave pública de Pi4J GPG.
- invocar el comando ‘apt-get update’ en el repositorio Pi4J APT para actualizar la base de datos de paquetes locales.
- invocar el comando ‘apt-get install pi4j’ para descargar y actualizar.

Se puede hacer un update y un upgrade cuando se necesite usando los siguientes comandos respectivamente:

```
sudo apt-get update  
sudo apt-get upgrade
```

En caso que se desee desinstalar Pi4J se ejecuta el comando

```
sudo apt-get remove pi4j o pi4j --uninstal
```

Para completar el paso anterior, y eliminar también el repositorio Pi4J de la lista de repositorios APT y la firma Pi4J GPG se ejecuta el comando

```
curl -s get.i4j.com/uninstal | sudo bash
```

Para mayor información, acceder a la página oficial de Pi4J [17]

6.3.1.3. Configuración para comunicación Bluetooth

Para hacer uso de la comunicación entre Raspberry otros dispositivos a través de un módulo Bluetooth es necesario habilitar y configurar el puerto serial.

Por defecto Raspberry Pi se encuentra configurada para escribir un mensaje de arranque al puerto serial con un baud rate de 115200 bps, mientras que el módulo Bluetooth HC-05, con el que se decidió trabajar, viene preconfigurado de fábrica a 9600 bps.

Se configura el baud rate de Raspberry Pi con un valor de 9600 bps. Para ello se modifican dos archivos de configuración.

Primero se modifica el archivo /boot/cmdline.txt que contiene opciones del kernel que son usados por el boot del sistema.

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200  
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline  
rootwait
```

Las opciones que interesan son console y kgdboc, ya que éstas son las que configuran el puerto serial /dev/ttyAMA0 a 115200 bps. Se reemplaza el baud rate por 9600 bps. Quedando la configuración de la siguiente manera:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,9600 kgdboc=ttyAMA0,9600  
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline  
rootwait
```

La segunda modificación que se realiza es en el archivo de configuración /etc/inittab. En el mismo se busca la línea.

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

La misma le indica al sistema que inicie una terminal en el puerto serial utilizando un baud rate de 115200bps, la cual se modifica por 9600bps.

```
T0:23:respawn:/sbin/getty -L ttyAMA0 9600 vt100
```

Luego de realizar dichos cambios es necesario reiniciar el sistema para que los cambios se hagan efectivos.

Para hacer usar el puerto serial desde una aplicación ejecutándose en Raspberry Pi, es necesario que el sistema no lo use como consola. Para esto se modifica los cambios realizados en las últimas configuraciones. Se recomienda hacer una copia de los archivos en caso de que alguno de los cambios no funcione.

En el archivo /boot/cmdline.txt se deben remover las dos referencias seriales eliminando las secciones que hacen referencia a console y kgdbo.

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

Finalmente en el archivo /etc/inittab se comenta la siguiente línea que hace referencia al paquete de consola serial.

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 9600 vt100
```

Al realizar estos cambios se reinicia el sistema. En consecuencia Raspberry Pi no utilizará el puerto serial para ninguna de sus tareas, quedando éste disponible para ser utilizado por cualquier aplicación como /dev/ttyAMA0.

6.3.1.4. Conexión de módulo Bluetooth

RBPi GPIO pin	Bluetooth module pin
5v (Pin2)	Vcc
GND (Pin6)	GND
TXD (Pin8)	RDX
RDX (Pin10)	TDX

Tabla 6.1: Pines de conexión entre Raspberry Pi B+ y módulo Bluetooth HC-05

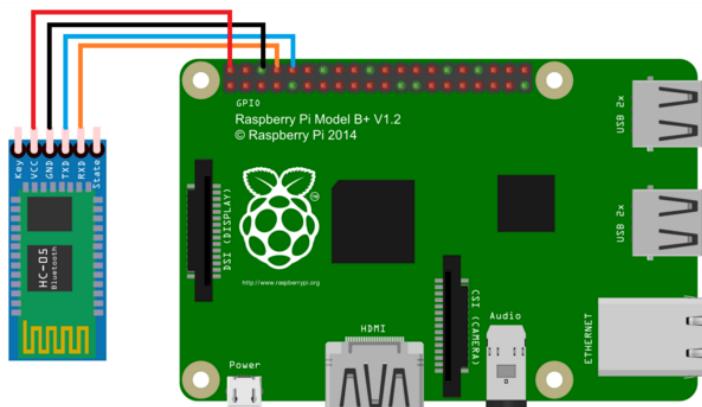


Figura 6.1: Conexión de módulo Bluetooth a Raspberry Pi B+

6.3.1.5. Configuración para comunicación I2C

El bus I2C permite la conexión de múltiples dispositivos, cada uno con una dirección única.

I2C es un protocolo utilizado para comunicaciones a corta distancia, generalmente para comunicar microcontroladores. Consta de tres líneas de conexión: datos, reloj y masa.

Se deben realizar configuraciones en el sistema operativo Raspbian, para poder utilizar Raspberry Pi B+ como dispositivo maestro y poder transmitir y recibir datos hacia y desde dispositivos esclavos.

El bus I2C se encuentra desactivado por defecto en el sistema operativo Raspbian. Para activarlo se debe modificar el archivo blacklist. Para ello se ejecuta un programa de edición de texto como:

```
>> sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Se debe comentar la siguiente línea:

```
blacklist i2c-bdm2708
```

Al comentar la línea se saca el protocolo I2C de la blacklist. Esto genera que el kernel lo cargue automáticamente al inicio del sistema operativo.

Luego se debe agregar el módulo I2C al kernel.

Se ejecuta:

```
>> sudo nano /etc/modules
```

y al final del archivo se agrega la siguiente línea:

```
i2c-dev
```

Como último paso se debe instalar el módulo python-smbus y los paquetes i2c-tools y git-core.

```
>> sudo apt-get install python-smbus i2c-tools git-core
```

Para que los cambios se hagan efectivos se debe reiniciar el sistema.

RBPi GPIO pin	Arduino proMini module pin
5v (Pin4)	Vcc
GND (Pin6)	GND
SCL (Pin5)	SCL (PinA5)
SDA (Pin3)	SDA (PinA4)

Tabla 6.2: Pines de conexión entre Raspberry Pi B+ y Arduino ProMini

6.3.1.6. Conexión I2C entre Raspberry Pi y actuadores

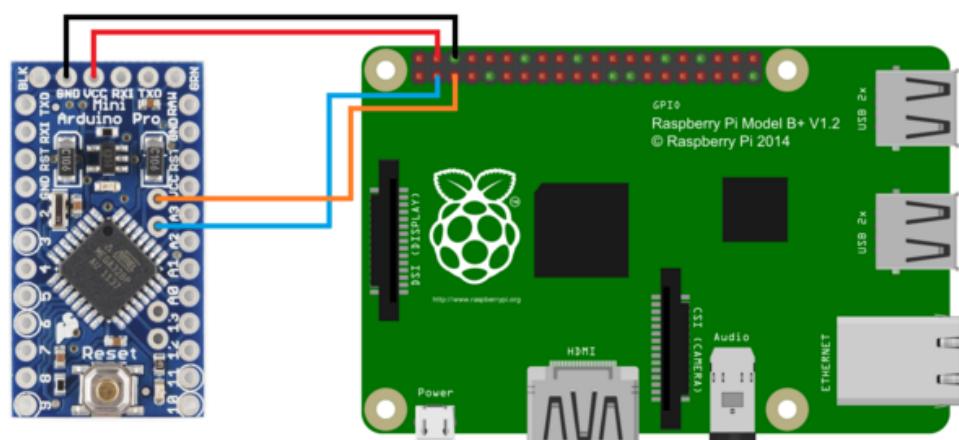


Figura 6.2: Conexión entre microcontroladores por protocolo I2C

6.4. Pruebas

6.4.1. Comunicación Bluetooth en Raspberry Pi

Caso de prueba 5	Comunicación Bluetooth
Objetivo	Verificar el correcto funcionamiento de la comunicación Bluetooth en Raspberry.
Prerrequisitos	<ul style="list-style-type: none">■ El sistema debe estar alimentado.■ Debe estar corriendo la aplicación en la placa de desarrollo.■ Debe estar conectado el módulo Bluetooth a la placa de desarrollo.■ Se debe haber configurado la comunicación serial.
Pasos	<ul style="list-style-type: none">■ Establecer la comunicación entre el robot y el dispositivo que lo controlará.■ Enviar un comando.
Resultado esperado	<ul style="list-style-type: none">■ Que el microcontrolador reciba el comando enviado por el usuario.■ Que el microcontrolador responda al usuario.
Resultado obtenido	<ul style="list-style-type: none">■ El microcontrolador recibe el comando.■ El microcontrolador envía un mensaje de respuesta al usuario.
Código de prueba	Anexo B.4

Tabla 6.3: Caso de prueba 5

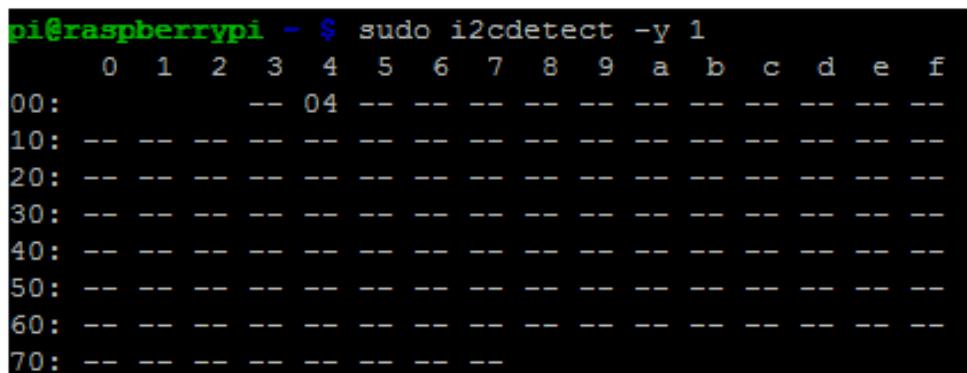
6.4.2. Detección de dispositivos conectados al bus I2C

Esta prueba se realiza conectado a Raspberry Pi la placa de desarrollo Arduino.

