

# Construcción de una plataforma robótica abierta para pruebas de desempeño de componentes y algoritmos

Daniel Garcés Márín

21 de junio de 2019

Tesis para obtener el grado de Ingeniero en Computación

Con apoyo del Laboratorio de Biorobotica

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	2
1.2. Planteamiento del problema . . . . .	3
1.3. Objetivos . . . . .	3
1.4. Descripción del documento . . . . .	4
<b>2. Antecedentes y Marco Teórico</b>	<b>7</b>
2.1. El Concepto <i>Open Source</i> . . . . .	8
2.2. Concepto de Robot Móvil Autónomo . . . . .	12
2.3. Esquema General de un Robot Móvil . . . . .	14
2.4. Sistema Operativo ROS . . . . .	16
2.5. Arduino . . . . .	22
2.6. Raspberry . . . . .	24
2.7. Estado de la técnica . . . . .	26
<b>3. Diseño y construcción de la plataforma</b>	<b>31</b>
3.1. Estructura General del Diseño de la Plataforma Robótica . . . . .	32
3.2. Sistema de Sensores . . . . .	33
3.3. Sistema de Actuadores . . . . .	39
3.4. Unidades de Procesamiento . . . . .	42
3.5. Sistema de Suministro de Potencia . . . . .	48
3.6. Sistema Mecánico Estructural . . . . .	54
<b>4. Arquitectura del software</b>	<b>63</b>
4.1. Sistema operativo . . . . .	64
4.2. Arquitectura de la Plataforma Robótica Móvil basada en ROS	65
4.3. Paquetes predeterminados de ROS, <i>ROS packages</i> . . . . .	66
4.4. Descripción de los paquetes de hardware . . . . .	68

4.5. Descripción de Tópicos y Servicios . . . . .	71
4.6. Descripción de los paquetes <i>launch</i> . . . . .	73
<b>5. Algoritmos para pruebas de desempeño</b>	<b>75</b>
5.1. Navegación Autónoma . . . . .	76
5.2. Planeación de Acciones . . . . .	78
5.3. Visión Computacional . . . . .	78
<b>6. Resultados</b>	<b>79</b>
<b>7. Conclusiones y Trabajo a futuro</b>	<b>81</b>
<b>A. Esquema General de un robot</b>	<b>83</b>
<b>B. Lista de componentes de la plataforma robótica</b>	<b>85</b>
<b>C. Esquema general de conexiones.</b>	<b>89</b>
<b>D. Proceso de ensamblaje</b>	<b>91</b>
D.1. Armado del chasis de aluminio estructural . . . . .	91
D.2. Montaje y cableado de los elementos laterales . . . . .	93
D.3. Montaje de los motores . . . . .	94
D.4. Montaje de los elementos interiores . . . . .	97
D.5. Cableado eléctrico de los elementos interiores . . . . .	98
D.6. Conexión de los componentes electrónicos . . . . .	99
D.7. Montaje de los elementos superiores . . . . .	101
D.8. Plataforma robótica móvil terminada . . . . .	103
<b>E. Arquitectura completa del sistema</b>	<b>105</b>
<b>F. Instalación del sistema</b>	<b>107</b>
F.1. Repositorio del proyecto . . . . .	107
F.2. Contenido del repositorio del proyecto . . . . .	108
F.3. Instalación de paqueteterías adicionales . . . . .	109
F.4. Compilación del workspace del sistema . . . . .	112
F.5. Modo de prueba . . . . .	112

# Índice de figuras

2.1.	Un hombre, una mujer y tres robots, escena de la puesta en escena de la obra teatral <i>R.U.R. de Karel Capek</i> . . . . .	12
2.2.	Robot móvil autónomo. . . . .	15
2.3.	Esquema de trabajo de ROS. . . . .	16
2.4.	Robot TurtleBot modelo 3. . . . .	26
2.5.	Robot PeopleBot . . . . .	27
2.6.	Mindstorm EV3 . . . . .	28
2.7.	Robot Justina de la UNAM. . . . .	30
3.1.	Esquema general de un robot. . . . .	32
3.2.	Elementos del sistema de sensores. . . . .	33
3.3.	Alarma de bajo voltaje para pilas LiPo de 2S a 6s . . . . .	34
3.6.	Telémetro láser de escaneo RPLIDAR A1 M8. . . . .	38
3.7.	Elementos del sistema de actuadores. . . . .	39
3.8.	Motor Gear EMG49. . . . .	40
3.9.	Esquemáticos del Motor-Gear EMG39. . . . .	41
3.10.	Elementos de las unidades de procesamiento. . . . .	42
3.11.	ARDUINO MEGA2560. . . . .	43
3.12.	Raspberry Pi 3 Model B. . . . .	45
3.13.	Roboclaw modelo 2x15A. . . . .	47
3.14.	Elementos del sistema eléctrico electrónico. . . . .	48
3.15.	Esquema del circuito eléctrico del robot. . . . .	49
3.16.	Batería LiPo 14.8 V. PowerHD . . . . .	50
3.17.	Botón de paro de emergencia estándar. . . . .	51
3.18.	Voltímetro digital 0-100V. . . . .	51
3.19.	Regulador de voltaje conmutado pololú D15V70F5S3. . . . .	52
3.20.	Elementos del Sistema Mecánico Estructural. . . . .	54
3.21.	Esquema del chasis propuesto. . . . .	55

*ÍNDICE DE FIGURAS*

3.22. Organización de los componentes propuesta. . . . .	57
3.23. Rueda 125 de Robo electronics. . . . .	58
3.25. Carcasa protectora para Arduino Mega. . . . .	60
3.26. Carcasa protectora para Raspberry Pi 3. . . . .	60
4.1. Arquitectura del sistema implementado en la plataforma robótica móvil. . . . .	65
A.1. Esquema general de un robot completo. . . . .	83
C.1. Esquema general de conexiones de los componentes electrónicos. . . . .	89
D.1. Chasis armado. . . . .	92
D.2. Chasis armado vista frontal. . . . .	93
D.3. Montaje de los motores y ruedas. . . . .	94
D.4. Soporte de motores y ruedas giratorias motadas. . . . .	95
D.5. Colocación de ambos motores. . . . .	95
D.6. Chasis armado, motores y ruedas montados. . . . .	96
D.7. Elementos interiores colocados. . . . .	97
D.8. Elementos interiores alambrados. . . . .	98
D.9. Conexiones Arduino Mega. . . . .	99
D.10. Placa fenólica construida. . . . .	100
D.11. Botón de paro de emergencia colocado. . . . .	102
D.12. Láser RPLIDAR colocado. . . . .	102
D.13. Plataforma Robótica Móvil armada. . . . .	103
E.1. Arquitectura completa del sistema implementado en la plataforma robótica móvil. . . . .	105

# Capítulo 1

## Introducción

Aquí va algo de introducción

## 1.1. Motivación

En años recientes hemos podido observar la presencia de la robótica en diversas actividades humanas partiendo del ámbito industrial, como serían las líneas de ensamble automatizadas, proyectos de investigación y desarrollo de carácter docente, hasta la presencia de los robots en los hogares favoreciendo el fuerte crecimiento de la domótica y el internet de las cosas. Partiendo de este escenario se hace evidente que aquellos que quieran comenzar a desarrollar en este campo tendrán que superar la clara ventaja que ofrecen muchos de los prototipos o modelos de robots con los que cuentan algunos sectores privados de la industria de la tecnología o centros de investigación; sin embargo cada uno de los diseños disponibles actualmente hace uso de los mismos conceptos mecánicos y eléctricos, ofreciendo claramente algunas ventajas en alguna condición o característica sobre otros dependiendo del objetivo que se busque con el diseño de robot.

Por lo que, en la búsqueda de ofrecer una solución para aquellos inmersos en el campo de la robótica, se propone ofrecer un diseño estándar y de carácter abierto de una plataforma robótica móvil usando características equivalentes a aquellas plataformas comerciales ofrecidas por empresas privadas, en la cual puedan comenzar de un punto estable de investigación y desarrollo de nuevos componentes, algoritmos o bien para ser utilizada como parte de material didáctico en instituciones académicas interesadas en la enseñanza de la robótica.

Siendo uno de los objetivos de la propuesta desarrollada en el presente trabajo que presente una naturaleza similar a los proyectos desarrollados bajo la filosofía Open Source, esto es, que cuente con un diseño en hardware y en software que pueda ser usado por los usuarios; incluso de ser posible fomentar la formación de una comunidad activa la cual, con el aporte de nuevo conocimiento y experiencias, desarrolle nuevas ideas, diseños o modificaciones al sistema. Podemos tomar como ejemplo de la efectividad de este movimiento los proyectos que involucran las plataformas Arduino, Raspberry o la comunidad de ROS.

## 1.2. Planteamiento del problema

En la actualidad existe un número limitado de plataformas robóticas móviles comerciales en las cuales aquellos interesados en el campo de la robótica puedan basarse para la prueba de algoritmos o hardware de diseño propio. Esto debido a que los proveedores manejan políticas de uso de sus productos en las cuales impiden modificaciones al hardware o bien no existen plataformas robóticas abiertas.

## 1.3. Objetivos

### Objetivo general

Diseño e implementación de una plataforma robótica abierta para uso docente y de investigación que ofrezca a aquellos inmersos en el campo de la robótica, características equivalentes a aquellas plataformas comerciales ofrecidas por empresas privadas, con la ventaja de que tanto el diseño en hardware y en software puede ser modificado por el usuario, adaptando el diseño de la plataforma robótica, a nivel de hardware, para disminuir su costo y hacerla más accesible.

### Objetivos particulares

- Desarrollar una plataforma robótica móvil siguiendo las metodologías de la iniciativa Open Source en cuestión de diseño de hardware y software.
- Documentar el diseño y características de todos los elementos mecánicos, eléctricos y electrónicos que se requieren para la construcción de una plataforma robótica base, así como el proceso de construcción de la misma.
- Desarrollar una arquitectura de software implementada en ROS con los algoritmos necesarios con el que se pueda tener un correcto funcionamiento de los componentes incluidos en la plataforma.
- Ofrecer alternativas para la construcción de la plataforma en caso de no contar con los elementos sugeridos y ofrecer soporte en la plataforma de software para dichos elementos.

- Implementar algoritmos usados en temas de navegación, visión computacional y planeación de acciones; consecuentemente implementar algoritmos para la calibración, adecuación y caracterización de sensores, así como el análisis de la información recopilada por los mismos.

## 1.4. Descripción del documento

El presente trabajo representa la investigación que se realizó en el campo de la robótica móvil autónoma, en base a la cual se realizó el diseño y construcción de los diferentes elementos y sistemas que integran un robot móvil; siendo cada una de las etapas de desarrollo desglosadas en los 7 capítulos que conforman el trabajo, organizados de la siguiente manera:

El primer capítulo, Introducción, busca asentar las bases con las cuales se planteo el proyecto, así como las razones y motivaciones que llevaron a la implementación del mismo. Tratando de exponer claramente los objetivos y alcances que se busca cumplir. Dando una breve introducción a lo que representa el trabajo presentado en este proyecto de tesis.

El segundo capítulo, Antecedentes y Marco Teórico, ofrece el panorama en el cual se encuentra involucrado el desarrollo del presente trabajo, así como los conceptos y definiciones que darán soporte al trabajo. Además de dar un contexto mas específico para cada uno de los módulos de desarrollo que se abarcarán en los capítulos consecuentes

El tercer capítulo, Diseño y Construcción de la Plataforma Robótica Móvil, se describe cada por categorías: mecánica, eléctrica-electrónica y .<sup>a</sup> nivel de software] los componentes, describiendo a detalle sus características, necesarios para desarrollar un plataforma robótica móvil que cumpla con el estándar propuesto en el presente trabajo e introducción durante el capítulo de Marco teórico.

El el cuarto capítulo, Programación de la Plataforma Robótica Móvil, se describe la estructura general de la arquitectura implementada en el robot así como cada uno de los nodos con los que se controla, comunica y obtiene información del hardware instalado sobre la plataforma móvil, así como ofrecer alternativas para el manejo de diferentes configuraciones en función de las características de los componentes.

En el quinto capítulo, Algoritmos para pruebas de desempeño, se describen algunos de los comportamientos mas recurrentes en el manejo de plataformas robóticas móviles con las cuales se pueden realizar un análisis de

la eficiencia del hardware y el software propuestos en el presente trabajo de tesis.

En el sexto capítulo, Resultados, se exponen las pruebas que se realizaron, además de la información obtenida mediante la implementación del hardware y software del robot móvil; se realiza un análisis de los errores presentados junto con las propuestas de solución que se consideraron. Para finalizar el presente proyecto de tesis, en el último capítulo se exponen las conclusiones y trabajo a futuro.



# Capítulo 2

## Antecedentes y Marco Teórico

En el presente capítulo se plasmará los conceptos que servirán como la base del diseño y desarrollo de la plataforma robótica móvil además de proponer los criterios que deberá cumplir una vez que se encuentre lista para su funcionamiento.

Se comenzará con la definición de Open source con el objetivo de conocer los criterios actuales con los que se puede afirmar que un proyecto de cualquier índole pueda ser clasificado bajo esta licencia, después se describirá a detalle el concepto de Robot móvil autónomo así como las características que se deben de considerar al momento de diseñar y desarrollar un proyecto de esta rama de la robótica.: posteriormente se expondrán los proyectos de ROS, Arduino y Raspberry Pi, destacando sus objetivos, metas y las características de los productos que nos ofrecen cada uno de ellos. Finalmente se mostrará el estado de la técnica presente en el ámbito de la robótica móvil autónoma con la finalidad de conocer los requerimientos y las opciones que ofrecen actualmente las empresas y entidades de investigación en la rama de la robótica móvil autónoma.

## 2.1. El Concepto *Open Source*

El concepto de *Open Source* refiere a aquello que cualquier persona puede modificar y compartir debido a que su diseño se encuentra abierto al público en general. Dicho término se originó en el contexto del desarrollo de software donde se buscaba diseñar un enfoque en la creación de programas de computadora. En la actualidad el término *Open Source* designa a un conjunto más amplio de valores en lo que se puede denominar como *la filosofía open source*, esto es, que toma en cuenta proyectos, productos o iniciativas que adoptan los principios del libre intercambio, participación colaborativa, la creación dinámica de prototipos, transparencia, meritocracia y desarrollo orientado a la comunidad. (*multiple authors, 2019g*).

Algunas las ventajas que ofrece esta iniciativa son:

- *Control.* Esto refiere a que se tiene un mayor control y conocimiento de lo que se encuentra desarrollando, siendo capaz de corregir o personalizar el producto o programa en su beneficio, incluso si los interesados no son desarrolladores de software.
- *Seguridad.* Este punto es meramente subjetivo, debido a que dentro de la comunidad OpenSource hay quienes consideran este tipo de desarrollos más seguros y estables que algunos desarrollos de licencia de propietario. Esto, junto con el punto anterior, debido a que regularmente se puede encontrar personas trabajando en mejorar y arreglar errores que el desarrollador original no contempló inicialmente; incluso realizándolo en tiempos en los que el desarrollador original no pueda invertir.
- *Estabilidad.* En conjunto con lo mencionado anteriormente en la actualidad muchas empresas y desarrolladores optan en utilizar programas OpenSource como herramientas de soporte para actividades críticas o fundamentales en sus desarrollos debido a la naturaleza de esta iniciativa existe un riesgo mínimo en que dichas herramientas o soporte desaparezca paulatinamente o por que el desarrollador original dejó de trabajar en ese programa o de implementar nuevas características.
- *Entrenamiento.* Este concepto se enfoca en los desarrolladores jóvenes que al adentrarse en este tipo de proyectos puede ayudarlos a convertirse en mejores programadores debido al carácter público del los

programas, los cuales pueden ser estudiados durante su etapa académica. Incluso puede servir para que desarrolladores tanto veteranos como iniciados puedan aprender a localizar errores, compartirlos con la comunidad y por medio de la re alimentación aprender a evitar ese tipo de problemas en un futuro.

En sus inicios el concepto de *Open Source Hardware* se encontraba asociado únicamente como derivado de la iniciativa Open Source (OSI), por lo que se definió como cualquier objeto físico, eléctrico o mecánico cuyo diseño se encontrara disponible al público de tal manera que permitiera a cualquier persona el construir, modificar, distribuir y utilizar dicho objeto (*Seaman, 1998*).

Actualmente la *Open Source Hardware Association* nos presenta la definición en la cual el Hardware de Fuentes Abiertas u Open Source Hardware (OSHW) es aquel hardware cuyo diseño se encuentra disponible al público para que cualquier persona pueda estudiar, modificar, materializar, distribuir y vender, tanto el original como otros objetos basados en el diseño inicial.

Por lo tanto el término *Open Source Hardware* se puede utilizar para denominar artefactos tangibles, maquinas, dispositivos u otros objetos del mundo físico, cuyo diseño ha sido publicado de tal manera en que cualquier persona interesada pueda fabricar, modificar, distribuir y usar el objeto y su diseño en cuestión; el propósito de esta definición es de proveer una guía para el desarrollo y evaluación de licencias para hardware de fuentes abiertas. Es por ello que en términos de distribución de debe cumplir con los siguientes criterios:

- *Documentación.* El hardware libre debe incluir documentación en forma de ficheros de diseño y deberá permitir modificación y redistribución de los mismos. En caso de que la documentación no acompañe al producto físico, deberá proporcionarse de manera clara el medio en que se pude obtener dicha información por no más que un razonable coste de reproducción, preferentemente por medio de una descarga de internet libre de cargo. La documentación deberá incluir los ficheros de diseño en un formato que permita introducir cambios, prohibiendo ficheros que intencionalmente oculten el diseño o substitutos en formato analógico alternativo al código informático compilado.
- *Alcance.* La documentación del hardware deberá especificar claramente que parte del diseño, sino todo, se libera bajo la licencia.

- *Programas informáticos necesarios.* En caso de que el diseño ajo licencia necesite un paquete informático, bien como parte del mismo o para operar de forma apropiada y cumplir sus funciones básicas, la licencia podrá requerir que se cumpliera alguna de las condiciones siguientes:
  - Que las interfaces harán de estar documentadas suficientemente como para considerar la posibilidad de crear un paquete en código abierto que permita al dispositivo operar de forma apropiada y cumplir sus funciones básicas.
  - Que el paquete informático necesario venga liberado bajo una licencia de código abierto aprobada por la OSI.
- *Obras derivadas.* La licencia deberá permitir modificaciones y obras derivadas, además de permitir la distribución de las mismas respetando los términos de licencia de la obra original. La licencia permitirá la fabricación, venta, distribución y uso de productos creados a partir de los archivos de diseño en sí mismos, y derivados de cualquiera de los anteriores.
- *Libre redistribución.* La licencia no podrá restringir a nadie de la venta o distribución de la documentación del proyecto. La licencia no podrá requerir el pago de derechos de autor por la mencionada venta. La licencia no podrá requerir ningún derecho de autor o tasa de venta de obras derivadas.
- *Atribución.* La licencia podría requerir que los documentos derivados y notificaciones de los derechos de copia asociadas con los dispositivos atribuyan la autoría del(os) autor(es) a la hora de distribuir ficheros de diseño, bienes manufacturados y/o productos derivados de los mismos. La licencia podría requerir que esta información se hiciera accesibles al usuario final utilizando el dispositivo, pero no podrá especificar el formato en se que muestre. La licencia podría requerir que las obras derivadas llevasen un nombre o número de versión distinto al del diseño original.
- *No discriminación a personas o grupos.* La licencia no puede discriminar ninguna persona o grupo de personas.

- *No discriminación a campos de aplicación.* La licencia no puede restringir a nadie de hacer uso del trabajo (incluyendo el objeto manufac-turado) en un campo específico de aplicación.
- *Distribución de la licencia.* Los derechos proporcionados por la licencia deberán ser aplicados a todos aquellos a los que sea redistribuido el trabajo sin la necesidad de ejecutar una licencia adicional.
- *La licencia no será específica a un producto.* Los derechos proporciona-dos por la licencia no dependen de que el trabajo licenciado sea parte de un producto determinado, es decir, si una parte de una obra licencia-da se usa y distribuye bajo los términos de la licencia, todos aquellos a los que se les redistribuya deberán tener los mismos derechos que proporcione la obra original.
- *La licencia no deberá restringir otro hardware o software.* La licencia no deberá colocar restricciones a elementos añadidos a la obra con el trabajo licenciado pero no derivados del el.
- *La licencia será neutra en términos tecnológicos.* Ninguna de las clausu-las de la licencia dependerá de una tecnología específica, componente, material o estilo de interfase o uso de la misma.

(Association, 2019)

## 2.2. Concepto de Robot Móvil Autónomo

La palabra robot tiene su origen en el vocablo checo *robo*ta el cual fue utilizado por primera vez en el año 1921 en la obra teatral "Rossum's Universal Robots" (R.U.R.) del autor checo Karel Capek; el significado de esta palabra en el idioma checo significa fuerza del trabajo o servidumbre (*Ollero Batutone, 2001*). Pese a que en el contexto de la obra dramática está palabra se utilizaba para referirse a humanos artificiales orgánicos, su uso fue asimilado más para referirse humanos artificiales mecánicos.

En la actualidad puede definirse un robot en función de su propósito y composición, sin embargo todos los robots creados comparten ciertas características fundamentales, por lo cual que podemos generalizar que un robot es aquel dispositivo mecánico o electrónico que está diseñado y programado para automatizar un proceso, con o sin supervisión humana, en el cual éste es controlado por un procesador y no necesariamente cuenta con una forma humanoide o busque asemejarse a un ser vivo, esto último se encuentra en función de su directiva (*Succar, 2018*).