Luego de realizar la conexión como se indicó en la figura 6.2, se debe ejecutar el comando:

```
>> sudo i2cdetect -y 1
```

Si se realizó la configuración y la conexión correctamente, se debe encontrar en la dirección 04h la placa de desarrollo Arduino como se observa en la imagen 6.3, ya que ésa es la establecida por el fabricante.



```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- 04 -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- --
```

Figura 6.3: Prueba de conexión bajo protocolo I2C

Si la placa Arduino no es identificada y aparece el siguiente mensaje de error:

```
pi@raspberrypi ~ \$ sudo i2cdetect -y 1
Error: Could not open file '/dev/i2c-1' or '/dev/i2c/1': No such file or
directory
```

Se debe agregar al directorio /boot/config.txt la siguiente línea:

```
dtparam=i2c1=on
```

6.4.3. Comunicación I2C entre Raspberry Pi y Arduino

Caso de prueba 6	Comunicación I2C
Objetivo	Verificar el correcto funcionamiento de la comunicación entre los microcontroladores.
Prerrequisitos	<ul style="list-style-type: none"> ■ Los dos microcontroladores deben estar alimentados. ■ En los dos microcontroladores debe estar corriendo la aplicación de envío y recepción de datos bajo el protocolo I2C. ■ Debe estar configurado el microcontrolador como I2C master y el microcontrolador secundario como esclavo. ■ Que el microcontrolador esclavo esté conectado al bus I2C y sea detectado.
Pasos	<ul style="list-style-type: none"> ■ Ejecutar código de prueba sobre la placa de desarrollo principal Raspberry Pi B+, que envíe y reciba mensajes mediante protocolo I2C hacia la placa de desarrollo Arduino de los controladores. ■ Ejecutar un código de prueba en Arduino que reciba y envíe los mensajes por I2C. ■ Mostrar los mensajes recibidos en Arduino por monitor serial. ■ Mostrar los mensajes recibidos en Raspberry Pi B+ a través de monitor serial con conexión ssh.
Resultado esperado	<ul style="list-style-type: none"> ■ Que el microcontrolador principal pueda enviarle datos al esclavo. ■ Que el microcontrolador esclavo pueda recibir los datos enviados. ■ Que el microcontrolador esclavo pueda enviarle datos al principal por el mismo bus. ■ Que el microcontrolador principal pueda leer los datos enviados por el esclavo
Resultado obtenido	<ul style="list-style-type: none"> ■ El microcontrolador principal detecta al microcontrolador esclavo. ■ El microcontrolador principal puede enviar y recibir datos del esclavo. ■ El microcontrolador esclavo puede recibir y enviar datos al principal

Tabla 6.4: Prueba de comunicación I2C

6.5. Resultados

La placa de desarrollo principal Raspberry Pi B+ queda funcionando correctamente con un sistema operativo.

Se dejó configurado el sistema operativo para hacer uso de los protocolos de comunicación Bluetooth e I2C.

6.6. Conclusiones

Luego de cumplir con los requerimientos de mayor riesgo, como son:

- Comunicación inalámbrica entre el usuario y el robot.
- Sistema mecánico holonómico.
- Sistema operativo embebido en el hardware.

se procede a la incorporación de sensores en el sistema y se desarrolla el software necesario para su uso.

CAPÍTULO 7

ITERACIÓN 5

7.1. Introducción

En esta iteración se realiza un rediseño del chasis para incorporar los sensores seleccionados. Se arma un nuevo prototipo con los mismos para ser usados en una iteración posterior. Además se desarrolla el hardware y software correspondiente para el funcionamiento de los sensores. El software se diseña y se desarrolla orientado a componentes para facilitar la incorporación de nuevos módulos al sistema.

7.2. Requerimientos

- Se deben incorporar sensores para reconocimiento del ambiente.
- La programación debe ser orientada a componentes.

7.3. Desarrollo

7.3.1. Rediseño de chasis para sensores

Se rediseña la estructura del robot para la incorporación de sensores y se distribuyen todos los componentes de forma que el centro de gravedad quede en el centro geométrico del robot. Además se incorpora la placa de desarrollo Raspberry Pi, para la cual se diseña un nuevo nivel sobre el chasis de la versión anterior.

En la imagen 7.1 se muestra la vista superior de la base del robot.

7.3.2. Arquitectura de software

Teniendo en cuenta la arquitectura de hardware representada en la imagen 3.1 de la iteración 1, se diseña la arquitectura de software como se muestra en la imagen 7.2, destacando cada uno de sus componentes.

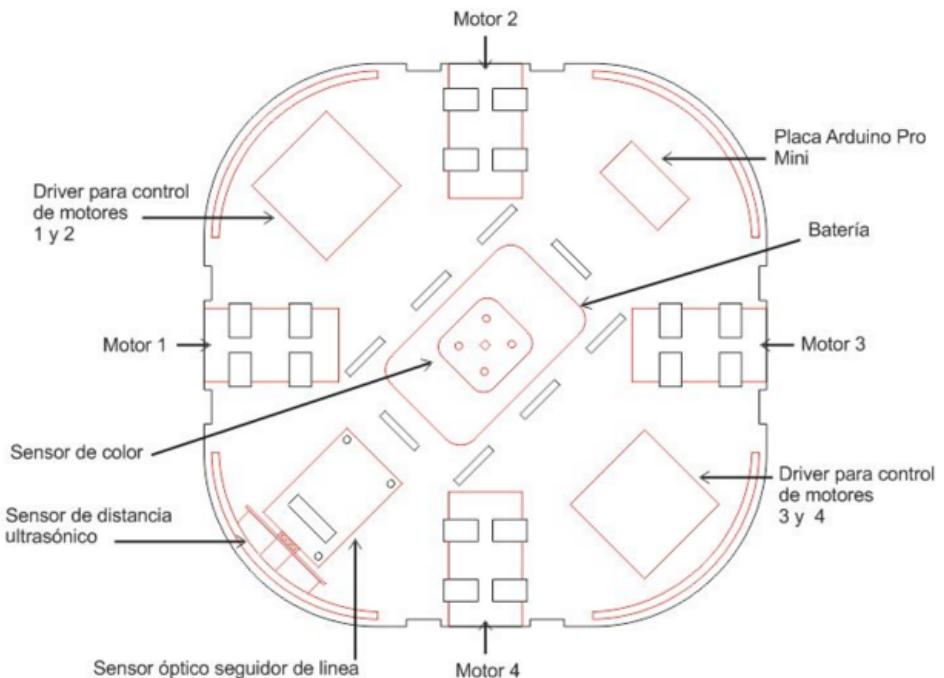


Figura 7.1: Base del robot

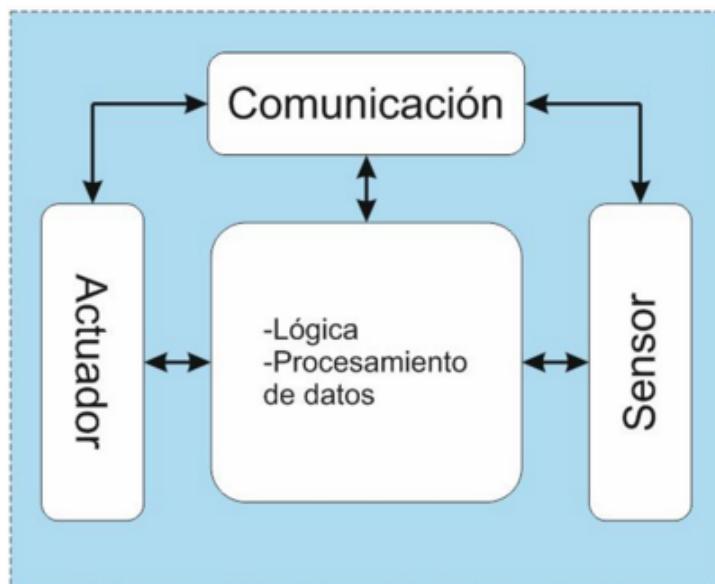


Figura 7.2: Arquitectura de software

7.3.3. Comunicación

Este componente es el encargado de realizar las comunicaciones entre los distintos componentes de hardware. La interfaz diseñada se presenta en la imagen 7.3

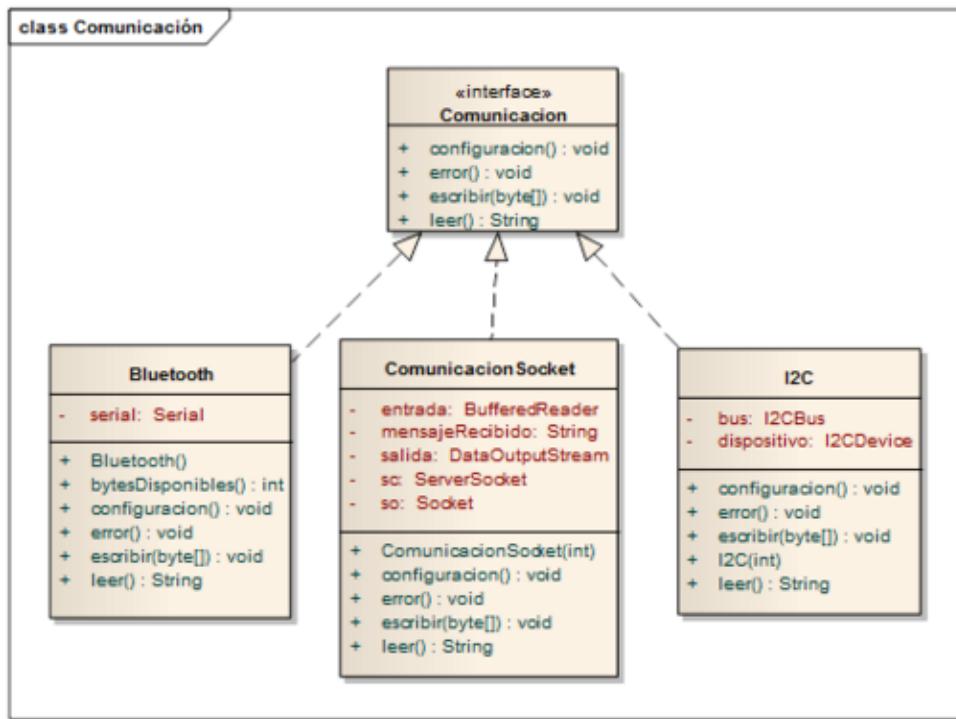


Figura 7.3: Interfaz de comunicación

Métodos de la interfaz:

```

public void escribir(byte[] buffer):
    //Método para enviar un dato .

public char leer():
    //Método para leer un dato .

public void error():
    //Método implementado para tratar los errores que pueden producirse durante
    la comunicación.

public void configuracion():
    //Método para setear los parámetros necesarios para establecer una
    comunicación .
  
```

Los tipos de comunicación implementados son los siguientes:

- I2C: Subclase que establece una comunicación mediante el protocolo I2C.
- Bluetooth: Subclase que establece una comunicación mediante protocolo RS232.
- ComunicacionSocket: Subclase que establece una comunicación entre procesos mediante sockets.

7.3.4. Actuadores

Los actuadores son los componentes encargados de ejecutar acciones en el robot según órdenes especificadas por la unidad de procesamiento o por el usuario.

La interfaz diseñada se puede observar en el diagrama de clases de la figura 7.4

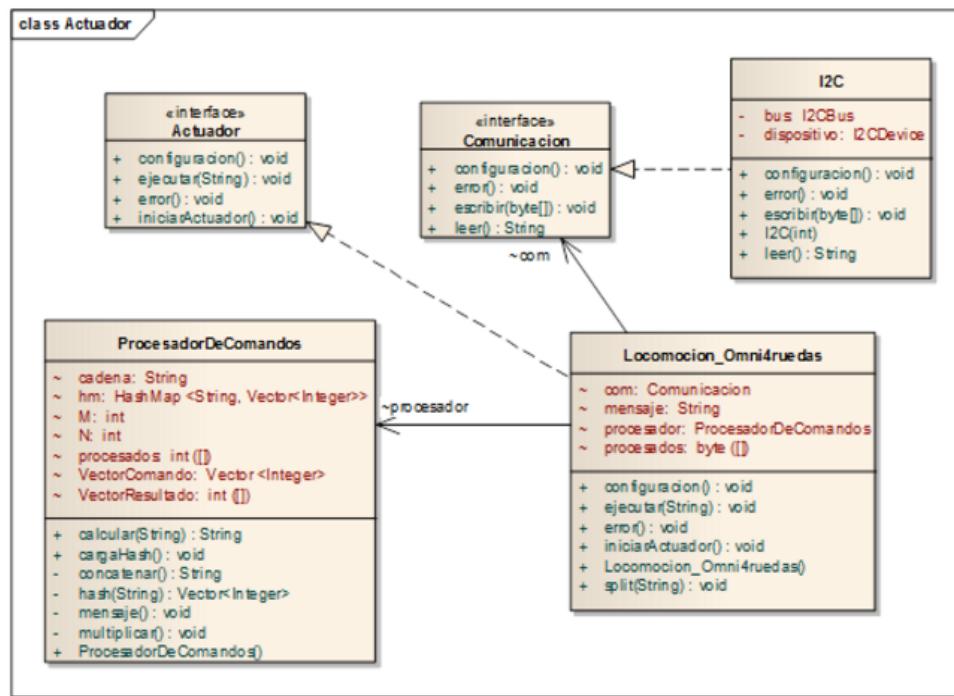


Figura 7.4: Interfaz de actuadores

Métodos de la interfaz:

```
public void error():
//Método implementado para tratar los errores que pueden producirse
durante la ejecución de un actuador.

public void configuracion():
//Método para setear los parámetros necesarios de cada actuador.

public void iniciarActuador():
//Inicializa el actuador para su funcionamiento.

public void ejecutar(String parametro):
// El actuador ejecuta una acción en base a un parámetro que se le indica.
```

7.3.4.1. LocomocionOmni4ruedas

Hermes II cuenta con un solo actuador (“LocomocionOmni4ruedas”) conformado por cuatro motores y una placa de desarrollo con un microcontrolador Atmega328 (Arduino

pro mini). Éste se comunica con la unidad central de proceso (Raspberry pi), a través del protocolo de comunicación I2C.

Para que el robot pueda responder a los movimientos indicados por el usuario, cada uno de los motores debe funcionar de forma independiente como se analiza en la sección 2.4.5 del marco teórico.

A continuación se analiza el envío de datos entre la unidad central de proceso y el actuador. En dicha comunicación se envían mensajes de tipo arreglo de bytes que constan de dos campos como se observa en la figura 7.5. El primer campo indica el tipo de mensaje, que simplifica el proceso de decodificación y en el segundo campo se coloca la información que se necesita transmitir.

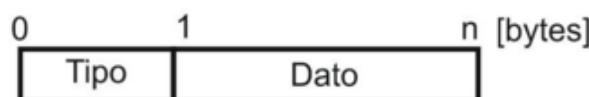


Figura 7.5: Estructura del mensaje I2C

Para el caso del control de los motores de un robot omnidireccional de cuatro ruedas.

Byte [0] = 1: Indica el tipo de mensaje a enviar.

Byte [1]: Representa el sentido de giro de los motores, cada motor con su correspondiente bit, siendo el bit menos significativo el motor 1 y el bit más significativo el motor 4. La convención utilizada para indicar sentido de giro es:

0 ->BACKWARD

1 ->FORWARD

Dado que los datos son interpretados en su recepción como bytes con signo, el valor transmitido no puede ser mayor a 127 ni menor a -128, y dado que los valores de PWM utilizados pueden exceder del límite superior, el valor se distribuirá en dos bytes. Los últimos ocho bytes indican la velocidad de cada motor, siendo la velocidad del motor 1 la suma de los bytes 2 y 3, de la misma forma se calculan los restantes.

Byte[2,3]: sumados forman el valor de PWM correspondiente al motor 1.

Byte[4,5]: sumados forman el valor de PWM correspondiente al motor 2.

Byte[6,7]: sumados forman el valor de PWM correspondiente al motor 3.

Byte[8,9]: sumados forman el valor de PWM correspondiente al motor 4.

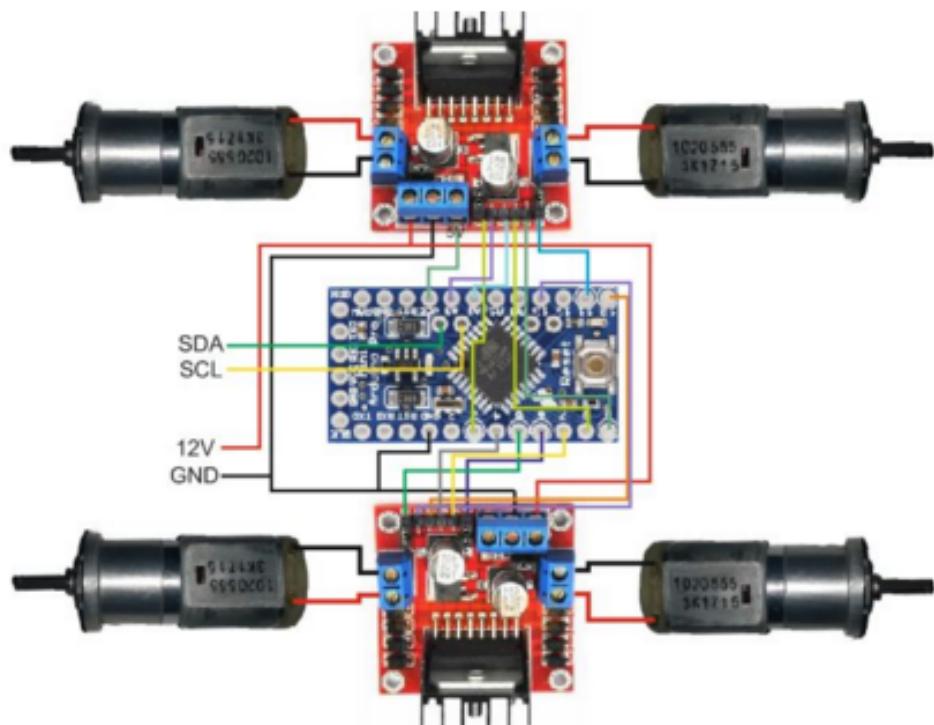


Figura 7.6: Conexión de motores

7.3.5. Sensores

Los sensores son los componentes encargados de recolectar datos del entorno en el que se encuentra el robot y ejecutar alguna acción en base a esta información. La interfaz desarrollada se muestra a continuación en el diagrama de clases de la figura 7.7.

Métodos de la interfaz:

```

public void configuracion():
    //Método para setear los parámetros necesarios de cada sensor.

public void error():
    // Método implementado para tratar los errores que pueden
    producirse durante la ejecución de un sensor.

public void finalizarSensor():
    // Método utilizado para finalizar la ejecución de un sensor.

public String get dato():
    // Método que devuelve un dato del entorno obtenido por
    el sensor.

```

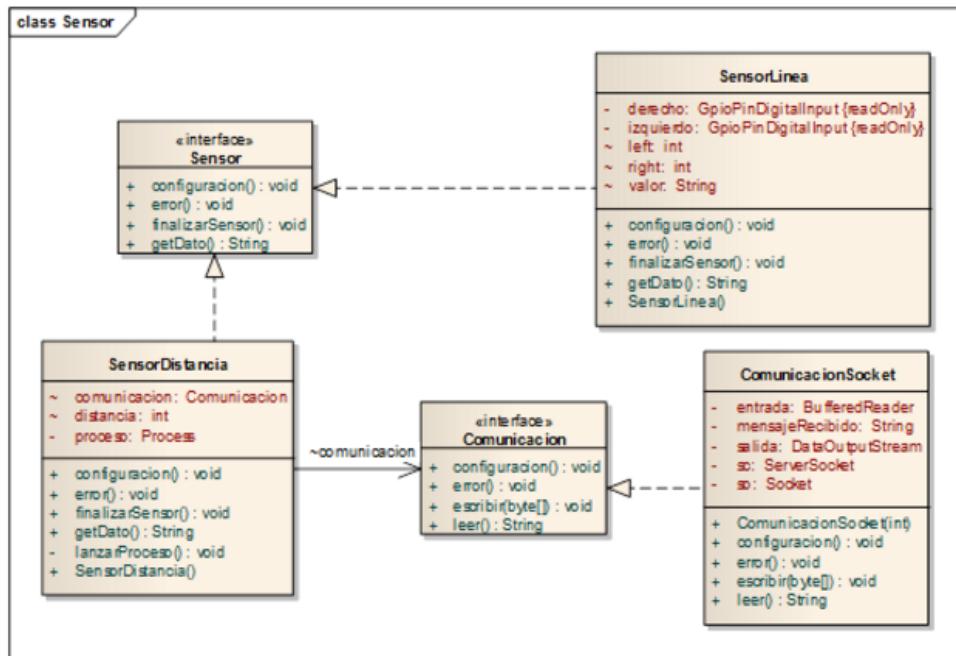


Figura 7.7: Interfaz de sensores

7.3.5.1. Sensor de línea

Se utilizan dos sensores infrarrojos TCRT5000 como el de la imagen 7.8, para diseñar un sensor seguidor de línea.

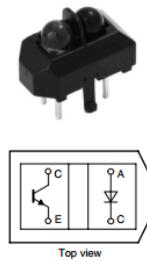


Figura 7.8: Sensor de proximidad

Las características del sensor TCRT5000 se presentan en la tabla 7.1.

El SensorLinea, es el encargado de sensar una línea que conforma un circuito tipo pista, para que el robot pueda desplazarse por dicho circuito de forma autónoma.

Los sensores ópticos se encuentran alineados de forma perpendicular al avance del robot a una distancia entre sí igual al ancho de la línea. Cada uno de los sensores toma datos de la línea, de forma que si el robot se desvía hacia uno de sus laterales, el sensor correspondiente a ese lateral cambia de estado. Los estados de ambos sensores son procesados para tomar decisiones respecto a la trayectoria.

Dimensiones (Largo x Ancho x Alto [mm])	10,2 x 5,8 x 7
Máxima distancia de funcionamiento [mm]	2,5
Corriente de salida típica IC [mA]	1
Longitud de onda del emisor [nm]	950

Tabla 7.1: Características sensor TCRT5000

Se desarrolla el circuito de imagen fig:seguidor, el cual posee dos disparadores smith trigger para digitalizar la señal de cada uno de los sensores TCRT5000.

Se prueba el circuito en protoboard y, posteriormente constatando el correcto funcionamiento, se realiza el diseño de la placa de circuito impreso (Printed Circuit Board, PCB) de la figura 7.10.