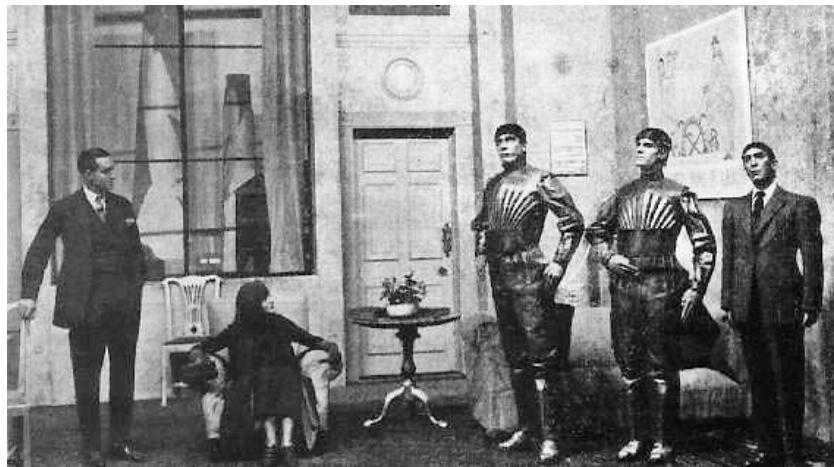


Figura 2.1: Un hombre, una mujer y tres robots, escena de la puesta en escena de la obra teatral *R.U.R.* de *Karel Čapek*.

Un robot se encuentra integrado por componentes mecánicos, eléctricos, electrónicos, de comunicación y de control; por lo cual suelen definirse tres secciones fundamentales para englobar ese conjunto de componentes: sección mecánica, sección eléctrica-electrónica y la sección de control y comunicación.

Conceptualmente un robot puede ser dividido en 3 sistemas en función de la implementación de los componentes que integran físicamente al robot: el sistema de percepción capaz de percibir el entorno y suministrar información del mismo, el sistema de control el cual debe ser capaz de realizar tomas de decisiones con base en el análisis de la información recolectada por el sistema de percepción para la posterior ejecución de una serie de directivas que guiarán el comportamiento del robot, y un sistema de actuadores que varían dependiendo del propósito del robot (*Ollero Baturone, 2001*).

El desarrollo de sistemas de percepción en robótica surge a partir de los progresos tecnológicos en sensores tales como los de visión, tacto, e incluso audición, sin embargo la percepción no solo involucra la captación de la información sensorial, sino también su tratamiento e interpretación. Por lo tanto, es necesario realizar una abstracción a partir de un cierto conocimiento previo de su entorno; es un hecho claro que la complejidad de la percepción artificial depende de lo estructurado que esté dicho entorno. El sistema de control es aquel encargado de analizar la información sensorial y realizar una planificación a partir de una serie de toma de decisiones ya predeterminadas y bucles de realimentación de la información suministrada, para poder realizar las acciones deseadas con los actuadores disponibles.

A lo largo de la historia los robots han mostrado una eficiencia a la hora de desarrollar algunas tareas y procesos que lleva a cabo el ser humano comenzando con prototipos sencillos hasta la fabricación de algunos autómatas cada vez más complejos según las ventajas que presentaran, encontrándose actualmente en diferentes sectores de producción industrial, investigación y servicios al ser humano.

El concepto y desarrollo de los robots móviles ha ido ligado con la necesidad de extender el campo de aplicación de la Robótica, además de tratar de incrementar la autonomía limitando todo lo posible la intervención humana.

Estos orígenes parten de la robótica industrial específicamente hablando de las tareas de manipulación, diferenciándose los robots móviles en que, a diferencia de los robots manipuladores, estos no se encuentran anclados en un sitio específico para la ejecución de sus labores. En cambio cuentan con la autonomía de movimiento, dicha autonomía se basa en un sistema de navegación automática, donde se incluyen tareas de planificación, percepción y control.

Una breve cronología de la historia de los robots móvil nos permite observar que desde los años 30 se tomaron como base los dispositivos electromecánicos para desarrollar funciones inteligentes como descifrar laberintos; uno

de los ejemplos a destacar sobre esta época es la tortuga Walter presentada en 1948, que podía reaccionar ante la presencia de obstáculos, subir pendientes y cuando la alimentación comenzaba a ser insuficiente podía dirigirse a una toma de corriente. En los años 60 se vuelve a trabajar en el desarrollo de robots móviles dotados de una mayor autonomía. La mayor parte de los prototipos se desarrollan empleando plataformas que soportan sistemas de visión, sin embargo, estos prototipos aún no podían navegar de manera autónoma con eficiencia. En los años 80 viene un incremento en la capacidad de procesamiento computacional y un mejor desarrollo de sensores, mecanismos y sistemas de control que dotan de una mayor autonomía a los nuevos prototipos de robots móviles (*Ollero Baturone, 2001*).

### 2.3. Esquema General de un Robot Móvil

Como ya definió previamente, un robot se compone de tres secciones principales: un sistema de percepción que le permiten observar su entorno, un sistema de control y un conjunto de actuadores, siendo estos los elementos básicos para cerrar la cadena de actuación-medidas-actuación.

Desde el punto de vista del procesamiento de la información, en la robótica se involucran funciones de control de percepción, planificación y movimiento ya que el sistema percepción es un conjunto de sensores internos y externos que nos permiten dotar de sentidos al robot suministrando información con la finalidad de aprender la realidad del entorno; estos sistemas hacen posible que el robot pueda adaptar automáticamente su comportamiento en función de las variaciones que se producen en su entorno, haciendo frente a situaciones imprevistas. El sistema de control involucra bucles de realimentación de la información sensorial suministrada por los sensores internos y los sensores, generando automáticamente acciones en función de la comparación de dicha información sensorial con patrones de referencia, además de que se encargan de la toma de decisiones del robot (*Ollero Baturone (2001)*).

En caso de los robots móviles la planificación tiene como objetivo encontrar una trayectoria desde una posición inicial a una posición objetivo, sin colisiones y minimizando un determinado índice de error; desde el punto de vista más simple el problema se plantea en un entorno que se supone conocido (existencia de un mapa previo ya definido en el sistema) y estático, que puede moverse en cualquier dirección, que se mueve suficientemente lento y que es capaz de seguir el camino de forma perfecta (*Ollero Baturone (2001)*).

En los robots móviles, el problema de navegación abarca la planificación global de la tarea, es decir, elegir una ruta adecuada, recorrer una trayectoria planeada y finalmente evitar obstáculos no esperados. En un robot móvil diseñado para interiores (como una sala o un departamento), la tarea consiste en determinar a qué habitación o lugar en específico hay que desplazarse, mientras que la ruta establecería el camino desde la posición inicial hasta una posición final, definiendo puntos intermedios de paso, y durante la trayectoria, evadir obstáculos no esperados (como una caja o una persona). Durante la ejecución de la tarea surgen algunos inconvenientes, el robot móvil puede desviarse de la trayectoria establecida debido a la acumulación de imprecisiones mecánicas y de control, esto hace que sea más difícil llegar a la posición final. La corrección de la acumulación de errores hace necesario el empleo de otros sensores (como los encoders en los motores), que nos informan qué tanto se ha desplazado el robot de acuerdo a la distancia que se le indicó recorrer, y si hay fallas, se compensan en ese momento. *Ollero Baturone (2001)*

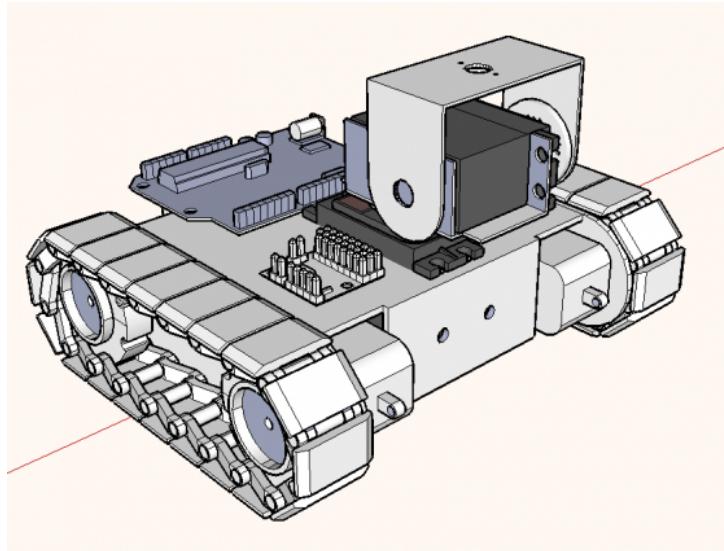


Figura 2.2: Robot móvil autónomo.

## 2.4. Sistema Operativo ROS

ROS, Robot Operating System por sus siglas en inglés o Sistema Operativo Robótico, surgió en base al planteamiento en el que se buscaba desarrollar software robusto y de propósito general para la robótica; en donde desde la perspectiva del robot los problemas que al entender humano parecen triviales llegan a variar considerablemente entre las instancias de tareas y entorno, por lo que al considerar las variaciones posibles es una tarea difícil para un individuo, laboratorio o institución. Como resultado ROS surgió de un trabajo en conjunto de desarrollo de robótica *ROS.org* (2019a).

En palabras de la organización ROS.org, el sistema operativo robótico ROS es una es un marco de trabajo flexible para el desarrollo de software para robótica, ofreciendo una amplia colección de herramientas, bibliotecas y convenciones que tiene como objetivo simplificar el proceso de desarrollo de comportamientos robóticos robustos y complejos implementados en una variedad de plataformas robóticas *ROS.org* (2019a).



Figura 2.3: Esquema de trabajo de ROS.

ROS ofrece una infraestructura de comunicación en la cual consiste en un middleware de comunicación entre procesos, este middleware provee de las siguientes facilidades:

- Paso de mensajes. Tomando en cuenta la importancia de un sistema de comunicación en el momento de implementar un sistema robótico. ROS ofrece la capacidad de comunicación entre nodos distribuidos por medio de la mecánica de suscribirse y publicar de manera anónima. Entre las ventajas que este tipo de mecánicas ofrecen es la de implementar interfaces concisas entre los nodos del sistema desarrollado, mejorando la encapsulación y la reutilización de código. Este tipo de comunicación se lleva a cabo por medio de canales denominados *tópicos*.
- Registro y reproducción de mensajes. Debido a que es sistema de publicar y suscribirse es de naturaleza anónima y asíncrona, los datos

transmitidos pueden ser fácilmente capturados y reproducidos sin algún cambio en el código, es por eso que se desarrolló la opción de que ROS capture la información transmitida por un proceso en un archivo para que posteriormente dicha información sea reproducida del mismo archivo por otro proceso. La abstracción del middleware para el paso de mensajes permite a cualquier proceso ser indiferente ante el origen de la información, ya sea transmitido desde otro proceso o bien que pertenezca a un archivo en cuestión; favoreciendo la reducción de esfuerzo en el desarrollo de un sistema y promoviendo la flexibilidad y modularidad del mismo.

- Llamadas a procedimiento remoto. Dada la naturaleza asíncrona de la publicación y suscripción de la mensajería funciona para una variedad de necesidades de comunicación en robótica, sin embargo al momento de requerir comunicación petición/respuesta síncrona entre procesos ROS ofrece la capacidad de cumplir ese requerimiento en la interfase de comunicación por medio del uso de servicios, que al igual que los tópicos la información enviada entre procesos siendo convocada por un servicio puede ser definida usando un simple mensaje IDL.
- Sistema de parámetros distribuidos. EL middleware de ROS también ofrece una manera en que los procesos comparten información de configuración a través de un almacén global de clave-valor; por lo que el sistema de permite fácilmente modificar la configuración de los procesos, incluso permitiendo a los procesos cambiar la configuración de otros procesos.

Adicionalmente al los componentes ofrecidos por el middleware previamente descritos, Ros provee bibliotecas y herramientas comunes específicas de robots con los que se podrá obtener un desempeño fluido. Entre lo que nos ofrece encontramos

- *Mensajería de robot estándar.*

Producto de un amplio debate entre desarrolladores se estableció un formato estándar de mensajes los cuales cumplen la mayoría de los casos de uso comunes en robótica. El contenido de estos mensajes son definiciones para conceptos geométricos tales como posiciones, transformaciones y vectores; para el uso de sensores, cámaras, unidades de

medición inercial (IMU por su siglas en inglés), telémetros láser; y para el campo de navegación cuenta con datos de odometría, rutas y mapas, Al usar la mensajería estándar se garantiza que el código desarrollado interopere perfectamente con el resto del ecosistema de ROS.

- *Biblioteca de la geometría del robot.*

Uno de los retos más importantes en el campo de la robótica es el mantener un seguimiento de la ubicación espacial de cada una de las partes del robot respecto a las demás, siendo un problema de considerable importancia en robots humanoides con múltiples partes móviles. Por lo que en pos de resolver este tipo de problemas en ROS se implementó la biblioteca **tr** (**trasnform**) la cual realiza un seguimiento de la localización de cada uno de los elementos en el sistema del robot; la librería permite gestionar datos de transformación de coordenadas para robots con más de cien grados de libertad y actualizar a ritmos de cien Hertz; la librería permite al usuario definir transformaciones estáticas y dinámicas, o bien transformar información obtenida por los sensores en cualquier marco de coordenadas del sistema.

- *Lenguaje de descripción del robot.*

Otro de los problemas comunes que se ha buscado solucionar con el uso de ROS es en contar con un procedimiento adecuado para la descripción de un robot de una forma legible por máquina. Para este problema ROS provee un conjunto de herramientas para la descripción y modelado de un robot en específico y con el cual pueda ser comprendido por el resto del sistema de ROS, en donde se encuentran incluidas herramientas tales como la anterior mencionada **biblioteca tf**, la **paquetería robot\_state\_publisher** y **rviz**. El formato con el que se describe el robot en ROS es el *Formato de descripción de robot unificado (URDF por sus siglas en inglés)* el cual consiste en un documento tipo XML en el cual se describen las propiedades físicas de un robot, tomando en cuenta las medidas de cada miembro y el tamaño de las ruedas, las localizaciones de los sensores y la apariencia visual de cada parte del robot.

Una vez definida la representación del robot, se puede generar un modelo tridimensional por medio del uso de la biblioteca **tf** para ser usado en simuladores y planeadores de acciones.

- *Llamadas de procedimiento remoto preferibles.*

Mientras los tópicos y servicios cubren la mayoría de las necesidades de comunicación en robótica, en algunos casos se requiere iniciar un comportamiento de búsqueda de objetivos, monitorizar el progreso, ser capaz de adelantarse en el camino y recibir notificaciones cuando el proceso se complete. ROS provee opciones para esos propósitos, estas opciones son similares a los servicios con excepción de que pueden reportar el progreso antes de mostrar el resultado final, y pueden ser priorizadas por el que proceso que las manda a llamar.

- *Diagnósticos.*

ROS provee de un procedimiento estandarizado para producir, recolectar y agregar diagnósticos sobre el robot, los cuales nos presentan el estado de robot de manera visualmente sencilla y con la cual podemos abordar los problemas a medida que surgen.

- *Estimación de la posición, Localización y Navegación.*

Ros además provee capacidades para facilitar el iniciar con un proyecto de robótica. Estos son paquetes de ROS los cuales resuelven problemas sencillos de robótica tales como:

- Estimación de la posición del robot.
- Localización de un robot en un mapa.
- Creación de un mapa para navegación.
- Navegación móvil

Uno de los aspectos más robustos de ROS es el conjunto de herramientas fuertemente desarrollado, estas herramientas admiten introspección, depuración, trazado y visualización del estado del sistema que se está desarrollando. La subyacente mecánica de publicación y suscripción permite espontáneamente revisar el flujo de datos que recorre el sistema, facilitando la fácil compresión y depuración de problemas tan pronto se presenten. Las herramientas de ROS toman ventaja de su capacidad de introspección frente una amplia variedad de herramientas gráficas y líneas de comando que simplifican el desarrollo y depuración. Entre las herramientas más importantes encontramos:

- **textbf{Herramientas de líneas de comando}**: Estas herramientas están enfocadas principalmente en el uso de un robot el cual no requiere necesariamente de una interfaz gráfica de usuario para su funcionamiento y uso; la funcionalidad principal y las herramientas de introspección son accesibles por medio de más de 45 herramientas de linea de comandos. Entre estas opciones se encuentran comandos para inicializar un conjunto de nodos, introspección de tópicos, servicios y acciones, registro y reproducción de datas, entre otros.
- **rviz**: Seguramente una de las herramientas más conocidas de ROS, **rviz** provee visualizaciones tridimensionales de propósito general de un grupo amplio de datos de sensores de diferente tipo y soporte a cualquier representación visual de un robot, tal y como se describió anteriormente.

**rviz** tiene la capacidad de visualizar los diferentes tipos de mensaje presentes en ROS, tales como el escaneo por láser, nubes de puntos tridimensionales e imágenes de cámaras. Además de utilizar la información de la biblioteca *tf* para mostrar en conjunto la información de los sensores con el modelo renderizado del robot.

- **rqt**: ROS provee un marco de trabajo e basados en *QT* para el desarrollo de interfaces gráficas para el robot. Con lo cual se pueden diseñar interfaces personalizadas por medio de componer y configurar la extensa biblioteca de complementos **rqt** incorporados en tabulaciones, pantallas divididas y otros diseños, también puede introducir nuevos componentes de interfaz escribiendo sus propios complementos de rqt. Entre las principales funciones que nos ofrece esta herramienta encontramos:

- **rqt\_graph**. Este complemento provee introspección y visualización en tiempo real del sistema de ros, mostrándonos los nodos activos y las conexiones entre ellos, y permitiendo una depuración simple y un entendimiento del sistema en marcha y como se encuentra estructurado.
- **rqt\_plot**. Este complemento permite estar constantemente monitorizando encoders, voltajes o cualquier elemento que pueda ser representados como variables respecto al tiempo. Permite también

seleccionar el tipo de gráfico que se adecue a las necesidades del usuario.

- rqt\_topic & rqt\_publisher. Este complemento tiene como objetivo facilitar el monitoreo y uso de los tópico, permitiendo monitorizar y realizar introspección del numero de tópicos que se encuentra publicando en el sistema. Además de permitir publicar mensajes dentro de cualquier tópico disponible, facilitando la experimentación en el sistema,
- rqt\_bag. Este complemento tiene como objetivo el almacenaje y reproducción de información obtenida usando ROS el formato *bag*; este tipo de archivos pueden ser creados y accesar a ellos gráficamente. El complemento permite registrar los datos en *bolsas*, y reproducir los tópicos seleccionados a partir de la *bolsa*, y visualizando el contenido de dicha *bolsa*, incluyendo imágenes o gráficos de algún valor a través del tiempo.

*ROS.org (2019d)*

Los conceptos descritos este aparatoado serán explorados e implementados a profundidad durante el desarrollo del capítulo 4 y el capítulo 5 del presente trabajo.

## 2.5. Arduino

Arduino es un plataforma electrónica de tipo OpenSource basada en el concepto de un uso sencillo del hardware y software; cuyo propósito es ofrecer una plataforma confiable y sencilla para prototipado de software y hardware. El lenguaje de programación de Arduino se encuentra basado en "Wiring", mientras que su entorno de desarrollo (IDE) en el entorno Processing (*multiple authors, 2019a*).

Algunas de las características destacables de la plataforma Arduino son:

- Bajo Costo. Debido a que la plataforma Arduino es relativamente mas barata a comparación de otras plataformas de microcontroladores; además de que el usuario interesado cuenta con los diseños con el que puede ensamblar su propia plataforma Arduino o bien comprar una plataforma Arduino pre-ensamblada con un costo bajo.
- Multiplataforma. El entorno de trabajo que provee la plataforma Arduino puede ejecutarse en diferentes plataformas (Windows, Linux, Mac) mientras que muchas plataformas de microcontroladores se encuentran en la plataforma Windows.
- Entorno de desarrollo claro y simple. El entorno de desarrollo de Arduino (IDE) es adecuado y sencillo para utilizar para desarrolladores principiantes pero con un alcance y flexibilidad para que desarrolladores veteranos encuentren adecuado el entorno de trabajo.
- Software Open Source. El software Arduino se encuentra desarrollado y publicado como un conjunto de herramientas de tipo Open Source, el cual se encuentra disponible gracias al desarrollo de programadores con mayor experiencia; además de que el software Arduino puede ser expandido por medio de bibliotecas desarrolladas con el lenguaje C++ y del desarrollo técnico por medio del lenguaje de programación AVR C (también conocido como ensamblador) para su implementación sobre el hardware que el Arduino tiene implementado. Esto conlleva que también cualquier interesado puede implementar su proyecto por medio del lenguaje de programación AVR C.
- Hardware Open Source. Como parte de la iniciativa OpenSource a la que pertenece Arduino, los diseños y planos de los circuitos de las plataformas de prototipado se encuentran disponibles en el sitio oficial

por medio de la licencia de Creative Commons. Con estos planos expertos en circuito o personas interesadas pueden desarrollar su propia plataforma de desarrollo, extendiéndola o mejorando en función de sus necesidades. El sitio incluso ofrece una opción para que los nuevos interesados puedan desarrollar su tablero de circuitos con una lista de las especificaciones de cada uno de los componentes necesarios y con el cual pueden entender el funcionamiento de la plataforma y el modo para ahorrar dinero.