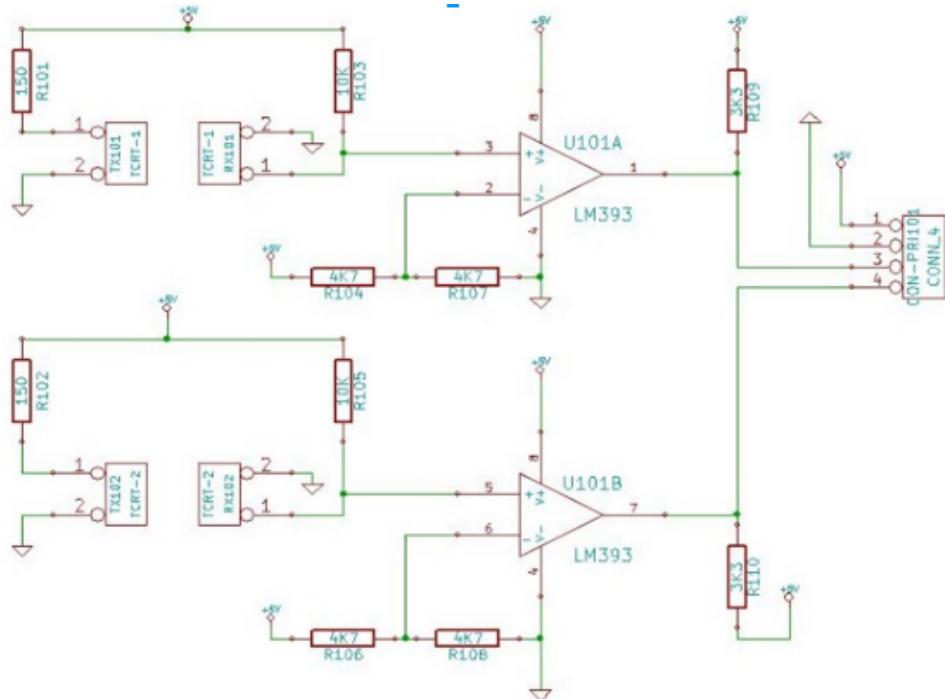


Figura 7.9: Diseño de circuito

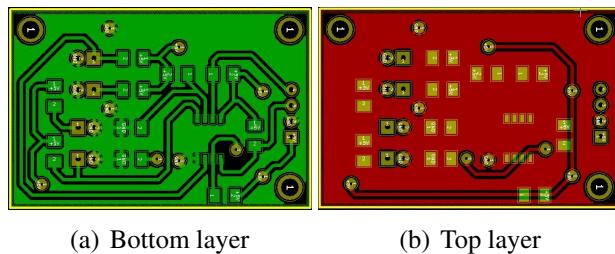


Figura 7.10: Diseño de pcb.

Una vez desarrollado el diseño del PCB, se procede a la producción de la placa y su posterior ensamblado. El sensor de línea ya terminado se muestra en la figura 7.11

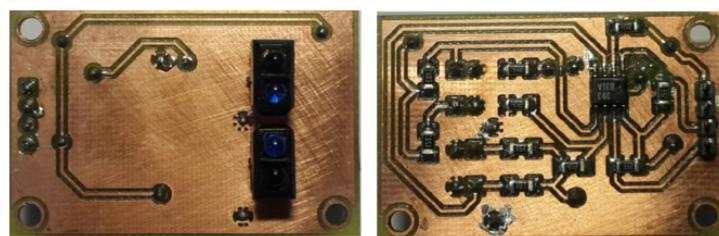


Figura 7.11: Detector de línea

7.3.5.2. Sensor de distancia

Se utiliza un sensor de distancia por sonar HC-SR04 (figura 7.12), el cual consta de un emisor y un receptor de ultrasonido. El principio de funcionamiento se basa en la velocidad del sonido y el tiempo que tarda la señal en recorrer una cierta distancia.



Figura 7.12: Sensor de distancia HC-SR04

Distancia mínima [cm]	2
Distancia máxima [cm]	400
Precisión [mm]	3
Alimentación [V]	5

Tabla 7.2: Características del sensor de distancia HC-SR04

El sensor se conecta a la placa de desarrollo principal como se muestra en la imagen ??.

Al desarrollar el software para este sensor se detecta que el hardware precisa un pulso de disparo de $10\mu s$ para iniciar una medicin.

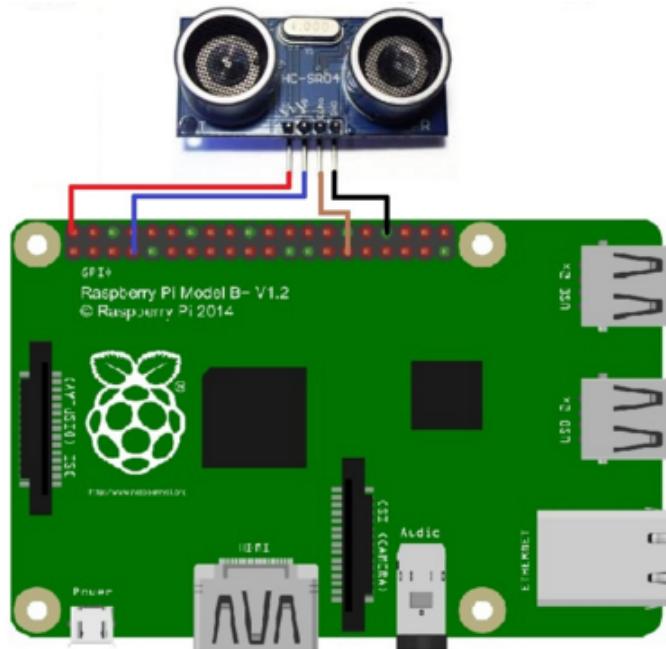


Figura 7.13: Conexión sensor HC-SR04 a Raspberry Pi B+

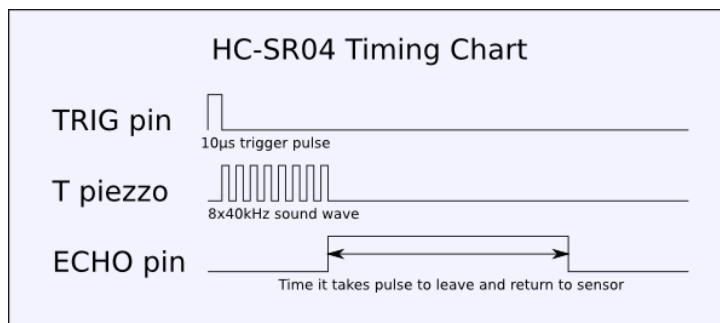


Figura 7.14: Señales sensor HC-SR04 [19]

Al generar esta señal en lenguaje de programación Java, haciendo uso de la biblioteca PI4J, se detecta que no es posible, comprobando empíricamente mediante un osciloscopio que el mínimo pulso posible de lograr es de 10ms.

Se desarrolla posteriormente el software en lenguaje Python y se comprueba que el sensor responde correctamente. Es por esto que se decide realizar el componente SensorDistancia en java, el cual ejecuta un proceso en Python, comunicándose ambos procesos mediante sockets. Una vez inicializado el sensorDistancia, éste devuelve un dato bajo demanda.

7.4. Pruebas

7.4.1. Actuador

Caso de prueba 7	Funcionamiento de actuador LocomocionOmni4ruedas
Objetivo	Verificar el correcto funcionamiento de los motores ante la solicitud de movimiento.
Prerrequisitos	<ul style="list-style-type: none">• El sistema debe estar alimentado.• El microcontrolador Atmega328 debe tener cargado el código de control de motores.• El código de prueba debe estar cargado en la placa de desarrollo principal Raspberry Pi.• Debe estar establecida la comunicación I2C entre la placa de desarrollo Raspberry Pi y el actuador.
Pasos	<ul style="list-style-type: none">• Enviar un comando.
Resultado esperado	<ul style="list-style-type: none">• Que el actuador reciba el dato.• Que el microcontrolador interprete el dato.• Que los motores giren en el sentido especificado por el dato.
Resultado obtenido	<ul style="list-style-type: none">• El actuador recibe el dato.• El actuador interpreta el dato.• Los motores giran correctamente.
Código de prueba	Anexo

Tabla 7.3: Caso de prueba 7

7.4.2. Sensor de línea

Caso de prueba 8	Funcionamiento de sensor de línea
Objetivo	Verificar que el sensor sea capaz de detectar si se encuentra sobre una superficie blanca, negra o en el límite entre ambas.
Prerrequisitos	<ul style="list-style-type: none"> • El sistema debe estar alimentado. • El sensor de línea debe estar correctamente conectado. • El código de prueba debe estar cargado en la placa de desarrollo principal Raspberry Pi.
Pasos	<ul style="list-style-type: none"> • Colocar el sensor sobre una superficie blanca y solicitar el dato. • Colocar el sensor sobre una superficie negra y solicitar el dato.. • Colocar el sensor sobre el borde de una línea, de modo que quede un sensor óptico en un fondo blanco y el otro en un fondo negro y solicitar el dato. • Colocar el sensor sobre el borde de una línea invirtiendo los fondos del punto anterior y solicitar el dato.
Resultado esperado	<ul style="list-style-type: none"> • Que se reciba un valor lógico "11b". • Que se reciba un valor lógico "00b". • Que se reciba un valor lógico "01b". • Que se reciba un valor lógico "10b".
Resultado obtenido	<ul style="list-style-type: none"> • El sensor detecta los cambios de color de la superficie.
Código de prueba	Anexo

Tabla 7.4: Caso de prueba 8

7.4.3. Sensor de distancia

Caso de prueba 9	Funcionamiento de sensor de distancia
Objetivo	Verificar que el sensor sea capaz de medir la distancia a la que se encuentra de un objeto.
Prerrequisitos	<ul style="list-style-type: none">• El sistema debe estar alimentado.• El sensor ultrasónico debe estar correctamente conectado.• El código de prueba debe estar cargado en la placa de desarrollo principal Raspberry Pi.
Pasos	<ul style="list-style-type: none">• Colocar un objeto a distancia mínima de lectura del sensor.• Solicitar la distancia.• Colocar un objeto a distancia máxima de lectura del sensor.• Solicitar la distancia.• Colocar un objeto a distancia intermedia de lectura del sensor.• Solicitar la distancia.
Resultado esperado	<ul style="list-style-type: none">• Que se reciba el valor de distancia a la que se encuentra el objeto.
Resultado obtenido	<ul style="list-style-type: none">• El sensor responde con éxito los datos solicitados.
Código de prueba	Anexo

Tabla 7.5: Caso de prueba 9

7.5. Resultados

Se obtuvo un chasis con el espacio suficiente para la incorporación de los sensores seleccionados. Se desarrolló y ensambló el hardware correspondiente al sensor seguidor de línea. Queda desarrollada una arquitectura de software orientada a componentes.

7.6. Conclusión

La programación de sensores, actuadores y comunicación se diseñó con interfaces para que el usuario tenga la facilidad de incorporar componentes sin realizar cambios significativos sobre las clases desarrolladas.

Todos los componentes del sistema quedan integrados para la realización de casos de uso en la próxima iteración.

CAPÍTULO 8

SISTEMA FINAL

8.1. Introducción

En esta iteración se realiza una integración de todos los componentes del sistema Hermes II y se analizan escenarios que se pueden dar haciendo uso de lo desarrollado a lo largo del proyecto.

8.2. Integración de componentes

A lo largo del proyecto se incorporaron nuevas funcionalidades y se mejoraron aspectos estructurales del robot con el fin de obtener un producto robusto.

En esta iteración se presenta la última versión del robot Hermes II en la cual se integran los diferentes componentes desarrollados.

En el comienzo del proyecto se diseñó la arquitectura de Hardware y en la iteración cinco la arquitectura de Software orientada a componentes. En la figura 8.1 se presenta el diseño de la arquitectura del sistema integrado.