*(multiple authors, 2019f).*

## 2.6. Raspberry

La Fundación Raspberry Pi, es una organización caritativa de origen inglés que tiene como objetivo el poder ofrecer las facilidades del desarrollo digital a todo el mundo, para que estos sean capaces de comprender y moldear este mundo digital que se encuentra en desarrollo, desarrollando sus capacidades de resolver una variedad de problemas y estar preparados para trabajar en el futuro sean cuales sean las oportunidades y necesidades que se les presenten (*multiple authors, 2019e*).

Este objetivo lo promueven mediante el desarrollo de computadoras de alto desempeño y bajo costo en las cuales las personas de todas las edades no acostumbradas al uso de tecnología puedan aprender y resolver problemas mientras se divierten; complementándolo con el desarrollo de planes educativos con los cuales la gente puede irse integrando poco a poco al computo y diseño digital, estos programas se caracterizan por ser de índole Open Source (mencionado anteriormente en este capítulo en el apartado 2.1) y se encuentran integrados por manuales, tutoriales, y algunos experimentos que puede recrearse fácilmente logrando que el usuario se familiarice con el uso de un equipo de cómputo y el desarrollo de actividades en éste, ya sea tanto como para un usuario que apenas va iniciando en este mundo digital como a interesados en el tema o docentes de cualquier nivel educativo.

Siendo su mayor desarrollo para cumplir con este objetivo los diferentes modelos de la tarjeta Raspberry Pi, la cual se define como un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple, esto refiere a un computadora completa en un solo circuito y cuyo diseño se centra en un sólo microprocesador con la memoria RAM, dispositivos de E/S y demás características de un computador funcional integrado en la placa base que suele ser generalmente de tamaño reducido. Raspberry Pi nunca ha declarado expresamente que su tarjeta sea de Open Hardware siendo ese un producto de marca registrada; al contrario del software el cual si es distribuido como Open Source, siendo su sistema operativo mas utilizado una versión adaptada de Debian conocido como Raspbian, aunque la tarjeta permite el uso de otros sistemas operativos como Windows 10 o Ubuntu (*multiple authors, 2019d*).

Entre las características generales de cada una de los modelos desarrollados por la fundación Raspberry Pi encontramos:

- Procesador Broadcom con arquitectura ARM.

- Memoria RAM, GPU; ambas variables dependiendo el modelo.
- Puertos USB, HDMI, Ethernet (salvo el primer modelo).
- Pines de Entrada/Salida de Propósito General o General Purpose Input/Output (GPIO).
- Conector para una cámara.
- Puerto para memoria SD o MicroSD dependiendo el modelo.

Para la creación de la tarjeta Raspberry Pi los diseños se basaron en el microcontrolador Atmel ATmega663 al rededor del año 2006, pero no fue hasta el año 2009 que la Fundación Raspberry Pi fue fundada en Caldecote South Cambridgeshire. Reino Unido como una asociación caritativa que es regulada por la comisión de caridad de Inglaterra y Gales. Durante sus primeros desarrollos el administrador de la fundación Eben Upton contacto a un grupo de académicos y entusiastas de la informática para crear un ordenador con la intención de animar a los niños a aprender informática textit(multiple authors, 2019c).

## 2.7. Estado de la técnica

En la actualidad podemos encontrar una amplia variedad de robots comerciales que buscan cumplir con las necesidades del diferentes mercados y aplicaciones que van del campo de la robótica educativa hasta los robots de servicio. Cada uno cumpliendo con diferentes requerimientos, en donde el diseño y características se encuentran en función de la aplicación y el entorno en el que el robot se va a desempeñar. Entre los robots más relevantes a nivel mundial que cumplen con requerimientos similares a los planteados en el objetivo del presente trabajo, encontramos:

- **TurtleBot**

El robot TurtleBot es un kit robótico personal de bajo costo que implementa open source software, creado por Melonee Wise y Tully Foote en Noviembre del 2010. El objetivo principal es el ofrecer la posibilidad de construir un robot que pueda interactuar en el interior de una casa, visualizar su entorno en 3D y tener el suficiente potencial mecánico para ejecutar algunas aplicaciones. El TurtleBot consiste en una base móvil, sensor de distancia 2D o 3D dependiendo del modelo, una computadora laptop o un ordenador de placa reducida (SBC por sus siglas en inglés), y un kit de hardware para el montaje de los elementos; en adición los usuarios pueden descargar el kit de desarrollo desde la página oficial de ROS. El principal objetivo TurtleBot es su sencillo diseño, fácil de armar y bajo costo, seleccionando productos comerciales o partes que fácilmente pueden ser manufacturados con materiales estándar (*TurtleBot, 2019*).

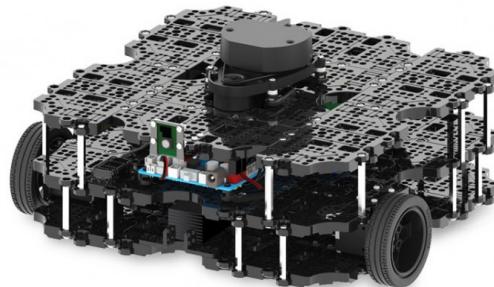


Figura 2.4: Robot TurtleBot modelo 3.

- ***PeopleBot***

El robot PeopleBot es una plataforma robótica móvil de tipo diferencial de alta calidad diseñada para el proyectos de interacción y servicio a los seres humanos. En términos de hardware el robo t cuenta con un chasis elevado, una fuente de alimentación disponible, puertos de entrada y salida de fácil acceso lo cual permite que la plataforma sea eficiente en el uso de sensores y actuadores interactivos. La plataforma robótica PeopleBot cuenta con 10 elementos de parachoques, arreglos de sensores tipo SONAR inferiores y superiores y haces de interrupción IR para detectar objetos entre la plataforma elevada y el robot; con el software opcional *Laser Mapping & Navigation System* y *MobileEyes*, PeopleBot puede mapear edificios y localizarse a unos pocos centímetros mientras viaja dentro de las áreas mapeadas. Con los accesorios adecuados, puede de ver la vista del robot de forma remota, hablar, reproducir y escuchar audio, y enviar el robot en patrulla (*Robots, 2011*).



Figura 2.5: Robot PeopleBot

- ***LEGO***

*Lego Mindstorm* es la división enfocada en robótica para niños la cual busca enseñar de una manera práctica y sencilla los elementos básicos de las teorías robóticas, como la unión de piezas y la programación de acciones de forma sencilla e interactiva. El concepto surgió con la idea de hacer accesible el desarrollo de robots móviles para el público en general con un costo bajo (aproximadamente \$200US) y con acceso a una computadora personal de escritorio (*Knudsen y Loukides, 1999*).

Actualemente encontramos la versión LEGO MINDSTORM EV3, el cual tiene como objetivo el crear, programar y manejar un robot LEGO de diseño personalizado de manera sencilla, inteligente y divertida. Cuenta con una aplicación de computadora conocida como *app EV3 Programmer* que incluye intrucciones de construcción, misiones de programación las herramientas necesarioas para programar los robots diseñados por el usuario, incentivando la creatividad de este (*LEGO, 2019*).

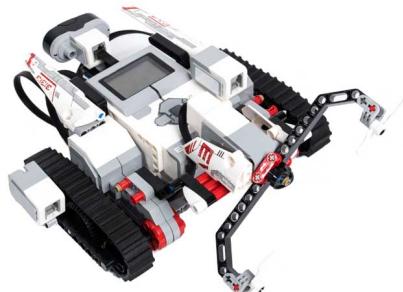


Figura 2.6: Mindstorm EV3

Hablando a nivel nacional, nuestro país siguiendo las tendencias del incremento en la inversión de la investigación y desarrollo de la robótica, ha aumentado el interés en la aplicación de la robótica pasando a ser uno de los principales compradores a nivel mundial. Es por esto que investigadores mexicanos han desarrollado diferentes prototipo que permiten apreciar el potencial que tiene la nación en el diseño y construcción de robots.

Entre los ejemplos más destacables encontramos:

- **Robot Golem:** es un robot de servicio diseñado y construido por el Grupo Golem que forma parte del Departamento de Ciencias de la Computación del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) de la Universidad Nacional Autónoma de México (UNAM). El robot es capaz de mantener una conversación sencilla y entender comandos sencillos de movimiento; la versión *Golem II+* es apto para interpretar expresiones lingüísticas y visuales al punto de ofrecer una visita guiada en el Departamento de Ciencias de la Computación. Finalmente la versión *Golem III* se encuentra orientado

al lenguaje y la comunicación, este robot piensa, habla y tiene la capacidad de recibir comandos, sostener una conversación en lenguaje hablado con humanos, además de hacer diagnósticos de su entorno, tomar decisiones y resolver problemas sencillos de la vida cotidiana. *Golem III* consta del siguiente hardware: un arreglo de sonares con 8 sensores, base PatrolBot, dos arreglos protectores cada uno con 5 sensores al tacto, bocinas bidireccionales Infinity de 3.5 pulgadas, computadora Cobra EBX-12, láserSick LMS-500, laptop Dell Precision M7510, láser localizador Hokuyo SOKUIKI, switch ethernet Black Box alimentado por USB con 5 puertos, cámara Microsoft Kinect 2, cámara de alta resolución Point Grey Flea USB 3, interfaz de audio 8SoundsUSB, tres micrófonos miniatura, micrófono direccional RODE VideoMic, torso, brazos y cuello construidos dentro del grupo.

- **Robot Markovito:** diseñado y construido por el grupo Markovito perteneciente al Laboratorio de Robótica de la Coordinación de Computacionales del Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE). es un robot basado plataforma PeopleBot, tiene las capacidades de ver, escuchar e interactuar con humanos en un entorno de la vida real; la función principal es asistir en diferentes labores a personas de la tercera edad, discapacitadas y enfermas. Este robot cuenta con sensores, cámaras, tres computadoras, dos controladores Kinect, micrófonos, bocinas y un brazo robótico. De esta manera, Markovito puede calcular distancias, reconocer objetos, detectar obstáculos, manipular objetos, así como interactuar con usuarios.
- **Robot Justina:** robot diseñado y construido un grupo integrado por investigadores y estudiantes de la Facultad de Ingeniería, del posgrado de Ingeniería Eléctrica de la UNAM, área de procesamiento de señales y del Posgrado de Ciencias e Ingeniería en Computación de la UNAM. También participan profesores de la Facultad de Ingeniería y del IIMAS de la UNAM, así como colaboradores que se encuentran en el extranjero. El robot Justina se trata de un robot en constante evolución, ya que, durante más de 12 años, se ha mejorado el software, sin embargo, el hardware se ha mantenido con pocos cambios. Uno de los elementos más interesantes de Justina es que puede reconocer personas e interactuar con ellas, como la posibilidad de recibir y seguir órdenes. Una de las claves para el reconocimiento facial es una cámara Kinect.

El sistema de operación de Justina 9 está basado en una arquitectura híbrida de robots móviles, que consiste en cuatro capas: Entrada, Planeación, Conocimiento y Ejecución, cada una de ellas cuenta a su vez con varios subsistemas que controlan en forma general la operación del robot. Operativamente, los componentes de software se comunican entre sí usando el mecanismo de comunicación entre procesos del sistema operativo para robots ROS.

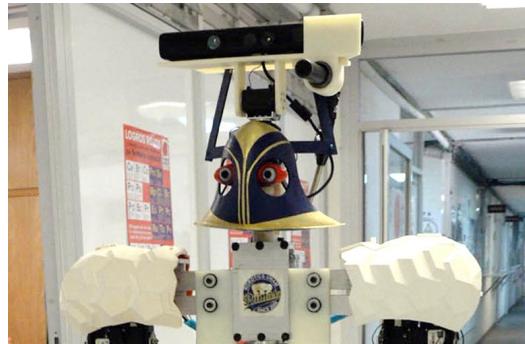


Figura 2.7: Robot Justina de la UNAM.

(Succar, 2018)

## Capítulo 3

# Diseño y Construcción de la Plataforma Robótica Móvil

En el presente capítulo se plantearán, en base a las definiciones descritas durante el capítulo 2, los sistemas, componentes y materiales que cumplen con las características para desarrollar una plataforma robótica móvil completamente funcional. Se comenzará exponiendo los diferentes sistemas en los cuales se dividirá la plataforma robótica para tener un claro control del hardware y software necesarios; posteriormente se describirá a detalle los componentes que actualmente se encuentra en el mercado así como las razones por las cuales se escogieron, separados en distintos sistemas que se describirán más adelante y que categorizan a sus componentes según la naturaleza o el propósito de éstos.

Finalmente se dispondrá de la descripción del proceso de ensamblaje de la plataforma robótica, que se podrá consultar en el anexo D y el listado completo de componentes necesarios en el anexo B al final del presente trabajo.

### 3.1. Estructura General del Diseño de la Plataforma Robótica

Retomando lo descrito en el capítulo 2 durante el apartado Esquema general de un Robot Móvil, podemos clasificar los componentes que integran a un robot ya sea por el sistema al que pertenezcan o por sus características, y basándonos en esa definición se expone a continuación un diagrama sencillo exemplificando la forma en que se catalogaran dichos cada uno de los componentes a lo largo del trabajo.

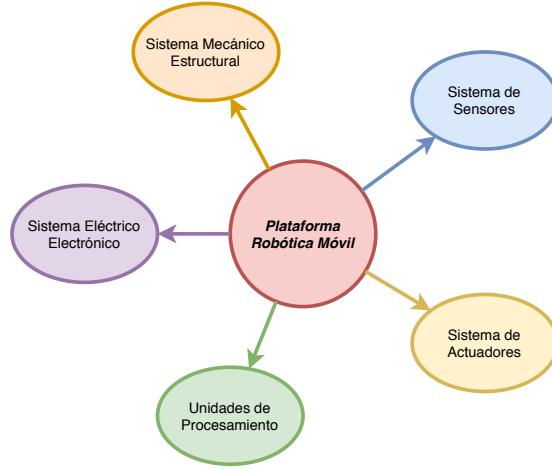


Figura 3.1: Esquema general de un robot.

En la figura 3.1 se muestran los cinco sistemas principales en las que se decidió catalogar los componentes del robot. Comenzando con el sistema de percepción donde se encuentran la mayor parte de los sensores y componentes electrónicos, además de los nodos que se encargarán de registrar la información obtenida; el sistema de actuadores se encarga a grande rasgos del movimiento de la plataforma robótica y de aquellos componentes cuyo comportamiento se programa donde se incluyen los nodos que ejecutan dichas instrucciones. Las unidades de procesamiento se compone de los elementos en los que se implementará la programación de la arquitectura del robot así como encargarse de gestionar los dos sistemas anteriores. El sistema eléctrico electrónico describe el diseño del circuito eléctrico, alimentación eléctrica de la plataforma robótica, así como los elementos tanto pasivos y activos necesarios para un correcto desempeño de cada uno de los compo-

nentes que requieran suministro eléctrico. El sistema mecánico estructural es una descripción detallada del diseño físico de la plataforma robótica.

## 3.2. Sistema de Sensores

Como se mencionó anteriormente, este sistema es el encargado de obtener la información del entorno en el que se encuentra el robot y que sea capaz de interpretarlo. Hablando específicamente de una plataforma robótica móvil nos interesa saber información sobre los elementos a su alrededor con el fin de realizar un mapa aproximado del entorno, y con el cual pueden calcularse rutas y evitar obstáculos; se requerirá en algunos casos la capacidad de percibir estímulos externos para alcanzar un objetivo o trazar una trayectoria no calculada usando el mapa antes descrito; además, es importante conocer algunos factores internos de la plataforma robótica móvil que podrían afectar el rendimiento de la misma tales como temperatura interna del robot o en nivel de energía de la fuente de alimentación.

En la figura 3.2 se realiza un breve listado de los componentes necesarios para cubrir las necesidades antes descritas. Ordenándolos en diferentes secciones dadas sus características:

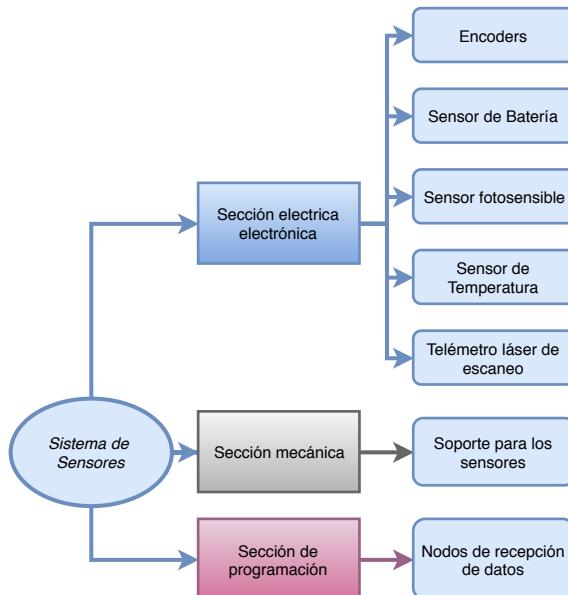


Figura 3.2: Elementos del sistema de sensores.

### Sensor de batería, Alarma de bajo voltaje para pilas LiPo

Durante el desarrollo de cualquier dispositivo electrónico uno de los principales elementos que se deben considerar respecta al suministro eléctrico; en el caso particular del presente trabajo se optó por el uso de una batería tipo LiPo la cual será descrita más adelante en la sección 3.5, por consecuencia se debe considerar un elemento que esté monitorizado constantemente nuestra fuente de suministro eléctrico para evitar que la batería o alguna de sus celdas se descargue a un nivel de voltaje que comprometa la integridad de la misma. La elección de este modelo sobre algunos más completos, aquellos que incluyen una serie de display 7 segmentos para mostrar el voltaje de la batería y son usados regularmente para drones de elaboración propia, es que inicialmente se tiene contemplado que exista un voltímetro colocado en una posición mas adecuada y visible para el usuario además de que este sea accionado solo cuando el usuario requiera conocer el valor del voltaje de la batería, en caso de contemplar una serie de pruebas o bien para considerar el recargar o cambiar la fuente de alimentación.

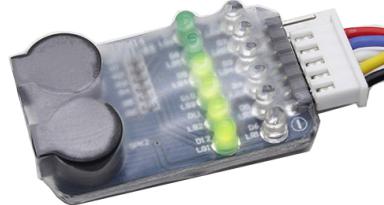


Figura 3.3: Alarma de bajo voltaje para pilas LiPo de 2S a 6S

Entre las ventajas que incluye el uso de este dispositivo se encuentran:

- La alarma LiPo es un componente pequeño, ligero y de fácil uso; debido a que basta con enchufar el conector al cable de balance de la batería LiPo para su funcionamiento.
- Peso de 9 g. y un tamaño de 38x35x11 mm.
- Es compatible con baterías LiPo de 2S a 6S.
- Tiene un consumo bajo de tensión, a diferencia de los componentes con display 7 segmentos los cuales tienen un consumo constante mas elevado de voltaje y tensión.

- Al momento de conectarse en el cable de balance comenzará inmediatamente con la detección de voltaje en las celdas individuales.
- Una alarma sonará y los indicadores LED cambiarán de color verde a rojo si alguna celda individual reduce su voltaje a 3.3 V o menos. Proporcionando así una señal mas eficiente para la reacción del usuario.

*Usando como referencia (HoobyKing, 2019)*

### Sensor fotosensible, TEMT6000 Ambient Light Sensor

La inclusión de este elemento entre los componentes de un robot móvil tiene como principal objetivo el probar de una manera sencilla los algoritmos de planeación y navegación de rutas con las que se podría calificar el desempeño del robot. Para este propósito se propondría un arreglo de al menos 4 de estos elementos para tener un rango de percepción adecuado.

En este apartado se optó por escoger los sensores de luz ambiental TEMT600 debido a lo práctico de su diseño y funcionamiento, ya que al funcionar bajo el principio de divisor de voltaje el manejarlo llega a resultar sencillo y casi intuitivo. El TEMT6000 es un foto-transistor de crecimiento epitaxial plano de tipo NPN, el cual se encuentra contenido en un molde transparente en miniatura para facilitar su montaje sobre un circuito impreso.



(a) Esquemático del Sensor (b) Sensor de luz ambiental  
TMMT6000. TMMT6000.

## 36 CAPÍTULO 3. DISEÑO Y CONSTRUCCIÓN DE LA PLATAFORMA

Entre sus características encontramos

- Se encuentra adaptado para tener respuesta similar al ojo humano, esto es, el dispositivo es sensible al espectro visible.
- Cuenta con un ángulo de percepción media de unos  $60^\circ$ .
- Registra la intensidad luminosa en lumenes (lux) y la traduce proporcionalmente a voltaje.
- Funciona con voltajes de entrada de entre 3.3 V y 5 V.

*Usando como referencia (VishaySemiconductors, 2019)*

### Sensor de temperatura, LM45 Texas Instruments

La importancia de considerar este elemento como parte de la plataforma robótica móvil recae en la necesidad de tener un control de las condiciones internas del robot para evitar que la electrónica se estropee, esto debido al calor generado ya sea por los actuadores mecánicos o bien por la temperatura del ambiente. En este apartado se hablará específicamente del Sensor de Temperatura LM35 de Texas Instruments debido a la versatilidad y fácil acceso que presenta este dispositivo. La serie de sensores LM45 son dispositivos de temperatura precisos y compuestos por un circuito integrado, el cual proporciona un voltaje de salida proporcionar con la temperatura en grados Centígrados del entorno; teniendo un comportamiento lineal en el cual cada grado Celsius equivale a 10 mV.

Por lo que las principales ventajas a destacar en el uso de este modelo serían:

- Está calibrado directamente en grados Celsius.
- La tensión de salida es proporcional a la temperatura, es decir, la salida tiene un comportamiento lineal con relación de  $1\text{ }^\circ\text{C} = 10\text{ mV}$ .
- Tiene una precisión garantizada de  $0.5\text{ }^\circ\text{C}$  a  $25\text{ }^\circ\text{C}$ .
- El rango de muestreo va de los  $-55\text{ }^\circ\text{C}$  a los  $150\text{ }^\circ\text{C}$ .
- Baja impedancia de salida.
- Cuenta con un rango de alimentación de 4 V a 20 V.

- Baja corriente de alimentación, cercana a los 60 [micro A], por lo que produce un efecto de auto calentamiento reducido.
- Bajo coste.
- No requiere de circuitos adicionales para su calibración.

*Usando como referencia (Texas Instruments, 2019)*



(a) Diagrama Esquemático del Sensor serie LM45. (b) Sensor de temperatura de la serie LM45 comercial.

Como se puede inferir, dadas las características mencionadas anteriormente es posible que este dispositivo sea instalado fácilmente en un circuito de control y poder ser instalado fácilmente en cualquier diseño que se pueda implementar en un robot móvil.

### Telémetro láser de escaneo RPLIDAR A1

El *escaner de alcance láser RPLIDAR A1*, es un telémetro láser cuyo diseño se basa en el principio de barrido de triangulación láser utilizando adquisición de la visión y hardware de procesamiento de alta velocidad desarrollado enteramente por la compañía Slamtec.



Figura 3.6: Telémetro láser de escaneo RPLIDAR A1 M8.

Entre las principales características que este modelo nos ofrece destacan:

- El modelo RPLIDAR mejora el diseño óptico y el sistema de algoritmos para obtener una frecuencia de muestreo de 8000 veces por segundo, además de una velocidad de escaneo configurable de 2 a 10 HZ, con la opción de ajustar la velocidad de exploración alternando la señal PWM del motor.
- El núcleo del RPLIDAR A1 gira en sentido de las agujas del reloj ofreciendo un escaneo de rango láser omnidireccional de 360 grados del entorno que lo rodea. Ofreciendo al usuario un mapa de contorno del medio ambiente.
- Cuenta con tecnología Plug & Play, por ende no es requerido ningún trabajo de codificación, basta con simplemente conectar el dispositivo a la computadora a través de un cable micro USB.
- Finalmente el RPLIDAR es ideal para la navegación y localización de robots, además de haber sido diseñado con el objetivo de aplicar el algoritmo SLAM.

*Usando como referencia (SLAMTEC, 2019)*

### 3.3. Sistema de Actuadores

Retomando lo descrito durante del capítulo 2 y la introducción del presente capítulo, el Sistema de actuadores aquel que, en función de los datos recopilados por el sistema de sensores y analizados por las unidades de procesamiento, ejecuta una serie de acciones de acuerdo al propósito del robot, esto se puede traducir en acciones de movimiento para cambiar su posición, acciones de manipulación de objetos, etc. En el caso específico de la plataforma robótica móvil que se describe en presente trabajo solo nos interesa acciones de movimiento con el fin de que el robot se traslada de una posición a otra y algunos señales visuales por parte del robot. Estas tareas pueden llevarse a cabo con los componentes que se muestran en la figura 3.7:

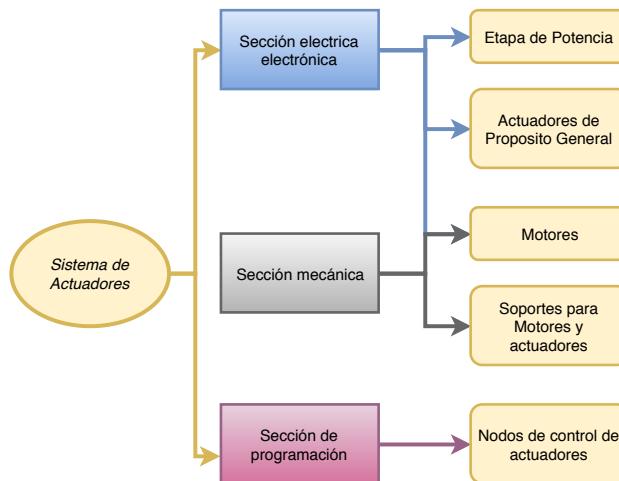


Figura 3.7: Elementos del sistema de actuadores.

Como se puede observar los elementos mas importantes son aquellos que se involucran en el movimiento de la plataforma robótica: etapa de potencia y motores, sin dejar aun lado posibles elementos que puedan ser añadidos en un futuro desarrollo del presente trabajo.

### **Motor-Gear EMG49**

El Motor-Gear con encoders modelo EMG49, es un motor de 24 V de alimentación completamente equipado con encoders y una caja de reducción de cambios a 49:1, este modelo de motor es ideal para proyectos de robótica de mediano o alta gama, razón por la cual se optó por el uso de este componente en el presente trabajo, ya que provee un buen control de manejo así como una constante realimentación para el usuario. También incluye un capacitor supresor de ruido estándar entre el bobinado de los motores.



Figura 3.8: Motor Gear EMG49.

Entre las características descritas encontramos

- La dirección del moto se encuentra integrado por un conector de 2 vías Excon tipo 3961 de 3.96mm, el cual cuenta con un cable estándar de 300 mm.
- Los encoders usan un conector de 4 vías JST, e igualmente como el anterior cuenta con un cable estándar de 300 mm.

Sus características de desempeño se muestran en el siguiente tabla:

El diseño físico del motor, así como las medidas del mismo se muestran a continuación se muestra a continuación:

Los conectores se clasifican en los colores que se muestran en la siguiente tabla:

*Usando como referencia (Electronics, 2019)*

	RATED	STALL	NO LOAD
Corriente	2100 [mA]	13 [A]	500 [mA]
Salida	34.7 [W]		
Torque	16 [kg/cm]		
Velocidad	122 [rpm]		143 [rpm]
Voltaje	24 [V]		

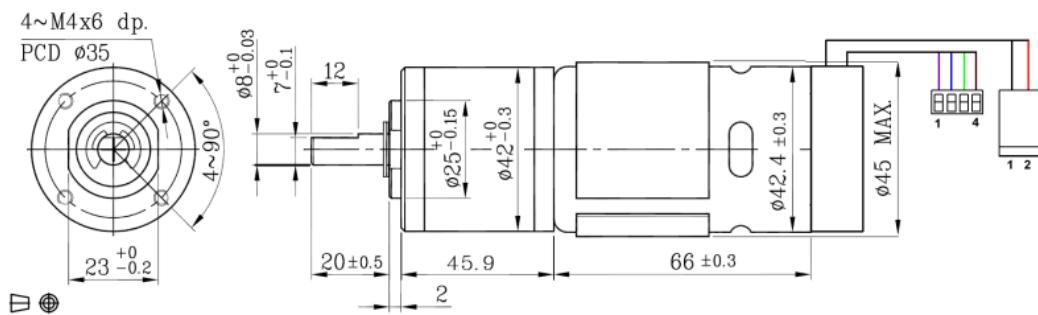


Figura 3.9: Esquemáticos del Motor-Gear EMG39.

Categoría	Color Cable	Conexion
Dirección Motor	Negro	MOTOR -
	Rojo	MOTOR +
	Morado	Sensor de efecto Hall B Vout
Encoders	Azul	Sensor de efecto Hall A Vout
	Verde	Sensor de efecto Hall GND
	Café	Sensor de efecto Hall Vcc

### 3.4. Unidades de Procesamiento

Las unidades de procesamiento son aquellos elementos que se encargan del análisis de la información suministrada por los sensores, este análisis se da en forma de bucles de realimentación generando acciones que comparan la información con patrones de referencia; además se encarga de la toma de decisiones del robot, siendo principalmente la planificación de trayectoria de la plataforma robótica móvil desde una posición inicial a una posición objetivo, sin colisiones y minimizando un determinado índice de error. Este sistema se encuentra implementado en los que se identificará como las Unidades de Procesamiento, donde gracias a la interacción de la arquitectura del sistema y los algoritmos desarrollados en conjunto permitirán un correcto manejo del control del robot en su totalidad.

En las unidades de procesamiento se contemplarían los dispositivos que implementen la arquitectura del sistema y los algoritmos para toma de decisiones, en otras palabras, visión computacional y navegación; mostrado continuación en la siguiente imagen:

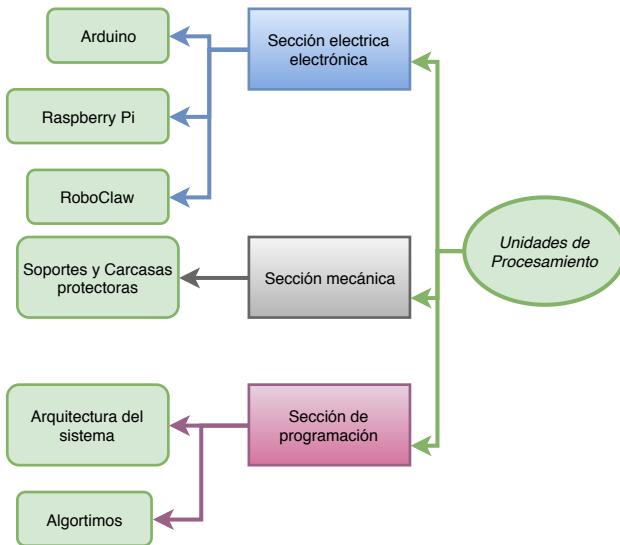


Figura 3.10: Elementos de las unidades de procesamiento.

### ARDUINO MEGA 2560

Recordando brevemente lo escrito durante el capítulo 2, marco teórico; Arduino es una plataforma electrónica que pertenece a la filosofía Open Source y que busca facilitar el uso de software y hardware a los usuarios interesados en desarrollos de proyectos y el prototipado de los mismos, ofreciendo las siguientes ventajas:

- Una plataforma confiable u de fácil uso.
- Bajo costo, de diseño Open Source Hardware.
- Multiplataforma.
- Entorno de desarrollo claro y simple, además de ser de código abierto (Open Source Software).

Debido a un análisis previo de las necesidades de la plataforma robótica en función de los sensores y la información que se recopilará y el papel de la tarjeta Arduino en el uso de algunos de los actuadores de la plataforma robótica, se concluyó que la opción más viable a utilizar sería el modelo ARDUINO MEGA 2560 el que cubriría dichas necesidades, considerando que es uno de los modelos más recomendados para proyectos en robótica.

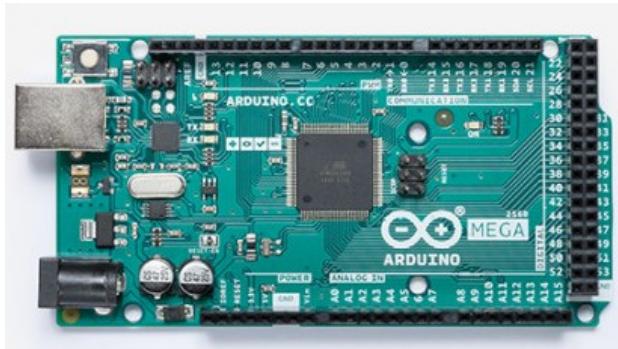


Figura 3.11: ARDUINO MEGA2560.

La plataforma de prototipado ARDUINO MEGA 2560 es fundamentalmente el modelo UNO pero cuyas características fueron mejoradas para proyectos con un grado mayor de complejidad, además de que encuentra basado en el microcontrolador ATmega2560.

#### 44 CAPÍTULO 3. DISEÑO Y CONSTRUCCIÓN DE LA PLATAFORMA

Entre las características mejoradas se encuentran:

- 54 pines de entrada y salida de tipo digital de los cuales 15 de ellos pueden ser usados como salidas tipo PWM, 16 pines de entrada tipo digital.
- Mayor espacio de memoria para el almacenamiento de los programas generadas.
- Es compatible con la mayoría de los shields desarrollados para el modelo UNO y las plataformas precedentes Duemilanove o Diecimila.

*Usando como referencia (multiple authors, 2019b)*

*Componentes adicionales requeridos:*

- Cable USB tipo A-B.
- Carcasa protectora Arduino MEGA.

### Raspberry Pi 3 Model B

Como ya se mencionó durante el capítulo 2 en el apartado de la Fundación Raspberry Pi, la tarjeta Raspberry Pi es un ordenador de placa reducida que tiene como objetivo estimular la enseñanza de la informática de manera generalizada en cualquier entorno potencial, específicamente hablando centros de enseñanza académica.

La elección del modelo 3 de la tarjeta Raspberry Pi con respecto a modelos anteriores, es debido a que entre sus características ya viene integrado con algunos dispositivos de comunicación que facilitarían su uso en la plataforma robótica móvil.



Figura 3.12: Raspberry Pi 3 Model B.

Entre las características que este modelo ofrece se tiene:

- Procesador Quad Core 1.2GHz Broadcom BCM2837 64bit, con 1 GB de RAM. Dispositivo BCM43438 para conexión LAN inalámbrica y Bluetooth Low Energy (BLE) integrado en la placa, siguiendo el protocolo 100 Base Ethernet.
- Cuenta con una serie de GPIO extendido de 40 pines. Salida de 4 polos estéreo y puerto de vídeo compuesto HDMI de tamaño completo y 4 puertos USB 2.
- Puerto de cámara CSI para conectar una cámara Raspberry Pi.
- Puerto de visualización DSI para conectar una pantalla táctil Raspberry Pi.

## 46 CAPÍTULO 3. DISEÑO Y CONSTRUCCIÓN DE LA PLATAFORMA

- Puerto micro SD para cargar su sistema operativo y almacenar datos.
- Fuente de alimentación Micro USB conmutada actualizada de hasta 2.5A.

*Usando como referencia (Foundation, 2019)*

*Componentes adicionales requeridos:*

- Cable USB - mini USB.
- Carcasa protectora Raspberry Pi.

### **RoboClaw 2x15A**

El dispositivo *RoboClaw 2x15A* es un controlador inteligente de motores, cuyo diseño tiene como objetivo el control de motores eléctricos con escobillas de corriente directa que requieran una corriente de alimentación en el rango de 15A a 30 A por canal. Entre las características que nos ofrece este modelo encontramos:

- La gestión de este dispositivo puede realizarse desde un puerto USB, puertos PWM, señal serial TTL, señales analógicas y microcontroladores. Es compatible con las tarjetas Arduino y Raspberry Pi.
- Admite niveles lógicos de 3.3V o 5V, interruptores de límite de desplazamiento, interruptores domésticos, interruptores de parada de emergencia, fuentes de alimentación, sistemas de frenos y clemas contactoras.
- Los motores de corriente continua brushed pueden ser controlados en bucle abierto o cerrado utilizando los modos de control de posición o velocidad.
- La funcionalidad de bucle cerrado ofrece un control absoluto sobre la velocidad, la velocidad y la dirección, independientemente de los cambios de carga.
- Se cuenta con varios comandos incorporados para controlar la aceleración, desaceleración, distancia, velocidad, detección de corriente, límites de voltaje, etc.

- Incorpora varias características de protección autocontroladas que incluyen: límites de temperatura, corriente, sobre voltaje y bajo voltaje, las cuales tiene como objetivo la protección ante daños en cualquier condición de operación; además de proporcionar varias configuraciones definibles por el usuario en términos de límite de corriente máxima y el mínimo de la caga de la batería para un control preciso.
- Ofrece las capacidades regenerativas, las cuales cargarán una batería de suministro durante la desaceleración o rotura.

*Usando como referencia (BasicMicro, 2019)*

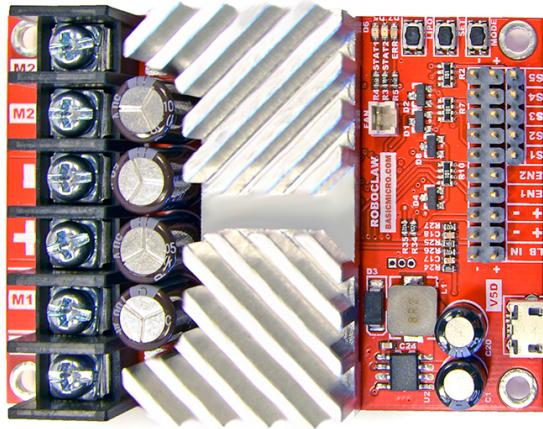


Figura 3.13: Roboclaw modelo 2x15A.

*Componentes adicionales requeridos:*

- Cable USB - mini USB.
- Cable puente(jumpers), FALTA.

### 3.5. Sistema de Suministro de Potencia

Este sistema, si bien no es mencionado explícitamente durante la definición del concepto de un Robot Móvil, representa como los sistemas anteriores, un pilar fundamental a tener en cuenta durante el diseño y construcción de cualquier robot móvil; esto es debido a que los elementos considerados en esta sección podrían influir en el rendimiento y las capacidades de hardware y software de la plataforma robótica móvil, estos elementos serían los que se muestran en la siguiente imagen:

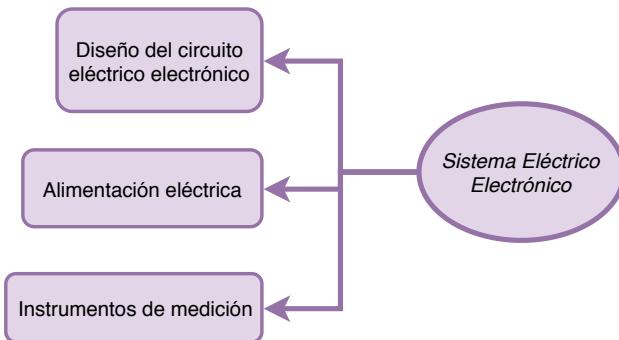


Figura 3.14: Elementos del sistema eléctrico electrónico.

La mayor parte de los elementos se involucran en el suministro de energía eléctrica a cada uno de los componentes de los sistemas anteriores, por lo cual debe existir un consenso en el cual se pueda obtener el mejor rendimiento de los componentes pasivos y activos con miras de aprovechar completamente la vida útil de los mismos.

### Esquema del circuito eléctrico del robot

A continuación se presenta un esquema que representa conceptualmente las conexiones eléctricas que existirán entre los componentes electrónicos en función de sus características.

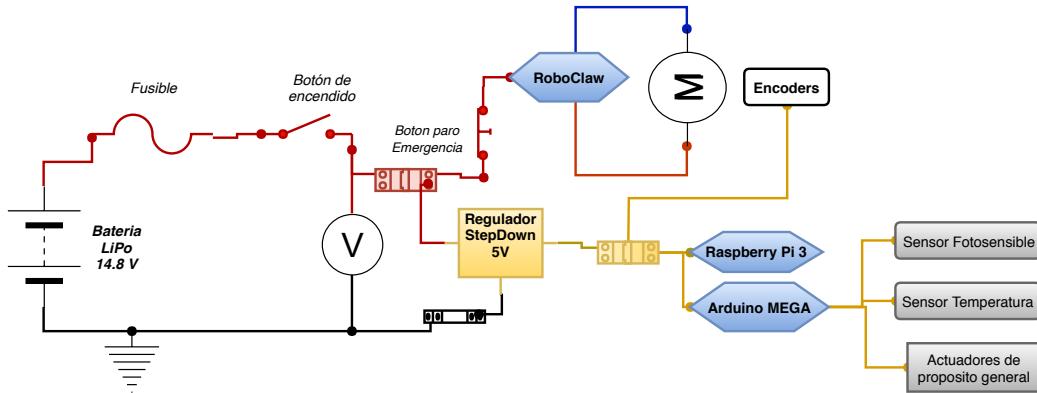


Figura 3.15: Esquema del circuito eléctrico del robot.

El esquema presentado en la figura 3.15 busca simplificar por medio de representación por bloques la forma en que se debe realizar el suministro eléctrico de cada uno de los componentes descritos durante el presente capítulo, capítulo 3. En caso de ser requerido un esquema más descriptivo, en el anexo C se encuentra la figura C.1 la cual describe a detalle las conexiones entre componentes.

## 50 CAPÍTULO 3. DISEÑO Y CONSTRUCCIÓN DE LA PLATAFORMA

### Batería LiPo PowerHD, 14.8V

Para el aspecto de la alimentación eléctrica de la plataforma robótica móvil lo primero que se debía considerar es el aspecto de la movilidad del robot por lo cual se debía eliminar las opciones que involucrasen fuentes de alimentación fija; siguiendo ese mismo hilo de pensamiento otro aspecto a destacar es la potencia requerida por los motores para un óptimo funcionamiento. Una vez planteadas dichas necesidades se propuso el uso de una batería de polímero de litio, comúnmente conocidas como baterías LiPo; específicamente hablando se propuso la *Batería LiPo 14.8V, 2200mAh 4S1P 35C del fabricante PowerHD*, la cual nos ofrece las siguientes características:

- Suministro de 14.8 V, de 4 celdas.
- 2200 mAh de descarga.
- 35C de tasa de descarga.
- Conector tipo JST-XH de carga.
- Conector tipo "T" de descarga.
- Peso de 239 gr.

*Usando como referencia (PowerHD, 2019)*



Figura 3.16: Batería LiPo 14.8 V. PowerHD

### Botón de paro de emergencias

Durante el diseño y la construcción de una plataforma robótica de cualquier índole se debe considerar elementos de seguridad que garanticen la integridad de los componentes propios del robot y la seguridad del usuario o del entorno en caso de que se presente un comportamiento inadecuado o imprevisto RoboCup@Home 2019: Rules and Regulations (draft) (2019).