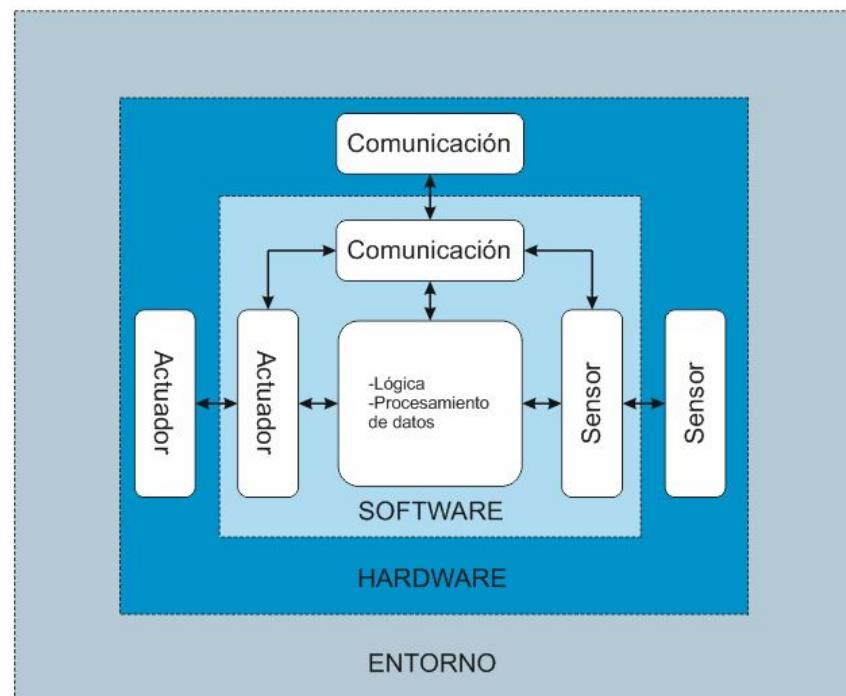


Figura 8.1: Sistema integrado

Se puede observar en la figura 8.2 el diagrama de clases del proyecto.

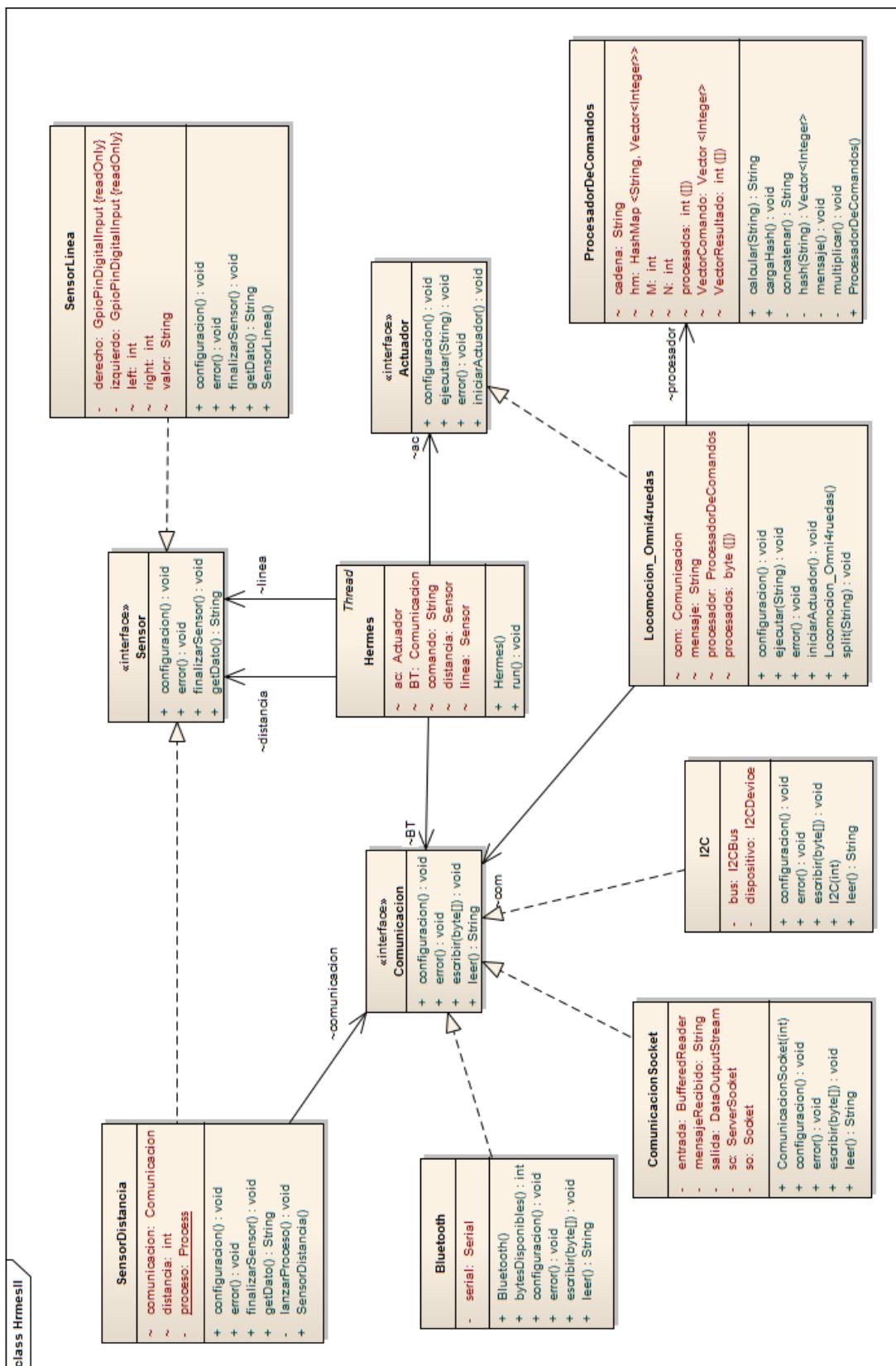


Figura 8.2: Diagrama de clases del sistema final

8.3. Aplicación Android

La aplicación desarrollada para smartphones con sistema operativo Android, fue desarrollada por el Ingeniero Rebolini Leandro. En la imagen 8.3 se muestra la interfaz de la aplicación.

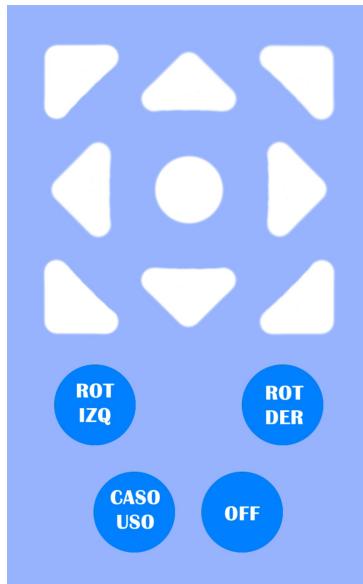


Figura 8.3: Aplicación android

Las aplicación posee las siguientes funcionalidades:

- Menú: Ingresando a la sección menú se linkea el dispositivo al robot. El proceso de desconexión con el robot se logra con solo cerrar la aplicación en el smartphone.
- Apagar: Presionando este botón se ejecuta un comando en el robot que se encarga de cerrar de forma adecuada el sistema operativo Raspbian del mismo.
- Comandos de dirección: Cada comando indicado con una flecha representa un movimiento del robot. Manteniendo presionado cualquiera de ellos, el robot se moverá en dicha dirección y al soltar el botón la misma enviará un comando frenar que lo detendrá.
- Comando Central: Se encuentra representado por un círculo central a los comandos de dirección, y puede darse la utilización que el desarrollador desee. En este momento envía un comando frenar al ser presionado.
- Comandos RotarDerecha y RotarIzquierda: Estos dos comandos hacen girar al robot en sentido horario y antihorario respectivamente al presionarlos, y al soltarlos envían un comando de frenado.
- CU: CU (Caso de Uso) es un comando que puede ser utilizado con la funcionalidad que el desarrollador desee. En nuestro caso, fue utilizado para ejecutar los casos de uso que se presentan más adelante en esta iteración. Al soltar este botón, no se produce ninguna acción, si se desea salir del modo CU, se vuelve a presionar este comando y el robot retoma el control manual.

8.4. Comunicación Smartphone-HermesII

La comunicación entre el dispositivo de control y Hermes II, se realiza mediante Bluetooth.

Los comandos enviados desde la aplicación al robot, son interpretados y procesados en la placa de desarrollo principal Raspberry Pi B+ de la forma que el desarrollador deseé.

Actualmente queda implementado en el robot un algoritmo que ejecuta una acción en base al carácter que recibe.

También se deja implementado un algoritmo que recibe un mensaje de tipo cadena de caracteres, que contiene la misma información representada de dos formas diferentes:

- $[AnguloDeAvance, ModuloDelVectorDireccion, MomentoAngular]$
- $[CoordenadaX, CoordenadaY, MomentoAngular]$

Ángulo de avance: Hace referencia a la dirección que se le da al robot para que se mueva.

Módulo del vector dirección: Al darse el ángulo de avance, se debe precisar junto con él un valor escalar que representa la velocidad a la que debe avanzar el robot en la dirección especificada.

Momento angular: Este valor indica la orden de rotar al robot, por ejemplo si se desea que el robot gire en sentido horario, los dos primeros valores deben ser cero, y momento angular positivo. Dependiendo del valor del momento angular, el robot girará a mayor o menor velocidad.

Coordinada X: Valor del eje de coordenadas para formar el vector dirección.

Coordinada Y: Valor del eje de ordenadas para formar el vector dirección.

Se implementarán ambos tipos de mensajes para una mayor flexibilidad, dejando como criterio de uso al programador.

El formato del mensaje es un buffer de 4 bytes, los cuales indican:

byte[0]: Tipo de mensaje.

Tipo de mensaje 1: indica que los tres datos siguientes son, ángulo de avance; módulo de vector dirección; momento angular.

Tipo de mensaje 2: indica que los tres datos siguientes son, coordenada X; coordenada Y; momento angular.

byte[1]: depende del tipo de mensaje.

byte[2]: depende del tipo de mensaje.

byte[3]: momento angular

8.5. Proceso de decodificación de los mensajes.

Una vez recibido el mensaje desde el dispositivo Bluetooth en la placa de desarrollo principal Raspberry Pi B+, este es procesado de la siguiente manera:

Se recibe un dato del tipo cadena de caracteres (4 bytes).

Se parsea la cadena de caracteres y se obtienen tres datos (ángulo, módulo, momento angular) que son los que representan la dirección de avance del robot y el sentido de giro respectivamente.

Se procesan estos datos en una matriz, la cual da como resultado cuatro valores escalares correspondientes a la velocidad y sentido de giro (según el signo) de los cuatro motores.

Los valores resultantes deben ser tratados para conformar el mensaje que se envía al actuador para que este se encargue de controlar los motores.

A partir de los cuatro datos devueltos por la matriz de transformación, se forma el mensaje a enviar por I2c al actuador de la siguiente manera: Se chequea si el dato es negativo o positivo para conformar el dato correspondiente al byte[1] indicando sentido de giro BACKWARD o FORWARD. Se toma el valor absoluto correspondiente a cada motor

Se chequea si los valores son mayor que 127, de ser así, se coloca el valor 127 en el primer byte correspondiente al motor, y el resto se coloca en el siguiente byte. Si el valor no es mayor que 127, se coloca directamente el valor en el primer byte del motor, y en el byte siguiente, se coloca un cero. De esta forma queda conformado el mensaje a enviar por I2c al actuador.