Por lo cual para este apartado se propone el uso de un *botón de paro de emergencias*, el propósito de este botón será cortar el suministro eléctrico de todos los actuadores de movilidad y manipulación, sin afectar el suministro de las unidades de procesamiento y los sensores; esto con el objetivo de evitar apagar todo todo el sistema de la plataforma robótica y tener que reiniciar nuevamente todo el sistema.



Figura 3.17: Botón de paro de emergencia estándar.

### Instrumentos de medición, Voltímetro

Retomando la idea expuesta en la sección 3.2 en el apartado del sensor de batería, sobre la importancia de estar monitorizando la fuente de voltaje se optó por incluir un voltímetro digital sencillo el cual proporcionara al usuario una señal claramente visible del voltaje con el que está contando la batería LiPo en uso.



Figura 3.18: Voltímetro digital 0-100V.

### Regulador de voltaje comutado Pololú.

Otro elemento considerado en el diseño de circuito eléctrico de la plataforma robótica mostrado en la figura 3.15 esquema del circuito eléctrico, es un regulador de voltaje por el cual puedan ser alimentados principalmente las unidades de procesamiento del robot, las cuales requieren un voltaje de máximo 5 V. Considerando la amplia gamma de modelos que podemos encontrar en el mercado se consideró que un *regulador de voltaje comutado de bajada* para suplir dicha tarea; específicamente hablando se optó por el *regulador de voltaje comutado pololú modelo D15V70F5S3*.

El regulador compacto de voltaje comutado de bajada D15V70F5S3, nos ofrece la posibilidad de tener una salida de voltaje baja, seleccionable por el usuario con una eficiencia aproximada del 90 % siendo superior que la mayoría de los reguladores lineales respecto a el cambio del voltaje de entrada al de salida, además de considerarse la perdida de tensión en el proceso. Entre las características que nos ofrece este modelo se tiene:

- Voltaje de salida configurable por el usuario, 3.3V o 5V.
- Voltaje de entrada de 4.5 V a 24 V.
- Corriente continua de 7 A, en función de la disipación térmica.
- Frecuencia de cambio a 700 kHz.
- Peso aproximado de 6 g, sin pines.

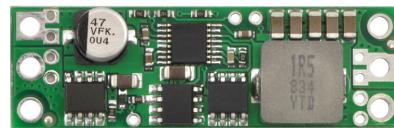


Figura 3.19: Regulador de voltaje comutado pololú D15V70F5S3.

*Usando como referencia (Pololú, 2019)*

**NOTA:** Actualmente este componente se encuentra descontinuado, por lo que los modelos sustitutos recomendados por el fabricante sería

### Cableado general

A continuación se muestra un breve listado de algunos de los componentes que deben considerarse para realizar correctamente el cableado de cada uno de los componentes, pero que no requieren una descripción extensa de sus características:

- Bornes clemas Weidmuller.
- Botón de encendido de balacín.
- Cable puente o cables dupon macho-macho.
- Cable puente o cables dupon macho-hembra.
- Cable puente o cables dupon hembra-hembra.
- Conductor de cable flexible calibre 8, colores amarillo, negro y rojo.
- Conector de batería tipo T macho.
- Conector molex hembra de 2 vías P156.
- Conector 2 mm hembra de 4 vías.
- Fusible americano 3 A.
- Header dupon hembra
- Placa fenólica perforada 4.5 cm
- Porta fusible americano de cartucho.
- Regleta de clemas conectadoras sencilla.

*Usando como referencia (Platt, 2012)*

### 3.6. Sistema Mecánico Estructural

De manera similar al apartado anterior, el sistema mecánico estructural no es mencionado explícitamente, sin embargo representa un pilar fundamental a tener en cuenta durante el diseño y construcción de cualquier robot móvil; se podría decir que en esta sección toma importancia el diseño físico-estético que se busque otorgarle a la plataforma robótica móvil basados en los elementos que conforman el robot móvil y en el propósito por el cual está siendo construido. Los elementos que se contemplan en esta sección son los que se muestran en la imagen a continuación:

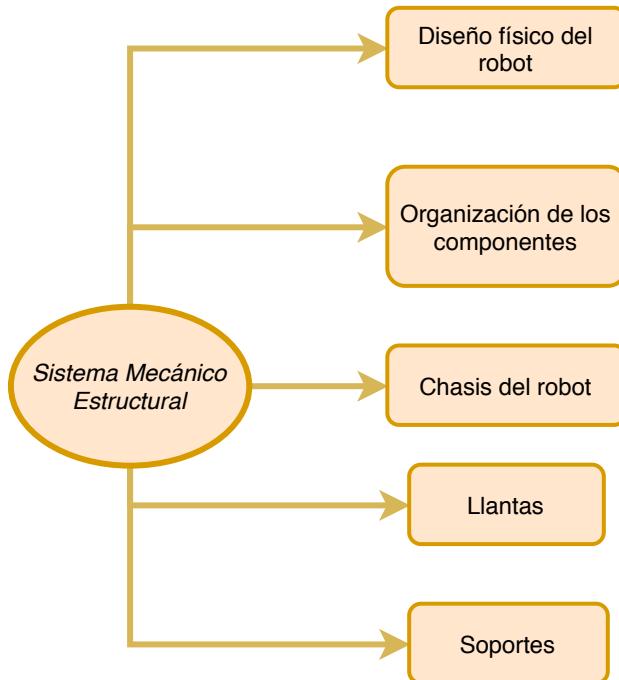


Figura 3.20: Elementos del Sistema Mecánico Estructural.

### Diseño del chasis

Uno de los aspectos de mayor importancia en el diseño de un robot es el propósito que se le va a dar, ya que en base a esto se definen sus características, componentes y el entorno en el que se encontrará trabajando. Por ello, hablando específicamente hablando de la plataforma robótica móvil que se está proponiendo en el presente trabajo se buscó cumplir las siguientes características:

- Un diseño sólido y sencillo para su montaje.
- Un diseño que ofrezca la capacidad de transporte y protección para sus componentes internos.
- Un diseño que sea fácilmente personalizable por el usuario.

Basándonos en los puntos anteriores, la propuesta del diseño para la plataforma robótica móvil se muestra en la siguiente figura 3.21:

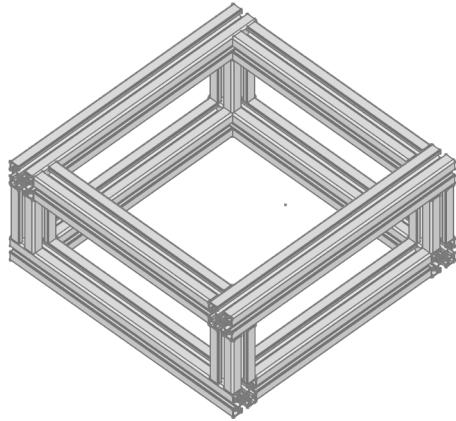


Figura 3.21: Esquema del chasis propuesto.

## 56 CAPÍTULO 3. DISEÑO Y CONSTRUCCIÓN DE LA PLATAFORMA

### *Materiales requeridos*

Respecto a los materiales de construcción se propone usar aluminio estructural o ranurado por ser un material que presenta una sustentabilidad y vida útil considerable para este tipo de proyectos, permite facilidad en modificaciones y al momento de reemplazar componentes en una estructura, además de que el peso ligero ayudará a aprovechar el máximo desempeño de los motores, finalmente se necesitan herramientas de bajo costo para el ensamblaje.

A continuación se presenta una lista de los materiales requeridos para el ensamblaje del chasis tal y como se presenta en el diseño descrito en la figura 3.21.

- Perfil de aluminio estructural o ranurado de 45x45 mm, centro de 100 mm para roscar a M12, tramo de 100 mm.
- Perfil de aluminio estructural o ranurado de 45x45 mm, centro de 100 mm para roscar a M12, tramo de 330 mm.
- Tornillo de union M12x30 mm para perfil 35.
- Tornillo cabeza de martillo M8X22 mm con tuerca collar M8.

### Organización de los componentes

En la presente imagen 3.22 se presenta una propuesta de la organización en la que se colocarán cada uno de los componentes descritos a lo largo de este capítulo, exceptuando los motores, soporte de motores, y ruedas.

El principal objetivo de esta propuesta es que los componentes tengan un orden natural en función de las conexiones tanto del cableado eléctrico como de comunicación entre algunos componentes. Además de tener definidas las posiciones en caso de que se tenga que modificar, reparar o reemplazar algún componente

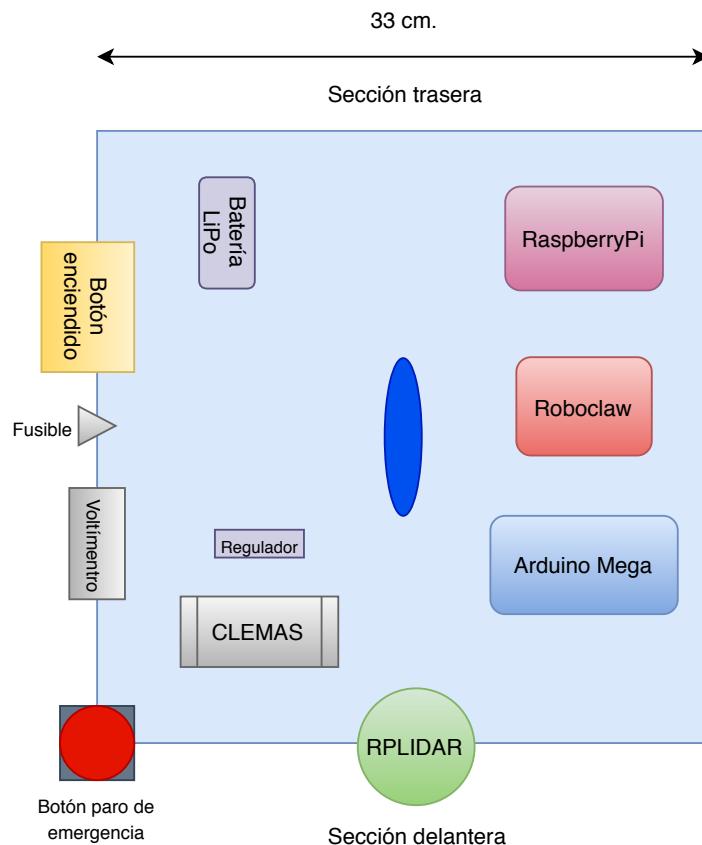


Figura 3.22: Organización de los componentes propuesta.

**Llantas, Rueda 125.**

Para este elemento se decidió utilizar las llantas recomendadas por el fabricante Robo electronics con el fin de aumentar la eficiencia y vida útil de los motores utilizados, además de no invertir en elementos extras tales como coples para otro tipo de llantas que no garanticen un correcto funcionamiento. Las llantas en cuestión están catalogadas bajo el nombre de Rueda 125", las características que describe el fabricante son las siguientes:

- Medidas de la rueda: 125 mm de diámetro, con un cubo de 8 mm de diámetro para un fácil acoplamiento al EMG49.
- La rueda cuenta con una banda de rondadura de goma de 28 mm de ancho.

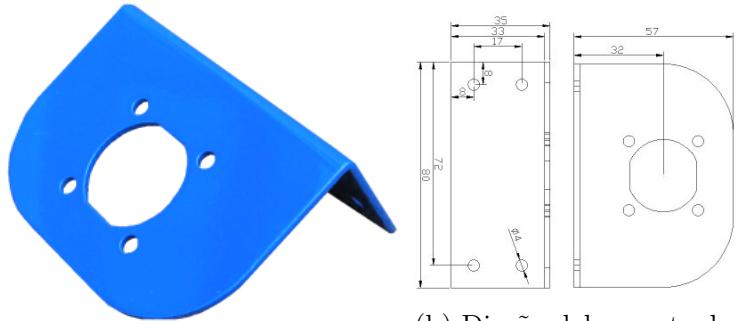


Figura 3.23: Rueda 125 de Robo electronics.

*Usando como referencia (Electronics, 2019)*

### Soporte de montaje EMG49

Un elemento necesario para el uso de los motores EMG49 descritos anteriormente en el Sistema de actuadores, son los *Soportes de montaje para los motores EMG49* o *EMG49 mounting bracket* que nos ofrece el mismo fabricante de los motores EMG49, los soportes se encuentran fabricados con aluminio fuere de 2 mm. de espesor con un acabado en esmalte azul. Como su nombre lo indica su propósito es el de ofrecer un soporte adecuado para los motores una vez estos se integren al cuerpo del robot.



(b) Diseño del soporte de monta-

(a) Soporte de montaje EMG49. je EMG49.

Usando como referencia (*Electronics, 2019*)

### Carcasas protectoras

En la sección 3.4 se consideró adecuado el proponer medidas protectoras para las unidades de procesamiento que se estaría utilizando, por lo cual se propone usar los siguientes diseños de carcasas protectoras para cada una de las siguientes tarjetas:

- *Arduino Mega 2560*



Figura 3.25: Carcasa protectora para Arduino Mega.

Diseño obtenido en (Rodriguez, 2019)

- *Raspberry Pi 3 Model B*



Figura 3.26: Carcasa protectora para Raspberry Pi 3.

Diseño obtenido en (M-P, 2019)

**Soporte general**

A continuación se realiza un breve listado de los elementos de sujeción que se utilizarán principalmente para dar un orden tanto a los componentes como al cableado con el objetivo de minimizar los riesgos de golpes físicos o desconexión de algún componente.

- Alambre recubierto o plastinudos.
- Cinchos de plástico, de diferentes medidas.
- Correas de velcro.
- Postes de plástico.
- Termoencogible o Thermofit, de diferentes medidas.



# Capítulo 4

## Arquitectura del Software para la Plataforma

En el presente capítulo se planteará el diseño de la arquitectura del sistema que se codificará para su implementación en el hardware de la plataforma robótica móvil, este sistema se diseñará usando como base las definiciones descritas en el capítulo 2 y considerando los componentes descritos durante el capítulo 3 del presente trabajo.

Se comenzará con la propuesta de la arquitectura general del sistema, identificando cada uno de los elementos ya existentes en ROS y los que se tendrán que desarrollar en función de los componentes de hardware disponibles, se dará una breve descripción de las paqueteterías predeterminadas de ROS que se utilizarán en el sistema y sus características principales. Se realizará un breve listado de los elementos y sus interacciones en el sistema, con el fin de presentar el funcionamiento a nivel de software de la plataforma robótica y dar paso a las posibles aplicaciones de las capacidades de la plataforma robótica móvil comenzando con comportamientos simples y dar paso al siguiente capítulo del presente trabajo.

Finalmente se dispondrá de un breve manual de instalación del software necesario para utilizar el sistema descrito en este capítulo, el cual se podrá consultar en el anexo F, así como el esquema completo de la arquitectura del sistema en el anexo E del presente trabajo.

## 4.1. Sistema operativo

En el capítulo 3 se mencionó que uno de los componentes principales en el rubro de las unidades de procesamiento sería la tarjeta Raspberry Pi 3 sobre la cual se correría la arquitectura de la plataforma robótica, y por ende el uso de un sistema operativo compatible con la tarjeta. Entre las opciones que nos ofrece la plataforma Raspberry que sean compatibles con ROS se escogió el sistema operativo *\_Ubuntu 16-04 LXDE* el cual es una ligera modificación al sistema operativo *Ubuntu 16.04 Xenial Xerus* donde se reemplaza el entorno de escritorio por defecto GNOME por el Lightweight X11, un entorno de escritorio ligero, de alto rendimiento y eficiencia a la hora de consumir recursos del hardware (*UBUNTU, 2019*). Además de contar con la versión de *ROS Kinetic* preinstalada, lo cual facilitaría el trabajo del usuario a la hora de instalar el sistema en su totalidad para el uso de la plataforma robótica móvil.

En caso de que el usuario no disponga con una tarjeta Raspberry o bien quiera trabajar con un equipo más familiar, esto es, una computadora laptop basta con instalar el sistema de la plataforma robótica en un equipo el cual cuente con un sistema operativo basado en Linux y que cuente con soporte de ROS Kinetic; en este caso se recomendaría el uso de Ubuntu 16.04 LST.

## 4.2. ARQUITECTURA DE LA PLATAFORMA ROBÓTICA MÓVIL BASADA EN ROS65

### 4.2. Arquitectura de la Plataforma Robótica Móvil basada en ROS

En el capítulo 3 se mencionó brevemente en los esquemas que describían cada uno de los sistemas en los que se dividió la plataforma robótica la programación que se consideraba pertinente realizar para tener un control adecuado de cada uno de los componentes instalados en la plataforma robótica.

En en esquema 4.1 se describe brevemente

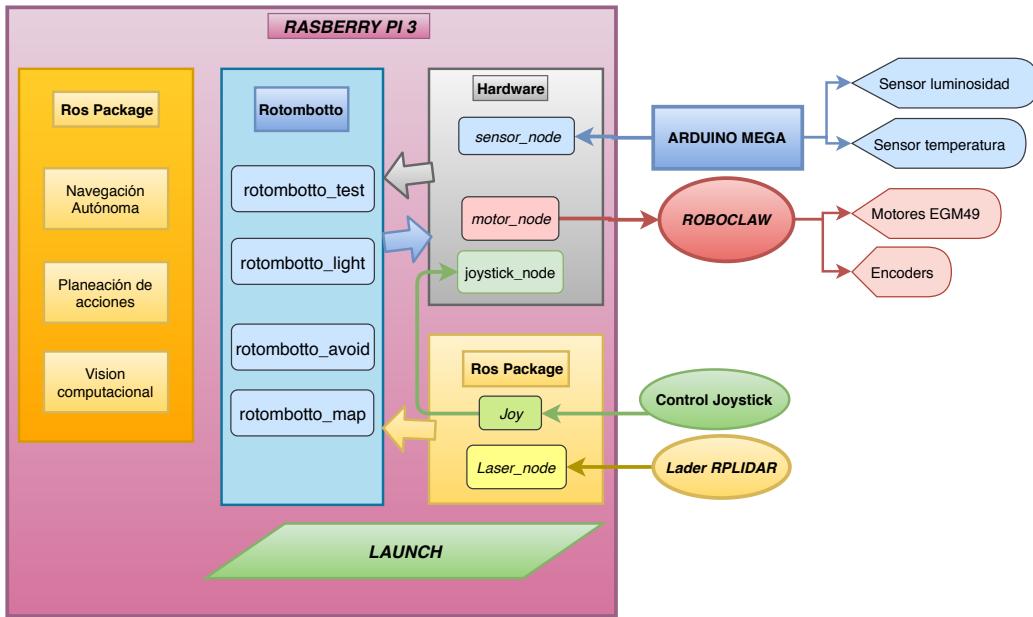


Figura 4.1: Arquitectura del sistema implementado en la plataforma robótica móvil.

### 4.3. Paquetes predeterminados de ROS, *ROS packages*

En esta sección se hablará de los paquetes usados en el sistema de la plataforma robótica móvil que ya se encontraban integrados en ROS por defecto. Los cuales en su mayoría se encuentran enfocadas en atender ciertos componentes de uso regular en el campo de la robótica, tales como telémetros láser, controles joystick, etc.

#### **Paquete Joy, *joy package***

El paquete joy es básicamente un driver de ROS realizado para el uso de un control joystick genérico en Linux. Este paquete cuenta con las siguientes características:

- El paquete contiene el nodo de nombre *joy\_node* el cual realiza la interconexión a nivel de hardware entre el control joystick genérico y ROS.
- El nodo principal publica el tópico *sensor\_msgs/Joy* el cual transmite la información del estado actual de los botones y las palancas del control joystick.

*Usando como referencia (ROS.org, 2019e)*

#### **Paquete rplidar, *rplidar package***

El paquete rplidar de ROS provee soporte básico para el uso del telémetro laser RPLIDAR A1, el cual consiste en un nodo que funge como driver resultado de usar el kit de desarrollo (SDK) del RPLIDAR en conjunto con ROS; entre las características:

- El nodo principal de nombre *rplidarNode*.
- El tópico que podemos usar al ahora de utilizar este paquete es el */sensor\_msgs/LaserScan* o bien ejecutado por medio del comando *scan (sensor\_msgs/LaserScan)*
- Este paquete ofrece dos servicios para el RPLIDAR: *stop\_motor (std\_srvs/Empty)* el cual sirve para detener el motor del RPLIDAR y el *start\_motor (std\_srvs/Empty)* el cual sirve para encender el motor del RPLIDAR.

*Usando como referencia (ROS.org, 2019i)*

### **Paquete rosserial, *rosserial package***

El paquete *rosserial* es un protocolo para encapsular mensajes de ROS estándar serializados y multiplexando varios tópicos y servicios para el uso sobre un dispositivo en específico que haga uso de un puerto serial o una red de socket. Las bibliotecas cliente permiten a los usuarios el uso sencillo de nodo de ROS y su ejecución en diferentes sistemas, estos *clientes* son puertos incluidos en la biblioteca *rosserial\_client* sin embargo la que nos importa para el desarrollo del sistema del presente trabajo es la biblioteca *rosserial\_arduino* (*ROS.org, 2019g*).

Al utilizar el paquete *rosserial\_arduino* el usuario puede usar ROS directamente con el IDE de Arduino, *rosserial* provee a ROS comunicación por protocolo apoyándose en el UART de Arduino. Esta paquetería permite a Arduino ser tomado como un nodo propio de ROS el cual puede publicar y suscribirse directamente a los mensajes de ROS, publicar trasformadas tf, y obtener el tiempo de sistema de ROS. La estructura de ROS se implementa por medio de una biblioteca de Arduino, como toda librería de Arduino *ros\_lib* funciona colocando la implementación de las librerías dentro del fichero el archivo que se está creando en el entorno de trabajo.

Para poder usar la biblioteca *rosserial* en el código de Arduino apropiadamente deben colocarse las siguientes cabeceras:

- `#include <ros.h>`
- `include <std_msgs/String.h>`

*Usando como referencia (ROS.org, 2019h)*

## 4.4. Descripción de los paquetes de hardware

En este apartado se describirá a detalle las paqueterías mostradas en la figura 4.1 de la arquitectura del sistema de la plataforma robótica móvil, así como sus características a considerar a la hora de implementarse.

### **Paquete Arduino, *arduino package***

Este paquete principalmente refiere al programa desarrollado para su uso en la tarjeta Arduino Mega, y el cual se encarga de obtener la información de los sensores disponibles en la plataforma robótica móvil, los cuales serían los sensores de luz y el sensor de temperatura. Este nodo se comunica por medio del puerto serial asignado al conectarse mediante el puerto USB.

- *Tópico que publica:* /hardware/arduino/data.

Se debe recalcar que este paquete trabaja a la par con el paquete integrado a ROS *rosserial* y el cual debe ser instalado previamente al uso del sistema aquí descrito. El proceso de instalación se describe en el anexo F del presente trabajo.

### **Paquete Motores, *motor package***

Este paquete se encarga del control de dirección y velocidad de los motores, por medio de los nodos que reciben la información del sistema y la transmiten a la tarjeta *roboclaw* usando como complemento un driver para el uso de esta última.

En este paquete se encuentran los siguientes nodos:

- *motor\_node*, es el nodo principal para el uso y control de la tarjeta *roboclaw*, el cual recibe las dirección y velocidad en las que se desea dirigir al robot.
  - *Tópicos a los que se subscribe:* /hardware/motors/speeds.
  - *Tópico que publica:* /hardware/motors/odometry.

- *motor\_test*, así como el anterior este nodo se encarga del uso y control de la tarjeta roboclaw, el cual recibe las dirección y velocidad en las que se desea dirigir al robot. Se diferencia principalmente por las constantes salidas a terminal de los datos que está recibiendo con el fin de revisar si existe alguna anomalía en el hardware de la plataforma robótica móvil.

- *Tópicos a los que se subscribe:* /hardware/motors/speeds.
- *Tópico que publica:* /hardware/motors/odometry.

Adicionalmente dentro de esta paquetería se encuentra el driver que se utilizará para la comunicación con la tarjeta *roboclaw*, donde el archivo se encuentra alojado en una subdirectorio de nombre */Hardware/motors/scripts/-driver\_roboclaw*.

#### **Paquete Sensores, *sensor package***

En paquete paquete se encuentran los nodos que se encargarán de la recolección de información trasnmitida por el arduino por medio del puerto serial. Los nodos que podemos encontrar en este paquete serían:

- *sensor\_node*, es el nodo principal que recolecta la información trasnmitida por el arduino por medio del puerto serial, y posteriormente segmenta dicha información en diferentes tópicos.
  - *Tópicos a los que se subscribe:* /hardware/arduino/data.
  - *Tópico que publica:* /hardware/sensors/luz, /hardware/sensors/-temp.
- *sensor\_test*, así como el nodo anterior recolecta y segmenta la información transmitida por el arduino, salvo que este muestra en pantalla constantemente los datos obtenidos con el fin de revisar si existe alguna anomalía en el hardware de la plataforma robótica móvil.
  - *Tópicos a los que se subscribe:* /hardware/arduino/data.
  - *Tópico que publica:* /hardware/sensors/luz, /hardware/sensors/-temp.

### Paquete Joystick, *joystick package*

En este paquete se encuentran los nodo que se encargarán de obtener los datos trasnmitidos por un control joystick alámbrico conectado mediante un puerto USB.

■ *joystick\_node*, es el nodo principal que se encarga de obtener los datos publicados del nodo predefinido por ROS para cualquier control tipo joystick que soporte la versión. Además de segmentar la información en 2 tópicos, el primero el cual envía la información de los botones y sticks, y el segundo que transforma la información de las palancas en información para el movimiento de la plataforma robótica.

- *Tópicos a los que se subscribe:* /hardware/joy.
- *Tópico que publica:* /hardware/joystick/data, /hardware/motors/speeds.

■ *joystick\_test*, se encarga de obtener los datos publicados del nodo predefinido por ROS para cualquier control tipo joystick que soporte la versión. Sin embargo solo se encarga de transformar la información de las palancas en información para el movimiento de la plataforma robótica.

- *Tópicos a los que se subscribe:* /hardware/joy.
- *Tópico que publica:* /hardware/joystick/data, /hardware/motors/speeds.

## 4.5. Descripción de Tópicos y Servicios

En el presente apartado se realizará un listado de los tópicos mencionados durante la sección 4.4 del presente capítulo, describiendo su propósito y el tipo de dato usado en el tópico.

### Tópicos predeterminados de ROS, *ROS topics*

Los tópicos descritos a continuación se encuentran como parte de los paquetes predefinidos de ROS, principalmente aquellos nodos que se usarán para la implementación del control joystick y el telémetro RPLIDAR.

- /hardware/joy o sensor\_msgs/Joy, este tópico se encarga de transmitir la información enviada por el control joystick dividido en dos secciones: la información de la posición de los ejes de las palancas y la información de cada uno de los botones, siendo 0 el estado en que no se encuentran presionados y 1 el estados en que se encuentran presionados.
- /std\_msgs/Laser\_scan, este tópico se encarga de publicar la información recopilada por el telémetro laser RPLIDAR. **Falta información**

### Tópicos de hardware, */hardware/... topics*

Los tópicos descritos a continuación se encuentran en su mayoría en la sección de hardware del sistema, y se encargan principalmente de transmitir la información que se obtiene directamente de los componentes o bien de transmitir la información para dar instrucciones a los elementos móviles y actuadores de la plataforma robótica móvil

- /hardware/arduino/data, este tópico se encarga de transmitir la información obtenida por el Arduino Mega, el cual contiene los datos de los sensores fotosensibles y el sensor de temperatura; el tipo de dato que se usa en este tópico es un Float32Multiarray de tamaño de 5.
- /hardware/joy, tópico predeterminado de ROS el cual obtiene la información de los botones y sticks analógicos del control compatible con la paquetería, dividiendo los datos en dos categorías: *buttons* y *axes* cuya longitud varía dependiendo de la característica del control utilizado; el tipo de dato que se usa en este tópico es un *msg* de tamaño variable en función del control.

- [/hardware/joystick/data](#), Este tópico rescata algunos valores específicos de ciertos botones para su uso a necesidad del usuario ; el tipo de dato que se usa en este tópico es un Float32Multiarray de tamaño de 3.
- [/hardware/motors/odometry](#), este tópico se encarga de la odometria ; el tipo de dato que se usa en este tópico es un Odometry() **Revisar este tipo de dato**.
- [/hardware/motor/speeds](#), este tópico se encarga de transmitir la información de dirección y velocidad asignada para el movimiento de la plataforma robótica móvil, la cual consta de dos velocidades en el rango de [-1,1] siendo la magnitud 1 la velocidad máxima alcanzada por el motor, y el signo la dirección del motor; el tipo de dato que se usa en este tópico es un Float32Multiarray de tamaño de 2.
- [/hardware/sensors/luz](#), este tópico se encarga de transmitir la información obtenida de los sensores de luz únicamente, ya habiendo sido segmentada del tópico de arduino ; el tipo de dato que se usa en este tópico es un Float32Multiarray de tamaño de 4.
- [/hardware/sensors/tempt](#), este tópico se encarga de transmitir la información obtenida del sensor de temperatura únicamente, ya habiendo sido segmentada del tópico de arduino ; el tipo de dato que se usa en este tópico es un Float32Multiarray de tamaño de 1.

### Servicios predeterminados de ROS, *ROS topics*

Los servicios descritos a continuación se encuentran como parte de los paquetes predefinidos de ROS, principalmente para el uso del telémetro RPLIDAR, y transmisión de algunos datos.

- [start\\_motor \(std\\_srvs/Empty\)](#) , este servicio tiene como función el iniciar el movimiento del motor del telémetro láser RPLIDAR.
- [stop\\_motor \(std\\_srvs/Empty\)](#) , este servicio tiene como función el detener el movimiento del motor del telémetro láser RPLIDAR.

## 4.6. Descripción de los paquetes *launch*

Uno de los elementos importantes para el uso de ROS y un manejo más dinámico del sistema desarrollado es el uso de los archivos *launch* los cuales son un conjunto de instrucciones en los cuales se describen que paquetes, nodos, tópicos y servicios serán usados a la hora de ejecutar dicho archivo; el uso de este tipo de archivos facilita el levantar cada uno de los elementos necesarios a la hoja de ejecutar el sistema y elimina la necesidad de ejecutar separadamente el comando *roscore* para inicializar ROS; *roslaunch* ofrece las ventajas de poder ser utilizado localmente o remotamente vía SSH, además de configurar parámetros para el *Parameter Server* de ROS, entre las opciones adicionales podemos reaparecer procesos que han sido terminados (*ROS.org, 2019f*).

Hablando específicamente del sistema de la plataforma robótica móvil el uso de los archivos *launch* nos servirá para ejecutar diferentes modalidades en las que se busca implementar comportamiento específico o bien una serie de acciones de configuración para el uso de la plataforma robótica móvil. Estos archivos *launch* se encuentran en el paquete de ROS de nombre ***rotombotto\_begin*** cuyo único propósito es el de implementar este tipo de archivos para el sistema de la plataforma robótica.

A continuación se listarán las modalidades que se encuentran disponibles en el sistema de la plataforma robótica móvil con las que el usuario puede comenzar a familiarizarse con la plataforma en funcionamiento:

- *Modo prueba*

Esta modalidad está planeada para ser utilizada una vez que el usuario termine de instalar el software necesario para usar la plataforma robótica móvil a su disposición y con la cual podrá revisar el correcto funcionamiento a nivel de hardware y software de todo los componentes del mismo. Adicionalmente en el anexo F se encuentra un breve tutorial de como utilizar esta modalidad. Se ejecuta por medio del siguiente comando:

```
>$ rosrun rotombotto_begin rotombotto_test.launch
```

- *Seguidor de luz*

Esta modalidad esta propuesta como uno de los comportamientos reactivos en robótica más sencillos de implementar y con la cual el usuario puede familiarizarse con el uso de los sensores fotosensibles junto con los motores con los que cuente la plataforma robótica móvil a su disposición. Se ejecuta por medio del siguiente comando:

```
>$ rosrun roomba begin roomba_light.launch
```

- *Modo evasión de obstáculos*

Esta modalidad, al igual que la anterior, esta propuesta como uno de los comportamientos reactivos en robótica más sencillos de implementar y con la cual el usuario puede familiarizarse con el uso del láser RPLIDAR junto con los motores para la navegación de la plataforma robótica móvil a su disposición. Se ejecuta por medio del siguiente comando:

```
>$ rosrun roomba begin roomba_avoid.launch
```

- *Modo seguidor y evasión de obstáculos*

Esta modalidad la cual combina los comportamientos anteriores tiene como objetivo que el usuario comience a analizar las interacciones entre los diferentes componentes disponibles en la plataforma robótica móvil, y las posibilidades con las que el usuario puede generar sus propios comportamientos en función de sus necesidades. Se ejecuta por medio del siguiente comando:

```
>$ rosrun roomba begin roomba_avoidLight.launch
```

- *Modo creación de mapa*

Finalmente esta modalidad tiene como objetivo la creación del un mapa del entorno en el que se encuentra la plataforma robótica móvil, por medio del láser RPLIDAR, con el cual el usuario podrá comenzar a desarrollar algoritmos de navegación autónoma, los cuales se describirán a detalle en el capítulo 5 del presente trabajo. Se ejecuta por medio del siguiente comando:

```
>$ rosrun roomba begin roomba_map.launch
```

# Capítulo 5

## Algoritmos para pruebas de desempeño

Descripción del Capítulo 5

## 5.1. Navegación Autónoma

Uno de los principales propósitos de los robots móviles autónomos es la resolución de tareas de navegación en ambientes desconocidos sin requerir la supervisión de un operador para el desempeño de sus funciones.; actualmente se emplean diversos campos con el propósito de obtener un comportamiento que exprese cierto grado de inteligencia por medio de sistemas de inspección y monitoreo; estos sistemas se programan por medio de la incorporación de algoritmos formales o heurísticos para lograr que el robot móvil autónomo pueda navegar en un ambiente desconocido. Buscando identificar los problemas a los que una entidad se enfrenta cuando navega en un entorno real desconocido se propusieron las siguiente tareas a llevar a cabo:

- Clasificación de patrones.
- Navegación y Planificación de la trayectoria.
- Sistema de evasión de obstáculos.
- Administración de la energía.
- Agentes asociados a la misión en particular.

Dicho esto, para cumplir el objetivo de que una entidad navegue en un ambiente desconocido ésta debe ejecutar las tareas más intuitivas que cualquier entidad que comparte el entorno con un humano; en caso de los humanos los sistemas de navegación no poseen un mapa o sistema detallados, en términos de un sistema formal de coordenadas en el que navegan, simplemente bajo la observación del entorno se va seleccionando en línea la decisión que debe tomarse para activar el mecanismo biomecánico con el fin de aportar la maniobra que le permita continuar el desplazamiento normal al objetivo propuesto. Por lo cual se propone que durante la fase de aprendizaje y navegación se incorporen dos agentes: un módulo de precisión y un módulo lógico, cuyas funciones son reducir el números de colisiones o errores para así acelerar el aprendizaje y dotar al sistema de información previa (*Ramírez, 2003*).

En base a lo definido e identificando las necesidades para el desarrollo de la plataforma robótica móvil con respecto al tema de navegación autónoma podemos encontrar las herramientas adecuadas que nos provee ROS como parte de su repertorio.

### Navigation Stack ROS

La pila de navegación o *Navigation Stack* de ROS, obtiene la información de odometría, transmisiones de sensores y una posición objetivo para obtener una salida de comandos de velocidad adecuados que serán enviados a la base móvil del robot, se debe tomar en cuenta que utilizar este conjunto de herramientas en un robot arbitrario puede ser complicado debido a los requerimientos del mismo *Navigation Stack*, entre los requerimientos encontramos:

- El robot en el cual se desea utilizar debe contar con ROS ejecutándose en el sistema del mismo.
- El sistema debe contar con un árbol de transformadas **tf**.
- Los datos publicados de los sensores deben contar con una correcta declaración de los tipos de mensajes de ROS (*ROS message*).
- El *Navigation Stack* requiere ser configurado con las medidas y características del robot con el propósito de obtener el mejor rendimiento posible.

Mientras el *Navigation Stack* se encuentra diseñado para ser de propósito general tanto como es posible, aún se requieren ciertas especificaciones de hardware que restringen su uso:

- Está diseñado para robots con accionamiento diferencial o que cuenten con ruedas holónómicas. Además de asumir que la base móvil es controlada por medio de comandos de velocidad con un formato de tipo: *velocidad en eje x*, *velocidad en eje y*, *velocidad theta*.
- Requiere un láser planar montado en la base móvil del robot, este láser tiene el propósito de realizar la construcción del mapa del entorno y localización del robot en dicho ambiente.
- El *Navigation Stack* fue desarrollado en base a un robot cuadrado, por lo que su desempeño tendrá mejores resultados en robots cuyos diseños sean similarmente cuadrados o circulares. Si bien trabaja indistintamente en robots con diferentes tamaños y formas, puede resultar un difícil su implementación en robots cuya forma sea un rectángulo alargado en un espacio reducido.

## 5.2. Planeación de Acciones

gmapping ROS

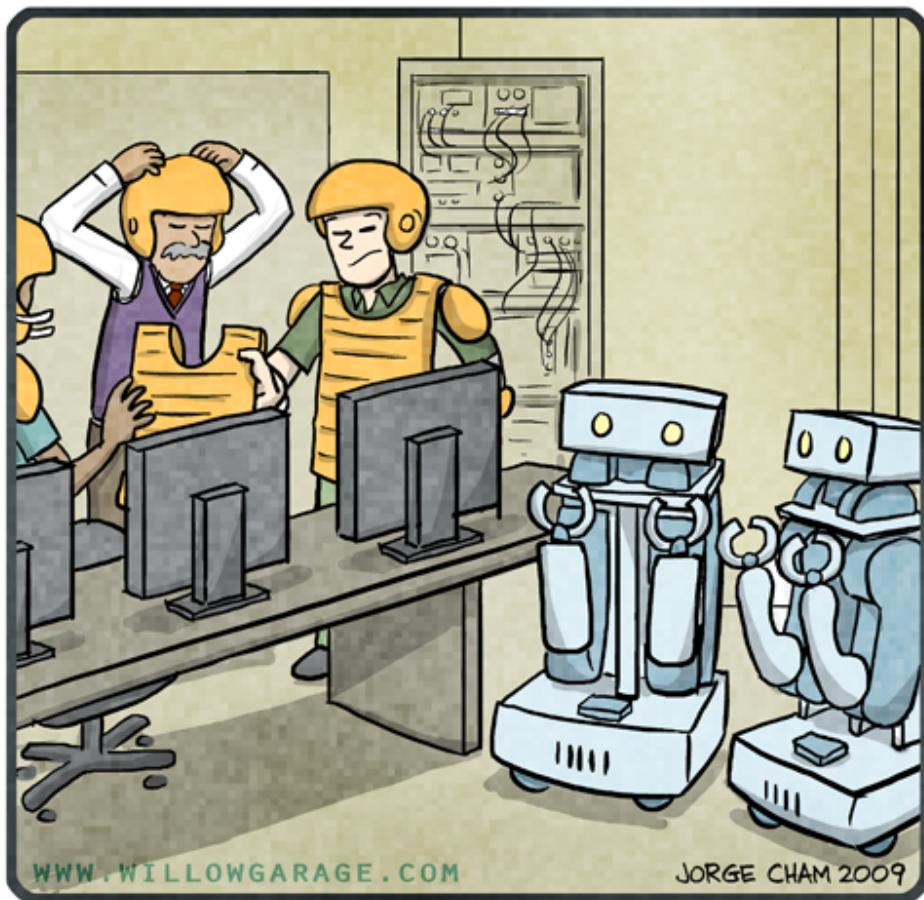
## 5.3. Visión Computacional

# Capítulo 6

## Resultados

Descripción del Capítulo 6

## R.O.B.O.T. Comics



"I HAVE A BAD FEELING  
ABOUT THIS DEMO."

# Capítulo 7

## Conclusiones y Trabajo a futuro

Concluyo que la realización de esta tesis resultó bastante interesante y de gran valor educativo para mi formación profesional



# Apéndice A

## Esquema General de un robot

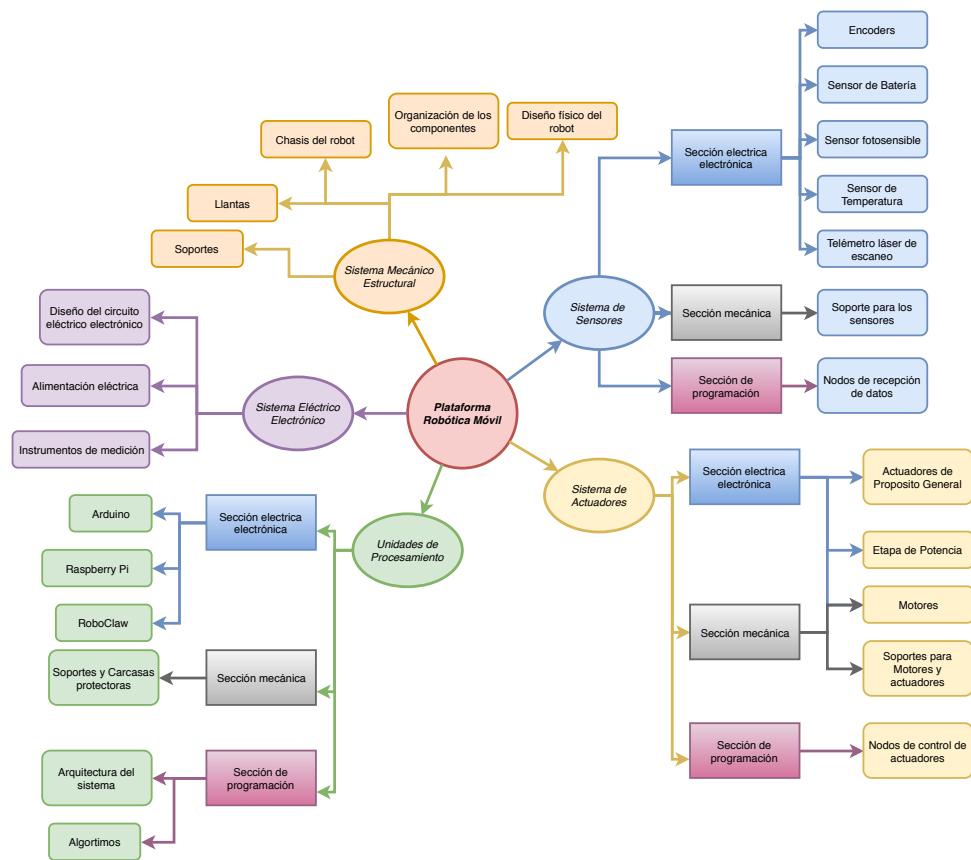


Figura A.1: Esquema general de un robot completo.