8.6. Casos de uso

A continuación se realizan los casos de uso que se consideran más relevantes.

8.6.1. Robot seguidor de línea

Caso de uso (CU-1)	Seguidor de línea negra
Actores	Usuario y robot
Tipo	Básico
Propósito	Haciendo uso del sensor seguidor de línea óptico el robot seguirá la trayectoria demarcada en el piso.
Resumen	El robot recibe la orden para seguir la línea negra.
Precondiciones	<ul style="list-style-type: none">• Estar establecida la conexión entre el dispositivo y el robot.• El sensor óptico debe estar conectado.• El robot debe estar ubicado sobre la línea negra.
Flujo principal	<ul style="list-style-type: none">• El usuario envía un comando al robot para comenzar a seguir la línea de forma autónoma.• El sistema toma datos del sensor para hacer funcionar los actuadores mediante los cuales corrige su trayectoria.
Excepciones	<ul style="list-style-type: none">• Que las conexiones no funcionen.• Que el robot pierda la línea.• Que el sensor deje de funcionar.
Poscondición	El robot sigue la línea negra de forma autónoma hasta que reciba el comando de finalización.

Tabla 8.1: Caso de uso 1

Caso de uso (CU-2)	Finalizar seguidor de línea
Actores	Usuario y robot
Tipo	Extendido
Propósito	Detener la ejecución autónoma del robot.
Resumen	El robot recibe la orden para finalizar la rutina seguir la línea.
Precondiciones	<ul style="list-style-type: none"> • Que el robot esté siguiendo la línea de forma autónoma. • Que las conexiones estén establecidas.
Flujo principal	<ul style="list-style-type: none"> • El usuario envía un comando al robot para finalizar el algoritmo seguidor de línea.
Excepciones	<ul style="list-style-type: none"> • Que las conexiones se hayan interrumpido.
Poscondición	El robot queda detenido a la espera de nuevos comandos.

Tabla 8.2: Caso de uso 2

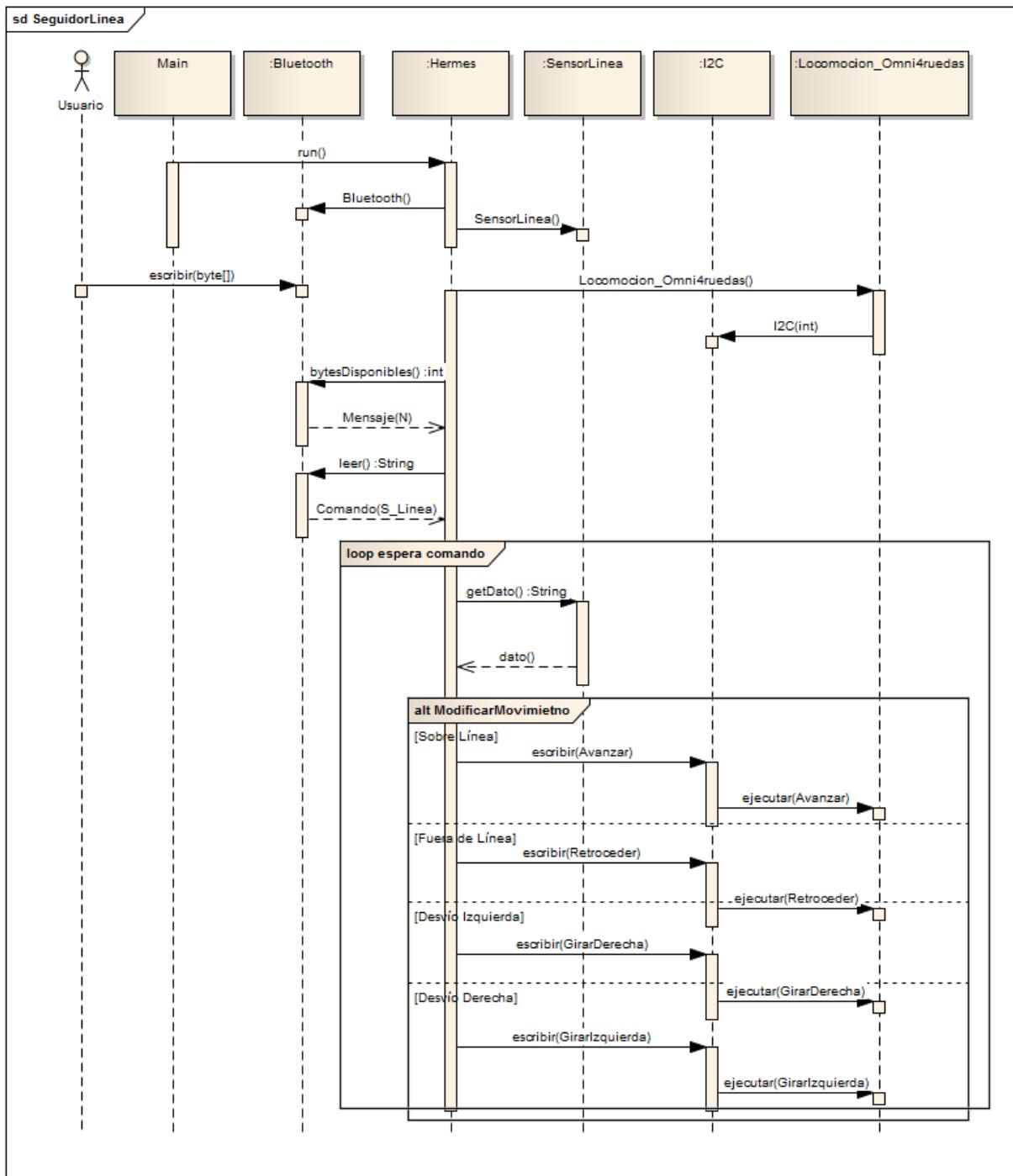


Figura 8.4: Diagrama de secuencia Seguidor de Línea

8.6.2. Movimiento mediante comandos

Caso de uso (CU-3)	Moverse
Actores	Usuario y robot
Tipo	Básico
Propósito	El robot debe realizar el movimiento correspondiente para cada comando.
Resumen	El robot recibe un comando, lo procesa y ejecuta un movimiento.
Precondiciones	<ul style="list-style-type: none">• Tener establecida la comunicación entre el robot y el usuario.
Flujo principal	<ul style="list-style-type: none">• El robot recibe comandos enviados por el usuario, los procesa y ejecuta acciones en los actuadores para realizar el movimiento.
Excepciones	<ul style="list-style-type: none">• Que las conexiones no funcionen.
Poscondición	El robot responde a los comandos enviados por el usuario.

Tabla 8.3: Caso de uso 3

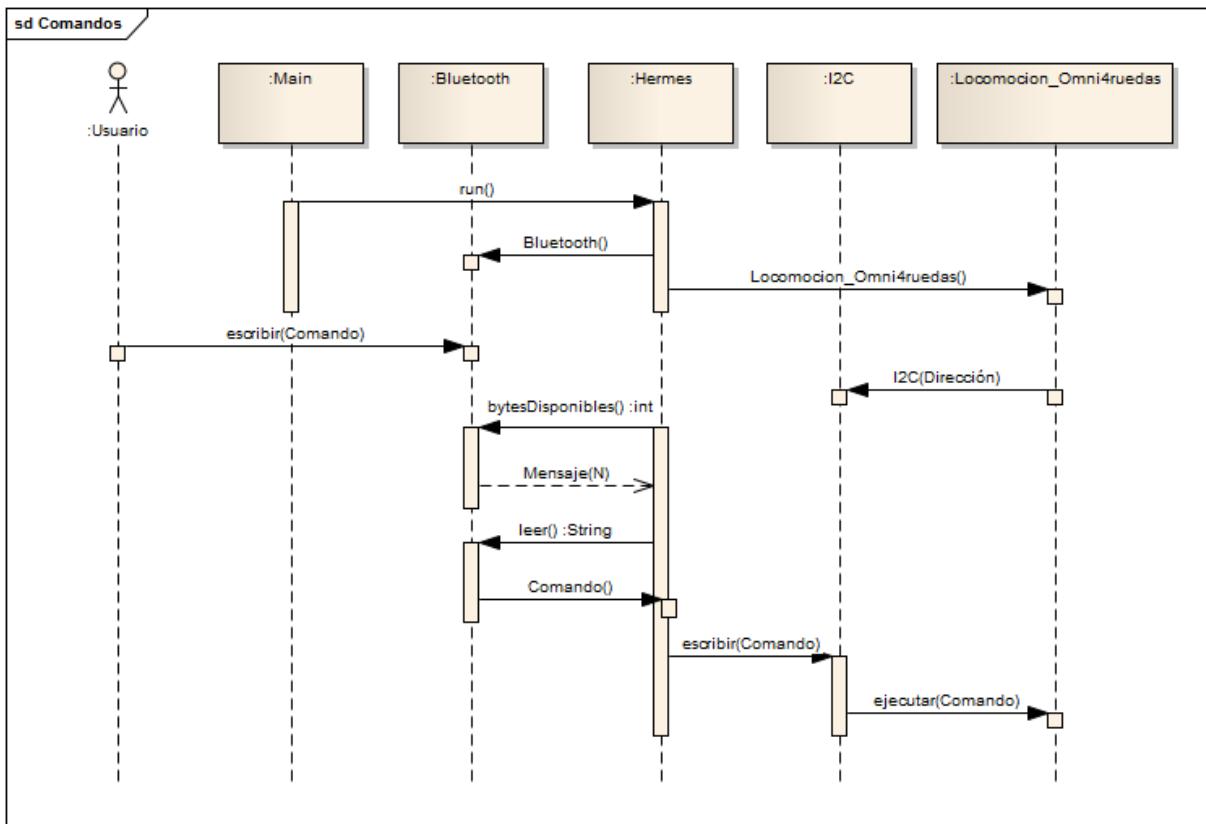


Figura 8.5: Diagrama de secuencia manejado por comandos

8.6.3. Movimiento con detección de obstáculos

Caso de uso (CU-4)	Movimiento con detección de obstáculos.
Actores	Usuario y robot.
Tipo	Básico.
Propósito	El robot debe poder avanzar y detectar obstáculos.
Resumen	El robot recibe un comando para comenzar a avanzar de forma autónoma detectando obstáculos.
Precondiciones	<ul style="list-style-type: none">• Tener las conexiones establecidas.• Tener el sensor de proximidad conectado.
Flujo principal	<ul style="list-style-type: none">• El usuario envía un comando al robot para comenzar a avanzar y detectar obstáculos en su camino de forma autónoma.• El sistema toma datos del sensor de proximidad para detectar obstáculos.• El robot ejecuta rutina de evasión de obstáculos.
Excepciones	<ul style="list-style-type: none">• Que las conexiones no funcionen.• Que no se mida correctamente la distancia.
Poscondición	El robot se mueve de forma autónoma evadiendo obstáculos.