# Apéndice B

## Listado completo componentes de la plataforma robótica

Listado completo componentes de la plataforma robótica móvil

■ *Sistema de Sensores*

- Alarma de bajo voltaje para pilas LiPo de 2S a 6S.
- Sensor de luz ambiental TEMT6000, Vishay Semiconductors.
- Sensor temperatura LM45, Texas Instrument.
- Telémetro láser de escaneo RPLIDAR A1.
- **Carcasa protectora laser**
- Soporte sensores luz

■ *Sistema de Actuadores*

- Motor-Gear EMG49, Robo ELectronics.
- Soporte Motores EMG49.
- LED's

■ *Unidades de control*

- Arduino MEGA 2560.
- Cable USB tipo A-B.
- Cable USB - mini USB.

## 86 APÉNDICE B. LISTA DE COMPONENTES DE LA PLATAFORMA ROBÓTICA

- Carcasa protectora Arduino MEGA.
- Carcasa protectora Raspberry Pi.
- Memoria MicroSDHC ADATA, clase 10, 16 GB.
- Raspberry Pi 3 Model B.
- Roboclaw 2x15A.

### ■ *Sistema Eléctrico Electrónico*

- Batería LiPo 14.8V, 2200 mAh.
- Bornes clemas Weidmuller.
- Botón de encendido de balancín.
- Botón de paro de emergencia.
- Cable puente o cables dupon macho-macho.
- Cable puente o cables dupon macho-hembra.
- Cable puente o cables dupon hembra-hembra.
- [Carcasa protectora batería LiPO](#).
- Conductor de cable flexible calibre 8, color amarillo/blanco.
- Conductor de cable flexible calibre 8, color negro.
- Conductor de cable flexible calibre 8, color rojo.
- Conductor de cable flexible calibre 8, color verde.
- Conector de batería tipo T macho.
- Conector molex hembra de 2 vías P156.
- Conector 2 mm hembra de 4 vías.
- Fusible americano 3 A.
- Header dupon hembra
- Placa fenólica perforada 4.5 cm
- Porta fusible americano de cartucho.
- Regleta de clemas sencilla.
- Regulador compacto de voltaje conmutado de bajada D15V70F5S3.
- Voltímetro. [Características](#)

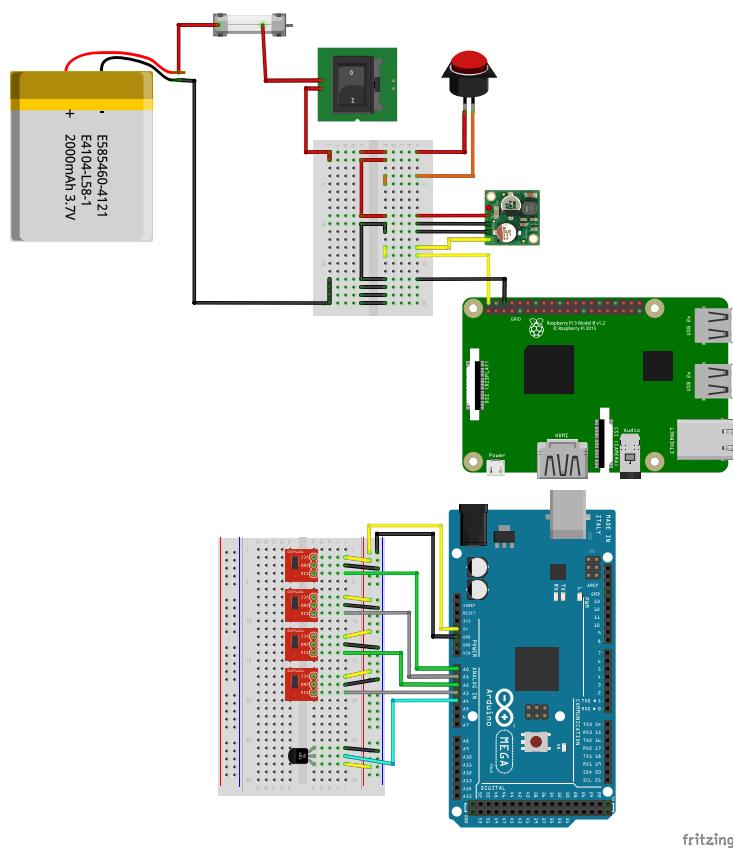
- *Sistema Mecánico*

- Bandas de velcro.
- Cinchos.
- Lámina PVC espumado.
- Perfil de aluminio estructural o ranurado de 45x45 mm, centro de 100 mm para roscar a M12, tramo de 100 mm.
- Perfil de aluminio estructural o ranurado de 45x45 mm, centro de 100 mm para roscar a M12, tramo de 330 mm.
- Rueda 125, Robo Electronics.
- Rueda giratoria de caucho.
- Tornillo de union M12x30 mm para perfil 35.
- Tornillo cabeza de martillo M8X22 mm con tuerca collar M8.

88APÉNDICE B. LISTA DE COMPONENTES DE LA PLATAFORMA ROBÓTICA

# Apéndice C

## Esquema general de conexiones.





## Apéndice D

# Proceso de ensamblaje plataforma robótica móvil

En el presente apéndice se describirá a detalle el proceso de construcción y ensamblaje de la plataforma robótica móvil con cada material y herramientas utilizados en el proceso, así como algunas consideraciones que se deberán tomar durante la construcción o en caso de que se quiera modificar el diseño propuesto en el presente trabajo.

### D.1. Armado del chasis de aluminio estructural

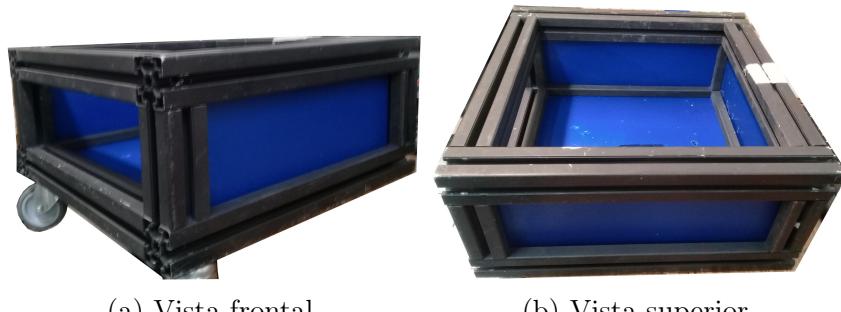
#### *Materiales requeridos*

- Lámina de PVC espumado
- Piezas de aluminio estructural, Tornillos de unión M12x30.

Para comenzar el proceso de ensamblaje del chasis debemos realizar unos cortes por medio del taladro y la broca para metal en los extremos de las piezas de aluminio a unos 5 cm. aproximadamente del borde de la pieza, y con un grosor suficiente para que pueda pasar la cabeza de tornillo de unión. Posteriormente se recomienda comenzar con la base cuadrada inferior del chasis e ir montando las piezas laterales, esto con el fin de ir obteniendo las medidas aproximadas para los cortes del PVC espumado de los paneles laterales, así como un aproximado de la localización de los componentes y demás elementos de sujeción.

Para los cortes de los paneles laterales se recomienda dejar un espacio de 2 cm de largo extra por cada lado y realizar un corte en escuadra en cada esquina, esto con el fin de que el pvc espumado tenga una figura similar a una cruz ancha y pueda insertarse en las ranuras de la estructura de aluminio para dar un mejor soporte.

Obteniendo un resultado similar al que se muestra las siguientes fotografías:



(a) Vista frontal. (b) Vista superior.

Figura D.1: Chasis armado.

Se recomienda que una vez instalados los paneles lateral izquierdo y el inferior se revise la localización de los componentes con el fin de hacer algunos cambios menores que favorezcan al usuario, además de comenzar a realizar las marcas para perforar elementos tales como los tornillos que se usarán para la sujeción de los demás componentes y las medidas de corte para los demás elementos como el botón de encendido, fusible, y el voltímetro.

## D.2. Montaje y cableado de los elementos laterales

### *Materiales requeridos*

- Botón de encendido de balancín.
- Conductor de cable flexible calibre 8, color rojo y color negro.
- Conector de batería tipo T macho.
- Porta fusible americano de cartucho, Fusible americano 3 A.
- Voltímetro.

La principal razón por la cual es recomendable comenzar con este apartado es debido a que una vez montados los motores supondrá complicado para el usuario manipular el panel lateral para la colocación y cableado de cada uno de los componentes, así como el tener que desmontarlos en caso de que exista algún error en el proceso de montaje. Además que dará una referencia para la localización de los elementos interiores que se montaran en el panel inferior.

Para este proceso lo principal fue supervisar el largo de los cables para evitar que fueran muy cortos para una conexión adecuada bien muy largo que representaran un problema de organización. Se recomienda que para los elementos como el porta fusibles los cuales tiene terminales frágiles se selle con silicon una vez que se termino y reviso la conexión.



Figura D.2: Chasis armado vista frontal.

### D.3. Montaje de los motores

#### *Materiales requeridos*

- Motor-Gear EMG49.
- Soporte Motores EMG49.
- Rueda giratoria de caucho.
- Tornillos cabeza de martillo MSX22.

En caso del montaje de los soportes que proporcionaba el fabricante se tuvo que realizar modificaciones con las cuales se pudieran montar en el aluminio estructural por medio de los tornillos de cabeza de martillo, tal y como se muestra en la imagen D.3, se tiene que tener especial cuidado con que la localización de los tornillos no afecte con el montaje posterior de los motores. En este caso en particular se utilizó las medidas 1.5 cm a lo largo y 1.7 a lo ancho para la localización del centro de la perforación a ambos lados del soporte. Una vez realizada dicha modificación se procedió a montar los soportes en la estructura de aluminio en la parte frontal de la plataforma robótica móvil.



(a) Vista superior.



(b) Vista frontal.

Figura D.3: Montaje de los motores y ruedas.

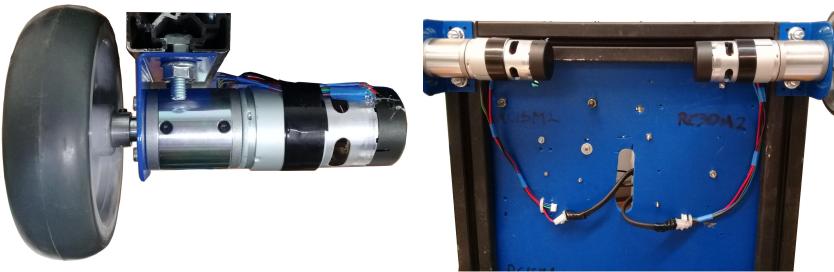
Posteriormente se procedió a montar la rueda giratoria del otro lado del chasis; por lo que se destaca que una vez instalados estos dos elementos será más difícil para el usuario manipular los paneles laterales, tal y como se muestra en la siguientes figuras:



(a) Vista frontal. (b) Vista superior.

Figura D.4: Soporte de motores y ruedas giratorias montadas.

Finalmente se montan los motores en los soportes, primeramente se debe revisar que los cables no creen conflictos en los cuales pongan en peligro su integridad, por lo que se recomienda fijar los cables a los motores tal y como se muestra en la figura D.5 A, y sujetar los cables de tal manera en que los conectores queden cerca del orificio realizado en el panel inferior para su conexión con la Roboclaw como se muestra en la figura D.5 B.



(a) Motor montado. (b) Vista superior.

Figura D.5: Colocación de ambos motores.

Una vez terminado el montaje de todos los elementos se debe revisar la estabilidad de estos elementos en caso de tener que realizar alguna otra modificación, sin embargo no se presentó tal caso.

Por lo que terminando este apartado tendremos la plataforma robótica como se muestra en la siguiente imagen:



Figura D.6: Chasis armado, motores y ruedas montados.

## D.4. Montaje de los elementos interiores

### *Materiales requeridos*

- Arduino MEGA 2560.
- Batería LiPo 14.8V, 2200 mAh.
- Raspberry Pi 3 Model B.
- Roboclaw.
- Bornes clemas Weidmuller, Regleta de clemas sencilla.
- Bandas de velcro, Cinchos.
- Regulador compacto de voltaje comutado de bajada D15V70F5S3.
- Tornillos y tuercas.

Para el desarrollo de este punto y basándonos en lo planteado anteriormente en la figura 3.22 se colocaron los elementos en el panel inferior el cual debió haber sido perforado antes del montaje de los motores como se mencionó previamente, sin embargo pueden existir algunas correcciones respecto a los componentes por lo que en caso de volver a perforar se debe tener cuidado con los cables de los motores que ya se encuentran montados en el panel inferior. Obteniendo el resultado que se muestra en la figura D.7.



Figura D.7: Elementos interiores colocados.

## D.5. Cableado eléctrico de los elementos interiores

### *Materiales requeridos*

- Cable puente o cables dupon macho-macho, macho-hembra y hembra-hembra.
- Conductor de cable flexible calibre 8, color rojo, color negro, color verde y color amarillo.
- Conector molex hembra de 2 vías P156.
- Conector 2 mm hembra de 4 vías.

Basándonos en los diagramas de conexiones mostrado en el anexo C el cableado de cada uno de los componentes se debe realizar teniendo en cuenta que el largo de los cables a usar para que en caso de que tenga que reemplazarse una terminal o bien un componente sea sencillo la manipulación y adecuación de los cables, además de que se recomienda estañar las puntas de los cables con soldadura para facilitar el agarre en algunas terminales de los componentes electrónicos. Además de tener cuidado al conectar los cables del panel lateral con las clemas montadas en el panel inferior. Obteniendo un resultado que se muestra en la figura D.8.

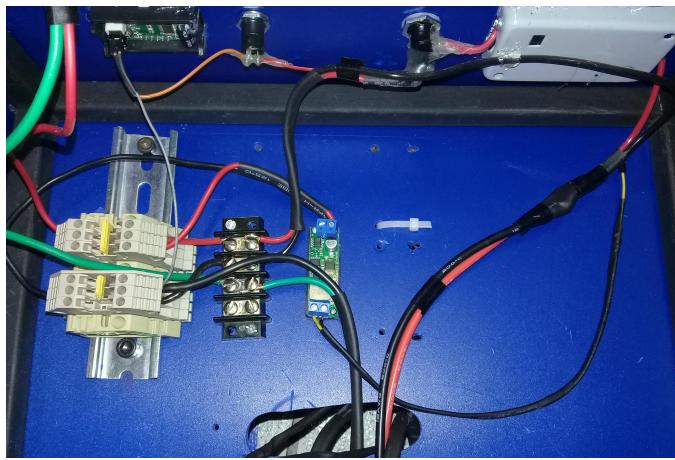


Figura D.8: Elementos interiores alambrados.

## D.6. Conexión de los componentes electrónicos

### *Materiales requeridos*

- Cable puente o cables dupon macho-macho, macho-hembra y hembra-hembra.
- Cable USB tipo A-B.
- Cable USB - mini USB.
- Header dupon hembra.
- Conductor de cable flexible calibre 8, color negro y color verde.
- Placa fenólica perforada 4.5 cm
- Sensor temperatura LM45, Texas Instrument.

Para comenzar con el cableado de los sensores de luz ambiental y el sensor de temperatura se decidió seguir el siguiente esquema de conexiones:

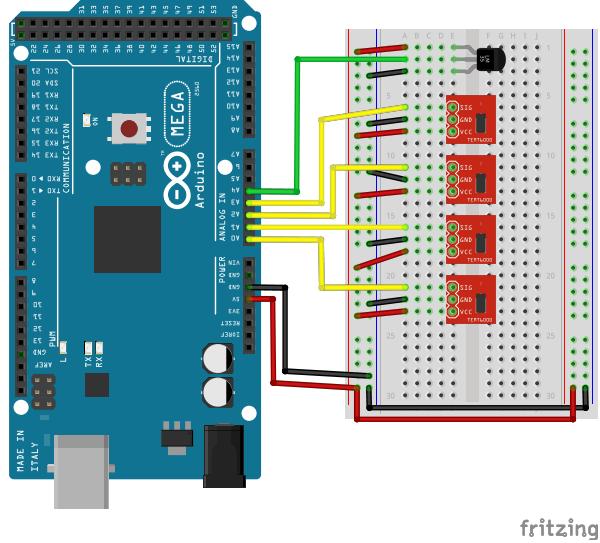


Figura D.9: Conexiones Arduino Mega.

Por lo que para buscar simplificar la distribución del voltaje de alimentación para cada uno de los componentes antes descritos se procedió a crear

un circuito sencillo en una placa fenólica perforada para evitar el uso de una protoboard dentro de la plataforma robótica móvil, si bien se pudo haber optado por la creación de una placa fenólica ya con todas las conexiones definidas podría resultar en una desventaja en caso de que se deseé añadir algunos componentes extras que requieran conexiones similares. La placa fenólica que se construyó se muestra en las siguientes imágenes:



Figura D.10: Placa fenólica construida.

Una vez finalizados estos preparativos se dispone a realizar las siguientes conexiones usando los cables jumper según sea el tipo que se requiere:

#### **Arduino Mega**

<i>Pin Arduino</i>	<i>Pin Placa</i>	<i>Descripción</i>
Vcc	Vcc Placa	Voltaje alimentación
GND	GND Placa	Tierra
A0	SL0	Entrada analógica
A1	SL1	Entrada analógica
A2	SL2	Entrada analógica
A3	SL3	Entrada analógica
A4	STempt	Entrada analógica

#### **Raspberry Pi 3**

<i>Pin Raspberry</i>	<i>Pin Regulador</i>	<i>Descripción</i>
Vcc(4)	Vcc Out	Voltaje alimentación
GND(6)	GND Out	Tierra

### Roboclaw

<i>Pin Roboclaw</i>	<i>Pin Motor</i>	<i>Descripción</i>
+	Vcc	Voltaje alimentación a 18 V
-	GND	Tierra de la batería
M1A	MotorDerA	Terminal de dirección A del Motor Derecho
M1B	MotorDerB	Terminal de dirección A del Motor Derecho
M2A	MotorIzqA	Terminal de dirección A del Motor Izquierdo
M2B	MotorIzqB	Terminal de dirección A del Motor Izquierdo
EN1	Encoders M Der	Terminales encoders del Motor Derecho
EN2	Encoders M Izq	Terminales encoders del Motor Izquierdo
+-	Vcc/GND	Alimentación de los encoders

Finalmente se conectan los dispositivos electrónicos: Arduino Mega y Roboclaw, a la tarjeta Raspberry Pi por medio de los puertos USB.

## D.7. Montaje de los elementos superiores

### *Materiales requeridos*

- Botón de paro de emergencia.
- Telémetro láser de escaneo RPLIDAR A1.

Para la colocación de estos elementos se buscó que el principal elemento de sujeción fuese la ranura del aluminio estructural combinado ya sea con los tornillos cabeza de martillo o un soporte de PVC espumado. Este apartado se deja a consideración del usuario debido al diseño que le sea más conveniente.

El montaje del botón de paro de emergencia se muestra en la figura D.10, hay que destacar que debe encontrarse ubicado en un lugar de fácil acceso para el usuario, y que no altere la toma de muestras del láser RPLIDAR, tal y como se muestra a continuación:



Figura D.11: Botón de paro de emergencia colocado.

Con respecto al montaje del láser RPLIDAR el soporte que se diseñe debe considerar el cable para el adaptador mini-usb que ofrece el fabricante, además de ubicarse en un espacio en el cual no se encuentren elementos propios del robot que dificulten las tomas de lecturas. Como se muestra en la figura a continuación:

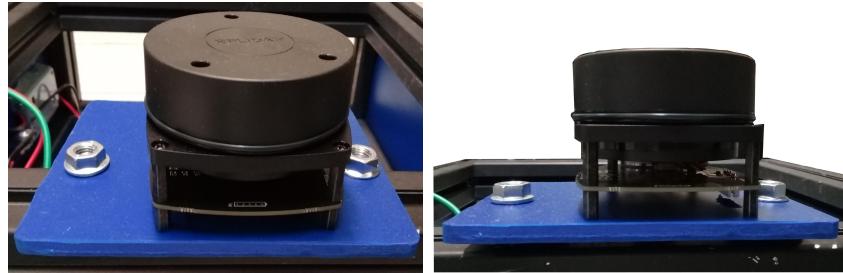


Figura D.12: Láser RPLIDAR colocado.

## D.8. Plataforma robótica móvil terminada

A continuación se presentan las fotografías del trabajo resultante del proceso de construcción descrito a lo largo de este apéndice.

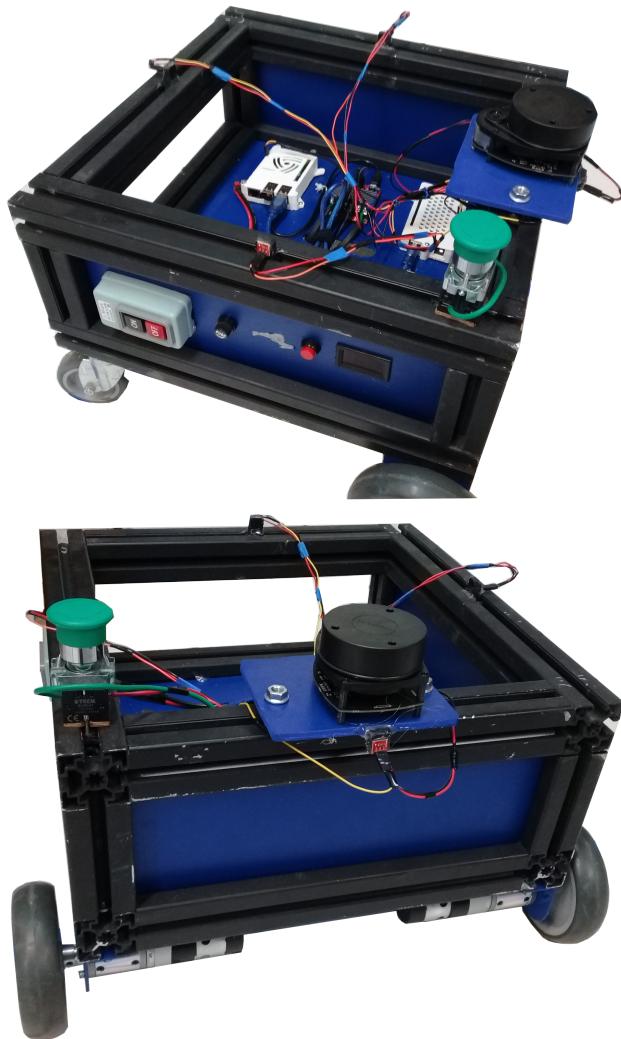


Figura D.13: Plataforma Robótica Móvil armada.