Tabla 8.4: Caso de uso 4

Caso de uso (CU-5)	Evasión de obstáculos.
Actores	Robot.
Tipo	Extendido.
Propósito	El robot debe tomar acciones para no colisionar.
Resumen	Una vez que el robot detecta un obstáculo debe ejecutar una rutina para modificar su trayectoria.
Precondiciones	<ul style="list-style-type: none"> • Tener las conexiones establecidas. • Tener el sensor de proximidad conectado. • Haber detectado un obstáculo.
Flujo principal	<ul style="list-style-type: none"> • El robot se detiene. • El sistema calcula una nueva trayectoria. • El sistema tomará datos del sensor para ver si puede avanzar en la nueva trayectoria. • Avanza en la nueva dirección.
Excepciones	<ul style="list-style-type: none"> • Que las conexiones no funcionen. • Que no se mida correctamente la distancia.
Poscondición	El robot avanza en la nueva dirección.

Tabla 8.5: Caso de uso 5

Caso de uso (CU-6)	Finalizar evasión de obstáculos.
Actores	Usuario y robot.
Tipo	Extendido.
Propósito	Detener la ejecución autónoma del robot.
Resumen	El robot recibe la orden para finalizar la rutina de evadir obstáculos.
Precondiciones	<ul style="list-style-type: none"> • Que el robot se esté moviendo en modo de detección de obstáculos. • Que las conexiones estén establecidas.
Flujo principal	<ul style="list-style-type: none"> • El usuario envía un comando al robot para detenerlo.
Excepciones	<ul style="list-style-type: none"> • Que las conexiones se hayan interrumpido.
Poscondición	El robot queda detenido a la espera de nuevos comandos.

Tabla 8.6: Caso de uso 6

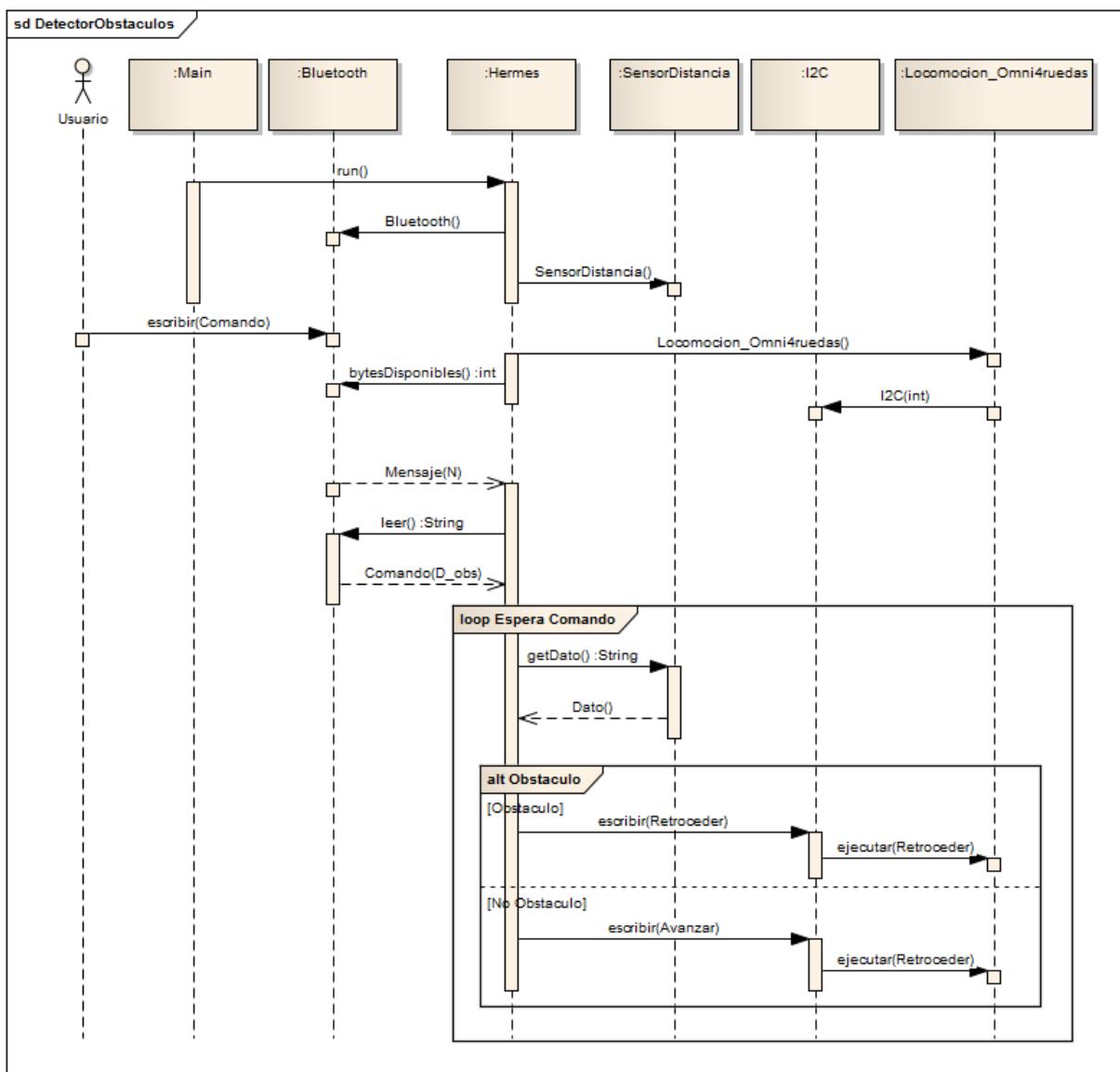


Figura 8.6: Diagrama de secuencia de evasor de obstáculos

8.7. Matriz de trazabilidad

Requerimientos \ Pruebas	A	B	C	D	E	F	G	H	I	J	K
1		X		X			X			X	X
2					X						
3								X	X	X	X
4									X	X	X

Tabla 8.7: Matriz de trazabilidad

Requerimientos:

1. Movimiento holonómico.
2. Comunicación usuario-robot.
3. Reconocimiento de ambientes mediante sensores.
4. Programación de hilos.

Casos de prueba:

- A. Comunicación Bluetooth (Arduino)
- B. Biblioteca
- C. Funcionamiento del robot
- D. Error de trayectoria
- E. Comunicación Bluetooth (Raspberry)
- F. Comunicación I2C
- G. Funcionamiento de actuador de locomoción
- H. Sensor de línea
- I. Sensor de distancia
- J. Caso de uso (1): seguidor de línea negra
- K. Caso de uso (4): movimiento con detección de obstáculos

RESULTADOS

Se obtuvo un robot omnidireccional de mecánica simple con cuatro motores de corriente continua anclados al chasis para la tracción y dirección del mismo.

Para permitir el movimiento de forma holonómica, se utilizaron cuatro ruedas omnidireccionales posicionadas de forma perpendicular a los ejes del plano cartesiano, disminuyendo las partes móviles del sistema.

Hermes II dispone de un sensor de distancia y dos de proximidad para reconocimiento de ambiente.

Se desarrolló un software en lenguaje Java, con arquitectura basada en componentes, para facilitar la incorporación de nuevos sensores, actuadores y sistemas de comunicación. Los componentes desarrollados para el robot pueden ser instanciados por el estudiante en la clase “Hermes” según sus necesidades.

En esta versión se incluyen a modo de ejemplo dos escenarios posibles haciendo uso de sensores. Estos ejemplos se encuentran desarrollados en el capítulo 7

Se deja a disposición del usuario el plano del robot y el manual con las piezas numeradas para facilitar el armado del mismo en el Anexo C. Además se incluye un cd en el cual se encuentran los archivos vectorizados de las piezas para que puedan ser construidas con sistema de corte láser, de bajo costo de proceso y materiales; y el software del proyecto.

CONCLUSIONES

Se diseñó y construyó un robot holonómico de cuatro ruedas omnidireccionales, las cuales permitieron implementar un sistema mecánico con pocas piezas móviles que incrementan el grado de confiabilidad mecánica. El sistema, al ser de mecánica simple, facilita su construcción y prolonga su vida útil.

Las ruedas utilizadas fueron el elemento disruptivo del proyecto, ya que permiten modificar la trayectoria del robot sin modificar la orientación de las mismas.

Hermes II posee un sistema operativo que permite al usuario desarrollar programación concurrente en distintos lenguajes. Cabe destacar que la versión actual cuenta con bibliotecas para programación en lenguaje Java.

Debido al diseño de piezas encastrables, el robot puede ser armado de forma fácil e intuitiva con materiales de bajo costo. Ésto permite al usuario participar desde la etapa de construcción mecánica y electrónica hasta la programación del robot.

En cuanto al costo, se obtuvo un versión básica del robot que cumple con el requerimiento de competitividad en el mercado, pero cuenta con la posibilidad de incorporación de nuevos sensores y actuadores para su ampliación.

Hermes II fue presentado en eventos públicos como Expocarrera, y Edutech (semana TIC) despertando curiosidad e interés en las personas presentes. Además se expuso en la academia de ciencias durante treinta días, donde fue utilizado y exhibido por estudiantes representantes de la Facultad.

Al inicio del proyecto se desarrolló un plan de negocios con el cual se obtuvo una beca otorgada por el Ministerio de Ciencia, Tecnología e Innovación Productiva, destinada a la construcción de las primeras diez unidades. Al momento de la presentación de este proyecto integrador, se encuentran disponibles los materiales para la construcción de los primeros 5 robots con el fin de ser utilizados como prueba en la asignatura programación concurrente.

Utilizar el método de desarrollo iterativo con entrega incremental permitió obtener rápidamente un primer prototipo sobre el cual se realizaron diversas pruebas, detectando falencias a solucionar en una nueva iteración.

TRABAJOS FUTUROS

- Incorporar nuevos sensores.
- Modificar la aplicaciil telno para que el robot se pueda desplazar en los 360° del plano.
- Desarrollar e incorporar un sistema de control de velocidad de los motores.
- Incorporar nuevos actuadores para interaccin otros robots.
- Dise desarrollar hardware para apagado seguro del robot.
- Dise desarrollar circuito impreso para la integraci los componentes de hardware del robot.
- Desarrollar una interfaz grca que permita la programacir bloques para uso escolar.
- Realizar bibliotecas para programaci lenguajes C++ y Python.
- Incorporar una FPGA al robot para programaci hardware.
- Realizar un programa donde se ponga en evidencia un problema de concurrencia entre hilos.
- Constatar mediante un grupo de estudiantes si el robot puede ser armado desde cero en seis horas como se estipula en el requerimiento y documentar los resultados obtenidos.

BIBLIOGRAFÍA

- [1] Lichtensztein Leandro, Córdoba 2013 . *Sistemas embebidos para la educación: Hermes*.
- [2] A.Ollero Baturone. Robótica: Manipuladores y robots móviles, Barcelona: Marcombo S.A, 2001
- [3] Ian Sommerville. *Ingeniería de software, 9th edition*, México:Pearson Educación S.A, 2011.
- [4] Wikipedia, the free encyclopedia, Free and open source software. <http://en.wikipedia.org>
- [5] Coordinación de ciencias computacionales <http://ccc.inaoep.mx/~esucar/Clases-irob/ir2-locomocion.pdf>.
- [6] Antonio Barrientos, Luis Felipe Peñín, Carlos Balaguer, Rafael Aracil. *Fundamentos de robótica*, Madrid: Mc Graw Hill, 2007.
- [7] CCMBenchmark group <http://es.ccm.net/contents/69-como-funciona-bluetooth>
- [8] Robots Argentina http://robots-argentina.com.ar/Comunicacion_busI2C.htm
- [9] García Cabral Ana Belén, Sagripanti Sergio Javier, Córdoba 2014 . *Estudio y rediseño de robot omnidireccional Hermes*
- [10] Probattery <http://www.probattery.com.ar/nueva/>
- [11] Antonio Barrientos, Luis Felipe Peñín, Carlos Balaguer, Rafael Aracil. *Fundamentos de robótica*, Madrid: Mc Graw Hill, 2007.
- [12] John J. Craig *Introduction to Robotics: Mechanics and Control, 2da edición*, Boston: Addison-Wesley Longman Publishing Co.Inc, 1989
- [13] Datasheet <http://pdf.datasheetcatalog.com/datasheet2/6/0ehgr3166z3dyrxp33y124rtqo3y.pdf>
- [14] Katsuhiko Ogata *Ingeniería de control moderna, 5ta edición*, Madrid: Pearson Educación S.A., 2010.
- [15] Hypertextual <http://bitelia.com/2013/12/raspberry-pi-distribuciones-linux>
- [16] RaspberryPi <http://www.raspberrypi.org/>
- [17] Pi4J <http://pi4j.com/>
- [18] Stack Exchange <http://raspberrypi.stackexchange.com/questions/27073/firmware-3-18-x-breaks-i2c-spi-audio-lirc-1-wire>
- [19] Blog Pablo Adrian <http://quieroentrarentupagina.blogspot.com.ar/2014/03/conectar-un-sensor-hc-sr04-con-la.html>

PARTE I

ANEXOS

ANEXO A

DESARROLLO DE BIBLIOTECA PARA EL CONTROL DE LOS MOTORES

A.1. DriverMotor.h

```
/*
DriverMotor.h - Libreria usada para el control de los motores.
Creada por Garcia y Sagripanti , Marzo 2015
*/
#ifndef DriverMotor_h
#define DriverMotor_h
#include "Arduino.h"

#define GIRO_MOTOR2_A 7 // pin digital , sentido de giro del motor1
#define GIRO_MOTOR2_B 4 // pin digital , sentido de giro del motor1
#define MOTOR2 6 // pin correspondiente al PWM del motor1

#define GIRO_MOTOR1_A 10 // pin digital , sentido de giro del motor2
#define GIRO_MOTOR1_B 13 // pin digital , sentido de giro del motor2
#define MOTOR1 5 // pin correspondiente al PWM del motor2

#define GIRO_MOTOR3_A A3 // pin digital , sentido de giro del motor3
#define GIRO_MOTOR3_B A2 // pin digital , sentido de giro del motor3
#define MOTOR3 3 // pin correspondiente al PWM del motor3

#define GIRO_MOTOR4_A 8 // pin digital , sentido de giro del motor4
#define GIRO_MOTOR4_B 9 // pin digital , sentido de giro del motor4
#define MOTOR4 11 // pin correspondiente al PWM del motor4

#define FORWARD 0
#define BACKWARD 1
#define RELEASE 2

class DriverMotor
{
public:
    DriverMotor(uint8_t motornum);
    void setSpeed(uint8_t motor_speed);
    void motorStart(uint8_t direccion);
private:
    uint8_t motornum;
    uint8_t pin_giro_A ;
    uint8_t pin_giro_B ;
    uint8_t pin_pwm;
```

```
};

#endif
```

A.2. DriverMotor.cpp

```
/*
DriverMotor.h - Libreria usada para el control de los motores.
Creada por Garcia y Sagripanti , Marzo 2015
*/
#include "DriverMotor.h"
#include "Arduino.h"

DriverMotor::DriverMotor(uint8_t num){
    motornum = num;
    switch(num){
        case 1:
            pin_giro_A = GIRO_MOTOR1_A;
            pin_giro_B = GIRO_MOTOR1_B;
            pin_pwm = MOTOR1;
            pinMode(GIRO_MOTOR1_A,OUTPUT);
            pinMode(GIRO_MOTOR1_B,OUTPUT);
            pinMode(MOTOR1,OUTPUT);
            break;
        case 2:
            pin_giro_A = GIRO_MOTOR2_A;
            pin_giro_B = GIRO_MOTOR2_B;
            pin_pwm = MOTOR2;
            pinMode(GIRO_MOTOR2_A,OUTPUT);
            pinMode(GIRO_MOTOR2_B,OUTPUT);
            pinMode(MOTOR2,OUTPUT);
            break;
        case 3:
            pin_giro_A = GIRO_MOTOR3_A;
            pin_giro_B = GIRO_MOTOR3_B;
            pin_pwm = MOTOR3;
            pinMode(GIRO_MOTOR3_A,OUTPUT);
            pinMode(GIRO_MOTOR3_B,OUTPUT);
            pinMode(MOTOR3,OUTPUT);
            break;
        case 4:
            pin_giro_A = GIRO_MOTOR4_A;
            pin_giro_B = GIRO_MOTOR4_B;
            pin_pwm = MOTOR4;
            pinMode(GIRO_MOTOR4_A,OUTPUT);
            pinMode(GIRO_MOTOR4_B,OUTPUT);
            pinMode(MOTOR4,OUTPUT);
            break;
    }
}

//con esta función se setea la velocidad del motor
void DriverMotor::setSpeed(uint8_t motor_speed){
```

```
if(motor_speed > 200) motor_speed = 200;
    analogWrite(pin_pwm, motor_speed);
}

void DriverMotor::motorStart(uint8_t direccion){
    switch(direccion){
        case 0: //FORWARD
            digitalWrite(pin_giro_A, 0);
            digitalWrite(pin_giro_B, 1);
            break;
        case 1: //BACKWARD
            digitalWrite(pin_giro_A, 1);
            digitalWrite(pin_giro_B, 0);
            break;
        case 2: //RELEASE
            digitalWrite(pin_giro_A, 0);
            digitalWrite(pin_giro_B, 0);
            analogWrite(pin_pwm, 0);
            break;
    }
}
```

ANEXO B

CÓDIGOS DE PRUEBA

B.1. Comunicación Bluetooth con Arduino

```
char incomingByte; // byte leido

void setup() {
    Serial.begin(9600); // abre el puerto serie a 9600 bps
}

void loop() {
    uint8_t i;

    if (Serial.available() > 0) { // verifica si hay datos disponibles
        incomingByte = Serial.read(); // lee los datos
        Serial.print("Recibi:");
        Serial.print(incomingByte); // Envía al dispositivo el dato
                                     recibido
    }
}
```

B.2. Biblioteca

```
#include <DriverMotor.h> //Usada para el control de los motores
#include <TimerOne.h> //Usada para medir las rpm de los motores

//Inicializacion de un motor
DriverMotor motor1(1);

int velocidad = 70; //velocidad inicial = 70
boolean sentido_giro = false;

void setup(){
    Timer1.initialize(1000000); // interrupción de timer cada 1 segundo
    Timer1.attachInterrupt(timer);
    Serial.begin(9600); //inicializamos la comunicación serial
}

void loop(){
    if(sentido_giro)
        horario();
    else
```

```

        antihorario();
    }

void horario(){ //sentido horario del motor
    motor1.motorStart(FORWARD);
    motor1.setSpeed(velocidad);
}

void antihorario(){ //sentido antihorario del motor
    motor1.motorStart(BACKWARD);
    motor1.setSpeed(velocidad);
}

void frenar(){ //detener el motor
    motor1.motorStart(RELEASE);
}

void timer(){
    //reiniciar la velocidad e invertir el sentido de giro
    if(velocidad == 100){
        frenar();
        velocidad = 70;
        sentido_giro=!sentido_giro;
    }
    //incrementar la velocidad
    else{
        velocidad = velocidad + 10;
    }
}

```

B.3. Funcionamiento del robot

```

#include <DriverMotor.h> //Usada para el control de los motores

DriverMotor motor1(1); // crea motor #1
DriverMotor motor2(2); // crea motor #2
DriverMotor motor3(3); // crea motor #3
DriverMotor motor4(4); // crea motor #4

char incomingByte; // para el byte leido
int velocidad = 80;
void setup() {
    Serial1.begin(9600); // abre el puerto serie a 9600 bps
}

void loop() {
    uint8_t i;

    if (Serial1.available() > 0) {
        incomingByte = Serial1.read();
        Serial.println(incomingByte);
        switch (incomingByte){
            case '1':

```

```

        avanzar ();
        break;
    case '2':
        retroceder ();
        break;
    case '3':
        girarIzq ();
        break;
    case '4':
        girarDer ();
        break;
    case '5':
        frenar ();
        break;
    default:
        break;
    }
}
}

void frenar (){
    motor1 . run (RELEASE);
    motor2 . run (RELEASE);
    motor3 . run (RELEASE);
    motor4 . run (RELEASE);
}

void avanzar (){
    motor1 . run (BACKWARD);
    motor1 . setSpeed (velocidad );
    motor2 . run (FORWARD);
    motor2 . setSpeed (velocidad );
    motor3 . run (FORWARD);
    motor3 . setSpeed (velocidad );
    motor4 . run (BACKWARD);
    motor4 . setSpeed (velocidad );
    delay (500);
    frenar ();
}

void retroceder (){
    motor1 . run (FORWARD);
    motor1 . setSpeed (velocidad );
    motor2 . run (BACKWARD);
    motor2 . setSpeed (velocidad );
    motor3 . run (BACKWARD);
    motor3 . setSpeed (velocidad );
    motor4 . run (FORWARD);
    motor4 . setSpeed (velocidad );
    delay (500);
    frenar ();
}

void girarIzq (){
    motor1 . run (BACKWARD);
    motor1 . setSpeed (velocidad );

```

```

        motor2.run(BACKWARD);
        motor2.setSpeed(velocidad);
        motor3.run(BACKWARD);
        motor3.setSpeed(velocidad);
        motor4.run(BACKWARD);
        motor4.setSpeed(velocidad);
        delay(500);
        frenar();
    }

void girarDer(){
    motor1.run(FORWARD);
    motor1.setSpeed(velocidad);
    motor2.run(FORWARD);
    motor2.setSpeed(velocidad);
    motor3.run(FORWARD);
    motor3.setSpeed(velocidad);
    motor4.run(FORWARD);
    motor4.setSpeed(velocidad);
    delay(500);
    frenar();
}

```

B.4. Comunicación Bluetooth con Raspberry Pi

```

import java.util.Date;
import com.pi4j.io.serial.Serial;
import com.pi4j.io.serial.SerialDataEvent;
import com.pi4j.io.serial.SerialDataListener;
import com.pi4j.io.serial.SerialFactory;
import com.pi4j.io.serial.SerialPortException;

public class SerialExample{
    public static void main(String args[])throws InterruptedException{
        final Serial serial = SerialFactory.createInstance();
        char recibido;
        try {
            serial.open(Serial.DEFAULT_COM_PORT, 9600);
            System.out.println("Comunicación establecida");
            for(;;){
                try {
                    serial.write("prueba_de_write");
                    recibido = serial.read();
                    System.out.println("lo que lei");
                    System.out.println(recibido);
                }
                catch(IllegalStateException ex){
                    ex.printStackTrace();
                }
                Thread.sleep(1000);
            }
        }
        catch(SerialPortException ex){

```

```
        System.out.println(ex.getMessage());
    return;
}
}
```

ANEXO C

MANUAL DE ARMADO

Despiece