## Apéndice E

# Arquitectura completa del sistema

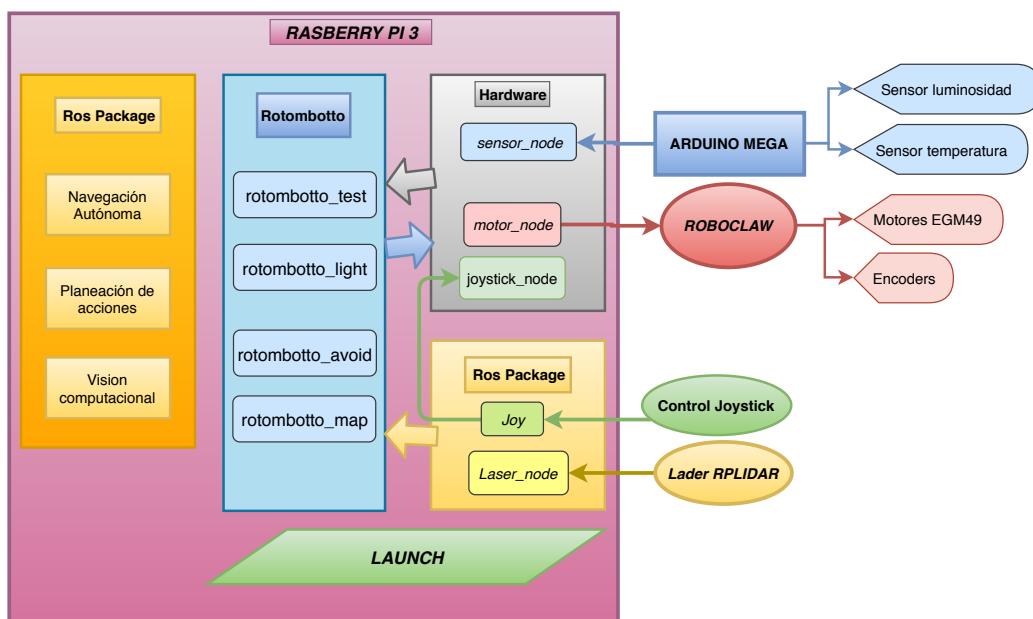


Figura E.1: Arquitectura completa del sistema implementado en la plataforma robótica móvil.



## Apéndice F

# Instalación del sistema de la plataforma robótica móvil

En el presente apéndice se describirá el proceso con el cual el usuario podrá descargar e instalar el sistema que se implementará en la plataforma robótica móvil descrita en el presente trabajo.

### F.1. Repositorio del proyecto

Buscando una óptima administración del sistema se decidió utilizar la plataforma de versiones Github para el almacenaje y distribución del proyecto del sistema desarrollado para la plataforma robótica móvil descrita en este trabajo. La liga de descarga para este repositorio es <https://github.com/G013/RotomBotto>.

Para su descarga por medio de terminal de comando se realizaría mediante el siguiente comando:

```
>$ git clone https://github.com/G013/RotomBotto.git
```

En caso de que el usuario no esté familiarizado con el uso de la plataforma Github y de la herramienta git se sugiere se consulte el siguiente sitio: <https://git-scm.com/book/es/v1/Empezando-Instalando-Git>.

## F.2. Contenido del repositorio del proyecto

Una vez descargado el repositorio se dispondrá del contenido organizado en ficheros como se describe a continuación:

- **Arduino\_files.** En este fichero se encuentra almacenado el archivo que se utilizarán para la programación del Arduino Mega utilizado en el robot de nombre *Arduino\_node.ino*, además de incluir un diagrama en PDF de conexión entre el Arduino Mega con los sensores disponibles en la plataforma robótica móvil.
- **catkin\_ws.** Fichero del workspace de ROS del sistema de la plataforma robótica móvil y donde se encuentra la mayor parte de la arquitectura del robot. Este fichero se encuentra instalado en la imagen del Sistema Operativo de la tarjeta Raspberry Pi usada en el proyecto que se describirá más adelante; o bien este fichero puede ser usado en un sistema operativo basado en Linux con soporte en ROS de preferencia por el usuario como se mencionó en el apartado 4.1.
- **Manufactura.** Este fichero almacena los archivos de las carcasa protectoras (archivos *.STL*) y los diagramas con los que se diseño el chasis de la plataforma robótica móvil (archivos *FCStd*) para que el usuario sea capaz de disponer de sus propios elementos cuando se requiera.
- **Media\_RotomBotto.** En este fichero se almacenará evidencia fotográfica y videográfica con el propósito de contar con contenido demostrativo del funcionamiento y desempeño de la plataforma robótica móvil.
- **RaspberryPi\_SD\_Image.** En este fichero se almacenará una copia de la imagen del sistema operativo utilizado para la implementación del software de la plataforma robótica móvil en la tarjeta Raspberry Pi 3 para una instalación más directa del proyecto desarrollado.
- **Readme.** Archivo el cual contiene una descripción general del proyecto como parte de la documentación de apoyo del repositorio.

## F.3. Instalación de paqueterías adicionales

En caso de que el usuario considere realizar manualmente la instalación de software necesario para el sistema de la plataforma robótica móvil se recomienda visitar alguna de las siguientes ligas a continuación:

- *Sistemas Operativos para Raspberry con soporte en ROS* <http://wiki.ros.org/ROSberryPi/Install>
- *Imagen SD del Ubuntu 16.04 LXDE.* <https://downloads.ubiquityrobotics.com/pi.html>

Para el uso de ciertos elementos del hardware de la plataforma robótica móvil que ROS ya tiene contemplado en algunas de sus paqueterías predefinidas, por lo cual se deben instalar para garantizar un correcto funcionamiento del sistema del robot, tal y como se mencionó en el capítulo 4 del presente trabajo.

### Paquete Joy

Para la instalación de este paquete de ROS se debe realizar mediante el comando:

```
>$ sudo apt-get install ros-indigo-joy
```

Una vez instalado se pueden realizar pruebas para revisar el correcto funcionamiento de la instalación y que el control joystick que se piense utilizar cuente con soporte en el software de la paquetería. Para comprobar el correcto funcionamiento del control joystick basta con conectarlo directamente a alguno de los puertos USB y teclear el siguiente comando en terminal:

```
>$ ls /dev/input/
```

Y buscar entre las opciones mostradas alguna cuyo nombre se asemeja a *jsX*.

Una vez comprobado que el control está siendo reconocido por el sistema operativo, se probará con el siguiente comando para revisar que las entradas del control están siendo captadas correctamente:

```
>$ sudo jstest /dev/input/jsX
```

Una vez que se comprobó que el control joystick funciona correctamente en el equipo se puede ejecutar el siguiente nodo en ROS para revisar el funcionamiento entre el control y ROS, mediante el siguiente comando:

- **Terminal 1:** Iniciando ROS y asignando el valor de  $X$  para que el sistema reconozca el control que se está usando.

```
>$ roscore
>$ rosparam set joy_node/dev "/dev/input/jsX"
```

- **Terminal 2** Ejecutando el nodo principal del paquete Joy

```
>$ rosrun joy joy_node}
```

- **Terminal 3:** Para revisar los datos publicados por el nodo por medio del tópico `/hardware/joy`.

```
>$ rostopic echo joy
```

*Usando como referencia (ROS.org, 2019c)*

#### Paquete `rosserial_arduino`

Para la instalación de este paquete de ROS se debe realizar mediante el comando:

```
>$ sudo apt-get install ros-indigo-rosserial-arduino
>$ sudo apt-get install ros-indigo-rosserial
```

Una vez terminada la instalación se procede a instalar el paquete en el workspace sobre el cual se está trabajando por medio de los siguientes comandos:

```
>$ cd catkin_ws/src
>$ git clone https://github.com/ros-drivers/rosserial.git
>$ cd catkin_ws
>$ catkin_make
```

Finalizada la instalación en el ambiente de trabajo de ROS se debe configurar el entorno de trabajo de Arduino para el uso de las librerías por medio de los siguientes comandos:

```
>$ cd <sketchbook>/libraries
>$ rm -rf ros_lib
>$ rosrun rosserial_arduino make_libraries.py .
```

Para revisar si el proceso se realizó correctamente se debe observar en el entorno de trabajo dentro de la pestaña ejemplos el apartado `ros_lib`.

*Usando como referencia (ROS.org, 2019b)*

### Paquete rplidar

Para el uso del láser RPLIDAR se debe descargar el software del paquete del siguiente repositorio: [https://github.com/robopeak/rplidar\\_ros](https://github.com/robopeak/rplidar_ros), una vez desempaquetado el contenido del repositorio bajo el nombre `rplidar_ros` se copia el fichero del paquete a la carpeta del proyecto `/RotomBot-to/catkin_ws/src/` y se compila el workspace del proyecto.

Una vez terminada la instalación del paquete, se procede a la revisión de los permisos del puerto serial una vez se haya conectado el láser RPLIDAR se deberán utilizar los siguientes comandos:

```
>$ ls -l /dev | grep ttyUSB
>$ sudo chmod 66 /dev/ttyUSB0
```

Una vez obtenidos los permisos para el uso del láser RPLIDAR, se pueden ejecutar los siguientes comandos para revisar el funcionamiento del láser por medio de rviz:

```
>$ roslaunch rplidar_ros view_rplidar.launch
#for rplidar A1/A2
$ >$ roslaunch rplidar_ros view_rplidar_a3.launch #for rplidar A3
```

Los servicios necesarios para manejar el motor del láser RPLIDAR son los siguientes:

```
>$ rosservice call /start_moto
>$ rosservice call /stop_motor
```

Para la revisión del tópico de la información obtenida por el láser RPLIDAR se utilizará el siguiente comando:

```
>$ rostopic echo /sensor_msgs/LaserScan
```

*Usando como referencia (ROS.org, 2019j)*

## F.4. Compilación del workspace del sistema

Una vez descargada la carpeta del repositorio se procederá a dirigirse al fichero del workspace del sistema por medio del siguiente comando:

```
>$ cd ~/Rotombotto/catkin_ws
```

En caso de que el usuario haya instalado la imagen del sistema operativo para la Raspberry encontrará una copia del fichero del repositorio con la que podrá ejecutar los comandos anteriores o bien el workspace del sistema se encontrará en la carpeta home de la misma.

Una vez ubicados en el fichero del workspace ejecutamos los siguientes comandos:

```
>$ catkin_make  
>$ source devel/setup.bash
```

Una vez finalizado la compilación del workspace el usuario podrá proceder a probar el sistema.

## F.5. Modo de prueba

Como se mencionó anteriormente en el capítulo 4, esta modalidad tiene como finalidad que el usuario pruebe que cada uno de los componentes disponibles en la plataforma robótica móvil disponibles funcionen adecuadamente, además de permitir que el usuario se familiarice con el entorno de trabajo de ROS. Esta modalidad utiliza algunas versiones modificadas de los nodos de los paquetes de hardware disponibles y los cuales constantemente están mostrando en pantalla la información recolectada y publicada por medio de los tópicos involucrados, con los cuales el usuario podrá revisar el funcionamiento del sistema y la interacción entre las paqueterías y sus nodos, además de que podrá detectar anomalías en caso de que algún elemento presente alguna falla. A continuación se describirán dos procedimientos con los que el usuario puede ejecutar la modalidad de prueba

### *Procedimiento Manual*

En este procedimiento se ejecutarán manualmente todos los nodos necesarios, lo cual será útil en caso de que se desee analizar solo una sección de los componentes, la ejecución se realiza por medio de los siguientes comandos:

- Terminal 1: Inicializar ROS

```
>$ roscore
```

- Terminal 2: Nodo JOY

```
>$ rosrun joy joy_node
```

- Terminal 3: Nodo Joystick

```
>$ rosrun joystick joystick_test.py
```

- Terminal 4: Configuración del puerto del Arduino

```
>$ rosrun rosserial_python serial_node.py /dev/ttyUSB0
```

- Terminal 5: Nodo Sensor

```
>$ rosrun sensor sensor_test
```

- Terminal 6: Nodo Motor

```
>$ rosrun motor motor_test.py
```

- Terminal 7: Nodo Laser

```
>$ rosrun rplidar_ros view_rplidar.launch
```

- Terminal 8: Nodo Rotombotto

```
>$ rorun rotombotto rotombotto_test
```

*Procedimiento por medio de archivo launch*

Con este procedimiento, como se describió en el capítulo 4, se pueden levantar todo el sistema por medio de un solo comando, como un breve ejemplo de describirá el código que respecta al archivo *rotombotto\_test.launch* que se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<launch>

<group ns = "ROS">
  <node pkg="rosserial_python" type="serial_node.py"
    name="serial_node">
    <param name="port" value="/dev/ttyACM0"/>
    <param name="baud" value="500000"/>
  </node>

  <include file="$(find rplidar_ros)/launch/rplidar.launch" />

  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find rplida
</group>

<group ns="hardware">
  <node name="joy_node" pkg="joy" type="joy_node"/>
  <node name="joystick_test" pkg="joystick" type="joystick_test.p
  <node name="sensor_test" pkg="sensor" type="sensor_test"
    output="screen"/>
  <node name="motor_test" pkg="motor" type="motor_test.py"
    output="screen"/>
</group>

<group ns="software">
  <node name="rotombotto_test" pkg="rotombotto" type="rotombotto_
    output="screen"/>
</group>

</launch>
```

Como se podrá ver en el archivo se declaran todos los paquetes y nodos que se requieren utilizar, separados en diferentes categorías a elección del usuario, además de realizar las configuraciones necesarias para el uso de ciertos componentes (configuración de permisos para el Arduino Mega). La ejecución de este procedimiento se realiza por medio del siguiente comando:

```
>$ rosrun rotombotto_begin rotombotto_test.launch
```

*Revisión de los tópicos*

En caso de que el usuario desee revisar la información transmitida por medio de los tópicos presente tras ejecutar alguno de los procedimientos deberá utilizar los siguientes comandos:

- Terminal 1: Joystick

```
>$ rostopic echo /hardware/joystick/data
```

- Terminal 2: Velocidades

```
>$ rostopic echo /hardware/motors/speeds
```

- Terminal 3: Arduino

```
>$ rostopic echo /hardware/arduino/data
```

- Terminal 4: Sensor luz

```
>$ rostopic echo /hardware/sensors/luz
```

- Terminal 5: Sensor temperatura

```
>$ rostopic echo /hardware/sensors/tempt
```

- Terminal 6: Laser

```
>$ rostopic echo
```

- Terminal 7

```
>$
```



# Referencias

- Association, O. S. H. A. O. S. H. (2019). *Definición.* urlhttps://www.oshwa.org/definition/spanish/. (Accedido 21-01-2019)
- BasicMicro. (2019). *Roboclaw 2x15a motor controller.* urlhttp://downloads.basicmicro.com/docs/roboclaw\_datasheet\_2x15A.pdf. (Accedido 6-02-2019)
- Electronics, R. (2019). *Emg49 gear motor with encoders.* urlhttp://www.robot-electronics.co.uk/emg49-gearmotor-with-encoder.html. (Accedido 21-01-2019)
- Foundation, R. P. (2019). *Raspberry pi 3 model b.* urlhttps://www.raspberrypi.org/products/raspberry-pi-3-model-b/. (Accedido 21-01-2019)
- HoobyKing. (2019). *Alarma de bajo voltaje para pilas lipo de 2s a 6s.* urlhttps://hobbyking.com/es\_es/hobbykingtm-lipoly-low-voltage-alarm-2s-6s. (Accedido 21-01-2019)
- Knudsen, J. B., y Loukides, M. (1999). *The unofficial guide to lego mindstorms robots.* O'Reilly.
- LEGO. (2019). *About ev3.* https://www.lego.com/en-us/mindstorms/about-ev3.
- M-P. (2019). *Raspberry pi 3 (b/b+), pi 2 b, and pi 1 b+ case with vesa mounts and more.* urlhttps://www.thingiverse.com/thing:922740. (Accedido 22-02-2019)
- multiple authors. (2019a). *Arduino home page.* urlhttps://www.arduino.cc. (Accedido 21-01-2019)
- multiple authors. (2019b). *Arduino mega 2560.* urlhttps://store.arduino.cc/arduino-mega-2560-rev3. (Accedido 21-01-2019)
- multiple authors. (2019c). *Getting started with the raspberry pi.*

- url`https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started.` (Accedido 21-01-2019)
- multiple authors. (2019d). *Raspberry fundation, about us.* url`https://www.raspberrypi.org/about/.` (Accedido 21-01-2019)
- multiple authors. (2019e). *Raspberry fundation home page.* url`https://www.raspberrypi.org/.` (Accedido 21-01-2019)
- multiple authors. (2019f). *What is arduino?* url`https://www.arduino.cc/en/Guide/Introduction.` (Accedido 21-01-2019)
- multiple authors. (2019g). *What is open source?* url`https://opensource.com/resources/what-open-source.` (Accedido 21-01-2019)
- Ollero Baturone, A. (2001). *Robótica: manipuladores y robots móviles.* MEXICO: ALFAOMEGA:.
- Platt, C. (2012). *Encyclopedia of electronic components volume 1: Resistors, capacitors, inductors, switches, encoders, relays, transistors* (Vol. 1). O'Reilly Media, Inc.
- Pololú. (2019). *Pololu step-down voltage regulator d15v70f5s3.* url`https://www.pololu.com/product/2111.` (Accedido 21-02-2019)
- PowerHD. (2019). *Batería lipo 14.8v, 2200mah 4s1p 35c.* url`https://sandorobotics.com/producto/hd2200xp-4s/.` (Accedido 24-02-2019)
- Ramírez, G. (2003). Método de aprendizaje simple para navegación de minirobots móviles rodantes. *Dyna,* 70(138).
- Robocup@home 2019: Rules and regulations (draft).* (2019). `http://www.robocupathome.org/rules/2019_rulebook.pdf.`
- Robots, A. M. (2011). *Peoplebot.* `https://www.generationrobots.com/media/PeopleBot-PPLB-RevA.pdf.`
- Rodriguez, W. (2019). *Arduino mega 2560 r3 enclosure.* url`https://www.thingiverse.com/thing:2397879.` (Accedido 22-02-2019)
- ROS.org. (2019a). *About ros.* url`http://www.ros.org/about-ros/.` (Accedido 21-01-2019)
- ROS.org. (2019b). *Arduino ide setup.* `http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup.`
- ROS.org. (2019c). *Configuring and using a linux-supported joystick with*

- ros.* url<http://wiki.ros.org/joy/Tutorials/ConfiguringALinuxJoystick>. (Accedido 21-03-2019)
- ROS.org. (2019d). *Core components.* url<http://www.ros.org/core-components/>. (Accedido 21-01-2019)
- ROS.org. (2019e). *joy.* url<http://wiki.ros.org/joy>. (Accedido 21-03-2019)
- ROS.org. (2019f). *roslaunch.* <http://wiki.ros.org/roslaunch>.
- ROS.org. (2019g). *rosserial.* <http://wiki.ros.org/rosserial>.
- ROS.org. (2019h). *rosserial\_arduino, tutorials, arduino ide setup.* [http://wiki.ros.org/rosserial\\_arduino/Tutorials/Arduino%20IDE%20Setup](http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup).
- ROS.org. (2019i). *rplidar.* <http://wiki.ros.org/rplidar>.
- ROS.org. (2019j). *rplidar.* <http://wiki.ros.org/rplidar>.
- Seaman, G. (1998). *How can hardware be ‘open’ ?* url[https://opencollector.org/Whyfree/open\\_hw.html](https://opencollector.org/Whyfree/open_hw.html). (Accedido 21-01-2019)
- SLAMTEC. (2019). *Rplidar-a1 360° laser range scanner.* url<http://www.slamtec.com/en/lidar/a1>. (Accedido 21-02-2019)
- Succar, L. E. S. (2018). *Robótica* (1.<sup>a</sup> ed.). Academia Mexicana de Computación, A.C.
- TexasInstruments. (2019). *Datasheet lm45.* url<http://www.ti.com/lit/ds/symlink/lm35.pdf>. (Accedido 21-01-2019)
- TurtleBot. (2019). *What is turtlebot?* url<https://www.turtlebot.com/about/>. (Accedido 21-02-2019)
- UBUNTU. (2019). *Lxde.* <https://help.ubuntu.com/community/LXDE>.
- VishaySemiconductors. (2019). *Vishay semiconductors, temt6000 ambient light sensor datasheet.* url<https://www.sparkfun.com/datasheets/Sensors/Imaging/TEMT6000.pdf>. (Accedido 30-01-2019)